Solaris 11.2 -
Erfahrungen aus der Praxis

Thomas Nau
Universität Ulm – kiz
Ulm

Schlüsselworte:
Solaris 11

Einleitung:
Von vielen tot-geglaubt oder gar tot-geschwiegen ist Solaris lebendiger denn je. So sind zwar seit der Übernahme der Entwicklungsverantwortung durch Oracle die Quellen – oft auch zum Bedauern des Autors – nicht mehr frei zugänglich, dies bedeutet jedoch nicht, dass die Weiterentwicklung des Betriebssystems an Dynamik eingebüßt hat. So arbeiten heute hinter den Kulissen vermutlich mehr Entwickler denn je an neuen Features oder an Verbesserungen und Anpassungen bereits vorhandener.

Neben den großen Neuerungen wie etwa der Integration von *Puppet* und *OpenStack* in Solaris 11.2, sind auch eine Vielzahl anderer, weniger bekannter, Features hinzugekommen die für die hauseigene IT oft erhebliche Vorteile bieten oder ihr gar vollkommen neue Möglichkeiten eröffnen.
Besonders Entwicklungen im und um den Bereich der Virtualisierung, etwa die Weiterentwicklung des Netzwerkstacks und der Zonen-Technologie, machen eine große Anzahl von Neuerungen aus.
Diese sollen die Zuverlässigkeit von Ethernet-basierten Netzwerken hinsichtlich einer gemeinsamen Nutzung durch Anwendungen mit unterschiedlichsten Anforderungen weiter verbessern aber auch eine weitere Konvergenz mit Speicher- und anderen Netzwerken ermöglichen. Andere Neuerungen sind der Notwendigkeit geschuldet, Cloud basierte Infrastrukturen oder cross-site Anwendungen überhaupt erst mit geringem Aufwand und größtmöglicher Sicherheit zu ermöglichen.
Ein wesentlicher Vorteil beim Einsatz von Solaris ergibt sich aus der Tatsache, dass Zonen – die seit langem in Solaris integrierte Virtualisierungstechnik – den Kernel des Wirts-Systems nutzen und daher mit diesem noch enger verflochten sind als para-virtualisierte Treiber dies vermögen. Kernelzonen helfen ab Solaris 11.2 in denjenigen Anwendungsszenarien, in denen ein gemeinsamer Kernel – etwa auf Grund des identischem Patchlevels – nicht ausreichend ist. Auch grundlegende Verbesserungen der Sicherheit sind im Zonen Umfeld zu finden.

Das Solaris Team des Kommunikations- und Informationszentrums (kiz) der Universität Ulm hatte im Rahmen des *Platinum Beta* Programms von Oracle nicht nur die Möglichkeit, die Entwicklung all dieser Features zu verfolgen, sondern konnte diese auch Monate vor der eigentlichen Freigabe testen und entsprechendes Feedback geben. Der folgende Artikel greift einige der neuen Features im Detail auf, die für die Produktivumgebung des kiz von besonderer Bedeutung sind.
Eine vollständige Übersicht über alle neuen Features stellt Oracle zur Verfügung.
`http://www.oracle.com/technetwork/server-storage/solaris11/documentation/solaris11-2-whatsnew-2191087.pdf`

Hintergrund des Autors:
Der Autor ist Leiter der Abteilung Infrastruktur des Kommunikations- und Informationszentrums (kiz) der Universität Ulm und gleichzeitig dessen stellvertretender Leiter. Das kiz trägt unter anderem die

Gesamtverantwortung für die universitäre IT-Infrastruktur, inklusive Telefonie, sowie die Versorgung der Wissenschaftler und Studenten sowohl mit elektronischen als auch mit Print-Medien. Die Kernaufgaben der Abteilung Infrastruktur umfassen hierbei insbesondere Planung, Weiterentwicklung und den Betrieb der Netzwerke, sowie aller zentralen Server. Zu diesen zählen neben Backup- und HPC-Systemen insbesondere auch die "virtuellen Welten" und die auf HA-Clustern basierenden Mail-, LDAP-, Portal-, Datenbank- und File-Server der Universität Ulm. Grundlage dieser kritischen IT-Dienste ist in den allermeisten Fällen Solaris 11.2 sowohl in der x86 als auch der SPARC Inkarnation.
Darüber hinaus ist die Abteilung sehr stark in Projekte[1] des Landes Baden-Württemberg eingebunden und erbringt in diesem Zusammenhang auch Dienstleistungen für weitere Hochschulen des Landes basierend auf dem leistungsfähigen Landeshochschulnetz BelWü[2].

Netzwerk:
Link Aggregation, Trunking und IP-Multipathing (*IPMP*), wie in Abbildung 1 dargestellt, bilden seit längerem die Basis für Ausfallsicherheit und/oder load-balancing im Solaris Netzwerk Stack.

