STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

Optimizing Complex Hierarchical Memory Systems Through Simulations

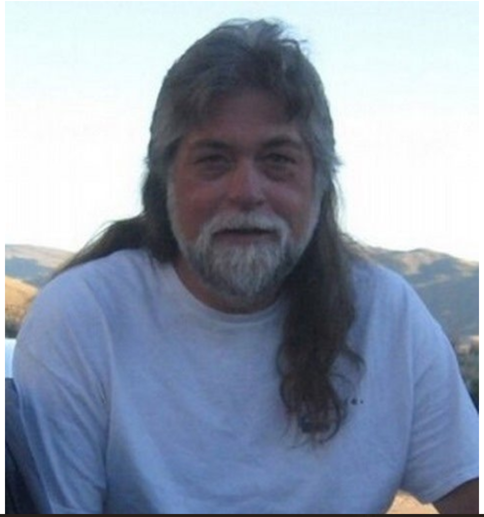
Andy Banta

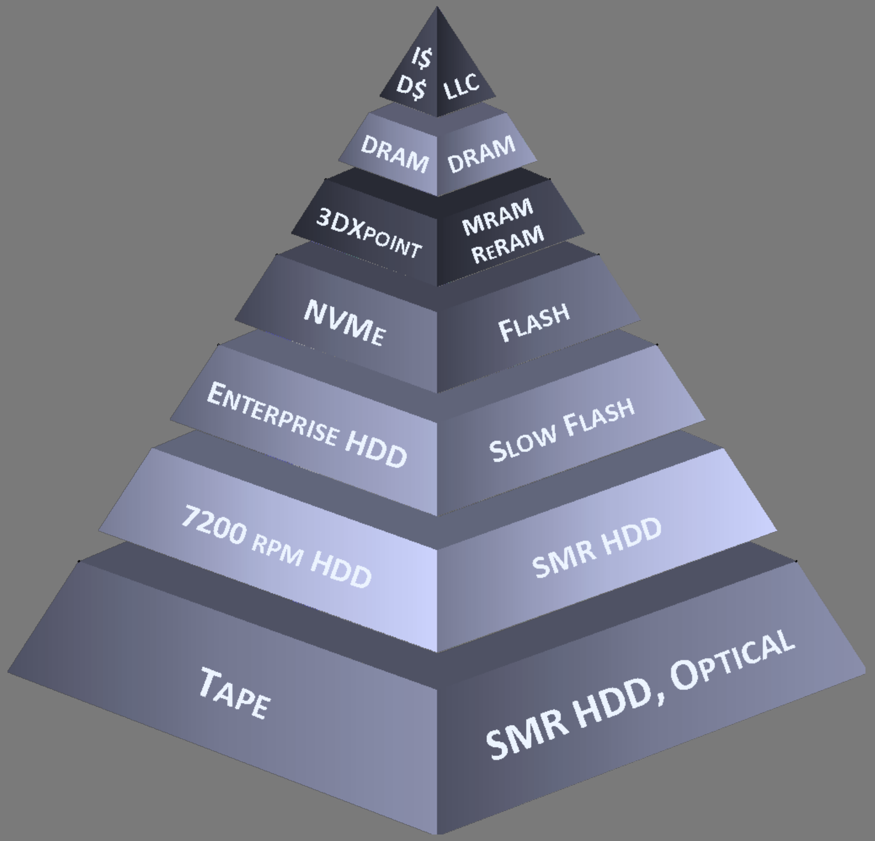
Storage Janitor, Magnition Inc

@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

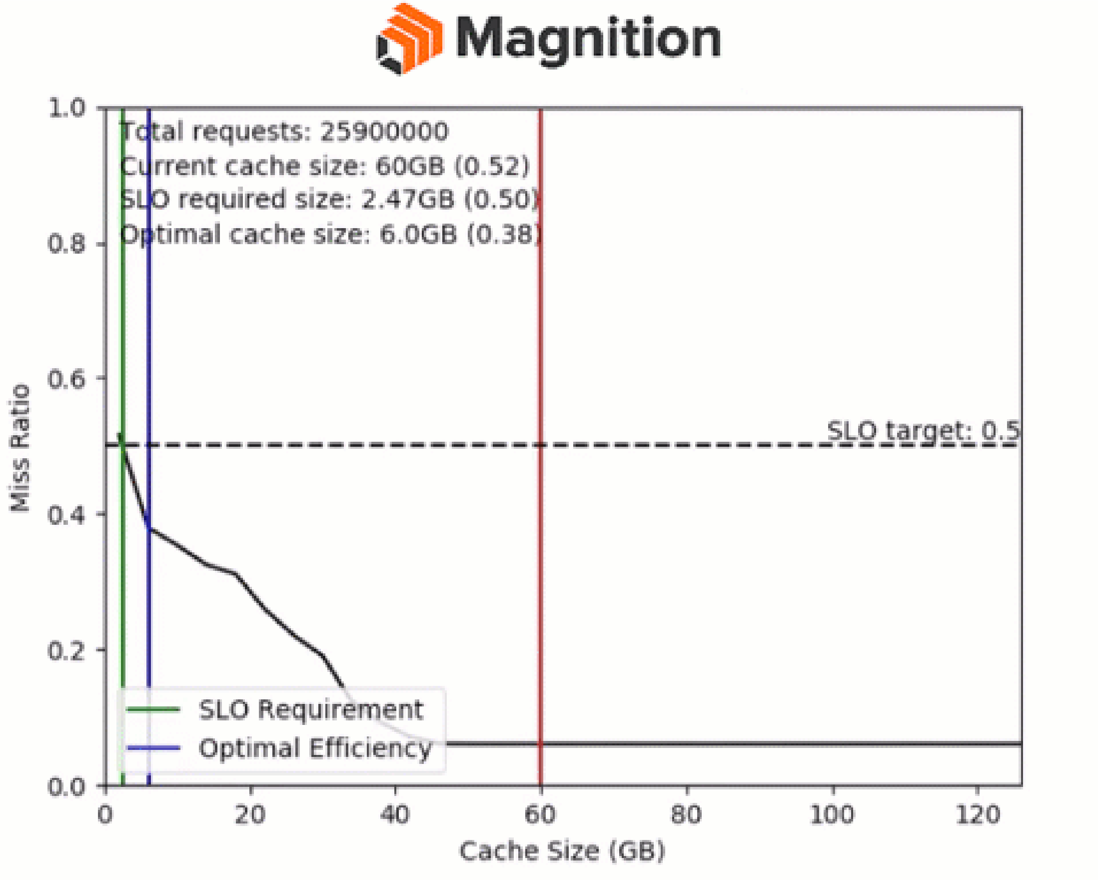
Our Approach to Simulations

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

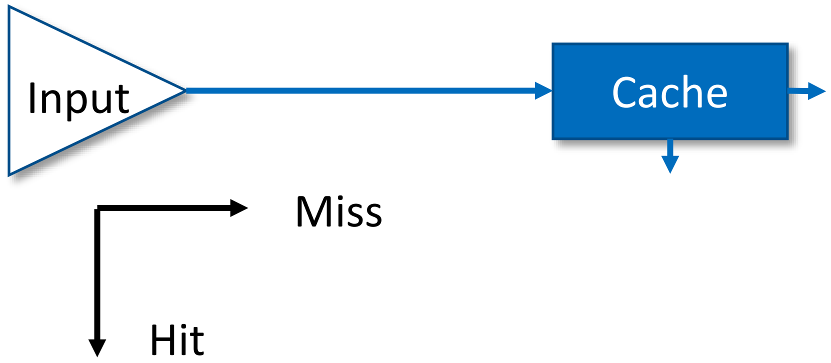
- Provide a framework to connect components
  - Lingua Franca provides this
  - Reactors represent system pieces
- Library of components ready to use
- Allows clients to build their own modules
- Basic set of building blocks
  - Cache
  - Media
  - Wire

