MetaPathways v2.0: A master-worker model for environmental Pathway/Genome Database construction on grids and clouds

Niels W. Hanson
Graduate Program in Bioinformatics
University of British Columbia, Canada
Email: nielsh@mail.ubc.ca

Kishori M. Konwar
Shang-Ju Wu
Steven J. Hallam
Department of Microbiology & Immunology
University of British Columbia, Canada
Email: shallam@mail.ubc.ca

***Abstract*—The development of high-throughput sequencing technologies over the past decade has generated a tidal wave of environmental sequence information from a variety of natural and human engineered ecosystems. The resulting flood of information into public databases and archived sequencing projects has exponentially expanded computational resource requirements rendering most local homology-based search methods inefficient. We recently introduced MetaPathways v1.0, a modular annotation and analysis pipeline for constructing environmental Pathway/Genome Databases (ePGDBs) from environmental sequence information capable of using the Sun Grid engine for external resource partitioning. However, a command-line interface and facile task management introduced user activation barriers with concomitant decrease in fault tolerance.**

Here we present MetaPathways v2.0 incorporating a graphical user interface (GUI) and refined task management methods. The MetaPathways GUI provides an intuitive display for setup and process monitoring and supports interactive data visualization and sub-setting via a custom Knowledge Engine data structure. A master-worker model is adopted for task management allowing users to scavenge computational results from a number of worker grids in an ad hoc, asynchronous, distributed network that dramatically increases fault tolerance. This model facilitates the use of EC2 instances extending ePGDB construction to the Amazon Elastic Cloud.

I. Introduction

The development of high-throughput sequencing technologies over the past decade has generated a tidal wave of environmental sequence information from a variety of natural and human engineered ecosystems, creating new computational challenges and opportunities [1], [2]. For example, one of the primary modes of inferring microbial community structure and function from environmental sequence information involves database searches using local alignment algorithms such as BLAST [3]. Unfortunately, BLAST-based queries have become increasingly inefficient on stand-alone machines as reference databases and sequence inputs increase in size and complexity. The advent of adaptive seed approaches such as LAST [4] for homology searches shows promise in overcoming runtime bottlenecks when implemented on grid or cloud computing resources [5]. However, many academic researchers simply do not have access to the technical or infrastructure requirements needed to achieve these calculations, and must turn to online information processing services.

The use of online services, such as Metagenome Rapid Annotation using Subsystem Technology (MG-RAST), increases user access to information storage, gene prediction, and annotation services [6], [7]. Although democratizing infrastructure access, the use of online services can insulate users from parameter optimization, create formatting, data transfer restrictions, and impose barriers to downstream analytic or visualization methods. Large grid computing resources offer an alternative by providing access to high-performance compute infrastructure on local or regional scales. Such computing environments are often composed of multiple grids implementing different batch scheduling and queuing systems e.g., Portable Batch System (PBS), TORQUE, Sun Grid Engine (SGE), or SLURM. Because most federated systems limit the number of jobs a user can submit into a batch-processing queue at one time, scheduling irregularities and job failure are not uncommon when attempting to process large datasets across multiple grid environments implementing different batch scheduling and queuing systems.

Task management can be improved with algorithms that split and load-balance across multiple grid environments, monitor job status to resubmit failed jobs or submit new jobs in succession and consolidate batch results upon completion. These improvements can increase fault tolerance and reduce manual administration requirements. The development of such a method requires a number of considerations: (1) job splitting and merging with appropriate checks on completion and correctness, (2) automated grid installation of search tools such as BLAST/LAST and required databases in a non-redundant manner, (3) bandwidth optimization, (4) minimization of redundant job processing on more than one grid, (5) tolerance to job failure or intermittent availability of computing resources, (6) load-balance to deal with slow, high-volume or small clusters, and (7) an efficient client tool to integrate processes running on the local users machine.

Practically speaking, searching a large set of query sequences against reference databases by splitting them into

smaller query files can be approximately viewed as an instance of the well-studied Do-All problem — *p processors must cooperatively perform n tasks in the presence of an Adversary* [8]. One of the most popular approaches is the master-worker model [9], where the master process sends tasks across the Internet to worker processes, and workers execute and report back the result. Although the literature is replete with such models [8], multiple grid computing imposes new management challenges due to indirect communication issues between master and worker via head nodes. The head nodes themselves have limited communication and compute capacity, and are restricted to performing simple calculations and job submissions. An optimal client tool should automatically minimize redundant or unproductive communications, e.g., frequently trying to submit jobs to or transfer results from a slow server, and adaptively increase their involvement should their performance improve.

