z/TPF V1.1

TPF Users Group - Fall 2009

z/TPF Performance for Modern Processor Architectures

Name: Mark Gambino
Venue: Main Tent

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Industry Trends in 1980s and 1990s

- **Processor (central processing unit – CPU) speeds increased roughly 60% per year**
- **Main memory access speeds only increased 10% per year**
- **Given these trends, it was only a matter of time before memory latency became the biggest inhibitor to system performance**
  - The “memory wall” is the growing disparity of speed between CPU and memory outside the CPU chip

Industry Trends This Decade

- **Processor (CPU) speeds have continued to increase, but at slower rates:**
  - Year 2000
    - IBM ***System z900*** clock speed was **1.1 GHz**
    - Intel ***Pentium 4*** clock speed was **400 MHz**
  - Year 2008
    - IBM ***System z10*** clock speed was **4.4 GHz**
    - Intel ***Core 2 Extreme*** clock speed was **1.6 GHz**
- Gap between CPU speed and memory access speed has widened even more

Gaze into the Crystal Ball

Industry Forecast

Rate of CPU performance increases will slow down

Primary way to increase overall server performance will be to add more CPUs

Dark Clouds on the Horizon?

- **Some server vendors are taking the approach that given where computer hardware technology is going, re-architecting applications will be needed to perform well on modern processors**
- **Redesigning/rewriting applications is very expensive and risky**
- **Emerging programming models such as parallel programming do not map very well to OLTP workloads**

Light at the End of the Tunnel

- **IBM believes you should leverage your existing assets (applications)**
- **Improve performance using a combination of hardware (System z processors) advancements and software (z/TPF operating system) optimizations for critical path code**
- **Several z/TPF performance enhancements have been made that require no application changes**

Understanding the Problems and Solutions

Memory Caches

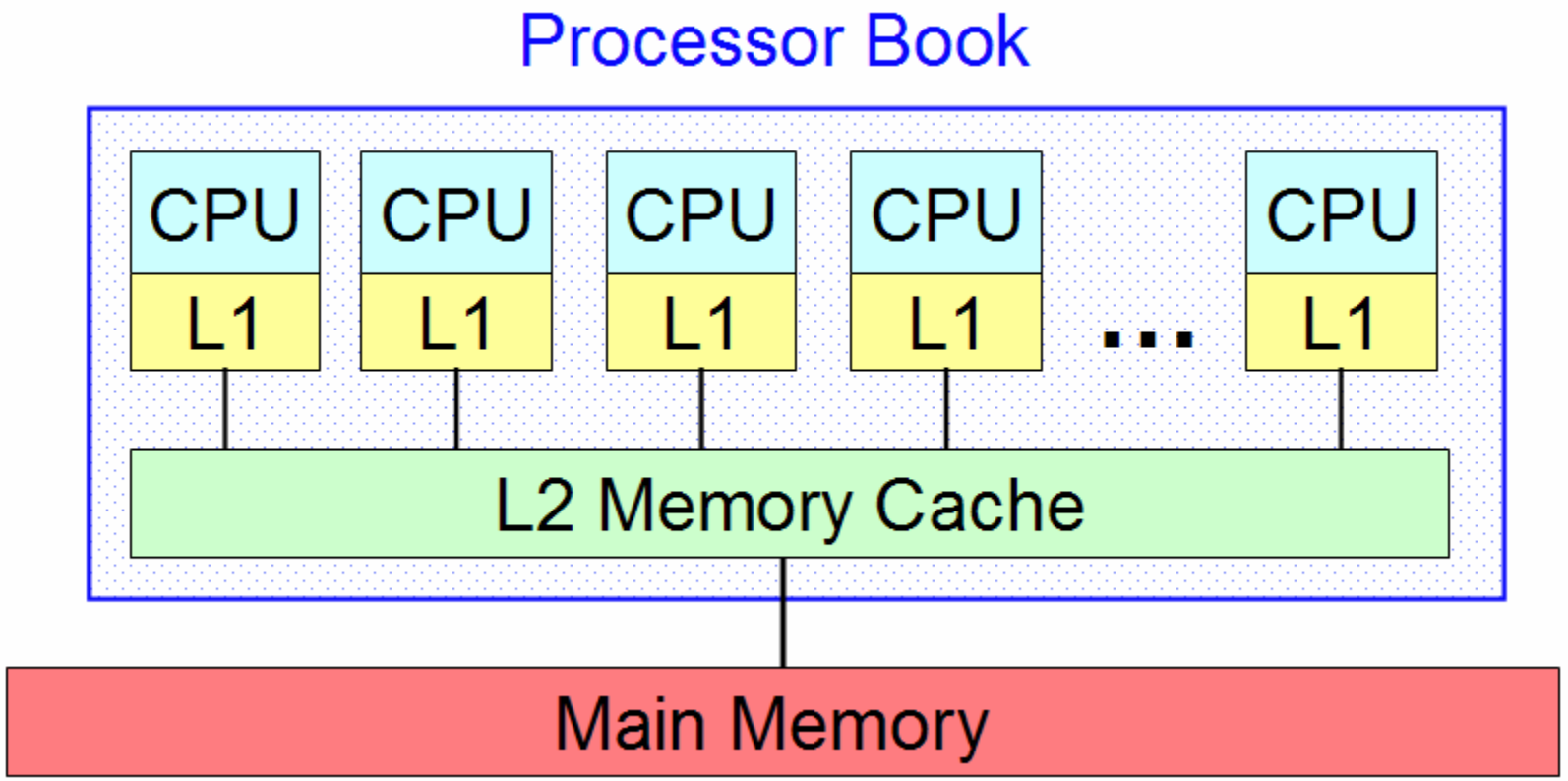
- **To improve memory access time, memory caches have been added on the CPU chip and between the CPU chip and main memory**
  - If data is found in memory cache (“cache hit”), no need to go to main memory
- **Main memory uses dynamic random access memory (DRAM) technology**
- **Memory caches use static random access memory (SRAM) technology**
  - ✓ SRAM is faster than DRAM
  - ✗ SRAM is larger and more expensive than DRAM

