The “Coolness” of Reliability and other tales …

Ali R. Butt

VirginiaTech
Invent the Future

Disk Storage Requirements

- **Persistence**
  - Data is not lost between power-cycles
- **Integrity**
  - Data is not corrupted, “what I stored is what I retrieve”
- **Availability**
  - Data can be accessed at any time

- Performance: Sustain high data transfer rates
- Efficiency: Reduce resource (energy, space) wastage

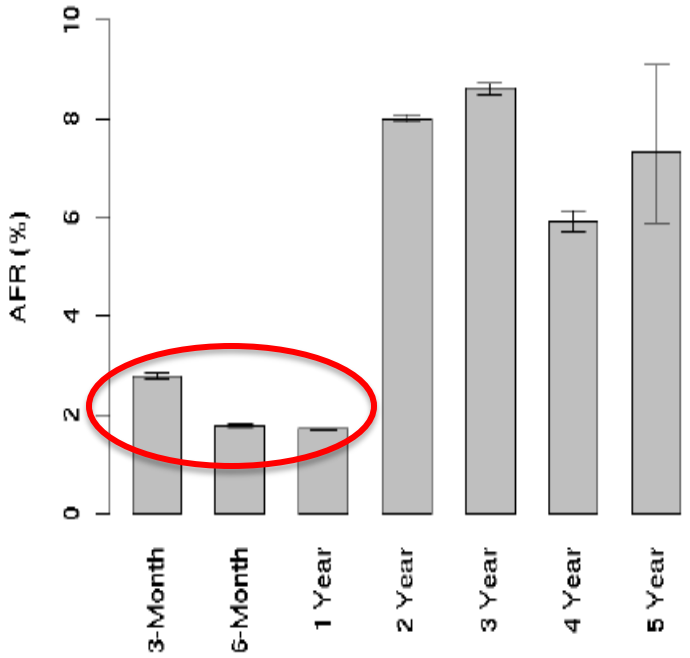
Modern Storage Systems Characteristics

- Employ 10s to 100s of disks
  (1000s not that far off)
- Package disks into storage units (appliances)
  - Direct connected
  - Network connected
- Support simultaneous access for performance
- Use redundancy to protect against disk failures

Large Number of Disks →Failures are Common

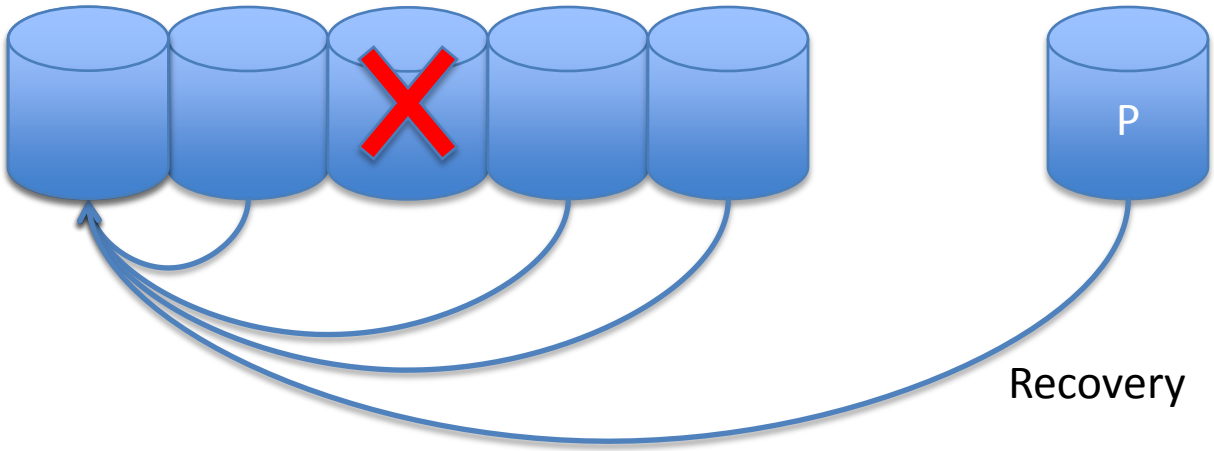
+ Aging does not have a significant effect
− Disks can fail in batches

Failure mitigation is critical

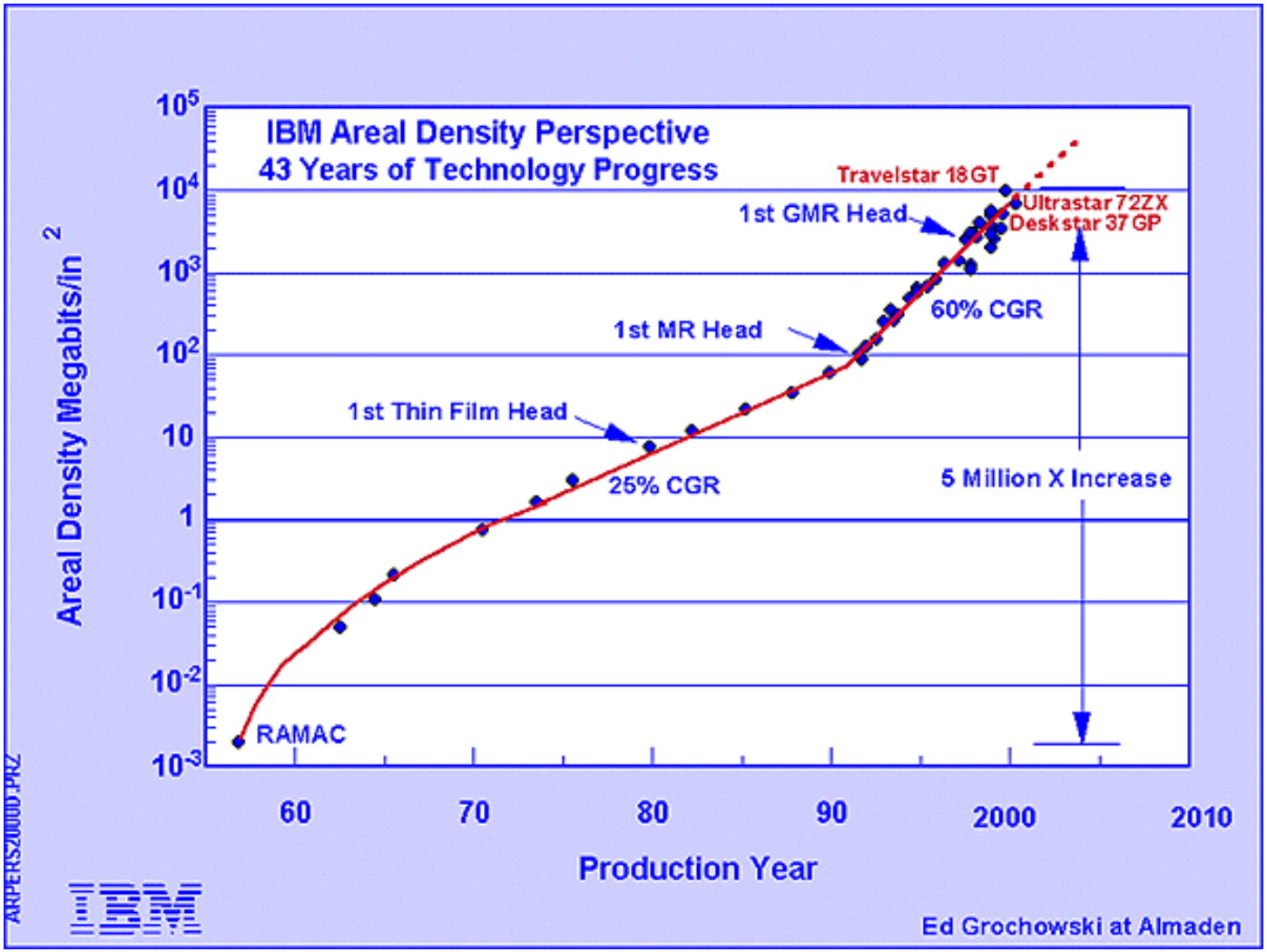


Annualized Failure Rates
(Failure Trends in a Large Disk Drive Population, Pinheiro et. al. FAST'07)

