ADAPT: Improving Data Transfers Using Performance-Based Adaptation and Policy-Based Resource Allocation

Ann L. Chervenak[1], Alex Sim[2], Junmin Gu[2], Rob Schuler[1]

[1]University of Southern California Information Sciences Institute

[2]Lawrence Berkeley National Laboratory

Motivation

Many science applications require transfer and staging of large datasets in preparation for analysis on shared computational resources

The success of these applications requires distributed data access with improved use of available resources

ADAPT Project (Adaptive Data Access and Policy-Driven Transfers)

- Goal: improve transfer throughput and latency for large data staging operations in a resource-constrained, shared environment

Techniques:

- **Performance-based transfer parameter adaptation**
- **Policy-based resource allocation**

Outline

- ADAPT Project
- System Overview
- Implementation
- Experiments – Case 1
- Experiments – Case 2
- Summary

ADAPT Project

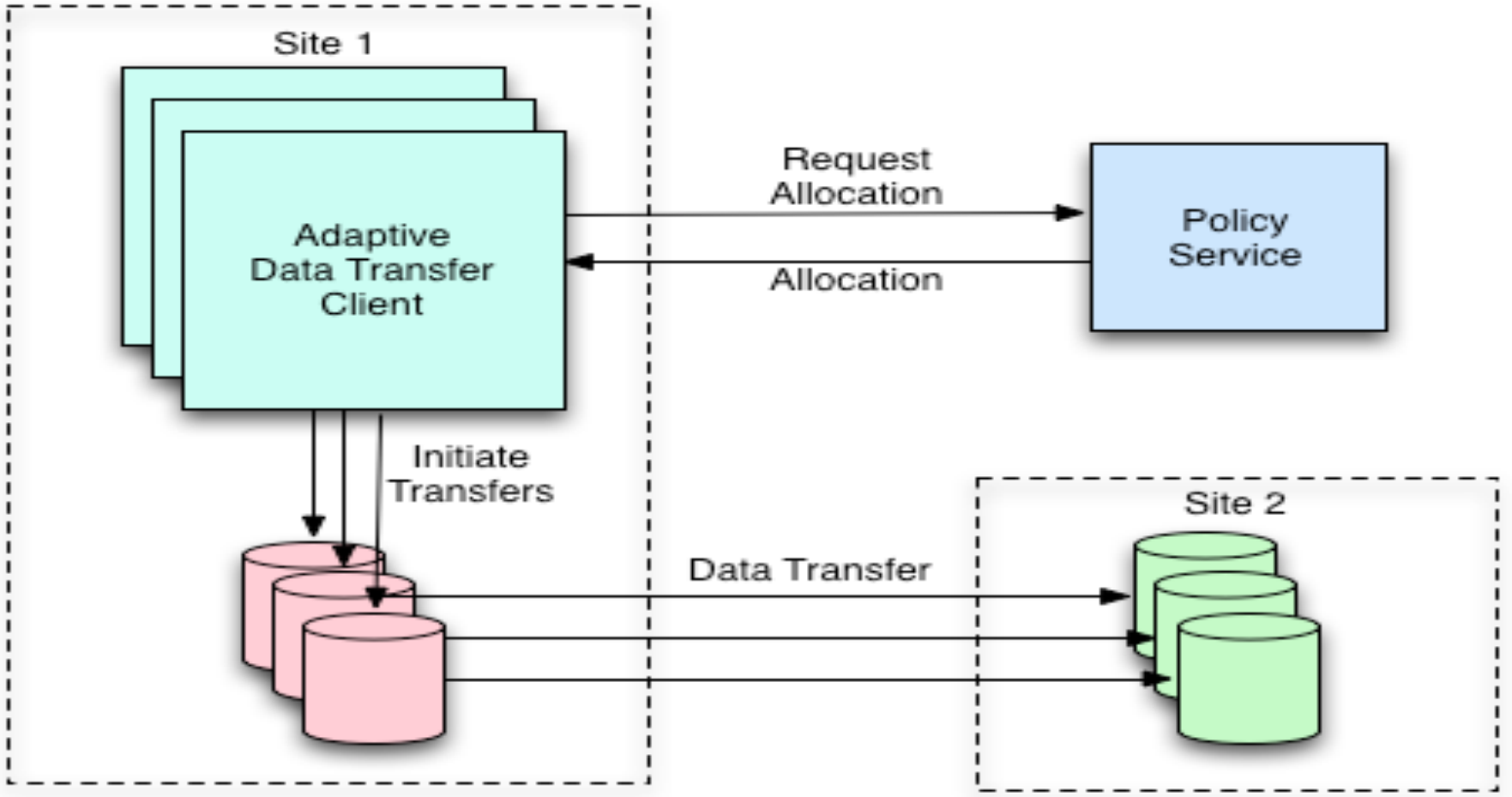
Provides performance-based transfer adaptation

- Select transfer properties based on past performance, available resources
- Adapt properties when observed performance changes due to dynamic load on storage, network, other resources

Policy-based resource allocation

- Based on Virtual Organization policies regarding resource allocation, priorities for resources, users
- Balance user requirements for data access with load on resources

System Overview



- Adaptive Data Transfer Client
- Policy Service (PS)

System Overview: Components

Policy Service (PS)

- Suggests resource allocations based on:
  - Available system resources
  - Site or Virtual Organization policies allocating resources for network data transfers, based on the several parameters
  - Input parameters:
    - Maximum concurrent streams between pair of source/ destination hosts
    - Maximum streams per client
    - Initial stream allocation (on a new request)
    - Update allocation increment

System Overview: Components

- Adaptive Data Transfer Client
  - Performs data transfers
  - Adapts transfer parameters within the resource allocation from Policy Service
  - Adapts transfer parameters based on recent transfer performance and resource availability
  - For long-running, multi-file transfers, client periodically:
    - Requests new advice from the PS
    - Adapts transfer parameters (concurrency, number of parallel streams, buffer size, etc.) based on resource performance and changes in resource availability

Implementation

Policy Service (PS)

- RESTful Web service implemented in Pyton
  - Webpy framework, CherryPy embedded HTTP server
- Service generates advice based on VO & site policies, knowledge of resources allocated
- Recommends resource allocation (streams)

Allocation policy

- Input parameters specify maximum streams between source, destination sites; max streams per client; initial stream allocation
- When a request for transfer advice arrives at PS:
  - Check resource allocation state to determine streams/bandwidth that have already been allocated
  - If unallocated streams/bandwidth remain below the threshold to satisfy the new request, then return advice to allocate requested resources
  - Once threshold is reached:
    - Refuse additional requests