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

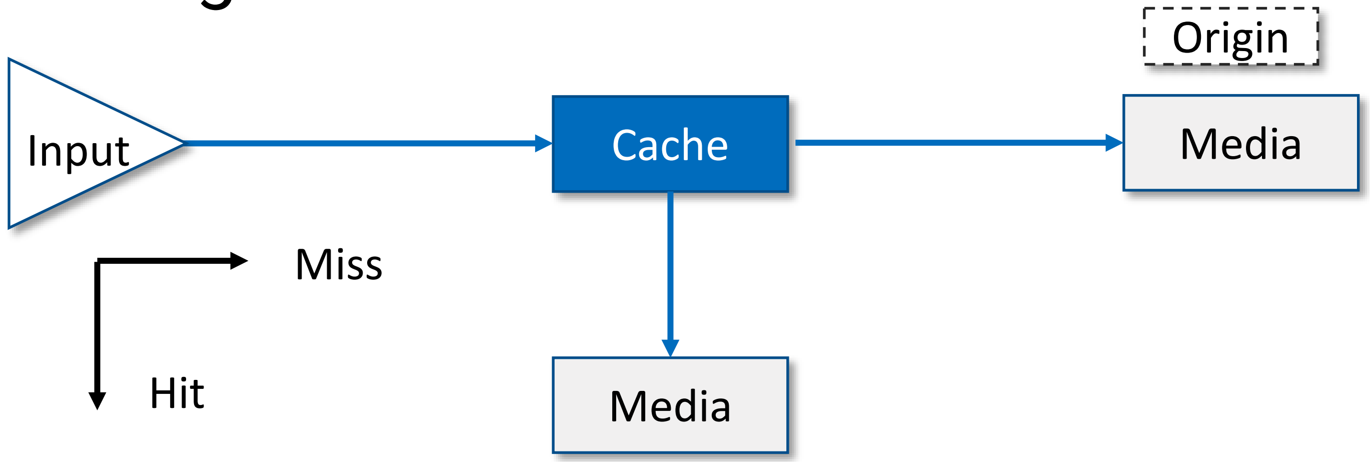
Composable components

- 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



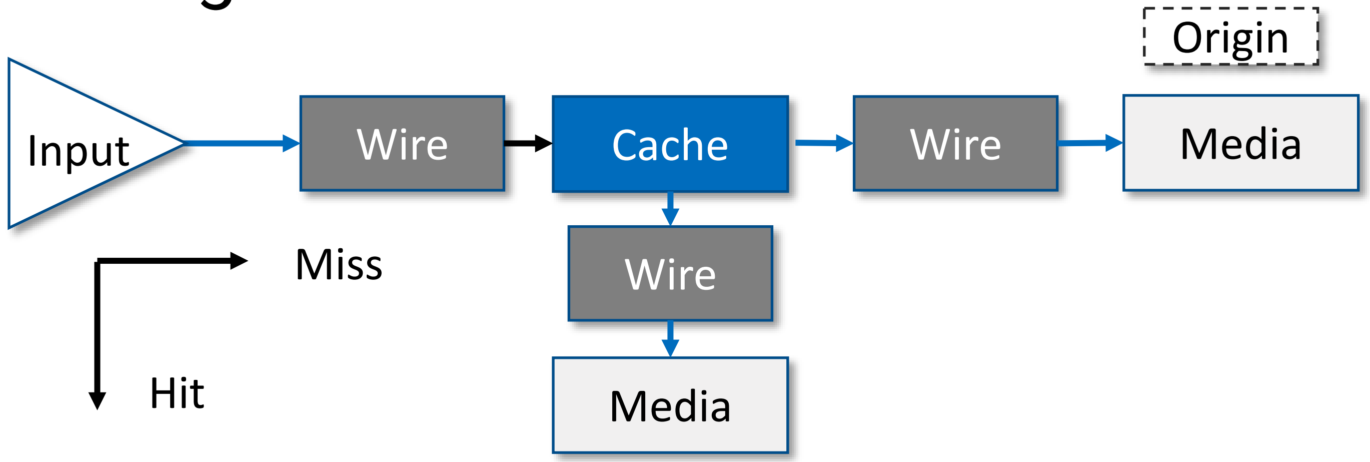
Composable components

- 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



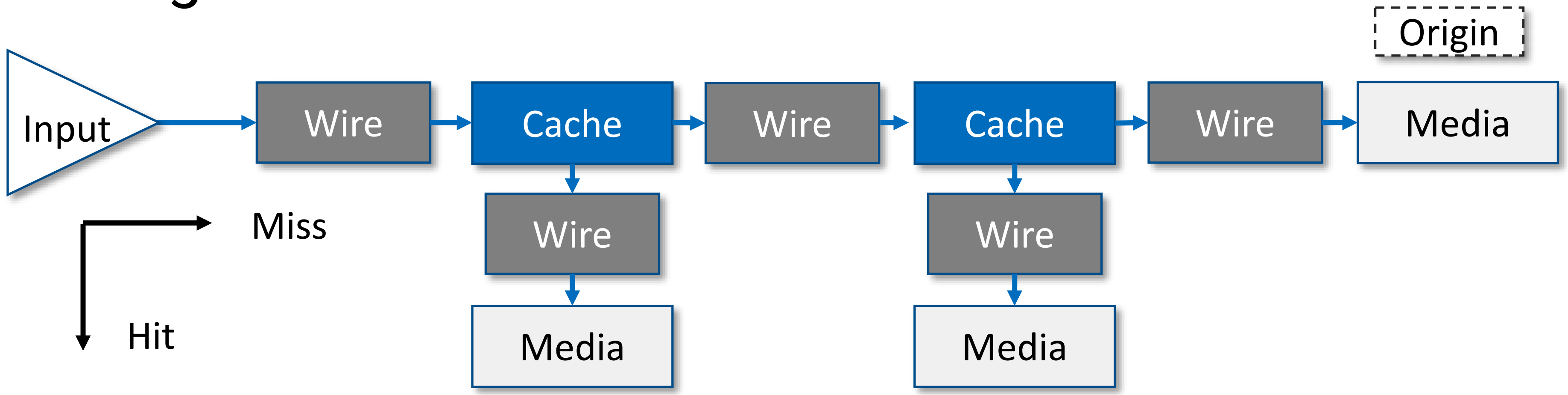
Composable components

- 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



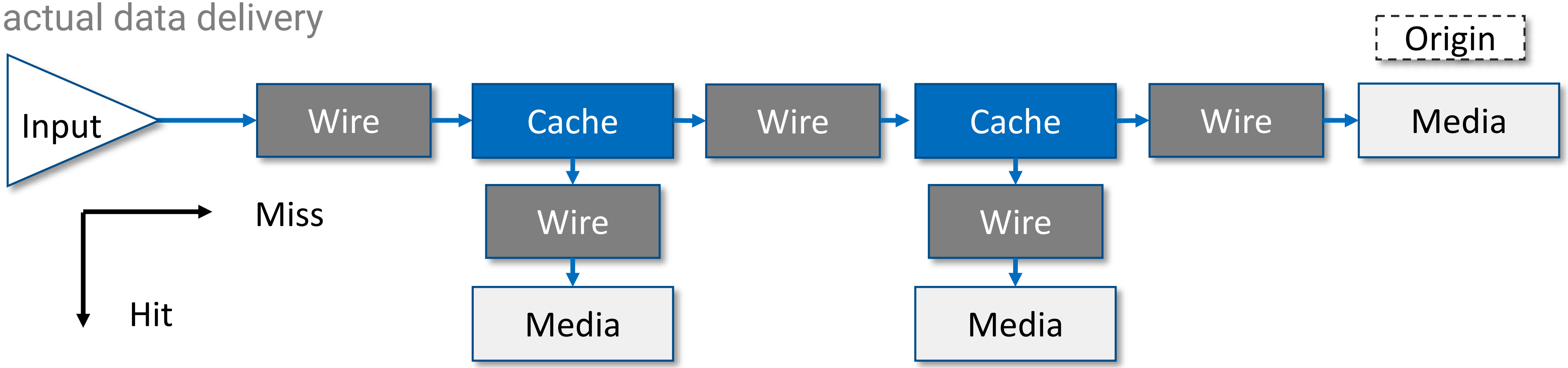
Composable components

- 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



