FALCON – A System for Reliable Checkpoint Recovery in Shared Grid Environments

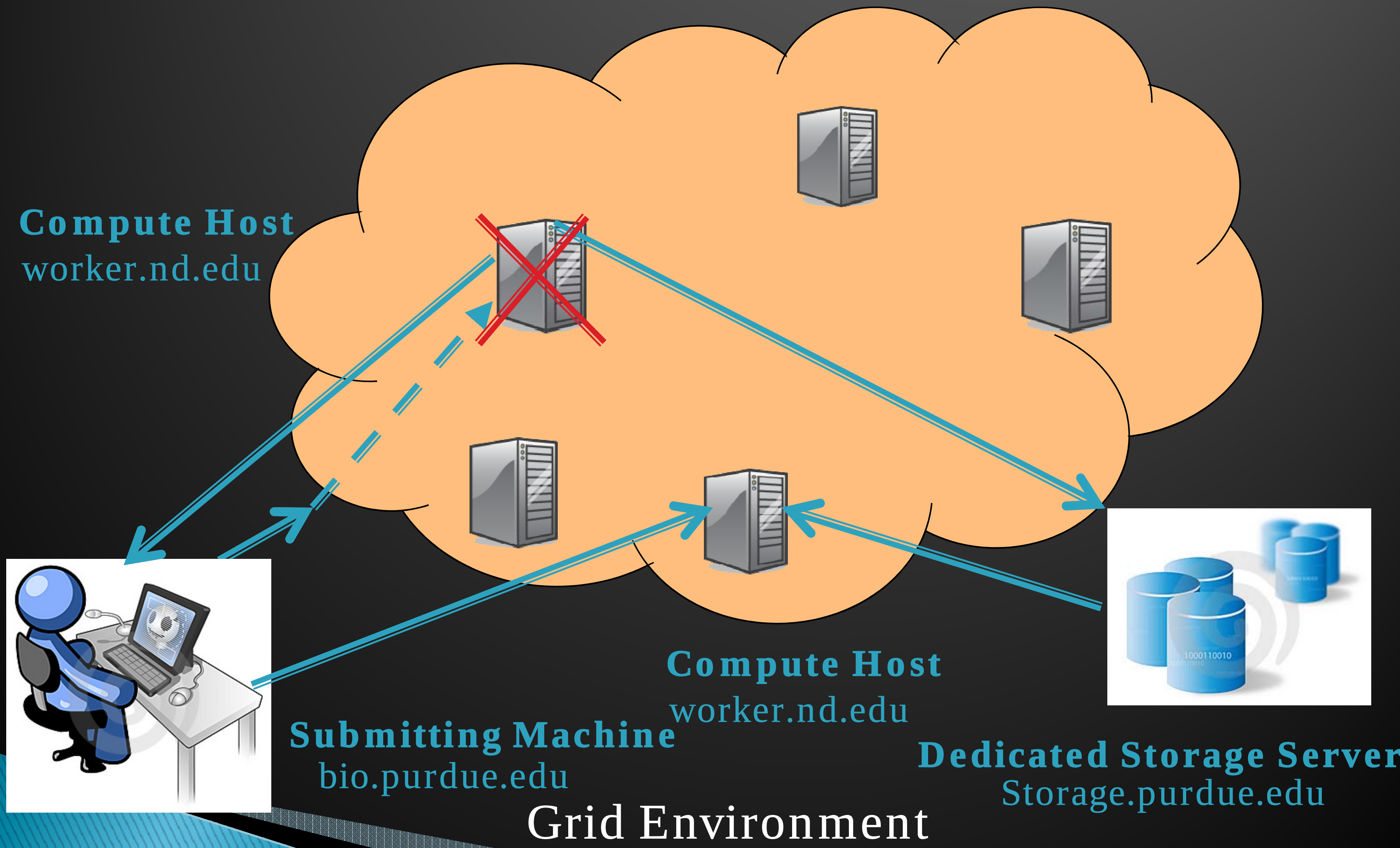
Tanzima Zerin Islam, Saurabh Bagchi
And
Rudolf Eigenmann
School of Electrical and Computer Engineering
Purdue University

Work supported by:
National Science Foundation
Purdue Research Foundation

Background

- What is a grid or cycle sharing (CS) system
  - Machines share their unused computation cycles
- What is a Fine-Grained CS (FGCS)?
  - Guest and host jobs can coexist
  - Example: Condor
- Resources are extremely volatile
  - In BoilerGrid (DiaGrid), eviction rate – 1.3 per job per hour on average
  - Checkpoint-recovery provides fault-tolerance

