Performance Analysis of Mixed Distributed Filesystem Workloads

Esteban Molina-Estolano, Maya Gokhale, Carlos Maltzahn, John May, John Bent, Scott Brandt

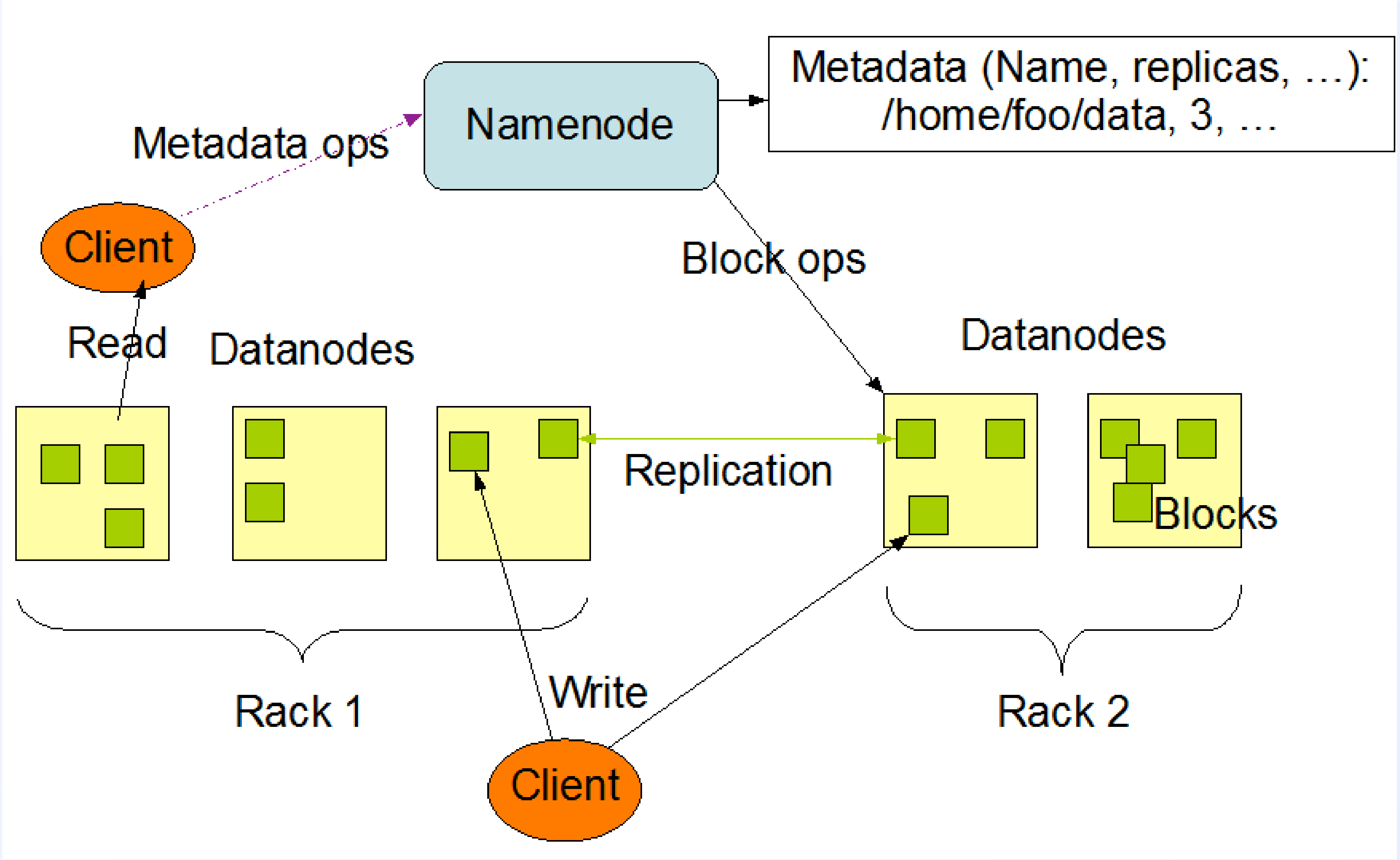
Motivation

- Hadoop-tailored filesystems (e.g. CloudStore) and high-performance computing filesystems (e.g. PVFS) are tailored to considerably different workloads
- Existing investments in HPC systems and Hadoop systems should be usable for both workloads
  - Avoid dedicating separate hardware for each type of workload
- Goal: Examine the performance of both types of workloads running concurrently on the same filesystem
- Goal: collect I/O traces from concurrent workload runs, for parallel filesystem simulator work

MapReduce-oriented filesystems

- Large-scale batch data processing and analysis
- Single cluster of unreliable commodity machines for both storage and computation
- Data locality is important for performance
- Examples: Google FS, Hadoop DFS, CloudStore

