A Case Study in Using Discrete-Event Simulation to Improve the Scalability of MG-RAST

Caitlin Ross,[‡] Misbah Mubarak,[†] John Jenkins,[†] Philip Carns,[†]
Christopher D. Carothers,[‡] Robert Ross,[†] Wei Tang,[*]
Wolfgang Gerlach,[†§] Folker Meyer[†]
[‡]Computer Science Department, Rensselaer Polytechnic Institute
[†]Mathematics and Computer Science Division, Argonne National Laboratory
[§]Computation Institute, University of Chicago
[*]Google, Inc. USA
rossc3@rpi.edu, mmubarak@anl.gov, jenkins@mcs.anl.gov, carns@mcs.anl.gov,
chrisc@cs.rpi.edu, rross@mcs.anl.gov, weitang@google.com,
wgerlach@mcs.anl.gov, folker@anl.gov

ABSTRACT

As the cost of DNA sequencing has decreased, computational biology data processing platforms are experiencing an increasingly large volume of data analysis requests. The metagenomics analysis server MG-RAST at Argonne National Laboratory, a computational biology data processing platform, is receiving several terabytes of data submissions per month. However, MG-RAST currently relies on a central object-based data store, Shock, for data access and storage that can become a bottleneck under high data transfer loads, adversely affecting the job response time for end users. In this work, we use a discrete-event simulation approach to explore the use of data proxies and an enhanced, proxy-aware scheduling methodology designed to reduce the movement of the intermediate data generated during workflow processing. In this approach, Shock is supplemented with proxy storage servers, employing solid state drives, to decentralize the management and hence reduce the movement of intermediate workflow results. Discrete-event simulation provides a way to evaluate the performance of MG-RAST with increased workloads without disrupting the production system. For our case study, we extrapolate scientific workflows obtained from MG-RAST to represent future usage trends. We demonstrate that the addition of proxies and the proxy-aware scheduling methodology significantly reduces the data movement overhead by distributing the data plane, leading to substantial improvement in end-user job response time.

[*]This work was performed while Wei Tang was a postdoctoral researcher at Argonne National Laboratory.

SIGSIM-PADS '16, May 15-18, 2016, Banff, AB, Canada
© 2016 ACM. ISBN 978-1-4503-3742-7/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2901378.2901387

CCS Concepts

•Computing methodologies → Discrete-event simulation;

Keywords

discrete-event simulation, big data, clouds, MG-RAST

1. INTRODUCTION

Because of the decrease in DNA sequencing costs in recent years, the field of bioinformatics has seen an exponential increase in data submission, which poses significant challenges for data management and analysis. For example, MG-RAST [11], a metagenomics [20] analytic service provided by Argonne National Laboratory (ANL), processed 1 Tbp data (10^{12} base pairs) in the first five years, but since mid-2011 it has processed over 80 Tbp of sequence data with thousands of job submissions per month. It is currently receiving 8–16 TB of data per month, which we expect to double within 18 months. Therefore, it is critical to have scalable computing and storage resources that can efficiently handle the growth in data submission and subsequent analysis.

MG-RAST currently uses a centralized data management approach for data access and management [19]. While this approach simplifies security and long term data curation, it is a potential bottleneck when receiving a large number of data submission requests from clients, especially when the requests span multiple sites with compute resources connected by a wide area network (WAN). In addition, task scheduling in MG-RAST is currently first-come first-served (FCFS) and does not take advantage of data locality for multiple WAN sites. Since the tasks in the MG-RAST pipeline generate a large amount of intermediate data, significant WAN data movement can result, leading to increased job response times for end users. To mitigate the performance limitations of a centralized server and to minimize unnecessary data movement overhead between multiple WAN sites, we propose the addition of proxy storage servers to the MG-RAST infrastructure, along with a proxy-aware scheduling methodology that exploits data locality by placing storage proxies based on solid-state drives (SSDs) at each WAN site.

We chose to use storage proxies instead of a distributed storage system for MG-RAST because the implementation of our proposed changes will require minimal changes to the existing MG-RAST infrastructure. Also, proxies help retain the useful properties of the central data store while providing the ability to load balance the data access requests and decrease the traffic to the central data server. The storage proxies used in this work are similar to web caches; however, web caches are used to store data that tends to be more frequently accessed, whereas in this situation, the proxies are being used as a local storage for the temporary intermediate data generated during job processing.

In this work, we use discrete-event simulation to evaluate the scalability of MG-RAST when using data proxies with and without a proxy-aware scheduling methodology. As opposed to modifying the production server, simulation is a valuable approach to quickly evaluate various configurations of data-analysis platforms in order to determine an efficient and cost-effective configuration. Our simulation work extends the MG-RAST workflow simulator developed by Tang et al. [18], which evaluated two data-aware scheduling policies running on the centralized MG-RAST design. The scheduling methodologies used in that work rely on the compute-to-data cost ratio of the tasks. Our work extends the simulation to explore a proxy-aware scheduling methodology that takes data locality into account by ensuring that all data related to a job is stored on a single proxy server local to a client. This approach helps reduce data transfers between the centralized data server and WAN sites and results in lower data movement overhead. The contributions of our work are as follows.

1. **Use of discrete-event simulation for evaluation of proxy-aware scheduling**: We use discrete-event based workflow simulation to evaluate the performance of the proposed proxy-aware scheduling methodology that exploits data locality to reduce the amount of data transfers over WAN sites. We can compare the performance of proxy-aware scheduling with FCFS and data-aware methodologies, both with and without the addition of proxy storage. We also perform simulations for two workload sizes, in order to show the behavior of proxy-aware scheduling as the MG-RAST servers become more congested.

2. **Improved performance of MG-RAST when using proxy-aware scheduling**: Our results show that the use of proxy-aware scheduling leads to decreased data movement overhead and improved job response time when compared with other scheduling methodologies that do not use storage proxies. When proxy-aware scheduling is used for a large workload, we get up to 286x speedup over FCFS scheduling and up to 150x speedup over other data-aware scheduling methods. The addition of the proxy servers necessary to implement the proxy-aware scheduling requires minimal changes to the current MG-RAST infrastructure. This fact, combined with the significant performance improvement, shows that our proposed setup is cost effective and highly efficient.

