Data-Aware Resource Scheduling for Multicloud Workflows: A Fine-Grained Simulation Approach

Wei Tang,* Jonathan Jenkins,* Folker Meyer,* Robert Ross,* Rajkumar Kettimuthu,*
Linda Winkler,* Xi Yang,† Thomas Lehman,† Narayan Desai,‡*

*Argonne National Laboratory, Argonne, IL, USA
†University of Maryland, College Park, MD, USA
‡Ericsson, San Jose, CA, USA
*{wtang, jekins, folker, rross, kettimut, winkler}@mcs.anl.gov
†{maxyang, tlehman}@umd.edu, ‡narayan.desai@ericsson.com

***Abstract*—Cloud infrastructures have seen increasing popularity for addressing the growing computational needs of today's scientific and engineering applications. However, resource management challenges exist in the elastic cloud environment, such as resource provisioning and task allocation, especially when data movement between multiple domains plays an important role. In this work, we study the impact of data-aware resource management and scheduling on scientific workflows in multicloud environments. We develop a workflow simulator based on a network simulation framework for fine-grained simulation for workflow computation and data movement. Using the workload traces from a production metagenomic data analysis service, we evaluate different resource scheduling mechanisms, including proposed data-aware scheduling policies under various resource and bandwidth configurations. The results of this work are expected to answer questions about how to provision computing resources for certain workloads efficiently and how to place tasks across multidomain clouds in order to reduce data movement costs for overall improved system performance.**

***Keywords*-data-aware scheduling, resource management, scientific workflow, cloud computing**

I. Introduction

Because of the increasing computational and data requirements, applications in science and engineering are demanding scalable computing resources. This situation has boosted the adoption of elastic cloud infrastructures. The flexible and diverse nature of the clouds, however, brings challenges for effective resource management, such as resource provisioning and allocation, especially when data movement between multiple domains plays an important role.

Two major questions must be answered in order to enable effective resource management in cloud environments. One concerns resource provisioning: How many computing resources need to be allocated to serve certain workloads (heavy or light, data-intensive or compute-intensive)? For a certain workload, using insufficient resources results in poor productivity; using excessive resources leads to unnecessary monetary costs. The second question concerns resource allocation: What resources are best used to complete these tasks? Using insufficient resources results in poor application performance or even failures, while overprovisioning resources for a task results in suboptimal resource utilization. Also, being oblivious to I/O data may result in substantial unnecessary data movement cost. Neither of these two questions has a straightforward answer because of the diversity of both application characteristics and system status.

In this work, we address the challenges in resource provisioning and resource allocation for multicloud scientific workflows that require task placement and data movement between distributed multidomain computing sites. To evaluate different resource plans and scheduling mechanism under various workload and system configuration, we develop a data-aware workflow simulator based on CODES simulation framework [2][7] to provide fine-grained simulation for workflow computation and data movement.

Further, we propose a couple of distributed data-aware workflow scheduling mechansims and evaluate them using cloud workload traces from a production metagenomics data analysis service: the MG-RAST metagenomic data analysis service[3][15][22]. We study the system performance variations measured by a number of metrics under diverse workload characteristics and resource configurations. The experimental results show how the performance metrics are influenced by resource management mechanisms. They also indicate that data-aware scheduling can provide considerable performance improvement for multidomain workflow applications. The observations of this work can guide efficient resource provisioning and allocation based on variable workload characteristics and system status.

II. Related Work

Cloud resources are being adopted to serve today's applications as the compute and data demands grow. For example, Lezzi et al. [13] proposed a programming framework that allows coordinated execution of applications in the EGI Federated Cloud. Mohamed et al. [16] implemented a genome analysis pipeline on the Microsoft Azure cloud. Different

from these studies that focus on porting applications into the cloud, we focus here on cloud resource management.

