On the Accuracy and Scalability of Intensive I/O Workload Replay

Alireza Haghdoost, Weiping He, Jerry Fredin*, David H.C. Du

*University of Minnesota, *NetApp Inc.*

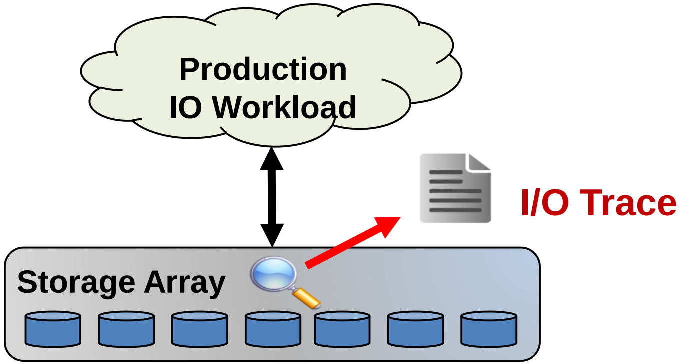
FAST '17

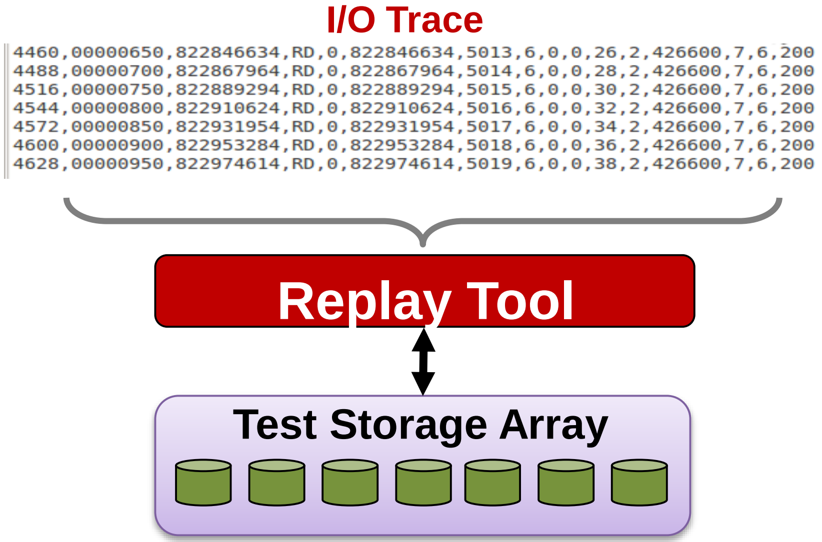
Center for Research in Intelligent Storage

University of Minnesota

What is IO Workload Replay ?

- Storage performance is heavily dependent on the workload
- Replay & Performance evaluation with real workloads
  - Does not require a real infrastructure to produce real workload
  - Does not expose a production systems to potential downtime risk





What are the Challenges?

- Replay Intensive IO workload with High Accuracy is challenging
  - Intensive Workload? (Give me example please….)

More than **100K** IOPS

Less than **2ms** Response Time

IO Workload for Enterprise Block Storage Arrays (SAN)

What are the Challenges?

- Replay Intensive IO workload with High Accuracy is challenging
  - Intensive Workload?
  - What Does High Accuracy Mean?
    1. Replay on Similar Storage
    2. Replay on Faster Storage

What Does High Accuracy Mean?

Replay on Similar storage: Reproduce the workload with the same

Throughput

Response time

IO Request Ordering

What Does High Accuracy Mean?

Replay on Faster storage: Emulate original application behavior

Replay with the same throughput and response-time that original app would do on the faster storage

Throughput

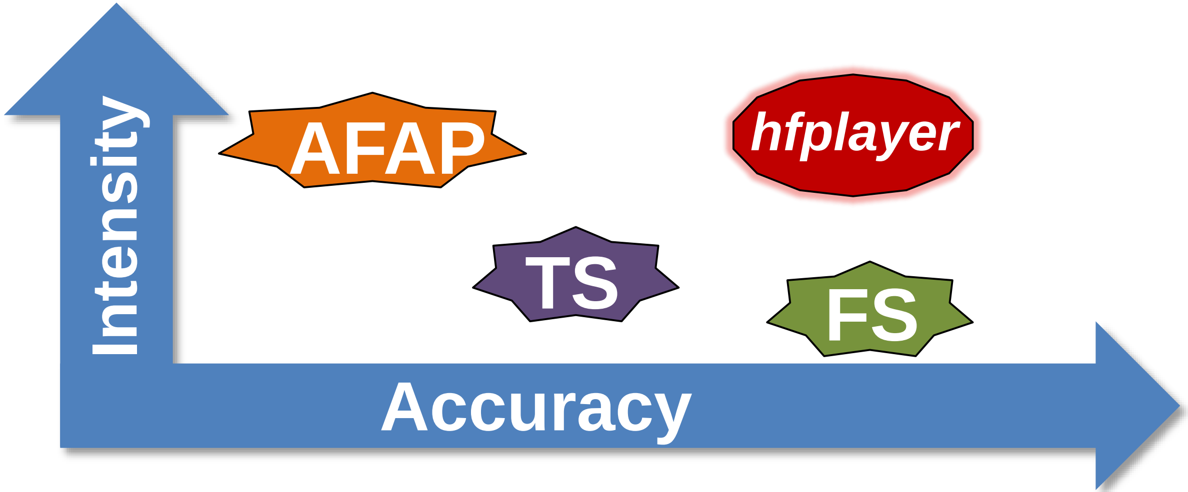
Response time

Existing Replay Approaches

1. Timestamp Based
   - Issue IO request in order based on their relative timestamp
   - Used to Replay on Similar Storage
   - Not capable to scale IO workload
2. As Fast As Possible
   - Issue IO request as fast as possible
   - Used to Replay on Faster storage
   - Not accurate, overlook computation and wait time
   - Ignore dependencies between IO requests
3. FileSystem or Application Trace Replay
   - IO dependency information is available
   - Does not scale well because of filesystem and application layers overhead

