StorNet: Integrating Storage Resource Management with Dynamic Network Provisioning for Automated Data Transfer

Final Report
August 2009 through July 2011

http://sdm.lbl.gov/stornet/
stornet@lbl.gov

Principal Investigators
Dantong Yu, Brookhaven National Laboratory
Arie Shoshani, Lawrence Berkeley National Laboratory

StorNet Team
Dimitrios Katramatos, Sushant Sharma (Brookhaven National Laboratory)
Junmin Gu, Vijaya Natarajan, Alex Sim (Lawrence Berkeley National Laboratory)

Abstract

Modern scientific data-intensive applications brought about the need for novel data transfer technologies and automated tools capable of effectively utilizing available raw network bandwidth and intelligently assisting scientists in replicating large volumes of data to desired destinations in a timely manner. In this final report of the project, we describe the design of StorNet, an integrated end-to-end resource provisioning and management system for high performance data transfers that can operate with heterogeneous network protocols and storage systems in a federated computing environment. StorNet allocates and co-schedules storage and network resources involved in data transfers. It is based on existing Berkeley Storage Manager, TeraPaths, and OSCARS capabilities. StorNet provides data intensive applications with the capability of predictable, yet efficient delivery of data at rates of multiple gigabits per second, bridging end-to-end advanced storage and network technologies in a transparent way.

Table of Contents

1 Introduction

Data-intensive application communities, including high energy and nuclear physics, astrophysics, climate modeling, nanoscale materials science, and genomics, just to name a few, are expected to generate exabytes of data over the next 5 years. Such data must be transferred, analyzed, and visualized by geographically distributed teams of scientists. This expectation of explosive growth in stored data and globally distributed data processing needs, underpinned by the maturing Grid and cloud computing technologies, has generated critical requirements for new predictable and well-behaved data transfer technologies and automated tools. To expedite scientific discoveries, these data transfer tools need to intelligently assist scientists in replicating large volumes of data whenever and wherever necessary. Existing data transfer techniques face unprecedented challenges in handling not only the sheer volume of data, but also the heterogeneous environment where data are imported from and exported to. An obstacle to managing these challenges is the inability to provide end-to-end bandwidth guarantees from the source storage systems to the destination storage systems. Furthermore, technology advancements give rise to performance improvements while also increasing the complexity of resource management and provisioning. Recently, two major research and education networks, ESnet, run by the US Department Of Energy (DOE), and Internet2, have been enhanced with advanced dynamic circuit switching technologies and network resource reservation systems to ensure on-demand bandwidth guarantees and Quality of Service (QoS). Data storage technologies have demonstrated significant improvements as well, through the use of advanced parallel file systems that enhance I/O bandwidth, and solid state disks (SSD) that can provide read/write access as much as ten times faster than hard drives. The StorNet project addresses the end-to-end resource provisioning and management issues encountered in automated data transfers by seamlessly integrating advanced network resource reservation capabilities with enhanced storage resource management technology.

2 Motivation and Background

Common requirements among today's experimental science applications that are of critical importance to the mission of large experimental facilities, such as the Large Synoptic Survey Telescope (LSST) [1], the Large Hadron Collider (LHC) [2], the Spallation Neutron Source (SNS) [3], the Advanced Photon Source (APS) [4], and the Relativistic Heavy Ion Collider (RHIC) [5], are: (i) intensive data transfers; (ii) remote visualizations of datasets and on-going computations; (iii) computational monitoring and steering; and (iv) remote experimentation and control. These applications utilize a wide variety of platforms, hardware, network, storage media, and software components to deliver the critical data storage functionality: file servers (NFS and AFS), various FTP file servers, mass storage systems, relational databases, and web servers for serving files and on-line streaming video. Storage and processing of raw data takes place at geographically distributed computing facilities; sharing data across the globe is realized through transfers over high-speed networks. Because the default network behavior is to treat all data flows equally, data flows of higher priority and/or urgency may be adversely impacted by competing data flows of lower priority. In distributed data-intensive environments, this can be a major problem that significantly degrades the effective "goodput" of the overall system. The policies and priorities of user communities cannot be effectively expressed or implemented in the network, except by very labor-intensive and error-prone human intervention.

There is an evident need for coordination between storage and network systems to better service the data transfers of the user community. From the network perspective, the capability to prioritize, protect, and regulate the various data flows becomes of high importance, because such a capability can be used for deterministically schedule network resources to support user community priorities and, furthermore, co-schedule associated resources such as storage systems. From the storage systems perspective, the source and destination storage systems need to have adequate bandwidth and storage allocation to take advantage of the network capabilities and increase the reliability and predictability of a transfer. Furthermore, data transfers typically take a long duration and transient failures are likely to occur, so failure detection and recovery mechanisms are also necessary.

The primary goal of StorNet is to achieve the coordination of storage and network resources by taking advantage of existing systems, some already used in production, making them interoperable, and augmenting their functionality. In addition to storage resource provisioning coordination between source and target storage systems, there needs to be bandwidth provisioning coordination between the storage systems and the underlying network resources.

StorNet is a joint project between Brookhaven National Laboratory and Lawrence Berkeley National Laboratory, funded for two years from August 2009 to July 2011. The systems used by StorNet are LBNL's storage resource manager known as the Berkeley Storage Manager (BeStMan) [6], BNL's TeraPaths end-to-end virtual network path reservation system [7], and ESnet's On-demand Secure Circuits and Advance Reservation System (OSCARS) [8] network provisioning tool supported by both ESnet and Internet2.

3 TECHNOLOGIES IN STORNET

3.1 Berkeley Storage Manager (BeStMan)

When dealing with storing large amounts of data, scientists need to interact with a variety of storage systems, each with different interfaces and security mechanisms, and to pre-allocate storage to ensure that data generation and analysis tasks can take place successfully. To accommodate this need, the concept of Storage Resource Managers (SRMs) was developed at Lawrence Berkeley National Laboratory (LBNL) [9,10].

SRMs are middleware components whose function is to provide a common storage access interface, dynamic space allocation, and file management for shared distributed storage systems. The SRM interface was standardized, and the specification led to the development of multiple SRMs that interoperate with each other by various institutions around the world [11,12,13]. SRMs are designed to provide support for storage space reservations, flexible storage policies, lifetime control of files to manage space cleanup, and performance estimation. The most recent version of an SRM developed at LBNL, is called the Berkeley Storage Manager, or BeStMan. BeStMan is designed in a modular fashion, so that it can be adapted easily to different storage systems (such as disk-based systems, mass storage systems, and parallel and distributed file systems, such as Lustre, GPFS, PVFS2 and HDFS) as well as use different transfer protocols (including GridFTP, FTP, BBFTP, HTTP, HTTPS). BeStMan is implemented in Java in order to be highly portable. It supports the SRM functions, and in addition, directory management and brokering service for accessing files in the distributed system. It manages queues of multiple requests to get or put files into spaces it manages, where each request can be for multiple files or entire directories. When managing multiple files, BeStMan can take advantage of the available network bandwidth by scheduling multiple concurrent file transfers.

