Biomedical Case Studies in Data Intensive Computing

Geoffrey Fox[1,2], Xiaohong Qiu[1], Scott Beason[1], Jong Choi[1,2], Jaliya Ekanayake[1,2], Thilina Gunarathne[1,2], Mina Rho[2], Haixu Tang[2], Neil Devadasan[3], and Gilbert Liu[4]

[1] Pervasive Technology Institute
[2] School of Informatics and Computing
[3] The Polis Center
[4] School of Medicine Indiana University
{gcf,xqiu,smbeason,jychoi,jekanaya,tgunarat,mrho,
hatang}@indiana.edu,
{ndevadas,gcliu}@iupui.edu

Abstract. Many areas of science are seeing a data deluge coming from new instruments, myriads of sensors and exponential growth in electronic records. We take two examples – one the analysis of gene sequence data (35339 Alu sequences) and other a study of medical information (over 100,000 patient records) in Indianapolis and their relationship to Geographic and Information System and Census data available for 635 Census Blocks in Indianapolis. We look at initial processing (such as Smith Waterman dissimilarities), clustering (using robust deterministic annealing) and Multi Dimensional Scaling to map high dimension data to 3D for convenient visualization. We show how scaling pipelines can be produced that can be implemented using either cloud technologies or MPI which are compared. This study illustrates challenges in integrating data exploration tools with a variety of different architectural requirements and natural programming models. We present preliminary results for end to end study of two complete applications.

Keywords: MapReduce, Clouds, MPI, Clustering, Sequencing, Dryad. Hadoop.

1 Introduction

Data Intensive Computing is very popular at this time. Partly this is due to the well understood data deluge with all activities including science, government and modern Internet (Web 2.0) systems all generating exponentially increasing data. One special driver is that Web Search and related data mining can use an especially simple programming model MapReduce of which there are now several implementations. It is attractive to understand how generally applicable MapReduce is to other data intensive problems as one can expect excellent commercial support for software in this area. We have looked at the impact of clouds and compared Yahoo (Hadoop) and Microsoft (Dryad) implementations of the MapReduce step presenting Dryad results here. We choose two biomedical applications. The first addresses the structure of Gene families and the processing steps involve sequence alignment, clustering and visualization after projecting sequences to 3 dimensions using Multidimensional

M.G. Jaatun, G. Zhao, and C. Rong (Eds.): CloudCom 2009, LNCS 5931, pp. 2–18, 2009.

scaling MDS. The second application involves correlating electronic patient records with environmental information (from Geographical Information Systems) associated with the patient location. Here the end to end study involves substantial data validation, processing with many standard tools such as those in R but also many possible other applications such as Multidimensional Scaling dimension reductions and Genetic algorithm based optimizations.

We present performance results from Tempest – An Infiniband connected 32 node system running Windows HPCS with each node having 24 cores spread over 4 Intel chips. Such a modest cluster can fully process all stages of the 35,000 element Alu study in less than a day and is suitable for up to 200,000 sequences even though all steps in analysis are of $O(N^2)$ time complexity. We estimate that a 1024 node Tempest architecture cluster would tackle well our million sequence goal. We find systems easy to use and program as well as giving good wall clock execution time. Some of our studies used a slightly older cluster Madrid with 8 nodes each with four AMD Opteron chips with 4 cores each. Section 2 presents some overall architecture comments while sections 3 and 4 describe the two main applications. Section 5 has conclusions and future work.

2 Data Intensive Computing Architecture

The computer architecture needed to support data intensive computing is obviously complex and varied. Here we do not discuss virtualization or issues of distributed systems which although important are not the topic of this paper. We abstract many approaches as a mixture of pipelined and parallel (good MPI performance) systems, linked by a pervasive storage system. Here we have many interesting possibilities including Amazon and Azure "Blob" storage, traditional supercomputer environment like Lustre plus importantly the file systems (such as Cosmos from Microsoft or HDFS from Hadoop) supporting the new MapReduce systems. These cloud/Web 2.0 technologies support a computing style where data is read from one file system, analyzed by one or more tools and written back to a database or file system. An important feature of the newer approaches is explicit support for file-based data parallelism which is needed in our applications. In figure 1, we abstract this disk/database-compute model and assume it will underlie many applications even when some of resources will be local and others in the cloud or part of a large grid. In figures 2 and 3 we give in more detail the data pipelines used in the applications of sections 3 and 4 respectively.

We record in table 1, the major facilities used in this study. Note they run Windows (HPC Edition) and stress both multicore and traditional parallelism. The largest Tempest cluster has 768 Intel Cores spread over 32 nodes while the smaller one Madrid has 128 Opteron cores spread over 8 nodes. Our work [5, 19, 21] stresses both Windows and Linux so we can explore Hadoop, Dryad and the emerging cloud approaches. This paper focuses on results from the Windows clusters.

Table 1. Hardware and software configurations of the clusters used in this paper. In addition a traditional 8-node Linux Cluster "Gridfarm" was used to run statistics package R in section 4.

System/ Size	CPU	Memory	Operating System	Network
Tempest 32 Cluster + 1 Head	4 Intel Six CoreXenon E7450 2.4 GHz	Cluster:48 GB Head: 24 GB 12 MB Cache	Windows Server 2008 HPC Ed. (SP1)	1 Gbps Ethernet
Madrid 8 Cluster + 1 Head	4 AMD Quad Core. Opteron 8356 2.3GHz	Cluster:16 GB Head: 8 GB 2 MB Cache	Windows Server HPC Ed. (SP1)	20Gbps Infiniband