Existing Replay Approaches

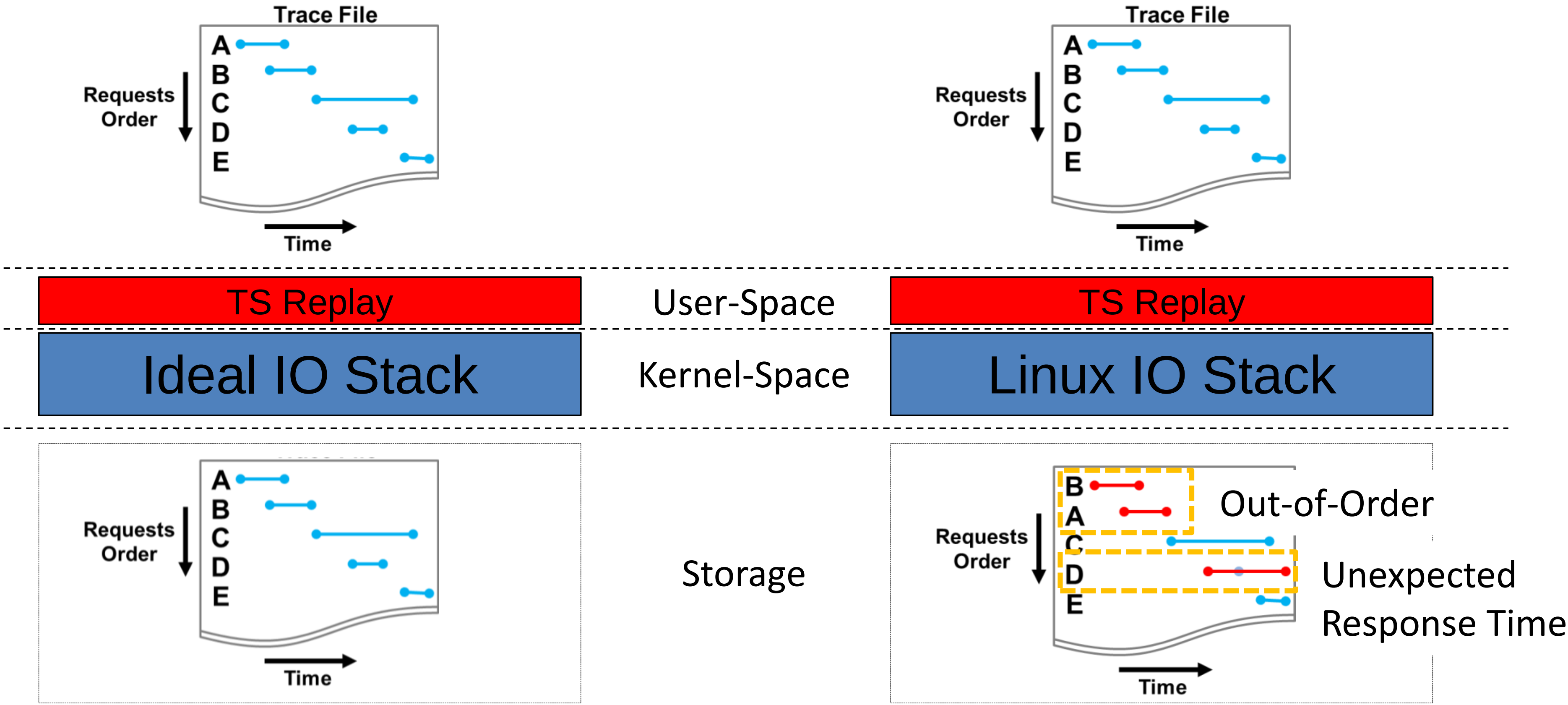
1. Timestamp Based (TS)
2. As Fast As Possible (AFAP)
3. FileSystem or Application Trace Replay (FS)



Agenda

- Motivation
- Replay on Similar Storage
- Replay on Faster Storage
- Evaluation

Unscaled Replay



Sources of Workload Replay Uncertainty

- Forced preemption of system call threads
- User space to Kernel space context switch
- Limited queuing in the IO stack
- Lack of inflight IO control

ftrace therapy

Sources of Workload Replay Uncertainty

☞ Forced preemption of system call threads

- User space to Kernel space context switch
- Limited queuing in the IO stack
- Lack of inflight IO control

