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, 2012

Project period of
August 1, 2011 through July 31, 2012

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

Project member: David Smith[1], Craig Ward[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

Abstract

Large-scale science applications are expected to generate exabytes of data over the next 5 to 10 years. With scientific data collected at unprecedented volumes and rates, the success of large scientific collaborations requires that they provide distributed data access with improved data access latencies and increased reliability to a large user community. To meet these requirements, scientific collaborations are increasingly replicating large datasets over high-speed networks to multiple sites.

The main objective of this project is to develop and deploy a general-purpose data access framework for scientific collaborations that provides fine-grained and adaptive data transfer management, lightweight passive performance monitoring, and enforcement of site and VO policies for resource sharing. Lightweight monitoring mechanisms collect monitoring information from data movement tools without putting extra loads on the shared resources. Data transfer management mechanisms select transfer properties based on each transfer's performance estimation, and adapt those properties when observed performance changes due to the dynamic load on storage, network and other resources. Finally, policy-driven resource management using Virtual Organization policies, regarding replication and resource allocation balances user requirements for data freshness with the load on resources.

Intellectual merit: The project will produce a software framework that will improve the ability of distributed scientific collaborations to provide efficient access to replicated datasets by a large community of users; this framework will combine fine-grained transfer management, light-weight monitoring, and transfer advice from policy- driven resource management.

Broader impact: The development will facilitate scientific advances in many domains that increasingly depend on replication and sharing of growing datasets.

This document gives an overview of the technical progress in the first year of the ADAPT project. The report covers the period from August 1, 2011 to July 31, 2012 and completes the 2012 annual project report requirement for the ADAPT project.

1 Technical background

Scientific data are now collected at unprecedented volumes and rates and need to be shared by increasing numbers of geographically distributed collaborators connected by high performance networks. The need to share massive amounts of scientific data can be found in almost every scientific domain, including high energy and nuclear physics, astrophysics, climate modeling, nanoscale materials science, and genomics. Over the next 5-10 years, applications in these domains are expected to generate exabytes of data. For the science to progress, the large and steadily increasing amount of data originating from these facilities must be continuously moved from the experimental facilities to scientists and to analysis, simulation, and storage facilities.

We plan to facilitate efficient movement of these large datasets across high performance science networks by implementing capabilities for lightweight, passive monitoring of end-to-end network performance for data transfers and by using this monitoring information to estimate future transfer performance. Large science collaborations typically support a large number of replication and download requests to share datasets among tens of thousands users in the community. When monitoring information is collected passively from data movement clients, the performance from the source to destination can be approximated from the historical information without putting extra loads on the resources by active measurement collection. Using such estimates, we will extend client tools to automatically set appropriate parameters for data transfers. These transfer parameters will also be influenced by policies specified by sites or at the Virtual Organization (VO) level regarding how network bandwidth may be shared among participants in a VO.

This adaptive data transfer method will improve efficient data access by many users and many data requests, as well as data and resource sharing within the user community.

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

- End-to-end light-weight performance monitoring. To optimize the accessibility of datasets, monitoring information from resources needs to be collected using light-weight mechanisms that do not put extra loads on the resources. We will collect passive monitoring information by enhancing data transfer clients to report transfer performance to a perfSONAR-based measurement archive, and to use this historical monitoring 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 will 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. When datasets are updated on the publishing site, these updates must be propagated efficiently to replicas of those datasets at mirror sites using an overall policy that balances user requirements for data freshness with the load on storage, network and other resources. Replicated datasets must be selected for efficient and optimal accessibility to the network and storage resources using a policy that balances user requirements for scalability and performance based on the end-to-end performance estimation. We propose to extend an existing Policy Service to enforce these policies for data management and resource selection.

To address these issues in a coordinated and well-defined way, the main objective of the project is to develop and deploy a general-purpose, reusable and expandable framework for optimizing the performance of data movements over the network for scientific collaborations by supporting light-weight performance monitoring and estimation, adaptive data transfer management and enforcement of site and VO policies for resource sharing. The framework will be 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 replication and user level replica and resource selection. These tools will be validated on emulation and distributed testbeds before

being deployed in large scale scientific collaborations. In addition, the team will release the software framework to the community under open source licenses.

1.1 Approach

Our project work includes three components, which we call the Policy Service (PS), Data Movement Service (DMS) and Passive Measurement Archive (PMA). This work will build on several existing components: a Policy Driven Data Management Service, several data movement tools including the Bulk Data Mover and SRM client services, and the perfSONAR for measurement data archive.

The Passive Measurement Archive (PMA) collects monitoring data passively from the Data Movement Services (DMS), and provides the historical measurements based on the existing perfSONAR archiving services; the Policy Service (PS) then makes a simple approximation of the end-to-end performance of transfers from the historical measurement information. The PS makes replication and resource selection decisions among the sites of collaboration, enforcing policies that describe data dissemination patterns and resource priorities and preferences specified by the collaboration. The Data Movement Service (DMS) uses information from the PS to manage data transfers and adaptively adjust data transfer properties such as parallelism, concurrency and buffer size.

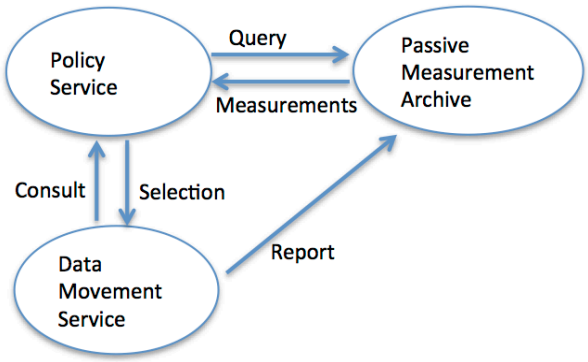

Figure 1: interaction diagram between components

The relationship between these components is shown Figure 1. In response to a request from the DMS, the PS will use its policies as well as historical measurements from the PMA to make a simple approximation, generate advice about transfer parameters and make resource selection decisions. The PS returns this advice to the DMS, which manages transfers and adjusts transfer parameters adaptively to accommodate changes in network and storage performance. The DMS reports performance results to PMA after each transfer completes. When a user community is large and active, such as Open Science Grid or Earth System Grid, the PMA should have enough historical measurement information to make good estimates of current end-to-end performance that are close to the actual performance, even if passive monitoring from the data movement itself does not include active probing of resources.

We plan to make an open interface and modular design for the components. Communications between PMA, PS and DMS need to be based on well-defined interfaces that allow other services and projects to benefit from this proposed work. Each service will have a modular design so that essential techniques may be deployed and called through libraries by other services and project components.

1.2 Passive Measurement Archive (PMA)

Large science collaborations such as ESG and OSG support a large number of replication and download requests to share datasets among tens of thousands users in the community. The data transfer performance information is collected passively from data movement clients when a client downloads a file from a data server or replicates a dataset from a source to a destination, and sent to the Passive Measurement Archive (PMA). The historical measurement information in PMA will be utilized for an approximation of the end-to-end performance throughput. The performance estimation will then be used in replica selection as well as in optimizing data transfer properties in the PS. PMA is a perfSONAR based measurement repository service that the historical information can be archived and queried through the common standardized interface. With collected transfer monitoring information, PS makes a simple approximation for expected network performance estimate for a pair of nodes from the data source to the client destination.

1.3 Data Movement Service

Using multiple data transfer streams is a common technique applied in the application layer to increase the network bandwidth utilization transfer service. To achieve high throughput, the number of multiple connections needs to be adjusted according to the capacity of the underlying environment. From the previous research, we have designed Adaptive Data Transfer (ADT) algorithm to calculate the dynamic transfer parameters such as the concurrency level from a simple throughput prediction model with information from current data transfer operations instead of using predictive sampling as proposed in other researches, and does not depend on external profilers for active measurements. The number of streams is set dynamically in an adaptive manner by gradually increasing the number of concurrent connections up to an optimal point. This also enables us to adapt varying environmental conditions to come up with a high-quality tuning for best system and network utilization. Gradually improving the concurrency level brings a near optimal value without the burden of complex optimization steps to find the major bottleneck in a data transfer. In this adaptive algorithm, a change in the performance is detected, and the number of concurrent connections is adjusted accordingly. Adaptive tuning by adjusting the concurrency level dynamically provides higher throughput. The ADT algorithm focuses on application level auto-tuning methodologies that are applied in user-space for better transfer performance.
The project plans to implement the ADT algorithm as a library, and extend the existing Data Movement Service clients, such as Bulk Data Mover (BDM) and SRM client tools that are being used in production in ESG and OSG. These enhancements will optimize transfer throughput performance adaptively, with the initial transfer parameters provided by the PS, and increase the overall performance in large scale data replications and data downloading in science collaborations by enabling these clients to work with policy and performance estimation from the PS based on historical measurements from PMA. We also plan to release the ADT library to the community to be used by other data transfer client tools.

