(V10) Linux News - Kernel Update / SLES10

Christian Bornträger
cborntra@de.ibm.com

Agenda

- **Binärkernelmodule (OCOs)**
- **Kernel**
  - Performance
  - Operating
  - Hochverfügbarkeit
  - Sicherheit
  - Virtualisierung
- **Compiler**
  - Allgemeine Verbesserungen
  - z-spezifische Verbesserungen
- **SLES10**

Binärkernelmodule/OCOs (Object-code only)

- **Grundprobleme**
  - OCO-Module müssen mit jeder Kernelversion neu erstellt werden
  - Distributionen sind nicht bereit, OCO-Module zu integrieren
- **Es gibt derzeit keine OCO-Module aus Böblingen**
  - lcs: Open Source seit 04.03.2002, integriert in 2.4.x
  - z90crypt: Open Source seit 31.07.2002, integriert in 2.4.x
  - qdio: Open Source seit 13.09.2002, integriert in 2.4.x
  - qeth: Open Source seit 30.06.2003, integriert in 2.4.x
  - tape_3590: Open Source seit 28.03.2006, integriert in 2.6.17
- **Strategie: Keine weiteren OCO-Module**

System z Kernel - Performance

- **Verbesserung der Skalierbarkeit**
  - TCP Segmentation Offload *(2.6.12, DW 1Q05, SLES9 SP2, RHEL4 U4)*
  - OSA 1920 *(2.6.12, DW 1Q05, SLES9 SP2, RHEL4 U4)*
  - Multiple Subchannel Sets *(2.6.16, DW 1Q06, SLES10)*
  - Linux PAV für LPAR *(2.6.18, kein DW)*
- **Zugriff auf Statistikdaten**
  - FCP Performance Statistiken *(unter Diskussion, DW 4Q05, SLES10)*
  - Zugriff auf Performancedaten für Kanalpfade *(in 2.6.17, no DW)*
  - Zugriff auf LPAR Performance Daten *(in 2.6.18, no DW, RHEL4 U4)*
- **Profiling**
  - OProfile Unterstützung *(in 2.6.12, DW 1Q05, SLES9 SP2, RHEL4)*
  - OProfile Kernel Call-Graph Unterstützung *(in 2.6.16, DW 1Q06, SLES10)*

Applikations- und Kernelprofiling (1)

- **OProfile ist das bedeutendste (Sampling) Profilingwerkzeug für Linux**
- **Gleichzeitiges Profiling für Applikationen und Kernel**
- **Gedacht für Entwicklung und Problemanalyse, nicht Überwachung**
- **Aktuellste Version http://oprofile.sourceforge.net/**
- **RHEL4: CD3 , SLES9: im SDK, SLES10: im SDK auf CD1**
- **Seit Kernel 2.6.16 wird Kernel-Callgraph-Profiling unterstützt**
  - Welche Funktion wird von welcher wie oft gerufen?
- **Vorbereitung: OProfile benötigt die Datei vmlinux für das Kernel Layout**

```
# gunzip /boot/vmlinux-2.6.16.21-0.8-default.gz
# opcontrol --vmlinux=/boot/vmlinux-2.6.16.21-0.8-default
```

- **Einfache Befehle zum Starten und Stoppen**

```
# opcontrol --start
# opcontrol -stop
```

- **Auslesen der Daten**

```
# opcontrol --dump
```

- **Anzeigen der Daten**

```
# opreport -l
```

Applikations- und Kernelprofiling (2)

- **Debug Pakete bzw. Programme mit Symbolen verbessern die Daten**
- **Beispiel: yast2 Aufruf**

```
53v15g06:/ # opreport -l
CPU: CPU with timer interrupt, speed 0 MHz (estimated)
Profiling through timer interrupt
samples  %        app name                 symbol name
23510    93.1975  vmlinux-2.6.16.21-0.8-default cpu_idle
293       1.1615  libstdc++.so.6.0.8       (no symbols)
218       0.8642  libzypp.so.1.0.2         (no symbols)
133       0.5272  libqt-mt.so.3.3.5        (no symbols)
119       0.4717  libboost_regex.so.1.33.1 (no symbols)
76        0.3013  ld-2.4.so                do_lookup_x
71        0.2815  libc-2.4.so              _int_malloc
63        0.2497  libc-2.4.so              malloc
53        0.2101  libc-2.4.so              free
45        0.1784  librpmdb-4.4.so          (no symbols)
44        0.1744  libc-2.4.so              _int_free
....
```

System z Kernel – Vereinfachung des Operating

- **Unterstützung des Communication Controller für Linux**
  - NCP CDLC Verbindungen *(2.6.15, DW 4Q05, SLES9 SP2, RHEL4 U3)*
  - OSA Layer 2 Sequenznummern *(2.6.14, DW 4Q05,SLES9 SP2, RHEL4 U3)*
- **FCP-Verbesserungen**
  - Punkt-zu-Punkt Verbindungen *(2.6.12, DW 1Q05, SLES9 SP2)*
  - SCSI Re-IPL *(2.6.14, DW 4Q05, SLES9 SP3)*
  - Zugriff auf die SCSI IPL Parameterliste *(2.6.15, DW 4Q05)*
- **z/VM Integration**
  - Ausführen von CP Befehlen *(2.6.13, DW 4Q05, SLES9 SP3, RHEL4 U3)*
  - 64-bit VMDUMP Unterstützung *(DW 4Q05, SLES9 SP3, RHEL4 U4)*

z/VM Integration – Ausführen von CP Befehlen (1)

- **vmcp Gerätetreiber**
  - Kerneltreiber zum Ausführen von Diagnose X'08' (virtual console)
  - Das Modul vmcp.ko muss geladen sein: `modprobe vmcp`
  - Devicenode /dev/vmcp (major number 10, dynamische minor)
  - Programme können mit open/write/read/ioctl und close den Devicenode direkt ansteuern
  - Verfügbar in RHEL4, SLES9, SLES10
  - Erfordert root-Rechte
- **vmcp Anwendungsprogramm (/sbin/vmcp)**
  - Setzt auf Gerätetreiber bzw. /dev/vmcp auf
  - Bestandteil der s390-tools
  - CP Befehle absetzen ist einfach:

```
# vmcp q cplevel
z/VM Version 5 Release 2.0, service level 0601 (64-bit)
Generated at 04/06/06 06:37:05 CST
IPL at 07/13/06 08:31:14 CST
```