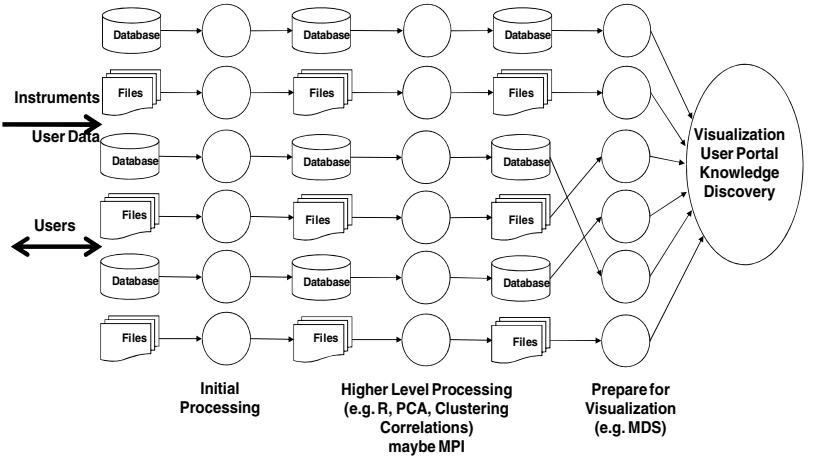

Fig. 1. A Data intensive computing architecture

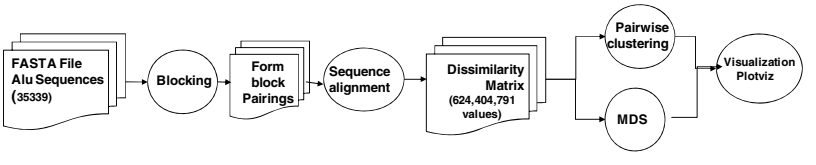

Fig. 2. Stages of Gene sequencing application processing pipeline

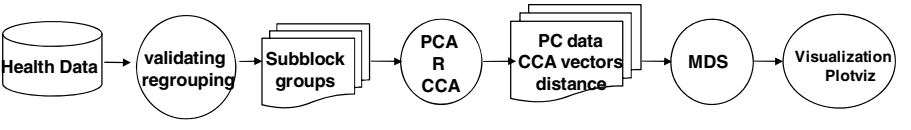

Fig. 3. Stages of health application processing pipeline

3 Gene Sequencing Applications

3.1 Alu Sequencing Studies

The Alu clustering problem [13] is one of the most challenging problem for sequencing clustering because Alus represent the largest repeat families in human genome. There are about 1 million copies of Alu sequences in human genome, in which most insertions can be found in other primates and only a small fraction (~ 7000) are human-specific insertions. This indicates that the classification of Alu repeats can be deduced solely from the 1 million human Alu elements. Notable, Alu clustering can be viewed as a classical case study for the capacity of computational infrastructures because it is not only of great biological interests, but also a problem of a scale that will remain as the upper limit of many other clustering problem in bioinformatics for the next few years, e.g. the automated protein family classification for a few millions of proteins predicted from large metagenomics projects.

3.2 Smith Waterman Dissimilarities

The first step is to identify human Alu gene sequences which were obtained by using Repeatmasker [14] with Repbase Update [15]. We have been gradually increasing the size of our projects with the sample in this paper having 35339 sequences (while we are preparing to analyze 300,000) and requires a modest cluster such as Tempest (768 cores). Note from the discussion in section 3.1, we are aiming at supporting problems with a million sequences -- quite practical today on TeraGrid and equivalent facilities given basic analysis steps scale like $O(N^2)$.

We used open source version [16] of the Smith Waterman – Gotoh algorithm SW-G [17, 18] modified to ensure low start up effects by each thread/processing large numbers (above a few hundred) at a time. Memory bandwidth needed was reduced by storing data items in as few bytes as possible.

3.2.1 Performance of Smith Waterman Gotoh SW-G Algorithm with MPI

The calculation of the 624 million independent dissimilarities is of course architecturally simple as each computation is independent. Nevertheless it shows striking structure shown in figure 4. As in previous papers [5, 21], we look at different patterns denoted as (Thread per process) x (MPI process per 24 core node) x (Number of Nodes) or in short the pattern *t*x*m*x*n*. We have for Tempest defined in table 1, $n <= 32$ and $txm <= 24$. We present results in terms of parallel overhead f(P) defined for Parallelism P by

$$f(P) = [PT(P) - P(Ref)T(Ref)] / (P(Ref)T(Ref)) \quad (1)$$

Where we set usually Ref =1 but later we use Ref as the smallest number of processes that can run job efficiently.

The striking result for this step is that MPI easily outperforms the equivalent threaded version of this embarrassingly parallel step. In figure 4, all the peaks in the overhead correspond to patterns with large values of thread count ***t***. On figure 4, we note that MPI intranode 1x24x32 pattern completes the full 624 billion alignments in

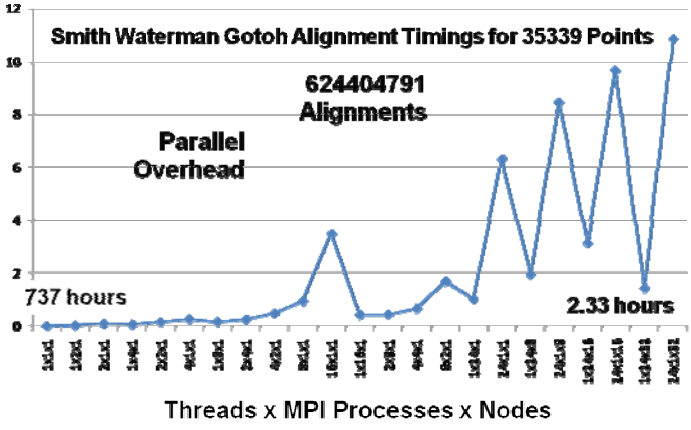

Fig. 4. Performance of Alu Gene Alignments for different parallel patterns

2.33 hours – 4.9 times faster than threaded implementation 24x1x32. This 768 core MPI run has a parallel overhead of 1.43 corresponding to a speed up of 316.

