Just-in-time Staging of Large Input Data for Supercomputing Jobs

Henry Monti, Ali R. Butt

VirginiaTech
Invent the Future

Sudharshan S. Vazhkudai

OAK RIDGE
National Laboratory

HPC Center Data Stage-in Problem

- Data stage-in entails moving all necessary input files for a job to a center's local storage
  - Requires significant commitment of center resources while waiting for the job to run
  - Storage failures are common, and users may be required to restage data
- Delaying input data causes costly job rescheduling
- Staging data too early is undesirable
  - From a center standpoint:
    - Wastes scratch space that could be used for other jobs
  - From a user job standpoint:
    - Potential job rescheduling due to storage system failure

⇒Coinciding Input Data Stage-in time with job execution time improves HPC center serviceability

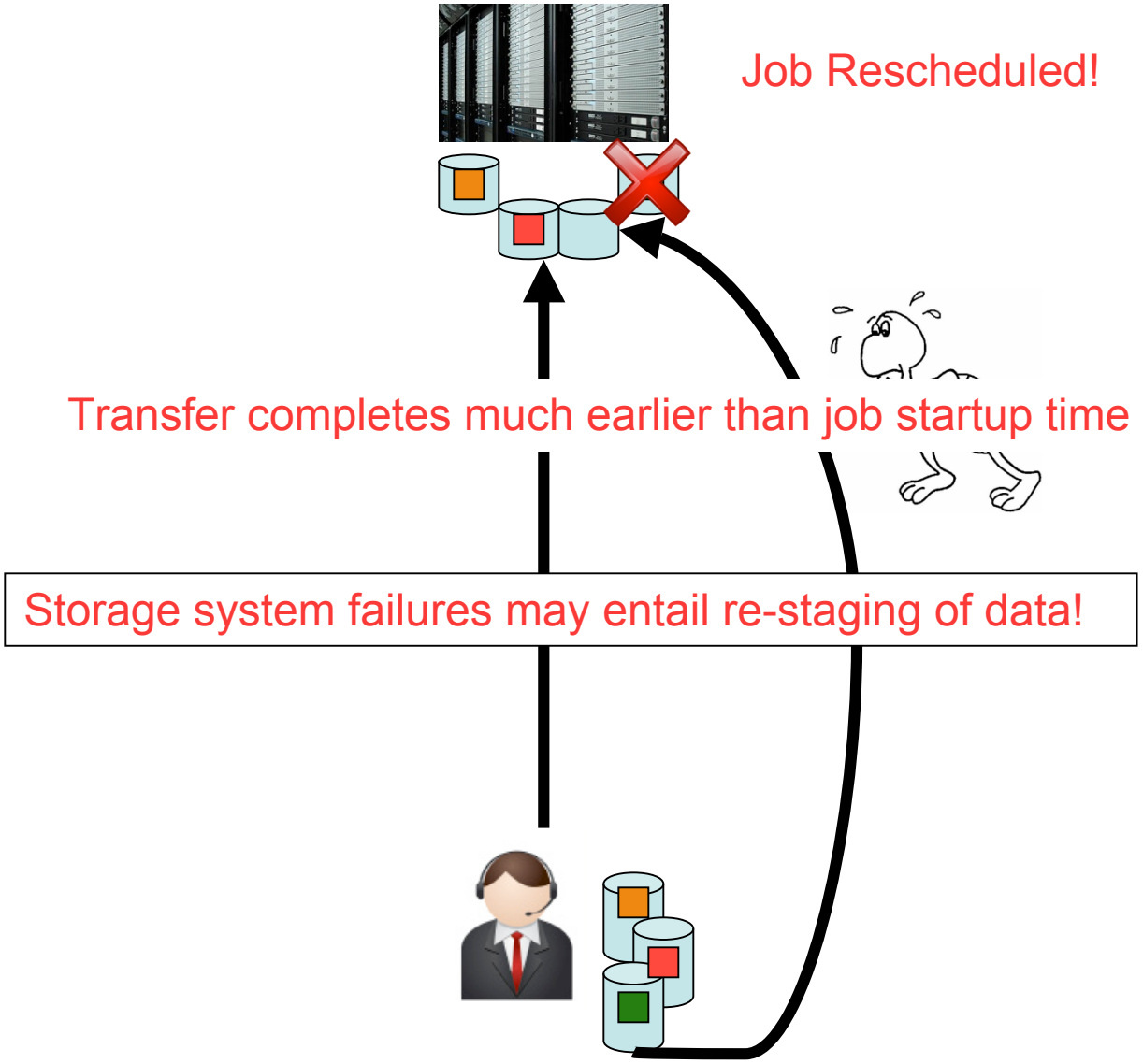
Current Methods to Stage-in Data

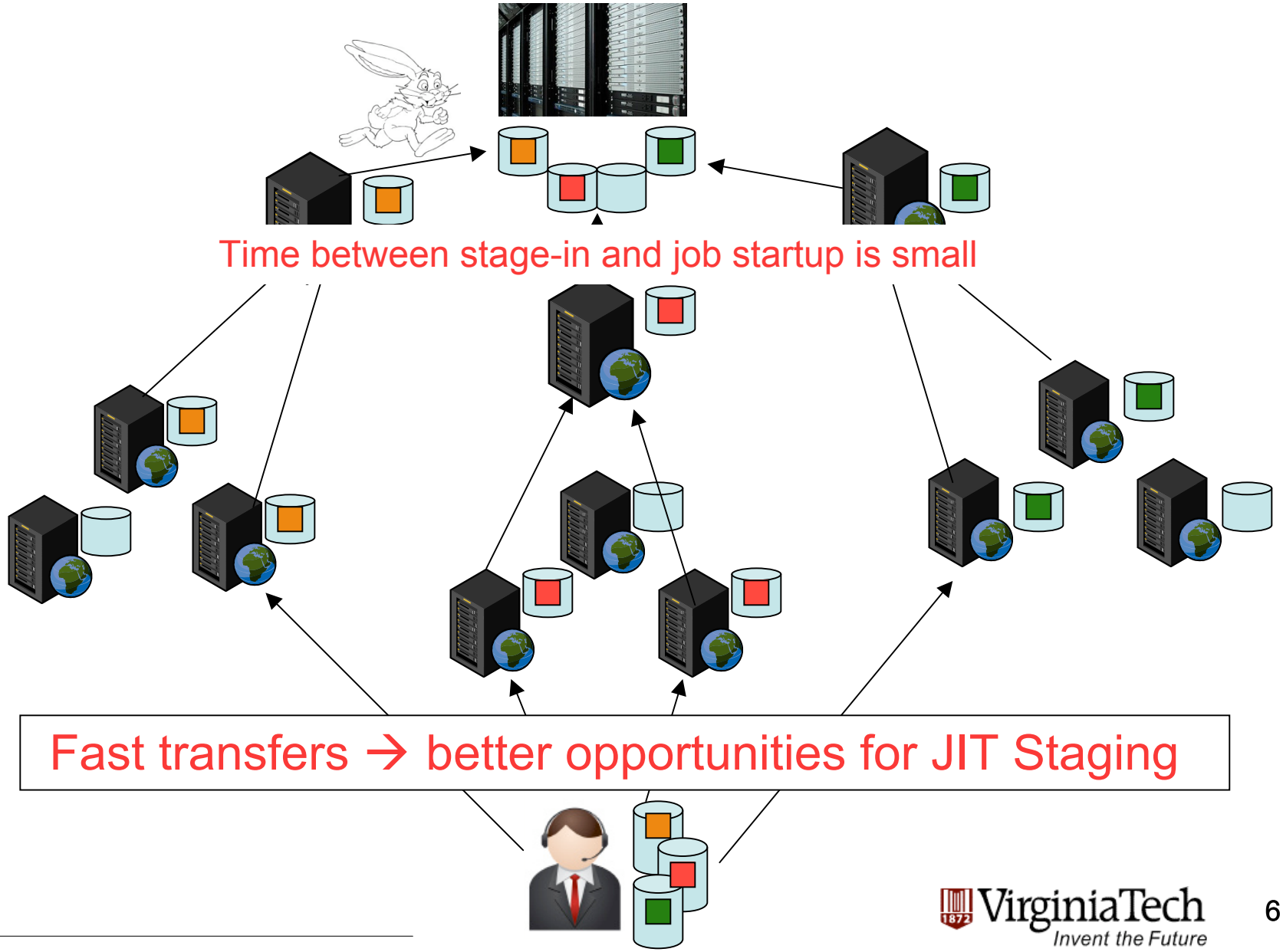
- No standardized method
- Employ common point-to-point transfer tools:
  - GridFTP, HSI, scp, …
- Limitations
  - Entail early stage-in to ensure data availability
  - Do not leverage orthogonal bandwidth
  - Oblivious to deadlines or job start times

