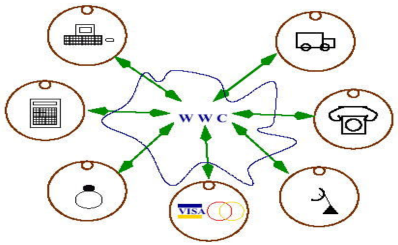
IOS: A Middleware for Decentralized Distributed Computing

Boleslaw Szymanski
Kaoutar El Maghraoui, Carlos Varela

Department of Computer Science
Rensselaer Polytechnic Institute
http://www.cs.rpi.edu/wwc

September 2005



Rensselaer

General Concepts

IOS Architecture

IOS Overview

- The Internet Operating System (IOS) is a decentralized middleware framework that provides:
  - Opportunistic load balancing capabilities
  - Resource profiling
  - Application-level profiling
- Goal:
  - Automatic Reconfiguration of applications in dynamic environments (e.g., Computational Grids)
  - Scalability to worldwide execution environments
  - Modular architecture enabling evaluation of different load balancing and resource profiling strategies

Reconfigurable Distributed Applications

- **Reconfigurable applications have several needs:**
  - **Availability**
    - resilience to failures
  - **Scalability**
    - Ability to use new resources while the application is executing
  - **Adaptability**
    - Ability to adapt to load fluctuations to achieve high performance
  - **Autonomy**
    - Self-configuration, self-healing, and self-organization

Resource Management Requirements

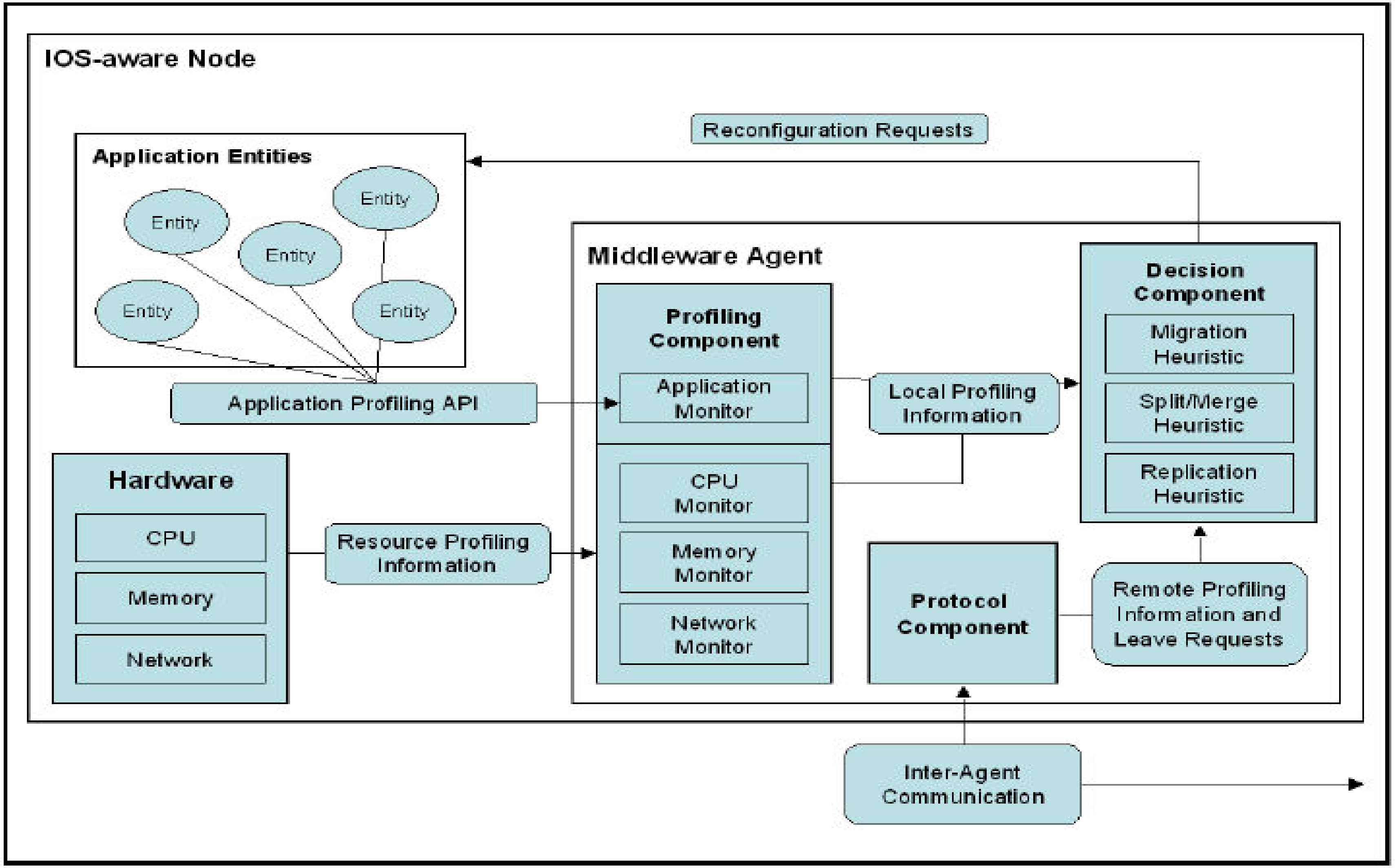
- **Resource Allocation**
  - **Pre-execution resource discovery**
  - **Pre-execution allocation of resources**
    - Could be informed: an initial good match with application's requirements
    - Could be non-informed: use any available resources and rely on the middleware to find the optimal allocation
- **Resource Reallocation/Reconfiguration**
  - **Dynamically and repeatedly modifying the mapping between application components and physical resources to respond to the dynamic behavior of the execution environment**
    - Need to support process/data migration
- **Resource Profiling**
  - **Continuous evaluation of both application-level requirements and resource-level characteristics.**

IOS Middleware

- Middleware Agents:
  - Encapsulate components for resource profiling and reconfiguration policies
  - Interface with high level applications

- Interfacing with IOS agents
  - Applications need to implement specific APIs to interface with IOS agents to exchange profile information and reconfiguration requests.
  - Applications need to support some level of component migration

IOS Architecture

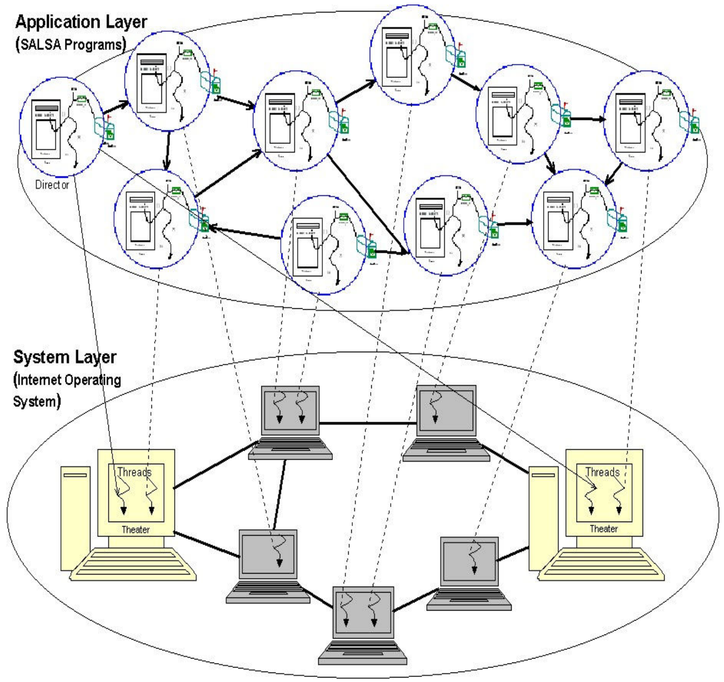


IOS Architecture

- **IOS middleware layer**
  - **A Resource Profiling Component**
    - Captures information about actor, network topologies and available resources
  - **A Decision Component**
    - Takes migration, split/merge, or replication decisions based on profile information
  - **A Protocol Component**
    - Performs communication with other agents in a virtual network (e.g., peer-to-peer, cluster-to-cluster, centralized)

Using the IOS middleware

- Start IOS Peer Servers: a mechanism for peer discovery
- Start a network of IOS theaters
- Write SALSA programs with actors and extend all actors to autonomous actors
- Bind autonomous actors to theaters
- IOS automatically reconfigures the location of actors in the network for improved performance of the application.
- IOS supports the dynamic addition and removal of theaters



Resource Sensitive Model

- Decision components use a resource sensitive model parameterized with application's profile information to decide how to balance the resources' consumption

- Reconfiguration decisions
  - Where to migrate
  - When to migrate
  - How many entities to migrate