Various researchers have addressed the resource management challenges with multicloud infrastructures. Keahey et al. [11] developed a multicloud autoscaling service for automatic resource provisioning based on user-customizable policies. Duplyakin et al. [9] proposed mechanisms for rebalancing multicloud resource deployment for improved workload execution performance. Zhou and He [24] proposed a transformation-based cost optimization for workflows in the cloud. Our work differs from such studies by highlighting the impact of data movement between multicloud sites, an issue that has becomeincreasingly significant in cloud computing.

Some existing work has optimized data movement in domains such as grid or data centers. For example, Kosar and Balman [12] proposed data-aware batch scheduling in grid computing to coordinate data transfer between distributed sites. Maheshwari et al. [14] used performance models to predict task execution and data transfer times between different sites to enhance workflow scheduling. Zou et al. [25][26] used adaptive data compression to accelerate data movement between simulation and data analysis machines in data centers. Our current work addresses the data movement problem in multicloud environments.

Event-driven simulation is a commonly used approach to aid in the design and evaluation of resource management mechanisms. For batch job scheduling, some production batch schedulers have a simulation mode, such as SLURM [23] and Cobalt [19]. For cloud computing, Calheiros et al. [5] developed CloudSim, a simulation toolkit that supports both system and behavior modeling of cloud system components such as data centers, virtual machines, and resource provisioning policies. Chen and Deelman [8] proposed WorkflowSim, an extended version of CloudSim that supports workflow management and scheduling.

We present here a distributed workflow simulator that differs from existing batch or cloud simulators because it can provide fine-grained data movement simulation; that is, bandwidth contentions between concurrent transfers over the network can be emulated based on an underlying network simulation framework.

III. System overview

The system model of this work is based on the AWE [1][20] workload management system, which executes distributed workflow in the server/clients model. The server receives job submissions and maintains a task queue. The clients on distributed computing resources gets tasks from the server and processes them locally. AWE uses the Shock data management system [4] for data storage and subsetting. More details about the computing platform can be found in our previous work [20]. Figure 1 illustrates an example of the Shock and AWE setup with two cloud sites connected.

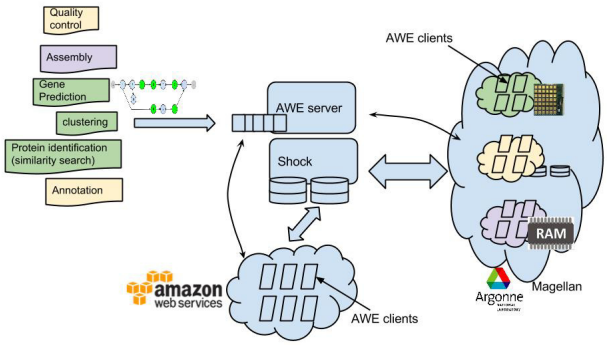


Figure 1. Shock/AWE diagram in a multicloud environment

In the AWE system, a job represents a workflow run for a specific set of input data. A task represents the computation for a particular stage in the workflow. A task can be split into one or multiple subtasks (we call each a "workunit"), each processing a subset of the input data of the same task. The workunit is the schedulable unit maintained in the queue. How to prioritize and allocate the queued workunits to different compute clients are the responsibility of the scheduler.

IV. Scheduling Methods

In this section, we first introduce the basic scheduling method in AWE, followed by enhanced data-aware methods.

A. Basic Scheduling Method

By default AWE uses a first-come, first-served (FCFS) policy to main the workunits. That is, the workunits are sorted by the time stamp of job submission, and hence all workunits belonging to the same job will have the same priority. AWE handles clients' checkout requests in a round-robin fashion in order to maintain load balancing. Specifically, the workunit checkout requests are handled immediately if eligible workunits are in the queue; otherwise, the requests are queued until eligible workunits are available. In this work, we assume the computing resources are homogeneous except for the network bandwidths to the data server.