  - Im Fehlerfall
    - Schreibt vmcp eine Fehlermeldung nach stderr
    - Beendet sich vmcp mit einem Rückgabewert ungleich 0

z/VM Integration – Ausführen von CP Befehlen (2)

- **Rückgabewerte können in bash mittels $? abgefragt werden:**

```
# vmcp <command>
<Antwort und/oder Fehlermeldung>
# echo $?
<0,1,2,3 oder 4>
```

- **0: Alles in Ordnung**
- **1: CP lieferte einen Status ungleich 0**
- **2: Der Anwortspeicher war zu klein. Das Kommando wurde trotzdem ausgeführt, aber die Antwort wurde abgeschnitten. Der Anwortspeicher kann mit der --buffer Option verändert werden**
- **3: Linuxfehler in vmcp. Das Modul ist nicht geladen, /dev/vmcp zeigt auf das falsche Gerät, der Speicher ging aus etc.**
- **4: Es wurde eine fehlerhafte Kommandozeile an vmcp übergeben. Der Syntax sollte überprüft werden.**

System z Kernel - Hochverfügbarkeit

- **Kernel**
  - Verbesserte Machine-check-Bearbeitung *(2.6.13, DW 4Q05, SLES10)*
- **DASD Unterstützung für**
  - Write barriers *(2.6.12, DW 4Q05, SLES10)*
  - Fast fail *(2.6.16, DW 1Q06, SLES10)*
  - Enhanced error reporting *(2.6.17, no DW)*
- **FCP**
  - Best-Effort SAN Fehlermeldungen *(2.6.16, DW 1Q06, SLES10)*

System z Kernel - Sicherheit

- **Hardwareunterstützung - Cryptokarten**
  - Crypto Express 2 Accelerator *(2.6.16, DW 4Q05, SLES9 SP3, RHEL4 U4)*
- **Hardwareunterstützung - z9 Prozessor**
  - Applikationen können AES,SHA und PRNG Crypto CP Assists benutzen
  - Kernel nutzt AES und SHA Crypto CP Assists *(2.6.16, DW 1Q06, SLES10)*
- **Erweiterung**
  - Secure Key Kryptographie *(geplant für 2.6.19, no DW)*

System z Kernel – Virtualisierung

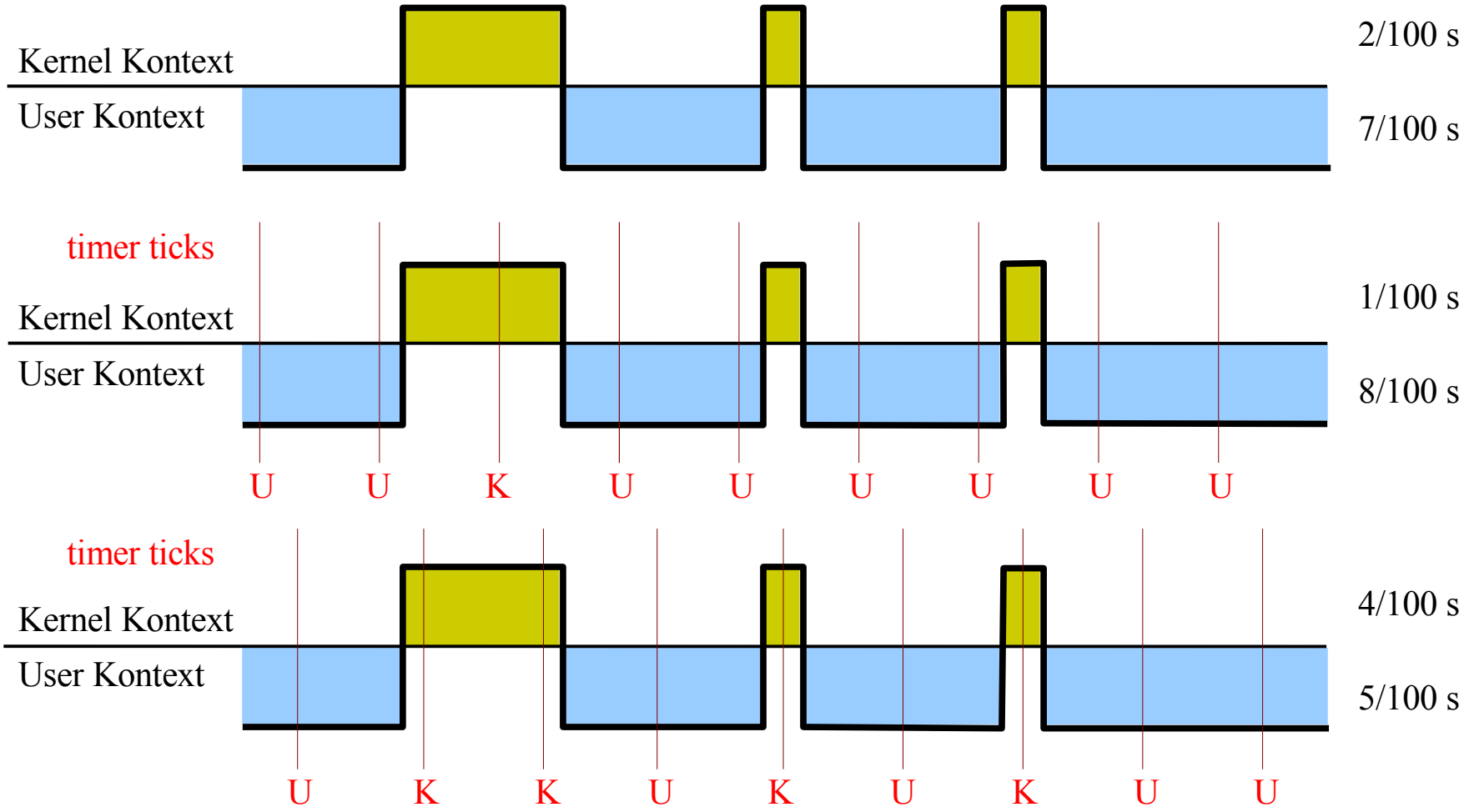
- **Prozessorvirtualisierung**
  - CPU Hotplug *(2.6.8, SLES9 SP1)*
  - Neues CPU- und Prozessaccounting *(2.6.11, DW 4Q05, SLES10)*
  - APPLDATA Erweiterungen für die oben genannten *(2.6.18, no DW)*
- **Verbesserte DCSS-Unterstützung**
  - DCSS EW/EN Layout *(in 2.6.10, DW 1Q05, SLES9 SP2, RHEL4 U4)*
  - DCSS als Swap *(2.6.10, DW 1Q05, SLES9 SP2, RHEL4 U4)*
  - Execute-in-Place Unterstützung in ext2 *(2.6.13, DW 4Q05, SLES10)*
- **Reduzierung des z/VM Overheads**
  - QDIO Pass-Through Stufe 2 *(2.6.16, DW 1Q06, SLES10)*
  - Collaborative Memory Management 2 *(nicht upstream, no DW)*
  - DASD Diag 64bit Support *(2.6.14, DW 1Q06, SLES10)*
- **Neue Funktionalität**
  - z/VM Watchdog *(2.6.10, DW 1Q05, SLES9 SP2, RHEL4 U3)*
  - FCP: N-Port-ID Virtualisierung *(2.6.14, DW 4Q05, SLES9 SP3)*
  - Guest LAN sniffer (tcpdump etc.) *(2.6.15, DW 1Q06, SLES10)*

