

MPO aware Oracle

Ravindra Talashikar

Sr Staff Engineer

Sun Microsystems

Ravindra.Talashikar@sun.com

Outline

- Motivation
- MPO in Solaris
- MPO aware Oracle
- Performance impact
- Conclusion

Motivation

- Sun/SPARC Enterprise servers have varying degree of NUMA characteristics
- Existence of MPO framework and API support in Solaris
- Solaris is MPO aware, applications can be made MPO aware for better performance
- Analysis of Memory Access Pattern for Oracle showed opportunities for better locality

MPO in Solaris

- MPO framework consists of :
 - > Concept of “lgroup” (locality group)
 - > CPU and Memory resources with certain affinity defined as a lgroup. e.g. A Uniboard on F-15K is one lgroup
 - > Concept of “home lgroup”
 - > Each LWP is assigned a lgroup as its home and is preferred for scheduling and memory allocation
 - > APIs for observability and control
 - > First introduced in Solaris 9 9/02 release

MPO in Solaris (contd...)

- Optimizations to improve locality of reference :
 - > Scheduler enhancements
 - > Scheduler prefers to schedule a runnable LWP on one of the CPUs in its home lgroup
 - > Affinities : STRONG/WEAK/NONE
 - > Memory allocator enhancements
 - > Private memory allocation preferably from home lgroup (heap, stack, ANON memory)
 - > Shared memory (ISM and non-ISM) allocation randomly across lgroups
 - > Allocation policies : first touch/next touch/random

MPO in Solaris (contd...)

- Lgroup APIs :
 - > APIs for an application to query and affect scheduling and memory allocation
 - > Observability
 - > meminfo(2) for address translation
 - > Control
 - > madvise(3C) to influence memory allocation
 - > liblgroup.so(3LIB) a new library of lgroup APIs. Introduced in Solaris 9 10/03 release
- Tools support :
 - > lgrpinfo, pmap -L, prstat -H, plgrp, ps -hH, pmadvise

MPO aware Oracle

- Objectives
 - > Increase locality of reference to shared memory (SGA) and private memory (PGA) access by Oracle processes
 - > Optimizations should benefit both - OLTP and DSS applications
 - > MPO specific optimizations should be enabled by default (OOB) for Oracle 10g

MPO aware Oracle (contd...)

- Architectural changes :
 - > Lgroup aware static memory allocation of SGA
 - > Partitioning of SGA into DISM based NUMA pools (Default/Keep/Recycle pools per NUMA pool)
 - > Oradism modified to use madvise(3C) with MADV_ACCESS_LWP
 - > Lgroup aware dynamic memory allocation within SGA
 - > Process and session state objects allocated close to foreground processes
 - > Allocation of buffer state objects
 - > Allocation of private strands for redo records
 - > Free buffer lookup by foreground processes within their respective NUMA pools

MPO aware Oracle (contd...)

- Architectural changes :
 - > Process affinity
 - > At least one dbwriter assigned exclusively per lgroup - cleaning dirty buffers from respective NUMA pool
 - > Logwriter scheduling close to logbuffer to improve affinity
 - > Foregrounds now have STRONG affinity for their home lgroup
 - > Enabling MPO
 - > Discovery at run time. Querying for available lgroups based on available CPUs
 - > Oracle instance is processor set aware. Memory and CPUs are used only from the processor set of an instance

Performance impact

- Performance impact of MPO depends on the ratio of local vs. remote memory accesses and is mainly controlled by the application
- MPO reduces stalls due to L2 cache misses by reducing remote memory accesses
- *Note : MPO does not reduce the total number of cache misses but the cost of servicing the cache misses*

Performance impact (contd...)

- Observations with MPO aware Oracle running on E-25k :
 - > OLTP
 - > 20% reduction in user level L2 cache misses serviced by remote lgroups
 - > 8% reduction in user CPI
 - > 6% performance gain in throughput
 - > 50% of L2 cache misses serviced locally, compared to 37% prior to MPO for 6 lgroup configuration
 - > DSS
 - > CPI much lower than OLTP
 - > Up to 10% performance gain from MPO optimizations
 - > 80% of L2 cache misses serviced locally

Performance impact (contd...)

- For OLTP :
 - > Post MPO, most of the L2 cache misses serviced remotely attribute to buffer cache access in SGA
 - > Difficult to improve locality for buffer cache access as it is shared by all the foreground processes across lgroups
 - > Equally sensitive to local and remote memory latency
 - > Typically user time accounts for 75% of CPU time. About 65% to 70% of user CPI attributed to stalls due to L2 cache misses
 - > Kernel component is higher than DSS and has higher (typically 75%) remote misses

Performance impact (contd...)

- For DSS :
 - > Higher user time – up to 90% of CPU time
 - > Most of the user level cache misses attribute to PGA access
 - > PGA allocated as heap or ANON memory in process's (Oracle PQ slave) address space
 - > Locality for PGA access benefits from MPO support in Solaris
 - > DSS applications can benefit more directly from any reduction in local memory latency

Conclusion

- Significant performance gain from MPO for OLTP and DSS applications by reducing user CPI
- Reducing remote memory accesses will be key to further improvement in OLTP performance
- Reducing local memory latency alone will have limited gains for OLTP applications
- However, impossible to achieve 100% locality without hardware support