B. Data-Aware Scheduling

When workflows are run in multicloud environments, data movement between different sites incur overhead. Since at different task stages the application have different requirements for data movement, the scheduler should be aware of these data requirements before allocating tasks among multiple sites. Thus we propose data-aware scheduling as an enhancement to the AWE scheduler. The basic idea is to make the scheduler aware of data characteristics and the distances of the computing resources and to allocate tasks effectively in order to reduce data movement overhead.

We quantify the data characteristic by the computational expensiveness that is correlated to compute to I/O ratio. Specifically, we consider task A is more "computationally expensive" than task B if for the same amount of input size, the compute time of task A is longer than that of task B, given the same compute and network configuration. We define the computational expensiveness as $E_c = T_{run}/S_{input}$, where T_{run} is the compute time and S_{input} is the size of the input data; the unit is sec/MB.

Based on historical job traces, we can determine the average E_c for different task type, and the scheduler can use that metric to determine whether a task is more computationally expensive (with higher E_c) or, in other words, which task is more data intensive (with lower E_c). Our data-aware scheduling methods are based on the task's E_c value and the distance between the computing resource to the data server. We present in this work two simple policies, namely "best-fit" and "greedy."

1) Best-fit: This is simplest and most straightforward way that allocates only the most computationally intensive task (with highest E_c) to the remote site, where the bandwidth to the data server is relatively low. Our previous work [18] revealed that the most compute-intensive task in MG-RAST is protein similarity search using BLAT [10]: every workunit with 50 MB of input data needs around 3 to 5 hours to process on an 8-core virtual machine, which is much more expensive than other task types. For the evaluation in this work, we implement best-fit to allocate only BLAT task to the remote site.

2) Greedy: Using the best-fit policy can guarantee that the remote site involves the least data movement overhead. The tradeoff is that the computing resources at the remote site can run only one type of task and may be idle sometimes even if another type of task is in the queue—a situation that clearly is harmful to the whole system utilization. As a compromise, we propose a "greedy" policy. Specifically, we first sort the task types in order of compute expensiveness E_c. Each time when a remote checkout happens, the client will first check whether there is a task belonging to the type with highest E_c. If not, the clients will check the task type with the second highest E_c, and so on. Eventually, a client will get a task instead of being idle.

V. AweSim: A Distributed Workflow Simulator

Simulation is an integral part of our workflow scheduler. To model the performance of distributed workflow in multicloud environments, we developed a simulator named AweSim[1] based on the CODES simulation framework [2][7]. The simulator can take workflow jobs as input and emulate workflow task parsing, task scheduling, and data movement.

The goal of the CODES project is to use parallel, fine-grained simulation to explore the design of large-scale storage or network architectures and distributed data-intensive science facilities. CODES is built on the Rensselaer Optimistic Simulation System (ROSS) [6], a discrete event simulation framework that allows simulations to be run in parallel. CODES can simulate disk I/O and network model in fine granularity. For example, the network congestions between multiple transfers are emulated by breaking the data load into small packets and queued up at the ends of the links that are configured with maximum bandwidths.

To simulate an application, one needs to implement the application model on top of ROSS and CODES. ROSS manages the simulation clock and events, and CODES implements I/O and network transfer models. The application model implements the use case components (implemented as "logical processes" or LPs) and their interactions. In this work, the application model is the workflow management in the multidomain clouds environment. Specifically, the LPs includes a job server (AWE server), a data server (Shock), and multiple computing clients (AWE clients) grouped by multiple sites. The bandwidth between each site and the data server can be configured as different values close to the real configurations. Figure 2 shows the LPs and network configurations of AweSim used in the experiments.

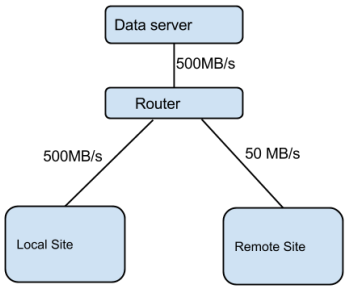


Figure 2. Simulation configurations

VI. Experiments

In this section, we present our simulation-based study of data-aware resource scheduling for multicloud workflows. We first introduce the simulation setup and some performance metrics, followed by the results and observations.

