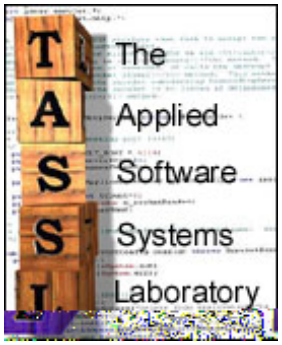
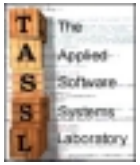


Adaptive Cluster Computing using JavaSpaces

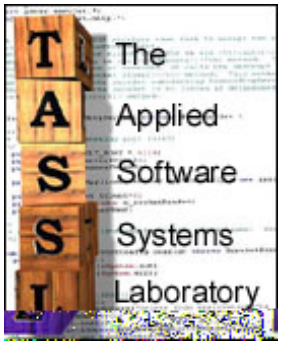
Jyoti Batheja and Manish Parashar

The Applied Software Systems Lab.
ECE Department,
Rutgers University

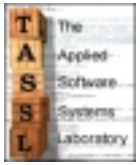


Outline

- Background
  - Introduction
  - Related Work
  - Summary of presented Framework
- Framework
  - Framework Architecture
  - Framework Operation
- Experimental Evaluation
- Conclusions & Future Work



Background



Introduction

Motivation

- Traditional approach adopted for HPC
- Potential for leveraging existing networked resources for HPC

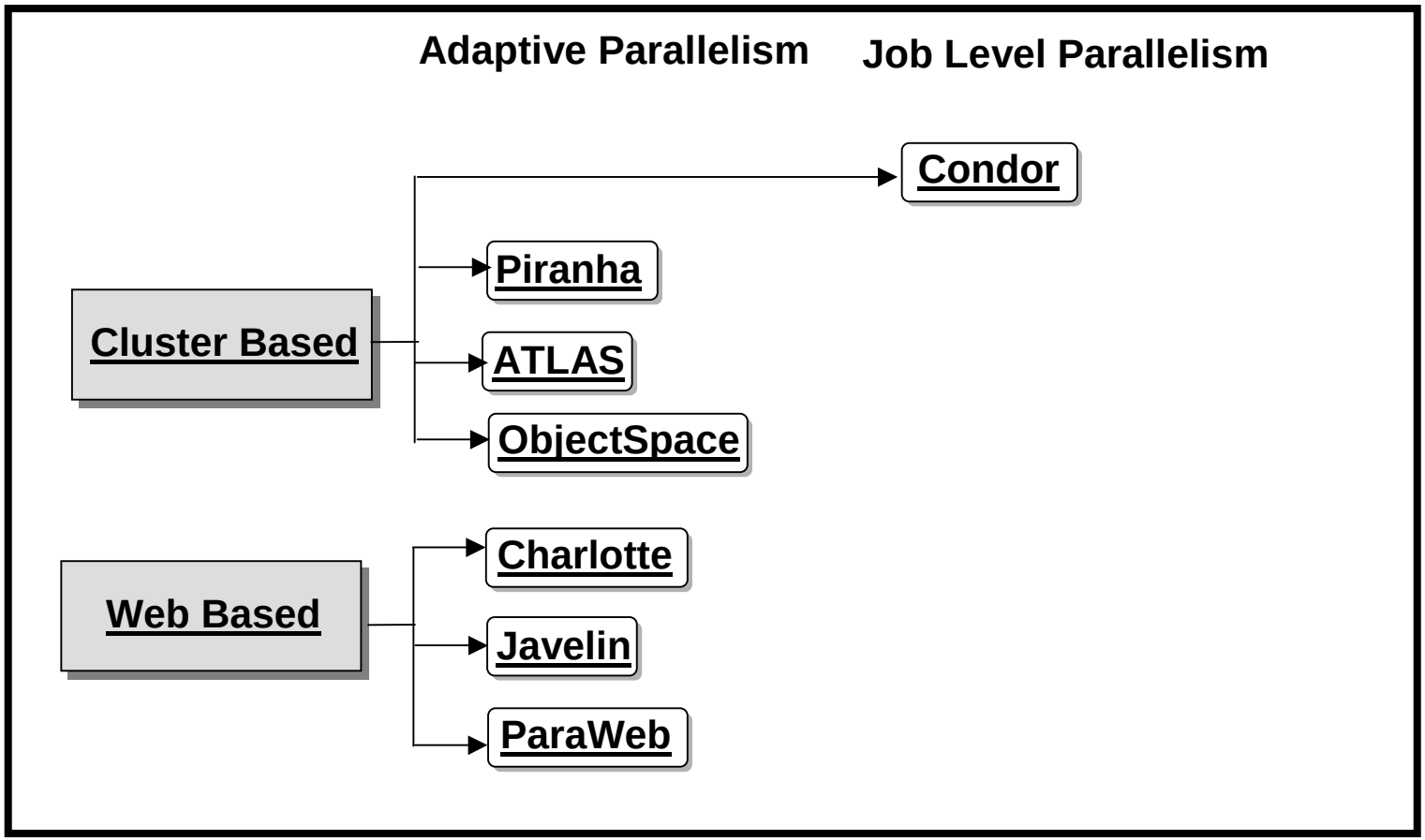
Objective is to design and implement a framework that:

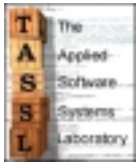
- Aggregates networked computing resources
- Non-intrusively exploits idle resources for Parallel/Distributed Computing

Challenges that need to addressed by the Framework

1. Adaptability to Cluster Dynamics
2. System Configuration and Management Overhead
3. Heterogeneity
4. Security and Privacy concerns
5. Non intrusiveness

Related Work





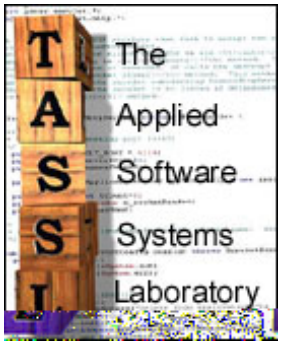
Summary of the Presented Work

Features

- Implementation using Java as the core programming language to leverage portability across heterogeneous platforms
- Offloading user responsibilities to the Framework for Resource Availability event triggers
- Minimized configuration overhead by employing remote node configuration techniques

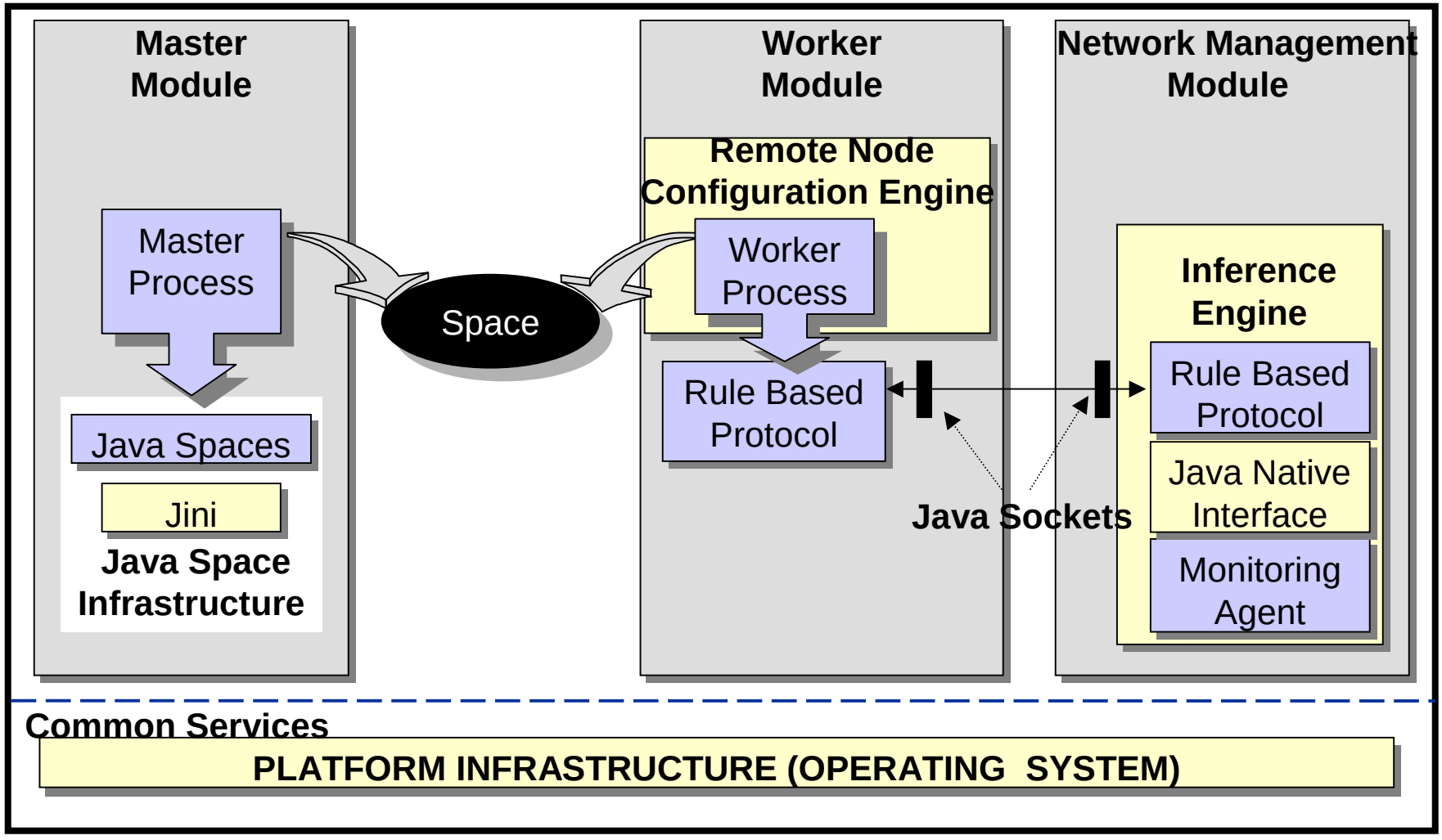
Targeted Applications

- High Computational Complexity
- Massively Partitioned Problems
- Small Input and Output Sizes

