Modeling and Replay of Storage I/O for Datacenter Workloads

Christina Delimitrou[1], Sriram Sankar[2], Kushagra Vaid[2], Christos Kozyrakis[1]
[1] Stanford University, [2] Microsoft

Introduction

Workload **Modeling** and **Generation** is important because:

- Replay of original application in all storage system configurations is **impractical**
- Datacenter Workloads are **not publicly available**
- **Storage System ~ 20-30% of TCO and power consumption** of the total system

GOAL: Design a tool that recreates representative datacenter I/O workloads with high fidelity

APPLICABILITY: SSD Caching, Defragmentation Benefits, Storage Consolidation, …

NOTE: Generation of the I/O access patterns NOT the application's functionality

Two Step Approach

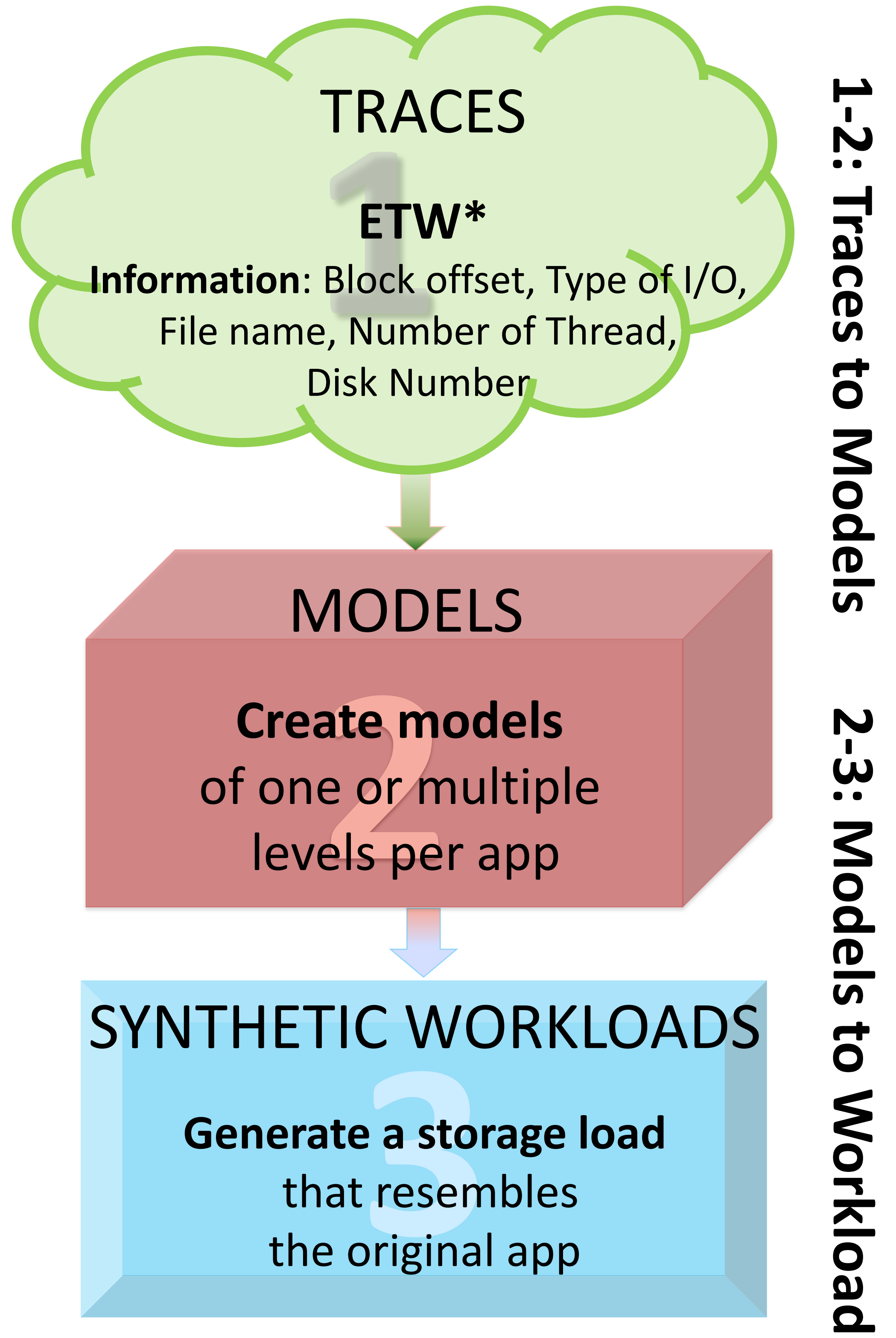

TRACES
1
ETW*
Information: Block offset, Type of I/O, File name, Number of Thread, Disk Number
MODELS
2
Create models of one or multiple levels per app
SYNTHETIC WORKLOADS
3
Generate a storage load that resembles the original app
1-2: Traces to Models
2-3: Models to Workloads

Figure 1: Two Step Modeling-Generation Approach

*Event Tracing for Windows

Model

State Diagram-Based Probabilistic Model:

- **State**: Logical block range on disk
- **Transition**: Probability of switching between block ranges
- **Stats**: rd/wr, rnd/seq, block size, inter-arrival time

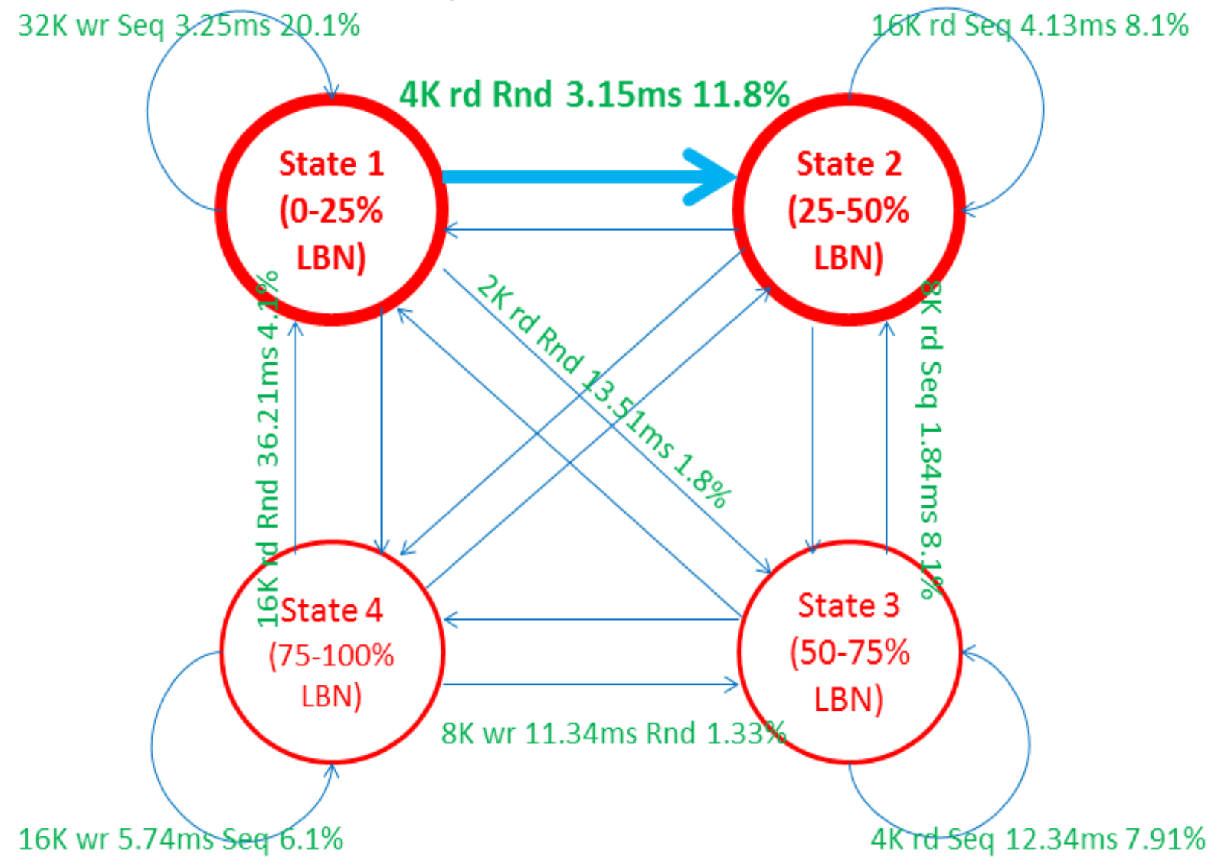


Figure 2: Simple State Diagram (1 level)

Extend the simple, one level model to a hierarchical representation.

