Workloads, Scalability and QoS Considerations in CMP Platforms

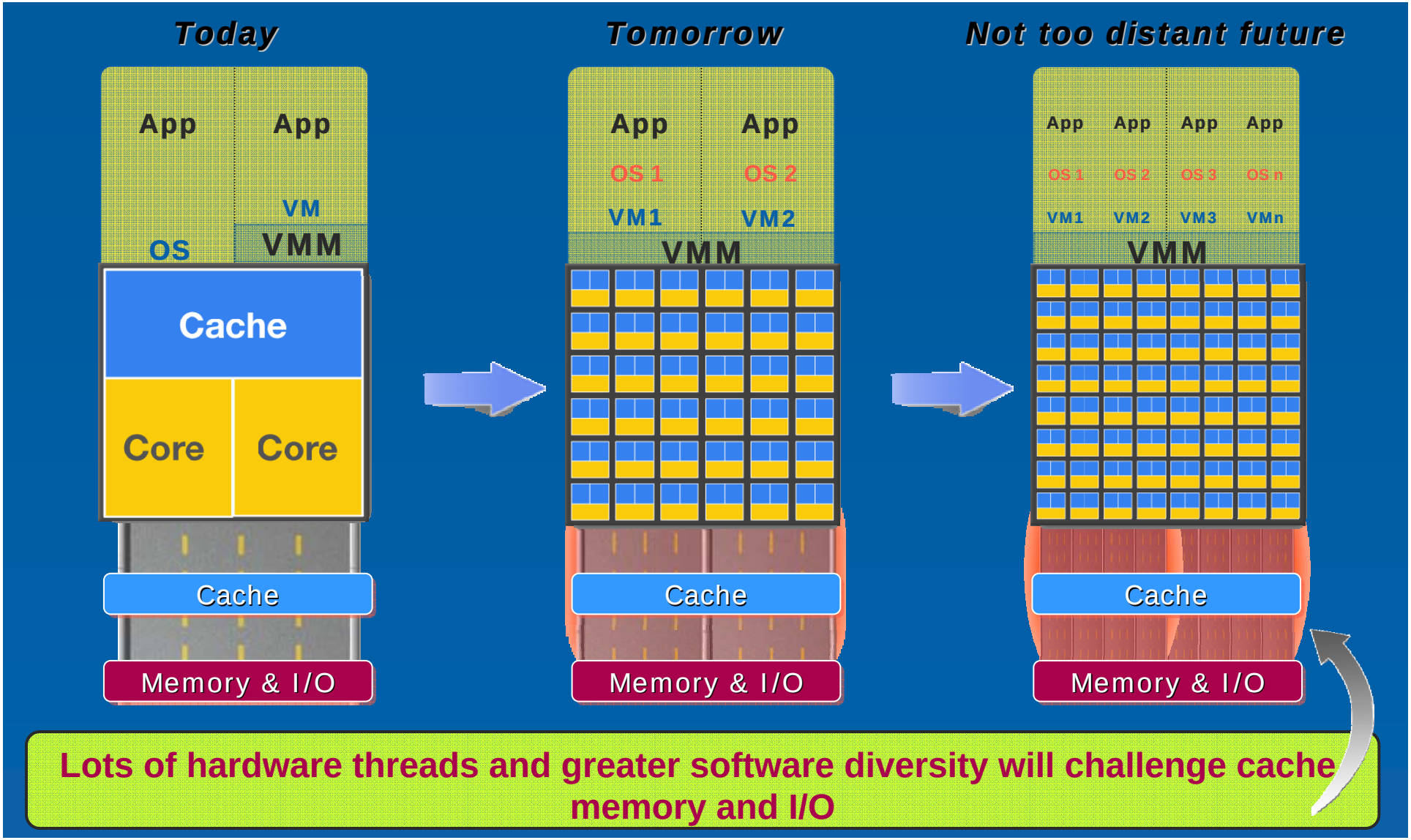
Presenter – Don Newell
Sr. Principal Engineer
Intel Corporation

Agenda

- Trends and research context
- Evolving Workload Scenarios
- Platform Scalability
- Platform QoS