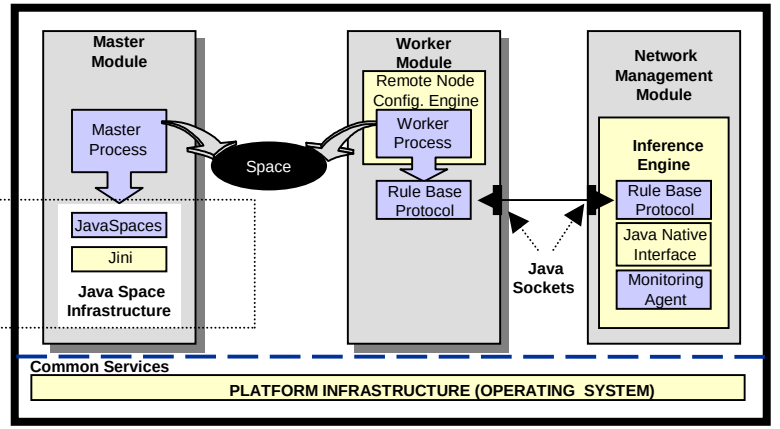


Framework Architecture

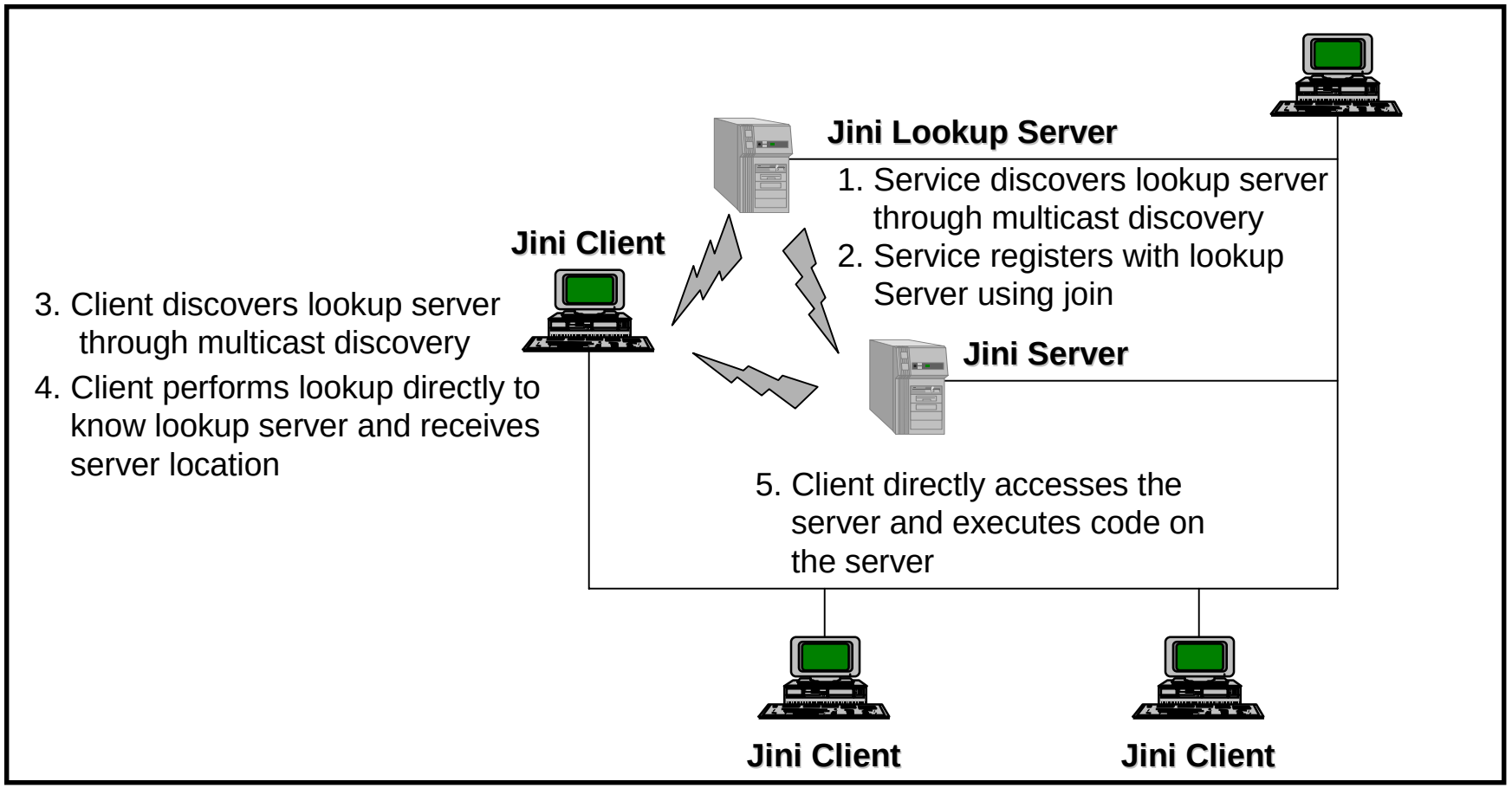
Architectural Overview



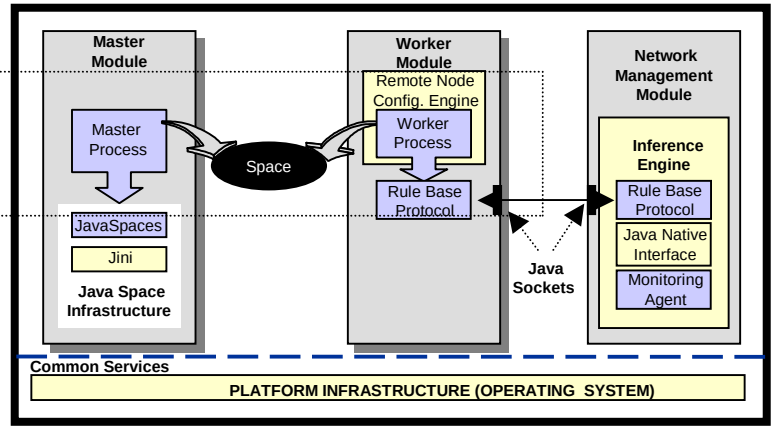
Architectural Components: Jini Infrastructure



