Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

AN APPLICATION OF PARALLEL MONTE CARLO MODELING FOR REAL-TIME DISEASE SURVEILLANCE

David W. Bauer Jr.

The MITRE Corporation
7515 Colshire Drive, McLean, VA 22102

Mojdeh Mohtashemi

MIT CS and AI Laboratory
32 Vassar Street, Cambridge MA 02139

ABSTRACT

The global health, threatened by emerging infectious diseases, pandemic influenza, and biological warfare, is becoming increasingly dependent on the rapid acquisition, processing, integration and interpretation of massive amounts of data. In response to these pressing needs, new information infrastructures are needed to support active, real time surveillance. Detection algorithms may have a high computational cost in both the time and space domains. High performance computing platforms may be the best approach for efficiently computing these algorithms. Unfortunately, these platforms are unavailable to many health care agencies. Our work focuses on efficient parallelization of outbreak detection algorithms within the context of cloud computing as a high throughput computing platform. Cloud computing is investigated as an approach to meet real time constraints and reduce or eliminate costs associated with real time disease surveillance systems.

1 INTRODUCTION

Timely detection of infectious disease outbreaks is critical to real time disease surveillance. Space-time detection techniques may require computationally intense search in both the time and space domains (Kulldorff 1997, Kulldorff et al. 2005). The real-time surveillance constraints dictate highly responsive monte carlo models that may be best achievable utilizing high throughput computing (HTC) platforms. We introduce here a system that efficiently parallelizes detection algorithms for execution in HTC environments, specifically Cloud Computing environments. A majority of outbreak detection algorithms are based on the method of monte carlo modeling (Metropolis and Ulam 1949, Hammersley and Handscomb 1964, Nance and Jr 1978, Rubinstein 1981) and can be efficiently parallelized following the work in (Wilmoth 1992, Nance 1995).

The University of Pittsburgh RODS Laboratory (Real-time Outbreak and Disease Surveillance) is the most widely known example of a surveillance system employing high performance computing (Espino et al. 2004, Espino et al. 2004). Using the Pittsburgh Super Computer, RODS is capable of monitoring over 3 million visits to over 137 emergency departments. In addition, RODS monitors 1262 retail stores across Pennsylvania for remedy purchases that could be indicators of outbreak.

Unfortunately, the health care community too frequently lacks the funding to support special purpose high performance computing environments. Health care agencies would benefit greatly from nearly zero cost solution that enhances their ability to perform disease surveillance.

Cloud computing offers a solution based on *opportunistic* or *volunteer* computing, where excess computing cycles are utilized to compute other tasks. Different from volunteer computing projects like SETI@Home however, patient data must be secured according to local, state and federal legal requirements such as the Health Insurance Portability and Accountability Act of 1996 (HIPAA). Most health care agencies already support IT infrastructures for their day to day operations. Therefore, the computing cloud is constructed based on opportunistically scheduling the existing IT computing resources. An implicit assumption is that these machines have been sufficiently secured to contain the patient data.

Opportunistic processing poses some unique challenges compared with dedicated computing resources. To support efficient computation of detection algorithms in this context requires highly parallelized computation in both the data and computational domains. Our proposed system decomposes both space-time and purely temporal detection algorithms in both domains to yield a nearly optimal parallelization given the available computing resources. While there are no data hazards that preclude the application from being embarrassingly parallel, the scheduling algorithm is a form of the bin packing problem, which is known to be NP-Complete. We illustrate parallel performance that is nearly optimal without *a priori* workload information.

In this paper, we describe the application of parallel monte carlo modeling methods to support real time disease surveillance. While we illustrate the efficiency of parallelization using a derivation of the scan statistic (Naus 1965) detection algorithm for disease surveillance in a metropolitan population, in fact any combination of detection algorithms is supported.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

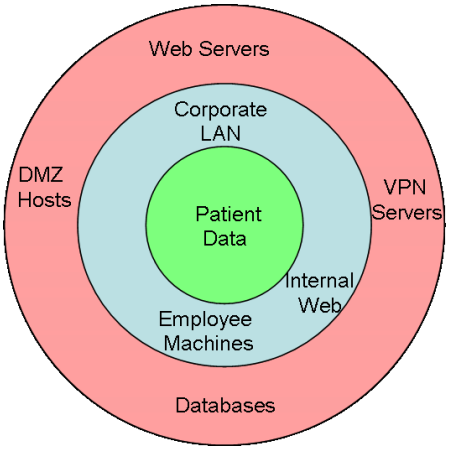


Figure 1: Conceptual health care agency IT infrastructure. Computers in the inner-most section (green) have been sufficiently secured to contain patient data, and would form the basis for a Cloud Computing platform.