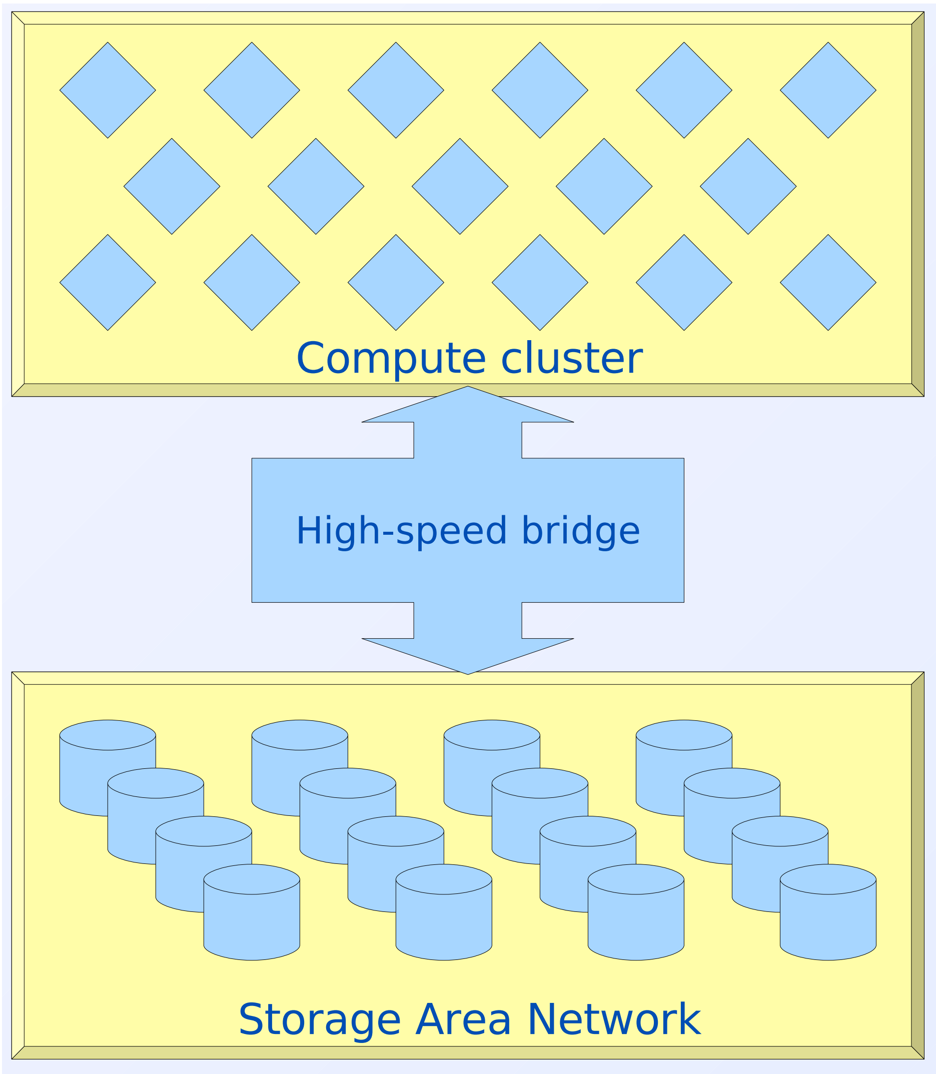
Hadoop DFS architecture



http://hadoop.apache.org

High-Performance Computing filesystems

- High-throughput, low-latency workloads
- Architecture: separate compute and storage clusters, high-speed bridge between them
- Typical workload: simulation checkpointing
- Examples: PVFS, Lustre, PanFS



Running each workload on the non-native filesystem

- Two-sided problem: running HPC workloads on a Hadoop filesystem, and Hadoop workloads on an HPC filesystem
- Different interfaces:
  - HPC workloads need a POSIX-like interface and shared writes
  - Hadoop is write-once-read-many
- Different data layout policies

Running HPC workloads on a Hadoop filesystem

- Chosen filesystem: CloudStore
  - Downside of Hadoop's HDFS: no support for shared writes (needed for HPC N-1 workloads)
  - Cloudstore has HDFS-like architecture, and shared write support

