

## Polar\*

The Polar\* Project, with members based at Manchester and Newcastle Universities, is concerned with distributed query processing in a grid environment.

In any distributed environment there are inevitably multiple related data resources, which may provide complementary or alternative capabilities. Where there is more than one database supported within a distributed environment, it is easy to envisage higher-level services that assist users in making use of them within a single application. For example, in bioinformatics, it is commonly the case that different kinds of data (e.g., DNA sequence, protein sequence, protein structure, transcriptome) are stored in different, specialist repositories, even though they are often inter-related in analyses.

Consider for example the two databases:

- GO, the Gene Ontology database; and
- GIMS, a genome database .

These might be located at two different sites and an installation of the sequence similarity program, BLAST, might be located at one or other site or elsewhere.

An example query identifying proteins similar to those with a particular GO term and accessing all three resources is:

```
select struct(A:c.proteinID, B:Blast(c.sequence))
from c in proteins, d in proteinTerms
where d.termID="8372" and c.proteinID=d.proteinID;
```

An example configuration where GO is installed in a MySQL database and GIMS in a Polar database is shown in Figure 1.

The BLAST server hosts a local sequence database, and an example operation, *blastall*, might be to compare each incoming sequence with those stored locally.

The distributed database community has long studied

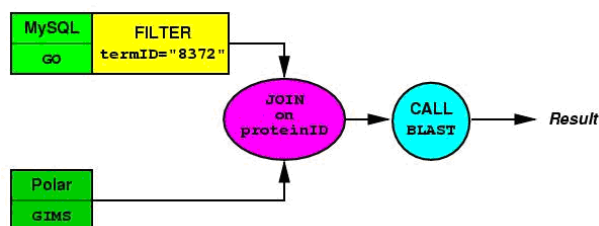
a range of problems related to supporting querying over autonomous pre-existing distributed databases:

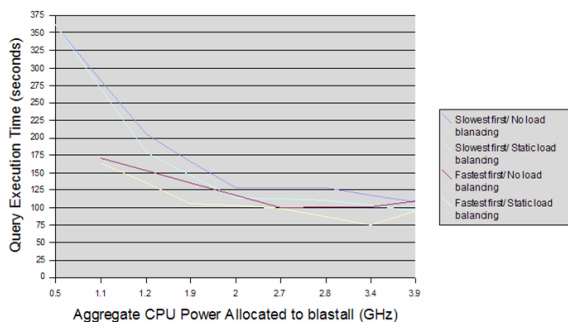
- The hardware and software systems can exhibit various forms of architectural heterogeneity, for instance with machines having different endianness, database systems having different levels of support for SQL and so on.
- The database model can differ, being for instance relational or object-oriented.
- The application schemas can exhibit semantic heterogeneity, representing the same real world entities in different ways and representing different real world entities in conflicting ways.
- The cost of transferring data between the sites of participating databases can be significant as those sites can be widely distributed.
- The characteristics of a particular physical database schema, such as cardinalities and access paths, are typically not known outside of the local database system, and may differ in form between database systems.

Commercial systems exist which can overcome these issues, but generally take a "hands-off" approach to the local databases; necessitate a somewhat static configuration; and involve the local databases considerably in the planning of distributed queries. The open and dynamic philosophy of Grid Computing motivates not only support for more dynamic integration, but also exploitation of any additional computational resources available to a particular virtual organisation.

In order to further understand the issues involved, the Polar\* project has implemented a prototype distributed query processing system over the Globus infrastructure [1]. While the issues of semantic heterogeneity are not addressed, a system of wrapper generation is defined for both data sources and analysis functions, to enable their representation in a common schema. Then, bulk import of logical and physical schemas from participating databases into, and consistent representation of all computational resources in, the global metadata permits direct application of the optimiser strategies of Polar, a shared-nothing parallel server. Moreover, since Polar runs on MPI, a Polar\* prototype can reuse much of Polar through

Figure 1





**Figure 2: Parallelisation of Example Query**

porting to the Globus enabled MPICH-G, thereby acquiring support for grid facilities such as secure single sign-on and executable staging.

Experiments were conducted with the bioinformatics example described above using a small collection of heterogeneous machines distributed between Manchester and Newcastle Universities. GO was installed on a dedicated server at Manchester and GIMS on a 2-way SMP at Newcastle. The BLAST services each hosted a subset of the Swiss-Prot database. The CPUs available to execute parts of the distributed query, including the two serving GIMS, have raw speeds ranging from 0.5 to 1.1 GHz, memory sizes ranging from 128 to 512 MBytes, and were running different versions of Linux and the Globus toolkit. In the example, the key issue for query planning is that of allocating the operation calls to available CPUs. Fig 2 shows the measured execution cost of the overall query when the degree of parallelization of the operation call is increased according to different policies. The CPUs are allocated either slowest-first or fastest-first. Optionally, the scheduler parameterizes the operator which implements the redistribution of tuples prior to the operation call to effect static load balancing with respect to the raw CPU speeds of the destinations.

Since the fastest processor is located at Manchester and the remainder at Newcastle, the results justify parallelization using resources distributed between the two sites in this case. Also, the results demonstrate that even a very simple load balancing scheme can give significant benefit when, as shown in Fig 3 for the particular *blastall* calls made, there is a significant variance in the distribution of individual operation call times.

For repeatability, the initial experiments used a fixed collection of fully available resources. Also, while the network is shared, the queries were not of such great



**Figure 3: Distribution of blastall Times**

size or so dominated by data transfer that the variability of network performance would be a serious issue. However, parallel and distributed query processing technology is seen to bring a useful high level expression of suitable distributed applications to the grid computing environment, yet is itself enhanced in the grid environment by having access to not only a pool of available computational resources but also access to relevant parameters of those resources through the grid information services in order to support effective query planning. In the experiments, only a single static parameter, the raw CPU speed was used, but more dynamic information is available through the grid information services or may be obtained by monitoring query execution.

In reality, the resources available to the DQP system would be autonomous. For example, during execution of a query, one or more resources could fail, or be made unavailable by its owner to the DQP system. However, such a query might be long running, and have completed substantial useful work. Thus, ongoing work is aiming to define and evaluate protocols which support tolerance to failure of one or more resources during execution of a distributed query.

[1] Jim Smith, Anastassios Gounaris, Paul Watson, Norman W. Paton, Alvaro A. A. Fernandes and Rizos Sakellariou, *Distributed Query Processing on the Grid*, to appear in *International Journal of High Performance Computing Applications*