Choose an optimal number of levels per application

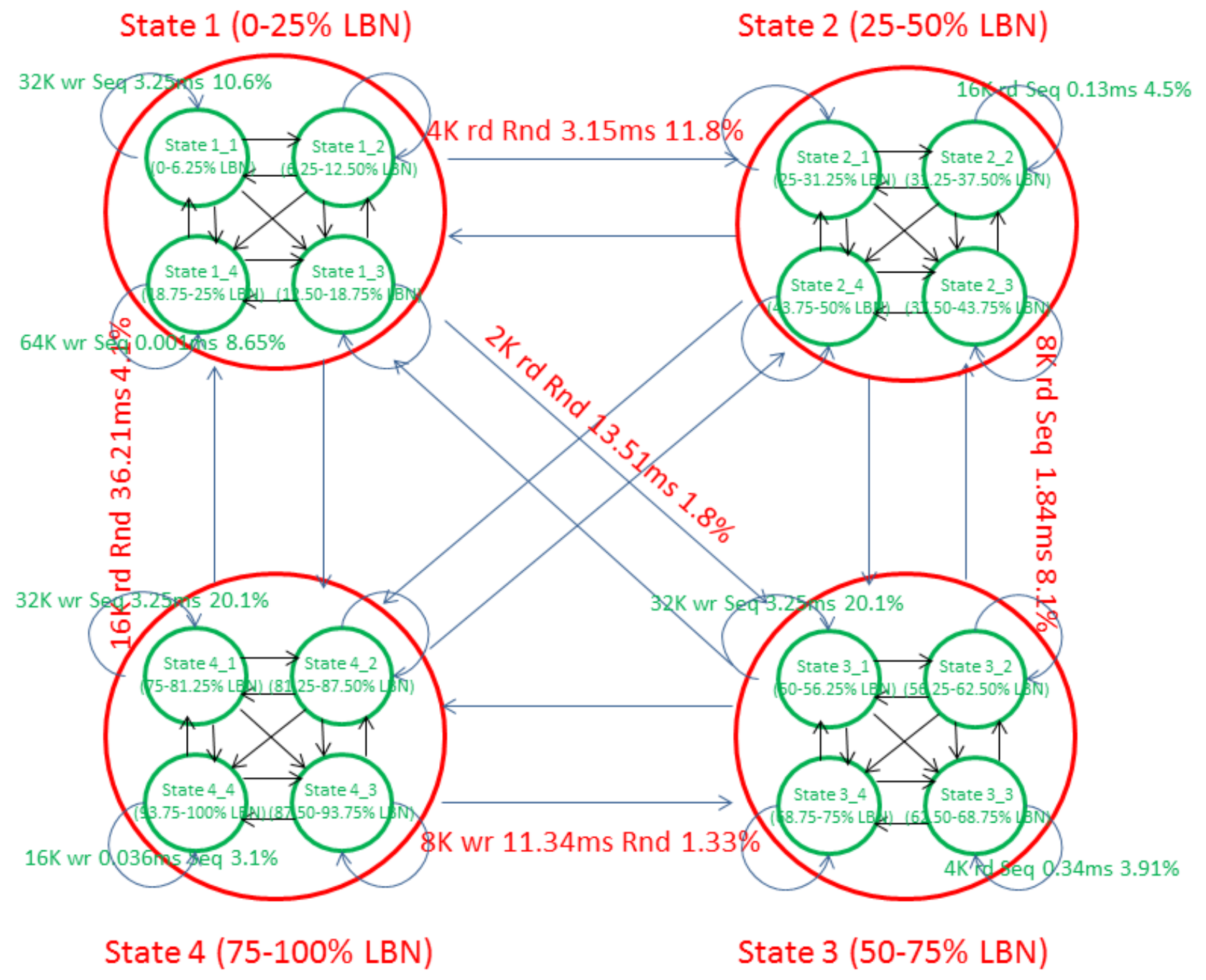

State 1 (0-25% LBN)
State 2 (25-50% LBN)
32K wr Seq 3.25ms 10.6%
16K rd Seq 0.13ms 4.5%
4K rd Rnd 3.15ms 11.8%
State 1_1 (0-6.25% LBN)
State 1_2 (6.25-12.50% LBN)
State 1_4 (18.75-25% LBN)
State 1_3 (12.50-18.75% LBN)
State 2_1 (25-31.25% LBN)
State 2_2 (31.25-37.50% LBN)
State 2_4 (43.75-50% LBN)
State 2_3 (37.50-43.75% LBN)
64K wr Seq 0.001ms 8.65%
16K rd Rnd 36.21ms 4.1%
2K rd Rnd 13.51ms 1.8%
8K rd Seq 1.84ms 8.1%
32K wr Seq 3.25ms 20.1%
32K wr Seq 3.25ms 20.1%
State 4_1 (75-81.25% LBN)
State 4_2 (81.25-87.50% LBN)
State 4_4 (93.75-100% LBN)
State 4_3 (87.50-93.75% LBN)
State 3_1 (50-56.25% LBN)
State 3_2 (56.25-62.50% LBN)
State 3_4 (68.75-75% LBN)
State 3_3 (62.50-68.75% LBN)
8K wr 11.34ms Rnd 1.33%
16K wr 0.036ms Seq 3.1%
4K rd Seq 0.34ms 3.91%
State 4 (75-100% LBN)
State 3 (50-75% LBN)

Figure 3: Hierarchical State Diagram (2 levels)

Reduce Model Complexity: Spatial Locality within a state rather than across states (**Hierarchical** rather than **Flat** representation)

Previous Tools

IOMeter is the most well-known open-source workload generator

Features	IOMeter	DiskSpd
Inter-arrival Time (mean or distribution)	✗	✓
Intensity Knob	✗	✓
Spatial Locality	✗	✓
Trace Replay	✗	✓
Different Levels of Granularity	✗	✓
File Accesses*	✗	✓

Table 1: IOMeter – DiskSpd Comparison

Implementation

1/4 : Inter-Arrival Times

Inter-arrival Time: The time between two subsequent I/O requests.

