Integrating Novel Methodologies, Tools, and IT Resources for Graduate Level Courses in High Performance Computing and Advanced Signal Processing Algorithms

Domingo A. Rodriguez[1], Nayda G. Santiago [2]

Abstract - **This work presents an approach at integrating novel methodologies for teaching graduate level courses in the areas of High Performance Computing (HPC) and Advanced Signal Processing Algorithms (ASPA) for Computer Engineering and Computer Science and Engineering Curricula. The novel teaching methodology presented here in High Performance Computing centers on the use of innovative empirical methods, i.e., exploratory data analysis, experiment design, etc., for studying computer performance, whereas an operator signal algebra approach is considered a novel methodology for the studying of Advanced Signal Processing Algorithms. The work also discusses an on going concerted effort at utilizing common tools and IT resources in both courses to provide students a holistic learning experience.**

Index Terms – Computer performance, computational signal processing, empirical methods, system modeling, simulation

INTRODUCTION

This article presents results of ongoing work on developing novel teaching methodologies in the areas of High Performance Computing (HPC) and Advanced Signal Processing Algorithms (ASPA) and the integration of these methodologies, along with teaching developing tools and Information Technology (IT) resources into graduate level courses for the Computer Engineering and Computer Science and Engineering curricula. The paper starts by providing a basic example of how the areas of HPC and ASPA closely interact in many engineering applications, creating in this manner the motivation and rational for addressing the issue of course integration. It continues with description of aspects in the areas HPC and ASPA as they are used in this work and proceeds by describing the novel teaching methodologies introduced in these areas. It then concludes with a set of guidelines of how to integrate developing tools and IT resources to strengthen the academic interaction between these courses and providing some recommendations for teachers.

The Modeling and simulation of engineering applications usually require a large amount of computational effort in order to arrive at solutions that closer resembles actual behavior of targeted physical systems and associated environment. In essence, engineering theoretical modeling can be described as the formulation of a set of mathematical expressions that provide a representation of a given physical system and its interaction with the environment. Engineering computer simulation is actuating the model through digital computing. As system modeling becomes more complex and elaborate in trying to provide a better description of the underlying physical system, the computational effort also tends to increase during the simulation stages. Then, in order to obtain results within reasonable time and available resources, efficient algorithms must be designed and developed to run on high performance digital computers. This is the case for the particular applications of active remote sensing for hydroecological applications; but, it applies to most fields and large scale engineering applications.

TEACHING HIGH PERFORMANCE COMPUTING

Throughout the years of advanced digital computation, the definition of High Performance Computing has been context dependent in the sense that its definition has been very closely influenced by issues such as the state of the arts of digital electronics, computer architectures, and programming languages and models as well as marketing trends and national budgetary policies. This work defines HPC as the computing activities associated with any computational unit or spatial aggregate of computational units capable of producing a computational task result in a period of time which is at least an order of magnitude lower than the time which would require a standard desktop computer to generate the same computational task result.

Students taking HPC have had experience with, both, shared memory systems and the use of OpenMP as well as distributed memory systems and the use of MPI. More involvement on specialized architectures and the Grid is envisioned as one of the evolutionary outcomes. To provide an example of how students normally engage in HPC course, take the case of a computing activity in a shared memory system using OpenMP as the tool for portable shared parallelism and

[1] Domingo A. Rodriguez, Automated Information Processing Group, University of Puerto Rico-Mayagüez, domingo@ece.uprm.edu
[2] Nayda G. Santiago, Parallel and Distributed Processing Group, University of Puerto Rico-Mayagüez, nayda.santiago@ece.uprm.edu

higher-level API. Basically, a student, as the skilled programmer writing an OpenMP engineering application, inserts appropriate OpenMP directives in a designed code for parallel processing and allows the parallelizing compiler to work in order to produce an execution code. The student then enters a process of performance tuning by using OpenMP Profiling tools followed by Performance Analysis Tools as well as the possible modification of OpenMP pragmas, compiler flags, and functional program structures.

TEACHING ADVANCED SIGNAL PROCESSING ALGORITHMS

The Advanced Signal Processing Algorithms (ASPA) course deals with providing students with the necessary skills for the analysis, design, and implementation of signal processing algorithms for large scale applications. A large scale application is defined as one which requires the utilization of high performance computing resources. This work deals with the formulation of computing methods for the action of operators on discrete, finite length signals. These operators are termed here *finite dimensional signal algebra* operators and they play a very important role in many scientific and engineering applications such as multimedia signal processing, digital communications, remote sensing imaging, time series and econometric modeling, channel error correcting codes, biomedical signal processing, and seismic signal processing.

