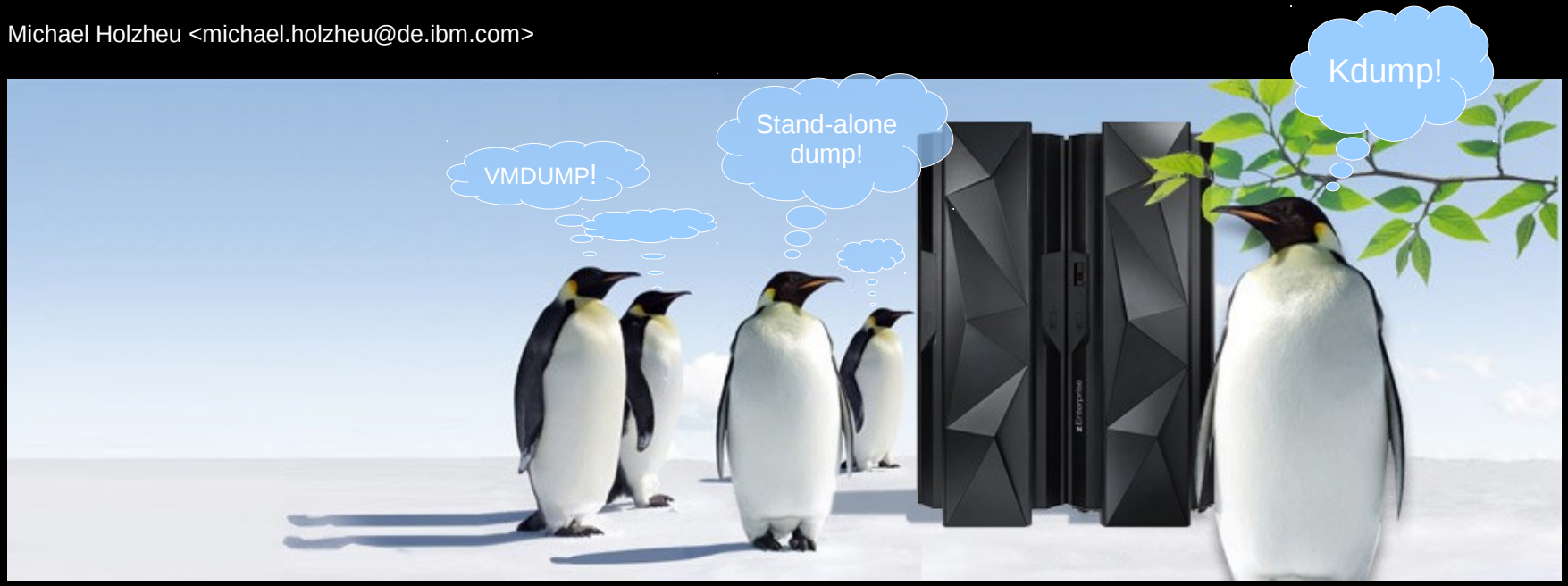
Kdump on the Mainframe

Michael Holzheu <michael.holzheu@de.ibm.com>



Trademarks & Disclaimer

AIX*
BladeCenter*
DataPower*
DB2*
FICON*
GDPS*
HiperSockets
IBM*
IBM eServer
IBM (logo)*
InfiniBand*
Parallel Sysplex*
POWER*
POWER7*
PowerVM
PR/SM
Smarter Planet
System x*
System z*
System z9*
System z10
WebSphere*
z9*
z10 BC
z10 EC
zEnterprise
zEC12
z/OS*
zSeries*
z/VM*
z/VSE

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs, and IFLs)

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at

www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

IBM

Contents

- Linux kernel dump history
- Traditional s390 Linux dump mechanisms
  - Stand-alone dump
  - VMDUMP
- Kdump on s390
- Kdump integration into the s390 dump environment

Before we start - Terms

- Mainframe
  - Big iron made by IBM
  - Long tradition (System/360 - 1964)
  - Strong RAS features
  - Other terms: System z, s390
- Linux on the mainframe
  - Since 1999 (2.2.13)
- Hypervisors: LPAR and z/VM
- Kernel dump
  - For kernel problems
  - Dump analysis tool “crash”

Linux kernel dump history

- 1999: Linux kernel crash dumps (LKCD)
- 2001: Linux on System z stand-alone dump
- 2002: Red Hat's Netdump
- 2004: Red Hat's Diskdump
- 2005: Kdump in Linux 2.6.13
- 2011: Kdump for Linux on System z