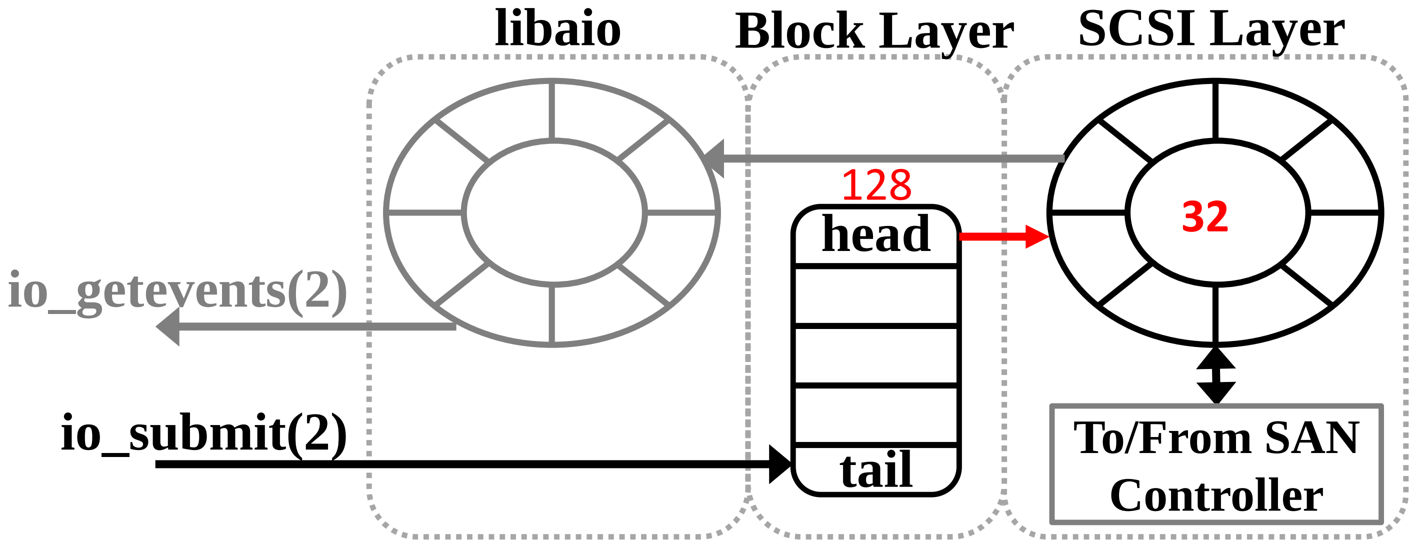
Sources of Workload Replay Uncertainty

- Forced preemption of system call threads
- User space to Kernel space context switch
- Limited queuing in the IO stack
- Lack of inflight IO control

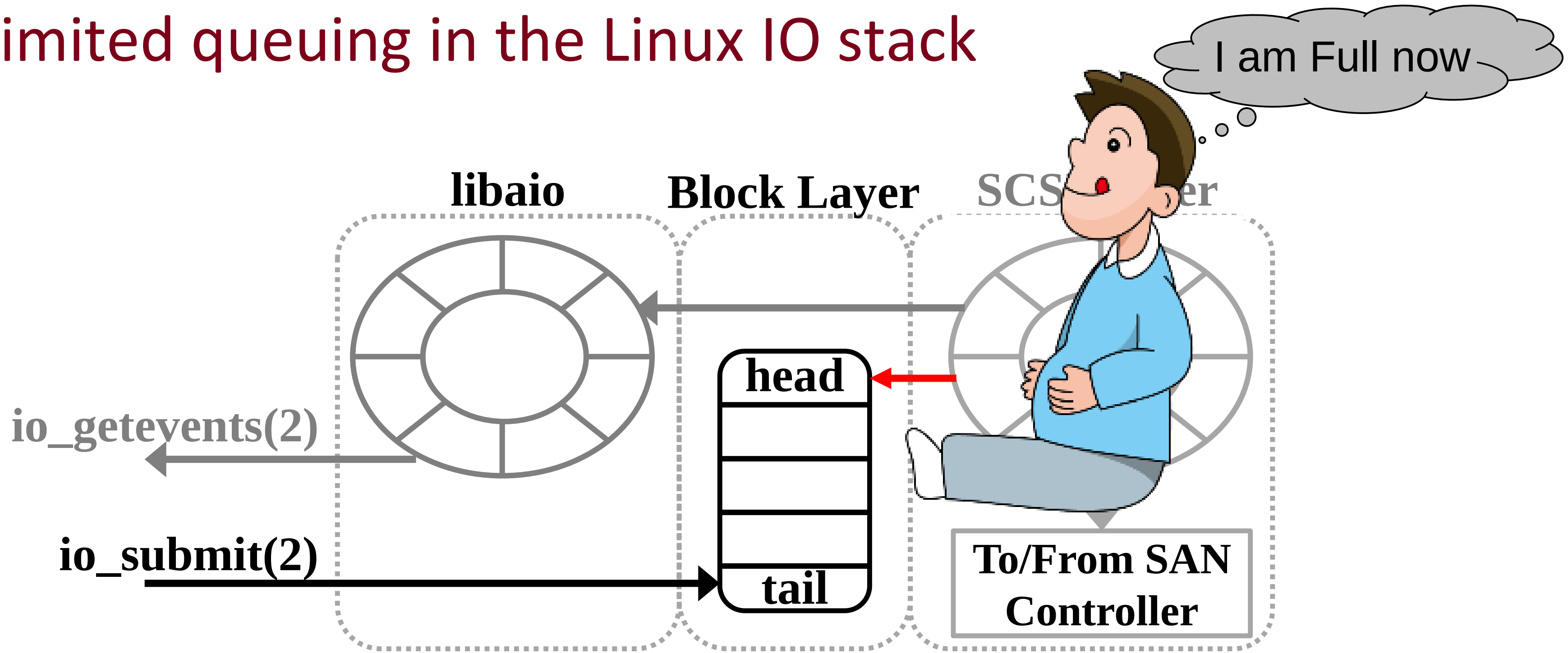
Sources of Workload Replay Uncertainty

- Forced preemption of system call threads
- User space to Kernel space context switch
- ☞ Limited queuing in the IO stack
- ☞ Lack of inflight IO control

Limited queuing in the Linux IO stack



Limited queuing in the Linux IO stack



Scheduling a Kernel Worker thread is expensive &
take unpredictable time &
Changes replayed IO ordering

Any Remedies?

