IDGF
International Desktop Grid Federation

Porting Applications to SG/DG Infrastructures

Tamas Kiss
University of Westminster, UK
Leader of the DEGISCO Application Support Service

ISGC 2011, Taipei, Taiwan
Which applications are suitable for SG/DG infrastructures?

- Applications should run on both SGs and DGs
  - SGs support a much wider scale of applications than DGs
  - We should assure that the application runs on DGs

- Requirements towards an application to be executable on DGs:
  - **Parallelization:**
    - Only *master/worker* or *parameter sweep* parallelisation
      - No MPI or internal communication between worker nodes
      - Nodes can only use the results of other nodes *through the server*
  - **Data handling:**
    - small or medium-sized (max. 100 MB per worker) inputs and outputs (especially for public DGs, could be larger in local DGs)
    - No shared data storage (in public DGs)
    - No confidential data (sent down to potentially un-trusted worker in public DG)
Which applications are suitable for SG/DG infrastructures?

- To achieve good performance
  - the execution time of individual jobs should be
    - **Minimum:**
      - over 10 minutes (otherwise the overhead caused by the DG will reduce the performance)
    - **Maximum:**
      - less than 2-3 hours (if longer, application level check-pointing is required to avoid loss of computation caused by user interventions)
      - the execution of individual jobs should take around the same amount of time (better scheduling, less load on the server)

- Operating systems
  - Depends on the DGs where the application will run
    - windows version may be required to utilise larger number of resources
SG/DG application scenarios

1. Application runs on DG and uses SG resources via the DG to SG bridge

2. Application submitted to SG and uses DG resources via the SG to DG bridge

3. Application uses both SG and DG resources via an external scheduling and job submission system
Scenario 1 – DG to SG via bridge

- User entry point is DG – using SG is completely transparent from user’s point of view
Scenario – DG to SG via bridge

Porting to DG to SG infrastructure:

1. Develop and test DG version of application
2. Develop Linux version of your client application to run on SG
3. Test client application on SG
4. Test application via the DG->SG bridge
5. Write Test report and documentation
6. Deploy application on production infrastructure
   • Send test report to EGI DG VO Admin for approval
   • Deploy your application in production
     • On a DG already connected to EGI DG VO
     • Connect your DG to EGI DG VO
Which DG applications are suitable for EGI DG VO?

- If an application is capable to run on a heterogeneous DG then it is typically capable to run in EGI too.
  - **Public DG** applications are almost certain to run on EGI resources too.
  - **Local DG** applications may require more thorough testing and some potential fine-tuning (may suppose less heterogeneous infrastructure).
- **E.g.**: may require large amount of memory available on local machines but not on every EGI site.
**Video Analysis** is a general term used to describe the use of advanced algorithms to process video data.

**ViSAGE** is a technology for processing of video streams using a GRID of computational nodes.

ViSAGE was developed by **Correlation Systems Ltd.** as EDGeS subcontractor.
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..
ViSAGE - Video Stream Analysis in a Grid Environment
By Correlation Systems Ltd. - Israel

Small local DG at Israel

DG to SG bridge

EGI DG VO
Scenario 2 – SG to DG via bridge

Desktop Grid 1

Desktop Grid n

EGI VO

EDGI Services
DG CE +
EDGI AR

WMS
and other
EGI services

SG (EGI) user

machine or portal

– User entry point is SG
– using DG is transparent from user’s point of view
Scenario 2 – SG to DG via bridge

Porting to SG to DG infrastructure (SG version available):
1. Develop and test DG version of application
2. Test application via the SG to DG bridge and write test report
3. Have your application validated
   • Validation is done by EDGI/DEGISCO Validation Team
   • Outcome: validation document
4. Publish your application in the EDGI Application Repository
   • DG Admins can download your application from the AR if they are ready to support it
   • EGI users can also find your application in the AR
   • Bridge uses the AR for checking the validity of the application at submission time
Benefits of using DGs for SG (EGI) users

• Large additional computing power can be utilized
  – Desktop Grids are easy-to-scale systems and able to collect 1-2 orders of magnitude more compute power than Service Grids
  – By interconnecting SG and DG systems SG users can transparently execute applications on any arbitrary platform involved in the new infrastructure

