

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



 D-Grid
 DGI AstroGrid MediGrid C1-Grid HEP InGrid TextGrid
 GridCSM
Grid Computing Service Manager

Astrogrid Portal
Portal
MDS
Drill-Down
Computing Elements
Cluster Workmodes
Sites and Warnings
Topologie
Details XML-Datas

HTTP Status 500 -

type Exception report

message

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

exception

```

org.globus.wsrfl.util.FaultHelper.toBaseFault(FaultHelper.java:282)
org.globus.mds.webmds.xmlSources.resourceProperties.ResourcePropertyHelper.runResourcePropertyQuery(ResourcePropertyHelper.java:164)
org.globus.mds.webmds.xmlSources.resourceProperties.ResourcePropertyHelper.queryResourceProperties(ResourcePropertyHelper.java:116)
org.globus.mds.webmds.xmlSources.resourceProperties.ResourcePropertyQueryNodeSource.getNode(ResourcePropertyQueryNodeSource.java:106)
org.globus.mds.webmds.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:743)
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-lsu-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.

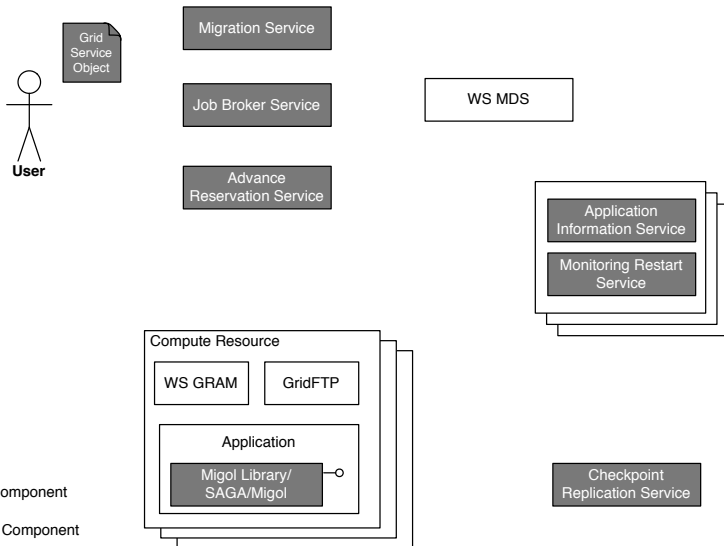


Outline

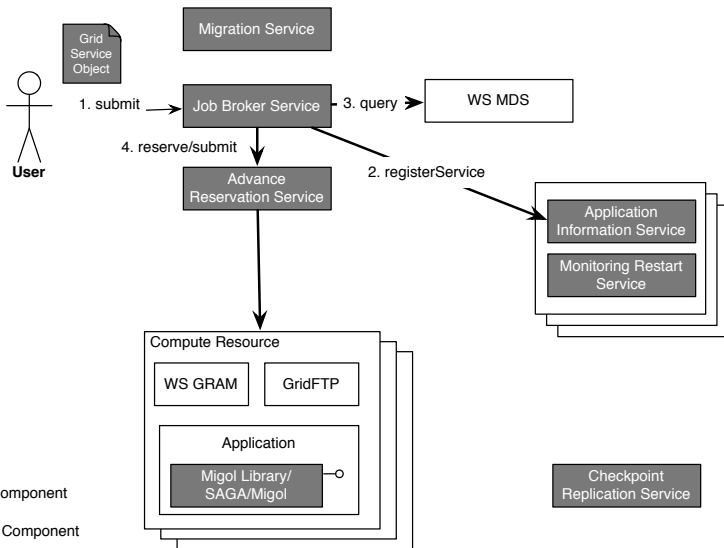
- 1 Motivation and Introduction
- 2 Related Work
- 3 Migol Architecture**
- 4 Fault Tolerance
 - Fault Tolerance of Migol Infrastructure
 - Fault Tolerance of Applications
- 5 Grid Experiments
- 6 Summary and Contributions



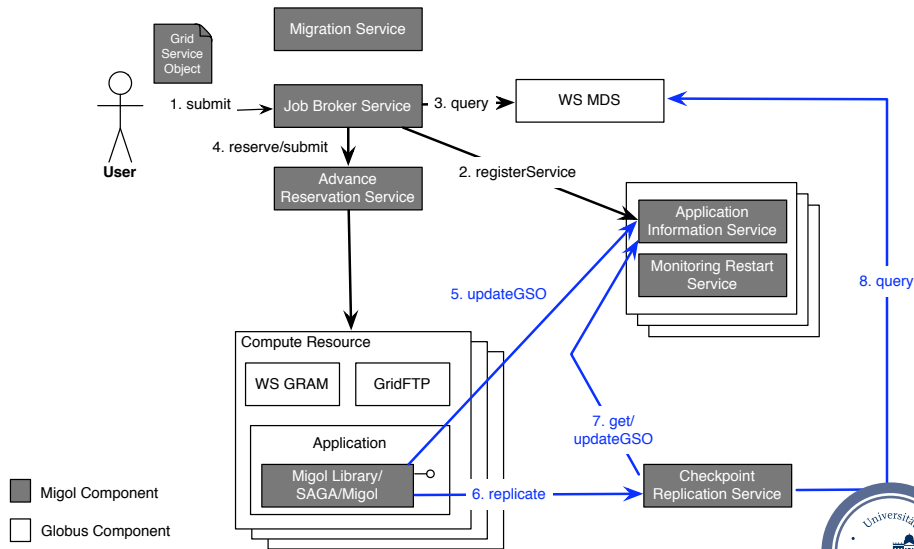
Migol Architecture



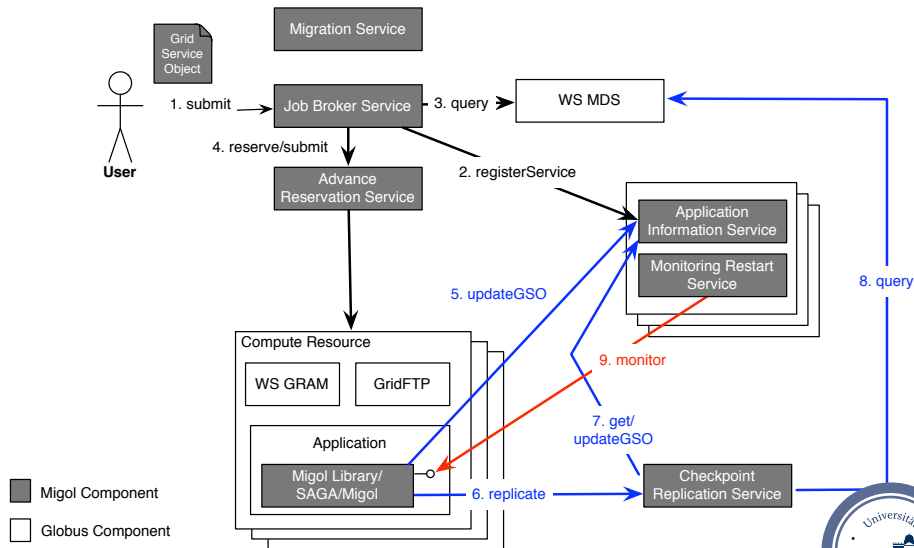
Migol Architecture



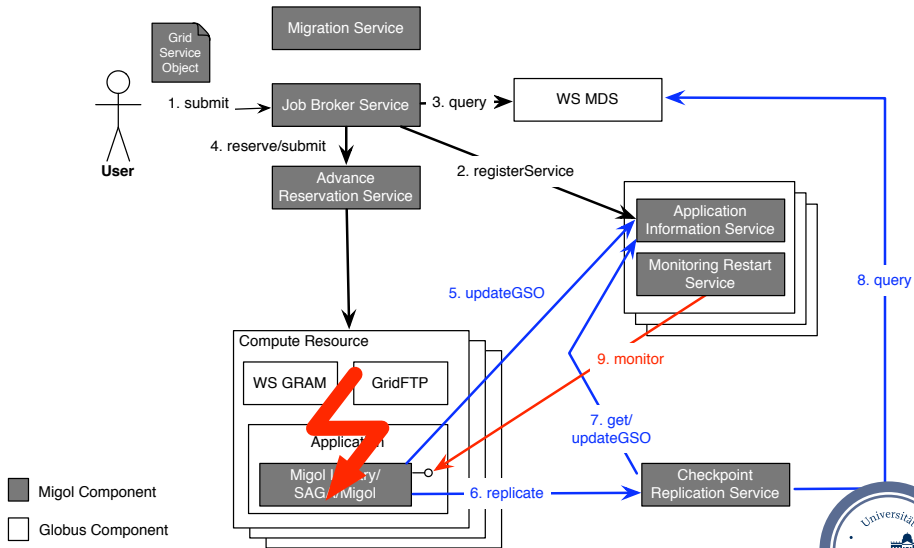
Migol Architecture



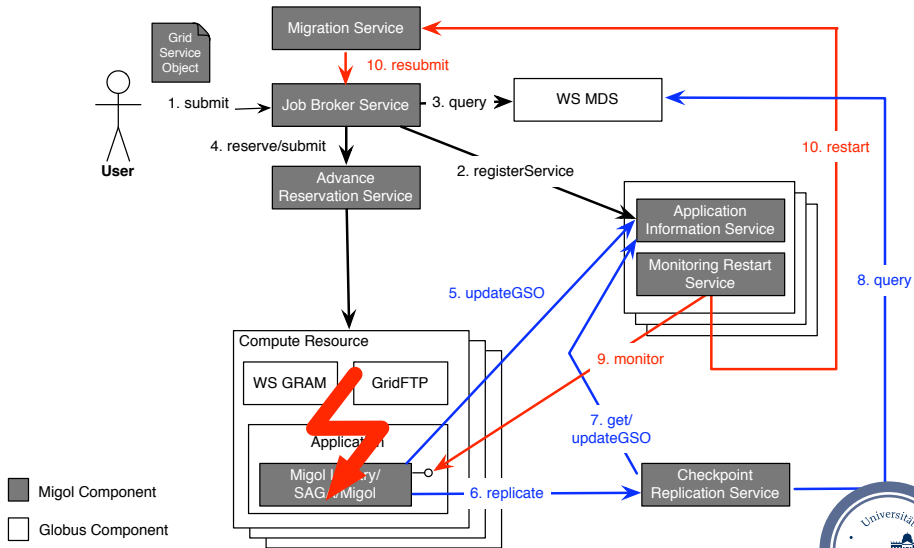
Migol Architecture



Migol Architecture



Migol Architecture

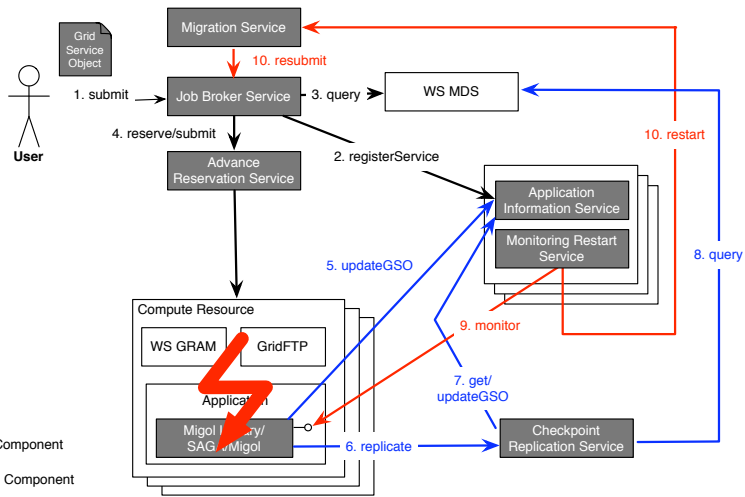


Outline

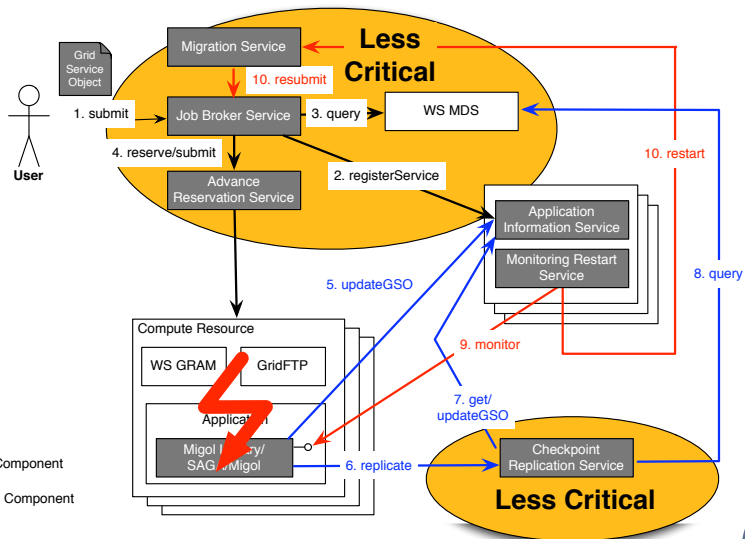
- 1 Motivation and Introduction
- 2 Related Work
- 3 Migol Architecture
- 4 Fault Tolerance**
 - Fault Tolerance of Migol Infrastructure
