CENTER FOR COMPUTATION
& TECHNOLOGY

Stork 1.0 and Beyond

Data Scheduling for Large-scale Collaborative Science

Mehmet Balman

Louisiana State University, Baton Rouge, LA, USA

Presented at **Condor Week 2009** April 20-April 23, 2009

Scheduling Data Placement Jobs

- Data Placement Activities
- Modular Architecture
  - Data Transfer Modules

  for specific protocols/services

- Throttle maximum transfer operations running
- Keep a log of data placement activities
- Add fault tolerance to data transfers

Job Submission

```
[   dest_url = "gsiftp://eric1.loni.org/scratch/user/";
    arguments = -p 4 dbg -vb";
    src_url = "file:///home/user/test/";
    dap_type = "transfer";
    verify_checksum = true;
    verify_filesize = true;
    set_permission = "755" ;
    recursive_copy = true;
    network_check = true;
    checkpoint_transfer = true;
    output = "user.out";
    err = "user.err";
    log = "userjob.log";
]
```

Agenda

- **Error Detection and Error Classification**
- Data Transfer Operations
  - Dynamic Tuning
  - Prediction Service
  - Job Aggregation
- Data Migration using Stork
    - Practical example in PetaShare Project
- Future Directions

Failure-Awareness

- Dynamic Environment:
- data transfers are prune to frequent failures
    - what went wrong during data transfer?
    - No access to the remote resources
    - Messages get lost due to system malfunction

- Instead of waiting failure to happen
    - Detect possible failures and malfunctioning services
    - Search for another data server
    - Alternate data transfer service

- Classify erroneous cases to make better decisions

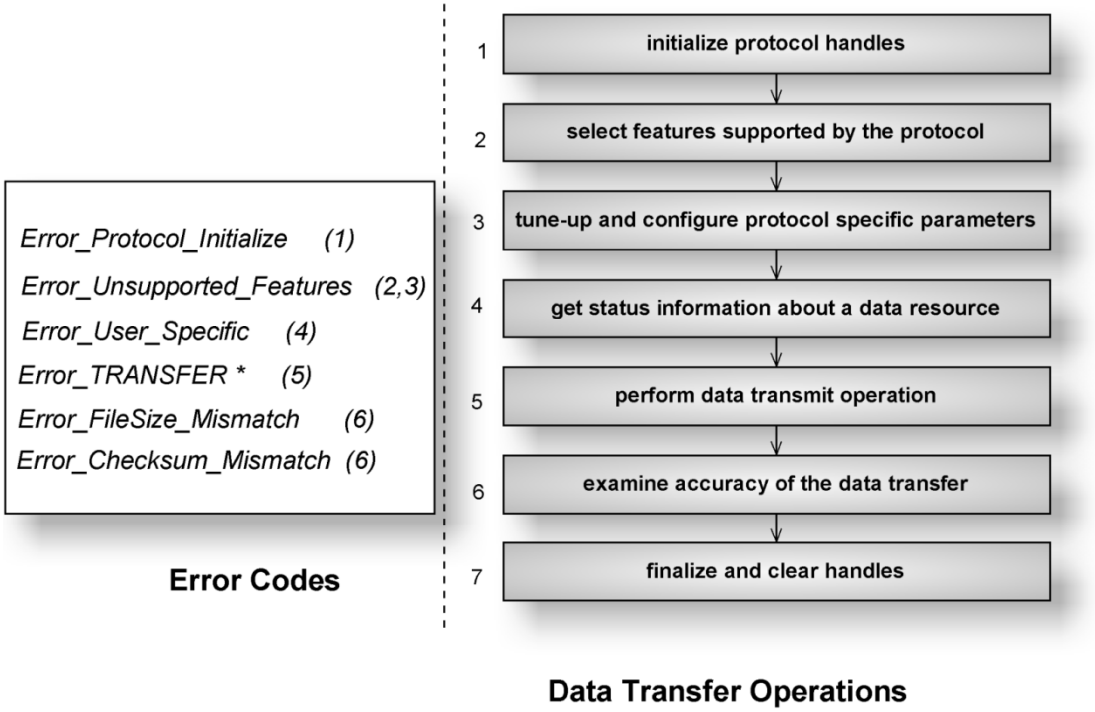
Error Detection

- Use Network Exploration Techniques
  - Check availability of the remote service
  - Resolve host and determine connectivity failures
  - Detect available data transfers service

  - should be Fast and Efficient not to bother system/network resources

- Error while transfer is in progress?
  - Error_TRANSFER
- Retry or not?
- When to re-initiate the transfer
- Use alternate options?

