Enhancing the effective utilisation of Grid clusters by exploiting on-line performability analysis

Anne Benoit, Murray Cole, Stephen Gilmore and Jane Hillston
Enhance project, LFCS, University of Edinburgh
Outline

1. Challenges of Grid performability
   - Addressing the challenges
   - Modelling and analysis

2. Performance modelling with process algebras
   - Performance Evaluation Process Algebra
   - PEPA model of jobs and servers
   - Analysis of the model

3. A failure/repair model
   - Analysis of the failure/repair model

4. Commentary and comparison

Stephen Gilmore. LFCS, University of Edinburgh. Enhance project: http://groups.inf.ed.ac.uk/enhance/
1. Challenges of Grid performability
   - Addressing the challenges
   - Modelling and analysis

2. Performance modelling with process algebras
   - Performance Evaluation Process Algebra
   - PEPA model of jobs and servers
   - Analysis of the model

3. A failure/repair model
   - Analysis of the failure/repair model

4. Commentary and comparison
Challenges of Grid performability

Grid computing is characterised by
- heterogeneity
Challenges of Grid performability

Grid computing is characterised by
- heterogeneity
- scheduling and re-scheduling
Challenges of Grid performability

Grid computing is characterised by

- heterogeneity
- scheduling and re-scheduling
- scale
Challenges of Grid performability

Grid computing is characterised by
- heterogeneity
- scheduling and re-scheduling
- scale
- failures
Grid computing is characterised by

- heterogeneity
  - provide a modelling language to describe diverse systems
- scheduling and re-scheduling
- scale
- failures
Challenges of Grid performability

Grid computing is characterised by

- heterogeneity
  - provide a modelling language to describe diverse systems
- scheduling and re-scheduling
  - provide a low-cost analysis of time-dependent behaviour
- scale

- failures
Challenges of Grid performability

Grid computing is characterised by

- **heterogeneity**
  - provide a modelling language to describe diverse systems

- **scheduling and re-scheduling**
  - provide a low-cost analysis of time-dependent behaviour

- **scale**
  - provide a parametric analysis which scales to large job sizes

- **failures**

Stephen Gilmore. LFCS, University of Edinburgh.

Enhance project: http://groups.inf.ed.ac.uk/enhance/
Grid computing is characterised by
- heterogeneity
  - provide a modelling language to describe diverse systems
- scheduling and re-scheduling
  - provide a low-cost analysis of time-dependent behaviour
- scale
  - provide a parametric analysis which scales to large job sizes
- failures
  - allow custom recovery procedures to be specified
Addressing the challenges

- Use a high-level programming model to structure code.

- Use a high-level modelling language to analyse performance.
Addressing the challenges

- Use a high-level programming model to structure code.
  - Cole’s “algorithmic skeletons” (eSkel library)
- Use a high-level modelling language to analyse performance.
Addressing the challenges

- Use a high-level programming model to structure code.
  - Cole’s “algorithmic skeletons” (eSkel library)
- Use a high-level modelling language to analyse performance.
  - Hillston’s Performance Evaluation Process Algebra (PEPA)
We describe the workload and the computing fabric as a high-level model in PEPA.
We describe the workload and the computing fabric as a high-level model in PEPA.

We map PEPA models into ordinary differential equations (ODEs) for solution.
We describe the workload and the computing fabric as a high-level model in PEPA.

We map PEPA models into ordinary differential equations (ODEs) for solution.

The analysis is supported by an automated tool which handles the transformation from the high-level process algebra model into ODEs and numerical integration.
Modelling and analysis

- We describe the workload and the computing fabric as a high-level model in PEPA.
- We map PEPA models into ordinary differential equations (ODEs) for solution.
- The analysis is supported by an automated tool which handles the transformation from the high-level process algebra model into ODEs and numerical integration.
- The results are returned to the user as a plot of the numbers of model components as a function of time.
Outline

1. Challenges of Grid performability
   - Addressing the challenges
   - Modelling and analysis

2. Performance modelling with process algebras
   - Performance Evaluation Process Algebra
   - PEPA model of jobs and servers
   - Analysis of the model

3. A failure/repair model
   - Analysis of the failure/repair model

4. Commentary and comparison

Stephen Gilmore. LFCS, University of Edinburgh.

Enhance project: http://groups.inf.ed.ac.uk/enhance/
PEPA components perform activities either independently or in co-operation with other components.
PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).
PEPA components perform activities either independently or in co-operation with other components.