Neues CPU- und Prozessaccounting (1)

- **Linux Anwendungen wie top und ps erhalten die Informationen vom Linux-Kernel**
- **Linux Zeitverwaltung basiert auf regelmäßigem „tick“**
- **Tick-basierendes Accounting ist ungenau**
  - Ungenauigkeiten liegen im Design
    - Werte sind auf nicht virtualisierten System auch ungenau, aber meist ok
  - Systeme mit virtuellen CPUs (z/VM, VMware, Xen, etc) verstärken das Problem:
    - Die reale CPU verbringt einen Teil der Zeit mit anderen Gästen
    - Zeitscheiben basieren auf „echter Zeit“, üblicherweise 5-6 Ticks
    - Prozesse bekommen Prozessorzeit anderer Gäste angerechnet
    - Prozesse können die komplette Zeitscheibe verlieren
  - Bisher hatte Linux echte und virtuelle Zeit nicht unterschieden
  - Konzept der Involuntary Wait/Steal time ist unbekannt
- **Wie kann man das Problem lösen?**
  - Wenn Umbau, dann richtig und exakt
  - Nutzung des zSeries CPU Timers
    - CPU Timer läuft nur, wenn die virtuelle CPU auf einer realen CPU ausgeführt wird

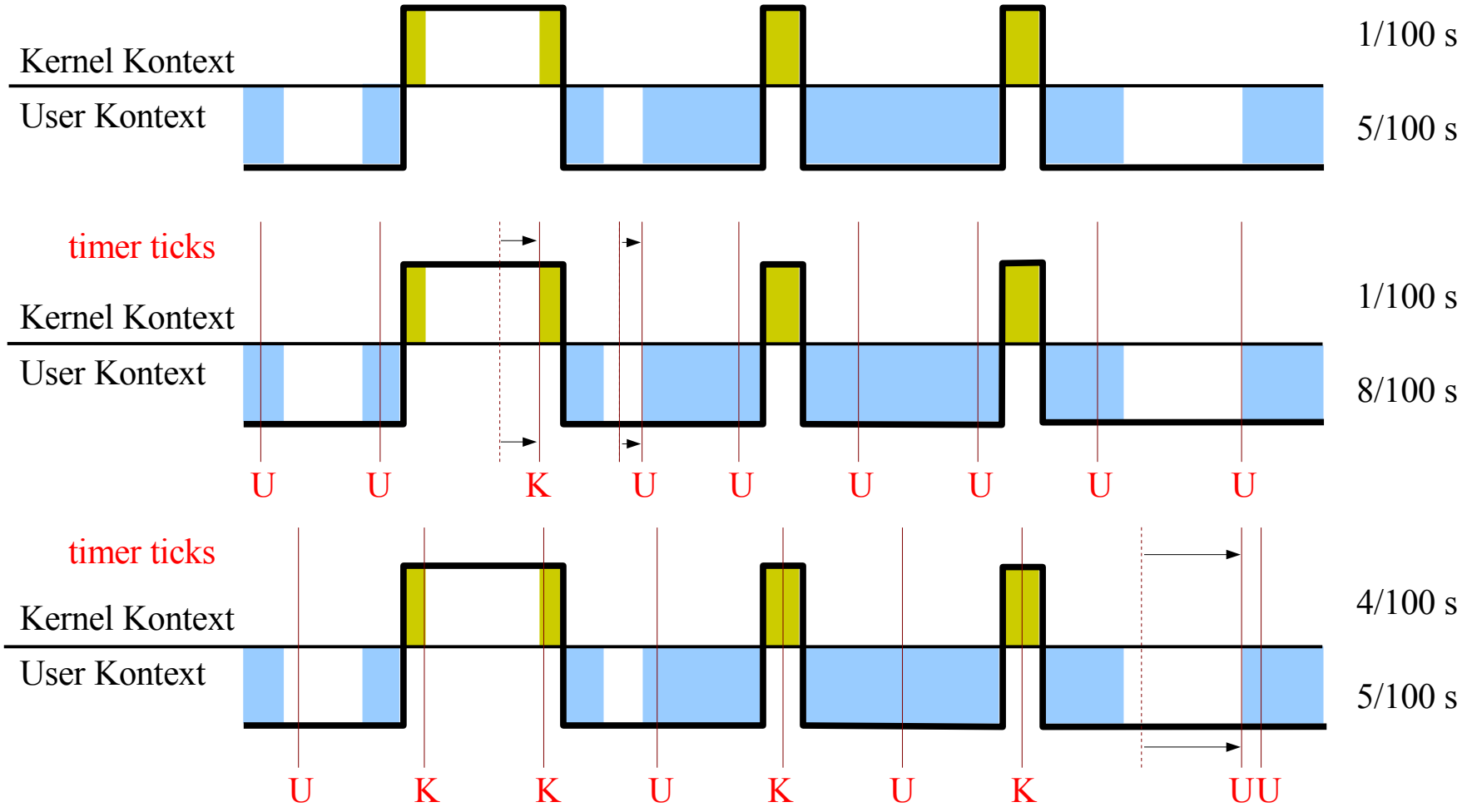
Neues CPU- und Prozessaccounting (2)

