A-DRM: Architecture-aware Distributed Resource Management of Virtualized Clusters

Hui Wang†*, Canturk Isci‡, Lavanya Subramanian*, Jongmoo Choi#*, Depei Qian†, Onur Mutlu*

†Beihang University, ‡IBM T. J. Watson Research Center, *Carnegie Mellon University, #Dankook University
Executive Summary

• Virtualized Clusters dynamically schedule a set of Virtual Machines (VM) across many physical hosts (called DRM, Distributed Resource Management)

• **Observation**: State-of-the-art DRM techniques do not take into account microarchitecture-level resource (cache and memory bandwidth) interference between VMs

• **Problem**: This lack of visibility into microarchitecture-level resources significantly impacts the entire virtualized cluster’s performance

• **Our Goal**: Maximize virtualized cluster performance by **making DRM microarchitecture aware**

• **Mechanism**: Architecture-aware Distributed Resource Management (A-DRM):
  1) Dynamically monitors the microarchitecture-level shared resource usage
  2) Balances the microarchitecture-level interference across the cluster (while accounting for other resources as well)

• **Key Results**: 9.67% higher performance and 17% higher memory bandwidth utilization than conventional DRM
Virtualized Cluster

Distributed Resource Management (DRM) policies

How to dynamically schedule VMs onto hosts?
Conventional DRM Policies

Based on operating-system-level metrics e.g., CPU utilization, memory capacity demand
Microarchitecture-level Interference

• VMs within a host compete for:
  – Shared cache capacity
  – Shared memory bandwidth

Can operating-system-level metrics capture the microarchitecture-level resource interference?
### Microarchitecture Unawareness

<table>
<thead>
<tr>
<th>VM</th>
<th>Operating-system-level metrics</th>
<th>Microarchitecture-level metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Utilization</td>
<td>Memory Capacity</td>
</tr>
<tr>
<td>App</td>
<td>92%</td>
<td>369 MB</td>
</tr>
<tr>
<td>App</td>
<td>93%</td>
<td>348 MB</td>
</tr>
</tbody>
</table>

**Diagram:**
- **CPU**: Blue arrows indicate data transfer among VMs and cores.
- **Memory Capacity**: Green arrows highlight memory access flow.
- **SAFARI**: Red and yellow boxes represent different applications running within VMs.
Impact on Performance

IPC (Harmonic Mean)

0.0 0.2 0.4 0.6

Conventional DRM

Host

Core0

VM

App

Core1

VM

App

DRAM

LLC

SWAP

Memory Capacity

CPU

CPU

App

STREAM

App

gromacs

SAFARI
We need microarchitecture-level interference awareness in DRM!
Outline

- Motivation
- A-DRM
- Methodology
- Evaluation
- Conclusion
A-DRM: Architecture-aware DRM

• **Goal**: Take into account microarchitecture-level shared resource interference
  – Shared cache capacity
  – Shared memory bandwidth

• **Key Idea**:  
  – Monitor and detect microarchitecture-level shared resource interference
  – Balance microarchitecture-level resource usage across cluster
Conventional DRM

- OS+Hypervisor
  - VM
    - App
  - CPU/Memory Capacity
- Profiler
- DRM: Global Resource Manager
  - Profiling Engine
  - Distributed Resource Management (Policy)
  - Migration Engine
- Controller

SAFARI
A-DRM: Architecture-aware DRM

**A-DRM: Global Architecture – aware Resource Manager**

- Profiling Engine
- Architecture-aware Interference Detector
- Architecture-aware Distributed Resource Management (Policy)
- Migration Engine

**Hosts**

- OS+Hypervisor
  - VM
    - App
  - CPU/Memory Capacity
  - Architectural Resources

**Controller**

  - Profiling Engine
  - Architecture-aware Interference Detector
  - Architecture-aware Distributed Resource Management (Policy)
  - Migration Engine
Architectural Resource Profiler

- Leverages the Hardware Performance Monitoring Units (PMUs):
  - Last level cache (LLC)
  - Memory bandwidth (MBW)

