Solaris 10 Container Guide

- Functionality status up to Solaris 10 10/09 and OpenSolaris 2009.06 -

Detlef Drewanz, Ulrich Gräf, et al.

Sun Microsystems GmbH
Effective: 30/11/2009

Functionalities

Use Cases

Best Practices

Cookbooks

Table of contents

Disclaimer

Sun Microsystems GmbH does not offer any guarantee regarding the completeness and accuracy of the information and examples contained in this document.

Revision control

Version	*Contents*	*Who*
3.1	30/11/2009 Adjustment with content of „Solaris Container Leitfaden 3.1“ Table of Content with HTML for better navigating through the document Correction „Patching of systems with local zones” Correction „Patching with zoneadm attach -u“ Correction Solaris Container Navigator	Detlef Drewanz Ulrich Gräf
3.0-en	27/11/2009 Review and corrections after translation Name doc to „Solaris 10 Container Guide - 3.0“	Detlef Drewanz Ulrich Gräf
3.0-en Draft 1	27/07/2009 Original English translation received	
3.0	19/06/2009 General corrections Additions in the General part, Resource Management Additions in patch management of zones Formatting: URLs and hyperlinks for better legibility Formatting: Numbering repaired Insertion: Possibilities for using ZFS in local zones	Detlef Drewanz, Uwe Furchheim, Ulrich Gräf, Franz Haberhauer, Joachim Knoke, Hartmut Streppel, Thomas Wagner, Heiko Stein
3.0 Draft 1	10/06/2009 More hyperlinks in the document General corrections Insertion: Solaris Container Navigator Incorporation: Functionalities Solaris 10 5/08 + 10/08 + 5/09 Insertion: Firewall between zones Revision: Zones and ZFS Revision: Storage concepts Revision: Network concepts Insertion: Dynamic configuration of devices in zones Revision: Resource management Addition: Patching zones Revision: Zones and high availability Insertion: Solaris Container Cluster Insertion: Solaris Container in OpenSolaris Insertion: Upgrade and patch management in a virtual operating environment	Dirk Augustin, Detlef Drewanz, Ulrich Gräf, Hartmut Streppel
2.1	13/02/2008 Incorporation of corrections (tagged vlanID)	Detlef Drewanz, Ulrich Gräf
2.0	21/01/2008 Incorporation of comments on 2.0-Draftv27 Insertion: Consolidation of log information Insertion: Cookbooks for resource management	Detlef Drewanz, Ulrich Gräf
2.0-Draftv27	11/01/2008 Incorporation of comments on 2.0-Draftv22 Incorporation Live Upgrade and zones Insertion of several hyperlinks Accelerated automatic installation of zones on a ZFS file system	Detlef Drewanz, Ulrich Gräf Bernd Finger, Ulrich Gräf
2.0-Draftv22	20/12/2007 Incorporation Resource Management Incorporation of Solaris 10 08/07 Incorporation of Solaris 10 11/06 Incorporation of comments Complete revision of the manual Incorporation "SAP Portals in Containers” Incorporation "Flying Zones” Revision "Zones and High Availability”	Detlef Drewanz, Ulrich Gräf Dirk Augustin Oliver Schlicker Detlef Ulherr, Thorsten Früauf
1.3	07/11/2006	

Version	***Contents***	***Who***
	Drawings 1 - 6 as an image	Detlef Drewanz
1.2	06/11/2006 General chapter virtualization Additional network examples	Detlef Drewanz, Ulrich Gräf
1.1	27/10/2006 Revision control table reorganized (the latest at the top) References amended Hardening of zones amended	Detlef Drewanz
1.0	24/10/2006 Feedback incorporated, corrections Use cases and network Zones added in the cluster Complete revision of various chapters	Detlef Drewanz Thorsten Früauf/Detlef Ulherr Ulrich Gräf
Draft 2.2	1st network example, corrections 31/07/2006	Ulrich Gräf
Draft 2.0	2nd draft published – 28/07/2006	Detlef Drewanz, Ulrich Gräf
Draft 1.0	1st draft (internal) – 28/06/2006	Detlef Drewanz, Ulrich Gräf

1. Introduction

[dd/ug] This guide is about Solaris Containers, how they work and how to use them. Although the original guide was developed in german [25], starting with version 3.1 we begin to deliver a version in english.

By making Solaris 10 available on 31st January 2005, an operating system with groundbreaking innovations has been provided by Sun Microsystems. Among these innovations are Solaris Containers that can be used - among other things - to consolidate and virtualize OS environments, to isolate applications, and for resource management. Solaris Containers can contribute considerably to the advanced reconfiguration of IT processes and environments, as well as to cost savings in IT operations.

Using these new possibilities requires know-how, decision guidance and examples which we have summarized in this guide. It is directed at decision makers, data center managers, IT groups and system administrators. The document is subdivided into the following chapters: Introduction, Functionality, Use Cases, Best Practices, Cookbooks, and a list of references.

A brief introduction is followed by the functionality part, which contains a description of today's typical data center requirements in terms of virtualization and consolidation, as well as a description and comparison of Solaris Container technology. This is followed by a discussion of the fields of application for Solaris Containers in a variety of use cases. Their conceptual implementation is demonstrated by means of Best Practices. In the chapter on Cookbooks, the commands used to implement Best Practices are demonstrated using concrete examples. All cookbooks were tested and verified by the authors themselves. The supplement discusses the specifics of Solaris Containers in OpenSolaris.

The document itself is designed to be a reference document. Although it is possible to read the manual from beginning to end, this is not mandatory. The manager of a data center gets an overview of the Solaris Container technology or have a look at the use cases. An IT architect goes over the Best Practices in order to build solutions. Meanwhile, a system administrator tests the commands listed in the cookbooks in order to gain experience. That is why the document offers something for everyone and in addition provides references to look into other areas.

Many thanks to all who have contributed to this document through comments, examples and additions. Special thanks goes to the colleagues (in alphabetical order): Dirk Augustin[da], Bernd Finger[bf], Constantin Gonzalez, Uwe Furchheim, Thorsten Früauf[tf], Franz Haberhauer, Claudia Hildebrandt, Kristan Klett, Joachim Knoke, Matthias Pfützner, Roland Rambau, Oliver Schlicker[os], Franz Stadler, Heiko Stein[hes], Hartmut Streppel[hs], Detlef Ulherr[du], Thomas Wagner and Holger Weihe.

Please do not hesitate to contact the authors with feedback and suggestions.

Berlin and Langen, November 2009

Detlef Drewanz (Detlef.Drewanz@sun.com), Ulrich Gräf (Ulrich.Graef@sun.com)

2. Functionality

2.1. Solaris Containers and Solaris Zones

2.1.1. Overview

[ug] Solaris Zones is the term for a virtualized execution environment – a virtualization at the operating system level (in contrast to HW virtualization).

Solaris Containers are Solaris Zones with Resource Management. The term is frequently used (in this document as well) as a synonym for Solaris Zones.

Resource Management has already been introduced with Solaris 9 and allows the definition of CPU, main memory and network resources.

Solaris Zones represent a virtualization at the interface between the operating system and the application.

- There is a global zone which is essentially the same as a Solaris operating system was in earlier versions
- In addition, local zones, also called nonglobal zones, can be defined as virtual execution environments.
- All local zones use the kernel of the global zone and are thus part of a single physical operating system installation – unlike HW virtualization, where several operating systems are started on virtualized hardware instances.
- All shared objects (programs, libraries, the kernel) are loaded only once; therefore, unlike for HW virtualization, additional consumption of main memory is very low.
- The file system of a local zone is separated from the global zone. It uses a subdirectory of the global zone's filesystem for a root directory (as in chroot environments).
- A zone can have one or several network addresses and network interfaces of its own.
- Physical devices are not visible in local zones (standard) but can optionally be configured.
- Local zones have their own OS settings, e.g. for name service.
- Local zones are separated from each other and from the global zone with respect to processes, that is, a local zone cannot see the processes of a different zone.
- The separation extends also to the shared memory segments and logical or physical network interfaces.
- Access to another local zone on the same computer is therefore possible through the network only.
- The global zone, however, can see all processes in the local zones for the purpose of control and monitoring (accounting).

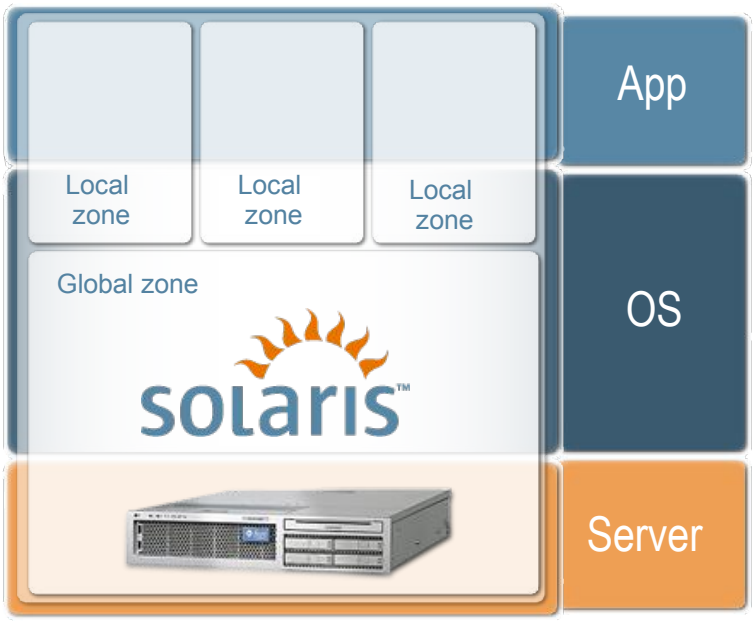


Figure 1: [dd] Schematic representation of zones

Thus, a local zone is a Solaris environment that is separated from other zones and can be used independently. At the same time, many hardware and operating system resources are shared with other local zones, which causes little additional runtime expenditure.

Local zones execute the same Solaris version as the global zone. Alternatively, virtual execution environments for older Solaris versions (SPARC: Solaris 8 and 9) or other operating systems (x86: Linux) can also be installed in so-called Branded Zones. In this case, the original environment is then executed on the Solaris 10 kernel; differences in the systemcalls are emulated.

Additional details are summarized in the following table:

Shared kernel:	The kernel is shared by the global zone and the local zones. The resources needed by the OS are needed only once. Costs for a local zone are therefore low, as measured by main memory, CPU consumption and disk space.
Shared objects:	In Unix, all objects such as programs, files and shared libraries are loaded only once as a shared memory segment which improves overall performance. For Solaris 10, this also includes zones; that is, no matter how frequently e.g. a program or a shared library is used in zones: in the main memory, it will occupy space only once. (other than in virtual machines.)
File system:	The visible portion of the file system of the local zone can be limited to one subtree or several subtrees of the global zone. The files in the local zone can be configured on the basis of directories shared with the global zone or as copies.
Patches:	For packages (Solaris packages) installed as copies in the local zone, patches can be installed separately as well. The patch level regarding non-Application patches should be the same, because all zones share the same kernel.
Network:	Zones have their own IP addresses on one or more virtual or physical interfaces. Network communication between zones takes place, if possible, via the shared network layers or when using exclusive IP-instances via external network connections.
Process:	Each local zone can see its own processes only. The global zone sees all processes of the local zones.
Separation:	Access to the resources of the global zone or other local zones, unless explicitly configured as such (devices, memory), is prevented. Any software errors that may occur are limited to their respective local zone by means of error isolation.
Assigned devices:	No physical devices are contained in the standard configuration of a local zone. It is, however, possible to assign devices (e.g. disks, volumes, DVD drives, etc.) to one or more local zones. Special drivers can be used this way as well.
Shared disk space:	In addition, further parts of the file tree (file systems or directories) can be assigned from the global zone to one or more local zones.
Physical devices:	Physical devices are administered from the global zone. Local zones do not have any access to the assignment of these devices.
Root delegation:	A local zone has an individual root account (zone administrator). Therefore, the administration of applications and services in a local zone can be delegated completely to other persons – including the root portion. Operating safety in the global zone or in other local zones is not affected by this. The global zone root has general access to all local zones.
Naming environment:	Local zones have an independent naming environment with host names, network services, users, roles and process environments. The name service of one zone can be configured from local files, and another zone from the same computer can use e.g. LDAP or NIS.
System settings:	Settings in `/etc/system` apply to the kernel used by all zones. However, the most important settings of earlier Solaris versions (shared memory, semaphores and message queues) can be modified from Solaris 10 onwards by the Solaris resource manager for each zone independently.

Table 1: [ug] Characteristics of Solaris 10 Zones

2.1.2. Zones and software installation

[dd] The respective requirements on local zones determine the manner in which software is installed in zones.

There are two ways of supplying software in zones:

1. Software is usually supplied in pkg format. If this software is installed in the global zone with *pkgadd*, it will be automatically available to all other local zones as well. This considerably simplifies the installation and maintenance of software since – even if many zones are installed – software maintenance can be performed centrally from the global zone.
2. Software can be installed exclusively for a local or for the global zone in order to e.g. be able to make software changes in one zone independent of other zones. This can be achieved by installation using special *pkgadd* options or by special types of software installations.

In any case the Solaris kernel and the drivers are shared by all zones but can be directly installed and modified in the global zone only.

2.1.3. Zones and security

[dd] By providing separate root directories for each zone, separate stipulations regarding security settings can be made by the local name service environments in the zones (RBAC – Role Based Access Control, passwd database). Furthermore, a separate passwd database with its own user accounts is provided in each zone. This makes it possible to build separate user environments for each zone as well as introducing separate administrator accounts for each zone.

Solaris 10 5/08, like earlier Solaris versions, is certified according to Common Criteria EAL4+. This certification was performed by the Canadian CCS. The Canadian CCS is a member of the group of certification authorities of Western states of which the Federal Office for Information Security (BSI, Bundesamt für Sicherheit in der Informationstechnik) is also a member. This certification is also recognized by BSI. A constituent component of the certification is protection against break-ins, separation and – new in Solaris 10 – zone differentiation. Details on this are available at:

http://www.sun.com/software/security/securitycert/

Solaris Trusted Extensions allow customers who are subject to specific laws or data protection requirements to use labeling features that have thus far only been contained in highly specialized operating systems and appliances. To implement labeled security, so-called compartments are used. For Solaris Trusted Extensions, these compartments are put into practice by Solaris zones.

2.1.4. Zones and privileges

[dd] Local zones have fewer process privileges than the global zone whereby some commands cannot be executed within a local zone. For standard configurations of zones, this access is permitted only in the global zone. The restrictions include, among other things:

- Configuration of swap space and processor sets
- Modifications to the process scheduler and the shared memory
- Setting up device files
- Downloading and uploading kernel modules
- For shared IP authorities:
  - Access to the physical network interface
  - Setting up IP addresses

Since Solaris 10 11/06, local zones can have additional process privileges assigned to them when zones are being configured that allow extended possibilities for local zones (but not all).

Potential combinations and usable privileges in zones are shown here:

http://docs.sun.com/app/docs/doc/817-1592/6mhahuotq?a=view

2.1.5. Zones and resource management

[ug] In Solaris 9, resource management was introduced on the basis of projects, tasks and resource pools. In Solaris 10, resource management can be applied to zones as well. The following resources can be managed:

- CPU resources (processor sets, CPU capping and fair share scheduler)
- Memory use (real memory, virtual memory, shared segments)
- Monitoring network traffic (IPQoS = IP Quality of Service)
- Zone-specific settings for shared memory, semaphore, swap (System V IPC Resource Controls)

2.1.5.1. CPU resources

[ug] Three stages of resource managements can be used for zones:

- Partitioning of CPUs in processor sets that can be assigned to resource pools. Resource pools are then assigned to local zones, thus defining the usable CPU quantity.
- Using the fair share scheduler (FSS) in a resource pool that is used by one or more local zones. This allows fine granular allocation of CPU resources to zones in a defined ratio as soon as zones compete for CPU time. This is the case if system capacity is at 100%. Thus, the FSS ensures the response time for zones, if configured accordingly.
- Using the FSS in a local zone. This allows fine granular allocation of CPU resources to projects (groups of processes) in a defined ratio if projects compete for CPU. This takes place, when the capacity of the CPU time available for this zone is at 100%. Thus, the FSS ensures the process response time.

Processor sets in a resource pool

Just like a project, a local zone can have a resource pool assigned to it where all zone processes proceed (`zonecfg: set pool=`). CPUs can be assigned to a resource pool. Zone processes will then run only on the CPUs assigned to the resource pool. Several zones (or even projects) can also be assigned to a resource pool which will then share the CPU resources of the resource pool.

The most frequent case is to create a separate resource pool with CPUs per zone. To simplify matters, the number of CPUs for a zone can then be configured in the zone configuration (`zonecfg: add dedicated-cpu`). When starting up a zone, a temporary resource pool is then generated automatically that contains the configured number of CPUs. When the zone is shut down, the resource pools and CPUs are released again (since Solaris 10 8/07).

Fair share scheduler in a resource pool

In the event that several zones run together in a resource pool, the fair share scheduler (FSS) allows the allocation of CPU resources within a resource pool to be managed. To this end, each zone or each project can have a share assigned to it. The settings for zones and projects in a resource pool are used to manage the CPU resources in the event that the local zones or projects compete for CPU time:

- If the workload of the processor set is less than 100%, no management is done since free CPU capacity is still available.
- If the workload is at 100%, the fair share scheduler is activated and modifies the priority of the participating processes such that the assigned CPU capacity of a zone or a project corresponds to the *defined share*.
- The *defined share* is calculated from the share value of an active zone/project) divided by the sum of the shares of all active zones/projects.

The allocation can be changed dynamically while running.

CPU resource management within a zone

In a local zone it is furthermore possible to define projects and resource pools and to apply CPU resources via FSS to projects running in the zone (see previous paragraph).

CPU capping

The maximum CPU usage of zones can be set (`cpu-caps`). This setting is an absolute limit with regard to the CPU capacity used and can be adjusted to 1/100 CPU exactly (starting with Solaris 10 5/08). With this configuration option, the allocation can be adjusted much more finely than with processor sets (1/100 CPU instead of 1 CPU).

Furthermore, CPU capping offers another control option if several zones run within one resource pool (with or without FSS) in order to limit all users to the capacity that will be available later on.

2.1.5.2. Memory resource management

[ug] In Solaris 10 (in an update of Solaris 9 as well), main memory consumption can be limited at the level of zones, projects and processes. This is implemented with the so-called resource capping daemon (*rcapd*).

A limit for physical memory consumption is defined for the respective objects. If consumption of one of the projects exceeds the defined limit, the *rcapd* causes little-used main memory pages of processes to be paged out. The sum of the space consumption of the processes is used as a measurement parameter.

In this manner, the defined main memory requirement is complied with. The capacity of processes in the corresponding object drops as they may need to swap pages back in again if necessary if the memory areas are used again. Continuous swapping of main memory pages is an indication that the available main memory is too low or that the settings are too tight or the application currently needs more than the negotiated amount of memory.

For simplification, starting with Solaris 10 8/07, memory limits can be set for each zone. The amount of physical memory used (*physical*), the virtual memory (*swap*) and `locked` segments (main memory pages that cannot be swapped, shared memory segments) can be limited. The settings for virtual and locked memory are hard limits, that is, once the corresponding value has been reached, a request of an application for more memory is denied. The limit for physical memory, however, is monitored by the `rcapd`, which successively swaps main memory pages if the limit is exceeded.

2.1.5.3. Network resource management (IPQoS = IP Quality of Service)

[ug] In Solaris 10 (also Solaris 9) it is possible to classify network traffic and to manage the data rate of the classes. One example is giving preference to a web server's network traffic over network backup. Service to the customer should not suffer while network backup is running.

Configuration is done using rules in a file that is activated with the command *ipqosconf*. The rules consist of a part that allows the user to classify network traffic, and actions to manage the data rate/burst rate. Classification can take place among other things according to some or all of the following parameters:

- Address of sender or recipient
- Port number
- Data traffic type (UDP, TCP)
- Userid of the local process
- Project of the local process (*/etc/project*)
- IP traffic TOS field (change priority in an ongoing connection)

2.1.6. User interfaces for zones

[dd] A variety of tools are available for working with zones and containers. Solaris itself provides a series of command line interface (CLI) tools such as *zoneadm*, *zonecfg* and *zlogin* that allow you to undertake the configuration and installation of zones on the command line or in scripts.

The Solaris Container Manager is available as a graphical user interface (GUI) (http://www.sun.com/software/products/container_mgr/). This is a separate product operated together with the Sun Management Center (SunMC). The Container Manager allows simple and rapid creation, reconfiguration or migration of zones through its user interface, and the effective use of resource management.

With its zone module, Webmin supplies a browser user interface (BUI) for the installation and management of zones. The module can be downloaded from http://www.webmin.com/standard.html as *zones.wbm.gz*.

2.1.7. Zones and high availability

[tf/du/hs] In the presence of all RAS capabilities, a zone has only the availability of a computer and it decreases with the number of components of the machine (MTBF).

If this availability is not sufficient, so-called failover zones can be implemented using the HA Solaris Container Agent, allowing zones to be panned among cluster nodes (from Sun Cluster 3.1 08/05). This increases the availability of the total system considerably. In addition, a container here becomes a flexible container. That is to say, it is completely irrelevant which of the computers participating in the cluster the container is running on. Relocating the container can be done manually by administrative actions or automatically in the event of a computer malfunction.

Alternatively, it is also possible using the HA Solaris Container Agent to start and to stop local zones including their services. That is, identical zones exist on several computers with identical services that have no(!) shared storage. The failover of a zone in such a configuration is not possible, because the zone rootpath cannot be moved between the zones. Instead one of the zones can be stopped and an identical zone can then be started on another system.

Container clusters (since Sun Cluster 3.2 1/09) offer a third option. A Sun Cluster, where virtual clusters can be configured, is installed in the global zone. The virtual clusters consist of virtual computer nodes that are local zones. Administration can be transferred to the zone operators. (see 2.1.9 Solaris container cluster (aka "zone cluster")).

2.1.8. Branded zones (Linux and Solaris 8/Solaris 9 compatibility)

[dd/ug] Branded zones allow you to run an OS environment which is different from the one that is installed in the global zone (BrandZ, since Solaris 10 8/07)

Branded zones extend zone configuration as follows:

- A brand is a zone attribute.
- Each brand contains mechanisms for the installation of a branded zone.
- Each brand can use its own pre-/post-boot procedures.

At runtime, system calls are intercepted and either forwarded to Solaris or emulated. Emulation is done by modules in libraries contained in the brand. This design has the advantage that additional brands can easily be introduced without changing the Solaris kernel.

The following brands are currently available:

- `native`: Brand for zones with the Solaris 10 version from the global zone
- `lx`: Solaris Container for Linux Applications (abbreviated: SCLA)
  With this, unmodified 32-bit Linux programs can be used under Solaris on x86 systems. To do so, a 32-bit Linux distribution that can use a Linux 2.4 kernel must be installed in the zone. The required Linux programs can then be used and/or installed in the zone.
  The Linux distribution and license are themselves not contained in Solaris and must be installed in each zone (new installations or copies of an existing installation). At least the following Linux distributions work:
  - CentOS 3.5 – 3.8, RedHat Linux 3.5 – 3.8 (certified)
  - Debian (unsupported, for instructions see blog by Nils Nieuwejaar http://blogs.sun.com/nilsn/entry/installing_a_debian_zone_with)

- `solaris8/Solaris9`: Solaris 8 or Solaris 9 container
  This allows you to operate a Solaris 8 or Solaris 9 environment in a zone (SPARC only). Such containers can be made highly available with the HA Solaris Container Agent. Such types of zones cannot be used as virtual nodes in a virtual cluster.

- `solaris10`: Solaris 10 branded zones for OpenSolaris are in the planning stage
  (see http://opensolaris.org/os/project/s10brand/)

2.1.9. Solaris container cluster (aka "zone cluster")

[hs] In autumn 2008, within the scope of the Open HA Cluster Project, *zone clusters* were announced. The latter has also been available since Sun Cluster 3.2 1/09 in a commercial product as *Solaris Container Cluster*. A Solaris Container Cluster is the further development of the Solaris zone technology up to a virtual cluster, also called "zone cluster". The installation and configuration of a container cluster is described in the Sun Cluster Installation Guide [http://docs.sun.com/app/docs/doc/820-4677/ggzen?a=view].

The Open HA Cluster provides a complete, virtual cluster environment. Zones as virtualized Solaris authorities are used as elements. The administrator of such an environment can see and notice almost no difference to the global cluster that is virtualized here.

Two principal reasons have advanced the development of virtual cluster technology:

- the desire to round off container technology
- the customer requirement to be able to run Oracle RAC (Real Application Cluster) within a container environment.

Solaris containers, operated by Sun Cluster, offer an excellent possibility to operate applications safely and with high availability in Solaris containers. Two options, Flying Containers and Flying Services, are available for implementation.

A clean separation in system administration allows container administration to be delegated to the application operators who install, configure and operate their application within a container. It was indeed unsatisfactory for an application operator to have administrator privileges in his containers but, on the other hand, to be limited in handling the cluster resources belonging to his zone.

In the cluster implementation prior to Sun Cluster 3.2 1/09, the cluster consisted mainly of the components installed, configured and also running in the global zone. Only a very small number of cluster components were actually active within a container, and even this allowed very limited administrative intervention only.

Now, virtual clusters make the zone administrator feel that he has almost complete control of his cluster. Restrictions apply to services that continue to exist only once in the cluster, such as e.g. quorum devices or even heartbeats.

Oracle RAC users could not understand that this product could not simply be installed and operated within a container. One has to know, however, that Oracle CRS, the so-called Oracle Clusterware – an operating system independent cluster layer – requires rights that the original safety concept of Solaris containers did not delegate to a non-global zone. Since Solaris 10 5/08 it is, however, possible to administer network interfaces even within a zone such that Oracle CRS can be operated there.

The goal of providing a cluster environment that does not require any adjustments of applications whatsoever has been achieved. Even Oracle RAC can be installed and configured just like in a normal Solaris instance.

Certification of Oracle RAC in a Solaris Container Cluster is currently (June 2009) not yet finalized. However, Sun Microsystems offers support for such an architecture.

It is also possible to install Oracle RAC with CRS in a Solaris container without a zone cluster but it will not yet be certified. The disadvantage of such a configuration consists in the fact that solely exclusive-IP configurations can be used, which unnecessarily increases the number of required network interfaces (if not using VLAN interfaces).

2.2. Virtualization technologies compared

[ug] Conventional data center technologies include

- Applications on separate computers
  This also includes multi-tier architectures with firewall, load balancing, web and application servers and databases.
- Applications on a network of computers
  This includes distributed applications and job systems.
- Many applications on a large computer

The separation of applications on computers simplifies the installation of the applications but increases administrative costs since the operating systems must be installed several times and maintained separately. Furthermore, computers are usually underutilized (< 30%).

Distributed applications are applications running simultaneously on several computers that communicate (MPP computers, MPI software, etc.) via a network (TCP/IP, Infiniband, Myrinet, etc.). For job systems, the computation is broken up into self-contained sub-jobs with dependencies and is carried out on several computers by a job scheduler who also undertakes data transport (grid system).

Both alternatives require high network capacity and appropriate applications and make sense only in areas where the applications are already adapted to this type of computing. Modifying of an application is a major step and is rarely performed for today's standard applications. But this technology can become more interesting in the future with new applications.

The manner in which mainframes and larger Unix systems are operated today is to run many applications in one computer. The advantages are that the systems have a better workload (several applications) and a lower number of operating system installations to be serviced. It is therefore exactly this variant which is of interest for consolidation in the data center.

The challenges consist in creating an environment for the applications where the latter can run independently (separation) while still sharing resources with each other to save costs. Particularly interesting areas are:

- Separation. How far separated are the environments of the applications?
- Application. How does the application fit into the environment?
- Effects on software maintenance
- Effects on hardware maintenance
- Delegation: Can administrative tasks be delegated to the environment?
- Scaling of the environment?
- Overhead of the virtualization technology?
- Can different OS versions be used in the environments?

A variety of virtualization techniques were developed for this purpose and are presented below.

For comparison, see also: http://en.wikipedia.org/wiki/Virtualization

2.2.1. Domains/physical partitions

[ug] A computer can be partitioned by configuration into sub-computers (domain, partition). Domains are almost completely physically separated since electrical connections are turned off. Shared parts are either very failsafe (cabinet) or redundantly structured (service processor, power supplies).

Advantages:

- Separation: Applications are well separated from each other; mutual influence via the OS or failed shared hardware is not possible.
- Application: All applications are executable as long as they are executable in the basic operating system.
- Scalability: The capacity of a virtualization instance (here: a domain) can be changed for some implementations while running (dynamic reconfiguration) by relocating hardware resources between domains.
- HW maintenance: If one component fails and the domain is constructed appropriately, the application can still run. Dynamic reconfiguration allows repairs to be performed while running (in redundant setups). A cluster must be set up only to intercept total failures (power supply, building on fire, data center failure, software errors).
- OS versions: The partitions are able to run different operating systems/versions.

Disadvantages:

- OS maintenance: Each machine has to be administered separately. OS installation, patches and the implementation of in-house standards must be done separately for each machine.
- Delegation: The department responsible for the application/service requires root privileges, or must communicate with computer operations regarding modifications. All aspects of the operating system can be administered in the physical partition. This can affect security and can become costly/time-consuming.
- Overhead: Each machine has a separate operating system overhead.

Sun offers domains in the high-end servers SunFire E20K, E25K, the mid-range servers SunFire E2900, E4900, E6900 and the Sun SPARC Enterprise M4000, M5000, M8000 and M9000.

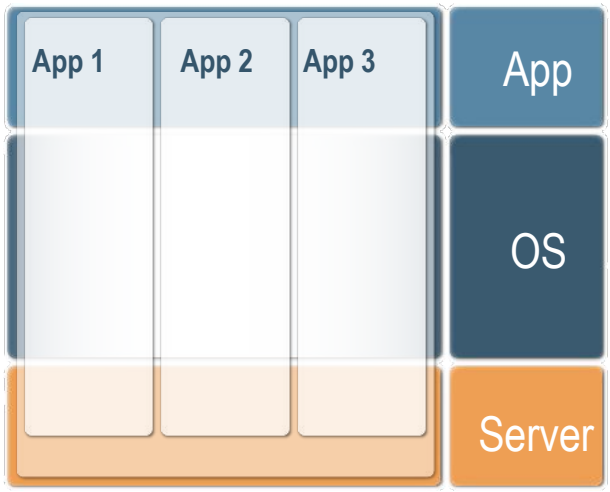


Figure 2: [dd] Domains/Physical domains

This virtualization technology is provided by several manufacturers (Sun Dynamic System Domains, Fujitsu-Siemens Partitions, HP nPars). HW support and (a little) OS support are required.

2.2.2. Logical partitions

[ug] A minimal operating system called the hypervisor, that virtualizes the interface between the hardware and the OS of a computer, runs on the computer's hardware. A separate operating system (guest operating system) can be installed on the arising so-called virtual machines.

In some implementations, the hypervisor runs as a normal application program; this involves increased overhead.

Virtual devices are usually created from real devices by emulation; real and virtual devices are assigned to the logical partitions by configuration.

Advantages:

- Application: All applications of the guest operating system are executable.
- Scalability: The capacity of a logical partition can be modified in some cases while running, when the OS and the hypervisor support this.
- Separation: Applications are separated from each other; direct mutual influence via the OS is not possible.
- OS versions: The partitions are able to run different operating systems/versions.

Disadvantages:

- HW maintenance: If a shared component fails, many or all logical partitions may be affected. An attempt is made, however, to recognize symptoms of future failure by preventive analysis, in order to segregate errors in advance.
- Separation: The applications can influence each other via shared hardware. One example for this is the virtual network since the hypervisor has to emulate a switch. Virtual disks which are located together on a real disk are “pulling away” the disk head from each other are another example of this behavior. To prevent this from happening, real network interfaces or dedicated disks can be used which, however, increases the cost for using logical partitions.
- OS maintenance: Each partition has to be administered separately. OS installation, patches and the implementation of in-house standards must be done separately for each partition.
- Delegation: If the department responsible for the application/service requires root privileges, or must communicate with computer operations regarding modifications. All aspects of the operating system can be administered in the logical partition. This can affect security and can become costly/time-consuming.
- Overhead: Each logical partition has its own operating system overhead; in particular the main memory requirements of the individual systems are maintained.

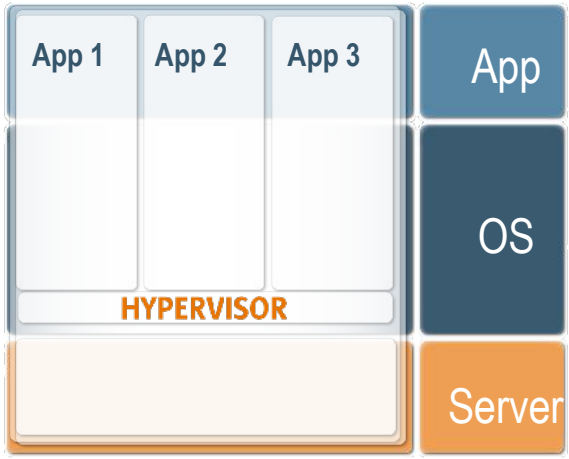


Figure 3: [dd] Logical partitions

Logical partitioning systems include the IBM VM operating system, IBM LPARs on z/OS and AIX, HP vPars, as well as VMware and XEN. Sun offers Logical Domains (SPARC: since Solaris 10 11/06) as well as Sun xVM VirtualBox (x86 and x64 architectures).
The Sun xVM server is in collaboration with the XEN community for x64 architectures in development. The virtualization component (xVM Hypervisor) can already be used since OpenSolaris 2009.06.

2.2.3. Containers (Solaris zones) in an OS

[ug] In an operating system installation, execution environments for applications and services are created that are independent of each other. The kernel becomes multitenant enabled: it exists only once but appears in each zone as though it was assigned exclusively.

Separation is implemented by restricting access to resources, such as e.g. the visibility of processes (modified procfs), the usability of the devices (modified devfs) and the visibility of the file tree (as with chroot).

Advantages:

- Application: All applications are executable unless they use their own drivers or other system-oriented features. Separate drivers can, however, be used via installations in the global zone.
- Scalability: Container capacity can be configured (through resource management, processor sets and CPU caps).
- Separation: Applications are separated from each other; direct mutual influence via the OS is not possible.
- OS maintenance: OS installation, patches and implementation of in-house standards must take place in a central location (in the global zone) only.
- Delegation: The department responsible for the application/ service requires root privileges for part of the administration. Here, it can obtain the root privileges within the zone without being in a position to affect other local zones or the global zone. The right to allocate resources is reserved to the global zone only.
- Overhead: All local zone processes are merely normal application processes from the point of view of the global zone. The OS overhead (memory management, scheduling, kernel) and memory requirements for shared objects (files, programs, libraries) are created only once. Each zone has only a small additional number of system processes. For that reason, it is possible to have hundreds of zones on a single-processor system.

