ACCURATE MODELING & GENERATION OF STORAGE I/O FOR DC WORKLOADS

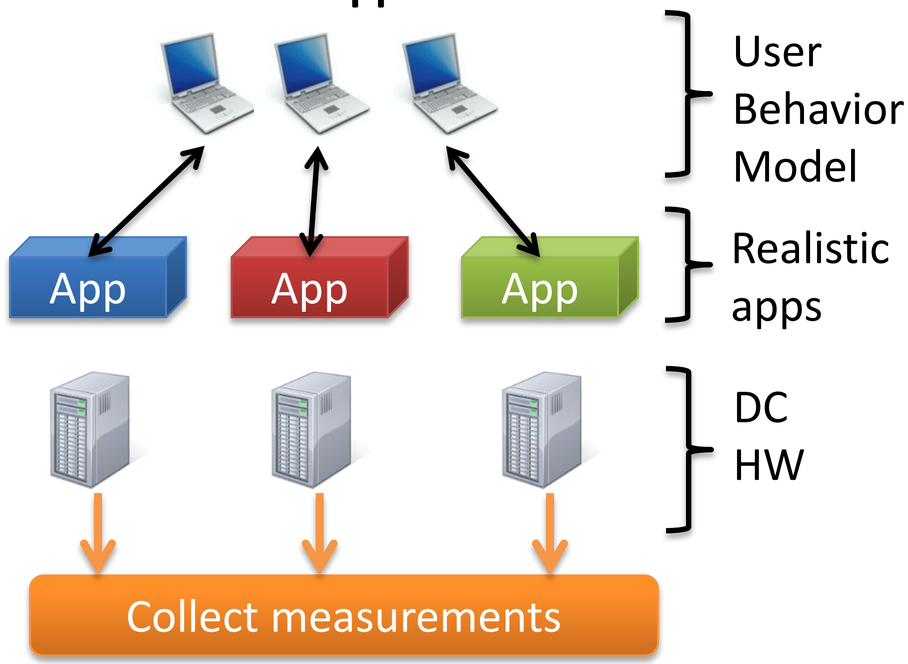
Christina Delimitrou[1], Sriram Sankar[2], Kushagra Vaid[2], Christos Kozyrakis[1]

[1]Stanford University, [2]Microsoft

EXERT– March 06th 2011

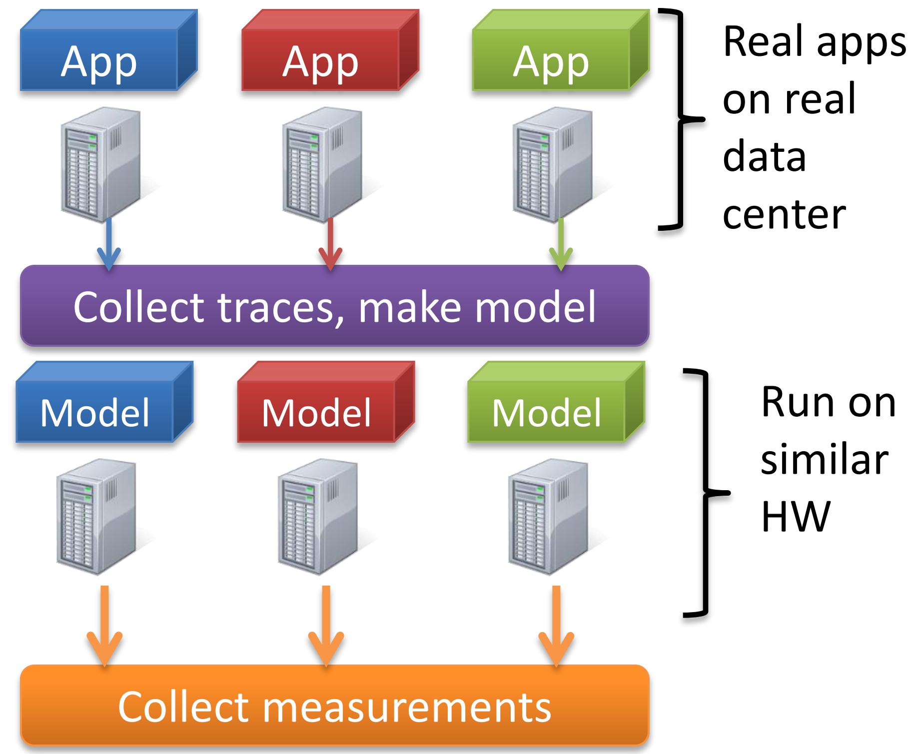
Datacenter Workload Studies

Open-source approximation of real applications



- Pros: Resembles specific real applications
- Pros: Can modify the underlying hardware
- Cons: Requires user behavior models to test
- Cons: Not exact match to real DC applications

