

Update on EZ-Grid

Priya Raghunath
University of Houston

PI : Dr Barbara Chapman
chapman@cs.uh.edu



Outline

- Campus Grid at the University of Houston (UH)
- Functionality of EZ-Grid
- Design and Implementation of EZ-Grid
- Integrated Resource Information Service (IRIS)
- Adaptive Fault-tolerance
- Conclusion





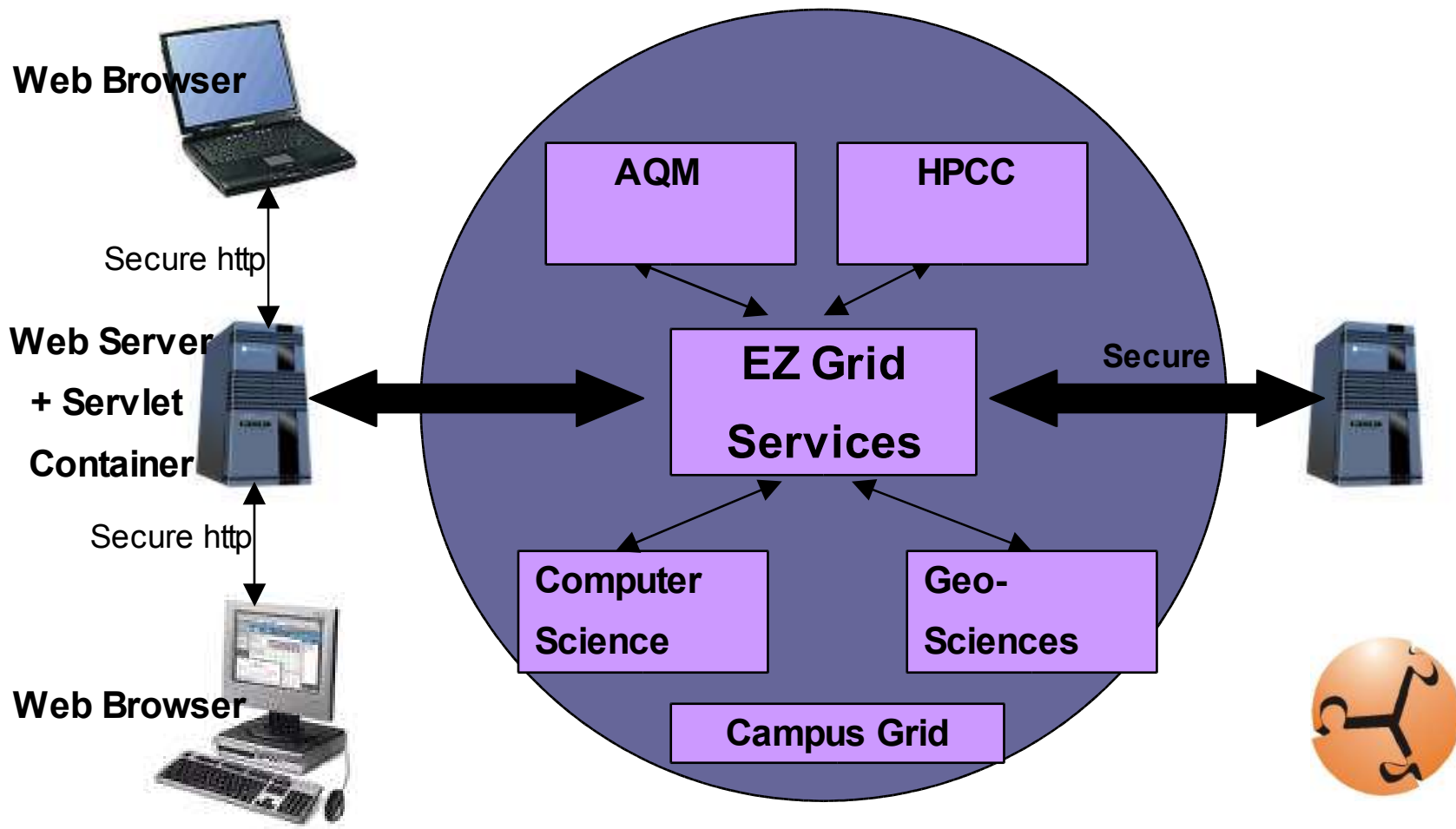
Campus Grid Setup

- Major resource for computational science is Sun cluster at HPC Center (HPCC)
 - Large-scale computing, data intensive computing
 - Shared file system
 - Job submission is managed by SGE
- Other resources include Solaris clusters and Linux clusters
 - Separately constructed,
 - Individually owned by various departments such as Computer Science, Geosciences and Math





