GSWC 2008

Proceedings of

The Third Annual Graduate Student Workshop on Computing

October 3rd, 2008
Santa Barbara, California

Sponsored by

Google

Microsoft

YAHOO!

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

Organized by

Fred Chong, *Advisor*

Cha Lee, *Co-chair/Industry Liaison*

Bita Mazloom, *Co-chair/Publishing Chair*

Petko Bogdanov, *Review Chair*

Chris Coffin, *Local Arrangement*

Ramya Raghavendra, *General Committee*

Yenting Ng, *General Committee*

Mohit Tiwari, *General Committee*

Fang Yu, *General Committee*

Sponsored by

Keynote Speakers

Peter Norvig, Google Research Director

Peter Norvig is a Fellow of the American Association for Artificial Intelligence and the Association for Computing Machinery. At Google Inc he was Director of Search Quality, responsible for the core web search algorithms from 2002-2005, and has been Director of Research from 2005 on. Previously he was the head of the Computational Sciences Division at NASA Ames Research Center, making him NASA's senior computer scientist. He received the NASA Exceptional Achievement Award in 2001. He has served as an assistant professor at the University of Southern California and a research faculty member at the University of California at Berkeley Computer Science Department, from which he received a Ph.D. in 1986 and the distinguished alumni award in 2006. He has over fifty publications in Computer Science, concentrating on Artificial Intelligence, Natural Language Processing and Software Engineering, including the books Artificial Intelligence: A Modern Approach (the leading textbook in the field), Paradigms of AI Programming: Case Studies in Common Lisp, Verbmobil: A Translation System for Face-to-Face Dialog, and Intelligent Help Systems for UNIX. He is also the author of the Gettysburg Powerpoint Presentation and the world's longest palindromic sentence.

Barney Pell, Microsoft Live Search Powerset

Barney Pell is Partner, Search Strategist and Evangelist at Microsoft. Dr. Pell was most recently Founder, CEO and CTO of Powerset, a VC-funded natural-language search company. Dr. Pell co-founded Powerset in 2005 after spending a year as Entrepreneur in Residence at Mayfield, a top venture capital firm in Silicon Valley. Powerset licensed and commercialized natural language technology developed over 30 years at Xerox PARC, growing to 70 people before being acquired by Microsoft in August 2008. Dr. Pell is a recognized expert and frequent speaker on topics including AI, search, and semantic technology. For over twenty years, Dr. Pell has pursued technical and commercial innovation in Artificial Intelligence as a researcher, manager, strategist and entrepreneur. Before joining Mayfield, Dr. Pell worked for NASA Ames Research Center as a Principal Investigator and Area Manager responsible for innovation in autonomous systems, human-centered computing, spoken dialog systems, search, collaboration, and the semantic web. Highlights of innovations during Dr. Pell's career at NASA include the development of the Remote Agent, the first AI system to fly onboard a deep space probe, mission critical software to support planning and science collaboration for the Mars Exploration Rovers mission, and Clarissa, the first spoken dialog system to fly in space.

Dr. Pell's research experience also includes time in the natural language processing group at SRI International, at the Swedish Institute for Computer Science, and at the Electro-Technical Laboratory in Japan. His entrepreneurial experience also includes serving as Chief Strategist and Vice-President of Business Development at StockMaster.com, a provider of internet-based stock-market analysis tools, and Vice President of Strategy for Whizbang! Labs, a provider of advanced text processing and search engine software. Dr. Pell also serves as an advisor or board member to several startup companies and educational nonprofits.

Discussion Panel

Arnab Bhattacharjee, Director of Engineering for Yahoo! Search Technology

Arnab is a search technology leader at Yahoo with 10 years of experience at Inktomi and then Yahoo Search. He leads a team of software engineers building crawling, web graphing and structured information extraction technology for web search. He received his BS in Computer Science from the Indian Institute of Technology and an MS in Computer Science from the University of Pennsylvania.

Herb Wildfeuer, Principal Engineer at Cisco Systems

Herb Wildfeuer is Principal Engineer in the Access Routing Technology Group of Cisco Systems. Herb is also Site Manager of the Goleta offices of Cisco that host the Voice and Video Digital Signal Processing engineering teams. Herb is currently involved in technology and product development in the fields of unified communications, IP telephony, video conferencing and IP video surveillance. Herb received the inaugural Pioneer Technology Award at Cisco for his innovative work in the area of Voice over IP and holds 11 patents in the area of IP telephony.

Thorsten von Eicken, CTO and Founder of RightScale

Before joining RightScale Inc., Thorsten was Chief Architect at Expertcity.com and Citrix Online, makers of GoToMyPC, GoToMeeting, GoToWebinar, and GoToAssist. He was responsible for the overall architecture of these online services and also managed the 24/7 datacenter operations which allowed him to acquire deep knowledge in deploying and running secure scalable online services. Thorsten received his Ph.D. from UC Berkeley and was a professor of computer science at Cornell University.

Simon Towers, Director of Adobe's Advanced Technology Labs

Simon is a director of a research lab at Adobe Systems. His research interests include the technologies for building Rich Internet Applications (ranging from virtual machines, compilers and development environments to security and web analytics) as well as advancing the notion of electronic documents (for example, dynamic re-layout to target different devices). At Adobe, he has also undertaken such tasks as leading the CEO's Technology Council and running programs to evaluate startup companies and emerging technologies. Prior to joining Adobe, he has worked for HP Labs and Microsoft. He has a DPhil from Oxford University.

Table of Contents

Posters

Catching Elephants with Mice: Sparse Sampling for Monitoring Sensor Networks

Sorabh Gandhi
Department of Computer Science
UC Santa Barbara
sorabh@cs.ucsb.edu

Subhash Suri
Department of Computer Science
UC Santa Barbara
suri@cs.ucsb.edu

Emo Welzl
Department of Computer Science
ETH Zurich
emo@inf.ethz.ch

ABSTRACT

We propose a scalably efficient scheme for detecting large-scale physically-correlated events in sensor networks. Specifically, it is shown that in a network of n sensors arbitrarily distributed in the plane, a sample of $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ sensor nodes (*mice*) is sufficient to catch any, and *only those*, events that affect $\Omega(\varepsilon n)$ nodes (*elephants*), for any $0 < \varepsilon < 1$, as long as the *geometry* of the event has a bounded VC-dimension. In fact, the scheme is provably able to estimate the size of an event within the approximation error of $\pm\varepsilon n/4$, which can be improved further at the expense of more mice. We support the theory developed with extensive simulations.

1. INTRODUCTION

Sensor networks are enablers of what has been called sensory ubiquity or omnipresence: tiny, inexpensive, untethered sensor devices can measure and observe various environmental parameters, often in hazardous or humanly inaccessible places, thereby allowing real-time and fine-grained monitoring of physical spaces around us. Many potential applications of this technology relate to surveillance [1] or environmental monitoring [2], necessitating large-scale networks spanning wide-spread geographical areas. In many of these applications, the phenomena of interest are *global* in the sense that they are not discernible at the level of individual nodes, and require corroborative input from many sensors—that is, events only become significant if sensed data of many nodes support them. Indeed, this issue of making global inferences from local data is characteristic of many distributed systems, but it takes on a uniquely geometric form due to the physical embedding of sensor networks. As an example, imagine a sensor network in an *environmental monitoring* application, e.g., to detect wild fires in a forest or to track pollution levels in a habitat [5]. An abnormal or above average sensor reading at a single node, or even several but widely scattered nodes, is hardly a cause for concern, as local and temporal variations are routine in nature. On the other hand, abnormal measurements by a large number of nodes concentrated in a *geographical neighborhood* suggest a significant event that may require immediate action. One can imagine many other examples of this kind: sudden *energy* drop among sensor nodes in a neighborhood, abnormal *congestion* in a region, correlated changes in the sensed data at nodes in a neighborhood, and so on. Since a centralized data collection approach, where a central processor continuously collects signals from all the nodes and learns the state of the network, does not scale well with the size and the complexity of large scale sensor networks, we seek more efficient solutions. In a nutshell, our approach will be to monitor only a small subset (*sparse sample*) of the nodes in the network, and *infer* the presence or absence of a significant event just from the signals received from this subset.

2. PROBLEM DESCRIPTION AND NOTATION

Suppose a set S of n sensor nodes is distributed in the two-dimensional plane. We will model each sensor as a point, with (x_i, y_i) as its coordinates. We make no assumption about the distribution of these points (sensors), so our results will hold for all, even adversarial, sets. In addition, there is one distinguished node u that acts as the central processing node, or the *base station*, and knows the locations of all the sensors in the network. A simple and abstract way to model an *event* is as a binary function over the xy-plane. We imagine that the function has value 1 over some geographical neighborhood (denoting the extent of the event) and 0 elsewhere.

Let $E \subseteq S$ be the set of nodes with boolean value 1. We call E a *large event* or, metaphorically, an *elephant* if $|E| \geq \varepsilon|S|$, for a predefined user parameter ε, where $0 < \varepsilon < 1$. That is, if at least ε fraction of the nodes are affected by the event, then we call it an elephant. Our goal is to choose a subset $M \subseteq S$ of sensors as monitors, or *mice*, and design an algorithm $\mathcal{A}$ such that, *given only the boolean values for the nodes of* M, the algorithm $\mathcal{A}$ can always decide whether an event is an elephant or not. Such a solution will be useful and scalable if the size of the set M is significantly smaller than the network size $|S|$. A moment's reflection, however, makes it clear that unless the elephants are somehow "nicely shaped," one cannot hope to catch them all by a sparse, or *even linear* size, sample. Indeed, even if we choose *half* of all the sensors as mice, the adversary can create an event E that includes all the remaining nodes, and the algorithm has no way to detect this elephantine event.

We use *Vapnik-Chervonenkis (VC) dimension* from computational geometry and statistics [4, 6] to charecterize the complexity of shapes. A pair $(\mathcal{X}, \mathcal{R})$ is called a *range space* if $\mathcal{X}$ is a ground set and $\mathcal{R}$ is a family of subsets of $\mathcal{X}$. The elements of the set $\mathcal{R}$ are often called the *ranges*. In our setting, $\mathcal{X}$ is the set of sensor nodes S, and $\mathcal{R}$ is a collection of subsets that arise from the events that we wish to detect. Given a subset $A \subseteq \mathcal{X}$, we say that the set A is *shattered* by the set of ranges $\mathcal{R}$ if all $2^{|A|}$ subsets of A can be obtained by intersecting A with an appropriate member of $\mathcal{R}$. That is, for every subset $B \subseteq A$, there exists a range $R \in \mathcal{R}$, such that $B = A \cap R$. The *VC dimension* of the range space $(\mathcal{X}, \mathcal{R})$ is the cardinality of the *largest* set $A \subseteq \mathcal{X}$ that is shattered by $\mathcal{R}$. The main connection of VC-dimensions to our problem is through the concept of ε-nets, which are defined as follows. Given a range space $(\mathcal{X}, \mathcal{R})$, a subset $B \subseteq \mathcal{X}$ is called an *ε-net* for $\mathcal{X}$ if, for any range in $\mathbf{r} \in \mathcal{R}$, whenever

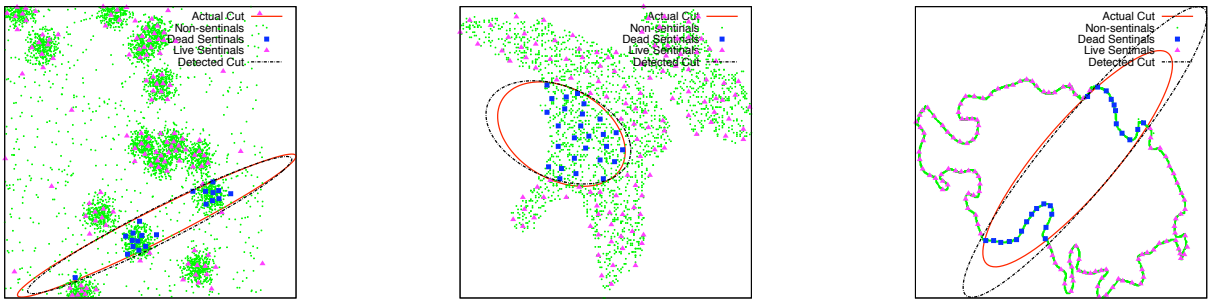


Figure 1: The figure shows detection of ellipse-shaped events on three different data sets: clustered, Lake, and Temperature. The maximum estimation error in these examples is again less than $0.01n$, while the target approximation guarantee is $0.1n$. The clustered data set has 2000 datapoints in clusters, the lake dataset consists of 2500 datapoints on the digitized boundary of Lake Huron, and the temperature dataset consists of 2000 randomly placed points on the boundary of temperature contour. In this picture, live sentinals refers to the mice for which the function value is 0 and dead sentinals refers to those with binary function value 1. The dotted line refers to the event detected and the solid line refers to the actual event.

$|\mathbf{r} \cap \mathcal{X}| \geq \varepsilon|\mathcal{X}|$, we also have that $\mathbf{r} \cap B \neq \emptyset$. In other words, a subset B is an ε-net if it has a non-empty intersection with any range that is an *elephant*. The ε-net theorem of Hausler-Welzl [4] shows that $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ independent random draws from $\mathcal{X}$, gives us an ε-net.

These ε-nets can be used to solve the problem described above but they suffer from *false alarms*. The scheme has no guarantee that the events caught are elephants. This is because the ε-nets offer only a *one-sided guarantee*: they only tell us that whenever a large event occurs, at least one node of the ε-net will also detect it. And this is the main contribution of this paper, we ensure a two sided guarantee, with not much increase in size. In the next section we describe the scheme used to detect elephants and charecterize the size of the sample set required.

3. CATCHING ELEPHANTS WITH GUARANTEES

We use the fact that the symmetric difference of pairs of ranges has bounded VC dimension (See [3] for proof). For ease of presentation, we will frame our discussion in terms of *circular* ranges, but it will be self-evident that the approach is completely general. The following algorithm is used to detect the elephants, where d is the maximum VC-dimension of the event we are trying to catch.

CatchElephants (S, d, ε)

1. Let $d' = O(d \log d)$ be the dimension of the symmetric difference of range spaces derived from dimension d ranges. Construct an $\frac{\varepsilon}{4}$-net for S of dimension d' (either by random sampling or deterministically).
2. Let M be the set of nodes chosen as the $\frac{\varepsilon}{4}$-net, namely, our monitors or mice. Let $T \subseteq M$ be the subset of monitors with boolean value 1—this is the intersection of the event with our monitoring set.
3. Compute a disk D containing the locations of all the nodes in T and excluding the locations of all the nodes in $M \setminus T$.
4. Compute the size $K = |S \cap D|$, the number of sensors in the network that lie inside the disk D. If $K \geq 3\varepsilon n/4$, then report the event as an elephant, with K as its predicted size. Otherwise, the event is not an elephant.

The following theorem establishes the correctness of this algorithm, and proves that the algorithm catches *all* events of size at least εn (the elephants) and *never* reports an event of size less than $\frac{1}{2}\varepsilon n$) (two-sided guarantee). (See [3] for proof.)

Theorem 1. *Let E be an event in the sensor network, and let K be its size estimated by the algorithm* CatchElephants. *Then,*

$$(K - \frac{\varepsilon n}{4}) \leq |E| \leq (K + \frac{\varepsilon n}{4}).$$

To give a taste of simulation results for this algorithm, Figure 1 shows events of a single shape family (ellipse) on three widely different data sets: clustered, lake, and temperature. We can see that our scheme performs quite well, and returns a plausible event boundary that is very close to the truth. For more detailed simulations, we point the reader to [3].

4. REFERENCES

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, et al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[2] A. Dhariwal, B. Zhang, B. Stauffer, C. Oberg, et al. NAMOS: Networked aquatic microbial observing system. In *ICRA*, 2006.

[3] S. Gandhi, S. Suri, and E. Welzl. Catching elephants with mice: sparse sampling for monitoring sensor networks. In *SenSys*, 2007.

[4] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

[5] K. Mayer, K. Ellis, and K. Taylor. Cattle health monitoring using wireless sensor networks. In *IASTED CCN*, 2004.

[6] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Data Stream Operators Under the Scanner: Challenges and Insights towards Efficient Parallelization *

Sudipto Das Shyam Antony Divyakant Agrawal Amr El Abbadi
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106, USA
{sudipto, shyam, agrawal, amr}@cs.ucsb.edu

ABSTRACT

Applications involving analysis of data streams have gained significant popularity and importance. In spite of the abundance of these algorithms, all known algorithms for answering data stream queries are sequential in nature. But the advent of multi-core architectures and their ubiquitous presence requires algorithms that can exploit the parallelism of these architectures. In this paper, we explore the challenges in parallelizing stream operators, in particular frequent elements and top-k queries, in the context of the parallelism offered by multi-core processors. We first analyze the different problems that arise when dealing with *intra-operator* parallelism and summarize the insights obtained from the different parallelization efforts. From the designs evaluated in this paper, it is evident that *intra-operator* parallelism is not intuitive and would require a redesign of the system.

1. INTRODUCTION

Data stream analysis forms an important class of applications where the data is streaming in, and processing has to be done in real time. An important distinction of data stream applications when compared to conventional database applications is that the stream algorithms can only make a single pass through the stream of tuples and since the stream can be potentially infinite, only a summary of the stream is stored instead of the entire stream. Analysis of the click streams in internet advertising is a good example of stream processing and its requirements. In Internet advertising, for determining the *Effective Cost Per Click* for an advertisement and in turn deciding which advertisements to display, a *publisher* needs to have an estimate of the number of *impressions* (the number of times an advertisement is rendered on the web page), and the *Click Through Rate* (i.e. the number of times the advertisement was clicked). This analysis requires real time frequency counting on the stream of clicks that is seen by the *publisher*.

Frequent elements [2, 5] and top-k [1, 2] queries are an important class of queries for stream analysis applications, and the research community has proposed different algorithms for answering these queries efficiently [1, 2, 4, 5]. A frequent elements query is interested in the elements whose frequency of occurrence is above a certain threshold. For example, a query of the form "advertisements that are clicked more than 0.1% of the total clicks" is a frequent elements query. On the other hand, a top-k query is interested in the k elements with the highest frequency at the instant when the query is being answered. Again, a query of the form "Top-25 most clicked advertisements" is a Top-k query.

Even though numerous algorithms have been proposed in the literature to answer these queries, all the proposed algorithms are serial in nature. But processor architectures have seen a recent shift in design where a single processor now consists of multiple cores which can execute instructions in parallel. The ubiquitous presence of these processors in almost all commodity as well as high-end computers necessitate the algorithms to be concurrent, so that multiple threads executing in parallel can efficiently exploit the available parallelism. In addition to *inter-operator* parallelism, where multiple operators execute independently and can execute in parallel on the different cores, *intra-operator* parallelism – where a single operator aims to utilize the available cores to improve the throughput of processing – is also important for long standing queries operating on huge amount of data. Data stream queries are typical examples of these long standing queries and are thus candidates for possible *intra-operator* parallelism.

In this paper, we explore the challenges in parallelizing stream operators, in particular frequent elements and top-k queries, in the context of the parallelism offered by the multi-core processors. We consider the *Space Saving* algorithm [5] which has a nice property that it can answer both the frequent elements and top-k queries. The rest of the paper explains the intuitive schemes for parallelizing this algorithm and provides insights into the different challenges in parallelization of frequent elements and top-k queries.

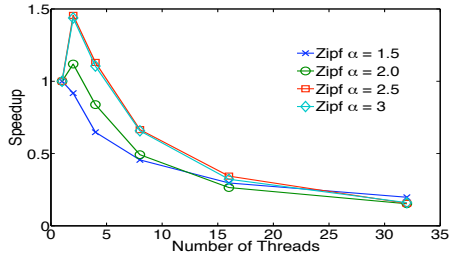
2. CHALLENGES IN PARALLELIZATION

The *Space Saving* algorithm [5] provides a fast and efficient technique for computing and answering the frequent elements and top-k queries over a stream. The algorithm provides an "integrated" solution for answering of frequent elements and top-k queries and requires $O(\frac{1}{\epsilon})$ space, where ϵ is the user specified error bound, and uses a structure called *Stream Summary* [2, 5] for maintaining the counters and keeping them sorted by the frequency of occurrence. In this section, we discuss the intuitive parallelization schemes of the *Space Saving* algorithm. These schemes are based on the way the threads share the *Stream Summary* structure, which would in turn determine how the threads execute the algorithm.

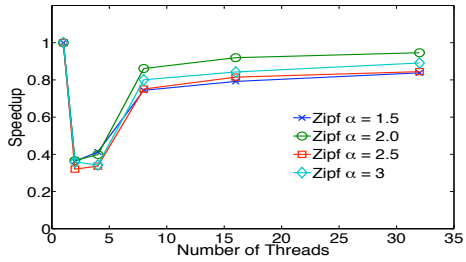
2.1 Independent Structures

This design corresponds to the *shared nothing* paradigm,

*This work is partly supported by NSF Grants IIS-0744539, IIS-0223022 and CNS-0423336



(a) Independent Structures



(b) Shared Structures

Figure 1: Intuitive parallelization schemes for the *Space Saving* algorithm. Speedup compared to a single thread of execution.

where the threads do not share any data or state information. The idea is to simulate sequential execution, and run multiple copies of the same algorithm executing on different partitions of data and eventually aggregating the results for answering the queries. It must be realized that there are two parts of the *Space Saving* algorithm, the frequency counting part, which counts the number of occurrences of an element, and the query part, where the frequency counts are used to answer the queries. Even though the frequency counting part can execute really in parallel, the local structures need to be merged to answer queries, and the frequency of merging the counters depends on the query frequency required by the application. As the number of parallel threads increases, the cost of frequency counting decreases, but the cost of merging increases. Figure 1(a) depicts this tradeoff. In this experiment, a stream of 5 million elements is processed, and a query is made every $50,000$ elements. The experiments are performed on an Intel QuadCore processor having 4 cores. The merge cost corresponds to a single merge – serial merge refers to a single thread merging all the structures, while hierarchical merge refers to a parallel merge similar to the merge phase of the *mergesort* algorithm. As is evident from Figure 1(a), this approach is not very scalable. This is because as the number of threads increases, the merge cost increases and so does the memory requirement, since the structure is replicated locally for all threads. The merge cost will increase further as the merge frequency increases.

2.2 Shared Structure

This design aims at solving the scalability issues of the algorithm described in Section 2.1. The intuitive solution is to use a shared *Stream Summary* structure. Since multiple threads are accessing the same structure, the threads must synchronize. Synchronization is achieved using locks and atomic operations supported by the underlying architecture. Figure 1(b) shows the same experiment (as in Section 2.1) for the shared structure. As is evident from Figure 1(b), using a shared structure does scale but the time taken is higher compared to those in Figure 1(a). This is because the threads spend a large fraction of the time contending for the shared resources, and since a large number of locks are being acquired and released, the overhead of locking and waiting for locks constitutes a significant portion of the time consumed.

It must be noted that in both the sub-figures in Figure 1, only the frequency counting time is reported, adding queries would only add to the overhead and further deteriorate the performance of the algorithms.

3. ANALYSIS AND INSIGHTS

It is evident from the experiments in Section 2 that the intuitive techniques for parallelization of the frequent elements and top-k queries do not scale with the number of available threads (or cores). This is of particular concern since the number of cores (or hardware threads) show a trend of doubling every processor generation. It further reiterates the significance of a scalable algorithm for the parallelization of stream operators so that they can exploit the available parallelism. An important observation from the intuitive parallelization schemes is that we need more than a "map-reduce" (analogous to the "independent" design in Section 2.1) or simple sharing (Section 2.2) for obtaining scalable performance. It must be noted that the threads are part of the same system. Therefore, instead of the threads contending with each other for shared resources, if the threads can be made to co-operate with each other, the overhead of contention is reduced, and the threads co-operate to perform useful operations instead of waiting for shared resources to become available. Therefore if the system is modeled based on the principles of *Delegation* (if multiple threads are contending for a shared resource, only one thread acquires the resource and all other threads delegate the request for that resource to the thread that has acquired the resource) and *Minimal Existence* (if a thread has acquired a resource, it will not contend with any other shared resource, and wherever required it will *delegate* requests to avoid contention), the contention overhead of the shared structure can be eliminated, while still retaining the advantages of a shared structure (i.e. small memory footprint and no merge cost). A detailed description and evaluation of the proposed technique are beyond the scope of the present paper and can be found in a separate technical report [3].

4. REFERENCES

[1] Das, G., Gunopulos, D., Koudas, N., and Sarkas, N. Ad-hoc top-k query answering for data streams. In *VLDB* (2007), pp. 183–194.