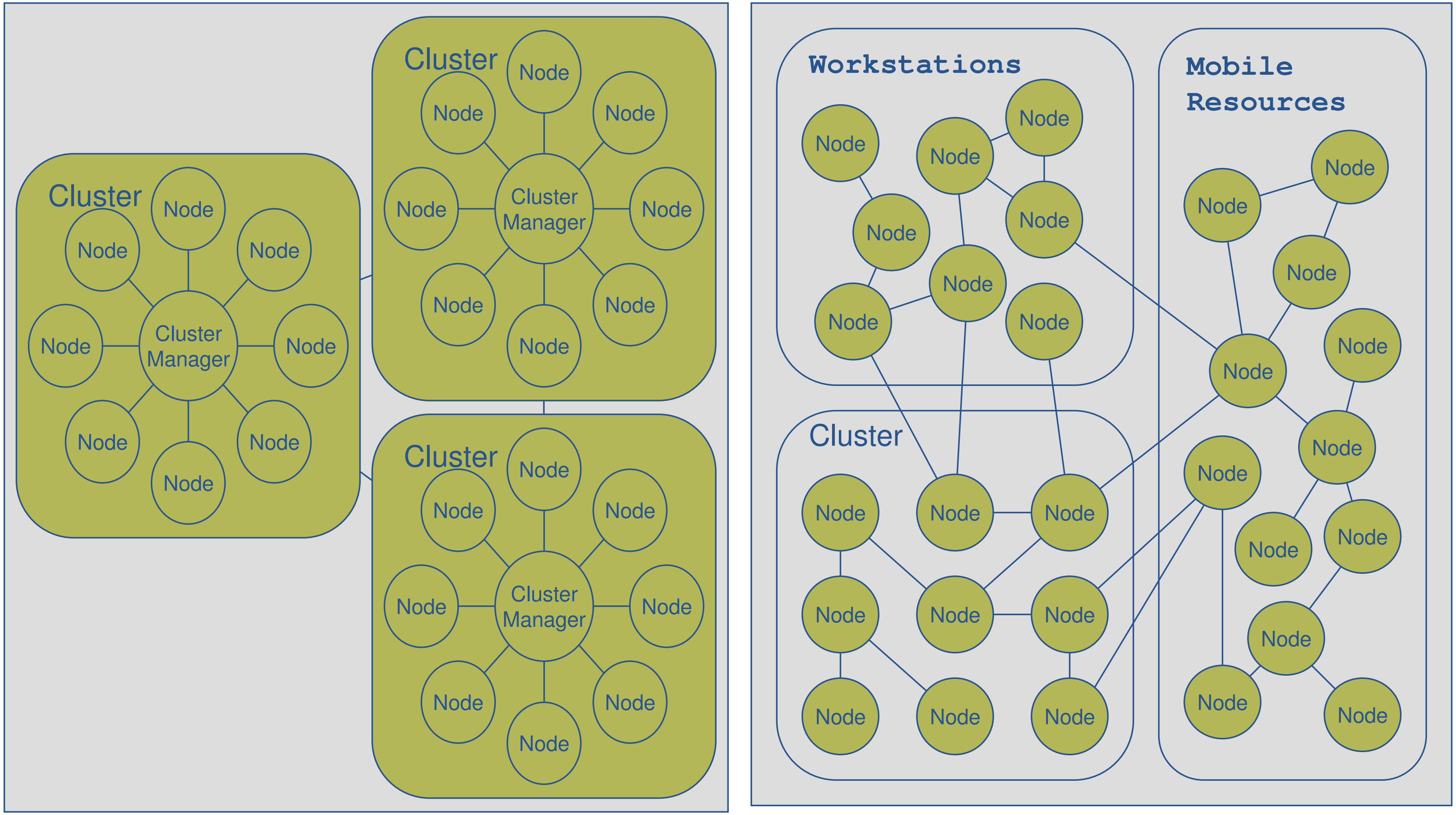
Virtual Topologies of IOS Agents

- **Agents organize themselves in various network-sensitive virtual topologies to sense the underlying physical environments**

- **Peer-to-peer topology: agents form a p2p network to exchange profiled information.**

- **Cluster-to-cluster topology: agents organize themselves in groups of clusters. Cluster managers form a p2p network.**

C2C vs. P2P topologies



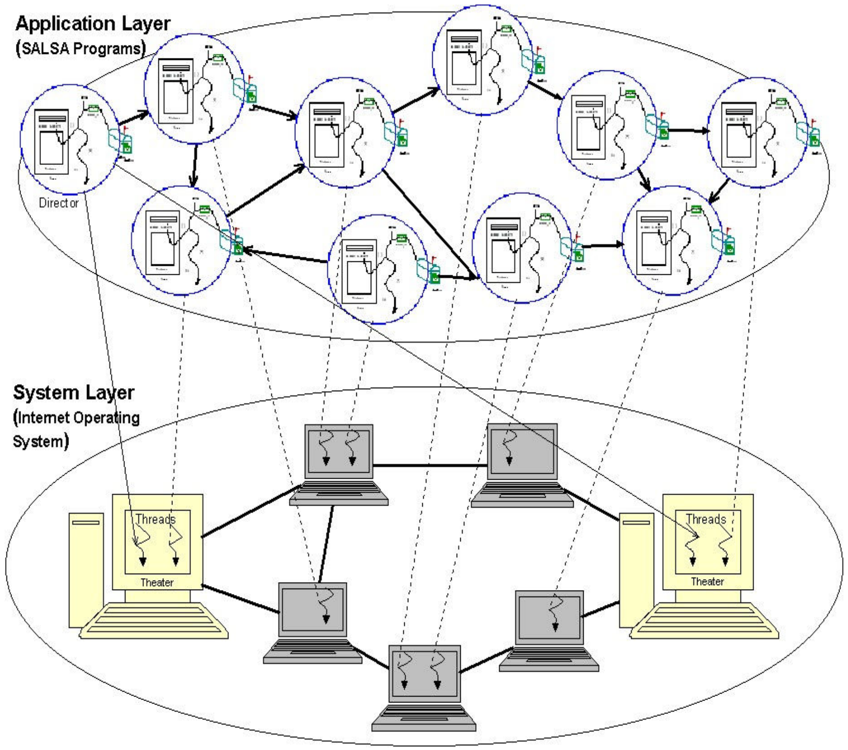
Rensselaer

IOS Load Balancing Strategies

- **IOS modular architecture enables using different load balancing and profiling strategies, *e.g.:***
  - **Round-robin (RR)**
  - **Random work-stealing (RS)**
  - **Application topology-sensitive work-stealing (ATS)**
  - **Network topology-sensitive work-stealing (NTS)**

Random Stealing (RS)

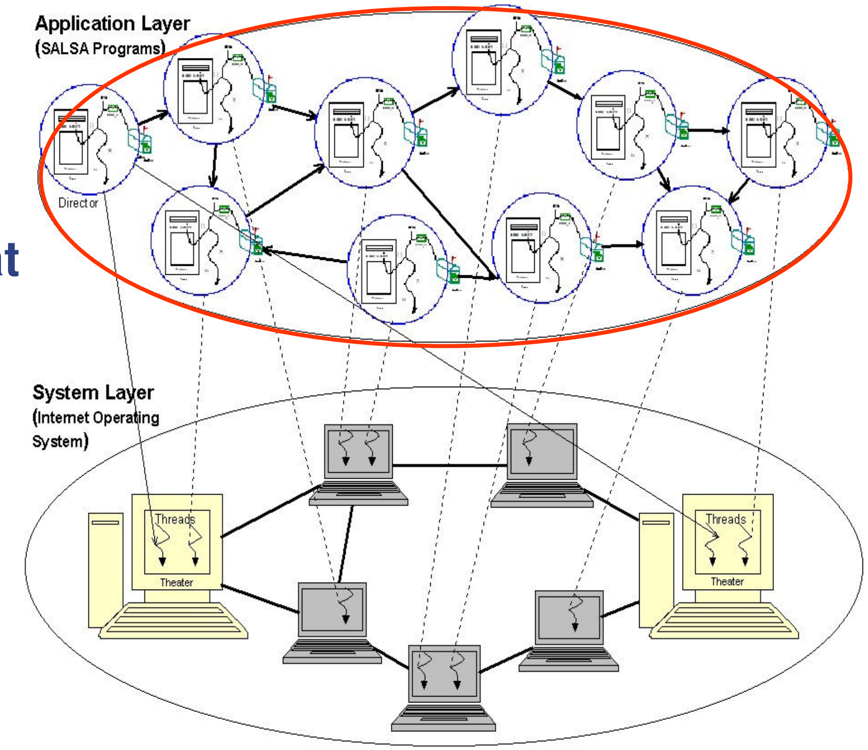
- Based on Cilk's random work stealing
- Lightly-loaded nodes periodically send work steal packets to randomly picked peer theaters
- Application entities migrate from highly loaded theaters to lightly loaded theaters
- Simple strategy: no broadcasts required
- Stable strategy: it avoids additional traffic on overloaded networks



Rensselaer

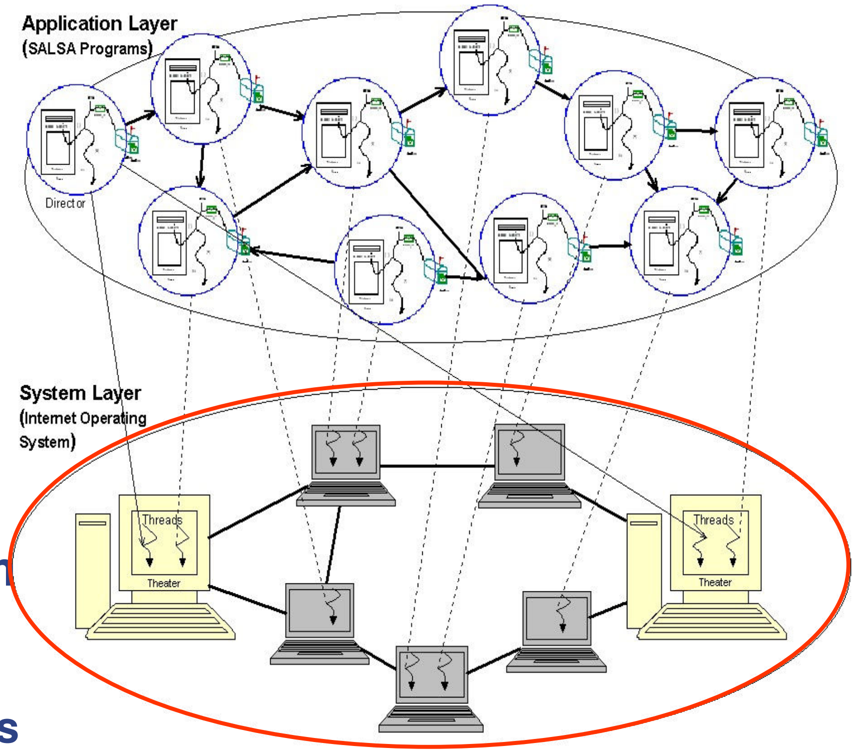
Application Topology-Sensitive Work-Stealing (ATS)

- An extension of RS to collocate components that communicate frequently
- Decision agent picks the component that will minimize inter-node communication after migration, based on
  - Location of acquaintances
  - Collected communication history
- Tries to minimize the frequency of remote communication to improve overall system throughput



Network Topology-Sensitive Work-Stealing (NTS)

- **An extension of ATS to take the network topology and performance into consideration**
- **Periodically profile end-to-end network performance among peer theaters**
  - **Latency**
  - **Bandwidth**
- **Tries to minimize the cost of remote communication improving overall system throughput**
  - **Tightly coupled entities stay within reasonably low latencies/ high bandwidths**
  - **Loosely coupled entities can flow more freely**