Block Layer queue size

=

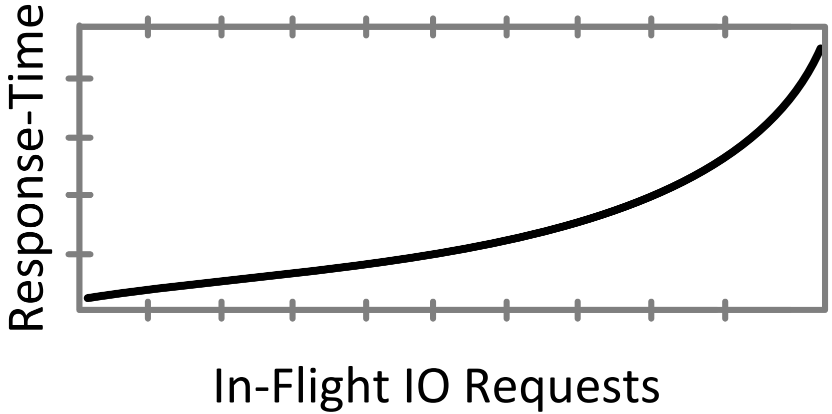
SCSI Layer queue size

=

Max Queue Depth that HBA support

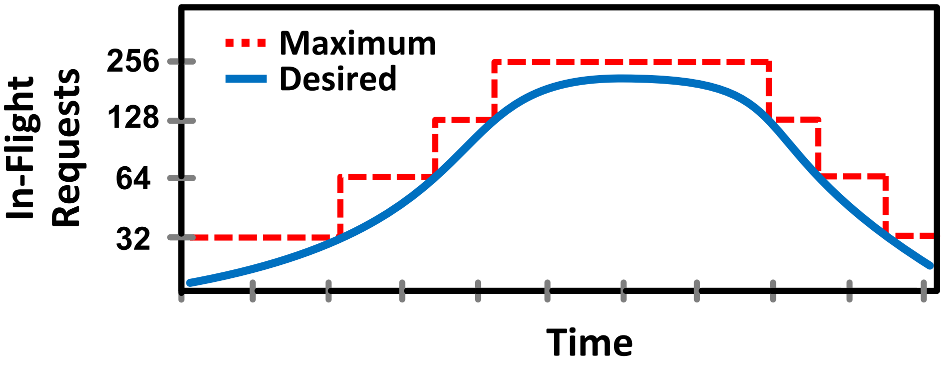
Any Side-effects ?

- The more in-flight requests sitting in the queue:
  - Exponential increase of response time



Any Remedies for Side-effects ?

- Dynamic In-Flight I/O Control implemented in the replayer
- Calculate desired number of in-flight requests
- Throttle the replay speed by bounding the number of in-flight request at run time

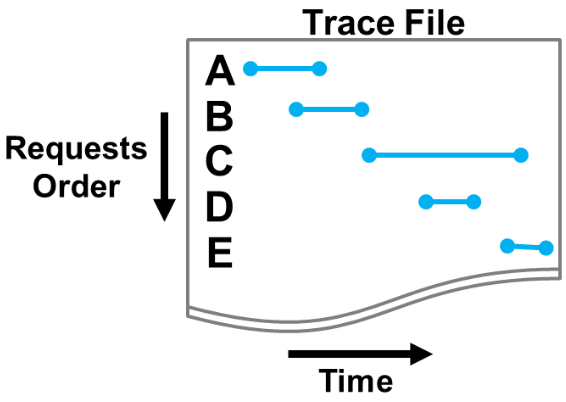


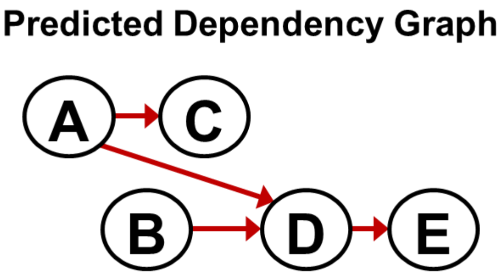
Agenda

- Motivation
- Replay on Similar Storage
- Replay on Faster Storage
- Evaluation

Inferring Potential Dependencies

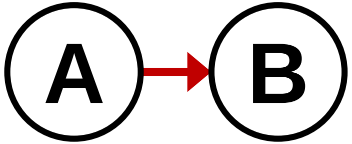
1. Assume all consecutive IO requests are dependent
2. Identify independent requests using their timing relationship
   - Requests with overlapped timing (e.g. A and B)
   - Requests with short think time (e.g. B and C)
3. Remove redundant dependencies

