

CS 267 HW 0

MARK HOEMMEN

1. THIS PERSON WHO IDENTIFIES HIMSELF AS ME

I am starting the second semester of the Ph.D. program in computer science at the University of California, Berkeley. The optimization of numerical methods, especially for numerical linear algebra, is my main focus. “Optimization” has two interlocking faces:

- (1) Mathematical: e.g. convergence and stability.
- (2) Implementation: compilers and code generation.

Just because a method X converges faster than Y in theory, doesn’t mean that X is faster in practice, due to the strange world of modern computer architecture. Balancing mathematical and implementation issues is my goal. The problems I help solve should ideally be somehow useful to someone in an application domain.

My mathematical background is reasonably good (in the sense of “oh I remember seeing that before,” not in the sense of being able to whip out a pencil and derive it all). I like linear algebra (especially iterative methods for linear systems), ODEs and PDEs, and analysis; I get by when talking about nonlinear optimization, logic, discrete mathematics and data structures. My background in the sciences is lacking.

I am also interested in object-oriented design and software engineering. Two years working part-time on a simulator for shipboard disasters have given me practical experience in these fields. I can code in C, C++ (yay for the STL!), Matlab, Gnuplot (a bit), Perl, Java and even a little Lisp (I’m learning, slowly), and have a little experience with SQL and Mathematica. Though I prefer the GNU development tools (emacs, gcc, ddd, autoconf) and Unix-like environments, I used Visual Studio 6.0 (out of dire necessity) at my job and had some familiarity with Win32 system calls.

So I’m a coder with mathematical training. Another potentially useful fact is that I can speak German and read technical documents in the language. The other important facts about me will become apparent over time.

2. WHAT I’M LOOKING FOR IN THIS CLASS

I’m hoping that the lectures expose us to many applications of parallel computing, in many different fields. Thorough discussion of the benefits and costs of certain parallel models and architectures – ideally, in a way more abstract and mathematical than “it works in practice,” although that is useful information too – would be helpful.

I would also like to work on a project with someone who is closely tied to an application. It would be interesting, but not strictly necessary, if the mathematics (as well as the algorithm design and coding) were challenging. I’m concurrently

enrolled in two other courses: numerical methods for PDEs (a continuation of last semester's ODEs course by Prof. John Strain), and algorithms for large data sets.

3. AN APPLICATION OF PARALLEL COMPUTING

Rather than choosing a large, well-funded project from a familiar place, I decided to look for something smaller-scale from outside the U.S. – more specifically, Berlin, where I spent the 2002–3 academic year. The main center for supercomputing in Berlin is the Konrad-Zuse-Zentrum für Informationstechnik, which is on the campus of the Freie Universität in Dahlem. I am less familiar with bioinformatics than with other typical parallel computing applications, so I found something from the ZIB-based Center for Genome-Based Bioinformatics (Berliner Zentrum für Genombasierte Bioinformatik, abbrev. BCB). The BCB cooperates with the Berlin university system, the Charité research clinic in downtown Berlin, and other medical institutions. Thomas Steinke of the ZIB, a figure apparently better known in the computational chemistry world, heads a small team whose objective is to deploy an environment for threading-based prediction of protein structure.

3.1. Threading and the Protein Folding Problem. Bioinformatics as a field helps bridge the gap between the collection of genomic data and the understanding and therapeutic exploitation of genomic expression. DNA-carried information expresses itself in proteins, whose interactions with each other and with other biochemicals govern almost all life processes. Thus, an important bioinformatics goal is to understand the mapping from a gene sequence to a protein's molecular structure.

In theory, a sequence of genes contains all the information needed to construct the protein it encodes.¹ However, understanding and manipulating a protein's behavior requires detailed knowledge of its three-dimensional molecular structure. This is the so-called *Protein Folding Problem*: how a protein molecule “folds” and configures itself in space, in response to the electromagnetic interactions between the protein's component atoms. Experiments using X-ray crystallography and NMR spectroscopy can reveal this structure, but are expensive, inflexible and slow [11].

