Elasticsearch as a Service on Private Cloud

LINE+ Cloud Service

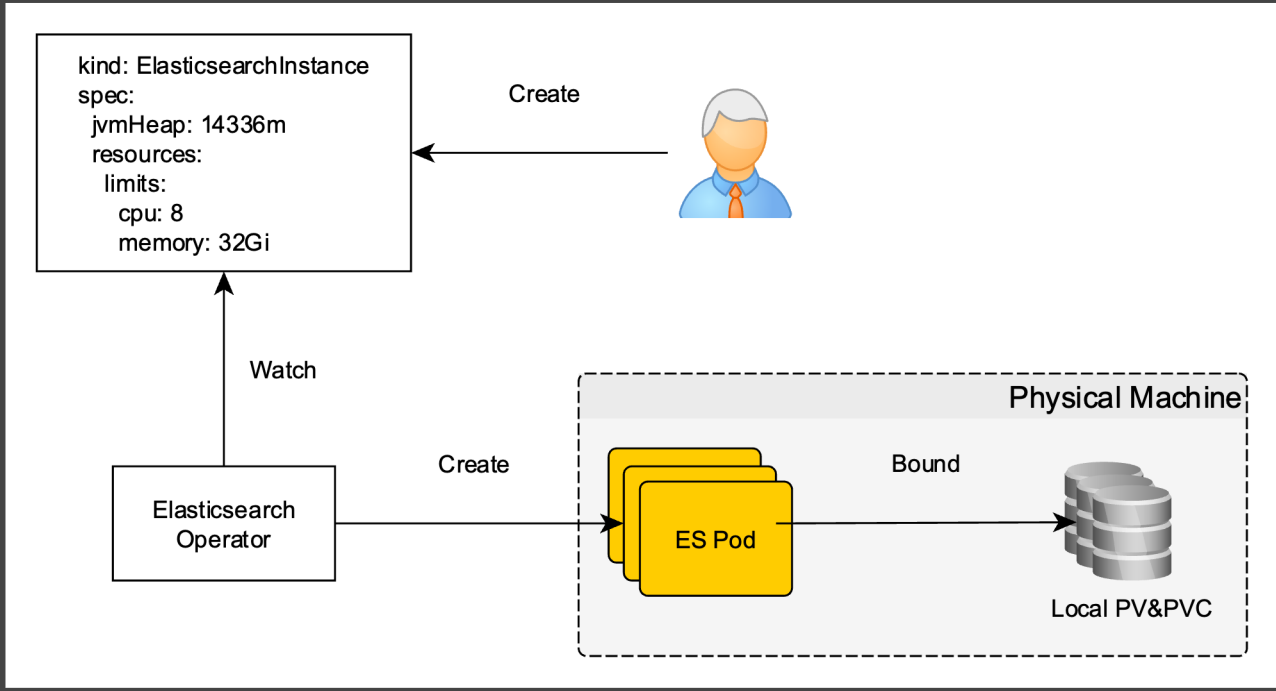
LINE

2022.11.1

Clusters	1,500+
Nodes	13,000+

Beginning Architecture

Whole Elasticsearch Clusters on Single Kubernetes Cluster



500 Elasticsearch clusters with 270 Baremetal worker nodes

Declarative approach with Custom Resources

Troubles…

Don’t Put All Your Eggs in One Basket!

Kubernetes outage resulted in outages of all Elasticsearch clusters

Traffic burst of one Elasticsearch cluster caused outage of other services

Troubles…

Don’t Put All Your Eggs
in One Basket!

Kubernetes outage resulted in outages of
all Elasticsearch clusters

Traffic burst of one Elasticsearch cluster caused outage of other services

Requirement 1

Although Kubernetes cluster is down, it should not affect to Elasticsearch clusters

