Application of high-performance computing to the reconstruction, analysis, and optimization of genome-scale metabolic models

Article *in* Journal of Physics Conference Series · July 2009

DOI: 10.1088/1742-6596/180/1/012025

CITATIONS
23

READS
206

3 authors, including:

Christopher S Henry
Argonne National Laboratory
208 PUBLICATIONS **7,599** CITATIONS

SEE PROFILE

Rick L Stevens
Argonne National Laboratory
256 PUBLICATIONS **25,190** CITATIONS

SEE PROFILE

Journal of Physics: Conference Series **180** (2009) 012025 doi:10.1088/1742-6596/180/1/012025

Application of high-performance computing to the reconstruction, analysis, and optimization of genome-scale metabolic models

Christopher S Henry,[1*] Fangfang Xia,[2*] Rick Stevens[1,2]

[1]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA
[2]Department of Computer Science, University of Chicago, Chicago, IL 60637, USA
[*]These authors contributed equally to this work.

Email: chenry@mcs.anl.gov

Abstract. Over the past decade, genome-scale metabolic models have gained widespread acceptance in biology and bioengineering as a means of quantitatively predicting organism behavior based on the stoichiometry of the biochemical reactions constituting the organism metabolism. The list of applications for these models is rapidly growing; they have been used to identify essential genes, determine growth conditions, predict phenotypes, predict response to mutation, and study the impact of transcriptional regulation on organism phenotypes. This growing field of applications, combined with the rapidly growing number of available genome-scale models, is producing a significant demand for computation to analyze these models. Here we discuss how high-performance computing may be applied with various algorithms for the reconstruction, analysis, and optimization of genome-scale metabolic models. We also performed a case study to demonstrate how the algorithm for simulating gene knockouts scales when run on up to 65,536 processors on Blue Gene/P. In this case study, the knockout of every possible combination of one, two, three, and four genes was simulated in the *i*Bsu1103 genome-scale model of *B. subtilis*. In 162 minutes, 18,243,776,054 knockouts were simulated on 65,536 processors, revealing 288 essential single knockouts, 78 essential double knockouts, 99 essential triple knockouts, and only 28 essential quadruple knockouts.

Introduction

Genome-scale metabolic models represent one important end product of the genome annotation process. These models provide a means of rapidly translating detailed knowledge of thousands of enzymatic processes into quantitative predictions of whole-cell behavior. They have been applied extensively to identify essential genes and genes sets, predict organism phenotypes and growth conditions, design metabolic engineering strategies, and simulate the effects of transcriptional regulation on organism behavior [1-4].

A genome-scale metabolic model of an organism consists of three primary components: (1) a list of the reactions that take part in the organism metabolism including data on reaction stoichiometry and reversibility, (2) a set of gene-protein-reaction (GPR) mappings that capture how genes in the organism encode enzymes and how these enzymes catalyze metabolic reactions, and (3) a biomass objective function that indicates which small molecules must be produced for an organism to grow and

divide [1]. All of these components are used in a method called flux balance analysis (FBA) to simulate organism metabolism in a set of specified environmental conditions.

FBA involves the use of linear optimization to define the limits on the metabolic capabilities of a model organism by assuming that the interior of the cell exists in a quasi-steady state [2-5]. This quasi-steady-state assumption is enforced by a set of linear mass balance constraints written for each metabolite included in the model.

$$\boldsymbol{N}\cdot\boldsymbol{v} = \boldsymbol{0} \quad (1)$$

In the mass balance constraints (Eq. 1), $\boldsymbol{N}$ is the m x r matrix of the stoichiometric coefficients for the r reactions and m metabolites in the genome-scale metabolic model, and $\boldsymbol{v}$ is the r x 1 vector of the steady-state fluxes through the r reactions in the model. Bounds are placed on the reaction fluxes ($\boldsymbol{v}$) depending on the reversibility of the reactions.

$$\text{-100 mMol/gm CDW hr} \leq v_{i,reversible} \leq \text{100 mMol/gm CDW hr} \quad (2)$$

$$\text{0.0 mMol/gm CDW hr} \leq v_{i,irreversible} \leq \text{100 mMol/gm CDW hr} \quad (3)$$

These mass balance constraints and reaction flux bounds form a set of underdetermined linear equations with many possible solutions. Because these equations are underdetermined, an optimization criterion is used to capture the most physiologically relevant region of the solution space. The optimization criteria vary depending on the application, but the most common criterion is the maximization of growth yield [5, 6]. Maximum growth yield is simulated by maximizing the flux through the biomass reaction in the model while the uptake of nutrients is fixed at a specific ratio. This is a meaningful optimization criterion because organisms have been observed to grow at the maximum predicted yield when nutrients are plentiful [7].

