ITR/SY: A Distributed Programming Infrastructure for Integrating Smart Sensors

NSF Program CCR-0121638

PIs: Umakishore Ramachandran
Kenneth Mackenzie
Steve DeWeerth
Irfan Essa
Thad Starner

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
Phone: (404) 894-5136
FAX: (404) 385-2295
e-mail: rama@cc.gatech.edu
WWW URL: http://www.cc.gatech.edu/~rama

Annual Report via Fastlane May 31, 2002

1 Activities and Findings

1.1 Research and Education

The proposed research is integrating sensing hardware, embedded processing and distributed system support to build a seamless programming infrastructure for ubiquitous presence applications. Fundamental invention and integration of techniques spanning programming idioms and runtime systems for distributed sensors, and building blocks for embedded processing are expected as the primary intellectual contributions of the proposed research. Interfacing these technologies to emerging applications on the one end and novel off-the-shelf sensors at the other end are secondary goals of the proposed research.

This subsection details the research accomplishments this past year as well as plans for the coming year.

1.1.1 Distributed Systems Technology

D-Stampede Programming System. We focus on an important problem in the space of ubiquitous computing, namely, programming support for the distributed heterogeneous computing elements that make up this environment. We address the interactive, dynamic, and stream-oriented nature of this application class and develop appropriate computational abstractions in the *D-Stampede* distributed programming system [1]. The key features of D-Stampede include indexing data streams temporally, correlating different data streams temporally, performing automatic distributed garbage collection of unnecessary stream data, supporting high performance by exploiting hardware parallelism where available, supporting platform and language heterogeneity, and dealing with application level dynamism.

Dead Timestamp Identification in Stampede. In a Stampede program, *threads* are loosely connected by *channels* that hold *timestamped* data items. There are two performance concerns when programming with

Stampede. The first is *space*, namely, ensuring that memory is not wasted on items that are not fully processed. The second is *time*, namely, ensuring that processing resource is not wasted on a timestamp that is not fully processed. We have introduced a single unifying framework, *dead timestamp identification*, that addresses both the space and time concerns simultaneously. Dead timestamps on a channel represent garbage. Dead timestamps at a thread represent computations that need not be performed. This framework has been implemented in the Stampede system. Experimental results showing the space advantage of this framework are presented in [2]. Using a color-based people tracker application, we show that the space advantage can be significant (up to 40%) compared to the previous garbage collection techniques in Stampede.

Towards Aspect-Oriented Programming Support for Cluster Computing. Interactive multimedia applications (such as audio/video processing) are good candidates for cluster computing. Such applications are best represented as coarse-grain dataflow graphs and are rich in pipelined, task, and data parallelism. Specification of the strategies for mapping computational abstractions to compute nodes, and their plumbing are two important issues in the design of such complex parallel and distributed applications. Due to the varieties of parallelism that are available in such applications, the space of strategies to be explored can be vast. We have developed an aspect-oriented programming language for cluster computing called *STAGES*. This language allows the algorithm design to be disentangled from the connection management and performance concerns. STAGES provides a simple syntax for specifying the connections among threads and data abstractions, and their mapping onto the nodes of the cluster. The current implementation targets the *Stampede* cluster programming library. However, the language is general and can be retargeted to a different set of abstractions. In [3], we present STAGES, its implementation, and its utility for mapping complex applications onto a cluster. We also present performance results from exploring the parallelism space for two such applications on a 17-node cluster of 8-way SMPs (Intel Xeon processors) interconnected by Gigabit Ethernet.

Interaction Between Stampede Runtime and Operating Systems. The Stampede system has been built as a runtime library on top of standard operating systems. In [4], we study the interaction between the Stampede runtime system and the underlying operating system. The study is conducted on two identical hardware platforms running Solaris and Linux, respectively. A cycle-accurate event logging facility using the CPU cycle counter is at the core of this study. There are several interesting insights coming from this study [4]. First, memory allocation does not take up a significant amount of the execution time despite the interactive and dynamic nature of the application domain. Second, the Stampede runtime does not pose a significant overhead over raw messaging for structuring such applications. Third, the results suggest that the thread scheduler on Linux may be more responsive than the one on Solaris. Fourth, the messaging layer spends quite a bit of time in synchronization operations.