Traditional Linux on System z dump mechanisms

System z stand-alone dump: How it works

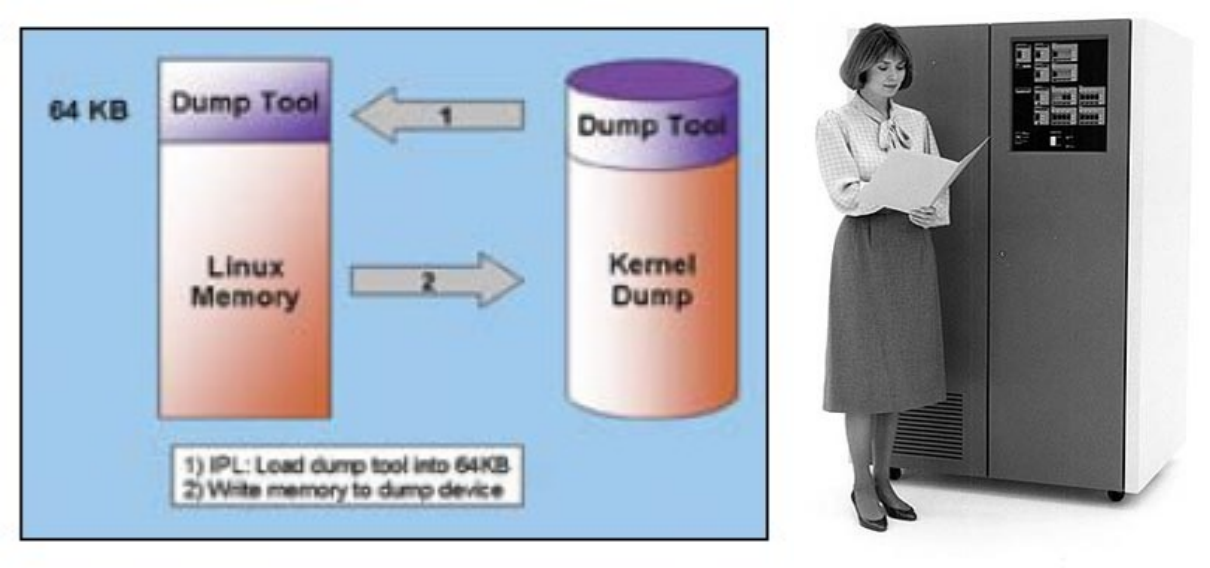
- Dump program is installed on dump device
- To trigger a dump the dump device is booted (IPLed)
  - Before dump program is loaded registers of boot CPU are stored
  - System resources survive boot process:
    - Memory
    - Register sets of non-boot CPUs
  - Dump program collects register sets of non-boot CPUs
  - Dump program writes dump do dump device
- Original OS is restarted and dump is copied from dump device
- Dump devices under Linux: DASD, Tape, and SCSI disks

Stand-alone dump: DASD and Tape

- DASD (also multi volume) or Tape cartridge prepared with small dump program written in assembler using CCWs

  $ zipl -d /dev/dasdc1

- Loaded into first 64 KiB (reserved by Linux on System z)
- Dump is written to dump device



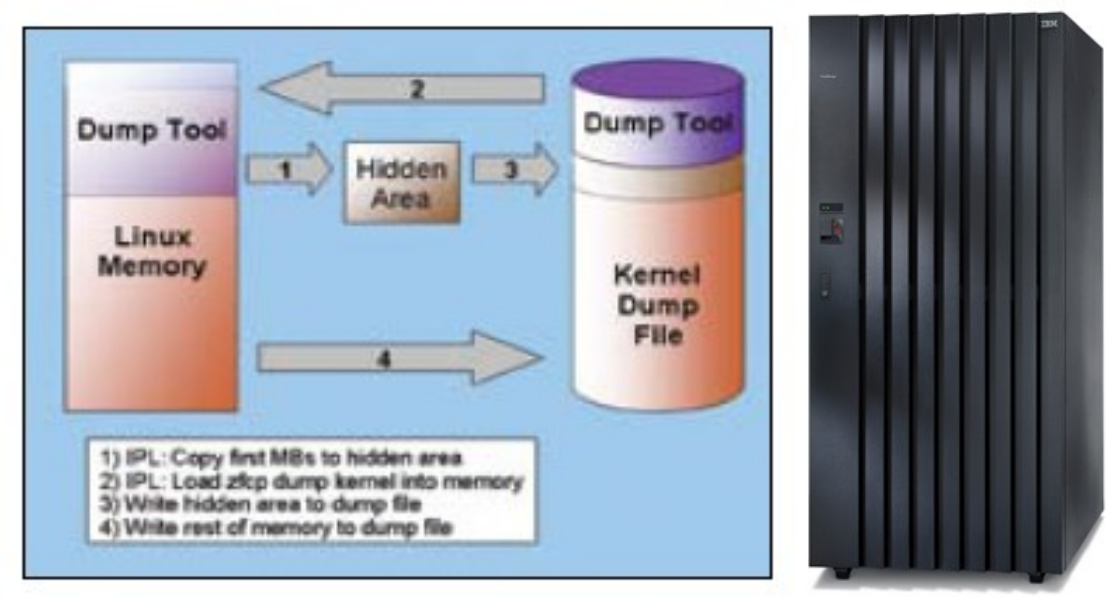
IBM DASD 3380 model CJ2 (1987)

