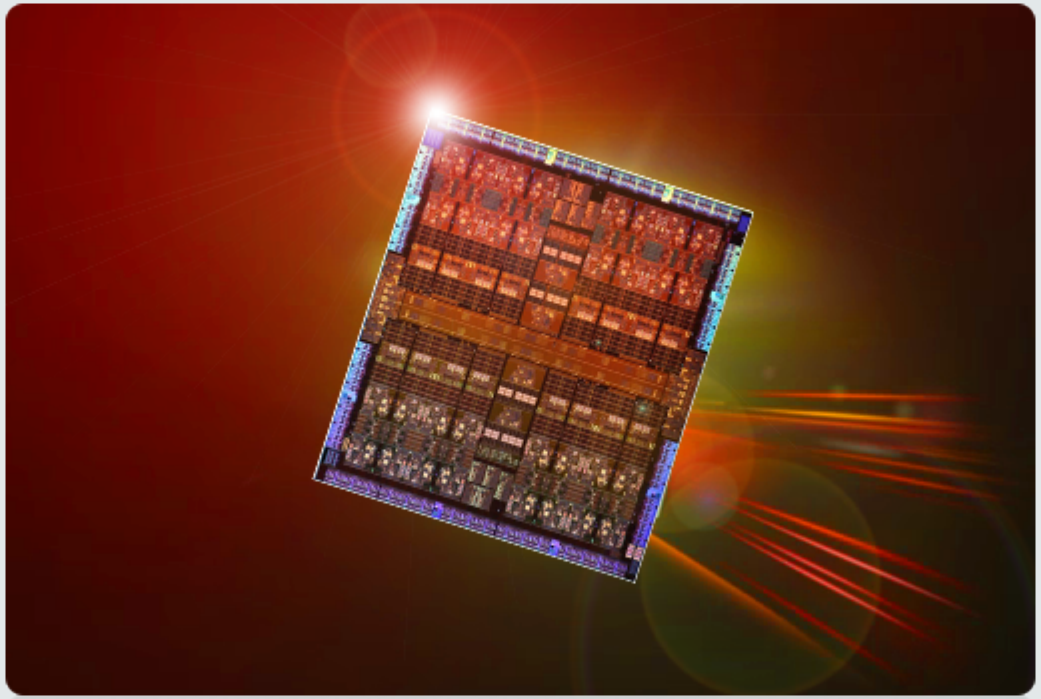
Next Generation SPARC Processor Cache Hierarchy

Ram Sivaramakrishnan
Hardware Director

Sumti Jairath
Sr. Hardware Architect

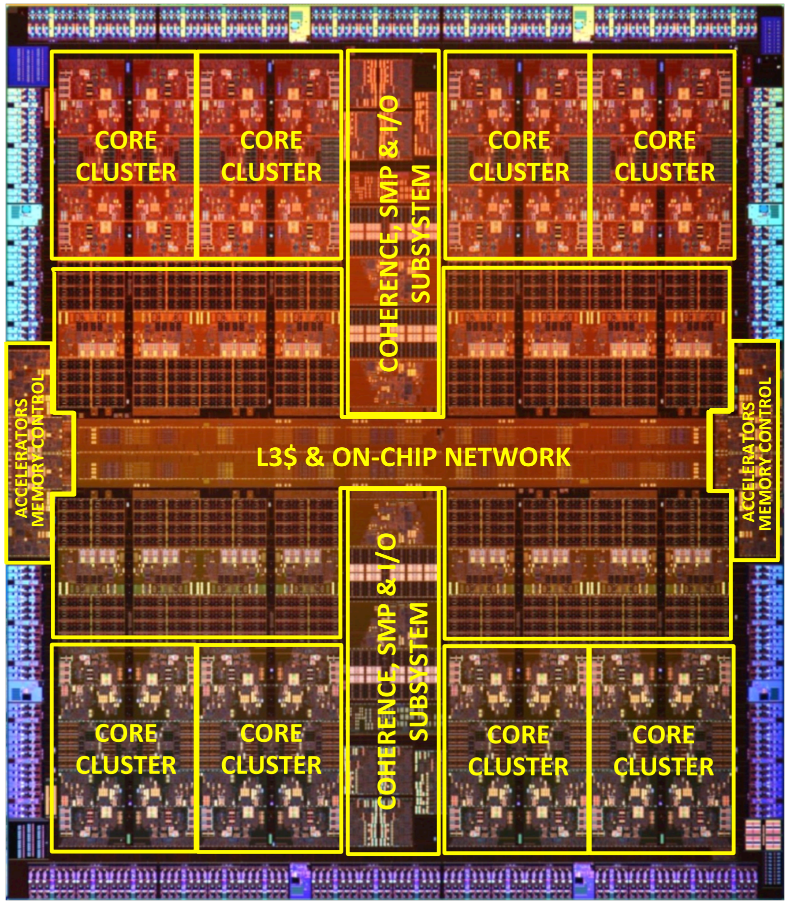


Safe Harbor Statement

Presentation Outline

- Overview
- Partitioned L3 Cache- Design Objectives
- Core Cluster and L3 Partition
- On-Chip Network and SMP Coherence
- On-Chip Protocol Features
- Workload Performance Optimization
- Conclusion

Processor Overview



The next generation SPARC processor contains

- 32 cores organized as 8 core clusters
- 64MB L3 cache
- 8 DDR4 memory schedulers (MCU) providing 160GBps of sustained memory bandwidth
- A coherency subsystem for 1 to 32 socket scaling
- A coherency and IO interconnect that provides >336GBps of peak bandwidth
- 8 Data Analytics Accelerators (DAX) for query acceleration and messaging

Why a Partitioned Last Level Cache?