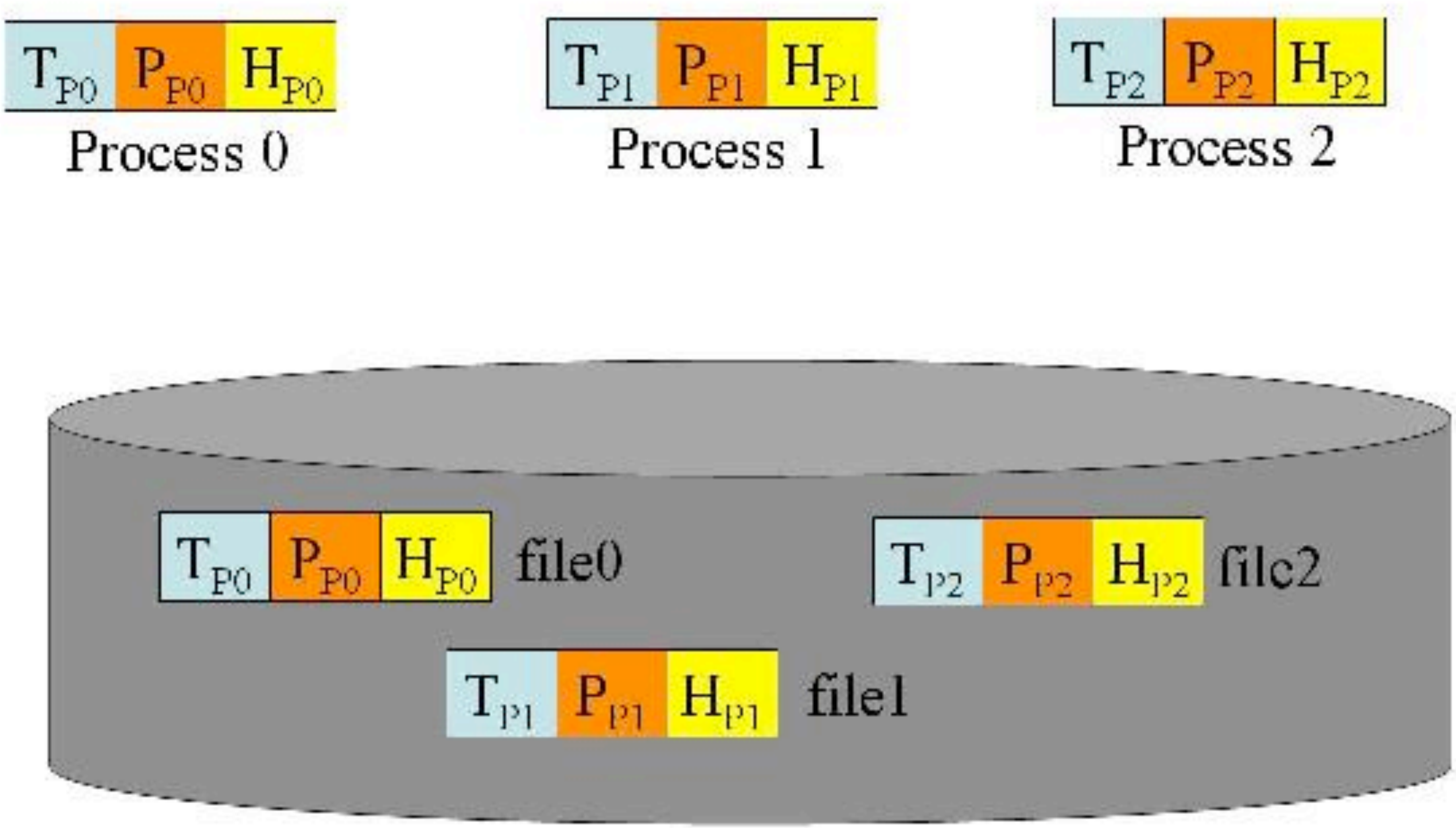
Running Hadoop workloads on an HPC filesystem

- Chosen HPC filesystem: PVFS
  - PVFS is open-source and easy to configure
  - Tantisiriroj et al. at CMU have created a shim to run Hadoop on PVFS
  - Shim also adds prefetching, buffering, exposes data layout