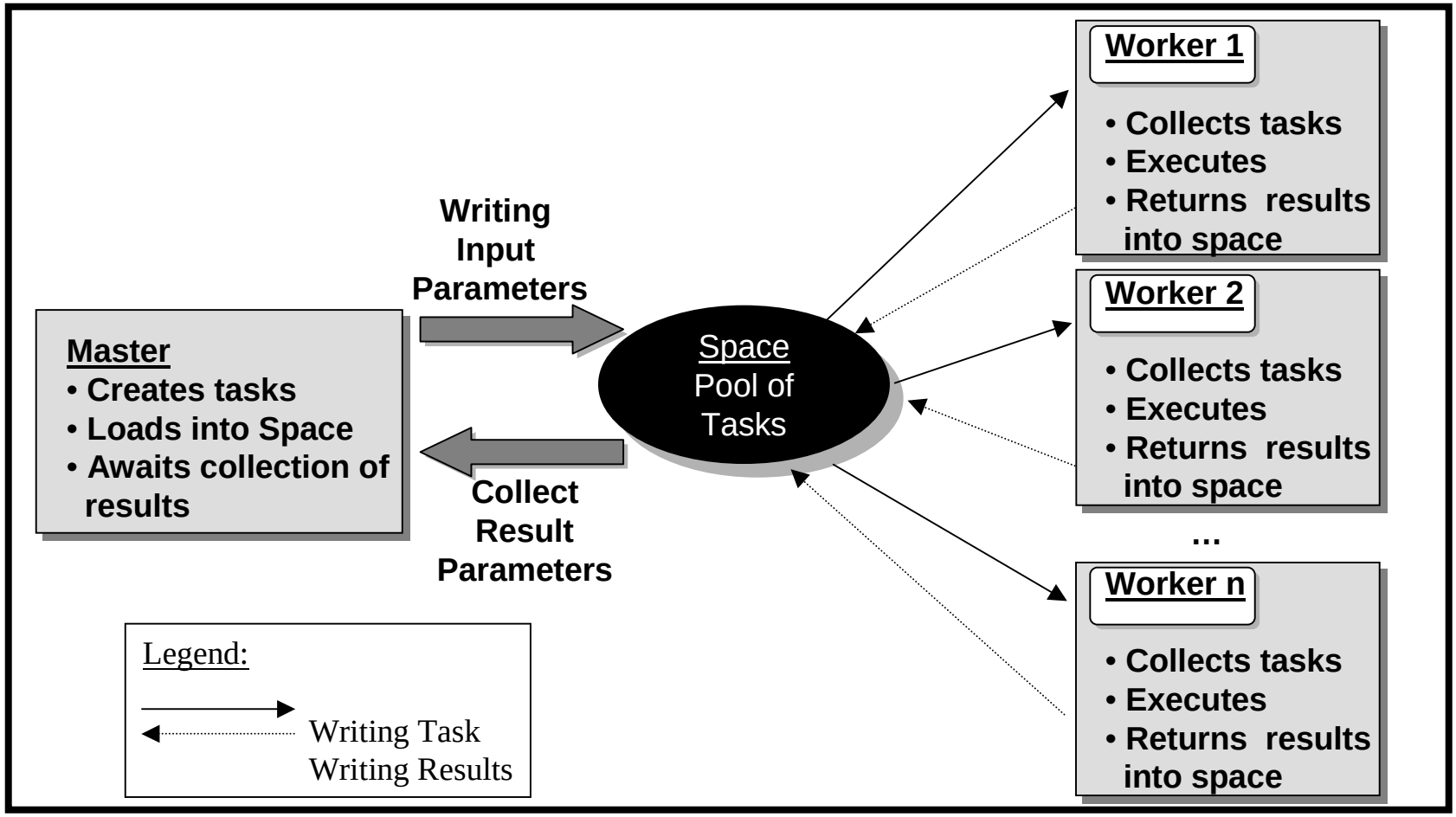
Jini Based Infrastructure



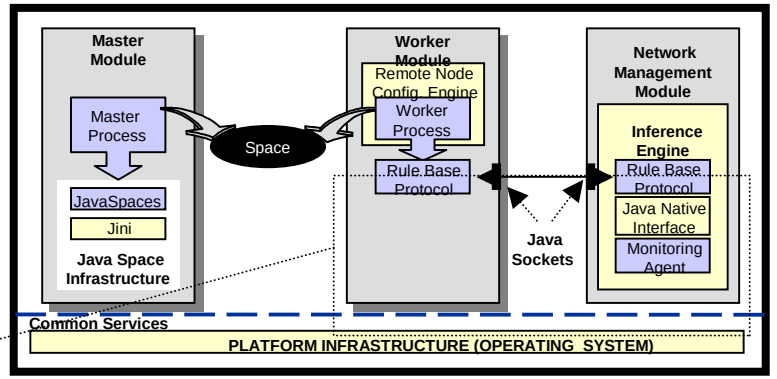
Architectural Components: Master-Worker Paradigm



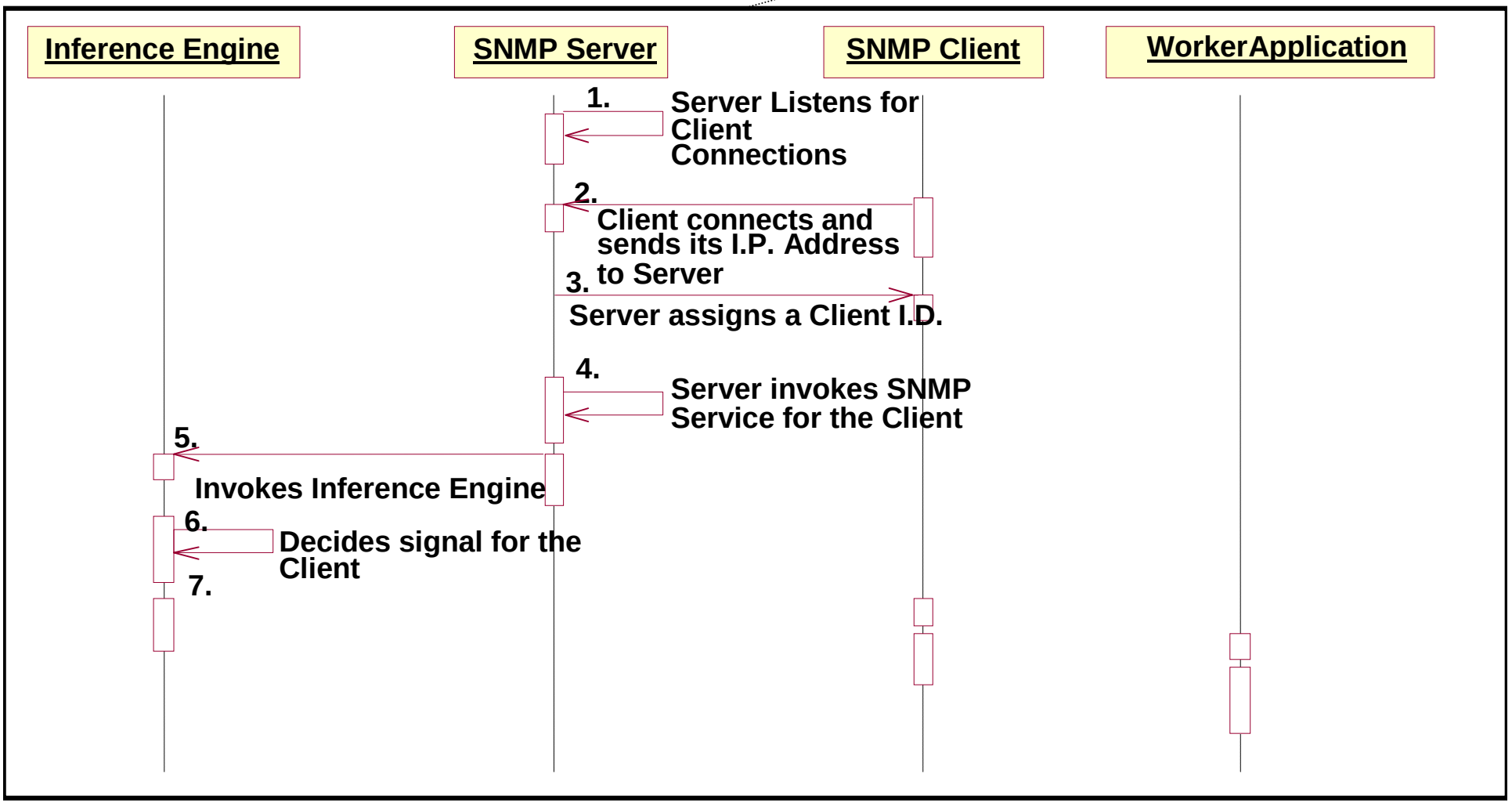
Master-Worker Interaction Diagram



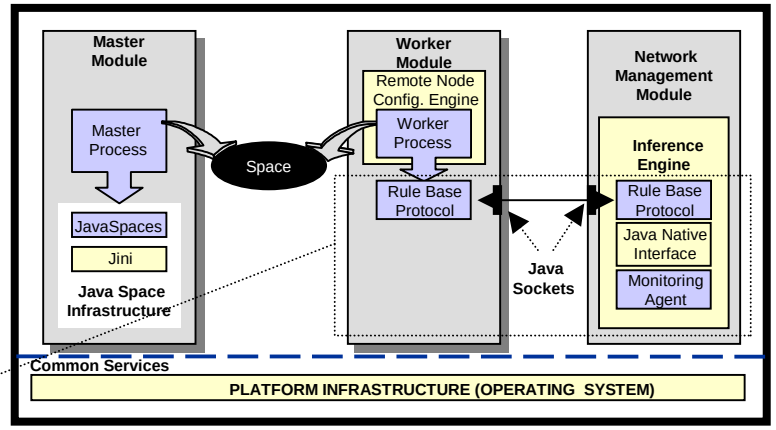
Architectural Components: Rule-Base Protocol



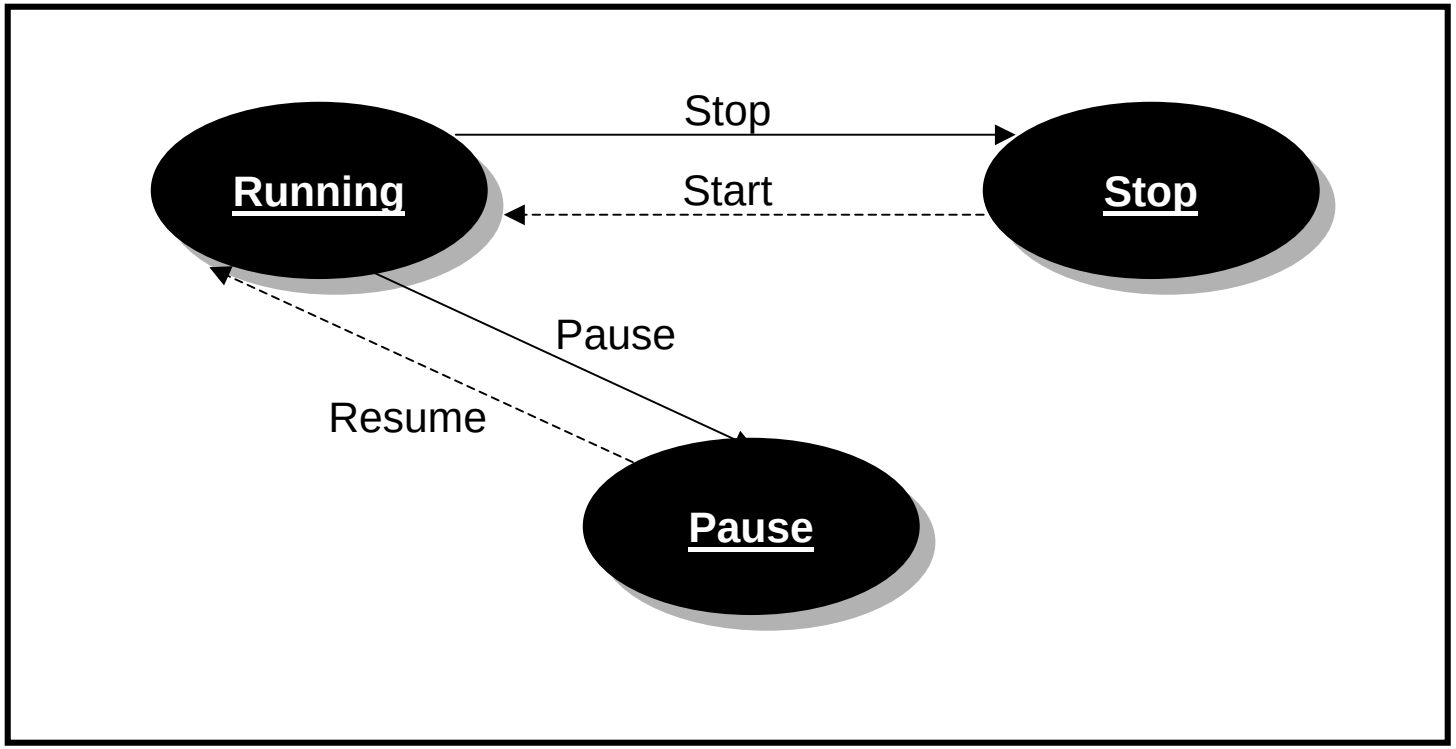
UML Interaction Diagram for Rule-Base Protocol