The FBA formulation described here forms the core of many different algorithms for the reconstruction, analysis, and optimization of genome-scale metabolic models. For this reason, software developed to run FBA in a high-performance computing (HPC) environment may be directly applied to solving many different problems. FBA is also an attractive algorithm to run with HPC because only a small amount of data is required to fully define the variables, constraints, and objective necessary to simulate even the largest genome-scale models, and once FBA has been run, the only result that must be returned is the objective and the list of reaction fluxes. Thus FBA requires minimal input and output, thereby improving scalability. Additionally, once the constraints and variables defining the FBA mass balances for a single genome-scale metabolic model have been initially loaded, many different simulations can be performed simply by adjusting the bounds on the loaded variables and repeating the optimizations. Thus, much work can be done without loading additional data, further improving scalability.

Here we describe the various FBA-based algorithms that currently exist for the reconstruction, analysis, and optimization of genome-scale metabolic models. We focus in particular on algorithms that are good candidates for processing in HPC, and we discuss the implications of running these algorithms in parallel. As a case study, we implemented the FBA algorithm for gene knockout simulation using MPI on Blue Gene/P, and we used this algorithm to simulate the 10^{10} possible quadruple gene knockouts for the *i*Bsu1103 genome-scale model of *B. subtilis*. This represents the first time that the essential combinations of up to four genes have been identified in *B. subtilis*.

1. Model reconstruction algorithms

In order to run FBA simulations in HPC for a large variety of different organism, genome-scale metabolic models must first be created for those organisms. Fortunately opportunities now exist for the use of HPC to accelerate this process. *Reconstruction* is the word used to describe the process of producing a functioning genome-scale model of an organism starting from the sequence of nucleotides in the organism's genome. This process includes gene calling, annotation, literature data mining, reaction mapping, biomass objective function assembly, error correction, and gap filling. Until recently, this process required years of manual effort to complete, and as a result, the rate of development of new genome-scale models lagged far behind the rate at which new genomes were being sequenced (figure 1) [8]. Fortunately, methods have recently emerged that expedite many of the

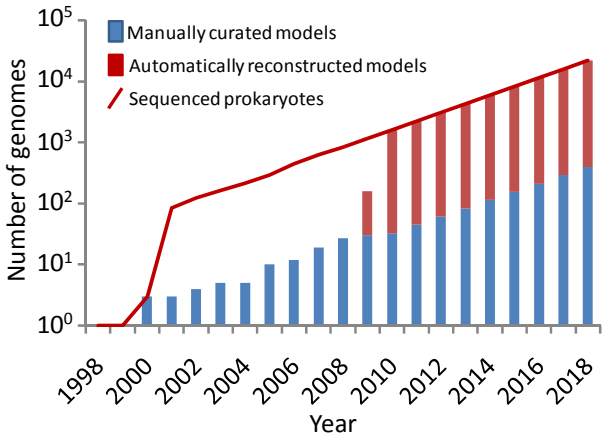

Figure 1. Past and projected rates of release of new genome sequences and new genome-scale metabolic models. Using manual reconstruction, the number of available models (blue bars) lags far behind the number of sequenced genomes (red line). However, with the introduction high-throughput reconstruction and the application of HPC to high-throughput reconstruction, model reconstruction can rapidly catch up to and keep pace with genome sequencing (red bars).

steps involved in the reconstruction process [9-16], and these methods may now be combined into a high-throughput, genome-scale reconstruction pipeline. With this high-throughput reconstruction pipeline, it will be possible to build up-to-date genome-scale models of every sequenced prokaryote on a monthly basis (figure 1). However, this lofty goal will require significant computational power to accomplish.

The rate limiting *computational* step in the high-throughput reconstruction process is an FBA-based algorithm called *auto completion*. Initial genome-scale metabolic models consisting only of reactions associated with genes, spontaneous reactions, and a biomass objective function always contain gaps that prevent them from producing all of the building blocks necessary for growth. As a result, these models are incapable of simulating growth using FBA. In auto completion, a mixed integer linear optimization problem (MILP) is solved to identify the minimal set of reactions that must be added to the initial model in order for every biomass component to be produced [16]. Often, multiple equally optimal minimal reaction sets satisfy these criteria. Recursive MILP [17] may be used to identify each of these solutions, and every solution identified can be preserved until the most physiologically realistic solution has been manually selected. Depending on the number of gaps in the initial model, run time for this algorithm can range from one minute to 24 hours. The long run time combined with the minimal input and output data size (table 1) means that scalability will not be an issue for the auto completion algorithm. The auto completion algorithm could be run in parallel to produce functioning genome-scale models for all ~800 prokaryotes with completely sequenced genomes today. Use of a parallel MILP solver [18] will make it possible to spread the work involved in producing each model across up to eight processors while retaining scalability. In this manner, the auto completion algorithm could be run on a maximum of 6,400 processors. Additionally, we anticipate the number of complete genome sequences to continue growing rapidly over the next decade (figure 1), meaning number of processors that could be consumed by the auto completion algorithm will also grow rapidly.

