A Resource Management System for Adaptive Parallel Applications in Cluster Environments

Sheikh K. Ghafoor, Tomasz A. Haupt, Ioana Banicescu, Ricolindo L. Carino , and Nisreen Ammari

Center for Advanced Vehicular Systems, Mississippi State University

Outline

- Introduction
- Related Work
- RMS for Adaptive Applications
- Experiments and Results
- Conclusions and Future Work

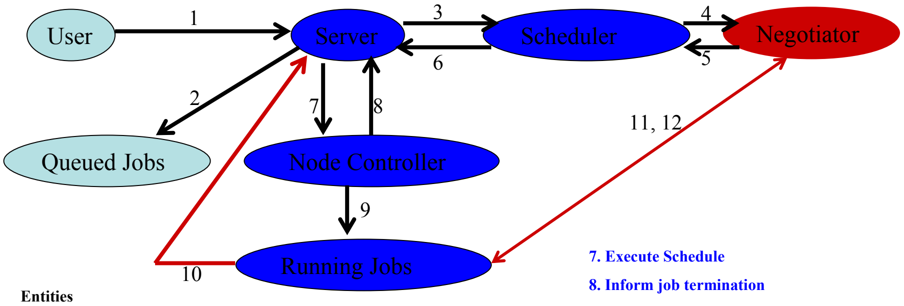
Adaptive Applications

- Classification of Parallel Applications (Feitelson and Rudolp)
  - Rigid application: Fixed no. of processors defined by users
  - Moldable application: Fixed processors defined by Resource Management Systems (RMS)
  - Evolving application: Processors vary during execution, initiated by application
  - Malleable application: Processors change, initiated by RMS
- Adaptive application
  - Evolving and malleable applications
  - Change resources during execution

Motivation

- Adaptive applications promises
  - Improved system utilization
  - Better application performance
  - New classes of parallel applications driven by unpredictable data and events
- Current job scheduler and resource management system are unable to handle adaptive applications efficiently.
  - Lack of infrastructure supports is a major obstacle for development of adaptive applications
  - Absence of large no. adaptive applications is a reason for lack of motivation for developing infrastructure

Adaptive Parallel System



Entities

1. user
2. Server
3. Scheduler
4. Negotiator
5. Node Controller
6. Queued Jobs
7. Running Jobs

Events

1. Submit Job
2. Queue Job
3. Initiate Scheduling Cycle
4. Send negotiation Information
5. Send Negotiation Status
6. Send Schedule
7. Execute Schedule
8. Inform job termination
9. Resource Reallocate
10. Send Negotiation Request
11. Request to adapt

12 Resource Negotiation

Related Work

- Stop and Restart Software (SRS)-Vadhiyar and Dongarra, 2003
- Dynamic Resource Management on Distributed Systems (DRMS)-Moreira and Naik, 1997
- Adaptive resource Allocation-Jha et. Al, 1996
- Malleable-Job System for Timeshared Parallel Machines-Kale, Kumar, and DeSouza, 2002
- Adaptive Multiblock Parti (AMP)-Edjlali, Agrawal, Sussman and Saltz, 1995

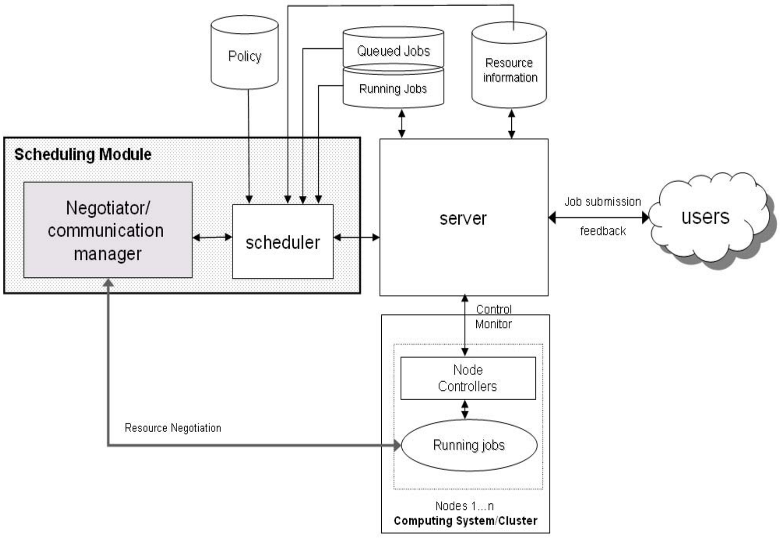
Research Issues

- Managing adaptive application is a complex and multi-faceted problem
  - Infrastructure support for adaptive applications
    - RMS support
    - Middleware support
  - Programming model adaptive applications