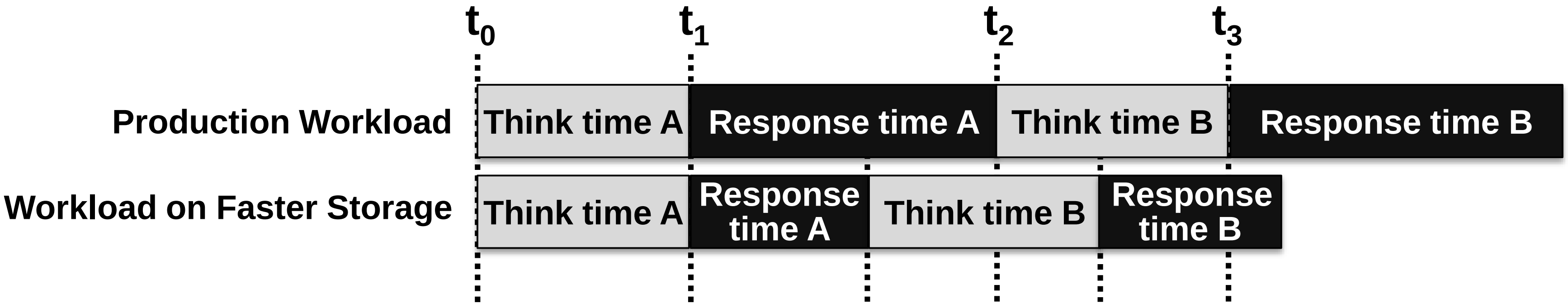




Replay on Scaled Storage

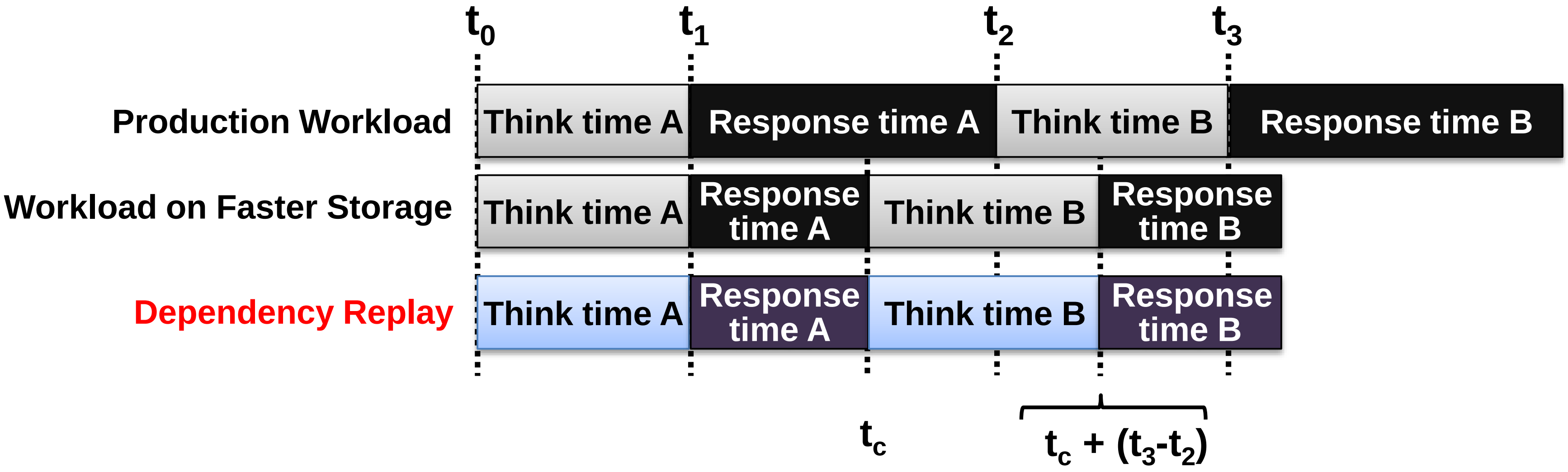
- Example:
  - IO request B is dependent to A





Replay on Scaled Storage

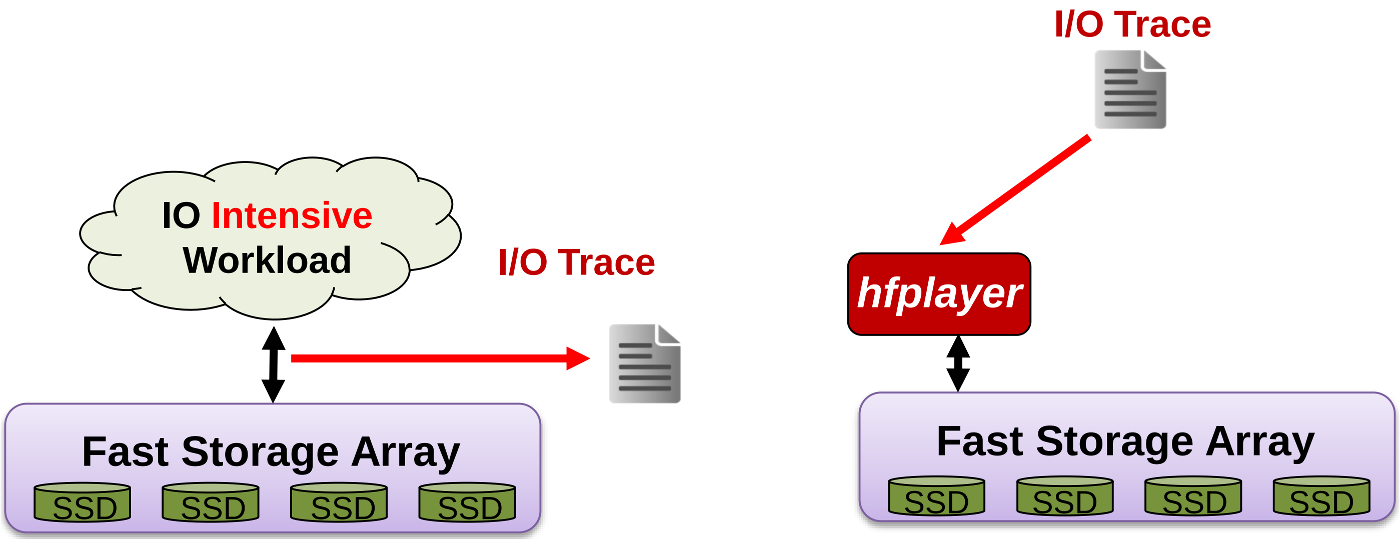
- Emulates production workload on scaled storage
- preserve think time and dynamically adjust issue time