[2] Das, S., Agrawal, D., and Abbadi, A. E. CAM Conscious Integrated Answering of Frequent Elements and Top-k Queries over Data Streams. In *DaMoN* (Vancouver, Canada, 2008), pp. 1–10.

[3] Das, S., Antony, S., Agrawal, D., and Abbadi, A. E. CoTS: A Scalable Framework for Parallelizing Frequency Counting over Data Streams. Tech. rep., UCSB.

[4] Manku, G. S., and Motwani, R. Approximate frequency counts over data streams. In *VLDB* (2002), pp. 346–357.

[5] Metwally, A., Agrawal, D., and Abbadi, A. E. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Trans. Database Syst. 31*, 3 (2006), 1095–1133.

Symbolic Encoding of String Lengths

Fang Yu
University of California, Santa Barbara
yuf@cs.ucsb.edu

Tevfik Bultan
University of California, Santa Barbara
bultan@cs.ucsb.edu

Oscar H. Ibarra
University of California, Santa Barbara
ibarra@cs.ucsb.edu

ABSTRACT

We present a novel construction for length automata which accept the unary or binary representations of the length of a regular language. The construction can be used for verification of systems having unbounded strings and integers.

1. MOTIVATION

This project is motivated by an increasing interest in static analysis for real-world programs [1, 4]. We model these programs as infinite state systems having string and integer variables and present an automata-based approach for verification of these systems. Particularly, we are interested in the relationship among the values of integer variables and the lengths of strings. In general, the precise analysis of infinite state system is undecidable due to the fact that the values of integer variables are unbounded, the lengths of string variables are unbounded, and the reachability analysis of such infinite state systems is undecidable. To overcome this complexity hurdle, checking these infinite-state systems can be achieved by encoding infinite sets of states as regular languages and computing conservative approximations of the reachable states. A conservative approach is commonly adopted using deterministic finite automata (DFAs) to approximate the set of string values that string variables can take at certain program points, as well as to widen the arithmetic constraints that symbolically represent the set of values that integer variables can take. Previous works on static analysis of infinite state systems focus on either string variables [1,3,6] or integer variables [2,5]. We are interested in both. We adopt [6] to deal with string variables and its operations, and [2] for integer variables. Both infinite states of string and integer variables are approximated as regular languages and encoded as DFAs. In addition, we investigate the length constraint on the language accepted by a string automaton. Based on this constraint, one can catch the relation among the lengths of string variables and the values of integer variables. We believe this relation can be used to perform sophisticated verification on systems having unbounded string and integer variables.

In this abstract, we show that the length of the language accepted by a DFA forms a semiliner set. Given an arbitrary DFA, we are able to construct DFAs that accept either unary or binary representation of the length of its accepted words. The unary automaton can be used to identify the coefficients of the semilinear set, while the binary automaton can be used to compose with other arithmetic automata on integer variables.

The performance of our analysis relies on efficient automata manipulation. We implement all functions using a symbolic automata representation (MBDD representation from the MONA automata package) and leverage efficient manipulations on MBDDs, e.g., determinization and minimization. We believe that the symbolic representation (compared to explicit representation) of automata can be better scaled to model large systems and facilitates our tool to analyze real-world programs.

2. LENGTH AUTOMATA CONSTRUCTION

We are interested in identifying what length can be among the accepted words of a string automaton. Given a string automaton M, we aim to construct a DFA M_b (over a binary alphabet) such that M_b accepts the binary encodings (from the least significant bit) of the lengths of the words accepted by M. The following property is well known.

Property 1: For any DFA M, $\{n | n = |w|, w \in L(M)\}$ forms a semilinear set.

However, identifying the length set of arbitrary regular language is not trivial, e.g., the length set of $((baaab)^+ab)^+$ is $\{7, 12, 14, 17, 19, 21, 22, 24, 25, 26, 27, 28\} \cup \{29 + k | k \geq 0\}$. We tackle this problem by constructing the automaton M_u that accepts the unary encodings of the lengths of the accepted words of a string automaton M. Given $M = \langle Q, q_0, B^k, \delta, F\rangle$, where each symbol is encoded as a k-bit string, M_u is constructed by determinizing the NFA $\langle Q, q_0, B^1, \delta', F\rangle$, where $\delta'(q, 1) = q'$ if $\exists \alpha, \delta(q, \alpha) = q'$. M_u uniquely identifies a semilinear formula $\bigvee_i x = c_i \vee \bigvee_j \exists k.x = C + r_j + Rk$, where c_i, r_j, C, R are constants, and $\forall i, c_i < C$, and $\forall j, r_j < R$. The length set is exactly the set of values of x that satisfy the formula.

In the following, we propose an incremental algorithm to construct a DFA that accepts the binary encodings (from the least significant bit) of the values of x that satisfy a given semilinear formula.

The construction is achieved by calling the procedure CONSTRUCT_BLA. At line 3, we first compute the set of reachable binary states Q^b by calling the recursive procedure ABS (Add Binary State). A binary state is the value of a triple (t, v, b). $t \in \{0, 1, 2\}$ is the type of the binary state, which indicates the meaning of the value of v and b. While $t = 0$, v is equal to the value of the binary word accepted from the initial state to the current state, and b is equal to the binary value of the previous bit in the word. While $t \neq 0$, v is equal to the remainder of which the dividend is the value of the binary word accepted from the initial state to the current state and the divisor is R; b is the remainder of which the dividend is the binary value of the previous bit

in the accepted word and the divisor is R. $t = 1$ indicates the value of the binary word accepted from the initial state to the current state is greater or equal to C; $t = 2$ indicates the value is less than C. Each binary state is further associated with an index, a true branch and a false branch, which are used to construct the state graph later. Briefly, ABS is a recursive procedure which incrementally adds the encountered binary state if it has never been explored. Initially, the binary state is $(0, 0, \perp)$. We assume $2 \perp = 1$. Since binary states are finite, ABS is guaranteed to terminate. Upon termination, all reached binary states are added to Q^b. For each binary state in Q^b, as line 4 to 9, we iteratively generate a state q, and set its transition and final status. We construct the final automaton at line 10.

Algorithm 1 ABS(Q, C, R, t, v, b)

```
1: if ∃q = (t, v, b) ∈ Q then
2:    return q.index;
3: else
4:    Create q = (t, v, b);
5:    q.index = ♯Q; q.true = q.false = −1;
6:    Add q to Q;
7:    if t == 0 ∧ (v + 2 × b ≥C) then
8:       q.true =ABS( Q, C, R, 1, (v+2×b)%R, (2×b)%R);
9:       q.false =ABS( Q, C, R, 2, v%R, (2 × b)%R);
10:   else if t == 0 ∧ (v + 2 × b < C) then
11:      q.true =ABS( Q, C, R, 0, v + 2 × b, 2 × b);
12:      q.false =ABS( Q, C, R, 0, v, 2 × b);
13:   else if t == 1 then
14:      q.true =ABS( Q, C, R, 1, (v+2×b)%R, (2×b)%R);
15:      q.false =ABS( Q, C, R, 1, v%R, (2 × b)%R);
16:   else if t == 2 then
17:      q.true =ABS( Q, C, R, 1, (v+2×b)%R, (2×b)%R);
18:      q.false =ABS( Q, C, R, 2, v%R, (2 × b)%R);
19:   end if
20:   return q.index;
21: end if
```

Algorithm 2 CONSTRUCT_BLA(C, R, $\mathcal{C} = \{c_1, c_2, \ldots c_n\}$, $\mathcal{R} = \{r_1, r_2, \ldots r_m\}$)

```
1: Q^b = ∅;
2: Q = ∅;
3: init =ABS( Q^b, C, R, 0, 0,⊥);
4: for each q^b ∈ Q^b do
5:    Add q = q_{q.index} to Q;
6:    δ(q, 1) = (q^b.true ≠ −1?q_{q^b.true} : q_sink);
7:    δ(q, 0) = (q^b.false ≠ −1?q_{q^b.false} : q_sink);
8:    F(q) = ((q^b.t == 0 ∧ ∃c ∈ C.q^b.v == c) ∨(q^b.t ==
      1 ∧ ∃r ∈ R.q^b.v == (r+C)%R) :' +'?'−');
9: end for
10: Construct M = ⟨Q ∪ {q_sink}, q_init, B^1, δ, F⟩;
```

Implementation.

We have implemented the above algorithms using MONA DFA packages. We give an example in Figure 1. The length automaton accepts the binary representatin of the semilinear set $\{7 + 5k|k \geq 0\}$. Consider the number 1087, whose binary encoding is 10000111111. One can check that the bit string from the least significant is accepted by the automaton shown in Figure 1.

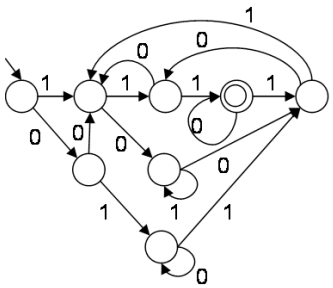


Figure 1: The Length Automata of $(baaab)^+ab$. The Semilinear Set is $\{7 + 5k|k \geq 0\}$

3. CONCLUSION

We have presented the algorithms to construct DFAs that accept unary or binary encodings of the length of a regular language.The DFAs that accept the binary encodings can be further composed to one multi-track arithmetic automaton [2] with other integer variables. This arithmetic automaton may catch the relation among the lengths of string variables and the values of integer variables. One can check the length properties of string variables or enforce the constraints on the lengths of string variables using the length set derived from the multi-track arithmetic automaton.

To complete our work, we will focus on: (1) the forward image computation of string and arithmetic automata against common string and arithmetic operations, (2) the widening operator to accelerate the fixed point computation, and (3) the general symbolic verification framework on systems having strings and integers. We plan to apply these techniques to check buffer overflows.

4. REFERENCES

[1] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, C. Kruegel, E. Kirda, and G. Vigna. Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications. In *Proceedings of the Symposium on Security and Privacy*, 2008.

[2] Constantinos Bartzis and Tevfik Bultan. Efficient symbolic representations for arithmetic constraints in verification. *Int. J. Found. Comput. Sci.*, 14(4):605–624, 2003.

[3] Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach. Precise analysis of string expressions. In *Proc. 10th International Static Analysis Symposium, SAS '03*, volume 2694 of *LNCS*, pages 1–18. Springer-Verlag, June 2003.

[4] Christian Kirkegaard, Anders Møller, and Michael I. Schwartzbach. Static analysis of xml transformations in java. *IEEE Transactions on Software Engineering*, 30(3), March 2004.

[5] Pierre Wolper and Bernard Boigelot. On the construction of automata from linear arithmetic constraints. In *TACAS*, pages 1–19, 2000.

[6] Fang Yu, Tevfik Bultan, Marco Cova, and Oscar H. Ibarra. Symbolic string verification: An automata-based approach. In *15th International SPIN Workshop on Model Checking of Software*, 2008.

Fast Annotation and Modeling with a Single-Point Laser Range Finder

Jason Wither
jwither@cs.ucsb.edu

Chris Coffin
ccoffin@cs.ucsb.edu

Jonathan Ventura
jventura@cs.ucsb.edu

Tobias Höllerer
holl@cs.ucsb.edu

ABSTRACT

This paper presents methodology for integrating a small, single-point laser range finder into a wearable augmented reality system. We introduce a method using the laser range finder to incrementally build 3D panoramas from a fixed observer's location. To build a 3D panorama semi-automatically, we track the system's orientation and use the sparse range data acquired as the user looks around in conjunction with real-time image processing to construct geometry around the user's position. Using full 3D panoramic geometry, it is possible for new virtual objects to be placed in the scene with proper lighting and occlusion by real world objects, which increases the expressivity of the AR experience.

1. INTRODUCTION

Mobile Augmented Reality (AR) allows users with wearable or portable computers to see and interact with virtual content that is located in and registered to the real world around them. In order to use the full interaction potential of this technology, AR research and application development increasingly focus on going beyond pre-created content. Implementing such expressive systems is difficult with traditional outdoor AR equipment however. For proper occlusion by real-world objects it is necessary to have a model of the real-world environment. In many outdoor systems a model is built as part of a preparatory offline process; however this is very time consuming and doesn't scale well.

Ideally, we would like a system that can provide correct occlusion of virtual annotations by the real world with very little effort. We would like this system to fit the framework of *Anywhere Augmentation*, requiring only negligible start up cost, no environment instrumentation, and only off-the-shelf hardware components.

In the spirit of this Anywhere Augmentation agenda, we decided to add a small, affordable, single-point laser range finder to our wearable system. With this new interactive sensing modality in place it is now possible to create 3D panoramas easily from a static location.

2. RELATED WORK

Several different approaches have been proposed for the problem of creating a panoramic environment map with depth information. The depth can be specified by the user in an interactive modeling system [6], however, the depth model produced by this type of system is only defined qualitatively. With multiple cameras [7], or a moving camera [5], panoramas with parallax can be automatically produced. More similar to our scenario is that of Bahmutov et al. [2]. They couple a 7x7 laser range finder array with a moving camera to produce highly detailed, textured indoor scene model.

Note the stark difference in focus between our static viewpoint technique and multi-view geometry approaches that recover or track sparse depth in moving user views [3], or recover depth information from landmark features in overlapping photographs [1].

3. HARDWARE AND CALIBRATION

The laser range finder we have chosen to use is an Opti-Logic RS400 which gives calibrated range readings continuously at 10 Hz and has a factory-specified range of 400 yards with accuracy of $\pm$ 1 yard. It weighs less than 8 ounces. We are also using a Pt. Grey Firefly MV camera, and a Garmin GPS 18 receiver. For orientation tracking we are using DiVerdi et al.'s [4] Envisor system. Envisor does completely vision based orientation tracking, using both sparse optical flow for frame to frame features and heavyweight landmark features. It builds an environment map on the fly, which we use for image processing.

We have calibrated the laser range finder to point parallel to the user's viewing direction. For an outdoor scene this means that the laser will always hit objects near the center of the users field of view. By measuring the baseline between the laser and camera we can further determine exactly what pixel in the image the laser is hitting depending on the distance returned by the range finder.

4. DEPTH MAP CONSTRUCTION

In this section we will describe using the laser range finder for the challenging task of building a full panoramic depth map around the user. As the user looks around the environment, our technique continually integrates color, depth, and temporal data to refine the estimated 3D model. In our experiments creating a number panoramic depth maps we found that the process of looking around to completely fill the full 360 degree depth map takes between two and four minutes, providing between one and three thousand depth samples from the laser range finder. This time could likely be reduced with a more robust tracking system.

To being creating a 3D panorama the user simply looks around. As the user pans the laser range finder, different objects are ranged. Because we only receive sparse depth samples compared to the resolution of the camera, we need to propagate point labels across the image. As a first step, we group the range points. Any time there is a large difference between one depth value and the next we conclude we have observed a group boundary. This technique for dividing the range points into groups is robust because of the high update rate of the range finder. Two consecutive updates will always have a small angle in between them, so our algorithm is unlikely to change groups when moving the range finder along a single surface.

One significant advantage to dividing the range points into groups is the semantic information gained about the spatial layout of the scene. We use this information to seed a diffusion based flood fill

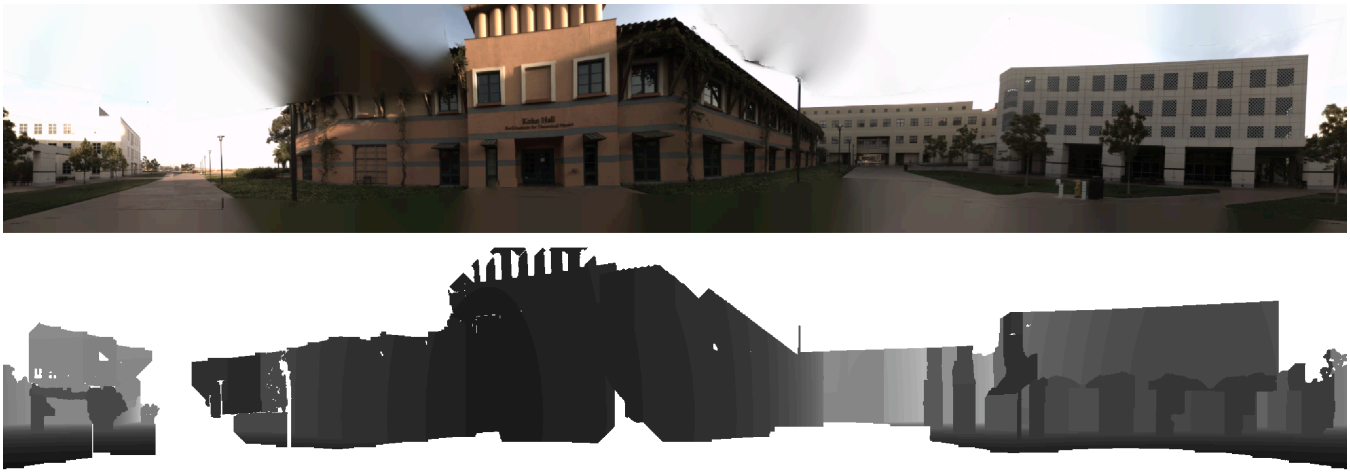


Figure 1: An example color panorama and automatically generated depth map pair. Darker regions of the depth map are closer to the user. To generate both images the user simply has to look around the scene. Both images are composed of the four surrounding faces of a cube map, and are not warped to a cylindrical projection. This cube projection causes the strange peak on the roof line in the center of the images.

process that expands the groups across the image. For this process a confidence value is associated with each pixel, which is highest at known sample points, and initially zero everywhere else. At each iteration, each pixel looks at its 8-connected neighborhood and averages its group (foreground or background) and confidence with neighboring pixels of higher confidence. This diffusion process is weighted by edge information as well.

We automatically detect object boundaries by examining the intensity gradient (using the 3x3 Sobel operator). The measure of boundary $E_p = f(\frac{dI}{dp})$ at pixel p is a function of the magnitude of the gradient. We use $f(x) = x^4$ so that the function will drop off quickly as the gradient decreases. To use the edge information to regulate diffusion the edge value at a neighboring pixel is subtracted from the neighbor's confidence value before being considered by the diffusion algorithm. The result is that pixels on an edge have a very low chance of diffusing their value, effectively stopping the diffusion along boundaries. One particular advantage of this method is its speed; on the GPU, 120 iterations per second can be achieved.

In areas of relatively dense laser range finder samples, the technique gives excellent segmentation along natural image boundaries. In areas with less dense sample resolution, the technique works less well, sometimes stopping short of filling the correct semantic region, or leaking through boundaries into incorrect semantic regions. However, the user can easily improve the results by adding more samples as appropriate.

From the group expansion we produce a group map which labels each pixel with its group number. Now, the depth of each pixel must be determined. To do this we try to model the scene with vertical planar surfaces. This is a reasonable thing to do for urban environments where most objects around a user are buildings. If we find that all of the points in a group are co-planar it is reasonable to assume that they are all on the same planar surface, and use that surface for smooth extrapolation across the whole group. By assuming that all planes are vertical we greatly reduce our search space, allowing us to create accurately oriented planes with a small number of points.

We use a perpendicular regression to find the best fit line through the set of 2D points where all heights are ignored. This regression minimizes the sum of squares of the perpendicular distances of each point from the line. We also use RANSAC to throw out any outliers in the set. Outliers are often created by small foreground objects like light poles that partially occlude the the surface of interest and can be ignored.

For a group with no detected planar objects, we take the average depth of the samples in the group as the depth of the entire group. Finally, we use the average height of a user to determine the ground plane, and add that plane to the our complete depth map.

5. CONCLUSIONS AND FUTURE WORK

We have presented methodology for use of a single-point laser range finder in an outdoor AR setting. We have presented results that demonstrate how a laser range finder can improve the AR experience by building a depth map around a static user. The depth maps we create are accurate enough to be useful for a number of AR applications, arguably the most useful of which is automatic occlusion of virtual objects by real world objects.

6. REFERENCES

[1] T. Aoki, T. Tanikawa, and M. Hirose. Virtual 3d world construction by inter-connecting photograph-based 3d models. *IEEE VR*, pages 243–244, 2008.

[2] G. Bahmutov, V. Popescu, and E. Sacks. Depth enhanced panoramas. *IEEE Visualization*, 2004.

[3] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE ICCV*, 2003.

[4] S. DiVerdi, J. Wither, and T. Höllerer. Envisor: Online environment map construction for mixed reality. In *IEEE VR*, 2008.

[5] S. Peleg and M. Ben-Ezra. Stereo panorama with a single camera. *IEEE CVPR*, 1999.

[6] H.-Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. *IEEE CVPR*, pages 427–433, 1998.

[7] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *ACM SIGGRAPH*, pages 299–306, 1999.

Mergeable-Cache: Exploiting Data-Similar Executions in Multicore Processors

Susmit Biswas, Frederic T. Chong, Diana Franklin, Timothy Sherwood
{susmit,chong,franklin,sherwood}@cs.ucsb.edu

ABSTRACT

While microprocessor designers turn to multicore architectures to sustain performance expectations, the dramatic increase in parallelism of such architectures will put substantial demands on off-chip bandwidth and make the memory wall more significant than ever. We propose that one potential application that will be profitable for multicore processors is in the support of many similar executions of the same program. We propose cache structures that detect data similarities and merge cache blocks, resulting in substantial savings in cache capacity and reductions in cache misses that go off chip. Specifically, we propose a *Mergeable* cache that merges duplicate cache lines owned by different processors. Merging duplicate data owned by different processes poses several challenging problems which lead to degradation in performance for a pure *Mergeable*-cache in program phases with low data similarity across executions. We address these issues in this paper, and propose a *Half-Mergeable* cache architecture which combines the benefits of both traditional cache and *Mergeable-cache*. Using trace-based simulation, we demonstrate that our proposed technique provides a scalable solution and leads to an order of magnitude reduction in main memory accesses. For 16 cores running 16 similar executions of the same application and sharing an exclusive 4-MB, 8-way L2 cache, the Half-Mergeable cache shows a L2-miss rate reduction of up to 19.56× and 1.44× on average, while posing an overhead on cache size of only 2.824% and negligible power overhead.

1. INTRODUCTION

As the difficulties of uniprocessor performance scaling have proven economically daunting, designers have turned to bandwidth (parallelism), rather than latency, to scale performance in future microprocessors. This approach has led to two significant challenges. First, as a single reference stream scales to tens, hundreds, or even thousands of reference streams, the memory system will struggle to service all of the requests in a timely manner. Second, careful programming will be required to parallelize applications to hundreds of cores. Programming both correct and efficient parallel code is challenging, and too few programmers have the expertise to accomplish both.

We explore an interesting class of applications that fall into a model which we term "multi-execution." Multi-execution refers to the execution of multiple copies of the same program, but with different input data or parameters. We believe that multi-execution could become a useful model of execution as multicores scale, because it is already used in many domains, and because no software changes are necessary to take advantage of a multicore system. Note that an alternative to multi-execution is to write an explicitly parallel program that takes many instances of an application and explicitly shares redundant data. This approach, however, is labor intensive, difficult to get correct, often requires source access to libraries and copyrighted/proprietary codes, and can miss substantial data similarity that can only be discovered with an efficient dynamic mechanism.

We explore the characteristics of several example applications from simulation, optimization, database, and learning domains. We find that the similarity of multi-execution working sets can be quite high, but careful design is needed to profitably exploit this similarity in a memory system. We present a *Mergeable cache* architecture which dynamically merges cache lines containing identical data from different instances of the same program running under multi-execution. We find that a Mergeable cache poses several challenging problems which lead to degradation in performance for execution phases with low data similarity. We address these problems, and propose a *Half-Mergeable* cache and protocol which combines the benefits of a Mergeable and a conventional cache.

We have implemented a trace-based simulation framework to demonstrate the strength of our approach. In this paper, we present results with 10 applications including 6 benchmarks from from SPEC2000 benchmark suite[1], and *LIBSVM, icsiboost, PyODE* and *NGSpice*. In this paper, we interchangably use processes and processors as, in our simulation based study, we assume that a single process executes in one processor.

2. MERGEABLE CACHE ARCHITECTURE

Although the data similarity across instances of our multi-execution applications is high, designing a cache architecture that can take advantage of this similarity without incurring excessive overhead or degrading conventional performance proves to be challenging. Specifically, our goal is to design an efficient hardware technique that dynamically finds identical data among different processes, merges such data, and splits them again when the processors write to the data. There are three significant technical issues which need to be addressed in merging identical data blocks from multiple processors.

1. Finding identical data is expensive if every access to the memory results in comparison with all valid cache lines. Searches must be minimized while the opportunity for identifying mergeable data is maximized.
2. Merged data needs to be organized in such a way that data is quickly found in a standard cache access.
3. Since all processors are running the same program, typical memory mapping would cause many (unmerged) accesses to align and cause many cache conflicts.