- Reports to Controller periodically
A-DRM: Architecture-aware DRM

**Architecture-aware Resource Manager**

- **Profile Engine**
- **Architecture-aware Interference Detector**
- **Architecture-aware Distributed Resource Management (Policy)**
- **Migration Engine**

**Hosts**

- OS+Hypervisor
  - VM
    - App
  - CPU/Memory
  - Architectural Resources

**Controller**

Architecture-aware Interference Detector

• **Goal**: Detect shared cache capacity and memory bandwidth interference

• Memory bandwidth utilization ($MBW_{util}$) captures both:
  – Shared cache capacity interference
  – Shared memory bandwidth interference

Key observation: If $MBW_{util}$ is too high, the host is experiencing interference
A-DRM: Architecture-aware DRM

OS+Hypervisor

VM
App

CPU/Memory

Architectural Resources

Profiler

Hosts

Controller

A-DRM: Global Architecture – aware Resource Manager

Profiling Engine

Architecture-aware Interference Detector

Architecture-aware Distributed Resource Management (Policy)

Migration Engine
A-DRM Policy

• Two-phase algorithm

• Phase One:
  – Goal: Mitigate microarchitecture-level resource interference
  – Key Idea: Suggest migrations to balance memory bandwidth utilization across cluster using a new cost-benefit analysis

• Phase Two:
  – Goal: Finalize migration decisions by also taking into account OS-level metrics (similar to conventional DRM)
A-DRM Policy: Phase One

- **Goal**: Mitigate microarchitecture-level shared resource interference
  - Employ a new **cost-benefit analysis** to filter out migrations that cannot provide enough benefit
  - Only migrate the least number of VMs required to bring the $MBW_{util}$ below a threshold ($MBW_{Threshold}$)
A-DRM Policy: Phase Two

• **Goals:**
  – Finalize migration decisions by also taking into account OS-level metrics
  – **Avoid new microarchitecture-level resource hotspots**
A-DRM Policy

• Two-phase algorithm

• Phase One:
  – Goal: Mitigate microarchitecture-level resource interference
  – Key Idea: Suggest migrations to balance memory bandwidth utilization across cluster using a new cost-benefit analysis

• Phase Two:
  – Goal: Finalize migration decisions by also taking into account OS-level metrics (similar to conventional DRM)
The Goal of Cost-Benefit Analysis

• For every VM at a contended host, we need to determine:
  – If we should migrate it
  – Where we should migrate it

• For each VM at a contended source, we consider migrating it to every uncontended destination

• We develop a new linear model to estimate the performance degradation/improvement in terms of time
Cost-Benefit Analysis

- **Costs** of migrating a VM include:
  1. VM migration cost ($Cost_{migration}$),
  2. Performance degradation at the destination host due to increased interference ($Cost_{dst}$)

- **Benefits** of migrating a VM include:
  1. Performance improvement of the migrated VM ($Benefit_{vm}$),
  2. Performance improvement of the other VMs on the source host due to reduced interference ($Benefit_{src}$)

Phase One of A-DRM suggests migrating a VM if

$$Benefit_{vm} + Benefit_{src} > Cost_{migration} + Cost_{dst}$$
Cost-Benefit Analysis

- **Costs** of migrating a VM include:
  1) VM migration cost ($Cost_{migration}$),
  2) Performance degradation at the destination host due to increased interference ($Cost_{dst}$)

- **Benefits** of migrating a VM include:
  1) Performance improvement of the migrated VM ($Benefit_{vm}$),
  2) Performance improvement of the other VMs on the source host due to reduced interference ($Benefit_{src}$)

Phase One of A-DRM suggests migrating a VM if

\[ Benefit_{vm} + Benefit_{src} > Cost_{migration} + Cost_{dst} \]
**Cost_{migration}**: VM migration

- VM migration approach used in A-DRM:
  - ‘Pre-copy-based’ live migration + timeout support

- High cost since all of the VM’s pages need to be iteratively:
  - scanned, tracked
  - transferred

- The migration time can be estimated similar to conventional DRM policies
Cost-Benefit Analysis

• Costs of migrating a VM include:
  1) VM migration cost \(Cost_{\text{migration}}\),
  2) Performance degradation at the destination host due to increased interference \(Cost_{\text{dst}}\)