Stand-alone dump: SCSI (zfcpdump)

- SCSI disk is prepared with Linux dump kernel and ramdisk

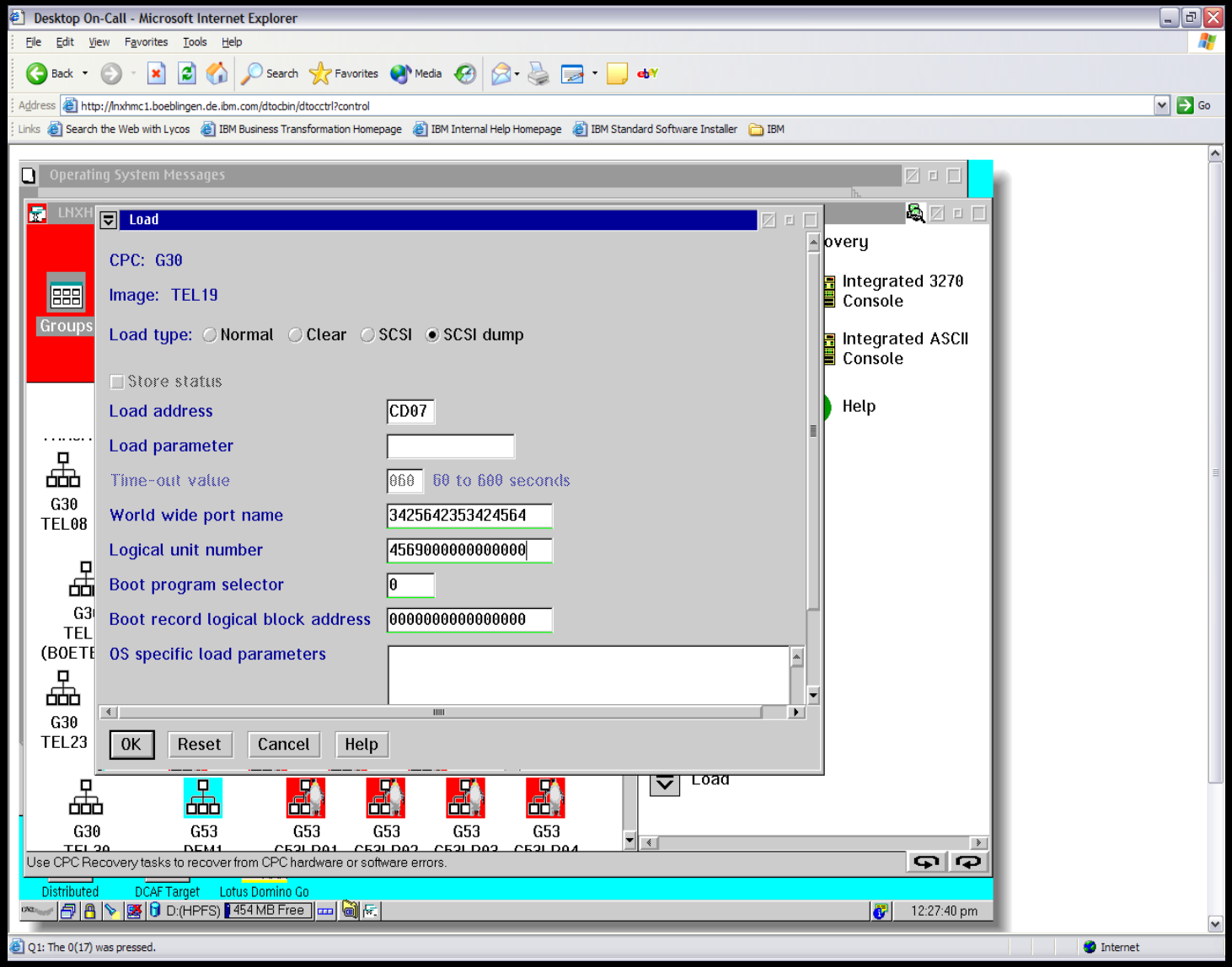
  $ zipl -D /dev/sda1

- At IPL time first part of memory and boot CPU registers are stored into data area provided by Hypervisor
- Linux dump tool reads saved memory from Hypervisor



IBM DS-8000

Trigger SCSI stand-alone dump via HMC IPL



Stand-alone dump: Accessing the dump

- Print information on dump

```
$ zgetdump -i /dev/dasdc1
General dump info:
  Dump created.......: Tue, 11 Sep 2012 08:18:14 +0200
  UTS node name......: r17lp31
  UTS kernel release.: 3.5.3-55.x.20120910-s390xdefault
  System arch........: s390x (64 bit)
  CPU count (real)...: 3

  Memory map:
    0000000000000000 - 00000000f7ffffff (3968 MB)
```

- Copy the dump

```
$ zgetdump /dev/dasdc1 > dump.s390
$ zgetdump /dev/ntibm0 -f elf > dump.elf
```

Stand-alone dump: Accessing the dump

- Mount the dump (also multi-volume)

```
$ zgetdump -m /dev/dasdc1 -f elf /mnt/
