Porting applications to the Grid using the EDGeS Application Development Methodology

Gabor Terstyanszky & Tamas Kiss
University of Westminster
EDGeS Application Support Service
Ad Emmen, AlmereGrid
EDGeS Dissemination & Training
Agenda

1. The EDGeS Application Support Service

2. The EDGeS Application Development Methodology (EADM)

3. Porting applications to BOINC Desktop Grids

4. Application case studies
   - Digital Alias-free Signal Processing (DASP)
   - Protein molecule simulations
   - Profiling hospitals based on patient re-admission
   - Video stream analysis
The EDGeS Application Support Service

- **Aims and objectives:**
  - facilitate the porting of applications to the EDGeS infrastructure by targeting
    - user communities already associated with EGEE or DGs
    - new user communities

- **Activities:**
  - identify user communities that require extra resources of the EDGeS infrastructure
  - develop a generic methodology for application porting
  - provide a service in order to aid the migration to and running of applications on the EDGeS infrastructure
The EDGeS User Community

• **Academic user community:**
  - 14 identified communities
    • Bioscience, healthcare, fusion energy, computational chemistry, etc.
    • 4 new academic user partners
  - Establishing the EDGeS User Forum

• **Industrial user community**
  - 5 identified companies
  - 3 new industrial user partners
  - Establishing the EDGeS Industry Forum
The EDGeS User Forum supports academic researchers who have applications that run on a Desktop or a Service Grid, or could run on a combined Desktop & Service Grid (EDGeS infrastructure).

The User Forum provides:

- Two meetings a year to meet face-to-face with the Grid application developers and other Grid application users
- Directory of EDGeS infrastructure resources
- Regular news feed on EDGeS infratsructure developments
- Guidance how to port applications on the Grid.
- Grid application development environment
- Potential access to the EDGeS infrastructure (European wide Grid) with large number of computer nodes
The EDGeS Industry Forum supports application users from companies and non-academic organisations that are interested in setting up a local Desktop Grid and getting applications to run on this local Grid or on the EDGeS infrastructure.

The Industry Forum provides:

- Two meetings a year to meet face-to-face with the Grid application developers and other Grid application users
- Directory of EDGeS infrastructure resources
- Regular news feed on EDGeS infrastructure developments
- Guidance how to port applications on the Grid.
- Grid application development environment
Agenda

1. The EDGeS Application Support Service
2. The EDGeS Application Development Methodology (EADM)
3. Porting applications to BOINC Desktop Grids
4. Application case studies
   - Digital Alias-free Signal Processing
   - Protein molecule simulations
   - Profiling hospitals based on patient re-admission
   - Video stream analysis
Why do we need a methodology?

**Motivations**

- Grid application development is very often ad-hoc
  - Developers do not follow any methodology
  - Poorly documented systems
  - User expectations not fulfilled
- Software design and development methodologies are too generic
  - Special focus is required when porting/developing an application to the EDGeS infrastructure

**EADM: EDGeS Application Development Methodology**
• Aims and objectives:
  – develop a generic methodology that addresses the problem of application porting and defines how the recommended software tools, developed by EDGeS, can aid this process.

• EADM – an iterative approach
  – EADM identifies well-defined phases that have a well-defined logical order. However, the overall process in most cases is non-linear allowing revisiting and revising the results of previous phases at any point.
EADM
Roles and Actors

- **Application developers** designed and implemented the application
- **Application administrators** are responsible for the installation and management of the application
- **Grid operators** operate the Grid infrastructure on which the ported applications are running
- **EDGeS systems analysts** capture user requirements and making conceptual design specification
- **EDGeS application programmers** are responsible for the implementation of the application migration
- **Application users (researchers):** run the application in their scientific research or industrial/business conduct
EADM
Actors and Porting Scenarios

application

has been ported to Grid

not ported to Grid

to be ported to desktop Grid

to be ported service Grid

? application developers ?

EDGeS system analysists
EDGeS application programmers

application administrators
Grid operators

application users
EADM – EDGeS Application Development Methodology

1. Analysis of current application
2. Requirements analysis
3. System design
4. Technical design
5. Implementation
6. Testing
7. Validation
8. Deployment
9. User support, maintenance & feedback
**EADM Stages – an overview**

*Earlier stages can be revisited at any time*

<table>
<thead>
<tr>
<th>Participants</th>
<th>EADM stages</th>
<th>Outcome</th>
</tr>
</thead>
</table>
| Application developers  
EDGeS systems analysts,  
Application users | 1. Analysis of current application | Application Description Template (ADT) |
| Application developers  
EDGeS systems analysts,  
Application users | 2. Requirements analysis | User Requirement Specification (URS) |
| Application developers  
EDGeS systems analysts,  
EDGeS application programmers,  
Grid operators | 3. System design | Systems Design Specification (SDS) |
EADM Stages – an overview