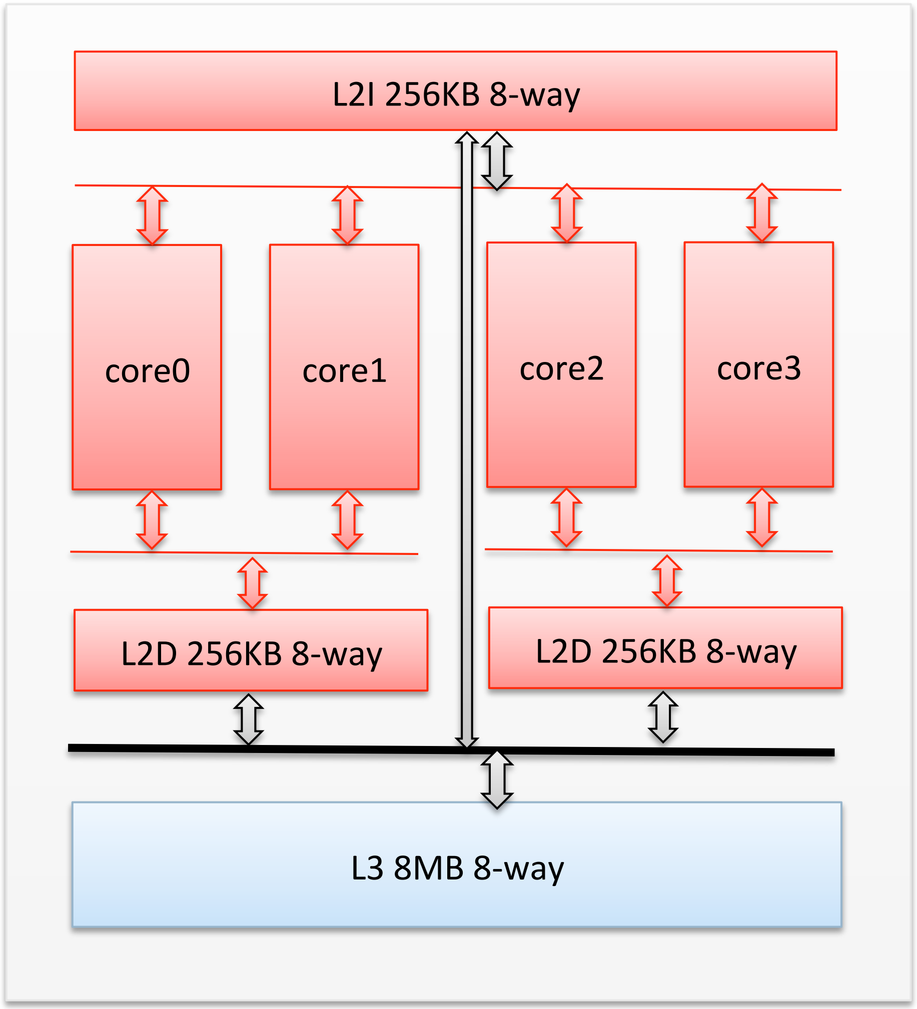
- A Logically Unified Last Level Cache
  - Latency scales up with core count and cache size
  - Benefits large shared workloads
  - Workloads with little or no sharing see the impact of longer latency
- Partitioned Shared Last Level Cache
  - A core can allocate only in its local partition
  - A thread sees a lower access latency to its local partition
  - Low latency sharing between partitions makes them appear as a larger shared cache

