LUCITE TECHNICAL TASK ORDER

LOCAL UNIVERSITY CONTRACT FOR INFORMATION TECHNOLOGY EXCHANGE

TITLE: Productive Use of Distributed Reconfigurable Computing Resources

GOVERNMENT PROJECT LEADER (S): Alan Hunsberger

Phone Number: 301-688-0245
Fax Number: 301-688- 0467
internet address: awhunsb@afterlife.ncsc.mil

TASK NO: **(**filled by L309)

DATE: **November 21, 2000**

(To be completed by proposal writers.)

UNIVERSITY PROJECT LEADER (S): **Tarek El-Ghazawi**
Phone Number: **703 993 3610**
Fax Number: **703 993 1980**
internet address: **tarek@gmu.edu**

Nikitas Alexandridis
Phone Number: **202 994-0523**
Fax Number: **202 994-0227**
internet address: **alexan@seas.gwu.edu**

UNIVERSITY (IES):
George Mason University (GMU)
George Washington University (GWU)
University of Southern California/Institute of Information Sciences (USC/ISI)

PRINCIPAL INVESTIGATOR (S): **Krzysztof Gaj**
Phone Number: 703 993 1575
Fax Number: 703 993-1601
internet address: kgaj@gmu.edu

PRINCIPAL INVESTIGATOR (S): **Brian Schott**
Phone Number: 703 812 3722
Fax Number:
internet address: bshott@east.isi.edu

OBJECTIVE: The long-term goal of the reconfigurable computing efforts at Lucite universities is to establish a local university expertise in this technology. The goal of this initial task is to add distributed processing tools to the current prototype reconfigurable computing platform at NSA.

TASK DESCRIPTION:

Background:

1. Reconfigurable computing systems are a new class of computing systems based on a network of hosts with attached field programmable gate array (FPGA) boards used as performance accelerators. The DARPA Adaptive Computing Systems (ACS) Program has been funding research in this area for several years and has produced prototype hardware systems and the software tools for using them. One approach that is being developed comes from the University of Southern California Information Sciences Institute under their System-Level Applications of Adaptive Computing (SLAAC) effort. In addition to USC/ISI, the SLAAC team includes Brigham Young University, Virginia Tech, Los Alamos National Labs, and others. This team has created and demonstrated a system-level methodology for heterogeneous adaptive computing and has defined a prototype reconfigurable computing system, the SLAAC Research Reference Platform (RRP), based on a cluster of FPGA-accelerated workstations with a scalable API and runtime software to support applications. A typical workstation in the SLAAC RRP is a PC running Linux and/or the Windows NT operating system. The FPGA accelerators attached to the workstations can be commercial boards, general-purpose research boards, or special purpose research boards. Boards currently in the RRP environment include the WildForce™ and WildStar™ boards from Annapolis Micro Systems and the SLAAC-1 and SLAAC-1V PCI boards developed by the SLAAC team. Other boards that may be added to the RRP environment include the Context Switching Reconfigurable Computing RCM board developed by Sanders Corp. and the RCA2 board follow-on developed by Los Alamos National Labs. Typically, the accelerator boards contain various combinations of devices such as microprocessor, memory, and field programmable gate array (FPGA) components. Applications are typically programmed for the accelerator boards using software compilers (e.g., prototype C or JAVA compilers) and electronic design automation tools (e.g., VHDL synthesis tools and FPGA place and route tools). Brigham Young Univ. is developing a Java-based hardware description language, JHDL, that currently supports WildForce™ and SLAAC-1. Virginia Tech is developing a high-level API for the RRP that extends the board-level APIs to fit into the RRP model. The RRP uses Myrinet networking products as the high-speed communications backbone.

2. RES is a simple scheduler developed at the IDA Center for Computing Sciences to allow users to distribute computations efficiently across unused network resources. RES is particularly engineered to target computations that can be partitioned into chunks of relatively independent work that require little or no inter-process communication. RES is currently coded to run on SUN (SunOS and Solaris), IBM RS600, SGI, HP, and DEC Alpha workstations. Some comparisons between the RES environment and the RRP environment might be useful. RES runs each process independently on a single host. There is no communication between RES processes. Each RES host also supports interactive operation with a local user. If the user needs the host, then the RES process is terminated and has to be restarted from scratch (RES was developed with the goal of not interfering with the normal work of a host to support an interactive user but RES may be updated to provide for checkpointing). RRP processes can run on several hosts with

significant communication among them; the RRP hosts are not generally set up for interactive operation with users (there is one master host that does interact with the user).

3. There are several other job schedulers available that are similar to RES. These include: the Portable Batch System (PBS) developed by Veridian Systems; MOSIX, a software package designed to enhance the Linux kernel with cluster computing capabilities; and LEGION, an enterprise solution developed by Applied MetaComputing and the University of Virginia. All of these have different capabilities, and none of them were developed with a view toward their use on reconfigurable computing systems.

Objective:

The long-term goal of the reconfigurable computing efforts at Lucite universities is to establish a local university expertise in this technology. The goal of this initial task is to add a scheduler/distributed processing tool to the current prototype reconfigurable computing platform at NSA. It is expected that this task will involve close collaboration of the proposer with USC/ISI and/or Virginia Tech. It is also expected that a proposer will have a continuing interest in follow-on reconfigurable computing tasks and will develop appropriate interactions with the DARPA ACS community. The task "Productive Use of Distributed Reconfigurable Computing Resources" consists of four parts:

Part 1 – An initial review of several job schedulers will be conducted to determine which is the best choice to continue with as the baseline for Parts 2, 3 and 4. The review should include RES, PBS, MOSIX and LEGION at a minimum. The output of the review will be a report summarizing and comparing the capabilities of each scheduler, a recommendation as to which one to choose as the baseline, and an explanation of the rationale for the recommendation. The recommendation should be based on considerations such as: ease of adaptability of the code to the reconfigurable computing platform; expected performance of the final scheduler; quality, reliability and maintainability of the final scheduler; and cost. The report should also explain the approach that would be taken in the follow-on work using the recommended baseline and should contrast this with approaches that would be needed under the other schedulers reviewed.

Part 2 – Adapt the baseline scheduler to run under Linux (e.g., RedHat Linux 6.1 or above) on a network of PCs, SUN workstations, and DEC Alpha workstations with attached FPGA boards. Allow a process to run on the FPGA accelerator board while a user interacts with the host workstation. Pick one available FPGA board as the initial target, but be able to continue the task with other boards to be specified.

Part 3 – Adapt the baseline scheduler as modified in Part 2 to interact with and distribute processes to a SLAAC RRP attached to the same network. The goal of this part is to have the scheduler interface with the SLAAC API to co-manage the RRP environment.

Part 4 - Demonstrate both extensions by running appropriate applications including a TBD set of the NSA high performance computing benchmarks.

Proposal Evaluation Criteria:

The following criteria will be used to evaluate proposals:

- technical expertise of the principal investigators and assistants (biographies should be included in the proposal),
- technical performance plan,
- proposed level-of-effort, cost (including hardware and software purchases), and schedule, and
- proposed collaborations.

