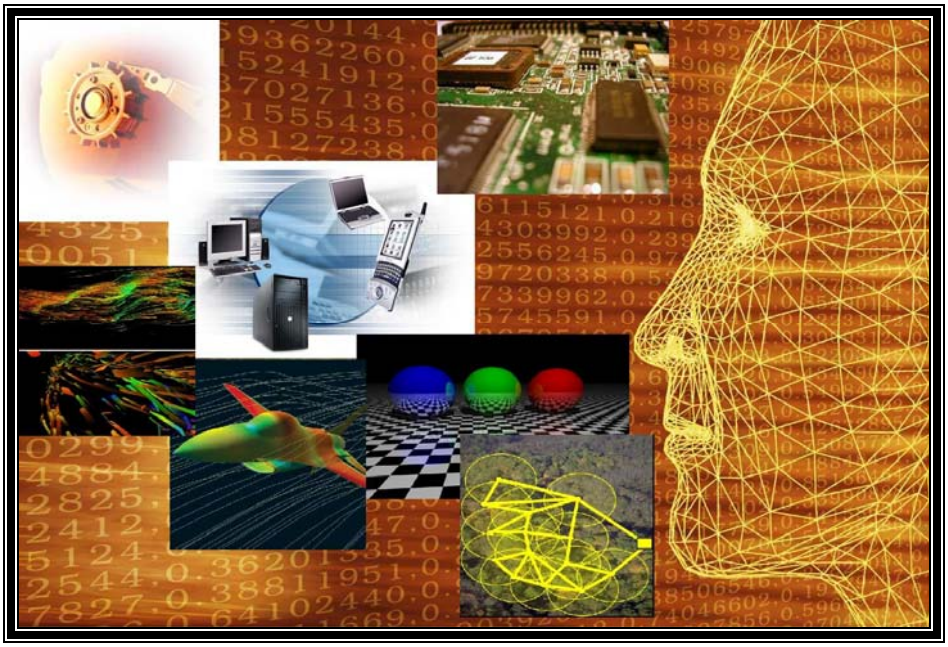
GSWC 2007



Proceedings of

The Second Annual Graduate Student Workshop on Computing

September 28th, 2007
Santa Barbara, California

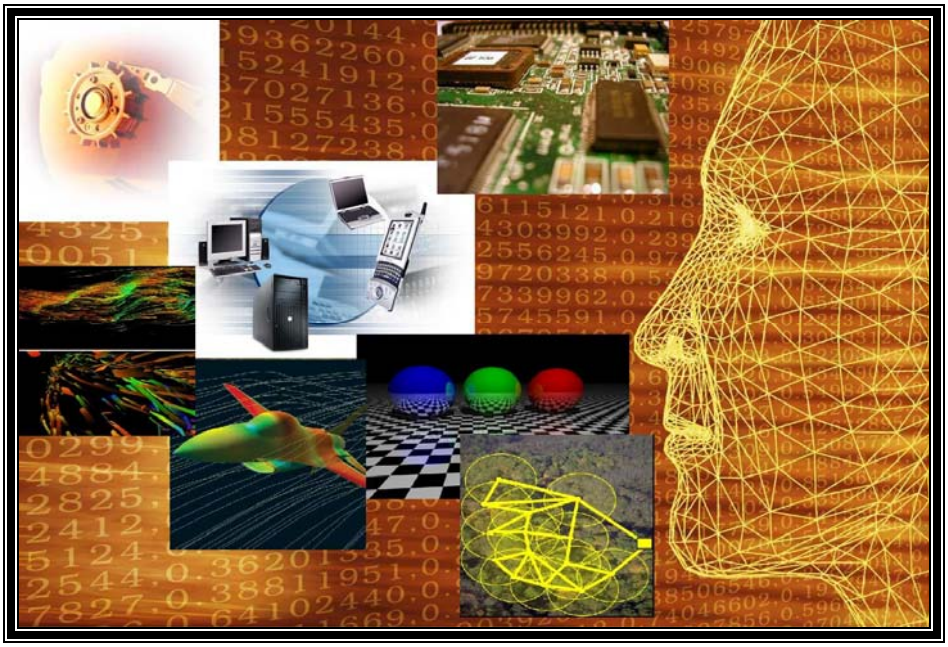
Sponsored by

Google

http://www.google.com

Department of Computer Science & Corporate Affiliates Program
University of California, Santa Barbara
http://www.cs.ucsb.edu, http://www.industry.ucsb.edu

GSWC 2007



Proceedings of

The Second Annual Graduate Student Workshop on Computing

September 28th, 2007
Santa Barbara, California

Sponsored by

Google

http://www.google.com

Department of Computer Science & Corporate Affiliates Program
University of California, Santa Barbara
http://www.cs.ucsb.edu, http://www.industry.ucsb.edu

Message from the Workshop Chairs

It is our pleasure to welcome you to the *Second Annual Graduate Student Workshop on Computing* (GSWC), a symposium bringing together researchers from various disciplines of Computer Science at the University of California, Santa Barbara.

The GSWC began in 2006 with the goal of providing exposure to the wide-ranging research endeavors of our department's brilliant graduate students. Papers for this conference are selected first and foremost on their technical quality, but are also reviewed for clarity and presentation. This year, the program committee selected 10 papers for inclusion in the workshop – at an acceptance rate of 34% – reflecting the high quality of research within our department. Additionally, we have organized a poster session in which a select group of graduate students will present their work. The committee has chosen 12 posters, providing flavors from a diverse breadth of research.

It takes a large, dedicated team to put together a successful workshop like the GSWC. We would like to thank our program committee members who devoted time and attention to selecting quality material for the workshop. We sincerely thank Google for their generous sponsorship of the GSWC. Our department staff – particularly Greta Carl-Halle, Amanda Hoagland and Julia Orr – was very responsive in offering technical and organizational support. The department chair, Professor Amr El Abbadi, has always been there to support our efforts and offer his assistance. We owe tremendous thanks to Andrew Elliot and Kelly Bret for helping us plan and coordinate logistics, and to the Corporate Affiliates Program for their support. Finally, we would like to thank Professor Fred Chong for offering us the opportunity to organize this workshop.

We hope that you will enjoy participating in the Graduate Student Workshop on Computing as much as we have enjoyed coordinating it. Here's to many more iterations of the GSWC in the years to come.

Sincerely,
Susmit Biswas and Robert Gilbert
GSWC 2007 Committee Chairs

Table of Contents

GSWC 2007 Program Committee Members

Committee Chairs:

Susmit Biswas

Robert Gilbert

Faculty Advisor:

Frederic T. Chong

Committee Members:

Matthew Allen

Sorabh Gandhi

Imran Patel

Stacy Patterson

Krishna Puttaswamy

Irfan Sheriff

Vishwakarma Singh

Sunil Soman

Mohit Tiwari

Jason Wither

Lamia Youseff

Administrative Support

Greta Halle
Student Affairs Manager
Department of Computer Science, UCSB

Amanda Hoagland
Graduate Program Coordinator
Department of Computer Science, UCSB

Kelly Bret
Public Event Manager
College of Engineering, UCSB

Andrew Elliott
Corporate Programs Manager
Engineering and the Sciences, UCSB

Workshop Schedule

8:45-9:00	Robert Gilbert *Program Chair*	*Opening Remarks*
Morning Session		
9:00-9:15	Shashidhar Mysore	*System Analysis through 3D-Integration*
9:15-9:30	Vikram Aggarwal	*The structure and generation of Non-graded Finite Difference Octree Grids*
9:30-9:45	Stacy Patterson	*Environmental Tomography*
9:45-10:00	Fang Yu	*Automated Size Analysis for Object-Oriented Systems*
10:00-10:15	Daniel A. Vaquero	*An Analysis of Shadows in Camera-Light Pairs and Its Application to Multiflash Depth Edge Detection*
10:15-10:30	Break	
10:30-11:30	**Russell Quong,** *Invited Speaker*	*Software at Google: Tolerance in the face of pretty much everything*
11:30-1:00	Poster Session Lunch	*Poster Presentations*
Afternoon Session		
1:00-1:15	Marco Cova	*Anomaly-based Detection of State Violations in Web Applications*
1:15-1:30	Aydin Buluc	*On the Representation and Multiplication of Sparse Matrices*
1:30-1:45	Marc Griesemer	*Modeling and Simulation of Protein-Protein Interactions in the Endoplasmic Reticulum*
1:45-2:00	Krishna Puttaswamy	*Searching for Rare Objects using Index Replication*
2:00-2:15	Lamia Youseff	*Evaluating the Impact of Xen on the Performance of NAS Parallel Benchmarks*
2:15-2:30	Amr El Abbadi *Department Chair*	*Closing Remarks*

System Analysis through 3D-Integration

Shashidhar Mysore, Banit Agrawal, Navin Srivastava, Sheng-Chih Lin,
Kaustav Banerjee, Timothy Sherwood
Department of Computer Science and Department of ECE
University of California, Santa Barbara
{shashimc,banit,sherwood}@cs.ucsb.edu, {navins, sclin, kaustav}@ece.ucsb.edu

ABSTRACT

While the number of transistors on a chip increases exponentially over time, the productivity that can be realized from these systems has not kept pace. To deal with the complexity of modern systems, software developers are increasingly dependent on specialized development tools such as security profilers, memory leak identifiers, data flight recorders, and dynamic type analysis. Reducing the performance penalty and complexity of these software tools is critical to those developing next generation applications, and many researchers have proposed adding specialized hardware to assist in profiling and introspection. Unfortunately, while this additional hardware would be incredibly beneficial to developers, the cost of this hardware must be paid on every single die that is manufactured. In this article we summarize our findings from a longer article [6] and argue that a new way to attack this problem is with the addition of specialized analysis hardware stacked vertically with the processor die using 3D-interconnect. This provides a modular "snap-on" functionality that could be included with developer systems, and omitted from consumer systems to keep the cost impact to a minimum.

1. 3D Introspection Overview

Developing high quality software for a modern computer system is no easy task. Performance critical applications are likely to execute for quadrillions of instructions, operate in a complex environment with multiple run-time components, and are increasingly responsible for managing various architectural resources including power and hardware threads. In order to battle this complexity, developers are becoming more dependent on sophisticated software analysis tools. While mixed static-dynamic analysis can be done completely in software through binary instrumentation, the amount of analysis that can be done at test-time is bounded by the performance impact that can be tolerated. In long running or interactive programs, this is especially critical. To enable runtime analysis with low overhead many researchers have proposed the development of specialized on-chip hardware modules that can assist software developers in building more secure, more bug free, and more efficient applications.

The primary goal of this article is to explore a new method by which analysis functionality can be added to a processor. Specifically, we propose a new and modular way to add analysis hardware to next generation processors through the use of 3D-interconnect. Several 3D-interconnect technologies, such as inter-die vias, are currently being evaluated in industry as a means of stacking multiple chips together. Some potential applications include the stacking of DRAM or bigger cache directly onto the processor die to alleviate memory pressure and designing stacked chips of multiple processors. While the details of this technology are more fully described in our paper [6], the main idea is that two pieces of silicon are fused together to form a single chip, and the two active layers of the silicon are connected through inter-die vias (called posts) which run vertically between them. This ability to interconnect multiple active layers means that we can consider optionally adding a layer to a processor specifically for analysis which would have easy access to most of the important signals of the system. A processor with this ability could be sold to developers, while commodity systems would simply not include this extra analysis layer.

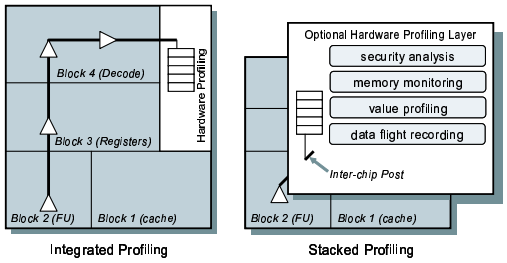


Figure 1: The traditional approach to attacking the hardware profiling problem involves integrating specialized profiling functionality on the same die as the processor. To gather information, long global wires are required which necessarily cross multiple functional blocks. To get high performance, buffers or pipeline latches are required, which in turn require access to silicon which makes for a big mess. Alternatively, in a stacked approach, only a single buffer is required to drive the post up to the analysis layer (which would be an optional feature for software developers)

This inter-chip 3D interconnect could take the form of any number of different competing technologies, including chip-bonding, Multi-chip Modules (MCM) [4], chip-stacking with vias [3], or even wireless superconnect [5]. While chip-bonding and MCM technology are already used in a variety of embedded contexts [2], more aggressive interconnect technologies are being heavily researched by several major industrial consortiums.

Specifically, this article describes the potential of 3D interconnect technology to enable new forms of introspective chips and fully elaborate on some of the advantages of 3D introspection over traditional hardware integration. In our original article [6], we precisely quantify both the chip bandwidth requirements for full introspection, and the relevant characteristics of 3D interconnect technology. We further quantify the increase in area, the interconnect overhead, and both the power and thermal impacts of such a design. A brief description of some of the advantages (more fully described and evaluated in the full paper) are:

1. Reducing Introspection Routing Problems - Instead of be-

ing forced to route performance data through other blocks, inter-die vias can move data out of plane to a layer specially constructed for gathering and analyzing run-time information as illustrated in Figure 1.

2. Decoupling Developer Needs from End User Systems - We advocate the sale of one type of processor which is *always fabricated with connections for hardware monitoring*. The difference between the system we sell to the consumer and the one that is sold to the developer is only whether the hardware monitor devices are actually stacked on top or not.