Computational solution is the logical next choice. As yet, however, direct solution of the folding problem has proven infeasible in most cases. Even small proteins have large numbers of atoms, and their molecular structure depends sensitively on the configuration of their atoms. For simulations that calculate only on the basis of physical laws, computational cost reaches the Petaflop/s range, even when using methods that approximate aggregations of atoms as a force field [8]. Petaflop/s computing power may not be available for a decade, and even then, the necessary approximations may spoil the results' accuracy.

As an alternative to purely physical law-based modeling, one may incorporate available experimental data into the simulation. Many proteins have been sequenced, and a number of searchable databases are available. Furthermore, in nature, a relatively small number ($O(10^3)$) of recurring structure components compose most known proteins. If an unknown sequence is found similar to a sequence whose structure is known, the structure resulting from the unknown sequence can be deduced. This

¹More precisely, a protein is determined by the sequence of its component amino acids, which in turn is determined by a gene sequence.

approach is called *threading*, also known as “fold recognition” or “inverse protein folding” (because it solves a kind of inverse folding problem). The method “threads” one protein’s sequence through another, looking for alignments that help minimize the free energy of the molecule.

Threading itself is not enough: an additional stage is needed, in which a “score function” (that maps a sequence to a real number) evaluates the quality of a computed alignment. Its global minimum is the sequence which corresponds to the structure that the function “represents.” The goal is minimization of the score function. This stage requires the most computation, and a number of optimization techniques have been tried. Steinke’s group uses a branch-and-bound optimizer, based abstractly on the algorithm of Lathrop and Smith [4]. Their paper only mentions that a MIMD implementation could conceivably divide up subsets (after a partition) among the nodes, and that a SIMD implementation could parallelize the (dynamic programming) calculation of the score function.

In a paper about a related algorithm, Yanev and Andonov point out that the amount of work required by each subtask is unknown before the subtask is executed [12]. Furthermore, if a subtask finishes early, it is (at least partially) successful, and all the other subtasks need to be notified so they can accept more relevant subtasks. Therefore, a parallel implementation of an algorithm of this type requires some sort of dynamic load balancing. An important test of an implementation is whether it achieves sufficient load balancing to keep a large number of processors busy.

3.2. The hardware. BCB has chosen Elfie, a 16 node GNU/Linux cluster delivered by Cray Deutschland GmbH, as their computational platform [2, 3]. Each node has two 2.2 GHz Pentium IV processors and 1 GB RAM, and Myrinet2000 and FastEthernet serve as the interconnection framework. A separate dual-processor machine handles a 0.5 TB RAID-5 storage device. While Elfie is no Top500-class machine, sixteen of the top 150 on the Top500 list are Pentium IV clusters (of at least 250 nodes), linked with either Myrinet or gigabit ethernet, which shows the potential of the technology [5]. A goal of the group is to deploy the most time-consuming computational components on a Cray X1, which holds significant positions on the Top500 list, running the LINPACK benchmark on 252, 124, 60 or even 40 nodes.

3.3. The software. The prototype system operates in a “pipeline” of three stages:

- (1) Pre-processing;
- (2) Threading;
- (3) Post-processing.

The first stage applies sequence-based (as opposed to structure-based, e.g. threading) algorithms to filter the data. Externally developed matching tools which are variants of BLAST (Basic Local Alignment Search Tool, see Altschul et al. [1]) are linked to MySQL-based databases (Interdom, COLUMBA) of known patterns. If a given test sequence is found to be highly similar to a known sequence, the pipeline is stopped. While filtering improves scoring accuracy in the next stage, sequence-based search techniques do not offer sufficient resolution to determine protein function [6], and so a structure-based method is needed.

The threading stage employs Xu’s scoring function [10] with Lathrop’s branch-and-bound optimizer [4].