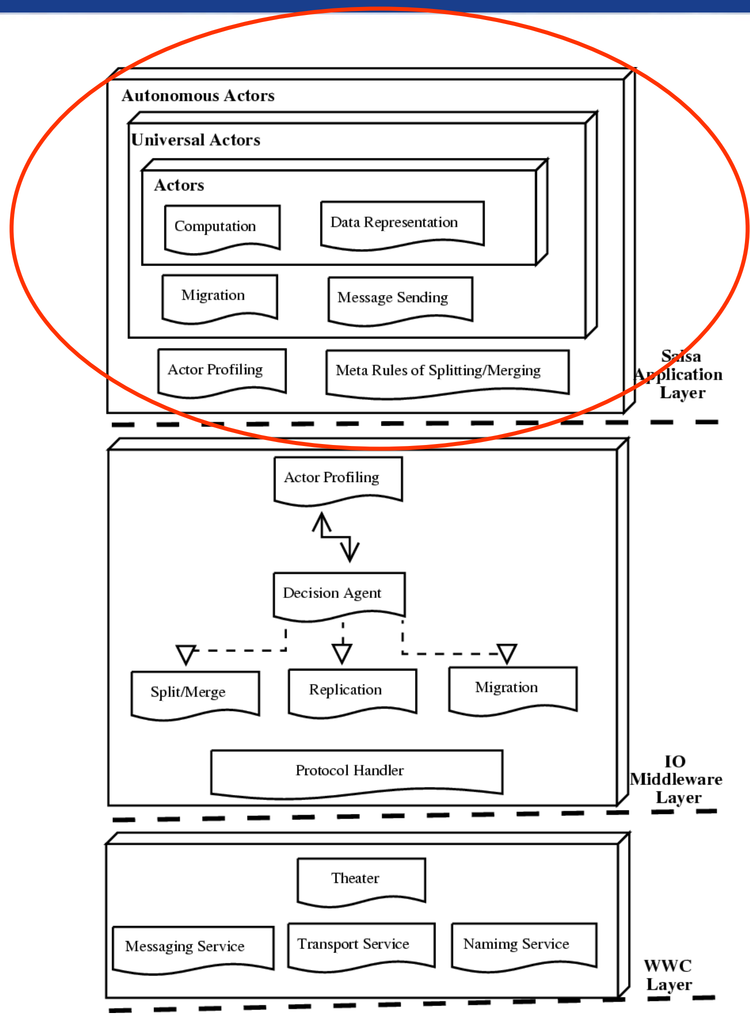
Application Case Studies

Actors and MPI programs

Rensselaer

Salsa and Autonomous Actors

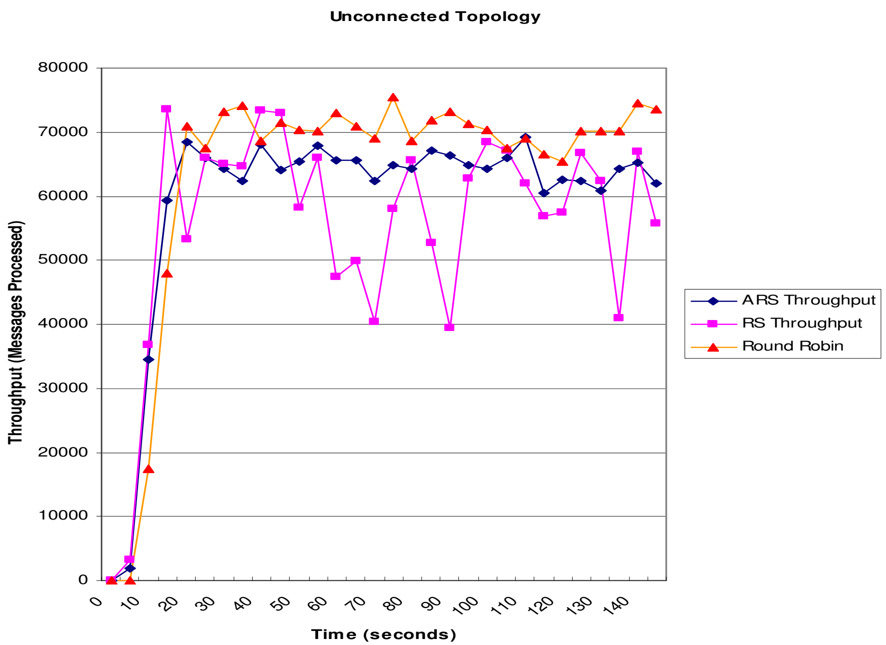
- **SALSA application layer**
  - Programming language constructs for actor communication, migration, and coordination.
- **Actors**
  - Unit of concurrency
  - Asynchronous message passing
  - State encapsulation
- **Universal actors**
  - Universal names
  - Location/theater
  - Ability to migrate between theaters
- **Autonomous actors**
  - Performance profiling to improve quality of service
  - Autonomous migration to balance computational load
  - Split and merge to tune granularity
  - Replication to increase fault tolerance

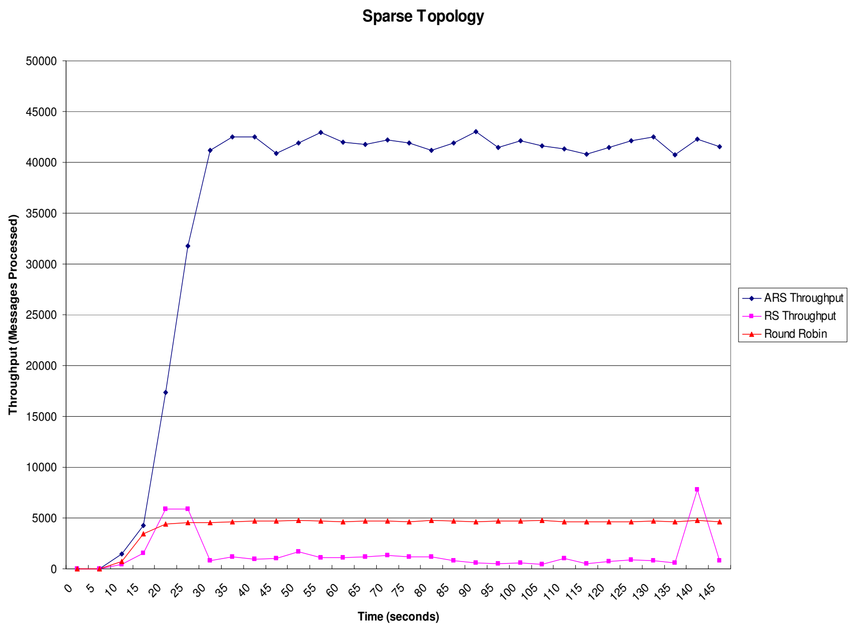


Rensselaer

Preliminary Results---Unconnected/Sparse

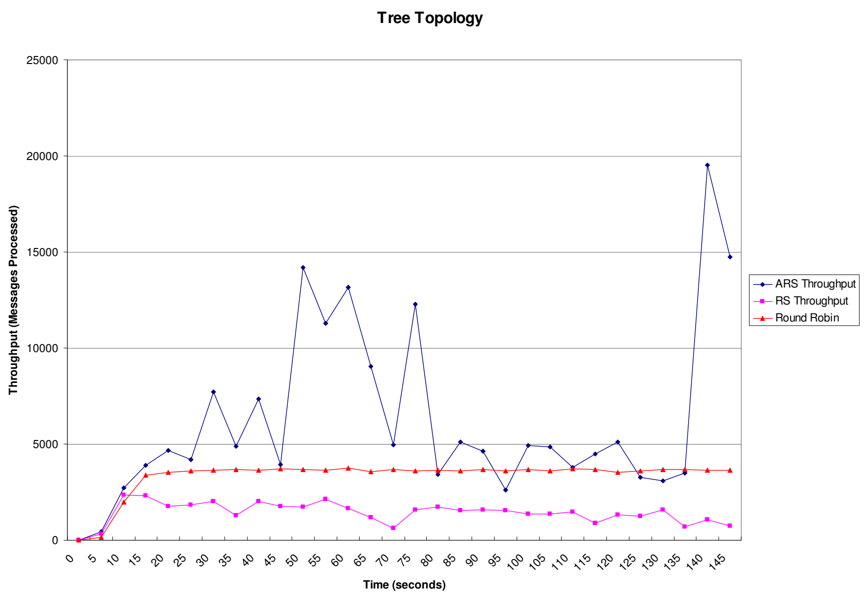
- Load balancing experiments use RR, RS and ATS
- Applications with diverse inter-actor communication topologies
  - Unconnected, sparse, tree, and hypercube actor graphs