The rate at which an activity is performed is quantified by some component in each co-operation. The symbol $\top$ indicates that the rate value is quantified elsewhere (not in this component).

\[(\alpha, r).P\] Prefix
\[P_1 + P_2\] Choice
\[P_1 \otimes P_2\] Co-operation
\[P/L\] Hiding
\[X\] Variable
$P_1 \parallel P_2$ is a derived form for $P_1 \otimes_\emptyset P_2$. 
Derived forms and additional syntax

\[ P_1 \parallel P_2 \text{ is a derived form for } P_1 \text{ } \Box \text{ } \emptyset P_2. \]

Because we are interested in transient behaviour we use the deadlocked process *Stop*.
$P_1 \parallel P_2$ is a derived form for $P_1 \blacklozenge_{\emptyset} P_2$.

Because we are interested in transient behaviour we use the deadlocked process $Stop$.

When working with large numbers of jobs and servers, we write $P[n]$ to denote an array of $n$ copies of $P$ executing in parallel.
$P_1 \parallel P_2$ is a derived form for $P_1 \subseteq \emptyset P_2$.

Because we are interested in transient behaviour we use the deadlocked process $Stop$.

When working with large numbers of jobs and servers, we write $P[n]$ to denote an array of $n$ copies of $P$ executing in parallel.

$$P[5] \equiv (P \parallel P \parallel P \parallel P \parallel P)$$
Consider jobs with a number of ordered stages. (Here three.)
Consider jobs with a number of ordered stages. (Here three.)

Jobs must be loaded onto a node before execution. Stage 1 must be completed before Stage 2 and Stage 2 before Stage 3. After Stage 3 the job is cleared by being unloaded from the node, and is then finished.
Consider jobs with a number of ordered stages. (Here three.)

Jobs must be loaded onto a node before execution. Stage 1 must be completed before Stage 2 and Stage 2 before Stage 3. After Stage 3 the job is cleared by being unloaded from the node, and is then finished.

Here the number of compute jobs is larger than the number of nodes available to execute them. Nodes specify the rate at which jobs are completed.
PEPA model of jobs and nodes

Jobs

\[
\begin{align*}
\text{Job} & \overset{\text{def}}{=} (\text{load}, \top).\text{Job1} \\
\text{Job1} & \overset{\text{def}}{=} (\text{stage1}, \top).\text{Job2} \\
\text{Job2} & \overset{\text{def}}{=} (\text{stage2}, \top).\text{Job3} \\
\text{Job3} & \overset{\text{def}}{=} (\text{stage3}, \top).\text{Clearing} \\
\text{Clearing} & \overset{\text{def}}{=} (\text{unload}, \top).\text{Finished} \\
\text{Finished} & \overset{\text{def}}{=} \text{Stop}
\end{align*}
\]
PEPA model of jobs and nodes

\[
\begin{align*}
\text{NodeIdle} & \overset{\text{def}}{=} (\text{load}, r_0).\text{Node1} \\
\text{Node1} & \overset{\text{def}}{=} (\text{stage1}, r_1).\text{Node2} \\
\text{Node2} & \overset{\text{def}}{=} (\text{stage2}, r_2).\text{Node3} \\
\text{Node3} & \overset{\text{def}}{=} (\text{stage3}, r_3).\text{Node4} \\
\text{Node4} & \overset{\text{def}}{=} (\text{unload}, r_0).\text{NodeIdle}
\end{align*}
\]
PEPA model of jobs and nodes

System

\[ \text{System} \]

\[ \text{NodeIdle}[100] \parallel \text{Job}[1000] \]

where \( L \) is \{load, stage1, stage2, stage3, unload\}. 

Stephen Gilmore. LFCS, University of Edinburgh.

Enhance project: http://groups.inf.ed.ac.uk/enhance/
Analysis of the model proceeds by choosing particular values for the rates. The values below are chosen to make the analysis easy to follow.

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_0$</td>
<td>1</td>
<td>(Un)loading takes one time unit</td>
</tr>
<tr>
<td>$r_1$</td>
<td>0.1</td>
<td>Stage 1 takes ten time units</td>
</tr>
<tr>
<td>$r_2$</td>
<td>0.05</td>
<td>Stage 2 takes twenty time units</td>
</tr>
<tr>
<td>$r_3$</td>
<td>0.025</td>
<td>Stage 3 takes forty time units</td>
</tr>
</tbody>
</table>
Analysis of the model: Nodes
Analysis of the model: Jobs

![Graph showing the analysis of the model: Jobs with time and value axes. The graph includes different colored lines representing various job statuses such as Clearing, Finished, Job, Job1, Job2, and Job3.](image-url)
Outline

1. Challenges of Grid performability
   - Addressing the challenges
   - Modelling and analysis

2. Performance modelling with process algebras
   - Performance Evaluation Process Algebra
   - PEPA model of jobs and servers
   - Analysis of the model

3. A failure/repair model
   - Analysis of the failure/repair model

4. Commentary and comparison
We take the modelling decision to ignore the potential failures which could occur during the very brief stages of loading and unloading jobs.
A failure/repair model

We take the modelling decision to ignore the potential failures which could occur during the very brief stages of loading and unloading jobs.