Last Level Cache Design Objectives

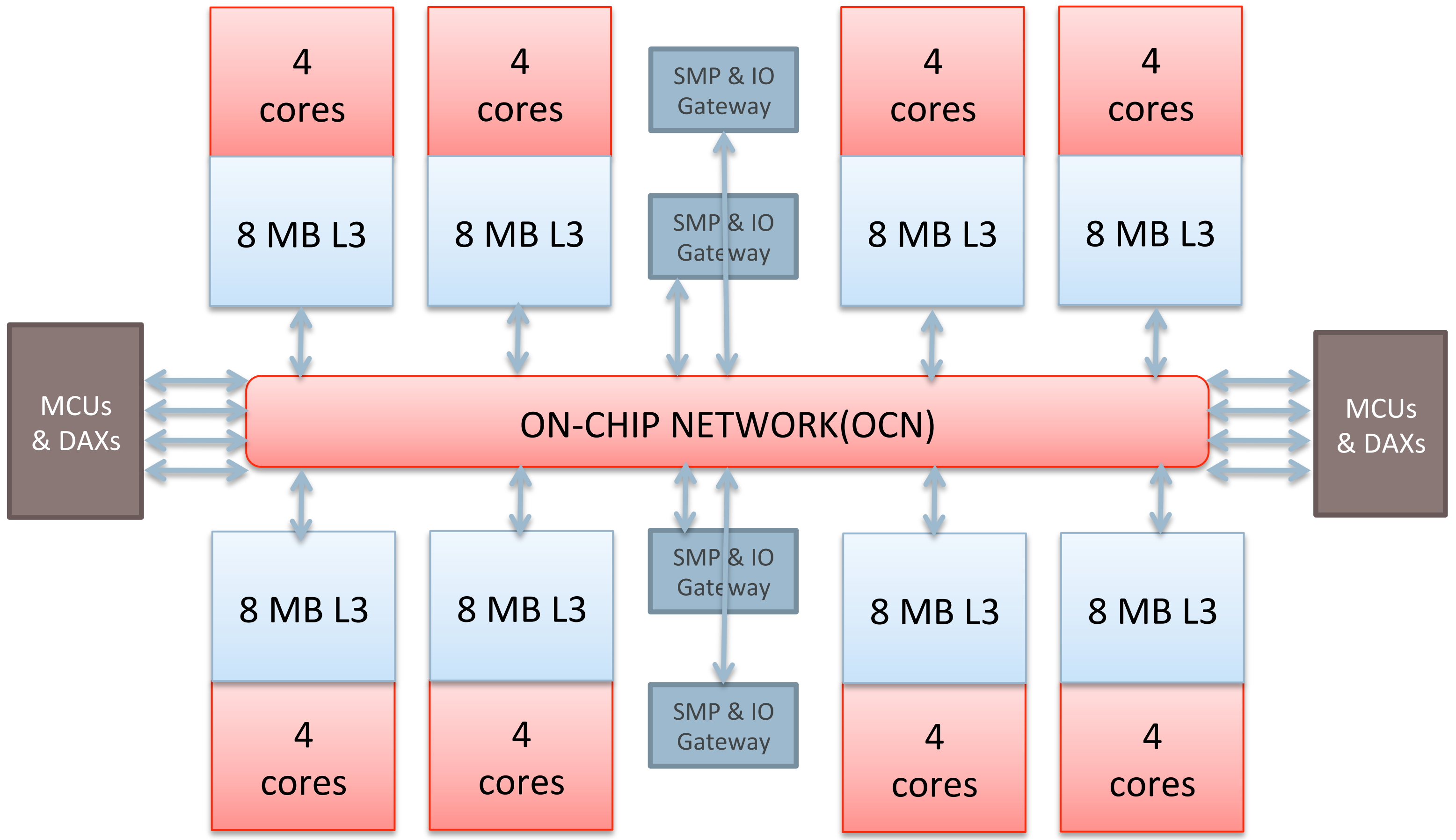
- Minimize the effective L2 cache miss latency for response time critical applications
- Provide an on-chip interconnect for low latency inter-partition communication
- Provide protocol features that enable the collection of partitions to appear as a larger shared cache

Core Cluster and Partition Overview

- Four S4 SPARC cores
  - 2-issue OOO pipeline and 8 strands per core
  - 16KB L1 I$ and a 16KB write-through L1 D$ per core
- 256KB 8-way L2 I$ shared by all four cores
- 256KB 8-way dual-banked write-back L2 D$ shared by a core pair
- Dual-banked 8MB 8-way L3$
- 41 cycle (<10ns) load to use latency to the L3
- >192GBps interface between a core cluster and its local L3 partition



Processor Block Diagram



- An On-Chip Network (OCN) that connects
  - 8 partitions
  - 4 SMP& IO Gateways
  - 4 scheduler and 4 DAX instances on each side
- 64GBps per OCN port per direction
- Low latency cross-partition access

Salient Features of the L3

- Supplier state per cache line to prevent multiple sharing partitions from sourcing a line
- Provides tracking of application request history for optimal use of the on-chip protocol features
- Idle partitions can be used as a victim cache by other active partitions
- Allows the DAX to directly deposit the results of a data analytics query or a message to be used by a consuming thread
- Pointer version per cache line for application data protection

On-Chip Network(OCN)

The OCN consists of three networks

- Requests are broadcast on four address planed rings, one per SMP gateway on chip
  - No queuing on the ring, maximum hop count is 11
- The Data network is constructed as a mesh with 10-ported switches
  - Provides ~512GBps per cross-section
  - Unloaded latency <6ns
- The Response network is a point to point network that aggregates snoop responses from all partitions before delivery to the requester

SMP Coherence

- Distributed Directory based SMP coherence for 8-way glue-less and 32-way glued scaling
- Directory is implemented as an inclusive tag rather than an L3 mirror
  - Reduces the needed associativity in the tag
- Inclusive tag sized at 80MB, 20-way set associative to minimize distributed cache evictions for throughput workloads
- Dynamic organization supports all socket counts from 1-8 in glue-less systems