HPC AND ASPA COURSE INTERACTION

The analysis and design of algorithms centers on the use of mathematical techniques to develop computational methods to perform a computational task. A computational method is defined as a structured set of algorithms. Great advances in sensor/effector technology, communications and networking technology, and computing and information processing technology are demanding new theories, methods, and techniques to improve a seamless understanding and interaction with a physical or sensory reality. For instance, better tools are needed for the tasks of storage, manipulation, representation, visualization, and rendering when dealing with very large amounts of signal-based content. A new approach, for instance, at improving on these tasks is to treat the content as signal sets carrying information which needs to be extracted. Treating signals as elements in prescribed sets allows to study structures associated with such sets and to apply operator theoretic methods in a generalized computing and information processing setting. This is the motivation for building a computation and information processing (CIP) environment.

A CIP environment uses operator methods, through algorithms, to assist at improving and enhancing the performance in effecting some of these tasks. Formally, CIP environment deals with the algorithmic treatment of signal-based large scale content in order to extract information relevant and important to a user. In this regard a CIP environment can be thought of as the aggregate of the following seven (7) components: A set of input entities, a set of output entities, a database infrastructure, a set of generalized computation and information processing operators, a set of composition rules for operators, a set of actions rules for operators to act on input entities in order to produce targeted and desired output, and a user interface.

When the HPC course and the ASPA courses are integrated, they complement the learning process by providing students with the skills to undertake large scale problem from both the signal processing perspective and the computational aspects of the problem. From the signal processing point of view, the student develops the computational and programming skills to solve applications where the signal processing theories can be tested. This presents an interesting set of problems to use as projects for ASPA courses such as analysis of radar data, multisensor system analysis, and others. Another aspect to consider is the treatment of computer performance data from the perspective of a signal coming from a source. This opens up a large number of possible venues for research and innovative methods in performance evaluation of HPC systems. From the computational and performance point of view, the ASPA course provides an application testbed where students can carry out performance studies, integrating experimentation concepts, statistics, performance tools, and measurements. New theories and ideas have emerged from the combination of these areas [1].

NOVEL TEACHING METHODOLOGY IN HPC

The new teaching methodology presented in this work centers on emphasizing computer performance as the main issue in HPC and in introducing innovative empirical methods to address automated computer performance measures. In essence, students are taught a methodology for aiding a scientific programmer to evaluate the performance of parallel programs on advanced architectures. The methodology applies well-defined design of experiments methods to the identification of relations among different levels in the process of mapping computational operations to high-performance computing systems. Statistical analysis is used for studying different factors that affect the mapping process of scientific computing algorithms to advanced architectures. The use of statistics for identification of relationships among factors has formalized the solution of the problem and this novel approach allows unbiased conclusions about results. Subset selection based on principal components was used to determine the subset of metrics required to explain the behavior of the system.

Performance data analysis is integral to the process of tuning parallel applications to advanced architectures. The traditional approach for performance tuning is through the process of data collection, analysis, and code optimization. In this approach the application programmer needs to understand instrumentation, learn the appropriate tools, and interpret data and its relation to the code, in order to optimize the code or system configuration, accordingly. This method is complex and prone to wrong interpretations.

Also, transformations applied to source code are hard to map to performance data. An alternative method is proposed that minimizes ambiguity when determining which factors to consider during a tuning process of a parallel application.

A performance problem-solving process starts with the analysis of the problem specification. Information needs to be collected about the programmer's goal and both the performance problem and the application itself. Once the application and performance goals are clear, the next step is to profile the code to identify possible functions to optimize. Analysis continues with the identification of possible factors affecting performance. These include environment factors, algorithms to solve those functions to optimize, and hardware specific factors. Next, a subset of factors is selected for the experiment, considering controllability, feasibility, practicability, and constraints.

The second step in the methodology is experiment specification. The theory of design of experiments (DOE) allow us to take an objective approach in the experimentation process Experimental relationships allow the identification of causality among variables. Studying all possible factors and levels of these factors is an intractable problem. A level refers here to the different possible values of one factor considered in an experiment. In order to obtain the total number of experimental runs, it is necessary to calculate all possible assignment of factors when varying all at a time. Once a decision on the factors and levels is taken, the next step is to select the random order in which the experimental runs will be executed. Randomization is required to avoid the influence of uncontrollable factors in the outcome[2].

