Setting up a Meta-Threading Pipeline for High-Throughput Structural Bioinformatics: eThread Software Distribution, Walkthrough and Resource Profiling

Michal Brylinski1,2* and Wei P. Feinstein1

1Department of Biological Sciences, Louisiana State University, Baton Rouge, LA 70803, USA
2Center for Computation & Technology, Louisiana State University, Baton Rouge, LA 70803, USA

Abstract
eThread, a meta-threading and machine learning-based approach, is designed to effectively identify structural templates for use in protein structure and function modeling from genomic data. This is an essential methodology for high-throughput structural bioinformatics and critical for systems biology, where extensive knowledge of protein structures and functions at the systems level is prerequisite. eThread integrates a diverse collection of algorithms, therefore its deployment on a large multi-core system necessarily requires comprehensive profiling to ensure the optimal utilization of available resources. Resource profiling of eThread and the single-threading component algorithms indicate a wide range of demands with respect to wall clock time and host memory. Depending on the threading algorithm used, the modeling of a single protein sequence of up to 600 residues in length takes minutes to hours. Full meta-threading of one gene product from E. coli proteome requires ~12h on average on a single state-of-the-art computing core. Depending on the target sequence length, the subsequent three-dimensional structure modeling using eThread/Modeller and eThread/TASSER-Lite takes additional 1-3 days of computing time. Using the entire proteome of E. coli, we demonstrate that parallel computing on a multi-core system follows Gustafson-Barnes' law and can significantly reduce the production time of eThread. Furthermore, graphics processor units can speedup portions of the calculations; however, to fully utilize this technology in protein threading, a substantial code development is required. eThread is freely available to the academic and non-commercial community as a user-friendly web-service at http://www.brylinski.org/ethread. We also provide source codes and step-by-step instructions for the local software installation as well as a case study demonstrating the complete procedure for protein structure modeling. We hope that genome-wide high-throughput structural bioinformatics using eThread will significantly expand our knowledge of protein structures and their molecular functions and contribute to the thriving area of systems biology.

Keywords: Protein structure prediction; Protein function annotation; Template-based modeling; Protein meta-threading; Structural bioinformatics; Proteome-scale modeling; High-performance computing

Abbreviations: GPU: Graphics Processor Unit; HPC: High-Performance Computing; PBS: Portable Batch System; SVM: Support Vector Machines

Introduction
In modern biological sciences, the focus has shifted from the study of individual molecules to the exhaustive exploration of molecular interactions at the systems level. This new paradigm has given rise to the rapidly developing domain of systems biology [1], which lies at the intersection of life and computer sciences. Systems biology is facilitated by whole genome sequencing [2] that routinely generates large datasets of protein sequences; nevertheless, the molecular structures and functions of many of these sequences often remain unknown. Computational methods for protein structure and function prediction are expected to bridge the gap between the number of known sequences and the number of fully annotated gene products, which are requisite for systems biology applications. Amongst many computational techniques developed over the past years, the most accurate algorithms in this field build on homology, i.e. they use information inferred from related proteins. Sequence-based methods can provide useful structural and functional information for a subset of target proteins; however, these algorithms typically require a high sequence identity to already annotated proteins to maintain a high accuracy [3]. As might be expected, this reduces the coverage of suitable targets, since for many proteins no close homologues are available in the public databases, e.g. the Protein Data Bank [4]. It has been demonstrated that relaxing the safe sequence similarity thresholds in sequence-based function annotation may lead to high levels of mis-annotation [5].

To address this issue, a number of techniques have been developed to search for low-sequence identity templates that can be used to construct the structural model of a target protein and to subsequently infer its molecular function. This is a major goal of contemporary structural bioinformatics, which aims at the high-throughput modeling of all gene products across the entire proteomes of various organisms in the so-called "twilight zone" of sequence similarity [6]. Protein threading [7] represents the latest trend in the development of template identification and alignment algorithms. These techniques have the desired capability to effectively deal with the complex and equivocal relations between protein sequence, structure and function, which are the major obstacles for standard bioinformatics approaches. Such
a structure-oriented approach holds a considerable promise to speed up genome-wide protein annotation [8], which certainly will have impact on many areas of modern molecular, cell and systems biology. It has a great potential to overcome the limitations of more traditional sequence-based approaches; however, at the cost of a significantly increased demand for computational resources.