ORACLE

OCN Protocol

The OCN protocol supports

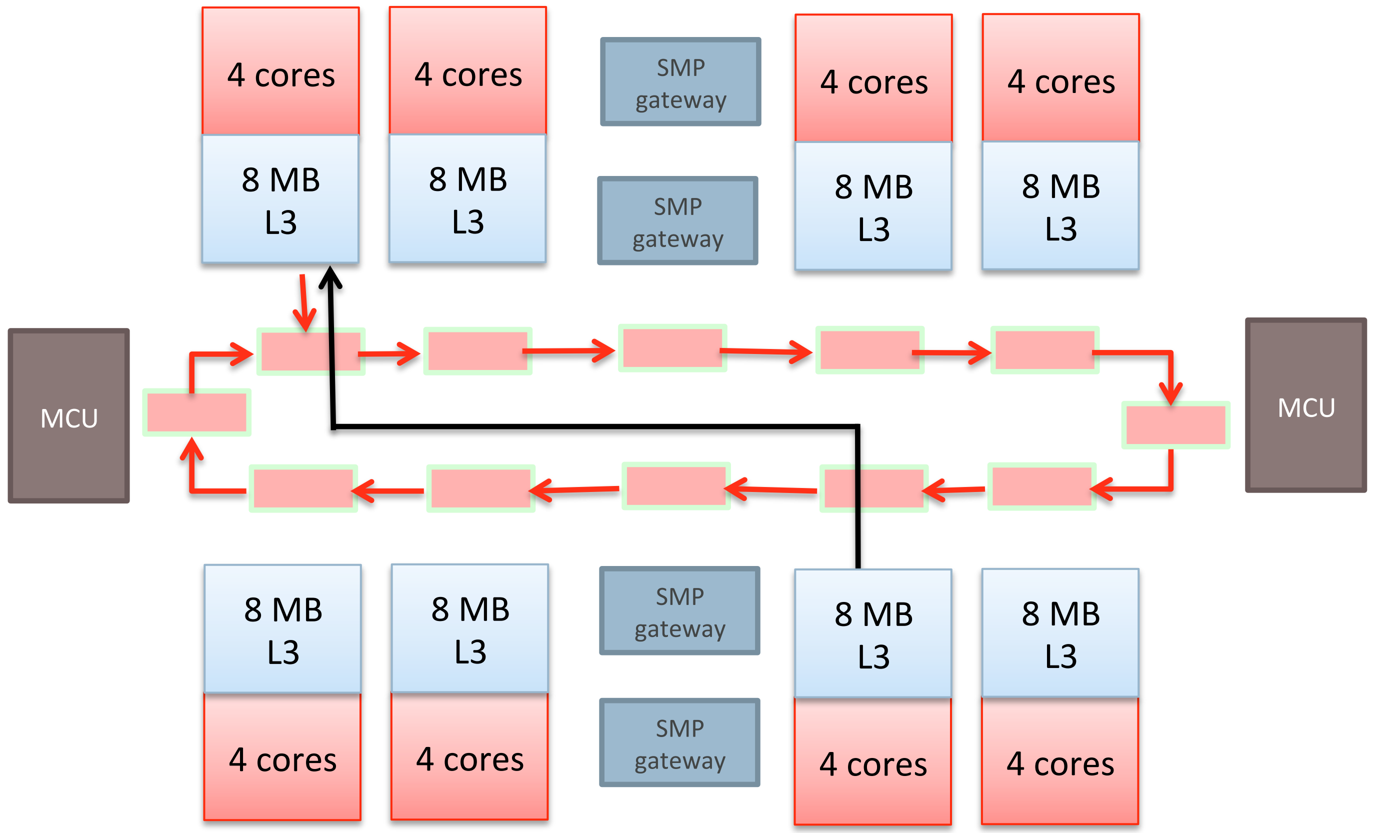
- Mediated and Peer-to-Peer Requests, Memory Prefetch
  - Protocol choice based on sharing characteristics of a workload
- Dynamically Partitioned Cache
  - Fair re-distribution of last level cache among varying number of active threads
- DAX Allocation
  - Coherent communication protocol between core and DAX

Mediated and Peer-to-Peer Requests

Address space sharing characteristics of a workload

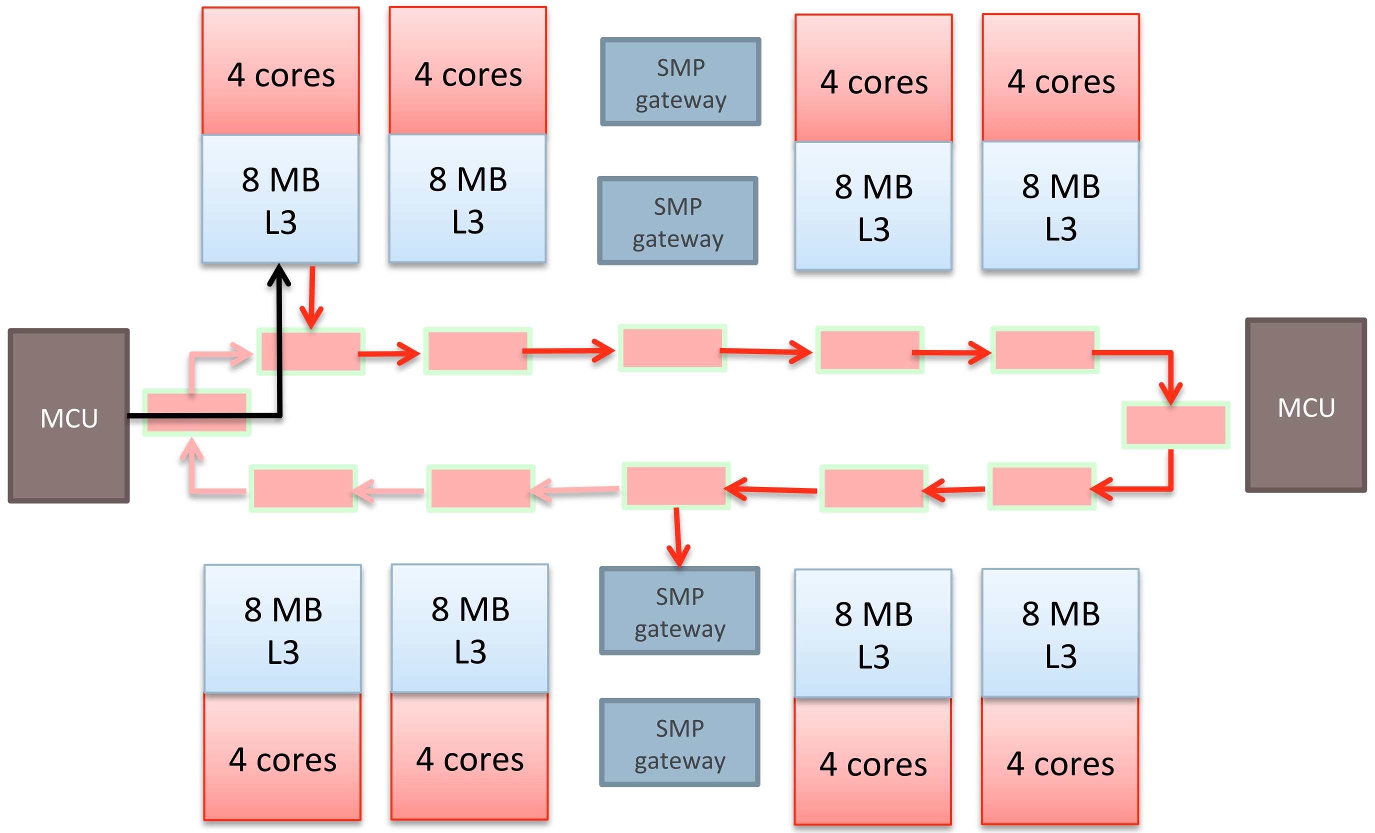
- Peer-to-Peer Requests
  - Broadcast to all partitions on the request network
  - Lowest latency path to get data from another partition (also referred to as C2C transfer)
- Mediated Requests
  - Unicast to coherence gateway. Other agents ignore the request
  - Is the fastest and most power efficient path to data from memory