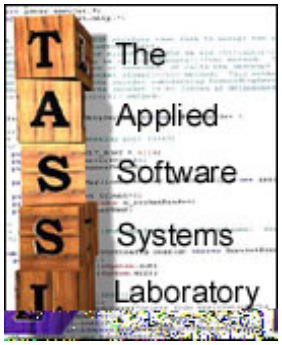


Architectural Components: Worker Execution States



Worker State Transition Diagram





Experimental Evaluation

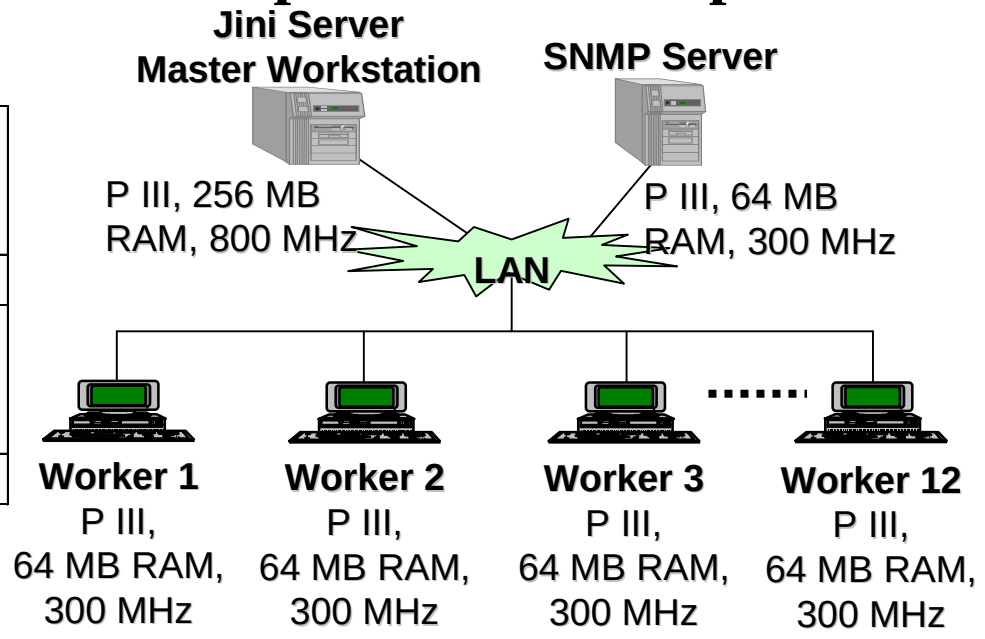
Experimental Evaluation

Application

- Parallel Monte Carlo Simulation for Stock Option Pricing
  - The problem domain is divided into 50 sub tasks, each comprising of 100 simulations
- Parallel Ray Tracing
  - A 600X600-image plane is divided into rectangular slices of 25X600 thus creating 24 independent tasks

Metrics	Option Pricing Scheme	Ray Tracing Scheme
Scalability	Medium	High
CPU Memory Requirements	Adaptable depending on number of simulations	High
Dependency	No	No

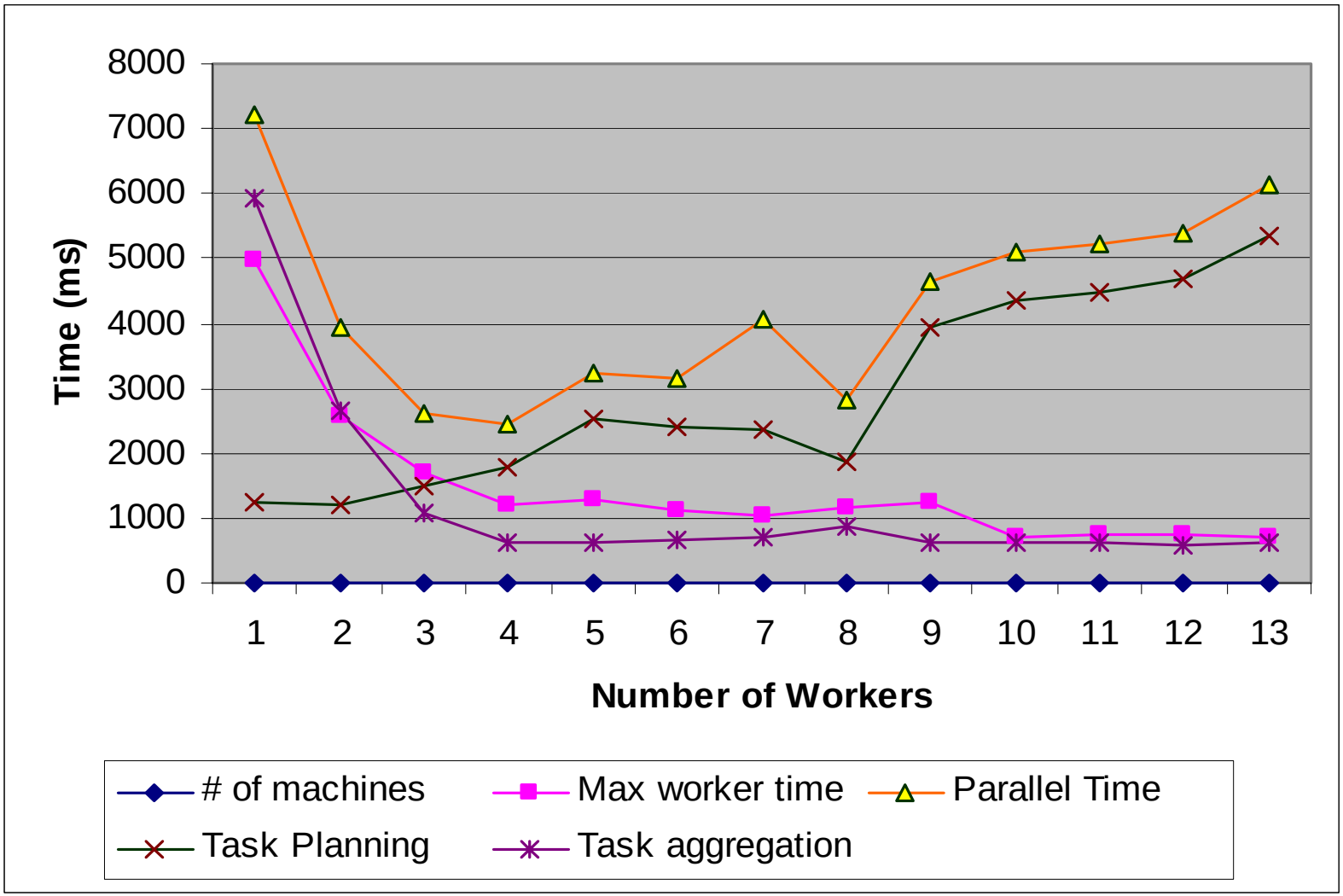
Experimental Setup



Experiment 1: Scalability Analysis

(Option Pricing Scheme)

Figure 1: Scalability Analysis for Option Pricing Scheme



Experiment 2: Adaptation Protocol Analysis
(Option Pricing Scheme)