Earlier stages can be revisited at any time

<table>
<thead>
<tr>
<th>Participants</th>
<th>EADM stages</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGeS application programmers, EDGeS systems analysts</td>
<td>4. Technical design</td>
<td>Technical Design Specification (TDS)</td>
</tr>
<tr>
<td>EDGeS application programmers, EDGeS systems analysts</td>
<td>5. Implementation</td>
<td>Incremental system prototypes</td>
</tr>
<tr>
<td>Independent testers, EDGeS application programmers, Application users</td>
<td>6. Testing</td>
<td>Consensus between developers and users</td>
</tr>
</tbody>
</table>

Earlier stages can be revisited at any time.
EADM Stages – an overview
Earlier stages can be revisited at any time

Participants

EDGeS validation team
EDGeS Grid operators
EDGeS systems analysts
Application users

EADM stages

7. Validation
8. Deployment
9. User support, maintenance and feedback

Outcome

Validated application
Application deployed in EDGeS application repository
Feedback, support and system modification

Earlier stages can be revisited at any time
Stage 1: Analysis of current application

**Aims and objectives:**
- describe the currently existing application in detail

**Participants:**
- Application developers, application administrators, EDGeS systems analysts, application users

**Questions to be answered (original application):**
- Problem domain
- Target user community
- Computing platform currently utilised
- Type of parallelism (if any)
- Current data access
- Current functionalities
- Other factors (licensing issues, programming language, operating system, memory usage, security, ethical or gender issues)
Stage 2: Requirements analysis

**Aims and objectives:**
- identify how the target user community will benefit from porting the application to the EDGeS platform

**Participants:**
- Application developers, application administrators, **EDGeS systems analysts, application users**

**Questions to be answered (ported application):**
- User requirements
- Desired functionalities
- Desired efficiency of execution
- Desired efficiency of data access
- Target computing platform
- Required user interface
- Other factors (licensing, security, ethical issues etc.)
Stage 3: System design

Aims and objectives:
- Design the ported application at conceptual level taking user requirements and technical feasibility into consideration

Participants:
- Application developers
- EDGeS systems analysts & EDGES application programmers,
- Application users
- EDGeS Grid operators

Questions to be answered:
- Identify target computing platform to be used as entry point - changes in application required?
- Identify/design user interface
- Define and design parallelisation principles
- Design data access and transfer scenarios
- Other factors
Stage 4: Technical design

**Aims and objectives:**
- prepare a detailed technical design specification that forms the basis of implementing the required changes when porting the application to the EDGeS platform

**Participants:**
- EDGeS systems analysts, EDGeS application programmers

**Questions to be answered:**
- *How the required changes will be implemented to utilise the target computing platforms*
- *How the user interface will be implemented*
- *How to implement the required parallelism*
- *How to implement the data access and transfer scenarios*
Stage 5: **Implementation**: implement the required changes in the application based on the detailed design specification

**EDGeS development infrastructure**

Stage 6: **Testing**: both functional and performance tests

**EDGeS test infrastructure**

Stage 7: **Validation**: every application is validated to assure they cause no harm to the underlying DG infrastructure

**EDGeS test infrastructure**

Stage 8: **Deployment**: validated applications are published in the EDGeS application repository where users can find and run the requested applications with the desired parameter values

**EDGeS production infrastructure**

Stage 9: **User support, maintenance and feedback**: full user support is provided by the EDGeS Application Support Service
Agenda

1. The EDGeS Application Support Service
2. The EDGeS Application Development Methodology (EADM)
3. Porting applications to BOINC Desktop Grids
4. Application case studies
   – Digital Alias-free Signal Processing
   – Protein molecule simulations
   – Profiling hospitals based on patient re-admission
   – Video stream analysis
Porting applications to a BOINC DG platform

What we have...
Sequential application

What we want...
Parallel application

Inputs

Master application

Outputs

Server

Worker nodes
How does a BOINC application work?

- Based on the master-worker concept
- The sequential application is divided into two parts
  - Master application
  - Worker application
- The master imitates the running of a single-threaded application for the user
  - The application behaves similarly to the sequential version from the user’s point of view
- The worker applications work on independent subtasks (work units) that can be done in parallel by different worker nodes
BOINC master and client sides

Master side

1. The user starts the master
2. Reads or creates inputs
3. Generates and submits work units
4. Waits for results
5. Validates the results
6. Assimilates the results
7. Returns the outputs

Client side

1. The BOINC client fetches work units from the server
2. The client downloads the inputs and the executables
3. The BOINC client starts the worker application
4. The worker application runs
5. The client uploads the results
What needs to be implemented?