1.1.2 Mechanisms for Embedded Architectures

Software Caching for Embedded Sensor Processing. Software caching reconciles programmability with cost for embedded processing associated with sensors in a distributed sensing infrastructure. Local processing is important to addressing the bandwidth hierarchy problem of distributed sensing. Numerous local processors demand both low cost and ease of programming in order to scale. Software caching is a novel technology that gives embedded processors containing only local, on-chip memory the programmability features of caching and memory management without the hardware cost.

Software caching with binary rewriting fits the embedded scenario in three ways. First, automatic caching enables the convenient programming model of a large address space at the embedded system. Second, an all-software implementation minimizes cost and (potentially) power over a hardware implementation while maximizing flexibility. Flexibility is crucial to achieving high, predictable hit rates which are necessary when cache miss time is high. Third, dynamic binary rewriting shifts some software caching overhead from the simple embedded system to the powerful server. The rewriter reduces the time and space of cache miss checks through specialization at the cost of extra overhead in servicing cache misses. The cache miss checks run in the embedded system while the misses are handled by the server.

Over the past year we have developed a two-piece software cache system [6, 7] where the server is any

Unix-based platform and client is a StrongARM-based device such as a Skiff or iPAQ. We are applying this system to run sensor-local software in the distributed sensing infrastructure project.

1.1.3 Sensor Technologies

Sensor Lab. We are provisioning a Sensor Lab within the College of Computing to support our research into flexible software infrastructure for pervasive, heterogeneous, sensor networks. Our lab will include modest quantities of a wide variety of devices, both wired and wireless, from simple commercially available (COTS) components to complex research prototypes.

Initially, we are in the process of deploying five technologies. The first is a commercial tag-based location system from Versus Technology (`www.versustech.com`) that uses IR sensors to locate suitably tagged resources (equipment, individuals, etc.). RF sensors are also used to provide a simple button-based back channel for notifications from tagged individuals. This system has been successfully installed in the Dartmouth computer science department and we are working with both researchers at Dartmouth and developers at Versus Technology.

The second technology is a grid of simple, low-cost wireless temperature sensors from Point Six (`www.pointsix.com`). These devices are not programmable and represent the low-end in the spectrum of sensor complexity. These sensors are easily relocatable and provide fixed-rate broadcast of temperature data via IR.

For the third technology, we are deploying commercial ultrasonic sensors from several vendors, based on the Polaroid 6500 processor board and Polaroid 7000 series transducers. We intend to make a variety of hardware modifications to these sensors to experiment with different location techniques. For example, we plan on modifying the chipset to output a "profile" (1D analog intensity vs. time) to help distinguish individuals from inanimate objects in a room. We also plan on deploying several sensors in a room and launching a pulse from a single sensor to measure transmitivity and scattering, as well as reflections.

For our fourth sensor technology, we are working with the well-known Berkeley MICA Motes, commercially available from Crossbow (`www.xbow.com`). These wireless devices are programmable via the TinyOS operating system and feature photo diodes, thermistors, accelerometers and microphones. We are combining our Mote-based efforts with colleagues in the Aware Home (Gregory Abowd, College of Computing).

The fifth technology we are currently working with is a sophisticated research prototype developed by Mark Smith at HP Labs known as the Smart Badge 4. This single-board computer runs Linux and supports a variety of sensors and interconnects, including wireless. Because it supports Linux, this device is an excellent testbed for experimenting with OS and architectural interactions with "smart" sensors.

Software Infrastructure. While a variety of software technologies exist to facilitate deployment of sensor networks, these tools are often provided by individual vendors to support their particular technology. Very little work has been done directed toward infrastructures specifically designed to support large-scale, heterogeneous, sensor networks.

Sensor-based applications in richly heterogeneous environments place a variety of demands on software infrastructure. Sensing devices must be dynamically activated and deactivated, tuned, maintained, and sometimes programmed in the field. Failure is potentially frequent (e.g. battery failure) depending on the delicacy of the equipment and harshness of the environment. Ironically, sensors must themselves be monitored for responsiveness and accuracy by higher levels of software. The location of mobile sensors must be tracked and inferred. Sensor data requires conditioning in device and application specific ways. Sensor data must be acquired as needed (either push or pull) and the rate and characteristics of data acquisition may vary dynamically during the lifetime of an application. In large-scale heterogeneous environments, sensor data streams should be published (described and registered) to facilitate efficient access and sharing. Data streams are often virtualized (filtered and processed and re-published) to provide cleaner, higher-level sensor data abstractions. Mechanisms are required to fuse, persist and query, recent and historical sensor data. Post-acquisition processing sometimes requires (e.g. image analysis) significant computational resources. In such cases, sensor data must be routed to appropriate computational clusters in a timely manner and the results of analysis must be made available to end applications. All of these requirements must be met in a

scalable, cross-platform and power-conserving way.