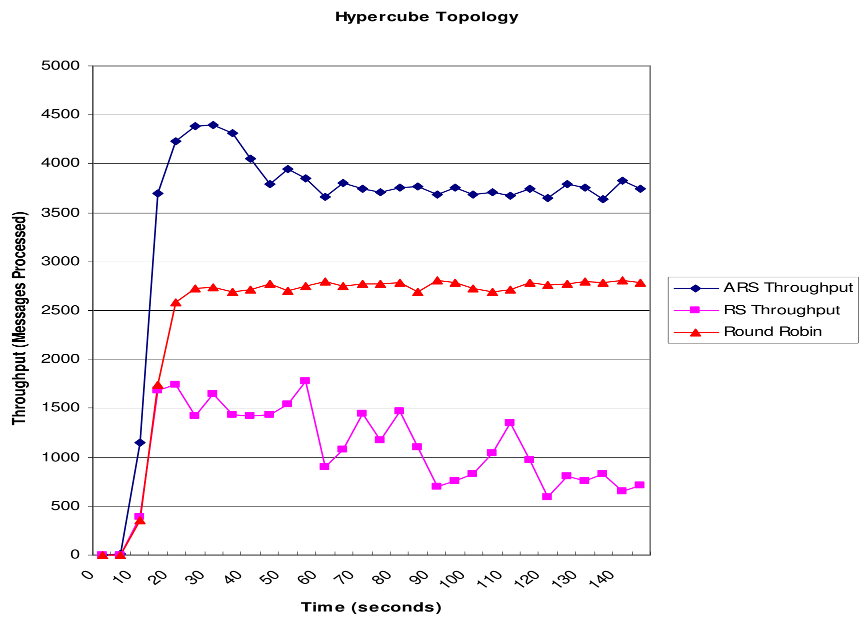




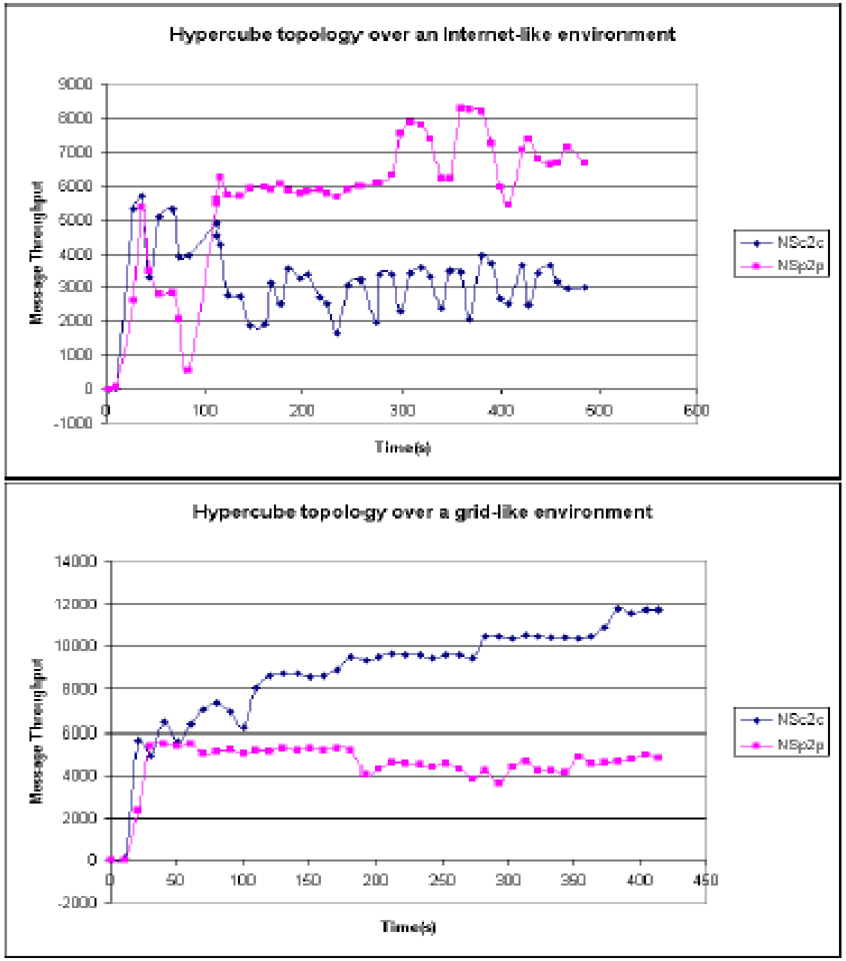
Tree and Hypercube Topology Results

- RS and ATS do not add substantial overhead to RR
- ATS performs best in all cases with some interconnectivity

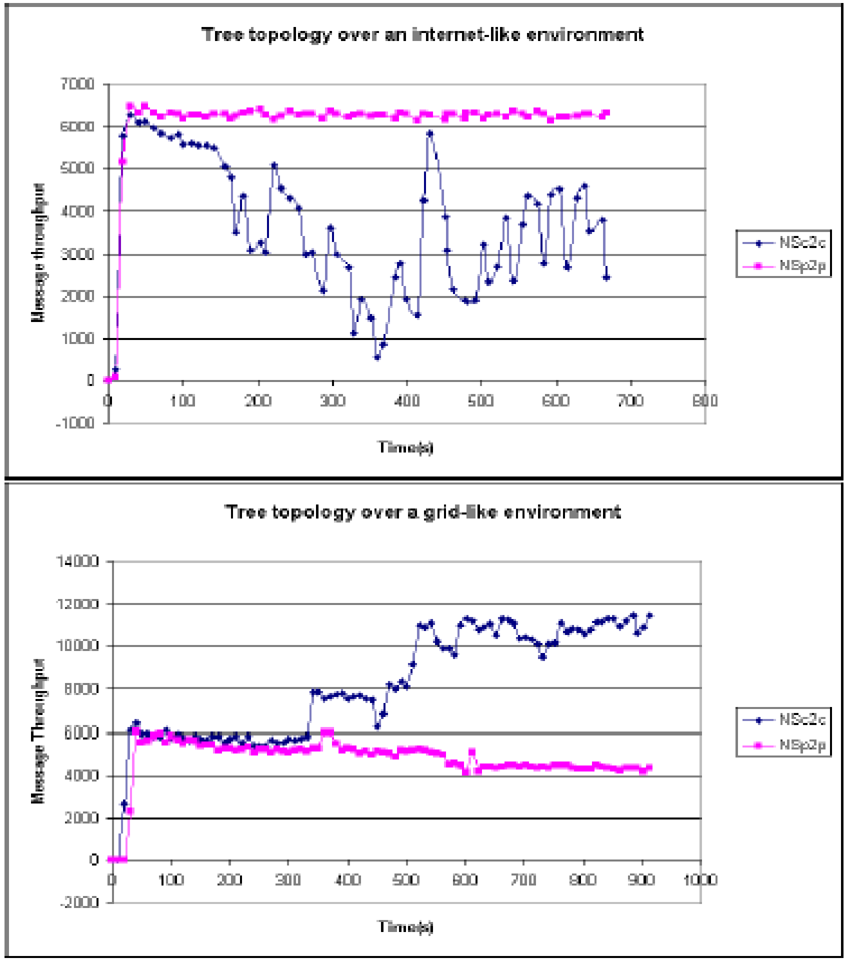




Results for applications with high communication to computation ratio

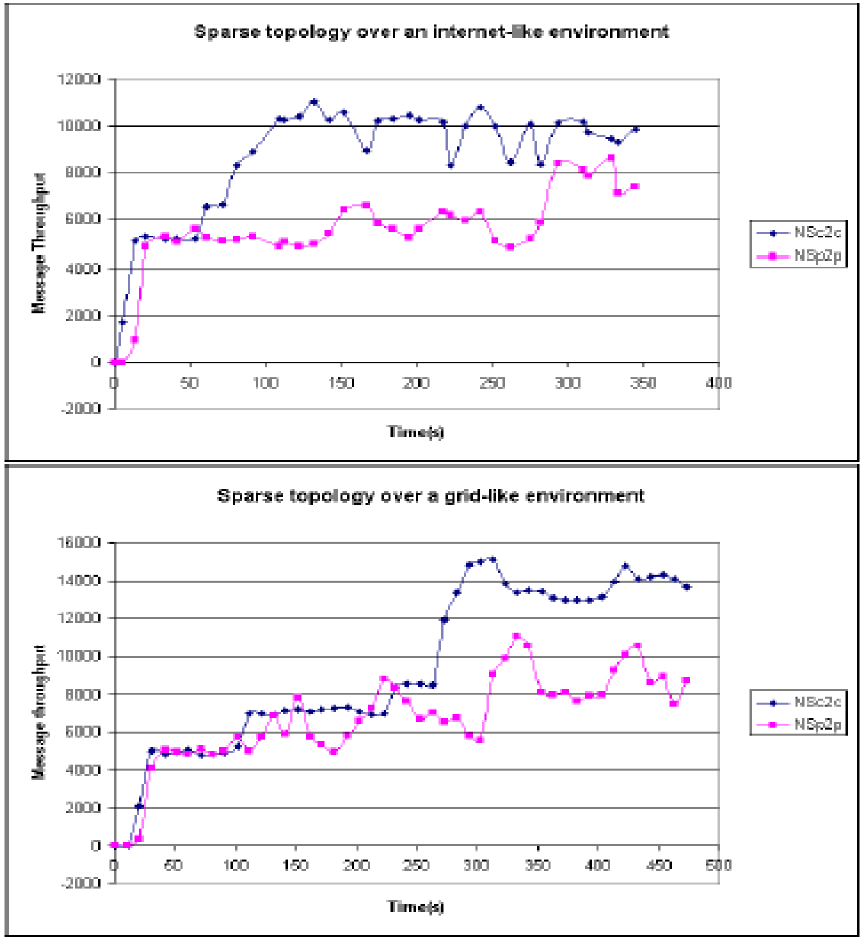


The hypercube application topology on Internet- and Grid-like environments.

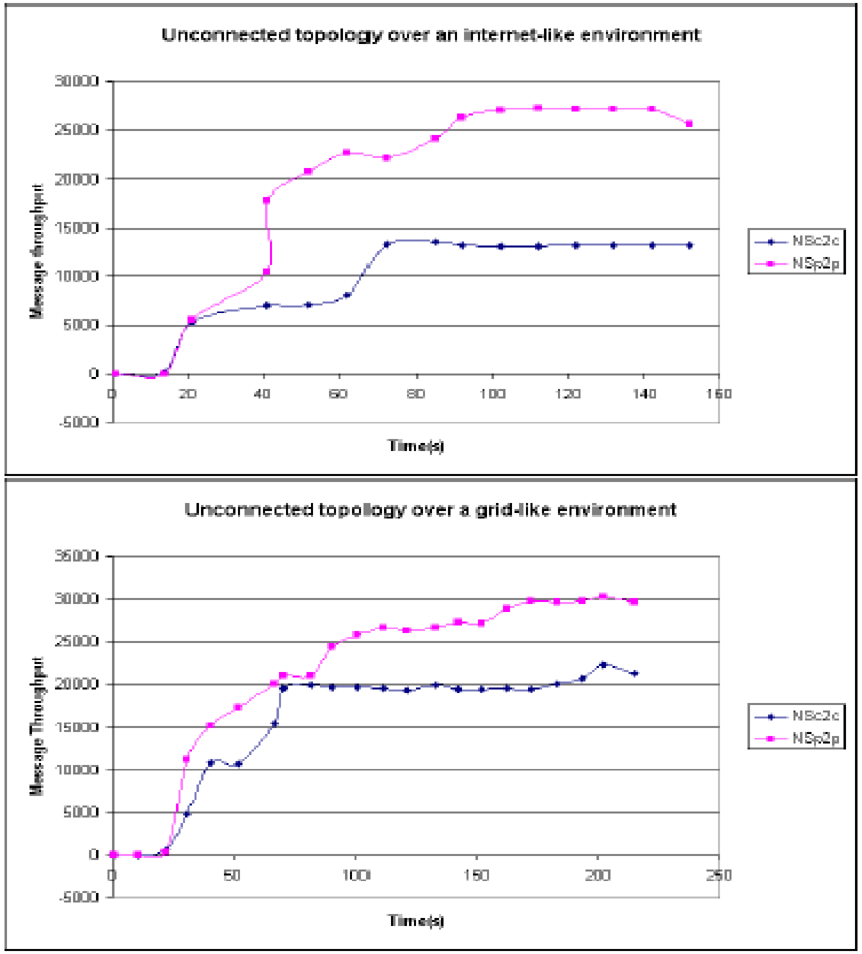


The tree application topology on Internet- and Grid-like environments.

