A Case for Dynamic Page Migration in Multiple-Writer Software DSM Systems

Thomas Repantis[1] Christos D. Antonopoulos[2]
Vana Kalogeraki[1] Theodore S. Papatheodorou[3]

[1]Department of Computer Science & Engineering University of California, Riverside

[2]Department of Computer Science The College of William & Mary

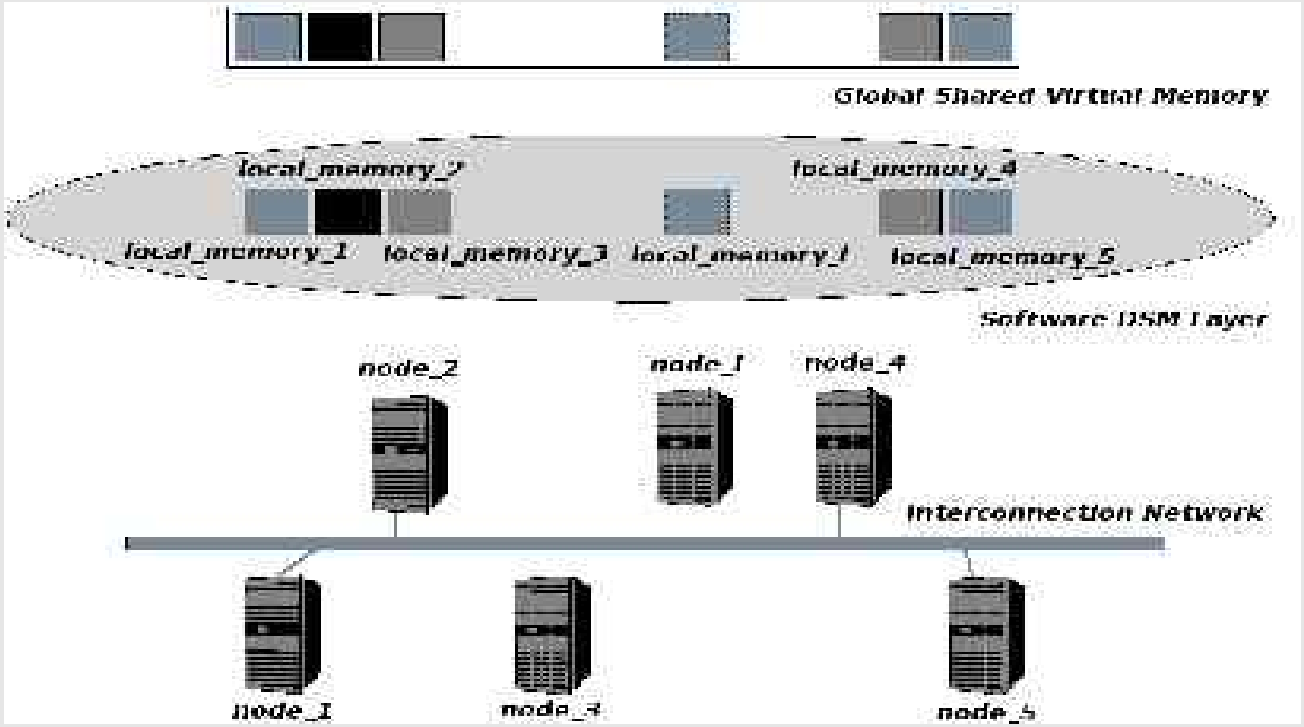
[3]Department of Computer Engineering & Informatics University of Patras, Greece

trep@cs.ucr.edu
http://www.cs.ucr.edu/~trep/

Outline

1. Introduction
2. Motivation
3. Multiple-Writer Dynamic Page Migration
   - On-Line Monitoring of Remote Diff Sizes
   - Migration Decisions
   - Propagation of Migration Decisions
   - Page Table Updates
4. Experimental Evaluation
5. Related Work
6. Conclusions and Future Work

Software DSMs



Distributed Shared Memory (DSM) architectures provide the notion of a globally shared address space to the programmer, although application processes may execute on different cluster nodes, with physically distributed memory

Home-Based SDSMs

In home-based software DSM systems each memory page has a home node:

- The home gathers all page modifications (diffs)
- All page access faults can be resolved with one request to the home of the particular page

Performance depends highly on the proper distribution of pages across nodes, with respect to the memory access pattern of the application:

- Remote page accesses incur extra communication
- Placement of memory pages together with the processes that access them is crucial

Motivation

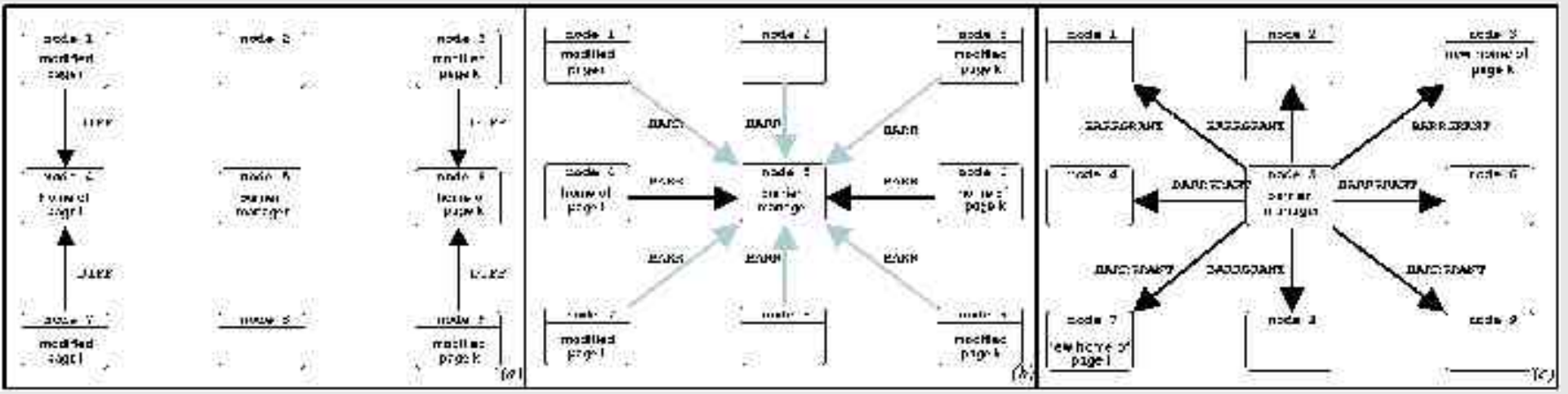
Dynamic page migration (explicitly moving the home of a page from some node to another at run-time):

- Reduces the average latency of memory accesses and the amount of network traffic, by increasing the locality
- Enables the system to adapt to the application' s memory access patterns
- Improves resource utilization, by taking into account memory availability
- Incurs lower overhead than thread migration

DPMIG:
A Multiple-Writer Dynamic Page Migration Protocol

- A node that heavily modifies a page gets a chance to become its new home
- Effective for both single- and multiple-writer memory access patterns
- Simple; utilizes existing communication messages
- 4 Phases:
  1. On-Line Monitoring of Remote Diff Sizes
  2. Migration Decisions
  3. Propagation of Migration Decisions
  4. Page Table Updates

Page Migration Operation



- According to the sizes of the modifications the migration decisions are made at the pages' homes
- The homes propagate the decisions to the barrier manager
- The barrier manager propagates the decisions to all other nodes

On-Line Monitoring of Remote Diff Sizes

- The home of each page:
  - Gathers all diffs coming from other nodes
  - Extracts and records the size of the modifications
- Local memory writes are detected but do not create diffs

Migration Decisions

- When a host reaches a barrier it identifies for all homed pages the main modifier and the extent of the modifications
- The page is migrated to the main modifier if:
  1. The modifications are above a certain threshold
  2. The page has not been modified by its current home since the last barrier
  3. The page was not migrated at the previous barrier

Propagation of Migration Decisions

- By propagating migration decisions only at barriers, we minimize the overhead of the migration mechanism (no extra synchronization phases or messages)
- We piggyback migration decisions in the existing “barrier request” and “barrier grant” messages

operation	from	to	data					
barrier	all nodes	barrier manager	lockid	vtnt 1	...	vtnt n	STOP	addr 1 hwmd 1 ... addr m hwmd m

operation	from	to	data					
barrier grant	barrier manager	all nodes	lockid	vtnt 1	...	vtnt n	STOP	addr 1 hwmd 1 ... addr k hwmd k