References:

1. DARPA ACS – http://www.darpa.mil/ito/research/acs/index.html

2. SLAAC

   USC/ISI – http://www.east.isi.edu/
   BYU – http://splish.ee.byu.edu
   VT – http://www.ee.vt.edu/~ccm & http://www.ee.vt.edu/~ccm/vtbyu
   LANL – http://www.lanl.gov/rcc

3. Boards and Networks

   Annapolis Micro Systems – http://www.annapmicro.com/
   Sanders - http://www.sanders.com/CSRC/
   Myrinet – http://www.myri.com/

4. Schedulers

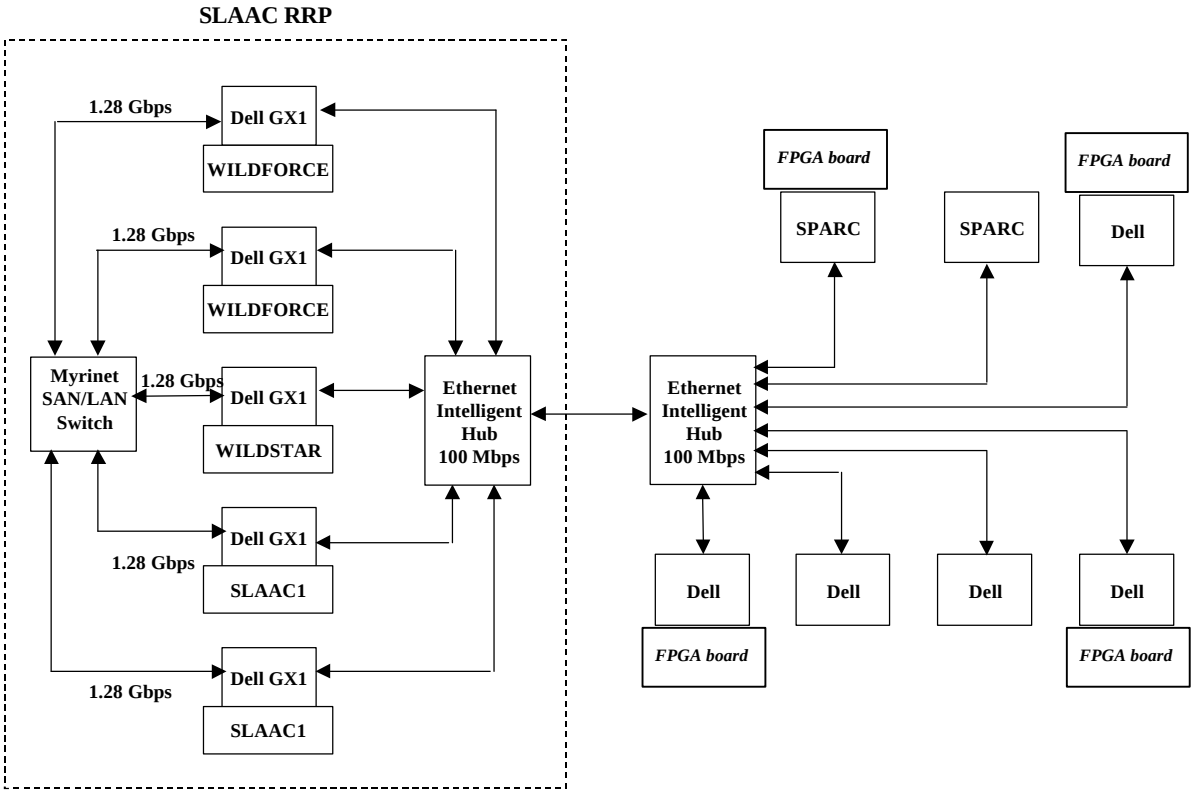
   RES: "RES: A Simple System for Distributed Computing", William W. Carlson, Institute for Defense Analyses/Center for Computing Sciences, May 28 1992

   PBS: see e.g., http://pbs.mrj.com/

   MOSIX: see e.g., http://www.cnds.jhu.edu/mirrors/mosix/

   LEGION: http://www.appliedmeta.com/

5. An Experimental Reconfigurable Computer



TOTAL FY2001 FUNDING: not to exceed $150K
Depending on results and presentations of this phase, follow-on options and additional funding may follow.

PROPOSAL DUE DATE: October 13, 2000/12-noon
TECHNICAL REVIEW COMPLETED: October 25, 2000

START DATE: November 28, 2000
COMPLETION DATE: September 30, 2001

GOVT TO PROVIDE:

- introductions to the DARPA ACS community,
- RES code, if needed (either from the Internet or possibly operational code), and

- high-level descriptions and general C code for the selected high performance computing benchmarks.

Deliverables:

Technical Deliverables

All technical work will be done in a professional manner. Technical reports and presentations will be provided in hardcopy and electronic form (MS Word or PowerPoint as appropriate). Software source code will be high quality, well documented, and delivered in an appropriate electronic format. The technical deliverables are:

- the recommendation report from Part 1 (delivered within 60 days of contract award),
- final source code and documentation (including users manuals),
- high-level descriptions and general C code for the selected high performance computing benchmarks, and
– demonstrations(s) of the final results of the effort.

Administrative Deliverables

All reports and presentations will be provided in hardcopy and electronic form (MS Word or PowerPoint as appropriate):

- quarterly progress reports (a few pages) describing developments, work completed, and funds expended during the period,
- recommendation report review meeting (to be held within 15 days of delivery of the Part 1 report),
- 6-month technical review meeting detailing developments, work completed, and funds expended on the task, and
- final review meeting detailing all developments, work completed, and funds expended on the task.

Depending on the results and presentation of this four-part phase, follow-on options may be supported.

This section completed by faculty Project Leader(s) (from a LUCITE university)

Executive Summary

In this research proposal we contemplate a joint effort by the George Mason University (GMU), George Washington University (GWU), and the University of Southern California Information Sciences Institute (ISI/USC) to investigate and develop an advanced system for harnessing the unused cycles of networked reconfigurable computing resources.

Reconfigurable computing combines the power of custom hardware and programmability. It is known to excel in applications that can benefit from fine-grain parallelism, such as image and signal processing. The advances in reconfigurable computing, has led to a substantial decrease in their costs and, therefore, rise in the number of the systems that are available on local and wide-area networks. In addition, it has led to research teams that are distributed over large geographic area, as it is the case with the SLAAC team and the DARPA ACS community. Therefore, effective utilization of reconfigurable computing resources over LANs for a single organization or for national research groups over the internet is a timely problem that must be addressed and solved.

Over the past decade, many research efforts have tackled the problem of harnessing networked conventional computers and supercomputers either to enable the execution of large scale problems over geographically dispersed resources, or to harness the idle cycles of computers in a lab to benefit research scientists who, otherwise, would not have access to significant computational power. The initiatives ranged from those by individual researchers or organizations, such the case with RES, Condor, and more recently MOSIX, to more strategic initiatives at the Federal level, such as the DARPA metacomputing program, or more recently the GRID computing initiative led by the DoE and NASA.