Peer to Peer Request Flow



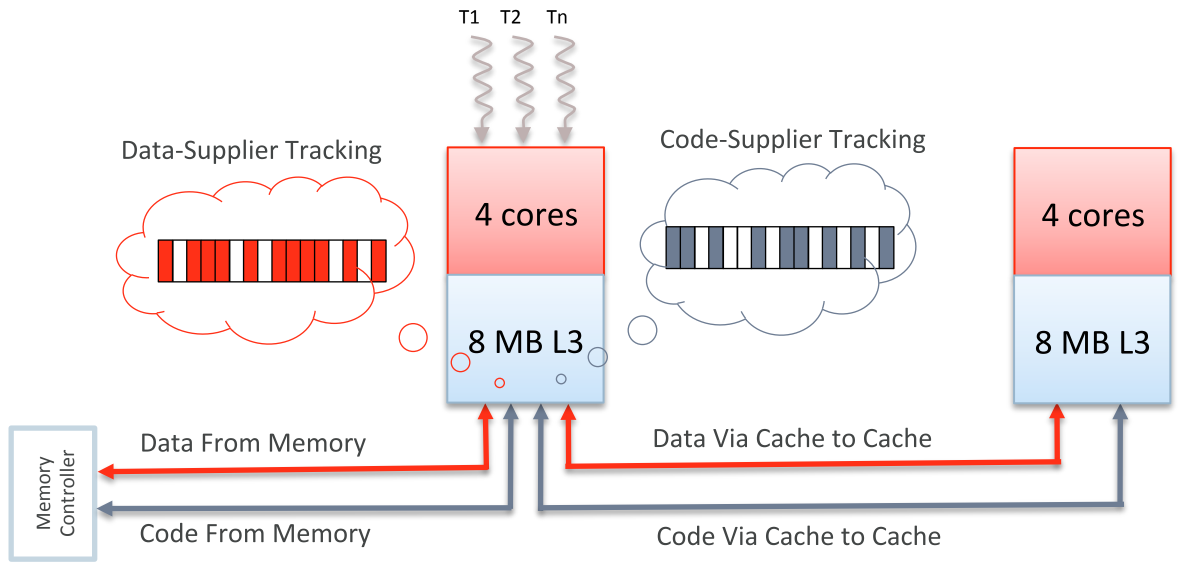
- L3 miss broadcast to all agents on the ring
- All L3 partitions lookup tag and respond with an Ack/Nack
- Supplier Partition sources data
- Lowest latency cross-partition transfer
- Turns into a mediated request on failure

Mediated Request Flow



- Unicast to SMP gateway
- SMP directory lookup and Memory Prefetch
- Memory read sent on the request network to the appropriate MCU instance (not shown)
- Memory provides data to the requester
- Lowest latency path to memory