RMS Requirement

- Negotiation Mechanism
- Negotiation Protocol
- Management of Additional Scheduling Events
- Scheduling Algorithm
- Modified Node Controller

RMS Architecture



Resource Negotiation Protocol

- The adaptive applications and RMS require to communicate and negotiate
- Different communication scenarios are possible
- Interactions between applications and RMS has similarities
  - Business deal on internet by automated agents
  - Web Service Agreement in Grid environment
- Resource Negotiation Protocol adopted
  - Two party business negotiation model
  - Subset of “language for agreement” of WS-Agreement specification

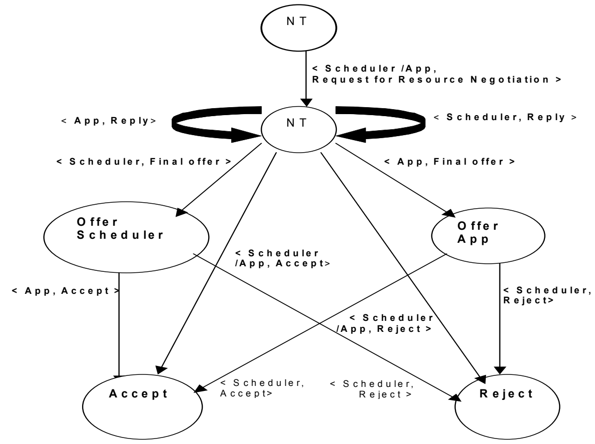
Protocol Requirements

- Supports all possible negotiation scenarios
- Supports Multi round negotiations
- Supports Negotiation of multiple resources
- Platform and language independent
- Simple
- Low operating overhead
- Easy to modify

Negotiation model

- The proposed resource negotiation model consists of two parties
- The RMS is one party and any adaptive application is the other party
- Any party may initiate the negotiation
- In the case of evolving jobs, applications initiate the negotiation
- In case of malleable jobs, the RMS initiates the negotiation
- Negotiation is done by exchanging Negotiation Template(NT)

Finite State Representation of Resource Negotiation Protocol



XML Instance of Negotiation Template

```
<?xml version="1.0" encoding="UTF-8"?>
<NT NT_AgreementID="1001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="NT.xsd">
<NT_Initiator>RMS</NT_Initiator>
<NT_Source>Sparrow-01-01.erc.msstate.edu</NT_Source>
<NT_Dest>Sparrow-01-02.erc.msstate.edu</NT_Dest>
<NT_Status>NT_NEGOTATION</NT_Status>
<Resource>
            <Type>CPU</Type>
            <Quantity>4</Quantity>
            <Usage>NT_REQUIRED</Usage>
            <Negotiability>NT_NEGOTIABLE</Negotiability>
            <Status>NT_NEGOTIATION</Status>
</Resource>
<Resource>
            <Type>MEM</Type>
            <Quantity>1024</Quantity>
            <Usage>NT_OPTIONAL</Usage>
            <Negotiability>NT_FIXED</Negotiability>
            <Status>NT_NEGOTIATION</Status>
</Resource>
</NT>
```

The Steps of the Protocol

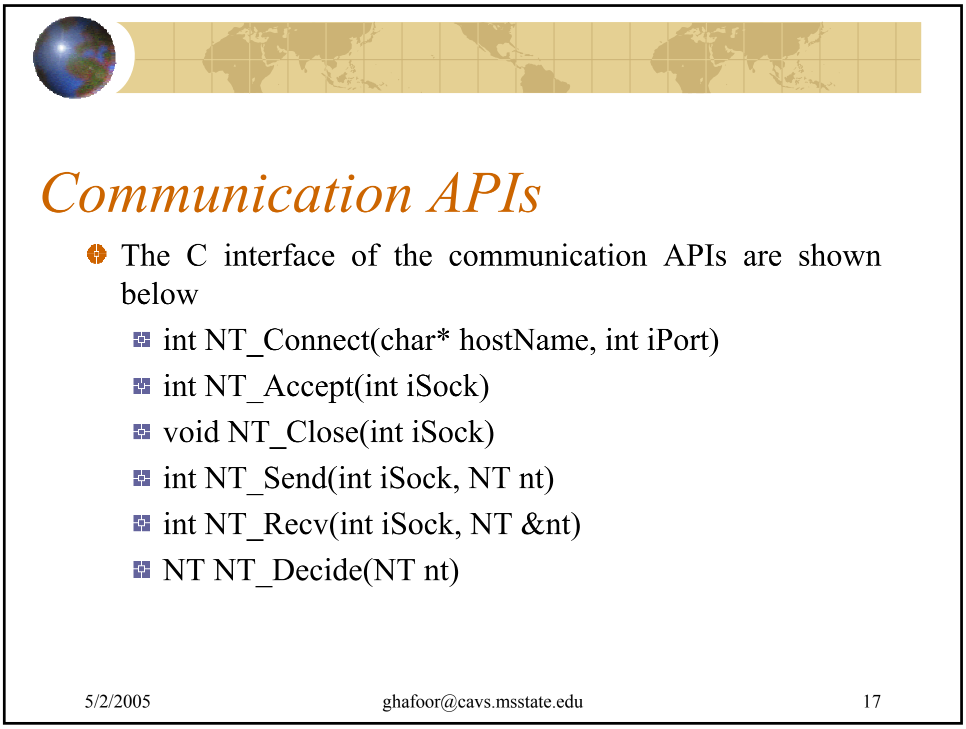
- The initiator establishes a connection with the other party.
- The initiator creates an NT and sends it to other party.
- The receiver examines the request and changes the current state of the NT and send it back to sender.
- Step 3 is repeated until one party stops negotiation or both commit to the current offer.
- Both parties accept an offer or it is rejected and the connection is closed.