To solve the first problem, we observe that applications are most likely to have identical data located at the same vir-

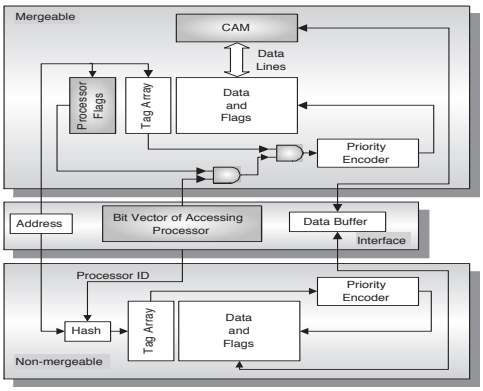


Figure 1: *Half-Mergeable* cache architecture. The cache is partitioned in Mergeable and non-Mergeable (traditional) cache segments. When a line is evicted from L1 and moved to L2, all lines of the cache set which the evicted line gets mapped to, are copied and compared in CAM.

tual address (but different process ID). Thus, we limit our searches to only data having the same virtual address. To perform this search efficiently, we must arrange to map all the relevant virtual addresses the same cache set.

Unfortunately, L2 caches, where line merging occurs, are typically physically-addressed. Searching all possible sets that could contain the same virtual address would be prohibitively expensive. In order to limit the searches, we use a *page coloring technique*[2] to assign virtual pages to physical pages such that the same virtual address across all processes map to the same set in the cache.

Because cache lines for the same virtual address map to same cache set, the search space for identical cache lines is bounded by associativity. Restricting the virtual page placementreduces the number of possible locations to search for identical cache line, but it has the potential to severely impact performance. Because all of the processes sharing the cache are virtually identical, they tend to use the same addresses at the same time, and if the data in that set are not identical, then the cache will exhibit very poor behavior due to conflict misses.

This is solved by skewing the address by the processor number so that same virtual address from all processors are not mapped to same cache set, and thereby, conflicts can be reduced[3]. This skewed addressing technique, however, is entirely at odds with our desire to keep merging candidates in the same cache set as the goal of reducing conflict misses and grouping merging candidates are conflicting.

We resolve this conflict by using a *Half-Mergeable* cache with static partitions of Mergeable and non-Mergeable segments. Data lines which can be merged reside in Mergeable segment, and lines with dissimilar content are moved to a non-Mergeable cache. When a cache line is evicted from an upper-level cache, it is written to the Mergeable segment at first. If a cache line is not able to merge with any pre-existing cache line having the same address and data, it replaces an old line, which is moved to the non-Mergeable segment. A Half-Mergeable cache provides the advantage of utilizing

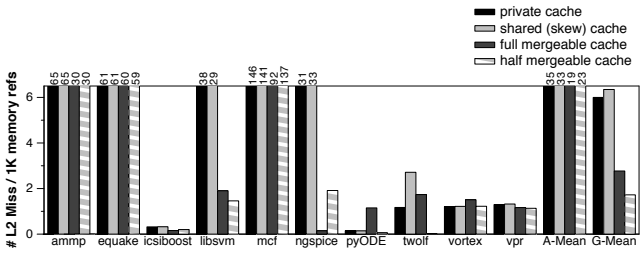


Figure 2: L2 cache miss count per 1000 memory references for 10 benchmarks.

the full cache when data similarity is low without increasing the cache miss rate significantly, yet storage efficiency is increased by merging identical lines in the Mergeable cache.

3. RESULTS AND CONCLUSION

We use the *total number of L2 miss × 1000 / total number of L1 accesses* as our metric in all results. Due to space constraints, we show only simulation results of 16 processors sharing a 4-MB, 8-way exclusive set-associative L2 cache with 32-KB private L1 cache in each processor in Figure 2. The L1 and L2 caches are exclusive. Because of increased conflicts in non-shared regions, the Full-Mergeable cache performs worse on *PyODE* and *255.vortex*, but the Half-Mergeable cache reduces the miss rate by 1.44× on average. In case of *255.vortex* and *PyODE*, the Full-Mergeable cache increases miss rate by 1.23× and 8.02× respectively, but *Half-Mergeable* cache shows only 0.34% increase in miss rate for *255.vortex*, and reduction in miss rate by 55.12% or 2.27× in *PyODE*. The geometric mean shows that, when applications are weighted more evenly, the Half-Mergeable cache outperforms the Full-Mergeable cache. The Full-Mergeable and Half-Mergeable cache provide a reduction in miss rate by a factor of 2.29x and 3.68x, respectively, over a shared cache using skewed addressing.

Our initial architecture which employs a static partition of cache into Mergeable and non-Mergeable segment shows the potential of merging similar data. Our next step is to design a dynamic cache-partitioning technique which will be able to manage mergeable data more efficiently, adjusting for different amounts of sharing. Furthermore, the multi-exection paradigm could be expanded to include simultaneously executing programs that are similar, but not identical. Different programs using many of the same libraries, for example, may be good candidates for multi-execution support. Since we have taken great care to support both low-similarity and high-similarity cases in our memory system design, it is our hope that a low-overhead multi-execution system can be used to support a wider scope of applications than explored in this initial study.

4. REFERENCES

[1] SPEC CPU2000: http://www.spec.org/cpu/.
[2] S. Bederman. Cache Management System Using Virtual and Real Tags in The Cache Directory. *IBM Technical Disclosure*, 21(11), April 1979.
[3] N. Topham and A. Gonzalez. Randomized Cache Placement for Eliminating Conflicts. *IEEE Transactions on Computers*, 48(2):185–192, February 1999.

Whiteboard Computing: Towards A Sketch-Centric Operating Environment

Ryan Dixon and Timothy Sherwood
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106-9560
{rsd,sherwood}@cs.ucsb.edu

ABSTRACT

The simplicity and accessibility of whiteboards provides an appealing avenue for sharing ideas and solving problems. Yet even as these surfaces sit at the center of computer-intensive work environments, whiteboards are largely relegated to serving as note-taking devices. We present our concept of whiteboard-based computing, highlight current hardware and software trends that have laid the groundwork for whiteboard-based systems, and describe our efforts in creating a whiteboard computing framework.

1. MOTIVATION

Whiteboards offer a simple and powerful means of describing and communicating complex ideas. Yet, while their utility as an integral problem solving tool is recognized across many disciplines, the whiteboard largely remains a static device with only limited integration into our current computer-based workflows. Our goal is to create the next generation whiteboard system whereby free-form input is used to not only draw and share static images, but also perform dynamic computations. It is our belief that current hardware and software trends support our vision.

With recent advances in multi-touch technology, a new market for devices larger than traditional computer displays appears to be emerging [3, 6]. As electronic displays continue to double in size approximately every year and a half, we are nearing a convergence in the landscape of desktop computing and wall-sized human-computer interaction [2]. While existing windows, mouse, and pointer interfaces are being adapted to better suit these changing demands, there is significant potential for new methods of interaction. Given the current trends, it is evident that both hardware and software will play a critical role in the evolution of this technology.

Over the past decade, numerous handcrafted, domain-specific software systems have been designed to handle free-form input [1, 4]. The first generation of prototypes is just now reaching a product-ready level of maturity; however, there is little to no coherence between any of the individual projects.

Ultimately, we desire to harness and combine these trends into a system that can compute upon free-form input from any number of domains without sacrificing the freedom provided by a conventional whiteboard. We believe this presents a number of new challenges to architect, system, and language designers alike, and we have started to take the necessary steps towards making this a reality.

2. WHITEBOARD TRAFFIC

To better understand the data requirements of whiteboard systems, using an eBeam capture device [5], we have collected roughly four months worth of input on our laboratory whiteboard. Our experiment does no recognition, instead it passively records statistics from the data points that are naturally written by our lab for computer architecture research so that we can understand the distribution of inputs that might be given to recognition systems. Beyond spatial coordinates, the raw data captured includes timing information, marker color, and marker size.

Over the survey period, approximately 10-15 lab members had access to the whiteboard. The sample period, spanning from November 19, 2007 to March 17, 2008, includes portions of Winter and Spring quarter, including a significant holiday break in between. Of the 120 days of the survey, 49 days (approximately 41%) contained board activity. The combined days of board activity produced over 300,000 input points and 11,420 input strokes. An input stroke is generated every time an input pen touches the whiteboard surface. A stroke accumulates all point data captured until the pen is lifted from the surface of the whiteboard. Our entire data collection consists of strokes composed of up to hundreds of points each. All of the observed traffic was consistently bursty, often occurring within the span of an hour, followed by hours of inactivity.

Throughout the duration of the survey, we also designed tools to help parse, interpret, and understand the whiteboard data. From this work, we have created the beginning of a whiteboard computing environment, where multiple concurrent applications are allowed to run and interact with free-form sketches and perform live computations.

3. WHITEBOARD FRAMEWORK

Our current work is focused on developing a whiteboard environment where users can freely sketch and view computations on-the-fly. At a high level, the board operating environment is a lightweight event-based system that is responsible for managing board real-estate, and access to annotated stroke data. Figure 1 illustrates the current board architecture and four of the distinct phases outlined below. In its initial state, a board hosts a set of board objects that will be responsible for parsing and interpreting all incoming stroke data. These objects largely define the functionality and behavior of the board.

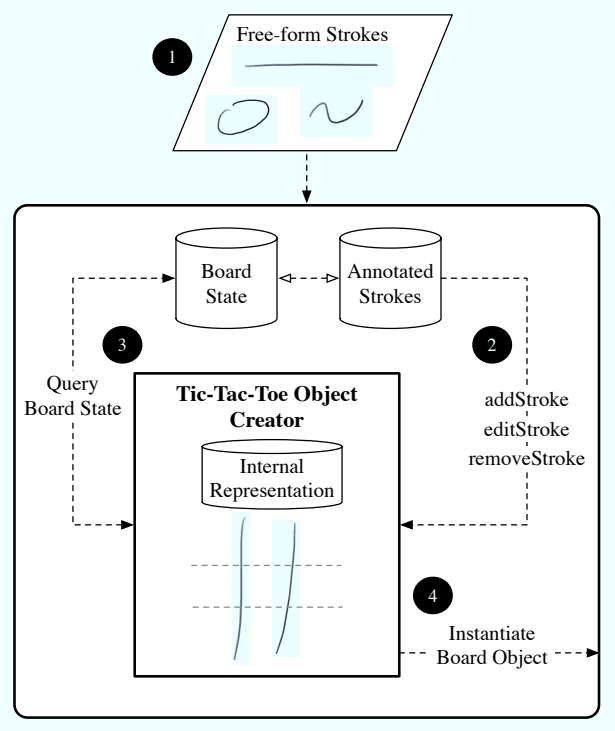


Figure 1: The board object event loop: (1) Stroke input, (2) Event processing, (3) Querying board state, and (4) Board object instantiation.

As a user draws, the board is responsible for collecting and storing representative stroke data. An initial pass is made over the data to annotate specific features that will likely be relevant to a number of board applications. These annotations might include features such as: fragmentation, where complex strokes are separated into simpler component pieces; beautification, where overlapping strokes are refined into a single representative stroke; and even character recognition. The initial board annotation phase enables the board to support gestures. Currently, only the scribble gesture is supported to enable full-stroke erasure.

After the initial annotation phase is complete, each of the board objects is presented an event, indicating that a new stroke has been added to the board. It is then up to each individual board object to determine what significance that new stroke has with respect to its own state of computation. A board object can also query the board for an up-to-date view of board strokes. Lastly, a board object has the potential to create and add new board objects to the board. In the case of Tic-Tac-Toe, a board object constantly monitors the board for the appearance of a Tic-Tac-Toe game. Once the necessary hash marks have been drawn, the board object instantiates a new Tic-Tac-Toe game.

Our design approach can, in many ways, be viewed as a generalization of the spreadsheet; portions of the board may interact and affect other regions of computation. As soon as a stroke is drawn, changes are immediately presented to the user. Figure 2 provides a sample of the applications we

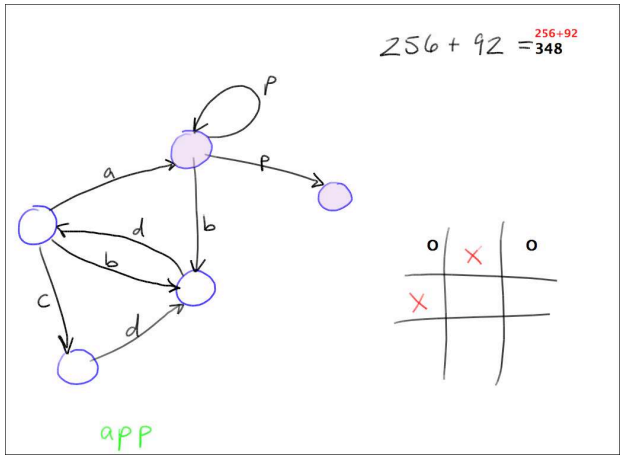


Figure 2: An example of three concurrent whiteboard applications: an equation solver, Tic-Tac-Toe, and a finite state machine.

have built thus far. These three applications exemplify the intended interaction with the whiteboard. For example, as input is written for the the finite state machine (the bottom text in Figure 2), the active states are automatically updated and highlighted. Likewise, when edges and nodes are added or removed from the state machine, changes are reflected immediately in the new graph.

Our work is still very much in the preliminary stages. Although we have not yet performed any definitive tests on the system, our initial feedback has been positive and we believe are initial efforts are a step in the right direction.

4. CONCLUSIONS

This work presents challenges across many domains. User interface decisions affect the stroke recognition process which, in turn, affects recognition accuracy and system performance. Many questions pertaining to system design remain open. It is our hope to not only provide a viable solution to the whiteboard computing problem, but also provide a starting point for future development in this area. We hope to inspire a combined effort between architecture, system, user interface, and language design experts to help further develop this idea.

5. REFERENCES

[1] C. Alvarado and R. Davis. Dynamically Constructed Bayes Nets for Multi-Domain Sketch Understanding. In *IJCAI '05*, August 1 2005.
[2] R. Dixon and T. Sherwood. Whiteboards that Compute: A Workload Analysis. In *IISWC '08*, 2008.
[3] J. Y. Han. Low-Cost Multi-Touch Sensing Through Frustrated Total Internal Reflection. In *UIST '05*. ACM, 2005.
[4] J. J. Laviola and R. C. Zeleznik. MathPad2: A System for the Creation and Exploration of Mathematical Sketches. *ACM Trans. Graph.*, August 2004.
[5] Lucidia. eBeam - Interactive Whiteboard Technology, 2008.
[6] Microsoft. Microsoft Surface - Surface Computing, 2008.

MyDepressedSpace: Classification and Search on MySpace Pages

Kathy Macropol, Camilla Fiorese and Madhu Venugopal
Dept. of Computer Science
University of California, Santa Barbara
{macropol, camilla, madhu.venugopal}@cs.ucsb.edu

ABSTRACT

Social networking websites, such as MySpace [6], have a wealth of data available on their users. This data represents both knowledge discovery opportunities for researchers, as well as important privacy issues for users. However, despite the many possibilities for the data mining of these pages, most research done has focused on only a few specific areas. Our aim in this paper is to look at a previously unconsidered area, namely medical or psychological data mining. We consider the classification of MySpace pages, using a naive Bayes classifier, based on the likelihood of the user to be depressed. We find that depression (bipolar and unipolar) can be predicted from the information on users' pages. By combining the results with a web search engine, we create a tool that allows a user to do a keyword search and obtain a list of related profiles containing individuals at high risk of being depressed. The analysis of information from depressed users reveals several major trends — some new, others confirmed in previous studies of depression. The results of our study indicate that the data mining of social networking websites can be useful for medical data mining, as well as an important privacy issue for their users.

1. INTRODUCTION

In recent years there has been considerable interest in social networking websites. With the largest of these sites consisting of millions of members, many research studies have been done focusing on the structure, privacy issues, and social implications of such networks [3]. In addition, each page may have a wealth of information on its user: from categorical information such as age and education, to blog pages containing details on a user's life. Studies have been done focusing on the mining of such information with the purpose of targeted advertising [10]. However, to the best of our knowledge, no study has been done that considers the feasibility of collecting and data mining these profiles with the purpose of medical or psychological analysis.

To discover if the data mining of these pages was feasible for such classification, we chose to focus on the problem of classifying users from the social networking website, MySpace, for visible signs of depression. (No distinction between bipolar and unipolar depression was made.) Our study focused on depression for two main reasons. First, as a serious medical condition that affects the lives of millions of individuals in the US, finding people suffering from depression is both a major health and potential privacy concern. Secondly, many signs of depression may be discoverable from the information contained within a user's blog (i.e. suicidal thoughts or attempts may be recorded in their entries).

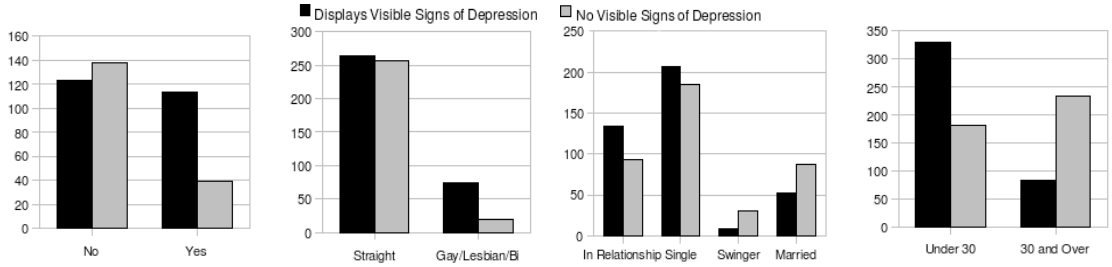
2. IMPLEMENTATION

To accomplish this, we first collected pages from 17,000 MySpace users (the profile and up to 50 blog pages per user) as a total dataset. To create a training set, 193 MySpace users (94 positive, 99 negative) were manually selected. Each user in the training set was classified as positive (depressed) or negative (not depressed). A positive record was determined by three criteria. 1.If the user stated in their blog that they had been diagnosed with depression. 2.If the user either had committed suicide, or stated within their pages that they had tried to commit suicide. 3.If the user confirmed in their blog that they self-harmed, cutting themselves. Negative profiles were those that did not meet these three criteria.

The dataset consisted of 791 attributes per user. Each attribute was a score (proportional to the TFIDF) calculated from the frequency of a certain phrase appearing in that user's blog. The 791 phrases were chosen by first finding the most frequent phrases in the blogs of the positive training set, and manually picking those that seemed most relevant (i.e. "I want to die", or "my life sucks"). The blogs were parsed and the scores calculated using a perl script written using the HTML::TreeBuilder and HTML::TokeParser modules available from CPAN [7].

A naive Bayes classifier was then created and trained using Weka's Java interface [9]. The classifier created had 81.3% accuracy and 97.9% precision when using 4-fold cross validation on the training set. After training, the classifier was run on the original 17,000 user pages to obtain scores (probabilities) for each user based on the likelihood of depression.

To allow easy access to the data and results, as well as create a potentially useful tool, the results were combined with a web search engine. Our web-based system utilized two open-source products from the Apache Software Founda-



(a) User Smokes (b) Orientation (c) Status (d) Age

Figure 1: Number of Depressed / Not Depressed Users per Selected Category

tion: Lucene [5] and Tomcat [8]. The MySpace pages were indexed in Lucene according to their HTML content and depression scores. This created a searchable index of MySpace users that could then be integrated with Tomcat and a Java Servlet. In the end, our system allowed users to perform keyword searches and obtain a list of related profiles containing individuals likely to be depressed. From this, we were able to analyze our results, as well as create a tool which may be useful for psychologists and social studies researchers.

3. RESULTS

Classification of the 17,000 MySpace users returned 417 with high ($\geq$0.9) scores for depression. To ensure high scores indicated higher chances of depression, the top 100 scoring users were manually categorized. Thirteen users were found to be false positives: two depression support groups, and the rest having blog posts with depressed poems or song lyrics. This meant 87% of the highest scoring users showed strong signs of depression, helping to confirm that high depression scores indicate higher probabilities of depression.

To analyze the returned results, the top 416 scoring profiles were collected and assumed to contain individuals suffering from depression. The lowest 416 scoring profiles (containing comparable blog lengths to the positive users) were collected as the negative dataset. These pages were parsed and statistics collected from the categorical information contained within them. From these results, several major trends were discovered. The top four categories containing the most substantial statistical differences are listed in Figures 1a–1d.

Figures 1a and 1b show that smokers and gay/lesbian/bisexual users are significantly more likely to be depressed, results confirmed in previous studies [1, 4]. Females had a 12% higher chance of depression than males, a finding also confirmed in [2]. In addition, as shown in Figure 1c, married users are less likely to be depressed, a result also found in [2]. However, an exception to this are individuals classified as Swingers. Interestingly, in a new finding, Swingers were the least likely to be depressed, being 15% less likely to be depressed than married users. Finally, from Figure 1d, users under 30 years of age had much higher chances of displaying signs of depression (a 39% increase). Previous studies found higher rates among this age range as well, though not with as large a gap [2]. Two issues unique to age may affect its results. One is that younger users may be more comfortable providing details of their life in their blogs. Another are the age limits given on websites such as MySpace, forcing some users to lie about their ages. These problems, connected to datasets taken from online websites, may make analysis and control of age difficult in the data mining of such data.

However, overall, our goals of analyzing the power of social network website data mining, as well as creating a tool for searching MySpace users ordered by likelihood of depression, were both met. The fact that analysis of our results produced major trends, both new and previously confirmed, shows that data collected from these websites are a valid source for medical data mining and the discovery of new knowledge. It also raises the need for better understanding of privacy issues online, as any random individual, company, or government may be able to use data mining to discover more about an individual than they may wish possible.

4. REFERENCES

[1] R. Brown, P. Lewinsohn, J.R. Seeley, and E.F. Wagner. Cigarette Smoking, Major Depression, and Other Psychiatric Disorders Among Adolescents. 35(12):1602–1610.

[2] R. Frerichs, C. Aneshensel, and V. Clark. Prevalence of Depression in Los Angeles County. 113(6):691–699.

[3] J.M. Kleinberg. Challenges in mining social network data: processes, privacy, and paradoxes. In *Proc of the 13th ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining*, San Jose, California, USA, August 2007.

[4] J. Lock and H. Steiner. Gay, Lesbian, and Bisexual Youth Risks for Emotional, Physical, and Social Problems: Results From a Community-Based Survey. 38(3):297–304.

[5] Lucene. `http://lucene.apache.org`.

[6] MySpace. `http://www.myspace.com`.

[7] Comprehensive Perl Archive Network. `http://www.cpan.org`.

[8] Tomcat. `http://tomcat.apache.org`.

[9] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

[10] W. Yang, J. Dia, H. Cheng, and H. Line. Mining Social Networks for Targeted Advertising. In *Proc for the 39th Hawaii Intl Conf on System Sciences*, Hawaii, USA, January 2006.

Are Your Votes *Really* Counted? Testing the Security of Real-world Electronic Voting Systems

Davide Balzarotti, Greg Banks, Marco Cova, Viktoria Felmetsger,
Richard Kemmerer, William Robertson, Fredrik Valeur, and Giovanni Vigna
Computer Security Group
{balzarot,nomed,marco,rusvika,kemm,wkr,fredrik,vigna}@cs.ucsb.edu

1. INTRODUCTION

Electronic voting systems play a critical role in today's democratic societies, as they are responsible for recording and counting the citizens' votes. Unfortunately, in an alarming number of cases, these systems have malfunctioned, suggesting that their quality is not up to the task. Recently, there has been a focus on the *security testing* of voting systems to determine if their defects could allow an attacker to control the results of an election.

We participated in two large-scale security testing projects: the California Top-To-Bottom Review (TTBR) of voting machines in July 2007 [3] and the Ohio's Evaluation & Validation of Election-Related Equipment, Standards & Testing (EVEREST) in December 2007 [2]. In the former, we evaluated the Sequoia voting system, while, in the latter, we evaluated the ES&S voting system. Our task was to identify, implement, and execute attacks that could compromise the confidentiality, integrity, and availability of the voting process. During our testing, we identified major flaws in both systems. As a result, the voting systems used in California were decertified and those used in Ohio were recommended for decertification.

In this paper, we briefly provide an overview of electronic voting systems and describe our experience with the security testing of two of such systems. We refer the interested reader to the full-length version of this work for more details [1].