**Master side**
- The user starts the master
  - Reads or creates inputs
  - Generates and submits work units
  - Waits for results
  - Validates the results
  - Assimilates the results
  - Returns the outputs

**Client side**
- The BOINC client fetches work units from the server
  - The client downloads the inputs and the executables
  - The BOINC client starts the worker application
  - The worker application runs
  - The client uploads the results

Needs implementation
Done by the user
Done automatically
Developing a BOINC application

1. Develop client application
2. Develop master application
3. Deploy the application on a BOINC server
4. Test the application – functionality and performance
5. Register the application to the EDGeS DG->SG bridge
6. Run the application

DC-API simplifies development
Developing a BOINC application

- **application**
  - **source code available**
    - DC-API
  - **only binary available**
    - GenWrapper

- **WS-PGRADE portal**

- **BOINC enabling**

- **GUI**

*Gabor Terstyanszky & Tamas Kiss
OGF 25, 02-06 March 2009*
Porting applications to a BOINC DG platform

DC-API: Distributed Computing Application Programming Interface
• Supports developing/adapting applications for BOINC-based desktop grids
• Allows easy implementation and deployment of distributed applications on multiple grid environments
• Hides the BOINC details and it’s simple and easy to use
• Supports a master-worker programming model
• Work units are sequential applications

XtremWeb API:
• No modification or recompilation of the application is required (but no assumption should be made on the host environment)
• **XtremWeb client API**: to implement a client for submitting and controlling jobs as well as retrieving jobs results
**DC-API Functionality**

**Grid System**

- **Validator**
  - BOINC API
- **Master Application**
  - DC-API
- **Client Application (DC-API)**
- **Client Application (DC-API)**

**Common features:**
- Configuration, logging, messaging
- Initialisation
- Input and output file identification
- Event processing (e.g. checkpointing
- Progress reporting
- Work unit deletion
- Work unit generation
- Work unit submission
- Event processing (e.g. results)
- Result assimilation

**Initialisation**

**Input and output file handling**

**Work unit generation**

**Work unit submission**

**Event processing (e.g. results)**

**Result assimilation**

**Work unit deletion**

**BOINC API**

**DC-API**
**GenWrapper technology**

**BOINC wrapper misses some functions:** patching config files on client machines, generating extra messages, managing independent jobs in a WU, etc.

**Generic wrapper**

- Based on GitBox and runs legacy applications without BOINCification
- Makes the BOINC API / DC-API available in POSIX shell scripting
- A shell interpreter is started instead of the real application that executes an application script
- The script
  - realizes BOINCification through script commands
  - may run legacy applications in any way
  - may perform any preparation on input/output files, environment, etc.
  - may do whatever you can do by a script
GenWrapper Structure

Acts as the BOINC client application

1. Downloads
BOINC core client

2. Downloads
Can do almost anything that a shell script can (e.g. can start the legacy application)

3. Extracts
Application

4. Executes
Launcher

5. Launches
GitBox

A ZIP file

3. Extracts
Profile script

4. Executes
Legacy application

5. Launches
GitBox

GenWrapper binaries

6. Gets executed by GitBox

Work unit

7. Produces
Application script

Input files

Output files

Can do almost anything that a shell script can (e.g. can start the legacy application)
Porting applications to a BOINC DG platform

WS-PGRADE portal:
- provides high level GUI to the EDGeS infrastructure
- supports the transparent exploitation of the EDGeS infrastructure at application and workflow level
- gUSE (grid User Support Environment)
  - is a grid virtualization environment, which exposes the grid as a workflow
  - enables the execution of workflows simultaneously in many grids no matter what their middleware is
- WS-PGRADE is the user interface to support
  - editing, configuring, publishing workflows (as grid applications)
- Jobs can be submitted to a BOINC DG
EDGeS Application Repository based on GEMLCA

- to store applications validated for the EDGeS infrastructure
- to browse, select, parameterise and execute validated applications through the WS-PGRADE portal
1. Developing the client application (DC-API)

- Initialisation of the DC-API
- Identification of input/output files (resolves physical/logical file names)
- Implementation of the concrete computation (one independent subtask)
- Processing incoming events (checkpointing, abort and messaging)
- Saving the state periodically (for client side checkpointing)
- Reporting fraction of the work completed
- Notifying the core client of the completion
2. Developing the master application (DC-API)