– **Load Simulator 1: Stop Signal**

– **Load Simulator 2: Pause Signal**

Figure 1(a): Graph of CPU Usage on Worker Nodes for Option Pricing Scheme

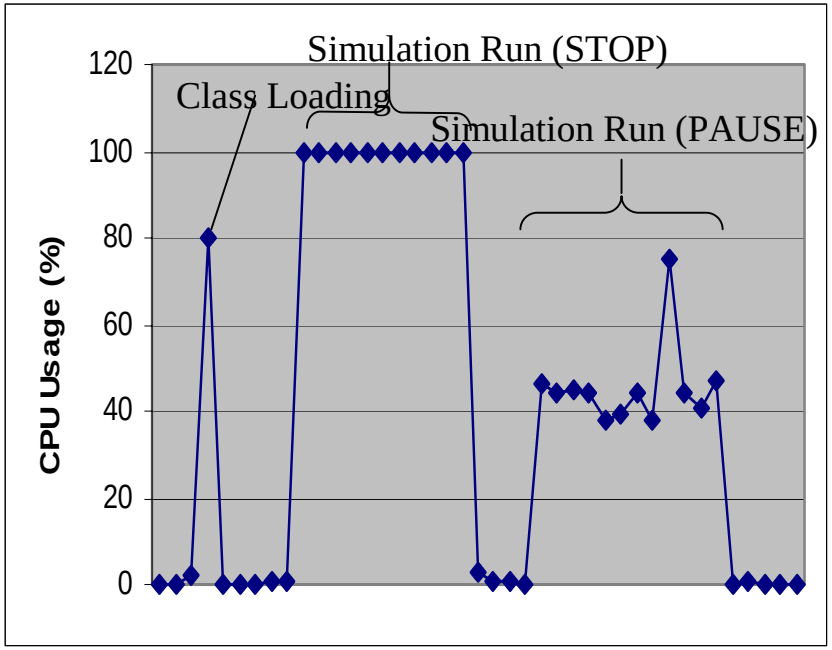
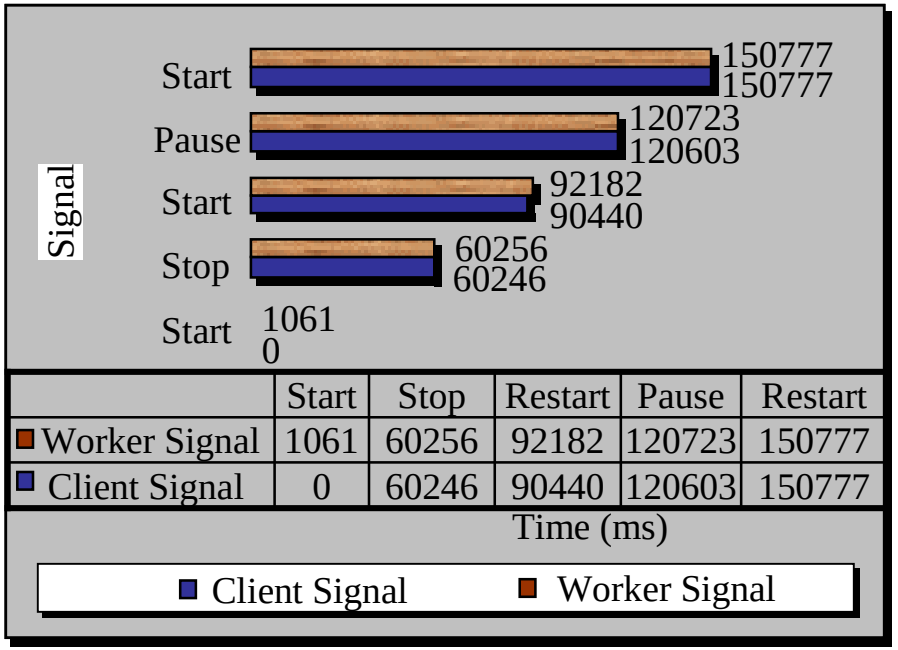


Figure 1(b): Graph of Worker Reaction Times for Option Pricing Scheme



	Start	Stop	Restart	Pause	Restart
Worker Signal	1061	60256	92182	120723	150777
Client Signal	0	60246	90440	120603	150777

Experiment 3: Dynamic Behavior Patterns under varying load conditions (Option Pricing Scheme)

Figure 1(a): Graph of dynamics under load showing time measurements for Option Pricing Scheme

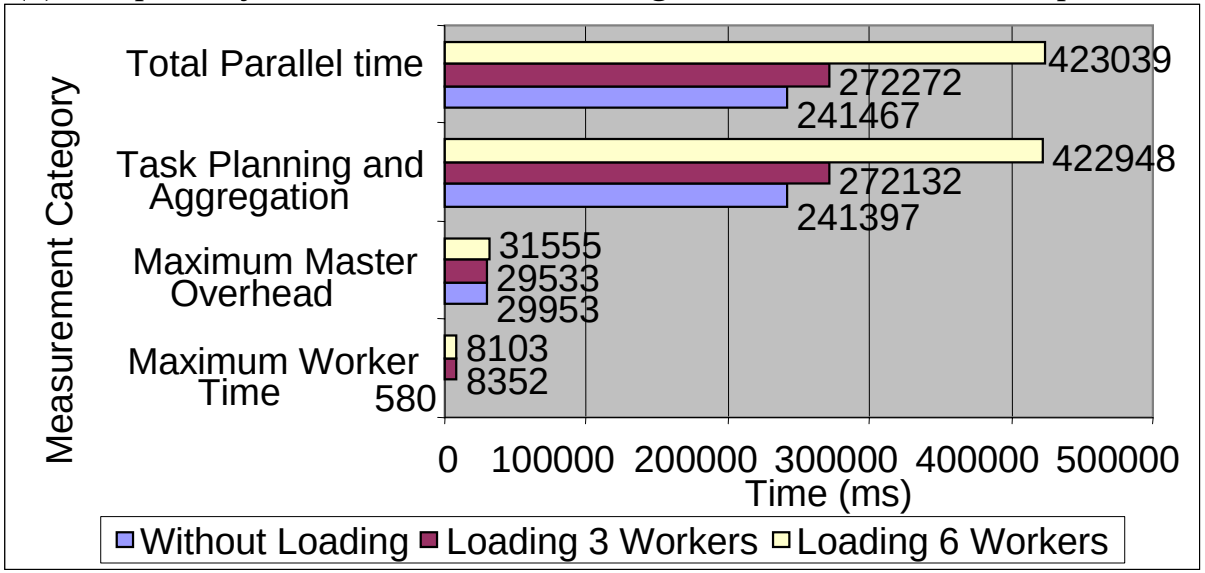
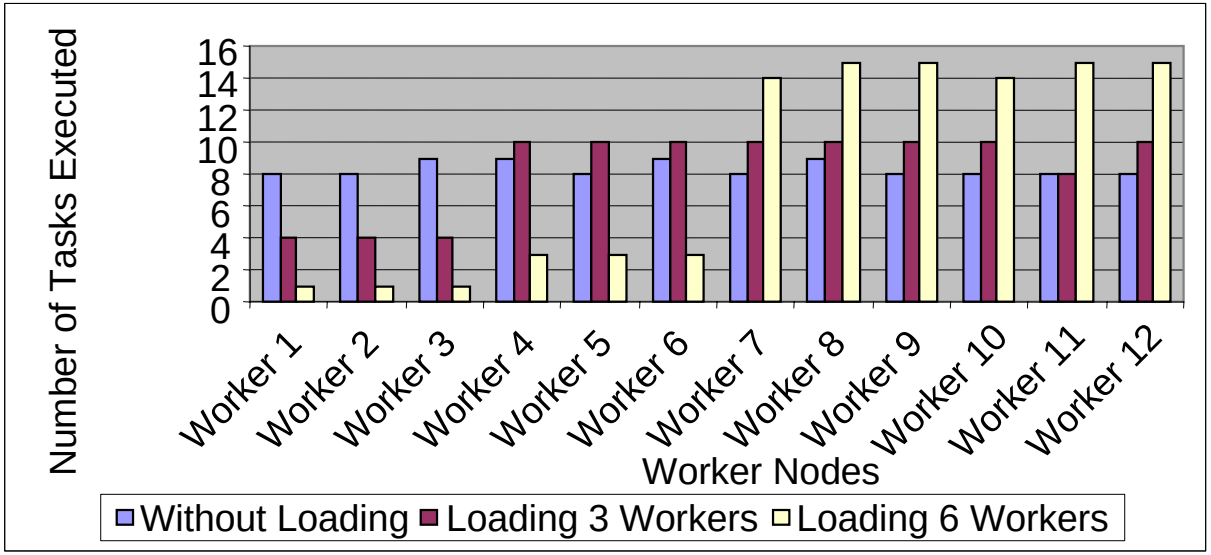
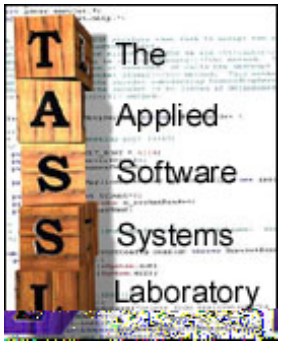
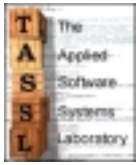


Figure 1(b): Graph of dynamics under load showing task distribution for Option Pricing Scheme





Conclusions and Future Work



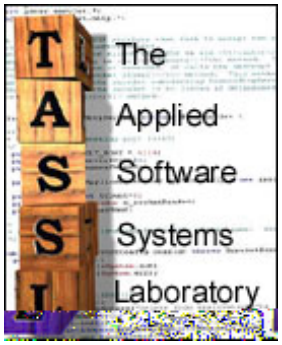
Conclusions

Summary

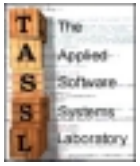
- Good Scalability for coarse grained applications
- Idle workstations can be effectively used to compute large parallel jobs
- Framework provides support for global code deployment and configuration management
- Monitoring and reacting to System State enables us to minimize intrusiveness to machines within the cluster

Future Directions

- Optimizations at the Application level
- Use of Transaction Management services to address Partial Failures
- Incorporating a distributed JavaSpace model to avoid a single point of resource contention or failure

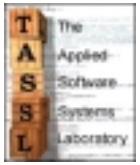


Additional Slides



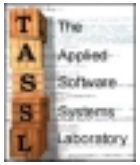
Summary of Related Cluster Based Parallel Computing Architectures

Architecture	Approach	Key Features	Strengths	Weaknesses
Condor	Cluster based, Job level parallelism, Asynchronous offloading of work to idle resources on a request basis	Queuing mechanism, Check pointing and process migration, Remote procedure call and matchmaking	Survives partial failure, preserved local execution environment	Incompleteness of checkpointing files. Difficulty in linking checkpointing code with user code for third party software.
Piranha	Cluster based, Adaptive parallelism, Centralized work distribution using the master worker paradigm	Implementation of Linda Model using tuple spaces, Recycles idle resources based on user defined criteria.	Survives partial failure, efficient cycle stealing	Requires modifications to the Linda systems. Supports only Linda Programs



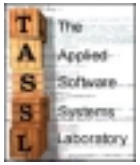
Summary of Related Cluster Based Parallel Computing Architectures (cont'd)

Architecture	Approach	Key Features	Strengths	Weaknesses
ATLAS	Cluster based, Adaptive Parallelism, Hierarchical work stealing	Implementation based on Cilk. Emphasis on thread scheduling algorithm based on work stealing techniques. Runtime library for work stealing, thread management, marshalling of objects.	Fault Tolerance, Localized computations: more bandwidth, reduced latencies, and natural mapping to administrative domains. Improved scalability	Global file sharing leads to conflicts in access control. Heterogeneity concerns: conversion mechanisms for native libraries, platform specific Java interpreter.
ObjectSpace /Anaconda	Cluster based, Job level parallelism at brokers	Eager scheduling, automated/manual job replication, Job level parallelism at broker level.	Periodic polling the health of participation hosts.	Fault tolerance is not supported among brokers.



