ON SPECIAL-PURPOSE HARDWARE CLUSTERS FOR HIGH-PERFORMANCE COMPUTATIONAL GRIDS*

Jeanne M. Lehrter†, Faisal N. Abu-Khzam†, Donald W. Bouldin‡,
Michael A. Langston†§ and Gregory D. Peterson‡

Abstract

The problem of incorporating advanced hardware resources into a grid computing environment is considered. An experimental special-purpose cluster of machines is designed and placed into service on a prototypical grid. The cluster is populated with CAD workstations, PCs and reconfigurable devices, and then used to develop accelerated implementations for amenable applications. Access mechanisms are devised so that users may request either software solutions, which may be satisfied by our cluster or a variety of other clusters resident on the grid, or accelerated solutions, which can only be satisfied by hardware-enhanced clusters such as ours. Preliminary results are described. A number of continuing research issues are addressed.

Key Words

Grid and Reconfigurable Computing, Algorithms for Heterogeneous Systems, Cluster Computing

1 Introduction

A computational grid may take a variety of forms, from a simple stand-alone collection of but a handful of identical processors, to vast networks with many types of compute engines. It can facilitate collaboration and data sharing. It can be used to capture cycles that would otherwise be wasted. In many high-performance applications, its inherent parallelism can even obviate the need for expensive supercomputers.

Reconfigurable hardware offers a means for combining the performance of custom IC design with the flexibility of software. Its most critical feature is the reconfigurable processing element which, in the current generation, is a Field-Programmable Gate Array (FPGA) chip. Because these elements can be dynamically reconfigured to implement application-specific computations, one can often achieve orders of magnitude improvements in price and performance over conventional processors for a variety of applications.

Both grid and reconfigurable computing are rapidly gaining in popularity. We are keenly interested in novel applications that can exploit the synergies of these two rather complementary technologies. A central goal is to provide grid-accessible solutions to diverse communities of scientists. Researchers having only desktop access to the grid can thus solve large computational problems, many of them once considered beyond reach, by calling on wide-area distributed computing resources and advanced hardware platforms.

2 The SInRG Project

Computational grids are rapidly maturing within the scientific community [5]. They have been a subject of intense study in both the university sector and the national research laboratories for several years. The recent announcement of NSF's $53M Distributed Terascale Facility [13], however, seems to have attracted more widespread attention. In response, strategic partnerships and commitments have been formed by an assortment of vendors, including IBM, Intel, Sun Microsystems and many others. Given its potential for providing an integrated, collaborative research environment, some have even predicted that an international grid will one day be the natural successor to the world wide web.

A major project underway here at Tennessee is the Scalable Intracampus Research Grid (SInRG) [3]. SInRG is an NSF-sponsored effort that deploys an on-campus infrastructure designed to mirror technologies and interdisciplinary collaborations characteristic of those to be encountered in a national technology grid. Thus SInRG provides an organizational microcosm in which key research challenges underlying grid-enhanced computation can be addressed. A simplified sketch of the SInRG topology is depicted in Figure 1. Although SInRG contains many links and switches not shown here, its overall structure is tree-like as suggested by this figure. At its leaves lie SInRG's Grid Service Clusters (GSCs), which we use as basic grid building blocks. As currently configured, four GSCs are designed to serve as special-purpose clusters. Others are intended for general use in computer science applications.

A variety of packages have been developed to help

*This research is supported in part by the National Science Foundation under grants EIA–9972889 and CCR–0075792, by the Office of Naval Research under grant N00014–01–1–0608, by the Department of Energy under contract DE–AC05–00OR22725, and by the Tennessee Center for Information Technology Research under award E01–0178–081.

†Department of Computer Science, University of Tennessee, Knoxville, TN 37996–3450.

‡Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, TN 37996–2210.

§Communicating author: langston@cs.utk.edu.

