Automating the Deployment, Execution & Fault Survival of MPICH-G2 jobs with WebCom-G

Adarsh Patil, Padraig J. O'Dowd and John P. Morrison
Computer Science Dept., University College Cork, Ireland
Email: {adarsh, p.odowd, j.morrison}@cs.ucc.ie

Introduction

This work investigates the use of WebCom-G [JPMP04] to handle the management & scheduling of MPICH-G2 [Kar] (MPI) jobs. Users can submit their MPI applications to a WebCom-G portal via a web interface. WebCom-G will then select the machines to execute the application on, depending on the machines available to it and the number of machines requested by the user. WebCom-G automatically & dynamically constructs a RSL script with the selected machines and schedules the job for execution on these machines. Once the MPI application has finished executing, results are stored on the portal server, where the user can collect them. A main advantage of this system is fault survival, if any of the machines fail during the execution of a job, WebCom-G can automatically handle such failures. Following a machine failure, WebCom-G can create a new RSL script with the failed machines removed, incorporate new machines (if they are available) to replace the failed ones and re-launch the job without any intervention from the user. The probability of failures in a Grid environment is high, so fault survival becomes an important issue.

- Globus: [FK97] provides the basic software infrastructure to build and maintain Grids. Grids are persistent environments that enable software applications to integrate instruments, displays, computational and information resources that are managed by diverse organisations in widespread locations.

- MPICH-G2: MPICH-G2 [Kar] is a grid-enabled implementation of the MPI v1.1 standard. It uses services from the Globus Toolkit (see Fig. 1) to handle authentication, authorisation, executable staging, process creation, process monitoring, process control, communication, redirecting of standard input (& output) and remote file access. As a result a user can run MPI programs across multiple computers at different sites using the same commands that would be used on a parallel computer or cluster. MPICH-G2 allows users to couple multiple machines, potentially of different architectures, to run MPI applications. It automatically converts data in messages sent between machines of different architectures and supports multi-protocol communication by automatically selecting TCP for inter-machine messaging and (where available) vendor-supplied MPI for intra-machine messaging.

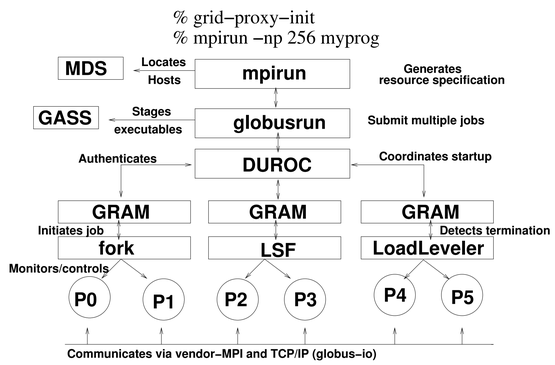

Figure 1: Overview of the startup of MPICH-G2 and the use of various Globus Toolkit components to hide and manage heterogeneity. (This diagram was taken from [Kar])

- WebCom-G: The WebCom-G Grid Operating System is a multi-layer platform for executing distributed applications on a large number of dispersed resources. Applications are separated from the underlying computing infrastructure and are specified as Condensed Graphs. Condensed Graphs are used to specify tasks and the sequencing constraints associated with them. WebCom-G supports fault tolerance/survival, load balancing, scheduling and security at different levels within its architecture. WebCom-G seeks to provide Grid access to non-specialist users and from the perspectives of application developers and end users, to hide the underlying Grid.

Managing MPICH-G2 Jobs with WebCom-G

Users can submit their MPI (MPICH-G2) jobs to a WebCom-G Portal for execution. Users can visit the WebCom-G Portal, upload their MPI code and specify the number of machines they require. WebCom-G firsts decides which machines the MPI application will execute on, once this is decided the MPI code is compiled into an executable which can be executed on these machines. If any errors occur during the compilation, they are returned to the user and the process aborted. Finally the application is scheduled for execution on the underlying resources and the following tasks are performed:

- A Resource Specification Language (RSL) script is built on the fly, depending on machine availability, path & package availability.
- The application is run using the RSL file.
- In the case of failure, the application is rescheduled with a new RSL file dynamically created by WebCom-G, from a new interrogation of the underlying resources. WebCom-G then re-launches the job with the new RSL file. No user intervention is required at this point.
- Finally results are sent to the Portal, for collection by the user.