$ ls /mnt
dump.elf
```

- Compress dump with makedumpfile

```
$ makedumpfile -d 31 /mnt/dump.elf dump.filtered
```

- Start crash dump analysis tool on dump

```
$ crash vmlinux /dev/dasdc1
$ crash vmlinux dump.filtered
$ crash vmlinux /mnt/dump.elf
```

Linux on System z dump mechanisms: Hypervisor dump

- z/VM VMDUMP
- Hypervisor writes dump to SPOOL space that can be accessed by the Linux guest OS
- Dump is non-disruptive
- Linux guest OS can receive dump with *vmur* tool
- Example:
  - Trigger VMDUMP via hypervisor console: `#cp vmdump`
  - Reboot Linux (optional) and logon
  - Receive dump:

```
$ vmur list
ORIGINID FILE CLASS DATE  TIME     NAME   TYPE DIST
T6360025 0463  DMP  06/11 15:07:42 VMDUMP FILE T6360025

$ vmur rec -c 463 dump
```

Linux on System z dump mechanisms: Automatic dump

- The dumpconf service (init script)
- Stand-alone dump and VMDUMP can be configured
- /etc/sysconfig/dumpconf

```
ON_PANIC=dump_reipl
DUMP_TYPE=ccw
DEVICE=0.0.4e13
```