Modern System Memory Hierarchy

- **Level 0 (L0) – Registers**
  - Small storage buffers in the CPU
- **Level 1 (L1) – On CPU chip memory cache**
  - Very small in size, typically SRAM
- **Level 2 (L2) – Off CPU chip memory cache**
  - Small in size, shared by multiple CPUs, typically SRAM
- **Level 3 – Main Memory**
  - DRAM or eDRAM
- **Level 4 – Virtual Memory**
  - Disk

The further away the data resides from the CPU, the more CPU cycles spent/wasted to access the data

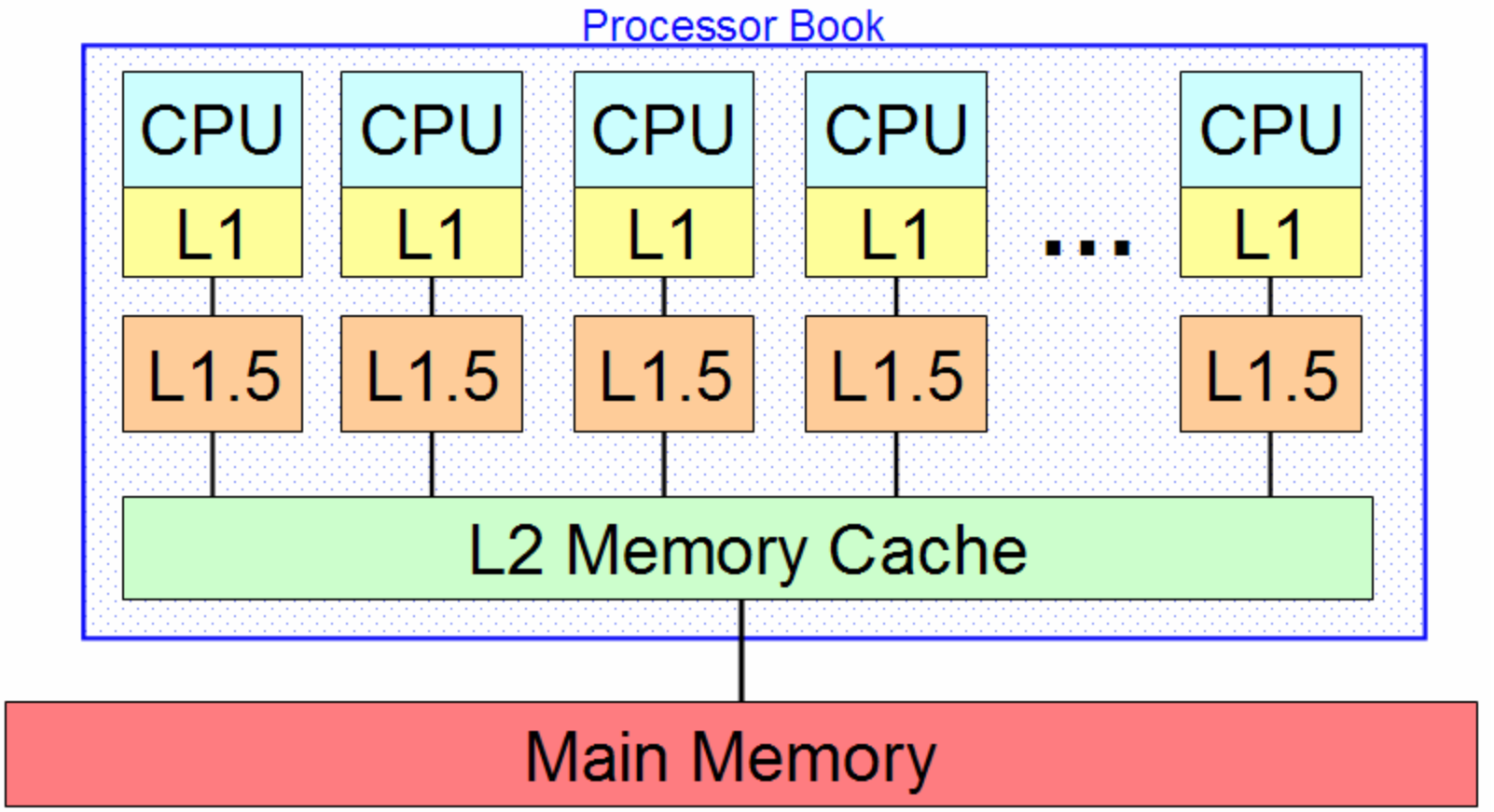
IBM System z9 CPUs and Memory Layout



IBM System z10 Memory Design Improvements

- **Methods to improve memory cache hit percentage include:**
  - Increasing the number of caches
  - Increasing cache size
- **z10 did both:**
  - Added an additional level of per CPU memory cache
  - L2 cache size increased by 20%