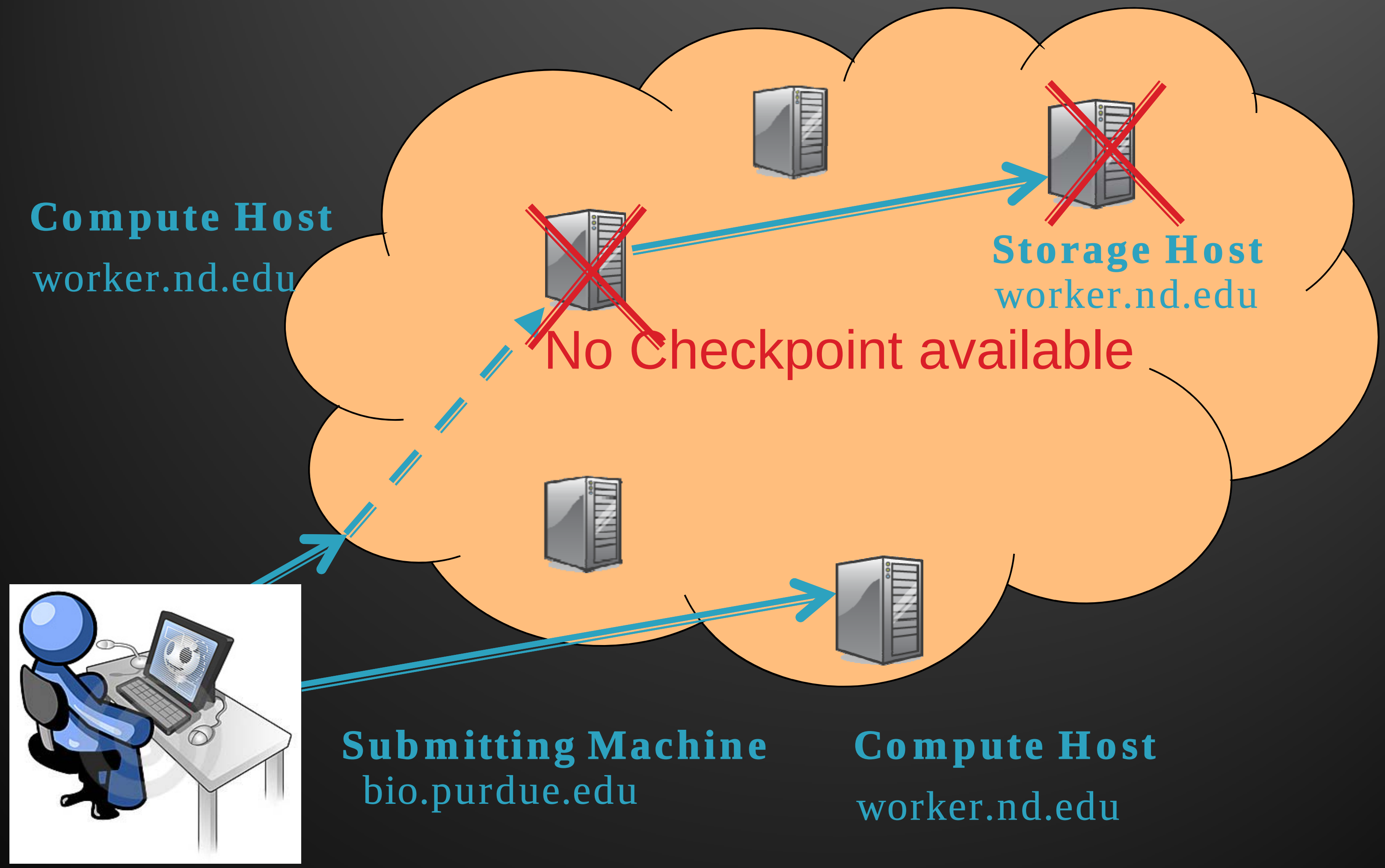
State of the Art Checkpoint-Recovery Scheme



Problem Motivation

- High overhead for application users
  - Submitting machine
    - If the submitter is behind a slow network (say, DSL modem)
  - Central storage server
    - High latency of transferring checkpoints back and forth between different university campuses (12% of the time)
    - High overhead when multiple machines are sending data to a single server
    - High overhead of sending data to a loaded server
- Stress on shared network resource
  - Transferring large amount of checkpoint data (gigabytes)
  - Transferring data across distant points in the network

