SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

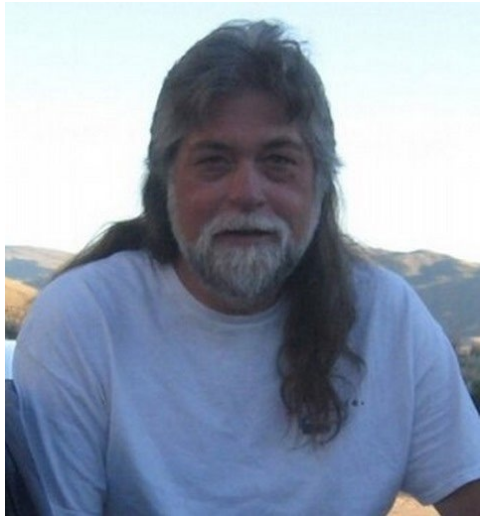
September 16-18, 2024
Santa Clara, CA

Exploring Optimizations of Content Delivery Networks

Andy Banta – Storage Janitor – Magnition IO

Andy Banta

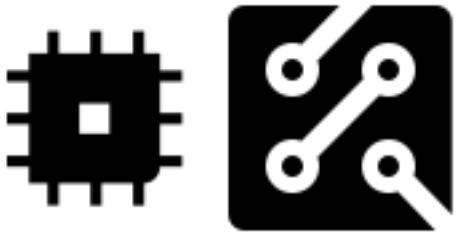
Magnition.io (Consultant)
SolidFire (VMware development, acq. by NetApp)
DataGravity (Container exploitation lead)
VMware (iSCSI Tech Lead, IPO)
Sun Microsystems (Initial Fibre Channel development)
Patent, early distributed network projects, data acquisition
@andybanta



Magnition

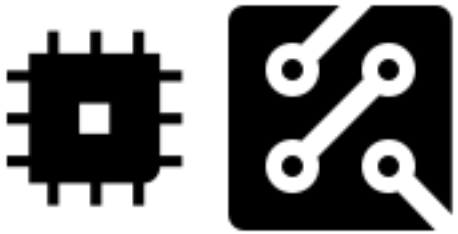
SDC 24

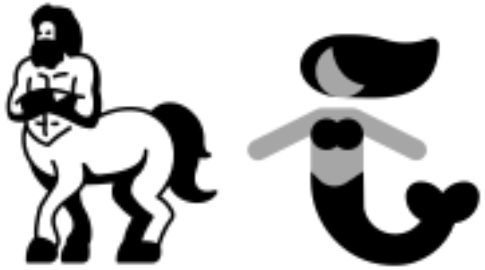
Engineering Simulation



Engineering Simulation

- Cheaper, faster, more flexible than system building
- Engineering design uses simulations, why not software?





How to Optimize All These Factors

- 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

Caching

- Hit-n-miss
- Promotion and demotion
- Complexity of tiering
  - How rapidly this becomes an unmanageable problem

Caching

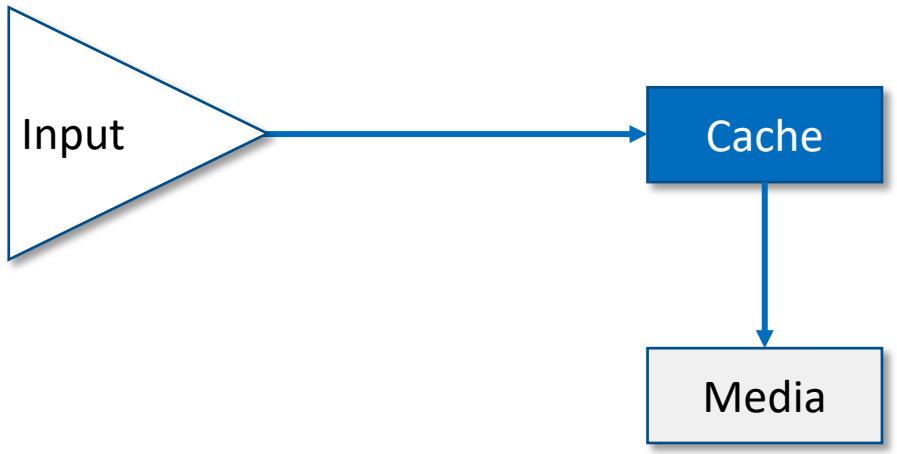
- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload

Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload

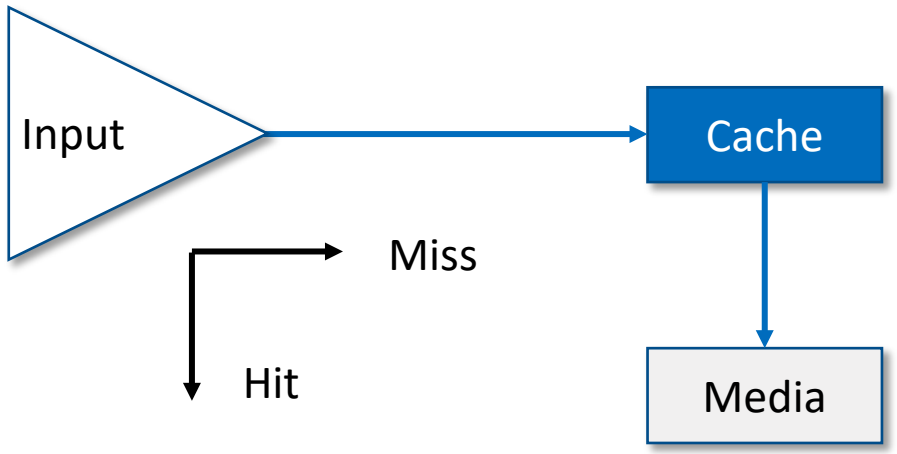
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload



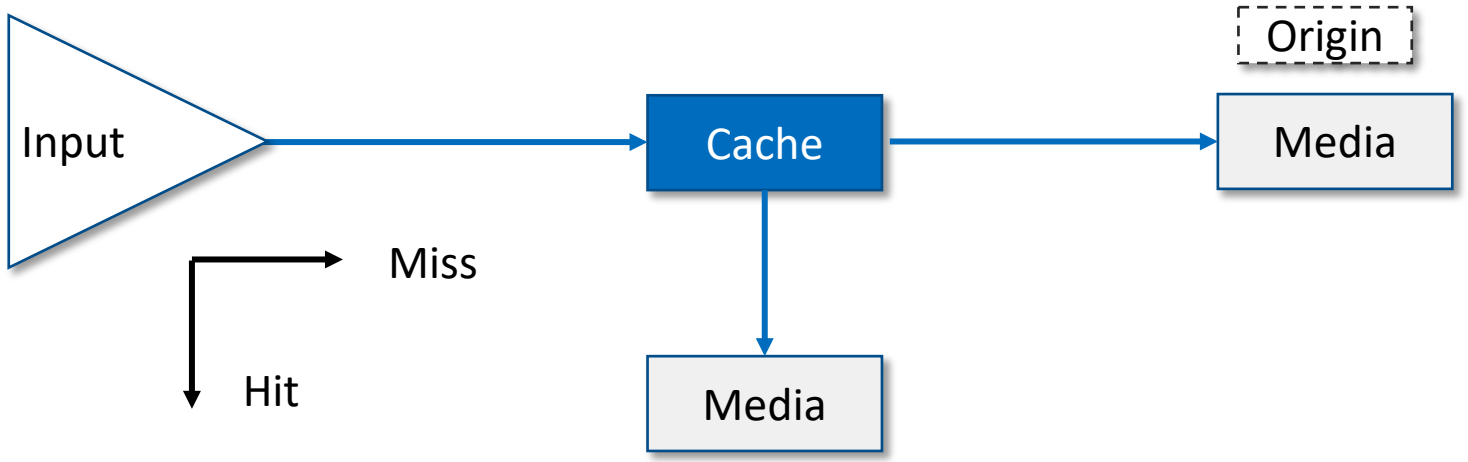
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload



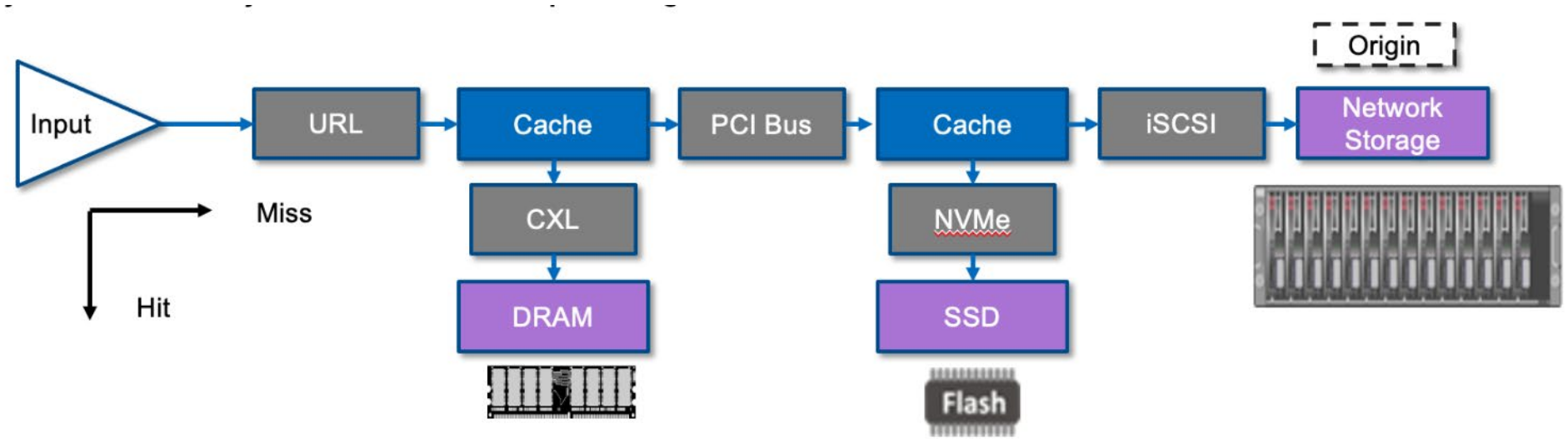
Caching

- Hit or miss
  - Populate based on use or prediction
  - Variety of algorithms for lookup, allocation, eviction and aging
  - Can be tuned for workload

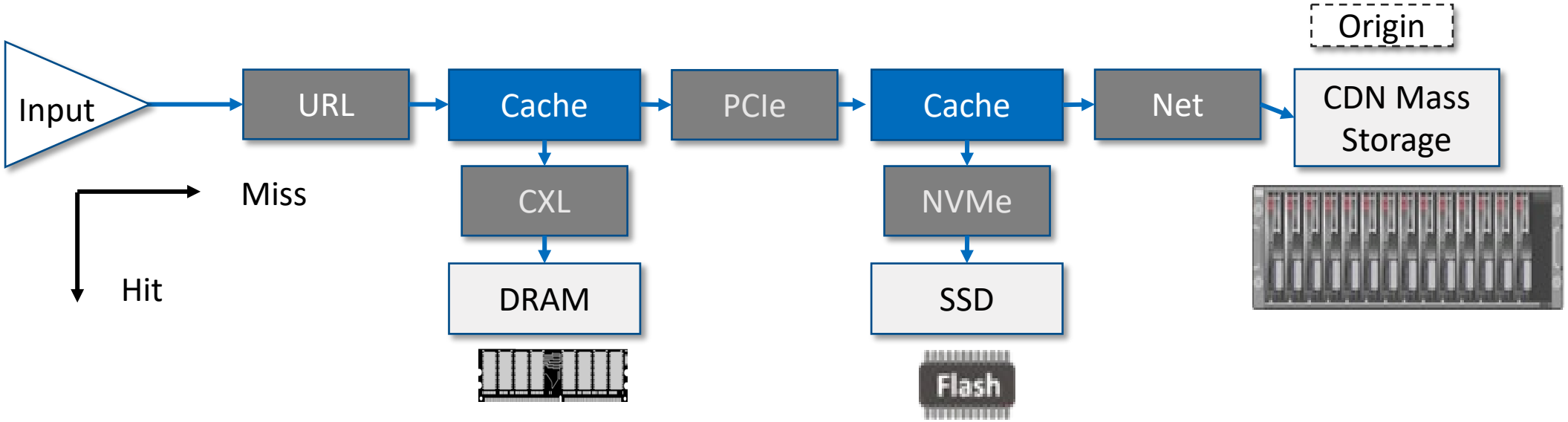


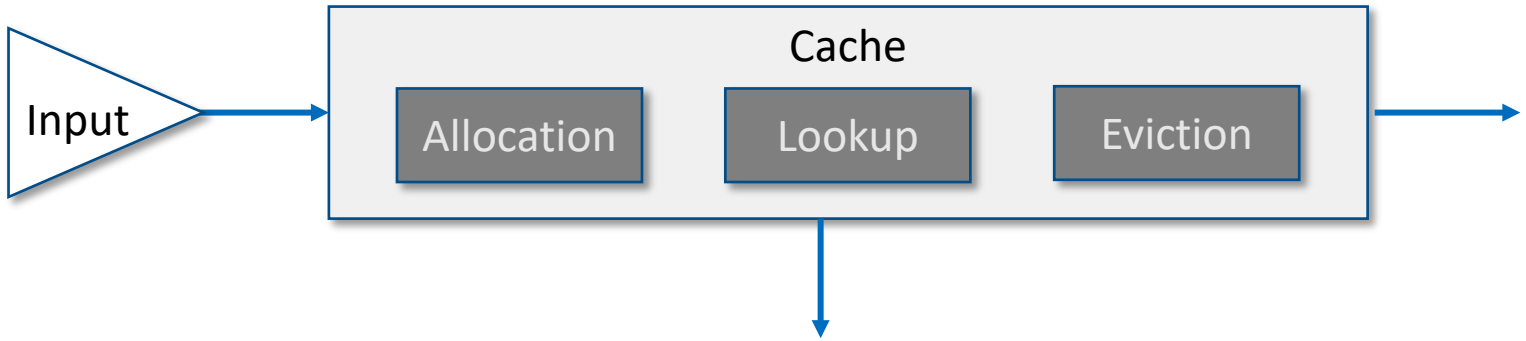
Multi-level Caching

- Currently used in Content Delivery Networks



Deterministic Behavioral Simulations





Time to Kick Out Old Eviction Algorithms

Introducing SIEVE

SIEVE: a New Eviction Algorithm

- Designed by Magnition intern
- Simple
- Efficient
- Improves scale and throughput

SIEVE operation

- Singly-linked list for eviction
- Pointer (“hand”) moves from tail to head, marks entries as unref’ed
- On cache hit, marks entry as referenced

An Example of SIEVE

SIEVE Operation

- On cache miss
  - Hand moves forward to next unreferenced entry and deletes it
  - New entry is added to the head of the list

An Example of SIEVE

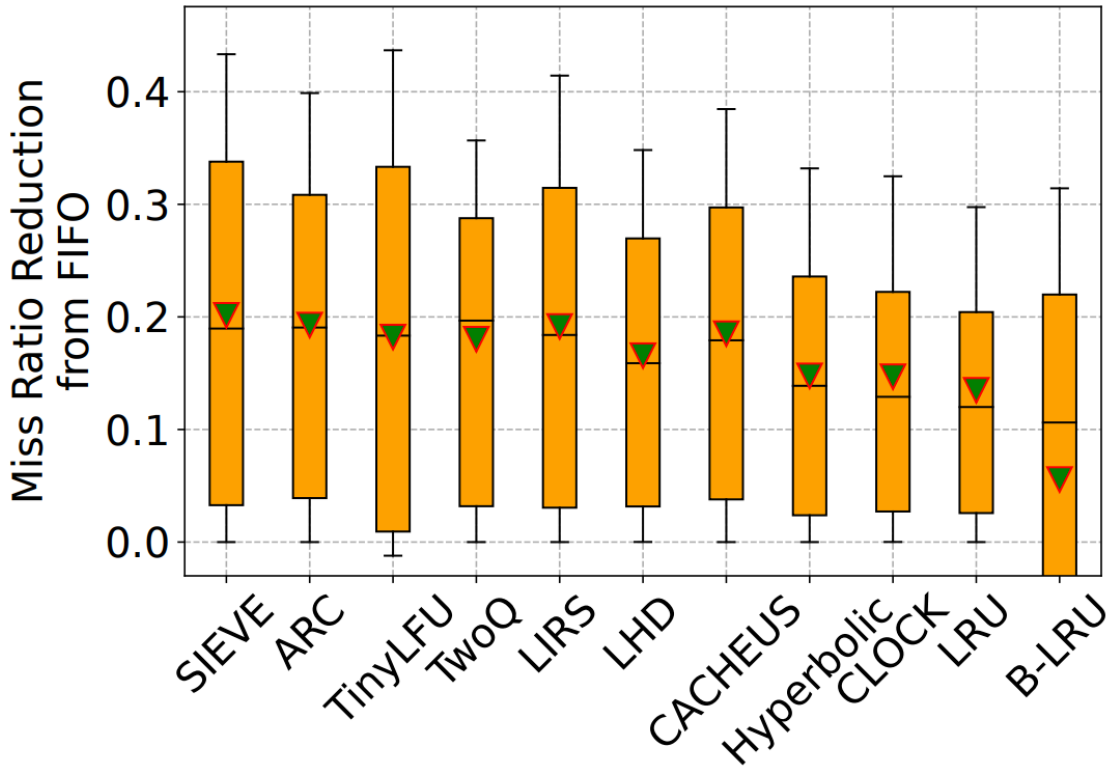
SIEVE Operation

- If hand reaches head, it moves to tail

An Example of SIEVE

SIEVE Efficiency

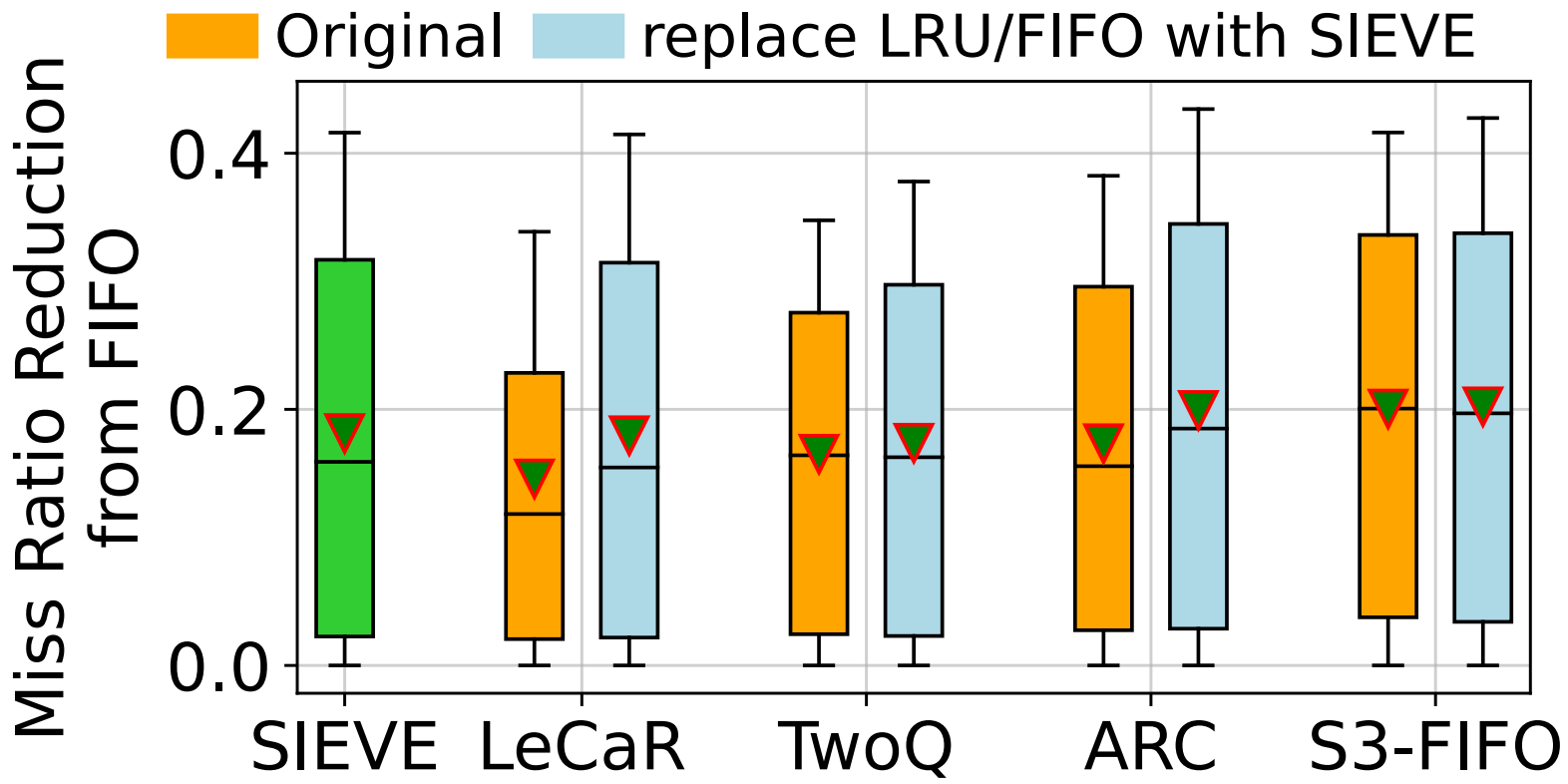
- Lower miss ratio in a variety of different applications



(a) CDN1 workloads, large cache, 1273 traces

SIEVE Efficiency

- SIEVE increases efficiency of more elaborate eviction algorithms



SIEVE Simplicity

- Modifications to existing algorithms are minimal
- Complete implementation is compact
  - Python sample implementation is 71 lines

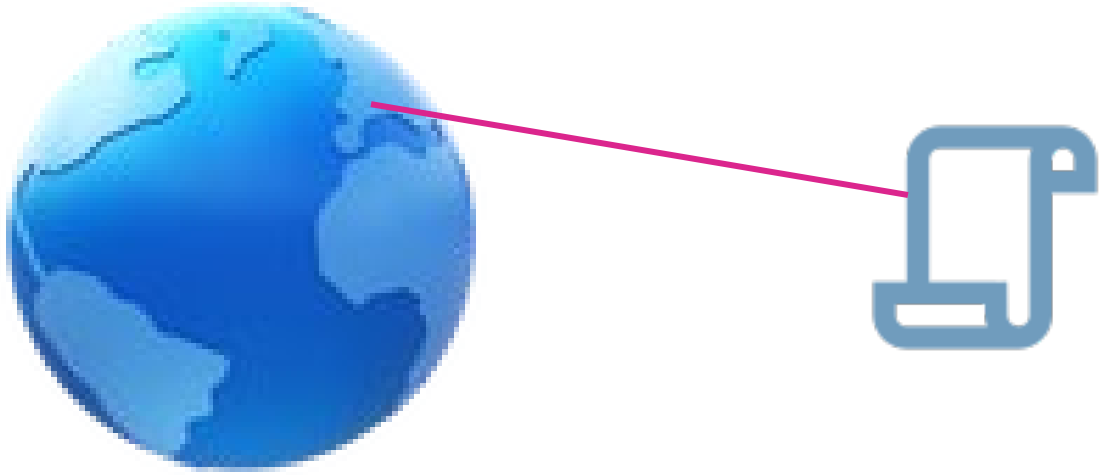
Cache library	Language	Lines	Hour of work
groupcache	Golang	21	<1
mnemonist	Javascript	12	1
lru-rs	Rust	16	1
lru-dict	Python + C	21	<1