3. **Extrapolated traces to represent future workloads**: To represent the future data growth trend in MG-RAST, we have used representative extrapolated traces from the production service to evaluate the proposed proxy infrastructure for MG-RAST, as well as compare the performance of proxy-aware scheduling against the previously proposed data-aware scheduling methodologies in [18]. The extrapolated trace is created through regression modeling of the relationships observed in the production trace data. Using the extrapolated traces in conjunction with discrete-event simulation enables the evaluation of MG-RAST under much larger workloads than would be possible using only the production traces.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 discusses the current and proposed infrastructures, while Section 4 describes the simulation design. In Section 5, we provide validation of the simulation as well as an evaluation of the proposed infrastructure. Section 6 summarizes our conclusions.

2. RELATED WORK

Tang et al. presented new approaches to workflow and data management systems, called AWE and Shock, respectively, that provided MG-RAST with scalable, portable, reusable, and reproducible data analysis capabilities [19]. Gerlach et al. presented an extension of the AWE and Shock ecosystem called Skyport [8]. Skyport uses Docker and Linux container virtualization technology to solve the problems involved in deploying software on multiple computing resources (e.g., dependencies on specific versions of software) while improving overall resource utilization compared with the previous virtual machine approach. Tang et al. additionally developed a discrete-event simulation framework, called AweSim, for evaluating MG-RAST deployments, focusing specifically on the effect of data-aware scheduling policies in deployments consisting of multiple sites connected by a wide-area network [18]. The WAN model of AweSim currently supports a latency and bandwidth network model that models contention at the endpoints, but not in the routing fabric. AweSim used production traces of MG-RAST workflows for evaluation, for which characterization work had previously been performed [17].

AweSim is based on the ROSS and CODES simulation frameworks. ROSS is the Rensselaer Optimistic Simulation System, a scalable discrete-event simulation framework providing sequential, synchronous parallel ("conservative") and speculative parallel ("optimistic") simulation capabilities [3]. The CODES simulation framework is built on top of ROSS and provides a comprehensive suite of HPC network, storage and workload models [6, 15, 12]. OMNeT++ [21] and ns-3 [13] are also discrete-event network simulation frameworks. Both frameworks provide various wired and wireless link layer protocol models; however, neither framework has HPC network models (e.g., dragonfly) readily available. We chose to use AweSim (and thus ROSS and CODES) in this work, because of the potential to extend AweSim in the future to use the HPC network models provided by CODES.

Event-driven simulation is also a commonly used approach to evaluate the performance of cloud systems. For example, CloudSim is a framework developed by Calheiros et al. that provides simulation of cloud infrastructures and services [2]. Network behavior can be modeled in CloudSim, however it uses a simple latency matrix that cannot capture network congestion. Chen and Deelman developed WorkflowSim, an extension of CloudSim, to allow for task dependency tracking and task clustering in workflow scheduling [4]. The system was subsequently used to model the Pegasus workflow management system [7].

The proxy approach has been employed in various situations. For example, Cirstea et al. provide a design for a prototype proxy cache for data located on a grid storage element [5]. Kungas and Dumas present a proxy cache implementation for SOAP traffic [10]. They show that the least recently used (LRU) replacement policy provides the best performance. Our work shows the addition of proxy servers in a different context: the proxy servers in our proposed infrastructure store the large volumes of intermediate data generated during job processing.

Finally, we discuss work being done in the area of job scheduling in scientific workflows at multiple WAN sites. Aside from Tang et al.'s exploration of scheduling in the context of MG-RAST [18], Jones et al. developed bandwidth-aware co-allocating meta-schedulers for mini-grids in order to improve the performance of simultaneously co-allocated jobs [9]. Specifically their methodology uses the intercluster network utilization to lower the impact of the network contention created by simultaneously co-allocated jobs. In contrast, our work explores a scheduling methodology that assigns a job to a single wide area site in order to reduce the movement of the intermediate job data generated. Szabo et al. [16] propose an evolutionary approach to task allocation on cloud resources that optimizes workflow runtime and the size of transferred data. Their approach uses a chromosome to encode the allocation of tasks to nodes and another chromosome to encode the execution order of tasks. Wang et al. [22] provide an improvement on work stealing methods. In their work, scheduling is distributed, and idle schedulers can steal the work of other schedulers, possibly incurring significant data movement overheads because of loss of data locality. Their solution is to organize work queues by data size and location in order to continue to provide load balancing while also taking data locality into account. The MG-RAST situation differs in that using proxy-aware scheduling allows for the exploitation of data locality to substantially reduce data movement overhead while also providing better load balancing of the tasks among clients at all WAN sites.

3. MG-RAST DESIGN

In this section, we first present the current design of MG-RAST, along with previously explored scheduling methodologies. We follow this with a discussion of our proposed proxy infrastructure, as well as our proxy-aware scheduling methodology.

3.1 Current Infrastructure

The system model of MG-RAST comprises the AWE [19] workflow management system and the Shock data management system [1].[1] The AWE workflow management system executes the biological analysis workflows on cloud resources. It is based on a client-server model where the AWE server schedules and assigns tasks of a job to the AWE clients, which then process the tasks locally. The AWE server receives job submissions, manages task dependencies, and performs scheduling.

The MG-RAST pipeline consists of various processing and analytical tasks. User submissions with one or more data sets are rendered into one job per data set at submission time. For each job, a number of potentially parallel tasks are generated by the AWE server using a workflow recipe. While there are data dependencies between the tasks of a given job, there are no data dependencies between tasks from different jobs. Each of the tasks is divided into one or more work units, each of which typically receives a small fraction of the input data due to the data-parallel nature of most steps.

The Shock object-based data management system is used to store biological sequences and the intermediate data generated during job execution on AWE clients. Shock is a centralized data server that handles all requests from the clients for transfer of data. It utilizes a REST API that makes it accessible from desktops, cloud, smartphones, or HPC systems.