Potential Solution and Challenges



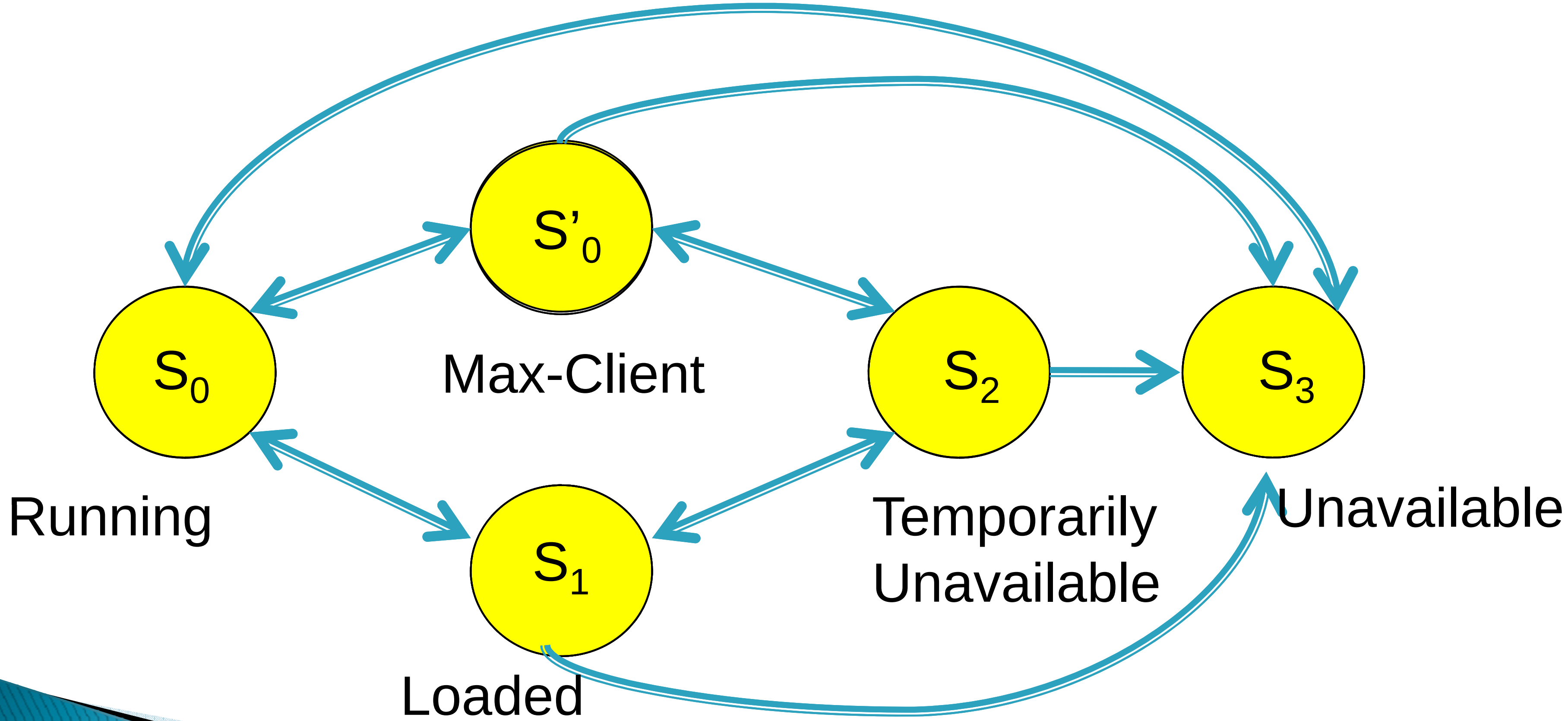
Shared Grid Environment

Contribution

- Goal: Can we improve the performance of the guest jobs by storing checkpoints in shared grid environment?
- Developed a reliable checkpoint-recovery system FALCON
  - Provides fault-tolerance through “Erasure Coding”
  - Selects reliable storage hosts which are nearby
    - Builds a failure model for storage hosts
  - Stores and retrieves checkpoints in efficient manner
- Deployed FALCON in BoilerGrid (DiaGrid)
  - Performance improvement of benchmark applications in production grid is between 11% to 44%

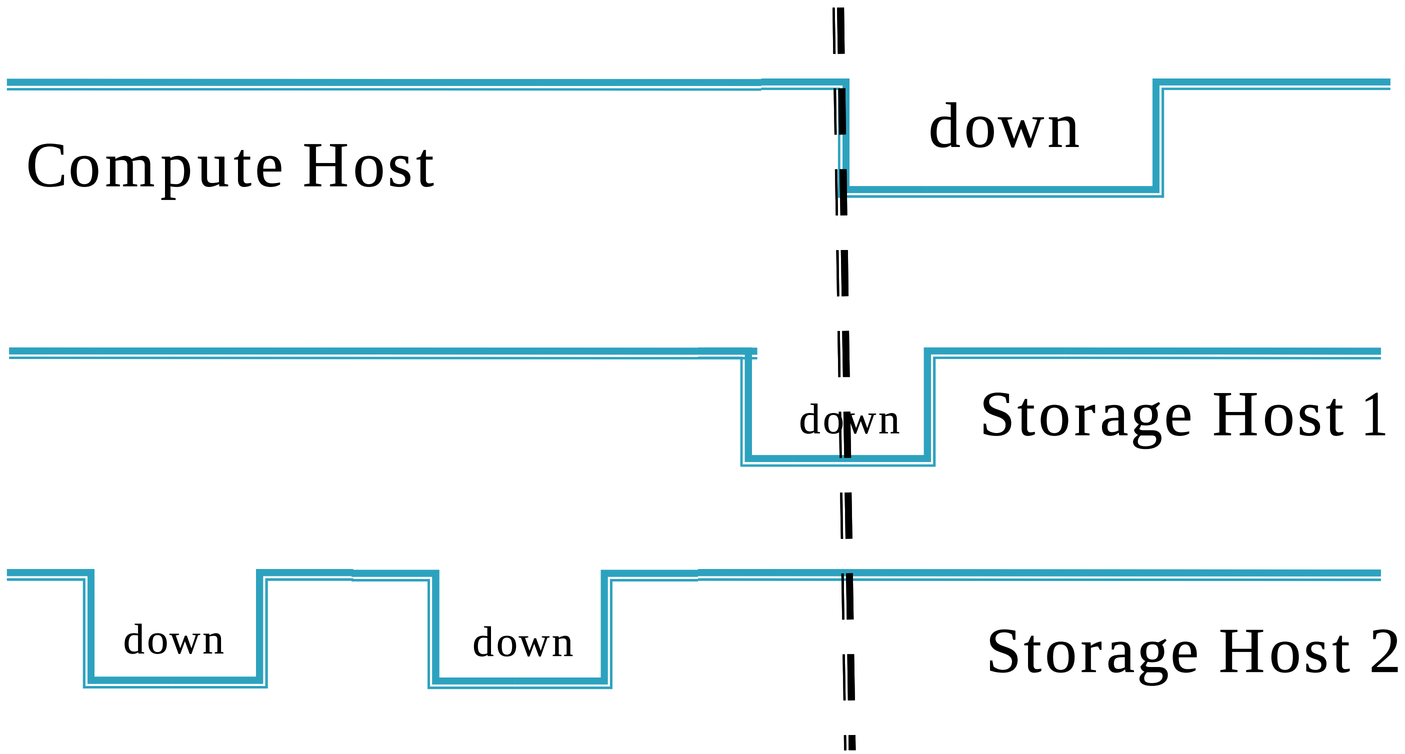
Failure Model

- Aids in predicting availability of the storage nodes
- Load: %utilization of I/O