Implementation

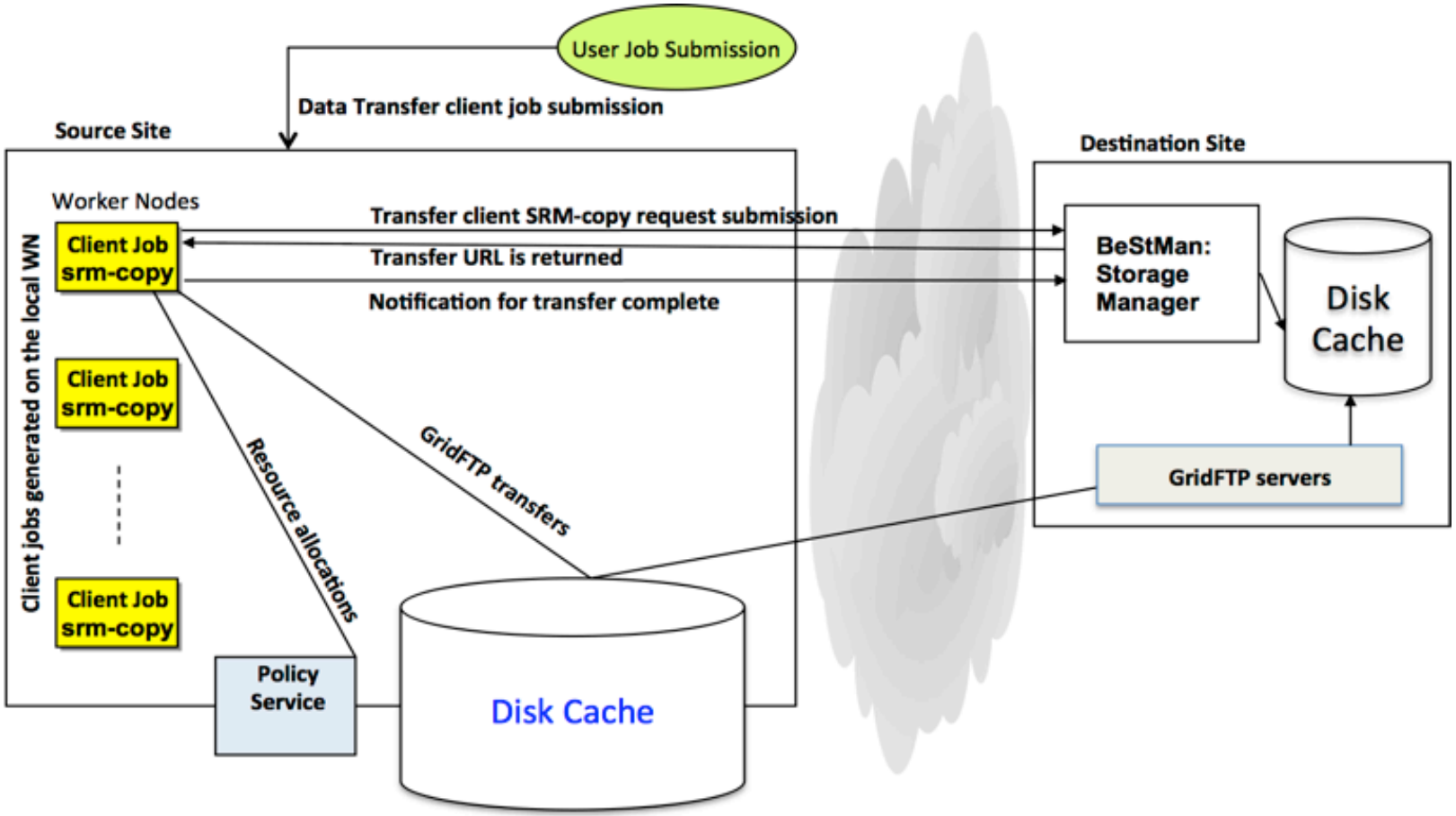
Adaptive Data Transfer Client

- Modified conventional srm-copy data movement client
- Stand-alone, command-line client implemented in Java
- Added an Adaptive Data Transfer (ADT) library
- Gradually adjust number of concurrent transfers to gain maximum aggregated throughput or up to the allocated resource limits by the PS

Experimental Set up

Users want to run analysis on an Open Science Grid site

Must first stage data from a remote data source



Experiment - Case 1

Performance of adaptive data transfers with srm-copy client

Transfer data from NERSC in Oakland, CA to Open Science Grid site at University of Nebraska at Lincoln (UNL), over 10Gbps link

8 srm-copy clients performing multi-file transfers: 260 Gbytes / 488 files

Long-running, multi-file transfers; adapt between completed transfers

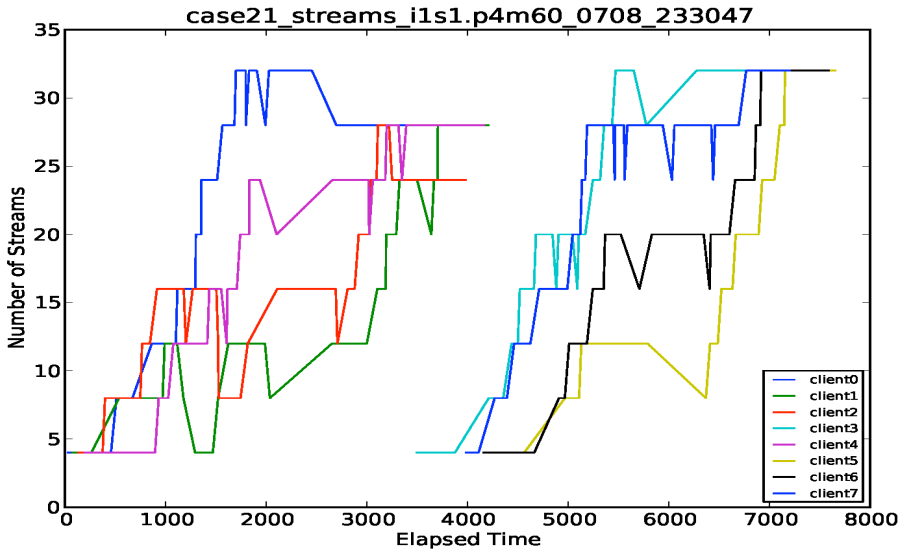
Default common parameters

Maximum total streams between source/destination	128
Number of clients	8
Maximum streams per client	32
Parallel streams per file	4
Adaptation increment/decrement	1 concurrency (4 streams)

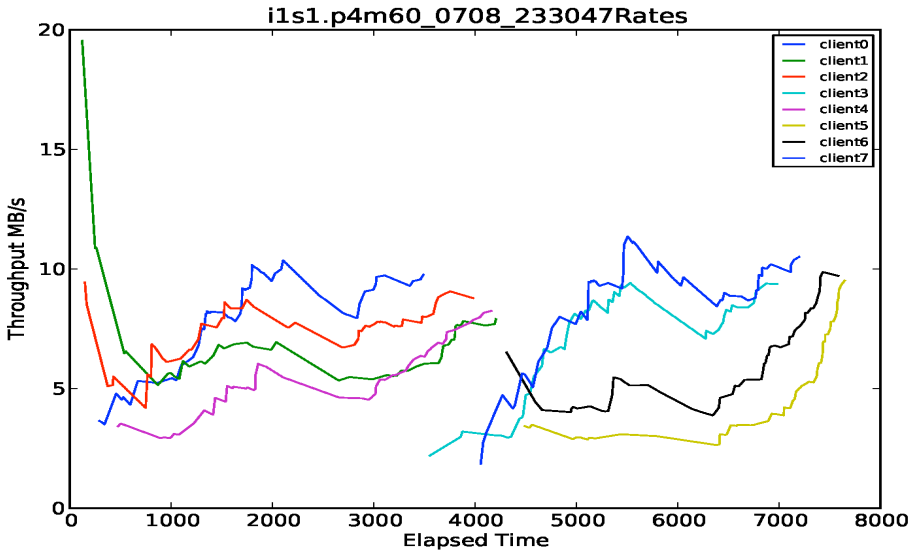
Slow Client-side Adaptation

Client Parameters	*Value*
Initial concurrency	1
Maximum concurrency	8
Adaptation delay time (update after how many transfers)	4
Policy Service Parameters	***Value***
Initial stream allocation	32
Update allocation increment	N/A