With the increasing rate of data-processing requests being submitted to MG-RAST, workload execution is being outsourced to multiple cloud resources and user machines. A number of organizations already contribute to the compute-intensive steps of the pipeline (e.g., similarity searches) by providing their own virtual machine based instances to analyze data. Because MG-RAST uses the FCFS scheduling policy, it does not take advantage of data locality for workflow scheduling. This, combined with the use of a centralized data server with multiple WAN sites, contributes to significant data movement overhead, which can slow down the end user's job response time. Additionally, an enormous number of data access requests can transform the Shock server into a potential bottleneck.

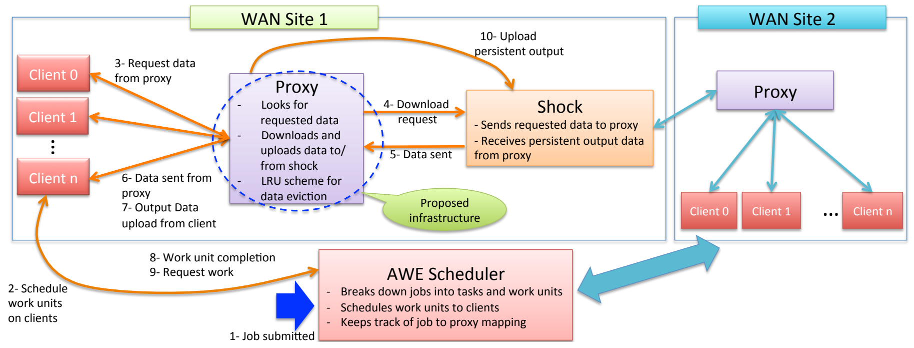
Tang et al. [18] explored three scheduling policies: FCFS, best-fit, and greedy. Best-fit and greedy are data-aware methods in which the AWE server uses the compute-to-data cost ratio of work units for scheduling decisions, putting work units with high compute-to-data ratio on sites with lower effective bandwidth and work units with comparatively lower compute-to-data ratio on sites with higher effective bandwidth. This strategy is made possible through the ability to examine and make cost predictions of each type of task/work unit, as done in Tang et al.'s workload characterization research [17]. For best-fit scheduling, only the most computationally expensive tasks are scheduled at low-bandwidth sites. Greedy scheduling, on the other hand, allows less computationally intensive tasks to run on low-bandwidth sites if no more computationally intensive tasks are pending.

3.2 Proxies and Proxy-Aware Scheduling

We propose adding one storage proxy server to each wide-area site in MG-RAST which would distribute the load of the centralized Shock data server. We also introduce a proxy-aware scheduling policy that adjusts the distribution of jobs among multiple WAN sites in order to exploit data locality and thus reduce I/O time. Figure 1 shows an architectural diagram of the proposed proxy infrastructure and its interactions with the AWE clients/server and Shock. The Shock data server is shown as part of WAN Site 1 in the figure since it is hosted locally at ANL. This is because the clients and proxies at this site have a relatively higher bandwidth connection to Shock than any remote site that is communicating over a wide area link.

Steps 3 through 7 in the figure highlight the proposed

[1]AWE and Shock can be found at https://github.com/MG-RAST/AWE and https://github.com/MG-RAST/Shock.



WAN Site 1
WAN Site 2
Client 0
Client 1
Client n
3- Request data from proxy
Proxy
Looks for requested data
Downloads and uploads data to/ from shock
LRU scheme for data eviction
4- Download request
5- Data sent
10- Upload persistent output
Shock
- Sends requested data to proxy
- Receives persistent output data from proxy
6- Data sent from proxy
7- Output Data upload from client
Proposed infrastructure
8- Work unit completion
9- Request work
2- Schedule work units on clients
AWE Scheduler
Breaks down jobs into tasks and work units
Schedules work units to clients
Keeps track of job to proxy mapping
1- Job submitted
Proxy
Client 0
Client 1
Client n

Figure 1: Proposed MG-RAST proxy infrastructure

changes to be made to the current MG-RAST infrastructure. As shown in the figure, the AWE server schedules the parallel tasks of a job so that all the data for a given job is stored on the same proxy. In proxy-aware scheduling, the AWE scheduler randomly chooses a proxy at either site when scheduling the first work unit in a job. For the given job, all work units will then be scheduled to a client at the same site as the chosen proxy. The AWE server keeps track of the job to proxy mapping so that the subsequent tasks and work units that comprise that job will be stored on that same proxy. Hence, all tasks of a job are computed at the same site, and the resulting data is stored in a single proxy, as opposed to sending tasks to different sites based on their compute-to-data cost ratio. This approach eliminates the need for proxies to transfer intermediate data between each other and the centralized Shock server, thus reducing the number of wide-area transfers. For MG-RAST, data dependencies are only between tasks within the same job. Therefore, intermediate data does not need to remain in proxy storage once its job is finished; it can simply be deleted from the proxy, instead of being sent back to Shock. Only the persistent and final output data from a job needs to be uploaded to Shock. As a proxy runs out of storage space, it will send final output data from completed jobs to Shock in order to free space for new job data.

As shown in Figure 1, clients send their data requests to a proxy server local to the clients, instead of directly to the centralized Shock server. When receiving a data access request from a client, a proxy checks whether it has the necessary data stored. If so, the proxy sends the data to the client. Otherwise the proxy requests the data from the Shock server before sending to the client. Since there is no one-to-one mapping from work units of a given task to work units of the next task in the pipeline, we design the proxies to store data at task-level granularity. Proxies also request data from Shock at task-level granularity, which reduces the number of accesses to Shock. In essence, when the proxies transfer data to and from Shock, all the data needed for the work units that compose that task is transferred. In contrast, transfers between proxies and clients is done at the work unit level, so the client receives only the data necessary for its assigned work unit. In the current MG-RAST infrastructure (i.e., without proxies), clients always request data from Shock at work unit granularity.