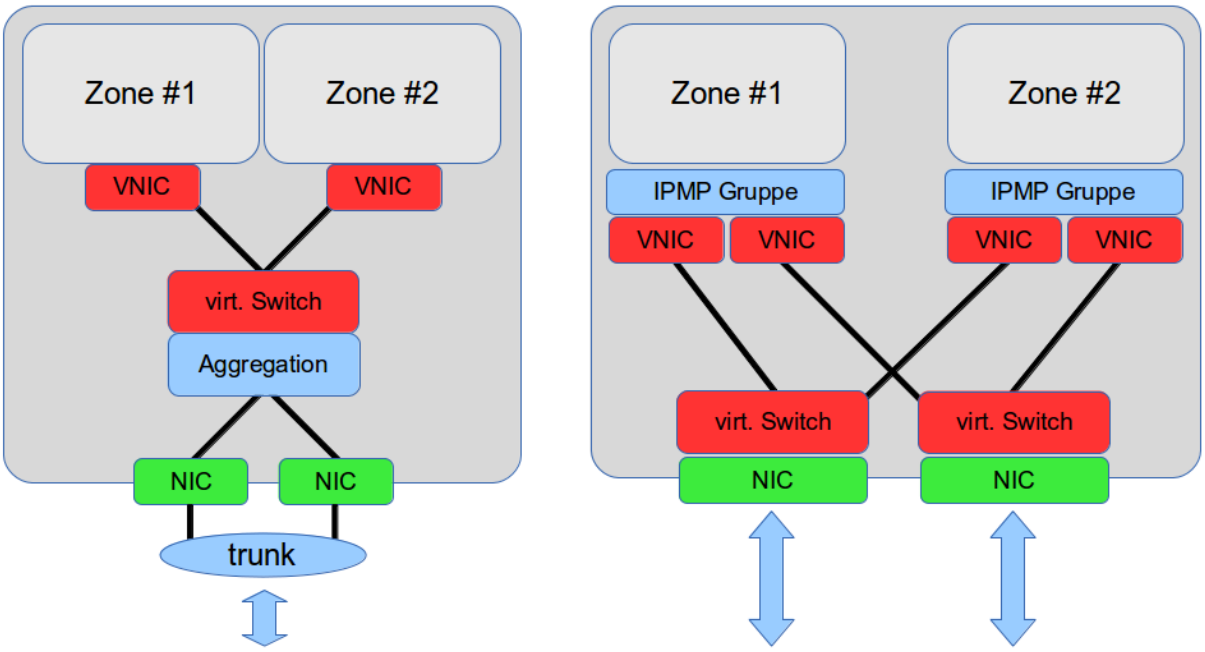

Abbildung 1: Link Aggregation Trunk versus IP-Multipathing (IPMP)

Beide Ansätze unterschieden sich jedoch gravierend in den individuellen Vor- und Nachteilen.

Technik	Pro	Contra
Link Aggregation Trunking	transparent für Zonen und virtuelle Maschinen; einfache Administration	erfordert spezielle Konfiguration der Switches

1 bwCloud: Standortübergreifende Servervirtualisierung
bw100G: Forschung und innovative Dienste für ein flexibles 100G-Netz in Baden- Württemberg
bwHPC, bwHPC-C5: http://www.bwhpc-c5.de
2 http://www.belwue.de

	erhöht die verfügbare Bandbreite	Beschränkt die Konnektivität auf einen einzelnen Switch oder verlangt proprietäre Protokolle
	automatisches failover/fallback	alle verwendeten Interfaces müssen identische Duplex Modi und Geschwindigkeiten haben
IP Multipathing (IPMP)	failover über mehrere Switches hinweg ohne Nutzung proprietärer Protokolle Switch Konfiguration nicht notwendig	erfordert individuelle Konfiguration für jede Zone oder virtuelle Maschine

Data-link Multipathing (*DLMP*, Abbildung 2) stellt seit Solaris 11.1 eine Technik bereit, die für die allermeisten Anwendungsszenarien das Beste aus beiden Welten vereinigt. Hochverfügbarkeit über mehrere Switches hinweg, ohne auf herstellerspezifische und proprietäre Protokolle zurückgreifen zu müssen, ist mit *DLMP* einfach realisierbar. Die Transparenz aus Anwendungssicht hinsichtlich der VNICs bleibt dabei vollständig erhalten. Lediglich load-balancing ist nur sehr eingeschränkt möglich, indem die zugeordneten VNICs „manuell“ auf die physikalischen Interfaces verteilt werden.

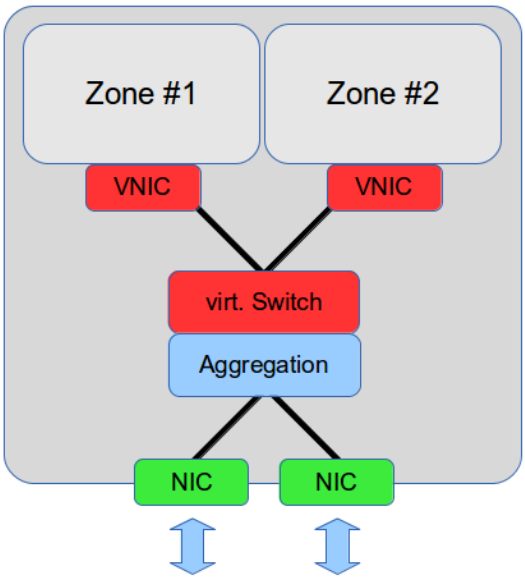

Abbildung 2: Data-link Multipathing (DLMP)

Um failover Entscheidungen treffen zu können war *DLMP* bisher von den Link Status Informationen der Interface-Karten abhängig. Eine aktive Überprüfung, wie sie bei *IPMP* stattfindet, war nicht möglich was zur Folge hat, dass gewisse Fehlersituationen auf dieser Ebene nicht erkennbar sind. Mit Solaris 11.2 hat sich dies durch die Einführung des Dienstes *svc:/network/dlmp:default* geändert. Das Setzen der link property *probe-ip* für das konfigurierte DLMP Interface erlaubt es dem Dämon *in.dlmpd(1m),* aktiv die Netzwerkkonnektivität in das lokale Subnetz mit ICMP Paketen zu überprüfen.

