A dynamic world, what can Grids do for Multi-core computing?

Daniel Goodman[1], Anne Trefethen[1] and Douglas Creager[2]
[1]Oxford University e-Research Centre, [2]Oxford University Computing Laboratory

Daniel.Goodman@oerc.ox.ac.uk
Oxford e-Research Centre, 7 Keble Road, Oxford, OX1 3QG
+44 (0)1865 610703

Introduction

We will provide an overview of current strategies for writing effective programs for multi-core processors. In this we will show that there are currently several models being developed, with many drawn from cluster computing. We however, believe that for a range of the environments that multi-core chips will operate in, models need to be more dynamic and as a result, a more appropriate approach is one which is similar to those used in "Grid" computing, both for large data centres (1) and for projects such as Climate*Prediction*.net (2).

In cluster computing it is normal for a task to be constructed to run on a specific number of the cluster nodes in order to achieve peak performance; this is because usually the cluster nodes are a homogenous set. However, with a multi-core computer this is rarely possible as the number of available cores will vary for reasons, ranging from those which are constant for the lifetime of the application, such as the application being run on different models of processor without being able to recompiling to match, to those which may vary from second to second such as other applications running at the same time. The applications running may take many forms, from of producers and consumers of data in a pipeline, to Daemons belonging to the operating system, to the users MP3 player or email client. The latter cases are still very much the preserve of scientific computing, as we will still expect scientific software packages to run on our desktops without requiring a dedicated computer. As some applications may not be sufficiently concurrent to run on all the available cores, it is clear that simply slicing the time and giving each application all the cores in a processors is not the best way to gain maximum performance. In addition such a solution could result in significant IO delays when performing operations such as pipelining. The alternative of allowing individual processors to come and go in order to support such demands on the computer could lead to an equal amount of resources being wasted if applications are unable to handle the changing environment. For example if the task was expecting all n cores, and was not able to neatly handle the dynamic nature of the allocation, all other cores may stall while one is unavailable. As such just compiling an application for a given number of threads and assuming each thread will be allowed to progress at roughly the same rate will not be suitable for all cases.

This implicitly heterogeneous nature to multi-core processors together with the growing use of GPU's for numeric computation means that the computers of the future (and in some cases present) will not only have many processors with different loads, but also many processors of different types, connected through different types of memory and communication architectures. The loss of the simplicity of homogeneous clusters will span from the desktop to high-performance computers such as HECToR (3). This change we believe will mean that these environments will have much in common with the models developed for Grid computing.

We will also identify projects such as LINQ from Microsoft (4) and Data Parallel Languages (5), that we believe while appearing unrelated, are in many ways closely related to models that are being developed for Grid computing.

Background

While in the past processors have been made faster through increasing the clock speed, designers are now reaching physical barriers that for the time being at least are making this untenable. As such designers are now looking at offering more performance through the inclusion of multiple independent processor cores in the same chip. While currently PC processors are only offering two or four core processors, graphics cards and cell processors are offering vastly more cores and it is expected that PC processors will follow suit over the coming years. With these changes comes an end to the ability to get best performance out of a processor with the single threaded programs that are common in most applications today. Instead it is necessary that either the programmer or tooling provides multi-threaded concurrency to allow the program to take advantage of the multiple heterogeneous cores. Ideas for how best to do this are drawn from a range of different areas, but the most prominent area is, unsurprisingly, that of cluster computing. Other areas where ideas are being drawn from include previous experience with shared memory super computers and functional programming.

Grid computing is a term that has been used to describe projects that attempt to take advantage of large amounts of distributed computing resources, often in a dynamic way, to address problems such as how to you get enough computing power to handle the data output from the Large Hadron Collider (LHC) at CERN? Or how do you analyse patient data collected in many different hospitals (6)? The proposed answers to these questions typically involve computing models for large, loosely coupled, distributed, utility computing systems spanning multiple locations, and often multiple organisations.

Bibliography

1. **Jeffrey Dean and Sanjay Ghemawat.** MapReduce: Simplified Data Processing on Large Clusters. *Operating Systems Design and Implementation.* 2004.
2. **Goodman, Daniel.** Introduction and Evaluation of Martlet, a Scientific Workflow Language for Abstracted Parallelisation. *WWW.* 2007.
3. **The HECToR Partners.** *HECToR.* [Online] http://www.hector.ac.uk/.
4. **Meijer, E., Beckman, B., and Bierman, G.** LINQ: reconciling object, relations and XML in the .NET framework. *ACM SIGMOD international Conference on Management of Data.* 2006.
5. **Blelloch, Guy E.** *NESL: A Nested Data Parallel Language.* Pittsburgh : Carnegie Mellon University, 1995. CMU-CS-95-170.
6. *On the development of secure service-oriented architectures to support medical research.* **Simpson A. C., Power D. J., Slaymaker M. A., Russell D., Katzarova M.** 2, s.l. : The International Journal of Healthcare Information Systems and Informatics, 2007.