Collaborative Research: SDCI Net: Policy-driven Large Scale Data Access Framework with Light-weight Performance Monitoring and Estimation (Adaptive Data Access and Policy-driven Transfers - ADAPT)

Award Numbers: 1127101 (USC/ISI), 1127039 (LBNL)

Annual Project Report

July 1, 2014

Project period of
August 1, 2013 through July 31, 2014

Principal Investigators
Ann Chervenak[1], Alex Sim[2]

Project member: Robert Schuler[1], Junmin Gu[2]
Project Website: http://sdm.lbl.gov/adapt/
Project email: adapt@hpcrd.lbl.gov

NSF Program Officer: Kevin L. Thompson

[1] University of Southern California, Information Sciences Institute
[2] Lawrence Berkeley National Laboratory

Table of Contents

1 Accomplishments - What was done? What was learned?

1.1 What are the major goals of the project?

Over the next 5 to 10 years, applications in many domains are expected to generate data at unprecedented volumes and rates. Many of these science domains include unique experimental facilities that generate large data sets, such as the Large Hadron Collider (LHC), the Large Synoptic Survey Telescope (LSST), and the Laser Interferometer Gravitational Wave Observatory (LIGO). Other domains generate large amounts of simulation data that must be shared, compared and analyzed. For science to progress, the large and steadily increasing amount of data originating from instruments and simulations must be efficiently accessed by scientists and disseminated throughout a scientific collaboration or Virtual Organization (VO) for analysis, simulation, and storage. While many large collaborations have access to high-performance networks, the growth in data volumes and demand for data movement increasingly stress networks and file transfer services. In particular, spikes in bulk data movement requests, when uncoordinated, can easily overwhelm file transfer resources, causing them to become sluggish and unresponsive, and may ultimately lead to bulk file transfer failures. A key unmet challenge for high-performance, data-instensive cyberinfrastructure is to coordinate bulk data movement to facilitate efficient transfer performance across a scientific collaboration.

The overall goal of the Adaptive Data Access and Policy-driven Transfers (ADAPT) project is to improve transfer throughput performance and latency for long-running, bulk data transfer operations in resource constrained environments with two techniques, 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, through a general purpose data access software framework.

In this project, we address the following issues with our software framework:

- Passive performance monitoring. To optimize the data accessibility, monitoring information from resources needs to be collected in a way that does not put extra loads on the resources. We collect passive monitoring information by enhancing data transfer clients to report transfer performance to a measurement archive, and to use the measurement information to generate a simple approximation of the data transfer performance.
- Adaptive data transfer. On high-performance resources, static data transfer properties can cause orders of magnitude performance degradations because of the dynamically changing shared environment. We implement adaptive transfer management methods for transfers in progress, responding to changes in the throughput performance and in the policies for their use.
- Policy-driven data transfer management. Efficient and optimal data accessibility must be based on a policy that balances user requirements for scalability and end-to-end resource performance. Policy administrators at the Virtual Organization or site level define the overall resource limits between hosts based on their resources (e.g. memory size, bus speed, storage speed) and the available network bandwidth between them. These limits are recorded and interpreted by environment-specific policy rules that determine, along with knowledge of the total resources allocated to other ongoing data transfers and their performances, what the recommended transfer properties limits should be for the client instance. We develop Policy Service to recommend these policies for data transfer management.

To address these issues in a coordinated and well-defined way, the main objective of the project is to develop and release a general-purpose, reusable and expandable framework for optimizing the performance of data movements over the shared network for scientific collaborations by supporting adaptive data transfer management, passive performance monitoring, and enforcement of site and VO policies for resource sharing. The framework is developed based on the existing tools that manage the access and replication of large scientific datasets, with additional capabilities for collecting transfer monitoring information, performing adaptive data transfer management based on both policy and performance constraints, and implementing new policies for VO level data distribution and user level data access. These software tools and framework have been validated on the testbeds before being released under open source licenses. This project enables improving efficient data access by many users and many data requests, as well as data and resource sharing within the user community.

1.2 What was accomplished under these goals?

1.2.1 Major activities

Major activities from year 3 of the project include:

- Completed implementation of the Policy Service that controls the allocation of streams to data transfer clients
- Evaluation of Policy Service
- Packaging and release of Policy Service under Apache open source license on GitHub repository
- Development and enhancements of Adaptive Data Transfer (ADT) and File Transfer Monitoring as library modules
- Enhancements to the adaptive SRM copy data transfer client by integrating ADT libraries and JSON instead of GSON for OSG open source standard
- Experimental evaluation between adaptive data transfers and non-adaptive data transfers with clustered client runs
- Packaging and release ADT libraries under BSD license
- Packaging and release of adaptive SRM copy clients under BSD license to Open Science Grid (OSG) software stack
- Presentation on ADAPT work at Supercomputing (SC13) Conference in November, 2013
- Paper and presentation at the 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2014
- Outreach to science communities in the Open Science Grid collaboration
- Outreach to iRODS (Integrated Rules-Oriented Data System) group at University of North Carolina (UNC); explored ways to integrate resource allocation policy service and adaptive data transfer client with iRODS
- Outreach to staff at Indiana University (IU) and Fermi National Accelerator Laboratory (FNAL) to explore ways to integrate ADAPT technologies with Open Science Grid Public Storage Service.
- These latter two outreach efforts led to a proposal for future work on ADAPT project with UNC, IU and FNAL.

Fig. 1 shows the components that we have developed and their interactions. They include 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 based on recent transfer performance. In this use case, multiple adaptive data transfer clients run on the source site. Each is 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.

The Policy Service (PS) is responsible for the allocation of available resources to data transfer clients that request them. The PS makes allocation decisions based on resource availability, PS parameters

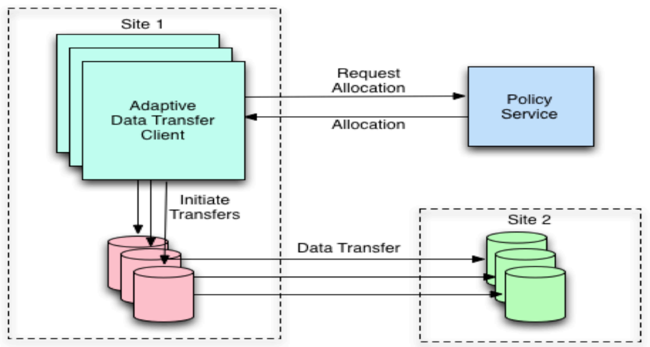


Figure 1: Service interactions for bulk remote data access

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. 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 concurrently between source and destination hosts. 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. Adaptation decisions are made by the transfer client based on the performance of recently completed 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; it then adapts to initiate more or fewer transfers. For detailed design approaches, references to the 2012[3] and 2013[4] annual reports are available.

In this third year of the project, we made enhancements to the performance and algorithms for the component services based on the general use cases in Open Science Grid (OSG). The Policy Service was re-implemented as a RESTful service written in the python language. The adaptive srm-copy data transfer client was enhanced for the OSG open source standard and for performance optimization within the resource allocation from the PS based on the recent performance measurements. We also have collected extensive measurements on the testbed to analyze and improve the effectiveness of the adaptive methods and policy rules, with clustered client runs simulating multiple users.