Trends

- Cores, Threads, and Virtual Machines



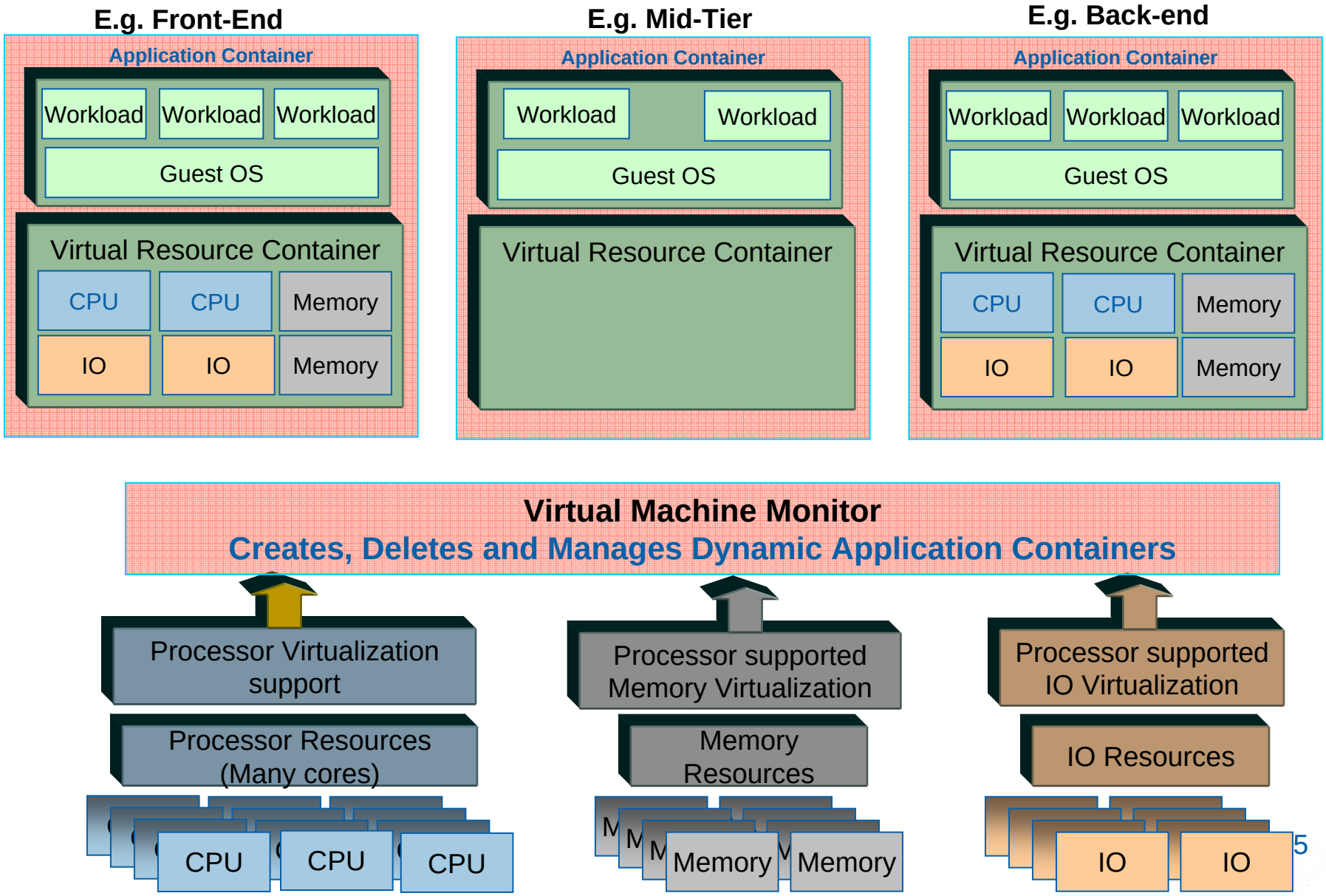
Trends - Discussion Points

Goal: Scaling up to 100's of logical processors on a single CPU die

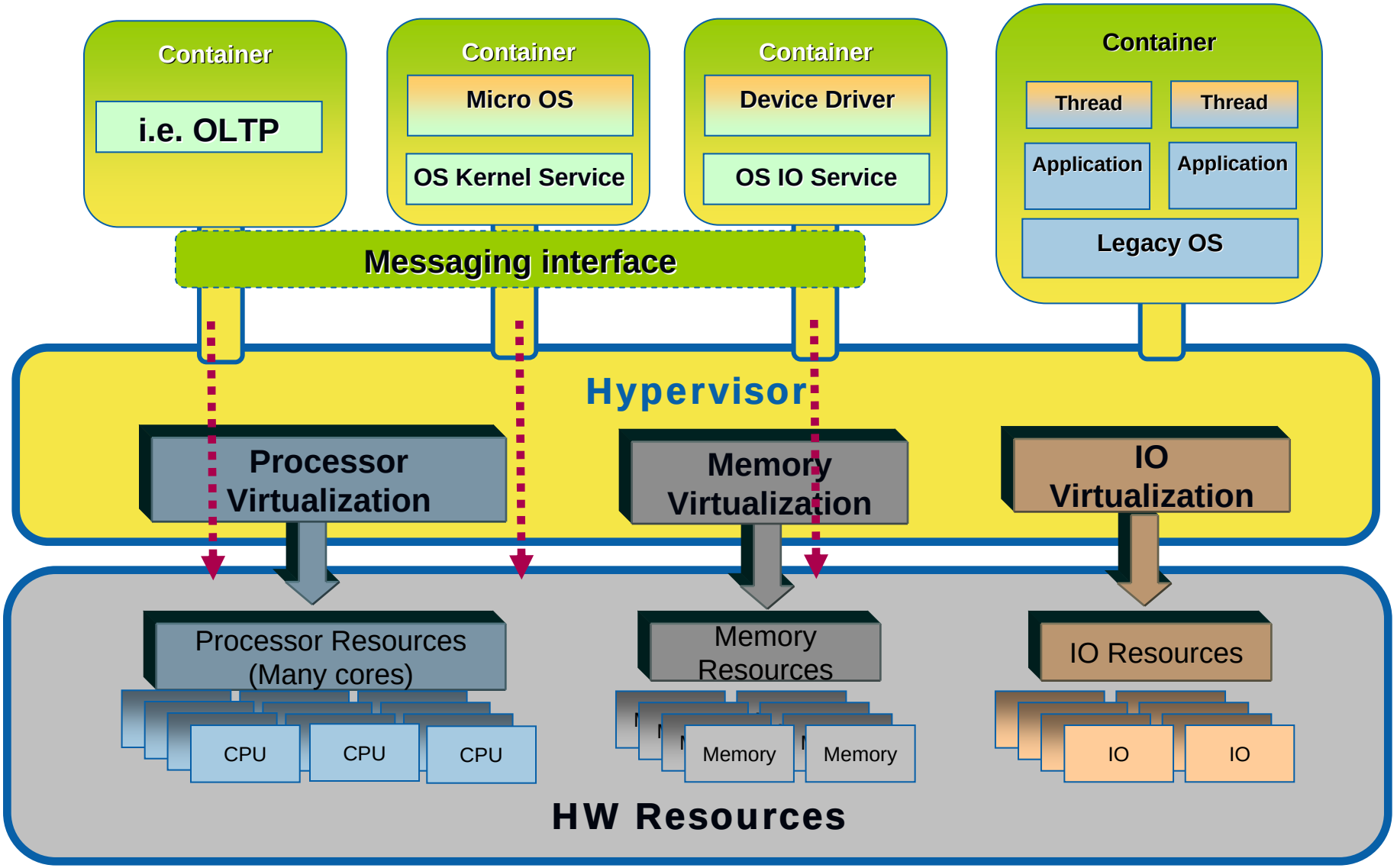
- Scaling hardware threads means provisioning and re-architecting other platform resources

- Scaling hardware threads means learning to share – QoS revisited

Trends - Pool of Virtual Resources

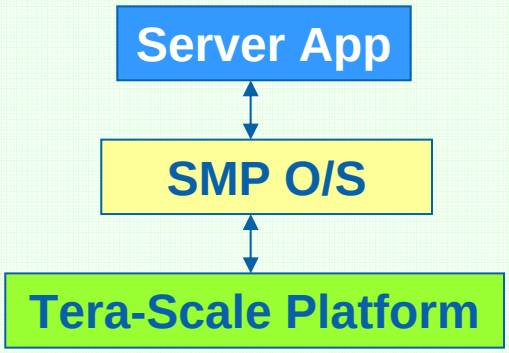


Trends - Decomposed OS

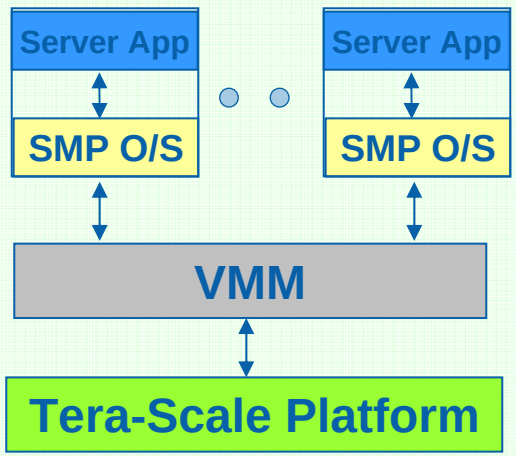


Platform Scaling

Tera-Scale Workload Scenarios



Highly Multi-threaded Server Workloads
OLTP, J2EE App Servers, E-commerce, ERP, Search, etc
(TPC-C, SPECjbb2005, SAP, SPECjappserver2004, etc)



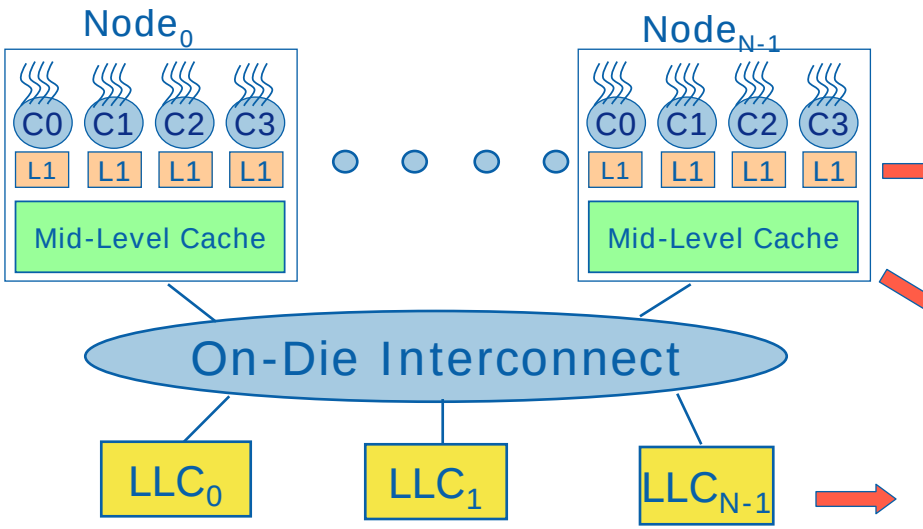
Virtualized Server Environments
(Workload Consolidation, Datacenter-on-chip Scenarios)

Cache / Memory Considerations:

Performance	**=> Overall Throughput**
Scalability	**=> Headroom**
Quality of Service	**=> Performance Isolation**

Let's begin with single server app……..
And follow up with the consolidated scenario…….

