Decoupling Datacenter Studies from Access to Large-Scale Applications: A Modeling Approach for Storage Workloads

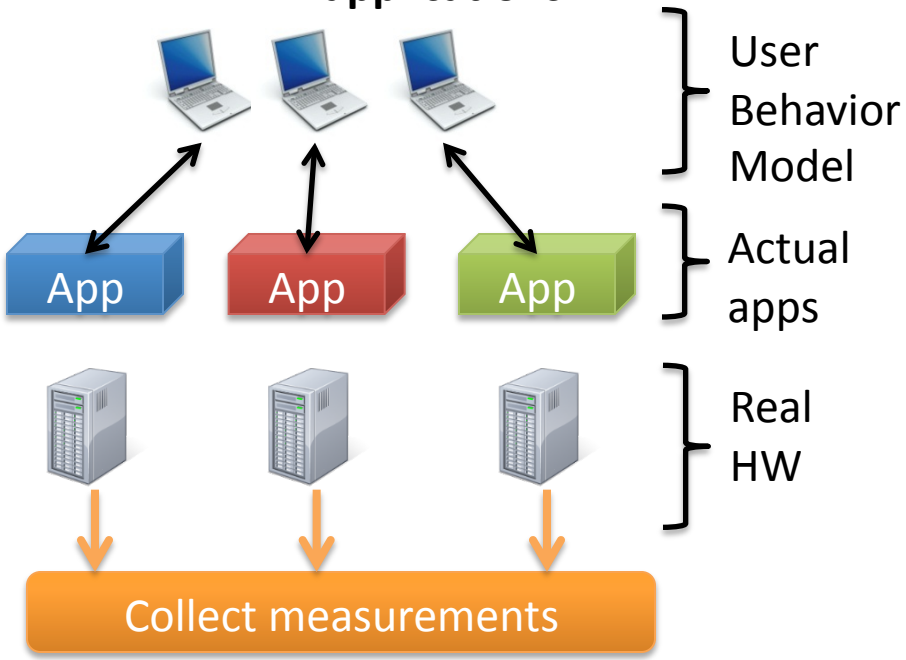
Christina Delimitrou[1], Sriram Sankar[2], Kushagra Vaid[2], Christos Kozyrakis[1]

[1]Stanford University, [2]Microsoft

IISWC– November 7th 2011

Datacenter Workload Studies

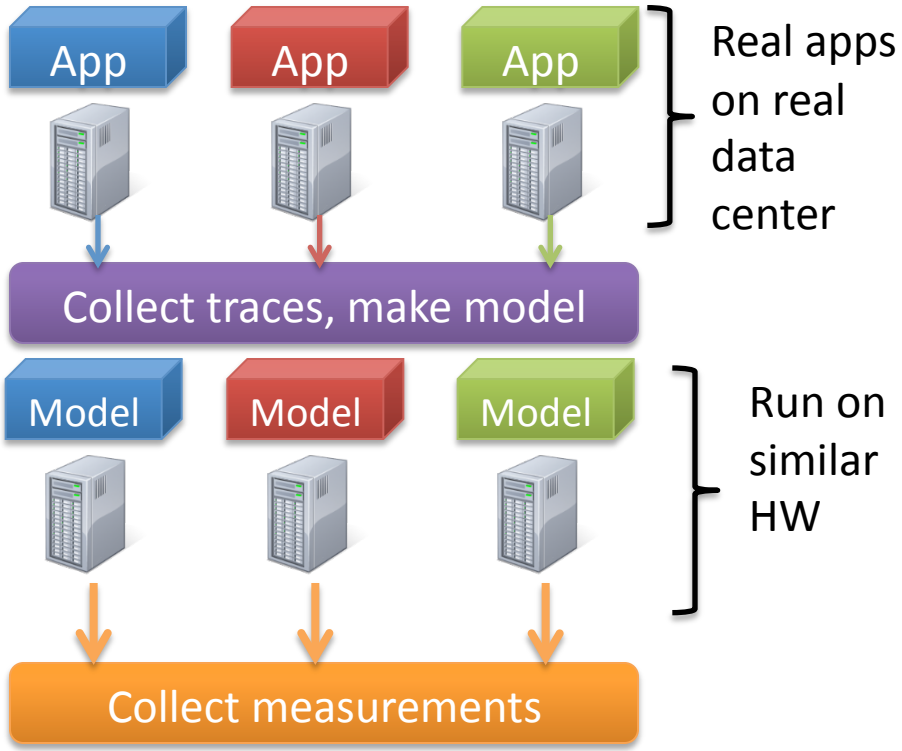
Open-source approximation of real applications



+ Pros: Resembles actual applications
+ Pros: Can modify the underlying hardware

- Cons: Not exact match to real DC applications

Statistical models of real applications

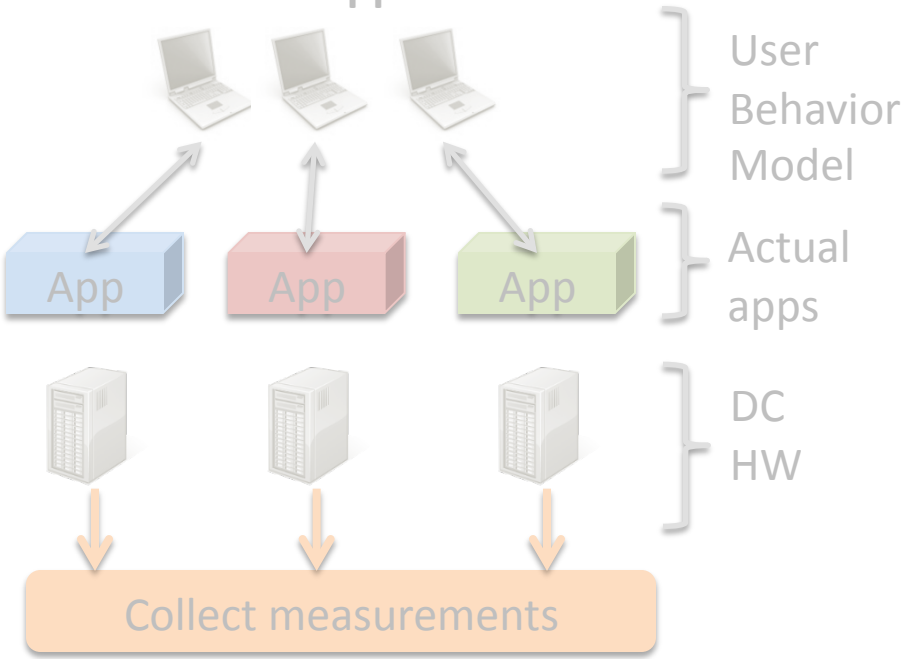


+ Pros: Trained on real apps – more representative

- Cons: Hardware and Code dependent
- Cons: Many parameters/dependencies to model