We will specifically address the following developmental challenges in adaptive data movement.

1) *Estimating initial number of multiple streams*. The latency directly affects throughput, and more transfer streams are needed to fill the network bandwidth when latency is higher. The challenge in ADT is to estimate the initial number of streams based on round-trip time (*RTT*) between the source and destination hosts. We will experiment with different models to estimate the initial number of transfer streams, which will be used for gradual adjustment for optimum tuning with the policy advice from PS.
2) *Determining the interval between the transfer parameter adjustment points*. The instant throughput performance is measured, but it may not be appropriate to make adjustment on the number of transfer streams after every measurement point. Considering the possibility of minor fluctuations in the network throughput, a threshold value is needed based on some transfer property such as the transferred data size before determining changes in the achievable throughput and adjusting the number of concurrent streams. We plan to determine the threshold value from the historical measurements in PMA.

1.4 The Policy Service

The project builds on a Policy Based Data Placement Service (PDPS), developed under funding from DOE and NSF. The original architecture of the PDPS is shown in Figure 2. During the first year of the project, we have significantly modified and improved this architecture, as described in Section 2.5.

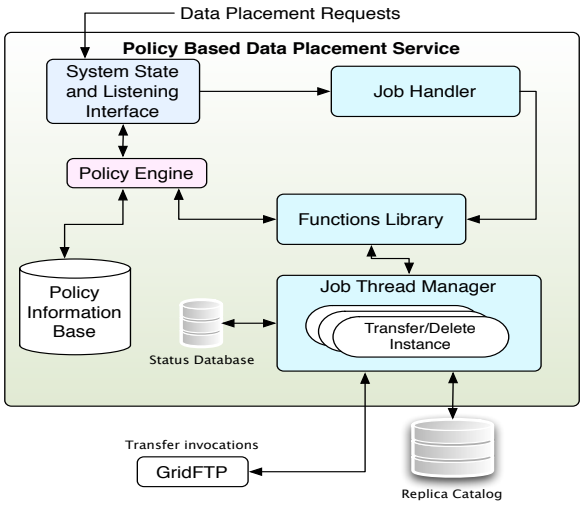

Figure 2: Architecture of PDPS

The original PDPS design included a *System State and Listening Interface* that received calls made to the PDPS, which were managed by the *Job Handler*. The system state interface also queried external monitoring services (e.g., Replica Catalogs) to determine the current state of the distributed system and created a system-wide snapshot of the current state. The *Policy Engine* periodically checked this snapshot of the state of the system against the policies stored in the *Policy Information Base (PIB)*. If the system state matched any policy's conditions, the Policy Engine performed the corresponding enforcement actions by invoking functions in the *Functions Library*. These enforcement actions included initiating data transfers via the *data transfer queue* and updating the system state in the *replica catalog* and *job status database*. The *Job Thread Manager* controlled a queue of transfer job instances that were invoked by the Functions Library and that initiated data transfer operations on external transfer services. The original PDPS implementation used the following existing components: third party GridFTP data transfer servers that moved the data; the Globus Replica Location Service as a replica catalog that was used both to determine current system state and record the existence of new replicas as they are created; and the Drools open source policy engine to enforce policies.

Our work addresses several challenges for policy-driven data movement of large-scale scientific datasets, including:

1) Understanding real-world VO-level policies that will affect the transfer advice provided to the DMS, including policies regarding resource preferences and resource allocations for users.
2) Using existing policy languages to express these VO policies and provide them as input to the policy engine in the PS
3) Using a policy engine to enforce policies and developing algorithms to resolve policy conflicts
4) Defining well-designed interfaces to the PS and DMS
5) Developing algorithms to generate transfer advice for the DMS that take into account both performance estimates and VO-level policy constraints, and studying how to indicate the level of certainty for the advice.

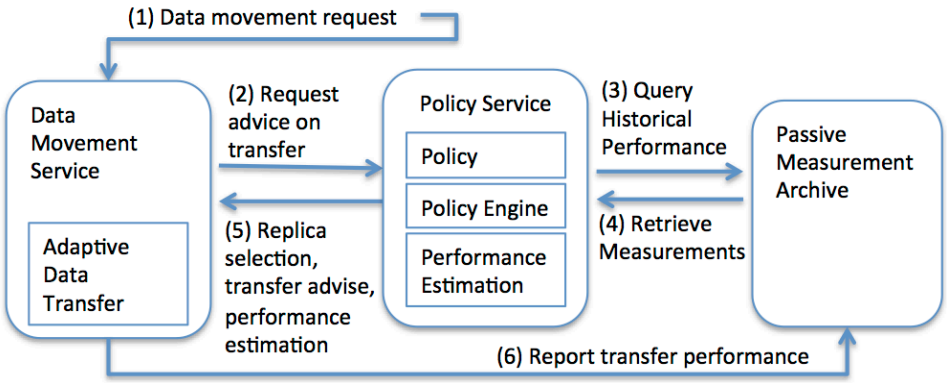

Figure 3: System interactions

1.5 Data Access Scenario to Illustrate Component Interactions

Our work supports a data access scenario (illustrated in Figure 3) that begins when a data movement request such as a download or data replication operation is (1) submitted to the DMS client. The DMS (2) requests advice on the transfer from the Policy Service (PS). The PS then (3) queries the PMA to obtain any historical performance measurements for the data source and destination pair of nodes. The PMA (4) returns historical measurements. The PS, using the simple approximation of the transfer performance based on the historical measurements as well as VO-level policies on resource preferences or limitations, generates advice for appropriate transfer parameters and (5) returns that advice to DMS. (An example of a resource preferences or limitations would be that a VO allows a transfer to consume at most 50% of available bandwidth between the source and destination sites.) Based on the transfer advice, the DMS then sets appropriate transfer parameters before initiating the data transfer. Later, the PS can provide updated advice to the DMS based on how well actual performance for a data transfer matches the predicted performance. For example, if transfer performance estimation indicates that transfers between a given source and destination are unexpectedly slow, the PS can use this information to update its advice to the DMS regarding the optimal parameters for that transfer. To support this functionality, we have designed add an interface between the PS and the DMS that allows the PS to accept requests for transfer advice from the DMS client and respond with advice on transfer properties (e.g., recommended concurrency levels and buffer sizes), replica selection when multiple copies are available, and estimates of transfer performance. We have also implemented an interface for the PS to query the PMA to retrieve historical measurement information. If source or destination has never been involved in any transfers previously, the PS uses the default values according to the policy and DMS will adjust the transfer adaptively. We are enhancing the policy logic in PS to take into account resource preferences and allocation policies specified by the VO in constructing transfer advice for the DMS.

2 Project Tasks

2.1 Component Design, Functional Enhancements and Interfaces

In the kickoff meeting in Oct. 2011, our development and deployment plans were divided into two phases to gain user experience and feedback. The initial development and deployment (phase 1) is designed to provide an optimized data transfer experience and easier transition from the current data movement practices in OSG and ESG to the new adaptive data movement. The second phase of the project is focused on the service-oriented functionality and generalized framework of the system. The two-phase project plans derived from specific use cases are illustrated in figures 4 and 5.

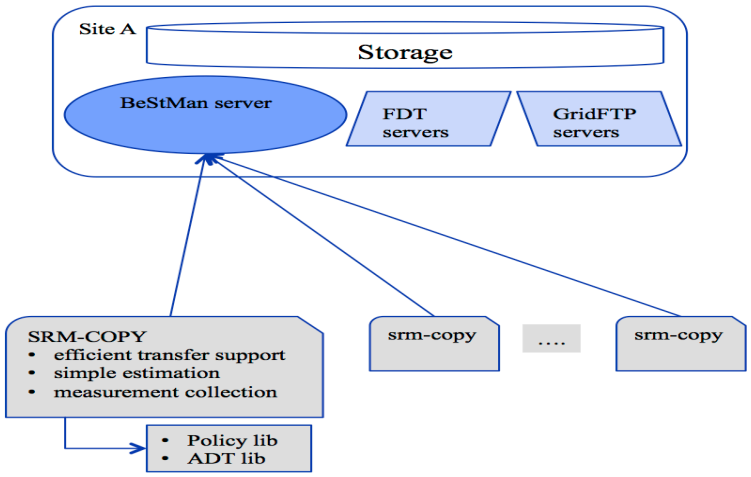

Figure 4: The initial development and deployment plan as a combined client tool

Figure 4 shows a use case where a data movement client interacts with a data storage site. This is a typical OSG use case for data staging. In phase 1 of the ADAPT project, we are enhancing the data movement

client (srm-copy) and integrating the client with the ADT and policy modules.