Aggregation anlegen und IP Adressen zuweisen

```
obi-wan# dladm create-aggr -m dlmp -l net2 -l net3 aggr0
obi-wan# ipadm create-ip aggr0
obi-wan # ipadm create-addr -T static -a 192.168.2.21/24 aggr0/v4

obi-wan # dladm show-aggr -x
LINK     PORT       SPEED DUPLEX   STATE     ADDRESS            PORTSTATE
aggr0    --         1000Mb full    up        0:10:e0:24:2a:bf   --
         net2       1000Mb full    up        0:10:e0:24:2a:be   attached
         net3       1000Mb full    up        0:10:e0:24:2a:bf   attached

obi-wan# dladm show-aggr -S
LINK       PORT           FLAGS STATE   TARGETS         XTARGETS
aggr0      net2           u---  unknown --              --
--         net3           u---  unknown --              --
```

Die flags-Spalte des letzten Kommandos spiegelt den Status der konfigurierten probing Mechanismen wider.

Position	State	Flag
#1	link state	link up (u), down (d) oder unknown (-)
#2	prober state	ICMP prober (p)
#3	L2 state	L2 active (2)
#4	L3 state	ICMP active (3)

Nach der Konfiguration der entsprechenden link-property sowie einer IP-Adresse als Ziel der ICMP Pakete ändert sich das Bild:

```
obi-wan# dladm set-linkprop -p probe-ip=+192.168.2.1 aggr0

obi-wan# dladm show-aggr -S
LINK       PORT           FLAGS STATE   TARGETS         XTARGETS
aggr0      net2           u-2-  active  --              net3
--         net3           u--3  active  hydra-storage   net2

obi-wan#  dlstat show-aggr -n -P all
 TIME   AGGR     PORT          LOCAL         TARGET  PROBE  NETRTT     RTT
0.06s  aggr0     net2           net2           net3    t24      --      --
0.06s  aggr0     net2           net2           net3    t24  1.96ms  2.08ms
0.32s  aggr0     net3   192.168.2.21    192.168.2.1    i24      --      --
0.32s  aggr0     net3   192.168.2.21    192.168.2.1    i24  0.12ms  0.86ms
0.84s  aggr0     net3           net3           net2    t25      --      --
0.84s  aggr0     net3           net3           net2    t25  1.86ms  1.95ms
1.20s  aggr0     net3   192.168.2.21    192.168.2.1    i25      --      --
1.20s  aggr0     net3   192.168.2.21    192.168.2.1    i25  0.13ms  1.10ms
...
```

Ein Reset der link-property schaltet das aktive Testen der Konnektivität wieder ab.

Das kiz hat in der Zwischenzeit alle ehemals auf *IPMP* basierenden Lösungen auf *DLMP* umgestellt, da sich hier der Administrationsaufwand und die Fehleranalyse oft sehr viel einfacher gestalten. Auch

iSCSI basiertes Multipathing wurde zu Gunsten der dargestellten DLMP basierten Lösung aufgegeben und an vielen Stellen gleichzeitig der Wechsel hin zu einer 10GE Infrastruktur vollzogen.

Viele Solaris 11.2 Verbesserungen im Bereich des Netzwerkstacks betreffen hauptsächlich den Bereich der Virtualisierung, so auch die neue Unterstützung für „reflective relays“ im Rahmen des Edge Virtual Bridging (EVB). Im Allgemeinen verlassen Netzwerkpakete von Gästen, die unterschiedliche VNICs aber das selbe physikalische Interface verwenden nie das Hostsystem sondern werden über einen virtuellen Switch geführt. In manchen Fällen kann es jedoch sinnvoll sein, die Kommunikation über einen externen Switch abzuwickeln, etwa wenn dort die ACLs hinterlegt sind. Sofern dieser Switch EVB und reflective relaying unterstützt kann Solaris 11.2 entsprechend konfiguriert werden und die Pakete entsprechend „routen“. Der Switch benötigt die besonderen Features, da ansonsten Pakete nicht auf dem Interface das Gerät verlassen können auf dem sie empfangen wurden.