Datacenter Workload Studies

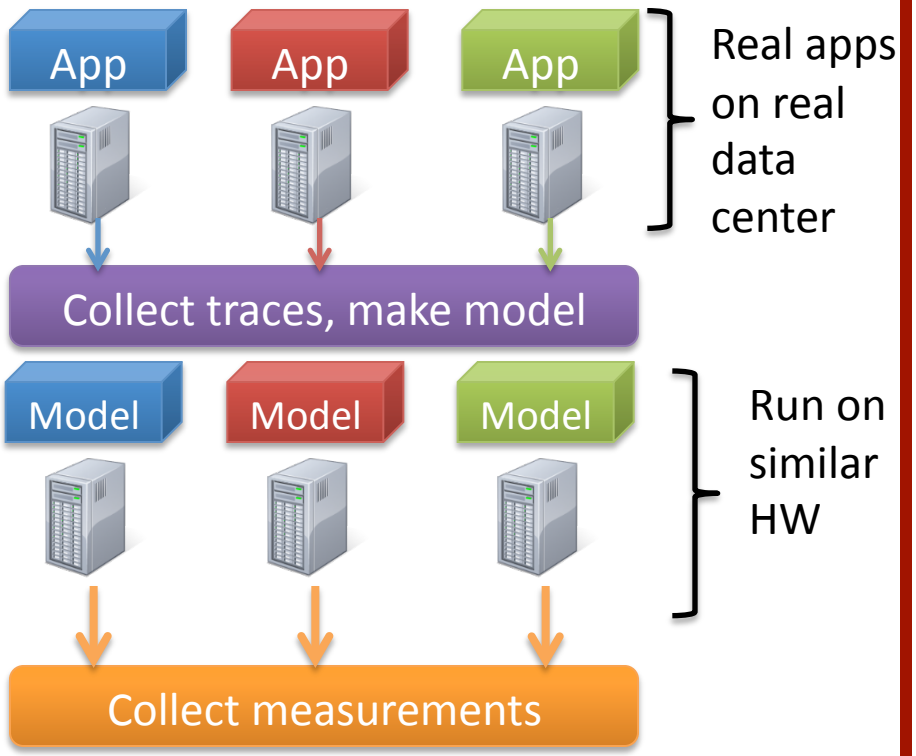
Open-source approximation of real applications



+ Pros: Resembles actual applications
+ Pros: Can modify the underlying hardware

- Cons: Not exact match to real DC applications

Use statistical models of real applications



+ Pros: Trained on real apps – more representative

- Cons: Hardware and Code dependent
- Cons: Many parameters/dependencies to model

Outline

- Introduction
- Modeling + Generation Framework
- Validation
- Decoupling Storage and App Semantics
- Use Cases
  - SSD Caching
  - Defragmentation Benefits
- Future Work

Executive Summary

- **Goal**
  - **Statistical model** for backend tier of DC apps + accurate **generation tool**
- **Motivation**
  - Replaying applications in many storage configurations is **impractical**
  - DC applications **not** publicly **available**
  - Storage system: **20-30%** of DC **Power/TCO**
- **Prior Work**
  - Does not capture key workload features (e.g., spatial/temporal locality)

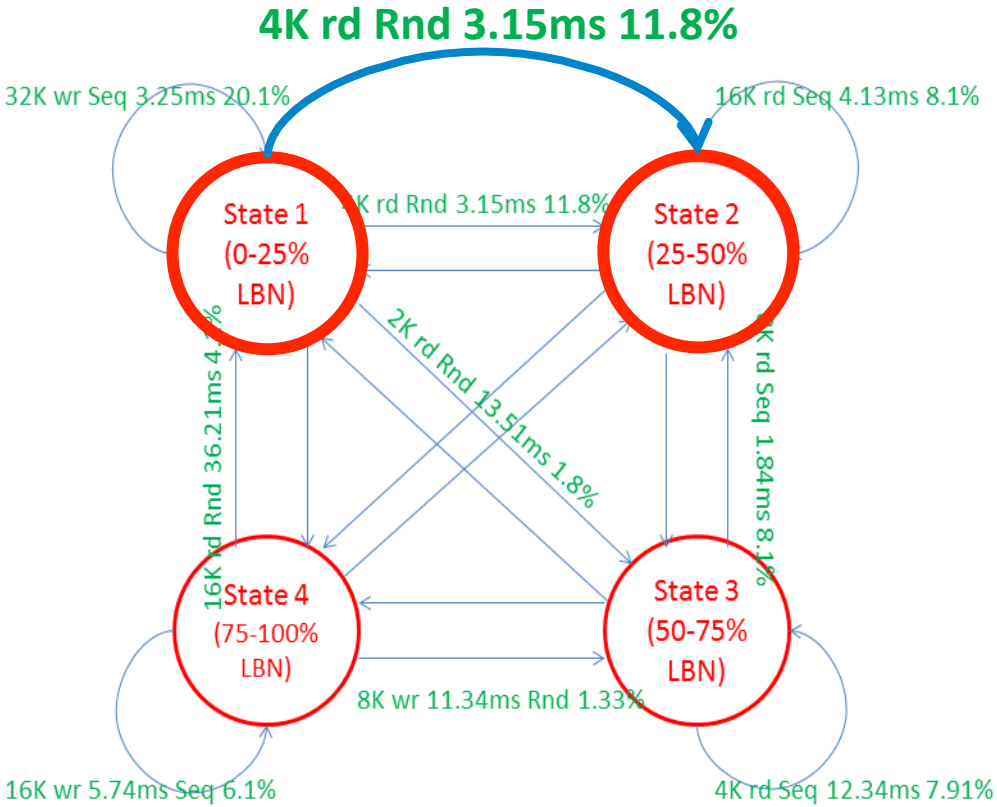
Executive Summary

- **Methodology**
  - Trace ten real large-scale Microsoft applications
  - Train a statistical model
  - Develop tool that generates I/O requests based on the model
  - Validate framework (model and tool)
  - Use framework for performance/efficiency storage studies

- **Results**
  - **Less than 5% deviation between original – synthetic workload**
  - **Detailed application characterization**
  - **Decoupled storage activity from app semantics**
  - **Accurate predictions of storage studies performance benefit**

