ReQueST: Resource Quantification for e-Science Technologies

Robert Atkey and Kenneth MacKenzie
LFCS, School of Informatics, University of Edinburgh
bob.atkey@ed.ac.uk, kwxm@inf.ed.ac.uk

17th July 2006

1 e-Science and the Grid

Increasingly, large scale science is being carried out through distributed global collaborations enabled by the Internet. Computational grids [4] provide powerful computational and data storage services via large-scale networks. Many areas of modern science, such as astronomy, particle physics, chemistry, biology and healthcare now have databases whose size is measured in terabytes or petabytes and the Grid provides an ideal means for accessing and processing such data. Some examples demonstrating the scale of current e-Science activities include:

- In 2007 the Large Hadron Collider (LHC) will come into service at CERN in Geneva. This will study particle collisions at very high energies. It is estimated that LHC will produce 15 petabytes (15 million gigabytes) of data per year, all of which will need to be stored, shared and analysed.
- In the USA, the Sloane Digital Sky Survey (SDSS) is currently in progress. This aims to provide detailed optical images covering more than a quarter of the sky, and a 3-dimensional map of about a million galaxies and quasars. Later surveys will collect specialised information to address fundamental questions about the nature of the Universe, the origin of galaxies and quasars, and the formation and evolution of our own Galaxy. The data which has been collected for the basic sky survey (and which can be accessed at the SDSS SkyServer website `cas.sdss.org`) already exceeds 14 terabytes. Data from the SDSS and other astronomical databases is also accessible via web services at the Virtual Observatory Web Services site, `www.voservices.org`.
- The IXI (Information eXtraction from Images) project uses the Grid to enable analysis of large amounts of medical imaging data. One aim is to help research on rheumatoid arthritis via extraction of information from magnetic resonance images. The image analysis algorithms involved can take several hours per image, so analysis of large quantities of images can take a very considerable time. Grid computation can speed this process up by distributing image analysis tasks to many different computers.

2 Resource Guarantees and PCC for the Grid

Large-scale e-Science projects can generate so much data that it is impractical to take the traditional route where each scientist keeps a local copy of the data on their computer for analysis. Instead, users will have to send programs across the network to be executed on or near the database itself. When the machines involved are shared between many users it is imperative that one user's program does not monopolise all the resources of the machine. The current mechanisms for resource specification on the Grid are somewhat crude. A user may supply estimates of execution time and memory requirements which are then used for scheduling jobs, but these estimates are usually just informed guesses, with no guarantee that they are accurate. This can lead to situations where a user submits a job which fails to complete due to lack of memory, for example: in this case the job will simply be terminated. The user (or their funding body) may have paid real money for access to the remote computer, and this money will have been wasted.

The ReQueST approach aims to avoid this type of problem. We hope to use advanced static analyses and Proof-Carrying Code to provide accurate and certified resource bounds for Grid applications. We distinguish between two different kinds of Grid application in current use: *compute grids*, where a user submits a piece of code and its data for processing somewhere on the Grid; and *data grids*, where large databases such as the ones mentioned earlier are made accessible via wide-area networks. Both of these applications will benefit from more information about resource usage, but we believe that in the immediate future data grids will have the most to gain from PCC. Current compute grids are usually comprised of clusters of nodes isolated from the network and arranged so that they may be easily reset to a known state, lessening the impact of malicious or badly-written code.

Data grids such as the Virtual Observatory allow remote users to submit queries against the data in order to retrieve subsets for local analysis. Such queries allow the selection of all objects within a region of the sky or all objects of a certain type, but more complex queries, involving detailed analysis of spectra for example, are limited by the capabilities of SQL. Most modern database server systems allow the installation of *stored procedures* (also known as *user defined functions*). Stored procedures are pieces of code that are executed within the database process and are used to augment the capabilities of the query language. Traditionally they have been written in proprietary extensions of SQL, but now they can be written in high-level languages such as Java or C#. As an example, a new function for analysing photographs taken by a telescope may be installed and subsequently used in normal SQL queries. Performing such analysis on the server reduces the amount of data that must be transferred. Further applications include custom federation of databases, where users submit queries on other remote databases from within queries.