Summary of Related Web Based Parallel Computing Architectures

Architecture	Approach	Key Features	Strengths	Weaknesses
Charlotte	Web based, Adaptive parallelism, Centralized work distribution using downloadable applets	Abstraction of Distributed Shared Memory, data type mapping and use of atomic update protocol. Directory look up service for idle resource identification, embedded lightweight class server on local hosts	Direct inter-applet communication, Fault tolerance, Eager scheduling and Two Phase idempotent execution	Centralized broker can be a potential bottleneck
Javelin	Web based, Adaptive parallelism, Centralized work distribution using downloadable applets	Heterogeneous, secure execution environment, portability to include Linda and SPMD programming models	Offline worker computation, Applet balancing scheme via broker registration	Inter applet communication routed via initiating broker: potential bottleneck



Summary of Related Web Based Parallel Computing Architectures (cont'd)

Architecture	Approach	Key Features	Strengths	Weaknesses
ParaWeb	Web based, Adaptive parallelism, Centralized work distribution using scheduling servers	Implementation of Java Parallel Runtime System to provide global shared memory and transparent thread management and Java Parallel Class Library to facilitate upload and download of execution code across the web	Clients can act as workers	JPRS requires modifying the Java Interpreter

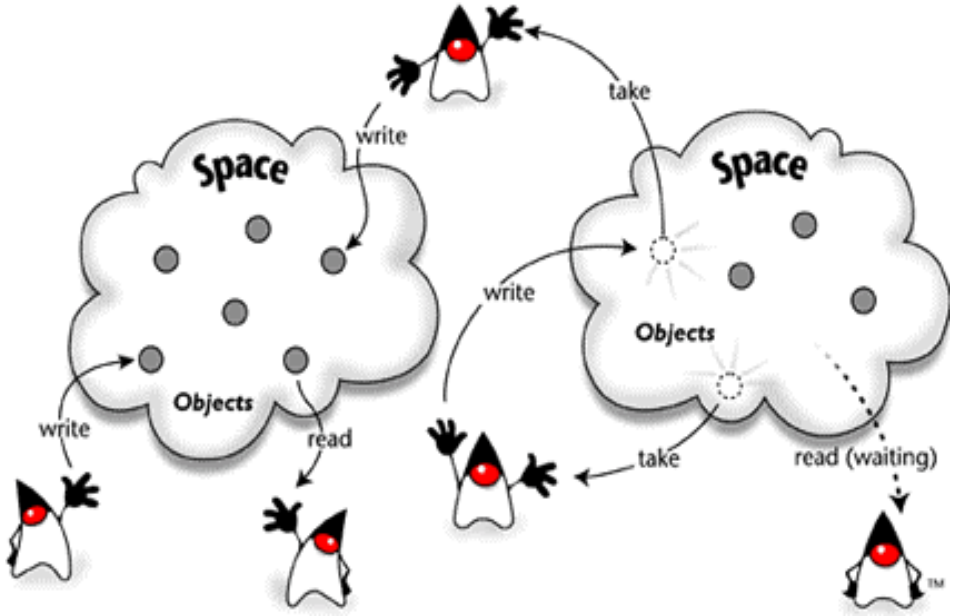
Open Issues

- Heterogeneity concerns
- Manual Management

Architectural Components: JavaSpace Service

Features

- Coordination tool for gluing processes together into a distributed application
- Supports heterogeneous computing
- Supports Dynamic Client Pool
- Provides a storage repository for Java Objects
- Provides a simplistic API for access of the Java objects in the space
  - write
  - read
  - take

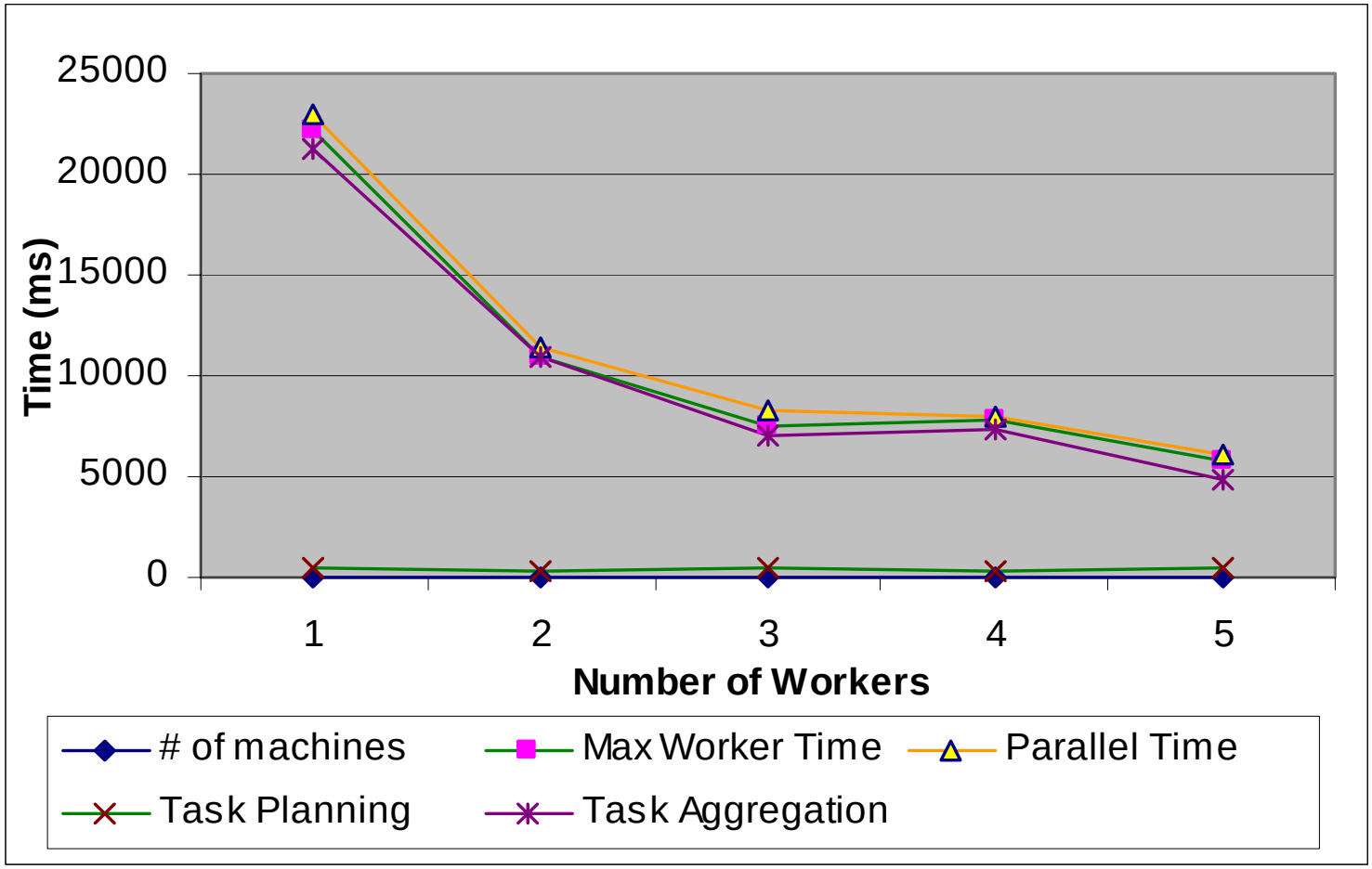


JavaSpaces TM Interaction: Source Java Web Site

Experiment 1: Scalability Analysis

(Ray Tracing Scheme)

