Run-time Adaptation of Grid Data Placement Jobs

George Kola, Tevfik Kosar and Miron Livny

Condor Project, University of Wisconsin

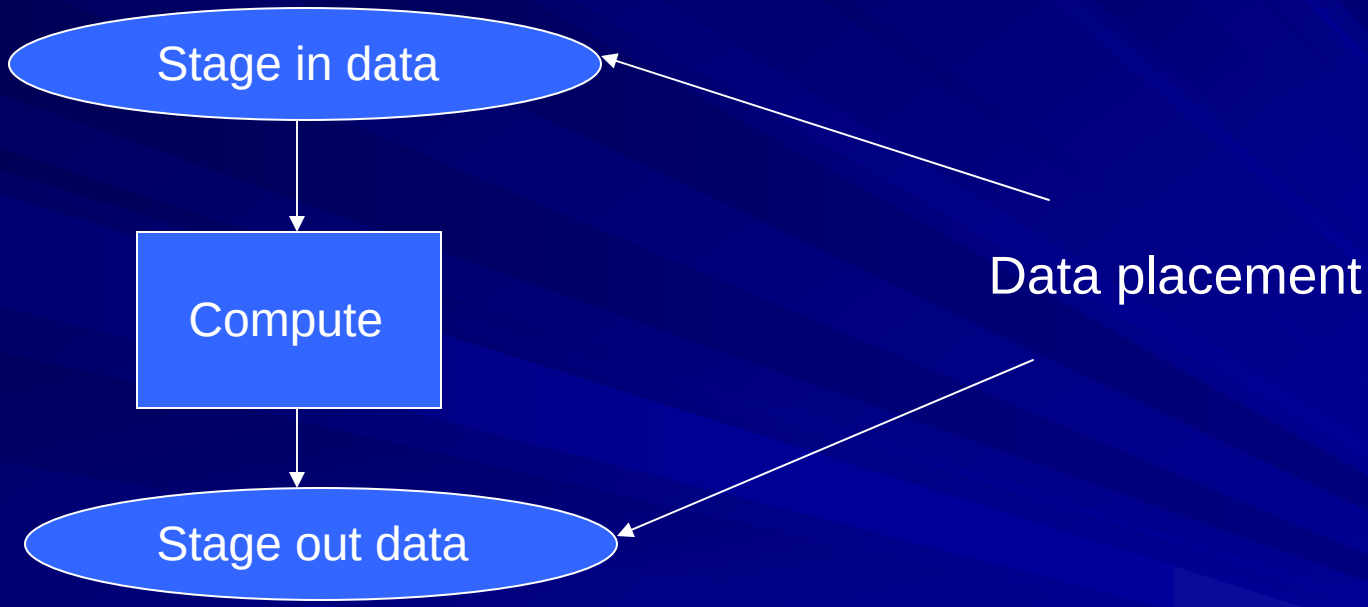
UNIVERSITY OF WISCONSIN MADISON

Introduction

- Grid presents a continuously changing environment
- Data intensive applications are being run on the grid
- Data intensive applications have two parts
  - Data placement part
  - Computation part

Data Placement

A Data Intensive Application



Data placement encompasses data transfer, staging, replication, data positioning, space allocation and de-allocation

Problems

- Insufficient automation
  - Failures
  - No tuning – tuning is difficult !
- Lack of adaptation to changing environment
  - Failure of one protocol while others are functioning
  - Changing network characteristics

