ulm university universität ulm

Live long and prosper
Solaris 11.4 Vorteile in der Praxis

Thomas Nau, kiz (Thomas.Nau@uni-ulm.de)

Übersicht

- Hintergrund
- ZFS und Fileservices
- DTrace
- StatsStore und Analytics
- Security
- Zonen
- Gemischtwarenladen
- Q & A

Über mich

- eigentlich Physiker
- stlv. Direktor des kiz und Leiter der Abteilung „Infrastruktur“
  - Balanceakt zwischen Technik und Management
- erste IT Berührung mit einer PDP-11 vor (sehr) langer Zeit
- Schwerpunkte UNIX und Storage Lösungen

Die Vorteile einer Abteilung

- ein Team, ein Ziel
  - Server- und Desktop-Betrieb, Netz- und TK-Gruppen sind Bestandteil des Teams
    - kein Elfenbeinturm sondern **„reality exposure“** und **„eat your own dog food“** Philosophie
    - Entscheidungen sind einfacher zu treffen und gemeinsam zu tragen
    - hilft realistische und umsetzbare Ziele zu definieren
- spannendes Arbeitsumfeld mit vielen Freiheiten für Leute die bereit sind Verantwortung zu übernehmen
  - „erlaubt ist was funktioniert und wartbar ist“
    bleeding edge ist kein Tabu

One team to serve them all

- die „Abteilung Infrastruktur“ des kiz erbringt Dienstleistungen für ~18.000 Personen
  - Netzwerke (LAN, MAN und WLAN) und Telefonie
    - Anbindung an das Landeshochschulnetz BelWue mit 100 Gbit
  - Betrieb der gesamten zentrale Server-IT inklusive ca. 700 Desktop PCs sowie Laptops
  - Dienstleistungen im Rahmen von Landeskooperationen
    - Backup Service für mehrere Universitäten in Baden-Württemberg
    - HPC Landes-Cluster „Computergestützte Chemie und Quantenwissenschaften“
    - bwCloud: „IaaS für Forschung und Lehre in Baden-Württemberg“
  - bedient Cloud-Hype, Next-Hype, Whatever-else Hype

Der Bericht über meinen Tod wurde stark übertrieben. (Mark Twain)

Quelle: Mark Twain im New York Journal, am 2. Juni 1897

Solaris Hintergrund

- auch nach 25+ Jahren noch eine unserer strategischen Server-Plattformen
  - viele Kooperationen reichen weit in die Sun Ära zurück
- Fokus auf Storage bzw. Filespace provider
  - NFS und CIFS-Server für > 600 clients, iSCSI Server, NFS Server für VMs, Mail, Backup, Nextcloud, ...
  - ZFS hat uns in der Vergangenheit oft gerettet, etwa bei Ransomware Befall, …
- Rolle im Anwendungsbereich rückläufig da ISV Support an vielen Stellen schwindet
  - Ausnahmen: Apache, SAP, DB-Server (PostgreSQL, Oracle), ...
- Gerüchte über den Tod von Solaris waren **nicht** hilfreich ebenso wie das diesbezügliche Schweigen von Oracle

Top Feature #1: ZFS und Fileservices

- weiterhin eines der herausragenden Merkmale von Solaris
  - leider wurde viel Potential durch die closed source Entwicklung vergeudet → Linux/FreeBSD haben inkompatible Pool Formate
  - OpenZFS hat viel Dynamik, sicherlich auch getrieben durch den „Abgesang auf Solaris“

ZFS Erweiterungen

ZPool Version	Feature
37 (Solaris 11.3)	LZ4 compression
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 (Solaris 11.4)	Device removal

ZFS „asynchronous dataset destroy“

- beschleunigt aus Sicht der Anwendung das Löschen von ZFS datasets
  - Solaris Kernel übernimmt die Funktion asynchron
  - Vorgang kann überwacht werden

```
# 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
...
```

ZFS „reguid: ability to change the pool guid“

- in OpenZFS bereits seit längerem verfügbar
- weist ZPools eine neue GUID (eindeutiger Identifier) zu
  - zwingend notwendig wenn z.B. rpools für virtuelle Solaris Systeme aus templates erzeugt werden
  - pools mit den identischen GUIDs lassen sich nicht am selben System mounten

