# Live long and prosper -
# Solaris 11.4 Vorteile in der Praxis

**Thomas Nau**
**Universität Ulm – kiz**
**Ulm**

**Schlüsselworte:**
Solaris, Solaris 11.4, Praxis

**Einleitung:**
Seit der Freigabe von Solaris 11.3 und dem Start des *Solaris 11.4 public beta program* vergingen nahezu zweieinhalb Jahre. Eine ungewöhnlich lange Zeit die, auf Grund von Oracle internen Umstrukturierungen und dem damit verbundenen erzwungenen Schweigen, zu erheblicher Verunsicherung der Solaris Gemeinde geführt hat. Vom Tod des Betriebssystems war die Rede oder vom Fortleben als seelenloser Zombie ohne echte Zukunft.

Gewiss lässt sich diese Gemeinde auch mit einem kleinen gallischen Dorf vergleichen und ihr Engagement ist, zumindest der Meinung des Autors nach, ein Grund für ein Produkt das in der Version sowohl Nutzer als auch Analysten mehr als positiv überrascht hat.

So wird Sicherheit noch wichtiger aber Verbesserungen sind in sehr vielen unterschiedlichen Bereichen zu finden. Solaris 11.4 versteht sich auf effizientere ZFS send-streams oder bietet Zonen auch eine Heimat in NFS Filesystemen. Entscheidend ist auch, dass viele der neuen oder verbesserten Eigenschaften zwar von Oracle unter dem Stichwort „cloud“ vermarktet werden aber auch ganz reale Vorteile im lokalen Umfeld bieten.

Das Infrastruktur 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 viele Monate vor der eigentlichen Freigabe testen und entsprechendes Feedback geben. Der folgende Artikel greift lediglich einige der neuen Features im Detail auf. Eine vollständige Übersicht über alle neuen Features stellt Oracle zur Verfügung:

`https://docs.oracle.com/cd/E37838_01/html/E60974/index.html`

**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-, Datenbank- und File-Server der Universität Ulm. Nach wie vor ist Solaris die Grundlage von vielen dieser kritischen IT-Dienste, insbesondere derer die „storage-lastig“ sind.

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].

**ZFS und Fileservices:**
Nach wie vor ist ZFS nach Ansicht des Autors eines der Herausstellungsmerkmale von Solaris. Leider wurde viel zukünftiges Potential dadurch zerstört, dass der Zugang zu den Quellen geschlossen wurde. Die daraus resultierende, und leider hinsichtlich des on-disk-format inkompatible, Parallelentwicklung von OpenZFS oder das Neuerfinden des Rades in Form von BTRFS machen es Administratoren in gemischten Solaris, FreeBSD und Linux-Umgebungen nicht einfacher.

Sicherlich auch bedingt durch den Abgesang auf Solaris hat die OpenZFS Entwicklung in den vergangenen zwei Jahren eine erhebliche Dynamik entfaltet. Dennoch ist die vollständige Integration in die Kernel also auch in Distributionen noch immer mit Schwierigkeiten technischer und juristischer Natur verbunden.

Doch nun zu den technischen Highlights des 11.4 Releases. Verwendet 11.3 die ZFS-Pool Version 37 ist die Zählung bei 11.4 auf 44 angestiegen.

| | |
|---|---|
| 38 | Xcopy with encryption |
| 39 | Resilver restart enhancements |
| 40 | New deduplication support |
| 41 | Asynchronous dataset destroy |
| 42 | Reguid: ability to change the pool guid |
| 43 | RAID-Z improvements and cloud device support |
| 44 | Device removal |

Leider schweigt sich die Dokumentation oft zu einzelnen Punkten im Detail aus, jedoch sind einige Entwicklungen im täglichen Betrieb äußerst hilfreich.

