

Utility-Based Data Transfers Scheduling between Distributed Computing Facilities

Xin Wang*, Wei Tang†, Rajkumar Kettimuttu†, Zhiling Lan*

*Illinois Institute of Technology, USA

†Argonne National Laboratory, USA

Motivation

Today's scientific applications increasingly involve large amounts of data. The needs for bulk data transfer between remote data centers or computing facilities are growing. Moving the increasing volume of data has imposed a heavy load on the networks between data centers. Especially when multiple data transfers compete for the limited bandwidth, coordinating and scheduling these transfers have become a challenge.

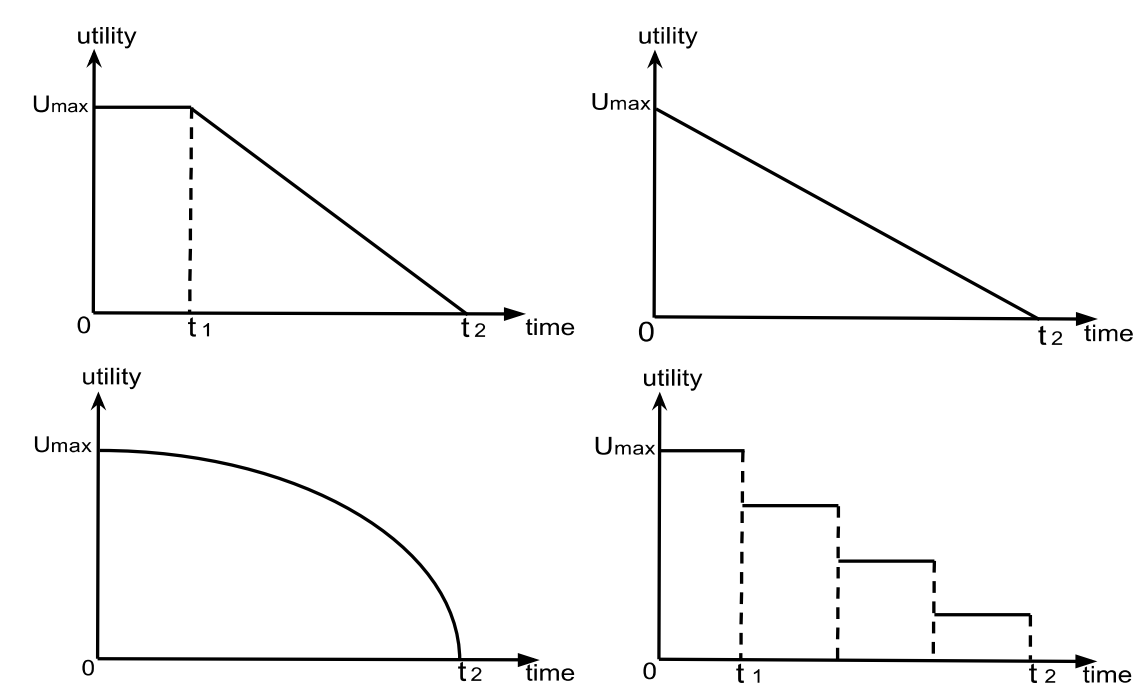
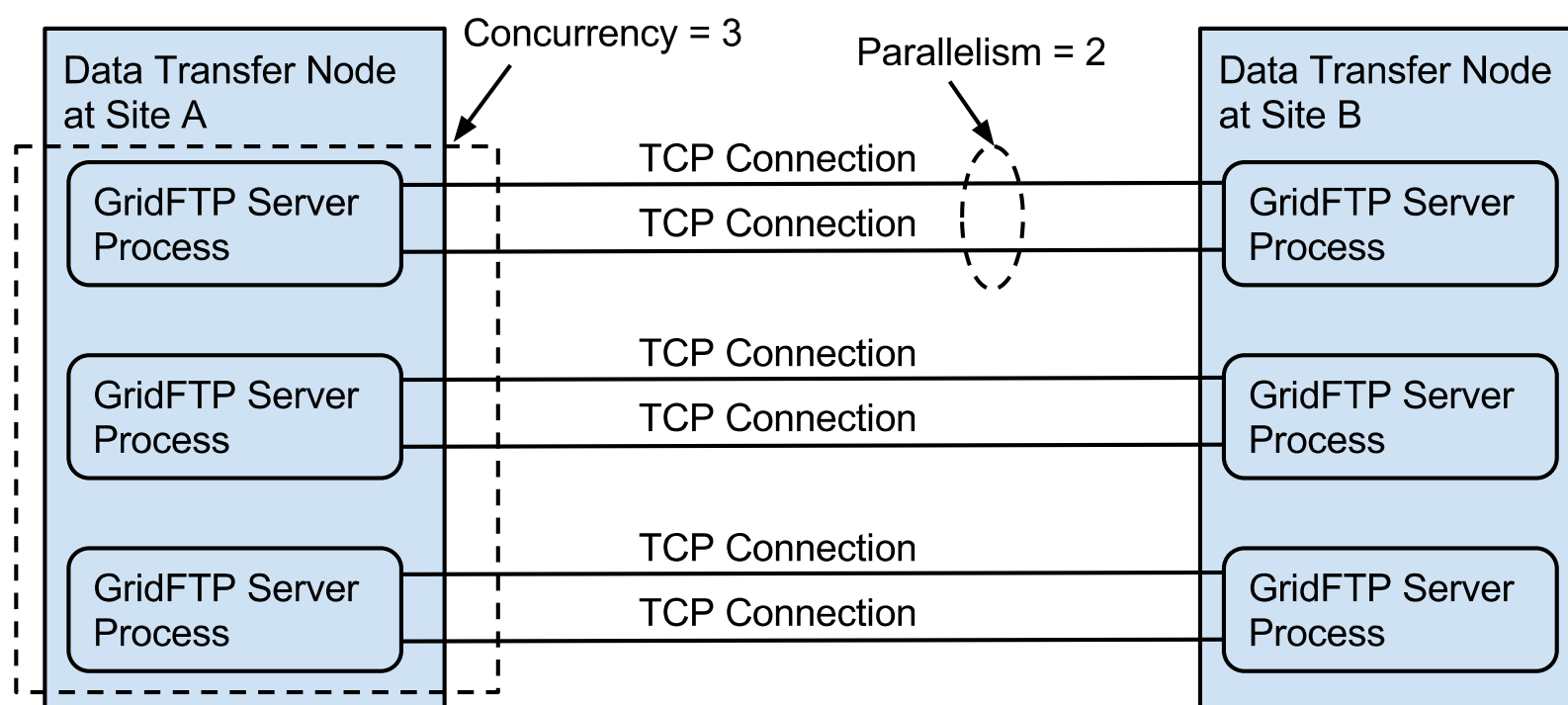
To address these problems, we designed and developed a data transfer scheduler to coordinate data transfer requests between distributed computing facilities. Our goal is to process data transfer requests in an coordinated fashion in order to improve system performance as well as user satisfaction.