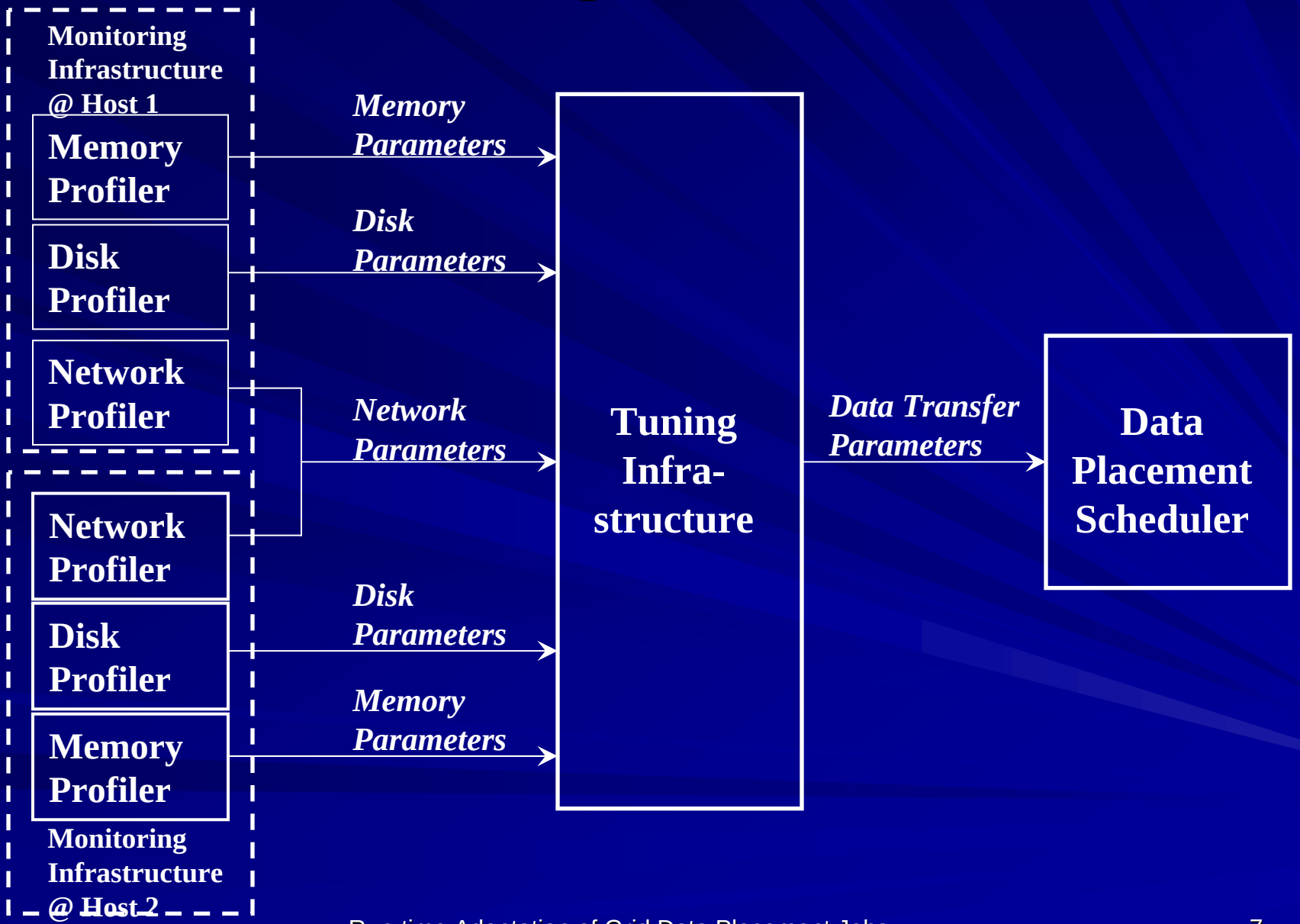
Current Approach

- Fedex
- Hand tuning
- Network Weather Service
  - Not useful for high-bandwidth, high-latency networks
- TCP Auto-tuning
  - 16-bit windows size and window scale option limitations

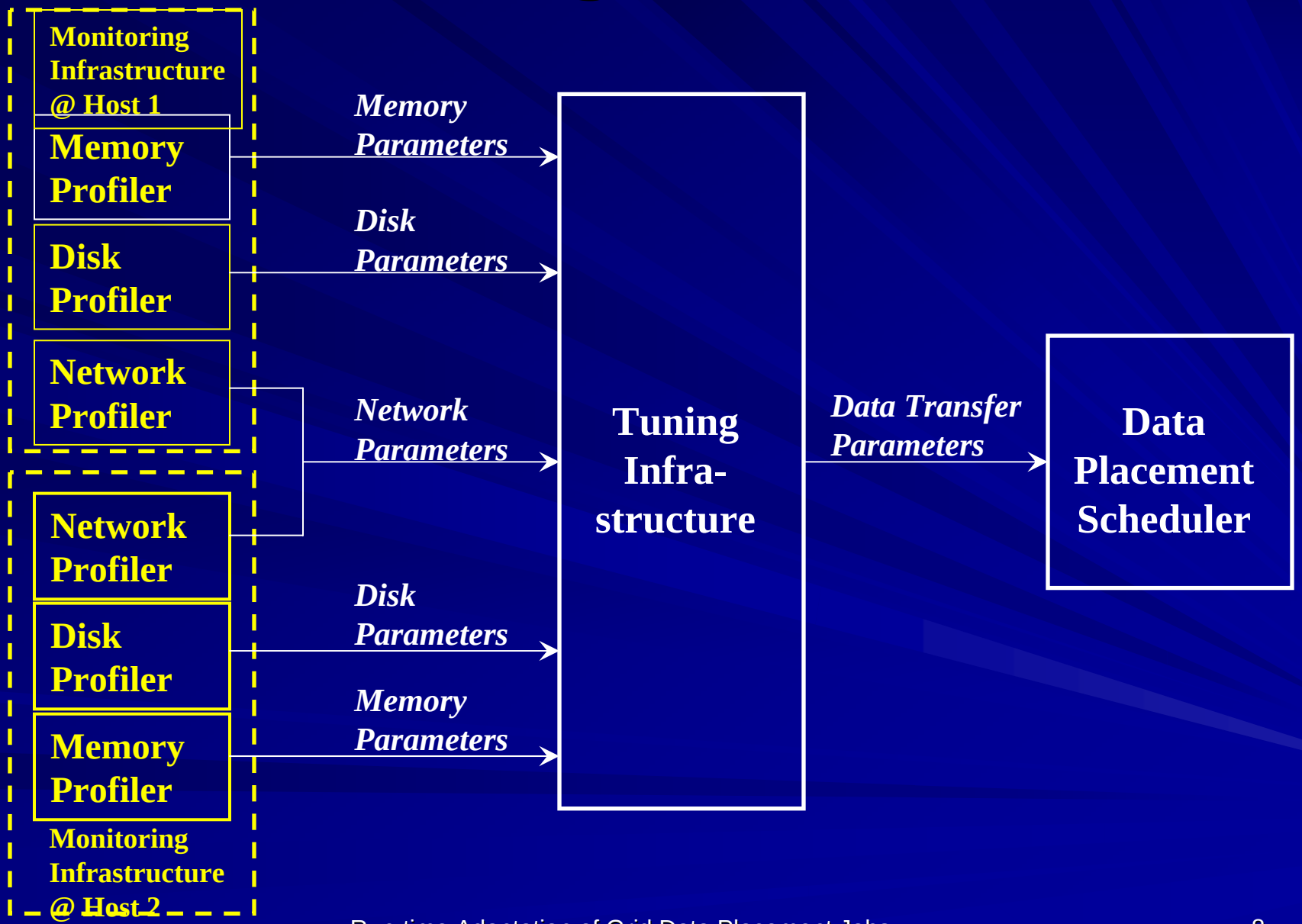
Our Approach

- Full automation
- Continuously monitor environment characteristics
- Perform tuning whenever characteristics change
- Ability to dynamically and automatically choose an appropriate protocol
- Ability to switch to alternate protocol incase of failure

