ReQueST : Resource Quantification for e-Science Technologies

David Aspinall, Robert Atkey, Stephen Gilmore, Kenneth MacKenzie, Don Sannella, Jaroslav Ševčík, Ian Stark
LFCS, School of Informatics, University of Edinburgh
http://groups.inf.ed.ac.uk/request/

Grid Computing

The goal of ReQueST is to apply the Proof-Carrying Code concept to Grid computing and e-Science.

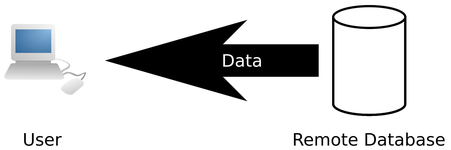
Grid computing aims to commoditise super-computing-level processing power and large shared databases. Using the Grid is meant to be as simple as using the electrical grid. The primary users are intended to be scientists dealing with the terascale data sets generated by modern e-Science experiments.

The Grid is built on code being executed remotely by untrusting hosts. Of particular importance is safe use of computing resources such as CPU time and memory space. A rogue piece of code must not be allowed to monopolise a shared resource.

Proof-Carrying Code and Shared Databases

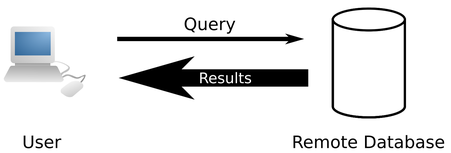
The traditional way to access remotely produced and curated databases was to download the data locally.

This approach is no longer feasible since modern databases can have tens of terabytes of data and are growing. For example, the Sloane Digital Sky Survey covers only ¼ of the sky, but still contains over 14TB of data relating to 215 million astronomical objects.



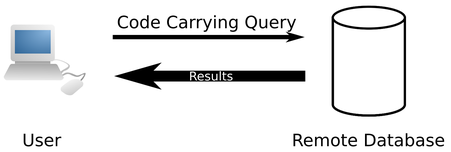
As a solution, scientific databases now allow the user to submit queries specifying the data the user requires.

This reduces the amount of data that must be transferred. However, the querying possibilities are limited by the database's SQL implementation and the amount of data can still be large.



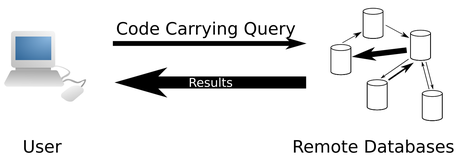
Many database servers allow the use of Stored Procedures: pieces of code that are executed within the server, extending the database's native queries.

Stored Procedures are traditionally written in a proprietary dialect of SQL, but now they can be written in languages such as Java and C#.

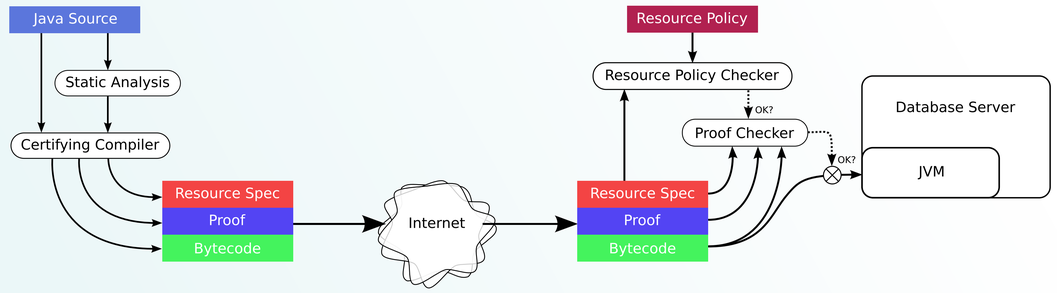


We intend that users submit queries along with custom analysis code that it is more efficient to run near the data. Such code is run directly inside the server and so database providers must ensure that a single user's query does not dominate the server.

In the future, multiple databases will be federated to create massive virtual databases. As a result the relationship between code producer and code consumer will become more and more distant and so the need for PCC will become greater.



Our PCC Infrastructure



We are building on the results of the MRG (Mobile Resource Guarantees) project previously completed at the University of Edinburgh and LMU Munich.

MRG developed a resource-certifying PCC infrastructure for a subset of Java bytecode and a functional high-level language, Camelot. ReQueST aims to take the results of MRG and apply them to a larger subset of Java bytecode and to use Java as the source language.

```
//@ \working_space 0;
public bool analyseAndSelect (Row r) {
      ...
}

//@ working_space \array_space (float, 200)
//@               + \object_space (Cache);
public void initCache () {
      ...
}
```

The `working_space` annotation specifies an upper bound on the heap space allocated.

We are using JML (the Java Modelling Language, see example) to specify the resources consumed by Java programs. JML's current design includes keywords for specifying resource usage. We have altered and simplified the design to make the annotations more useful and also verifiable.

We have implemented an extension to ESC/Java2 to verify JML specifications of heap space allocation. ESC/Java2 does not produce independently checkable proofs (and indeed is unsound). We intend to produce a certifying compiler that produces machine checkable proofs of resource bounds as illustrated in the PCC infrastructure diagram above.

One source of difficulty is that mobile programs will often call system methods whose implementation and resource usage is platform-dependent. To deal with this we have developed a resource specification scenario in which the code producer provides a certificate of resource usage which is parametric with respect to platform-dependent system calls. The certificate format allows the code consumer to perform a rapid check to reject code which does not satisfy the consumer's requirements on resource usage.

To ensure the soundness of our infrastructure, we are also developing an executable specification of the Java Virtual Machine in Coq. We intend to use this to develop a sound logic in which to write proofs of resource bounds and to extract a certified proof checker.

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