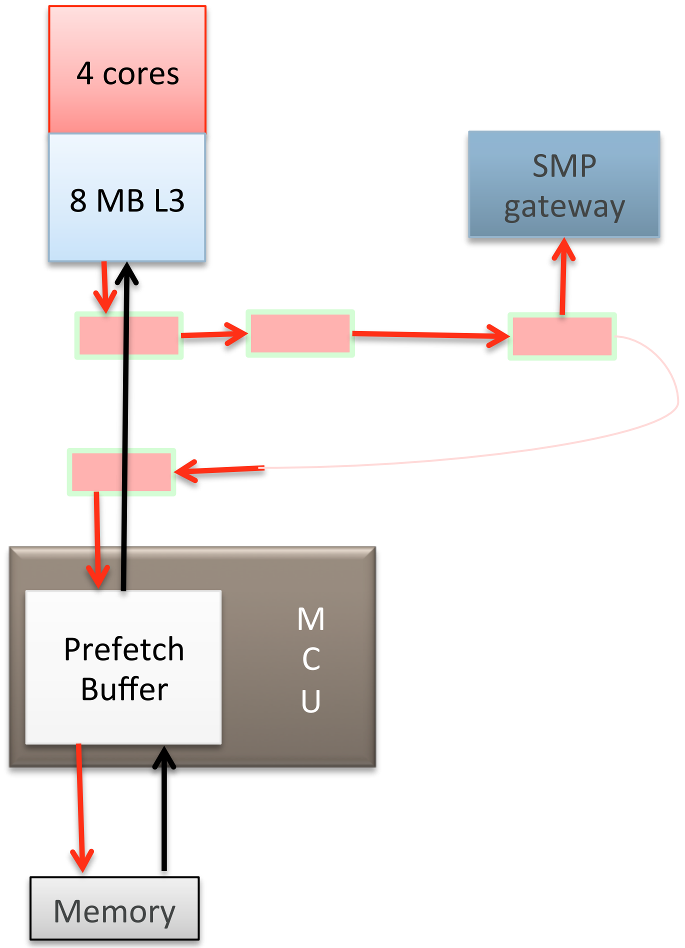
Dynamic Request



Cache-Line Supplier Tracking

- Tracks how its L3 misses are served and determines if a mediated or peer-to-peer request is appropriate.
- High cache-to-cache rate drives peer-to-peer requests
- High memory return drives mediated requests
- Temporal sharing behavior is effectively tracked using a 64-bit history register
- Code and Data are tracked independently.

Memory Prefetch



- L3 miss partition can request for a speculative memory read
- Supported for peer-to-peer and mediated requests
- Data returning from memory is preserved in a 64 entry (256 entries per chip) buffer until a memory read is received
- Allows for faster memory latency for workloads with low sharing
- Is dynamically enabled using code and data source tracking

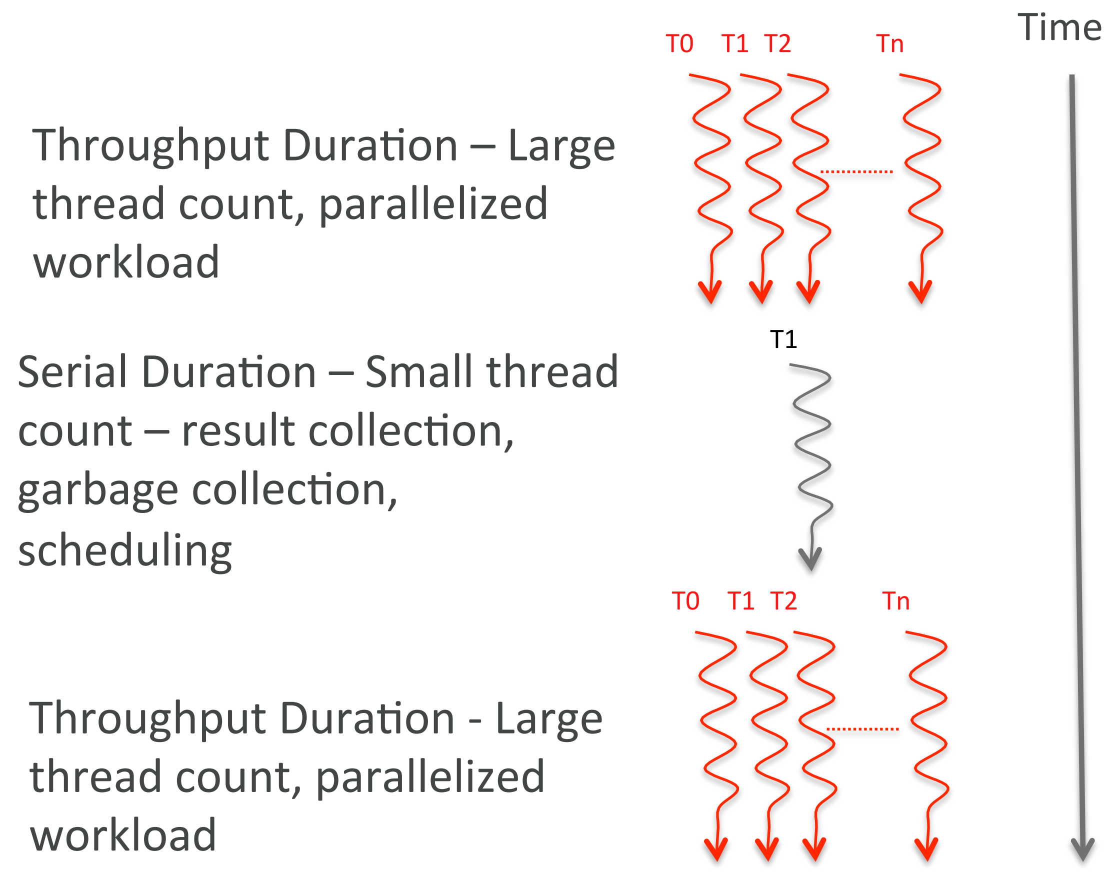
Workload Caching Characteristics

Workload Examples	Instructions	Data	Data Source	Performance Needs
Individual Programs, Virtual Machines	Not Shared	Not Shared	Instructions and Data from Memory	• Instruction and Data - Memory Prefetch. • Predictable QoS
Multiple Programs Working on Common Data – e.g. Database Shared Global Area (SGA)	Not Shared	Shared	Instructions from Memory, some Data as C2C	• Instruction Memory Prefetch • Fast Data C2C
Partitioned Data Processing – e.g. Analytics	Shared	Not Shared	Instructions as C2C, Data from Memory	• Fast Instruction C2C • Data Memory Prefetch • High Bandwidth Low-Latency LLC Hits • QoS - Predictability
Linked Lists, Control Data Structure	Shared	Shared	Both Instructions and Data as C2C	• Fast Instruction and Data C2C

Dynamically Partitioned Cache

- For optimal use of the cache under variable load, a partition can use another partition as a victim cache
- A victim partition can either accept or reject the request based on a heuristic

