A Case for Tracking and Exploiting *Inter-node* and Intra-node Memory Content Sharing in Virtualized Large-Scale Parallel Systems

Lei Xia   Peter Dinda

Northwestern University

{lxia, pdinda}@northwestern.edu

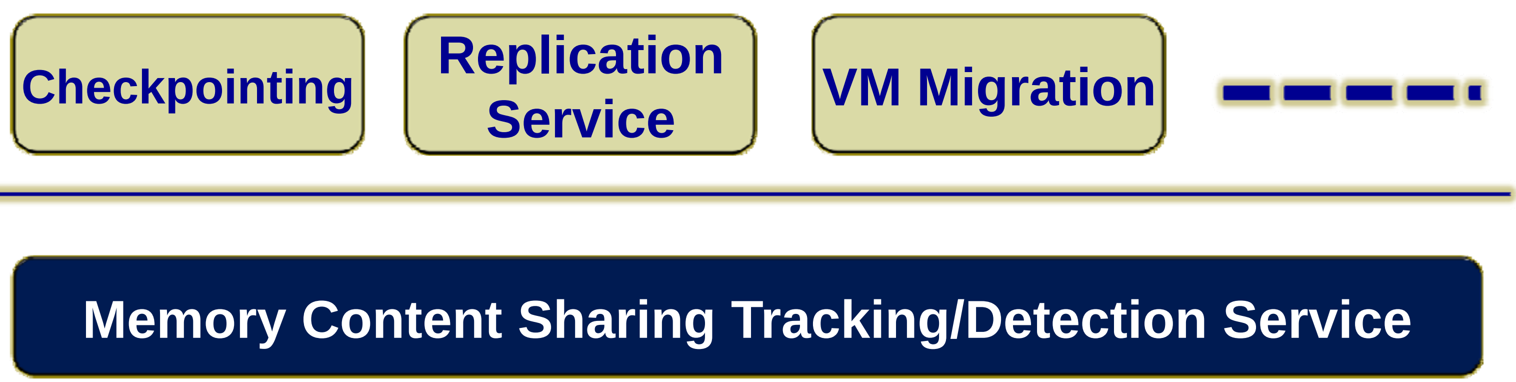
http://v3vee.org

Overview

- Many services can be simplified and enhanced by leveraging the memory content sharing within individual nodes and *across* nodes
- Detailed study of the memory content sharing in scientific workloads
- A proposed service for scalable identifying and tracking of ***inter-node*** memory content sharing in large-scale parallel systems

Motivation

- Many services in HPC systems can be simplified and improved by leveraging the intra- and *inter-node* memory content sharing
- Content-sharing detection is a common primitive that can be factored out of these services



Content-Sharing Aware Checkpointing

- **Checkpointing is important**
  - Widely used for fault tolerance in HPC systems [AGARWAL-ICS'04, MOODY-SC'10]
  - Larger checkpoint size, more checkpoints
    - 50~200TB/step, MTTR ~ 10 minutes [MOODY-SC'10]
- **Content-sharing-Aware Checkpointing**
  - Save only one copy of each distinct content (block) across the system
    - Reduced checkpoint file size
    - Reduced I/O and network traffic

Virtual Machines Co-Migration

- **Virtual machine migration in HPC**
  - Migrating a single VM [CLARK-NSDI'05, SAPUNTZAKIS-OSDI'02] /a set of VMs [NISHIMURA-CCGRID'07]
  - Fault tolerance, easy maintenance, load balancing [NAGARAJAN-ICS'07]
- **Content-sharing detection can benefit**
  - Single VM migration: Reconstruct VM memory from multiple source VMs
    - VM starts faster on remote host
  - Collective VM co-Migration: Migrate only one copy of each distinct memory content across all VMs
    - Reduce network traffic to migrate the set of VMs

Memory Replication System for High Availability

- **Redundant Systems by Replication**
  - Enhance availability and reliability [FERREIRA-SC'11, NATH-NSDI'06]
  - Maintain a certain copies for each memory page in system
  - Costly, large amount of memory needed
- **Memory Replication System using Content-sharing Service**
  - Reduce memory usage by exploiting existing content redundancy in applications
  - Avoid creating memory replicas explicitly when there are memory pages with same contents already exists in remote nodes