Background

We apply our method on which GridFTP is used to manage bulk data transfers via wide-area network.

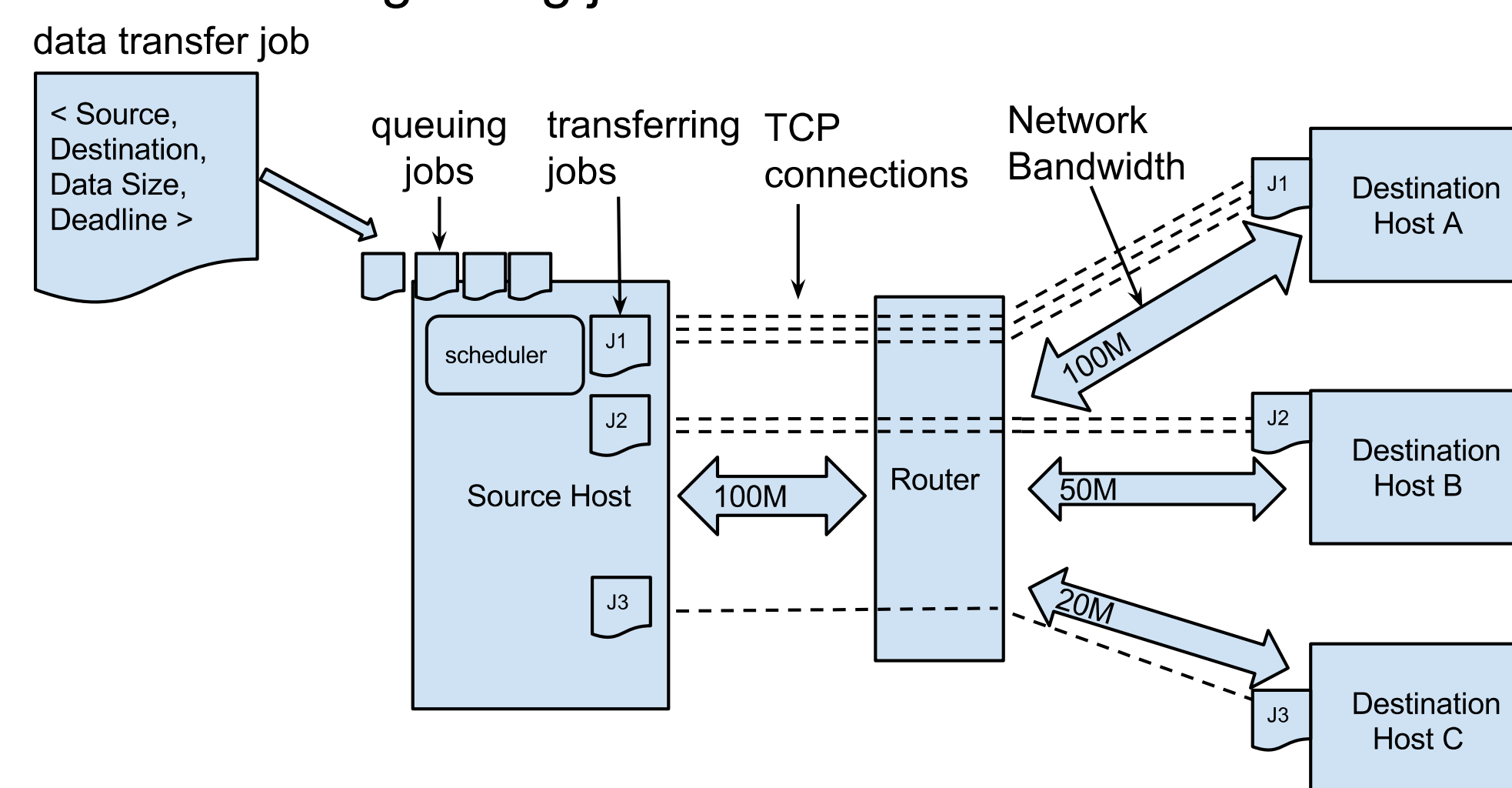
In GridFTP transfers, two key performance-tuning mechanisms include:

- Parallelism
- Concurrency



User satisfaction can be measured by utility function. The utility function is a function of job turnaround time and can be used to represent the value (or utility) that the user attaches to the job completion.

Maximizing aggregate job utility is consistent with an enhanced overall user satisfaction regarding job turnaround.



Working diagram. A source host is connected with three destination hosts via a router. A scheduler inside the source host will coordinate the data transfers and makes following decisions at each scheduling iteration: (1) which jobs should be started now and (2) how many TCP connections to assign to each job.

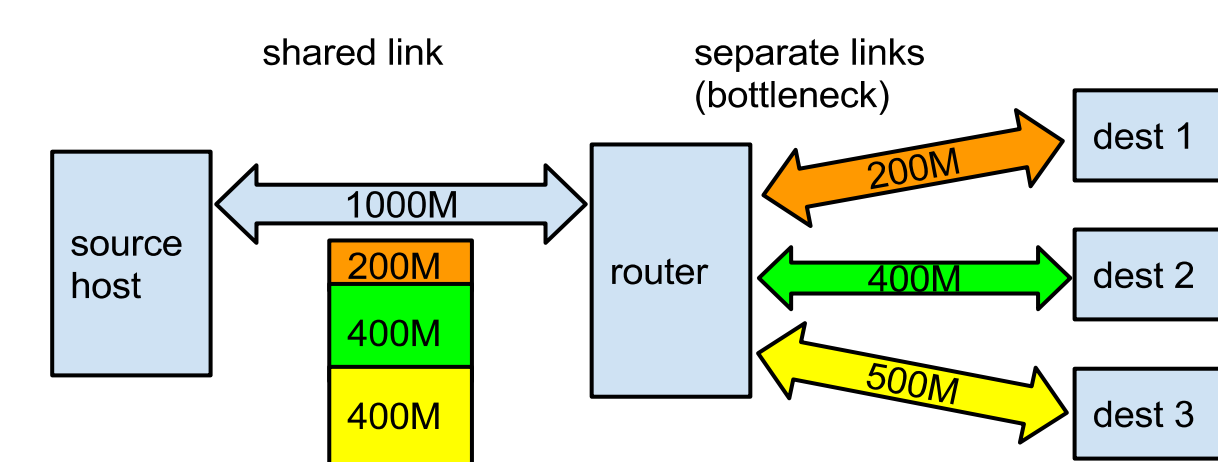
Problem Statement

- A scheduler coordinates data transfer requests (jobs) from one source host to multiple destination hosts

- queue prioritizing (temporal)
- connection allocation (spatial)

- Improve system performance and user satisfaction

- minimizing job turnaround time
- maximizing aggregate job utility



Max-min Fairness: the lowest demand is maximized; only after the lowest demand on the network resource has been satisfied will the second-lowest demand be maximized; and so on.

