Optimizing Complex Hierarchical Memory Systems Through Simulations

Andy Banta
Storage Janitor, Magnition Inc
@andybanta
Andy Banta

Magnition.io (Consultant)
SolidFire (VMware development)
DataGravity (Container exploitation lead)
VMware (iSCSI Tech Lead)
Sun Microsystems (Initial Fibre Channel development)
Patent, early distributed network projects, data acquisition
@andybanta
The Challenge
Modern compute and storage system use multiple layers interacting in multiple ways

HOW CAN CURRENT TECHNOLOGY ACHIEVE...

- Latency control
- Multi-tenant thrash remediation
- Correct tier sizing
- Workload-awareness
- Hot working set management
- Latency and throughput SLAs
- Memory capacity planning

AS MORE HARDWARE LAYERS ADD COMPLEXITY?
ABOUT MAGNITION
STORAGE PERFORMANCE, REINVENTED

World’s First Real-Time Data Placement Optimization
Patented technology is a first for the industry.

Proven At-Scale, with Production Workloads
Use customer traces to fully test diverse workloads in real-time.

Peer-Reviewed and Published in Leading Journals
Multiple industry articles published and reviewed.

Award-Winning, Patented Technology
3-time award winner for innovative technology.
Stimulating Simulations

Our Approach to Simulations
A different approach to optimization

- Compose simulations of complex memory and storage
- Break the simulation into components
- Allows the components to be assembled like building blocks
- Provide reasonable but constrained set of variables
- Run simulations with synthetic data or actual IO traces
Value of simulations

- Faster and easier to prototype
- Minimal up-front hardware spend
- Great opportunities for optimizations
- Loads of simulations are done at ASIC level
  - The same practices should apply to component and software levels
- Choose three
  1. Lower cost
  2. Higher speed
  3. More flexibility
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of components ready to use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of ready components for use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of ready components for use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of ready components for use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of ready components for use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire
Composable components

- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of ready components for use
- Allows clients to build their own modules
- Basic set of building blocks

- Cache
- Media
- Wire
Media component

- Memory, disk, cloud storage
- Introduce distinct delays
  - MQSim
- Parallel access
  - Contention delays
  - Queueing
- Only need to simulate delay
  - Not actual data delivery

![Diagram showing media component flow with Input, Wire, Cache, Wire, Cache, Wire, Media nodes, and connections for Hit and Miss paths.](image)
Media component

- Memory, disk, cloud storage
- Introduce distinct delays
  - MQSim
- Parallel access
  - Contention delays
  - Queueing
- Only need to simulate delay
  - Not actual data delivery

```
Input -> Wire -> Cache -> Wire -> Cache -> Wire -> Media
```

- Miss
- Hit
Media component

- Memory, disk, cloud storage
- Introduce distinct delays
  - MQSim
- Parallel access
  - Contention delays
  - Queueing
- Only need to simulate delay
  - Not actual data delivery

![Diagram showing the flow of data from Input to Media through Cache and Wire connections, including Miss and Hit pathways, with DRAM and SSD involved.](image-url)
Media component

- Memory, disk, cloud storage
- Introduce distinct delays
  - MQSim
- Parallel access
  - Contention delays
  - Queueing
- Only need to simulate delay
  - Not actual data delivery

Diagram:

- Input
- Wire
- Cache
- Wire
- Cache
- Wire
- Network Storage

Path:
- Hit -> Wire -> Cache -> Wire -> Cache -> Wire
- Miss -> Wire

Components:
- Origin
- DRAM
- SSD
- Flash
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing

Diagram:
- Input
- Wire
- Cache
- Wire
- Cache
- Wire
- Network Storage

Flow:
- Miss
- Hit
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing

![Diagram of wire component]

Input → URL → Cache → Wire → Cache → Wire → Network Storage

- Miss
- Hit

- Cache
- Wire
- DRAM
- SSD
- Flash

Origin
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing

![Diagram showing data flow through various components such as URL, Cache, PCI Bus, Cache, Wire, Network Storage, with input and output paths marked as Hit and Miss.](image-url)
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing
Wire component

- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing

Diagram:
- Input
- URL
- Cache
- PCI Bus
- Cache
- iSCSI
- Origin
- Network Storage