Disadvantages:

- HW maintenance: If a shared component fails, many or all zones may be affected. Solaris 10 recognizes symptoms of a future failure through FMA (Fault Management Architecture) and can deactivate the affected components (CPU, memory, bus systems) while running, or instead use alternative components that are available. Through the use of cluster software (Sun Cluster), the availability of the application in the zone can be improved (Solaris Container Cluster/ Solaris Container Agent).
- Separation: The applications can influence each other through shared hardware. That influence can be minimized in Solaris with resource management and network bandwidth management.
- OS versions: Different operating systems/versions are possible with branded zones only. Here, a virtual process environment for another operating system is created in one zone but the kernel of the global zone is used by the branded zones as well.

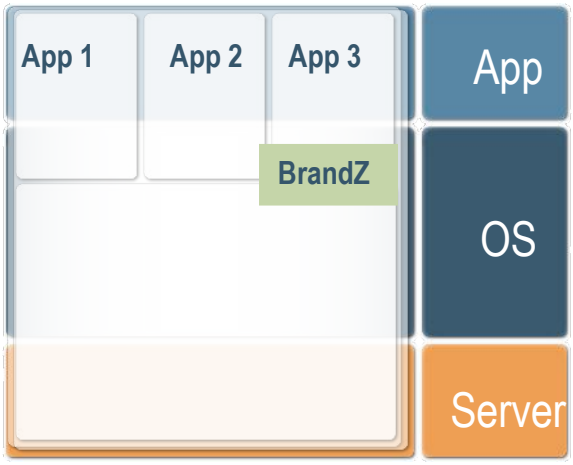


Figure 4: [dd] Container (Solaris zones) in an OS

Implementations in the BSD operating system are Jails, in Solaris: zones, and in Linux the vserver project. HW requirements are not necessary.

2.2.4. Consolidation in one computer

[ug] The applications are installed on a computer and used under different userid. This is the type of consolidation feasible with modern operating systems.

Advantages:

- Application: All applications are executable as long as they are executable in the basic operating system and do not use their own OS drivers. However, there are restrictions if different versions of the application with a defined install directory are required, or if two instances require the same userid (e.g. Oracle instances), or if the configuration files are located in the same locations in the file system.
- Scalability: The capacity of an application can be modified online.
- OS maintenance: OS installation, patches and implementation of in-house standards must take place for one OS only. That is to say, many applications can be run with the administrative effort for one machine only.
- Overhead: Overhead is low since only the application processes must run for each application.

Disadvantages:

- HW maintenance: If a shared component fails, many or all applications may be affected.
- OS maintenance: Administration becomes complicated as soon as applications are based on different versions of a software (e.g. versions for Oracle, Weblogic, Java, etc.). Such a system becomes difficult to control without accurate documentation and change management. Any error in the documentation that is not noticed immediately can have fatal consequences in an upgrade (HW or OS) later on.
- Separation: The applications can influence each other through shared hardware and the OS. In Solaris, the influence can be reduced by using resource management and network bandwidth management.
- Delegation: The department responsible for the application/service requires root privileges for a portion of the job control or must communicate with computer operations regarding modifications. This can therefore affect security or become more costly/time-consuming.
- OS versions: Different operating systems/versions are not possible.

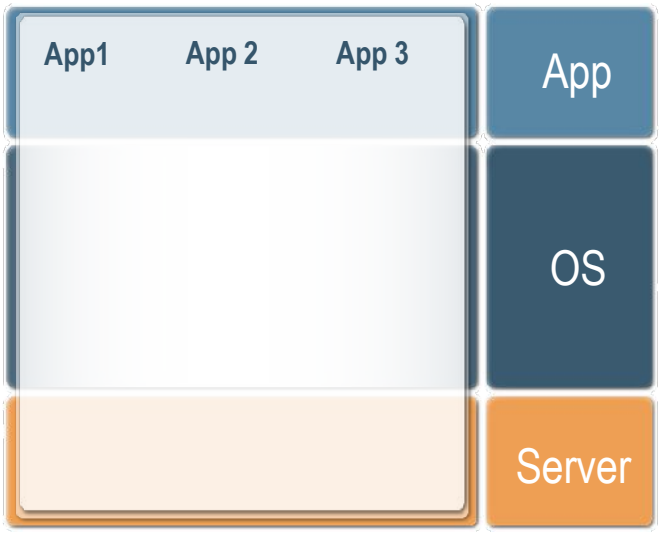


Figure 5: [dd] Consolidation in one computer

Many modern operating systems facilitate this type of consolidation, which allows several software packages to be installed, if necessary, even different versions of the same software.

2.2.5. Summary of virtualization technologies

[ug] The virtualization technologies discussed above can be summarized in the following table – compared to installation on a separate computer.

	Separate computer	*Domains/ Physical partitions*	*Logical partitions*	*Containers (Solaris zones) in an OS*	*Consolidation in one computer*
Separation	+	*O*	*+(Software)* *- (Hardware)*	*+ (Software)* *- (Hardware)*	-
Application	+	+	+	+	-
SW maintenance	-	-	-	+	*O*
HW maintenance	+	+	*O*	*O*	*O*
Delegation	-	-	-	+	-
Scalability	*O*	+	*O*	+	+
Overhead	-	-	-	+	+
OS versions	*one each*	*several*	*several*	*one*	*one*

Table 2: [ug] Summary of virtualization technologies

Key for the meaning of the symbols:

- + good
- O is neither good nor bad
- - means: has disadvantages

While separation is of course best in a stand-alone computer. Physical partitions use the same cabinet with shared power supplies (fire can harm each partition), although the partitions are independent. Lpars, LDom and containers are separated in the OS environment only.

Consolidation in an operating system reveals all that is visible in the applications to each application. In all other cases, applications are separated.

Unified SW maintenance can only be performed with zones and consolidation in one computer. The other technologies require multiple maintenance.

HW maintenance on the machine of the application is practicable only for domains or separate computers without affecting other applications, unless mobile Lpars/LDom or cluster technologies with flying zones are used.

The delegation of portions of administrative tasks is possible for containers only. All other technologies require that the tasks be defined exactly and that dedicated roles are assigned. This is costly and time-consuming.

Scalability for separate computers and Lpars is limited based on the hardware (defined performance of the shared interconnect) while domain capacity can be customized by additional hardware. Containers and consolidation on one computer run without adaption of the application on bigger computers. Additional reserves can be used by relocating additional containers to this computer.

Overhead is higher for separate computers and physical and logical partitioning because one operating system with CPU and memory requirements runs per application. Containers and consolidation on one computer share one operating system and are therefore considerably more economical with regard to their consumption of resources. The lower the resource requirements of an application, the more pronounced the effect.

The OS version of each operating system installation must be maintained separately. This means overhead. Therefore, for separate computers as well as physical and logical virtualization, more effort must be expended than for containers or consolidation on one computer. However, multiple OS versions enable the use of different versions of the OS if required by the applications. The assessment in this regard depends on data center policies and operation purposes.

The overhead of several OS instances can be reduced by using management software such as e.g. Sun xVM OpsCenter, which requires a certain investment.

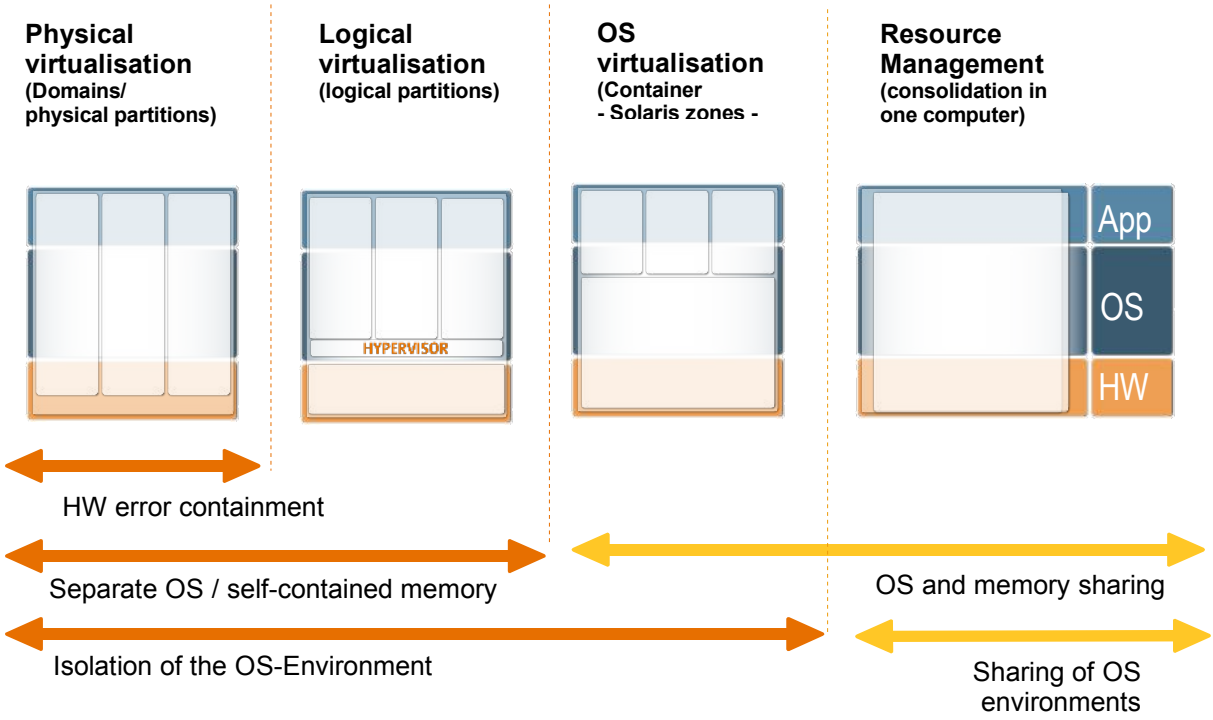


Figure 6: [dd] Comparison of virtualization technologies

3. Use Cases

The following chapter discusses a variety of use cases for Solaris Containers and evaluates them.

3.1. Grid computing with isolation

Requirement

[ug] There is a need within a company to use free cycles on existing computers to perform a certain amount of computing work in the background.

Grid software such as Sun GridEngine is suitable for this. However, the processes then running in the grid should be blocked from seeing other processes in the system.

Solution

[ug] Sun GridEngine is installed in one zone on computers that have spare capacity:

- Sparse-root zones are used (4.1.1 Sparse-root zones).
- Software and data are installed in the global zone per NFS.
- Sun GridEngine is installed in the local zone.
- Automatic creation of the zone (5.1.14 Automatic quick installation of zones).
- Software installation per lofs from a global zone filesystem

Assessment

[ug] This use case has the following characteristics:

- Computers can use spare capacities at defined times.
- Other application data remain protected from inspection. This frequently increases the willingness of the persons responsible for the application to make the computers available for this purpose.
- The applications department can "sell" the spare capacity to grid users.

Above all, computing capacities are economized.

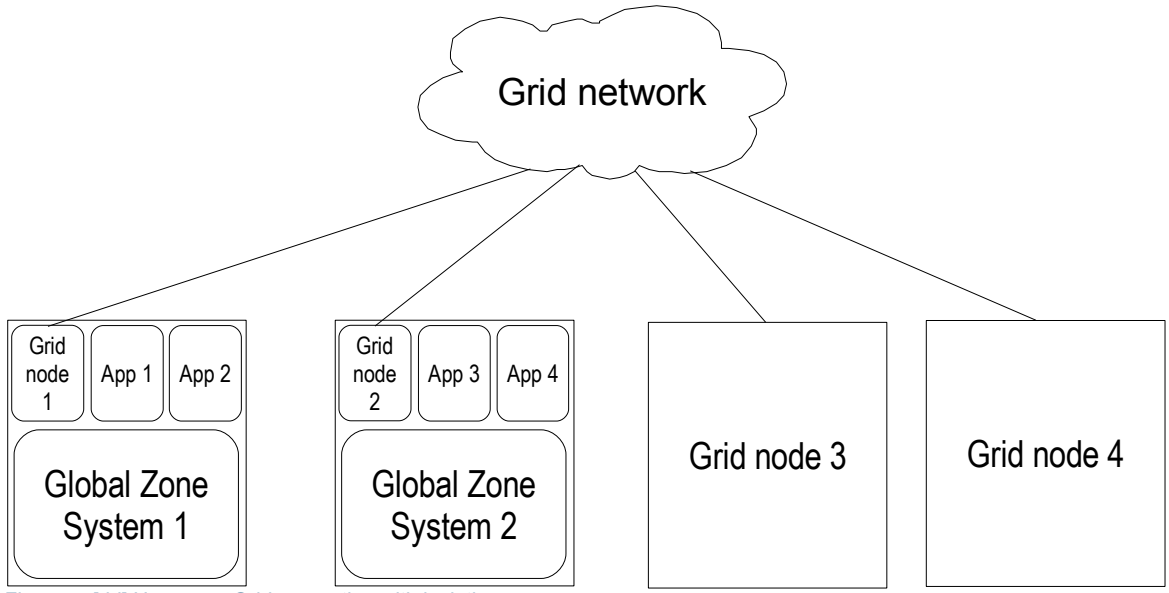


Figure 7: [dd] Use case: Grid computing with isolation

3.2. Small web servers

Requirement

[ug] One of the following situations exists:

- An Internet Service Provider (ISP) would like to have the option to set up web servers automatically, without additional costs. Based on this technology, the ISP wants to create an attractive offer for web servers with root access.
- The data center of a major company has many requests from departments for internal web servers for the purpose of information exchange between the departments. It is in the customers' interest to deposit web content as simply as possible, without many rules and arrangements (little effort). Therefore, the requesting department aims at a web server that nobody else works with. The data center is interested in keeping administrative costs low. It should therefore not be a separate computer. The data traffic will most likely be small and does not justify a full computer of their own.

Solution

[ug] The web servers are implemented by automatically installed zones. The following details are used in particular:

- Sparse-root zones, that is, zones inherit everything, if possible, from the global zone (4.1.1 Sparse-root zones).
- The software directory of the web server, e.g. `/opt/webserver`, is not inherited (`inherit-pkg-dir`) in order to facilitate different versions of the web server.
- Automatic creation of a zone per script
- Automatic system configuration in the zone with `sysidcfg`.
- Automatic IP address administration.
- Option: Automatic quick installation of a zone.
- Option: Application administrator with root access.
- Option: Software installation per mount.

Assessment

[ug] This use case has the following characteristics:

- The operating division's expenses for creating the zones are low.
- Since the expected load is very small, the consolidation effect is very large since only active processes in the zone require resources.
- The users of the automatically generated web servers in the zones are free to use a variety of different versions without having to re-educate themselves. That is to say, their costs are low and there is hardly any need for training.
- The web servers can work with the standard ports and the preset container IP addresses – no special configurations are required as when several web servers are to be operated on one system.

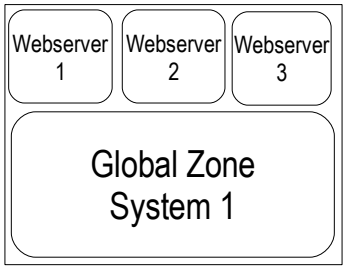


Figure 8: [dd] Use case: Small web server

3.3. Multi-network consolidation

Requirement

[dd] A company uses several different networks that are separated either by firewalls or by routers. Applications are run in the individual networks. The company would like to use the applications from different networks or security areas together on one physical system as an application itself does not require the capacity of a single system.

Solution

[dd] The individual applications are installed in one zone each. Zones are clustered together on physical servers according to certain criteria (redundancy, similar application, load behavior, etc.). Routing between the zones is switched off to separate the networks. The following details are used in particular:

- Creation of zones.
- Zones as runtime environments for one application each.
- Routing of the global zone on the interfaces is switched off so that zones cannot reach each other. That is, the zones can only reach addresses in their respective network.
- Use of exclusive-IP instances.

Assessment

[dd] This use case has the following characteristics:

- The network structure is simplified by economizing routes and routers.
- The number of required systems is reduced.
- Applications can be organized according to new aspects, e.g. all web servers on a physical server, or e.g. T2000 are used for web servers, T1000 are used for proxy servers, UltraSPARC IV+ systems for all databases, etc.
- The global zone can be used as the central administrative authority for all zones in a system. A separate administrative network can be placed on the global zones.
- Application administration is located within the zone. If the same applications are clustered together on systems, an application administrator can administer all applications in the zones out of the global zone more easily, or can simplify administration by the use of sparse root zones.

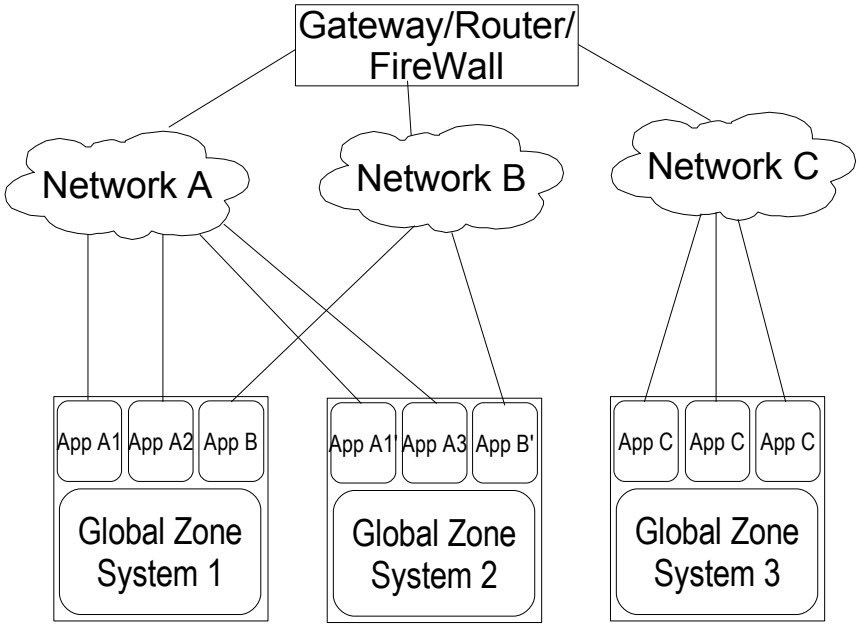


Figure 9: [dd] Use case: Multi-network consolidation

3.4. Multi-network monitoring

Requirement

[dd] A company has several different networks that are separated into several levels either by firewalls or by routers. A variety of computers are installed in the individual networks. Administration is to be simplified, and the company would like to be able to "look into" all the networks directly from a central location and administer without having to connect the networks by routing.

Solution

[dd] A central monitoring and administrator server is installed. On this server, several zones are created that have each a connection to a network. Monitoring or administration of the computers of the individual networks is done from the zones. The following details are used in particular:

- Sparse-root zones, that is, the zones inherit everything, if possible, from the global zone.
- All zones use the same monitoring and administration tools.
- Monitoring data are stored in file systems that are shared between zones.
- Data can be evaluated from a local zone or centrally from the global zone.
- From a central location (the global zone), central configuration files can be distributed directly to all zones or to all systems in the networks. Circuitous paths via routers and firewalls are omitted.
- Routing between zones must be turned off.
- Option: Use exclusive-IP instances.

Assessment

[dd] This use case has the following characteristics:

- The operating division's expenses for creating the zones are low.
- The administrative overhead decreases for systems in the networks since no multiple login via routers or firewalls must be performed.
- A single point of administration can be created.
- Relief of the strain on routers and firewalls stemming from network load and additional configurations.
- Use of uniform monitoring tools.
- Use of uniform configurations is simplified.

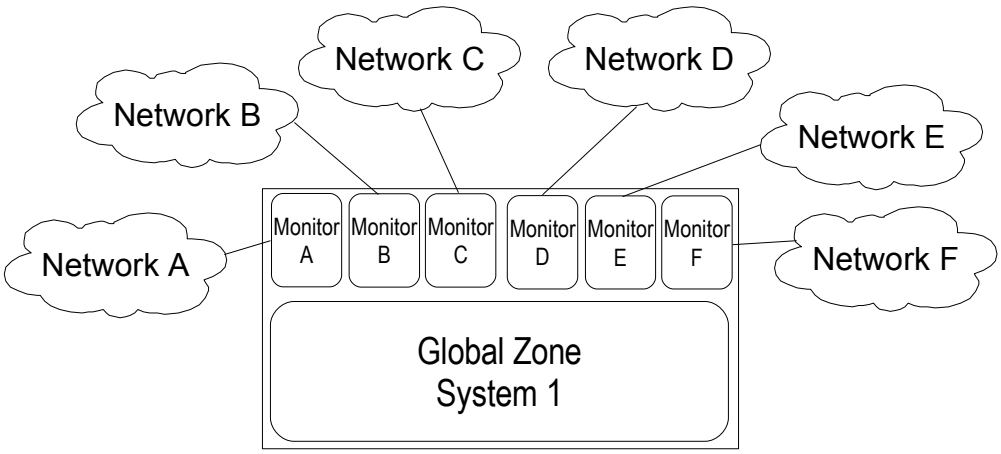


Figure 10: [dd] Use case: Multi-network monitoring

3.5. Multi-network backup

Requirement

[dd] A company has several different networks that are separated in different stages either by firewalls or by routers. Different computers are installed in the individual networks. The backup is to be simplified by allowing direct system backups in the individual networks to be performed from one location without having to connect the networks by routing.

Solution

[dd] A server with backup server software is installed. Several zones are created on this server that have a network connection to each network. Backup clients are started in the zones that communicate with the backup server in the global zone via the internal network or that write directly on an available backup device as a backup server. The following details are used in particular:

- Sparse root zones, that is, the zones inherit everything possible from the global zone.
- The backup client or backup server software is installed separately for each zone.
- A device is provided from the global zone to a local zone.
- Network setup to connect from the global to the local zone.

Assessment

[dd] This use case has the following characteristics:

- The operating division's expenses for creating the zones are low.
- Backup and restore can be organized and carried out from one central location.
- Higher backup speeds can be achieved by direct backup. Routers or firewalls are not burdened by backup data.
- Possibly, licensing costs are saved for backup software.
- Backup hardware is shared among departments and networks.

It has, unfortunately, turned out so far that some software manufacturers have not yet released backup server software for application in zones. Therefore, this use case is applicable with some restrictions only. Backup clients have already been released for application in zones here and there.

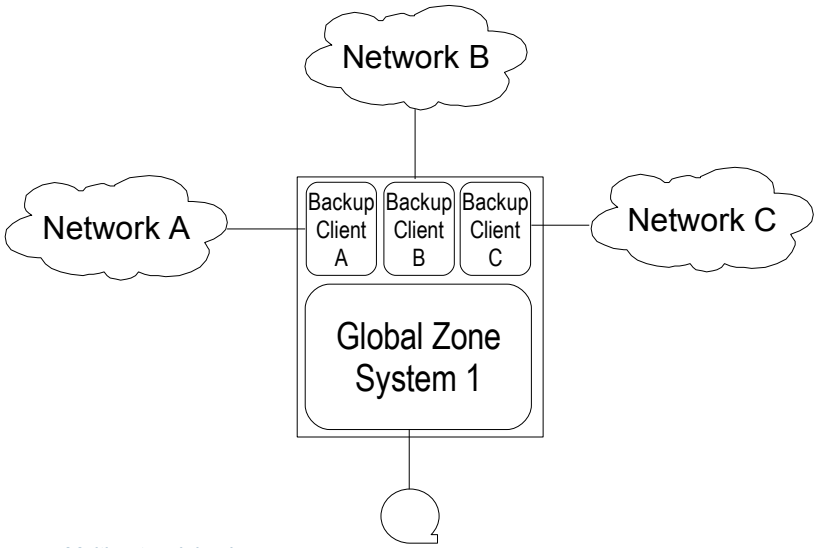


Figure 11: [dd] Use case: Multi-network backup

3.6. Consolidation development/test/integration/production

Requirement

[ug] Usually, further systems supporting the same application exist while an application is in production:

- Development systems
- Test systems
- Integration systems, with simulation of the application environment where applicable
- Disaster recovery systems

Solution

[ug] The different systems can be implemented on a computer in zones. The details:

- Sparse-root zones, that is, the zones inherit everything possible from the global zone.
- Option: Resource pools with their own processor set for production.
- Option: Application administrator with root access.
- Option: Software installation per mount.
- Option: OS release upgrade per flash image.

Assessment

[ug] This use case has the following characteristics:

- Fewer computers need to be operated overall. Therefore, savings can be made on space, power consumption and air conditioning.
- The applications can be tested on exactly the same environment that they are to be run on later.
- Switch to a new software version in production can be achieved simply by rerouting the load to the zone with the new version. Installation after test is not required.

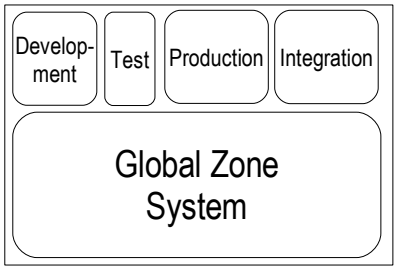


Figure 12: [dd] Use case: Consolidation development/test/integration/production

3.7. Consolidation of test systems

Requirement

[ug] To test software and applications, there are many test systems in the data center environment that are only ever used for tests. They are mostly used only for qualitative tests where systems are not stressed as with performance tests. However, switching between test environments by re-installation or restore of a shared test computer is out of question. The testers want to access their environment without waiting period or hassle over the right to access. Test systems are therefore underutilized.

Solution

[ug] Test systems are implemented by zones on a larger server. The details:

- Sparse root zones / whole root zones, as needed.
- File system decisions analogous to the production system.
- Option: Resource management with processor sets for simultaneous tests.
- Option: Automatic zone creation.
- Option: Software installation per mount.
- Option: Moving the zone among computers.
- Option: Automatic quick installation of a zone.

Assessment

[ug] This use case has the following characteristics:

- The operating division requires far fewer computers serving as test systems. Far fewer installations need to be performed.
- Expensive systems for performance tests only need to be purchased once, and not for every application. Performance tests on the machines shared by the use of zones must, however, be coordinated (operation).
- The operators can access the test installation at any time, qualitatively, at any rate; possibly not with full performance.

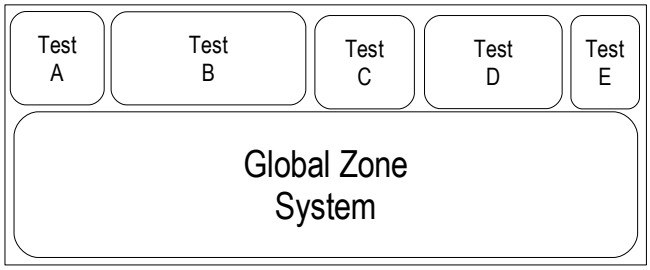


Figure 13: [dd] Use case: Consolidation of test systems

3.8. Training systems

Requirements

[ug] In training departments, computers that are provided for training participants (including pupils/students) must frequently be reset.

Solution

[ug] The training systems are implemented by automatically installed zones:

- Sparse-root zones, that is, the zones inherit everything possible from the global zone.
- Automatic zone creation per script.
- Automatic system configuration in the zone with `sysidcfg`.
- Automatic IP address administration.
- Option: `/opt` is not inherited (`inherit-pkg-dir`), for installation training.
- Option: Automatic quick installation of a zone.
- Option: Application administrator with root access.
- Option: Installation of shared software per mount

Assessment

[ug] This use case has the following characteristics:

- Operating the training computers is extremely simple since zones can easily be created again for the next course.
- Training participants themselves are root in the zone, which allows them to perform all essential administrative functions without posing a hazard to the total system. Re-installation of the computer is thus not necessary.
- The trainer can see the work of the training participants from the global zone.
- The costs for the training system therefore decrease drastically.

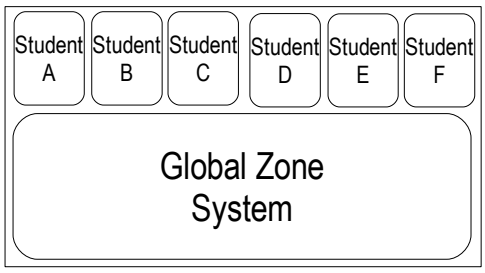


Figure 14: [dd] Use case: Training systems

3.9. Server consolidation

Requirement

[ug] In a data center, several applications run whose workload is too low (often much less than 50%). The computers themselves usually require a lot of electricity, cooling and space.

Solution

[ug] Several applications are consolidated in zones on one computer. The details:

- This can usually be done with sparse root zones.
- Install exactly one application per zone.
- Option: Software installation by shared directory .
- Option: Automatic software installation.
- Option: Resource management with resource pools.
- Option: Resource management with fair share scheduler.
- Option: Network resource management with IPQoS.
- Option: High availability by the use of a cluster.
- Option: Software installation per provisioning

Assessment

[ug] This use case has the following characteristics:

- The operating division's costs are lower since less space, energy and cooling are required.
- The application department has the same encapsulation of the application and precisely defined interfaces. The availability of the application increases.

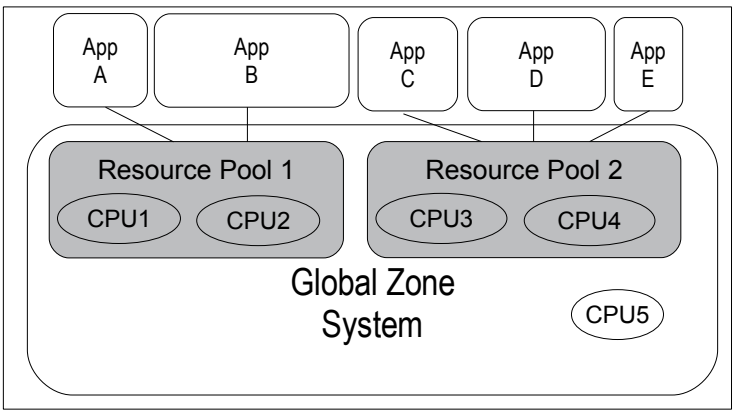


Figure 15: [dd] Use case: Server consolidation

3.10. Confidentiality of data and processes

Requirement

[ug] In the data center, applications are running on different computers because

- Certain departments want to be certain that data and processes are not seen by other departments.
- Services for different customers are to be consolidated. The customers request confidentiality of data and processes (which would allow conclusions to be drawn regarding business processes, if applicable).

Solution

[ug] Customer applications are installed in different zones:

- Applications are installed in local zones only .
- The file systems/devices with the data for the respective customers are made available in the corresponding zones only. This ensures confidentiality of the data even without strict discipline with respect to file access rights.
- Option: Software installation within the zone in a local file system present only in that zone, or a non-shared `/opt`.

Assessment

[ug] This use case has the following characteristics:

- The operating department achieves better utilization of systems.
- The operators/customers retain confidentiality.
- Basic costs are lower, by which the service can be offered more cheaply due to a higher profit margin.

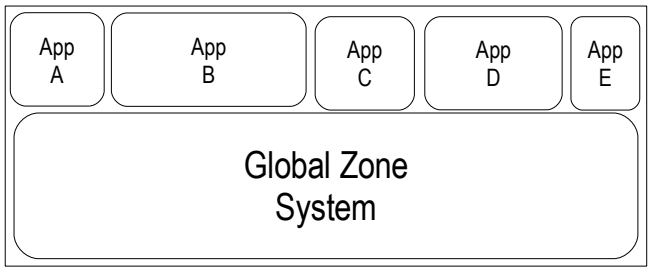


Figure 16: [dd] Use case: Confidentiality of data and processes

3.11. Test systems for developers

Requirement

[ug] Developers need test systems to test their application. Frequently, the interaction of several computers must be tested as well. Resources for the installation of test systems are usually limited. Since the developers spend most of their time working on developing, test systems have a low workload.

- It is possible to share the use of test systems but it increases the time until the test is performed since systems must be reinstalled or recreated.
- Scheduling conflicts can occur if different developers want to access systems at the same time.
- Backup/restore or installation are usually done by the operating department. The conventional use pattern of test systems creates a certain amount of overhead here.

Solution

[ug] Developers' test systems are implemented by means of medium-sized computers with zones. The details:

- Sparse root zones/whole root zones, as needed.
- Data are stored locally.
- Automatic zone creation.
- Automatic system configuration within the zone with sysidcfg.
- Automatic IP address administration.
- The developer becomes application administrator with root access.
- Software to be tested is installed locally.
- Option: Software installation (in part) per mount.

Assessment

[ug] This use case has the following characteristics:

- The operating department must operate far fewer test systems; each new test system therefore only requires disk space, a separate computer is not needed.
- The creation of test systems can be fully automated; the overhead accrued for the operating department is therefore much lower.
- Developers can retain several test systems at the same time; only the ones currently in use must be booted up. A comparison of functionalities in the different versions is possible without much effort.
- No waiting period (for restore) or coordination (with other tests run on the machine) is required to perform a test.

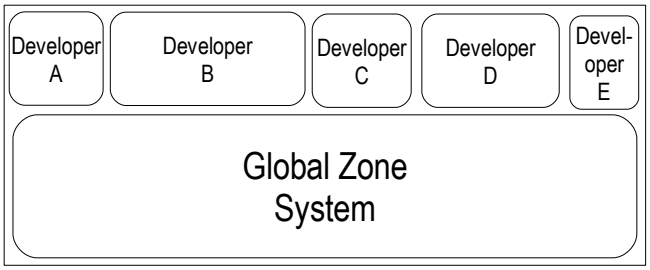


Figure 17: [dd] Use case: Developer test systems

3.12. Solaris 8 and Solaris 9 containers for development

Requirement

[ug] There are still systems running in companies under Solaris 8 or Solaris 9 with self-developed software, and it must be possible for these systems to continue software development or debugging.

- The operation of old systems under Solaris 8 or Solaris 9 is still necessary
- Rarely used development systems are relatively expensive (ongoing maintenance)
- Migrating the development to current systems does not guarantee executability of the software developed on the old systems.

Solution

[ug] On a current system under Solaris 10, Solaris 8 or Solaris 9 zones are set up with the development environment installed for the appropriate operating system. The details:

- Branded zones, that is, the environment of an older Solaris version is active within the zone.
- Installation of an archived legacy system.
- CPU resource management based on the Solaris fair share scheduler.
- Memory resource management.

Assessment

[ug] This use case has the following characteristics:

- Applications are operated and further developed using the original version. Costly migration of the application to a new version is not necessary.
- The original development software can continue to be used; there are no compatibility problems.
- Current hardware based on Solaris 10 can be used.
- Environmental costs such as electricity and cooling are lower, and capacity is higher than with an old computer.
- The advanced software can be installed and operated both on a system under Solaris 8 or Solaris 9 as well as in a system with a corresponding branded zone.

3.13. Solaris 8 and Solaris 9 containers as revision systems

Requirement

[ug] For legal reasons or due to revision requests, it is necessary to have certain systems available for years under Solaris 8 or Solaris 9.

- Operating old systems under Solaris 8 or Solaris 9 is expensive (maintenance costs).
- Old hardware perhaps has not a maintenance contract.
- Migration to more up-to-date systems is expensive.
- Migration to an up-to-date operating system or to up-to-date hardware requires a new software version.