ZFS „reguid: ability to change the pool guid“

```
# mkfile -v 1g /var/tmp/pool1_dev

# zpool create pool1 /var/tmp/pool1_dev

# zpool export pool1
# zpool import -d /var/tmp/pool1_dev
  pool: pool1
    id: 15312272360447757 55
 state: ONLINE
action: The pool can be imported using its name or numeric
identifier.
config:

        pool1                   ONLINE
          /var/tmp/pool1_dev    ONLINE
```

ZFS „reguid: ability to change the pool guid“

```
# cp -p /var/tmp/pool1_dev /var/tmp/pool2_dev

# zpool import -d /var/tmp/pool2_dev
  pool: pool1
    id: 15312272360447757 55
 state: ONLINE
action: The pool can be imported using its name or numeric
identifier.
config:

        pool1                   ONLINE
          /var/tmp/pool2_dev    ONLINE

# zpool import -d /var/tmp/pool2_dev pool2
cannot import 'pool2': no such pool available

# zpool import -d /var/tmp/pool2_dev \
        15312272360447757 55 pool2
```

ZFS „reguid: ability to change the pool guid“

```
# zpool list
NAME    SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
pool2  1016M   104K  1016M   0%  1.00x  ONLINE  -
rpool   278G  98.7G   179G  35%  1.53x  ONLINE  -

# zpool import -d /var/tmp/pool1_dev pool1
cannot import 'pool1': no such pool available

# zpool import -d /var/tmp/pool1_dev \
                15312272360447757 55 pool1
cannot import '15312272360447757 55':
a pool with that name is already created/imported,
and no additional pools with that name were found

# zpool reguid pool2

# zpool get guid pool2
NAME   PROPERTY  VALUE                  SOURCE
pool2  guid      3415702692117888181  -
```

ZFS „reguid: ability to change the pool guid“

```
# zpool import -d /var/tmp/pool1_dev \
        15312272360447755755 pool1

# zpool list
NAME    SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
pool1  1016M   150K  1016M   0%  1.00x  ONLINE  -
pool2  1016M   106K  1016M   0%  1.00x  ONLINE  -
rpool   278G  98.7G   179G  35%  1.53x  ONLINE  -

# zpool get guid pool1 pool2
NAME   PROPERTY  VALUE                  SOURCE
pool1  guid      15312272360447755755   -
pool2  guid      3415702692117888181    -
```

ZFS „device removal“

- meiner Einschätzung nach einer der ältesten feature-requests in Solaris ZFS
- wurde in mehreren Schritten implementiert
  - das Entfernen von cache-, log-, spare und meta-devices ist schon seit längerem möglich, ebenso wie das detaching von Platten aus gespiegelten vdevs
- **GAF** (größter anzunehmender Fehler) ist das Verwechseln von `zpool attach` mit `zpool add`
  - bisherige Abhilfe bestand darin den Pool neu aufzusetzen und die Daten aus dem Backup o.ä. zu restaurieren
    - ggf. Verlust der snapshots
- in 11.4 können *vdevs* entfernt werden sofern genügend Kapazität im Pool verbleibt

ZFS „device removal“

```
# mkfile -v 8g /tank/dev1 /tank/dev2
/tank/dev1 8589934592 bytes
/tank/dev2 8589934592 bytes

# zpool create test /tank/dev1

# 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  -
```

ZFS „device removal“

```
# zpool add test /tank/dev2

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

        NAME              STATE       READ WRITE CKSUM
        test              ONLINE         0     0     0
          /tank/dev1      ONLINE         0     0     0
          /tank/dev2      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   -
```

FAIL!

ZFS „device removal“

zpool monitor -t resilver
zeigt den aktuellen Fortschritt an

```
# zpool remove test /tank/dev2

# zpool attach test /tank/dev1 /tank/dev2

# 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 …

config:
        NAME                    STATE     READ WRITE CKSUM
        test                    ONLINE       0     0     0
          mirror-0              ONLINE       0     0     0
            /tank/dev1          ONLINE       0     0     0
            /tank/dev2          ONLINE       0     0     0
```

ZFS „device removal“, doing crazy things

- Pool Layout umwandeln: 3-fach Spiegel → RAID-Z2
- Variante #1
  - aus jedem *mirror-vdev* eine Platte entfernen und mit diesen ein *RAID-Z2-vdev* anhängen
    - Nachteil: Redundanz sinkt auf n+1
    - Vorteil: im Pool vorhandener freier Platz ist primär unerheblich jedoch ist ggf. eine Kombination mit Variante #2 notwendig
- Variante #2
  - es werden komplette *mirror-vdev* entfernt und aus diesen Platten ein *RAID-Z2-vdev* an den Pool angehängt
    - Vorteil: immer n+2 Redundanz
    - Nachteil: Umwandlung bei hoher Platzbelegung nicht möglich