In particular, meta-threading techniques are widely used in structural bioinformatics. These methods operate by considering outputs from a variety of individual threading algorithms; the combined predictions have a higher chance to be accurate than those produced by a single method. An example of such an approach is recently developed cThread, which integrates ten state-of-the-art single-threading algorithms additionally supported by machine learning to provide a unified resource for protein structure and function modeling [9]. Here, the scientific challenge is to efficiently combine multiple algorithms to significantly increase the overall accuracy over the individual methods and to push the envelope of systems-level template-based protein structure modeling and functional annotation. Meta-threading pipelines also render significant challenges at the level of practical implementation and the optimal utilization of computing resources. These can be considered as heterogeneous collections of algorithms and computational techniques that may significantly differ from each other in terms of the required wall clock time, memory usage, I/O operations and network bandwidth. They also may or may not share common input files or the access to external data libraries (eg. template libraries), and often employ a complicated system of dependencies between individual jobs. To the best of our knowledge, the majority of currently implemented meta-threading pipelines are available as pseudo-gateways, i.e. web interfaces that query several publicly available CGI servers over the Internet using simple web tools such as wget or curl.

A local installation of the meta-threading pipeline on a high-performance computing (HPC) platform provides the most reliable and robust solution for high-throughput structural bioinformatics [10]. However, running a diverse collection of algorithms on a large multi-core system necessarily requires comprehensive resource profiling to ensure the optimal utilization of available resources. In this communication, we describe the stand-alone software distribution of cThread, which can be deployed on any modern Linux-based HPC system. We perform a comprehensive resource profiling, analyze the computational requirements and discuss the achievable models of parallelism. We also touch on the possibility of accelerating the computations by graphics processors (GPUs). Finally, we provide a case study to demonstrate the practical application of this software on production platforms. cThread is freely available for academic and non-commercial users at www.brylinski.org/etthread.

Materials and Methods
Overview of cThread pipeline

cThread is a meta-threading procedure that combines predictions from ten state-of-the-art single-threading algorithms: COMPASS [11], CS/CSI-BLAST [12], HHpred [13], HH-MMER [14], pTools [15], pGenThreader [16], SAM-T2K [17], SPARKS [18], SP3 [18] and Threader [7], see table 1. It also uses NCBI BLAST [19] for the construction of sequence profiles and PSIRED [20] for the prediction of secondary structure. Currently, two structure modeling algorithms can be used in cThread to construct the three-dimensional structures of target proteins: Modeller [21] and TASSER-Lite [22]. cThread/Modeller requires template pre-clustering, which is calculated by MaxCluster. In addition to target-to-template alignments generated by cThread, cThread/TASSER-Lite needs long-range inter-residue contacts, which can be predicted for a given target sequence using Con tact (included in the cThread software distribution). Moreover, both modeling protocols employ several popular tools for protein structure modeling, e.g. Jackal [23] and Pulchra [24]. Typically, multiple models are generated for a given target sequence. To rank them and assign the prediction confidence, we developed cRank, which uses individual scoring functions provided by Modeller [21] and TASSER-Lite [22], as well as DFIRE [25], fr-TM-align [26], TM-score [27], DSP [28] and Stride [29]. The complete list of structure-based tools required by cThread is shown in table 2. cThread, cContact and cRank also employ several machine learning models to improve prediction accuracy; we use two software packages that offer various Support Vector Machines (SVM) algorithms: LIBSVM [30] and SVM^light [31], see table 3.

