Dr. Juergen Doelle
09 May 2014

IBM

Linux on System z -
Automatic CPU and memory resource management for z/VM Linux guests

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

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.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- SUSE is a registered trademark of Novell, Inc. in the United States and other countries.
- Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc.
  in the U.S. and other countries.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

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

Agenda

1. **Introduction**
   - **Objectives**
   - **cpuplugd.conf**
   - **Environment**
   - **Manual Sizing Results**
2. CPU Management
   - Known Issues
   - Setup
   - Results
3. Memory Management
   - Known Issues
   - Unplugging/Plugging Memory
   - Setup
   - Results
4. The Update Interval
5. Note!
6. Summary

Introduction

- **The issue**
  - Sizing of z/VM guests
    - Manually sizing can only be optimized specific for one certain workload type and level
    - Inadequate sizing might also impact other guests
- **The ideal solution**
  - Use a common sizing for all guests, reflecting the highest resource usage planned for the guests
  - Let the guest size itself according to the resources needed within a that range
- **The tool: cpuplugd daemon**
  - Important update: since SUSE Linux Enterprise Server (SLES) 11 SP2 and Red Hat Enterprise Linux (RHEL) 6.2 which greatly enhances the capability to define rules and the available performance parameters for the rule set.

Objectives

- **This session**
  - Compares cpu and memory management capabilities of the cpuplugd with a manually optimized setup
    - Show how close the automatic management can come to the manually optimized setup
  - Explains the mechanisms to define rules
  - Identifies the differences and capabilities of various rules
- **More information**
  - A detailed discussion and samples can be found in the Paper “Using the Linux cpuplugd Daemon to manage CPU and memory resources from z/VM Linux guests” at http://www.ibm.com/developerworks/linux/linux390/perf/tuning_cpuhotplug.html#cpuplugd
  - See also “Device Drivers, Features, and Commands” at http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html Chapter “cpuplugd - Control CPUs and memory”
  - manpages
    - “man cpuplugd.conf”
    - “man cpuplugd”

cpuplugd configuration

- **cpuplugd behavior is controlled by a configuration file**
  - Default configuration file is /etc/sysconfig/cpuplugd
- **Elements of the config file**
  - Pre-defined static variables (assigns a number), CPU_MIN="1"
  - Pre-defined dynamic variables (assigns a formula) CMM_INC="meminfo.MemFree / 40"
  - User-defined variables (number or a formula) pgscan_d="cpustat.system + …"
  - Rules
- **Variables can refer to**
  - predefined keywords, e.g. *user*, *system*, *idle*
  - CPU usage from /proc/stat and /proc/loadavg by specifying *cpustat.<name>*
    - **Note:** cpustat values are accumulated values since IPL from all CPUs!
  - memory usage from /proc/meminfo and /proc/vmstat data via *meminfo.<name>, vmstat.<name>*
  - historical data for variables from cpustat, meminfo, and vmstat by appending [<history level>] to the name: cpustat.system*[3]*
- **Rules**
  - HOTPLUG - used to enable CPUs
  - HOTUNPLUG - used to disable CPUs
  - MEMPLUG - used to increase the available memory
  - MEMUNPLUG - used to decrease the amount of memory

cpuplugd rules - Sample

- **Objective: add a CPU when less than 10% of one CPU remains unused**
- Define **variables** to calculate the used CPU ticks as difference from the current and the last interval
  → cpustat.<name> values are accumulative and the sum from all CPUs !

```
user_0="(cpustat.user[0]-cpustat.user[1])"
nice_0="(cpustat.nice[0]-cpustat.nice[1])"
system_0="(cpustat.system[0]-cpustat.system[1])"
```

- **Calculate** the average of the used CPU ticks and devide it by the total amount of ticks from all CPUs to get the ratio of the CPU load,
  → result value between '0' and '1', where '1' means all CPUs are used

```
CP_Active0="(user_0+nice_0+system_0)/(cpustat.total_ticks[0]-cpustat.total_ticks[1])"
```

- Do the same for a another interval in the **past**, e.g. two periods ago

```
user_2="(cpustat.user[2]-cpustat.user[3])"
nice_2="(cpustat.nice[2]-cpustat.nice[3])"
system_2="(cpustat.system[2]-cpustat.system[3])"
CP_Active2="(user_2+nice_2+system_2)/(cpustat.total_ticks[2]-cpustat.total_ticks[3])"
```

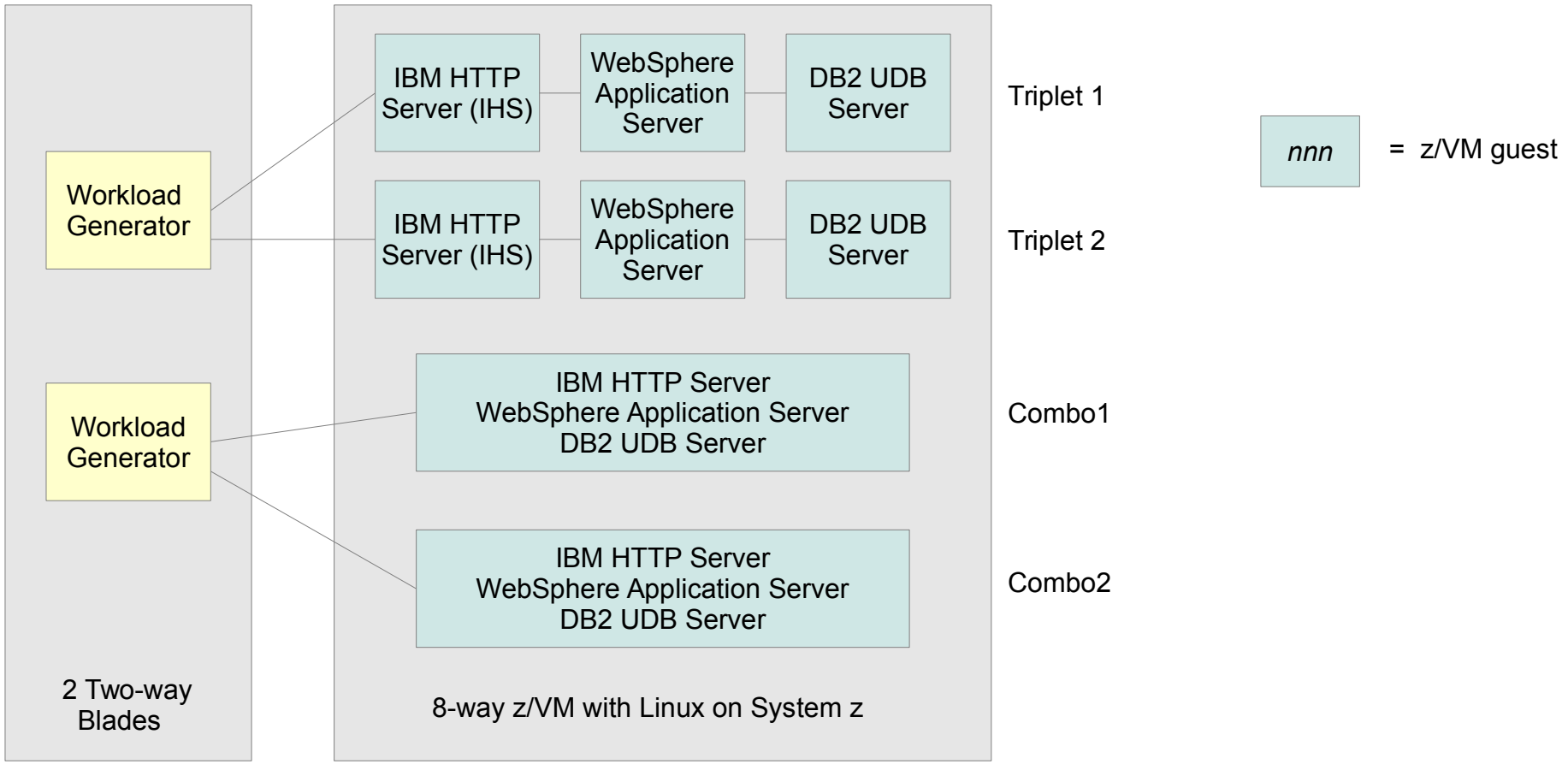
- Calculate the average of CPU load from these two intervals

```
CP_ActiveAVG="(CP_Active0+CP_Active2)/2"
```