- Initialisation of the DC-API – master configuration
- Setting up callbacks (result, sub-result and message processing)
- Work unit generation and submission
- Processing events (invoke callback functions)
- Processing results (via a call-back function)
- Creating the final result (assimilation) - optional
- Validation (compares redundant results, grants credits) – not part of DC-API – use BOINC validation framework
3. Deploying the application

- Install the client
- Install the master
4. Application monitoring

Checking the progress of the computation on the website

- The BOINC project provides two web sites to check the progress:
  - Public website for everyone
  - Private website for the administrators

- Both web sites can be used to check the progress of the computation proving different level of details
### 4. Application monitoring

**Public website - server status information**

Server status

14 Oct 2008 15:46:21 UTC

#### Server status

<table>
<thead>
<tr>
<th>Program</th>
<th>Host</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-driven web pages</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>upload/download server</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>scheduler</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>feeder</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>transitioner</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>protein_trivial_validator</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>dsp_trivial_validator</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>file_deleter</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
<tr>
<td>db_cache</td>
<td>staff-pc27</td>
<td>Running</td>
</tr>
</tbody>
</table>

**Status of server components**

**Database/file status**

<table>
<thead>
<tr>
<th>State</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results ready to send</td>
<td>0</td>
</tr>
<tr>
<td>Results in progress</td>
<td>0</td>
</tr>
<tr>
<td>Workunits waiting for validation</td>
<td>0</td>
</tr>
<tr>
<td>Workunits waiting for assimilation</td>
<td>0</td>
</tr>
<tr>
<td>Workunits waiting for deletion</td>
<td>0</td>
</tr>
<tr>
<td>Results waiting for deletion</td>
<td>0</td>
</tr>
<tr>
<td>Transitioner backlog (hours)</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Running:** Program is operating normally
- **Not Running:** Program failed or ran out of work (or the project is down)
- **Disabled:** Program has been disabled by staff (for debugging/maintenance)

---

Gabor Terestyanszky & Tamas Kiss
OGF 25, 02-06 March 2009
## 4. Application monitoring

### Admin website - individual results

**EDGeS: Results**

Query: select * from result limit 11

11 records match the query. Displaying 1 to 11.

<table>
<thead>
<tr>
<th>result ID</th>
<th>WU ID</th>
<th>server state</th>
<th>outcome</th>
<th>client state</th>
<th>validate state</th>
<th>delete state</th>
<th>exit status</th>
<th>host (user)</th>
<th>app ver</th>
<th>received or deadline or created</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>6</td>
<td>In Progress [4]</td>
<td>Init [0]</td>
<td>New [0]</td>
<td>Inital [0]</td>
<td>Inital</td>
<td>0</td>
<td>12 (Gabor Terstyanszky)</td>
<td>0</td>
<td>11 Nov 2008 15:51:12 UTC</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>In Progress [4]</td>
<td>Init [0]</td>
<td>New [0]</td>
<td>Inital [0]</td>
<td>Inital</td>
<td>0</td>
<td>12 (Gabor Terstyanszky)</td>
<td>0</td>
<td>11 Nov 2008 15:51:12 UTC</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>In Progress [4]</td>
<td>Init [0]</td>
<td>New [0]</td>
<td>Inital [0]</td>
<td>Inital</td>
<td>0</td>
<td>12 (Gabor Terstyanszky)</td>
<td>0</td>
<td>11 Nov 2008 15:51:12 UTC</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>In Progress [4]</td>
<td>Init [0]</td>
<td>New [0]</td>
<td>Inital [0]</td>
<td>Inital</td>
<td>0</td>
<td>12 (Gabor Terstyanszky)</td>
<td>0</td>
<td>11 Nov 2008 15:51:12 UTC</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>In Progress [4]</td>
<td>Init [0]</td>
<td>New [0]</td>
<td>Inital [0]</td>
<td>Inital</td>
<td>0</td>
<td>12 (Gabor Terstyanszky)</td>
<td>0</td>
<td>11 Nov 2008 15:51:12 UTC</td>
</tr>
</tbody>
</table>
4. Application monitoring