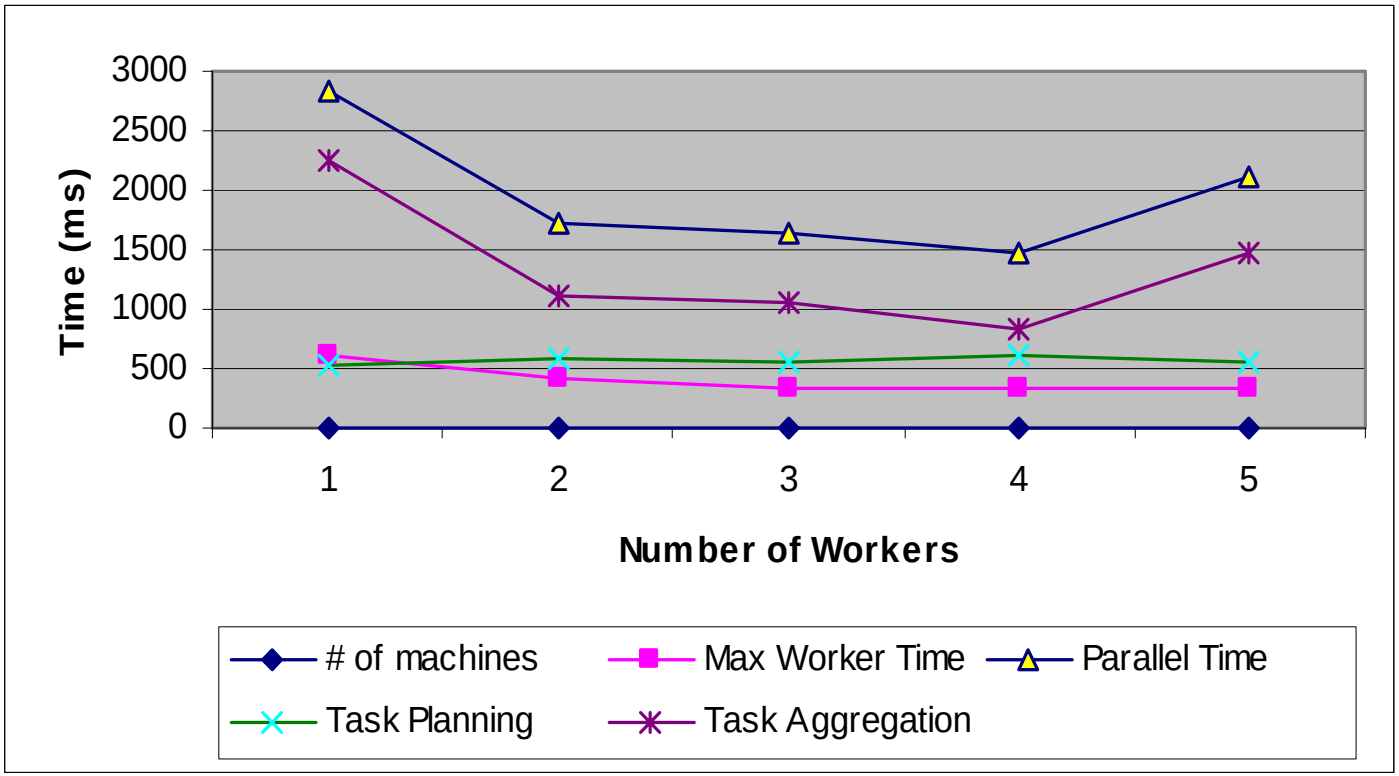
Figure 2: Scalability Analysis for Ray Tracing Scheme



Experiment 1: Scalability Analysis

(Pre-fetching Scheme)

Figure 3: Scalability Analysis for Pre-fetching Scheme based on Page Rank



Experiment 2: Adaptation Protocol Analysis
(Ray Tracing Scheme)

Figure 2(a): Graph of CPU Usage on Worker Nodes for Ray Tracing Scheme

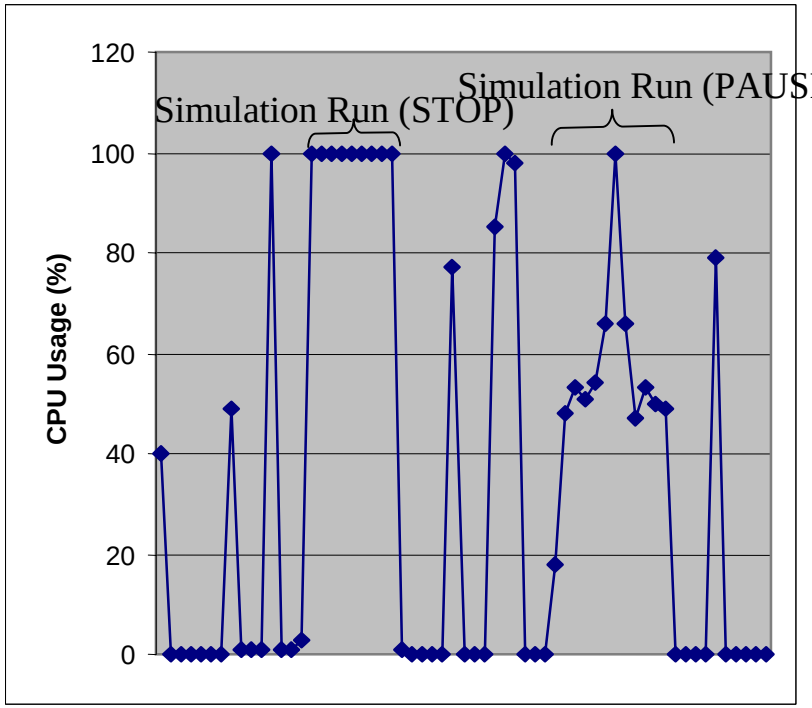
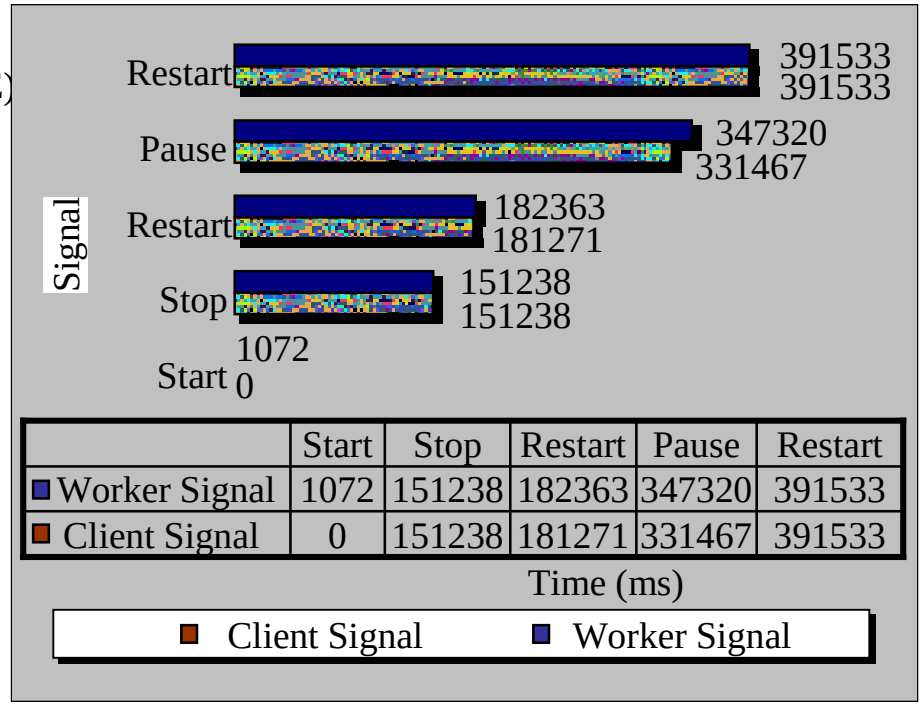


Figure 2(b): Graph of Worker Reaction Times for Ray Tracing Scheme



	Start	Stop	Restart	Pause	Restart
Worker Signal	1072	151238	182363	347320	391533
Client Signal	0	151238	181271	331467	391533

Experiment 2: Adaptation Protocol Analysis
(Pre-fetching scheme)

Figure 3(a): Graph of CPU Usage on Worker Nodes for Pre-fetching Scheme

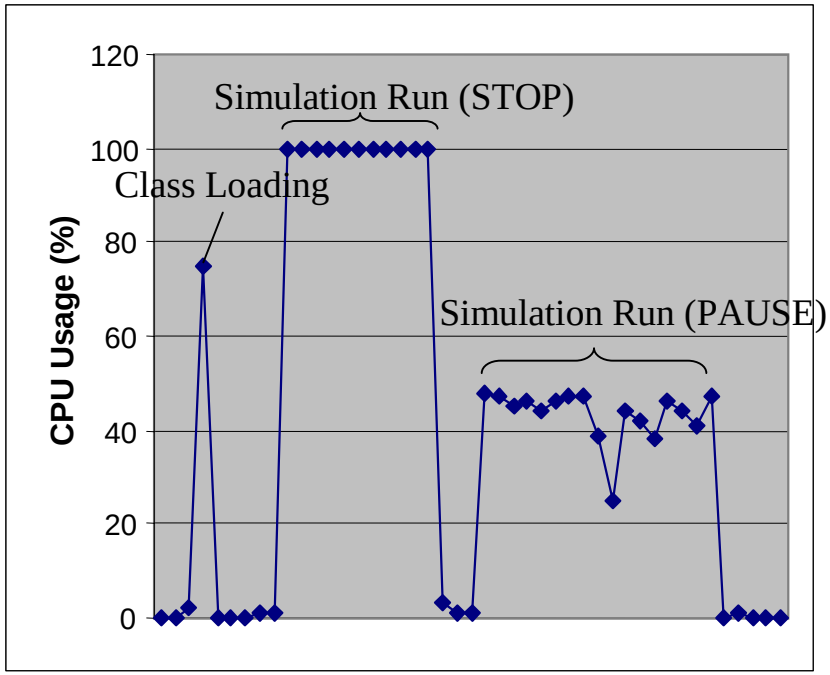
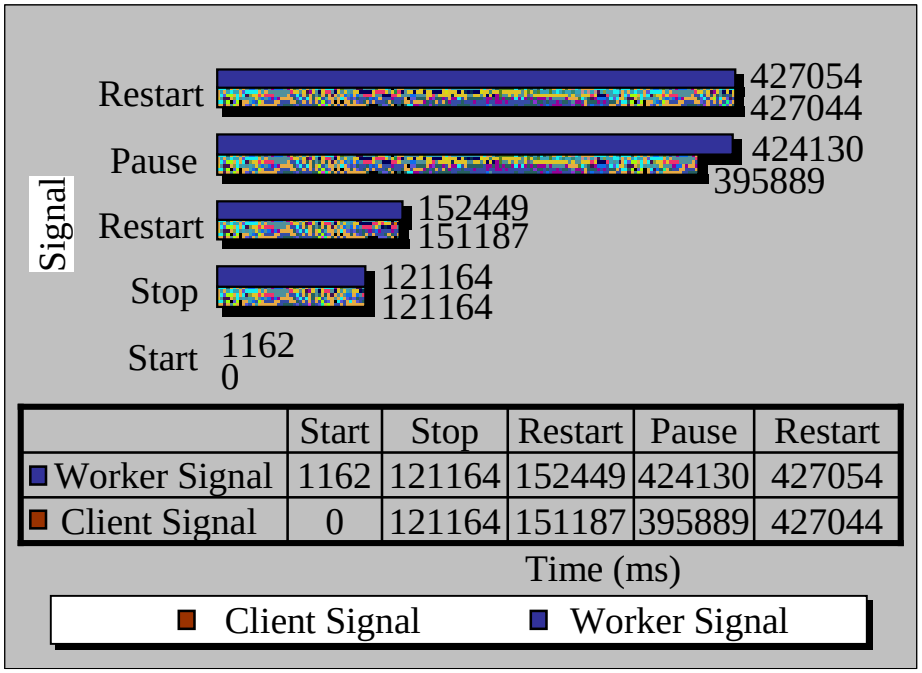