Tolerating Disk Failures using RAID



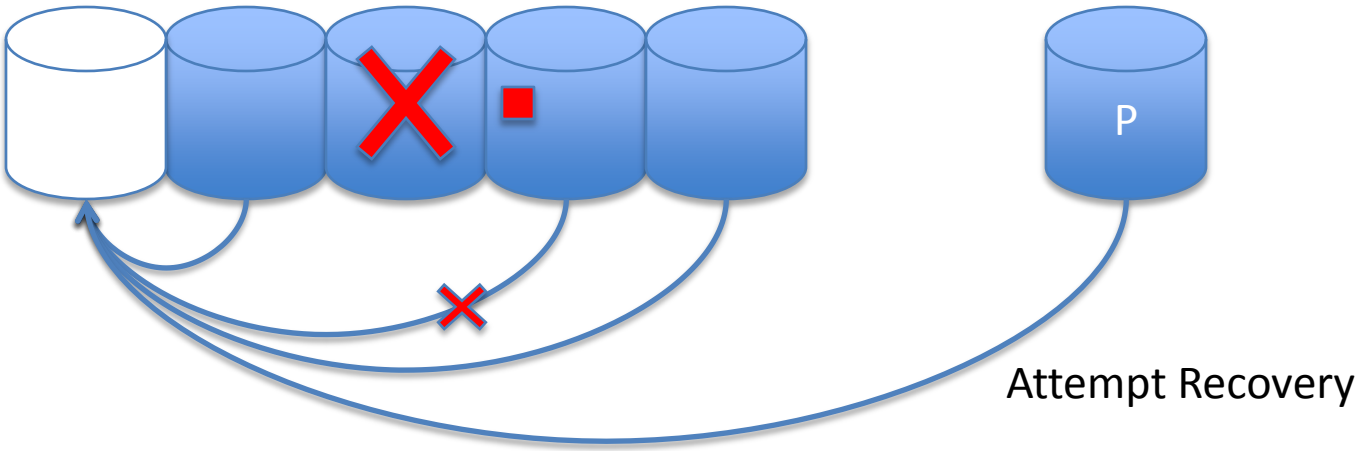
Growing Disk Density



How Latent Sector Errors Occur?

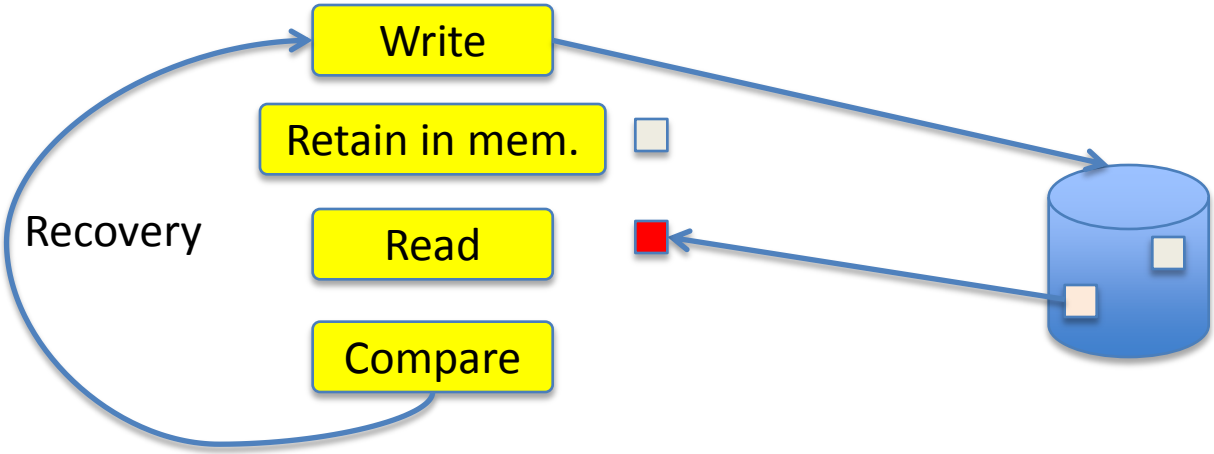
- OS writes data to disk, perceives it to be successful
- Data is corrupted due to bit flips, media failures, etc.
- Errors remain undiscovered (hidden)
- Later OS is unable to read data → ERROR

Effect of Latent Sector Errors



Data Loss

Protecting Against Latent Errors: Idle Read After Write (IRAW*)