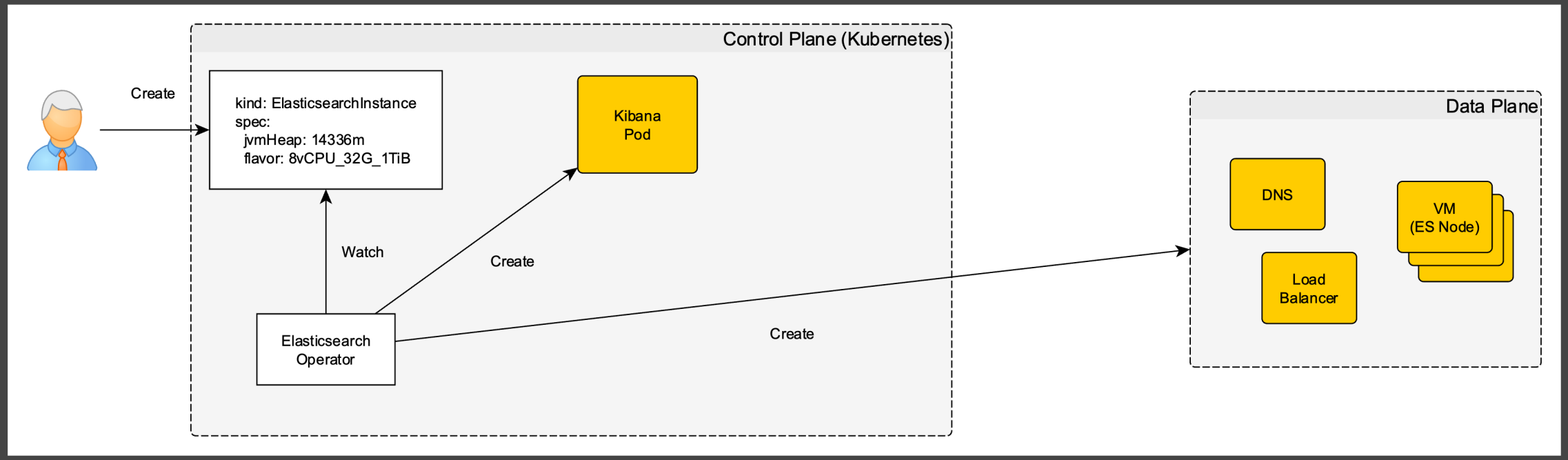
Requirement 1

Although Kubernetes cluster is down, it should not affect to Elasticsearch clusters

