CheatSheet z10 #47 zTidBits Processor Storage Subsystem

System z10's Cache
"The HW / SW Synergy"

•Caches are of two general categories: private and shared.
- There is a one-to-one correspondence between a CP and a private cache, that is, a cache is private to a particular CP.

•In the System z environment, shared caches are also implemented in which multiple CPUs share a single cache.
- Specifically, the z10 processor storage subsystem includes three caches:
  Private level 1 (L1) caches of 128 KB for data and **64 KB for instructions**.
  Private level 1.5 (L1.5) caches of 3 MB unified for data and instructions.
  Shared level 2 (L2) caches of 48 MB unified for data and instructions.

•There are up to four L2 shared caches in a single symmetric multiprocessor system
- Each L2 cache is shared by all of the CPUs on that processor unit (PU) book.
  NOTE: The L2 shared cache is an extension of the binodal L2 cache first implemented in the S/390 Enterprise Server 9672 G5 mainframe.

•Accessing a line in the L1 cache is faster than accessing it in the L1.5 cache, which is faster than accessing it in the L2 cache.
- To reduce the latency delay, if a line misses the L2 cache but is found in both main memory and a remote L2 cache, the remote cache provides the line.
- A cache is considered remote if it is located on a different PU book from the one on which the CP is located.
- Fetching the line from a remote L2 cache is less than half the latency than fetching it from main memory.
  NOTE: Accessing a line in the local L2 cache can be more than three times faster than accessing the same line in any of the remote L2 caches.

•Since the S/390 G5, the line or block size at all cache levels on System z mainframes has been 256 bytes.
- In other words, all data and instructions are transferred between main memory (central storage; CSTOR), caches, and the CPUs in 256-byte blocks.

•When a line is first referenced by a CP, the line is installed in the associated shared L2 cache and in the L1.5 and L1 caches.
- Each level of the cache hierarchy is a superset of the subordinate caches, meaning that every line in the L1 cache also exists in the associated L1.5 and L2 caches.
  > This simplifies coherency protocols and minimizes latencies.
  > Cache is designed as a ***store-through*** *cache*, which means that altered data is synchronously stored into the next level.

•If CPU1 references a line that has already been installed in the cache of CPU2, the L2 cache associated with CPU2 has all the information necessary to ensure cache coherency.
- A single cache line has one of two basic states: **shared** or **exclusive**.
  > A shared cache line cannot be updated or modified. Therefore, it can exist simultaneously in the caches of more than one CP.
  > On the other hand, a line in the exclusive state is exclusive to one CP and, therefore, that CP may modify the line.
- If an exclusive line is referenced by another CP, the state of the line is changed to shared.
  >If any stores (modifications) are pending, they must first complete before the line can be copied to the other CP.
- When two CPUs share a common line, the line resides in the private caches of both CPUs.
  > If the CPUs share an L2 cache, only one copy of the line is needed in the L2.
  > Although, if the CPUs span multiple L2 caches, a copy of the line resides in each of the L2s.

•If a line exists in an L2 cache in the shared state and another CP requests the line to be installed in its cache in a shared state as well, the L2 cache can honor the request Immediately, no pending stores are then possible on this line.
- While most instructions are never modified and therefore exist as shared in all caches, most data is eventually updated, requiring the line to be held exclusive during the modification.
- A subsequent fetch of an exclusive line by another CP cannot be honored immediately by the L2 cache holding the line because coherency protocols must be followed.

•When a line "***ages out***" (i.e., when a new line replaces an existing line, usually the oldest) of an L1 cache, it still exists in the L1.5 cache.
- Subsequently, when that line ages out of the L1.5 cache, it still exists in the L2 cache.
  NOTE: For exclusive lines, the L1.5 cache sends a signal to the L2 cache that the line no longer exists in the L1.5 cache. The L2 cache marks the exclusive line "exclusive to no CP." If another CP requests this line, the holding L2 cache can immediately honor the request for this line because no stores are pending in the L2 cache and no L1 cache or L1.5 cache is holding the line.
- The L1.5 cache age-out enhancement provides ***a great benefit to performance*** compared with traditional coherency protocols.
  NOTE: When a subsequent CP requests the line, the latency is half with this enhancement. Because this is built into the hardware, all programming levels can benefit!

•**SO WHAT DOES THIS ALL MEAN**: Significant potential for performance improvement exists by redispatching interrupted units of work to the same physical CP where they ran prior to the interruption, thereby maintaining affinity to private or shared caches.