Storage Repository Selection

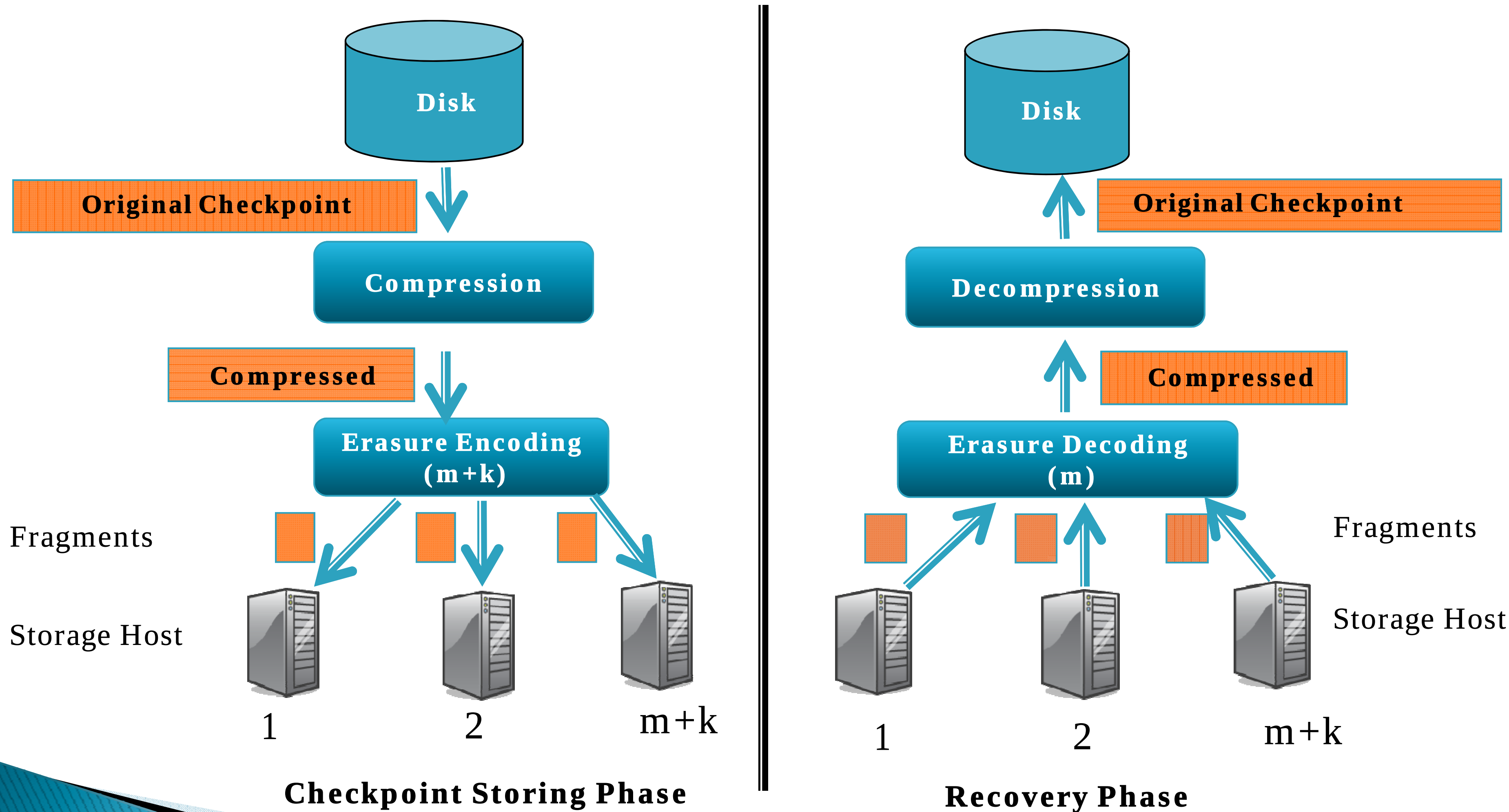
- Predict availability of storage nodes
  - Correlated temporal reliability