Number of streams used for slow client side adaptation



Throughput for slow client side adaptation



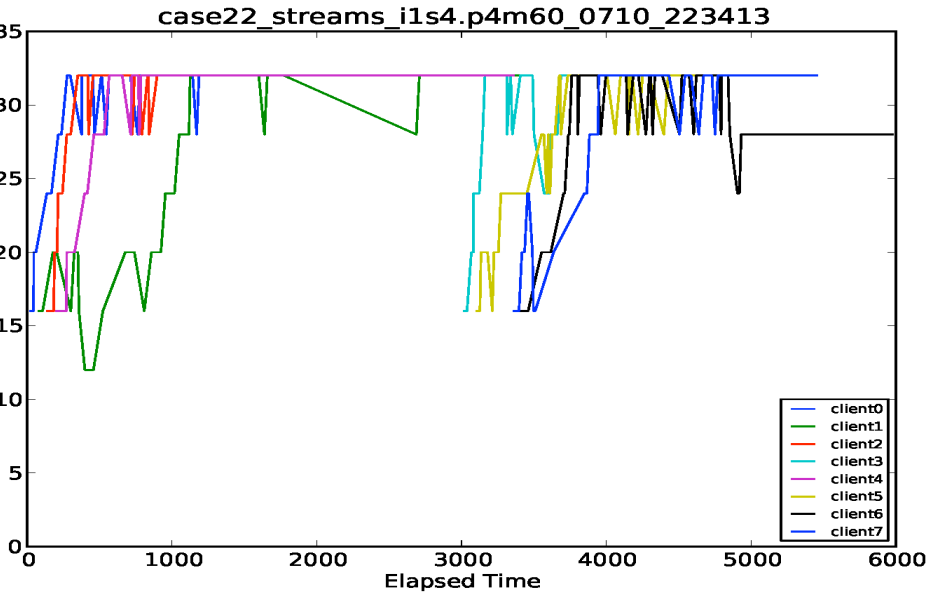
All adaptation takes place on the client side

Each client slowly adapts the concurrency of its transfers up to the allocation

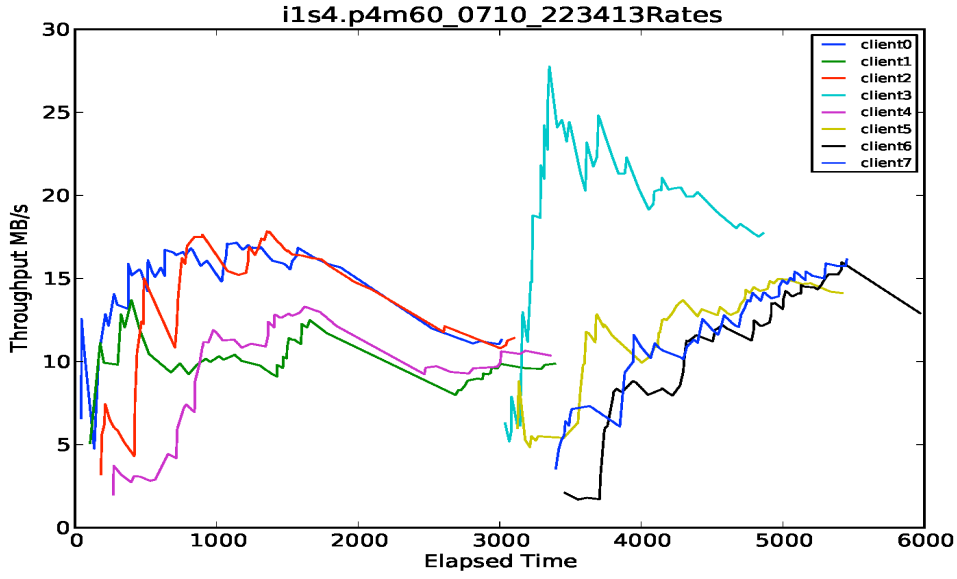
Fast Client-side Adaptation

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

Number of streams used for slow client side adaptation



Throughput for slow client side adaptation

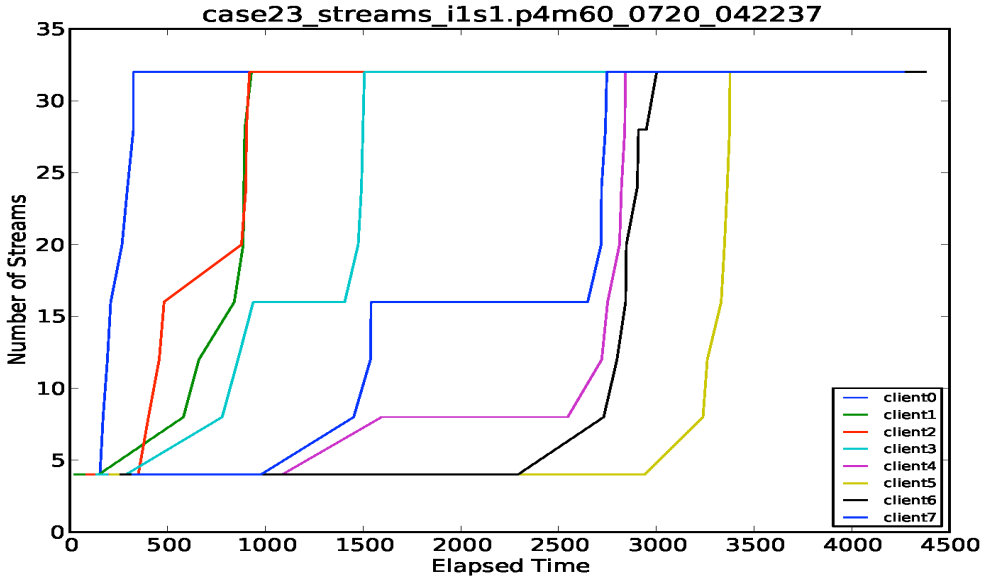


The higher initial concurrency and the faster adaptation rate have a significant effect on the performance of the transfers.

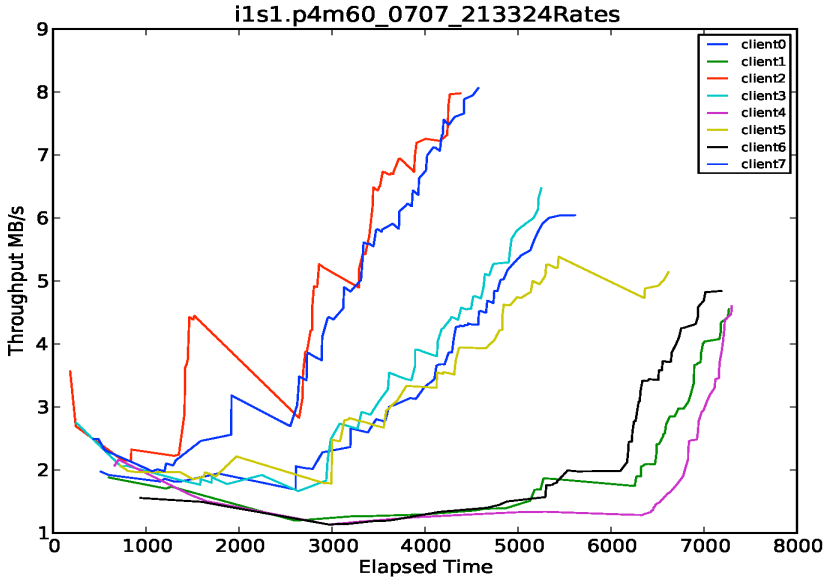
Policy-based Resource Allocation Slow Increase