Dynamically Partitioned Cache

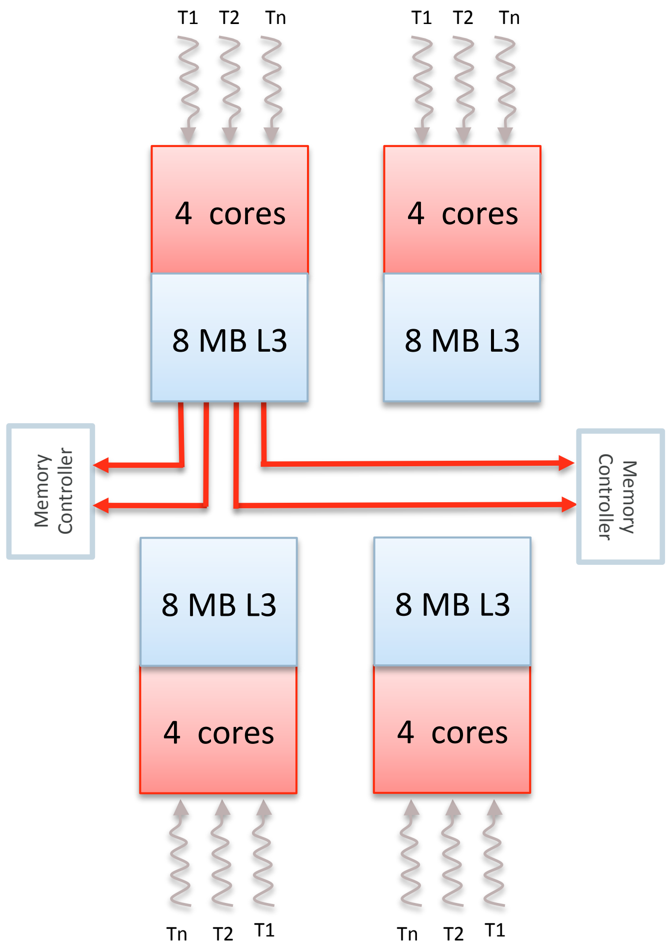


Time
T0 T1 T2 Tn
Throughput Duration – Large thread count, parallelized workload
T1
Serial Duration – Small thread count – result collection, garbage collection, scheduling
T0 T1 T2 Tn
Throughput Duration - Large thread count, parallelized workload

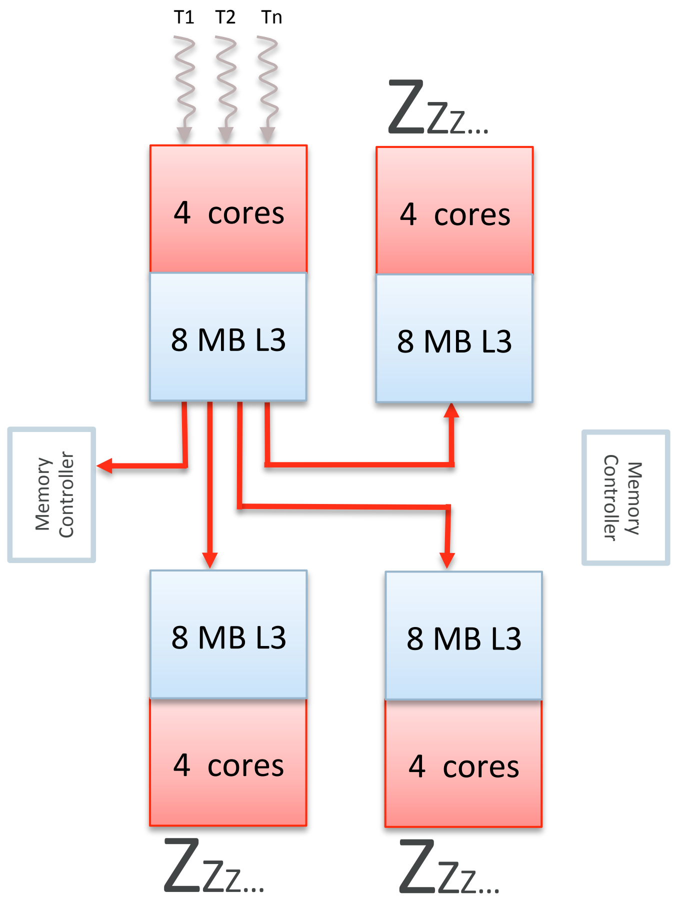
Workload Active-Threads Timeline

- Partitioned L3 Cache
  - Optimal for "Throughput Duration" of the workload
- Unified L3 Cache
  - Optimal for "Serial Duration" of the workload
- Dynamically Partitioned Cache
  - Adapts with workload behavior
  - L3 Partitions join or disjoin based on active threads in partition