Inter-arrival Times ≠ Outstanding I/Os

Generating inter-arrival times both **static** and with **time distributions**

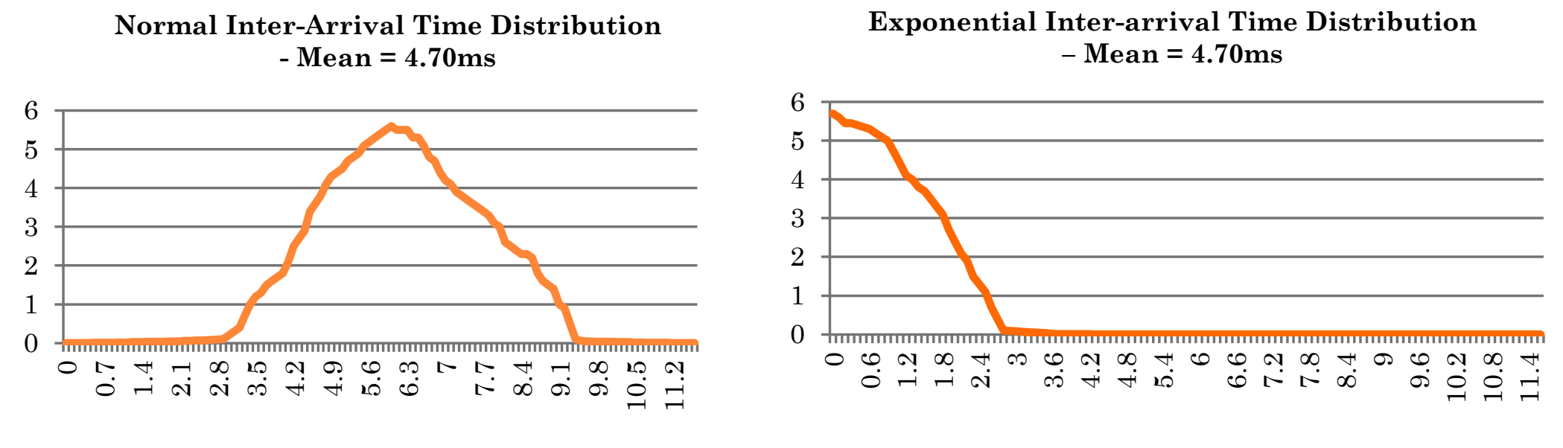


Figure 4: Normal and Exponential Inter-arrival Time Distributions

2/4 : Thread Weights

One thread = One transition in the state diagram

Specific I/O characteristics **per thread.**

Thread Weight: The proportion of accesses for one transition. Thread weights are satisfied with less than 0.05% deviation