Dr. Sim and Dr. Chervenak presented our work during the Supercomputing 2013 in Denver, Co. (Nov. 2013). We presented our results, and discussed the possible collaboration with audiences. Our goal there was to reach out researchers and infrastructure providers about our tools and form relationships where our software can be deployed on their sites. Based on those interactions, we plan to test the latest version of our software on the shared environment where higher throughput can be achieved.

Dr. Chervenak presented a paper at the 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing in Turin, Italy in February, 2014. The goal of this submission and presentation was to increase awareness of our work internationally, to reach beyond the U.S. communities that typically attend conferences such as Supercomputing and meetings such as the Open Science Grid All Hands Meeting.

Our outreach efforts included lengthy interactions with the iRODS team at U. North Carolina/Rennaissance Computing Institute to discuss a strategy for integrating the ADAPT technologies with the iRODS data system. We also had lengthy discussions with teams from Indiana University and Fermi National Accelerator Laboratory on how to integrate ADAPT functionality into the Open Science Grid Public Storage Service; OSG users are encouraged to use the Public Storage Service to move large data sets on or off OSG storage resources. The Public Storage Service currently uses an iRODS back end, and we discussed integrating an ADAPT-enabled iRODS service as the back end for OSG Public Storage. Both the iRODS and Public Storage teams are very interested in getting the performance benefits offered by ADAPT software using adaptive data transfers and VO-level resource allocation. These discussions led to an NSF proposal submitted to the SSI call.

1.2.2 Specific Objectives - Development and enhancements in the third year of the project

We study two techniques that improve the use of available resources for large, long-running, multi-file

[3] http://sdm.lbl.gov/adapt/docs/ADAPT-Report-2012-final.pdf
[4] http://sdm.lbl.gov/adapt/docs/ADAPT-Report-2013.pdf

transfers. First, we use Virtual Organization and site policies to influence the allocation of resources such as available transfer streams to clients. Second, we show the effect of adaptation of transfer parameters for multi-file transfers, where the adaptation is based on recent performance.

1.2.2.1 Policy-Based Resource Allocation

The main operations of the PS include the following:

- Before initiating transfers, a 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. Once the transfer client receives its allocation, it uses its own algorithms to adapt within that allocation based on recent transfer performance.
- The Policy Service may receive periodic requests from clients for 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 allocation and by how much.
- After a client completes its outstanding transfers, it informs the Policy Service, which releases all resources allocated to the client and makes them available to other clients.

The Policy Service provides a framework for custom policy definitions written in a simple scripting language (python). The service is distributed with a parameterized Greedy policy for allocating resources for network data transfers, which depends 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. Fig. 2 shows the Greedy policy algorithm.

TABLE I. GREEDY POLICY PARAMETERS

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

```
Require: $s_i$: initial streams allocation specified by policy; $s_u$: update increment streams allocation specified by policy; $s_x$: maximum streams allowed between endpoints specified by policy; $s_c$: maximum streams allowed for a single client, specified by policy.
 procedure PROVISION($t$)
    $t \leftarrow$ transfer resource request with (source[$t$], dest[$t$]) and steams[$t$]
    $s_a \leftarrow$ allocated streams between (source[$t$], dest[$t$])
    $s_v \leftarrow \min(s_{cmax} - \text{streams}[t], s_{pmax} - s_a)$      // Available streams
    if $s_v = 0$ then
       // No available streams for transfer request
       return $t$
    else if streams[$t$] = 0 and $s_v > s_i$ then
       // Enough streams for initial allocation
       streams[$t$] $\leftarrow s_i$
       $s_a \leftarrow s_a + s_i$      // Update total allocated streams
    else if streams[$t$] > 0 and $s_v > s_u$ then
       // Enough streams for update allocation
       streams[$t$] $\leftarrow$ streams[$t$] + $s_u$
       $s_a \leftarrow s_a + s_u$      // Update total allocated streams
    else
       // Allocate remaining available streams to initial or update request
       streams[$t$] $\leftarrow$ streams[$t$] + $s_v$
       $s_a \leftarrow s_a + s_v$      // Update total allocated streams
    end if
    return $t$
end procedure
```

Figure 2: Resource allocation algorithm used by the Greedy Policy for an initial allocation request by a client.

The parameters in and the current implementation allow the PS to support a range of policies. For a greedy allocation strategy that allocates available streams to the first clients that request them, 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. Our experiments show the performance of long-running, multi-file transfers using both greedy and incremental allocation strategies. To support alternate resource allocation policies, users can write custom policy definitions using the python scripting language.
While the PS provides recommended resource allocations, it is up to clients to keep their utilitzation within recommended limits; the PS does not enforce resource limits at the client.

1.2.2.2 Client Side Transfer Adaptation

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

TABLE II. ADAPTIVE TRANSFER CLIENT PARAMETERS

Adaptive Transfer Client Parameter	***Definition***
Initial concurrency, c	Number of active transfers initiated by a client when it begins transferring data.
Maximum concurrency, c_{max}	Maximum number of active file transfers by a client. The client may reach this maximum value by adaptation.
Parallelism, p	Number of parallel streams per file transfer
Adaptation delay time, d	How often the client updates a resource allocation from the Policy Service; expressed as number of completed transfers before adaptation occurs.
Adaptation increment/ decrement, $\boldsymbol{\Delta}$	How much the concurrency level increases/ decreases when the client adapts up or down within its resource allocation.
Threshold, T	Difference between current and past performance that triggers adaptation of concurrency level.

```
Require: Q: queue of files to be transferred between source and dest.; c: initial client concurrency; Δ: adaptation increment/decrement delta;
d: adaptation delay; p: parallel streams per file transfer.
 procedure AdaptTransferClient(Q, c, Δ, d, p)
   t ← initialize a transfer request between (source, dest) of Q
   Provision(t)          // request initial allocation from Policy Service
   $c_{max}$ ← floor(streams[t] / p)   // convert streams to concurrency
   c ← min(c, $c_{max}$)   // limit concurrency parameter, if necessary
   k ← d                 // set counter for next adaptation
   while Q not empty do
     if $k \le 0$ then      // due for client adaptation
       k ← d                   // reset counter
       Provision(t)            // request updated allocation from PS
       $c_{max}$ ← floor(streams[t] / p)  // convert streams to concurrency
       c ← Adapt(c, Δ, $c_{max}$)   // adapt concurrency up or down
     end if
     F ← pop at most c transfer jobs from Q
     // …perform F transfers concurrently, wait for completion…
     k ← k – c           // decrement transfer counter
   end while
 end procedure

 procedure Adapt(c, Δ, $c_{max}$)
   T ← user specified transfer rate adaptation threshold
   $r_{last}$ ← state of last recorded transfer rate  // source to destination
   $r_{rate}$ ← test of current transfer rate          // source to destination
   $r_{delta}$ ← $r_{rate}$ - $r_{last}$
   if abs($r_{delta}$) > T then            //  change exceeds threshold
     if $r_{delta}$ < 0 then
       c ← max(0, c – Δ)       // decrease concurrency
     else
       c ← min(c + Δ, $c_{max}$)   // increase concurrency
     end if
   end if
   return c
end procedure
```

Figure 3: Adaptation algorithm used by Adaptive Transfer Client. The AdaptTransferClient procedure processes a batch of transfer jobs and uses the Adapt procedure to adjust concurrency up or down.