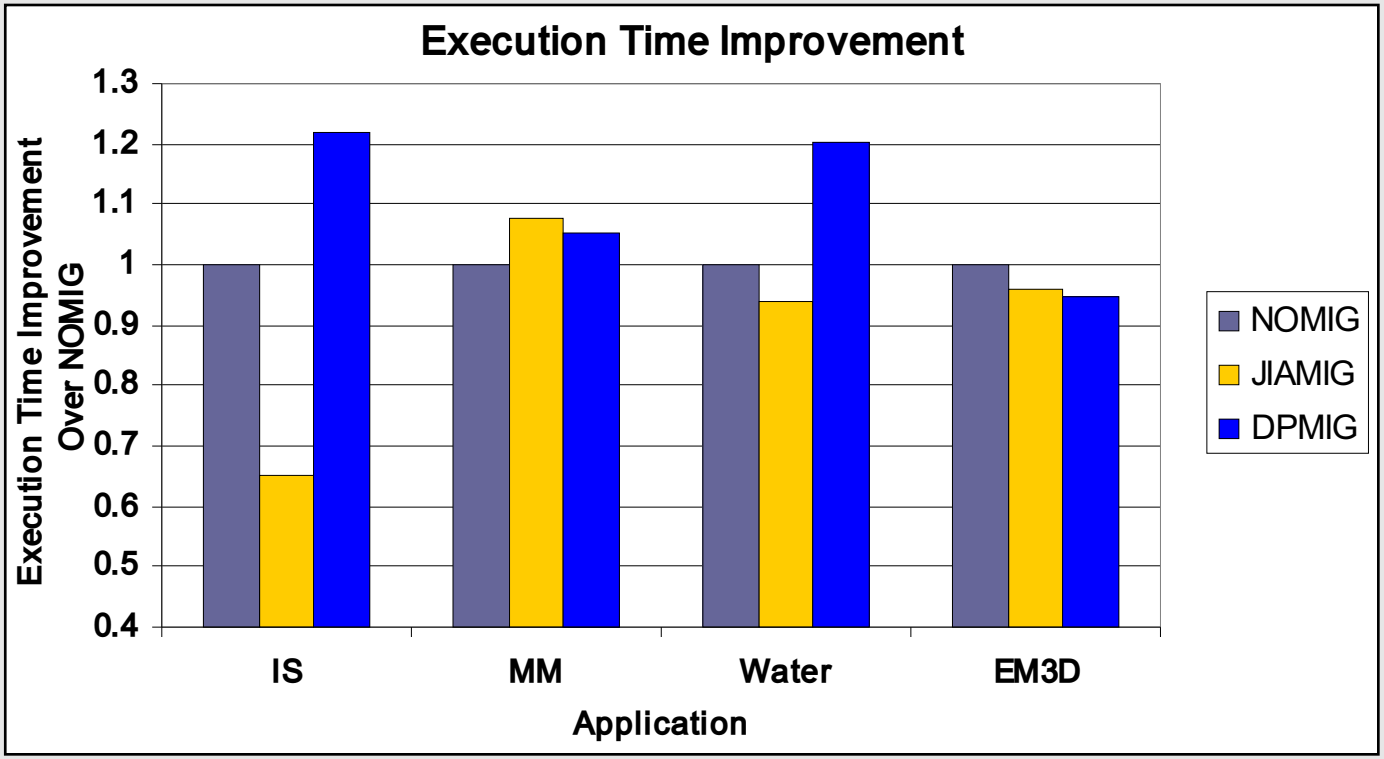
Page Table Updates

- Each host updates its page tables (global, home, cache) according to the migration decisions
- If the new home was a single modifier and has a valid copy of the page locally cached, no page transfer is needed
- We ensure that the page is not accessed at the new home and not unmapped at the old home before the migration is complete