• Benefits of migrating a VM include:
  1) Performance improvement of the migrated VM \(Benefit_{\text{vm}}\),
  2) Performance improvement of the other VMs on the source host due to reduced interference \(Benefit_{\text{src}}\)

Phase One of A-DRM suggests migrating a VM if

\[Benefit_{\text{vm}} + Benefit_{\text{src}} > Cost_{\text{migration}} + Cost_{\text{dst}}\]
**Cost_{dst}: Performance Degradation at dst**

• The migrated *vm* competes for:
  – Shared cache capacity
  – Shared memory bandwidth

• Performance at *dst* degrades due to:
  – Increase in memory bandwidth consumption
  – Increase in the memory *stall* time experienced by VMs
Cost-Benefit Analysis

- **Costs** of migrating a VM include:
  1. VM migration cost ($Cost_{migration}$),
  2. Performance degradation at the destination host due to increased interference ($Cost_{dst}$)

- **Benefits** of migrating a VM include:
  1. Performance improvement of the migrated VM ($Benefit_{vm}$),
  2. Performance improvement of the other VMs on the source host due to reduced interference ($Benefit_{src}$)

Phase One of A-DRM suggests migrating a VM if

$$Benefit_{vm} + Benefit_{src} > Cost_{migration} + Cost_{dst}$$
Benefit_{vm}: Performance improvement of \textit{vm}

- The performance of migrated \textit{vm} improves due to:
  - Lower contention for memory bandwidth
  - Lower memory \textit{stall} time
Cost-Benefit Analysis

- **Costs** of migrating a VM include:
  1) VM migration cost ($Cost_{migration}$),
  2) Performance degradation at the destination host due to increased interference ($Cost_{dst}$)

- **Benefits** of migrating a VM include:
  1) Performance improvement of the migrated VM ($Benefit_{vm}$),
  2) Performance improvement of the other VMs on the source host due to reduced interference ($Benefit_{src}$)

Phase One of A-DRM suggests migrating a VM if

$$Benefit_{vm} + Benefit_{src} > Cost_{migration} + Cost_{dst}$$
**Benefit}_{src}: Performance improvement at src

- The performance at src improves due to:
  - Reduced memory bandwidth consumption
  - Reduced \textit{stall} time experienced by VMs
A-DRM Policy

• Two-phase algorithm

• Phase One:
  – Goal: Mitigate microarchitecture-level resource interference
  – Key Idea: Suggest migrations to balance memory bandwidth utilization across cluster using a new cost-benefit analysis

• Phase Two:
  – Goal: Finalize migration decisions by also taking into account OS-level metrics (similar to conventional DRM)
Outline

• Motivation
• A-DRM
• Methodology
• Evaluation
• Conclusion
Evaluation Infrastructure

- 2/4 dual-socket Hosts
  - Two 4-core Xeon L5630 Processors (Westmere-EP) with hyperthreading disabled
    - L1/L2/shared LLC: 32KB/256KB/12MB
  - One 8GB DDR3-1066 DIMM per socket

VM Images placed in shared storage (NAS)

OS and Hypervisor:
- Fedora 20 with Linux Kernel version 3.13.5-202
### DRM Parameters

- **Baseline:** Conventional DRM [Isci et al., NOMS’ 10]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU overcommit threshold ( CPU_{\text{Threshold}} )</td>
<td>90%</td>
</tr>
<tr>
<td>Memory overcommit threshold ( MEM_{\text{Threshold}} )</td>
<td>95%</td>
</tr>
<tr>
<td>Memory bandwidth threshold ( MBW_{\text{Threshold}} )</td>
<td>60%</td>
</tr>
<tr>
<td>DRM scheduling interval ( \text{ scheduling interval} )</td>
<td>300 seconds</td>
</tr>
<tr>
<td>DRM sliding window size</td>
<td>80 samples</td>
</tr>
<tr>
<td>Profiling interval ( \text{ profiling interval} )</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Live migration timeout ( \text{ live migration timeout} )</td>
<td>30 seconds</td>
</tr>
</tbody>
</table>
Workloads

