Linux Technology Center

Linux on System z Performance CMM

Martin Kammerer
martin.kammerer@de.ibm.com

visit us at http://www.ibm.com/developerworks/linux/linux390/perf/index.html

2009-03-04

CMM

- 2 methods available:
  - VMRM-CMM (VM Resource Manager – Cooperative Memory Management) aka CMM1
    - Balooning technique
    - The z/VM resource manager controls the size of the guests
  - CMMA (Collaborative Memory Management Assist) aka CMM2
    - Guest page hinting technique
    - Allows CP to “steal” pages based on the usage information
    - Target is to identify unused pages and non-dirty pages with a backing
- Available with SLES10 SP1
- Both methods show performance improvements when z/VM hits a system memory constraint.

CMM activation

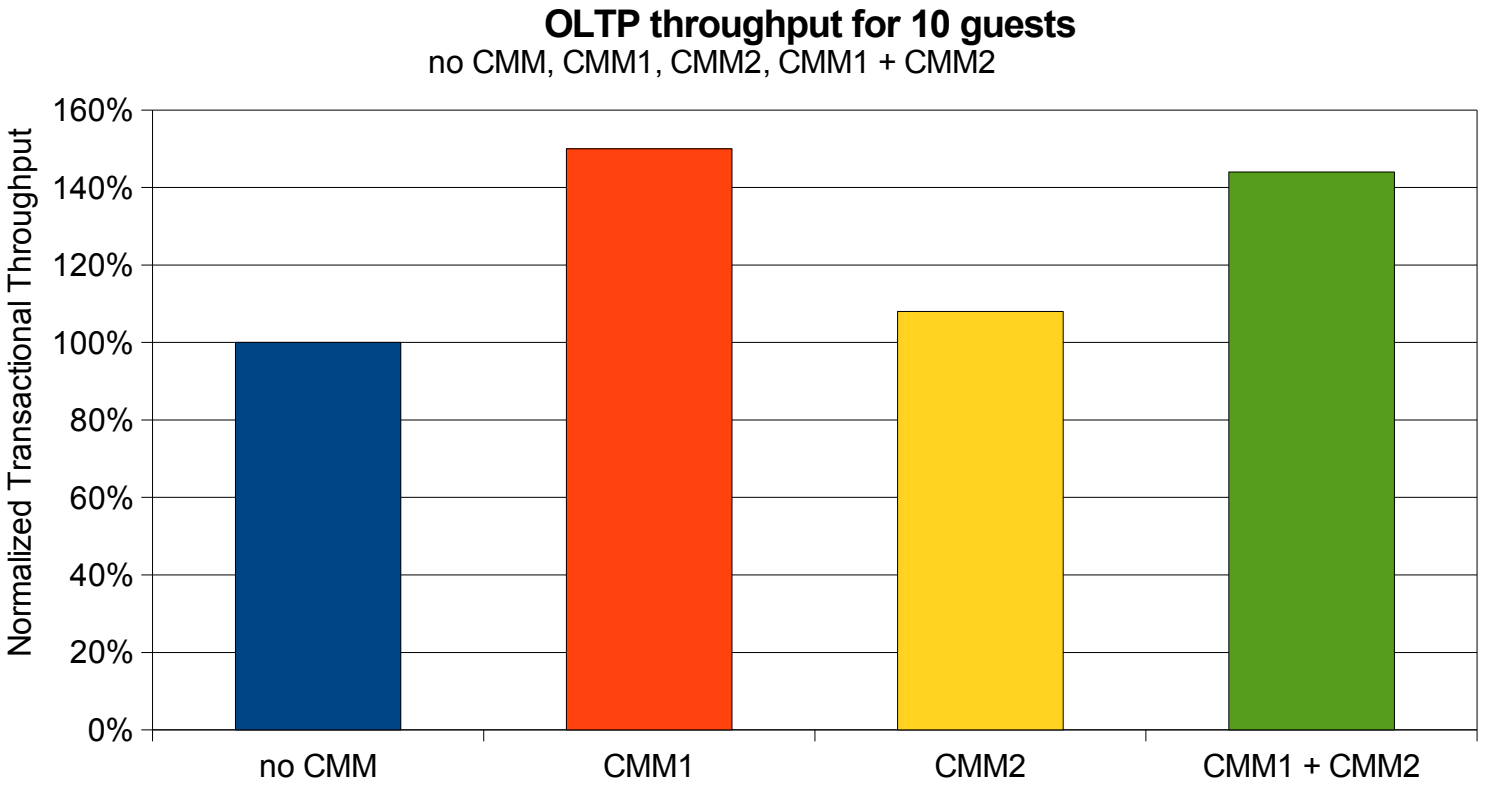
- CMM1
  - z/VM: **`NOTIFY MEMORY <guestname1> <guestname2> ...`**
  - Linux: load the kernel module with **`modprobe cmm`**
- CMM2
  - z/VM: **`CP SET MEMASSIST ON ALL`**
  - Linux: Set kernel parameter **`cmma=on`** in **`/etc/zipl.conf`**