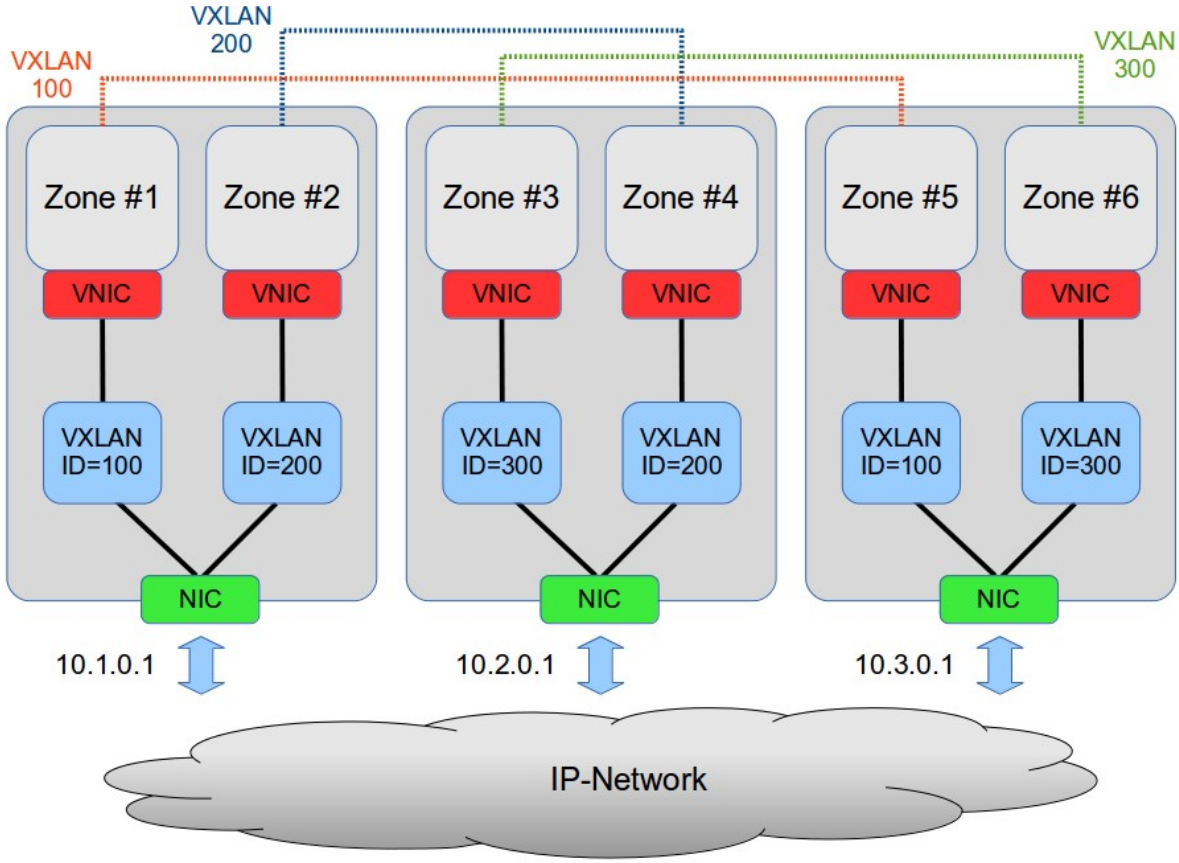

Abbildung 3: VXLAN Infrastruktur

Auch Virtual Extensible Local Area Networks (VXLANs) kommen meist in Cloud Umgebungen zum Einsatz wenn physikalische Server in getrennten Layer-2 Netzen stehen, etwa geographisch getrennt oder in unterschiedlichen Gebäuden auf dem selben Campus. Erschwerend kommt hinzu, dass oft IP-Adressbereiche und damit auch ACLs an physikalische Netzwerke gekoppelt sind. Dadurch wird eine Migration von Diensten und virtuellen Maschinen zwischen Standorten schwierig oder gar unmöglich. VXLANs lösen dieses Problem transparent für die jeweiligen Gastsysteme und Anwendungen, indem Layer-2 Pakete über eine bestehende Layer-3 Infrastruktur transportiert werden wobei die 24-bit lange VXLAN ID bis zu 16 Millionen Segmente abdeckt. Abbildung 3 verdeutlicht dies.

VXLANs in Solaris 11.2 können mit IPsec kombiniert werden. Darüber hinaus sind gängige Tools wie *wireshark(1), tshark(1)* oder *snoop(1m)* in der Lage die VXLAN Pakete, sofern nicht verschlüsselt, zu dekodieren.

Verbesserungen in wie *dladm(1m), tcpstat(1m), ipstat(1m)* oder *dlstat(1m)* erleichtern Administratoren die tägliche Arbeit. Details finden sich in der jeweilige Dokumentation und den Release Notes.

```
alderaan# tcpstat -c -l 5 10 1
Please wait...
ZONE     PID PROTO  SADDR              SPORT DADDR              DPORT   BYTES
global   965 TCP    mosel.rz.uni-ulm     666 home.rz.uni-ulm.   2049    3.5M
global   965 TCP    buche.rz.uni-ulm     874 home.rz.uni-ulm.   2049  190.0K
global   965 TCP    hannover.rz.uni-     865 home.rz.uni-ulm.   2049   19.6K
global   965 TCP    home.rz.uni-ulm.    2049 mosel.rz.uni-ulm    666   16.2K
global   965 TCP    hof.rz.uni-ulm.d     799 home.rz.uni-ulm.   2049   14.4K
Total: bytes in:  3.7M bytes out: 41.8K
```

Service Management:
Auch 10 Jahre nach Einführung der *Service Management Facility* (SMF) in Solaris 10 setzen auch heute noch viele Administratoren init-Skripte für eigene Anwendungen ein. Dafür gibt es jedoch nach den in 11.1 eingeführten und in 11.2 fortgeschriebenen Erleichterungen im Umgang mit SMF nur noch wenig Rechtfertigung. Für die einfache Aufgabe, einen eigenen Dämon beim booten des Systems zu starten reicht seit 11.1 ein einziges Kommando um das notwendige XML Manifest File zu erzeugen.

```
alderaan# svcbundle -o mydaemon.xml \
            -s service-name=mysite/mydaemon \
            -s model=daemon \
            -s start-method="/usr/local/bin/mydaemon"
```

Nach ggf. notwendigen Anpassungen etwa hinsichtlich der Service Abhängigkeiten genügen zwei weitere Kommandos um den Dienst zu aktivieren.

```
alderaan# cp mydaemon.xml /lib/svc/manifest/site
alderaan# svcadm restart manifest-import
```

Ein wenig genutztes Feature, das Erzeugen von Service-Instanzen, stellte sich in unserem Umfeld für einige Dienste als echte Erleichterung dar. So definieren wir etwa für jeden *zpool* eigene Bacula File-Daemon Backup Instanzen, die mit dem Export bzw. Import (de)aktiviert werden.

```
jedi# svcs */bacula*
STATE          STIME    FMRI
disabled       Sep_01   svc:/application/backup/bacula-fd:readonly
disabled       Sep_01   svc:/application/backup/bacula-fd:default
online         Sep_05   svc:/application/backup/bacula-fd:home
online         Sep_05   svc:/application/backup/bacula-fd:cifs
```

Die Instanz *bacula-fd:home* wurde mit folgenden Kommandos unter Solaris 11 erzeugt.
Anmerkung: diese unterscheidet sich geringfügig von Solaris 10.

```
INSTANCE=home
SVC="bacula-fd:${INSTANCE}"
EXEC=/opt/bacula/bin/amd64/bacula-fd
```

```
LOG="/var/svc/log/application-backup-bacula-fd:${INSTANCE}.log"
CFG="/opt/bacula/etc/bacula-fd-${INSTANCE}.conf"

svccfg -s bacula-fd add "$INSTANCE"

svccfg -s $SVC setprop restarter/logfile  = astring: "$LOG"
svccfg -s $SVC setprop conf-file/entities = fmri: "file://localhost/$CFG"
svccfg -s $SVC setprop start/exec         = astring: "\"$EXEC -c $CFG\""

svccfg -s $SVC addpg general framework
svccfg -s $SVC setprop general/complete   = astring: \"\"

svcadm refresh $SVC
svcadm enable  $SVC
```