2. ELECTRONIC VOTING SYSTEMS

Electronic voting systems are large-scale, complex, distributed systems, whose components range from general-purpose PCs to optical scanners and touch-screen devices, each running some combination of commercial off-the-shelf components, proprietary firmware, or full-fledged operating systems. The components that most frequently are part of an electronic voting system are:

- DRE – Direct Recording Electronic voting machine. A device to record the voter's choices. This is usually a touch-screen device where the voter casts his/her vote.
- VVPAT – Voter-Verified Paper Audit Trail. A paper-based record of the choices selected by the voter. The VVPAT printer is hooked to the DRE and the paper record is viewable by the voter, but it is under a transparent cover so that it cannot be modified other than through the normal voting process.
- EMS – Election Management System. The system responsible for the initialization of the components that collect the votes and also for the final tallying of the votes. The EMS is usually located at election central.
- Optical Scanner. An optical reader that counts votes cast on paper ballots. There is usually one scanner at each polling site and one at election central (e.g., for the counting of absentee ballots).
- DTD – Data Transport Device. Storage devices to transfer data between different components of the systems. These devices are used to transport ballot information to the DREs and optical scanners at the polling site and to transport voting results to the EMS.

Figure 1 represents how these components interact during a typical election process. In the following, we describe the main steps of this process.

Prior to the election, ballot information is prepared on the election management system at election central. This information may be directly entered into the DREs and the optical scanners, or it may be written onto DTDs that are sent to the polling places. Paper ballots for each of the polling places are also prepared at election central.

On election day, prior to the start of the voting process, if needed, optical scanners and DREs are initialized with the appropriate ballot information at the polling site, using the DTDs. Then, DREs are tested with sample votes to see if they record everything accurately. The optical scanners are tested in a similar way. If the DREs and the scanners pass the pre-election testing, then they are ready to be used for voting.

When a voter comes to the polling place, he/she registers at a desk. Then, the voter is given a token (e.g., a smart card) to insert into the DRE to start voting. The voter's choices are displayed on the DRE screen and are also printed on the VVPAT.

If paper ballots are used, the voter is given a ballot and a marking device to cast his/her vote. When the voter is through, the ballot is handed to an official who inserts it into the optical scanner to be read and recorded. Some optical scanners will report an undervote (voting for less than n choices when n are supposed to be picked) or an overvote (voting for more than n choices when n is the maximum number that can be marked). If this is the case, the voter is given the opportunity to correct his/her vote.

After the election is closed, the results from each of the DREs and scanners at a polling place are collected on a DTD and returned to election central, where they are read into the election management system to produce a tally for the entire area.

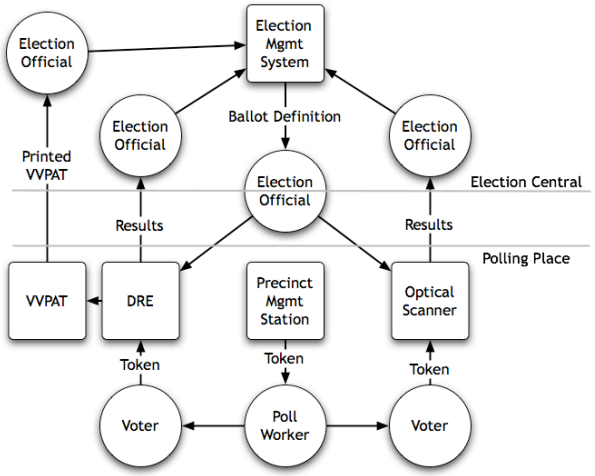


Figure 1: High-level overview of an election system's components and of the election process.

3. EXPERIENCE IN TESTING REAL-WORLD VOTING SYSTEMS

We were asked to evaluate the security of two electronic voting systems: one produced by Sequoia, the other by ES&S. As part of our evaluation, we developed a security testing methodology and a number of tools. For reasons of space, we will only briefly mention these. Instead, we will discuss more in detail the findings of our testings.

We have developed a five-step testing methodology that can help security engineers in designing experiments to evaluate the security of an electronic voting system. This is a high-level approach that focuses on finding bugs and design errors that can potentially be exploited by an attacker. The methodology guides testers in the process of gathering the information and resources required for the testing, identifying possible attack vectors and defining threat models, attacking a single component of the voting system, and leveraging an initial compromise to take control of the entire system.

We also developed a number of custom tools in order to perform the required analysis. These include firmware readers and writers, support tools for the debugging of DREs, DTD readers and writers, and a framework to easily patch a firmware and test it.

By using our methodology and tools, we discovered a number of previously-unknown vulnerabilities in both the Sequoia and ES&S systems. Perhaps the most troubling finding was the pervasive presence of exploitable software defects (e.g., buffer overflows) allowing the execution of arbitrary code of an attacker's choosing. In some cases, these defects allowed us to gain full control of the affected machines. Another area of significant concern was the general lack or misuse of cryptographic techniques to ensure the integrity of critical election data. These oversights allow an attacker, for instance, to forge authentication tokens and cast a vote multiple times. A third problematic area was the incompleteness of the specification of the system requirements and the misconfiguration of the system environments. For example, the "autorun" feature was enabled in Sequoia's EMS and could be used by an attacker to covertly run a malicious program on the system. Finally, contrary to the claims of the vendors, the physical seals protecting access to critical components of the machines were, in almost all cases, not tamper-proof or even tamper-evident.

These vulnerabilities pervade each vendor's voting machines and allow a multitude of serious attacks to be executed under several threat models. Furthermore, when considered in the context of the system as a whole, they allow the execution of sophisticated attack scenarios. We designed, implemented, and successfully tested several of these scenarios on each of the analyzed systems.

As an example, we describe here a virus attack we executed against the ES&S system. In an ES&S voting system deployment, an attacker with access to a DRE loads a malicious firmware containing the virus into the machine either by exploiting a vulnerability or by directly modifying the on-board flash memory. When a master DTD is inserted into the DRE to initialize it for the election, the malicious firmware installs a copy of the virus on the DTD itself. Subsequent uses of the DTD to initialize other DREs result in those machines being infected through a ballot-loading exploit.

During pre-election logic and accuracy tests, the firmware behaves as expected. During the election, however, the malicious firmware carries out vote stealing attacks. Examples of these attacks include:

- Modifying the ballot such that the favored candidate is voted for even if not selected.
- Modifying uncompleted ballots when the voter has fled (a "fleeing voter" is a voter that has only partially completed the voting process and has left the voting station).
- Printing a ballot summary and indicating that voting is complete, waiting until the voter has left, voiding the requested selections, and then casting a modified vote.
- Simply casting a modified ballot that disagrees with the paper audit trail.

After the election has ended, the infected DTD is transported by an elections official to the county elections office, where the votes are transferred into the EMS. During this operation, a vulnerability in the EMS is exploited such that the virus is installed also in the EMS, allowing the possibility of further attacks against the election.

After the tallying and reporting process has completed, the virus remains dormant on the EMS host until the next election. At this time, the virus will infect the master DTD that is programmed to initialize the DREs for that jurisdiction, and the infection cycle will continue.

4. REFERENCES

[1] D. Balzarotti, G. Banks, M. Cova, V. Felmetsger, R. Kemmerer, W. Robertson, F. Valeur, and G. Vigna. Are Your Votes *Really* Counted? Testing the Security of Real-world Electronic Voting Systems. To appear in *Proceedings of the International Symposium on Software Testing and Analysis*, 2008. Available at `http://www.cs.ucsb.edu/~marco/data/papers/issta08_evote.pdf`

[2] P. McDaniel, M. Blaze, and G. Vigna. EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing. Ohio Secretary of State's EVEREST Project Report, December 2007.

[3] G. Vigna, R. Kemmerer, D. Balzarotti, G. Banks, M. Cova, V. Felmetsger, W. Robertson, and F. Valeur. Security Evaluation of the Sequoia Voting System. Top-To-Bottom Review of the California Voting Machines, July 2007.

Quantum Online Memory Checking

Qingqing Yuan
qqyuan@cs.ucsb.edu

Wim van Dam
vandam@cs.ucsb.edu

ABSTRACT

We consider the problem of storing information on an unreliable server, whose memory can be modified by a malicious party. Our main task is to design an online memory checker that is able to verify that the stored information has not been corrupted. When storing n bits of public information, the memory checker is allowed to have s private, reliable bits and for each bit retrieval it can query the public memory for t bits of information. Previously it has been shown that for classical memory checkers there is a lower bound $s \times t \in \Omega(n)$. Here we study quantum memory checkers that have s private qubits and are allowed to quantum query the public memory using t qubits. We propose a quantum online checker that usee $s \in O(\log n)$ qubits of local memory and $t \in O(\log n)$ qubits of communication with the public memory by using quantum fingerprints.

1. INTRODUCTION

The problem of memory checking was first introduced by Blum *et al.* [1] as an extension from the field of program checking. In this problem, a memory checker receives a sequence of "store" and "retrieve" operations issued to an unreliable memory from a user, and it receives responses to these requests from the memory. By making additional requests to the unreliable memory and using a small private and reliable memory for storing additional information, the checker is required to give correct answers to the retrieve operations with high probability. An online memory checker must detect the error immediately after receiving an errant response from the memory.

There are two main complexity measures regarding memory checkers: the *space complexity*, which is the size of its private reliable memory, and the *query complexity*, which is the number of queries made to the public (unreliable) memory per user request. The goal is to have a secure checker with low space complexity and query complexity against any probabilistic polynomial time adversary corrupting the publc memory.

With s be the space complexity and t the query complexity of an online memory checker, both Blum *et al.* [1] and Naor and Rothblum [3] proved that for classical online memory checking one has the lower bound $s \times t \in \Omega(n)$.

Here we present an online memory checker that uses quantum fingerprints, and which requires only $s \in O(\log n)$ bits of private memory and $t \in O(\log n)$ queries to the public memory. Specifically we show that for an error rate $\epsilon > 0$, it is sufficient for the memory checker to privately keep $O(\log(1/\epsilon))$ copies of the quantum fingerprints of the public memory with each fingerprint requiring $O(\log n)$ qubits.

2. CLASSICAL AND QUANTUM MEMORY CHECKERS

Here we extend the definition of memory checker to the quantum setting.

DEFINITION 1. ***Memory Checkers (see [1, 3]).*** *A memory checker is a probabilistic Turing machine $\mathcal{C}$ with five tapes: a read-only input tape for $\mathcal{C}$ to read the requests from user $\mathcal{U}$, a write-only output tape for $\mathcal{C}$ to write its response to the user's requests or that the memory $\mathcal{M}$ is "`buggy`", a write-only tape for $\mathcal{C}$ to write requests to $\mathcal{M}$, a read-only tape for $\mathcal{C}$ to read the response from $\mathcal{M}$, and a read-write work tape as a secret, reliable memory.*

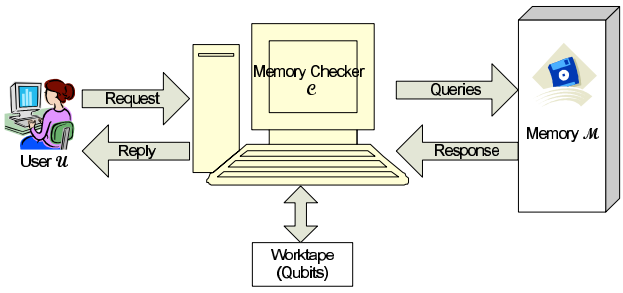


Figure 1: A quantum mechanical memory checker: The user presents classical "store" or "retrieve" request to the checker, which, with high probability, returns the correct answer or reports `buggy` when the memory has been corrupted. In the quantum mechanical scenario, the checker can make quantum queries to the memory, such that it acquires a superposition of values. In addition, the checker is also allowed to have a small, private and secure worktape that consists qubits.

In our quantum mechanical extension of this definition the input and output tape between $\mathcal{C}$ and $\mathcal{U}$ both remain classical, as well as $\mathcal{M}$. The checker $\mathcal{C}$, however, is now allowed to make quantum queries to the memory $\mathcal{M}$ and the secret

work-tape of $\mathcal{C}$ and the two read and write-only tapes between $\mathcal{C}$ and $\mathcal{M}$ now support quantum bits. This model is illustrated in Fig. 1.

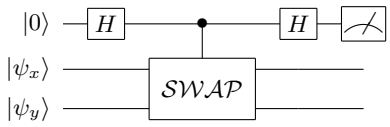
There are two important measures of the complexity of a memory checker: the size s of its secret memory (the *space complexity*) and the number t of bits exchanged between $\mathcal{C}$ and $\mathcal{M}$ per request from the user (the *query complexity*). Obviously, if the secret memory is sufficiently large, the solution to this problem is trivial as $\mathcal{C}$ can simply store the n bits on its work-tape. More interesting is the case where the space complexity t is sublinear (typically logarithmic) in n. As noted in the Introduction, it is known that for classical online memory checkers, we have the lower bound $s \times t \in \Omega(n)$ [3]. Below it will be shown that with quantum memory checkers one can get an exponential reduction on this lower bound.

3. QUANTUM ALGORITHM FOR ONLINE MEMORY CHECKING

In this section, we have the main theorem of this paper.

THEOREM 1. *There exists a quantum online memory checker with space complexity $s \in O(\log n)$ and query complexity $t \in O(\log n)$, where n is the size of the string being stored. This checker answers the user correctly with constant probability at least $1-\epsilon$ (with $\epsilon < \frac{1}{2}$)) when the memory $\mathcal{M}$ acts correctly, and it outputs* `buggy` *with probability at least $1-\epsilon$ when $\mathcal{M}$ has been corrupted.*

Let $x = x_1 \dots x_n$ be the string that the user $\mathcal{U}$ wants to write to the public memory $\mathcal{M}$. The memory checker $\mathcal{C}$ computes an error correcting code of x and writes the codeword $E(x)$ to $\mathcal{M}$. Here we use a locally decodable code (see for example Katz and Trevisan [2]) such that a single bit x_j of the original data can be probabilistically reconstructed by reading a small number of locations in the encoding $E(x)$.



The quantum algorithm for the online memory checker uses this code as follows. The memory checker maintains k copies of the quantum fingerprint $|\psi_x\rangle = \sum_j (-1)^{E_j(x)} |j\rangle$ of x in its private memory (the value of k will be determined later) and every time a "retrieve" instruction comes, it obtains k copies of fingerprints $|\psi_y\rangle$ of the current state of the public memory $\mathcal{M}$. Using the "controlled-SWAP" circuit above, the checker compares these new quantum fingerprints with those in the checker's private memory. The checker can detect any malicious changes that would corrupt the decoding of $E(x)$ to the public memory with high probability. When analyze the correctness and security of this quantum online memory checker, we have the following lemma.

LEMMA 1. *If a checker uses error correcting codes with Hamming distance between at least deltam (where $\delta > 0$ is a constant) and $\lceil \frac{\log \epsilon}{\log(1-2\delta+2\delta^2)} \rceil$ copies of $|\psi_y\rangle$ and $|\psi_x\rangle$, then the checker will detect the difference between the two fingerprints with probability at least $1-\epsilon$.*

Due to the space limit, we omit the proof of Lemma 1 and Theorem 1 here.

4. AN OPEN PROBLEM

The quantum online memory checker in this article employs quantum mechanism both in its local memory and in the communications with the public memory. A variation of this model is a checker that stores quantum information in its local memory, but communicates in classical bits with the public memory.

In a simultaneous message protocol, Regev and De Wolf have shown that, if one message is quantum while the other is restricted to be classical,it requires a $\Omega(\sqrt{n/\log n})$ bits/qubits to compute the EQUALITY function [4], and hence such a hybrid setting is not significantly more efficient than the purely classical setting. This result however does not directly translate into a new lower bound on the $s \times t$ complexity for quantum memory checking with classical communication.

Using the same techniques as in [3], a quantum online memory checker with classical queries can be reduced to a modified "consecutive messaging" (CM) protocol. In this CM protocol, Alice is allowed to send quantum messages to Carol and publish a quantum public message, while Bob is restricted to classical messages. For this CM protocol, there is an efficient solution as follows: after having received an input x, Alice computes her quantum fingerprints $|\psi_x\rangle$ and publishes them as a public message; Bob, receiving y, computes his quantum fingerprints $|\psi_y\rangle$ and compares it with $|\psi_x\rangle$; Bob then sends Carol the result of the controlled-SWAP test, who outputs the final result. The communication complexity for this protocol is $O(\log n)$.

Due to the difference between the quantum-classical CM model and simultaneous messaging protocols for EQUALITY testing, it is not easy to draw a conclusion for the lower bound of quantum online memory checking with classical communications. Nevertheless we conjecture that there is no efficient quantum online memory checker for this setting.

5. CONCLUSION

In this paper, we consider the problem of constructing an online memory checker. By using the quantum fingerprints, we reduce the space complexity s and query complexity t from $s \times t \in \Omega(n)$ to $s \in O(\log n)$ and $t \in O(\log n)$.

6. REFERENCES

[1] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[2] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of ACM Symp. on Theory of Computing*, pages 80–86, 2000.

[3] M. Naor and G. N. Rothblum. The complexity of online memory checking. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 573 – 584, 2005.

[4] O. Regev and R. de Wolf. Personal communication. Auguest 2007.

Managing Big Dataflow Tags with a Small Cache of Large Ranges

Mohit Tiwari, Banit Agrawal, Shashidhar Mysore, Jonathan K Valamehr, Timothy Sherwood

1 Introduction

Web applications routinely handle sensitive data such as financial transactions and private health records. While this makes security and privacy management critically important, a recent security assessment of 250 Web applications by the Application Defense Center found that at least 92% of the applications were vulnerable to some form of attack [11]. In addition, software "bugs" themselves have been estimated to cost the US economy $59.5 billion annually [6]. Increasingly sophisticated dynamic dataflow tracking tools are being widely seen as a critical part of the solution to these problems. In this paper, we present a dataflow tag management scheme that can be a key step in the practical realization of these tools.

2 Dataflow Tracking: Big Tags, Big Cost

The ability to tag and track data as it pumps through high performance microprocessors is proving increasingly beneficial, as it enables run-time checks for malicious code-injection [2], helps uncover cross-site scripting attacks [3, 5], makes privacy easier to manage [8], aids tracking memory errors more effectively [9], and helps to slice and examine full systems in novel ways [4]. All of these techniques rely on ISA-level extensions – shadowing all architecturally visible state with tags; creating dataflow rules that ensure tags are effectively tracked during execution; and forming policies around these tags that address issues of software security, debugging, and performance.

Many proposals have explored how tag tracking can be efficiently integrated into different stages of the processor pipeline to meet various analysis goals with low overheads [2, 8, 9, 3]. While current hardware proposals support *1 or 2 bit tags per word*, software-only schemes that use large *32-bit tags per byte* have been shown to be useful in many scenarios. For example, dataflow tracking schemes like [10] use 32-bit tags to not only identify a security breach but also provide precise information about the malicious input and the exploit's execution path; [1] uses 32-bit tags to track null pointer exceptions back to the program counter that stored the null value; and [4] uses 32-bit tags to track network packets across language runtimes, OS boundaries and over the network to understand and visualize complex distributed systems. Further, as shown in our original article [7], language runtimes like Sun JDK and programs that operate on strings like Firefox, bzip, parser etc. perform byte-level accesses once in every 10 to 20 x86 instructions (average over all SPEC programs is once in 1904 x86 instructions). Existing schemes for storing tags do not scale to support such emerging tools, and using an existing tag cache for large, byte-level tags introduces an average slowdown of 9X for a 4KB tag cache.

Ideally, hardware support for a dataflow tracking system should store large multi-bit tags efficiently and support fine-grain tracking at low performance overhead. We have observed that dataflow tags naturally exhibit a high degree of spatial-value locality, and can be stored in compressed form as tags which cover non-aligned, contiguous ranges of memory. However, there are several challenges we need to overcome. One, even though tags naturally group into contiguous address ranges, they do so incrementally through a barrage of stores (as opposed to a single allocation) and for each update, all possible range overlaps need to be resolved to store completely non-overlapping ranges. Two, the maximumnumber of ranges that are created during a program execution varies from a few tens to a few thousands, and while the least number of ranges across benchmarks is surprisingly small, the largest number precludes storing *all* the ranges on-chip. This means we have to maintain a **cache** for tags and deal with 'misses' on tag reads and evictions when new tag ranges are created. Three, incremental aggregation causes the cache to store unaligned ranges that vary from just 4B to almost 4GB in size. For example, a range cache snapshot for `gcc` shows more than 82% of the cached ranges were below 64B in size while the largest was over 2MB; this rules out using fixed (say, page) size range entries. And finally, frequent updates (every 5 to 6 x86 instructions on average) require stitching together ranges with very high throughput. In this paper, we propose a *Range Cache* that associates tags with non-aligned ranges and can track 32-bit tags at byte-level granularity while still supporting the required rate of updates.

3 Range Cache Architecture

The Range Cache stores a set of 128 ranges each having a $start$ and end address. Each range has an associated metadata which in our prototype is 32-bits. A *read request* should return the tags of the ranges it overlaps, while an *update request* sets the new tag value to be now associated with the new range of addresses. The tag values previously associated with those addresses are overwritten, thus each individual address has one and only one tag associated with it. This new range can overlap with existing ranges in complex ways that may require the existing ranges to be split up so as to always store non-overlapping ranges.

Figure 1 shows the basic 2-stage pipeline at the core of our approach. Ranges are stored directly as a set of memory cells for $start$ and end with a separate comparator on each memory address. Deciding whether a given address overlaps with a range in the set then translates into a simple parallel comparison of the query address over the set of stored $start$ and end values (in stage 1 of the pipeline).

The state machine in the second stage handles all of the overlapping ranges that can occur (explained in detail in [7]). It handles the overwhelmingly common cases fast: read requests that hit only one stored range complete without stalls, as do "silent" updates that do not modify the tag value. Of the remaining updates, most updates hit only one range and is handled with a 2-cycle stall, while updates that overlap multiple ranges take a few more cycles. The worst cases are when reads miss or when an update creates a new range that causes the range cache to overflow. Both cases require a software handler to fetch or write back a range entry to the secondary tag store. The secondary store is implemented as a two-level trie with the first level nodes storing tags for 64B of the address space and the leaf nodes storing byte-level tags if required. On a read miss, we only fetch a range up to 64B in size so that fetching in a huge range doesn't become a bottleneck.

Finally, to reduce pipeline stalls, we use the fact that two pipelined accesses are only dependent on one another if they access the same ranges. By speculating that all accesses are independent, and squashing those that are not, we have shown that the stall cycles can be reduced by 32% on average([7]).

3.1 Results and Evaluation

We have implemented a synthesizable RTL model of the Range Cache in Verilog. We also use this RTL model to count the exact number of cycles required in those cases where complex operations are required (such as deleting multiple range entries for example) so that we can accurately reflect those counts in our performance

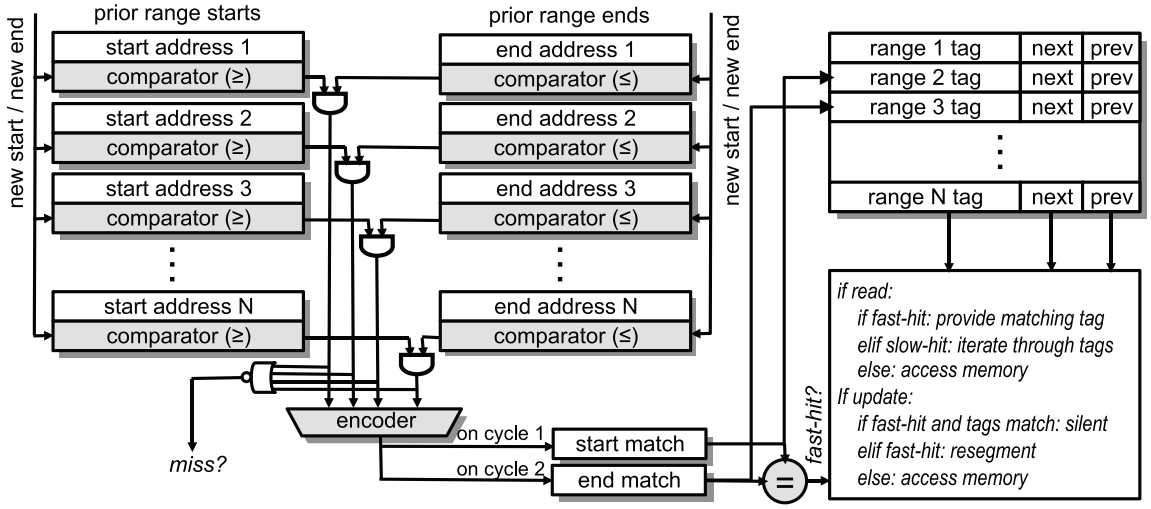


Figure 1: Range Cache Architecture: The first pipeline stage matches the start and end points of the new range, while the second stage is a state machine that performs tag lookup and decides on the aggregated range entry to be written back.

estimation. This hardware design also allowed us to estimate the area of our approach. The controller itself is just under 3000 logic gates (which does not include the memory for the actual tag storage itself), and when combined with the required memory, a 128-entry range cache storing 32-bit tags is approximately the same size as 4KB of memory.