Tera-Scale Cache Hierarchy Design



Small L1 shared between multiple threads on a core

Moderate L2 shared between cores within the node

Distributed L3 shared by all nodes within the socket

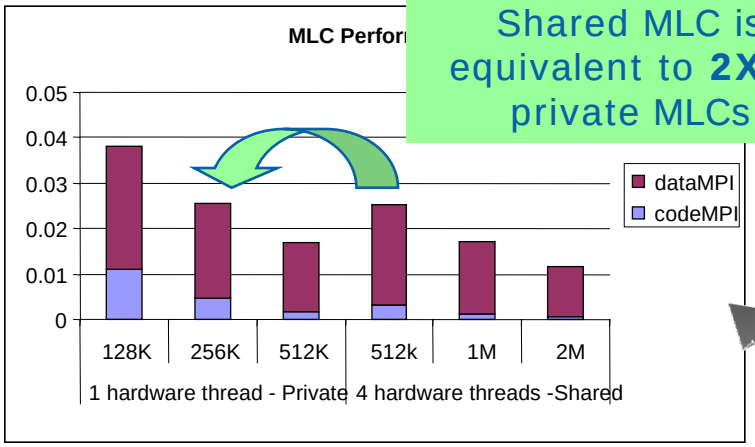
Hierarchical Sharing Benefits

-- Reduces Replication

-- Increases Effective Cache Size

-- Better localized communication

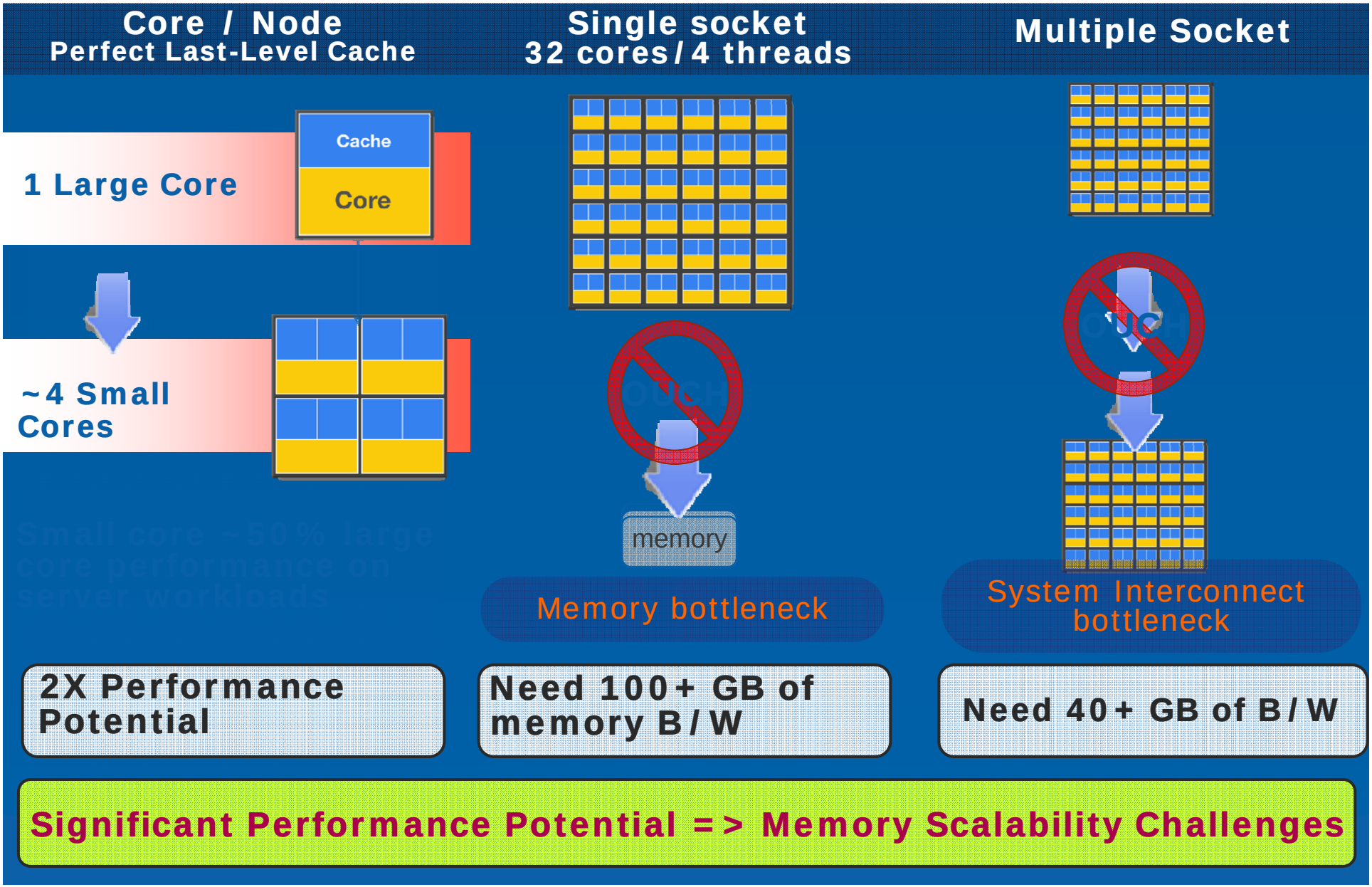
-- Reduced interconnect pressure



Shared MLC is equivalent to **2X** of private MLCs

Hierarchical sharing increases caching effectiveness significantly

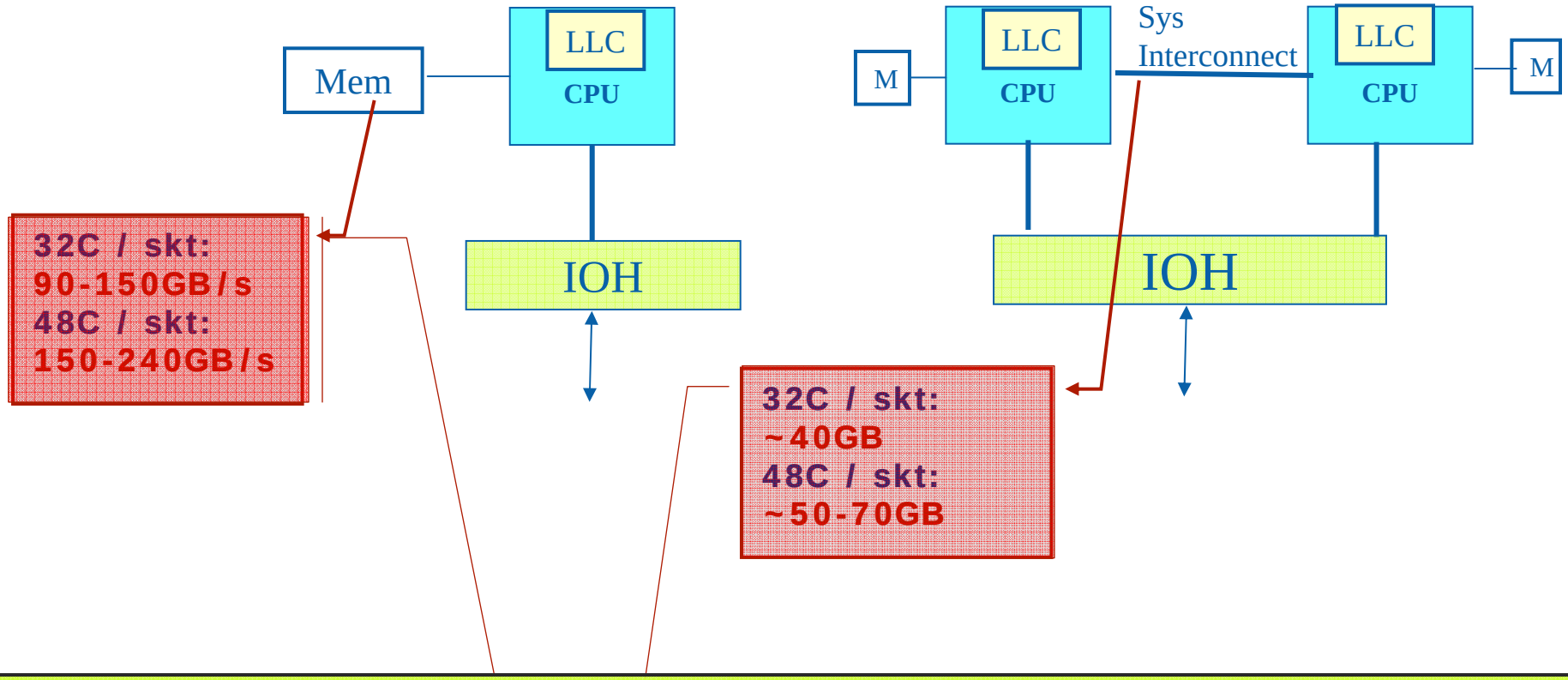
OLTP Tera-Scale Case Study



Scalability Issues

Tera-scale Headroom requirements

- Start with 32 cores in 1st gen; Maybe grow to 48 cores in next gen?
- How much memory & interconnect bandwidth will we need?



Scaling Issues Just Get Worse

On-Socket DRAM Caches
(For Memory Scalability)

Enable Large Capacity L4s

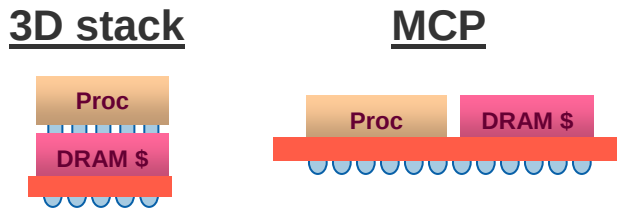
- Low Latency
- High Bandwidth

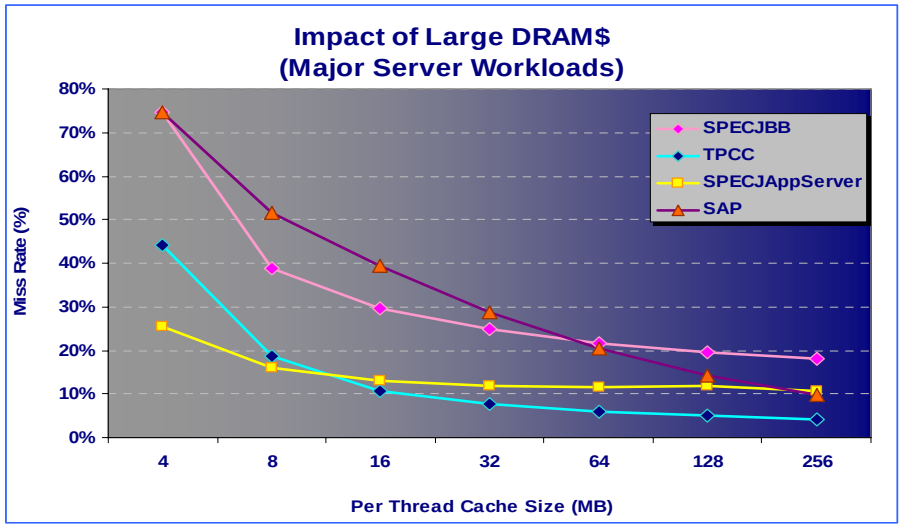
Technologies

- 3D Stacking
- Multi-chip Packages (MCP)

Benefits

- Significant reduction in miss rate
- Avoids bandwidth wall





QoS and Performance Management

Background for QoS Discussion

Trends

- Increasing Core Count for Performance
- Increasing Workload Diversity
  - *Multi-Workload Scenarios in the Client*
  - *Virtualization and Consolidation in the Server*
  - *Heterogeneous Architectures for Graphics*

Observations

- Multi-core enables simultaneous execution of multiple workloads
- But not all Workloads are equal -- users do have preferences

How well does the user-preferred application run?
Should platforms optimize for the user-preferred application?

