Finding Protein Binding Sites Using Volunteer Computing Grids

Travis Desell, Lee A. Newberg, Malik Magdon-Ismail, Boleslaw K. Szymanski, and William Thompson

Abstract This paper describes initial work in the development of the DNA@Home volunteer computing project, which aims to use Gibbs sampling for the identification and location of DNA control signals on full genome scale data sets. Most current research involving sequence analysis for these control signals involve significantly smaller data sets, however volunteer computing can provide the necessary computational power to make full genome analysis feasible. A fault tolerant and asynchronous implementation of Gibbs sampling using the Berkeley Open Infrastructure for Network Computing (BOINC) is presented, which is currently being used to analyze the intergenic regions of the *Mycobacterium tuberculosis* genome. In only three months of limited operation, the project has had over 1,800 volunteered computing hosts participate and obtains a number of samples required for analysis over 400 times faster than an average computing host for the *Mycobacterium tuberculosis* dataset. We feel that the preliminary results for this project provide a strong argument for the feasibility and public interest of a volunteer computing project for this type of bioinformatics.

1 Introduction

Cutting edge computational science is requiring larger and larger computing systems as the size and complexity of scientific data continues to increase far faster than advances in processor speeds. This is particularly true in the area of computa-

Travis Desell
University of North Dakota, Grand Forks, North Dakota, USA, e-mail: tdesell@cs.und.edu

Lee A. Newberg, Malik Magdon-Ismail and Boleslaw K. Szymanski
RPI, Troy, New York, USA, e-mail: [leen,magdon,szymansk]@cs.rpi.edu

William Thompson
Center for Computational Molecular Biology, Department of Applied Mathematics, Brown University, Providence, Rhode Island, USA e-mail: william_thompson_1@brown.edu

tional biology, as biologists are gathering data on the full genomes of many different species. In the domain of biological sequence analysis, high-dimensional integration allows the identification and location of DNA control signals (*cis*-regulatory elements such as protein binding sites) in analyses of dozens of co-regulated genes at a time. Identifying and locating these *cis*-regulatory elements at the genome and multi-genome scales continues to be too complex for today's clusters and clouds. However, emerging peta-scale systems (those providing petaFLOPS, millions of billions of FLoating point OPerations per Second, of computational power) can enable this computationally challenging research, which will add significantly to understanding of the cellular processes of a diverse set of organisms, including organisms for disease, biofuel production, and environmental bioremediation.

Apart from a few of the world's fastest supercomputers[1], volunteer computing projects such as Stanford's Folding@HOME [1] and the Berkeley Open Infrastructure for Network Computing (BOINC) [2] have also reached peta-scale levels of computing power. BOINC in particular is a very interesting computing environment for new scientific projects, in that it provides an open source environment for developing projects and has a highly active social community which actively seeks out new projects to participate in. Further, there is very low overhead to starting a BOINC project, as purchasing and maintaining a server machine is a fraction of cost of purchasing and maintaining supercomputer, and many volunteers actively upgrade and purchase new equipment while the hardware in a supercomputer remains fixed and degrades over time.

However, traditional numerical integration methods are highly sequential, relying on variations of Monte-Carlo Markov Chains (MCMC), which makes them not well suited to volunteer computing environments, which are inherently heterogeneous and can be extremely volatile with volunteered hosts frequently joining and leaving, and potentially returning malicious results. Current parallel approaches to MCMC are not fault tolerant and cannot scale to the number processors used by peta-scale systems. Additionally, many peta-scale systems, including the world's fastest supercomputer and many BOINC computing projects, are accomplishing these speeds by using graphical processing units (GPUs). However, utilizing GPUs for full genome analysis proves problematic, as performing computation requiring full genome scale data may not fit into a GPUs limited memory.

This paper presents preliminary work done in the development of DNA@Home, a BOINC volunteer computing project started with the goal of performing sequence analysis at the full genome level. The project extends the Gibbs Sampling algorithm, an MCMC approach for finding transcription factors [3, 4, 5, 6] which also has uses in other scientific disciplines [7, 8, 9, 10], by adding parallelism and fault tolerance making it suitable for use on a volunteer computing grid. This asynchronous Gibbs Sampling approach is currently being used by DNA@Home to analyze the full genome of *Mycobacterium tuberculosis* consisting of 2,066 intergenic regions (segments of DNA between known genes) totaling 350,825 nucleotides. Volunteered hosts report a set of samples at a rate of approximately 0.6 every second, with a sam-

[1] http://www.top500.org/

pling task taking on average 22.4 minutes – which equates to a 896 times speedup over a single average speed CPU. After only three months of operation, more than 1,800 volunteered hosts have participated in the project with over 1,600 staying active since joining, which highlights the potential for this approach to reach the scales required to look for transcription factors using the full genomes of complex prokaryotes (bacteria) and eukaryotes (even humans).