- And now, the CPU plugging **rule:**
  → scale with the amount of CPUs to get the total load in units of CPUs

```
HOTPLUG="((1-CP_ActiveAVG)*onumcpus)<0.1"
```


The test environment



- **System utilization:**
  - Triplet 1 - low utilization — each system < 1 CPU
  - Triplet 2 - medium utilization — load on WebSphere Application Server > 1 CPU
  - Combo1- highly utilized system — load around 2 CPUs
  - Combo2- highly utilized system — load around 2 CPUs

The manually sized setup

Guest	configured CPUs	CPU LOAD [IFL]	Configured Memory [MB]	Allocated Memory [MB]
lnweb1	1	0.1	342	160
lnwas1	1	0.7	1,600	1,342
lnudb1	1	0.2	512	490
lnweb2	1	0.2	342	151
lnwas2	2	1.1	1,600	1,388
lnudb2	1	0.4	512	490
lncombo1	3	2.0	2,300	1,919
lncombo2	3	2.0	2,300	1,916
Total	13	6.8	9,508	7,857
virtual/physical	163%	85%	46%	38%
Overcommitment	yes	no	no	no

- **LPAR size was 8 CPUs and 20GB memory**
- **Allocated memory is the sum of resident pages in z/VM**
- **Only the virtual CPUs are overcommitted**
- **Sizing consideration:**
  - The combos have 3 virtual CPUs, the CPU load is in average 2 IFLs, there are peaks > 2 IFLs
  - lnwas2 uses slightly more than 1 IFL
  - All other systems are low utilized

Agenda

1. Introduction
   - Objectives
   - cpuplugd.conf
   - Environment
   - Manual Sizing Results
2. **CPU Management**
   - **Known Issues**
   - **Setup**
   - **Results**
3. Memory Management
   - Known Issues
   - Unplugging/Plugging Memory
   - Setup
   - Results
4. The Update Interval
5. Note!
6. Summary

CPU management - known issues and limitations

- **It might happen that a system managed by cpuplugd at high CPUs loads using FCP disks the database hangs**
  - Symptoms: all processes accessing files in /proc hang and can not be terminated by kill -9
  - Logging off the guest and restart resolves the hang situation
  - Concerns RHEL 6 and SLES11, related with CPU plugging
- **Fix**
  - fix is released for
    - SLES11 SP3 kernel level 3.0.93-0.8.2
    - SLES11 SP2 kernel level 3.0.93-0.5.1
    - RHEL 6.5 kernel level 2.6.32-393
- **The usage of cpuset and taskset is not compatible with the usage of cpuplugd!**
  - These tools could be used to bind processes/tasks to certain CPUs
  - cpuplugd is not aware of these sets

