Research Infrastructure Proposal for the IRF: GATE Application Services for the Large Data Collider

What is the Large Data Collider (LDC)?

The vision: provide research infrastructure for IR.

The hardware: "An SGI Altix 4700 with 80 processor nodes with 360 GB random access memory operate in the form of a large virtual system called LDC (Large Data Collider), modelled on CERN's LHC (Large Hadron Collider)."

Next steps: software infrastructure to support running experiments on the Altix. This note proposes one element of that software.

What are GATE Application Services (GAS)?

GATE (http://gate.ac.uk/) is a platform for R&D in human language processing with a component-based architecture. GAS layers a Service-Oriented Architecture on top of GATE to provide distribution of compute-intensive tasks over multiple processors. It is transparent to the external user how many machines are actually used to execute a particular task.

The GAS architecture utilises two types of components:

- The web service *endpoint*, that accepts requests from clients and queues them for processing.
- One or more *workers* that take the queued requests and process them.

The two sides communicate using the Java Messaging System (JMS), a framework for reliable messaging between Java components. If a particular service is heavily loaded it is a simple matter to add extra worker nodes to spread the load, and workers can be added or removed dynamically without needing to shut down the web services. The configuration and wiring together of these components is handled using the Spring Framework.

GAS is one part of SAFE, the Semantic Application Factory Environment (http://gate.ac.uk/safe).

Attracting Researchers to the LDC

The IRF can become a supplier of research infrastructure for IR and other language-related work. The first steps have been taken by funding the LDC hardware and providing a first data set for experimentation.

One problem is that the startup cost for a researcher wishing to use the LDC is still relatively high, involving adaptation of their experimental setups to remote working on a highly parallel machine, marshalling of data and results to share with their home systems, and so on.

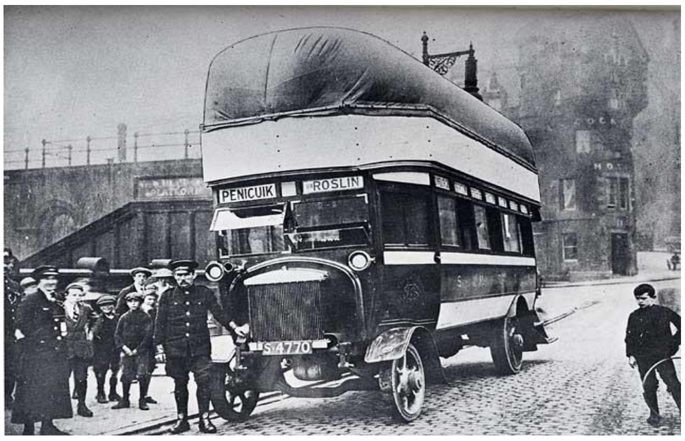
This proposal is to remove some of that load and make experimentation on the LDC easier for a large class of research tasks. In doing so we'll take a step towards a new type of virtual publication, where the data, software and platform for an experimental result are all persisted and made available to the peer community. We'll call this this part of the picture PERLS: Persistent Experiment Repositories for Language Science.

The proposal: GAS on the LDC

The proposal is to deploy GAS on the LDC and provide mechanisms for researchers to upload their experiment definitions, have them executed on the parallel hardware, and the results made available back to them over the web. Coupled with GATE's integrated development environment this will enable remote development, debugging and evaluation of experimental language analysers at a larger scale than is currently possible on typical experimental setups.

GAS's J2EE-based SOA has a number of benefits in this context:

- It allows us to exploit hardware parallelism without changing our code as the SGI Linux port already supports process-level task distribution across processor nodes. (In effect GAS on the LDC will behave as if there were 80 high-spec servers linked together by a high speed bus and a storage area network.)
- Distributed computing and parallelisation require a lot of heavy lifting; in our case a lot of this work is done for us by libraries from Sun and the Java community, leading to low development overhead, high robustness and longevity.



PERLS: Persistent Experiment Repositories for Language Science

"Science" has as many definitions as there are philosophers, but an important element of several of its forms is *experiment*. Practitioners form hypotheses, design procedures to test those hypotheses, and publish the results in ways intended to allow their peers to reproduce them.

A common problem, especially in areas related to human behaviour or intelligence, is precisely how to measure results, and progress in computational processing of human language has increasingly been driven by the provision of standard test collections and evaluation metrics that provide a level playing field on which to compare hypotheses and related experimental systems. This practice has contributed greatly to experimental repeatability, which is a key factor in the sharing of results across a research community. Without the infrastructure that test collections and evaluation tools provide it is much more difficult to reproduce published work, much more difficult to confirm theories, and much more difficult to distinguish a novel contribution from a reiteration.

(This discussion may seem commonplace to those from other disciplines; the issue commonly arises in fields connected in one way or another to the field of Artificial Intelligence. This is probably partly for cultural reasons related to the practitioner groups and partly because of unresolved difficulties in defining the subject matter (quite understandable in this case, where we're rather like lab rats who've been promoted to principal investigators and now have to study ourselves!). Consequently the increase in experiment-driven work described above has often gone hand-in-hand with a step back from the project of simulating intelligence and the adoption of more limited goals, for example the turn from language understanding research to information extraction research in the 1990s.)

Three factors in recent history mean that it has now become possible to extend the reach of our experimental infrastructures in several ways, and in doing so increase the power and impact of our research. Over the last several decade there has been a long-running conformance of compute power increase to Moore's 'law', and at the same time network bandwidth has similarly extended in both volume and geographical reach. Lastly, software and the platforms that it runs on have become mobile (able to move between different hardware systems). Taken together, these factors mean we can publish not only our results, but the complete set of platform, software, configuration, intermediate data and measurement tools that underly the results, and we can do so in forms which allow the dynamic recombination of the elements of our work in new experiments of ourselves and our colleagues. For the areas of Information Retrieval, Natural Language Processing and Speech Recognition, we can start to build Persistent Experiment Repositories for Language Science: PERLS.

Characteristics of PERLS repositories:

- Distributed reproducible experiments. Web-based interfaces and mobile code that allow experiments defined and executed in one place to be persisted in another and retrieved and reproduced from a third.
- Multiple repositories. Setting up a repository needs to be easy enough for research groups of a handful of people to create and maintain their own.
- Platform neutral. Software system longevity is typically compromised by technology churn in the underlying platform. One answer to this is to use free software, which has a better track record (and isn't under pressure to force users to upgrade through incompatible changes). Another answer is to use OS-level virtualisation (Xen, VMWare, VirtualBox, etc.) to provide computational environment persistence.
- Versionned. Both software and data need to be persisted in version controlled storage.
- Informed by related Grid and eScience programmes.