3. Opening the Door to Heavy Weight Analysis - Stacking a hardware monitor on top of the main processor is the potential it has to open new avenues of research in heavyweight dynamic program analysis.

2. 3D Technology

This section gives a brief overview about the through-via 3D technology which we have used in our paper for evaluating our proposal for a 3D introspection engine.

Manufacturing Posts Between Two Die - One popular method of fabricating 3D integrated chips is to bond together two fully processed wafers on which transistors and wires have been fabricated, such that the wafers completely overlap. The top wafer is first thinned to approximately 10-50μm. Optically adjusted bonding is then used to stick this layer to the bottom wafer using an organic adhesive layer (2μm) of polyimide. After metalization is done on both layers and prior to the bonding process, electrical connections are needed between the two wafers. The connection is made by inter-chip vias, which are etched through the inter-metal-layer dielectric on the top wafer, the thinned top Si wafer itself and through the cured adhesive layer. The inter-chip vias are then formed in these etched holes using chemical vapour deposited (CVD) tungsten which can withstand the high temperatures (400^oC) of the wafer bonding process. In a modern process, these vertical interconnects typically have cross-sections of 5μm x 5μm and height of 30-40 μm, whereas a normal metal wire's cross section is of the order of 1μm x 1μm [1]. We refer to these inter-chip vias as posts. A second approach relies on thermo-compression bonding between metal pads in each wafer. In this case, Cu-Ta pads on both wafers serve as the electrical contacts between the inter-chip vias on the top thinned Si wafer and the uppermost interconnects on the bottom Si wafer. These processes, as well as other processes (for 3D integration of VLSI chips) are described in [2, 3].

3. Results and Conclusion

Enabling programmers and software developers to more easily track down bugs, identify performance bottlenecks, and secure their code against attacks needs to be one of the primary concerns of system designers at all levels, including computer architects. Even today, software bugs are so damaging and widespread that they cost the U.S. economy an estimated $59.5 billion annually (more than half a percent of the GNP). Although it is certainly not possible to remove all errors, it is estimated that more than a third of the cost associated with bugs could be eliminated through an improved testing and analysis infrastructure [7]. The problem of inefficient and buggy software is not going to be helped by the fact that the amount of hardware complexity exposed to the programmer is growing rapidly on desktop and server machines in the form of threading, parallelism, and complex application middleware. To cope with this complexity, and to ensure the quality of software infrastructures, an increased reliance on sophisticated software analysis and testing tools seems inevitable. Complex pointer errors, memory leaks, race conditions, and performance anomalies may manifest themselves during tests, but finding them requires sifting through a sea of runtime data.

One of the biggest advantages of our approach is that the cost of specialized analysis hardware is *decoupled* from the highly cost sensitive consumer market. In doing so, users can still buy their cheap high performance machines because the only extra hardware they are paying for are stubs. The additional cost of the hardware to perform online analysis, the cost of the interconnect to route the performance data, and the cost of the complexity of handling that global interconnect, are all eliminated. The hardware stubs that are left increase area and power by no more than $0.021mm^2$ and 0.9% respectively, numbers which might be further reduced with careful design. At the same time, developers and users both benefit from the increased analysis power of dynamic monitoring tools. Even though our argument, like most arguments in systems, is economic in nature, in the full paper [6] we have used the metrics of area, power, routability, and temperature to quantify one possible design. While the thermal impact of stacking two hot cores together is always a concern in 3D design, we show that the effect is manageable for both our sample system and for a system 8 times more powerful. Given that developers would need to pay more for this additional hardware anyways, the incremental cost of additional cooling should be a minor.

4. References

[1] International technology roadmap for semiconductors, 2001.

[2] Kaustav Banerjee, Shukri J. Souri, Pawan Kapur, and Krishna C. Saraswat. 3-D ICs: A Novel Chip Design for Improving Deep Submicron Interconnect Performance and Systems-on-Chip Integration. *Proceedings of the IEEE*, 89(5):602–633, May 2001.

[3] Benkart et al. 3D Chip Stack Technology Using Through-Chip Interconnects. *IEEE Design and Test of Computers*, 22(6):512–518, Nov/Dec 2005.

[4] Claude Massit and Nicolas Gerard. Three-dimensional multichip module United State Patents, US 5373189, December 1994.

[5] Miura et al. A 195Gb/s 1.2W 3D-Stacked Inductive Inter-Chip Wireless Superconnect with Transmit Power Control Scheme. In *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pages 264–265, Feb 2005.

[6] Shashidhar Mysore, Banit Agrawal, Navin Srivastava, Sheng-Chih Lin, Kaustav Banerjee, and Tim Sherwood. Introspective 3D chips. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 264–273, New York, NY, USA, 2006. ACM Press.

[7] RTI. The Economic Impacts of Inadequate Infrastructure for Software Testing. Technical Report NIST Planning Report 02-3, National Institute of Standards and Technology, May 2002.

The structure and generation of Non-graded Finite Difference Octree Grids

Vikram Aggarwal
University of California
Santa Barbara
CA, USA, 93117
vik@cs.ucsb.edu

John R. Gilbert
University of California
Santa Barbara
CA, USA, 93117
gilbert@cs.ucsb.edu

Frederick Gibou
University of California
Santa Barbara
CA, USA, 93117
fgibou@cs.ucsb.edu

ABSTRACT

Fluid flow problems are often solved computationally, and solve a major application for supercomputers and large computing clusters. The computation kernel in these problems is a linear system solve $Ax = b$. In the recent work by [2], adaptive grids are used to solve fluid flow problem, to a high degree of accuracy. While Finite Difference adaptive grids make it easier to discretize the domain, and form much simpler discretization schemes than Finite Element methods, the linear system is harder to solve. In the work by [2], for example, the matrices are unsymmetric. We aim to find out more about the structure of these matrices, specifically for 3D geometries in various cases. Towards this, we have written general purpose codes to generate such geometries, and tools to help visualize them either as static images, or video. Once we generate the Finite Element Grids, we can evaluate the matrix properties. While the matrix is unsymmetric, we note that it has a lot of structure, and examine possible features that can be exploited in a linear solver for such systems.

1. MOTIVATION

Computers are increasingly used to solve various important engineering problems, pertaining to fluid flow, materials research, among many others. In most such engineering problems, the computational kernel involves solving a linear system of the form $Ax = b$, where A is a square matrix. This linear system solve completely dominates the time required to compute the final solution. This computational cost necessitates both increased research and increased funding into faster computers, and also algorithms to speed up the linear system solve. Our work focusses on algorithmic advances, to speed up the cost of the linear solve.

A common method to solve the linear system solve involves preconditioning, where the original matrix A is multiplied by a matrix M^{-1}, called the preconditioner. While there is a large body of literature on preconditioning techniques, making a good preconditioner often relies on a combination of Linear Algebra theory and knowledge of the underlying physical process. For this purpose, we are interested in the structure of the matrix A, and in gaining insight into the physical process that generates it.

We are interested in solving large fluid flow problems, over 3D domains. The equations governing fluid flow are the Navier Stokes equations, with a Poisson equation being solved at every iteration. In the work by [1], the Level Set Method [3] and adaptive grids are used to solve this system to first order accuracy. In this case, the matrix is symmetric, and the authors use the Conjugate Gradient method to solve the linear system. The recent work by [2] discretizes the Navier Stokes equation on completely adaptive grids, and obtains second order accuracy. In the newer work, however, the higher order accuracy makes the matrix non-symmetric and thus Conjugate Gradients cannot be used. This second order technique is the main focus of this paper. The matrices of interest are generated using the technique in [2], for 3D domains. The matrix sizes can get quite large, as increasing the size allows researchers to model either larger volumes, or refine the existing volume to much finer detail.

2. GENERATOR

In our work, we investigate the structure of these matrices. First, we describe our matrix generator, called GeomOct, which is a robust, flexible tool to generate such grids. GeomOct generates general Octree geometries, over an arbitrary 3D domain. The tool is written with a view towards correctness, speed, and flexibility. Further, it can either be used as a library, or as a stand-alone program, making it attractive to researchers burdened with generating Octree geometries. Information can be stored at either the cell centers, or the cell nodes, to allow for simulation of a wide variety of physical processes. Once the octree grid is generated, helper methods are provided to iterate over all the octree nodes in depth first order, or all the corners.

For most common uses, a single program is provided, where the user has to specify the function which returns a signed distance function for the interface. This program is easy to use and extend, all a casual user needs to specify is a signed distance function.

2.1 Working

In 3D, the volume is automatically discretized using Octrees. The grid generator requires a signed distance function ϕ, where $\phi(x)$ is the distance from the fluid interface. $\phi(x)$ is negative when x is inside the fluid, and positive outside, and zero on the interface itself. GeomOct refines the vol-

Graduate Student Research Conference 2007, Santa Barbara, USA

ume around the interface to the maximum possible depth allowed by the user. While this formulation is motivated by the Level Set Method approach, it is not limited to the Level Set Method. Complex geometries, for example a hull of ship, could be modelled similarly by considering the hull-water interface, where the value of $\phi(x)$ is negative for all the points inside the hull. In either case, the refinement is carried out to surround the interface with fine cells. Elsewhere in the domain, there is no refinement. This ensures that all the computational effort is spent in the region of interest. We do not impose a graded grid, where the volume of neighboring cells differs only by a constant factor. No such artificial constraints are imposed.

Once the volume is discretized, the user can choose to run iterators over all the octree nodes, or all the corners in a straightforward manner. One such iterator assigns pressure and density values to all the cell corners. Another computes the matrix from a Finite Difference formulation. A third could print out the dependencies between the corners, for a Multigrid-like method. The sequence of iterators depends on the user's choice, and many iterators could be run on the same mesh, once it is constructed.

Since the octree is just a simple data object, there is also the possibility of keeping many octrees in memory. This could be to model different parts of the same computation, or to compare two octrees simultaneously. The amount of octrees, or the amount of refinement is limited only by the memory on the machine. We have successfully generated grids of up to 65 million unknowns with the current code. Figure 1 shows the corners of a spherical geometry. The grid has been cut away to reveal the spherical shell to the right. One can immediately notice that the density of points is the greatest near the spherical shell, and sparse when we move away from the shell. The shell is the region of interest in most computations, and this method devotes the maximum computational elements (and resources) to the region of interest.

Our tools have been designed to be of use in both matrix analysis, and in visualization. As a result, we generate matrices that can be imported into common matrix analysis tools like Matlab, or Octave. We also generate grids which can be imported into freely available visualization software, to allow researchers to generate beautiful images and movies. The tool, and all supporting code is available through our website[1] under the GNU General Public License (GPL).

3. ANALYSIS

Once the octree grids are generated, we proceed to show the structure of matrices that arise from Finite Difference approximations on these simple geometries. We examine these matrices to isolate the reasons for the non-symmetric values. We show how to approximate these matrices by symmetric matrices, and also how these matrices can be approximated by symmetric, positive definite matrices. We justify our choice of approximations by appealing to the underlying physical process. In many cases, the symmetric approximations model the eigenvalues of the original matrix closely. While this does not immediately lead to robust linear solvers for this system, it does suggest possible avenues for investigation. In order to solve large 3D grids, we believe it is crucial to get an understanding of these properties.

[1]http://gauss.cs.ucsb.edu/

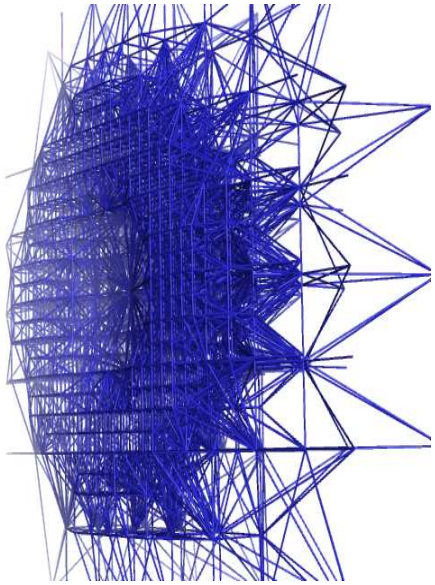


Figure 1: A cut-away of a spherical grid. Note the sparseness of points far away from the spherical shell.

4. CONCLUSION

We demonstrate a 3D grid generator for fully adaptive meshes, which is capable of being run through a library call, or as a stand-alone program. We hope the flexibility of our tool will motivate researchers to download it and generate 3D geometries. We hope that this tool also encourages more research into the matrices of 3D Octree geometries.

We analyze the matrix structure of such 3D geometries, and list some key properties of such grids, especially for the second order accurate method of [2]. In particular, we show that while the grids are unsymmetric, they have a significant symmetric component, and that symmetric approximations to these grids can be quite close. Our goal is to solve large 3D geometries, and we believe this work is promising step in that direction, and the insight gained shall be invaluable to researchers evaluating such geometries in the future.

5. REFERENCES

[1] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *SIGGRAPH*, pages 457–462, 2004.

[2] C. Min and F. Gibou. A second order accurate projection method for the incompressible navier-stokes equations on non-graded adaptive grids. *Journal of Computational Physics*, 219(2):912–929, 2006.

[3] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry*. Mathematics. Cambridge University Press, 1999.