The cpuplugd setup for CPU management

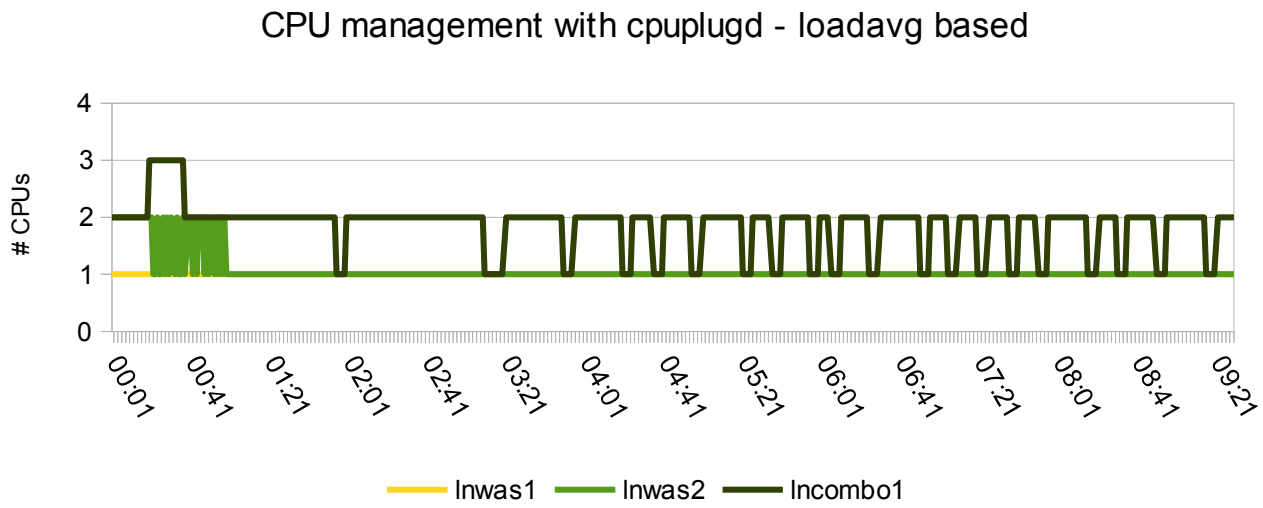
Manually sizing

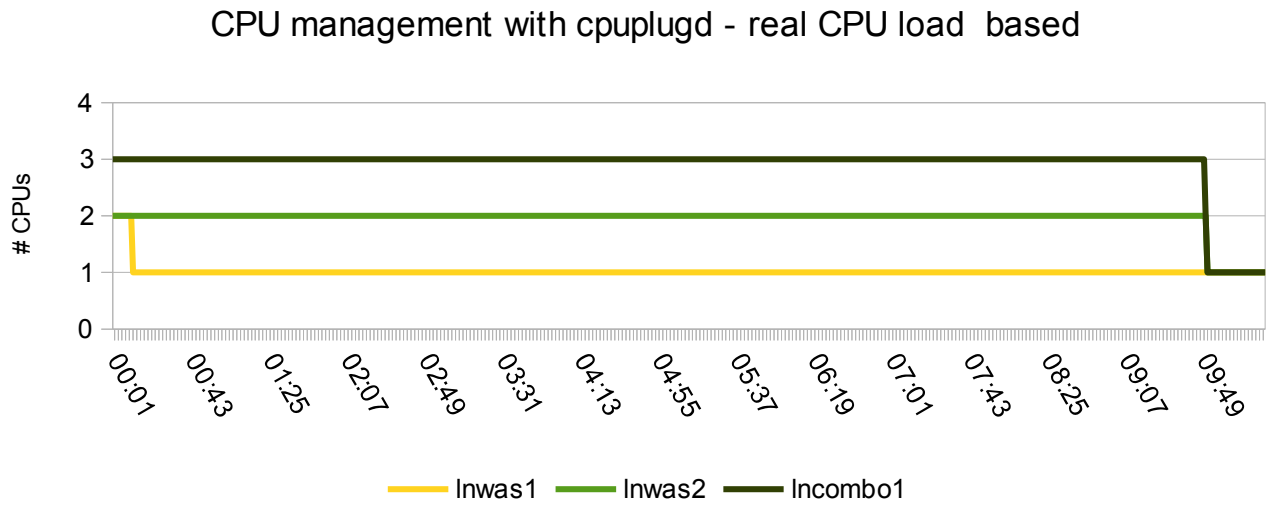
- **Adapt workload and guest size to maximize the throughput and minimize resources used**
- **The LPAR load is around 85% from 8 IFLs**

cpuplugd sizing

- **Size all guests with 4 virtual CPUs and same guest size as in the manually sized setup**
- **Use cpu plugging/unplugging rules to manage the active CPUs based on**
  - Parameter loadavg or
  - Real CPU load (user, system, nice, idle, iowait etc)

CPU management





- **Results**

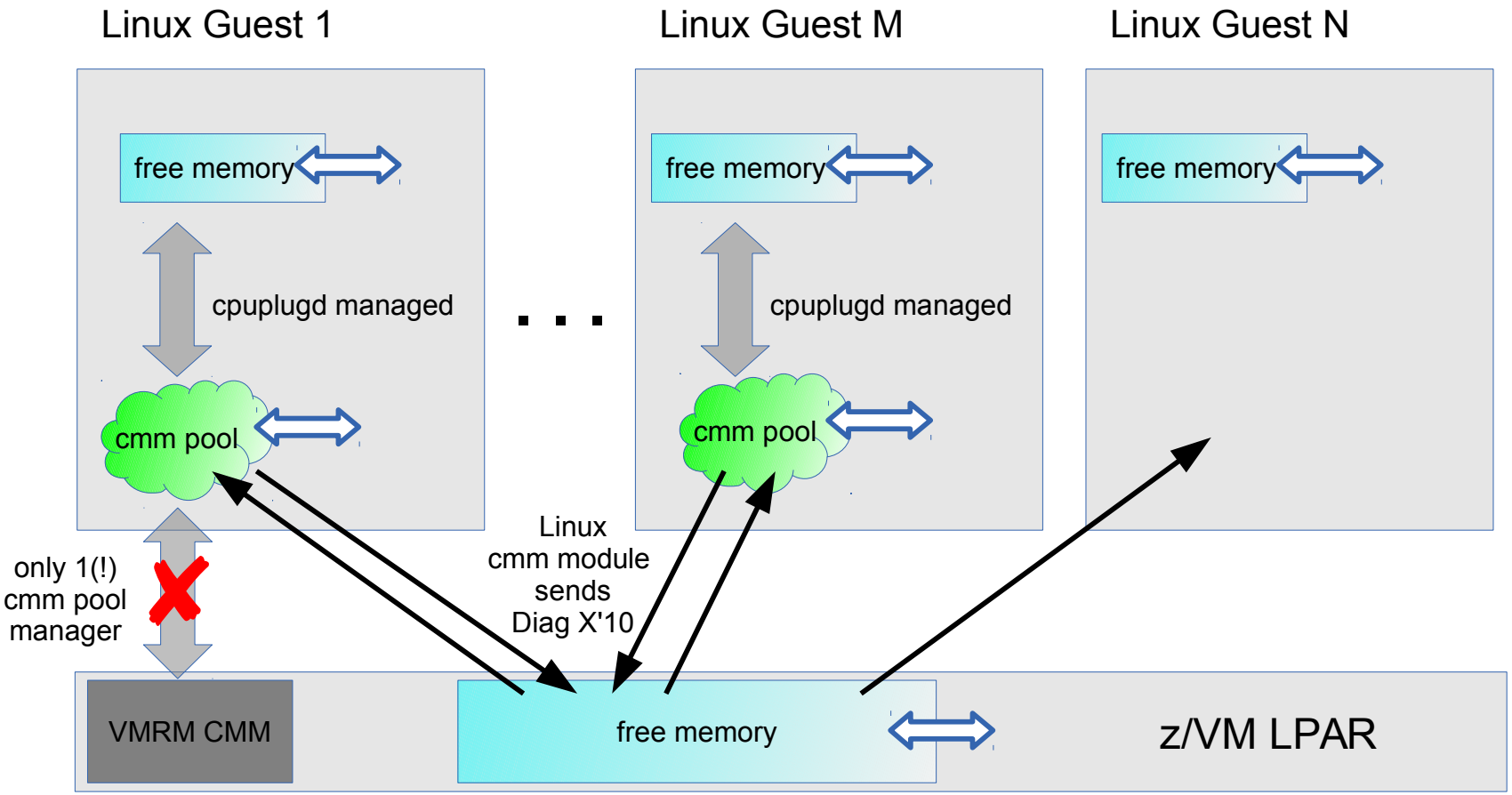
Rules	TPS*	Relative CPU load*
loadavg based	88%	84%
cpu load based	96%	96%
	higher is better	lower is better
	*100% is the manual-sized run	