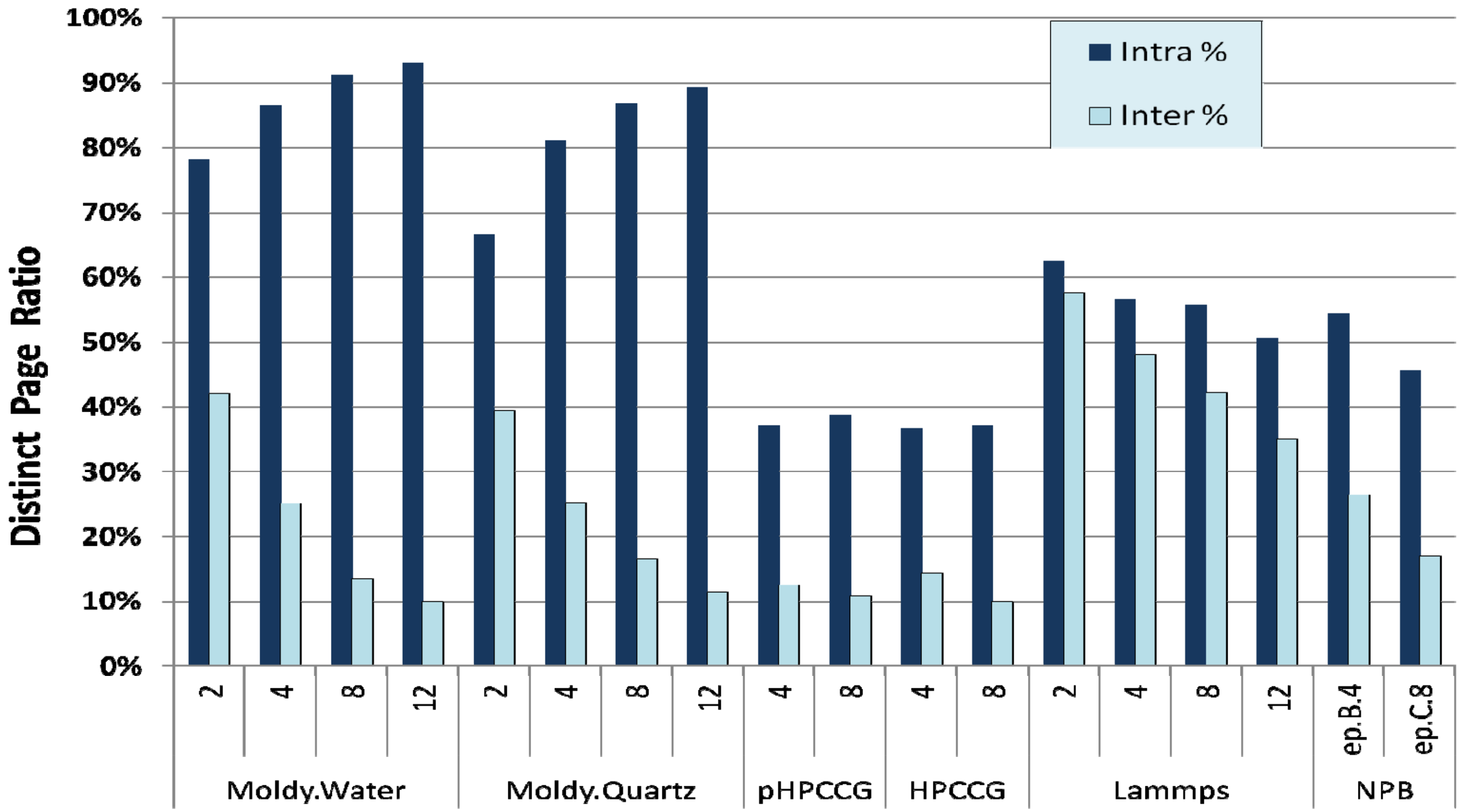
More ……

- **Determining good points for system checkpointing and migration**
  - Monitoring amount of sharing over time
  - Suggests a good time for checkpoint/migration
- **Power efficient system support**
  - Lowering power to saving power
  - Could reduce system stability/availability intentionally
  - Transparently enhance the lowered availability to users through content-share aware redundancy

Memory Content Sharing in Scientific Workloads

- Experimental Study:
  - **Goal**: Examine ***intra-*** and ***inter-***node memory content sharing in parallel applications.
  - **Benchmarks**: Moldy, NAS, HPCC, Lammps and Miniapps
  - **Method**: run a set of parallel applications & benchmarks on a cluster
    - Stop all processes periodically, dump the memory content of each process, generate hash for each memory block
    - Compare the hash to analysis the number of content-shared blocks within and across nodes
    - Percentage of pages in system that have unique content