⚠ An affinity node is a set of CPUs that share a level of hardware cache, which minimizes cache coherency and main memory delays. The WLM obtains physical topology information about the placement of the CPUs on the System z10 server and groups them into affinity nodes. Polarization is the degree of affinity between logical to physical CPs.

•Prior to the introduction of the System z10 mainframe, both z/OS and LPAR hypervisor were much less sensitive to the CP topology.
•The z/OS operating system would dispatch a unit of work to any available and ready logical CP (SMP), regardless of where that unit of work had been running prior-(Figure A)
- In addition, as a second dispatching stage in a shared LPAR environment, the LPAR hypervisor (architecturally considered part of System z hardware) would dispatch logical CPs to physical CPs without maintaining a strong affinity between these two entities potentially creating instruction to cache latencies.

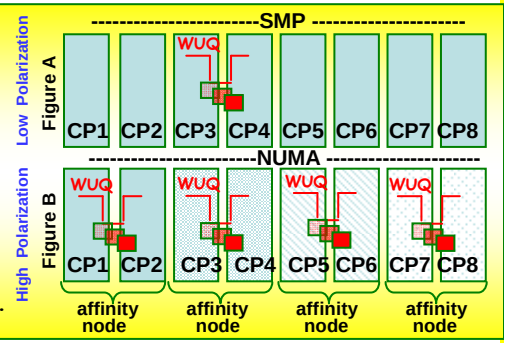
•The best way to avoid the added delays described above is to keep each unit of work on the same CP for the life of that unit of work.
- However, typically System z runs at high CP utilizations, which means that waiting for the same CP to become available to redispatch a unit of work can result in unacceptable delays. To balance these two factors—keeping units of work on the same CP and minimizing delays, ***SMART*** dispatching was introduced.

• **HiperDispatch** using NUMA algorithms aim to maintain each ***affinity node*** on CPs that share an L2 cache.
- Thus, the extra delays to move a line from one L2 to another are avoided.
- Instrumentation (CPMF) data on System z was essential in tuning HiperDispatch scheduling algorithms to achieve this balance.

Correlation between work unit queues (WUQs) and affinity nodes on a system with eight CPs and two CPs per affinity node.

(Figure A) WUQ without affinity boxes represent z/OS dispatcher queued work units.

(Figure B) WUQ to affinity node correlation on a system with eight CPs and two CPs per affinity node.



[Non-Uniform Memory Architecture (NUMA) is a computer memory design used in multiprocessors, where the memory access time depends on the memory location relative to a processor].

•A necessary prerequisite for any software cache optimization with regard to affinity dispatching of work is that the *virtualization layer* between the logical CPs ,which is visible to software, and the physical CPs (the LPAR hypervisor) are maintained.
•An interface between hardware and software is required that allows the software to obtain information about the physical placement of all CPs relative to caches and main memory.
- This is required for the cases in which a unit of work cannot be dispatched to exactly the same CP on which it had been running before, because the CP is in use when the redispatch should occur. In this case, there is a conflict between the need for immediate access to a CP (high external throughput of work and lower response time) and the attemp to optimize cache or main memory access paths (high internal throughput of the machine).
- The conflict occurs when trying to redispatch to a specific, but momentarily unavailable, CP, but it can be resolved by redispatching the unit of work in the above example to another CP ready for work that shares levels of cache with the preferred CP.

•The z10 server fulfills the requirement to maintain affinity between logical and physical CPs and to provide information to z/OS about the physical topology of CPs relative to caches and main memory.
-These capabilities can be enabled by z/OS on a per-partition basis by switching the partition into HiperDispatch mode via a new z10 architecture-level specific instruction.

•When a partition is switched into HiperDispatch mode, the LPAR hypervisor creates affinities between the logical CPs as observed by the OS and the physical CPs.
•Because the System z platform allows the definition of multiple partitions that share a pool of physical CPs, there are different levels of affinity between logical and physical CPs.
•Each partition can have up to as many logical CPs configured as there are total shared physical CPs, thus allowing each partition to potentially access the full pool of shared CPs.
•Other examples of improved cache performance is prefetch data, a new cache-directive z10 instruction, initiates the bring in and release of lines from and to the processor storage subsystem.

IBMs new System z10 hardware instrumentation, the central processor measurement facility (CPMF), supports software performance optimizations by providing counters and sampling allowing the software to measure central processor (CP) activities to determine hotspots. The CPMF is nondisruptive, has low overhead, and can run in multiple logical partitions simultaneously and is available to each CP.