Ein weiteres Einsatzgebiet im kiz sind die Datenbankserver, die eine Vielzahl von DB-Instanzen mit teilweise unterschiedlichen Versionen bereit stellen.
Lassen sich mit SMF Anwendungen, und auch mehrere Instanzen davon, einfach managen so bleibt dennoch das Problem, dass viele Anwendungen die benötigten Konfigurationen nicht in Form vom SMF properties bereitstellen bzw. auswerten sondern auf Konfigurationsdateien zurückgreifen. Die mit 11.2 eingeführten SMF stencils schließen diese Lücke zumindest für Applikationen mit geringen Ansprüchen an die Komplexität der Konfigurationsdateien. Der neue Mechanismus erzeugt dabei die erforderlichen Dateien aus SMF properties ohne dass eine Anpassung der Anwendung notwendig wird. Ein Beispiel an Hand des Apache Webservers findet sich im Blog von Jörg Möllenkamp unter `http://www.c0t0d0s0.org/archives/7715-New-Solaris-11.2-features-SMF-stencils.html`

Zonen:
Bereits in Solaris 11 wurde das Konzept der *read-only non-global-zones* eingeführt, das oft auch unter dem Begriff *immutable zones* (IMZ) bekannt ist. Hinsichtlich der Konfiguration derartiger Zonen ist lediglich die Wahl eines entsprechenden *file-mac-profile* notwendig.

file-mac-profile	Beschränkungen
none	keine; normale Zone
strict	read-only Zone (immutable)
fixed-configuration	Verzeichnisse unterhalb von */var* sind schreibbar sofern diese keine Konfigurationsdateien enthalten
flexible-configuration	wie *fixed-configuration* aber auf */etc* ausgedehnt

Der Nachteil des bisherigen Ansatzes liegt auf der Hand. Administratoren der globalen Zone können, unabhängig von den Einstellungen für die NGZ, Dateien verändern.

Um diese potentielle Lücke zu schließen bringt 11.2 zwei Neuerungen mit sich. Dies ist zum einen das Konzept eines *trusted-path,* also eines Zugangswegs der als sicher eingestuft ist. Im Falle von nicht-globalen Zonen ist dies *zlogin(1)* mit der neuen option „*-T*“. Ein login über diesen Mechanismus erlaubt auch Änderungen in der IMZ ohne dass diese explizit in den update-Modus gebootet werden muss.

Mit dieser Voraussetzung, dem *trusted-path*, ist auch eine Erweiterung des Sicherheitskonzepts auf die globale Zone realisierbar. Der sichere Weg führt hier ausschließlich über Konsole, grafisch oder auch seriell. Casper Dik stellt in seinem Blog einige kurze Besipiele zur Verfügung.
`https://blogs.oracle.com/casper/entry/solaris_11_2_immutable_global`

Für besonders sensible Systeme kann das Konzept der *immutable-global-zone* noch mit der Funktion des *verified-boot* kombiniert werden. Hierdurch werden sowohl die Bootblöcke der Platten, als auch der Kernel und alle zugehörigen Module mit digitalen Unterschriften signiert und ein Systemstart ist, je nach Einstellung, nur nach erfolgreicher Verifikation möglich.
`http://docs.oracle.com/cd/E36784_01/html/E37121/gmwdf.html`

Unified Archives (UA):
UAs schließen nicht nur die Lücke, die das Fehlen der *flash-archive* Unterstützung bei der Installation hinterlassen hat sondern sie bieten viel weitergehende Möglichkeiten. Basierend auf ZFS streams können sie mehrere Solaris Instanzen enthalten, die unabhängig voneinander referenzierbar sind, also z.B. als direkte Basis zur Installation einer Zone genutzt werden können. Der *Automated Installer* (AI) unterstützt das Format ebenfalls. Des weiteren können *unified archives* auch so erzeugt werden, dass das direkte Booten vom Medium und damit ein disaster recovery möglich ist.

```
jedi# archiveadm create --recovery no_more_disasters.uar
```

Die Server *Smith* und *Morpheus* hosten mehrere Zonen, basierend auf iSCSI shared-storage.

```
morpheus# zoneadm list -cv
  ID NAME          STATUS      PATH                    BRAND      IP
   0 global        running     /                       solaris    shared
   1 wiki          running     /zoss/wiki              solaris    excl
   2 flare         running     /zoss/flare             solaris    excl
   6 ftp           running     /zoss/ftp               solaris    excl
  15 proxy         running     /zoss/proxy             solaris    excl
   - idp-test      configured  /zoss/idp-test          solaris    excl
   - idp           configured  /zoss/idp               solaris    excl
   - mailtest      configured  /zoss/mailtest          solaris    excl
   - sofa          configured  /zoss/sofa              solaris    excl
   - svn           configured  /zoss/svn               solaris    excl
   - unixtest      configured  /zoss/unixtest          solaris    excl
```

Um nun beispielsweise einige der Zonen zu Testzwecken auf anderen Systemen zu installieren, wird mit dem folgenden Kommando ein UA erzeugt. Hierbei werden andere Zonen bzw. auch besonders große Datasets von der Archivierung ausgeschlossen, um Platz zu sparen.

```
morpheus# archiveadm create \
              --exclude-zone=ftp --exclude-zone=proxy \
              --exclude-dataset=flare_rpool/rpool/data/ai \
              --exclude-dataset=rpool/uar \
              --skip-capacity-check \
              /rpool/uar/testing.uar
Initializing Unified Archive creation resources...
Unified Archive initialized: /rpool/uar/testing.uar
...
```