Figure 2 shows a representative example of best-fit scheduling and proxy-aware scheduling. In the figure, each horizontal line represents a different client at that site, with a limited bandwidth available between the two WAN sites. The figure shows Jobs A and B, with various work units being computed for each job. Each box represents a work unit, with work units from the same task being given the same coloring. Circles group work units that feed into the next task in the pipeline. For example, task 9 in the MG-RAST pipeline depends on both tasks 5 and 8, so work units for tasks 5 and 8 can be grouped together. The dotted lines pointing from one group of tasks to another show the dependencies of the tasks.

In the example, Job A initially has tasks 5 and 8 being computed concurrently because there are no dependencies between these two tasks. Task 9 is dependent on both tasks' output data. Job B shows tasks 2 and 3 being computed, where task 3 is dependent on task 2. The left side of the figure shows best-fit scheduling without using proxies. Task 5 is the most computationally expensive, as found in [17] (i.e., its relative cost of I/O is the lowest among all tasks), so its work units can be computed at either site. The right side of the figure shows the same situation using proxy-aware scheduling, where an entire job is computed by clients at a single site, because all intermediate data of a job is being stored on the same proxy.

In the best-fit scheduling example, task 5's work units at the remote site take longer to complete because of the need to retrieve the data from Shock and send the output back. The clients at WAN site 1 become idle waiting for the work units at WAN Site 2 to complete, so the AWE server starts scheduling another job's tasks to those clients. When Job A's task 5 is completed, those clients are now busy on another job, and have to wait longer to start Job A's task 9. In comparison, for proxy-aware scheduling the jobs can be split among sites, and the same jobs can be completed in less time.

4. SIMULATION DESIGN

To explore our proposed proxy architecture, we extended the AweSim simulator, introduced in Section 2. Although ROSS is capable of parallel execution, we use sequential execution in this work, because the scale of the simulations performed still experiences good performance with an event rate of up to 2.4 million events per second. We first provide a brief overview of AweSim before describing our extensions.

AweSim has five distinct simulation entities (known as logical processes, or LPs, in ROSS) to represent the MG-RAST analytic service: (1) an LP representing the Shock

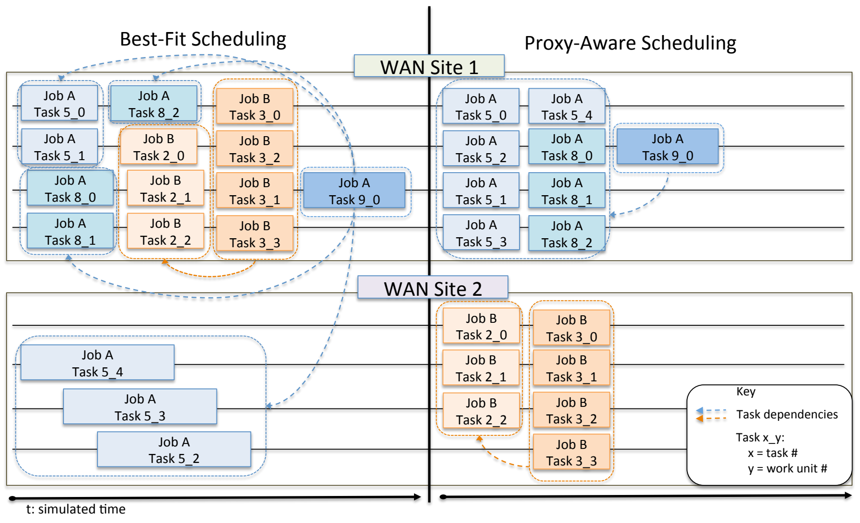


Figure 2: Representative examples of best-fit and proxy-aware scheduling

data server, (2) an LP representing the AWE scheduler, (3) an LP representing each AWE client, (4) a router LP representing the communication between the Shock server and AWE entities, and (5) an LP representing the WAN endpoints. All LP types have one instantiation, with the exception of the AWE client, which can have any number of instantiations, and the WAN endpoints, which have one instantiation per site. The AWE client LPs can be grouped into multiple sites, with different effective bandwidths to and from the Shock LP. The simulation proceeds through the issuance of time-stamped messages or events between LPs, representing data transfers, work unit executions, and control events, among others.

In this work, we have extended AweSim to include an LP representing the storage proxy discussed in Section 3.2. In the simulations, we place one proxy at each WAN site. The proxies have a configurable amount of storage and use a least recently used (LRU) protocol for evicting data. If the data evicted is intermediate job data that is no longer needed, it is simply deleted. Otherwise the data is sent back to the central Shock data server.

The event flow of the MG-RAST model is based on the the MG-RAST infrastructure diagram shown in Figure 1. The various event types used in AweSim are as follows:

- Work unit assignment: Upon scheduling a work unit to a client, the AWE scheduler issues an event to the client notifying them of the task type and which dataset to process.

- Data download request: This event is issued from client to proxy to request the data necessary for processing the assigned work unit. If the proxy does not already have the requested data, the proxy requests the data from the router, which forwards the request to the Shock server. In configurations that do not use the proxy LP, the client sends this event to the router instead, which then sends the download request event to Shock.

- Data download: After receiving a request for data download, Shock sends an event with the data to the router, which forwards it to either the proxy or the client, depending on whether the configuration includes proxy LPs. The proxy also uses this event type to send the requested data to the client. The amount of time to transfer the data is based on the size of the dataset and the bandwidth configured between the endpoints.

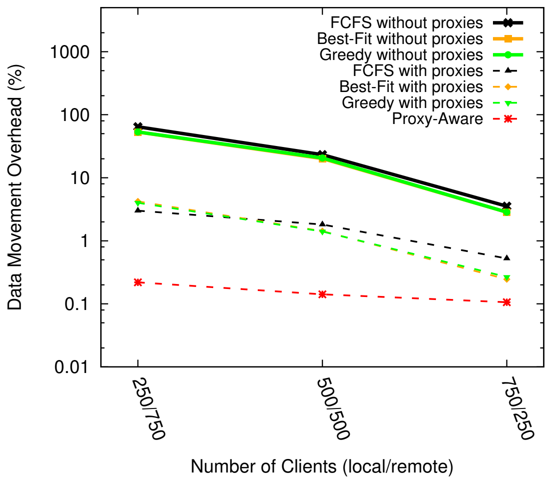
- Data upload: After the client performs the necessary computation for its work unit, it sends an event to upload the output data to either a proxy or the router, dependent on the simulation configuration. If sent to the proxy, the proxy stores the data, evicting stored data by a LRU protocol if necessary. The proxy sends this upload event to the router for any evicted data that must be stored on Shock. Again, the time to transfer data is based on the size of the dataset and the configured bandwidth.