Results for applications with low communication-to-computation ratio



The sparse application topology on Internet- and Grid-like environments.



The unconnected application topology on Internet- and Grid-like environments.

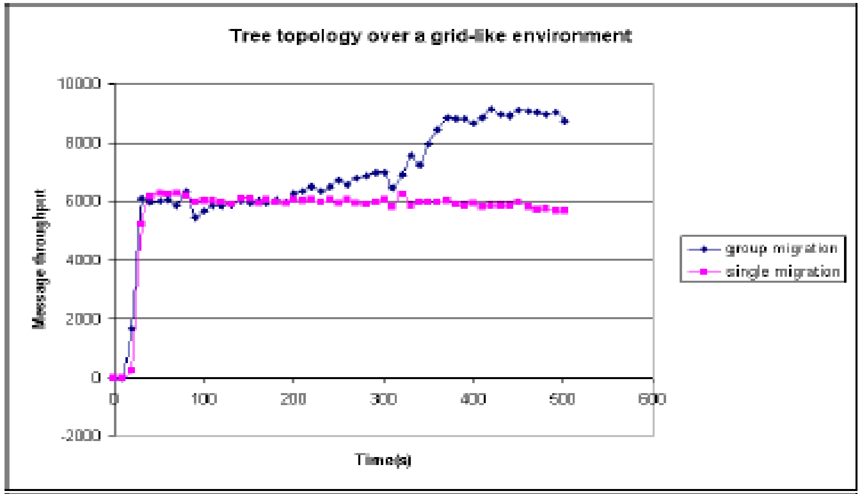
Load Balancing Strategies for Internet-like and Grid-like Environments

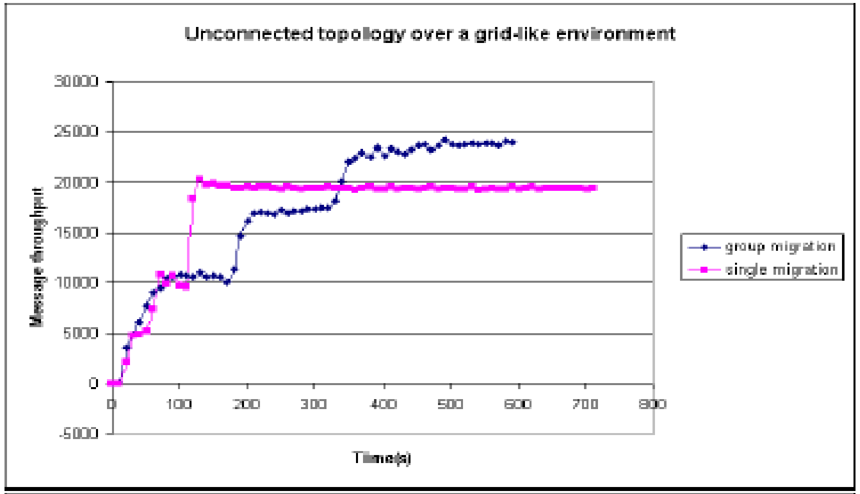
- **Simulation results show that:**
  - **The peer-to-peer protocol performs better for applications with high communication-to-computation ratio in Internet-like environments**
  - **The cluster-to-cluster protocol performs better for applications with low communication-to-computation ratio in Grid-like environments**

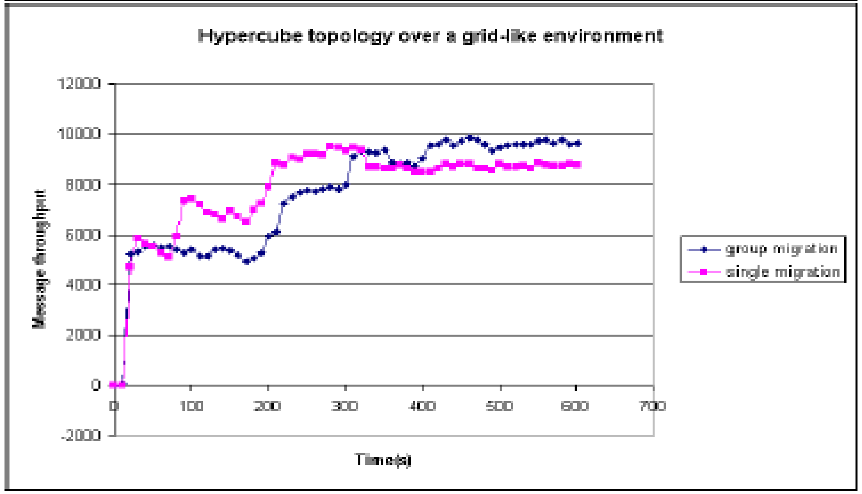
Migration Policies

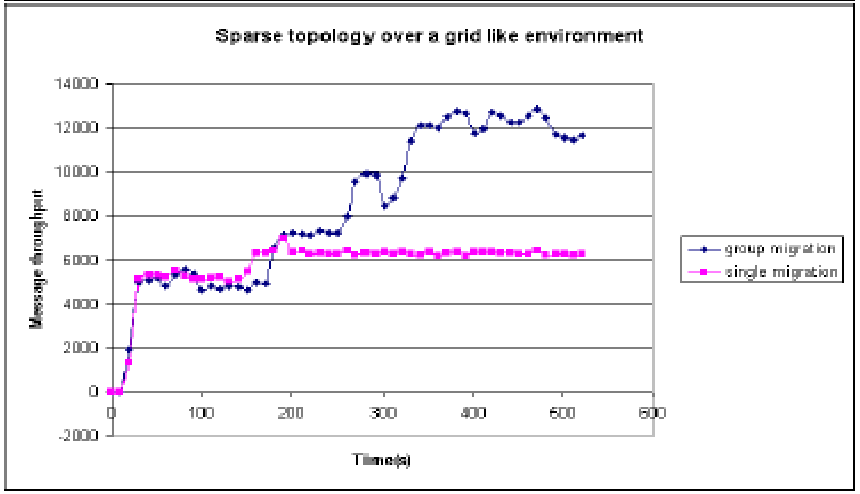
- Group Migration performs better for the four application topologies.
- Single Migration has a more stable behavior of the application's topology throughput
- Future Work: Evaluation of migration policies for different sizes of actors.

Single vs. Group Migration







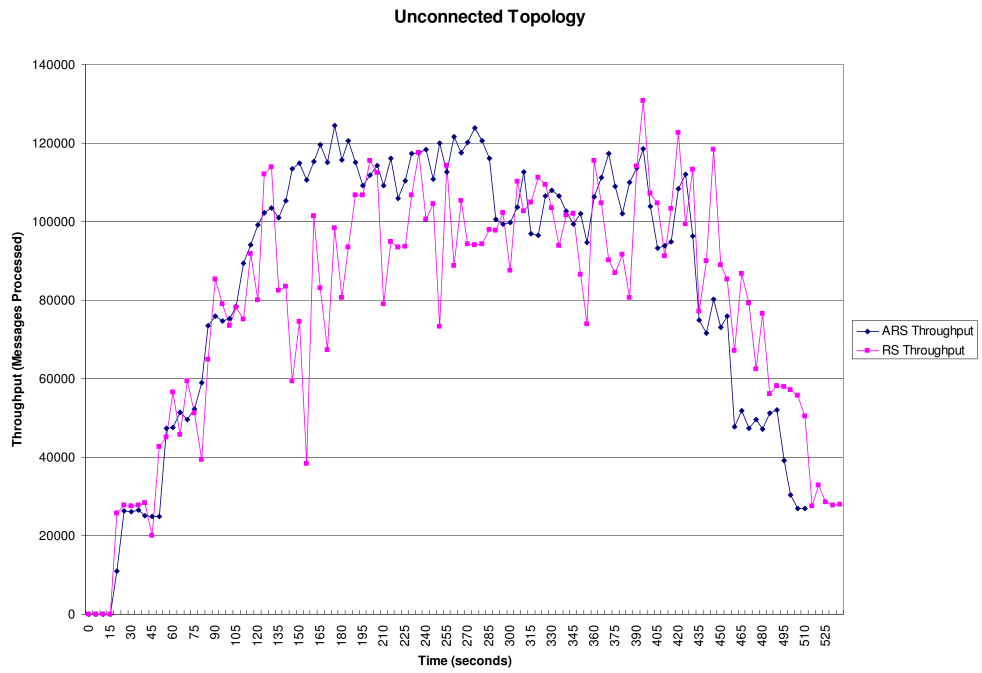


Single vs. group migration for the tree and hypercube application topologies.

Single vs. group migration for the unconnected and sparse application topologies.

Dynamic Networks

- **Theaters were added and removed dynamically to test scalability.**
- **During the 1st half of the experiment, every 30 seconds, a theater was added.**
- **During the 2nd half, every 30 seconds, a theater was removed**
- **Throughput improves as the number of theaters grows.**



Rensselaer

Experiences with MPI programs

- **Motivation**
  - **Message Passing interface (MPI) is the de-facto standard to implement single program multiple data (SPMD) parallel applications.**
  - **Widely used in the scientific community**

- **MPI Challenges on Dynamic Grids**
  - **Tailored for tightly coupled systems/static environments**
  - **Dynamic reconfiguration**
    - Process mobility
    - Scale to accommodate joining nodes
    - Shrink to accommodate leaving nodes
  - **Fault-tolerance**
  - **Level of involvement of end-users in effective resource management (e.g. Application-level load balancing)**