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

Stream allocation for slow increases in policy service allocation



Throughput

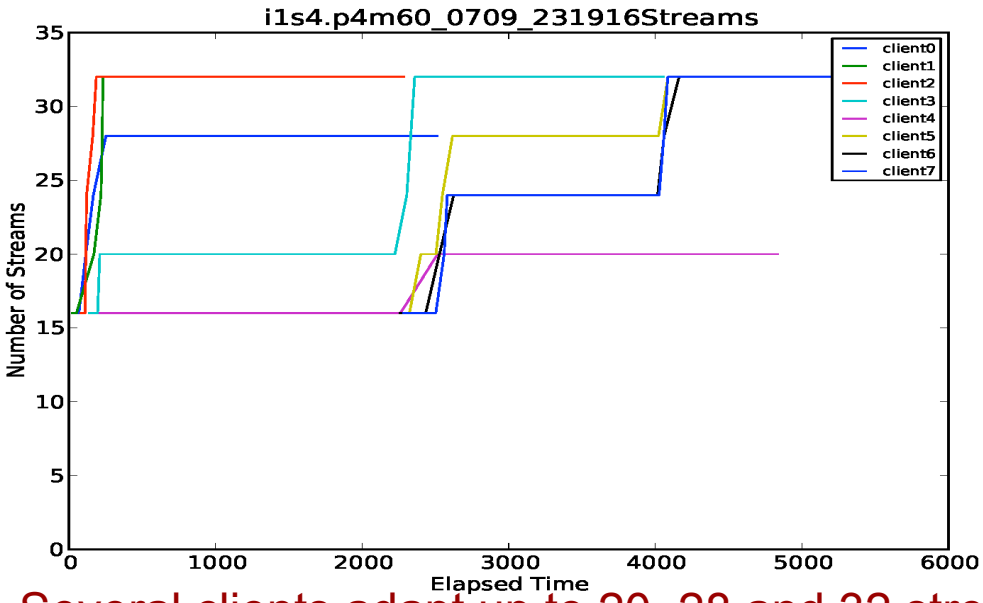


Higher throughputs are reached after clients update their allocations and increase their concurrency several times.

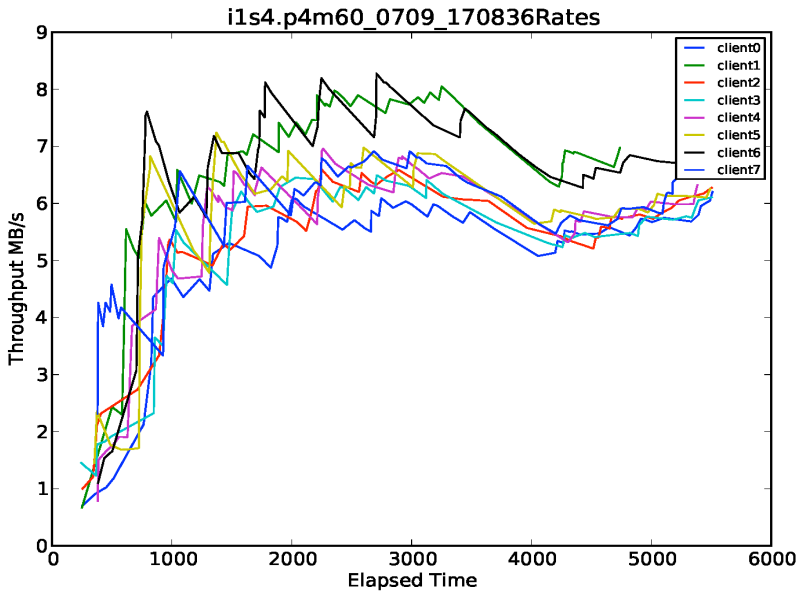
Policy-based Resource Allocation
Fast Increase

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

Stream allocation for fast increases in policy service allocation

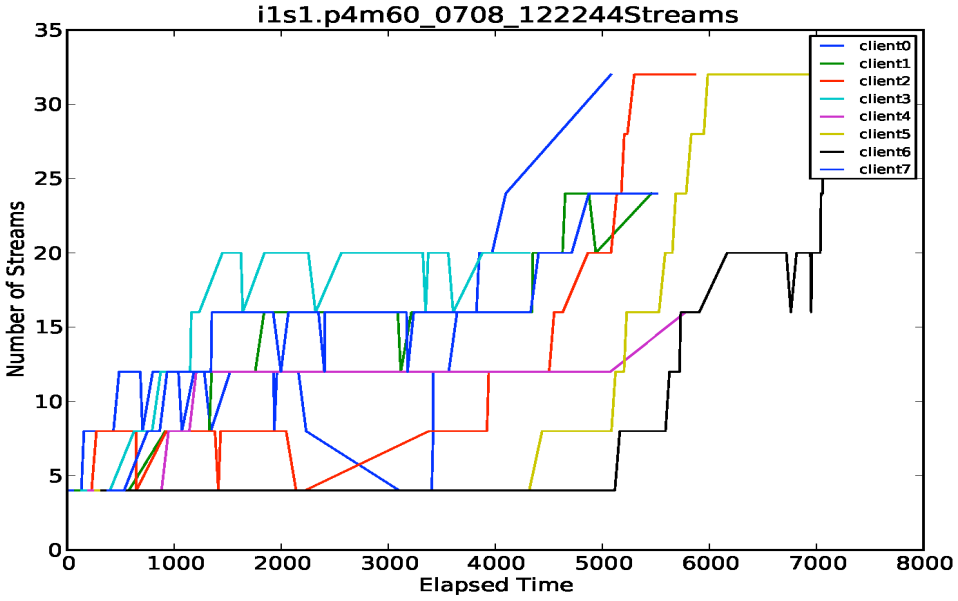


Throughput

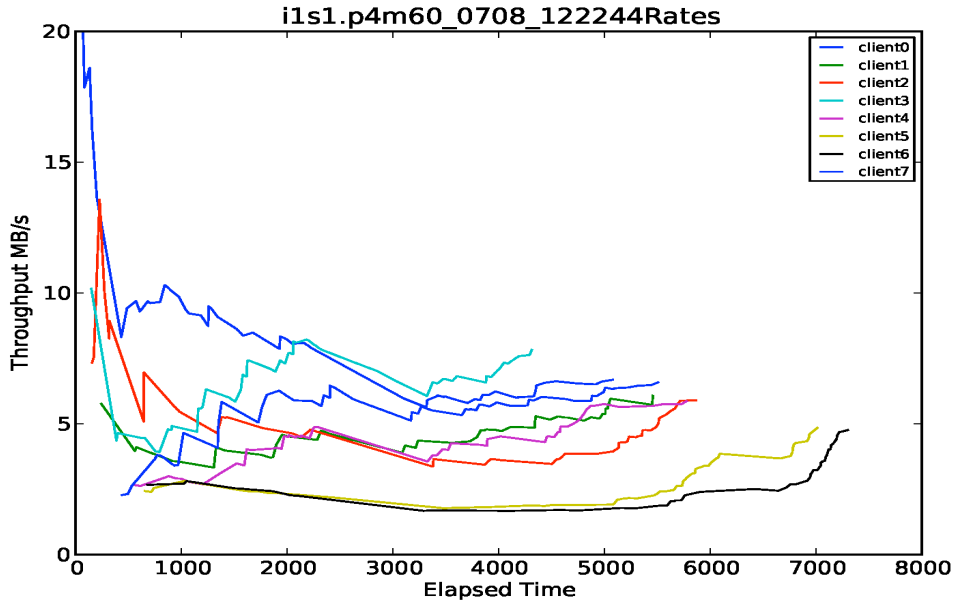


Several clients adapt up to 20, 28 and 32 streams, forcing the last three clients to wait until those first clients complete their transfers and release the streams.