**Asynchronous dataset destroy** beschleunigt das Löschen von ZFS datasets aus Sicht der Anwendung erheblich da die Operationen asynchron vom Kernel übernommen werden. In der Vergangenheit war das Löschen von ZFS Filesystemen, vor allem wenn diese viele snapshots enthielten, oft mit langen Wartezeiten verbunden da die Operationen im Bezug auf das `zfs destroy` Kommando synchron ausgeführt wurden. Diese starre Kopplung wurde in 11.4 aufgelöst wie das folgende Beispiel verdeutlicht. Der shell-prompt erscheint sofort, auch wenn im Hintergrund – asynchron im Kernel – noch einiges zu tun ist. Der Fortschritt ist einfach mittels `zpool monitor -t destroy` überwachbar. Dies sei am Beispiel eines dataset mit 200G und 30 snapshots verdeutlicht.

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

```
# zfs destroy -r smb/cifs/gruppen
# zpool monitor -t destroy 1

POOL         PROVIDER  TOTAL SPEED TIMELEFT
smb          destroy   199G 0      unknown
smb          destroy   198G 838M   4m02s
smb          destroy   197G 1.17G  2m48s
smb          destroy   197G 797M   4m12s
smb          destroy   197G 598M   5m37s
smb          destroy   197G 478M   7m01s
smb          destroy   197G 399M   8m25s
smb          destroy   170G 858M   35s
smb          destroy   170G 32.6M  42s
smb          destroy   170G 3.36G  50s
…
smb          destroy   6.75G 582M  11s
smb          destroy   6.75G 485M  14s
smb          destroy   3.54G 953M  3s
smb          destroy   3.54G 794M  4s
smb          destroy   3.54G 662M  5s
smb          destroy   3.54G 551M  6s
smb          destroy   3.54G 459M  7s
```

**Reguid: ability to change the pool guid** wird benötigt um zpools eine neue GUID zuweisen zu können. Notwendig wird dies immer wenn ein Pool bzw. die zugehörigen Platten, etwa im Bereich der Virtualisierung, geklont werden, ein Standardvorgehen wenn aus einem template-System neue instanziiert werden. Haben Pools die selbe GUID so lassen sie sich auf Grund dieses Konflikts nicht auf ein und dem selben System mounten. Diese Möglichkeit steht in OpenZFS bereits seit längerem zur Verfügung.

**Device removal** ist seit Jahren auf der Wunschliste vieler Systemadministratoren und wurde über die Jahre hinweg in kleinen Schritten implementiert. So ist sowohl das Entfernen von Platten aus ZFS-Spiegeln als auch das von spare-, cache-, log- oder meta-devices schon seit längerem möglich. Neu in 11.4 ist die Möglichkeit auch toplevel-vdevs aus dem Verbund zu lösen. Zwar scheiden sich lange die Geister ob es im tägliche Betrieb wirklich Erleichterungen bringt, doch war es in unserem Umfeld bereits in zwei Szenarien hilfreich.

Häufig wird bei ZFS zu einem späteren Zeitpunkt ein device als Spiegel zu einem existierenden device hinzugefügt. Problematisch wird es, wenn zpool add mit zpool attach verwechselt wird, denn damit wird nur der Pool vergrößert und keine Redundanz geschaffen. Die einzige Abhilfe war bisher den Pool neu aufzusetzen und die Daten aus dem Backup zurück zu spielen. Mit 11.4 lässt sich jetzt jedes vdev entfernen sofern die zurückbleibenden genug Kapazität bereit stellen um die zusätzliche Daten aufzunehmen.

```
# mkfile -v 8g /tank/vdev1 /tank/vdev2
/tank/vdev1 8589934592 bytes
/tank/vdev2 8589934592 bytes

# zpool create test /tank/vdev1

# zpool status test
  pool: test
 state: ONLINE
  scan: none requested
```

```
config:

               NAME           STATE      READ WRITE CKSUM
               test           ONLINE        0     0     0
                 /tank/vdev1  ONLINE        0     0     0

errors: No known data errors

# dd if=/dev/urandom of=/test/BIG bs=1024k count=40000

# zpool list test
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
test  7.94G  4.96G  2.97G  62%  1.00x  ONLINE  -

# zpool add test /tank/vdev2

# zpool status test
  pool: test
 state: ONLINE
  scan: none requested
config:

               NAME           STATE      READ WRITE CKSUM
               test           ONLINE        0     0     0
                 /tank/vdev1  ONLINE        0     0     0
                 /tank/vdev2  ONLINE        0     0     0

errors: No known data errors

# zpool list test
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
test  15.9G  4.96G  10.9G  31%  1.00x  ONLINE  -

# zpool remove test /tank/vdev2    # <-- this is where the magic happens

# zpool attach test /tank/vdev1 /tank/vdev2

# zpool list test
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
test  7.94G  4.96G  2.97G  62%  1.00x  ONLINE  -

# zpool status test
  pool: test
 state: ONLINE
  scan: resilvered 4.96G in 11s with 0 errors on Sat Sep 22 15:59:17 2018

config:

               NAME                     STATE      READ WRITE CKSUM
               test                     ONLINE        0     0     0
                 mirror-0               ONLINE        0     0     0
                   /tank/vdev1          ONLINE        0     0     0
                   /tank/vdev2          ONLINE        0     0     0

errors: No known data errors
```

