High Performance Computing for Disease Surveillance

David Bauer[†], Brandon W. Higgs[†], and Mojdeh Mohtashemi[†‡]

The MITRE Corporation[†],
7515 Colshire Drive, McLean VA 22102
{dwbauer,bhiggs,mojdeh}@mitre.org
http://www.mitre.org/

MIT CS and AI Laboratory[‡],
32 Vassar Street, Cambridge MA 02139
mojdeh@mit.edu
http://www.mit.edu/

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. Space-time detection techniques 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 techniques. Our work focuses on efficient parallelization of these computations on a Linux Beowolf cluster in order to attempt to meet these real time needs.

Key words: HPC, High Performance Computing, Parallel Computing, Disease Surveillance, Beowolf cluster

1 Introduction

Timely detection of infectious disease outbreaks is critical to real time surveillance. Space-time detection techniques may require computationally intense search in both the time and space domains [5, 6]. The real time surveillance constraints dictate highly responsive models that may be best achievable utilizing high performance computing platforms. We introduce here a technique that performs efficiently in parallel on a Linux Beowolf Cluster.

In the special case of parallelization, Amdahl's law [1] states that if F is the portion of an application that is sequential, then the maximum speedup S that can be achieved using N processors is given by Equation 1.

$$S_{max} = \frac{1}{F + \frac{(1-F)}{N}} \ . \tag{1}$$

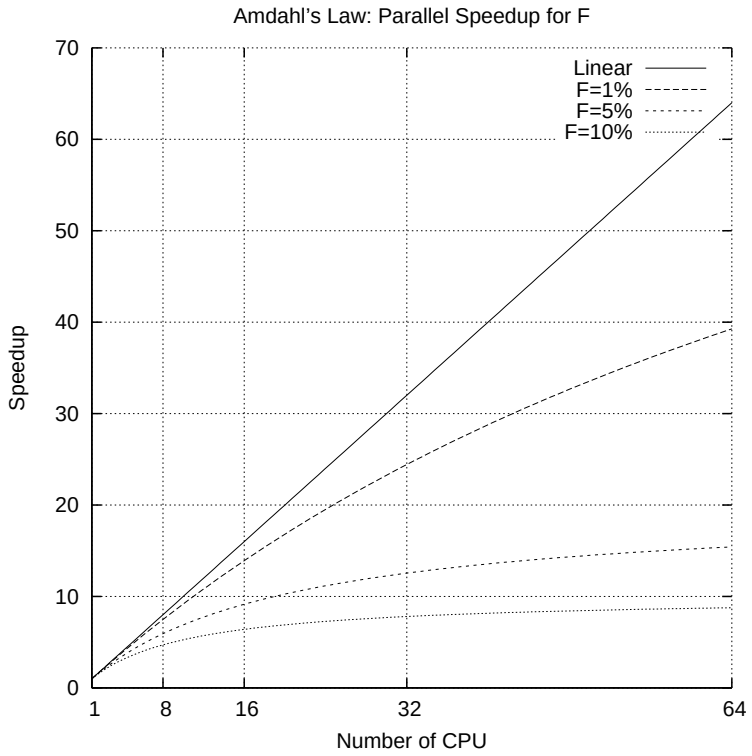


Fig. 1. Amdahl's Law illustrates effective speedup.

As $N \to \infty$, S_{max} approaches $1/F$, and thus the amount of improvement is limited by the sequential portion of the application. Conversely, as $F \to 0$, S_{max} goes to N, that is the ideal parallelization, where each additional processor contributes fully to the performance of the application. In practice, this is a difficult goal to achieve. Efficient parallelization requires F be small as N increases. Figure 1 illustrates that, for $N = 64$ processors and $F = 10\%$ sequential, that the expected speedup approaches, but cannot exceed a 10-fold improvement in the running time. Finally, we define the *efficiency* of the parallelization by the Equation 2:

$$E_N = \frac{S_{max}}{N} \ . \tag{2}$$

The traditional approaches towards parallelizing an application fall into two categories: *(i) data and (ii) algorithm decomposition*. Because of the high degree of data dependency within the scan statistic algorithm, we focus on decomposing the application in the data domain. We modify the scan statistic data domain by computing individual addresses, thereby making the scope of the calculation significantly larger, based on the $O(N^2)$ algorithm complexity. While the overall

running time of the application is larger on a single CPU, we will show that it can be computed in less time utilizing multiple processors executing in parallel by minimizing F.