We note that the genome annotation and literature data-mining steps of the reconstruction process will probably always require significant manual effort to complete. However, the Rapid Annotation using Subsystems Technology (RAST) method developed within the SEED annotation framework [11, 19, 20] makes it possible for these steps to be performed for every known genome simultaneously and to be extended to new genomes with minimal effort. In RAST, most well studied biological functions have been divided into a set of subsystems (e.g., glycolysis) that are assigned to an expert annotator. That annotator performs the genome annotation and literature data-mining steps of the reconstruction process for that subsystem using a set of comparative genomics tools to project curations and annotations across all known genomes simultaneously. Whenever a new genome sequence (or set of sequences) is integrated into the SEED, annotators will curate the addition of those genomes to each of their subsystems to ensure that the subsystem is correctly propagated into the new genome. Similarly,

as new literature emerges, annotators study this literature to determine how the reported results will impact the annotations in their subsystems. In this manner, annotation and literature data-mining efforts are able to keep pace with the sequencing of new genomes, making high-throughput reconstruction possible despite the need for some manual curation.

Table 1. HPC considerations model reconstruction, analysis, and optimization algorithms

Algorithm	Input	Output	Run Time per Simulation	Approximate Thread Count
Reconstruction algorithms				
Automated completion	~2 MB/thread	1 KB/thread	1-24 hours	$O(10^3)$ genomes
Optimization algorithms				
Gap filling	~2 MB/thread	1 KB/thread	1-60 min	O(10) models $O(10^2)$ errors
Gap generation	~2 MB/thread	1 KB/thread	1-4 hours	O(10) models $O(10^2)$ errors
Analysis algorithms				
Growth simulation	~1 MB broadcast	8 B/thread	~0.01 sec	$O(10^3)$ models $O(10^2)$ media
Gene knockout	~1 MB broadcast	8 KB/thread	~0.01 sec	$O(10^3)$ models $O(10^2)$ media $O(10^{3k})$ knockouts
Gene knockout with thermodynamic constraints	~2 MB broadcast	8 KB/thread	~3 hours	$O(10^3)$ models $O(10^2)$ media $O(10^{3k})$ knockouts
Reaction classification	~2 MB broadcast	16 KB/thread	~2 minutes	$O(10^3)$ models $O(10^2)$ media
Reaction classification with thermodynamic constraints	~2 MB broadcast	16 KB/thread	~6 hours	$O(10^3)$ models $O(10^2)$ media
Metabolic engineering	~2 MB /thread	1 KB/thread	1-4 hours	$O(10^3)$ models $O(10^2)$ media $O(10^2)$ products $O(10^2)$ raw materials

2. Model analysis algorithms

High-throughput implementation of the auto completion algorithm will enable the AGRO pipeline to produce thousands of new genome-scale metabolic models, paving the way for the use of HPC in the analysis of these models as well. Numerous FBA-based algorithms exist for analyzing genome-scale models to produce predictions of growth conditions, essential genes, metabolite concentration ranges, and metabolic engineering strategies. Application of HPC to these algorithms will produce a wealth of valuable new data.

2.1. Growth simulation

All published genome-scale metabolic models include one or more intracellular compartments separated from an extracellular compartment by a cell membrane. Only nutrients with explicit transport reactions in the model are allowed to pass into the intracellular compartment(s) from the extracellular compartment. While mass balances are maintained in the intracellular compartments, metabolites in the extracellular compartment may have a net consumption or production by the cell. This compartmental structure makes it possible to use FBA to simulate growth in a variety of environments by adjusting the set of compounds that are allowed to be consumed by the cell from the

extracellular compartment. For example, when modeling rich media, all amino acids, nucleotides, and many cofactors may be consumed from the environment. When modelling minimal media, only glucose, ammonia, phosphate, sulphate, and various ions may be consumed from the environment.