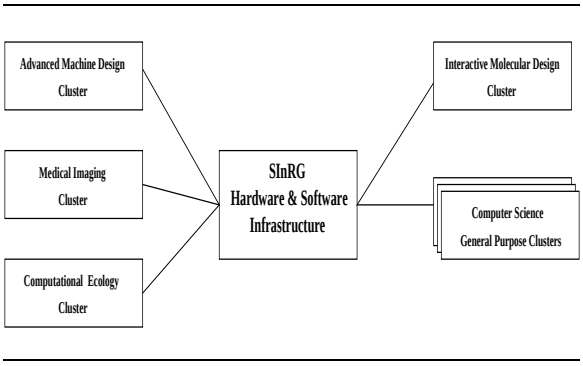

Figure 1. *An Overview of the SInRG Project.*

manage distributed computing resources. These include NetSolve [2], Condor [11], Globus [4], Legion [6], Harness [7] and others. In our work we have employed middleware from the NetSolve and Internet Backplane Protocol [1] packages. NetSolve provides a versatile client/agent/server system that unites and manages disparate computational resources. With it, users can remotely access a variety of hardware platforms and software components distributed across a network. One of NetSolve's main tenets is that users should not be required to expend time and energy finding available resources. Thus, when given a computational request, one of NetSolve's functions is to search for resources, choose those most appropriate, use them to satisfy the request, and return the answers to the user. NetSolve includes a number of other management functions, including load balancing, fault tolerance and integrated distributed storage, as well as security mechanisms such as client-to-server authentication. The Internet Backplane Protocol provides access mechanisms for remote computation and storage. Among its goals are standardization, interoperability and scalability. It manages global memory scheduling, and uses a network's physical resources to optimize data movement. This can result in a uniform, application-independent interface to network storage, giving the user an opportunity to capitalize on data locality and handle buffer resources more efficiently.

3 A Grid Service Cluster for Advanced Machine Design

The number of computational resources in a GSC is arbitrary, as is the GSC's network topology. Inherent in this model is the assumption that the GSC has an owner, who may or may not have any relationship to other grid owners and users. By joining the grid, the owner agrees that other grid participants may use the owner's GSC via the system's middleware (in our case, NetSolve). A GSC may be nothing more than a collection of general-purpose compute nodes. Alternately, it may be geared to a particular set of applications and thus contain graphics workstations or a variety of other high-end devices. As examples, the Computational Ecology GSC includes a symmetric multiprocessor and our GSC contains reconfigurable hardware.

A system augmented with reconfigurable technology can be a powerful computational tool, and can often outperform conventional general-purpose computers. This is particularly evident when an application is systolic in nature, or when it calls for limited-precision or bit-wise operations. FPGA-enhanced systems are evolving rapidly, for example, with the incorporation of on-board RAM. At a fundamental level, however, the FPGA can be viewed mainly as a versatile collection of vast numbers of configurable logic blocks (CLBs) with programmable connections [12].

As with many other new computing technologies, the hardware aspects alone have been the primary focus of development. This has created a corresponding need for algorithm design, support software and process automation. We are especially interested in applying our results to high-performance, grid-based environments with fast interconnects, CAD workstation clusters and PCs equipped with FPGAs. The aforementioned GSC of the SInRG project is an excellent example of this type of platform. In a computational grid, reconfigurable resources such as these can be extremely competitive with Application-Specific Integrated Circuits (ASICs), without an ASIC's inherent drawbacks. For example, reconfigurable components (unlike ASICs) can be built with off-the-shelf parts and can be shared by many applications. Moreover, the FPGAs can be used for prototyping and other applications, and can be reconfigured quickly to reflect changing needs across the grid. Thus the enormous benefits of hardware acceleration are achieved, but in a manner that can be dynamically refocused and fine-tuned to each new problem.

Our GSC currently comprises Xilinx Virtex parts. With it we can reconfigure either an entire FPGA or mere FPGA sections on demand. This degree of flexibility can sometimes be a great asset, because partial reconfiguration typically takes only a fraction of the time needed for full reconfiguration. Furthermore, we are in the process of supplementing our GSC with Pilchard boards [10], in which an FPGA is plugged directly into a PC's memory slot. See Figure 2. We anticipate that this platform will yield lower cost, higher bandwidth and a simplified interface.

4 Access Mechanisms

Although much progress has been made in hardware speed, cost and efficiency, the concomitant need for software support has not been well satisfied. Yet such support is sorely required if the user need no longer be a CAD expert to benefit from acceleration via reconfigurable hardware. To meet this requirement, we have employed the NetSolve middleware as an interface to ease the use of FPGAs. In particular, we have adopted one of NetSolve's resource sharing mechanisms, the Problem Description File (PDF). A PDF is a means for itemizing the relevant I/O specifications, libraries and so forth needed to pass parameters and execute programs on a particular grid element. It requires roughly a

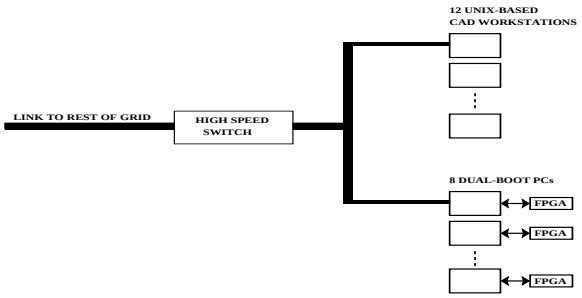

Figure 2. *A Grid Service Cluster with Reconfigurable Hardware.*

couple of dozen lines of code, depending on the calling sequence of the program in question. See Figure 3 for a sample template. Background information about PDFs, working examples and details on their design can be found in [9].