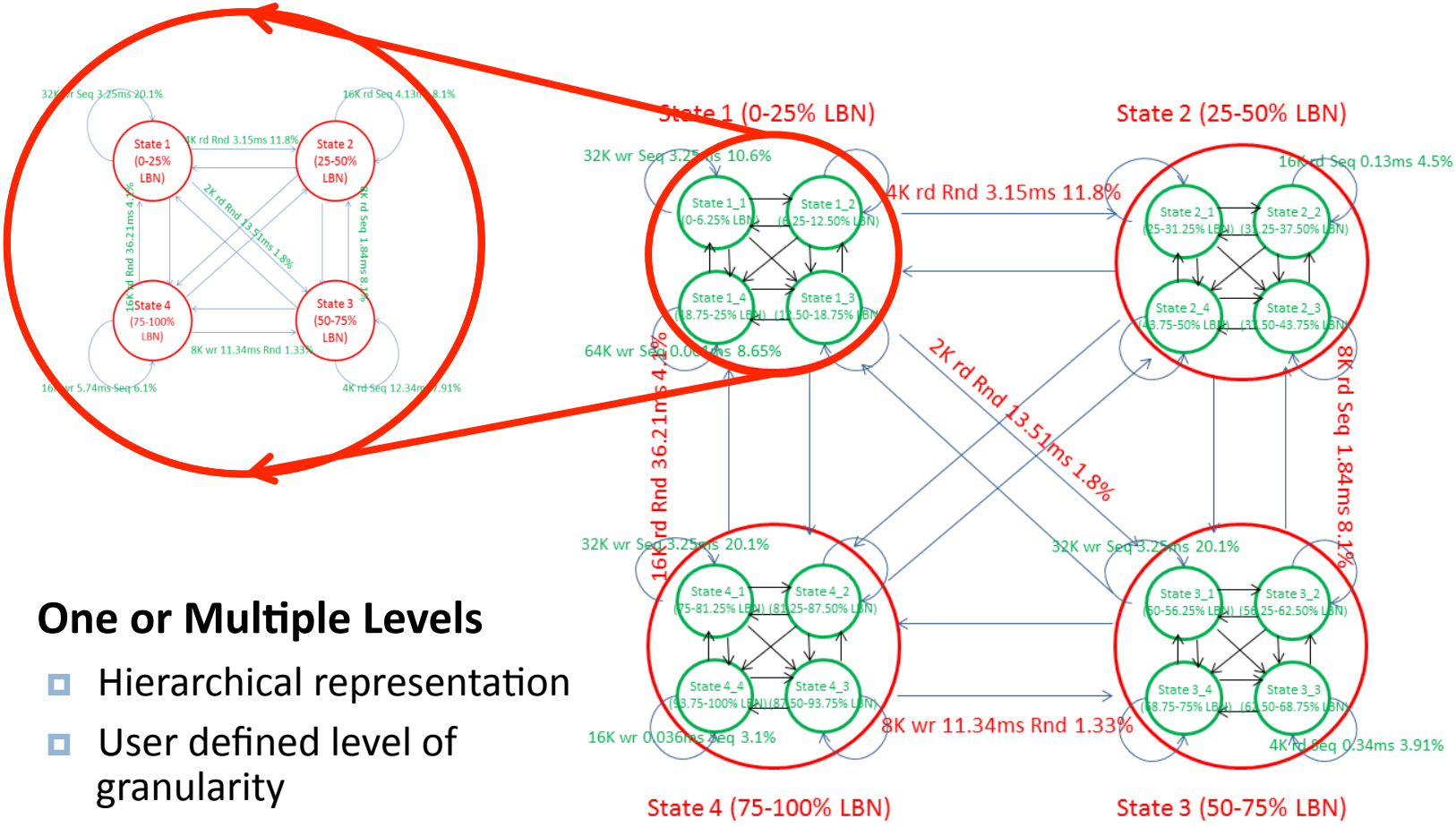
Model

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



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

Hierarchical Model



- **One or Multiple Levels**
  - Hierarchical representation
  - User defined level of granularity

Comparison with Previous Tools

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

Δ of 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 use case

Implementation (1/3): Inter-arrival times

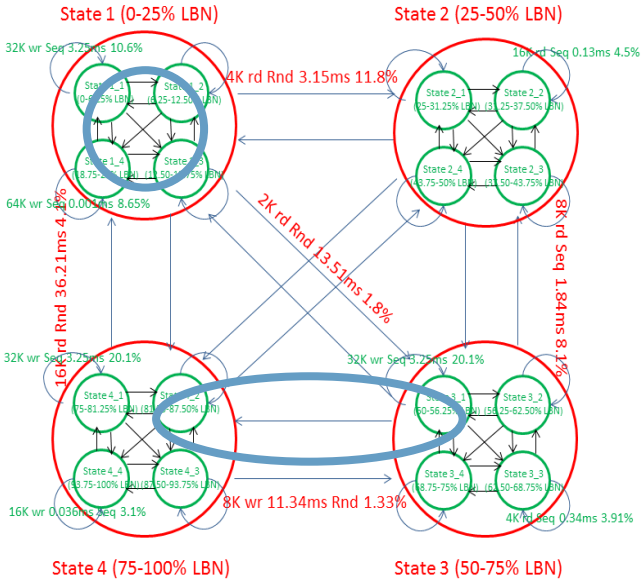
- **Inter-arrival times ≠ Outstanding I/Os!!**
  - **Inter-arrival times**: Property of the workload
  - **Outstanding I/Os**: Property of system queues
  - Scaling inter-arrival times of independent requests => more intense workload
- **Previous work relies on outstanding I/Os**
- **DiskSpd: Time distributions** (fixed, normal, exponential, Poisson, Gamma)
- Each transition has a ***thread weight***, i.e., a proportion of accesses corresponding to that transition
  - Thread weights are maintained both over short time intervals and across the workload's run

Implementation (2/3): Understanding Hierarchy

Levels++ -> Information++ -> Model Complexity++

Propose **hierarchical** rather than **flat** model:

- Choose **optimal number of states per level** (minimize inter-state transition probabilities)
- Choose **optimal number of levels** for each app (< 2% change in IOPS)



- Spatial locality **within** states rather than **across** states
- Difference in performance between **flat** and **hierarchical** model is **less than 5%**
- Reduce model complexity by **99% in transition count**

Implementation 3/3: Intensity Knob

- Scale inter-arrival times to emulate more intensive workloads
- Evaluation of faster storage systems, e.g., SSD-based systems
- ***Assumptions***:
  - Most requests in DC apps come from different users (independent I/Os), so scaling inter-arrival times is the expected behavior in the faster system
  - The application is not retuned for the faster system (spatial locality, I/O features remain constant) – may require reconsideration

Methodology

1. **Production DC Traces to Storage I/O Models**
   I. Collect traces from production servers of a real DC deployment
   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 **storage workload model with one or multiple levels** (XML format)

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 synthetic workloads for performance, power, cost-optimization studies.