Adaptation decisions are made by the transfer client based on the performance of recently completed 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; it then adapts to initiate more or fewer transfers. This decision of whether to adapt the concurrency of client transfers is based on earlier work. 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 by more than the threshold value, the client decreases its concurrency level; if the performance difference is higher than the threshold, the client increases its concurrency for subsequent transfers to take advantage of additional available bandwidth.

A data transfer client can keep the allocation given to it by the PS until it completes all outstanding transfers; return to the PS any resources that it does not need; and periodically request additional resources from the PS. When a client receives an additional stream allocation from the PS, it may further adapt its concurrency within the new resource limits. A client may receive a large allocation of streams from the PS but still be configured to begin transfers with a low initial concurrency and adapt to consume more streams over time. Our experiments vary the initial concurrency as well as the rate of adaptation.

1.2.3 Significant results - Evaluation results

Our evaluation results are in two parts. In the first part of our evaluation, we use a testbed with constrained resources to model bulk data movement between HPC facilities with contention for available resources. For these experiments, a client node at Lawrence Berkeley National Laboratory (LBNL) transfers data to the University of Nebraska at Lincoln (UNL). These experiments show the advantage of our techniques in realistic resource constrained environments. In the second part of the evaluation, we illustrate the operation and performance of client side adaptation and policy-based resource allocation techniques on long-running data transfers over a relatively high-performance network between the Parallel Distributed Systems Facility at the National Energy Research Scientific Computing Center (NERSC) in Oakland, CA and UNL. This second set of results shows the tradeoffs and capabilities of adaptive transfers and policy-based resource allocation.

1.2.3.1 Experimental use case on resource-constrained environment

To demonstrate the effects of a resource-constrained environment on transfer performance, we utilized a single node at LBNL on which we ran 8 clients, allowing them to compete for resources. The node's configuration included a Quad CPU, 8GB DDR2 memory, 4TB RAID array, and a Gigabit network interface, running the latest Ubuntu OS. At UNL, we used a frontend BeStMan storage manager[5] and 9 GridFTP frontend storage nodes with backend HDFS storage with a 10 Gbps connection to the wide area network. The network between LBNL and UNL crosses the ESnet[6] and Internet2[7] domains. The network and resources at both ends are shared with other traffic. The 8 clients at LBNL transfer a 260 GByte data set made up of 488 files. Each client transfers 60 or 62 files.

In the following experiments, 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 four experimental scenarios, in which we increased the total number of streams and the maximum streams per client for each scenario to increase the load on the infrastructure. For each

[5] http://sdm.lbl.gov/bestman/

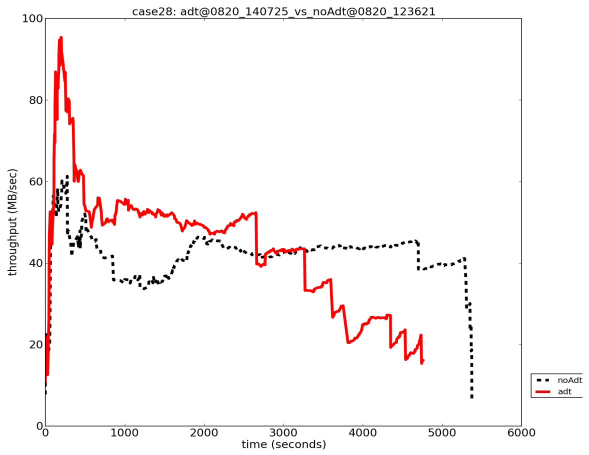
[6] http://www.es.net

[7] http://www.internet2.edu

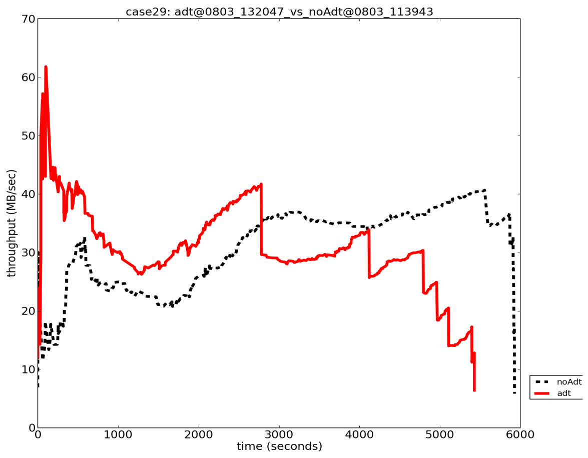
scenario, we ran the experiments three times; each run produced similar performance. Below, we show one representative graph for each scenario. In the Fig. 4, the red line shows the aggregate throughput in MBytes/second of adaptive, policy-based clients; the black line shows the throughput of non-adaptive clients.

TABLE III. PARAMETERS FOR COMPARATIVE EXPERIMENTS

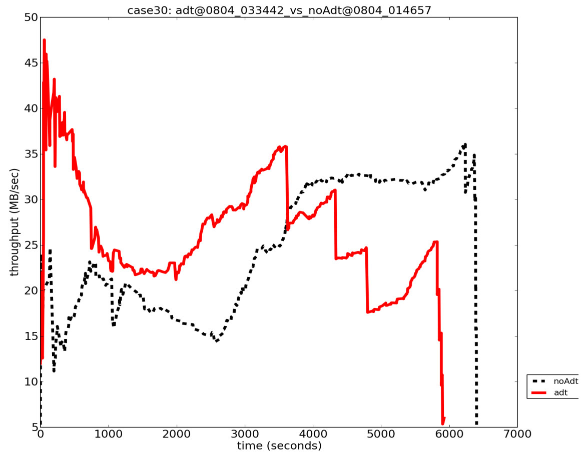
Parameters for all Comparative Experiments	***Value***
Maximum total streams between source/destination (scenarios 1 through 4)	256, 384, 512, 640
Number of clients	8
Parallelism: streams per file transfer	4
Adaptation increment/decrement	2 concurrency (8 streams)
Initial Streams for adaptation (scenarios 1 through 4)	16, 16, 16, 32
Maximum streams per client for adaptation (scenarios 1 through 4)	64, 96, 128, 160
Non-adaptive concurrency (scenarios 1 through 4)	8, 12, 16, 20