ZFS „device removal“, doing crazy things

```
# zpool iostat -v test
                  capacity     operations     bandwidth
pool           alloc   free   read  write   read  write
-------------  -----  -----  -----  -----  -----  -----
test           1.24G  6.70G      0     91     82  9.58M
  mirror-0      315M  1.68G      0     23     52  2.41M
    /tank/d01      -      -      0      5    184  2.41M
    /tank/d02      -      -      0      5    131  2.41M
    /tank/d03      -      -      0      5    131  2.41M
  mirror-1      316M  1.68G      0     21      0  2.38M
    /tank/d04      -      -      0      4    131  2.39M
    /tank/d05      -      -      0      4    131  2.39M
    /tank/d06      -      -      0      4    131  2.39M
  mirror-2      320M  1.67G      0     23     30  2.38M
    /tank/d07      -      -      0      5    162  2.39M
    /tank/d08      -      -      0      5    131  2.39M
    /tank/d09      -      -      0      6    131  2.39M
  mirror-3      320M  1.67G      0     23      0  2.41M
    /tank/d10      -      -      0      5    131  2.41M
    /tank/d11      -      -      0      5    131  2.41M
    /tank/d12      -      -      0      5    131  2.41M
```

ZFS „device removal“, doing crazy things

```
# zpool remove test mirror-2 mirror-3

# zpool status -v test
…
  scan: resilver in progress since Sun Nov  4 11:41:43 2018
    1.24G scanned
    640M resilvered at 53.4M/s, 100.00% done, 1s to go
config:

        NAME                      STATE     READ WRITE CKSUM
        test                      ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            /tank/d01             ONLINE       0     0     0
            /tank/d02             ONLINE       0     0     0
            /tank/d03             ONLINE       0     0     0
…
          mirror-2                REMOVING     0     0     0
            /tank/d07             REMOVING     0     0     0
            /tank/d08             REMOVING     0     0     0
            /tank/d09             REMOVING     0     0     0
…
```

ZFS „device removal“, doing crazy things

```
# zpool add test raidz2 /tank/d{07,08,09,10,11,12}

# zpool status -v test
…
        NAME                      STATE     READ WRITE CKSUM
        test                      ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            /tank/d01             ONLINE       0     0     0
            /tank/d02             ONLINE       0     0     0
            /tank/d03             ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            /tank/d04             ONLINE       0     0     0
            /tank/d05             ONLINE       0     0     0
            /tank/d06             ONLINE       0     0     0
          raidz2-4                ONLINE       0     0     0
            /tank/d07             ONLINE       0     0     0
            /tank/d08             ONLINE       0     0     0
            /tank/d09             ONLINE       0     0     0
            /tank/d10             ONLINE       0     0     0
            /tank/d11             ONLINE       0     0     0
            /tank/d12             ONLINE       0     0     0
```

ZFS „device removal“, doing crazy things

```
# zpool remove test mirror-0 mirror-1

# zpool status -v test
  pool: test
 state: ONLINE
  scan: resilvered 1.25G in 31s with 0 errors on …

config:

        NAME                      STATE      READ WRITE CKSUM
        test                      ONLINE        0     0     0
          raidz2-4                ONLINE        0     0     0
            /tank/d07             ONLINE        0     0     0
            /tank/d08             ONLINE        0     0     0
            /tank/d09             ONLINE        0     0     0
            /tank/d10             ONLINE        0     0     0
            /tank/d11             ONLINE        0     0     0
            /tank/d12             ONLINE        0     0     0

errors: No known data errors
```

Weitere neue ZFS features

- scheduled scrub
  - voreingestellt startet scrubbing einmal pro Monat für jeden Pool
  - wird über die *zpool-property* `scrubinterval` gesteuert; auf `manual` gesetzt wird der Automatismus für diesen Pool deaktiviert
  - die *property* `lastscrub` zeigt das Datum des letzten Laufs an
- raw send streams
  - die Daten komprimierter ZFS datasets wurden bei der Übertragung mittels `zfs send / receive` erst dekomprimiert und dann wieder komprimiert
  - diese Ineffizienz läßt sich mit `zfs send -w compress` ab Solaris 11.4 vermeiden