• As a consequence we get:
  – reduced turnover time
  – improved fault-tolerance (redundant computing)
  – higher throughput
SG to DG Application Example
VisIVO - Visualisation Interface to the Virtual Observatory

- a suite of software tools for creating customized views of 3D renderings from astrophysical data tables

- **User community**: INAF Catania (Osservatorio Astrofisico di Catania) + University of Portsmouth

- **Grid enabled version**:
  - For g-Lite based grids
    - Runs on the Cometa Consortium Grid – Catania, Messina, Palermo
  - Ported to the EDGeS platform to utilize Desktop Grid resources
  - Subcontractor in EDGI to further develop ported application
Application porting

- Desktop Grid version is deployed and running on UoW Local DG and EDGeS@home
- Application is validated and published in Production Application Repository
- Application runs from EGEE to DG (UoW Local DG and EDGeS@home)

Data distribution

- Medium sized input files (up to 100 Mbytes) are currently feasible

Work in EDGI:

- Division of input file (potentially GBytes) and better data distribution using ADICS will be investigated
- VisIVO Web portal will be connected to DG infrastructure – potential access by the general public in museums
Scenario 3 – SG/DG resources but not through EDGeS/EDGI bridges

- Using external job submission and scheduling system to submit jobs to both SG and DG resources
  - P-GRADE/WS P-GRADE portal supports this scenario by default
    - E.g.: CancerGrid application uses EGI and DG resources via the WS P-GRADE portal
  - Could be justified with specific user requirements
    - E.g.: WISDOM project uses only pull jobs on EGI that are unsuitable to be bridged to DG
  - Both solutions use EDGeS/EDGI technology: 3GBridge
Scenario 3 – DG submitter for WISDOM

- WISDOM: Meta middleware to submit pull (pilot) jobs to EGI
- The DG submitter:
  - Submit push (direct) jobs to the DG when EGI resources are overloaded
For the Developer

- **DG version of the application has to be developed**
  - Only low level APIs (e.g. BOINC API)
  - Specific expertise is required
  - Platform dependent solutions (different BOINC/XtremWeb version)

- **Application composition**
  - Creation of workflows is not supported

For the user

- **Transparency for the user**
  - User would require the same or similar user interface for DG and SG applications
  - Total lack of UI for DG systems (e.g. only BOINC Admin can run the application)

- **Only validated applications can be run on a DG**
  - Requires an application repository
  - Publish, select, download, parameterize and execute applications
Tools supporting application development and execution on SG/DG platforms

**API level support — for the developer**

**DC-API:**
- Provides a uniform interface for different Grid systems (BOINC, Condor, XtremWeb)

**GenWrapper:**
- Generic wrapper to port legacy applications to a BOINC platform without “Boincification”

**High level graphical user interface — for the developer and the user**

**WS-PGRADE portal:**
- to support the transparent exploitation of SG/DG systems at workflow level

**EDGeS Application Repository:**
- Publish, select, download, parameterise and execute validated applications
Porting applications to a DG platform

What we have...
Sequential application

What we want...
Parallel application

Inputs

Outputs

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**
- 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

Presentation title: Porting Applications to DGs
Author: Tamas Kiss
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

Presentation title: Porting Applications to DGs
Author: Tamas Kiss
Developing a DG application

- Application
  - Source code available
    - DC-API
  - Only binary available
    - GenWrapper

- WS-PGRADE portal

- EDGeS Application Repository

- BOINC enabling
- GUI

Presentation title: Porting Applications to DGs
Author: Tamas Kiss
Application porting with DC-API

- **Distributed Computing Application Programming Interface**
- Allows **easy** implementation and deployment of distributed applications on **multiple Grid** environments
  - Back-end currently available for: BOINC, Condor, XtremWeb
  - Simplifies the development process when compared to native (e.g. BOINC, XtremWeb) APIs
  - Application can run on other Grid middleware without any modification

- **Features, properties of a DC-API application:**
  - Master-worker concept
  - Work units are sequential applications
  - Support for limited messaging between the master and the clients (status, control messages)
  - No direct communication between clients
DC-API Functionality

