Efficient Data Staging Using Performance-Based Adaptation and Policy-Based Resource Allocation

Ann L. Chervenak[1], Alex Sim[2], Junmin Gu[2], Robert Schuler[1], Nandan Hirpathak[1]

[1]University of Southern California
Information Sciences Institute
Marina del Rey, CA, USA
{annc, schuler, nandan}@isi.edu

[2]Lawrence Berkeley National Laboratory
Scientific Data Management Research Group
Berkeley, CA, USA
{asim, jgu}@lbl.gov

***Abstract*—Before scientific analyses run on shared infrastructure, such as the Open Science Grid or XSEDE, scientists must often transfer or *stage* key data sets those resources. Often these datasets consist of many files that may be transferred by multiple clients in parallel. We study two techniques that improve the use of available resources for these large, long-running, multi-file transfers. First, we adapt transfer parameters for multi-file transfers based on recent transfer performance. Second, we use VO and site policies to influence the allocation of system resources for transfers, such as available transfer streams. We describe our system design and summarize its implementation and performance.**

Keywords—data transfer, performance-based adaptation, policy-based resource allocation

I. INTRODUCTION

Scientific collaborations are generating data at ever-increasing rates using scientific instruments (e.g., the Large Hadron Collider [1], the Large Synoptic Survey Telescope [2], the Laser Interferometer Gravitational Wave Observatory [3]), large-scale simulation (e.g., climate modeling), and sensors. These data are then shared by scientists across a resarch collaboration or Virtual Organization (VO) [4]. These data sets must be transferred efficiently among collaborating institutions. One important use case for scientific data transfer is *data staging* where a scientific data set is transferred or staged from the site where data are created or stored to a computational site where analysis will take place. Staging data for large computational jobs is a resource intensive activity found in many scientific applications and environments [5-7].

Data staging transfer performance is influenced by several factors, including the characteristics of client and server machines (memory, network buffers, etc.), storage resources (disk IO bandwidth), and the networking environment (network interfaces, network bandwidth). A common problem is that over-provisioning these resources results in suboptimal transfer throughput. Not only do some jobs get "starved out," but overall transfer performance is reduced due to the congestion caused by over-provisioning.

Our goal is to improve transfer throughput and latency for data staging operations. We apply two techniques to adapt the resources and parameters used for long running, multi-file data transfers: *policy-based allocation of data transfer resources* based on Virtual Organization (VO) and site level policies and *client-side adaptation of transfer parameters* based on recent transfer performance.

Our driving use case involves staging a large, multi-file data set from one site to another in preparation for conducting an analysis of the data set on a high performance computing platform (the destination site). This is reflective of many applications that run on shared infrastructure, such as the Open Science Grid [8] and XSEDE [9] in the United States.

We developed two components to implement our techniques, as shown in Fig. 1: a Policy Service (PS) that is responsible for allocating resources to data transfer clients based on VO and site policies and a modified data transfer client that adapts data transfer parameters within the allocation provided by the PS baesd on recent transfer performance.

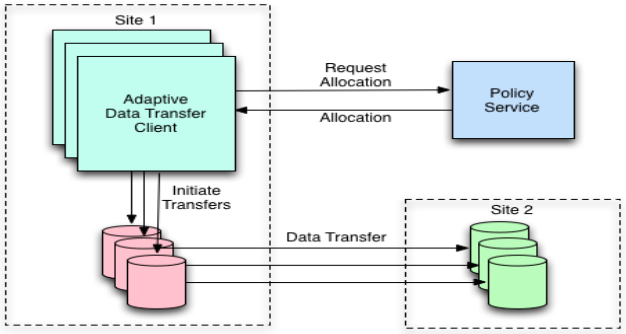

Fig. 1. Service interactions for data transfer use case

The figure shows a data staging operation taking place from Site 1, where data are stored, to the destination Site 2, where analysis will be performed. At the source site, multiple adaptive data transfer clients are each responsible for transferring a portion of the multi-file data set to the destination site. Each transfer client first communicates with the Policy Service to request an allocation of resources for transferring data; once it receives an allocation, the transfer client initiates data transfers from the source site to the destination and adapts the parameters for subsequent transfers within its allocation based on recent performance. Periodically, the clients communicate with the Policy Service to update their resource allocations.