Weitere neue ZFS features

- resumable send streams
  - Verbindungsabbrüche bei der Übertragung von ZFS datasets mittels `zfs send / receive` waren in der Vergangenheit „ein Quell von Freude“ für Administratoren
    - alle Daten des betroffenen snapshots mussten neu gesendet werden, unerheblich ob der Verbindungsabbruch zu Beginn oder kurz vor Ende stattfand
  - Solaris 11.4 fügt zusätzliche Informationen in den stream ein die ein wieder Aufsetzen unter Erhalt der gesendeten Daten erlaubt
    - mit `zfs send -s nocheck` lässt sich das neue default Verhalten abschalten
    - die Option `-C` wertet sowohl für `send` als auch `receive` die neuen Metadaten aus

ZFS „resumable send streams“

- Beispiel: lokale Replikation wird nach 15 Sekunden abgebrochen indem die Prozesse beendet werden

```
# 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
```

ZFS „resumable send streams“

- die Option `-C` erlaubt es Sender und Empfänger auf den letzten erfolgreich verarbeiteten Block zu synchronisieren
  - die Ausgabe wurde zur Verbesserung der Lesbarkeit 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 …
```

Und noch mehr neue ZFS nahe features

- SMB 3.1.1 Unterstützung
  - unterstützt Verschlüsselung sowie mehrere parallele Datenkanäle was den Durchsatz steigert
- NFS 4.1
- ZFS *properties* `defaultreadlimit`, `readlimit`, `effectivereadlimit`, `defaultwritelimit`, `writelimit` und `effectivewritelimit`
  - helfen die IO-Bandbreite die einem Pool zur Verfügung steht gerechter auf einzelne datasets zu verteilen
- `cp -z`
  - kopiert sehr effizient Dateien mittels ZFS *copy-on-write* sofern sich Quelle und Ziel im selben ZPool befinden