3.2 TeraPaths

The TeraPaths project [7] at Brookhaven National Laboratory (BNL) has been developing a host-to-host network resource reservation tool. TeraPaths utilizes a combination of DiffServ-based LAN QoS with WAN MPLS tunnels and dynamic circuits to establish host-to-host virtual paths with QoS guarantees. These virtual paths prioritize, protect, and regulate network flows in accordance with site agreements and user requests, and prevent the disruptive effects that conventional network flows can bring to one another.

Providing a host-to-host virtual network path with QoS guarantees (e.g. guaranteed bandwidth) to a specific data flow requires the timely configuration of all network devices along the route from a given source to a given destination. Typically, such a route passes through multiple administrative domains and there is no single control center able to perform the configuration of all devices involved. TeraPaths achieves this goal by directly configuring end-site LAN domains and interfacing with OSCARS for WAN domains.

The TeraPaths system [14] has a fully distributed, layered architecture (see Figure 1) and interacts with the network from the perspective of end-site users/applications. The local network of each participating end-site is under the control of an End-Site Domain Controller module (ESDC). The site's network devices are configured by one or more Network Device Controller modules (NDCs). NDCs play the role of a "virtual network engineer" in the sense that they securely expose a very specific set of device configuration commands to the ESDC module. The core of each TeraPaths site service is a Distributed Services Module (DSM). The DSM has the role of coordinating all network domains along the route between two end hosts (each host belonging to a different end-site) to timely configure the necessary network segments that establish a desired host-to-host path. The DSM interfaces with the ESDCs of its own and other remote sites to configure the path within the end-site LANs (direct control), and furthermore interfaces with WAN controlling software (OSCARS Inter-Domain Controllers - IDCs) to bring up the necessary path segments through WAN domains (indirect control).

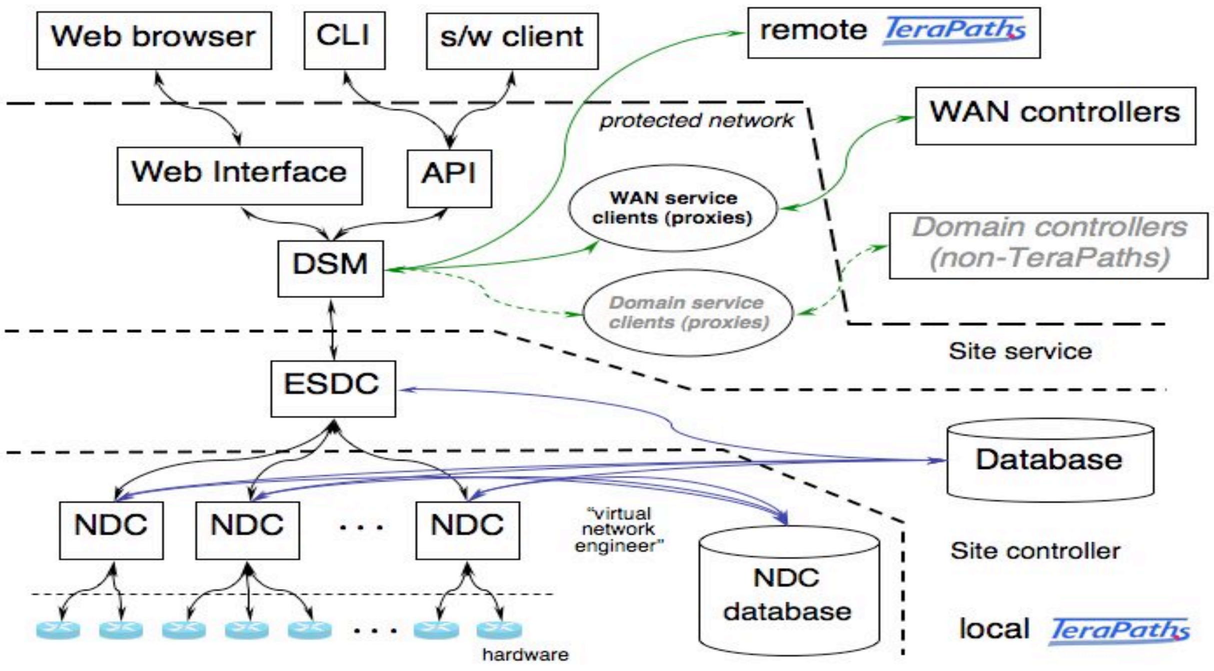

Figure 1: The TeraPaths site architecture.

3.3 On-demand Secure Circuits and Advance Reservation System (OSCARS)

OSCARS [8] is a guaranteed bandwidth provisioning system for DOE's ESnet standard IP network and advanced Science Data Network (SDN). It was designed specifically to meet the requirements of data-intensive scientific applications through dynamically provisioned virtual paths with guaranteed QoS, and has effectively demonstrated that an end site can reserve bandwidth within ESnet to accommodate deadline-based scheduling. OSCARS initially provided guaranteed bandwidth circuits within ESnet in the form of MPLS tunnels (layer 3). Through the collaboration between ESnet and Internet2, the system evolved into a more general Inter-Domain Controller (IDC) which provides not only MPLS tunnels within ESnet, but also guaranteed bandwidth layer 2 circuits within and between ESnet's Science Data Network (SDN) and Internet2's Dynamic Circuit Network (DCN) [15].

4 HIGH-LEVEL ARCHITECTURE OF STORNET

StorNet is a versatile, end-to-end, performance-guaranteed data transfer system based on an existing storage resource management system (BeStMan), and a tool for providing virtual paths with bandwidth guarantees (TeraPaths). By integrating and optimizing storage and network bandwidth provisioning and storage space reservation together in an end-to-end manner, StorNet provides data transfer applications with guaranteed and predictable QoS. At the core of the project is a flexible protocol that enables BeStMan to interoperate with TeraPaths instances, which in turn interoperate with OSCARS Inter-Domain Controllers (IDCs), and negotiate the reservations of virtual network paths with guaranteed QoS parameters spanning multiple network domains. Subsequently, BeStMan uses the established virtual paths to perform data transfers with increased reliability and predictability in terms of bandwidth utilization and transfer duration. The protocol also aims to provide users and applications with capabilities to detect and recover from failures, not only within the network, e.g. due to failed connections, but also within the storage sites, e.g., due to malfunctioning hardware/software.