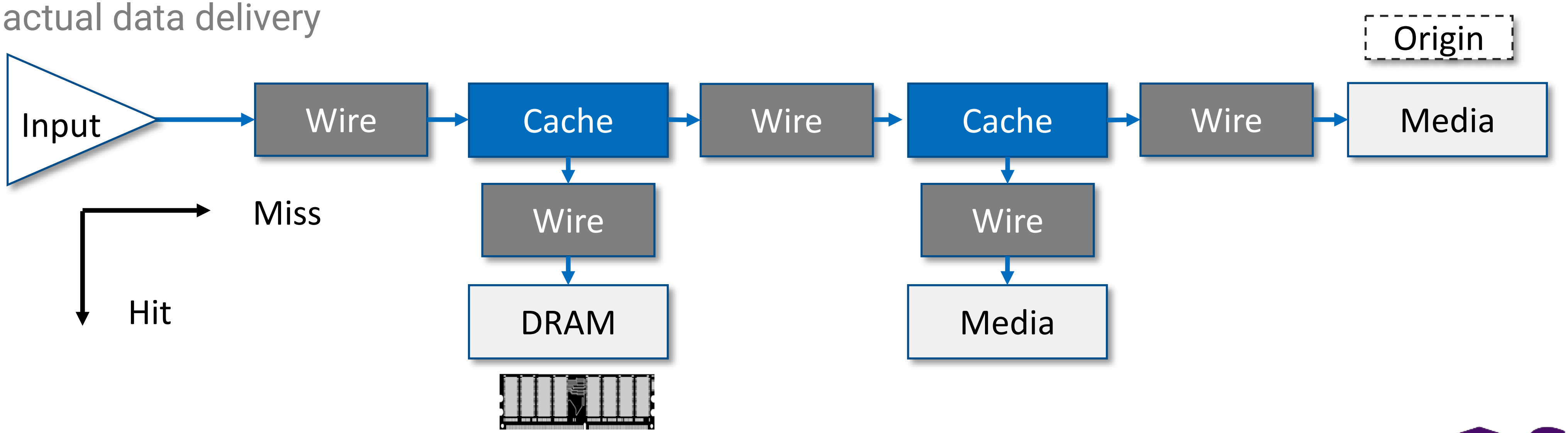
Media component

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



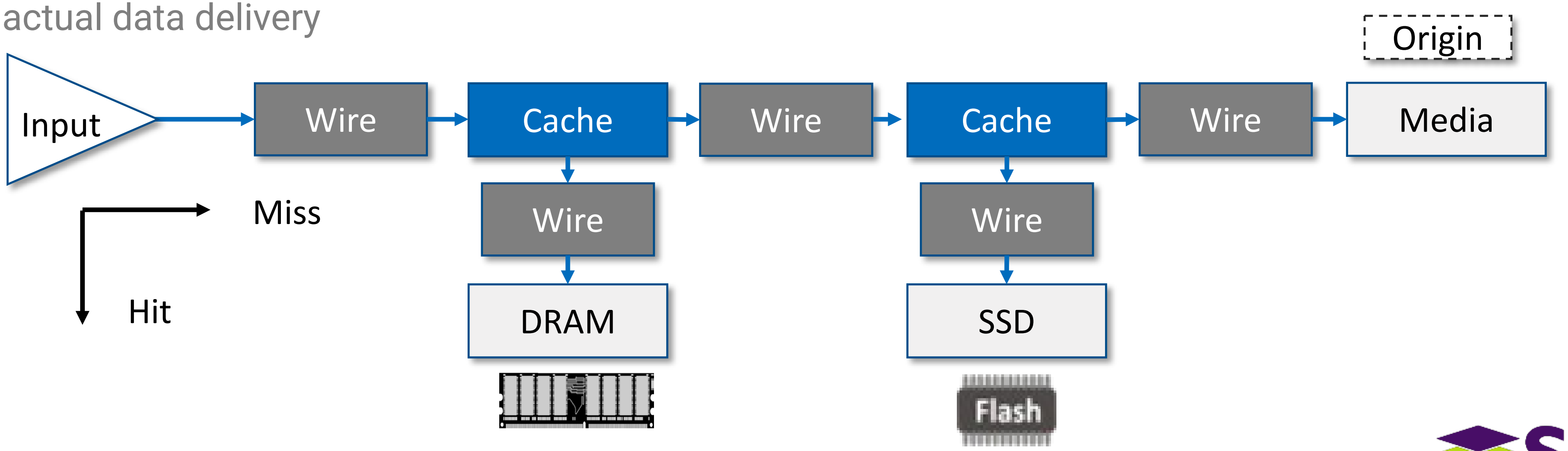
Media component

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



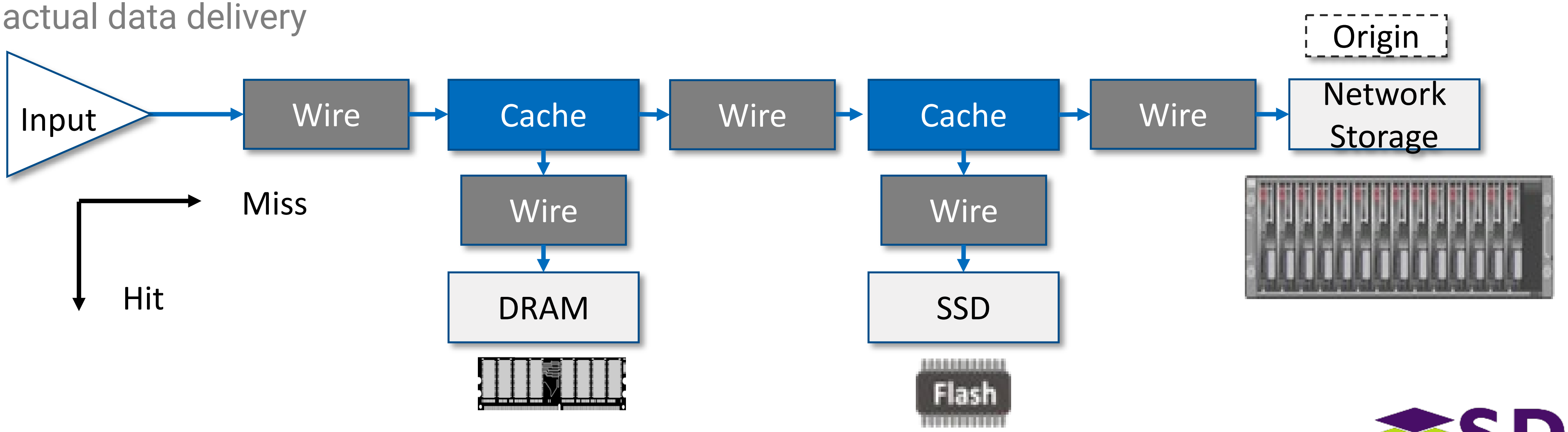
Media component

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



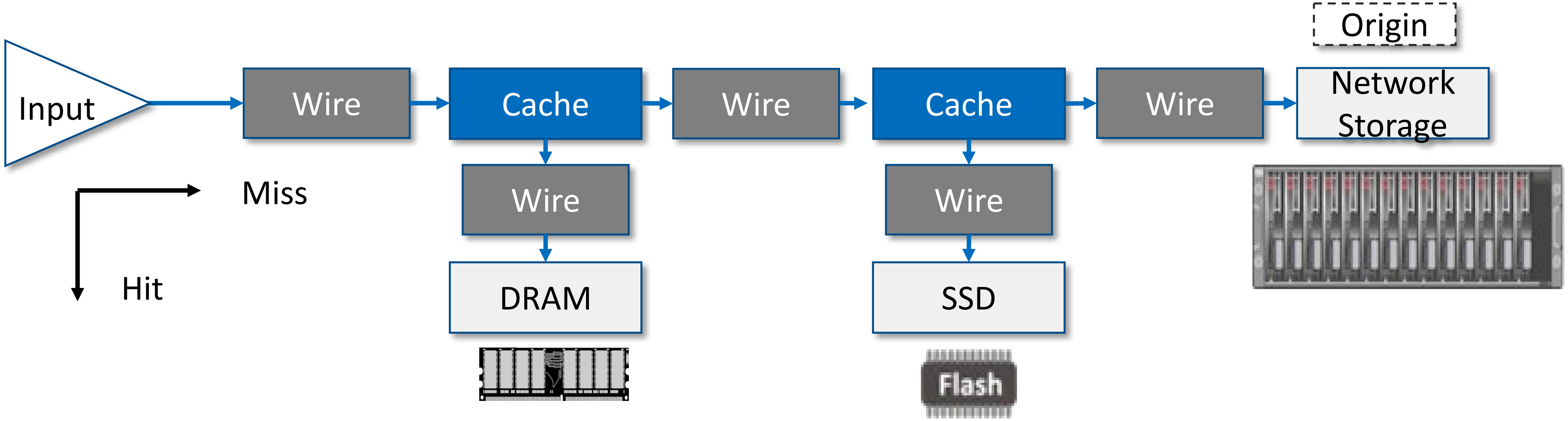
Media component

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



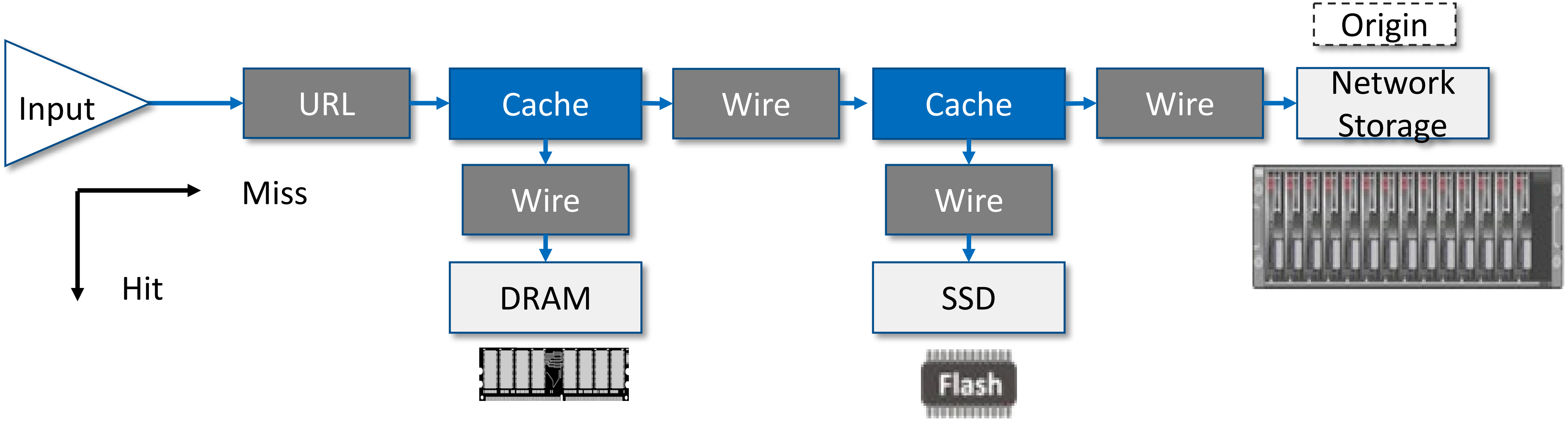
Wire component