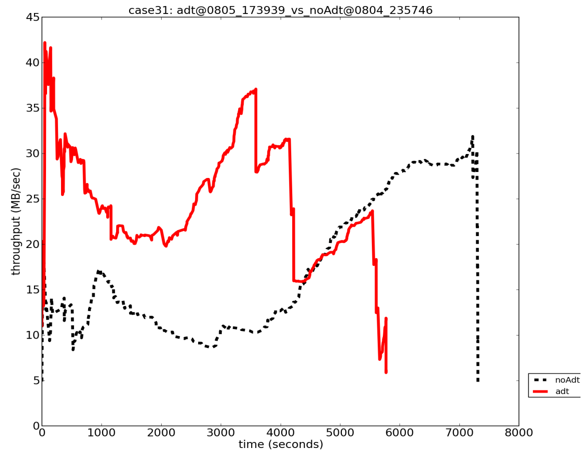
(a)



(b)



(c)



(d)

Figure 4: (a) Scenario 1, aggregate throughput for adaptive vs. non-adaptive transfers, max. 256 total streams, max. 64 streams per adaptive client, (b) Scenario 2, aggregate throughput for adaptive vs. non-adaptive transfers, max. 384 total streams, max. 96 streams per adaptive client, (c) Scenario 3, aggregate throughput for adaptive vs. non-adaptive transfers, max. 512 total streams, max. 128 streams per adaptive client, (d) Scenario 4, aggregate throughput for adaptive vs. non-adaptive transfers, max. 640 total streams, max. 160 streams per adaptive client.

In Fig. 4-a, we show an experimental run for the first scenario, with 256 total streams and a maximum of 64 streams per client. The non-adaptive experiments used 8 clients with a concurrency of 8 simultaneous transfers, each with parallelism of 4 streams per transfer, for a total of 32 streams per client and 265 total streams. For the first half of the experiment, adaptive transfers (red line) have greater aggregate throughput than the non-adaptive transfers (black line). The adaptive transfers finish about 500 seconds (over 8 minutes) faster.

In Fig. 4-b, we show an experimental run for scenario 2 with 384 total streams and maximum streams per client of 96. The non-adaptive experiments used 8 clients with a concurrency of 12 and parallelism of 4. Compared to the previous figure, there is a larger vertical separation between the adaptive throughput (red line) and the non-adaptive throughput (black line) for the first half of the experiment. The adaptive transfers still finish approximately 500 seconds earlier than non-adaptive transfers.

In the third scenario (Fig. 4-c), the total number of streams increased to 512 and the maximum streams per client were 128. The non-adaptive experiments used 8 clients, each with 16 concurrent transfers and parallelism of 4. The aggregate throughput of the adaptive transfers continues to increase compared to the non-adaptive transfers as the contention for resources on the infrastructure increases.

In the fourth scenario (Fig. 4-d), which reflects the highest contention for available resources of these scenarios, the total number of streams was 640, with a maximum of 160 streams per client. Non-adaptive concurrency increased to 20. Fig. 16 shows much higher throughput achieved by adaptive transfers compared to non-adaptive (i.e., the larger vertical separation between the red and black lines). The adaptive transfers completed approximately 1500 seconds (25 minutes) faster.

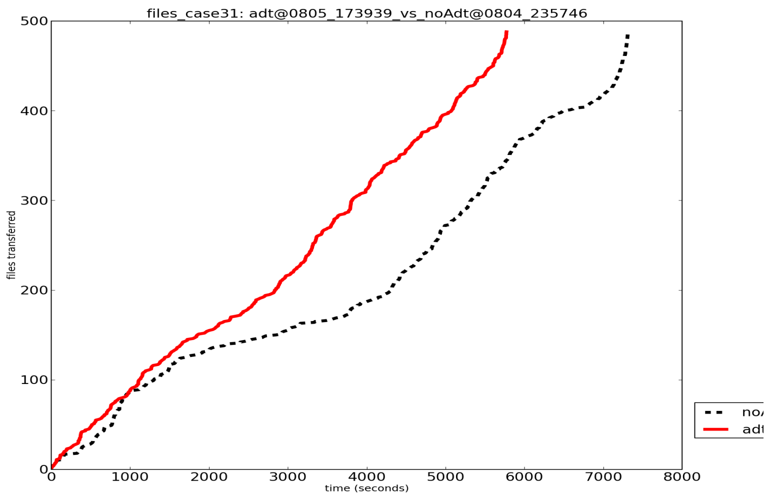


Figure 5: Scenario 4: Shows number of files transferred (vertical axis) vs. time (seconds, horizontal axis) for adaptive and non-adaptive transfers.

Finally, Fig. 5 presents cumulative file transfer completions for the pervious experiment (Scenario 4 in Fig. 4-d). File transfers complete significantly faster for adaptive vs. non-adaptive transfers, which benefits applications that can make progress as soon as needed data are available.

These experiments show a significant advantage in throughput and overall transfer time for adaptive, policy-based transfers compared to non-adaptive transfers on resource-constrained infrastructure. In the most resource constrained experiment (Figs. 4-d and 5), the total transfer time is reduced by approximately 20%.

1.2.3.2 Experimental use case for policy analysis

We also ran experiments to investigate the tradeoffs of a range of stream allocation policies and faster vs. slower client-side adaptation. We transferred the same 260 Gbyte data set from NERSC to UNL. Our experiments used 8 job submission nodes at the source site at NERSC, each of which runs SL6 with GPFS backend storage. Both NERSC and UNL have 10 Gbps connections to the wide area network, which crosses the ESnet and Internet2 domains. The network and resources at both ends are shared with other traffic, which causes performance variations in our results. Fig. 6 shows this setup.

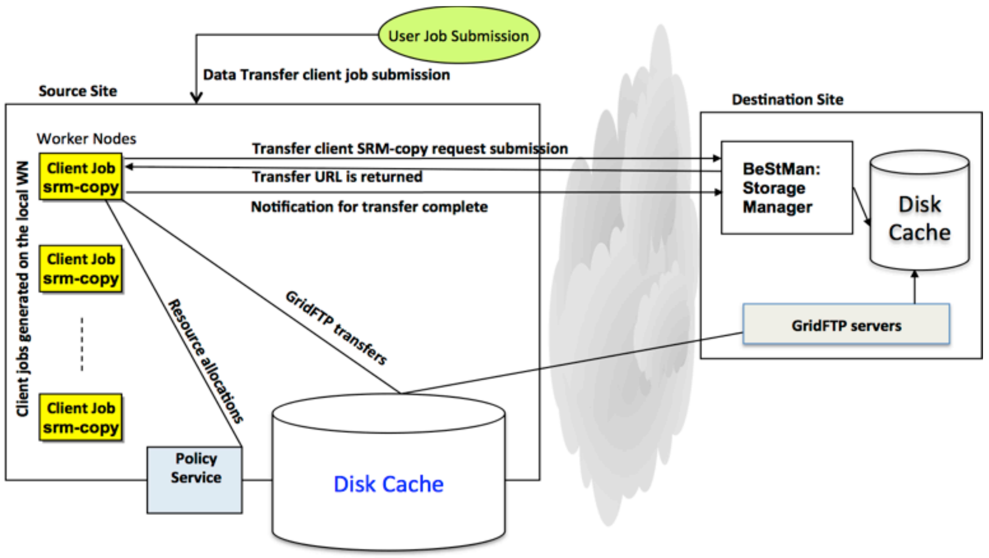


Figure 6: Experimental testbed setup

Table IV shows common parameters for the following experiments that use adaptation or policy-based allocation.