Experimental Infrastructure

- **Workloads – Original Traces:**
  - ***Messenger***, ***Display Ads***, ***User Content*** (Windows Live Storage) (SQL-based)
  - ***Email***, ***Search*** and ***Exchange*** (online services)
  - ***D-Process*** (distributed computing)
  - ***TPCC***, ***TPCE*** (OLTP workloads)
  - ***TPCH*** (DSS workload)

- **Trace Collection** and **Validation Experiments**:
  - Server Provisioned for SQL-based applications:
    - 8 cores, 2.26GHz
    - Total storage: **2.3TB HDD**

- **SSD Caching** and **IOMeter vs. DiskSpd Comparison**:
  - Server with SSD caches:
    - 12 cores, 2.27GHz
    - Total storage: **3.1TB HDD + 4x8GB SSD**

Validation

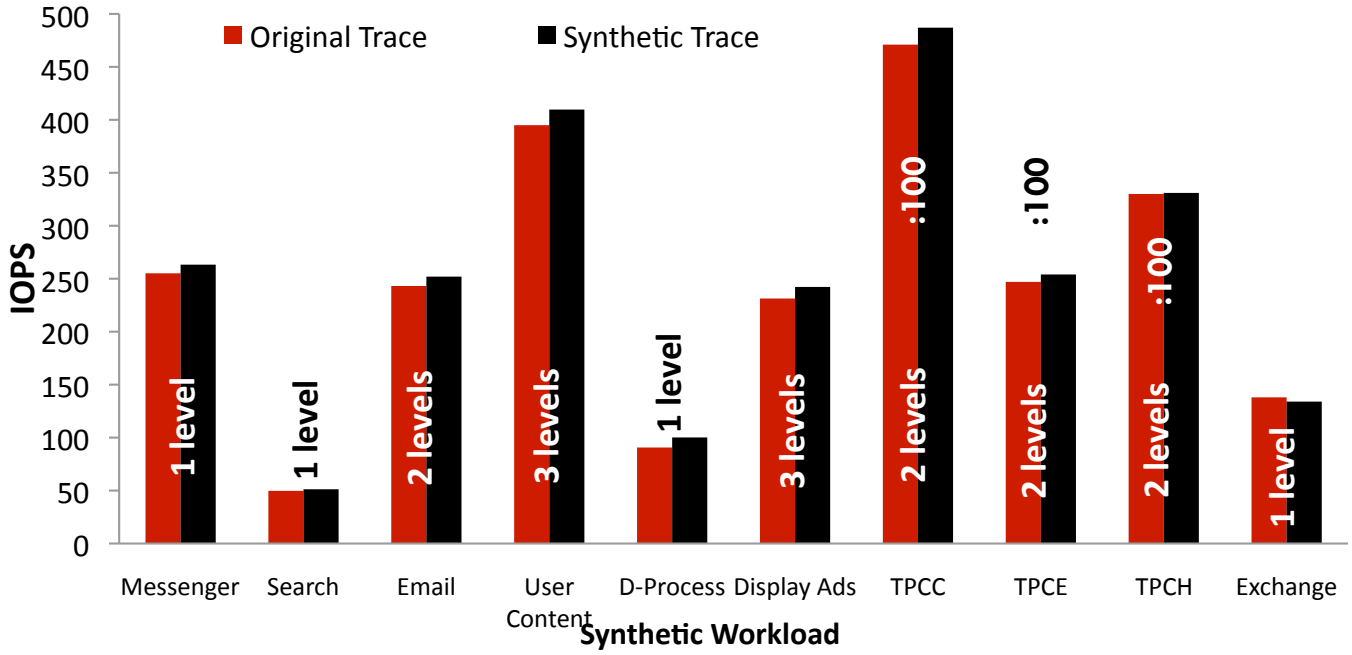
- Compare statistics from original app to statistics from generated load
  - Models developed using 24h traces and multiple levels
- Synthetic workloads ran on appropriate disk drives (log I/O to Log drive, SQL queries to H: drive)

Metrics	Original Workload	Synthetic Workload	Variation
Rd:Wr Ratio	1.8:1	1.8:1	0%
Random %	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

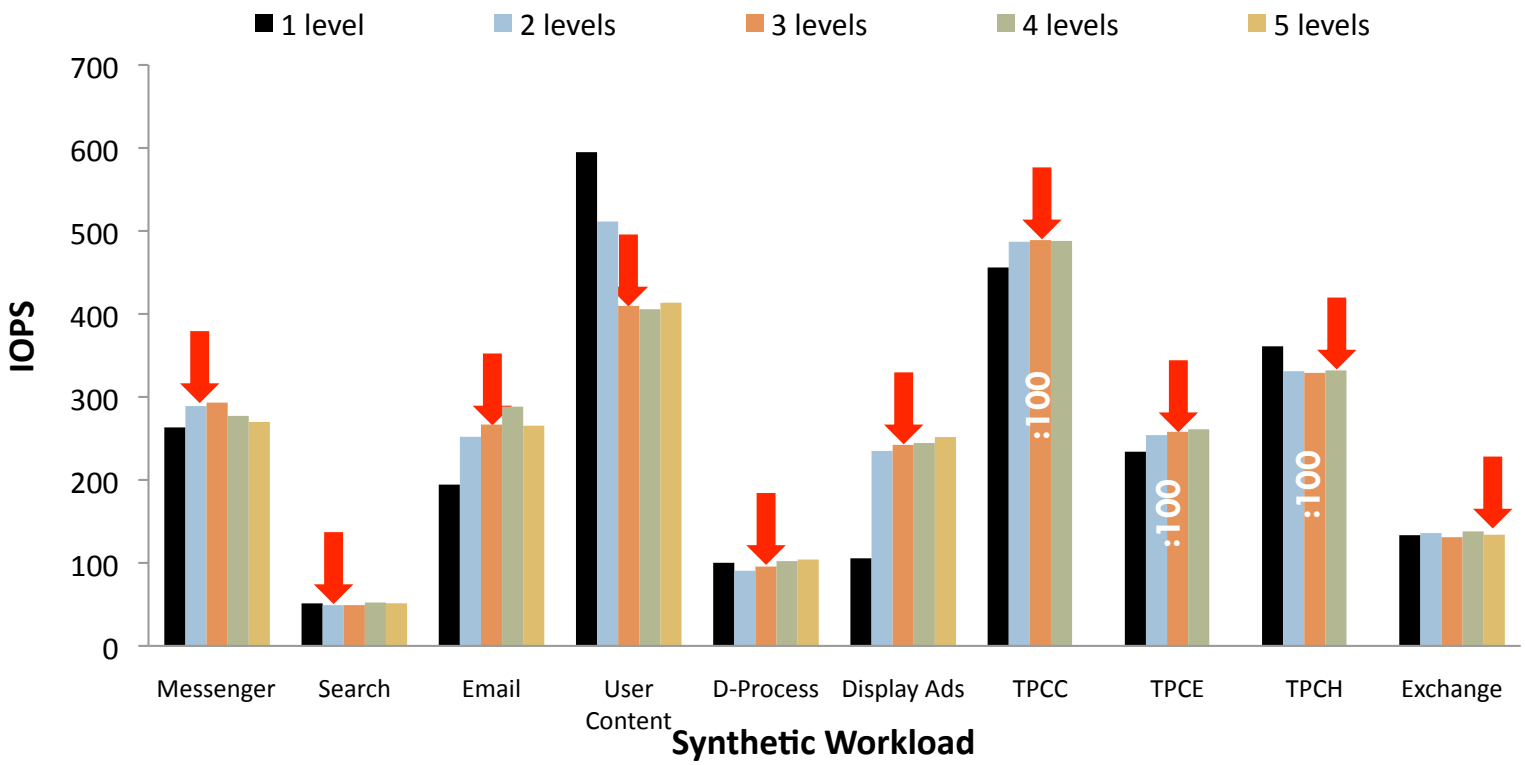
- Compare statistics from original app to statistics from generated load
  - Models developed using 24h traces and multiple levels
- Synthetic workloads ran on appropriate disk drives (log I/O to Log drive, SQL queries to H: drive)