Das zweite Szenario ist dem ersten vergleichbar, nämlich die Umstellung großer Pools mit den bei uns in der Vergangenheit oft eingesetzte 3-fach Spiegeln auf eine 4+2 RAIDZ2 Konfiguration. Erstere basieren bei uns häufig auf SAS-JBODs mit 3x24 Platten.

Für die Umstellung, „inline“ sozusagen, existieren hinsichtlich der Platten zwei Szenarien. Zum einen könnte von jedem *mirror-vdev* je eine Platte detached und aus diesen ein neuer RAIDZ2 Pool gebaut werden. In diesem Szenario gibt es keine Einschränkungen hinsichtlich des Platzes jedoch geht die Redundanz von n+2 auf n+1 zurück. Die andere Lösung ist das Entfernen von *top-level-vdevs*, sofern ausreichend Platzreserven vorhanden sind, um aus den dann freien Platten den neuen Pool zu erzeugen. Dieses Szenario benötigt auch erheblich mehr Zeit, je nach Menge der Daten.

**Hinweis:** es können auch mehrere *top-level-vdevs* mit nur einem Kommando entfernt werden.

Neben diesen neuen Möglichkeiten wurde ZFS auch an anderer Stelle verbessert, jedoch ohne dass dies zwingend Niederschlag in der Versionsnummer des Pools gefunden hat.

**ZFS Scheduled Scrub** fiel dem Autor als neues feature das erste Mal auf als er sich über die lange anhaltende Plattenaktivität eines Fileserver wunderte. Zpools werden mit 11.4 als default-Einstellung einmal pro Monat mittel *scrubbing* auf ihre Datenintegrität geprüft. Konfigurierbar ist das Zeitintervall über die *pool-property* namens *scrubinterval*. Auf *manual* gesetzt wird der Automatismus außer Kraft gesetzt. Die property *lastscrub* enthält den Startzeitpunkt des zuletzt ausgeführten *scrubbings*. Weitere Details finden sich auf:

`https://docs.oracle.com/cd/E37838_01/html/E61017/gbbym.html#SVZFSgsexm`

**ZFS Raw Send Streams** erlauben den effizienteren Umgang mit komprimierten Daten bei der Übertragung mit `zfs send/receive`. Diese wurden in der Vergangenheit auf dem Quellsystem dekomprimiert und auf dem Zielsystem erneut komprimiert. Mittels `zfs send -w compress` lässt sich dies vermeiden und so ggf. limitierte Netzwerkbandbreite besser ausnutzen.

Verbindungsabbrüche bei der Übertragung von ZFS hatten in der Vergangenheit zwingend einen Neustart der Übertragung für das fehlgeschlagene dataset zur Folge. Dabei war es unerheblich ob die Unterbrechung bereits zu Beginn oder erst kurz vor Ende der Übertragung auftrat.

