



# Enhancements in Stork Data Placement Scheduler

Mehmet Balman, Tevfik Kosar

Center for Computation and Technology, Louisiana State University



## STORK: A Scheduler for Data Placement Activities

Data management has been a crucial problem in every stage of computer engineering, from micro to macro level systems. We focus on data access and data placement problems in distributed systems for large scale applications. We study aggregation of requests in order to increase the throughput especially for transfers of small data files. We also explore the possibility of an efficient error detection and reporting system for distributed environments. In addition, we are investigating techniques to make use of replicas for multi-source downloads. We also work on several enhancements like file similarity analysis and semantic compression methods to reduce total transfer size.

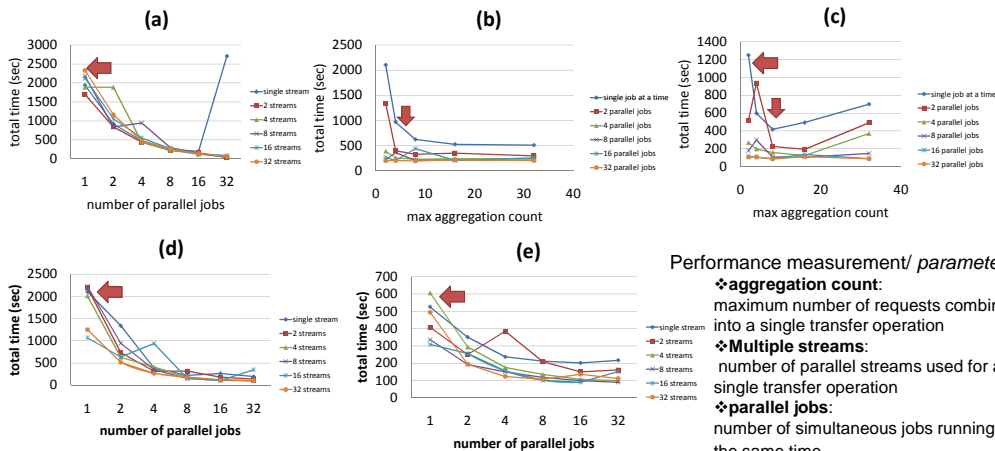
## Aggregation of Data Placement Jobs

According to the file size and source/destination pairs, data placement jobs are combined and processed as a single transfer job. Information about the aggregated job is stored in the job queue and it is tied to a main job which is actually performing the transfer operation such that it can be queried and reported separately.

We have seen vast performance improvement, especially with small data files, simply by combining data placement jobs based on their *source* or *destination* addresses. The main performance gain comes from decreasing the amount of protocol usage and reducing the number of independent network connections. Thus, Stork makes better use of underlying infrastructure by coordinating and arranging data placement jobs.

## Experiments on LONI (Louisiana Optical Network Initiative)

✓ **Test-set:** 1024 transfer jobs from Ducky to Queenbee (rtt avg 5.129 ms) - 5MB data file per job



Performance measurement/ parameters:

- ♦ **aggregation count:** maximum number of requests combined into a single transfer operation
- ♦ **Multiple streams:** number of parallel streams used for a single transfer operation
- ♦ **parallel jobs:** number of simultaneous jobs running at the same time.

**Fig:** Effects of parameters over total transfer time of the test-set

- without job aggregation – number of parallel jobs vs number of multiple streams
- transfer over single data stream – aggregation count vs number of parallel jobs
- transfer over 32 streams – aggregation count vs number of parallel jobs
- at most 2 requests are aggregated – number of parallel jobs vs multiple streams
- at most 16 requests are aggregated – number of parallel jobs vs multiple streams

