A Dependable Middleware for Enhancing the Fault Tolerance of Distributed Computations in Grid Environments

André Luckow

Potsdam University
Institute for Computer Science
Operating Systems and Distributed Systems

Potsdam, July 23, 2009
Outline

1 Motivation and Introduction
2 Related Work
3 Migol Architecture
4 Fault Tolerance
5 Grid Experiments
6 Summary and Contributions
Motivation

- “A distributed system is a system on which I cannot get any work done because some machine I never heard of has failed.” L. Lamport
- “All things fail all the time.” W. Vogels

Failures are inevitable in complex and large-scale systems such as clusters and Grids.
Motivation

- “A distributed system is a system on which I cannot get any work done because some machine I never heard of has failed.” L. Lamport
- “All things fail all the time.” W. Vogels

Failures are inevitable in complex and large-scale systems such as clusters and Grids.
Motivation

Motivation

HTTP Status 500 -

```
Type: Exception report

Message: The server encountered an internal error () that prevented it from fulfilling this request.

Exception:

org.globus.wsrf.util.FaultException
org.globus.mds.webmds.xmlSources.xmlDOMNode.NodeXmlSource.getXmlSource(NodeXmlSource.java:128)
org.globus.mds.webmds.WebmdsServlet.getSource(WebmdsServlet.java:426)
org.globus.mds.webmds.WebmdsServlet.doGet(WebmdsServlet.java:170)
javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
```

Note: The full stack trace of the root cause is available in the Apache Tomcat/5.0 logs.

Apache Tomcat/5.0

(D-Grid MDS)
Motivation

A new message has been posted to TeraGrid News.

Sites: LONI
Affected Resources: loni-1su-qb

Start time: 10-JUL-2009 05:00 CT
End time: 10-JUL-2009 17:00

Due to the loss of air conditioning, all running jobs have been killed on Queen Bee and the compute nodes have been shutdown. When air conditioning is restored, Queen Bee will go back into production. Sorry for the inconvenience.

Posted on 10-JUL-2009 05:51 US Pacific by Sam White
The Migol Framework

- **Migol (Migration in the Grid OGSA Lite)** – A fault-tolerant service framework for long-running Grid applications.

- Migol provides an autonomic, self-healing environment for starting, monitoring, and migrating applications in the Grid.

- Migol is based on the Globus Toolkit 4.

- **Challenges:**
  - Grids are highly complex:
    - Dynamic.
    - Heterogeneous.
    - Different administrative domains.
    - Many levels of software and hardware.
  - Complex failure modes (e.g. network partitionings).
  - Accurate failure detection.
  - Achieving Consensus.
The Migol Framework

- **Migol (Migration in the Grid OGSA Lite)** – A fault-tolerant service framework for long-running Grid applications.

- Migol provides an autonomic, self-healing environment for starting, monitoring, and migrating applications in the Grid.

- Migol is based on the Globus Toolkit 4.

- Challenges:
  - Grids are highly complex:
    - Dynamic.
    - Heterogeneous.
    - Different administrative domains.
    - Many levels of software and hardware.
  - Complex failure modes (e.g. network partitionings).
  - Accurate failure detection.
  - Achieving Consensus.
The Migol Framework

- **Migol (Migration in the Grid OGSA Lite)** – A fault-tolerant service framework for long-running Grid applications.

- Migol provides an autonomic, self-healing environment for starting, monitoring, and migrating applications in the Grid.

- Migol is based on the Globus Toolkit 4.

**Challenges:**

- Grids are highly complex:
  - Dynamic.
  - Heterogeneous.
  - Different administrative domains.
  - Many levels of software and hardware.

- Complex failure modes (e.g. network partitionings).

- Accurate failure detection.

- Achieving Consensus.
Outline

1 Motivation and Introduction

2 Related Work

3 Migol Architecture

4 Fault Tolerance
   - Fault Tolerance of Migol Infrastructure
   - Fault Tolerance of Applications

5 Grid Experiments

6 Summary and Contributions
Related Work

- **Fault Tolerant MPI:**
  - FT-MPI (Fagg et al.)
  - MPICH-V (Bouteiller et al.)
  - EasyGrid (Silva et al.)