Dieses Problem wird durch **Resumable ZFS Send Streams** adressiert. Dazu fügt Solaris 11.4 zusätzliche Informationen in den send-stream ein. Durch die Verwendung der Option `-s nocheck` lässt sich dieses neue default Verhalten unterdrücken. Das nachfolgende Beispiel verdeutlicht die Möglichkeiten. Auf dem Testsystem wird ein snapshot mittels send/receive repliziert, die Prozesse jedoch nach 15 Sekunden unterbrochen.

```
# zfs list -o space pool/src
NAME        AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
pool/src    12.7T  14.5G     65.4M   14.5G              0          0

# ( zfs send -R pool/src@sync | zfs receive -d pool/dst ) & \
    sleep 15 ; kill -9 %1

# zfs list -I resumable
NAME              USED  AVAIL  REFER  TYPE        STATE
pool/dst/src     5.77G  12.7T  5.77G  filesystem  resumable
```

Die neu hinzugekommene Option `-C`, sowohl für `zfs send` als auch `zfs receive`, erlauben es, den letzten erfolgreich übertragenen Block zu finden und von dort im stream fortzufahren. Der Lesbarkeit halber wurde die Ausgabe gekürzt:

```
# zfs receive -C pool/dst/src | \
    zfs send -v -C -R pool/src@sync | \
    zfs receive -v -d pool/dst

  found bookmark 1:5b2e:5b200:13d4f8400… for pool/src@snap-17h01
  send bookmark 5b2e:5b200 in estimating full stream … (size = 10.7G
…
estimated total stream size: 10.7G
    preserving fresh bookmark dataset pool/dst/src
  receive bookmark: 5b2e:5b200 : '' for 'smb/dst/src' for …
resuming receiving full stream of smb/src@snap-17h01 into …
…
```

Für die nachfolgend genannten Erweiterungen sei hinsichtlich Details auf die Dokumentation von Solaris 11.4 verwiesen:

`https://docs.oracle.com/cd/E37838_01/html/E61017/index.html`

- `cp -z` kopiert sehr effizient große Dateien, sofern sich Quelle und Ziel im gleichen ZFS-dataset befinden, indem es den ZFS copy-on-write (COW) Mechanismus auf die Blöcke der Datei anwendet. Siehe auch *reflink(3C)*.
- Die ZFS-properties *writelimit* und *readlimit* helfen die Bandbreiten die dem gesamten Pool zur Verfügung stehen gerechter auf einzelne datasets und ihre in der Regel virtuellen Nutzer zu verteilen. Ein ähnlicher Steuerungsmechanismus für IOPS ist nicht implementiert.
- Durch Setzen der ZFS property `aclinherit=passthrough-mode-preserve` lässt sich die Interoperabilität hinsichtlich der Vererbung von Zugriffsrechten in Umgebungen verbessern, bei denen auf Daten via SMB und NFS zugegriffen wird.
- Der Support der SMB Version 3.1.1 kann, geeignete clients vorausgesetzt, den Durchsatz deutlich verbessern da parallel mehrere Kanäle verwendet werden können. Außerdem ist es möglich den Datenverkehr zu verschlüsseln.

**Analytics und Web Dashboard:**
Der neue so genannte *StatsStore* von Solaris 11.4 sammelt und konsolidiert neben Prozess und Kernel Statistiken auch Audit-Events usw. und lässt sich durch eigene Anwendungen erweitern. Dabei greift er auf bereits lange verfügbare Mechanismen, etwa das *kstat-Interface* und *DTrace*, zurück.
Neben einem Kommandozeilen-Interface ist insbesondere ein Web basiertes Dashboard Teil von 11.4. Dieses stellt wichtige Parameter wie RAM, Systemlast, Netzwerkauslastung, … übersichtlich dar und kann dabei sowohl auf aktuelle als auch auf historische Daten zurückgreifen. Die Darstellung sowie die Art und Zahl der Daten, die gesammelt werden sollen ist konfigurierbar.

Eine Beschreibung aller Möglichkeiten würde jedoch den Rahmen dieses Artikels sprengen, daher sei neben den folgenden Screenshots auf die Dokumentation verwiesen.