- Data upload acknowledgement: After the data is uploaded, Shock sends an acknowledgement event to the router, which sends the acknowledgement to either the proxy or client, dependent on the given configuration. In configurations where proxies are used, the proxy sends this acknowledgement to the client when it stores the data sent from the client.

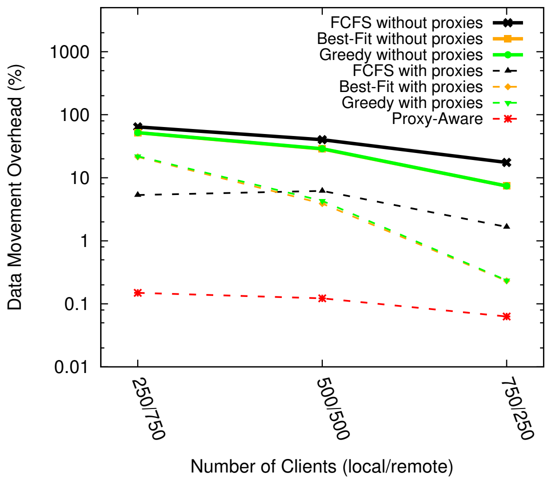
- Work unit completion: Once a client has received an acknowledgement that its data has been uploaded, it sends the AWE scheduler an event to signify that the work unit has been completed.

- Work request: At the beginning of the simulation and anytime a client has received an acknowledgement from uploading a work unit's output data, it sends a message to the AWE scheduler to request work.

In the LP representing the AWE scheduler, we have implemented proxy-aware scheduling as previously described in Section 3.2. In our modified version, AweSim can also be configured to use proxies in conjunction with FCFS, best-fit, and greedy scheduling methodologies.



(a) 64 TB workload



(b) 128 TB workload

Figure 3: Average data movement overhead for proxy-aware, best-fit, greedy and FCFS scheduling algorithms with 1000 clients and an input of 64 TB (a) and 128 TB (b).

5. EVALUATION

In this section, we present an evaluation of the proposed proxy-aware scheduling methodology for MG-RAST using discrete-event simulation. First we describe the experimental setup. Then we provide validation of the simulation, followed by experimental results.

5.1 Experimental Setup

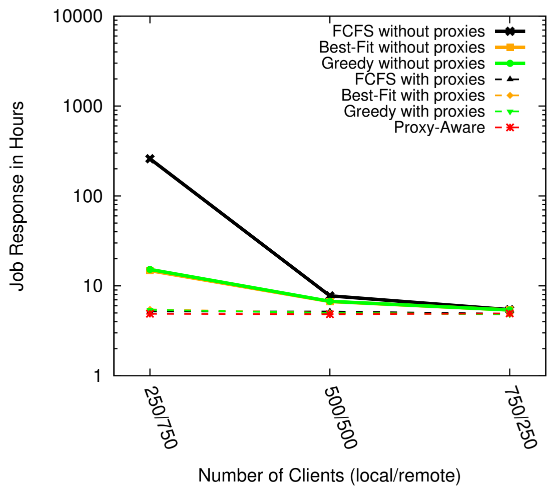
For the experiments, we perform two-site simulations. One site is configured as "local" with respect to Shock, with a comparatively faster connection to the centralized data store. The other site is configured as "remote," with a comparatively slower connection. Currently, MG-RAST runs with 50 to 150 clients [18]; however, with the anticipated increase in data submissions, more clients will be necessary. Therefore, we look at three configurations with a total of 1,000 clients: equal numbers of clients at each site (500 local, 500 remote), more clients at the local site (750 local, 250 remote), and more clients at the remote site (250 local, 750 remote). We configure each site to have one proxy with sufficient storage space to hold the intermediate output data. As stated previously, there are no dependencies between different jobs, so intermediate job data can be deleted from the proxy once it is no longer needed, freeing up storage space for new jobs to store their intermediate data. With our problem size, a 4 TB proxy storage size is sufficient.

We use the same Shock-to-site bandwidth as in the original AweSim work [18]: 500 MB/s download and 100 MB/s upload for the local site, and 10 times slower download and upload rates for the remote server. Between WAN sites, the bandwidth is 50 MB/s, which is the same as the download bandwidth between the remote site and Shock. The bandwidth between the clients and proxies at a given site is 750 MB/s, and the latency is 0.1 ms, which is based on the Samsung SSD PM863 specifications [23, 14].

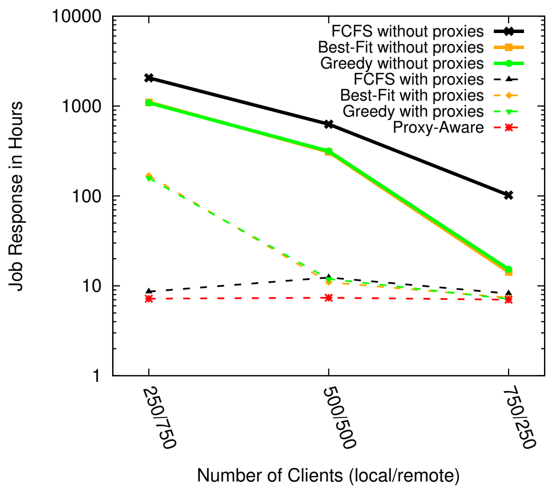
The metrics we use for evaluation are job response time, client utilization, daily workload volume, and data movement overhead.