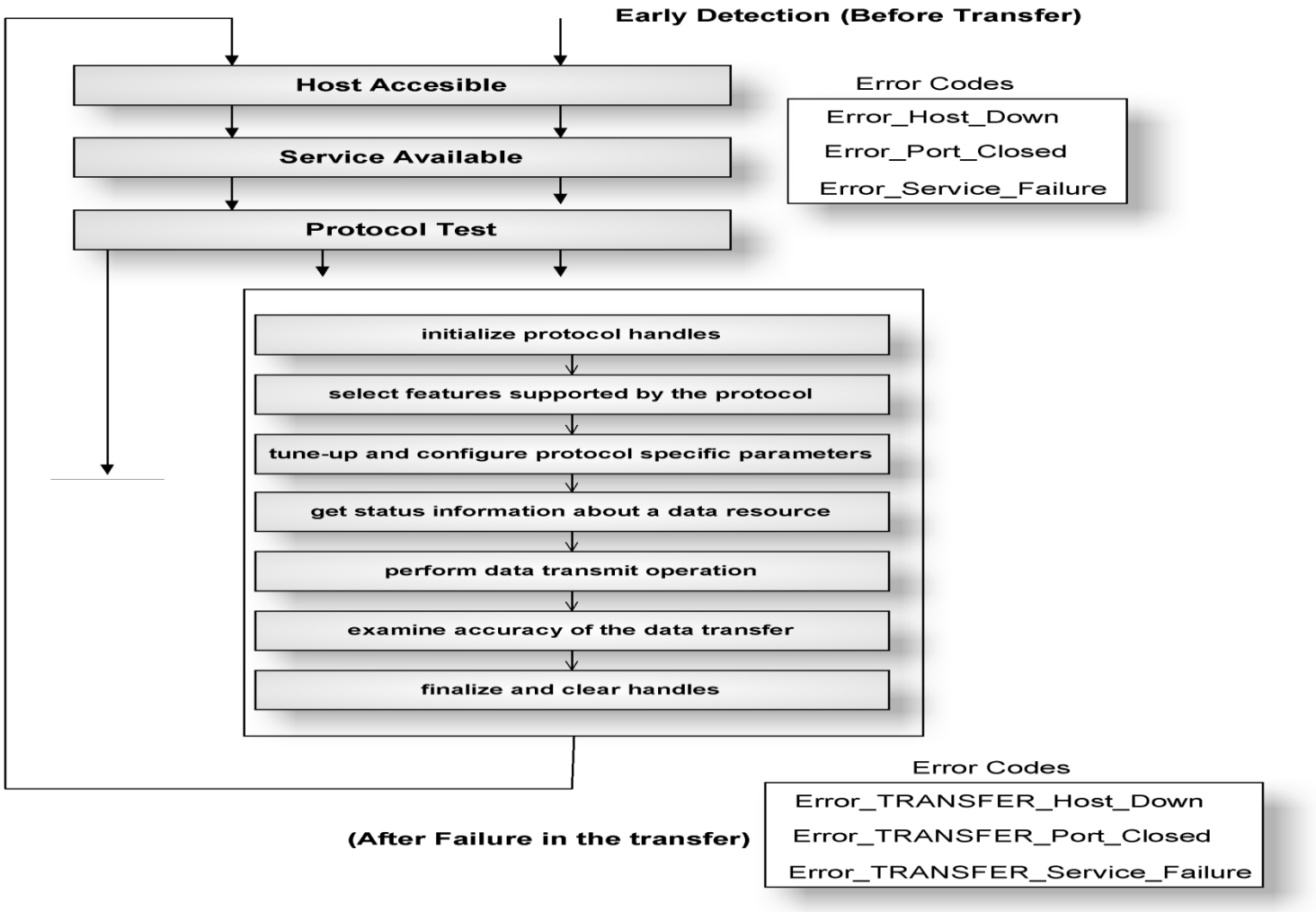
Error Classification



- Recover from Failure
- Retry failed operation
  - Postpone scheduling of a failed operations
- Early Error Detection
  - Initiate Transfer when erroneous condition recovered
  - Or use Alternate options

- Data Transfer Protocol not always return appropriate error codes
- Using error messages generated by the data transfer protocol
- A better logging facility and classification