## Connection with CyberTools

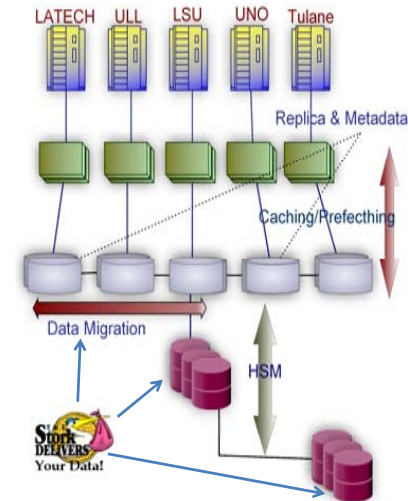
### PetaShare Core Architecture

#### Two types of data movement:

- o First, data needs to be perfected from low level storage layers to the higher levels such that management of data access has to be handled in an efficient manner.
- o Second, data should be migrated between those five contributing institutions; moreover, data should be scheduled and moved between distributed sites and the clients.

Protocols:

file://	->	local file
ftp://	->	FTP
http://	->	HTTP
gsiftp://	->	GridFTP
srb://	->	SRB (Storage Resource Broker)
irods://	->	IRODS



## Error Detection and Recovery

Stork, data placement scheduler, checks network connection and availability of data transfer protocol beforehand with the help of new network exploration module. We have implemented error detection and classification as new features inside Stork. Our experiments, in which we are generating artificial errors for testing purpose, shows that current data transfer protocol are not always able to generate adequate log information; therefore we also focus on tracing the transfer job and preparing the infrastructure to explore dynamic instrumentation while transfer is in progress.

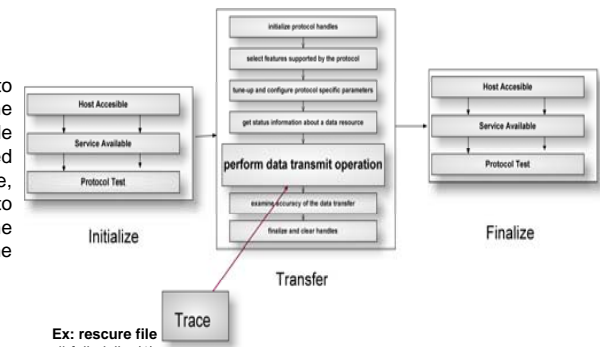
### stork.globus-url-copy:

(supports wildcards and recursive copy)

Stork GridFtp data transfer module is able to verify the successful completion of the operation by controlling checksum and file size. Besides, it can recover from a failed operation. In case of a retry from a failure, scheduler informs the transfer module to recover and restart the transfer using the information from a rescue file created by the transfer module.

#### Stork.globus-url-copy features

- ckp | -checkpoint - use a rescue file for checkpointing
- ckpdebug | -checkpoint-debug
- ckpfile <filename> | -checkpoint-file <filename> checkpoint filename. Default is "<pid>.rescue"
- cksm | -checksum > checksum control after each transfer
- pcheck | -port-check check network connectivity and availability of the protocol



#### Ex: rescue file

```
#failed_list (1):
gsiftp://dsl-turtle06.csc.lsu.edu/tmp/test/x_ttest2.tar file://tmp/x_ttest2.tar
#expanded_url_list (7):
gsiftp://dsl-turtle06.csc.lsu.edu/tmp/test/out file://tmp/out
gsiftp://dsl-turtle06.csc.lsu.edu/tmp/test/test2/test22/ file://tmp/test2/test22/
#transferred_list (2):
# gsiftp://dsl-turtle06.csc.lsu.edu/tmp/test/test1/ file://tmp/test1/
# gsiftp://dsl-turtle06.csc.lsu.edu/tmp/test/test2/ file://tmp/test2/
```

## Acknowledgement

**DSL** Distributed System Laboratory  
www.dsl.csc.lsu.edu

**Stork** Your Data!  
www.storkproject.org

This project is in part sponsored by the National Science Foundation under award numbers CNS-0619843 (PetaShare) and EPS-0701491 (CyberTools), and by the Board of Regents, State of Louisiana, under Contract Number NSF/LEQSF (2007-10)-CyberRII-01.