Physical testing systems

The primary testing system is HP Proliant DL 180 G6 server which has 2 Intel Xeon E5645 6-core processors running at 2.4GHz and it is equipped with 48GB of memory. Additionally, the following three systems were used for the benchmarking of GPU-BLAST: 1) dual Intel Xeon E5620 4-core processor running at 2.4GHz, equipped with 24GB RAM and NVIDIA Tesla M2050, 2) dual Intel Xeon E5540 4-core processor running at 2.5GHz, equipped with 24GB RAM and NVIDIA Tesla M2070, and 3) single Intel Xeon E5540 4-core processor running at 2.5GHz, equipped with 36GB RAM and NVIDIA Tesla C2075.

Simulated multi-core systems

We constructed 18 virtual multi-core systems, equipped with 6, 8, 12, 16, 24 and 32 computing cores, and 1, 2 and 4GB of RAM per core. We also designed a simple job scheduling system that assigns jobs to the computing cores using the following rules: 1) the total number of concurrently running jobs must be less or equal to the number of cores, 2) the total memory for running jobs cannot exceed the host shared

---

Software | Purpose | Link
---|---|---
COMPASS | Protein threading/ fold recognition | http://prodata.swmed.edu/compass/compass.php
CS/CSI-BLAST | Protein threading/ fold recognition | http://toolkit.tuebingen.mpg.de/cs_blast
HHpred | Protein threading/ fold recognition | http://toolkit.tuebingen.mpg.de/hhpred
HMMER | Protein threading/ fold recognition | http://hmmer.janelia.org
NCBI BLAST | Sequence alignment | http://blast.ncbi.nlm.nih.gov/Blast.cgi
pGenThreader | Protein threading/ fold recognition | http://bioinf.cs.ucl.ac.uk/psipred/?program=mgenthreader
PSIPRED | Secondary structure prediction | http://bioinf.cs.ucl.ac.uk/psipred
SAM-T2K | Protein threading/ fold recognition | http://compbio.soe.ucsc.edu/papers/sam_doc/sam_doc.html
SPARKS/SP3 | Protein threading/ fold recognition | http://sparks.infomatics.iupui.edu/index.php?pageLoc=Services
Threader | Protein threading/ fold recognition | http://bioinf.cs.ucl.ac.uk/software_downloads/threader

Table 1: Sequence-based tools. Components of the cThread pipeline for protein threading and sequence analysis.
library due to the number and length of template structures, the chain reports the percentage of time spent on threading (11,468 structures) and domain-only (10,013 structures). Figure 1B the calculations.

The least expensive algorithms, with respect to the total CPU time. well with the target sequence length; however, the algorithms differ method. In all cases, except for HHMER, the simulation time scales

Testing system are shown in and 8GB of RAM per node.

many HPC systems; for example, currently the largest HPC cluster significant memory consumption and computing time. However, throughout 96% of the simulation time (Figure 2A, SAM-T2K). Furthermore, in most cases, the required memory scales well with the target protein length. It is also important to note that all these algorithms do not depend on each other therefore can be efficiently processed in parallel.

Datasets

The resource profiling is carried out on a dataset of 275 proteins randomly chosen from the original eThread benchmarking dataset [9]. These proteins were selected to uniformly populate 11 bins with 25 structures in each bin. The bins evenly span the range of the target sequence length between 50 and 600 residues.

As a benchmarking dataset for the simulated systems-level modeling, we selected the complete proteome of Escherichia coli K-12 [32], which comprises 4,646 gene products 50-600 amino acids in length. eThread meta-threading pipeline employ ten individual threading algorithms, thus the total number of jobs needed to process E. coli proteome is 46,460. The expected memory consumption and computing time for each job was calculated based on meta-threading profiling results obtained on the primary testing system.

Results

Profiling of meta-threading components

Individual single-threading components of the eThread pipeline significantly differ from each other with respect to the wall clock time and memory consumption. Both resources are typically limited on many HPC systems; for example, currently the largest HPC cluster in the state of Louisiana, Queen Bee (http://www.loni.org/systems/), allows jobs to run for up to 48 hours and features 8 computing cores and 8GB of RAM per node.