Admin website - result summary

<table>
<thead>
<tr>
<th>EDGeS: Result summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>11 results</strong></td>
</tr>
<tr>
<td><strong>'Over' results</strong></td>
</tr>
<tr>
<td><strong>'Success' results</strong></td>
</tr>
<tr>
<td><strong>'Client error' results</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inactive</td>
<td>0</td>
</tr>
<tr>
<td>Uncert</td>
<td>0</td>
</tr>
<tr>
<td>Uncert (no work seq)</td>
<td>0</td>
</tr>
<tr>
<td>In Progress</td>
<td>2</td>
</tr>
<tr>
<td>Over</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>0</td>
</tr>
<tr>
<td>Success</td>
<td>1</td>
</tr>
<tr>
<td>Couldn't send</td>
<td>0</td>
</tr>
<tr>
<td>Client error</td>
<td>0</td>
</tr>
<tr>
<td>No reply</td>
<td>0</td>
</tr>
<tr>
<td>Didn't need</td>
<td>0</td>
</tr>
<tr>
<td>Validate error</td>
<td>0</td>
</tr>
<tr>
<td>Client detached</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validate state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>0</td>
</tr>
<tr>
<td>Valid</td>
<td>1</td>
</tr>
<tr>
<td>Invalid</td>
<td>0</td>
</tr>
<tr>
<td>Skipped</td>
<td>0</td>
</tr>
<tr>
<td>Inconclusive</td>
<td>0</td>
</tr>
<tr>
<td>Too late</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Delete state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>0</td>
</tr>
<tr>
<td>Ready to delete</td>
<td>0</td>
</tr>
<tr>
<td>Deleted</td>
<td>4</td>
</tr>
<tr>
<td>Delete Error</td>
<td>0</td>
</tr>
<tr>
<td>Total files deleted</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downloading</td>
<td>0</td>
</tr>
<tr>
<td>Downloaded</td>
<td>0</td>
</tr>
<tr>
<td>Compute error</td>
<td>0</td>
</tr>
<tr>
<td>Uploading</td>
<td>0</td>
</tr>
<tr>
<td>Uploaded</td>
<td>0</td>
</tr>
<tr>
<td>Aborted</td>
<td>0</td>
</tr>
</tbody>
</table>
5. Application testing

- The parallelised application should be compared to the original sequential version
- The outputs should be identical (e.g. md5 checksum validation)
- Unit testing (individual parts of the application)
6. Application performance

- The performance of a DG is typically non-deterministic
  - Measuring the performance is not straightforward
  - Very different results in different runs

- Possible Performance metrics
  - Overall performance of the application
  - Performance of individual nodes
  - Hosts participating in the calculation
  - Speedup compared to the sequential version
6. Application performance
Overall performance of the application

On the public website
Performance of the DG in GFlops for four weeks
6. Application performance

Performance of individual nodes

On the admin website

<table>
<thead>
<tr>
<th>Host ID</th>
<th>IP address</th>
<th>Name</th>
<th>RAC</th>
<th>Total credit</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>161.74.69.155</td>
<td>staff-pc42</td>
<td>0.00</td>
<td>0.2</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
<tr>
<td>2</td>
<td>161.74.69.155</td>
<td>staff-pc42</td>
<td>0.03</td>
<td>0.3</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
<tr>
<td>3</td>
<td>161.74.69.155</td>
<td>staff-pc42</td>
<td>0.05</td>
<td>34.0</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
<tr>
<td>4</td>
<td>161.74.66.145</td>
<td>ZEN-TEST</td>
<td>0.00</td>
<td>0.0</td>
<td>Genuine Intel(R) Pentium(R) 4 CPU 2.00GHz [x86 Family 15 Model 4 Stepping 1]</td>
</tr>
<tr>
<td>5</td>
<td>161.74.66.145</td>
<td>ZEN-TEST</td>
<td>0.00</td>
<td>0.0</td>
<td>Genuine Intel(R) Pentium(R) 4 CPU 2.00GHz [x86 Family 15 Model 4 Stepping 1]</td>
</tr>
<tr>
<td>6</td>
<td>10.20.3.32</td>
<td>CITL0303 nes venax .ac.uk</td>
<td>0.03</td>
<td>32.2</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
<tr>
<td>7</td>
<td>10.20.3.30</td>
<td>CITL0304 nes venax .ac.uk</td>
<td>0.07</td>
<td>8.7</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
<tr>
<td>8</td>
<td>10.20.3.31</td>
<td>CITL0302 nes venax .ac.uk</td>
<td>0.05</td>
<td>25.7</td>
<td>Genuine Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz [x86 Family 6 Model 15 Stepping 2]</td>
</tr>
</tbody>
</table>
6. Application performance

**Speedup to the sequential version**

- Not provided by BOINC, needs to be measured separately

![Graph showing speedup vs. # of work units]
Utilising the EDGeS infrastructure

DG (BOINC) -> SG (EGEE) connection

- The EDGeS project makes it possible to run applications on EGEE through the DG->SG bridge
- In order to achieve that a DG->SG bridge should be connected to the DG server
- The bridge itself is a client of the DG server and can send work units as EGEE jobs into EGEE
Connecting the DG->SG bridge to the DG server