In the next section, we present the design of the Policy Service for resource allocation and the adaptive data transfer client. Next, we discuss our implementation and breifly describe the performance of our techniques. We present related work and summarize our results.

II. SYSTEM DESIGN

A. Policy-Based Resource Allocation

The Policy Service (PS) allocates available resources to data transfer clients. The PS makes its allocation decisions based on resource availability, PS parameters specified by site or Virtual Organization administrators, and on the policies that have been implemented in the service. For example, PS parameters may specify the maximum number of streams that may be allocated to all clients transferring data between a source and destination, or they may limit the resources allocated to a single client.

Before initiating its data transfers, an adaptive transfer client first requests a resource allocation from the Policy Service. If resources are available, the PS determines an appropriate allocation for the client based on its policies and parameters and returns that allocation to the client. The client then uses its own algorithms to adapt within that allocation based on the transfer performance.

Periodically, data transfer clients may again contact the Policy Service to request adjustments to their resource allocations. If additional resources are available, the PS again consults its policies and parameters to determine whether to increase the client's resource alloction and by how much. It then returns the new allocation to the requesting client.

Once a data transfer client completes its outstanding transfers, it informs the PS, which releases resources allocated to the client and makes them available to other clients.

The Policy Service supports a range of policies for allocating resources for network data transfers, depending on the value of several input parameters, defined in Table I. The current PS determines the allocation of transfer streams to each client. (Other possible approaches include allocating bandwidth or limiting the transfers allowed per client.) PS parameters specify the maximum number of streams that may be allocated between a pair of source/destination hosts and to each client. Additional parameters specify the number of streams that may be allocated on an initial request from a client and on an update request; both of these are subject to the availability of unallocated streams between source and destination hosts.

TABLE I. POLICY SERVICE PARAMETERS

Policy Service Parameter	***Definition***
Maximum total streams for source/destination	Maximum concurrent streams active between a pair of source/destination sites
Maximum streams per client	Maximum allocation to a single client from the Policy Service
Initial stream allocation	On a new client request, the Policy Service attempts to allocate this many streams (subject to resource availability)
Update allocation increment	On an update client request, the Policy Service attempts to increase the allocation by this value (subject to resource availability)

These parameters allow the PS to support a wide range of policies. For example, for a *greedy* allocation strategy that allocates available streams to the first clients that request them, the PS parameters can specify that transfer clients receive a large initial stream allocation and that allocations are incremented by a large value in response to update requests. To support a more *incremental* increase in resource usage by clients, the PS parameters can specify smaller values for initial stream allocations and update increments.

While the PS provides recommended resource allocations, it is up to clients to keep their utilitzation within recommended limits; the current PS implementation does not enforce resource limits at the client.

B. Client Side Transfer Adaptation

The adaptive data transfer client requests a resource allocation from the Policy Service before it begins its data transfer operations. Once resources have been allocated to a client by the Policy Service, the client manages and adapts its transfers within that allocation limit based on recent transfer performance and client configuration parameters. A data transfer client may transfer multiple files between a source and destination host. In such cases, the client can modify or *adapt* its transfer parameters (concurrency, number of parallel streams, buffer size, etc.) for subsequent transfers each time a transfer completes, as described below.

The data transfer client uses the parameters shown in Table II.

TABLE II. DATA TRANSFER CLIENT PARAMETERS

Data Transfer Client Parameter	***Definition***
Initial concurrency	Number of active transfers initiated by a client when it begins transferring data
Maximum concurrency	Maximum number of active file transfers by a client. The client may reach this maximum value by adaptation.
Parallelism	Number of parallel streams per file transfer
Adaptation delay time	How often the client updates a resource allocation from the Policy Service; expressed as number of completed transfers before adaptation occurs.
Adaptation increment/ decrement	How much the concurrency level increases/ decreases when the client adapts up or down within its resource allocation.