In the StorNet framework, we follow a layered approach to compose the functionality of multiple systems and achieve the overall goal of efficient, high-performance data transfers, as shown in figure 2. The framework comprises four layers: 1) the data plane consists of disk and/or tape storage systems, site LANs and WAN backbone; 2) the control plane includes storage resource schedulers, LAN QoS provisioning and circuit utilization systems, and WAN backbone bandwidth and circuit provisioning systems based on MPLS/GMPLS traffic engineering; 3) the management plane monitors resource functionality and performance, diagnoses faults and coordinates fault recovery attempts; and 4) the service plane which reserves resources and exposes the functionality of individual systems, while also providing authentication and authorization. The service plane interacts with the control plane to dedicate data plane resources to meet data transfer and storage requirements based on application requests. In Figure 2, the horizontal direction represents end-to-end functionality. Each plane's components serve an end-to-end goal. The data plane is the vehicle of a data transfer between end site storage systems via the interconnecting network. The management plane provides an "enterprise" view of performance metrics, which can be used for diagnosing problems. The control plane enacts the service plane's directions into system configurations that physically provision the required resources. The service plane negotiates the reservation of resources across domains so that an application request can be accommodated. The vertical direction represents system integration. BeStMan schedules and coordinates access to storage systems, storage bandwidth and data transfers at the behest of an application request. TeraPaths schedules end-site LAN bandwidth and configures LAN devices to dedicate this bandwidth to specific network traffic. OSCARS schedules and provisions bandwidth in the WAN domains that interconnect the end-sites.

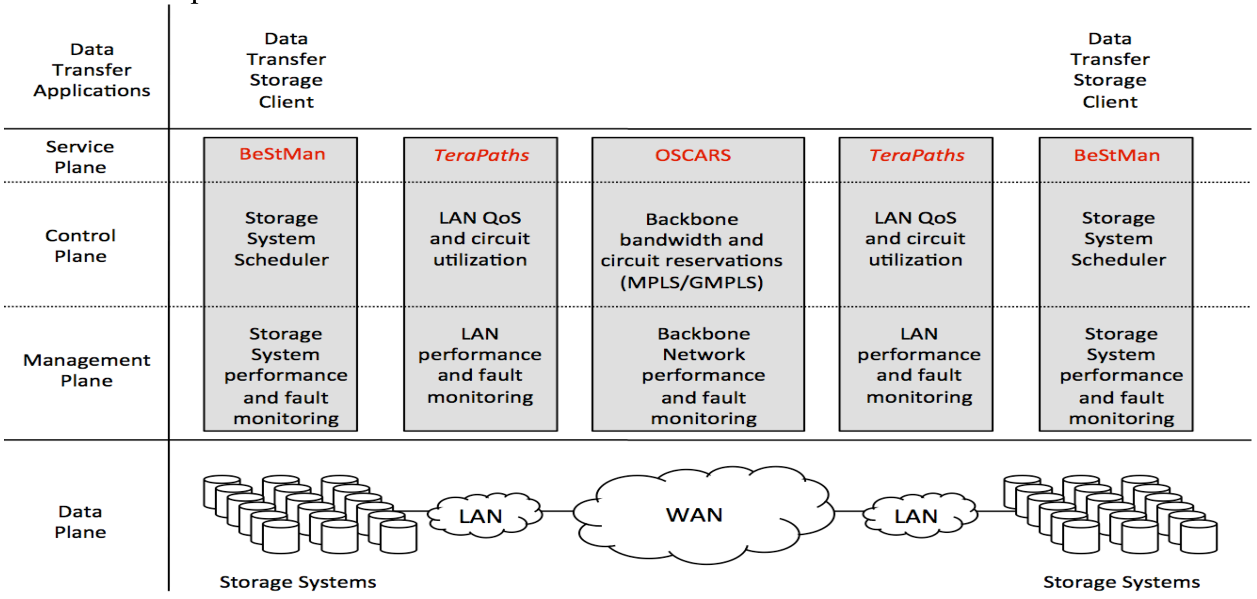
Data Transfer Applications
Data Transfer Storage Client
Data Transfer Storage Client
Service Plane
BeStMan
TeraPaths
OSCARS
TeraPaths
BeStMan
Control Plane
Storage System Scheduler
LAN QoS and circuit utilization
Backbone bandwidth and circuit reservations (MPLS/GMPLS)
LAN QoS and circuit utilization
Storage System Scheduler
Management Plane
Storage System performance and fault monitoring
LAN performance and fault monitoring
Backbone Network performance and fault monitoring
LAN performance and fault monitoring
Storage System performance and fault monitoring
Data Plane
LAN
WAN
LAN
Storage Systems
Storage Systems

Figure 2: Efficient, high performance data transfers require interaction and cooperation among components within a set of conceptual functionality planes.

System interactions take place at the service plane layer. Triggered by a client's request, end-site BeStMans first coordinate between themselves to reserve storage space, and decide the parameter space, in terms of maximum bandwidth and maximum time to completion, that satisfies the request. This parameter space is then passed to TeraPaths as a request for network bandwidth reservation. TeraPaths instances coordinate between themselves to match the BeStMan request to LAN resource availability. Subsequently, TeraPaths generates corresponding requests for WAN bandwidth reservations and submits them to OSCARS. When multiple WAN domains are involved, OSCARS IDCs coordinate in a daisy-chain manner to establish the path interconnecting the end-sites; however, this is done transparently, i.e., TeraPaths only interacts with one IDC (see Figure 3).

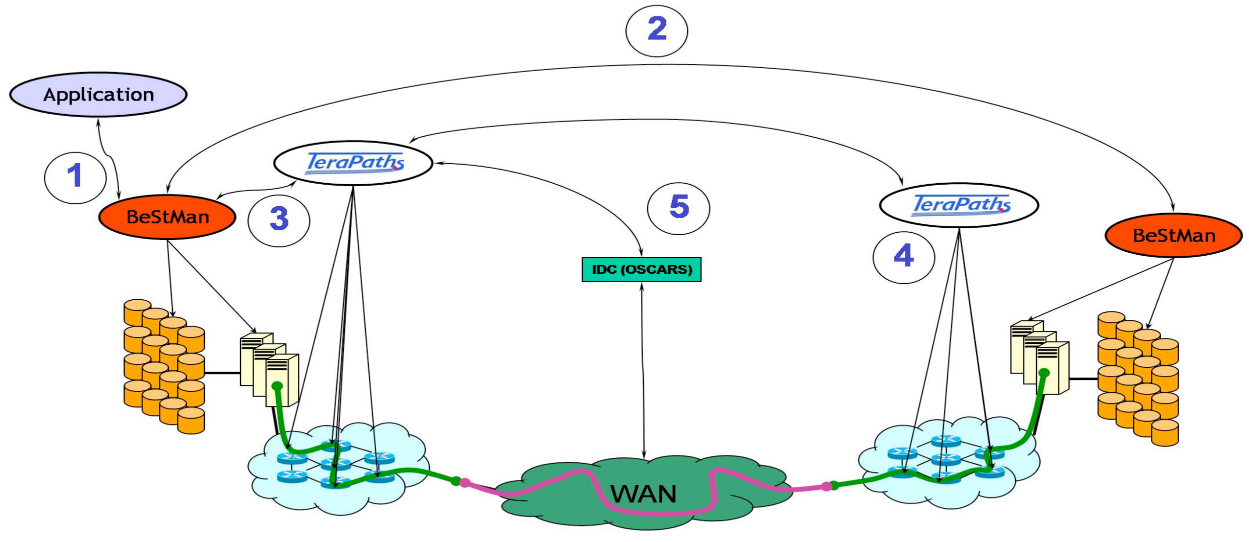

Figure 3: StorNet workflow

5 FUNCTIONALITY DESIGN

The focus of this section is on the design and enhancements of the components in the service plane, especially the communication and coordination of bandwidth between BeStMan, TeraPaths, and OSCARS.