The results of meta-threading resource profiling on the primary testing system are shown in figures 1 and 2. Figure 1A shows the average wall clock time required for each single-threading component method. In all cases, except for HHMER, the simulation time scales well with the target sequence length; however, the algorithms differ with respect to the total CPU time. The least expensive algorithms, CSI-BLAST, pTTools and hmmmer, require at most minutes, whereas the most expensive Threadder typically needs several hours to complete the calculations. eThread uses two template libraries: full-chain (11,468 structures) and domain-only (10,013 structures). Figure 1B reports the percentage of time spent on threading through a particular library; due to the number and length of template structures, the chain library requires slightly longer simulation times. Furthermore, several algorithms use PSI-BLAST to construct a sequence profile for a given target, which is often the most time consuming step. Blue pie slices in figure 1B show that for HHPred, COMPASS and pGenThreadder, 88%, 62% and 51% CPU cycles are used up by the sequence profile construction, respectively.

Individual threading algorithms also differ with respect to the memory utilization, see figure 2A. CSI-BLAST, pTTools, Threadder and HMMER require the least amount of memory; whereas, HHPred, COMPASS and pGenThreadder, all of which employ PSI-BLAST for the construction of sequence profiles, need significantly more RAM. Figure 2B shows that for HHPred, COMPASS and pGenThreadder, 86%, 78% and 93% of the memory was utilized during the sequence profile construction, respectively. SAM-T2K has the highest memory requirement because it launches BLASTP, which loads a large sequence library into memory. The actual threading calculations use only ~2% of the memory; however, throughout 96% of the simulation time (Figure 1B, SAM-T2K). Furthermore, in most cases, the required memory scales well with the target protein length. It is also important to note that all these algorithms do not depend on each other therefore can be efficiently processed in parallel.

Simulated multi-core system running meta-threading

We conduct a simple computer experiment to show that meta-threading pipelines follow Gustafson-Barsis' law, which states that computations involving arbitrarily large data sets can be efficiently parallelized [33]. Figure 3 presents the simulated operation of three virtual systems equipped with 8, 16 and 32 cores, and 1GB of RAM per core, processing 46,460 individual threading jobs for E. coli proteome. The CPU utilization is almost constantly 100% throughout the operation time, with the exception for two larger systems, where a few high memory jobs initially saturated the available host memory and blocked the remaining cores for a short period of time, see figures 3B and 3C. Moreover, because of the job prioritization, high memory jobs, e.g. SAM-T2K, pGenThreadder and COMPASS, as well as long jobs, e.g. Threadder, were selected for execution before other threading algorithms.

The total computer time required for completing meta-threading against E. coli proteome is shown in figure 4. For instance, one needs 6 years, 303 days and 17 hours to complete the calculations on a single computing core. Using 100 nodes of the aforementioned Louisiana

<table>
<thead>
<tr>
<th>Software</th>
<th>Purpose</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFIRE</td>
<td>Protein conformation free energy score</td>
<td><a href="http://sparks.informatics.tupui.edu/hzhou/dfire.html">http://sparks.informatics.tupui.edu/hzhou/dfire.html</a></td>
</tr>
<tr>
<td>DSSP</td>
<td>Secondary structure assignment</td>
<td><a href="http://swift.cmbi.ru.nl/dssp/dssp.html">http://swift.cmbi.ru.nl/dssp/dssp.html</a></td>
</tr>
<tr>
<td>h-TM-align</td>
<td>Protein structural alignment</td>
<td><a href="http://cssebiology.gatech.edu/h-tm-align">http://cssebiology.gatech.edu/h-tm-align</a></td>
</tr>
<tr>
<td>MaxCluster</td>
<td>Protein structure comparison and clustering</td>
<td><a href="http://www.stbg.bio.ic.ac.uk/~maxcluster/index.html">http://www.stbg.bio.ic.ac.uk/~maxcluster/index.html</a></td>
</tr>
<tr>
<td>Modeller</td>
<td>Protein structure modeling</td>
<td><a href="http://saliab.org/modeller">http://saliab.org/modeller</a></td>
</tr>
<tr>
<td>Pulchra</td>
<td>All-atom reconstruction from the backbone Ca atoms</td>
<td><a href="http://cssebiology.gatech.edu/PULCHRA">http://cssebiology.gatech.edu/PULCHRA</a></td>
</tr>
<tr>
<td>Stride</td>
<td>Secondary structure assignment</td>
<td><a href="http://webclu.bio.wzw.tum.de/stride">http://webclu.bio.wzw.tum.de/stride</a></td>
</tr>
<tr>
<td>TASSER-Lite</td>
<td>Protein structure modeling</td>
<td><a href="http://cssebiology.gatech.edu/TASSER-Lite">http://cssebiology.gatech.edu/TASSER-Lite</a></td>
</tr>
<tr>
<td>TM-score</td>
<td>Protein structure similarity measure</td>
<td><a href="http://zhanglab.ccmb.med.umich.edu/TM-score">http://zhanglab.ccmb.med.umich.edu/TM-score</a></td>
</tr>
</tbody>
</table>