Campus Grid





EZ-Grid

- High-level environment for job submission and resource usage control
- Uses Globus tools for middleware
 - Security, resource management, information services, data transport etc.
- EZ-Grid Goals
 - Developing generic brokerage systems for multi-site computing
 - Enforcing absolute control and priority for resource providers
 - GUI for primary functions such as Job Submission, Job Management and file handling
 - Customizable User Profiles





Previous Architecture of EZ-Grid

- Client-Server Architecture
- EZ-Grid Client
 - Provides functionalities such as:
 - Logging into the grid (create grid proxy), GUI, Brokering tools to assist in scheduling of user jobs
- EZ-Grid Server
 - Provides functionalities such as :
 - Management of remote resources
 - Usage Control of the resources by enforcing owners policies
 - Job Submission





Limitations of Client-Server EZ-Grid Model

- Difficult to deploy
 - Needs to be installed on each of the resource from which user needs to access.
- Lack of modularization
 - Client side of the client-server system includes both the GUI and the business logic
- Hard to maintain
 - Client software maintenance becomes harder with distributed clients



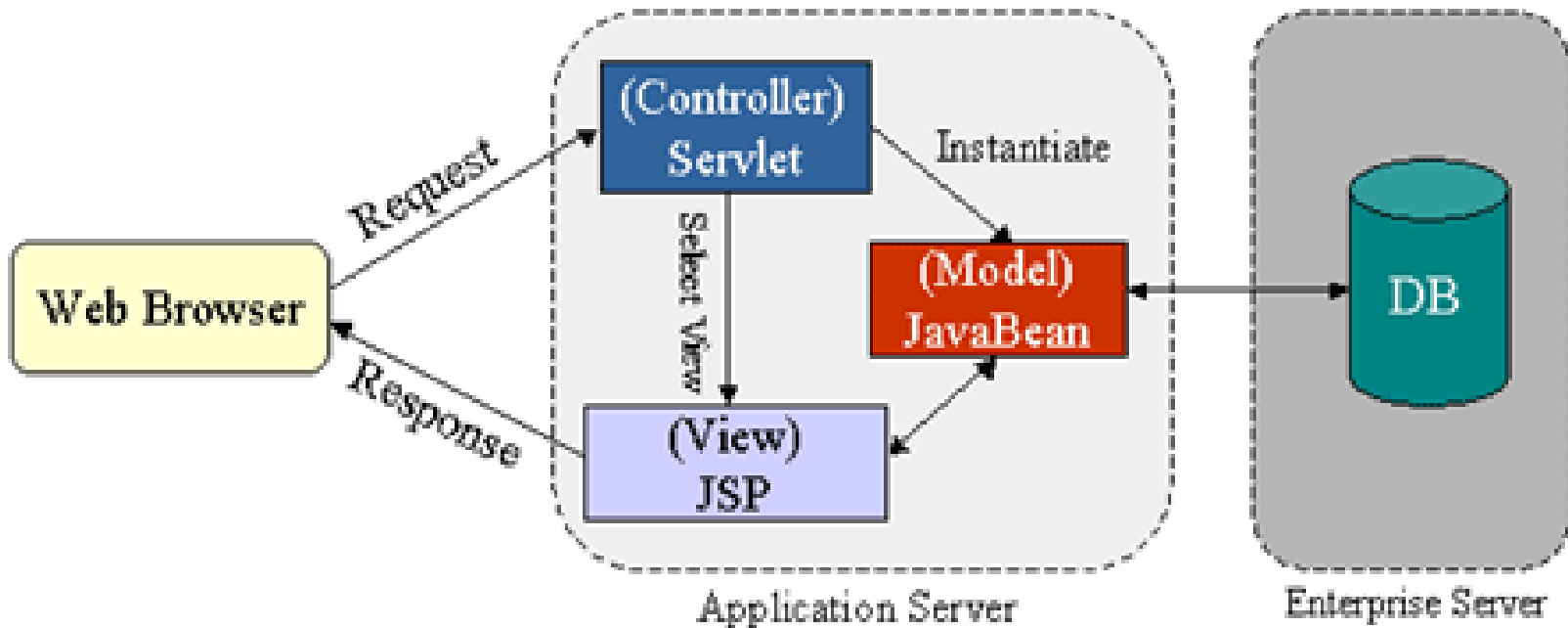


New Architecture of EZ-Grid

- Highly Modular
 - Presentation layer and business logic are separated
- Enhanced Web-based portal
- Convenient, ubiquitous and high-level access to the grid
- Model-View-Controller (MVC) as the design pattern
 - Separate the application logic (model) from the way it is represented to the user (view)
 - Matches user actions with appropriate model
 - Uses Java Server Technology: JSP + Servlets + Struts web framework
- Struts is a framework based on JSP Model 2