Astronomers and mathematicians have encountered similar data processing challenges, and software has been developed for task management across distributed compute networks. For example, SETI@HOME takes advantage of idle CPU hours from volunteers willing to install a custom software to search the cosmos for intelligent life [10], and PrimeNet has a similar implementation for calculating the largest Mersenne prime numbers (http://www.mersenne.org/primenet/). Both projects are tightly controlled to accomplish monolithic tasks under the auspices of major organizations, and the software is not readily transferable to academic users interested in alternative applications such as sequence homology searches. The recent development of MetaPathways [11], an analytical pipeline for sequence annotation and pathway inference using the PathoLogic algorithm and Pathway Tools to construct environmental Pathway/Genome Databases (ePGDBs) [12], [13], [14]. Though MetaPathways uses facile methods to manage homology searches through the integration of an individual compute grid, it does not take advantage of all available compute resources should more grids become available, nor does it overcome challenges associated with ad hoc distributed compute networks.

Here we describe MetaPathways v2.0, representing a series of improvements to the existing pipeline that attempt to address the aforementioned computational and data integration issues. First, automated multiple grid management allows computationally intensive sections of the pipeline to be performed by multiple compute grids simultaneously in an ad hoc distributed system, accommodating dynamic availability, addition, and removal of compute clusters. Because compute cluster access is not generally available to the typical user or research scientist, this implementation includes a module for use of the Amazon Elastic Compute Cloud (EC2) (http://aws.amazon.com/ec2/) through integration with the StarCluster library (http://star.mit.edu/cluster/), enabling those without easy access to compute clusters the ability to create their own. Finally, we improve the usability of MetaPathways through the development of a graphical user interface (GUI) for parameter setup, run monitoring, and result management. Further, the integration and efficient query of results is empowered via a custom Knowledge Engine data structure. Use of this structure is integrated into customized data summary tables, visualization modules, and data export features for down-stream analysis (e.g., the Metagenome Analyzer (MEGAN) [15] and the R programming environment). The GUI interface is written and designed with the Qt 5.0 library in C++ under the LGPL license, and is compatible with Windows, Mac OS X, and Linux-based operating systems.

II. Implementation

A. Multi-grid brokering

MetaPathways v2.0 now has ability to coordinate and manage the computation of sequence homology searches on a multitude of compute grids implementing the Sun Grid Engine/Open Grid Engine [16] or TORQUE (http://www.adaptivecomputing.com/) batch-job queuing systems. Expanding from an individual compute grid to many grids in an ad hoc asynchronous distributed network incurs a number of additional algorithmic challenges in terms of job coordination, worker monitoring, fault tolerance, and efficient job migration. The previous implementation of MetaPathways controlled an individual grid. In the current version, a variant of the master-worker model (perhaps a 'master-cluster' model) has been implemented, with the local machine taking on the role of a master Broker, coordinating a set of worker grids that compute tasks on the Brokers behalf. This setup is analogous to the way operating systems and Internet supercomputers queue requests and return results.

The Broker, operating on the local machine, commissions worker grids to setup as "BlastServices" that compute individual sequence homology jobs much in the same way as the individual grid did in MetaPathways v1.0. However, the availability of many worker grids with varying levels of throughput in an ad hoc distributed network greatly increases the complexity and asynchronicity of job distribution and result harvesting. The Broker not only has to monitor jobs in the context of worker grids and specific samples, but also has to monitor worker throughput and initiate job migration in order to be resilient to overly slow or ineffective workers (Figure 1). Our model assumes an Adversary can sabotage the distributed setup in three ways: (1) a grid core can fail or become ineffective, causing a loss of jobs currently being computed on that core, (2) an entire grid can fail or become ineffective, requiring all jobs to be migrated to other girds, and (3) a sporadic or failed Internet connection increases the asynchronicity of the system, affecting the reliable submission and harvesting of results. The Broker handles these three situations through a combination of job resubmission, job migration to other worker grids, and an escalating back-off in job submission and job harvesting from problematic grids.