In this effort, we tackle the problem of effective utilization of networked reconfigurble computers as follows. We propose to start with a comprehensive comparison of the schedulers that can be potentially adopted for this effort. These will include the core set RES, PBS, MOSIX, LFS, and LEGION. In addition to these schedulers, we also consider many other middleware systems that can be leveraged from other research efforts including Condor, from the University of Wisconsin, and Globus, which is developed by Argone National Labs as a part of the Grid Computing initiative. The conceptual comparison will address the presence and the extent of the implementation of key features such as scalability, load balancing, job migration, fault-tolerance, and security. A selected set of these middleware systems will be adapted to Linux. Already many of them support Linux or at least Unix based systems. Porting these systems to Linux will bring this effort one step closer to its target, and will enable an experimental quantitative comparison study of this smaller group of candidate systems. In this step we will subject selected middleware systems to a workload based on the NSA benchmarks on Beowulf machine, used to emulate networked resources scenarios, and measure performance related metrics such as average response time, average turn around, throughput, and scheduling software overhead. A final selection of one system will be made at that point and this system will be extended to consider reconfigurable computing resources, with SLAAC-1, SLAAC-1V,

WILDFORCE™ and WILDSTAR™ cards in mind. The selected system will be integrated with the SLAAC API and extended to all boards supported by the SLAAC software system. Furthermore, scheduling strategies will be extended to allow this system and SLAAC to co-manage reconfgurable resources including the SLAAC research reference platforms. Benchmarking and demonstration using selected NSA benchmarks and realistic loading scenarios will be conducted. Results will be compared with those obtained from conventional computers and supercomputers. GMU has already many of these applications implemented in C and in C with MPI.

1. INTRODUCTION

Reconfigurable computing exploits high degrees of parallelism and bit-level optimizations found in custom hardware implementations, yet retains the aspects of a general purpose programmable computer in that applications can be downloaded into FPGA devices on the order of tens of milliseconds. Although many early reconfigurable computers were expensive complicated monolithic FPGA arrays, most modern commercial and research systems have evolved into relatively less expensive PCI workstation accelerators. Therefore, the number of reconfigurable computing resources available on computer networks has been growing rapidly in recent years. However, these systems are still expensive compared to commodity workstations, and it is important to try to maximize their utilization.

The use of reconfigurable computers as accelerators did not only enable their availability over computer networks, but also created the potential for leveraging currently available distributed computing systems middleware to work with reconfigurable computers. Of particular interest to this work are middleware systems that can harness the unused cycles of networked reconfigurable resources in an organization to benefit more scientists, who are not the primary users of those systems. It can also allow scientists to amass larger number of such resources to solve large problems. Research for such middleware dates back to the early 1990's and among such systems that have emerged and have been used successfully over the past decade are RES and CONDOR, and more recently MOSIX, to mention a few.

Luckily, the past couple of years have reenergized the advancement of such systems with the emergence of the Grid Computing initiative. This effort has resulted in a wealth of new and renovated systems such as Legion, Globus, NetSolve, and CONDOR-G, along with complementary software libraries such as MPI-G. These efforts make our proposed work very timely for two reasons. First, it has dramatically increased the knowledge base about such systems and provided many new advances in this direction. Secondly, since Grid Computing targets the Internet in the first place, security of such systems became a much more important issue. Therefore, more work and solutions in this area are now available, which would be of interest to NSA. In fact, the DoE is now considering a Kerberos enabled version of Globus, for use over the Accelerated Strategic Initiative (ASCI) resources in the geographically distributed

DoE labs. While this may not be an immediate target for this proposed work, such capabilities will be needed in the future to address the needs of geographically distributed research teams, such as the SLAAC project team members, where it would be of interest for each of those members to access the reconfigurable computing resources of the others.

In this research proposal we consider extending one of the distributed computing systems described in order to be able to work with reconfigurable computing resources. RES seems to be very attractive due to its simplicity. However, with the abundant developments and changes in this field over the past decade and the emerging Grid Computing technologies, all these systems must be examined and compared methodically. This comparison must be conducted against a set of well thought out design requirements based on application needs and using measurable metrics. Therefore, these comparisons will be conducted both conceptually and experimentally. The most adequate system or combination of systems will be adopted and extended. Those extensions will also include interoperating with the SLAAC system in order to provide an overall architecture which exhibits sufficient generality and abstraction that simplify the incorporation and management of new reconfigurable resources, including the SLAAC RRP, while allowing the integration of new upgrades of the underlying distributed computing system.

Demonstration of the final design will be conducted on an experimental platform at GMU using selected NSA benchmarks, under realistic loading scenarios, put together into representative experiments that represent the actual working environment. Specifically designed performance metrics will be developed and used to provide a measurable understanding of the system behavior, exposing strengths as well as areas for further improvements.

In order to accomplish the described work, the following four tasks (parts) will be performed in the order given.

2. PROJECT DESCRIPTION

PART 1: COMPARATIVE STUDY OF RELATED DISTRIBUTED COMPUTING SYSTEMS

A. Conceptual Comparative Study

With the progress in distributed computing, from the work on networks of workstations and clusters of workstations, at UCB and UWM, to Grid Computing, many middleware systems have been developed over the years to facilitate application deployment over networked resources. These systems, exhibit a very wide range of functionalities and characteristics. Furthermore, some of such systems could provide the same functionalities, but use different design principles. In some instances, the differences between these systems are quite obvious, but often such differences are subtle. For example, Globus and Legion are two distributed computing (GRID) systems that have much similar functionality, but follow different design principles where Legion is object oriented and Globus is not. The ways these systems are even defined by their designers tend to blur the differences between them when they are considered at the first glance. For instance, Globus is defined as a grid computing tool kit, while Legion is defined as a distributed operating system, thus, while they do more or less similar things. Therefore,

	Scalability	Load balancing	Fault Tolerance	Security	Information Services	Remote I/O	Low/High Overhead	Type of design
RES								
PBS								
Legion								
Mosix								
LSF								
Condor								
Globus								
NetSolve								
NWS								
Computing Portals								
APST								
Compaq DCE								

Table 1. An Example Feature Comparison of Middleware Systems

these systems must be differentiated and gauged using a consistent set of generic definitions and metrics. These differences must be determined with respect to: 1) features, 2) architecture, and 3) performance. In this task this will be accomplished through the thorough analyses of the concepts behind each of the considered systems and its characteristics. Table 1 gives an example of the systems and features that can be considered in this study. **Many comparison features in addition to those shown in table 1 will be also considered. Those include Ease-of-Adapting the middleware to reconfigurable computing, Ease-of-Use of the middleware, and**

Maintainability of the code. Table 2 gives a quick idea of how systems will be architecturally compared. Prior to the architectural comparisons, a generic reference model for a layered middleware system architecture will be constructed to guide the comparisons.

	Distribution of Control	Design Type	Operating Systems	Communicat-ion Architecture
RES				
PBS				
Legion				
Mosix				
LSF				
Condor				
Globus				
NetSolve				
NWS				
Computing Portals				
APST				
Compaq DCE				

Table 2. Example of Architecture Comparison Attributes of Middleware

The initial list of systems to be considered in this study is shown in Table 1 and Table 2. Many of these systems may have features beyond what is needed, at least initially, such as Globus and Legion, or may have fewer features than needed, such as APST. However, we will remain open to examining each of them thoroughly to determine which one is worth extending and whether a combination of more than one tool should be utilized. The initial list includes RES from NSA, AppLes from the Grid Computing Laboratory at UCSD, Condor from University of Wisconsin, Globus from ANL, Legion from UVA, Load Sharing Facility (LSF), MOSIX, NetSolve, Network Whether Service (NWS), PBS from Viridian and Computing Portals from Indiana U.. Dr. El-Ghazawi has had discussions with some of the leaders of these efforts to secure access to the source code of their systems. Among those that he discussed the subject with, at a recent Grid computing meeting at NASA Ames Research Center, are Dr. Bill Nitzberg, director of engineering at Viridian, Professor Jack Dongarra and Professor Dennis Gannon. Dr. Nitzberg, e.g., has agreed to furnish our team with the source code of the professional version of PBS free of charge, which has a lot more features than the open source version.

