MOOSE
Microarray Optimal Oligo Selection Engine

cs267 Project
D Bonachea, J Chapman & N Putnam

Introduction

The DNA Microarray

DNA microarrays provide researchers with a high throughput, genome-wide technique for measuring the instantaneous state of a culture of living cells. These arrays consist of tens of thousands of microscopic, robotically created spots of DNA molecules of a specific sequence, chemically attached to a solid substrate.[1]

The arrays take advantage of the fundamental property of complementary binding of DNA molecules. A single-stranded DNA molecule binds with extremely high specificity to a molecule of complementary sequence. This Watson-Crick base pairing, or hybridization, allows a DNA microarray to serve as a sorting platform on which a mixture of thousands of species of DNA molecules can be separated, and the concentration of each measured relative to the others.

A particularly powerful example of such an experiment, and the application which drove the development of this technology, is the genome-wide comparative gene expression experiment.[2]

The genome-wide comparative gene expression experiment.

The fundamental discovery of molecular biology is that a gene (a stretch of DNA sequence on a chromosome) encodes one (or more) proteins. The DNA encoding of the gene is *transcribed* to form a messenger RNA (mRNA) which is *translated* into a protein.

The regulation of the production of specific proteins is the factor that regulates life processes at the molecular level: it determines the differentiation of cells into different tissues and regulates the cell division process. Even in the simplest organisms, the network of molecular interactions responsible for this regulation is very complicated.

DNA microarrays provide a powerful experimental probe of this complicated regulatory network. In this method, two cell samples are harvested, and the mRNA purified from each. Each purified mRNA solution represents a snapshot of the level of transcriptional activity of all the genes. These mRNAs are then used as templates in a reverse transcription reaction, which produces complementary DNA (cDNA). The cDNA can be chemically modified by the attachment of a fluorescent dye molecule. Each sample can be “labeled” with a different color fluor.

The two samples can now be mixed and placed in contact with the microarray. If each feature in the array contains immobilized copies of the DNA sequence complementary to one of the genes in the organism under study, and if all the genes differ in sequence from one another significantly enough, the relative intensity of the fluorescence signal from each dye color at each spot is a measure of the relative rate of transcription of the corresponding gene.

Other applications of microarray technology include genomic copy number studies, and the regulation of translation.[3]

Probe Selection and Array Design

Two main strategies are commonly employed to derive the DNA samples which are deposited robotically to create the array. In the first, relatively long lengths of DNA (of order 1000 base pairs) are amplified using PCR (polymerase chain reaction). This method relies on the existence of a suitable template from which probe sequences can be amplified up. Usually the templates used are cDNA libraries.

The second strategy (the one which is the focus of this project) is to use chemically synthesized probes. The advantages are that the samples can be produced more easily, without the maintenance and handling of large bacterial clone libraries. This method is also well suited to the creation of microarrays from genomic sequences. When the sequence of an organism has been determined, an array can be designed to detect transcripts of all predicted genes, and even to distinguish splice variants of predicted and known genes. In the cDNA approach, the array is necessarily biased towards genes that are already known to be highly expressed under some conditions, because of the way cDNA libraries are constructed.

Because of the limitations of DNA synthesis technology, this second method requires the use of much shorter probe sequences of around 100 base pairs. In this project, we take up the computational problem of selecting these probe sequences. For each gene, an identifying probe sequence must be selected . The probe oligonucleotide sequences must satisfy the following constraints:

1. Melting Temperature (TM): all of the probe oligos must have similar melting temperature. The melting temperature of an oligonucleotide sequence is defined as the temperature at which one half of the molecules of the sequence are in duplex with their complement and half are single-stranded (under given solution conditions). In order for hybridization (duplex-formation) to occur optimally in the experiment, all duplexes must melt within a small window of temperature (typically a few degrees). Thus, the TM of each candidate oligo must be calculated.

2. Secondary Structure: in order for the bimolecular hybridization reaction to proceed efficiently, nucleation sites for duplex-formation must be readily available on both probe and target molecules. The formation of secondary structure interferes with the accessibility of duplex nucleation sites by occupying these bases intra-molecularly. Because the intramolecular secondary structure hybridization reaction is first-order, it proceeds rapidly, significantly reducing the efficiency of the (second-order) bimolecular hybridization. Therefore, for optimal duplex yield, we seek to minimize the secondary structure of the optimal probes under the experimental conditions. Thus, we must predict the likelihood of secondary structure formation for each candidate oligo.

3. Uniqueness: ideally, each probe oligo would preferentially bind only a single gene target, yielding a unique identifier of the expression level of said gene. Cross-hybridization of probe-target pairs makes evaluation of the experimental results extremely difficult. Therefore, if possible, an optimal probe should bind with high probability only to its intended target. Thus, we must predict the duplex binding affinity for all candidate oligo probes with all gene sequences that will be present in the solution.

Specifically, we have chosen to predict a set of 70-mer oligonucleotide probe sequences optimally designed for microarray expression level experiments on the yeast *Saccharomyces cerevisiae*. The entire genome of this organism is known and contains approximately 6000 coding sequences of average length around 1000 bases. For practical experimental reasons, we seek probes with TM within a few degrees of 70°C, and generated from the 700-base tail-end of the coding sequences. Thus, we must calculate melting temperature and secondary structure for approximately 700X6000 = 4.2 Million 70-mer sequences, and screen some appreciable subset of these for uniqueness against all 6000 genes. An efficient parallel implementation is necessary if this method is eventually to be used on the much larger human genome comprised of some 100,000 genes of similar length.

T_M Calculation

The thermostability of a nucleic acid duplex has been shown to depend primarily on nearest-neighbor base-stacking interactions[4,5]. Thermodynamic parameters for the ten unique Watson-Crick nearest neighbor dinucleotide interactions have been experimentally measured and used to predict the melting behavior of nucleic acid duplexes from their primary sequence.[6,7,8,9,10] The explicit expression for the melting temperature is given by:

$$T_M = \Delta H° / (\Delta S° + R \ln C_T) \tag{1}$$

where $\Delta H°$ is the enthalpic contribution to the free energy ($\Delta G°$), $\Delta S°$ is the entropic contribution, C_T is the total oligonucleotide strand concentration, and R is the gas constant (1.987 cal/K mol). A term proportional to the natural logarithm of the salt concentration is added under conditions other than 1M salt. It should be noted that the total strand concentration (C_T) is a rather ill-defined quantity for the microarray experiments of interest because the oligonucleotides are bound to a solid substrate, whereas eq. (1) assumes free strands in solution. Thus, some entropic correction is expected to be necessary for precise calculations. The enthalpy and entropy terms are to be calculated using the following:

$$\Delta H° = \Delta h_i + \Sigma_x \Delta h_x \tag{2}$$

$$\Delta S° = \Delta s_i + \Sigma_x \Delta s_x + \Delta s_{sym} \tag{3}$$

where $\Delta h/s_i$ is the enthalpy/entropy associated with initiation of a duplex, the $\Delta h/s_x$ are nearest-neighbor stacking enthalpies/entropies, and Δs_{sym} is an entropic correction employed in the case of self-complementary oligonucleotide sequences. Several groups have tabulated these values for DNA, RNA, and hybrid DNA-RNA duplex formation[4-7,11,12]. Throughout our implementation, we have used those values given in ref. 6.

