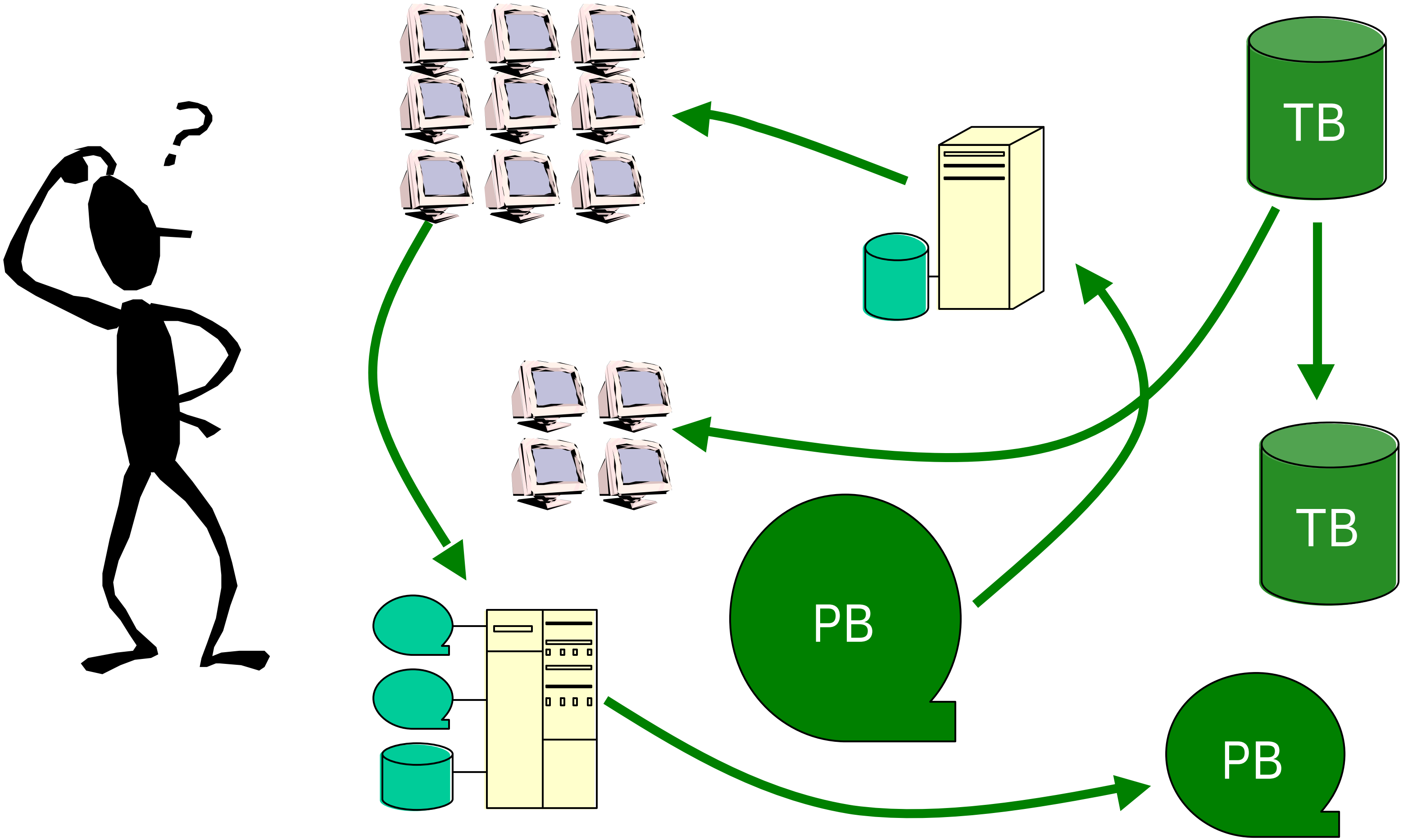
The Stork DELIVERS Your Data!

STORK: Making Data Placement a First Class Citizen in the Grid

Tevfik Kosar
University of Wisconsin-Madison

May 25th, 2004
CERN

Need to move data around..



While doing this..

- Locate the data
- Access heterogeneous resources
- Face with all kinds of failures
- Allocate and de-allocate storage
- Move the data
- Clean-up everything

➡ **All of these need to be done reliably and efficiently!**

Stork

- A scheduler for data placement activities in the Grid
- What Condor is for computational jobs, Stork is for data placement
- Stork comes with a new concept:

  "Make data placement a first class citizen in the Grid."

Outline

- Introduction
- The Concept
- Stork Features
- Big Picture
- Case Studies
- Conclusions