TABLE IV. COMMON PARAMETERS USED FOR ADAPTIVE EXPERIMENTS

Common Parameters for all Adaptive Experiments	*Value*
Maximum total streams between source/destination	128
Number of clients	8
Maximum streams per client	32
Parallel streams per file	4
Adaptation increment/decrement	1 concurrency (4 streams)

1.2.3.2.1 Slow Client Side Adaptation

First, we isolate the effect of slow client side adaptation. Parameters for this experiment are shown in Table V. In this experiment, the PS has no role beyond its initial allocation to each client. All adaptation takes place on the client side.

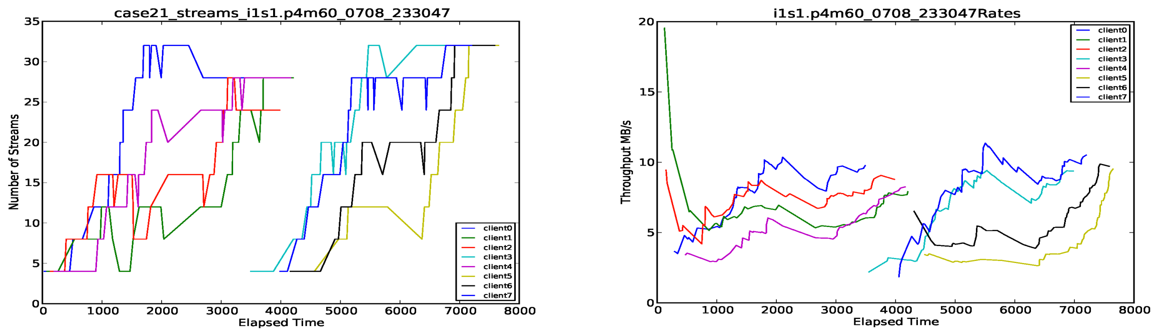
Fig. 7 show the performance for one of the three runs for this experiment. Fig. 7-a shows the number of streams being used by each client on the vertical axis, with the horizontal axis showing elapsed time. Based on the parameters in Table V, the first four clients that consulted the Policy Service were allocated 32 streams out of the 128 total streams available between the source and destination; the remaining 4 clients had to wait until one or more of those clients completed their transfers and freed streams for the remaining clients.

TABLE V. PARAMETERS USED FOR SLOW CLIENT SIDE ADAPTATION

Client Parameters	*Value*
Initial concurrency	1

Client Parameters	***Value***
Maximum concurrency	8
Adaptation delay time (update after how many transfers)	4
Policy Service Parameters	***Value***
Initial stream allocation	32
Update allocation increment	N/A

Within the 32 allocated streams, each client slowly adapts the concurrency of its transfers, beginning with one transfer that uses 4 parallel streams, and increasing to a maximum of 8 concurrent transfers that each use 4 streams, or a maximum of 32 streams per client. Each client is configured to adapt its concurrency after 4 transfers complete; it may then increase or decresase its concurrency by one transfer (4 streams) based on recent performance. Fig. 7-a shows each client slowly adapting its concurrency up or down, with some clients eventually reaching the maximum of 8 transfers (32 streams). Once a client completes its transfers, the PS frees up the client's stream allocation. The PS then allocates 32 streams to one of the waiting clients. That client then begins data its transfers with a concurrency of one and adapts based on performance.



(a) (b)

Figure 7: (a) Number of streams used for slow client side adaptation, (b) Throughput for slow client side adaptation

Fig. 7-b shows the corresponding throughput graph for this experiment. The first clients initially receive higher bandwidth, but once four clients are active, throughput per client drops. As the individual clients increase their concurrency, the bandwidth increases to 5 to 10 MB/sec for each client. Bandwidth drops again when the second group of four clients starts to transfer at low concurrency and then adapts to use higher bandwidth.

We ran these experiments three times on different days and times of the day. We observed a range of performance based on the load on the infrastructure, as expected when using shared infrastructure at the source and destination sites and shared networking betweeen sites.

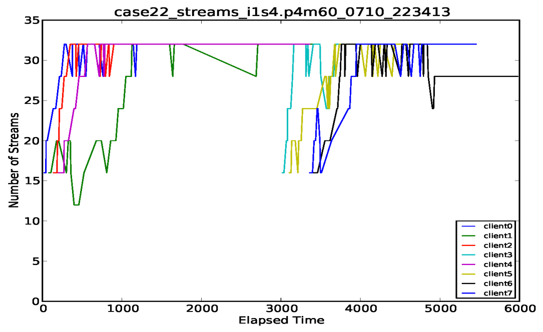
1.2.3.2.2 Fast Client Side Adaptation

In the next set of experiments, we measured faster client side adaptation, where the client increases or decreases its concurrency by one transfer after every 2 completed transfers. In this scenario, the Policy Service again allocates 32 streams per client when it first gets a request for a client allocation, and then it has no further role in the adaptation. The client begins with a concurrency of 4 transfers (16 streams). The experimental parameters are summarized in Table VI.

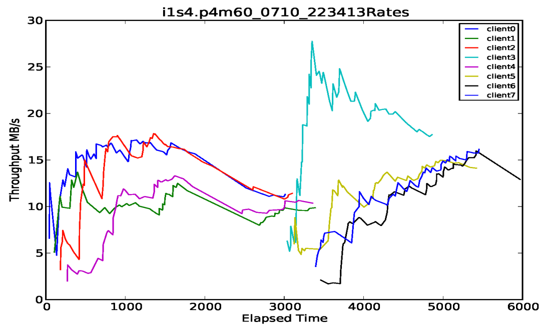
TABLE VI. PARAMETERS USED FOR FAST CLIENT SIDE ADAPTATION

Client Parameters	***Value***
Initial concurrency	4
Maximum concurrency	8
Adaptation delay time (update after how many transfers)	2
Policy Service Parameters	***Value***
Initial stream allocation	32

Client Parameters	***Value***
Update allocation increment	N/A



(a)



(b)

Figure 8: (a) Number of streams used for fast client side adaptation, (b) Throughput for fast client side adaptation

Fig. 8 shows the streams and throughput used during one run of these experiments. Fig. 8-a shows that each client begins with a concurrency of 4 transfers (16 streams) and quickly adapts up or down by concurrency of 1 each time two transfers complete. Fig. 8-b shows that the throughput achieved when using this initial concurrency level of 4 transfers and adapting quickly ranges from 5 to 25 MB/sec per client. Thus, fast client side adaptation can quickly scale aggregate throughput to consume available resources. Because of the large variations in load on our shared infrastructure, it is challenging to do direct comparisons of experiments (e.g., slow vs. fast client adaptation).