Solution

[ug] Legacy systems are run in Solaris 8 or Solaris 9 containers on an up-to-date Solaris 10 system. The details:

- Branded zones, that is, the environment of an older Solaris is active within the zone.
- Installation of an archived legacy system.
- CPU resource management based on Solaris Fair Share Scheduler.
- Memory resource management.
- Application administrator or the administrator of the previous legacy system administrator get root access in the zone.

3.14. Hosting for several companies on one computer

Requirement

[ug] An application service provider operates systems for a variety of companies. The systems are underutilized.

Solution

[ug] The applications are consolidated into zones on one computer. The details:

- Separate file systems for each zone.
- Option: Server integration into the company networks via NAT.
- Option: Software installation per mount.

Assessment

[ug] This use case has the following characteristics:

- The application service provider saves data center space, energy and cooling. He is therefore able to offer his services at a lower price.
- The ASP's customer can purchase the service at a more favorable price.

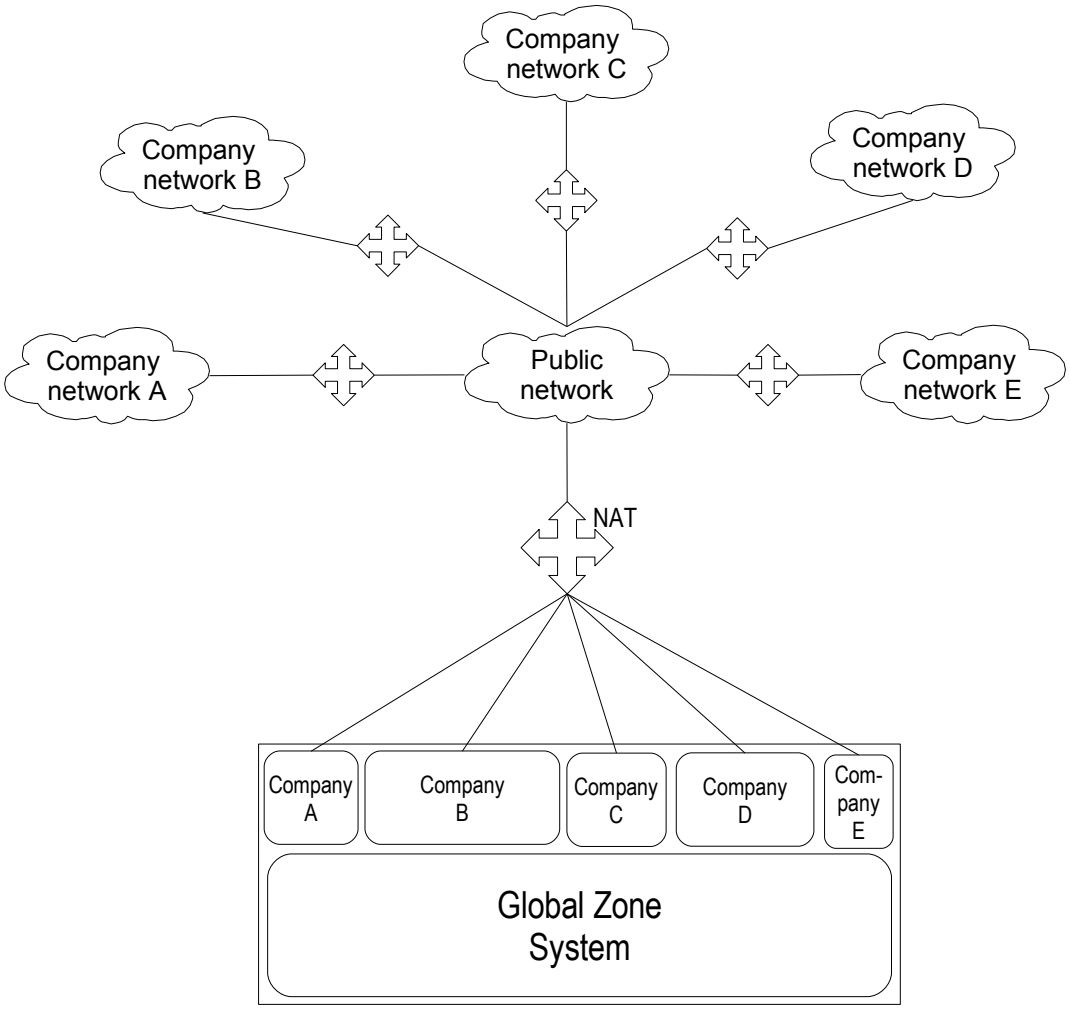


Figure 18: [dd] Use case: Hosting for different companies on one computer

3.15. SAP portals in Solaris containers

Requirement

[da] The operation of SAP system environments is becoming more complex. This results in:

- Application-specific requirements
- Standardization defaults
- Simple commands and scripts for basic administration
- Automatization potential of processes pertinent for operation
- Cost reduction requirements
- Simple consolidation options by Solaris containers
- Software licenses (virtualization tools, volume manager, snapshot technologies, etc.)
- Optimization of workloads
- High availability requirements
- Flexibility by relocatable containers
- Increased data security
- "Complex" customer requirements
- Cloning of SAP systems without downtime
- New SAP installation environments in minutes
- Minimizing/optimizing upgrade times
- Revision control of SAP systems

Solution

[da] If one considers current SAP system landscapes one can see that 70-80% of the systems are not productive systems. These, however, require the greatest amount of maintenance and administrative effort and thereby cause the greatest cost. Therefore, an "Easy-Start“ solution was developed for these systems which is described in "Sun Solaris and SAP":
http://de.sun.com/servicessolutions/virtualization/ressourcen.jsp

- All SAP applications are run in Solaris 10 containers on shared storage
- Whole-root zones provide the required level of flexibility and safety for SAP operation
- ZFS as the basis for container implementation, to minimize downtime for SAP systems
- A toolset consisting of several different scripts allows quick and easy construction of a virtualization solution with Solaris containers and ZFS for SAP systems.

Assessment

[da] This use case has the following characteristics:

- Quick introduction of container and ZFS technology.
- The operating department incurs only few expenses for creating the zones.
- Great consolidation effect.
- High flexibility

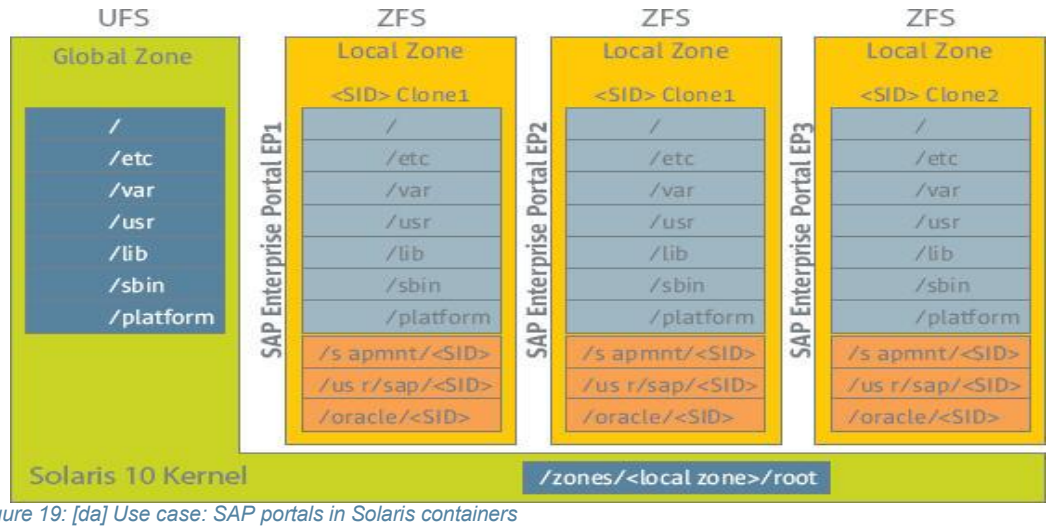


Figure 19: [da] Use case: SAP portals in Solaris containers

3.16. Upgrade- and Patch-management in a virtual environment

Requirement

[da] Virtualization by means of Solaris Containers allows the application to be disengaged from the hardware. An application can thus be run very simply on different servers. Data center operations must ensure the availability of these applications by means of suitable measures. This includes scheduled downtime requirements but also the requirement to protect against unscheduled downtime. Scheduled downtime is required for preventive maintenance of any IT infrastructure. By far the most scheduled downtime is required for patching systems. Solutions that merely focus on transferring an application from one machine to another therefore completely miss the actual point. Simple and automated release management is much more important for the entire operating system. Downtimes for updating the application and for patching the operating system (OS) in a virtualized environment can therefore be regarded as completely independent of each other.

Solution

[da] By using live upgrade and upgrade-on-attach associated with ZFS, the required prerequisites for efficient release management arise as a result for data center operations.

Installing a kernel patch for an operating system is possible without downtime for the application. An operating system can be upgraded while running by using live upgrade. Activation will occur at a later, planned point in time which in many data centers fits in with the service level agreement.

Solaris live upgrade is the optimal procedure for performing an upgrade for such a "virtualized" system or installing patches. The procedure, in particular by the use of ZFS, is characterized by the fact that the length of the required maintenance window and thus the downtime of system applications are minimal. This also always implies that all applications installed on the system are affected simultaneously. If several applications with different maintenance windows are run together on one system, a live upgrade cannot be performed.

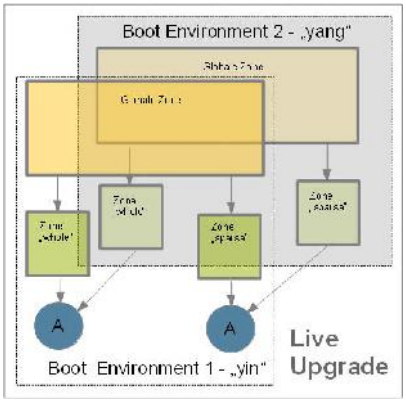


Figure 20: [da] Live upgrade

With the new update-on-attach technology for local zones, another mechanism is available in Solaris by which this problem can be solved. It allows an update to be performed on a Solaris Container including the application, scheduled and within the maintenance period defined for the application in the Service Level Agreements (SLAs). In the process, the actual update is done by simply relocating the containers to another system with a newer version of the operating system. Relocatable containers also allow an application "to be protected" from a live upgrade. If operations cannot find a common maintenance window for the applications on a system, individual containers that contain these applications can be relocated to other systems with the same version of the operating system. This occurs systematically during the timeslot available for the application respectively. The concept was published in a whitepaper that can be accessed through the following link:

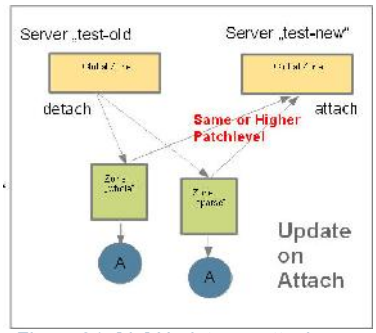


Figure 21: [da] Update-on-attach

http://wikis.sun.com/display/SAPonSun/Links+(to+SAP+and+or+Solaris+topics)

Assessment

[da] This use case has the following characteristics, among others:

- Minimal downtime required through live upgrade and ZFS
- High flexibility and "plan-ability" through update-on-attach
- High security through "rollback" options
- Simple administration and simple operation
- Technologies are available in Solaris 10 free of licensing fees

3.17. "Flying zones" – Service-oriented Solaris server infrastructure

Requirement

[os] A highly available virtualization platform to run business-critical applications should meet the following requirements:

- Breaking the rigid dependency between services and hardware
- Application-oriented and workload-optimized utilization of resources
- No physical separation of production and integration environments
- Functionality in case of disaster

Solution

[os] Business-critical applications are run in Solaris 10 zones both on SPARC as well as on x64 systems. Functionality in case of disaster and location-spanning shifting of zones occurs within a grid cluster by means of Sun Cluster 3.1. The details:

- Sparse root zones, that is, the zones inherit everything possible from the global zone
- Manual relocation of zones.
- Automatic failover of zones with SC 3.1, container monitoring with SC 3.1 or optional monitoring at the application level by special tools.
- CPU resource management based on Solaris Fair Share Scheduler.
- Memory Resource Management.
- Application administrator with root access.
- Surveying the workload and the resource requirements on a dedicated staging server of the cluster (figure).

Assessment

[os] This use case has the following characteristics:

- Up to 24 grid clusters (with up to 8 nodes) based on SPARC or Sun x64-systems. Average CPU workloads of up to 60% can be achieved with up to 8 zones per Solaris instance.
- Consolidation factor of currently 5 on average, that is, significant savings are achieved for space, power consumption and air conditioning.
- Distinct increase in efficiency while operating by using a standardized environment (only one type of OS, only two kernel and CPU architectures).
- Staging server as part of the grid cluster allows accurate measuring of resource requirements and detailed capacity planning for the grid cluster.
- Production and integration environments are operated in a mixed and location-spanning manner. In a disaster situation, integration environments are shut down and production environments are started on the nodes of the remaining data center.

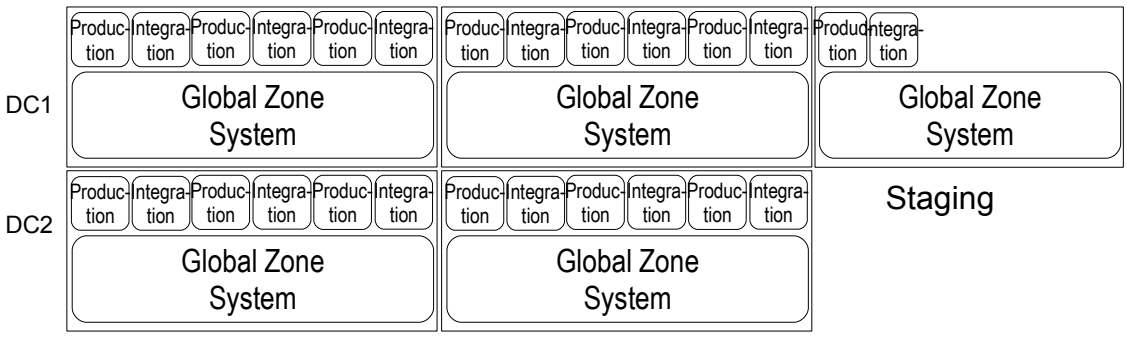


Figure 22: [os] Use case: "Flying zones" - Services-oriented Solaris server infrastructure

3.18. Solaris Container Cluster (aka "zone cluster")

Requirement

[hs]

- In a virtualized environment based on Solaris containers, the administrator of a zone should also be able to administer the cluster part of the application in the zone.
- The use of "capped" containers to minimize (Oracle) licensing fees should also be expanded to include Oracle RAC (Real Application Cluster) environments (certification by Oracle is not yet completed as of today (June 2009)).

Solution

[hs] Virtual clusters where containers form the virtual nodes of the cluster are implemented through the implementation of Solaris Container Clusters. The administrator of the global Sun Cluster environment creates the virtual cluster and allocates the required resources to it, just like the global administrator allocates resources to individual containers. These include:

- Main memory
- CPUs
- File systems
- Network addresses

Now, the administrator of the local zones that constitute the virtual nodes of the container cluster has all administrative rights available that allow him to manage highly available services with the assistance of Sun Cluster. These include:

- Setting up resource groups
- Setting up resources
- Starting and stopping resource groups and resources

Assessment

[hs] This use case has the following characteristics:

- Virtualization down to the cluster level
- Delegation of administration even of the (virtual) cluster to the container administrator
- Reduction of licensing costs by using "capped" containers even in the Oracle RAC environment (status of certification by Oracle still "pending").

4. Best Practices

The following chapter describes concepts for the implementation of architectures with Solaris containers.

4.1. Concepts

4.1.1. Sparse-root zones

[dd] Sparse root zones are zones that inherit the following directories from the global zone as *inherit-pkg-dir*:

- */lib*
- */platform*
- */sbin*
- */usr*

- The software packages are installed only once centrally in the global zone from the OS and application software and made available in all local zones as *inherit-pkg-dir*. An *inherit-pkg-dir* is a read-only loopback mount from the global zone to the local zone. The packages appear to be installed in the local zone by corresponding entries in the pkg database.
- Software maintenance (update, patching) of inherited packages is simply done centrally for all installed zones from out of the global zone.
- Sparse root zones are installed and patched considerably quicker by inheritance.
- */opt* is not set up as *inherit-pkg-dir*.
- Disk space for *inherit-pkg-dir* is occupied only once in the global zone on the hard drive and is used multiple times in the sparse root zones. Thus, only a minimum amount of disk space which, depending on the architecture and the scope of the software installation, amounts to about 70 MB (x86/x64) or 100 MB (SPARC), is required by a sparse root zone.
- Programs and libraries are loaded into the main memory only once, exactly like shared memory. Since sparse root zones share the same programs and libraries with other zones, space in the main computer memory is occupied only once, regardless of how many times it is accessed and how many zones use it. If the same software is used frequently in different zones, this will lead to economization in required main memory.
- Sparse root zones are not suitable for running different software versions in zones if this software is located in an *inherit-pkg-dir*. Either whole root zone configurations should be used for this, or the software must not be installed in an *inherit-pkg-dir*.
- If the software installation process requires writing permission in an *inherit-pkg-dir* of the local zone, a whole root zone configuration should be selected.

4.1.2. Whole-root zones

[dd] Whole root zones have no *inherit-pkg-dir*, that is, when the zone is created, the packages are copied from the global zone into the local zone. This does not include packages that were installed exclusively for use in the global zone (e.g. the kernel and the driver) or packages installed with *pkgadd -G*.

- Whole root zones are almost completely independent from the global zone since the packages are present as copies and can be modified and patched separately.
- The required space for a whole root zone encompasses approx. the requirements of a complete Solaris installation.
- Creating and patching a whole root zone requires more time.
- Apart from the kernel, whole root zones do not share any programs and libraries among each other or with the global zone. They are therefore loaded into the main memory again, regardless of whether another zone has already loaded its own copy of the same program into the main memory. This leads to an increased need for main memory resources required by whole root zones.

4.1.3. Comparison between sparse-root zones and whole-root zones

[dd] From the considerations listed above, a comparison can be drawn between sparse-root zones and whole-root zones. In the vast majority of cases, sparse-root zones are used.

	Sparse root zone	*Whole root zone*
Disk space required	*70 - 100 MB*	*2600 - 3600 MB*
Quick creation of zones	+	-
Quick patching of zones	+	-
Central software installation and maintenance	+	+
Efficient use of main memory	+	-
Independent installation, modification and patching of parts of the operating system within the zone	-	+
Write option in system directories	-	+

Table 3: [dd] Comparison between sparse-root zones and whole-root zones

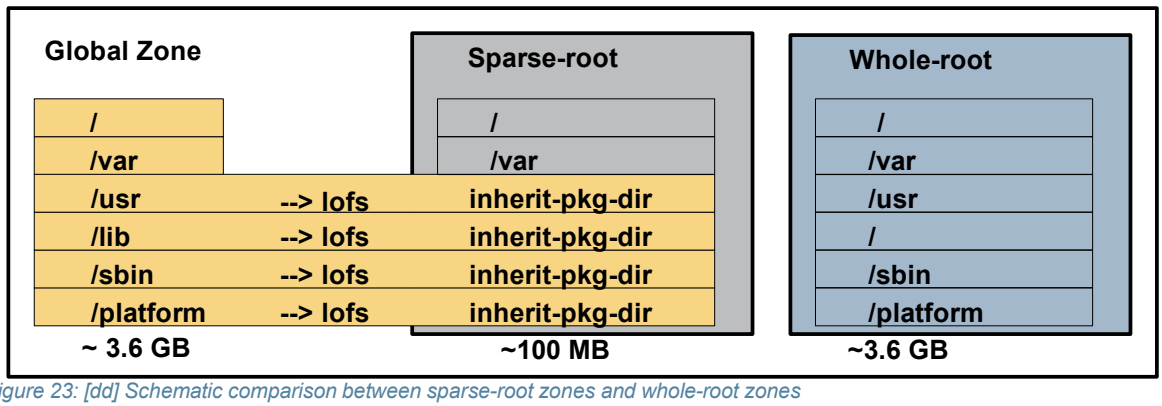


Figure 23: [dd] Schematic comparison between sparse-root zones and whole-root zones

4.1.4. Software in zones

[dd] For users and operators, the question arises whether their software runs in a local zone and also whether it is certified for that purpose by the software manufacturer. Applications can be qualified according to `http://developers.sun.com/solaris/articles/zone_app_qualif.html` for use in local zones.

As a general rule, applications that get by with normal user privileges also run in a local zone.

Software that requires special privileges or must run under root privileges should be considered separately. Direct access to devices, device drivers, separate kernel modules, ISM or direct access to the network and network modules are considered to be particularly critical. Details on this can be read at http://developers.sun.com/solaris/articles/application_in_zone.html under "Bringing your Application into the Zone".

If the software is able to run in a local zone, the installation and configuration mechanism of the application must furthermore be considered. Thus, care must be taken that during the installation in local zones no write operations are required in `inherit-pkg-dir`, e.g. for an installation in /usr or /usr/local, if /usr is an `inherit-pkg-dir` for a sparse root zone.

4.1.5. Software installations in Solaris and zones

[dd] The zones' directory structure is determined mainly from the need to install software with special needs in this area. Prior to creating the zones, this question should also be discussed extensively with regards to the intended purpose of the zone. In general, the following types of software installations are possible in Solaris:

- Software installation from an archive – referred to below as non-pkg software
  - The software is installed per installation script by copying from an archive or a directory.
  - No information on the installed software is contained in the pkg database.
  - Software management must be carried out by tools belonging to the software itself or by simple Solaris programs such as `cp, mv, rm,...`
- Software installation as a package with `pkgadd` – referred to below as pkg software, etc.
  - The software is distributed in pkg format and installed with `pkgadd`.
  - The pkg information is present in the pkg database under /var/sadm/install.
  - The pkg database is used for pkg management such as update, patch, remove and zone installation.

Furthermore, it must be decided whether software is to be installed in all zones (global and local) or only in a certain zone, and who has permission to install the software (the global zone or the local zone).

The following chapters describe various software installation scenarios related to the usability of the software.

4.1.5.1. Software installation by the global zone – usage in all zones

- non-pkg software
  - Software A is installed by the global zone, e.g. in /software/A.
  - /software/A is made available to all local zones as a read-only directory (e.g. zonecfg: add fs).
  - /software/A/cfg is a soft link according to e.g. /opt/A/cfg and contains the zone-dependent configuration.
  - /software/A/data is a soft link according to e.g. /opt/A/data and contains the zone-dependent data.
  - The software can be used in all zones.
  - Only the configuration and the software data can be modified by the zones.
- pkg software
  - Software P is installed by the global zone with `pkgadd` in the global zone and all local zones.
  - An entry is made in the pkg database in all zones.
  - If the file to be installed is not located in an `inherit-pkg-dir`, the file is copied into the corresponding directory.
  - The configuration and data directories of software P can be placed in areas writable for the local zones by corresponding soft links (see non-pkg software).
  - The software can be used in all zones.
  - Only the configuration and the software data can be modified by the zones.

4.1.5.2. Software installation by the global zone – usage in a local zone

- non-pkg software
  - Software A is installed by the global zone, e.g. in /zone-z/software/A
  - /zone-z/software/A is made available as a writable directory of local zone z (e.g. zonecfg: add fs)
  - The configuration and the data of software A can be modified by zone z, since this part is available and writable exclusively by this zone.
- pkg software
  - This case is not recommended for pkg software. In this case, it is better to install the software directly in the local zone (installation by the local zone – usage in the local zone).

4.1.5.3. Software installation by the global zone – usage in the global zone

- non-pkg software
  - Software A is installed by the global zone e.g. in /software/A
  - /software/A is available to the global zone as a writable directory.
  - Configuration and data of software A can be modified by the global zone since this part is available and writable exclusively by this zone.
- pkg software
  - Software P is installed in the global zone with `pkgadd -G`.
  - In the global zone, an entry is made in the pkg database.
  - The software can be used in the global zone only.
  - The configuration and the data of the software can be modified from the global zone.

4.1.5.4. Installation by the local zone – usage in the local zone

- non-pkg software
  - /software/A is available to the zone as a writable directory.
  - Software A is installed by the zone e.g. in /opt/software/A.
  - The configuration and the data of software A can be modified by the zone since this part is available and writable exclusively by this zone.
- pkg software
  - Software P is installed by the zone with `pkgadd -G`.
  - The installation and configuration directory of the package must be available and writable by the zone. If this is not the case, e.g. with /usr/local (if /usr is an `inherit-pkg-dir` for a sparse-root zone), a soft link /usr/local --> ../../opt/local, for example, can be set up in the global zone. This points with /opt/local into a writable part of the local zone.
  - An entry in the pkg database is made for the zone.
  - The software can only be used in this zone.
  - The configuration and the data of the software can be modified by the zone.

4.1.6. Storage concepts

4.1.6.1. Storage for the root file system of the local zones

[ug] It is usually sufficient for several zones to share a file system.

If the influence from the zones to the file system is to be uncoupled, it is recommended to give each zone its own file system.

With the new ZFS file system by Solaris 10, this can be achieved with little effort. Since Solaris 10 10/08, ZFS can be used as zone root. Prior to Solaris 10 10/08, although it was possible to run a zone on ZFS, an upgrade was not supported; therefore, Solaris 10 10/08 is urgently recommended for zones on ZFS.

With the availability of iSCSI in Solaris 10 8/07 as a client initiator and as target emulator, this technology as well can be used as a storage technology for zone root file systems. The mobility of zones, e.g. in connection with `zoneadm detach` / `zoneadm attach`, thereby increases by the flexible utilization of iSCSI volumes in different systems with network access.

For a system that is to have a high number of zones installed on it, using a storage subsystem with cache for the root file systems is recommended. Each OS instance and by association each zone write messages, log entries, utmp entries, etc. on the root file system at irregular intervals. The applications also use the root file system for logs if applicable. In many instances, this can lead to a bottleneck on the file system (less critical for ZFS).

4.1.6.2. Data storage

[ug] Data storage (files, databases) is planned in the same way as for dedicated computers. The storage devices are connected to the computer that contains the zones. The file systems are mounted directly in the global zone and transferred to the zones per loopback mount (`zonecfg:add fs`). Then, the file systems can also be used for data transfer by other zones, if they are configured.

Alternatively, it is possible to have file systems (`zonecfg:add fs`) mounted by the `zoneadmd(1M)` directly when booting the zone, thereby making them available to individual zones exclusively. Since in this case access to the raw device and the block device is not possible within the zone, such a file system while booting a zone is mounted with `zoneadm -z <zone> boot` by the global zone at e.g. /zones/<zonename>/root/<filesystem> and appears in the local zone as /<filesystem>. However, due to **bug 6495558**, the file system will not be checked or repaired by `zoneadm` in case of need. To plan the file system layout of zones, alternatives should therefore be used if necessary.

Devices, e.g. for databases, can also be specifically assigned to a local zone (`zonecfg:add device`). Assigning the same device to several zones is useful in exceptional cases only; coordination of the device use is strongly advised.

4.1.6.3. Program/application storage

[ug] This procedure (see above) can also be selected for programs used by the applications.

All software can be installed in a global zone directory, and this directory can be assigned to the local zones (`zonecfg:add fs`). This is comparable to software being located on an NFS server in the data center; here, a local file system is used which is made available to the zone. Thereby, the software is available to all zones after being installed once. The computer's autonomy remains untouched. This type of software distribution is useful for non-pkg software only. pkg software is distributed or inherited to the zones by the zone's installation mechanisms.

4.1.6.4. Root disk layout

[dd] Depending on availability requirements, root disks within a system are mirrored via internal disks or made available through a variety of controllers and external storage devices. Even nowadays, the entire OS installation is installed in `/` in many cases, that is, without any further partitioning into `/usr` or `/opt`. Only `/var` must be set up as a separate file system in order to avoid flooding / by e.g. big log files. Furthermore, additional space must be provided for a live upgrade scheduled for later.

Once the number of required slices per disk is depleted, `/var/crash` can be translocated into `/var`, or the use of soft partitions can be considered.

Frequently, `/usr/local` is needed as a writable directory in a local zone. If for example a sparse-root zone is used and `/usr` is an `inherit-pkg-dir`, then it is not possible to write into `/usr/local` in the local zone. In such cases, it is advisable to set up a soft link in the global zone, e.g. `/usr/local -> ../../opt/local`. Since `/opt` is normally separate for each zone, writing permission exists for each of these zones. It is important to use relative links so that even an access from the global zone to e.g. `/zones/zone-z/root/usr/local` ends up in the correct directory.

The companion DVD software is installed as `/opt/sfw`. If it is to be installed centrally, `/opt/sfw` would furthermore need to be specified as `inherit-pkg-dir`. StarOffice for example installs as `/opt/staroffice`.

File system sizes vary depending on the type and scope of the installations.

An example root disk layout is shown here (5.1.3 Root disk layout).

4.1.6.5. ZFS within a zone

[ug] Solaris 10 6/06 makes the new ZFS file system available for the first time. Since ZFS is not based on a special mounted device, a mechanism is implemented in the `zonecfg` command by which a ZFS file system can be transferred to a zone. This is achieved through the `zonecfg` subcommand `add dataset` . The ZFS file system then becomes visible in the zone and can be used there and administered (with limitations).

With ZFS, file system attributes can be specified that are activated immediately. Here, the attributes `quota` (maximum allocatable disk space) and `reservation` (guaranteed disk space) are of interest for the file system. If a file system is transferred to a zone through `add dataset`, the administrator of the global zone can specify the attributes `quota` and `reservation`. The administrator of the local zone cannot modify these parameters in order to access the resources available in other zones.

ZFS allows file systems to be hierarchically structured. This is also true for the ZFS file systems transferred to a zone, which can be further subdivided by the zone administrator. If the attributes `quota` and `reservation` are specified for the transferred file system, the limitation will also continue to apply within the zone. Consequently, the resources available within the zone are limited.

User-defined attributes for ZFS file systems were introduced with Solaris 10 5/08. These are particularly useful, especially in the case of ZFS file systems transferred to zones, for documenting ZFS file system use (5.1.12.10 User attributes for ZFS within a zone).

4.1.6.6. Options for using ZFS in local zones

[hes] Depending on the manner of configuration of ZFS in zones, there are different application options for ZFS in zones.

ZFS operation in a local zone	*Allocation of an individual ZFS within a zone – legacy mount*	*Adding of a ZFS dataset to a zone / Creation of a ZFS in the local zone*	*Adding of a ZFS volume dataset to a zone*	*Using of a ZFS filesystem via lofs*
umount	no	yes	yes	no
destroy	no	yes	no	no
create snapshot	no	yes	no	no
zfs set	no	yes	no	no
ZFS mount visible in global zone	no	no	no	yes

Table 4: [hes] Options for using ZFS in local zones

4.1.6.7. NFS and local zones

[ug] The use of zones does not change anything in the global zone with respect to NFS. A local zone can mount file systems from NFS servers. The following restrictions must be observed:

- A local zone cannot be used as a Solaris NFS server, that is, it cannot serve any file systems itself since the NFS service runs in the kernel and cannot yet run in a local zone.
- With a userland NFS server (e.g. Sourceforge.net: unfs3, not delivered with Solaris) a zone can be used as an NFS server.
- A local zone should not mount a file system from its global zone. This seems to be possible since the mount is possible, but loss of data can occur (**bug 5065254**)

4.1.6.8. Volume manager in local zones

[ug] One frequently asked question is how to use a volume manager in a local zone. Unfortunately, this is not possible.

On the one hand, a volume manager such as the Solaris Volume Manager (SVM) or the Veritas Volume Manager (VxVM) needs drivers that cannot be loaded separately in a local zone. On the other hand, a volume manager creates device nodes in /dev which are used to access the volumes that have been created. It is not possible to create a device node inside of a local zone, since this would represent a security hole. If a zone would be able to create any device node, then a zone administrator could create a device node for a disk that is not assigned to the zone, and would have finally read- or write-access to that data.

That is why the creation of device nodes within a local zone is forbidden by restricting privileges for systemcalls inside a local zone. However, a volume manager needs these functions and can therefore not operate within a local zone.

4.1.7. Network concepts

4.1.7.1. Introduction into networks and zones

[dd] A network address is not mandatory when configuring a zone. However, services within a zone can only be reached from the outside through the network. Therefore at least one network address per zone is typically required (configurable with `zonecfg`).

A zone's network address can be placed as a virtual interface in any physical interface (*`shared IP instance`*) or directly in a physical interface assigned exclusively to the zone (*`exclusive IP instance`*). The different types of IP instances are explained in (4.1.7.4 Exclusive IP instance).

For shared IP instances, routes for the local zone networks can be entered in the zone configuration. This ensures that when the zone is booted, the corresponding routing entry exists in the global zone. The IP stack of the global zone contains and manages all routes of the shared IP instance. Exclusive IP instances manage their own routing tables and are assigned to exactly one zone.

4.1.7.2. Network address management for zones

[ug] DHCP is not possible for addresses of zones with shared IP instances since DHCP is based on the HW address of the network interface (MAC). Zones use a virtual address on a shared network interface: therefore, they have the same MAC address as the interface in the global zone. The management of network addresses for zones must therefore take place in a different manner.

Basically, the following types of management are possible:

- Manual list-keeping.
  When configuring the zone, the IP address must therefore be tagged in the list.
- Predefining the IP addresses with the zone name in name service.
  When configuring the zone, a script can thus automatically detect the IP address of the zone if the IP name can be computed from the zone name (Cookbook).
- If many zones are to be set up on a system, it is advisable to allocate an entire range of IP addresses in advance where the network address is equal to the intended zone name. This ensures definite allocation.
- Integration into the IP naming system of the target environment,
  f.e. integration into the organizational processes of the company's IP allocation.

4.1.7.3. Shared IP instance and routing between zones

[dd] Each zone has at least one IP address of its own and its own TCP and UDP port numbers. Applications that are used in zones attach themselves to the IP addresses visible in the zone and also use them as sender addresses. This allows logical network separation between the zones.

If zones are located in different logical subnets as a result of corresponding address allocation, and if it is necessary that the zone communicates with other networks, separate routes must exist for each zone. These are placed in the global zone by means of zone configuration since the routing table is located in the TCP/IP stack which is shared among all zones (*`shared IP instance`*). If such a route is set up for a zone, inter-zone communication (local zone to local zone) takes place directly via the shared IP instance. If this inter-zone communication is to be prevented, so-called reject routes must be used that prevent any communication between two IP addresses of a single IP instance.

Another way to inhibit communication between shared-IP zones is by configuration of the IP Stack with ndd:

```
ndd -set /dev/ip ip_restrict_interzone_loopback 1
```

This can also be set into /etc/system to make it permanently.

If targeted communication between two local zones is required but if it should be conducted e.g. via an external router, load balancer or firewall, NAT-capable routers must be used. Corresponding setups are discussed in section 5. Cookbooks.

4.1.7.4. Exclusive IP instance

[dd] With exclusive IP instances, an almost complete separation of the network stacks between zones is achieved (from Solaris 10 8/07).