- **„Echte“ CPUs und Tick-Accounting**



Neues CPU- und Prozessaccounting (3)

- **Virtuelle CPUs und Tick-Accounting**

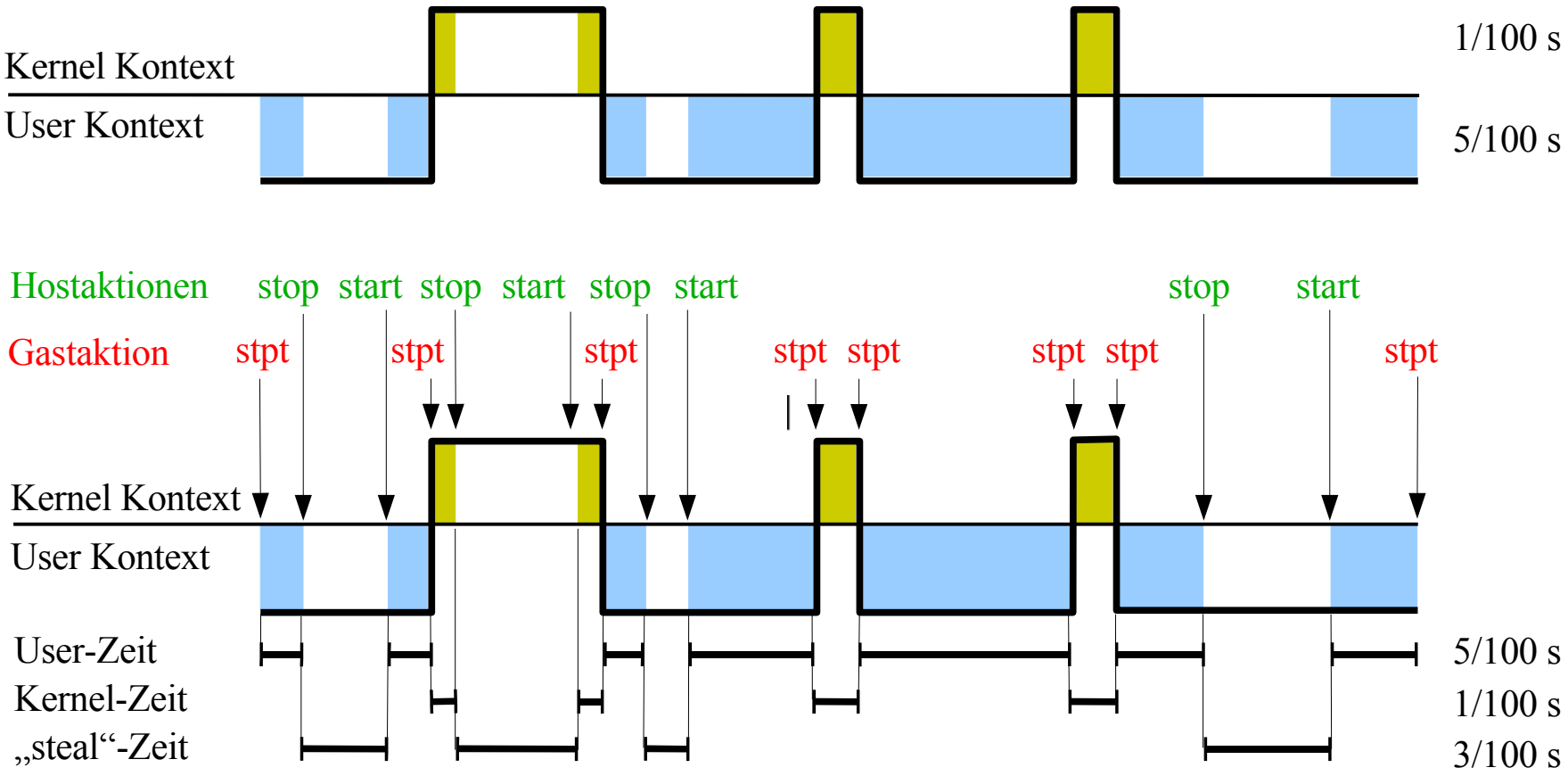


Neues CPU- und Prozessaccounting (4)

- **Der Wert der Zeit die von Linux berichtet wird hat sich geändert**
  - Vorher: Anteil an der virtuellen CPU
  - Neu: Anteil an der realen CPU
  - Neu: Zusätzliche Angabe der steal time – Anteil an der realen CPU der von anderen Gästen benutzt wird
- **Genauere Zeitwerte**
  - Interne Genauigkeit: 1 Mikrosekunde
  - Berechnung bei jedem Kontextwechsel
  - Umwandlung in HZ (1/100 Sekunde) bei Anzeige
- **Alle Linux Programme zeigen die genauen Werte an**
  - Bis auf steal time. Alte top Versionen addieren steal auf idle
- **Tools, welche „fehlerhafte“ Linuxwerte korrigieren rechnen jetzt falsch**
  - Man muss zwischen “guten” und “schlechten” Linuxen unterscheiden

Neues CPU- und Prozessaccounting (5)

- **Genaues Accounting mittels STPT Instruktion (store cpu timer)**



Neues CPU- und Prozessaccounting (6)