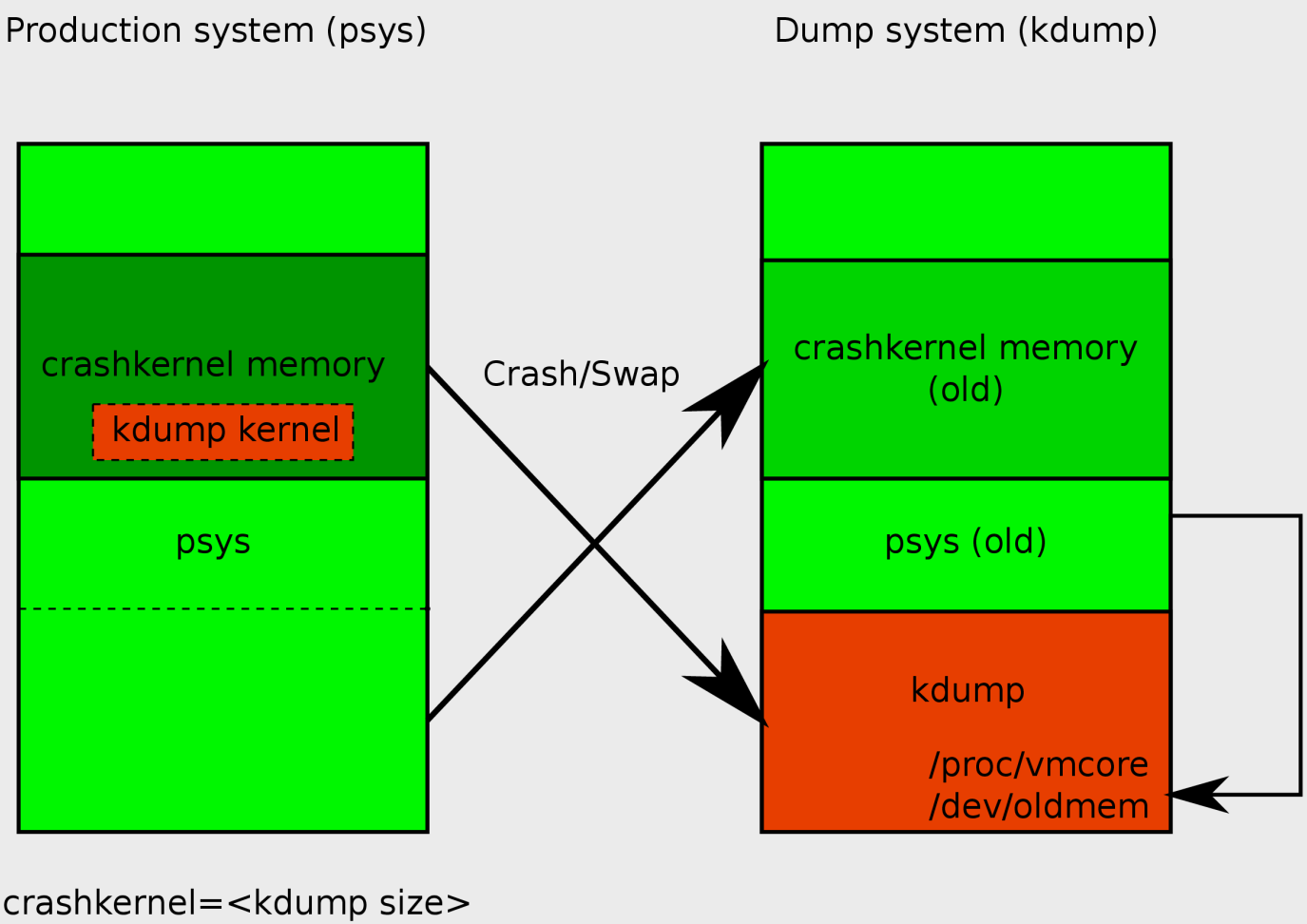
- System z Linux kernel panic code triggers IPL of stand-alone dump tool or VMDUMP

Advantages of traditional System z dump

- Trigger is almost 100% reliable (IPL and VMDUMP always works)
- No memory overlay of dump program and dump trigger code possible
- Different code (to the crashed one) writes dump (DASD, Tape and VMDUMP)
- Very little memory overhead
- Early and late kernel problems can be dumped
- Full device reset is done by IPL (no pending interrupts)

Kdump on System z

Kdump on System z: Overview



IBM

Kdump on System z: How to prepare?

- Reserve memory for kdump kernel with “crashkernel” parameter
  - Example: **`crashkernel=128M`**
- Boot production system
- Load kdump kernel into production system
  - Service kdump:

```
# service kdump start
```

  - Manual:

```
# kexec -p /boot/image \
    --command-line="$(cat /proc/cmdline | \
    sed -e 's/crashkernel=[^ ]*//')"
```

Kdump on System z: How to verify the setup?

- Is crashkernel memory defined?

```
$ grep Crash /proc/iomem
30000000-3fffffff : Crash kernel
```

- Is kdump kernel loaded?