The SW-G alignment performance is probably dominated by memory bandwidth issues but we are still pursuing several points that could affect this but not at our highest priority as SW-G is not a dominant step. We have tried to identify the reason for the comparative slowness of threading. Using Windows monitoring tools, we found that the threaded version has about a factor of 100 more context switches than in case where in MPI we have one thread per process. This could lead to a slow down of threaded approach and correspond to Windows handing of paging of threads with large memory footprints [30]. We have seen this effect in many related circumstances. There is also an important data transfer effect that we discuss in the following subsection.

3.2.2 The O(N^2) Factor of 2 and Data Transfer

There is a well known factor of 2 in many O(N^2) parallel algorithms such as those in direct simulations of astrophysical stems. We initially calculate in parallel the Distance or Dissimilarity D(i,j) between points (sequences) i and j and as discussed above this is done in parallel over all processor nodes selecting criteria $i < j$ to avoid calculating both D(i,j) and the identical D(j,i). This can require substantial file transfer as it is unlikely that nodes requiring D(i,j) in a later step, will find that it was calculated on nodes where it is needed.

For example the MDS and PW(PairWise) Clustering algorithms described in next 2 sections, require a parallel decomposition where each of N processes (MPI processes, threads) has 1/N of sequences and for this subset $\{i\}$ of sequences stores in memory D($\{i\}$,j) for all sequences j and the subset $\{i\}$ of sequences for which this node is responsible. This implies that we need D (i,j) and D (j,i) (which are equal) stored in different processors/disks). This is a well known collective operation in MPI called either gather or scatter. Note that we did NOT get good performance for data transfer of D(i.j) to its needed final processor from either MPI (it should be a seconds

on Petabit/sec Infiniband switch) or Dryad. We intend to make the needed collective (reduction) primitives more precise and expect substantial performance improvement. However, for the results presented here the timings include the I/O necessary to write results from each process to local disk. An additional step was necessary in our processing workflow to combine the results into a single file used in downstream processing such as clustering and MDS.

3.2.3 Use of Dryad in Smith-Waterman Computation

We performed a detailed study [19, 29] of DryadLINQ – Microsoft's implementation of MapReduce [31, 32] for the computation described in previous subsection for MPI. It is nontrivial to produce final output – the D(i,j) in a form suitable for use in next stage of pipeline – this is currently a single final file holding all the independent dissimilarities.

We adopted a coarse grain task decomposition approach [29] which requires minimum inter-process communicational requirements to ameliorate the higher communication and synchronization costs of the DryadLINQ parallel runtime compared to MPI. To explain our algorithm, let's consider an example where N gene sequences produces as discussed above, a pairwise distance matrix of size NxN. We decompose the computation task by considering the resultant matrix and group the overall computation into a blocks by dividing original matrix into BxB subblocks where B is a multiple (>2) of the available computation nodes. As discussed above, due to the symmetry of the distances D(i,j) and D(j,i) we only calculate the distances in the blocks of the upper triangle of the blocked matrix. Diagonal blocks are especially handled and calculated as full sub blocks. As the number of diagonal blocks is B and total number B(B+1)/2, there is no significant compute overhead added by ignoring symmetry in diagonal blocks. The blocks in the upper triangle are partitioned (assigned) to the available compute nodes and an DryadLINQ "Apply" operation is used to execute a function to calculate (N/B)x(N/B) distances in each block. After computing the distances in each block, the function calculates the transpose matrix of the result matrix which corresponds to a block in the lower triangle, and writes both these matrices into two output files in the local file system. The names of these files and their block numbers are communicated back to the main program. The main program sorts the files based on their block number s and perform another DryadLINQ "Apply" operation to combine the files corresponding to a row of blocks in a single large row block.

Figures 5, 6 and 7 present our initial results. Fig. 5 compares the DryadLINQ performance with that of MPI showing the DryadLINQ performance lies between two different versions of the MPI code [29]. In figure 6, we take a fixed Alu dataset of 10,000 sequences and compare its performance as a function of the number of nodes. Note in all these figures we scale results so perfect scaling would correspond to a flat curve independent of the value of the abscissa. Figure 6 only shows some 20% increase in execution time as the core count increases. One problem with current MapReduce implementations is that they are not set up to do dynamic scheduling that help efficiency of a pleasing parallel job mix of inhomogeneous tasks. We examine this in fig. 7 where we artificially increase the scatter in the sequence lengths of the input data. The "real" data shown in figures 5 and 6 has a standard deviation of the

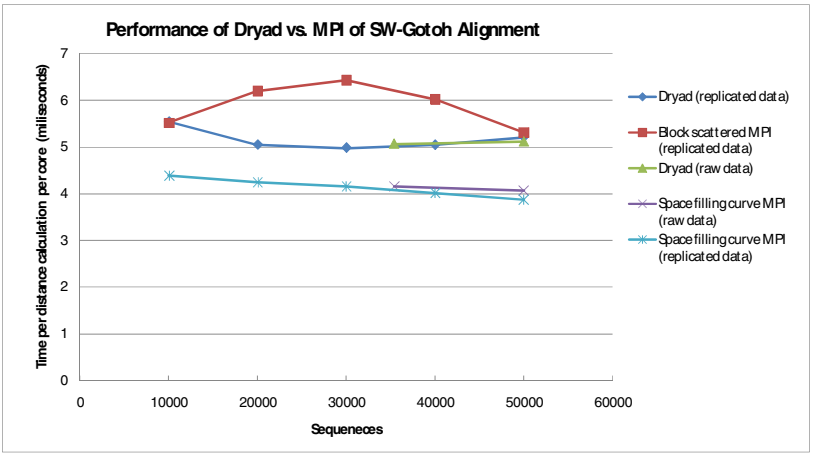

Fig. 5. Comparison of Dryad MapReduce framework with MPI on Smith Waterman Gotoh distance calculations on first step of Alu sequence study pipeline as a function of number of sequences from 10,000 to 50,000. Those "marked replicated" are generated artificially to have uniform inhomogeneity. The raw data for 35339 and 50,000 sequences are also shown.