SIEVE Resources

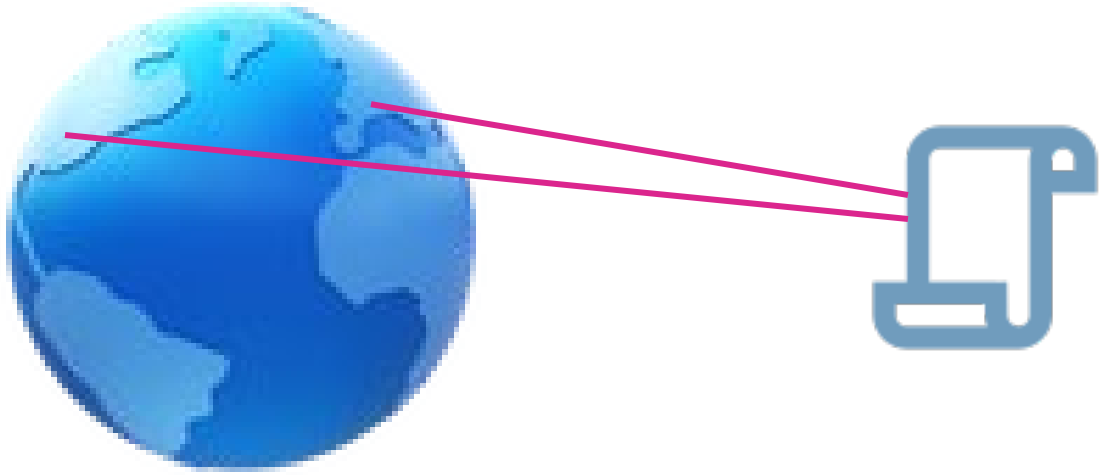
- Blog: https://yazhuozhang.com/blog/2023/12/17/sieve-is-simpler-than-lru/
- Github source: https://cachemon.github.io/SIEVE-website/
- Research paper: https://junchengyang.com/publication/nsdi24-SIEVE.pdf
- Shortcomings
  - Not scan resistant
  - Therefore not a good candidate for storage caches