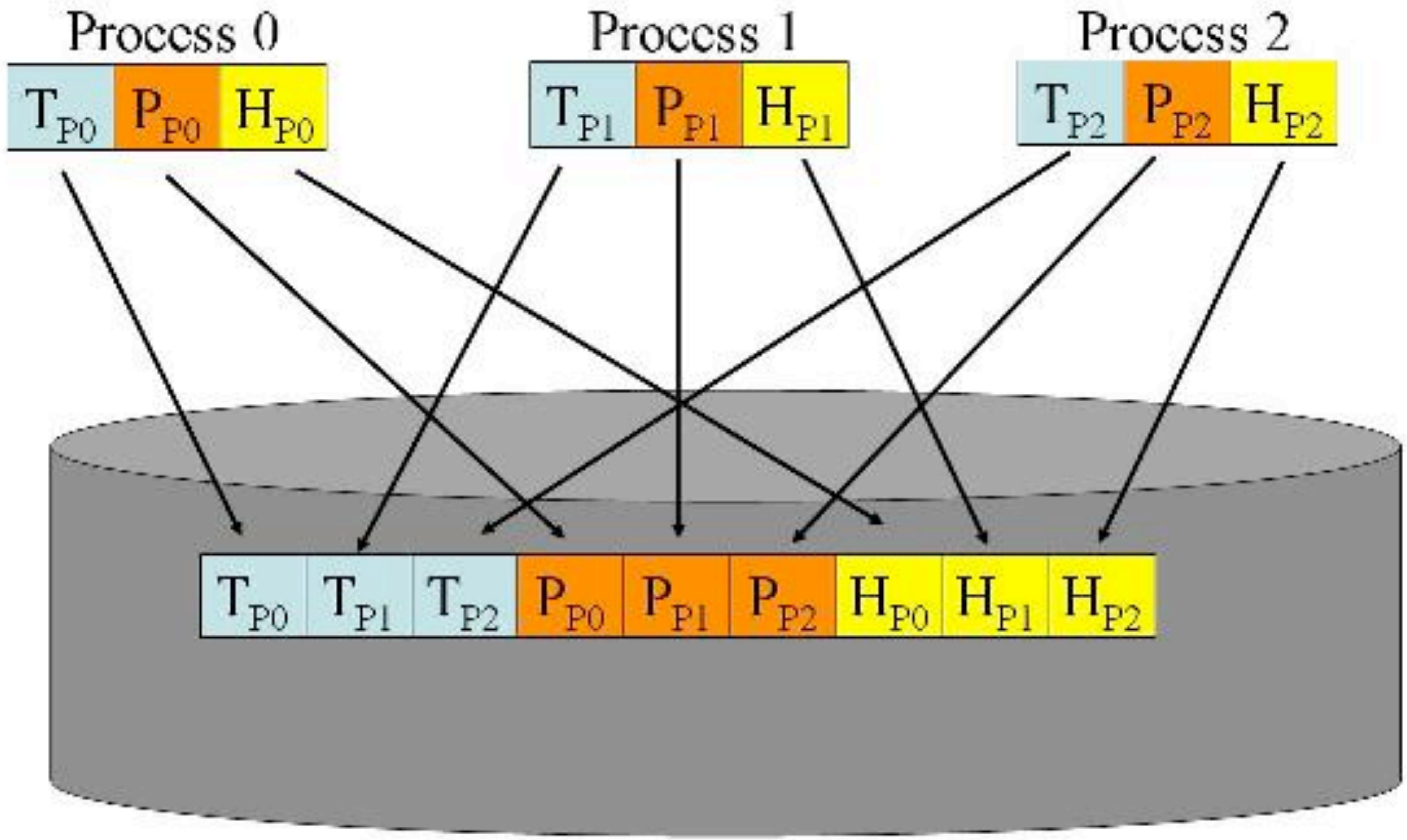
The two concurrent workloads

- IOR checkpointing workload
  - writes large amounts of data to disk from many clients
  - N-1 and N-N write patterns
- Hadoop MapReduce HTTP attack classifier (TFIDF)
  - Using a pre-generated attack model, classify HTTP headers as normal traffic or attack traffic