The paper proceeds as follows. Section 2 describes Gibbs Sampling and the challenges involved in using it to analyze large scale data. Section 3 describes how Gibbs Sampling was extended for use on volunteer computing grids. Section 4 provides preliminary results describing the performance and activity of DNA@Home. Finally, concluding remarks and future work are discussed in Section 5.

2 Gibbs Sampling for Finding Transcription Factors

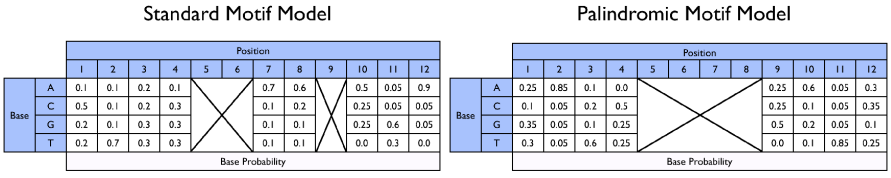

Fig. 1 *Motif models* for different types of transcription factor binding sites (sites where proteins bind to DNA). Transcription factors are non-exact which makes them computationally demanding to find. These binding sites need to be modeled probabilistically. Additionally, certain positions within the binding site may be non-binding (represented as 'X's in the figures). One goal of this research is to develop an asynchronous Gibbs sampling algorithm that can use massive scale cyber-infrastructure to find these binding sites within multiple full genomes, something not possible with current cyber-infrastructure.

Gibbs sampling is used by computational biologists to find transcription factor binding sites or gene regulatory elements – sites where proteins bind to DNA [11, 12]. It is a variant of Markov Chain Monte-Carlo (MCMC), where every step of the random walk must satisfy the following criteria:

$$P_i * R_{i,j} = P_j * R_{j,i} \tag{1}$$

where P_i is the probability of state i being a solution, and P_j is the probability of state j being a solution. It is sufficient for P_i and P_j to be relatively correct, as opposed to the exact probability, as this is typically unknown, while the relative probabilities of two states can be calculated more easily. $R_{i,j}$ and $R_{j,i}$ are the *transition probabilities*, or the probability that the state will move from i to j and vice versa. Fulfilling this detailed balance equation (Equation 1) ensures that a long

enough Markov chain will consist of states sampled in proportion to the probability distribution of the function being integrated.

The problem state of finding transcription factor binding sites consists of multiple *motif models*, probabilistic representations of DNA patterns, of varying lengths (see Figure 1). A transcription factor can bind to different sequences of DNA, so binding sites are non-exact. A motif model represents the different probabilities of each position within the motif being a DNA letter. Additionally, motif models can have skipped positions (represented by 'X's in Figure 1) and be palindromic, where tail of the motif model is the reverse complement of the front.

There are two main computational challenges involved in using Gibbs sampling to find transcription factors binding sites. First, determining the next state in the random walk can be computationally demanding. For prokaryotes, typical parameter sizes computable by current cyber infrastructure consist of less than five motifs, 12 to 24 nucleotides wide, being sampled from within 3 to 30 intergenic regions of less than 500 nucleotides, usually from one species. Eukaryotes have shorter motif sizes, 6 to 12 base pairs, however the intergenic regions are larger, 1,000 to 10,000 nucleotides. Each step requires that for every intergenic region, all the motifs must be regenerated from the samples within the other intergenic regions, and then these new motif models are used to resample within the excluded intergenic region. Other parameters that are modified during the random walks are the length of the motifs and positions of the skipped positions. Because of this, increasing the motif size, the number of motifs, intergenic regions and the region size will all increase the computational complexity.

Second, there is no known way to start the random walk from an unbiased initial position. To avoid bias in the samples taken by the random walk, Gibbs sampling performs a *burn-in* period, where the states visited by the random walk are not used as samples. Following the burn-in period, a sample can be taken after every step in the random walk. There are two main problems within the burn-in and sampling periods. First, the length of the burn-in period required to eliminate any bias in the selection of the starting state is unknown, as is the number of samples required to adequately capture the problem space. There have been a few approaches to address this issue [13, 14], but this is still an open problem. Further, as the problem size increases, the increase in burn-in period and number of samples required also increases super-linearly, at the very least.

Because of these problems, being able to perform full genome scale analysis requires massive amounts of computing power, which we feel can be effectively provided by a volunteer computing project such as DNA@Home.