Less than 5% difference in throughput

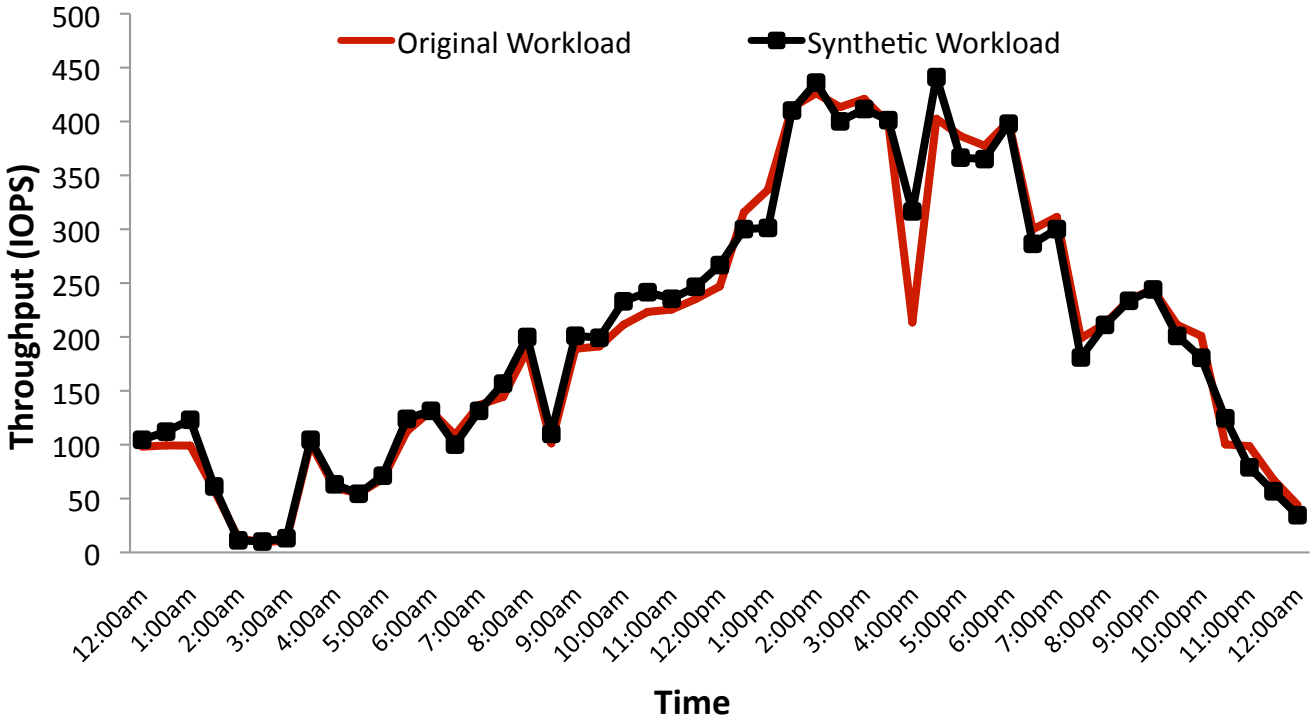
Choosing the Optimal Number of Levels

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



Validation

- Verify the accuracy in storage activity fluctuation

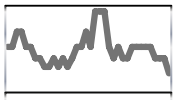
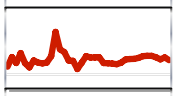
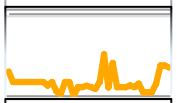
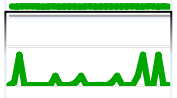


Less than 5% difference in throughput in most intervals and on average

Decoupling Storage Activity from App Semantics

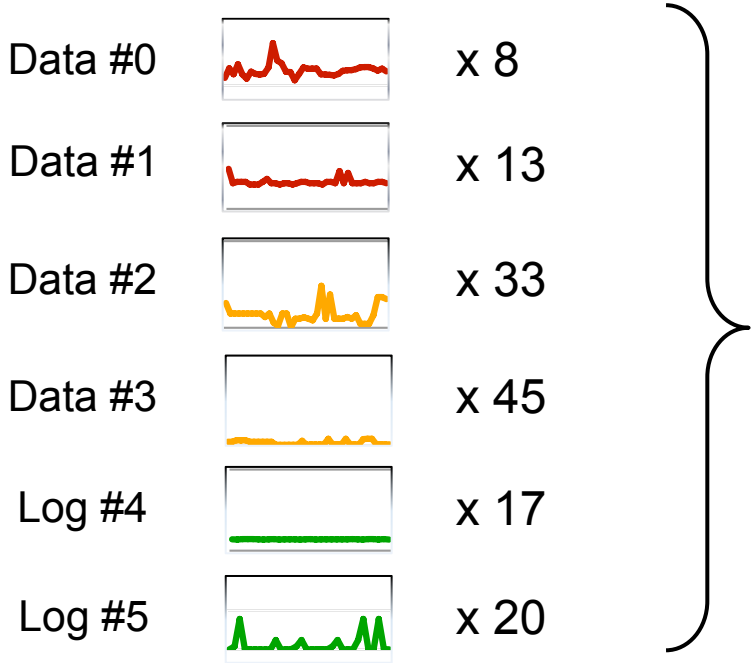
- Use the model to categorize and characterize storage activity per thread
- Filter I/O requests per thread and categorize based on:
  - Functionality (Data/Log thread)
  - Intensity (frequent/infrequent requests)
  - Activity fluctuation (constant/high request rate fluctuation)

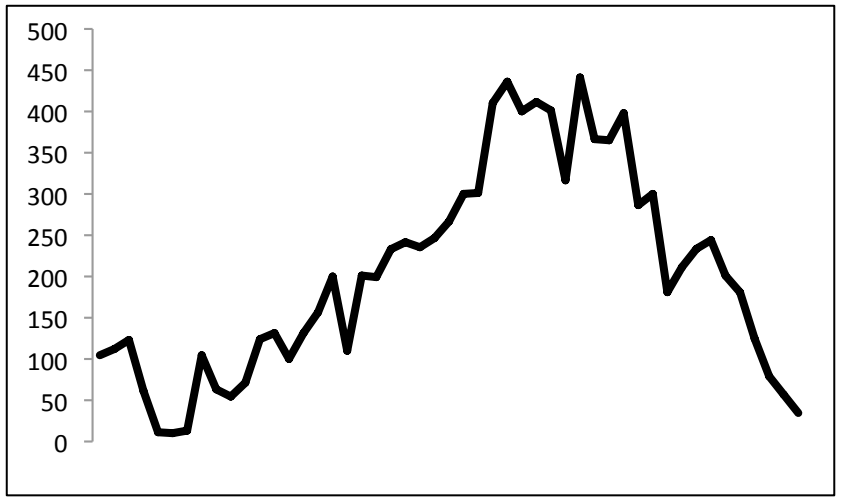
Per Thread Characterization for Messenger