The Big Picture



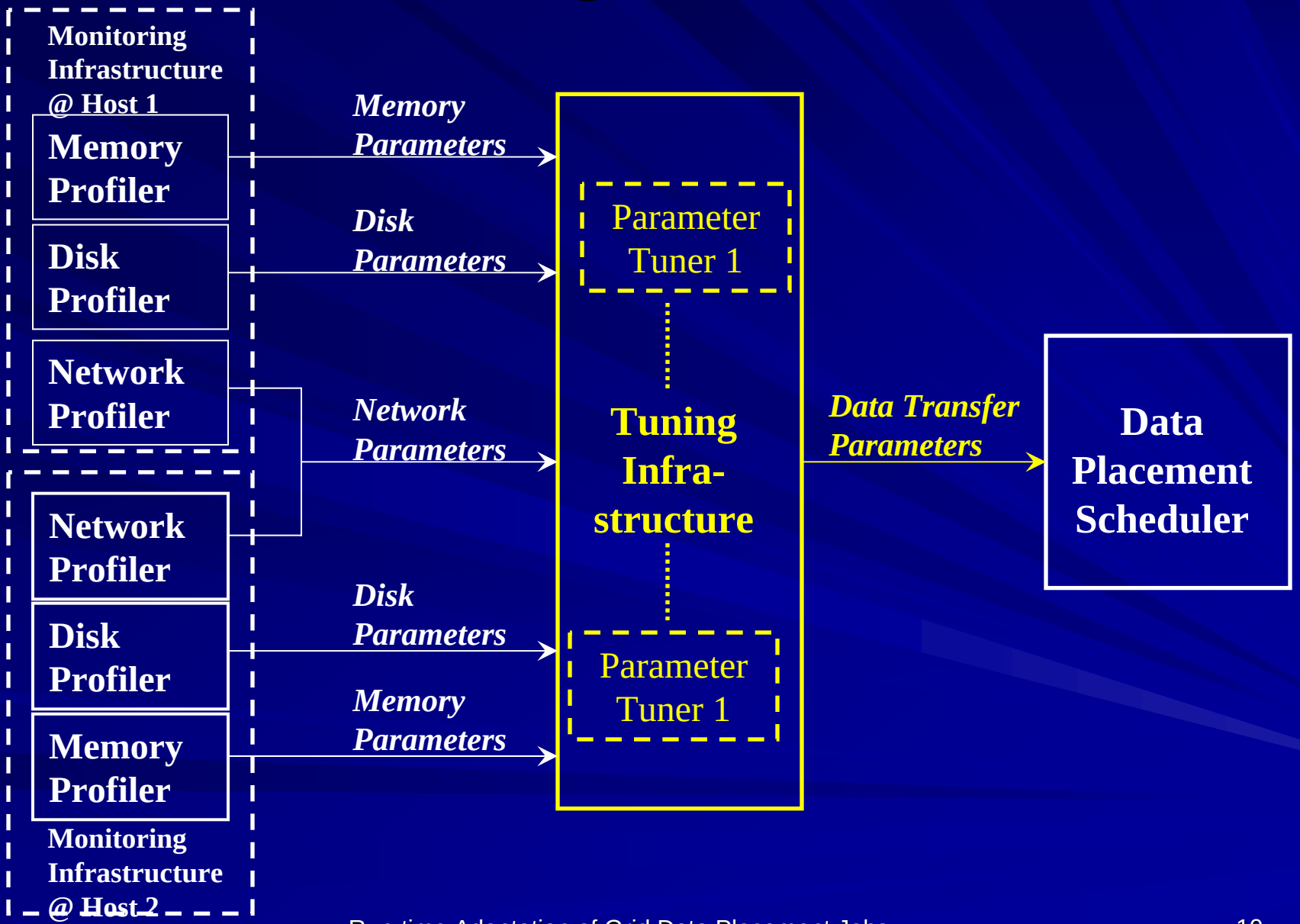
The Big Picture



Profilers

- Memory Profiler
  - Optimal memory block-size and incremental block-size
- Disk Profiler
  - Optimal disk block-size and incremental block-size
- Network Profiler
  - Determines bandwidth, latency and the number of hops between a given pair of hosts
  - Uses pathrate, traceroute and diskrouter bandwidth test tool