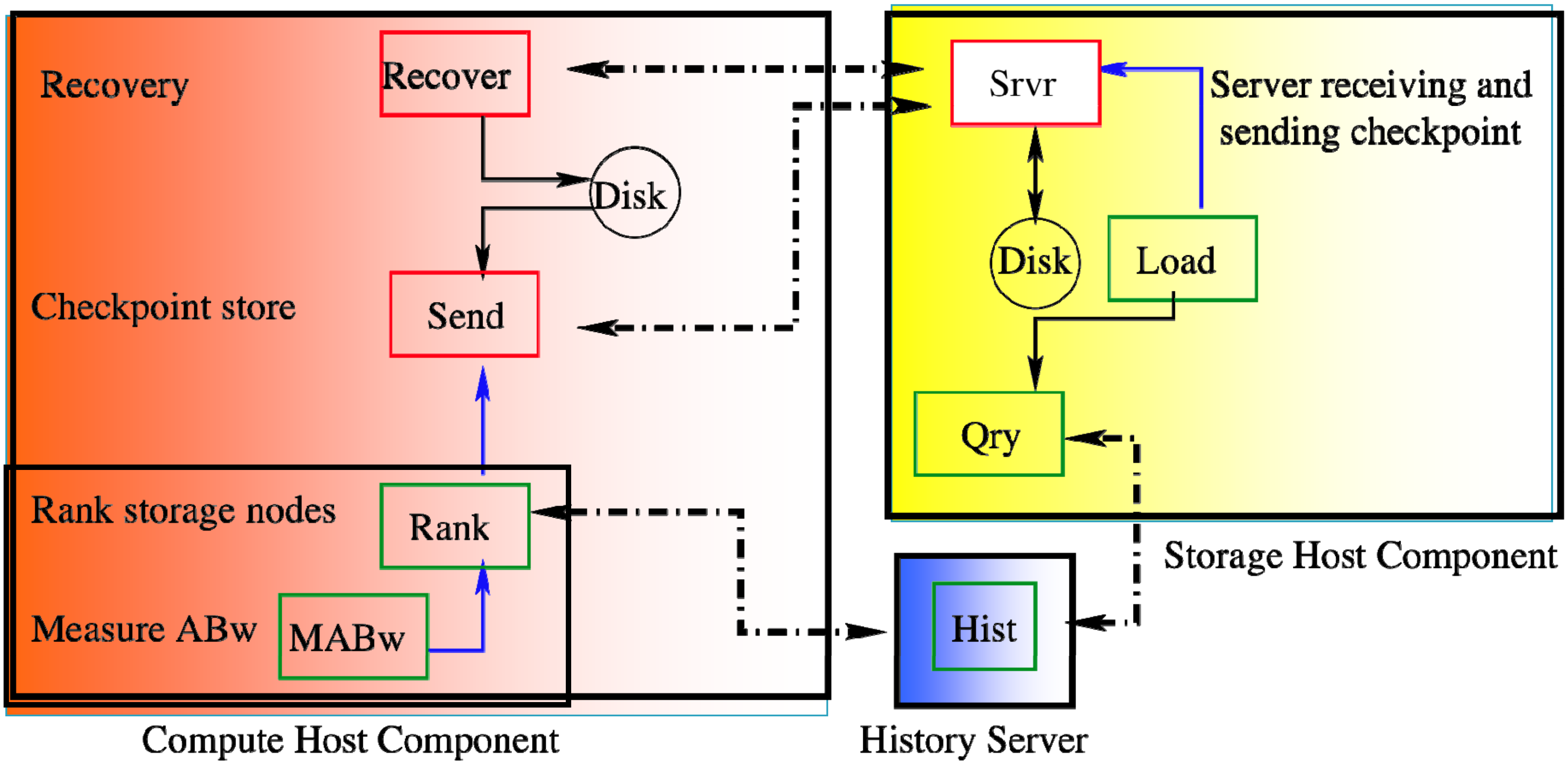
Storage Repository Selection

- Calculate network transfer overhead
  - Network Overhead = Amount of data to send (MB) / Available Bandwidth between a storage host and a compute host
- Minimize an objective function
  - Objective function: checkpoint storing overhead – benefit from the fact that a job can restart from the last saved state
  - Overhead includes network overhead
  - Benefit computed using the correlated temporal reliability
- Select a set of $m+k$ storage nodes that minimizes this objective function