A zone with an exclusive IP instance has its own copy of variables and tables that are used by the TCP/IP stack. As a result, this zone has its own IP routing table, ARP table, IPsec policies, IP filter rules or ndd settings. Within the zone with an exclusive IP instance, the use of `ifconfig(1M) plumb`, `snoop(1M)`, `dhcp(5)` or `ipfilter(5)` is possible.

To use exclusive-IP at least one separate physical or tagged VLAN network interface has to be assigned to a zone and the ip-type attribute has to be set accordingly (`zonecfg: set ip-type`). Tagged VLAN interfaces get their own names that are composed as follows:

```
Interface = Interfacename<Instance + (VLAN-ID * 1000)>
```

e.g. bge4003 uses the VLAN with ID 4 on bge3

The admin of the local zone himself configures the IP address on the interface. The following interfaces are currently GLDv3-enabled: bge, e1000g, ixgb, nge, rge, xge. Although the older ce-interfaces are not GLDv3-enabled, but they are supported as well.

Further details on supported network interface cards can be found at:
http://opensolaris.org/os/project/crossbow/faq/#ip_instances

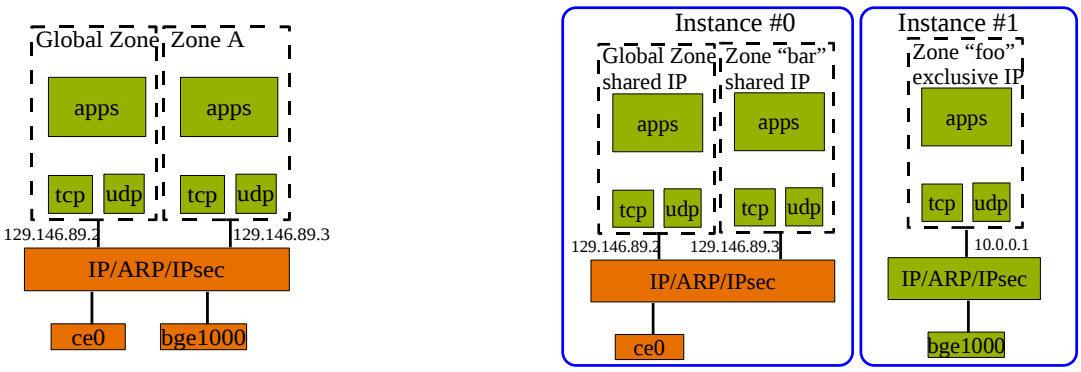


Figure 24: [dd] Comparison between shared IP instance (left) and exclusive IP instance (right)

4.1.7.5. Firewalls between zones (IP filter)

[dd] Firewalls can be used in zones to filter packages or for NAT. Thus, for example, the data traffic between certain zones can be completely prevented or limited for a network port. If the firewall is to be used on the system itself together with the zones, two versions can be distinguished in general:

- For shared IP zones: The firewall and its rules must be configured, started and administered in the global zone.
- For exclusive IP zones: The firewall and its rules must be configured, started and administered in the respective local zone.

IP filter is part of Solaris 10 and OpenSolaris. It contained as a firewall and was extended such that it can also be used as a firewall between a system's shared IP zones.

(Cookbook: 5.2.5 IP filter between shared IP zones on a system)
(Cookbook: 5.2.6 IP filter between exclusive IP zones on a system)

4.1.7.6. Zones and limitations in the network

[dd] Zones have different limitations related to network configurations.

The following table shows the differences separated by zone type and IP instance type. Some functionalities can be configured and used within zones themselves (+), affect only the functionality of zones, or can be used by services within a zone, or cannot be used at all in local zones (-).

Functionality	***Global zone***	***Local zone with exclusive IP instance***	***Local zone with shared IP instance***
DHCP client	+	+	-
DHCP server	+	+	-
ifconfig(1M)	+	+	-
IPfilter	+	+	Effect on zone (filter)
IPMP	+	+	Usable in zone (redundancy)
IPQoS	+	Effect on zone (bandwidth)	Effect on zone (bandwidth)
IPsec	+	+	Usable in zone (communication)
Multicast	+	+	-
ndd parameter	+	+	-
NFS server	+	-	-
RAW network access	+	+	-
Routing daemon	+	+	-
Routing settings	+	+	Effect on zone (routing)
snoop(1M)	+	+	-
Solaris network cache and accelerator (NCA)	+	-	-

Table 5: Zones and limitations in the network

4.1.8. Additional devices in zones

4.1.8.1. Configuration of devices

[ug] In principle, a local zone uses no physical devices. To use network interfaces exclusively in one zone, the zone has to be configured as an exclusive IP zone (4.1.7.4 Exclusive IP instance).

Disks or volumes can be inserted in a zone with *zonecfg* (5.1.12.6 Using a DVD drive in the local zone). To activate such a modified configuration, the zone has to be restarted. Devices can also be brought into a zone dynamically with *mknod* without a shutdown of a zone having(5.1.12.7 Dynamic configuration of devices).

Device removal is done by removal with *zonecfg* or, for dynamically added devices, by deleting the device node.

In principle, other devices such as for example serial interfaces can be passed on to local zones as well. The task of the administrator of the global zone or the person responsible for the configuration of the local zones is to coordinate use of the devices and to avoid double allocation or double usage from global zone and local zone.

4.1.8.2. Static configuration of devices

[ug] Devices for a zone are defined during zone configuration using *zonecfg* with the command `add device`.

Solaris then generates the device for the zone by means of a device node in the file system */dev* of the zone. The `/dev` directory of the zone is located in the `zonepath` of the zone under the subdirectory `dev`. Devices are deleted by deleting them in the zone configuration. At the next reboot of the zone , the device is removed from its `/dev` directory.

This is one of the differences between the global zone and the local zones, since in the local zone, the device node is located directly in the `/dev` directory. The global zone only contains a symbolic link in `/dev` to a device entry in `/devices`. All devices recognized by the system, with their device nodes, are located in `/devices`.

A zone is not able to create device nodes by itself in its `/dev` since the command *mknod(1M)* and the system call *mknod(2)* is forbidden by privileges in a local zone. Furthermore, a zone can itself only mount file systems with the option `nodevices` whereby the device entries in this file system are not usable.

The security of zones with respect to devices is based exactly on these measures because no zone can obtain access to devices that are not configured for it.

4.1.8.3. Dynamic configuration of devices

[ug] Rebooting the local zone is sometimes not desirable although new devices have to be added to, for example, to procure additional space for a database.

Then, new devices can be added dynamically to the zone by emulating the process of static device configuration.

First, running *ls -lL <device>* in the global zone determines which device node lies behind the device; the important outputs here are the `major` and `minor` numbers. The `major` number specifies the driver and the `minor` number is something like the serial number of the device.

Next, the device for a zone can be created with *mknod* by generating the corresponding node in the `/dev` tree of the target zone. For security reasons, this can only be done by the administrator of the global zone (since the local zones lack privileges after all).

If access to the device is to be revoked, the node can be deleted in the `/dev` of the zone. In doing so, care must be taken that programs or mounts that have opened the device still retain access (as usual in Unix). A mount should be removed before access is terminated.

If the device is to be used in the zone permanently, the corresponding entry should be made in the zone configuration as well (example in 5.1.12.7 Dynamic configuration of devices).

4.1.9. Separate name services in zones

[ug] Name services include among other things the hosts database and the userids (*passwd*, *shadow*) and are configured with the file */etc/nsswitch.conf,* which exists separately in each local zone. Name services are therefore defined in local zones independent of global zones. The most important aspects thereto are covered in this section.

If one adopts the recommendation stated in this document that no applications should run in the global zone, then the global zone also does not need to be integrated into NIS or LDAP. This further limits access from the outside and reduces the dependency of the global zone from other computers (name services server).

4.1.9.1. hosts database

[ug] Computers that should be addressable by name must be recorded here. No automatic copy of */etc/hosts* from the global zone takes place when the zone is installed (completely in the sense that a separate OS environment exists in the local zone). It is of course a better alternative to use a name service such as NIS, DNS or LDAP. In an automatic installation, this can be set up via a sysidcfg file.

4.1.9.2. User database (passwd, shadow, user_attr)

[ug] User settings in local zones can be complemented by a name service as with a separate computer. Care should be taken that user names can be dissimilar in different zones; in particular in monitoring from the global zone (with ps) the names configured in the global zone are displayed. A copy of files from the global zone is not recommended, a name service such as NIS or LDAP is more suitable.

4.1.9.3. Services

[ug] The */etc/services* or the corresponding name service must also be adjusted to the applications running in the zone.

4.1.9.4. Projects

[ug] To locally run resource management using a Fair Share Scheduler, or extended accounting, in a local zone, the corresponding name service database in */etc/project* or the corresponding name service in the zone must be adjusted.

4.2. Paradigms

Paradigms are design rules for the construction of zones. Depending on the application, a decision must be made which one of them should be applied.

4.2.1. Delegation of admin privileges to the application department

[ug] Administration of an application can be delegated to the department responsible for the application. Through zone isolation, the root administrator can influence only resources that are assigned to the zone. This also applies to other defined privileged users in the zone (see process privileges, `ppriv`).

- If a zone is assigned to a shared IP instance, the network can only be configured in the global zone.
- If a zone had an exclusive IP instance assigned to it, the administrator of this zone can undertake the network configuration for the zone independently.
- File systems can be specified from the global zone (`zonecfg add fs`).
- File system administration can be handed over to the administrator of the local zone (`zonecfg add device`).

4.2.2. Applications in local zones only

[ug] If local zones are used for applications, it is recommended not to use the global zone for applications as well.

- Only then it is possible for computer hardware administration to remain purely with the platform administrators of the global zone.
- Platform administrators are the administrators who administer the global zone. They have access to the hardware (cabinet, power cable, hard drives) and perform the Solaris installation in the global zone. The platform administrator is also responsible for kernel patching and rebooting the entire system.
- It is not necessary to give the application admin access to the global zone.
- If the application admin needs root access, he/she can receive the root password for the local zone of his/her application. He/she must then, however, assume responsibility for the availability of the application, in consultation with operations.
- Requests for disk space are submitted through platform administration who can assign resources to the corresponding local zone, following approval by storage administration (if separate).
- For network configuration, a distinction must be made between a shared and an exclusive IP instance. Contrary to a shared IP instance, the administrator of a zone with exclusive IP instance can undertake network administration himself.

In the global zone, only system-oriented applications are installed that are required for management, monitoring or backup/restore purposes. To increase availability, cluster software can be used as well.

Advantages:

- Responsibilities for system, storage and application can be distinctly separated.
- Root-user access to the basic system is restricted to system administration. This results in improved stability.

Disadvantages:

- Some applications are not yet explicitly released for use in zones. Usually, the applications work in zones but they have not yet been certified by the manufacturer of the application.

4.2.3. One application per zone

[ug] Another paradigm is to always install one application per zone.
A zone has very little overhead; basically, only the application processes are separated from the rest of the system by tagging them with the zone ID. This is implemented by the zone technology.
The profits gained by this decision are very high:

- The administrator of the application can receive the password for root in the local zone without being in a position to jeopardize the operation of the entire computer in case of error.
- If many applications are consolidated, the number of users with root privileges is reduced.
- If you would like to install the application automatically, it is much easier since you know for sure that there is no other application that has modified the system.
- Dependencies/malfunctions between applications in the file system and/or configuration files are omitted completely, which results in safer operation.

4.2.4. Clustered containers

[tf/du] Clustered containers can be considered and configured as virtual nodes or as resources. By means of containers as virtual nodes, virtual clusters, so-called Solaris container clusters, can be implemented since Sun Cluster 3.2 1/09. More information is provided in the following chapter.

In a cluster, operation as a black box container or in fully formed resource topologies is possible. In a black box container, the applications run are configured in the container only. The cluster does not know them; they are controlled within the container by Solaris only. This leads to particularly simple cluster configurations.

For a fully formed resource topology, each application is run under cluster control. Each application is thus integrated into the cluster as a resource and is started and monitored by it in the correct sequence. For fully formed resource topologies in containers, the HA container agent, or containers as "virtual nodes", have been available since Sun Cluster 3.2. For a container as a virtual node, applications are moved among containers bundled into resource groups. Several resource groups can run in one container and pan independently of one another.

Both concepts, HA containers, and containers as virtual nodes, have their own strengths. Helpful criteria for selecting the right concept for the respective use case can be found in "Sun Cluster Concepts Guide for Solaris OS“: http://docs.sun.com/app/docs/doc/819-2969/gcbkf?a=view

In cluster topology, applications in a container can be brought under cluster control. If containers are run as virtual nodes, standard agents or self-written agents are used for application control. If containers are run under the control of the HA container agent, selected standard agents, or the shell script or SMF component of the Sun Cluster Container Agent, can be used. When using the HA container agent, hybrids between black box and fully formed resource topologies are permitted at any time. For virtual node containers, a fully formed resource topology is required. Clusters can be run in active-passive configuration or in active-active configuration.

- Active-passive means that one node serves the configured containers or applications and the second node waits for the first one to fail.
- Active-active means that each node serves containers or applications. If the capacity of each node is insufficient for all containers or applications, reduced performance must be accepted if a node fails.

If the requirement consists in achieving high availability in a data center, or high availability between two data centers, Sun Cluster is sufficient for scheduled and emergency relocation of containers or applications among computers. The maximum distance between two cluster nodes used to be set to 400 km for Sun Cluster and required the application of certified DWDMs (Dense Wave Division Multiplexers). Nowadays, evidence of maximum latency of 15 ms (one-way) and a maximum bit error rate (BER) of 10^{-10} is considered sufficient.

If these requirements are met it unfortunately does not yet mean that the performance requirements of the application are also met. That is to say, if the latency of the mirroring or replication technology involved does not meet the performance requirements, Sun Cluster Geographic Edition can be used instead.

Sun Cluster Geographic Edition assumes clusters running in several data centers. The storage used is replicated by means of suitable techniques from data center A to data center B. Currently, Sun StorEdge Availability Suite (AVS) as a host-based replication, Hitachi Truecopy and EMC SRDF as a controller-based replication and, with Sun Cluster 3.2 1/09, Oracle DataGuard as an application-based replication are integrated. Sun Cluster Geographic Edition allows you to move the containers/services from data center A to data center B. If a data center fails, Sun Cluster Geographic Edition suggests data center panning. Panning is performed following confirmation by an

administrator. With the software products described here, the requirements with respect to visualization and flexibilization of containers right up to disaster recovery concepts can be covered completely.

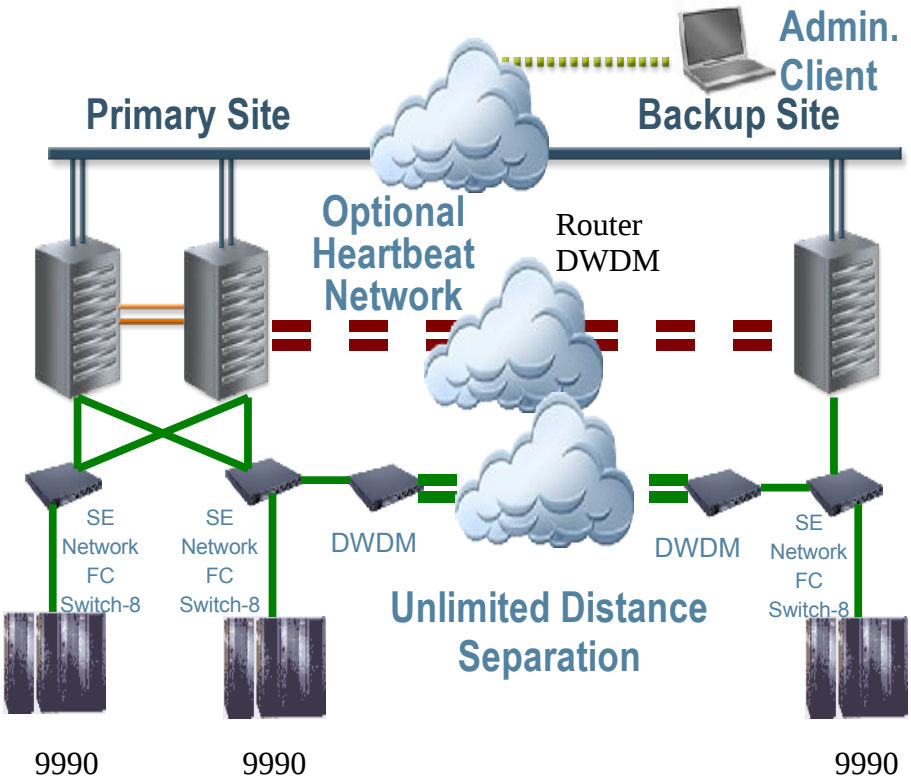


Figure 25: [tf/du] Typical configuration: Sun Cluster Geographic Edition

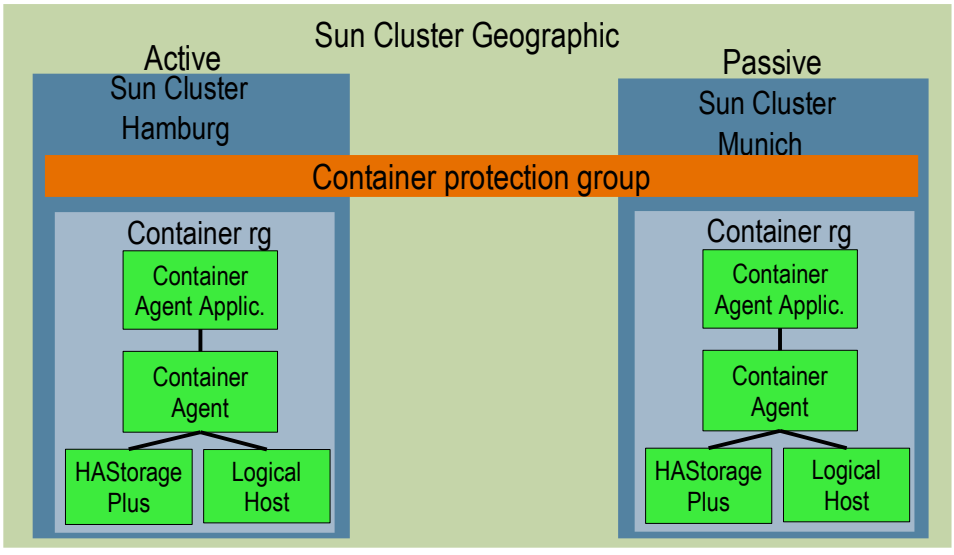


Figure 26: [tf/du] Resource groups and protection group topology

The hardware architecture shown above consists of a dual-node cluster made of SF6900 servers in the primary data center and a single-node cluster in the alternative data center designated for disaster recovery. The storage systems used are Sun StorageTek 9990 systems and data replication is done via Truecopy. Control is via Sun Cluster Geographic Edition. If disaster recovery is not desired, each data center can also be operated independently. In the resource group topology shown above, the active cluster manages a container. The passive cluster is used for disaster recovery purposes. The clusters form a partnership and manage the resource groups in a so-called protection group.

4.2.5. Solaris Container Cluster

[hs] One of the essential properties of containers is the possibility to delegate administrative tasks to the administrator or the user of one or more containers. If high availability of a container is ensured by the use of the HA Solaris container agent, this is not visible to the administration of the container concerned - which is usually the desired situation. If, however, the container-related administration of cluster resources is to be delegated to the container administrator as well, a container cluster can be used.

A container cluster consists of several containers, that are consolidated into one virtual cluster. The resources required for running the container and the container cluster are provided by the administrator of the global zone. Use of these resources, among other things setting up resource groups and their resources, is the responsibility of the administrator of the container cluster.

The administration of resources within a container cluster is the same as administration in the global zone. Exceptions are Quorum and Cluster Interconnect, which are administered exclusively in the global zone. Thereby, the administrator of the container cluster can focus on highly available services and their integration into the cluster.

The principal tasks involved in administering container clusters consist in the identification and allocation of resources required from the global zone to the local zones that form a container cluster. They include:

- IP addresses (container clusters can only work with IP-type shared)
- CPUs
- Main memory
- File systems or raw devices

Reproducible, systematic naming for resources is a great convenience for error-free administration of complex container cluster environments.

4.3. Configuration and administration

4.3.1. Manual configuration of zones with zonecfg

[ug] The command *zonecfg* is used to configure a zone; see the example in the Cookbook.

The configuration merely describes the directory (*zonecfg: zonepath*) in which the files for the zone are to be placed and how system directories of a local zone are created from the directory contents from the global zone (copy/loopback mount). Additional options are configuration of network addresses, file systems, devices, etc.

Each zone, once configured, can also be used as a template for additional zones. This cloning of configurations considerably simplifies the creation of zones that have similar configurations.

We advise against configuring zones by editing the XML-files under */etc/zones* since this procedure is very error-prone and is therefore not supported.

4.3.2. Manual installation of zones with zoneadm

[ug] Once a zone has been configured, the next step consists of its installation with *zoneadm -z <zone> install* (see Cookbook). In the process, all packages (Solaris pkg) that do not belong to the Solaris kernel are installed into the zone, while the files that are available from the global zone via loopback mount are not copied. During this process, the post-install scripts of the packages for the zone are also executed. In the end, the zone has its own package database (*/var/sadm/install/**). The packages of the global zone are used as the source for the packages to be installed on the local zone.

After this , the zone can be booted.

4.3.3. Manual uninstallation of zones with zoneadm

[ug] If a zone is not needed anymore, it can be uninstalled with

zoneadm -z <zone> uninstall. This will remove all files in the zone path, even those that were created after installation.

The configuration of the zone will remain.

4.3.4. Manual removal of a configured zone with zonecfg

[ug] To remove a zone, uninstallation should be performed first. Next, the configuration of the zone can be removed with the command *zoneadm -z zone destroy*.

4.3.5. Duplication of an installed zone

[ug] Zones can be duplicated rapidly if the *inherit-pkg-dir* configurations are identical. Then, a zone can be created as a template and quickly duplicated for each zone by copying the contents.

Since Solaris 10 11/06, this can be done with the command *zoneadm clone .* Additionally to copying the directories some tests are performed.

Starting with Solaris 10 5/09 the cloning for a zone on ZFS, the clone functionality of ZFS is used automatically, which considerably accelerates zone duplication.

(5.3.7 Duplicating zones with zoneadm clone or

5.3.8 Duplicating zones with zoneadm detach/attach and zfs clone).

4.3.6. Standardized creation of zones

[ug] Zones can be created in the following manner:

- with the command *zonecfg* and then with *zoneadm* to install the zone
- with the container manager in the Sun Management Center (SunMC)
- with Webmin, a free management tool included with Solaris 10/OpenSolaris (Solaris zones module: latest version *zones.wbm.gz* downloadable at http://www.webmin.com/standard.html)
- by scripts that call *zonecfg* and *zoneadm*

The container manager and Webmin are suitable for admins that rarely set up zones.

With *zonecfg*, zones can be created according to site standards by setting up zones as templates. A zone can then be used as a template with the *zonecfg* subcommand *create -t <template>* .

As a general rule, some guidelines are specified locally, for example:

- Which file systems are to be inherited from the global zone (*inherit-pkg-dir*).
- File systems with software that is to be able to be used by each zone.
- Shared directories from the global zone.
- Whether the zone is to be automatically started when the computer is rebooted.
- The resource pool and other resource settings for the zone.

These standard settings can be stored with a template zone and used again. However, the IP address and the zone directory have to be always be configured with *zonecfg*; also, the computer name (*/etc/nodename*), the time zone and the name service must be configured when the zone is booted for the first time. For the automated creation of zones, the N1 Service Provisioning System (N1SPS) can be used.

4.3.7. Automatic configuration of zones by script

[dd] Zones can be created and configured automatically by a shell script. The creation of zones follows previously specified policies. The following steps must be performed in the script:

- The *zonecfg* command is called by a script that sets the IP address, zone directory, file systems, resource pool according to policies (Cookbook), for example according to:
  - IP address is the address of the IP name of *z-<zone>*
- The zone directory is set e.g. to */export/zone2/<zone>* for this class of zones.
- The settings requested by the zone immediately after booting can be carried out with a file named *sysidcfg* which is placed into the */etc* directory of the zone prior to booting the zone (*<zonepath>/root/etc*). See the Cookbook for more details.

4.3.8. Automated provisioning of services

[dd] With the automatic configuration of zones per script and the stipulation of applying exactly one service per zone, the process of service provisioning can be automated as well. The N1 Service Provisioning System (N1SPS) http://www.sun.com/software/products/service_provisioning/index.xml or a separate script can be used. If provisioning of the service is well automated, this type of service can be applied to many zones very quickly. The following steps are required:

- Software installation
- Creation or copying of application data
- Modification of the zone in order to cover the software requirements
  - User accounts, environmental settings, log files
- Organize start of service
  - Install SMF service or rc* scripts

Corresponding steps with respect to exporting the service and the service data must be provided to remove or to move a service into another zone.

4.3.9. Installation and administration of a branded zone

[ug] Branded zones are supported since Solaris 10 8/07. Solaris for x86 contains the brand *lx* which allows a Linux environment to be made available in a zone. This type of zone can be created as follows:

- Zone configuration:
  Just like other zones, the zone must be configured with *zonecfg*. However, the brand must be set to *lx*. This can be done by using the template *SUNWlx* or by setting up the zone manually from the blank zone (*create -b*) and setting the attribute brand to *lx*.
- During installation, the file tree of a suitable Linux system which is copied to the corresponding location in zone path can be used directly in the zone.
- It is also possible to install the zone directly from the supported distributions (CD/DVD).

The migration of Linux zones (*zoneadm detach/attach*) is simplified since the package and patch status for branded zones are not examined.

Duplication (*zoneadm ... clone*) of Linux zones located on ZFS is accelerated considerably by the automatic use of *zfs clone*.

4.4. Lifecycle management

4.4.1. Patching a system with local zones

[dd/ug] In a Solaris system with native zones, the local zones always have the same patch status as in the global zone. This is independent of whether the local zone is a sparse root zone or a whole root zone. Therefore, patches on this type of system are installed in the global zone and thereby automatically in all local zones.

- Contrary to procedures prior to Solaris 10 10/08, zones can be patched, while not booted.
- The patching process requires that the zones to be patched are in "installed" status.
- The configured file systems (per `zonecfg` and `/etc/vfstab` of the local zone) must be mountable through the global zone.
- For patching, the configured file systems of a zone are mounted, the patch is installed and the file systems are unmounted again. This process runs separately for each patch.

When installing a patch cluster, this procedure is performed successively for all patches and zones. Therefore, patching can take several hours or days if there are many patches to be installed (e.g. patch clusters) and many zones present.

There are a variety of considerations as to how the expenditure of time to patch many zones and thus the downtime of zones can be reduced:

1. The patch utilities have been optimized such that simultaneous patching of many local zones is possible. These utilities were made available in June 2009 by patches 119254-66 (SPARC) and 119255-66 (x86).
   (see also http://blogs.sun.com/patch/entry/zones_parallel_patching_feature_now)
   This functionality has been integrated into Solaris 10 10/09.

   The time gained by using these new utilities is enormous. Jeff Victor has done some research on this in his blog. http://blogs.sun.com/JeffV/entry/patching_zones_goes_zoomUtilities
   To use the mechanism, it is mandatory that the readme.txt of the patch is observed as the number of zones to be patched in parallel must be configured first (see /etc/patch/pdo.conf). From it follows that the number of zones to be patched in parallel depends on the number of CPUs available in the system.(max. number of parallel zones to be patched = number CPU * 1.5) **Older tools for simultaneous patching of zones should not be used anymore.**
2. So as not to affect the currently running system by patching procedures, a separate patch server can also be used. In this procedure the current zones are duplicated and the copies of the zones are moved to a patch server via `zoneadm detach` / `zoneadm attach`. The patches are then installed on this server. Next, the zones are moved to a server that already contains the current patches in the global zone.

Comments:

- In general, the goal is to ensure that local zones contain the same patch status of the operating system as the global zone. This is mandatory for system libraries.
- Using a different patch status in zones can be considered for the application software used only.
- If patches are to be installed in a certain zone only, e.g. in whole root zones, to test patches, then patches can be installed in a zone with `patchadd -G`.

The following URL contains further information and recommendations on the topic of patching: http://www.sun.com/bigadmin/hubs/documentation/howto/patch.jsp

Further information on the topic of patching can also be found here (http://blogs.sun.com/patch/) and here (http://www.sun.com/bigadmin/features/articles/patch_management.jsp).

4.4.2. Patching with live upgrade

[ug/dd] Starting with Solaris 10 8/07, live upgrade can also be applied to patching on zones if the zonepath is located on ZFS. (See also Cookbook: 5.3.11 Using live upgrade to patch a system with local zones). To do so, a copy of the active Solaris boot environment with all its local zones installed is created through `lucreate(1M)`. Then the patches are installed into this copy using `luupgrade -t`. A `luactivate(1M)` activates the boot environment, which is active after the next reboot. By using live upgrade, the downtime of services during patching can be drastically reduced since the patch process runs parallel to the system in production. Downtime is required only to activate the updated boot environment by rebooting. It must be taken into account that live upgrade generates an I/O-load that must in addition be made available by the server and the disk system during the runtime of `lucreate` and `luupgrade`.

4.4.3. Patching with upgrade server

[ug] A zone is transported from the production computer to a so-called upgrade server (`zoneadm detach` and `zoneadm attach`) that has the same version as the production server. On this upgrade server, the upgrade or the installation of patches is then carried out. Subsequently, the zone will have the new patch version. Variants:

- The upgrade server can then serve as the new production computer.
- A cluster (Sun cluster) can also be upgraded by installing the patches in the zones on an upgrade server, then installing the patches in a cluster node and then moving the zones there. Next, the first cluster node can be updated.
- If the zones are supposed to continue running, only one copy of the zone is moved and the application is not started during the upgrade.

Thereby, the total run time of the upgrade, which depends on the number of patches and the number of zones, is not so important anymore. Production continues to run during the upgrade process.

4.4.4. Patching with zoneadm attach -u

[ug/dd] With Solaris 10 10/08, the command `zoneadm attach -u` is available with which a zone can be updated to the status of the new target system during `zoneadm attach`. But this does not provide a new upgrade option.

However, it is a precondition for `zoneadm attach -u` that the patch history is quite identical and that the target system does not have a package containing an older version because it is not possible to downgrade a package. This also holds true for a package's old patch status. A `zoneadm attach -u` will nor work between arbitrary systems. Systems should be administered accordingly for this purpose, such as for example in a cluster. Patches, installed by this method, can not be backed out.

`zoneadm attach -u` installs only patches of packages, that have the option `SUNW_PKG_ALLZONES=true` set. These are typically OS-Packages, but not that of applications. If `zoneadm attach -u` will be used for patching, it is important to know that patches of applications will maybe not or not completely be installed. These patches has to be post-installed to stay in sync with the patch- and package-database.

With this knowledge and extensive evaluation `zoneadm attach -u` could be a very powerful functionality for patching. Especially in time-critical situations is this method faster than parallel patching with `patchadd`.

To summarize `zoneadm attach -u` will only work under certain preconditions and does not help to apply all patches. It is therefore not usable as a normal patch/upgrade process.

4.4.5. Moving zones between architectures (sun4u/sun4v)

[ug] `zoneadm attach -u` allows moving zones between the two current SPARC hardware architectures. The precondition for this is, however, that the same packages are installed and that the patch history is as identical as possible. It is advisable to prepare such a zone move by patching the systems always in parallel.

4.4.6. Re-installation and service provisioning instead of patching

[dd] Patching of zones can force zones into single user mode, when system patches are applied. Zone patching can therefore lead into halting the services provided. This service interruption can vary in length depending on the type of the patches, the number of zones and the number of patches to be installed. To shorten this period and to achieve predictability of service downtime, re-installation can be chosen over individual patching.

Zones can be created rapidly in almost any number. These zone installations are considerably quicker than the re-installation of a computer or the individual patching of zones. It should therefore be generally considered as a datacenter method to carry out re-installation of zones instead of patch or upgrade. Newly created zones after all automatically have the current software or patch status of the global zone after they have been created. Subsequently, the application data and, the application are moved into the new zone. For this time the zone is not in production.

One of the prerequisites for this procedure is the fact that the application, the configuration and the data can be moved among zones. The following things must be taken into account:

- Binary programs are installed in zones using the standard procedure for creating zones.
- Service configurations and data are stored in separate directories or file systems in the zone.
- Other modifications to the zones are recorded or can be detected and extracted automatically (e.g. with *bart(1m)*)
- To "move" a zone, the service is "moved". This is done by copying the configuration and the service data (if necessary by relocating the file system) as well as by customizing the zone created to the service requirement. This can be done by extracting, transporting and unpacking the modification from the original zone to the new zone.

4.4.7. Backup and recovery of zones

[dd] A Backup of Zones can be created by using backup tools from the global zone. The backup can contain the zone's application data or save them separately. The following things are required for zone backups:

- Backup of zone configuration with
  zonecfg -z <zone1> export -f <zone1>.zonecfg
  or directly of the file */etc/zones/<zone>.xml*
- If necessary, backup of the associated line in */etc/zones/index*
- Backup of the directory tree e.g. /zones/zone1 using backup tools

For recovery, the steps must be carried out in reverse.

- create */zones/zone1*
- *chmod 700 /zones/zone1* (standard for zone directories)
- If necessary, prepare submounts, if e.g. */zones/zone1/root/var* should be a separate file system
- Install backup
- Where applicable, adjust *<zone1>.zonecfg* (new path, other mounting points)
- Create the zone configuration with *zonecfg -z <zone1> -f <zone1>.zonecfg*
- Use `zoneadm attach` to attach the zone

It must be taken into account when doing a backup of the zone directory that sparse root zones contain *inherit-pkg-dir*. To back up directories such as e.g. /usr, /sbin per zone as well is certainly only useful in the fewest of cases. That is, information must be provided in the backup configuration whether and how *inherit-pkg-dir* should be co-backed up.

The Sun StorEdge Enterprise Backup Software (EBS) can e.g. be controlled via .nsr files in directories such that such directories are not backed up as well.

Veritas Netbackup (NBU) has a central configuration available. That is, in this case, corresponding configurations must be installed centrally using global rules. If zone migration and backup/restore from the global zone is scheduled for later, this configuration must accordingly be constructed in a cross-system manner.

Use of a backup client in the global zone for several local zones allows licensing fees to be saved. To back up special application data (e.g. databases), the backup client or the backup module must, if need be, run directly in the local zone (e.g. backup of Oracle with RMAN).

4.4.8. Backup of zones with ZFS

[ug] Starting with Solaris 10 10/08, zones on ZFS are officially supported. This considerably simplifies the backup of zones. Zones are installed on ZFS and a zone backup can be implemented via a ZFS file system snapshot.

To do so, the zone to be backed up is shut down, a snapshot of the file system where the zone is located is made (with *zfs snapshot <zfs-filesystem@snapshot>)* and the zone is restarted. The snapshot can then be backed up in parallel with zone operation.

The snapshot can also be used for upgrade purposes on another system by means of *zoneadm attach*, by transporting the contents there using *zfs send* and *zfs receive*.