In Section 2 we outline the high throughput cloud computing platform created within an existing IT infrastructure. Section 3 describes the architecture and design principles of the system. Sections 4 and 5 present the scenario and algorithms used to test the performance of the system. In Sections 6 and 7 we analyze the ability of the system to meet real time constraints for this scenario. We summarize our results in Section 8.

2 CLOUD COMPUTING

Admittedly, the phrase *Cloud Computing* is nothing more than a popular phrase to describe the idea that applications (typically web applications) can be executed within the "cloud" of the Internet. The term "cloud" refers to the concept that the underlying hardware and network details of the computing platform are not known. Cloud computing platforms are beginning to emerge in the commercial world, most notably is Amazon's Elastic Computing Cloud (EC2). Amazon data centers are provisioned to handle the heaviest buying season, Christmas, but for much of the year these data centers are under utilized. EC2 users can purchase these excess cycles, as well as storage space, at a fixed unit cost based on their actual usage. One high profile example is the popular image-sharing web application Flikr that exists almost entirely within Amazon's EC2 platform.

The Health Insurance Portability and Accountability Act of 1996 (HIPAA, Title II) restricts the ways in which health care agencies must manage and control patient information, typically precluding them from taking advantage of such commercial offerings as Amazon's EC2 platform. One major goal of our distributed system is to create a trusted computing cloud within the existing IT realm of a health care agency. To minimize costs associated with software, we are releasing an open-source, real-time disease surveillance system that takes advantage of a health care agency's excess computing cycles to construct a private cloud computing platform for high throughput computing and real time disease surveillance.

Figure 1 conceptualizes an example health care agency IT infrastructure. The outer section in red contains machines that are open to access from the Internet. The inner blue section contains machines internal to the LAN, that are not accessible via the Internet, but also not trusted to contain local copies of patient data. The inner-most section in green indicates machines that have been secured sufficiently to contain copies of patient data. It is from this inner-most area that trusted machines could be used to create a secure Cloud Computing platform.

The main benefits of this approach are two-fold: *i) the creation of a virtually zero-cost, trusted high throughput computing platform, and ii) the ability to meet the real time constraints of a disease surveillance system through efficient parallelization.*

3 SYSTEM ARCHITECTURE

Figure 2 illustrates the system architecture of our online outbreak detection system. Patient complaint is collected into a patient record database, and information such as symptoms and spatial coordinates are recorded. The Unified Search Framework (Ye and Kalyanaraman 2003) implements the detection algorithm, which queries the patient records for symptoms matching the desired disease for which to perform outbreak detection. The USF schedules the monte carlo simulations with the Cloud Computing environment, and decomposes the patient data for parallelization. The central cloud node is responsible for scheduling the USF generated workload over the cloud compute nodes. The central cloud node schedules work based on an opportunistic policy that permits computation on a cloud compute node if and only if the node in question has a five minute load average below 30%. The central cloud node is also responsible for restarting computations scheduled to cloud compute nodes that have failed for any reason, as well as re-scheduling workload elements between compute nodes as they become tasked over the runtime. For example, if a compute node initially has a negligible load average, the central node schedules work to that compute node. If the load average (external to the model) on that node increases beyond 30%, the compute node vacates the task and informs the central cloud node. The central cloud node then can reschedule that task on any other available machine.

The cloud computing environment relies on the process scheduling system, Condor (Basney and Livny 1999, Livny et al. 1997). For completeness, Figure 2 illustrates the Condor software components within each cloud node type as, Start, Central Master, Collector, Negotiator and Scheduler within the Cloud Computing Environment box.

Google Earth is used to visualize the computed data. The data is stored within the cloud computing storage environment, represented in the figure as the Disease Surveillance Database.