3 Gibbs Sampling on Volunteer Computing Grids

Effectively distributing Gibbs sampling is non-trivial, as Markov Chain random walks are inherently sequential. This makes it difficult to effectively utilize the computing power offered by massive scale computing systems, like volunteer computing

grids. The application used for computation on the volunteered computing hosts is a modification of the Gibbs sampling algorithm described in Section 2 that performs partial random walks, which are chained by the server-side *parallel walk sampling* software. Section 3.1 describes how the standard Gibbs sampling algorithm has been extended to use asynchronous communication and to provide fault tolerance and error checking. This *parallel walk sampling* enables the computational biologists participating in DNA@Home to utilize these massive scale computing environments. Section 3.2 gives details on how this was implemented and optimized for use with BOINC's open source software.

3.1 Parallel Walk Sampling

While sampling using parallel walks does not reduce the burn-in time, it is an effective way of reducing the sampling time, providing linear speedup in the number of samples generated after the walks have completed their burn-in period. In parallel walk sampling, each Gibbs sampling walk starts from an independent state, then after they have completed the burn-in period to eliminate the initial bias, samples are collected in parallel. For full genome scale data, an extremely large number of samples is required to gather an accurate picture of the sampling space to find these transcription factors, so having a large number of parallel walks being computing by volunteer computing hosts is an effective way of gathering enough samples in a reasonable amount of time.

However, at the massive computing scale simply running the Gibbs sampling walks in parallel is not sufficient as any processor failure will lose the burn-in steps and any samples collected, wasting all that work. It also does not take into account processors dynamically joining and leaving in the case of a volunteer project. Because of this, DNA@Home uses an asynchronous approach to perform the parallel walks, as shown in Figure 2. The BOINC server stores the last reported position of each walk along with any reported samples. Host processors perform partial walks, by contacting the BOINC server and requesting an initial starting state. The server will also specify a starting seed for randomization and the length of the walk the host will compute. This will allow the server to send the same partial walk to multiple hosts for DNA@Home so results can be compared from different hosts for validation. In this way, individual processors can fail or leave and the server can send that particular processors partial walk to another host; significantly reducing the impact of failures.

3.2 BOINC Implementation Details

The BOINC architecture (see Figure 3) consists of a set of server-side daemons which control the generation, scheduling and validation of tasks, or *workunits*, and

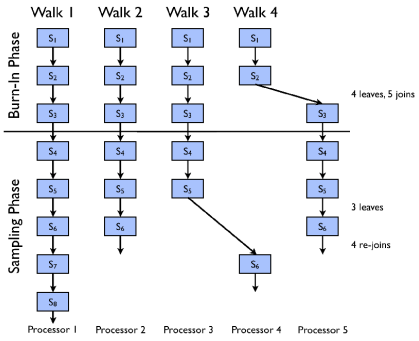

Fig. 2 A strategy for Gibbs sampling using parallel walks. Each arrow represents a *workunit*, or processing job, where a processor receives an initial state with depth x, S_x, and reports a final state with depth y, S_y. Each workunit has a fixed length walk (in this case 1). After each walk completes its burn-in period, samples can be taken. Processors can join and leave, restarting from walks of previously left processors.

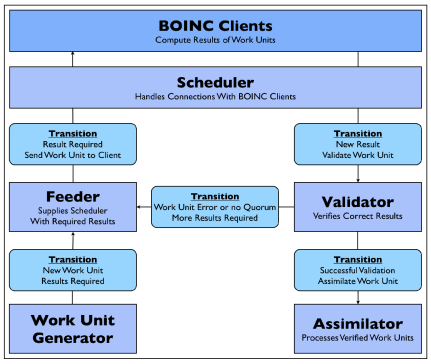

Fig. 3 The BOINC architecture consists of multiple server-side daemons which handle work generation, validation and scheduling of work to clients. Tasks that clients will be or are currently computing are called *workunits* and clients report *results* to those workunits. A transitioner daemon updates the state of the workunits and results in the database, which triggers the other daemons to act on them.

the individual *results* from clients. After a workunit is generated by a *work generator* daemon, the *transitioner* daemon updates the database with state changes for that workunit and its results; which trigger the other daemons. The *feeder* controls what workunits are available to be sent to clients, while the *scheduler* determines which clients should receive what workunits. The *validator* compares results to ensure that they are correct, after which the *assimilator* handles the processing of valid results.