Environmental Tomography

Stacy Patterson† Bassam Bamieh‡ Amr El Abbadi†

†Department of Computer Science
University of California, Santa Barbara
{sep, amr}@cs.ucsb.edu

‡Department of Mechanical Engineering
University of California, Santa Barbara
bamieh@engineering.ucsb.edu

1. INTRODUCTION

Mobile phones are ubiquitous computing devices. If these devices are coupled with sensors, they can provide a powerful computing platform for large-scale sensing applications. Phones can be equipped with sensors that measure the quantity of harmful pollutants such as sulfur dioxide or contaminants such as radiation, and the sensor readings acquired from these devices can be used to detect and monitor the levels of these harmful substances in populated areas. Since modern mobile phones are GPS-enabled, it is possible to record not only the concentration of the sensed phenomenon, but also the exact location of the sensor reading. This location information opens the door for the creation of detailed spatial models of the physical data distribution.

However, the mobile computing platform also presents several challenges. Mobile devices have limited power and storage capacities, and users may only be willing to contribute a small fraction of these resources to a sensing application. Users move in independent, unpredictable patterns, and a sensing application cannot expect that a device take sensor readings in predefined locations, nor can it expect that sensor readings can be taken at every point in a region. Finally, by reporting location information along with sensor readings, users are forced to reveal their locations. Users may not be willing to participate in a program that requires them to divulge this private location information.

We propose Environmental Tomography, an approach for environmental sensing and spatial data modeling using mobile devices that overcomes these challenges. Tomography has long been used in medical imaging techniques such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) [1]. For example, in a CT scan, two dimensional X-rays are taken in multiple directions, and these two dimensional projections are combined to reconstruct a three-dimensional image. Similarly, in Environmental Tomography, one dimensional projections, sums of sensor readings, are collected along fixed query paths across the region, and the projections are used to reconstruct an estimate of the underlying data distribution.

Environmental Tomography provides a number of benefits. The data collection approach is designed to overcome the challenges of mobile networks. No requirements are placed on the individual dynamics of any mobile device, and the resource requirements at each device are low. Since sensor readings are aggregated, there is no need to report the location of any device. Therefore, the privacy of user locations is preserved. The technique is also robust to query failure and partial information. Accurate estimates can be generated from a limited number of projections and can be further refined as additional results are reported.

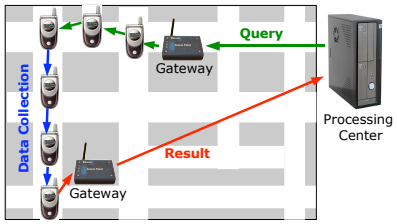


Figure 1: System Architecture

2. ENVIRONMENTAL TOMOGRAPHY

A high level view of the proposed system architecture is shown in Fig. 1. The system consists of a collection of mobile phones that are equipped with sensors and also with Bluetooth or 802.11 radios for communication with other nearby devices. Phone users may move about in arbitrary patterns, and devices may join (power on) and leave (power off) the system at any time. While individual user mobility patterns are not necessarily predictable, we expect that there is some predictability to the network as a whole. Specifically, during certain times of the day, namely rush hour, it is reasonable to expect that there is a high density of users, and therefore of mobile phones, along roads and walkways. We assume that along these paths, the mobile devices form a connected network.

The system also has several fixed gateways that are deployed throughout the sensing region. The gateways are capable of local wireless communication, and they have reliable connections to the processing center. The processing center is responsible for issuing queries for data collection and performing tomographic reconstruction on the query results to generate an estimate of the entire physical distribution.

Data Collection

The data collection process is illustrated by the arrows in Fig. 1. The processing center creates the query message which contains the specification of the *query path*. The path is defined by *start* and *end* coordinates and the distance between sensor readings. For simplicity we assume the query paths are straight line segments, though it is possible to specify more complex path types. This specification completely determines the set of coordinates, or *sampling points*, at which readings should be taken. The query also contains a *sum* field that is updated as the readings are collected.

The processing center sends the query message to the gateway that is closest to the start coordinates, where it is in-

troduced into the mobile network. To route queries, our approach relies upon two established routing protocols for mobile ad-hoc networks: a greedy cartesian routing protocol such as Greedy Perimeter Stateless Routing (GPSR) [2] and Trajectory Based Forwarding (TBF) [3]. In GPSR, the message is routed from source to destination in a greedy manner where each device forwards the message to the neighboring device that is closest to the destination. In TBF, the goal is to ensure that the message follows a specified trajectory. In this case, each device forwards the message to the neighbor that lies on or near that trajectory.

Each query is routed from the gateway to the start coordinates using greedy routing. Once the query reaches its start coordinates, TBF is used to forward the query message along the specified trajectory, the query path. Whenever a device with the message moves on (or near) a sampling point as defined in the query, the device takes a sensor reading and adds it to the *sum* field in the query. The updated query is then forwarded to the next hop along the query path. When the query reaches the end coordinates, it is routed back to the nearest gateway using greedy routing, The gateway then sends the message back to the processing center.

When the processing center receives the query results, it performs tomographic reconstruction to generate the estimate of the distribution from the aggregate measurements. We explain this process below.

Tomographic Reconstruction

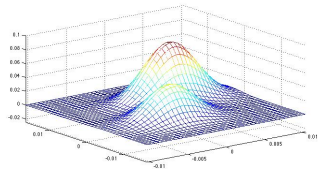
We represent the results of the sum queries by the vector m where each component m_i corresponds to the result of the i'th query path. Let f be a two-dimensional data distribution; $f(x_j^i, y_j^i)$ is the measurement at the j'th sampling point on the i'th query path. We define an operator A that generates the vector m from an underlying distribution f by computing the sums of measurements along M query paths

$$A(f) \;:=\; \begin{bmatrix} \sum_j f(x_j^1) \\ \vdots \\ \sum_j f(x_j^M) \end{bmatrix}.$$

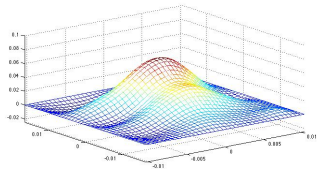
The goal of tomographic reconstruction is to find the distribution that satisfies the equation $A(f) = m$, which is a linear system of equations in the unknown f. Unlike in medical imaging where one is able to take projections along many paths in every direction, in Environmental Tomography, the choice of paths along which aggregates can be taken is restricted by the layout of the roads. This limitation makes the reconstruction problem more difficult as the system is underdetermined. There are, in fact, infinitely many solutions that satisfy $A(f) = m$. In this case, one must specify criteria that define an optimal solution $\hat{f}$ from among this set of feasible solutions. A common approach is to define the optimal solution as the solution where the L^2 norm, $||f||_2$, is minimized. However, in the case of a physical phenomenon such as pollution, the solution with the minimum L^2 norm is not the solution that most accurately reflects the physical data distribution. Therefore, we augment the optimization criteria to include physically meaningful constraints, specifically constraints relating to the properties of diffusion.

Any diffusive process, such as diffusion of a gaseous substance, satisfies the diffusion equation

$$\frac{\partial}{\partial t} f(t,x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) f(t,x,y) =: \triangle f.$$



(a) Original Distribution



(b) Estimate

Figure 2: Tomographic Reconstruction using 19 query paths along streets of Midtown Manhattan

If we also assume that the distribution f is quasi-static, then the time variation term $\partial f/\partial t$ is small, implying that $\triangle f$ should be small. Incorporating the assumptions about diffusion into our optimization, we use a minimization criterion that is a weighted combination of $||f||_2$ and $||\triangle f||_2$. Experimentally, this criterion yields estimate distributions that closely resemble the actual distributions.

The tomographic reconstruction problem can then be formulated as the following optimization problem: minimize $< f, Qf >$, where $Q := \left(I + \alpha \triangle^2\right)$, subject to the constraint $A(f) = m$. It can be shown that the optimal solution is

$$\hat{f} = Q^{-1}A^*(AQ^{-1}A^*)^{-1}m.$$

An example of an estimate generated using Environmental Tomography is shown in Fig. 2. We use a synthetic data distribution of two Gaussian curves, representing two sources of contaminant with different magnitudes. We use Midtown Manhattan as the underlying road map. The estimate is generated from only 19 query paths along city streets and avenues with 533 sampling points in total. The reconstruction produces two curves similar to the original distribution and also gives a good approximation of the location of the peaks, indicating the sources of the contamination.

3. DISCUSSION

We have shown the feasibility of a ubiquitous sensing application that preserves the privacy of user location information. Our solution is also robust to the dynamics and geographical limitations of mobile networks. We plan to expand this work to refine the reconstruction process and to address issues such as optimal query path selection, the role of mobile network density, and the effects of noisy sensor readings and GPS inaccuracies.

4. REFERENCES

[1] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. SIAM, 2001.

[2] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *MobiCom '00*, pages 243–254, 2000.

[3] B. Nath and D. Niculescu. Trajectory based forwarding and its applications. In *MobiCom '03*, pages 260–272, 2003.

Automated Size Analysis for Object-Oriented Systems

Fang Yu
Computer Science Department
University of California
Santa Barbara, CA 93106, USA
yuf@cs.ucsb.edu

ABSTRACT

This work investigates automated verification of size properties of collection types in the design languages for object-oriented systems. The goal is to automatically verify invariants about the sizes of the collections of a class with respect to the pre and post-conditions of the methods of that class. The presented approach is based on a *size abstraction* that abstracts away the contents of the collections while preserving the constraints on their sizes. A case study was conducted on the OCL specification of the Java Card API. The presented verification approach discovered specification errors in 26 out of the 150 methods in the 31 classes.

1. INTRODUCTION

Ensuring the correctness of object-oriented designs is a crucial problem due to ubiquity of object-oriented programming. Most common way of searching for errors in software systems is testing, which cannot guarantee absence of bugs, i.e., testing is not sound. Model checking, on the other hand, is a sound technique that can be used to prove system correctness, but suffers from the state explosion problem [1]. State explosion problem is particularly troublesome for collection types, where the contents of each item in a collection contributes to the size of the state space. We propose size abstraction and analysis in order to achieve scalable verification in the presence of collection types. Size abstraction abstracts away the contents of collections to prevent state explosion, and still enables sound size analysis by providing an over-approximation of the behaviors of the system.

Automated size analysis is motivated by the observation that collection sizes carry sufficient information to prove an interesting class of properties. Although we compromise precision by ignoring contents of a collection, we achieve scalable verification, and successfully leverage model checking techniques to verify real-world applications. We believe that specification and analysis of size properties is important and promising for several reasons. First, size properties are commonly used in object oriented models and they do not require an extra specification effort for software developers who use object oriented modeling languages. Second, violation of size properties is the cause of many types of security vulnerabilities such as buffer overflows. Last, effective automated verification of size properties can be achieved by using abstractions that focus on size properties, and by using domain specific and efficient automated verification techniques that target verification of systems characterized by arithmetic constraints.

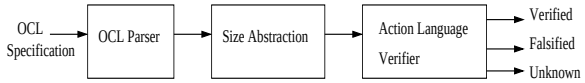


Figure 1: The OCL Size Analysis Framework

2. SIZE ANALYSIS FOR OCL

We present tools and techniques for size analysis of Object Constraint Language (OCL) specifications. OCL [5] is a specification language for describing constraints on object-oriented models. OCL is primarily used for specifying class invariants on fields and associations, and pre and post conditions of class methods. One of the most basic tools in object oriented modeling is the specification of cardinalities of associations. These specifications correspond to arithmetic constraints on the number of objects that are associated with another object. We refer to these invariants as *size properties*, since they constrain the sizes of the associations.

Our framework is depicted in Figure 1. We parse the OCL specification and automatically translate the pre and post condition of each method into its corresponding Action Language module. After this translation we use an infinite state model checker, called Action Language Verifier (ALV) [6], for *size analysis*. ALV is an infinite state model checking tool that can verify or falsify (by generating counter-examples) using approximate fixpoint computations. The checking status is unknown when the computation fails to converge.

The OCL type system consists of basic types (such as booleans and integers), user-defined types (i.e., classes), and collection types. A *Collection* is an essential data type in object-oriented modeling, since an association between two classes correspond to a relationship between one object and a collection of other objects. OCL supports three types of collections: *Set*, *Bag* and *Sequence*. We define the size abstraction using an abstraction function that transforms OCL expressions by mapping expressions on collection types to expressions on integers.

Size abstraction abstracts the contents of the collections, but preserves the constraints on their sizes. We have implemented a tool which automates this abstraction by converting OCL expressions on collections to arithmetic expressions on their sizes. Below, we demonstrate our size abstraction using the *update* method of the *PIN* class in the Java Card API 2.1.1 [4].

```
context OwnerPIN::update(newpin: Sequence(Integer),
offset:Integer,length: Integer, e:Integer)
pre: newpin->notEmpty()
     and offset >= 0
```

```
     and offset+length <= newpin->size()
     and length >= 0
post:(
1:     thrownException=thrownException@pre
2:     and self.pin->subSequence(0,length)
          =newpin->subSequence(offset, offset+length)
      )or(
3:     thrownException=thrownException@pre->including(e)
4:   and length > self.maxPINSize
      )or(
5:     thrownException=thrownException@pre->including(e)
6:   and systemInstance->notEmpty()
      )
```

The corresponding automatically generated Action Language specification is as follows:

```
module updateMod()
pre: newpin > 0 and offset >= 0
       and length + offset <= newpin
       and length >= 0 and
post:(
1:   (thrownExceptions' = thrownExceptions
2:   and tmp8 =tmp9
     and (tmp8 = length - 0 + 1 and pin' >= length
         or tmp8 = pin' and pin' < length)
     and (tmp9 = length + offset - offset + 1
           and newpin' >= length + offset
         or tmp9 = newpin'
           and newpin'< length + offset)
     ) or (
3:   thrownExceptions' = tmp10
     and tmp10 = thrownExceptions + 1
4:   and length > maxPINSize'
     ) or (
5:   thrownExceptions' = tmp11
     and tmp11 = thrownExceptions + 1
6:   and systemInstance' > 0) ); endmodule
```

In the example above we labeled the Action Language and OCL specifications to indicate the parts that correspond to each other. The formal semantics of size abstraction can be found in [7]. The abstract OCL specification conservatively approximates the behavior of the concrete specification which means that if the abstract system satisfies a size property it is guaranteed that the concrete system also satisfies the property.