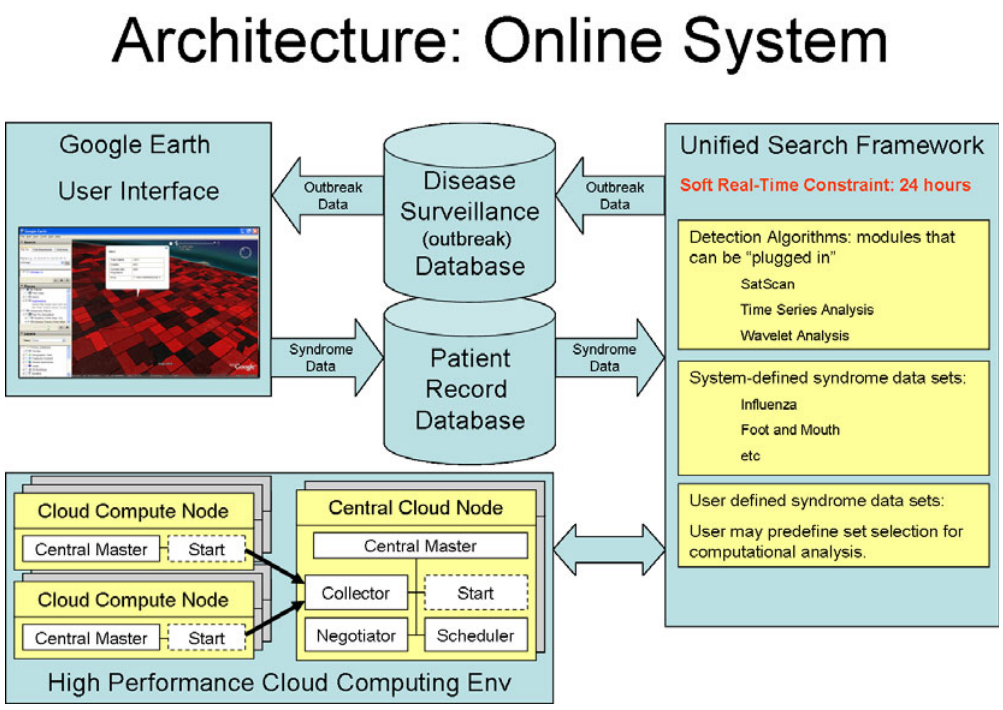


Figure 2: Conceptual health care agency IT infrastructure. Computers in the inner-most section (in green) have been sufficiently secured to contain patient data, and would form the basis for a Cloud Computing platform.

4 AN APPLICATION: SPACE-TIME PERMUTATION SCAN STATISTIC

Variations to both the scan statistic introduced by (Kulldorff et al. 2005) and the method for fast detection of spatial over-densities, provided by (Neill and Moore 2004), is implemented here as a suitable method for early detection of outbreaks in the metropolitan population, particularly for those time/region-specific increases in case frequency that are too subtle to detect with temporal data alone. Similar to the overlapping windows in the method proposed by (Kulldorff et al. 2005), the scanning window utilizes multiple overlapping cylinders, each composed of both a space and time block, where time blocks are continuous windows (i.e. not intermittent) and space blocks are geographic-encompassing regions of varying size. However, instead of circles of multiple radii, a square grid approach, similar to that provided by (Neill and Moore 2004) is implemented here.

Briefly explained below (see (Kulldorff et al. 2005) for complete algorithm details), for each grid element, the expected number of cases, conditioned on the observed marginals is denoted by μ where μ is defined as the summation of expected number of cases in a grid element, given by Equation 1,

$$\mu = \sum_{(s,t)\in A} \mu_{st} \tag{1}$$

where s is the spatial cluster (e.g., zip codes, census tracts, individual addresses) and t is the time span used (e.g., days, weeks, months, etc.) and

$$\mu_{st} = \frac{1}{N}\left(\sum_{s} n_{st}\right)\left(\sum_{t} n_{st}\right) \tag{2}$$

where N is the total number of cases and n_{st} is the number of cases in either the space or time window (according to the summation term). The observed number of cases for the same grid element is denoted by n. Then the Poisson generalized likelihood ratio (GLR), which is used as a measure for a potential outbreak in the current grid element, is given by Equation 3 (Kleinman et al. 2005):

$$\left(\frac{n}{\mu}\right)^{n}\left(\frac{N-n}{N-\mu}\right)^{(N-n)}. \tag{3}$$

Since the observed counts are in the numerator of the ratio, large values of the GLR signify a potential outbreak.

To assign a degree of significance to the GLR value for each grid element, Monte Carlo hypothesis testing (Dwass 1957) is conducted, where the observed cases are randomly shuffled proportional to the population over time and space and the GLR value is calculated for each grid element. This process of randomly shuffling is conducted over 999 trials and the random GLR values are ranked. A p-value for the original GLR is then assigned by where in the ranking of random GLR values it occurs.

5 MODELING SCENARIO

The San Francisco Department of Public Health (SFDPH), Tuberculosis Program provided the data. The geospatial information in the data consist of precise locations of 392 homeless individuals infected with tuberculosis (TB). The primary residences of these individuals were used to identify their geographical coordinates (latitudes and longitudes) using ArcGIS v9.0 (ESRI). The census tract information for identifying the tracts in which the homeless individuals reside, were obtained from generalized extracts from the Census Bureau's TIGER geographic database provided by the US Census Bureau (`http://www.census.gov/geo/www/cob/index.html`). The total number of unique census tracts for our metropolitan area is 76, and the total number of individual addresses is 392.

The scanning window in our scan statistic model utilizes multiple overlapping grid elements, where the space blocks may be either census tracts, or individual addresses. We present results for both scenarios as a comparison of the effective amount of parallelism available in the spatial domain.