This calculation is rather straightforward, but a few computational optimizations were employed - these will be described in the Implementation section below. The resulting melting temperature distribution is depicted in Figure 1. As noted in that figure, the global T_M distribution (under solution conditions of 50 mM salt concentration, 50 nM C_T) approximates a Gaussian with peak at 68° C and 9° C full-width-at-half-maximum, but the distributions for individual genes are not randomly sampled from this global distribution (i.e. individual gene distributions do not necessarily resemble the global distribution). Therefore, a screening of probe sequences with the T_M criterion produces widely varying numbers of acceptable sequences, ranging from all to none (or very few).

This creates the potential for significant load imbalance in later phases of the calculation (ways of resolving this will be discussed in Implementation and Future Goals, below).

Secondary Structure (***My*** ***S***econdary ***S***tructure ***T***echnique)

The formation of stable intramolecular duplexes can be predicted using the same thermodynamic base-stacking parameters as described in the T_M calculation. In this case, we are concerned with the stability of short regions of duplex formed when the oligonucleotide folds back upon itself. Several approaches to predicting these intramolecular secondary structures for RNA molecules (for which these structures often play a key role in functionality) have been described in the literature[13.] Essentially the problem amounts to a constrained free energy minimization. In addition to the stacking energies discussed above, there exists a hierarchy of topological contributions to the free energy to account for entropic effects of loop formation within the oligonucleotide. These effects have also been exhaustively tabulated (see for instance ref. 5) and are used in RNA structure prediction packages such as MFOLD (see ref. 8) and the VIENNA RNA Package (see ref. 9). However, for our application we have designed a heuristic algorithm that answers the somewhat simpler question: is it likely that a stable secondary structure *exists* (regardless of what that structure actually is) for a given oligonucleotide at a given temperature? We use a greedy algorithm to find a set of non-overlapping self-complementary regions within the oligonucleotide using only the base-stacking free energy parameters. Our method ignores all issues of topology and thereby is liable to produce non-physically realizable "secondary structures". Nonetheless, we feel that this heuristic is adequate for screening potential probe sequences for possible secondary structure because it is extremely conservative. We have compared the results of MySST to those predicted by the VIENNA-fold package and found this assumption to be justified (see Figure 2).

The basic MySST algorithm involves scanning the lower triangle of an n x n matrix (where the probe oligonucleotide is of length n), M_{ij}, for stable stretches of paired bases. The matrix M is defined by the following:

$$M_{ij} = \begin{cases} 1 \text{ if } f_i = \bar{r}_j \\ 0 \text{ otherwise} \end{cases}$$

where f is the forward oligonucleotide sequence, r is the reverse sequence, and barring represents the Watson-Crick complementation (i.e. A↔T, C↔G). Note that stretches of duplex-forming sequence consist of a series of bases in the forward strand paired with their complementary bases in the reverse strand. Thus, by searching for stretches of consecutive non-zero entries along the counter-diagonals of the matrix M, we identify possible duplex regions (see Figure 3). In practice, the MySST algorithm makes a sorted list of all such stretches of length greater than two base pairs that are stable (based on dinucleotide stacking energies). This list is greedily trimmed by taking the most stable duplex regions that do not overlap with stretches that have already been taken. The final result is a set of non-overlapping complementary intramolecular duplex sequences and a global stability (free energy) for this set. As noted, the "structure" predicted is neither guaranteed to be of minimal energy, nor even physically reasonable. Nonetheless, for a random sample of 60-mers taken from the *Saccharomyces cerevisiae* (yeast) coding sequence, we find that MySST does rather well compared to the VIENNA fold package and runs as much as ten times faster on a single Pentium II workstation.

Quantitatively, those sequences which exhibit zero secondary structure propensities by the MySST heuristic are ≥ 95% likely to be predicted to have no secondary structure by VIENNA (i.e. ≤ 5% false positive error rate). Conversely, MySST only finds approximately 20% of all sequences predicted by VIENNA to have no structure (i.e. ≈ 80% false negative error rate). This is acceptable in the sense that it provides a more stringent (or conservative) estimate of the secondary structure content than the slower package (see Figure 2). It should be noted that the stringency of this portion of the screening is readily varied by tuning the threshold of allowed secondary structure stability – if instead of accepting only those probe candidates with exactly zero stability, we allow an 0.5 kcal/mol window, the MySST algorithm is 97% accurate in its predictions and finds approximately 50% of those sequences predicted to have stability below the same threshold by VIENNA. In the end, all predictions should be verified by a more exact secondary structure predictor. This procedure validates the MySST predictions without the excessive cost of running the accurate predictor algorithm for all potential probe sequences. (This is one reason why calling the project MOOSE is a bit of false advertising – the selected oligonucleotide probe sequences are not guaranteed to be the *optimal* set with respect to secondary structure content).

Uniqueness:

Each selected probe must bind specifically to its target gene. This requires that each probe sequence be compared against the coding regions of all the other genes, and the stability of hybridization with each of them estimated. This estimate has several ingredients, including a statistical-mechanical model for the thermodynamic stability of a (possibly mismatched) DNA duplex, and a search algorithm. Given two DNA sequences of lengths n and m, there are $\binom{n+m}{n}$ possible gapped alignments of the sequences. Given a scoring scheme, a dynamic programming algorithm can be used to find the optimal alignment with O(nm) operations and O(nm) storage of a particular gene-oligo pair. There are NM such pairs to consider where N is the number of oligos. The total complexity is therefore O(nmNM), which is very large. Heuristic algorithms have been developed which are very likely to find the optimal alignment for biologically relevant scoring systems without performing the full dynamic programming calculation.

In current biological research applications, the most commonly asked sequence comparison question is "Is this sequence likely to be closely related in an evolutionary way to a previously-characterized sequence." A commonly used application for searching a database for sequences that are similar to a given query sequence is BLAST[14,15,16]. The basic BLAST algorithm works by performing an efficient search for exact matches of a minimum size between the query and the database. These exact 'seed' matches are then extended using dynamic programming in a way that allows gapped alignments. This exploits the fact that, in biological applications, one is typically interested in the type of match that has at least one exact match of an appreciable length within it. Matches that don't can't be found at all using BLAST.

As in the case of our screen for secondary structure, we are most interested in an efficient, conservative method for eliminating candidate probes which are likely to hybridize inappropriately. We use the same model and parameters for the stability of DNA duplex that we used in that case, and a similar approach in combining the contributions of short exact matches.

In our implementation, we search for progressively longer exact matches between a candidate oligo and a given gene, computing a running sum of the contribution of these matches to an estimate of the total hybridization stability. As longer matches are found, the contribution of all matches of the next smaller size which they contain are eliminated from the tally. While this algorithm significantly over estimates the stability of a hybrid between the two molecules (because it allows physically unrealizable hybridization structures), we can compute it efficiently, and our hope is that it can be shown to be a usable surrogate for the true stability which is more expensive to compute. The algorithm is conservative in the sense that it will not miss true matches longer than the threshold size, so it quickly identifies genes with essentially no hybridization potential.