Optimizing CDNs at Scale

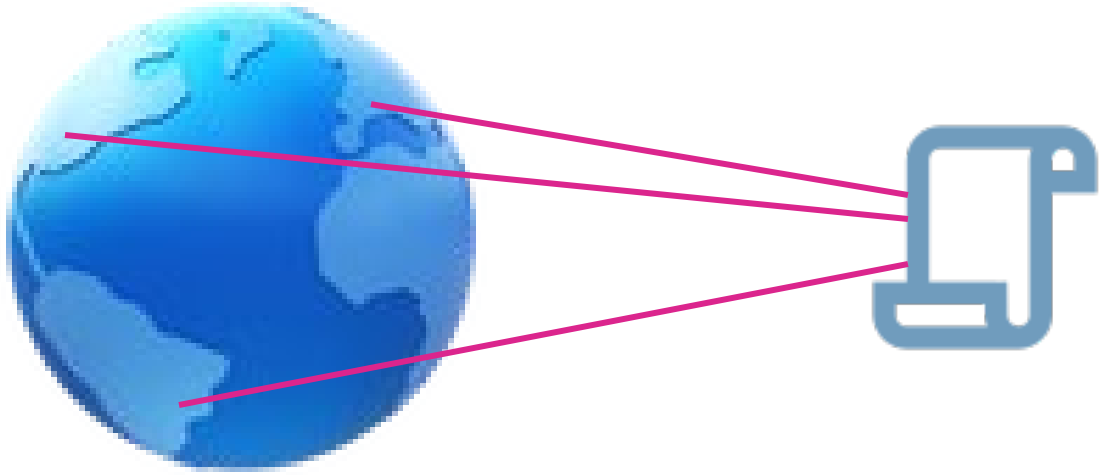
Content Delivery Networks



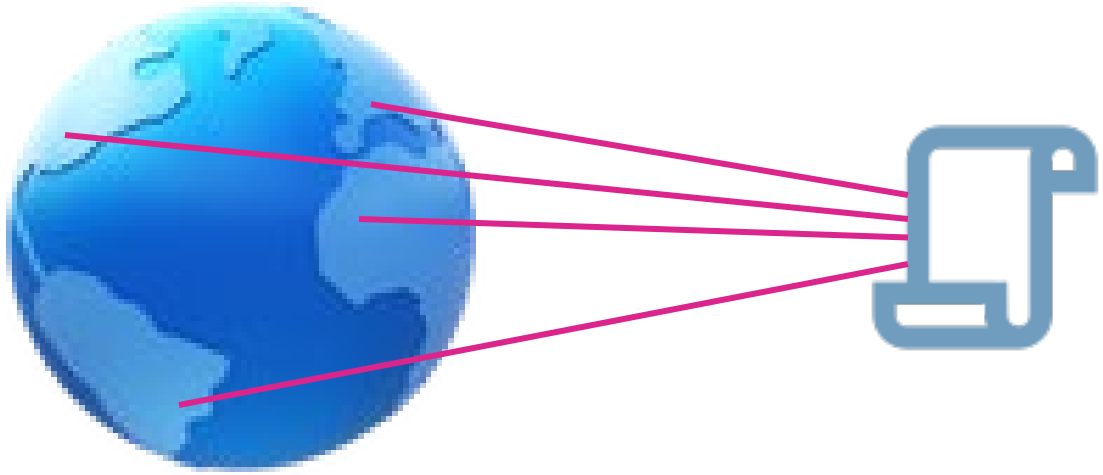
Content Delivery Networks



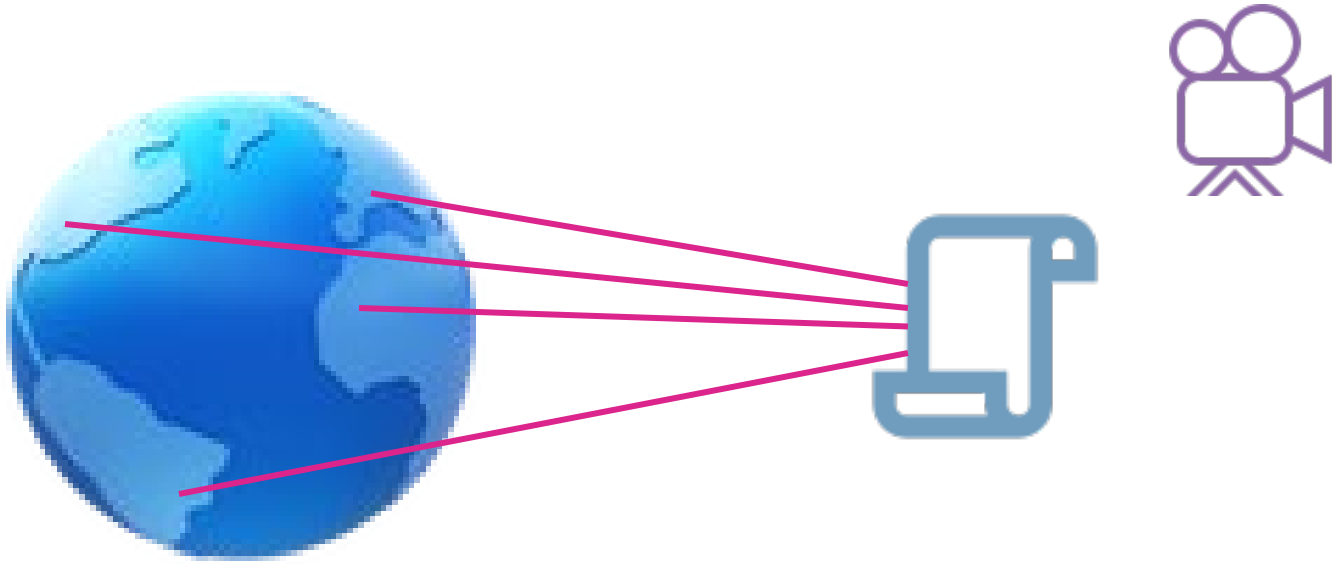
Content Delivery Networks



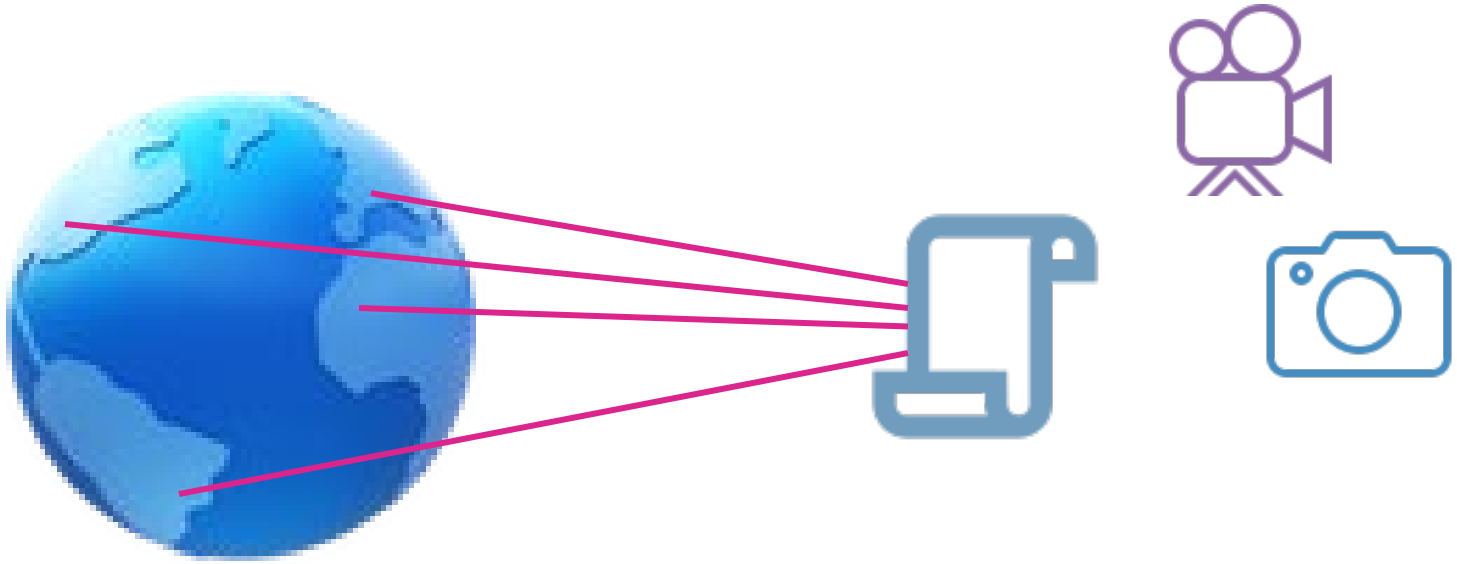
Content Delivery Networks



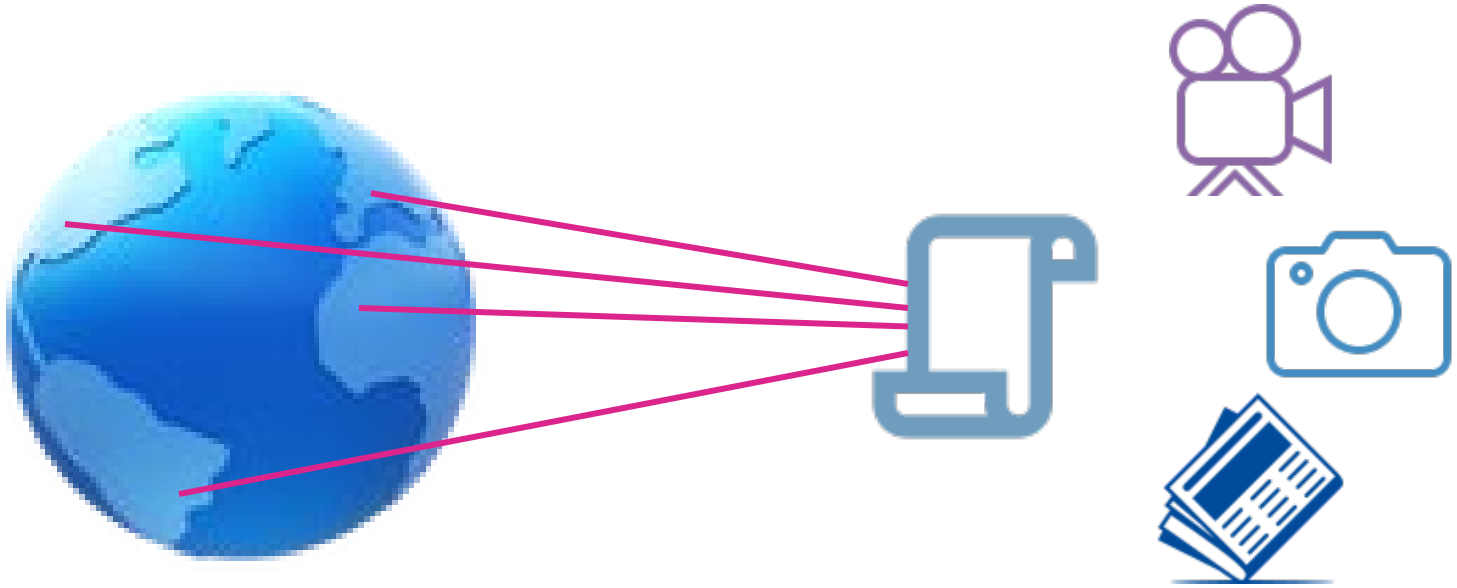
Content Delivery Networks



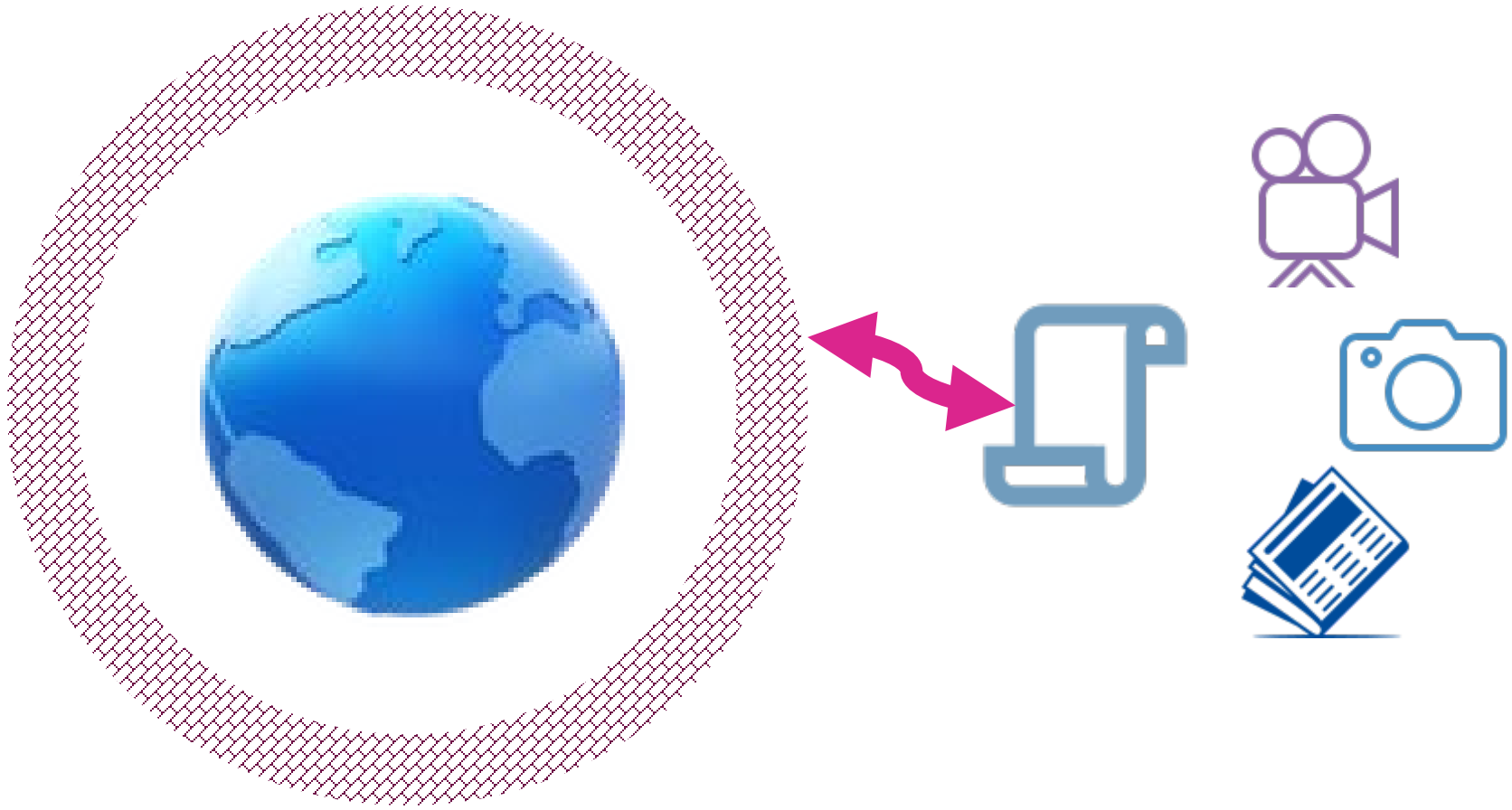
Content Delivery Networks



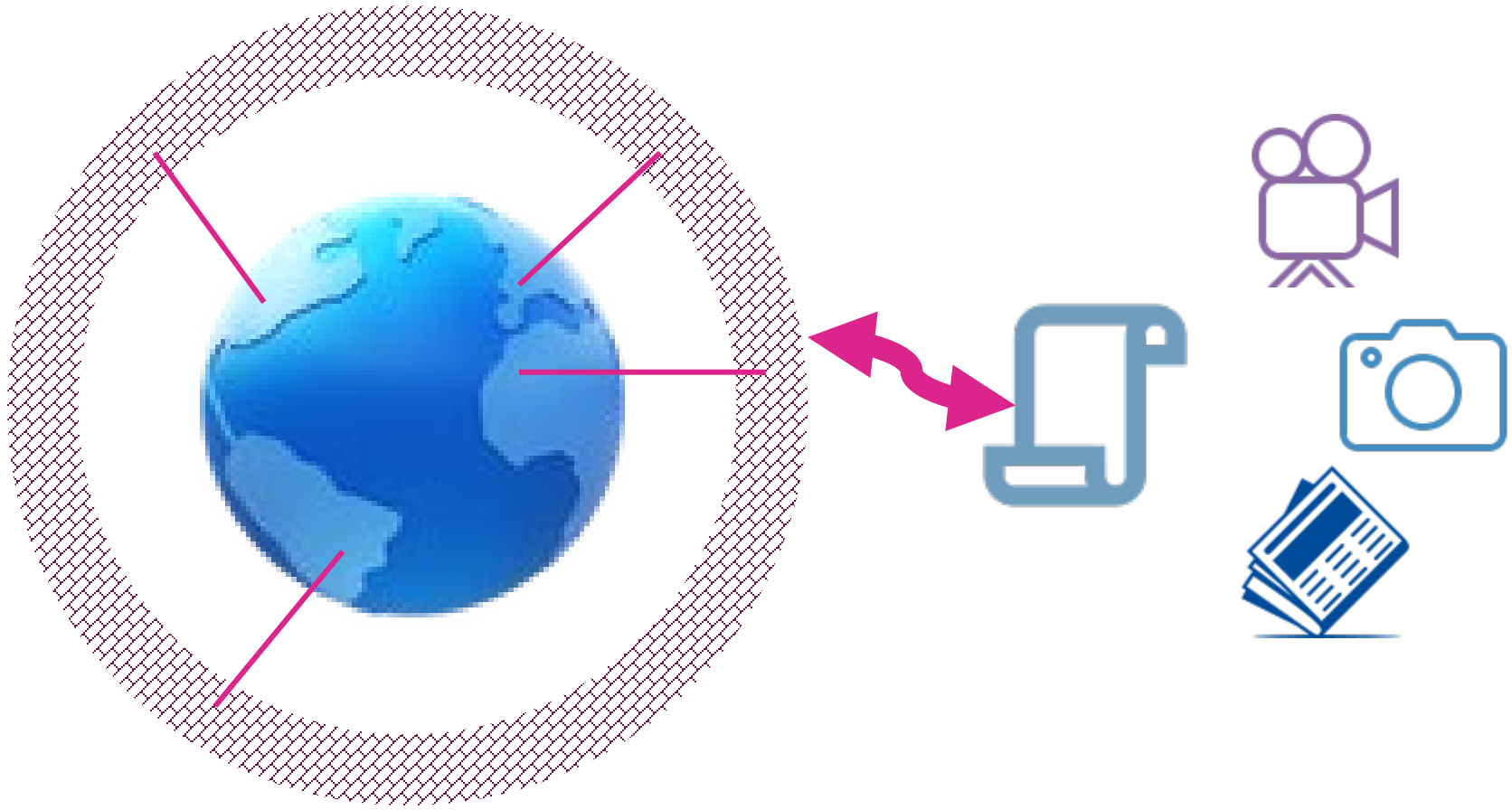
Content Delivery Networks



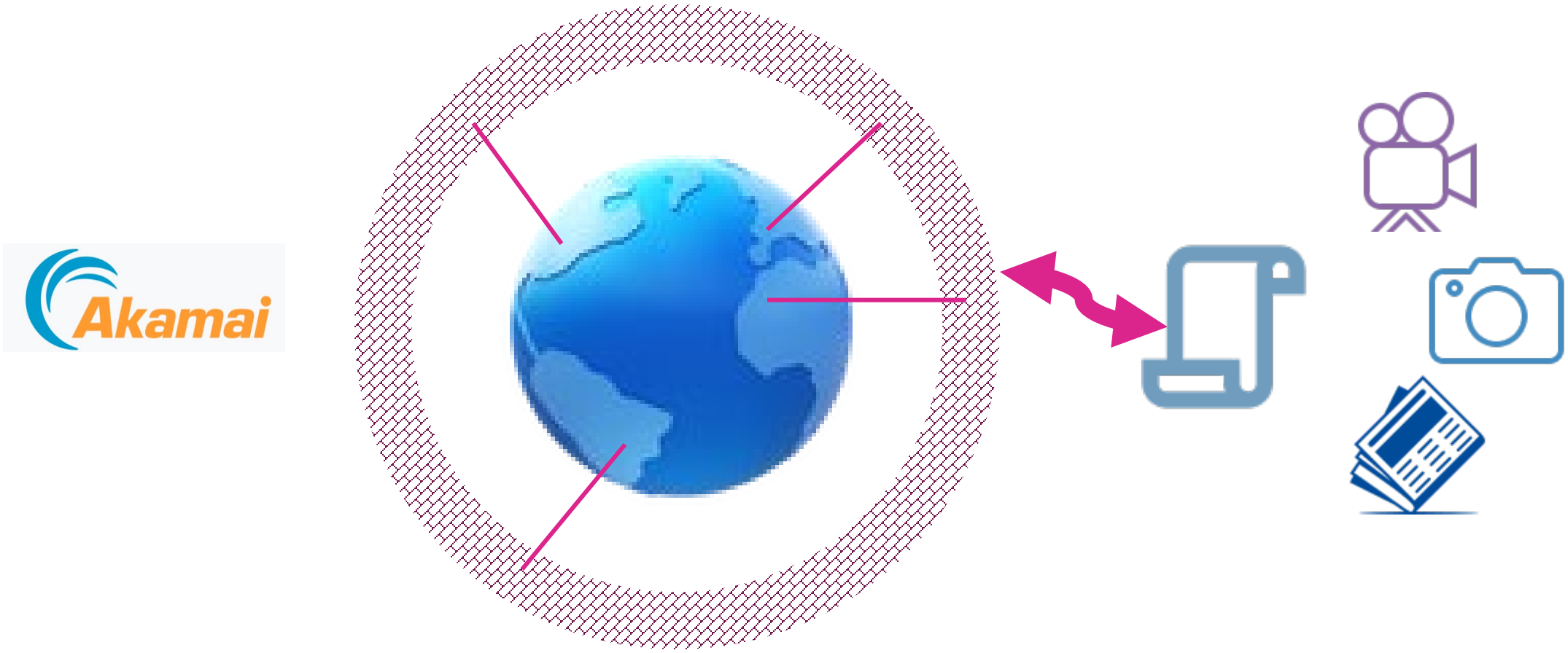
Content Delivery Networks



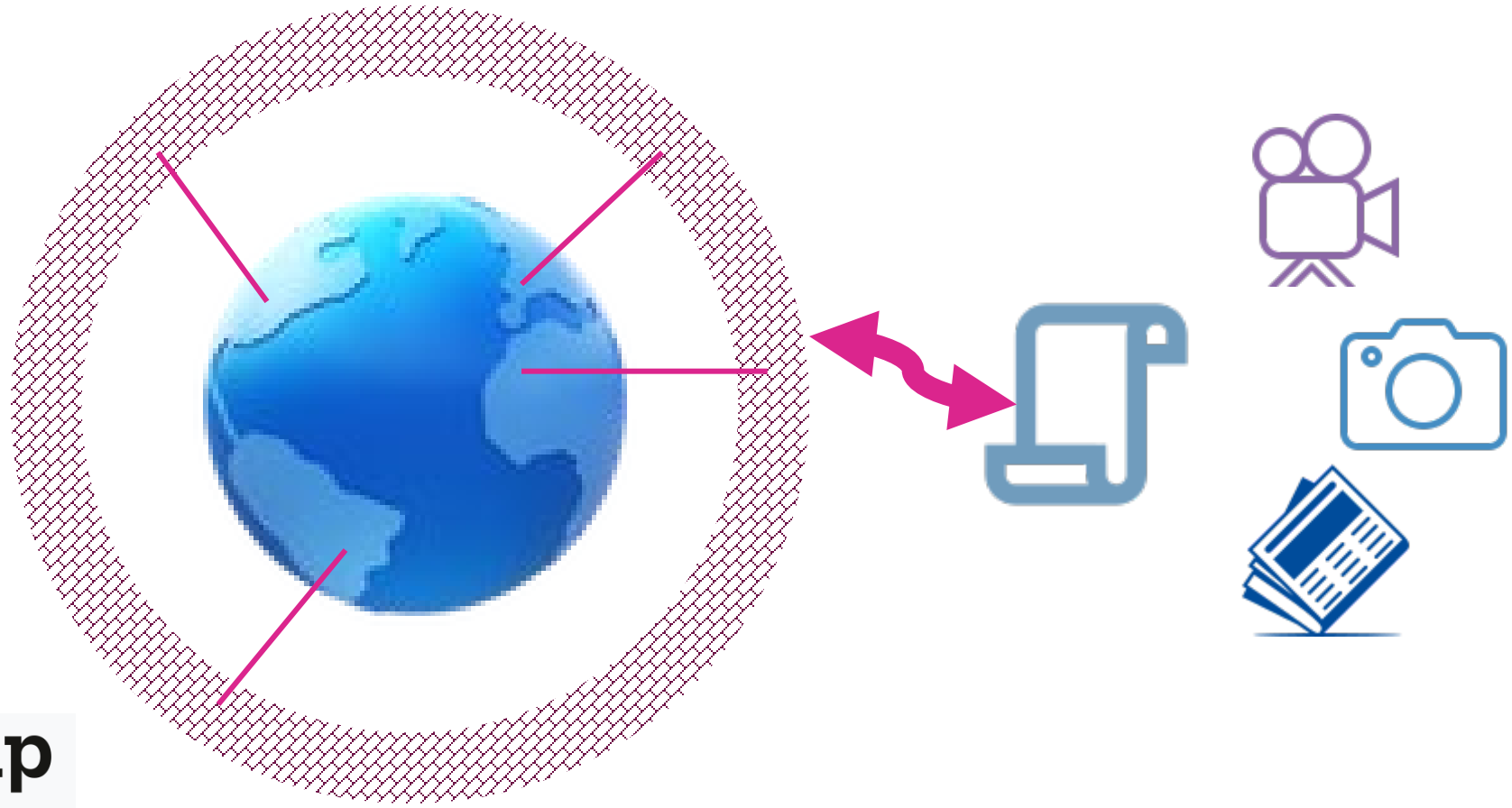
Content Delivery Networks



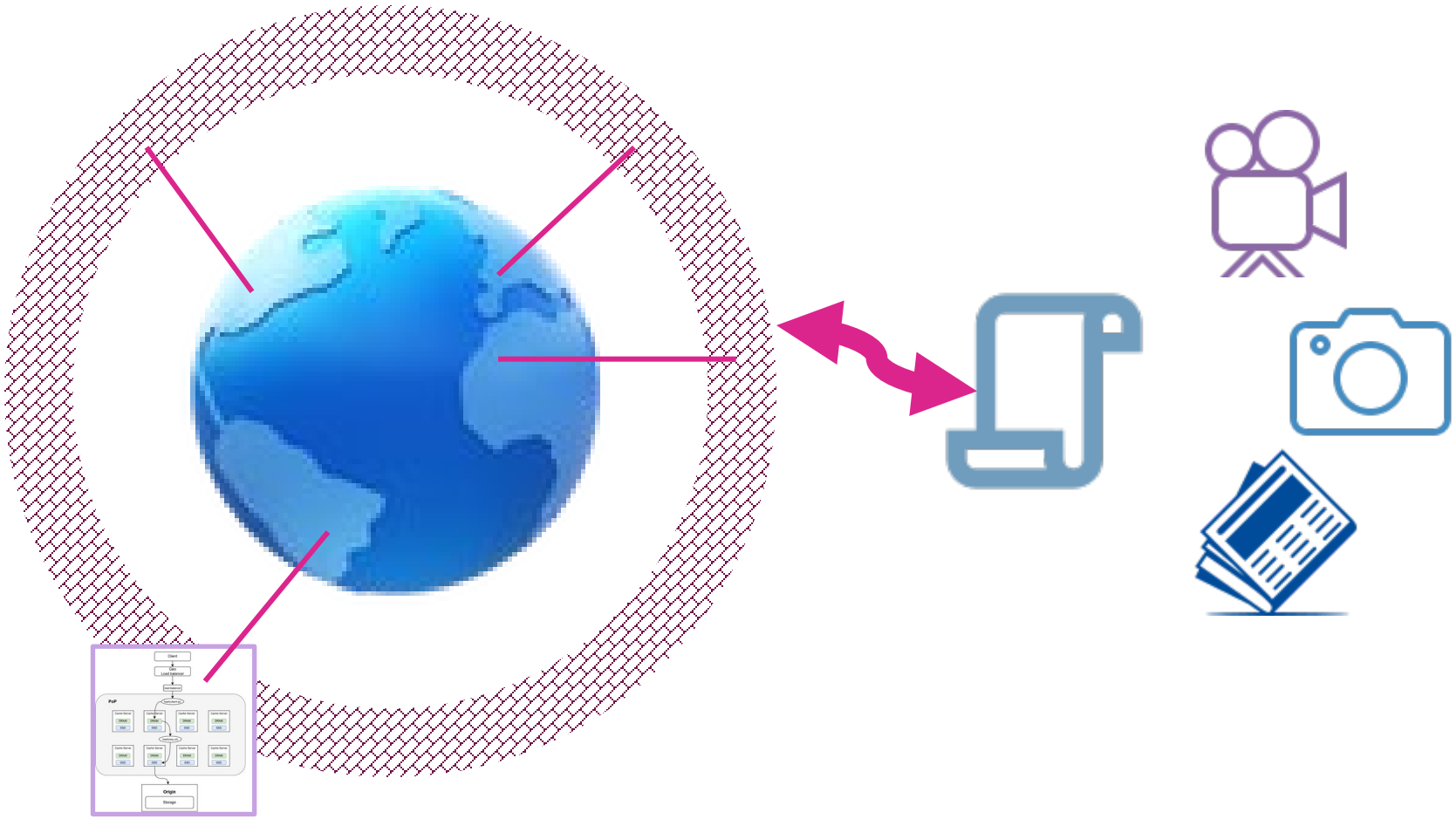
Content Delivery Networks



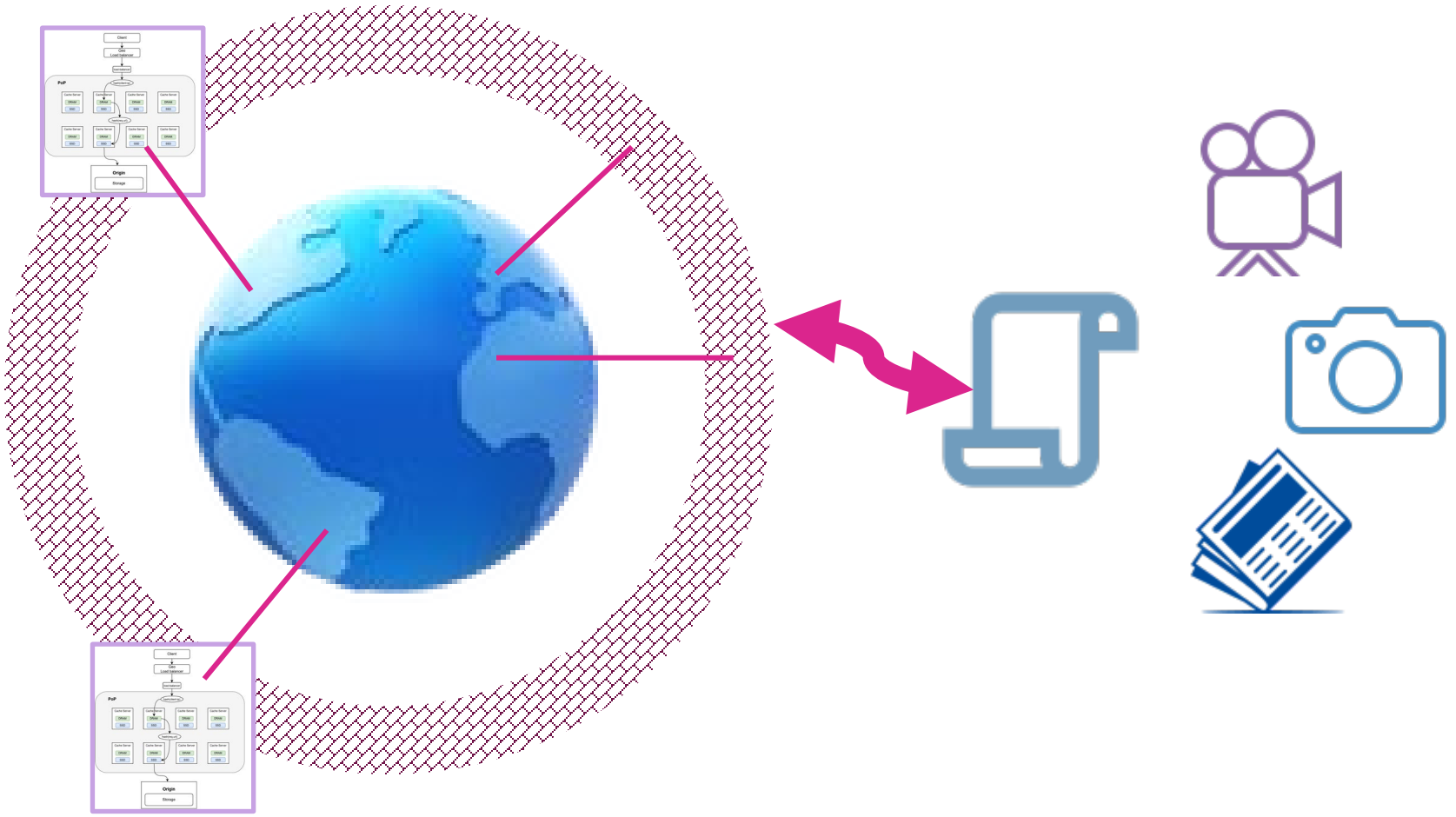
Content Delivery Networks



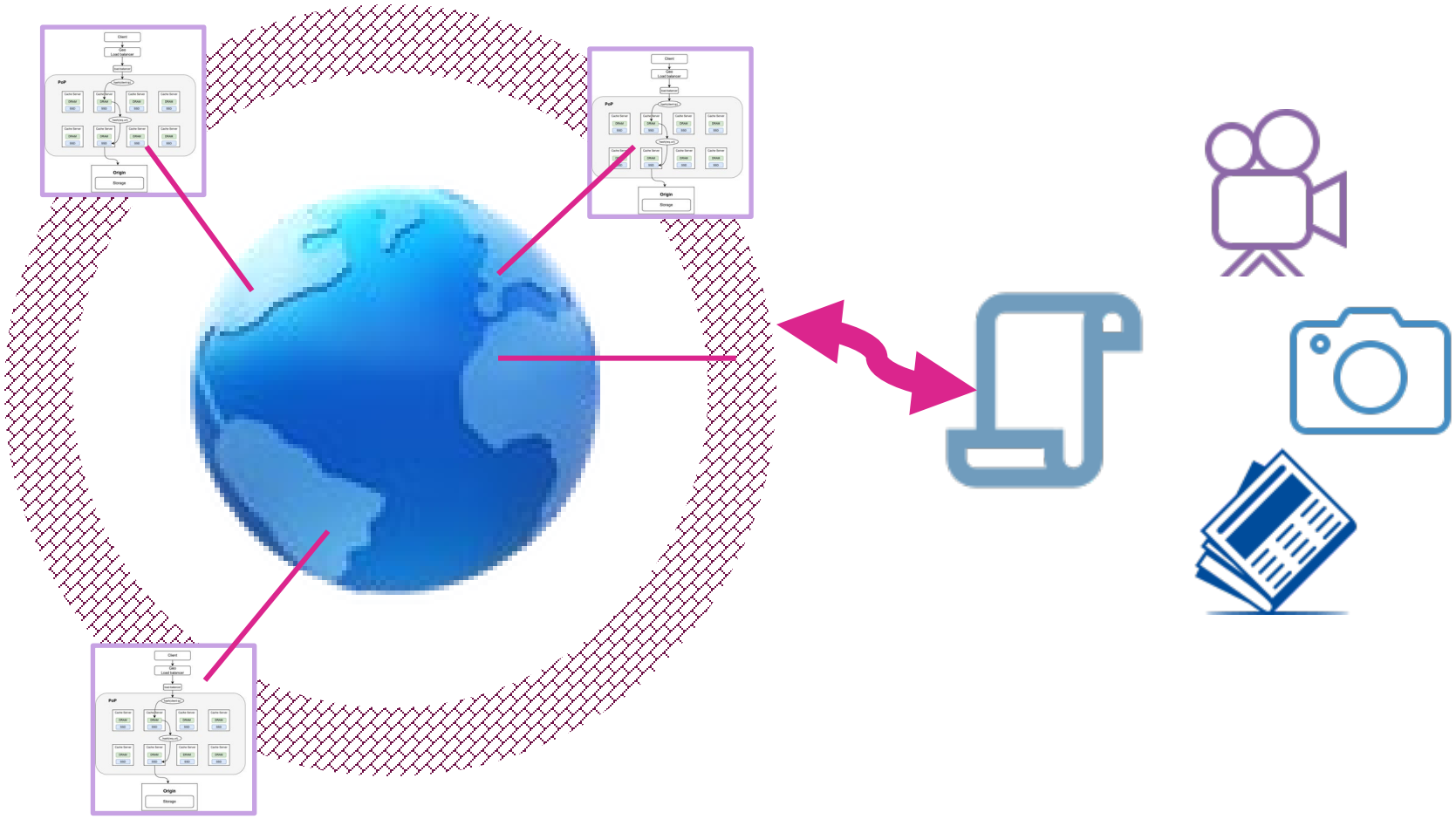
Content Delivery Networks



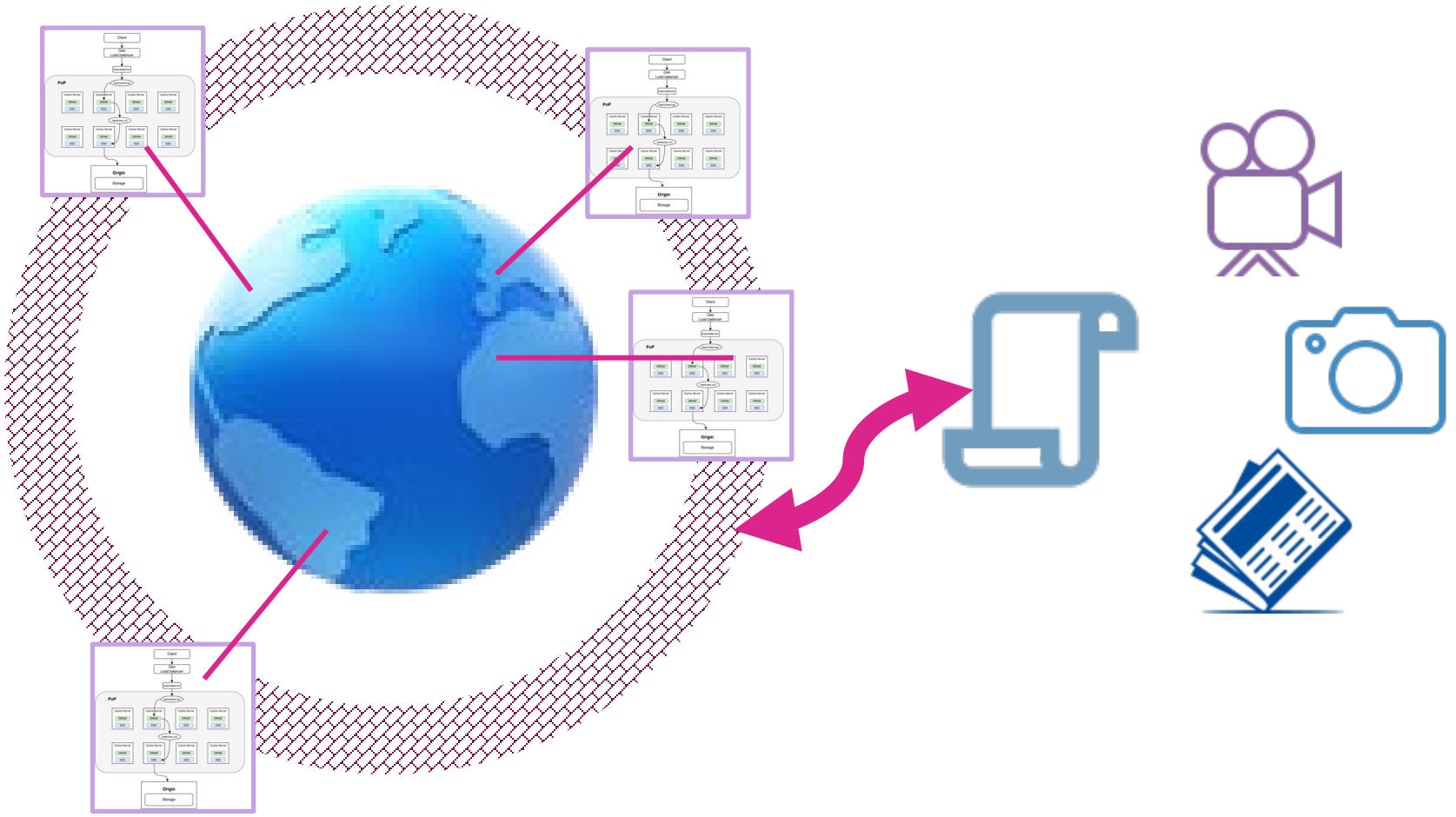
Content Delivery Networks