The Big Picture



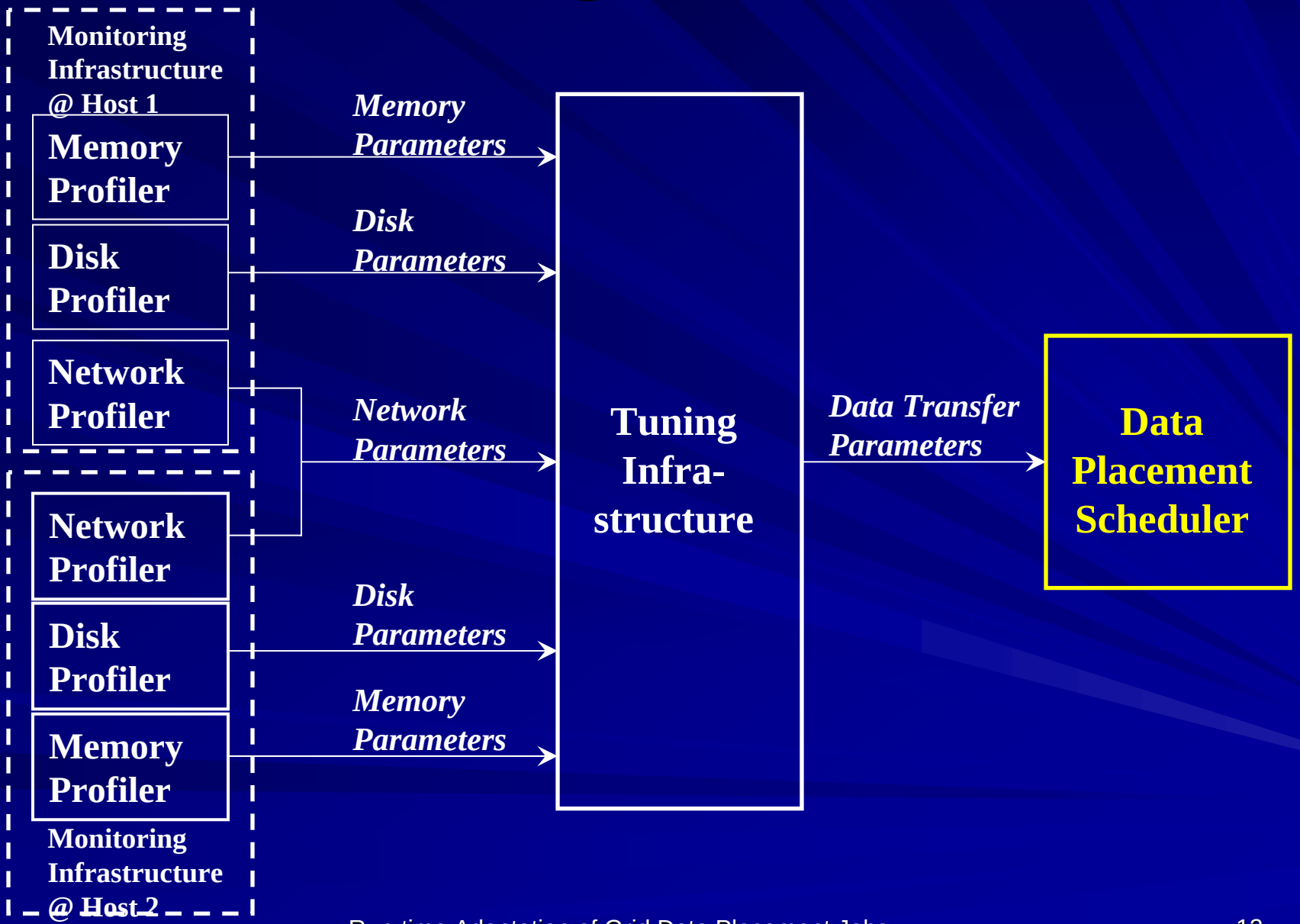
Parameter Tuner

- Generates optimal parameters for data transfer between a given pair of hosts
- Calculates TCP buffer size as the bandwidth-delay product
- Calculates the optimal disk buffer size based on TCP buffer size
- Uses a heuristic to calculate the number of tcp streams

  No of streams = 1+ No of hops with latency > 10ms

  Rounded to an even number

The Big Picture



Data Placement Scheduler

- Data placement is a real-job
- A meta-scheduler (e.g. DAGMan) is used to co-ordinate data placement and computation
- Sample data placement job

```
[
dap_type = “transfer” ;
src_url = “diskrouter://slic04.sdsc.edu/s/s1” ;
dest_url=“diskrouter://quest2.ncsa.uiuc.edu/d/d1”;
]
```