- **Fault Tolerant Grid Frameworks:**
  - Globus Heartbeat Monitor (Stelling et al.)
  - Cactus (Lanfermann et al.)
  - Condor-G (Livny et al.), Condor/PGRADE (Kovacs et al.)

- **Checkpoint Replication:**
  - Replica Location Service, Data Replication Service (Chervenak et al.)

Frameworks provide fault tolerance only for a limited number of faults and often introduce single points of failures.
Related Work

- Fault Tolerant MPI:
  - FT-MPI (Fagg et al.)
  - MPICH-V (Bouteiller et al.)
  - EasyGrid (Silva et al.)

- Fault Tolerant Grid Frameworks:
  - Globus Heartbeat Monitor (Stelling et al.)
  - Cactus (Lanfermann et al.)
  - Condor-G (Livny et al.), Condor/PGRADE (Kovacs et al.)

- Checkpoint Replication:
  - Replica Location Service, Data Replication Service (Chervenak et al.)

Frameworks provide fault tolerance only for a limited number of faults and often introduce single points of failures.
Outline

1 Motivation and Introduction

2 Related Work

3 Migol Architecture

4 Fault Tolerance
   ■ Fault Tolerance of Migol Infrastructure
   ■ Fault Tolerance of Applications

5 Grid Experiments

6 Summary and Contributions
Migol Architecture
Migol Architecture

1. submit
2. registerService
3. query
4. reserve/submit

User

Job Broker Service

Advance Reservation Service

WS MDS

Application Information Service

Monitoring Restart Service

Migration Service

Compute Resource

WS GRAM

GridFTP

Application

Migol Library/SAGA/Migol

Checkpoint Replication Service

Migol Component

Globus Component
Migol Architecture

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/ updateGSO
8. query

Migol Component

Globus Component
Migol Architecture

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/updateGSO
8. query
9. monitor
Migol Architecture

User

1. submit

Job Broker Service

2. registerService

Advance Reservation Service

3. query

Migration Service

4. reserve/submit

WS MDS

5. updateGSO

Application Information Service

6. replicate

Monitoring Restart Service

7. get/updateGSO

Migration Service

8. query

9. monitor

Checkpoint Replication Service

Migol Component

Globus Component

André Luckow (Potsdam University)
Migol Architecture

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/updateGSO
8. query
9. monitor
10. restart
10. resubmit
Outline

1 Motivation and Introduction
2 Related Work
3 Migol Architecture
4 Fault Tolerance
   ▪ Fault Tolerance of Migol Infrastructure
   ▪ Fault Tolerance of Applications
5 Grid Experiments
6 Summary and Contributions
Fault Tolerance of Migol Infrastructure

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/updateGSO
8. query
9. monitor
10. resubmit
10. restart
10. resubmit
Fault Tolerance of Migol Infrastructure

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/updateGSO
8. query
9. monitor
10. resubmit
10. restart

Less Critical

Migration Service
Job Broker Service
Advance Reservation Service
Application Information Service
Monitoring Restart Service
Checkpoint Replication Service

Job Submission Process:
1. User submits a job.
2. Job Broker Service registers the service.
3. Query service.
4. Reserve/submit service.
5. Update GSO.
6. Replicate.
7. Get/Update GSO.
8. Query.
10. Resubmit/restart.

Components:
- Migol Component
- Globus Component
Fault Tolerance of Migol Infrastructure

1. submit
2. registerService
3. query
4. reserve/submit
5. updateGSO
6. replicate
7. get/updateGSO
8. query
9. monitor
10. resubmit
10. restart

Critical

Less Critical

Migol Component

Globus Component

André Luckow (Potsdam University)
Fault Tolerance of the Application Information Service (AIS)
Fault Tolerance of the Application Information Service (AIS)

Web Service Ring
Replication Protocol:
Grid-enabled, token-based ordering and membership protocol (Luckow et al, 2008)
Fault Tolerance of the Application Information Service (AIS)

Grid

Site 1
Application Information Service

Site 2
Application Information Service

Web Service Ring Replication Protocol:
Grid-enabled, token-based ordering and membership protocol (Luckow et al, 2008)