CMM large application scenario - test setup

- Requirements
  - A software product should be tested which is frequently used by customers
  - The application should require and use large quantities of memory
  - The memory should be overcommitted
- Test environment
  - LPAR with z/VM, 10 CPUs, 80 GB central, 4 GB expanded storage
  - 10 SLES10 SP1 Linux guests, each with 3 virtual CPUs and 16 GB memory
  - All Linux guests require 3x of the available CPU resources and 2x of the available z/VM guest storage
  - OLTP database workload was chosen as Linux application
  - Tests with CMM1, CMM2 and a combination of both
- Expectation
  - Both features should improve the overall system performance and increase the overall throughput

CMM real life scenario - measurement results

- 50% throughput improvement with CMM1
- No big improvement when using only CMM2



CMM real life scenario - conclusions

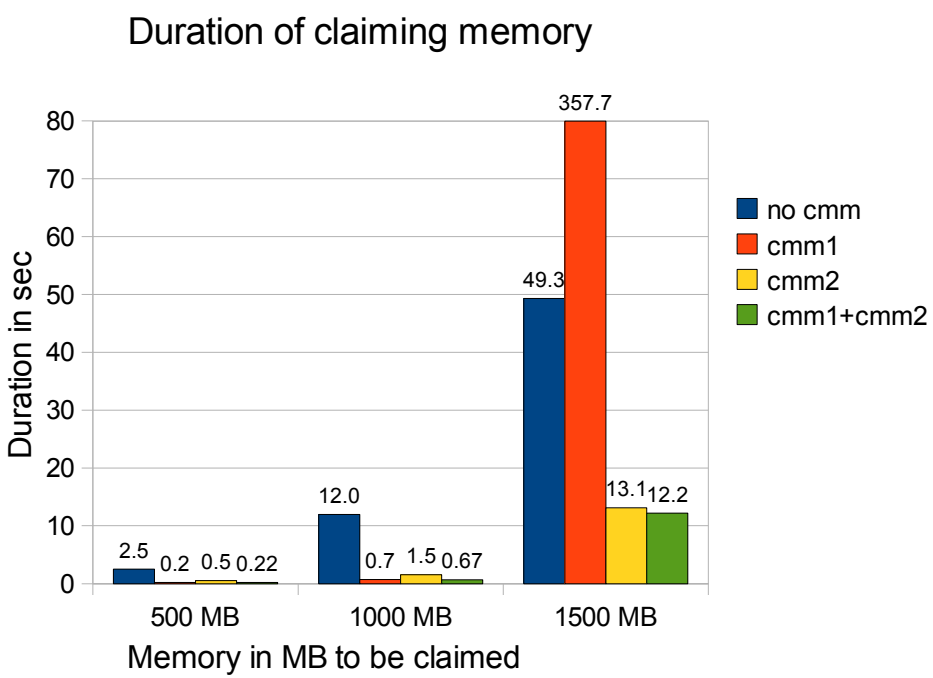
- Why did CMM1 help to improve the performance?
  - Initially the memory is 2x overcommitted
  - Then VMRM instructs the Linux guests to shrink their page cache containing the database buffer pool and the file system cache
  - The reduction is done in the file system cache, keeping the full amount of database buffer pool
  - Each Linux guest reduced its memory to approx. 8.5 GB, overcommitment decreases
  - Disk I/O is not suffering significantly from a smaller file cache
  - Transaction throughput is not impacted heavily
- Why did CMM2 not help to improve the performance?
  - The memory is always 2x overcommitted
  - The Linux guests always want their allocated memory because the database workload occupies all available file cache with its disk I/O
  - Each Linux guest continues to occupy 16 GB memory
  - With CMM2 Linux sets the page status and z/VM can select the non-dirty pages of the page cache to reuse for another guest
  - Each Linux guest claims memory as soon as he is scheduled