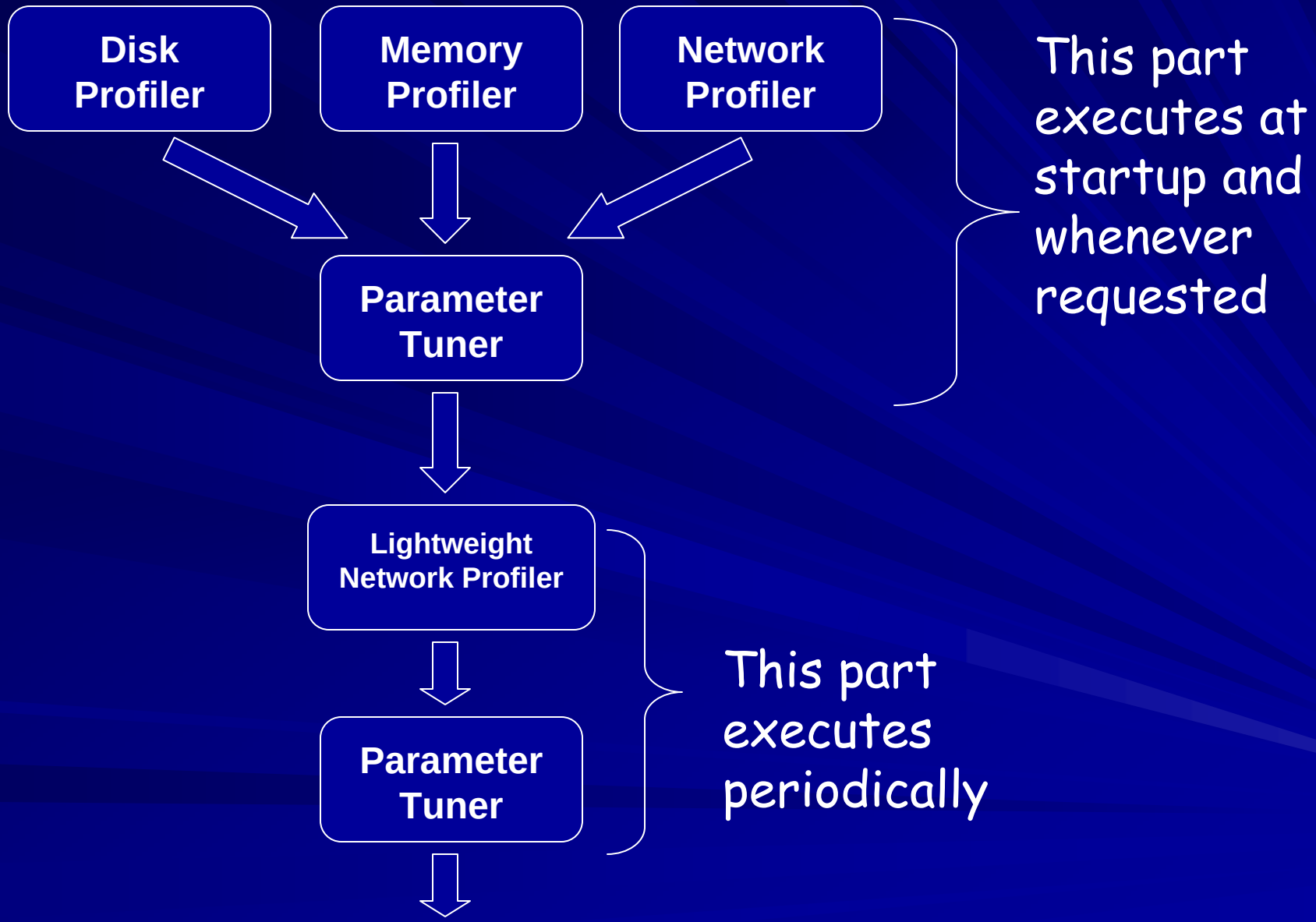
Data Placement Scheduler

- Used Stork, a prototype data placement scheduler
- Tuned parameters are fed to Stork
- Stork uses the tuned parameters to adapt data placement jobs

Implementation

- Profilers are run as remote batch jobs on respective hosts
- Parameter tuner is also a batch job
- An instance of parameter tuner is run for every pair of nodes involved in data transfer
- Monitoring and tuning infrastructure is coordinated by DAGMan