The simplicity of the linear optimization algorithm means that the growth of a single model may be simulated in many different media conditions on a single processor with great speed (~70 media conditions/second). The slow step of the algorithm is the initial loading of the model into the optimization solver, so it is essential for efficiency that simulation software interacts with the optimization software through an API as this allows variable bounds and problem objectives to be adjusted on the fly without reloading the model. This simulation speed indicates that a very large number of simulations must be performed in order to justify the application of HPC.

The 324 media conditions included in the Biolog phenotyping arrays currently represent the most significant large set of media conditions that would be informative to run growth simulations on with a large set of models. Biolog phenotyping arrays are experimental devices that measure the ability of an organism to respire in hundreds of distinct media environments [21]. Each media environment in the phenotyping array contains a different nutrient, making the Biolog array useful as a means of determining whether an organism has a transporter and a catabolic pathway for many different carbon, nitrogen, phosphate, and sulphate sources. Even with thousands of models and hundreds of media conditions, growth simulation alone generates insufficient computational demand to require HPC. However, growth simulation is still a feasible application for HPC when combined with other algorithms. For example, gene knockout simulations must often be performed in a variety of media conditions.

2.2. Gene knockout simulation

Gene essentiality prediction is one of the primary uses for genome-scale metabolic models. Knowledge of essential or coessential genes in pathogenic organisms is valuable for identifying target proteins for the development of new antibacterial agents. Knowledge of essential and coessential genes in industrial organisms is also useful for ensuring that proposed metabolic engineering strategies avoid disruption of these genes.

To identify essential and coessential genes, gene knockouts are simulated in the models by using the GPR associations to identify reactions exclusively encoded by knocked out genes, setting the bounds on the flux through these reactions to zero, and maximizing the flux through the biomass reaction. If the maximum flux through the biomass reaction is zero, the knocked out gene(s) are predicted to be essential (for a single gene) or coessential (for multiple genes). The nutrients present in the environment during the knockout have a significant effect on gene essentiality, and for this reason knockouts often must be simulated in multiple media conditions. Like the growth simulation, knockout simulations can be performed very rapidly (~50 knockouts/sec), meaning any application of HPC to gene knockout simulation must involve billions of distinct simulations on thousands of processors. For example, HPC could be applied to the simulation of every possible single, double, and triple gene knockout in every sequenced pathogen on a hundred different media conditions representing distinct physiological environments for infection. With 10^2 pathogens, 10^2 media conditions, and 10^6 double knockouts, this scenario involves 10^{10} distinct simulations. With perfect scalability on 100,000 processors, the run time would be approximately 24 minutes.

2.3. Reaction classification

The gene knockout simulation algorithm provides a means of classifying the genes included in a genome-scale model. A similar FBA-based algorithm exists to classify the reactions in the model. This algorithm, called *flux variability analysis* (FVA), involves minimizing and maximizing the flux through each individual reaction in the model; model reactions are then classified based on the minimum and maximum fluxes calculated [22]. If the min and max are both positive or both negative, the reaction is essential; if they are both zero, the reaction is nonactive; if one is zero and the other is

nonzero, the reaction is nonessential but active; and if one is positive and the other is negative, the reaction is nonessential but reversibly active.

The HPC scenario for the FVA algorithm is essentially identical to the scenario for single gene knockout simulation. Both algorithms have the same run time and require approximately the same number of simulations. However, it is sometimes useful to combine FVA with gene knockout simulation to study how reaction behavior is affected by gene knockout, which involves $O(10^6)$ simulations per model studied.

2.4. Thermodynamics-based metabolic flux analysis

One issue with standard FBA that can significantly impact the accuracy of the predictions generated is that thermodynamically infeasible flux profiles still satisfy the mass balance constraints and flux bounds in FBA. In thermodynamics-based metabolic flux analysis (TMFA), the mass balance constraints of FBA are supplemented by a set of thermodynamic constraints (Eq. 4) in order to eliminate thermodynamically infeasible fluxes from the feasible solution space [23]. With some additional modifications, these constraints also allow for the exploration of feasible ranges for metabolic concentrations [23].

$$\left(\sum_{j}^{m} n_{i,j} U_i \right) v_i < 0 \quad (4)$$

In the thermodynamic constraints, U_j' is the chemical potential of metabolite j, $n_{i,j}$ is the stoichiometric coefficient for compound j in reaction i, and v_i is the flux through reaction i. Because compounds shared by multiple reactants must have the same potential, these constraints prevent thermodynamically infeasible reaction sets from carrying flux. Unfortunately, these constraints are also nonlinear, forcing the problem to be formulated as either a mixed integer linear program or a quadratic program. While the algorithms for growth simulation, gene knockout, and FVA may all be supplemented with thermodynamic constraints to improve accuracy, the nonlinear formulation of TMFA decreases the rate at which simulations may be run, from 70 simulations per second to 0.25 simulations per second. Of course, this also improves the parallel scalability of these algorithms and increases the need for HPC.