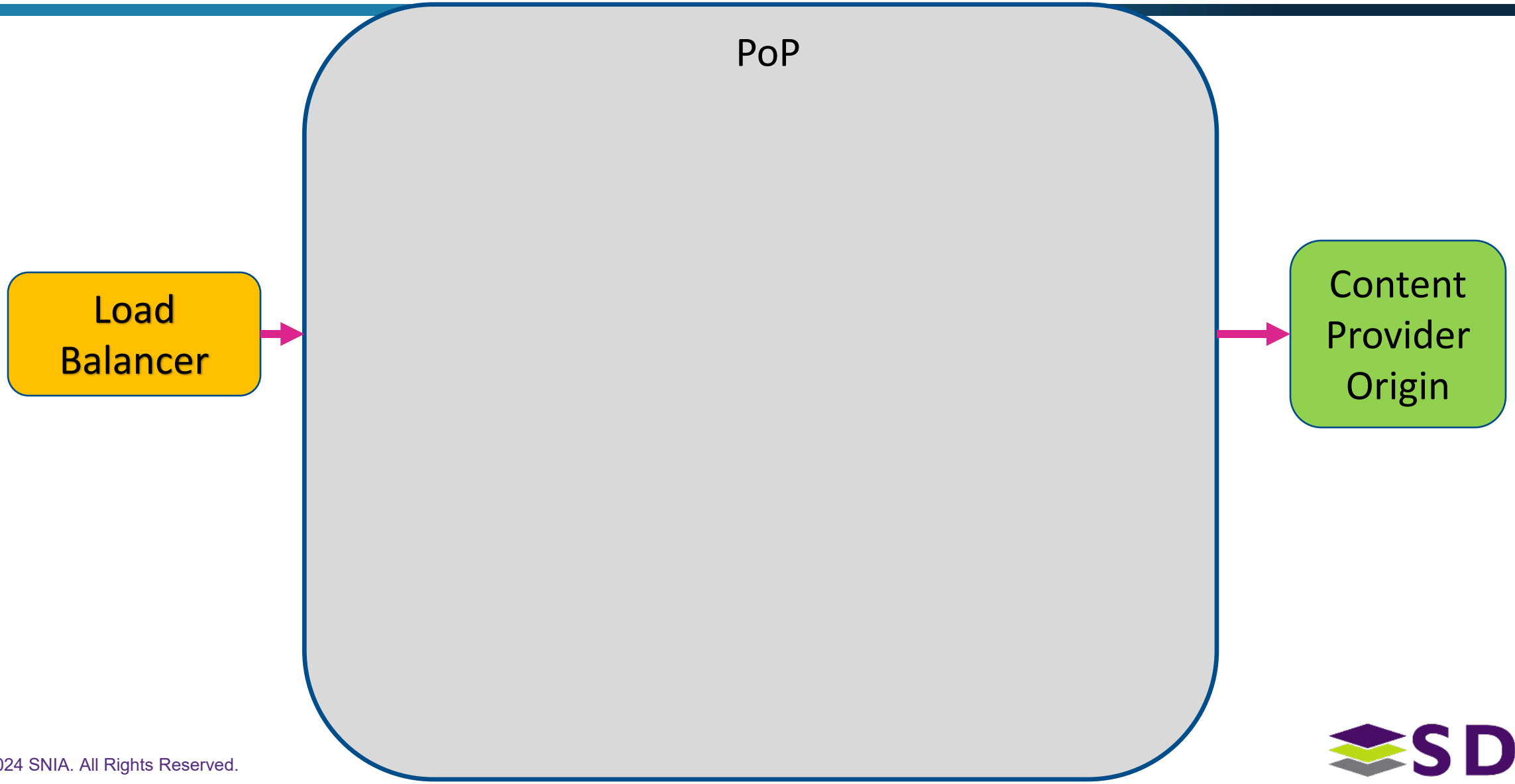
Content Delivery Networks



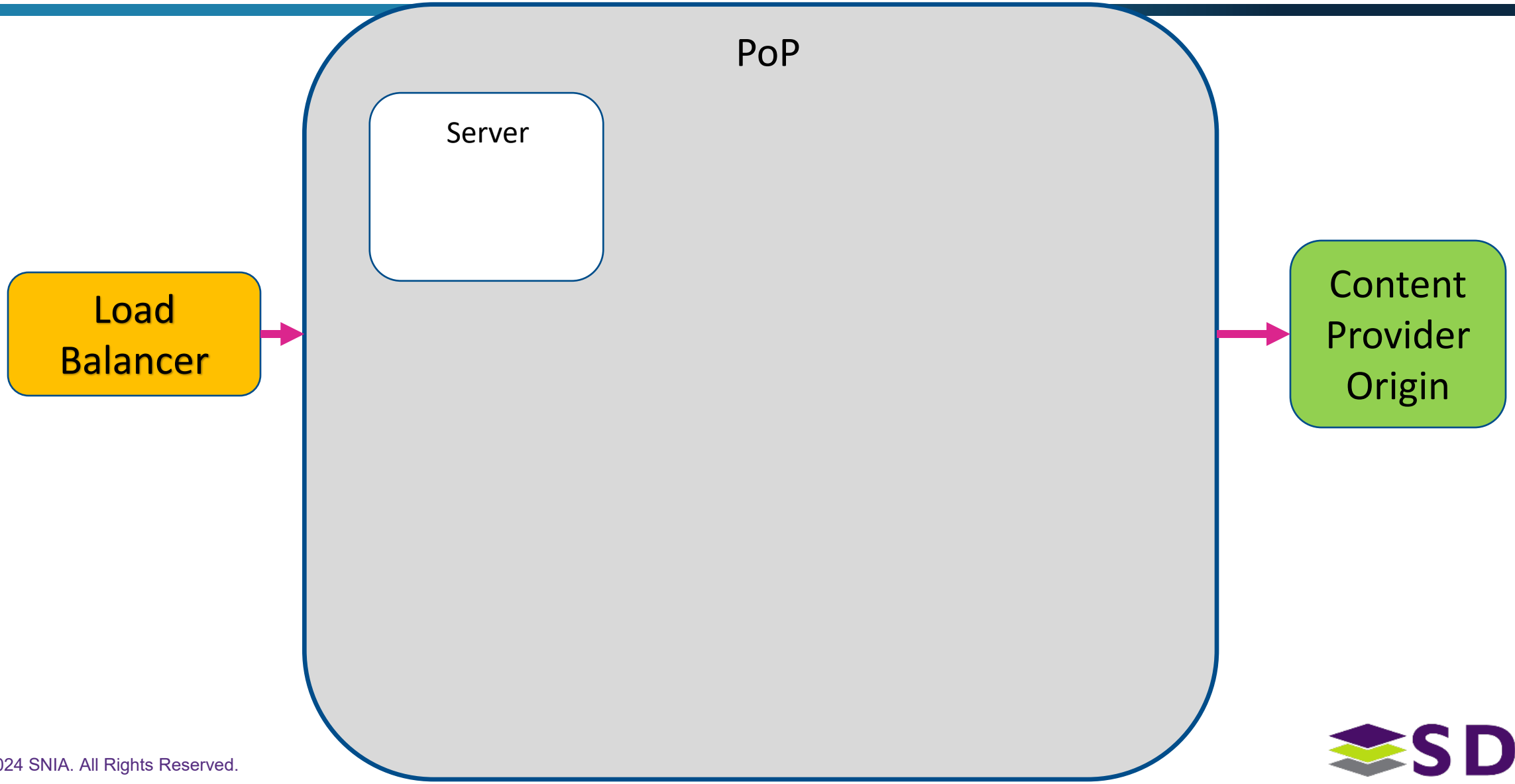
Content Delivery Networks



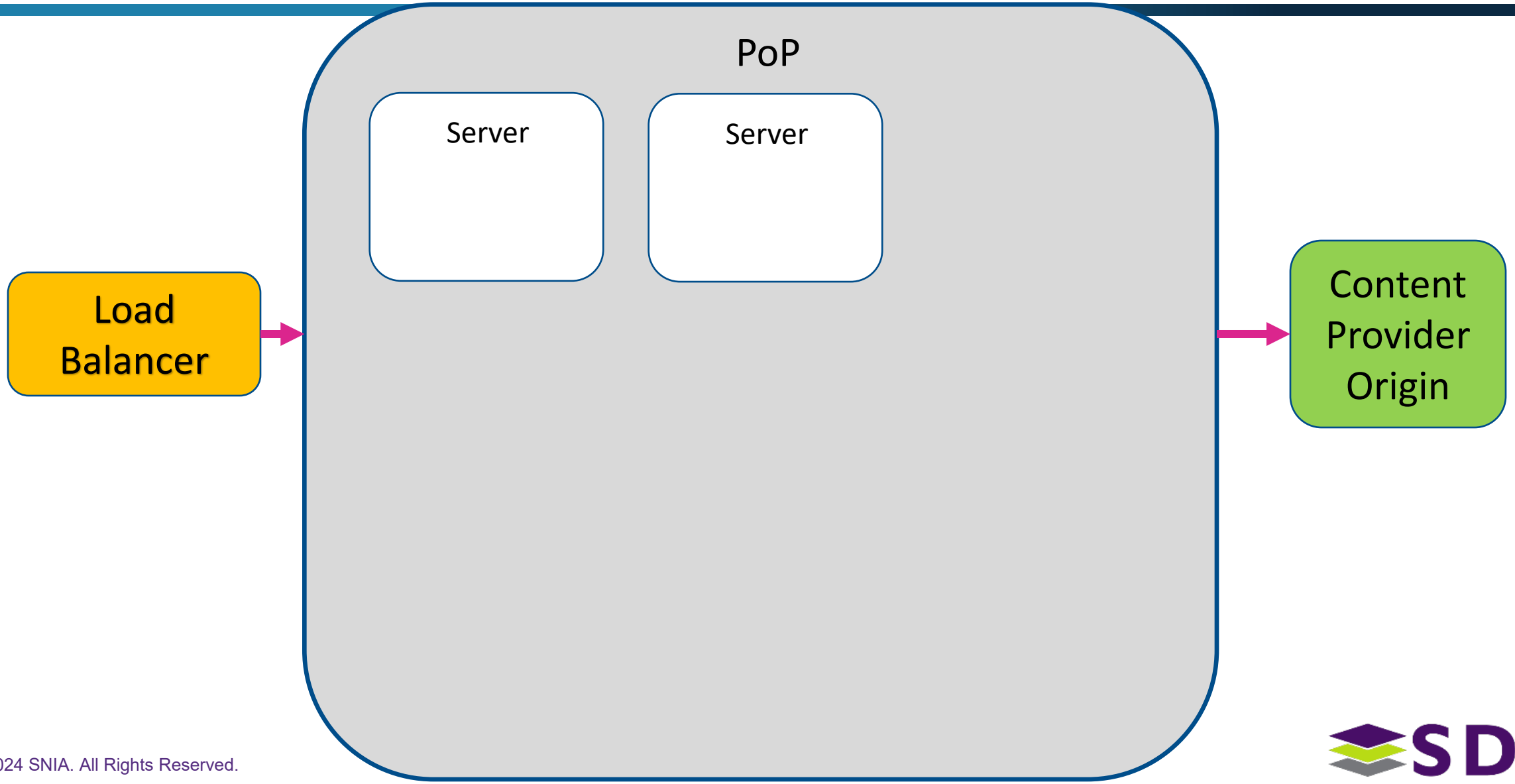
CDN PoP



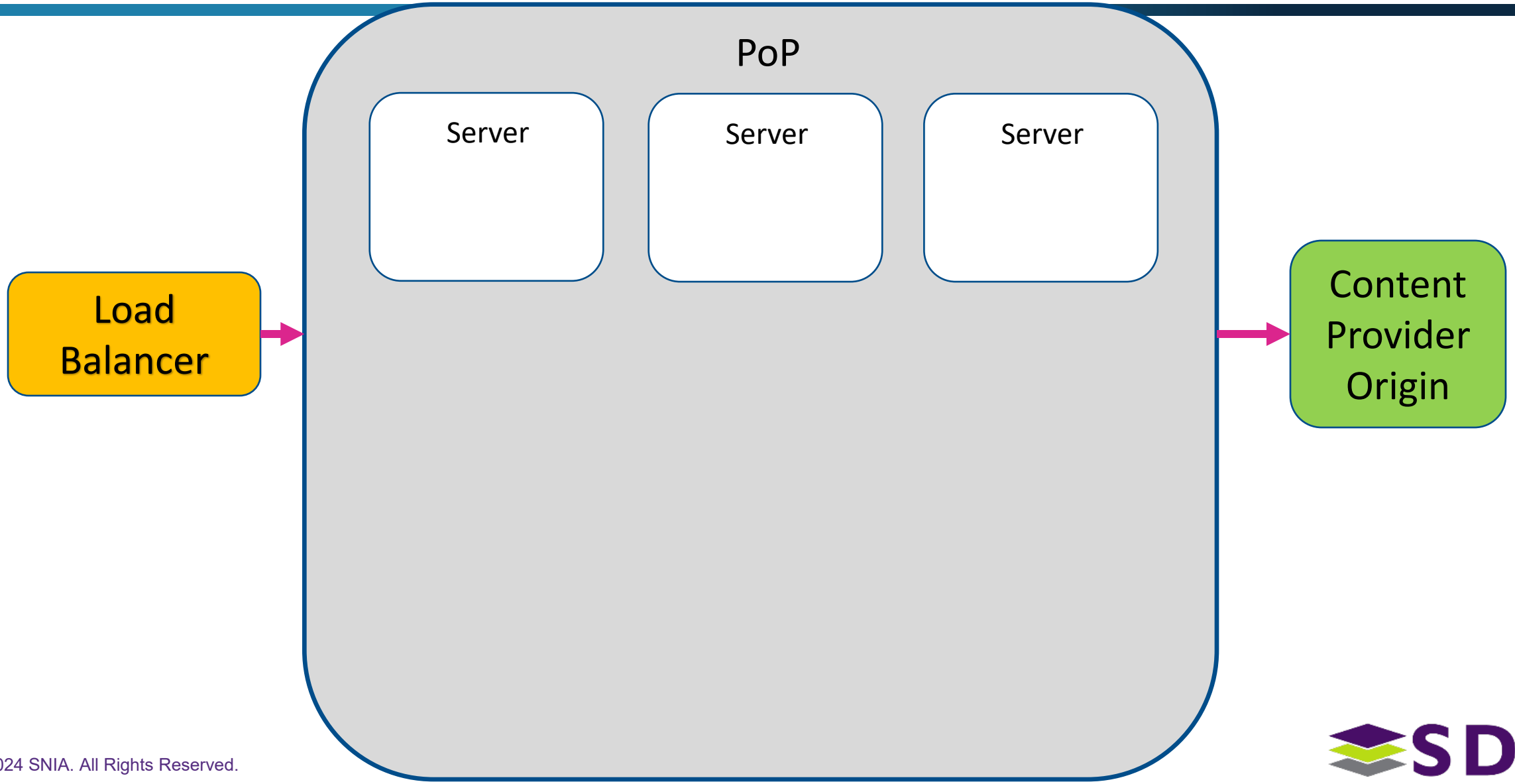
CDN PoP



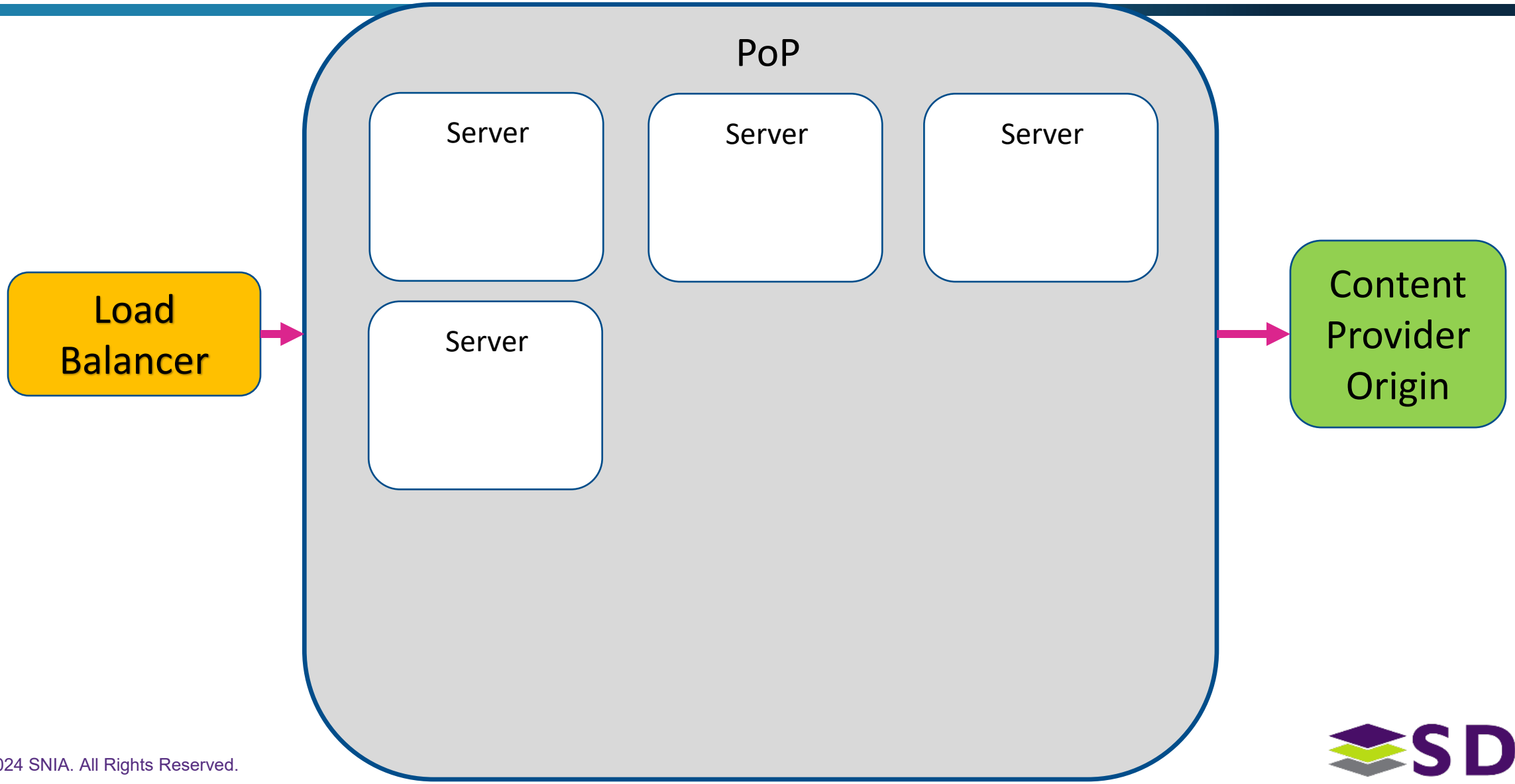
CDN PoP -- Compute



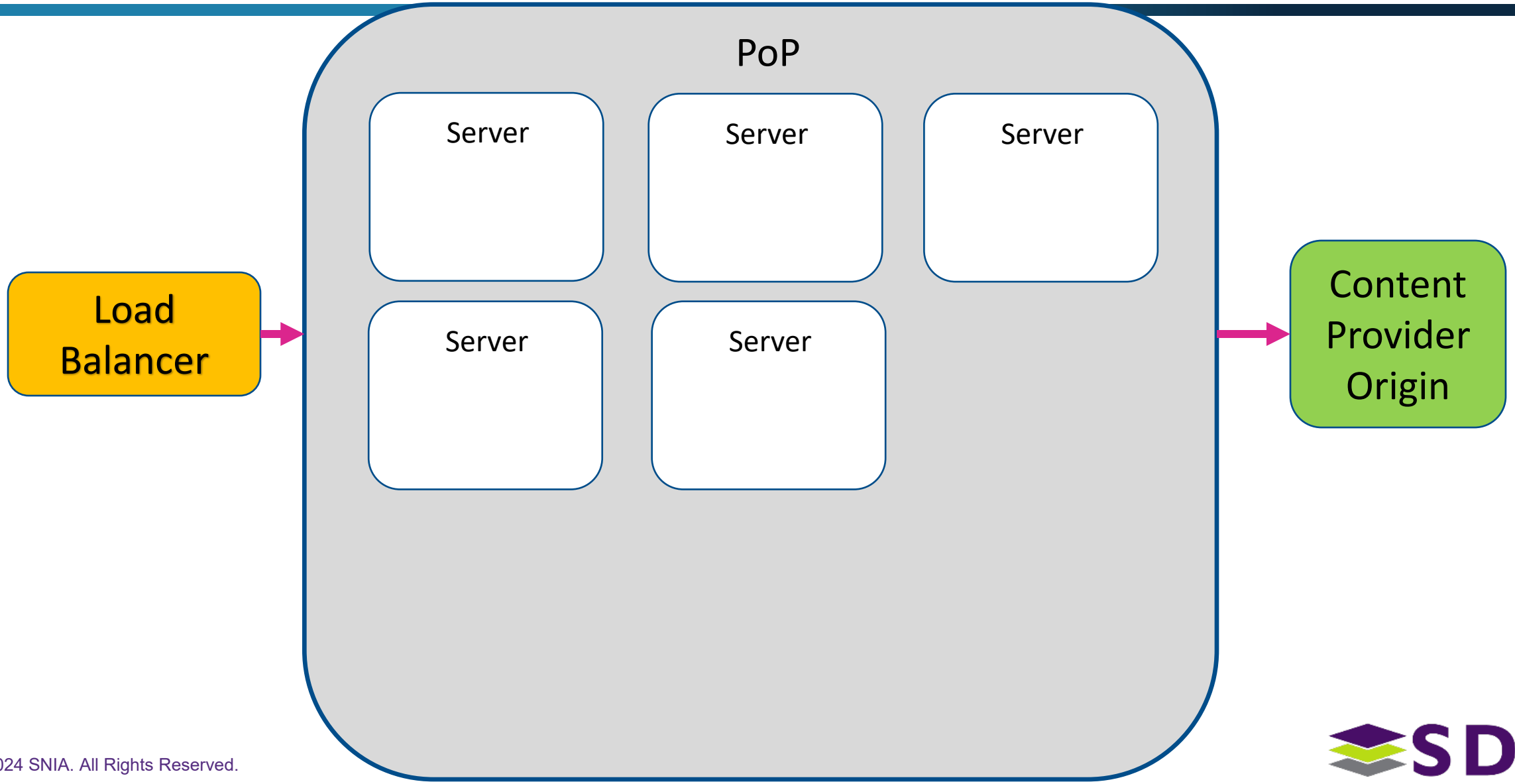
CDN PoP -- Compute



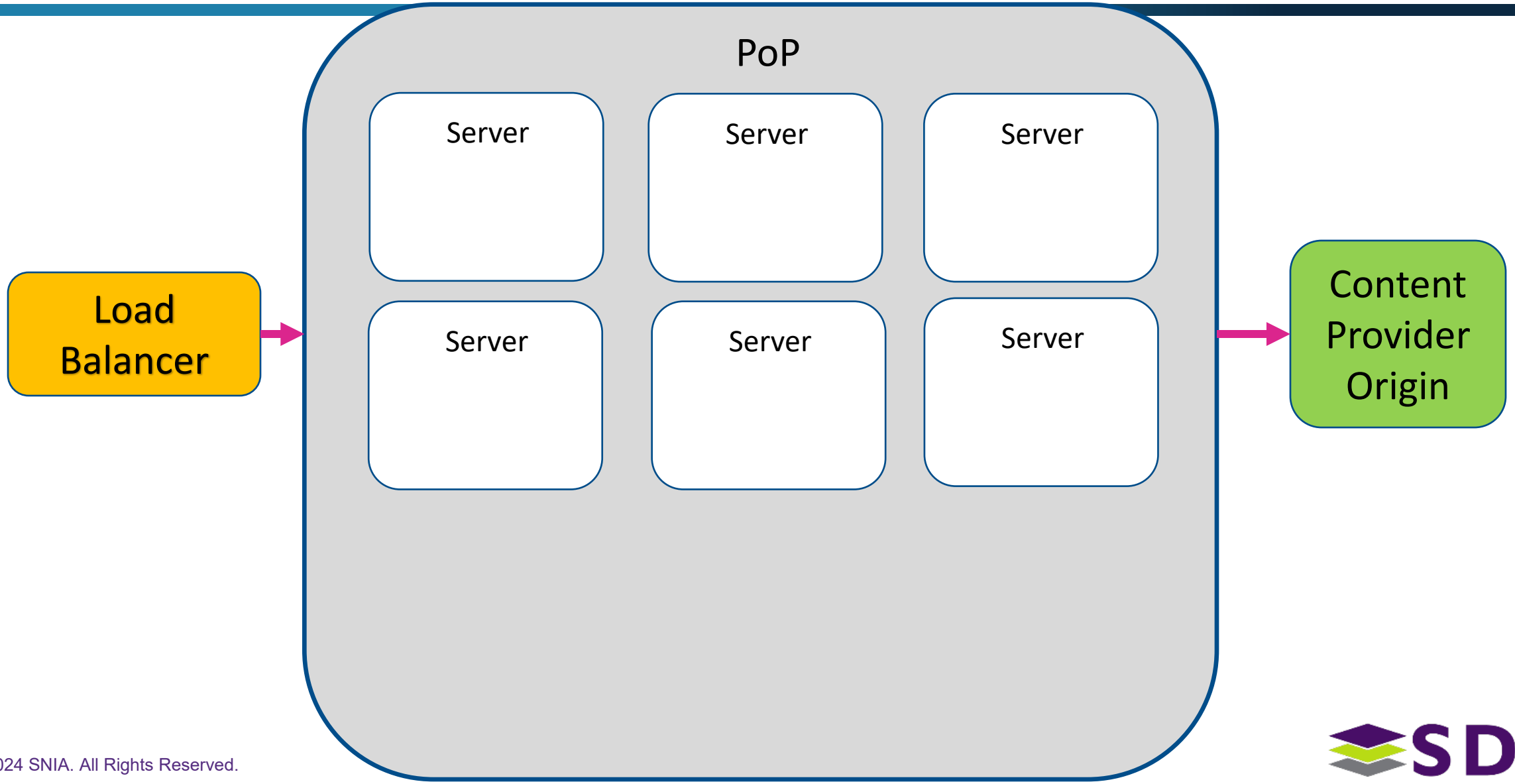
CDN PoP -- Compute



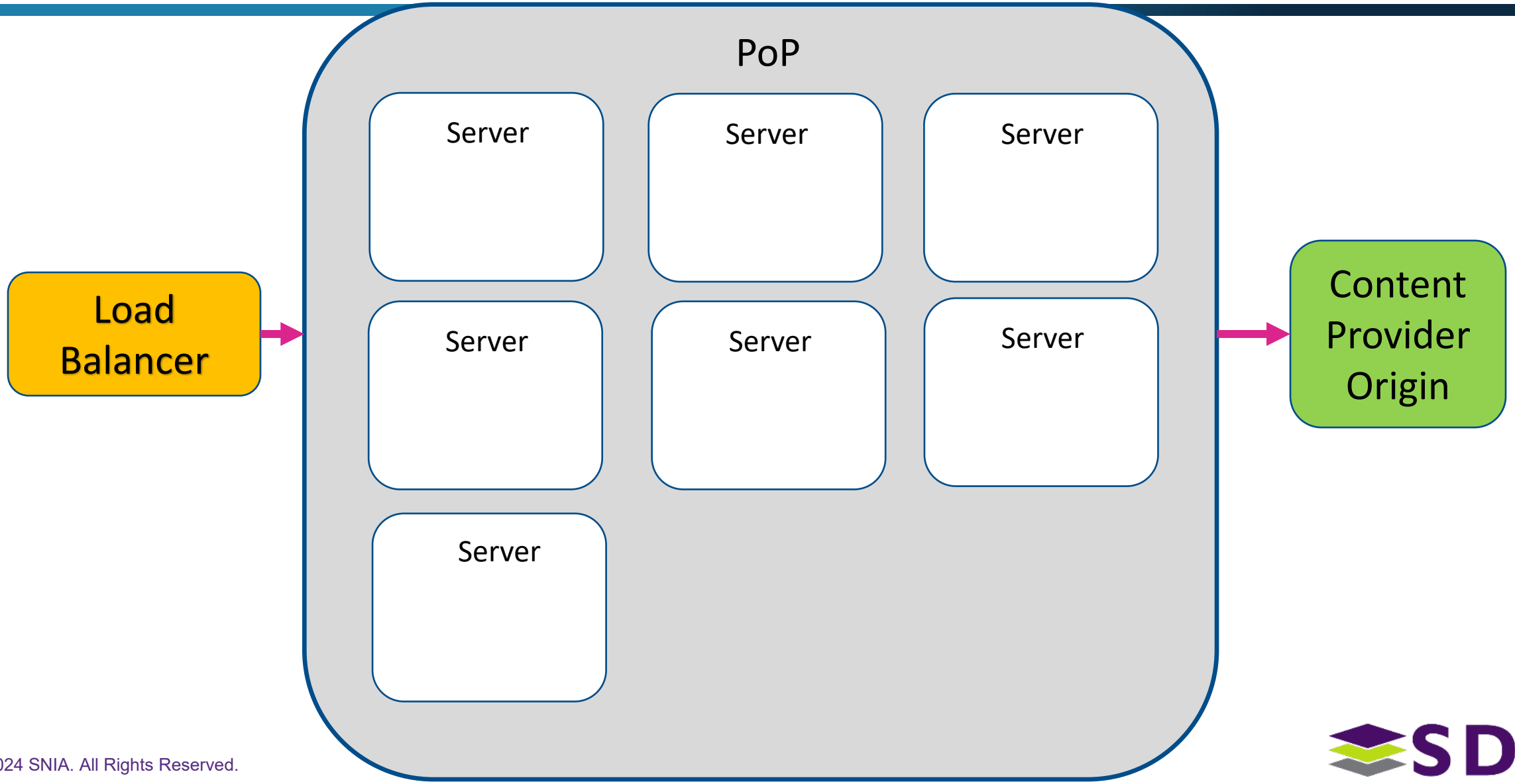
CDN PoP -- Compute



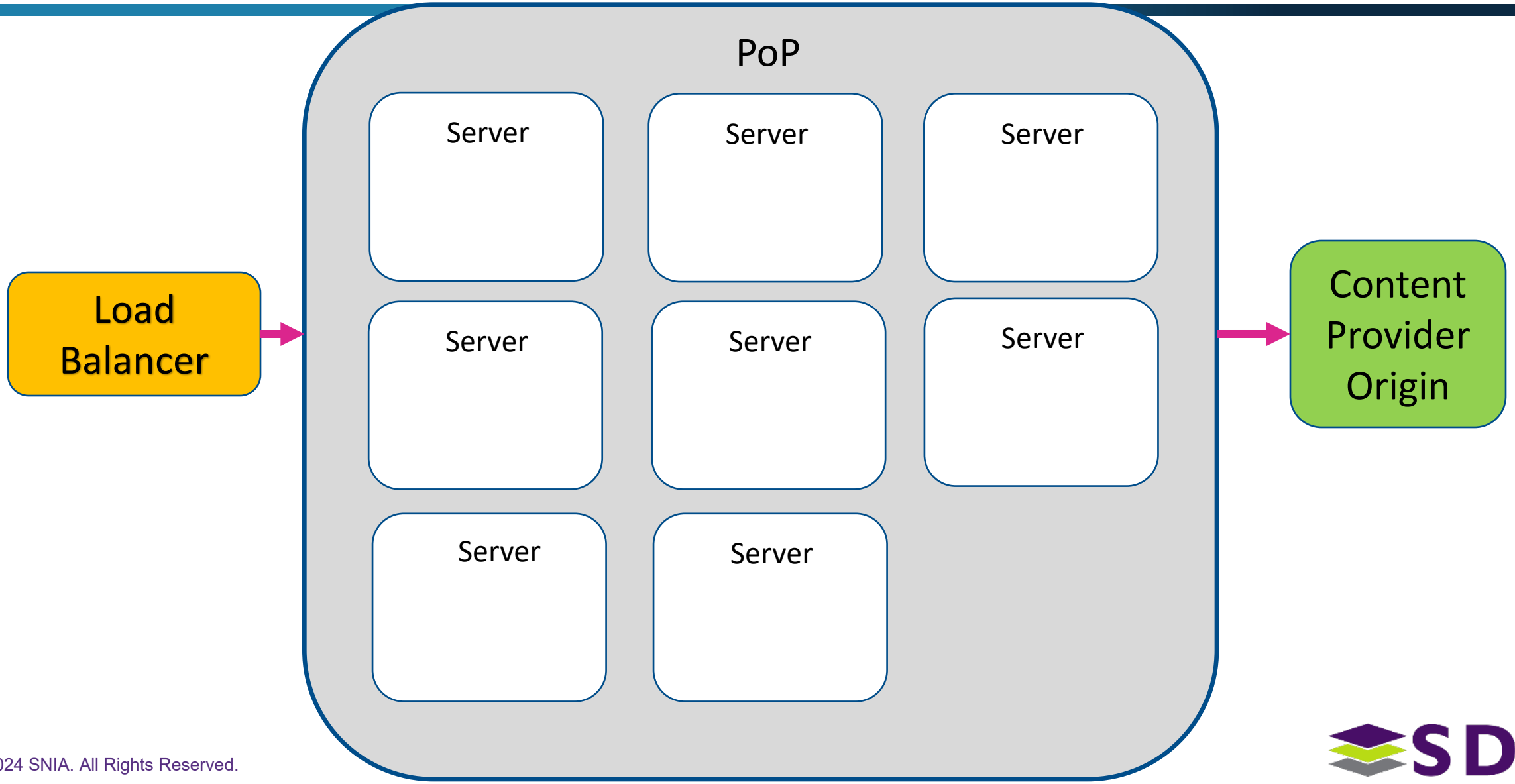
CDN PoP -- Compute



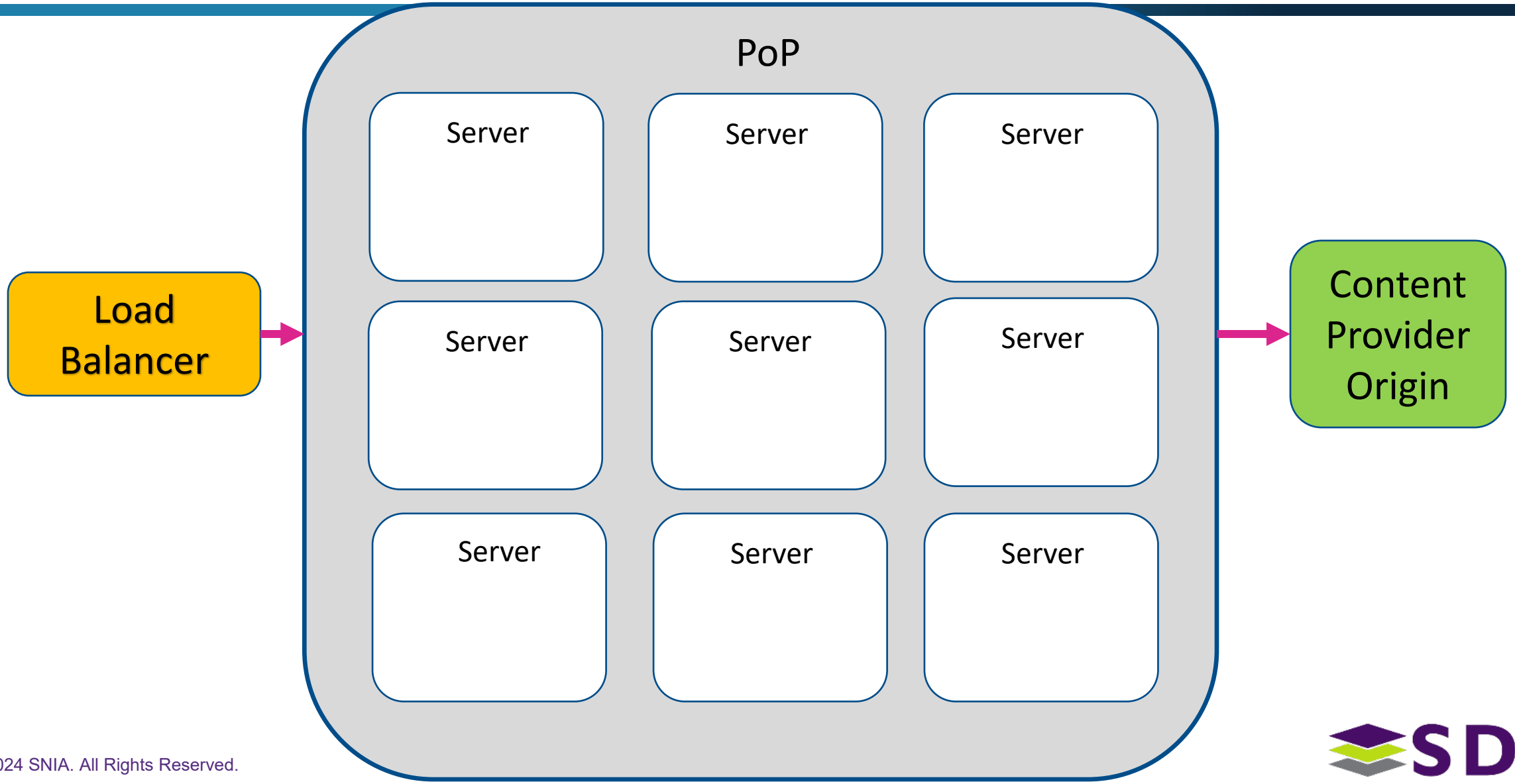
CDN PoP -- Compute



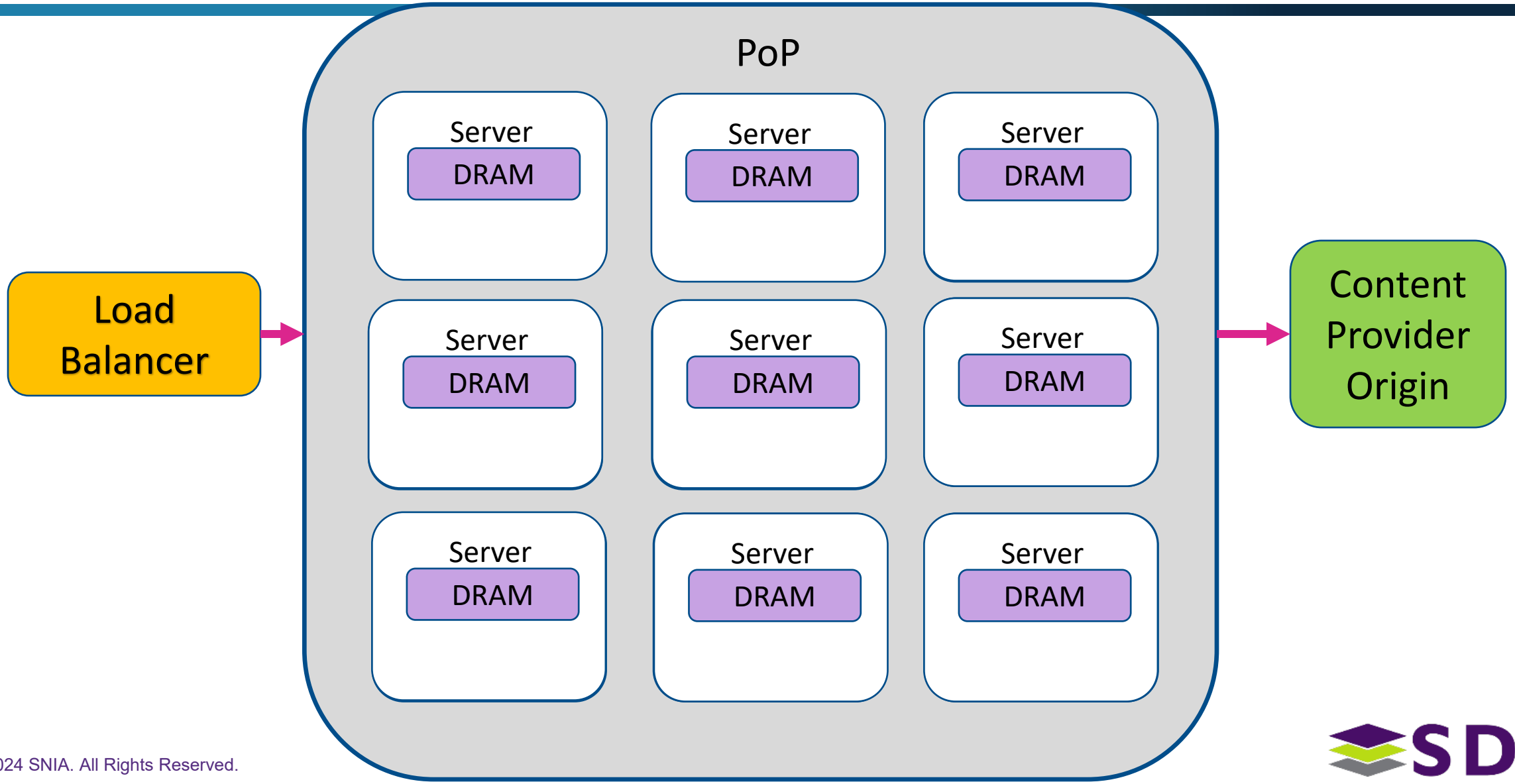
CDN PoP -- Compute



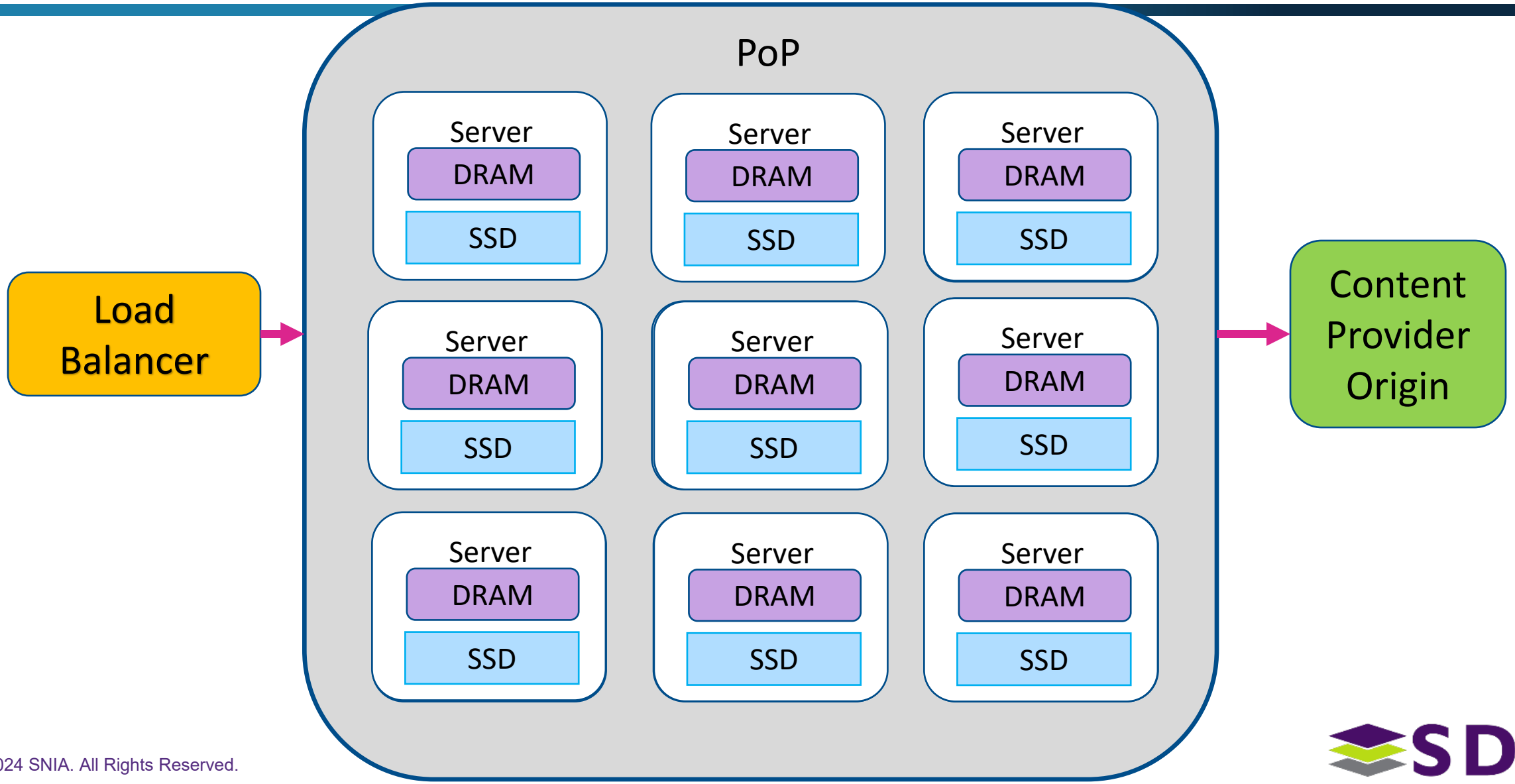
CDN PoP -- Compute



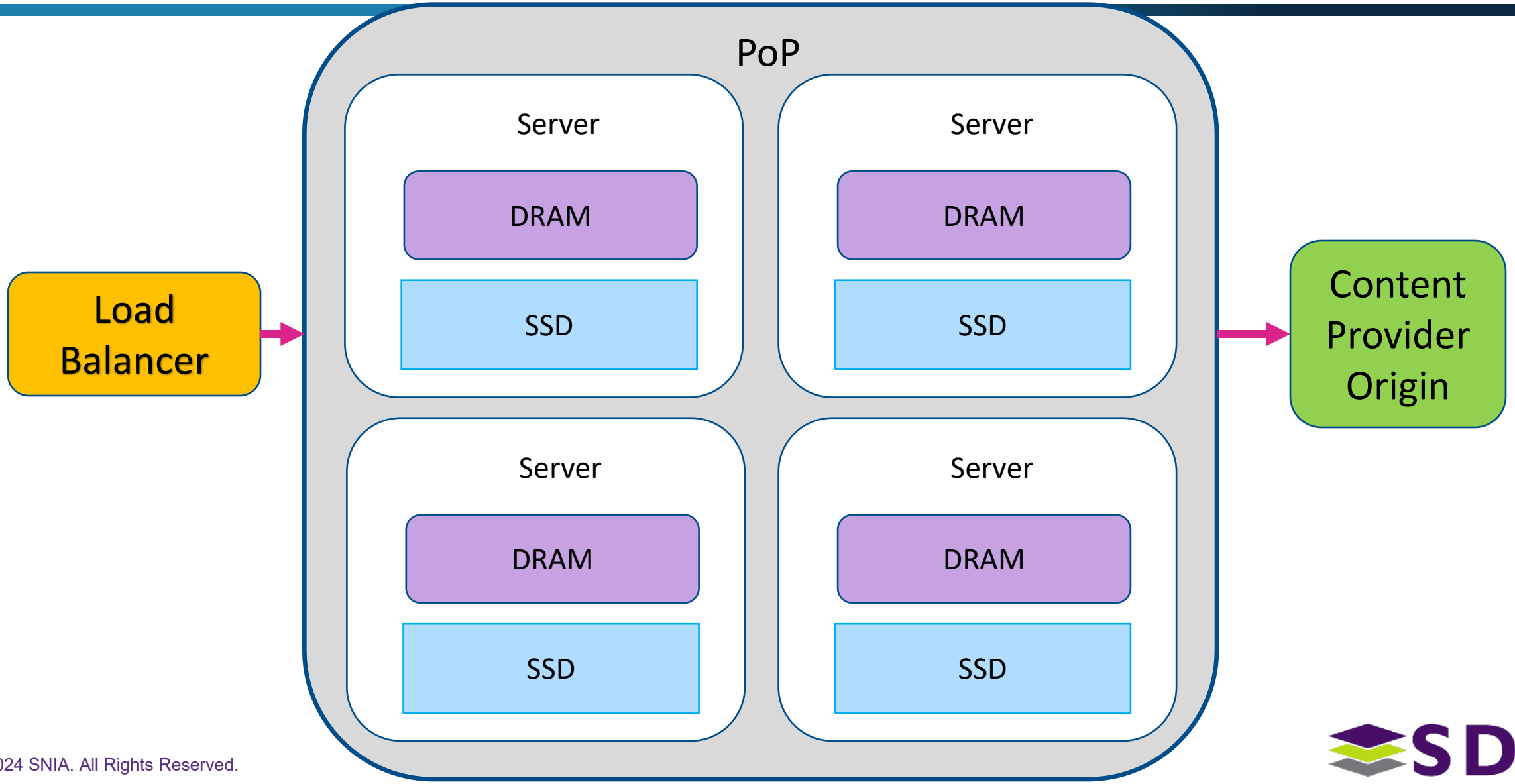
CDN PoP – Compute + Memory



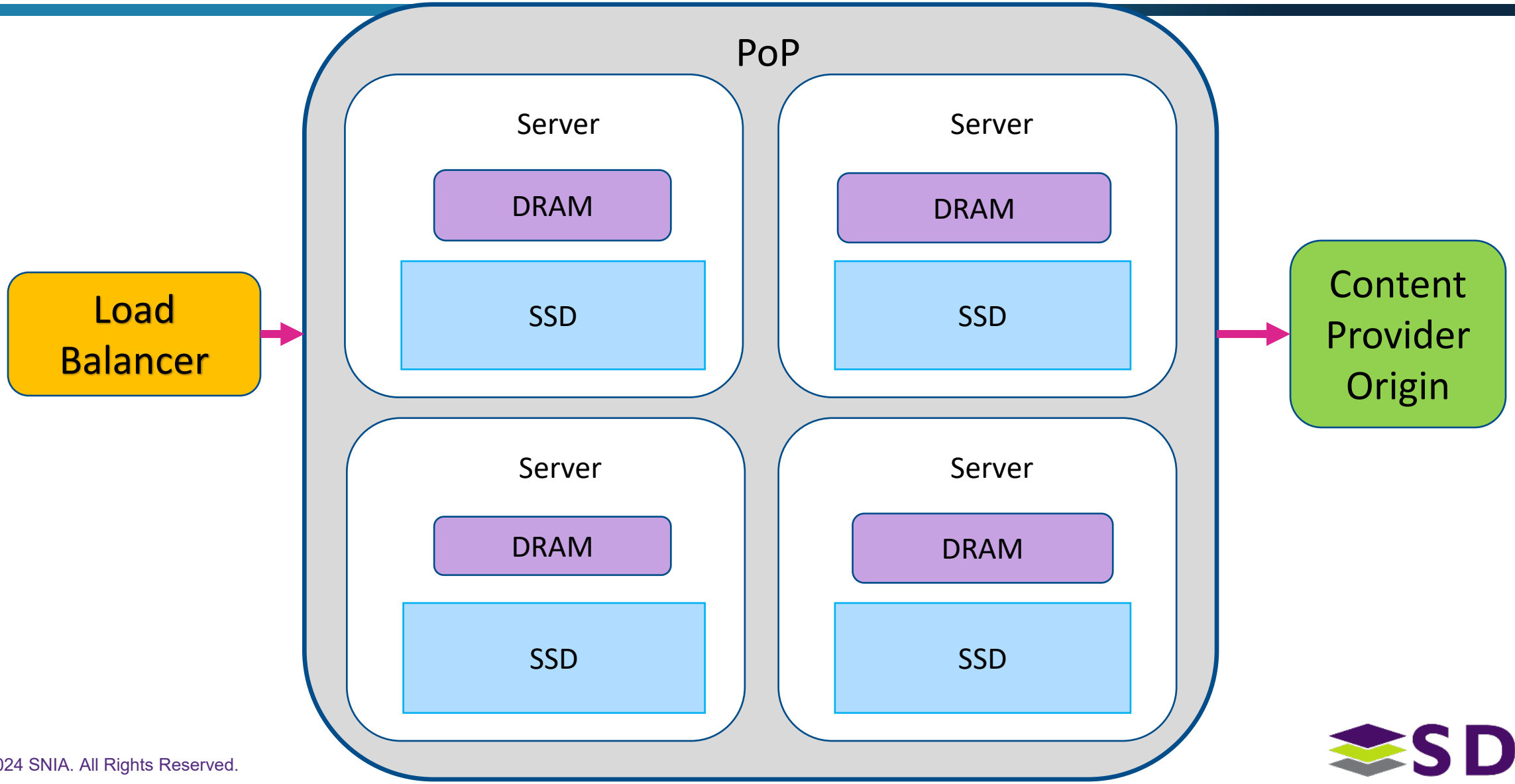
CDN PoP – Compute + Memory + Storage