The prospect of running untrusted code that can possibly access valuable resources such as the network from within the database process itself is an ideal application for proof-carrying code. Database server software already dynamically prevents stored procedures from accessing the local disk or using the network, but does not constrain their resource usage. We will use a certifying compiler which not only produces a compiled version of the source program, but also uses the results of static analyses to calculate bounds for the resource consumption of the program and produce a condensed proof that the program satisfies these bounds; the database server can then check the proof before executing the stored procedure.

Ultimately, it is likely that the two kinds of grid will converge, and PCC will have a role to play in complex federations of computing and data resources.

In the earlier Mobile Resource Guarantees project [5] we developed cost models and proof techniques for Java bytecode and a PCC system based on bytecode generated from a high-level functional language called Camelot. The ReQueST project is adapting these techniques for application to Grid programs; we aim to extend to Java as a high-level language and introduce techniques for handling external library code (for instance, compiled from C or Fortran) whose source is not available.

3 Contributions

Static analysis of resource usage. We wish to provide an automated framework which integrates well with established programming practice, and to this end we aim to use the ESC/Java2 [3] static analysis tool as a resource bound analyser. ESC/Java2 accepts Java programs which have been augmented with Java

Modelling Language (JML) specifications of program behaviour, and performs automated static checking for violation of these specifications. JML already has some support for annotations specifying time and space usage, but these are somewhat limited. We have already made some progress in improving and extending the JML resource-related annotations and modifying ESC/Java so that it can verify specified bounds on heap space allocation. Further details appear in [2].

ESC/Java2 does not produce independently verifiable proofs (and indeed is unsound) so it is not suitable for PCC. However, with more work, it will be useful for verifying heap space bounds on users' jobs before they are submitted to compute grids, ensuring that the resource allocation they have requested is sufficient. We are currently investigating the use of the Coq proof assistant for formalising Java bytecode to provide a sound, verified basis for PCC.

Resource policies. In [1] we have investigated resource specifications and policies. In our scenario, a code producer provides per-method resource bounds which are functions of the sizes of input arguments. Meanwhile, the code consumer has a policy which provides a promise that method arguments shall not exceed certain sizes, but requires in turn that the method's resource consumption shall not exceed a certain constant. The form of our policies means that it is easy for a consumer to test whether its resource policy is satisfied provided the producer's claimed bounds actually hold. If the test fails then a job can be rejected immediately; if the test succeeds then the consumer can go on to check a proof that the producer's code actually satisfies the given bounding function. Our policies can be made parametric in platform-dependent library functions, so that it is unnecessary for the producer to know the precise resource behaviour of a target platform in advance. See [1] for more details.

An application to Astronomy. We are currently collaborating with researchers at the Royal Observatory, Edinburgh. We are investigating the possibility of using static analysis and PCC to certify resource properties of stored procedures running on astronomical databases.

Acknowledgment. This work was funded by the ReQueST grant (EP/C537068) from the Engineering and Physical Sciences Research Council.

References

[1] David Aspinall and Kenneth MacKenzie. Mobile resource guarantees and policies. In *Proc. Intl. Workshop on Construction and Analysis of Safe, Secure and Interoperable Smart Devices (CASSIS 2005)* (LNCS 3956), pages 16–36. Springer-Verlag, 2006.

[2] Robert Atkey. Specifying and verifying heap space allocation with JML and ESC/Java2. In 8th Workshop for Formal Techniques for Java-like Programs (FTfJP2006), Nantes, July 2006.

[3] David R. Cok and Joseph R. Kiniry. ESC/Java2: Uniting ESC/Java and JML. In Gilles Barthe, Lilian Burdy, Marieke Huisman, Jean-Louis Lanet, and Traian Muntean, editors, *CASSIS 2004*, volume 3362 of *LNCS*, pages 108–128, January 2005.

[4] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] Donald Sannella, Martin Hofmann, David Aspinall, Stephen Gilmore, Ian Stark, Lennart Beringer, Hans-Wolfgang Loidl, Kenneth MacKenzie, Alberto Momigliano, and Olha Shkaravska. Mobile Resource Guarantees. In *Trends in Functional Programming*, volume 6. Intellect, September 2005.