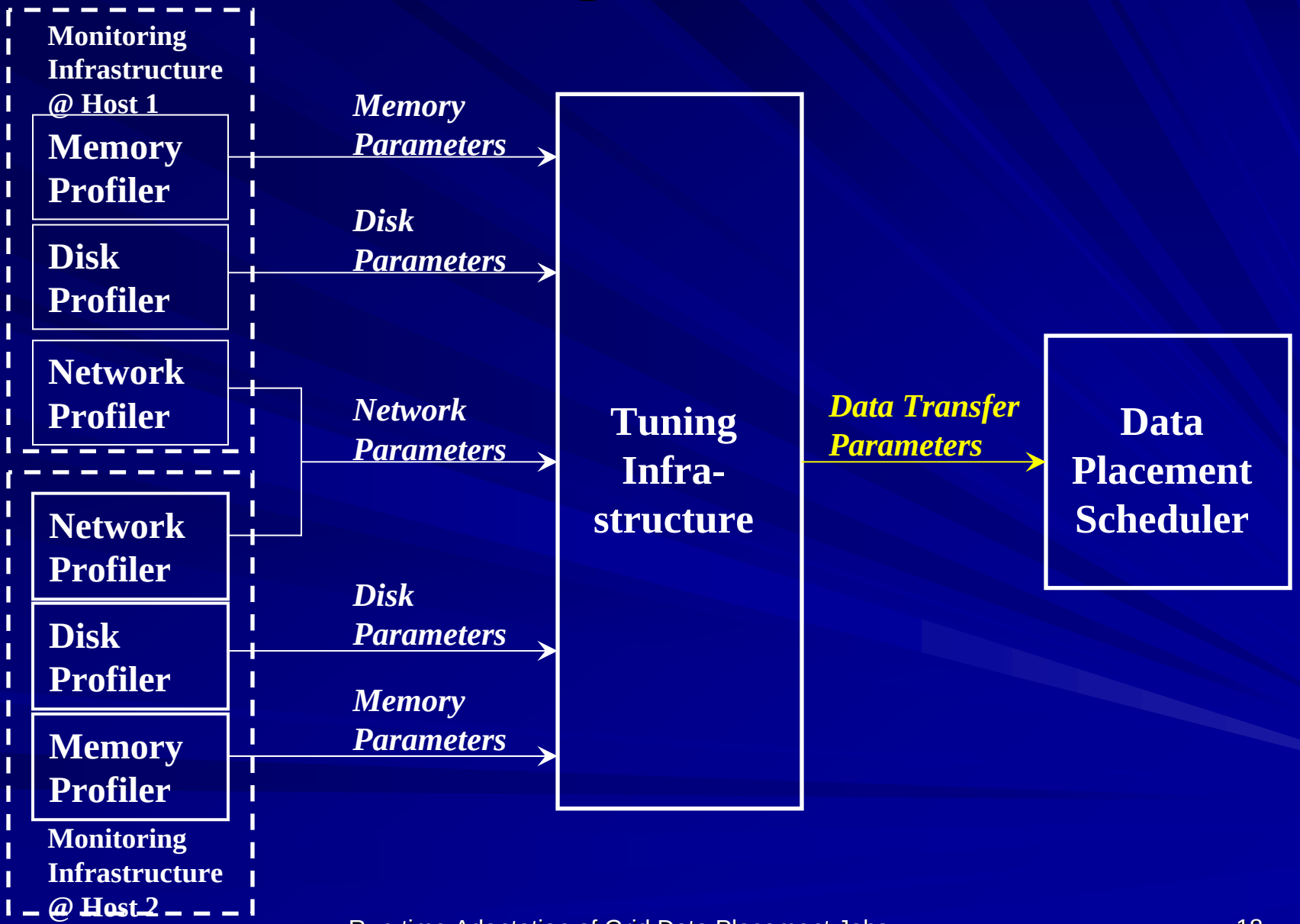
Coordinating DAG



Scalability

- There is no centralized server
- Parameter tuner can be run on any computation resource
- Profiler data is 100s of bytes per host
- There can be multiple data placement schedulers

The Big Picture



Scalability

- There is no centralized server
- Parameter tuner can be run on any computation resource
- Profiler data is 100s of bytes per host
- There can be multiple data placement schedulers

Dynamic Protocol Selection

- Determines the protocols available on the different hosts
- Creates a list of hosts and protocols in ClassAd format

e.g.

```
[ hostname="quest2.ncsa.uiuc.edu" ;
  protocols="diskrouter,gridftp,ftp" ]
[ hostname="nostos.cs.wisc.edu" ;
  protocols="gridftp,ftp,http" ]
```

Dynamic Protocol Selection

```
[
dap_type = “transfer” ;
src_url = “any://slic04.sdsc.edu/s/data1” ;
dest_url=“any://quest2.ncsa.uiuc.edu/d/data1” ;
]
```

- Stork determines an appropriate protocol to use for the transfer
- In case of failure, Stork chooses another protocol

Alternate Protocol Fallback

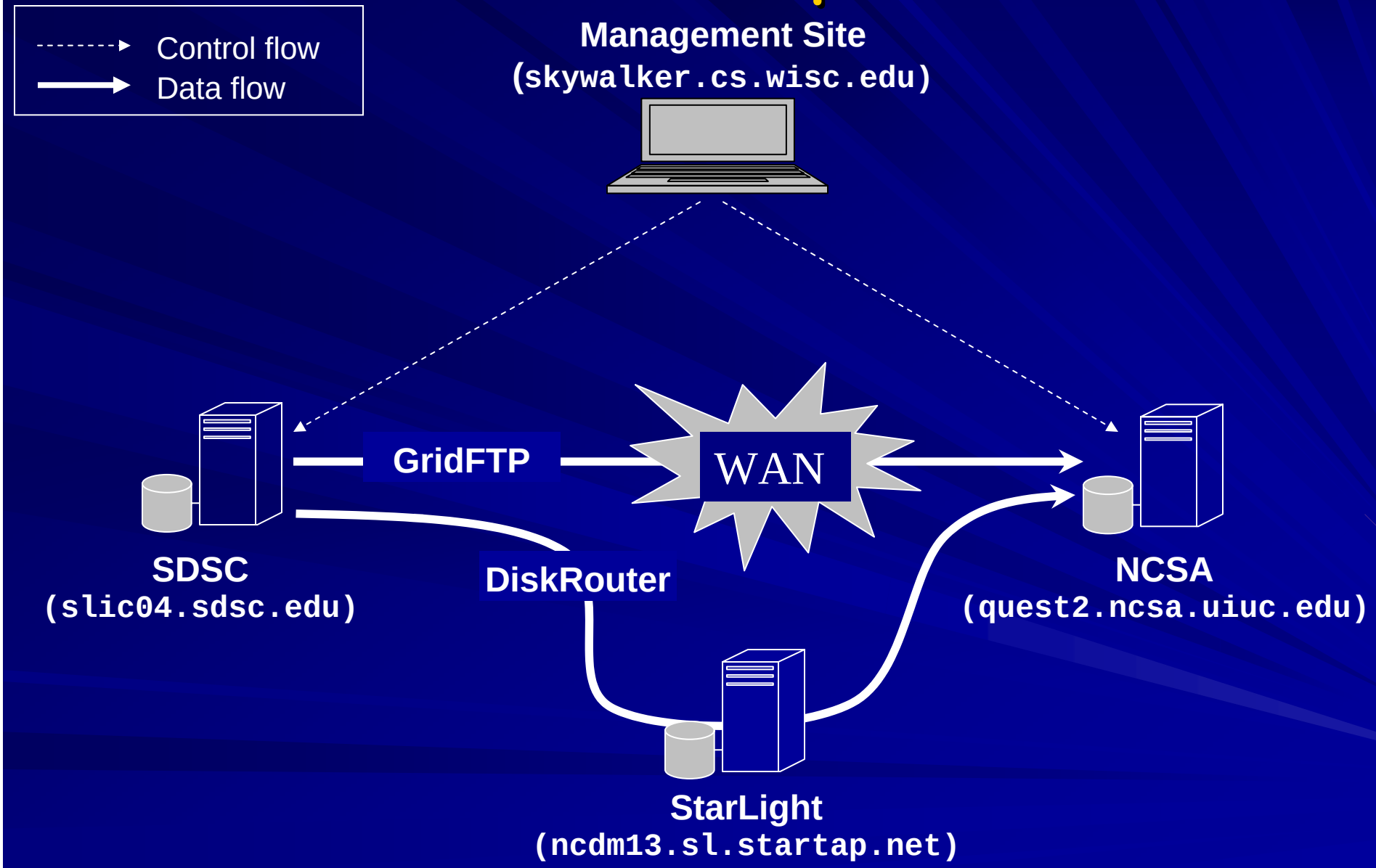
```
[
dap_type = “transfer”;
src_url = “diskrouter://slic04.sdsc.edu/s/data1”;
dest_url=“diskrouter://quest2.ncsa.uiuc.edu/d/data1”;
alt_protocols=“nest-nest, gsiftp-gsiftp”;
]
```