Error Reporting

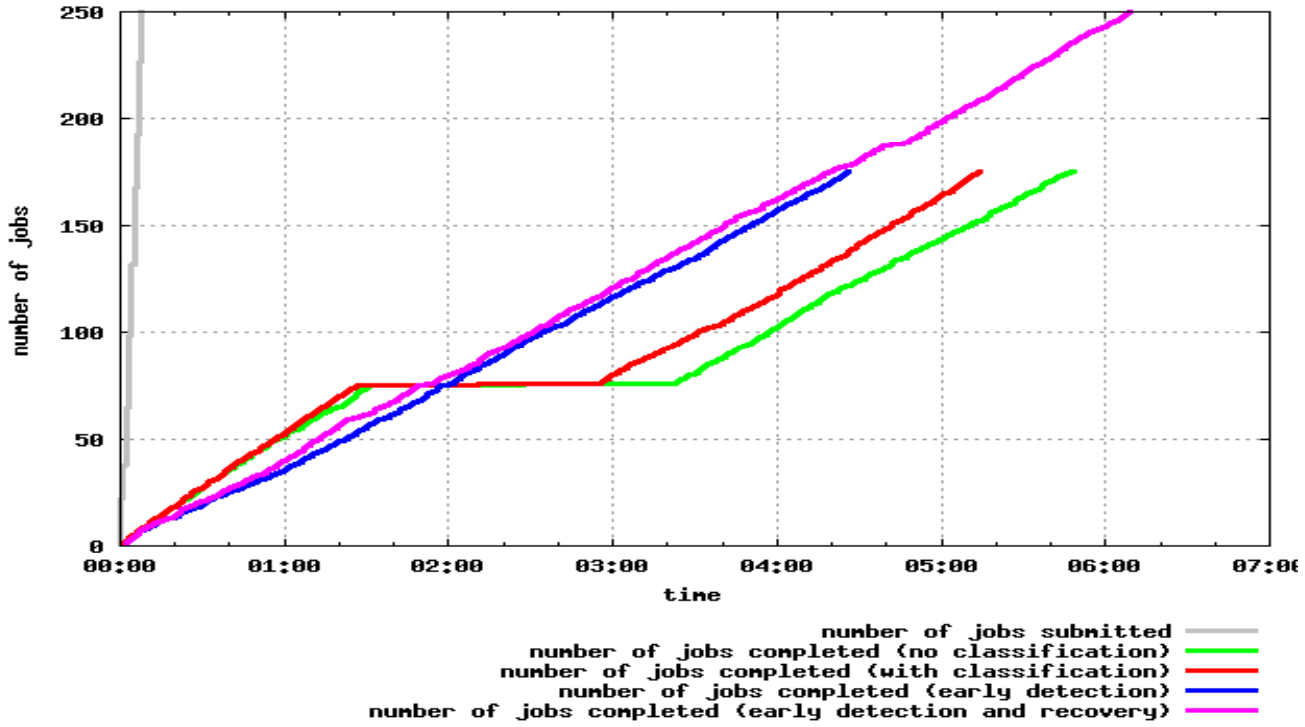


Failure-Aware Scheduling

Scoop data - Hurricane Gustov Simulations

Hundreds of files (250 data transfer operation)

Small (100MB) and large files (1G, 2G



New Transfer Modules

- Verify the successful completion of the operation by controlling checksum and file size.

- for GridFTP, Stork transfer module can recover from a failed operation by restarting from the last transmitted file. 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 checkpoint-enabled transfer module.

- Replacing Globus RFT (Reliable File Transfer)

Agenda

- Error Detection and Error Classification
- ***Data Transfer Operations***
  - Dynamic Tuning
  - Prediction Service
  - Job Aggregation
- Data Migration using Stork
    - Practical example in PetaShare Project
- Future Directions

Tuning Data Transfers

- Latency Wall
  - Buffer Size Optimization
  - Parallel TCP Streams
  - Concurrent Transfers
- User level end-to-end Tuning