- Port your client applications to Linux
- Open port 80 on the firewall if necessary (local DG)
- Optionally, create a new user for the bridge (easier monitoring)
- Register the BOINC project on the bridge and provide myProxy information for authentication
Agenda

1. The EDGeS Application Support Service
2. The EDGeS Application Development Methodology (EADM)
3. Porting applications to BOINC Desktop Grids

4. Application case studies
   - Digital Alias-free Signal Processing
   - Protein molecule simulations
   - Profiling hospitals based on patient re-admission
   - Video stream analysis
Implementation environment
The University of Westminster Local DG (WLDG)

1. New Cavendish Street 576 nodes
2. Marylebone Campus 559 nodes
3. Regent Street 395 nodes
4. Wells Street 31 nodes
5. Little Tichfield Street 66 nodes
6. Harrow Campus 254 nodes
Implementation environment
The University of Westminster Local DG (WLDG)

- DG Server on private University network
- Over 1500 Windows PCs from 6 different campuses
  - All machines running windows
  - Most of the machines have dual core processors
- Running the Local SZTAKI Desktop Grid package
- BOINC clients are installed automatically and maintained by specifically developed Novell ZENworks objects
- All workers are registered under the same user account.
- The workers are only available for DG computation when they are switched on but not used by students
## Applications ported by EDGeS

<table>
<thead>
<tr>
<th>Application</th>
<th>Organisation</th>
<th>Runs on Desktop Grid</th>
<th>Runs on EGEE (EDGes VO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Stream Analysis in a Grid Environment (VISAGE)</td>
<td>Correlation Systems Ltd - Israel</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Digital Alias-free Signal Processing</td>
<td>University of Westminster</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Protein Molecule Simulation using Autodock</td>
<td>University of Westminster</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E-Marketplace Model Integrated with Logistics (EMMIL)</td>
<td>SZTAKI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Application</td>
<td>Organisation</td>
<td>Runs on Desktop Grid</td>
<td>Runs on EGEE (EDGes)</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------------------------------------------</td>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Anti-cancer Drug Design (CancerGrid)</td>
<td>SZTAKI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cellular Automata based Laser Dynamics (CALD)</td>
<td>University of Seville and University of Westminster</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Signal and Image Processing using GT4 Tray</td>
<td>Forschungs zentrum Karlsruhe</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Analysis of Genotype Data (Plink)</td>
<td>Atos Origin</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
## Applications ported by EDGeS

<table>
<thead>
<tr>
<th>Application</th>
<th>Organisation</th>
<th>Runs on Desktop Grid</th>
<th>Runs on EGEE (EDGes VO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Audio Retrieval using TRIANA (DART)</td>
<td>Cardiff University</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fusion Plasma Application (ISDEP)</td>
<td>BIFI</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3-D Video Rendering using Blender</td>
<td>University of Westminster</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Profiling Hospitals in the UK based on Patient Readmission Statistics</td>
<td>University of Westminster</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
**Users:** Centre for Systems Analysis – University of Westminster

- Traditional DSP -> Uniform sampling
  - Suffers from aliasing

**Aim:** digital alias-free signal processing (DASP)

- One solution is Periodic Non-uniform Sampling (PNS)

- The DASP application designs PNS sequences

- Selection of optimal sampling sequence is a computationally expensive process
  - A linear equation has to be solved and a large number of solutions (in the magnitude of $10^{10}$ has to be compared.

- The analysis of the solutions are independent from each other, suitable for DG parallelisation
EDGeS Ported Applications
DASP – Parallelization

Master

Solve

Find best Permutation for solution 1, 1+m, 1+2m …

$q_r, q_{r+1}, \ldots, q_{2r-1}$

Computer 1

Locally best solution

Find best Permutation for solution 2, 2+m, 2+2m …

$q_r, q_{r+1}, \ldots, q_{2r-1}$

Computer 2

Locally best solution

Find best Permutation for solution m, 2m, 3m, …

$q_r, q_{r+1}, \ldots, q_{2r-1}$

Computer m

Locally best solution

Workers

Find globally best solution

Master

Gabor Terstyanszky & Tamas Kiss
OGF 25, 02-06 March 2009
EDGeS Ported Applications
DASP – Desktop Grid Solution

Define input parameters (T, L, …)
Define number of work units

BOINC Server

DC-API
Create work units
Collect and process results

Download: executable and solution vector
Upload: result (best local J value)