Opportunity: Applications with Much Inter-node Sharing



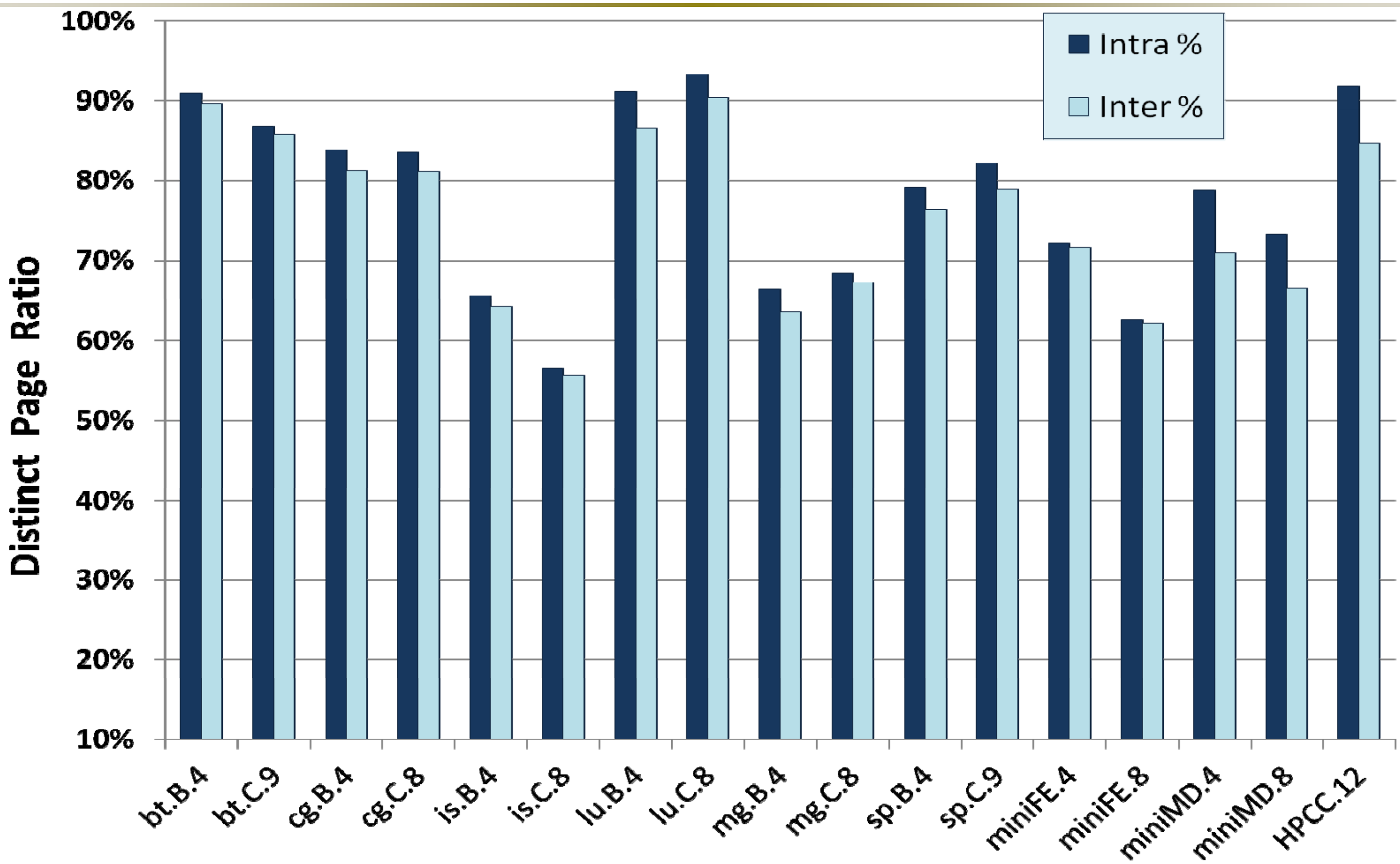
Many applications have much *inter-node* sharing but little intra-node sharing

Potential Memory Gains

Problem Size	Number of Nodes	Total	Intra-Distinct	Inter-Distinct
128x128x256	2	29 MB	19 MB	11 MB
256x256x256	4	161 MB	131 MB	41 MB
512x512x256	6	489 MB	417 MB	91 MB
1024x1024x256	8	1337 MB	1161 MB	220 MB
2048x2048x256	10	3057 MB	2706 MB	426 MB
4096x4096x256	12	5324 MB	4753 MB	612 MB

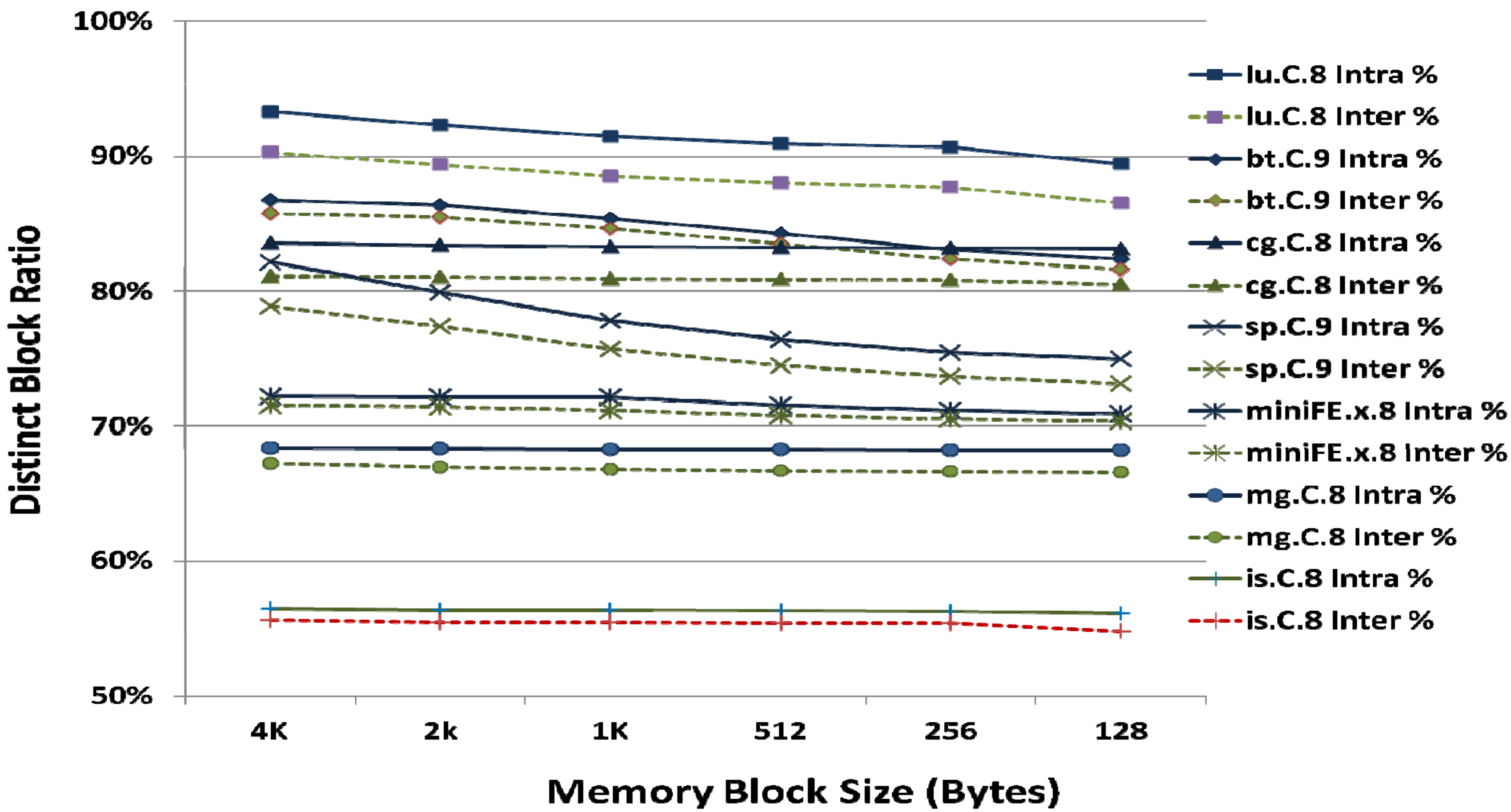
Memory that could be potentially reduced when inter-node content sharing is removed