MPI/IOS Research Goals

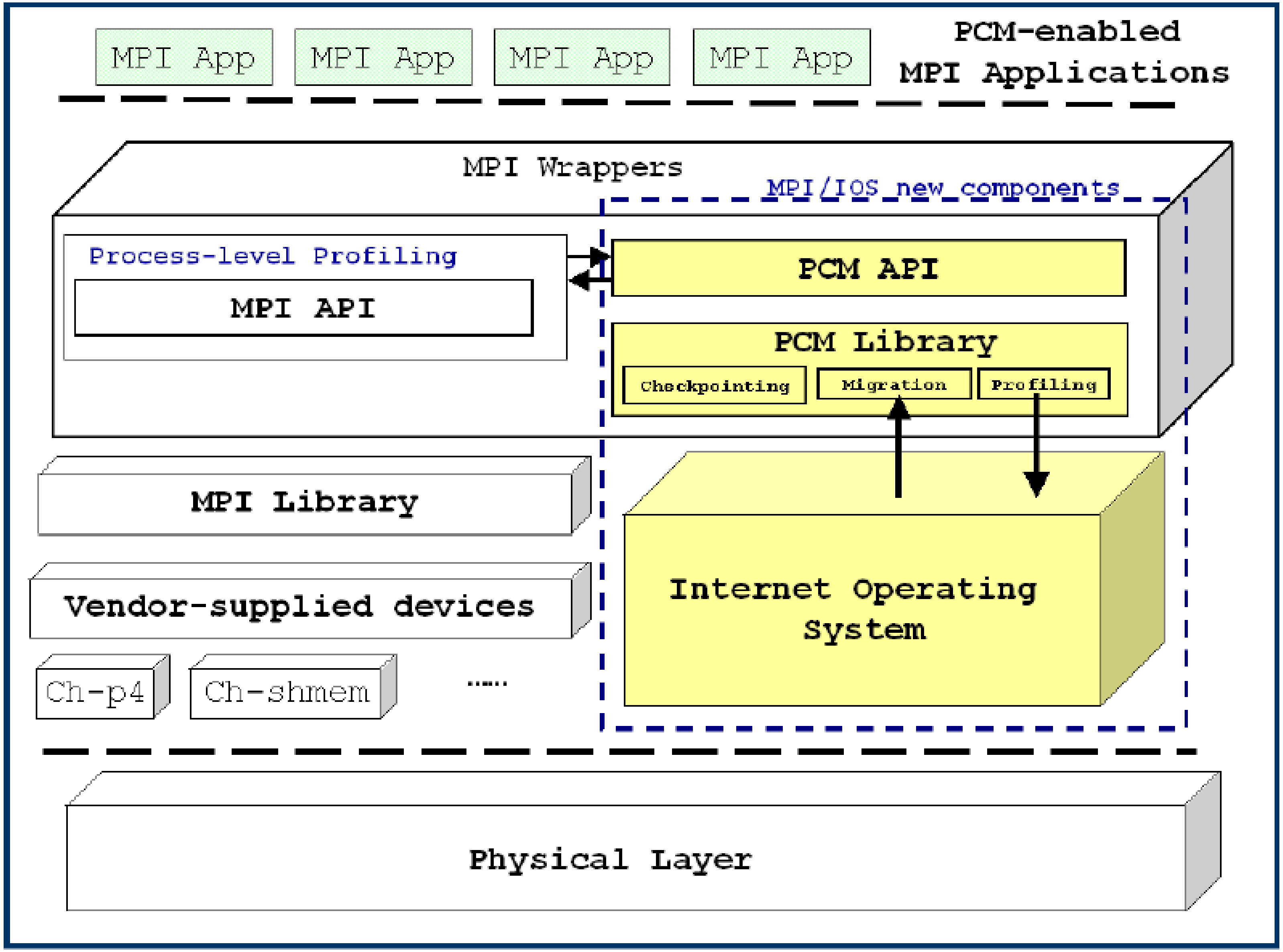
- Enabling MPI on Grid environments
  - A cost-effective solution for scientific computing
  - A scalable solution for high performance distributed applications
- Goals
  - Extending MPI to support the dynamic reconfiguration and high adaptability to dynamic computational grids
  - Improving performance and fault-tolerance of the code
  - Minimalizing changes to legacy code

MPI/IOS Framework

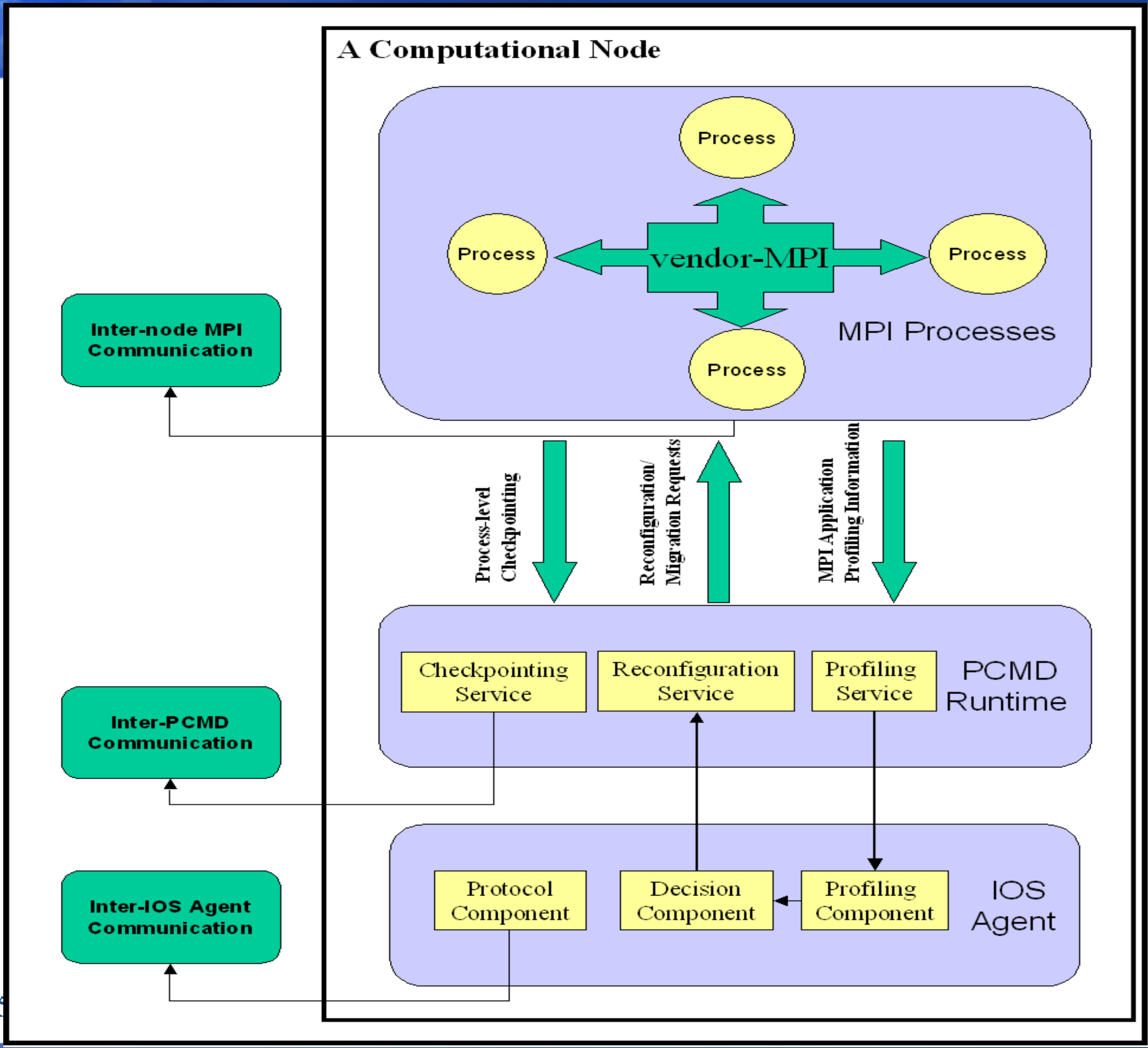
- Extending MPI with:
  - Semi-transparent checkpointing
  - Process migration support
  - Integration with IOS for middleware-triggered reconfiguration

Architecture

- The MPI/IOS runtime architecture consists of the following components:
  - MPI applications
  - Wrapped MPI that includes a Process Checkpointing and Migration (PCM) API
  - The PCM library, and wrappers for all MPI native calls
  - The MPI library, and
  - The IOS runtime components.


MPI App
MPI App
MPI App
MPI App
PCM-enabled
MPI Applications
MPI Wrappers
MPI/IOS_new_components
Process-level Profiling
MPI API
PCM API
PCM Library
Checkpointing
Migration
Profiling
MPI Library
Vendor-supplied devices
Ch-p4
Ch-shmem
......
Internet Operating
System
Physical Layer