Nodes:
- Miss
- Hit

Devices:
- CXL
- DRAM
- NVMe
- SSD
- Flash

Connections:
- Input to URL
- URL to Cache
- Cache to PCI Bus
- PCI Bus to Cache
- Cache to iSCSI
- iSCSI to Origin
- Origin to Network Storage
Gas, Grass or Cache

No free ride with hierarchical memory
Cache component

- Easily build basics like lookups, allocation, and eviction
- One (or more) hit path
- One (or more) miss path
- Many choices for variability
Cache allocation

- Another building block
- Variable vs fixed
  - Object or Block
- Memory schemes
  - Slab, memalloc, persistent memory
- SSD fill buffers
Lookup

- Hashing and locating algorithm
- Miss algorithms
  - Trigger allocation or eviction
  - Trigger speculative fill
  - Pending misses
- Ageing
  - Dependent on eviction
- Element invalidation
Evictions

- Who to evict
- Culling invalidations
- SSD eviction
Duplicity is tricky

Two-tier cache issues
Multi-layer cache complexity

- All of these variables become a huge matrix with multiple layers
  - Cache algorithms
  - Media types
  - Interconnect type
  - Shared vs distributed
Simulation Summation

Modeling and collecting results for a two-layer cache
GUI demo

- Show the graphical representation of the two case configs
- Video to be added
UI demo

- Cache drilldown
Simulation code

- UI generated from code
- Code simulates component
Workloads matter

- No artificial workloads
- Content delivery
- Multiple sources
- (Need a CDN trace, not a syscall trace)
Matrix of results

- Number of PoPs
- Different L1 vs L2 algorithms
Matrix of results

- Number of PoPs
- Different L1 vs L2 algorithms

<table>
<thead>
<tr>
<th>Number of L1 caches</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of L2 caches</td>
<td>1</td>
<td>0.993</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.990</td>
<td>0.996</td>
<td>0.999</td>
<td>0.999</td>
<td>1.000</td>
<td>0.999</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.965</td>
<td>0.991</td>
<td>0.993</td>
<td>0.996</td>
<td>0.998</td>
<td>0.998</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.912</td>
<td>0.965</td>
<td>0.991</td>
<td>0.993</td>
<td>0.996</td>
<td>0.998</td>
<td>0.998</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.886</td>
<td>0.945</td>
<td>0.973</td>
<td>0.989</td>
<td>0.992</td>
<td>0.995</td>
<td>0.995</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.853</td>
<td>0.910</td>
<td>0.947</td>
<td>0.966</td>
<td>0.977</td>
<td>0.989</td>
<td>0.987</td>
<td>0.993</td>
<td>0.995</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.846</td>
<td>0.900</td>
<td>0.937</td>
<td>0.957</td>
<td>0.970</td>
<td>0.979</td>
<td>0.987</td>
<td>0.989</td>
<td>0.992</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.832</td>
<td>0.887</td>
<td>0.924</td>
<td>0.946</td>
<td>0.961</td>
<td>0.973</td>
<td>0.976</td>
<td>0.988</td>
<td>0.988</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.821</td>
<td>0.876</td>
<td>0.914</td>
<td>0.936</td>
<td>0.952</td>
<td>0.966</td>
<td>0.970</td>
<td>0.979</td>
<td>0.987</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.816</td>
<td>0.867</td>
<td>0.906</td>
<td>0.928</td>
<td>0.945</td>
<td>0.959</td>
<td>0.964</td>
<td>0.976</td>
<td>0.980</td>
</tr>
</tbody>
</table>

Miss Ratio
L1:urlhashLB, LRU; L2:urlhashLB, LRU
Heat map

- Ability to compare sets of results
Solving the Halting Problem

Wrap up and conclusions
As an example, a current customer has achieved the following measurable outcomes with Magnition:

Experiments per day per engineer:
- Without Magnition: 2
- With Magnition: 50,000+

Parameter variations tested before prod release:
- Without Magnition: 50
- With Magnition: 1,000,000+

Workload performance improvement using our products to find optimal out-of-the-box settings: 10-50%+
Today I Learned

1. Complex systems can be modularized rapidly into simulated components
2. Real-world problems can be analyzed efficiently using simulations
3. Modern simulators allow faster and more thorough cost and performance analysis than direct experimentation
Please take a moment to rate this session.

Your feedback is important to us.
Section Title

Section Subtitle
Light Slide Title

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
        - Bullets 5
Dark Slide Title

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
      - Bullets 5