1.2.3.2.3 Policy Service Resource Allocation: Slow Increases

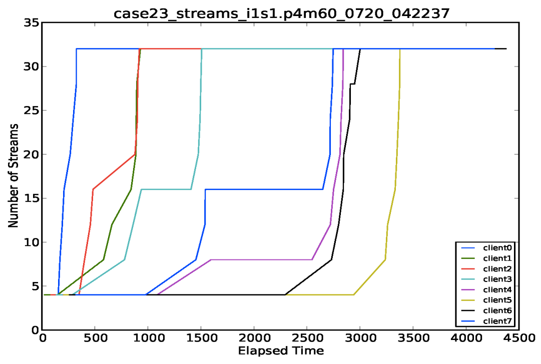
Next, we isolated the effect of the Policy Service (PS), which provides an allocation of streams to each client. In this experiment and the next, the data transfer client does no performance-based adaptation of the number of streams. After it sends an initial or update request for an allocation to the PS, the transfer client simply sets its concurrency level based on the allocation it receives. The current implementation of the PS only increases the allocation to each client if additional resources are available; it does not decrease the allocation, but instead waits for the data transfer client to return streams if they are no longer needed. In future work, we will modify the PS to decrease allocations based on transfer performance or on site or VO policies and to handle exceptional situations such as non-responsive clients.

Experimental parameters for slow increases in resource allocation by the PS are summarized in Table VII. Fig. 9-a shows the streams used by clients for this experiment. The PS initially allocates 4 streams to each client. A client requests an updated allocation after 4 transfers complete; the PS then allocates 4 additional streams if they are available. When a client receives an allocation from the PS, it initiates transfers at the maximum concurrency allowed by that allocation (up to a concurrency of 8 for this experiment). The clients do not adapt based on performance.

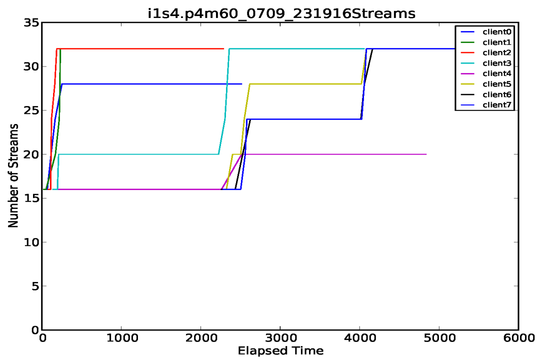
TABLE VII. PARMETERS USED FOR SLOW INCREASES IN PS ALLOCATION

Client Parameters	***Value***
Initial concurrency	1
Maximum concurrency	8
Adaptation delay time (update after how many transfers)	4
Policy Service Parameters	***Value***
Initial stream allocation	4
Update allocation increment	4

We omit the corresponding throughput graph for space reasons; it shows that the slow increases in resources allocated by the PS are reflected in the throughput achieved. Up to 8 clients transfer data concurrently, with each transferring at 1 to 8 MB/sec; higher throughputs are reached after clients update their allocations and increase their concurrency several times.



(a)



(b)

Figure 9: (a) Stream allocation for slow increases in policy service allocation, (b) Stream allocation for fast increases in policy service allocation

1.2.3.2.4 Policy Service Resource Allocation: Fast Increases

In the next experiment, we again isolate the effect of increasing allocations by the PS, this time using faster increases in those allocations. When a new request arrives from a data transfer client to the PS, the PS allocates 16 streams (concurrency level of 4) to the client. Each time a client completes two transfers, it requests an updated allocation from the PS. When additional resources are available, the PS provides 4 additional streams, allowing the client to increase its concurrency by 1. As in the last experiment, the client does no performance-based adaptation. It only updates its concurrency level to use the allocated streams provided by the PS. These parameters are summarized in Table VIII.

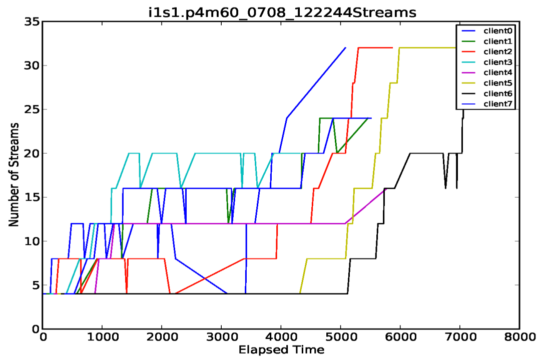
TABLE VIII. PARMETERS USED FOR FAST INCREASES IN PS ALLOCATION

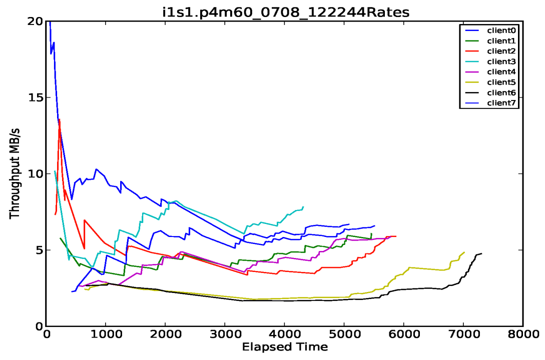
Client Parameters	***Value***
Initial concurrency	4
Maximum concurrency	8
Adaptation delay time (update after how many transfers)	2
Policy Service Parameters	***Value***
Initial stream allocation	16
Update allocation increment (streams)	4

Fig. 9-b shows the stream usage for one run of this experiment. The initial clients that consult the PS receive their allocation of 16 streams and begin transferring data. Before the remaining clients can all be given a stream allocation, the first clients request updated allocations from the PS. The graph shows that several clients adapt up to 20, 28 and 32 streams, respectively, forcing the last three clients to wait until those first clients complete their transfers and release the streams. The corresponding throughput graph (omitted for space reasons) shows that the first five clients adapt quickly to increase their bandwidth (up to 5 to 18 MB/sec per client), while the remaining three clients wait for available streams.

1.2.3.2.5 Combined Effect: Slow Client Side Adaptation and Slow Increases in Policy Service Allocation

Our earlier experiments isolated the effect of client side adaptation or increases in resource allocation by the PS. Next, we combine these techniques to measure their collective effect. First, we look at the combined effect of slow client side adaptation and slow increases in allocations by the PS. In this experiment, the PS provides an initial allocation of 4 streams to each client; on an update request, the PS provides 4 additional streams if they are available. Each data transfer client starts with an initial concurrency of one (4 streams) and updates its concurrency by up to one transer (4 streams) when 4 transfers complete. The parameters are the same as those shown in Table VII; the difference for this experiment is that the transfer client is adapting based on recent performance and not merely using the allocation provided by the PS.