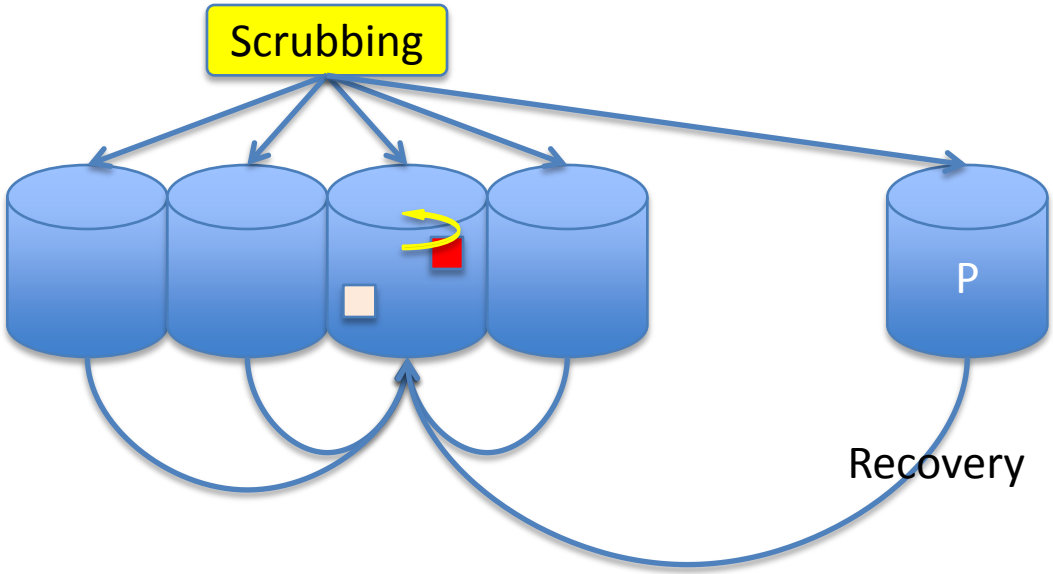


- IRAW can improve data reliability

  → **Check reads are done when disk is idle**

*Idle Read After Write, Riska and Riedel, ATC'08

Protecting Against Latent Errors: Disk Scrubbing*



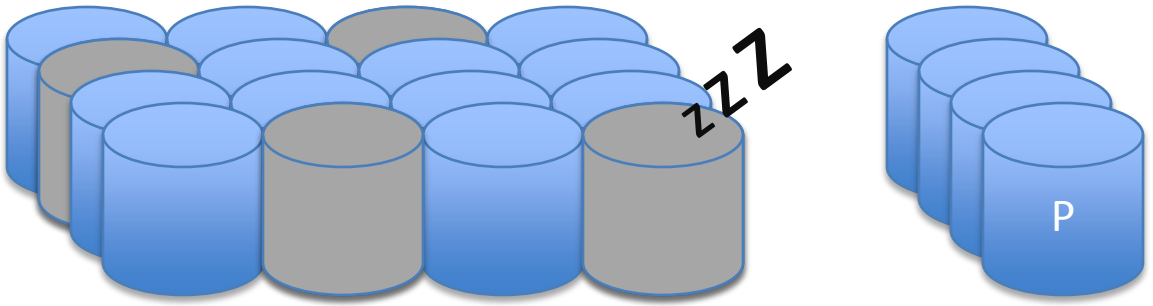
- Scrubbing improves data reliability

  → **Scrub during idle periods**

* Disk scrubbing in large archival storage systems, Schwarz et. al., MASCOTS'04

A Large Number of Disks can Consume Significant Energy

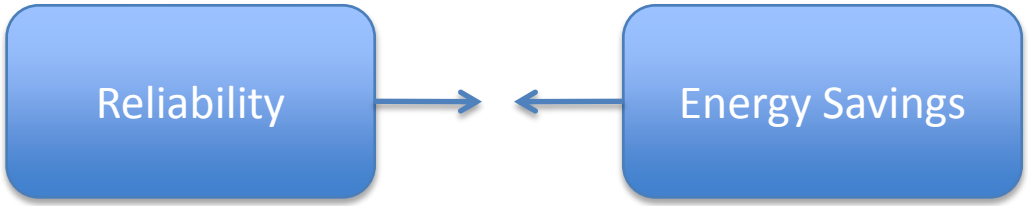
$$



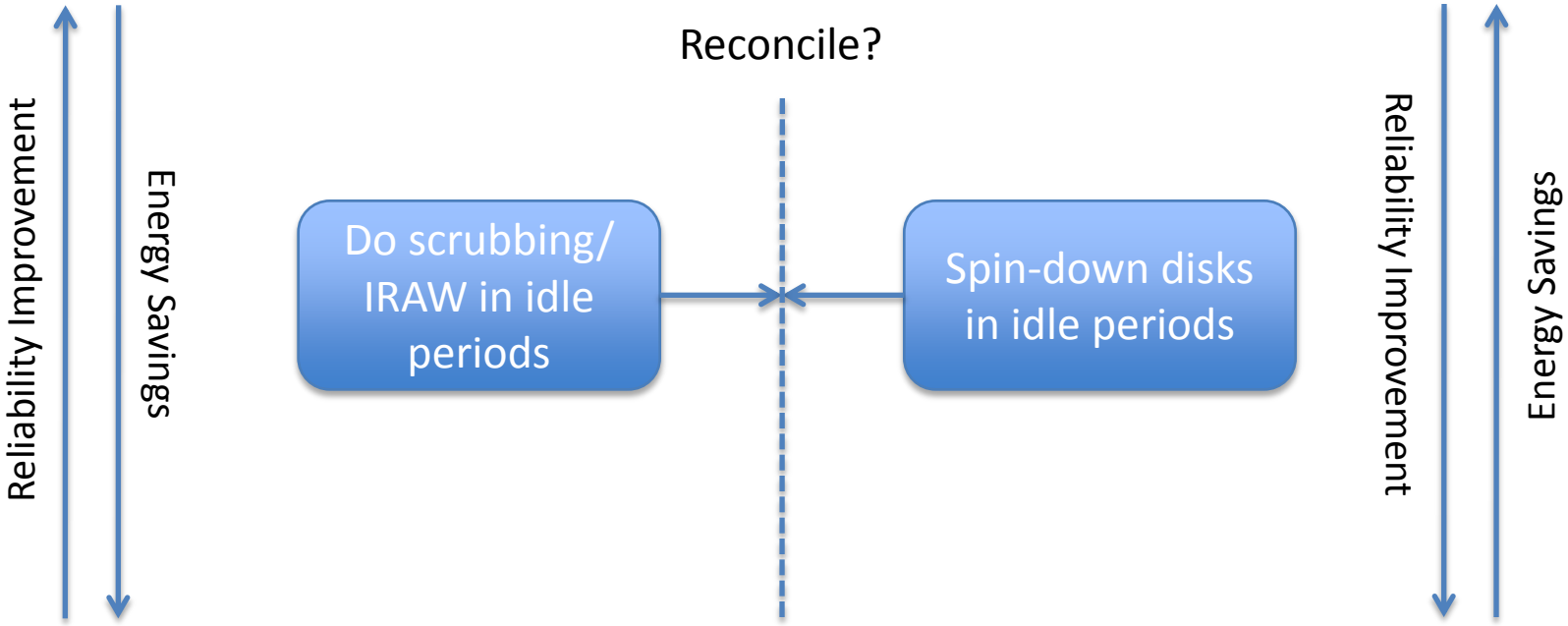
- Spinning-down disks saves energy

  → **Spin-down disks during idle periods**

Reliability or Energy Savings? Or Both?



Reliability Vs. Energy Savings: Which Way To Go? *



- Similar trade-offs present themselves in energy-performance optimization domain
  - Energy-delay product (EDP):
    A flexible metric that finds a balance between saving energy vs. improving performance

* On the Impact of Disk Scrubbing on Energy Savings, Wang, Butt, Gniady, HotPower'08

Energy-Reliability Product (ERP)

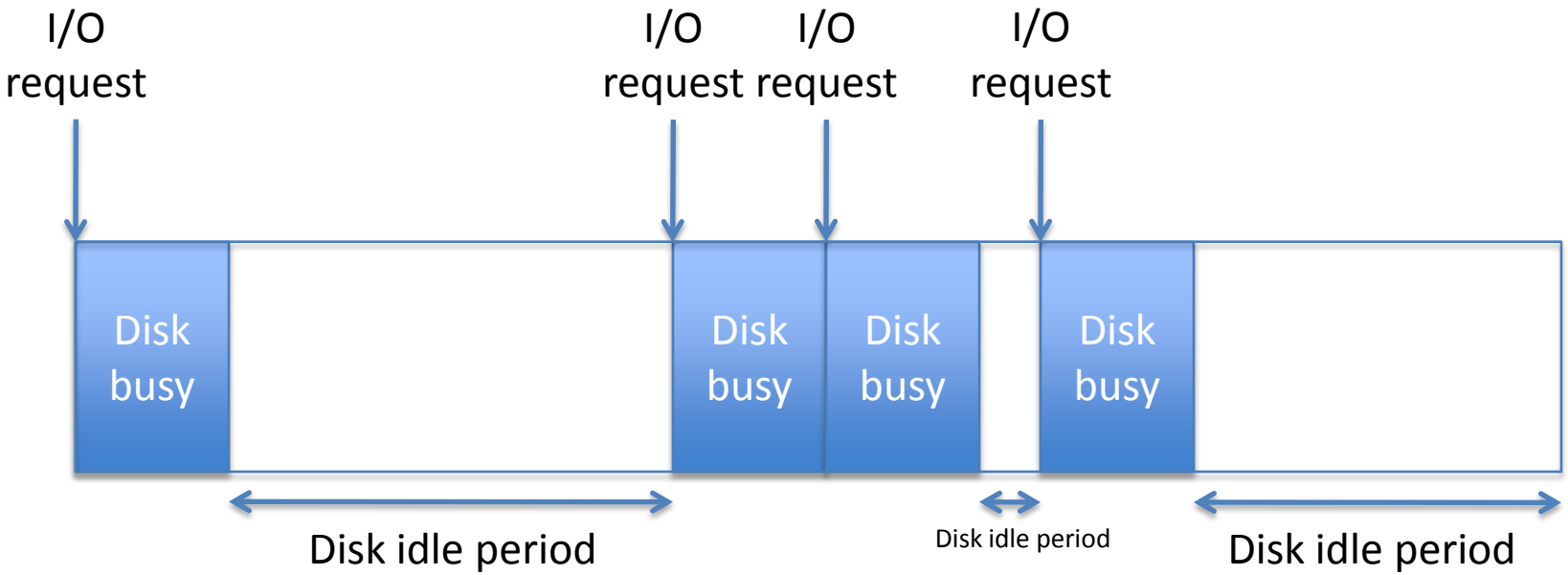
- A new metric that considers both energy and reliability

  ERP = Energy Savings * Reliability Improvement

- Can ERP help us reconcile energy & reliability?
  - Want good energy savings
  - Want to improve reliability

- **Goal: Maximize ERP**

Background: Anatomy of a Disk Idle Period



Measuring Reliability

- A common metric: Mean Time to Data Loss (MTTDL)
  - Higher value of MTTDL → Better reliability
- For scrubbing, MTTDL can be expressed in terms of Scrubbing Period
  - Definition: Time between two scrubbing cycles
  - Shorter scrubbing period, higher MTTDL
    - Detailed models of MTTDL for scrubbing have been developed [Iliadis2008, Dholakia2008]

Determining ERP

- ERP = Energy Savings $*$ Reliability Improvement
- ERP can be expressed in terms of MTTDL:
  - ERP = Energy Savings $*$ Increase in MTTDL