Applications with Intra-node Sharing



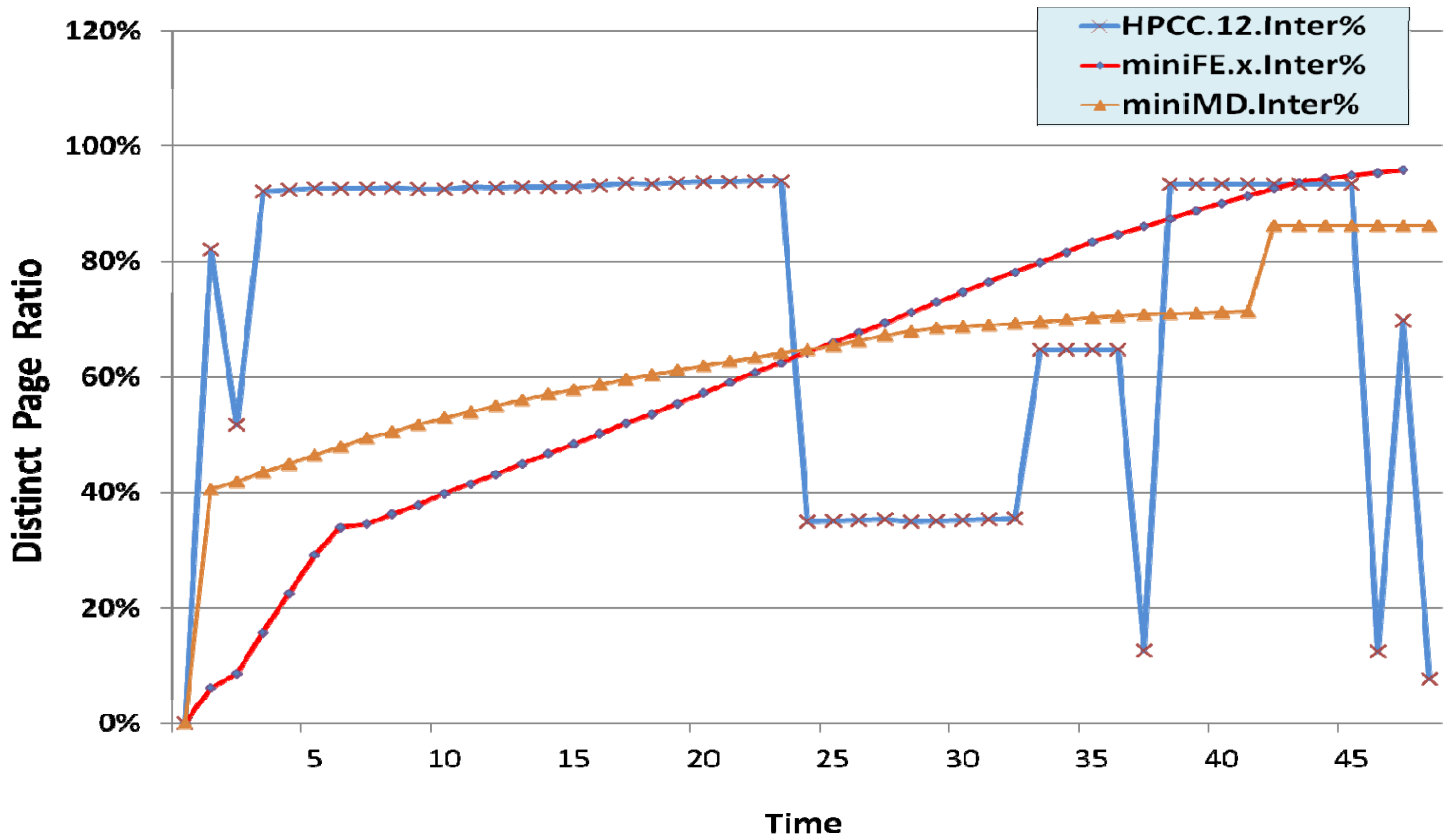
Some applications have little inter-node sharing but some intra-node sharing