- **Objectives are the key!**
- **CPU plugging rules based on loadavg provide**
  - A slower reacting system which adds CPUs restrictively
  - → Appropriate for high levels of CPU overcommitment
- **CPU plugging rules based on real cpuload provide**
  - A fast reacting system
  - → Appropriate for short response times

Agenda

1. Introduction
   - Objectives
   - cpuplugd.conf
   - Environment
   - Manual Sizing Results
2. CPU Management
   - Known Issues
   - Setup
   - Results
3. **Memory Management**
   - **Known Issues**
   - **Unplugging/Plugging Memory**
   - **Setup**
   - **Results**
4. The Update Interval
5. Note!
6. Summary

Memory management using cpuplugd



- The cpuplugd daemon puts memory pages into the 'balloon' (cmm pool)
- cmm module signals z/VM the availability of eligible pages frames via Diag X'10
- z/VM can easily take these page frames to serve memory requests from any guest
- Pages taken back from the balloon to Linux memory by cpuplugd daemon result in page fault at access, then a page frame is provided by z/VM

Memory management - known issues and limitations

- **CMM**
  - The Linux cmm module must be loaded, because it provides the memory pool
  - cpuplugd and VMRM-CMM are two independent managers for the same resources
    - **Do not manage a cpuplud managed system additionally with VMRM-CMM!**
- **For memory management with cpuplugd, APAR VM65060 is highly recommended.**
  - otherwise Linux experiences extreme high CPU steal values (up to 100%) when the cmm pool increases
  - Execution of Diag X'10' without APAR VM65060 blocks (all CPUs of) the virtual machine!

Linux memory management – unplugging memory

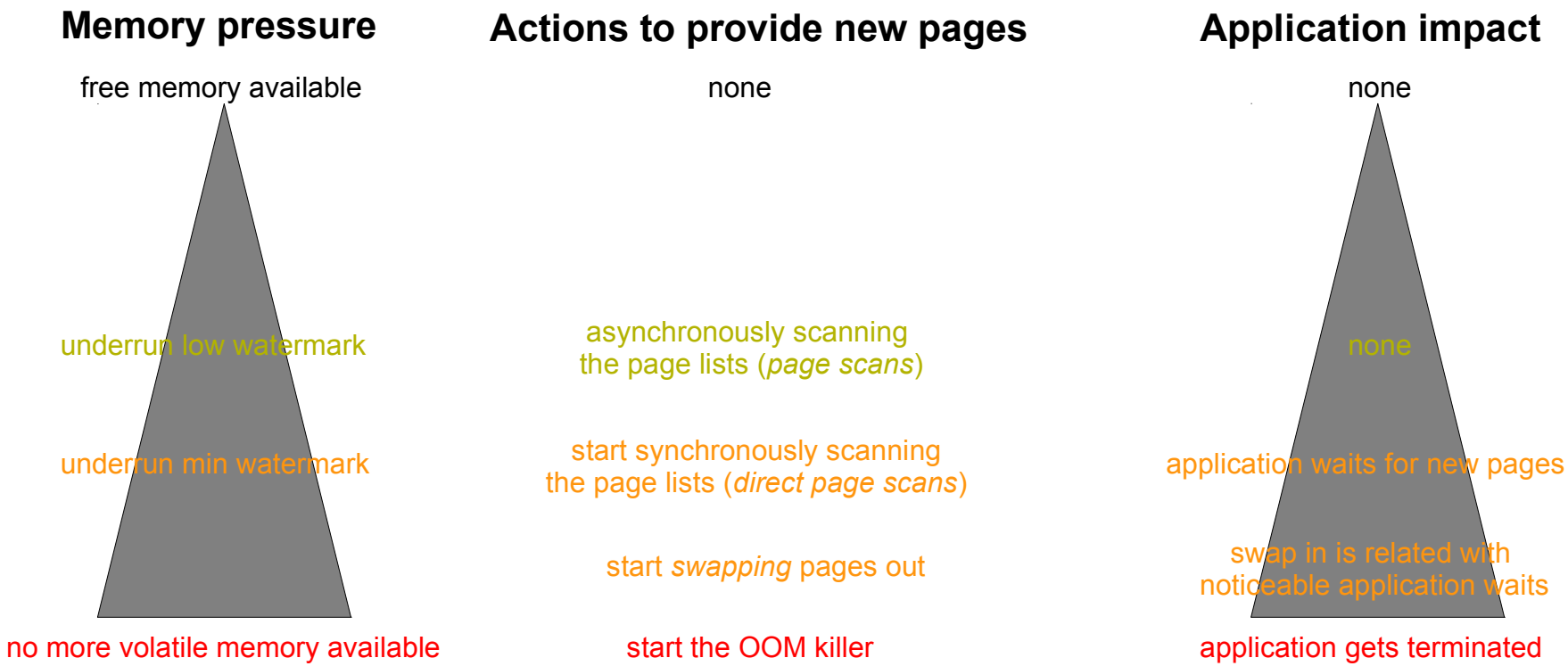
- **Objective: release free memory to z/VM, but what is free memory?**

Memory category	consider as free
Free memory	yes
Used memory	no
*Buffers**	no!
Cache, consist of	
- Page cache	often
*- Shared memory in 4KB pages***	no!
Large pages or *Huge pages (*pages > 4 KB, e.g. 1 MB) ***	no

* B*uffers* → used by the kernel, a shortage here may lead to unpredictable effects
** Shared memory→ very critical: used database buffer pools
*** Huge pages defined via vm.nr_hugepages count for the 'normal' Linux memory management always as used (even they are not really in use)

- **Page cache**
  - Calculate *page cache* as the difference between *cache* and *shared memory*: *meminfo.Cached – meminfo.Shmem*
  - Consider *Page Cache* as 'nearly' free, except for file system intensive workloads like TSM

Memory management mechanisms – plugging memory



- **With increasing level of memory pressure, the kernel mechanisms change**
  - from asynchronously scanning the page lists for freeable memory pages
  - to synchronously scans (called direct scans) and increasing swapping activities
- **With the occurrence of direct scans the application impact starts**
  - Latest when swap in occurs the application experience noticeable waits
- **The last possibility, when no memory can be found, is the out-out-memory killer (OOM) which terminates applications**

The cpuplugd setup for CPU + memory management

- **cpuplugd sizing**
  - Size all guests with 4 virtual CPUs and and 5 GB memory
  - Use cpu plugging/unplugging rules to manage the active CPUs based on real CPU load (user, system, nice, idle,iowait etc)
- **Common setup**
  - minimum cmm pool size: 0 (CMM_MIN="0")
  - maximum cmm pool size: the system size (5 GB) minus 256 MiB (CMM_MAX="1245184")
    → allows the cmm pool to grow to the maximal possible size.
  - Reserving 256 MiB for the kernel was intended as a safety net,
    → if our rules work well the size of the cmm pool should never approach the maximum value.
- **Use memory plugging/unplugging rules to manage the amount of available memory based on**
  - Normal page scan rates
  - Direct page scan rates
  - Various definitions for memory eligible for the cmm pool (free memory, page cache, etc)
- **Asymmetric CMM pool management!**
  - Decrement the pool fast when memory is required
  - Increment the pool slowly to avoid high frequent size changes