N-to-N example



N-to-1 strided example

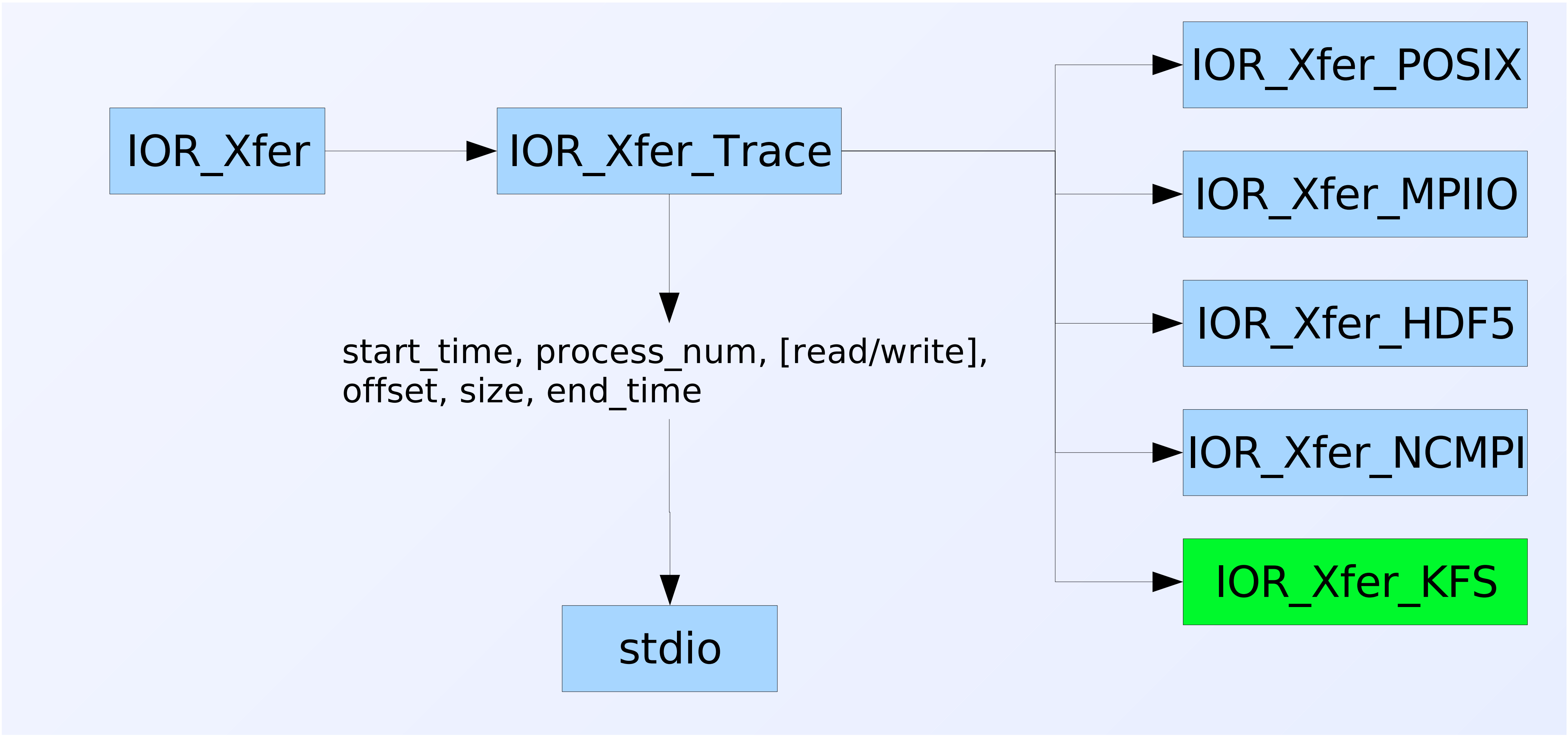


Tracing infrastructure

- We gather traces to use for our parallel filesystem simulator
- Existing tracing mechanisms (e.g. strace, Pianola, Darshan) don't work well with Java or CloudStore
- Solution: our own tracing mechanisms for IOR and Hadoop