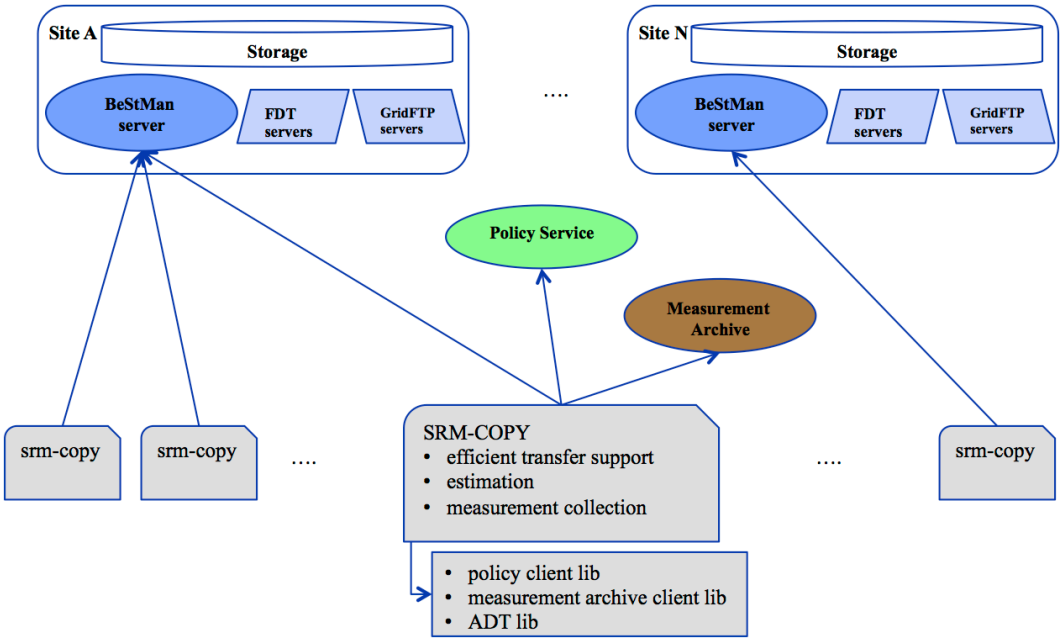

Figure 5: The second phase development and deployment plan as component services

Figure 5 shows a use case for phase 2, where data movement clients interact with multiple storage sites. This is a typical use case in OSG and ESG to stage a dataset for analysis or to retrieve a dataset from multiple storage repositories. In phase 2, we will enhance the data movement client integrated with ADT as a library module and deploy the policy logic as a service; we will also integrate perfSONAR as the measurement archive.

In addition to the existing functions from each component, the following enhancements have been discussed to the components in this project scope.

2.2 Adaptive Data Transfer Library

Adaptive Data Transfer (ADT) library calculates the dynamic transfer parameters such as the concurrency level from a simple throughput prediction model with information from current data transfer operations instead of using predictive sampling, and does not depend on external profilers for active measurements. The number of streams is set dynamically in an adaptive manner by gradually increasing the number of concurrent connections up to an optimal point. This enables us to adapt varying environmental conditions to come up with a high-quality tuning for best system and network utilization. In this library, a change in the performance is detected, and the adjusted number of concurrent connections is calculated accordingly. The ADT library has been developed as a general transfer adaptation library.

- The ADT library will be integrated into the client tools designated in the project, as well as distributed as a software library under open source license.
- ADT library is given a "current" transfer rate, and returns the number of streams to use.
- It is not going to make a transfer rate (or performance) estimation. It does not have any knowledge of the on-going performance rate information, transfer volume, or source-destination information.
- Internally, it calculates the number of streams compared to the past adjustments based on the previous research model.
- ADT library is per-process (or per-job) adaptive transfer adjustment module to maximize the data throughput for the job. It has no knowledge of other jobs on the same site or global information on the network.

2.2.1 ADT Library Interface

Int getNumberOfStreamsSuggestion (int currentThroughput,
int achievableThroughputPercentage,
int RTT, // Round Trip Time
int maxStreams)

This function is to provide a good predication at the number of connection streams to use for the client. We take consideration of several factors, namely, current network throughput, maximal connection streams , round trip time between source and target hosts, and achievable throughput percentage. ADT iterates through number of streams to find the one that gets the best throughput.

2.2.2 ADT Library Development Progress

ADT library has been developed as a general transfer adaptation library, and its first version is integrated into the SRM client data movement tool (srm-copy). The defined interface and the expected functionalities are implemented and tested.

2.3 Passive Measurement Archive

The data movement clients collect the data transfer performance information when the clients download files from a data server or replicate a dataset from a source to a destination, and send those collected information to the Passive Measurement Archive (PMA). The historical measurement information in PMA will be utilized for an approximation of the end-to-end performance throughput. The performance estimation will then be used in replica selection as well as in optimizing data transfer properties in the PS. In the first phase of the project, data movement clients and the PS share the historical performance information in a data log file as the PMA. Passive Transfer Monitor (PTM) module will be developed as an extension library of the client tools to retrieve the current transfer statistics from the data log file. In the second phase of the project, a perfSONAR based measurement repository service will be used as the PMA, so that the historical information can be archived and queried through the common standardized interface.

- A new client transfer monitoring module, Passive Transfer Monitor (PTM) will be developed as an extension library of the client tools to retrieve the current transfer statistics in the data log file.
- The current transfer statistics include the current transfer rate, total transferred data, current number of transfer streams, data source and destination.
- In the second phase of the development and deployment, the current transfer statistics will be archived into the perfSONAR-based repository as PMA.

2.3.1 PTM Library Interface

Long[] getStatistics () returns, in order,
currentTxfRate in MB/sec,
totalTransferredData in MB,
currentNumberOfStreams

This function is to gather information from current or historical runs of the client (srm-copy). It returns the statistics from the most recent transfer performed by the client, which includes the transfer rate (currentTxfRate), and size of total data transferred, and number of connection streams used.

2.3.2 PTM Library Development Progress

PTM library has been developed as a general transfer log reporting library. The log format follows the common logging practice guide for Grid logging as defined in http://escholarship.org/uc/item/1jz4k8hd. Its first version is integrated into the SRM client data movement tool (srm-copy). The defined interface

and the expected functionalities are implemented, and tested with PS component.

2.4 Data Movement Service

The project plans to implement the ADT library as defined above, and extend the existing Data Movement Service (DMS) clients, such as Bulk Data Mover (BDM) and SRM client tools that are being used in production in ESG and OSG. These enhancements and integration will show the adaptive data transfer improving the data access efficiency. The transfer throughput performance will be optimized adaptively, with the initial transfer parameters provided by the PS, and the adaptive data movement and the effect of the policies increase the overall performance in large scale data replications and data downloading in science collaborations. Each new component will be developed as modular components so that other data movement clients can easily integrate and use the adaptive data movement method.
During the first phase of the project, a new connection module, Adaptive Transfer Manager (ATM) have been developed as an extension of SRM client tools (e.g. srm-copy), and it will integrate original client tool, ADT library, PTM library and policy module library. Client tools will support Fast Data Transfer (FDT) data transfer protocol in the second phase of the project.

2.4.1 ATM Library Interface

Int getNumberOfStreamsSuggestion ()

The ATM library coordinates between the policy module and the ADT module. This function will first get the connection limit from policy module, taking this as the maximum connection streams, it gets the suggested connection streams from ADT and returns it to the caller.

2.4.2 Integrated Data Movement Client Development Progress

The ATM module has been developed and integrated with the SRM client data movement tool (srm-copy). The data movement functionality is enhanced and integrated with newly developed components, ADT library, PTM library, ATM module and PS module. The integrated test results are described in the later section, showing efficiency improvement with adaptive data transfer method and the effect of the policies.

2.5 Policy Service and Library

In this section, we present the redesigned Policy Service (PS) that we have implemented in the first year of the ADAPT project to provide policy-based transfer advice. The original Policy Service design, already described in Section 1.4, addressed the needs of scientific workflows, including ordering transfers based on the needs of the workflow and replica selection when multiple copies of a needed file are available. The use case for adaptive file transfers in the ADAPT project involves producing advice for file transfers with specified source and destination physical locations, taking into account VO policies that place limits on transfer parameters and overall research usage. Hence the Policy Service was redesigned to address these use cases.

In the first year of the project (Phase 1), the Policy Service was developed as a library module that can optionally be included with the srm-copy client to provide advice the number of parallel streams and other transfer rate parameters to use for each srm-copy transfer. The overall goal of the ADAPT project is to achieve higher throughput for transfers occurring in the environment. Each srm-copy client contains its own Policy Module. Network administrators at the Virtual Organization level define the overall resource limits between hosts based on their hardware resources (e.g. memory size, bus speed, storage speed) and the available network throughput 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 srm-copy transfers, what the actual recommended parameter limits should be for the srm-copy client instance. As the srm-copy client performs the transfer, it actively communicates with the

Policy Service to receive the most up-to-date advice on transfer parameter limits and informs the Policy Service of its chosen transfer parameters. The detailed ADAPT Policy Specification is included in Appendix A.