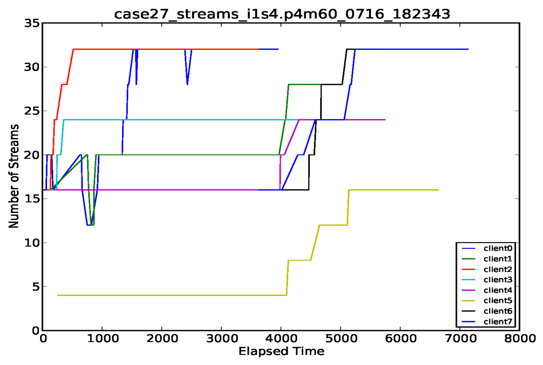
(a) (b)

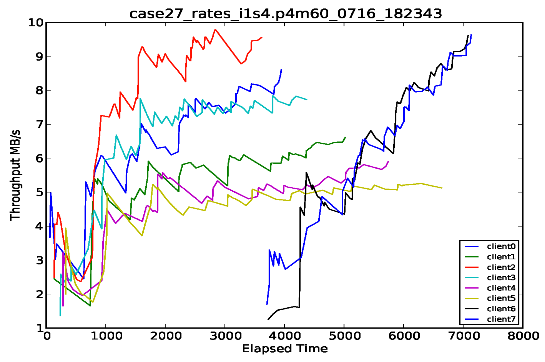
Figure 10: (a) Stream used for combined slow PS allocation, client adaptation, (b) Throughput for combined slow PS allocation, client adaptation

Fig. 10-a shows the streams consumed by the 8 clients in one iteration of this experiment. As expected, each client starts out with 4 streams and slowly adapts toward a maximum of 32 streams per client. The corresponding throughput graph is shown in Fig. 10-b. The throughput achieved for these transfers is 2 to 10 MB/second per client after all 8 clients are active.

1.2.3.2.6 Combined Effect: Fast Client Side Adaptation and Fast Increases in Policy Service Allocation

Next, we ran experiments that combined fast adaptation by the data transfer client with fast increases in stream allocation by the PS. The PS initially allocates 16 streams to a client if those streams are available; for an update request, the PS allocates 4 additional streams (if available). The parameters are the same as those in Table VIII. The client decides whether to adjust its concurrency up or down by one transfer (4 streams) after 2 transfers complete. Results for one experimental run are shown in Fig. 11. Fig. 11-a shows that five clients start with 16 streams and adapt quickly towards 32 streams. One client receives an initial allocation of 4 streams because all other streams have been allocated. Two clients wait for earlier clients to finish. Fig. 11-b shows client throughput. This experiment shows that fast adaptation and fast allocation increases quickly scale to consume available resources.





(a) (b)

Figure 11: (a) Stream used for combined fast PS alloction, client adaptation, (b) Throughput for combined fast PS alloction, client adaptation

1.2.3.3 Summary of experimental results

The experiments from section 2.2.3.2 illustrate the operation and tradeoffs of client side adaptation and policy-based stream allocation techniques used separately and in combination. The results show that the use of fast client-side adaptation allows transfer clients to quickly saturate high bandwidth networks (perhaps with few competing clients) without overprovisioning resources, while slow adaptation is better suited to scenarios with network contention where the goal is to share bandwidth more fairly among clients. For the Policy Service, the results demonstrate that granting larger allocations with a first-come-first-served strategy can significantly increase throughput for the earliest requesters. This approach may inform the design of VO policies that seek to minimize the makespan for early requesters while delaying (effectively queueing) the start time for later requesters. Conversely, VO policies that allocate fewer resources may be better suited to minimize overall makespan (for all requesters), which may be especially beneficial where jobs can be parallelized and start when a subset of input files is available. These results can thus be used to inform the specification of VO policies and for tailoring the resource allocations to the needs of VO clients and their transfer resources.

1.2.4 Key outcomes or other achievements

During the third year of the project, we have developed significant enhancements to the component services for performance and algorithms based on the general use cases in Open Science Grid (OSG). We generalized and deployed the policy logic as Policy Service. We significantly enhanced the adaptive SRM copy client with the Adaptive Data Transfer (ADT) library module.

We also have collected extensive experimental measurements to analyze and improve the effectiveness of the adaptive methods and policy rules.

Dr. Sim and Dr. Chervenak presented our work during the Supercomputing 2013 in Denver, Co. (Nov. 2013).

We published our results in the 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing, in February 2014. Dr. Chrevenak presented the work at the conference.

We conducted outreach to important science communities with the goal of technology transfer. We developed a plan for integration of ADAPT technologies with the iRODS Integrated Rule Oriented Data System project as well as the Open Science Grid Public Storage Service, which OSG users are encouraged to use to transfer large data sets to/from OSG storage resources. These outreach activities led to an NSF proposal that would fund future work on ADAPT technologies and technology transfer.

This project has contributed to improving the data accessibility in a large scientific collaboration, providing efficiency in data movement and enforcement of policies for data access and resource sharing, especially on the resource-constraint shared environment.

Summary of Experiments: Our experiments illustrate the operation and tradeoffs of client side adaptation and policy-based stream allocation techniques used separately and in combination. The results show that the use of fast client-side adaptation allows transfer clients to quickly saturate high bandwidth networks (perhaps with few competing clients) without overprovisioning resources, while slow adaptation is better suited to scenarios with network contention where the goal is to share bandwidth more fairly among clients. For the Policy Service, the results demonstrate that granting larger allocations with a first-come-first-served strategy can significantly increase throughput for the earliest requesters. This approach may inform the design of VO policies that seek to minimize the makespan for early requesters while delaying (effectively queueing) the start time for later requesters. Conversely, VO policies that allocate fewer resources may be better suited to minimize overall makespan (for all requesters), which may be especially beneficial where jobs can be parallelized and start when a subset of input files is available. These results can thus be used to inform the specification of VO policies and for tailoring the resource allocations to the needs of VO clients and their transfer resources.

1.3 What opportunities for training and professional development has the project provided?

In its third year, the project provided professional development for two programmers who have participated in both development and research efforts. Robert Schuler at USC/ISI developed the policy logic, participated in evaluation experiments, packaged the software for release to the community, and supervised graduate student Nandan Hirpathak. Junmin Gu, the developer at LBNL was responsible for the enhancements to the adaptive data movement client, evaluation experiments and packaging of the adaptive transfer libraries and modified data transfer client. This project has provided rich challenges for these developers, advancing their professional skills and development.

In addition, Nandan Hirpathak, a graduate student at USC, did directed research for summer and fall of 2013 on the project. This work provided his first exposure to graduate level research activities. He was involved in management of the large amounts of measurement data produced by our experiments.

1.4 How have the results been disseminated to communities of interest?

Dr. Sim and Dr. Chervenak presented our work during the Supercomputing 2013 in Denver, Co. (Nov. 2013). We presented our results, and discussed the possible collaboration with audiences. Our goal there was to reach out researchers and infrastructure providers about our tools and form relationships where our software can be deployed on their sites. Based on those interactions, we plan to test the latest version of our software on the shared environment where higher throughput can be achieved.

Dr. Chervenak presented a paper at the 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing in Turin, Italy in February, 2014. The goal of this submission and presentation was to increase awareness of our work internationally, to reach beyond the U.S. communities that typically attend conferences such as Supercomputing and meetings such as the Open Science Grid All Hands Meeting.