- **Änderungen in top**
  - rot – neue Semantik
  - blau – neuer Wert

```
top - 09:50:20 up 11 min,  3 users,  load average: 8.94, 7.17, 3.82
Tasks:  78 total,   8 running,  70 sleeping,   0 stopped,   0 zombie
 Cpu0 : 38.7%us,  4.2%sy,  0.0%ni,  0.0%id,  2.4%wa,  1.8%hi,  0.0%si, 53.0%st
 Cpu1 : 38.5%us,  0.6%sy,  0.0%ni,  5.1%id,  1.3%wa,  1.9%hi,  0.0%si, 52.6%st
 Cpu2 : 54.0%us,  0.6%sy,  0.0%ni,  0.6%id,  4.9%wa,  1.2%hi,  0.0%si, 38.7%st
 Cpu3 : 49.1%us,  0.6%sy,  0.0%ni,  1.2%id,  0.0%wa,  0.0%hi,  0.0%si, 49.1%st
 Cpu4 : 35.9%us,  1.2%sy,  0.0%ni, 15.0%id,  0.6%wa,  1.8%hi,  0.0%si, 45.5%st
 Cpu5 : 43.0%us,  2.1%sy,  0.7%ni,  0.0%id,  4.2%wa,  1.4%hi,  0.0%si, 48.6%st
Mem:    251832k total,   155448k used,    96384k free,     1212k buffers
Swap:   524248k total,    17716k used,   506532k free,    18096k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20629 root      25   0 30572  27m 7076 R 55.2 11.1   0:02.14 cc1
20617 root      25   0 40600  37m 7076 R 47.0 15.1   0:03.04 cc1
20635 root      24   0 26356  20m 7076 R 42.3  8.4   0:00.75 cc1
20638 root      25   0 23196  17m 7076 R 27.0  7.2   0:00.46 cc1
20642 root      25   0 15028 9824 7076 R 18.2  3.9   0:00.31 cc1
20644 root      20   0 14852 9648 7076 R 17.0  3.8   0:00.29 cc1
   26 root       5 -10     0    0    0 S  0.6  0.0   0:00.03 kblockd/5
  915 root      16   0  3012  884 2788 R  0.6  0.4   0:02.33 top
    1 root      16   0  2020  284 1844 S  0.0  0.1   0:00.06 init
```

Neues CPU- und Prozessaccounting (7)

- **Änderungen in /proc/stat**
  - rot – neue Semantik
  - blau – neuer Wert

```
# cat /proc/stat
  cpu  212314 0 31246 74377 4152 79 535 1900
  cpu0 107657 0 15727 35701 1967 38 267 955
  cpu1 104656 0 15518 38675 2185 40 267 944
  intr 317360 280140 37220
  ctxt 346461
  btime 1141129302
  processes 69331
  procs_running 1
  procs_blocked 0

  cpu <user> <nice> <system> <idle> <iowait> <irq> <softirq> <steal>
  intr <total number of interrupts> <ext.interrupts> <i/o interrupts>
  ctxt <number of context switches>
  btime <boot time in seconds since the Unix epoch>
  processes <number of processes created>
  procs_running <number of processes currently running>
  procs_blocked <number of processes currently blocked>
```

Reduzierung des z/VM Overheads (1)

- **QDIO Pass-Through Stufe 2**
  - Reduziert den CP Overhead für zfcp und qeth
  - Nutzt z/VM 5.2 Funktion "QDIO Enhanced Buffer-State Management" (QEBSM) f+r OSA, HiperSockets und FCP auf z9-109, z990 und z890
  - Erlaubt Linux Gästen QDIO Operationen direkt auf dem Kanal auszuführen, ohne Intercept durch z/VM
  - Kein unnötiges Kopieren in z/VM (Shadow Queueing)
  - Linux QDIO erkennt, wenn Hardware und Softwareunterstützung vorhanden sind
  - Wichtig: z/VM APAR VM63838 (PTF UM31710) behebt kritische Probleme mit SLES10 auf zVM 5.2

Compiler – Allgemeine Funktionen

- **Allgemeine Verbesserungen der Optimierung**
  - SSA-Infrastruktur (GCC 4.0)
  - Optimierungen über Funktionsgrenzen (GCC 4.1)
- **Sprachen und Sprachfeatures**
  - Fortran 95 Unterstützung (GCC 4.0)
  - Decimal Floating Point Unterstützung (GCC 4.2)
- **Weitere Verbesserungen**
  - Stack Protector (GCC 4.1)
  - Eingebaute atomare Operationen (GCC 4.1)

Compiler – System z Funktionen

- **System z9 109 Prozessor Unterstützung (GCC 4.1)**
  - Ausnutzung der „e*xtended immediate facility*“
  - Auswahl mittels `-march=z9-109` / `-mtune=z9-109`
- **Unterstützung für 128-bit IEEE quad “long double” Datentyp (GCC 4.1)**
  - Größerer Exponent und Mantisse
  - Auswahl mittels `-mlong-double-128`
- **Stackoverflow Vermeidung/Erkennung für Linux Kernel (GCC 4.0)**
  - Zur Compilezeit: `-mwarn-framesize` / `-mwarn-dynamicstack`
  - Zur Laufzeit: `-mstack-size` / `-mstack-guard`
  - Reduzierung des Stackframegröße : `-mpacked-stack`
- **GCC Unterstützung für z/TPF (GCC 4.0/4.1)**
  - z/TPF nutzt Linux / GCC als Crossbuild-Entwicklungsplattform
  - Neues target: `s390x-ibm-tpf`

Compiler – System z Geschwindigkeit

- **System z Architektur**
  - Verbesserte Behandlung des condition codes (GCC 4.0)
  - Verbessertes Funktionsprolog und -epilog Scheduling (GCC 4.0)
  - Verbessertes Nutzung von Speicher-zu-Speicher Instruktionen (GCC 4.0)
  - Unterstützung von „Sibling Call“ (GCC 4.0)
  - Verbesserte Nutzung von String Instruktionen (SRST, MVST, ..) (GCC 4.1)
  - Besseres Verfolgen von Registern (r13, r6, ...) (GCC 4.1)
  - Ausnutzung der LOAD ZERO Instruktion (GCC 4.1)
  - ICM/STCM, BRCT, vararg enhancements (GCC 4.1)
- **Performanceverbesserung 8%**
  - Standard Integer Industriebenchmark
  - Vergleich GCC 3.4 und GCC 4.1 auf System z