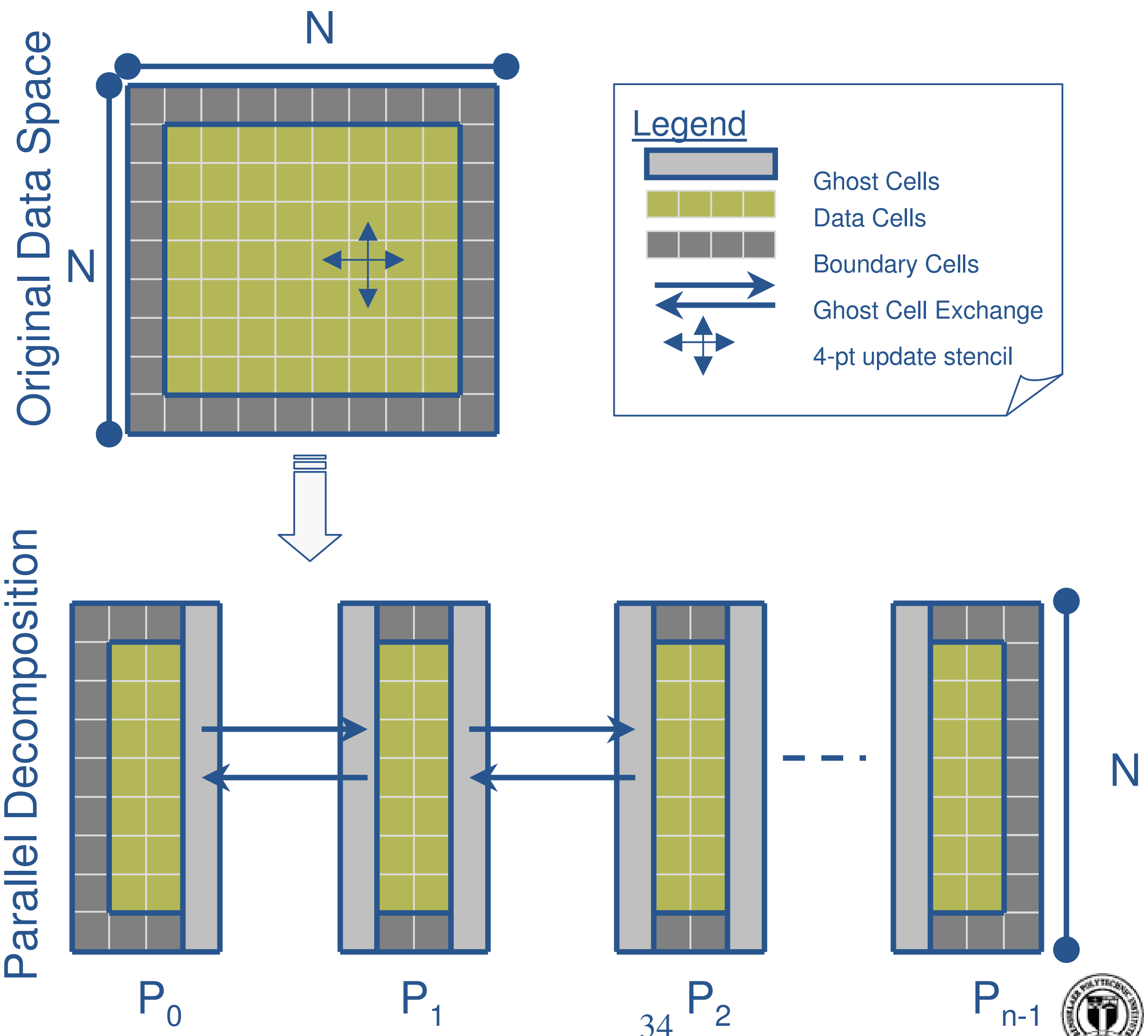
MPI/IOS interactions



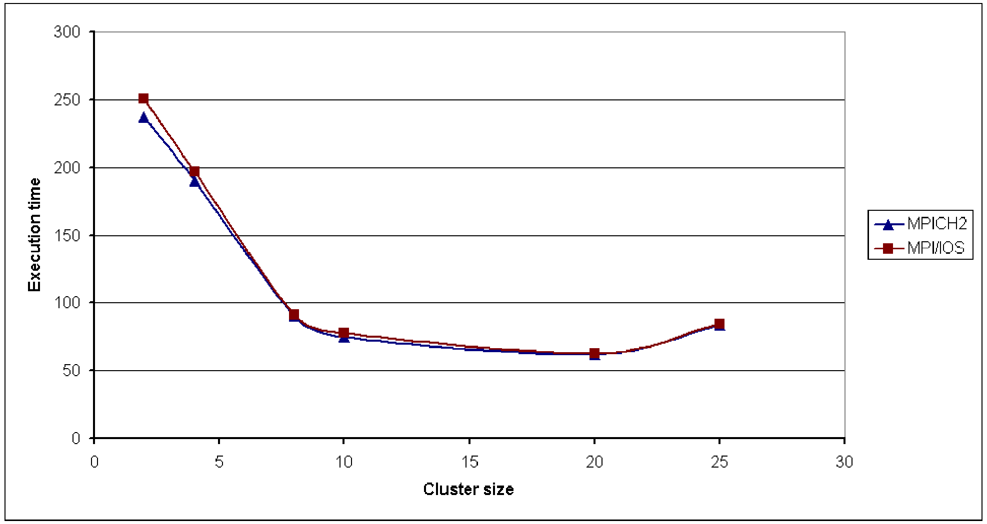
Case Study: Heat Diffusion Problem

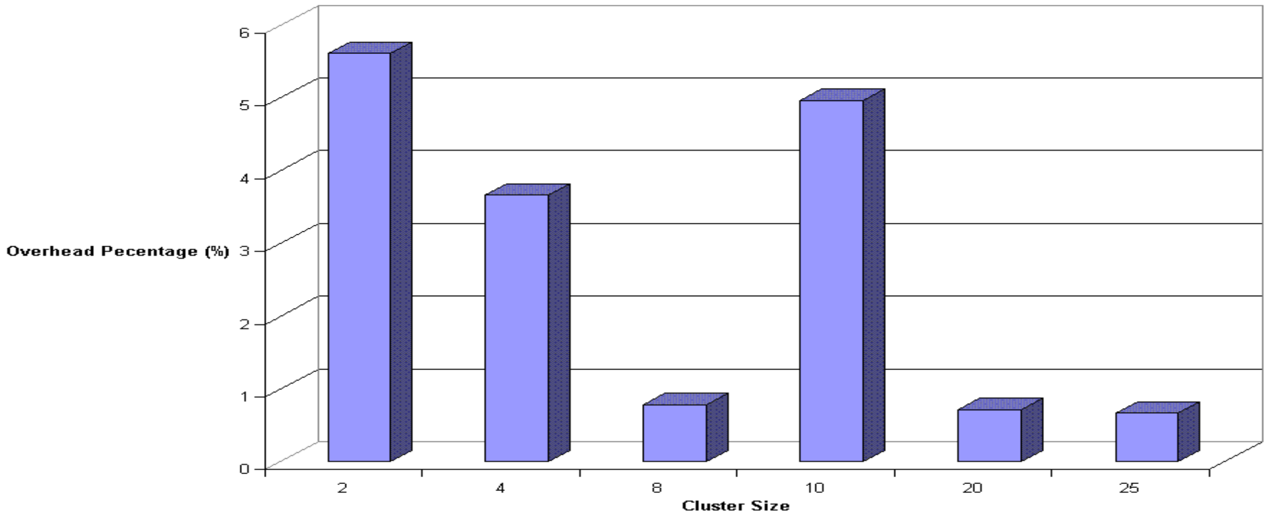
- A problem that models heat transfer in a solid
- A two-dimensional mesh is used to represent the problem data space
- An Iterative Application
- Highly synchronized

Parallel Decomposition of the Heat Problem



Empirical Results: Overhead of the PCM library

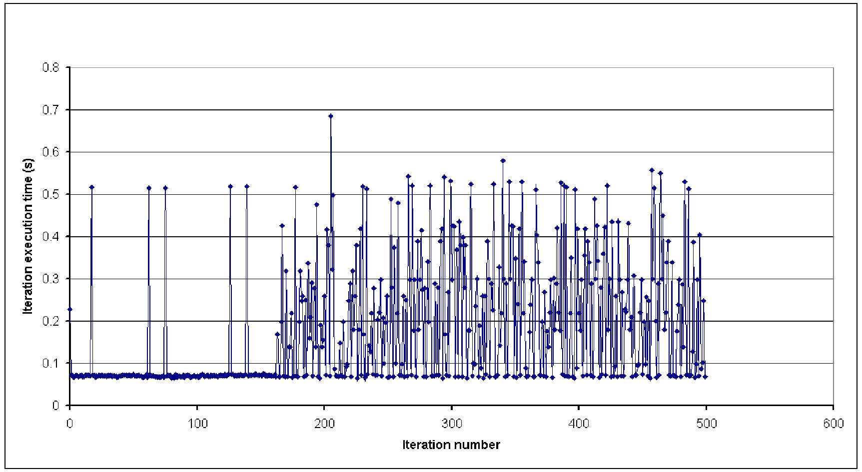




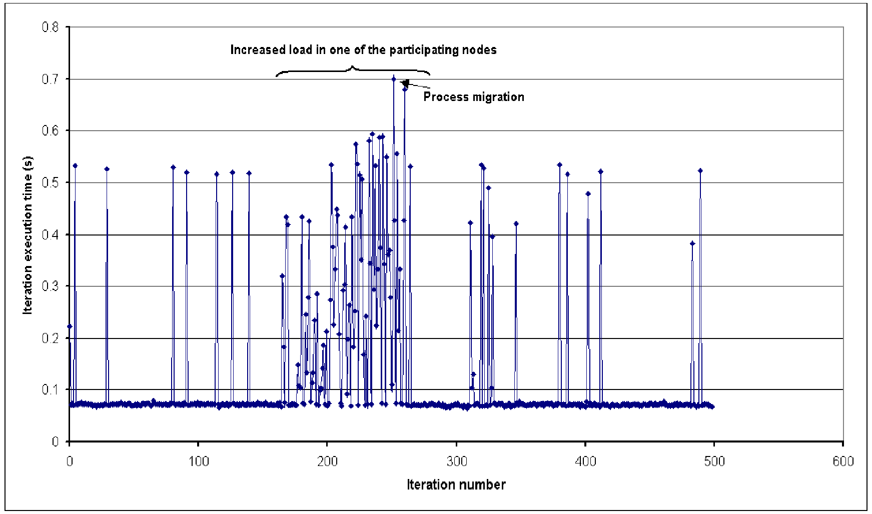
Rensselaer

Process migration evaluation

- **Breakdown of the execution time of 2D heat application iterations on a 4 node cluster using MPICH2**



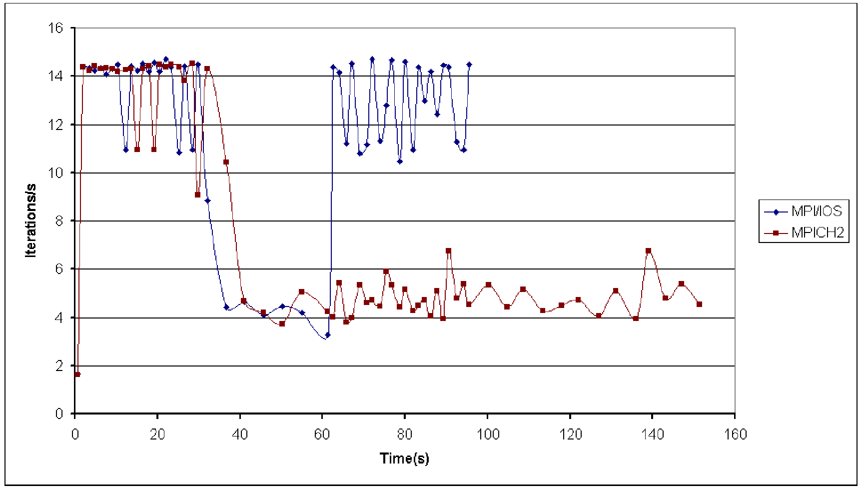
- **Breakdown of the execution time of 2D heat application iterations on a 4 node cluster using MPI/IOS prototype**