Checkpoint-Recovery Scheme



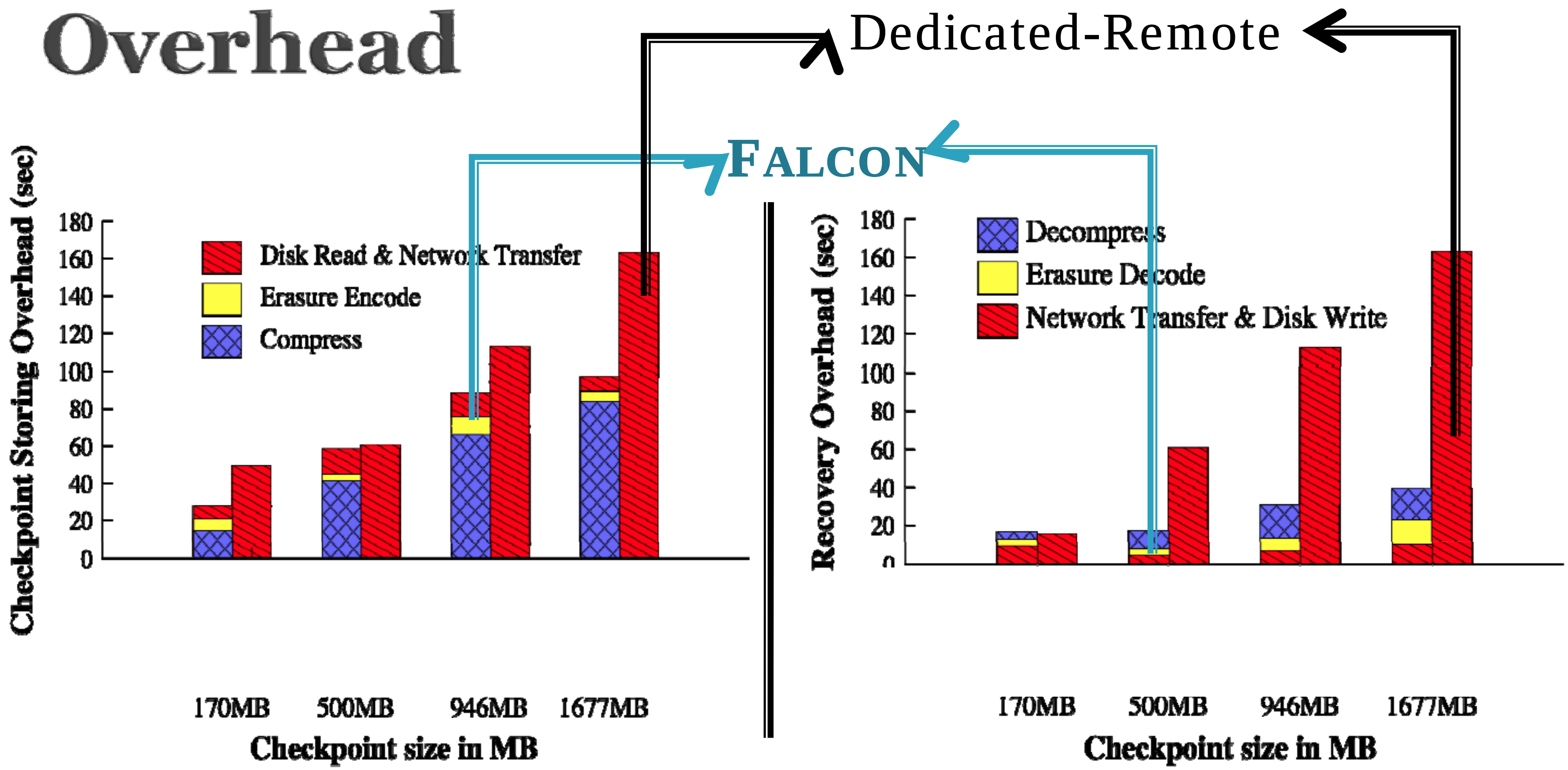
Structure of FALCON



Evaluation

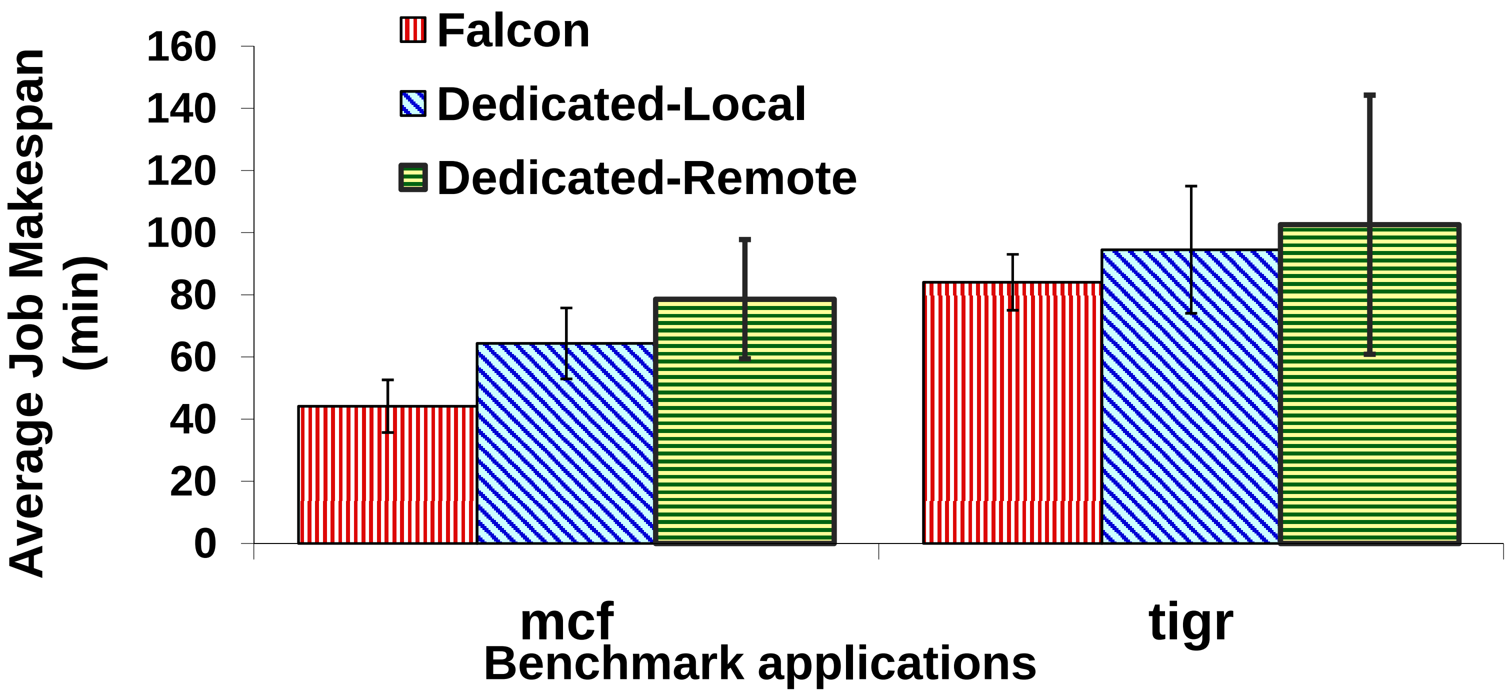
- Overall performance evaluation:
  - Average job makespan – the time a job takes to complete
- Efficiency of the checkpoint-recovery schemes:
  - Checkpoint storing overhead
  - Recovery Overheads
- Setup:
  - Submitted jobs to BoilerGrid
  - Applications – MCF (SPEC CPU 2006), TIGR (BioBench)
  - Erasure encoding parameters: m=3, k=2

