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

Christina Delimitrou¹, Sriram Sankar², Kushagra Vaid², Christos Kozyrakis¹

¹Stanford University, ²Microsoft

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

Collect measurements
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

Collect measurements

User Behavior Model
Realistic apps
DC HW

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

Real apps on real data center

Collect traces, make model

Collect measurements

Run on similar HW
Outline

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

<table>
<thead>
<tr>
<th>Features</th>
<th>IOMeter</th>
<th>DiskSpd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-Arrival Times</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Intensity Knob</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Spatial Locality</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Temporal Locality</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Granular Detail of I/O Pattern</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Individual File Accesses*</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

* 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)

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

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**

![Graph showing comparison between original and synthetic traces for various applications.](image-url)
**CHOOSING THE OPTIMAL NUMBER OF LEVELS**

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

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

**Messenger Throughput**

![Graph showing throughput over time](image)

- Throughput (IOPS)
- Time
- Original Trace
- Synthetic Trace
**Comparison with IOMeter 1/2**

Comparison of Performance Metrics in Identical Simple Tests

<table>
<thead>
<tr>
<th>Test Configuration</th>
<th>IOMeter (IOPS)</th>
<th>DiskSpd (IOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K Int. Time 10ms Rd Seq</td>
<td>97.99</td>
<td>101.33</td>
</tr>
<tr>
<td>16K Int. Time 1ms Rd Seq</td>
<td>949.34</td>
<td>933.69</td>
</tr>
<tr>
<td>64K Int. Time 10ms Wr Seq</td>
<td>96.59</td>
<td>95.41</td>
</tr>
<tr>
<td>64K Int. Time 10ms Rd Rnd</td>
<td>86.99</td>
<td>84.32</td>
</tr>
</tbody>
</table>

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 access to app code or full app deployment
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

![Storage Speedup for SSD caching](chart.png)
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)

<table>
<thead>
<tr>
<th>Workload</th>
<th>Rd</th>
<th>Wr</th>
<th>Before Defrag</th>
<th>After Defrag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Random</td>
<td>Seq</td>
</tr>
<tr>
<td>Messenger</td>
<td>62.8%</td>
<td>34.8%</td>
<td>83.67%</td>
<td>15.35%</td>
</tr>
<tr>
<td>Email</td>
<td>52.8%</td>
<td>45.2%</td>
<td>84.45%</td>
<td>13.74%</td>
</tr>
<tr>
<td>Search</td>
<td>49.8%</td>
<td>45.14%</td>
<td>87.71%</td>
<td>8.46%</td>
</tr>
<tr>
<td>Live Storage</td>
<td>58.31%</td>
<td>39.39%</td>
<td>93.09%</td>
<td>5.48%</td>
</tr>
<tr>
<td>D-Process</td>
<td>30.11%</td>
<td>68.76%</td>
<td>73.23%</td>
<td>26.77%</td>
</tr>
<tr>
<td>Display Ads</td>
<td>96.45%</td>
<td>2.45%</td>
<td>93.50%</td>
<td>4.25%</td>
</tr>
</tbody>
</table>
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
  - ...
Thank you