Table 2: Structure-based tools. Components of the eThread pipeline for protein structure modeling and analysis.

<table>
<thead>
<tr>
<th>Software</th>
<th>Purpose</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBSVM</td>
<td>Support Vector Machines for classification and regression</td>
<td><a href="http://www.csie.ntu.edu.tw/~cjlin/libsvm">http://www.csie.ntu.edu.tw/~cjlin/libsvm</a></td>
</tr>
</tbody>
</table>

Table 3: Machine learning tools. Machine learning algorithms used by eThread for various classification and ranking tasks.
Figure 1: Resource profiling for threading component methods. (A) Average ± standard deviation wall time for individual components of eThread plotted as a function of the target protein length; (B) percentage of wall time used by individual components of eThread to thread against chain and domain template libraries. Profile construction using NCBI BLAST is benchmarked independently from threading when possible.

Figure 2: Resource profiling for threading component methods. (A) Average ± standard deviation memory consumption for individual components of eThread plotted as a function of the target protein length; (B) percentage of memory used by individual components of eThread to thread against chain and domain template libraries. Profile construction using NCBI BLAST is benchmarked independently from threading when possible.
HPC cluster Queen Bee, each equipped with 8 cores and 8GB of RAM, the entire *E. coli* proteome could be processed in 3 days and 3 hours. However, increasing the size of the host shared memory does not shorten the simulation time. This is because the computations are dominated by low memory jobs, e.g., *E*/*T* reader. Figure 3A demonstrates that the memory was fully utilized only throughout around 50% of the total computing time, which leaves a substantial room to maneuver job allocation. Figure 4 inset shows that a meta-threading pipeline closely follows Gustafson-Barsis’ law, i.e., doubling the number of CPU cores shortens the computing time by a factor of 2. We identify three factors responsible for this performance: 1) a large set of diverse protein threading jobs, 2) a considerable margin for the memory utilization on modern HPC systems, and 3) the lack of dependencies between individual jobs. Consequently, proteome-wide meta-threading applications are perfectly parallelizable at the task level.

**Profiling of eThread**

*e*Thread is a meta-predictor that integrates outputs from individual threading components. It operates in two modes: structural and functional. The former identifies the most confident structural templates and constructs consensus target-to-template alignments for use in protein structure modeling. The latter additionally evaluates selected templates for the utility in function assignment and considers a variety of protein molecular functions: ligand, metal, inorganic cluster, protein and nucleic acid binding. Figure 5 shows that the memory required by *e*Thread is well correlated with the target protein length, whereas the average wall clock time is characterized by larger standard deviations; this is because the simulation time also depends on the number of identified templates. Moreover, including the functional component fourfold increases the wall clock time and the memory.
consumption. Nevertheless, eThread does not require significant computing resources. Running in a structural (functional) mode, in most of the cases, it completes within 30 (80) minutes and consumes up to 0.6 (2.3) GB of memory.