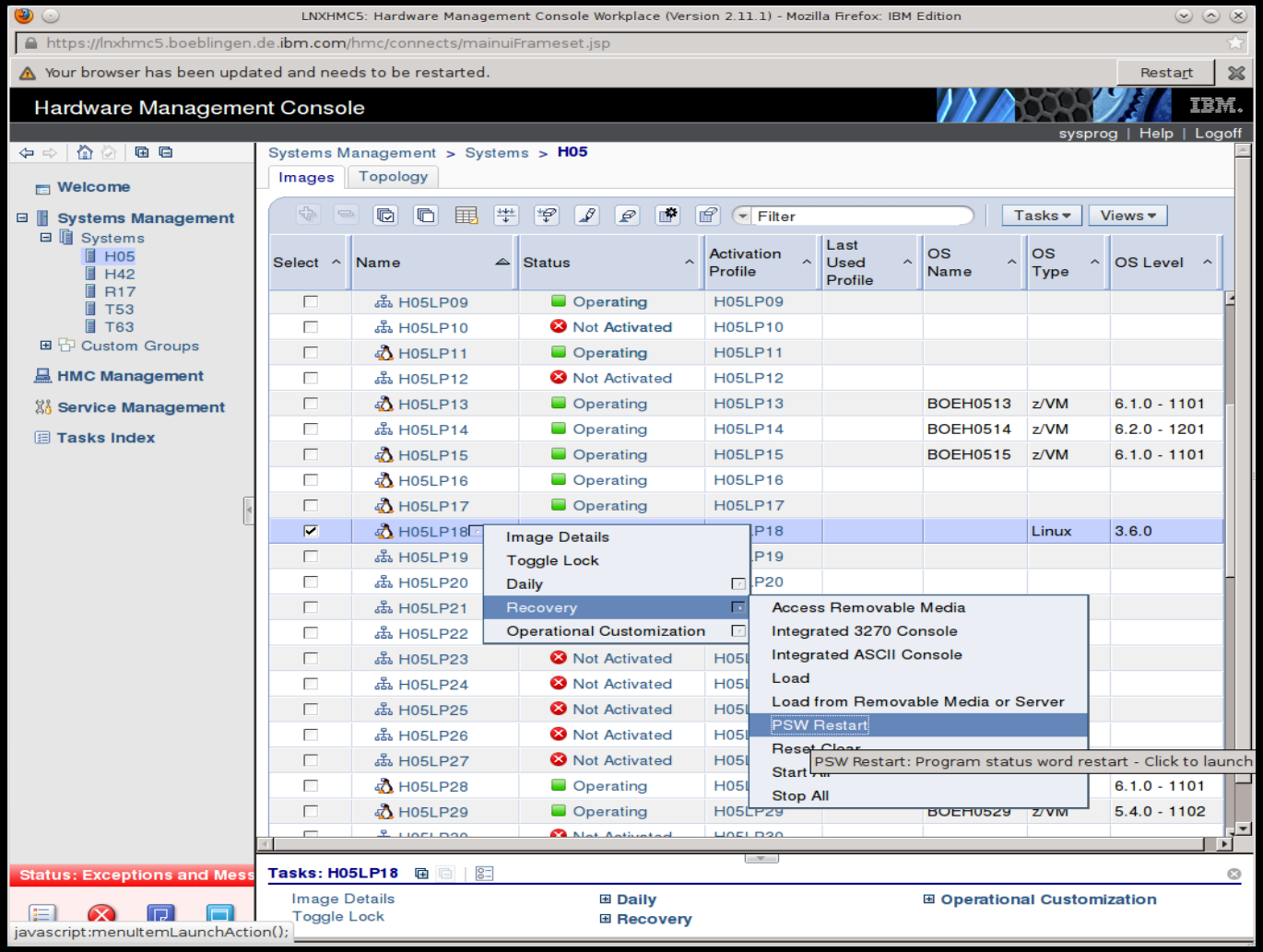
```
$ lsshut
Trigger           Action
=======================
Halt              stop
Restart           kdump,stop
Panic             kdump,stop

$ service kdump status
kdump is operational
```

IBM

Kdump on System z: How to trigger the dump?

- Kernel panic (automatically)
- PSW restart (manually)
  - **z/VM**: `#cp system restart`
  - **LPAR / HMC**: LPAR->Recovery->PSW Restart
- PSW restart (automatically with z/VM watchdog)
  - `$ modprobe vmwatchdog cmd="system restart" nowayout=1`
  - Start watchdog timer:
    `$ echo 1 > /dev/watchdog`
- Magic sysrq 'c'rash (manually - forced panic)
  - `"^-c"` on 3270 or HMC console
  - `$ echo c > /proc/sysrq-trigger`

Kdump on System z: PSW restart on HMC (LPAR)



Kdump on System z: Copy dump from /proc/vmcore

- Copy uncompressed to local / remote disk:

```
# cp /proc/vmcore /dumps
```

```
# scp /proc/vmcore user@host:mydumps/
```

- Copy compressed and filtered to local disk:

```
# makedumpfile -c -d 31 /proc/vmcore dump.kdump
```