2.5.1 Policy Architecture

The ADAPT Policy Service architecture for the first phase consists of the following components:

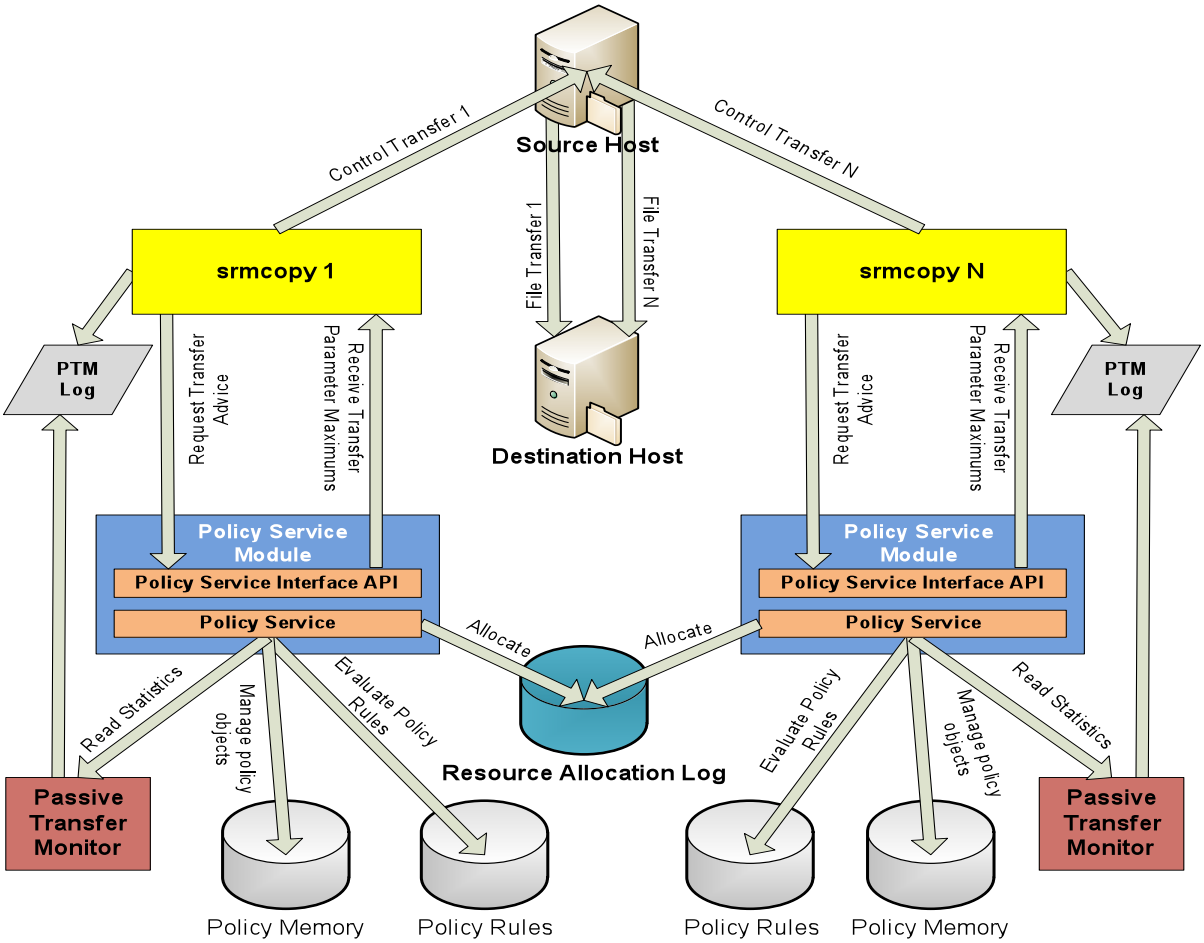

Figure 6: The Policy Service architecture

1) *Policy Service Module* - Java library that contains the policy service.
2) *Policy Service Interface API* - Java API used to communicate with the Policy Service.
3) *Policy Service*
   - manages policy sessions that contain policy rules and persistent policy memory
   - manages a *Passive Transfer Monitor (PTM)* instance
   - shares resource allocation information in the *Resource Allocation Log*.
4) *Policy Rules* - enforces the behavior of how transfers and cleanups are handled in the system.
5) *Policy Memory* - retains the current knowledge of information stored in policy. Its state is persisted between transfer requests.
6) *Passive Transfer Monitor (PTM)* - contains statistics that can be used for initial transfer parameters.

7) *Resource Allocation Log* - shared database by all srm-copy clients in the environment, allowing all processes to effectively share resource allocations amongst their common host machines.

2.5.2 Policy Operations

Next, we describe the operations supported by the Policy Service.

- **Initialization**

The initialization of the Policy Service includes the following steps. When an srm-copy client is executed to perform one or more transfers in serial, the srm-copy client instantiates the Policy Service library. The Policy Service library initializes itself based on Virtual Organization-specified transfer parameters in its configuration file. These parameters include:

- *default_max_rate*: the default maximum transfer rate to recommend to srm-copy for its transfer.
- *default_max_streams*: the default maximum number of parallel streams to recommend to srm-copy for its transfer.
- *maxBandwith*: name of file that contains the list of max bandwidth limits defined between two hosts.
- *maxParallelStreams*: name of file that contains the list of max parallel stream limits defined between two hosts.
- *policyFile*: name of file that contains the policy rules that are interpreted by the Policy Service.
- *passiveTransferMonitorPolicyFile*: name of file that contains the policy rules that are additionally interpreted if the Passive Transfer Monitor (PTM) is used.
- *passiveTransferMonitorLog*: name of file that contains initialization parameters used when the PTM is first invoked.
- *resourceAllocationDatabaseUrl*, *resourceAllocationDatabaseUser*, *resourceAllocationDatabasePassword*: parameters to use to connect to a shared resource allocation database.

Finally, if the *Passive Transfer Monitor (PTM)* module is included, it is initialized (with parameters from the *passiveTransferMonitorPolicyFile*, if specified) and stored in policy memory.

- **Initial Transfer Requests**

When an srm-copy client initiates a new transfer, it requests initial advice on the transfer from the Policy Service library. The Policy Service library retrieves information on the current resource allocations of the transfer's source and destination hosts and runs its policy rules to determine its advice. This advice is based on the current aggregate number of streams or rate that have already been allocated between the transfer's source and destination hosts and whether these exceed the defined site maximum. The advice also depends on whether the Passive Transfer Module exists and contains statistics on the parameters used for a previous transfer. The options include the following:

- If the currently aggregated number of streams or rate between the transfer's source and destination hosts exceed the defined site maximum, the maximum number of streams or rate for the transfer is set to the defined site maximum divided by the current number of transfers between them.
- If the currently aggregated number of streams or rate between the transfer's source and destination hosts is below the defined site maximum and the PTM exists and contains statistics on the previous transfer's number of streams and rate, the maximum number of streams or rate for the transfer is set to the PTM statistics.

- If the currently aggregated number of streams or rate between the transfer's source and destination hosts is below the defined site maximum and no PTM statistics exist, the defined default number of streams or rate is assigned to the transfer.

The srm-copy client receives the advice on its transfer and adjusts its transfer parameters to be at or below the recommended limits from policy. Then the srm-copy client informs the Policy Service library of its chosen transfer parameter values so that the PS library can properly record the actual resources allocated between the hosts.

- **Ongoing Transfer Updates**

The srm-copy client continues to contact the Policy Service library while the transfer is occurring to receive advice on transfer parameter adjustments. This process is similar to the request for initial transfer advice.

- **Transfer Complete**

The srm-copy client informs the Policy Service library when the transfer is completed so that its allocated resources can be freed up for current and future transfer requests.

- **Logging**

Activity in the Policy Service is logged through log4j. Transfers and cleanup operations that are started, have completed, or have failed are written to a Transfer Statistics Log. This allows aggregate statistics to be retrieved on the number of total transfers completed, total cleanups completed, etc. The number of currently allocated streams and the rate for each ongoing transfer are recorded in the Resource Allocation Log.

2.5.3 Policy Rules

The Policy Service was implemented using the Drools open source policy engine. The policy rules that were developed for the first phase in the project are described in Appendix A.

2.5.4 Policy Service and Library Development Progress

The development of the first phase of the Policy Service library has been completed and successfully unit tested for the identified use cases and policy details. The distributable package containing the Policy Service library is available on an ISI download site - http://software.misd.isi.edu/java/policy/adapt/policy-adapt-latest.tar.gz. The Policy Service library installation and user documentation, along with API specifications, have been posted to the ADAPT project page. Finally, the Policy Service library has been integrated and tested with the SRM client tool (srm-copy).