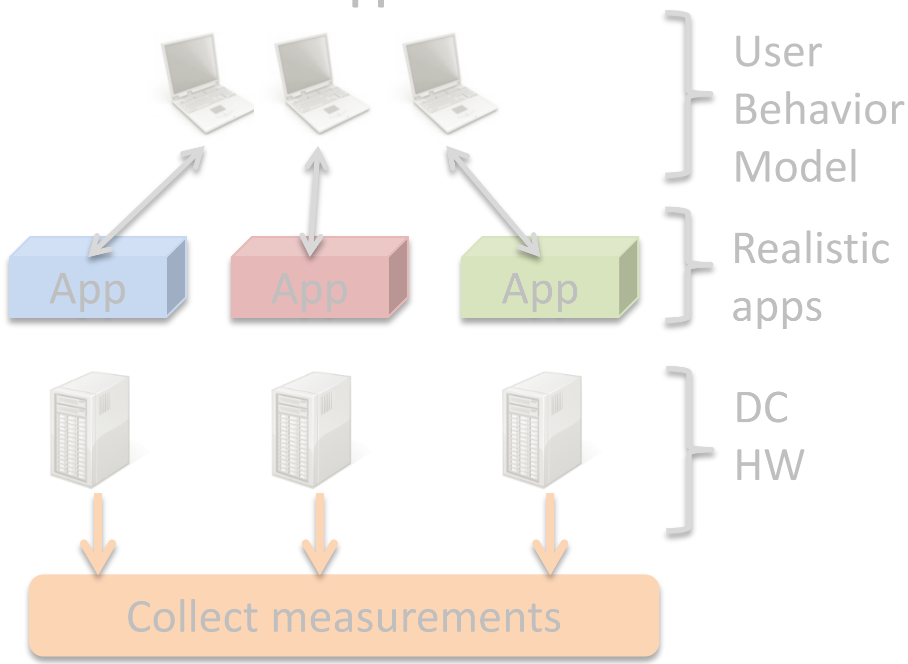
Statistical models of real applications



- Pros: Models of real large scale application – closer resemblance
- Pros: Enables “real” app studies
- Cons: Hardware and Code dependent
- Cons: Many parameters/dependencies to model

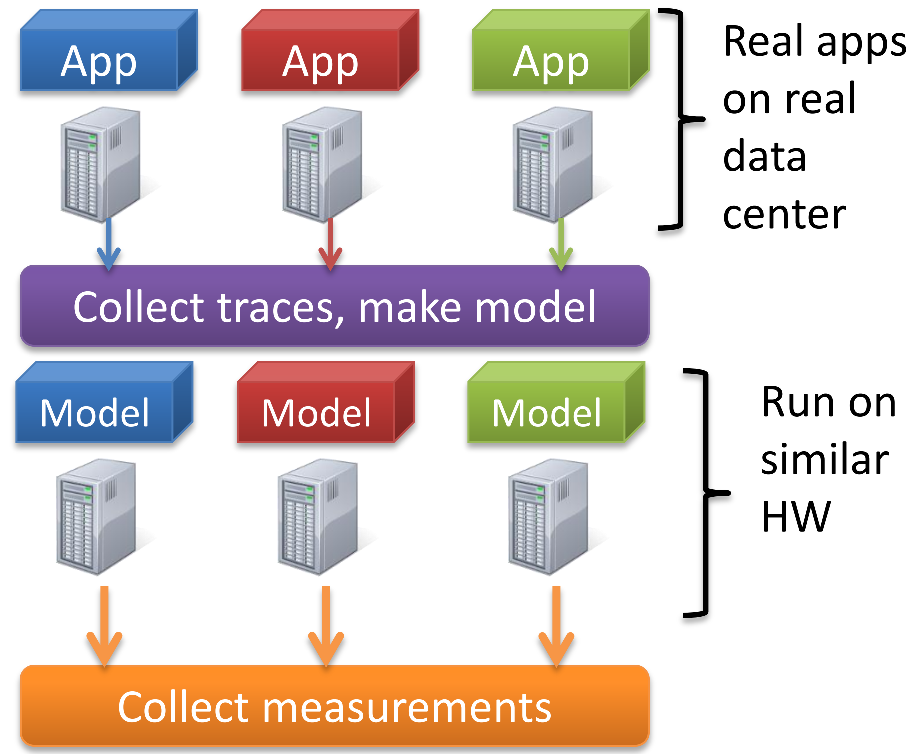
Datacenter Workload Studies

Open-source approximation of real applications



- Pros: Resembles specific real applications
- Pros: Can modify the underlying hardware
- Cons: Requires user behavior models to test
- Cons: Not exact match to real DC applications

Use statistical models of real applications



- Pros: Models of real large scale application – closer resemblance
- Pros: Enables "real" app studies
- Cons: Hardware and Code dependent
- Cons: Many parameters/dependencies to model

OUTLINE

- Introduction/Goals
- Comparison with previous tools
  - IOMeter vs. DiskSpd
- Implementation
- Validation
- Tool Applicability
  - SSD caching
  - Defragmentation Benefits
- Future Work