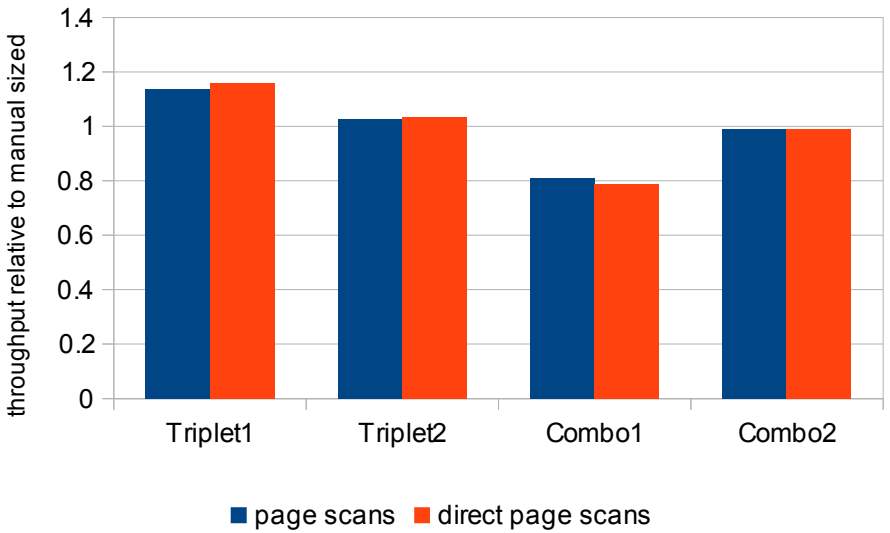
Memory plugging – identify suitable parameters

Plug memory (Increase), if		Unplug memory (shrink), if		Relative TPS*	Relative LPAR CPU load*	Guest size [MiB]*	
Parameter	Rate [pages/sec]	considered Memory type	% of total memory			Linux	z/VM
page scans	> 20	free (cache + buffers)	> 10% or > 50%	97%	99%	132%	115%
		free	> 10%	97%	98%	131%	107%
		free + page cache	> 10%	96%	99%	120%	114%
direct scans	> 20	**free + page cache**	**> 10%**	**96%**	**99%**	**109%**	**105%**
		free + page cache	> 5%	93%	98%	97%	99%
				higher is better	lower is better	closer to 100% is better	
				*100% is the manually sized run			

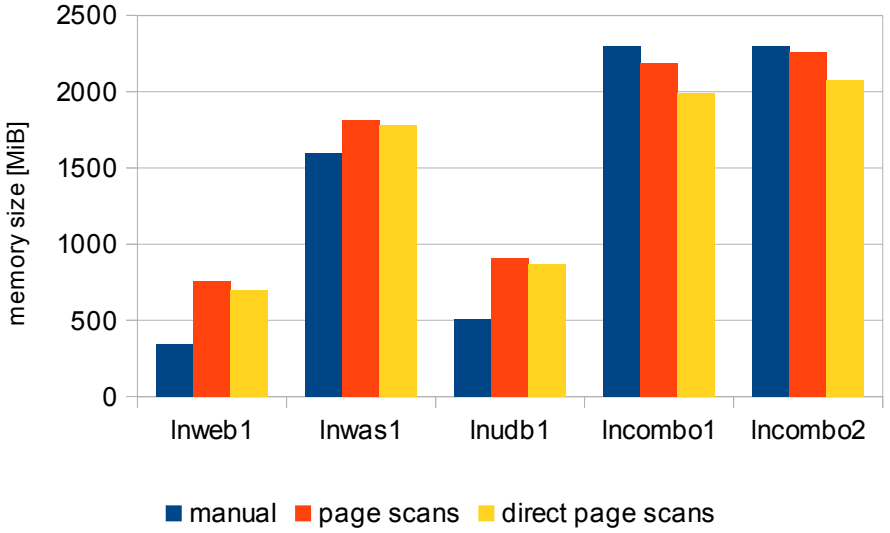
- **Optimization only possible with respect to one goal: throughput or memory size**
- **For unpluging memory**
  - free memory + page cache was a good indicator for free memory
- **For plugging memory**
  - Optimize for transactional Throughput
    - Use page scans as trigger to plug memory, allow 10% or more free memory
    - Leads to larger guests
  - Memory size
    - Use direct page scans as trigger to increase memory size
    - Throughput is only slightly lower than for rules with page scans
    - The lower the limit for free memory, the smaller gets the guest and the worse the throughput

Memory plugging – page scans vs direct page scans

Throughput, Linux managed via cpuplugd



Linux memory sizes managed via cpuplugd



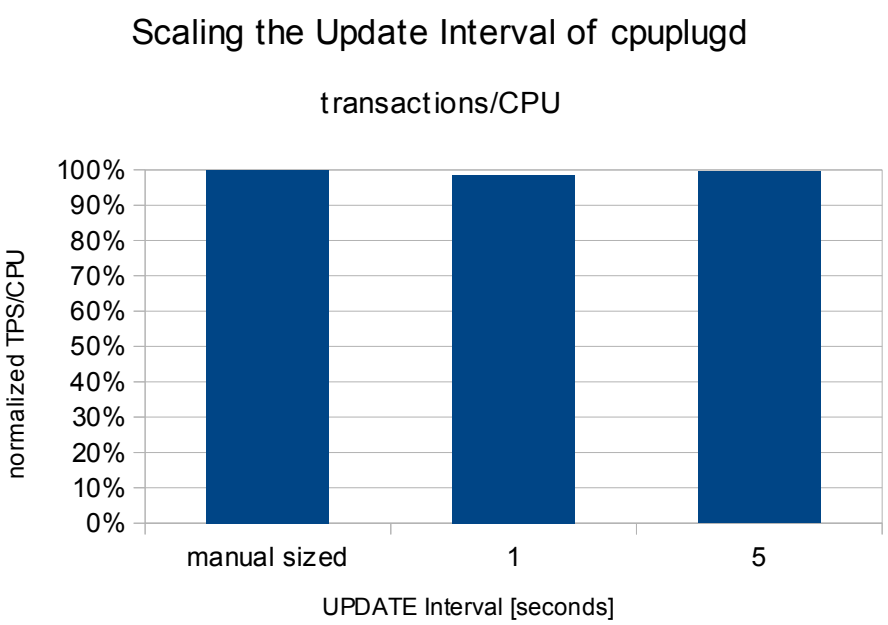
- **Throughput variation is moderate**
- **Guest size varies heavily, the impact of the rules is server type dependent**
  - WebServer and Database server: rules could be even more restrictive
  - WebSphere Application Server: direct scan based rule is good
  - Combo Systems: prefer page scan based rules (lncombo1 was swapping)
- **Consider a setup with server type specific rules, like**