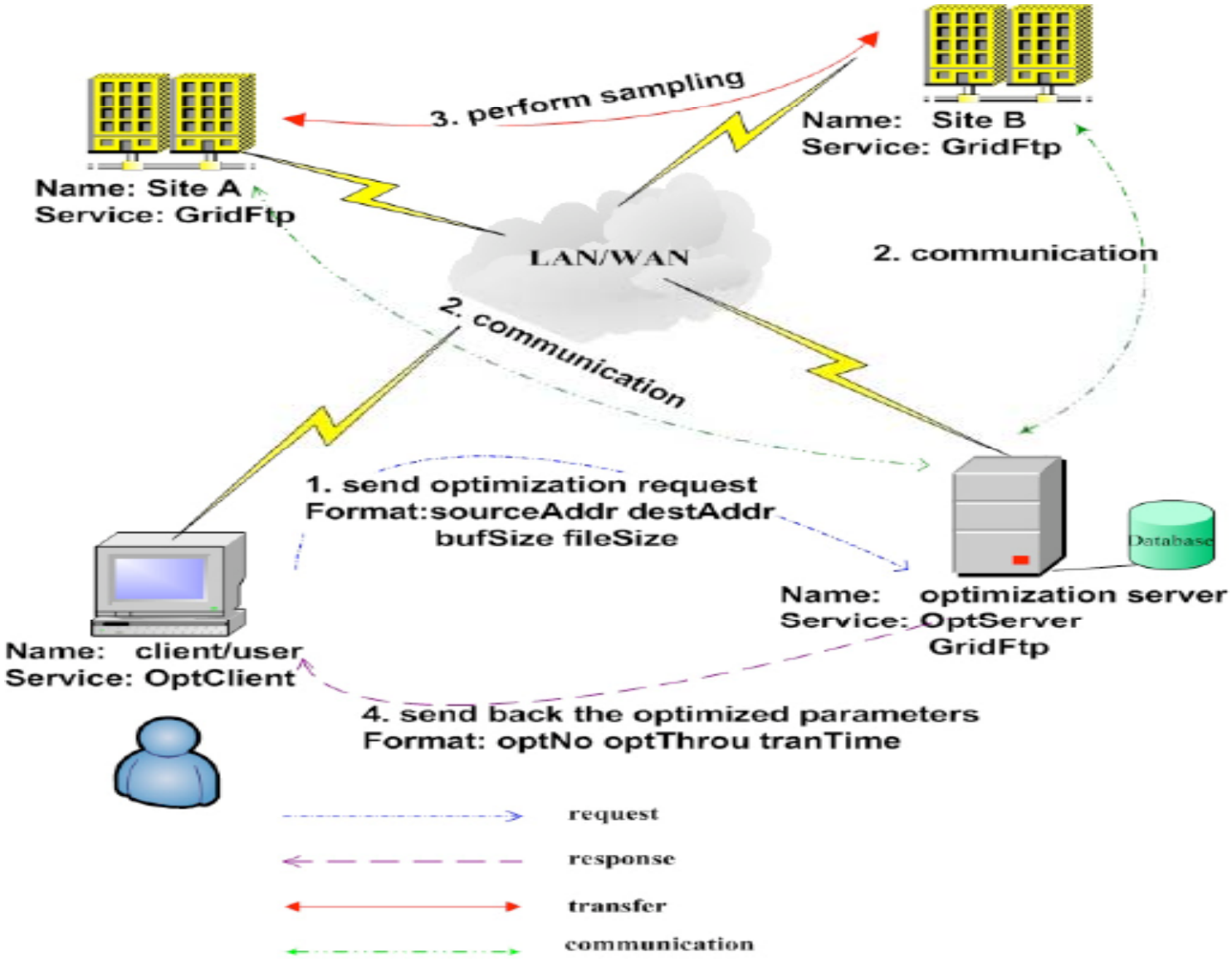
Parallelism

- (1) the number of parallel data streams connected to a data transfer service for increasing the utilization of network bandwidth
- (2) the number of concurrent data transfer operations that are initiated at the same time for better utilization of system resources.

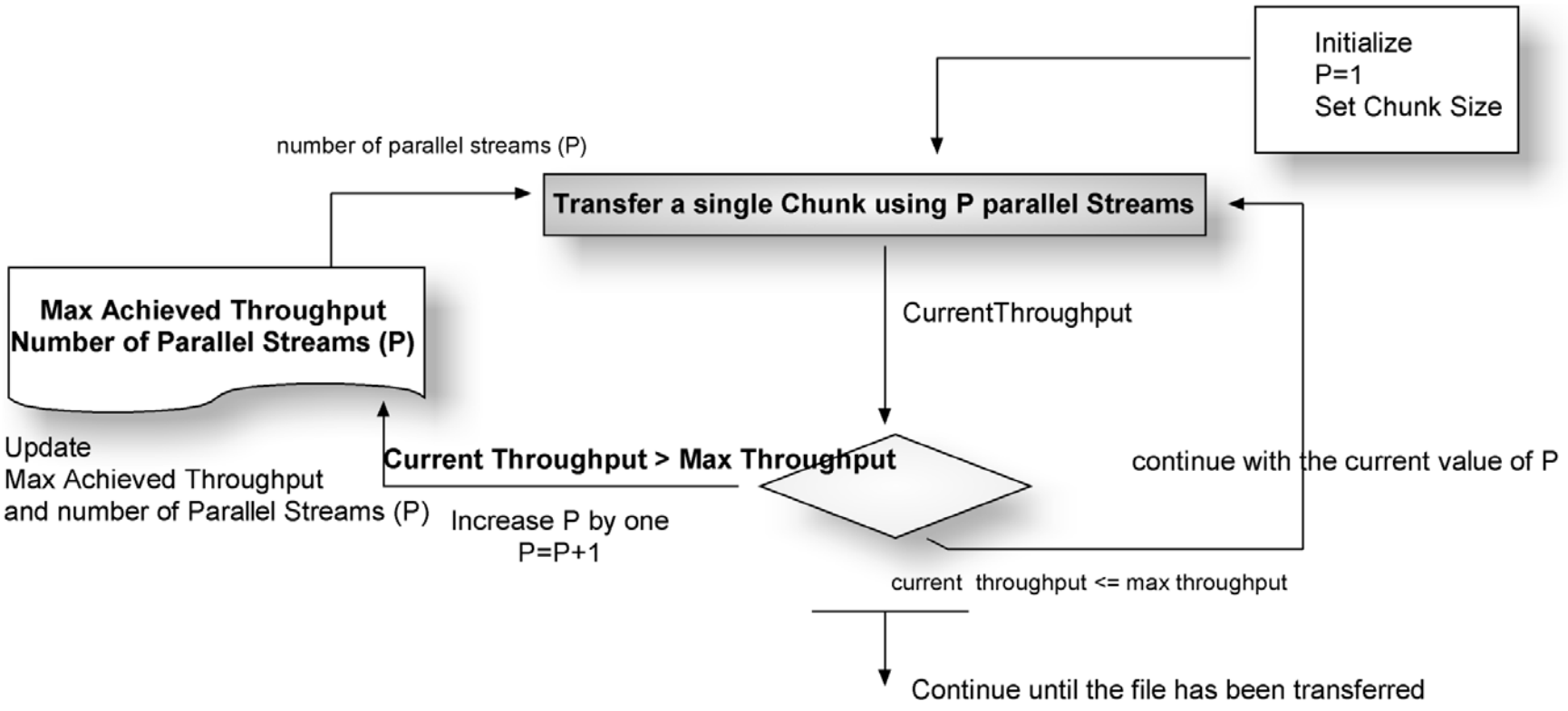
Parameter Estimation

- come up with a good estimation for the parallelism level
  - Network statistics
  - Extra measurement
  - Historical data
- Might not reflect the best possible current settings (Dynamic Environment)