Thread Type		Functionality	Intensity	Fluctuation	Weight
Total		Data + Log	High	High	1.00
Data #0		Data	High	High	0.42
Data #1		Data	High	Low	0.27
Data #2		Data	Low	High	0.13
Data #3		Data	Low	Low	0.18
Log #4		Log	High	Low	5E-3
Log #5		Log	Low	High	4E-4

Decoupling Storage Activity from App Semantics

- Reassemble the workload from the thread types:





- Recreate correct mix of threads (types + ratios) -> same storage activity as original application **without** requiring knowledge on application semantics
- **Decouples storage studies from application semantics**

Comparison with IOMeter 1/2

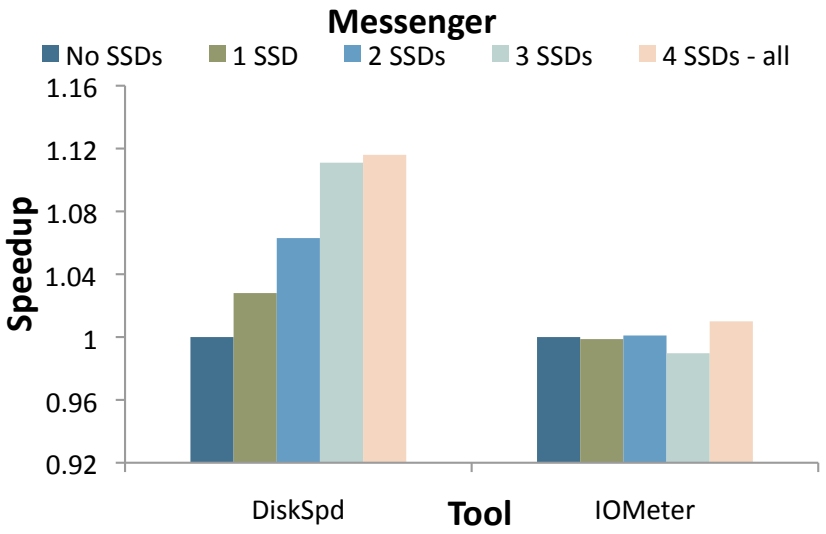
- **Comparison of performance metrics in identical simple tests (no spatial locality)**

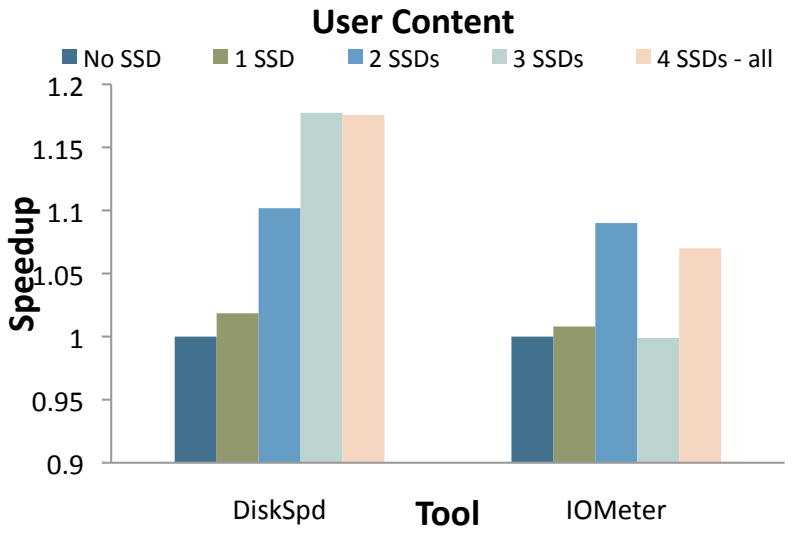
Test Configuration	IOMeter (IOPS)	DiskSpd (IOPS)
4K Int. Time 10ms Rd Seq	97.99	101.33
16K Int. Time 1ms Rd Seq	949.34	933.69
64K Int. Time 10ms Wr Seq	96.59	95.41
64K Int. Time 10ms Rd Rnd	86.99	84.32

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., User Content)

Applicability – Storage System Studies

1. SSD Caching

- Add up to 4x8GB SSD caches, run the synthetic workloads
- On average 31% speedup