Experimental Setup

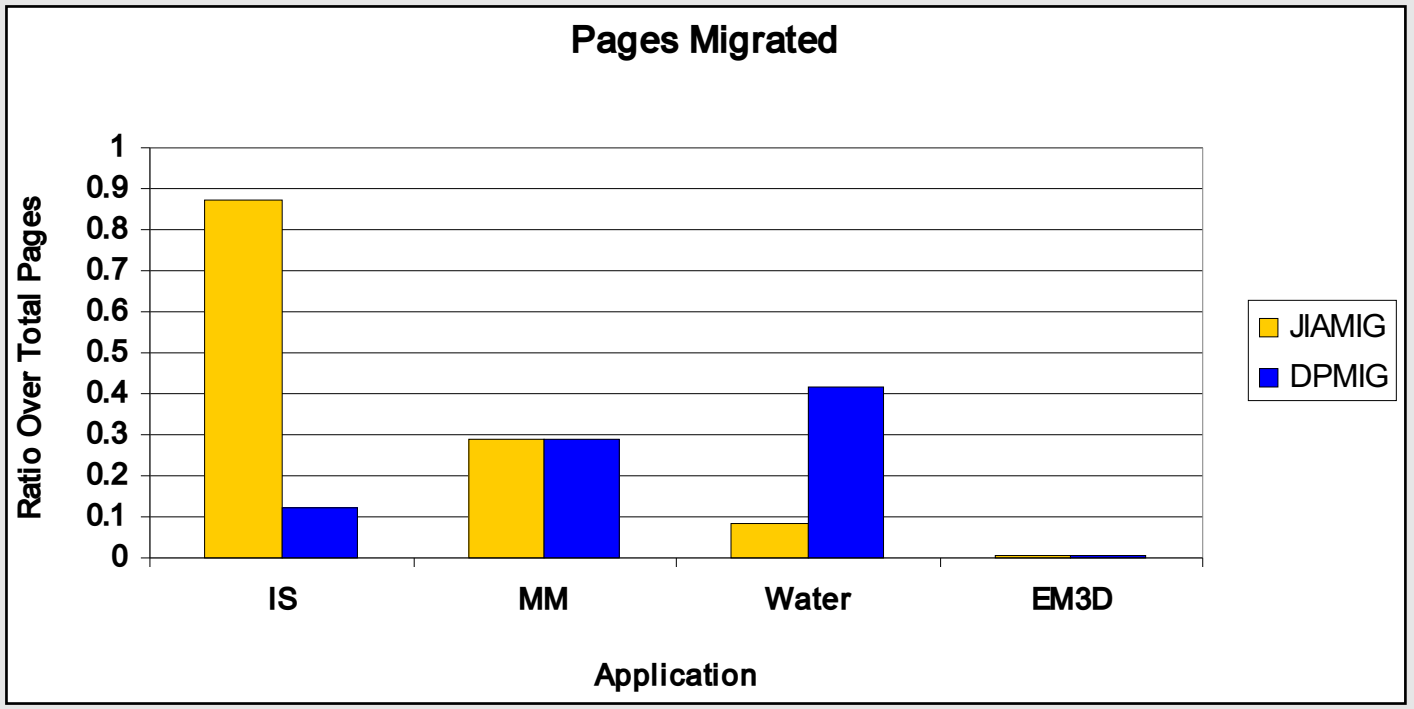
- We implemented our protocol (DPMIG) on the JIAJIA SDSM and compared it to JIAJIA's single-writer page migration protocol (JIAMIG) and to operation without migration (NOMIG)
- Cluster of 8 AMD Athlon XP nodes, connected with 100MBps Ethernet, running Linux 2.6.7
- 4 application benchmarks: Water from SPLASH-2, IS from NAS, MM - a matrix multiplication, and EM3D from the JIAJIA distribution

Execution Time Improvement



- DPMIG achieves improvement up to 22%
- Only MM offers a chance for a single-writer page migration protocol