SLES 10 (1)

- **Basiert auf Linux 2.6.16 mit allen 2.6.16-Funktionen**
- **DCSS Unterstützung für Execute-in-place und Swapping**
- **xDR Unterstützung (GDPS light für Linux). SLES integriert sich in GDPS und kann mit z/OS bezüglich Site-Failover kommunizieren**
- **Yast-Unterstützung für**
  - VDISK
  - Dump
- **root - Device auf**
  - iSCSI
  - Multipath-Geräten
- **Booten und Installation über zfcp CD/DVD**
- **Layer2 - Unterstützung in qeth**
- **Neues CPU- und Prozessaccounting**

SLES 10 (2)

- **DASD Erweiterungen:**
  - Eindeutige IDs in /sys/bus/ccw/x.x.xxxx/uid
  - Dasd DIAG ist 64bit-fähig (ab z/VM 5.1)
- **Änderungen der Infrastruktur seit SLES9**
  - Udev hat hotplug ersetzt
    - Bessere Skalierbarkeit, Startzeit
    - /dev ist dynamisch
  - YaST unterstützt „persistente“ Gerätenamen
    - Booten von SAN/zfcp-Geräten
  - NOVELL APPAmor
- **Was wird nicht mehr in SLES10 unterstützt?**
  - LD_ASSUME_KERNEL
  - IBM JFS
  - IP-Netzwerk über CTC, ESCON und IUCV

- **Siehe auch: (V14) Novell SLES10 'Your Linux is Ready'**

Zum Nachlesen

- **Device Drivers, Features, and Commands – SC33-8289-01:**
  `http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26cdd01.pdf`
- **How to use Execute-in-Place Technology with Linux on z/VM – SC33-8287-00:**
  `http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26che00.pdf`
- **Using the Dump Tools – SC33-8290-00:**
  `http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26cdt00.pdf`
- **How to setup / use multipathing on SLES:**
  `http://support.novell.com/techcenter/sdb/en/2005/04/sles_multipathing.html`

Fragen & Diskussion

Weiteres

Linux on System z Distributionen (Kernel 2.6)

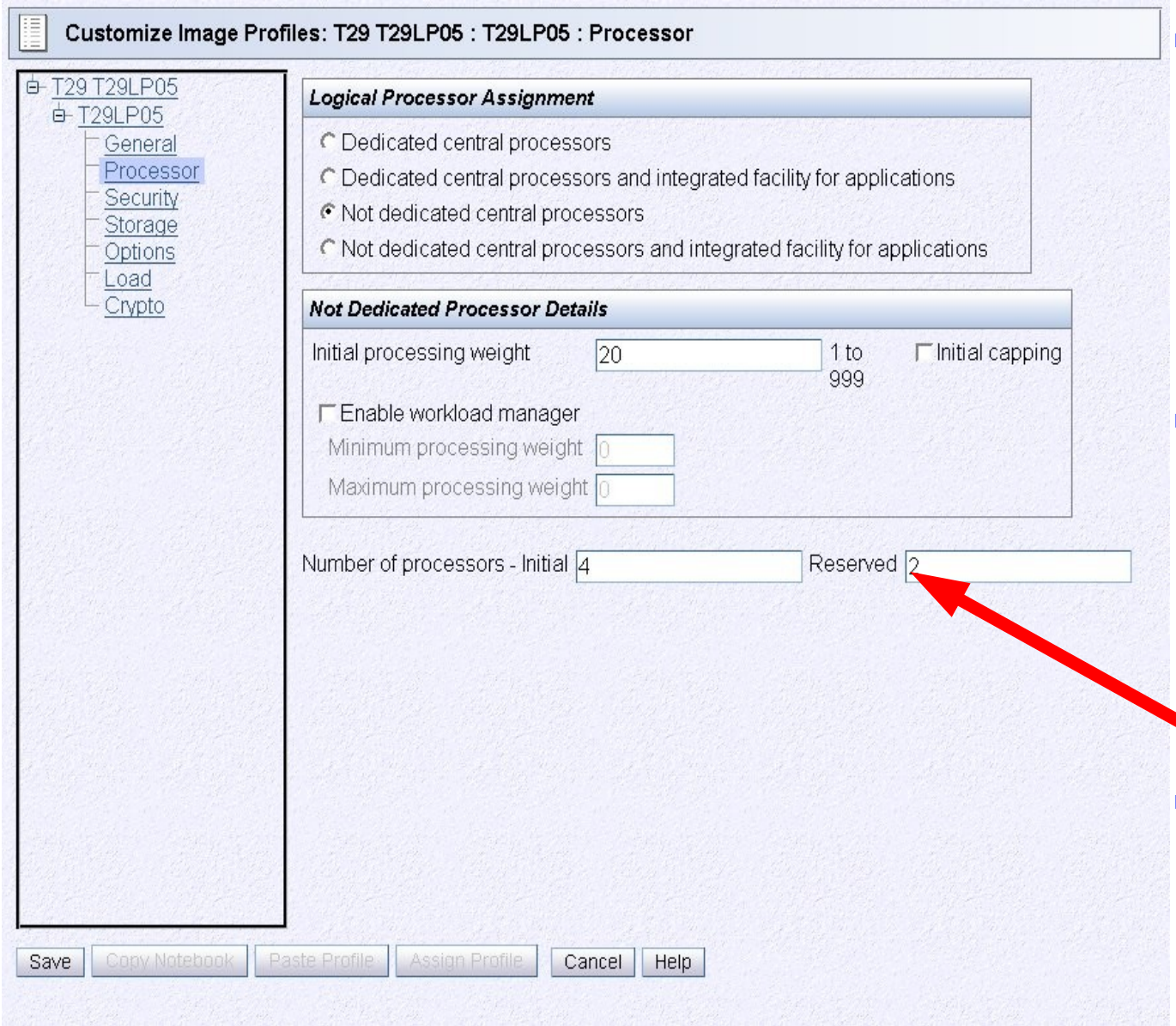
- **SUSE Linux Enterprise Server 9 (GA 08/2004)**
  - Kernel 2.6.5, GCC 3.3.3
  - Service Pack 3 (GA 12/2005)
- **SUSE Linux Enterprise Server 10 (GA 07/2006)**
  - Kernel 2.6.16, GCC 4.1.0
- **Red Hat Enterprise Linux AS 4 (GA 02/2005)**
  - Kernel 2.6.9, GCC 3.4.3
  - Update 4 (GA 07/2006)
- **Red Hat Enterprise Linux AS 5 (Betaphase)**
  - Kernel 2.6.x, GCC 4.1.x (t.b.d.)