2.5. Metabolic engineering

One of the major industrial applications for genome-scale metabolic models is the design of metabolic engineering strategies for inducing the production of organic chemicals such as ethanol, 1,3-PDO, and 3HP by microorganisms [24]. Often the natural metabolic network of an organism will be missing one or more of the enzymatic steps required to produce a desired product. The *pathway addition* algorithm involves identifying the minimal number of new reactions that would have to be engineered into an organism to allow for the production of a desired product from a specified raw material with a minimum yield. Mathematically this algorithm is identical to the automated completion algorithm; uptake of the selected raw material is fixed at one; production of the desired product is constrained to be greater than the minimum yield; and the number of nonnative reactions allowed to carry flux in the final solution is minimized. In the HPC scenario for this algorithm, the pathway addition algorithm could be run for 10^2 different raw materials, 10^2 different potential products, and 10^3 different organisms. In total, 10^7 simulations would be required, each with a run time ranging from one minute to one hour. Completion of this HPC run would provide the following insights: the ideal host organism to produce each desired product, the ideal raw material to produce each desired product from, the maximum yield attainable for each product from each raw material in each organism, and the number of new enzymes that would have to be added to produce each product in each organism.

The OptStrain [25] and OptKnock [26] algorithms are also used extensively to design metabolic engineering strategies. The algorithms take the problem a step further by determining how the existing metabolism of an organism must be modified to drive flux from a specified raw material towards a

desired product. These algorithms are also MILP optimizations that require one minute to one hour to complete.

3. Model optimization algorithms

One of the major uses for the gene knockout and growth simulation algorithms is the validation of genome-scale metabolic models against experimentally determined gene essentiality data [27] and Biolog phenotyping array data [21]. Large-scale gene knockout and phenotyping experiments have produced a tremendous amount of experimental data that may be compared with the model predictions, and the validation that this comparison provides is important not only for assessing model accuracy but also for assessing the accuracy of the annotations on which the models are based.

Inconsistencies between model predictions and experimental data indicate that the model and possibly the underlying annotations contain an error that must be identified and fixed. The large size of genome-scale models, however, makes the manual identification of errors a monumental task. In order to assist in this task, the GrowMatch algorithm was introduced [9]. This algorithm attempts to automatically repair model errors that cause prediction inconsistencies by adding or removing reactions from the model. The GrowMatch algorithm consists of two stages, each representing a potential application for HPC: gap filling, and gap generation.

3.1. Gap filling

The gap-filling stage of GrowMatch attempts to correct prediction errors for conditions where growth is not predicted to occur *in silico* while growth is observed *in vivo* (called false negative predictions). This type of inconsistency is corrected by either adding one or more new reactions to the model or converting existing irreversible reactions into reversible reactions. The gap-filling algorithm is an MILP identical to the automated completion algorithm, but the run time is typically shorter (one minute to one hour) because of the smaller size of the gaps being filled. Also, like the auto completion algorithm, the gap-filling algorithm often produces multiple equally optimal solutions for filling each gap in the model that causes an incorrect prediction. Once the gap filling has been completed for every false negative prediction condition in the model, a reconciliation optimization must be performed [12] to select the optimal combination of all solutions that corrects as many errors as possible with minimal additions to the model. In the HPC scenario, the gap-filling algorithm would be run in parallel for all conditions with false negative predictions in all models with available data. Unoptimized models have 10^2 false negative predictions on average, and currently experimental gene essentiality or Biolog phenotyping array data exists for 10 models. Thus, up to 10^3 simultaneous simulations can be running for one minute to one hour. However, the number of organisms with available experimental data is likely to grow rapidly over time, which will result in a similar growth in the number of gap-filling simulations required by the GrowMatch process.

3.2. Gap generation

The gap generation stage of GrowMatch attempts to correct prediction errors for conditions where growth is predicted to occur *in silico* while growth is not observed *in vivo* (called false positive predictions). This type of inconsistency is corrected by either removing one or more reactions from the model or converting existing reversible reactions into irreversible reactions. Like the gap-filling algorithm, the *gap generation* algorithm is an MILP, but the problem formulation (which is described in detail elsewhere [9, 12]) is much more complex. As a result, the run time is longer, ranging from one to four hours. Like the gap-filling algorithm, the gap generation algorithm often produces multiple equally optimal solutions, and a reconciliation optimization is necessary [12] to identify the optimal combination of all alternative solutions that corrects as many errors as possible. Also like gap filling, the HPC scenario for *gap generation* involves running in parallel for all conditions with false positive predictions in all models with available data, adding up to a total of 10^3 required simulations.