Es ist oft sinnvoll, für die Erzeugung des Archivs ein eigenes ZFS Filesystem anzulegen und dieses via Kommandozeile auszunehmen. In unserem Fall *rpool/uar.*

Ein listing des Archivs zeigt dass es neben der globalen Zone des Systems auch noch zwei weitere installierbare Zonen namens *flare* und *wiki*, hier fett dargestellt, enthält.

```
morpheus:# archiveadm info -v /rpool/uar/testing.uar
Archive Information
          Creation Time:  2014-09-28T11:31:55Z
            Source Host:  morpheus
           Architecture:  i386
       Operating System:  Oracle Solaris 11.2 X86
       Recovery Archive:  No
              Unique ID:  e0dbf57f-b579-c6c6-c64e-da8850304310
         Archive Version:  1.0

Deployable Systems
          'global'
             OS Version:  0.5.11
              OS Branch:  0.175.2.1.0.5.2
              Active BE:  s11u2-1_5_0
                  Brand:  solaris
            Size Needed:  6.5GB
              Unique ID:  f33ece58-2562-e9b4-ecd2-fafa328adb25
               AI Media:  0.175.2_ai_i386.iso
              Root-only:  No
            'flare'
             OS Version:  0.5.11
              OS Branch:  0.175.2.1.0.5.2
              Active BE:  solaris-8
                  Brand:  solaris
            Size Needed:  31.1GB
              Unique ID:  a96d3fd6-9578-605f-99ab-b3adb7b497ae
               AI Media:  0.175.2_ai_i386.iso
              Root-only:  Yes
            'wiki'
             OS Version:  0.5.11
              OS Branch:  0.175.2.1.0.5.2
              Active BE:  zone-4
                  Brand:  solaris
            Size Needed:  4.7GB
              Unique ID:  e94bb6c6-c0c4-4692-8b4e-dfd030c4a6ee
               AI Media:  0.175.2_ai_i386.iso
              Root-only:  Yes
```

Jetzt kann das Archiv auf das Zielsystem übertragen und dort beispielsweise die Zonen – im Beispiel die Zone *wisdom* – wie gewohnt installiert werden. Alle dazu notwendigen Informationen, auch die über die Zonenkonfiguration, sind im Archiv enthalten.

```
jedi# archiveadm info testing.uar
Archive Information
          Creation Time:  2014-09-28T11:31:55Z
            Source Host:  morpheus
           Architecture:  i386
       Operating System:  Oracle Solaris 11.2 X86
     Deployable Systems: global,flare,wiki

jedi# zonecfg -z wisdom create  -a /root/testing.uar -z wiki
```

Eine Schwierigkeit stellt hierbei ggf. die Tatsache dar, dass die im Archiv enthaltenen Zonen auf shared storage aufsetzen, der am Zielsystem nicht verfügbar ist.

```
jedi # zonecfg -z wisdom export
...
add rootzpool
add storage \
   iscsi://hydra-storage/luname.naa.600144F09BC5CB0000005368ADBC0005
add storage \
   iscsi://typhon-storage/luname.naa.600144F0AFFC0B0000005368ADFD0005
end
```

Dies lässt sich vor der eigentlichen Installation mit wenigen Handgriffen beheben. Anschließend kann die Zone installiert und gestartet werden. Hierbei wird automatisch ein neues ZFS Filesystem für die Zone angelegt.

```
jedi # zonecfg -z wisdom
zonecfg:wisdom> remove rootzpool
zonecfg:wisdom> commit
zonecfg:wisdom> exit

jedi# zoneadm -z wisdom install -a /root/testing.uar -z wiki
The following ZFS file system(s) have been created:
    rpool/zoss
    rpool/zoss/wiki
Progress being logged to
/var/log/zones/zoneadm.20140928T131636Z.wisdom.install
    Installing: This may take several minutes...
...
 Updating non-global zone: Zone updated.
                   Result: Attach Succeeded.

        Done: Installation completed in 282.104 seconds.
  Next Steps: Boot the zone, then log into the zone console (zlogin -C)
              to complete the configuration process.
```

Zonen existieren seit längerem bereits in sogenannten *brands,* also besonderen Ausprägungen, nämlich derzeit

- solaris: standard Solaris 11 Zone
- solaris10: Solaris 10 Zone
- labeled: trusted extensions

Mit 11.2 sind Solaris Kernel Zonen als neuer brand *solaris-kz* neu hinzugekommen. Im Gegensatz zu allen anderen brands wird hier pro Zone ein eigener Kernel gestartet der aus Sicht der Verwaltung usw. vollkommen unabhängig von dem der globalen Zone ist. Dies gilt insbesondere für das Paket-System IPS aber auch für die Sichtbarkeit der Prozesse aus der globalen Zone und die Version des Kerns. Ein wesentlicher Vorteil für den Einsatz im kiz ist die Möglichkeit, mehrere Solaris CIFS Server im Rahmen einer Konsolidierung auf einer physikalischen Hardware einsetzen zu können. Dies ist im Gegensatz zu NFS Servern mit den bisher verfügbaren Zonen Modellen nicht möglich, ist jedoch zwingend wenn mehrere *Active Directory* Domänen eingebunden werden müssen.
Durch die Anforderungen an bestimmte CPU features sind ältere Systeme ggf. nicht in der Lage als Hostsystem für Kernel Zonen zum Einsatz zu kommen. Die Hard- und Software Voraussetzungen sind hier zusammengefasst:

http://docs.oracle.com/cd/E36784_01/html/E37629/gnwoi.html

Sun Fire x4150		Sun Fire x4170-M3	
# virtinfo		# virtinfo	
NAME	CLASS	NAME	CLASS
non-global-zone	supported	non-global-zone	supported
		kernel-zone	supported