Here we discuss the implementation of our distribution algorithm as executed by the Broker. First, smaller independent jobs are created from a large individual sample. The Broker then submits jobs in a modified round-robin fashion up to the queue limit allowed by different clusters. The round robin

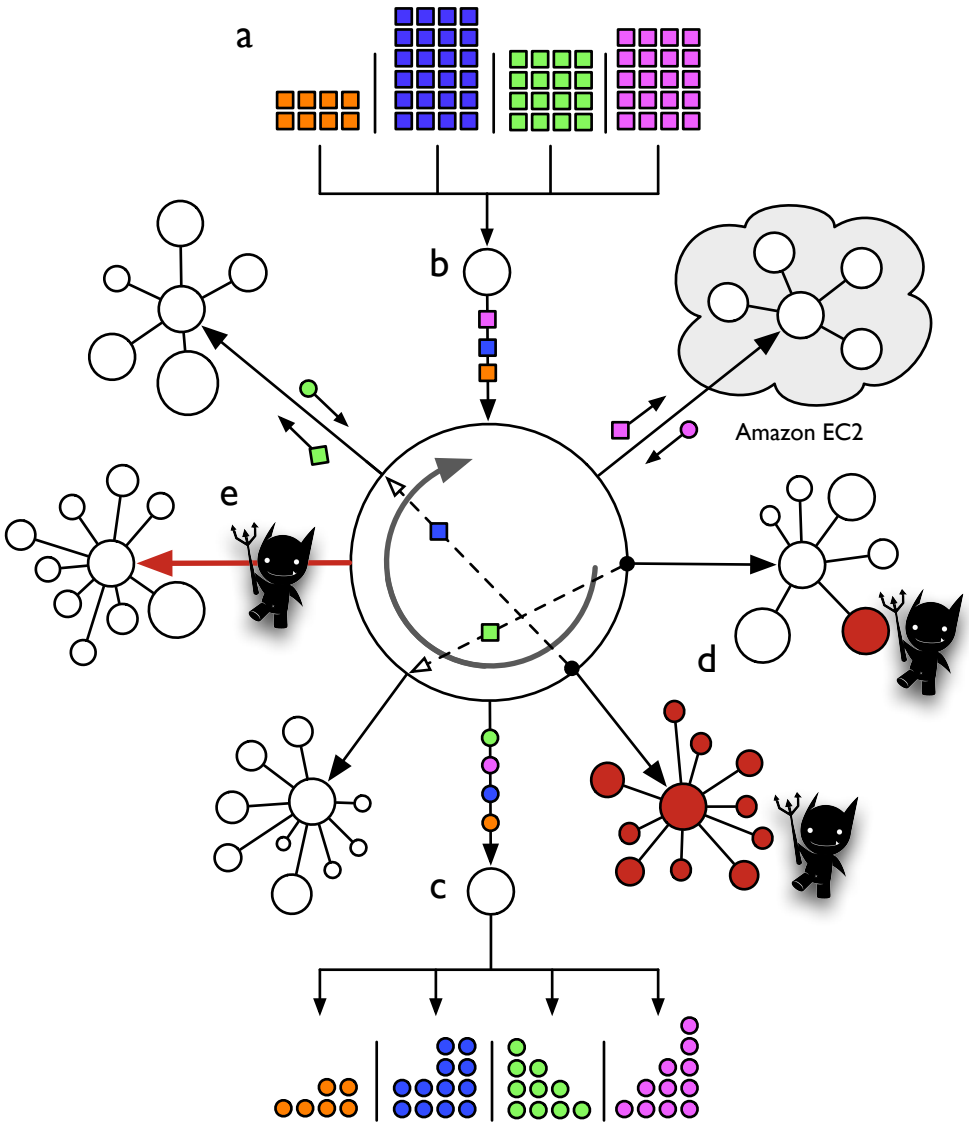

Fig. 1. A master-worker model for sequence homology searches. Sequence homology searches being embarrassingly parallel, the master broker breaks a BLAST job into equal sized sub-tasks (a). The broker then sets up blast services (with all the required executables and databases) on each of the available worker grids, ready to handle incoming tasks. Jobs (squares) are submitted in a round robin manner to each of the grids (b). The broker then intermittently harvests results (circles) from each of the services as they become available, de-multiplexing if there are multiple samples being run (c). An Adversary can cause nodes and whole grids to fail at random, the broker handles this by migrating lost jobs (dashed lines) to alternative grids (d). An Adversary can also cause an intermittent or failing Internet connection (red line), and the broker handles this through an exponential back-off, or eventually migrating jobs to other worker girds if latency becomes excessive (e).

submission is modified by a delay counter (one for each cluster), which is governed by exponential back off for each unsuccessful connection, submission, or harvesting failure. Once all the jobs are submitted but not all results are available, then incomplete jobs are resubmitted to other clusters in counter rank order. Finally, once all the jobs are finished and results are retrieved, then consolidation is performed on the clients machine.