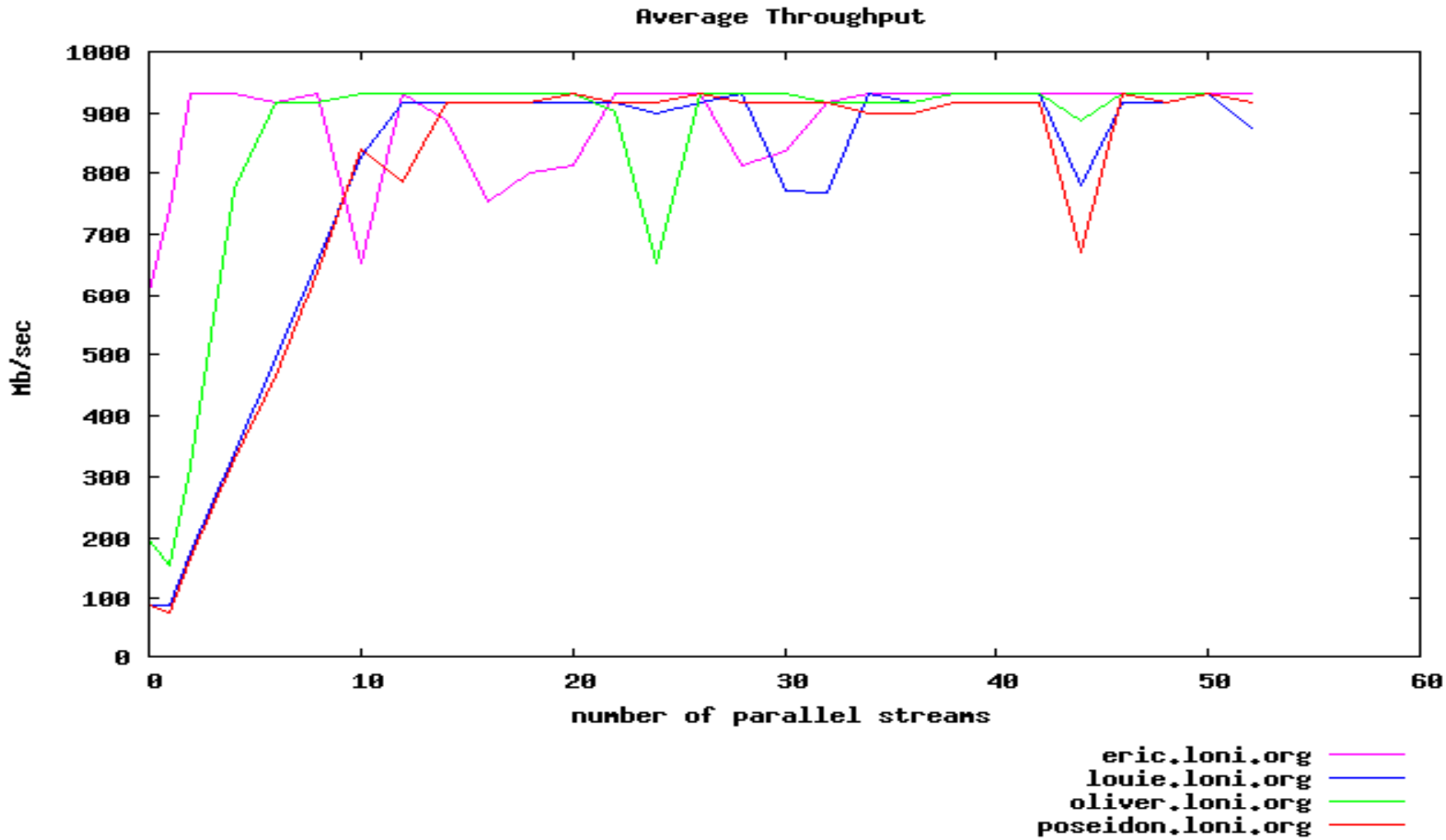
Optimization Service



Dynamic Tuning

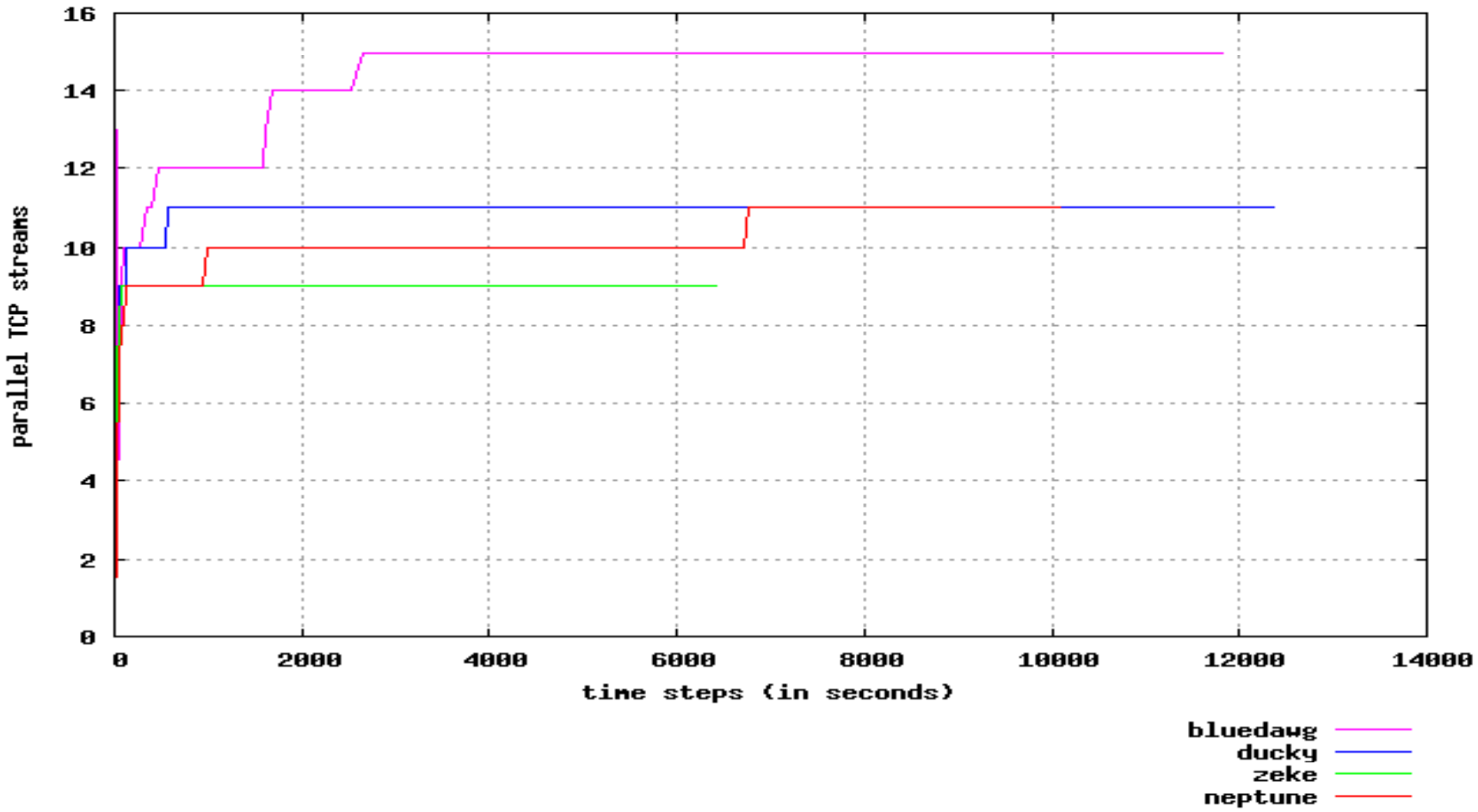


Average Throughput using Parallel Streams