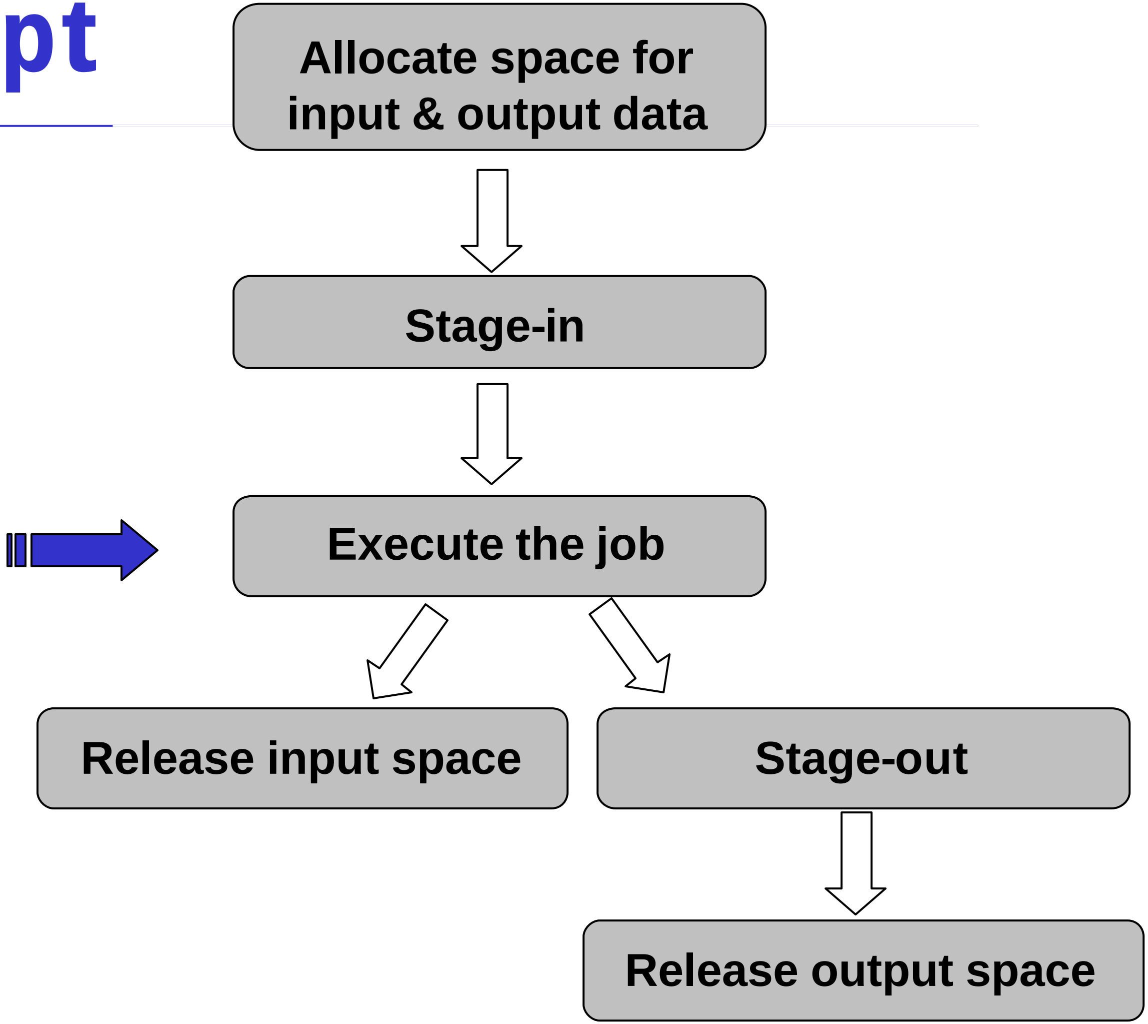
The Concept

- Stage-in
- Execute the Job
- Stage-out

▨ Individual Jobs

The Concept

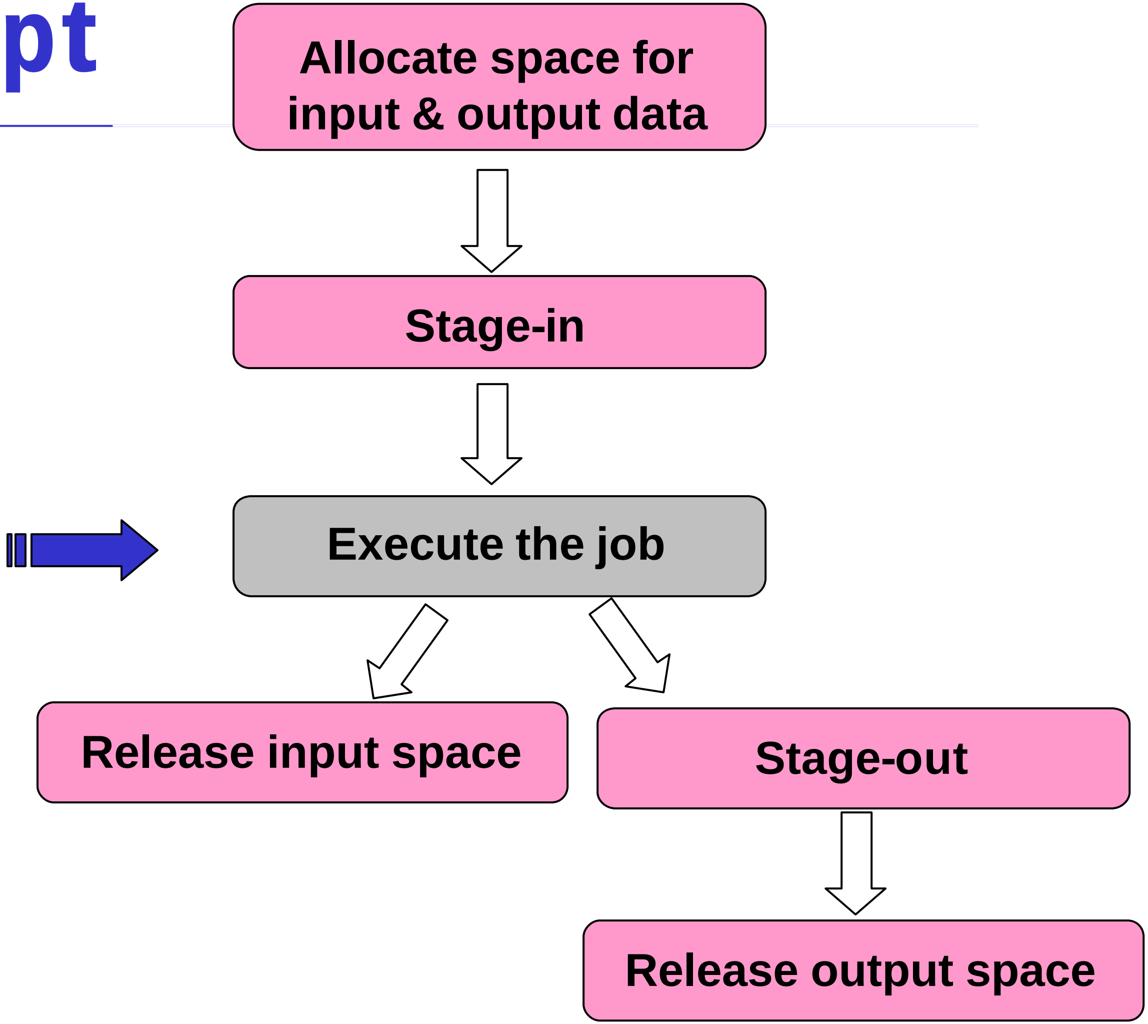
- Stage-in
- Execute the Job
- Stage-out



▨ Individual Jobs

The Concept

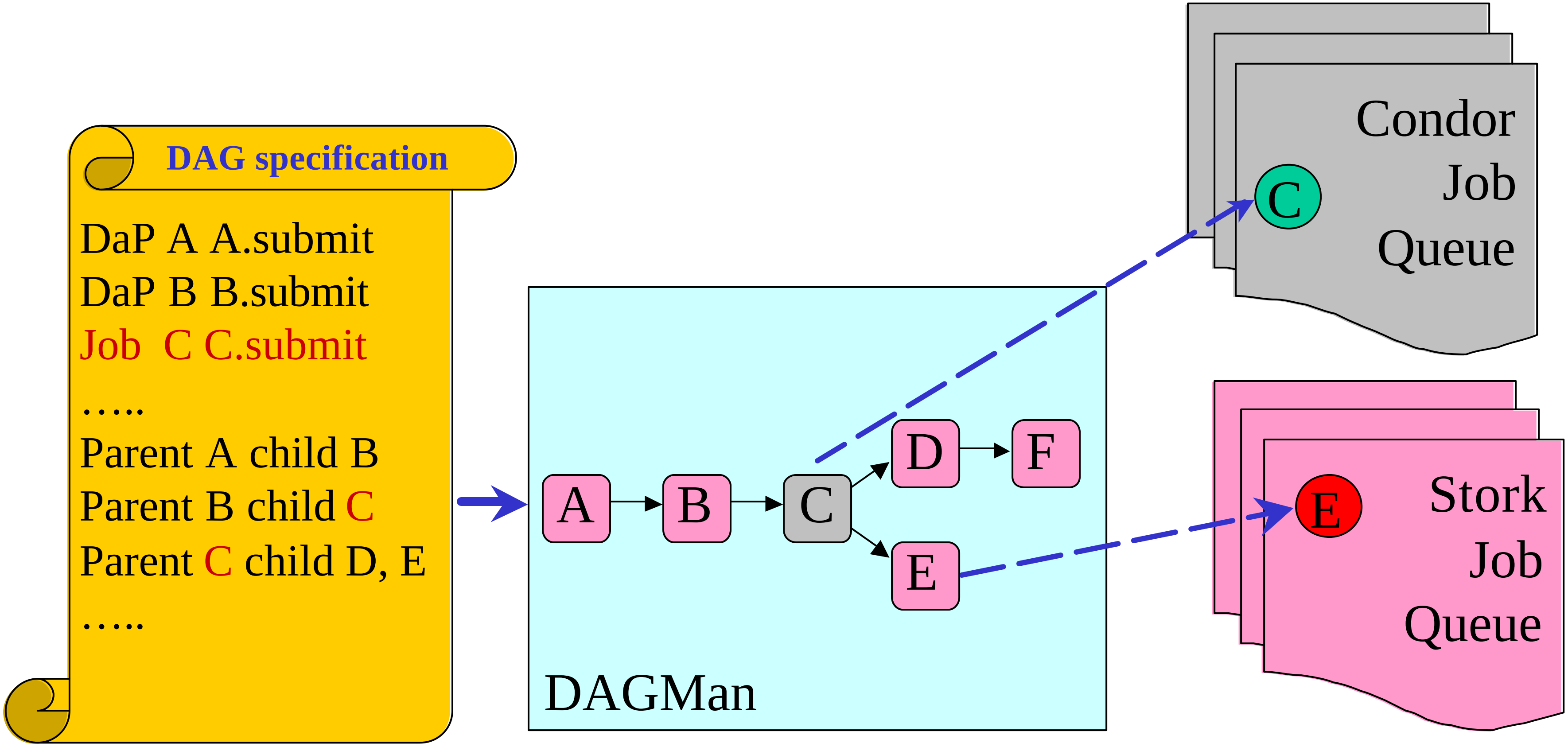
- Stage-in
- Execute the Job
- Stage-out



🟥 Data Placement Jobs

▨ Computational Jobs

The Concept



DAG specification

DaP A A.submit
DaP B B.submit
Job C C.submit
…..
Parent A child B
Parent B child C
Parent C child D, E
…..