Connectivity or response issues are handled by the exponential back-off of job distribution, while throughput issues are handled by the migration of outstanding or pending jobs to more available workers. On each unsuccessful attempt to submit or harvest results, the broker waits some specified period of time before retrying. This time exponentially increases on each successive failure until an upper maximum is reached (by default set to 60 minutes). This behavior is a well-accepted method in Queueing theory, which limits the number of connections attempted to slower grids. To handle job migration, we have implemented a heuristic to migrate jobs; a job is migrated when its pending time at a particular grid is six standard deviations greater than the average job completion time over the previous hour. Such tasks are then

migrated to a grid with the lowest expected completion time, proportional to the number of outstanding jobs. This ensures that jobs are not readily passed to grids with a higher load.

B. Amazon Elastic Cloud Integration

MetaPathways v2.0 enables researchers to take advantage of existing compute grids using the TORQUE or SunGrid queue submission engine, requiring only login credentials and basic information about the queue submission system. However, there are many instances where convenient access to a compatible compute cluster is simply unavailable or difficult. Here, we have integrated StarCluster (http://star.mit.edu/cluster/), an open source cluster-computing toolkit for Amazons Elastic Compute Cloud (EC2), in order to automate and simplify the setup of a compatible compute grid. EC2 grids are setup with a custom MetaPathways Amazon Machine Image (AMI) containing MetaPathways code, system appropriate executables, and compiled databases, reducing setup bandwidth and latency. EC2 grids are specified similar to other compute grids; however, the user must provide Amazon Web Services (AWS) credentials: an access key, a "secret" access key, and an user ID, obtained by registering for an Amazon EC2 account.

C. Graphical User Interface & Data Integration

The newest iteration of MetaPathways expands upon its initial release with the addition of a minimalistic, simple-to-use GUI aimed to allow users to process, monitor, integrate, and analyze large environmental datasets more effectively. A MetaPathways run is set up via a configuration window that allows specification of input and output files, quality control parameters, target databases, executable stages, and grid computation (Figure 2). Available grid systems and credentials are added and stored in an "Available Grids" window, allowing the user to add additional grids when credentials become available. Currently, grid credentials using the Sun Grid or TORQUE job distribution system are supported, though expansion to include other systems like SLURM and PBS is anticipated. Once a run has started, an execution summary is displayed showing run progress through completed and projected pipeline stages along with a number of logs showing the exact commands of each processing step (Figure 3).

Modern next-generation environmental sequence datasets often require the annotation of millions of reads, contigs, or open reading frames (ORFs). Not only is this annotation computationally intensive, the interpretation, integration, and analysis of gigabytes of output files is as well. Here, MetaPathways v2.0 has implemented a custom Knowledge Engine data structure and file indexing scheme that connects data primitives, such as reads, ORFs, and metadata, and projects them onto a specified classification or hierarchy (e.g., KEGG [17], COG [18], MetaCyc [12], and the NCBI Taxonomy database [19]) (Figure 4). The benefit of this custom abstraction is flexibility and performance; data primitives are extensible to new environmental data types such as transcriptome and proteomics reads, and related knowledge transformations like coverage statistics or numerical normalizations can be customized for them if necessary. Performance wise, data can now be quickly queried for specific results faster than typical database setups like MySQL or Oracle which are forced to have a general structure. Through integration with the Knowledge Engine data structure, data subsets can be easily selected to drive inquiry through custom look-up tables and

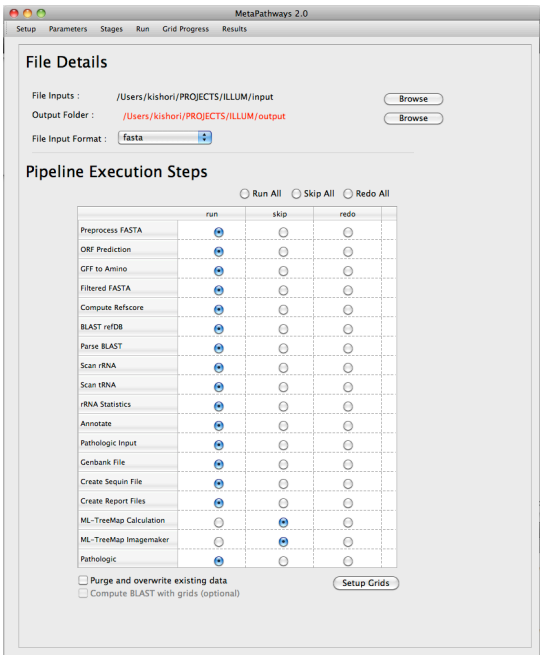

Fig. 2. Configuring a MetaPathways run. Specific pipeline execution stages can run, skipped, or redone by setting the appropriate radio button. A check box at the bottom indicates if the currently configured set of external worker grids should be used for this run. Additional grids can be configured with their credentials by clicking the "Setup Grids" button.