The data collection step is the only one determined particularly by the computer system, language, and tools used. This is due to the large variation of metrics available for different computer systems and at different levels. During this step, we identify which metrics are measurable for the paradigms and systems being used. Specifically, we identify the instrumentation tools that are available and the metrics that are measurable at the operating system, application, and hardware levels. Then from these, for a given paradigm, we select the APART-recommended set of metrics. Important metrics suggested by the application programmer should also be selected. Once a set of performance metrics is selected, instrumentation is activated to collect the data. Code is compiled and linked as needed, and performance data are collected during execution.

After data collection, analysis begins, and the performance metric dataset is first formatted to support the statistical techniques. For one experiment, a matrix format is used. Each element of the matrix is either an average or absolute metric value. An average value, M_{avg}, is computed as the sum of all metric sample values divided by the number of samples, where the samples of the metric values are taken during execution time only. For example, page faults per second might be measured as an average. An absolute value, M_{abs}, is a metric whose value is obtained as a total at the end of execution time only. Total execution time is an example of an absolute metric. One experiment consists of R experimental runs in a predefined random order. This random order determines the precision obtained in the results. Let P denote the number of performance metrics measured during an experimental run. Let r denote the experimental run where $0 \le r \le R-1$ and p is the metric identification number where $0 \le p \le P-1$. This results in the following data format for one experiment:

$$X = \begin{bmatrix} M_k(0,0) & \cdots & M_k(0,P-1) \\ M_k(1,0) & \cdots & M_k(1,P-1) \\ \vdots & \ddots & \vdots \\ M_k(R-1,0) & \cdots & M_k(R-1,P-1) \end{bmatrix} \quad (1)$$

where $M_k(r,p)$ denotes average or absolute metric value for experimental run r and metric p, and k is either avg or abs metric value. Each column of this performance data matrix contains the measurement of one performance metric over a set of experimental runs and each row contains information about one experimental run. Several statistical techniques may be applied to this matrix to extract unbiased information on the statistics of the system and the performance of the application on this particular system .

NOVEL TEACHING METHODOLOGY IN ASPA

Since a very large class of signal processing applications treat signals as discrete entities with finite support, the novel methodology presented in this work centers on the use of signal algebra and operator algebra theoretic methods tools to assist in the analysis and design of signal processing algorithms [4].

Emphasis is given to *modular and scalable* computing methods for Communications Signal Processing applications and their implementation using high performance computer systems. These methods deal with the algorithmic treatment of *finite duration* digital signals in order to extract information important to a user or software/hardware agent. The modular and scalable approach to these computing methods implies that the functions and structures of the algorithmic treatment should adapt to changes in the scales of an associated target system and the size or dimensionality of the signals to be processed. The algorithmic treatment concentrates on understanding fundamental principles involved in Signal Processing to *observe, quantify, represent, transform, qualify, and render* information-carrying signals in a sensory reality. Here an algorithm is described as a "well-defined" procedure to solve a problem in a finite number of steps.

A problem is anything which requires a solution. Communications Signal Processing is defined here as an area dealing with analysis, design, and implementation, of signals and systems for the transmission and reception of communications signals. A communication signal is defined as a signal appearing in any of the stages of an arbitrary communications system. We concentrate on systems designed for digital communication and processing of signals to transmit and receive information, and the way they implement the discrete Fourier transform and other operators used in one-dimensional finite duration digital signal processing. One of the most useful ideas resulting from the integration of these two courses is the concept of a HPC system as a communication channel whose signal is the set of performance metrics. This signal treatment of performance data provides the mechanism to extract information from the signal and possibly use it to automate performance tuning of applications to HPC systems. Concepts such as data mining, feature subset selection, and information content are then applied to these signals coming from the HPC system to identify system performance problems [4].

INTEGRATING TEACHING DEVELOPING TOOLS

Guidelines for integrating teaching developing tools for courses in HCI and ASPA center on the concept of treating a large scale signal processing applications as applications which may be distributed in time and space and require a large amount of computational effort. Interconnecting the application in space usually requires a distributed computer system and the time interconnection is usually accomplished through parallel processing systems. An example is presented in Figure 1 below where the signal processing application deals with sensor array signal processing.