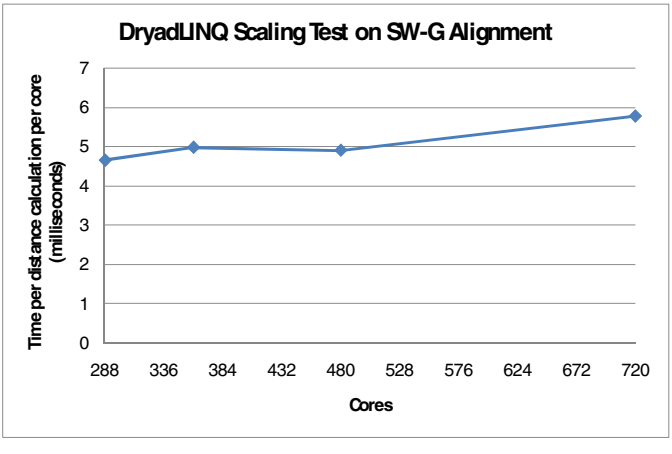

Fig. 6. Scaling test of Dryad MapReduce framework on Smith Waterman Gotoh distance calculations on first step of Alu sequence study pipeline as a function of number of cores

length that is 10% of the mean length. However in figure 7 we increase the standard deviation up to 75% of the mean. By randomizing the sequences in each block we are still able to maintain good performance as seen in fig. 7.

Note that fig. 7 shows both the actual computation and total time including data reorganization for the MPI step. Currently we are extending these results to quantify the comparison between Hadoop [33] and DryadLINQ. In previous work, we have found similar performance between these two MapReduce implementations with Dryad showing somewhat better results [34, 35] in some cases.

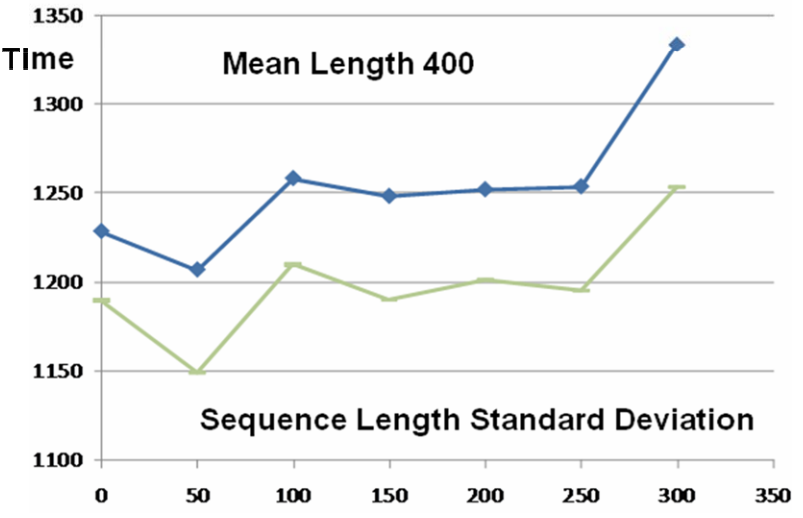

Fig. 7. Study of DryadLINQ processing time on Tempest with statistically load balanced inhomogeneous data with a fixed mean length of 400 characters. The top curve shows the total processing time while the bottom has the final data distribution stage excluded.

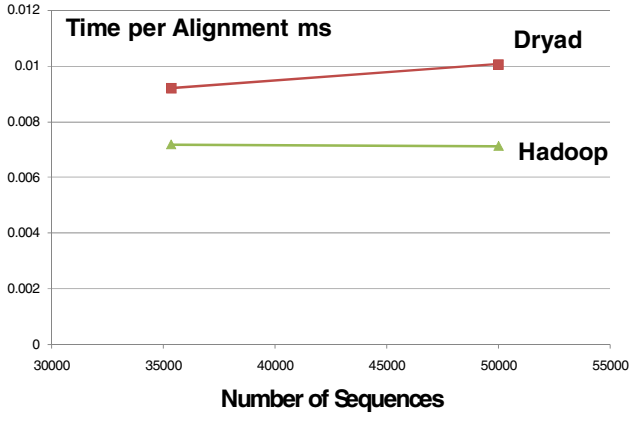

Fig. 8. Study of DryadLINQ (on Windows HPCS) v Hadoop (on RedHat Enterprise Linux) processing Smith Waterman pairwise computations on an IBM IDataplex for samples of 35,339 and 50,000 sequences. The results are presented as time per pair.

We performed initial experiments on an IBM IDataplex with 32 nodes each with 32GB memory and two Intel Xeon L5420 CPU's with 4 cores at 2.50GHz. Surprising the Java alignment code used in Hadoop ran faster (20%) per distance calculation than the C# version and the results in fig. 8 have been corrected for this. Even with this, the Dryad run is somewhat slower than Hadoop in fig. 8.

3.3 Pairwise Clustering

As data sets increase in size, we expect some applications to require particularly robust algorithms that are as insensitive as possible to well known difficulties such as “trapping in local minima”. This increases computing challenge which grows to accommodate data set size and the needed increased robustness of results. For example, clustering methods like Kmeans are very sensitive to false minima but some 20 years ago a more robust EM (Expectation Maximization) method using annealing (deterministic not Monte Carlo) was developed by Ken Rose (UCSB) [1], Fox and others [4]. In this algorithm, the annealing is in distance (as represented by D(i,j)) resolution. One slowly lowers a Temperature T that implements an algorithm sensitive to distance scales of order $T^{0.5}$. This method has the interesting feature that it automatically splits clusters when instabilities detected. Further it has a highly efficient parallel algorithm which we have studied in detail in earlier papers on smaller problems [5]. These clustering approaches are fuzzy methods where points are assigned probabilities for belonging to a particular cluster.