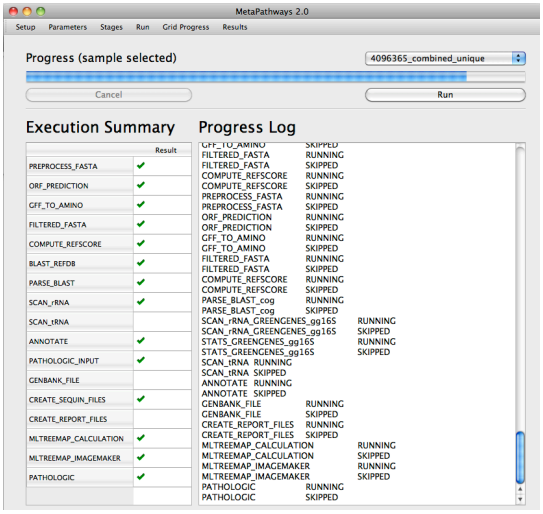

Fig. 3. Once a MetaPathways run is configured and started, progress is monitored using two coordinated windows. Execution progress and processed results for a specific sample are displayed as a series of tabs containing tables, graphs or other visualizations, and can easily be expanded should additional pipeline stages or changes occur. Different samples can be selected using the drop-down combo-box above.

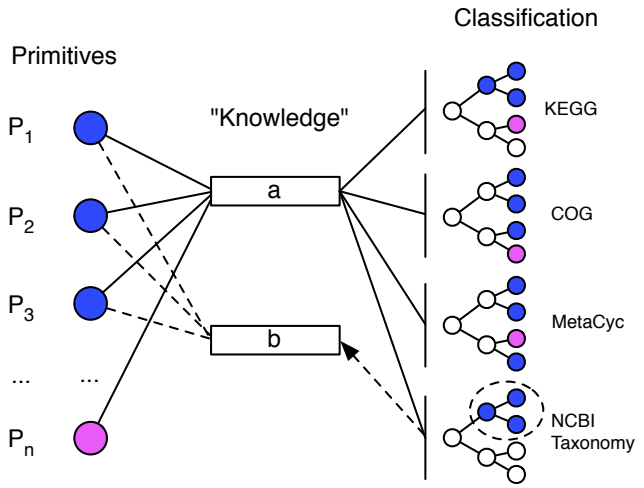

Fig. 4. The MetaPathways 'Knowledge Engine' data structure. One of the recent developments to enable the efficient exploration of large environmental sequence datasets is the Knowledge Engine data structure. This data structure views the original reads or ORFs of a sample as data primitives, which can be summarized by projection onto a series of classification schemes. Pointer following is a computationally efficient operation so the identification and enumeration of data primitives is robust to large samples consisting of millions of annotations. We have nicknamed the connection between primitives and classification schemes as Knowlege objects (a). Through the exploration of data projected on classification schemes new Knowledge objects can be created (dashed lines) through selection of an interesting data subset (b). Once defined and created, Knowledge objects can be projected onto data summarizing modules like tables or visualizations, and these, in turn, can be used to create new Knowledge objects, enabling the efficient and interactive query of millions of annotations.

visualization modules, or exported in a variety of formats for downstream analysis (e.g., custom .csv, .tsv tables, nucleotide and protein fastas, MEGAN compatible csv, etc.).

The Knowledge Engine can be exploited to enhance any variety of data visualization or summary modules. Here MetaPathways v2.0 has implemented three particular modules: a Large Table implementation exploiting the order statistic algorithm to allow the manual browsing and search of tables with million of rows (Figure 5a), a Tree-taxonomy Browser for the display of taxonomic annotation (Figure 5b), and a Contig Viewer for the inquiry of annotated ORFs on individual reads or assembled contigs (Figure 5c). Each module enables the selection of subsets and projections of reads, ORFs, or Contigs onto the above classification schemes for intuitive data exploration and analysis. Additionally, export functions for sequences, reads, or ORF annotations facilitate downstream analyses like tree-building, R, or analytical software like MEGAN.