4. Case study: application of HPC to gene knockout simulation

As a case study for the application of HPC to an FBA algorithm, we implemented the gene knockout simulation algorithm using MPI on the Blue Gene/P supercomputer located at Argonne National Laboratory. The Blue Gene/P consists of 40 racks each containing of 1,024 chips with four 850 MHz PowerPC 450 processors on each chip, for a total of 163,840 processors. Blue Gene/P was selected for our case study because the massive number of available processors presents a unique opportunity to demonstrate the scalability characteristics of an FBA algorithm on a truly large scale.

B. subtilis was selected for this study because it is one of the most well studied prokaryotes (second only to *E. coli*) and because knowledge of essential combinations of genes will be beneficial to continuing efforts to produce a minimal strain of *B. subtilis* [28, 29]. The knockout simulations were performed by using the largest and most up-to-date model of *B. subtilis* available, *i*Bsu1103[12]. This model captures 1,103 of the 4,105 genes in the *B. subtilis* genome, meaning that the total pool of genes considered in our gene knockout analysis will be 1,103.

The gene knockout simulations were carried out in multiple stages, with the single gene knockouts performed first, followed by the double, triple, and quadruple gene knockouts. This multistage strategy was necessary because essential gene combinations identified in earlier stages had to be filtered out of the combinations explored in later stages (table 2). For example, it is unnecessary to simulate double gene knockout combinations that include genes found to be essential during the single knockout study because double knockouts involving essential genes will always be essential as well.

Table 2. Gene knockout combinations considered at each stage of the knockout algorithm

	Single KOs	**Double KOs**	**Triple KOs**	**Quadruple KO**
Naïve estimate	1,103	607,753	223,045,351	61,337,471,525
Estimate based on single KO		331,705	89,892,055	18,248,087,165
Estimate based on double KO			89,870,917	18,243,796,151
Estimate based on triple KO				18,243,776,054
Lethal knockouts identified	288	78	99	28

The parallel FBA algorithm was implemented in two versions that exploit two levels of parallelism respectively: fine-grained loop-level parallelism (v1) and coarse-grained embarrassing parallelism (v2). In both approaches, a master node is used to load input files from the file system and broadcast the buffers to every slave node. In the fine-grained approach, the master node loops over the whole set of gene knockout combinations and dispatches a subset to a slave node whenever it becomes available. The granularity (G) is decided based on the total number of processors to strike a balance between minimizing interprocessor communication and balancing slave workload. The actual job sizes are randomly drawn from Uniform ($G/2$, $3G/2$) to avoid clashes of communication. In the coarse-grained approach, instead of getting their jobs from the master node, each slave determines the work that must be performed on based on the slave's rank; the slaves skip through the universal list of genes knockout combinations to find the knockouts they must simulate.

When applied to the simulation of every possible triple gene knockout, the fine-grained loop-level parallelism (v1) approach yielded a high peak performance in moderate parallel settings but scaled poorly beyond 1,024 processors because of the significant interprocessor communication involved in this approach. The coarse-grained embarrassing parallelism (v2) approach initially took a slight performance hit compared with the v1 algorithm because of the overhead involved in generating the complete knockout list on every node. However, this algorithm gave near linear scalability up to 4,096 nodes (figure 2). Because every possible combination of single, double, and triple gene knockout could be simulated in under 10 minutes on 4,096 nodes (one rack of Blue Gene/P running in virtual node mode), it is not unreasonable to expect the coarse-grained approach to scale well beyond 4,096 nodes since at this run time overhead begins to rival computation. We confirmed this hypothesis by

simulating quadruple gene knockouts on 65,536 in 162 minutes. The throughput from the quadruple knockout run scaled almost linearly with all of the triple knockout runs (figure 2).

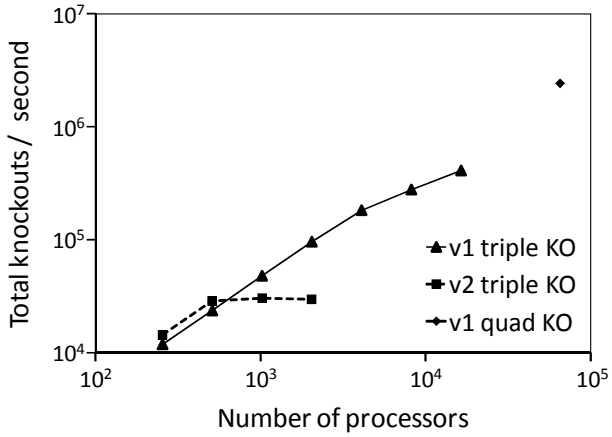

Figure 2. Scalability plot for gene knockout simulations. Here we show how the total number of knockouts simulated every second scaled with the number of processors used in the simulations for multiple parallel algorithms. Algorithm v1 scaled linearly up to 4,096 processors when simulating all possible triple knockouts and up to 65,536 processor when simulating all possible quadruple knockouts.

