The influence of “the Grid” on 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 show that for a range of environments models used for “Grid” computing are the most effective approach for multi-core systems. We will provide an overview of current strategies for writing effective programs for multi-core processors. There are several models being developed many drawn from cluster computing. We believe that for a range of environments that multi-core chips will operate in, models need to be more dynamic and as a result, a more appropriate approach is that which is similar to those used in “Grid” computing, both for large data centres [1] and for projects such as Climate*Prediction*.net [2].

The future (if not present) desktop computer will consist of a collection of different computational processing units including gpus, multi-core processors and the like. The result is a highly heterogeneous system. This heterogeneity of platform is likely to be reflected throughout the scales of computing from the desktop to the high-performance systems.

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 usually because the cluster nodes are a homogenous set. 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 being unable to compile the application for the specific model of processor, to those which may vary from second to second such as other applications running at the same time. The applications running may take the form 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 be inappropriate to run in a sufficiently concurrent way to use all the available cores, coupled with the potential for IO and cache thrashing issues when performing pipelining, 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 such circumstances. 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. For example if the task was expecting all n processors, and was not able to neatly handle the dynamic nature of the allocation, all other processors 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 coupled with the growing use of GPU’s for numeric computation means that the computers of the future will not only have many processors with different loads, but also many processors of different types, coupled with different memory architectures. The loss of the simplicity of homogeneous clusters will span from the desktop to super computers such as HECToR [4], and this why we believe that these environments will have much in common with the models developed for Grid computing.

In addition we will identify projects such as LINQ from Microsoft [5] and Data Parallel Languages [6], that we believe while appearing unrelated, are in many ways closely related to models that are being developed for Grid computing.

References

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. **Mathworks.** *MATLAB.* Natick, MA : Mathworks, Inc., 1999.
4. **The HECToR Partners.** *HECToR.* [Online] http://www.hector.ac.uk/.
5. **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.
6. **Blelloch, Guy E.** *NESL: A Nested Data Parallel Language.* Pittsburgh : Carnegie Mellon University, 1995. CMU-CS-95-170.

7. *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.