3. A CASE STUDY

We conducted a case study on the Java Card API, which is the first open application programming interface for smart cards. Since smart cards are designed to be the next generation IDs, correctness of the Java Card API specification is extremely important. The OCL specification of the Java Card API [4] contains of 31 classes with 150 methods. Using ALV we were able to verify or falsify (by generating counter-examples) all the classes in the Java Card API specification. For the falsified classes, we identified the errors in the corresponding method specifications by tracing the counter examples generated by ALV.

The verification results of invariant consistency checking are shown in Table 1. ALV verified 26 out of the 31 classes and falsified the other 5 classes. Each class specification is checked within 10 seconds and 20MB. For these falsified classes, one can find a counter example violating the class invariant even if engineers implement the class methods accordingly.

Size abstraction reduces the state space of the system and, hence, the cost of automated verification, and focusing on size properties enables us to use efficient, domain specific model checking techniques for automated verification.

Class	M	R	tran+ver	Mem
AID	7	F	0.06s+0.03s	2273k
		Y	0.06s+0.06s	2322k
APDU	14	V	0.38s+0.12s	18248k
Applet	7	V	0.06s+0.01s	1532k
CardException	4	V	0s+0s	406k
CardRuntimeException	4	V	0s+0s	323k
Cipher	6	V	0.02 s+2.05s	2998k
CryptoException	2	V	0s+0s	135k
DESKey	2	V	0.01s+0.01s	422k
Dispatcher	5	V	0.01s+0.01s	635k
DSAKey	6	V	0.06s+6.2s	7840k
DSAPrivateKey	8	V	0.11s+2.61s	4170k
DSAPublicKey	8	V	0.11s+2.62s	4170k
CardRemoteObject	2	V	0s+0s	135k
JCSystem	11	F	1.08s+0.15s	18571k
		Y	1.09s+0.19s	18571k
KeyBuilder	1	V	0.01s+0s	135k
KeyEncryption	2	F	0.01s+0s	118k
		Y	0s+0s	131k
KeyPair	5	V	0s+0s	1044k
MessageDigest	3	V	0.01s+0s	397k
OwnerPIN	9	F	0.08s+0.52s	7725k
		Y	0.1s+0.4s	5091k
PIN	4	F	0.03s+0.33s	5693k
		Y	0.03s+0.23s	3670k
PINException	2	V	0.01s+0s	135k
RandomData	3	V	0s+0s	401k
RMIService	2	V	0s+0s	414k
RandomData	3	V	0s+0s	401k
RSAPrivateCrtKey	10	V	0.2s+7.31s	6087k
RSAPrivateKey	4	V	0.03s+0.05s	1008k
RSAPublicKey	4	V	0.03s+0.05s	1008k
SecurityService	3	V	0.01s+0s	520k
Service	3	V	0.01s+0s	270k
TransactionException	3	V	0s+0s	135k
UserException	3	V	0s+0s	270k

Table 1: Verification of the Java Card API OCL specification. M: No. of methods, R: Result (F:Falsify/V:Verify), tran: Translation time, ver: Verification Time. Y: Found a counter example.

4. CONCLUSION AND FUTURE WORK

We discussed automated size abstraction and analysis for object-oriented systems. The experiments indicate our abstraction is precise enough to verify/falsify target systems, while coarse enough to perform complex model checking tasks efficiently. A possible extension to this work would be applying size abstraction to other design languages, such as Java Modeling Language [3] and Alloy [2], as well as conducting other case studies.

5. REFERENCES

[1] E. Clarke, O. Grumberg, and D. Peled, Model Checking, *MIT Press*, Jan. 2000.

[2] Daniel Jackson. "Alloy: A Lightweight Object Modelling Notation." *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 2, pp. 256-290, 2002.

[3] Gary T. Leavens, Albert L. Baker, and Clyde Ruby. JML: A Notation for Detailed Design. In Haim Kilov, Bernhard Rumpe, and Ian Simmonds (editors), Behavioral Specifications of Businesses and Systems, chapter 12, pp. 175-188. Copyright Kluwer, 1999.

[4] Daniel Larsson and Wojciech Mostowski. "Specifying Java Card API in OCL." OCL 2.0 Workshop at UML 2003, San Francisco, *Electronic Notes in Theoretical Computer Science*, vol. 102 pp. 3-19, 2004,

[5] Jos Warmer and Anneke Kleppe. "The Object Constraint Language: Precise Modeling with UML." Addison-Wesley, 1998.

[6] Tuba Yavuz-Kahveci, Constantinos Bartzis, and Tevfik Bultan. "Action Language Verifier, Extended." In *Proc. CAV '05*, LNCS 3576, pp. 413-427, 2005.

[7] Fang Yu, Tevfik Bultan, Erik Peterson, "Automated Size Analysis for OCL." In *Proc. ACM SIGSOFT ESEC/FSE '07*, pp. 331-340, 2007.

An Analysis of Shadows in Camera-Light Pairs and Its Application to Multiflash Depth Edge Detection

Daniel A. Vaquero, Matthew Turk
Four Eyes Lab
Department of Computer Science
University of California, Santa Barbara
{daniel,mturk}@cs.ucsb.edu

1. INTRODUCTION

The analysis and interpretation of shadows is an important and challenging problem in computer vision. They often appear in real-world scenes, leading to failures in vision tasks such as segmentation, tracking, and recognition. On the other hand, shadows carry valuable 3D information about surfaces in the scene and can be used as a positive source of information for many applications, such as estimation of heights of buildings from aerial images [2], interactive applications [4] and non-photorealistic rendering [1]. Recently, a multiflash imaging method that exploits shadows created with lights close to the camera [3] was proposed. This technique combines shadow information from a collection of images taken using flashes at different positions in order to detect depth edges (discontinuities in the depth map of a scene as computed from the camera's point of view), which provide useful geometric information about the 3D shape of the objects.

In this work, we derive a characterization of which depth edge orientations, as projected onto the camera's image plane, can potentially be associated with cast shadows in a given camera-light pair. The usefulness of our theoretical analysis is then demonstrated on the problem of depth edge detection with multiflash imaging. We show that at least three flashes are required in order to extract all depth discontinuities from a general scene. In addition, the theory enables us to characterize the missed depth edges in a two-flash setup. Optimal light placement positions are presented and failure cases inherent to the detection algorithm are discussed. Experiments with two-flash setups and a four-flash setup illustrate the theoretical results. In this extended abstract, we outline the basic ideas and concepts involved. A more comprehensive and rigorous study is described in a submitted conference paper [5].

2. THE SHADOW SPACE OF A CAMERA-LIGHT PAIR

One question that arises when building a camera-light setup is: for which depth edge orientations and locations in the image will the light source cast a thin sliver of shadow along the edge? In order to answer this question, we start by defining, for a given point in the image, a **space of edge-shadows**. This is done by considering an horizontal edge passing through that point and having a shadow cast above it, and then rotating it by all angles from 0 to 2π radians. An **edge-shadow** $e(\theta)$ is the horizontal edge rotated by θ radians counterclockwise (with the shadow rotated together). Such space represents all possible edge orientations that might pass through a specific pixel in the image. Each orientation appears twice in the space, associated to angles α and $\alpha + \pi$, for $0 \leq \alpha < \pi$. The shadow is cast along one of the edge's sides for the first case and along the opposite side for the second case.

Another key component in our study is the **epipolar geometry** of camera-light pairs. The basic idea is to analyze how the light rays that emanate from the light source in the 3D world project onto the camera's image plane, assuming a pinhole camera model and a small baseline (i.e., close to the camera) point light source. The projection of the light source onto the camera's image plane is called the **light epipole**. Similarly, the projections of the light rays leaving the light source are called **light epipolar rays**. We can classify the geometry of the light epipolar rays into three different cases according to the relative position of the light source: for a light placed in the plane parallel to the image plane that contains the camera's center of projection, the light epipole is at infinity and the light epipolar rays are parallel; for a light in front of the camera, the light epipolar rays are radial divergent, starting from the light epipole; and for a light behind the camera, the light epipolar rays are radial convergent, pointing toward the light epipole.

In order to avoid issues that can be analyzed separately, we work under the assumptions that the light source is a point light source, the light distribution over the scene is uniform, the objects being imaged have no specular reflections ([3] includes a solution to deal with this) and the camera-light distance is large enough to produce detectable shadows in the image, but also small enough to prevent detached shadows (a multibaseline setup is suggested in [3] to address baseline issues). We then define a space $\mathcal{S} = (i, j, \theta)$, where (i, j) are coordinates of points in the image plane, and $\theta \in [0, 2\pi)$ indexes the edge-shadow $e(\theta)$ in the space of edge-shadows. The **space of possible shadows** for a camera-light pair is a subset of $\mathcal{S}$, constructed by taking, for each point (i, j) in the image plane, the light epipolar ray that passes through (i, j). Such ray defines a π-length open interval of edge-shadows that might be generated at that point, built from the simple observation that the ray's origin and the shadow are at opposite sides of the edge. Thus, by computing such interval for every (i, j) in the image plane, we obtain the subset of $\mathcal{S}$ that characterizes the edge locations and orientations that will have a shadow cast along it if a picture is taken using the camera-light pair considered.

3. MULTIFLASH DEPTH EDGE DETECTION ANALYSIS

The multiflash technique [3] is based on the principle that when an image is taken from a scene illuminated by a light source close to the camera, thin slivers of shadow are cast along depth edges. The shadow position depends on the relative camera-flash position: for example, when the flash is placed to the right hand side of the camera, shadows are cast along the left hand side of the objects. To obtain the depth edges, ratio images that accentuate the shadow regions are computed by dividing each captured image by an approximation of a shadow-free image, computed by taking the maximum of all captured images. The light epipolar rays are then traversed starting from the light epipole. Pixels are marked as depth edge pixels when a sharp negative transition in intensity is found in the intensity profile along the epipolar ray in a ratio image [3]. The following questions arise: how many flashes should we use, and how should we place them such that no depth edges are missed by the detection algorithm? For example, in a two-flash setup with lights in the camera plane and placed to the left and to the right of the camera, horizontal depth edges are missed. It is also known that each depth edge must be shadowed in at least one image and not be shadowed in at least one other image.

Building on the concept of **space of possible shadows** for a camera-light pair, we show that in a two-flash setup there will always be depth edges missed by the algorithm, even if the lights are placed outside of the camera plane. Moreover, it is optimal to place the lights in a way such that the light epipolar rays arriving at each point (i, j) in the image plane come from opposite directions. Such configurations are the ones in which the two light sources are collinear with and located on opposite sides of the camera's center of projection. Depending on where the lights are placed, the pattern of missed edges changes. For $N \geq 3$ flashes, it is possible to detect all depth edges by placing the flashes in the camera plane in positions corresponding to the N vertices of a regular N-gon, except for scenes in which there are regions where all flashes cast a shadow. This can happen in some special cases when two edges intersect. More flashes would be necessary in order to illuminate the shadowed area, or the N flashes could be rotated in such a way that one of them illuminates the problematic area – however, the problem could appear in other areas after doing this. Further research might investigate a way of iteratively analyzing the scene in order to overcome this issue.

4. EXPERIMENTS

We performed some experiments with the objective of illustrating the theory. Four configurations were built: a two-flash setup with flashes to the left and right of the camera, a two-flash setup with flashes above and below the camera, a two-flash setup with flashes in front and behind the camera, aligned with the optical axis, and a four-flash setup combining the first two setups. In the third case, as the lights are placed within the optical axis, the camera "sees" the light in front, and the light behind the camera gets physically blocked by the camera. In order to overcome these difficulties, we devised a scheme using two plate beamsplitters [5], also known as half-mirrors.