4.4.9. Migration of a zone to another system

[dd] Since Solaris 10 11/06, the functionality of migration of installed local zones from one system to another is possible. This migration is done using stopped zones.

To migrate a zone, the zone is stopped and the entire zone configuration and the list of installed patches and packages is filed in an xml file in the zone root directory of the zone using *zoneadm detach*. The directory can be moved to a new system, and the zone is brought to executable status again through *zoneadm attach*. The installed packages and patches as well as the architecture (e.g. *sun4u* or *sun4v* or *i386*) must be identical on both systems. Alternatively, a running zone can be copied (*zoneadm clone*) and the clone migrated to another system via *detach* / *attach*. (5.3.5 Zone migration among systems)

4.4.10. Moving a zone within a system

[ug] In order to move a zone within a system to another file system, the command *zoneadm move* can be used. To do so, the zone must first be shutdown so the zone path can be moved to another position in the file system. The files are moved or, in the event of file system change, copied, and the zone configuration is adjusted.

Zones can be migrated in particular from UFS to ZFS (and the other way round).

(5.3.6 Zone migration within a system)

4.5. Management and monitoring

4.5.1. Using boot arguments in zones

[dd] When booted, zones write to their own console and normally start up directly into multi-user mode.

To test the start-up of SMF services in zones, however, it can also be necessary to get more extensive starting protocols. For this case, zones can be started with boot arguments (e.g. *-m verbose*), thus providing more detailed information on the booting process.

To perform software installations or to perform patching in zones, it is often useful to boot the zone into single-user mode. This can also be implemented by means of boot arguments.

These boot arguments are specified during zones configuration with *zonecfg* or during zones boot with *zoneadm*. One example of this is shown in (5.3.2 Boot arguments in zones).

4.5.2. Consolidating log information of zones

[dd] The use of zones as a runtime environment for services leads to an increase in the number of operating system environments that are part of an architecture. Although this is intended for reasons of enclosure or modularization of applications, it can lead to another problem.

Since each zone and its applications maintain their own log files, these are now present in larger quantities. As a result, the analysis of log files for the entire architecture as a whole can be achieved only with increased effort.

Log information can be consolidated by NFS, with a syslog server, by file synchronization or with application-specific means.

1. To summarize log information by NFS, the following procedure can be used:
   1. Log files are written directly in the local file system via the applications in the local zones (if applicable, observe or organize the rotation of log files).
   2. The relevant log files forming all local zones of a system are collected centrally in the global zone. They can be collected on an NFS directory that is mounted in the global zone only. The transparent remote access to zone data can be achieved by means of the zone concept (e.g. access to `/<zonepath>/root/var/log/logfile.log`).
   3. All log files belonging to an architecture can be accessed where one or all NFS directories of all global zones are available.
2. Log files are first copied to a local file system like in 1. and then transferred to a remote system by means of `rcp`, `rdist`, `scp` or `ssh`. This variant can have security risks due to the use of the copy mechanism.
3. The `syslogd` can send log information directly to a remote system which can be the global zone or a central system in the network.

4.5.3. Monitoring zone workload

[ug] With the `prstat` command (since Solaris 8), processes with the highest workload can be viewed similar to the command `top`, which is well-known on other platforms. In Solaris 10, the command `prstat` has been extended by the option `-Z`, which allows the user to see a summary display of the workload for each zone (even the global zone). Thus, zone status is easy to monitor.

4.5.4. Extended accounting with zones

[ug] Extended accounting was introduced with Solaris 9 as a complement to the traditional Unix accounting. In extended accounting the data fields to be recorded can be selected from a superset of fields. Solaris 10 also provides the name of the zone as an additional optional data field.

Therefore, extended accounting can be configured in the global zone such that the zone name is written together with each accounting data set. The data (e.g. CPU time used) can be summarized separately according to zones and can be fed into capacity planning or accounting.

4.5.5. Auditing operations in the zone

[dd] Auditing can be used to monitor system activities. A system audit takes place in the global zone. The audit can also be configured to monitor activities in a local zone. Additionally the administrator of a zone is able to monitor a zone's user processes. Auditing in local zones cannot monitor kernel activities but user activities within the zones.

4.5.6. DTrace of processes within a zone

[dd/ug] DTrace can be used to examine processes in zones. To do so, DTrace scripts can be extended by the variable *zonename* in order to e.g. only trace system calls of a zone

```
global# dtrace -n 'syscall:::/zonename=="sparse"/
       {@[probefunc]=count()}'
```

or to measure the I/O of zones.

```
global# dtrace -n io:::start'{@[zonename] = count()}'
```

Starting with Solaris 10 11/06, DTrace can also be applied to processes within the own local zone. The privileges *dtrace_proc* and *dtrace_user* need to be assigned to the zone (*zonecfg:set limitpriv=default,dtrace_proc,dtrace_user*). Kernel tracing, and the tracing of processes in the global zone, remains excluded. DTrace within a zone is of interest especially for those use cases where a zone is used for software development or where the administration of a zone was delegated. In this case, the local zone administrator is able to analyze independent of the platforms administrator.

4.6. Resource management

4.6.1. Types of resource management

[dd] There are 3 different types of resource management in all:

- Fair resources: Here, all resources are distributed fairly among all requesters and according to the defined rules. Example: Fair share scheduler for CPU resources
- Capped resources: Resources can be seen by all requesters. As the upper limit of a resource is the capping value specified. Unused resources can also be used by others. Example: CPU capping
- Exclusive resources: Resources are assigned exclusively to the requester and can be used by him only. Unused resource shares are not available to others. Example: resource pools with processor sets

4.6.2. CPU resources

[ug] CPU resources can be limited on several different levels in Solaris 10:

- Capping of CPU time for a zone
- Resource pools with processor set (global zone only)
- Fair share scheduler between zones in a resource pool (global zone only)
- Fair share scheduler between projects in a zone

4.6.2.1. Capping of CPU time for a zone

[dd] For this purpose, a CPU capping value is assigned to a zone in the zone configuration. Parts of a CPU can be specified here, e.g. 0.5 CPUs or 1.57 CPUs. The usable CPU time of a zone is capped at this contingent. This allows finely granulated CPU assignment on a number of zones even if less than a whole CPU is to be assigned to a zone. In addition, the zone will not use more CPU resources, although more CPU time would be available (e.g. at times when the total system is underutilized).

CPU caps must be used where finely granulated CPU rations are required and an upper limit must be observed. The setting using CPU caps is exact to 1/100 CPUs.

(Cookbook: 5.5.2 Limiting the CPU usage of a zone (CPU capping))

4.6.2.2. General resource pools

[ug] Resource pools are defined in the global zone to which a processor set can be assigned (see commands *poolcfg* and *pooladm*).

- A processor set contains an upper and a lower limit for the number of processors. This can be used for dynamic allocation of processors using *poold*.
- These definitions are static and withstand rebooting.
- A dynamic change is also possible with the *prctl* command.
- A zone (or a project) can be assigned to such a resource pool. This is done with *zonecfg: set pool=<name>* or dynamically with *poolbind*. All processes of the zone (or of the project) then run in this resource pool and thus on the CPUs of the processor set defined for the resource pool.
- If no processor set is assigned, the processes in the resource pool run on the general processor set (*pset_default*) which contains all processors by default and where the system activities take place as well.

Resource pools make sense, if more than one zone (if necessary with FSS) are to run in one resource pool (and on one processor set).

Resource pools are also appropriate, if individual projects from the global zone are to run in a resource pool in addition to the local zones. It is, however, recommended not to run any applications in the global zone due to higher complexity.

4.6.2.3. Fair share scheduler (FSS)

[ug] When multiple zones are running in one resource pool, then the distribution of CPU time among these zones is configurable. This configuration is active, when the workload of the processor set approaches 100% (full load). The process priorities will then be modified by the FSS to achieve the configured CPU share. However, so long as the utilization of the CPU workload stays below 100%, the FSS does not intervene.

- For configuration, the fair share scheduler must be activated in the resource pool.
- The value of *zone.cpu-shares* must be set in the zone by *zonecfg: add rctl.*
- From Solaris 10 8/07, the number of shares can be adjusted more easily with *zonecfg: set cpu-shares=<number>*.
- During full load (100% CPU workload in the resource pool), all the share values of the active zones are added up and the priorities of processes in these zones are modified such that the share in CPU power corresponds to the ratio *zone.cpu-shares*/<sum-of-shares>.
- The fair share scheduler can also be used in-between projects and mixed between zones and projects in a resource pool.

Comments:

- The project *cpu-caps* also allows to firmly assign the processors of a resource pool under partial load
- Since the fair share scheduler can change the runtime behavior of the operating system, it is recommended to always, if possible, set up a separate resource pool with processor set where the FSS is to be used.

4.6.2.4. Fair share scheduler in a zone

[ug] If two or more applications within a zone are to receive differing amounts of CPU capacity, the FSS can also be used in the zone. It is not possible to configure Processor sets within a zone since the zones' access to processors is restricted.

- The administrator of a zone can configure the resource pools (*pooladm, poolcfg, poolstat*).
- Projects can be configured in the zone (*projects, projadd, projmod, projdel, ...*)
- The fair share scheduler must be activated in the resource pool of the global zone.

4.6.2.5. Dynamic resource pools

[ug] From Solaris 10 8/07 on, the creation of processor sets has been simplified for the following standard case: Resource pool with one processor set for one zone. Configuration takes place in the *zonecfg* command with *add dedicated-cpu.*

- During the start of the zone, the system generates a temporary resource pool with processor set that corresponds to the configuration, and binds the zone to this resource pool.
- The parameters can be changed dynamically just like for general resource pools.
- When the zone is stopped, the resource pool is deleted and the CPUs are released again.
- Since this setting occurs in the zone configuration, it is taken along automatically when a zone gets moved to a different system. For general resource pools, this configuration must be transferred manually.
- Dynamic resource pools are especially suitable on systems with many CPUs (large computers, CMT systems).

(Cookbook: *5.5.9 Dynamic resource pools*)

4.6.2.6. Lightweight processes (LWP)

[ug] The limiting of lightweight processes is important for zones. Without such a limit, a denial-of-service attack on the system from one zone can occur. A lightweight process is required to allow a thread to run. If a process runs amok within a zone (e.g. a self-starting script that continues to generate new processes over and over), then the maximum number of LWP's could be reached on the system and/or the system could run slowly.

- In the zone configuration, the value of *zone.max-lwps* which limits the number of lightweight processes can be set with *zonecfg: add rctl.*
- From Solaris 10 8/07, the number of LWP's can be set more simply by *zonecfg: set max-lwps=<number>*.

4.6.3. Limiting memory resources

[ug] Memory usage by zones is calculated almost exactly (since Solaris 10 8/07). This is done in the following way: First, the set of all memory segments of the processes in the zone is determined. Shared segments appear only once in this set. Next, the memory usage of the segments is added up. This calculation shows the memory requirements of the zone almost precisely. The memory usage of the zone can be displayed with `prstat -Z`, with SWAP indicating the virtual memory used and RSS the physical memory used. This calculation also works for projects (for example in zones) with `prstat -J` . As soon as a limit has been configured, the status of the zones can be displayed with `rcapstat`.

The inaccuracy of the calculation above is found in the fact that for example the segment of the /usr/lib/libc.so library is used by the global zone and by all zones that inherit the /usr/lib tree. It is fully counted for each participating zone but in reality exists only once. So, the zone needs this space but shares it with others and should therefore only have a portion of it counted. This only pertains to shared libraries and program code. Essential memory segments are the data sectors of running programs and shared segments (e.g. of databases) that do not go beyond the scope of a zone.

(Cookbook: 5.5.11 Implementing memory resource management for zones)

4.6.3.1. Assessing memory requirements for global and local zones

[ug] The global zone meets the following framework conditions of a Solaris operating system:

- Solaris requires approx. 512 MB - 1 GB main memory, depending on the services activated
- Furthermore, 1/32 of the real main memory is kept free for the `free` list (`minfree` setting); the space also serves as disk cache.
- If much traditional I/O takes place with large files (`read()` and `write()` instead of `mmap()`), 12% (approx. 1/8) of the main memory is still required for the corresponding buffer (`segmap` setting).

The global zone without applications of a 32 GB system therefore requires about 2 GB; with lots of traditional I/O the global zone needs 6 GB of memory.

The usage of a local zone can be verified with `prstat -Z`. Without applications, it amounts to about 5 - 10 MB, depending on the services installed.

4.6.3.2. Limiting virtual memory

[ug] (From S10 8/07) The entire virtual memory is managed by the kernel. The amount of virtual memory the processes of a zone are able to use can be limited. This is done by setting the `swap` value is set to a certain quantity in the zone configuration using `zonecfg: add capped-memory`. This setting cannot be changed by the zone administrator.

As soon as this amount of memory is being used in the zone, applications in the zone are not able to to request any new memory anymore. The amount of virtual memory is thus hard-limited (capped). This is the setting by which the amount of memory a zone uses can be limited. This configuration should be used first before thinking of a physical memory limit.

4.6.3.3. Limiting a zone's physical memory requirement

[ug] The physical memory used by all zone processes can be limited by setting the value `physical` in the zone configuration to a certain quantity with `zonecfg: add capped-memory`. This setting can not be changed in the zone by the administrator (since S10 8/07).

The behavior of processes in the zone first remains the same. If a virtual memory page has not been used for a long time, it is swapped to the swap-space (disk). If a swapped memory page is used, it is swapped in through Solaris' paging mechanism and uses a physical memory page. When a zone with physical memory limitation boots up, the resource capping daemon (`rcapd`) is automatically configured and started in the global zone.

As soon as the `rcapd` determines that a zone uses more than the configured quantity of physical main memory, it will actively swap main memory pages of the zone until the limit is observed; the zone's activity is obstructed and the zone's performance deteriorates. This setting allows the user to limit the effect on other zones if there is an overall misconfiguration (e.g. database set too big). To limit the size of a zone, the virtual memory setting should be used.

4.6.3.4. Limiting locked memory

[ug] Real time programs and databases can establish the locking of virtual memory pages in the main memory. To do so, the programs require the privilege (`proc_lock_memory`) which must be configured for the zone. Databases in part use memory locking for shared segments to optimize performance (ISM – intimate shared memory). Nowadays, however, DISM (Dynamic ISM, e.g. with Oracle) is also used frequently, which will only establish memory locking when it becomes possible.

The amount of locked memory in a zone can be configured for a zone by setting the `locked` value to a certain amount with `zonecfg: add capped-memory`. The administrator of the local zone is not able to change this setting.

Locked memory enhances the performance of the application. But it is also a disadvantage because it is subtracted directly from the available main memory. It is not available for other purposes.. Whether a process requires locked memory can be learned from the documentation of the application or with the command `pmap -x <pid>`.

This setting should be made if a process within a zone requires locked memory in order to run, or to perform (DISM). Size limitation, however, should be done with the setting for virtual memory (`swap`).

4.6.4. Network limitation (IPQoS)

[dd] The IP traffic of a zone to an IP address can be limited with IPQoS. The command `ipqosconf(1M)` creates the file `/etc/inet/ipqosinit.conf` in the global zone which contains the configuration for IPQoS. This configuration is used to make only a specific network bandwidth available to certain zones.

4.6.5. IPC limits (Semaphore, shared memory, message queues)

[ug] IPC settings can be done by means of settings in the project since Solaris 9; this also applies of course to settings within a zone.

From Solaris 10 8/07, upper limits for these values can be set in the zone configuration. These values can be modified in the zone configuration or from the global zone. The administrator of the local zone is not able to change these values.

Adjustable parameters are:

- `max-msg-ids` maximum number of message queue IDs
- `max-sem-ids` maximum number of semaphore IDs
- `max-shm-ids` maximum number of shared memory IDs
- `max-shm-memory` maximum size of shared memory in the zone

With this, it can be ensured that the appropriate settings cannot be exceeded in the zone.

4.6.6. Privileges and resource management

[ug] In Solaris 10, privileged system calls are examined at a fine-granular level. Privileges that allow using these calls can be configured. This technology has been adopted from Trusted Solaris. For example, the authorization to mount a file system can be transferred to a user. This can be configured using role based access control (RBAC).

Local zones have only a subset of the privileges active compared to the global zone. This is the main protection mechanism to separate local zones. A root process in the local zone basically lacks the authorization to view processes outside the zone and to access all hardware.

Solaris 10 11/06 made it possible to assign additional privileges for the zone in the zone configuration with `zonecfg: set limitpriv=...` .

These privileges are, among others, DTrace, lock memory and network raw access. An exact listing of privileges in zones and their allocation can be found at:

http://docs.sun.com/app/docs/doc/817-1592/6mhahuou9?a=view

4.7. Solaris container navigator

[dd] The following segment navigates through the considerations required prior to the application of Solaris containers. These include both the examination of applications for compatibility with Solaris containers as well as the selection of various configuration options, and aspects of container installation.

This navigator is based both on Best Practices as well as on experience and does not claim to be complete. Tips and amendments are welcome at any time.

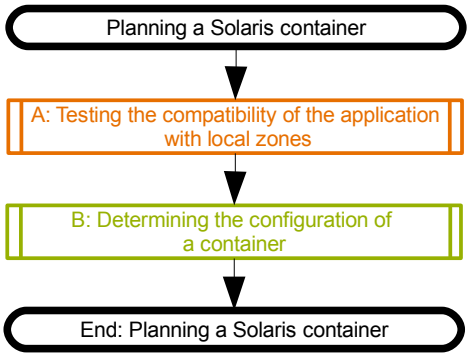


Figure 27: [dd] Planning a Solaris container

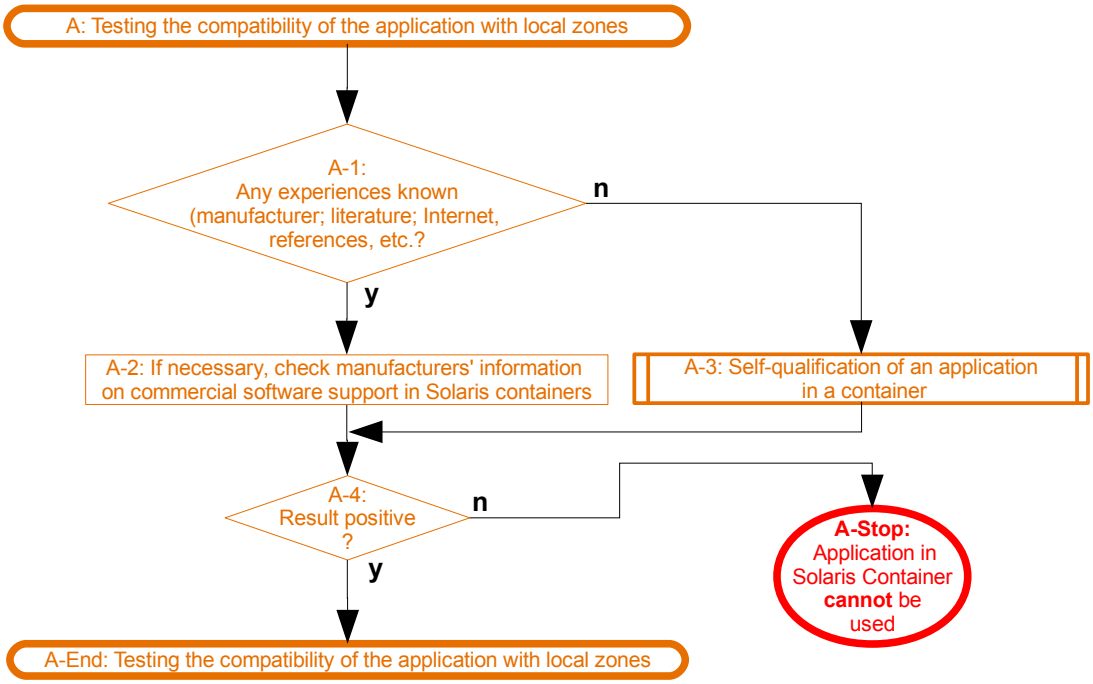


Figure 28: [dd] Testing the executability of the application in a container

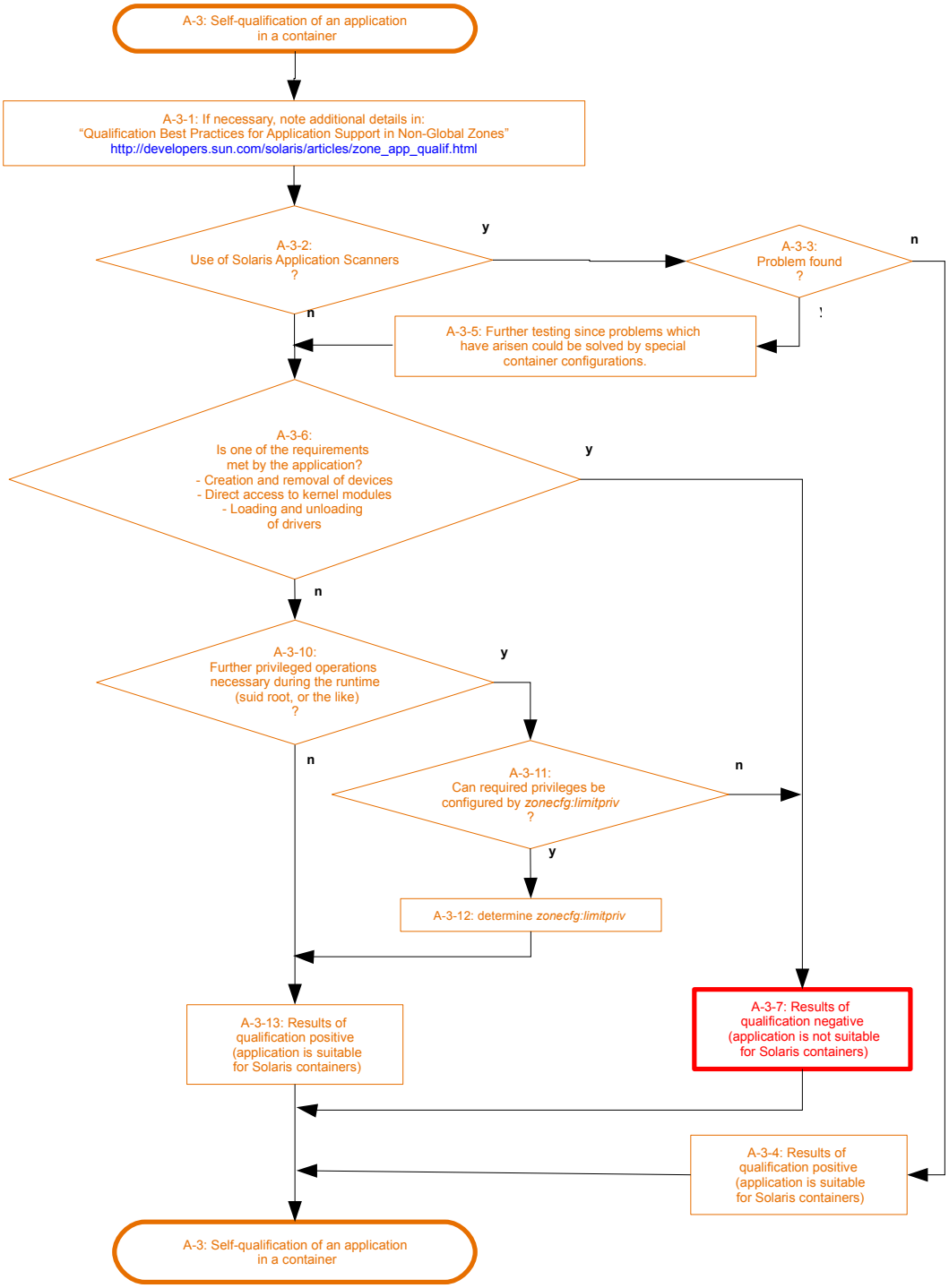


Figure 29: [dd] Self-qualification of an application in a container

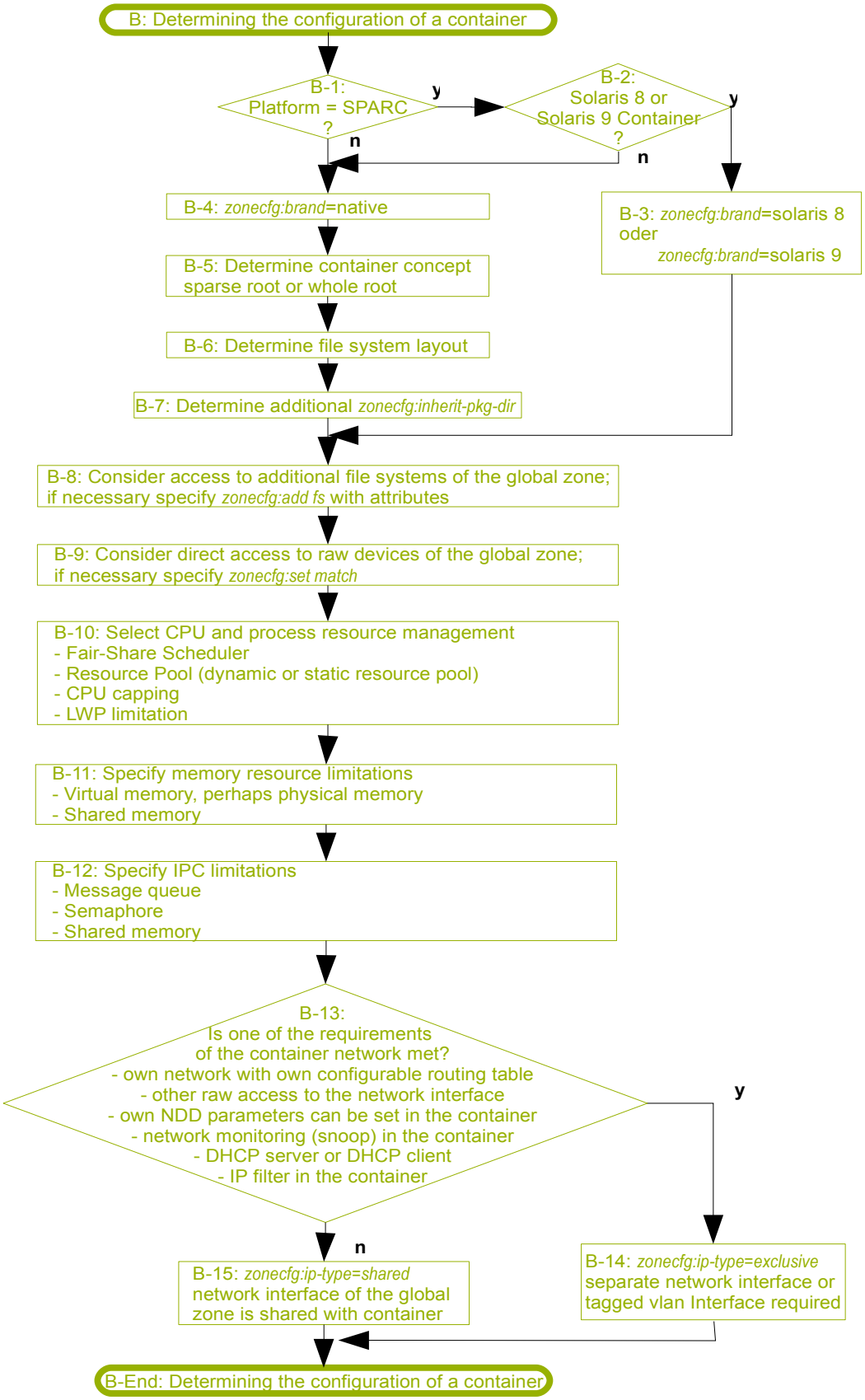


Figure 30: [dd] Determining the configuration of a container

5. Cookbooks

The Cookbooks chapter demonstrates the implementation of conceptional Best Practices with concrete examples.

5.1. Installation and configuration

5.1.1. Configuration files

[dd] **File:** `/etc/zones/index`
Lists all configured zones of a system, incl. zone status and root directory.

```
# DO NOT EDIT: this file is automatically generated by zoneadm(1M)
# and zonecfg(1M).  Any manual changes will be lost.
#
global:installed:/
sparse:installed:/zones/sparse:ae343d81-3439-cd4d-ff52-f110feeff8d2
whole:configured:/zones/whole
```

File: `/etc/zones/<zonename>.xml`
Contains the configuration of a zone <zonename> and can also be used as a template (`zoneadm: create -t <zone>`)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE zone PUBLIC "-//Sun Microsystems Inc//DTD Zones//EN"
"file:///usr/share/lib/xml/dtd/zonecfg.dtd.1">
<!--
    DO NOT EDIT THIS FILE.  Use zonecfg(1M) instead.
-->
<zone name="sparse" zonepath="/zones/sparse" autoboot="false"
pool="sparse">
  <inherited-pkg-dir directory="/lib"/>
  <inherited-pkg-dir directory="/platform"/>
  <inherited-pkg-dir directory="/sbin"/>
  <inherited-pkg-dir directory="/usr"/>
  <inherited-pkg-dir directory="/opt"/>
  <network address="192.168.1.20" physical="bge0"/>
</zone>
```

File: `/etc/zones/SUNWblank.xml`
Determines the standard guideline by which blank zones (command `zoneadm create -b`) are created.

```
<!DOCTYPE zone PUBLIC "-//Sun Microsystems Inc//DTD Zones//EN"
"file:///usr/share/lib/xml/dtd/zonecfg.dtd.1">
<zone name="blank" zonepath="" autoboot="false">
</zone>
```

File: `/etc/zones/SUNWdefault.xml`
Determines the standard guideline by which default zones (sparse root) are created (do not change).

```
<!DOCTYPE zone PUBLIC "-//Sun Microsystems Inc//DTD Zones//EN"
"file:///usr/share/lib/xml/dtd/zonecfg.dtd.1">
<zone name="default" zonepath="" autoboot="false">
  <inherited-pkg-dir directory="/lib"/>
  <inherited-pkg-dir directory="/platform"/>
  <inherited-pkg-dir directory="/sbin"/>
  <inherited-pkg-dir directory="/usr"/>
</zone>
```

File: `/var/sadm/install/gz-only-packages`
Lists all packages that were installed in the global zone with `pkgadd -G` and are not used for the creation of local zones. This file should not be modified manually.

5.1.2. Special commands for zones

[dd/ug] The creation and usage of zones in Solaris 10 is done by the following commands:

Command	Description
zoneadm(1M)	Administration of zones (installation/uninstallation, booting/rebooting/halting, listing)
zonecfg(1M)	Zone configuration
zlogin(1)	Zone login from the global zone -C Connection to the zone console
zonename(1)	Print the name of the current zone

Table 6: [dd/ug] Commands for zones

With the following commands, administrative actions with respect to zones are possible:

Command	Description
dladm(1M)	Administration of data links
ifconfig(1M)	zone <zonename> assigns a virtual IP address to a zone -zone gives a virtual IP address back to the global zone -Z executes the command for all interfaces in the zone
ipcrm(1)	-z <zonename) deletes a message queue, a semaphore set or a shared memory ID of a zone
patchadd(1M)	-G installs a patch in the current zone only (caution!)
pkill(1)	-z <zoneid> applies pkill to processes in zone zoneid
pkgadd(1M)	-G installs a package in the current zone only (caution!)
pooladm(1M)	Activation and deactivation of resource pools
poolbind(1M)	-i zoneid <zonename> binds all processes of the zone <id> or <zonename> to the resource pool
poolcfg(1M)	Configuration of Ressource Management
prctl(1)	-i zoneid <zonename> Displaying and modification of resource controls of zone processes
priocntl(1)	-i zoneid <zoneidlist> Displaying and changing the scheduling parameters in a zone.
rctladm(1M)	Displaying and modification of global settings for resource controls
renice(1)	-i zoneid <zoneidlist> modifies the priority of running processes in <zoneidlist>

Table 7: [dd/ug] Administrative commands for zones

The following commands allow information to be displayed depending on the zones:

Command	***Description***
df(1M)	-Z displays the zone mounts
ifconfig(1M)	zone <zonename> assigns a virtual IP address to a zone -zone gives a virtual IP address back to the global zone -Z executes the command for all interfaces in the zone
ipcs(1)	-z <zonename> displays interprocess communication parameters of a zone
pgrep(1)	-z <zoneid> applies pgrep to processes in zone zoneid
poolstat(1M)	Displays statistics of resource pools
prctl	-i zoneid <zonename> Displays and modifies resource controls of zone processes
priocntl(1)	-i zoneid <zoneidlist> shows the scheduling parameters of the processes of a zone zoneid
prstat	-z <zonename> displays the process statistics of the processes of <zonename> -Z displays information on processes and zones together
ps(1)	-o pid,...,zone zone exists as a format field for configurable outputs -z <zonename> displays only <zonename> processes -Z displays the zone that the process belongs to
rcapstat(1)	-z displays statistics for capped zones
rctladm(1M)	Display and modification of global settings for resource controls

Table 8: Commands used to display zone information

5.1.3. Root disk layout

[dd] The following table gives an example for a root disk layout of a system with a local zone. The assumptions on the size of the file systems represent empirical values and examples and can vary depending on the scope of the software installation and local requirements.

Path	***File system size (example)***	***Comment***
`/`	8 GB	`/opt` for the global zone can be contained in this
`swap`	2 GB	depending on the size of the main memory, requirement and experience The dump device is also configured in the swap device.
`/var`	6 GB	depending on requirement and experience
`/var/crash`	4 GB	can also be located within `/var` if necessary Size depends on main memory size and experience
`ABE (/)`	8 GB	Alternative boot environment (ABE) for `/`
`ABE (/var)`	6 GB	ABE for `/var` `/var/crash` and `swap` can be shared among the two BE
`metadb`	100 MB	SVM meta database if the Solaris Volume Manager is used
`/zones`	2 GB	root file system for all zones, if several zones share a file system
`/zones/zone1`	1 GB	root file system exclusively for zone1
`/zones/root`	1 GB	root file system for several zones if several zones share a file system and if `/var` of the local zone is to be separate. Further partitioning e.g. into /zones/root/zone2, /zones/root/zone3, etc.
`/zones/var`	1 GB	/var file system for several zones if several zones are to share a file system but `/var` is separated and monitored separately. Further partitioning e.g. in `/zones/var/zone2`, `/zones/var/zone3`, etc.
`/app1`	1 GB	contains the application that is to run within the zone `/app1` is e.g. mounted in the zone on `/opt/app1`, is separated from rootfs of the zone and can also be relocated independent of the zone.

Table 9: [dd] Root disk layout

Since Solaris 10 10/08, a root file system and the zone path can also be set up on ZFS. This allows ZFS file systems to be used for the individual file systems, and it is not necessary to provide a large number of slices and volumes. Care must be taken, however, that all file systems present within a zpool can only be moved/migrated/relocated to another system together. Therefore, if the plan is to migrate individual zones to other systems via `zoneadm detach ... attach`, these zones should have their own zpools assigned, or the file systems should be relocated via `zfs send ... receive`.

5.1.4. Configuring a sparse root zone: required Actions

[dd] To change a sparse root zone into a whole root zone it is necessary to re-install the zone after change of the configuration. Therefore, before starting the configuration, the type of zone to be created must be selected wisely. Decision support aids have already been discussed in this document. The following information is required for the configuration:

- Determination of zone name
- Determination of the root directory of the zone
- Assignment of an interface
- Assignment of an IP address
- In this example, `/opt/sfw` is specified as `inherit-pkg-dir` since software from the companion DVD is installed here and can thus be used by all local zones once it is installed in the global zone. This decision must be made before the zone is created and can subsequently only be changed by reinstalling the zone.
- The directory `/opt` is copied and remains writable for the local zone.

Following configuration, the zone must be installed and initialized prior to first use (see 5.1.6 Zone installation).