These parameters specify an initial *concurrency* or number of simultaneous data transfers that are initiated by the client when it begins transferring data as well as a maximum concurrency that the client may reach via adaptation. Another parameter specifies a *parallelism* level, or number of streams used per file transfer; this value is constant for all transfers initiated by the client in our implementation. Two parameters control client-side *adaptation*, which modifies the concurrency of the data transfers in response to observed performance. The *adaptation delay time* parameter specifies the frequency of adaptation in terms of how many transfers complete before an adaptation occurs. The *adaptation increment/decrement* parameter specifies how much the concurrency level increases or decreases during adaptation. For fast adaptation, the client may be configured to adapt more frequently and/or to increase or decrease concurrency by a large increment. For slow adaptation, parameters may specify a long adaptation delay time and/or a small increment for changing concurrency level.

To decide whether to adapt its parameters for subsequent transfers, the data transfer client predicts whether the overall throughput will be improved by increasing or decreasing the number of concurrent transfers for the client. This decision is

based on earlier work [10, 11]. When the adaptive data transfer client completes the number of transfers specified by the adaptation delay time parameter, it compares the performance of the transfers that just completed (e.g., elapsed transfer time, total throughput) with previous measured performance between the same pair of source and destination hosts. If the performance difference exceeds a threshold set by the adaptive transfer client, the client adjusts the concurrency for its subsequent transfers up or down by the value of the adaptation increment/decrement parameter. If recent performance is lower than past performance, the client decreases its concurrency; if recent performance is higher, the client increases the concurrency of subsequent transfers to take advantage of additional available bandwidth. If recent performance is close to past performance, the concurrency level is unchanged.

A data transfer client may periodically request an updated resource allocation from the PS and then further adapt within the new resource limits.

A client may receive a large allocation of streams from the Policy Service, but still be configured by its parameters to begin transfers with a low initial concurrency level and adapt up to consume more streams over time.

III. IMPLEMENTATION

A. Policy Service

We implemented the Policy Service as a RESTful Web service in the Python programming language on the Web.py framework [12]. Currently, it uses the lightweight CherryPy embedded HTTP server [13] and maintains state in memory. The resource representation is communicated using the JavaScript Object Notation (JSON) format [14]. The PS provides an abstraction layer between the Web service protocol handling and the policy implementation. This enables users of the PS to implement or customize transfer allocation policies.

B. Adaptive Data Transfer Client

The Adaptive Data Transfer Client is a standalone, command-line client implemented in the Java programming language. It extends the conventional SRM-Copy command-line client [15] with the adaptive data transfer (ADT) library. The ADT library adjusts transfer parameters based on the observed throughput performance differences, as described in the last section. It also communicates with the PS over an HTTP connection to negotiate transfer request allocations. The client application manages multiple third-party GridFTP transfers [16] between source and destination sites.

IV. EVALUATION

Next, we briefly describe our experimental results with our techniques for improving data staging performance using client side transfer adaptation and policy-based resource allocations. We transferred a large data set in the wide area, from Lawrence Berkeley National Laboratory in Berkeley, CA, to the University of Nebraska at Lincoln in Lincoln, NE. The goal of this experiment was to show the advantage of using our techniques in resource-constrained environments. We used a single node at LBNL on which we ran 8 adaptive data clients, allowing them to compete for resources.

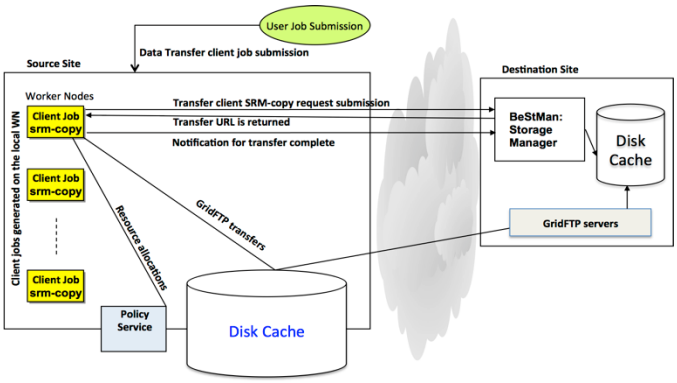

Fig. 2. Experimental testbed setup

At UNL, the test run used a front end BeStMan storage manager (sdm.lbl.gov/bestman/) and 9 gridftp frontend storage nodes with backend HDFS storage. Both LBNL and UNL have 10 Gbps connections to the wide area network. The network between the sites crosses domains from ESnet (www.es.net) to Internet2 (www.internet2.edu). The network and resources are shared with other traffic. Fig. 2 shows this setup. The 8 clients at LBNL transferred an existing data set of size 260 Gbytes made up of 488 files. Each client transfers 60 or 62 files. The node's configuration included a Quad CPU, 8GB DDR2 memory, 4TB RAID array, and 1 Gbps network interface.