A. Simulation Setup

The job trace is collected from the production MG-RAST service [3][15] running in the Magellan cloud [17], which contains all the jobs completed in April 2014. Specifically, the trace includes 3,193 complete jobs, 31,930 tasks, and 64,026 workunits. More detailed information about MG-RAST tasks' compute and I/O characteristics can be found in our previous work [18].

The output of the simulation is another event series including the time stamps of all major activities, such as compute start and completion and data transfer start and

[1]https://github.com/wtangiit/awesim

completion. Based on the output trace, we can measure the system performance with a number of metrics, described in following subsection.

B. Evaluation Metrics

We evaluated system performance in terms of the following metrics representing both user and system interests.

- *Job response time*, or job turnaround time, is measured by the time period between job submission and completion, including the queuing time and processing time.
- *System utilization rate* in a certain period of time is measured by the proportion of utilized computing cycles to the total number of cycles.
- *Data analysis throughput* is measured by the processed data size in a unit time, which reflects the system productivity of processing data.
- *Data movement overhead* for a job in this work is measured by the proportion of time spent on data movement between the computing resource and data server compared with the total job processing time.

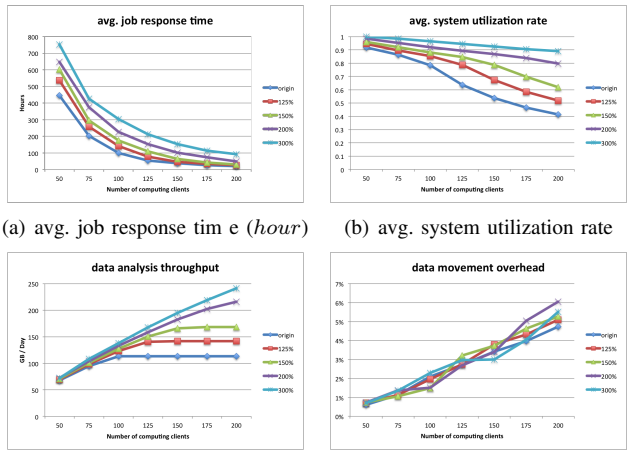
C. Simulation for One Site

We first conduct a set of simulations for one site only, in order to validate the behavior of the simulator and explore the impact of resource provisioning plans under various system loads. In the experiments, we configured one computing site with various numbers of clients (ranging from 50 to 200 by steps of 25) to represent different capacity plans (in practice, we use a varying number of clients, ranging from 50 to 150, based on availability). Besides the original workload, we varied the workload and identified four additional different workloads with 25% more, 50% more, doubled, and tripled density compared with the original one [21]. We configured the queuing policy as FCFS. Figure 3 shows the experimental results.

Average job response time (Figure 3(a)) and average system utilization (Figure 3(b)) show the trends: the former decreases as the number of computing clients increases, and increases as the workload system load gets heavier; the latter acts in the opposite manner. The difference is that the job response time is more sensitive to system load when the number of clients is low, whereas the utilization is more sensitive to system load when the number of clients is high.

Data analysis throughput (Figure 3(c)), measured in gigabytes input data per day, increases as the number of clients increases at the beginning, but it goes flat after some point even when more clients are added. This point is different for different system loads. Generally, the heavier the system load is, the more the throughput can be improved by adding more clients. In other words, the system throughput can be increased by adding more computing resources but also will be limited by the workloads.

Data movement overhead (Figure 3(d)) increases only as the number of clients increases and has no significant



(a) avg. job response tim e ($hour$) (b) avg. system utilization rate

(c) throughput (GB/Day) (d) data movement overhead

Figure 3. Performance evaluation for different resource provisioning plans under various workloads. The legends represent different workload densities compared with the original one. x-axis: number of computing clients.

correlation with the system load. The reason is that the data transfer is highly influenced by the contention of the network between computing clients and the data server: more clients mean more concurrent data transfers to the data server.