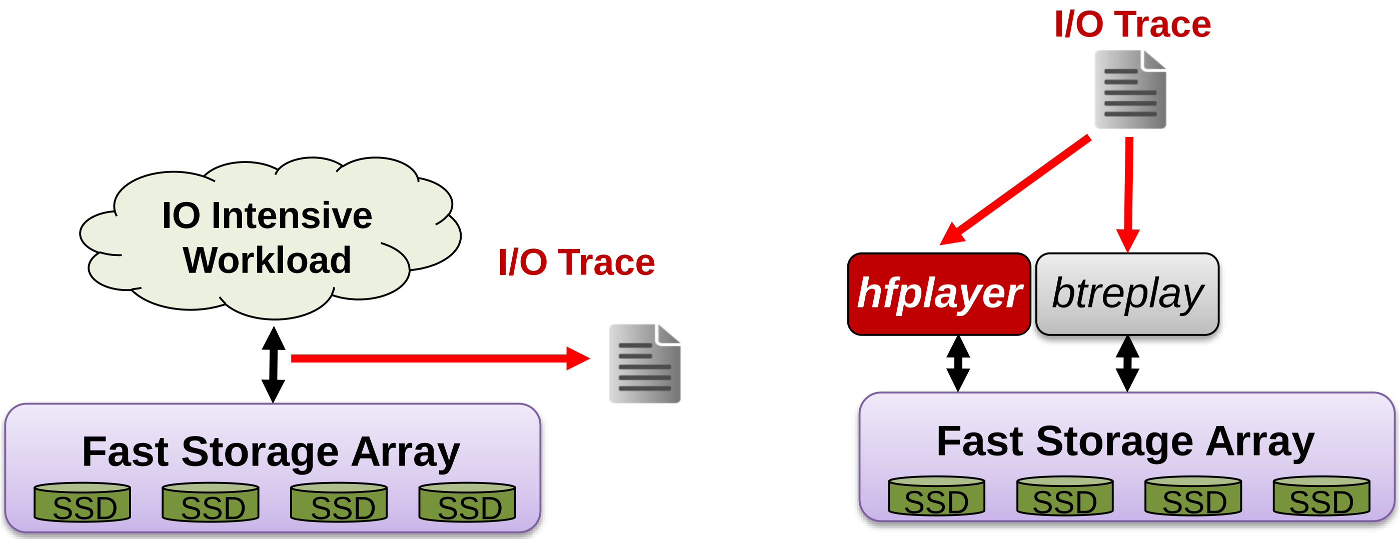
Agenda

- Motivation
- Replay on Similar Storage
- Replay on Faster Storage
- Evaluation

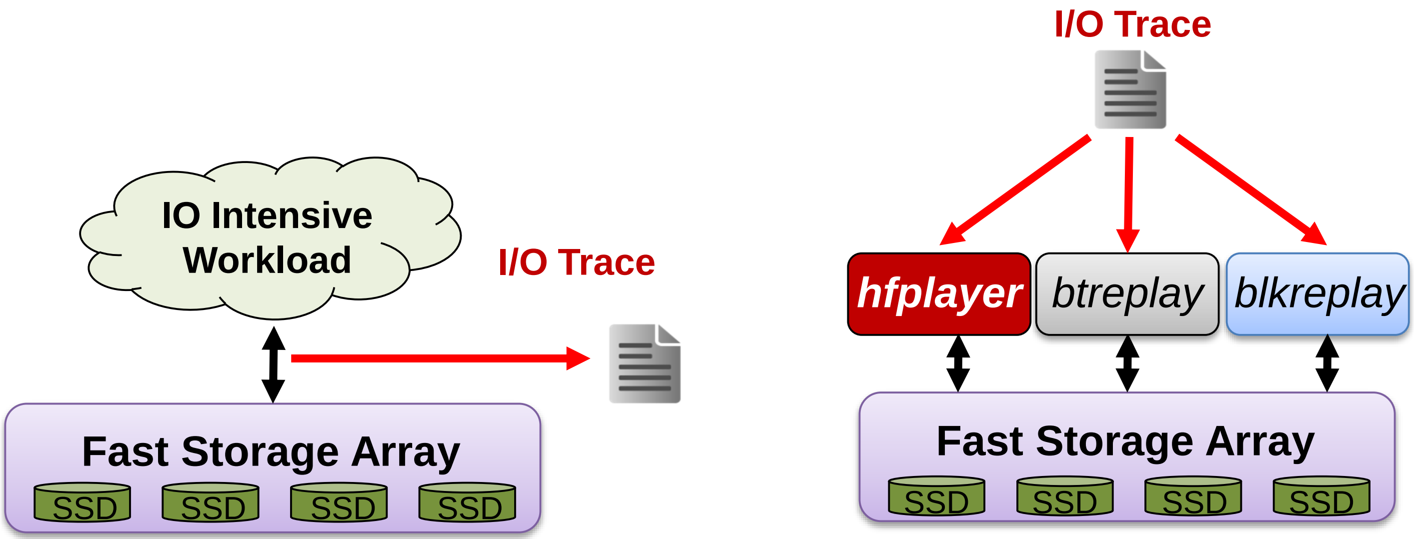
Evaluation Methodology on Unscaled Storage