2.6 Component Interactions for the Initial Phase of the Development and Deployment

Figure 7 shows the interactions between the components for the first phase of the development. srm-copy calls time estimation to ATM via (1) with data volume and source/destination site information. Then ATM calls the PS via (3) to find out the estimation and number of maximum transfer streams for the client, and returns the estimation and the number of transfer streams to srm-copy. The PS may collect the current transfer statistics (e.g. transfer rate, transferred data volume) from PTM via (5) that srm-copy has recorded to determine the past transfer measurements as well as measurements from on-going transfers. srm-copy may call ATM for adjustment on the number of transfer streams via (1) while the data transfer is in progress, and ATM will contact the PS via (3) for the various transfer limits (e.g. instant max limit on the number of transfer streams), and call ADT for the adjustments via (2). The adjusted data streaming information will then go back to srm-copy as well as to the PS. The PS can save the current streaming information in the "resource allocation logs" for "global" view of the total number of streams on the site. This "resource allocation logs" can be shared among policy modules in different srm-copy processes, so that policy modules can determine the "global view" on the site.

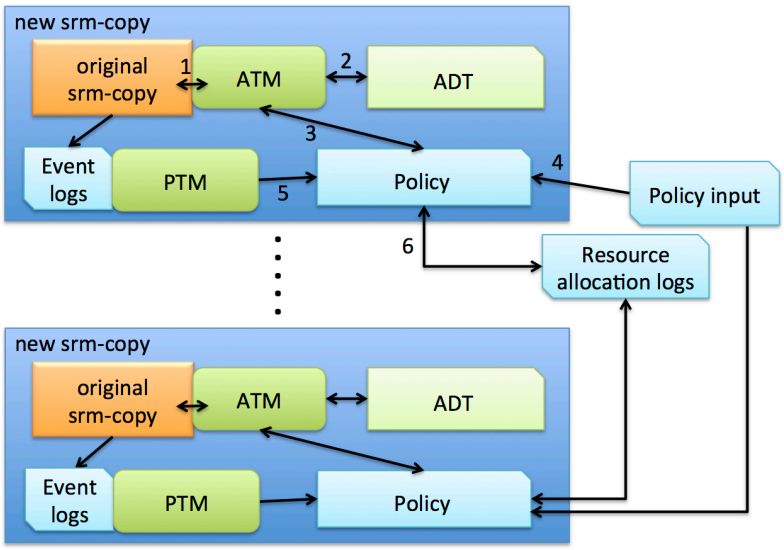

Figure 7: the interactions between the components for the first phase of the project

Figure 8 shows a typical use case for OSG-based analysis. This same scenario can be applied to STAR experiment and to others. The numbered steps show the ordered operations. Users with the current srm-copy and the new extended srm-copy with ATM, ADT, PTM and PS would not see the differences in the usage and operations, but only in the performance.

The PS library during the first phase is integrated with client jobs (i.e., srm-copy), running on separate worker nodes but sharing logs from a central disk for a "global view" of how resources are allocated in the distributed system. The policy "input files" are shared by all srm-copy processes with policy modules, and the PS knows the site resource status, so it can determine instant limits per srm-copy job.

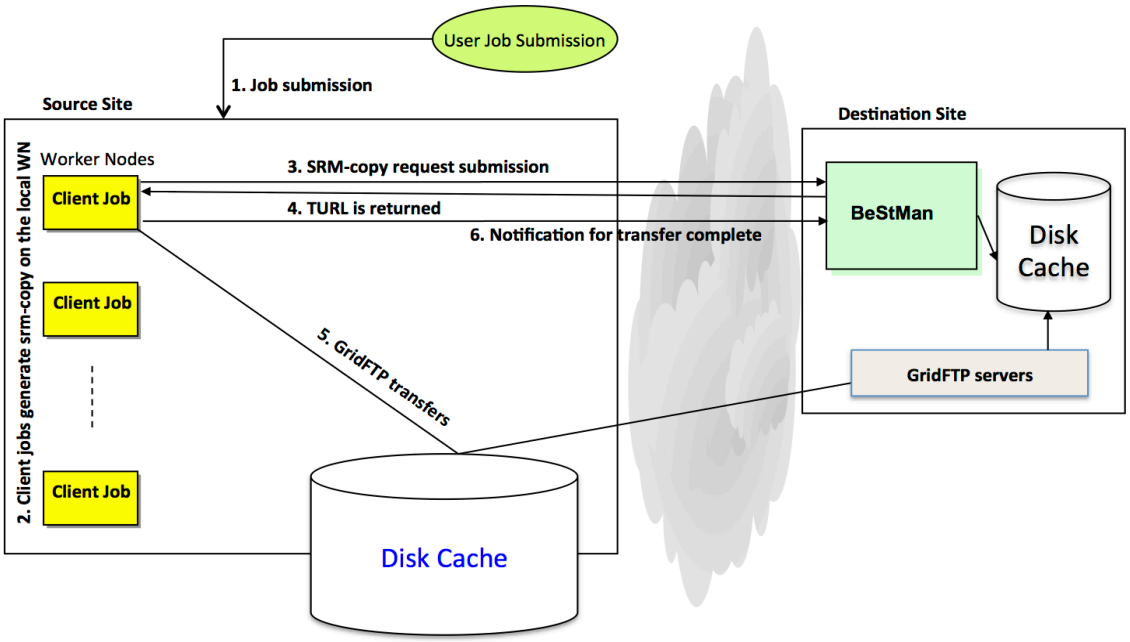

Figure 8: A Tupical use case for OSG- based analysis. E.g. Source at PDSF/NERSC and Destination at UNL or one of the OSG sites

Figure 9 shows another sample use case: making the 3rd party transfers. The numbered steps show the ordered operations with the client job (e.g., srm-copy). Again, users with the current srm-copy and the new extended srm-copy with ATM, ADT, PTM and PS would not see the differences in the usage and

operations, but only in the performance.

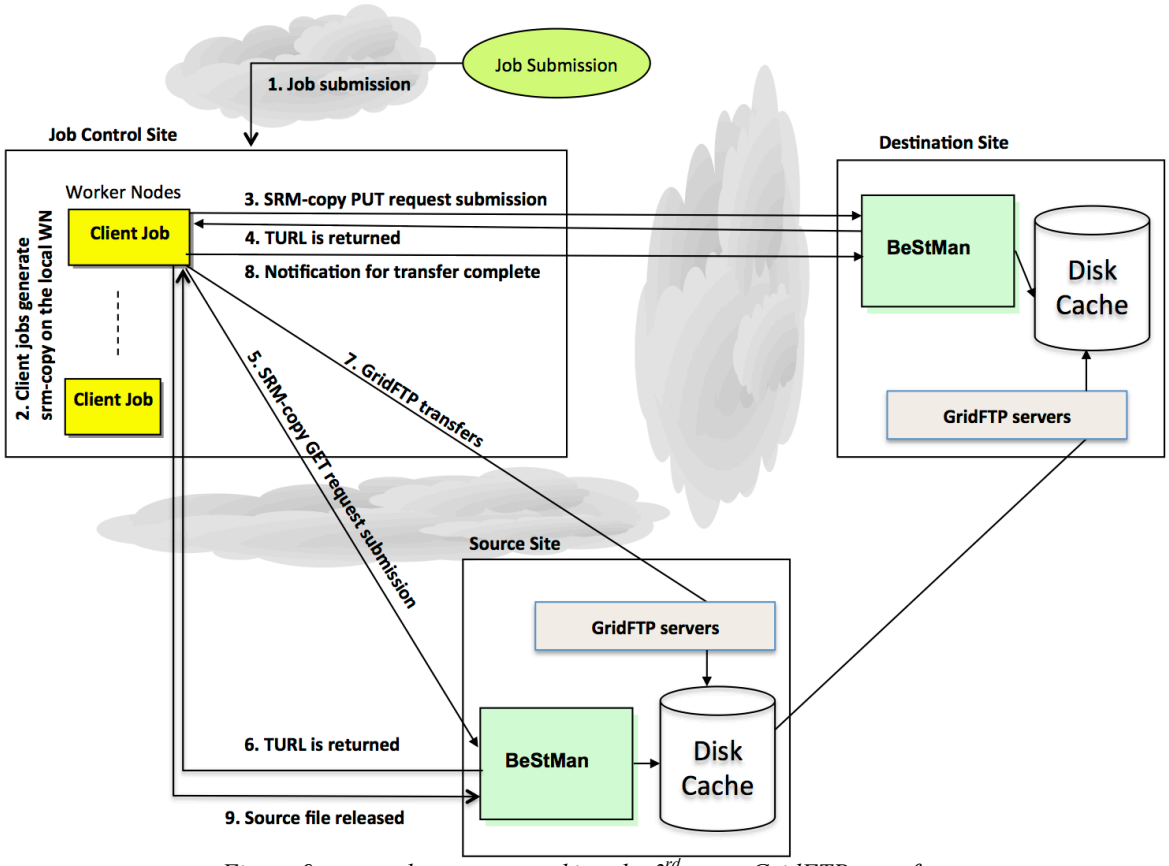

Figure 9: a sample use case making the 3rd party GridFTP transfers

2.7 Component Interactions Plan for the Second Phase of the Project

Figure 10 shows a use case for the second phase of the project, expanded from figure 9, with PS as an external service and with the external measurement archives (PMA). There can be many source and destination sites, involved in the VO data traffic, and they would affect the transfer performance of each data transfer significantly. In the lower-right corner, a new client job is shown, which is evolved from the figure 7.