We illustrate the utility of the PDF with an example, the Fast Fourier Transform (FFT). Our GSC, like many others, has in residence efficient software versions of the FFT. These are easy to install, and generally written in C or FORTRAN. Once compiled, they are of course run on a CPU. Unlike other GSCs, however, we also have in residence an accelerated version of the FFT that uses FPGAs to speed the computation. Installing the accelerated version requires a bit more effort. First, code written in VHDL or some other hardware description language is needed. This code then goes through a synthesis process, whereby the application is mapped onto the FPGA hardware. The result is a configuration file that, when loaded onto the FPGA, defines how its CLBs should be set so that the desired functionality is realized.

Users with accounts inside our GSC may of course access either FFT without the aid of NetSolve. Suppose now that a user elsewhere wishes to determine what gain if any is achievable by executing a hardware FFT. As SInRG is currently configured, our GSC is the grid's only computational resource with the ability to satisfy the user's need. The user may not necessarily know this, naturally, but by what process can he or she seamlessly take advantage of our reconfigurable implementation? We have resolved this question by providing NetSolve with two PDFs, one for a software FFT, and one for a hardware FFT. Thus, when a remote user requests an FFT from NetSolve, one of two actions is taken. If the request specifies a software FFT, then NetSolve searches the grid for an available computational node that is appropriately configured, that is, one with a resident software FFT whose PDF has already been furnished to NetSolve. This element may be in our GSC or elsewhere. By the same token, if the request specifies an accelerated FFT, then NetSolve has the information in hand to determine that under current conditions our GSC is the

@PROBLEM Program Name
@INCLUDE <Location of Source Code>
@LIB Supporting Library Information
@LANGUAGE Source Language
@PATH /NetSolve Category/
@DESCRIPTION
@INPUT Number of Inputs
    @OBJECT Input Type Information
    @OBJECT Input Type Information
    @OBJECT Input Type Information
@OUTPUT Number of Outputs
    @OBJECT Output Type Information
    @OBJECT Output Type Information
@COMPLEXITY Asymptotic Information
@CALLING_SEQUENCE
    @ARG Mode of Parameter
    @ARG Mode of Parameter
    @ARG Mode of Parameter
    @ARG Mode of Parameter
    @ARG Mode of Parameter
@CODE
Program Name(Parameters);
@END_CODE

Figure 3. *A Sample PDF Template for a Program with Three Inputs and Two Outputs.*

only site of choice. Accordingly, NetSolve will select our GSC and direct that we run the hardware implementation for the user.

Through the use of multiple codes and their corresponding PDFs, we seek to provide a measure of robustness, flexibility and redundancy. Additionally, we hope to make available an environment conducive to experimentation and testing. For example, how large must an input file be before one FFT is superior to another? The answer to this question may vary from user to user, and can depend on a number of factors including communications bandwidth, middleware overhead and relative processor speeds. Thus, in this situation, the user may wish to experiment with an assortment of test data sets to determine the threshold at which a local FFT (if one is available) is outpaced by a NetSolve software FFT and when, in turn, a NetSolve software FFT is outpaced by a NetSolve hardware FFT.

5 Preliminary Results

Our new boards have just come on line as this paper goes to press. Installing a system as novel as the Pilchard has not gone without encountering a few expected (yet still seemingly capricious) obstacles. Nevertheless, we have been able to conduct limited preliminary testing using freshly-minted Pilchard codes for both the FFT and the Data En-

cryption Standard (DES).

As file sizes grow, hardware-accelerated solutions have predictably been faster than software-only implementations, with FFT benefiting more than DES from the use of FPGAs. Configuration file loading and data transfer rates appear to have dramatic effects. These and other variables result in a wide variation in speedup factors. In the case of DES, hardware has sometimes been only a few times faster than software; for the FFT, it has often been a few hundred times faster.

Observed behavior suggests that subfiles sizes, I/O transfers and so forth should be managed carefully in this environment if one desires peak performance. NetSolve overhead looks to be particularly significant unless data set sizes are relatively large. In any event, we have found that the PDF makes it a snap for remote users to gain access to accelerated solutions, even if they are unfamiliar with the temperamental details and idiosyncrasies of hardware implementations.