BOINC workers
## EDGeS Ported Applications
### DASP - Performance test results

<table>
<thead>
<tr>
<th>Period T (factor)</th>
<th>Sequential</th>
<th>DG worst</th>
<th>DG median</th>
<th>DG best</th>
<th># of work units</th>
<th>Speedup (best case)</th>
<th># of nodes involved (median)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>13 min</td>
<td>9 min</td>
<td>7 min</td>
<td>4 min</td>
<td>50</td>
<td>1.86</td>
<td>59</td>
</tr>
<tr>
<td>20</td>
<td>2h 29min</td>
<td>111 min</td>
<td>43 min</td>
<td>20 min</td>
<td>100</td>
<td>7.45</td>
<td>97</td>
</tr>
<tr>
<td>22</td>
<td>26h 40min</td>
<td>5h 1min</td>
<td>3h 24min</td>
<td>2h 31min</td>
<td>723</td>
<td>10.6</td>
<td>179</td>
</tr>
<tr>
<td>24</td>
<td>~820 h</td>
<td>n/a</td>
<td>n/a</td>
<td>17h 54min</td>
<td>980</td>
<td>45.81</td>
<td>372</td>
</tr>
</tbody>
</table>
EDGeS Ported Applications
DASP - Performance test results

Speedup

<table>
<thead>
<tr>
<th>Factor</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

Gabor Terstyanszky & Tamas Kiss
OGF 25, 02-06 March 2009
- Develop Linux (32 bit, 64 bit) clients
  - use only statically compiled libraries
- Configure the firewall to allow connections from the bridge.
- Register the BOINC project running the DASP on the DG->SG bridge.
  - provide MyProxy information of BOINC administrator
- Create a new dedicated DG user representing the DG->SG bridge
  - to allow the easier monitoring of work units executed by the bridge
EDGeS Ported Applications
DASP - Running through the DG->SG Bridge

Master: Define input parameters (T, L,...) Define number of work units

BOINC to EGEE bridge

Add user credentials
Submits as job to EGEE

BOINC Server

Pulls work units

DC-API: Creates work units, collects and processes results

WMS

UoW Local Desktop Grid

EDGeS Ported Applications

Workers: Download: executable and solution vector
Upload: result (best local J value)

EGEE EDGeS VO

BOINC workers
EDGeS Ported Applications
DASP - Problems with the current algorithm

- **Inefficient load balancing**
  - solutions of the equation should be grouped based on the execution time required to analyse individual solutions

- **Inefficient work unit generation**
  - some of the solutions should be divided into subtasks (more work units)

Limits to the possible speed-up
User-community/application developers to consider redesigning the algorithm
EDGeS Ported Applications

Protein Molecule Simulation using AutoDock

Users:
- Department of Molecular & Applied Biosciences at UoW

Application:
- a suite of automated docking tools
- designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure

application components:
- **AutoGrid** pre-calculates grids describing the target protein
- **AutoDock** performs the docking of the ligand to a set of grids
*AutoGrid* does not require Grid resources

One run of *AutoDock* finishes in a reasonable time on a single PC

However, thousands of scenarios have to be simulated and analysed to get stable and meaningful results.

- *AutoDock* has to be run multiple times with the same input files but with random factors
- Simulations runs are independent from each other – suitable for DG
EDGeS Ported Applications

Protein Molecule Simulation - Algorithm

<table>
<thead>
<tr>
<th>Master application on DG Server</th>
<th>Worker application on node 1</th>
<th>……</th>
<th>Worker application on node n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master application Starts</td>
<td>Decompress Inputs</td>
<td></td>
<td>Decompress Inputs</td>
</tr>
<tr>
<td>AutoGrid</td>
<td>Run AutoDock</td>
<td></td>
<td>Run AutoDock</td>
</tr>
<tr>
<td>Version</td>
<td>Write log file</td>
<td></td>
<td>Write log file</td>
</tr>
<tr>
<td>Run AutoGrid</td>
<td>Send log file back</td>
<td></td>
<td>Send log file back</td>
</tr>
<tr>
<td>Compress log files</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create identical work units</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compress Inputs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal Version</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## EDGeS Ported Applications

### Protein Molecule Simulation – Performance results

<table>
<thead>
<tr>
<th># of work units</th>
<th>Sequential</th>
<th>DG worst</th>
<th>DG median</th>
<th>DG best</th>
<th>Speedup (best case)</th>
<th># of nodes involved (median)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2h 50min</td>
<td>64 min</td>
<td>62 min</td>
<td>42 min</td>
<td>4.05</td>
<td>7</td>
</tr>
<tr>
<td>100</td>
<td>~28 h 20 min</td>
<td>113 min</td>
<td>51 min</td>
<td>49 min</td>
<td>34.69</td>
<td>55</td>
</tr>
<tr>
<td>1000</td>
<td>~283 h 20 min</td>
<td>2h 50min</td>
<td>139 min</td>
<td>124 min</td>
<td>137.10</td>
<td>252</td>
</tr>
<tr>
<td>3000</td>
<td>~850 h</td>
<td>4h 28min</td>
<td>4h 28min</td>
<td>4h 28min</td>
<td>190.30</td>
<td>469</td>
</tr>
</tbody>
</table>
EDGeS Ported Applications