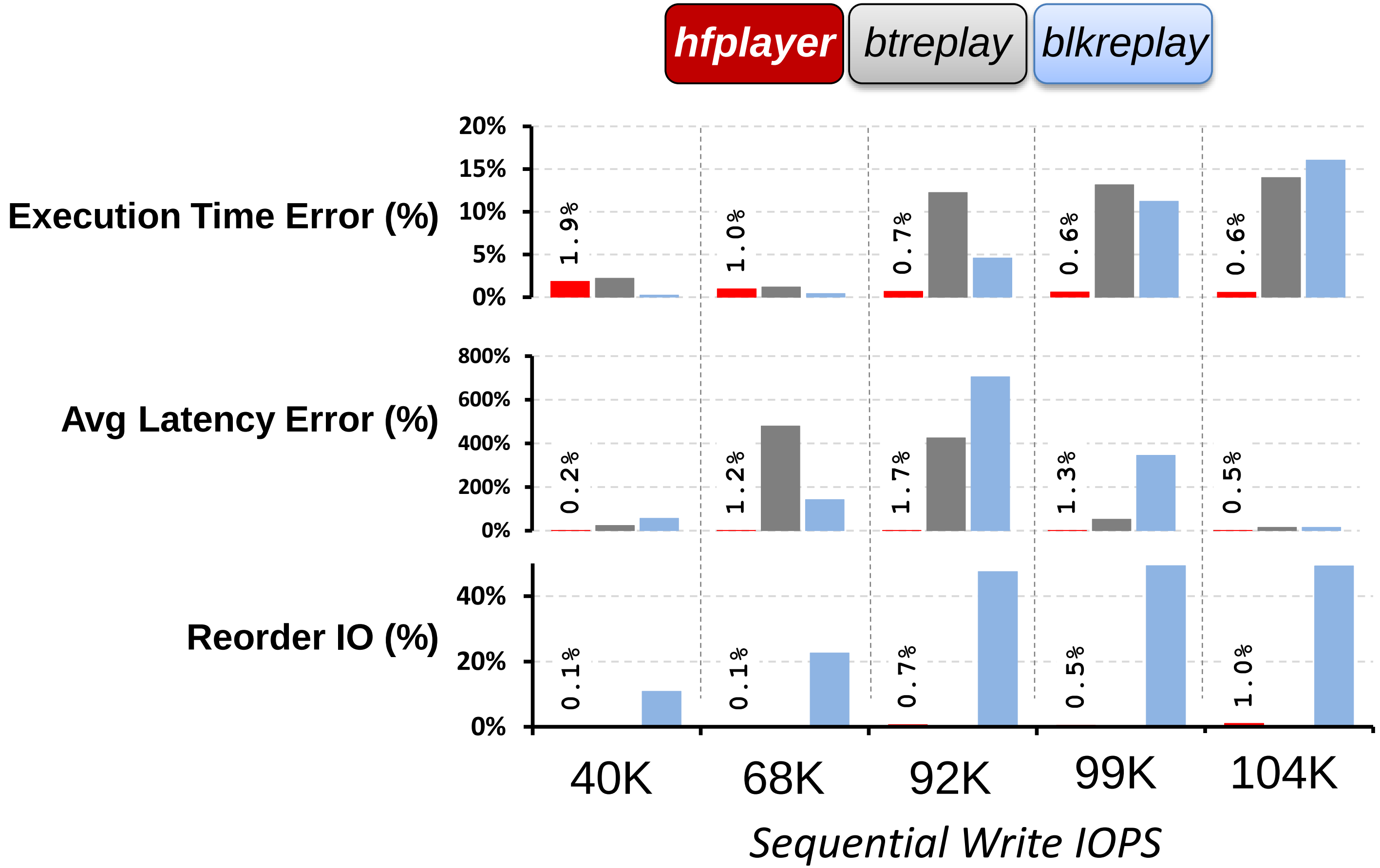


Evaluation Methodology on Unscaled Storage

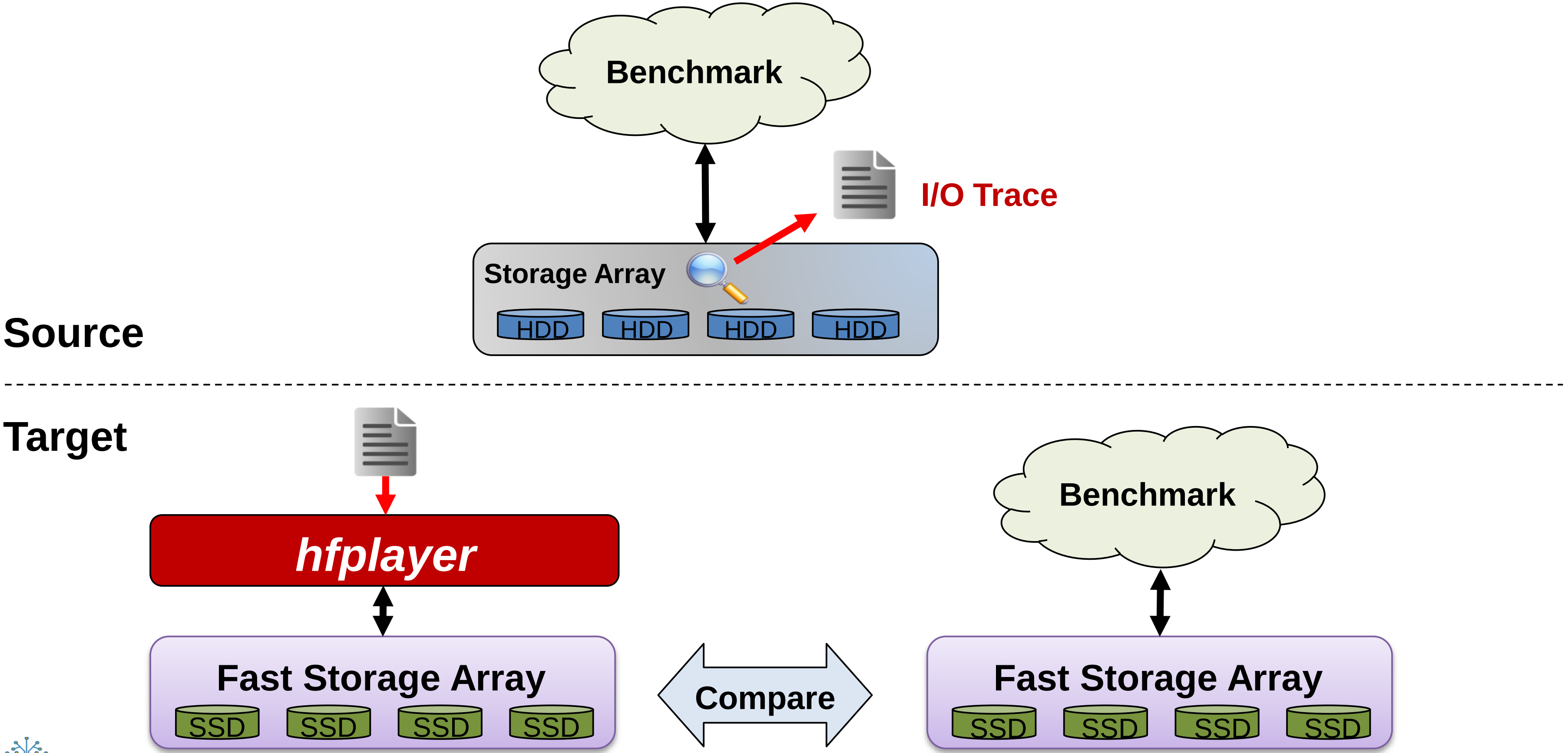


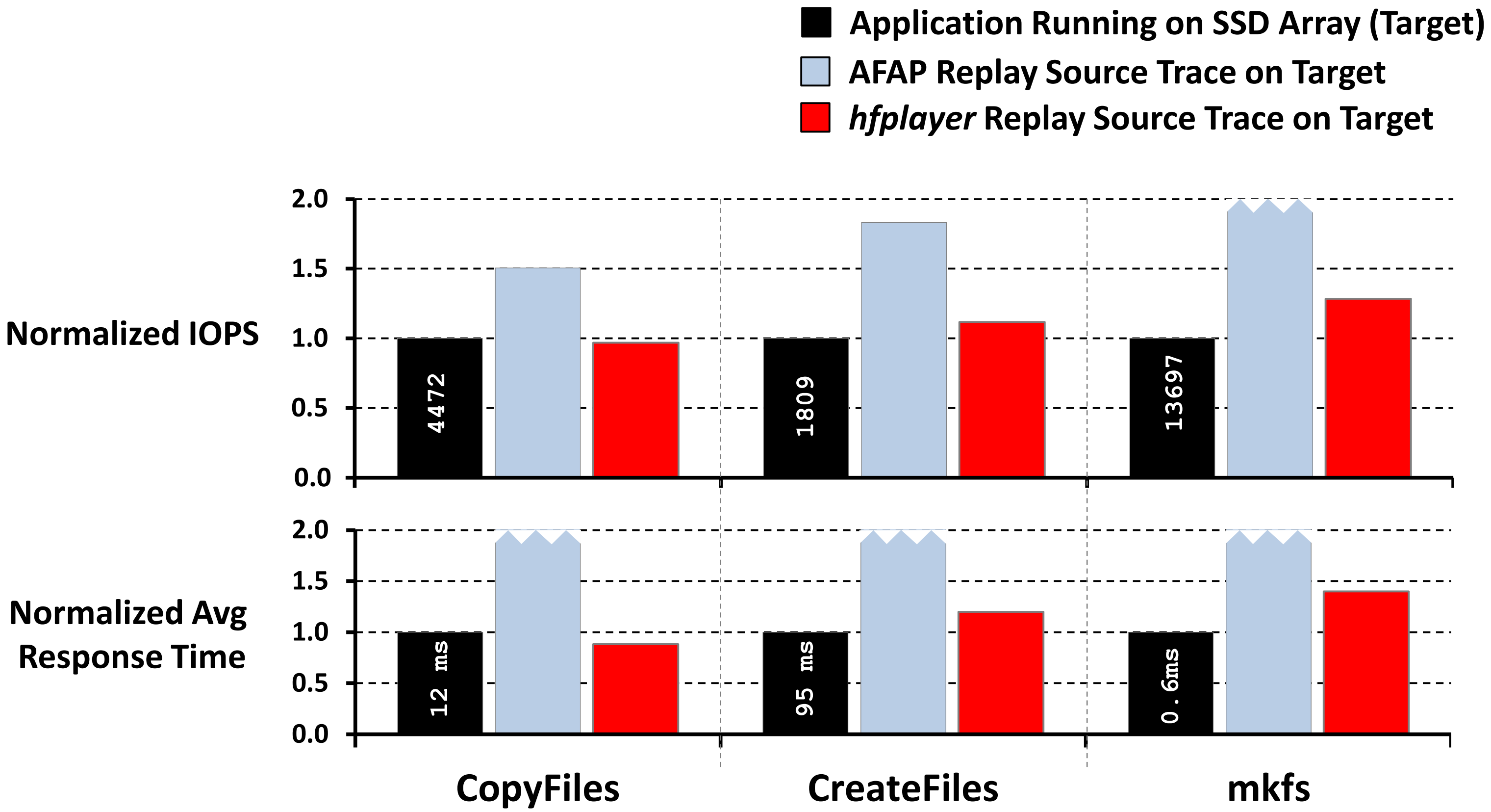
Evaluation Methodology on Unscaled Storage




hfplayer
btreplay
blkreplay
Execution Time Error (%)
20%
15%
10%
5%
0%
1.9%
1.0%
0.7%
0.6%
0.6%
Avg Latency Error (%)
800%
600%
400%
200%
0%
0.2%
1.2%
1.7%
1.3%
0.5%
Reorder IO (%)
40%
20%
0%
0.1%
0.1%
0.7%
0.5%
1.0%
40K
68K
92K
99K
104K
Sequential Write IOPS

Evaluation Methodology on Scaled Storage




Application Running on SSD Array (Target)
AFAP Replay Source Trace on Target
hfplayer Replay Source Trace on Target
Normalized IOPS
2.0
1.5
1.0
0.5
0.0
4472
1809
13697
Normalized Avg Response Time
12 ms
95 ms
0.6ms
CopyFiles
CreateFiles
mkfs

Conclusions

- Provide detailed analysis of replay uncertainty sources
- Introduced new methods to infer IO decencies from block IO trace
- Introduced new replay methods with more accuracy on both scaled and unscaled storage
- *hfplayer* is open source and available in GitHub

http://GitHub.com/UMN-CRIS

Questions

Alireza Haghdoost, alireza@cs.umn.edu

http://GitHub.com/UMN-CRIS