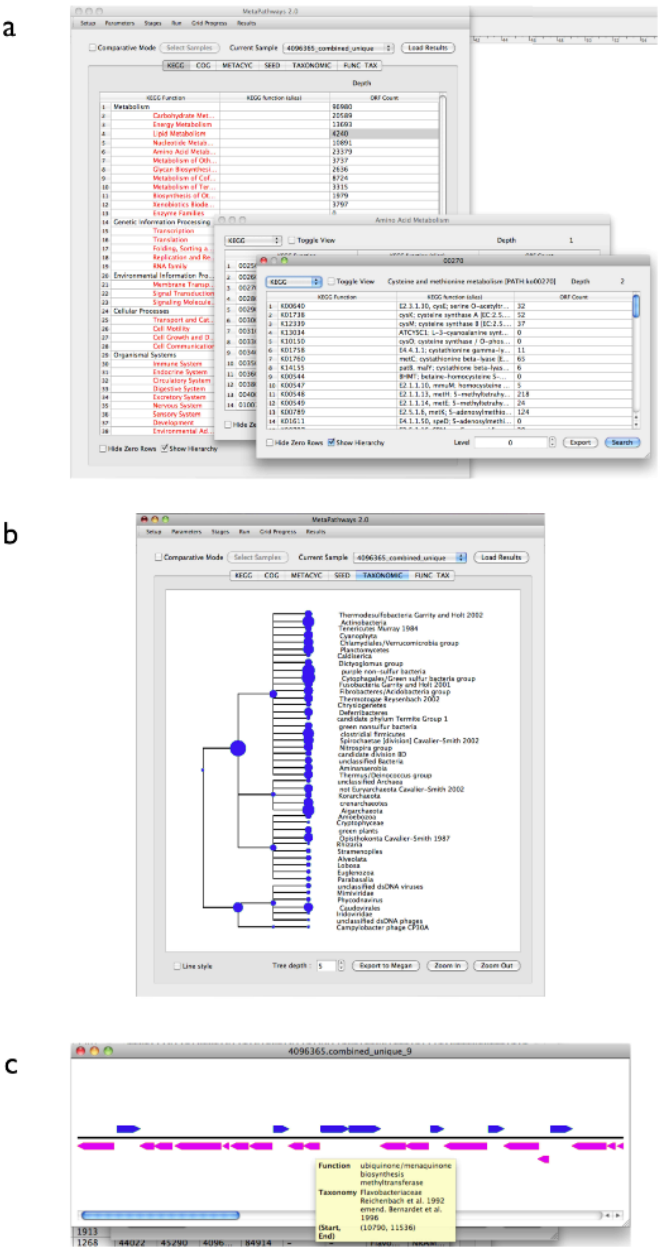

Fig. 5. Knowledge Engine summary using Large Tables, Tree-Taxonomy Browser, and Contig Viewer. MetaPathways v2.0 has integrated its Knowledge Engine data structure into three data summary and visualization modules: a Large Table module that allows for the efficient query, look-up, and sub-setting of reads, ORFs, statistics, annotations and their hierarchical classification (e.g., KEGG, COG, MetaCyc, SEED, NCBI Taxonomy) (a); A Tree-taxonomy Browser of annotated ORF on the NCBI Taxonomy classification using the LCA and Megan declinations (b); a Contig Viewer enabling the browsing of contigs or long reads for ORF positions, and functional and taxonomic annotations in tool-tips (c).

III. Results

A. Heterogeneous Compute Grid Performance

To illustrate the ability of the task distribution algorithm to migrate tasks between a number of different homogeneous compute clusters, we ran a demonstration run on three different compute grids simultaneously (Table I). Two systems (Bugaboo and Jasper) from WestGrid, one Compute/Calcul Canadas High-performance computing consortia, and HallamGrid, an internal grid of 10 nodes built of heterogeneous Apple Mac Pro desktop computers. Connection to WestGrid systems was through the Internet, while HallamGrid was connected through local area network. The Bugaboo and Jasper clusters have 4584 and 4160 cores, respectively, running the Torque or

SunGrid Engine queuing systems with at least 2GB of RAM available per core. HallamGrid ran the SunGrid Engine and consisted of 15 nodes with 4-8 cores each with a variety of processor types. Locally, MetaPathways v2.0 was running on a Mac Pro desktop computer with Mac OSX 10.6.8 with 2 2.4 Ghz Quad-Core Intel Xeon processors and 16 GB of 1066 Mhz DDR3 RAM.

We ran MetaPathways v2.0, configured to the above grids, on a Illumina metagenome, predicting 106,500 ORFs from 127,821 assembled contigs using standard MetaPathways parameters [11]. These ORFs were annotated against COG database using the BLAST algorithm. Split into batches of 500 sequences each, this resulted in 213 sequence homology jobs. MetaPathways distributed these jobs to the network, completing in 0.5 real-world hours. A transfer matrix describes the job distribution behavior of the Broker (Table II). Diagonal values represent jobs completed at a particular grid, while off-diagonal values represent the transfer of a job from one grid to another by the Broker. Such matrices could be used as a tool to assess grid behavior; a high number of off-diagonal values would indicate a high number of transfers and therefore low performance.