Checkpoint Storing & Recovery Overhead



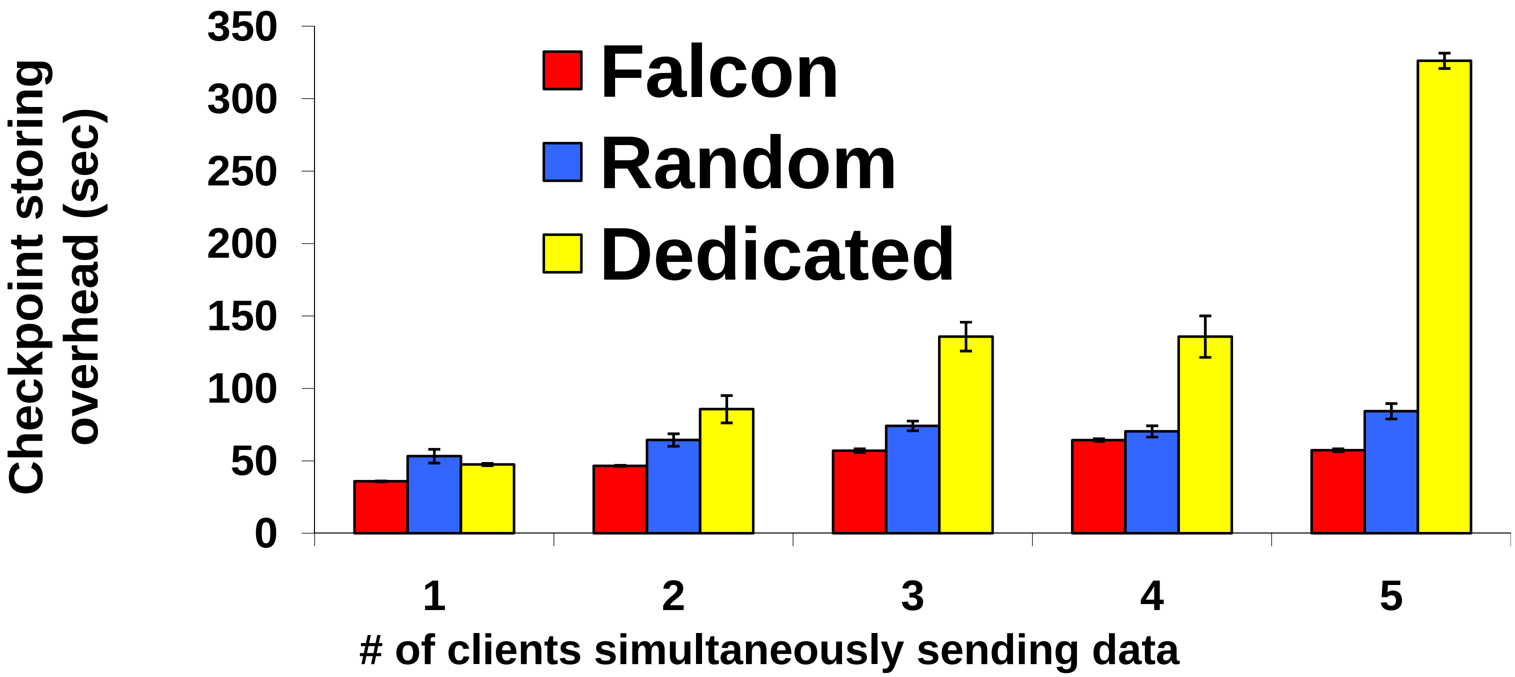
- Performance of Falcon scales with the increase in the checkpoint sizes
- Lower network transfer overhead and lower utilization of shared network bandwidth

Overall Performance Comparison



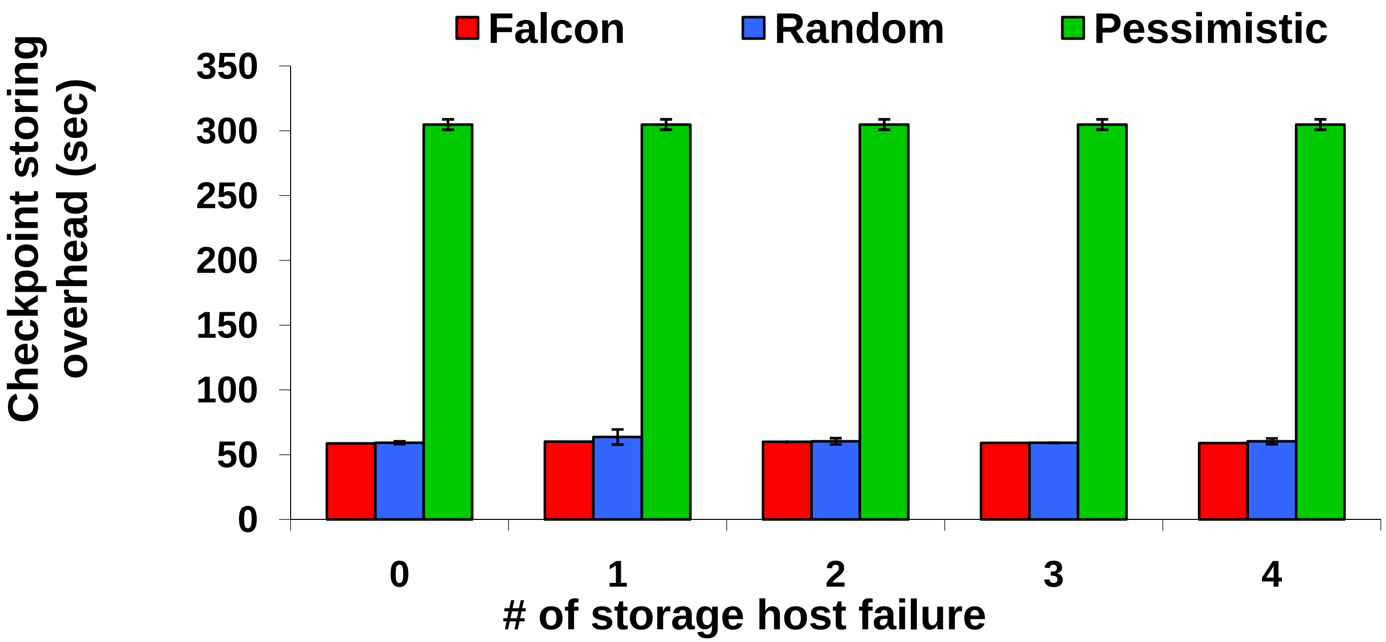
- Performance improvement of the applications are between 11% and 44%