3/4 : Intensity Knob

Evaluation of different storage system configurations (Disk vs. SSD)

Scale the inter-arrival times (more or less intense workload) without retuning the application

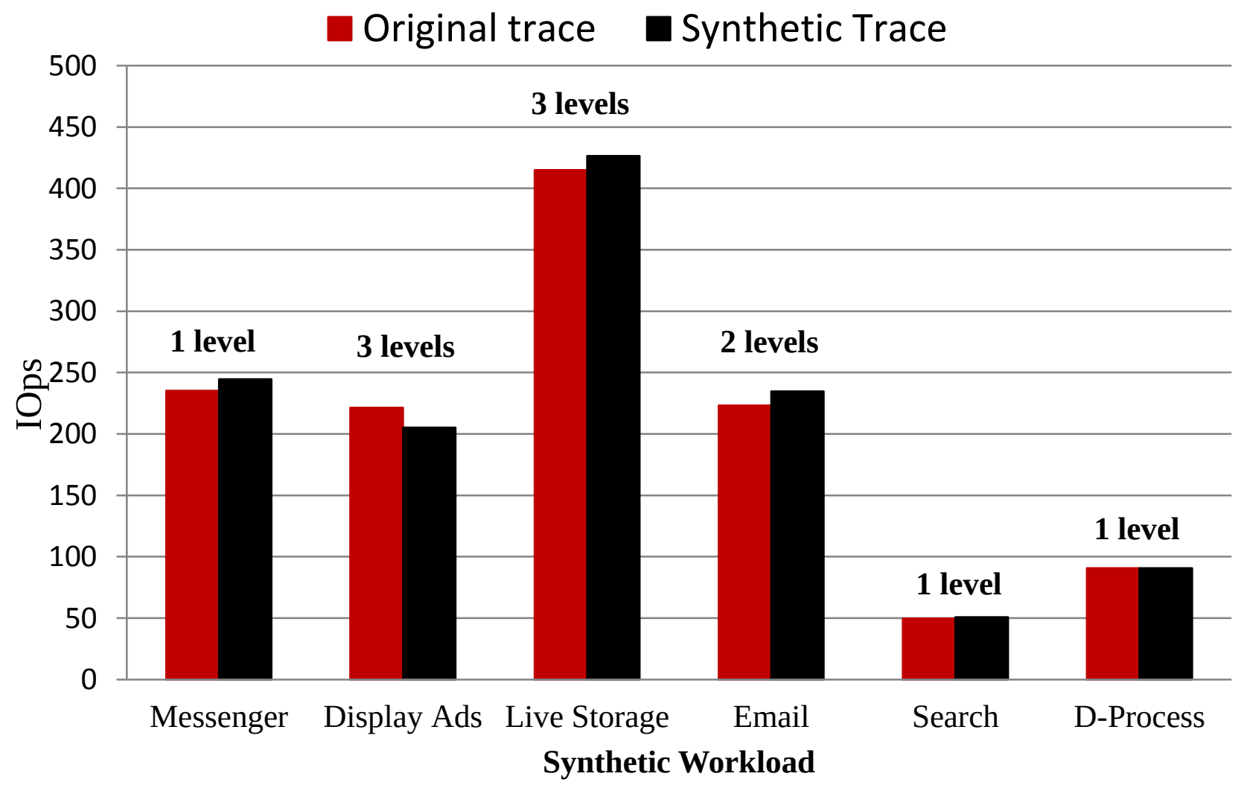
4/4 : Trace Replay

Replication of the exact same I/O request (block offset, type, block size)

Applicability: Error Detection in large-scale DBs.