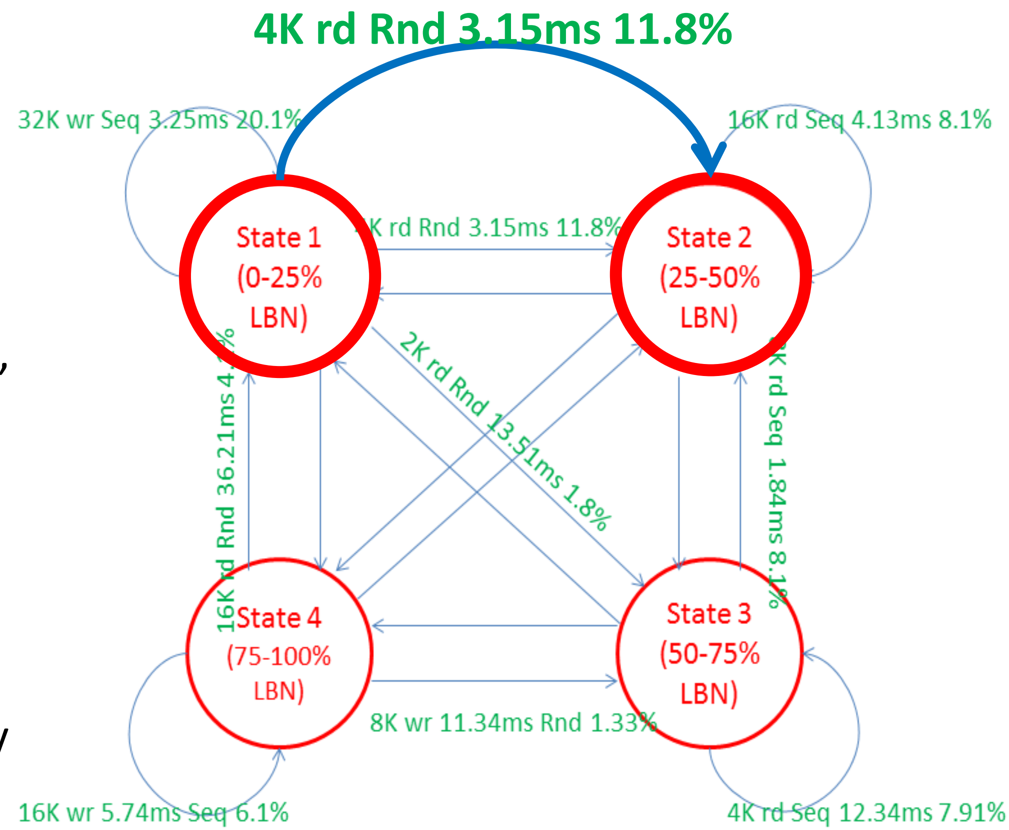
INTRODUCTION

- **GOAL**: Develop a statistical model for I/O accesses (3rd tier) of datacenter applications and a tool that recreates them with high fidelity
  - Replaying the original application in all storage configurations is **impractical** (time and cost)
  - DC applications are **not** publicly **available**
  - Storage System accounts for **20-30%** of **Power/TCO** of the system
- **Methodology**
  - Trace real data center workloads
    - Six large scale Microsoft applications
  - Design the storage model
  - Develop a tool that generates I/O requests based on the model
  - Validate model and tool (not recreating the app’s functionality)
  - Use the tool to evaluate storage systems for performance and efficiency

Model

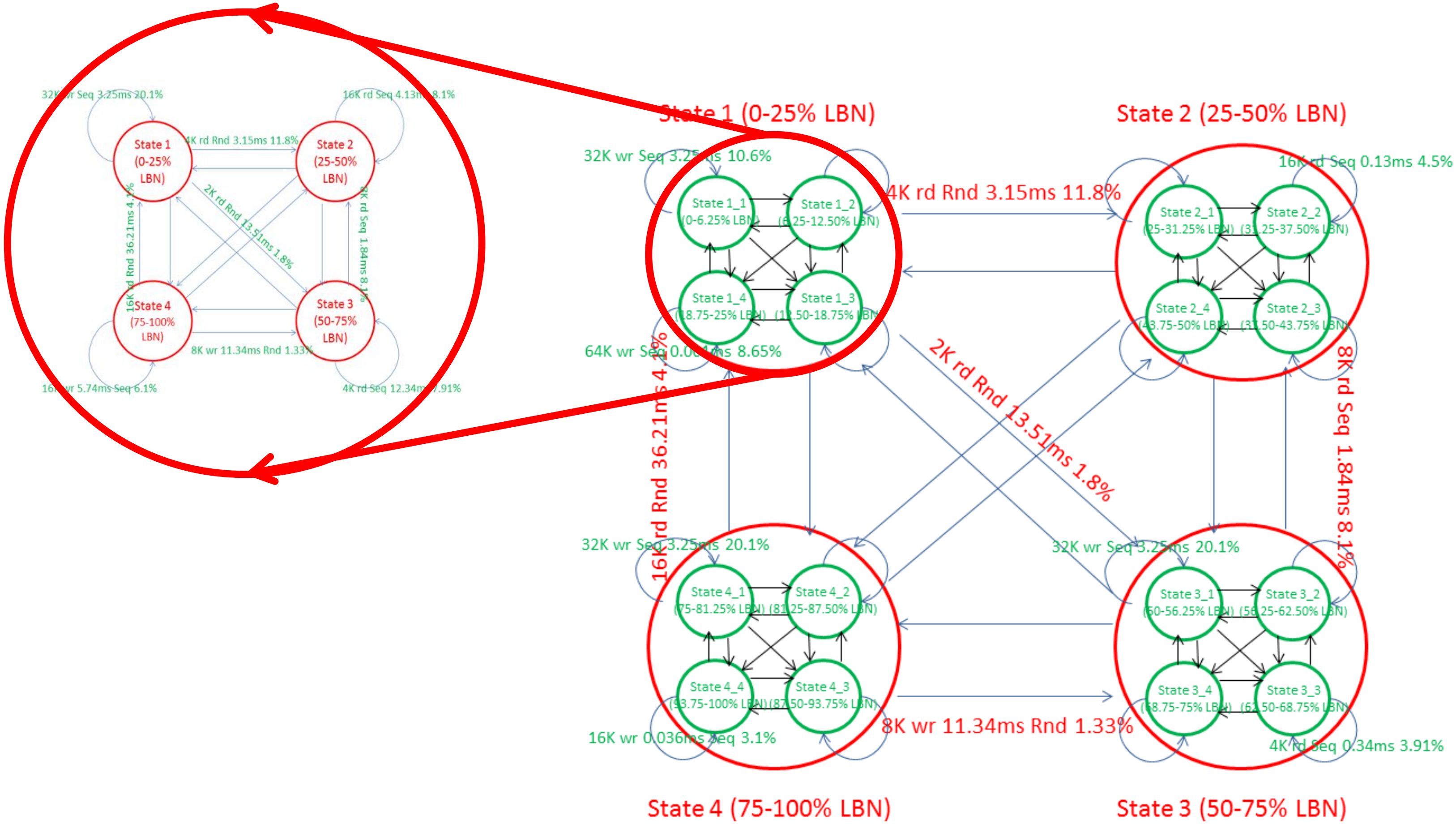
- **Probabilistic State Diagrams**
  - **State**: Block range on disk(s)
  - **Transition**: Probability of changing block range
  - **Stats**: rd/wr, rnd/seq, block size, inter-arrival time