B. An Experimental Performance Study

This study overlaps Task 1 and Task 2, as it may imply porting selected systems to Linux. Very often, a design with an extensive set of noble features end up with undesirable performance behaviors. Therefore, before making any commitments to which system to select based on the

feature and architectural study, one should, when feasible, study the performance of candidate systems experimentally under workloads emulating real-life scenarios. This can be accomplished by developing a number of NSA benchmarking applications both sequentially in C and in parallel with C plus MPI. Using a controlled environment, such as a Beowulf System, instances of these applications can be generated and run on partitions of the Beowulf system, to emulate typical loading and operational scenarios, under the middleware to be tested. Appropriate performance instrumentation tools will be then used to monitor and measure key performance parameters.

In our case, it turns out that such study is feasible for the following reasons: 1) many of the cited systems support Linux already [e.g., PBS, Condor, Mosix support Linux, to name a few; and even those that do not support Linux, support some other UNIX operating system], 2) Many of the cited systems will be eliminated based on the conceptual study, 3) GMU has an extensive experience with the NSA benchmark high-performance computing suite and has already implemented a number of its application programs in C and MPI (in addition to UPC) under the UPC effort, 4) GMU has extensive expertise with Beowulf systems as well as how measure their performance and has access to 3 of these systems, all of which are operated under Dr. El-Ghazawi's supervision.

Examples of the performance metrics that can be measured are:

Average response time- how long it takes before a request start to execute on the average. This will be measured both in time units as well as a ratio to the execution time length of the task.

Average turn around time – is the average time from submitting a job till completing it. Will be measured in absolute time units as well as a ratio to the execution time.

Utilization - average busy time span to available time span. Time span is measured in time*resource units.

Throughput- Overall amount of work performed per unit time, e.g., MFLOPS or applications per second.

Load imbalance- this will be measured as the variability in the busy fraction of the time of the different resources.

Scheduling overhead- this will be determined by measuring the time to complete an application scenario under the scheduling middleware and the time to run the same schedule of activities without the scheduler. The percentage increase in time due to the scheduler will be the overhead.

Fault tolerance- this will be measured by emulating failures at some of the Beowulf nodes and measuring the effect on the throughput, average response time, utilization, and imbalance.

Based on this study, one system or a combination of a few systems will be selected to extend for reconfigurable computing resources.

PART 2: ADOPTING THE BASE SYSTEM TO RECONFIGURABLE COMPUTING

The selected job scheduler will be initially adapted to run under one operating system, Linux, with one type of the accelerator board, e.g. SLAAC-1V PCI. Future extensions to other operating systems (NT, Solaris), and other types of boards will be investigated. For example, the SLAAC API developed at USC ISI and Virginia Tech currently support commercial Annapolis Microsystems WildForce™ and WildStar™ boards, and research boards such as the Sanders context-switching CSRC board. It is worth noting here that some of Dr. El-Ghazawi's students have previously participated under summer internships at ISI/USC in porting the SLAAC API for some of these boards, WildForce™ and WildOne™, from NT to Linux.

The following components of the distributed computing system may need to be extended with additional features and parameters to accommodate the use of the FPGA-based accelerators:

a. execution host daemon processing
b. user interface
c. scheduling.

2.1 Execution host daemon processing

An execution host is the place where the accelerator boards reside, therefore, the most substantial and extensive changes will need to be made to the control software operating on the execution workstations.

The role of this control software, typically executed as a demon process, is to

a. gather statistics about the current state of the execution host;
b. communicate a status of the host, including the list of currently available local resources to the scheduler;
c. activate new jobs assigned to the given execution host by the scheduler;
d. monitor the local users' activity and determine whether jobs assigned by the scheduler may continue, should be suspended or killed, or possibly should be resumed after earlier suspension.

Gathering statistics about the current state of the *execution host* needs to be substantially extended. It will be divided into two major tasks:

a. gathering information about the utilization of CPU, physical memory, virtual memory, and input/output for processes running on the workstation itself;
b. gathering information about the utilization of resources of the accelerator boards, such as the number of boards in use, number of FPGA devices used on each board, utilization of each of the FPGA devices, amount of the on-board memory in use, the amount of communication between the board and the application program.

The former task can be achieved without any changes to the daemon software. Gathering this kind of data is typically supported by the operating system. For example, under Unix, the information about the workstation resources utilization is collected by the local statistics daemon and can be easily acquired by the daemon associated with the scheduler.

The latter task must be implemented from scratch. The standard SLAAC API will be used to support collecting this information. The amount and type of information collected by the execution host daemon will be chosen based on the detailed analysis of the capabilities of the SLAAC APIs and the influence of the knowledge of the given parameter on the performance of the baseline scheduler.
The following information should be made available to the daemon in the form of configuration tables,

a. list of boards installed on the execution host, together with their detailed characteristics;
b. list of the host applications that can utilize the boards;
c. the correspondence between each application and the boards it can run on.
d. provide accounting for the users usage

These tables should be updated any time a new board or a new application is installed on the host.
When the execution host daemon is started, it should transfer all the above information to the scheduler. During the regular operation, the daemon will need to keep track which applications are currently running, which boards are occupied, and for how long any board or application was idle.

Detecting local users' attempt to use accelerator boards is another important task of the daemon. The local users should have priority over the tasks assigned by the scheduler. As a result, whenever a local user is starting an application that attempts to make use of an accelerator board, this action should be detected by the daemon. Depending on the set of parameters, the daemon will either kill the application that is currently using the board, or suspend its execution. The suspension of the application would need to be followed by storing the entire context of the board, including its configuration and internal state of all registers and memories, so this context can be restored in the future. We will investigate the feasibility of such context saving, taking into account the capabilities of available accelerator boards and functions provided by SLAAC API.

In case, saving the board context appeared feasible, it would give a good foundation for checkpointing, i.e., temporary storage of the board's internal state for the purpose of fault tolerance. If the board, host application, or workstation crashed, the checkpointing would prevent against loosing the results of all earlier computations.

Our initial assumption will be that each board is assigned entirely to a single application. In the future, we can consider sharing the boards among multiple applications.

On the other hand, we will allow two applications to run simultaneously on the same execution workstation, under assumption that none or only one of these applications is using an accelerator board, and the application started by the local user has a higher priority than the application started remotely by the scheduler.

2.2 User interface

Changes in the user interface will be much less extensive compared to changes in the daemon controlling the execution host.
The primary modifications will be aimed at providing a web-based interface which allows a user to specify the following new parameters in the request to the scheduler:

a. list of names of applications able to perform a requested job;
b. types of boards these applications are able to run at;

c. estimated execution time, assuming the use of aforementioned boards;
d. estimated utilization of the board.

Additionally, the other standard parameters could be specified by the user, including: physical and virtual memory size, number of input/output operations, working directory, retry policy in case of the program execution failure, etc.

2.3 Scheduling

Assuming the correct choice of the baseline scheduler system, only minor changes will be required in the scheduling unit, responsible for controlling the distribution of tasks among the available computing resources.