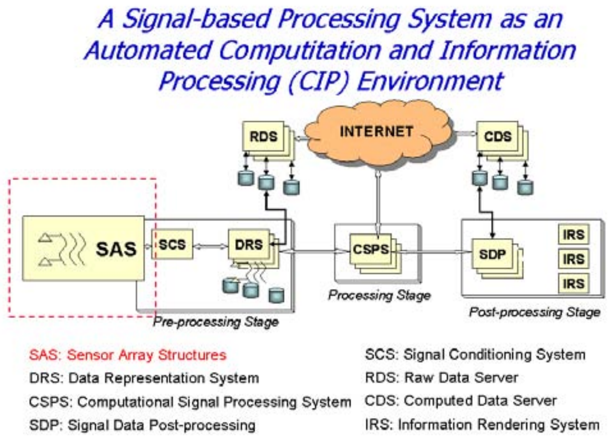
A Signal-based Processing System as an Automated Computitation and Information Processing (CIP) Environment
RDS
INTERNET
CDS
SAS
SCS
DRS
CSPS
SDP
IRS
IRS
IRS
Pre-processing Stage
Processing Stage
Post-processing Stage
SAS: Sensor Array Structures
DRS: Data Representation System
CSPS: Computational Signal Processing System
SDP: Signal Data Post-processing
SCS: Signal Conditioning System
RDS: Raw Data Server
CDS: Computed Data Server
IRS: Information Rendering System

FIGURE 1
HPC SIGNAL PROCESSING APPLICATION EXAMPLE

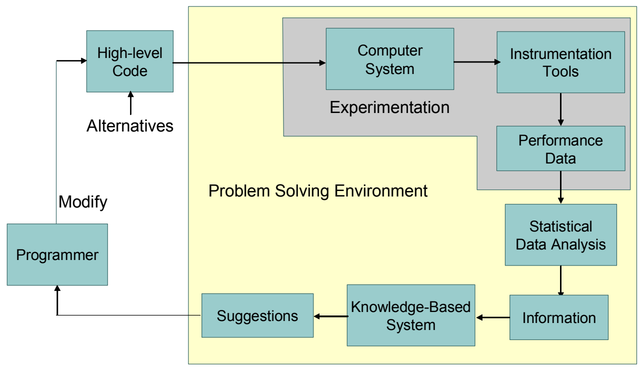

FIGURE 2
HPC PERFORMANCE DATA ANALAYSIS METHODOLOGY

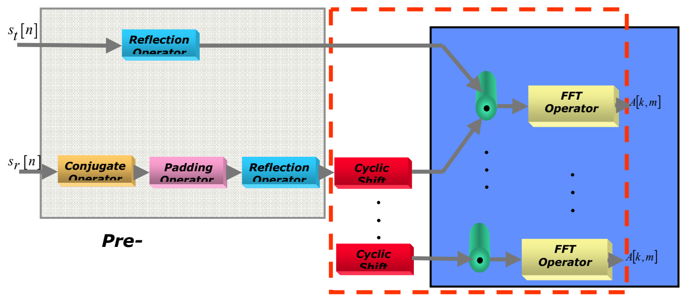

FIGURE 3
HPC COMPUTATIONAL SIGNAL EXAMPLE

CONCLUSIONS

This work presented results of an ongoing process for integrating novel methodologies, tools, and IT resources for graduate courses in the areas of High Performance Computing and Advanced Signal Processing Algorithms. The authors feel that this integration process is contributing to the overall teaching improvement of areas.

ACKNOWLEDGMENT

We would to thank all the undergraduate and graduate students who have contributed to this work.

REFERENCES

[1] Santiago, N. G., Rover, D. T., and Rodriguez, D., "A statistical approach for the analysis of the relation between low-level performance information, the code, and the environment", *The 4th Workshop on High Performance Scientific and Engineering Computing with Applications, HPSECA-02*, August 2002, pp. 282-289.

[2] C. Huallaprimachi, D. Rodriguez, "Java-based Tool for Synthetic Aperture Radar Image Analysis," Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, Rome, Italy, 2003.

[3] Riley, G. D. and Gurd, J. R., "Requirement for automatic performance analysis APART". Technical Report FZJ-ZAM-IB-9910, Central Institute for Applied Mathematics, Research Centre Jülich, November 1999.

[4] Santiago, N. G., Evaluating Performance Information for Mapping Algorithms to Advanced Architectures. PhD thesis, Michigan State University, 2003.