The parallel walk sampling strategy used by DNA@Home combines the assimilator and work generator into a single daemon. A database is used to store the parameters to multiple searches which can be performed concurrently, each consisting of any number of motif models of any given size and type (*e*.g. standard, reverse complement or palindromic) and a given dataset (*e*.g. *Mycobacterium tuberculosis* or *Yersinia pestis*). When a search is started, a fixed number of walks are created, each with a randomly selected set of sampled positions within the intergenic sequences. While burn-in is being performed on a particular walk, the results of that workunit will report the final positions walked to by the Gibbs sampling algorithm. When the results for a workunit are validated (if two or more separate hosts report the same final sample positions those results are considered valid), a new workunit is generated which includes those final positions as a new starting position. The current position of that walk is updated in the database, and its burn-in depth is increased.

After a walk has completed its burn-in period, a flag is set telling the workunits to additionally report every site sampled during its random walk. When the results for one of these sampling workunits are validated (when the final positions and

accumulated samples of two or more results are the same), the server saves the accumulated samples and generates a new sampling workunit from the new end position, updating the number of samples taken by that walk in the database.

Various flags have also been specified for these workunits to enable more reliability in the turnaround time for any given workunit. While the number of results required for validation (or *quorum*) is two, BOINC's `target_nresults` flag has been set to three for every workunit. This causes the scheduler to send out the same task to three hosts initially (instead of two), which reduces the impact of one of those hosts returning an erroneous result, or from leaving the project and never returning a result.

4 Preliminary Results

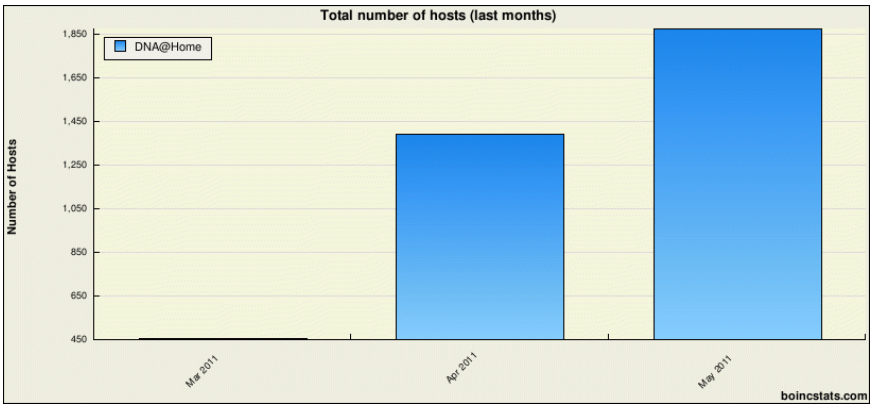

Fig. 4 The number of hosts that have participated in DNA@Home since it began sending work to volunteers in March, 2011, as recorded by the boincstats.com project tracking website. In only three months, DNA@Home has had over 1,600 hosts participate, with new hosts joining daily.

After only three months of limited operation, with small batches of workunits sent out periodically, DNA@Home has already had over 1,800 hosts participate and return valid results to the project (Figure 4 shows the increase in host participation since DNA@Home started sending out workunits)[2]. Binary versions of the application run on volunteered hosts is provided for both 32 and 64 bit versions of Windows, Mac OS X and Linux, which allows any type of computer to participate in the project. These volunteers were only 'recruited' through word of mouth and DNA@Home's listing on various BOINC project tracking websites. We feel that this provides a significant example of the public interest in this type of project,

[2] http://boincstats.com/stats/project_graph.php?prdna&view=hosts

and its ability to achieve the required computational power for full genome scale analysis.

DNA@Home is currently using a dataset consisting of the intergenic regions of the *Mycobacterium tuberculosis* genome. In this dataset there are 2,066 intergenic regions, consisting of a total of 350,825 nucleotides. With approximately 1,600 active volunteers, it takes less than a week to gather 30,000,000 samples after a burn-in of 1,000,000 steps using 3,000 parallel walks, using a reverse complement motif model of length 16. A workunit consists of 10,000 steps, which over the set of all volunteers takes on average 22.4 minutes. Approximately 0.6 results are reported every second, for a speedup of 896 times an average processor. Alternately, to perform a burn-in of 1,000,000 steps and 30,000,000 samples on an average processor would take 2,893 days to complete, so DNA@Home can obtain a useful amount of samples over 400 times faster.