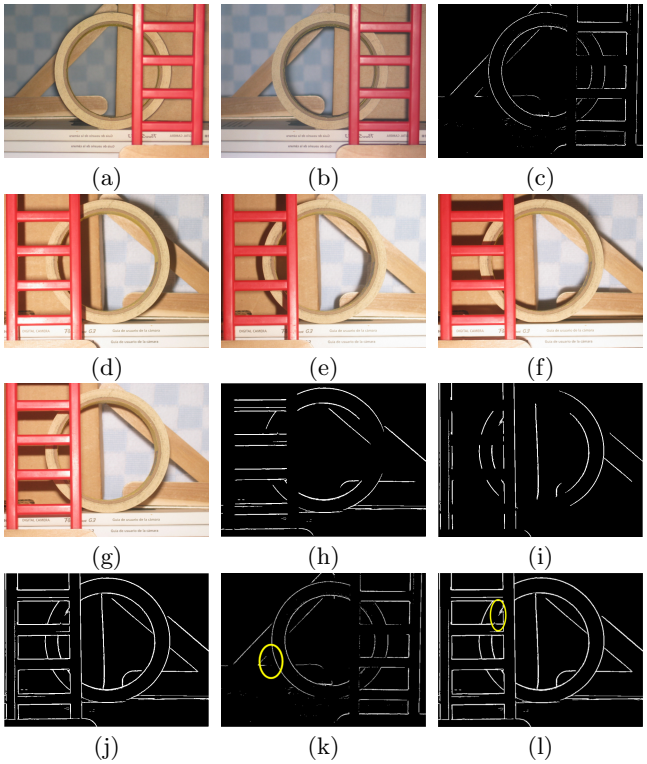


Figure 1: Depth edge detection. a) flash in front; b) flash behind; c) depth edges for front-behind; d) flash to the left; e) flash to the right; f) flash below; g) flash above; h) depth edges for above-below; i) depth edges for left-right; j) depth edges for four-flash; k-l) failure cases.

The images in Figure 1 show the obtained results. The left-right and above-below arrangements fail to detect horizontal and vertical depth edges, respectively; radial edges are difficult for the front-behind setup. Two examples of the failure case can be seen in Figures 1(k-l): the circled regions contain areas that are in shadow in all images.

5. REFERENCES

[1] D. Akers, F. Losasso, J. Klingner, M. Agrawala, J. Rick, and P. Hanrahan. Conveying shape and features with image-based relighting. In *IEEE Visualization*, 2003.

[2] R. Irvin and D. McKeown. Methods for exploiting the relationship between buildings and their shadows in aerial imagery. *IEEE Systems, Man, and Cybernetics*, 19(6):1564–1575, 1989.

[3] R. Raskar, K. Tan, R. Feris, J. Yu, and M. Turk. A non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *SIGGRAPH 2004 / ACM Transactions on Graphics*, 2004.

[4] J. Segen and S. Kumar. Shadow gestures: 3D hand pose estimation using a single camera. In *Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pages 479–485, Fort Collins, USA, 1999.

[5] D. A. Vaquero, R. Feris, M. Turk, and R. Raskar. *Submitted.*

Invited Talk

Software at Google: Tolerance in the Face of Pretty Much Everything

Russell Quong
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, California - 94043, USA

Abstract

The challenges of writing software for a large internet service company, like Google, differ in many ways from historic academic research. Classic hard problems such as super clever algorithms and fine-grained parallelization are not issues. Instead, we have to cope with lack of reliability at every level, due to both our internal size and the size of our user base. Internally, Murphy's law holds. Externally, the data we process is a free for all. We addresses these points at a high level, and gives some examples of recent work showing how we tolerate pretty much everything.

Bio

Russell W. Quong received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Palo Alto, CA. He was an Assistant Professor in the School of Electrical Engineering at Purdue University, West Lafayette, IN until 1996.

Russell worked briefly at the startup that became VA Linux and then spent 5 years at Sun Microsystems doing system architecture performance modeling. He also taught night classes at ITU for many years. He currently works for Google as a software engineer.

Characterization of Error-Tolerant Applications while Protecting Control Data

Susmit Biswas ⋆, Darshan D. Thaker †, Diana Franklin ‡,
John Oliver †, Derek Lockhart ‡, Tzvetan Metodi †, Frederic T. Chong⋆

⋆*University of California, Santa Barbara*
†*University of California, Davis*
‡*California Polytechnic State University, San Luis Obispo*
contact: susmit@uc.ucsb.edu

As the minimum feature size of process technologies continues to decrease, microprocessor designers are faced with new reliability challenges. Feature sizes of less than $0.25\mu m$ result in an increased likelihood of noise-related faults that are the result of electrical disturbances in the logic values held in circuits and on wires [3] . Natural radiation such as neutrons produced by cosmic rays and alpha particles generate electron-hole pairs as they pass through a semiconductor device. This may lead to transient faults that cause single bit upsets, which in turn may introduce a logical fault in the circuit.

A considerable amount of recent research has focused on understanding how errors in low-level circuits manifest themselves in the architecture. Much of the error-tolerance has focused on preventing any errors from affecting the running program. One can run two copies of the program, utilizing Simultaneous Redundant Multithreading. Weaver et al [6] developed techniques to reduce the probability or errors and reducing the impact of errors. Reis et al [4] propose allowing the user to switch between levels of reliability at the software level. That way, unimportant applications like web-surfing will not pay the costs, whereas banking applications would. We focus on allowing lower reliability *within* an application, as opposed to across applications.

As embedded applications become increasingly demanding, a systems approach must be applied to provide the performance needed within cost and power constraints. This paper focuses on designing embedded systems that exploit application tolerance to reduced accuracy. Such tolerance is already often used to accomodate variations in quality of service in communication and network performance. We suggest that trends in technology and usage motivate "pushing" this tolerance into the microarchitecture of embedded processors. To do so, we must understand, at an application level, how errors affect the running program.

To manage this interaction between the microarchitecture and applications, we leverage the following key observation: computations involving control are much more sensitive to inaccuracy than others [5]. We propose using static analysis to identify instructions leading to control decisions. Note that, although our focus is error-tolerant applications, our solutions are not application-specific.

We focus on application classes that do not require full accuracy to get their intended results. This can occur in several ways. First, applications that interact with human senses are very tolerant to slight inaccuracies. For example, phone lines do not carry sound perfectly, yet are sufficient for human perception. Second, there are numerical and search algorithms that expect to iterate until an adequate answer is attained.

In this study, we identify several applications that are tolerant of errors to varying degrees. All applications are part of SPEC CPU2000 or MiBench . Perceptual applications are more tolerant than decision-making applications. In order to evaluate each application, we define a fidelity measure for this application. This is typically some sort of distance from the optimal solution. For some applications, we have also defined a *fidelity threshold*, which is a subjective measure on how much inaccuracy a user would tolerate. We summarize the applications we study and the fidelities we define for each one in Table 1. All of these applications have in common the fact that they can tolerate errors in just certain areas of the algorithms. It is important in our work that we analyze where the algorithms are tolerant to errors. We will then explore how to design the harware to protect important data. Somewhat surprisingly, we find that we can perform our protection at the assembly level with an automatic compiler. The programmer identifies which functions can tolerate some error to their data, and the compiler tags instructions that do not affect the control operations. We found in a previous study[5, 1] that protecting data used for control increases the fidelity of MPEG dramatically. Our goal for perfoming data flow analysis is to identify *arthimetic instructions* that lead to a change in control flow. The technique we employ is used in contemporary compilers to determine *reaching definitions* [2], which enable optimizations such as loop-invariant code motion and copy propagation. We start at the last instruction of a basic block and move in the direction opposite program flow, tagging arithmetic instructions that *do not* influence control flow. Our compiler does not perform inter-procedural analysis. We perform static analysis at the MIPS assembly level and run the tagged executables on Simplescalar for functional simulation. For less error-tolerant applications, we introduced an absolute number of errors into the running program. For more error-tolerant applications, we introduced errors at a certain rate, expressed in errors per second (on a 1GHz machine). All of our error rates were much higher than current soft error rates in order to analyze how tolerant these applications are. In Figure 1, we show that it takes more than 100 errors per second before Susan shows any frame loss due to the SNR being too low. In addition, although the unprotected execution suffers no catastrophic errors, the fidelity is substantially lower than with protection. For Susan, disabling protection leads to very poor fidelity of output, however it does not crash the application. We find that without protecting control data, there is little to no error tolerance, even in applications that are designed for tolerating network errors. We also see that even with our static analysis, some failures do occur because of we do not perform memory disambiguation. So although we make great strides in protecting control data, we do not protect everything.

We find that with static analysis, applications can be protected such that their tolerance to errors is greatly improved. Moreover, the

Application	Description	Fidelity Measure
Susan	edge detection	Imagemagick comparison
MPEG	video encoding	% frames not dropped
MCF	vehicle scheduler	% extra time in schedule
Blowfish	encryption	% bytes correct from original
GSM	speech encoding/decoding	signal-to-noise difference
ART	image recognition	error in confidence of match

Table 1: Summary applications and their fidelity measures

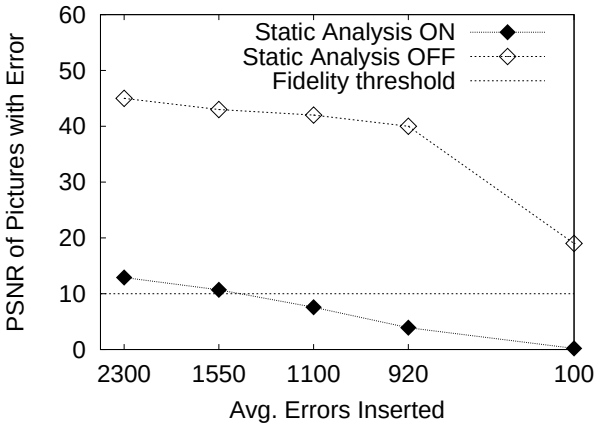


Figure 1: Susan Results

fraction of dynamic instructions related to control structures is often small when compared to overall execution. This indicates that only moderate effort is necessary for an architecture to protect these instructions through redundancy.

1. REFERENCES

[1] Darshan D. Thaker and Diana Franklin and John Oliver and Susmit Biswas and Derek Lockhart and Tzvetan S. Metodi and Frederic T. Chong. Characterization of Error-Tolerant Applications When Protecting Control Data. In *IISWC-2006*, October 2006.

[2] S. S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.

[3] N.Cohen, T.S.Sriram, N.Leland, D.Moyer, S.Butler, and R.Flatley. Soft error considerations for deep-submicron cmos circuit applications. *IEDM99*, pages 315–319, December 1999.

[4] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, D. I. August, and S. S. Mukherjee. Software-controlled fault tolerance. *ACM TACO*, 2(4):366–396, Dec 2005.

[5] D. D. Thaker, D. Franklin, V. Akella, and F. T. Chong. Reliability requirements of control, address, and data operations in error-tolerant applications. *WAR05*.

[6] C. T. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt. Reducing the soft-error rate of a high-performance microprocessor. *IEEE Micro*, 24(6):30–37, Nov 2004.

Anomaly-based Detection of State Violations in Web Applications

Marco Cova, Viktoria Felmetsger, Giovanni Vigna
Computer Security Group
{marco,rusvika,vigna}@cs.ucsb.edu

1. INTRODUCTION

In recent years, web applications have become tremendously popular, and nowadays they are routinely used in security-critical environments, such as medical, financial, and military systems. As the use of web applications for critical services has increased, the number and sophistication of attacks against these applications have grown as well.

In this work, we concentrate on workflow violation attacks, detection and prevention of which is not fully addressed by existing approaches. These attacks exploit logical errors in web applications in order to bypass the intended workflow of the application. The intended workflow of a web application represents a model of the expected user interactions with the application. Examples of workflow violation attacks include authentication and authorization bypass, parameter tampering, and code inclusion attacks.

Workflow attacks can be very difficult to detect "from the outside," that is, using sensors that only analyze HTTP requests and responses in isolation. As a result, we believe that a more effective approach to the detection of this type of attacks consists of monitoring, at runtime, the *state of the web application* "from the inside." In this context, the state of a web application at a certain point in the execution is the information that survives a single client-server interaction: in other words, the information associated with the *user session*.

We introduce a novel approach, called Swaddler, to the detection of workflow attacks that analyzes the internal state of a web application using anomaly detection techniques. We implemented our approach as an Intrusion Detection System for PHP applications.

2. APPROACH

In Swaddler, we associate each instruction of the application with a model of the state in which that instruction is normally executed. We instrument the PHP interpreter to extract state data from an application. Swaddler operates in one of two modes: *training* or *detection*. In training mode, the application is executed and models of the normal application state are derived for each basic block in the application. After these models are established, the system switches to detection: the execution of the application is monitored and any anomaly in the observed state is reported as an attack.

Swaddler consists of two main components: the *sensor* and the *analyzer*. The sensor is represented by the instrumentation code, which collects the application's state data (i.e., the values of state variables) at the beginning of each basic block, and encapsulates them in an *event* that is sent to the analyzer. An event generated by the sensor defines a mapping between the variable names and their current values. For each basic block of the application, the analyzer maintains a *profile*, i.e., a set of statistical models used to characterize certain features of the state variables. In training mode, profiles for application blocks are established using the events generated by the sensor (see Figure 1), while in detection and prevention modes these profiles are used to identify anomalous application states. When an anomalous state is detected, the analyzer raises an alert message, and, optionally, it can immediately stop the execution of the application.

3. IMPLEMENTATION

In the current prototype, the *sensor* is implemented as a module of the open-source Zend Engine interpreter [3], which implements a virtual machine that is responsible for parsing programs written in PHP and compiling them into an intermediate format, which is then executed. In our implementation, whenever the execution of a basic block of a PHP script is requested (e.g., in response to a user's request to a web application), the Zend Engine calls a handler function that passes information about state variables to the *analyzer*.

Our implementation of the *analyzer* module is based on a modified version of the `libAnomaly` framework [2]. The anomaly detection process uses a number of different models to identify anomalous states for each basic block of a web application. The task of a model is to assign a probability value to a feature of a state variable or a set of state variables associated with the block that is about to be executed. This value reflects the probability of the occurrence of a given feature value with regards to an established model of "normality." The overall anomaly score of a block is derived as the weighted sum of the probability values calculated by the models associated with the variables in the block.