- 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



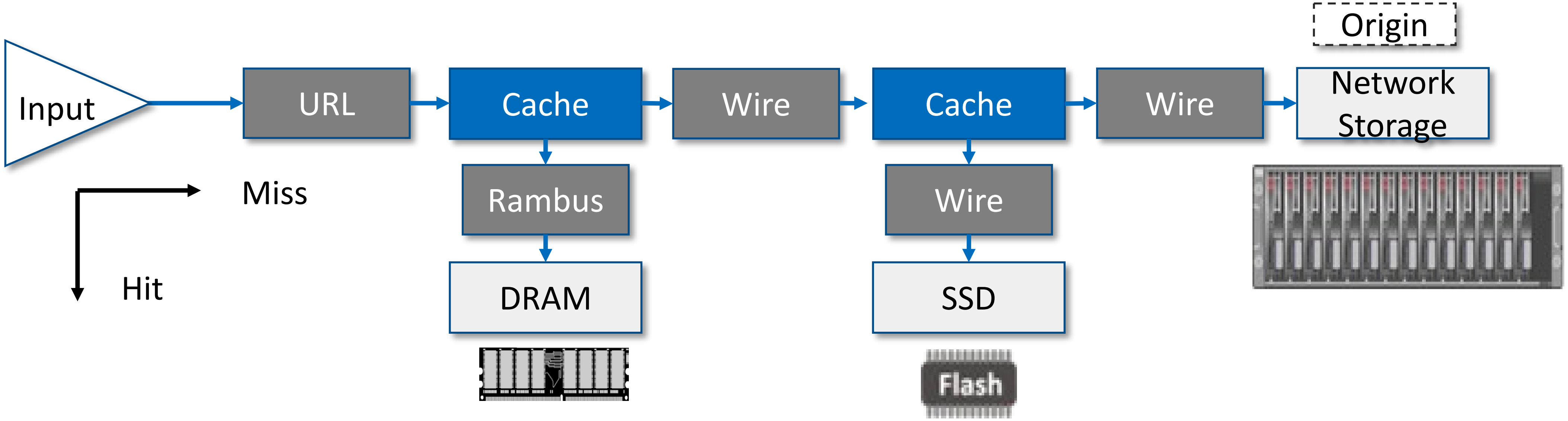
Wire component

- 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



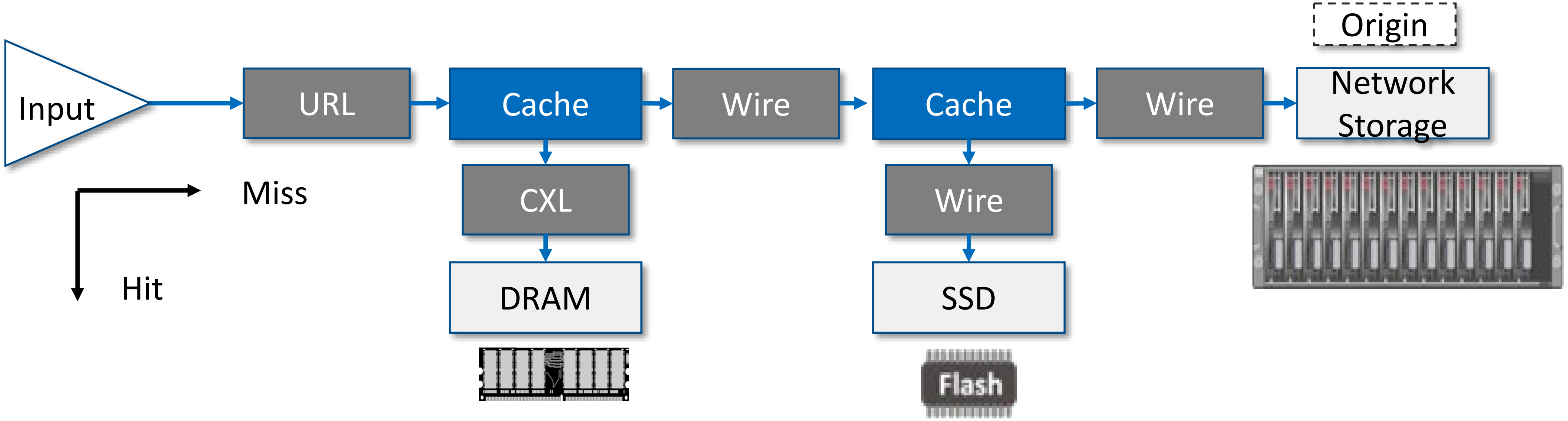
Wire component

- 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



Wire component

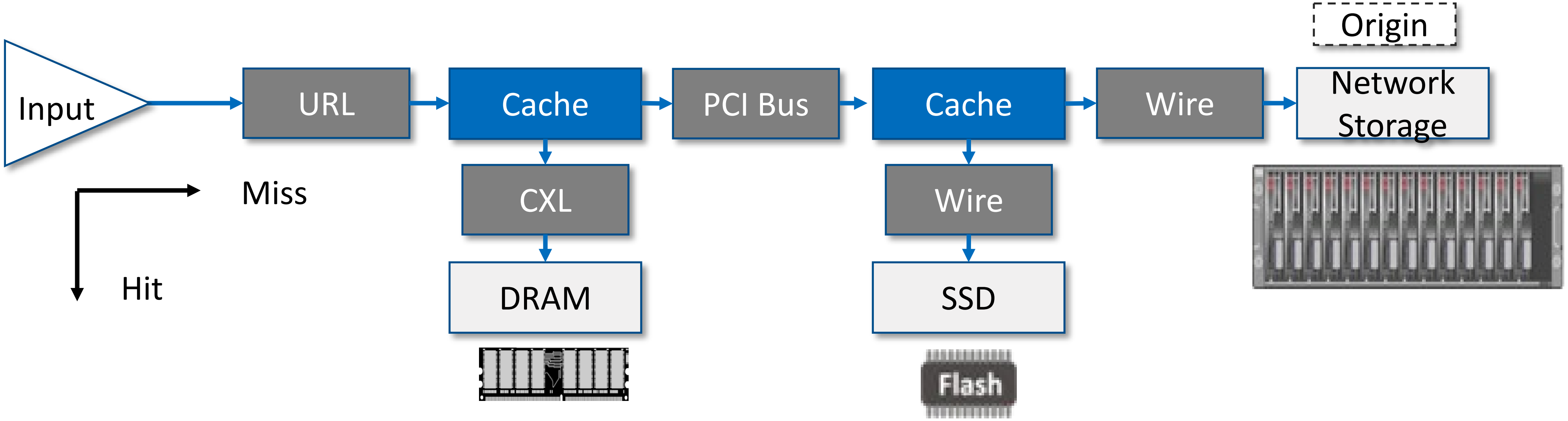
- Memory bus, disk controller, network
- Can multiplex and change form of IO request
- Even type of wire can be variable
  - Type of memory bus
  - Hops in network topology
- Delays introduced by wire, contention, queueing