There are striking differences between the parallel pattern dependence of figures 4 and 9 which shows the performance of Pairwise Clustering. In all cases MPI is used as communication mechanism between nodes but we can use any mix of threading and MPI on a single node. For figure 4 intranode MPI always gave best performance but in figure 9, intranode threading is the best at high levels of parallelism but worst at low parallelism. We have analyzed this in detail elsewhere and found it is a consequence of MPI communication overheads that increase as data parallel unit (of

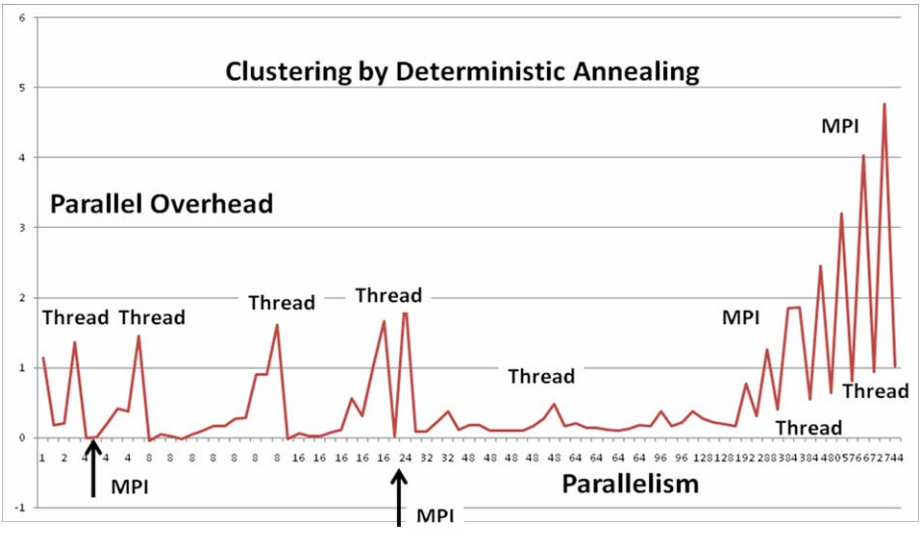

Fig. 9. Paralled Overhead (equation 1) normalized to 4 way parallel MPI job. It is plotted upto 744 way parallel case.

size 35339/(m n)) decreases. For large data parallel units MPI is fastest but for smaller ones used here in production, threading. The poor threaded performance for low levels of parallelism is due to context switches for large memory jobs discussed in section 3.2.1.

The original clustering work was based in a vector space (like Kmeans) where a cluster is defined by a vector as its center. However in a major advance 10 years ago [2, 3], it was shown that one could use a vector free approach and operate with just the dissimilarities D(i,j). This unfortunately does increase the computational complexity from O(N) to O(N^2) for N sequences. It appears however more natural and even essential for studies of gene sequences which do not have Euclidean vectors easily associated with them. We completed these pairwise vector free algorithms and implemented them in parallel. We have discussed elsewhere [5] detailed algorithm and performance issues. Here we report the clustering as part of a large end to end component of our "Million Sequence Analysis as a Service project". All capabilities discussed in this paper will be made available as cloud or TeraGrid services over the next year.

3.4 Multidimensional Scaling MDS

Given dissimilarities D(i,j), MDS finds the best set of vectors $\underline{\mathbf{x}}_j$ in any chosen dimension d (d=3 in our case) minimizing:

$$\Sigma_{i,j} \text{ weight}(i,j) \left(D(i,j)^m - |\underline{\mathbf{x}}_i - \mathbf{x}_j|^n\right)^2 \quad (2)$$

The form of the weight(i,j) is chosen to reflect importance of a point or perhaps a desire (Sammon's method with weight(i,j)=1/ D(i,j)) to fit smaller distance more precisely than larger ones. The index n is typically 1 (Euclidean distance) but 2 also useful. m is 1 in this paper but m=0.5 is interesting.

We have previously reported results using Expectation Maximization and we are exploring adding to this deterministic annealing to improve robustness. Here we use a different technique exploiting that (2) is "just" χ^2 and one can use very reliable nonlinear optimizers to solve it [20]. We have implemented and got good results with the Levenberg–Marquardt approach (adding suitable multiple of unit matrix to nonlinear second derivative matrix) to χ^2 solution.

This "MDS as χ^2" approach allows us to incorporate some powerful features including very general choices for the weight(i,j) and n. Our MDS service is fully parallel over unknowns $\underline{\mathbf{x}}_i$. Further it allows "incremental use"; fixing an MDS solution from a subset of data and adding new points at a later time. One can also optimally align different versions of MDS (e.g. different choices of weight(i,j) to allow precise comparisons. All our MDS services feed their results directly to powerful Point Visualizer. Figure 10 shows the end to end Alu study after SW-G alignments, pairwise clustering and MDS projection. One sees three small clusters red (2794 points), yellow (3666) and green (1838 sequences) isolated from larger (27041) collection of blue sequences that are presumably older. Note that total time for all 3 steps on the full Tempest system is about 6 hours and clearly getting to a million sequences is not unrealistic and would take around a week on a 1024 node cluster.

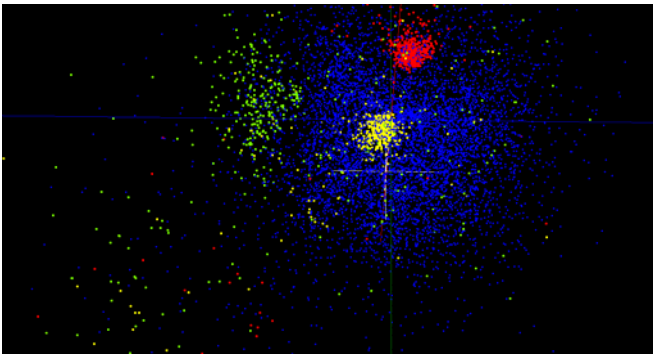

Fig. 10. Pairwise Clustering of 35339 Alu Sequences visualized with MDS

4 Linking Environment and Health Data

4.1 Introduction

Another area where our tools are naturally used comes in Geographical information systems where we have already presented results [21]. Here we link environmental and patient (health) data. This is challenging as a community's vulnerability and impact may depend on special concerns like environmentally sensitive areas or historical structures, socioeconomic conditions, and various social concerns such as the degree of public trust, education levels, literacy, and collective action and solidarity. The event impact must account for a blend of physical and social measures.