Site 2
Application Information Service
Replication Service
Plugin API
WS RRP
JGroups

JGroups:
Java-based toolkit for group communication (Ban, 1998)
WS RRP
Total Ordering Protocol

André Luckow (Potsdam University)
WS RRP

Total Ordering Protocol

- register service

AIS 1

AIS 2

AIS 3

AIS 4

token seq: 101
next seq: 2
rtr: {}
messages: {register_service[id:1;...]}

André Luckow (Potsdam University)
WS RRP
Total Ordering Protocol

- register service

AIS 1

AIS 2

AIS 3

AIS 4

token seq: 102
next seq: 2
rtr: {}
messages: {register_service[id:1;...]}

André Luckow (Potsdam University)
WS RRP
Total Ordering Protocol

AIS
register
service
token seq: 103
next seq: 2
rtr: {}
messages: {register_service[id:1;...]}

André Luckow (Potsdam University)
WS RRP
Total Ordering Protocol

token seq: 104
next seq: 2
rtr: {}
messages:
{register_service[id:1;...]}
WS RRP
Total Ordering Protocol

AIS 1

register service

token seq: 105
next seq: 2
rtr: {}
messages: {}

AIS 2

AIS 3

AIS 4
Failure Detection

![Diagram showing fault tolerance in a system with three nodes (Node 1, Node 2, Node 3) and emphasizing the token roundtrip time, message transmission time, and processing time.]

- Token roundtrip time
- Message transmission time
- Processing time
- Message transmission timer
- Token roundtrip timer

André Luckow (Potsdam University)
WS RRP
Failure Detection

Exponential averaging of round trip times:

\[
\begin{align*}
\text{mean}_\text{rtt} &= \alpha \cdot \text{mean}_\text{rtt} + (1 - \alpha) \cdot \text{current}_\text{rtt} \\
\text{dev}_\text{rtt} &= \alpha \cdot \text{dev}_\text{rtt} + (1 - \alpha) \cdot |\text{mean}_\text{rtt} - \text{current}_\text{rtt}| \\
\text{est}_\text{rtt} &= \text{mean}_\text{rtt} + 4 \cdot \text{dev}_\text{rtt}
\end{align*}
\]

Load factor for approximation of processing time:

\[
\text{load}_\text{factor} = \text{number}_\text{active}_\text{nodes} \cdot \text{number}_\text{msg} \cdot \text{MSG}_\text{PROC}_\text{TIME}
\]

Timeout:

\[
\text{token}_\text{timeout} = \text{est}_\text{rtt} + \text{load}_\text{factor}
\]
WS RRP
Failure Detection

Exponential averaging of round trip times:

\[
\begin{align*}
\text{mean}_\text{rtt} &= \alpha \cdot \text{mean}_\text{rtt} + (1 - \alpha) \cdot \text{current}_\text{rtt} \\
\text{dev}_\text{rtt} &= \alpha \cdot \text{dev}_\text{rtt} + (1 - \alpha) \cdot |\text{mean}_\text{rtt} - \text{current}_\text{rtt}| \\
\text{est}_\text{rtt} &= \text{mean}_\text{rtt} + 4 \cdot \text{dev}_\text{rtt}
\end{align*}
\]

Load factor for approximation of processing time:

\[
\text{load}_\text{factor} = \text{number}_\text{active}_\text{nodes} \cdot \text{number}_\text{msg} \cdot \text{MSG}_\text{PROC}_\text{TIME}
\]

Timeout:

\[
\text{token}_\text{timeout} = \text{est}_\text{rtt} + \text{load}_\text{factor}
\]
WS RRP
Failure Detection

Exponential averaging of round trip times:

\[
\begin{align*}
\text{mean}_\text{rtt} &= \alpha \cdot \text{mean}_\text{rtt} + (1 - \alpha) \cdot \text{current}_\text{rtt} \\
\text{dev}_\text{rtt} &= \alpha \cdot \text{dev}_\text{rtt} + (1 - \alpha) \cdot | \text{mean}_\text{rtt} - \text{current}_\text{rtt} | \\
\text{est}_\text{rtt} &= \text{mean}_\text{rtt} + 4 \cdot \text{dev}_\text{rtt}
\end{align*}
\]