  - siehe auch `reflink(3C)`

Solaris 11.4
DTrace update

Heute 12:00

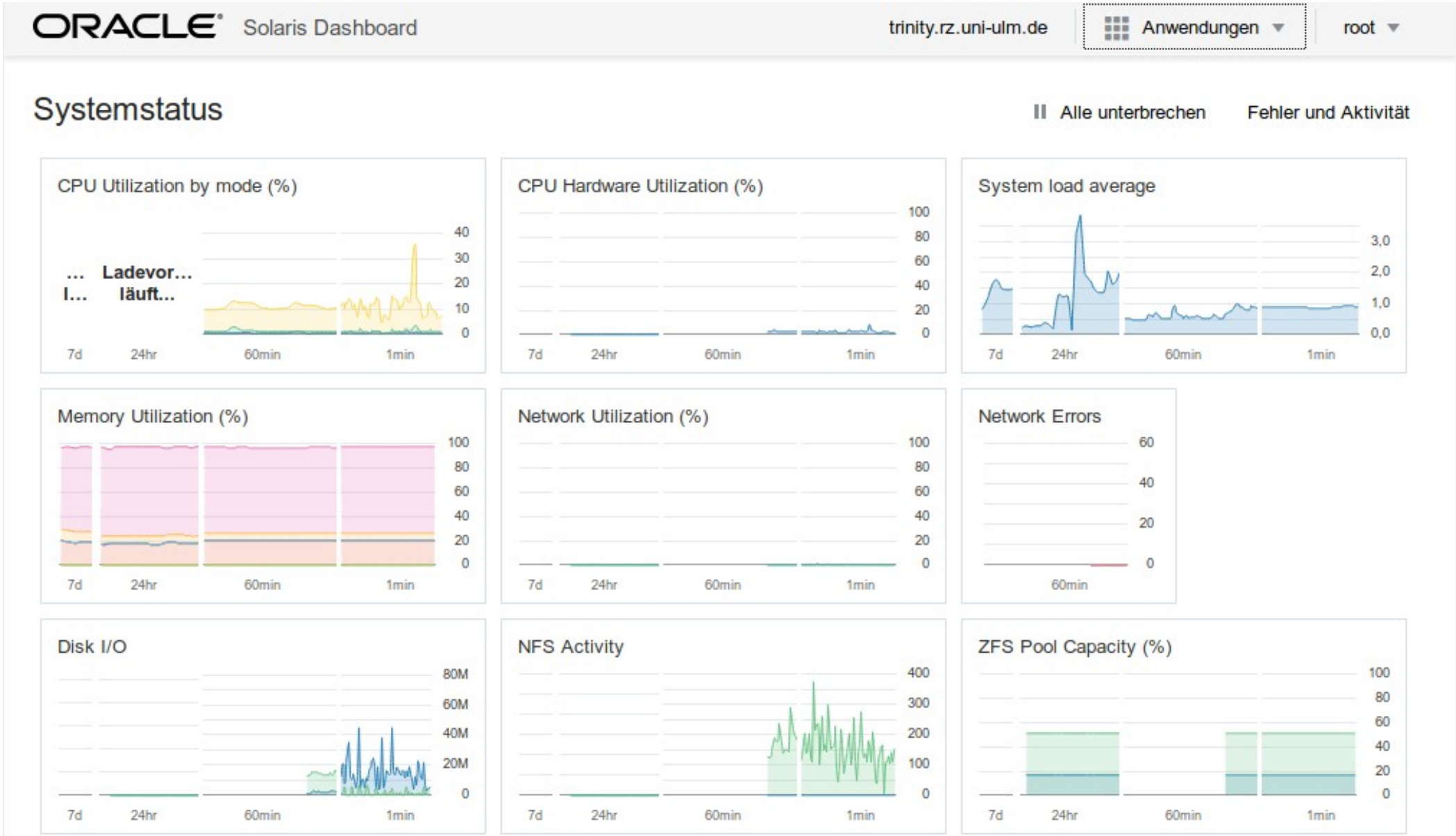
Solaris 11.4
Collecting data

StatsStore

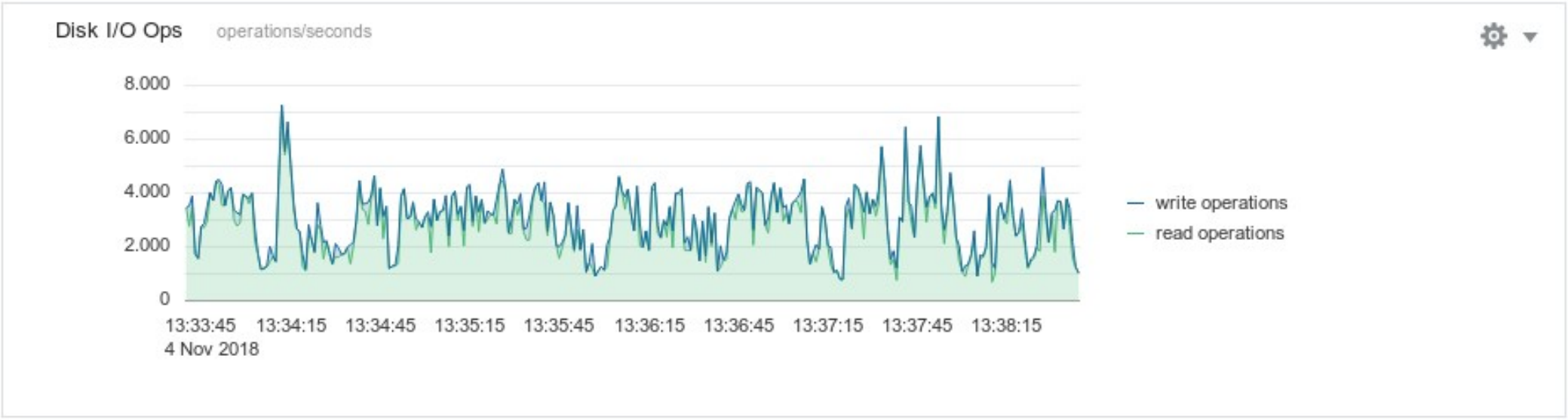
- vereinheitlicht Sicht auf Performance-Daten und Ereignisse
  - kstat(), audit-event, fault-events (FMA), …
  - Hardware events
  - Informationen über Prozesse, ...
- einheitlicher Namensraum
  - Klassen, Ressourcen, Statistiken und Partitionen
- automatisierte Erfassung und Konsolidierung durch SMF-service

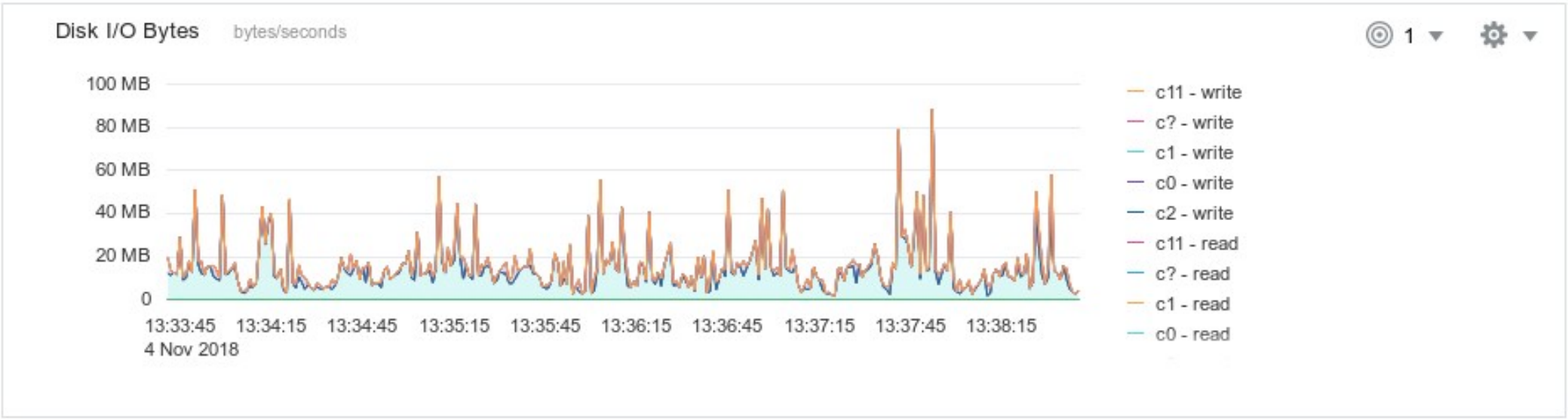
  svc:/system/sstore
- Ausgabe auch im CSV oder JSON Format
  - ssid(7), sstore.csv(5), sstore.json(5)

StatsStore Zugriff via Dashboard



Disk-IO





Security und Compliance

- der mit 11.3 eingeführte *security compliance check* läßt sich nun zentral steuern

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

- der aus OpenBSD stammende packet-filter *PF* übernimmt die Aufgaben des in 11.3 als deprecated geführten *IPfilter*
  - die Konvertierung der Regeln ist meist trivial und wird von einem Skript auf Wunsch unterstützt
  - PF besitzt eine deutlich mächtigere Regelsprache u.a. mit *include-statements* und *macros*
  - mit *tables* lassen sich effizient und dynamisch lange Listen mit Hosts oder Netzwerken in Regeln verwalten
    - laut OpenBSD Handbuch lassen sich Tabellen mit 50.000 Einträgen ebenso schnell durchsuchen wie herkömmliche Listen mit nur 50 Einträgen

PF packet filter

- Beispiele für *macros* und *tables*

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


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


# 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
```