 |

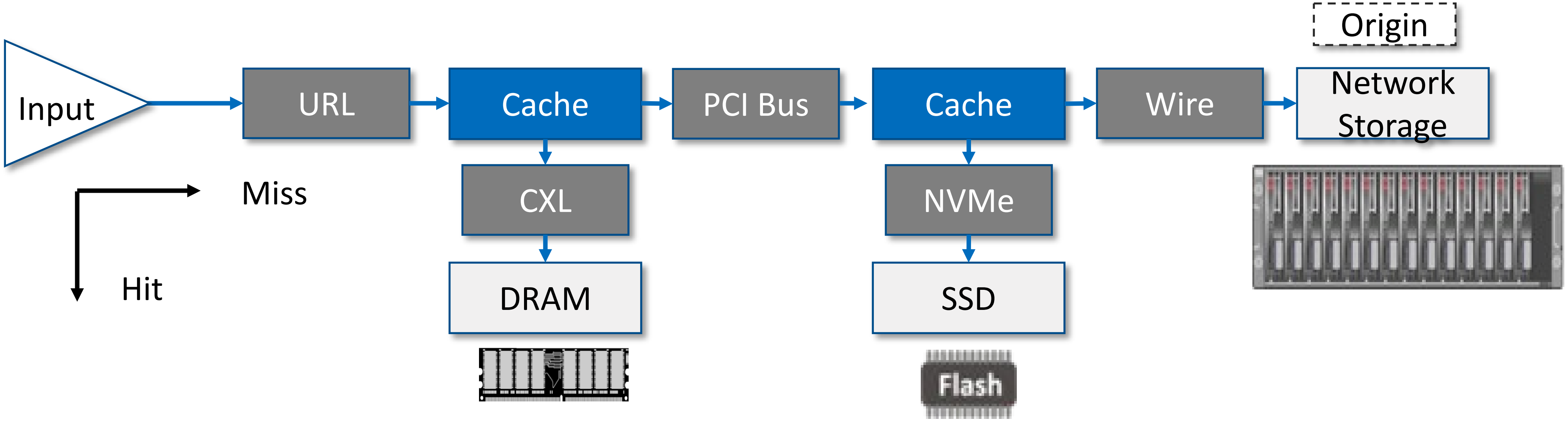
Wire component

- 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



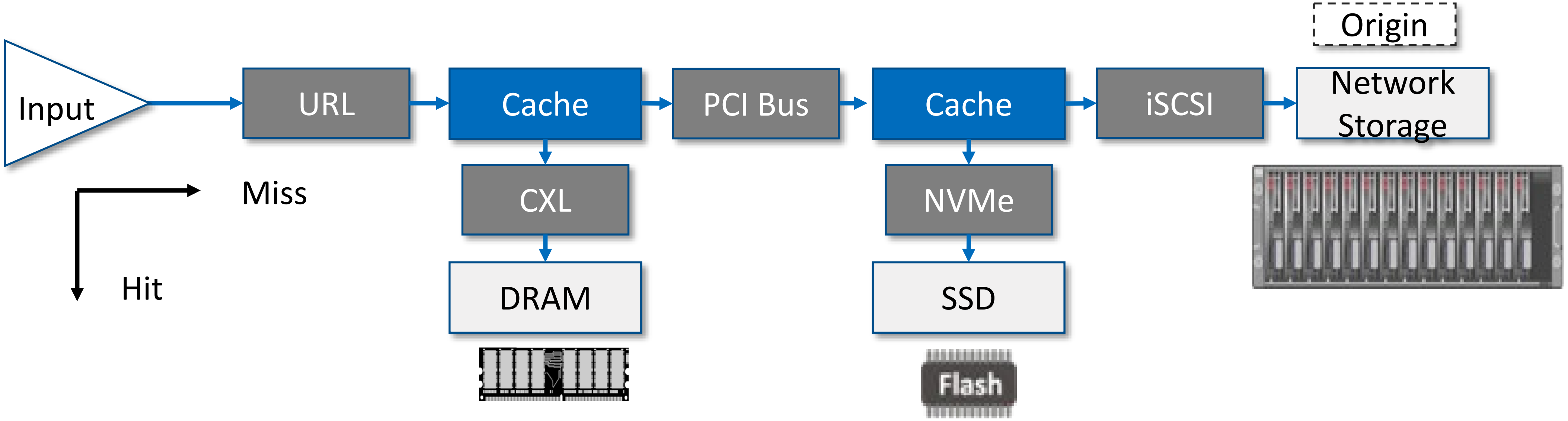
Wire component

- 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



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



 |

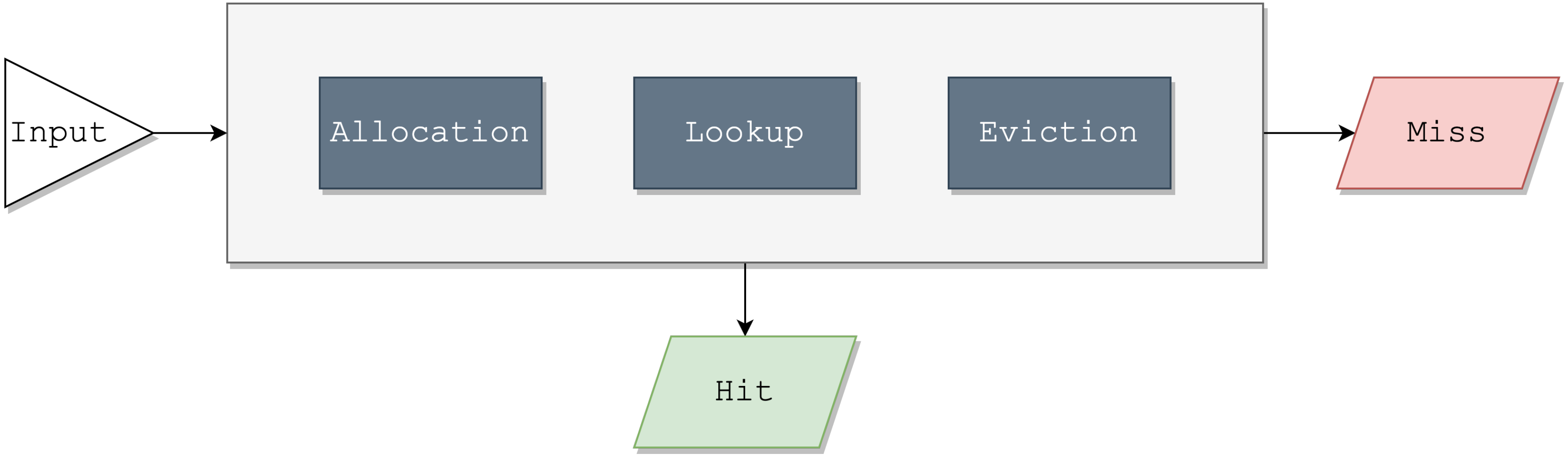
Gas, Grass or Cache

No free ride with hierarchical memory

SDC 23

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