Table I
NOMENCLATURE

Symbol	Definition	Symbol	Definition
N	# of destinations	BW	shared bandwidth
Q_i	queue for dest. host i	$size$	job size
F_i	fair share for Q_i	W	window size
M	# of jobs	T_w	job waiting time
C_{max}	max. connections	T_e	estimated transfer time
C_{avail}	available connections	t_c	job complete time
C_j	connections for job j	$U_j(\cdot)$	utility for job j

The scheduler solves the following problem and finds the optimal C_j for each job:

$$\begin{aligned} \max & \left(\sum_{j=1}^W U_j(T_w + T_e) \right) \\ \text{s.t.} & \sum_{j=1}^W C_j = C_{avail} \end{aligned} \quad (1)$$

The estimated utility of assigning C_j TCP connections to job j , which is defined as:

$$T_e = \frac{size_j}{f_i * BW} + \text{overhead}, \quad (2)$$

f_i is the fraction of bandwidth of the i -th destination host:

$$f_i = C_j / (F_i * C_{max}). \quad (3)$$

Simulation Framework

- Event-driven simulator built on top of the CODES network simulation framework

- Takes inputs such as various job workloads, scheduling policies, and network configurations.

- Dsim emulates data transfer scheduling with two internal components, queue manager and scheduler

- Analyze the performance metrics based on the generated output events.

Algorithm 1: Utility-Based Data Transfer Scheduling

Input: a set of data transfer jobs J
Output: a subset of jobs J_s that can start, each with an assigned number of C (connections)

```
1  $Q_c = \text{assign\_queue\_C\_min\_max\_fairness}(J, BW)$ ;  
2  $J_s = []$ ;  
3 foreach  $q$  in  $Q_c$  do  
4    $J_c = \text{select\_candidate\_jobs\_by\_window}(q, W)$ ;  
5    $J_s^q = \text{sched\_C\_by\_utility}(J_c)$ ;  
6    $J_s.append(J_s^q)$ ;  
7 end  
8  $\text{start\_transfers}(J_s)$ ;
```

Step 1: Bandwidth Allocation

- Allocate the shared bandwidth for different destination
- Use max-min fairness approach
- Assign the number of TCP connections to different destinations to resolve bandwidth allocation

Step 2: Job Prioritizing and Selection

- Base job prioritizing methods:
 - First-come, first-serve (FCFS)
 - Short-job-first (SJF), jobs are sorted based on the ratio of waiting time and job size ($T_w/size$)
- Use sliding window to select the first W jobs in the queue as candidates for scheduling

Step 3: Utility-Based Connection Allocation

- Assign each candidate job certain TCP connections to achieve maximum aggregate utility by solving equations on the right.
- Greedy algorithm:
 - Initially evenly distribute the total available TCP connections to each candidate job
 - Conduct connection exchange repeatedly, at which reduce one connection from a job and add it to another job to increase aggregate utility
 - Stop when the aggregate utility cannot be increased anymore

Experiments

- Simulating 2 hour real job traces from data transfer node(DTN) at Stampede to three different destination DTNs.
- Categorize jobs into small jobs($\leq 1G$) and large jobs($> 1G$)
- Conduct experiments with different numbers of maximum TCP connections varying from 10 to 50.
- Define equal utility function for each job within category