 - Fault Tolerance of Applications
- 5 Grid Experiments
- 6 Summary and Contributions



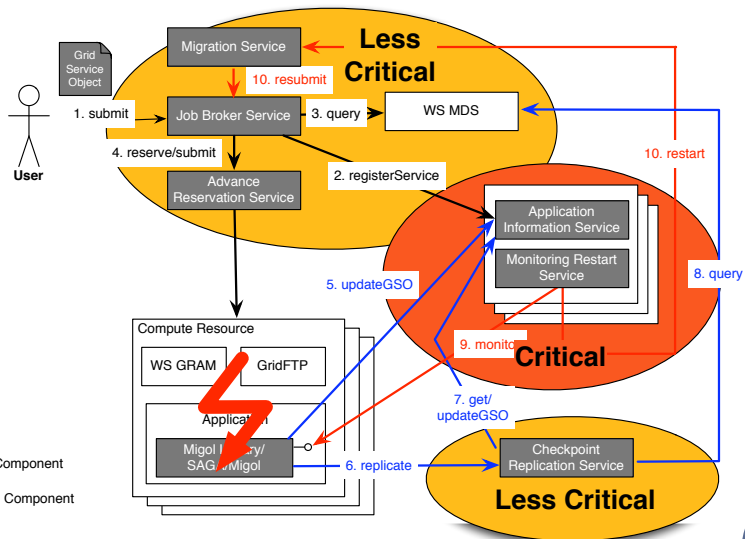
Fault Tolerance of Migol Infrastructure



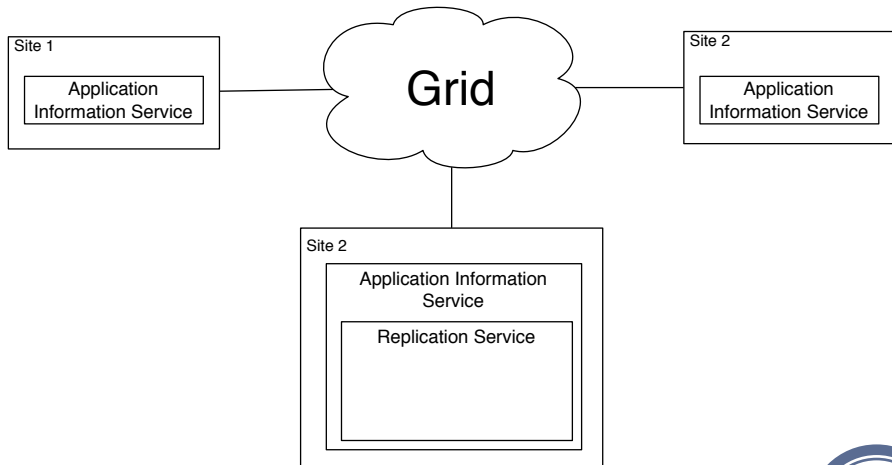
Fault Tolerance of Migol Infrastructure



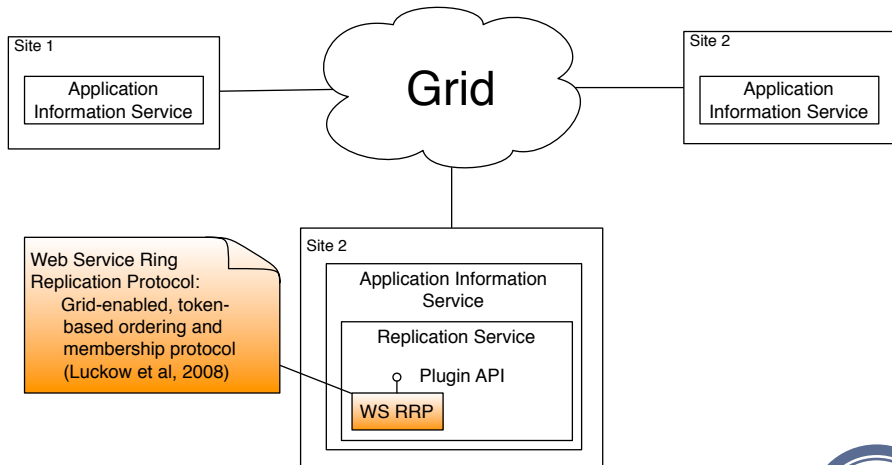
Fault Tolerance of Migol Infrastructure



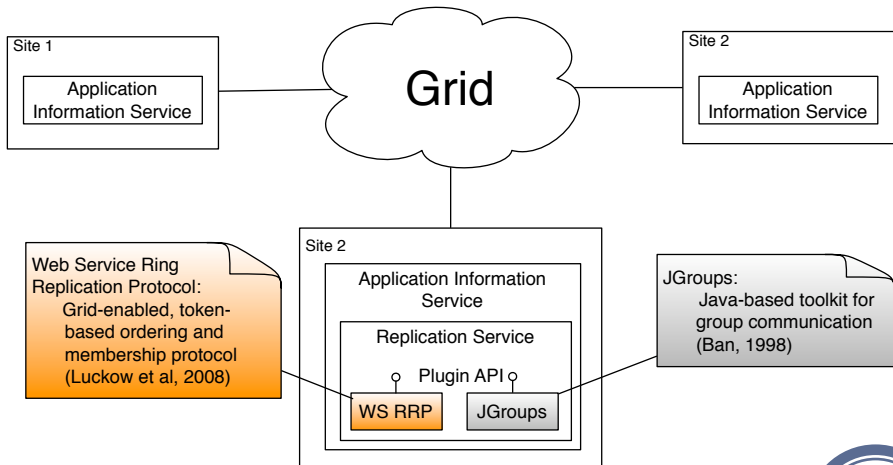
Fault Tolerance of the Application Information Service (AIS)



Fault Tolerance of the Application Information Service (AIS)

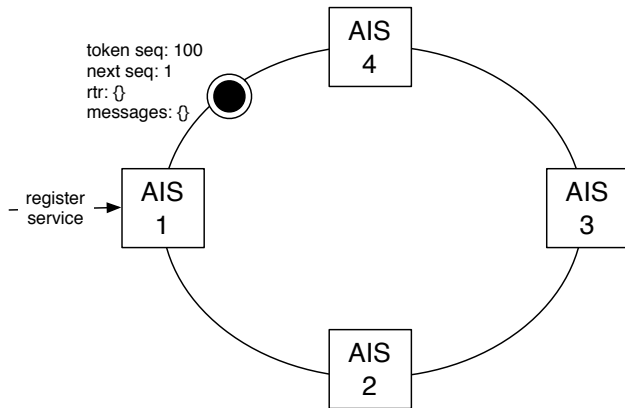


Fault Tolerance of the Application Information Service (AIS)



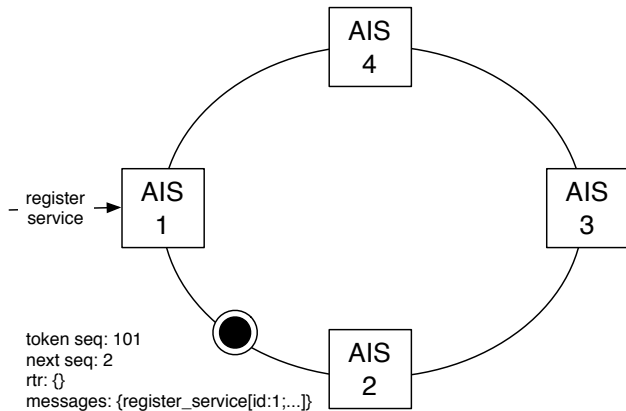
WS RRP

Total Ordering Protocol



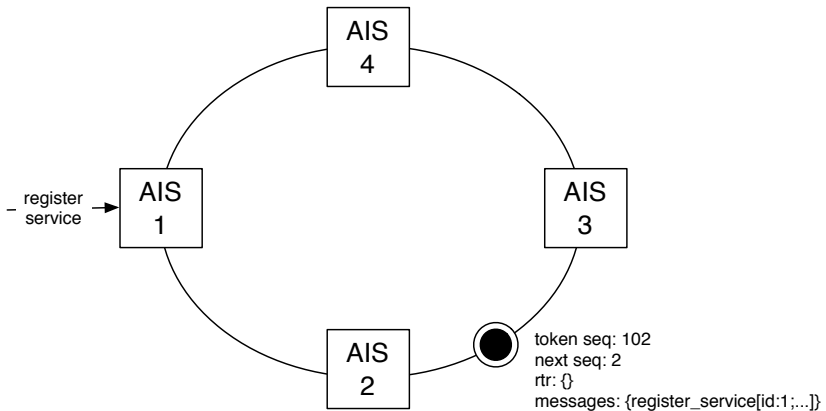
WS RRP

Total Ordering Protocol



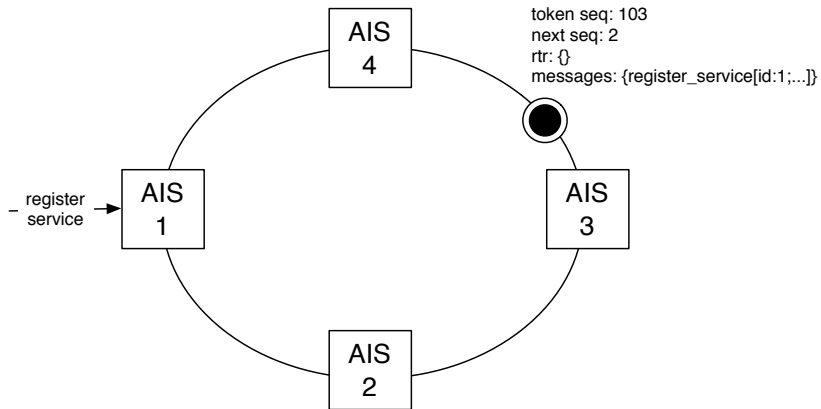
WS RRP

Total Ordering Protocol



WS RRP

Total Ordering Protocol



WS RRP

Total Ordering Protocol

token seq: 104