- Copy compressed and filtered to remote disk:

```
# makedumpfile -F -c -d 31 /proc/vmcore | \
    ssh user@tuxmaker "cat > dump.kdump_flat"
```

- Run crash directly on /proc/vmcore

```
# crash vmlinux vmlinux.debug /proc/vmcore
```

- Normally the kdump service script copies /proc/vmcore

Kdump on System z: Reboot original system

- After /proc/vmcore has been processed, production system can be rebooted:

```
# reboot
```

- Normally the kdump service script does reboot automatically

IBM

Disadvantages of kdump

- Pre-loaded kdump kernel can be overlayed
- Kdump trigger code can be overlayed
- Kdump needs quite a lot of memory
- Early boot problems can't be dumped

IBM

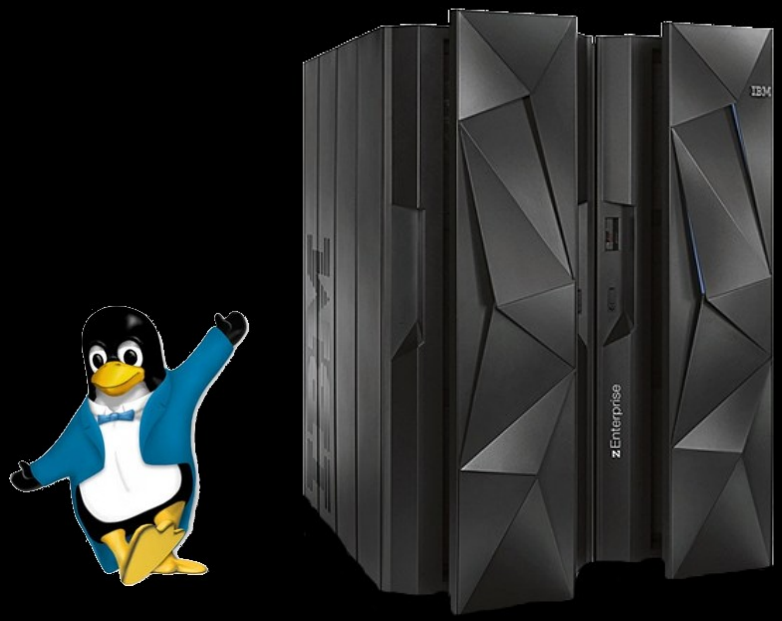
So why kdump on System z?

- Dump time and size can be reduced by page filtering with makedumpfile
- Dump disk space sharing is possible for server farms using network dump
- Dump setup is made easier using existing kdump setup GUIs of Linux distributions, e.g. system-config-kdump or yast
- The integration with the Linux on System z stand-alone dump tools ensures that the dump reliability with kdump can be almost as high as with the current solution

IBM

What is special for kdump on System z?

- On z/VM diagnose 10 is used to release the reserved crashkernel memory. Real/backed memory is required only for the kdump image and ramdisk (currently about 10 MiB). After some time z/VM will page out even this memory. Then no real memory will be wasted.
- On System z crashkernel memory is removed from the kernel page tables. Therefore the likelihood of memory corruption is reduced.
- On System z diagnose 308 is called before kdump is executed. That performs a CPU and I/O subsystem reset. So kdump on s390 is safe against old pending/ongoing I/O.
- No mem/cpu hotplug issues. Especially important because of cpuplugd.



Kdump integration into System z environment

IBM

Use stand-alone dump tools for kdump failure recovery (1/2)

- Kdump is still not 100% reliable
  - Pre-loaded kdump kernel / ramdisk can be overwritten by device DMA
  - Kdump trigger code (panic/PSW restart) might be not functional
  - Early boot problem cannot be dumped until kdump is loaded
  - Kdump system itself can have problems (e.g. not enough memory)
- Automatic kdump failure recovery:
  - Configure traditional System z dump on panic (dumpconf)
  - When it is detected that kdump is corrupt (via checksums), instead of kdump the System z shutdown actions for panic and PSW restart will be run and stand-alone dump is created
- Manual intervention:
  - If kdump failed, it is still possible to create a manual s390 stand-alone dump

IBM

Use stand-alone dump tools for kdump failure recovery (2/2)

- When kdump failed during kdump execution and afterwards a stand-alone dump is created, the resulting dump contains two system states:

```
# zgetdump -i /dev/dasdb1
zgetdump: The dump contains "kdump" and "production system"
          Access "production system" with "-s prod”
          Access "kdump" with "-s kdump"
          Send both dumps to your service organization
```