- **Weitere**
  - Debian, Slackware, ...
  - Service und Support von Drittanbietern verfügbar

Neue Funktionalität (1)

- **z/VM Watchdog**
  - Seit z/VM 5.1 Unterstützung eines Watchdogs mittels Diagnose X'288'
  - vmwatchdog.ko Kernelmodul
  - Wenn ein Gast über einen definierten Zeitraum nicht mehr reagiert, wird ein vordefiniertes CP Kommando ausgeführt
    - z.B: Re-Ipl or Dump
  - Benötigt ein Programm, welches den Treiber bedient, z.b:
    - http://www.ibiblio.org/pub/Linux/system/daemons/watchdog/
    - Tivoli System Automation
  - Vmwatchdog unterstützt alle gängigen Watchdogs ioctls
  - Das Laden von DCSS-Segmenten kann viel Zeit beanspruchen, es sollte nicht zusammen mit vmwatchdog benutzt werde
- **Modulparameter**
  - cmd=<> : Welches Kommando soll z/VM ausführen?
  - nowayout=<0|1>: bei 1, kann vmwatchdog nicht mehr angehalten werden
  - conceal=<0|1>: ein- (1) und ausschalten (0, default) des “protected application environment” in dem Gäste vor unerwartetem CP READ geschützt werden

CPU hotplug (1)



- **CPU hotplug kann CPUs deaktivieren**
  - Laufende Prozesse werden auf andere CPUs migriert
  - Interrupts, Softirq etc. werden deaktiviert
  - CPU wird mit SIGP STOP angehalten
- **CPU hotplug kann CPUs aktivieren, welche im „Stopped“ Status sind**
  - CPU wird gestartet
  - Interrupts, Softirqs werden aktiviert
  - Prozesse werden zusätzlich auf die neue CPU verteilt
- **CPU hotplug kann keine „reserved CPUs“ aktivieren**

CPU hotplug (2)

- **/sys/devices/system/cpu/ enthält einen Ordner pro erkannter CPU**
- **Aktivieren und Deaktivieren ist einfach**
  - Deaktivieren: `echo 0 > /sys/devices/system/cpu/cpu<n>/online`
  - Aktivieren: `echo 1 > /sys/devices/system/cpu/cpu<n>/online`
- **In z/VM können zusätzliche CPUs online definiert werden**
- **Hinzufügen weitere CPUs in Linux erfordert Vorbereitung**
  - Der Kernel Parameter additional_cpus reserviert Kontrollstrukturen für zukünftige CPUs, z.B. "`additional_cpus=4`" in der parameters-Zeile der zipl.conf (+zipl und Reboot)
- **Beispiel: Hinzufügen einer 2ten CPU**
  - additional_cpus Parameter muss aktiv sein
  - `# modprobe vmcp`
  - `# vmcp define cpu 1`
  - `CPU 01 defined`
  - `# echo 1 > /sys/devices/system/cpu/cpu1/online`
- **Beispiel: zweite CPU eines Linux Gastes entfernen**
  - `# echo 0 > /sys/devices/system/cpu/cpu1/online`
  - Achtung: die CPU darf nicht in z/VM detached werden, CP würde des Gast resetten

Reduzierung des z/VM Overheads (2)

- **Verringerung der Spinlockkosten**
  - Linux auf Mehrprozessorsystemen nutzt Spinlocks für in-Kernel Synchronisation in kurzen kritischen Regionen
  - Spinlocks sind eine Form des Aktiven Wartens und belegen die CPU
    - “schlecht” für virtualisierte Systeme
  - System z und zSeries bietet Diagnose X'44' um die aktuelle Zeitscheibe aufzugeben
    - Alte Kernelversionen: Diagnose X'44' nach jedem Schleifendurchlauf
  - DIAG X'44' ist keine kostenlose Operation (SIE break, z/VM Code)
  - Deshalb: neue Logik zur Performanceverbesserung
    - Normales aktives Warten
    - Aufruf von Diagnose X'44' nach x nicht erfolgreichen Wartevorgängen
    - Diagnose X'44'-Rate sinkt, bessere Gesamtperformance
  - Voreinstellung ist 1000maliges Prüfen
  - Wert ist konfigurierbar mit /proc/sys/kernel/spin_retry
    - Deaktivieren: `echo 0 > /proc/sys/kernel/spin_retry`
    - Standard: `echo 1000 > /proc/sys/kernel/spin_retry`

Ausnutzung von DCSS

- **Verbesserte Fehlermeldungen**

```
# echo NE > /sys/devices/dcssblk/add

SEGMENT NE NOT LOADED RC=-2
```

vs.

```
extmem warning:segment_type: diag returned error 44
dcssblk warning: cannot load/query segment NE, does not exist
```