We evaluate the miss-rate and estimate the performance impact across several dataflow tracking tools – 1-bit Taint Tracking for an online bookstore application developed on Ruby-on-Rails framework with Mongrel web server, 2-bit Definedness Tracking similar to Valgrind-Memcheck on SPEC benchmarks and Java-based XML parser, and 32-bit Dataflow Tomography correlating usage of network data bytes for various networking applications like ssh, scp, lynx, traceroute, and the Ruby-on-Rails based online bookstore application. Instructions executed by the range cache miss-handler and memory hierarchy pollution are the two major contributors to performance overhead, and while in this section we show that range cache rarely misses, a detailed breakdown of sources of slowdown can be found in our original article [7].

Figure 2 shows the average miss rate of a range cache for all tested tools while the cache size varies from 4 to 128 entries. Results show that a range cache improves markedly with size up to 32 entries and the improvement plateaus out thereafter. At 128 entries, the average miss rate for definedness tracking on SPEC benchmarks is 1.57% (sans `ammp` the average drops to 0.07%), while the network tainting tools average is 0.54%. In comparison, a conventional 4KB tag cache misses 20% for definedness tracking and 30% for network taint tracking. The upshot of these misses is that a conventional tag cache introduces up to 6X as many memory references and 10X as many L2 misses per 1000 instructions as the native program, while the range cache introduces 1.5X the memory references and 2X as many L2 misses per 1000 instructions. As a result, a tag cache introduces a slowdown of almost 9X over native execution while a range cache limits the slowdown to 1.52X.

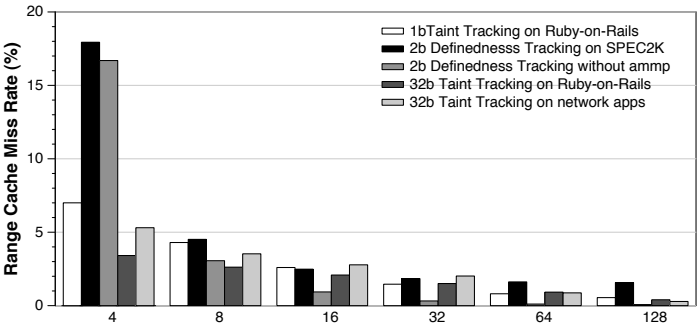


Figure 2: The graph shows miss rate of Range Cache for varying number of range entries

References

[1] M. D. Bond, N. Nethercote, S. W. Kent, S. Z. Guyer, and K. S. McKinley. Tracking bad apples: reporting the origin of null and undefined value errors. *SIGPLAN Not.*, 42(10):405–422, 2007.

[2] J. R. Crandall and F. T. Chong. Minos: Control Data Attack Prevention Orthogonal to Memory Model. In *MICRO 37: Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, pages 221–232, Washington, DC, USA, 2004. IEEE Computer Society.

[3] M. Dalton, H. Kannan, and C. Kozyrakis. Raksha: A Flexible Information Flow Architecture for Software Security. In *34th Intl. Symposium on Computer Architecture (ISCA)*, 2007.

[4] S. Mysore, B. Mazloom, B. Agrawal, and T. Sherwood. Understanding and Visualizing Full Systems with Data Flow Tomography. In *ASPLOS-XIII: Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2008.

[5] J. Newsome and D. Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *12th Annual Network and Distributed System Security Symposium (NDSS '05)*, 2005.

[6] NIST News Release. http://www.nist.gov/public_affairs/releases/n02-10.htm. 2002.

[7] M. Tiwari, B. Agrawal, S. Mysore, J. K. Valamehr, and T. Sherwood. A Small Cache of Large Ranges: Hardware Methods for Efficiently Searching, Storing, and Updating Big Dataflow Tags. In *Proceedings of the International Symposium on Microarchitecture (Micro)*, 2008.

[8] N. Vachharajani, M. J. Bridges, J. Chang, R. Rangan, G. Ottoni, J. A. Blome, G. A. Reis, M. Vachharajani, and D. I. August. Rifle: An architectural framework for user-centric information-flow security. In *MICRO 37: Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, pages 243–254. IEEE Computer Society, 2004.

[9] G. Venkataramani, B. Roemer, Y. Solihin, and M. Prvulovic. MemTracker: Efficient and Programmable Support for Memory Access Monitoring and Debugging. In *13th International Symposium on High-Performance Computer Architecture (HPCA-13)*, February 2007.

[10] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. February 2007.

[11] WebCohort Inc. Only ten percent of web applications are secured against common hacking techniques. http://www.imperva.com/company/news/2004-feb-02.html. 2004.

Automated Verification of MVC Web Applications

[Extended Abstract] *

Chris Bunch
University of California, Santa Barbara
cgb@cs.ucsb.edu

Taylor Ettema
University of California, Santa Barbara
tettema@cs.ucsb.edu

ABSTRACT

Recent years have witnessed an explosion of web application development and mainstream use, with the much-hyped "Web 2.0" phenomena of increasingly dynamic sites and user-generated content only serving to fuel the growth. The security and reliability of these web applications is critical, yet few tools currently exist to verify correctness. Large application development frameworks exist to speed up development and deployment, though major vendors do not offer the power of correctness checking long provided by tools traditionally applied to critical, compiled applications. We introduce a web application verification framework that allows the developer to enforce state machine semantics on a web application, and enables the checking of CTL properties against the whole application space. We demonstrate that an enforced state machine model aids in the design of a more complete, secure application, and by allowing the designer to verify CTL properties, potentially serious logic errors can be easily avoided.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Software / Program Verification —*Model checking*

General Terms

Model checking, Model-View-Controller architecture

Keywords

WebFlow, SMV, PHP/Zend

1. INTRODUCTION

Automated verification has long proven itself an invaluable tool in verifying correctness of compiled, realtime, critical applications and embedded systems [1]. As web applications gradually replace traditional, locally installed/compiled applications in many critical domains, verification of this new programming model has become of significant importance.

Verification of web applications is a relatively recent area of study, in which classic verification techniques are applied to web application code in order to prove the correctness of various aspects of the system. However, the challenges of verifying web applications are quite different from traditional applications. Perhaps the most significant difference is the nature of the state, and the transitions between states. A typical application can be modeled as a state machine in a relatively straight-forward way, as the program code has control over execution flow. The state of the machine can be modeled as a union of the program counter and the contents of registers and memory. The state of a web application, however, is less straight-forward, as a non-trivial use case of a typical web application consists of dozens of transactions over a period of time, each triggered by an HTTP request that can invoke various fragments of code across the entire site.

The key realization in verifying web applications is that the most interesting properties that would benefit greatly from verification span over many HTTP requests. This assertion suggests that line-by-line verification of source code, while mildly useful, does not catch the most damaging logic errors that require a series of client-server interactions before they are exposed. Furthermore, this type of bug is increasingly pervasive, as many developers fail to grasp the inherent security risks associated with web application development. Some of the most common bugs stem from a visit to a particular script/page that was unanticipated by the developer under the present conditions. While many modern application development frameworks have sought to alleviate this problem, they have not eliminated it.

In order to introduce CTL verification to web applications, we have developed a plugin for the popular Zend Framework for PHP that allows the developer to describe in XML the desired functionality of the application as a state machine. The plugin ensures that only the transitions described by the XML specification are allowed, thus ensuring that the specification is an accurate depiction of the web application's flow of execution. Additionally, we provide a verification application that converts an XML specification into an executable SMV model, allowing the user to run CTL formulas against the model. The results of the SMV run are displayed to the user, providing either a reassuring confirmation of sound correctness or an indication of a property violation, along with a characteristic execution trace.

2. WEB APPLICATION STATE MACHINE PLUGIN

Building an automated verification framework for modern MVC-based web applications first required a method of defining and enforcing state machine semantics on a web application's flow of execution. We drew our inspiration from

*A full version of this paper is available at `http://cs.ucsb.edu/~cgb/papers/automatedVerification.pdf`

```
<state id='index'>
    <epsilon_edge to='login'/>
    <predicate_edge name='loginForm' label='yes' to='home' />
</state>

<state id='login' on_post='loginForm'>
    <epsilon_edge to='index'/>
    <predicate_edge name='loginForm' label='yes' to='home' />
    <predicate_edge name='loginForm' label='no' to='login' />
</state>

<state id='home'>
    <epsilon_edge to='index'/>
</state>
```

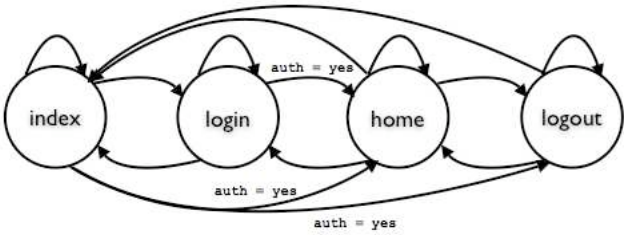


Figure 1: Representation of XML transitions as a finite state machine

the Spring Framework for Java [2], which includes a novel feature called "WebFlow" which allows the developer to segment the web application into distinct "flows" that each have an enforced flow of execution, as specified by an XML document. By leveraging the extendible capabilities of the Zend Framework, we developed a router plugin (WASM) that provides similar functionality, reading a state machine specification as input, while enforcing the described flow of execution with each HTTP request.

Upon visiting the application for the first time, a session is established. The session stores the last state the user visited, along with the values of any atomic properties that have been computed in the past. Each request submitted by the user is routed through the WASM plugin before dispatching. The requested state (such as a page view) is compared with the last visited state. If an edge connecting the last state to the requested state exists, and all the predicates on the edge (if any) are satisfied, then the router allows the request to be dispatched unmodified. However, if there is a problem with the request, the router defaults back to the last viewed state, effectively performing a page refresh.

3. AUTOMATED VERIFICATION FRONT-END TOOL

By exploiting the MVC design pattern and creating the WASM Zend Framework plugin, we now have an enforceable state machine that can be modeled with a standard verifier tool. While verification and state machine modeling of web applications have each been done independently [3], our key contribution is combination of these two elements in order to provide the ability to perform meaningful property verification on real-world applications.

The WASM Verifier Tool (WASM-VT) provides this capability, bridging the gap from the XML transition specification to an executable SMV model that can be verified. WASM-VT is, in itself, a web application, that is designed to be used by the developer to perform verification tasks. The user starts by providing the tool with the XML transition specification file. WASM-VT parses the file and transforms it into an executable SMV file behind the scenes. Meanwhile, the user is presented with a list of states and atomic properties that the XML file contains. The user is then prompted for a list of CTL properties to verify. Upon submission, the tool invokes SMV in the background, verifies the properties, and returns the result to the user. The tool either indicates that the formula(s) hold, or, in the event of a violation, the user is presented with a characteristic path of execution that violated the constraint.

4. LIMITATIONS AND FUTURE WORK

Currently, WASM is restricted to boolean logic for verification of state transitions. More advanced conditional checks (such as bounds checking) could prove very useful to the developer, providing the capability to design more complicated transition predicates that rely on the value of an integer, for example. This capability is not trivially implementable, however, and we plan to investigate the viability of this addition to our verifier in future work.

5. CONCLUSION

In this work, we present a viable automated verification solution for MVC web applications written in PHP using the Zend Framework. By controlling the dynamic behavior of the application using the WASM plugin, we are able to provide more meaningful static verification opportunities to the developer. In what may be the first deployment of automated verification on web applications at the session scope (i.e. verifying logic across a series of request-response events), we exploit the organization inherent to the MVC architecture in order to expose a system of states, transitions, and atomic properties that can serve as input to a verification tool. WASM and the WASM-VT application provides a verification framework for web applications that can be used to check correctness of truly useful properties across the scope of the entire application, from the first client request to the last server response.

6. REFERENCES

[1] E.M. Clarke, O.M. Grumberg, and D.A. Peled. *Model Checking.* 1999.

[2] K. Donald, E. Vervaet. *Spring Web Flow*, July 2005. `http://springframework.org/webflow`

[3] E. Di Sciascio , F.M. Donini, M. Mongiello, R. Totaro, and D. Castelluccia. *Design Verification of Web Applications Using Symbolic Model Checking.* 2005

CycleNet: Empirical Analysis of 802.15.4 in Mobile Scenarios

[Extended Abstract]*

Navraj Chohan and Camilla Fiorese
Dept. of Computer Science
University of California, Santa Barbara
{nchohan, camilla}@cs.ucsb.edu

ABSTRACT

We present empirical data of communication between 802.15.4 devices in static and mobile scenarios. We evaluate the body factor between two devices in a cycling environment and show it has a significant impact on communication. Our findings show that RSSI is severely hampered as an indicator of reception quality and that LQI serves as a much more reliable indicator in a mobile setting. Additionally, we show that different bicycling speeds do not adversely affect 802.15.4 communication.

1. INTRODUCTION

With the rising cost of oil, interest in alternative means of commuting has increased tremendously. Throughout the world, bicycles are the main source of transportation and bicycling is participated in as both sport and hobby. Nevertheless, many cities lack adequate bicycle paths and accommodations for cyclists. With foreseen greater interest in bicycling, cyclist themselves can take part in networking amongst fellow riders to share information with each other and city planners. Lightweight, low power, and autonomous embedded devices can allow for a greater experience for people throughout the bicyclist spectrum; from the casual rider to the cycling enthusiast.

We examine different aspects of inter-cycle communication to better understand factors which can affect application and protocol design for people-centric sparse mobile networks. Our research looks to model mobile to mobile communication between bicycles using 802.15.4 devices with empirical measurements. The devices we use for our experiments are MICAz motes which are produced by Crossbow. They have the CC2420 Chipcon radio which uses the unlicensed 2.4 GHz ISM spectrum with a data rate of 250kbps. The MAC layer is CSMA and the physical layer does QPSK modulation. It provides a received signal strength indicator (RSSI) and a link quality indicator (LQI) for each packet received. The RSSI is the power of the signal at the receiver and the LQI value is the chip error rate of the demodulated signal. Packet reception rate (PRR) is extracted through the use of sequence numbers. Our mote software is written in TinyOS.

The case of bicycles communicating statically is first presented to attain a sound understanding of human interference. The body of the rider is shown to give large amounts of attenuation. We call this source of attenuation the *body factor*. Furthermore, we show that within close range (one bike length) the limited speeds of bicycles have little to no affect on bicycle communication in comparison to the static scenario. Yet, the body factor is a large source of attenuation in which the communication range is cut dramatically. This is in large part due to the orientation of cyclists in relation to each other. In general, there is a leader and a series of followers. Our experiments for both static and mobile cases are reflective to these formations. Likewise, our research caters to bicyclists who participate in groups rather than by themselves.

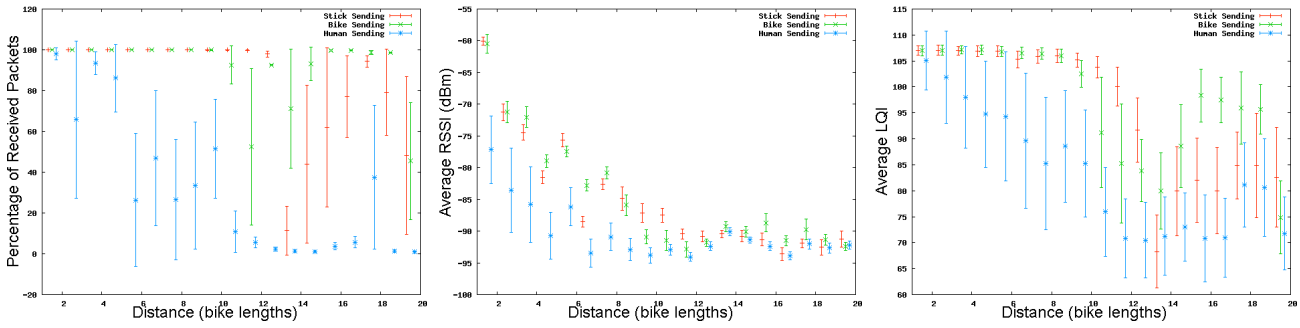
There is a dearth of research in mobile communication for people-centric sensing. Previous studies have had limited findings on either the body factor or mobile scenarios with low power radios. Much previous work has looked at static communication which reports phenomena such as asymmetric links, gray receptions areas, antenna orientation sensitivity, height sensitivity, and other complex radio behavior [4, 5, 2, 6]. In [3] the authors present empirical experiments for the characterization of 802.15.4 devices worn by people in low mobility environments. But, there has yet to be any characterization of people-centric communication in highly mobile environments. Our research is the first study to deal directly with the issue of speed variability when characterizing 802.15.4 communication. In addition, we extrapolate on the body factor for such environments.

Our work looks at low power inter-bicycle communication for systems such as [1] and presents the factors which can cause unreliable connectivity in mobile bicycle networks.

2. RESULTS

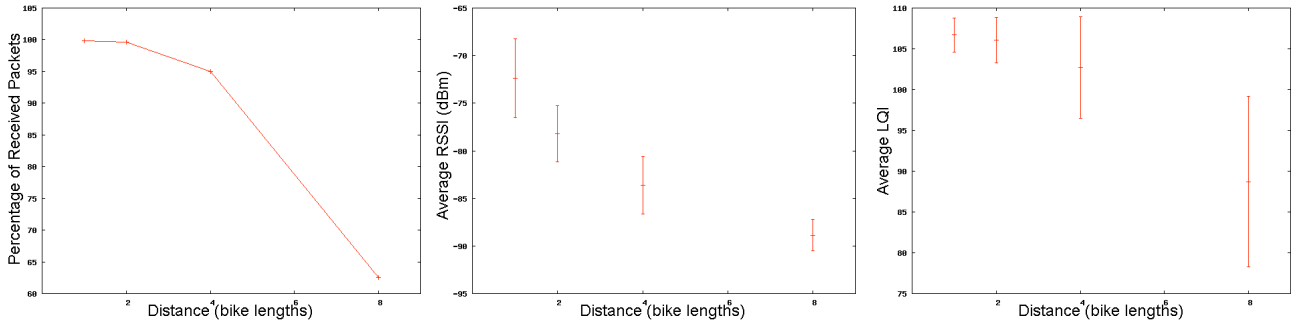
Figure 1 shows the PRR, RSSI, and LQI for our first experiment where a sender node was followed by a series of receiver nodes all located on an open field. The sender node was ei-

*A full version of this paper is available at http://www.cs.ucsb.edu/research/tech_reports/



(a) PRR (b) RSSI (c) LQI

Figure 1: Packet reception quality for the static scenario



(a) PRR (b) RSSI (c) LQI

Figure 2: Packet reception quality for the mobile scenario

ther mounted on a wooden stick, a bike, or a bike with a human, respectively. With the stick or bike experiment the reception quality is near 100% until the 11th bike length at which point there is a drop off. With the introduction of the body factor there is a dramatic decrease in packet reception. The disturbance is also seen in the RSSI and LQI. The RSSI enters a gray zone, where it is no longer a good indicator of reception quality, when past 11 bike lengths. Yet, with the addition of the body factor this gray zone is reached in half the distance. Moreover, there is great variance with the body factor when under 11 bike lengths showing that packet reception can wildly vary even in close proximity.

Further experiments show that with different body types, where our experiments ranged from 125lbs to 207lbs, the body factor gave the same amount of attenuation. Figures for that experiment are not shown here.

From the results and insights we gained from a static setting we transition to a mobile scenario where the body factor is always present.

We found that with varying speeds of 5, 10, and 15mph the reception quality was the same throughout when bicycling directly behind the sender. Packet reception stayed close to 99% which was also seen in the LQI and RSSI.

Figure 2 shows the reception quality between a sender and receiver traveling at 10mph at different distances of 1, 2, 4, and 8 bike lengths. Consider the average PRR for the four different distances; the reception experience a tremendous degradation when the distance increases. For 8 bike lengths the reception is below 65%, far lower than our reported static case without body interference. We observe the degradation of the signal with the RSSI entering the gray area. But, LQI still proves to be a better predictor of PRR.

3. REFERENCES

[1] S.B. Eisenman, E. Miluzzo, N.D. Lane ans R.A. Peterson, G.S. Ahn, and A.T. Campbell. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2007.

[2] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An Empirical Characterization of Radio Signal Strength Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas. *Springer-Verlag Heidelberg Lecture Notes in Computer Science*, 3868(2006), 2006.

[3] E. Miluzzo, X. Zheng, K. Fodor, and A.T. Campbell. Radio Characterization of 802.15.4 and its Impact on the Design of Mobile Sensor Networks. *Springer-Verlag Heidelberg Lecture Notes in Computer Science*, 4913(2008), 2008.

[4] J. Polastre, R. Szewczyk, D. Culler, and W.S. Conner. Telos: Enabling Ultra-Low Power Wireless Research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, Los Angeles, California, USA, April 2005.

[5] K. Srinivasan and P. Levis. RSSI is Under Appreciated. In *Workshop on Embedded Networked Sensors (EmNets)*, May 2006.

[6] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

EUCALYPTUS *

An Open Source Infrastructure for Cloud Computing Research

Chris Grzegorczyk
grze@cs.ucsb.edu

Dan Nurmi
nurmi@cs.ucsb.edu

Graziano Obertelli
graziano@cs.ucsb.edu

Sunil Soman
sunils@cs.ucsb.edu

Lamia Youseff
lyouseff@cs.ucsb.edu

Dmitrii Zagorodnov
dmitrii@cs.ucsb.edu

Rich Wolski
rich@cs.ucsb.edu

ABSTRACT

Elastic Computing, Utility Computing, and Cloud Computing are synonymous terms referring to a popular virtualization based computing paradigm that allows users to "rent" Internet-accessible computing capacity on a for-fee basis. While a number of commercial enterprises currently offer Cloud hosting services and several proprietary software systems exist for deploying and maintaining a computing Cloud, standards-based open-source systems have been few and far between.

EUCALYPTUS is an open-source software infrastructure implementing "cloud computing" on clusters. The overarching objective is to provide an overlay platform amenable to installation and use on systems typically available to members of the research community. Currently, EUCALYPTUS is compatible with Amazon's EC2 in the scope of its functionality and can be utilized using Amazon's own tools. The system uses only commonly-available open source tools making it easy to install, modify, extend and maintain. In this paper, we discuss the design properties of EUCALYPTUS and evaluate an example installation by comparing it with EC2 to validate our approach.

1. INTRODUCTION

Cloud Computing has emerged as a popular virtualization-based computing paradigm and has seen rapid uptake in the small-and-medium business e-commerce market place. First deployed by Amazon.com, a number of service hosting enterprises and technology providers have since developed "utility," "cloud", or "elastic" product and/or service offerings [3, 1, 9, 8]. Fundamentally, the cloud computing model is to provide a large user base with the ability to program some specified fraction of the resources hosted by a scalable service provider (e.g., Google [7], Amazon [4], SalesForce [10], 3Tera [2], etc.) through one or more well defined service interfaces. However, the proprietary nature of existing systems has resulted in a derth of information about how they work: fundamental questions are unasked or unanswerable. Currently, it is not possible (or at least not easy) for researchers to build, deploy, modify, instrument, or experiment with a cloud infrastructure under their own control.

Here we present EUCALYPTUS – Elastic Utility Computing Architecture for Linking Your Programs to Transiently Useful Systems – an open source service overlay that implements cloud computing as a user hosted service on top of existing resources. A EUCALYPTUS installation aggregates a set of virtualization-enabled hardware resources into a "cloud" allowing a user to control collections of networked virtual machine instances. This paper focuses on specific innovations needed to provide an easy-to-install and maintain cloud service related to virtual machine control, network overlay provisioning, security and user management, and modularity/extensibility.

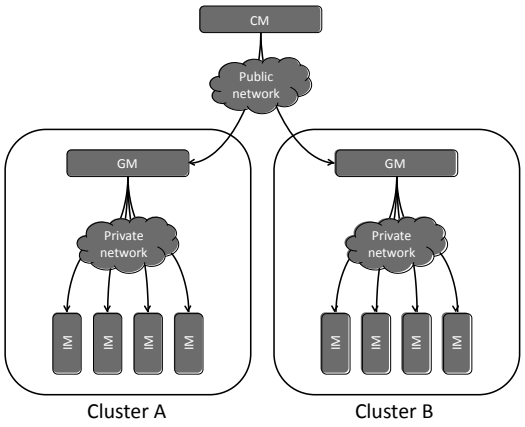


Figure 1: EUCALYPTUS architecture overview.

