High Fidelity Simulation of Distributed Applications

Vijay Kalpathi Ramanathan

Master’s Defense
The University of Kansas
09/02/2003

Committee:
Dr. Jerry James (Chair)
Dr. Douglas Niehaus
Dr. Susan Gauch
Outline

• Motivation
• Related Work
• Design of Simulation Environment
• Implementation
• Testing
  – Token Ring
  – Bully Algorithm
• Conclusions & Future Work
Motivation

• Distributed Applications
  – Debugging/Testing
  – Difficult to Control
  – Random context switches
    • Reproducibility

• Simulation
  – Building
  – Debugging/Testing
  – Consistent Application code
Related Work

• KU PNNI Simulator
  – Developed at KU

• MONARC Distributed System Simulation
  – Developed at Caltech
  – CERN (Particle physics lab)
KU PNNI Simulator

- Describe/test/instrument PNNI
- Test PNNI Performance
- Reactor
- Scheduling
  - Virtual Time
- Real ATM switch software
KU PNNI Simulator

• Disadvantages

  – Generic Application Simulation

  – Reproducibility of execution sequence

  – Network Modeling
MONARC Distributed System Simulation

• Distributed Computing
  – Physics data processing
• Process oriented
• Java Multi-threading
• Network Model
  – LAN/WAN
MONARC Distributed System Simulation

• Disadvantages
  – Generic Application Simulation
  – Reproducibility of execution sequence
Design

- Single Process
  - Thread per node
  - BThreads
    - User-level threads
    - One kernel thread
  - BERT Reactor
- Virtual Timeline
  - Timing
  - Network Delays
- Network Models
Classes

- **Application**
  - Base class
  - Simulation information
- **SimComm**
  - Communication
- **Network**
  - Network delay
- **SetUp**
- **SignalThreads**
Implementation

- Configuration file
- *SetUp*
- Scheduler
  - Enqueue
    - Virtual Time
  - Dequeue
    - Head
Implementation

• *SimComm*
  – Queues
  – Blocking I/O
  – Timestamp
    • *Network*
  – Receive message
    • Virtual Time > Message TS
    • Wait on condition variable

• *SignalThreads*
  – Increases Virtual Time
  – Signal condition variable
BERT REACTOR

SetUp

The base class of the application object

Reads config file and creates app objects.

SimComm

Responsible for communication

Config file/Netspec (future)

Network

Models the network

Get delay

Return delay

Moves app object to waiting queue using condition variables

Application

The base class of the application object

SimComm

Responsible for communication

Network

Models the network

Config file/Netspec (future)

App

Scheduler queues

Waiting

Ready

SimComm

Responsible for communication

Network

Models the network

Get delay

Return delay

Moves app object to waiting queue using condition variables

Application

The base class of the application object

SimComm

Responsible for communication

Network

Models the network

Get delay

Return delay

Moves app object to waiting queue using condition variables

SimComm

Responsible for communication

Network

Models the network

Get delay

Return delay

Moves app object to waiting queue using condition variables

App

Scheduler queues

Waiting

Ready

BThreads

BERT REACTOR
Application

• Simulated Mode
  – Virtual time
  – State
  – Wrapper functions for SimComm
  – Global instance

• Distributed Mode
  – Wrapper functions for SimComm
SimComm

• Simulated Mode
  – Communication
  – Queues
    • Uni-directional
    • 2 per process pair
    • Array
    • Connection descriptor – Array index

• Distributed Mode
  – Socket API wrapper
  – Connection descriptor = socket
SimComm Queue Allocation

• Queue state
  – INVALID, VALID, CLOSED
  – Allocated in pairs ( (0,1), (2,3) ,(4,5),…)
  – Table of allocated queues
  – Connection descriptor: Write Queue
  – 1’s compliment: Read Queue