6 Directions for Ongoing Research

We continue to investigate techniques for exploiting special-purpose hardware in the context of high-performance computational grids. In general, the use of something akin to a grid service cluster augmented with reconfigurable technology appears to be an attractive approach.

Access techniques are less well understood. The NetSolve model seems promising as long as the user needs only to pass I/O parameters and is satisfied with solutions already in residence at some site(s) on the grid. In such a setting, the simple PDF suffices. If code is to be passed, however, then a different model (e.g., Condor) will probably be required. It is easy to see how code passing may make sense for software solutions. On the other hand, it is much less clear how it can be used to achieve hardware acceleration. This is because VHDL, configuration files and the like are in general much more architecture dependent than, say, C routines and object codes.

Timing issues are also in need of study. For example, when should a configuration file be loaded? Ideally, an FPGA should be programmed in advance, that is, preloaded with the proper bitmap before it is called upon to execute. Otherwise, the configuration time can in some cases dwarf the execution time unless the data set to be operated upon is huge. How can preloading be accomplished? Is some form of pipelining helpful? Are analogs of well-known memory paging methods (e.g., the LRU rule) reasonable? Furthermore, in addition to the initial loading a configuration file, one may have to face the problem of dynamic reconfiguration as application environments change in real time.

Partitioning is another problem of significance. This can be required, for example, when an application is too large to fit onto a single FPGA. We have looked at this problem before [8, 14], but new platforms such as the Pilchard system dramatically change the standard view of multi-FPGA topologies, and thus may require radically new partitioning strategies.

Finally, what is a proper means of benchmarking hardware-enhanced clusters? With NetSolve, for example, a series of LINPACK-style benchmarks are provided. Although these are exemplary tools for gauging the relative performance of platforms running software applications, they do not appear to be quite the right type of metric for evaluating the effectiveness of hardware acceleration across the grid.

7 Summary

We believe the timely confluence of reconfigurable and grid technologies gives rise to many appealing opportunities. It also presents several interesting challenges, a number of which we have detailed here. We are enthusiastic about the possibilities opened up by this work. We envision that FPGA-based systems have the potential to play a pivotal role in the long-range success of the grid computing paradigm.

References

[1] M. Beck. IBP, the Internet Backplane Protocol. See http://icl.cs.utk.edu/ibp.

[2] J. Dongarra. The NetSolve middleware package. See http://icl.cs.utk.edu/netsolve.

[3] J. Dongarra, M. W. Berry, M. Beck, J. Gregor, M. A. Langston, T. Moore, J. S. Plank, P. Raghavan, M. G. Thomason, R. C. Ward, and R. M. Wolski. SInRG, a Scalable Intracampus Research Grid. See www.cs.utk.edu/sinrg.

[4] I. Foster and C. Kesselman. The globus project. See www.globus.org.

[5] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Harcourt, 1998.

[6] J. French, A. Grimshaw, M. Humphrey, and W. Wulf. Legion, a worldwide virtual computer. See legion.virginia.edu.

[7] A. G. Geist. The harness parallel virtual machine project. See www.csm.ornl.gov/harness.

[8] M. T. Jones, M. A. Langston, and Padma Raghavan. Tools for mapping applications to CCMs. In *International Conference on Configurable Computing Technology and Applications*, 1998.

[9] J. M. Lehrter. *On a Grid-Based Interface to a Special-Purpose Hardware Cluster*. M.S. Thesis, University of Tennessee, 2002.

[10] P. H. W. Leong, M. P. Leong, O. Y. H. Cheung, T. Tung, C. M. Kwok, M. Y. Wong, and K. H. Lee. Pilchard – a reconfigurable computing platform with memory slot interface. In *IEEE Symposium on Field-Programmable Computing Machines*, 2001.

[11] M. Livny and M. Solomon. Condor, a high throughput computing system. See www.cs.wisc.edu/condor.

[12] R. Murgai, Y. Nishizaki, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Logic synthesis for programmable gate arrays. In *ACM/IEEE Design Automation Conference*, 1990.

[13] NSF. The distributed terascale facility. See www.nsf.gov/od/lpa/news/press/01/pr0167.htm.

[14] S. Ong, N. Kerkiz, B. Srijanto, C. Tan, M. A. Langston, D. F. Newport, and D. W. Bouldin. Automatic mapping of multiple applications to multiple adaptive computing systems. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2001.