In the first stage of our knockout study 1,103 single gene knockouts were simulated in *B. subtilis*, resulting in the identification of 288 essential genes. In the next stage 331,705 double gene knockouts were simulated, 78 of which were found to be lethal combinations. In the third stage 89,870,917 triple gene knockouts were simulated with 99 combinations identified as lethal. In the final stage 18,243,776,054 quadruple knockouts were simulated with only 28 lethal combinations identified (table 2). The complete list of the essential gene sets in *B. subtilis 168* identified during this case study can be found here: http://bioseed.mcs.anl.gov/~chenry/BSubEssentials.txt.

5. Conclusions

Numerous opportunities exist for the application of HPC to the reconstruction, analysis, and optimization of genome-scale metabolic models. The field of biology is evolving rapidly, and high-throughput techniques for genome sequencing, gene knockout and mutation, and phenotype analysis are producing an explosion of new experimental data. HPC has an important role to play in ensuring that analytical techniques like FBA can keep up with this explosion. The analysis presented here shows that only with the computational power provided by HPC will the FBA algorithms existing today be capable of handling the explosion of biological data anticipated in the near future.

The results of the quadruple knockout simulation case study clearly show that FBA algorithms can scale well on massively parallel architectures like Blue Gene/P. The caveat to this statement is that sufficient work must be available to drown out the overhead involved in distributing and initializing problem data on a large number of processors. For standard linear programming simulations like the gene knockout algorithm, results indicate that linear scaling can be achieved if there are at least 20,000 simulations per processor. However, we expect that FBA algorithms involving MILP formulations (like TMFA, GrowMatch, or auto completion) will scale linearly with far fewer simulations per processor because of the longer run times involved.

The biological results of the quadruple knockout case study were surprising, as one might intuitively expect to see more lethal double knockouts than lethal single knockouts, more lethal triple knockouts than lethal double knockouts, and so on. Instead the simulation results indicate that *B. subtilis* is remarkably robust to multiple gene knockout. There are two explanations for this nonintuitive result: (1) a large fraction of the essential metabolic functions in *B. subtilis* are associated with essential genes, meaning they have no genetic redundancy, and (2) when a metabolic function required for growth does have genetic redundancy or biochemical redundancy, it is very redundant, requiring three or more knockouts before functionality is lost. Apparently, the increased robustness that is gained by preserving redundant copies of the genes encoding essential functions is not worth

the additional cost of copying and maintaining those redundant copies. Thus it is likely that when redundancy does occur in the genome, it exists to increase flexibility in the expression of the functions related with these redundant genes, rather than as a mechanism to enhance robustness to mutation.

Acknowledgments
This work was supported in part by the U.S. Department of Energy under contract DE-ACO2-06CH11357. We thank the Argonne Leadership Computing Facility and INCITE program for compute time spent on Blue Gene/P. We thank the SEED development team for advice and assistance in using the SEED annotation system.