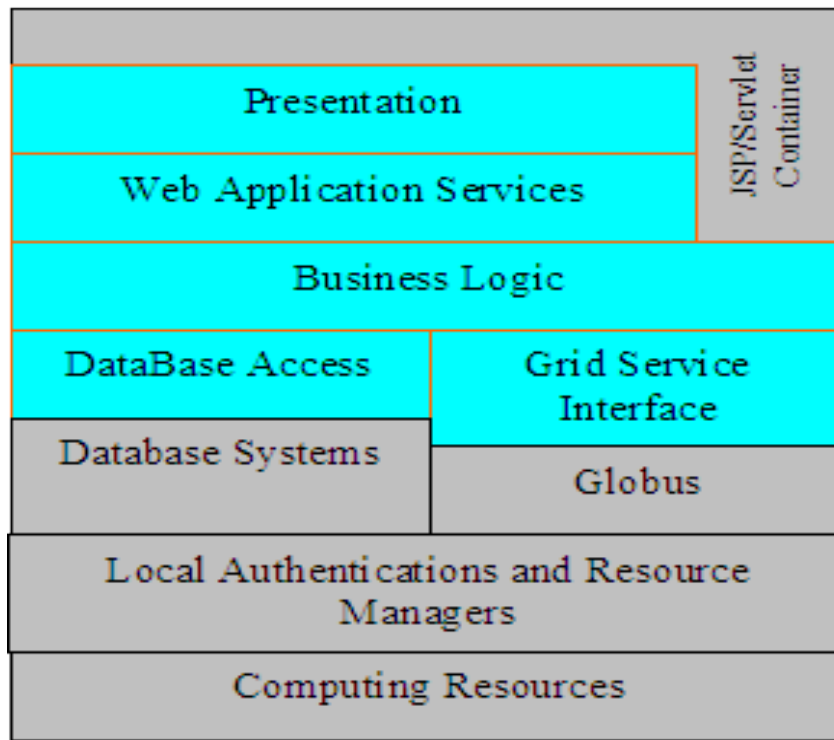



JSP Model 2 Design (MVC)





EZ-Grid Architecture



 EZ-Grid portal Components





EZ-Grid Design

- Presentation layer
 - HTML/JSP web pages containing customizable GUI to provide users with convenient and useful graphical interfaces
- Web application service layer
 - Uses struts framework to organize/model the generic web application services in the standard MVC paradigm
- Business Logic
 - The code is encapsulated as Java beans components
- Grid Service Interface Layer
- Database Layer





Driving Application-Air Quality Modeling

Air quality initiative with federal funds

- Work to model ozone problems in Greater Houston area
- Assess strategies for compliance with ozone standards by 2007
- Challenges
 - Time critical results
 - Different components execute on heterogeneous environments
 - Interaction between components
 - Reliable execution



Development Stages of EZ-Grid


- Stage 1
 - Basic functionalities such as authentication, maintenance of user profile and job profiles, file handling and job executions
- Stage 2
 - Maintenance of Data archives, resource information (static and dynamic), resource brokerage and precise usage control
- Stage 3
 - Fault tolerance and project management functions





EZ-Grid User Interface

Address: http://grid.hpctools.uh.edu:8080/aqmg/addJobProfile.jsp



[Home](#) [User Info](#) [Log Off](#)

Add New Job Profile

Job Name

Executable

Output Files

Machine	File	
<input type="text" value="kodos.hpcc.uh.edu"/>	<input type="text" value="MMOUT_DOMAIN1"/>	<input type="button" value="Add >>"/>
		<input type="button" value="<< Remove"/>

File_List

file:///kodos.hpcc.uh.edu/MMOUT_DOMAIN3

file:///kodos.hpcc.uh.edu/MMOUT_DOMAIN2

Input Files

Machine	File	
<input type="text"/>	<input type="text"/>	<input type="button" value="Add >>"/>
		<input type="button" value="<< Remove"/>


File_List

file:///kodos.hpcc.uh.edu/MMINPUT_DOMAIN1

file:///kodos.hpcc.uh.edu/MMINPUT_DOMAIN2

file:///kodos.hpcc.uh.edu/MMINPUT_DOMAIN3

Category Arguments

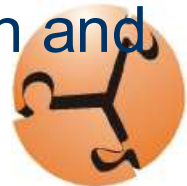


University of Houston



Integrated Resource Information Service (IRIS)