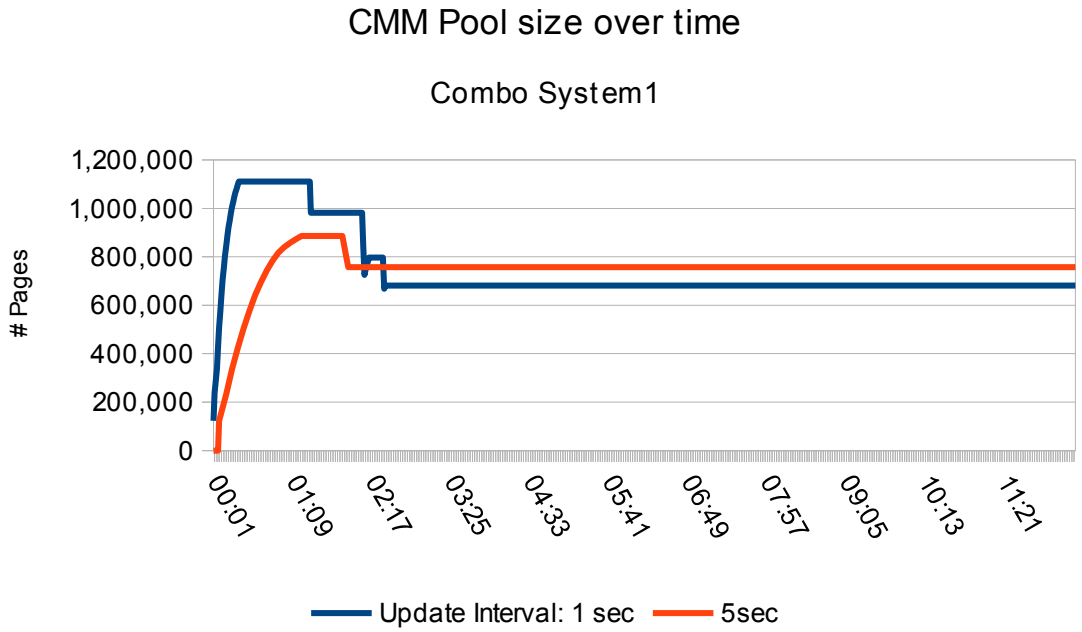
Server type	CMM_INC [pages]	CMM_DEC [pages]	Plug memory, when	Unplug memory, when
Web server	free memory /40	total memory/40	direct scans > 20	(free mem+page cache) > 5%
WebSphere Apps. Server	free memory /40	total memory/40	direct scans > 20	(free mem+page cache) > 5%
Database server	(free mem+page cache)/40	total memory/40	direct scans > 20	(free mem+page cache) > 5%
Combo	free memory /40	total memory/40	normal scans > 20	(free mem+page cache) > 10%

Agenda

1. Introduction
   - Objectives
   - cpuplugd.conf
   - Environment
   - Manual Sizing Results
2. CPU Management
   - Known Issues
   - Setup
   - Results
3. Memory Management
   - Known Issues
   - Unplugging/Plugging Memory
   - Setup
   - Results
4. **The Update Interval**
5. Note!
6. Summary

Update interval





- **Update interval was scaled from 1 – 5 seconds**
- **CPU cost per transaction is in the same order as the manual sized run (w/o cpuplugd) for all cases**
  - CPU cost of cpuplugd management is ignorable, when there are no high frequent changes
- **CMM pool behavior: the longer the update interval the smoother are the changes**
- **Small intervals provide a fast reacting, sensitive system**
- **Large intervals provide a slow reacting system**
  - How fast reactions are required?

Agenda

1. Introduction
   - Objectives
   - cpuplugd.conf
   - Environment
   - Manual Sizing Results
2. CPU Management
   - Known Issues
   - Setup
   - Results
3. Memory Management
   - Known Issues
   - Unplugging/Plugging Memory
   - Setup
   - Results
4. The Update Interval
5. **Note!**
6. **Summary**

Note!

- **Middleware typically determines the available resources on startup**
  - Amount of CPUs
  - Amount of memory
- **Performance relevant decisions are based on that**
  - Degree of parallelism
  - Buffer sizes
- **Recommendation: Start the servers first and cpuplugd later!**

- **More details to the Oracle database**
  - The database becomes aware when the amount of available CPUs is changing
    - The number of available CPUs is determined several times per second
  - To avoid throughput limitation, the plugging rule should always keeps more than one CPU free, for example: `HOTPLUG="((1 - CP_Active) * onumcpus) < 1.1"`
- **More details to the WebSphere Application Servers**
  - Many WebSphere Application Server workloads run better with 2 CPUs than with one.
  - Consider defining CPU_MIN=2 and stack multiple JVM

- **Treating page cache as free memory might be not appropriate for file system intensive workloads like TSM**

Summary

- **The new version of cpuplugd introduces a new systems management based on system resource usage.**
  - Start gathering experiences in a test environment and monitor the system to verify the impact of your rules.
  - Install APAR VM65060 and the upcoming Linux kernel updates
  - Do not manage a system with cpuplugd which is also managed by VMRM-CMM, or has cpuset or taskset defined
  - Plugging rules have a higher priority than the unplugging rules. That protects the system if the unplug rule is too aggressive.

- **The tested automated sizing results in a sizing nearly as good as manually sized, but adding the possibility to react on changes in load and resource requirements**
  - The manually optimized setup fits for only one workload scenario
  - The automated management can react on workload changes and adjust the used resources as needed

- **The cpuplugd simplifies the setup of z/VM guests significantly and provides accurate sizing according to the load of the Linux guest**

Recommendations

- **The presentation/paper suggests one set of rules which works well for all used server types**
  - The result can be improved further with middleware specific modifications
  - It seems that single server guests are easier to manage than mixed server guests
- **Be aware of middleware specific requirements, e.g. middleware may vary the amount of work scheduled depending on the amount of available CPUs**
- **Recommendation is to use 1 second as update interval**
  - If a slowly reacting system behavior is the target, build averages over larger interval
  - The overhead for evaluating the rules is very small even at the shortest Udate interval value

Questions ?