```
global# zonecfg -z zone1
zone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/zones/zone1
zonecfg:zone1> add inherit-pkg-dir
zonecfg:zone1:inherit-pkg-dir> set dir=/opt/sfw
zonecfg:zone1:inherit-pkg-dir> end
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=bge0
zonecfg:zone1:net> set address=192.168.1.1/24
zonecfg:zone1:net> end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> info
zonename: zone1
zonepath: /zones/zone1
brand: native
autoboot: false
bootargs:
pool:
limitprivs:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
inherit-pkg-dir:
        dir: /opt/sfw
net:
        address: 192.168.1.1/24
        physical: bge0
zonecfg:zone1> exit
global#
```

5.1.5. Configuring a whole root zone: required Actions

[dd] Whole root zones do not contain *inherit-pkg-dir* and are generated with *zonecfg create* from the default file */etc/zone/SUNWdefault.xml*. Subsequently, all *inherit-pkg-dir* are removed with *remove inherit-pkg-dir*. We advise against using *zonecfg create -b* because such a zone is created with the default from */etc/zones/SUNWblank.xml* which does not necessarily need to correspond to a whole root zone. The following information is required to configure a whole root zone:

- Determination of zone name
- Determination of the root directory of the zone
- Assignment of an interface
- Assignment of an IP address

Following configuration, the zone must be installed and initialized prior to first use (see 5.1.6 Zone installation).

```
global# zonecfg -z whole
whole: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:whole> create
zonecfg:whole> set zonepath=/zones/whole
zonecfg:whole> remove inherit-pkg-dir dir=/sbin
zonecfg:whole> remove inherit-pkg-dir dir=/usr
zonecfg:whole> remove inherit-pkg-dir dir=/platform
zonecfg:whole> remove inherit-pkg-dir dir=/lib
zonecfg:whole> add net
zonecfg:whole:net> set physical=bge0
zonecfg:whole:net> set address=192.168.1.1/24
zonecfg:whole:net> end
zonecfg:whole> verify
zonecfg:whole> commit
zonecfg:whole> info
zonename: whole
zonepath: /zones/whole
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
net:
        address: 192.168.1.1/24
        physical: bge0
zonecfg:whole> exit
global#
```

5.1.6. Zone installation

[dd] Before using a zone for the first time it must be installed according to your configuration. The installation time varies depending on whether a sparse-root zone or a whole-root zone is installed. Furthermore, the amount of the software installed in the global zone, the disk technology used (SATA, IDE, SCSI or FC) and whether a write cache is present in the disk subsystem used for the zone root determines the time required to install a zone. This time can vary between 3 and 20 minutes for a sparse-root zone.

```
global# zoneadm -z zone1 install
Preparing to install zone <zone1>.
Creating list of files to copy from the global zone.
Copying <2373> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <985> packages on the zone.
Initialized <985> packages on zone.
Zone <zone1> is initialized.
Installation of these packages generated warnings: <SUNWxwfnt
SUNWxwcft SUNWolrte SUNWpprou SUNWjxmft SUNWi1of SUNWjxcft SUNWxwoft>
The file </zones/zone1/root/var/sadm/system/logs/install_log> contains
a log of the zone installation.
global#
```

After the zone is installed, a `zlogin -C <zone>` must be performed after the first boot-up of the zone in order to carry out the required initializations for the services, the root password, name service, locale, type of terminal at the console, and the time zone. This completes zone installation.

5.1.7. Zone initialization with sysidcfg

[dd] A zone can also be initialized through the file `sysidcfg(4)`. This file can be created separately and contains the appropriate parameters. An OS instance starting up for the first time uses the `sysidcfg` file, when located in `/etc`.

So for zones, it works in the following way:

1. Configure the zone (`zonecfg`)
2. Install the zone (`zoneadm`)
3. Create the `sysidcfg` file and copy it into the `/etc` of the zone (this is located in the global zone under `<zonepath>/root/etc`)
4. Boot the zone

The following example shows one possible sysidcfg(4) file.

```
global # cat /zones/zone1/root/etc/sysidcfg
root_password=xyz12321zyx
name_service=none
system_locale=C
terminal=vt100
timezone=CET
timeserver=localhost
network_interface=PRIMARY {hostname=zone1}
security_policy=limited
nfs4_domain=dynamic
```

5.1.8. Uninstalling a zone

[dd] Installed zones are uninstalled by `zoneadm -z <zone> uninstall`. In doing so, the following actions are carried out:

- Deletion of data in the `zonepath` subdirectory
- Conversion of zone status in `/etc/zones/index` into `configured`

5.1.9. Configuration and installation of a Linux branded zone with CentOS

[dd] Linux branded zones can only be set up on x86/x64 systems. The Linux distribution is not a Solaris component and must be provided separately (here, as ISO images).

```
global# zonecfg -z centos
centos: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:centos> create -t SUNWlx
zonecfg:centos> set zonepath=/zones/centos
zonecfg:centos> add net
zonecfg:centos:net> set physical=bge0
zonecfg:centos:net> set address=192.168.1.100
zonecfg:centos:net> end
zonecfg:centos> add attr
zonecfg:centos:attr> set name=audio
zonecfg:centos:attr> set type=boolean
zonecfg:centos:attr> set value=true
zonecfg:centos:attr> end
zonecfg:centos> verify
zonecfg:centos> commit
zonecfg:centos> exit
global# zoneadm -z centos install -v -d /dist/brandz/images
A ZFS file system has been created for this zone.
Verbose output mode enabled.
Installing zone "centos" at root "/zones/centos"
  Attempting ISO-based install from directory:
    "/dist/brandz/images"
Checking possible ISO
  "/dist/brandz/images/CentOS-3.8-i386-bin1of3.iso"...
  added as lofi device "/dev/lofi/1"
Attempting mount of device "/dev/lofi/1"
  on directory "/tmp/lxiso"... succeeded.
Attempting to read "/tmp/lxiso/.discinfo"...
Unmounting device "/dev/lofi/1"... succeeded.
...
...

Completing installation; this may take a few minutes.
Setting up the initial lx brand environment.
System configuration modifications complete.
Installation of CentOS 3.8 to zone
  'centos' completed Mon Aug  6 16:42:55 CEST 2007.

Installation of zone 'centos' completed successfully.

Details saved to log file:
    "/zones/centos/root/var/log/centos.install.1594.log"
global#
```

5.1.10. Configuration and installation of a Solaris 8/Solaris 9 container

[ug] Solaris 8 containers and Solaris 9 containers can be created using 4 simple steps.

1. Planning how the data areas and network interfaces of the sources under Solaris 8 (or Solaris 9) can be represented in the Solaris 10 system.
2. Then the Solaris 8 system is archived. The P2V tool supplied can be used for this, but also any other archiving tool such as tar or cpio.
3. Next, the zone is configured as a Solaris 8 container (or a Solaris 9 container).
4. The archived system is installed with `zoneadm install`.
5. Data directories are transported to the new storage architecture.
6. The system is ready for testing.

5.1.11. Optional settings

[dd] Once the zone is configured, changes can be made to the configuration. Some examples are listed in the following sections based on the sparse-root zone zone1. Modifications to the zone configuration are activated by restarting the zone.

5.1.11.1. Starting zones automatically

[dd] Zones can be configured such that they are started up immediately after booting the system. In the original configuration, zones must be started manually by the administrator.

```
global# zonecfg -z zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> commit
zonecfg:zone1> exit
global#
```

5.1.11.2. Changing the set of privileges of a zone

[dd] The set of privileges of a local zone can be extended by certain privileges. In this example, the privileges `dtrace_proc` and `dtrace_user` are additionally required within a zone to apply DTrace to local processes within the own zone (see *4.5.6 DTrace of processes within a zone*).

```
global# ppriv -l zone | wc -l
global# 68

zone1# ppriv -l zone | wc -l
zone1# 34

global# zonecfg -z zone1
zonecfg:zone1> set limitpriv=default,dtrace_proc,dtrace_user
zonecfg:zone1> commit
zonecfg:zone1> exit

zone1# ppriv -l zone | wc -l
zone1# 36
```

5.1.12. Storage within a zone

[dd] Storage can be used in different ways in local zones. The type of use can be classified in the following ways:

- Device or NFS
  - Use thedevice in the local zone directly
- Mount
  - The global zone provides a file system per lofs to the local zone
  - The global zone mounts a file system when the local zone is booted
  - The zone mounts a file system from a device
- Access
  - read/write
  - read only
  - Share writable file system with several zones

5.1.12.1. Using a device in a local zone

[dd] If direct access to a device is required by the local zone, the device must be made available by the corresponding zone configuration. Once the zone has been restarted, the device entry is available under /dev, as seen from the local zone.

```
global# zonecfg -z zone1
zonecfg:zone1> add device
zonecfg:zone1:device> set match=/dev/rdsk/c1d0s0
zonecfg:zone1:device> end
zonecfg:zone1> info device
device:
        match: /dev/rdsk/c1d0s0
zonecfg:zone1> commit
zonecfg:zone1> exit
```

5.1.12.2. The global zone supplies a file system per lofs to the local zone

[dd] `/export/globalhome` of the global zone is to be mounted as writable in the local zone as `/export/home`. Since `/export/globalhome` is already available in the global zone, the file system can be added as a loopback filesystem. If the loopback filesystem is mounted as writable, both the global zone as well as zone1 can write on the file system.

```
global# zonecfg -z zone1
zonecfg:zone1> add fs
zonecfg:zone1:fs> set special=/export/globalhome
zonecfg:zone1:fs> set dir=/export/home
zonecfg:zone1:fs> set type=lofs
zonecfg:zone1:fs> add options rw
zonecfg:zone1:fs> info
fs:
        dir: /export/home
        special: /export/globalhome
        raw not specified
        type: lofs
        options: [rw]
zonecfg:zone1:fs> end
zonecfg:zone1>
```

5.1.12.3. The global zone mounts a file system when the local zone is booted

[dd] File systems can be provided to a local zone by the global zone not only as loopback filesystems. The following example shows how to mount a file system as an UFS directly when the zone is booted. The file system is mounted by the global zone in `/zones/<zonepath>/root`.

Please note: According to **RFE 6495558**, for filesystems that have been mounted in this manner by `zoneadm -z zone1 boot`, no repair of corrupt file systems will be performed. This can therefore prevent zones from starting.

```
zonecfg:zone1> add fs
zonecfg:zone1:fs> set dir=/mnt
zonecfg:zone1:fs> set special=/dev/dsk/c1d0s0
zonecfg:zone1:fs> set raw=/dev/rdsk/c1d0s0
zonecfg:zone1:fs> set type=ufs
zonecfg:zone1:fs> add options rw
zonecfg:zone1:fs> info
fs:
        dir: /mnt
        special: /dev/dsk/c1d0s0
        raw: /dev/rdsk/c1d0s0
        type: ufs
        options: [rw]
zonecfg:zone1:fs> end

global# zoneadm -z zone1 reboot
global# df -kZ /dev/dsk/c1d0s0
Filesystem       kbytes    used   avail  capacity  Mounted on
/dev/dsk/c1d0s0  7064379 3794979 3198757 55%     /zones/zone1/root/mnt
```

5.1.12.4. The local zone mounts a UFS file system from a device

[dd] The local zone itself can also mount filesystems. To do so, the corresponding block and raw device must be assigned to the zone. Once the zone has been restarted, the devices are available in the zone.

```
zonecfg:zone1> add device
zonecfg:zone1:device> set match=/dev/dsk/c1d0s0
zonecfg:zone1:device> end
zonecfg:zone1> add device
zonecfg:zone1:device> set match=/dev/rdsk/c1d0s0
zonecfg:zone1:device> end
zonecfg:zone1> info device
device:
        match: /dev/dsk/c1d0s0
device:
        match: /dev/rdsk/c1d0s0
zonecfg:zone1> commit
zonecfg:zone1> exit
```

After installing and booting of the zone the mount can be done inside of `zone1`:

```
zone1# ls /dev/dsk
c1d0s0
zone1# mount /dev/dsk/c1d0s0 /mnt
zone1# df -k /dev/dsk/c1d0s0
Filesystem         kbytes    used   avail capacity  Mounted on
/dev/dsk/c1d0s0 7064379 3794979 3198757   55%    /mnt
global# df -kZ /zones/zone1/root/dev/dsk/c1d0s0
Filesystem      kbytes    used   avail capacity  Mounted on
/zones/zone1/root/dev/dsk/c1d0s0
                7064379 3794979 3198757   55%    /zones/zone1/root/mnt
```

5.1.12.5. User level NFS server in a local zone

[ug] The native NFS in the Solaris kernel can currently not be used as a server within a local zone.

Instead, the unfs3d – an OpenSource NFS server – can be used (see the unfs3 project at SourceForge) which runs completely in userland and is therefore not subject to this limitation. To do so, the RPC configuration must be changed and the unfs3d called up instead of the Solaris NFS daemon.

5.1.12.6. Using a DVD drive in the local zone

[dd] If access to a DVD drive is required in a local zone, the DVD drive for the zone must be assigned to the local zone in the zone configuration. In addition, it must be prevented that the volume daemon of the global zone automatically mounts the DVD once it is placed into the drive. To do so, the smf-service `svc:/system/filesystem/volfs` in the global zone must be turned off. This service is not available to local zones. Therefore, DVDs in local zones must be mounted by command line. Once the zone has been restarted, the device is available to the zone. This is verifiable by the entry in `/dev/dsk`. It is furthermore visible from the global zone that a device entry now exists in `/zones/zone1/dev/dsk`.

```
zonecfg:zone1> add device
zonecfg:zone1:device> set match=/dev/dsk/c2t0d0s2
zonecfg:zone1:device> end
zonecfg:zone1> commit
zonecfg:zone1> exit

global# svcadm disable volfs

. . . . reboot der Zone . . . .

zone1# ls /dev/dsk
c2t0d0s2
zone1# mount -F hsfs /dev/dsk/c2t0d0s2 /mnt
zone1# df -k /dev/dsk/c2t0d0s2
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c2t0d0s2    2643440 2643440       0   100%    /mnt

global# df -kZ /zones/zone1/root/dev/dsk/c2t0d0s2
/Filesystem       kbytes  used   avail capacity  Mounted on
/zones/zone1/root/dev/dsk/c2t0d0s2
                  2643440 2643440    0   100%    /zones/zone1/root/mnt
```

Alternatively, the DVD can be mounted as a loopback file system in one or several zones. In this case, it must be mounted by the global zone and cannot be ejected until all zone loopback mounts are ended.

5.1.12.7. Dynamic configuration of devices

[ug] Devices can be assigned to a zone statically (permanently) by zonecfg. To activate them, however, it is necessary to reboot the zone:

```
global# zonecfg -z test
zonecfg:test> add device
zonecfg:test:device> set match=/dev/dsk/c1t1d0s1
zonecfg:test:device> end
zonecfg:test> verify
zonecfg:test> commit
zonecfg:test> exit
```

For dynamic configuration, the device's major and minor number must be determined. This information can be obtained with the `ls` command in the global zone. The option `-L` is important here because it tracks symbolic links up until the actual entry in /devices:

```
global# cd /dev
global# ls -lL /dev/rdsk/c1t1d0s1
crw-r-----   1 root     sys      118,   1 Dec  5 19:33 rdsk/c1t1d0s1
global#
```

With this information, a corresponding entry can then be made in the /dev directory of the local zone using the command *`mknod`*. The device type (here: *`c`* = character device) and the major and the minor device number must be specified (here: 118 and 1 from the ls -lL output). In a next step, the appropriate access privileges should be set with *`chmod`*.

```
global# cd ...zonepath.../dev
global# mknod rdsk/c1t1d0s1 c 118 1

global# ls -lL rdsk/c1t1d0s1
crw-r--r--   1 root     root     118,   1 Jan 29 16:06

global# chmod o-r rdsk/c1t1d0s1
global# ls -lL rdsk/c1t1d0s1
crw-r-----   1 root     root     118,   1 Jan 29 16:06 rdsk/c1t1d0s1
```

Now the device is immediately visible and usable within the zone.

```
test# ls -l /dev/rdsk/c*
crw-r--r--   1 root root 118, 1 Jan 29 16:06/dev/rdsk/c1t1d0s1
bash-3.00 ls -l /dev/rdsk/c*
total 0
crw-r-----   1 root     root     118,   1 Jan 29 16:06 c1t1d0s1

test# reboot
```

For permanent use and for the purpose of documentation, the device should also be entered in the zone configuration.

```
global# zonecfg -z test
zonecfg:test> add device
zonecfg:test:device> set match=/dev/rdsk/c1t1d0s1
zonecfg:test:device> end
zonecfg:test> verify
zonecfg:test> commit
zonecfg:test> exit
global#
```

If the device is then deleted from the zone configuration, it will also be removed from the /dev directory of the local zone after the next reboot. This procedure is preferable to manual removal from the /dev directory.

```
global# zonecfg -z test
zonecfg:test> remove device match=/dev/rdsk/c1t1d0s1
zonecfg:test> verify
zonecfg:test> commit
zonecfg:test> exit
global#
```

5.1.12.8. Several zones share a file system

[dd] The zone model makes it very easy for several zones to share a writable file system. This becomes possible if the global zone mounts a file system and makes this same file system available to several zones as a read/write loopback file system (Cookbook *5.1.12.2 The global zone supplies a file system per lofs to the local zone*). From the point of view of Solaris, the operations on the filesystem are like are equal writing operations in a local file system. The applications must implement file-locking as usual, it works like with a shared filesystem.

This corresponds to an NFS file system being accessed by several clients. The global zone corresponds to the NFS server and the local zones correspond to the NFS clients.

5.1.12.9. ZFS in a zone

[ug] A ZFS file system can be assigned to a zone such that the zone administrator can use it as the base for further configuration and administration.

- The ZFS can be assigned to the zone through `add dataset` of the `zonecfg` command.
- Within the zone, the administrator can derive additional ZFS file systems from the file system by `zfs create`.
- The attribute quota specified in the global zone cannot be changed or exceeded in the zone.
- However, the zone administrator can specify the mount point within the zone.

```
# zpool create -m none tank
# zfs create tank/zone1
# zfs set quota=1g tank/zone1
# zfs set mountpoint=/mnt tank/zone1
#
# zonecfg -z zone1
zonecfg:zone1> add dataset
zonecfg:zone1:device> set name=tank/zone1
zonecfg:zone1:device> end
zonecfg:zone1> commit
zonecfg:zone1> exit

. . . reboot der Zone zone1 . . .

. . . login in die Zone zone1 . . .

zone1# zfs list
NAME              USED  AVAIL  REFER  MOUNTPOINT
tank             30.2M   10G    18K   none
tank/zone1       16.3M  983M  16.3M  /mnt
```

5.1.12.10. User attributes for ZFS within a zone

[ug] ZFS allows additional attributes to be managed for each file system that are stored together with the file system (since Solaris 10 5/08). It is necessary, that the name of these additional attributes contains a ":". These so-called user attributes can both be viewed manually as well as be used in scripts. The advantage is that they can be stored within the file system, allowing the documentation to remain with the file system even when used by another computer.

```
# zfs set admin:application="DB Webportal"          tank/db234_01
# zfs set admin:importance="critical"               tank/db234_01
# zfs set admin:costcenter="47110815"               tank/db234_01
# zfs set admin:application="App-Server Webportal"  tank/web234_01
# zfs set admin:costcenter="47110815"               tank/web234_01
# zfs get all  tank/web234_01 | fgrep admin:
tank/web234_01  admin:application  App-Server Webportal   local
tank/web234_01  admin:costcenter   47110815               local
#
# zfs get -H -o value admin:costcenter tank/db234_01
47110815
```

5.1.13. Configuring a zone by command file or template

[dd] Zones can be configured by using command files for *zonecfg* or by the use of templates. This allows quick and automatic configuration of many zones avoiding errors.

- Using the *zonecfg* command file
  A command file is set up for automatic configuration of the zone.
  1. Create zonecfg command file
     - with *zonecfg -z <zone> export -f <file>* of an existing zone
     - alternatively: create the zone config file with a text editor, one command per line
  2. Use the command file to configure a new zone
     – *zonecfg -z <zone> -f <file>*
- Using templates
  A new zone is configured by using an already configured zone. Any necessary configuration changes are done after that
  – *zonecfg -z <zone> create -t <template-zone>*

5.1.14. Automatic quick installation of zones

[ug] The installation of zones takes just a few to several minutes, depending on the speed of the file system (disks) and the zone software to be copied. This can be reduced considerably if many zones are to be created that are identical with respect to their configuration (essential criteria for this are the *inherit-pkg-dir* settings).
Preparation:

1. *zonecfg* of a template zone with the desired configuration,
   which must not be used later on (for example with the name *template1*).
   In so doing, *zonepath* must be set, the configuration of the network address should be absent because the zone merely serves as a template after all and is not to become operational.
2. Install the zone *template1* with *zoneadm*.
3. Save the contents of the zone *template1* in a tar file *template1.tar* .
   (Note: `cp -r` is not suitable for copying zones!)

Creation of duplicated zones:

1. Configure with *zonecfg* using the template function
   (*create -t template1*).
   (The *inherit-pkg-dir* settings must not be changed!)
2. Change *zonepath* with *zonecfg* (can be done by script)
3. Enter the network address(es) with *zonecfg*.
4. Unpack *template1.tar* below *zonepath* of the new zone.
5. <zone> attach
6. Boot the new zone for operation.

Since the template zone cannot be started up directly, the following procedure is required to install patches:

1. Configure the zone *template1* (manually or with *sysidcfg* file)
2. Install the patches (it is best to have all configured zones booted up)
3. De-configure the zone *template1* with the command
   zlogin template1 sys-unconfig
4. Re-create the zone content file *template1.tar* .

Alternatively, the command *zoneadm -z template1 clone* can be used instead of a manual copy.

5.1.15. Accelerated automatic creation of zones on a ZFS file system

[bf/ug] If a zone is configured on a ZFS file system, it can be duplicated very quickly by using ZFS snapshots. This procedure is described below by means of an example script. The script is available for download at http://blogs.sun.com/blogfinger/entry/how_to_create_a_lot.

In the first part of the script, the most important parameters for the zones are to be defined. These include for example:

- Number of zones to be created
- Network address range
- Name of network interface
- Net mask
- Gateway
- Base zone name (supplemented with number for the zone name)
- Zone directory (supplemented with zone name)
- Name of the zone that is used as the basis for cloning
- Information for the `sysidcfg` file
- Start status for the zone after installation

Once these settings have been made, the script can create the zones automatically and start in the configured state. More details on the script are available in the blog entry.

5.1.16. Zones hardening

[dd] To harden Solaris, the Solaris Security Toolkit is recommended as a general rule. Complete procedures and mechanisms can be found here:
http://www.sun.com/products-n-solutions/hardware/docs/Software/enterprise_computing/systems_management/sst/index.html

Within the toolkit, the features that are required to harden sparse-root or whole-root zones are described. Details on this can be found here:
http://www.sun.com/products-n-solutions/hardware/docs/html/819-1503-10/introduction.html#pgfId-1001177

With Solaris 10 11/06, the feature "Secure by default" was introduced for network services which allows all network services except for `sshd` to be turned off or reconfigured by calling up `netservices limited` such that they will only react to requests by `localhost`. As a result, considerable safeguarding of zones in networks is possible using simple means.

5.2. Network

5.2.1. Change network configuration for shared IP instances

[dd] For an already configured zone with a shared IP instance, it may be necessary to change the physical interface or the network address.

```
global# zonecfg -z zone1
zonecfg:zone1> info net
net:
        address: 192.168.1.1/24
        physical: bge1
zonecfg:zone1> select net physical=bge1
zonecfg:zone1:net> set physical=bge0
zonecfg:zone1:net> set address=192.168.2.1/24
zonecfg:zone1:net> info
net:
        address: 192.168.2.1/24
        physical: bge0
zonecfg:zone1:net> end
zonecfg:zone1> commit
zonecfg:zone1> exit
global#
```

5.2.2. Set default router for shared IP instance

[dd] In the zone configuration, the default router can be specified for a zone with a shared IP instance. Using this configuration option ensures that a certain network route is available in the global zone when a zone is booted (since Solaris 10 11/08)

```
global# zonecfg -z keetonga
zonecfg:keetonga> select net physical=e1000g0
zonecfg:keetonga:net> set defrouter=1.2.3.1
zonecfg:keetonga:net> info
net:
     address: 1.2.3.4/24
     physical: e1000g0
     defrouter: 1.2.3.1
zonecfg:keetonga:net> end
```

5.2.3. Network interfaces for exclusive IP instances

[dd] In order to use exclusive IP instances, GLDv3-driver enabled network interfaces or tagged VLAN interfaces are required and are collectively termed GLDv3-interfaces below. These drivers include: bge, ce, e1000g, eri, hme, ixgb, nge, nxge, rge, xge. Further details on supported adapters can be found at: http://opensolaris.org/os/project/crossbow/faq/#ip_instances. The adapters in the system can be displayed with `dladm show-link`.

Each exclusive IP instance must have a physical interface or a tagged VLAN interface assigned. Therefore, the required number of interfaces results from the number of zones with an exclusive IP instance required. If tagged VLAN interfaces are used, the maximum number of zones within a VLAN results in the number of physical interfaces required. The reason for this lies in the creation rule for tagged VLAN interfaces:

`Interface = Interfacename<instance + (VLAN-ID * 1000)>`

e.g. bge1000 uses the VLAN with ID 1 on bge0

or

bge4003 uses the VLAN with ID 4 on bge3

5.2.4. Change network configuration from shared IP instance to exclusive IP instance

[dd] Zones that are already configured are run with shared IP instances up to Solaris 10 11/06. With the introduction of Solaris 10 8/07, it is possible to run zones with an own IP stack. Such a zone needs a different configuration, where ip-type is set to exclusive and the zone needs a physical interface or a tagged VLAN interface assigned. In this example, a VLAN with VLAN-ID 1 on interface bge0 is assigned to the zone. The device entry is created automatically by the global zone when the zone is started up. The IP address is assigned to the interface by the zone itself.

```
global# zonecfg -z zone1
zonecfg:zone1> info net
net:
        address: 192.168.2.1/24
        physical: bge0
zonecfg:zone1> info ip-type
ip-type: shared
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> verify
net: address cannot be specified for an exclusive IP type
zone1: Invalid argument
zonecfg:zone1> remove net physical=bge0
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=bge1000
zonecfg:zone1:net> end
zonecfg:zone1> info net
net:
        address not specified
        physical: bge1000
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> exit
```

5.2.5. IP filter between shared IP zones on a system

[dd] IP filters can be used to filter network packages between shared IP zones. To do so, IP filter is configured and started in the global zone but filters the data traffic between zones according to the rules.

It should be noted that data traffic between shared IP zones does not leave the TCP/IP-stack of the system. For this data traffic also to be considered by the IP filter, the line `set intercept_loopback true;` must be set in the IP filter configuration.

The following example filters the entire data traffic between two zones (zone keetonga: 192.168.1.210; zone haitoda: 192.168.1.200).

```
global# cd /etc/ipf
global# more ipf.conf
     set intercept_loopback true;
     block in from 192.168.1.210/32 to 192.168.1.200/32
     block out from 192.168.1.210/32 to 192.168.1.200/32
     block in from 192.168.1.200/32 to 192.168.1.210/32
     block out from 192.168.1.200/32 to 192.168.1.210/32
global# svcadm enable ipfilter
```

The following example shows how ssh connections from zone keetonga to zone haitoda are filtered after a present IP filter configuration has been changed and reloaded.

```
global # more ipf.conf
set intercept_loopback true;
block in proto tcp from 192.168.1.210/32 to 192.168.1.200/32 port =
ssh