In the experiment, we compare the performance of fast increases in policy based allocation and fast data transfer client adaptation with the performance of non-adaptive transfers. Table III shows parameters for our experimental scenario. We ran the experiment three times; each run produced similar patterns in the performance. Below, we show representative graphs for the experiment.

TABLE III. PARAMETERS FOR COMPARATIVE EXPERIMENTS

Parameters for all Comparative Experiments	***Value***
Maximum total streams between source/destination	640
Number of clients	8
Parallelism: streams per file transfer	4
Adaptation increment/decrement	2 concurrency (8 streams)
Initial Streams for adaptation	32
Maximum streams per client for adaptation	160
Non-adaptive concurrency	20

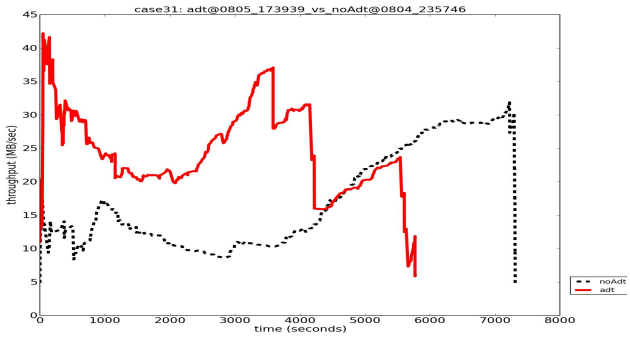

Fig. 3. Scenario 4, cumulative throughput for adaptive vs. non-adaptive transfers, max. 640 total streams, max. 160 streams per adaptive client

Fig. 3, shows the performance for this scenario, which reflects high levels of contention for available resources. It shows a significant increase in throughput achieved by adaptive transfers (red line) compared to non-adaptive (black line). The adaptive transfer of the entire data set completed approximately 20% or 1500 seconds faster than the non-adaptive. The experiment shows a significant advantage in throughput and overall transfer runtime for the adaptive, policy-based transfers compared to non-adaptive transfers on resource-constrained infrastructure.

V. Related Work

Policies for data management: In earlier work, we studied policy-based data staging for scientific applications [17, 18]. Large scientific collaborations have used policies for scientific data management. The integrated Rule-Oriented Data System (iRODS) [19] uses a rule based system to implement data management constraints. Policy-based systems for scientific data management and dissemination include the Physics Experiment Data Export (PheDEx) system [20] for high energy physics and the Lightweight Data Replicator (LDR) [21] for gravitational wave physics; these systems use VO policies for distribution and replication of data. By contrast, our focus is on using VO and site policies to control resource allocation.

Data transfer adaptation: Our work builds on earlier research on an Adaptive Data Transfer (ADT) algorithm that calculates dynamic transfer parameters such as the concurrency level from a simple throughput prediction model with information from current data transfer operations [10, 22].. This is in contrast to predictive sampling proposed in other research [23] and does not depend on active performance measurements.

VI. Conclusions

We presented two techniques to adapt the resources and parameters used for long running, multi-file data transfers: policy-based allocation of data transfer resources based on VO and site level policies and client-side adaptation of transfer parameters based on recent transfer performance. Our goal is to avoid overprovisioning of resources that results in suboptimal transfer throughput. Our results show that these techniques provide significant throughput and completion time improvements in resource constrained environments.

Acknowledgment

This work was supported in part by the National Science Foundation (NSF) Office of Cyberinfrastructure under awards 1127101 (USC/ISI) and 1127039 (LBNL) and by the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research, Office of Science, under Contract DE-AC02- 05CH11231. This work used resources from the Open Science Grid, which is supported by NSF and the DOE Office of Science. We thank Brian Bockelman, Garhan Attebury and Carl Lundstedt at U. Nebraska, Lincoln, and Iwona Sakrejda at NERSC for their support for our experiments.