The following extensions might be analyzed as an introduction to future implementations:

a. Interactive vs. batch scheduling

The operation of the scheduler might be much more efficient and fair if the scheduler knows in advance the entire set of jobs for a given period of time. This situation often happens in real life, when after regular office hours and during the weekend, only earlier specified batch jobs are being executed. The objective of our extension would be to give users the capability to submit jobs directly to the scheduler for execution in the batch mode, with the requirement to have the job completed by the specified period of time. The scheduler would make the optimum decision about the distribution of resources for the entire period of time, in such a way to meet timing requirements of the largest group of users.

b. User priorities

Typically, job schedulers work under the policy that all users have the same priority and should be given a possibly equal access to computing resources. We will investigate changes in the scheduling algorithms resulting from assigning different priorities to groups of users.

c. Security

A large number of existing job schedulers have very limited security mechanisms that might not be sufficient for a given application.

The goal of the security services in the job scheduler would be to:

a. protect against unauthorized use of computing resources by third parties;
b. protect against unauthorized access to any files created on the execution host;
c. provide authentication and confidentiality to all messages exchanged among users, scheduler, and execution hosts.

All these tasks should be implemented in a way that, on one hand, provides a high level of security, but on the other hand is easy to use and does not impose any substantial performance penalty.

The baseline scheduler will be analyzed in order to determine to what extent it fulfills these security requirements.

d. Migration

In case the selected job scheduler supports job migration, an effort will be made to extend this capability to computational jobs running on the reconfigurable boards.

PART 3: MANAGING SLAAC RRP CLUSTERS

In addition to the single-processor single-accelerator per job programming model as described above, some DoD applications require more logic and/or memory than a single FPGA accelerator board can provide, and consequently require multiple boards to operate simultaneously. The SLAAC ACS API facilitates this by allowing multiple FPGA boards to work data-synchronously using FIFO channels over a local system bus or high-speed network. The Research Reference Platform (RRP) shown at right is a cluster of FPGA-accelerated workstations on a Myrinet gigabit network. The FPGA boards used in the RRP include SLAAC-1, SLAAC-1V, WildForce™, and WildStar™. The entire system is controlled by a single host program, using the SLAAC API. This task will investigate techniques for managing and scheduling jobs on these cooperative clusters. From the point of view of the naive job scheduler, the entire cluster is visible as a single node. The only difference between this cluster node and regular nodes based on a single accelerator board is a substantially greater computational power. However, the big difference between computational resources of RRP and other FPGA-accelerated workstations may cause additional challenges in job scheduling. For example, the RRP is naturally very well suited for large time-consuming jobs, but may be inappropriate for small tasks. Small tasks would incur the same setup and task management overhead as large tasks yet underutilize the compute resources available using the naive approach. A more efficient approach is to partition the cluster as needed for the available tasks. We will investigate the feasibility and performance characteristics of sharing RRP among a large set of small independent jobs.

Scheduling tasks on cooperative computational clusters brings a number of challenges in addition to the benefits. Introducing RRP as a computational node, allows the users to execute very complex tasks reserved traditionally for supercomputers. These applications often exploit massive amounts of parallelism inherent in these systems. In particular, RRP supports inter-process communication, based on MPI. Cluster applications may have complex data movement patterns among individual nodes, where I/O performance can have a significant effect on overall application performance. Unforeseen performance degradation can occur unless I/O is taken into account during task scheduling. The daemon operation will be substantially more complicated in the case of the RRP, because of the extended difficulty of gathering statistical information about all internal nodes, and the difficulty of monitoring behavior of multiple local users. However, these techniques will significantly increase the utilization of RRP-like cluster resources in comparison to statically scheduled clusters. Job migration might be particularly useful in this environment, where small tasks could migrate into the unutilized nodes of large managed clusters

PART4: BENCHMARKING AND DEMONSTRATION

A. Experimental Testbed

In order to test and demonstrate the operation of our system, upon development, we propose to construct a reconfigurable computing testbed at the GMU High-Performance Computing Lab at the School of Computational Sciences for the use by all team members. The lab has already two Linux based clusters and some reconfigurable computing resources, including the WildForce™ and WildOne™, and relevant software development tools. These capabilities will be augmented with SLAAC-1 and SLAAC-1V boards that are to be provided to GMU at no cost to the project. Funding from this project will be used to add a few boards and Myrinet™ capabilities to create a testbed, which is composed of a SLAAC RRP, and workstations equipped with FPGA accelerators.

B. Benchmarking Applications

As per the LUCITE Task Order, NSA high-performance computing benchmark applications will be selected to demonstrate the system and its concepts. Our team has extensive experience with many of these benchmarks. As a part of the NSA UPC project, Dr. El-Ghazawi and his students have implemented many of these benchmarks, such as Nqueens, FFT, and Radix Sorting sequentially and in parallel using both UPC and C with MPI. This will also allow the understanding of the relative performance of supercomputers and FPGA based reconfigurable systems. Furthermore, Dr. El-Ghazawi's students have previously developed image processing applications for the WildForce™ and WildOne™, while Dr. Gaj's students developed decryption applications.

C. Experiments and Demonstrations

Using the aforementioned testbed and the extended middleware system, the selected applications will be launched under real-life-like operational scenarios to demonstrate the functionality of the system. Performance metrics such as those in Task 1 will be used to monitor and expose any areas of the design that need improvement and assess the real value of this system in a quantitative manner.

References

[AppLes] Grid Computing Laboratory (AppLes) http://apples.ucsd.edu/

[Condor] http://www.cs.wisc.edu/condor/

[Globus] http://www.globus.org/

[Jones99] M. Jones, L. Scharf, J. Scott, C. Twaddle, M. Yaconis, K. Yao, P. Athanas, and B. Schott, char`Implementing an API for Distributed Adaptive Computing Systems, Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, April, 1999.

[Legion] http://www.appliedmeta.com/

[LSF] Load Sharing Facility http://www.platform.com

[Mosix] http://www.cnds.jhu.edu/mirrors/mosix/

[NetSolve] http://www.cs.utk.edu/netsolve/

[NWS] http://nws.npaci.edu/NWS/

[PBS] http://pbs.mrj.com/

[Portal] grid portal collaboration: http://www.ipg.nasa.gov/portals/

[DARPA-ACS] http://www.darpa.mil/ito/research/acs/index.html

[SLAAC-ISI] http://www.east.isi.edu/
[SLAAC-BYU] http://splish.ee.byu.edu
[SLAAC-VT] http://www.ee.vt.edu/~ccm

[SLAAC-VT&BYU] http://www.ee.vt.edu/~ccm/vtbyu
[SLAAC-LANL] http://www.lanl.gov/rcc

[AMI] http://www.annapmicro.com/
[Sanders] http://www.sanders.com/CSRC/
[Myrinet] http://www.myri.com/
[Carlson92] RES: "RES: A Simple System for Distributed Computing", William W. Carlson, Institute for Defense Analyses/Center for Computing Sciences, May 28 1992
[Schott99] B. Schott, S. Crago, R. Parker, L. Carter, C. Chen, J. Czarnaski, M. French, I. Hom, T. Tho, and T. Valenti, char`Reconfigurable Architectures for System-Level Applications of Adaptive Computing,char` submitted to VLSI Design special issue on reconfigurable computing, 1999

3. PROJECT MANAGEMENT, MILESTONES, AND DELIVERABLES

A. Project Management

All experimental work on this project will take place at the George Mason University (GMU) high-performance computing lab (HPCL) at the School of Computational Sciences (SCS), where students from GMU SCS, GMU ECE, and GWU ECE will collaborate under the supervision of the P.I.s. Drs. Alexandridis, El-Ghazawi, Gaj, Mr. Schott. Dr. El-Ghazawi will have the oversight of the whole program, Dr. Gaj will focus on the reconfigurable computing aspects and Dr. Alexandridis will provide directions for the distributed systems matters. Mr. Schott will focus on keeping the project progress consistent with the national adaptive computing systems (ACS) and SLAAC community efforts, and will advice the team on all technical issues that have to do with the SLAAC hardware and the SLAAC API. The GMU P.I.s will meet weekly and the project leaders and P.I.s will meet biweekly to steer the effort. The project leaders and the P.I.s and Mr. Hunsberger will hold a monthly progress meeting.