**Profiling of protein structure modeling**

In eThread, the three-dimensional structural models of target proteins can be constructed using two modeling protocols: Modeller [21] and TASSER-Lite [22]. Resource profiling for eThread/Modeller are shown in figure 6A. Both simulation time and the memory used are correlated with the target protein length. This protocol has relatively low hardware requirements; even long target sequences typically need less than 0.5GB of RAM and up to 50 hours of CPU time. Compared to eThread/Modeller, eThread/TASSER-Lite requires significantly more resources, see figure 6B. The duration of structure assembly and refinement simulations in TASSER-Lite is limited to 48 hours, so the total wall clock time does not keep growing exponentially for sequences longer than 300 residues. Both modeling protocols profiled here, eThread/Modeller and eThread/TASSER-Lite, typically generate full chain models within 1-3 days on a single computing core.

eThread/Modeller comprises two modeling stages: structure assembly by Modeller followed by model ranking using eRank. There is very little overhead resulting from eRank; it completes within seconds, therefore it is not included in the profiling results. In contrast, structure modeling using eThread/TASSER-Lite consists of four consecutive stages: residue contact prediction using eContact, threading by Prospector [34], structure assembly/refinement by TASSER and model ranking by eRank. As shown in figure 6C, structure assembly and refinement is the most computationally intense and consumes 80% of the total CPU time. TASSER-Lite also includes additional threading using Prospector. This modeling stage is the most memory intense and

---

**Figure 5:** Resource profiling for eThread. Average ± standard deviation wall time and memory consumption for the identification of (A) structural and (B) functional templates by eThread plotted as a function of the target protein length. Percentage of (C) wall time and (D) memory used by eThread to identify structural and functional templates in chain and domain libraries.

**Figure 6:** Resource profiling for protein structure modeling. Average ± standard deviation wall time and memory consumption for the model construction using (A) eThread/Modeller and (B) eThread/TASSER-Lite plotted as a function of the target protein length. Percentage of (C) wall time and (D) memory used by the individual components of the eThread/TASSER-Lite modeling protocol: eContact, Prospector, TASSER-Lite and eRank/TASSER-Lite.
accounts for 65% of the total memory utilization of up to 1.65GB, see figure 6d. eContact, which predicts long-range inter-residue contacts before TASSER simulations get started, and eRank, which ranks the constructed models, extend the modeling time by only 5% and have relatively small memory requirements compared to the remaining modeling stages.

**Potential for GPU acceleration**

Heterogeneous HPC systems that include massively parallel graphics processors (GPUs) are quickly becoming popular, mainly because of their remarkably high performance-to-cost ratio. Consequently, GPU-accelerated supercomputers show an exponential growth in the Top500 ranking, with 52 systems powered by NVIDIA Tesla GPUs currently on the list compared to only 10 systems in 2010. Bioinformatics and systems biology are examples of many rapidly developing research areas that are moving towards heterogeneous computing architectures [35]; GPU implementations of several popular bioinformatics tools have been reported recently. Bioinformatics software available for GPUs include both sequence, e.g. CUDA-BLASTP [36], GPU-BLAST [37], CUDASW++ [38] and GHOSTM [39], as well as structure alignment algorithms, e.g. ppsAlign [40] and TM-score-GPU [41]. Most of the component methods integrated into the eThread pipeline do not have a GPU implementation, with the exception for BLASTP, which is used by SAM-T2K. To the best of our knowledge, GPU-BLASTP has not been benchmarked against its serial CPU version in a meta-threading environment.