Not an optimal approach for HPC data stage-in

Our Contribution: A Just-in-time Data Stage-in Service

- Coincides data stage-in with job start time
- Attempts to reduce overall scratch space usage
- Uses intermediate locations for temporary storage
- Provides for quick restaging after storage failures
- Integrates with real-world tools
  - Portable Batch System (PBS)
  - BitTorrent
- Supports a fault-tolerant way to stage data while efficiently utilizing scratch space


Job Rescheduled!
Transfer completes much earlier than job startup time
Storage system failures may entail re-staging of data!


Time between stage-in and job startup is small
Fast transfers → better opportunities for JIT Staging

Challenges Faced in JIT Staging

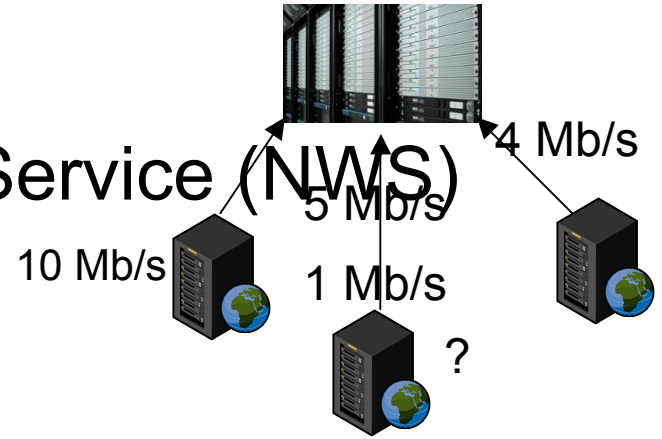
1. Obtaining accurate job start times
2. Adapting to dynamic network behavior
3. Ensuring data reliability and availability
4. Managing deadlines during stage-in
5. Utilizing intermediate nodes
6. Providing incentives to participate

Obtaining Accurate Job Start Times

- Accurate estimates of job start time needed to avoid job rescheduling
- Solution: Use Batch Queue Prediction (BQP)
  - Provides statistical upper bound on job wait
  - Predicts probability of job starting by the deadline
- Obtain predictions of job start time from BQP
- Stage-in data using this deadline

Adapting Data Distribution To Dynamic Network Behavior

- Available bandwidth can change
  - Distribute data randomly – may not be effective
  - Utilize network monitoring
- Solution: Use Network Weather Service (NWS)
  - Provides bandwidth Measurement
  - Predicts future bandwidth



- Choose dynamically changing data paths
- Select enough nodes to satisfy a given deadline
- Monitor and update the selected nodes

Protecting Data from Intermediate Storage Location Failure

- Problem: Node failure may cause data loss
- Solution:
  1. Use data replication
     - Achieved through multiple data flow paths
  2. Employ Erasure coding
     - Can be done by the user or at the intermediates