global # ipf -F a -f /etc/ipf/ipf.conf
```

5.2.6. IP filter between exclusive IP zones on a system

[dd] The usual configuration rules for IP filters must be followed for the use of IP filters in exclusive IP zones. This is possible since, for exclusive IP instances, the physical network port was assigned to the zone.

After configuring the IP filter per zone, IP filter is activated in each zone to work independently in each IP instance. The corresponding command is: `svcadm enable ipfilter`

5.2.7. Zones, networks and routing

[dd/ug] The following sections describe scenarios in zones, networks and routing settings. The following restrictions exist:

- In the directly connected networks, the same IP address must not be assigned twice. If this is unavoidable due to organizational circumstances, NAT routers (scenario 3) must be used for partitioning.
- Routing between the addresses of zones with shared IP occurs in the system. External routing can only be forced by means of a NAT router or by inhibiting routing between zones with ndd: `ndd -set /dev/ip ip_restrict_interzone_loopback 1`
- The network separation s implemented in Solaris at the logical TCP/IP level. This is sufficient for many cases of application.
- If separation is required at the physical network level, it can be implemented by separate systems, Solaris domains or – since Solaris 10 8/07 – by exclusive IP instances.

5.2.7.1. Global and local zone with shared network

[dd/ug] Two local zones, zone1 and zone2, are located in the same network segment as the global zone.

- Each local zone can use the same network interface as the global zone.
- Routing set up for the global zone also applies to the local zones. All zones (global and local) can communicate with each other.

Implementation:

- Zones are set up with the network interface of the global zone; if this is `bge0`, the setup `set physical=bge0` is done with `zonecfg: add net`.
- Each local zone must receive an address from the network of the global zone.

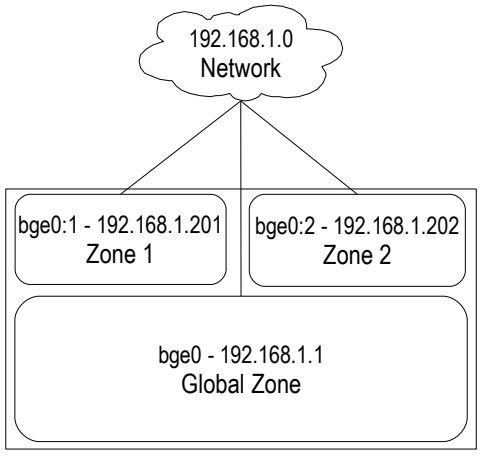


Figure 31: [dd] Global and local zone with shared network

5.2.7.2. Zones in separate network segments using the shared IP instance

[dd/ug] Two local zones, zone1 and zone2, are located in separated network segments and provide services for these network segments.

- Each local zone should have its own physical interface in the network segment.
- No other network is connected to the network segment.
- Routing is not used.
- There should be no communication between local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- The network interface provided for the local zone (e.g. `bge1`) must not be used elsewhere in the global zone.
- To activate the interface for the local zones, the interface must be plumbed (but not enabled):
  `ifconfig bge1 plumb down`
  The interface has the address 0.0.0.0 but is not active.
- No routing entries are made.
- Option: To enable communication between the global and a local zone, the corresponding interface with an address that is located in the network of the local zone must be configured in the global zone.

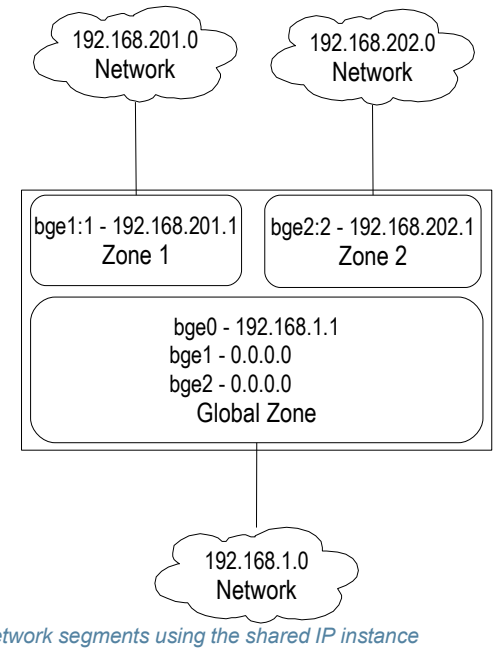


Figure 32: [dd] Zones in separate network segments using the shared IP instance

5.2.7.3. Zones in separate network segments using exclusive IP instances

[dd/ug] Two local zones, zone1 and zone2, are located in separated network segments and provide services for these network segments.

- Each local zone should have its own physical interface.
- No additional network is connected to the network segment.
- Routing is not used.
- There should be no communication between the local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- A separate GLDV3 interface (e.g. `bge1` and `bge2`) is provided for each zone. These interfaces must not be used elsewhere in the global zone.
  ```
  zone1-zonecfg: add net physical=bge1
  zone2-zonecfg: add net physical=bge2
  ```
- The zone configuration for zone1 and zone2 is indicates the use of exclusive IP instances.
  ```
  zonecfg: set ip-type=exclusive
  ```
- The IP addresses are defined inside of the zones.
  ```
  Zone 1: /etc/hostname.bge1
  Zone 2: /etc/hostname.bge2
  ```
- No routing entries in the zones.
- Option: To enable communication between the global and the local zone, an interface that is located in the network of the local zone must be configured in the global zone.
- By the use of exclusive IP instances, communication between the zones or between the zones and the global zone takes place only if corresponding routing entries exist in the zones and if a physical network connection exists between the zone interfaces.

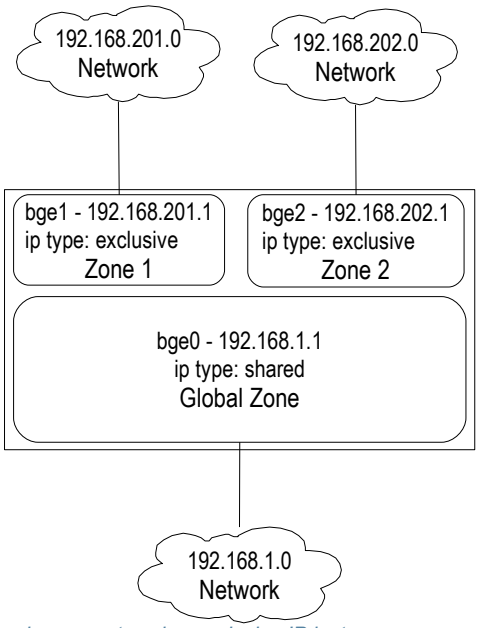


Figure 33: [dd] Zones in separate network segments using exclusive IP instances

5.2.7.4. Zones in separate networks using the shared IP instance

[dd/ug] Two local zones, zone1 and zone2, are located in separated networks and provide services for other networks.

- Each local zone should have its own physical interface in the network.
- Additional networks are connected to the network segment.
- Routing is used.
- There should be no communication between the local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- The network interface intended for the local zone (e.g. `bge1`) must not be used elsewhere in the global zone.
- To prepare for local zones, the interface for a local zone must be plumbed (but not enabled):
  `ifconfig bge1 plumb down`
  Thereby, the interface gets the address 0.0.0.0 but is not active.
- The network configuration of the zones is established by setting the zones to the `ready` status.
  `zoneadm -z zone1 ready`
  `zoneadm -z zone2 ready`
  The addresses listed in the configuration (`zone1: 192.168.201.1` and `zone2: 192.168.202.1`) are now active.
- The routes of the local zones are specified with `zonecfg:set defrouter`.
  `set defrouter=192.168.201.2`
  `set defrouter=192.168.202.2`
- In order to avoid communication between the local zones through the shared TCP/IP stack, reject routes must be set in the global zone that prevent communication between two IP addresses (or the use of ipfilter).
  `route add 192.168.201.1 192.168.202.1 -interface -reject`
  `route add 192.168.202.1 192.168.201.1 -interface -reject`
  Alternatively the interzone loopback can be restricted:
  `ndd -set /dev/ip ip_restrict_interzone_loopback 1`
- The zones can now be booted for operation:
  `zoneadm -z zone1 boot`
  `zoneadm -z zone2 boot`
- Option: To allow communication between the global and the local zone, an interface which is located in the logical network of the local zone must be configured in the global zone.

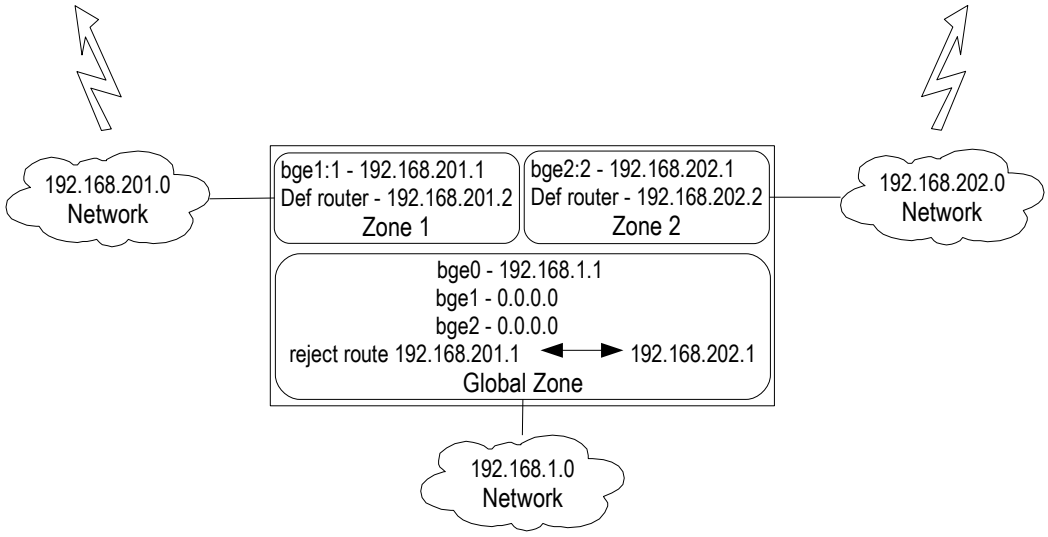


Figure 34: [dd] Zones in separate networks using the shared IP instance

5.2.7.5. Zones in separate networks using exclusive IP instances

[dd] Two local zones, zone1 and zone2, are located in separated networks and provide services for other networks.

- Each local zone should have its own physical interface in the network.
- Additional networks are connected to the network segment.
- Routing is used.
- There should be no communication between local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- A separate GLDV3 interface (e.g. *bge1* and *bge2*) is provided for each zone. These interfaces must not be used elsewhere in the global zone.
  ```
  zone1-zonecfg: add net physical=bge1
  zone2-zonecfg: add net physical=bge2
  ```
- The zone configuration for zone1 and zone2 is converted to the use of exclusive IP instances.
  ```
  zonecfg: set ip-type=exclusive
  ```
- In the zones, the IP addresses and the default router are specified in the usual way.
  ```
  Zone 1: /etc/hostname.bge1
  Zone 2: /etc/hostname.bge2
  /etc/defaultrouter
  ```
- Through the exclusive IP instances, communication between the zones or between the zones and the global zone takes place only if corresponding routing entries exist in the zones and if a physical network connection exists between the zone interfaces.

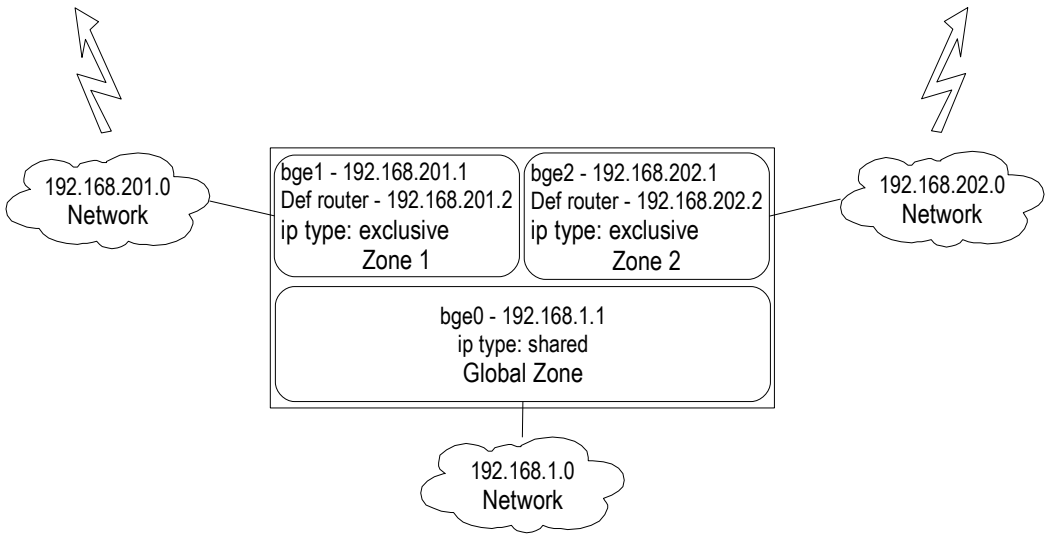


Figure 35: [dd] Zones in separate networks using exclusive IP instances

5.2.7.6. Zones connected to independent customer networks using the shared IP instance

[dd/ug] Two local zones, zone1 and zone2, are located in separated networks and provide services for a variety of customers in their own networks.

- Each local zone should have its own physical interface in the network.
- Additional customer networks are connected to the network segment.
- Allocation of addresses in the networks is not coordinated; one address can be allocated multiple times (once per customer network). Usually companies use private IP networks (10.x.y.z, 192.168.x.y) internally, therefore the allocation of the same IP address at different customers is highly probable.
- It should be possible to reach zones zone1 and zone2 from other networks.
- Zones zone1 and zone2 should not be able to initiate connections to other networks.
- There should be no communication between the local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- The network interface provided for the local zone (e.g. `bge1`) must not be used elsewhere in the global zone.
- To prepare for local zones, the interface must be plumbed (but not enabled):
  ```
  ifconfig bge1 plumb down
  ```
  Thereby, the interface gets the address 0.0.0.0 but is not active.
- The zones' network configuration is established by setting the zones to the `ready` state.
  ```
  zoneadm -z zone1 ready
  zoneadm -z zone2 ready
  ```
  The addresses listed in the configuration of the zones (`zone1: 192.168.201.1` and `zone2: 192.168.202.1`) are now active.
- The routes of the local zones are specified with `zonecfg:set defrouter`.
  ```
  set defrouter=192.168.201.2
  set defrouter=192.168.202.2
  ```
- So that no communication takes place between the local zones through the shared TCP/IP stack, reject routes must be set in the global zone that prevent communication between two IP addresses.
  ```
  route add 192.168.201.1 192.168.202.1 -interface -reject
  route add 192.168.202.1 192.168.201.1 -interface -reject
  Alternatively the interzone loopback can be restricted:
  ndd -set /dev/ip ip_restrict_interzone_loopback 1
  ```
- The zones can now be booted for operation:
  ```
  zoneadm -z zone1 boot
  zoneadm -z zone2 boot
  ```
- The default router is a NAT router that hides the IP address of the local zone from the customer. On the customer's side, it is configured with an IP address from the customer's network, thus, address conflicts can not occur.
- Option: To enable communication between the global and the local zone, an interface that is located in the logical network of the local zone must be configured in the global zone.

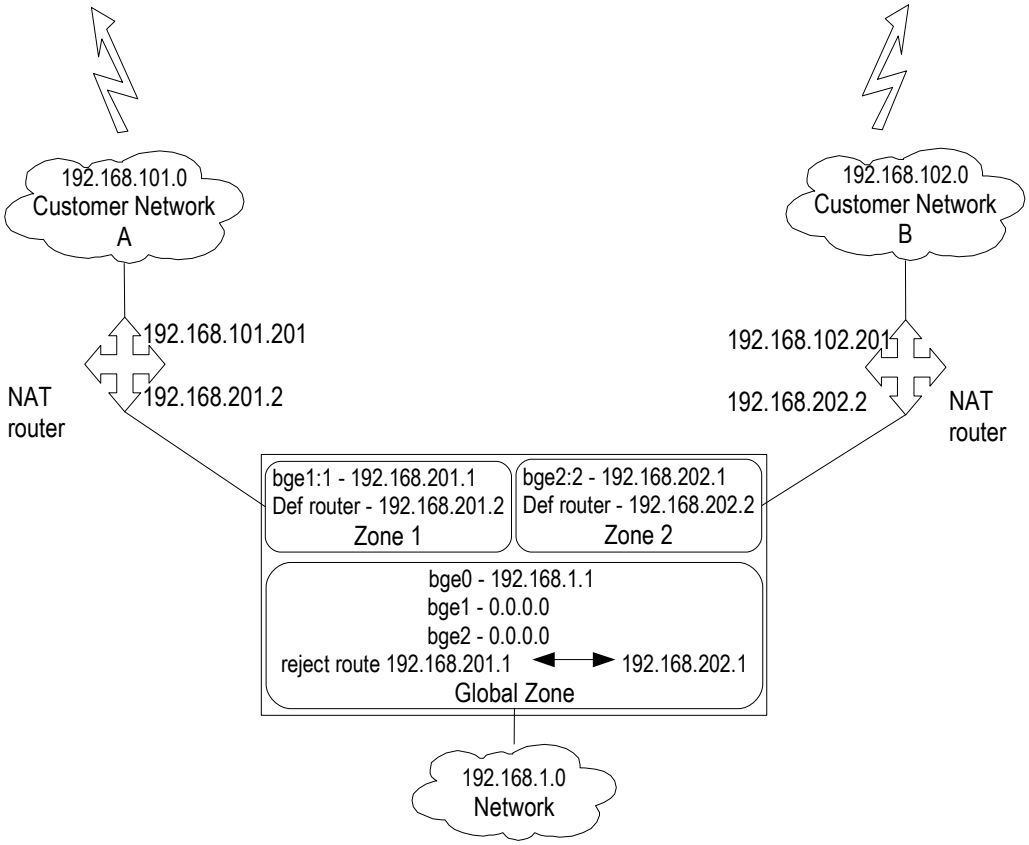


Figure 36: [dd] Zones connected to independent customer networks using the shared IP instance

5.2.7.7. Zones connected to independent customer networks using exclusive IP instances

[dd/ug] Two local zones, zone1 and zone2, are located in separated networks and provide services for a variety of customers in their own networks.

- Each local zone should have its own physical interface .
- Additional customer networks are connected to the network segment.
- Allocation of addresses in the networks is not coordinated; an address can be allocated multiple times (once per customer network). Considering today's customary use of private IP addresses, this is somewhat probable.
- It should be possible to reach the zones zone1 and zone2 from other networks.
- Zones zone1 and zone2 cannot initiate any connections to other networks.
- There should be no communication between local zones.
- Communication between the global zone and the local zones is not intended.

Implementation:

- A separate GLDV3 interface (e.g. *bge1* and *bge2*) is provided for each zone. These interfaces must not be used elsewhere in the global zone.
  ```
  zone1-zonecfg: add net physical=bge1
  zone2-zonecfg: add net physical=bge2
  ```
- The zone configuration for zone1 and zone2 is converted to the use of exclusive IP instances.
  ```
  zonecfg: set ip-type=exclusive
  ```
- IP addresses and the default router are specified in the zones in the usual way.
  ```
  Zone 1: /etc/hostname.bge1
  Zone 2: /etc/hostname.bge2
  /etc/defaultrouter
  ```
- Communication between the zones or between the zones and the global zone takes place only if corresponding routing entries exist. Additionally a physical network connection has to exist between the interfaces of the zones.
- The default router is a NAT router that hides the IP address of the local zone from the customer. On the customer's side, it is configured with an IP address from the customer's network; thus, address conflicts can not occur.

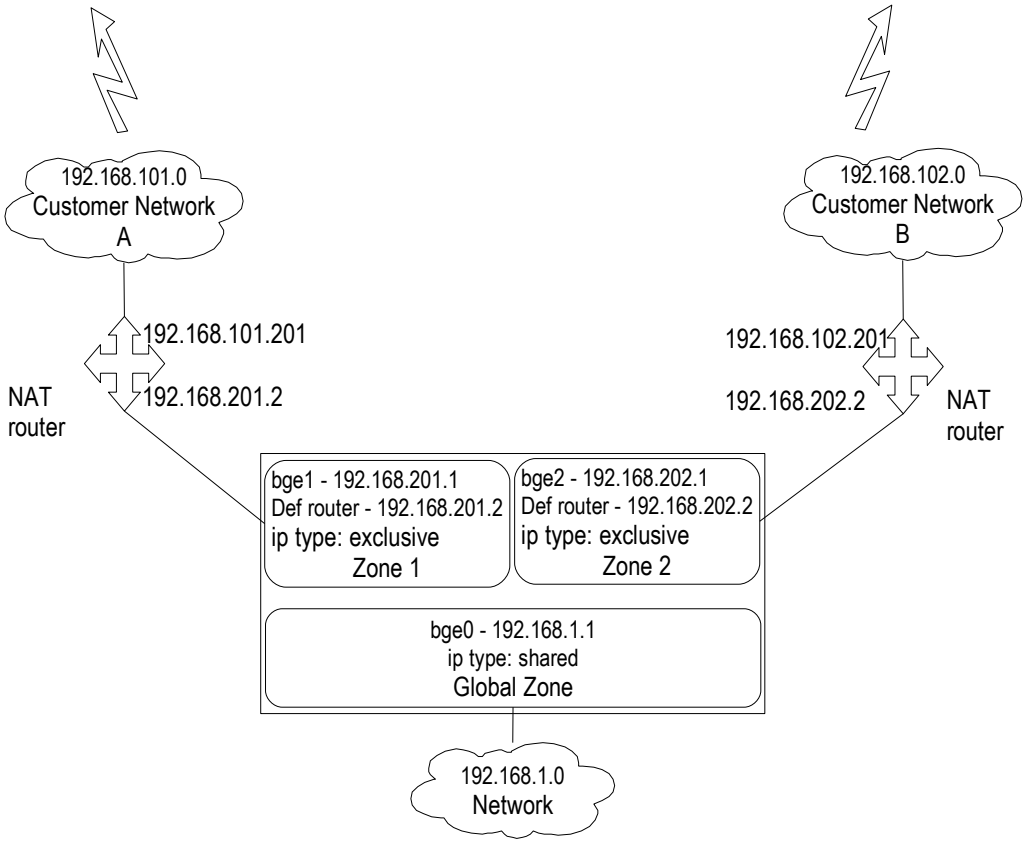


Figure 37: [dd] Zones connected to independent customer networks using exclusive IP instances

5.2.7.8. Connection of zones via external routers using the shared IP instance

[dd/ug] A web server in zone1 is contacted from the internet and needs the application server in zone2 to fulfill the orders.

- Zone1 should be connected to the internet through a separate network.
- The connection from zone1 to zone2 should take place through an external load balancing router. For reasons of clarity, no additional instances for web and application servers are contained here.
- Direct communication between the local zones should not be possible, but rather through the external router instead.
- Communication between the global zone and the local zones is not intended.

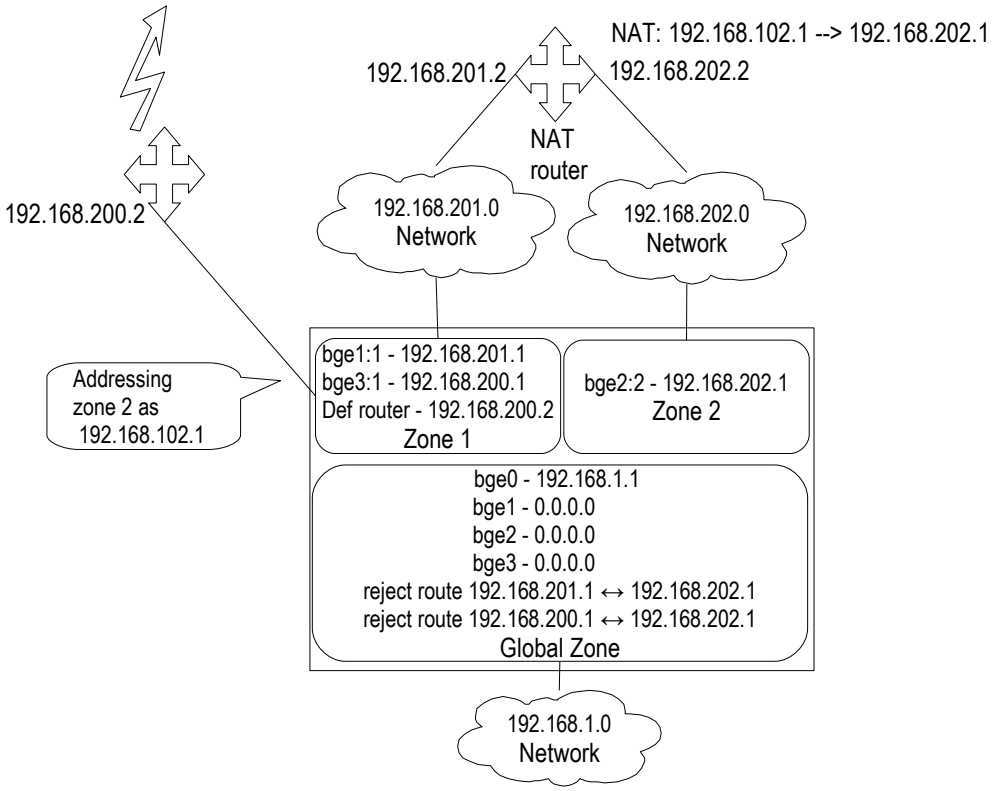
Implementation:

- The network interfaces provided for the local zones (`bge1, bge2` and `bge3`) must not be used elsewhere in the global zone.
- To prepare for local zones, the interfaces must be plumbed (but not enabled); thereby, the interfaces receive the address 0.0.0.0:
  ```
  ifconfig bge1 plumb down
  ifconfig bge2 plumb down
  ifconfig bge3 plumb down
  ```
- The network configuration of the zones is established by setting the zones to the `ready` status.
  ```
  zoneadm -z zone1 ready
  zoneadm -z zone2 ready
  ```
  The addresses listed in the zone configuration are now active.
  (`zone1: 192.168.201.1,192.168.200.1` and `zone2:192.168.202.1`)
- A default route is specified for communication of the zone zone1 to the internet.
  ```
  zonecfg:set defrouter=192.168.200.2
  ```
  In addition, a route is required to the apparent address of zone2 behind the NAT router.
  ```
  route add 192.168.102.0 192.168.201.2
  ```

- In order to avoid communication between the local zones through the shared TCP/IP stack, reject routes must be set in the global zone that prevent communication between the IP addresses of the two zones (or the use of ipfilter).
```
route add 192.168.201.1 192.168.202.1 -interface -reject
route add 192.168.202.1 192.168.201.1 -interface -reject
route add 192.168.200.1 192.168.202.1 -interface -reject
route add 192.168.202.1 192.168.200.1 -interface -reject
```
- Zones can now be booted up for operation:
```
zoneadm -z zone1 boot, zoneadm -z zone2 boot
```
- The reject route leads to the complete prevention of communication between zone1 and zone2 which, however, is required in this scenario according to the above specifications.
  Therefore, the configured default router must support NAT. It must convert the address `192.168.102.1` into the address `192.168.202.1`.
  Communication via the NAT router thereby bypasses the reject routes.
- Option: To allow communication between the global and the local zone, an interface that is located in the logical network of the local zone must be configured in the global zone.

The procedure is as follows:

- An HTTP request is made to zone zone1 from the outside.
- It is able to process parts of the request by itself but another part must come from the application server that is addressed via the address 192.168.102.1.
- This address is routed via the NAT router which converts the address into the address 192.168.202.1 on the other side.
- This is the address of zone zone2 which carries the application server that processes the missing parts of the request and sends them back through the existing connection.



5.2.7.9. Connection of zones through an external load balancing router using exclusive IP instances

[dd/ug] A web server in zone1 is contacted from the internet and needs the application server in zone2 to fulfill the orders.

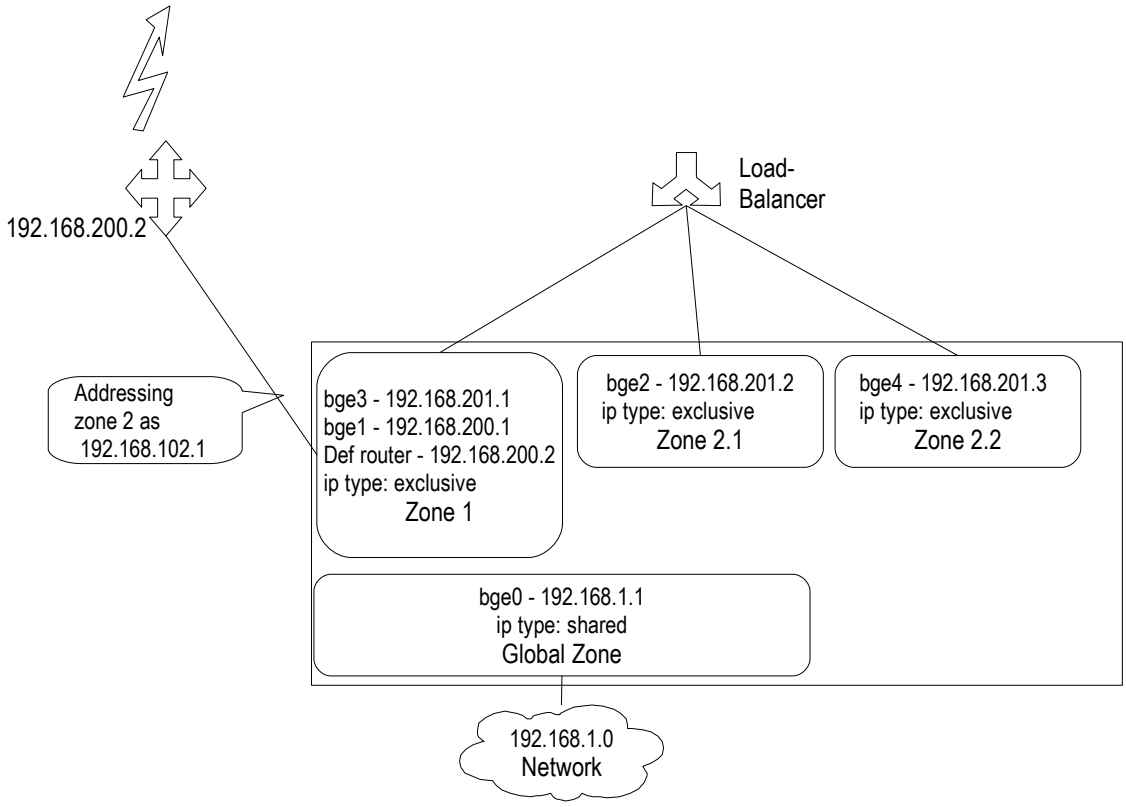
- Zone1 should be connected to the internet through a separate network.
- The connection from zone1 to zone2 should take place through an external load balancing router. For reasons of clarity, no additional instances for web and application servers are contained here.
- Direct communication between the local zones should not be possible but rather through the external router instead.
- Communication between the global zone and the local zones is not intended.

Implementation:

- A separate GLDV3 interface (e.g. *bge1*, *bge2*) is provided for each zone. In addition, *bge3* is assigned to zone1 for direct communication with zone2. These interfaces must not be used elsewhere in the global zone.
  ```
  zone1-zonecfg: add net physical=bge1
  zone1-zonecfg: add net physical=bge3
  zone2-zonecfg: add net physical=bge2
  ```
- The zone configuration for zone1 and zone2 is converted for the use of exclusive IP instances.
  ```
  zonecfg: set ip-type=exclusive
  ```
- In the zone zone1, the IP addresses and the default router are specified in the usual way. Zone2 does not require a default route since the only communication intended is that with zone1. *bge3* and *bge2* are connected through a load balancing router and configured accordingly.
  ```
  Zone 1: /etc/hostname.bge1
  Zone 1: /etc/hostname.bge3
  /etc/defaultrouter
  Zone 2: /etc/hostname.bge2
  ```
- To increase security, the communication between zone1 and zone2 can in addition also be filtered with a separate firewall. Through the use of exclusive IP instances, communication between the zones or between the zones and the global zone will only take place if corresponding routing entries exist in the zones, and if a physical network connection exists between the interfaces of the zones.

The procedure is as follows:

- A HTTP request is made to zone1 from the outside.
- It is able to process parts of the request by itself but another part must come from the application server that can be reached via the address 192.168.102.1 in zone2.
- This address is reached via the interface *bge3*.
- Further cascading via additional zones, zone2.1, zone2.2, zone2.3, etc. is possible.


192.168.200.2
Load-
Balancer
Addressing
zone 2 as
192.168.102.1
bge3 - 192.168.201.1
bge1 - 192.168.200.1
Def router - 192.168.200.2
ip type: exclusive
Zone 1
bge2 - 192.168.201.2
ip type: exclusive
Zone 2.1
bge4 - 192.168.201.3
ip type: exclusive
Zone 2.2
bge0 - 192.168.1.1
ip type: shared
Global Zone
192.168.1.0
Network

5.3. Lifecycle management

5.3.1. Booting a zone

[dd] `zoneadm -z <zone> boot` starts up a zone, mounts the file systems, initializes the network interfaces, sets the resource controls and starts the service manager of the zone. When the zone is first started, as with Solaris re-installation, all smf service manifests are imported and the initial smf repository for the zone is created.

```
global# zlogin -C zone1
[Connected to zone 'zone1' console]

[NOTICE: Zone booting up]


SunOS Release 5.10 Version Generic_118855-15 64-bit
Copyright 1983-2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: zone1

zone1 console login:
```

In addition, the directory `<zonepath>/dev` is created, which contains all devices that the zone can access.

```
global# ls /zones/zone1/dev
arp         dtrace      msglog      rdsk        syscon      ticlts      vt00
conslog      dtremote    null          rmt           sysevent    ticots
zconsole
console     fb0         poll        sad         sysmsg      ticotsord   zero
cpu         fd          pool        stderr      systty      tty         zfs
crypto      kstat       ptmx        stdin       tcp         udp
cryptoadm   log         pts         stdout      tcp6        udp6
dsk         logindmux   random      swap        term        urandom
```

5.3.2. Boot arguments in zones

[dd] The zone configuration, just as the command `zoneadm -z <zone> boot`, can be extended by boot arguments (`bootargs`). Thus for example a singular boot process with detailed information on troubleshooting, or booting the zone into single-user mode, is possible.

```
global# zoneadm -z keetonga boot -s
global# zlogin -C keetonga
[Connected to zone 'keetonga' console]

[NOTICE: Zone booting up]


SunOS Release 5.10 Version Generic_137138-09 64-bit
Copyright 1983-2008 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Booting to milestone "milestone/single-user:default".
Hostname: keetonga
Requesting System Maintenance Mode
SINGLE USER MODE

Root password for system maintenance (control-d to bypass):
```

Alternatively, set the boot arguments permanently in a zone configuration:

```
global# zonecfg -z keetonga
zonecfg:keetonga> info bootargs
bootargs:
zonecfg:keetonga> set bootargs="-m verbose"
zonecfg:keetonga> info bootargs
bootargs: -m verbose
zonecfg:keetonga> commit
zonecfg:keetonga> exit
global# zoneadm -z keetonga halt
bash-3.00# zlogin -C keetonga
[Connected to zone 'keetonga' console]

[NOTICE: Zone booting up]

SunOS Release 5.10 Version Generic_137138-09 64-bit
Copyright 1983-2008 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
[ system/filesystem/root:default starting (root file system mount) ]
[ network/loopback:default starting (loopback network interface) ]
[ network/pfil:default starting (packet filter) ]
[ system/installupdates:default starting (system update installer) ]
[ system/boot-archive:default starting (check boot archive content) ]
[ milestone/name-services:default starting (name services milestone) ]

... deleted outputs ...

[ network/inetd:default starting (inetd) ]
[ system/system-log:default starting (system log) ]
[ system/console-login:default starting (Console login) ]

keetonga console login:
```

5.3.3. Software installation per mount

[ug] In order to allow an application to run in the zone, additionally installed software is required, except in trivial cases (Apache, Perl, etc.) This software is very often not in Sun pkg format.

One method of distributing this software is to maintain a shared software repository in the global zone and to make it completely or partially visible in the zones:

- Globally, there is the directory `/software`
- Software is installed in the directory `/software/product/version/` e.g.:
  - `/software/oracle/9i_05/`
  - `/software/oracle/9i_06/`
  - `/software/oracle/10g_01/`
  - `/software/siebel/3.14/`
  - `/software/tomcat/3.7/`
- With `zonecfg: add fs`, the required directories in the respective zones are made visible as read-only (with lofs).

This ensures that even software usage can be tracked in order to monitor licensing fees if necessary.

It is easier to install the entire directory `/software` in each zone. Different ways will then need to be used to monitor software usage (*audit* and *dtrace*).

5.3.4. Software installation with provisioning system

[ug] The N1 SPS software can provision software in zones as well. The requirements are:

- A writable directly where the software can be installed. This can be `/opt`. `/opt` is not allowed to be configured as `inherit-pkg-dir` in this case.
- If the software need to be installed under `/usr` or in a different system directory, this can be implemented either with a whole-root zone or with a writable directory (e.g. `/usr/local`) available in the `/usr` - tree (created with `zonecfg:add fs`).
- The software needs to be able to run in a zone (see above).

5.3.5. Zone migration among systems

[dd] Starting with Solaris 10 11/06, zones can be moved among physical systems. This can be achieved with `zoneadm detach/attach` . The following conditions must be observed before zones can be migrated:

- Zones must be stopped prior to migration
- The patch and package status between the two systems must be identical

A local zone can be migrated as follows between systems:

- Detach the zone with `zoneadm -z <zone> detach`
- Move the `zonepath` directory to the target system, e.g. in the SAN or with `tar/cp`
- Set up the zone configuration `zonecfg -z <zone> create -a <zonepath>`
- Attach the zone `zoneadm -z <zone> attach`

An installed local zone is detached prior to migration. This process generates all information required to attach this zone to another system. The information is stored in the file `<zonepath>/SUNWdetached.xml`. The status of the zone is converted from `"installed"` to `"configured"`. Thus, this zone will subsequently no longer be considered for package installations, patches, booting, etc.

Prior to attaching, the zone configuration can be generated from the `SUNWdetached.xml` file. Running `attach` will check on the target system whether the local zone fits on the target system. In addition, it is checked whether the installed packages and patches in the migrated zone and the global zone of the target system have the same version.

5.3.6. Zone migration within a system

[ug] Let us assume that a zone named "test" is to be moved to another directory. Currently, this zone is located on /export/home/zone/test (zonepath).

```
global# zoneadm list -vc
  ID NAME       STATUS     PATH                       BRAND    IP
   0 global     running    /                          native   shared
  22 test       running    /export/home/zone/test     native   shared
```

Before moving it, the zone must be halted:

```
global# zoneadm -z test halt
```

In a short amount of time, the zone can then be moved with `zoneadm move`. The duration depends on whether the target directory is located in the same filesystem (implementation with mv) or in a different filesystem (zonepath must be copied), and depending on the contents (sparse-root/whole-root zone). The move takes several minutes.

```
global# zoneadm  -z test  move  /container/test
Moving across file-systems; copying zonepath /export/home/zone/test...
Cleaning up zonepath /export/home/zone/test...
global#
```

The configuration of the zone is adjusted as well:

```
global# zonecfg -z test info
zonename: test
zonepath: /container/test
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
inherit-pkg-dir:
        dir: /opt

global# zoneadm list -vc
  ID NAME           STATUS     PATH                     BRAND    IP
   0 global         running    /                        native   shared
  23 test           running    /container/test          native   shared
```

In this example, the zone was moved from a UFS to a ZFS directory.

```
global# df -k /export/home /container
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c1t1d0s6    5848297 2839898 2949917    50%    /export/home
container            1007616   90758  916031    10%    /container
```

5.3.7. Duplicating zones with zoneadm clone

[ug] Zone installation can be accelerated with *zoneadm ... clone*. In this example, a zone named test is already configured and installed.

```
global# zoneadm list -vc
  ID NAME        STATUS     PATH                         BRAND    IP
   0 global      running    /                            native   shared
   - test        installed  /container/test              native   shared
```

This zone serves no as the basis for another zone installation. The configuration of the zone test is as follows:

```
global# zonecfg -z test info
zonename: test
zonepath: /container/test
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
inherit-pkg-dir:
        dir: /opt
```

To copy the configuration, it is selected with *zonecfg ... export* and the new zone is created from it directly. Only the zonepath still needs to be adjusted; this is done with the single-line variant of zonecfg .

```
global# zonecfg -z test export  |  zonecfg -z test1
test1: No such zone configured
Use 'create' to begin configuring a new zone.

global# zonecfg -z test1 set zonepath=/container/test1
```

Now, zone `test1` is configured in exactly the same way as zone `test` but has its own `zonepath`.

```
global# zonecfg -z test1 info
zonename: test1
zonepath: /container/test1
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
inherit-pkg-dir:
        dir: /opt

global# zoneadm list -vc
  ID NAME         STATUS     PATH                         BRAND    IP
   0 global       running    /                            native   shared
   - test         installed  /container/test              native   shared
   - test1        configured /container/test1             native   shared
```

Next, zone `test1`, which so far has only been configured, can be installed by duplicating the zone `test`. Zone test1 can be used immediately afterwards.

```
global# zoneadm -z test1 clone test
Cloning zonepath /container/test...
grep: can't open /a/etc/dumpadm.conf
global#

global# zoneadm -z test1 boot
root@se-ffm-03-um-test [283] # zoneadm list -vc
  ID NAME         STATUS     PATH                         BRAND    IP
   0 global       running    /                            native   shared
  25 test1        running    /container/test1             native   shared
  - test          installed  /container/test              native   shared
```

In this example, no network is configured for zone `test`, which is of course unrealistic in thereal world. If a network is configured, it must be of course reconfigured after configuring the zone `test1` like the zonepath was reconfigured, since otherwise both zones will have the same network address and be unable to run simultaneously.

5.3.8. Duplicating zones with zoneadm detach/attach and zfs clone

[ug] First, the zone "test" is moved to its own ZFS file system. The file system must only be available from root otherwise an error message will appear.