One example is the SAVI Community Information System (www.savi.org)1 is one of the nation's largest community information systems [22]. SAVI, designed to improve decision-making in Central Indiana communities, includes over a ~22 million individual data values, and provides over 161,322 event datasets, 3,099 basic indicators on the socio-economic conditions, health, economy, housing, and many other aspects of the community. Further it makes them available for 11 types of geographic areas, such as census tracts, neighborhoods, and school corporations. The SAVI system is now being used by a variety of other sectors for community development, public health research, education, program planning, disaster mitigation planning and more. Only recently has the field of social epidemiology begun to develop the theoretical tools that make possible the identification of explanatory pathways from the physical and social infrastructure to health-related behaviors, which then lead to adverse health outcomes [23-25]. We see geographic clustering [21, 36] in many health outcomes because social environment has an effect on health and/or health behaviors [26-28].

4.2 Correlating Environment and Patient Data

We used an ongoing childhood obesity study as our first application to test the relevance of our tools in the area of linking environment and social/health data. [6-7] Obesity is presently one of the most pervasive, serious, and challenging health problems facing the world. Over the past 30 years, the obesity rate has nearly tripled

for children ages 2 to 5 years (from 5 to 14 percent) and tripled for youth ages 12 to 19 years (from 5 percent to 17 percent). The obesity rate for children 6 to 11 years of age has quadrupled from 4 to 19 percent. What is causing the dramatic and threatening rise in obesity? Bray concisely captured the etiology of obesity in metaphor: "Genes load the gun, the environment pulls the trigger." 23 Genetic factors are thought to account for 25-40% of the variance in BMI (Body Mass Index) by determining differences in such things as resting metabolic rate and weight gain in response to overfeeding. However, it is highly improbable that changes in genetic factors explain the rapid increases in obesity prevalence over the past two decades. [26] Rather the obesity epidemic is almost certainly rooted in environmental factors that promote excessive caloric intake and sedentary lifestyle [8].

In addition to physical environmental factors, social environmental factors also have bearing on obesity by facilitating or constraining behavior. Specific social environmental factors that have been examined include crime, safety, social support, social networks, and neighborhood socioeconomic status. Perceived (or actual) lack of a safe environment is a significant barrier to physical activity. According to a study conducted by the Centers for Disease Control in 2004, persons who perceived their neighborhoods as less than extremely safe were more than twice as likely to have no leisure-time physical activity, and those who perceived their neighborhoods as not at all safe were nearly three times as likely to have no leisure-time physical activity. Research also indicates that parental concerns about traffic and crime have a strong influence on children's physical activity levels and that child and parent perceptions of the environment are as important as the actual environment.

This motivates studies that study linkage between patient health and environment factors. We can examine urban planning data that provides information on characteristics of the built environment, such as street features, land use mix, and neighborhood greenness. We examine insurance information from patient medical records as an indicator of family-level social environment. We examine U.S. Census and Uniform Crime Report information for areas surrounding patients' residential addresses as indicators of neighborhood social environment. Here we are setting up the infrastructure linking the tool R with our other tools described in section 3 and only have preliminary results on this use case for our new generation of large scale data analysis tools. As there are some 30 patient attributes and over one hundred environmental attributes, tools like MDS that reduce dimensionality were a focus.

4.3 Canonical Correlation Analysis and Multidimensional Scaling

The canonical correlation analysis (CCA) is a tool of multivariate statistical analysis for finding correlations between two sets of variables [9, 10]. Here we are applying CCA to correlate patient health and environmental factors. Our full data set we used for this research consists of over 314,000 real-life patient records collected over 15 years and measured on about 180 variables, mostly related with biological and environmental factors. We stored our full data set (with size 832 MB) in a database system for easy exploration and fast extraction. Among the full data set, we only used the cleanest data for our initial studies. For performing CCA over the patient data set and conducting various kinds of statistical analysis, we used R, one of the most well-known statistical computing environments. R expedites complicated statistical data

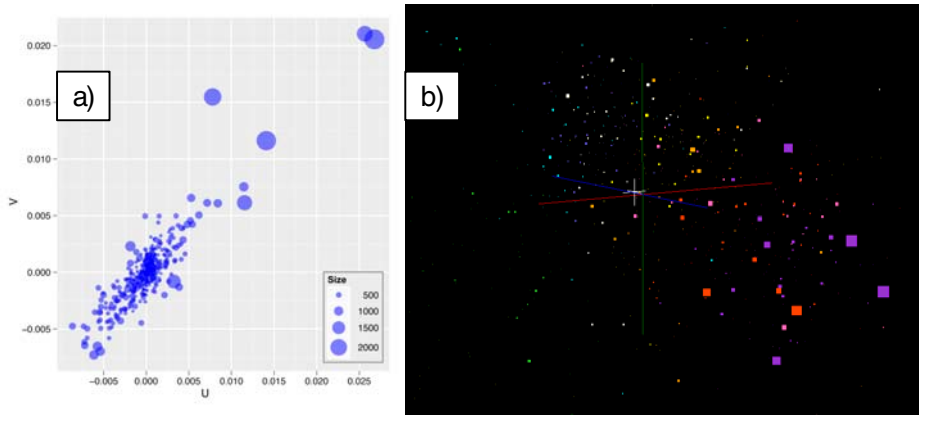

Fig. 11. a) The plot of the first pair of canonical variables for 635 Census Blocks and b) the color coded correlation between MDS and first eigenvector of PCA decomposition

manipulations with ease by utilizing highly optimized and multi-threaded numeric packages, such as BLAS, Goto-BLAS [11], and ATLAS [12]. Another advantage in using R is that we can use various open-source add-on packages for additional functionalities. For example, with the help of packages for databases, such as PostgreSQL and MySQL, we can directly access the data stored in the database system.