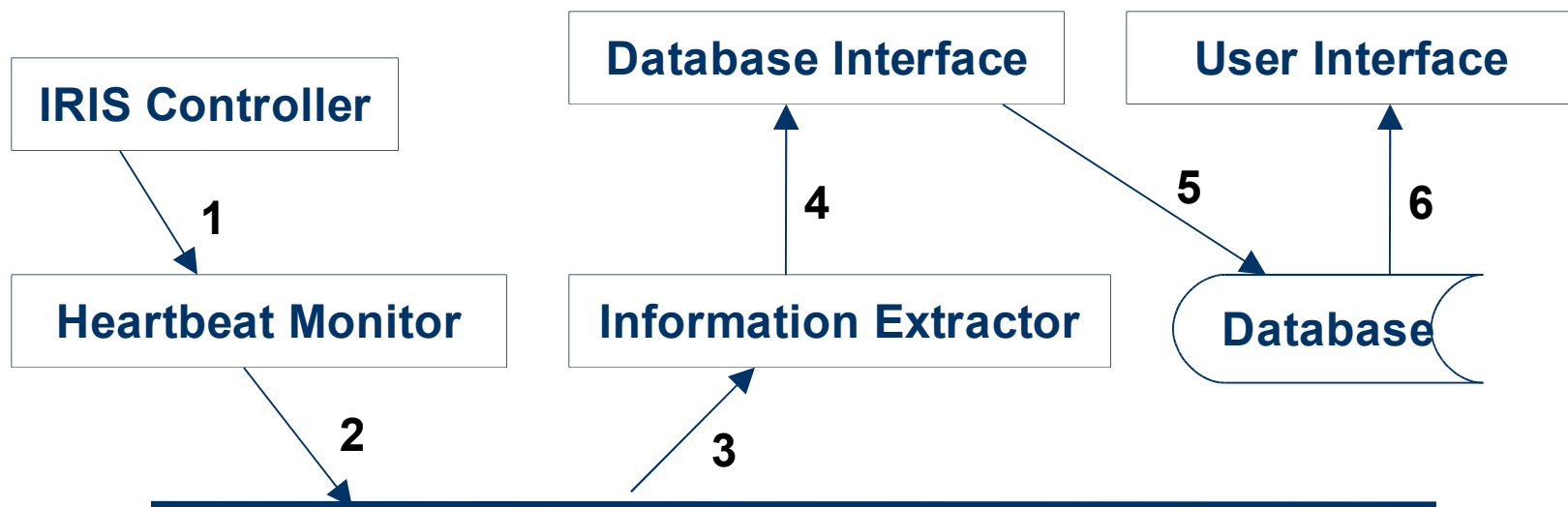
- Estimates the **queue wait time** a user could experience based on the number of nodes desired
- Aids the user in node selection
- Information updated in real time
- Profiles every user based on past usage. Helps users whose jobs have consistent **execution times**
- **Job Turnaround Time:** Queue Wait Time + Execution Time
- The IRIS backend database is used for storing the relevant information. Useful for grid administration and monitoring.





IRIS Architecture

Web / Command Line Based Reporting Interface



Resource Management Systems (SGEEE etc.)

Grid Resources





IRIS Sample Output

*Select * from **USER** where Owner = 'archit';*

Owner	Num of Jobs	Avg Exec Time	Avg Wait Time	Max Exec Time	Min Exec Time	Tool
Archit	1444	16386.44	5957.49	1136383	11	IRIS

Select Day, count() from **JOB** group by Day;*

Day	count(*)
Fri	1746
Mon	1768
Sat	926
Sun	891
Thu	1674
Tue	1362
Wed	1714

*Select Wait, Date, Time from **PROBE**
where Num_Nodes=6;*

3157 , Fri 28 Jun 2003, 18:54:57





Current Techniques for providing Fault Tolerance

- Checkpointing by saving the entire application image consumes time and disk space thereby affecting performance
- Some solutions have benefit of saving only the minimal state needed to recover however the degree of application fault-tolerance is usually fixed
- High degree of fault-tolerance hurts performance even if resources is very stable
- More recent trends are in the area of reconfigurable applications





Proposed Solution

- Adaptive fault-tolerance balanced with performance and resource stability
- Reconfigurable execution adapted to the changing computation power, such as node addition, failure, network bandwidth change, etc