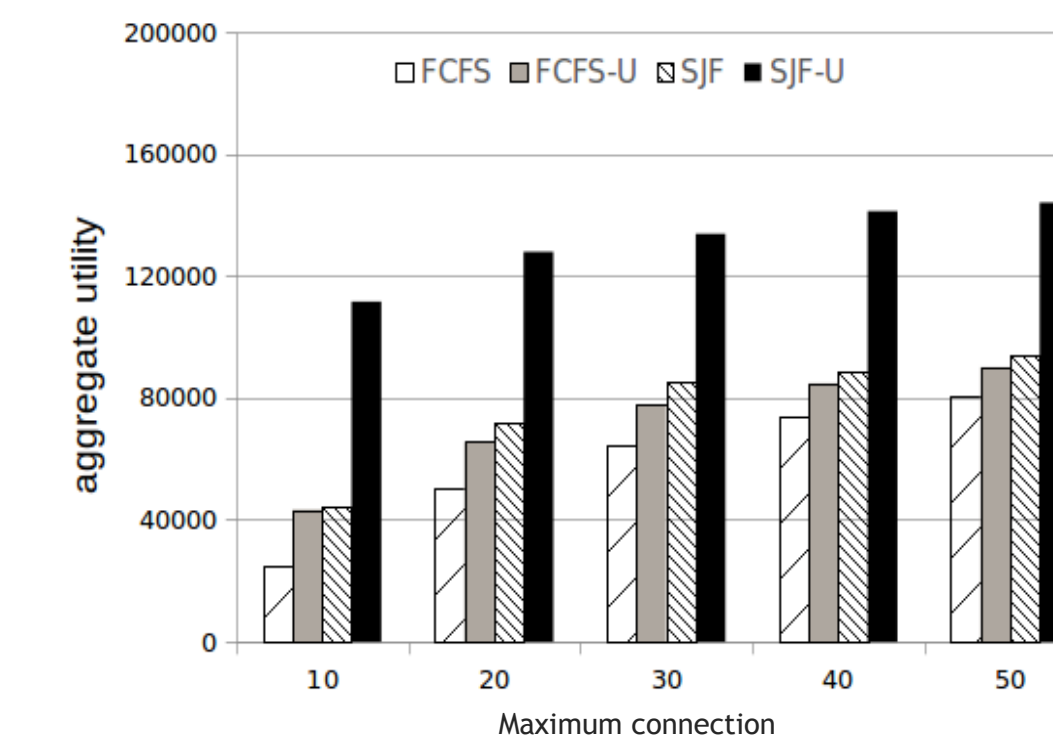
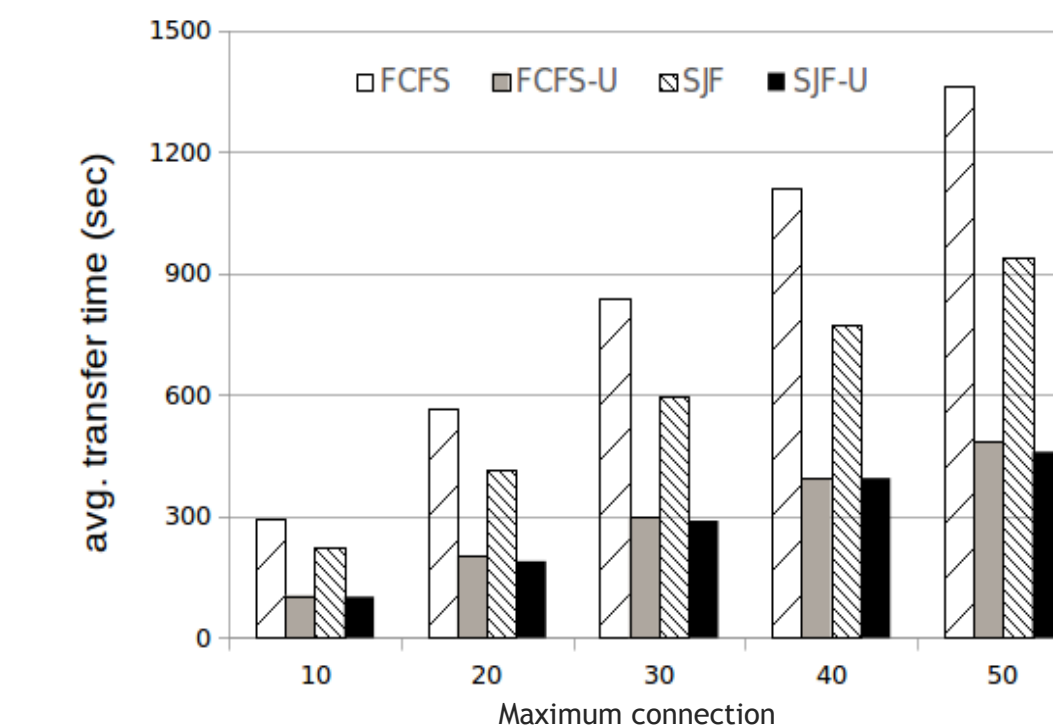