We are working toward providing support for heterogeneous sensor networks within the D-Stampede programming infrastructure. Our Sensor Lab will serve as a testbed to focus and validate design, realize and optimize implementation, and inspire and confirm novel sensor-based applications.

Our preliminary infrastructure efforts have involved the design and implementation of a data fusion library for use in D-Stampede. The primary abstraction in this design is a *fusion channel*. A fusion channel is a specialized Stampede channel that abstracts the application of a programmer-supplied fusion function to sets of correlated data items, acquired from a programmer-specified set of input channels. Items are correlated by virtual timestamp and we have devised mechanisms to synchronize and appropriately pace virtual time. Fusion can be activated "on demand" or in a "data driven" manner, providing both "push" and "pull" styles of data flow. Fusion channels can be composed to form fusion pipelines. Fusion channels offer the same benefits as Stampede channels (garbage collection, transparent cross-platform access, random access by virtual timestamp, etc.). We have paid careful attention to caching and flow-regulation to provide good performance.

We designed fusion channels based on three "real-world" fusion applications solicited from our colleagues at Georgia Tech. The first application is a multi-camera 3d image reconstruction system used for teleconferencing (Matt Hans, ECE and HP labs). The second involves fusion of location data to increase accuracy from cooperative teams of robots (Tucker Balch, College of Computing). The third involves fusion in a variety of contexts in a system for the specification, simulation and execution of robot plans (Ron Arkin, College of Computing).

In the near future we hope to add further support for synchronization and possibly incorporate power-saving time synchronization algorithms for wireless devices. We also plan to explore infrastructural support for naming, description and discovery of sensor services along with high-level access (DB queries) to persistent sensor data. In addition, we are exploring the introduction of quality-of-service directives into Stampede applications to allow system-wide adaptation of resource management when insufficient resources are available to meet resource demands. We are using the metaphor of selective attention in biological systems to guide this work.

All of these mechanisms are being encapsulated within a *media broker architecture* on top of D-Stampede. The media broker will serve as a clearing house for applications that need sensor data. The key idea is to virtualize sensors so that they can be accessed in an application specific way via the media broker. We hope to provide stable releases of D-Stampede to these and other colleagues to use for sensor and fusion related applications.

1.1.4 Application Driven Studies

On-line Television Channel Recommender. One of our first applications is an agent that monitors a large number of television channels and recommends ones with similar content to the one a user is watching. As a concrete example, during the Persian Gulf war, military commanders used civilian news agencies for some of their reconnaissance. However, one news watcher can not hope to watch the content of the many channels providing coverage. Our "TV Watcher" is designed to capture the closed captioning of the channel the user is currently watching and compare this information to the closed captioning on other channels. Using information retrieval techniques and user interfaces similar to the Remembrance Agent [10], the TV Watcher presents the user with one line summaries and a thumbnail video of similar news stories on other channels. Thus, the user may see that another channel has new or different information and can change to that channel with a simple keystroke for a different perspective.

This application and future variants stress our architecture in several ways. First, each video channel is captured by a separate process with current PC hardware limited to two to three such processes per machine. A small stream of information (the closed captioning text) is continuously sent to a correlation engine. If the channel is selected to be displayed by the correlation or user interface processes, the stream may increase significantly. Thus, the flow and demand for information from the capture cards (or "sensors" in this system) is continuous and dynamic. In future variants where image correlation or OCR is performed on the video channels to help determine relevance, code or correlation databases may also be pushed back toward the sensors.

Due to the extensive undergraduate involvement of this project and personnel changes, we have not yet written a paper on this project. However, we are currently porting the original demonstration prototype to a more formal D-Stampede version of the system for study.

Aware Environments. One goal of our ongoing project is to build and study the importance of an anthromorphic agent in the home environment. We envision an agent that can represent the house (i.e. be the face of the house) and interact with the residents naturally. At present, we are developing various toolkits to support the building of such agents with faces. These agents need to be aware of the residents, track where they are, make eye contact and converse with naturally. After the technology and constructions goals are achieved, we intend to evaluate the importance of such anthromorphic interfaces.