- Single or Multiple Levels
  - Hierarchical representation
  - User defined level of granularity



(Reference: S.Sankar et al. (IISWC 2009))

Hierarchical Model



Comparison with Previous Tools (IOMeter)

- IOMeter is the most well-known open-source I/O workload generator
- DiskSpd is a workload generator maintained by the windows server perf team

Features	IOMeter	DiskSpd
Inter-Arrival Times (static or distribution)	✗	✓
Intensity Knob	✗	✓
Spatial Locality	✗	✓
Temporal Locality	✗	✓
Granular Detail of I/O Pattern	✗	✓
Individual File Accesses*	✗	✓

* more in defragmentation application

IMPLEMENTATION

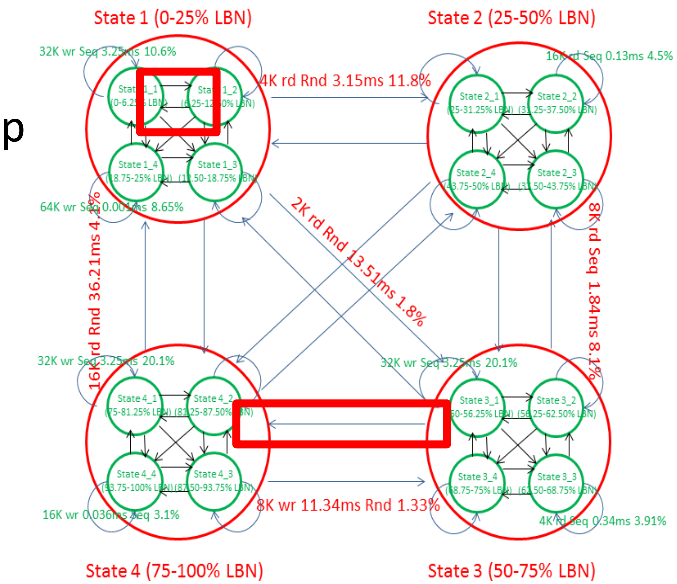
- 1/4: **Inter-arrival Times**:
  - Default version: **Outstanding I/Os**
  - **Inter-arrival Times ≠ Outstanding I/Os!!**
    - Inter-arrival Times: Property of the Workload
    - Outstanding I/Os: Property of System Queues
    - **Scaling inter-arrival times of independent requests => more intense workload**
    - **Scaling queue length of the system ≠ more intense workload**
  - Current version: **Static & Time Distributions** (normal, exponential, Poisson, Gamma)

- 2/4: **Multiple Threads** and **Thread Weights**
  - Default version: Multiple threads with the **same I/O characteristics**
  - Each transition in the model has different I/O features
  - Current version: Multiple threads with **individual I/O characteristics**
  - *Thread Weight*: Proportion of accesses corresponding to a thread (= transition)

IMPLEMENTATION

- **3/4: Understanding Hierarchy**
  - Increase levels -> More detailed information
  - Choose an **optimal number of levels** for each app

  - **In depth** rather than **"flat"** representation
    - Spatial Locality **within** states rather than **across** states
    - Difference in performance between **"flat"** and **"hierarchical"** model is **less than 5%**.



- **4/4: Intensity Knob**
  - Scale the inter-arrival times to emulate more intense workloads
  - Evaluation of faster storage systems, e.g. SSD-based
  - ***Assumptions***:
    - Most requests in DC apps come from different users -> independent I/Os
    - The application is not retuned in the faster system (spatial locality, I/O features remain constant)

Methodology

1. **Production DC Traces to Storage I/O Models**
   I. Collect traces from production servers (for various apps)
   II. **ETW : Event Tracing for Windows**
      I. Block offset, Block size, Type of I/O
      II. File name, Number of thread
      III. …
   III. Generate the **state diagram model with one or multiple levels** (XML format)
      - The model is trained on real DC traces

2. **Storage I/O Models to Synthetic Storage Workloads**
   I. Give the state diagram model as an input to DiskSpd to generate the synthetic I/O load.
   II. Use the synthetic workloads for performance, power, cost-optimization studies.

Experimental Infrastructure

- **Workloads – Original Traces:**
  - Messenger (SQL-based)
  - Display Ads (SQL-based)
  - WLS (Windows Live Storage) (SQL-based)
  - Email (online service)
  - Search (online service)
  - D-Process (distributed computing)

- **Traces Collection** and **Validation Experiments**:
  - Server Provisioned for SQL-based applications:
    - 8 cores, 2.26GHz
    - 5 physical volumes – 10 disk partitions total storage: **2.3TB HDD**
    - Synthetic workloads ran on corresponding disk drives (log I/O to Log drive, SQL queries to H: drive)

- **SSD Caching** and **IOMeter vs. DiskSpd Comparison**:
  - Server with SSD caches:
    - 12 cores, 2.27GHz
    - 4 physical volumes – 8 disk partitions total storage: **3.1TB HDD + 4x8GB SSD**