Resource Management

Capitalist

- No management of resources
- If you can generate more requests, you will use more resources
- Grab as you will
- ***E.g. All of today's policies***

Communist/Fair

- Fair distribution of resources
- Give equal share of resources to all executing threads
- Does not necessarily guarantee equal performance
- ***E.g. Partitioning resources for fairness and isolation***

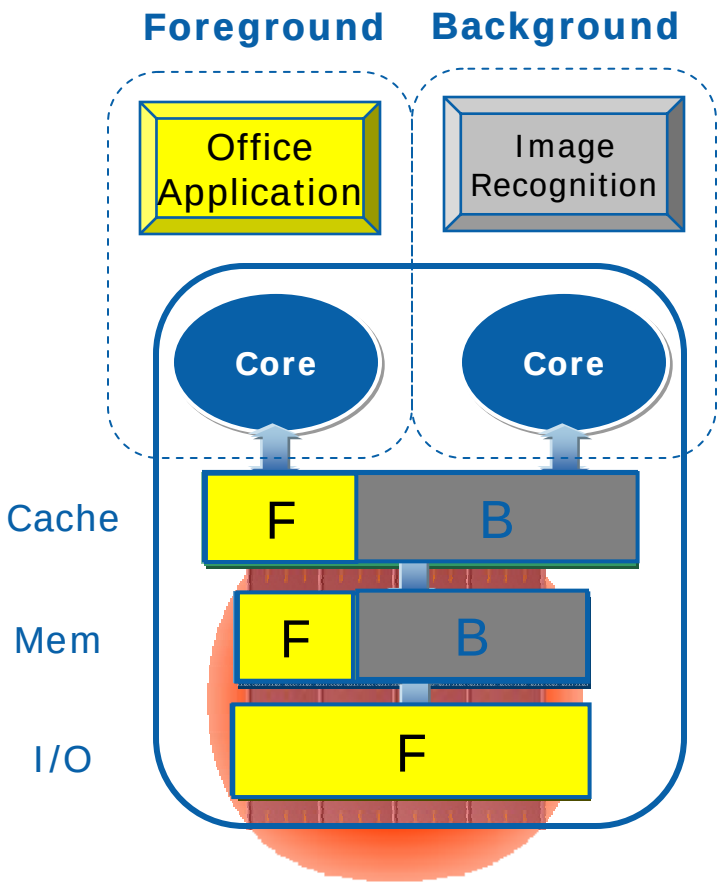
Elitist

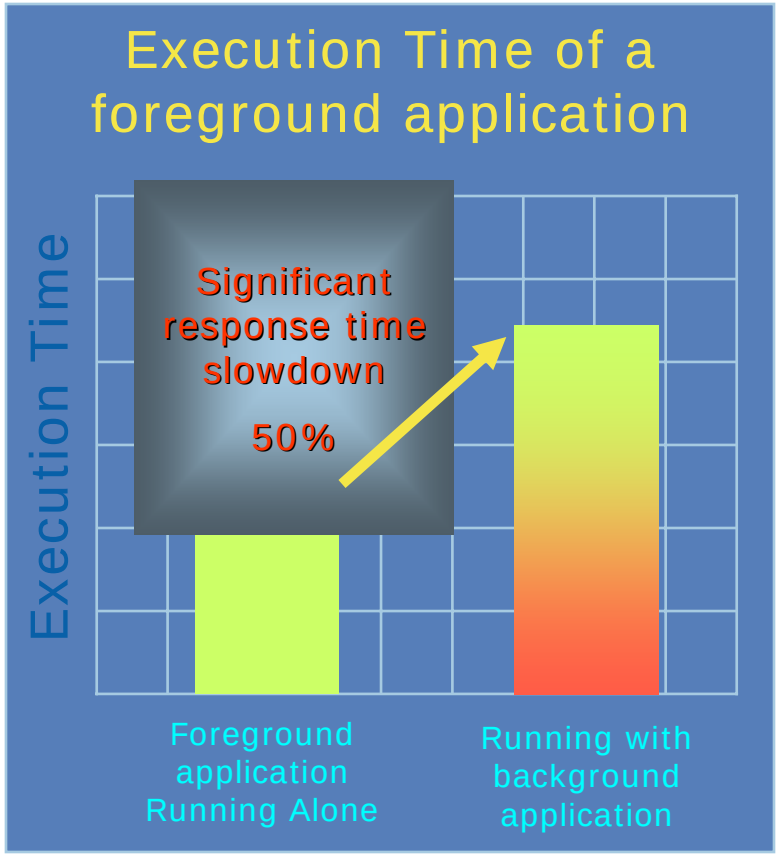
- Focus on individual efficiency
- Provide more performance and resources to the VIP
- Limited resources to non-VIP
- ***E.g. Service Level Agreements, Foreground/Background***

Utilitarian

- Focus on overall efficiency
- Give more resource to those that need it the most, less to others
- ***E.g. Cache-friendly vs. Unfriendly, resource-aware scheduling***

The Multi-Workload Problem





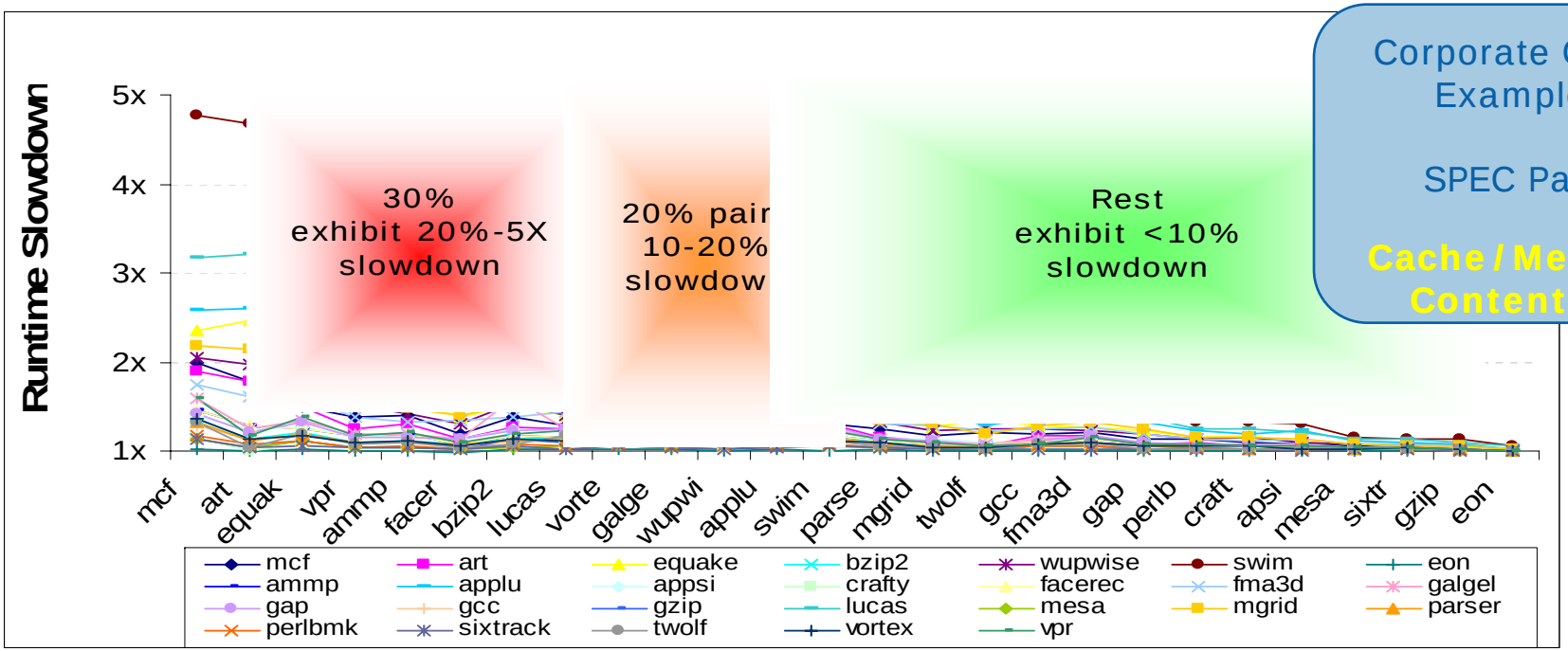
Conroe core 2 Duo Measurements

Platform does not distinguish in resource allocation

Preferred (foreground) application can suffer significant slow down

Platform QoS can improve user-preferred application performance

Contention - Client side examples



Measured on a Dual core Conroe (4M cache, DDR2 667)

Corporate Client Example:

SPEC Pairs

Cache / Memory Contention

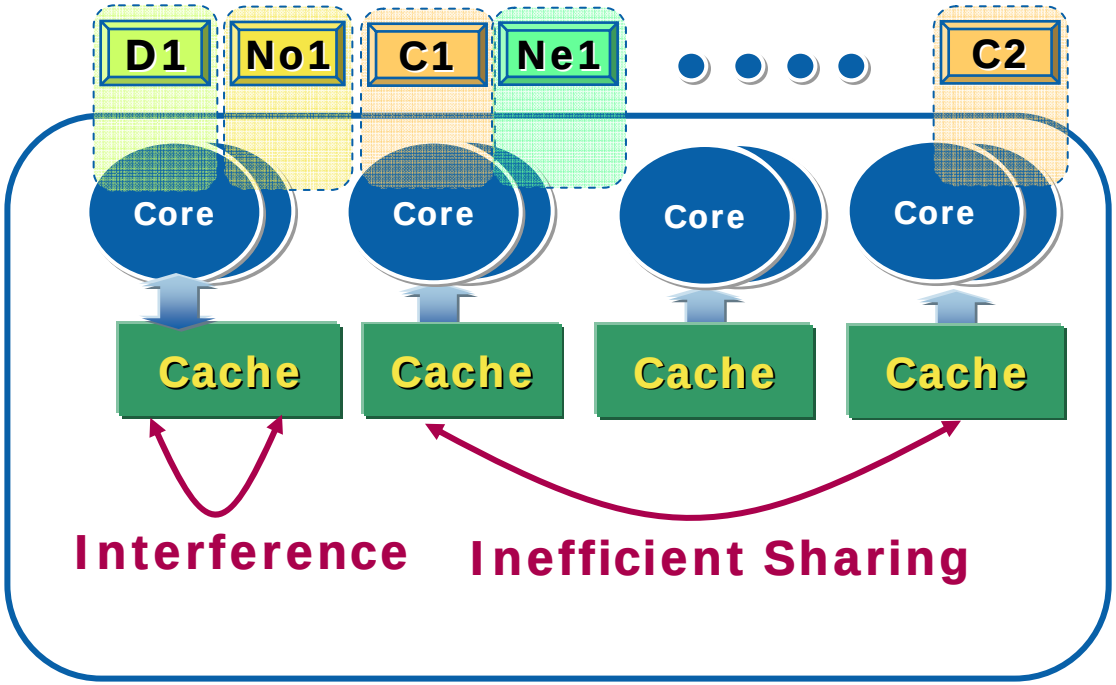
Similar data collected for server applications

Resource contention will impair the performance of important apps

(Performance Differentiation)

Application Behavior & Overall Performance

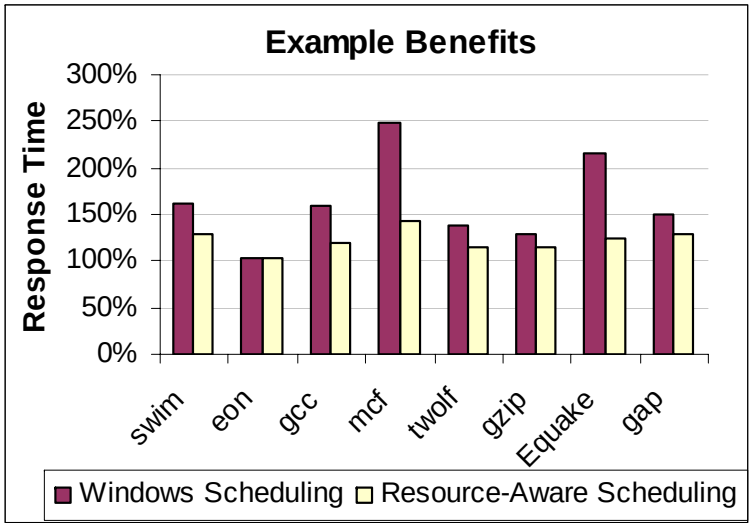
Destructive (streaming) **No**rmal (typical) **C**onstructive (co-op threads) **Ne**utral (little data)



Many Heterogeneous Applications

Potential to improve overall performance

Monitor resource usage and group/partition accordingly



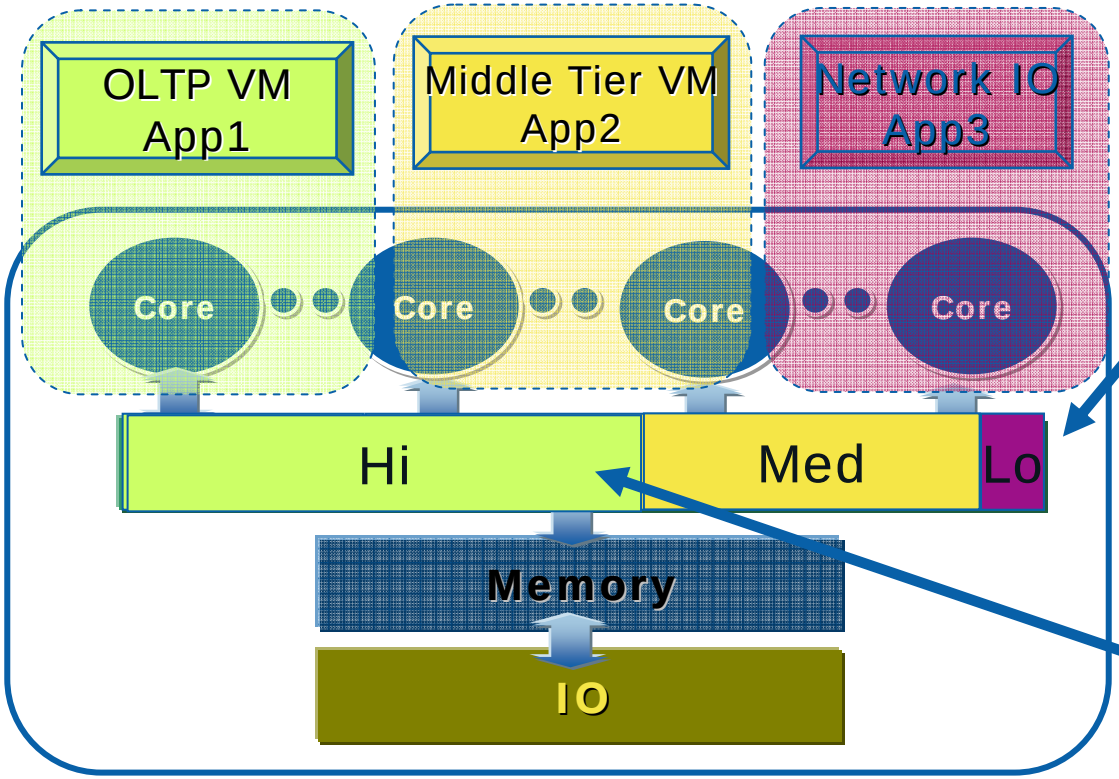
Clovertown (8Core / 8App) Experiments
(used destructive, normal and neutral apps)

Managing resource contention can improve overall throughput too

(Performance Management)

Service Level Agreements in the Enterprise

Server Consolidation



Many Different Virtual Machines

Utility SLA

Disparate resource usage can cause performance isolation concerns

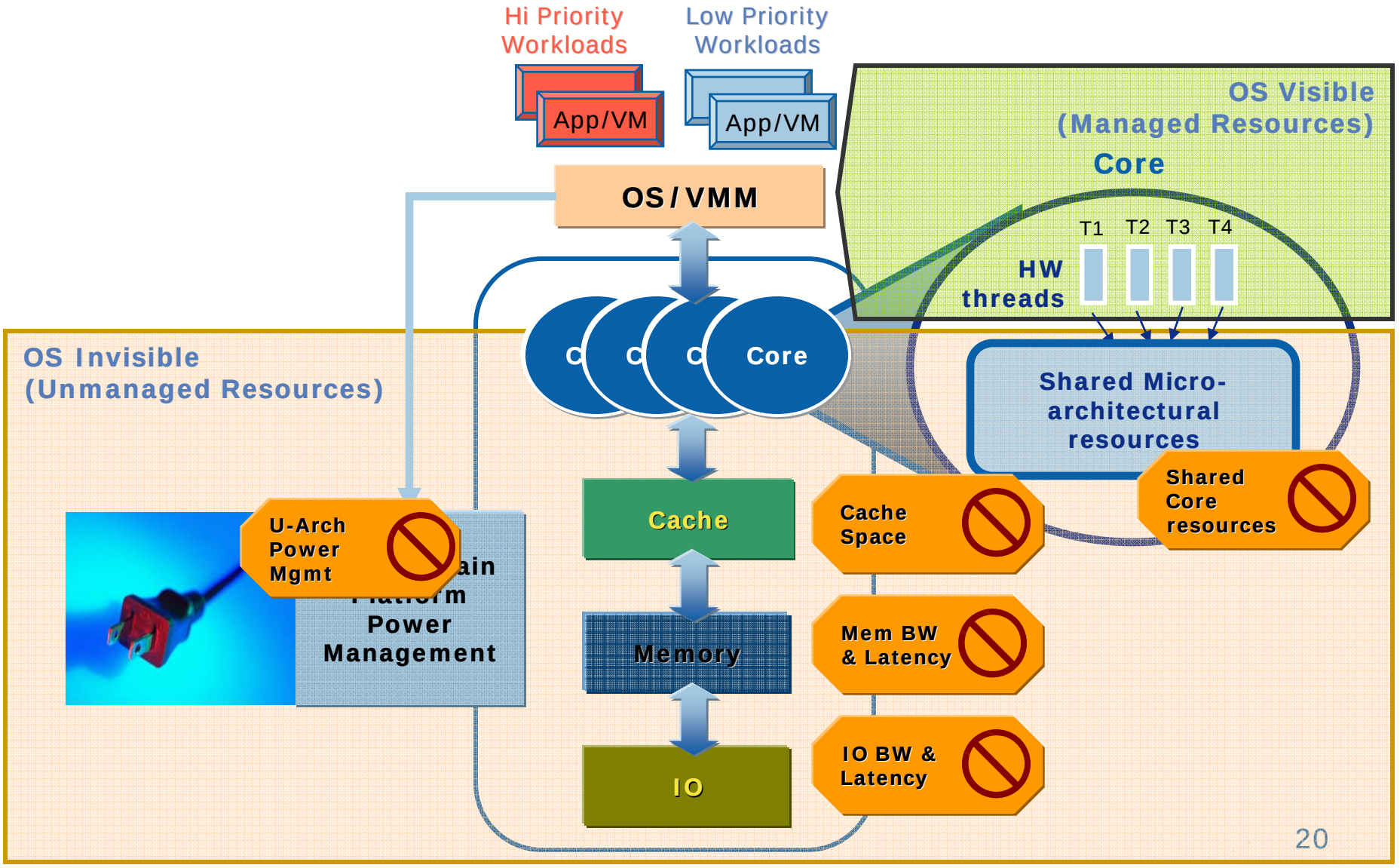
SLA management

Prioritized resource allocation can help address performance isolation and SLA

Disparate resource usage and contention hurts SLAs

(Need for Performance Isolation and SLA enforcement)