1.5 What do you plan to do during the next reporting period to accomplish the goals?

We requested a 6-month extension, but the remaining funds for the project are modest. For this period, we will focus primarily on outreach activities and technology transfer.

1.5.1 Outreach, working with communities

- During the extended period of the project, one of our highest priorities will be technology transfer and identifying communities that will benefit from the ADAPT technologies. We will continue discussions on the integration of our software framework with the Open Science Grid Public Storage Service and the iRODS service. We will present our most recent results at the Supercomputing conference through presentations and submission of a workshop paper. At the SC conference, we will reach out other communities and work for the goal of increasing our work's visibility in the international community.

1.5.2 Evaluation, software releases and test runs

- As funds allow, we will continue testing the functionality, performance and scalability of the ADAPT software.
- We will focus on evaluation and releasing of the source code under open-source licenses to the community.

2 Products - What has the project produced?

2.1 Publication

- "*Efficient Data Staging Using Performance-Based Adaptation and Policy-Based Resource Allocation*", A. L. Chervenak, A. Sim, J. Gu, R. Schuler, N. Hirpathak, The 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2014.

2.2 Presentation

- "*ADAPT: Improving Data Transfers Using Performance-Based Adaptation and Policy-Based Resource Allocation*", Presentation in SC'13, Nov 2013.

2.3 Websites

- Main ADAPT Project Website:
  http://sdm.lbl.gov/adapt/
- Software:
  Adaptive Data Transfer (ADT) client: https://codeforge.lbl.gov/projects/adapt/
  Policy Service: http://github.com/robes/adapt-policy-service

2.4 Other products (Software)

- Adaptive Data Transfer (ADT) library under BSD open source license:
  http://codeforge.lbl.gov/projects/adapt/
- Adaptive srm-copy under BSD open source license:
  http://codeforge.lbl.gov/projects/bestman/, http://codeforge.lbl.gov/projects/adapt/
- Policy Service under Apache open source license:
  http://github.com/robes/adapt-policy-service

2.5 Data archives (logs)

- https://sdm.lbl.gov/adapt/docs/adapt-runs-201306.tar.gz
- https://sdm.lbl.gov/adapt/docs/adapt-runs-201307.tar.gz

3 Participants & Other Collaborating Organizations - Who has been involved?

3.1 What individuals have worked on the project?

Name	Most Project Senior Role	Nearest Person Month Worked
Ann Chervenak	PD/PI	5
Robert Schuler	Developer	3
Junmin Gu	Developer	6
Alex Sim	Co-PI	1
Nandan Hiparthak	Graduate student	2

3.2 What other organizations have been involved as partners?

University of Southern California Information Sciences Institute
Lawrence Berkeley National Laboratory

3.3 Have other collaborators or contacts been involved?

No.

4 Impact - What is the impact of the project? How has it contributed?

4.1 What is the impact on the development of the principal discipline of the project?

This project is focused on computer systems and networking, with the goal of improving transfer throughput and latency for bulk data transfer operations between sites. This use case is typical of the scientific workloads that run on the Open Science Grid and other national cyberinfrasturcture. The project has received increased visibility in the last year, with interest from participants in the Supercomputing 2013 conference and from attendees at the Euromicro PDP2014 conference based on our presentations there. We have seen our paper from the NDM 2012 Workshop appearing in the reference lists of papers by submitted by colleagues on other projects. We are encouraged that the ideas that we are exploring in this project are being followed by key players in the larger networking and systems community.

The third year of the project has included increasing outreach to key communities. Our outreach to the IRODS (integrated Rule Oriented Data System) at University of North Carolina's Renaissance Computing Instiute (RENCI) have led to discussions about integrating the performance benefits of ADAPT software into iRODS. Discussions with researchers at Indiana University and Fermi National Accelerator Laboratory have focused on integrating this functionality into the Open Science Grid Public Storage Service, which OSG users are encouraged to use when reading/writing large data to OSG storage resources. These outreach efforts led to a joint proposal to the NSF SSI call that would fund these integration efforts with iRODS and the OSG Public Storage Service.

In the last months of the project, we will continue our outreach and software support efforts.

4.2 What is the impact on other disciplines?

The impact on other disciplines is indirect. By improving data transfer performance and the use of resources in constrained environments, we will enable a range of scientific applications and workflows to run more efficiently on leadership class facilities, national infrastructure such as Open Science Grid, and Clouds. This in turn will lead to improved science results for applications that run faster and/or at larger scales than before.

4.3 What is the impact on the development of human resources?

Several developers and student researchers have worked on the ADAPT project, gaining valuable experience in algorithm design, implementation, performance evaluation and analysis, and using those analyses to drive further improvements in project software. This project provides exposure to interesting research and development questions regarding efficient use of available resources shared among competing workloads and the use of performance-based adaptation in data transfers. The work has led to two publications to date, with more in progress, that included participation from developers and student researchers.

4.4 What is the impact on physical resources that form infrastructure?

The goal of this software is to improve the utilization of existing national scale infrastructure, such as Open Science Grid and XSEDE, by maximizing the throughput of data transfers on available infrastructure and reducing the time to transfer large data sets that are needed for computations running on those resources. The software itself is designed to be lightweight and to place minimal burden on the infrastructure, while improving the performance of data transfers already running on that infrastructure.

4.5 What is the impact on institutional resources that form infrastructure?

The proposed research should result in improved utilization of institutional infrastructure, such as the resources that each institution contributes to national cyberinfrastructure such as Open Science Grid and XSEDE. In particular, the Virtual Organization-level resource allocation strategies that we have developed are designed to take a broad view of ongoing activities at the infrastructure and virtual organization level and to allocate resources for data transfers based on knowledge of ongoing activities across one or more VOs and the priorities and preferences set by VO and site administrators.

4.6 What is the impact on information resources that form infrastructure?

Nothing to report

4.7 What is the impact on technology transfer

We have begun discussions on technology transfer with the IRODS (integrated Rule Oriented Data System) project at University of North Carolina's Renaissance Computing Instituce (RENCI) and with researchers at Indiana University and Fermi National Accelerator Laboratory who have developed and deployed the Open Science Grid Public Storage Service. Both groups would like to gain the performance benefits offered by ADAPT technology in their existing systems. These discussions led to a joint proposal to the NSF SSI call that would fund these integration efforts with iRODS and the OSG Public Storage Service, effectively transferring the technology to widely used data services.

In addition, the adaptive srm-copy client developed in the ADAPT project has been included in the most recent OSG software stack, allowing clients to optionally use the adaptive capabilities during transfer.

4.8 What is the impact on society beyond science and technology?

Nothing to report

5 Changes and Problems

5.1 Changes in approach and reasons for change

Nothing to report.

5.2 Actual or Anticipated problems or delays and actions or plans to resolve them

Nothing to report.

5.3 Changes that have significant impact on expenditures

Nothing to report.

5.4 Significant changes in use or care of human subjects

Nothing to report.

5.5 Significant changes in use or care of vertebrate animals

Nothing to report.

5.6 Significant changes in use or care of biohazards

Nothing to report.

6 Special Requirements

Nothing to report.