The third stage, homology modeling, builds the 3-D model of the protein, based on the set of likely folds chosen in the threading stage. Steinke observes that this is the most computationally expensive stage. It is surprising that he devotes no space to describing the algorithms used in this step, although he carefully cites both the algorithm and scoring function that implement threading. For a similar pipeline, Xu et al. estimates that homology modeling consumes 80% of the total computational cost; the software responsible for this stage is apparently considered a black box. One package used by Xu's group is Nest/Jackal; its manual mentions nothing of parallel processing support [9]. Even if the parallel code runs infinitely fast, if homology modeling requires 80% of the runtime, the maximum parallel speedup is 1.25. This would certainly cause a bottleneck.

BCB's prototype is implemented in C++ with MPI used for message passing. Considering Yanev and Andonov's comments about dynamic load balancing, some form of message-passing seems the natural method for the problem. (Yanev and Andonov suggest a particular form of callbacks.) However, without information about the most computationally intensive stage, homology modeling, one cannot determine whether message-passing is optimal.

A similar pipeline has been implemented around the PROSPECT threading system at ORNL [6]. It is accessible via a web interface [7]. Steinke claims that his group's design is similar, but unlike ORNL's is not committed to a client-server model. The ORNL pipeline also links together a number of computational tools from various sources. It has a sophisticated "pipeline manager" that distributes computational jobs dynamically, either to ORNL's RS/6000 SP cluster or to a 64-node GNU/Linux cluster.

3.4. Results. Early results for computationally intensive problems (that might require a day or more on a single node, though results are not presented for a single node), show relatively poor scaling. For a realistic benchmark, 4 nodes takes 16 hours, 8 nodes 11 hours, 16 nodes 9 hours, and 32 nodes 8 hours. (See [8], Figure 2, which measures the wall-clock time in hours for a benchmark set of 572 sequences threaded against a database of about 37500 known sequences.) The algorithm's results are compared with experimental verifications.

REFERENCES

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool*, *J Mol Biol.* **215** (1990), no. 3, 403–410.
2. Berliner Centrum für Genombasierte Bioinformatik, *Elfie cluster at ZIB*, <http://elfie.bcbio.de/>, 2002.
3. A. Kerl, Jürgen Hochlenert, Dietmar Becker, and Siegfried Jesenko, *Website of Cray Computer Deutschland GmbH*, <http://www.cray.de/>, 2004.
4. R. H. Lathrop and T. F. Smith, *Global optimum protein threading with gapped alignment and empirical pair score functions*, *J. Mol. Biol.* **255** (1996), 641–665.
5. H. Meuer, E. Strohmaier, J. Dongarra, and H. D. Simon, *Current Top500 list*, <http://www.top500.org/list/2003/11/>, November 2003.
6. Manesh J. Shah, Sergei Passovets, Dongsup Kim, Kyle Ellrott, Li Wang, Inna Vokler, Philip F. LoCasccio, Dong Xu, and Ying Xu, *A computational pipeline for protein structure prediction and analysis at genome scale*, *Bioinformatics* **19** (2003), no. 15, 1985–1996.
7. ———, *ORNL Protein Analysis Pipeline*, <http://compbio.ornl.gov/proteinpipeline>, 2004.

8. T. Steinke, *Alignment and threading on massively parallel computers*, <http://www.bcbio.de/Report2003/12-Steinke03.pdf>, December 2003, Annual report.
9. J. Z. Xiang, *Jackal: A protein structure modeling package*, <http://honiglab.cpmc.columbia.edu/programs/jackal/>, October 2002.
10. Y. Xu, D. Xu, and E. Uberbacher, *An efficient computational method for globally optimal threading*, *J. Comp. Biol.* **5** (1998), no. 3.
11. Jacqueline Yadgari, Amihood Amir, and Ron Unger, *Representation and data structure of genetic algorithms for protein threading*, <http://citeseer.nj.nec.com/yadgari97representation.html>, 1997.
12. Nicola Yanev and Rumens Andonov, *Solving the protein threading problem in parallel*, <http://citeseer.nj.nec.com/567624.html>.

E-mail address: mhoemmen@cs.berkeley.edu