• 55 Workloads chosen from:
  – PARSEC (10)
  – SPEC CPU 2006 (28)
  – NAS Parallel Benchmark (14)
  – STREAM (1)
  – Microbenchmark (2)

• Classified based on memory intensity:
  – memory-intensive (memory bandwidth larger than 1GB/s)
  – memory-non-intensive
Outline

• Motivation
• A-DRM
• Methodology
• Evaluation
  1. Case Study
  2. Heterogeneous Workloads
  3. Per-Host vs. Per-Socket Interference Detection
• Conclusion
1. Case Study

- 14 VMs on **two** 8-core hosts
- Initially:
  - Host A: 7 memory-intensive VMs (STREAM)
  - Host B: 7 memory-non-intensive VMs (gromacs)
CPU Util [%]

Mem Capacity Util [%]

MBW Util [%]

Host A

Host B

SAFARI
**CPU Util [%]**

- **Host A**
- **Host B**

**Mem Capacity Util [%]**

- **Host A**
- **Host B**

**MBW Util [%]**

- **Host A**
- **Host B**
- By migrating VMs using online measurement of microarchitecture-level resource usage, A-DRM:
  - Mitigates resource interference
  - Achieves better memory bandwidth utilization
Outline

• Motivation
• A-DRM
• Methodology
• Evaluation
  1. Case Study
  2. Heterogeneous Workloads
  3. Per-Host vs. Per-Socket Interference Detection
• Conclusion
2. Heterogeneous workloads

- 28 VMs on **four** 8-core hosts
- Unbalanced placement according to intensity

- Workloads (denoted as \( iXnY-Z \)):
  - \( X \) VMs running **memory-intensive** benchmarks
  - \( Y \) VMs running **memory-non-intensive** benchmarks
  - \( Z \) indicates the two different workloads under the same intensity
**Performance Benefits of A-DRM**

- Compared to traditional DRM scheme:
  - Performance improves by up to 26.6%, with an average of 9.7%
  - The higher the imbalance between hosts, the greater the performance improvement
• The higher the imbalance between hosts, the greater the number of migrations
Cluster-wide Resource Utilization

- Average memory bandwidth utilization improves by 17%
- Comparable CPU and memory capacity utilization
Outline

• Motivation
• A-DRM
• Methodology
• Evaluation
  1. Case Study
  2. Heterogeneous Workloads
  3. Per-Host vs. Per-Socket Interference Detection
• Conclusion
Per-Host vs. Per-Socket Interference Detection

Per-Host

Host A

Socket 1

Core

VM

App

Core

VM

App

LLC

DRAM

Socket 2

Core

VM

App

Core

VM

App

LLC

DRAM

QPI

Host B

Socket 1

Core

VM

App

Core

VM

App

LLC

DRAM

QPI

Socket 2

Core

VM

App

Core

VM

App

LLC

DRAM

Per-Socket
Performance Benefits of Per-Host vs. Per-Socket

- Per-Socket Detection achieves better IPC improvement than Per-Host Detection
Outline

• Motivation
• A-DRM
• Methodology
• Evaluation
• Conclusion
Conclusion

• Virtualized Clusters dynamically schedule a set of Virtual Machines (VM) across many physical hosts (called DRM, Distributed Resource Management)

• Observation: State-of-the-art DRM techniques do not take into account microarchitecture-level resource (cache and memory bandwidth) interference between VMs

• Problem: This lack of visibility into microarchitecture-level resources significantly impacts the entire virtualized cluster’s performance

• Our Goal: Maximize virtualized cluster performance by making DRM microarchitecture aware

• Mechanism: Architecture-aware Distributed Resource Management (A-DRM):
  1) Dynamically monitors the microarchitecture-level shared resource usage
  2) Balances the microarchitecture-level interference across the cluster (while accounting for other resources as well)

• Key Results: 9.67% higher performance and 17% higher memory bandwidth utilization than conventional DRM
A-DRM: Architecture-aware Distributed Resource Management of Virtualized Clusters

Hui Wang†*, Canturk Isci‡, Lavanya Subramanian*, Jongmoo Choi#*, Depei Qian†, Onur Mutlu*

†Beihang University, ‡IBM T. J. Watson Research Center, *Carnegie Mellon University, #Dankook University