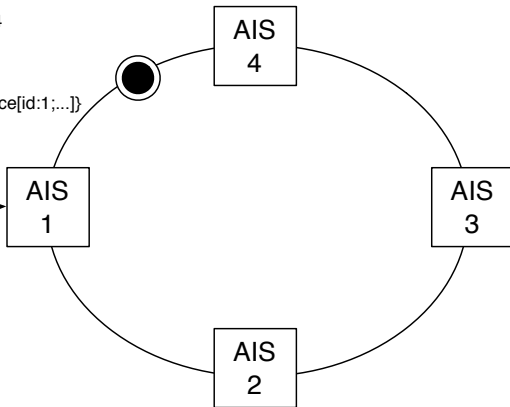
next seq: 2

rtr: {}

messages:

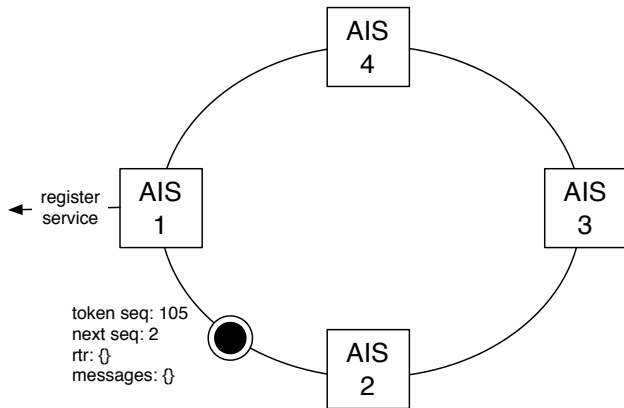
{register_service[id:1;...]}

- register
service →

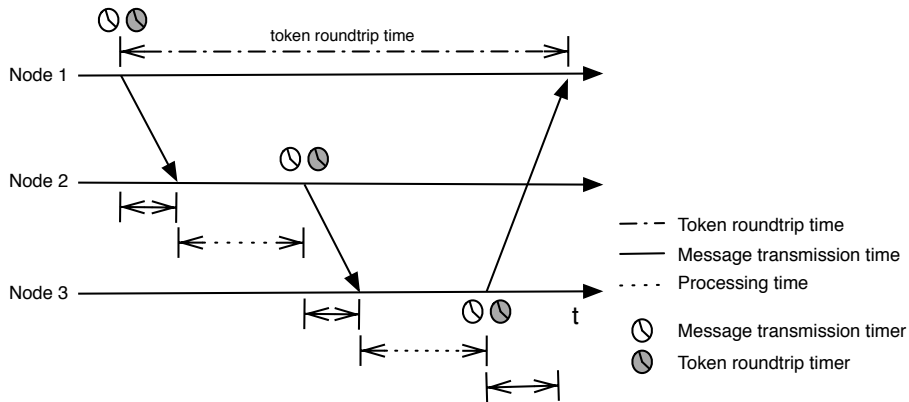


WS RRP

Total Ordering Protocol



Failure Detection



WS RRP

Failure Detection

Exponential averaging of round trip times:

$$\begin{aligned} \text{mean_rtt} &= \alpha \cdot \text{mean_rtt} + (1 - \alpha) \text{ current_rtt} \\ \text{dev_rtt} &= \alpha \cdot \text{dev_rtt} + (1 - \alpha) | \text{mean_rtt} - \text{current_rtt} | \\ \text{est_rtt} &= \text{mean_rtt} + 4 \cdot \text{dev_rtt} \end{aligned}$$

Load factor for approximation of processing time:

$$\text{load_factor} = \text{number_active_nodes} \cdot \text{number_msg} \cdot \text{MSG_PROC_TIME}$$

Timeout:

$$\text{token_timeout} = \text{est_rtt} + \text{load_factor}$$



WS RRP

Failure Detection

Exponential averaging of round trip times:

$$\begin{aligned} \text{mean_rtt} &= \alpha \cdot \text{mean_rtt} + (1 - \alpha) \text{ current_rtt} \\ \text{dev_rtt} &= \alpha \cdot \text{dev_rtt} + (1 - \alpha) | \text{mean_rtt} - \text{current_rtt} | \\ \text{est_rtt} &= \text{mean_rtt} + 4 \cdot \text{dev_rtt} \end{aligned}$$

Load factor for approximation of processing time:

$$\text{load_factor} = \text{number_active_nodes} \cdot \text{number_msg} \cdot \text{MSG_PROC_TIME}$$

Timeout:

$$\text{token_timeout} = \text{est_rtt} + \text{load_factor}$$



WS RRP

Failure Detection

Exponential averaging of round trip times:

$$\begin{aligned} \text{mean_rtt} &= \alpha \cdot \text{mean_rtt} + (1 - \alpha) \text{ current_rtt} \\ \text{dev_rtt} &= \alpha \cdot \text{dev_rtt} + (1 - \alpha) | \text{mean_rtt} - \text{current_rtt} | \\ \text{est_rtt} &= \text{mean_rtt} + 4 \cdot \text{dev_rtt} \end{aligned}$$

Load factor for approximation of processing time:

$$\text{load_factor} = \text{number_active_nodes} \cdot \text{number_msg} \cdot \text{MSG_PROC_TIME}$$

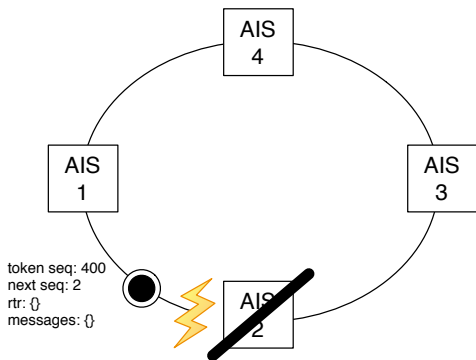
Timeout:

$$\text{token_timeout} = \text{est_rtt} + \text{load_factor}$$



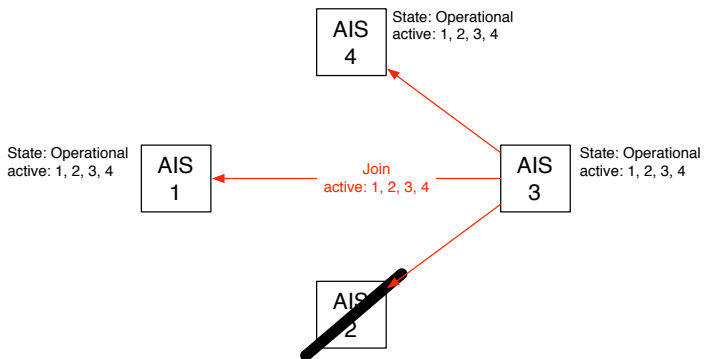
WS RRP

Membership Protocol



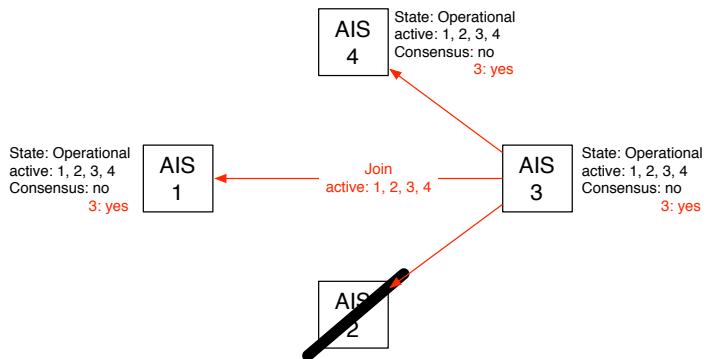
WS RRP

Membership Protocol



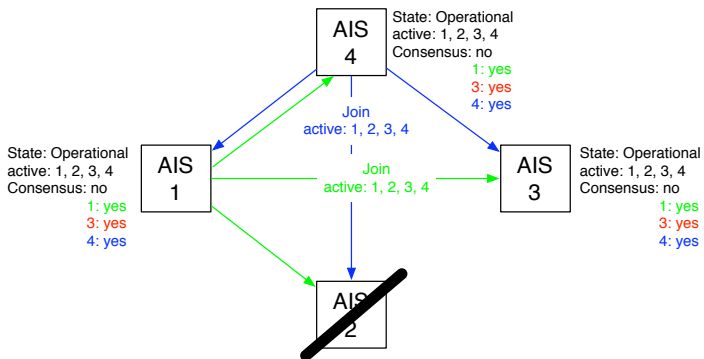
WS RRP

Membership Protocol



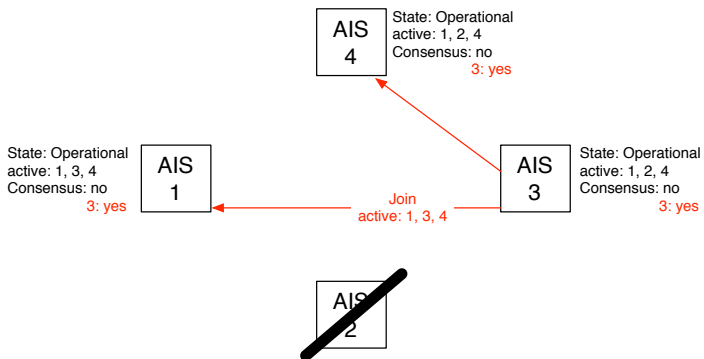
WS RRP

Membership Protocol



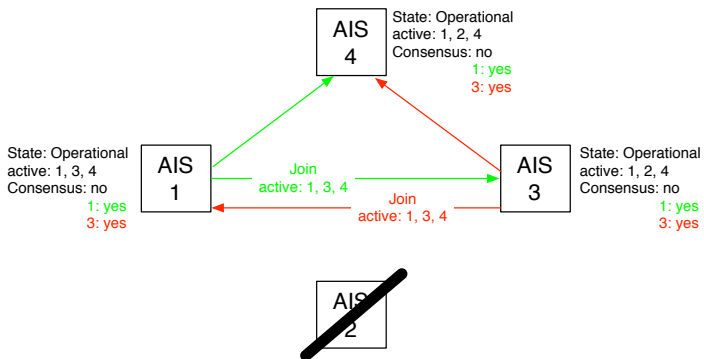
WS RRP

Membership Protocol



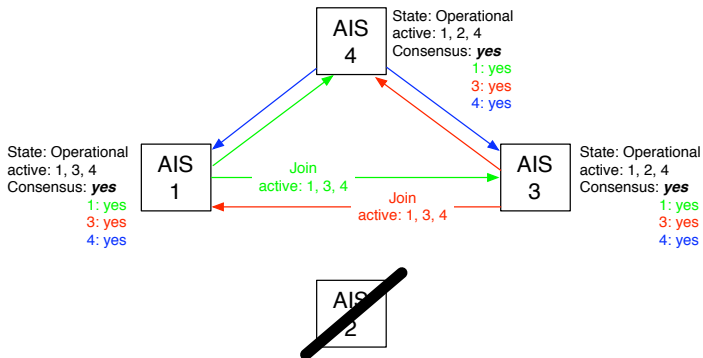
WS RRP

Membership Protocol



WS RRP

Membership Protocol



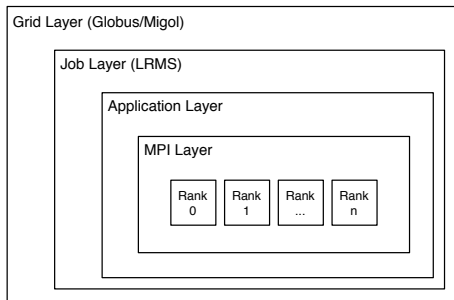
Outline

- 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



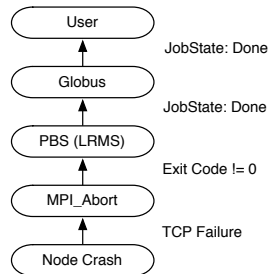
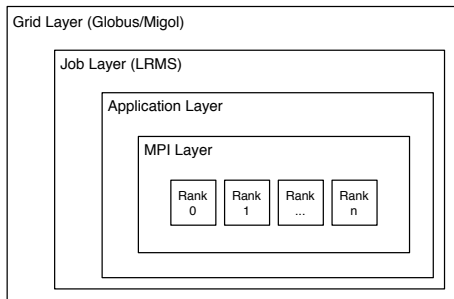
Fault Tolerance of Applications

Failure Propagation



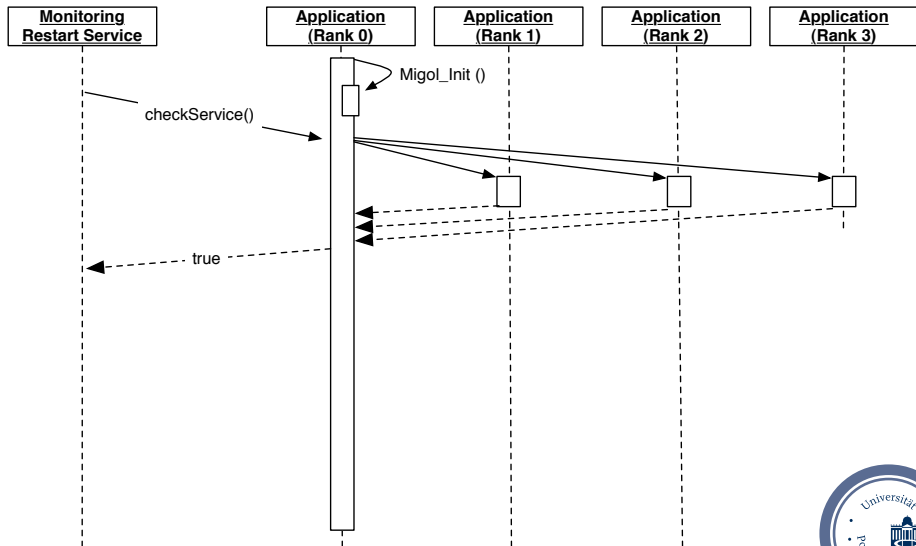
Fault Tolerance of Applications