In practice, the word sizes we use are broken into three ranges, and a method optimized for use in each range is used. For short words of length n, we use the binary representation of each of the 4^n possible words as the index into a bit vector which encodes the presence or absence of that word in the gene under consideration. Each oligo is then compared to this representation of the gene's contents. For longer words, it's more efficient to construct a hash table that stores the presence or absence in the sequence of the possible words. Words up to a fixed length of 16 are considered in this systematic manner, and each match of length 16 is extended to find all longer matches.

The algorithm also uses several optimizations based on the characteristics of the problem to prune the search space and reduce total running time. The primary observation is that a particular gene-oligo pair which doesn't have a match of size N can't contain a match of size N+1 or greater – therefore, as we iterate from lower to higher match sizes, the list of oligos under consideration in the pairing process rapidly decreases to nothing, often reaching zero before the highest size being hashed is reached (in which case we eliminate the iterations for those sizes altogether). Another small optimization that we implemented is performing the energy calculation for smaller subsumed matches (which must be subtracted from the running total) incrementally within the energy calculation for the subsuming match.

In this way, each oligo which survives the Tm and secondary structure prediction phases of the calculation is compared to each gene, excluding the gene which it is intended to probe. For each oligo, the stability of the most stable predicted cross-hybridization is stored. This allows the selection, for each gene, of the oligo which has the lowest predicted probability of cross-hybridization.

This method may fail for genes which are a part of a family of very similar genes. For such a gene, every oligo considered may well match perfectly against another member of the family. In such cases, the desired output may be an oligo that treats all members of the family as a single species, and has minimum cross hybridization with genes outside the family. Such cases can be detected in a post processing step and treated as a special case at the end of the calculation.

Future Goals

There are several directions in which we plan to improve the current implementation of MOOSE. A major algorithmic goal is to improve load balance in the final (most expensive) stage of the calculation – the uniqueness test. Currently, a form of self-scheduling is used to distribute work in the first stage (T_M and secondary structure calculation). As described in the section on implementation, processing elements "pick up" new genes from a queue as they finish the first two calculations on all of the potential probe oligos from their current gene. Because these calculations are purely dependent on individual oligo sequences, this scheduling method assures load balance for the initial phase. However, because the number of oligos which survive these first two screenings varies from gene to gene (see T_M Calculation above), a significant level of load imbalance can arise in the final stage if each processor only works on those oligos that it found to satisfy the initial screenings. Therefore, a fast and cheap heuristic for the redistribution of oligos to processors should be developed for the final stage of the calculation.

Because MOOSE performs a unique calculation whose global results (i.e. a final set of predicted probe sequences that uniquely identify their targets) are completely novel and ultimately can only be verified by performing the physical microarray experiments, the major future goal of the project is experimental validation of its predictions. Although the results of each of the component calculations can be compared to those of other (perhaps more well accepted) algorithms (e.g. BLAST[17], VIENNA, MFOLD), it must be recognized that these too are only heuristics, and ultimate validation must come from actual experimental evidence. Such data would allow future versions of MOOSE to be improved through a refinement process controlled by "real-life" performance. We recognize that this refinement may require the development of more accurate methods for predicting uniqueness and secondary structure (perhaps employing some form of dynamic programming for local optimizations of potential sequence alignments).

Other proposed "tune-ups" for MOOSE include the addition of an automated post-process validation step using more expensive algorithms to verify the MOOSE output and enhanced predictions in cases where unique probe sequences are unlikely to exist. This problem arises because there exist gene families in which a collection of coding sequences are as much as 98% identical over their entirety. Thus, a probe chosen from any of these is likely to bind all members of the family (as it should). MOOSE would ideally recognize this fact, and optimize selectivity considering binding only to those genes outside the family. In such a case the output should be not just a list of predicted probes, but also a list (associated with each probe sequence) of each target gene in the family that is likely to be bound degenerately.

Implementation

We implemented the MOOSE algorithm in Titanium[18 19 20 21], a high-performance, explicitly-parallel dialect of Java designed at UC Berkeley to support SPMD scientific programming. The language has many advantages, one of which it provides automatic platform-independence across a variety of distributed and shared-memory parallel platforms. The language is sufficiently high-level that the programmer can focus his or her effort on the scientific details of the application, instead of worrying about the problems that traditionally make parallel programming difficult (deadlock, memory management, etc).

High-level design decisions

We made several high-level decisions which affected our system-wide design. The first decision (proposed in BLAST and others), was to use a "packed" memory representation for genetic sequences, where each base is represented using only 2-bits, and w/2 bases are packed into each word of memory (where w is the word size of the machine). This decision provides two distinct advantages. The first advantage is that it provides w/2-way low-level data parallelism in many of the operations performed - for example, given an appropriate choice of bit representation, base complementation can be accomplished on w/2 bases with a single bitwise complement instruction, and a hybridization test can similarly be performed using a bitwise XOR.

The second advantage to the packed base representation is that it reduces the memory requirements for base pair storage by a factor of four, which benefits all levels of the system - main memory requirements, cache performance, and file and network I/O performance. These are important because MOOSE is a very memory-intensive application, and good performance is crucially dependent on cache behavior and main memory usage. In processing the genes for more complex organisms (e.g. the human genome, which has been estimated at over 3 billion base pairs), the file and network I/O performance impact could become significant. MOOSE utilizes a new binary file format we developed (called MGF) that employs this packed base pair representation - we also developed utilities to convert back and forth between MGF format and the traditional FASTA textual gene file format. In addition to the 4x space savings provided by the packed representation, the MGF format removes the text-parsing overhead implied by the FASTA format, and most importantly facilitates random-access to particular genes and even particular gene subsequences in the on-disk representation. This random access enables crucial optimizations in the expression of gene-parallel algorithms, where each processor only needs to read a particular gene fragment that it cares about. Such a format may also be beneficial and applicable in applications (such as gene sequencers) that may wish to write large amounts genetic data in parallel.

Implementation issues – sequential optimizations

The MOOSE algorithm uses three criteria (melting temperature, secondary structure, and uniqueness) to evaluate potential oligos, and these criteria are applied in the order of computational efficiency (see figure 4). This approach is beneficial because each criteria narrows down the number of oligos that need to be considered by the subsequent, more expensive computations. The figure also illustrates typical selectivity statistics for the various phases (the fraction of candidate oligos input to a phase that subsequently get passed on to the next phase) . However, selectivity can vary considerably depending on the user-defined parameters such as optimal melting temperature, acceptable melting temperature range, and secondary structure energy threshold. The user may also specify a minimal number of oligos to be generated by each phase (this is especially important for genes in which there are very few oligos satisfying the user's criteria, and the system must be directed to choose the most favorable of the unqualified oligos in order to end up with at least a few oligos for each gene).

Figure 4 also illustrates the approximate oligo throughput for each of the phases on a single-processor Sun/Ultra-1/170 with 125 MB of memory. The running time for each phase is approximately linear in the number of input candidate oligos (the running time for the uniqueness phase also depends linearly on the size of the genome database, which in this case is the ~6000 gene yeast genome). In order to achieve good running times for each of the phases, MOOSE utilizes several sequential algorithmic optimizations.