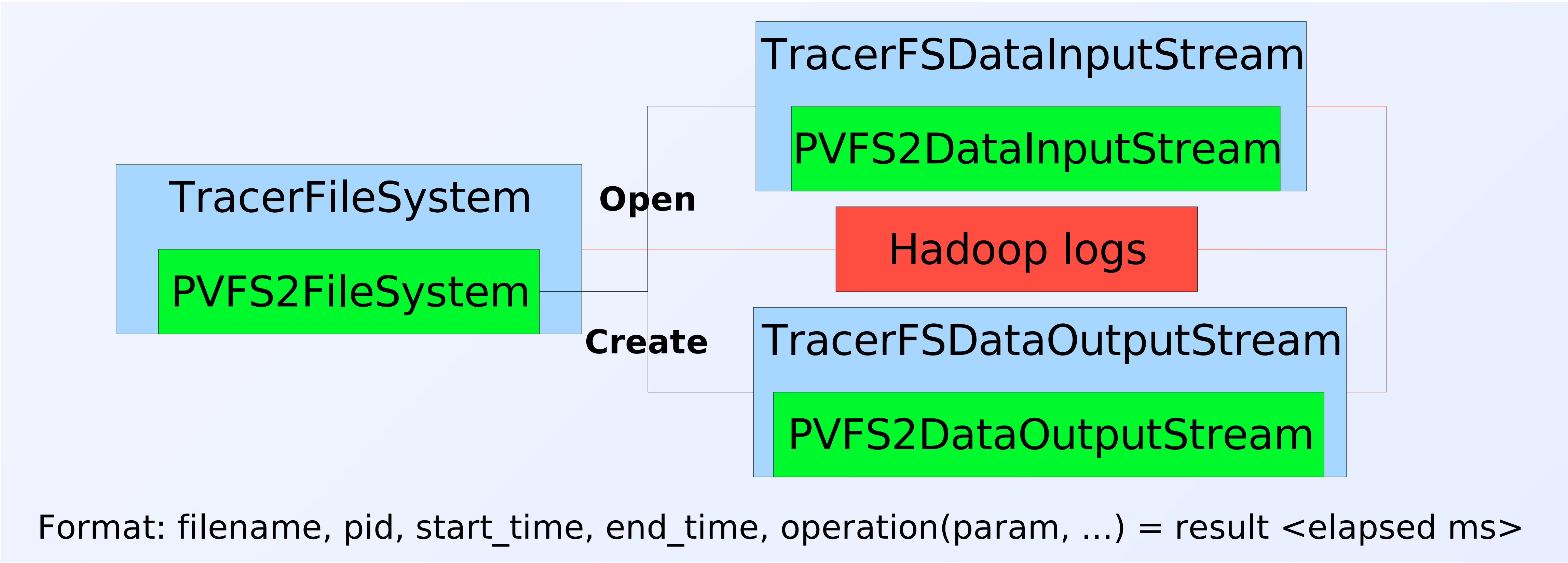
Tracing IOR workloads

- Trace shim intercepts I/O calls, sends to stdio



Tracing Hadoop

- Tracing shim wraps filesystem interfaces, sends I/O calls to Hadoop logs



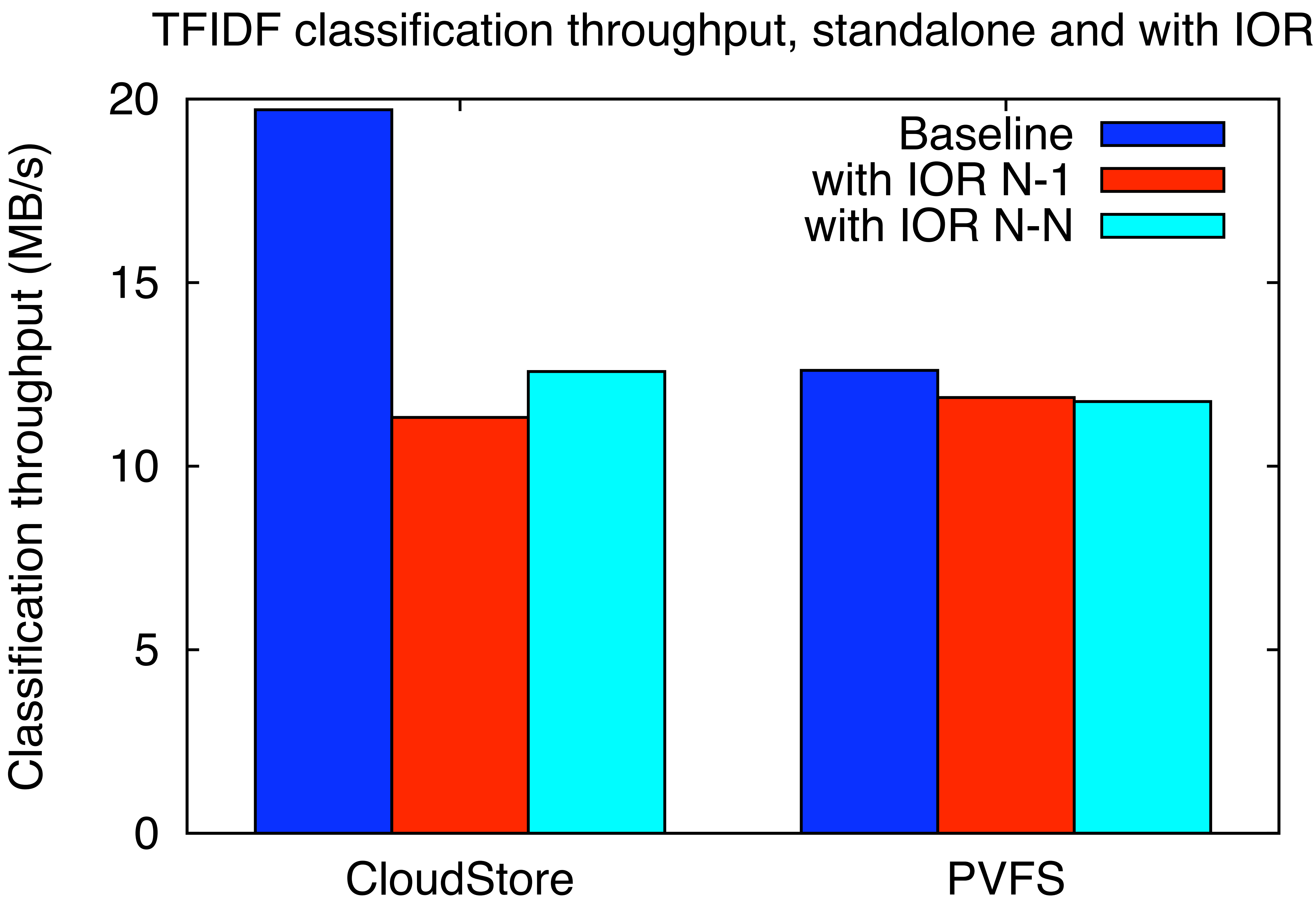
Experimental Setup

- System: 19 nodes, 2-core 2.4 GHz Xeon, 120 GB disks
- IOR baseline: N-1 strided workload, 64 MB chunks
- IOR baseline: N-N workload, 64 MB chunks