Shared Platform Resources



Problem Summary

- **CMP ➔ Many heterogeneous threads, apps, VMs**

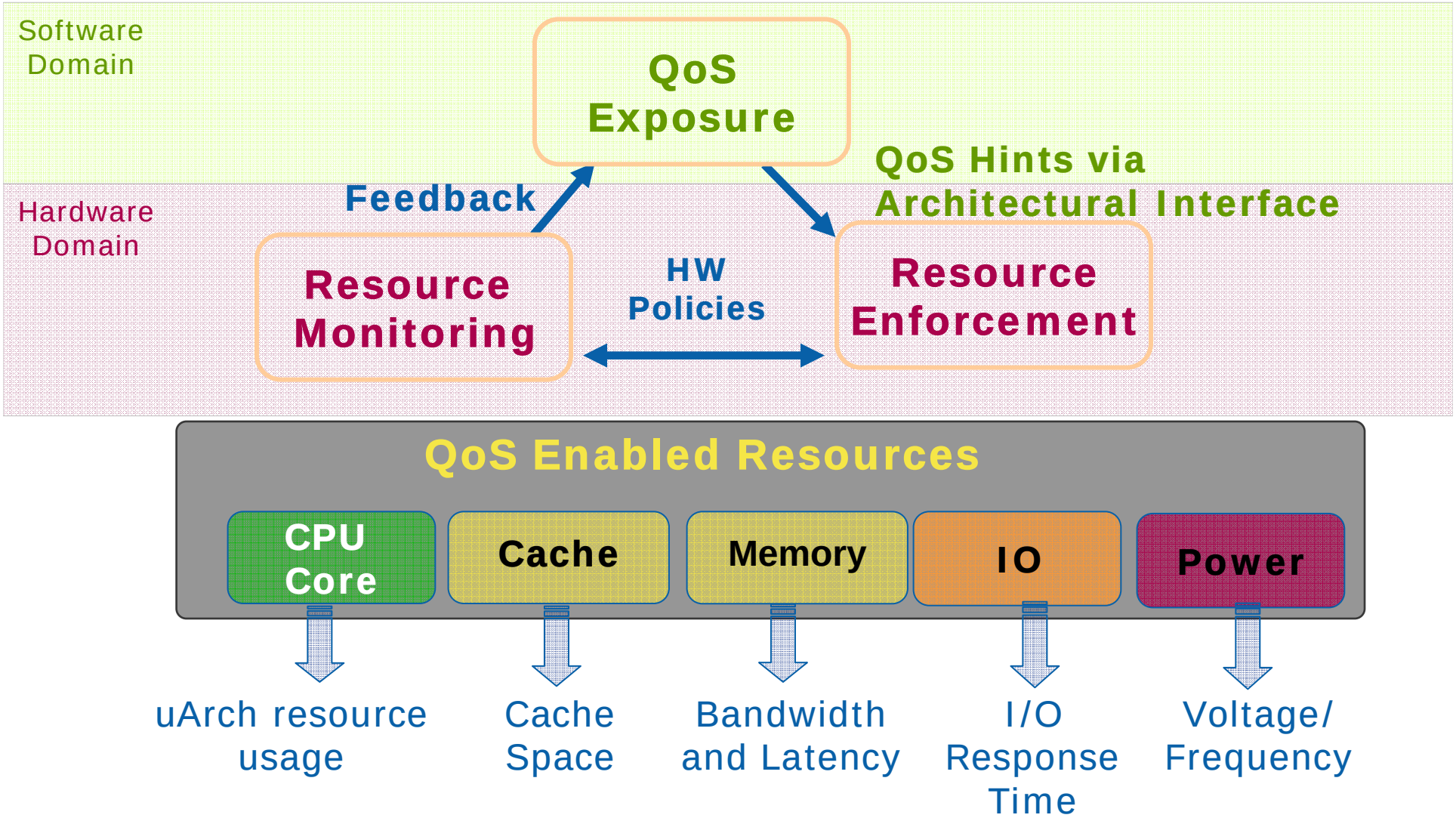
Not all applications are equal

- **Users have preferences**
  - End-users (client) want to treat foreground preferentially
  - End-users (server) want service level differentiation (SLA) or isolation
- **Applications use resources differently**
  - Destructive vs. Constructive vs. Neutral Threads
  - No performance management to protect from bad behavior

- **Priority-based OS scheduling no longer sufficient**
  - With more cores, OS will allow high and low priority applications to run simultaneously and contend for resources
  - Low priority applications will steal platform resources from high priority apps ➔ loss in performance & user experience

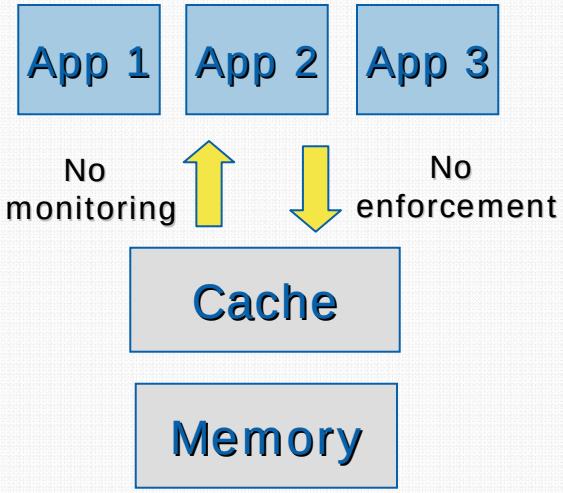
- **Platform has no support for application differentiation**
  - Platform has no knowledge of preferences or resource usage
  - Platform has no support for fine-grained tracking of many shared resources that have significant performance value

Platform QoS



Goals – Preferential Treatment of VIP, Better Overall Throughput

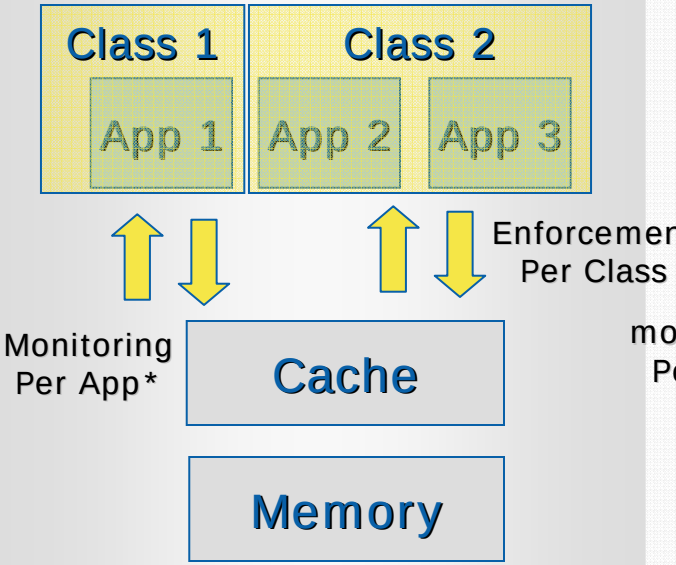
Visible QoS Spectrum (Cache/Memory)



No determinism

No QoS

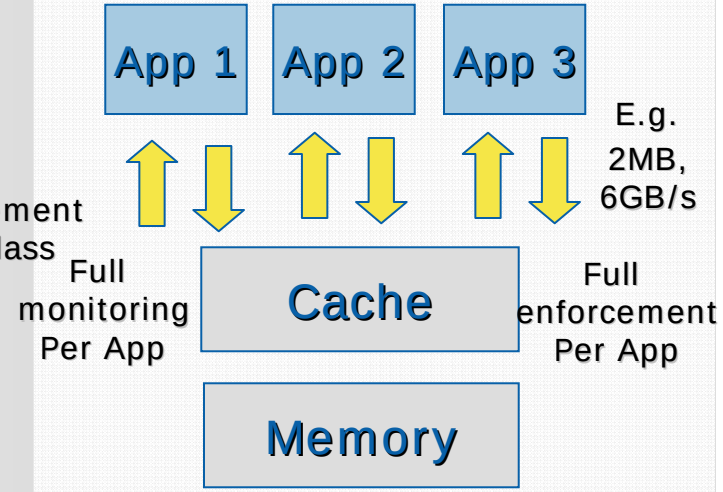
- No complexity
- ~ Good resource usage due to greedy approach
- ~ Okay throughput



Classes of Service

Possible Approach

- Incremental Complexity
- Bridges the gap
- Extensible Architecture
- Satisfies Requirements for perf management, perf. differentiation