Failure Propagation



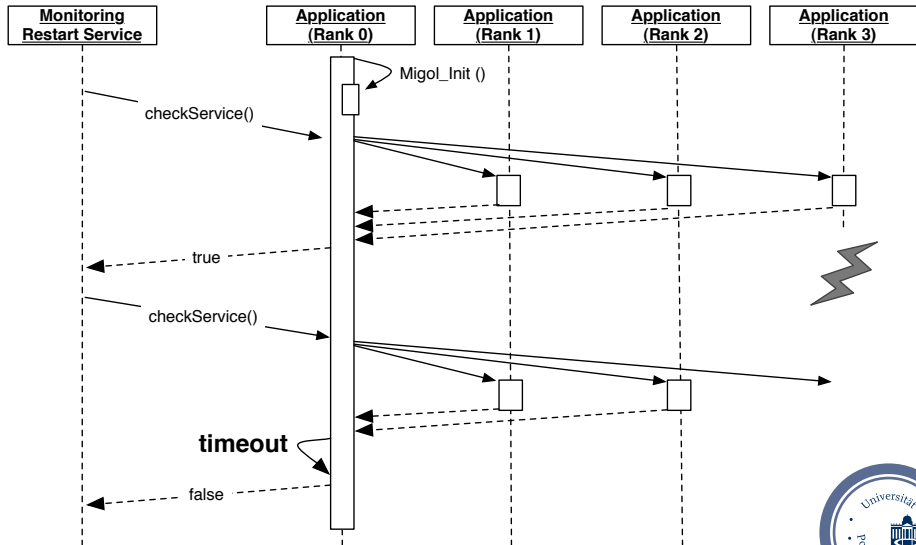
Fault Tolerance of Applications

Failure Detection



Fault Tolerance of Applications

Failure Detection



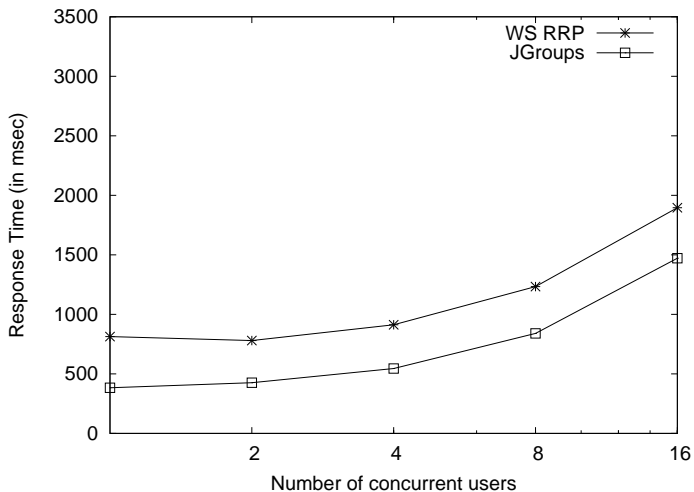
Outline

- 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)

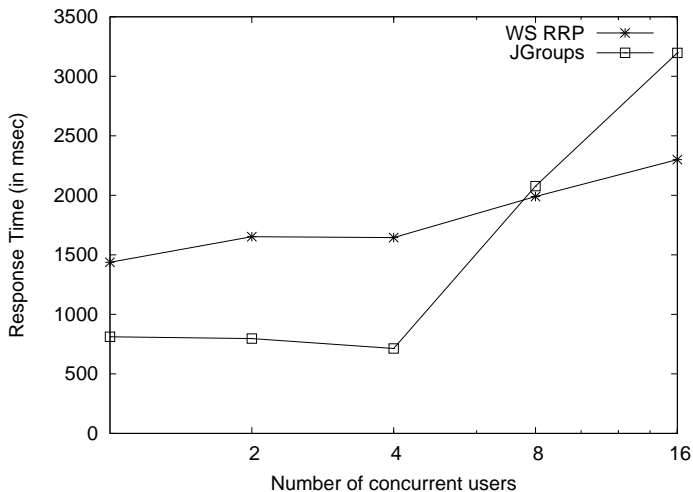


Testbed: Einstein Cluster (4 sites)



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:

Runtime	53 days 23 h
Number of job failures	83
Number of AIS recon- figurations	4
Mean Time To Failure	14 h 26 min
Mean Time To Repair	4 min 11 s (automatic) 1 h 7 min (w/ manual)
Manual Recoveries	3



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:

Runtime	53 days 23 h
Number of job failures	83
Number of AIS recon- figurations	4
Mean Time To Failure	14 h 26 min
Mean Time To Repair	4 min 11 s (automatic) 1 h 7 min (w/ manual)
Manual Recoveries	3



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.