Content Sharing using Smaller Block Size



Reducing the block size does not help much to find more content sharing

Memory Content Sharing over Time



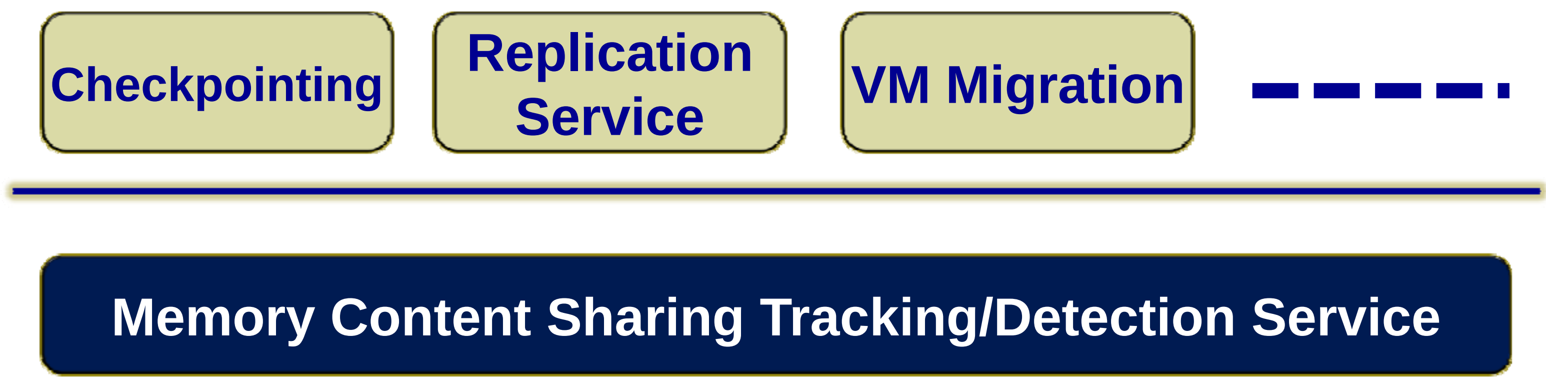
Different level of content sharing over time

Experimental Study Summary

- **Intra**- and **inter**-node memory content sharing is common in parallel applications
- There is opportunity for exploiting this memory content sharing to benefit many services in HPC systems
- An online content-sharing detection system is needed

A Tracking System Could be Built

- Content-sharing detection/tracing is a common primitive that can be factored out of these services



Memory Content-Sharing Detection System

- Detecting and tracking content-sharing in the system
  - ***Inter-node*** and intra-node memory content sharing
- Providing the content-sharing status to up-level services
- Advisory system
  - Best effort service with low performance overhead
  - Could have false positives/false negatives
- Online detection system

Information Provided by the Detection System

- Degree of memory content sharing
  - Percentage of pages in system that have unique content
- Replica discovery
  - Find all instances of specific page content
- Find hot or cold page contents
  - Number/Locations of memory blocks with more than/less than k copies in the system