Implementation

- Developed a rudimentary RMS
  - FCFS scheduler
  - Negotiation Manager
  - Server
  - Node Controller
- Defined and implemented two set of APIs
  - Set for communication and negotiation
  - Set for creation and manipulation of NT

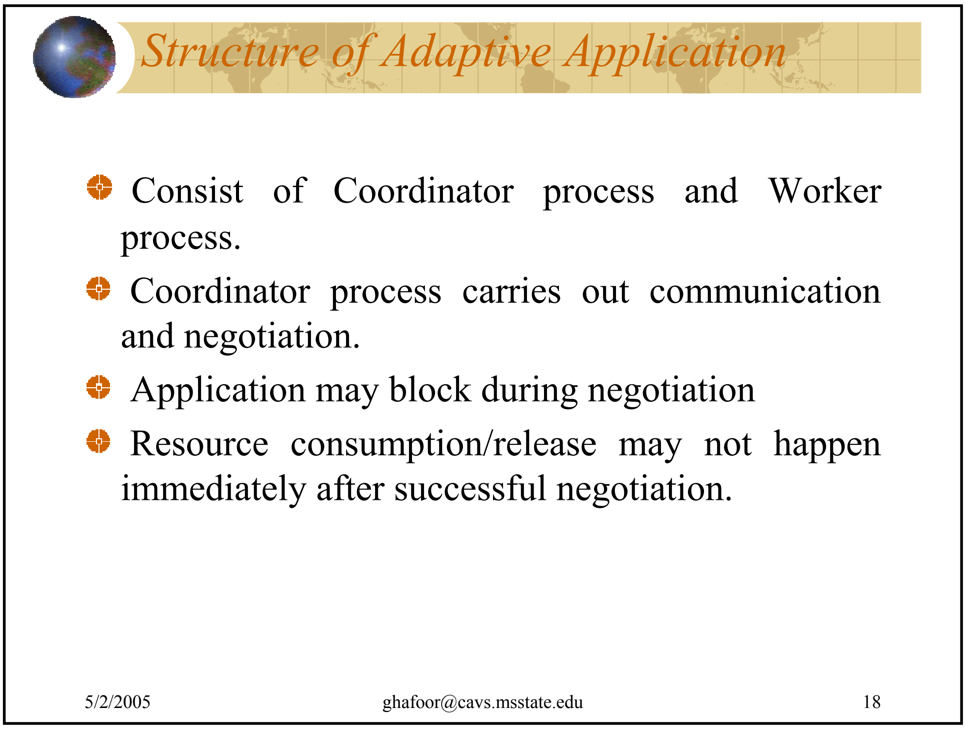
Developed few test applications

  - Rigid, malleable and evolving parallel N-body simulation



Communication APIs

- The C interface of the communication APIs are shown below
  - int NT_Connect(char* hostName, int iPort)
  - int NT_Accept(int iSock)
  - void NT_Close(int iSock)
  - int NT_Send(int iSock, NT nt)
  - int NT_Recv(int iSock, NT &nt)
  - NT NT_Decide(NT nt)