Described below are the proposed areas of collaboration between the GWU team (Prof. Alexandridis and a GWU student), and the GMU team:
Prof. Alexandridis will participate as a Co-PI to advise the team and help supervise the students in the reconfiguration and partitioning issues.
The GWU team will participate in joint meetings to discuss the progress in the project and provide ideas on enhancing and expanding the effort.
Prof. Alexandridis will also make himself available to participate in any related workshops and conference meetings.
A GWU student will become available to work on the project. He will become part of the joint team, will assist in the efforts of hardware and programming issues, and will be making himself

available to work with the GMU students and participate in the experiments to be performed using GMU's lab facilities.
The GWU student will work with the available prototype hardware systems and the software tools for using them.
Along with the GMU PI, Prof. Alexandridis will be collaborating in preparing the quarterly progress reports and will participate in the technical review meetings
The GWU team will also assist in preparing the final report and will provide suggestions on possible follow-up efforts to be supported by Lucite

B. Project Schedule

11/15/00	Start and procure equipment
12/31/00	Conceptual comparative analysis of systems completed
2/15/01	Compared systems extended to Linux and experimental comparative study completed and system selected
4/1/01	System adapted to support a selected board
5/15/01	System adapted to support two more boards
	Technical Review Meeting
7/15/01	System adapted to support SLAAC RRP
9/15/01	2-3 selected benchmark programs implemented for the boards
10/15/01	System Demonstration
11/14/01	Final Report
	Final Review Meeting

C. Deliverables

Technical reports and presentations will be provided in hardcopy and electronic form (MS Word or PowerPoint as appropriate). Software source code will be well documented, and delivered in an appropriate electronic format. The technical deliverables are:

- the recommendation report from Part 1,
- final source code and documentation (including users manuals),
- high-level descriptions and general C code for the selected high performance computing benchmarks, and
– demonstrations(s) of the final results of the effort.

In addition, the following will be provided:

- quarterly progress reports (a few pages) describing developments, work completed, and funds expended during the period,
- recommendation report review meeting (to be held within 15 days of delivery of the Part 1 report),
- 6-month technical review meeting detailing developments, work completed, and funds expended on the task, and
- final review meeting detailing all developments, work completed, and funds expended on the task.

4. SECOND YEAR OPTION

While the specifics of tasks for a second year can be negotiated, we propose here 3 activities that can constitute the tasks that can be pursued of the second year as a follow up on this first year base effort. These tasks are: A) Extensions and Improvements, B) Incorporating GRID

computing and security infrastructure, and C) Establishing a problem solving environment. All or some of these tasks can be carried out at a negotiated level of detail that would be consistent with the level of effort in that proposal.

Task1: Extensions and Improvements

Based on the lessons learned from the benchmarking activities in year one, many opportunities will arise for improving the base system. Performance evaluations can even demonstrate which areas need further work. Furthermore, additional desirable features that can be feasible to include as they provide generous performance and usability return for moderate effort will be also considered. Examples of the features to be examined for possible incorporation into the base system are: checkpointing and recovery, job migration and load balancing, priority scheduling, improved user interface, and remote data input/output and display.

Task 2: GRID Computing and Security Infrastructure

Improving security is very desirable in such distributed environment. As the number of resources could be quite large, a number of unique problems could arise such as the need for having the users login in into all machines and getting authenticated by all machines explicitly. From a user's perspective, a user should be able to log once and get to be authenticated automatically by all systems that he/she is eligible to use in a session.

In addition, it would be of interest to extend this system to support national research groups, such as the SLAAC team or the ACS community who are geographically dispersed allowing them to use each other's resources and/or collaborate on experiments from far apart places. Therefore, we believe that considering the augmentation of the grid computing concepts is of interest. This can be done by examining the developments in grid computing and investigate the opportunities for leveraging the advances and investments in this initiative.

Task 3: PROBLEM SOLVING ENVIRONMENT

While a system will be adopted to enable users to run their applications on this distributed and shared environment, the user will have to worry about learning a significant number of systems commands and how to use them. In this task, we propose to add an application layer, to turn in into a problem solving environment similar in concept to NetSolve, where application developers can easily understand how to use the system and have a quick turn around for getting work done. In NetSolve, for example, users can get a view similar to that of using Matlab, thus, stay closer to the application and away from the system implementation details.

6. Curriculum Vita'
A.Senior Personnel

TAREK El-GHAZAWI
School of Computational Sciences
George Mason University
Fairfax, VA 22030
(703)993-3610
hpc.gmu.edu/~tarek
tarek@gmu.edu

EDUCATION

Ph.D., Electrical and Computer Engineering, New Mexico State University, May 1988
M.S., Electrical and Computer Engineering, New Mexico State University, May 1984
B.S., Communications and Electronics Engineering, Helwan University, May 1980

PROFESSIONAL EXPERIENCE

George Mason University, *Associate Professor,* 1998-present.
Florida Institute of Technology, Associate Professor, 1997-1998.
George Washington U., Electr. Eng. & Comp. Science Dept, *Assoc. Research Prof.*, 1992-97.
Helwan Univ., Dept Communications & Electronics Eng., (EGYPT), *Assistant Prof.*, 1988-89.
Frostburg State University, Department of Computer Science, *Assistant Professor,* 1989-1990.
USRA/CESDIS, *Visiting Scientist*, High-Performance Computing, 1992-2000
NMSU Electronic Vision Analysis Lab. (EVAL), NM, *Graduate Research Assistant,*1985-88.

PROFESSIONAL SOCIETIES

IEEE Senior Member; ACM Member; Phi Kappa Phi, member.

SELECTED SPONSORED RESEARCH

1. Grid Computing in the NASA Context. RIACS, NASA Ames Research Center, Tarek El-Ghazawi (P.I.). (10/00-8/01).

2. Towards an Efficient Shared Memory Parallel C Standard. National Security Agency. Tarek El-Ghazawi (P.I.) and Guy Robinson (Co-P.I). (1/00-12/01)

3. MAPS: Mathematical Applications with a Parallel-Beowulf System. National Science Foundation. Tarek El-Ghazawi (Co-PI), with James Gentle (P.I.), Estela Blaisten, Rainland Lohner, John Wallin and and Edward Wegman (Co-Pis). (10/99-9/01)

SELECTED PUBLICATIONS

1. T. El-Ghazawi and G. Frieder. Redundant Arrays of Inexpensive Disks (RAID). The Encyclopedia of Computer Science, 4rth Edition. MacMillan Reference. Editors: A. Ralston, E. Reilly, and D. Hemmendiger. 2000.

2. A.Meajil, T.El-Ghazawi, T. Sterling. "Characterizing and Representing Workloads for Parallel Computer Architectures". Journal of Systems Architecture. Vol. 46, No. 1. Elsevier Science, North-Holland. January 2000.

3. Zomaya, T. El-Ghazawi, and O. Frieder. "Parallel and Distributed Computations for Data Mining". IEEE Concurrency. IEEE CS. Vol. 7, No. 4. October-December 1999.