- Publications in different conferences proceedings and two journals: EuroPVM 2005, PDCS 2006, GES2007, FGCS 2008, IEEE NCA 2008, IEEE e-Science 2008, Phil Trans A 2009 (further references see dissertation).
- OGF SAGA CPR Draft, 2008.



Thank You!

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



Publications I

-  M. Gusowski, A. Luckow, B. Schnor, and M. Schütte.
Experiences with a Resilient, MPI-based Master-Worker Application in a Failure-Prone Grid Environment.
Technical report, University of Potsdam, Potsdam, 2008.
-  N. Hallama, A. Luckow, and B. Schnor.
Grid Security for Fault Tolerant Grid Applications.
In *ISCA 19th International Conference on Parallel and Distributed Computing Systems*, pages 76–83, San Francisco, USA, 2006.
-  J. Jeske, A. Luckow, and B. Schnor.
Reservation-based Resource-Brokering for Grid Computing.
In *Proceedings of German e-Science Conference*, Baden-Baden, Germany, 2007.



Publications II



A. Luckow.

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

In Proceedings of 22nd PARS Workshop, 2009.



A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor.

Distributed Replica-Exchange Simulations on Production Environments using SAGA and Migol.

In 4th IEEE International Conference on e-Science, Indianapolis, IN, USA, 2008.



A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor.

Adaptive Replica-Exchange Simulations.

Royal Society Philosophical Transactions A, 2009.



Publications III



A. Luckow and B. Schnor.

Migol: A Fault Tolerant Service Framework for Grid Computing – Evolution to WSRF.

In *Proceedings 2. Workshop: Grid-Technologie für den Entwurf technischer Systeme*, pages 47–54, Dresden, 2006.



A. Luckow and B. Schnor.

Adaptive Checkpoint Replication for Supporting the Fault Tolerance of Applications in the Grid.

In *7th IEEE International Symposium on Network Computing and Applications*, Boston, USA, 2008.



Publications IV



A. Luckow and B. Schnor.

Migol: A Fault-Tolerant Service Framework for MPI Applications in the Grid.

Future Generation Computer Systems – The International Journal of Grid Computing: Theory, Methods and Application, 24(2):142–152, 2008.



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.

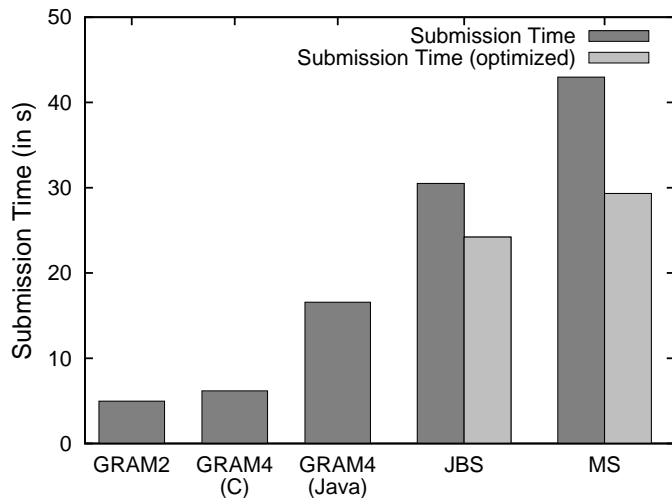


Backup



Grid Experiments

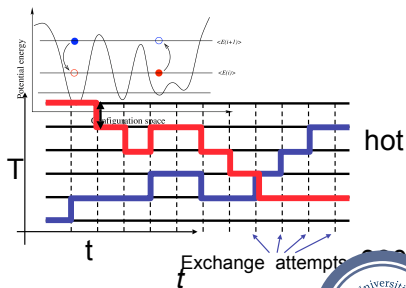
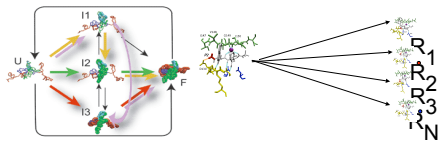
Job Submission and Migration @ UP Grid



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.



Grid Experiments

Replica-Exchange @ TeraGrid