Structure of Adaptive Application

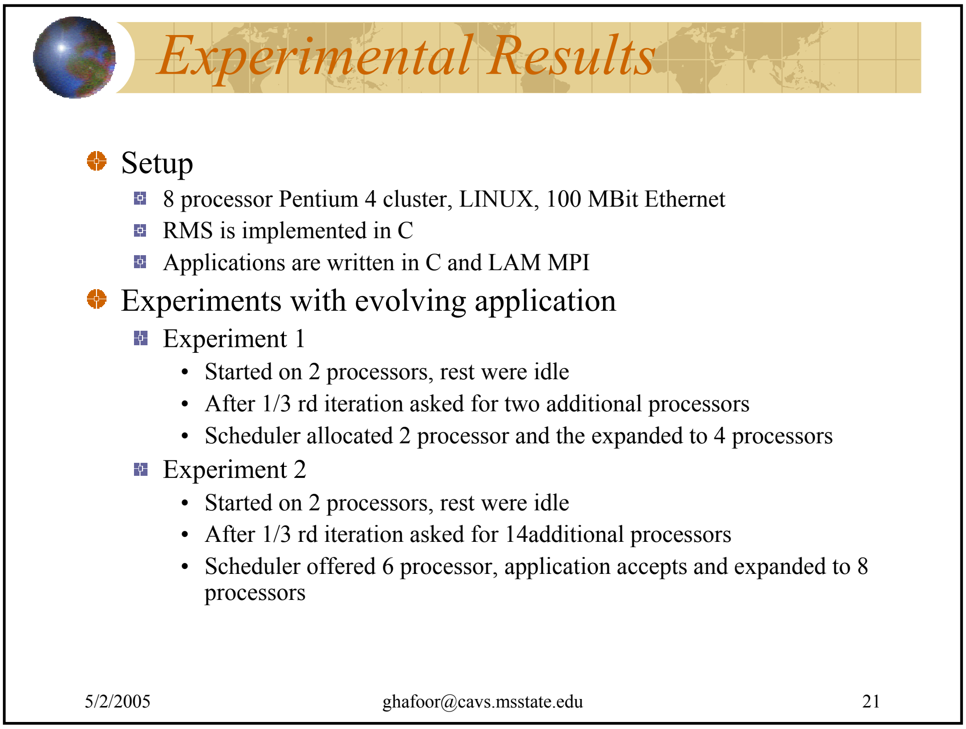
- Consist of Coordinator process and Worker process.
- Coordinator process carries out communication and negotiation.
- Application may block during negotiation
- Resource consumption/release may not happen immediately after successful negotiation.

Structure of Malleable Application

- Start the malleable application
- Start negotiation thread
- Run application specific computation
- Check for adaptation request
- If adaptation is requested, then adapt
- Repeat until computations are complete
- Stop negotiation thread
- End Application

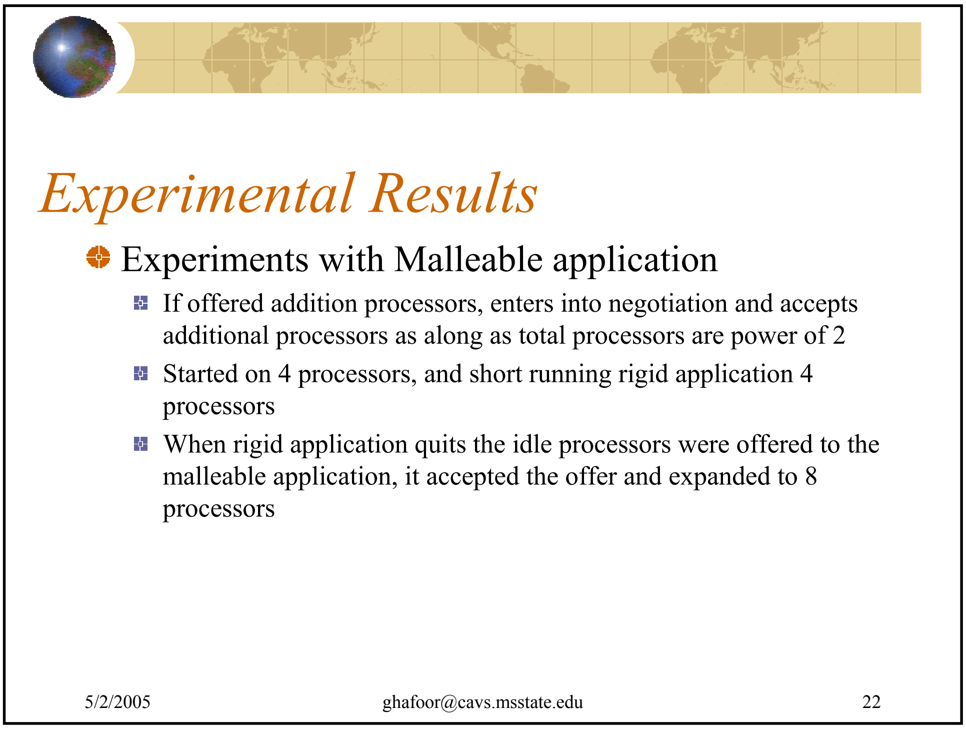
Structure of Evolving Application

- Start the evolving application
- Run application specific computation
- When additional resources are required stop computation
- Request additional resources and carry out negotiation
- If negotiation is successful adapt
- Repeat step 2 – 5 as necessary
- End Application