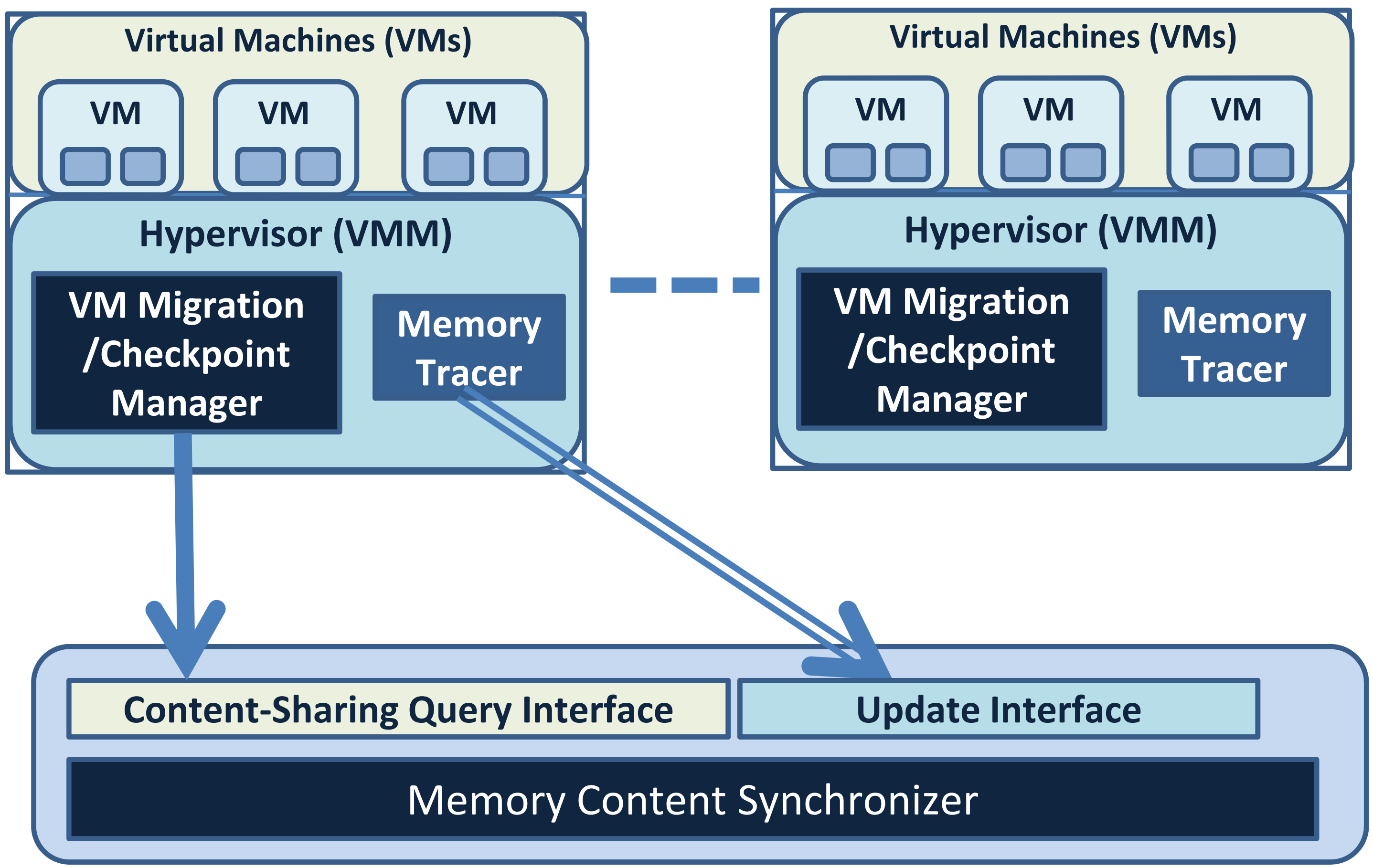
Challenges

- **Scalability**
  - Scale from small of number of nodes to large scale system
  - Decentralized Control
    - Centralized control prevents scalability and is a single point of failure
    - All information collection/computation are distributed on all nodes
- **Online Detection**
  - Dynamic detection/tracking of memory content sharing in system

Assumptions

- High throughput/low latency network
  - Network scales as size increases
  - Supercomputer network (such as mesh)
  - *Network synchronization between nodes are much faster than distributed systems*
- Node failure is independent
  - System can rely on replication for fault tolerance
  - *The system can replicate control information across more than more node to provide fault tolerance*
- Securely controlled environments
  - No critical security concerns
  - *Can use less CPU-intensive no-cryptographic hash functions*

System Architecture



Content-Sharing Detection System

Proposed Approach

- **Front-end: Memory Tracer**
  - Running in each node
  - Track memory updates
    - dirty bit in page table entry
  - Rehash all updated pages
    - Periodically
    - Event-driven (performance counter, etc)
  - Send new hashes to back-end
    - Determine which node it should send the hash
      - by only the hash value itself (consistent hash)
  - Locate memory pages given its hash