For our space window, we restricted the geographic squares to sizes ranging from 0.02 km to 1 km in size, where for each separate sized-square, a neighboring square was allowed to overlap at half of the width on each side. For different sized squares that had a perfect intersect of the same cases, the smallest square was retained. Then, the total number of space squares sampled for the census tract centroids was 441, while the total for individual residences was 4,234.

For our time window case counts we specified time window ranges of 4 to 48 weeks at an interval of 4 weeks, spanning a period of ten years. In addition, our experimentation specified starting week intervals of 1 and 4.

We utilize the Unified Search Framework (USF) developed at RPI for spawning the workload onto the cluster compute nodes, and collecting the overall runtime of the model (Ye and Kalyanaraman 2003). Our expectation is that the spatial blocks will exhibit a uniformly distributed workload, and that will translate into a highly efficient parallel model, although we cannot know the distribution until runtime.

6 SCALABILITY ANALYSIS

We now present a scalability analysis to determine the best approach for balancing the computational workload across multiple processors. This analysis leads to a general form solution for any problem in the scan statistic algorithm data domain; we assume no a priori knowledge of the region or time series data.

The computational workload is defined by the modeling scenario parameters in Table 1. The data domain for the scan statistic algorithm has both spatial and temporal elements. The first set of parameters pertain to the granularity of the experimentation in each domain. The second set of parameters pertain to the data used by the scan statistic algorithm in each domain. Then the maximum resolution for parallel computation is defined by computing the algorithm for a single spatial square and a single overlapping region in the spatial domain, and a single case count and a single time window step in the time domain.

The goal of the decomposition is to parallelize 343,980 (CT) or 3,302,520 (IA) individual computations efficiently. The task of scheduling these computations to multiple processors, which is known to be an NP-Complete problem. Because any NP-Complete problem may be restated as any other NP-Complete problem, we restate the multiprocessor scheduling problem as the bin-packing problem where the CPUs are the bins into which the individual computations must be packed.

A straightforward parallelization approach would be to schedule each individual computation to the next free CPU. This heuristic is known as the *first-fit algorithm* and solves the bin-packing problem in $\theta(n \log n)$ time (Johnson, Demers, Ullman, Garey, and Graham 1974). Unfortunately, this approach does not work well because of the start-up costs associated with initializing and executing the program $10^5 - 10^6$ times (unless you have tens to hundreds of thousands of CPUs available).

A second parallelization approach, that reduces the startup costs, would be to schedule all time domain elements for a centroid to a CPU. This approach yields better results, but is not optimal. The processing time varies widely over the set of centroids, and the running time is dominated by the longest running centroid.

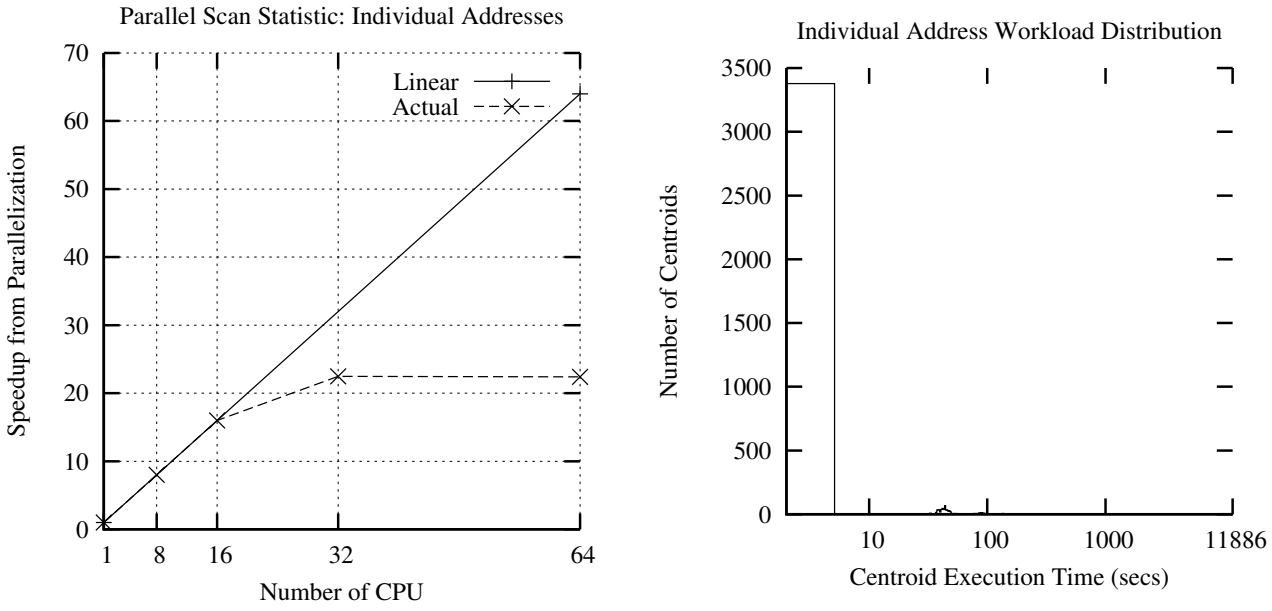
It is clear that for a general form solution we must have a parallelization approach that balances the workload in both the space and time domains, and minimizes the startup costs of the application. We now formulate the expected runtime for the spatio-temporal decomposition over multiple processors:

$$Runtime = \frac{\sigma \cdot c_{startup} + \sigma_{total} \cdot c_{execution}}{N_{cpu}} \tag{4}$$

where σ_{total} denotes the total number of computations in the problem, σ denotes the number of computations per CPU, and N_{cpu} indicates the number of CPUs utilized. $c_{startup}$ is the average amount of time required to initialize the program (i.e., the time required to load the input data into memory, initialize variables, etc.). $c_{execution}$ is the average amount of time required to execute a single computational element (i.e., the scan statistic computation given a single space square, spatial overlap, time window case count and time window step).

Table 1: Scan Statistic Model State Space.

	Min	Max	Step	Size
Model Parameters				
Spatial Square Overlap	0.02km	1km	variable	4,234
Time Window Case Counts (weeks)	4	24	4	6
Data Parameters				
Individual Address Regions	1	396	1	392
Time Window Length (weeks)	1	520	4	130
Total State Space Size				3,302,520



(a) Parallelization Speedup

(b) Workload Distribution

Figure 3: Spatial domain scheduling performance for individual addresses.

Equation 4 states that the runtime of the application is dependent on the amount of time spent starting the application plus the amount of time performing computations, per CPU. A simple example using a single CPU and $\sigma = 1$ leaves us with the sequential runtime:

$$Runtime_{seq} = c_{startup} + \sigma_{total} \cdot c_{execution}. \tag{5}$$

and the formulation is *structurally persistent* (Shi 1996) (i.e., computes the same number of computations in both the sequential and parallel cases).

Then Equation 4 can be simplified to:

$$Runtime = \frac{\sigma \cdot c_{startup}}{N_{cpu}} + \frac{\sigma_{total} \cdot c_{execution}}{N_{cpu}} \tag{6}$$

which illustrates that the first term is the only variable for any problem. The first term in the expression relates to the cost of parallelization overhead, which can be minimized by constraint that the ratio $\sigma : N_{cpu} = 1$. Then the optimal runtime becomes:

$$Runtime_{opt} = c_{startup} + \frac{\sigma_{total} \cdot c_{execution}}{N_{cpu}} \tag{7}$$

and when $c_{startup}$ is very small it can be ignored and our equation agrees with Amdahl's prediction for an application where the sequential portion is zero.

To meet the constraints that $\sigma : N_{cpu} = 1$, the application can only be started once per CPU. Then the application must also contain a scheduling algorithm that balances the workload. We chose round-robin scheduling because it has a straightforward implementation. We show how we can apply the first-fit heuristic in this context to arrive at a nearly optimal, general-form solution for parallelizing the scan statistic algorithm.

7 PERFORMANCE STUDY

In this section we report the performance results using round robin scheduling for mapping the model to processors. We study the decomposition of the problem in both the space and time data domains, and then apply the first-fit heuristic to arrive at a near optimal parallelization. We present results based on both census tracts and the more compute intensive individual addresses.

To simplify the performance results, we do not perform measurements with external loads on the cluster computer used. The goal of the paper is not to study the dynamic nature of the computing environment, but rather to understand the parallel performance of the monte carlo models for the application of outbreak detection.

7.1 Computing Testbed and Experiment Setup

The testbed used is a Red Hat Enterprise Linux 9.0 cluster consisting of 16 machines or compute nodes, for a total of 64 processors. The nodes are inter-connected via a dedicated gigabit ethernet switch. Each node's hardware configuration consists of a dual-processor, dual-core AMD Opteron 275 server and 8GBs of main memory. The AMD Opteron 200-series chip enables 64-bit computing, and provides up to 24GB/s peak bandwidth per processor using HyperTransport technology. The DDR DRAM memory controller is 128-bits wide and provides up to 6.4GB/s of bandwidth per processor. Our RAM configuration consisted of 4 2GB sticks of 400MHz DDR ECC RAM in 8 banks.