Figure 7 shows wall clock times for serial NCBI BLASTP compared to that provided by GPU-BLAST [37] collected on three systems equipped with different GPU cards. Interestingly, the speedup depends on the target sequence length; this is likely due to the overhead caused by transferring the library data to the accelerator. Longer sequences require more calculations, thus the parallel processing by multiple GPU cores results in significantly shorter simulation times compared to the serial version. The speedup starts at 1.4, 1.5 and 1.7 for sequences 50-100aa, and reaches 2.1, 2.2 and 2.5 for sequences 550-600aa on Tesla M2050, M2070 and C2075 card, respectively. Without the laborious porting of the source codes of individual protein threading algorithms to CUDA, the construction of sequence profiles using PSI-BLAST would be the next logical step to speed up the entire pipeline by accelerating HHpred, COMPASS and pGenThreader; however, it is not currently available. Nevertheless, the Authors of GPU-BLAST noted that PSI-BLAST can be implemented on the GPU similarly to BLASTP and similar speedups can be expected [37]. Once available, it could give a boost to meta-threading pipelines by moving the sequence profile construction to GPU accelerators.

**Software walkthrough and a case study**

eThread is freely available to the academic community. Web-based eThread provides a user-friendly interface for a fast and easy access to the entire software package. Once the target amino acid sequence is submitted with selected structure prediction options, the modeling results can be downloaded to a local machine or displayed directly on the website, see figure 8 for a snapshot of prediction results. In figure 8A, the top-ranked structural model predicted using eThread/Modeller is visualized in the Astex Viewer Java applet [42]. The estimated TM-score of 0.753 for this model suggests a high modeling confidence. The Ramachandran plot in figure 8B shows that 87.8% amino acids colored in red; here, a threshold of 90% is commonly accepted to define high quality models. The web server also provides other results from the model quality check by PROCHECK [43] with respect to main-chain and side-chain parameters, e.g. peptide bound planarity, bad non-bonded interactions and Ca tetrahedral distortion. These may help users assess how a predicted structure compares with well-refined experimental structures (Figure 8C).

In addition to the web-based service, we also provide the source code of eThread allowing users to install the software package and build protein structures locally. There are three major steps for the local setup including the installation of: 1) required Perl modules, 2) third-party software for single-threading algorithms, and 3) eThread...
software and the corresponding threading libraries. We note that all third-party programs are free for academic and non-commercial use; however, users are responsible for obtaining software licenses and complying with legal and other requirements. Upon the completion of the software installation, all single-threading tools as well as eThread will be available locally for the identification of structural templates and the construction of target-to-template alignments. Finally, structural modeling protocols, Modeller [21] and TASSER-Lite [22], can be used to build the three-dimensional model for a given target sequence followed by a simple all-atom refinement using molecular mechanics.

As a proof of concept, we use eThread to build a structural model for the 59-residue fragment of an uncharacterized protein from domestic horse [44] (Uniprot ID: F6VMN7), for which the experimental structure is unavailable. The first step is the construction of threading alignments. We start with obtaining the amino acid sequence of F6VMN7 in FASTA format from UniProt [45]. Next, we deploy each protein threading/fold recognition algorithm with the sequence of F6VMN7 as an input to identify suitable templates and to generate the corresponding target-to-template alignments. Structural templates are selected from both full-chain and domain-only libraries. Because different threading algorithms generate target-to-template alignments in different formats, the output files are converted to the eThread format using conversion scripts included in the eThread software distribution. Next, the target-to-template alignments constructed by individual threading algorithms are concatenated as one of the input files for eThread. In addition to this input, two other files, the target sequence in FASTA format and the threading libraries, are required to generate the final consensus alignments. As the second step, we use Modeller and TASSER-Lite separately to build structure models from eThread alignments. Typically, more than one model is predicted, therefore model construction is followed by ranking and quality assessment. Specifically, for eRank/Modeller, three input files, including the target F6VMN7 sequence and two files generated by PSSIPRED and eThread, are required to rank the constructed models and assign TM-scores as confidence estimates. In figure 9A, the top-ranked eThread/Modeller model is shown in PDB format and its molecular structure is visualized in VMD [46]. For eThread/TASSER-Lite, the inter-residue contacts are first predicted using eContact; subsequently, residue contacts, target sequence and eThread alignments are used as input to construct the three-dimensional model. Similar to eRank/Modeller, eRank/TASSER-Lite is then deployed to rank protein models and assign the modeling confidence. The resulting top-ranked model constructed for F6VMN7 by eThread/TASSER-Lite is shown in figure 9B. Figure 9C shows the global superposition of the top-ranked models generated by eThread/Modeller and eThread/TASSER-Lite. Both models are remarkably similar to each other (RMSD is 0.74Å), which indicates a high confidence for the structure modeling of this target. However, there exist some differences. For example, the model built using eThread/Modeller (yellow) has longer beta sheets compared to that using eThread/TASSER-Lite (green), with approximately 27% and 13% of residues assigned by STRIDE [29] to the β-sheet conformation, respectively. It suggests that despite the likely correct global topology, the model constructed by eThread/TASSER-Lite may require more rigorous local structure refinement to improve the secondary structure content [9].