The core idea in CCA is to find an optimal linear projection of two sets of data in a sense that the correlation of them in the projected space, also called "canonical space", is maximized. More specifically, for the given two sets of data matrix X and Y, the CCA seeks two optimal projection vectors $\underline{a}$ and $\underline{b}$, which make the following correlation maximum:

$$\rho = \mathrm{corr}(\underline{U}, \underline{V}) ,$$

where $\underline{U} = \underline{a}^T X$ and $\underline{V} = \underline{b}^T Y$ are vectors in the canonical space. One can see that the vector $\underline{U}$ and $\underline{V}$, known as *canonical correlation variables*, are the new representation of the data matrix X and Y in the canonical space, transformed by the projection vector $\underline{a}$ and $\underline{b}$ respectively..

In our project, the CCA is a good match as we have two sets of data – patient and environmental data – and want to find out which variables of environmental data have some connections to patient's obesity or more generally their health. For this purpose, we can use X as an environmental data and Y as a patient data with the CCA formalism to find the best optimal canonical variables $\underline{U}$ and $\underline{V}$, which maximize the correlation between the patient and the environmental data. As an alternative to CCA, which maximizes vector in both data sets, one can find the vectors $\underline{a}$ and $\underline{b}$ by fixing the vector in one sector. For example with our health data set, we can find new projection vector $\underline{a}$ by fixing $\underline{b}$ in terms of Principle Components (PC) of the patient data matrix Y.

Since the well known CCA algorithm itself is not our focus in this paper, we will not present more details in As an example of CCA results to the patient data set, we

found the optimal correlation in the canonical space (Figure 11a)). Those results can feed in to the MDS to find more robust structures in 3-dimension (Figure 11b). More details can be found in [9, 10, 30]. Each point corresponds to one of 635 Census blocks. We color projections on a green (lowest) to red/mauve (highest) scale and see clear clustering of the different colors in different regions of MDS. The low (green) values occur together and are well separated from the high weighted red and mauve points. In these plots the MDS was weighted (using weight(i,j) in equation (2)) proportional to the number of patients in block. Figures 11b) show correlations for a pure principal component analysis PCA and similar results are seen for the optimal CCA vector $\underline{U}$. The correlation between PCA eigenvector and MDS decomposition was 0.86 (using CCA applied to MDS and environmental data) where as correlation is 0.67 between MNDS and the optimal vector $\underline{U}$ from patient analysis.

In processing CCA in our project, we have used R as statistical computing environments to utilize various matrix manipulation and linear algebra packages with efficiency. Also, by building R with multi-threaded enabled BLAS libraries, we got parallel speed up in our 8 core Linux cluster nodes "Gridfarm". We are continuing these studies on Tempest using a Genetic algorithm to find the optimal set of environment features and this type of loosely coupled problem is suitable for clouds and we will look at an Hadoop/Dryad implementation. We believe linking of R and MapReduce will be very powerful as is the current MPI link.

5 Conclusions

This paper examines the technology to support rapid analysis of genome sequencing and patient record problems that typify today's high end biomedical computational challenges. As well as our local sample problems, we would like to refine and test the technology on a broader range of problems. To encourage this, we will make key capabilities available as services that we eventually be implemented on virtual clusters (clouds) to address very large problems. Relevant services we will make available include the basic Pairwise dissimilarity calculations, R (done already by us and others), MDS in EM and χ^2 forms; the vector and pairwise deterministic annealing clustering including support of fingerprints and other "unusual" vectors. Our point viewer (Plotviz) will be made available either as download (to Windows!) or as a Web service. We note all our current code is written in C# (high performance managed code) and runs on Microsoft HPCS 2008 (with Dryad extensions).

Cloud technologies such as Hadoop and Dryad are very attractive as robust commercially supported software environments. Further many services will run on virtual machine based clouds such as Azure and EC2. In our two problems, we have found steps that currently need MPI. These run well but do not have the intrinsic flexibility and robustness of MapReduce. We note that our MPI applications are not like traditional particle dynamics or differential equation solvers with many small messages. Rather they can be implemented efficiently with only MPI Barrier, Broadcast and Reduction operations which are typical of linear algebra. We are exploring [34, 35] extensions of MapReduce – we call them MapReduce++ -- that support the iterative structure of parallel linear algebra. We expect these to be an

attractive implementation paradigm for biomedical applications and allow easy porting of our current MPI codes.

In summary, we've shown two examples of data intensive science applications in area of biology and health using several modern technologies. We suggest that these ideas will support new generations of large scale data analysis systems for patient records, demographic data and next generation gene sequencers.