CRIS
Center for Research in Intelligent Storage

University of Minnesota

IO Workload Trace

- Sample Block IO Trace

```
5104,7887.23,16826100,READ,30,7380094846374360,435166255,6,1,0,264,8,5473136,7,6,2.53
5132,7887.79,16827288,READ,31,7380094846375548,435166256,6,1,0,272,8,5438348,7,6,2.22
5160,7888.33,16828444,READ,32,7380094846376704,435166257,6,1,0,280,8,5471428,7,6,2.73
5188,7888.89,16829640,READ,33,7380094846377900,435166258,6,1,0,288,8,5436700,7,6,2.45
5216,8889.43,16830800,WRITE,0,7380094846379060,435166259,7,1,0,296,8,5470904,7,6,2.48
5244,8889.93,16831864,WRITE,1,7380094846380124,435166260,7,1,0,304,8,5438088,7,6,2.10
5272,8890.44,16832956,WRITE,2,7380094846381216,435166261,7,1,0,312,8,5470144,7,6,2.13
5300,8890.95,16834044,WRTIE,3,7380094846382304,435166262,7,1,0,320,8,5448352,7,6,2.91
```

Dependent or Independent? That is the question.

(Fake) Hamlet

hfplayer Internal Architecture

- Dependency analyzer module
- Worker threads issue IO requests
  - According to their scheduled timestamp in unscaled replay mode
  - After all of its parents plus their corresponding think time in scaled replay mode