5.1 Resource Co-Scheduling

In extreme scale science environments, the resources located at each site, such as computing power and storage space, have to be allocated jointly with network resources to achieve a cost-effective and reliable data transfer and sustain the desired overall performance of distributed tasks. For instance, a site with rich storage resources may not be a good candidate for data backup if its network connectivity with other sites is poor. In such an environment where users share and compete for resources, it is critical to achieve efficient resource utilization with suitable co-scheduling schemes. StorNet addresses a general Resource Co-Scheduling (RCS) problem: given a set of limited resources of different types and a variety of requests from data-intensive applications, determine how to optimally allocate and schedule the resources required by each application. For example, consider an application performing a time-constrained end-to-end data transfer. To reliably transfer data from source storage to destination storage over the network at expected rates and meet its deadline, this application may simultaneously require a bandwidth-guaranteed network circuit and a number of dedicated CPUs and disk storages. We therefore need to jointly allocate and co-schedule all required types of resources.

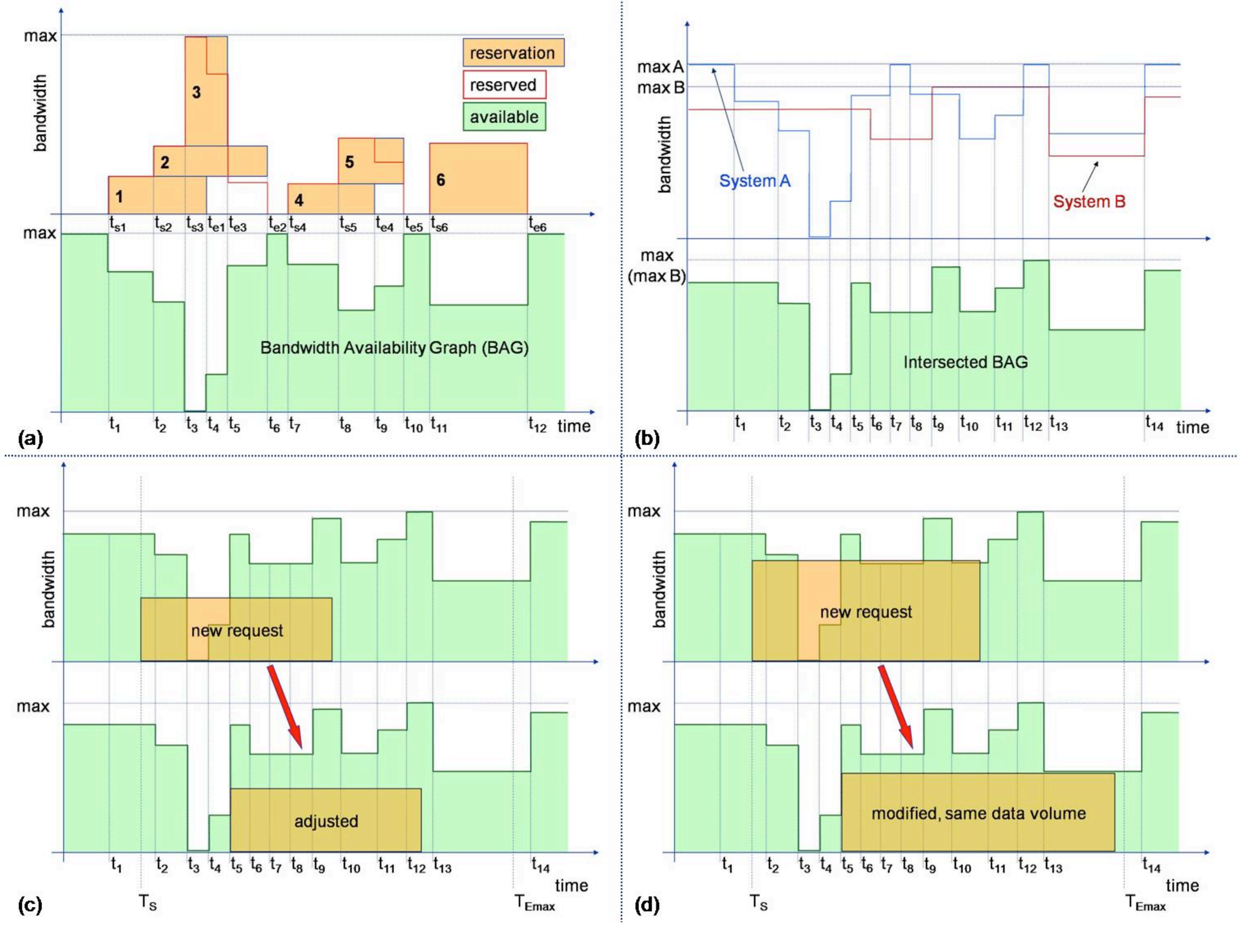

Figure 4: Bandwidth Availability Graphs

In the context of the StorNet project, we have developed for this purpose an analytical model of resource co-scheduling, based on the concept of an end-to-end Bandwidth Availability Graph (BAG). We assume that the utilization of each resource type can be scheduled by advance reservations with specific start and end time and constant bandwidth allocation for their duration. The bandwidth allocation of such a set of reservations can be aggregated and subsequently subtracted from the maximum bandwidth availability for the overall time period to yield the BAG for the resource of interest (see Figure 4a). The maximum availability can vary with time, but typically can be considered constant, at least within known time intervals. As such, a BAG is a step function. For a storage system, for example, the maximum availability could be the total achievable transfer rate, and for a network domain the maximum achievable bandwidth. Individual BAGs can be intersected to express the minimum availability of the initial BAGs at any given time, which provides the overall availability of resources across any number of systems (see Figure 4b). The intersection of all BAGs of source and destination storage systems and interconnecting network domains yields the end-to-end BAG.

Subsequently, a new request for reserving that resource can be represented by a rectangle (see Figure 4c). If the rectangle fits into the overall BAG, then the request can be satisfied. A request may be flexible in terms of start time, duration, and/or bandwidth so that the rectangle can be modified to fit into the graph (see Figure 4d). In the latter case, the area of the rectangle represents the total volume of data to be transferred, and any modification to the start time, duration, and/or bandwidth must result in a rectangle with the same area as the initial one. The objective of fitting the request rectangle is to obtain a solution (i.e., a set of reservation parameters acceptable across all systems) that optimally satisfies the request. As optimal, we define a solution that satisfies the request according to the requestor's preferences. We have primarily considered the cases of shortest transfer duration and earliest finish time.