The new enhanced srm-copy in Figure 10 is simplified from the first phase. The PS becomes an external service, and ATM connects the PS through a web service. Current transfer statistics from the client tools will be archived in perfSONAR-type of measurement archives (PMA), through PTM. The PS will be able to connect the measurement archives for historical performance measurements via perfSONAR-based interface. Also, network performance estimation service from other projects may be available with a well-defined interface to get a performance estimate for a pair of source and destination.

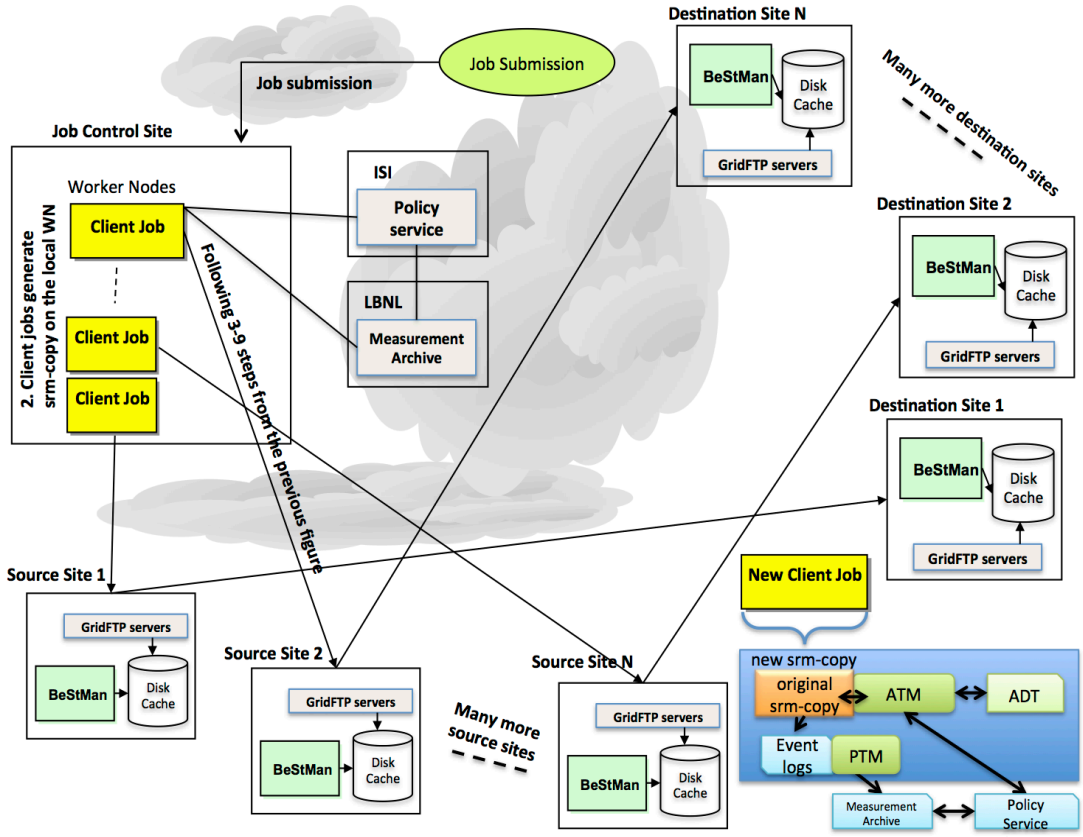

Figure 10: The second phase development and deployment with a use case

2.8 Testing Use Case and Measurements

2.8.1 OSG Use Case

Most of OSG use cases send the data to the OSG resources for data staging before submitting the data analysis jobs. We have collaborated with University of Nebraska at Lincoln (UNL), one of the OSG sites, and the STAR experiment at RHIC for the test. Data are being generated on a large cluster at PDSF/NERSC and are then transferred to UNL. We have run the new enhanced srm-copy jobs on NERSC/PDSF worker nodes, and tested all aspects of the first phase system with GridFTP transfers from NERSC to UNL for data analysis on OSG resources. The use case is similar to the cases in Figure 8 and Figure 9.

2.8.2 Measurement Collection

Data transfer performances with the ADAPT enhancements with the ADT and Policy modules have been collected and analyzed in Figure 11.

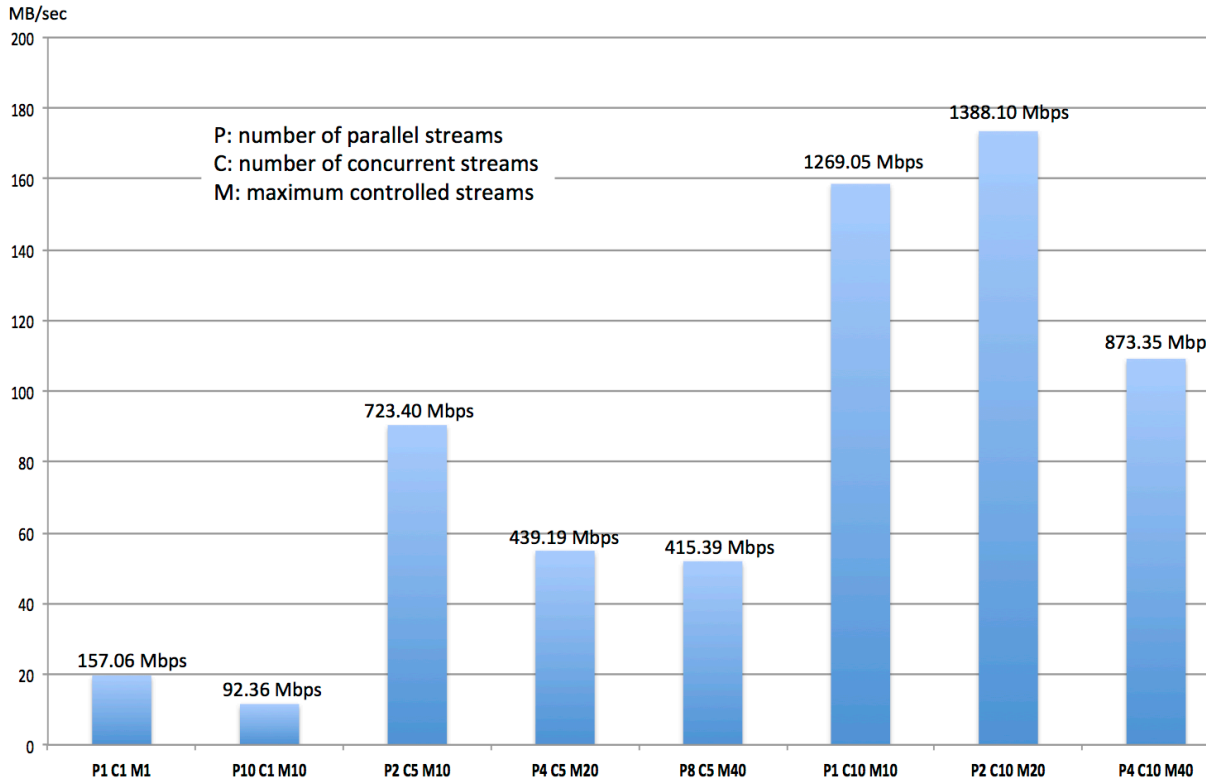

Figure 11: The aggregated throughput plot for the data transfers from NERSC/PDSF to UNL

Each bar in the graph represents throughput (in MB/sec) achieved using a combination of transfer parameters, including the number of parallel streams, number of concurrent transfers and maximum allocated streams between the two sites. In the left bar of the graph, these parameters are all equal (P=C=M=1), and there is no effect from ADAPT enhancements. When the concurrency C=1 (as in the two leftmost bars in Figure 11), only one data transfer at a time will be initiated by the srm-cpy client between the two sites, resulting in the lowest transfer rates.