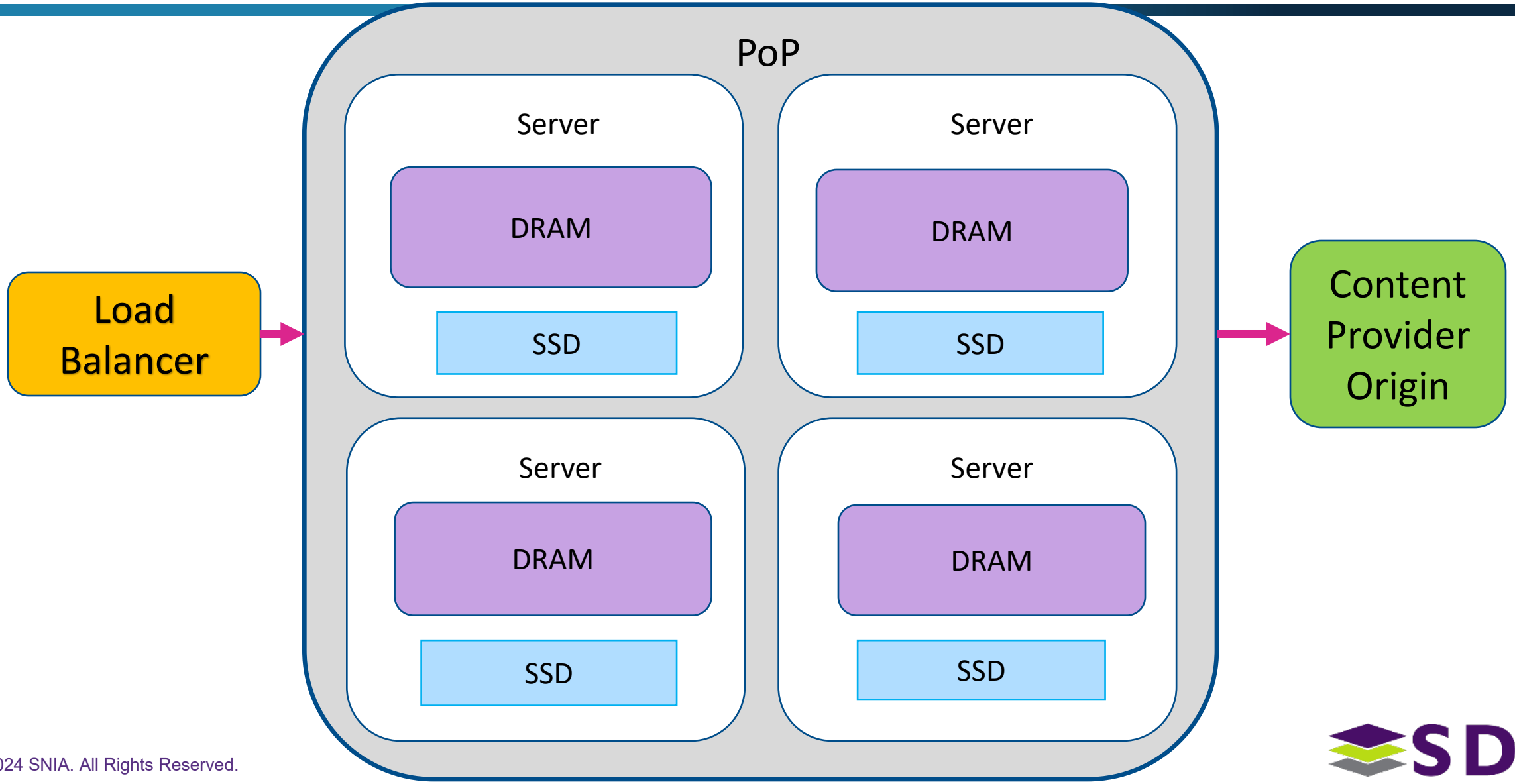
CDN PoP – Compute + Memory + Storage



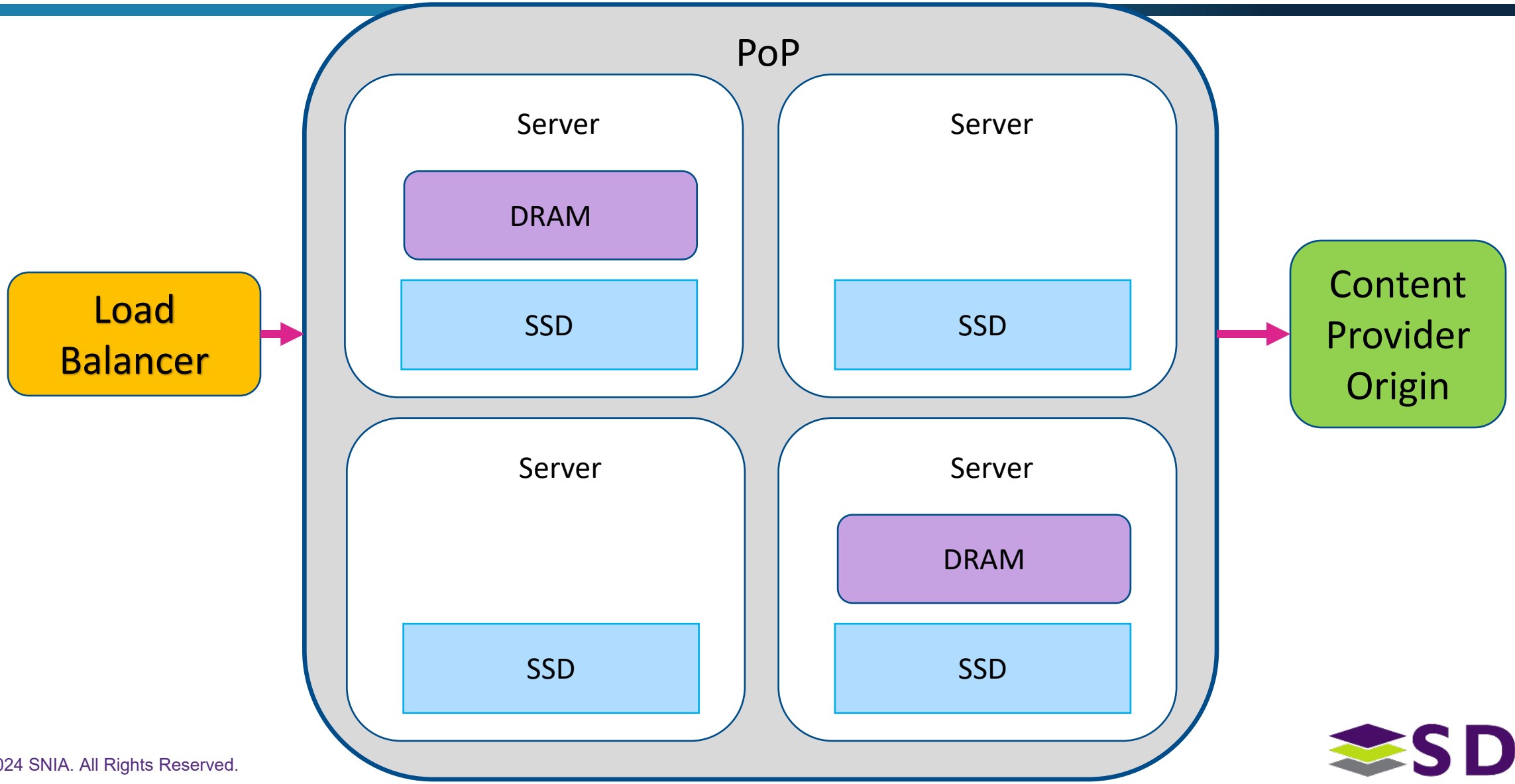
CDN PoP – Compute + Memory + Storage



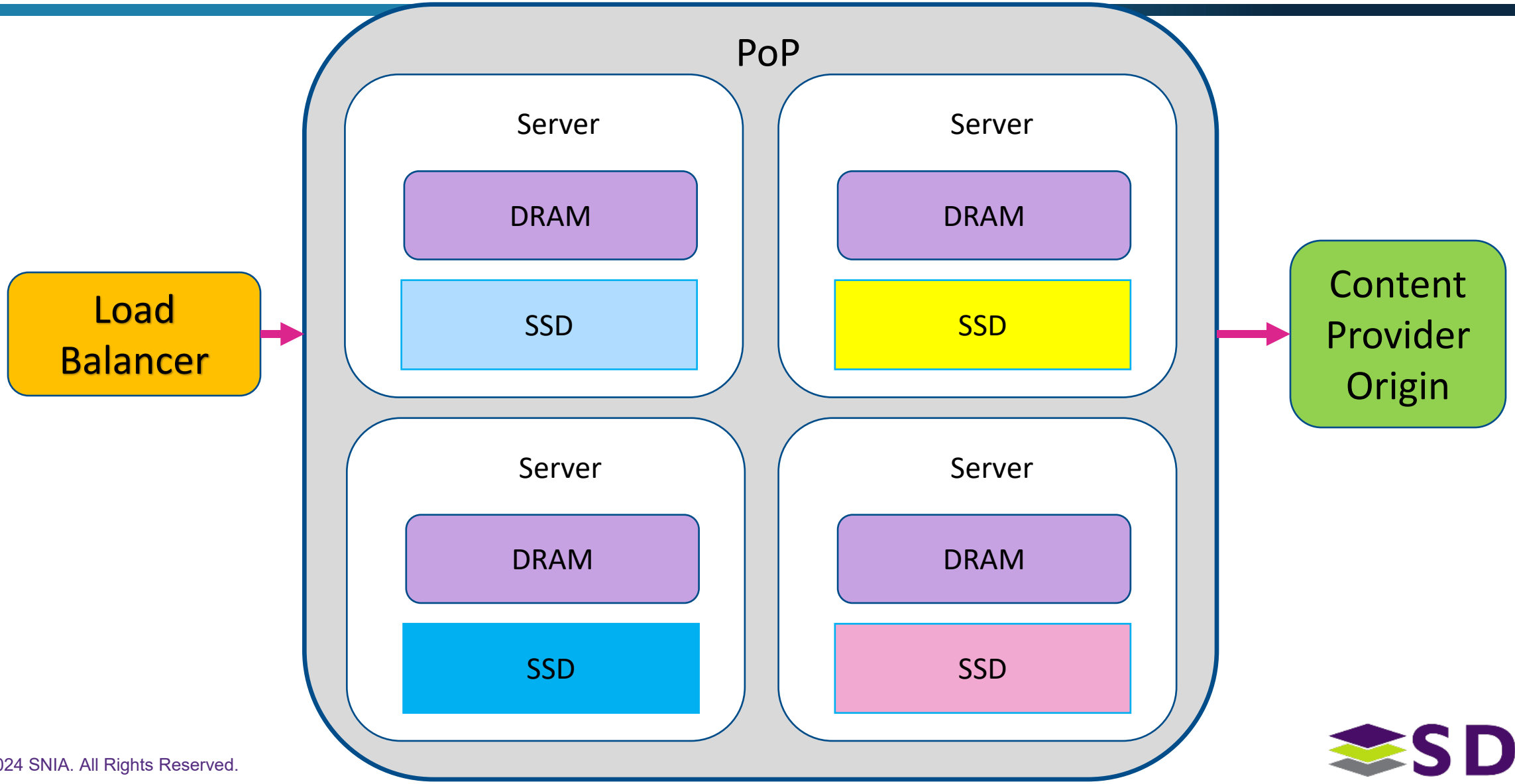
CDN PoP – Compute + Memory + Storage



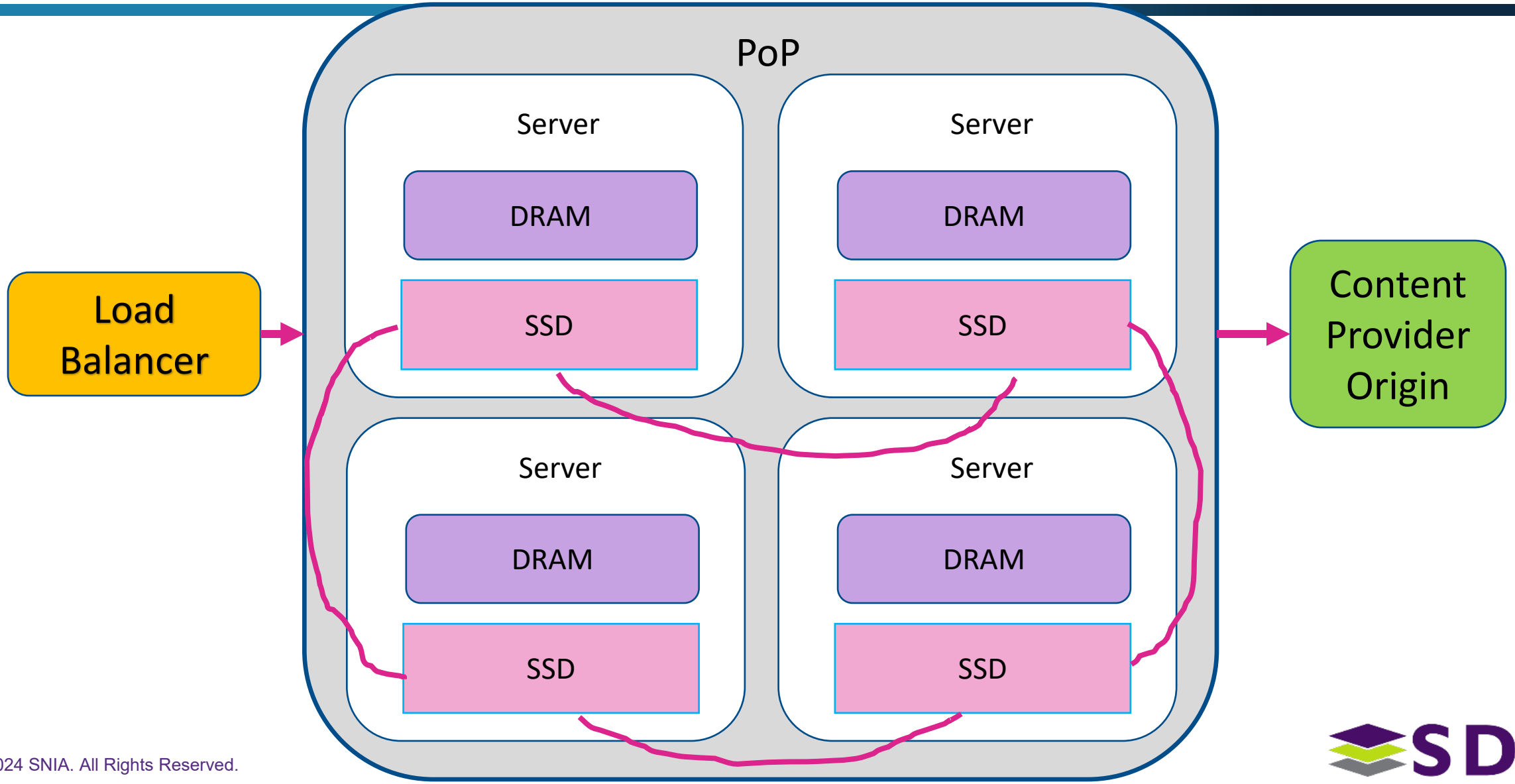
CDN PoP – Compute + Memory + Storage



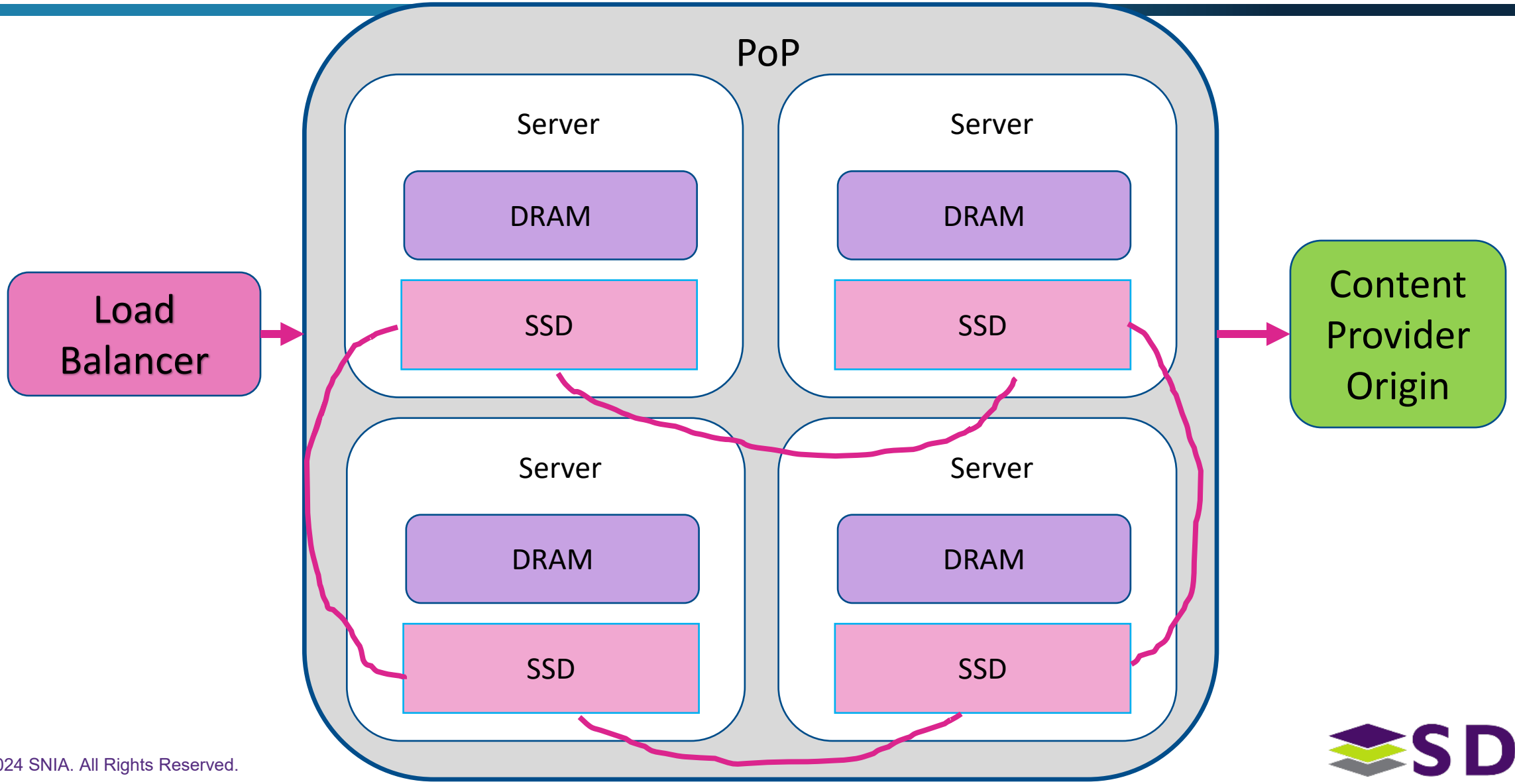
CDN PoP – Compute + Memory + Storage



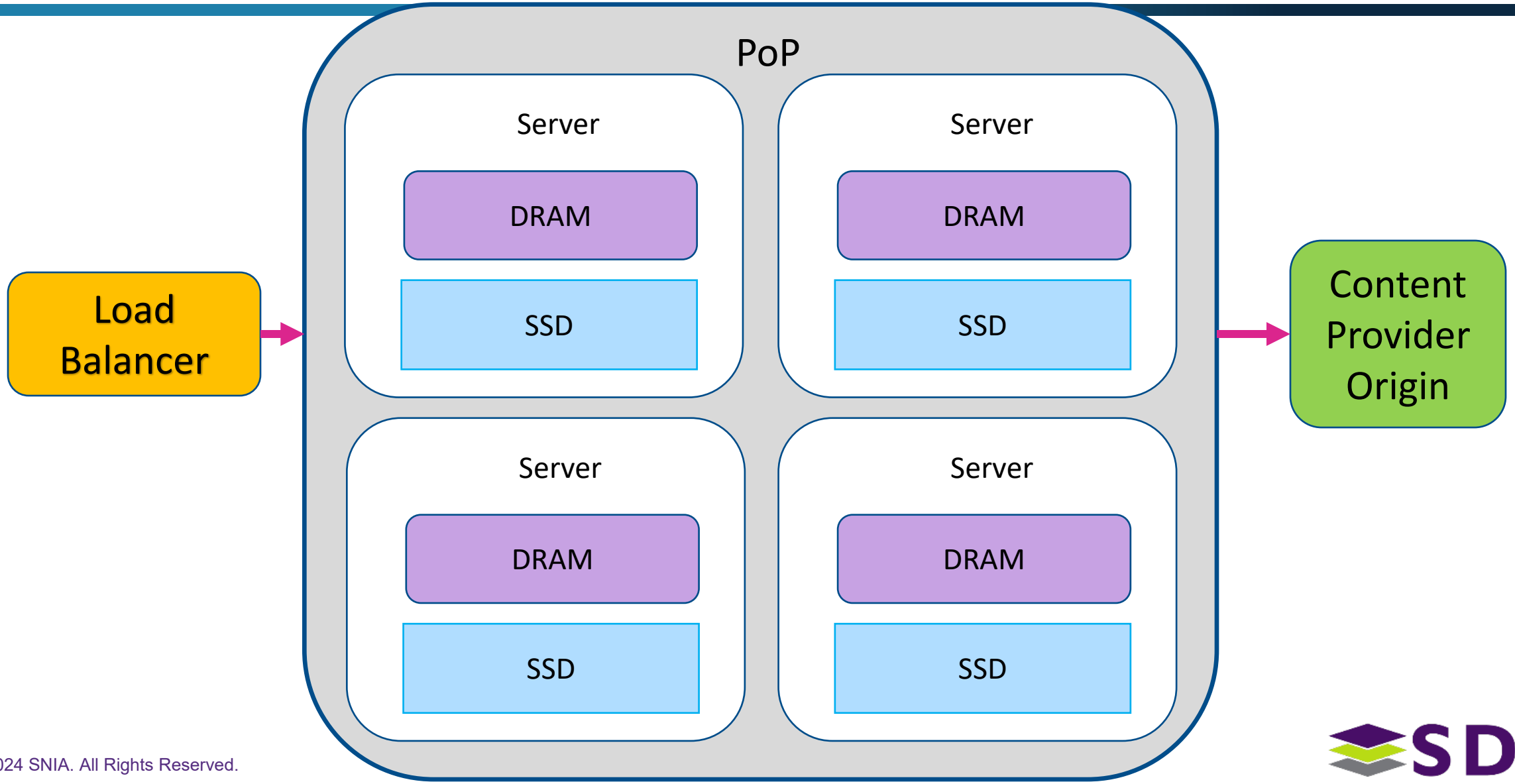
CDN PoP – Compute + Memory + Storage



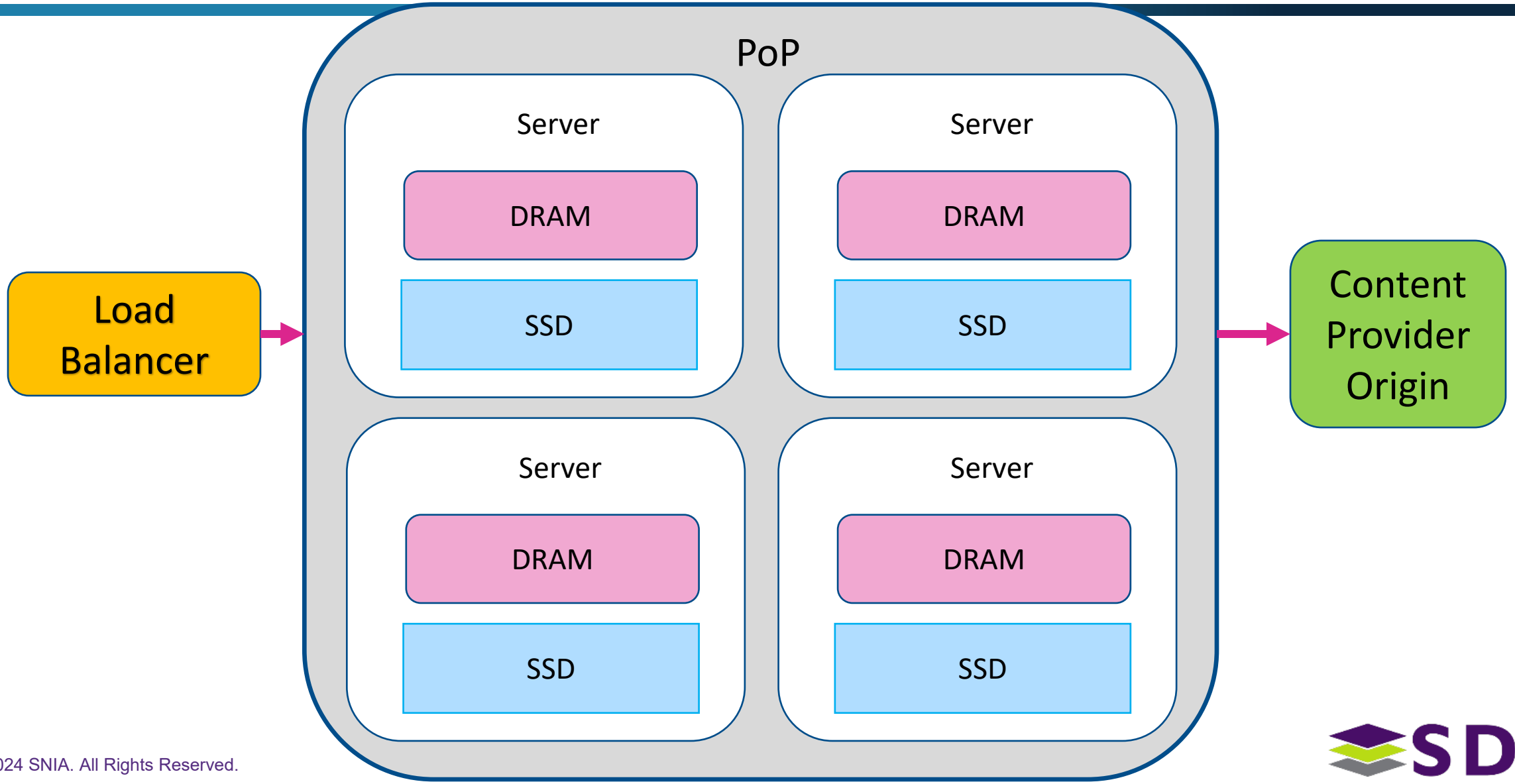
CDN PoP – Compute + Memory + Storage



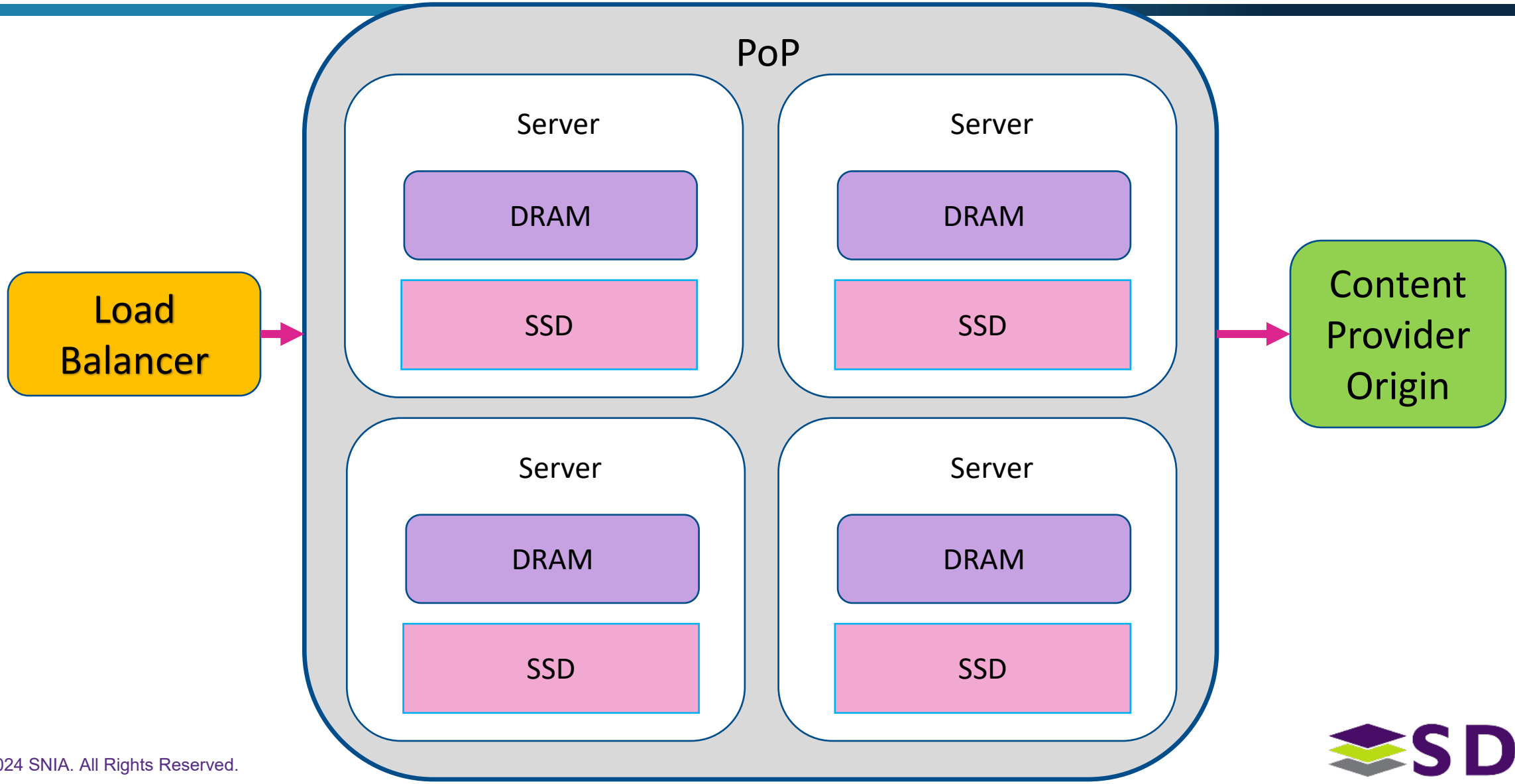
CDN PoP – Compute + Memory + Storage



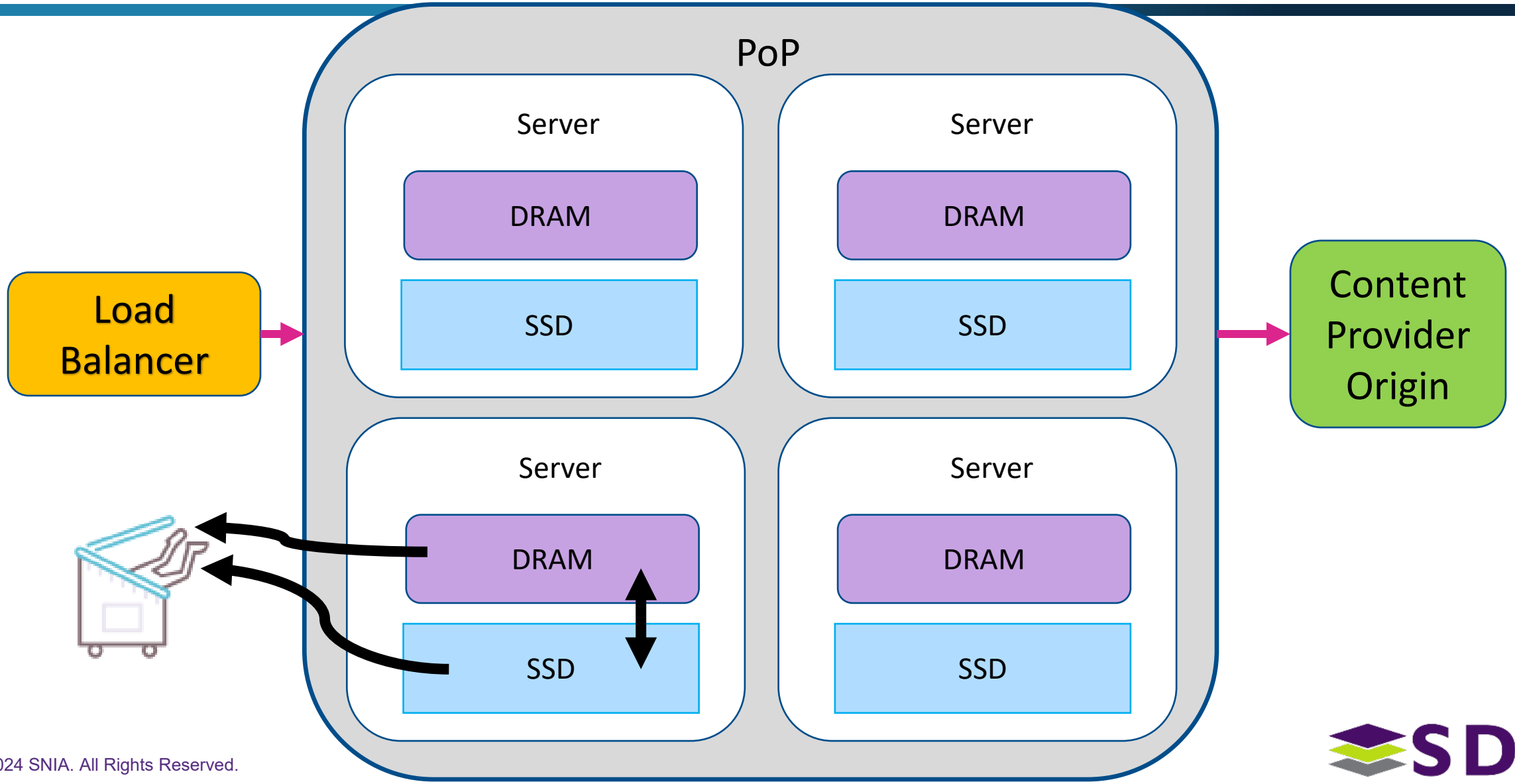
CDN PoP – Compute + Memory + Storage



CDN PoP – Compute + Memory + Storage



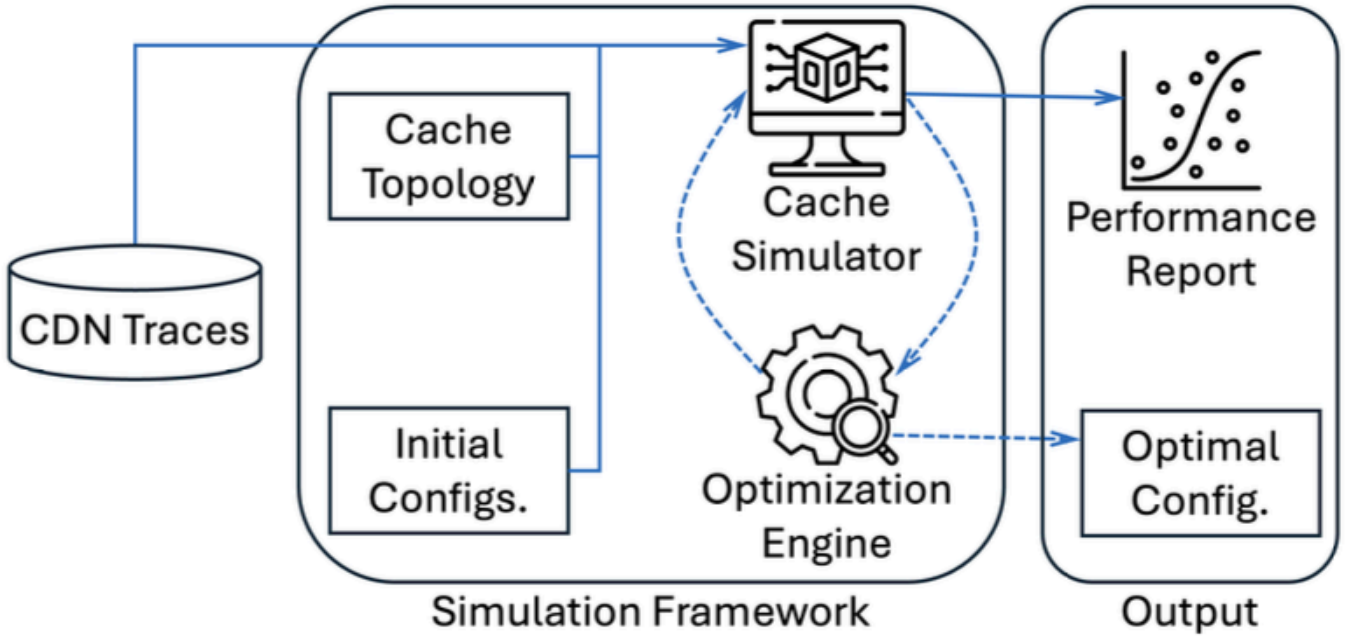
CDN PoP – Compute + Memory + Storage



Case Studies

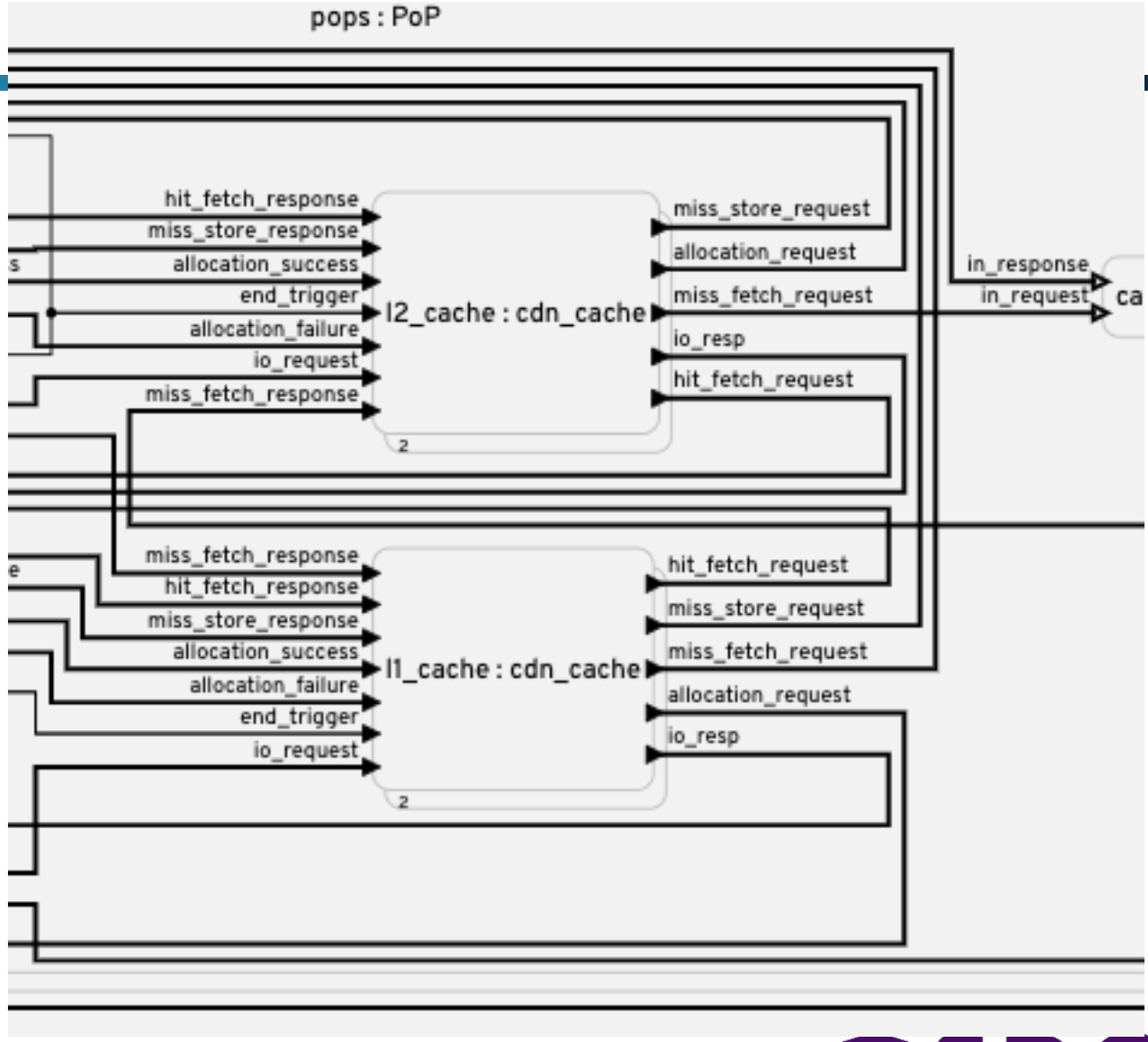
How to Build a Model