`libAnomaly` provides a number of built-in models that can be combined to model different features of a variable or a set of variables. In Swaddler, we used a number of existing models to represent the normal values of single variables. These models are used to characterize such features of a variable as its normal length (*Attribute Length* model), the structure of its values (*Attribute Character Distribution* model), the set of all the possible

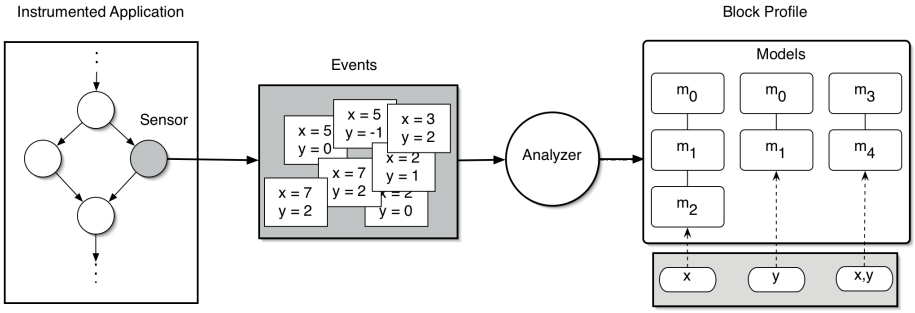


Figure 1: Swaddler in Training Phase.

Table 1: Detection effectiveness and performance overhead.

Application	Training Set Size (# requests)	Clean Set Size (# requests)	False Positives	Attack Set Size (# requests)	Attacks Detected	Avg. Instrumentation Overhead (ms)	Avg. Detection Overhead (ms)
BloggIt	9779	1586	0	15	15	5	8
PunBB	10200	1360	5	1	1	23	115
Scarf	9615	1000	1	10	10	3	13
SimpleCms	9333	1969	0	10	10	1	5
WebCalendar	19800	3300	1	1	1	5	75

values (*Token Finder* model), etc. We also developed two additional models to capture relationships among multiple variables associated with a block (a *Variable Presence or Absence model* and a *Likely Invariants* model, based on the Daikon system [1]).

4. EVALUATION

We evaluated our system on five real-world, publicly available PHP applications: a blog application, a discussion board, a conference management system, an online calendar and a content management system. These applications were chosen because they are a representative sample of the different type of functionality and levels of complexity that can be found in commonly-used PHP applications. Furthermore, each of the test applications is vulnerable to at least one workflow attacks.

The evaluation consisted of a number of tests in a live setting with each of the test applications. Attack-free data was generated by running scripts controlling a browser component to simulate user activity. In particular, we first identified the set of available user profiles (e.g., administrator and guest user) and their corresponding atomic operations (e.g., login and post a new message), and then we combined these operations to model a typical user's behavior.

We used this technique to generate three datasets: the first was used for training the `libAnomaly` models, the second for choosing suitable thresholds, and the third one was the clean dataset used to estimate the false positive rate of our system.

Since it was not sensible to collect real-world attack data by making our testbed publicly accessible, attack data was generated by manually performing attacks against each application, while clean background traffic was directed to the application by using the user simulation scripts. We used the resulting datasets to assess the detection capability of our system.

To evaluate the effectiveness of our approach we trained our system on each of the test applications. Then, we recorded the number of false positives generated when testing the application with attack-free data and the number of attacks correctly detected when testing the application with malicious traffic.

Table 1 shows the results of the tests assessing the detection capabilities and the overhead of our tool.

In our experiments, Swaddler detected all the attacks that exploited known vulnerabilities, and was also able to detect a novel file injection vulnerability that we identified in BloggIt. Swaddler raised only a few false positives, which were caused by the execution of parts of the applications that were exercised by a limited number of requests during the training phase.

Our system introduces overhead (1) to analyze and instrument the compiled code of the requested application's page ("instrumentation overhead"), and (2) to determine whether the current state is anomalous ("detection overhead"), whenever a basic block is entered during execution. Our results indicate that the total overhead is acceptable for most applications.

5. REFERENCES

[1] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 2007.

[2] The Computer Security Group at UCSB. libAnomaly Project Homepage. `http://www.cs.ucsb.edu/~seclab/projects/libanomaly`.

[3] Zend. Zend Engine. `http://www.zend.com/products/zend_engine`.

On the Representation and Multiplication of Sparse Matrices

Aydın Buluç
Combinatorial Scientific Computing Lab
Department of Computer Science
University of California, Santa Barbara
aydin@cs.ucsb.edu

John R. Gilbert
Combinatorial Scientific Computing Lab
Department of Computer Science
University of California, Santa Barbara
gilbert@cs.ucsb.edu

1. INTRODUCTION

Array-based graph algorithms have emerged as a means of solving numerous challenges of developing parallel graph algorithms. By exploiting the duality between matrices and graphs, array-based algorithms aim to apply the existing knowledge on parallel matrix algorithms to parallel graph algorithms. One of the key primitives in array-based graph algorithms is computing the product of two sparse matrices (SpGEMM) over a semiring. Most interesting graphs, such as the WWW graph, finite element meshes, planar graphs and trees, are sparse. In this work, we consider a graph to be sparse if $nnz = O(n)$ where nnz is the number of edges and n is the number of vertices. Using a dense matrix multiplication algorithm for SpGEMM is overkill since the current fastest matrix multiplication algorithm has complexity $O(n^{2.38})$ [3]. Furthermore, fast dense matrix multiplication algorithms operate on a ring instead of a semiring; which makes them unsuitable for most of the graph algorithms, especially the ones that are modeled as a path problem [1, 6].

In this paper, we present a parallel algorithm for multiplying two sparse matrices over a semiring. Our algorithm uses a 2D block data decomposition scheme for sparse matrices. To the best our best knowledge, 2D checkerboard decomposition schemes have not been studied in the context of parallel sparse matrix multiplication (ParSpGEMM) although they have been proven to be useful and more efficient than 1D decomposition schemes in the dense case [2, 4]. In addition to being more efficient, 2D block decomposition has the natural ability to express the computation as algebraic operations on submatrices (blocks).

2. MATRIX STORAGE FORMAT

The most widely used data structures for sparse matrices are CSC (Compressed Sparse Column) [5] and its transpose CSR (Compressed Sparse Row). Using CSC (or CSR) in the case of 2D block decomposed data, however, leads to inefficiencies that can not be compensated by any algorithm that holds its data in those formats. Assume, on the average there are c nonzero elements in each column of the matrix. Each processor is going to have a submatrix of size $n/\sqrt{p} \times n/\sqrt{p}$. Storing each of the submatrices in CSC storage scheme, which has $O(n+nnz)$ space requirements, would make the overall storage to $O(n\sqrt{p} + nnz)$ which is a factor of $\sqrt{p}$ worse than the optimal, compared to the space requirements of CSC on a single processor. This situation is depicted in Figure 1 where the number of nonzeros in a single column of the submatrices, $nnz(j)$, goes to zero as $p \to \infty$.

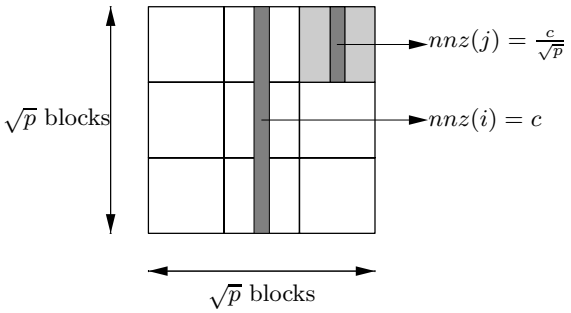


Figure 1: 2D-Decomposition of a Sparse Matrix

Consequently, we have designed a new data structure called *SparseDComp*, with three main requirements in mind:

1. The storage requirements should be $O(nnz)$
2. It should efficiently support the sequential algorithm.
3. It should allow fast access to both columns and rows.

SparseDComp is composed of two parts: DCSR (Doubly-Compressed Sparse Row) and DCSC (Doubly-Compressed Sparse Column) which are derived from CSC and CSR. Those parts can be viewed as lexicographically sorted arrays. DCSR is first sorted by rows and then by columns within each row whereas DCSC is first sorted by columns and then by rows within each column. Here, we will go over it using an example. Think about an 9-by-9 matrix A with 4 non-zeros which has the following representation in triplets format:

$$A = \{(5, 0, 0.1), (7, 0, 0.2)(3, 6, 0.3), (1, 7, 0.4)\}$$

Note that the indices start from zero. The resulting matrix A is illustrated in Figure 2 in DCSC format (DCSR is omitted for space reasons). The AUX arrays are only required to fulfill the third requirement and not used during the multiplication.

3. SEQUENTIAL ALGORITHM

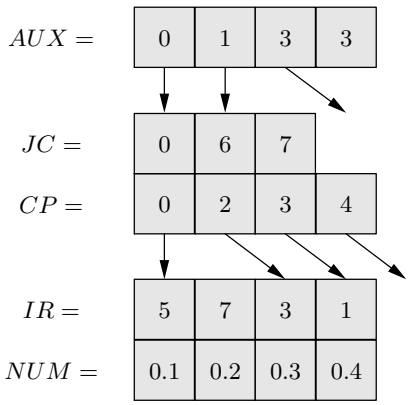


Figure 2: Matrix A in DCSC format

ParSpGEMM relies on the existence of an efficient sequential sparse matrix multiplication algorithm for doing the updates on local submatrix that each processor has. The idea behind SpGEMM is to use the outer product formulation of matrix multiplication efficiently. In outer product formulation, the i^{th} column of A and the i^{th} row of B is multiplied and we get a rank-1 matrix. Our algorithm has a preprocessing step where the intersection $Isect_c = A.dcsc.jc \cap B.dcsr.ir$ is found, which gives us the exact set of indices that we need to do the outer product. Then, the outer product formulation is transformed into Algorithm 1

Algorithm 1 Smart outer product formulation of C=A*B

1: $Isect_c = A.dcsc.jc \cap B.dcsr.ir$
2: **for all** $i \in Isect_c$ **do**
3: $\quad C = C + A(:, i) \cdot B(i, :)$
4: **end for**

We can think of each entry in $Isect_c$ as a list of size $nnz(A(:, i)) + nnz(B(i, :))$. Combining this with the fact that all the elements within a given column or row index i is sorted according to their row or column respectively values (since IR is sorted within every column/row), we conclude that the problem is very similar to k-way merging where the goal is to merge k sorted lists [1]. The only difference is that we will never explicitly contruct the lists and we will compute their elements one-by-one on demand. The merging algorithm uses a heap of size $k = |Isect_c|$, where the value of a heap entry is its NUM value and the key of a heap entry is its memory location in row/column major order layouts if we were to use a dense two-dimensional array layout to describe our matrix.

The final phase of the algorithm constructs the DCSC and DCSR structures from that lexicographically ordered stacks. This can be done in time $O(nnz(C))$ time and space as long as the final data structure is $O(nnz)$ as in the case of DCSC and DCSR.

4. PARALLEL ALGORITHM

Our parallel algorithm, which is given in Algorithm 2 in its naive form, works in a similar way with *SUMMA* (Scalable Universal Matrix Multiplication Algorithm) [4]. However, instead of broadcasting a subcolumn/subrow in every iteration as SUMMA does, our algorithm broadcasts a whole submatrix along the row/column. SUMMA indeed utilized from blocking during broadcasting by sending b subcolumns/subrows instead of one in each iteration thus reducing latency and increasing performance. The broadcasting behavior of our algorithm may be viewed as an exaggerated blocking case where we send $b = n/\sqrt{p}$ subcolumns/subrows in every iteration.

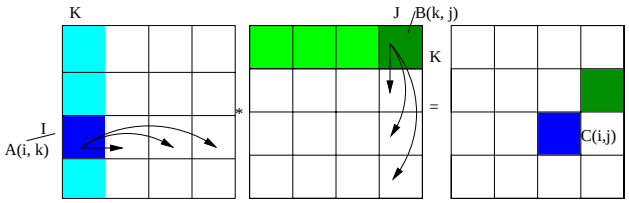


Figure 3: Execution of ParSpGEMM

Algorithm 2 Naive ParSPGEMM

Require: A is $n \times n$, B is $n \times n$
1: $myrow \leftarrow world.rank()/\sqrt{world.size()}$
2: $mycol \leftarrow world.rank()\%\sqrt{world.size()}$
3: $LocalC \leftarrow zeros(n, n)$
4: **for** $k = 0$ to $\sqrt{world.size()}$ **do** {Executes $\sqrt{p}$ times}
5: $\quad$ **if** $k == mycol$ **then**
6: $\quad\quad RowBroadcast(LocalA)$;
7: $\quad$ **else**
8: $\quad\quad ARecv \leftarrow RecvBroadcast()$;
9: $\quad$ **end if**
10: $\quad$ **if** $k == myrow$ **then**
11: $\quad\quad ColBroadcast(LocalB)$;
12: $\quad$ **else**
13: $\quad\quad BRecv \leftarrow RecvBroadcast()$;
14: $\quad$ **end if**
15: $\quad LocalC += ARecv \times BRecv$
16: **end for**

5. REFERENCES

[1] A. V. Aho and J. E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1974.

[2] L. E. Cannon. *A cellular computer to implement the kalman filter algorithm*. PhD thesis, Montana State University, 1969.

[3] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6, New York, NY, USA, 1987. ACM Press.