5 Conclusions and Future Work

This paper presents an asynchronous and fault tolerant implementation of parallel Gibbs sampling for use on the DNA@Home volunteer computing project. Preliminary results show DNA@Home gathering a required number of samples for analysis of the intergenic regions of the full *Mycobacterium tuberculosis* genome in less than a week, a speedup of over 400 times an average single processor. Additionally, with only limited operation over three months, the DNA@Home project has already grown to over 1,600 active volunteers, contributing 1,800 computing hosts, highlighting the public interest and potential for this type of project to analyze even larger genomes from more complex bacteria or even humans. Further, the implications of this research go beyond DNA sequence analysis, as Gibbs sampling is popular and efficient way to sample from complex probability distributions that are not easily reduced to the common tractable probability distributions. It thus has general applicability in statistics where it allows one to sample from a (Bayesian) posterior probability distribution. It also has immediate, and fairly generally applicable, uses in numerical integration where it is almost always significantly more efficient than approaches that approximate a multi-dimensional integral by evaluating the integrand at evenly spaced points or at points chosen uniformly at random.

Future research for the DNA@Home project involves development of a graphical processing unit (GPU) version of the Gibbs sampling application, which has been attempted by other groups [15, 16], as these provide a significant amount of the computational power of the BOINC computing system. Additionally, the development of a web-based interface for participating biologists would ease the use of the system for large number of researchers. Finally, we wish to investigate asynchronous methods for decreasing the time to complete the burn-in period for Gibbs sampling, which could dramatically reduce the time required to gather the number of samples required for this type of analysis.

Acknowledgment

The authors would like to thank the many volunteers at the DNA@Home volunteer computing project for making this research possible. This material is based upon work supported by the National Science Foundation under Grant No 0947637 and by the Department of Energy under Grant DE-FG02-09ER64756.

References

1. V. Pande *et al.*, "Atomistic protein folding simulations on the submillisecond timescale using worldwide distributed computing," *Biopolymers*, vol. 68, no. 1, pp. 91–109, 2002, peter Kollman Memorial Issue.
2. D. P. Anderson, E. Korpela, and R. Walton, "High-performance task distribution for volunteer computing." in *e-Science*. IEEE Computer Society, 2005, pp. 196–203.
3. C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton, "Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment," *Science*, vol. 262, no. 5131, pp. 208–214, 1993.
4. A. S. Bais, N. Kaminski, and P. V. Benos, "Finding subtypes of transcription factor motif pairs with distinct regulatory roles," *Nucleic Acids Research*, 2011.
5. G. D. Stormo, "Motif discovery using expectation maximization and gibbs sampling," in *Computational Biology of Transcription Factor Binding*, ser. Methods in Molecular Biology, I. Ladunga, Ed. Humana Press, 2010, vol. 674, pp. 85–95.
6. S. Challa and P. Thulasiraman, "Protein sequence motif discovery on distributed supercomputer," in *Proceedings of the 3rd international conference on Advances in grid and pervasive computing*, ser. GPC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 232–243.
7. X. Zhang, "Automatic feature learning and parameter estimation for hidden markov models using mce and gibbs sampling," Ph.D. dissertation, UNIVERSITY OF FLORIDA, 2009.
8. J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 363–370.
9. X. Tan, W. Xi, and J. S. Baras, "Decentralized coordination of autonomous swarms using parallel gibbs sampling," *Automatica*, vol. 46, no. 12, pp. 2068 – 2076, 2010.
10. D. Salas-Gonzalez, E. E. Kuruoglu, and D. P. Ruiz, "Modelling with mixture of symmetric stable distributions using gibbs sampling," *Signal Processing*, vol. 90, no. 3, pp. 774 – 783, 2010.
11. L. A. Newberg, W. A. Thompson, S. Conlan, T. M. Smith, L. A. McCue, and C. E. Lawrence, "A phylogenetic gibbs sampler that yields centroid solutions for cis-regulatory site prediction," *Bioinformatics*, vol. 23, pp. 1718–1727, July 2007.
12. W. A. Thompson, L. A. Newberg, S. Conlan, L. A. McCue, and C. E. Lawrence, "The gibbs centroid sampler," *Nucleic Acids Research*, vol. 35, no. Web-Server-Issue, pp. 232–237, 2007.
13. N. Lartillot, "Conjugate gibbs sampling for bayesian phylogenetic models." *Journal of Computational Biology*, vol. 13, no. 10, pp. 1701–1722, 2006.
14. A. Gelman and D. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical Science*, vol. 7, pp. 457–511, 1992.
15. L. Yu and Y. Xu, "A parallel gibbs sampling algorithm for motif finding on gpu," in *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on*, 2009, pp. 555 –558.
16. L. Kuttippurathu, M. Hsing, Y. Liu, B. Schmidt, D. L. Maskell, K. Lee, A. He, W. T. Pu, and S. W. Kong, "Decgpu: distributed error correction on massively parallel graphics processing units using cuda and mpi," *BMC Bioinformatics*, vol. 12, no. 85, 2011.