- For scrubbing, MTTDL is inversely proportional to scrubbing period

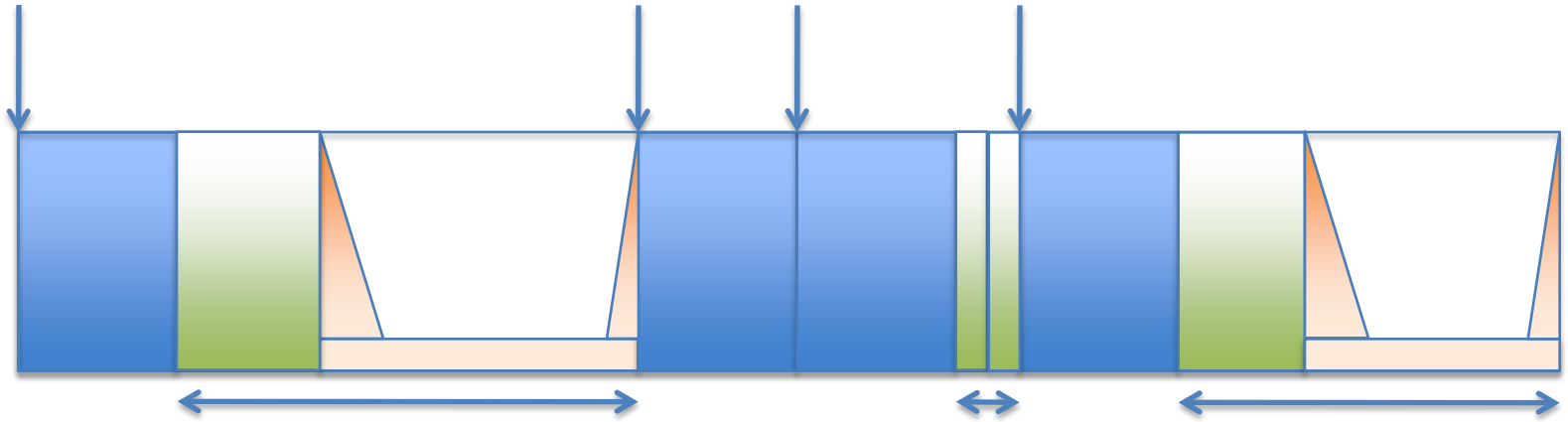
→ **ERP ≈ Energy Savings $*$ 1/Scrubbing Period**

Validation of ERP

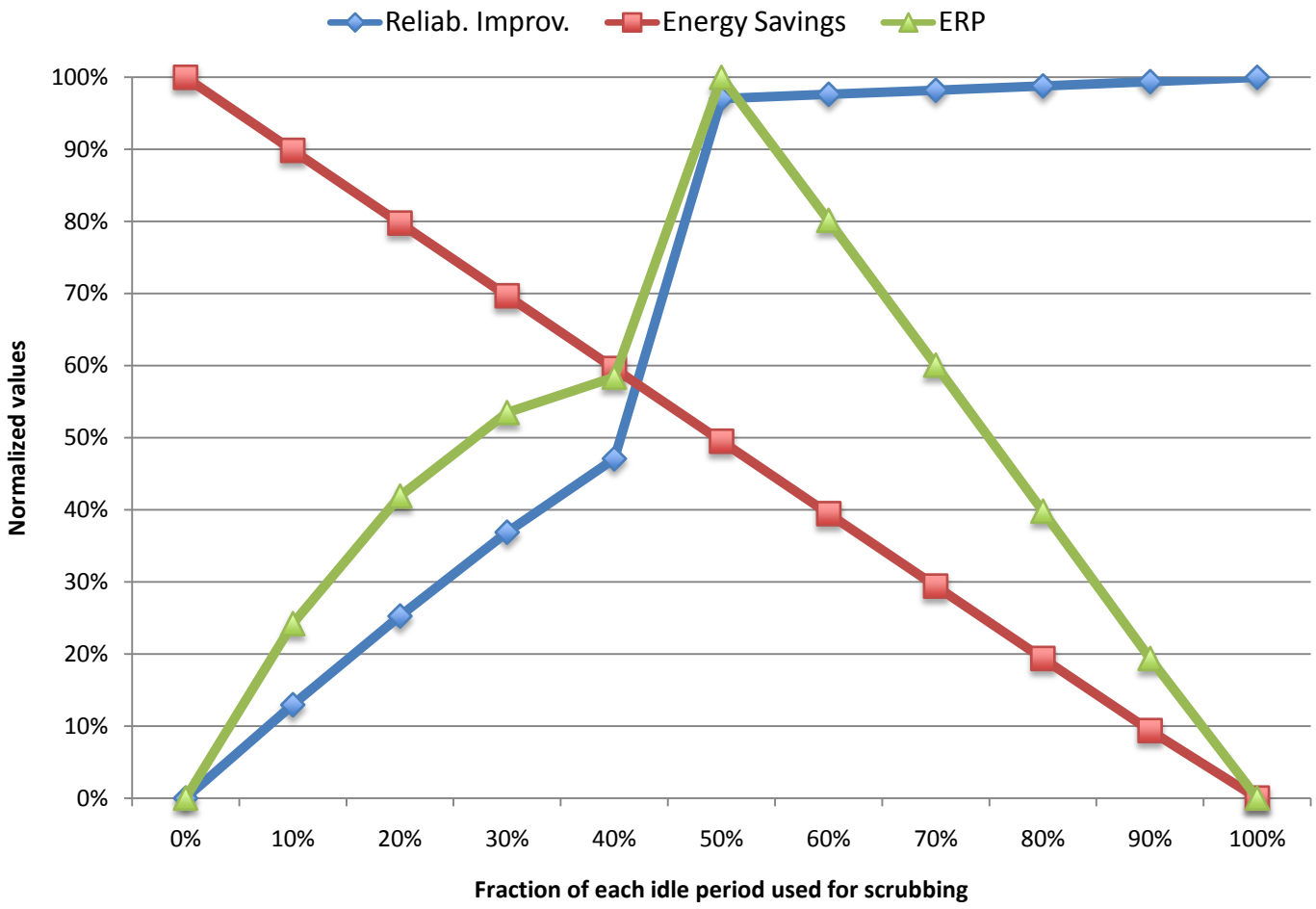
- Employ trace-driven simulation on scrubbing and disk spinning-down
- Use traces of typical desktop applications:
  - Mozilla, mplayer, writer, calc, impress, xemacs

Time-Share Allocation

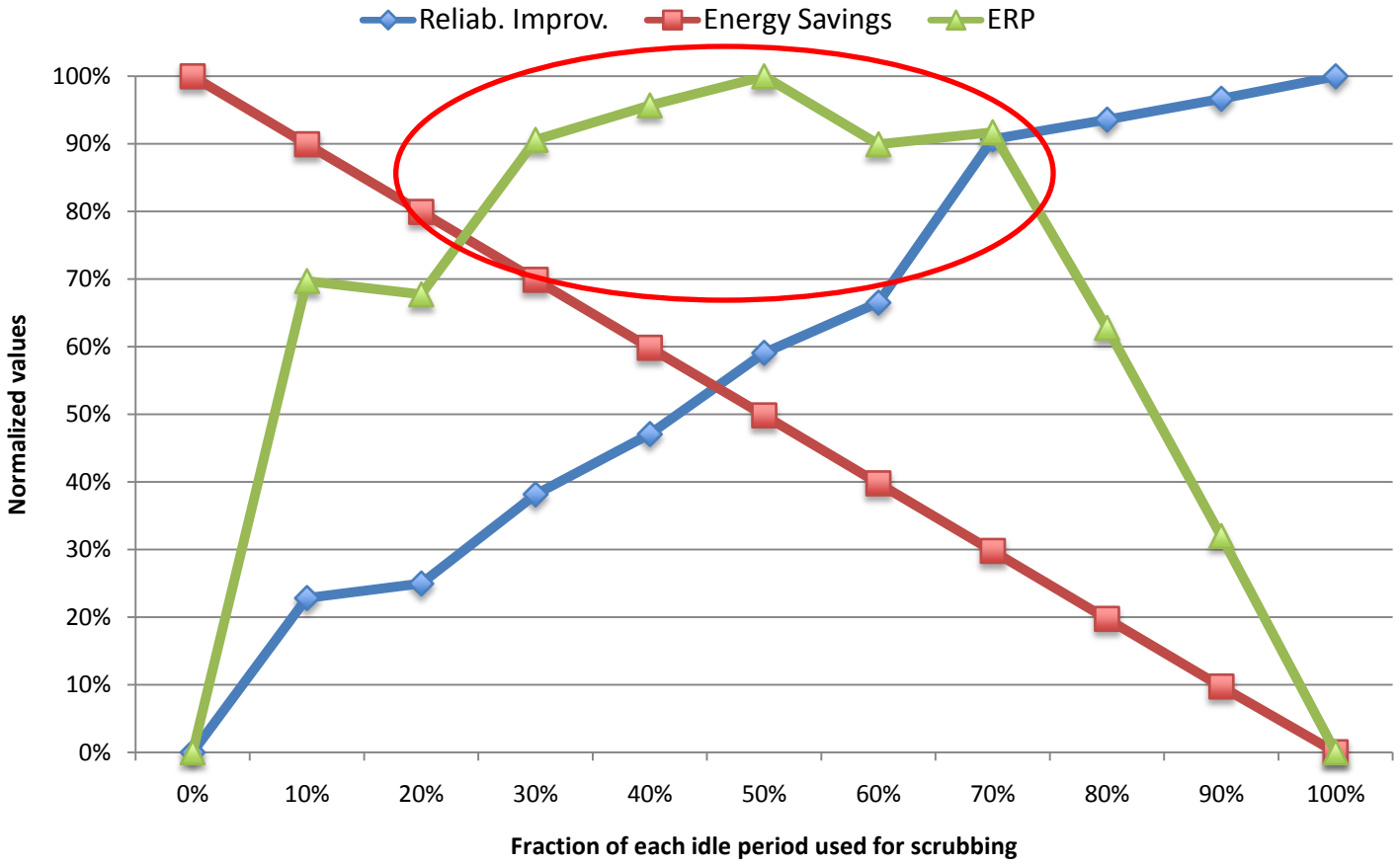
- Preset fraction of idle period used for scrubbing, rest for spinning-down
  - Disk not spun-down during short idle period
  - Optimization: use entire short periods for scrubbing