- Conceptual view of the design



Details, Details

- Cheaper, faster and more flexible than hardware models
- Each component can be modeled
- Variables are easy to introduce



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);
    =}
```


Case Studies

- Cloudflare and Wikimedia
- Genuine workloads tested
- Variants of baseline algorithms
  - Number of L1 and L2 caches
  - Promotion and demotion policies
  - Eviction policies of L1 and L2
  - DRAM/SSD ratios
  - Load balancer algorithm
- ~125k variants, each run against a day's worth of traffic

Variables

```
storage_capacity:
  value: [549755813888]
cache_size_ratio:
  value: [0.001, 0.005, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5]
store_on_miss:
  value: ["false"]
store_from_origin:
  value: ["true"]
move_on_hit_source:
  value: ["false"]
move_on_hit_sink:
  value: ["false"]
move_on_hit_count:
  value: [1]
eviction_types:
  value: [FIFO, SIEVE, CLOCK, LRU]
eviction_methods:
  value: ["&lru_eviction_methods"]
page_size:
  value: [4096]
l2_server_lb:
  name: L2_SERVER_LOADBALANCER
selection_types:
  value: [USR_DEF_SELECTION]
selection_methods:
  value: ["&url_hash_lb_methods"]
l2_servers:
  value: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 30]
  name: L2_SERVER
  switch:
    value: [enable]
  storage_media:
    value: [ssd_lite]
  storage_capacity:
    value: [549755813888]
  store_on_miss:
    value: ["false"]
  store_from_origin:
    value: ["true"]
  move_on_hit_source:
    value: ["false"]
  move_on_hit_sink:
    value: ["false"]
  eviction_types:
    value: [FIFO, SIEVE, CLOCK, LRU]
  eviction_methods:
    value: ["&lru_eviction_methods"]
  page_size:
    value: [4096]
pop_serializer:
  name: POP_SERIALIZER
data_mover:
  name: POP_L1_TO_L2_MOVER
  switch:
    value: [disable]
```

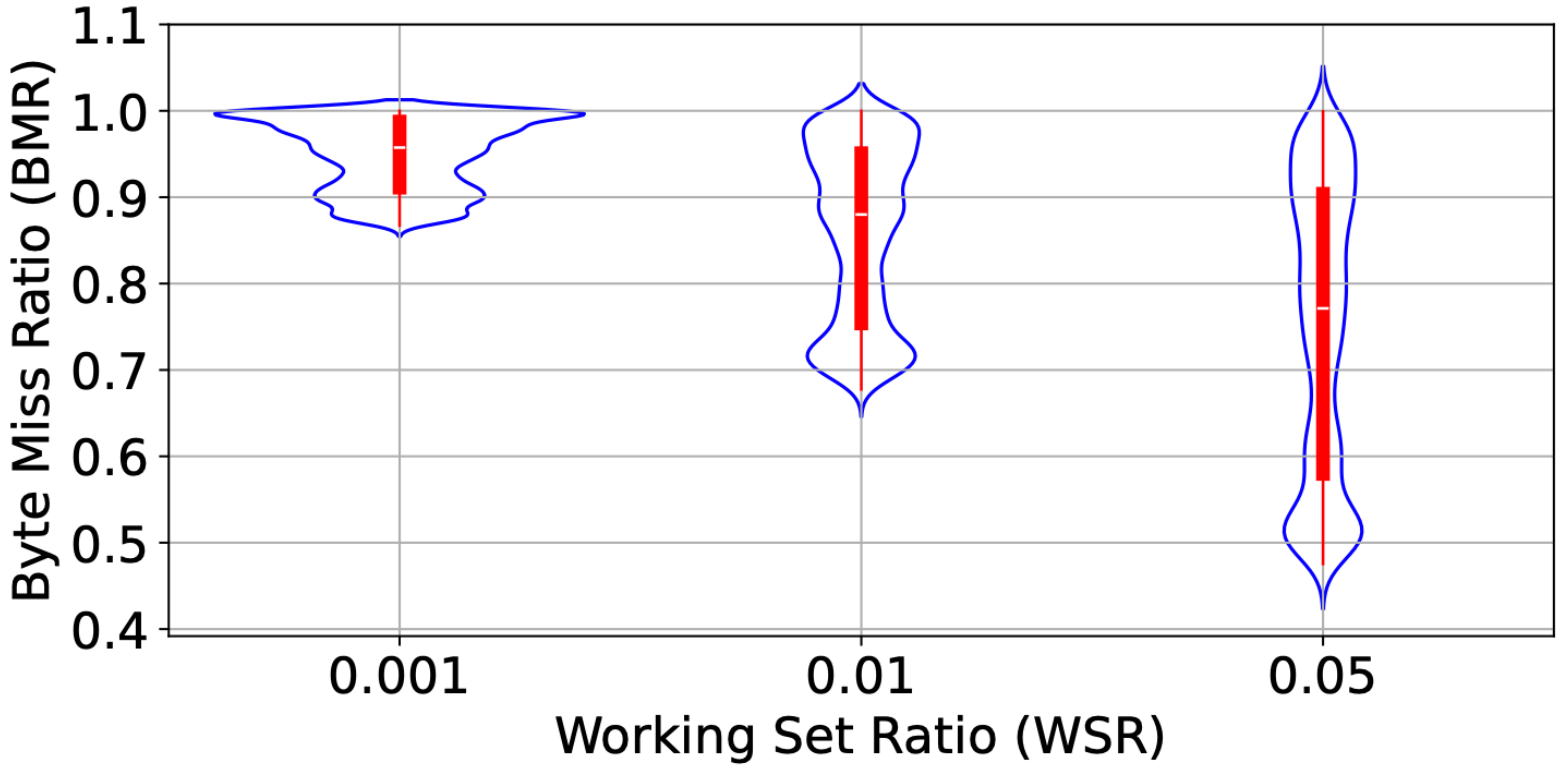
Wiki Topology