Although we expect the surveillance system to execute within a cloud computing environment, deployed within a health authorities local area network and utilizing machines with other primary purposes, we utilize a cluster computer here to study the performance of the workload balancing. Before we can be concerned with the latencies associated with communicating data over the network, and the availability of computing power over the machines used, we must first demonstrate that the system can calculate the data in real-time.

7.2 Spatial Domain Scheduling

The first scheduling algorithm we demonstrate is a decomposition of the data in spatial domain. Here, each spatial centroid is mapped to a CPU and computes the complete time domain. From Table 1, we are performing parallelization based on the spatial square overlap parameter. For census tracts, the total number of centroids, $c_{startup}$ is 441, and for individual addresses, 4,234.

Performing detection based on individual addresses allows for a higher degree of parallelism within the application, although the runtime is far greater as there are more centroids to compute. As with census tracts, the efficiency of the parallelization is high up to 16 CPUs. Figure 3a shows an efficiency of 70% for 32 CPUs, an improvement over census tracts, based on a 22-fold speedup in the runtime.

Table 2: Parallel Performance of the Space Domain Scheduler.

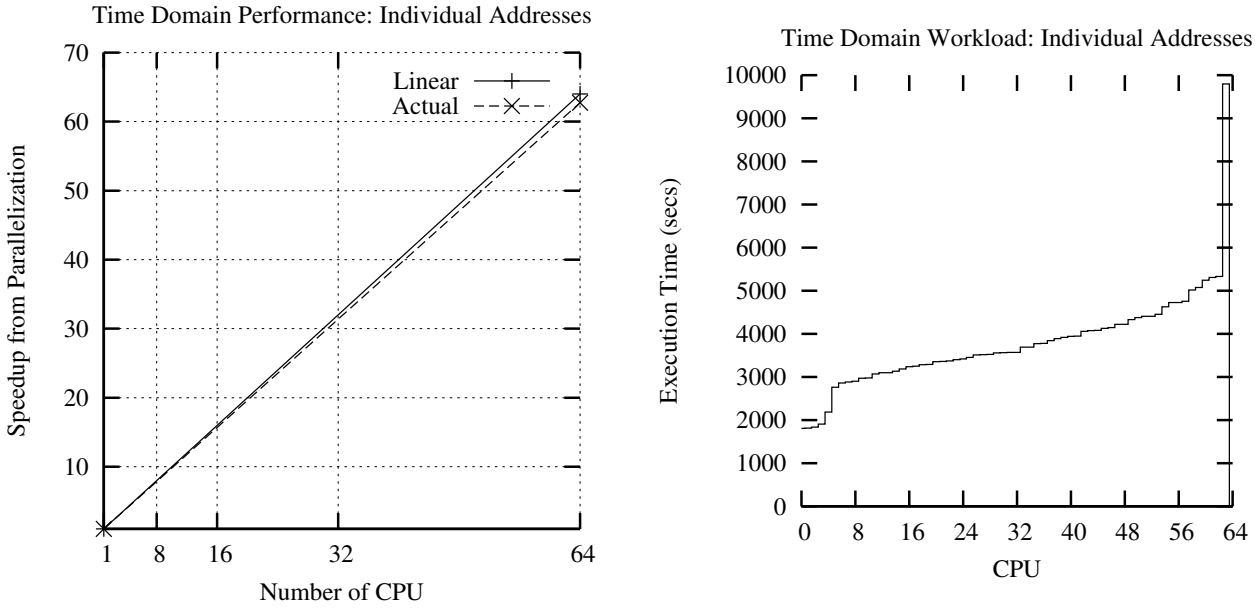
Data Set	# CPU	Runtime (secs)	Speedup	Eff (%)
Census	1	103,129	-	-
Census	8	13,672	7.54	94.25
Census	16	6,940	14.86	92.875
Census	32	6,977	14.78	46.18
Census	64	7,055	14.61	22.82
IA	1	267,029	-	-
IA	8	33,305	8.0	100
IA	16	14,401	16.0	100
IA	32	11,886	22.4	70.18
IA	64	11,918	22.4	35.0

The histogram in Figure 3b illustrates that the workloads based on individual addresses are uniformly distributed for the most part, although there is one address that requires over 3.3 hours to compute. Because the runtime for the longest census tract is smaller than the runtime of the longest individual address (6,977 vs. 11,886 seconds), the individual address decomposition is not able to complete in less time. The parallel performance for either are relatively close; execution times for all experiments are shown in Table 2.

7.3 Temporal Domain Scheduling

The second scheduling algorithm we illustrate is a data decomposition in the temporal domain. Here, each CPU processes a subset of the time series workload for each centroid in the problem. From Table 1, we are using the time window parameter for parallelization. Figure 6 illustrates the decomposition of the time domain by segmenting the time series into N_{cpu} equal size subsets. Each CPU is enumerated from $i : 1..N_{cpu}$ and computes the algorithm for the i^{th} position in each subset.