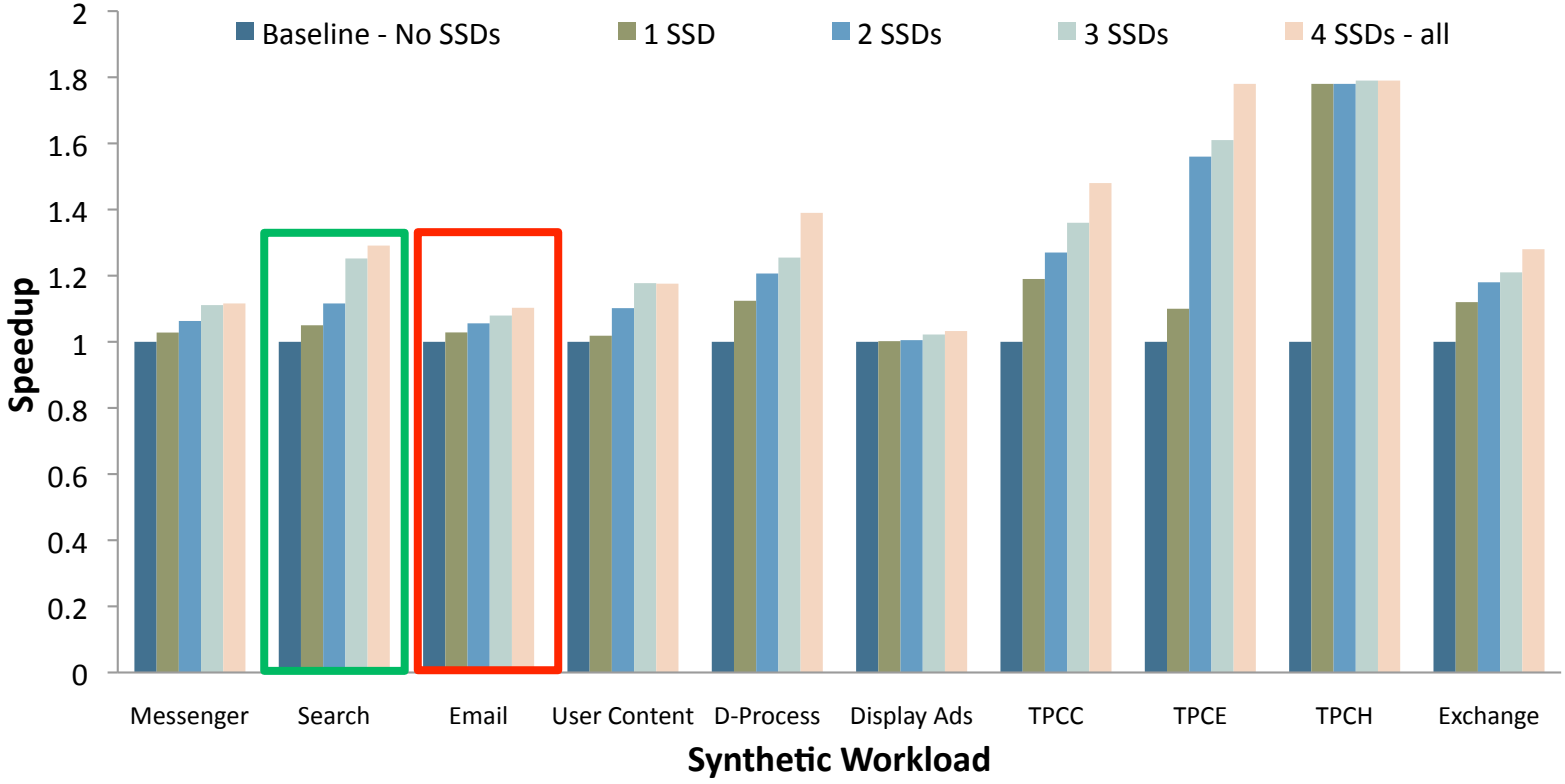
2. Defragmentation Benefits

- Rearrange blocks on disk to improve sequential characteristics
- On average 24% speedup, 11% improved power consumption

The modeling framework made these studies easy to evaluate without access to application code or full application deployment

SSD Caching

- Evaluate progressive SSD caching using the models
- Take advantage of spatial and temporal locality (frequently accessed blocks in SSDs)
- **Significant benefits - Search**: High I/O aggregation
- **No benefits - Email**: No I/O aggregation

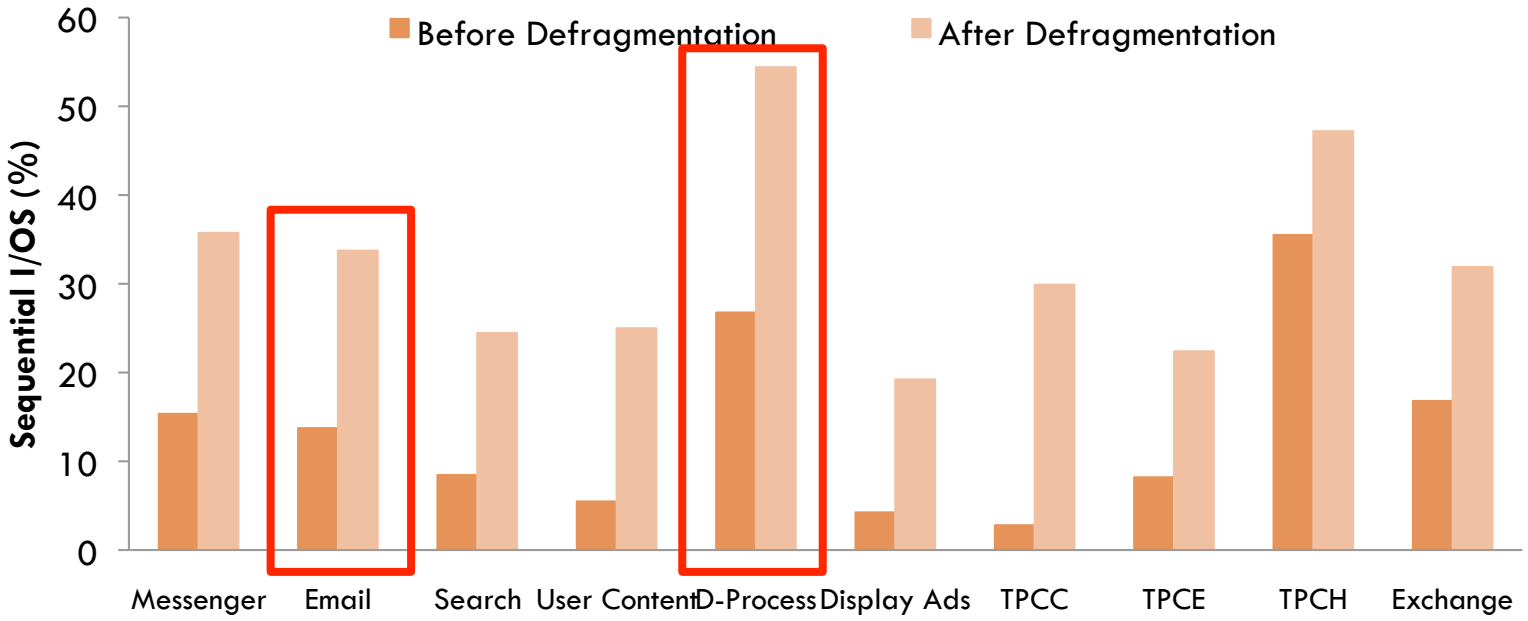


Defragmentation

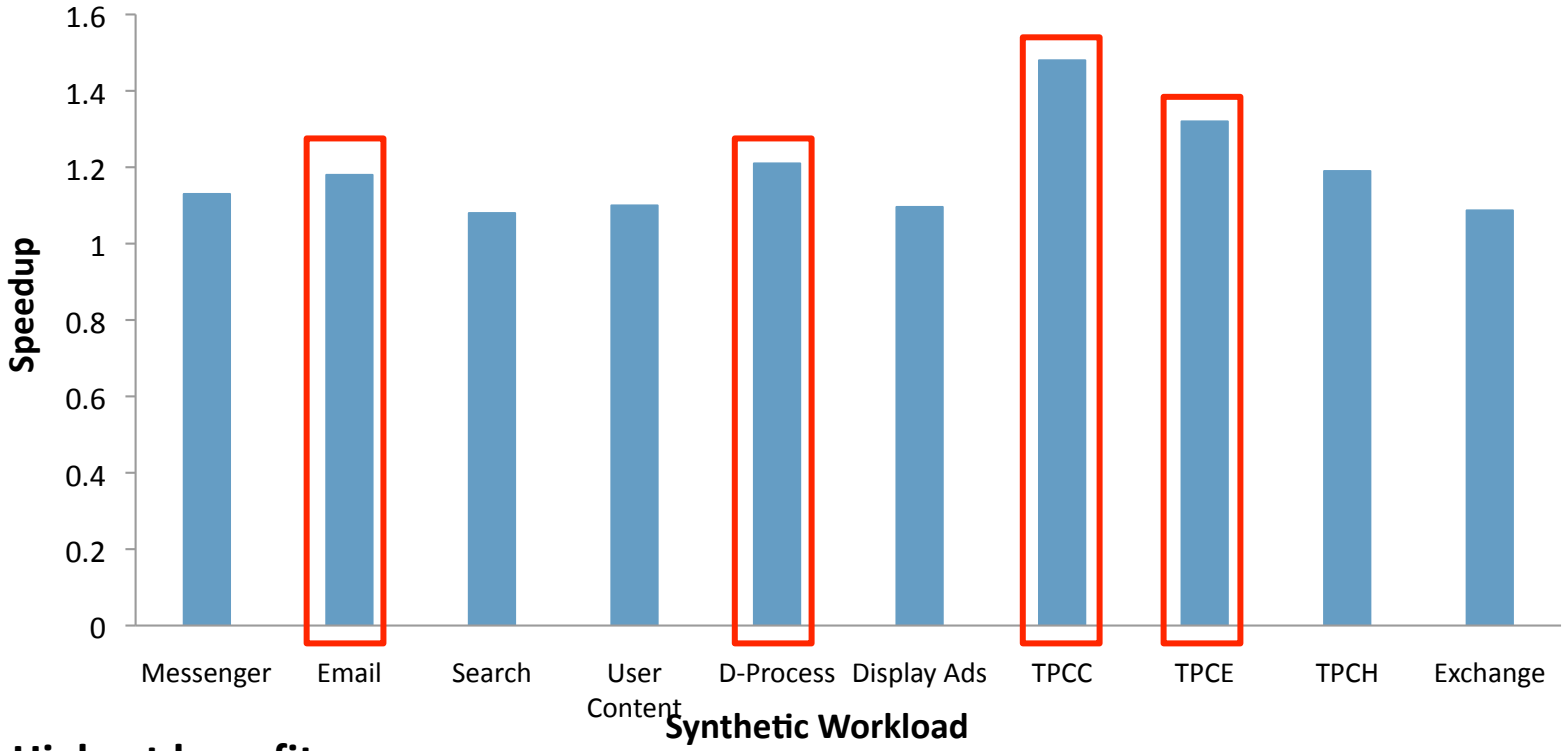
- Disks favor Sequential accesses, BUT, in most applications:

  Random > 80% - Sequential < 20%

- Quantify the benefit of defragmentation using the models by rearranging blocks/files without actually performing defragmentation
- Evaluate different defragmentation policies (e.g., partial, dynamic)

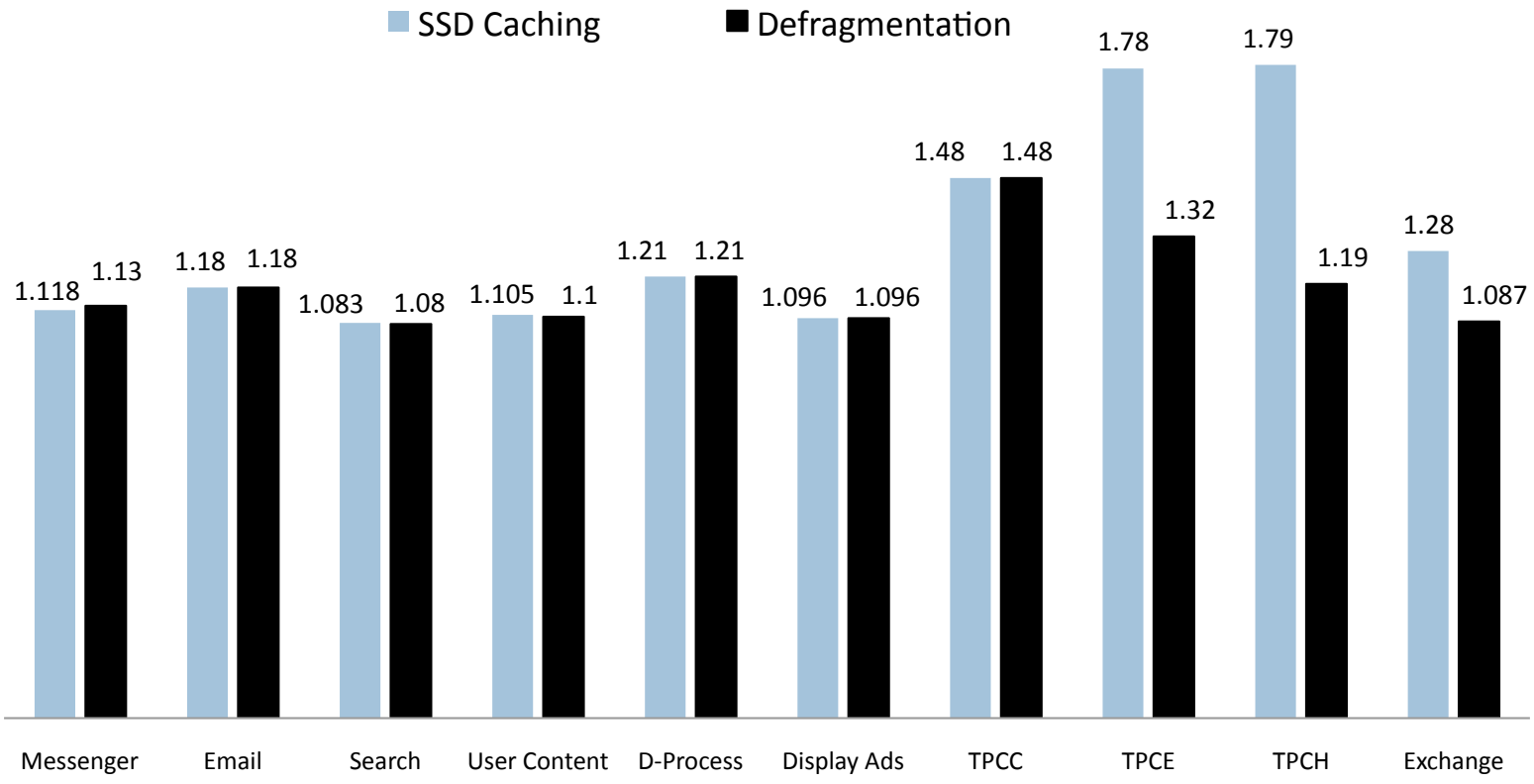


Defragmentation



- **Highest benefits:**
  - **TPCC/TPCE** which benefit from accessing consecutive database entries
  - **D-Process** and **Email** which have the highest Write/Read ratios

SSD Caching vs. Defragmentation



Most beneficial storage optimization depends on the application and system of interest

Conclusions

- Simplify the study of DC applications

- **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

- Full application models + full system studies (future work)

Questions??

Thank you

Contact:

cdel@stanford.edu

srsankar@microsoft.com

Defragmentation

- Disks favor Sequential accesses, BUT, in most applications:

  Random > 80% - Sequential < 20%

- Quantify the benefit of defragmentation using the models by rearranging blocks/files without actually performing defragmentation
- Evaluate different defragmentation policies (e.g., partial, dynamic)

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%**
User Content	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%**
TPCC	68.8%	31.2%	97.2%	**2.8%**	71.1%	**29.9%**
TPCE	91.3%	8.7%	91.9%	**8.2%**	77.7%	**22.4%**
TPCH	96.7%	3.3%	65.5%	**35.5%**	52.8%	**47.2%**
Exchange	32.0%	68.1%	83.2%	**16.8%**	68.1%	**31.9%**