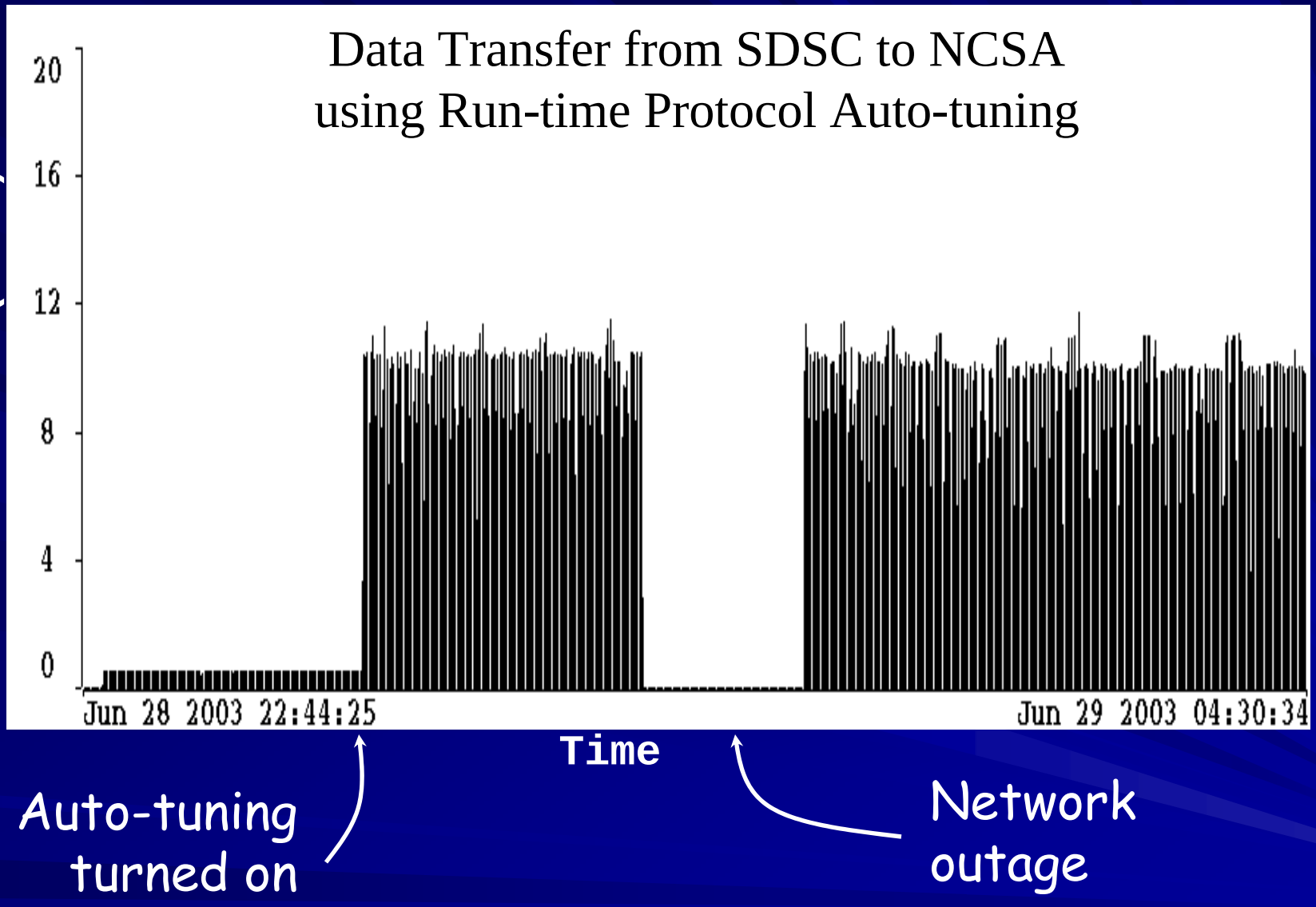
In case of diskrouter failure, Stork will switch to other protocols in the order specified