Load factor for approximation of processing time:

\[
\text{load}_\text{factor} = \text{number}_\text{active}_\text{nodes} \cdot \text{number}_\text{msg} \cdot \text{MSG}_\text{PROC}_\text{TIME}
\]

Timeout:

\[
\text{token}_\text{timeout} = \text{est}_\text{rtt} + \text{load}_\text{factor}
\]
WS RRP
Membership Protocol
WS RRP
Membership Protocol

AIS 1
State: Operational
active: 1, 2, 3, 4

AIS 2
State: Operational
active: 1, 2, 3, 4

AIS 3
State: Operational
active: 1, 2, 3, 4

Join
active: 1, 2, 3, 4

AIS 4
State: Operational
active: 1, 2, 3, 4
WS RRP
Membership Protocol

**AIS 1**
State: Operational
active: 1, 2, 3, 4
Consensus: no
3: yes

**AIS 2**

**AIS 3**
State: Operational
active: 1, 2, 3, 4
Consensus: no
3: yes

**AIS 4**
State: Operational
active: 1, 2, 3, 4
Consensus: no
3: yes

Join
active: 1, 2, 3, 4
WS RRP
Membership Protocol
WS RRP
Membership Protocol

AIS 1
State: Operational
active: 1, 3, 4
Consensus: no
3: yes

AIS 4
State: Operational
active: 1, 2, 4
Consensus: no
3: yes

AIS 3
State: Operational
active: 1, 2, 4
Consensus: no
3: yes

Join
active: 1, 3, 4

André Luckow (Potsdam University)
**WS RRP**

Membership Protocol

- **AIS 1**: State: Operational, active: 1, 3, 4, Consensus: no
  
- **AIS 2**: (Inactive)
  
- **AIS 3**: State: Operational, active: 1, 2, 4, Consensus: no
  
- **AIS 4**: State: Operational, active: 1, 2, 4, Consensus: no
  
- **Join**: active: 1, 3, 4
**WS RRP**

Membership Protocol

- **AIS 1**
  - State: Operational
  - active: 1, 3, 4
  - Consensus: yes
  - 1: yes
  - 3: yes
  - 4: yes

- **AIS 2**
  - (Removed)

- **AIS 3**
  - State: Operational
  - active: 1, 2, 4
  - Consensus: yes
  - 1: yes
  - 3: yes
  - 4: yes

- **AIS 4**
  - State: Operational
  - active: 1, 2, 4
  - Consensus: yes
  - 1: yes
  - 3: yes
  - 4: yes

- **Join**
  - active: 1, 3, 4
Outline

1. Motivation and Introduction
2. Related Work
3. Migol Architecture
4. Fault Tolerance
   - Fault Tolerance of Migol Infrastructure
   - Fault Tolerance of Applications
5. Grid Experiments
6. Summary and Contributions
Fault Tolerance of Applications

Failure Propagation

Grid Layer (Globus/Migol)
Job Layer (LRMS)
Application Layer
MPI Layer

Rank 0
Rank 1
Rank ...
Rank n
Fault Tolerance of Applications

Failure Propagation
Fault Tolerance of Applications

Failure Detection

Monitoring
Restart Service

Application (Rank 0)
Application (Rank 1)
Application (Rank 2)
Application (Rank 3)

Migol_Init()

checkService()

true
Fault Tolerance of Applications

Failure Detection

- Monitoring
- Restart Service
- Application (Rank 0)
- Application (Rank 1)
- Application (Rank 2)
- Application (Rank 3)

Migol_Init()

checkService()

true

timeout

false

André Luckow (Potsdam University)
Outline

1. Motivation and Introduction
2. Related Work
3. Migol Architecture
4. Fault Tolerance
   - Fault Tolerance of Migol Infrastructure
   - Fault Tolerance of Applications
5. Grid Experiments
6. Summary and Contributions
Grid Experiments

WS RRP/JGroups: Load Test with 4 Sites (Cluster)