z10 CPUs and Memory Layout



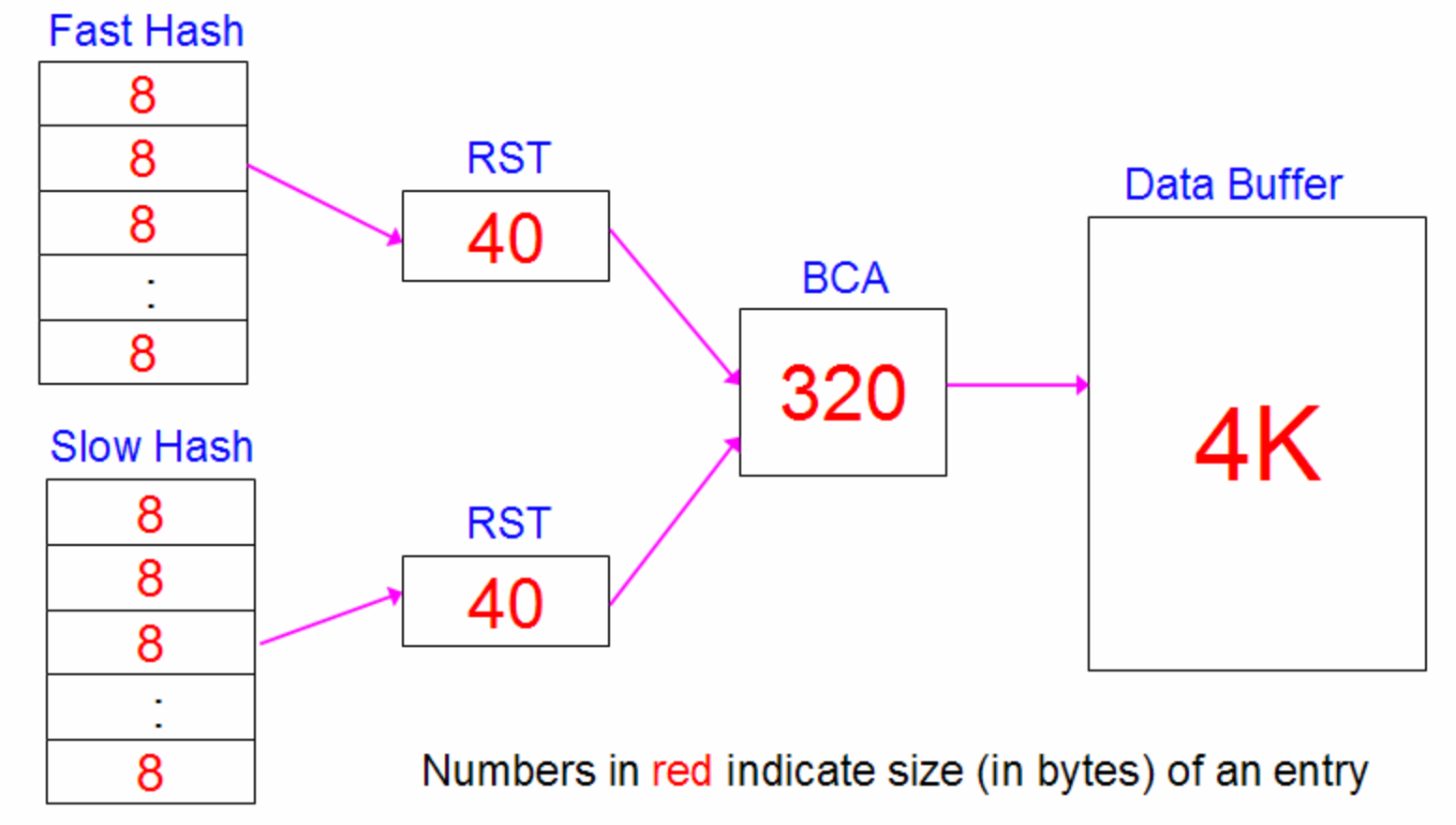
Cache Footprint

- **IBM System z memory cache lines are 256 bytes**
- **“Cache footprint” refers to the number of memory cache lines accessed**
- **Reducing the cache footprint improves memory cache efficiency**
  - Less likely to bump needed data out of cache
  - More likely to get cache hit on needed data

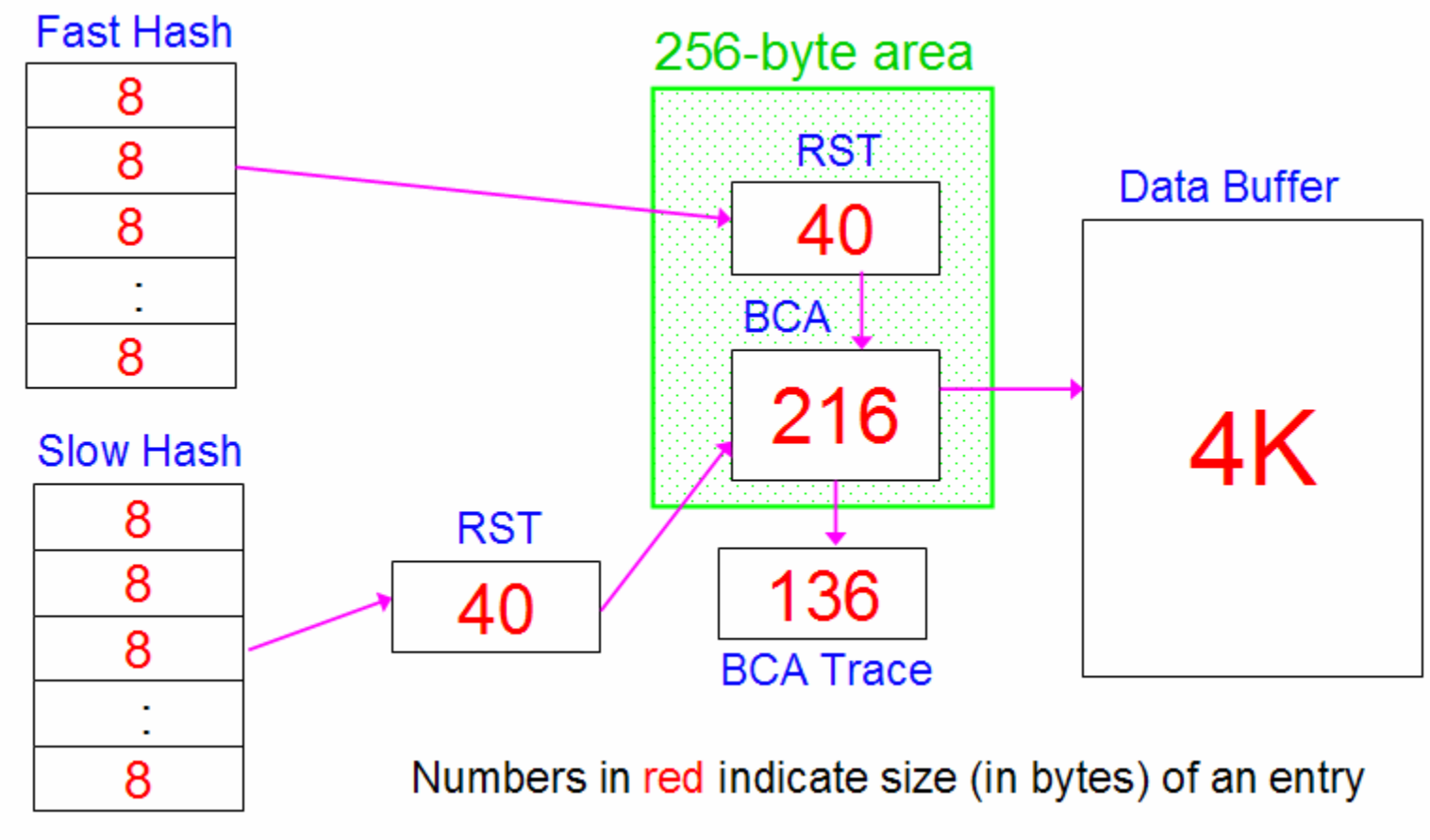
Reducing Cache Footprint in Heavily Used z/TPF Code