[4] R. A. V. D. Geijn and J. Watts. SUMMA: scalable universal matrix multiplication algorithm. *Concurrency: Practice and Experience*, 9(4):255–274, 1997.

[5] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in matlab: design and implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992.

[6] R. E. Tarjan. A unified approach to path problems. *J. ACM*, 28(3):577–593, 1981.

Modeling and Simulation of Protein-Protein Interactions in the Endoplasmic Reticulum

Marc Griesemer
Department of Computer Science
University of California
Santa Barbara, CA 93106
marcgri@cs.ucsb.edu

Linda Petzold
Department of Computer Science
University of California
Santa Barbara, CA 93106
petzold@cs.ucsb.edu

ABSTRACT

This research explores modeling and simulation of cellular systems in order to understand molecular interactions between proteins. Specifically, it models protein-protein interactions involved in translocation in the endoplasmic reticulum (ER) of yeast cells. This computational work complements experiments performed at the University of Delaware and modeling work in the Doyle Group (Chemical Engineering) at UCSB.

1. INTRODUCTION

Protein-protein interactions underlie the functions of the cell. The endoplasmic reticulum (ER) is the staging ground for protein-related processes in eukaroytic cells. In translocation, newly synthesized proteins enter the ER and are immediately sought by chaperones, specific proteins that aid in protein folding, maturation, and transport [3]. BiP is the resident molecular chaperone in the ER of yeast (*Saccharomyces cerevisiae*). It binds to protein chains at pores on the ER's surface membrane and prevents them from backsliding out of the ER. It is not known, however, how inhomogeneous concentrations of BiP drive its role in translocation. It is also uncertain to the extent BiP interacts with the cochaperone Sec63, which enhances BiP binding to proteins through structural changes in the chaperone.

Our goals in this modeling effort are twofold: (1) to generate an improved understanding of protein interactions accounting for spatialization and effects; and (2) to examine molecular gradients due to inhomogeneous concentrations of the chaperone BiP in the ER. *In vivo* laboratory experiments at the University of Delaware will attempt to verify our findings and allow for further refinements of the model.

2. MODELS

Modelers have attempted to discern BiP's role in assisting translocation of proteins into the ER. Our work is not specifically concerned with the mechanism of translocation itself, but rather how the BiP-Sec63 interaction enhances translocation [2]. Experiments have posited that Sec63 recruits BiP to the membrane surface to perform translocation. Given that the populations of BiP and the other proteins in the system range from tens to hundreds of thousands, a deterministic model of molecular concentrations seems to be justified. To this end, we first constructed an ODE model, and then extended it to a PDE model to capture the spatiotemporal dynamics.

2.0.1 ODE model

Our core model is described by a system of ordinary differential equations (ODEs). It is a 7 state, 13 parameter model that represents the interactions of BiP with the cochaperone Sec63 (J) and unfolded proteins (U), and is shown as a schematic in Figure 1. We conducted simulations and collected time series data of the concentrations of each state. The results confirmed an important point; that an excess of BiP in the ER leads to behavior in which BiP strongly interacts with other proteins.

2.0.2 PDE Model

Proceeding from the ODE model, our next objective was to describe the distribution of BiP in the ER due to spatial effects. The partial differential equation (PDE) model describes translocation. The model incorporates: (1) chemical reactions representing transitions between states in the ODE system which take place on the surface membrane (or reaction zone), and (2) diffusion into the lumen (interior) of the ER. This spatially dependent system of equations was approximated by the method of finite differences (Figure 2). The method of lines transforms the PDE into a series of ODEs which then can be solved easily by numerical software.

3. RESULTS

Using the DASSL(DASPK) ODE/DAE solver [1], we ran simulations on the model until all species reached steady-state. The system starts from conditions where all the BiP is free and present in the reaction zone. Diffusion is fast, equalizing the gradients of free BiP across the ER (< 1 ms). Reactions then take place on a slower timescale (< 5 s). The output was the concentration of each state in the reaction zone and in the lumen.

A measurement of the ER PDE model is to determine the ratio of surface concentration of BiP versus the concentration in the lumen. This gives an indication of the homogeneity of BiP and can be verified experimentally. The main simulation scenario is where free BiP to diffuse while the BiP bound to the other players (namely Sec63 and unfolded protein) remains in the reaction zone. One can then attempt to make predictions for BiP and translocation.

We found that BiP preferentially remains in the reaction zone, with a concentration of 7.06 times than in the lumen of the ER. This result illustrates that BiP does interact strongly with unfolded protein in the process of transloca-

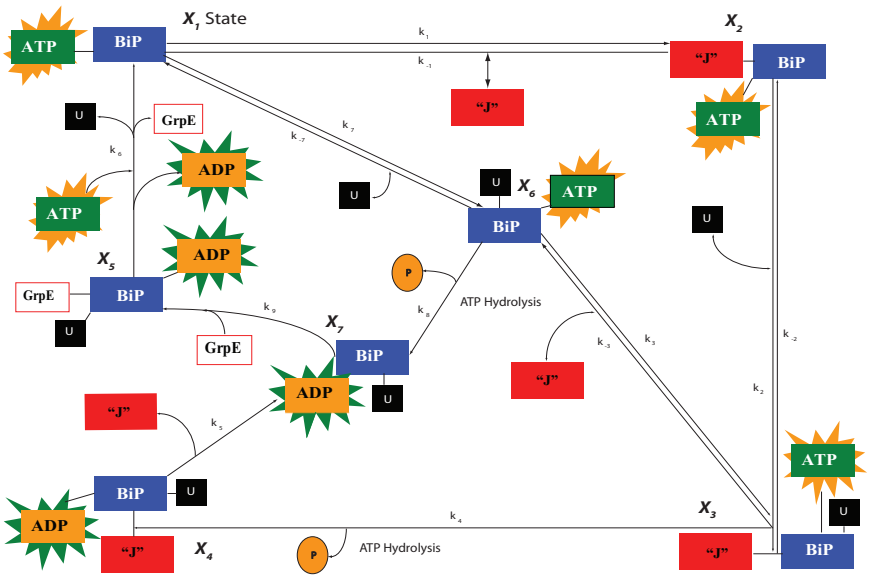


Figure 1: The ODE model consisting of 7 states, representing the interaction of BiP with the other proteins in the system.

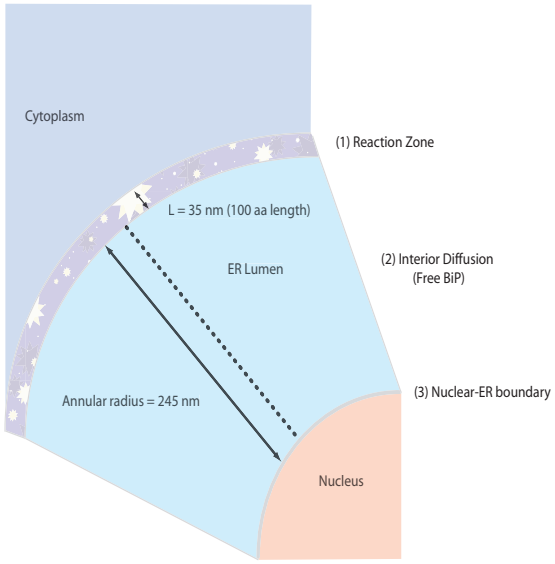


Figure 2: The PDE model consists of a surface zone and ER lumen represented by reaction-diffusion equations.

tion. In contrast, when translocation was inhibited, the concentration ratio of BiP in the ER was a base level of 1.44.

4. CONCLUSION

In creating these models, we have taken a step in making predictions of determining the relative importance of BiP in the translocation of proteins. The model can be extended, if necessary, to incorporate more complicated interactions, or to incorporate stochasticity.

5. ACKNOWLEDGEMENTS

This research was supported by NSF Grant DGE02-21715.

6. REFERENCES

[1] K. E. Brennan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial Value Problems and Differential-Algebraic Equations*. SIAM Classics Series, Philadelphia, Pennsylvania, 1996.

[2] T. Elston. Models of posttranslational protein translocation. *Biophysical Journal*, 76(5):2235–2251, May 2000.

[3] S. I. Nishikawa, J. L. Brodsky, and K. Nakatsukasa. Roles of molecular chaperones in endoplasmic reticulum quality control and er-associated degradation. *Journal of Biochemistry*, 137(5):551–5, May 2005.

Searching for Rare Objects using Index Replication

Krishna P. N. Puttaswamy, Alessandra Sala and Ben Y. Zhao
Computer Science Department, U. C. Santa Barbara
{*krishnap, ravenben*}*@cs.ucsb.edu sala@dia.unisa.it*

I. Introduction

Searching for objects is a fundamental problem faced by unstructured peer-to-peer file-sharing networks. Various algorithms such as Flooding, Random Walks [1] and their variants have been proposed to address this problem. However, most of these algorithms are only effective for locating popular objects, including algorithms used by popular deployed networks such as Gnutella and Kazaa. Studies have shown that in the widely-used Gnutella network, as much as 18% of all queries return no responses even when results are available [3]. Finding rare objects (those with few replicas) in unstructured networks is generally ineffective (in terms of search success) and inefficient (in terms of overhead and response time) compared to similar operations in their structured counterparts.

Existing work has explored several approaches to improve search recall. One approach is to use higher Time To Live (TTL) for search algorithms. However, higher TTL values significantly increases overall bandwidth consumption, and provide diminishing returns, particularly for coarse-granular search algorithms such as Flooding [3]. An alternative is to utilize object replication strategies to improve search success. But these techniques require replicating entire objects, incurring significant overheads for storage and network bandwidth and limiting their applicability.

Because these unstructured file-sharing systems account for a major portion of traffic in today's Internet [5], any technique to improve search recall for rare objects must be light-weight, effective, and easily deployable. Our focus in this work is to answer the following question: *Can we develop simple techniques to improve the search effectiveness for rare items, and also reduce the bandwidth overhead incurred?*

We believe intelligent index replication can provide the solution to this question. Not only does proactive index replication incur much lower overhead compared with data replication, previous work has shown it to be effective at improving the scalability of unstructured networks [1]. Our work explores the use of multi-hop index replication, which can significantly improve the cover region or effective search space of these overlays, while incurring little overhead compared to alternatives such as data replication. We explore the effectiveness of two-hop index replication, and propose different variants that traverse the overhead-performance tradeoff. To quantify the impact and feasibility of our proposed techniques, we evaluate them in a full-system Gnutella simulation using real measurement traces.

II. System Design

Existing unstructured networks have one main problem: they are highly limited in their efficiency to locate rare items.

Two main techniques lie at the core of our solution to this problem: Supernode-Constrained Random Walk (SCRW), and two-hop index replication. The main intuition behind them is to use the high-capacity nodes to build search-freeways, which the queries enter once and exit with the object (or its pointer). In order to ensure that queries terminate with high success rate, we need to efficiently place the indices on these search-freeway nodes. Our topology construction and search algorithms address the former while our two-hop replication achieves the latter.

1) Topology Construction and Search: A careful inspection of the topology construction algorithms used in state-of-the-art overlays [1], and in deployed unstructured overlays [2], reveal that they are similar in that they use high capacity nodes as main paths for search and try to propagate the index from low capacity nodes to these main paths. Based on this observation, our algorithm uses Gia's topology adaptation algorithm while building the network topology and assign indegree to nodes based on their capacity. Along with this, it tags the nodes with very high capacity as supernodes forming a stable path with large capacity.

In addition, we use supernode-constrained random walk (SCRW) as our search algorithm. The main difference between SCRW and a normal random walk is that in SCRW the nodes always forward the query to one of its randomly selected supernode neighbors – not just any random neighbor. If no supernodes are found in its routing table, then the node forwards the query to one of its neighbor selected at random.

2) Index Replication: One-hop replication is shown to scale unstructured networks significantly. We extend this one-hop replication to two hops in this work. Note that extending index replication to two hops might lead to an explosion in the amount of index stored on high-capacity nodes. So, managing the index replication and storage overhead becomes critical. We explore three different replication strategies.

Full replication. In this strategy, each node sends its index to all of its one-hop neighbors in its routing table, just like in one-hop replication. All of the one-hop neighbors, in turn, forward this index to all of their one-hop neighbors, except the source node. This strategy, effectively, reduces to a two-hop flooding of indices around the nodes.

Square-root replication (Sqrt). In this strategy, each node does a one-hop replication first. All the one-hop supernode

neighbors – only the supernode neighbors, randomly pick a small set of their supernode neighbors and forward the index to them. This set size is equal to the square-root of the number of supernodes of these one-hop supernode neighbors. Thus, this strategy is a simple one-hop replication augmented with square-root replication only at the supernodes. The main intuition behind this strategy is that by replicating on the supernodes, we are favoring SCRW to find objects quickly while reducing the amount of replication and its cost.

Constant replication. Instead of choosing different number of supernode neighbors for replication, in this strategy, we fix the number of supernode neighbors chosen to a constant. Every node does a one-hop replication of indices. Following this, all the one-hop non-supernode neighbors do another one-hop replication, while the supernodes propagate the index to only a constant number of their supernodes. The main reason of choosing this algorithm is to reduce the indexing load on supernode further.

Detailed theoretical analysis of each of these approaches can be found in [4]. We show that Sqrt replication provides the near-optimal tradeoff in the performance-overhead sprecturm.