References

[1] "The Large Hadron Collider," http://lhc.web.cern.ch/lhc/.

[2] "Large Synoptic Survey Telescope (LSST)," http://www.lsst.org/lsst.

[3] LIGO Project, "LIGO - Laser Interferometer Gravitational Wave Observatory," http://www.ligo.caltech.edu/.

[4] J. Cummings, T. Finholt, I. Foster, C. Kesselman, and K. A. Lawrence, "Beyond Being There: A Blueprint for Advancing the Design, Development, and Evaluation of Virtual Organizations " Final report from the NSF workshop on Building Effective Virtual Organizations, http://www.ci.uchicago.edu/events/VirtOrg2008/ 2008.

[5] D. Hasenkamp, A. Sim, M. Wehner, and K. Wu, "Finding tropical cyclones on a cloud computing cluster: Using parallel virtualization for large-scale climate simulation analysis," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 201-208.

[6] A. Sim, D. Gunter, V. Natarajan, A. Shoshani, D. Williams, J. Long, J. Hick, J. Lee, and E. Dart, "Efficient Bulk Data Replication for the Earth System Grid," in *International Symposium on Grid Computing, Data Driven e-Science: Use Cases and Successful Applications of Distributed Computing Infrastructures (ISGC 2010)*, 2010.

[7] R. Kettimuthu, A. Sim, et al., "Lessons learned from moving Earth System Grid data sets over a 20 Gbps wide-area network," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 316-319.

[8] "Open Science Grid," http://www.opensciencegrid.org/.

[9] "XSEDE Extreme Science and Engineering Discovery Environment," https://http://www.xsede.org/.

[10] A. Sim, M. Balman, D. Williams, A. Shoshani, and V. Natarajan, "Adaptive Transfer Adjustment in Efficient Bulk Data Transfer Management for Climate Datasets," in *the 22nd International Conference on Parallel and Distributed Computing and Systems (PDCS 2010)*, 2010.

[11] J. Gu, D. Smith, A. L. Chervenak, and A. Sim, "Adaptive Data Transfers that Utilize Policies for Resource Sharing," in *2nd Int'l Workshop on Network Aware Data Management (NDM 2012), in conjunction with SC12 Conference* Salt Lake City, UT, 2012.

[12] "web.py," http://webpy.org/.

[13] "CherryPy," http://cherrypy.org/.

[14] "Introducing JSON," http://json.org/.

[15] "SRM Clients," https://sdm.lbl.gov/srmclients/.

[16] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, "The Globus Striped GridFTP Framework and Server," in *SC '05 Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, Seattle, WA, 2005, p. 54.

[17] S. Bharathi and A. Chervenak, "Scheduling Data-Intensive Workflows on Storage Constrained Resources," in *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science (WORKS09), in conjunction with the SC09 Conference*, Portland, Oregon, 2009.

[18] A. L. Chervenak, D. E. Smith, W. Chen, and E. Deelman, "Integrating Policy with Scientific Workflow Management for Data-Intensive Applications," in *7th Workshop on Workflows in Support of Large-Scale Science (WORKS 12), in conjunction with SC12 Conference* Salt Lake City, UT, 2012.

[19] M. Hedges, A. Hasan, and T. Blanke, "Management and preservation of research data with iRODS," in *ACM First Workshop on CyberInfrastructure: Information Mgmt. in eScience*, 2007, pp. 17-22.

[20] J. Rehn, T. Barrass, D. Bonacorsi, J. Hernandez, I. Semeniouk, L. Tuura, and Y. Wu, "PhEDEx high-throughput data transfer management system," in *Computing in High Energy and Nuclear Physics (CHEP) 2006*, Mumbai, India, 2006.

[21] LIGO Project, "Lightweight Data Replicator, http://www.lsc-group.phys.uwm.edu/LDR/."

[22] M. Balman and T. Kosar, "Dynamic Adaptation of Parallelism Level in Data Transfer Scheduling," in *Second International Workshop on Adaptive Systems in Heterogeneous Environments*, 2009.

[23] E. Yildirim, M. Balman, and T. Kosar, "Dynamically Tuning Level of Parallelism in Wide Area Data Transfers," in *2008 Intl. workshop on Data-aware distributed computing (DADC '08)*, 2008, pp. 39-48.