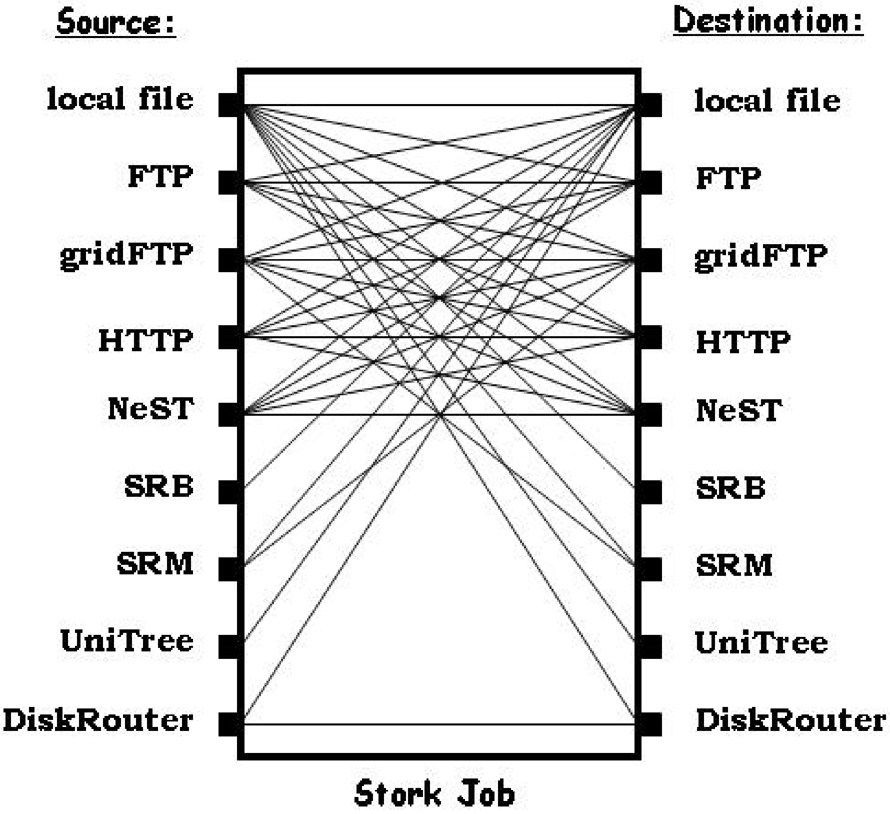
Why Stork?

- Stork understands the characteristics and semantics of data placement jobs.
- Can make smart scheduling decisions, for reliable and efficient data placement.

Understanding Job Characteristics & Semantics

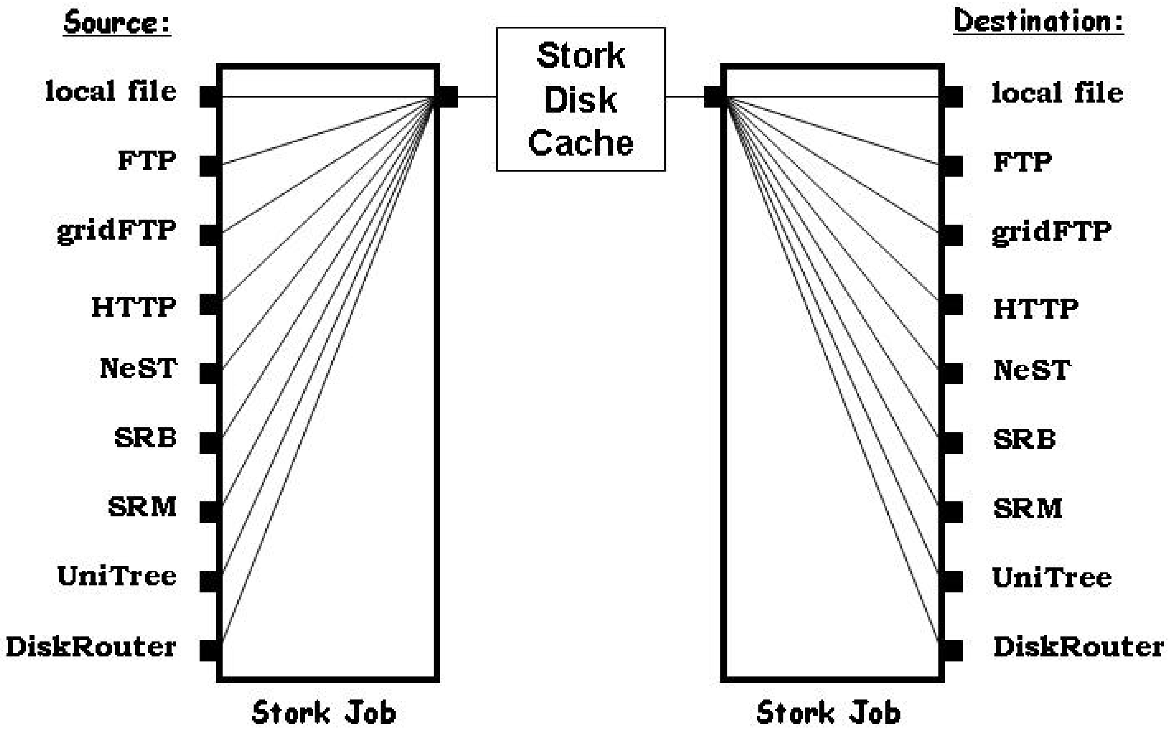
- Job_type = transfer, reserve, release?
- Source and destination hosts, files, protocols to use?
  - Determine concurrency level
  - Can select alternate protocols
  - Can select alternate routes
  - Can tune network parameters (tcp buffer size, I/O block size, # of parallel streams)
  - …

Support for Heterogeneity



Protocol translation using Stork memory buffer.

Support for Heterogeneity



Protocol translation using Stork Disk Cache.

Flexible Job Representation and Multilevel Policy Support

```
[
        Type     = “Transfer”;
        Src_Url = “srb://ghidorac.sdsc.edu/kosart.condor/x.dat”;
        Dest_Url = “nest://turkey.cs.wisc.edu/kosart/x.dat”;
        ……
        ……
        Max_Retry = 10;
        Restart_in = “2 hours”;
]
```

Failure Recovery and Efficient Resource Utilization

- Fault tolerance
  - Just submit a bunch of data placement jobs, and then go away..
- Control number of concurrent transfers from/to any storage system
  - Prevents overloading
- Space allocation and De-allocations
  - Make sure space is available

Run-time Adaptation

- Dynamic protocol selection

```
[
  dap_type = “transfer”;
  src_url    = “drouter://slic04.sdsc.edu/tmp/test.dat”;
  dest_url  = “drouter://quest2.ncsa.uiuc.edu/tmp/test.dat”;
  alt_protocols = “nest-nest, gsiftp-gsiftp”;
]

[
  dap_type = “transfer”;
  src_url    = “any://slic04.sdsc.edu/tmp/test.dat”;
  dest_url  = “any://quest2.ncsa.uiuc.edu/tmp/test.dat”;
]
```

Run-time Adaptation

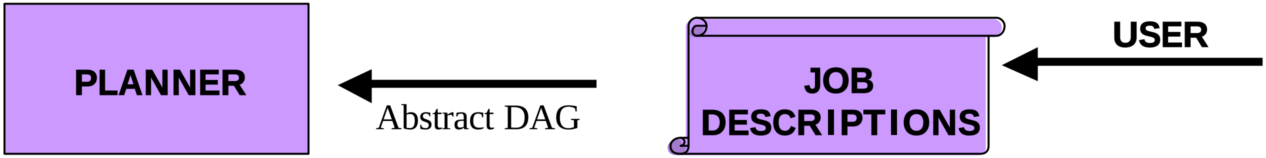
- Run-time Protocol Auto-tuning

```
[
    link       = “slic04.sdsc.edu – quest2.ncsa.uiuc.edu”;
    protocol = “gsiftp”;

    bs         = 1024KB;        //block size
    tcp_bs    = 1024KB;        //TCP buffer size
    p           = 4;
]
```

Outline

- Introduction
- The Concept
- Stork Features
- Big Picture
- Case Studies
- Conclusions