Conclusion

Systems biology is emerging as a promising discipline in the field of biology. Powered by modern computer technologies, it aims to help comprehend molecular interactions at the systems level. Towards this goal, acquiring extensive knowledge of protein structures and their functions is essential. Continuing advancements in sequencing technologies spark off the rapid accumulation of gene and gene product sequences; yet, the annotation of these sequences is falling far behind. Therefore, a high-throughput protein annotation is a daunting task in bioinformatics. Up to date, various computational tools have been developed to reach this goal. Different from sequence-based methods that heavily depend on high sequence identity to already annotated protein sequences, structure-based methods are making headway in function inference in the “twilight zone” [6] of sequence identity. Consequently, the genome-wide coverage of annotated proteins can be systematically expanded. Among many template-based approaches,
protein meta-threading is of particular interest, primarily because this approach integrates multifaceted factors to enhance the prediction accuracy. Towards this effort, we developed eThread that combines ten single-threading algorithms supported by machine learning to identify suitable templates for the prediction of protein structure and function [9].

The heterogeneous collection of algorithms used in eThread creates a challenge for the optimal utilization of system resources. In this communication, we thoroughly profile eThread and its component methods in terms of the total wall clock time and memory consumption, as well as the resource distribution at major computing stages. The profiling results show that the total CPU time and memory utilization differ dramatically among single-threading methods; yet, the overall resource required typically scales well with the target protein length. Furthermore, in a simple experiment using several simulated multicore systems, we show that meta-threading pipelines closely follows Gustafson-Barsis’ law [33], thus systems-level applications, eg. genome-wide modeling of protein structure and function, are exemplary tasks for large computer clusters.

In addition to parallel computing using multiple CPU cores, we also examine whether using a GPU-accelerated platform would shorten the production time. The benchmarking results are encouraging; however, to significantly speed up protein meta-threading pipelines requires a substantial code development and porting individual algorithms to CUDA. Here, one of the most promising targets for GPU computing is PSI-BLAST, which is used by several component methods. A GPU implementation of this algorithm could significantly accelerate the entire meta-threading pipeline. Similarly to GPU computing, alternative technologies, such as Intel Many Integrated Core architecture, also hold a considerable promise to speed up bioinformatics applications.

We provide a user-friendly web service freely to the academic community and non-commercial users; we also provide source code of eThread, which can be deployed locally on a high-performance computing platform for high-throughput protein structure and function modeling. The web-based gateway, stand-alone software, benchmarking results and datasets, as well as documentation and illustrative tutorials are available at www.brylinski.org/ethread.

Acknowledgements

This study was supported by the Louisiana Board of Regents through the Board of Regents Support Fund [contract LEQSF (2012-15)-RD-A-05]. We are grateful to NVIDIA for the donation of Tesla C2075 through their Academic Partnership Program. We also thank Drs. Mark Jarrell and Juana Moreno, leaders of the LA-SIGMA GPU team (http://lasigma.loni.org/), for providing an access to testing systems equipped with Tesla M2050 and M2070 cards, and Mr. Jason Rappleye, currently Production Engineer at Facebook, for contributing a script for the memory usage profiling. Portions of this research were conducted with high performance computational resources provided by Louisiana State University (HPC@LSU, http://www.hpc.lsu.edu) and the Louisiana Optical Network Institute (LONI, http://www.loni.org/).

References