- **z/TPF runs very I/O intensive workloads**
- **Typically around 10-20% of processing time on z/TPF is in the virtual file access (VFA) code**
- **Accessing a record in VFA using fast path hash touched 3-5 cache lines just to get to the data buffer**
- **APAR PJ36371 restructured VFA control blocks**
  - Moved DEBUGV trace area to its own block
  - Combined fast path RST entry and BCA entry into the same cache line (same 256-byte area, 256-byte aligned)
    - Reduced cache footprint for fast path down to 2 cache lines

Original VFA Control Block Structures and Chaining



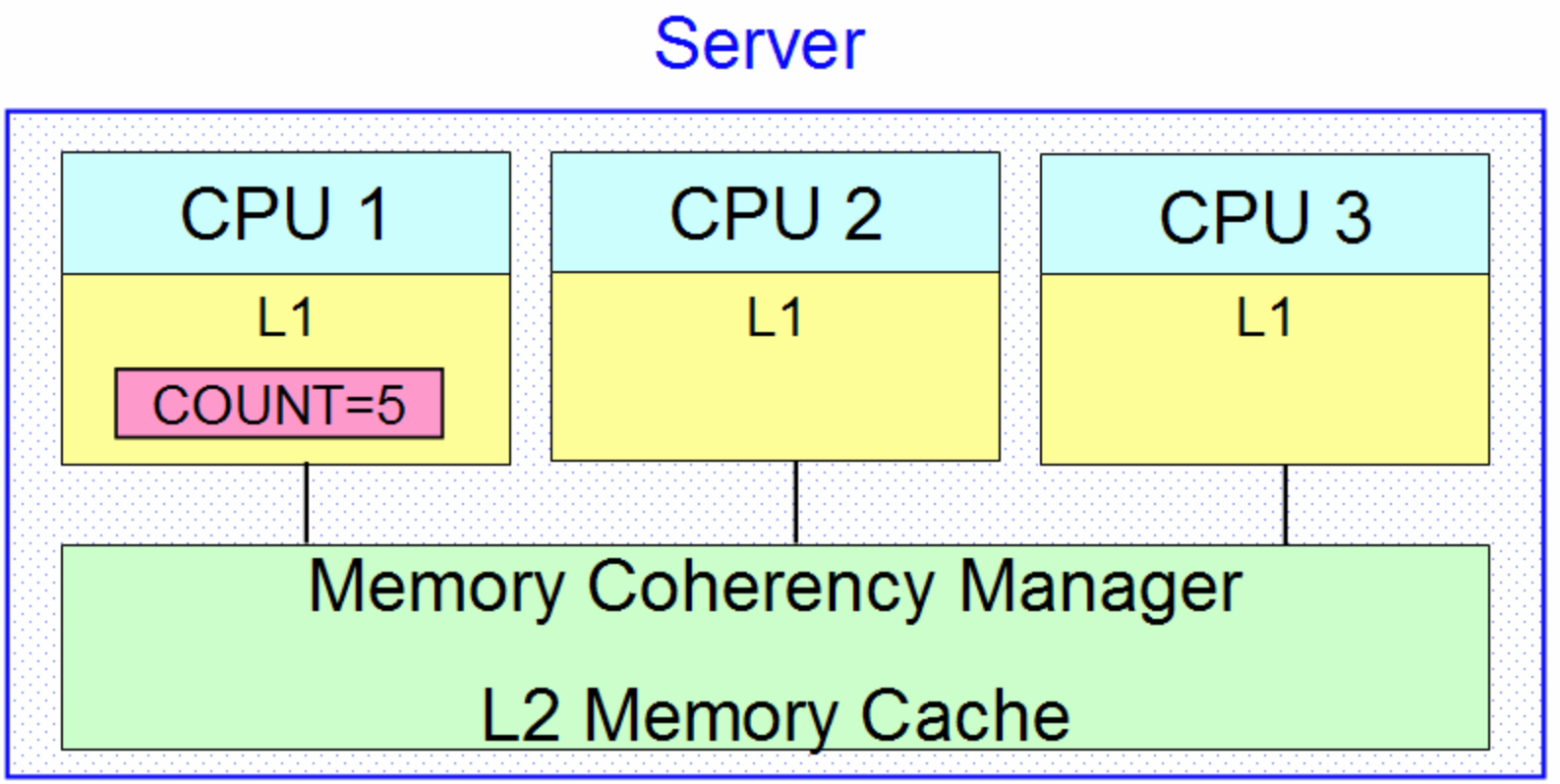
PJ36371 - VFA Control Block Structures and Chaining