Time-Share Allocation for Mozilla



Time-Share Allocation in Xemacs

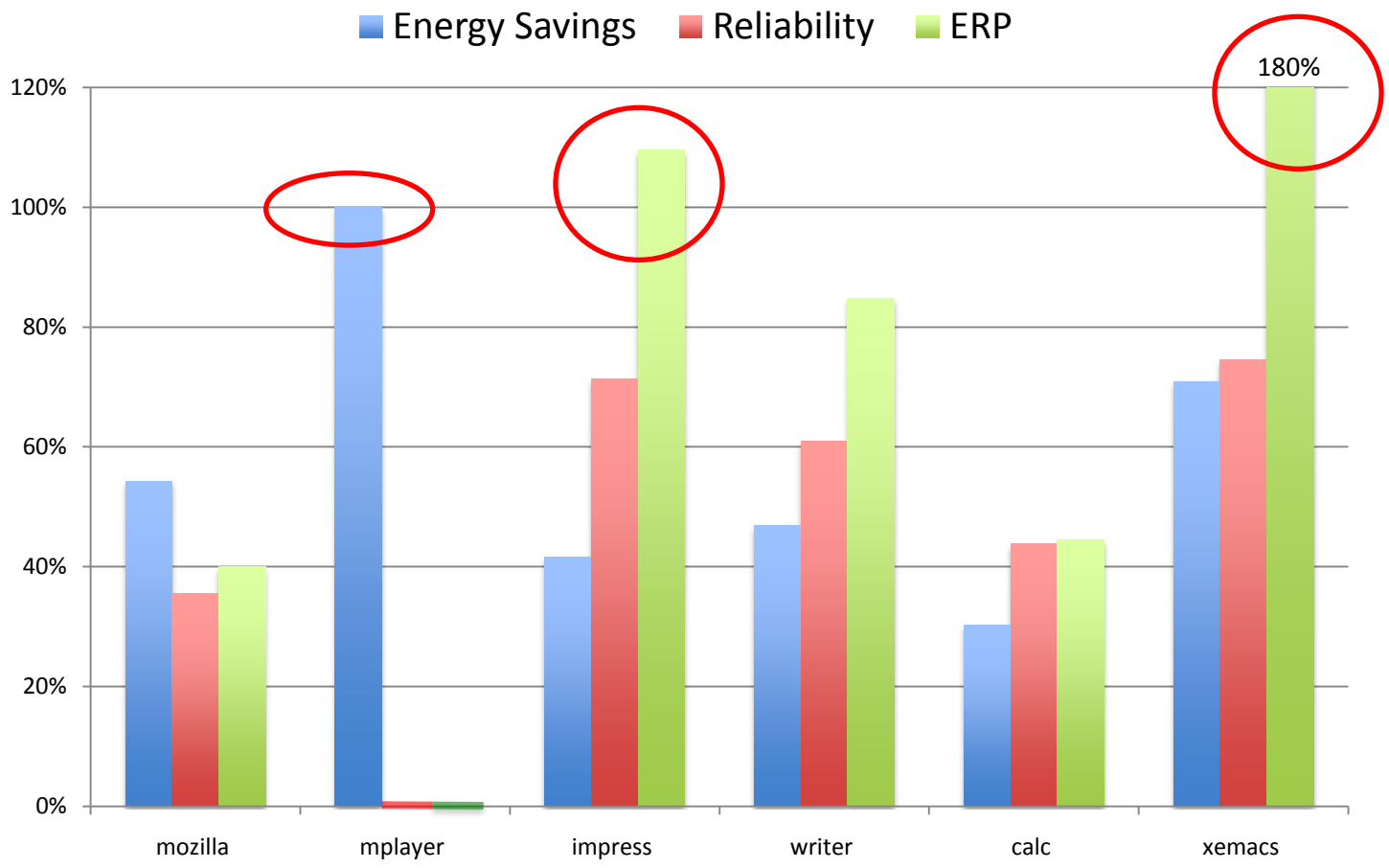


ERP captures a good trade-off point b/w energy savings & reliability improvements

Applying ERP

- Dividing each idle period is impractical
  - Duration unknown
  - Spin-down/up overheads
- Use each idle period for only one task, scrubbing or spinning-down
  - We evaluate three such schemes:
    - Two-phase allocation
    - Scrub only in small idle periods
    - **Alternate allocation**

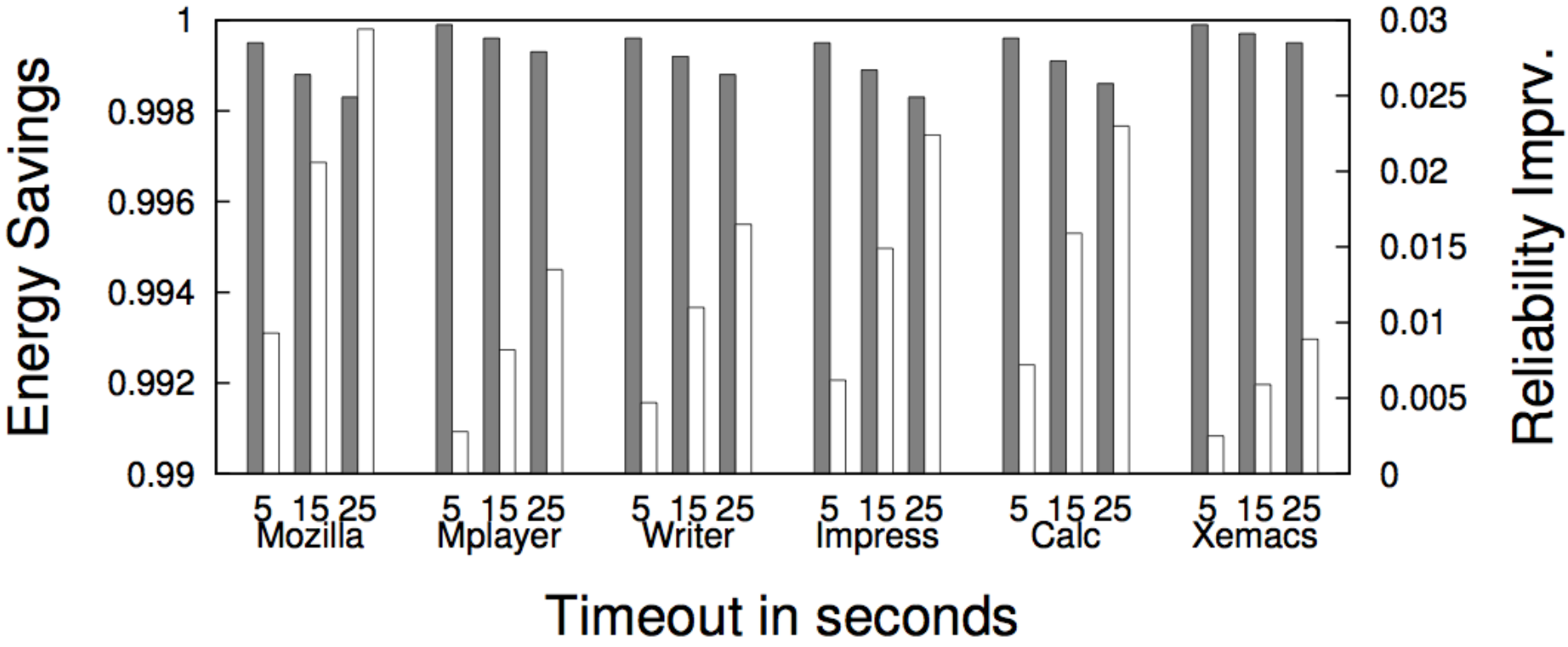
Result: Alternate Allocation



ERP in Timeout-based Approach

- Information about future I/Os is not known a-priori
- Use a timeout-based approach
  - Penalty if another access comes right after spin-off
  - Timeout periods before spin-off are wasted
    - Can be used for scrubbing