`https://docs.oracle.com/cd/E37838_01/html/E56520/index.html`

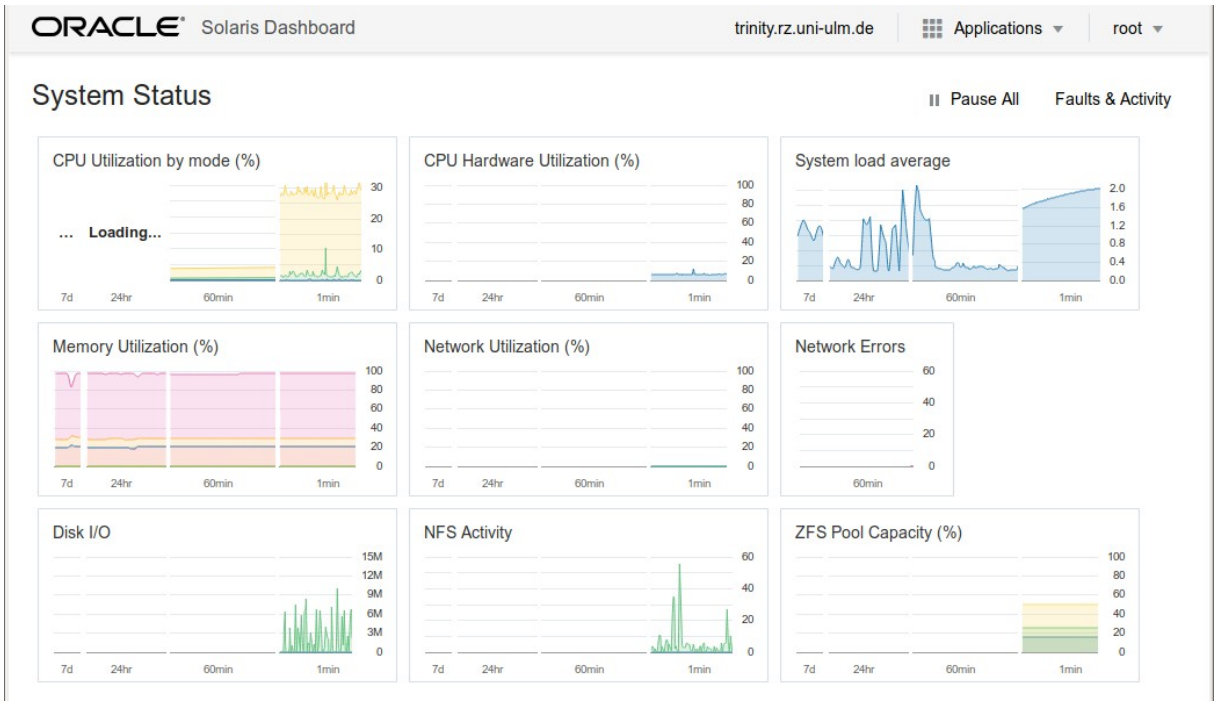


*Abbildung 1: Übersicht*

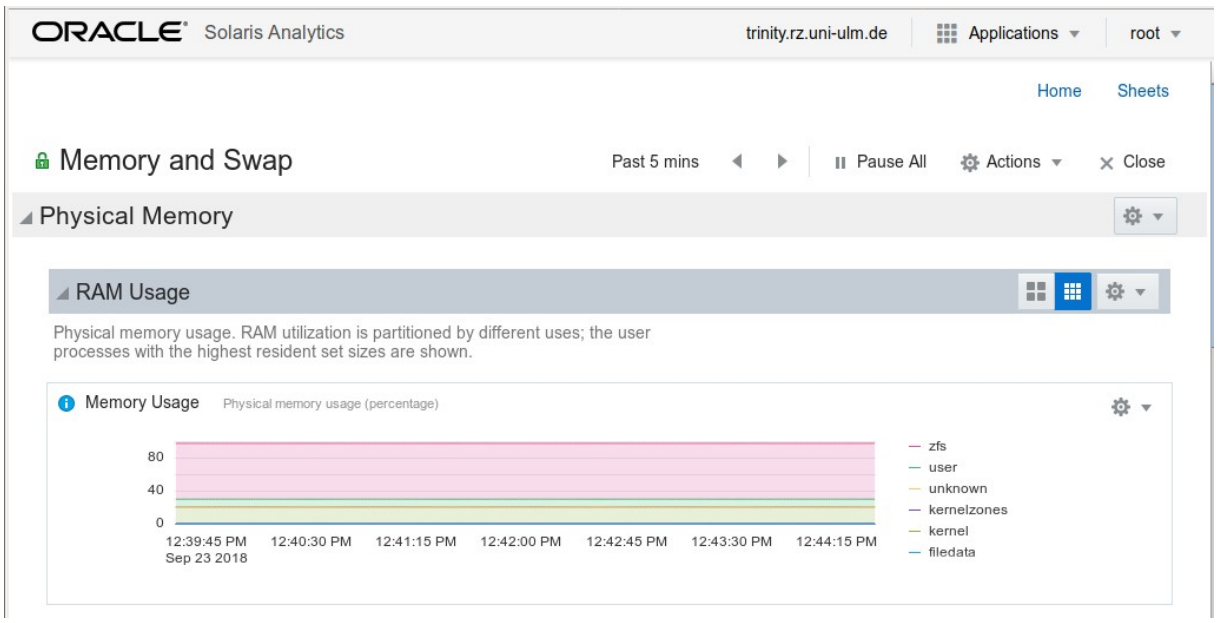


*Abbildung 2: Memory und Swap*

Doch auch im Kleinen finden sich durchaus hilfreiche Änderungen was das Monitoring des Systems angeht. Neben Erweiterungen in DTrace, dies wird der Autor in einem eigenen Artikel behandeln, lassen sich mit *fsstat(8)* nun auch IO-Latenzen in Filesystemen messen.

```
# fsstat -l zfs 1
 read read   read write write write rddir rddir rddir
  ops bytes  time   ops bytes  time   ops bytes  time
63.6M 1.28T 9.00n 23.0M  697G  615n  267M 58.6G 1.52u zfs
```

```
  12 2.49K 54.0n     1 4.11K 37.0n   780 32.5K  183n zfs
   0     0    0n     3 11.3K 23.0n     2   296 99.0n zfs
   0     0    0n     0     0    0n     0     0    0n zfs
```

**Security und Compliance:**
Nicht nur für Umgebungen, die verpflichtet sind Standards einzuhalten und dies nachzuweisen bringt der mit 11.3 eingeführte *security compliance check* Vorteile. Auch alle anderen Umgebungen lassen sich mit dem Tool einfach hinsichtlich einer guten Basiskonfiguration überprüfen. Mit 11.4 können diese Tests nun auch zentral gesteuert von einem System aus durchgeführt werden. Für Details sei an dieser Stelle lediglich auf die Dokumentation verwiesen.

`https://docs.oracle.com/cd/E37838_01/html/E61020/index.html`