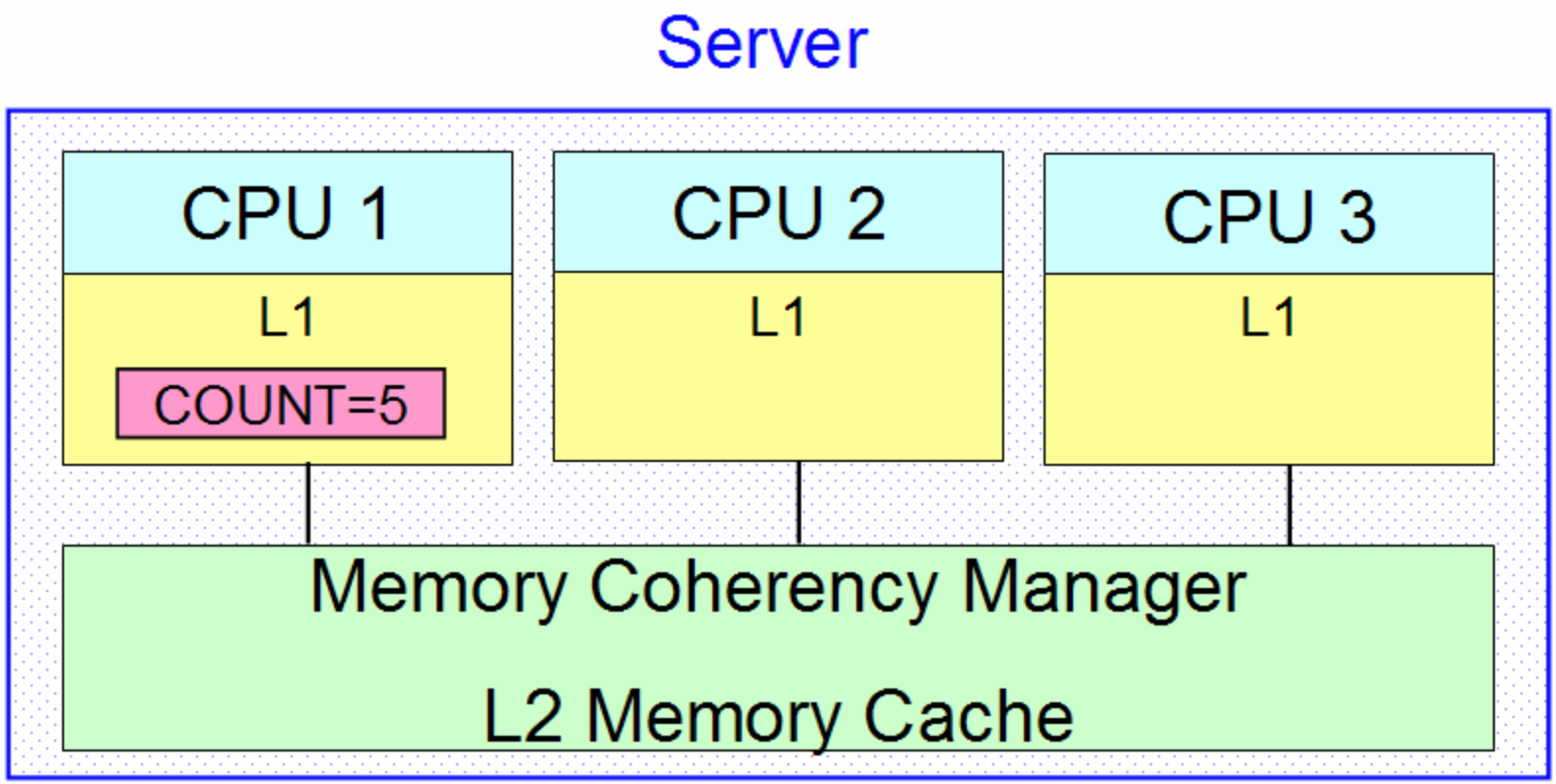
Symmetric Multiprocessing (SMP) Memory Considerations

- **Two or more CPUs sharing memory**
  - Called tightly coupled support in TPF
- **Has been the norm for servers for a long time**
- **Now the norm for desktop client machines as well**
- **Want to avoid memory cache thrashing in critical path code**
  - Data in memory that is updated **very** frequently by multiple CPUs
  - Strains the memory subsystem as cached data is constantly being invalidated in particular CPU caches as the data is updated by different CPUs