The T_m calculation is sequence dependent – it depends on the dinucleotide content of the oligos. Because the temperature calculation runs first (the input is the target gene tail itself) and the candidate oligos are an overlapping set (which means the T_m of the next oligo is mathematically related to the T_m of the current oligo), it is convenient to calculate the temperature incrementally as the oligos are generated from the gene. As the T_m phase iterates over a gene, it keeps the running sum of the dinucleotide entropies and enthalpies in the current oligo – then generating the T_m for the next oligo simply requires adding the effect of the new dinucleotide that entered on one end, and subtracting the effect of the dinucleotide which has exited on the other end (stored in a cyclic buffer for efficiency).

Implementation issues – parallelism

The basic unit of parallel task assignment in our algorithm is the gene - each gene is assigned to a processor which is responsible for finding oligos within that gene - the calculation for each gene can proceed completely independently from the other processors, with very little synchronization and zero communication. The only deep issue that arises is a load balancing question –the melting temperature and secondary structure distributions for individual genes varies widely, which means the oligo yield of the first two phases will also vary in a (generally unpredictable) manner from one gene to the next. This means that a static work partitioning scheme (for example, a cyclic distribution of genes to processors) is likely to produce poor CPU load balance across nodes, because some processors will end up with significantly more work to do.

For this reason, our MOOSE implementation utilizes self-scheduling to dynamically partition the work. In the first phase, genes are kept on a centralized task queue and the processors repeatedly fetch a small number of genes (a work block of a tunable size) from the front of the queue and perform the first two computations (T_m and SS prediction) on them. Depending on the gene lengths and the selectivity realized by the T_m prediction for each particular gene, the various processors will end up performing significantly different numbers of (relatively expensive) secondary structure predictions. The processors that receive the more computationally intensive genes will end up consulting the task queue less frequently, and the resultant load imbalance for the first phase is unlikely to be much more than the average time required to process a single work block. This suggests that the optimal work block would be as small as possible (i.e. a single gene), however there is communication and synchronization overhead associated with fetching work from the task queue that makes larger work blocks more favorable. Figure 5 illustrates the running times and load imbalance for the first phase that result from different work block sizes. As expected, larger work block sizes reduce the synchronization overhead in the task queue (and therefore the total runtime) but increase the load imbalance.

In order for this algorithm to truly scale to a large number of processors, it will probably be necessary to implement a hierarchical, decentralized task queuing strategy, thereby reducing the synchronization overheads by removing the centralized serial bottleneck (however, the current version of MOOSE does not yet implement this). Another area for future work that we've recently discovered is that true scalability requires inserting a global load rebalancing step between the calculation phases to ensure that all the processors perform the uniqueness calculation on approximately the same number of oligos in the second phase. Currently, the processors perform the uniqueness phase on the oligos that happen to reside on that processor following the first phase. Because the number of oligos produced in phase one varies considerably across processors, we observe a substantial load imbalance during the second phase (as illustrated in figure 7).

The running time for the uniqueness calculation can be reasonably predicted from the number of oligos and the size of the gene database, so we plan to implement a work partitioner that seeks to equalize the number of oligos assigned to each processor for the uniqueness phase, while minimizing the amount of communication required to shuffle the oligos between processors in accomplishing this balance. One other desirable constraint that must be satisfied is to keep all the oligos for a particular gene together on a single processor. This problem is similar to a bin-packing problem, with the additional communication minimization constraint – it's unlikely that producing an optimal solution to the distribution problem would be computationally favorable in terms of the total running time (i.e. the time to generate the solution must be less than the time to process the load imbalance which is seeks to correct). However, it should be possible to implement a fast approximation heuristic that produces a reasonable answer that smoothes out the work distribution somewhat.

Figure 7 demonstrates the speedup graph for the algorithm running on the Berkeley NOW[22], processing 128 genes (70,789 candidate oligos) from the yeast genome across a varying number of processors. As expected, the speedup is currently limited by the load imbalance in the second phase, but once this problem is corrected, we expect the speedup to be nearly linear. Phase one speedup for 64 processors is reduced due to the scarcity of work at that level of parallelism and the bottleneck caused by the centralized task queue. Running times for this size problem are about 30 minutes for a single processor.

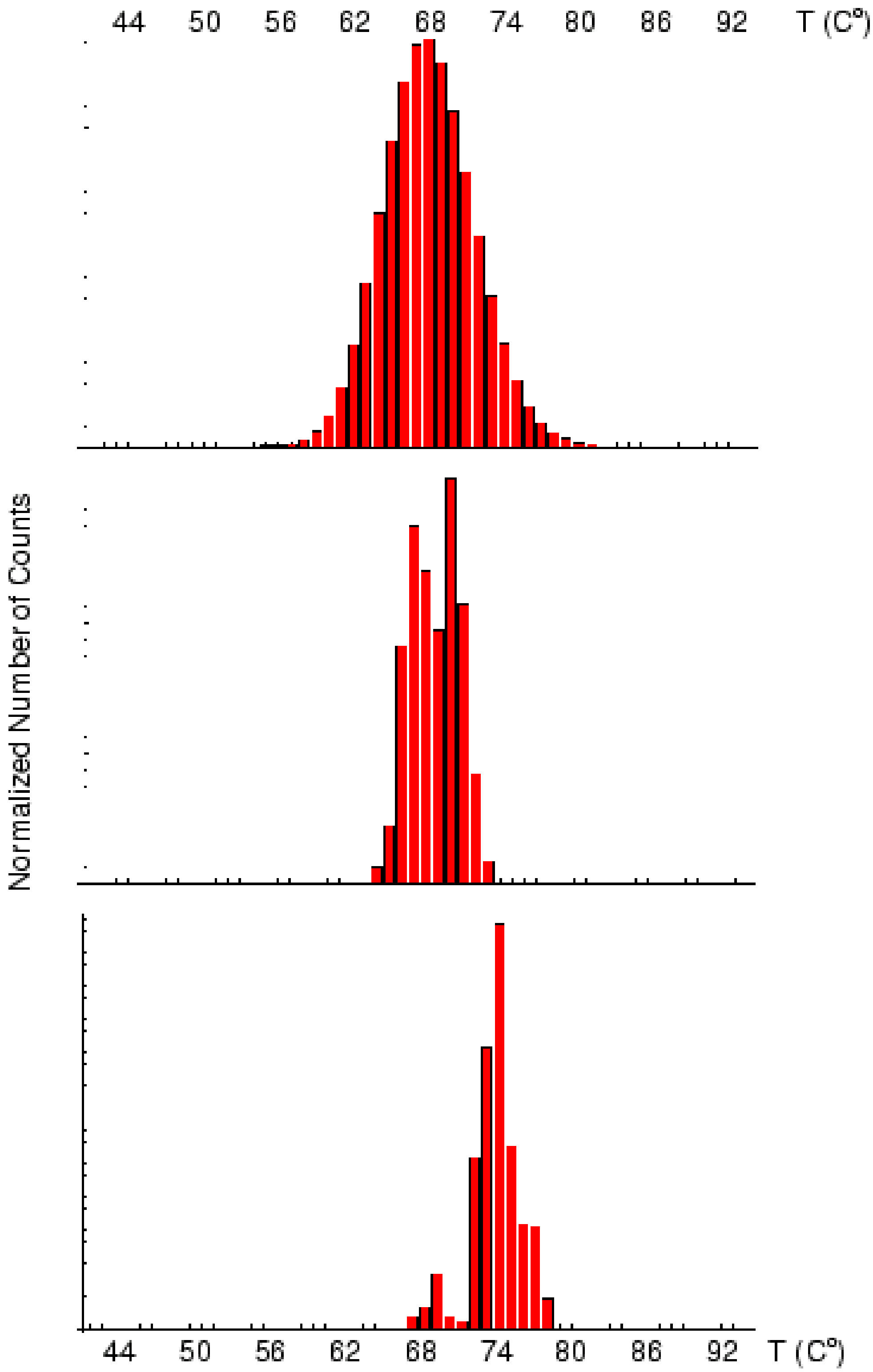

Figure 1: Melting temperature distributions. At top the normalized TM distribution for the entire set of 3.5 M 70-mer oligos is shown. It approximates at Gaussian distribution