Real World Experiment

DPOSS data had to be transferred from SDSC located in San Diego to NCSA located at Chicago

Real World Experiment




Data Transfer from SDSC to NCSA
using Run-time Protocol Auto-tuning
Transfer Rate (MB/s)
20
16
12
8
4
0
Jun 28 2003 22:44:25
Jun 29 2003 04:30:34
Time
Auto-tuning
turned on
Network
outage

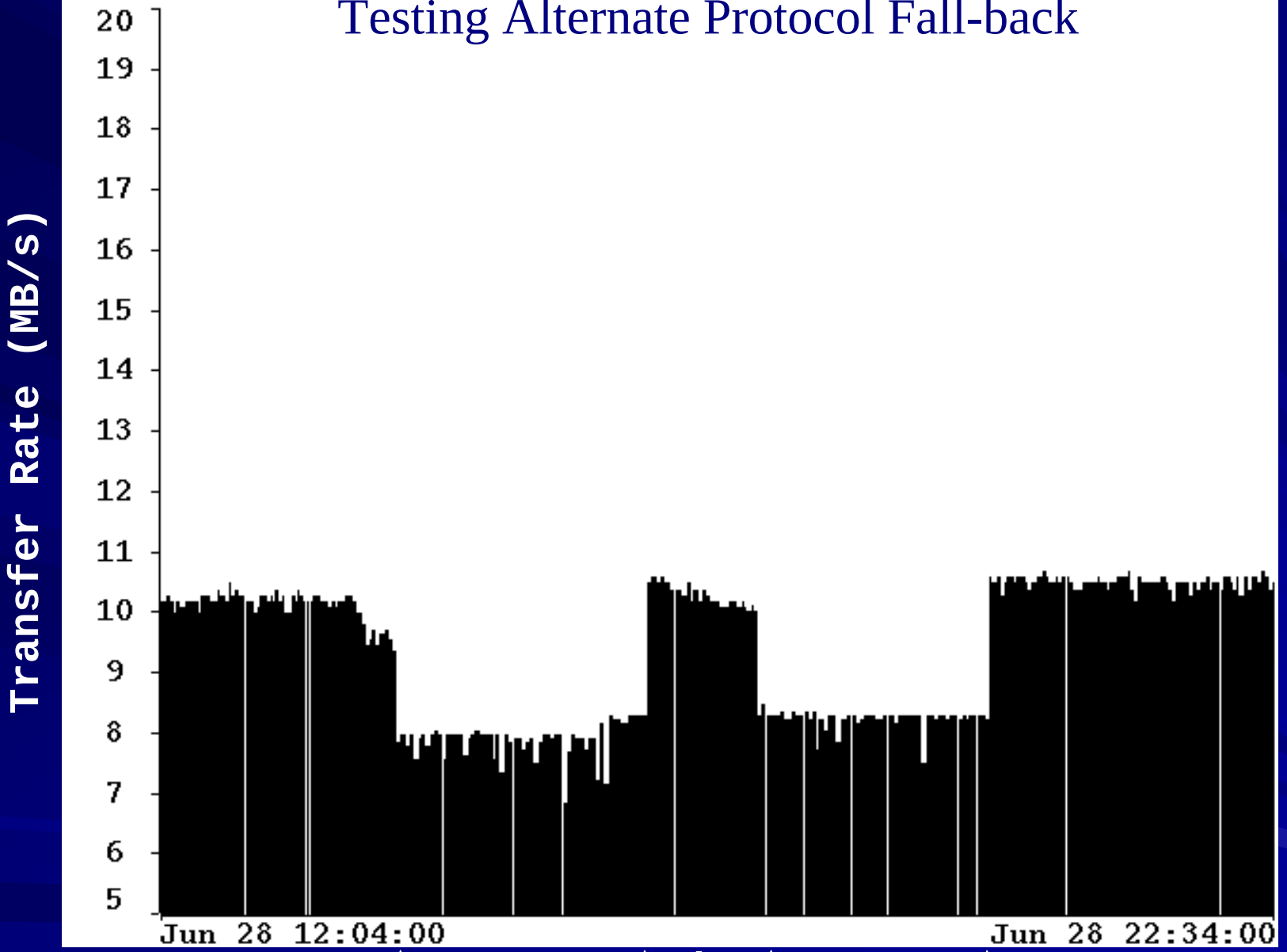
Parameter Tuning

Parameter	Before auto-tuning	After auto-tuning
Parallelism	1 TCP stream	4 TCP streams
Block size	1 MB	1 MB
TCP buffer size	64 KB	256 KB

Testing Alternate Protocol Fall-back

```
[
dap_type = “transfer”;
src_url =
  “diskrouter://slic04.sdsc.edu/s/data1”;
dest_url=“diskrouter://quest2.ncsa.uiuc.e
  du/d/data1”;
alt_protocols=“nest-nest, gsiftp-gsiftp”;
]
```

Testing Alternate Protocol Fall-back



Conclusion

- Run-time adaptation has a significant impact (20 times improvement in our test case)
- The profiling data has the potential to be used for data mining
  - Network misconfigurations
  - Network outages
- Dynamic protocol selection and alternate protocol fall-back increase resilence and improve overall throughput

Questions ?

For more information you can contact:

- George Kola : kola@cs.wisc.edu
- Tevfik Kosar: kosart@cs.wisc.edu

Project web pages:

- Stork: http://cs.wisc.edu/condor/stork
- DiskRouter: http://cs.wisc.edu/condor/diskrouter