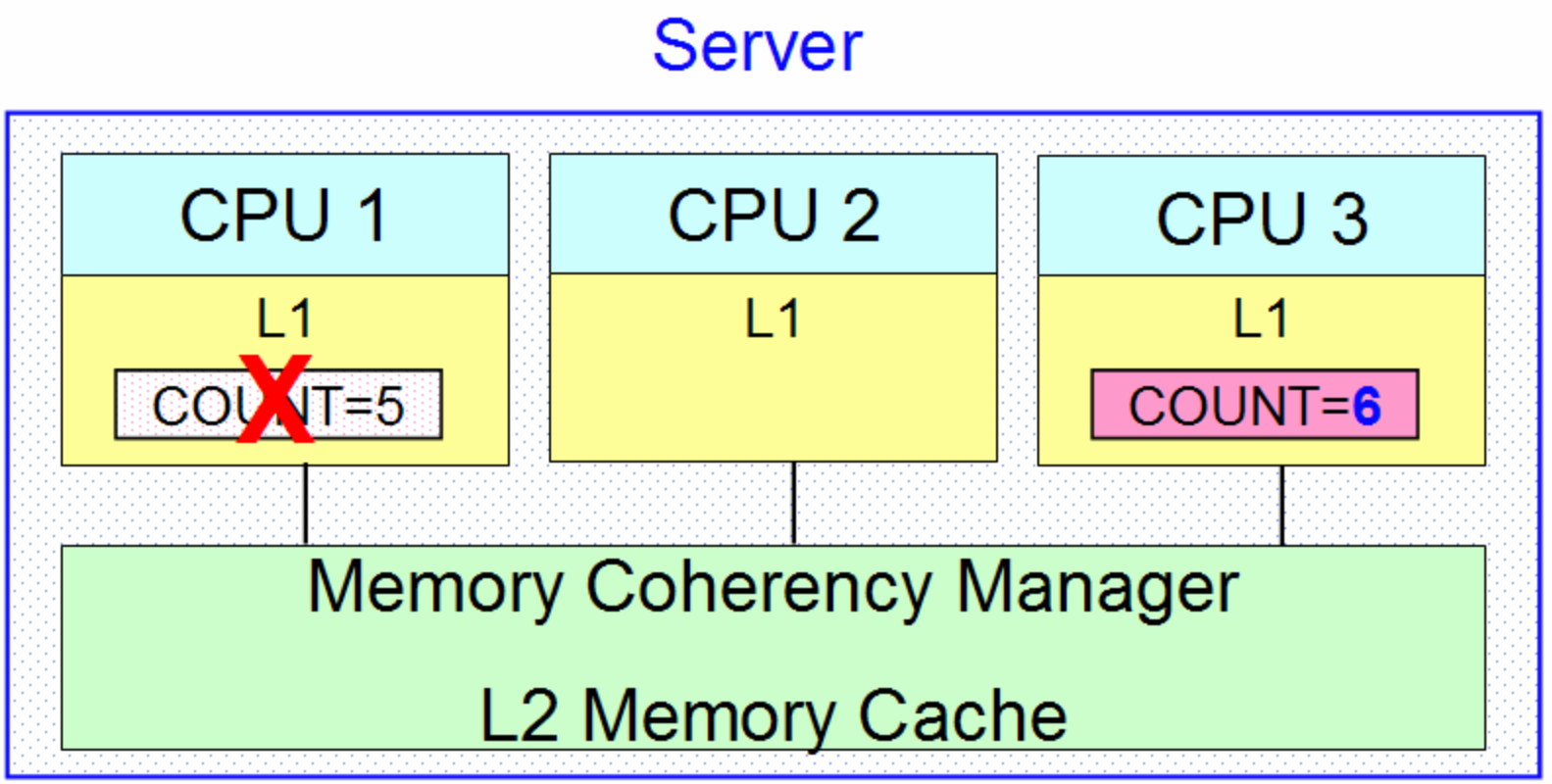
Step 1 - Data in L1 Cache on CPU 1



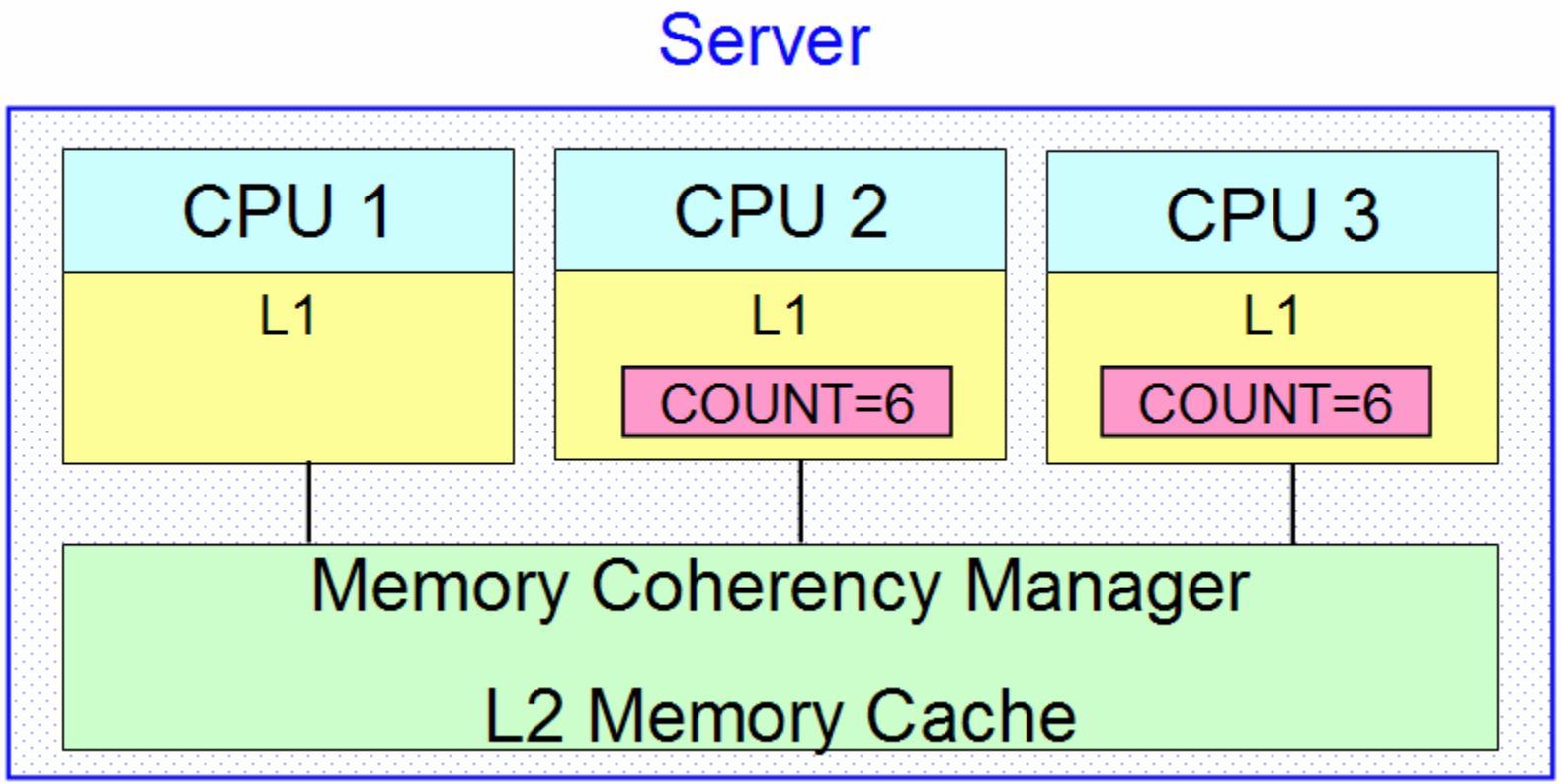
Step 2 - Data Accessed by CPU 3



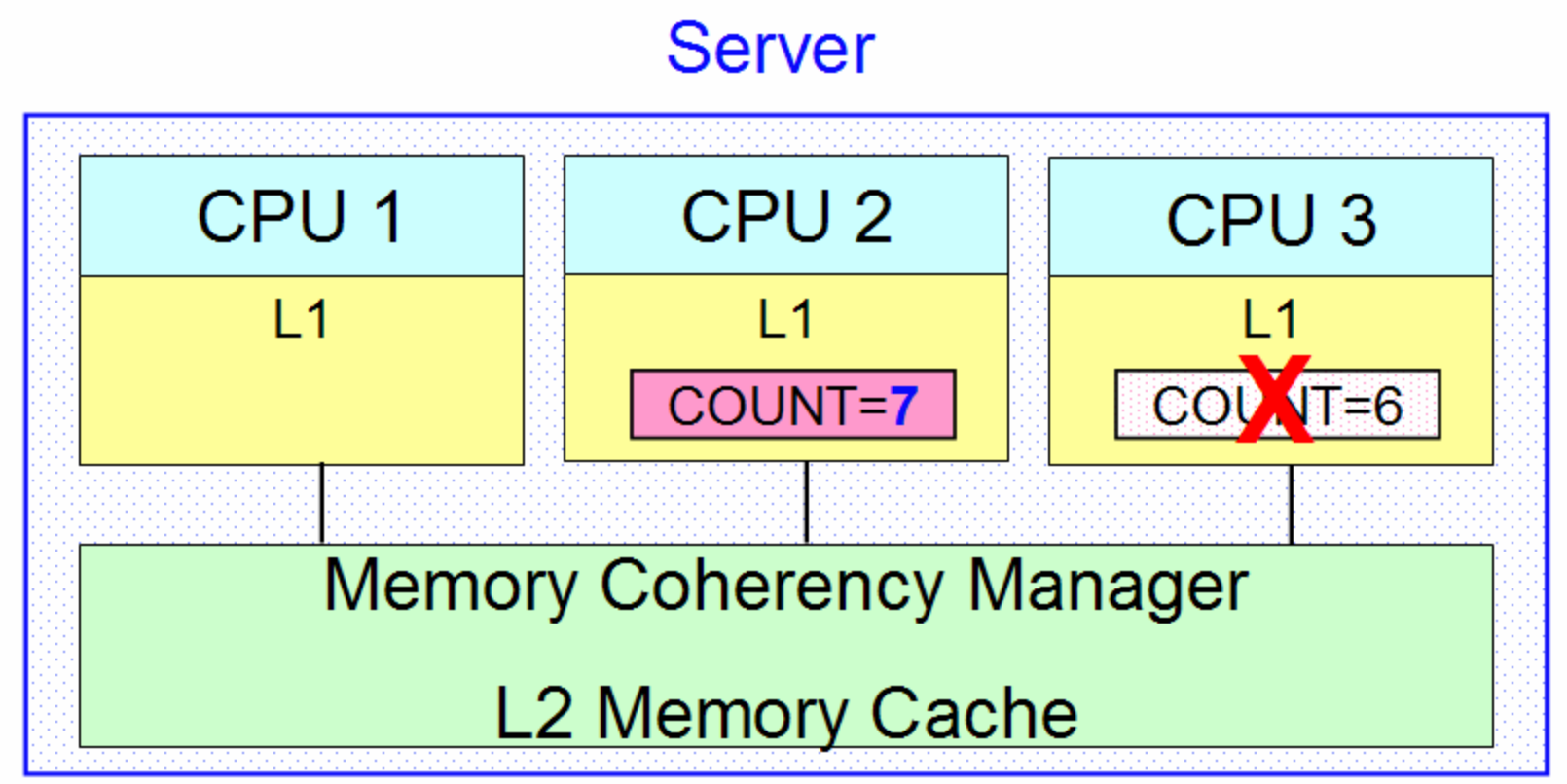
Step 3 - Data Updated by CPU 3



Step 4 - Data Accessed by CPU 2



Step 5 - Data Updated by CPU 2



Common Scheduler Algorithm Analysis

- **Shared queue**
  - ✓ Most efficient from pure mathematical point of view
  - x Heavy lock contention and cache thrashing overhead on modern processors
    - x Problem gets worse as number of CPUs is increased
- **Round robin**
  - x Has been proven to not work well for mixed workloads
- **Least Queue**
  - x Requires that you look at each CPU's queue size every time a task needs to be scheduled
  - x Queue sizes constantly changing, cache thrashing overhead
    - x Problem gets worse as number of CPUs is increased

New z/TPF Scheduler Meets the Challenge

- **Scheduler (CPU loop) is executed +10K times per second on a large z/TPF server**
- **APAR PJ35517 provides a new z/TPF scheduler**
- **Eliminates shared work queues (like shared ready list)**
- **Very efficient memory caching**
  - Test results have shown that when selecting an I-stream to assign a work item to, data necessary to select an I-stream is in cache +95% of the time
- **Scheduler overhead does not increase as the number of CPUs (I-streams) is increased**
- **Dynamically adjusts to changing conditions**
  - Examines list sizes and utilization of each CPU
  - No user tuning necessary

CPU Affinity versus End User Response Time

- **When a running process does a DASD I/O operation (process was running on CPU X)**
