# Data Scheduling for Large Scale Distributed Applications

Mehmet Balman and Tevfik Kosar
Center for Computation & Technology and
Department of Computer Science
Louisiana State University

Center for Computation & Technology

AT LOUISIANA STATE UNIVERSITY

## Motivation

- Focus on *data-intensive* distributed computing
- Describe data scheduling approach to manage large scale scientific/commercial applications
- Identify parameters affecting *data transfer*
- Analyze different data placement scenarios
- Develop a strategy to schedule data transfers according to characteristics of dynamically changing distributed environments



## Outline

- Data Intensive Applications
- Possible Problems
- Motivating Remarks
- Data Placement Challenge
  - Data Transfer in a Single Host
  - Data Placement between a Pair of Hosts
  - Data Placements from Multiple Servers to a Single Server
  - Data Placement between Distributed Servers
- Data Intensive Scheduling
- Methodology
- Conclusion and Future Works



## Applications in Science

- Astronomy
  - SuperNova
    - Producing terabytes of data per day
  - LSST(Large Synoptic Survey Telescope)
    - Scanning the sky and producing ten terabytes of data per simulation
- Bimolecular computing
  - Generate huge data sets to be shared between geographically distributed sites
- Climate research
  - Data from every measurement/simulation is more than one terabyte
- High Energy Physics (Cern)
  - Real-time processing of petabytes of data

## Applications in Business

- Credit Card Fraud detection
  - (historical data, analyze transactions, detect fraud)
- Data mining for brokerage and customer services
- Oil and electronic design companies
  - (long term batch processes)
- Medical institutions
  - (computational networks, large image files to be transferred)
- Small transactions, large data transfers, mixed workload (financial applications)
- Complex workflow characteristics
- High capacity, fast storage systems



## Data management Challenges

- CMS (high energy physics simulation)
  - Staging scripts are started to run after the execution (whether or not former job is scheduled or requesting input)
  - Data transfer is not considered as an asynchronous process
  - Storage space is not allocated and staging is not scheduled before data movement
- Blast (bioinformatics)
  - pre-processing / post-processing data transfer scripts (from/to execution site)
  - Data transfer is not optimized
  - Redundant copies, network overload due to many simultaneous transfers, storage space may get full



## Data Placement

*data:*

- characterizes dependencies and connections in the overall process
- should be moved to supply input for the next computing processes
- between geographically separated systems or between different tasks serving for different purposes.

*data placement:*

- coordinately movement of any information between related steps in the general workflow.

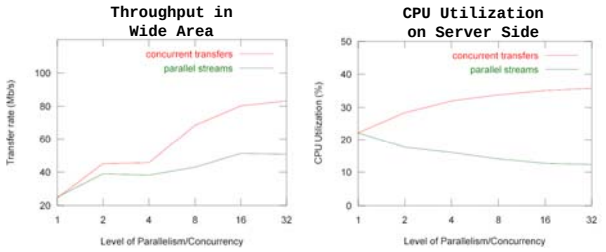
- Scientific experiments using *geographically separated* and *heterogeneous* resources necessitated *transparently accessing* distributed data and analyzing huge collection of information.



## The Problem

- Immense data sets and highly distributed networks
- Data management happens to be more demanding than computational requirements in terms of needed resources
- Data placement considered as a side effect of computation and either embedded inside the computation task or managed by simple scripts.
- *Data placement* is *a part of the job scheduling* dilemma and it should be considered as a crucial factor affecting resource selection and scheduling in distributed computing.

## Motivating Remarks

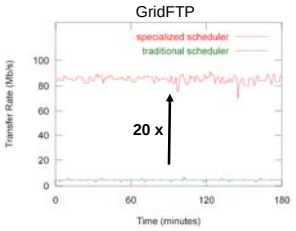



- Definitions:
  - Concurrency: transfer 1 file using n parallel streams
  - Parallelism: transfer n files at the same time



## Motivating Remarks

- Optimization according to:
  - Job type
    - transfer, allocate, release, locate..
  - protocols to use
  - Available storage space
  - Best parallelism level
  - Best route
  - Best network parameters
    - tcp buffer size
    - I/O block size
    - # of concurrent streams