References
1. Edwards JS, Palsson BO: The Escherichia coli MG1655 in silico metabolic genotype: its definition, characteristics, and capabilities. *Proceedings of the National Academy of Sciences of the United States of America* 2000, 97:5528-5533.
2. Papoutsakis ET, Meyer CL: Equations and calculations of product yields and preferred pathways for butanediol and mixed-acid fermentations. *Biotechnology and Bioengineering* 1985, 27:50-66.
3. Jin YS, Jeffries TW: Stoichiometric network constraints on xylose metabolism by recombinant Saccharomyces cerevisiae. *Metab Eng* 2004, 6:229-238.
4. Varma A, Palsson BO: Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia-coli W3110. *Applied and Environmental Microbiology* 1994, 60:3724-3731.
5. Varma A, Palsson BO: Metabolic capabilities of Escherichia-coli. 2. Optimal-growth patterns. *Journal of Theoretical Biology* 1993, 165:503-522.
6. Varma A, Palsson BO: Metabolic capabilities of Escherichia-coli.1. Synthesis of biosynthetic precursors and cofactors. *Journal of Theoretical Biology* 1993, 165:477-502.
7. Edwards JS, Ibarra RU, Palsson BO: In silico predictions of Escherichia coli metabolic capabilities are consistent with experimental data. *Nature Biotechnology* 2001, 19:125-130.
8. Feist AM, Herrgard MJ, Thiele I, Reed JL, Palsson BØ: Reconstruction of Biochemical Networks in Microbial Organisms. *Nat Rev Microbiol* 2009, 7:129-143.
9. Kumar VS, Maranas CD: GrowMatch: an automated method for reconciling in silico/in vivo growth predictions. *PLoS Comput Biol* 2009, 5:e1000308.
10. Kumar VS, Dasika MS, Maranas CD: Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics* 2007, 8:212.
11. Aziz RK, Bartels D, Best AA, DeJongh M, Disz T, Edwards RA, Formsma K, Gerdes S, Glass EM, Kubal M, Meyer F, Olsen GJ, Olson R, Osterman AL, Overbeek RA, McNeil LK, Paarmann D, Paczian T, Parrello B, Pusch GD, Reich C, Stevens R, Vassieva O, Vonstein V, Wilke A, Zagnitko O: The RAST Server: rapid annotations using subsystems technology. *BMC Genomics* 2008, 9:75.
12. Henry CS, Zinner J, Cohoon M, Stevens R: iBsu1103: a new genome scale metabolic model of B. subtilis based on SEED annotations. *Genome Biol* 2009:in press.
13. Kummel A, Panke S, Heinemann M: Systematic assignment of thermodynamic constraints in metabolic network models. *BMC Bioinformatics* 2006, 7:512.
14. Jankowski MD, Henry CS, Broadbelt LJ, Hatzimanikatis V: Group contribution method for thermodynamic analysis of complex metabolic networks. *Biophys J* 2008, 95:1487-1499.
15. DeJongh M, Formsma K, Boillot P, Gould J, Rycenga M, Best A: Toward the automated generation of genome-scale metabolic networks in the SEED. *BMC Bioinformatics* 2007, 8:-.
16. Suthers PF, Dasika MS, Kumar VS, Denisov G, Glass JI, Maranas CD: A genome-scale metabolic reconstruction of Mycoplasma genitalium, iPS189. *PLoS Comput Biol* 2009, 5:e1000285.
17. Lee S, Phalakornkule C, Domach MM, Grossmann IE: Recursive MILP model for finding all the alternate optima in LP models for metabolic networks. *Computers & Chemical Engineering* 2000, 24:711-716.

18. Ladányi L, Ralphs TK, Saltzman MJ: Implementing Scalable Parallel Search Algorithms for Data-intensive Applications. *The Proceedings of the International Conference on Computational Science* 2002:592.
19. Overbeek R, Disz T, Stevens R: The SEED: A peer-to-peer environment for genome annotation. *Communications of the Acm* 2004, 47:46-51.
20. Seed Viewer - Home [http://seed-viewer.theseed.org/]
21. Bochner BR: Global phenotypic characterization of bacteria. *Fems Microbiol Rev* 2009, 33:191-205.
22. Mahadevan R, Schilling CH: The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metab Eng* 2003, 5:264-276.
23. Henry CS, Broadbelt LJ, Hatzimanikatis V: Thermodynamics-based metabolic flux analysis. *Biophys J* 2007, 92:1792-1805.
24. Willke T, Vorlop KD: Industrial bioconversion of renewable resources as an alternative to conventional chemistry. *Applied Microbiology and Biotechnology* 2004, 66:131-142.
25. Pharkya P, Burgard AP, Maranas CD: OptStrain: a computational framework for redesign of microbial production systems. *Genome Res* 2004, 14:2367-2376.
26. Burgard AP, Pharkya P, Maranas CD: OptKnock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and Bioengineering* 2003, 84:647-657.
27. Gerdes S, Edwards R, Kubal M, Fonstein M, Stevens R, Osterman A: Essential genes on metabolic maps. *Curr Opin Biotechnol* 2006, 17:448-456.
28. Morimoto T, Kadoya R, Endo K, Tohata M, Sawada K, Liu S, Ozawa T, Kodama T, Kakeshita H, Kageyama Y, Manabe K, Kanaya S, Ara K, Ozaki K, Ogasawara N: Enhanced Recombinant Protein Productivity by Genome Reduction in Bacillus subtilis. *DNA Res* 2008, 15:73-81.
29. Fabret C, Ehrlich SD, Noirot P: A new mutation delivery system for genome-scale approaches in Bacillus subtilis. *Mol Microbiol* 2002, 46:25-36.