Wesentlich für uns war die Umstellung des lokalen *packet filter.* Wurde *IPfilter* mit Solaris 10 eingeführt, kam in 11.3 der OpenBSD basierte PF hinzu. Dieser ist nach dem Wegfall des *IPfilter* supports in 11.4 der einzig unterstützte Code.

Die Migration der meist sehr einfachen lokalen Regeln gestaltet sich meist trivial. Insbesondere da PF eine weitaus mächtigere Regelsprache bereit stellt als dies bei *IPfilter*, auch im Zusammenspiel mit *IPpools*, der Fall war. Auch die Möglichkeit, mittels *include statement* die Regeln auf mehre Dateien zu verteilen hat sich in der Praxis als nützlich erwiesen. *Macros* stellen dabei Nutzer spezifische Variablen zur Verfügung, die IP-Adressen, Portnummer aber auch PF Listen enthalten können.

```
friend1      = "192.168.1.1"
friend2      = "192.168.1.2"
all_hosts  = "{" $friend1 $friend2 "}"
```

*Macros* werden zu dem Zeitpunkt zu dem der Regelsatz geladen wird als Textersatz ausgewertet. Sie sind daher nicht dynamisch und erzeugen , im Falle langer Listen, auch nur bedingt effiziente Regeln. Abhilfe schaffen *tables*. Die darin abgelegten IP-Adressen bzw. IP-Bereiche können jederzeit und ohne den Regelsatz neu zu laden dynamisch verändert werden. Darüber hinaus ist das Durchsuchen sehr effizient. Laut OpenBSD Handbuch werden Tabellen mit 50.000 Einträgen nahezu gleich schnell durchsucht wie kurze mit lediglich 50 Adressen. Der Syntax ist einfach:

```
table <goodguys> { 192.0.2.0/24 }
table <rfc1918>  const { 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
table <spammers> persist
```