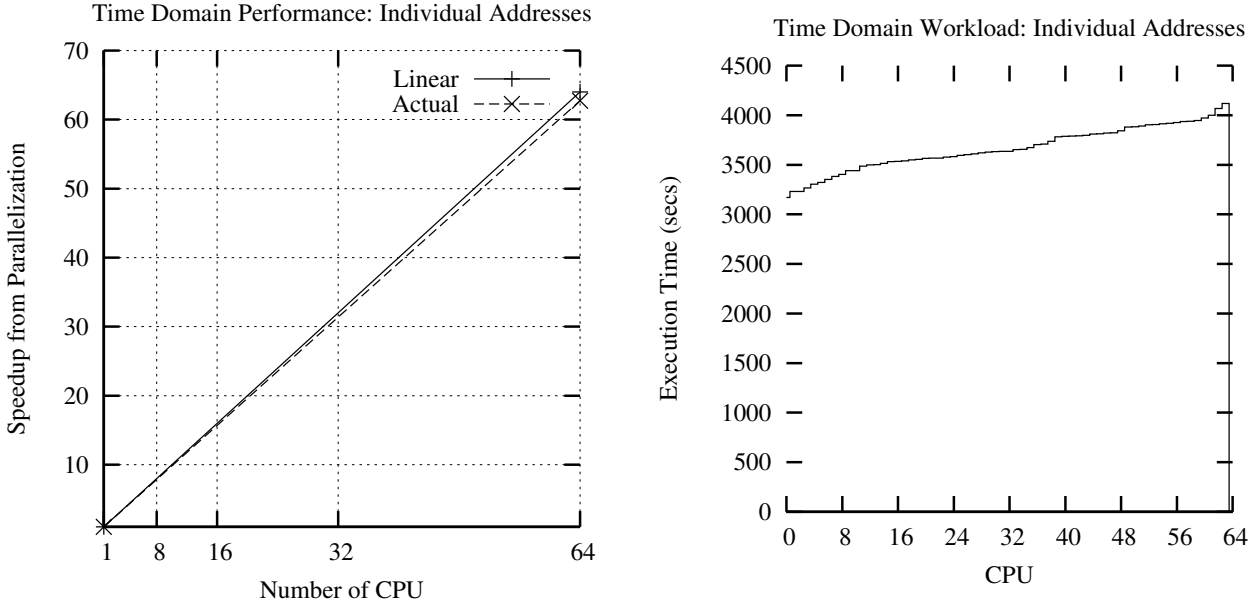
Figure 4a focuses on the temporal decomposition for 64 CPUs. Due to limited resources, we were unable to collect intermediate results for 8, 16 and 32 processors. However, the speedup using 64 processors is close to 30-fold, a 50% improvement over purely spatial scheduling.



(a) Parallelization Speedup

(b) Workload Distribution

Figure 4: Temporal domain scheduling performance for individual addresses.



(a) Parallelization Speedup

(b) Workload Distribution

Figure 5: Spatio-temporal domain scheduling performance for individual addresses.

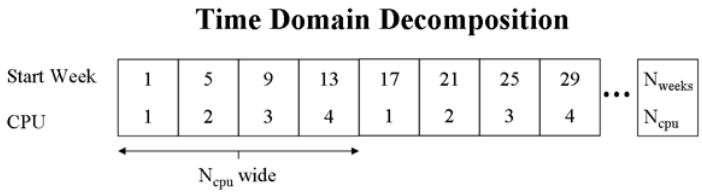


Figure 6: Decomposition of the time series.

Figure 4b illustrates the per CPU runtime. The performance improvement in this scheduling algorithm is due to the longest running task time reduction from 11,886 seconds to under 10,000 seconds. However, the workload is still unbalanced.

7.4 Spatio-Temporal Scheduling

The final scheduling algorithm we illustrate is a data decomposition in both the spatial and temporal domain. Here, each CPU processes a subset of the time series workload, offset for each consecutive centroid computed. This has the effect of round robin allocating the time domain as well so that no one CPU ends up computing the longest period for each time series.

Figure 5a illustrates nearly optimal performance for 64 CPUs based on the longest running task time of 4,119 seconds. Figure 5b shows that the workload is nearly balanced across all CPUs.

8 CONCLUSIONS & FUTURE WORK

The reality of the situation is that many health services agencies cannot set aside funds for high performance computing platforms such as cluster computers. We have demonstrated highly efficient parallel computation for monte carlo models for outbreak detection. We have shown that high throughput computing platforms, such as a Cloud Computing environment based on an existing IT infrastructure, can be low-cost while still meeting the needs of real time surveillance constraints.