4. S. Alaoui, Frieder, T. El-Ghazawi, Bellaachia, and A. BenSaid. "A parallel genetic algorithm for task mapping on parallel machines". Future Generation Computer Systems. Elsevier Science, North-Holland (in press).

5. S. Nastea, O. Frieder, and T. El-Ghazawi. "Load-Balanced Sparse Matrix-Vector Multiplications on Highly Parallel Computers." Journal of Parallel and Distributed Computing. Vol. 46, Number 2. Academic Press. Novembr, 1997.

6. A.Meajil, T.El-Ghazawi, T. Sterling. "Performance Prediction Based on Workload Similarity". Supercomputer. Vol. 13, August 1997.

7. T. El-Ghazawi, P. Charlemwat, J. Le Moigne. "Wavelet-Based Image Registration on Parallel Computers ", Supercomputing'97, IEEE CS, San Jose, November 1997.

8. M.Berry and T. El-Ghazawi. Parallel Input/Output Characteristics of NASA Science Applications," *International Parallel Processing Symposium (IPPS'96),* Honolulu, IEEE CS Press, April 1996

Kris Gaj
ECE Dept., George Mason University
4400 University Drive, Fairfax, VA 22030
E-mail: kgaj@gmu.edu; Tel.: (703) 993-1575

Education

e. Ph.D. in Electrical Engineering, *with special distinction*
Warsaw University of Technology, Warsaw, Poland
Thesis title: *"Hybrid cryptographic protocols based on the DES and RSA cryptosystems"*

1988 M.Sc. in Electrical Engineering, *with special distinction*
Warsaw University of Technology, Warsaw, Poland

Experience

1998-present: Assistant Professor, Electrical and Computer Engineering, George Mason University, Fairfax, VA

1994-1998: Research Associate, Department of Electrical Engineering, University of Rochester, Rochester, NY

1993: Co-founder and President: Enigma Information Security Systems, Warsaw, Poland

1992-1993: Research Associate: Faculty of Electronics, Institute of Telecommunications, Warsaw University of Technology, Warsaw, Poland

1991: Visiting Scholar, Simon Fraser University, Vancouver, Canada, School of Computing Science

Sponsored research

1999-2001: NSF, *Integrated Laboratory for Design and Testing of Digital Circuits based on Reconfigurable Hardware*, PI

1996-1997: Army Research Office, *Quantum Computing with Superconducting Electronics*, co-PI

1992-1993: Polish Committee of Scientific Research, *Computer System for Protecting Secure Transmission of Information in Computer Networks*, PI

1990-1991: Polish Committee of Scientific Research, *High-Accuracy Design Evaluation System – HADES*, co-PI

Selected publications

1. K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays," submitted to the Cryptographer's Track of RSA Security 2001, available at http://ece.gmu.edu/crypto/publications.htm

f. P. Chodowiec, P. Khuon, and K. Gaj, "Fast implementations of secret-key block ciphers using mixed inner- and outer-round pipelining," submitted to the FPGA'2001.

g. K. Gaj and P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates using Reconfigurable Hardware," *Proc. 3rd Advanced Encryption Standard Conference*, New York, April 2000, available at http://ece.gmu.edu/crypto/publications.htm.

h. K. Gaj, Q. P. Herr, V. Adler, A. Krasniewski, E. G. Friedman, and M. J. Feldman, "Tools for the Computer-Aided Design of Multi-Gigahertz Superconducting Digital Circuits," *IEEE Trans. Appl. Supercond*., vol. 9, 1999, pp. 18-38.

5. K. Gaj, C.-H. Cheah, E. G. Friedman, and M. J. Feldman, "Functional Modeling of RSFQ Circuits Using Verilog HDL," *IEEE Trans. Appl. Supercond.*, vol. 7, 1997, pp. 3151-3154.

6. K. Gaj, Q. P. Herr, V. Adler, D. K. Brock, E. G. Friedman, and M. J. Feldman, “Toward a Systematic Design Methodology for Large Multigigahertz Rapid Single Flux Quantum Circuits,” *IEEE Trans. Appl. Supercond*., vol. 9, 1999, pp. 4591-4606.
7. K. Gaj, E. G. Friedman, and M. J. Feldman, “Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits,” *Journal of VLSI Signal Processing*, vol. 16, 1997, pp. 247-276 (*invited paper*).
8. K. Gaj, *The Enigma Cipher: Methods of Breaking*, Transport and Communications Press, Warsaw, Poland, 1989, 200 pages (submitted to Artech House for publication in English).

B. Sample of Potential GRAs

Burcea Mihai

Date of birth: March the 23rd, 1977

Work and research experience:

- 1996 – won a special award (the 4th place) in the Scientific Research Contest of the Politehnica University of Bucharest, the Department of Mathematics, with a project concerning the decomposition of monoids by using automata.
- June 1999 – September 1999 – Network and System Administrator at SC Radiotel srl, Internet Service Provider; the job included maintaining the network connections, technical and customer support, connecting customers, maintaining the system.
- February 2000 – June 2000 – teaching assistant for "Parallel Computing Architectures" (students in the 4th year of study) in the Computer Science Department. I have taught, among other things, the Pthreads POSIX standard, OpenMP, and OCCAM.
- May 2000 – won 2nd prize in the Scientific Research Contest in the Computer Science Department, with a part of my diploma project, "Simulation of Photon Transport on a Cluster of Workstations using a Monte Carlo Method".
- June 2000 – Diploma Project - "Simulation of Photon Transport on a Cluster of Workstations using a Monte Carlo Method". The application, written in C using the MPI library (MPICH, WMPI and LAMMPI implementations), uses a series of indexes (computed by a physics application, XCOM) to model and simulate the propagation of gamma rays (in particular, photons) through any material that can be described through its chemical formula or percentile composition. The program has been successfully implemented and run on 3 platforms with minimal or no modifications (the only modifications needed were for configuring the MPI): a cluster of 8 Linux boxes, a cluster of 4 PCs running WindowsNT, and a Sparc multiprocessor system. To evaluate the cluster' as well as the program's performance, both from the network point of view and from the MPI point of view, a tool called Vampire was used.
- October 2000 – teaching assistant in "Local Area Networks", "Computer Systems Structure", and "Parallel Computing Architectures" (the last one in the spring term).
- September 1999 – present: started as a Cisco Networking Academy student, and starting with the spring of 2000, I am also a certified instructor for teaching at the local and regional Cisco Networking Academy. This implies teaching Cisco Networking Academy instructors, who in return will be teaching their students.

There have also been several smaller projects during my 5 years in college, but I figure that they were not important enough to be worth mentioning; such as a file archiver, a client – server system that communicated over a network (also using broadcasts), and so on.

Jacek Rafa³ Radzikowski

E-mail: jradziko@gmu.edu

Education

2000-present PhD studies in Information Technology
Department of Electrical and Computer Engineering
George Mason University, Vairfax VA, USA

1997-2000 PhD studies in Information Security
Institute of Telecommunications
Warsaw University of Technology, Warsaw, Poland.

1996 MS in Computer Science, Institute of Control and Computation Engineering
Warsaw University of Technology, Warsaw, Poland.

MS Thesis: A Concept and Implementation of Software for Realistic Visualization of 2D Manifolds from 4D space

Professional and research interests

Information security, programming, computer networks (especially WANs),UNIX, CISCO routers and access servers, designing and programming microcomputer systems.