2. EUCALYPTUS

The architecture of EUCALYPTUS is simple, flexible and modular with a hierarchical design reflecting common resource environments found in academic settings. The system allows users to start, stop, and access virtual machines using Amazon EC2's SOAP and Query interfaces (i.e., with Amazon's provided tools). Currently, we support VMs that run atop the Xen [5] hypervisor, but plan to add support for others [6, 12] in the near future.

Three tiers of components comprise a EUCALYPTUS installation (Figure 1): **Node Manager** controls the execution, inspection, and termination of VM instances on the host where it runs. **Cluster Manager** gathers information about and schedules VM execution on specific nodes

9*http://EUCALYPTUS.cs.ucsb.edu/

and manages the virtual network. **Cloud Manager** is the entry-point into the cloud for users and administrators. It processes user-initiated or administrative requests, making high-level VM instance scheduling decisions, processing service-level agreements (SLAs) and maintaining persistent system and user metadata.

Our design makes the following contributions as a research platform: First, EUCALYPTUS leverages existing Linux packaging support. The target resources need only run a standard Xen-enabled kernel. Second, well-defined *interfaces* enable extension and modification through a language-agnostic web services approach and traditional language-level methods. Additionally, we make use of WS-Security policies to secure internal communication. Third, a EUCALYPTUS installation can function in an environment where only a certain "head node" is externally routable while "compute nodes" are on a private network. Fourth, a virtual network connects instances across multiple clusters to a virtual distributed ethernet isolated from the physical network [11].

3. EVALUATION

We have installed EUCALYPTUS on a small cluster and made it publicly available as Our Public Cloud (OPC) [1]. The cluster consists of 7 nodes on an isolated network and one frontend providing external network access. Each system has two Intel Xeon 3.2GHz processors, 3GB of RAM and 40GB of disk.

The OPC is used here as the venue for an experiment illustrating the cost in time for instantiating a collection of instances. For this experiment, we measure the total time between an instance execution request to the point when we can first detect that the instance is running. Figure 2 shows two empirical cumulative distribution functions that allow us to examine both the magnitude and variance of time taken to create instances in EC2 and the OPC. Each point represents the percentage (Y axis) of instance creation trials that took at least the number of seconds denoted at the point's corresponding position on the X axis. For both cases (one and eight instances) all empirical quantiles in the OPC case are lower than those of EC2. The implementation of the OPC does seem to compare well with that of the system it emulates suggesting that its implementation is, at least, relatively high performance.

4. CONCLUSION

In this work, we EUCALYPTUS: an open-source implementation of a cloud computing system. Presently, we (and our users) have successfully deployed the complete system on resources ranging from a single laptop (EC2 on a laptop) to small Linux clusters (48 to 64 nodes). Benchmarking EUCALYPTUS against EC2 reveals that it is relatively efficient. We plan to exploit EUCALYPTUS as a platform for investigating new ideas such as dynamic SLA generation, virtual networking topologies, virtual IP mobility, secure cloud infrastructure, and novel user/administrator interfaces.

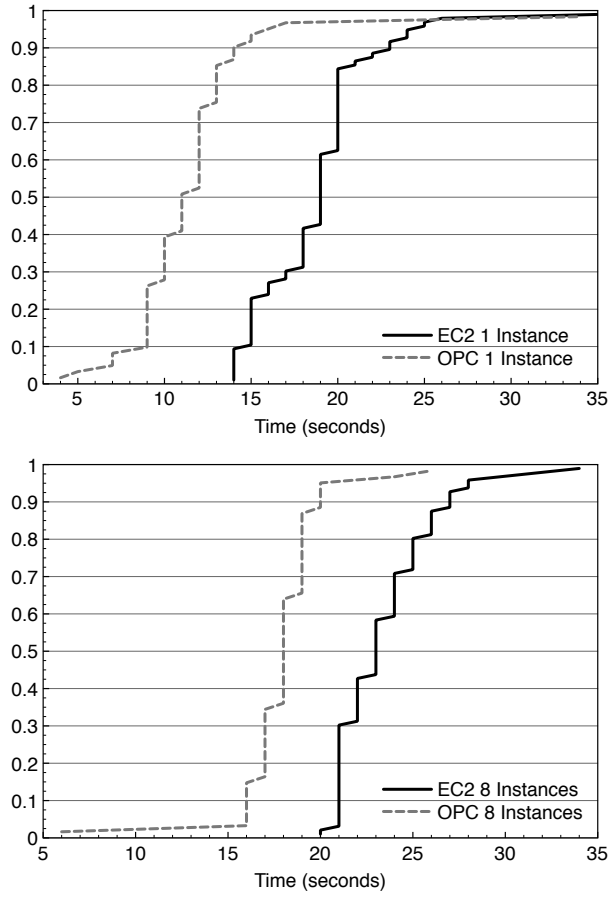


Figure 2: Empirical CDF comparing the number of seconds taken to start one and eight VM instances within EC2 and EUCALYPTUS.

5. REFERENCES

[1] 3tera. 3tera. "http://3tera.com".

[2] 3Tera home page. `http://www.3tera.com/`.

[3] Amazon. Amazon Elastic Compute Cloud (Amazon EC2). "http://ec2.amazonaws.com/".

[4] Amazon.com home page. `http://www.amazon.com/`.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[6] F. Bellard. QEMU, a Fast and Portable Dynamic Translator. *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, pages 41–46, 2005.

[7] Google – `http://www.google.com/`.

[8] Google. Google's AppEngine. "http://appengine.google.com".

[9] S. Microsystems. Network.com. "http://network.com".

[10] Salesforce Customer Relationships Management (CRM) system. `http://www.salesforce.com/`.

[11] Virtual distributed ethernet (vde) home page – `http://vde.sourceforge.net/`.

[12] Vmware home page – `http://www.vmware.com`.

[1] `http://eucalyptus.cs.ucsb.edu/wiki/EucalyptusPublicCloud`

Analyzing Performance and Efficiency of Smoothed Particle Hydrodynamics

Rama C. Hoetzlein
Media Arts & Technology Program
University of California, Santa Barbara
Santa Barbara, CA 93106-5110
rch@umail.ucsb.edu

Tobias Höllerer
Department of Computer Science
University of California Santa Barbara
Santa Barbara, CA 93106-5110
holl@cs.ucsb.edu

ABSTRACT

With the rise in performance of modern GPUs, Smoothed Particle Hydrodynamics is an increasingly attractive solution for real-time simulation of fluid flows in visual effects for film and games. Starting with simulations of 2000 particles at 20 frames per second in 2003 [4], smoothed particle hydrodynamics has now been simulated with over 242,000 particles at 4 fps using GPUs [6]. While performance has clearly increased, the terms *performance* and *efficiency* are often used interchangeably. Results of simulations, as published in graphics journals such as SIGGRAPH, are typically reported by giving the number of particles and frame rate for a particular combination of CPU and graphics hardware. This makes comparisons of algorithm implementations difficult since authors must deduce algorithm efficiency for differing hardware. The development of concrete metrics of performance and efficiency will facilitate better comparison between results. Simple metrics are presented here with an analysis of real-time simulations over the past five years.

Categories and Subject Descriptors

I.3.5.i [**Object Modeling**]: Physically based modeling; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Smoothed particle hydrodynamics, Simulation, GPU

1. INTRODUCTION

We present a simple metric for evaluating performance and efficiency of real-time particle-based simulations. These metrics suggest new areas for increasing efficiency of smoothed particle hydrodynamics, which are incorporated into our fluid simulator, FLUIDS v.1. In addition, we use these metrics to deduce trends in algorithm implementation over time. Our initial findings show that while performance has greatly increased due to new GPU hardware, there has been a gradual decline in algorithm efficiency over time.

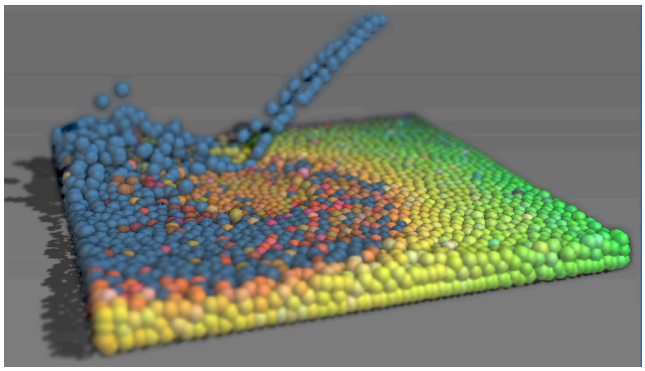


Figure 1: FLUIDS v.1 simulating 3000 particles at 45 fps with shadow maps and depth of field. Paint mixing is simulated with colored particles.

2. METHODS

To standardize simulation results we create simple metrics for performance and efficiency. First, we combine frames per second and number of particles to measure raw performance as the number of particles simulated per millisecond.

$$\mathbf{P}_{raw} = \#particles * fps * (1/1000)$$

Second, we estimate algorithm efficiency as the performance achieved on normalized hardware by dividing raw performance by hardware performance as measured in gigaflops.

$$\mathbf{E} = \mathbf{P}_{raw}/\mathbf{P}_{hardware}$$

The units of E are number of particles simulated in 1 ms on 1 gigaflop of hardware. Table 1 shows the author, year, performance and efficiency of several key real-time smoothed particle hydrodynamics papers from 2003 to 2008. All methods compared use a spatial grid to give $O(kn)$ behavior. Raw simulation rates are given without rendering. While the typical spatial grid technique used in SPH scales linearly, we also observed efficiency differences based on number of particles. In the table, the highest performance measure provided by the author in reported.

Performance estimates for CPU versus GPU hardware are more difficult as the GPU implementations introduce parallelism and memory transfer overhead. Raw results are reported, so these factors will be observed in our efficiency measures. In the future we hope to quantify and eliminate this overhead from our metrics. NVIDIA's own demo achieves the highest GPU efficiency (5.69), similar to Harada

Table 1: Performance and efficiency of real-time smoothed particle hydrodynamics simulations.

Author	Year	# Particles	FPS	Performance	Hardware	Gflops	Efficiency
Muller[4]	2003	2200	20	44.00	P4 1.8 ghz	3.6	12.22
Amada[1]	2003	2000	30	60.00	P4 2.8 ghz	5.6	10.71
Kontar[3]	2004	2000	18	36.00	XP 2200	4.4	8.18
Horvath[2]	2007	30000	0.50	15.00	P4 3.0 ghz	6.0	2.50
Harada (CPU)[6]	2007	262144	0.15	39.32	X6800 2.93	5.86	6.71
Harada (GPU)	2007	262144	4.23	1108.86	8800GTX	345.6	3.21
Harada[7]	2007	49153	17	835.60	8800GTX	345.6	2.42
Zhang[8]	2007	60000	15	900.00	8800GTX	345.6	2.60
NVIDIA[5]	2008	32768	60	1966.08	8800GTX	345.6	5.69
FLUIDS v.1 (CPU)	2008	3000	45	135.00	XP64 3.2 ghz	6.4	21.09

for the CPU (6.71), but still well below Muller's original paper (12.22) which implements a cache-coherent algorithm to optimize for the CPU. This suggests that both industry and academic implementations have not yet been fully optimized for the GPU.

The historical trends are also interesting. While overall performance has jumped by 20x due to new hardware, algorithm efficiency appears to have gradually declined even for the same hardware. While providing generous estimates for GPU parallelism overhead, implementations still appear to be less efficient than earlier authors. This supports our view that standardized measures of performance and efficiency are needed for publications in this area.

3. IMPLEMENTATION

A study of algorithm efficiency has suggested specific areas for improvement. Several of these improvements are incorporated into FLUIDS v.1, a simple, fast, open source implementation of Smoothed Particle Hydrodynamics. While currently running on the CPU only, FLUIDS v.1 is shown to be 3x more efficient than Harada [6], suggesting our GPU version should support 200,000 particles at 30 fps.

Smoothed Particle Hydrodynamics is a Lagrangian solution to the Navier-Stokes fluid equations. Simulation consists of three basic steps: 1) Computing density and pressure of all particles. 2) Computing forces on particles, 3) Advancing the simulation by integration. The essence of the SPH technique is that density, pressure, and force are computed by considering weighted contributions from neighboring particles.

A purely naïve implementation of density in step #1 requires $O(n^2)$ calculations. By inserting particles into a grid a particle need only check neighboring grid cells for contributing particles within a given radius, called the smoothing radius r. We notice that following Muller, all publications we studied use a grid size equal to r, requiring that each particle check 27 grid cells (3x3x3). However, if a grid size of $2r$ is used, then only 8 grid cells (2x2x2) must be checked.

Other efficiency gains in the SPH algorithm were found through basic programming: eliminating variables, manually in-lining vector algebra. Taking a pure programming perspective, the SPH steps are essentially doubly-nested loops. Thus we improve efficiency in FLUIDS v.1 by reducing the inner loop for force computation to just 11 lines of C.

4. CONCLUSIONS

It is well known that choice of algorithms has the most significant effect on scalability. However, for a given algorithm the details of implementation can still influence efficiency by orders of magnitude. This constant factor is often a make-or-break effect in real-time applications. More importantly, in the development of real-time fluid simulation, poor metrics for reporting academic results have possibly masked a steady decline in algorithm efficiency behind rapid advances in hardware performance. This situation may be improved in the future by making clear distinctions between performance and efficiency. FLUIDS v.1 demonstrates a stable, fast, open source implementation of Smoothed Particle Hydrodynamics with a measured efficiency at least twice that of other implementations. Current goals include optimization and measurement of a GPU version of FLUIDS.

5. REFERENCES

[1] T. Amada. Real-time animation of water. Retrieved: http://chihara.aist-nara.ac.jp/people/2003/takasi-a, 2003.

[2] P. Horvath and D. Illes. Sph-based fluid simulation for special effects. In *Central European Seminar on Computer Graphics. Conference, April 2007.*, 2007.

[3] S. Kontar. Real-time fluid simulation. Master's thesis, Brno Univ. of Technology, Brno, Czech Republic, 2004.

[4] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH symposium on Computer animation*, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[5] NVIDIA. Particle demo from CUDA SDK 10, 2008.

[6] S. K. T. Harada and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *The Visual Computer*, Petropolis, Brazil, 2007. Springer. From proceedings of Computer Graphics International.

[7] S. K. T. Harada, M. Tanaka and Y. Kawaguchi. Real-time particle-based simulation on GPUs. In *ACM SIGGRAPH 2007 posters*, page 52, New York, NY, USA, 2007. ACM Press.

[8] Y. Zhang, B. Solenthaler, and R. Pajarola. GPU accelerated SPH particle simulation and rendering. In *ACM SIGGRAPH 2007 posters*, page 9, New York, NY, USA, 2007. ACM Press.

Client and Server Verification for Web Services Using Interface Grammars*

Graham Hughes Tevfik Bultan Muath Alkhalaf
Computer Science Department
University of California
Santa Barbara, CA 93106, USA
{graham,bultan,muath}@cs.ucsb.edu

ABSTRACT

Web services provide a promising framework for developing interoperable software components that interact with each other across organizational boundaries. For this framework to be successful, the client and the server for a service have to interact with each other based on the published service interface specification. If either the client or the server deviate from the interface specification, the client-server interaction will lead to errors. We present a framework for checking interface conformance for web services. Given an interface specification, we automatically generate web service server stubs (for client verification) and drivers (for server verification) and then use these stubs and drivers to check the conformance of the client and server to the interface specification. We implemented this framework by using interface grammars as the interface specification language. We developed an interface compiler that automatically generates a stub or a driver from a given interface grammar. We conducted a case study by applying these techniques to the Amazon E-Commerce Service.

1. INTRODUCTION

In providing a framework that enables web accessible software applications to interact with each other through the Internet, Web services are providing a promising next step in the evolution of electronic commerce. A crucial attribute of the web services framework is interoperability; software components from independent organizations should be able to talk to each other. This is achieved through interface specifications. Thus, conformance to these interface specifications becomes very important if this attribute is to be upheld. If either the client or the server deviate from the interface specification, the client-server interaction will lead to errors. Unfortunately it may not be easy to test the client and server together since they may not belong to the same organization.

*This work is supported by NSF grants CCF-0614002 and CCF-0716095.

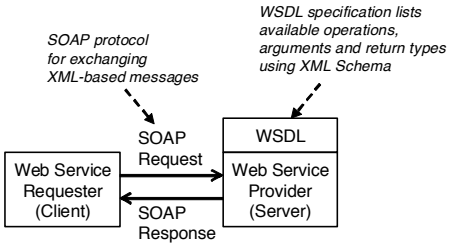


Figure 1: Basic architecture for web services

We present a framework that addresses this problem. Our basic idea is, given an interface specification, to write interface grammars for the specification; from those automatically generate web service stubs (for client verification) and drivers (for server verification); and finally use these stubs and drivers to check the conformance of the client and the server to the interface specification. We have developed a tool that automatically generates stubs and drivers from a given interface grammar. We target web services using SOAP [3] and WSDL [4] to communicate; their basic architecture is shown in Figure 1.

2. INTERFACE GRAMMARS

In previous work [1, 2], we have proposed *interface grammars* as a new language for the specification of component interfaces. The core of an interface grammar is a set of production rules that specifies all acceptable method call sequences for the given component. Given an interface specification for a component, our interface compiler generates a stub for that component. This stub is a table-driven top-down parser that parses the sequence of incoming method calls (i.e., the method invocations) based on the interface grammar defined by the interface specification.

We extend the usual notion of a grammar in a few important ways to achieve this. We permit nonterminals to have data parameters, which use call-by-value-return semantics. This permits us to model ephemeral data, like the unique identifier of an object, in our grammar as well as generate recursive data structures. We permit semantic predicates, which are blocks of code that must evaluate to true for a production to be available. Finally we permit semantic actions, which may execute arbitrary Java code but do not influence the parse.

```
stack ← [start, ⊥]
while stack ≠ [⊥] do
    o ‖ stack ← stack
    if o = nt(v1, ..., vn) then                    ▷ nonterminal
        Choose a production P of o;
        a = ⟨⟨bind o's arguments to v1, ..., vn⟩⟩;
        stack ← [a] ‖ P ‖ stack
    else if o = [[p]] ∧ ¬p then                    ▷ semantic predicate
        fail
    else if o = ⟨⟨a⟩⟩ then                         ▷ semantic action
        Perform a
    else if o = ?m(v1, ..., vn) then
        Receive the incoming method call m;
        Bind m's arguments to v1, ..., vn
    else if o = ¿m(v1, ..., vn) then
        Return from m with the values of v1, ..., vn
    else if o = !m(v1, ..., vn) then
        Call m with the values of v1, ..., vn
    else if o = ¡m(v1, ..., vn) then
        m returns;
        Bind m's return values to v1, ..., vn
```

Figure 2: Generic stub/driver pseudocode generated by the interface compiler

For the purposes of the web service verification, the method calls in the interface grammar represent the web service operations used. A detailed description of the interface grammar semantics can be found in [2, 1].

3. INTERFACE GRAMMAR COMPILER

To make these grammars useful, we must generate an actual web service driver and a web service stub in order to perform client and server verification. We achieve this using our interface compiler. Our compiler generates a web service stub when given a specification for a web service server, and a web service driver when given an interface for a web service client. A web service stub generated by our interface compiler is a top-down parser that parses the sequence of incoming SOAP requests. A web service driver generated by our interface compiler is a language generator which generates a sequence of SOAP requests based on the interface grammar.

In Figure 2 we show the structure of a generic stub/driver generated by our interface compiler. Here the semantics of **choose** and **fail** depend on the environment we are running in. In a conventional JVM, we could have **choose** randomly select one of the options and **fail** throw an exception. When running in a model checker, **choose** and **fail** hook into the model checker's internal backtracking to exhaustively explore the entire state space.

In order to make the generated stub/driver more efficient, we pre-compute some information that might be useful in choosing the next production. The details of this optimization are discussed in our earlier work [1], but briefly we use a modified LL(1) table parser whenever possible.

4. EXPERIMENTS

To demonstrate the value of our approach, we have performed two distinct classes of experimentation on the Amazon E-Commerce Service (AWS-ECS). The first, the client verification we detail here, involves verification of a demonstration client for the AWS-ECS. We generate a stub for the SOAP communication layer so that we can verify the client without connecting to the AWS-ECS and without any network communication. The second, the server verification, involves connecting directly to AWS-ECS itself and checking the AWS-ECS implementation.

We performed our experiments on a demonstration of programming technique called the AWS-ECS Java Sample, supplied by Amazon. This client performs no validation on its input whatsoever; it is solely intended as an example for how to use the interfaces. We use it to demonstrate the bug finding capabilities of our approach. The client makes two major classes of errors: 1) input errors that the client ought to be catching but doesn't, and 2) control flow errors that represent execution sequences that are locally valid but globally wrong. An example of the former is passing a string when AWS-ECS expects an integer; an example of the latter is trying to modify contents of the cart after the cart has been cleared. We will present data that demonstrates detection of multiple examples of both classes of error in a handful of minutes by running the client under the JPF model checker using our framework.

As well as verifying the client, we want to verify the server implementation. Our framework achieves this as well, by using an interface specification to generate SOAP requests, which are then sent to the web service. This is, in essence, an on-the-fly sentence generator for the interface grammar; we generate legal sequences of requests and verify that the server behaves in the expected manner. We have measured and will present data that demonstrates the effectiveness of different metrics for coverage and different sentence generation algorithms. Our experiments here discovered some omissions in the AWS-ECS API documentation, validating our approach.

5. SUMMARY

We have proposed and implemented a framework for conducting modular verification of web services based on interface grammars. We use interface grammars to specify the interfaces of web services. Using our interface compiler, these interface grammars are automatically converted to web service stubs/drivers to enable modular verification. We applied these techniques to a client for the key interfaces of the Amazon E-Commerce Service and also to the Amazon E-Commerce Service server directly, and have demonstrated that our approach is feasible and efficient.

6. REFERENCES

[1] G. Hughes and T. Bultan. Extended interface grammars for automated stub generation. In *Proceedings of the Automated Formal Methods Workshop (AFM 2007)*, 2007.

[2] G. Hughes and T. Bultan. Interface grammars for modular software model checking. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '07)*, pages 39–49, 2007.

[3] Simple object access protocol (soap) 1.1. W3C Note 08, `http://www.w3.org/TR/SOAP/`, May 2000.

[4] Web services description language (WSDL) 1.1. http://www.w3.org/TR/wsdl.

Accelerating Stochastic Simulation Algorithm for Chemically Reacting Systems on the Graphics Processing Unit *

Hong Li
hongli@cs.ucsb.edu

Linda Petzold
petzold@cs.ucsb.edu

ABSTRACT

Traditional deterministic approaches for simulation of chemically reacting systems fail to capture the randomness inherent in such systems at scales common in intracellular biochemical processes. The Stochastic Simulation Algorithm (SSA) then has been proposed. Many realizations are required to capture accurate statistical information of the solution. This carries a very high computational cost. The current generation of graphics processing units (GPU) is well-suited to this task. We have been accelerating the SSA with the NVIDIA GeForce 8800 GTX and get about 200 times performance improvement which illustrates the power of this technology for this important and challenging class of problems.

1. INTRODUCTION

Although the traditional deterministic approaches are sufficient for most systems, they fail to capture the natural stochasticity in some biochemical systems formed by living cells [2, 3], in which the small population of a few critical reactant species can cause the behavior of the system to be discrete and stochastic. The dynamics of those systems can be simulated accurately using the stochastic simulation algorithm (SSA) of Gillespie [2, 3]. For many realistic biochemical systems the computational cost of simulation by the SSA can be very high. Often, the SSA is used to generate large (typically ten thousand to a million) ensembles of stochastic realizations to approximate probability density functions of species populations or other output variables. Thus the computation can become intractable. Here we introduce an efficient parallelization of ensembles of SSA simulations for chemically reacting systems on the low cost graphics processing unit (GPU) NVIDIA GeForce 8800GTX [1].

*This work was supported in part by the U.S. Department of Energy under DOE award No. DE-FG02-04ER25621, by the National Science Foundation under NSF awards CCF-0428912, CTS-0205584, CCF-326576, and by the Institute for Collaborative Biotechnologies through grant DAAD19-03-D004 from the U.S. Army Research Office.

2. USING THE GRAPHICS PROCESSOR UNIT AS A DATA PARALLEL COMPUTING DEVICE

The Graphics Processing Unit (GPU) is a dedicated graphics card for personal computers, workstations or video game consoles. Recently, GPUs with parallel programming capacities have become available. The GPU has a highly parallel structure with high memory bandwidth and more transistors devoted to data processing than to data caching and flow control (compared with a CPU architecture)[1]. The GPU architecture is most effective for problems that can be implemented with stream processing and using limited memory. Single Instruction Multiple Data (SIMD), which involves a large number of totally independent records being processed by the same sequence of operations simultaneously, is an ideal general purpose graphics processing unit (GPGPU) application.