## Approach

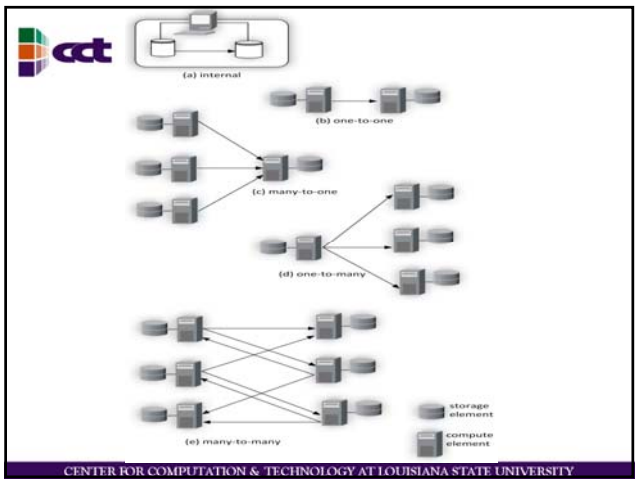
- Analyze data placement for possible use cases
- Start from very simple scenarios
- Discover key attributes affecting performance in each scenario
- Expand to more complex use cases

- Use key attributes affecting data placement to optimize data transfer during the scheduling of distributed jobs.



## Approach

- Data Transfer in a Single Host
- Data Placement between a Pair of Hosts
- Data Placements from Multiple Servers to a Single Server
- Data Placement between Distributed Servers






## Data Transfer in a Single Host

- Space limitations
- Storage management constraints
- Administrative decisions
- Performance implications
- Data formatting

- Dataware house applications
- Data hierarchy:
  - application, database, historical information



## Data Transfer in a Single Host

- Backup
  - multiplexing several files to a single tape device
- I/O scheduling in traditional Operating Systems
  - (Multiple disks, multiple data channels).

- Fairness between multiple I/O requests
- Real-time requirements
- Load balancing
- Memory cache
- Partitioning data



## Data Transfer in a Single Host

- Server load, CPU and memory utilization
- Available storage space, and space reservation
- Multiplexing and partitioning techniques
- Buffer size and cache usage
- File system performance (e.g., block size, etc.)
- Protocol and device performance
- I/O subsystem and scheduling.

## Data Placement between a Pair of Hosts

- Downloading / uploading a file over network
- Environment:
  - Changing network conditions, heterogeneous operating systems, available resources
- Simultaneous TCP streams

  The number of maximum allowable concurrent connections depends on both network and server characteristics, and selecting an appropriate value for each channel that will optimize the transfer in long term is a challenging issue in networked environments.



## Data Placement between a Pair of Hosts

- Use of multiple connections is declared as a TCP-unfriendly activity, but outstanding performance results were obtained by employing simultaneous streams in the recent experiments
- GridFTP: multiple TCP streams and configurable buffer sizes
- Specialized TCP protocols (fastTCP, sTCP) which changes TCP features for large data transfers
- Network tools (cFlowd, NetFlow) to measure the network metrics



## Data Placement between a Pair of Hosts

- Different structures in communication layer
- Network Characteristics
  - LAN, WAN (Congestion), high-speed with channel reservation

Expanding parameters:

- simultaneous TCP connections,
- TCP tuning (send/receive buffer size, TCP window size, MTU, etc.),
- Data transmission protocol performance (GridFTP, ftp, etc.).
- ordering data transfer task for maximum throughput



## Data Placement from Multiple Servers to a Single Server

- Data objects are stored on separate storage servers (i.e. load balancing, reliability, performance.)
- Concurrency is obtained by opening multiple TCP streams between a pair of server. On the other hand, parallel connections are maintained by multiple threads serving each data transfer from different sites.
- Parallel TCP connections
- Network properties such as bandwidth, failure rate, congestion, and server parameters like CPU load, memory and communication protocol have influences on parallelization.

## Data Placement from Multiple Servers to a Single Server

- Ordering data placement tasks:
  - Optimize the transfer so that maximum number of tasks can start execution
    (i.e. each task is waiting separate file sets)
  - Small files over high bandwidth connections
    - Maximum number of transferred files
  - Large files and closer storage servers
- Maximize throughput
- Different user with different priorities
- Multiple replicas of data in different sites
  - Transfer different parts from different replica servers
- Merging several transfers together



## Data Placement from Multiple Servers to a Single Server

- Different characteristics when uploading and downloading
  - (read / write overhead )

Additional factors:

- Parallel network connections
- Network bandwidth and latency
- Use of replicated data and caching



## Data Placement between Distributed Servers

- Data servers are distributed on different locations and available network is usually shared like Internet; therefore, minimizing the network cost by selecting the path which gives maximum bandwidth and minimum network delay to obtain high speed transfer will increase the overall throughput
- Distributed and shared resources
- Dynamically changing network
  - (Bandwidth , Latency)
  - predicted network parameters
- Server attributes
  - (CPU load, available disk space)



## Data Placement between Distributed Servers

- Data movement jobs are competing with each other
- We necessitate a central scheduling mechanism which can collect information from separate sites and schedule data placement jobs such that maximum throughput in minimum time is achieved.