At present, this project is pursuing two main types of face models:

1. One is the Face Robot (called "Pong") constructed as a research prototype by IBM's Blue Eyes Group (`www.almaden.ibm.com/cs/blueeyes/`). "Pong" is an attentive robot, uses the technique uses special cameras installed as eyes that track the eyes of the person in front of it. Our goal is to design a tracking system which can be later used for face recognition purposes. It is currently being integrated with an audio system to achieve improved performance and reliability.
2. We are also building detailed graphical models as a part of a related project that will also be used to develop graphical agents for the house. These graphical agents will be aware of the residents, and interact with them in a natural manner. See `www.cc.gatech.edu/cpl/projects/animated-speakers/` for more details.

An important aspect of this work is the fusion of sensory information from various types of sensors and integrating them for higher-level interpretation that supports the face-to-face interaction between the face of the house and the user [11]. For this reason, we are building our algorithms using the media broker architecture that is being implemented on top of D-Stampede.

1.2 Training and Development

1.2.1 Undergraduate Research Participation

We continue to attract bright and interested undergraduates to research projects in our group. Undergraduate participation in research within the College is facilitated by the excellent UROC program (`www.cc.gatech.edu/program/uroc`), coordinated by Amy Bruckmann. A variety of institute-wide programs are also available (`www.undergraduateresearch.gatech.edu`) including a special fund sponsored by the president of Georgia Tech and several NSF-sponsored projects (URIP, SURF, etc.). In addition, we sponsor the Systems Hackfest group each semester that includes 5-10 undergraduates participating in research-related projects for fun or course credit.

Here are some projects that undergraduates have worked on that we have been demonstrating to visitors to Georgia Tech:

Integrating Selective Attention and Soft Caching in the Stampede Framework. The work brings together smart sensing, programmable and cheap hardware platforms to serve as data aggregators for smart sensors, and a distributed programming model that allows the seamless programming of a continuum of sensors, actuators, and backend high-performance clusters. Accomplishments to date include development of a retinal camera, a softcache architecture for just in time loading of hot code and data on data aggregators, and extension of the Stampede distributed programming model to wireless and wired devices.

Distributed Selective Video Streaming using D-Stampede. D-Stampede is a distributed programming system with support for temporally indexing stream data emanating from several sources. In this work, we illustrate the ability to compose applications using D-Stampede with multiple cameras, making efficient use of network bandwidth. Local processing at each camera sends the centroid value to a head end. A thread at the head end instructs the camera with the highest score (indicative of the location of most action at the current time) to stream full video. A GUI allows a manual over-ride as well. This system has applications in distributed surveillance, and assisted living.

1.3 Outreach

The College of Computing at Georgia Tech continues its popular summer intern program (funded by ONR) for promising undergraduate students from Colleges and Universities who do not have the infrastructure to support research in computing. As an example, in the summer of 2000, we had participants from Spelman College, University of Puerto Rico, Florida A&M, Talladega College, Morris Brown College, and Morehouse College. The intent is to match these students with faculty and graduate students who can mentor them through an eight-week research project. This program provides the interns with a rich summer research experience, and serves as a port-hole for what awaits them if they decide to go to graduate school.

We have held several open-house events in 2001 to acquaint researchers from within and outside the College of Computing to the technologies being developed in our group.

Finally, in November 2001 we held a successful Symposium on Systems and Networking Technologies at Georgia Tech (`www.cc.gatech.edu/~rama/tech-symp.pdf`). The intent was to feature cutting-edge research from leading industries. The Symposium allowed Tech students to build bridges with industrial researchers and also provided a showcase for ongoing systems research at Georgia Tech.

1.3.1 Advisory Board

Prof. Ramachandran is also a PI on an NSF research infrastructure (RI) award. As part of managing the NSF RI award, we hold an annual research advisory board meeting with external board members. The board members for our 2001 meeting included: Roger Haskin (IBM Almaden), Jim Rehg (formerly at Compaq CRL), Yousef Khalidi (Sun), Gita Gopal (HP labs), Rick Schlichting (AT & T), Willy Zwaenepoel (Rice), Kai Li (Princeton), and Mary Vernon (UW-Madison).