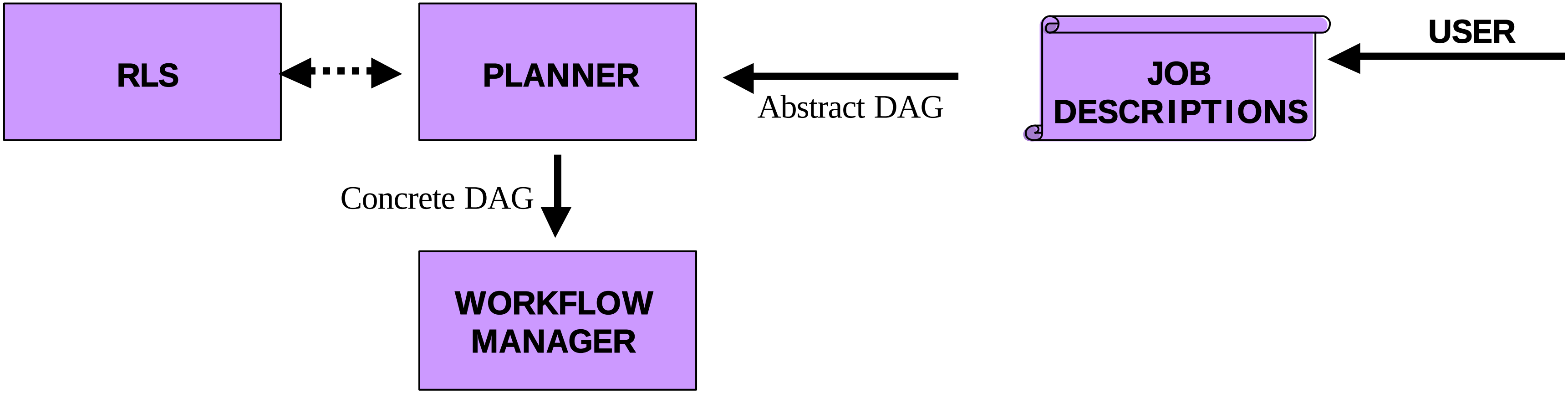


PLANNER

Abstract DAG

JOB DESCRIPTIONS

USER



RLS

PLANNER

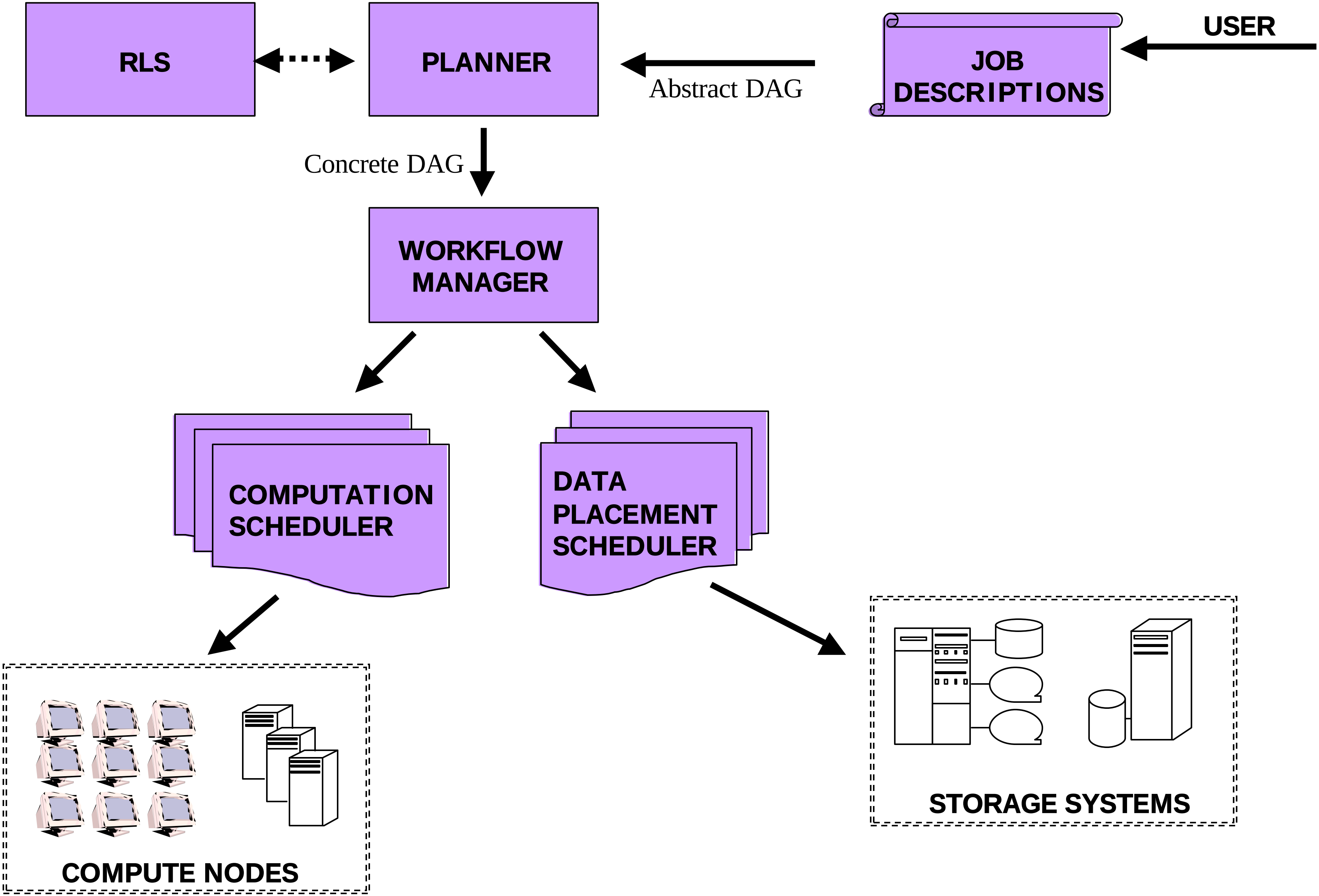
Abstract DAG

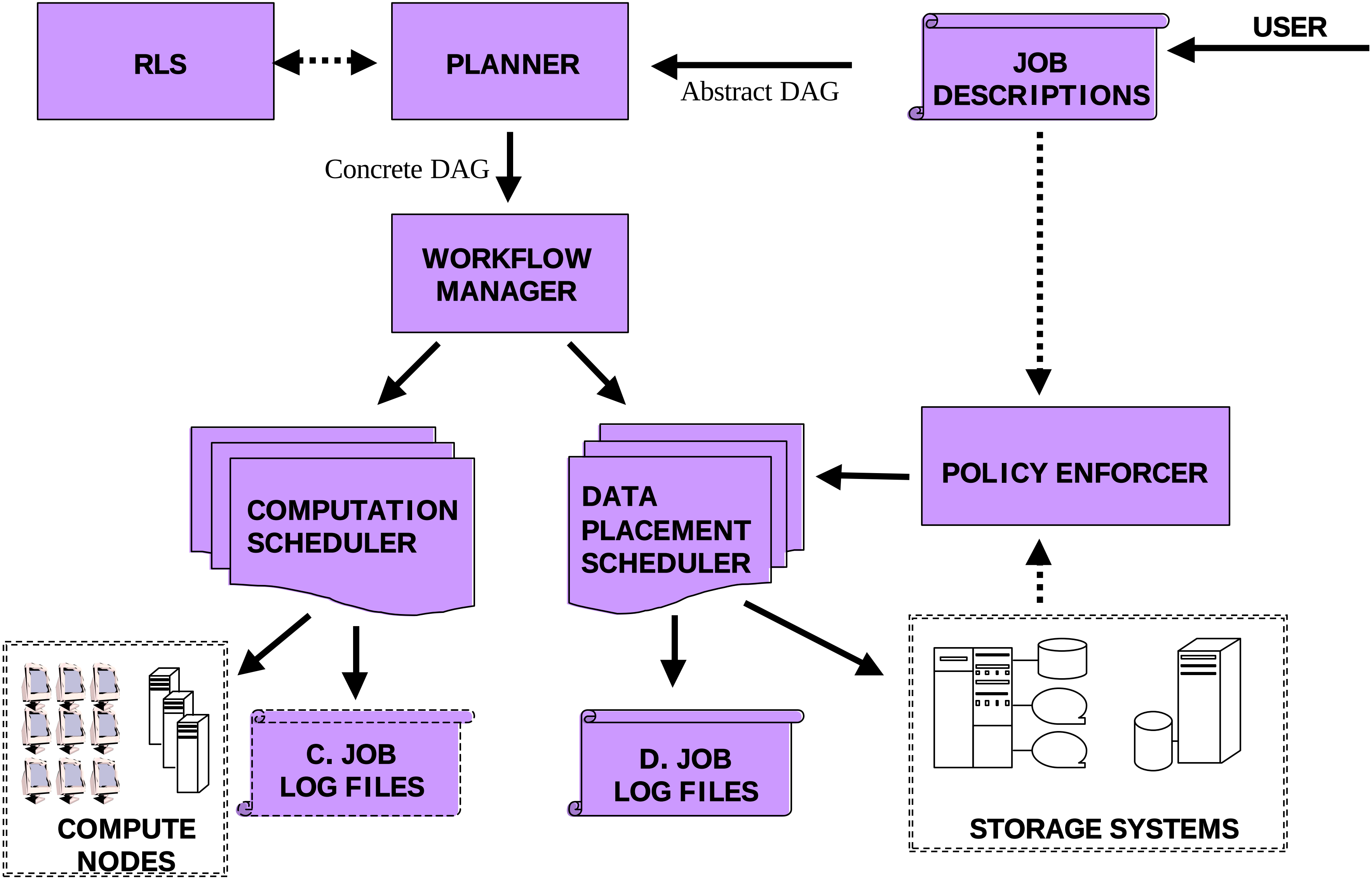
JOB DESCRIPTIONS

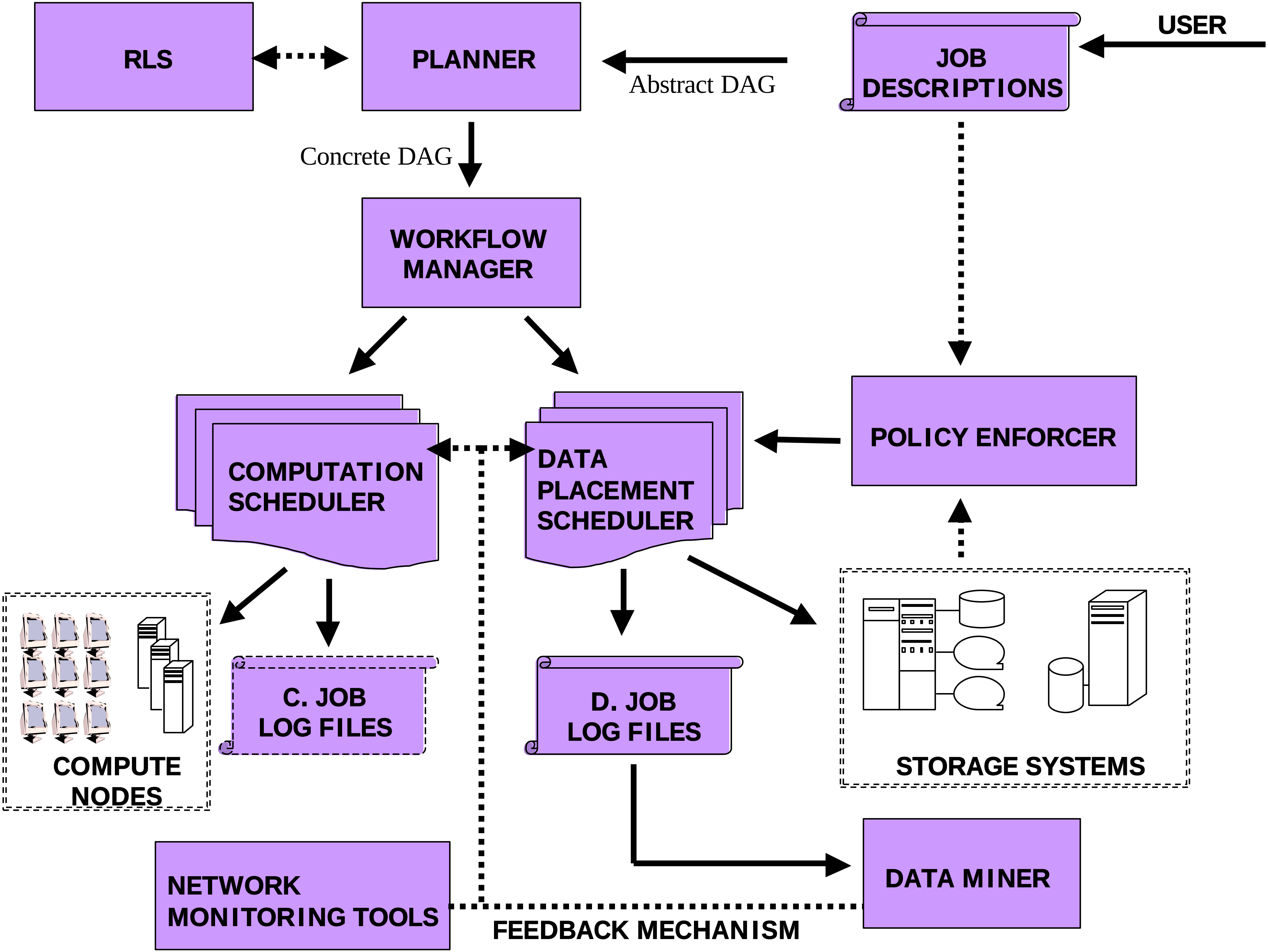
USER

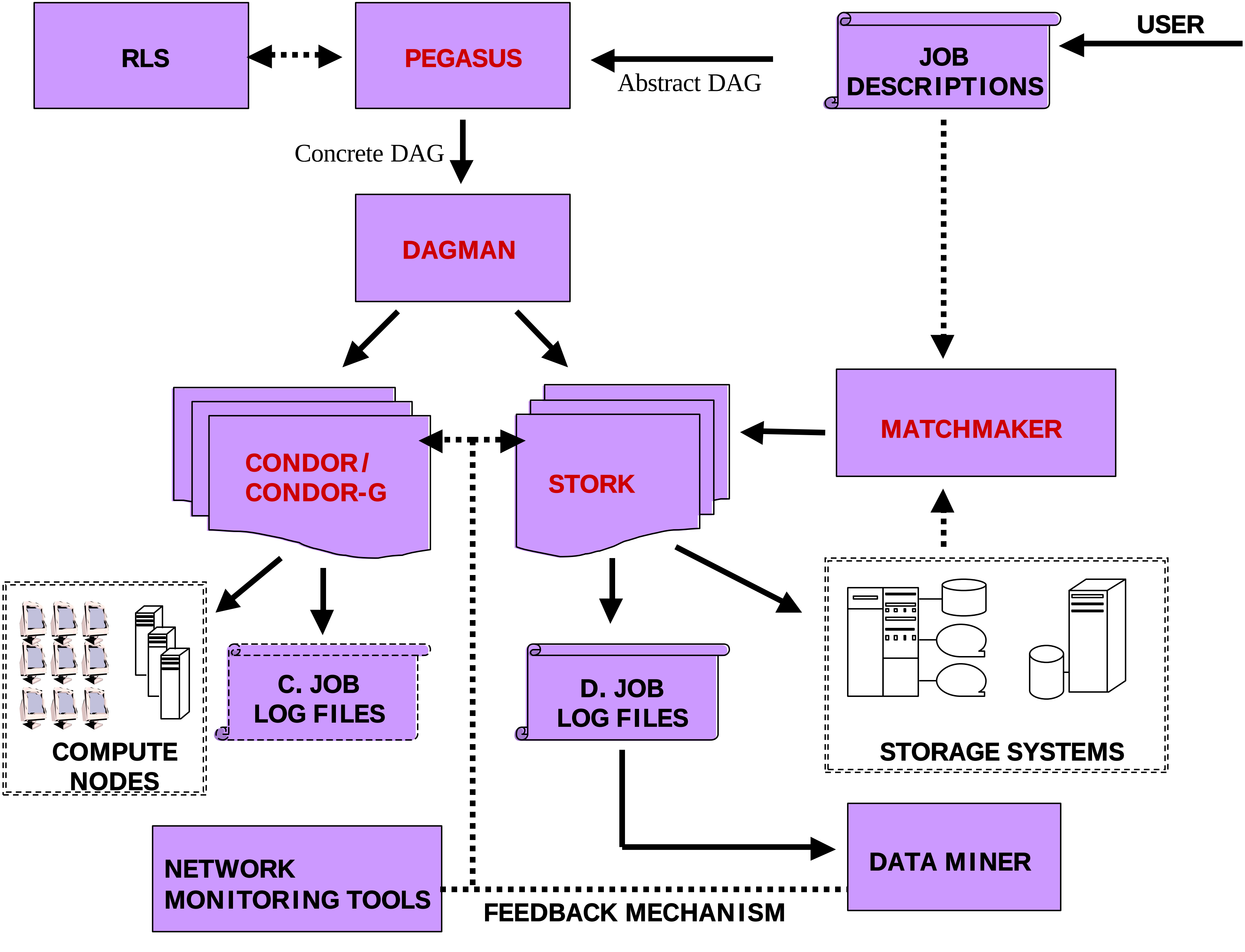
Concrete DAG

WORKFLOW MANAGER


RLS
PLANNER
Abstract DAG
JOB DESCRIPTIONS
USER
Concrete DAG
WORKFLOW MANAGER
COMPUTATION SCHEDULER
DATA PLACEMENT SCHEDULER
COMPUTE NODES
STORAGE SYSTEMS


RLS
PLANNER
USER
JOB DESCRIPTIONS
Abstract DAG
Concrete DAG
WORKFLOW MANAGER
COMPUTATION SCHEDULER
DATA PLACEMENT SCHEDULER
POLICY ENFORCER
COMPUTE NODES
C. JOB LOG FILES
D. JOB LOG FILES
STORAGE SYSTEMS