Handling Simultaneous Clients



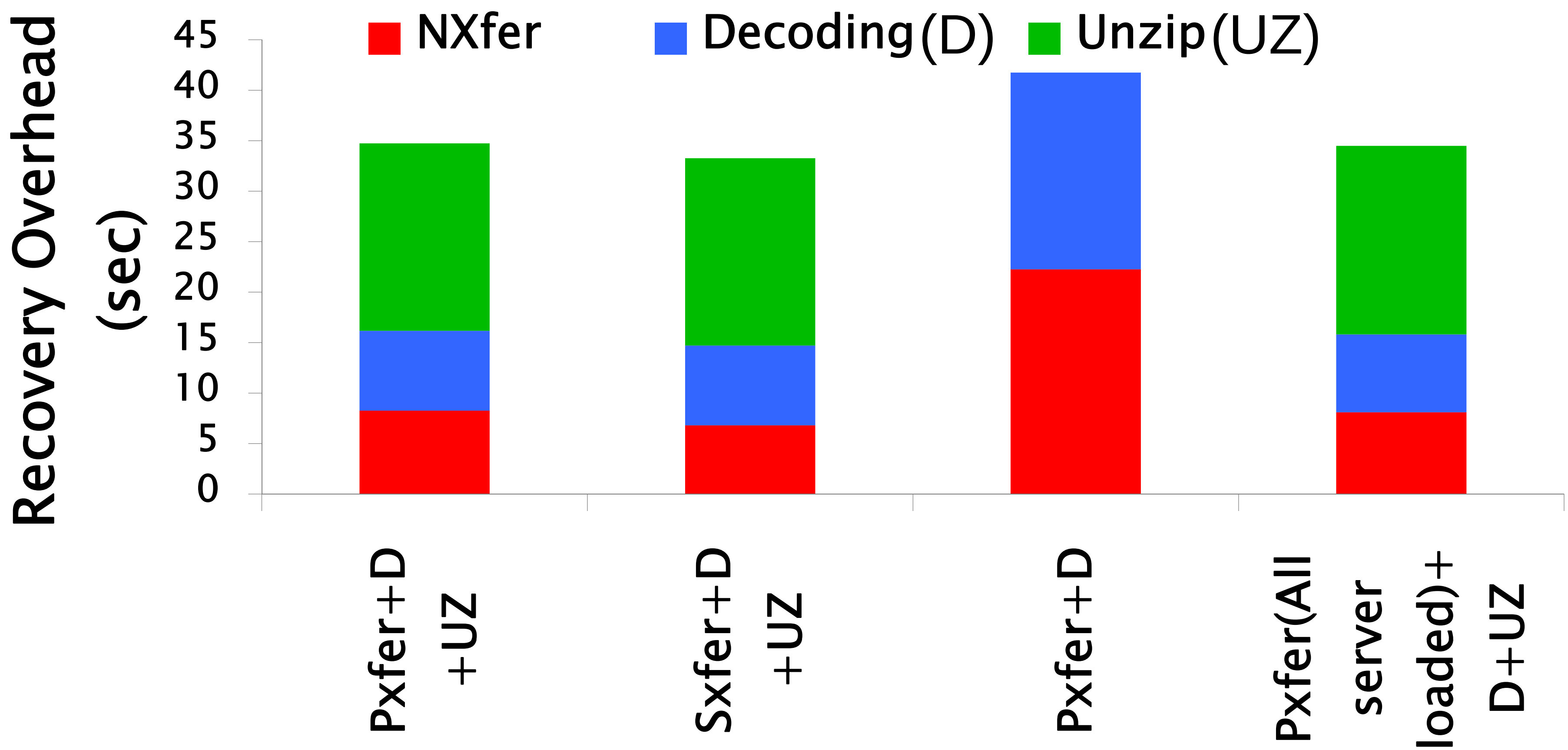
- Performance of dedicated scheme suffers
- Performance of Random scheme suffers because of choosing machine behind slow network

Handling Storage Failures



- Robustness at no extra cost for Falcon
- Pessimistic incurs large overhead

Contributions of Components



- Pxfer – parallel network transfer, Sxfer – serial network transfer
- Largest contribution comes from compression

Conclusion

- Developed a multi-state failure model for storage nodes
  - Also provides load balancing
- Developed a failure-aware storage selection technique
- Checkpoint-recovery scheme
  - Fault-tolerant
  - Scalable
  - Robust
- All the components are user level applications
- No simulation, no synthetic checkpoint
- User level checkpoint
- Question: Can we improve the performance of the guest jobs by storing checkpoints in a shared grid environment?
  - Answer: Yes FALCON can

Thank You

Future Direction

- How about taking advantage of the multiple cores available on the compute hosts?
- How about looking at other system parameters in addition to the I/O load to predict the failure states of the storage hosts?
- How to provide security to the storage hosts in such a shared grid environment?