This block is reserved for use in documentation and should not be used in any real networks.
Please see more details at
http://www.iana.org/go/rfc5737

PF packet filter

- „dynamische“ Regeln um scanner zu blockieren

```
  # 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>
```

- cleanup

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

Zone update

- NFS wird als Speicher für *zones-on-shared-storage* unterstützt
  - ist den Admins vertrauter als iSCSI jedoch ist letzteres unserer Erfahrung nach performanter
- jede Zone wird durch eigenen SMF-service verwaltet
  - dadurch können Abhängigkeiten beim Start berücksichtigt werden
  - zusätzlich verwaltet der *zone-restarter* drei queues unterschiedlicher Priorität: high, normal, low
  - sehr gute Zusammenfassungen von Jan Pechanec und Joost Pronk Van Hoogeveen:

    https://blogs.oracle.com/solaris/zones-delegated-restarter-and-smf-goals

    https://community.oracle.com/docs/DOC-1025366

Zone update

- ZFS datasets können an laufende Zonen dynamisch, also ohne einen Neustart zu erfordern, delegiert werden

```
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;"
```

Noch einige Kleinigkeiten

- Free and Open Source Software (FOSS)
  - update auf viel aktuellere Versionen, etwa PHP 7.1, GCC 7.3
  - Wechsel auf de-facto Standards
    - SunSSH → OpenSSH 7.5
    - Sun Mozilla LDAP → OpenLDAP 2.4.45
    - IPFilter → OpenBSD PF
  - neu dabei sind u.a. GO und LLVM/Clang 6.0
- Anpassung der manual-sections an BSD, MacOS und Linux: 1m, 4, 5, 7 → 8, 5, 7, 4
  - **Tipp:** `man prinf.3`
- dehydration von *unified-archives* (nur für clone archives)
  - hilft ISVs bei der Weitergabe von Archiven

Q & A