Timeout-based Allocation



Small contributions in reliability makes this approach impractical

Thoughts on ERP

- ERP is a intuitive metric for capturing the combined effect of disk scrubbing and spinning-down for saving energy
- ERP can be successfully applied to compare approaches mixing scrubbing and spinning-down

- Future Work
  - Develop a reliability model for IRAW
  - Validate ERP with other workloads
  - Extend our model with multi-speed disks

Role of Storage Errors in HPC Centers

- **Problem**: Large storage systems are error prone
- **Solution 1**: Improve redundancy, add/replace disks
  - Costly, especially for high-speed scratch storage system
  - Mired with acquisition issues, red-tape
- **Solution 2**: Reduce duration of usage
  - Adds software complexity

- We opt for reducing duration of HPC scratch usage

HPC Center Data Offload Problem

- Offloading entails moving large data between center and end-user resources
  - Failure prone: end resource unavailability, transfer errors

  ➔ Offloading errors affect Supercomputer serviceability
- Delayed offloading is highly undesirable
  - From a center standpoint:
    - Wastes scratch space
    - Renders result data vulnerable to purging
  - From a user job standpoint:
    - Increased turnaround time if part of the job workflow depends on offloaded data
    - Potential resubmits due to purging

Upshot: **Timely offloading can help improve center performance**

- HPC acquisition solicitations are asking for stringent uptime and resubmission rates (NSF06-573, …)

Current Methods to Offload Data

- Home grown solutions
  - Every center has its own
- Utilize point-to-point (direct) transfer tools:
  - GridFTP
  - HSI
  - scp
  - …

Limitations of Direct Transfers

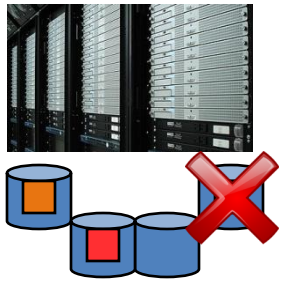
- Require end resources to be available
- Do not exploit orthogonal bandwidth
- Do not consider SLAs or purge deadlines

Not an ideal solution for data-offloading

A Decentralized Data-Offloading Service*

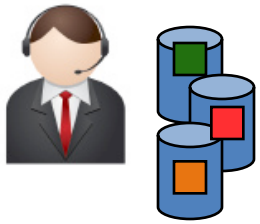
- Utilizes army of intermediate storage locations
- Exploits nearby nodes for moving data
- Supports multi-hop data migration to end users
- Decouples offloading and end-users availability
- Integrates with real-world tools
  - Portable Batch System (PBS)
  - BitTorrent
- Provides multiple fault-tolerant data flow paths from the center to end users

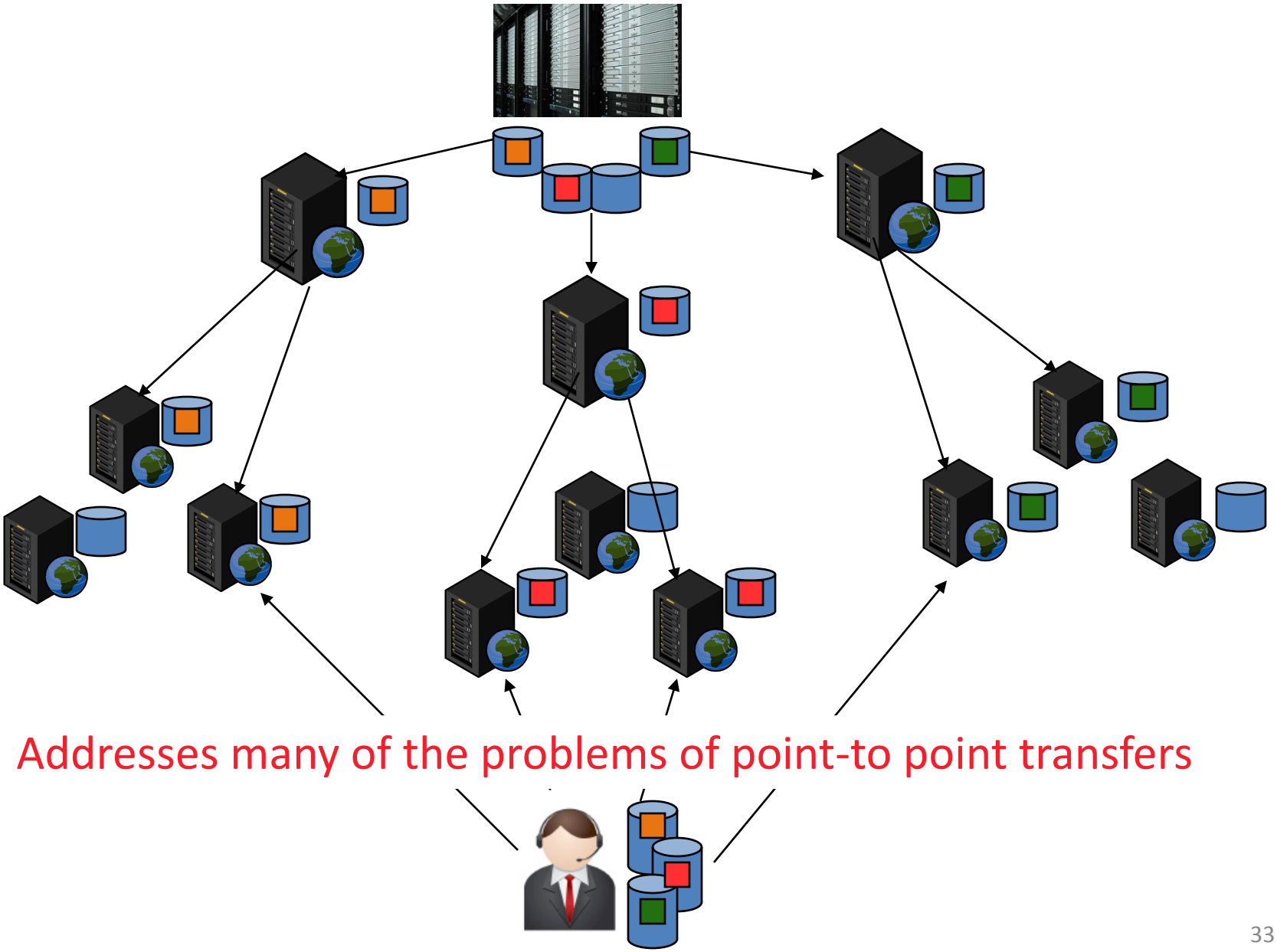
* Timely Offloading of Result-Data in HPC Centers, Monti, Butt, Vazhkudai, ICS'08



Transfer limited by end-user available bandwidth

Delayed transfer & storage failures may result in loss of data!