RLS
PLANNER
Abstract DAG
USER
JOB DESCRIPTIONS
Concrete DAG
WORKFLOW MANAGER
COMPUTATION SCHEDULER
DATA PLACEMENT SCHEDULER
POLICY ENFORCER
COMPUTE NODES
C. JOB LOG FILES
D. JOB LOG FILES
STORAGE SYSTEMS
NETWORK MONITORING TOOLS
DATA MINER
FEEDBACK MECHANISM

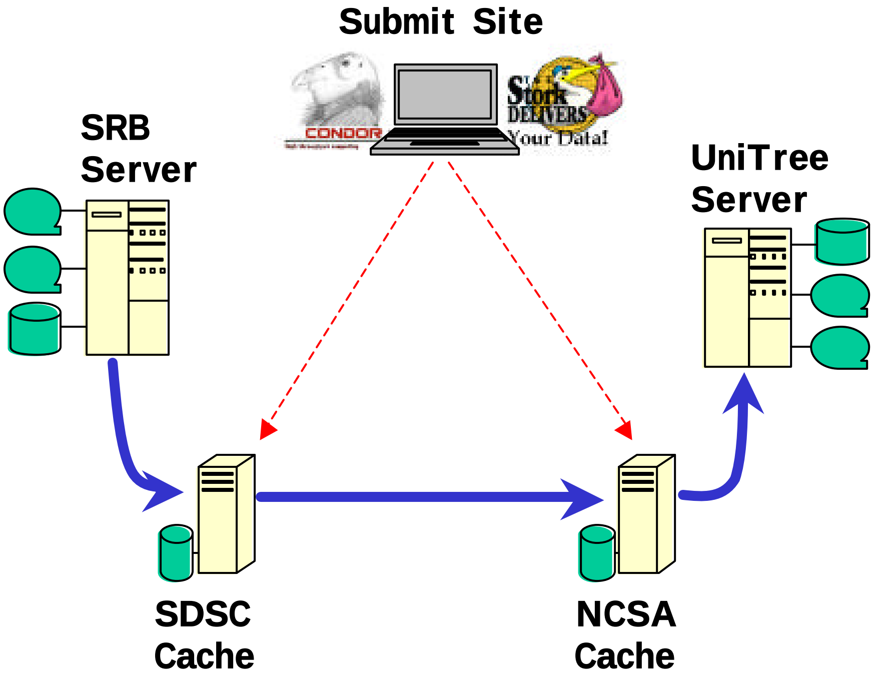

RLS
PEGASUS
Abstract DAG
JOB DESCRIPTIONS
USER
Concrete DAG
DAGMAN
MATCHMAKER
CONDOR / CONDOR-G
STORK
COMPUTE NODES
C. JOB LOG FILES
D. JOB LOG FILES
STORAGE SYSTEMS
NETWORK MONITORING TOOLS
DATA MINER
FEEDBACK MECHANISM

Outline

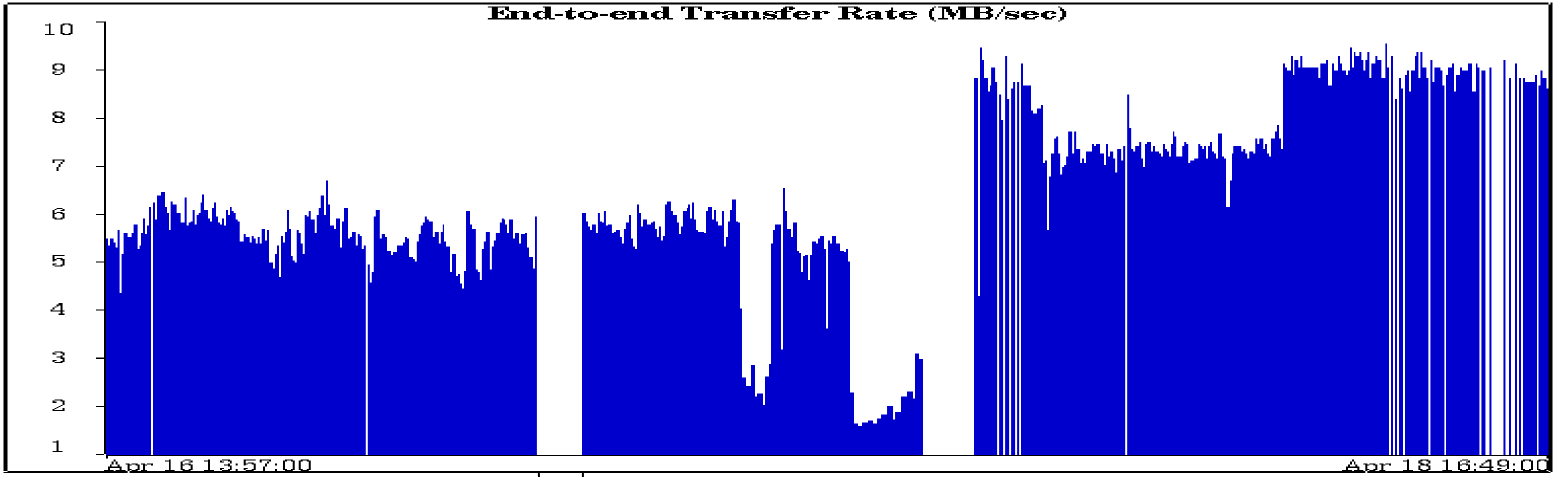
- Introduction
- The Concept
- Stork Features
- Big Picture
- Case Studies
- Conclusions

Case Study I:
SRB-UniTree Data Pipeline

- Transfer ~**3 TB** of DPOSS data from SRB @SDSC to UniTree @NCSA
- A data transfer pipeline created with Stork