D. Simulation for Two Sites

We next explore the performance of running workflow over multiple sites where data movement is required in between. Specifically, we simulate a real case for MG-RAST where a dedicated pool of computing clients is available at a "local site" and additional clients can join from a "remote site." The local site is closer to the data server as its bandwidth to the data server is high. In contrast, the remote site has lower bandwidth to the data server. Based on real system observation for MG-RAST setting, we configured the download and upload bandwidths of the local site as 500 MB/s and 100 MB/s, respectively; those of the remote site are 10 times slower, respectively.

In the simulation, we studied four cases (with number of clients at local/remote site): all local clients as baseline (150/0), more clients at the local site (100/50), equal number of clients at both sites (75/75), and more clients at the remote site (50/100). We fix the total number of clients to 150, which is in the range of that we use for MG-RAST.

Figure 4 shows the experimental results. For the first three metrics ($resp$, $util$, $thpt$), a higher proportion of remote clients results in a moderate or slight degradation of performance; and, in particular, the system utilization and throughput are less sensitive to the ratio of the remote clients. On the other hand, the data movement overhead is significantly sensitive to the remote client ratio. As shown in Figure 4(d), the overhead is increased exponentially—doubled every time the remote client ratio increases.

The reason for the performance variations is that more

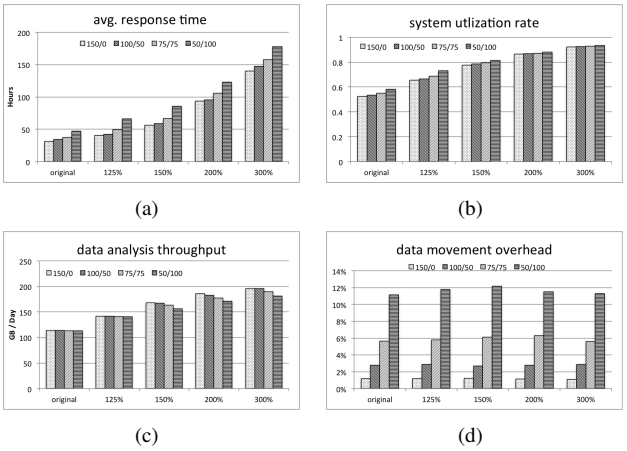


Figure 4. Performance evaluation for running job over two sites (one local and one remote). Number of clients in local/remote sites: 150/0 (all local, baseline), 100/50 (more local), 75/75 (equal), 50/100 (more remote). x-axis: system loads.

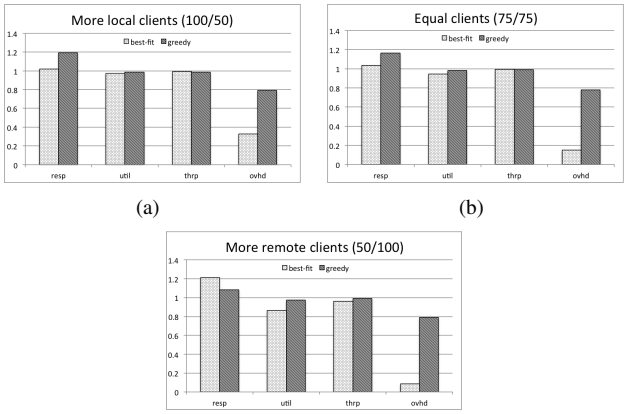


Figure 5. Performance variation brought by data-aware scheduling. x-axis: system load; y-axis: the ratio (relative values) of the performance values with data-aware scheduling (best-fit and greedy) to those without data-awareness. X-axis: examined metrics. resp and ovhd are less the better (< 1 means improvement); util and thpt are the more the better (< 1 means degradation).

data movement has been incurred over the lower-bandwidth network between multiple sites. The first three metrics are impacted only moderately because they are affected only indirectly by the 'data movement since the computing capacities remain unchanged (the same total number of computing clients is configured). On the other hand, the data overhead is directly impacted by the multisite deployment with a lower bandwidth in between. Thus, we need data-aware task allocation to reduce the data movement overhead.