- **Further information is located at**
  - More details and samples in the Paper
    "Using the Linux cpuplugd Daemon to manage CPU and memory resources from z/VM Linux guests"
    http://www.ibm.com/developerworks/linux/linux390/perf/tuning_cpuhotplug.html#cpuplugd
  - Linux on System z – Tuning hints and tips
    http://www.ibm.com/developerworks/linux/linux390/perf/index.html
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

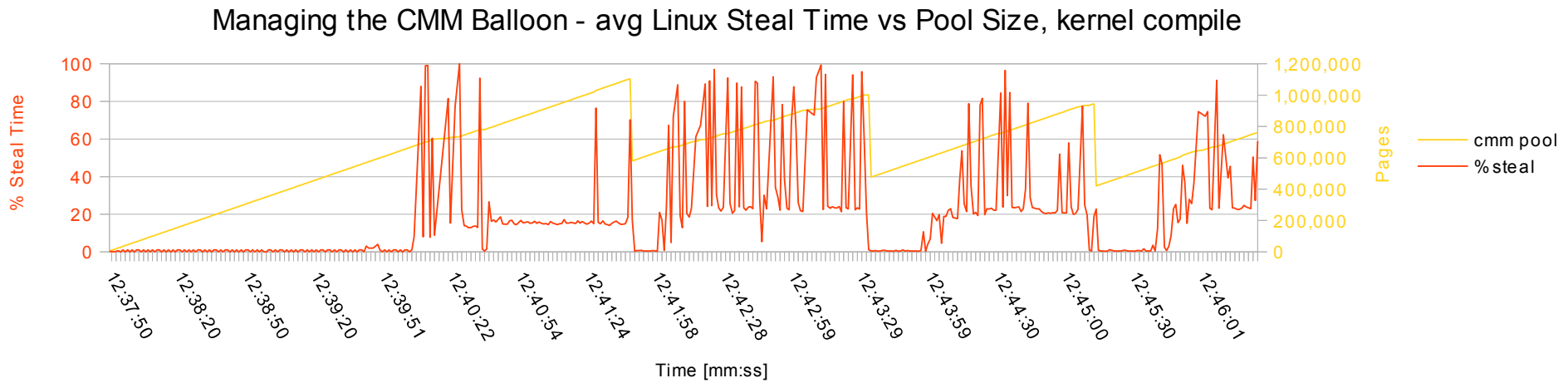
Dr. Juergen Doelle

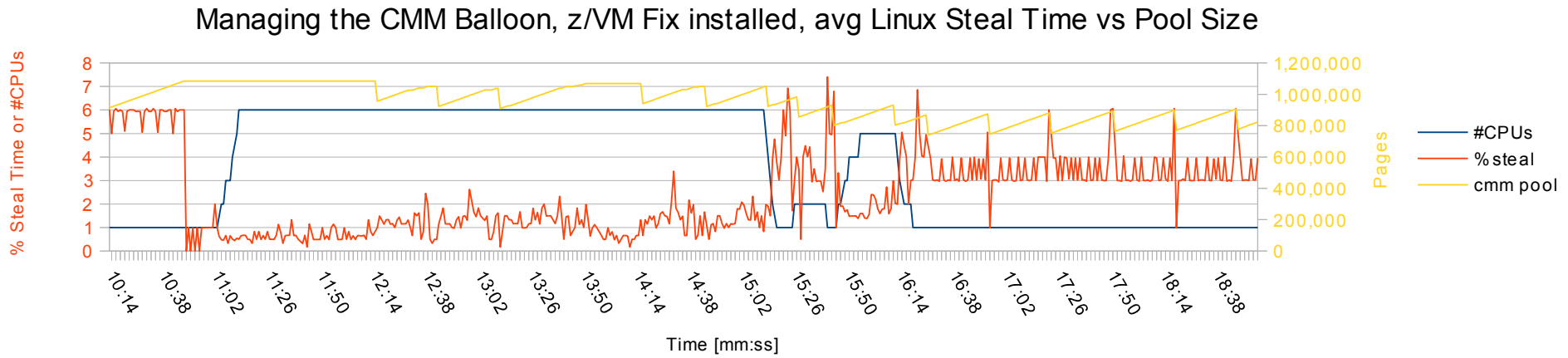
Linux on System z
System Software
Performance Analyst