  - Redispatching the process on the same CPU (CPU X) is best from a memory cache point of view
    - Most of the working set of memory being used by the process is likely still in cache on CPU X
  - If there is a large number of tasks already queued on CPU X
    - Redispatching the process on a different CPU may improve end user response time

Optimizing Memory Caching **AND** Response Time

- **If ECB currently running on CPU X issued a DASD I/O and there is no I-stream affinity for this ECB**
  - Before APAR PJ35517
    - Chances that the ECB was redispatched on same CPU (CPU X) was 1 in N, where N is the number of CPUs
    - Percentage of ECBs defined as AFFINITY=NONE had to be kept low
  - With APAR PJ35517
    - ECB redispatched on same CPU (CPU X) +99% of the time
    - ECB only redispatched on a different CPU if CPU X's ready list queue size significantly larger than other CPUs' ready lists
    - Percentage of ECBs defined as AFFINITY=NONE can be increased
    - Enables balanced CPUs utilization and consistent response time at high CPU utilization

Translation Look-Aside Buffer (TLB)

- **Cache used to speed up virtual to real memory address translation**
- **Caches recently accessed virtual addresses and key information from their corresponding page table entries**
- **Hardware cache per CPU on System z**
- **Virtual address translation**
  - ✓ TLB hit – entry found in the TLB
  - ✗ TLB miss – entry not found in TLB
    - Must examine segment and page table entries to resolve address
    - Takes more time to resolve the address