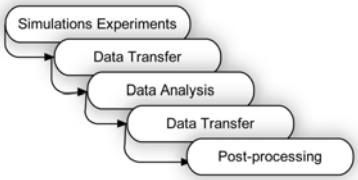
| | In Single Host | Between a Pair of Hosts | Multiple Servers to Single Server | Between Distributed Servers |
|---|---|---|---|---|
| Available Storage Space | ✓ | ✓ | ✓ | ✓ |
| CPU Load and Memory Usage | ✓ | ✓ | ✓ | ✓ |
| Transfer Protocol Performance | | ✓ | ✓ | ✓ |
| Number of Concurrent Connections | | ✓ | ✓ | ✓ |
| Network Bandwidth and Latency | | | ✓ | ✓ |
| Number of Parallel Streams | | | ✓ | ✓ |
| Ordering of Data Placement Tasks | | | | ✓ |



## Data Flow

We can simply classify the steps in a large scale application as follows;

- Obtain data from experiments or simulate to generate data
- Transfer data and organize for pre-processing
- Data analysis and recognition
- Move data for post-processing
- Visualization or interactive analysis



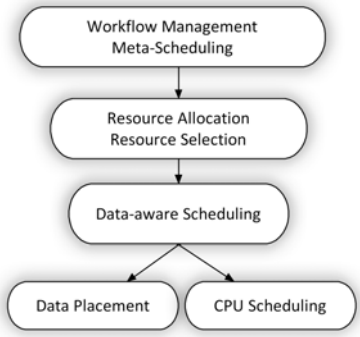



## Data Aware System Model

- We necessitate workflow managers to define the dependencies of execution sequences in the application layer.
- Higher level planners are used to select and match appropriate resources.
- A data-aware scheduler is required to organize requests and schedule them not only considering computing resources but also data movement issues.
- We have data placement modules and traditional CPU schedulers to serve upper layers and complete job execution or data transfer.



## Data Aware System Model



## Data Aware System Model

- Data placement jobs have been categorized in different types (transfer, allocate, release, remove, locate, register, unregister) and it has been stated that each category has different importance and optimization characteristics.

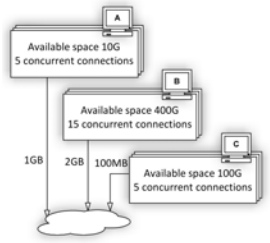


## Methodology

- First, concentrate on simple data placement scenarios.
- By the help of experiments and measurements in real life situations, investigate crucial factors influencing data transfer operations.
- Extend to more complex cases by searching new effective attributes



## Methodology




File 1: 50MB from server A to server C
File 2: 10G from server C to server A
File 3: 10G from server B to server C
File 3: 50G from server B to server C
File 3: 100G from server B to server C



## Methodology

- Study characteristics of data placement jobs to provide a model for scheduling.
- Develop a strategy to schedule data transfers in distributed environments.

## Conclusion

- Define data scheduling approach for large scale scientific /commercial applications
- Identify structures in the overall model
- Analyze different scenarios for possible use cases
- Discover key attributes affecting performance
- Use key attributes during scheduling to optimize data transfer



## Thanks
## Questions & Comments

**•Related Sites**

- **•STORK:** http://www.cs.wisc.edu/condor/stork/
- **•Petashare:** www.petashare.org

**This work was supported by :**
•NSF grant CNS-0619843
•Louisiana BoR-RCS grant LEQSF (2006-09)-RD-A-06
•CCT General Development Program



## Previous Work

**STORK:**

**A Scheduler for Data Placement Activities**

Data placement activities as **"first class citizens"** in the Grid just like the computational jobs.

**http://www.cs.wisc.edu/condor/stork/**


**Support for heterogeneously**
Protocol translation using Stork memory buffer/Disk Cache
**Flexible Job Representation and Multilevel Policy Support**
**Run-time adaptation**
Dynamic protocol selection, Run-time Protocol Auto-tuning
**Failure Recovery and Efficient Resource Utilization**



## Petashare Project

www.petashare.org

**An innovative distributed data archival, analysis and visualization instrument for data intensive collaborative research.**

**"PetaShare** will enable transparent handling of underlying data sharing, archival, and retrieval mechanisms; and will make data available to the scientists for analysis and visualization on demand."

**PetaShare** will respond to an urgent need of scientists who are working with large data generation, sharing and collaboration requirements."

## Thank you

»Questions?



## Acknowledgement

This work was supported by

NSF grant CNS-0619843,
Louisiana BoR-RCS grant LEQSF
(2006-09)-RD-A-06,
and CCT General Development Program.

CENTER FOR COMPUTATION & TECHNOLOGY AT LOUISIANA STATE UNIVERSITY