- Then copy both dumps for analysis:

```
# zgetdump /dev/dasdb1 -s kdump > dump.kdump.s390

# zgetdump /dev/dasdb1 -s prod > dump.prod.s390
```

- ... or mount dumps, for example:

```
# zgetdump -m /dev/dasdb1 -s prod /mnt
```

Summary

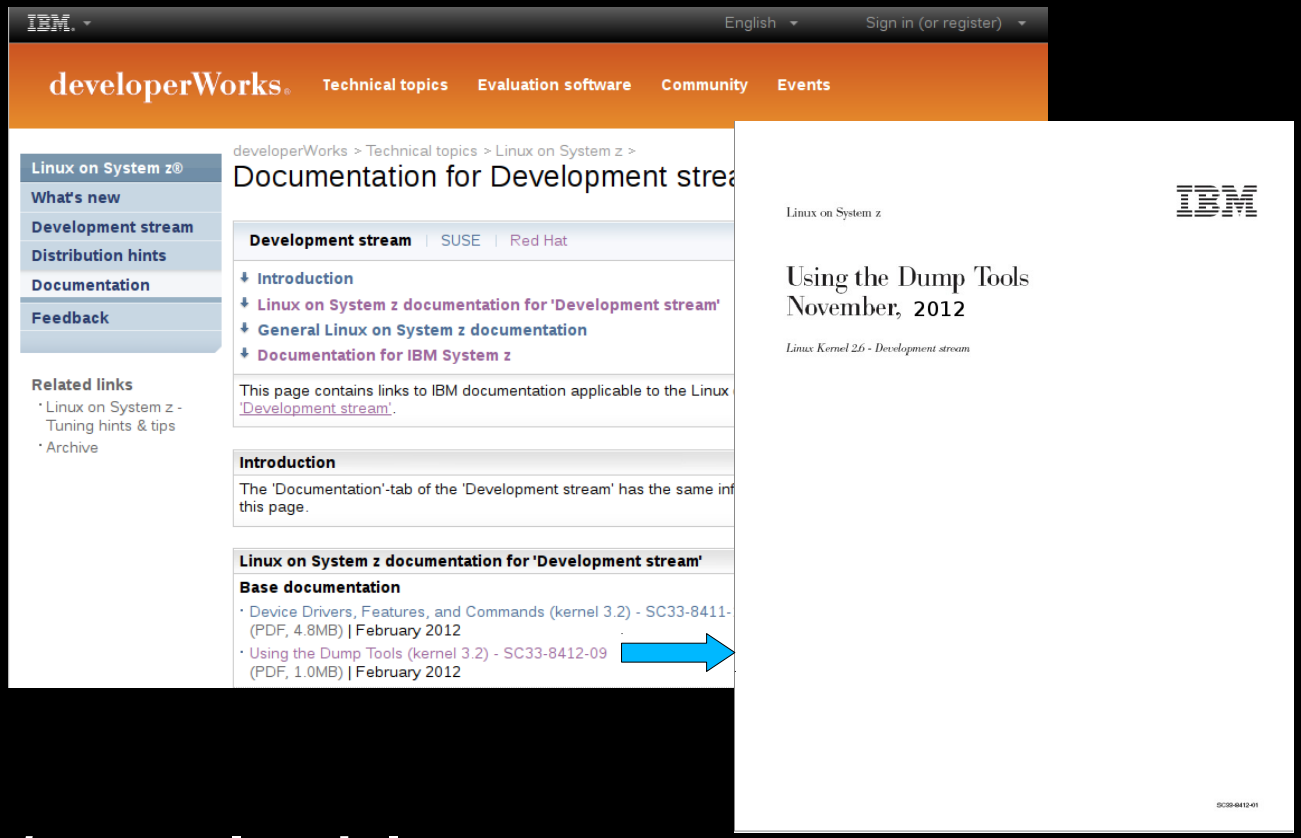
IBM

Get the best of both worlds

- Get great kdump features like dump filtering for System z
- Get reliable and resource friendly kdump implementation using System z features
- Still have stand-alone dump tools in the unlikely case that kdump fails, for example early crashes or kdump memory overlay

More Information

- **Using the dump tools book**
  http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html



- **z/Journal article**
  http://enterprisesystemsmedia.com/article/linux-on-system-z-kernel-dumps