E. Simulation for Data-Aware Scheduling

In this section, we evaluate data-aware task allocation. We examine the "best-fit" and "greedy" policies for the two-site cases. We compare the relative performance variations as shown in Figure 5. Specifically, we normalized the base performance values (base scheduler) to 1, and calculated the relative values for the performance values of test cases with data-aware scheduling (on Y-axis).

As shown in the figure, generally, both data-aware scheduling policies can brought moderate to slight changes to the first three metrics (response time, utilization, and throughput) and more noticeable variations to the data overhead. In other words, data-aware scheduling can significantly improve data movement overhead with slight impact to other system performance metrics.

Further, best-fit performs better than greedy when the remote clients are not dominant (the first two cases); it has less impact to response time and much more improvement on data movement overhead. On the other hand, the greedy policy does better when remote clients are dominated (the third case) in terms of mitigating the negative impact brought by best-fit on response time and utilization, although the improvement on overhead has been reduced as a tradeoff.

We further look in to data movement overhead and utilization distinguished by different client sites (local or remote), in order to get more insights on how allocating tasks to remote site will impact system performance and how data-aware scheduling policies will mitigate the impacts.

Table I shows the data movement overhead for different client groups, distinguishing local and remote sites. That is, for three multisite cases, we count the average overhead among the clients within a same site. As shown in the table, with base scheduling policy, the data overhead of the remote sites is much higher than that of the local sites. This indicates that the significantly increased average data overhead for the two-site case seen in Figure 4(d) is due mostly to the overhead incurred by moving data from/to remote sites.

Table I
DATA OVERHEAD BY CLIENT GROUPS (AVERAGE SYSTEM LOAD)

	100/50		75/75		50/100	
policy	local	remote	local	remote	local	remote
base	0.9%	6.6%	0.8%	11.0%	0.7%	17.0%
best-fit	1.1%	0.5%	1.1%	0.7%	1.0%	1.0%
greedy	0.7%	5.3%	0.7%	8.7%	0.7%	13.6%

VII. CONCLUSION AND FUTURE WORK

As scientific workflows increasingly adopt multisite elastic cloud infrastructures, resource management and scheduling become a challenge as data movement costs between multiple sites become nontrivial. Hence, it is important to model and understand the compute and data movement dynamics under various workloads, system status, and scheduling mechanisms.

To this end, we developed a data-aware distributed workflow simulator (AweSim) based on a fine-grained network simulation framework (CODES). Further, with real job traces from a production data analysis service (MG-RAST)

in the cloud, we explored the performance variation under different workloads and resource plans. We also proposed a couple of data-aware scheduling policies and evaluated them with positive results.

ACKNOWLEDGMENTS

This work is supported by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357.

REFERENCES

[1] AWE project. https://github.com/MG-RAST/AWE

[2] CODES project. http://press3.mcs.anl.gov/codes/

[3] MG-RAST website. http://metagenomics.anl.gov

[4] Shock project. https://github.com/MG-RAST/Shock

[5] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya,"CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, 41(1):23–50, 2011.

[6] C. Carothers, D. Bauer, and S. Pearce, "ROSS: A high-performance, low memory, modular time warp system," in *Proc. of Workshop on Paral. and Distri. Simulation*, 2000.

[7] J. Cope, N. Liu, S. Lang, P. Carns, C. Carothers, R. Ross, "CODES: Enabling co-design of multi-layer exascale storage srchitectures," in *Proc. of Workshop on Emerging Supercomputing Technologies*, 2011.

[8] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *IEEE International Conference on e-Science*, 2012.

[9] D. Duplyakin, P. Marshall, K. Keahey, H. Tufo, and A. Alzabarah, "Rebalancing in a multi-cloud environment," in *Proc. of ACM Workshop on Scientific Cloud Computting*, 2013.