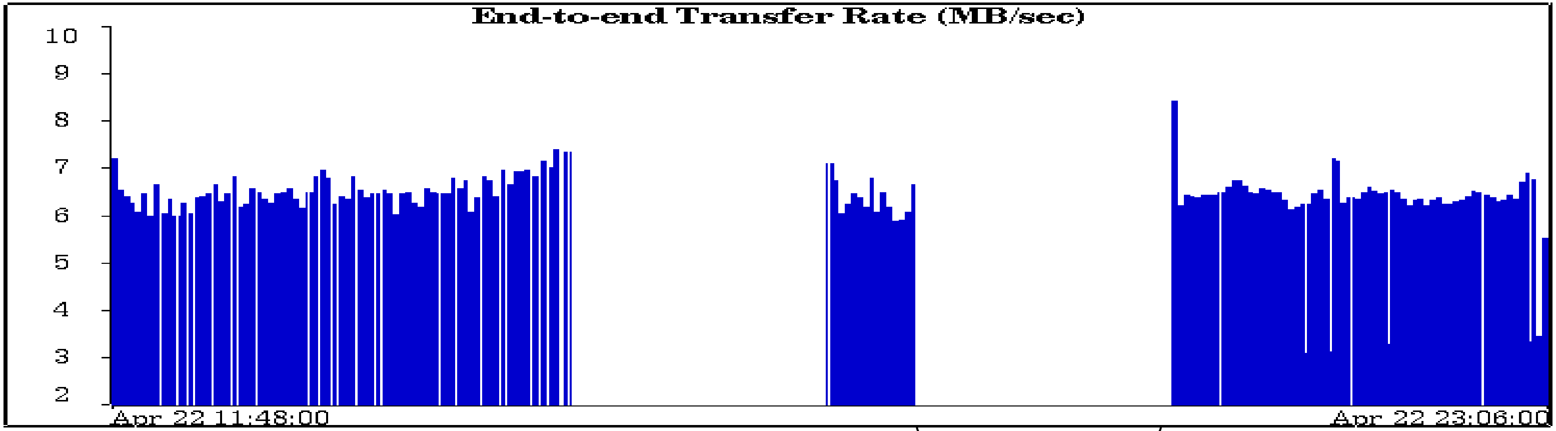


Failure Recovery



UniTree not responding

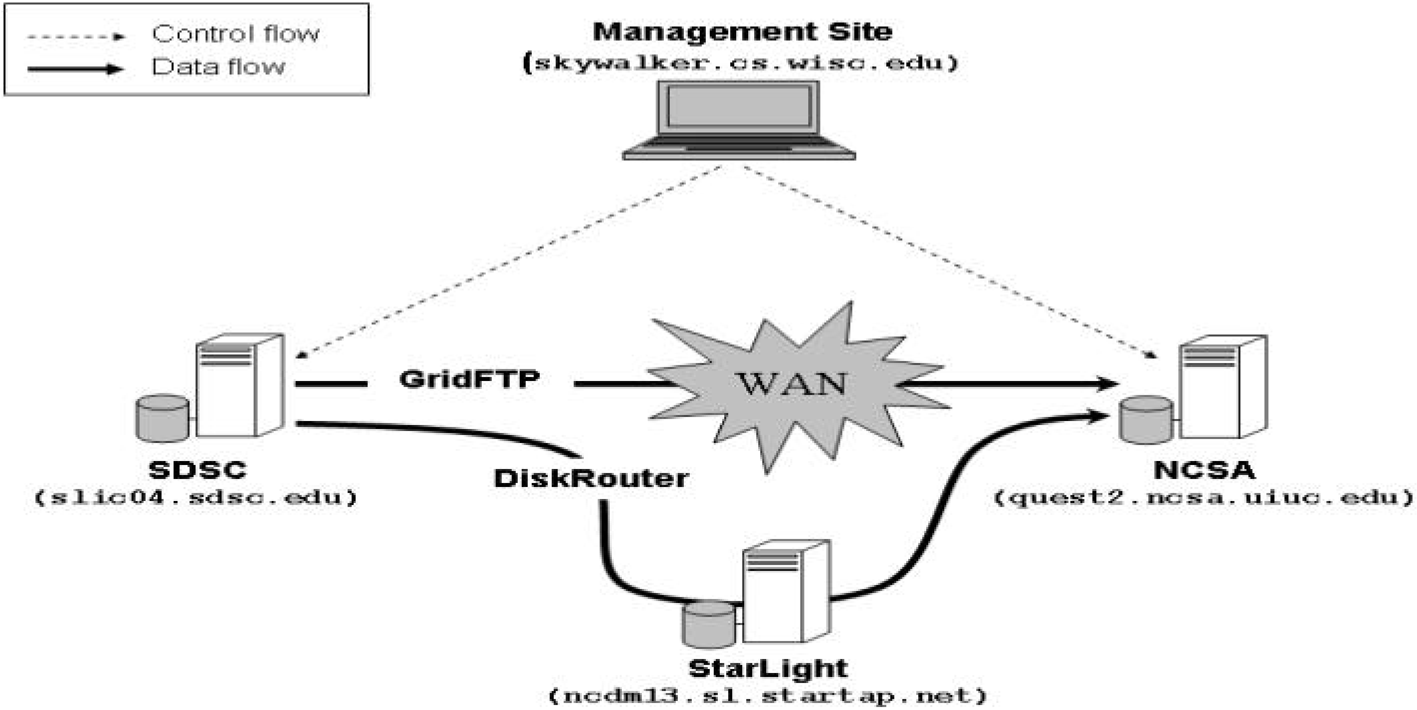
Diskrouter reconfigured
and restarted



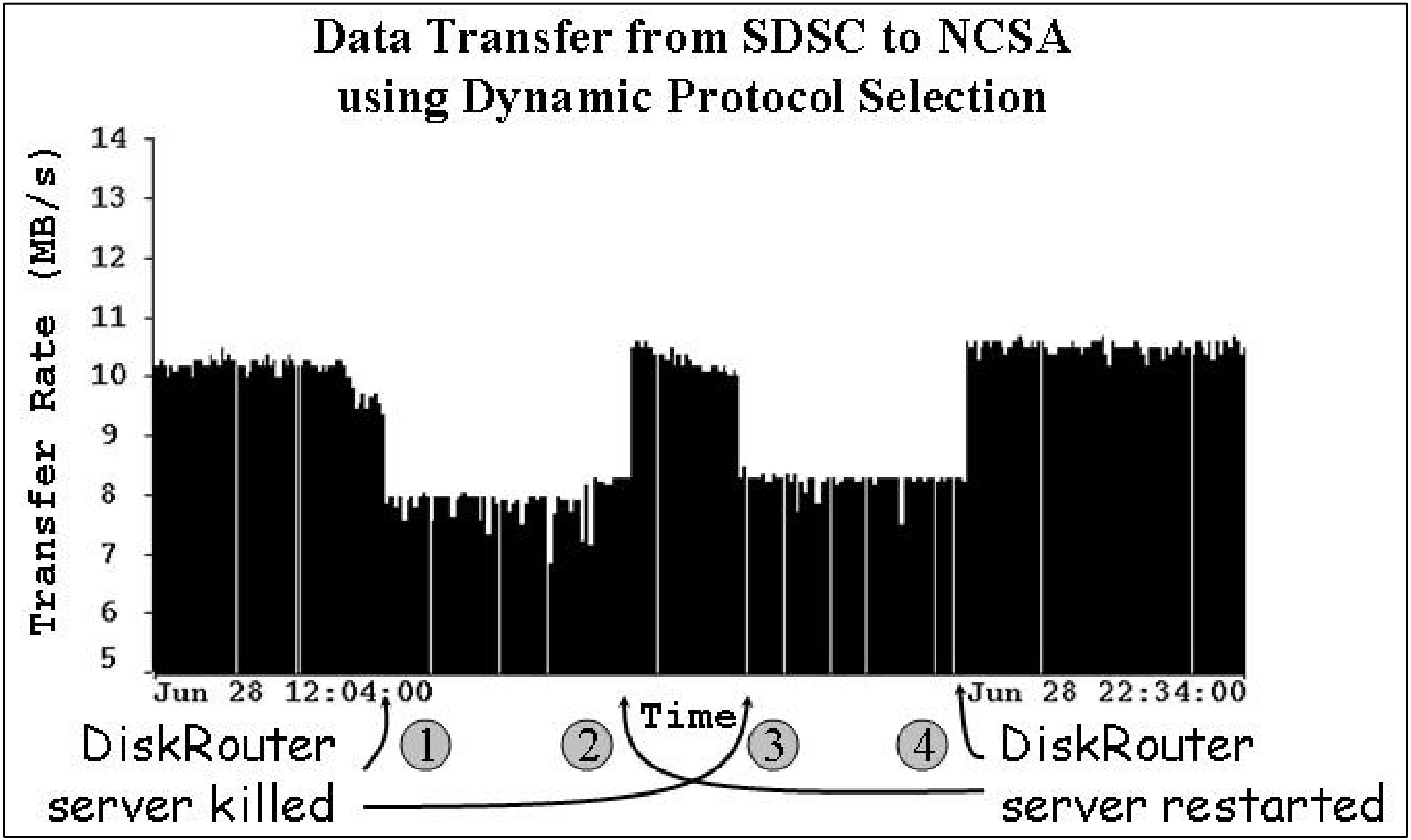
SDSC cache reboot &
UW CS Network outage

