Consolidation of Real-Time Systems into a Multi-Core Platform

Hyoseung Kim
hyoseung@cmu.edu

Raj Rajkumar
raj@ece.cmu.edu

Carnegie Mellon University
Consolidation onto Multicores

• Benefits
  – Reduces the number of CPUs and wiring harness among them
  – Leads to a significant reduction in cost and space requirements
Key Challenges

- **Timing predictability** without losing efficiency
- **Temporal isolation** among consolidated workloads
  (Quantification & minimization of temporal interference)
- Use of standardized **COTS** multi-core platforms
Multi-Core Memory Hierarchy

Core 1

L1/L2 Cache

Core 2

L1/L2 Cache

Core 3

L1/L2 Cache

Core P

L1/L2 Cache

Last-level Shared Cache

Memory Controller

DRAM Bank 1

DRAM Bank 2

DRAM Bank 3

DRAM Bank B

Cache interference

Memory bus interference

DRAM bank interference
**Last-Level Shared Cache**

- A large cache shared among processing cores
  - Reduces memory access time (7 – 10x faster than DRAM access)
  - Mitigates DRAM bandwidth consumption
  - Allows high cache utilization

*Intel Core i7
8-15 MB L3 Cache*

*Freescale P4080
2MB L3 Cache*
Uncontrolled Use of Shared Cache

1. Inter-core Interference

- Core
- Private Cache
- Shared Cache

2. Intra-core Interference

- Core
- Private Cache
- Shared Cache

* PARSEC Benchmark on Intel i7
Coordinated Cache Management

• Challenges
  – **Uncontrolled shared cache**: Cache interference penalties
  – **Cache partitioning**:
    • Limited number of cache partitions
    • Cache under-utilization

• **Key idea**: Controlled sharing of partitioned caches while ensuring timing predictability

  1. Provides **predictability** on multi-core real-time systems
  2. Addresses the **limitations of cache partitioning**

Coordinated Cache Management

Reserved cache sharing: Mitigate the problems with page coloring

Considerations
1. Analyzing intra-core interference
2. Guaranteeing memory requirements
Coordinated Cache Management

1. Per-core Cache Reservation
2. Reserved Cache Sharing
3. Cache-Aware Task Allocation

Tasks + Bounded Penalties

Memory partitions Cache partitions

Core 1 Core 2 ...

Cache-Aware Task Allocation
- Algorithm to allocate tasks and cache partitions to cores
- Exploits the benefits of cache sharing by load concentration
Multi-Core Memory Hierarchy

- Core 1: L1/L2 Cache
- Core 2: L1/L2 Cache
- Core 3: L1/L2 Cache
- Core P: L1/L2 Cache

- L1/L2 Cache interference
- Memory bus interference
- DRAM bank interference

- Last-level Shared Cache
- Memory Controller
- DRAM Bank 1
- DRAM Bank 2
- DRAM Bank 3
- DRAM Bank B
DRAM Organization

DRAM Rank

Command bus
Address bus
Data bus

DRAM Chip

Command bus
Address bus
Data bus

DRAM access latency depends on which row is currently in the row buffer
Memory Controller

Request buffer

Memory scheduler

Two-level hierarchical scheduling structure (FR-FCFS)

Memory interference

- Intra-bank interference at bank scheduler
- Inter-bank interference at channel scheduler
**DRAM Bank Partitioning & Sharing**

- **DRAM bank partitioning**: software method to assign dedicated DRAM banks to each core (or task)
  
  \[(1) \text{ w/o bank partitioning} \]
  
  \[
  \begin{array}{c}
  \text{Core 1} \\
  \text{Bank 1 Scheduler} \\
  \text{Channel Scheduler}
  \end{array} \\
  \begin{array}{c}
  \text{Core 2} \\
  \text{Bank 2 Scheduler} \\
  \text{Channel Scheduler}
  \end{array}
  \]

  **Intra-bank and inter-bank interference**

  \[(2) \text{ w/ bank partitioning} \]
  
  \[
  \begin{array}{c}
  \text{Core 1} \\
  \text{Bank 1 Scheduler} \\
  \text{Channel Scheduler}
  \end{array} \\
  \begin{array}{c}
  \text{Core 2} \\
  \text{Bank 2 Scheduler} \\
  \text{Channel Scheduler}
  \end{array}
  \]

  **Inter-bank interference only**

- **Problems**
  - Limited number of DRAM bank partitions
  - Reduced usable memory size

  **DRAM bank sharing**
Bounding Memory Interference Delay

- Provides an upper bound on memory interference delay
  - Intra-bank and inter-bank interference delays
  - Private and shared DRAM banks