TABLE I
OVERVIEW OF GRID HARDWARE SPECIFICATIONS.

System	Cores	Nodes	Memory	Connectivity
Bugaboo	4584	446	16-24GB/Node	Internet
Jasper	4160	400	16-24GB/Node	Internet
HallamGrid	80	10	32-64GB/Node	LAN

TABLE II
MATRIX OF COMPLETED JOBS (DIAGONAL VALUES) AND JOB TRANSFERS (OFF-DIAGONAL).

	Jasper	Bugaboo	HallamGrid
Jasper	49	0	0
Bugaboo	2	10	8
HallamGrid	0	0	164

IV. DISCUSSION

MetaPathways enables streamlined functional, taxonomic, and pathway-centric analysis of environmental sequence information when compared to existing methods based on KEGG pathways and SEED subsystems. This version of the pipeline focused on improving pipeline scalability through the use of a robust master-worker model to control compute grid submission. This system potentially allows collaborating research labs to take advantage of under-used idle cycles on their heterogeneous in-house networks, and to facilitate the use of external grids when additional dedicated CPU cycles are necessary. We also enable the setup and use of Amazon EC2 instances for those who do not have convenient access to large federated compute grids. Further, we improved the usability and monitoring capacity of MetaPathways through the integration of a GUI to monitor analytical progress, while also providing an efficient framework to bring users closer to large environmental samples for hypothesis generation. The custom Knowledge Engine data structure drives the efficient query and sub-setting of the processed data, allowing the efficient analysis of samples containing millions of annotations. We can think of no other stand-alone software tool that will allow for comparable analysis on such large datasets.

We have demonstrated that the Brokers distribution algorithm has good empirical performance on a small number of grids. However, the current implementation uses a very general Queueing theory heuristic. The interactions between the client and clusters with varying loads, user behaviors, and administrative activities could be viewed as a non-cooperative game where the various players are competing for resources. From a Game theory perspective there could be interesting equilibrium conditions that result within this dynamic network. Not only would these results be theoretically interesting, they could have critical implications to the performance of the algorithm when large grid networks are being managed.

Aside from run-time improvements, additional data transformation and visual analysis modules, expansion on existing taxonomic binning and marker gene identification methods are needed. These include coverage statistics for assembled sequence information, data matrices and interactive visualizations indicating numerical abundance and taxonomic distribution of enzymatic steps. Additionally, the applications of self-organizing maps and automated methods to append single cell or population genome assemblies to the NCBI hierarchy for more accurate taxonomic binning are needed. Further, as transcriptomic and proteomic datasets are becoming increasingly available, they represent new data primitives that need to be extended into the Knowledge Engine data structure and analysis modules. More generally, reference databases for 5S, 7S and 23S RNA genes and updates to the current MetaCyc database that include more biogeochemically relevant pathways are needed to improve BLAST and cluster-based annotation efforts. Finally, more experience and operational insight is needed in constructing, comparing and interpreting ePGDBs to identify potential sources of error and inform ongoing Pathway Tools development efforts.

V. CONCLUSION

MetaPathways v2.0 improves upon the previous modular annotation and analysis pipeline by providing users with improved performance and usability in high-performance computing through a master-slave grid submission algorithm. The addition of a GUI significantly improves software use and adoption, providing users better control over their pipeline runs. It is extensible to the data volume of novel genomic and transcriptomic datasets, and generates useful data products for microbial community structure and functional analysis including phylogenetic trees, taxonomic bins and tabular annotation files. The MetaPathways v2.0 software, installation instructions, tutorials and example data can be obtained from http://github.com/hallamlab/MetaPathways/ or http://hallam.microbiology.ubc.ca/MetaPathways.

COMPETING INTERESTS

The authors are not aware of any competing interests.

AUTHORS CONTRIBUTIONS

KMK, NWH, and SW conceived, developed, and implemented the master-worker model and the GUI and visualization modules. NWH and SW drafted and edited the manuscript. SJH assisted in writing the manuscript and oversaw the project.