The scan statistic was first introduced by Naus for cluster detection in 1965 [7] and later implemented and refined in other work [4–6, 9, 10], primarily for aberrant event detection such as those provided for surveillance purposes. The idea relies on a scanning window that utilizes multiple overlapping cylinders, each composed of both a space and time block, where time blocks are continuous windows (i.e., non-intermittent) and space blocks are geographic encompassing circles of varying radii, such as zip codes or census tracts.

In this paper, we focus our modeling efforts on clustering individual addresses as opposed to agglomerative regions (i.e. zip codes, census tracts, etc.) while using a derivation of the scan statistic [7] for disease surveillance in a metropolitan population. The use of individual addresses with the modified scan statistic is found to be a more sensitive measure for detection than census tract centroids. The former method also provides smaller time windows for significantly detected signals, which can speed the response time of an outbreak.

2 An Application: Space-time Permutation Scan Statistic

Variations to both the scan statistic introduced by [6] and the method for fast detection of spatial over-densities, provided by [8], 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 [6], 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 [8] is implemented here.

Briefly explained below (see [6] 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 3,

$$\mu = \sum_{(s,t)\in A} \mu_{st} \ . \tag{3}$$

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

$$\mu_{st} = \frac{1}{N} \left(\sum_{s} n_{st} \right) \left(\sum_{t} n_{st} \right) \ . \tag{4}$$

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 5 [4]:

$$\left(\frac{n}{\mu}\right)^n \left(\frac{N-n}{N-\mu}\right)^{(N-n)} . \tag{5}$$

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 [2] 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.

3 Performance Study

3.1 Computing Testbed and Experiment Setup

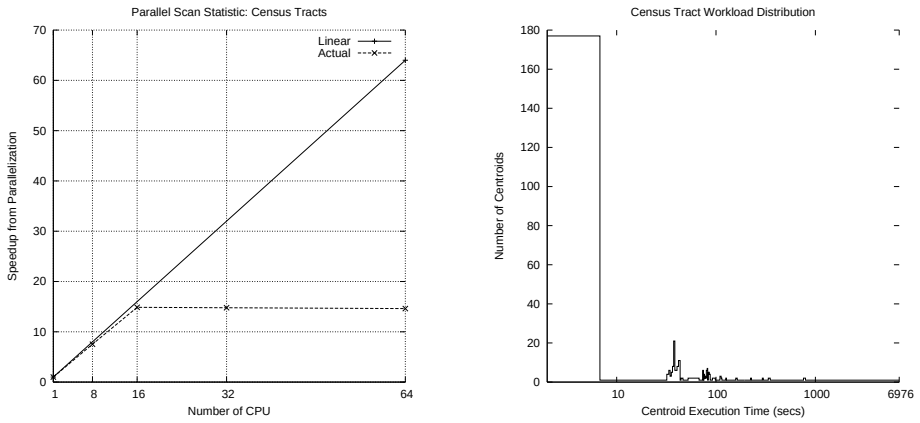
The Hive cluster at MITRE is a Red Hat Enterprise Linux 9.0 cluster consisting of 16 machines or compute nodes, for a total of 64 processors. The nodes are inter-connected via a dedicated gigabit ethernet switch. Each node's hardware configuration consists of a dual-processor, dual-core AMD Opteron 275 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.

3.2 Model Scenario

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

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 total number of unique census tracts for our metropolitan area is taken to be 76. We mapped each individual location to the corresponding census tract using ArcGIS v9.2 (ESRI).