- **Job response time** is the elapsed time (in hours) from when a job is first submitted until the job is completed.
- **Client utilization** is measured as the percentage of time that clients spend processing a work unit, including the download and upload of inputs and results.
- **Data movement overhead** is the percentage of time to transfer data relative to the total compute and data transfer time for each client. We report the data movement overhead averaged over all clients.
- **Daily workload volume** is the total input data size in GiB divided by the simulated time to complete all jobs in days. This metric is constrained by the job arrival times in the workload trace being used for simulation, such that any given workload has a maximum daily workload volume regardless of the system configuration.

For the experiments, we use a production MG-RAST job trace that contains data for four months of job submissions. The trace contains 12,483 jobs, 124,830 tasks, and 190,205 work units. Since the number of data submission requests being submitted to MG-RAST is increasing tremendously, we extrapolate the production traces to create additional jobs on the fly during simulation. These extrapolated traces represent a future trend of the growth in data analysis requests. We used regression models to model the relationships in the trace data. The time it takes a client to execute a work unit is modeled with respect to the input size of the work unit coming from the trace. The output size of a work unit is also modeled with respect to its input size. The input size of a task is modeled with respect to the output size of the previous task that it is dependent on. This is necessary in cases where MG-RAST will output the data in multiple file formats for a task, but the next dependent task will use data from only one of the file formats. We also use a uniform random number generator to add randomness to the



(a) 64 TB workload



(b) 128 TB workload

Figure 4: Average job response time for proxy-aware, best-fit, greedy and FCFS scheduling algorithms with 1000 clients and an input of 64 TB (a) and 128 TB (b).

trace values determined by the regression models. Job arrival times of the extrapolated trace are determined using a Poisson distribution.

We configure the simulations to model a time period of 100 days, and we perform the experiments with two workload sizes. The smaller workload uses a total job input size of 64 TB; the larger workload has a total job input size of 128 TB. The four-month production trace, which provides approximately 10.5 TB of input job data, is used with the extrapolated trace. Job arrival times of the four-month production trace were compressed to fit in this 100-day simulated time period. Using the extrapolated trace along with the production trace results in approximately 37,500 jobs, 375,000 tasks, and 806,000 work units for the 64 TB workload and 68,900 jobs, 689,000 tasks, and 1,560,000 work units for the 128 TB workload.

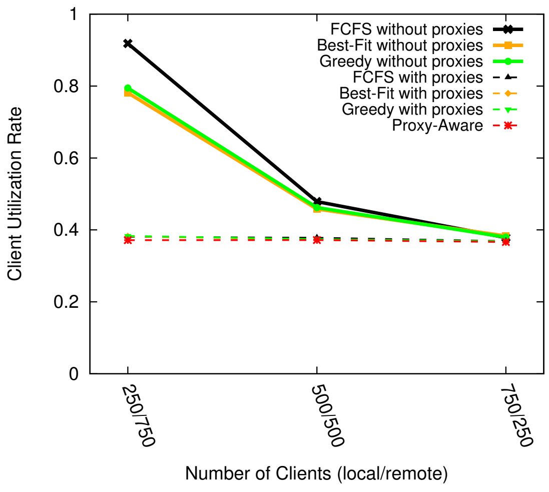
5.2 Validation

The behavior of the previous version of AweSim (i.e., without the proxy LP) was validated by Tang et al. [18] for one-site simulations by comparing simulation output with metrics computed by using production MG-RAST traces. To validate the behavior of AweSim with the addition of proxies, we compare the output of our extended version with the original simulator, in order to ensure that certain metrics stay the same. We are unable to perform further validation of the proxies with production MG-RAST data, as the proxies have not been implemented into MG-RAST. However, we perform both one and two site simulations and check that both versions perform the same amount of work and that the amount of data transferred is the same. For data transfer, we compare the amount of data uploaded and downloaded between clients and the central Shock server in the prior version (i.e., without proxies) with the amount transferred between clients and proxies in our extended version. We expect these values to be the same in each version because the proxies are now distributing the load of Shock and will, at some point, hold all job input and output data, as well as intermediate job data. For the simulations with a 64 TB input workload, the amount of data uploaded by clients is approximately 354 TB and the amount downloaded is approximately 455 TB. For the 128 TB workload, the amount of data uploaded and downloaded by clients is approximately 725 TB and 929 TB, respectively.

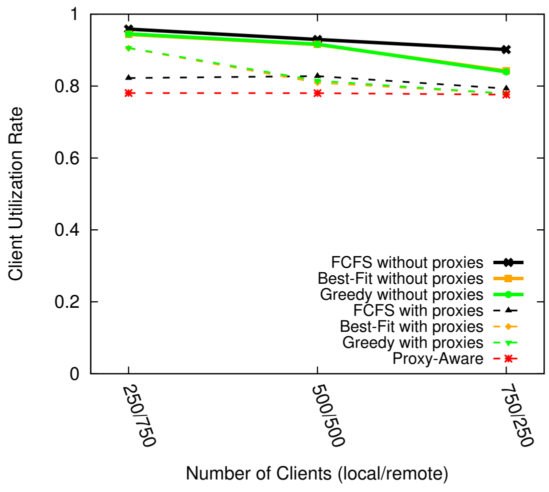
5.3 Experimental Results

To explore the effectiveness of proxy-aware scheduling, we compare the FCFS, best-fit, and greedy scheduling policies with the proxy-aware scheduling. For FCFS and the data-aware methods, we perform simulations with and without proxy storage. Data movement overhead is shown in Figure 3 for both 64 TB and 128 TB workload simulations. In all three client configurations, FCFS, best-fit, and greedy scheduling without proxies have the highest data movement overheads. As the number of remote clients increase, so does the overhead for all three scheduling methodologies. In all configurations, using proxies in conjunction with these scheduling methodologies provides lower data movement overhead. When there are more local clients or equal numbers of clients at each site, best-fit, and greedy scheduling with proxies have a lower overhead than does FCFS with proxies. When there are more remote clients than local clients, this situation reverses, with the difference becoming more pronounced in the 128 TB workload. FCFS with proxies has a lower overhead in this client configuration, because best-fit and greedy are more restrictive in assigning work units to the remote site clients, making it is more difficult for scheduling at the remote site to exploit data locality to reduce data movement.