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



Stream used for combined slow PS allocation, client adaptation

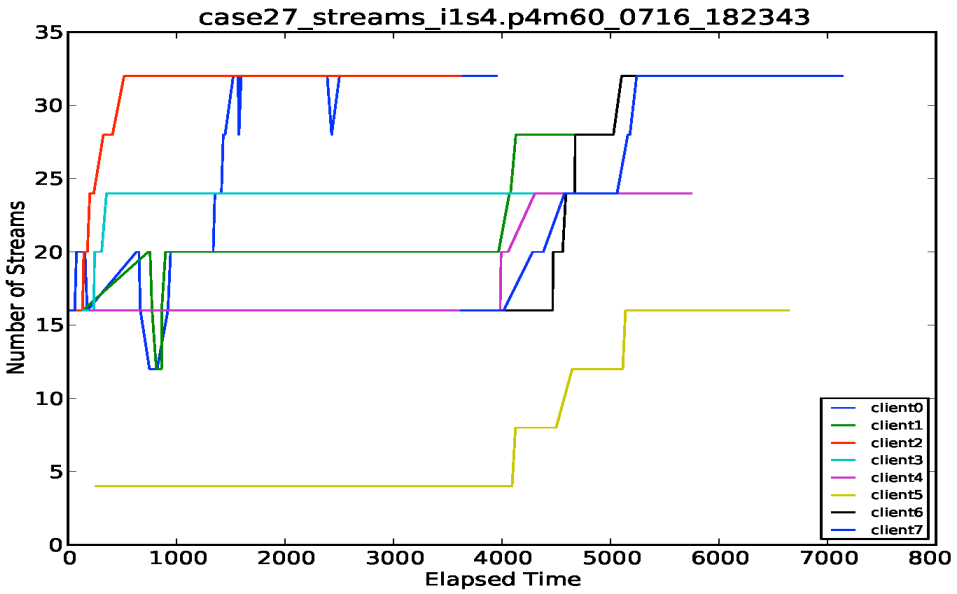


Throughput for combined slow PS allocation, client adaptation

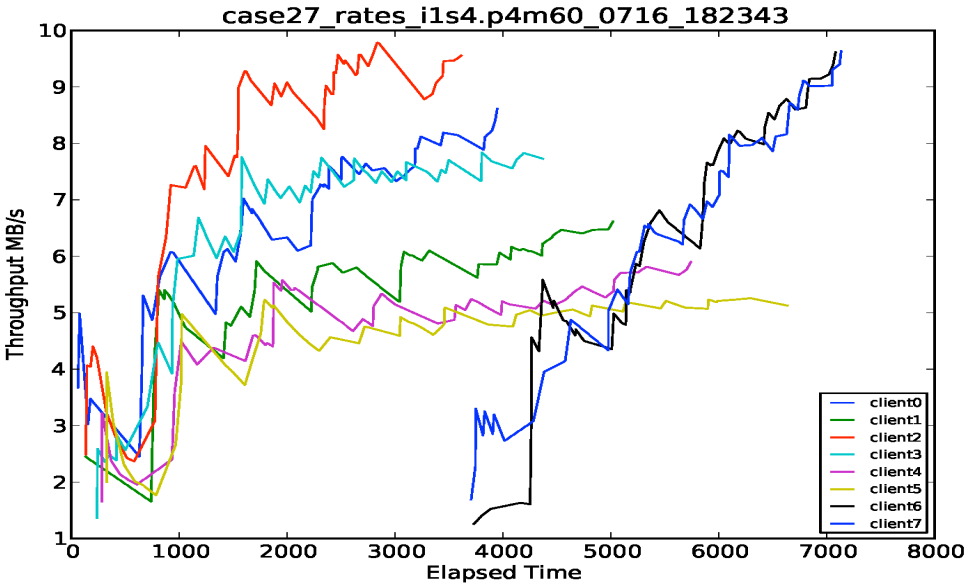
Each client starts out with 4 streams and slowly adapts toward a maximum of 32 streams per client.

The throughput achieved for these transfers is 2 to 10 MB/second after all 8 clients are active.

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



Stream used for combined fast PS allocation, client adaptation



Throughput for combined fast PS allocation, client adaptation

- Most clients starting out with 16 streams and adapting fairly quickly towards 32 streams.
- One client receives an initial allocation of only 4 streams from the PS because all other streams have been allocated.
- Two clients wait for earlier clients to finish and release resources

Experiment – Case 2

Compare performance of adaptive data transfers with unmodified srm-copy client performance, under resource-constrained environment

Transfer data from LBNL in Berkeley, CA to Open Science Grid site at University of Nebraska at Lincoln (UNL), over 1Gbps link

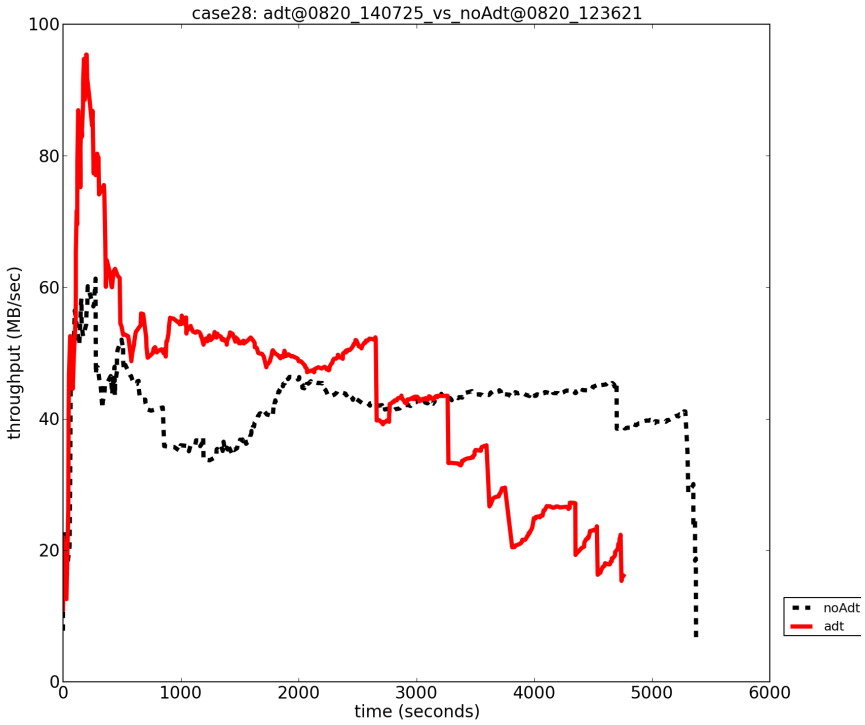
8 srm-copy clients performing multi-file transfers: 260 Gbytes / 488 files