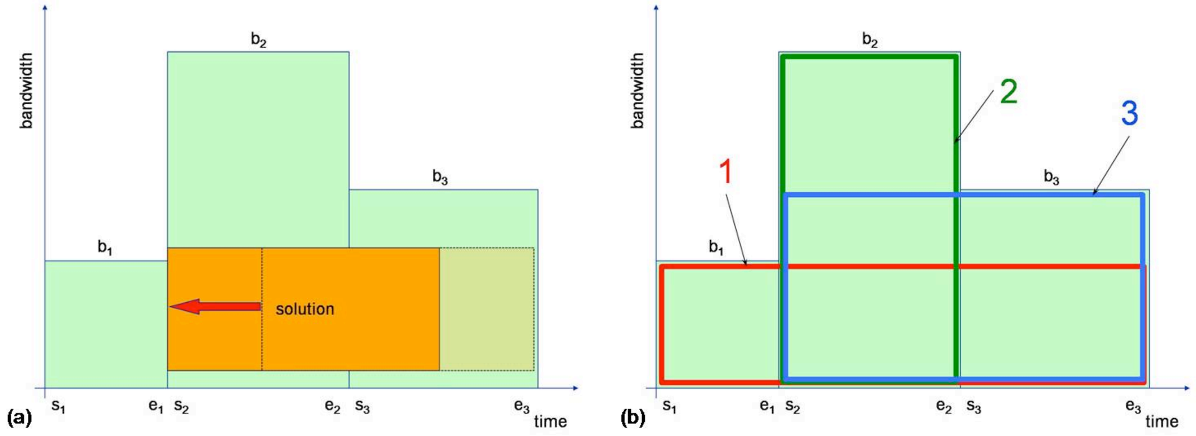

Figure 5: Finding a solution

Fitting the request rectangle can be approached as a variation of the problem of finding the largest rectangle under a histogram with *n* adjacent rectangles, which can be solved in *O(n)* time [16]. More specifically, a BAG is represented by a sequence of *n* windows [s_i,e_i], where *s* is the start time and *e* the end time, each with constant bandwidth b_i. Firstly, we observe that an optimal solution can always start from some s_i. If an optimal solution starts elsewhere within a window, we can move the whole solution (rectangle) to the left until it starts from an s_i point; the solution will have identical duration but earlier finish time (see Figure 5a). Secondly, we can obtain exactly *n* rectangles {*start*[*i*],*end*[*i*]} with bandwidth b_i, where *start* and *end* denote the start and end time of the largest rectangle containing window *i* (see Figure 5b). We can then fit a request with a given same data volume in these *n* "largest" rectangles and choose the one with the shortest duration or the earliest finish time, depending on the request's preferences.

Based on the above observations, we can obtain the optimal solution in *O(n)* time using one of the known algorithms for the problem of finding the largest rectangle under a histogram [17]. In other words, given a BAG, it will take *O(n)* time to obtain *start*[*i*] and *end*[*i*] of the largest rectangle corresponding to every window in the BAG. An example of such an algorithm that uses a stack is shown in figure 6. The idea is that the stack holds a series of windows that have increasing heights; windows are enumerated from left to right.

StorNet approaches schedule negotiation in a top-down direction across systems, i.e., narrowing down the solution space is first performed at the BeStMan level, then at the TeraPaths level, and finally at the OSCARS level. This is done for two major reasons: firstly, because the availability of resources within each system must take into account the aspects of system-wide policies and user privileges; and secondly, because the amount of effort for figuring out a solution is reduced. Although BAG intersection is commutative, using a separate scheduling component is not feasible, as it would require systems belonging to different administrative domains to reveal non-public information in a bottom-up manner. For example, it is not expected that OSCARS will reveal to TeraPaths all schedule information pertaining to a network path of interest. In contrast, in the top-down direction one system passes to another only non-sensitive information required to obtain solutions satisfying the original request. An additional incentive for minimizing the candidate solution set is that the current implementation of OSCARS does not support negotiation with BAGs and candidate solutions have to be tried one-by-one in a time-expensive trial-and-error manner. Reducing the number of options based on previous constraints reduces the search space and therefore the interaction with OSCARS.

```
Append a dummy window with zero height and zero width to the list of windows
Merge successive windows of equal bandwidth (enforce $b_q$ <> $b_p$, $q=p+1$, $p=1..n$)
For each window $i$ from left to right {
 If (stack.empty() == true or bandwidth of current window $b_i$ > stack.top().bandwidth) {
     push window $i$ , set start[$i$] = $s_i$
 } else {
     repeat {
             pop window $j$, set end[$j$] = $s_i$ , set $k$ = $j$
           } until (stack.empty() == true or $b_i$ >= stack.top().bandwidth)
     if (stack.empty() == false and stack.top().bandwidth == $b_i$) {
         set start[$i$] = start[stack.top()]
     } else {
         set start[$i$] = start[$k$], where $k$ is the last popped window
     }
     push window $i$
 }
}
```

Figure 6: Pseudo-code for stack-based largest rectangle algorithm

5.2 BeStMan Functionality Enhancements

The data transfer protocols currently used by BeStMan, such as GridFTP, assume best-effort IP networks, and improve performance with a large number of TCP streams for long, round-trip connections. Fairness and efficiency are adversely affected by such a brute force data transfer method. The primary goal of StorNet is to provide data transfers with QoS guarantees and to move away from the best-effort data transfer paradigm that does not provide delivery time assurance. To support network and storage co-scheduling, the existing data transfer module in BeStMan is being extended to reserve end-to-end network bandwidth and intelligently optimize storage space and network bandwidth allocation, thus increasing transfer reliability. This extension reduces the "impedance mismatch" between end user data transfer applications, storage, and the network. In order to keep track of bandwidth reservations and commitments, BeStMan is also being enhanced with a backend database service. This will provide persistent store for tracking user requests, storage space allocations, and bandwidth allocations.

The enhanced BeStMan is designed to achieve the best solution for user requests. Users can specify whether they prefer earlier time solutions or shortest transfer, and they provide BeStMan with a desired time of completion. The BeStMan at the target site (pulling the data) also needs to have the logic to communicate with the BeStMan at the source site to find out what is its bandwidth availability. The BeStMan at the source site returns the availability "graph" for the requested period of interest (i.e. till maximum time), in the form of a sequence of windows. The BeStMan at the target site then finds a common schedule, and provides that to TeraPaths. The API for BeStMan-TeraPaths interaction is described in section 5.4.

5.3 TeraPaths Functionality Enhancements

To accommodate the functionality required for StorNet, TeraPaths is being enhanced along two main directions: interaction with BeStMan and core extensions to support negotiation between end-sites and with OSCARS. Communication and coordination with BeStMan is supported by a BeStMan-to-TeraPaths (StorNet) API module that interprets and validates BeStMan requests and passes them along to the main system through the TeraPaths API. The former API is essentially a wrapper of the latter. The choice of using an API wrapper allows us to standardize and simplify the interaction between BeStMan and TeraPaths, so that future revision of one API will not necessarily affect the other API or BeStMan's clients. Core extensions to TeraPaths are primarily necessary for supporting negotiation between end-site instances through BAGs, calculation of solutions spaces by fitting requests into intersected BAGs, and negotiation with OSCARS by applying a trial-and-error approach on the set of candidate solutions obtained from the fitting process.