IBM Deutschland Research
& Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

Backup

Memory plugging and kernel compile – Apar VM65060



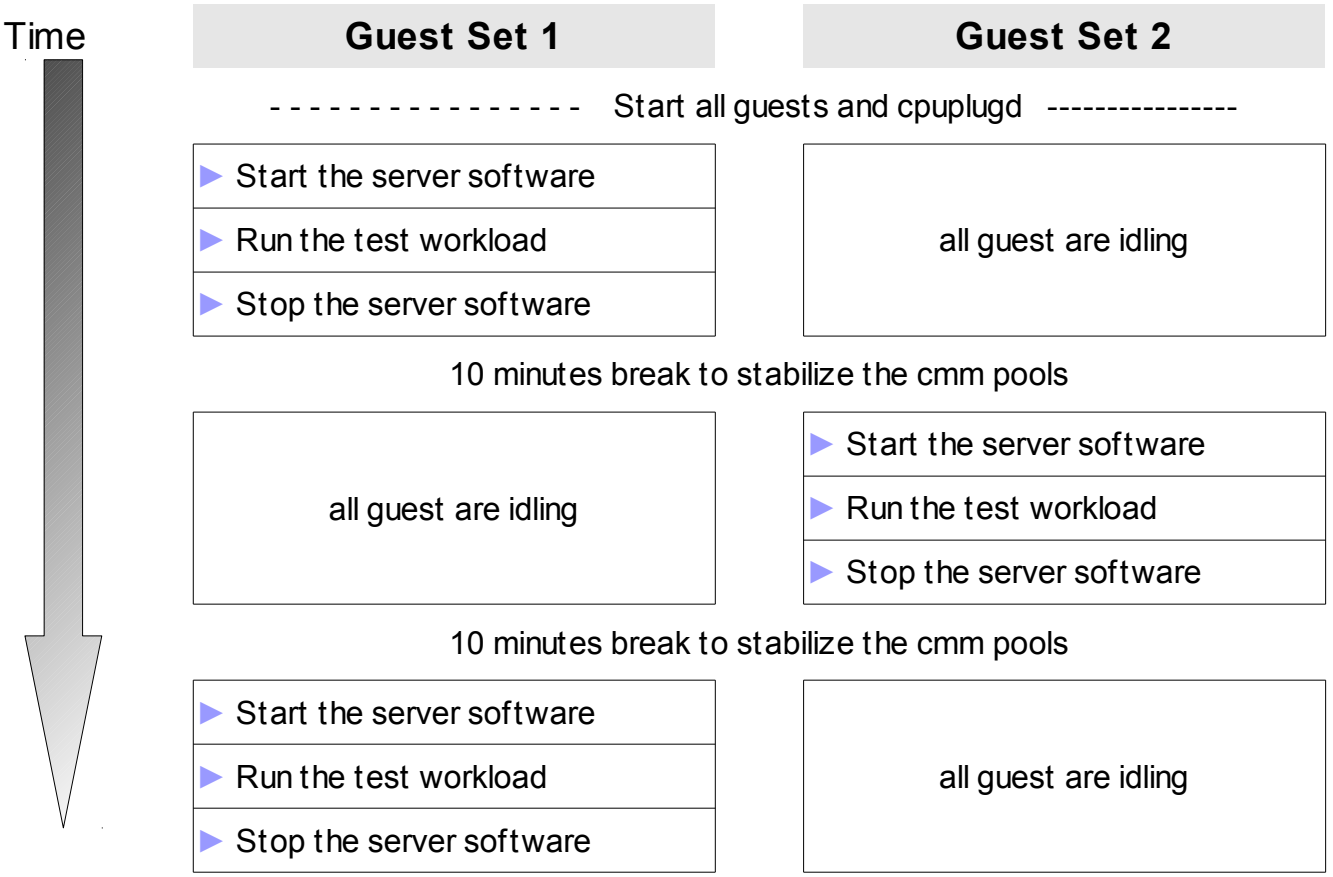


- Execution of Diag X'10' without APAR VM65060 blocks (all CPUs of) the virtual machine
  - Diag X'10 is used from Linux to inform z/VM that a memory page is free and can be 'stolen'
  - Applies for z/VM 5.4, 6.1, and 6.2
- **The fix reduced the steal time from above 80% to mostly below 4%**

Dynamic behavior

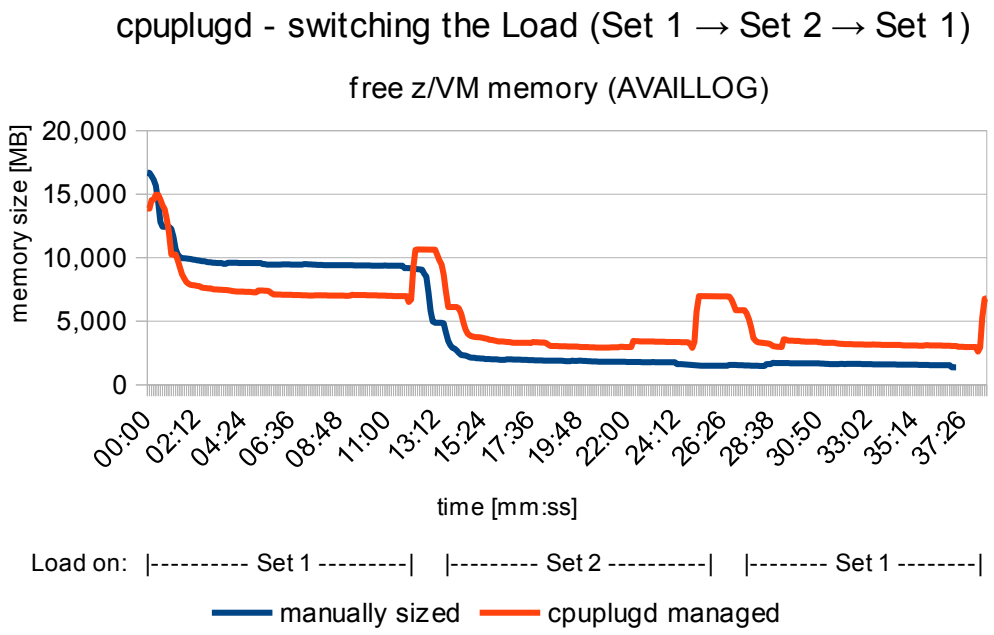
- **Use two sets of guests in the existing setup**
  - 2 x (2 Triplets + 2 Combos) = 16 guests

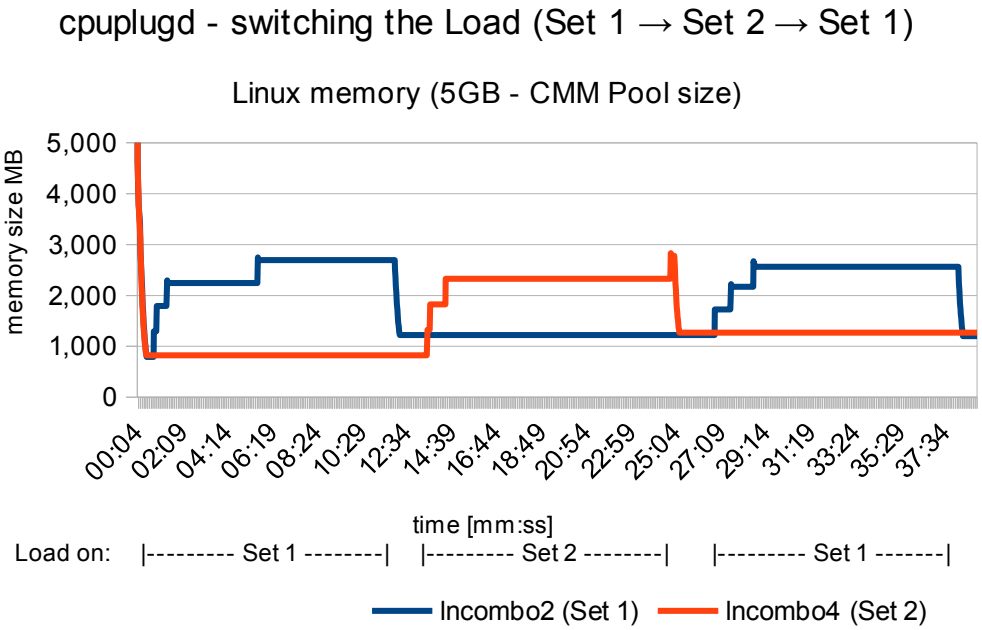
- **Test sequence**



- **This test should show, how the automated sizing with cpuplugd reacts on the workload shifts**

Dynamic behavior - results





- **in the z/VM view**
  - The cpuplugd environment wins after the first switch, it needs about 1.5 GB less memory
- **in the Linux view**
  - The major part of the memory is given back! About 1GB stays allocated → page cache
- **Conclusion**
  - The memory is given to the cmm pool from the cpuplugd daemon, but z/VM has sufficient free memory for new requirements
  - In Linux the memory must actively be given back from the application, e.g. terminating the application or shrinking buffers