peaked at 68°C with full-width-at-half-maximum 9°C. Below, the distributions for two randomly selected genes are shown (each containing 631 70-mer potential oligo probe sequences). It should be noted that distributions from individual gene-tails do not necessarily resemble the global distribution. This means that different genes will have greatly different numbers of potential probe sequences based on the TM screening - this affects the load balance in later stages of the algorithm.

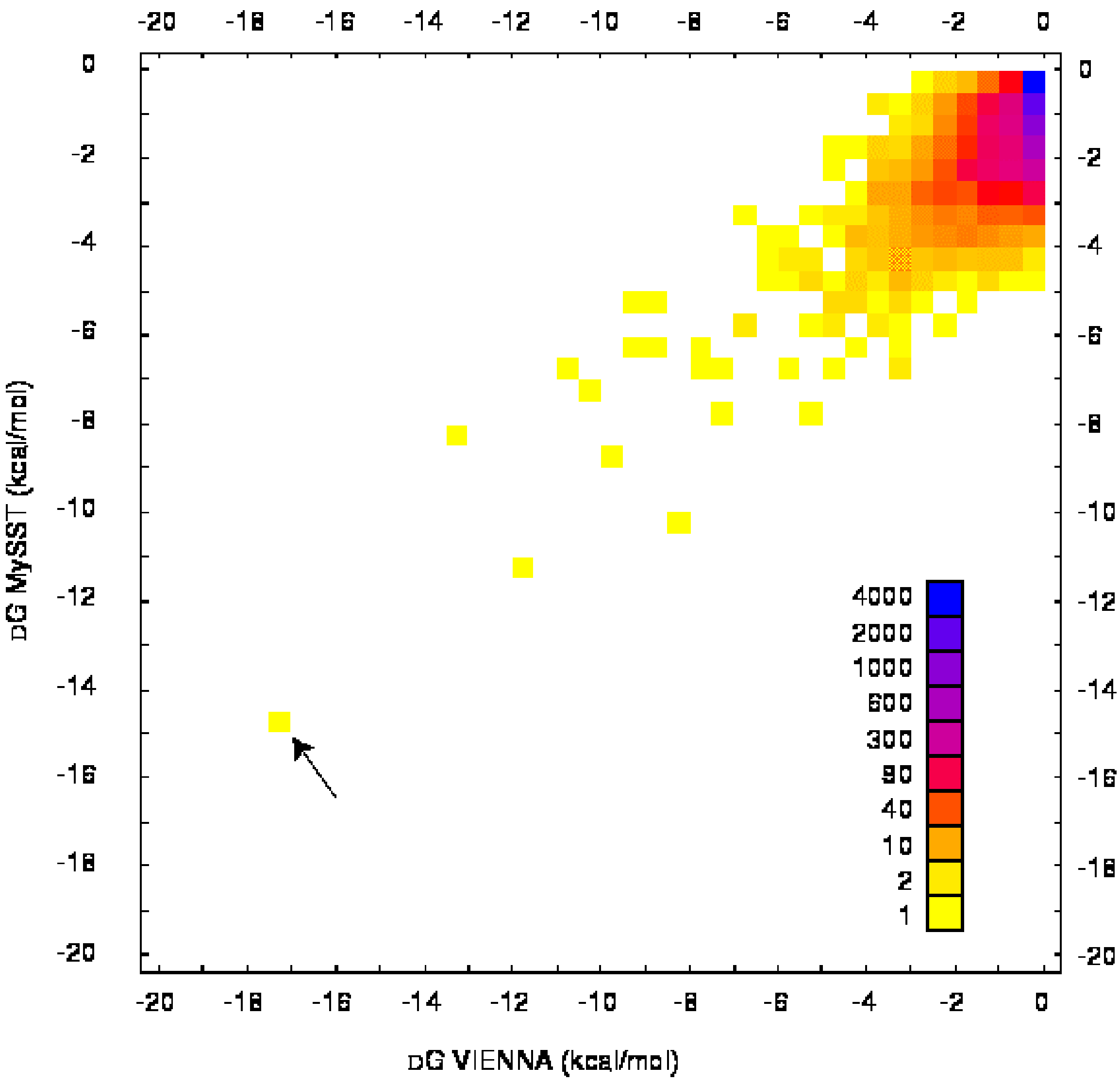

Figure 2: Performance of MySST algorithm relative to VIENNA. The plot is a two-dimensional histogram of the logarithmic free energy distribution for 10000 60-mers sampled from yeast coding sequence. The secondary structure stabilities were calculated at 70°C for both methods. The VIENNA package used RNA folding parameters with G-U base-pairs disabled. The linear correlation coefficient for the raw predictions is 0.68. Interestingly, both algorithms identify the same lowest energy structure (indicated by arrow). As binned here (0.5 kcal/mol bins in each dimension), 97% of the lowest-energy-bin sequences predicted by MySST are also in the lowest-energy bin as predicted by VIENNA. Conversely, 48% of lowest-bin structures predicted by VIENNA are found to be lowest-bin by MySST.