Requirement 2

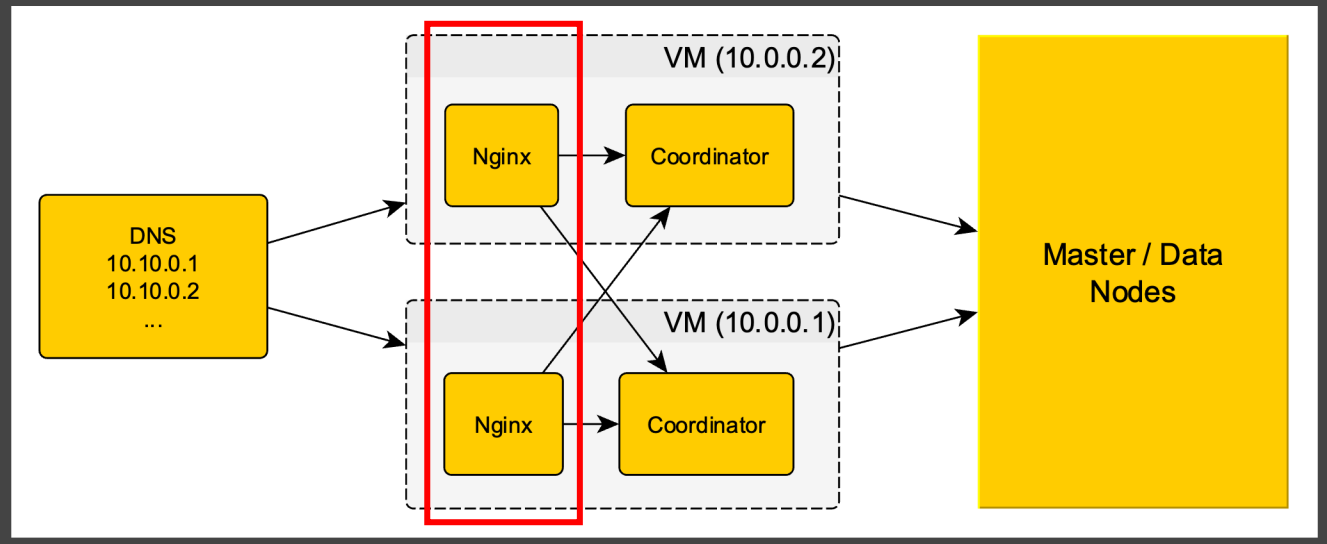
Keep declarative approach of Kubernetes

New Mixed-in Architecture



Control-Plane on Kubernetes, Data-Planes on Virtual Machines

Endpoint Architecture for ES

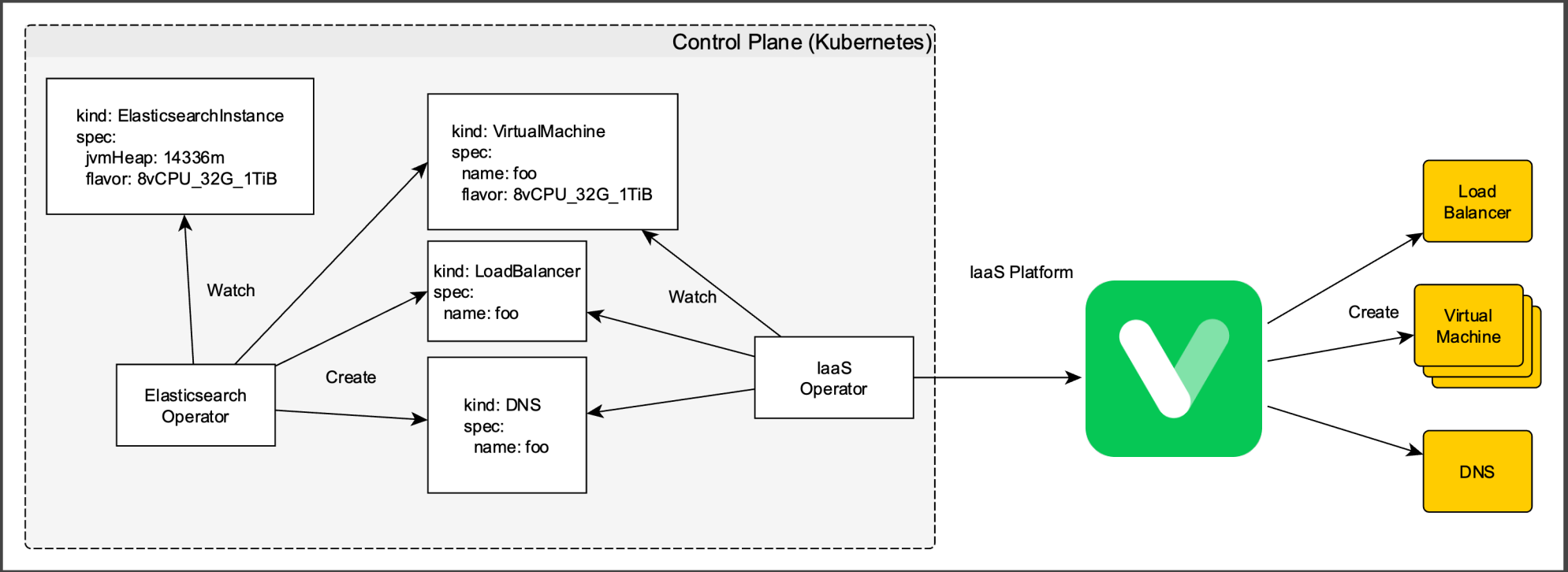