[10] W. J. Kent, "BLAT–the BLAST-like alignment tool," *Genome Research*, 12(4):656–664, 2002.

[11] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, "Infrastructure outsourcing in multi-cloud environment," in *Proc. of Workshop on Cloud Serv. Fed. 8th Open Cirrus Summit*, 2012.

[12] T. Kosar and M. Balman, "A new paradigm: Data-aware scheduling in grid computing," *Future Generation Computer Systems*, 25(2009):406–413, 2009.

[13] D. Lezzi, F. Lordan, R. Rafanell, and R. Badia, "Execution of scientific workflows on federated multi-cloud Infrastructures," in *Euro-Par 2013: Parallel Processing Workshops, volume 8374 of Lecture Notes in Computer Science*, pages 136–145, Springer, 2014.

[14] K. Maheshwari, E.-S. Jung, J. Meng, V. Vishwanath, and R. Kettimuthu, "Improving multisite workflow performance using model-based scheduling," in *Proc. of Int'l Conf. on Parallel Programming*, 2014.

[15] 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. Edwards, "The metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes," *BMC Bioinformatics*, 9(386):1–8, 2008.

[16] N. Mohamed, H. Lin, and W.-C. Feng, "Accelerating data-intensive genome analysis in the cloud," in *Proc. of Int'l Conf. on Bioinformatics and Computational Biology*, 2013.

[17] L. Ramakrishnan, P. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu, "Magellan: Experiences from a science cloud," in *Proc. of International Workshop on Scientific Cloud Computing*, 2011.

[18] W. Tang, J. Bischof, N. Desai, K. Mahadik, W. Gerlach, T. Harrison, A. Wilke, and F. Meyer, "Workload characterization for MG-RAST metagenomic data analytics service in the cloud," in *Proc. of IEEE Int'l Conf. on Big Data*, 2014.

[19] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-Aware, Utility-Based Job Scheduling on Blue Gene/P Systems," in Proc. of IEEE Int'l Conf. on Cluster Computing, 2009.

[20] W. Tang, J. Wilkening, N. Desai, W. Gerlach, A. Wilke, and F. Meyer, "A scalable data analysis platform for metagenomics," in *Proc. of IEEE International Conference on Big Data*, 2013.

[21] Dan Tsafrir, "Modeling, evaluating, and improving the performance of supercomputer scheduling," *Technical Report 2006-78, School of Computer Science and Engineering, the Hebrew University (PhD Thesis)*, 2006.

[22] A. Wilke, E. Glass, D. Bartels, J. Bischof, D Braithwaite, M. D'Souza, W. Gerlach, T. Harrison, K. Keegan, H. Matthews, R. Kottmann, T. Paczian, W. Tang, W. Trimble, P. Yilmaz, J. Wilkening, N. Desai, and F. Meyer, "A metagenomics portal for a democratized sequencing world," *Methods in Enzymology*, 531:487–523, 2013.

[23] A. Yoo, M. Jette, and M. Grondona, "SLURM: Simple Linux utility for resource management," *Job Scheduling Strategies for Parallel Processing, volume 2862 of Lecture Notes in Computer Science*, pages 44–60, Springer-Verlag, 2003.

[24] A. C. Zhou and B. He, "Transformation-based monetary cost optimizations for workflows in the cloud," *IEEE Transactions on Cloud Computing* 2, no.1, 85–98, 2014.

[25] H. Zou, Y. Yu, W. Tang, and H. Chen, "Improving I/O performance with adaptive data compression for big data applications," in *Proc. of International Parallel and Distributed Processing Symposium Workshops and Phd Forum*, 2014.

[26] H. Zou, Y. Yu, W. Tang, and H. Chen, "FlexAnalytics: A Flexible Data Analytics Framework for Big Data Applications with I/O Performance Improvement," *Big Data Research*, 1:4–13, 2014.