**Grid System**

- **Validator**
  - Initialisation
  - Work unit generation
  - Input and output file handling
  - Work unit submission
  - Event processing (e.g. results)
  - Result assimilation
  - Work unit deletion

- **Master Application**
  - Configuration, logging, messaging

- **BOINC API**
  - DC-API

- **Client Application**
  - DC-API
  - Initialisation
  - Input and output file identification
  - Event processing (e.g. checkpointing)
  - Progress reporting

- **Common features:**
  - Configuration, logging, messaging
Develop client application (with 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
Develop master application (with 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*
GenWrapper technology

**Generic wrapper**
- 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)

Application

Launcher

3. Extracts

A ZIP file

Profile script

Legacy application

GenWrapper binaries

4. Executes

5. Launches GitBox

6. Gets executed by GitBox

Work unit

Application script

Input files

Output files

7. Produces

Presentation title: Porting Applications to DGs
Author: Tamas Kiss
**WS-PGRADE portal:**
- provides high level GUI to the EDGI/DEGISCO infrastructure
- Can submit workflows to various Grid middleware: g-Lite, GT2, GT4 and BOINC
- supports the transparent exploitation of the EDGI/DEGISCO infrastructure at application and workflow level

**WS-PGRADE scenarios:**
1. Connect to any DG and use the DG to SG bridge
2. Connect to SG (EGI VO) and use the SG to DG bridge to
3. Connect to DGs and EGI VOs at the same time
Scenario 1: Executing workflows on BOINC DGs and through the 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

Pulls work units

EGI DG VO

WMS

Submits as job to EGI

Adds user credentials

BOINC to EGI bridge

Submitter submits jobs and retrieve results via 3G Bridge

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

UoW Local Desktop Grid

Define input parameters and number of work units

gUSE WS P-GRADE portal
Scenario 1: Executing workflows on BOINC DGs and through the DG->SG Bridge
Scenario 2: Executing workflows through the SG->DG Bridge from P-GRADE portal

P-GRADE portal

SeeGrid

WMS +
other EGI
services

EDGeS CE machine

End User

Create and submits workflows to EGI resources

UoW Local DG

VOCE

WMS +
other EGI
services

LCG-CE

W
N
W
N
W
N
W
N

EDGeS@home
- three sided negotiation between buyers, sellers and third party logistics providers
- Aims to optimise the total cost
- Ported to EGEE as a parameter sweep application in the SEE Grid project
- Large matrix solving – matrix can be separated and solution parallelised
- Ported to EDGeS using GenWrapper
Supporting application developers and end-users
The Application Support Service

• **Aims and objectives:**
  – facilitate the porting of applications to the combined SG/DG infrastructure by targeting
    • user communities already associated with EGI or DGs
    • new user communities

• **Activities:**
  – develop a generic methodology for application porting
  – identify user communities that require the power of the SG/DG infrastructure
  – provide a service in order to aid the migration to and running of applications on the SG/DG infrastructure
    • **European users:** EDGI Application Support Service / IDGF European Chapter
    • **Non-European users:** DEGISCO Application Support Service / IDGF International Chapter
• **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 stages that have a suggested logical order. However, the overall process in most cases is non-linear allowing revisiting and revising the results of previous phases at any point.
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
- Systems design and development methodologies are too generic
  - Special focus is required when porting/developing an application to a SG/DG platform

EADM: EDGeS Application Development Methodology
1. Analysis of current application
2. Requirements analysis
3. Systems design
4. Detailed design
5. Implementation
6. Testing
7. Validation
8. Deployment
9. User support, maintenance & feedback
### SG/DG Applications on Production infrastructure (ported by EDGeS and DEGISCO)

<table>
<thead>
<tr>
<th>community</th>
<th>number of applications</th>
<th>academic</th>
<th>industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bioscience</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Healthcare</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Physics</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Audio and video processing</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Business</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Applied mathematics</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Engineering</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>31</strong></td>
<td><strong>28</strong></td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>
Thank you for your attention ...

Any questions?

Please contact us if you need support in porting your application!

Email: kisst@wmin.ac.uk

Join the International Desktop Grid Federation:

http://desktopgridfederation.eu