Kernel Zonen lassen sich wie ihre älteren Geschwister über Konfigurationsdateien einrichten und vor dem Starten konfigurieren. Zu beachten ist, dass einige Parameter nicht frei gewählt werden können bzw. gesetzt werden müssen. So ist nur eine exklusive Nutzung des IP Stacks möglich. Darüber hinaus sind Ressourcen wie etwa der physikalische Speicher zu setzen. Als Ausgangspunkt kann das in Solaris verfügbare Template dienen.

```
buymore# zonecfg -z NerdHerd
Use 'create' to begin configuring a new zone.
zonecfg:NerdHerd> create -t SYSsolaris-kz
zonecfg:NerdHerd> export
create -b
set autoboot=false
set autoshutdown=shutdown
set hostid=0x14aea40e
add anet
set lower-link=auto
set configure-allowed-address=true
set link-protection=mac-nospoof
set mac-address=auto
set id=0
end
add device
set storage=dev:/dev/zvol/dsk/%{global-rootzpool}/VARSHARE/zones/%
{zonename}/disk%{id}
set bootpri=0
set id=0
end
add capped-memory
set physical=2G
end
NerdHerd: keysource not exported: does not exist

zonecfg:NerdHerd> select capped-memory
zonecfg:NerdHerd:capped-memory> set physical=8G
zonecfg:NerdHerd:capped-memory> end
zonecfg:NerdHerd> add virtual-cpu
zonecfg:NerdHerd:virtual-cpu> set ncpus=2
zonecfg:NerdHerd:virtual-cpu> end
zonecfg:NerdHerd> commit
zonecfg:NerdHerd> exit
```

Im Anschluss kann die Zone wie gewohnt installiert, konfiguriert und gestartet werden.
AI Manifeste lassen sich übrigens in 11.2 mit Hilfe eines Wizzards auf dem AI Server direkt im Browser erstellen sofern diese Möglichkeit nicht durch die Administratoren deaktiviert wurde.
http://ai.yourdomain:5555

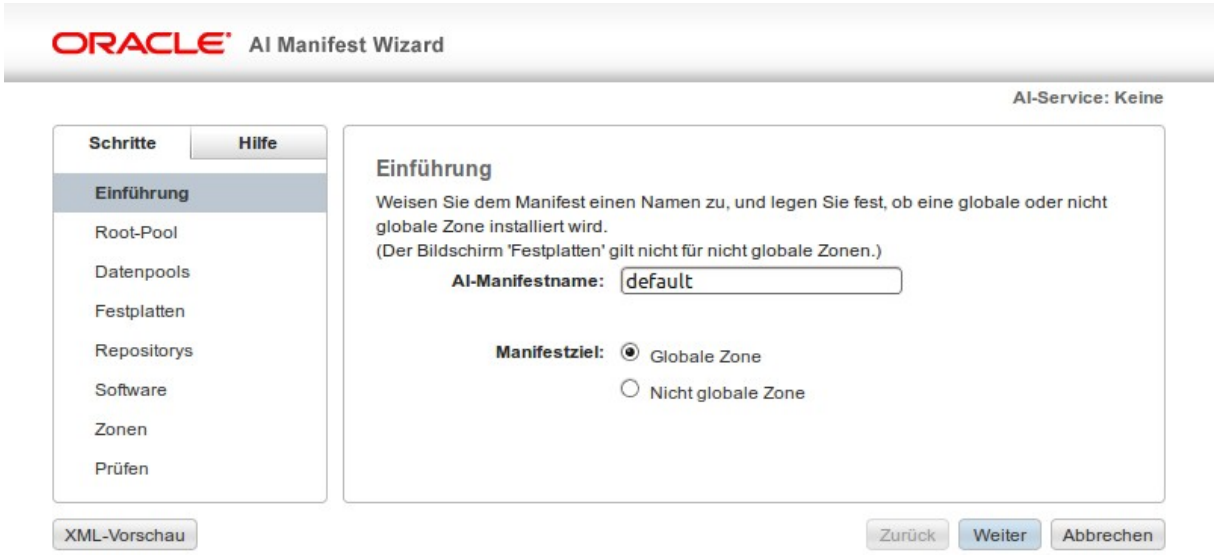

Abbildung 4: AI Wizard

Zusätzlich unterstützt der *Automated Install* Server jetzt auch Installationen über HTTPS.

Alternativ ist es möglich native Zonen (nur Solaris 11) in Kernel Zonen zu migrieren.
`http://docs.oracle.com/cd/E36784_01/html/E37629/docinfo.html`
`http://www.oracle.com/technetwork/articles/servers-storage-admin/howto-create-kernal-zones-s11-2251331.html`

Neben den Verbesserungen in Sachen Sicherheit sowie der Möglichkeit, Kernel-nahe Anwendungen auf einer physikalischen Hardware zu konsolidieren bieten Kernel Zonen auch die Möglichkeit, sie im Betrieb? zwischen Systemen zu migrieren sofern einige Hardware Voraussetzungen erfüllt sind. Zu diesen gehört ein gemeinsamer Storage-Bereich auf alle Systeme zugreifen können. Außerdem müssen Plattform und CPU Revisionen der verwendeten Systeme kompatibel sein. In Jeff Victor's Blog finden sich hierzu weitere Details.
`https://blogs.oracle.com/JeffV/entry/migratory_solaris_kernel_zones`