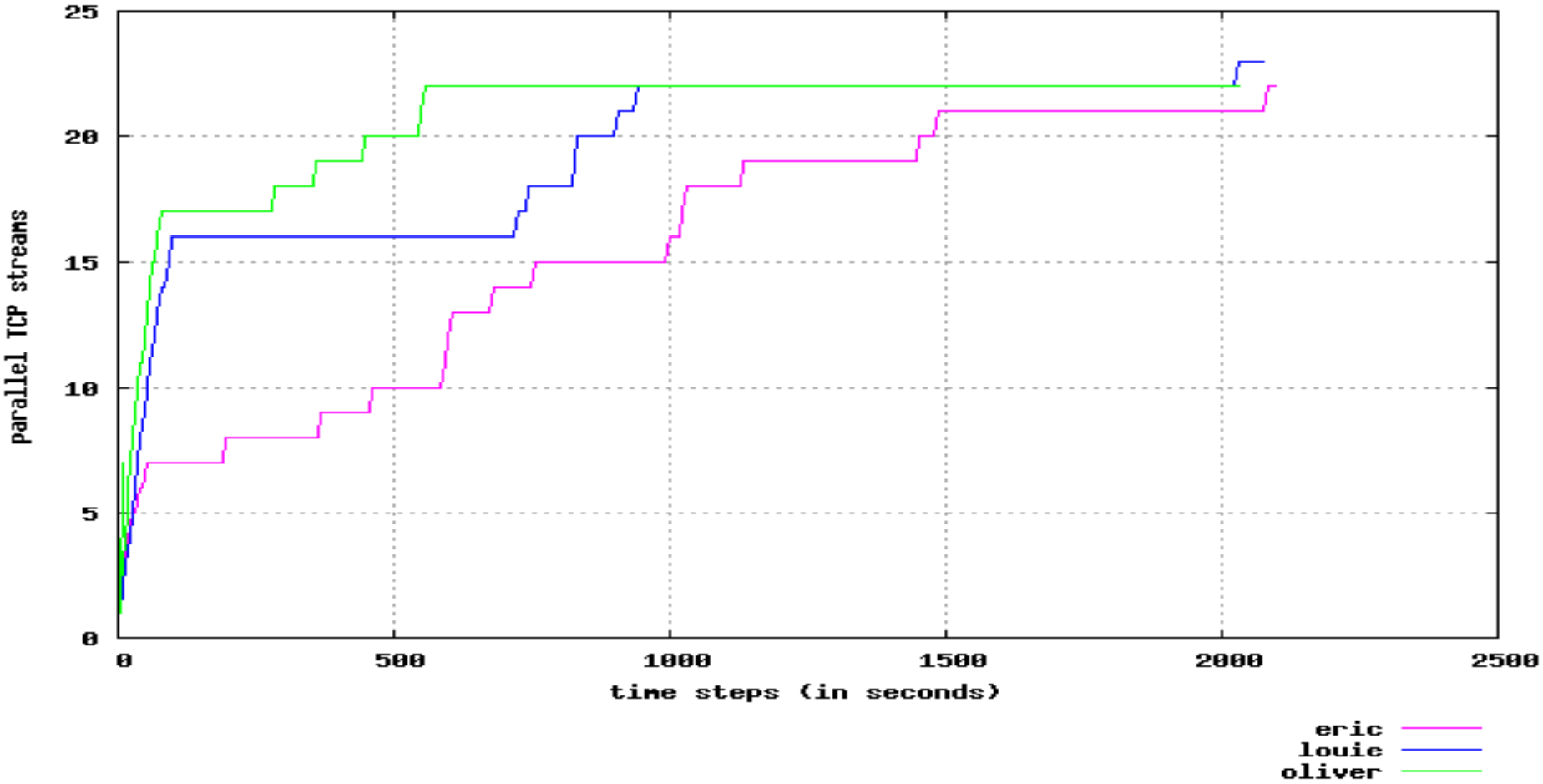
Experiments in LONI (www.loni.org) environment - transfer file to QB from Linux m/c