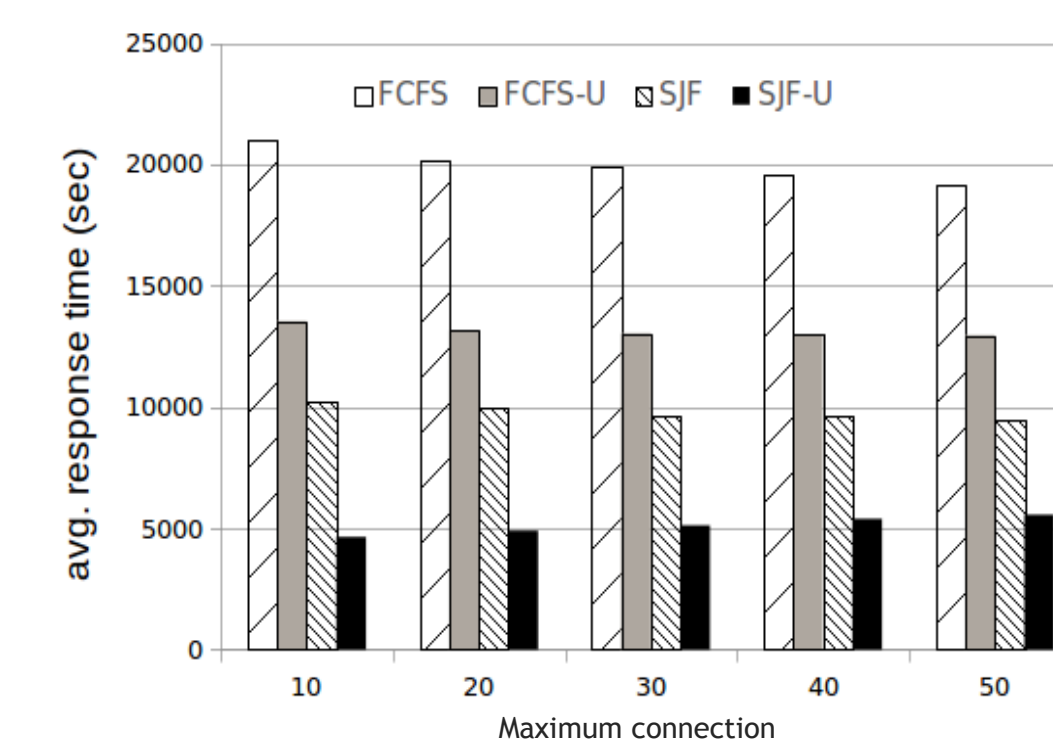
JOB TRACE STATISTICS