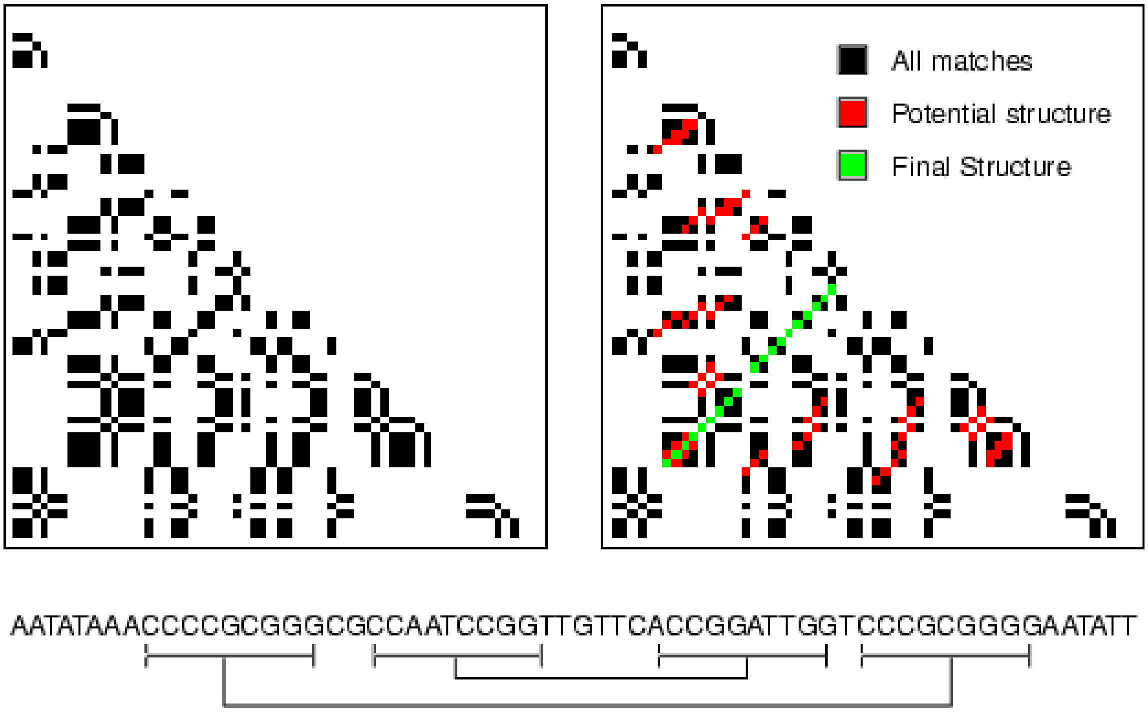

Figure 3: MySST results for a 60-mer sequence. For the sequence given above, the complete M matrix is shown at left (because of symmetry in the forward and reverse sequences only half of the entries are required). At right the potential duplex-forming base pair sequences are shown in red; the final predicted structure in green. This particular sequence contains the lowest energy secondary structure found in a sample of 10000 60-mers chosen from the yeast coding sequence database. The VIENNA fold package found the identical structure for this sequence (which was also the lowest energy structure it found in this sample of the yeast database). The two pairs of duplex-forming regions are indicated in the above sequence.

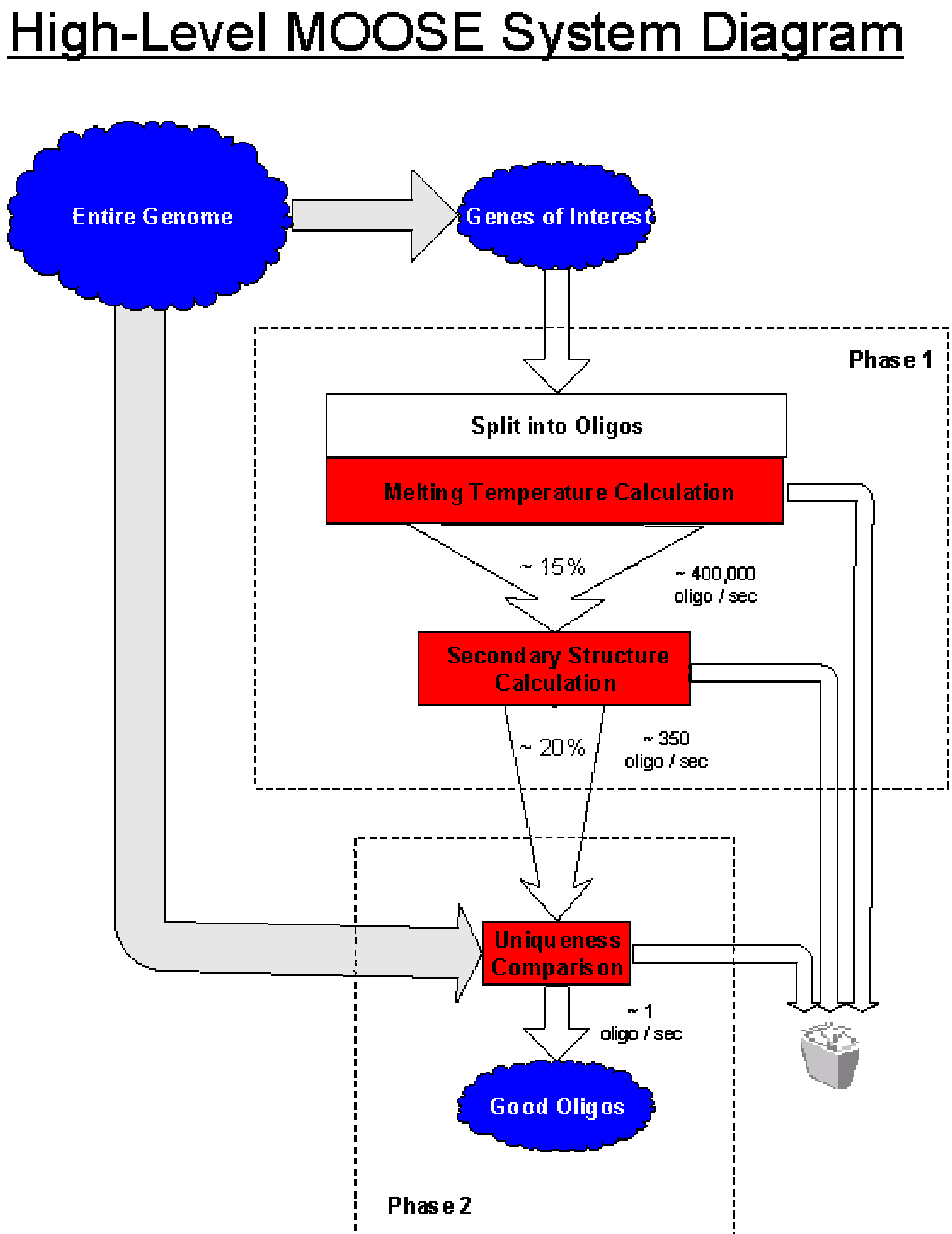
High-Level MOOSE System Diagram
Entire Genome
Genes of Interest
Phase 1
Split into Oligos
Melting Temperature Calculation
~ 15%
~ 400,000
oligo / sec
Secondary Structure Calculation
~ 20%
~ 350
oligo / sec
Uniqueness Comparison
~ 1
oligo / sec
Good Oligos
Phase 2

Figure 4.