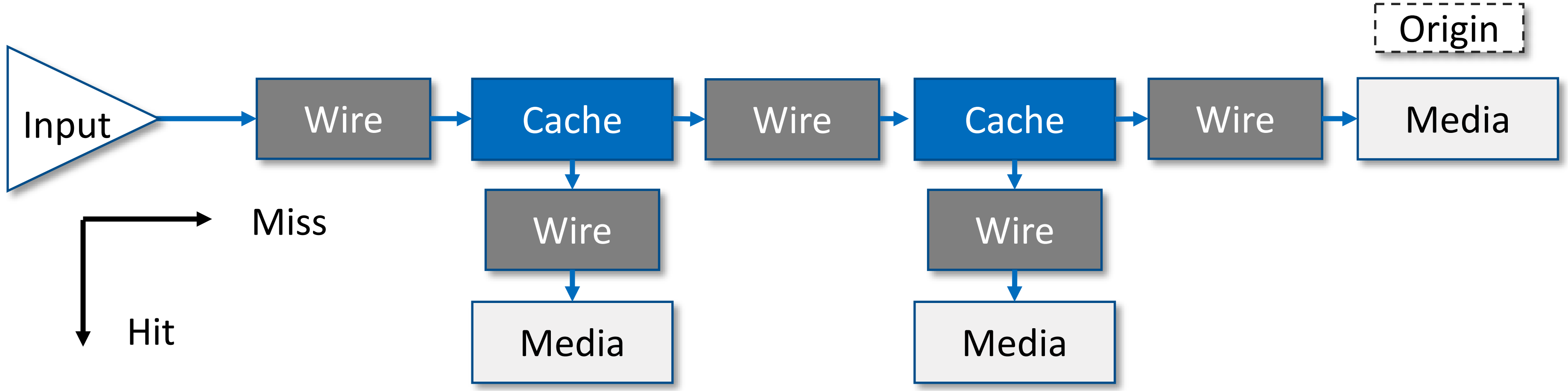
- Who to evict
- Culling invalidations
- SSD eviction

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

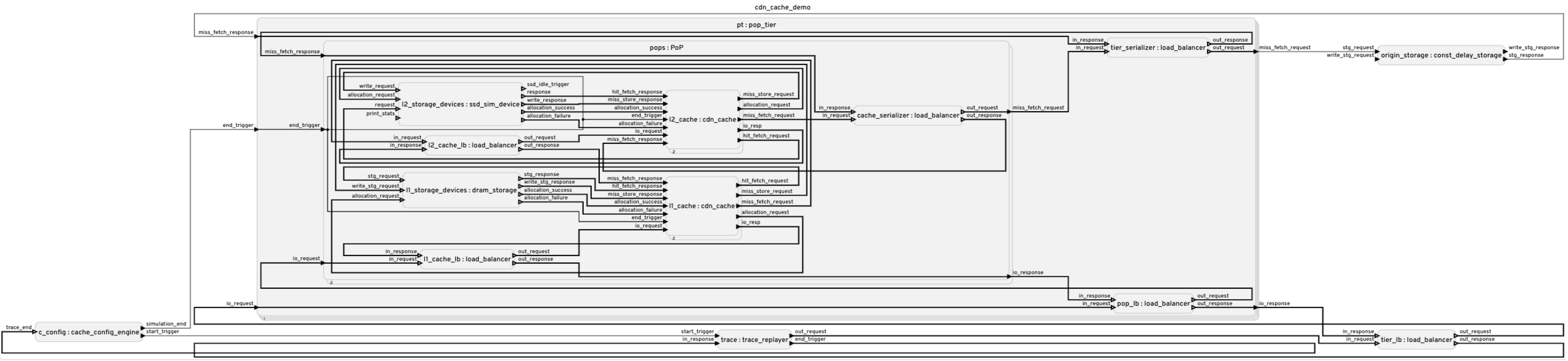


Simulation Summation

Modeling and collecting results for a two-layer cache

GUI demo

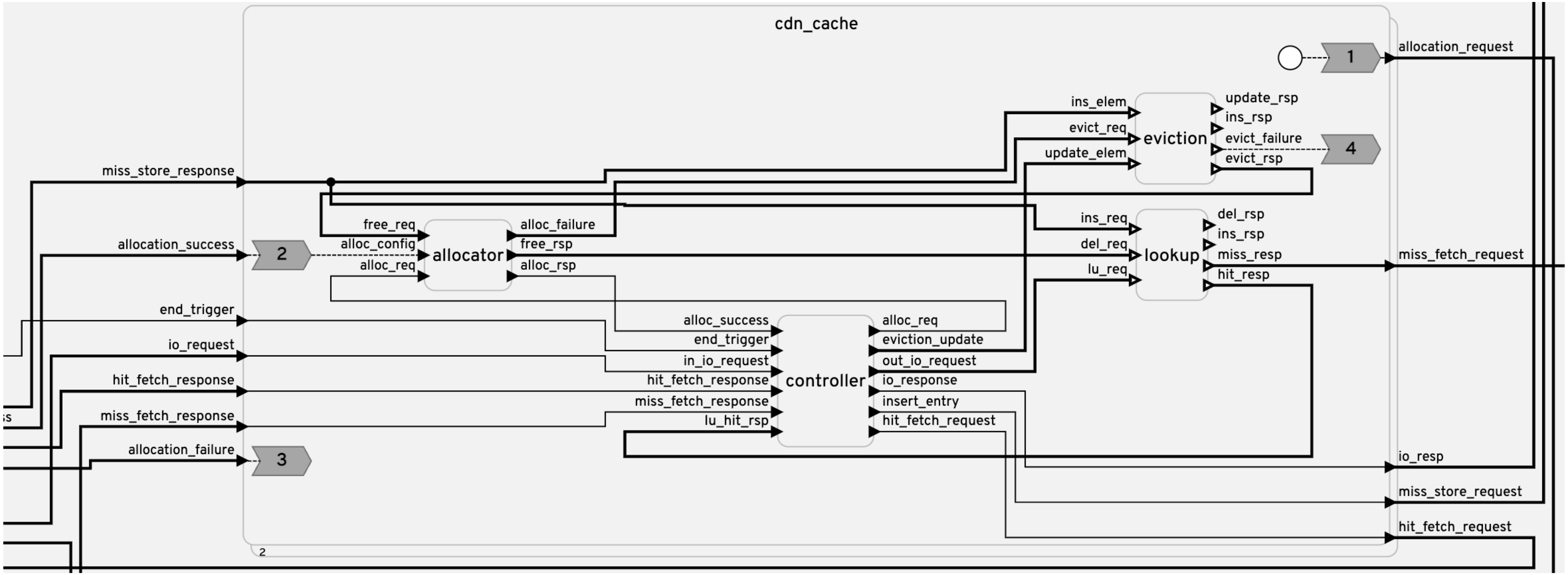
- Show the graphical representation of the two case configs
  - Video to be added