Software problem

Case Study -II



Dynamic Protocol Selection



Data Transfer from SDSC to NCSA
using Dynamic Protocol Selection

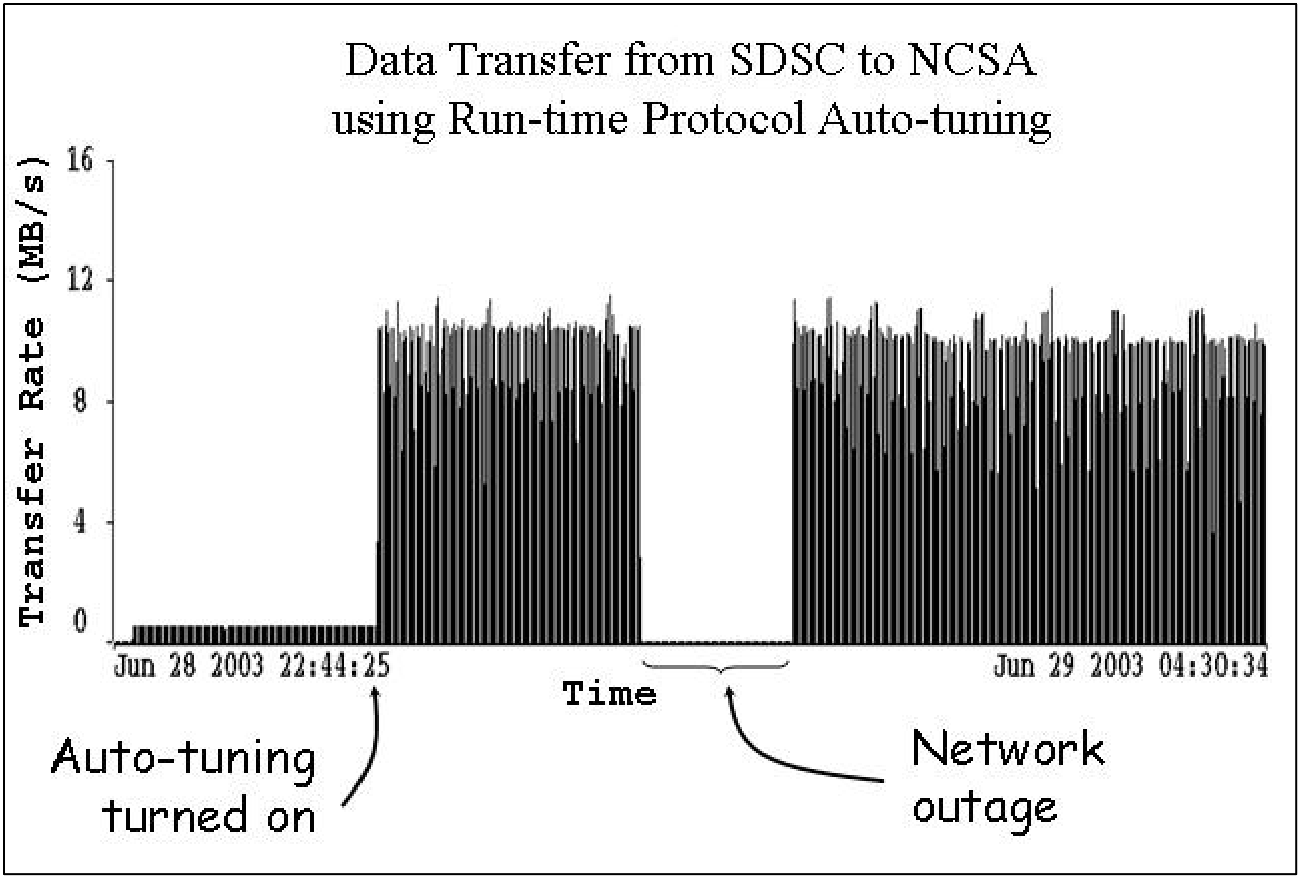
Runtime Adaptation

Before Tuning:

- parallelism = 1
- block_size = 1 MB
- tcp_bs = 64 KB

After Tuning:

- parallelism = 4
- block_size = 1 MB
- tcp_bs = 256 KB

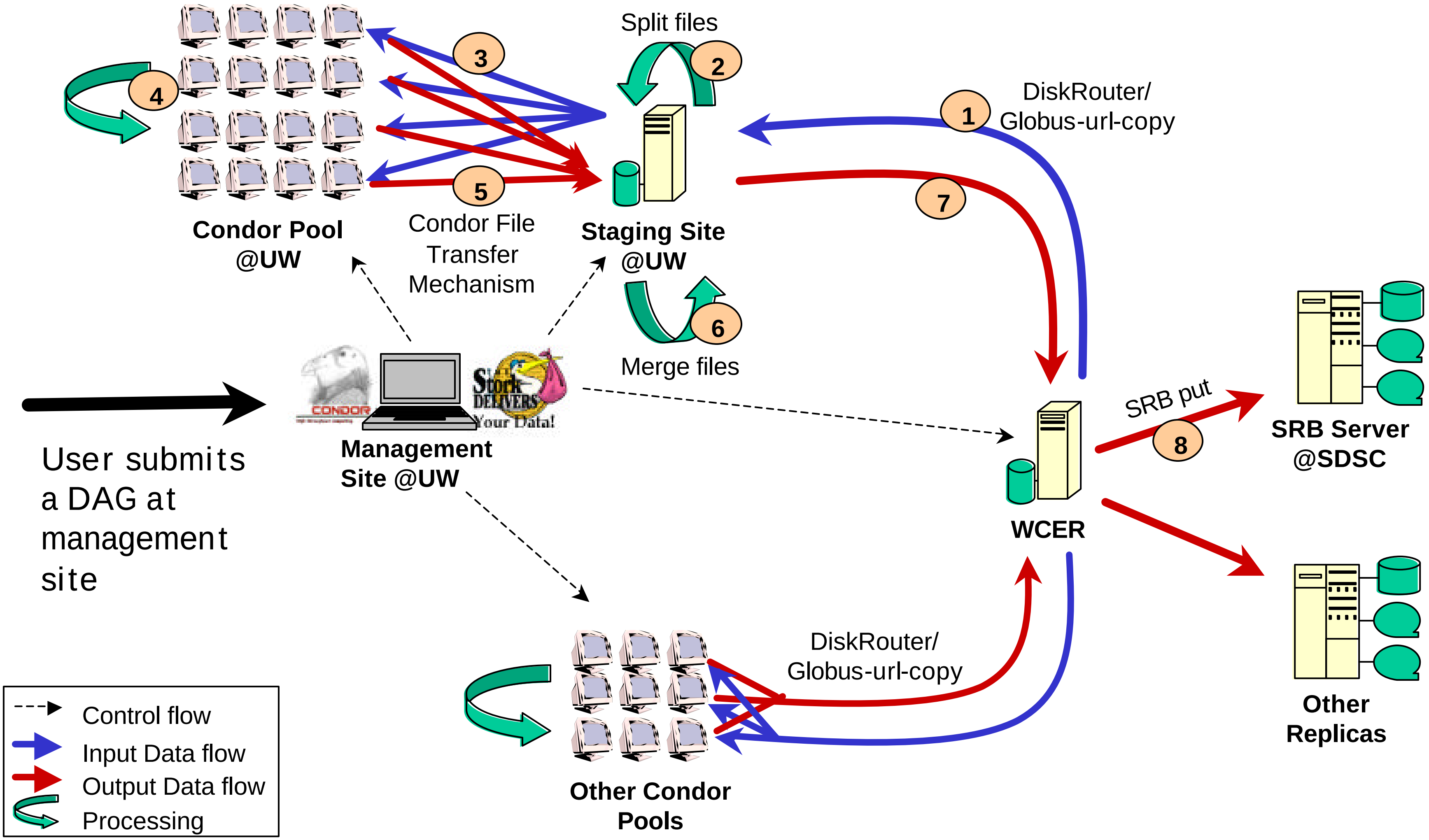


Data Transfer from SDSC to NCSA using Run-time Protocol Auto-tuning

Auto-tuning turned on

Network outage

Case Study - III



User submits a DAG at management site

Management Site @UW

Condor Pool @UW

Condor File Transfer Mechanism

Staging Site @UW

Split files

Merge files

DiskRouter/ Globus-url-copy

WCER

SRB put

SRB Server @SDSC

Other Replicas

Other Condor Pools

DiskRouter/ Globus-url-copy

1 2 3 4 5 6 7 8

Control flow

Input Data flow

Output Data flow

Processing

Conclusions

- Regard data placement as individual jobs.
- Treat computational and data placement jobs differently.
- Introduce a specialized scheduler for data placement.
- Provide end-to-end automation, fault tolerance, run-time adaptation, multilevel policy support, reliable and efficient transfers.

Future work

- Enhanced interaction between Stork and higher level planners
  - better coordination of CPU and I/O
- Interaction between multiple Stork servers and job delegation
- Enhanced authentication mechanisms
- More run-time adaptation

Related Publications

- Tevfik Kosar and Miron Livny. “Stork: Making Data Placement a First Class Citizen in the Grid”. In *Proceedings of 24th IEEE Int. Conference on Distributed Computing Systems (ICDCS 2004),* Tokyo, Japan, March 2004.
- George Kola, Tevfik Kosar and Miron Livny. “A Fully Automated Fault-tolerant System for Distributed Video Processing and Off-site Replication. To appear in *Proceedings of 14th ACM Int. Workshop on etwork and Operating Systems Support for Digital Audio and Video (Nossdav 2004),* Kinsale, Ireland, June 2004.
- Tevfik Kosar, George Kola and Miron Livny. “A Framework for Self-optimizing, Fault-tolerant, High Performance Bulk Data Transfers in a Heterogeneous Grid Environment”. *In Proceedings of 2nd Int. Symposium on Parallel and Distributed Computing (ISPDC 2003),* Ljubljana, Slovenia, October 2003.
- George Kola, Tevfik Kosar and Miron Livny. “Run-time Adaptation of Grid Data Placement Jobs”. In *Proceedings of Int. Workshop on Adaptive Grid Middleware (AGridM 2003)*, New Orleans, LA, September 2003.

You don't have to FedEx your data anymore.. Stork delivers it for you!

- For more information:
  - Email: kosart@cs.wisc.edu
  - http://www.cs.wisc.edu/condor/stork