• Example A→B A=0, B=1
  0:A→B 1:B→A
Communication

int getConnection(struct sockaddr_in *my_ID, 
                 struct sockaddr_in * peer_ID );

int Send(int ID ,void ** data ,int size );

int Recv(int ID ,void ** data ,int size );

int Close(int ID);
SignalThreads

- Inherits Application
- Array of application object pointers
- Sorts array by virtual time
  - Resembles ready queue
- First Blocked object
  - Increases virtual time
  - Signals condition variable
Testing

- Token Ring Network
  - Simulated mode
  - Distributed mode

- Bully Algorithm
  - Simulated mode
  - Distributed mode
Token Ring Network

- Application object → Node
- Connects to neighbors
- Read from previous node
- Print Token
- Write to next node
- Loop token
Token Ring Design

- ID, IP address, Port number, Starter
- IP address/Port number of neighbors
- Set up
  - Starter
    1. Next node
    2. Previous node
  - Others
    1. Previous node
    2. Next node
## Configuration file

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Start Position</th>
<th>End Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Next node pointer</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Previous node pointer</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>IP address</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>Port number</td>
<td>32</td>
<td>36</td>
</tr>
</tbody>
</table>

ID0100050002diannao.ittc.ku.edu10001  
ID0200010003diannao.ittc.ku.edu10002  
ID0300020004diannao.ittc.ku.edu10003  
ID0400030005diannao.ittc.ku.edu10004  
ID0500040001diannao.ittc.ku.edu10005
Distributed mode

- User
  - Path to configuration file
  - Number of loops
  - ID of node
- Application run in main thread
- ID01 starter
- Run 5 instances
Distributed mode - Output

diannao [205] % SetUp config.txt 2 ID01
Machine ID ID01
LOOP COUNT 1
Token: TOKEN
Machine ID ID01
LOOP COUNT 2
Token: TOKEN
diannao [10] % SetUp config.txt 2 ID02
Machine ID ID02
LOOP COUNT 1
Token: TOKEN
Machine ID ID02
LOOP COUNT 2
Token: TOKEN
<table>
<thead>
<tr>
<th>Output (Cont.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>diannao [9] % SetUp config.txt 2 ID03</td>
</tr>
<tr>
<td>Machine ID ID03</td>
</tr>
<tr>
<td>LOOP COUNT 1</td>
</tr>
<tr>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>Machine ID ID03</td>
</tr>
<tr>
<td>LOOP COUNT 2</td>
</tr>
<tr>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>diannao [9] % SetUp config.txt 2 ID04</td>
</tr>
<tr>
<td>Machine ID ID04</td>
</tr>
<tr>
<td>LOOP COUNT 1</td>
</tr>
<tr>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>Machine ID ID04</td>
</tr>
<tr>
<td>LOOP COUNT 2</td>
</tr>
<tr>
<td>Token: TOKEN</td>
</tr>
</tbody>
</table>
diannao [9] % SetUp config.txt 2 ID05
Machine ID ID05
LOOP COUNT 1
Token: TOKEN
Machine ID ID05
LOOP COUNT 2
Token: TOKEN
Simulated Mode

- User
  - Path to configuration file
  - Number of loops
- ID01 starter
- SetUp creates 5 instances
- Application run in user-level threads
- SignalThreads
Simulated Mode - Output

```

diannao [2] % SetUp config.txt 2
Machine ID ID01
LOOP COUNT 1
Token: TOKEN
Machine ID ID02
LOOP COUNT 1
Token: TOKEN
Machine ID ID03
LOOP COUNT 1
Token: TOKEN
Machine ID ID04
LOOP COUNT 1
Token: TOKEN
Machine ID ID05
LOOP COUNT 1
Token: TOKEN
```
<table>
<thead>
<tr>
<th>Machine ID</th>
<th>ID01</th>
<th>LOOP COUNT</th>
<th>2</th>
<th>Token: TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine ID</td>
<td>ID02</td>
<td>LOOP COUNT</td>
<td>2</td>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>Machine ID</td>
<td>ID03</td>
<td>LOOP COUNT</td>
<td>2</td>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>Machine ID</td>
<td>ID04</td>
<td>LOOP COUNT</td>
<td>2</td>
<td>Token: TOKEN</td>
</tr>
<tr>
<td>Machine ID</td>
<td>ID05</td>
<td>LOOP COUNT</td>
<td>2</td>
<td>Token: TOKEN</td>
</tr>
</tbody>
</table>
Virtual Timeline