In ADAPT, the ADT module manages the number of active concurrent transfers (C) and the number of parallel streams (P) for each file transfer, to get the best throughput within the number of total streams (M) provided by the policy module. When there are more data streams, the aggregated throughput rate is higher in general. However, the transfer rate per file may decrease when there are more data streams than the host resources and network can accommodate, such as in cases with a higher number of parallel streams that use more CPU and other system resources. The negative effect on the transfer performance is more significant when more data streams oversubscribe the available network bandwidth. For example, consider the middle three bars in the graph, which show performance with a concurrency level of 5 simultaneous transfers. As the parallelism level of each transfer increases from 2 to 8, the aggregate bandwidth achieved drops from 723 Mbits/sec to 415 Mbits/sec, because the available resources between the source and destination nodes are oversubscribed. Similarly, for the right three bars of the graph, with a concurrency level of 10, the overall bandwidth achieved drops when parallelism increases from 2 to 4.

The ADAPT enhancements help clients to manage the number of data streams to achieve the best overall throughput. For example, Figure 12 shows the effect of ADAPT enhancements. In our experiments, the maximum number of data streams between source and destination hosts (M=10, 20, 40) is set by the policy rules. The experiments show the effect of ADAPT enhancements that vary the number of parallel

streams per transfer and the number of concurrent transfers; these are managed by the ADT module using the transfer advice from the policy module, so that each file transfer can reach its maximum throughput. When the maximum data streams is set by the policy rules, the aggregated throughput is increased by adjusting the number of parallel streams and the number of concurrent transfers.

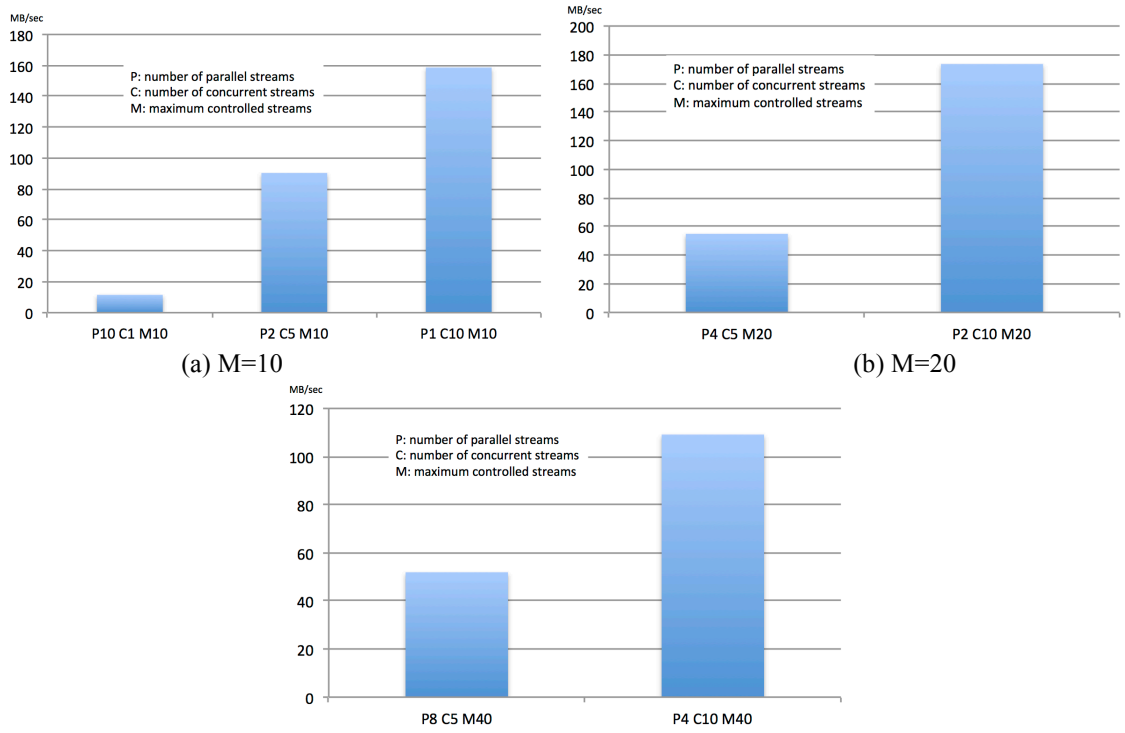

Figure 12: The aggregated throughput plot for the ADAPT managed data transfers from NERSC/PDSF to UNL

The test runs and collection of measurements are still in progress, and we should be able to analyze the effects ADAPT enhancements in greater detail in the coming months.

3 Additional information about the ADAPT project

1) Project web: http://sdm.lbl.gov/adapt/
2) Project mailing list: adapt@hpcrd.lbl.gov
3) Project kick-off meeting on 10/2011 at LBNL.
4) Project discussion over conference calls every other week.

4 Plans for Year 2

Next, we describe our proposed work for Year 2 of the ADAPT project.

4.1 Continued testing of phase 1 functionality

- We will continue testing functionality and performance of the ADAPT software between nodes on the Open Science Grid (OSG). Current tests are performed between NERSC/PDSF and University of Nebraska at Lincoln (UNL).
- These tests will include single client and multi-client tests.

4.2 Improvements to networking software (srm-copy, DML, and BDM)

- Enhance srm-copy to have Policy Service as an option.
- Integrate adaptation, policy logic into other data tools produced by LBNL team, including Bulk Data Mover (BDM) and Data Mover Lite (DML).
  - Take advantage of heavy use of webstart DML by the climate community.
- Build ADT module as an independent library.
- Make use of the Policy Service in these tools.
- For users downloading the climate data using DML from ESG data sites, they can consult the Policy Service to get advice on what transfer parameters they should use.

4.3 Improvements to Policy Service

- Deploy the policy engine as a service, making it easier for the clients to share state related to resource allocation. This work will include:
  - Modifying the policy library to remove the existing resource allocation log database.
  - Building a web service interface around the existing library.
  - Making an option to enable/disable use of policy service in the clients.
- Explore richer policies for managing transfer resources
- Conduct an experimental evaluation to determine which of the current and planned policies are most effective in reducing transfer times, increasing overall throughput.

4.4 Working with applications

- Continue to work on deploying and using the software on Open Science Grid (including PDSF cluster at NERSC and clusters at UNL).
- Continue enhancing srm-cpy and deploying within the OSG software stack.
- Work to engage the climate community, which is currently making extensive use of Data Mover Lite (DML) client.
  - Build on our historical ties with the Earth System Grid project.
  - Put efforts of having DML included in the Earth System Grid P2P software stack in the fall time-frame.

5 Summary

The goal of the ADAPT project is to develop and deploy a general-purpose data access framework for scientific collaborations that provides fine-grained and adaptive data transfer management, lightweight passive performance monitoring, and enforcement of site and VO policies for resource sharing. This project contributes to improve the data accessibility in a large scientific collaboration, providing efficiency in data movement and enforcement of policies for data access and resource sharing.

During the first year of the project, we developed all basic components in the software framework, and combined as a client tool for the user experiences and feedback. We also have collected measurements with initial evaluation indicating efficiency. During the second year of the project, we will build service framework and reach out science communities for collaboration.

Appendix A: ADAPT Policy Specification

1 Policy Module Interface

public interface PolicyService

Service that manages transfer advice based on VO-level policies

public final class Transfer extends AbstractEntity implements java.lang.Comparable<Transfer>

Defines a transfer that is being sent to the policy service for advice.

RESOURCE_ALLOCATION_LOG

Manages the current resources allocated for all transfers using policy

- getResourceAllocation(String transferId)
- addResourceAllocation(String transferId, URI source, URI destination, int transferStreams, float rate)
- updateResourceAllocation(String transferId, int transferStreams, float rate)
- removeResourceAllocation(String transferId)
- getAggregatedTransferStreams(String sourceHost, String destinationHost)
- getAggregatedRate(String sourceHost, String destinationHost)

RESOURCE_ALLOCATION_ENTRY

An entry in the resource allocation log

- getId()
- getSource()
- getDestination()
- getTransferStreams()
- getRate()

2 Use Cases

2.1 New Transfer

An LBNL client tool, such as srm-copy or BDM, requests advice from policy for a new data transfer operation. Basic course of action is following:

1. Client tool is executed to transfer data from a source to destination.
2. Client tool sends a TRANSFER request via the ATM to the policy module for advice.
    a. source=<source URI of the transfer>
    b. destination=<destination URI of the transfer>
    c. properties
        i. local_file_host=<FQHN of local file> if this is a two-party download operation, not specified for third-party transfers
        ii. data_volume=<size of the data to be transferred, in bytes>
3. Policy module receives new TRANSFER request and inserts it into policy memory with a unique ID.
4. Policy module applies policy rules to the TRANSFER request.
    a. One or more policy rules may modify the max_streams TRANSFER property.
    b. One or more policy rules may modify the max_rate TRANSFER property
    c. Policy module writes new TRANSFER entry to the resource allocation log using the max_streams value.
5. Policy module responds with a modified TRANSFER.
    a. source=<source URI of the transfer>
    b. destination=<destination URI of the transfer>
    c. id=<unique identifier assigned to the TRANSFER in the policy memory>
    d. properties

i. local_file_host=<FQHN of host> if previously set in the request.
ii. data_volume=<size of the data to be transferred, in bytes> as previously set in the request.
iii. max_streams=<number of parallel streams> the max number of parallel streams allowed for the transfer.
iv. max_rate=<estimated transfer rate in Kb/sec> the max transfer rate to use for the transfer.