Eine weitere Verbesserung ist die nun vorhandene Möglichkeit, die Konfiguration laufender Zonen ohne reboot zu verändern. Dazu kennt *zonecfg(1m)* nun die Option „-r“. Zu beachten ist lediglich, dass die Änderungen nur bis zum nächsten reboot der Zone aktiv sind sofern deren Konfiguration nicht auch entsprechend angepasst wurde. Ein zusätzliches Interface zur Laufzeit ist damit kein Problem.
http://docs.oracle.com/cd/E36784_01/html/E37628/gogda.html

```
jedi# zlogin yoda
[Connected to zone 'yoda' pts/6]
yoda:.../~# dladm
LINK                CLASS     MTU    STATE    OVER
net0                vnic      1500   up       ?


jedi# zonecfg -z yoda -r \
          "add anet;set linkname=net1;set lower-link=net3;end;commit"
zone 'yoda': Checking: Adding anet linkname=net1
zone 'yoda': Applying the changes
```

```
yoda:.../~# dladm
LINK                CLASS     MTU    STATE    OVER
net0                vnic      1500   up       ?
net1                vnic      1500   down     ?
```

Im übrigen können Zonen jetzt auch auf einfache Weise mit *zoneadm(1m)* umbenannt werden.

Einfach und gut:
Oft sind es die einfachen Dinge die das Leben der Administratoren vereinfachen. Zu diesen gehören neben der herausragenden binär-Kompatibilität viele Verbesserungen unterschiedlichster Funktionen.

So stehen jetzt parallelisierte Versionen von (De)Komprimierungs-Tools stehen mit pbzip2(1) und pixy(1) bereit. Mit ihnen lässt sich die Komprimierung großer log-Dateien oder *tar*-Archive erheblich beschleunigen wie das folgende Beispiel zeigt. Ausgangspunkt ist ein 1.1GB großes *tar*-Archiv von */usr/lib*.

```
obi-wan# time bzip2 < usr_lib.tar > /dev/null
real    2m19.602s
user    2m19.184s
sys     0m0.397s

obi-wan# time pbzip2 -p8 < usr_lib.tar > /dev/null
real    0m19.102s
user    2m30.868s
sys     0m0.720s
```

Zpools lassen sich ab Solaris 11.2 mit der neuen Kommandozeilenoption „-N“ importieren ohne die Filesysteme zu mounten oder via NFS/CIFS zu exportieren. Auch das resilvering von zpools wurde vor allem für RAIDZ* pools erheblich beschleunigt.

zfs send liefert jetzt eine Schätzung über die Größe der zu übertragenden Daten die (meist) gut mit der benötigten Gesamtzeit korreliert.

```
yoda# zfs snapshot -r ai@sync-2014-09-29-10h04
yoda# zfs send -v -R -I \
        ai@sync-2014-08-30-18h03 ai@sync-2014-09-29-10h04 |
        rsh windu zfs receive -x compression -v -d -F mirror/yoda
sending @sync-2014-08-30-18h03 to ai@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/iso@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/packages@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/repo@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/repo/s11u2-ga@sync-2014-09-29-10h04
sending full stream to ai/repo/s11u2-2_8_0@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/repo/s11u2-1_5_0@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/repo/s11u2b42@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/repo/kiz@sync-2014-09-29-10h04
sending @sync-2014-08-30-18h03 to ai/target@sync-2014-09-29-10h04
estimated stream size: 5.72G
```

Vergleichbar hilfreich im täglichen Administratoren-Leben ist die Option „-u“ für *netstat(1m)* mit der sich einfach PID, Nutzer und Anwendung pro Verbindungs-Endpunkt herausfinden lassen.

Die neue Kommandozeilenoption „-L“ von svcs(1) gibt, je nach Kombination mit anderen Optionen, den Namen der zum jeweiligen SMF Service gehörenden log-Datei, die letzten 10 Zeilen oder deren gesamten Inhalt aus.

```
obi-wan# svcs -L bacula-fd:www
/var/svc/log/application-backup-bacula-fd:www.log

obi-wan# svcs -xL bacula-fd:www
svc:/application/backup/bacula-fd:www (Bacula FileDaemon)
 State: online since September 29, 2014 08:23:29 AM MEST
   See: /var/svc/log/application-backup-bacula-fd:www.log
Impact: None.
   Log:
[ Sep 18 12:59:58 Enabled. ]
[ Sep 18 12:59:58 Executing start method ("...shortend to fit..."). ]
[ Sep 18 12:59:59 Method "start" exited with status 0. ]
[ Sep 29 08:23:27 Executing start method ("...shortend to fit..."). ]
[ Sep 29 08:23:29 Method "start" exited with status 0. ]

   Use: 'svcs -Lv svc:/application/backup/bacula-fd:www' to view the
complete log.
```

Zusammenfassung:
In dem mehr als ein Jahr andauernden *Solaris Platinum Beta* Programm von Oracle haben sich die 2-4 wöchentlichen Release-Zyklen allesamt als – wie von Solaris gewohnt – sehr stabil erwiesen. Neben den „großen“ Erweiterungen *Puppet* und *OpenStack* wurden in dieser Zeit auch eine ganze Reihe weiterer Features und Verbesserungen eingeführt, die in den Bereichen Sicherheit und Virtualisierung spürbar positive Auswirkungen auf den Produktionsbetrieb des kiz hatten und weiterhin haben.

Danksagung:
Mein besonderer Dank für die fortlaufende Unterstützung mit Anregungen, Ideen und Korrekturen gilt meinem Kollegen Dr. Harald Däubler.

Kontaktadresse:
Thomas Nau
Universität Ulm – kiz
Albert Einstein Allee 11
D-89081 Ulm

Telefon:	+49 (0) 731 50-22464
Fax:	+49 (0) 731 50-12-22464
E-Mail	Thomas.Nau@uni-ulm.de
Internet:	http://www.uni-ulm.de/einrichtungen/kiz