Figure 3(b): Graph of Worker Reaction Times for Pre-fetching Scheme



	Start	Stop	Restart	Pause	Restart
Worker Signal	1162	121164	152449	424130	427054
Client Signal	0	121164	151187	395889	427044

Experiment 3: Dynamic Behavior Patterns under varying load conditions (Ray Tracing Scheme)

Figure 2(a): Graph of dynamics under load showing time measurements for Ray Tracing Scheme

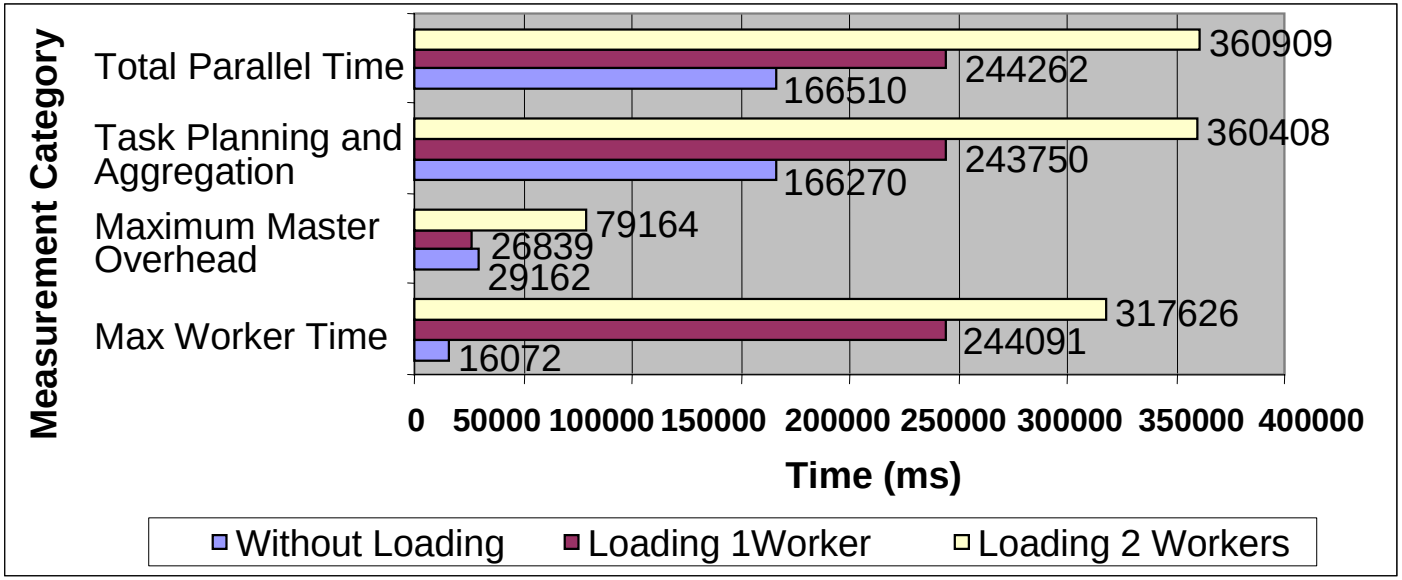
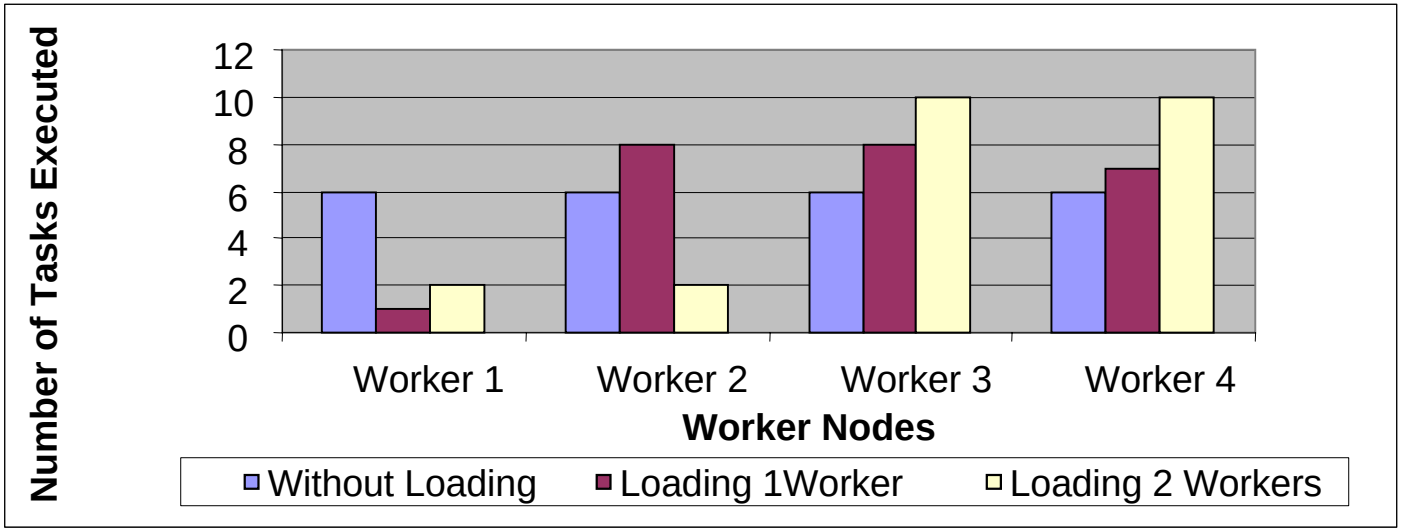


Figure 2(b): Graph of dynamics under load showing task distribution for Ray Tracing Scheme



Experiment 3: Dynamic Behavior Patterns under varying load conditions (Pre-fetching Scheme)

Figure 3(a): Graph of dynamics under load showing time measurements for Pre-fetching Scheme

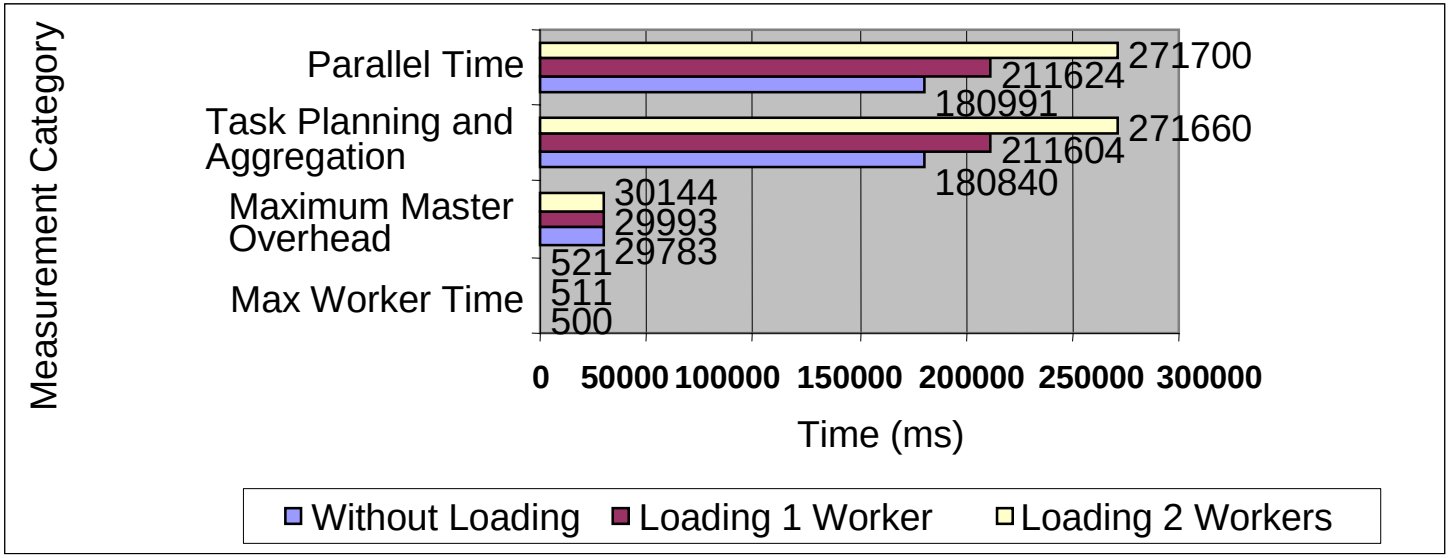


Figure 3(b): Graph of dynamics under load showing task distribution for Pre-fetching Scheme