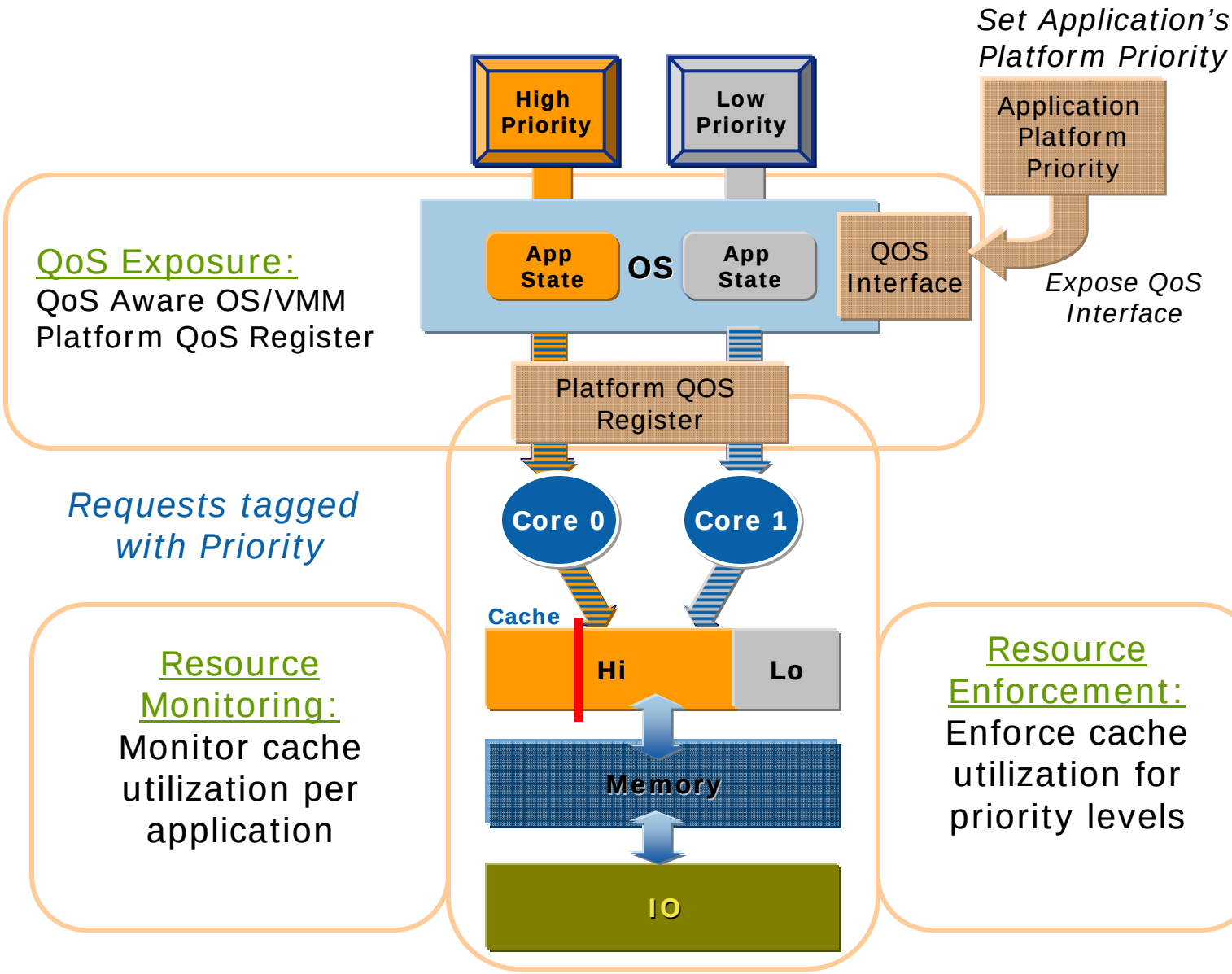


Guarantee per App

Per-App Partitions

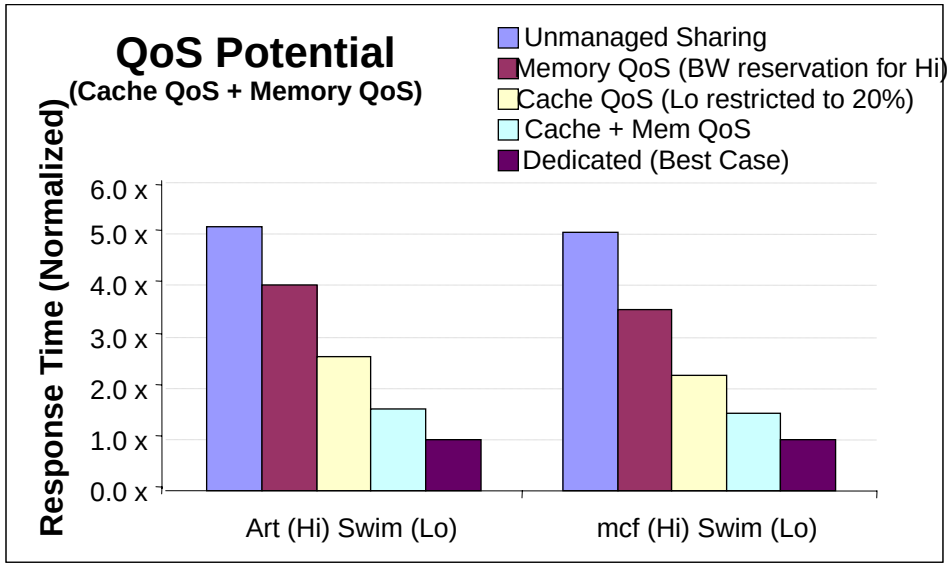
- Significant Complexity
- ~ Lower resource usage due to per VM guarantee
- ~ Poorer throughput