5.4 BeStMan-TeraPaths Web-Services Interface Design

We designed a BeStMan-TeraPaths web-service interface to describe the functions necessary for a BeStMan server to request network bandwidth from TeraPaths service. The goal of the API is to enable BeStMan to negotiate bandwidth with TeraPaths. The important functionalities reflected in the interface are bandwidth reservation, commitment, modification, and cancelation. The interface also includes status check and time-out extension. Necessary information, such as data volume, source and target resource availability, resource time frames, and other attributes, is provided to TeraPaths when requesting network bandwidth. In Figure 3, we showed the sequence of communications between the components. In Step (1), which involves getting the request from the application, a list of files or a directory are provided as well as source and destination information. In addition, a window of desired start time and maximum completion time is provided. In step (2), the source and target BeStMan servers communicate with each other to reserve storage space and to determine the maximum bandwidth they can both use during the requested window. Once this is determined, BeStMan communicates with TeraPaths in step (3), and provides a sequence (start time, end time) of non-overlapping windows, and maximum bandwidth for each. Based on this information, TeraPaths schedules its local area resources, and then negotiates with OSCARS for wide area resources. The communications in steps (4) and (5) are internal to TeraPaths, and only the resulting reserved window is communicated to BeStMan. A typical scenario is that BeStMan first tries to make a temporary network bandwidth reservation. If such a reservation is possible, TeraPaths returns a request token, along with an expiration time and available windows for the available resources. Once BeStMan determines that it can work with the result from TeraPaths, it commits the reservation to lock in the network resources. Otherwise, BeStMan modifies its input and submits a new request. In case of a failure, BeStMan can request Terapaths to find a window that goes beyond the max completion time (by not specifying a max completion time), and this can be returned as an alternative to the user, who may accept or reject it. This possibility was not exercised in the prototype implementation, but left as future work. However, the interface is designed to accommodate that by allowing max completion time to be unspecified (i.e. open ended).

5.5 BeStMan-TeraPaths Web-Services Interface Specification

5.5.1 General

- Dates/times
    - Dates/times are stored as long integers.

- CompletionTime
    - Measured in seconds.
    - A value that is equal to or less than zero implies no restriction on completion time.

- Bandwidth
    - Network bandwidth is measured in kbps (kilobits per second).

5.5.2 Type Definitions

<u>Underlined</u> attributes are REQUIRED. The required attributes must be parsed correctly and must give proper error messages when not supported.

- ESchedulePreference
    - Enumeration with values:
        - ANY
        - EARLIEST_COMPLETION_TIME
        - SHORTEST_TRANSFER_DURATION
    - A user can indicate preference on how the requests shall be handled.

- TRequestReference
    - Fields:
        - String <u>requestId</u>
        - String userId

- This structure is used to refer to a request to TeraPaths. It is the input of the request handling functions like commitRequest(), cancelReqeust() etc.
- userId is a user-supplied id for authorization by TeraPaths.

▪ **TReserveRequest**

- Fields:
    - String userId
    - TBandwidthRequestParameters <u>desiredValues</u>
    - long timeout
    - ESchedulePreference schedulePreference
- This structure is used to construct the reservation request to TeraPaths. It is the input of the function reserveRequest().
- userId is a user-supplied id for authorization by TeraPaths. The transfer related parameters are contained in the structure TBandwidthRequestParameters.
- The input value "timeout" is for a client to specify the time (subject to configuration limits) that TeraPaths will wait for a temporary reservation to be committed.

▪ **TResponse**

- Fields:
    - TReturnStatus <u>status</u>
    - String requestId
    - long requestExpirationTime
    - TBandwidthRequestParameters[] arrayOfReservationData
- This structure is used by TeraPaths to respond to a reservation request. It is the output of the function cancelRequest(), commitRequest(), modifyRequest(), reserveRequest() and extendTimeoutRequest().
- TeraPaths server must return a status to all the function calls.
- Upon success for a reserveRequest(), TeraPaths must return:
    - The values that are reserved for this client in arrayOfReservationData,
    - A request id "requestId" if involved, so the client can later cancel this reservation (through BeStMan) if desired.
- If the request cannot be processed right away, TeraPaths must return
    - A request id "rid" so the client can use it to check status (through BeStMan).

▪ **TReturnStatus**

- Fields:
    - EStatusCode <u>code</u>
    - String explanation
- A return status is used for the status of a reservation request.
- Return status consists of a code, which is described by TStatusCode, and the explanation.

▪ **EStatusCode**

- Enumeration with values:
    - ACTIVATED
    - CANCELLED

- ◦EXPIRED
- ◦FAILURE
- ◦NO_AUTHORIZATION,
- ◦NO_SOLUTION
- ◦NO_SUCH_REQUEST,
- ◦NOT_SUPPORTED,
- ◦QUEUED,
- ◦RESERVED
- ◦TEMPORARY
- ◦TIMED_OUT

- If a reservation is made within the client's desired parameters, the status code is TEMPORARY. A request id is expected in this case. Client is expected to call commitRequest() within a given time to confirm with the reservation. Once committed successfully, the status is set to RESERVED. If commitRequest was not called timely, the status of the request will be TIMED_OUT.
- Once the status is RESERVED, client is advised to check status again, until, the status becomes ACTIVATED. This indicates the bandwidth reservation is materialized with the underlying system. Client can now use the bandwidth.
- If a reservation can not be processed right away, the status code shall set to be QUEUED and a request id is provided. Clients are expected to pull status periodically until a final status is reached.
- NO_SUCH_REQUEST is returned from getReservationStatus() for a request id that is not currently active (QUEUED or RESERVED).

▪ **TBAGInfo**
- Fields:
  - ◦long segmentBandwidth
  - ◦long segmentEndTime
  - ◦long segmentStartTime

▪ **TFlowInfo**
- Fields:
  - ◦String key
  - ◦String value

▪ **TBandwidthRequestParameters**
- Fields:
  - ◦TBagInfo[] bagInfo
  - ◦long volumn
  - ◦TFlowInfo[] flowInfo
- A client provides desired begin time (optional), volume in MB (required), max bandwidth (required), max completion time (optional) to TeraPaths along with source and destination hosts to get a reservation for file transfer.
- Bandwidth is measured in kbps (kilobits per second).
- maxCompletionTime is measured in seconds.

- **TBandwidthResponseParameters**
  - Fields:
    - long availableBandwidth
    - long beginTime
    - long endTime
    - TFlowInfo[] flowInfo
    - String reservationId
    - String reservationStatus

- **TModifyRequest**
  - Fields:
    - String requestId
    - String uid
    - TReservationModifcationSet[] modificationSet

- **TReservationModificationSet**
  - Fields:
    - String reservationId
    - TReservationModificationParameters[] modificationParameters

- **TReservationModificationParameters**
  - Fields:
    - String modificationOperation
    - TModificationInfo modificationInfo

- **TModificationInfo**
  - Fields:
    - String key
    - String value

5.5.3 Function Definition

- **reserveRequest**
  - in:
    - TReserveRequest <u>reserveRequest</u>
  - out:
    - TResponse <u>reserveResponse</u>

  - The output of this function contains: status code (mandatory), request id (if reservation were made) and reserved values (if reservation were made.)
  - The status code NOT_SUPPORTED is not applicable here.
  - The request id is expected if the status codes returned are one of QUEUED and TEPORARY.