While adding proxies decreases the data movement overhead of all scheduling methodologies, proxy-aware scheduling is the only methodology to decrease to under 1% in all client configurations for both workloads. When the workload doubles, proxy-aware's overhead slightly decreases, because the time the clients spend computing increases at a faster rate than the time the clients spend waiting on data

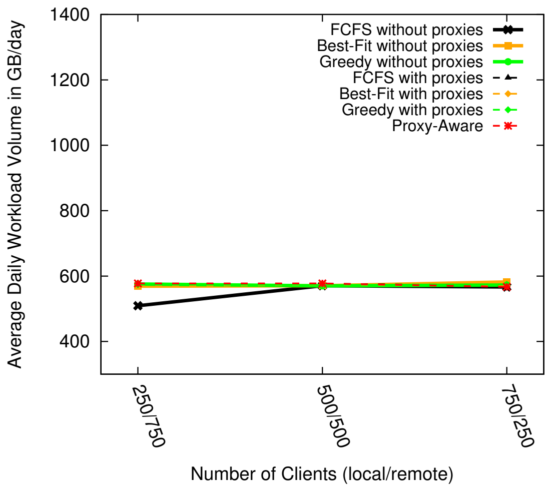


(a) 64 TB workload

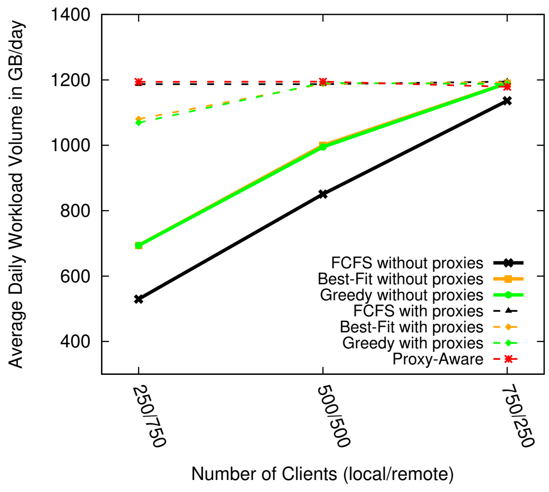


(b) 128 TB workload

Figure 5: Average client utilization for proxy-aware, best-fit, greedy and FCFS scheduling algorithms with 1000 clients and an input of 64 TB (a) and 128 TB (b).



(a) 64 TB workload



(b) 128 TB workload

Figure 6: Average daily workload volume for proxy-aware, best-fit, greedy and FCFS scheduling algorithms with 1000 clients and an input of 64 TB (a) and 128 TB (b).

transfer. The reason proxy-aware scheduling provides low data movement overhead in all cases is that the data is almost always on the proxy when the clients request it, since the intermediate data generated by tasks are never transferred to the central data server. The data movement overhead metric is taking into account the movement of initial job data from Shock to the proxy and then on to a client; however, this transfer is negligible compared to the amount of time the clients spend actually computing work units. When using proxies (for any scheduling methodology), the transfer of a job's final output data from a proxy to Shock is not factored into the data movement overhead, as the clients do not have to wait on this transfer. The clients only need to wait for the acknowledgment that the proxy has received the data before requesting more work from the AWE server. Whereas with the FCFS and the data-aware scheduling methods without proxies, the upload of final output data is accounted for in the calculation of data movement overhead, because the clients wait for an acknowledgment from the central Shock server before requesting more work.

The low data movement overheads observed in the proxy-aware scheduling also lead to a consistently small average job response time of approximately 5–7 hours, regardless of workload size, as shown in Figure 4. For the 64 TB workload, the addition of proxies improves the job response time, especially for the 250/750 client configuration, where the speedup is 3x over greedy and best-fit scheduling without proxies and the speedup is 52x over FCFS without proxies.

For the 128 TB workload, proxies still provide an improved job response time over not using proxies, but proxy-aware scheduling has the lowest job response time in all client configurations. In this larger workload, the largest improvements in job response time are seen in the case where there are more clients at the remote site. In this case, proxy-aware scheduling has a speedup of 286x over FCFS without proxies and 150x over greedy and best-fit without proxies. Proxy-aware scheduling has only a 1.1x speedup over FCFS with proxies, but a 22x speedup over best-fit and greedy with proxies. These large improvements in job response time with the addition of proxies are due to the fact that clients are spending much less time in downloading and uploading the data than with the previous scheduling methodologies.

Client utilization rate is shown in Figure 5. For the 64 TB workload, not using proxies results in a high client utilization rate in the case of more remote clients. The addition of proxies to any scheduling methodology keeps client utilization to approximately 37% in all client configurations. In the 128 TB case, client utilization is much higher for all scheduling methodologies in all client configurations. However, the addition of proxies results in a lower client utilization rate, with proxy-aware scheduling providing the lowest utilization in all configurations because clients are spending less time waiting on data transfer. Thus, with the addition of proxies, the clients can handle more work, even when there are a large number of clients at the remote site.

The results for average daily workload volume are shown in Figure 6. For the smaller workload, all scheduling methodologies, except FCFS without proxies, have a consistent average daily workload volume of approximately 575 GB/day in all client configurations. Since the workload volume is limited by the job arrival times in the workload trace, approximately 575 GB/day is the maximum daily volume for the 64 TB workload. Similarly for the 128 TB workload, approximately 1200 GB/day is the maximum. If the system is unable to keep up with the arrival rate of jobs for a given configuration, then the average daily workload volume for that configuration will be lower than the maximum. Proxy-aware scheduling and FCFS with proxies are the only methods that consistently meet the maximum average daily workload volume in all configurations, for both workloads tested. Greedy and best-fit scheduling with proxies perform the same except in the 250/750 client configurations, where the workload volume is reduced to approximately 1075 GB/day.