QoS Aware Architecture - Cache



Cache / Memory QoS Benefits

Client SPEC Case Study

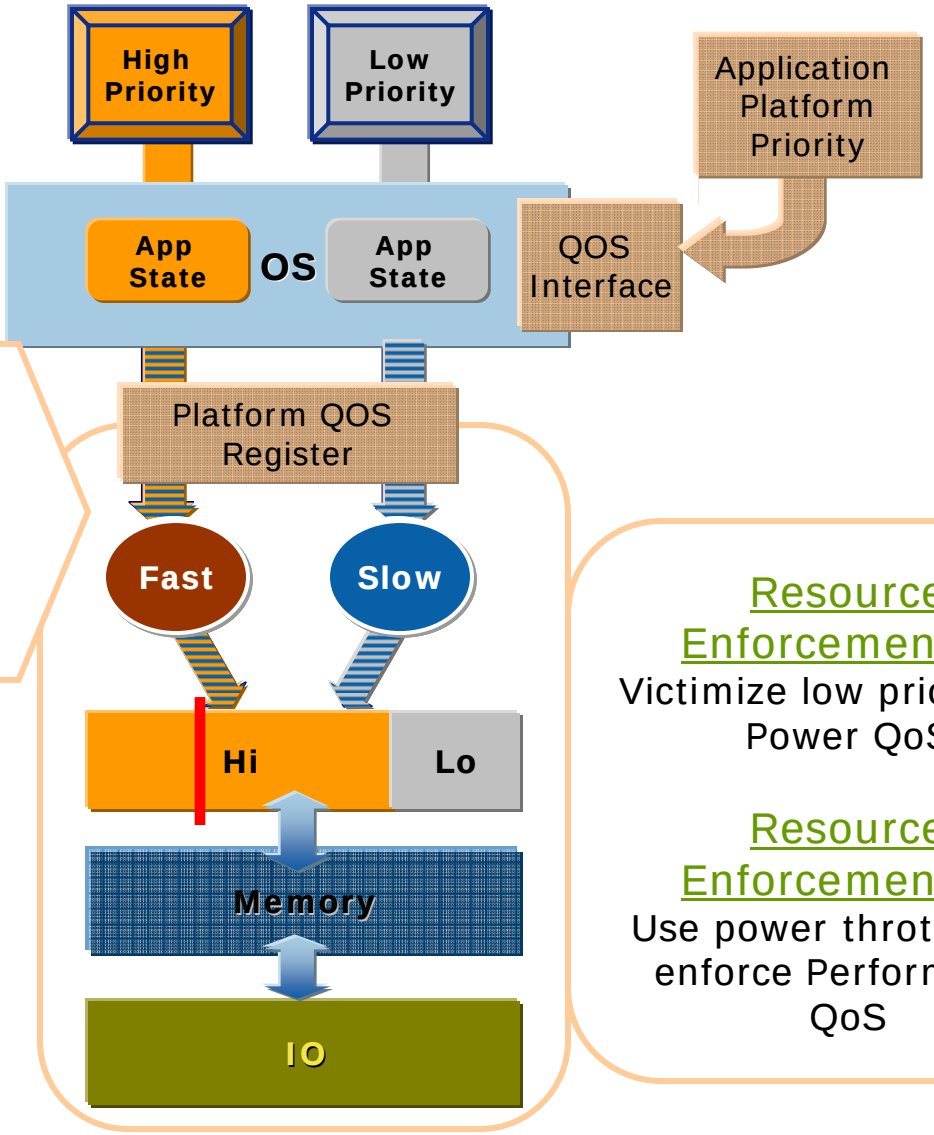


Response Time -- Lower is better

Based on Measurements, Simulations and Analytical Projections

Significant Benefits of Cache / Memory QoS

QoS Aware Architecture: Power



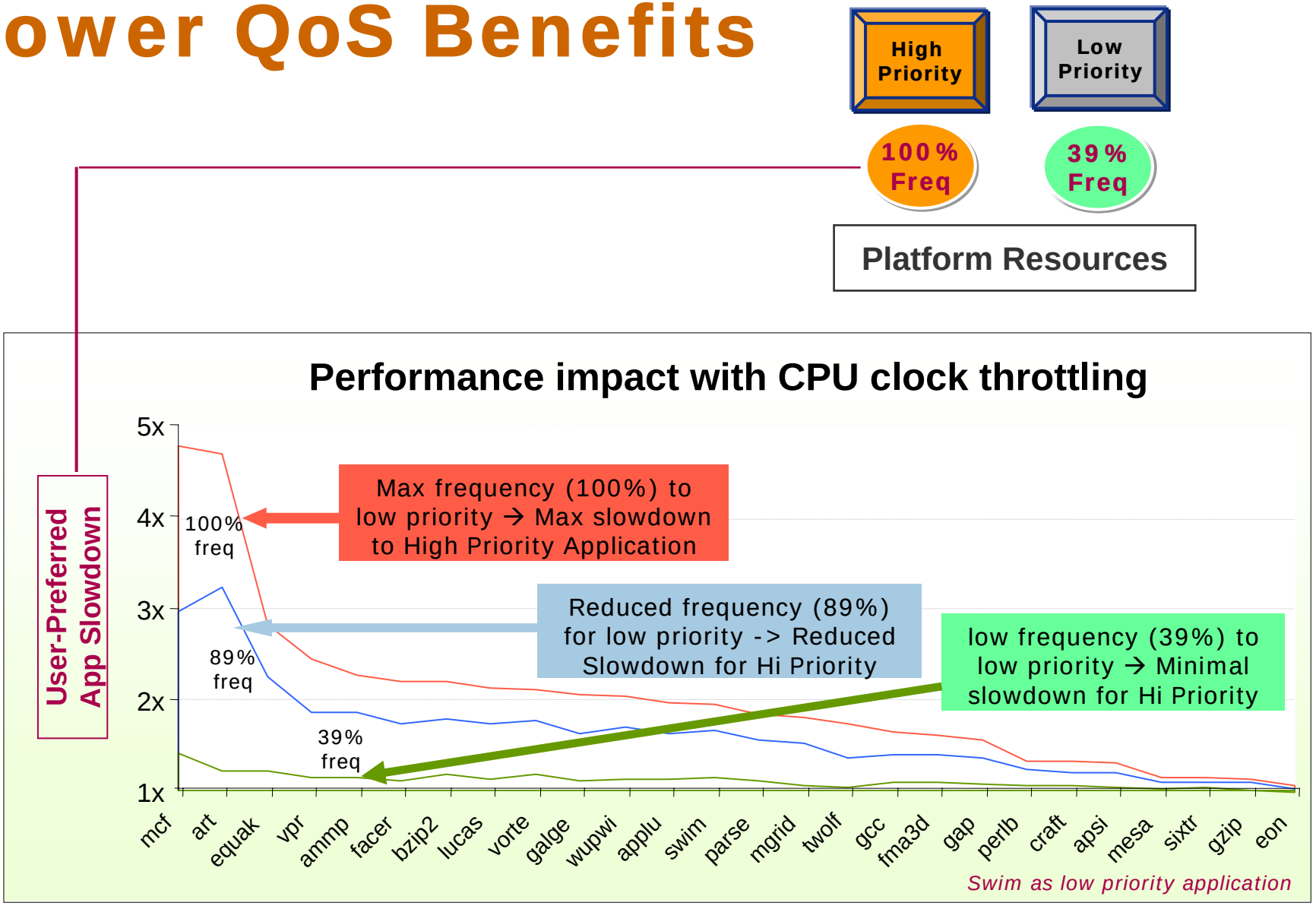
Power management knobs
- Voltage/Frequency scaling
- Issue restriction
- Gating

QoS Monitoring
Cache, Memory, IO, Power

Resource Enforcement#1
Victimize low priority for Power QoS

Resource Enforcement#2
Use power throttling to enforce Performance QoS

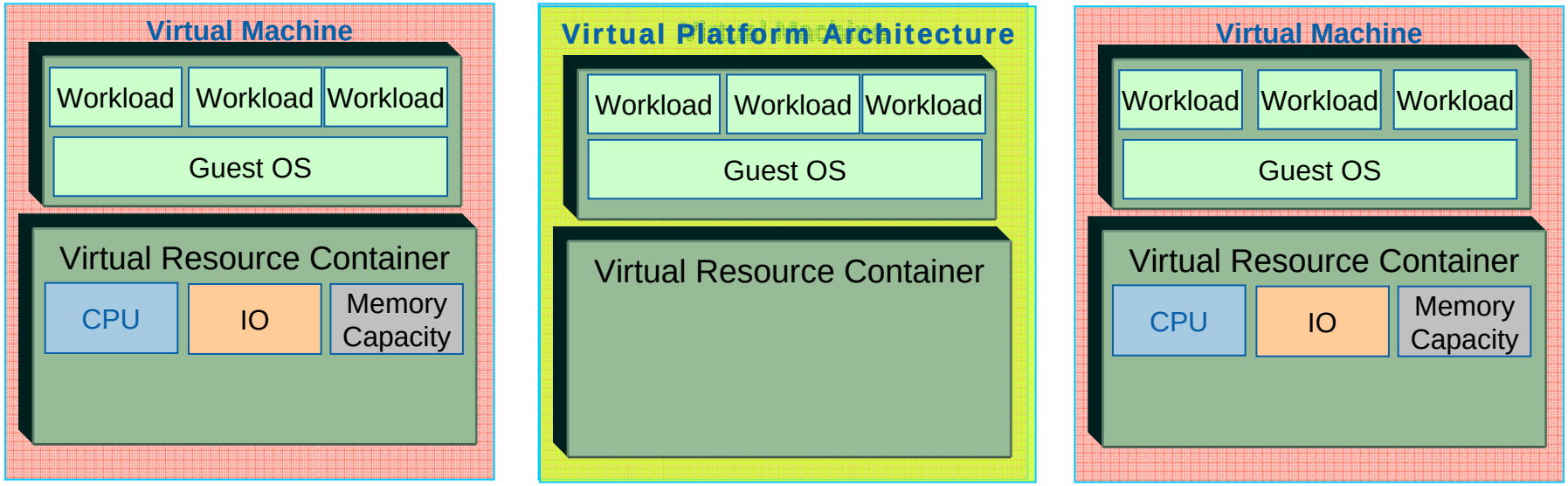
Power QoS Benefits



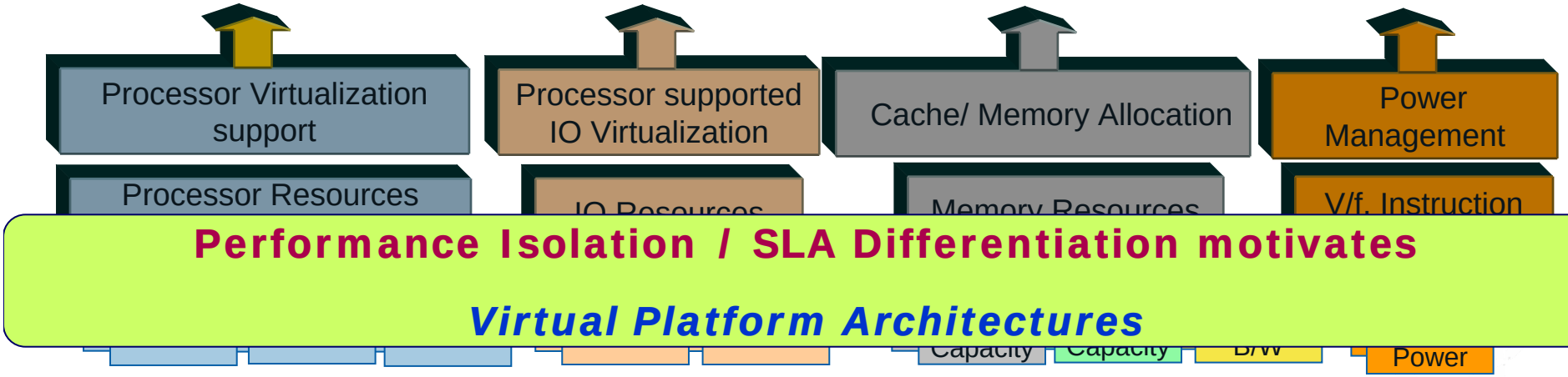
Improves performance of the user-preferred application

Virtualization: From VMs to VPAs

(Managing Transparent Resources)



Virtual Machine Monitor manages CPU, IO and memory capacity
Virtual Platform Architectures manages performance with PQoS



Performance Isolation / SLA Differentiation motivates

Virtual Platform Architectures

Summary

Large-scale CMP is going to happen

- –Lots of work to be done to identify and remove platform and architectural limitations preventing applications and execution environments from scaling up to 100's of logical processors on a single CPU die

- Scalability concerns can be addressed
  - –Hierarchy of Shared Caches
  - –Large DRAM caches
- QoS concerns can be addressed
  - –Dynamic Cache Allocation (Cache QoS)
  - –Dynamic Power Management (Power QoS)
- Smart performance management requires more visibility be available to the execution environment
  - –Resource utilization counters for schedulers, etc.