- TFIDF baseline: classify 7.2 GB of HTTP headers
- Mixed workloads:
  - IOR N-1 and TFIDF, IOR N-N and TFIDF
  - Checkpoint size adjusted to make IOR and TFIDF take the same amount of time

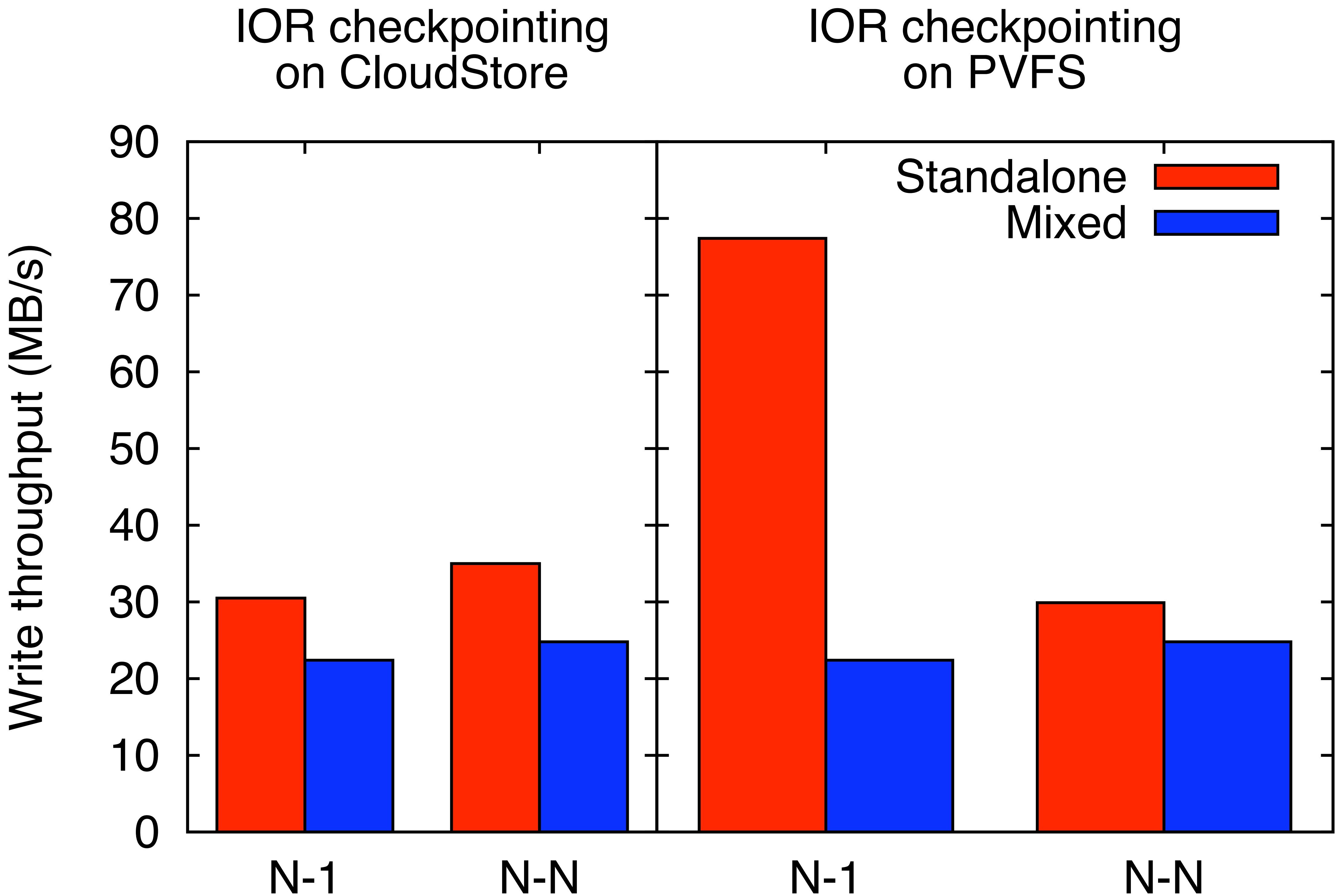
Naive performance predictions

- Each workload will perform better on its native filesystem
- Each workload will be slowed down considerably in the mixed experiments