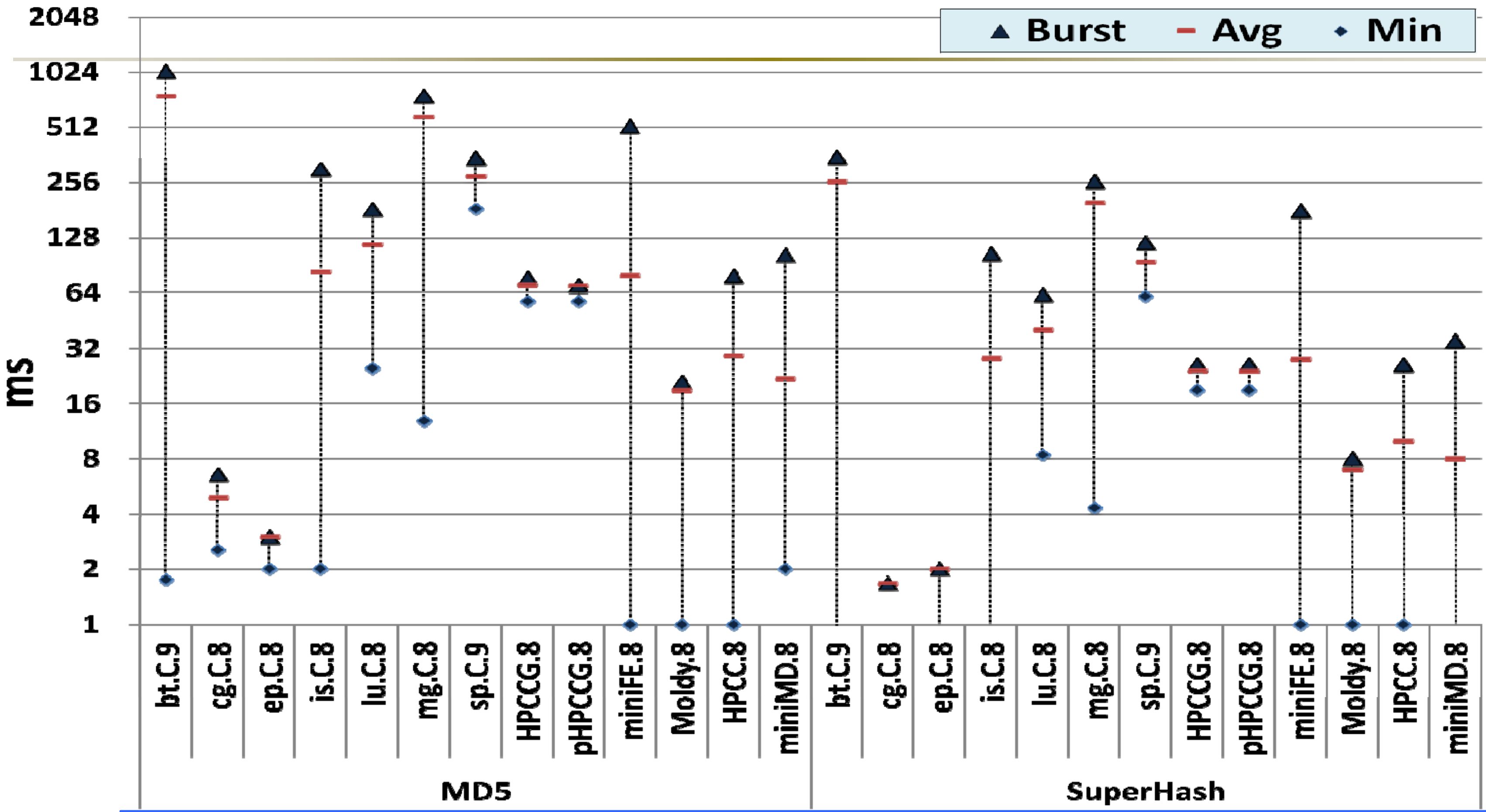
Proposed Approach

- **Back-end: System-wide DHT**
  - Collect and maintain all hashes of distinct memory pages in the system
  - Compute global sharing information
  - Handle queries from clients/services manager
  - Fault Tolerance of DHT
    - DHT is split into partitions
    - Each partition is stored in more than one node for redundant and fault tolerance
    - Synchronization/consistence of partition on update

Overhead Study

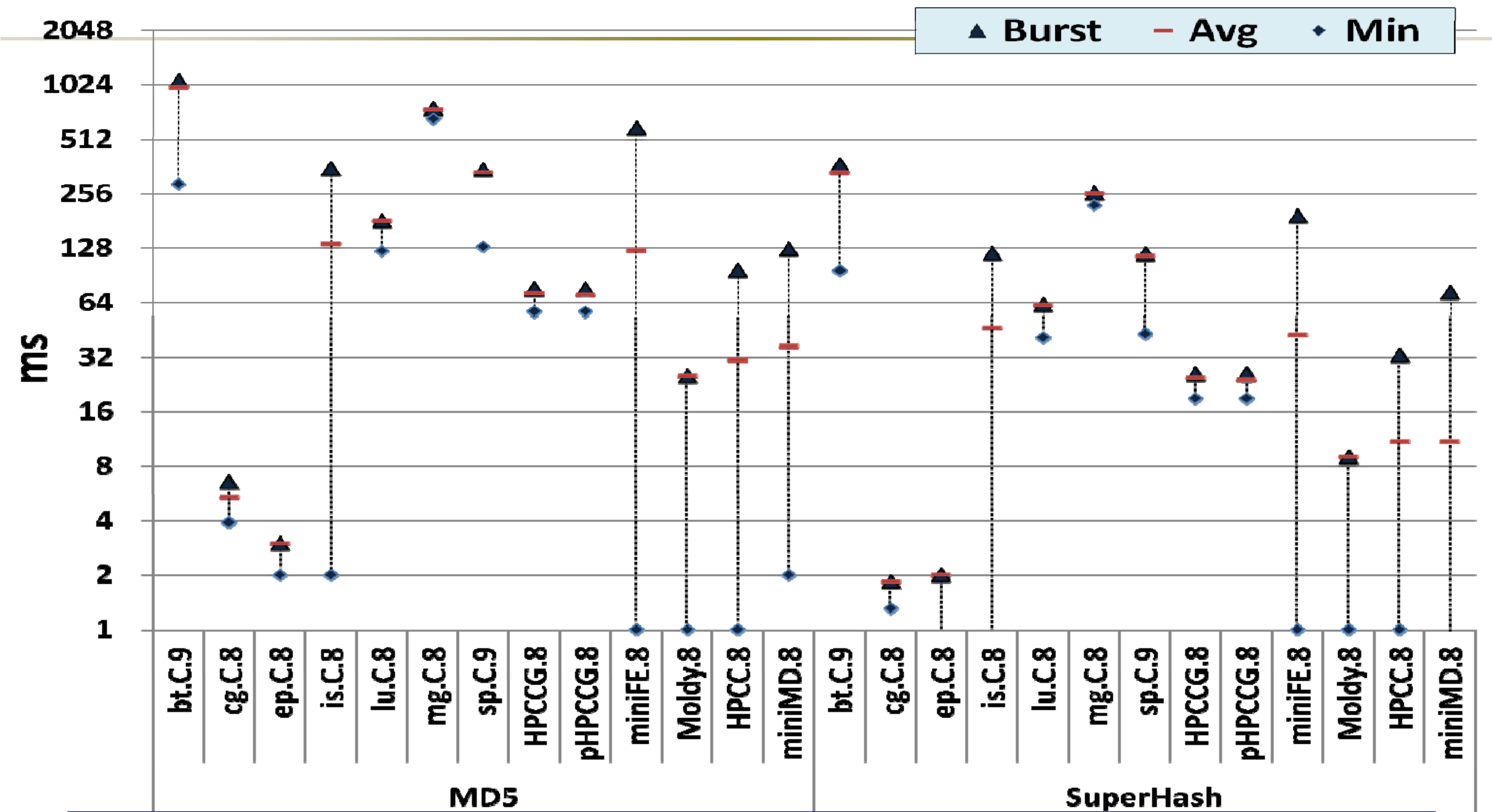
- System overheads of memory tracer
  - CPU overheads to scan and rehash memory pages
  - Network overheads to send hashes to DHT

Per Node CPU Overhead of the Memory Tracer (Interval: 2s)