As discussed in Section 5.2, when proxies are not used, the amount of data downloaded from and uploaded to Shock in the 64 TB workload simulations is approximately 455 TB and 354 TB, respectively. In these simulations, Shock is accessed by the clients about 806,000 times. Adding proxy storage (to any scheduling policy) reduces the number of Shock accesses to approximately 37,000. The amount of data transferred to Shock is decreased as well, to approximately 64 TB downloaded and 39 TB uploaded. For the 128 TB simulations without proxies, the amount downloaded from Shock is 929 TB, the amount uploaded to Shock is 725 TB, and the number of accesses to Shock is approximately 1,560,000. Using proxies for any scheduling methodology reduces these amounts to 128 TB downloaded from Shock, 79 TB uploaded to Shock, and 69,000 Shock accesses.

6. CONCLUSION

MG-RAST, the metagenomics analytic service at Argonne National Laboratory, is seeing a tremendous growth in data processing requests. MG-RAST currently uses a central data store, Shock, for data management. While the central data store has the advantages of long-term data curation and added security, it can become a bottleneck for large volumes of data submissions. We propose the addition of proxy storage servers to the MG-RAST infrastructure, which store the intermediate data generated by jobs to help reduce the number of data access requests to Shock. We also propose a proxy-aware scheduling methodology, which takes data locality into account while scheduling jobs. Discrete-event simulation is used to evaluate our proposed changes to the MG-RAST infrastructure and the behavior of our proposed proxy-aware scheduling methodology. By extrapolating the production traces from MG-RAST, we are able to evaluate the various scheduling methodologies used for MG-RAST under increased workloads. We have demonstrated that the addition of proxy storage servers can substantially decrease the data movement overhead between multiple WAN sites and lead to a significant improvement in end-user job response time. Using proxies speeds the job response time for all scheduling methodologies evaluated, but using a proxy-aware scheduling methodology provides the most speedup over all client configurations in larger workloads.

Acknowledgments

This material was based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computer Research (ASCR) under contract DE-AC02-06CH11357.

7. REFERENCES

[1] J. Bischof, A. Wilke, W. Gerlach, T. Harrison, T. Paczian, W. Tang, J. Wilkening, N. Desai, and F. Meyer. Shock: Active storage for multicloud streaming data analysis. In *2nd IEEE/ACM International Symposium on Big Data Computing*. IEEE, 2015.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De 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.

[3] C. D. Carothers, D. Bauer, and S. Pearce. ROSS: A high-performance, low-memory, modular time warp system. *Journal of Parallel and Distributed Computing*, 62(11):1648–1669, 2002.

[4] W. Chen and E. Deelman. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th International Conference on E-Science*, pages 1–8. IEEE, 2012.

[5] T. C. Cirstea, J. J. Keijser, O. A. Koeroo, R. Starink, and J. A. Templon. A scalable proxy cache for grid data access. In *Journal of Physics: Conference Series*, volume 396. IOP Publishing, 2012.

[6] J. Cope, N. Liu, S. Lang, P. Carns, C. Carothers, and R. Ross. Codes: Enabling co-design of multi-layer exascale storage architectures. *Proceedings of the*

Workshop on Emerging Supercomputing Technologies, 2011.

[7] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 2014.

[8] W. Gerlach, W. Tang, K. Keegan, T. Harrison, A. Wilke, J. Bischof, M. D'Souza, S. Devoid, D. Murphy-Olson, N. Desai, et al. Skyport: Container-based execution environment management for multi-cloud scientific workflows. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*, pages 25–32. IEEE Press, 2014.

[9] W. M. Jones, L. W. Pang, W. B. Ligon, and D. Stanzione. Bandwidth-aware co-allocating meta-schedulers for mini-grid architectures. In *Cluster Computing, 2004 IEEE International Conference on*, pages 45–54. IEEE, 2004.

[10] P. Küngas and M. Dumas. Configurable SOAP proxy cache for data provisioning web services. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1614–1621. ACM, 2011.

[11] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. 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(1):386, 2008.

[12] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns. A case study in using massively parallel simulation for extreme-scale torus network codesign. In *Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation*, pages 27–38. ACM, 2014.

[13] ns-3. https://www.nsnam.org/. Accessed Mar. 15, 2016.

[14] Samsung PM863 and SM863 for Data Centers. http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_PM863_and_SM863_Brochure_web.pdf. Accessed Jan. 21, 2016.

[15] S. Snyder, P. Carns, J. Jenkins, K. Harms, R. Ross, M. Mubarak, and C. Carothers. A case for epidemic fault detection and group membership in HPC storage systems. In *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, pages 237–248. Springer, 2014.

[16] C. Szabo, Q. Z. Sheng, T. Kroeger, Y. Zhang, and J. Yu. Science in the cloud: allocation and execution of data-intensive scientific workflows. *Journal of Grid Computing*, 12(2):245–264, 2014.

[17] 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 *Big Data (Big Data), 2014 IEEE International Conference on*, pages 56–63. IEEE, 2014.

[18] W. Tang, J. Jenkins, F. Meyer, R. Ross, R. Kettimuthu, L. Winkler, X. Yang, T. Lehman, and N. Desai. Data-aware resource scheduling for multicloud workflows: A fine-grained simulation approach. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 887–892. IEEE, 2014.

[19] W. Tang, J. Wilkening, N. Desai, W. Gerlach, A. Wilke, and F. Meyer. A scalable data analysis platform for metagenomics. In *Big Data, 2013 IEEE International Conference on*, pages 21–26. IEEE, 2013.

[20] T. Thomas, J. Gilbert, and F. Meyer. Metagenomics – a guide from sampling to data analysis. *Microb Inform Exp*, 2(3):1–12, 2012.

[21] A. Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM 2001)*, volume 9, page 65, 2001.

[22] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang, and I. Raicu. Optimizing load balancing and data-locality with data-aware scheduling. *2014 IEEE International Conference on Big Data (Big Data)*, pages 119–128, 2014.

[23] Why SSDs are Awesome. http://www.samsung.com/global/business/semiconductor/minisite/SSD/global/html/whitepaper/whitepaper01.html. Accessed Jan. 21, 2016.