- **commitRequest**
  - in:
    - TRequestReference <u>request</u>

- out:
    - TResponse response

- Client needs to call this function to confirm the reservation of the solution provided by the outcome of reserveRequest() to TeraPaths.

- **modifyRequest**
    - in:
        - TModifyRequest modifyRequest
    - out:
        - TResponse response

    - The client uses this function to make changes to the reserved request.

- **extendTimeOutRequest**
    - in:
        - TRequestReference request
    - out:
        - TResponse response

    - If unable to call commitRequest() within the time specified from the outcome of resreveRequest(), client can use this function to extend the timeout.

- **statusRequest**
    - in:
        - TRequestReference request
    - out:
        - TResponse status response

    - The status of a reservation, as with the output of the reserveRequest() function, contains: status code (mandatory), request id and reserved values (if a reservation was made).
    - All status codes defined in TStatusCode can be returned. If a reservation with the specified rid has expired, FAILURE will be returned along with an explanation.

- **cancelRequest**
    - in:
        - TRequestReference request
    - out:
        - TResponse status response

    - Client calls this function to cancel a reservation with the specified rid (through BeSMan).
    - If there is a valid reservation with this rid, TeraPaths is expected to honor the cancelation.
    - If the request with this rid is still queued, TeraPaths shall stop processing the request.

- In other cases, for example, when the request does not exist at all, or the reservation with this rid has already expired, TeraPaths takes no action.

6 SUPERCOMPUTING 2010 DEMONSTRATION

The StorNet functionality was demonstrated at SuperComputing 2010 using an early prototype implementation of the architecture. The demonstration included actual 10 GB file transfers between BNL and University of Michigan using the current TeraPaths testbed. The testbed setup is shown in figure 7, and the demonstration results in figure 8. Heavy interference traffic allowed best effort transfers to only reach roughly 8 MB/s (8c), while transfers with StorNet could be tuned to desired levels of bandwidth (8a,b,d) unaffected by network congestion. Because of the low performance of best effort transfers, a smaller file of 1 GB size was used to save demo time.

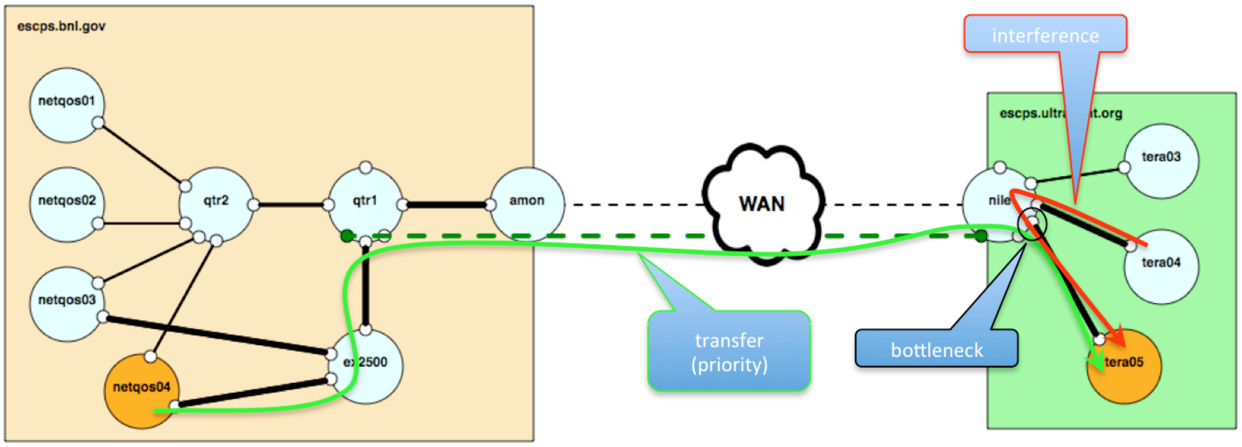

Figure 7: StorNet SC’10 demo setup: severe congestion imposed on the router-side interface of host tera05 (red arrow).

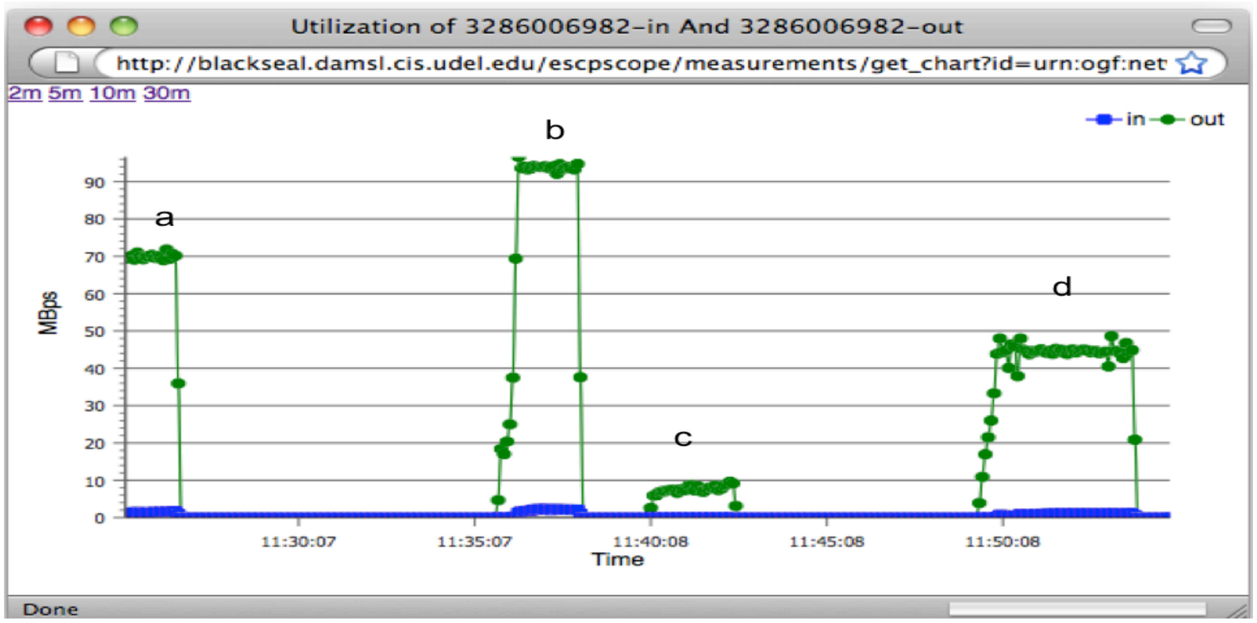

Figure 8: StorNet SC’10 demo: network interface traffic at sending host: (a), (b), (d) transfer of a 10 GB file with 70MB/s (partially displayed), 95 MB/s, 45 MB/s reservations respectively; (c) transfer of a 1 GB file w/o reservation.