 |

UI demo

- Cache drilldown



Simulation code

- UI generated from code
- Code simulates component

```
reactor cdn_cache (bank_index:int(0), pop_tier_id:int(0), pop_id:int(0), n_ports:int(1), cache_level:string("L1"), cache_size:uint32_t(4096), page_size:uint32_t(
    input io_request:cdn_cache_request_t;
    output io_resp:cdn_cache_response_t;
    output hit_fetch_request:cdn_cache_request_t;
    input hit_fetch_response:cdn_cache_response_t;
    output miss_fetch_request:cdn_cache_request_t;
    input miss_fetch_response:cdn_cache_response_t;
    output miss_store_request:p_cdn_cache_entry_t;
    input miss_store_response:p_cdn_cache_entry_t;
    output allocation_request:uint32_t;
    input allocation_success:uint64_t;
    input allocation_failure:int;
    input end_trigger:uint32_t;
    logical action sch_response(0):cdn_cache_response_t;
    logical action sch_eviction(0):cdn_cache_request_t;
    logical action sch_insertions(0):cdn_cache_entry_t*;
    state free_space:uint32_t(0);

    CacheCtrl = new controller<int> (cache_level = cache_level, name = "cache_controller", pop_tier_id = pop_tier_id, pop_id = pop_id, cache_id = bank_index, log
    LookUp = new lookup<cdn_cache_request_t, p_cdn_cache_entry_t, cdn_response_tuple_t> (n_ports = 1, log_level = {=LOG_DEBUG_LEVEL=});
    Eviction = new eviction<p_cdn_cache_entry_t, cdn_cache_request_t, cdn_response_tuple_t> (evict_methods = {=&lru_eviction_methods=}, n_ports = 1, eviction_typ
    Allocator = new allocator<cdn_cache_request_t, p_cdn_cache_entry_t, cdn_response_tuple_t> (log_level = {=LOG_DEBUG_LEVEL=});

    reaction (startup) -> allocation_request {=
        self->free_space = self->cache_size - (self->cache_size % self->page_size);

        LOG_INFO (self->log_level, "(%lld, %u) physical_time:%lld "
                    "cdn_cache_%d_%d %s_%d startup paged_cache_size:%u",
                    lf_time_logical_elapsed(), lf_tag().microstep, lf_time_physical_elapsed(),
                    self->pop_tier_id, self->pop_id, self->cache_level, self->bank_index, self->free_space
        );

        lf_set (allocation_request, self->free_space);
    =}
```

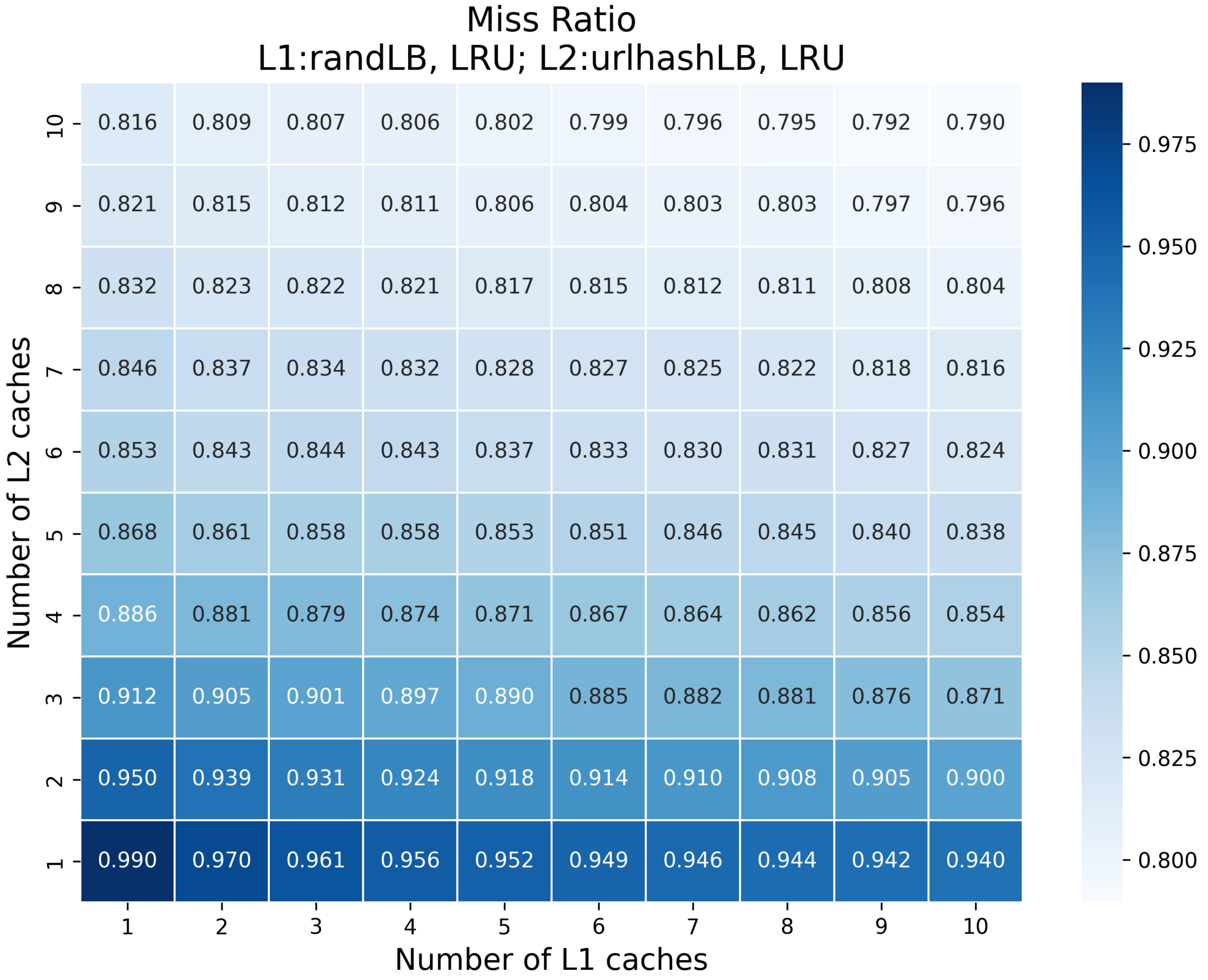

Workloads matter

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

```
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
poll([{fd=61, events=POLLIN}], 1, 3000) = 0 (Timeout)
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
semop(8126470, [{0, -1, SEM_UNDO}], 1)  = 0
semop(8126470, [{0, 1, SEM_UNDO}], 1)   = 0
poll([{fd=61, events=POLLIN}], 1, 3000^Cstrace: Process 14046 detached
```