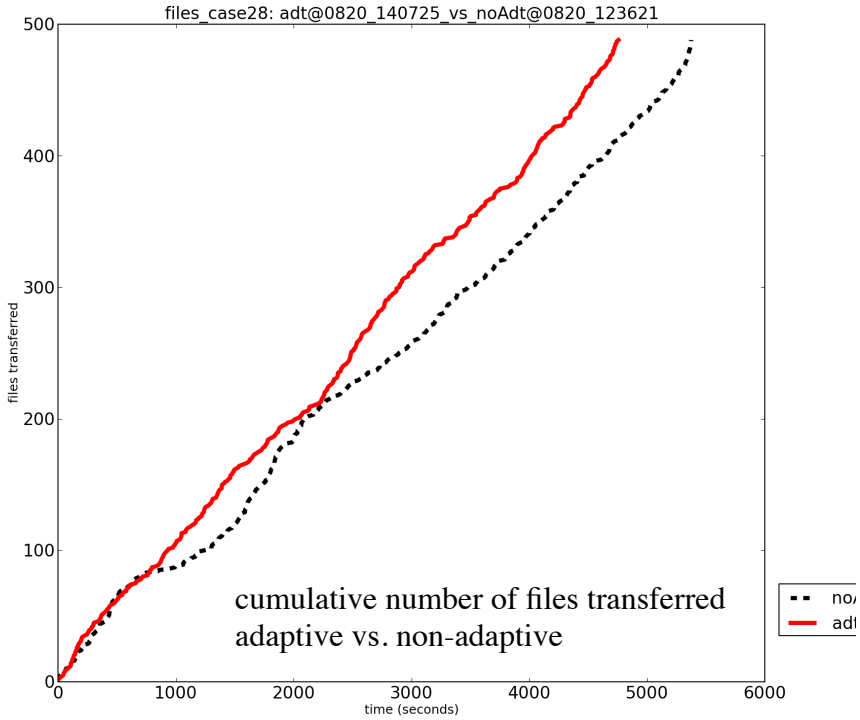
Default common parameters

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

Scenario 1

Cumulative throughput for adaptive vs. non-adaptive transfers, max. 256 total streams, max. 64 streams per adaptive client

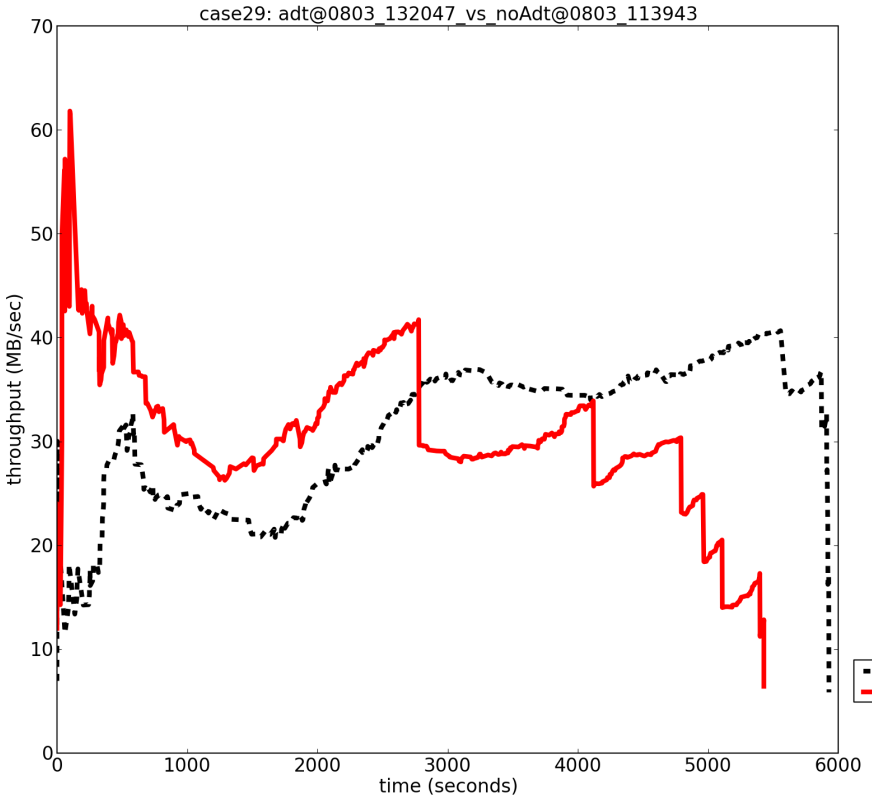


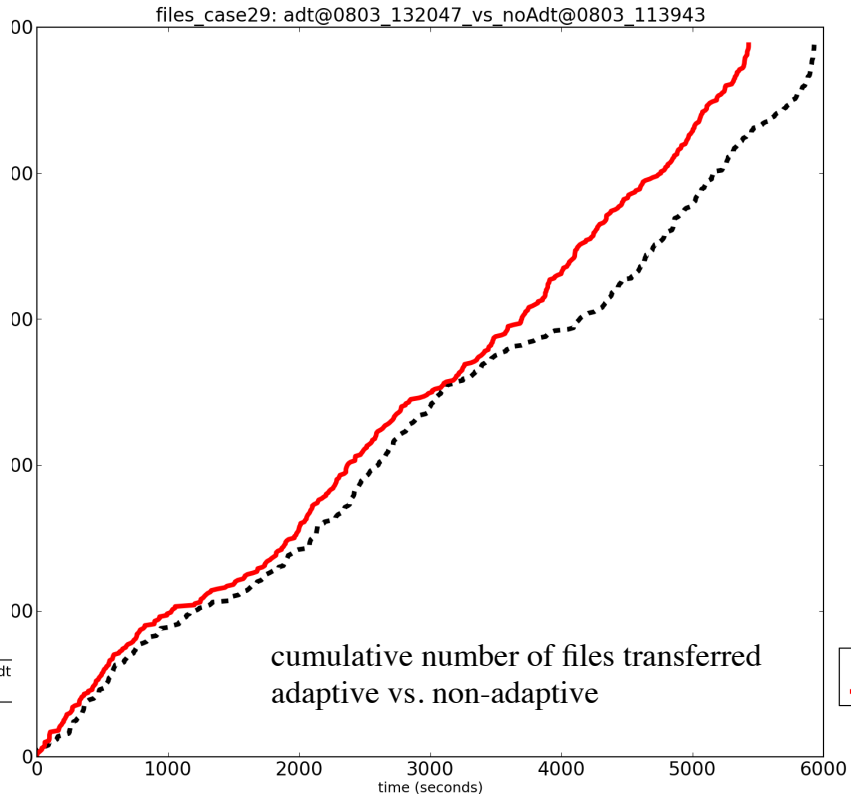


The adaptive transfers (red line) have greater cumulative throughput than the non-adaptive transfers (black line).