Experimental Results

- Setup
  - 8 processor Pentium 4 cluster, LINUX, 100 MBit Ethernet
  - RMS is implemented in C
  - Applications are written in C and LAM MPI
- Experiments with evolving application
  - Experiment 1
    - Started on 2 processors, rest were idle
    - After 1/3 rd iteration asked for two additional processors
    - Scheduler allocated 2 processor and the expanded to 4 processors
  - Experiment 2
    - Started on 2 processors, rest were idle
    - After 1/3 rd iteration asked for 14additional processors
    - Scheduler offered 6 processor, application accepts and expanded to 8 processors



Experimental Results

- Experiments with Malleable application
  - If offered addition processors, enters into negotiation and accepts additional processors as along as total processors are power of 2
  - Started on 4 processors, and short running rigid application 4 processors
  - When rigid application quits the idle processors were offered to the malleable application, it accepted the offer and expanded to 8 processors

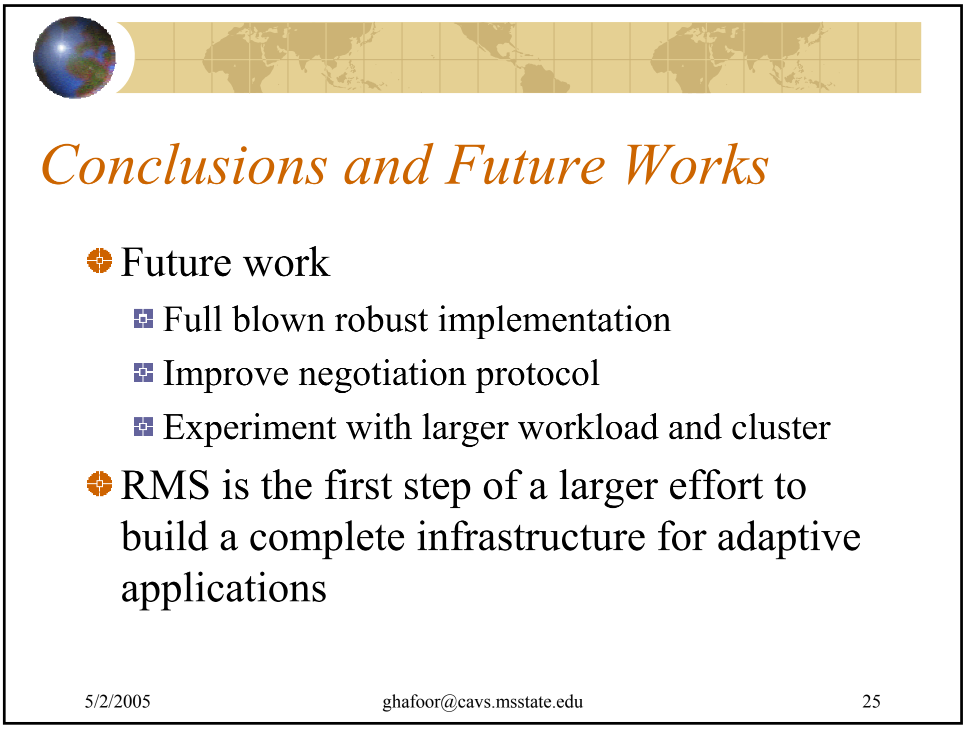
Experimental Results

- Communication Overhead
  - Dummy application: creates NT and send it back and forth to RMS
  - Total time for negotiation (opening connection + send and receive NT+ closing connection) for different rounds of negotiation is measured.

No. of Negotiation Round	1	2	3	4	5
Time in ms	0.03	0.04	0.12	0.20	0.28

Conclusions and Future Works

- Adaptive applications promises
  - Better application performance
  - Improved system utilization
  - New class of parallel application driven by unpredictable events and data
- Prototype implementation is just a proof of concept.
  - Adaptive application is possible
  - The negotiation protocol works and overhead is very low.



Conclusions and Future Works

- Future work
  - Full blown robust implementation
  - Improve negotiation protocol
  - Experiment with larger workload and cluster
- RMS is the first step of a larger effort to build a complete infrastructure for adaptive applications

Questions?