References

1. Rose, K.: Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems. Proceedings of the IEEE 80, 2210–2239 (1998)
2. Hofmann, T., Buhmann, J.M.: Pairwise data clustering by deterministic annealing. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 1–13 (1997)
3. Klock, H., Buhmann, J.M.: Data visualization by multidimensional scaling: a deterministic annealing approach. Pattern Recognition 33(4), 651–669 (2000)
4. Granat, R.A.: Regularized Deterministic Annealing EM for Hidden Markov Models, Ph.D. Thesis, UCLA (2004)
5. Fox, G., Bae, S.-H., Ekanayake, J., Qiu, X., Yuan, H.: Parallel Data Mining from Multicore to Cloudy Grids. In: Proceedings of HPC 2008, High Performance Computing and Grids Workshop, Cetraro Italy, July 3 (2008)
6. Liu, G., Wilson, J., Rong, Q., Ying, J.: Green neighborhoods, food retail, and childhood overweight: differences by population density. American Journal of Health Promotion 21(I4 suppl.), 317–325 (2007)
7. Liu, G., et al.: Examining Urban Environment Correlates of Childhood Physical Activity and Walkability Perception with GIS and Remote Sensing. In: Geo-spatial Technologies in Urban Environments Policy, Practice, and Pixels, 2nd edn., pp. 121–140. Springer, Berlin (2007)
8. Sandy, R., Liu, G., et al.: Studying the child obesity epidemic with natural experiments, NBER Working Paper in (May 2009), `http://www.nber.org/papers/w14989`
9. Hardoon, D., et al.: Canonical correlation analysis: an overview with application to learning methods. Neural Computation 16(12), 2639–2664 (2004)
10. Härdle, W., Simar, L.: Applied multivariate statistical analysis, pp. 361–372. Springer, Heidelberg (2007)
11. Goto, K., Van De Geijn, R.: High-performance implementation of the level-3 blas. ACM Trans. Math. Softw. 35(1), 1–14 (2008)
12. Whaley, R., Dongarra, J.: Automatically tuned linear algebra software. In: Proceedings of the 1998 ACM/IEEE conf. on Supercomputing (CDROM), pp. 1–27 (1998)
13. Batzer, M.A., Deininger, P.L.: Alu repeats and human genomic diversity. Nat. Rev. Genet. 3(5), 370–379 (2002)
14. Smit, A.F.A., Hubley, R., Green, P.: Repeatmasker (2004), `http://www.repeatmasker.org`
15. Jurka, J.: Repbase Update: a database and electronic journal of repetitive elements. Trends Genet. 9, 418–420 (2000)
16. Waterman, S.: Software with Gotoh enhancement, `http://jaligner.sourceforge.net/naligner/`
17. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology 147, 195–197 (1981)

18. Gotoh, O.: An improved algorithm for matching biological sequences. J. of Molecular Biology 162, 705–708 (1982)
19. Ekanayake, J., Balkir, A.S., Gunarathne, T., Fox, G., Poulain, C., Araujo, N., Barga, R.: DryadLINQ for Scientific Analyses. In: Proceedings of eScience conference (2009), http://grids.ucs.indiana.edu/ptliupages/publications/DryadLINQ_for_Scientific_Analyses.pdf
20. Kearsley, A.J., Tapia, R.A., Trosset, M.W.: The Solution of the Metric STRESS and SSTRESS Problems in Multidimensional Scaling Using Newton's Method, technical report (1995)
21. Qiu, X., Fox, G.C., Yuan, H., Bae, S.-H., Chrysanthakopoulos, G., Nielsen, H.F.: Parallel Clustering and Dimensional Scaling on Multicore System. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 407–416. Springer, Heidelberg (2008)
22. Frederickson, K.E.: Enhanced Local Coordination and Collaboration through the Social Assets and Vulnerabilities Indicators (SAVI) Project. In: Proceedings of the American Public Health Association Annual Conference, Washington, D.C (1998)
23. American Public Health Association, National Public Health Week, Eliminating Health Disparities: Communities Moving from Statistics to Solutions, Toolkit (2004)
24. Berkman, L.F., Glass, T.: Social integration, social networks, social support, and health. In: Berkman, L.F., Kawachi, I. (eds.) Social Epidemiology, pp. 137–173. Oxford University Press, New York (2000)
25. Shaw, M., Dorling, D., Smith, G.D.: Poverty, social exclusion, and minorities. In: Marmot, M., Wilkinson, R.G. (eds.) Social Determinants of Health, 2nd edn., pp. 196–223. Oxford University Press, New York (2006)
26. Berkman, L.F., Kawachi, I.: A historical framework for social epidemiology. In: Berkman, L.F., Kawachi, I. (eds.) Social Epidemiology, pp. 3–12. Oxford Univ. Press, New York (2000)
27. Kawachi, I., Berkman, L.F. (eds.): Neighborhoods and Health. Oxford University Press, New York (2003)
28. Robert, S.: Community-level socioeconomic status effects on adult health. Journal of Health and Social Behavior 39, 18–37 (1998)
29. Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., Gannon, D.: Cloud Technologies for Bioinformatics Applications. In: 2nd ACM Workshop on Many-Task Computing on Grids and Supercomputers (SuperComputing 2009), Portland, Oregon, November 16 (2009), http://grids.ucs.indiana.edu/ptliupages/publications/MTAGS09-23.pdf
30. Fox, G., Qiu, X., Beason, S., Choi, J.Y., Rho, M., Tang, H., Devadasan, N., Liu, G.: Case Studies in Data Intensive Computing: Large Scale DNA Sequence Analysis as the Million Sequence Challenge and Biomedical Computing Technical Report, August 9 (2009), http://grids.ucs.indiana.edu/ptliupages/publications/UsesCasesforDIC-Aug%209-09.pdf
31. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: Distributed data-parallel programs from sequential building blocks. In: European Conference on Computer Systems (March 2007)
32. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, Ú., Gunda, P., Currey, J.: DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. In: Symposium on Operating System Design and Implementation (OSDI), CA, December 8-10 (2008)
33. Apache Hadoop, http://hadoop.apache.org/core/

34. Ekanayake, J., Qiu, X., Gunarathne, T., Beason, S., Fox, G.: High Performance Parallel Computing with Clouds and Cloud Technologies (August 25, 2009) (to be published as book chapter),
http://grids.ucs.indiana.edu/ptliupages/publications/cloud_handbook_final-with-diagrams.pdf
35. Ekanayake, J., Fox, G.: High Performance Parallel Computing with Clouds and Cloud Technologies. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, Springer, Heidelberg (2009),
http://grids.ucs.indiana.edu/ptliupages/publications/cloudcomp_camera_ready.pdf
36. Qiu, X., Fox, G.C., Yuan, H., Bae, S.-H., Chrysanthakopoulos, G., Nielsen, H.F.: Parallel Clustering And Dimensional Scaling on Multicore Systems. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 407–416. Springer, Heidelberg (2008),
http://grids.ucs.indiana.edu/ptliupages/publications/hpcsApril12-08.pdf