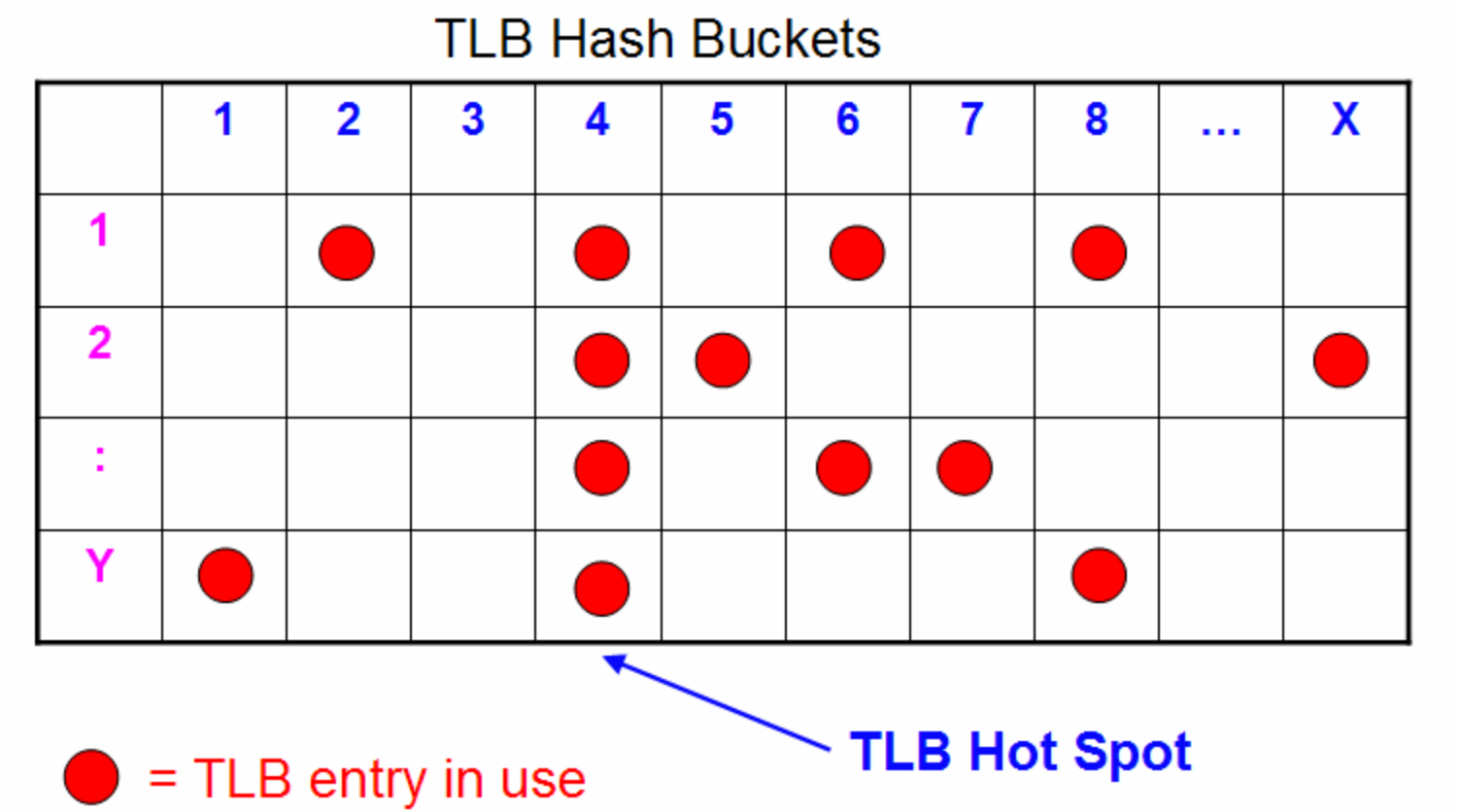
TLB Structure and Usage

- **TLB contains X number of hash buckets**
- **Each hash bucket has Y number of entries**
- **Virtual address translation**
  1. Hash the virtual address to determine which of the X hash buckets to use
  2. Search the Y entries in that hash bucket for a match
     - If match found (TLB hit), use the entry
     - If no match found (TLB miss)
       - Translate the address using segment and page tables
       - Reuse one of the entries in this bucket to save the translated address

Try to Avoid TLB Hot Spots

- **Ideally the working set of virtual addresses used by processes on a CPU results in an even distribution across the TLB hash buckets**
- **If a large amount of working set virtual addresses accessed use the same TLB hash bucket**
  - TLB hit rate is reduced (TLB miss rate is increased)
  - Virtual address translation time increased
    - CPU utilization increased as well

TLB Example



Improving TLB Usage on z/TPF

- **APAR PJ35230**
  - Before
    - Starting virtual address of preallocated ECB private area was the same for all ECBs
  - After
    - Starting virtual address of preallocated ECB private area varies ECB to ECB
- **APAR PJ32288**
  - Before
    - Starting virtual address of preallocated ECB heap and program stack were the same for all ECBs
  - After
    - Starting virtual address of preallocated ECB heap and program stack vary ECB to ECB

Address Translation Tables

- **z/Architecture mode with dynamic address translation (DAT) mode enabled**
- **Region tables point to segment tables**
- **Segment table**
  - Maps 2G of memory
  - Contains 2048 entries, each pointing to a page table
- **Page Table**
  - Each page table maps 1M of memory
  - Contains 256 entries, each pointing to 4K of storage

Enhanced DAT Feature

- **New hardware feature on IBM System z10**
- **Segment table entry**
  - Can still point to a page table mapping the 1M of memory (traditional DAT)
  - Can also set flags to indicate this entry points to 1M of contiguous storage (enhanced DAT)
- **Improves TLB usage if multiple pages (4K areas) are accessed within 1M of enhanced DAT mapped storage as the segment table entry information is cached in the TLB**

Enhanced DAT Support for z/TPF

- **APAR PJ36565**
- **z/TPF automatically determines if the processor supports enhanced DAT**
- **If supported, the following tables are mapped using enhanced DAT**
  - Control program (CP)
  - System work blocks (SWBs)
  - I/O control blocks (IOBs)
  - Virtual file access (VFA) tables and data buffers

What Does it All Mean?

- **System z hardware improvements coupled with software performance enhancements continue to make z/TPF a clear choice for mission critical applications**
  - Leverage your investment in existing applications
  - Excellent price performance for adding new functions or applications
- **z/TPF is well positioned for modern processor architectures (current and predicted)**

Summary

- **Memory cache improvements**
  - APAR PJ36371
- **New z/TPF scheduler**
  - APAR PJ35517
- **TLB improvements**
  - APARs PJ32288 and PJ35230
- **Enhanced DAT**
  - APAR PJ36565

Trademarks

- IBM, System z9, and System z10 are trademarks of International Business Machines Corporation in the United States, other countries, or both.

- Intel, Intel Inside (logos), MMX, Celeron, Intel Centrino, Intel Xeon, Itanium, Pentium and Pentium III Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

- 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.
- 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.

- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.