VALIDATION

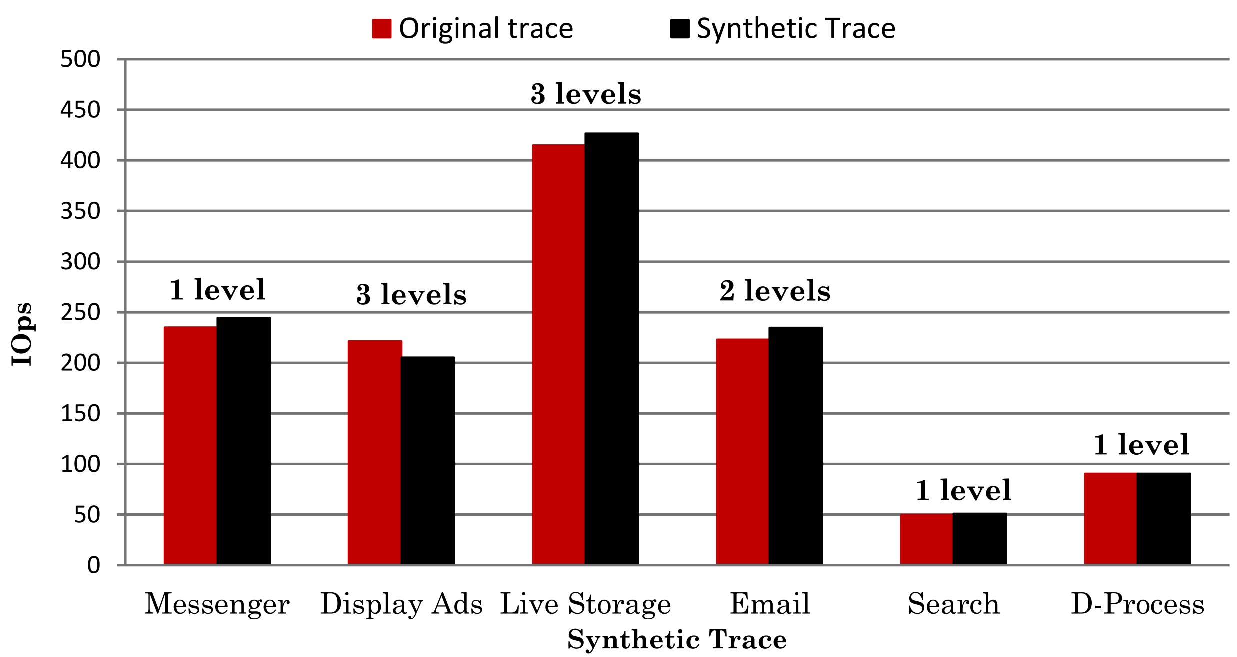
- Collect 24h long production traces from original DC apps
- Create one/multiple level state diagram models
- Run the synthetic workloads created based on the models
- Compare original – synthetic traces (I/O features + performance metrics)

Metrics	Original Workload	Synthetic Workload	Variation
Rd:Wr Ratio	1.8:1	1.8:1	**0%**
% of Random I/Os	83.67%	82.51%	**-1.38%**
Block Size Distr.	8K(87%) 64K (7.4%)	8K (88%) 64K (7.8%)	**0.33%**
Thread Weights	T1(19%) T2(11.6%)	T1(19%) T2(11.68%)	**0%-0.05%**
Avg. Inter-arrival Time	4.63ms	4.78ms	**3.1%**
Throughput (IOPS)	255.14	263.27	**3.1%**
Mean Latency	8.09ms	8.48ms	**4.8%**

Table: I/O Features – Performance Metrics Comparison for Messenger

VALIDATION

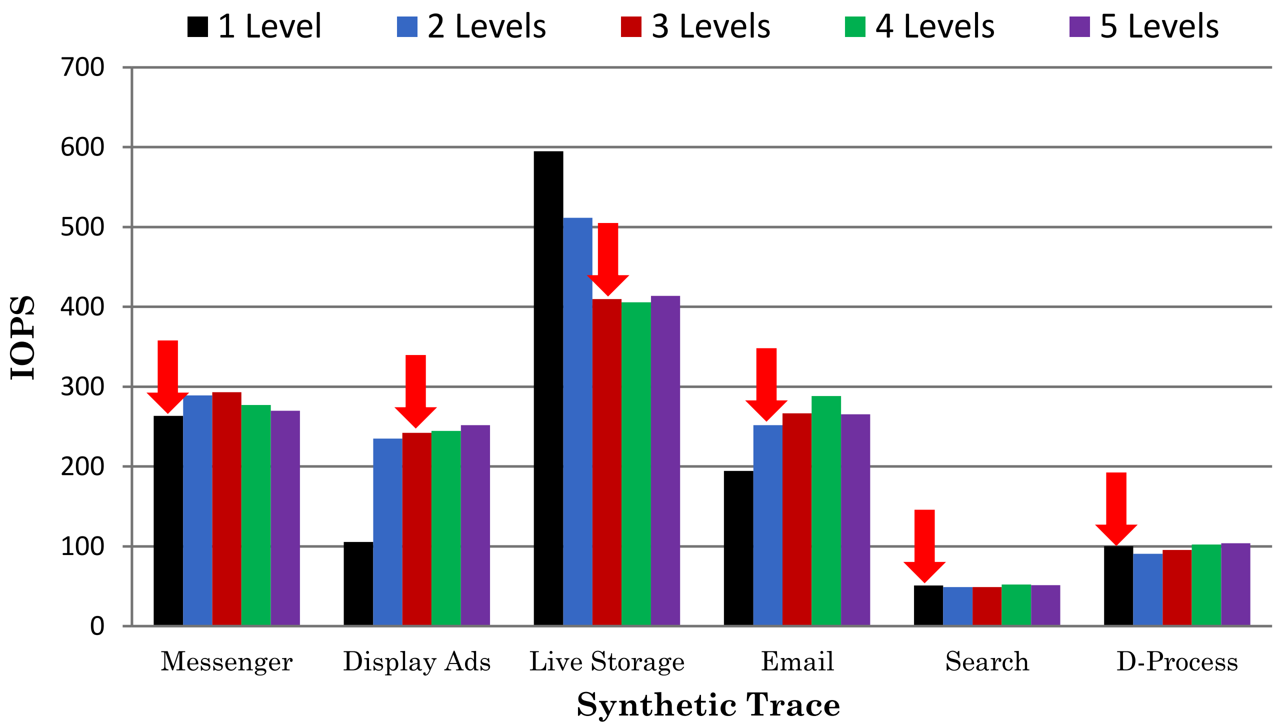
- Collect 24h long production traces from original DC apps
- Create one/multiple level state diagram models
- Run the synthetic workloads created based on the models
- Compare original – synthetic traces (I/O features + performance metrics)