The Compute Unified Device Architecture (CUDA) Software Development Kit (SDK), supported by the NVIDIA Geforce 8 Series, supplies general purpose functionality for non-graphics applications to use the processors inside the GPU. We have been worked on the NVIDIA 8800 GTX chip which has 128 stream processors on a GeForce 8800 GTX chip, divided into 16 clusters of multiprocessors as shown in Figure 1 [1]. Each multiprocessor has 16 KB shared memory which brings data closer to the ALU. The processors are clocked at 1.35 GHz with dual processing of scalar operations supported. The maximum observed bandwidth between system and device memory is about 2GB/second.

3. PARALLELISM ACROSS THE SIMULATIONS

Our focus is on computation of ensembles of SSA realizations. Ensembles of SSA runs for chemically reacting systems are very well-suited for implementation on the GPU through the CUDA. The simulation code can be put into a single kernel running in parallel on a large set of system stat vectors $X(t)$. The large set of final stat vectors $X(t_{final})$ will contain the desired results.

The initial conditions $X(0)$ and the stoichiometric matrix ν originally will be in the host memory. We must copy them

[1]At the time of this writing, the NVIDIA GeForce 8800GTX costs about $200.

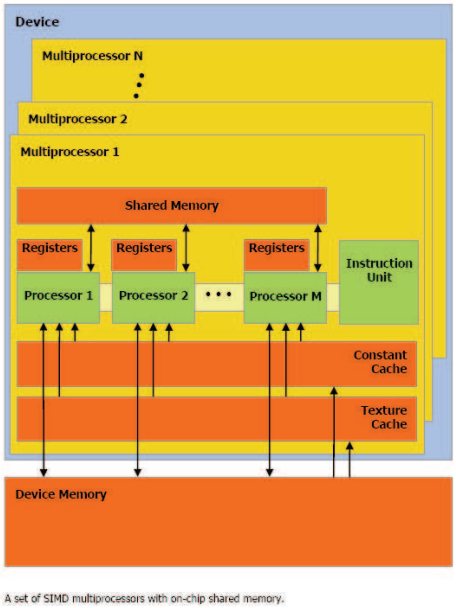


Figure 1: Hardware Model

to the device memory by CUDAMemcpy in the driver running on the the CPU. We minimize the transfer between the host and device by using an intermediate data structure on the device and batch a few small transfers into a big transfer to reduce the overhead for each transfer. Next, we need to consider the relatively large global memory vs. the limited-size shared memory. The global memory adjacent to the GPU chip has higher latency and lower bandwidth than the on-chip shared memory. It takes about 400-600 clock cycle latency to access the global memory vs. 4 clock cycles to read or write the shared memory. To effectively use the GPU, our simulation makes as much use of on-chip shared memory as possible. We load $X(0)$ and the stoichiometric matrix ν from the device memory to the shared memory at the very beginning of the kernel, process the data (propensity calculation, state vector update, etc.) in shared memory, and write the result back to the device memory at the end. Because the same instruction sequence is executed for each data set, there is a low requirement for flow control. This matches the GPU's architecture. The instruction sequence is performed on a large number of data sets which do not need to swap out, hence the memory access latency is negligible compared with the arithmetic calculation.

Given the total number of realizations of SSA to be simulated, the number of threads per block and the number of blocks must be carefully balanced to maximize the utilization of computation resources. For stochastic simulation, we can't use too many threads per block since there is only a limited shared memory and all system state vectors and propensities have been put in shared memory for efficient frequent access. Thus the number P of threads per block should satisfy $(N + M) * 4 * P + \alpha < 16K$, where N is the number of chemical species, M is the number of reactions, 4 is the size (in bytes) of an integer/float variable, $16K$ is the maximum shared memory we can use within one block, and α is the shared memory used by the random number generator (this is relatively small).

Statistical results can only be relied on if the independence of the random number samples can be guaranteed. Thus generating independent sequence of random numbers is one of the important issues of implementing simulation for ensembles of stochastic simulation algorithms in parallel. We chose the Mersenne Twister [5] from the literature in our application [6]. This method has passed many statistical randomness tests including the stringent Diehard tests [6]. The fully tested MT random number generator can efficiently generate high quality, long period random sequences with high order of dimensional equidistribution. Another good property of the MT is its efficient use of memory. Thus it is very suitable for our application.

4. CONCLUSIONS

The SSA is the workhorse algorithm for discrete stochastic simulation in systems biology. Even the most efficient implementations of the SSA can be very time-consuming. Often the SSA is used to generate ensembles (typically ten thousand to a million) of stochastic simulations. The current generation of GPUs appears to be very well-suited for this purpose. On the two model problems we tested, we observed speedups about 200 times for the GPU, over the time to compute on the host workstation. With this impressive performance improvement, in one day we can generate data which would require more than six months of computation with the sequential code. The details of the work can be found in [4].

This technology is not quite ready for the novice user. Programs must be written to be memory efficient, with the GPU architecture in mind. The computation is limited to single precision, the API does not yet have all the features to take full advantage of the architecture, transferring data to the GPU is time consuming, the CPU and GPU cannot compute simultaneously, and there are a number of problems and limitations that NVIDIA is still working on.

5. REFERENCES

[1] N. Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. http://developer.download.nvidia.com.

[2] D. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.

[3] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[4] H. Li and L. Petzold. Stochastic simulation of biochemical systems on the graphics processing unit. 2007. Submitted.

[5] M. Matsumoto and T. Nishimura. Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator . *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8:3–30, 1998.

[6] N. F. members. NVIDIA forums. http://forums.nvidia.com.

Multimodal Photo Annotation and Retrieval on a Mobile Phone

Xavier Anguera Miro
Telefónica Investigación y Desarrollo

JieJun Xu
Computer Science
University of California, Santa Barbara

Nuria Oliver Ramirez
Telefónica Investigación y Desarrollo

Mobile phones are becoming multimedia devices. It is common to observe users capturing photos and videos on a regular basis. As the amount of digital multimedia content expands, it becomes increasingly difficult to find specific images in the mobile device. Previous studies on this area mostly focus on mono-modal input query (text, audio or image), which do not take full advantage of the multimodal information stored with the images [3, 4].

We present a multimodal and mobile image retrieval prototype named MAMI (Multi-modal Automatic Mobile Indexing) [1, 2]. It allows users to annotate, index and search for digital photos on their phones via speech or image input. A key advantage of MAMI is that it is implemented as a stand-alone application which runs in real-time on the phone. Therefore, users can search for photos in their personal archives without the need of connectivity to a server. Two different multimodal approaches are used for image retrieval. The first approach consists of a set of fusion techniques, and the second approach is a novel algorithm called multimodal redundancy reduction (MR2 algorithm).

1. Multimodal Processing

Acoustic and image features are the two major modalities used in the system. Mel Frequency Cepstrum Coefficients (MFCC) and Edge Histogram Descriptor (EHD) of MEPG-7 are used to describe the audio and visual feature associated with an image respectively. In addition, Dynamic Time Warping (DTW) algorithms and standard Euclidean distance are used for similarity measure in the audio and visual space.

2. Image Retrieval Approach 1 – Multimodal fusion

Multimodal fusion typically works at the feature or at the decision levels. We have considered fusion techniques at the decision level as they allow the use of different algorithms for characterizing and comparing the features vectors in each of the modalities. In particular, we have opted for the weighed-sum (linear combination) approach, due to its simplicity and reasonable level of tolerance to noise in the input data. Three alternatives of the weighed-sum approaches are implemented in our system, which include empirical weight determination and automatic weight selection via variance-mean normalization for both Gaussian and non-Gaussian metric space. A set of experiments to search for the closest picture given a multimodal query (audio and visual) were carried out by six users. Experimental results show that multimodal fusion significantly outperforms mono-modal in terms of image retrieval accuracies.

3. Image Retrieval Approach 2 – MR2 algorithm

A novel approach called Multimodal Redundant Reduction (MR2) algorithm is proposed here to maximize the probability a desired picture will appear in the top-N images. Note that redundant images are very common in personal image databases, because users typically take more than one picture in the same scene (and context) to ensure the desired content has been captured. However in terms of image search in mobile phone, we want to reduce the number of redundant images and cover a broader range of images in the small screen. The basic idea is to exploit the information contained in the modality *not used* in the input query (e.g. visual information for audio queries and vice versa) to avoid showing redundant images. The algorithm is shown as following:

Input: a given query Q (either acoustic or image)
while $n <= N$ do
  Retrieve the next closest image Xn to Q (either in acoustic or visual space)
  if $n = 1$ then
    Set $X1$ as the first output.
  else
    Compute the closest K images ($Ym=1 \ldots K$) to Xn given the modality *not*
    used in query Q
    if ($Ym=1 \ldots K$) contains any images in $X1 \ldots n-1$
      discard Xn
    else
      include Xn to the output and $n = n + 1$
    end if
  end if
end while
Output: list of N most relevant pictures shown on screen

(a) (b)

Figure 1. Screen shot of the system with (a) and without (b) the use of MR2 algorithm. Note that the top two 'bench' images in (a) are consider redundant and thus collasped into one representative image in (b)

4. Overall Results

The multimodal experiments were performed in a per speaker basis, given that the MAMI prototype is intended to be a personal system. For each picture in each of the participant's databases, we searched for the best (or *N*-best depending on the test performed) set of pictures with smallest distance to the input image, either in the image or acoustic space. A metric of accuracy is reported as the % of times that the correct picture - *i.e* .the picture comes from the same context as the desired one was retrieved.

Query	MonoModal	MR2	Fusion
Audio	94.48%	95.04%	95.63%
Visual	83.87%	84.92%	

Table 1. Overall results show both MR2 and Fusion techniques outperform MonoModal methods.

5. Conclusion

In this work, we have demostrated a mobile image annotation, indexing and retrieval prototype, which makes use of multiple image modalities efficiently. We believe that multimodal approaches will continue to play an important role in the years to come.

6. Reference

[1] X. Anguera and N. Oliver. MAMI: Multimodal annotations on a mobile phone. In Proceed. of Intl. Conf. on Mobile HCI (MobileHCI-08), 2008.

[2] X. Anguera, N. Oliver, and M. Cherubini. Multimodal and mobile personal image retrieval: A user study. In Proceed. of SIGIR Workshop on Mobile Information Retrieval (MobIR'08), 2008.

[3] C. Gurrin, G. J. F. Jones, H. Lee, N. O'Hare, A. F. Smeaton, and N. Murphy. Mobile access to personal digital photograph archives. In Proc. of MobileHCI '05: Proc. of the 7th int. conf. on HCI

[4] A. Hwang, S. Ahern, S. King, M. Naaman, R. Nair, and J. Yang. Zurfer: mobile multimedia access in spatial, social and topical context. In MULTIMEDIA '07: Proc. of the 15th int. conf. on Multimedia

MeshMon: A Multi-tiered Framework for Wireless Mesh Network Monitoring

Ramya Raghavendra, Prashanth A. K. Acharya, Elizabeth M. Belding, Kevin C. Almeroth
{ramya, acharya, ebelding, almeroth}@cs.ucsb.edu

Large scale IEEE 802.11 mesh networks promise to be a significant method of providing Internet connectivity in several cities and towns. In addition to these metro-scale deployments, wireless mesh networks (WMN) have been proposed to provide connectivity in rural environments, especially in developing countries around the world. Such large scale mesh networks consist of hundreds to thousands of mesh routers and may be used by thousands of users. The presence of numerous wireless devices, including mesh routers and client devices, in a single administrative domain increases the complexity of the difficult task of managing these large scale mesh networks.

We believe the network administrator's ability to manage and troubleshoot these networks in real-time is a critical factor that contributes to the success of WMNs. These administrative tasks, however, present several new challenges compared to traditional wireline networks. In particular, the design of a network monitoring system is non-trivial because of the multi-hop architecture of these mesh networks and the inherent wireless-related properties of 802.11-based devices. For instance, the performance of the devices in these networks may be impacted by entities outside the network, i.e. the surrounding environment or devices that are not part of the network but share the frequency spectrum.

In addition, the large number of proprietary protocols and algorithms used by different IEEE 802.11 client vendors and the interaction among these clients is not well understood. Unlike in WLANs, the backhaul links used for communication between mesh routers and the Internet Gateway consist of relatively low bandwidth multi-hop wireless links. Therefore, control traffic required for remote monitoring and administration of these mesh routers must be minimal, so as not to consume a significant portion of the available bandwidth. The unreliable nature of wireless links may result in gaps in the collected data, preventing the timely measurement analysis. Finally, unlike wired networks, the physical location of the mesh routers provides a strong spatial aspect to all data used in management and troubleshooting of mesh networks. Therefore, data from different routers that share spectrum in a geographical region may need to be analyzed in correlation with each other.

Although traditional infrastructure WLANs present similar monitoring challenges and requirements, network monitoring solutions developed for WLANs cannot be directly applied to WMNs. Most monitoring solutions for commercial WLANs only use a small fixed subset of the large set of available metrics to minimize the data collection and processing overhead. This approach may fail to capture data needed to diagnose a detected problem. Previous research has shown that the diagnosis and root cause analysis of many network faults requires a complete trace of the packets in the network [1, 2]. Unfortunately, the capture and remote analysis of all data packets is infeasible in a mesh network as the bandwidth requirements are prohibitive. Further, monitoring systems that use a large set of metrics (or detailed packet traces) require resource intensive computation and thus may be unsuitable for real-time identification and remediation of problems. From our own experience in the development of a real-time network visualization tool, we found that the speed of metric collection/generation, rather than visual rendering of the data, is the computational bottleneck [3].

For the above reasons, there is a need for a methodology of monitoring and metric collection in WMNs that is bandwidth-efficient, scalable with respect to the number of devices in the network, and able to provide a comprehensive set of metrics that can be used to identify all problems in the network. Such a solution would facilitate centralized administration of a large network and also enable the use of tools, such as network visualization, to monitor the network health in real-time.

In this paper, we present MeshMon, a network monitoring framework that enables real-time identification and troubleshooting of problems in WMNs. A key observation that guides the design of MeshMon is that comprehensive metric collection is required only when there are problems in the network. A small subset of these metrics, called baseline metrics, are sufficient when the network performance is satisfactory, and can be used for coarse identification of potential problems. We propose a stateful method that intelligently adapts the metric collection process to capture the most relevant set of metrics. When the baseline metrics indicate the possible presence of a problem, the system transitions to collect a more detailed set of metrics. The goal of this methodology of metric collection is to reduce the volume of data that needs to be collected and processed without sacrificing the ability to diagnose problems in the net-

Trace	CBR	Replay
Faults Injected	30	30
Faults Detected	27	25
False Positives	8	10
Overhead Reduction	68%	64%

Table 1: Fault diagnosis performance of MeshMon.

work.

In this work we develop the idea dynamic and scalable hierarchical metric collectionin the context of mesh networks. Mesh networks offer additional complexity as compared to WLANs because a monitoring system should address problems that affect mesh routers as well as those that affect client devices. Therefore, MeshMon incorporates metrics associated with mesh routing and connectivity into the hierarchical metric collection, in addition to metrics associated with client devices. Our design ensures that even in situations where a problem scenario is reflected in both sets of metrics (mesh related and client access related), MeshMon can successfully isolate the root cause of the problem.

The system is evaluated by injecting faults into the network and comparing the number of faults detected with the number injected.

A prototype of the MeshMon system has been implemented on the UCSB Meshnet. The implementation involves simple extensions to the madwifi driver as well as software at the user level.

Evaluations are conducted with two types of traffic: a) constant rate flows which we call the CBR traffic, and b) traces from a large WLAN, which we call the Replay traffic. In each of the scenarios, eight laptops act as clients connected to the UCSB mesh network. In the first scenario, each laptop sends CBR traffic at a constant rate of 1Mbps to the gateway. In the second scenario, we use the WLAN traces collected from the IETF 67 wireless network to extract link layer data traffic patterns and use this information to replay the traffic on the mesh testbed [4]. [1]

Our general evaluation methodology is as follows. We inject a set of faults into the system. The nodes run MeshMon and attempt to diagnose the faults through increased metric collection and send alerts to the central controller when the fault is detected. We quantify the diagnosis accuracy by comparing the inferred fault and its source with the original fault we injected. We inject faults in both the client access layer and the mesh layer.

Fault diagnosis accuracy and overhead reduction: The complete set of results from the experiments is presented in Table 1. Of the total 60 faults injected in the two scenarios, 52 were successfully detected by MeshMon. The average reduction in overhead for the two scenarios was 66%. In other words, MeshMon was able to detect a high percentage (86.6%) of faults using only one-third of the monitoring

[1]A detailed description of both scenarios is in a full version of this extended abstract.

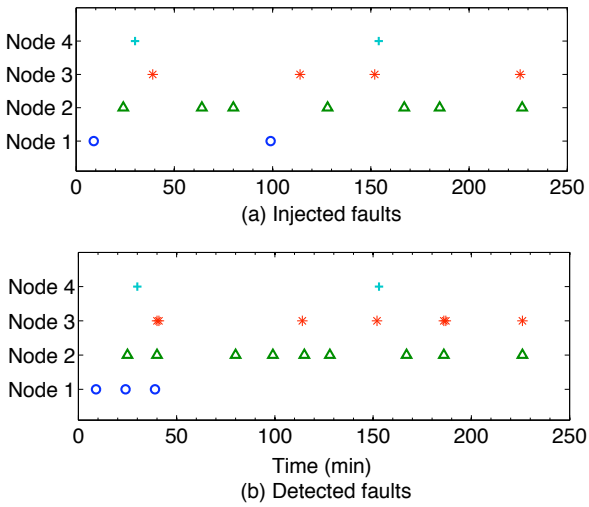


Figure 1: Time series of events (faults injected and faults detected) at the mesh layer in a representative experiment trial.

bandwidth as compared to the simple approach of using all the available metrics. For our simple testbed setup with 15 nodes and a maximum of one client per mesh node, the simple monitoring approach collected about 400MB of monitoring data for a four hour period, while MeshMon required about 134MB. This is an encouraging result that indicates that MeshMon can scale better and can support larger mesh networks.

The results in Table 1 indicate a high number of false positives and hence we further investigate this behavior. We observe that for some injected faults, the central controller receives alerts from multiple mesh routers. MeshMon currently does not have the capability of correlating alerts posted by multiple mesh routers. Such a capability would enable MeshMon to distinguish a fault that simultaneously impacts the performance of multiple mesh routers and reduce the misleading false positive rate.

1. REFERENCES

[1] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis," in *Proc. of SIGCOMM*, Pisa, Italy, Sep. 2006.

[2] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benkö, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker, "Automating Cross-Layer Diagnosis of Enterprise Wireless Networks," in *Proc. of SIGCOMM*, Kyoto, Japan, Aug. 2007.

[3] A. Jardosh, P. Suwannatat, T. Hollerer, E. Belding, and K. Almeroth, "SCUBA: Focus and Context for Real-time Mesh Network Health Diagnosis," in *Proc. of PAM*, Cleveland, OH, Apr. 2008.

[4] R. Raghavendra, E. M. Belding, K. Papagiannaki, and K. C. Almeroth, "Understanding Handoffs in Large IEEE 802.11 Wireless Networks," in *Proc. of IMC*, San Diego, CA, Oct. 2007.

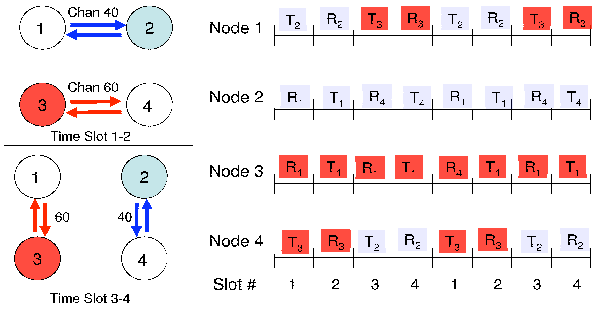
FreeMAC: Implementing a Multi-Channel TDMA MAC on 802.11 Hardware

Ashish Sharma, Elizabeth M. Belding
Department of Computer Science
University of California, Santa Barbara CA 93106
{asharma, ebelding}@cs.ucsb.edu

Modern wireless devices offer increased software control over radio communication parameters. In the future, the trend of making the hardware more programmable is expected to grow due to the emergence of cognitive and software defined radio networking. Since a large portion of the MAC protocol is implemented in software, with the firmware providing a set of functional primitives, it is possible to design and implement alternate MAC protocols in real testbeds equipped with commodity 802.11 devices [1, 2, 3]. Validation of protocols on real testbeds helps characterize the additional system constraints that are difficult to capture in a simulated environment. This work demonstrates FreeMAC – a framework that enables the design and implementation of a general class of *multi-channel* MAC protocols, with strict timing requirements, on a typical Linux system.

Multi-channel MAC protocols aim to improve the capacity of a wireless network by time-multiplexing the operation of nodes on orthogonal channels. Generally such protocols rely on precise time scheduling of channel switch operations, usually every 20-80 ms to limit latency. TDMA based MAC protocols require that packets are transmitted during designated time slots. However, implementing such MAC protocols on a Linux system using commodity 802.11 hardware involves several systems challenges. The focus of the FreeMAC platform lies on enabling the primitives for the deployment of such MAC protocols on existing systems. FreeMAC allows tight control over several radio parameters using API functions. We also propose a novel approach of programming the *beacon interval* to invoke periodic hardware interrupts that can be used to bound the latency in scheduling of time-sensitive MAC functions using kernel timers.

We use the FreeMAC framework to implement a simple proof of concept multi-channel TDMA based MAC on a testbed of 4 laptops, each running kernel 2.6.15.7 on the Ubuntu Linux distribution. Each laptop is equipped with an AR5212 chipset-based LinkSys 802.11 a/b/g PCMCIA card. We use the OpenHAL port of MadWiFi - an open source driver for Atheros chipset-based commodity 802.11 wireless devices. In our testbed setup, as shown in Figure , we demonstrate the capability of the FreeMAC framework to implement both single-channel (for nodes 2 and 3) as well as multi-channel (for nodes 1 and 4) TDMA-style MAC protocols. The TDMA schedule of our testbed comprises of four 50ms timeslots (the slot duration is reconfigurable), operating on two orthogonal channels. Each node transmits in two of these slots and receives on the remaining two. We eliminate per-packet acknowledgements and reduce the contention period to a minimum to ensure a TDMA style MAC implementation.



By exporting the programmability available at the radio hardware as API functions, platforms like FreeMAC can pave the way for increased cross layer interactions and efficient network protocol designs. We plan to use the FreeMAC platform to design and validate dynamic TDMA-style multi-channel MAC protocols for deployment in rural mesh networks.

1. REFERENCES

[1] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMAC - Flexible Wireless Research Platform. In *HotNets'05*, College Park, Maryland, USA, Nov 2005.

[2] A. Sharma and E. M. Belding. FreeMAC: Framework for Multi-Channel MAC Development on 802.11 Hardware. In *PRESTO'08*, Seattle, WA, USA, Aug 2008.

[3] A. Sharma, M. Tiwari, and H. Zheng. MadMAC: Building a Reconfigurable Radio Testbed Using Commodity 802.11 Hardware. In *WSDR '06*, Reston, VA, USA, Sep 2006.

CoBRa: Content Based Ranking for Documents

Vishwakarma Singh*
Dept. of Computer Science
University of California
Santa Barbara
vsingh@cs.ucsb.edu

Sayan Ranu *
Dept. of Computer Science
University of California
Santa Barbara
sayan@cs.ucsb.edu

ABSTRACT

We propose a new self learning model to rank a set of related documents based on their content. We create a model for each document and learn a parent model. All the documents are ranked based on their similarity with the parent model. We use this to re-rank top-k results generated by a search engine for a *key word*. It dramatically improves the ranking of content rich pages with a very small computation time of 10 ms.

1. MOTIVATION

Key Word based web search engines are already a billion dollar industry. As the industry grows both in users and the data, there is also a natural demand to improve the quality of service. There has been a decade of research to improve search quality without compromising on service time. Research spans from improving *key word* search engines to developing context and natural language based search engines.

State of the art for the search engines is to rank pages by an aggregate score of **page rank** and **tf-idf**. Some engines like *ASK* exploit user clicks over a period a time to improve the quality. Web Link network, user clicks and frequency of the key word in web pages provide a good start to rank the pages but are only partially effective since they don't consider the semantic of a page. Learning the semantic of page from its content calls for complex algorithm which are still far from yielding satisfactory results.