```
global# zfs create container/ztest
global# ls -ld /container/ztest
drwxr-xr-x   2 root     root           2 Feb 19 16:00 /container/ztest
global# chmod og-rwx /container/ztest
global# ls -ld /container/ztest
drwx------   2 root     root           2 Feb 19 16:00 /container/ztest
```

Next, the zone is moved to the separate ZFS file system. The configuration is adjusted automatically; to do so, it must of course be shut down.

```
global# zoneadm -z test move /container/ztest
Moving across file-systems; copying zonepath /container/test...
Cleaning up zonepath /container/test...
```

The zone is detached, and the current status of the file system is recorded with `zfs snapshot` as a read-only version (the gold standard). The original zone can then be activated again.

```
global# zoneadm -z test detach
global# zfs snapshot container/ztest@gold
global# zoneadm -z test attach
global# zoneadm -z test boot
```

A copy of the zone is created by generating a writable version of the zone file system with `zfs clone`.

```
global# zfs clone container/ztest@gold container/ztest2
global# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
container              181M   803M  88.7M  /container
container/ztest       91.8M   803M  88.7M  /container/ztest
container/ztest@gold  3.07M      -  90.4M  -
container/ztest2          0   803M  90.4M  /container/ztest2
```

In order to activate the zone copy, it must be configured, and the zone path must be adjusted. Subsequently, it is sufficient to use *`zoneadm attach`* and the zone can be started up.

```
global# zonecfg -z test export | zonecfg -z test2
test2: No such zone configured
Use 'create' to begin configuring a new zone.
global# zonecfg -z test2 set zonepath=/container/ztest2
global# zoneadm -z test2 attach
global# zoneadm -z test2 boot
```

In the zone, the name of the OS instance is still the same of course since the file `/etc/nodename` has not yet been changed. Furthermore, it makes sense to adapt the network addresses as well, just like the zonepath adaptation performed here.

Additional zones with the same OS content can now be created very quick by repeating the steps *`zfs clone`* / *`zonecfg`* / *`zoneadm attach`*.

In the same way, zones can also be moved to other systems with *`zoneadm detach / attach`*. A prerequisite is that the same package and patch status is present on the target system. By using the option *`zoneadm attach -u,`* a zone can be moved to a system with a newer status, however, downgrading of patches must not occur.

Since Solaris 10 5/09, *`zfs clone`* is used automatically with the command *`zoneadm -z zone clone`* if the zone is located on a ZFS file system. This facilitates the quick creation of zones even more.

5.3.9. Moving a zone between a sun4u and a sun4v system

[ug] Currently, two architectures with SPARC-processors are available from Sun Microsystems that are both supported by Solaris 10. The sun4u architecture in the larger data center computers with higher single processor performance, and the sun4v architecture in the CMT computers (Chip Multi Threading) with many cores and threads as well as virtualization support. The computer architectures differ slightly at the HW/SW interface, a fact that is reflected in a Solaris 10 installation by different contents in the filesystem.

It can be necessary in data centers for zones to be moved among the various architectures.

First, a zone is created as a template on the sun4u system `tiger;` the configuration looks as follows:

```
root@tiger [6] # zonecfg -z template-u info
zonename: template-u
zonepath: /zone/template-u
brand: native
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
inherit-pkg-dir:
        dir: /opt
```

The installation:

```
root@tiger [8] # zoneadm -z template-u install
Preparing to install zone <template-u>.
Creating list of files to copy from the global zone.
Copying <2968> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1515> packages on the zone.
Initialized <1515> packages on zone.
Zone <template-u> is initialized.
The    file    </zone/template-u/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

Next, the configuration is saved for reuse as a file (under /zone):

```
root@tiger [18] # zonecfg -z template-u export >template-u.export
```

A new zone with the same configuration can be created as follows on the sun4u system tiger:

```
root@tiger [19] # zonecfg -z u0 <template-u.export
u0: No such zone configured
Use 'create' to begin configuring a new zone.

root@tiger [20] # zonecfg -z u0 set zonepath=/zone/u0

root@tiger [21] # zoneadm -z u0 clone template-u
Cloning zonepath /zone/template-u...
```

Next, the zone is to be transported to a sun4v system named `bashful`. To do so, the contents and the configuration are saved:

```
root@tiger [23] # cd /zone
root@tiger [23] # tar cEvf u0.tar u0                # zone sichern

root@tiger [24] # zonecfg -z u0 export >u0.export  # Konfiguration
```

The zone is created by means of the saved configuration:

```
root@bashful [40] # zonecfg -z u0v </net/tiger/zone/u0.export
u0v: No such zone configured
Use 'create' to begin configuring a new zone.
```

The root of the zone is installed from the tar-archive:

```
root@bashful [41] # cd /zone

root@bashful [42] # tar xEf /net/tiger/zone/u0.tar
```

And the last step is adapting the HW-architecture:

```
root@bashful [43] # zoneadm -z u0v attach -u /zone/u0
zoneadm: zone 'u0v': The zone was not properly detached.
        Attempting to attach anyway.
WARNING: package operation in progress on the following packages:
    SUNWcsr SUNWcslr SUNWmlibk SUNWmlibl TSBWvplr FJSVs8brandr FJSVhea
SUNWxwplt   SUNWxorg-cfg   SUNWxwinc   SUNWxorg-xkb   SUNWxorg-server
SUNWxorg-client-docs   SUNWxorg-doc   SUNWxorg-devel-docs   SUNWxwpmn
SUNWapchu SUNWapchd SUNWtcatu SUNWapch2u SUNWapch2d SUNWcsl SUNWcsu
SUNWdtrc   SUNWPython   SUNWwbdev   SUNWdtcor   SUNWdtbas   SUNWnamdt
SUNWupdatemgru SUNWdtdte SUNWdttsu SUNWodtts SUNWdtdst SUNWdtma
SUNWhea   SUNWcry   SUNWfirefox-devel   SUNWfmd   SUNWgnome-panel-devel
SUNWiiimu SUNWlxsl SUNWlxml SUNWprd SUNWj5rt SUNWj5rt:none SUNWj5rtx
SUNWj5dmo SUNWj5dmx SUNWj5dev SUNWj5man SUNWbzip SUNWesu SUNWmdu
SUNWfirefox SUNWnfssu SUNWzfsu SUNWkrbu SUNWgnome-libs SUNWipfu
SUNWkdcu SUNWnisu SUNWdpl SUNWpsu SUNWmdb SUNWpr SUNWpprou SUNWppro-
plugin-sunos-base SUNWpcu SUNWsmcmd SUNWlvma SUNWsshcu SUNWmcos
SUNWgnome-desktop-prefs SUNWxwice SUNWxwfnt SUNWxwopt SUNWxwacx
SUNWxim SUNWxi18n FJSVxwpsr SUNWplow SUNWtltkm TSBWvplu SUNWppm
SUNWdclnt SUNWwbmc SUNWlvmg SUNWwbapi SUNWwbcou SUNWmgapp SUNWtlsu
SUNWsfwhea SUNWopenssl-include SUNWimagick SUNWsmbau SUNWgnome-img-
editor SUNWgcmn SUNWxwrtl SUNWPython-share SUNWsfman SUNWgnome-panel-
share SUNWgnome-desktop-prefs-share SUNWbind SUNWgnome-themes-share
SUNWgnome-img-viewer-share SUNWgnome-display-mgr-share SUNWgnome-img-
editor-share SUNWgnome-libs-share SUNWjss SUNWgnome-file-mgr-share
SUNWgnome-session-share SUNWxvnc SUNWmctag SUNWmcon SUNWucbt SUNWxcu6
Getting the list of files to remove
Removing 5 files
Remove 53 of 53 packages
Installing 12 files
Add 21 of 21 packages
Updating editable files
The file </var/sadm/system/logs/update_log> within the zone contains a
log of the zone update.
```

The last lines show the adaptation.

For this test, two additional packages had originally been installed on the computer "tiger" that had to be removed first (command `pkgrm`) for `zoneadm attach -u` to work. Thus, it is important for the package and the patch status to be identical if possible, or that the target system represents an upgrade and not a downgrade in any package, because the update process is not able to handle that.

5.3.10. Shutting down a zone

[dd] Zones can be shut down from the local zone itself or from the global zone. Depending on which option is used, running services are either completed or simply stopped.

- `zoneadm -z zone halt` is called in the global zone, halts a zone and stops all processes in the zone immediately.
- `zlogin <zone> init 5` is called in the global zone, changes over to the zone and shuts down the zone using the service manager, thus ending all processes properly.
- `init 5` is called in the zone itself, shuts down the zone using the service manager and stops all processes properly.

5.3.11. Using live upgrade to patch a system with local zones

[dd] According to *4.4.2 Patching a system with local zones*, systems with local zones can be patched with of live upgrade.

To be able to use live upgrade , several additional terms will need to be defined first:

- Boot environment (BE): the sum total of all OS and application software as well as file systems of a Solaris instance
- Shared file system: a file system that is used (mounted) by more than one BE
- Non-shared file system: a file system that is used by only one BE

The following is assumed:

- Solaris 10 8/07 (or higher) is already installed
- All currently required patches according to Infodoc 72099 are installed
- The OS is installed in a file system (f.e. `/dev/dsk/c1t0d0s0`)
- The current BE contains another file system mounted to /shared (`/dev/dsk/c1t0d0s3`)
- `/shared` must be entered in the `/etc/vfstab`, so that live upgrade will subsequently be able to recognize the filesystem as shared
- Two sparse-root zones are installed, one in a non-shared and one in a shared filesystem.
  - zone1: zonepath=/non-shared/zone1
  - zone2: zonepath=/shared/zone2

First, a copy of the current BE is created.

```
lucreate -c s10-807 -n s10-807+1 -m /:/dev/dsk/c1t0d0s4:ufs
```

Shared file systems and swap are not copied automatically. If necessary, the copy can be configured per `lucreate` parameter. Furthermore, the name assigned to the current BE here is s10-807 and for the BE to be created the name assigned is s10-807+1. By `lucreate`, the content is copied from /non-shared/zone1 to /non-shared/zone1 of BE s10-807+1. Since zone2 is located in a shared file system, the zone within this file system is copied to /shared/zone2-s10-807+1. This ensures that each BE uses the correct zonepath from the shared file system. It must therefore be ensured that there is sufficient room available in the shared file system.

Next, the inactive BE s10-807+1 including its zones can be patched, while production continues to run on the active BE. The patches are installed as follows:

```
luupgrade -t -n s10-807+1 -s <Path to Patches> <patch1> <patch2>
```

Note: Those who prefer to perform a live upgrade on the BE on a newer Solaris instead of patching must use the following command:

```
luupgrade -u -n s10-807+1 -s <Path to install OS-image>
```

For x86/x64 systems, another step is required to update the boot archive since drivers or system files may also have been modified by installing the patch. First, the new BE has to be mounted:

```
lumount s10-807+1
```

The BE is now available e.g. under /.alt.s10-807+1. Next, the boot archive of this BE is updated and the BE is unmounted again.

```
bootadm update-archive -R /.alt.s10-807+1
luumount s10-807+1
```

Finally, the new BE can be activated. To run the new BE, the total system must be restarted. It is important that live upgrade performs a final synchronization when turning the current BE off. It is therefore critical that the command `init 6` is used (**under no circumstances the command** `reboot`).

```
luactivate s10-807+1
init 6
```

When the new BE starts up for the first time, the global zone, as well as the "old" and the "new" zones, are synchronized according to `/etc/lu/synclist`. To do so, live upgrade needs access to the "old" and the "new" zones when starting the new BE up for the first time.It must be ensured that the shared file systems are also mounted automatically in the new BE when booted. This is required for the proper completion of live upgrade.

Now, the patching of zones with live upgrade is completed and all zones are operational.

5.4. Management and monitoring

5.4.1. DTrace in a local zone

[dd] Since Solaris 10 11/06, DTrace can be applied within local zones to processes of this zone. To enable DTrace, it is necessary to extend the set of privileges for the local zone with `dtrace_proc` and `dtrace_user`. Without these privileges, no DTrace probes will be available in the zone.

No DTrace probes available inside of zone1:

```
zone1# dtrace -l | tail +2
zone1#
```

Adding DTrace capability to zone configuration:

```
global# zonecfg -z zone1
zonecfg:zone1> set limitpriv=default,dtrace_proc,dtrace_user
zonecfg:zone1> commit
zonecfg:zone1> exit
```

For example, the pid provider can be used to trace a process in the own zone. `dtrace -n 'pid<pid>:::entry {trace(probefunc)}'`

5.4.2. Zone accounting

[ug] With the command `acctadm`), extended accounting can be switched on. In the predefined resource profile `extended`, also the name of the zone is written to the accounting records. This allows accounting data to be associated to their respective zones. It is possible to and to summarily account for zone consumption without elaborately having to assign the commands to applications, as required in traditional Unix accounting.

With Solaris 10, a library (*libexacct(3LIB)*) and an example program (*/usr/demo/libexacct/*) are included that allow the accounting records to be analyzed easily.

5.4.3. Zone audit

[dd] Audit can be used in two different ways regarding local zones:

- Audit is configured in the global zone. By setting the `zonename` policy in `/etc/security/audit_startup`, audit enters the zone name in each audit record. With `auditreduce -z <zonename>` , the corresponding audit records are extracted and can be analyzed with `praudit`. The configuration and collection of audit data is done completely from the global zone.
- Audit is configured in the global zone. In addition, the `perzone` policy is set in `/etc/security/audit_startup`. Thereby, each zone starts its own `auditd` and keeps its own configurations and log files per zone. Control of the audit configuration is assigned to the administrator of the local zone.

When auditing is needed, the decision for one of the two configuration options will be done depending on control an access standards of the datacenter operations.

5.5. Resource management

5.5.1. Limiting the /tmp-size within a zone

[dd] In many cases, `/tmp` is used as `tmpfs` in swap. This leads to the swap area being shared by all zones by /tmp in each zone. Although the `/tmp` areas are visible to each zone itself, global resources are used. To limit the use of space, `/tmp` should be mounted in the `/etc/vfstab` of zones with the `size` option.

```
swap    -       /tmp    tmpfs   -       yes     size=250m
```

However, this limitation can be changed by the root administrator of the zone.

Yet, this area is counted as virtual memory of the zone and is therefore also limited by setting `zonecfg:add capped-memory: set swap= <value>`.

5.5.2. Limiting the CPU usage of a zone (CPU capping)

[dd] Since Solaris 10 5/08, it is possible to cap the CPU time for a zone. CPU shares can be specified as potential capping values. Thus, the value 1.5, for example, represents a whole CPU and a half. With respect to the syntax, care must be taken to specify `set ncpus=1.5` (locale=C) or `set ncpus=1,5`, depending on the locale setting of the session by the administrator. The setting is accurate to 1/100 CPUs.

```
zonecfg:keetonga> add capped-cpu
zonecfg:keetonga:capped-cpu> set ncpus=1.5
zonecfg:keetonga:capped-cpu> end
zonecfg:keetonga> info
capped-cpu:
     [ncpus: 1.50]
rctl:
     name: zone.cpu-cap
     value: (priv=privileged,limit=150,action=deny)
```

This setting is a hard limit, that is to say, the set CPU capacity cannot be exceeded.

5.5.3. Resource pools with processor sets

[ug] In Solaris (since Solaris 9), CPUs can be portioned out to resource pools. Solaris zones can be assigned to these resource pools; this allows CPU resources to be separated easily:

- With the commands `poolcfg` and `pooladm`, resource management is activated and the additional resource pool is created.
- With the command `poolcfg`, the processor set (`pset`) is created and assigned to the resource pool.
- With the command `zonecfg`, the attribute `pool` of the zone can be changed and the new resource pool can be set there. This will then be the setting that is valid if the zone is rebooted without further measures.
- With the command `poolbind`, the allocation of zones to resource pools can be changed dynamically. This has no effect on the default setting done with `zonecfg`; after a restart of the zone the setting from the zone configuration is re-established.

The processes of a zone that are allocated to a resource pool will then only run on the processors that belong to the processor set of this resource pool.
If several zones are allocated to a resource pool, the ratio of the CPU time used can be adjusted between the zones with the fair share scheduler.
The number of CPUs assigned to the processor set is a hard limit and cannot be exceeded by the processes of the zone

5.5.4. Fair share scheduler

[ug] The ratio of CPU usage between zones or projects can be set.

This is implemented by he so-called fair share scheduler. CPU shares are allocated as follows:

- For zones, by using `add rctl` and the attribute `zone.cpu-shares` in the `zonecfg` command.
- For projects, this is set in the project database (`/etc/project`, or NIS or LDAP) and the attribute `project.cpu-shares` (global and/or local zone).
- The zones/projects must be allocated to the same resource pool.
- In the resource pool, the fair share scheduler must be activated.
- In the global zone, the resource pool can also be limited to a subset of processors in the system.

The operating system provides for fairness, if the CPUs of the resource pool are used to capacity. The allocated shares of the projects and the zones with active processes are used to calculate the nominal value for CPU usage. The nominal value of CPU usage is calculated with the formula: (zone shares) * (number of CPUs) / (sum of the shares of the active zones).

If the CPU usage approaches 100% and the amount of consumed CPU time is larger than the nominal value, the fair share scheduler takes countermeasures by reducing priority.

Furthermore:

- The settings for CPU `shares` can be dynamically set with the command `prctl`.
- The affiliation of a zone with a resource pool can be changed at any time with the command `poolbind`.

If CPU usage in a resource pool is noticeably smaller than 100%, the fair share scheduler does not control CPU usage. In this situation, a zone in the resource pool can use more than it is actually entitled to. If this is not desired, additional limitation with processor sets or cpu capping is recommended.

5.5.5. Static CPU resource management between zones

[ug] Static resource management between zones determines how resources should be distributed in a normal case following a booting routine:

- To create a resource pool, the commands `poolcfg` and `pooladm` are used.
- The affiliation of a zone to a resource pool for `zonecfg` can be set with the attribute `pool`.
- The settings for fairness between the zones are adjusted with `add rctl` in `zonecfg`. We recommend setting `zone.cpu-shares` (CPU share) and additionally `zone.max-lwps` (maximum number of threads).

5.5.6. Dynamic CPU resource management between zones

[ug] This refers to commands that allow the user to reset resource controls while the zones are running in order to react to a temporary load situation.

The commands `prctl` and `poolbind` are used. If necessary, new resource pools can temporarily be created with `poolcfg` and `pooladm` to separate loads.

5.5.7. Static CPU resource management in a zone

[ug] The ratio of CPU usage between applications in a zone can be adjusted as well.

This is done with the definition of resource pools within the zone and use of the fair share scheduler with projects inside of the zone. Allocation of CPUs to resource pools is not possible within the zone.

- The allocation of applications to projects is done by configuration in the files `/etc/project` and `/etc/user_attr` .
- In `/etc/project`, it can be defined which resource pool a project should run in.
- To create a resource pool in the zone, the commands `poolcfg` and `pooladm` are used. However, processors cannot be assigned.
- In the resource pool, the fair share scheduler (FSS) must be activated.

5.5.8. Dynamic CPU resource management in a zone

[ug] This refers to commands used to reset the resource controls within the zone while running, in order to react to a temporary load situation.

The commands `prctl` and `poolbind` can be used to change share values of projects or associations of project to resource pools. If necessary, new resource pools can temporarily be

created with `poolcfg` and `pooladm`.

5.5.9. Dynamic resource pools for zones

[dd] As already described in *4.6.2.5 Dynamic resource pools*, dynamic resource pools can very easily be used for zones since Solaris 10 8/07. The number of CPUs required is specified in the zone configuration. If the corresponding zone is started up, a temporary resource pool is created and assigned to the starting zone. This resource pool will exist for the duration of the zone and will also be destroyed once the zone is turned off.

CPUs can be specified by a fixed number of CPUs (e.g. `set ncpu=10`). In this case, this number must be made available to the zone at the time it starts up and will also remain assigned to it for the duration of its runtime.

If a CPU range (e.g. `set ncpus=10-14`) is stated, the minimum number of CPUs must be available at the time the zone is started up. During the zone's runtime, the `poold` of the zone assigns CPUs dynamically depending on demand by the zone and depending on the availability of CPU in the system. (For `poold` to run, the service `/system/pool/dynamic` must be activated.) If several resource pools compete for available CPU, a decision is made based on `importance` as to which resource pool is served preferentially.

```
global # svcs dynamic
STATE          STIME    FMRI
online         Jan_19   svc:/system/pools/dynamic:default
global # zonecfg -z zone1
zonecfg:zone1> add dedicated-cpu
zonecfg:zone1:dedicated-cpu> help
zonecfg:zone1:dedicated-cpu> set ncpus=10-14
zonecfg:zone1:dedicated-cpu> set importance=30
zonecfg:zone1:dedicated-cpu> end
zonecfg:zone1> commit
zonecfg:zone1> exit
```

Once the zone is started, poolstat can be used to display the status of available resource pools.

```
global # poolstat 5
                           pset
 id pool                  size used load
  0 pool_default            32 0.02 0.04
<<State change>>
                              pset
 id pool                  size used load
  0 pool_default            18 0.61 0.05
 12 SUNWtmp_zone1           14 0.00 0.03
                              pset
 id pool                  size used load
  0 pool_default            18 0.03 0.07
 12 SUNWtmp_zone1           14 1.40 0.08
```

If during the runtime of a zone, in the absence of automatic influence by `poold`, a zone is to have additional CPUs assigned to it, this must be done by using the command `poolcfg`. However, the change in the number of CPUs must remain within the specified CPU range.

- Increasing the number of CPUs in a resource pool

```
poolcfg -dc 'transfer 3 from pset pset_default to SUNWtmp_zone1'
```

- Reducing the number of CPUs in a resource pool

```
poolcfg -dc 'transfer 2 from pset SUNWtmp_zone1 to pset_default'
```

Adjustment of the resource pools with `zonecfg` is static and is analyzed only when the zone is started, that is, when the resource pool is created. A change of the CPU range during the runtime of the zone can be made with `poolcfg`.

```
poolcfg -dc 'modify pset SUNWtmp_zone1 (uint pset.min=15;uint
pset.max=20)'
```

5.5.10. Limiting the physical main memory consumption of a project

[dd] To limit the physical main memory of a project, the resource capping daemon *rcapd(1M)* can be used. If the resident set size (RSS) of a project exceeds the capping target (*rcap.max-rss* in bytes), *rcapd* reduces the RSS and swaps used memory pages to the paging device. *rcapd* is configured by *rcapadm(1M)* and monitored with *rcapstat(1)*. The *rcapd* can be used in zones in connection with projects and can limit the physical main memory consumption of a project within a zone.

It is recommended to limit virtual memory (called *swap*) first and then considering a limit for physical main memory, since this limit is a hard limit and is noticed more quickly in case of a misconfiguration (f.e. 20 GB cache for the database instead of 2 GB). In such a case, the sole limitation of physical memory consumption would have an extreme effect on the performance of this zone and potentially affect other zones as well.

5.5.11. Implementing memory resource management for zones

[dd] According to *4.6.3 Limiting memory resources*, the virtual, physical and locked main memory of a zone can be limited starting with Solaris 10 8/07. The configuration for this takes place in the zone configuration.

```
global # zonecfg -z zone1
zonecfg:zone1> add capped-memory
zonecfg:zone1:capped-memory> set physical=50m
zonecfg:zone1:capped-memory> set swap=200m
zonecfg:zone1:capped-memory> set locked=20m
zonecfg:zone1:capped-memory> end
zonecfg:zone1> commit
zonecfg:zone1> exit
```

The configuration of capping parameters for a zone and restart of a zone leads to the automatic start-up of the *rcapd* daemon in the global zone, which undertakes resource management. The activity of *rcapd* can be observed with *rcapstat -z*.

```
global # rcapstat -z
    id zone            nproc    vm   rss   cap    at avgat    pg avgpg
    21 zone1              29   42M   57M   50M   12M    0K 8584K    0K
    21 zone1              29   42M   57M   50M 6752K    0K 6712K    0K
    21 zone1               -   26M   29M   50M    0K    0K    0K    0K
    21 zone1               -   26M   29M   50M    0K    0K    0K    0K
    id zone            nproc    vm   rss   cap    at avgat    pg avgpg
    21 zone1               -   18M   22M   50M    0K    0K    0K    0K
    21 zone1               -   17M   18M   50M    0K    0K    0K    0K
    id zone            nproc    vm   rss   cap    at avgat    pg avgpg
    22 zone1              29   42M   57M   50M   12M    0K 8672K    0K
    22 zone1              29   42M   57M   50M   10M    0K 7776K    0K
    22 zone1               -   42M   43M   50M    0K    0K    0K    0K
```

Memory capping settings with *zonecfg* are static. If changes to the settings during the zone's runtime are required, they can be carried out using the command *rcapadm* without rebooting the zone.

```
rcapadm -z zone1 -m 40m
```

Settings for *swap* (= virtual memory), locked memory and other resource controls of a zone can be queried at runtime with `prctl -i zone <zone>`.

```
global # prctl -i zone zone1
zone: 22: zone1
NAME    PRIVILEGE       VALUE    FLAG   ACTION RECIPIENT
zone.max-swap
        privileged       200MB      -   deny     -
        system          16.0EB    max   deny      -
zone.max-locked-memory
        privileged      20.0MB      -   deny     -
        system          16.0EB    max   deny     -
zone.max-shm-memory
        system          16.0EB    max   deny     -
zone.max-shm-ids
        system          16.8M     max   deny     -
    . . .
zone.max-lwps
        system          2.15G     max   deny     -
zone.cpu-shares
        privileged          1       -   none     -
        system          65.5K     max   none        -
```

Changing these settings at runtime can be done using the command `prctl`.

```
global # prctl -r -t privileged -n zone.max-swap -v 180m -e deny -i
zone zone1
global # prctl -n zone.max-swap -i zone zone1
zone: 22: zone1
NAME    PRIVILEGE       VALUE    FLAG   ACTION    RECIPIENT
zone.max-swap
        privileged       180MB      -   deny     -
        system          16.0EB    max   deny     -
```

Supplement

A. Solaris Container in OpenSolaris

A.1. OpenSolaris – general

[dd] In 2005, Sun Microsystems started OpenSolaris as an OpenSource project in order to support and advance the developer community around Solaris (http://www.opensolaris.org/os/).

In May 2008, the first OpenSolaris operating system was completed (http://www.opensolaris.com/). The first distribution, OpenSolaris 2008.05, was addressed to individual desktop and server users especially in the x86 market, Web 2.0 developers and HPC-customers. With versions OpenSolaris 2008.11 and OpenSolaris 2009.06, further distributions were released that are oriented towards advancing the Solaris kernel and the diverse Solaris technologies.

OpenSolaris 2009.06 has been available since June 2009 not only for x86 systems but also for SPARC systems (http://www.opensolaris.com/learn/features/whats-new/200906/).

In addition to the integration of a variety of OpenSolaris projects, the OpenSolaris operating system features many conceptual innovations compared to the Solaris 10 distribution. This includes among other things:

- Distribution on a live CD
- Very easy graphic installation
- Introduction of an Internet-based package management system (Image Packaging System - IPS) (http://www.opensolaris.org/os/project/pkg/)
- New package manager for installation of IPS packages
- New update mechanism for easy upgrade of the operating system
- Automated installation by using the Automatic Installer (AI) (http://www.opensolaris.org/os/project/caiman/auto_install/)

A.1. ipkg-Branded zones

The new packaging system led to the introduction of a new zone brand that contains IPS-based zones. With respect to their functionality and use, OpenSolaris zones are comparable to Solaris 10 zones. Likewise, the resource manager functionalities of OpenSolaris are at least identical to those of Solaris 10 and in part enhanced. The following differences exist between the ipkg brand and the native brand:

- ipkg zones are very small (approx. 200 MB) and contain only the most necessary packages in their basic installation. As a result, a zone installation is very fast. Additional packages can simply be added.
- ipkg zones are currently always whole root zones. The zonecfg feature inherit-pkg-dir is geared towards SVR4 packages and cannot be applied to IPS packages. The possibilities of sparse-root zones with OpenSolaris are currently (June 2009) being discussed.
- To install a zone, the system requires access to an Internet-based package repository which is normally predefined by the global zone. During installation, however, an alternative repository can be specified as well (e.g. `zoneadm -z keetonga install -a ipkg=http://pkg.opensolaris.com:80`) that contains the packages that should be used to create the zone. Since the http protocol is used for communication, no direct connection to the Internet is required. Connection via a http proxy is sufficient.
- ipkg zones are independent of the packaging system of the global zone.

These are the differences at the current status (June 2009). The functionalities of OpenSolaris continue to be in a very dynamic development process. Therefore this list will continue to change as development progresses.

A.1. Cookbook: Configuring an ipkg zone

The configuration of the zone is done as usual with *zonecfg(1M)*.

```
root@cantaloup:~# zonecfg -z keetonga
keetonga: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:keetonga> create
zonecfg:keetonga> set zonepath=/zones/keetonga
zonecfg:keetonga> add net
zonecfg:keetonga:net> set physical=e1000g0
zonecfg:keetonga:net> set address=192.168.1.21/24
zonecfg:keetonga:net> end
zonecfg:keetonga> info
zonename: keetonga
zonepath: /zones/keetonga
brand: ipkg
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid:
net:
     address: 192.168.1.21/24
     physical: e1000g0
     defrouter not specified
zonecfg:keetonga> verify
zonecfg:keetonga> commit
zonecfg:keetonga> exit
```

zoneadm list displays the ipkg brand.

```
root@cantaloup:~# zoneadm list -cv
ID NAME        STATUS     PATH                    BRAND    IP
0  global      running    /                       native   shared
-  keetonga    configured /zones/keetonga         ipkg     shared
```

A.2. Cookbook: Installing an ipkg zone

```
root@cantaloup:~# zoneadm -z keetonga install
A ZFS file system has been created for this zone.
   Publisher: Using opensolaris.org (http://pkg.opensolaris.org/release/).
       Image: Preparing at /zones/keetonga/root.
       Cache: Using /var/pkg/download.
Sanity Check: Looking for 'entire' incorporation.
  Installing: Core System (output follows)
DOWNLOAD                        PKGS       FILES     XFER (MB)
Completed                       20/20      3021/3021   42.55/42.55

PHASE                           ACTIONS
Install Phase                   5747/5747
  Installing: Additional Packages (output follows)
DOWNLOAD                        PKGS       FILES     XFER (MB)
Completed                       37/37      5598/5598   32.52/32.52

PHASE                           ACTIONS
Install Phase                   7332/7332

        Note: Man pages can be obtained by installing SUNWman
 Postinstall: Copying SMF seed repository ... done.
 Postinstall: Applying workarounds.
        Done: Installation completed in 648,952 seconds.

  Next Steps: Boot the zone, then log into the zone console
              (zlogin -C) to complete the configuration process
root@cantaloup:~#
```

B. References

[1] Jeff Victor, "Solaris Containers Technology Architecture Guide", Sun Blueprint, May 2006, http://www.sun.com/blueprints/0506/819-6186.html

[2] Jeff Victor, "Zones and Containers FAQ", OpenSolaris FAQ, http://opensolaris.org/os/community/zones/faq/

[3] Sun Microsystems Inc., "Solaris Containers Learning Centre", http://www.sun.com/software/solaris/containers_learning_center.jsp

[4] Joost Pronk van Hoogeveen,"Working with Solaris Containers and the Solaris Service Manager", Sun Blueprint, May 2006, http://www.sun.com/blueprints/0506/819-4328.html

[5] Menno Lageman, "Solaris Containers --What They Are and How to Use Them", Sun Blueprint, May 2005, http://www.sun.com/blueprints/0505/819-2679.pdf

[6] Sun Microsystems Inc., "System Administration Guide: Solaris Containers-Resource Management and Solaris Zones", Solaris 10 Manual, 2006, http://docs.sun.com/app/docs/doc/817-1592

[7] OpenSolaris Project: "Crossbow: Network Virtualization and Resource Control" http://opensolaris.org/os/project/crossbow/

[8] Websphere Application Server in Container http://www.sun.com/software/whitepapers/solaris10/websphere6_sol10.pdf and http://blogs.sun.com/roller/page/sunabl?entry=websphere_deployment_on_solaris_10

[9] Statement by IBM that Websphere MQ is only supported in global zones and whole root local zones http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg21233258

[10] Solaris Containers at Apache Software Foundation http://www.tbray.org/ongoing/When/200x/2006/03/06/Apache-Server

[11] Oracle in Container http://www.sun.com/blueprints/0505/819-2679.pdf pp. 22-33 and Oracle Metalink https://metalink.oracle.com (Note: 317257.1)

[12] Sun Cluster Data Service for Solaris Containers Guide http://docs.sun.com/app/docs/doc/819-2664

[13] Webserver/ftpd and Backup/Restore in a zone http://www.sun.com/blueprints/0506/819-6186.pdf pg. 8 and pp. 10-13

[14] Qualification Best Practices for Application Support in Non-Global Zones http://developers.sun.com/solaris/articles/zone_app_qualif.html

[15] "Bringing Your Application Into the Zone". http://developers.sun.com/solaris/articles/application_in_zone.html

[16] Sreekanth Setty, "Deploying Sun Java Enterprise System on the Sun Fire T2000 Server using Solaris Containers", Sun Blueprint, August 2006, http://www.sun.com/blueprints/0806/819-7663.pdf

[17] http://en.wikipedia.org/wiki/Virtualization

[18] Zone-specific settings in later Solaris updates: http://www.opensolaris.org/os/community/arc/caselog/2006/496/spec-txt

[19] Graphical monitoring of zone load http://www.asyd.net/home/projects/zonestats

[20] Jeff Savit, "Energy Efficiency Strategies: Sun Server Virtualization Technology", Sun Blueprint, August 2007, http://www.sun.com/blueprints/0807/820-3023.html

[21] Harry J. Foxwell, Menno Lageman, Joost Pronk van Hoogeveen, Isaac Rozenfeld, Sreekanth Setty and Jeff Victor, "The Sun BluePrints Guide to Solaris Containers: Virtualization in the Solaris Operating System", Sun Blueprint, October 2006, http://www.sun.com/blueprints/1006/820-0001.html

[22] Sybase: Best Practices for Running ASE 15.0 in Solaris Containers http://www.sybase.com/detail?id=1041285 http://wikis.sun.com/display/SolarisContainers/Sybase

[23] Solaris Container Cluster in Sun Cluster Installation Guide http://docs.sun.com/app/docs/doc/820-4677

[24] System Administration Guide: Virtualization Using the Solaris Operating System http://docs.sun.com/app/docs/doc/819-2450?l=en&q=Virtualizati

[25] Solaris Container Leitfaden - http://mediacast.sun.com/users/Detlef..Drewanz/media/solaris-container-leitfaden.pdf/details