```
storage_capacity:
  value: [549755813888]
cache_size_ratio:
  value: [0.001, 0.005, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5]
store_on_miss:
  value: ["false"]
store_from_origin:
  value: ["true"]
move_on_hit_source:
  value: ["true"]
move_on_hit_sink:
  value: ["false"]
move_on_hit_count:
  value: [1, 2, 3, 4]
eviction_types:
  value: [FIFO, SIEVE, CLOCK, LRU]
eviction_methods:
  value: ["&lru_eviction_methods"]
page_size:
  value: [4096]
l2_server_lb:
  name: L2_SERVER_LOADBALANCER
selection_types:
  value: [LB_1_ON_1_WIRING_DEFAULT]
selection_methods:
  value: ["&url_hash_lb_methods"]
l2_servers:
  value: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 30]
  name: L2_SERVER
  switch:
    value: [enable]
  storage_media:
    value: [ssd_lite]
  storage_capacity:
    value: [549755813888]
  store_on_miss:
    value: ["false"]
  store_from_origin:
    value: ["false"]
  move_on_hit_source:
    value: ["false"]
  move_on_hit_sink:
    value: ["true"]
  eviction_types:
    value: [FIFO, SIEVE, CLOCK, LRU]
  eviction_methods:
    value: ["&lru_eviction_methods"]
  page_size:
    value: [4096]
pop_serializer:
  name: POP_SERIALIZER
data_mover:
  name: POP_L1_TO_L2_MOVER
  switch:
    value: [enable]
```

Cloudflare Topology

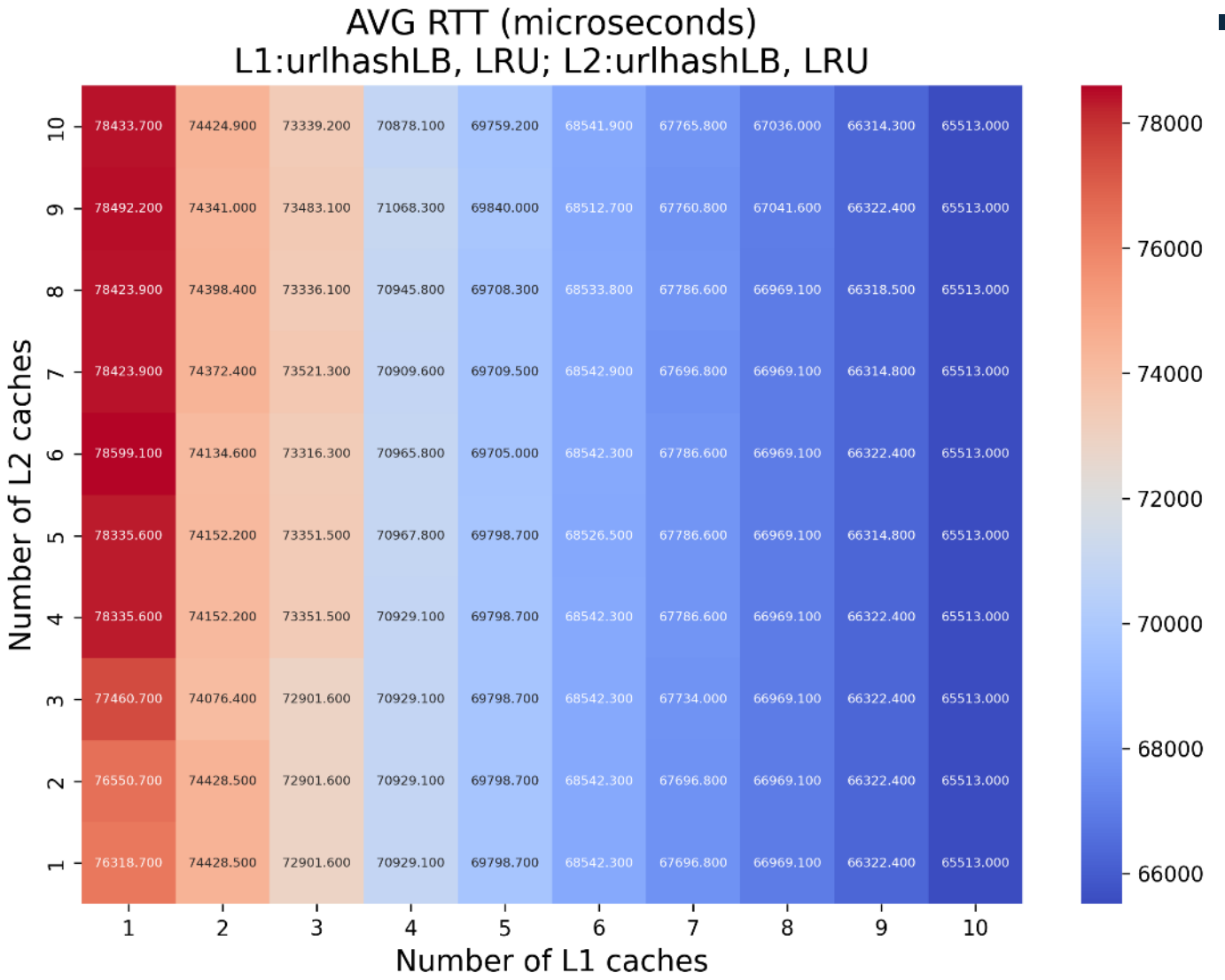
Lots of Variability, Lots of Results

- Multi-objective Optimization

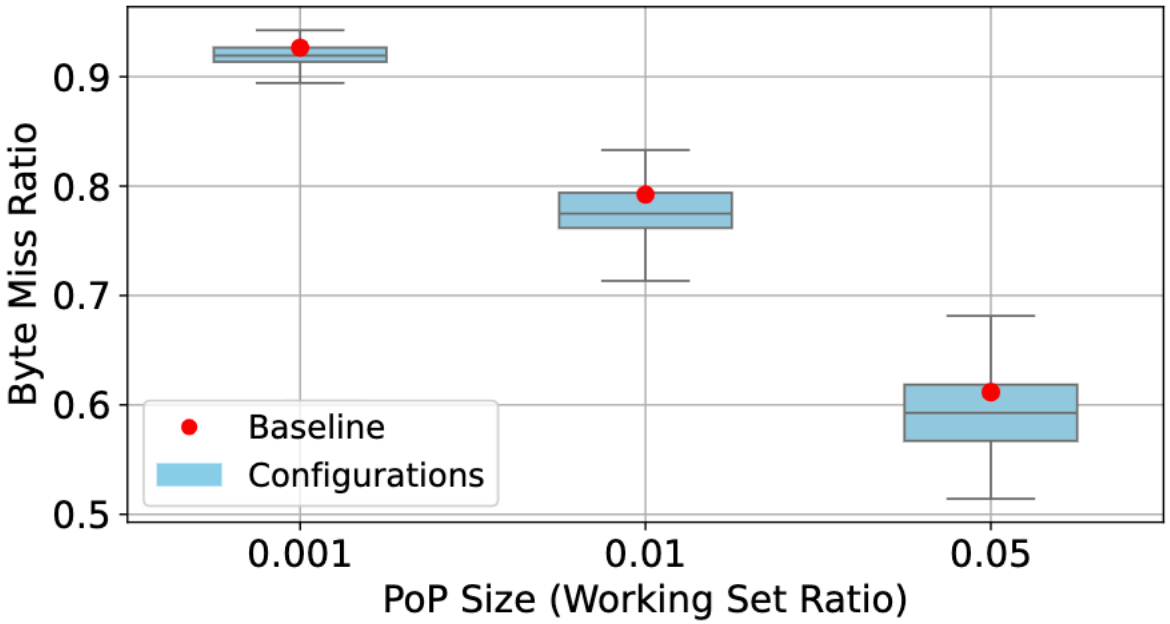


Results Are Easy to Compare

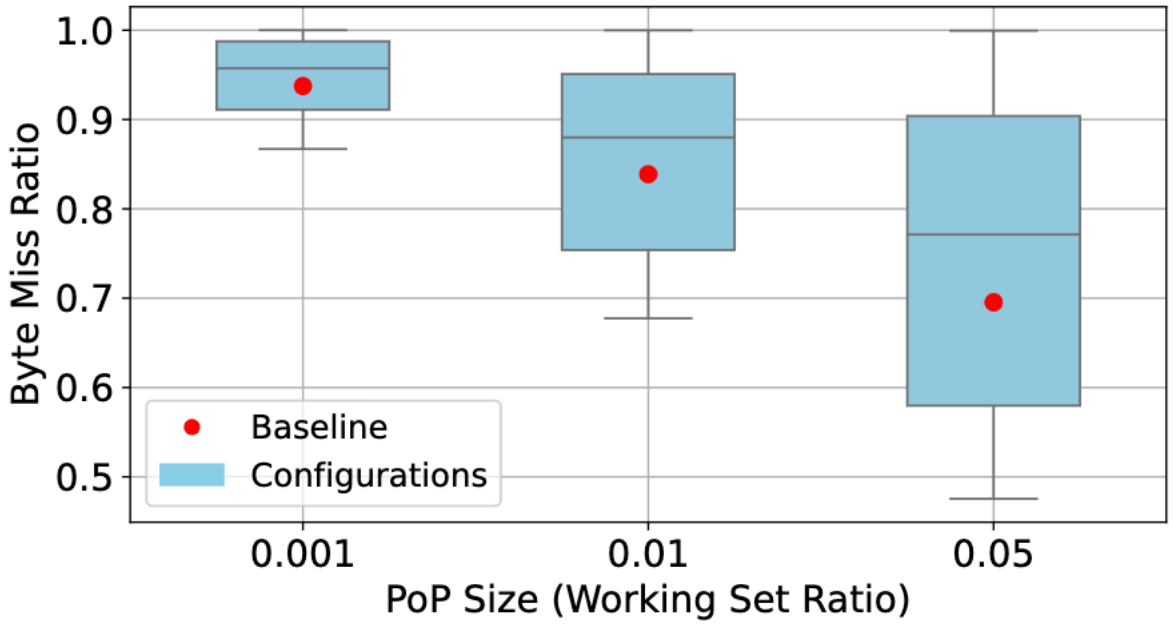
- See many thousands of runs
- More variables = more options



Case Study Results

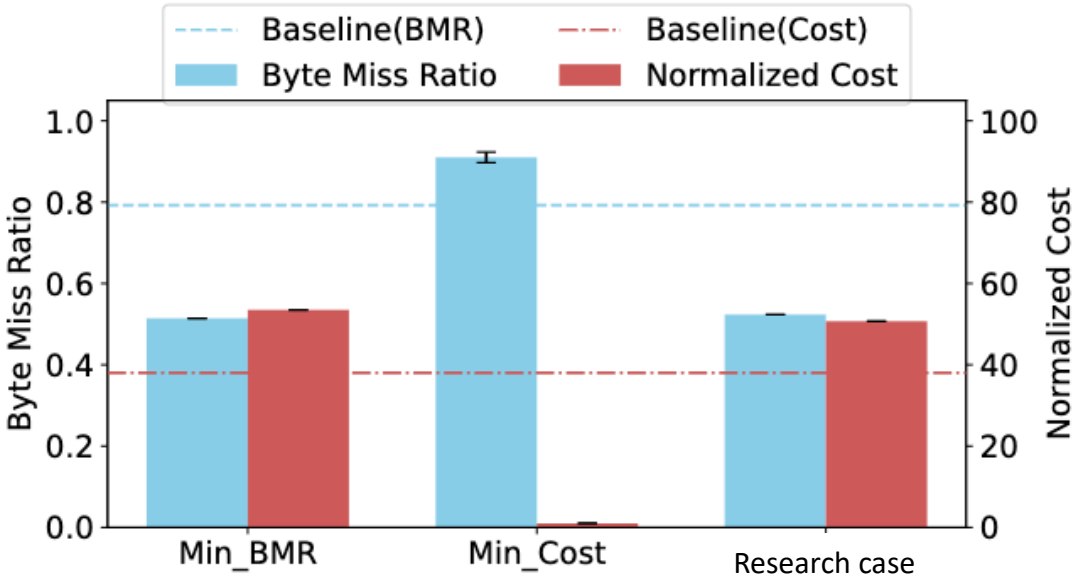


Wikimedia

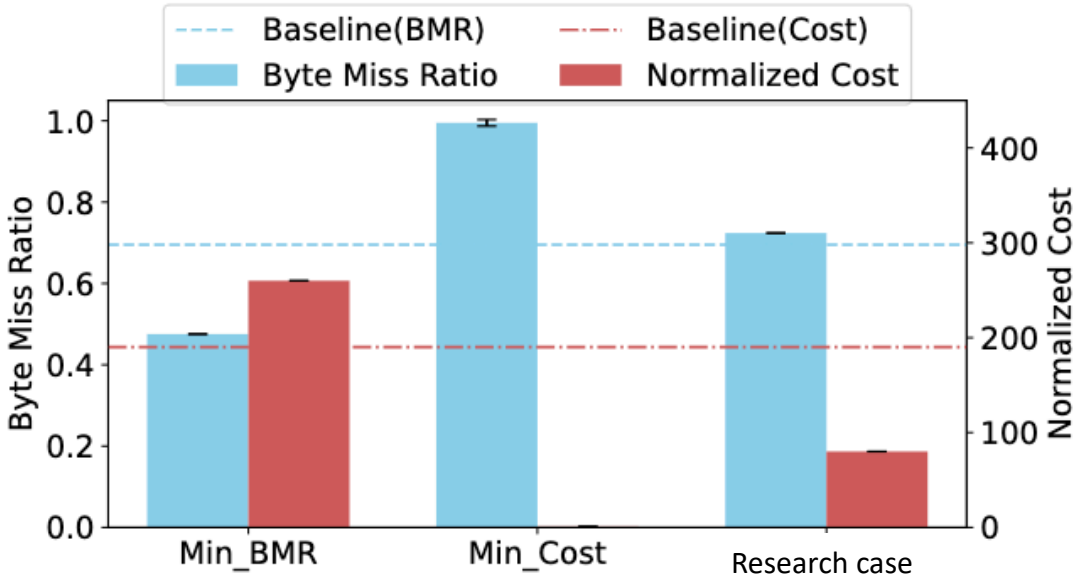


Cloudflare

Case Study Results

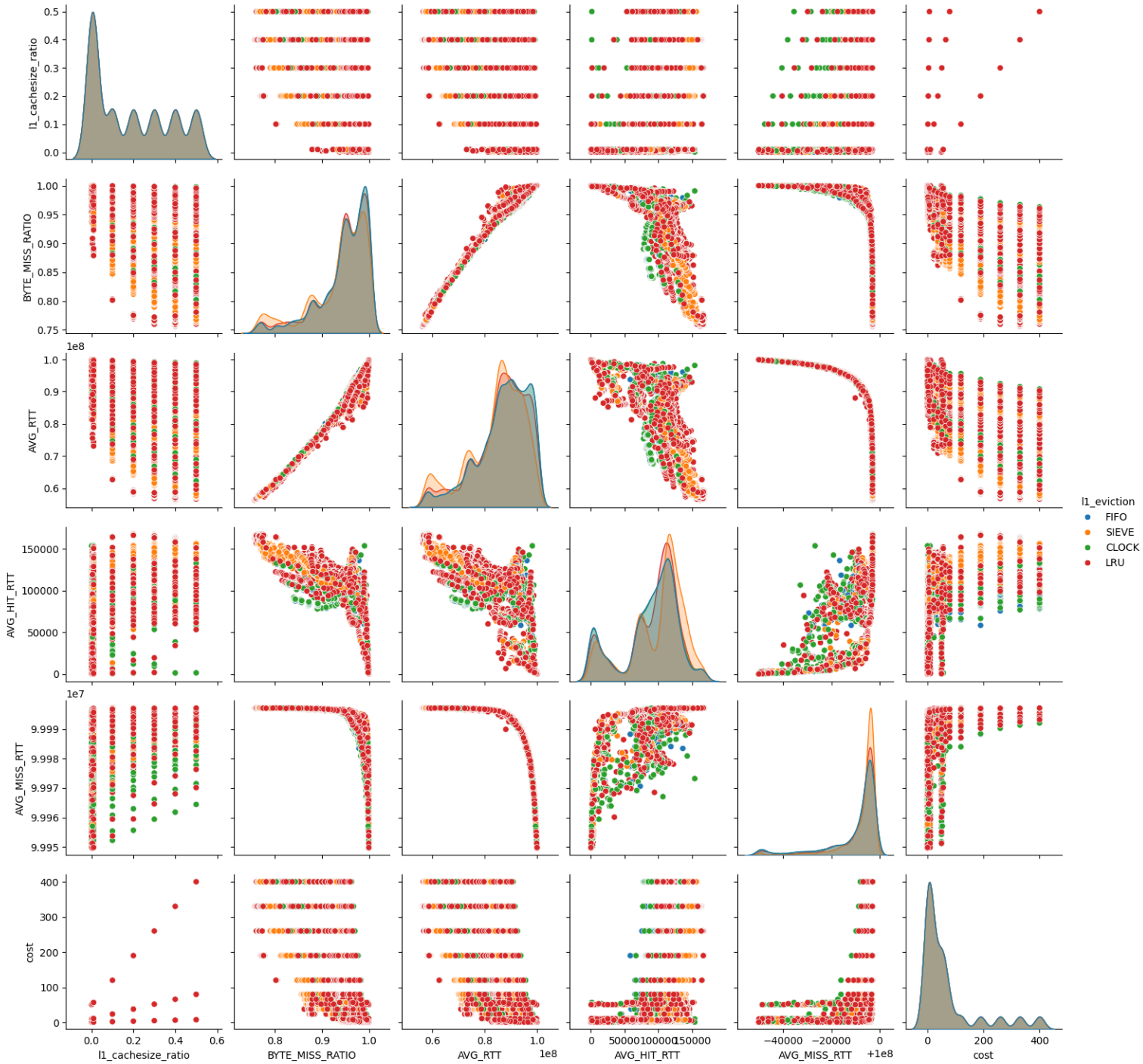


Wikimedia

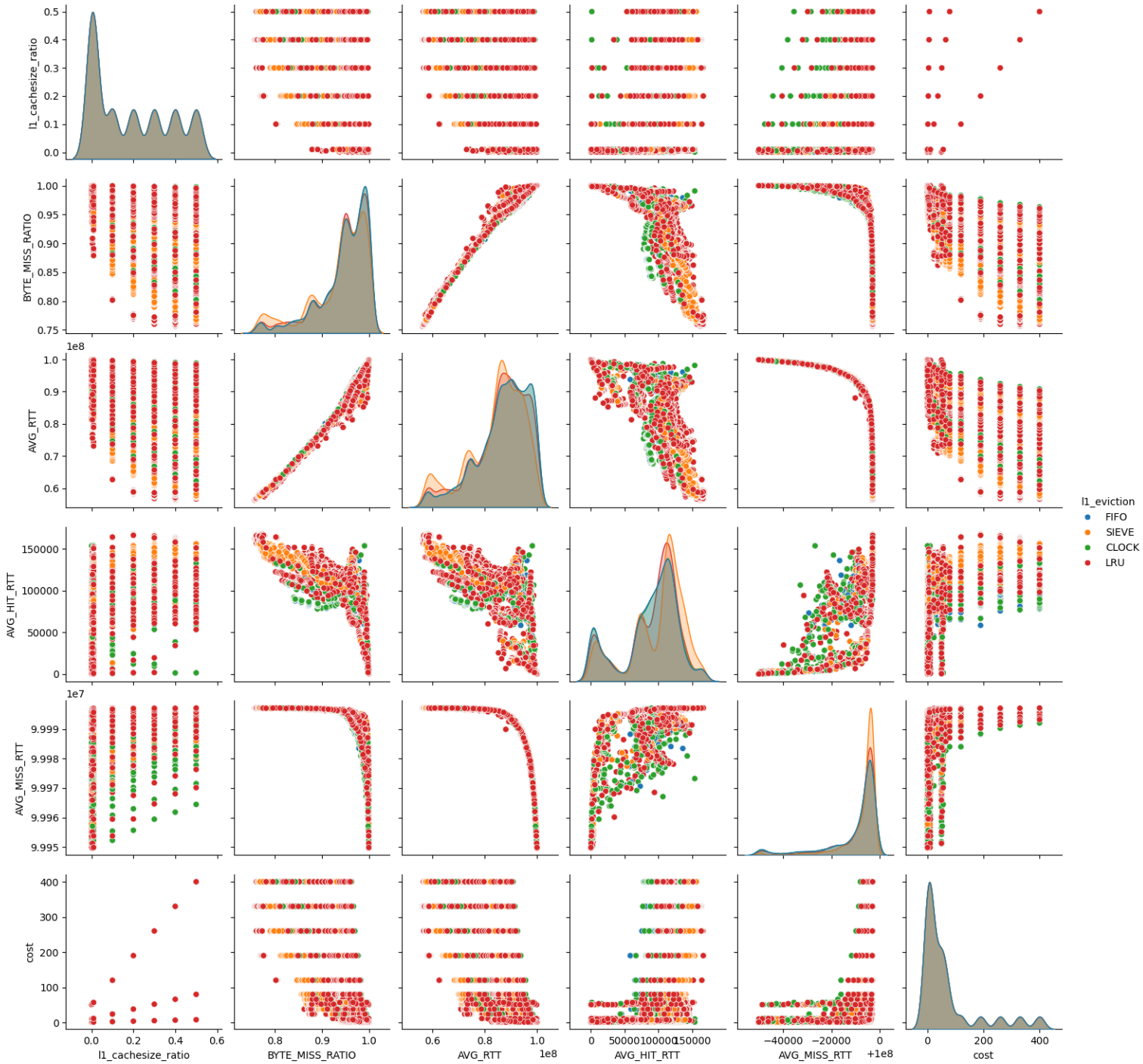


Cloudflare

Results

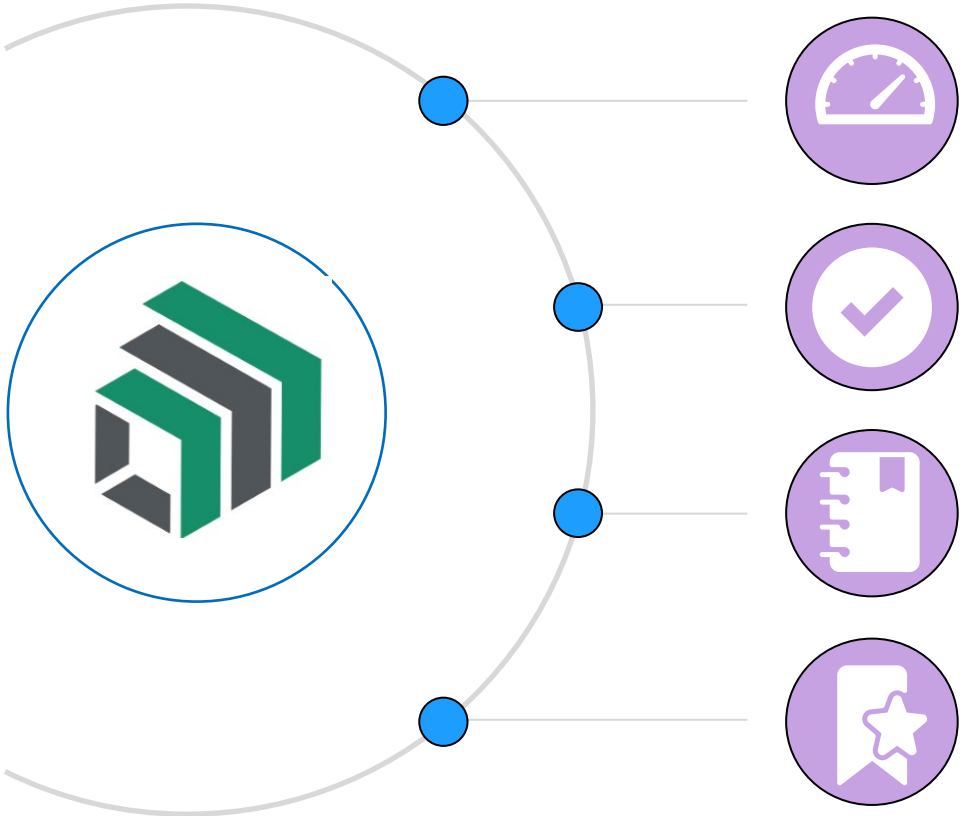


Results



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.

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%+**

I Learned Something Today

1. Caching is used for more than storage
2. Real world CDN implementations can be improved with low effort
3. Large scale simulations can drive huge gains in efficiency and cost

Please take a moment to rate this session.

Your feedback is important to us.