Request-driven bounding
Task’s own memory requests

Job-driven bounding
Interfering memory requests during the task execution

Response-Time Based Schedulability Analysis

Multi-Core Memory Hierarchy

Operating System

Core 1
L1/L2 Cache

Core 2
L1/L2 Cache

Core 3
L1/L2 Cache

Core P
L1/L2 Cache

Last-level Shared Cache

Memory Controller

DRAM Bank 1

DRAM Bank 2

DRAM Bank 3

DRAM Bank B

Cache interference

Memory bus interference

DRAM bank interference

Task $\tau_1$
Task $\tau_2$
Task $\tau_3$
Task $\tau_4$
Task $\tau_5$
Task $\tau_6$
Linux/RK

- CMU’s open source real-time extensions to the Linux kernel
- Resource kernel (RK) approach
- Resource reservation: tasks can reserve a portion of system resources
  - Ex) CPU reserve: (budget, replenishment period, CPU affinity, policy)
- Guarantees resource allocations at admission time
- Enforces resource usage
Multi-Core Memory Management

- **Linux/RK memory manager**
  - Supports software cache and DRAM partitioning
  - Allows tasks to specify their memory size, cache, DRAM bank demands

**Task i: memory reservation**
- Memory size = \( m \) pages
- Cache partitions = \( h \)
- DRAM bank partitions = \( b \)

**Task i: allocated pages**

Unallocated physical pages
Cache Interference

• Task execution time and L3 misses
  – Intel i7 2600 3.4GHz quad-core, 8MB L3 cache, DDR3-1333
Memory Interference

- Shared DRAM Bank

12x increase due to memory interference
Memory Interference

- Private DRAM Bank

4.1x increase → DRAM bank partitioning helps reducing the memory interference

Our analysis enables the quantification of the benefit of DRAM bank partitioning
Memory Interference

- Private DRAM Bank under moderate memory contention

Our analysis bounds memory interference delay with low pessimism under both severe and moderate memory contentions.

Average over-estimates are 8%
(13% for a shared bank)
Multi-Core Virtualization
Real-Time System Virtualization

• **Barrier to consolidation**
  – Each app. could have been developed independently by different vendors
    • Bare-metal / Proprietary OS
    • Linux
  – Legacy software
  – Different license issues

• **Consolidation via virtualization**
  – Each application can maintain its own implementation
  – Minimizes re-certification process
  – IP protection, license segregation
  – Fault isolation
VM Scheduling Structure

- Two-level hierarchical scheduling structure
  - Task scheduling and VCPU scheduling
Virt/RK

- **Real-time virtualization** with resource kernel approach
  - **CPU reservation** for VCPUs
  - **Memory reservation** for VMs

**VM Resource Reservation**
- VCPU1: 30% of physical CPU
- VCPU2: 30% of physical CPU
- VM1: 20% of host memory w/ cache & DRAM bank partitioning

- Current implementation: Virt/RK::KVM-x86
- Under development: Virt/RK::KVM-ARM, Virt/RK::L4
Resource Sharing in Virtualization

• Consolidation inevitably causes the sharing of resources
  – Sensors
  – Network interfaces
  – I/O devices
  – Shared data

  Requires mutually-exclusive locks to avoid race conditions

• Long blocking time in a virtualized environment
  – VCPU level preemptions
  – VCPU budget depletion

Need a synchronization mechanism with bounded blocking times for multi-core real-time virtualization
vMPCP

• Virtualization-aware Multiprocessor Priority Ceiling Protocol
  – Provides **bounded blocking time** on accessing shared resources in multi-core virtualization
    • Hierarchical priority ceilings
    • Two-level priority queue for a mutex waiting list
  – Optional VCPU budget overrun to reduce blocking times
  – VCPU budget replenishment policies
    • Periodic server
    • Deferrable server
  – Implemented on Virt/RK::KVM-x86

Case Study: Effect of vMPCP

Baseline
Virtualization-unaware synchronization protocol (MPCP)

vMPCP
Virtualization-aware synchronization protocol

vMPCP yields 29% shorter task response time with shared resources
Summary

• **Workload Consolidation into a Multi-core Platform**
  – *Predictability* without losing *efficiency*
  – Quantification and minimization of *temporal interference*

• **Multi-core memory hierarchy**
  – Coordinated cache management
    • Cache partitioning, reserve, and sharing
  – Bounding memory interference
    • DRAM partitioning and sharing → *Quantifies* the benefit of partitioning
    • Memory-bus and DRAM bank interference analysis

• **Real-time virtualization**
  – *Virt/RK*: resource reservation approach
  – *vMPCP*: resource sharing for real-time virtualization
What next?

• Modeling state-of-the-art computer architecture techniques
  – Hardware prefetchers
  – Future memory controllers
  – Future DRAM designs

• Extending existing real-time systems work to virtualization
  – Diverse locking protocols (e.g., RW-lock)
  – Fault tolerance mechanisms
  – Hardware accelerations (e.g., GPGPU)