Scenario 2

Cumulative throughput for adaptive vs. non-adaptive transfers, max. 384 total streams, max. 96 streams per adaptive client

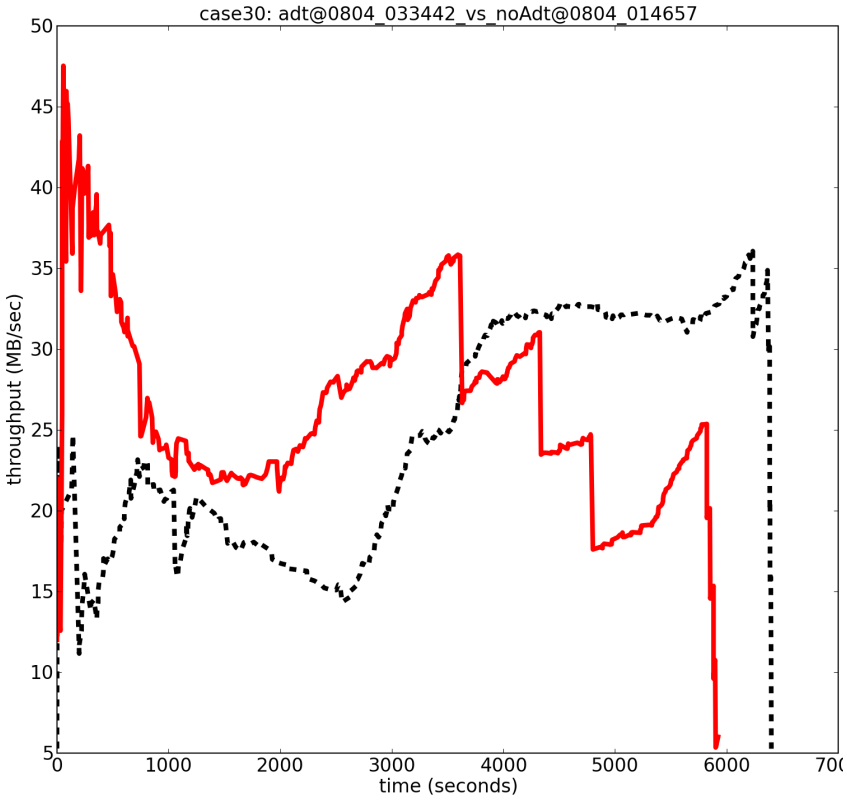


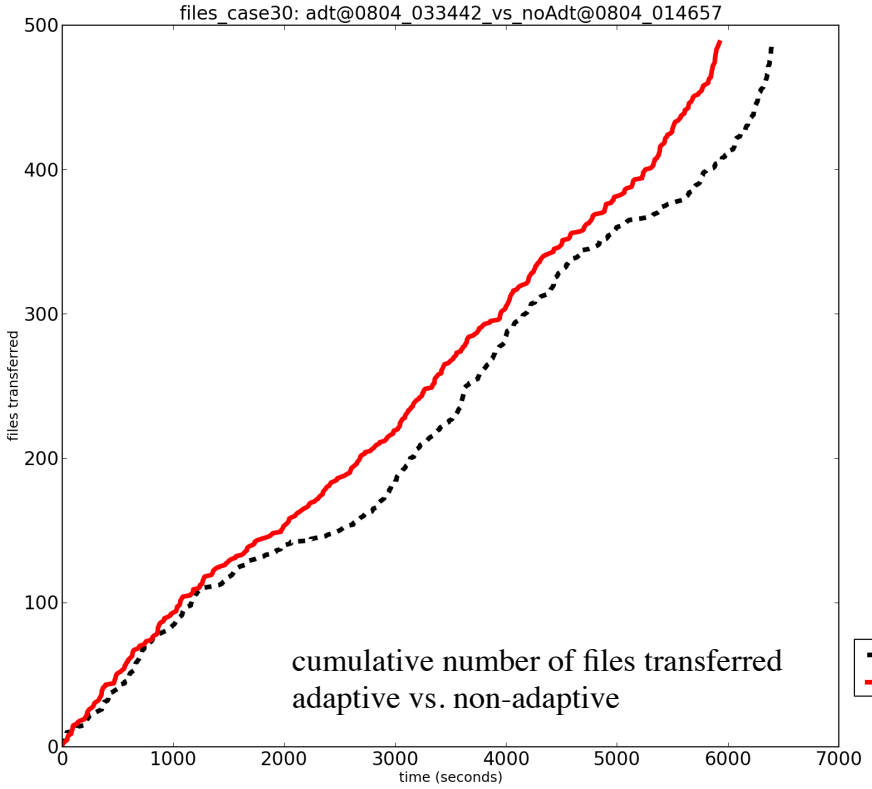


There is a larger vertical separation between the adaptive throughput (red line) and the non-adaptive throughput (black line).

Scenario 3

Cumulative throughput for adaptive vs. non-adaptive transfers, max. 512 total streams, max. 128 streams per adaptive client

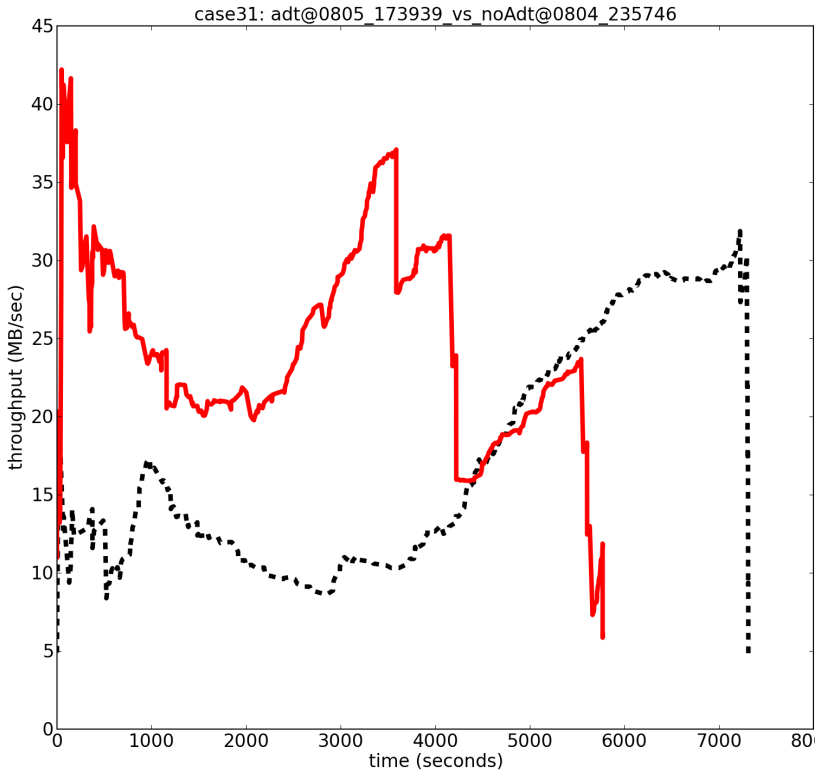


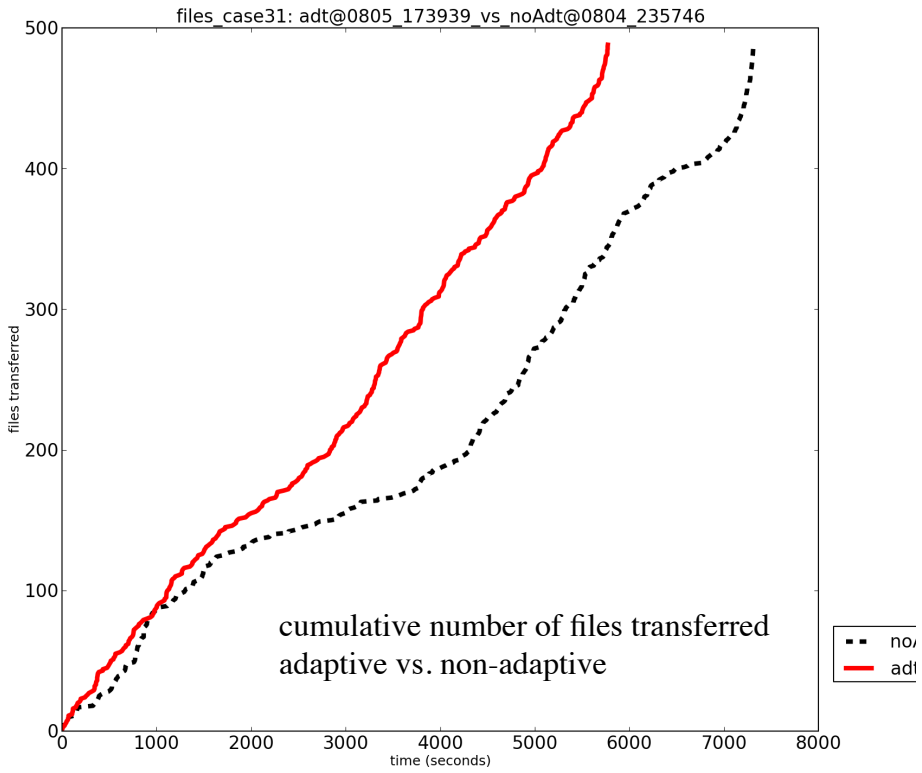


The cumulative throughput of the adaptive transfers continues to increase compared to the non-adaptive transfers as the contention for resources on the infrastructure increases.