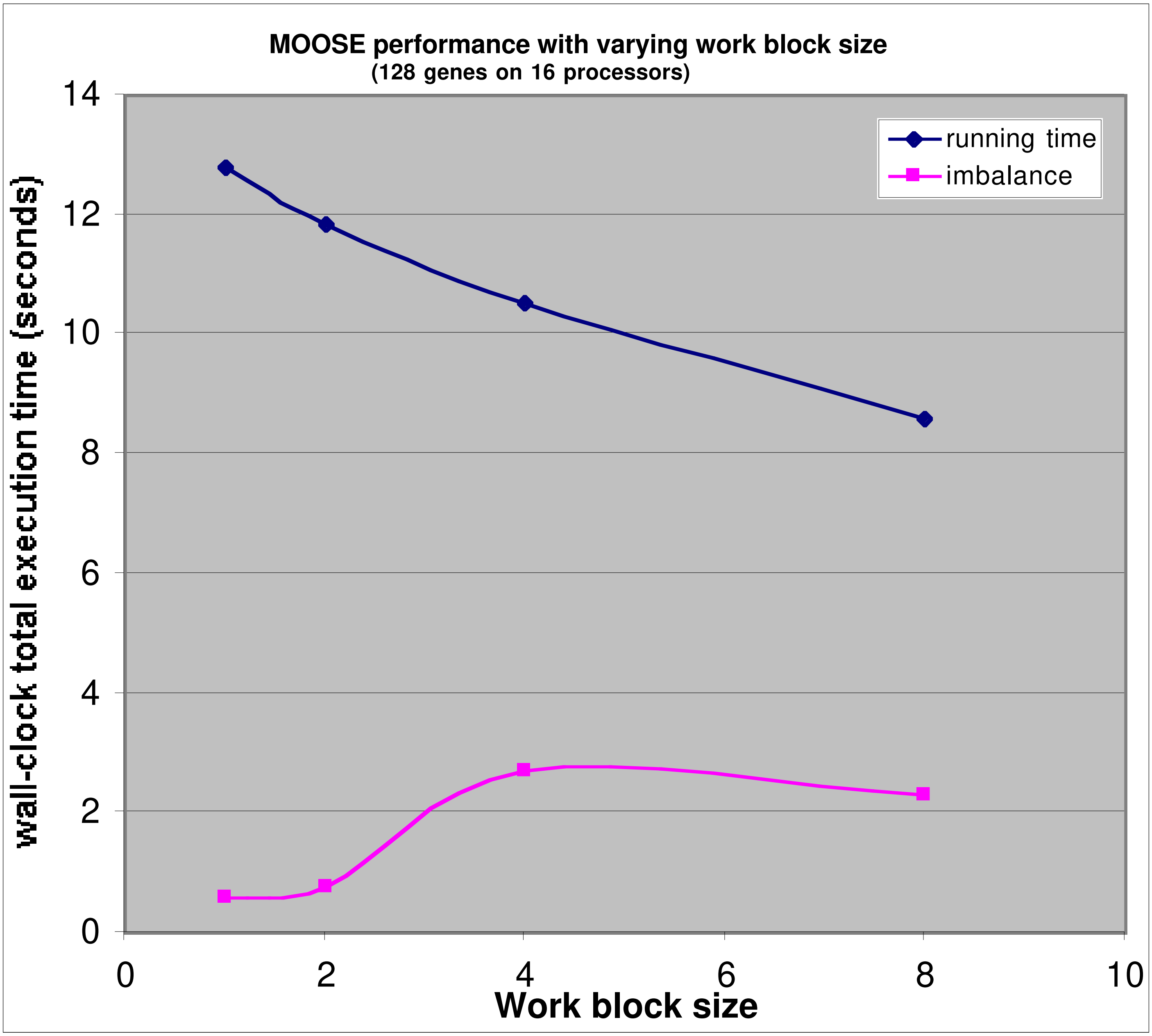
MOOSE performance with varying work block size
(128 genes on 16 processors)
wall-clock total execution time (seconds)
Work block size
running time
imbalance
0
2
4
6
8
10
12
14
0
2
4
6
8
10

Figure 5

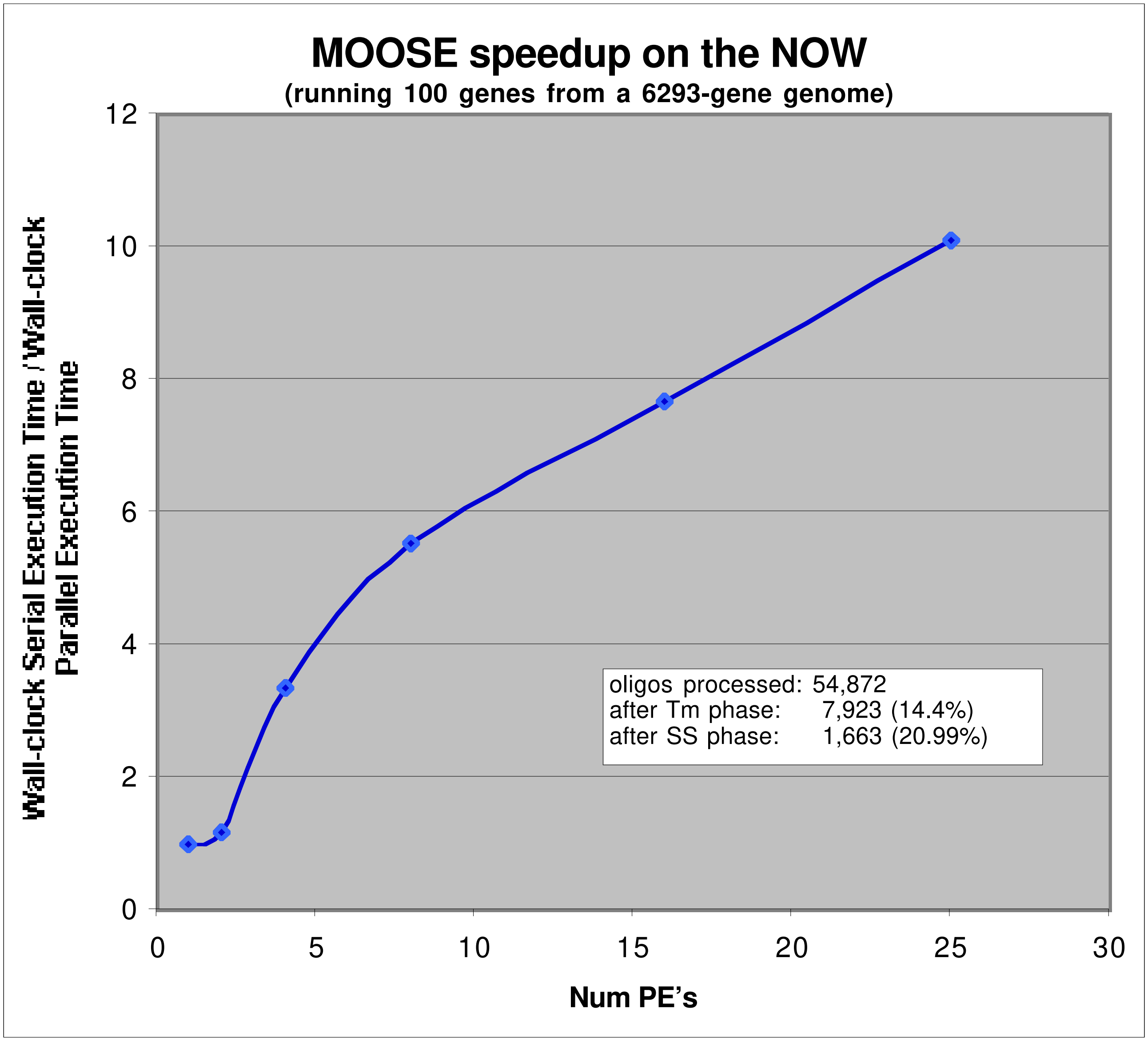
MOOSE speedup on the NOW
(running 100 genes from a 6293-gene genome)
Wall-clock Serial Execution Time / Wall-clock Parallel Execution Time
Num PE's
0
2
4
6
8
10
12
0
5
10
15
20
25
30
oligos processed: 54,872
after Tm phase: 7,923 (14.4%)
after SS phase: 1,663 (20.99%)