Demand for fast search restricts usage of complex algorithm and processing of huge data online. Search algorithm based on the content of a page will require a model that represents its semantics and an efficient algorithm to rank the pages based on the model. Given a good model, getting practical online time will be difficult as the number of web pages containing a word ranges in thousands. An interesting observation from the web industry can be used to partially incorporate content based ranking in the existing systems and improve customer satisfaction. We observe that most of the web users only click through top 4 or 5 results for a key word. Therefore, it should be sufficient for most of the practical purposes that we just rerank top-k results returned by an existing search engine.

In this paper we develop a new model to capture semantics of a page and a new learning technique (CoBRa) to re-rank the top-k results returned by an existing search engine based on content. Our experiments reveal that CoBRa dramatically improves the rank of content rich pages with only a marginal increase in the computation cost in the existing systems. CoBRa is more suited to rank documents than web pages.

*Authors contributed equally to the work

2. PAGE MODEL

Semantic of a page or a document better determines its relationship to a given key word than just the occurrence of the word in it. A web page semantically related to some other subject may contain few occurrences of the key word as helper. Therefore, there is a need of a framework to effectively and efficiently capture semantics. Existing search engines store a single token (1-gram phrase) based histogram of a page as its description. This histogram assumes total independence between tokens and has no information of the structure.

We extend the existing histogram model (H) by including n-gram phrases which captures the semantics more powerfully than 1-gram phrases. A classic example is *data mining*. *data* and *mining* words can separately mean different things in different context but their co-occurrence defines a subject in the Computer Science. This extension not only fits the state of the art but is also cost effective. n-gram phrases are obtained using natural language processing packages like NLTK [3]. Occurrence of phrases of size more than 3 are rare. Therefore, we only include phrases of at most size 3 in a histogram. We do not consider different forms of the same word. After extracting the phrases, we use Porter Stemming algorithm [4] to extract the word stem and then build a histogram (H). We normalize the frequency of each phrase (t) by the total count of the phrase in the page.

$$p(t, H) = \frac{frequency(t, H)}{\sum_{1 \leq i \leq \|H\|} frequency(t_i, H)} \quad (1)$$

3. RANKING MODEL

In this section, we use the extended histogram (H) to develop a new ranking method (CoBRa) that self learns a model without any user input. Intuitively, we hypothesize that all the web pages (page models) related to a subject are derived from a parent model (M) that in someway captures the characteristic of each page. If we use M to randomly generate pages then the given set of web pages will be a subset of this random population. An example is English literature. Given an English language dictionary of n-gram phrases and its Grammar, every piece of English literature can be derived by some combination of these two entities. Our page model is representative of such a dictionary.

We aggregate information from the page models (H) to construct a weighted parent model (M). Parent model consists of all the distinct stemmed phrases found in any page model. Weight of a phrase in M is given by

$$w(t, M) = m_t \times \sigma_t \tag{2}$$

$$m_t = \frac{\sum_{0<i<K} p(t, H_i)}{K} \tag{3}$$

$$\sigma_t = \sqrt{\frac{\sum_{0<i<K} (p(t, H_i) - m_t)^2}{K - 1}} \tag{4}$$

This formulation gives high weight to the phrases which occurs uniformly across all the web pages with high probability. Phrases occurring with high probability in few pages or with very low probability in all pages will be weighted less.

We rank the pages by its similarity to the parent model. Parent model can be assumed to be an ideal page that best describes the key word semantically. Therefore, the web page having highest similarity to the parent model should be considered best result for the query. We measure the similarity as inverse of the distance between a page model (H) and the parent model (M). There are sophisticated methods in literature like EMD [5] and KL [2] divergence to measure distance between two distributions but are computationally intensive. Therefore, we use simple and efficient weighted L_1 distance between two models.

$$d(H, M) = \sum_{1 \le i \le \|H\|} \|w_{t_i} - p_{t_i}\| \tag{5}$$

$$dist(H, M) = \frac{\|M\|}{\|H\|} \times d(H, M) \tag{6}$$

$$score\ s = \frac{1}{dist(H, M)} \tag{7}$$

Page with the highest score is ranked first.

4. IMPROVING COBRA

We improve the ranking methodology (CoBRa) by weighing each phrase in the parent model by its score in the WORDNET [6] ontology. WORDNET scores a word against a query word based on its distance from it on a prebuilt ontology. This helps to capture the similarity of phrase to the query word and hence the semantic meaning. Let ws_t be the score of a phrase from wordnet. Distance between H and M is modified as

$$d(H, M) = \sum_{1 \le i \le \|H\|} \frac{\|w_{t_i} - p_{t_i}\|}{ws_t} \tag{8}$$

5. EXPERIMENTS

In this section we describe the experimental setup and give qualitative result. We use the top 15 results of google for a given *key word* to re-rank. We neither maintain a cache of web pages in our server nor their phrase histogram. Solely for the purpose of experimentation, we create page model at run time by fetching web pages from their respective servers. Histograms should be computed off line and stored for all practical purpose. We give a diagrammatic representation of the steps involved in the page model preparation in Figure 1. One of the major challenges is to extract only meaningful text from the pages. A web page contains query related text,

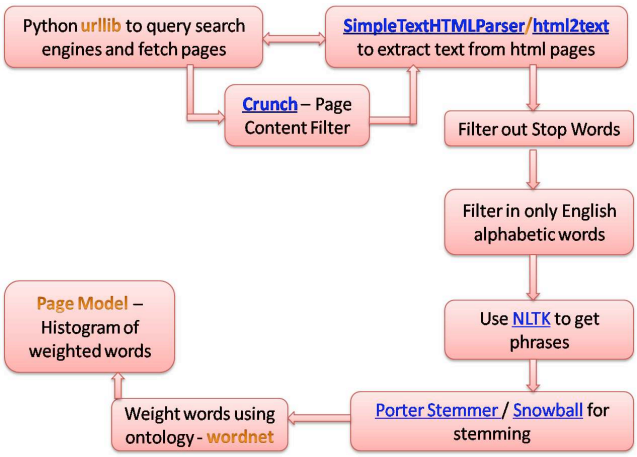


Figure 1: Steps for preparing Page Model

menus, ads, links, scripts, style sheets and images. Though menus, ads, link titles and image captions can provide important context sensitive information but in this paper we limit ourselves with only alphabetic text in a page. Once we have the page model for the top 15 pages, we execute CoBRa to rerank them.

Time analysis: COBRA gives an amazing performance to re-rank top-15 pages by content. On an average it takes 10ms to rerank the pages. A key word search takes time between few milliseconds to seconds. We can see that CoBRa adds delta cost to the existing search cost and can be easily integrated with these systems.

Qualitative Analysis: Quality is subjective to human need and thinking. Same page can be rated differently by different people for the same key word. So, there is no standard way to quantify page rankings. To validate our claim we present a pilot implementation at our server [1]. We also provide a collection of comparative results for some hand picked key words from various fields. Standard practice is to open the service to beta users and collect their feedback on the quality. Given the resource constraints, currently we are not able to pursue this methodology.

6. CONCLUSION

We develop an efficient methodology to rerank top-k results of a search engine by content. It requires minimal changes to the already existing techniques and costs an average time of 10 ms. It gives a remarkable improvement in the rank of the content rich pages.

7. REFERENCES

[1] http://128.111.44.206/cobra/.
[2] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, pages 79–86, 1951.
[3] http://nltk.org/index.php/Main_Page.
[4] http://tartarus.org/ martin/PorterStemmer/.
[5] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. *ICCV*, pages 59–66, 1998.
[6] http://wordnet.princeton.edu/.

Depth Compositing for Augmented Reality

Jonathan Ventura
Department of Computer Science
University of California, Santa Barbara
jventura@cs.ucsb.edu

Tobias Höllerer
Department of Computer Science
University of California, Santa Barbara
holl@cs.ucsb.edu

1. INTRODUCTION

Correct handling of occlusion is a significant challenge when compositing real and virtual content, whether it be for augmented reality or film. For film, the solution is often solved offline by arduously creating alpha mattes by hand. In the augmented reality context, the compositing must be real-time, so offline solutions are not possible. Occlusions are usually disregarded in augmented reality, so that virtual objects are always rendered on top of physical objects. If a highly accurate 3D model of the scene is available, then depth compositing can be performed automatically; however, such models can be difficult to create, and limit the range of the system to the extent of the model.

We have developed a method for automatic depth compositing which uses a stereo camera, without assuming static camera pose or constant illumination. The traditional approach to automatic depth compositing with a stereo camera uses the normal SAD block matching algorithm, and copies the disparity map into the z-buffer [7, 5]. These approaches result in disparity maps which have noise in texture-less regions and/or at edge boundaries. Berger's work [1] uses snakes to find the outlines of occluded objects. However, this approach has not been proven to be real-time or sufficiently robust.

Kolmogorov et al. describe the Layered Graph Cut algorithm to fuse color and stereo likelihood for binary segmentation [3]. They learn color and depth distributions for foreground and background from hand–labelled data, and then use these distributions in a graph cut which encourages consistent labeling within smooth regions. In our work we extend the Layered Graph Cut to general depth compositing by decoupling the color and depth distributions, so that the depth distribution is determined by the disparity map of the virtual scene to be composited in.

2. OUR APPROACH

The central assumption behind our approach is that the scene consists of distinct foreground and background which can be segmented using color and contrast cues alone. Only foreground objects may occlude virtual objects. Color segmentation is first used to separate foreground and background. Then, the depth distribution is used to determine visibility of foreground objects (or conversely, occlusion by virtual objects).

In our experiments, we determine a depth probability distribution for the real image by stereo analysis. The color segmentation is initialized by manual labeling of a single frame in the compositing sequence, and modeled using histograms. However, it is important to note that this compositing algorithm is independent of the particular methods used to acquire depth and perform color segmentation. Other possible methods for depth acquisition include time-of-flight sensing [2], coded aperture [4], or structured light [8]. Many more sophisticated approaches for color segmentation exist, such as background subtraction or adaptive mixture models [6].

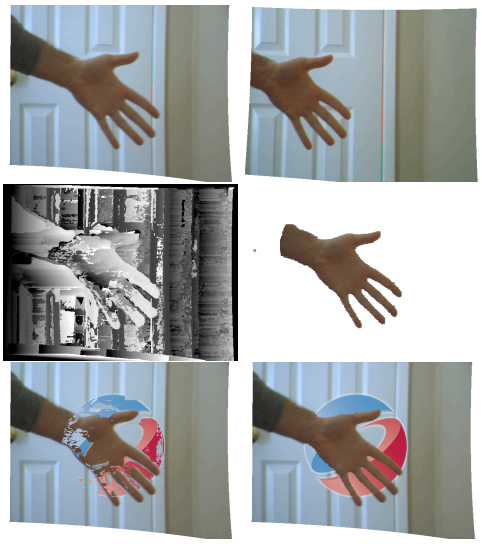


Figure 1: Depth compositing example (left to right, top to bottom): left and right stereo pair; disparity map and color segmentation; compositing based on disparity alone; compositing with graph cut.

3. METHOD

Ultimately, we wish to label each pixel as either background, visible foreground, or occluded foreground. The compositing algorithm consists of two steps. First, we use a color-based graph cut to separate foreground from background. Second, we use a depth-based graph cut to separate visible foreground pixels from those occluded by the virtual object.

The energy E to be minimized by the graph cut is defined as:

$$E(\mathbf{z};\mathbf{x}:\Theta) = U(\mathbf{z};\mathbf{x}:\Theta) + V(\mathbf{z};\mathbf{x}) \quad (1)$$

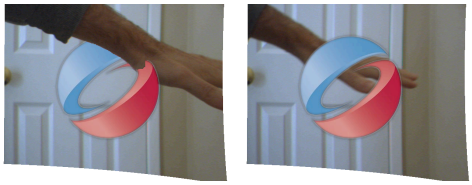


Figure 2: Partial occlusion (left) and full occlusion (right).

where $\mathbf{z}$ contains the color pixels in the left image, $\mathbf{x}$ is the labeling, and V is the usual contrast term [3]. For color segmentation, Θ contains the color distribution parameters and U is the color likelihood. For depth segmentation, Θ contains the match likelihoods (for all disparities) between the left and right image, and U aggregates the corresponding visibility likelihood.

Figure 3 gives an example of the compositing process. The top left image shows the color-based segmentation into foreground and background. The stereo pair is then analyzed to determine a probability distribution over the disparity space for each pixel; the top right image gives the maximum likelihood disparity map. A second graph cut determines visibility of foreground pixels based on the stereo analysis, shown bottom left. Finally, the real and virtual content is combined in the bottom right image.

4. RESULTS AND EVALUATION

We take as our example an augmented reality scenario where the user is interacting with virtual content using his/her hands. The hands are the foreground objects, and the rest of the scene is the background. The color distributions were modeled using two-dimensional histograms. (We use the Cb and Cr channels of YCbCr color space, which is convenient for skin segmentation because skin does not contain much green.) The logo is composited in at constant depth.

As Figure 1 shows, the graph cut achieves a clean composite of a hand with a virtual object, as opposed to the composite using a stereo depth map, which contains significant noise. We improve over solely using the stereo map in two ways: first, we use color segmentation to avoid putting background in front of the virtual content; and second, we use a boundary term which reduces noisy labeling in smooth regions such as the textureless areas of the hand. This composition technique enables bare hand interaction with virtual content with correct visual occlusion.

Figures 2 and 3 give examples of both partial occlusion using this compositing algorithm. With color segmentation alone, we have no information about visibility of foreground pixels. By using depth cues, however, the algorithm can accurately determine visibility. These results show that our algorithm can successfully handle all types of occlusions (including partial occlusions) by fusing color and depth.

Performance is two frames per second on a 2 Ghz machine. An easy speedup is possible by parallelizing the stereo analysis. The stereo likelihoods can be computed at each disparity independently. This suggests performance gains either using multiple CPUs or a GPU implementation.

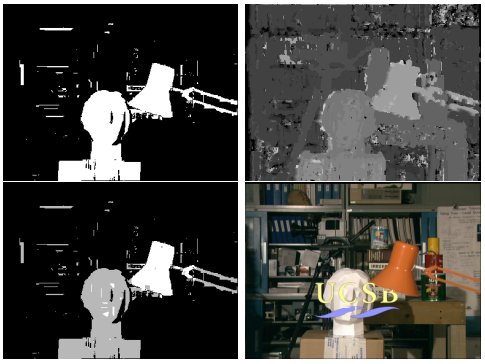


Figure 3: Depth compositing example with standard Tsukuba stereo pair (left to right, top to bottom): foreground / background segmentation based on color; maximum likelihood disparity map; visibility segmentation based on disparity likelihood; compositing result.

Acknowledgments

This work was supported by the NSF IGERT program in Interactive Digital Multimedia, NSF Grant No. 0221713.

5. REFERENCES

[1] M. O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *CVPR '97: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[2] R. Gvili, A. Kaplan, E. Ofek, and G. Yahav. Depth keying. In *Stereoscopic Displays and Virtual Reality Systems X*. SPIE, 2003.

[3] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[4] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. In *Proceedings of ACM SIGGRAPH 2007*, 2007.

[5] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, 2002.

[6] C. Stauffer and W. Grimson. Adaptive background mixure models for real-time tracking. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, 1998.

[7] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D graphics*, 1995.

[8] L. Zhang, B. Curless, and S. M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *CVPR '03: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

Strategy-Proof Wireless Spectrum Auctions

Xia Zhou, Sorabh Gandhi, Subhash Suri, and Haitao Zheng
Department of Computer Science
University of California, Santa Barbara
{xiazhou, sorabh, suri, htzheng}@cs.ucsb.edu

1. INTRODUCTION

An increasing number of users, homes and enterprises rely on wireless technology for their daily activities. However, the growth of wireless networks has been fundamentally limited by the inefficient distribution of radio spectrum. Historical static allocations have led to an artificial shortage of spectrum. This misallocation has prompted a wide-spread interest in an open, market-based approach for redistributing the spectrum where new users can gain access to the spectrum they desperately need and existing owners can gain a financial incentive to "lease" their idle spectrum.

Auctions are among the best-known market-based allocation mechanisms due to their perceived fairness and allocation efficiency. In this work, we consider a dynamic spectrum auction system akin to the eBay marketplace that serves and scales to many small players without manual mediation. In this marketplace, wireless nodes request spectrum in their *local* neighborhood in *short-terms*. These small players request spectrum based on present demand and pay for what they really need without burdensome up-front investment in the FCC-style auctions.

One critical requirement to initiate the proposed marketplace is to ensure that auctions are quickly conducted to enable on-demand short-term spectrum redistribution. To address this challenge, we propose VERITAS, a truthful (or strategy-proof) and efficient spectrum auction. A truthful auction guarantees that if a bidder bids the true evaluation of the resource, its utility will not be less than that when it lies. In this extended abstract, we outline the basic idea of VERITAS design. A more comprehensive description and rigorous study can be found in [2].

2. CHALLENGES OF TRUTHFUL AND EFFICIENT SPECTRUM AUCTION DESIGN

In this section, we describe the problem of truthful auction design, and its challenges in spectrum auctions.

2.1 Problem Definition

We consider a collusion-free spectrum auction setting, where one auctioneer auctions k channels to n bidders. We assume that the channels have uniform characteristics and values, so that bidders request spectrum by submitting the number of channels they demand and the per-channel prices they would like to pay. To make the problem tractable, we represent the conflict condition among bidders by a conflict graph – two bidders either interfere with each other and cannot use the same channels, or can reuse the same channels simultaneously.

Let us first introduce some notations and then give the problem definition of truthful spectrum auction.

b_i – The *per-channel bid* submitted by bidder i.
v_i – i's *per-channel valuation* describing the true price i is willing to pay for each channel.
p_i – The *clearing price* charged for each winner i.

u_i – i's *utility* as the residual worth of the channels. That is, $u_i = v_i \cdot d_i^a$ - p_i if i obtains d_i^a channels, and 0 if it obtains none.

DEFINITION 1. *A truthful auction is one in which no bidder i can obtain higher utility u_i by setting $b_i \neq v_i$.*

DEFINITION 2. *An efficient and a truthful spectrum auction is one which is truthful and maximizes the efficiency of spectrum usage subject to the interference constraints.*

2.2 Challenges in Spectrum Auctions

Given the above definitions, we now describe the two unique properties that set spectrum auctions fundamentally different from (and much more difficult than) conventional multi-unit auctions. First, spectrum can be spatially reused concurrently – two conflicting bidders must not use the same channels simultaneously yet well-separated bidders can. While a conventional auction with n bidders and k channels can only have at most k winners, spectrum auction can have more than k winners. Let's consider a simple example of $n = 3$ bidders competing for $k = 2$ channels, each requesting 1 channel. Figure 1(left) plots the conflict graph, where each vertex represents a bidder and two vertices share an edge if they conflict. A conventional auction will sell channels to at most 2 bidders, while an *efficient* spectrum auction can assign channels to all 3 bidders. Second, the conflict constraints among bidders are in general heterogenous, making the problem of optimizing spectrum allocation NP-complete [1] even when each bidder requests one channel.

In [2], we show that these unique properties of spectrum allocation bring significant challenges into truthful and efficient spectrum auction designs. Existing truthful designs in conventional auctions, such as secondary pricing auctions and VCG-style auctions, either fail to be truthful, require exponential computational complexity, or significantly degrade spectrum utilization when applied to spectrum auctions. Thus we need new designs for truthful and efficient spectrum auctions.

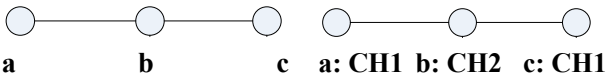


Figure 1: An illustrative example on spectrum allocations. (Left) The conflict graph of a network with 3 bidders. (Right) The optimal spectrum allocation when there are 2 channels.

3. VERITAS AUCTION DESIGN

Motivated by the observations from Section 2, we propose VERITAS, a *truthful* and *computational-efficient* spectrum auction design that also *utilizes spectrum efficiently*. VERITAS consists of a greedy spectrum allocation algorithm to distribute channels among bidders and a pricing mechanism to charge winning bidders. By strategically designing the greedy allocation algorithm, VERITAS achieves similar spectrum utilization/efficiency as the well-known spectrum allocation algorithms in polynomial time. By designing the pricing mechanism to charge bidders by their critical values, VERITAS enforces auction truthfulness despite the complex heterogeneous interference constraints.

We design VERITAS to support a diverse form of spectrum requests. In this section, we introduce VERITAS's main algorithm with *strict requests*, where bidder i requests spectrum by d_i channels and only accepts either 0 or d_i channels.

3.1 Main Algorithm

VERITAS–Allocation Based on the sorted bid set, VERITAS allocates channels from the highest bidder to the lowest bidder. For each bidder i, the allocation algorithm will allocate i channels if there are enough channels to satisfy i.

VERITAS–Pricing VERITAS charges each winning bidder based on the bid of its *critical neighbor*.

DEFINITION 3. *Given a set of bids, a critical neighbor $\mathbb{C}(i)$ of bidder i is a i's conflicting neighbor where if i bids lower than $\mathbb{C}(i)$, i would not win, and if i bids higher than $\mathbb{C}(i)$, i would win in the auction.*

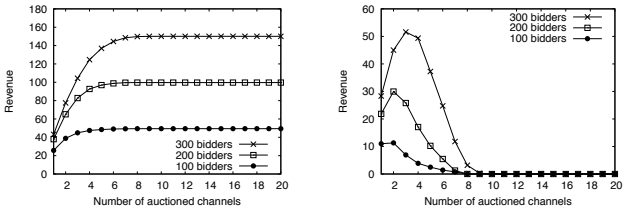
3.2 VERITAS's Truthfulness

We prove VERITAS's truthfulness by three steps: (1) We prove that VERITAS's allocation is *monotonic* – given other bidders' bids, if a bidder is allocated by bidding b, then it will also be allocated by bidding higher than b. (2) We show that for each bidder, there exists a *critical value* such that the bidder wins by bidding higher than this value and loses by bidding lower. (3) We show that by charging winners based on their respective critical values, no bidder can obtain higher utility by bidding other than its true value. Proof details can be found in [2].

3.3 VERITAS's Complexity

THEOREM 1. *VERITAS runs in time $O(n \log n + nk|E|)$, where $|E|$ is the number of edges in the conflict graph G, n is the number of bidders, and k is the number of channels auctioned. Because $|E| \leq \frac{n(n-1)}{2}$, VERITAS runs in time less than $O(n^3k)$.*

In [2], we also show that VERITAS can be easily extended to other bidding formats: (i) *range requests* where bidder i requests d_i channels but expects to receive any number of channels between 0 and d_i; (ii) *contiguous requests* where the channels in strict requests or range requests assigned to i must be contiguously aligned.



(a) Untruthful auction (b) Truthful auction

Figure 2: Comparing VERITAS to untruthful revenue maximizing spectrum auction in revenue.

4. EXPERIMENTAL RESULTS

We perform experiments to explore the property of truthful auctions by comparing VERITAS to a revenue-maximizing yet non-truthful auction. Note that this comparison is unfair because bidders in non-truthful auctions have no incentives to bid their true values. Nevertheless, we plot both results based on the same set of bids to compare the trends. We apply a distance-based interference model to produce the conflict graph. Assume bidders are in a square 1×1 area, and each requests one channel. Each bidder's true evaluation (and hence its bid) is uniformly distributed over $(0, 1]$.

As can be seen in Figure 2, untruthful and truthful auctions behave differently in terms of revenue as the number of channels increases. The revenue of the non-truthful auction increases and then levels off as the number of channels auctioned increases, while the revenue of VERITAS starts to drop beyond a certain number of channels.

This significant difference comes from the fact that the two auctions have fundamentally different charging mechanisms: the non-truthful auction charges winners by their actual bids while the truthful auction VERITAS charges winners by the bids of their critical neighbors. In the non-truthful auction, the increase of channels also increases the number of winners, and hence the revenue which is the sum of winners' bids. In VERITAS, although the number of winners increases, the charges to individual winners decrease as the pool of losing bidders shrinks.

Motivated by this interesting phenomenon, we propose to let the auctioneer determine the number of channels auctioned to maximize its revenue. In [2] we discussed analytically how to choose the number of channels to auction given the information of conflict graph and bid distribution.

5. REFERENCES

[1] JAIN, K., PADHYE, J., PADMANABHAN, V. N., AND QIU, L. Impact of interference on multi-hop wireless network performance. In *Proceedings of MOBICOM* (2003).

[2] ZHOU, X., GANDHI, S., SURI, S., AND ZHENG, H. ebay in the sky: Strategy-proof wireless spectrum auctions. In *Proceedings of MOBICOM* (2008).

http://gswc.cs.ucsb.edu/