Less than 5% difference in throughput

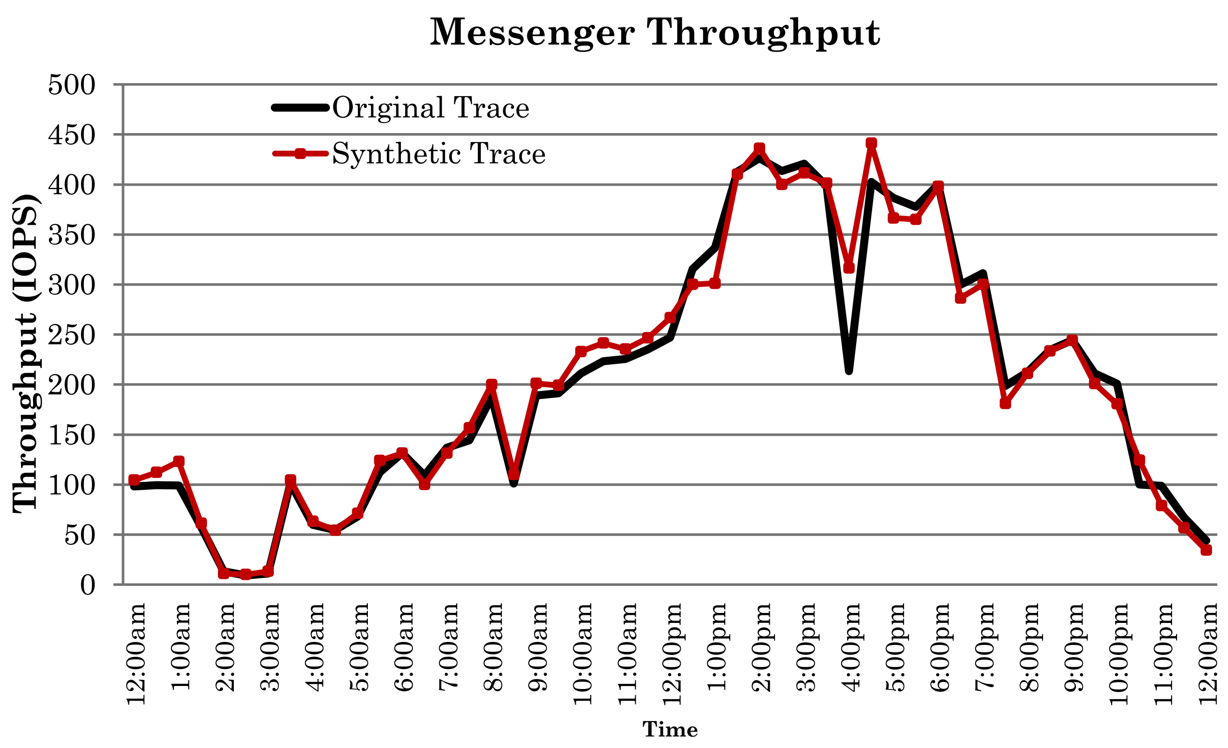
Choosing the Optimal Number of Levels

- **Optimal Number of Levels**: First level after which **less than 2%** difference in **IOPS**.



VALIDATION – ACTIVITY FLUCTUATION

- Inter-arrival Times averaged over small periods of time
- <u>**Captures the fluctuation (peaks, troughs) of storage activity**</u>



Comparison with IOMeter 1/2

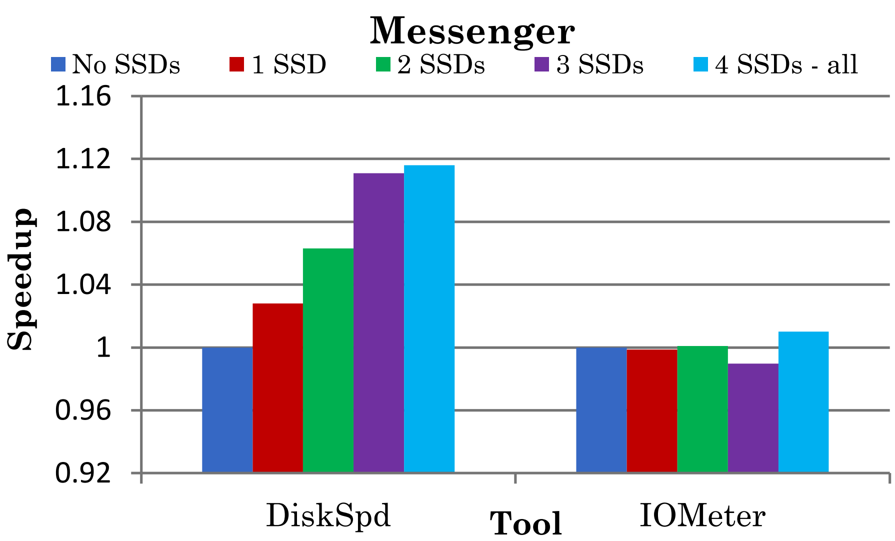
- **Comparison of Performance Metrics in Identical Simple Tests**