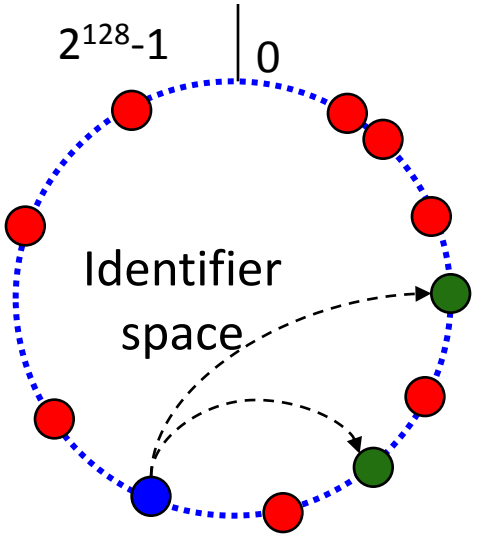
Addresses many of the problems of point-to point transfers

Challenges Faced in Our Approach

1. Discovering intermediate nodes
2. Providing incentives to participate
3. Addressing insufficient participants
4. Adapting to dynamic network behavior
5. Ensuring data reliability and availability
6. Meeting SLAs during the offload process

1. Intermediate Node Discovery

- Utilize DHT abstraction provided by structured p2p networks
- Nodes advertise their availability to others
- Receiving nodes *discovers* the advertiser



- Discovered nodes utilized as necessary

2. Incentives to Participate in Offload Process

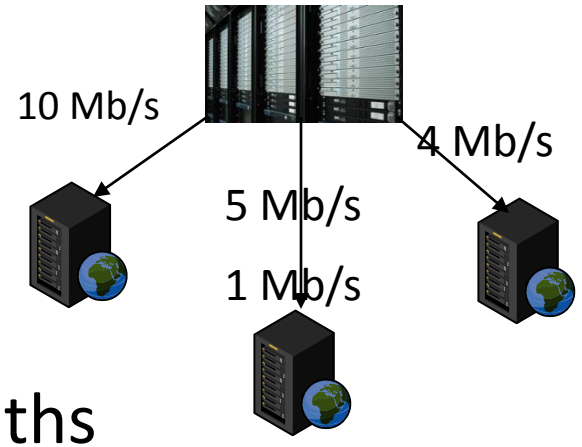
- Modern HPC jobs are often collaborative
  - "Virtual Organizations" - set of geographically distributed users from different sites
  - Jobs in TeraGrid usually from such organizations
- Resource bartering among participants to facilitate each others offload over time
- Nodes specified and trusted by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

3. Addressing Insufficient Participants

- Problem: Sufficient participants not available
- Solution: Use Landmark Nodes
  - Nodes that are stable and available
  - Willing to store data
- Leverage out-of-band agreements
  - Other researchers who are also interested in the data
  - Data warehouses
    - cheaper option than storing at the HPC center
- Note: Landmark Nodes used as a safety net!

4. Adapting Data Distribution To Dynamic Network Behavior

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



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

5. 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 at the Center or intermediates
     - End user may pay for coding at the Center

6. Managing SLAs during Offload

- Use NWS to measure available bandwidths
  - Use Direct if it can meet SLA
  - Otherwise, perform decentralized/staged offload

- In case end host fails or cannot meet SLA
  - Utilize decentralized offload approach

$$T_{offload} < Min(D_{purge}, J_{SLA})$$

Integrating Staged Offload with PBS

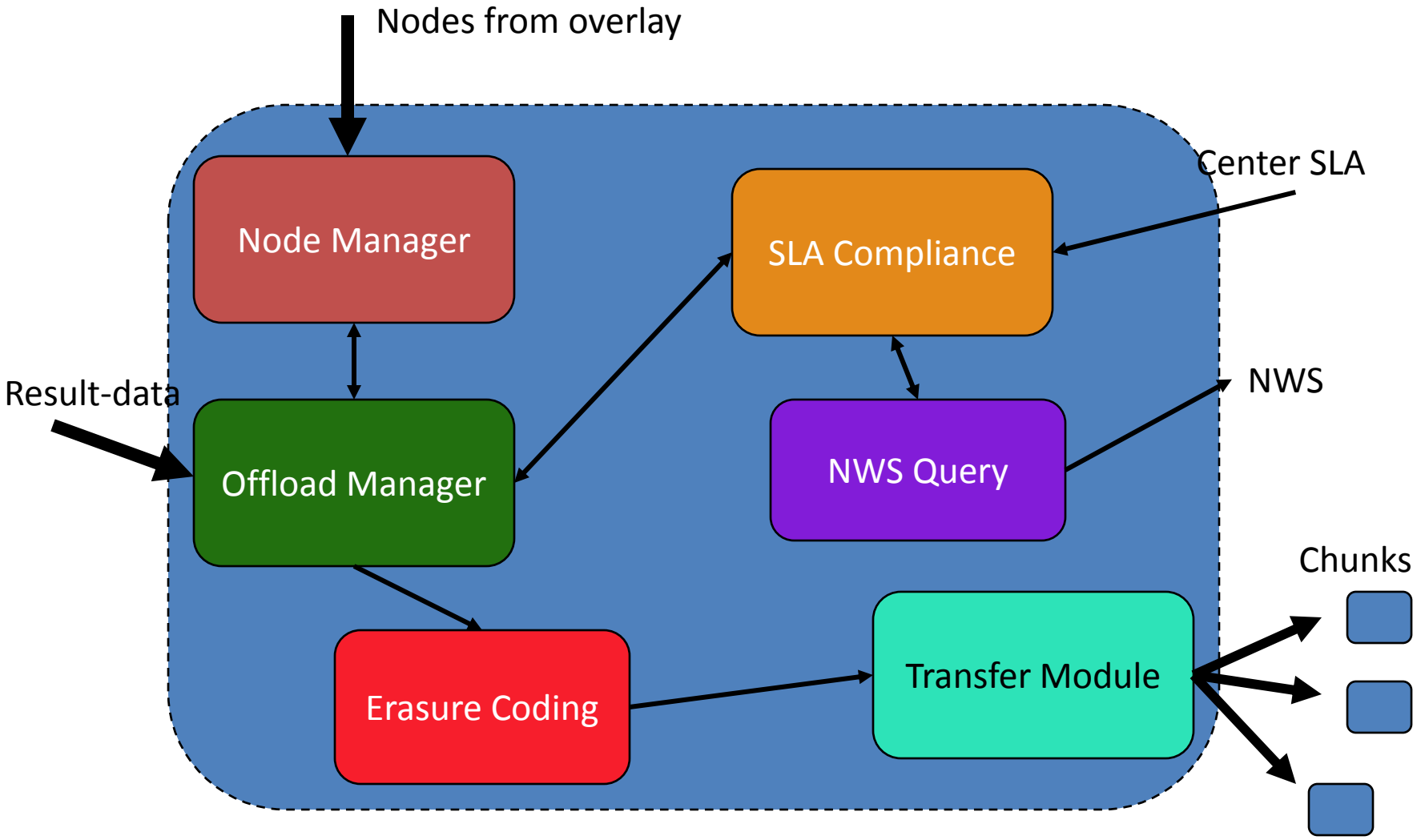
- Provide new PBS directives
  - Specifies destination, intermediate nodes, and deadline

```
#PBS -N myjob
#PBS -l nodes=128, walltime=12:00
mpirun -np 128 ~/MyComputation
#Stageout Output DestinationSite
#InterNode node1.Site1:49665:50GB
...
#InterNode nodeN.SiteN:49665:30GB
#Deadline 1/14/2007:12:00
```

Adapting BitTorrent Functionality to Data Offloading

- Tailor BitTorrent to meet the needs of offloading
- Restrict the amount of result-data sent to a peer
  - Peers with less storage than the result-data 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

Putting it all Together



Evaluation Objectives

1. Compare with direct transfer, and BitTorrent
2. Observe how system reacts to failures and bandwidth fluctuations:
   a. How SLAs are enforced?
   b. How fault tolerance is achieved?
3. Validate our method as a viable alternative to other offloading methods

Evaluation: Experimental Setup

- PlanetLab test bed
  - 22 PlanetLab nodes:

    center + end user + 20 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	286	727	1443	5834
Offload	38	95	169	570
Push	82	179	349	1123
Pull	29	93	202	562