Operating systems

Solaris (administration), Linux (administration and programming), QNX (administration and programming), CISCO IOS

Programming languages and environments

C, C++, Java, Perl, HTML, Pascal, Object Pascal, Fortran, Prolog, PLC ladder programming language, Basic, Logo.
Assembly languages of Z80, 8051, Atmel AVR, and x86 (basic),Unix shell, applications for X-Windows environment, TCP/IP networking, Linux kernel modules.

Hardware

Z80, Atmel AVRs, 8051, basic knowledge of Abel AHDL, basic knowledge of VHDL, design and implementation of digital circuits, design of PCBs.
CAD tools: Eagle, Orcad.

Foreign languages

English, German (passive), Russian (passive).

Pawel CHODOWIEC

Phone: 385-0248 (home) or 993-1561 (university)
E-mail: pchodowi@gmu.edu

EDUCATION

George Mason University, Electrical and Computer Engineering, Fairfax, USA
PhD in Electrical Engineering (Sep. 2000 - present)

George Mason University, Electrical and Computer Engineering, Fairfax, USA
MS in Computer Engineering (2000)

Warsaw University of Technology, Faculty of Electronics and Information Technology, Warsaw, Poland
BS in Telecommunications (1998)

EXPERIENCE

- Implementation of the Mars, RC6, Rijndael, Serpent and Twofish ciphers in Xilinx FPGA devices
  - Part of the research under supervision of Dr. Kris Gaj leading to comparison of new cryptographic algorithms competing to become a new American Encryption Standard (February 1999 – October 2000).
  - Features:
    - target devices: Xilinx XC4000XL and VIRTEX FPGA families
    - description language: VHDL
    - analyzed different architectures and trade-offs between speed and area

- Project of the hardware encryption system for PC hard drives based on FPGAs
  - Subject of the BS degree project (October 1997 - September 1998)
  - Developed for Enigma - Information Security Systems, Warsaw
  - Features:
    - uses ATA-2 interface
    - based on Altera FLEX10K30 - described in AHDL

PUBLICATIONS

- P. Chodowiec, P. Khuon, and K. Gaj, “Fast implementations of secret-key block ciphers using mixed inner- and outer-round pipelining,” submitted to the FPGA’2001.
- K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays," submitted to the Cryptographer’s Track of RSA Security 2001, available at http://ece.gmu.edu/crypto/publications.htm
- K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays." September 1, 2000

- K.Gaj and P.Chodowiec, “Comparison of the hardware performance of the AES candidates using reconfigurable hardware”, *Third Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000
- P.Chodowiec and K.Gaj, “Implementation of the Twofish Cipher Using FPGA Devices”, Technical Report, George Mason University, July 1999

Alexandru Vlad Staicu

Date of birth: June 1, 1976

EXPERIENCE:

September 1998 – July 1999

- working in R&D department for RADCOM SRL (a private telecommunication company) for the development of an Embedded System for Mobile Unit Tracking
  - Hardware:
    - mobile platform with a Phillips XA controller;
    - GSM, GPS and digital radio modules, communicating through serial lines (RS 232) with the XA controller;
    - LCD and keyboard for user interface;
  - Software:
    - µCOS – a multitasking kernel especially developed for 80C51 micro-controller family;
    - ANSI C and XA assembler as programming languages;
  - I have implemented the MAP27 medium access protocol on the XA platform and I have also developed some of the drivers necessary for the I/O devices mentioned above.

May 2000

- won the second prize in the annual Scientific Research Contest with a project named “Photon Transport Simulation on a Cluster of Workstation Using the Monte Carlo Method”

<u>June 2000</u> – Diploma project

- “Photon Transport Simulation on a Cluster of Workstation Using the Monte Carlo Method”
- Hardware:
  - 4 PC’s (Pentium – 120MHz) running WinNT;
  - 8 PC’s (Pentium I & Pentium II) running Linux RedHat;
  - 1 Cisco LightStream 1010 ATM switch + 4 ATM NIC’s
- Software:
  - MPI (WMPI, LAM MPI, MPICH);
  - Visual C++ 5.0
  - C/UNIX

    - XCOM - automatic interaction constants generator
    - SPRNG – a very good parallel random number generator
    - VAMPIR and other performance profiling tools
- I have participated at the implementation of the Monte Carlo algorithm and at the performance evaluation.

July 2000 – present

- working at RADCOM SRL in a large project: "CORBA platform for Internet access using mobile telephony terminals"
- Software:
    - Java;
    - CORBA (VisiBroker ORB implementation);
    - Jbuilder – as programming environment;
    - WAP – WML (Wireless Application Protocol for mobile Internet access)
- Hardware:
    - PC's running WinNT and Linux and interconnected by Ethernet;
- I have implemented some WAP services that were in fact CORBA server objects;

I am also teaching assistant in "Parallel Computing Architectures", "Computer System Structure" courses and I am Cisco Networking Academy instructor

Sébastien Chauvin

Objectives
To participate in the development of a real time project to know if my today hobby can become a part of my tomorrow job.

Education
2000- present
GMU: Graduate Student in High-Performance Computing
1999-2000
UTBM : University of technology of Belfort Montbeliard (France)
Learning computer science (engineer degree in 2002).
1997-1999
General scientific and technologic studies at UTBM.

Computer skills
C, UPC, MPI, C++, Assembler (saturn, Z80, x86, 68HC11, Cop8, Atmel Avr)

VHDL, Lisp, Pascal, Java, SQL, Visual Basic.

Hardware and software development with microcontrolers (multiple projects).
Driver writing for linux.

Leadership
1999-2000
Vice-president of the UTBM robotic club which participate to a famous cup in France of self-governing robots.
Project manager of electronic and software team (10 students).
Conceived PCB with microcontrolers, analog circuits and HF components.
Developped Real-time application (motor control, communication).
Established partenership with more than 10 companies.

1998-1999
Project manager of electronic and software team in the UTBM robotic club.

Languages
French, English, German

Activities
Road cycling, chess, cooking.

NGUYEN NGUYEN

EDUCATION:

- B.S. EE at University of Maryland at College Park, 12-1997. *Summa Cum Laude.* GPA 3.92/4.0
- M.S. EE expected 12-2000, George Mason University.

EMPLOYMENT:

- 1/98 – Present: **SALIX Technology, Inc.** – **Gaithersburg, MD**. *Staff Engineer*. Worked DS3 Line Interface Unit of the Internet Telephony product SALIX-7750. Wrote functional and design specifications. Performed board and logic designs. Used both VHDL and schematic capture tool to design CPLD and FPGA. Developed embedded software using C/C++ in pSOS operating system environment. Wrote device drivers for various devices on the board. Developed diagnostic software, and worked on the

Announcement Message Service. Developed software utilities to facilitate hardware design process.

- 06/96-09/96 and 01/97-02/97: **Comsat Mobile Communication – Clarksburg, MD**. *Telecommunication Intern*. Performed and analyzed voice and data tests through satellite network. Implemented International Billing System using C++. Generated satellite traffic reports using C and Unix shell scripts.

- 09/95-02/96: **MPR Engineering Associates – Alexandria, VA**. *Engineer Aide*. Checked calculation and verified technical report. Performed some Unix system administration work. Assisted engineers with software development.

HONORS

- UMD University Scholarship.

- Blender Scholarship for Academic Excellence

PROFESSIONAL SOCIETIES

- Student member IEEE Computer and Communications Societies

- Student member of ACM