Use Nginx for collecting logs and metrics for each request

DNS round robin + Nginx in clusters for logging

Depend on retrying logic in client

Declarative Approach



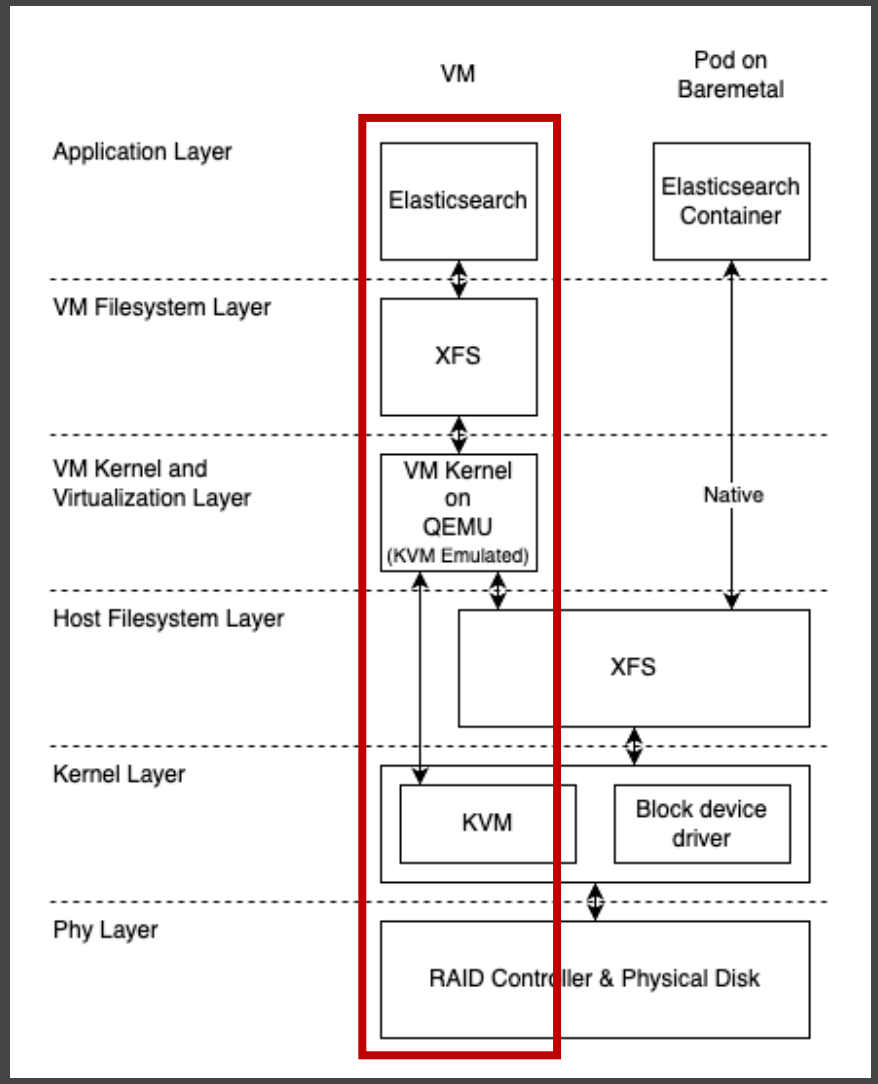
Develop operator to make IaaS resources(VM, DNS, Load Balancers…) declarative objects