Dynamic Setting of Parallel Streams



Experiments in LONI (www.loni.org) environment - transfer file to QB from IBM m/c

Dynamic Setting of Parallel Streams



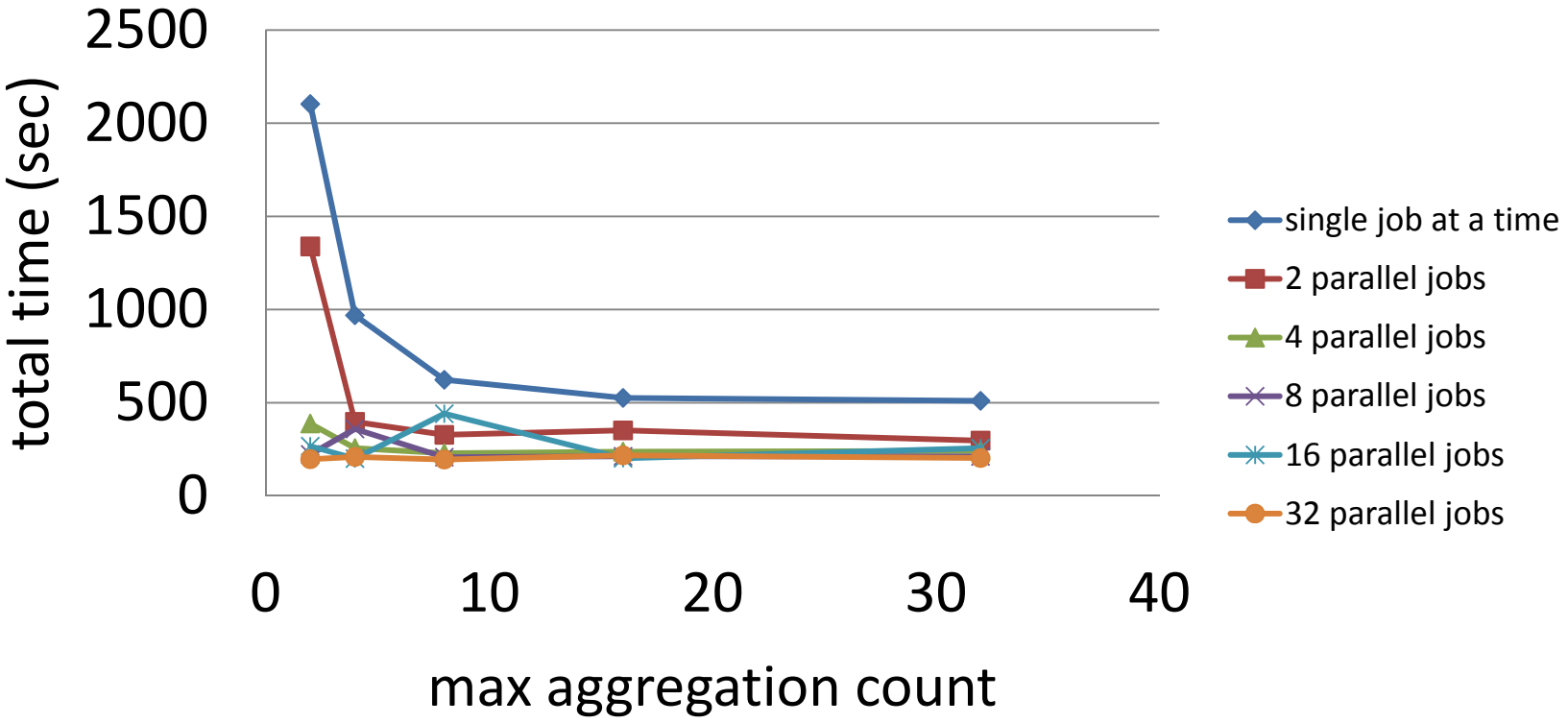
Experiments in LONI (www.loni.org) environment - transfer file to QB from Linux m/c

Job Aggregation

- 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.
- Hence, aggregation is transparent to the user

- We have seen vast performance improvement, especially with small data files,
- simply by combining data placement jobs based on their *source* or *destination* addresses.
  - decreasing the amount of protocol usage
  - reducing the number of independent network connections