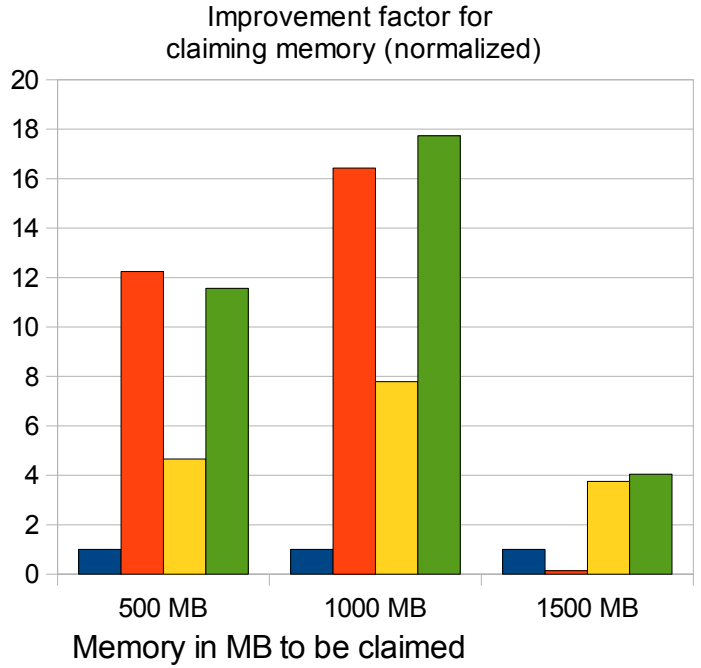
Special CMM2 scenario - setup

- Idea
  - In the real life scenario all Linux guests were permanently busy
  - Memory requirements were constantly on the same level
  - A situation with a guest suddenly claiming a big number of pages was not yet tested
- Test environment
  - 15 guests, touching all their memory, all z/VM storage used
  - A guest now claims 500 MB, 1 GB, or 1.5 GB of memory
  - We measure the duration of these operations
- Expectation
  - Both features should perform this exercise faster than a setup without CMM used

Special CMM2 scenario – measurement results

- In case of sudden memory claims CMM2 is the better choice
- CMM1 is doing well if the amount of requested pages is not too high





Special CMM2 scenario - conclusion

- Why is CMM2 good in all test cases?
  - z/VM can see which pages are clean
  - There is a big amount of non-dirty pages since we did not modify them
  - z/VM simply takes enough of these pages and assigns them to the requesting Linux guest
- Why is CMM1 good in the small and medium request test case and bad in the large request test case?
  - Before pages can be given to the requesting Linux guest we first have to process the shrink step to provide enough pages
  - Shrinking is done in Linux in decrements of 1 MB
  - In the small and medium request test case there was still enough memory left so that the shrink operation could complete in a short time
  - In the large request test case the Linux guests were busy to find free pages and the shrinking took extremely long
  - Finally the Linux guests started swapping

CMM overall conclusion

- CMM1
  - In cases of memory overcommitment CMM1 shrinks the Linux guests
  - If the initial memory definition and allocation for the Linux guest was roughly sized too high, CMM1 will correct this
  - Memory reduction in the Linux guests avoids frequent claims when each guest is dispatched
  - Effectiveness depends on how much page cache can be removed from the Linux guests without impacting the guest performance too much
- CMM2
  - Linux guests provide z/VM the page states so that z/VM can “steal” guest pages which can easily be recreated in situations of memory claims
  - The effectiveness depends on the amount of unused or non-dirty pages with backing that can be identified
- CMM1 + CMM2
  - The best choice to be prepared for all situations

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

DB2*
DB2 Connect
DB2 Universal Database
e-business logo
IBM*
IBM eServer
IBM logo*
Informix®
System z
Tivoli*
WebSphere*
z/VM*
zSeries*
z/OS*
ECKD
Enterprise Storage Server®
FICON
FICON Express
HiperSocket
OSA
OSA Express

* Registered trademarks of IBM Corporation

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.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective 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.