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



(a) Parallelization Speedup

(b) Workload Distribution

Fig. 2. Performance results for census tracts.

was retained. 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 were used with time windows of 4 to 72 weeks spanning a period of ten years.

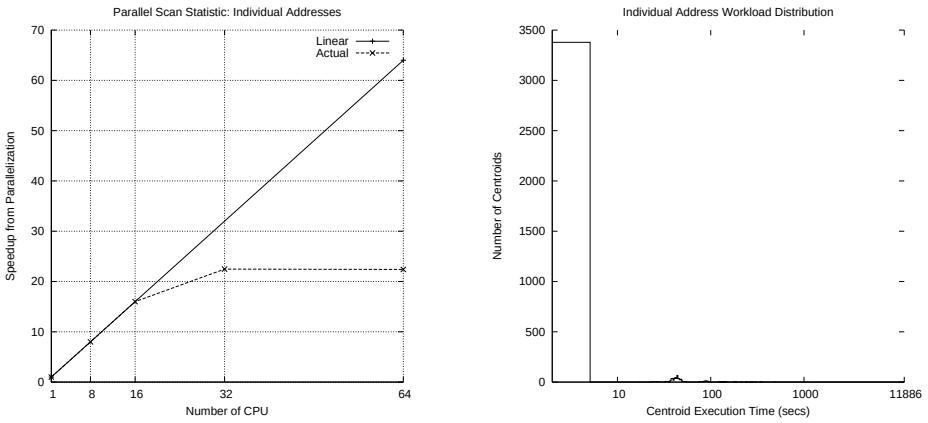
The scan statistic model is thus parallelized by mapping the individual spaces onto CPUs for processing. We utilize the Unified Search Framework (USF) developed at RPI for spawning the jobs onto the cluster compute nodes, and collecting the overall runtime of the model [11]. Our expectation is that the 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.

3.3 Census Tract Performance

For comparison purposes, we first perform a parallelization of the scan statistic using census tracts. Our model consists of 441 centroids, and we computed mappings over 1, 8, 16, 32 and 64 processors. The purpose is to illustrate the effective scalability over a varying number of processors.

Figure 2a shows that using 16 processors achieves an efficiency of 93%, reaching an almost 15-fold improvement over sequential. When we utilize 32 or 64 CPUs, the efficiency drops to under 45%, indicating that there is a large sequential component to the data set once the runtime improves to around 7,000 seconds.

The issue here is that the workloads are not uniform in terms of computational workload, with some spatial squares requiring far more effort. The workload distribution is described by the histogram displayed in Figure 2b, and the longest running job last 6,976 seconds. In the 8 and 16 CPU cases, the run-



(a) Parallelization Speedup

(b) Workload Distribution

Fig. 3. Performance results for individual addresses.

ning times of all other centroids consumed at least this much time, and so the efficiency of the computation is high. The 32 and 64 CPU cases are unable to improve on the running time of the longest centroid, and so the efficiency drops as more CPU are added. To keep the efficiency high would require either increasing the total workload in the system, or parallelization within the scan statistic algorithm for the longest running centroids.

Although the efficiency of the parallelization limits us to making effective use of no more than 16 CPUs, we do report a significant improvement on the overall runtime of the algorithm for this data decomposition. The runtime dropped from over 28 hours to just under 2 hours.

3.4 Individual Address Performance

Decomposing the data domain into individual addresses allows for a higher degree of parallelism within the application. As with census tracts, the efficiency of the parallelization is high up to 16 CPUs. Figure 3a shows an efficiency of 68.75% for 32 CPU, an improvement over census tracts, and a 22-fold improvement in the runtime.

It is clear from the histogram in Figure 3b that the workloads based on individual addresses are more uniformly distributed, allowing for more efficient use of the cluster resources. However, the overall running time of the parallel application continues to be dominated by the most computationally complex centroid. Because the runtime of the longest census tract is smaller than the runtime of the longest individual address (6,977 seconds vs. 11,886), the individual address decomposition is not able to complete in less time on the cluster. The overall runtime for either are relatively close; execution times for all experiments are shown in Table 1.