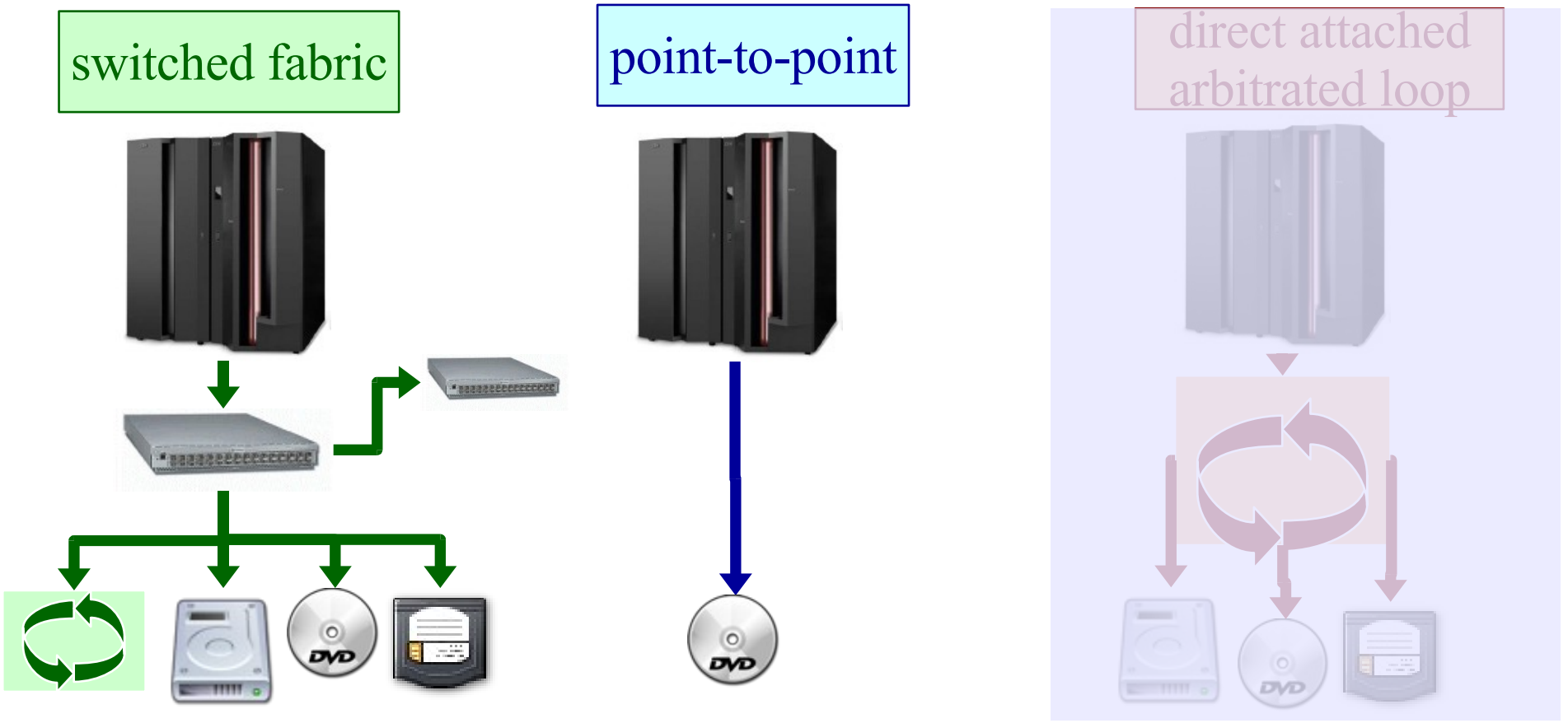
- **Unterstüzung für mixed mode DCSS Segmente**
  - Swap auf DCSS
  - DCSS mit einer Seite EW und den Rest EN
    - z/VM hält eine Seite in SPOOL, alle anderen Seiten in der DPA oder im Pagingspace
  - Keine Kanalprogramme, nur z/VM Speicherverwaltung
  - Geringe SIE break-Rate
- **Execute-in-Place wurde in ext2 integriert**
  - Keine externen xip2 Patches notwendig
  - Nutzung auch im Embedded Linux: mehr Tester

Reduzierung des z/VM Overheads (3)

- **Dasd Diagnose Treiber ist 64bit-fähig**
  - DASD diag ist alternative Methode zum Zugriff auf Platten mit konstanter Blockgröße, z.B. vdisks
  - 31bit Version seit jeher verfügbar
    - Lief nur im 31 bit Addressing Modus
    - Maximal 32 bit Blocknummer
  - Neue z/VM Versionen (ab.5.1) unterstützen im Diagnose X'250' Interface 64 bit Addressing und Blocknummern
  - Verringert des z/VM Overhead im Vergleich zum FBA Treiber

FCP Topologie

- **zFCP unterstützte bisher “switched fabric“**
- **Punkt-zu-Punkt/point-to-point Verbindungen sind jetzt auch unterstützt**
- **Keine Unterstützung für “direct attached arbitrated loop“**



Neue Funktionalität (2)

- **FCP: N-Port-ID Virtualisierung**
  - FCP Adapter nutzen einen “world wide port name” (WWPN) um sich im Netz zu identifizieren (ähnlich einer MAC-Addresse)
  - Auf Maschinen vor z9 hat jeder Gast die gleiche WWPN benutzt
    - SAN-basierte Zugriffskontrolle war nicht möglich
  - NPIV ist neuer Industriestandard für multiple WWPNs auf einem physikalischen Port
  - Virtuelle WWPNs werden von der z9 Hardware zugewiesen
  - Der erste Switch muss NPIV beherrschen
  - Zugriffskontrolle jetzt mit normalen SAN-Methoden möglich

z/VM Integration - 64-bit VMDUMP Unterstützung

- **Vmconvert unterstützt 64-bit VMDUMP Format**
  - IBM Service benötigt oft einen Linux Dump zum Debuggen von Problemen
  - Klassische Variante: Linux DUMP-Disk
    - `zipl -d /dev/dasdxx` als Vorbereitung der Platte
    - Die CPUs müssen gestoppt werden und ein store status ist notwendig
    - Ipl der Dumpplatte; warten bis disabled wait
    - Ipl Linux
    - `zgetdump` zum Lesen des Dump
  - Was, wenn keine DASD als Dump-Platte vorbereitet wurde? vmdump!
    - Dump erzeugen: `#cp vmdump`
    - Ipl CMS: `#cp i cms`
    - Dump laden: `dumpload`
    - Dump übertragen, z.B. mit ftp (bin, locsite fix 80)
    - Vmconvert kann den VM-Dump in einen Linux-Dump umwandeln:
      `vmconvert -f <vmdump.datei> -o <linuxdump.datei>`