<table>
<thead>
<tr>
<th>ID01</th>
<th>VT</th>
<th>ID02</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- GENERATE
- WAIT (1000000000)
- RECV PRINT PASS

VT 0 1292875492
Virtual Timeline

ID03

<table>
<thead>
<tr>
<th>BLOCKS</th>
<th>WAIT (2292875492)</th>
<th>RECV PRINT PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VT</td>
<td>0</td>
<td>1292875492</td>
</tr>
</tbody>
</table>

ID04

<table>
<thead>
<tr>
<th>BLOCKS</th>
<th>WAIT (3560630144)</th>
<th>RECV PRINT PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VT</td>
<td>0</td>
<td>2560630144</td>
</tr>
</tbody>
</table>
# Virtual Timeline

<table>
<thead>
<tr>
<th>ID05</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS</td>
</tr>
<tr>
<td>WAIT (4834618240)</td>
</tr>
<tr>
<td>RECV PRINT PASS</td>
</tr>
</tbody>
</table>

| VT 0 | 3834618240 5095471568 |
Bully Algorithm

• Application object → Node
• Elect Leader
• Send Election message to higher nodes
• Lose if receive acknowledgement
• Winner: Highest ID
• Better Test
  – Communication
  – Concurrency
Bully Algorithm Design

- ID, IP address, Port number
- IP address/Port number of all others
- Set up
  - Identify lower/higher nodes
  - Connect to all
- Receive election message from lower nodes
  - Send acknowledgement
- Send election message to higher nodes.
- Check for acknowledgement
- Win if no acknowledgement
## Configuration file

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Start Position</th>
<th>End Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>IP address</td>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>Port number</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>

0001diannao.ittc.ku.edu10001  
0002diannao.ittc.ku.edu10002  
0003diannao.ittc.ku.edu10003  
0004diannao.ittc.ku.edu10004  
0005diannao.ittc.ku.edu10005  
0006diannao.ittc.ku.edu10006  
0007diannao.ittc.ku.edu10007  
0008diannao.ittc.ku.edu10008  
0009diannao.ittc.ku.edu10009  
0010diannao.ittc.ku.edu10010
Distributed mode

• User
  – Path to configuration file
  – ID of node

• Application run in main thread

• Run 10 instances
Diannao [27] % SetUp config.txt 0001
Diannao [27] % SetUp config.txt 0002
Diannao [27] % SetUp config.txt 0003
Diannao [27] % SetUp config.txt 0004
Diannao [27] % SetUp config.txt 0005
Diannao [27] % SetUp config.txt 0006
Diannao [27] % SetUp config.txt 0007
Diannao [27] % SetUp config.txt 0008
Diannao [27] % SetUp config.txt 0009
Diannao [3] % SetUp config.txt 0010
My ID is 0010. I have won the election.
Simulated Mode

- User
  - Path to configuration file
- \textit{setUp} creates 10 instances
- Application run in user-level threads
- \textit{signalThreads}
Simulated mode - Output

diannao [6] % SetUp config.txt
My ID is 0010. I have won the election.
Conclusions

• Novel Approach
• User-level thread library
• Reactor
• Debugging
• Consistent Application code
  – C++ objects
Conclusions

- Communication
- Network models
- Virtual timeline
  - Sequence events
  - Message delivery delayed
- Token Ring
- Bully Algorithm
Future Work

• Reactor
  – Record debugging info
  – Replay execution

• Network Model
  – Constant delay
  – Dynamic
  – Message size, Bandwidth, source, destination
Thank You