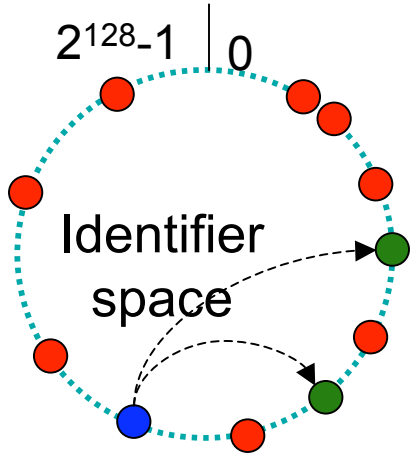
Managing Deadlines during Stage-in

- Use NWS to measure available bandwidths
  - Use Direct if it can meet a deadline
  - Otherwise, perform decentralized stage-in
- If end host fails or cannot meet deadline
  - Utilize decentralized stage-in approach

$$T_{Stage} <= T_{JobStartup}$$

Intermediate Node Discovery

- User specifies known and trusted nodes
- Utilize P2P Overlay
- Nodes advertise their availability to others
- Receiving nodes *discovers* the advertiser



- Discovered nodes utilized as necessary

P2P Data Storage and Dissemination

- P2P-based storage
  - Enables robust storage of data on loosely coupled distributed participants: CFS, PAST, OceanStore, …
- P2P-based multicast
  - Enables application-level one to many communication
- Example: BitTorrent
  - Uses a scatter-gather protocol to distribute files
  - Leverages Seeds - peers that store entire files
  - Employs a tracker to maintain lists of peers
  - Uses a “torrent file” containing metadata for data retrieval

Incentives to Participate in Stage-in Process

- Modern HPC jobs are often collaborative
  - "Virtual Organizations" - set of geographically distributed users from different sites
  - Jobs in TeraGrid usually from such organizations
- Resource bartering among participants to facilitate each others stage-in over time
- Nodes specified and trusted by the user

Integrating Stage-in with PBS

- Provide new PBS directives
  - Specifies destination, intermediate nodes, and deadline

```
#PBS -N myjob
#PBS -l nodes=128, walltime=12:00
mpirun -np 128 ~/MyComputation
#Stagein file://SubmissionSite:/home/user/input1
#InterNode node1.Site1:49665:50GB
...
#InterNode nodeN.SiteN:49665:30GB
#Deadline 1/14/2007:12:00
```

Adapting BitTorrent Functionality to Data Stage-in

- Tailor BitTorrent to meet the needs of our stage-in
- Restrict the amount of result-data sent to a peer
  - Peers with less storage than the input size can be utilized
- Incorporate global information into peer selection
  - Use NWS bandwidth measurements
  - Use knowledge of node capacity from PBS scripts
  - Choose the appropriate nodes with storage capacity
- Recipients are not necessarily end-hosts
  - They may simply pass data onward

Evaluation: Experimental Setup

- Objectives
  - Compare with direct transfer, and BitTorrent
  - Validate our method as an alternative to other stage-in methods
- PlanetLab test bed
  - 6 PlanetLab nodes:

    center + end user + 4 intermediate nodes
- Experiments:
  Compare the proposed method with
  - Point-to-point transfer (scp)
  - Standard BitTorrent
- Observe the effect of bandwidth changes

Results: Data Transfer Times with Respect to Direct Transfer

File Size	100 MB	240 MB	500 MB	2.1 GB
Direct	172	351	794	3082
Client Offload	139	258	559	2164
Pull	43	106	193	822

A JIT stage-in is capable of significantly improving transfer times

Times are in seconds

Results: Data Transfer Times with Respect to Standard BitTorrent

Phase	BitTorrent	Our Method
Send to all intermediate nodes (Client Offload)	2653	2164
HPC Center download (Pull)	960	822

Monitoring based stage-in is capable of outperforming standard BitTorrent

Times are in seconds
Transferring 2.1 GB file

Conclusion

- A fresh look at Data Stage-in
  - Decentralized approach
  - Monitoring-based adaptation
- Considers deadlines and job start times
- Integrated with real-world tools
- Outperforms direct transfer by **73.3%** in our experiments

Future Work

- Measuring scratch space savings
- Measuring potential job delays
- Testing other stage-in scenarios
- Contact
  - Virginia Tech.
    - Distributed Systems and Storage Lab. http://research.cs.vt.edu/dssl/
    - {hmonti, butta}@cs.vt.edu
  - ORNL
    - http://www.csm.ornl.gov/~vazhkuda/Storage.html
    - vazhkudaiss@ornl.gov