Category	Job Count	Avg.Size(MB)	Small Job	Large Job
Gordon	797	1068.43	61%	39%
Mason	624	816.38	63%	37%
Yellowstone	897	1032.36	63%	37%
Total	2248	985.20	61%	39%

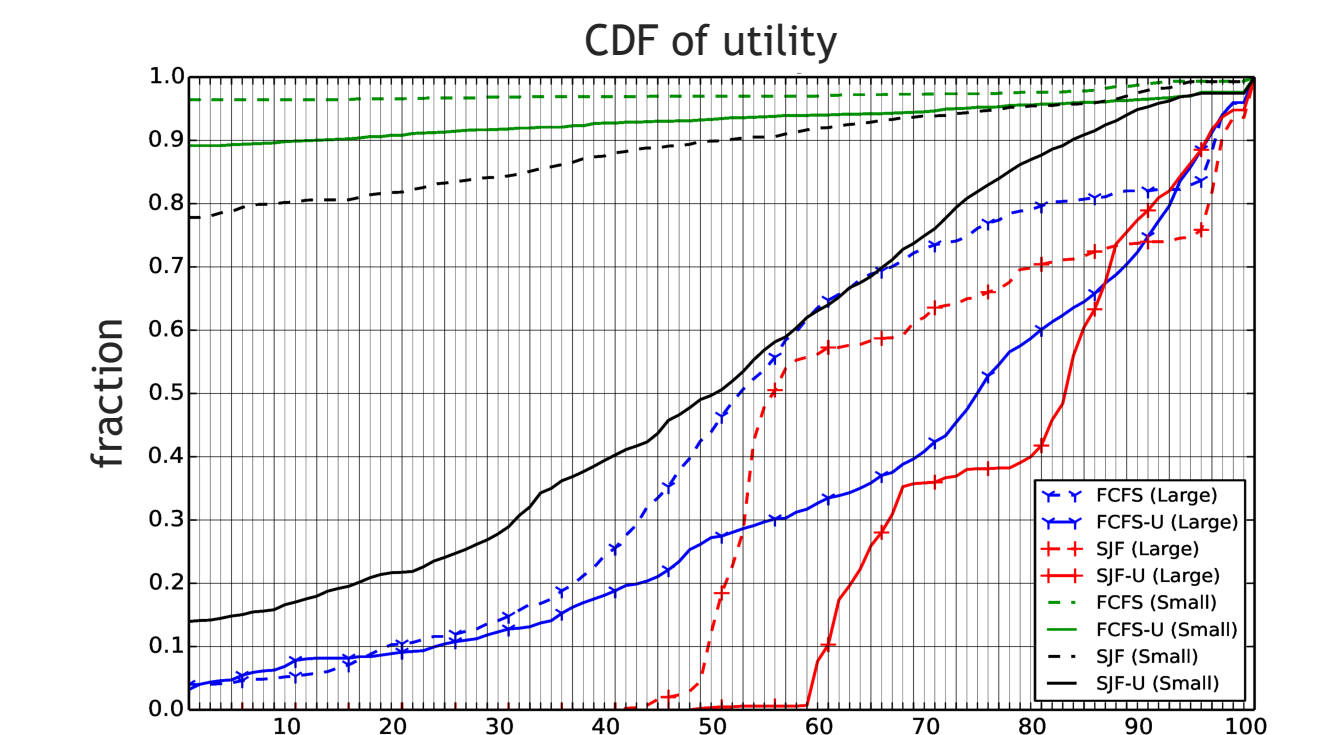
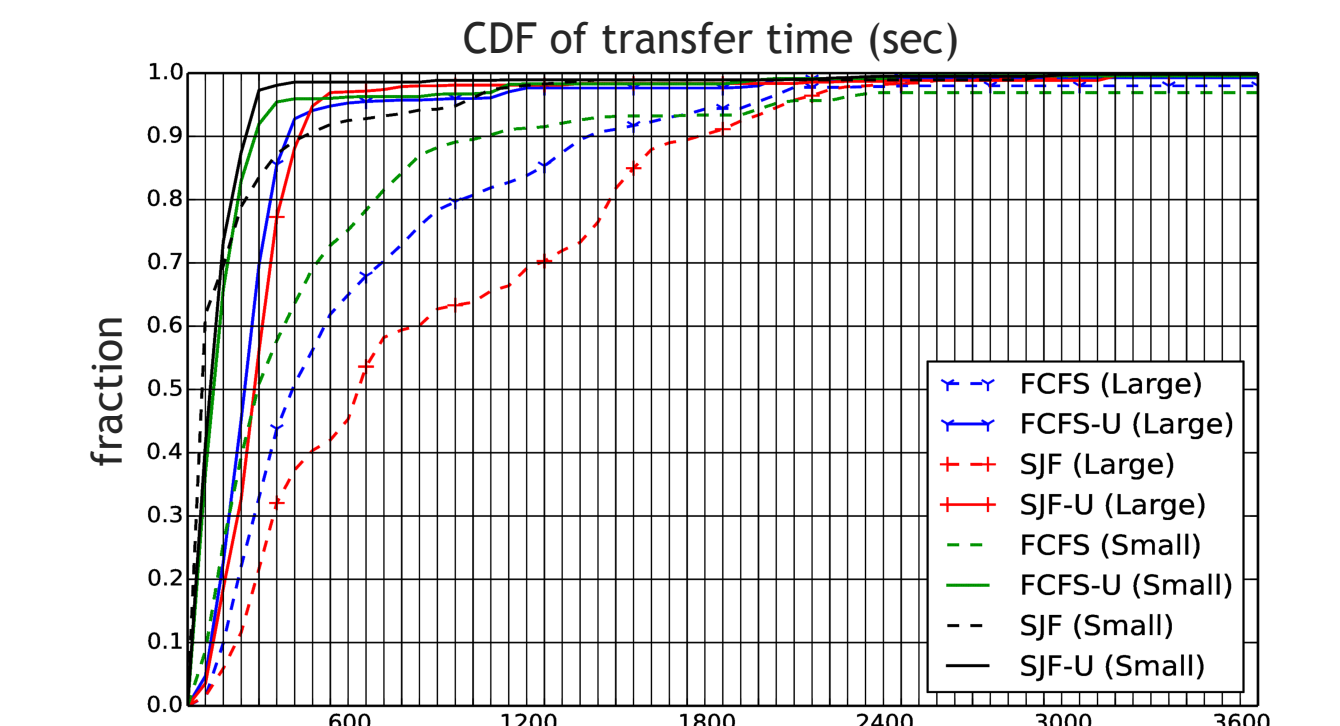
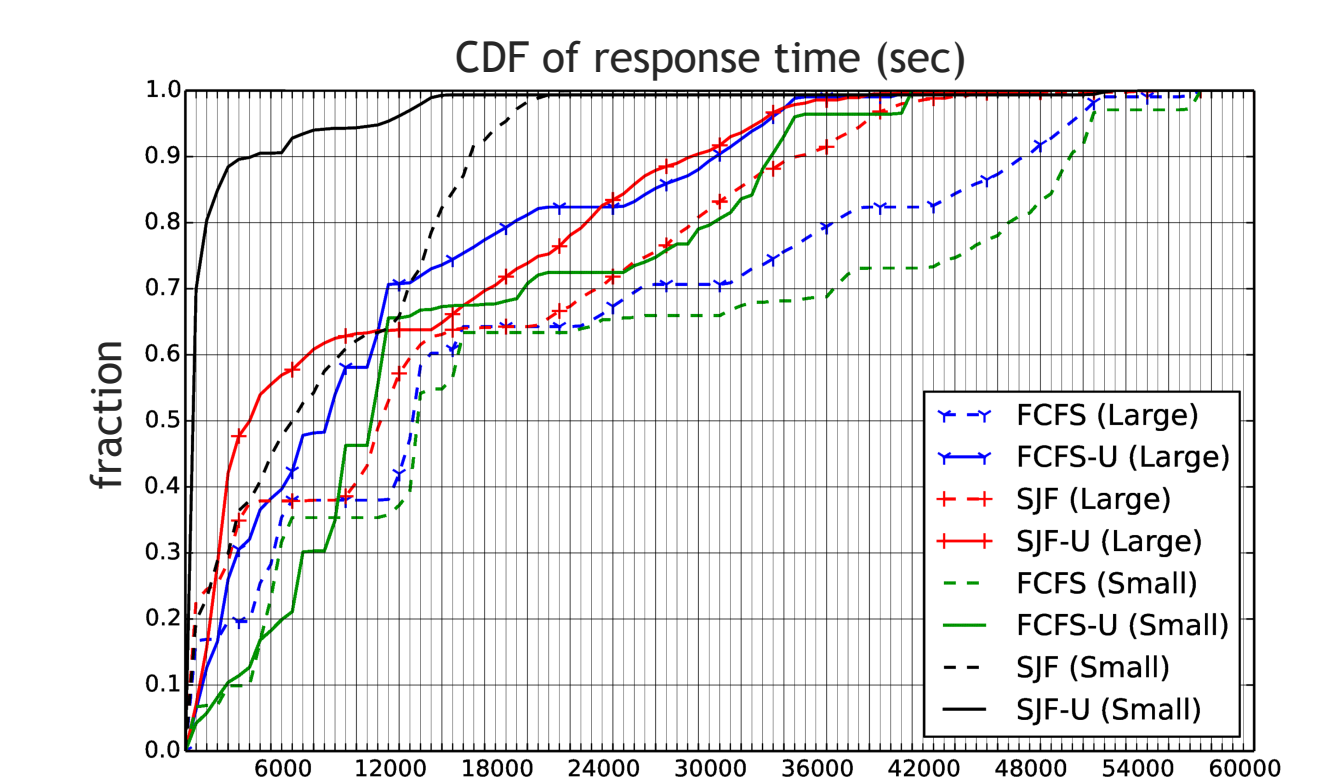
Four Scheduling Policies:

- FCFS
- FCFS-U
- SJF
- SJF-U

Performance Improvement



Performance CDF



The utility optimization model considerably improves job response time, transfer time and aggregate utility.

Utility function improves avg. response time for both large and small jobs and improves data transfer time and aggregate utility more for small jobs.

Future Work

- Diverse job types (mixture of batch jobs and real-time jobs with deadlines)
- Various utility functions for different job types
- More complex network topology
- Introducing dynamic network bandwidth configurations

References

- [1] Dsim project repo: <https://github.com/xwang149/Dsim>
- [2] C. B. Lee, and A. E. Snavey. Precise and realistic utility functions for user-centric performance analysis of schedulers. In Proc. of 16th international symposium on High Performance Distributed Computing, 2007.
- [3] R. Kettimuthu, G. Vardoyan, G. Agrawal, P. Sadayappan. Modeling and optimizing large-scale wide-area data transfers. In Proc. of CCgrid, 2014.
- [4] W. Tang, J. Jenkins, F. Meyer, R. Ross, R. Kettimuthu, L. Winkler, X. Yang, T. Lehman, and N. Desai. Data-aware resource scheduling for multi-cloud workflows: A fine-grained simulation approach. In Proc. of IEEE International Conference on Cloud Computing Technology and Science, 2014.