Das Schlüsselwort `persist` weist dabei den Kernel an, die Tabelle im Speicher zu halten, auch wenn sie aktuell keine Einträge enthält. So lässt sie sich dann dynamisch anpassen, entweder mit Hilfe des PF control tools

```
# pfctl -t spammers -T add 203.0.113.0/24
# pfctl -t spammers -T show
# pfctl -t spammers -T delete 203.0.113.0/24
```

oder aber über die Filterregeln selber. Die nachfolgende Regel sorgt dafür, dass Hosts mit mehr als 16 Verbindungen zum lokalen System bzw. solche mit mehr als 15 Verbindungsversuchen in 5 Sekunden der Tabelle `<bruteforce>` hinzugefügt werden. Gleichzeitig erzwingt `flush global`, dass alle Einträge in der PF state table für die IP-Adresse gelöscht werden. **Achtung:** im Zusammenspiel mit dem `block` statement kann diese alle bestehenden Verbindungen von der Quelladresse unterbrechen.

```
  # ssh access with scanning protection
pass in inet proto tcp from any to any port 22 flags S/SA keep state  \
     ( max-src-conn 16, max-src-conn-rate 15/5, overload <bruteforce> \
       flush global )
block in quick from <bruteforce>
```

Um Adressen nicht dauerhaft zu blockieren, sorgt ein cron-job dafür, dass periodisch alle Einträge aus der `<bruteforce>` Tabelle entfernt werden, die mindestens 30 Minuten alt sind.

```
# crontab -l | grep pfctl
* * * * * /usr/sbin/pfctl -q -t bruteforce -T expire 1800
```

**Zonen:**
ZOSS (zones on shared storage) unterstützen mit 11.4 auch NFS als zu Grunde liegendes Filesystem. Zwar ist hinsichtlich der Performance iSCSI als Basistechnologie je nach Anwendung nach unserer Erfahrung im Vorteil, jedoch ist NFS für die meisten Administratoren sehr viel vertrauter und für eine Vielzahl von Szenarien daher als Basis einfacher aufzusetzen.

Praktisch ist auch die neue Möglichkeit ZFS datasets dynamisch an laufende Zonen als dortigen Pool delegieren zu können. Vor 11.4 war hier ein Neustart der Zone erforderlich.

```
trinity# zfs create tank/dynamic
trinity# zonecfg -z cifs -r "add dataset; set name=tank/dynamic; end"

cifs# zpool import dynamic
cifs# zpool list
NAME      SIZE  ALLOC   FREE  CAP  DEDUP    HEALTH  ALTROOT
dynamic   278G   141G   137G  50%  1.00x    ONLINE  -
rpool    97.5T  25.4T  72.1T  26%  1.00x    ONLINE  -
smb      97.5T  25.4T  72.1T  26%  1.00x    ONLINE  -
cifs# zpool export dynamic

trinity# zonecfg -z cifs -r "remove dataset name=tank/dynamic;"
```

Mit dem neuen Solaris Release wird nun auch jede Zone durch einen eigenen SMF-Service verwaltet. Damit können nun nicht nur Abhängigkeiten zwischen den Zonen sondern auch hinsichtlich anderer SMF-Services der globalen Zone definiert werden.

Zusätzlich zu den explizit definierten Abhängigkeiten kann die Priorität einer Zone beim Bootvorgang noch generisch spezifiziert werden. Dazu verwaltet der *zone-restarter* drei Queues die von hoher hin zu niedriger Priorität abgearbeitet werden. Mittels *zonecfg(8)* lässt sich eine Zone einer dieser drei Queues zuordnen. Weitere Details sind in *svc.zones(8)* zu finden.

```
set boot-priority={ high | normal | low }
```

Jan Pechanec und Joost Pronk Van Hoogeveen haben die Möglichkeiten sehr gut zusammengefasst:

```
https://blogs.oracle.com/solaris/zones-delegated-restarter-and-smf-goals
https://community.oracle.com/docs/DOC-1025366
```

**Free Open Source Software (FOSS):**
Der Rückgang der Verbreitung von Solaris, vor allem im Bildungsumfeld und kleineren Umgebungen, macht sich insbesondere bei der Verfügbarkeit von aktuellen Softwarepaketen in frei zugänglichen Repositorien bemerkbar. So halten die engagierten Personen hinter OpenCSW[3] nach wie vor jede Menge davon bereit, jedoch machen die Abhängigkeiten oft die Installation einer Vielzahl von Paketen notwendig die als (zu) alte Version auch in Solaris zu finden sind.