Model & Tool Validation

- **Collect Traces of Original Applications**
- **Create One/Multiple Level State Diagrams**
- **Compare I/O characteristics and Performance Metrics between Original and Synthetic Traces**



NOTE: In all cases < 1% variance between runs

Figure 5: Validation of Throughput

Applicability

1. SSD Caching

Progressive SSD caching (0-4 SSDs)
Storage I/Os for most applications are very aggregated in space

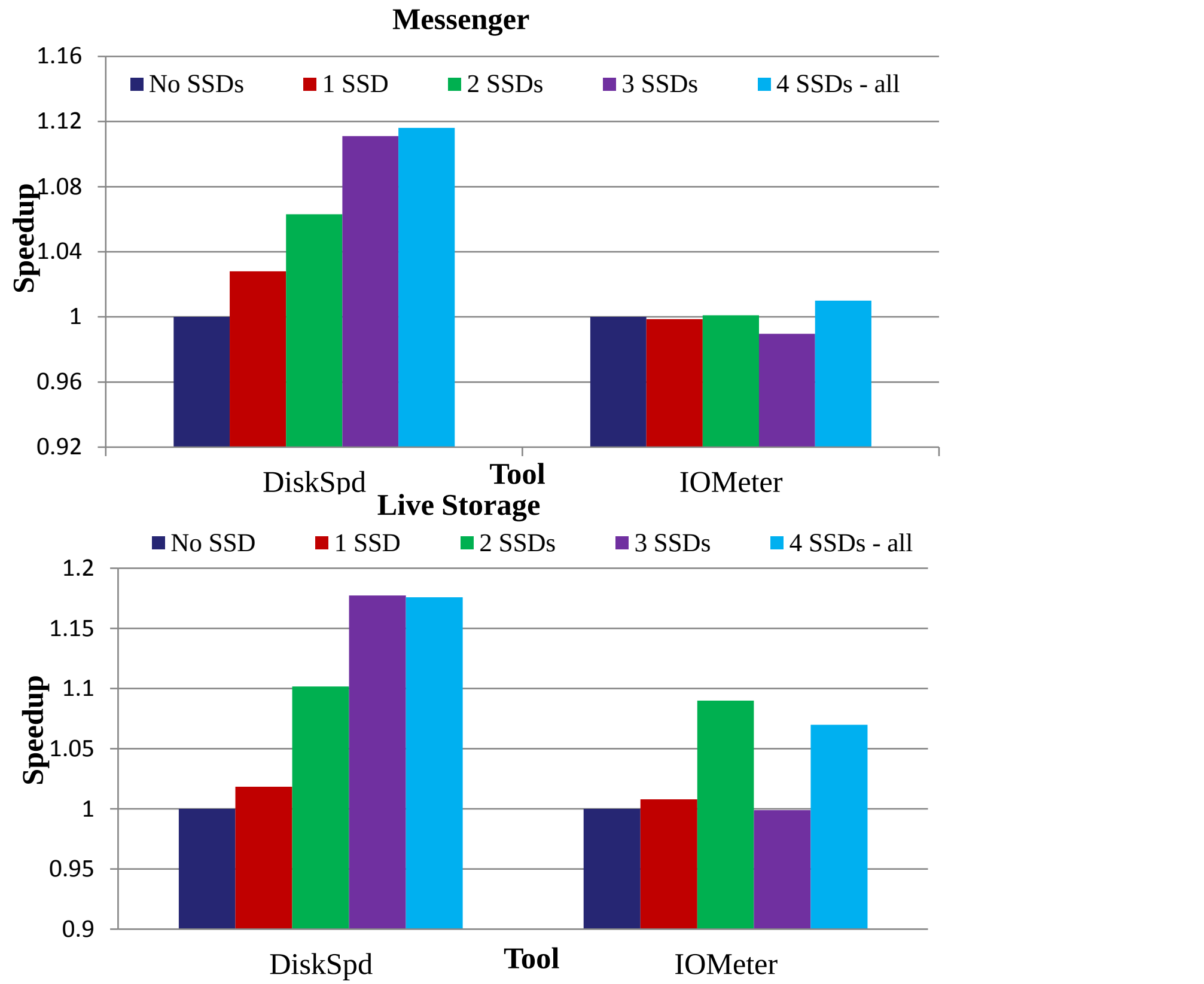


Figure 6: DiskSpd – IOMeter Comparison. Using IOMeter either has NO SPEEDUP (6.a) or INCONSISTENT SPEEDUP (6.b) with increasing number of SSDs

2. Defragmentation Benefits

Random > 80% - Sequential < 20% for most DC applications

