Utilising Located Functions to Model and Optimise Distributed Computations

Stephen Crouch, Peter Henderson, Robert John Walters
University of Southampton,
Highfield, Southampton,
United Kingdom,
SO17 1BJ
{stc,ph,rjw1}@ecs.soton.ac.uk

Abstract

With developments in Grid computing and Web based data storage the task of orchestrating computations is becoming ever more difficult. Identifying which of the available computation resources and datasets to use is not trivial: it requires reasoning about the problem itself and the cost of moving data to complete the computation efficiently.

This paper presents a conceptual notation and performance model that enables e-researchers to reason about their computations and make choices about the best use of resources.

1. Introduction

Developments in Grid computing and Web based data storage [1] are enabling large computations using distributed resources. As exemplified by Web Services, current modelling approaches abstract away details of the location of functions and data to permit attention to be concentrated on the calculations; users are discouraged from considering location. However, many of these computations involve massive processing power and enormous datasets so locations have to be considered if they are to be performed efficiently and reliably.

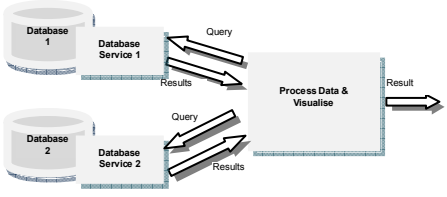

Figure 1: An operation on a distributed system

In this paper we propose a notation, "located functions" which permits concise description of operations using distributed resources which can include information about the location of the elements and we describe a simple measure to assist with making rational decisions about which permutation of an operation to employ.

2. Located Functions

Consider a task which combines and formats data from two databases (see Figure 1).

$$f\left(g\left(D_1, D_2\right), h\left(D_1, D_3\right)\right)$$

Figure 2: An example expression

Adopting the Web Services view, this operation may be reduced to the expression in Figure 2.

2.1. Located Data

However, when performing computations, locations are important; functions and data need to be co-located which implies movement and there is the question of how best to orchestrate these encounters. Located functions provide a notation for reasoning about this problem.

Located functions "decorate" elements of the expression with location information (see Figure 3 in which D_1 is available at location 1, D_2 is in location 2 and D_3 is in location 3).

$$f\left(g\left(1:D_1, 2:D_2\right), h\left(1:D_1, 3:D_3\right)\right)$$

Figure 3: Including Data Locations

If *f,g,h* are available throughout the system, Figure 3 contains all the information necessary

to make rational decisions about how to evaluate the expression: one of D_1, D_2 has to be moved in order to evaluate *g* and one of D_1, D_3 has to be moved in order to evaluate *h*.

2.2. Locating functions

Locations where *f,g,h* are available can be added, as shown in Figure 4.

$$1/2 : f\left(2 : g\left(1 : D_1, 2 : D_2\right), 1/3 : h\left(1 : D_1, 3 : D_3\right)\right)$$

Figure 4: Including Data and Function locations

It is now clear that the result of one of *g* or *h* has to be moved and there is choice for the execution of *h*. In deciding, available processing power at locations is a factor as is the volume of data which has to be processed: many Grid operations work on very large datasets [1-3].

Identifying the best locations to perform calculations is complex; in this small example there are over twenty possibilities. Using data about processing power, sizes of datasets and bandwidths between locations we can estimate a relative cost for each strategy. For data movement, the cost is the size of the data divided by bandwidth. For functions, we suggest the size of the parameters divided the location's processing power. For example, the cost of evaluating *h* is given by the cost processing *h* where it is available plus the cost of any movement of its inputs. If the size of dataset 1 is 10 and dataset 3 is 100, the data throughput at 1 and 3 are 5 and 1000, respectively, the bandwidth from 1 to 3 is 10, then the cost for running *h* is one of:

$$(0 + (100/10)) + (10 + 100)/5 = 32 \quad \text{(executed at 1)}$$

$$((10/10) + 0) + (10 + 100)/1000 = 1.11 \quad \text{(executed at 3).}$$

The cost of executing *g* is calculated similarly. When evaluating the cost of *f*, the costs of the sub-computations need to be added, as appropriate, giving a figure for the whole computation from which to decide the details of how to perform the whole calculation.

2.3. Issues in Real Grid Deployments

Appling this technique to real systems, there are additional factors which need consideration including bandwidth throttling to obtain a level of fairness between clients [4] and security [5]. These can be included in our model by appropriate manipulation of inter-site bandwidths, notwithstanding the underlying networking infrastructure. For simplicity, we have assumed static bandwidth configurations in our example.

3. Conclusions

When modelling real complex systems, achieving the correct level of abstraction is important [6]. We have described "located functions", a notation which provides a representation for problems which involve the combination of distributed data and processing resources. We have also illustrated how the notation might be used to arrive at a time-like approximation which can be used to decide which datasets should be moved and which of the possible locations for the evaluation of each element is to be preferred.

4. References

[1] J. Bradley, C. Brown, B. Carpenter, V. Chang, J. Crisp, S. Crouch, D. de Roure, S. Newhouse, G. Li, J. Papay, C. Walker, and A. Wookey, "The OMII Software Distribution," in *UK e-Science All Hands Meeting 2006 (NeSC 2006)*, Nottingham, UK., pp. 748-753.

[2] F. Berman, A. J. G. Hey, and G. C. Fox, *Grid Computing: Making the Global Infrastructure a Reality*: John Wiley and Sons Ltd, 2003.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scaleable Virtual Organization," *International Journal of Supercomputer Applications and High Performance Computing,* vol. 15, pp. 200-222, 2001.

[4] A. Hagin, N. Hagin, and V. Voinov, "Providing Quality of Service on the Web Using Bandwidth Throttling.," in *5th Workshop of the OpenView University Association OVUA'98* Rennes, France, 1998.

[5] I. Foster, K. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *5th ACM conference on Computer and communications security*, 1998.

[6] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Sixth Symposium on Operating System Design and Implementation (OSDI'04)* San Francisco, CA, 2004.