7 Summary

Effective and robust data transfer is essential to current day scientific applications, and is a major concern for future scientific work as the volumes of data collected and shared grow exponentially. In order to address this problem, two aspects need to be supported: 1) ways to reserve and guarantee bandwidth in network and storage, and 2) ways to coordinate and synchronize bandwidth reservations in all components from source to destination. Thus, storage systems need to be instrumented to support such bandwidth reservations, and coordinate with local area and wide area network bandwidth provisioning. For this purpose, there need to exist components that can control the bandwidth reservations, provision the bandwidth, and ensure that the allocated bandwidth is used effectively. In this work, we take advantage of existing storage and local network middleware technologies (called BeStMan and TeraPaths, respectively) to pursue this goal. Bandwidth provisioning in the WAN is realized by having TeraPaths negotiate with ESnet's OSCARS provisioning system. This coordinated approach is achieved by enhancing the existing middleware systems with APIs for negotiating end-to-end bandwidth reservations and obtaining monitoring information. The design of such APIs has been described.

8 Future Work

With data volume now growing exponentially, the StorNet is working to improve various aspects of the system by standardizing APIs and generalizing implementations from the prototypes, which should better meet the needs of applications and scientific communities. It is unclear how the community's needs will be met should our work end with this funding period.
Should financial support continue, a list of goals is proposed for the next a couple years:

1. Develop a general-purpose framework for end-to-end co-scheduling enabling any components with the APIs;
2. Provide the wide-area co-scheduler with an availability graph, to be flexible as to the capability of the WAN scheduler.

The main outcome of this future work is a middleware infrastructure with well-defined interfaces, where various coordinated components can be plugged into the framework. Once the reservation is made, the client can provide the reservation information to the data transfer coordinator optimizing the performance to take full advantage of the reserved end-to-end bandwidth.

Acknowlegements

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 (LBNL), and DE-AC02-98CH10886 (BNL). We also thank Shawn McKee from the University of Michigan for his help with running StorNet tests between BNL and University of Michigan.

References

[1] LSST: http://www.lsst.org/lsst

[2] LHC – The Large Hadron Collider: http://lhc.web.cern.ch/lhc/
[3] Spallation Neutron Source (SNS): http://neutrons.ornl.gov/aboutsns/aboutsns.shtml
[4] Advanced Photon Source (APS): http://www.aps.anl.gov/
[5] Relativistic Heavy Ion Collider (RHIC): http://www.bnl.gov/rhic
[6] Berkeley Storage Manager: http://sdm.lbl.gov/bestman/
[7] D. Yu and D. Katramatos, TeraPaths: http://www.terapaths.org
[8] OSCARS: https://oscars.es.net/OSCARS/docs/
[9] Shoshani, A., Sim, A., and Gu, J.: Storage Resource Managers: Middleware Components for Grid Storage. In: Nineteenth IEEE Symposium on Mass Storage Systems, 2002
[10] Shoshani, A., Sim, A., and Gu, J.: Storage Resource Managers: Essential Components for the Grid. Chapter in book: Grid Resource Management: State of the Art and Future Trends, Edited by J. Nabrzyski, J. M. Schopf, and J. Weglarz, Kluwer Academic Publishers, 2003
[11] A. Sim, A. Shoshani, F. Donno, J. Jensen: Storage Resource Manager Interface Specification V2.2 Implementations Experience Report. GFD.154, Open Grid Forum, Aug. 2009
[12] A. Sim, A. Shoshani (Editors): The Storage Resource Manager Interface Specification Version 2.2. GFD.129, Open Grid Forum, Document in Full Recommendation, Feb. 2008
[13] Lana Abadie, Paolo Badino, Jean-Philippe Baud, Arie Shoshani, Alex Sim, et al.: Storage Resource Manager version 2.2: design, implementation, and testing experience. In: Conference for Computing in High Energy and Nuclear Physics, 2007
[14] Dimitrios Katramatos, Bruce Gibbard, Dantong Yu, Shawn McKee: TeraPaths: End-to-End Network Path QoS Configuration Using Cross-Domain Reservation Negotiation. In: 3rd International Conference on Broadband Communications, Networks, and Systems (BROADNETS 2006), 2006
[15] Internet2 Dynamic Circuit Network (DCN): http://www.internet2.edu/network/dc/
[16] Carroll Morgan: Chapter 21:The Largest Rectangle under a Histogram. Programming from Specifications, 2nd edition, Prentice Hall International (UK) Limited, October 1998
[17] ACM Collegiate Programming Contest 2003/2004: http://www.informatik.uni-ulm.de/acm/Locals/2003/html/judge.html

Appendix A PAPERS AND PRESENTATIONS

A.1 Papers

- Junmin Gu, Dimitrios Katramatos, Xin Liu, Vijaya Natarajan, Arie Shoshani, Alex Sim, Dantong Yu, Scott Bradley, Shawn McKee, "*StorNet: Co-Scheduling of End-to-End Bandwidth Reservation on Storage and Network Systems for High Performance Data Transfers*", Proceedings of IEEE INFOCOM HSN 2011, Shanghai China, 2011
- Junmin Gu, Dimitrios Katramatos, Xin Liu, Vijaya Natarajan, Arie Shoshani, Alex Sim, Dantong Yu, Scott Bradley, Shawn McKee, "*StorNet: Integrated Dynamic Storage and Network Resource Provisioning and Management for Automated Data Transfers*", Proceedings of the 18th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2010), Taipei Taiwan, 2010

A.2 Presentations

- Alex Sim, "*Co-Scheduling of Network and Storage with StorNet*", Open Science Grid All Hands Meeting, 3/8/2011

- Junmin Gu, Dimitrios Katramatos, Vijaya Natarajan, Arie Shoshani, Alex Sim, Dantong Yu, "*An End-to-End Provisioning and Management Framework for High Performance Data Transfers*", demonstration at ACM/IEEE Supercomputing (SC 2010), 2010
- Junmin Gu, Dimitrios Katramatos, Xin Liu, Vijaya Natarajan, Arie Shoshani, Alex Sim, Dantong Yu, "*StorNet: Co-Scheduling of Network and Storage Bandwidth*", poster at ACM/IEEE Supercomputing (SC 2009), 2009
- Arie Shoshani, Dantong Yu, "*StorNet: Co-Scheduling Network and Storage with TeraPaths and SRM*", DOE Network Research Meeting, 28-29 Sep. 2009

Appendix B PATENT AND OPEN SOURCE LICENSE

B.1 Patent

- US Patent serial no. 61/393,750. "*Co-scheduling of network resource provisioning and host-to-host bandwidth reservation on high-performance network and storage systems"*, filed on Oct. 15, 2010.

B.2 Open Source License

- BeStMan (LBNL reference number CR-2404) is under an open source license, BSD with Grant-back provision (https://sdm.lbl.gov/wiki/Software/BeStMan/License), and available on https://codeforge.lbl.gov/projects/bestman/.

- TeraPaths is in process of being released under open source license, and will be publicly available within 2012.