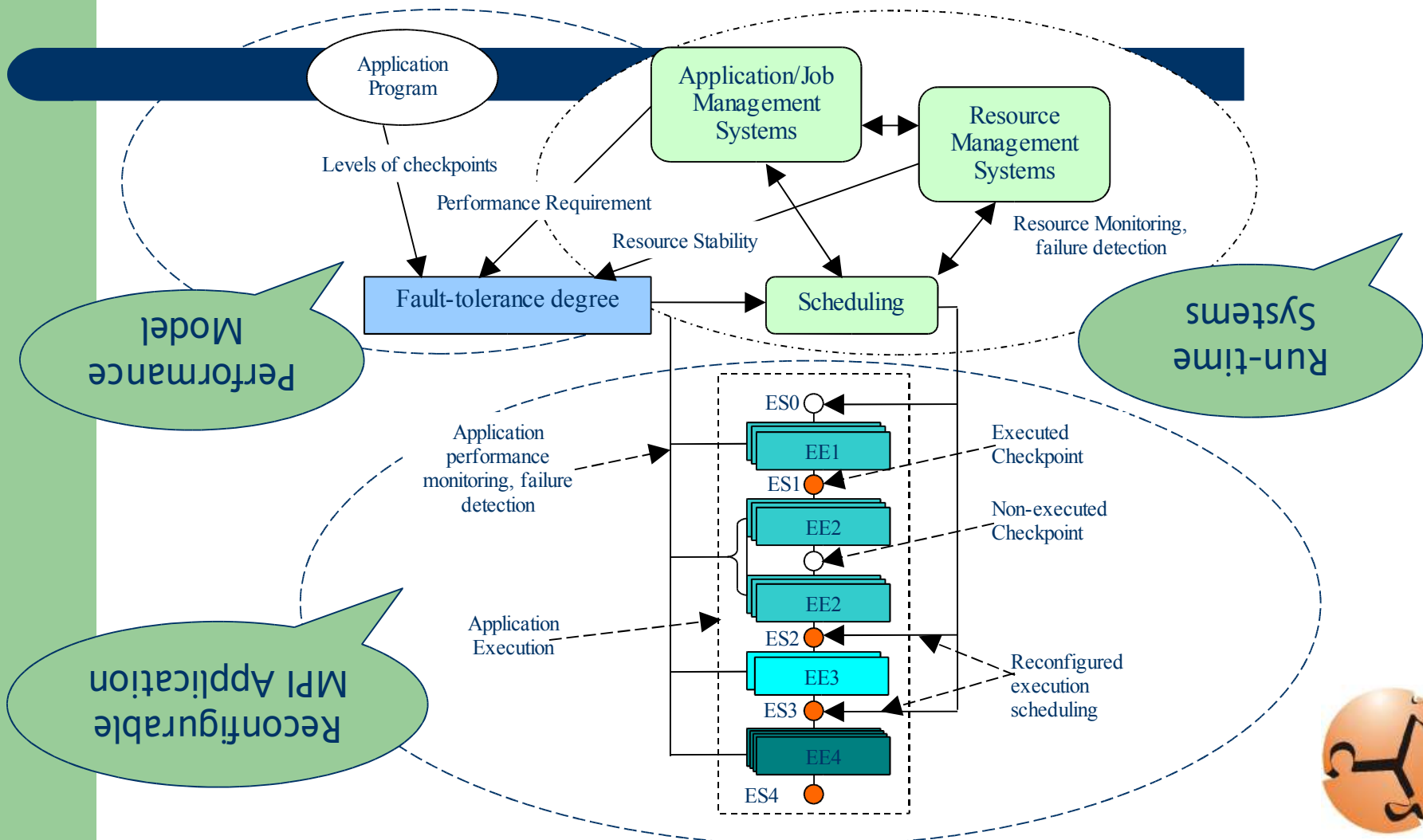


Proposed Solution

- Reconfigurable MPI with application-level checkpointing programming
- Run-time systems, providing resource and application monitoring, and
- Performance model to decide run-time balance
 - To guide programmer to setup different degrees of fault-tolerance in programs
 - To guide run-time systems to decide the best balance between performance, resource stability and fault-tolerance



Proposed Solution





Conclusion

- Grids need an convenient interface in order to allow easy access to grid services
- EZ-Grid is taking efforts in that direction
- Moving towards OGSA Compliance
- Grid Engine software provides a powerful tool for building campus grids such as the one at UH
- Investigating general framework for Policy
- Fault-tolerance without compromising on performance is an important need for grids