Dynamically Partitioned Cache - Example



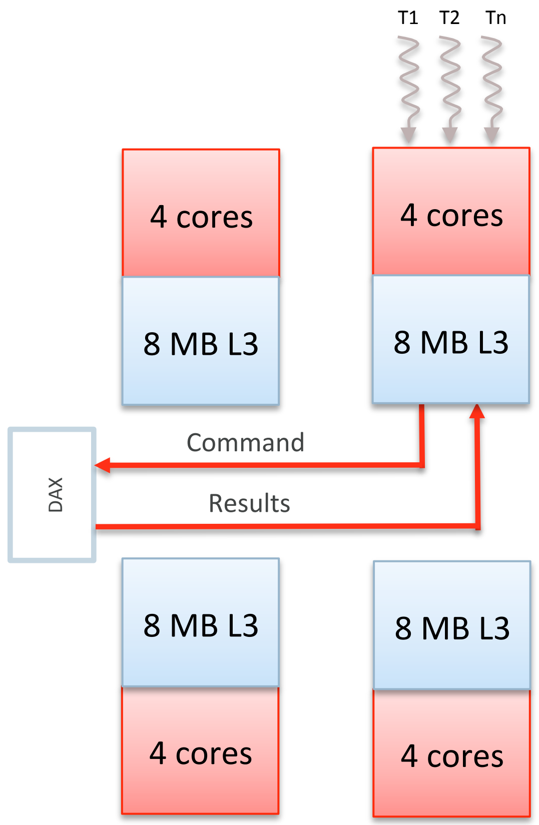
Throughput Duration



Serial Duration

- Active workload on a partition evicts cache lines.
- Evictions are absorbed by other idle partitions
- Increases Last-Level-Cache size for active threads.

Core-DAX Handshake

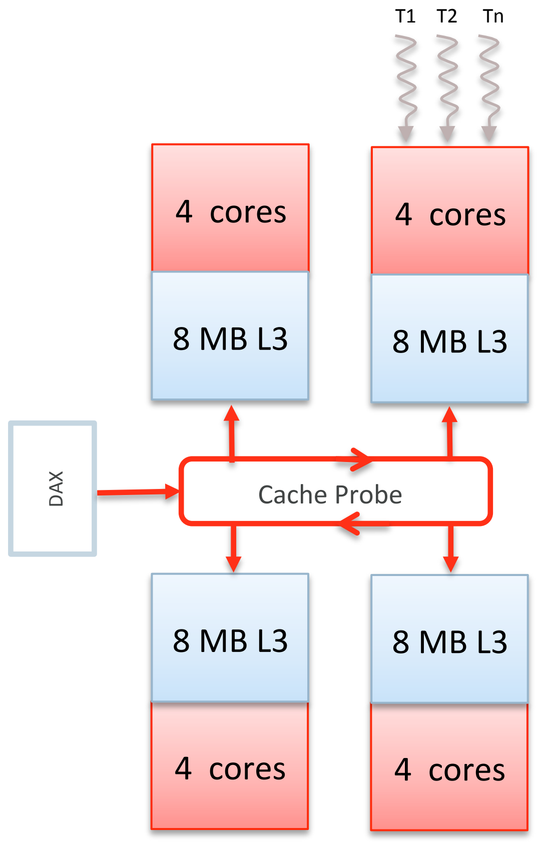


- Core-DAX handshake is cache coherent
- Data produced by cores can be read from the cache or memory by the DAX
- Results of the DAX operation are deposited in the cache or memory for consumption by core

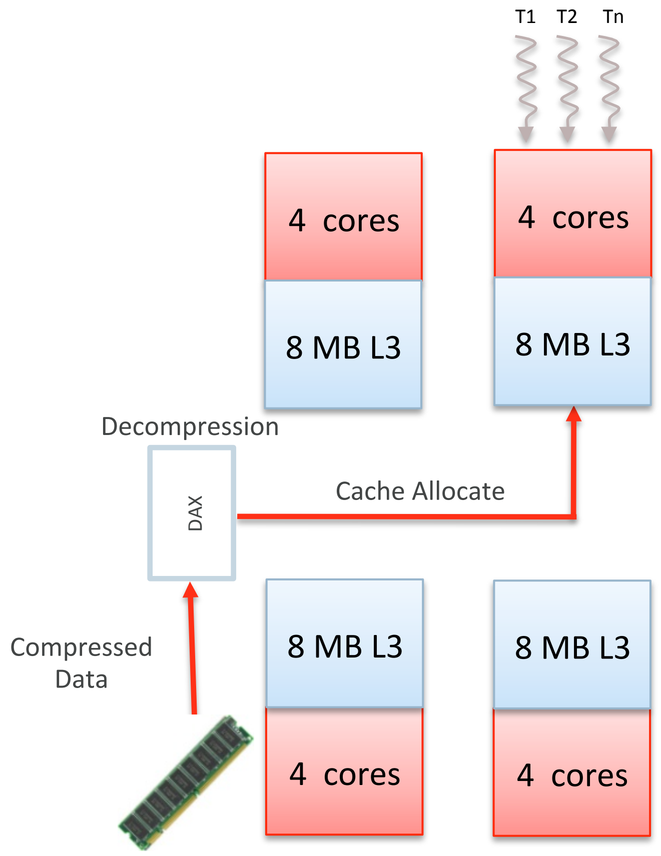
DAX Allocation

- The DAX can directly deposit messages and results of a query into any L3 partition
- DAX DMA buffers are reused in order to mitigate dirty evictions from the cache
  - A probe instruction is used to determine if the buffer was previously allocated in any partition
  - If so, that partition is chosen for allocation

Dynamic Core-DAX Pairing



Cache Probe



Cache Allocate

- DAX broadcasts a probe request to discover the recipient thread's partition.
- Results from the DAX are deposited in the recipient thread's partition.

Conclusion

The next generation SPARC processor cache hierarchy

- Significantly improves the SOC response time over the previous generation
- Provides a low latency, high bandwidth, dynamically partitioned cache
  - Actively tracks workload behavior
  - Applies the appropriate flavor of the on-chip protocol for optimal performance
- Enables a low latency, high bandwidth DAX interface for effective off-load of acceleration tasks.

Hardware and Software

Engineered to Work Together

ORACLE®