ACKNOWLEDGMENTS

This work was carried out under the auspices of the Genome Canada, Genome British Columbia, the Natural Science and Engineering Research Council (NSERC) of Canada, the Canadian Foundation for Innovation (CFI) and the Canadian Institute for Advanced Research (CIFAR). The Western Canadian Research Grid (WestGrid) provided Access to high-performance computing resources. KMK was supported by the Tula Foundation funded Centre for Microbial Diversity and Evolution (CMDE) at UBC. We would like to thank, Peter Karp, Tomer Altman, and the SRI International staff for invaluable comments related to design ethos and implementation of MetaPathways.

REFERENCES

[1] M. B. Scholz, C.-C. Lo, and P. S. G. Chain, "Next generation sequencing and bioinformatic bottlenecks: the current state of metagenomic data analysis." *Current Opinion in Biotechnology*, vol. 23, no. 1, pp. 9–15, Feb. 2012.

[2] N. Desai, D. Antonopoulos, J. A. Gilbert, E. M. Glass, and F. Meyer, "From genomics to metagenomics." *Current Opinion in Biotechnology*, vol. 23, no. 1, pp. 72–76, Feb. 2012.

[3] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, Sep. 1997.

[4] S. M. Kiełbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith, "Adaptive seeds tame genomic sequence comparison." *Genome Res*, vol. 21, no. 3, pp. 487–493, Mar. 2011.

[5] G. Bell and T. Hey, "Beyond the data deluge," *Science*, vol. 323, no. 5919, pp. 1296–1297, Mar. 2009.

[6] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. A. Edwards, "The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes." *BMC Bioinformatics*, vol. 9, p. 386, 2008.

[7] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko, "The RAST Server: rapid annotations using subsystems technology." *BMC Genomics*, vol. 9, p. 75, 2008.

[8] C. Georgiou and A. A. Shvartsman, "Cooperative Task-Oriented Computing: Algorithms and Complexity," *Synthesis Lectures on Distributed Computing Theory*, vol. 2, no. 2, pp. 1–167, Jul. 2011.

[9] E. Christoforou, A. F. Anta, C. Georgiou, M. A. Mosteiro, and A. Sánchez, "Applying the dynamics of evolution to achieve reliability in master–worker computing," *Concurrency and Computation: Practice and Experience*, vol. 25, pp. 2363–2380, 2013.

[10] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@ home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[11] K. M. Konwar, N. W. Hanson, A. P. Pagé, and S. J. Hallam, "MetaPathways: a modular pipeline for constructing pathway/genome databases from environmental sequence information." *BMC Bioinformatics*, vol. 14, no. 1, p. 202, 2013.

[12] R. Caspi, H. Foerster, C. A. Fulcher, R. Hopkinson, J. Ingraham, P. Kaipa, M. Krummenacker, S. Paley, J. Pick, S. Y. Rhee, C. Tissier, P. Zhang, and P. D. Karp, "MetaCyc: a multiorganism database of metabolic pathways and enzymes." *Nucleic Acids Research*, vol. 34, no. Database issue, pp. D511–6, Jan. 2006.

[13] P. D. Karp, S. M. Paley, M. Krummenacker, M. Latendresse, J. M. Dale, T. J. Lee, P. Kaipa, F. Gilham, A. Spaulding, L. Popescu, T. Altman, I. Paulsen, I. M. Keseler, and R. Caspi, "Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology." *Brief. Bioinformatics*, vol. 11, no. 1, pp. 40–79, Jan. 2010.

[14] P. D. Karp, M. Latendresse, and R. Caspi, "The pathway tools pathway prediction algorithm." *Stand Genomic Sci*, vol. 5, no. 3, pp. 424–429, Dec. 2011.

[15] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster, "MEGAN analysis of metagenomic data." *Genome Res*, vol. 17, no. 3, pp. 377–386, Mar. 2007.

[16] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," in *CCGRID-01*. IEEE Comput. Soc, 2001, pp. 35–36.

[17] M. Kanehisa and S. Goto, "KEGG: kyoto encyclopedia of genes and genomes." *Nucleic Acids Research*, vol. 28, no. 1, pp. 27–30, Jan. 2000.

[18] R. L. Tatusov, D. A. Natale, I. V. Garkavtsev, T. A. Tatusova, U. T. Shankavaram, B. S. Rao, B. Kiryutin, M. Y. Galperin, N. D. Fedorova, and E. V. Koonin, "The COG database: new developments in phylogenetic classification of proteins from complete genomes." *Nucleic Acids Research*, vol. 29, no. 1, pp. 22–28, Jan. 2001.

[19] S. Federhen, "The NCBI Taxonomy database." *Nucleic Acids Research*, vol. 40, no. Database issue, pp. D136–43, Jan. 2012.