A staged offload is capable of significantly improving offload times

Times are in seconds

Results: Data Transfer Times with Respect to Standard BitTorrent

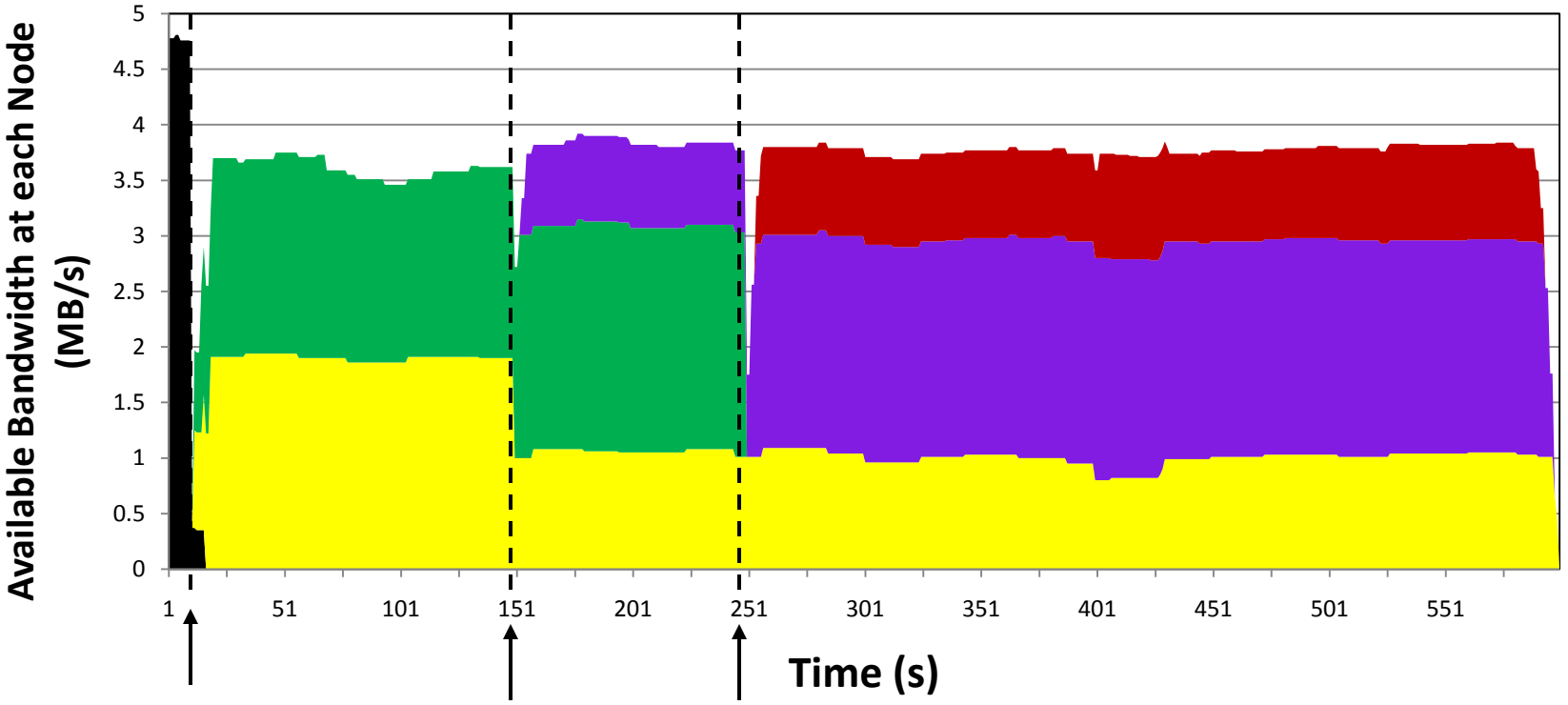
Phase	BitTorrent	Our Method
Send one copy from center (Offload)	1172	570
Send to all intermediate nodes (Push)	1593	1123
Submission site download (Pull)	571	562

Monitoring based offload is capable of outperforming standard BitTorrent

Times are in seconds

Transferring 2.1 GB file

Results: Adapting to Dynamic Network Behavior

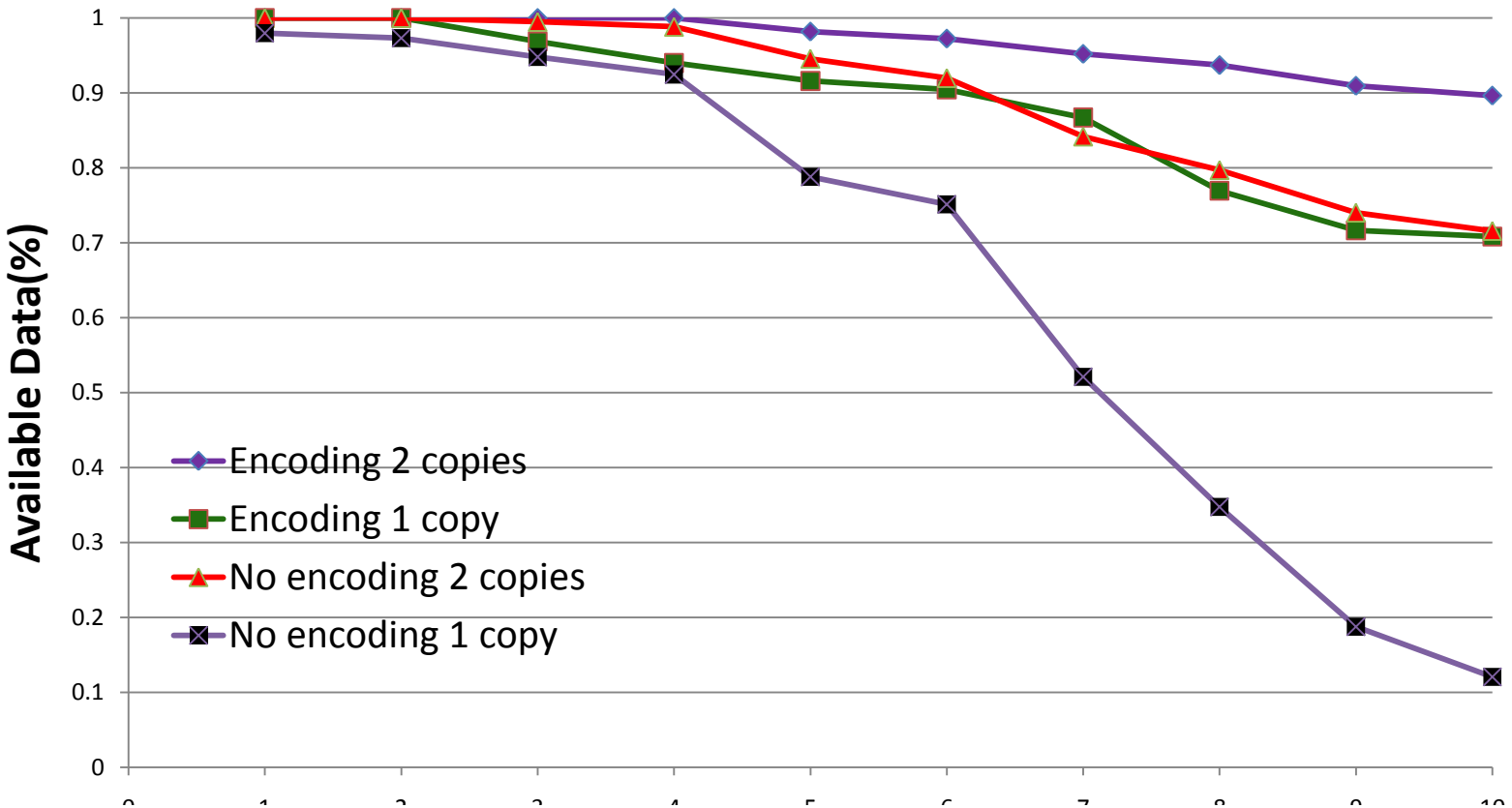


A staged offload is capable of adapting to bandwidth changes or failures

SLA is 600 seconds

Transferring 2.1 GB file

Results: Replication vs. Erasure Coding



A staged offload can protect data even when many nodes fail

Transferred 2.1 GB file

Randomly failed 10 nodes during the transfer

Thoughts on Eager Offloading

- A fresh look at Offloading
  - Decentralized approach
  - Monitoring-based adaptation
- Considers SLAs and purge policies
- Integrated with real-world tools
- Provides high reliability for data
- Outperforms direct transfer by **90.2%** in our experiments

Some projects we are involved in

- Enabling High-performance I/O for asymmetric multi-core systems (GPUs, PS3, …)
- Simulation/capacity planning tools for cloud computing
- Advanced data caching and prefetching
- Just-in-time Data Staging
- Managing HPC center scratch space as a hierarchical cache
- I/O shaping for HPC applications
- Hybrid disk modeling
- Real-time data processing for advanced nano-bionics

- On the web: http://research.cs.vt.edu/dssl
- Contact email: butta@cs.vt.edu