In the future, we would like to investigate the possibility of connecting multiple surveillance systems to create a regional or national surveillance system. Another challenging area for this work is to investigate the possibility of constructing computation clouds in developing regions of the world.

REFERENCES

Basney, J., and M. Livny. 1999. Deploying a high throughput computing cluster. In *High Performance Cluster Computing: Architectures and Systems, Volume 1*, ed. R. Buyya. Prentice Hall PTR.

Dwass, M. 1957. Modified Randomization Tests for Nonparametric Hypotheses. *The Annals of Mathematical Statistics* 28 (1): 181–187.

Espino, J., M. Wagner, C. Szczepaniak, F. Tsui, H. Su, R. Olszewski, Z. Liu, W. Chapman, X. Zeng, L. Ma et al. 2004. Removing a barrier to computer-based outbreak and disease surveillance–the RODS Open Source Project. *MMWR Morb Mortal Wkly Rep* 53:32–9.

Espino, J., M. Wagner, F. Tsui, H. Su, R. Olszewski, Z. Lie, W. Chapman, X. Zeng, L. MaL, and J. Dara. 2004. The RODS Open Source Project: Removing a Barrier to Syndromic Surveillance. *Medinfo* 2004:1192–6.

Hammersley, J., and D. Handscomb. 1964. *Monte Carlo methods*. Methuen.

Johnson, D., A. Demers, J. Ullman, M. Garey, and R. Graham. 1974. Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal on Computing* 3:299.

Kleinman, K. P., A. M. Abrams, M. Kulldorff, and R. Platt. 2005. A model-adjusted space–time scan statistic with an application to syndromic surveillance. *Epidemiology and Infection* 133 (03): 409–419.

Kulldorff, M. 1997. A spatial scan statistic. *Communications in Statistics-Theory and Methods* 26 (6): 1481–1496.

Kulldorff, M., R. Heffernan, J. Hartman, R. Assuncao, and F. Mostashari. 2005. A space-time permutation scan statistic for disease outbreak detection. *PLoS Med* 2 (3): e59.

Livny, M., J. Basney, R. Raman, and T. Tannenbaum. 1997, June. Mechanisms for high throughput computing. *SPEEDUP Journal* 11 (1).

Metropolis, N., and S. Ulam. 1949. The Monte Carlo Method. *Journal of the American Statistical Association* 44 (247): 335–341.

Nance, R. 1995. Monte Carlo simulation of three-dimensional hypersonic flows on parallel architectures. Master's thesis, North Carolina State University.

Nance, R., and C. Jr. 1978. Some Experimental Observations on the Behavior of Composite Random Number Generators. *Operations Research* 26 (5): 915–935.

Naus, J. 1965. The Distribution of the Size of the Maximum Cluster of Points on a Line. *Journal of the American Statistical Association* 60 (310): 532–538.

Neill, D., and A. Moore. 2004. A fast multi-resolution method for detection of significant spatial disease clus-

ters. *Advances in Neural Information Processing Systems* 10:651–658.

Rubinstein, R. 1981. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc. New York, NY, USA.

Shi, Y. 1996. Reevaluating Amdahl's Law and Gustafson's Law. *Computer Sciences Department, Temple University (MS: 38-24), Oct*.

Wilmoth, R. 1992. Adaptive domain decomposition for monte carlo simulations on parallel processors. *AIAA Journal* 30 (10): 2447–2452.

Ye, T., and S. Kalyanaraman. 2003. A unified search framework for large-scale black-box optimization. Technical report, Rensselaer Polytechnic Institute, ECSE Department, Networks Lab.

AUTHOR BIOGRAPHIES

DAVID W BAUER JR is a Lead Modeling & Simulation Systems Engineer at the MITRE Corporation. He earned a Ph.D., M.S., and B.S. from Rensselaer Polytechnic Institute in 2005, 2004, and 2000, respectively. Prior to joining MITRE, he was a research scientist with AT&T and GE. His research interests include parallel and distributed systems, cloud computing, large-scale simulation, wired and wireless networking, and computer architecture. His e-mail address is <bauerd@cs.rpi.edu>.

MOJDEH MOHTASHEMI has a PhD in Computer Science from the Massachusetts Institute of Technology. Her research covers topics in population dynamics of infectious diseases, early detection of infectious disease outbreaks, transients and population vulnerability in health and disease, mathematical modeling of complex biological and social systems, and evolutionary biology. She is currently directing the project in biomathematics and infectious disease surveillance at the MITRE Corporation. She is also a research affiliate of the MIT Department of Computer Science. Her e-mail address is <mojdeh@mitre.org>.