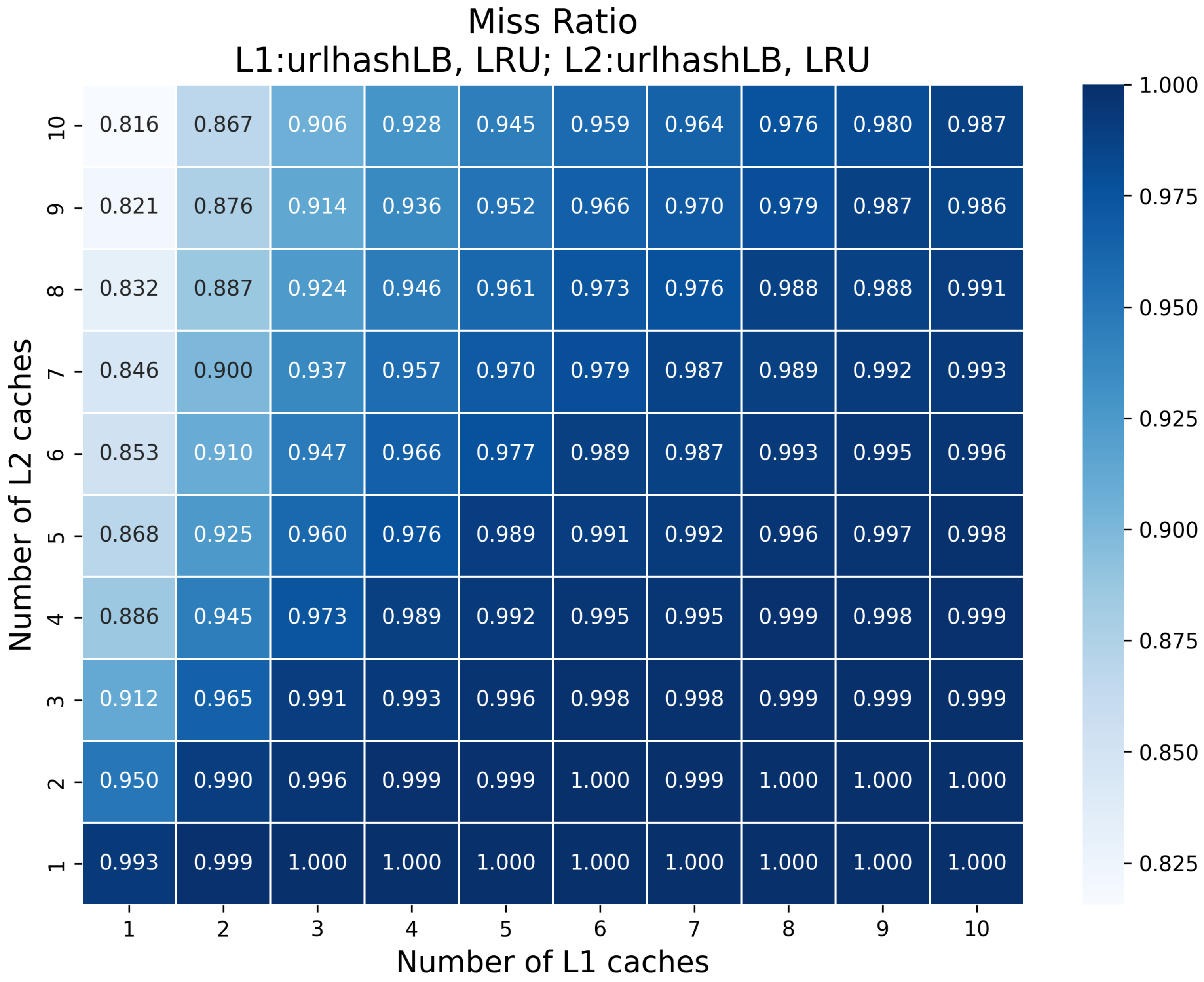

Matrix of results

- Number of PoPs
- Different L1 vs L2 algorithms



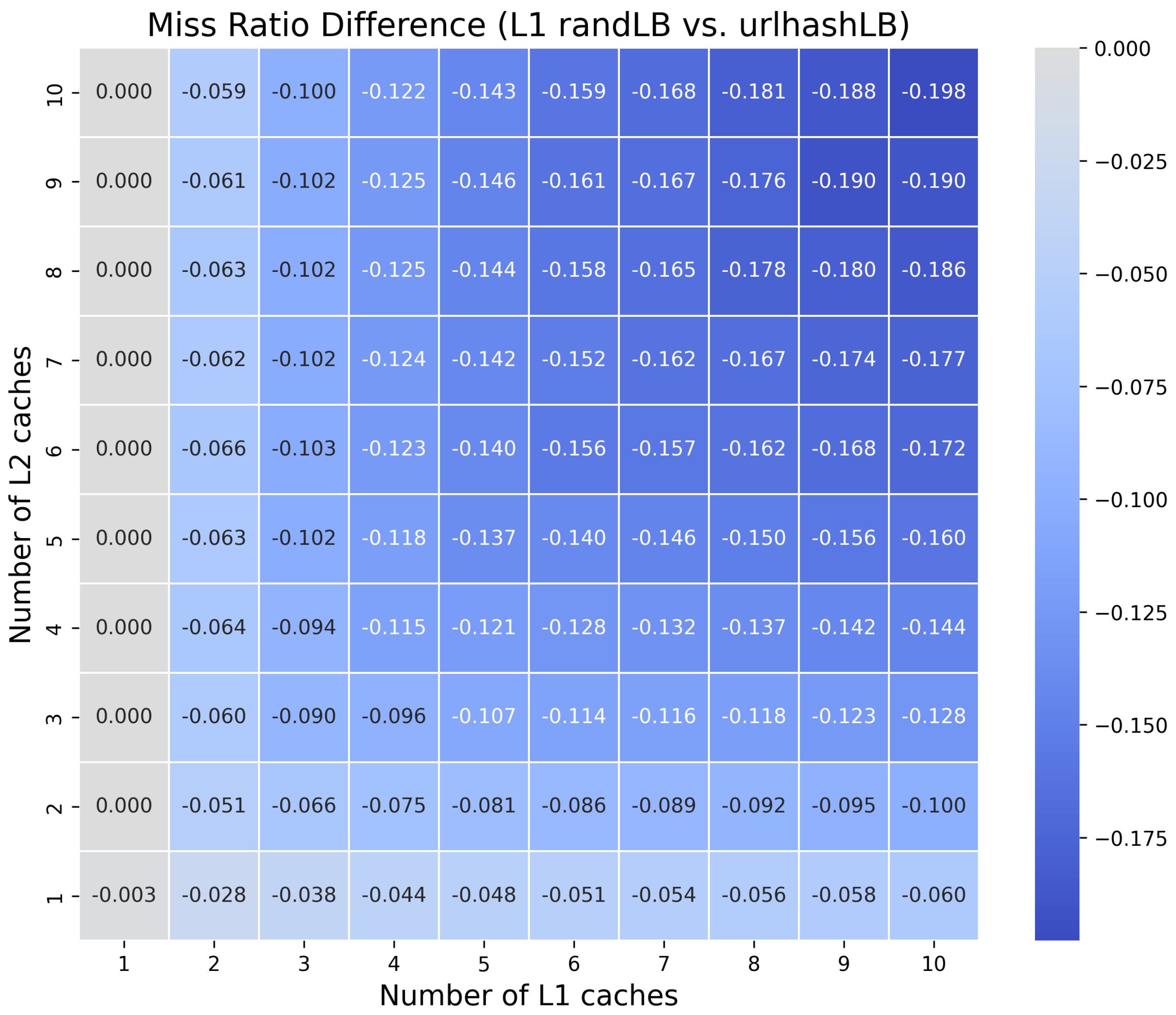
Matrix of results

- Number of PoPs
- Different L1 vs L2 algorithms



Heat map

- Ability to compare sets of results



Solving the Halting Problem

Wrap up and conclusions

RESULTS WITH MAGNITION

PROVEN IN MARKET TODAY

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

Section Title

Section Subtitle

SDC 23

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