6. Client tool receives TRANSFER response from Policy module and begins the data transfer based on parameters in the response.

2.2 File Transfer Adjustment

An LBNL client tool, such as srm-copy or BDM, requests advice from policy for an adjustment to a data transfer operation that is in progress. Basic course of action is following:

1. Client tool sends a TRANSFER request via the ATM to the Policy module for advice on an existing transfer.
    a. source=<source URI of the transfer>
    b. destination=<destination URI of the transfer>
    c. id=<previously assigned ID of the transfer in policy memory>
    d. properties
        i. local_file_host=<FQHN of local file> if this is a two-party download operation, not specified for third-party transfers
        ii. data_volume=<size of the data to be transferred, in bytes>
2. Policy module receives the updated TRANSFER request and updates the instance in memory.
3. Policy module applies policy rules to the TRANSFER request.
    a. One or more policy rules may modify the max_streams TRANSFER property.
    b. One or more policy rules may modify the max_rate TRANSFER property.
4. Policy module responds with a modified TRANSFER.
    a. source=<source URI of the transfer>
    b. destination=<destination URI of the transfer>
    c. properties
        i. local_file_host=<FQHN of local file> if previously set in the request.
        ii. data_volume=<size of the data to be transferred, in bytes> as previously set in the request.
        iii. max_streams=<number of parallel streams> the recommended number of parallel streams to perform the transfer.
        iv. max_rate=<estimated transfer rate in Kb/sec> the estimated transfer rate that the client will get from the transfer.
5. Client tool receives TRANSFER response from Policy module and calls the ADT for adjustments using the TRANSFER properties.
6. Client tool receives adjustments from the ADT and applies them to the current transfer.
7. Client tool sends an updated TRANSFER to the Policy module via the ATM.
    a. source=<source URI of the transfer>
    b. destination=<destination URI of the transfer>
    c. id=<previously assigned ID of the transfer in policy memory>
    d. properties
        i. local_file_host=<FQHN of local file> if this is a two-party download operation, not specified for third-party transfers
        ii. data_volume=<size of the data to be transferred, in bytes>
        iii. adjusted_streams=<number of parallel streams>, applied to the data transfer based on ATM estimates.
        iv. adjusted_rate=<current transfer rate in KB/sec> the current rate reported by the

client tool.

8. Policy module applies policy rules to the updated TRANSFER.
    a. Policy module updates TRANSFER entry in the resource allocation log based on adjusted_streams and/or adjusted_rate.

2.3 File Transfer Complete

An LBNL client tool, such as srm-copy or BDM, completes a data transfer that the policy module is aware of. Basic course of action is following:

1. Client tool sends an updated TRANSFER request via the ATM to the Policy module.
    a. id=<previously assigned ID of the transfer in policy memory>
    b. properties
        i. local_file_host=<FQHN of destination> if this is a two-party download operation, not specified for third-party transfers
        ii. data_volume=<size of the data that was transferred, in bytes>
        iii. adjusted_streams=<number of parallel streams> that were in use when transfer completed.
        iv. adjusted_rate=<current transfer rate in KB/sec> the transfer rate that was recorded when transfer completed.
        v. STATUS=COMPLETED
2. Policy module receives the updated TRANSFR request and updates the instance in memory.
3. Policy module applies policy rules to the TRANSFER request.
    a. Policy module removes the TRANSFER entry from the resource allocation log.
    b. Policy module removes the TRANSFER from local policy memory.

2.4 File Transfer Failure

An LBNL client tool, such as srm-copy or BDM, experiences a fatal error during data transfer that the policy module is aware of. Basic course of action is following:

1. Client tool sends an updated TRANSFER request via the ATM to the Policy module.
    a. id=<previously assigned ID of the transfer in policy memory>
    b. properties
        i. STATUS=FAILED
2. Policy module receives the updated TRANSFER request and updates the instance in memory.
3. Policy module applies policy rules to the TRANSFER request.
    a. Policy module removes the TRANSFER entry from the resource allocation log.
    b. Policy module removes the TRANSFER from policy memory.

3 Policy Details

3.1 Administered Defaults

The following describes the parameters that are adjustable in properties files and are interpreted by rules files.

max_bandwidth_sites.txt

- Lists the max total bandwidth limit between two sites (in KB/s). The max bandwidth per instance divides the max by the number of known transfers.
- Parameters: <source host> <destination host> <max bandwidth>

max_streams_per_process_sites.txt

- Lists the max streams allowed between two sites. The max streams per instance divides the max by the number of known transfers.
- Parameters: <source host> <destination host> <max streams>

3.2 Rules

The following describes the rules that are applied in the Policy module.

Insert a new transfer into policy memory

- A new request for a transfer is added to policy memory.
- Preconditions:
  - no TRANSFER exists in memory with the new transfer request's transfer_id, source, and destination
- Results:
  - TRANSFER is inserted into policy memory.

Set default max streams for new transfer when no PTM is available

- A default number of streams are set for a new transfer using the administered default.
- Preconditions:
  - TRANSFER with no max_streams property set
  - PTM has not been loaded or initialized
- Results: max_streams property is written to TRANSFER

Set default max bandwidth for new transfer when no PTM is available

- A default bandwidth is set for a new transfer using the administered default.
- Preconditions:
  - TRANSFER with no max_rate property set
  - PTM has not been loaded or initialized
- Results: max_rate property is written to TRANSFER

Set max streams to PTM number of streams for new transfer

- The previously used PTM streams value is set for a new transfer
- Preconditions:
  - PTM loaded and initialized
  - TRANSFER with no max_streams property set
- Results: max_streams property is written to TRANSFER

Set max rate to PTM rate for new transfer

- The previously used PTM rate value is set for a new transfer
- Preconditions:
  - PTM loaded and initialized
  - TRANSFER with no max_rate property set
- Results: max_rate property is written to TRANSFER

Read resource allocation log

- The resource allocation log is read to determine how many instances are utilizing throughput and parallel streams on the sites
- Preconditions:
  - TRANSFER
  - No resource allocation info on TRANSFER
- Results: Resource allocation entries written to memory for TRANSFER

Enforce limit for max_streams_per_process on a transfer

- The max_streams_per_process limit is enforced on a TRANSFER
- Preconditions:
  - TRANSFER
  - max_streams_per_process limit for sites in TRANSFER
  - resource allocation information for sites in TRANSFERmax_streams property in

TRANSFER > (max/number of transfers in resource allocation)

- Results: max_streams property is modified in TRANSFER to be the max

Enforce limit for bandwidth on transfer

- The max_bandwidth_sites limit is divided among resource allocated TRANSFER instances and enforced on a TRANSFER
- Preconditions:
  - TRANSFER
  - max_bandwidth_sites limit for sites in TRANSFER
  - resource allocation information for sites in TRANSFER
  - max_rate property in TRANSFER > (max/number of transfers in resource allocation)
- Results: max_rate property in TRANSFER are modified to be max - total allocated

Create resource allocation entry for transfer

- A new transfer is recorded in the resource allocation log.
- Preconditions:
  - TRANSFER (adjusted_streams, adjusted_rate properties set)
  - No resource allocation record for TRANSFER
- Results: TRANSFER is written to resource allocation log with adjusted streams, adjusted_rate information

Update resource allocation entry for transfer

- An existing transfer is updated with the actual transfer information
- Preconditions:
  - TRANSFER (adjusted_streams, adjusted_rate properties set)
  - Resource allocation entry exists for TRANSFER
- Results: TRANSFER record in resource allocation log is updated with adjusted_streams, adjusted_rate information

Remove resource allocation entry for completed transfer

- An existing transfer is updated as completed and is removed from the resource allocation log
- Preconditions:
  - TRANSFER (status=COMPLETED property)
  - Resource allocation entry exists for TRANSFER
- Results: TRANSFER record in resource allocation log is removed

Remove resource allocation entry for failed transfer

- An existing transfer is updated as failed and is removed from the resource allocation log
- Preconditions:
  - TRANSFER (status=FAILED property)
  - Resource allocation entry exists for TRANSFER
- Results:
  - TRANSFER record in resource allocation log is removed
  - TRANSFER object in policy is removed

Record completed transfer

- An existing transfer that completed properly is recorded in policy logs
- Preconditions:
  - TRANSFER (status=COMPLETED property)
  - No resource allocation entry for TRANSFER
- Results:

- TRANSFER is recorded in policy logs (rate, streams, source, destination)
- TRANSFER object in policy is removed