Protein Molecule Simulation – Performance results

Speedup

# of work units

Speedup

0

20

40

60

80

100

120

140

160

180

200

3000

2000

1000

100

10
The DG->SG bridge has already been configured to the WLDG

- every application in the WLDG runs under the same BOINC project
  - easier maintenance of the infrastructure

- running AutoDock through bridge required only the implementation and deployment of a Linux based client
  - no further configuration or development is necessary
EDGeS Ported Applications
Protein Molecule Simulation - Running through DG- > SG Bridge

End User

Define input parameters and number of work units

BOINC workers

Workers: Download executable and input files
Upload: result

BOINC Server

G-USE DG Submitter submits jobs and retrieve results via 3G Bridge

Pulls work units

BOINC to EGEE bridge

EGEE EDGeS VO

WMS

Adds user credentials

Submits as job to EGEE

Define input parameters and number of work units

End User

Submitter submits jobs and retrieve results via 3G Bridge

gUSE WS P-GRADE portal

Protein Molecule Simulation - Running through DG -> SG Bridge
EDGeS Ported Applications

Protein Molecule Simulation - Running through DG->SG Bridge

WS-PGRADE portal

UoW Local DG

EDGeS: Result summary

<table>
<thead>
<tr>
<th>102 results</th>
<th>'Over' results</th>
<th>'Success' results</th>
<th>'Client error' results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server state</td>
<td># results</td>
<td>Validate state</td>
<td># results</td>
</tr>
<tr>
<td>Inactive</td>
<td>0</td>
<td>Initial</td>
<td>2</td>
</tr>
<tr>
<td>Unsent</td>
<td>0</td>
<td>Valid</td>
<td>1</td>
</tr>
<tr>
<td>Unsent (in prog)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Progress</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Over</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outcome</td>
<td># results</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confirmed</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client error</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

File Delete state

<table>
<thead>
<tr>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
</tr>
<tr>
<td>Ready to delete</td>
</tr>
<tr>
<td>Deleted</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>Total files deleted</td>
</tr>
</tbody>
</table>

Ported Applications

Protein Molecule Simulation - Running through DG->SG Bridge
• AutoDock scales better than the DASP
• Better performance on Linux systems (due to CygWin on Windows)
  – Performance will be better through the DG->SG bridge
• AutoDock is a legacy application, source code not available - further improvement is not possible
Users:
- Health and Social Care Modeling Group at UoW

Aim:
- profiling hospitals based on patient readmission profiles

Challenge:
- The Hospital Episode Statistics dataset includes 80 million episodes in total for a seven financial year period.
- Implementing a statistical model using the complete population dataset is unfeasible.
- Hierarchical cluster analysis is performed where the data is divided into clusters and then sampled
- The samples assigned a rank value independently
- The process has to be repeated thousands of times to get statistically valid results
EDGeS Ported Applications

Patient Readmission

**Master**: defines input dataset (CSV file), number of work units and the number of outputs

**BOINC Server**
- Pulls work units
- DC-API: Creates work units, collects results, aggregates clients’ output to one file

**BOINC workers**
- Workers: download R interpreter and cluster CSV file, run sampling and R script, upload result

**UoW Local Desktop Grid**

**BOINC to EGEE bridge**
- Adds user credentials
- Submits as job to EGEE

**WMS**
- EGEE EDGeS VO

---

Gabor Terstyanszky & Tamas Kiss
OGF 25, 02-06 March 2009
Users:
  – police & traffic controllers

Aim:
  – processing video streams and identifying movements

Challenge:
  - video analysis is used to process video data based on advanced algorithms
  - video analysis is a compute intensive and may take a long time
  - run Visage, developed by Correlation Systems Ltd, on Grid in order to decrease the execution time
Visage processes Image pairs over the BOINC/EDGeS grid and paints movement in yellow.

Insert video source:

Insert Url of Server:

Set mode to EDGeS

Video options: forward, backward, pause, play..
EDGeS Ported Applications

ViSAGE - Video Stream Analysis in a Grid Environment
Correlation Systems Ltd.
ViSAGE processes image pairs over the BOINC grid and paints movement in yellow
Thank you for your attention ...

For more information please visit the EDGeS Website:
Join the EDGeS User & Industry Forum here!
http://www.edges-grid.eu/

Please contact us if you need support in porting your application to EDGeS!
Email: kisst@wmin.ac.uk