Pages Migrated



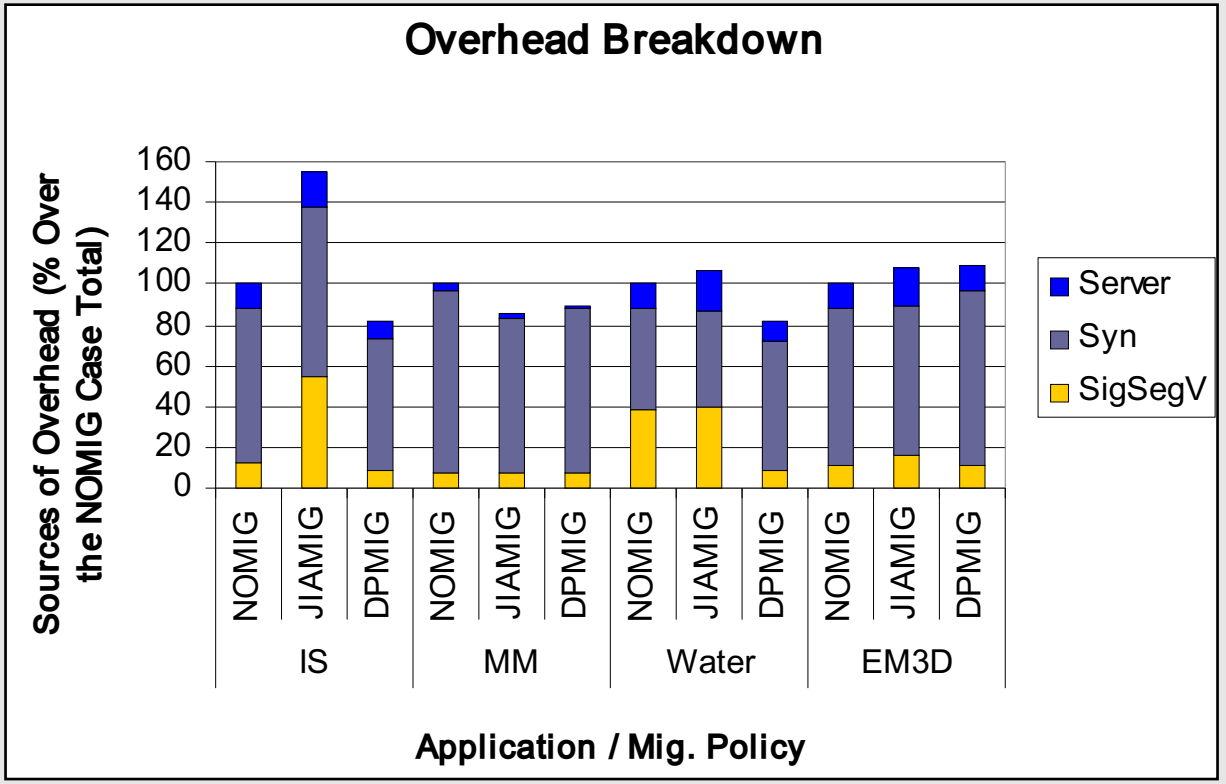
- The ratio of home migrations is not by itself a safe indicator of a protocol's effectiveness
- EM3D allows for an optimal off-line initial allocation of shared data

Network Traffic

	IS	MM	Water	EM3D
NOMIG	5266.05	816.52	320.15	165.88
JIAMIG	5264.35	81.57	320.46	222.74
DPMIG	4065.37	81.64	223.09	174.84

- Page placement by DPMIG achieves an average 30.51% reduction in network traffic compared to NOMIG, while JIAMIG achieves 9.95%
- Even in cases where the DPMIG has no room to improve locality (EM3D) it will not deteriorate performance
- Traffic reduction does not improve execution time if the data transfer does not happen in the execution path (MM)

Overhead Breakdown



- DPMIG reduces overall overhead by 16.12% on average (38.84% SigSegV, 29.06% Server, 11% Syn) compared to NOMIG, while JIAMIG increases it by 38.65% (227.10% SigSegV, 45.74% Server, 4.09% Syn)

Related Work

- Page migration protocols that detect single-writer memory access patterns:
  - Fang et al (Cluster'04)
  - MHLRC (ICPP'99)
  - Orion (WSDSM'99)
  - JIAJIA (WSDSM'99)
- Multi-homed and homeless protocols:
  - AHLRC (IPDPS'04)
  - JUMP (PDPTA'99)
- Page migration on NUMA hardware DSMs:
  - Nikolopoulos et al (ICS'00, SC'00, ICPP'00)

Conclusions and Future Work

- DPMIG:
  - A simple dynamic page migration mechanism for home-based SDSMs
  - Applicable for both single- and multiple-writer memory access patterns
  - Strives to keep migration-related communication minimal
- Future Work:
  - Automatic calculation of the migration threshold at run-time
  - Evaluation of prefetching and precomputation in SDSMs

Thank You!

Questions / Comments?

A Case for Dynamic Page Migration in Multiple-Writer Software DSM Systems

Thomas Repantis[1] Christos D. Antonopoulos[2]
Vana Kalogeraki[1] Theodore S. Papatheodorou[3]

[1]Department of Computer Science & Engineering University of California, Riverside

[2]Department of Computer Science The College of William & Mary

[3]Department of Computer Engineering & Informatics University of Patras, Greece

trep@cs.ucr.edu
http://www.cs.ucr.edu/~trep/