We model a failure and repair cycle taking a job back to re-execute the present stage (rather than restart the execution of the job from the beginning).
### Nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeIdle</td>
<td>$(\text{load}, r_0).\text{Node1}$</td>
</tr>
<tr>
<td>Node1</td>
<td>$(\text{stage1}, r_1).\text{Node2} + (\text{fail1}, r_4).\text{NodeFailed1}$</td>
</tr>
<tr>
<td>Node2</td>
<td>$(\text{stage2}, r_2).\text{Node3} + (\text{fail2}, r_4).\text{NodeFailed2}$</td>
</tr>
<tr>
<td>Node3</td>
<td>$(\text{stage3}, r_3).\text{Node4} + (\text{fail3}, r_4).\text{NodeFailed3}$</td>
</tr>
<tr>
<td>Node4</td>
<td>$(\text{unload}, r_0).\text{NodeIdle}$</td>
</tr>
<tr>
<td>NodeFailed1</td>
<td>$(\text{repair1}, r_5).\text{Node1}$</td>
</tr>
<tr>
<td>NodeFailed2</td>
<td>$(\text{repair2}, r_5).\text{Node2}$</td>
</tr>
<tr>
<td>NodeFailed3</td>
<td>$(\text{repair3}, r_5).\text{Node3}$</td>
</tr>
</tbody>
</table>
With regard to the rates of failure of jobs, we estimate that one in ten jobs may fail during stage 3 (and so one in 20 during stage 2 and one in 40 during stage 1) and that the cost of repairs is relatively high, perhaps requiring a reboot of the failed node.

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_4$</td>
<td>0.0025</td>
<td>On average 1 in 10 stage 3 jobs will fail</td>
</tr>
<tr>
<td>$r_5$</td>
<td>0.0025</td>
<td>Repairing may require the reboot of a node</td>
</tr>
</tbody>
</table>
Analysis of the failure/repair model: Nodes
Analysis of the failure/repair model: Jobs
Outline

1. Challenges of Grid performability
   - Addressing the challenges
   - Modelling and analysis

2. Performance modelling with process algebras
   - Performance Evaluation Process Algebra
   - PEPA model of jobs and servers
   - Analysis of the model

3. A failure/repair model
   - Analysis of the failure/repair model

4. Commentary and comparison
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit **steady-state** and **transient** analysis (by solving the CTMC).
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit steady-state and transient analysis (by solving the CTMC).

- Steady-state is cheaper but less informative. Transient is more informative but more expensive.
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit steady-state and transient analysis (by solving the CTMC).

Steady-state is cheaper but less informative. Transient is more informative but more expensive.

Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.
Previous performance modelling with PEPA used continuous-time Markov chains (CTMCs). These admit steady-state and transient analysis (by solving the CTMC).

Steady-state is cheaper but less informative. Transient is more informative but more expensive.

Major drawback: state-space explosion. Generating the state-space is slow. Solving the CTMC is slow.

In practice effective only to systems of size $10^6$ states, even when using clever storage representations.
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Major benefit: avoids state-space generation entirely.
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Major benefit: avoids state-space generation entirely.

Major benefit: ODE solving is effective in practice. (Suitable for on-line scheduling?)
Mapping PEPA to ODEs admits *course-of-values* analysis by solving the ODE (akin to transient analysis).

Major benefit: avoids state-space generation entirely.

Major benefit: ODE solving is effective in practice. (Suitable for on-line scheduling?)

Effective for systems of size $2^{10^6}$ states and beyond.
Commentary and comparison

- Analysis capabilities
  - Numerical integration, course-of-values analysis
  - Verification at process algebra level (freedom from deadlock)
Commentary and comparison

- **Analysis capabilities**
  - Numerical integration, course-of-values analysis
  - Verification at process algebra level (freedom from deadlock)

- **High-level abstract models**
  - Efficient solution of scalable models
  - Not yet applied at the application level
The *Enhancing the Performance Predictability of Grid Applications with Patterns and Process Algebras* (Enhance) project is funded by the Engineering and Physical Sciences Research council grant number GR/S21717/01.

http://groups.inf.ed.ac.uk/enhance/