Performing Defragmentation during low throughput requirement phases improves performance/efficiency

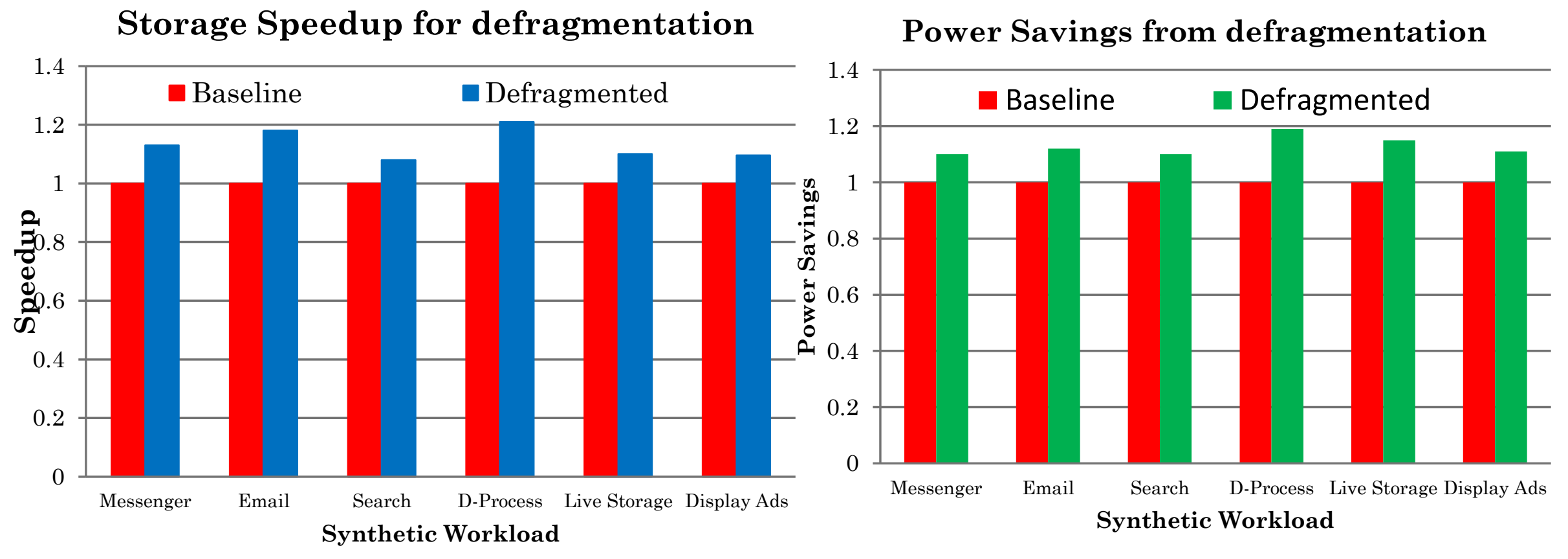


Figure 7: Storage Speedup and Power Savings from Defragmentation

Conclusions and Future Work

- **Model and Generate** representative DC storage I/O loads with **high fidelity** and **density in time**
- **Use the tool** to motivate two important challenges in DC storage system design: **SSD caching and the benefits from Defragmentation without the requirement for access to app code or full application deployment**

FUTURE WORK:

- Evaluate **energy efficiency** for SSD caching and defragmentation
- **Expand a similar methodology to other parts of the system to create a Complete Workload Model** with applications in virtualization, etc.