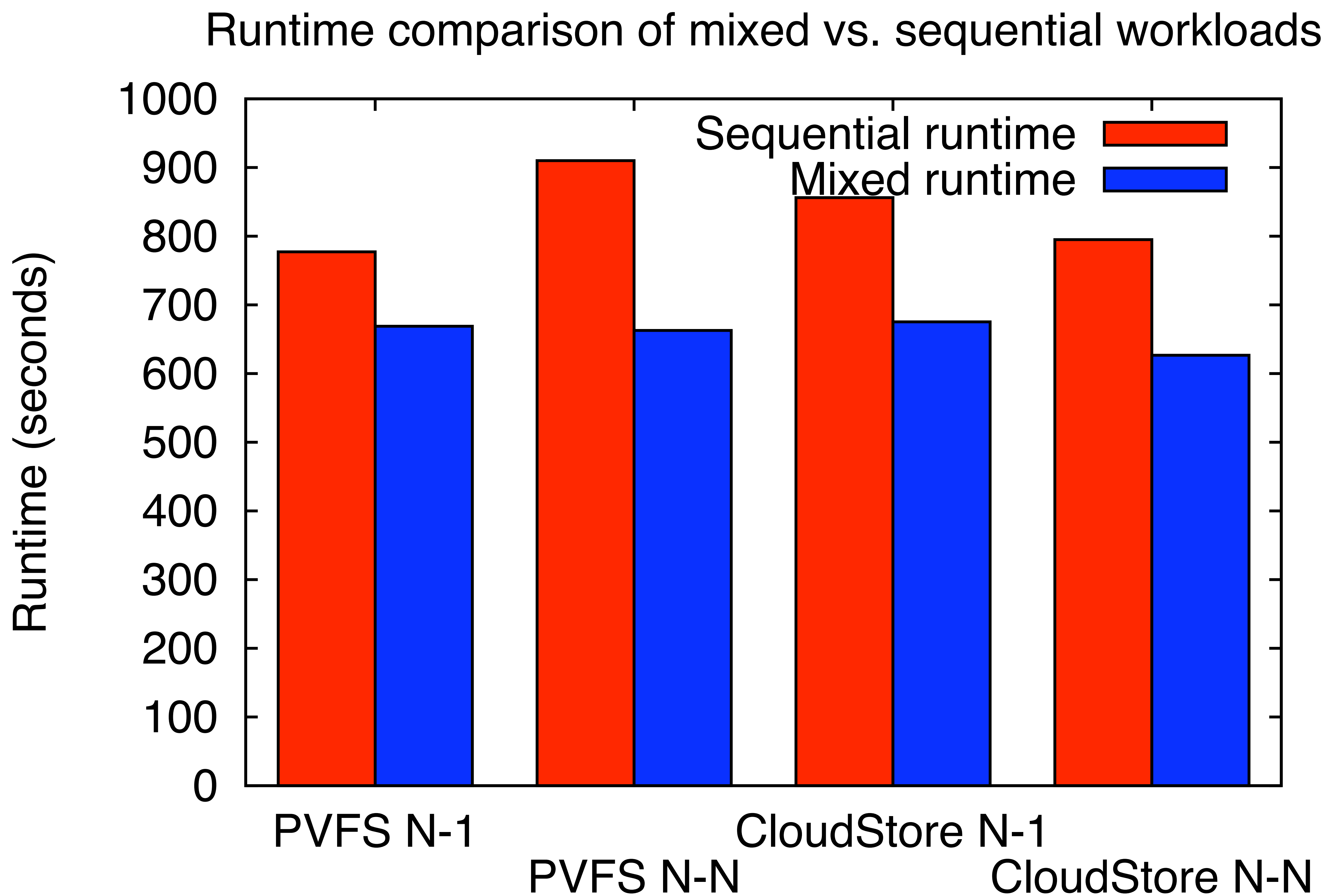
Experimental results



Experimental results



Experimental Results



Conclusions

- Developed I/O tracing mechanisms for IOR benchmarks and Hadoop MapReduce
- Analyzed performance of mixed MapReduce and HPC benchmarking workloads on PVFS and CloudStore
  - TFIDF on PVFS is barely slowed down by IOR
  - All other mixed workloads significantly slowed
  - If only total elapsed time matters, the mixed workloads are faster
- Future work: use experimental results to improve parallel filesystem simulator

Questions?