Table 1. Parallel Execution Performance of the Scan Statistic Model.

Data Set	# CPU	Runtime in Seconds	Speedup	Efficiency (%)
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
Individual	1	267,029	-	-
Individual	8	33,305	8.0	100
Individual	16	14,401	16.0	100
Individual	32	11,886	22.4	70.18
Individual	64	11,918	22.4	35.0

The sequential running time for individual addresses is nearly 3 times larger than for the census tract data, from 28 to 74 hours. The parallelization improvement yields a decrease in the runtime of the model, to 3.3 hours.

4 Footnote on Parallelism

In [3], researchers at Sandia illustrate that the size of the model should be scaled with the number of processors utilized. By fixing the size of the model, it would appear that Amdahl's law would dictate only *embarrassingly parallel* (F very close to 0) applications would see a speedup using more than 100 CPUs. However, whenever additional computing capacity is available, it is more common that the size of the problem grows to consume that capacity. It may be more meaningful to propose a fixed *runtime*, rather than a fixed *problem size*.

In fact, this situation is encountered here. The size of our data set has not been scaled with the number of processors utilized. If we had done so, then the efficiency of the parallelization would have been much closer to 100% for the 32 and 64-CPU cases. This stems from the fact that the sequential portions of the application (time for start-up, serial bottlenecks, etc) do not increase as the problem size increases.

5 Conclusions & Future Work

We have proposed a highly efficient parallel computation technique for the real time surveillance of infectious disease outbreak detection. We have shown that high performance computing platforms, such as a Linux Beowolf Cluster, can come closer to meeting the needs of real time surveillance constraints.

In this paper we have reported results on the parallelization of a scan statistic model that has been modified to compute individual addresses as well as

agglomerate regions, such as census tracts. We have shown that while the sequential execution time of the model is significantly larger, we can equalize the magnitude of the running time through parallelization of the model.

In the future we plan to investigate methods of parallelizing the scan statistic algorithm, to further increase the efficiency of the model on multiple processors. One expected outcome of this effort would be to balance the workload within a single centroid over multiple processors, and thus balance the non-uniform workload across the cluster computer. Our intent would be to improve the efficiency of the parallel scan statistic model.

References

1. Amdahl, G.: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. In AFIPS Conference Proceedings (1967) 30:483-485
2. Dwass M.: Modified randomization tests for non-parametric hypotheses. In The Annals of Mathematical Statistics (1957) 29:181187
3. Gustafson, J.: Reevaluating Amdahl's Law. In Communications of the ACM (1988) 31(5):532-533
4. Kleinman KP, Abrams AM, Kulldorff M, Platt R: A model-adjusted space-time scan statistic with application to syndromic surveillance. In Epidemiology and Infectection (2005) 000:1-11
5. Kulldorff M.: A spatial scan statistic. Communications in Statistics: Theory and Methods (1997) 26:1481-1496
6. Kulldorff M, Heffernan R, Hartmann J, Assuncao R, Mostashari F: A space-time permutation scan statistic for disease outbreak detection. In Public Library of Science (2005) 2(3)
7. Naus J.: The distribution of the size of maximum cluster of points on the line. In Journal of the American Statistical Association (1965) 60:532-538
8. Neill DB, Moore AM.: A fast multi-resolution method for detection of significant spatial disease clusters. In Advances in Neural Information Processing Systems (2003) 16
9. Wallenstein S.: A test for detection of clustering over time. In American Journal of Epidemiology (1980) 111:367-372
10. Weinstock MA.: A generalized scan statistic test for the detection of clusters. In International Journal of Epidemiology (1982) 10:289-293
11. Ye T., Kalyanaraman S.: A Unified Search Framework for Large-scale Black-box Optimization. Rensselaer Polytechnic Institute, ECSE Department, Networks Lab (2003)