Scenario 4

Cumulative throughput for adaptive vs. non-adaptive transfers, max. 640 total streams, max. 160 streams per adaptive client





These experiments show a significant advantage in throughput and overall transfer runtime for the adaptive, policy-based transfers compared to non-adaptive transfers on resource-constrained infrastructure.

In the most resource constrained experiment (Scenario 4), the total transfer time is reduced by approximately 20%.

Summary

ADAPT project goals

- Avoid overprovisioning of resources that results in suboptimal transfer throughput
- Provide simple transition from current data movement practices to policy-based, adaptive data movement

ADAPT software stack

- Provides significant throughput and completion time improvements in resource constrained environments
- Provides better resource utilization and higher throughput and performance in a shared resource environment

Plans for next phase:

- Integrating perfSONAR as performance monitoring archive
- Exploring richer policies for managing resources, adaptation
- Work with application communities to deploy and evaluate ADAPT software

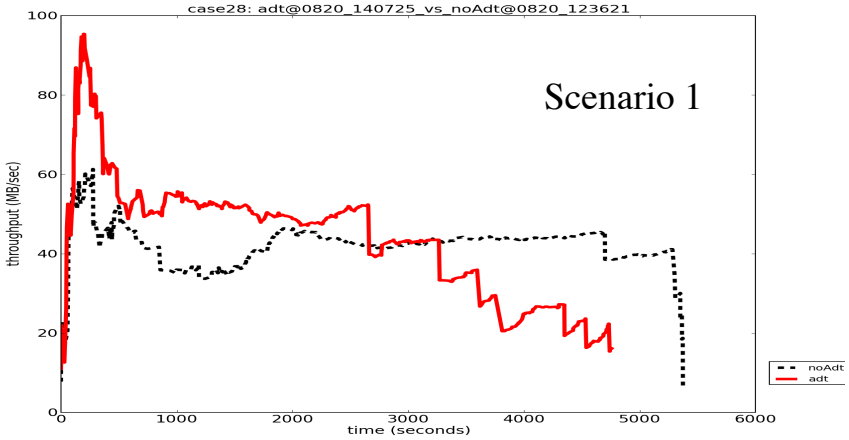
Acknowledgments

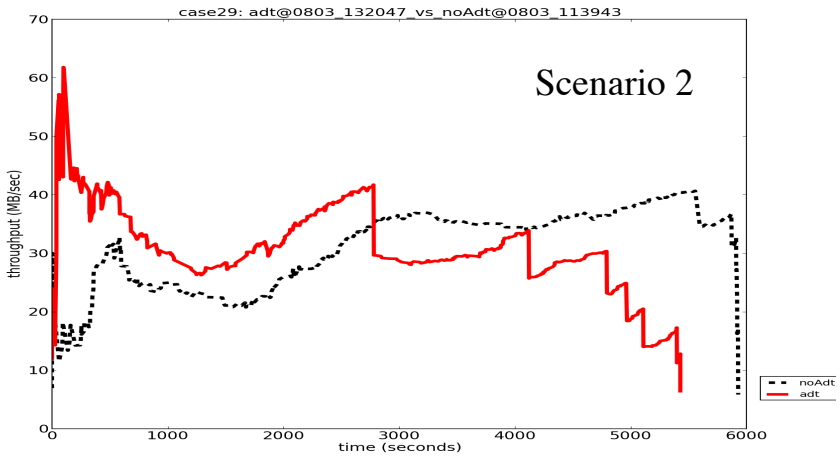
This work was supported in part by:

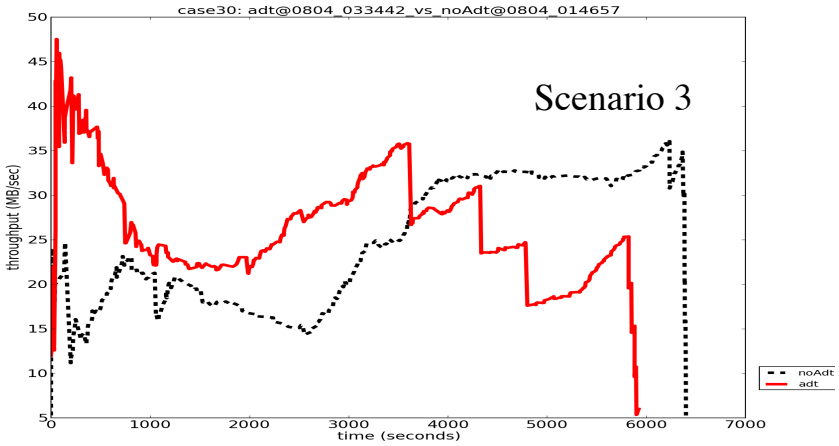
- The National Science Foundation Office of Cyberinfrastructure under award 1127101 (USC/ISI) and award 1127039 (LBNL)
- The Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02- 05CH11231
- This work used resources provided by the Open Science Grid, which is supported by NSF and the U.S. DOE Office of Science
  - We thank Brian Bockelman, Garhan Attebury and Carl Lundstedt at University of Nebraska, Lincoln and Iwona Sakrejda at National Energy Research Scientific Computing Center for their support on our experiments

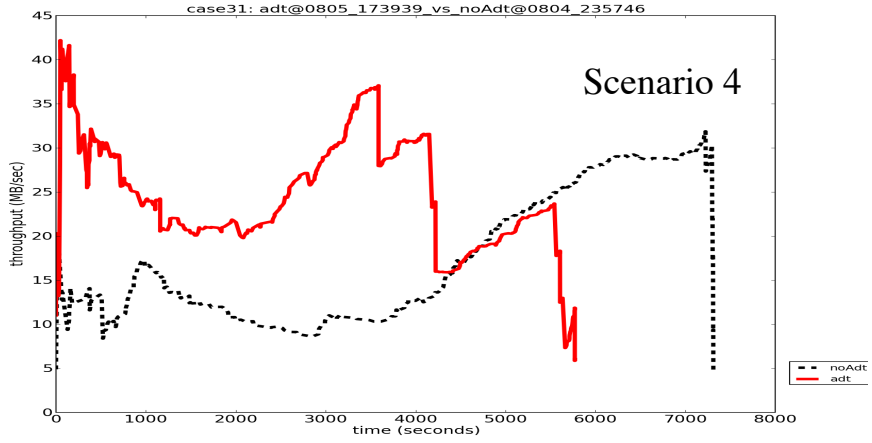
Extra

Cumulative number of files transferred for adaptive vs. non-adaptive transfers









Cumulative throughput for adaptive vs. non-adaptive transfers