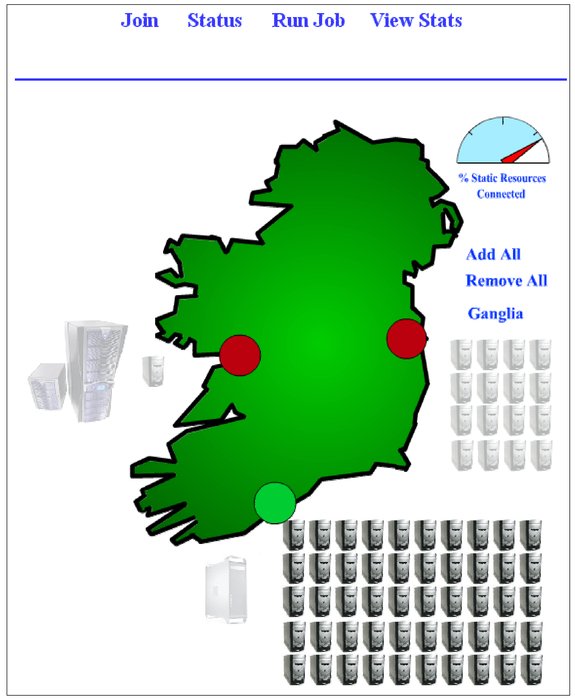

Figure 2: Graphical view of the resources connected to the WebCom-G Portal

Overview of the MPICH-G2 Condensed Graph Application

This section presents an overview of the MPICH-G2 Condensed Graph application that was developed, in order to use WebCom-G to manage MPICH-G2 jobs (see Fig. 3).

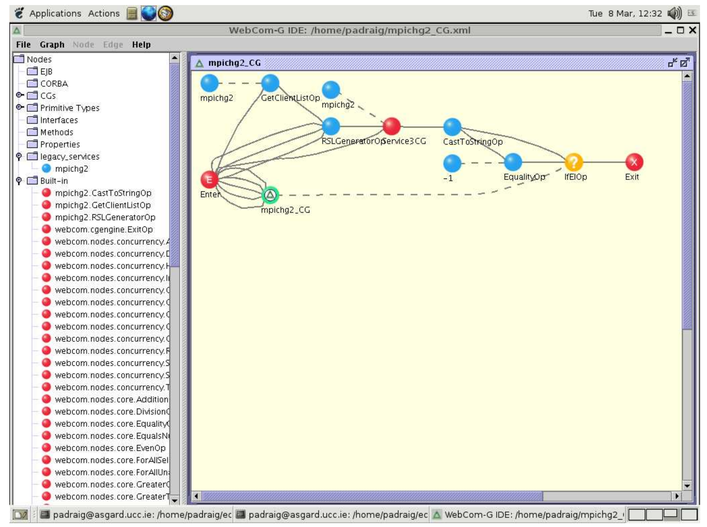

Figure 3: The MPICH-G2 Condensed Graph Application developed in the IDE.

Overview of the Main Nodes in the MPICH-G2 Graph:

- GetClientListOp: This node takes two parameters, the name of a service, which in the case of this graph is mpichg2 and the number of machines required. The node when executed outputs a list of machine names (no greater then the number of required machines). This node uses the current list of available client machines connected to the (WebCom-G) Portal and the Interrogator Database to see which of these clients can execute MPICH-G2 jobs and then creates a list of machines that can be used in the execution of the MPICH-G2 job.
- RSLGeneratorOp: This node generates an RSL file which can be used to execute the MPICH-G2 job. It takes the list of machine names generated by the GetClientListOp node, the name of the executable. the directory to run the MPICH-G2 job from and the arguments to pass to the job. From these, this node generates the RSL Script.
- Service3CG: This node executes the MPICH-G2 job with the generated RSL script and the timeout if specified by the user. This node monitors the execution of the job and in the event of a job failure/crashing, detects the job failure and reports it.
- IfElOp: When this node fires, if the MPICH-G2 job failed it will evaluate to true, in which case the *MPICHG2_CG* node fires, this node is really just a recursive call to the graph itself and causes it to execute again. (When the graph is re-executed, any machines that have failed with not be in the list generated by the *GetClientListOp* node.) Otherwise the job will have executed correctly and the graph exits and returns the results to the Portal for collection by the user.

References

[FK97] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[JPMP04] David A. Power John P. Morrison, Brian Clayton and Adarsh Patil. WebCom-G: Grid Enabled Metacomputing. *The Journal of Neural, Parallel and Scientific Computation. Special Issue on Grid Computing.*, 2004(12)(2):419–438, April 2004.

[Kar] Nicholas T. Karohis. MPICH-G2: A grid-enabled implementation of the message passing interface. http://citeseer.ist.psu.edu/554400.html.