Much of the research work being conducted for the ITR project uses the RI equipment. Thus the advisory board meeting features the research work of the ITR project quite prominently and thus obtains valuable feedback and establishes connections with industries.

The next board meeting has been tentatively scheduled for Fall of 2002.

2 Publications and Products

2.1 Publications

See the references at the end of this document.

2.2 Web Site

Please visit the project web site at www.cc.gatech.edu/~rama/ubiq-presence/

3 Contributions

The activities we are currently undertaking have resulted in a significant number of publications and software artifacts. These are listed in the references at the end of this report.

3.1 Human Resources

We have roughly 15 graduate students, 10 undergraduate students, and 4 research scientists working in areas related to this project.

3.2 Research and Education

The research artifacts from the project are finding their way into graduate courses. Professor Ramachandran taught a special topics course entitled, "Pervasive computing with distributed sensors," (URL: `www.cc.gatech.edu/classes/AY2002/cs8803e_spring`). Some of the students in the course used D-Stampede as an implementation vehicle for the course project.

Further we are in the process of establishing a *sensor lab* via funding from an NSF RI award. This lab will serve both the research agenda of this ITR award as well as curricular development both in the College of Computing and School of Electrical and Computer Engineering at Georgia Tech.

4 Special Requirements

The total budget for this 5-year proposal is $1.35M. However, due to fiscal constraints NSF chose to front-load the total budget by awarding $750,000 in the first year. The understanding with the program manager (Dr. Helen Gill) is that our spending plan for the award will be more balanced over the 5-year period despite the front-loaded nature of the allocation. Hence, we have over $500,000 still remaining from the first year allocation.

References

[1] Sameer Adhikari, Arnab Paul, and U. Ramachandran. "D-Stampede: Distributed Programming System for Ubiquitous Computing," (To Appear in) *22nd International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, July, 2002.

[2] Nissim Harel, Hasnain A. Mandviwala, Kathleen Knobe, Umakishore Ramachandran. Dead Timestamp Identification in Stampede. (To appear in) *International Conference on Parallel Processing 2002*.

[3] Matthew David Wolenetz, Hasnain A. Mandviwala, Sameer Adhikari, Yavor Angelov, Umakishore Ramachandran, Kenneth Mackenzie, James Matthew Rehg. Towards aspect-oriented programming support for cluster computing. Unpublished manuscript. April 2002.

[4] Arnab Paul, Nissim Harel, Sameer Adhikari, Bikash Agarwalla, Umakishore Ramachandran, Ken Mackenzie. "Interaction Between Stampede Runtime and Operating Systems" Submitted to *ACM OSDI 2002*.

[5] A Comparative Study of Stampede Garbage Collection Algorithms. Nissim Harel, Hasnain A. Mandviwala, Kathleen Knobe, Umakishore Ramachandran. Unpublished manuscript. April 2002.

[6] Chad Huneycutt, Joshua B. Fryman, and Kenneth Mackenzie. Software Caching using Dynamic Binary Rewriting for Embedded Devices. (To appear in) *International Conference on Parallel Processing 2002*.

[7] Joshua Fryman, Chad Huneycutt and Kenneth Mackenzie. Investigating a SoftCache via Dynamic Rewriting. *Fourth Workshop on Feedback-Directed and Dynamic Optimization (FDDO)*, Austin, TX. 2001.

[8] Kenneth Mackenzie, Eric Hudson, Drew Maule, Sundaresan Jayaraman and Sungmee Park. A Prototype Network Embedded in Textile Fabric. *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, Atlanta, GA. 2001.

[9] Adam Johnson and Kenneth Mackenzie. Pattern Matching in Reconfigurable Logic for Packet Classification. *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, Atlanta, GA. 2001.

[10] B. Rhodes and T. Starner. Remembrance Agent: A continuously running automated information retreival system. *Proc. of Pract. App. of Intelligent Agents and Multi-Agent Tech. (PAAM)s*, London, April, 1996.

[11] Stillman, S. and I. Essa. "Towards Reliable Multimodal Sensing in Aware Environments" *Perceptual User Interfaces (PUI 2001) Workshop (held in Conjunction with ACM UIST 2001 Conference)*, November 2001.

[12] Moore D.and I. Essa. Recognizing Multitasked Activities using Stochastic Context-Free Grammar, *Proceedings of AAAI Conference*, Edmonton, CA, August 2001.