III. Gnutella Full-System Simulation

To understand the performance of various search and replication algorithms in a real unstructured network, we simulated a complete Gnutella-like network. To make our simulation realistic, we obtained the network topology, the files stored in the nodes, the number of files stored, and the file distribution from Gnutella measurement study traces [6]. We extracted a Gnutella network topology with approximately 72K ultrapeers and 760K leafpeers from one of these topology traces. Since the leafpeers do not participate in query forwarding in Gnutella, we considered only the ultrapeers in our network topology. Then we extracted the list of files stored on 72K random nodes in the Gnutella trace and assigned them to the nodes in our topology. Researchers have empirically shown that this randomized placement of files on nodes approximately represents the real network, since there is very little correlation between file locations and the network topology [6]. There were approximately 27 million files, in total, with 7 million unique files assigned to nodes in our network. Since we are using the real traces to build our network, the object popularity in the network follows the same popularity we see in a real network.

To evaluate the search recall for rare objects, we preprocessed the files on the nodes in our network and queried for only the files with exactly 3 replicas in the network (approx. 300K of them). Furthermore, to make a fair comparison across the different search techniques, we explicitly limited the bandwidth that can be consumed by our tests. We limited the number of messages that could be forwarded in the entire network, and once this overhead cap is reached all search messages are dropped wherever they are in the network. No additional messages are forwarded and no new search queries are introduced.

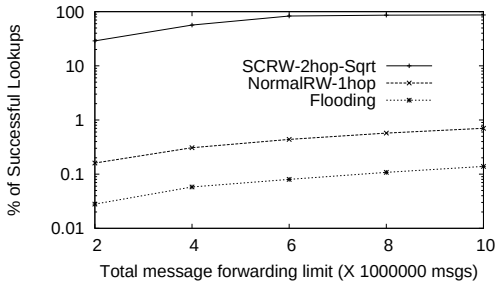


Fig. 1. Comparison of lookup success of different strategies under bandwidth cap. Note the log scale on y-axis.

We examine the performance of three different search algorithms in this setup. In addition to the SCRW with Sqrt two-hop index replication and Gia's biased randomwalk, we implemented a simple flooding algorithm. While this flooding technique is more reminiscent of early versions of Gnutella, we are flooding between superpeers, unlike the original flat Gnutella network. We experimented with different flooding depths and found that a flooding depth of 3 provides good performance in our experiment setup. We compare this against a random walk depth of 500 for the Gia and SCRW. Note that random walk with depth of 500 incurs significantly less overhead than flooding with TTL of 3, which on average reaches 42K nodes.

Figure 1 shows the performance of the three algorithms, for same overhead, expressed as a percentage of total search queries that finish successfully. We see clearly that SCRW with Sqrt replication is more than 600 times better than simple flooding, and nearly 200 times better than normal RW with just one-hop replication. The improvement decreases at higher TTL values mainly because of the diminishing returns in lookup success experienced by SCRW.

IV. Conclusion

While unstructured file-sharing networks have been successful at delivering popular content to its users, its low search recall of rare objects has limited its effectiveness and flexibility. We explore the effectiveness of multi-hop index replication, which is easily-deployable and lightweight in overhead. We then evaluate our approach on a large simulation networks from Gnutella measurements. Results show that using the same search bandwidth as flooding techniques, our technique would improve lookup of rare objects from less than 0.1% to more than 80%.

References

[1] Chawathe, Y., et al. Making gnutella-like p2p systems scalable. In *Proc. of SIGCOMM* (August 2003).
[2] Limewire. http://www.limewire.org.
[3] Loo, B. T., et al. Enhancing p2p file-sharing with an internet-scale query processor. In *Proc. of VLDB* (2004).
[4] Puttaswamy, K. P. N., Sala, A., and Zhao, B. Y. Searching for rare objects using index replication. Tech. Rep. 2007-09, UC Santa Barbara, July 2007.
[5] Saroiu, S., et al. An analysis of internet content delivery systems. In *Proc. of OSDI* (December 2002).
[6] Zhao, S., Stutzbach, D., and Rejaie, R. Characterizing files in the modern gnutella network: A measurement study. In *Proc. of MMCN* (January 2006).

Evaluating the Impact of Xen on the Performance of NAS Parallel Benchmarks

Lamia Youseff[α] Rich Wolski[α] Brent Gorda[β] Chandra Krintz[α]

[α]Department of Computer Science
University of California, Santa Barbara
{lyouseff, rich, ckrintz}@cs.ucsb.edu

[β] Lawrence Livermore National Lab (LLNL)
bgorda@llnl.gov

ABSTRACT

In this work, we investigate the efficacy of using paravirtualizing software for performance-critical HPC kernels and applications. We present a comprehensive performance evaluation of Xen, a low-overhead, Linux-based, virtual machine monitor, for paravirtualization of HPC cluster systems at LLNL. We investigate subsystem and overall performance using a wide range of benchmarks and applications. We employ statistically sound methods to compare the performance of a paravirtualized kernel against three Linux operating systems: RedHat Enterprise 4 for build versions 2.6.9 and 2.6.12 and the LLNL CHAOS kernel. Our results indicate that Xen is very efficient and practical for HPC systems.

1. INTRODUCTION

Virtualization is a widely used technique in which a software layer multiplexes lower-level resources among higher-level software programs and systems. Historically, it has come at the cost of performance due to the additional level of indirection and software abstraction necessary to achieve system isolation. Recent advances in Virtual Machine Monitors (VMM) technology however, address this issue with novel techniques. One such technique is paravirtualization which is the process of strategically modifying a small segment of the interface that the VMM exports along with the OS that executes using it. Consequently, Systems have the potential for improved scalability and performance over prior VMM implementations. A large number of popular VMMs employ paravirtualization in some form including Xen [6], and VMWare [4]. VMM systems, in addition enable application and full-system isolation (sand-boxing), OS-based migration, distributed load balancing, OS-level check-pointing, non-native (cross-system) application execution, and support for multiple or customized operating systems.

Despite the potential benefits, performance advances, and recent research indicating its potential [5, 10], virtualization is currently not used in high-performance computing (HPC). One reason for this is the perception that the remaining overhead that VMMs introduce is unacceptable for performance-critical applications and systems. The goal of our work is to evaluate empirically the degree to which this perception is true. In particular, we rigorously investigate the overhead imposed by the Xen paravirtualization system for HPC systems using current HPC commodity hardware at LLNL. In this abstract, we present our evaluation of the impact of paravirtualization on the performance of NAS parallel benchmarks (NPB). NPB evaluate the efficency of highly parallel HPC systems in handling crtitical operations that are part of the simulation of future space missions.

2. METHODOLOGY

Our experimental hardware platform consists of a four-node cluster of Intel **E**xtended **M**emory 64 **T**echnology (EM64T) machines. Each node consists of four Intel Xeon 3.40 GHz processors, with a 4GB of RAM. The nodes are interconnected with an Intel PRO/1000, 1Gigabit Ethernet network fabric. We perform our experiments by repeatedly executing the benchmarks for 50 runs and collecting the performance data to compute the average across runs. In addition, We perform a *t-test* at the $\alpha \geq 0.95$ significance level to compare the means of two sets of experiments.

We empirically compare four different HPC Linux operating systems. The first two are current releases of the RedHat Enterprise Linux 4 (RHEL4) system: v2.6.9 and v2.6.12. We also evaluate the CHAOS kernel. CHAOS is the **C**lustered, **H**igh-**A**vailability, **O**perating **S**ystem [3] from LLNL, which is based on RHEL v2.6.9. Our Xen-based Linux kernel is RHEL4 v2.6.12 with a Xen 3.0.1 patch.

We used a set of HPC benchmarks, which consists of micro, macro-benchmarks, and real HPC applications to evaluate the performance of the different subsystems as well as the full system. Our micro-benchmark subset includes programs from the HPC Challenge and LLNL ASCI Purple Benchmark suite, and are specifically designed to evaluate distinct performance characteristics of the different subsystems; including computational, communicational, memory and disk I/O characteristics. To evaluate the full system, we employ several popular macro-benchmarks from the NAS Parallel benchmark suite [2]. We also used a real HPC application, but we include here only the performance analysis of the NPB benchmarks due to space limitation. For the detailed study, readers are encouraged to refer to our papers [7, 8, 9].

3. NPB PERFORMANCE ANALYSIS

NAS parallel benchmarks (NPB) mimic the computational, communicational and data movement characteristics of large scale computational fluid dynamics applications. Figure 1 shows the performance of the NPB codes (x-axis) for our different kernels relative to CHAOS (y-axis). *EP, IS* and *MG* are the Embarrassingly Parallel, Integer Sort, and Multi grid codes respectively, while *LU* is the Linear solver code and *CG* is the Conjugate gradient code. We present two

different metrics for each of the five benchmarks. The left five sets of bars reflect total execution time (lower is better), while the right five are for the total millions of operations per second (Mops) the benchmarks achieve (higher is better).

All of the kernels perform similarly for EP, IS, and MG. The little differences between the bars, though visually different in some cases, are not statistically significant using the t-test. This is interesting since the benchmarks are very different in terms of their behavior: EP performs distributed computation with little communication overhead, IS performs a significant amount of communication using collective operations, and MG employs a large number of blocking send operations. In all cases, paravirtualization imposes no statistically significant overhead.

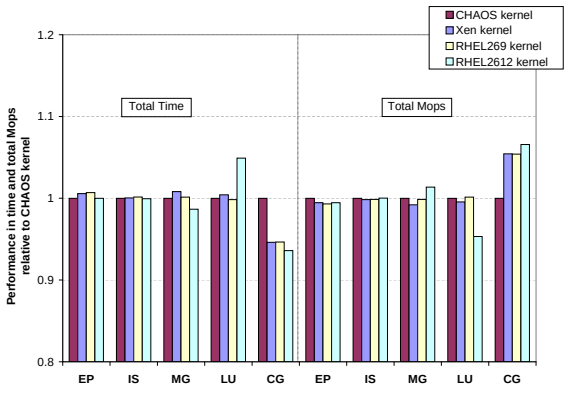


Figure 1: shows the NAS Parallel benchmarks performance relative to CHAOS.

On the other hand, LU decomposition shows a performance degradation of approximately 5% for RHEL2.6.12 for both total time and Mops. The reason for this is due to overhead this kernel places on computation. CHAOS optimizes this overhead away and RHEL2.6.9 makes up for this loss due to its low overhead on MPI-based network latency.Xen implements a different CPU scheduling policy: a very efficient implementation of the borrowed virtual time (BVT) scheduler [1]. However, Xen network performance places a subtle performance penalty on MPI-based LU code performance. A combination of the scheduling policy and network performance enabled by Xen enables the Xen system to avoid the overhead which was endured by RHEL2.6.12.

The *C*onjugate *G*radient (CG) code computes an approximation to the smallest eigenvalue of a large sparse matrix. CG executes slower using CHAOS than using the other kernels by about 5%. The statistical difference however was not significant. In summary, Xen performs consistently comparable to CHAOS and the two RHEL kernels and delivers performance similar to that of natively executed parallel applications.

4. CONCLUSIONS

Paravirtualizing systems such as Xen, expose opportunities for improved maintenance and customization for HPC systems. In this work, we evaluate the overhead of using Xen in an HPC environment. We compare three different Linux configurations against a Xen-based kernel. We perform experiments using micro- and macro-benchmarks from the HPC Challenge, LLNL ASCI Purple, and NAS parallel benchmark suites among others, as well as using a large-scale, HPC application for simulation of oceanographic and climatologic phenomena. As a result, we are able to rigorously evaluate the performance of Xen-based HPC systems relative to non-virtualized system for subsystems independently and in ensemble. Our results indicate that, in general, the Xen paravirtualizing system poses no statistically significant overhead over other OS configurations currently in use at LLNL for HPC clusters – even one that is specialized for HPC clusters

As part of future work, we are investigating techniques for high-performance check-pointing and migration of full systems to facilitate load balancing, to isolate hardware error management, and to reduce down time for LLNL HPC clusters. We are also investigating techniques for automatic static and dynamic specialization of OS images in a way that is application-specific [5, 10].

5. REFERENCES

[1] K. Duda and D. Cheriton. Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose schedular. In *Symposium on Operating Systems Principles*, pages 261–276, 1999.

[2] D. B. et al. The NAS Parallel Benchmarks 2.0. *The International Journal of Supercomputer Applications*, 1995.

[3] R. B. et al. Achieving Order through CHAOS: the LLNL HPC Linux Cluster Experience, June 2003.

[4] J. Sugerman et al. Virtualizing I/O devices on VMware workstations hosted virtual machine monitor. In *USENIX Annual Technical Conference*, 2001.

[5] C. Krintz and R. Wolski. Using phase behavior in scientific application to guide linux operating system customization. In *Workshop on Next Generation Software at IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, April 2005.

[6] P. Barham et al. Virtual machine monitors: Xen and the art of virtualization. In *Symposium on Operating System Principles*, 2003.

[7] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Evaluating the performance impact of xen on mpi and process execution for hpc systems. In *Proceedings of the First International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, November 2006.

[8] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Paravirtualization for hpc systems. In *Proceedings of Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC)* best paper award winner, December 2006.

[9] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Paravirtualization for HPC Systems. Technical Report Technical Report Numer 2006-10, Computer Science Department University of California, Santa Barbara, Aug. 2006.

[10] L. Youseff, R. Wolski, and C. Krintz. Linux kernel specialization for scientific application performance. Technical Report UCSB Technical Report 2005-29, Univ. of California, Santa Barbara, Nov 2005.