Testbed: Einstein Cluster (4 sites)

André Luckow (Potsdam University)
Grid Experiments

WS RRP/JGroups: Load Test with 4 Sites (Grid)

Testbed: 2 sites EC² and 2 sites D-Grid
Grid Experiments
Long Run @ UP Grid

- **Scenario:**
  - Cellular Automat (MPI-based Mesh Computation)
  - Grid Resources: Einstein (PBSPro), Highland (Torque), Uranus (Condor)
  - Migol infrastructure: Flotta, Hoy, Stronsay (all Potsdam University)

- **Interim Results:**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>53 days 23 h</td>
</tr>
<tr>
<td>Number of job failures</td>
<td>83</td>
</tr>
<tr>
<td>Number of AIS reconfigurations</td>
<td>4</td>
</tr>
<tr>
<td>Mean Time To Failure</td>
<td>14 h 26 min</td>
</tr>
<tr>
<td>Mean Time To Repair</td>
<td>4 min 11 s (automatic)</td>
</tr>
<tr>
<td></td>
<td>1 h 7 min (w/ manual)</td>
</tr>
<tr>
<td>Manual Recoveries</td>
<td>3</td>
</tr>
</tbody>
</table>
Grid Experiments
Long Run @ UP Grid

■ Scenario:
  ■ Cellular Automat (MPI-based Mesh Computation)
  ■ Grid Resources: Einstein (PBSPro), Highland (Torque), Uranus (Condor)
  ■ Migol infrastructure: Flotta, Hoy, Stronsay (all Potsdam University)

■ Interim Results:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>53 days 23 h</td>
</tr>
<tr>
<td>Number of job failures</td>
<td>83</td>
</tr>
<tr>
<td>Number of AIS reconfigurations</td>
<td>4</td>
</tr>
<tr>
<td>Mean Time To Failure</td>
<td>14 h 26 min</td>
</tr>
<tr>
<td>Mean Time To Repair</td>
<td>4 min 11 s (automatic)</td>
</tr>
<tr>
<td></td>
<td>1 h 7 min (w/ manual)</td>
</tr>
<tr>
<td>Manual Recoveries</td>
<td>3</td>
</tr>
</tbody>
</table>
Summary and Contributions

**Major Contributions:**
- New approach for systematic fault tolerance within Grid environments.
- Novel replication protocol for active replication within Grids.
- Successful implementation and validation of the system within the D-Grid and TeraGrid.
- Multiple applications use Migol today: AMIGA, Replica-Exchange.


Thank You!

“Design for failure and nothing will really fail.” (Simone Brunozzi)
Summary and Contributions

Publications I


Publications II

A. Luckow.
A Dependable Middleware for Enhancing the Fault Tolerance of Distributed Computations in Grid Environments.

A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor.
Distributed Replica-Exchange Simulations on Production Environments using SAGA and Migol.

A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor.
Adaptive Replica-Exchange Simulations.
A. Luckow and B. Schnor.

A. Luckow and B. Schnor.
Adaptive Checkpoint Replication for Supporting the Fault Tolerance of Applications in the Grid.
A. Luckow and B. Schnor.
Migol: A Fault-Tolerant Service Framework for MPI Applications in the Grid.

A. Luckow and B. Schnor.
Service Replication in Grids: Ensuring Consistency in a Dynamic, Failure-Prone Environment.
In *Proceedings of Fifth High-Performance Grid Computing Workshop in conjunction with IEEE International Parallel & Distributed Processing Symposium, Miami, USA, 2008.*
Backup
Grid Experiments

Job Submission and Migration @ UP Grid

![Bar chart showing submission times for GRAM2, GRAM4 (C), GRAM4 (Java), JBS, and MS, with submission times and optimized submission times compared.]
Grid Experiments

Replica-Exchange Simulations

- **Replica-Exchange (RE) simulations** are used to understand important physical phenomena – from protein folding to binding affinity calculations for computational drug discovery.

- **Pleasingly distributed:** In principal loosely coupled – however some synchronization required between tasks.

![Diagram](image-url)
Grid Experiments
Replica-Exchange @ TeraGrid