Figure 6

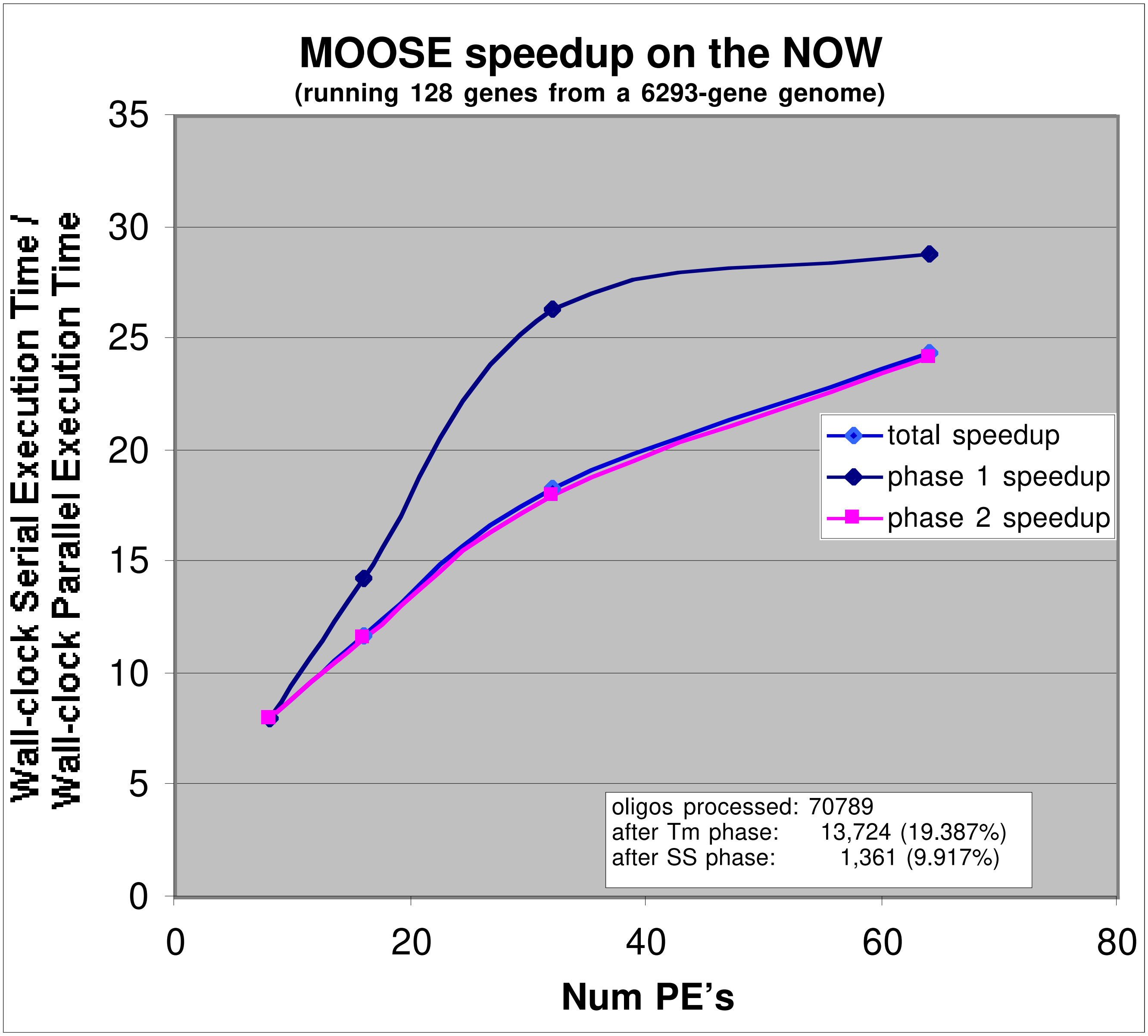
MOOSE speedup on the NOW
(running 128 genes from a 6293-gene genome)
Wall-clock Serial Execution Time /
Wall-clock Parallel Execution Time
Num PE's
0
5
10
15
20
25
30
35
0
20
40
60
80
total speedup
phase 1 speedup
phase 2 speedup
oligos processed: 70789
after Tm phase: 13,724 (19.387%)
after SS phase: 1,361 (9.917%)

Figure 7.

[1] Schena, M, D Shalon, R W Davis & P O Brown (1995) *Science* **270** 467-470.
[2] DeRisi, J *et al.* (1996) *Nature Genet*. **14** 457-460.
[3] DeRisi, J, V R Iyer, P O Brown (1997) Science **278** 680-686.
[4] Wells, R.D., Larson, J.R., Grant, R.C., Shortle, B.R. & Cantor, C.R. (1970) *J. Mol. Biol*. **54**, 465-497.
[5] Crothers, D.M. & Zimm, B.H. (1964) *J. Mol. Biol*. **9**, 1-9.
[6] Tinoco, I., Jr., Uhlenbeck, O.C. & Levine, M.D. (1971) *Nature (London)* **230**, 363-367.
[7] Breslauer, K.J., Frank, R., Blocker, H. & Marky, L.A. (1986) *Proc. Natl. Acad. Sci. USA* **83**, 3746-3750.
[8] Freier, S.M. *et al.* (1986) *Proc. Natl. Acad. Sci. USA* **83**, 9373-9377.
[9] SantaLucia, J., Jr., Allawi, H.T., Seneviratne, PA. (1996) Biochemistry 35, 3555-3562.
[10] SantaLucia, J., Jr. (1998) *Proc. Natl. Acad. Sci. USA* **95**, 1460-1465.
[11] Hofacker, I.L. , Fontana, W. , Stadler, P.F. , Bonhoeffer, S. , Tacker, M. & Schuster P. (1994) *Monatshefte f. Chemie* 125: 167-188.

[13] Walter, A.E., Turner, D.H., Kim, J., Lyttle, M.H., Muller, P., Mathews, D.H. & Zuker,M. (1994*) Proc. Natl. Acad. Sci. USA* **91**, 9218-9222.

[15] Altschul, S F, T L Madden, A A Schaffer, J Zhang, Z Zhang, W Miller & D J Lipman (1997) *Nucleic Acids Research* **25** 3389-3402.
[16] Altschul, S F, W Gish, W Miller, E W Myers & D J Lipman (1990) *J. Mol. Biol.* **215** 403-410.

[18] Yelick, K. et al., "Titanium: A High-Performance Java Dialect", ACM 1998 Workshop on Java for High-Performance Network Computing, Stanford, CA (February 1998).

[19] Hilfinger, P., "Titanium Language Working Sketch, rev 0.22", (September, 1999).
[20] Titanium Project web page. http://www.cs.berkeley.edu/Research/Projects/titanium/

[22] Culler, D. et al., "Parallel Computing on the Berkeley NOW", 9th Joint Symposium on Parallel Processing (1997). NOW Project web page. http://now.cs.berkeley.edu/