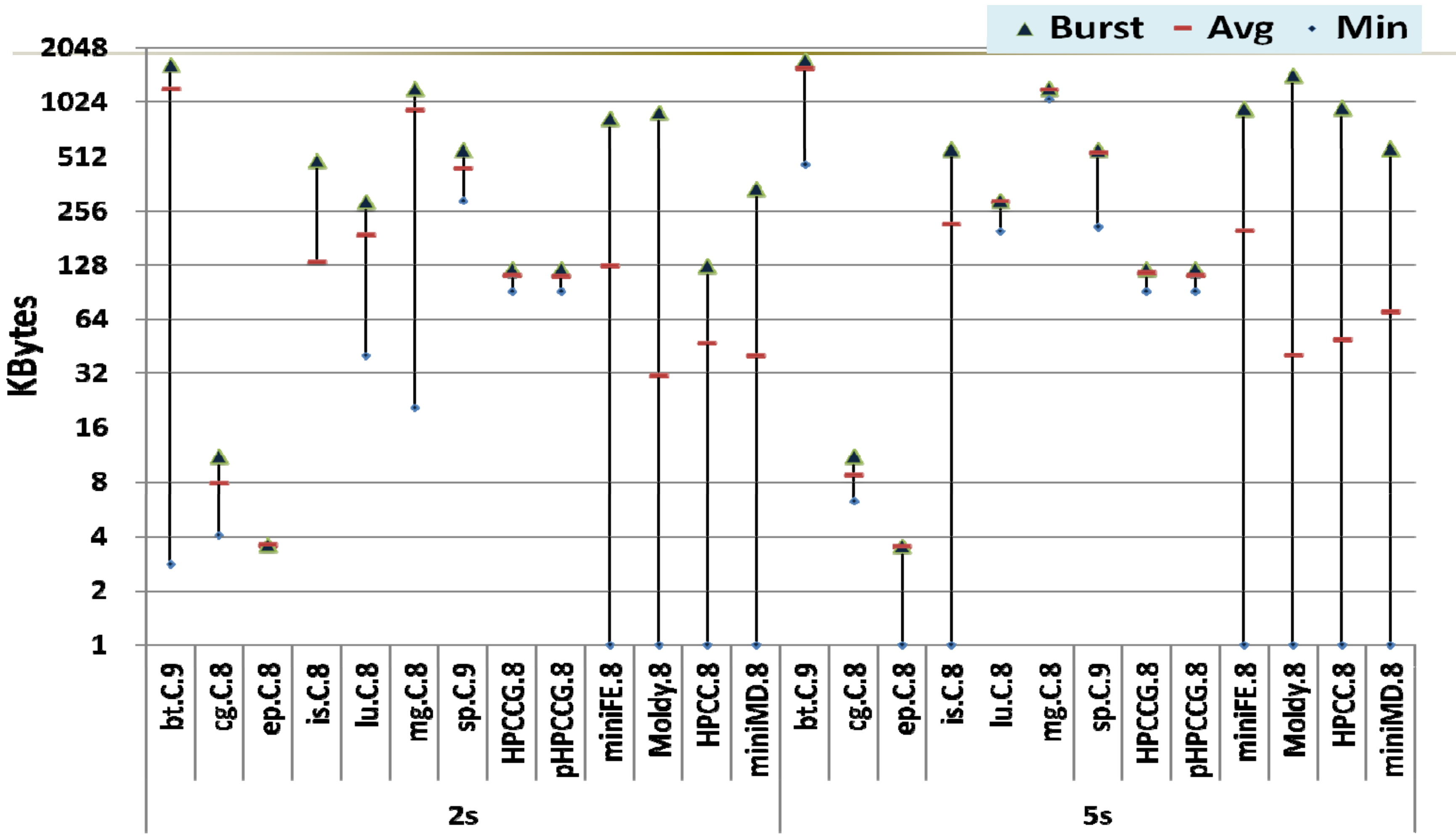
- Average: less than 128ms (6.4% overhead),
- Burst Updates: less than 512ms (25% overhead)
- SuperHash: 2% average/8.3% burst

Per Node CPU Overhead of the Memory Tracer (Interval: 5s)



- **Average: less than 128ms (2.6% overhead),**
- **Burst Updates: less than 512ms (10% overhead)**
- **SuperHash: <1%/3.3%**

Per Node Network Overhead of the Memory Tracer (Interval: 2s/5s)



- **Average: less than 512KB**
- **Burst Updates: less than 1500KB**

Related Works

- **Content-based page sharing**
  - Reduce memory usage for co-located VMs in individual host
  - Xen, Vmware, Difference Engine [OSDI'08]
  - SBLLmalloc [IPDPS'11]
- **Memory Buddies** **[VEE'09]**
  - Find better co-located decisions by VMs' memory footprint
  - Central control node
- **VM Migration**
  - Live Gang Migration **[HPDC'11]**
    - Optimization for migrating group of co-located VMs
  - VM Flock **[HPDC'11]** and Shrinker **[EuroPar'11]**
    - VM Migration across datacenter
    - Locate memory pages and disk blocks in destination datacenter

Summary

- Scalable tracking of *inter-node* memory content sharing would be a powerful primitive in parallel systems
  - Various services would greatly simplified and enhanced if such a system existed
- Intra-/Inter-node memory content sharing is common in scientific workloads
  - There are opportunities to exploiting the content sharing
- A proposed approach for scalable identifying and tracking of ***inter-node*** memory content sharing in large-scale systems

- Thanks, Questions??

- **Lei Xia**
  - Ph.D candidate, Northwestern University
  - *lxia@northwestern.edu*
  - *http://www.cs.northwestern.edu/~lxi990*

- V3VEE Project: http://v3vee.org