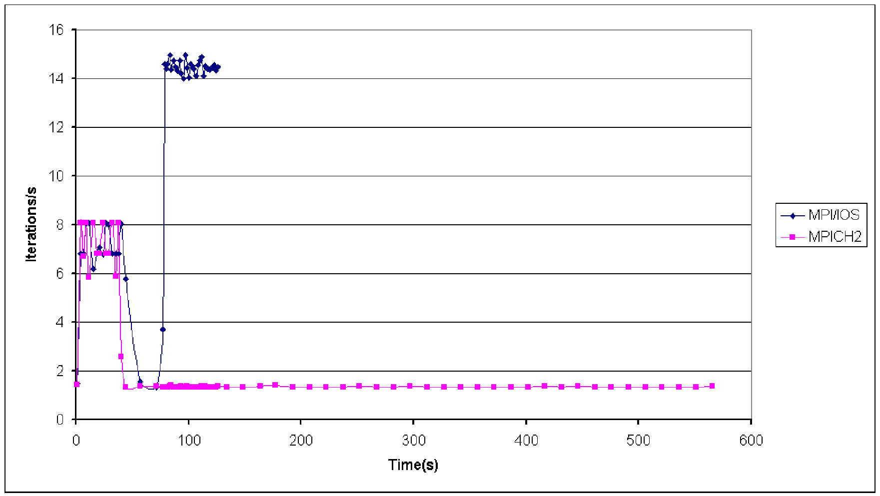
Rensselaer

Process migration evaluation

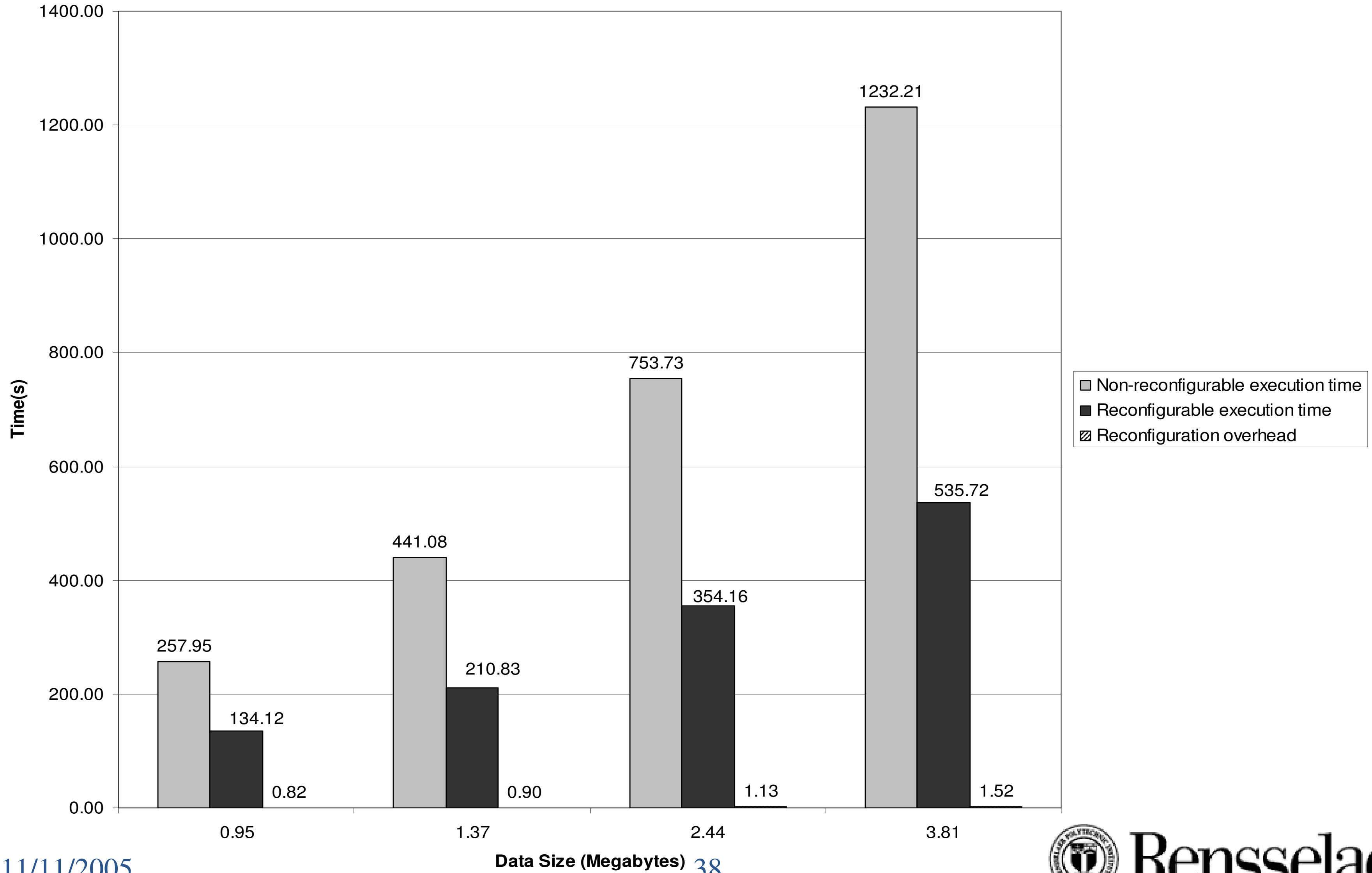
- Measured throughput of the two-dimensional heat application using MPICH2 and MPI/IOS. The applications adapted to the load change by migrating the affected process to one of the participating nodes in the case of MPI/IOS.



- Measured throughput of the 2D heat application using MPICH2 and MPI/IOS. The applications adapted to the load change by migrating the affected process to a fast machine that joined the computation in the case of MPI/IOS.

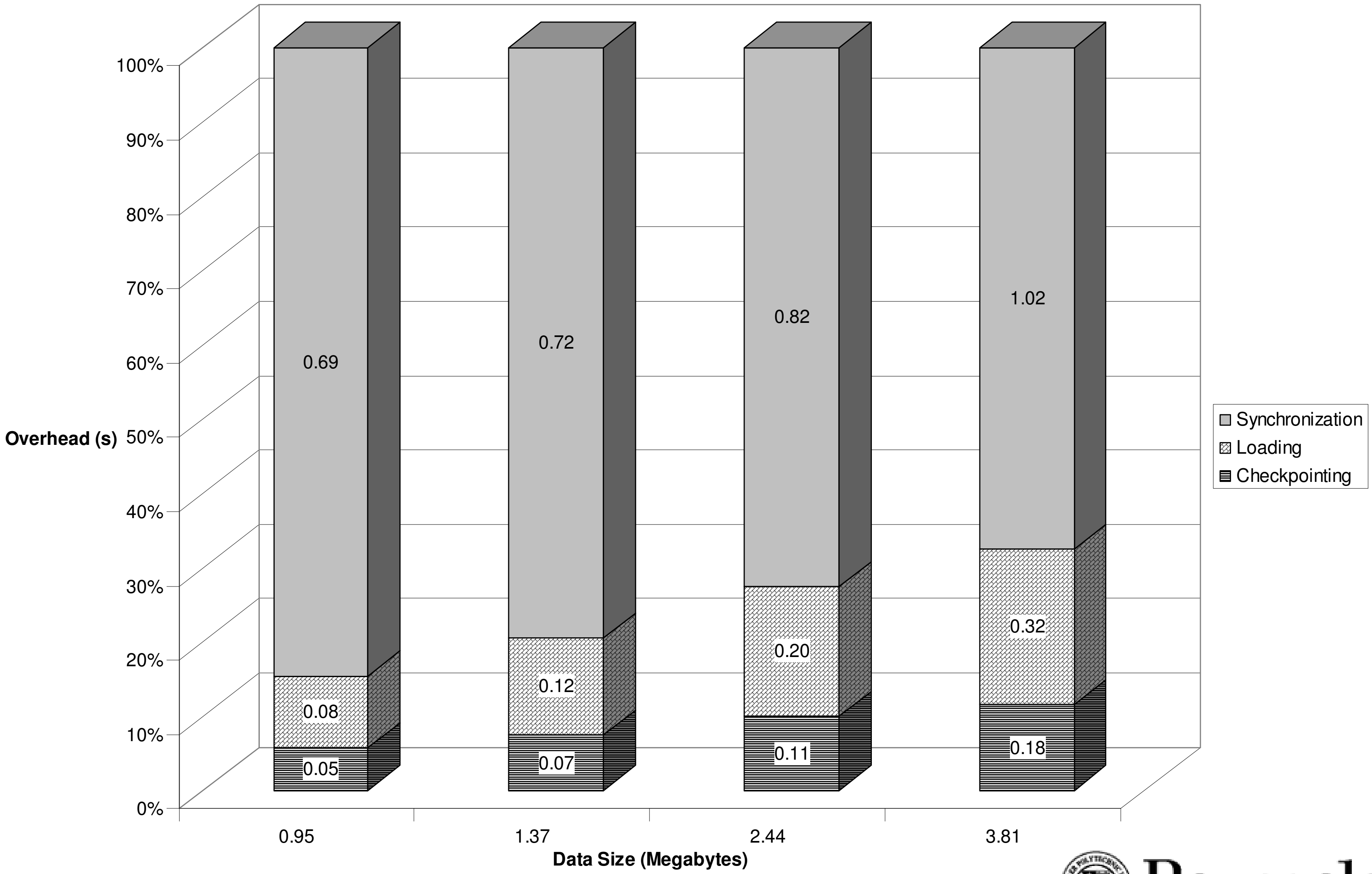


Reconfiguration Overhead



Rensselaer

Breakdown of Reconfiguration Cost



Related and Ongoing Work

Rensselaer

Related Work– Work Stealing/Internet Computing/P2P

- Work stealing
  - Cilk's runtime system for multithreaded parallel programming
  - Cilk's scheduler's techniques of work stealing
  - R. D. Blumofe and C. E. Leiserson, "Scheduling Multithreaded Computations by Work Stealing", FOCS 94
- Internet Computing
  - SETI@home (Berkeley)
  - Folding@home (Stanford)
- P2P systems
  - Distributed Storage: Freenet, KaZaA
  - File Sharing: Napster, Gnutella

Related Work-- Globus/NWS

- **Globus:**
  - **A toolkit to address issues related to the development of grid-enabled tools, services and applications**
  - **www.globus.org**

- **NWS**
  - **A distributed system that periodically monitors and dynamically forecasts the performance of various network and computational resources**
  - **http://nws.cs.ucsb.edu/**

Ongoing Work

- Effective decentralized grid load balancing algorithms.

- Virtual surgical planning simulation on the Rensselaer Grid
  - Scan of real patient is processed to extract solid model and inlet flow waveform.
  - Model is discretized and flow equations solved.
  - Multiple alterations to model are made within intuitive human computer interface and evaluated similarly.
  - This simulation will be run on Rensselaer Grid using MPI/IOS

- Other potential target applications:
  - ScalaPACK: a popular software package for linear algebra.
  - PCA: principal component analysis using massive data sets