Job Aggregation



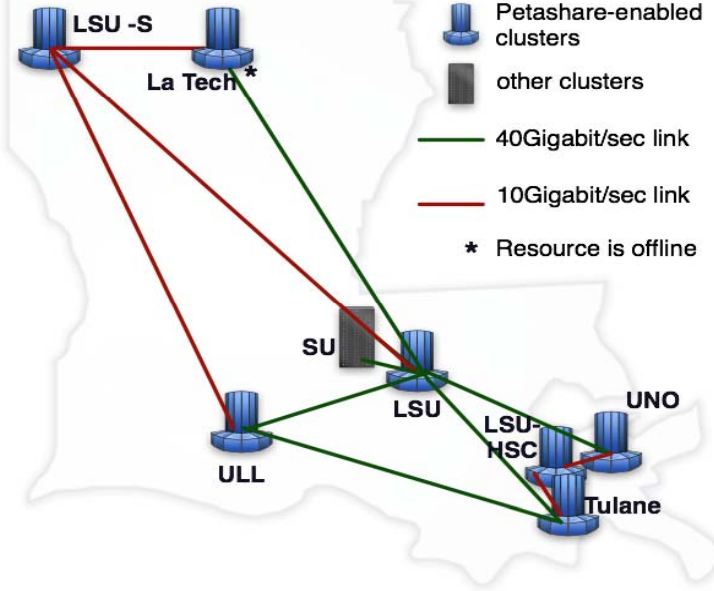
Experiments on LONI (Louisiana Optical Network Initiative) **:**
1024 transfer jobs from Ducky to Queenbee (rtt avg 5.129 ms) - 5MB data file per job

Agenda

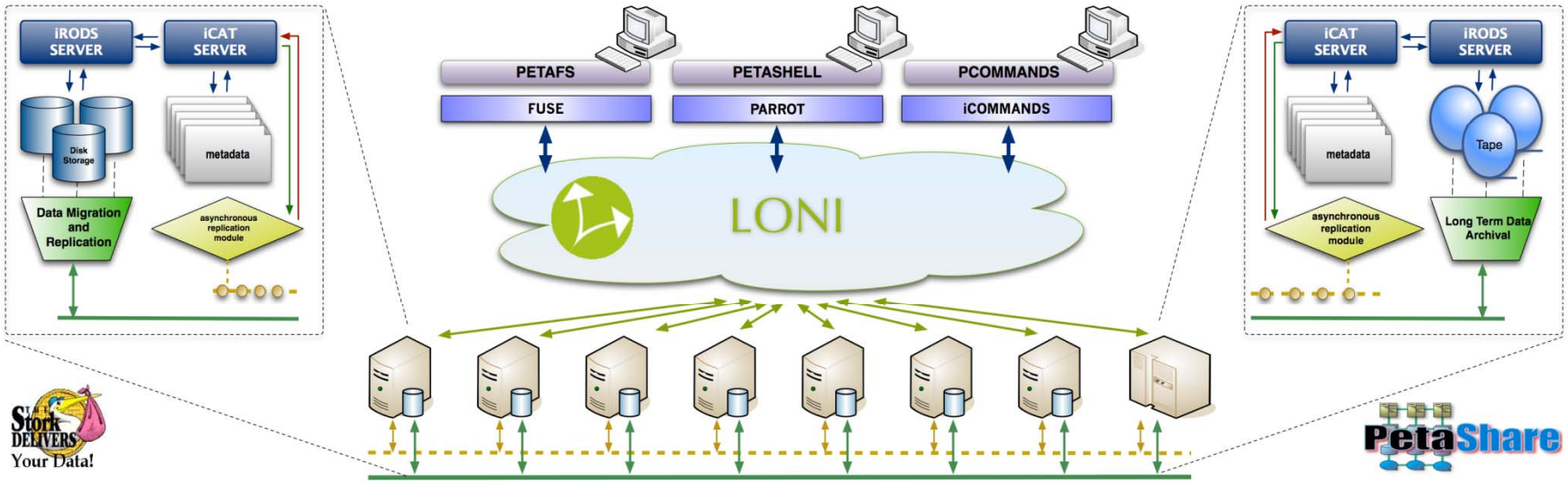
- Error Detection and Error Classification
- Data Transfer Operations
  - Dynamic Tuning
  - Prediction Service
  - Job Aggregation
- **Data Migration using Stork**
    - Practical example in PetaShare Project
- Future Directions

PetaShare

- Distributed Storage for Data Archive
- Global Namespace among distributed resources
- Client tools and interfaces
  - Pcommands
  - Petashell
  - Petafs
  - Windows Browser
  - Web Portal
- Spans among seven Louisiana research institutions
- Manages 300TB of disk storage, 400TB of tape



Broader Impact



Fast and Efficient Data Migration in PetaShare

LSU

Future Directions

Stork: Central Scheduling Framework

- Performance bottleneck
  - Hundreds of jobs submitted to a single batch scheduler, Stork
- Single point of failure

Future Directions

Distributed Data Scheduling

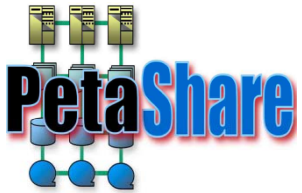
- Interaction between data scheduler
- Manage data activities with lightweight agents in each site
- Better parameter tuning and reordering of data placement jobs
  - Job Delegation
  - peer-to-peer data movement
  - data and server striping
  - make use of replicas for multi-source downloads

Questions?

Team:

Tevfik Kosar kosar@cct.lsu.edu
Mehmet Balman balman@cct.lsu.edu
Dengpan Yin dyin@cct.lsu.edu
Jia "Jacob" Cheng jacobch@cct.lsu.edu

www.cct.lsu.edu



www.petashare.org

www.cybertools.loni.org

www.storkproject.org