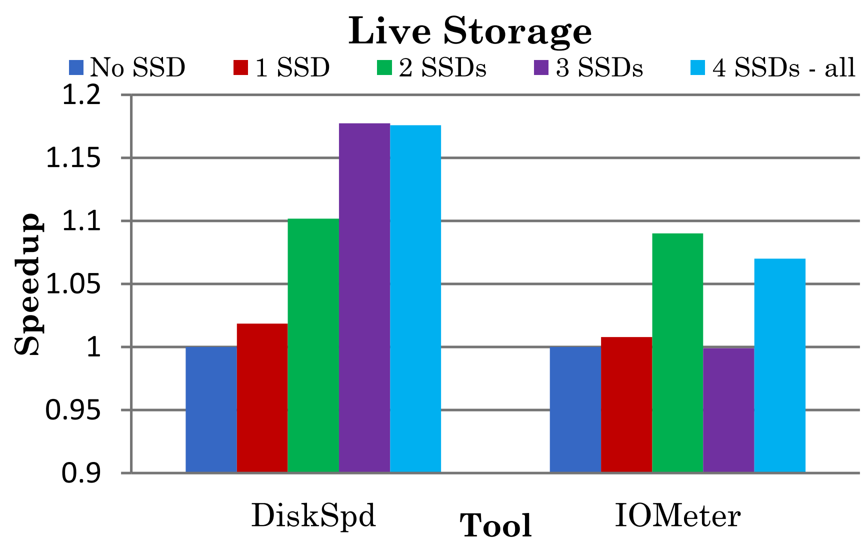
Test Configuration	IOMeter (IOPS)	DiskSpd (IOPS)
4K Int. Time 10ms Rd Seq	97.99	101.33
16K Int. Time 1ms Rd Seq	949.34	933.69
64K Int. Time 10ms Wr Seq	96.59	95.41
64K Int. Time 10ms Rd Rnd	86.99	84.32

Less than 3.4% difference in throughput in all cases

Comparison with IOMeter 2/2

- **Comparison on Spatial-Locality Sensitive Tests**





- No speedup with increasing number of SSDs (e.g. Messenger)
- Inconsistent speedup as SSD capacity increases (e.g. Live Storage)

APPLICABILITY – STORAGE SYSTEM STUDIES

1. SSD caching

- Add up to 4x8GB SSD caches, run the synthetic workloads
- Average 18% speedup

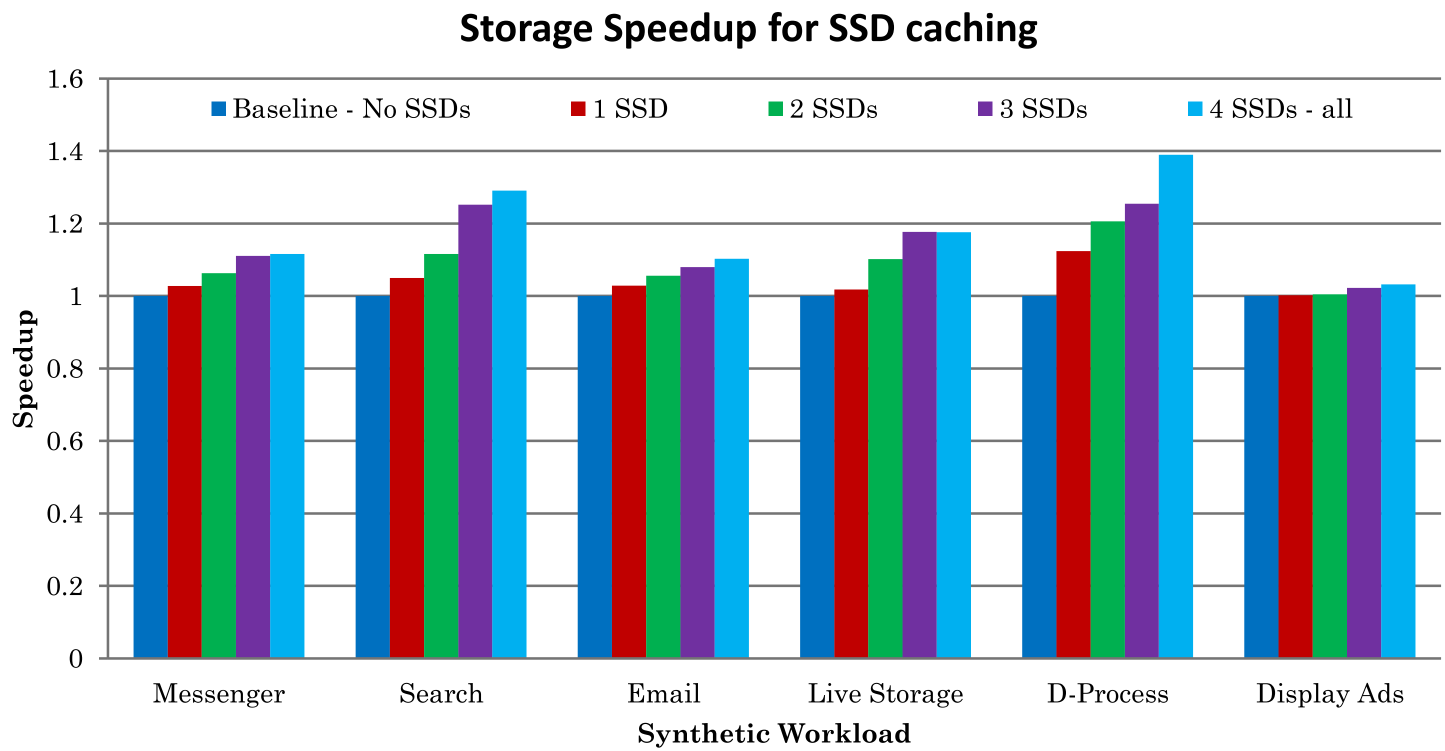
2. Defragmentation Benefits

- Rearrange blocks on disk to improve sequential characteristics
- Average 14% speedup, 11% improved power consumption