Noch problematischer ist die Tatsache, dass viele kleinere Open Source Projekte unter Solaris nur noch mit erheblichem Aufwand überhaupt zu compilieren sind, da die Entwickler keinen Zugang zu Testsystemen oder kein Interesse an Solaris haben.

Erfreulich ist daher die Tatsache, dass Solaris 11.4 hier mit häufig eingesetzten Programmen wie Apache, PHP, den Skriptsprachen Python und Perl aber auch der GNU Entwicklungsumgebung sehr viel moderner geworden ist. Bleibt zu hoffen, dass die kommenden Software Updates (SRUs) hier öfter aktualisieren als dies zum Teil in der Vergangenheit der Fall war.

Die folgende Tabelle fasst einige für uns wichtige Versionsstände zusammen (Stand 09/2018). Nicht berücksichtigt sind Erweiterungen wie etwa Perl-Module, ...

| **Software** | **Solaris 11.4** | **Web** |
|---|---|---|
| Apache (2.4) | 2.4.33 | 2.4.34 |
| PHP | 5.6.36, 7.1.17 | 5.6.38, 7.1.22 |
| Perl | 5.22.1, 5.26.2 | 5.22.4, 5.26.2, 5.28.5 |
| Python | 2.7.14, 3.4.8, 3.5.3 | 2.7.15, 3.4.9, 3.5.6 |
| GCC | 5.5, 7.3 | 7.3, 8.2 |
| Firewall | OpenBSD PF 5.5 | 5.5 |
| Kerberos | MIT 1.16.0 | 1.16.1 |
| LDAP | OpenLDAP 2.4.45 | 2.4.46 |
| Samba | 4.7.6 | 4.7.10, 4.9.0 |
| SSH | OpenSSH 7.5 | 7.8 |

Neu dabei sind unter anderen auch GO und LLVM/Clang 6.0. Neben dem *refresh* hat sich Solaris 11.4 auch von einigen alten Sonderlösungen getrennt. So sind die noch auf Sun zurückgehenden eigenen Implementierungen *Mozilla LDAP, SunSSH* aber auch *IPfilter* in 11.4 verschwunden. Sicherlich kein Nachteil für die Zukunft. Differenzierter ist dies aber sicherlich bei dem nun als „obsolete“ markierten OpenStack zu sehen, wurde es doch erst mit 11.2 eingeführt.

**Weitere Kleinigkeiten:**
Sowohl die Abschnitte 1m, 4, 5 und 7 der online-manuals als auch die zugehörigen Unterbereiche wurden in die Abschnitte 8, 5, 7 und 4 verschoben. Damit verwendet Solaris nun die selbe Einteilung wie BSD, MacOS und Linux.

3 https://www.opencsw.org/

Hilfreich für die Anzeige eines bestimmten Manuals ist die Unterstützung der folgendes Syntax:

```
# man printf.3
```

Unified Archives (UAR) unterstützen für clone-Archive, nicht für recovery-Archive, die sogenannte dehydration. Dabei werden nicht-editierbare Elemente und hardlinks aus dem Archiv entfernt sofern das zugehörige IPS-Paket dies zulässt. Bei der Installation des Archivs wird dann auf die Pakete des Hosts bzw. dessen publisher zurück gegriffen. Damit einher geht auch eine deutliche Reduktion des Platzbedarfs. Für ein typisches Serversystem ohne 3rd-party Anwendungen schrumpft das Archiv von 2.6GB auf nur 0.8GB. Diese Art von Archiven ist darüber hinaus auch für ISVs interessant da hier das Urheberrecht von Oracle sehr viel einfacher zu berücksichtigen ist.

**Zusammenfassung:**

In dem mehr als zwei Jahre 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, wie *Analytics,* wurden in dieser Zeit auch eine ganze Reihe weiterer Verbesserungen eingeführt, die in vielen Bereichen 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 |