Using the model/tool made these studies easy to evaluate without <u>access to app code</u> or <u>full app deployment</u>

SSD CACHING

- Evaluate progressive SSD caching using the models
- Take advantage of spatial and temporal locality (frequently accessed states cached in SSDs)
- **Open-loop approach**: The workload is not retuned when switching to SSDs

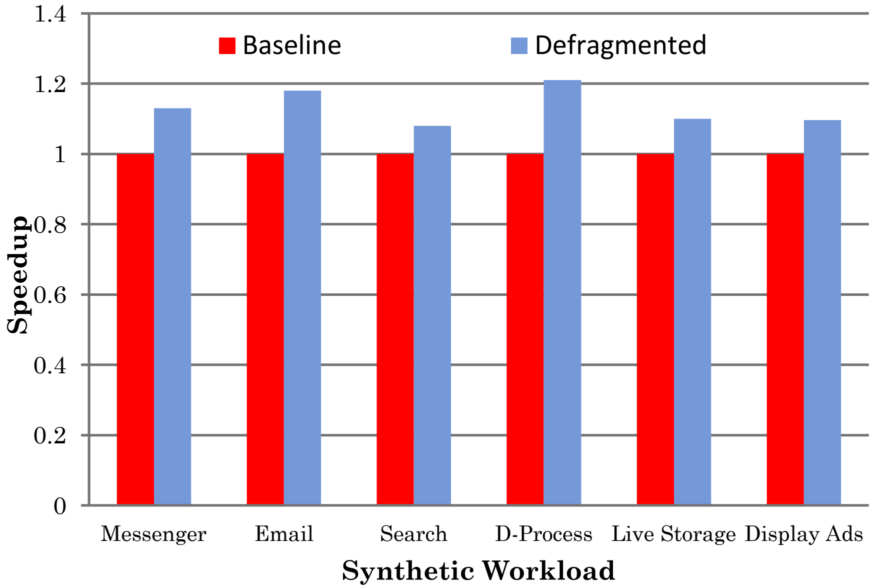


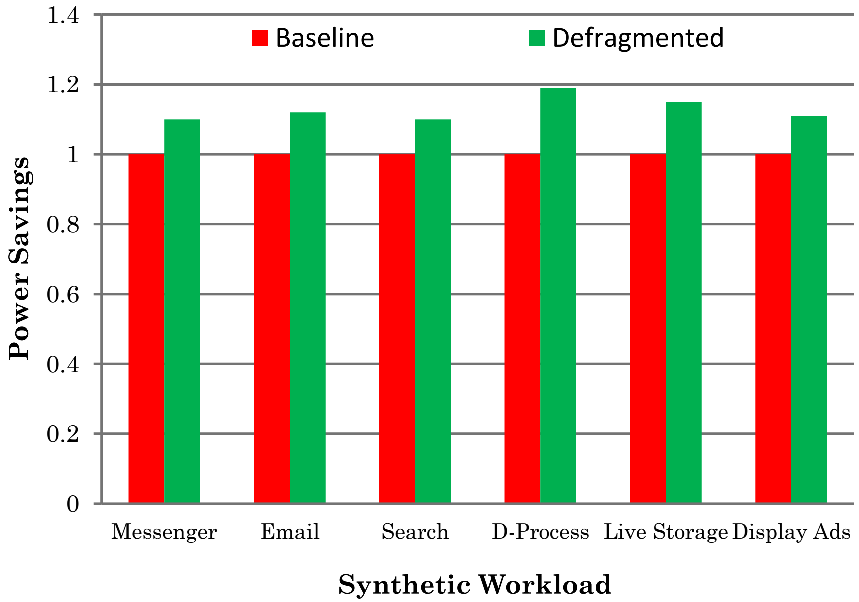
MOTIVATION FOR DEFRAGMENTATION

- Disks favor Sequential accesses, BUT, in most applications:
  - **Random > 80% - Sequential < 20%**
- Quantify the benefits of defragmentation using the models by rearranging blocks/files without actually performing defragmentation
- Evaluate different defragmentation policies (partial, optimal time for defragmentation)

Workload	Rd	Wr	Before Defrag		After Defrag	
			Random	Seq	Random	Seq
Messenger	62.8%	34.8%	83.67%	**15.35%**	63.17%	**35.74%**
Email	52.8%	45.2%	84.45%	**13.74%**	61.64%	**33.74%**
Search	49.8%	45.14%	87.71%	**8.46%**	70.87%	**24.46%**
Live Storage	58.31%	39.39%	93.09%	**5.48%**	73.21%	**24.99%**
D-Process	30.11%	68.76%	73.23%	**26.77%**	45.36%	**54.41%**
Display Ads	96.45%	2.45%	93.50%	**4.25%**	78.50%	**19.23%**

MOTIVATION FOR DEFRAGMENTATION





- **D-Process** and **Email** experience the highest benefit: 18-20% speedup and 14-20% in power consumption (highest Write/Read ratios)

CONCLUSIONS

- Studying DC applications is hard…
- **Modeling and Generation Framework:**
  - An accurate hierarchical statistical model that captures the fluctuation of I/O activity (including **spatial + temporal locality**) of **real DC applications**
  - A tool that recreates I/O loads with high fidelity (I/O features, performance metrics)
- This infrastructure can be used to make **accurate predictions** for storage studies that would **require access to real app code or full app deployment**
  - SSD caching
  - Defragmentation Benefits
  - Many more (ongoing work)…

Future Work

- **Full application models**
  - Capture all tiers (trace requests – in-depth approach)
  - Capture CPU, Memory, Network, I/O behavior (in-breadth approach)
  - Correlations between system parts

- **System Studies**
  - Application Consolidation
  - VM Migration
  - Power Management Techniques
  - Server Provisioning Studies
  - …

Questions??

Thank you