Fulfilled requirements, another problem arises

The Limitations

Full-virtualization is really hurt to us

Increasing latency

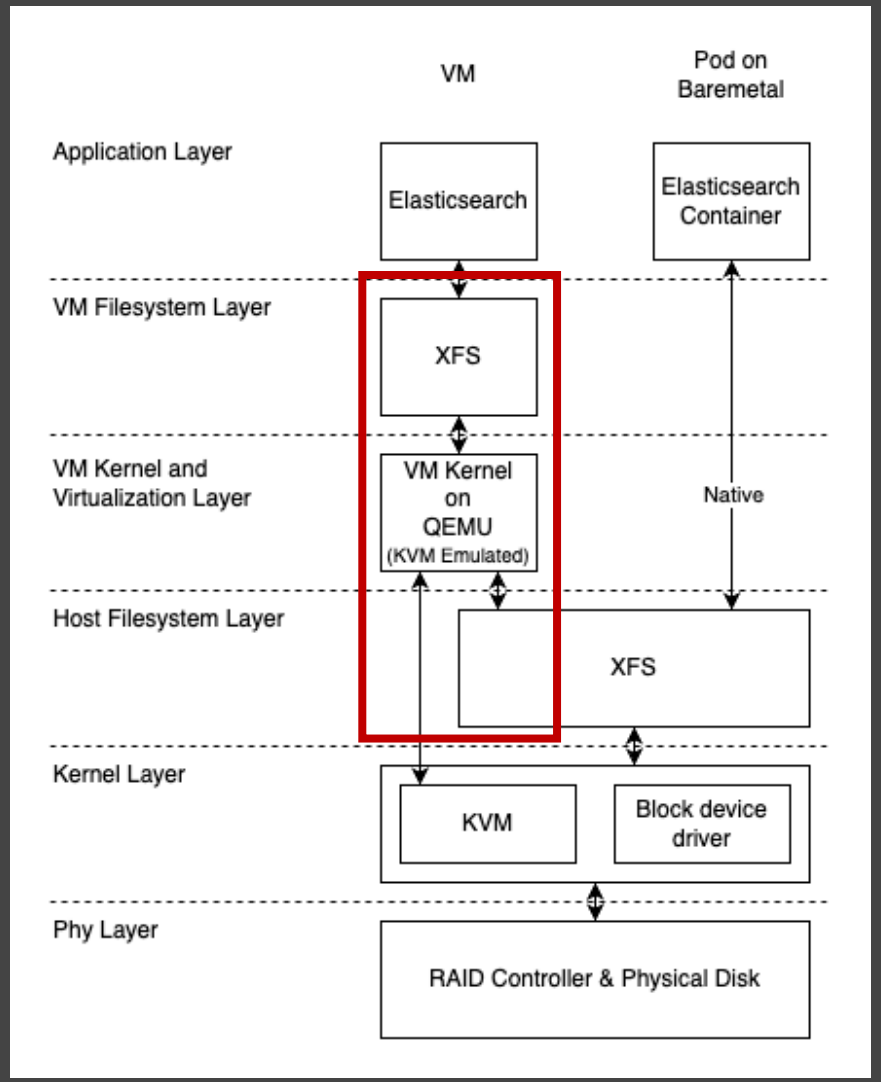


The Limitations

Full-virtualization is really hurt to us

Increasing latency

Decreasing I/O throughput



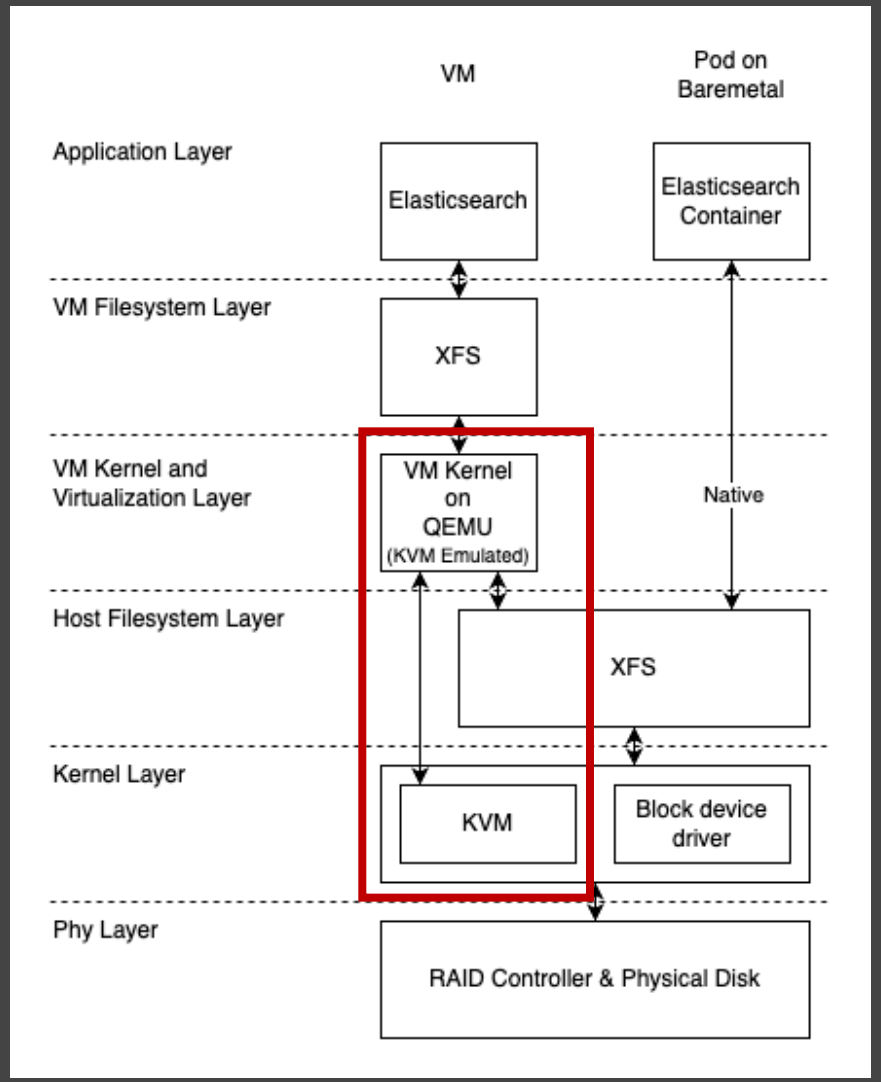
The Limitations

Full-virtualization is really hurt to us

Increasing latency

Decreasing I/O throughput

Bad neighborhood problem



The Limitations

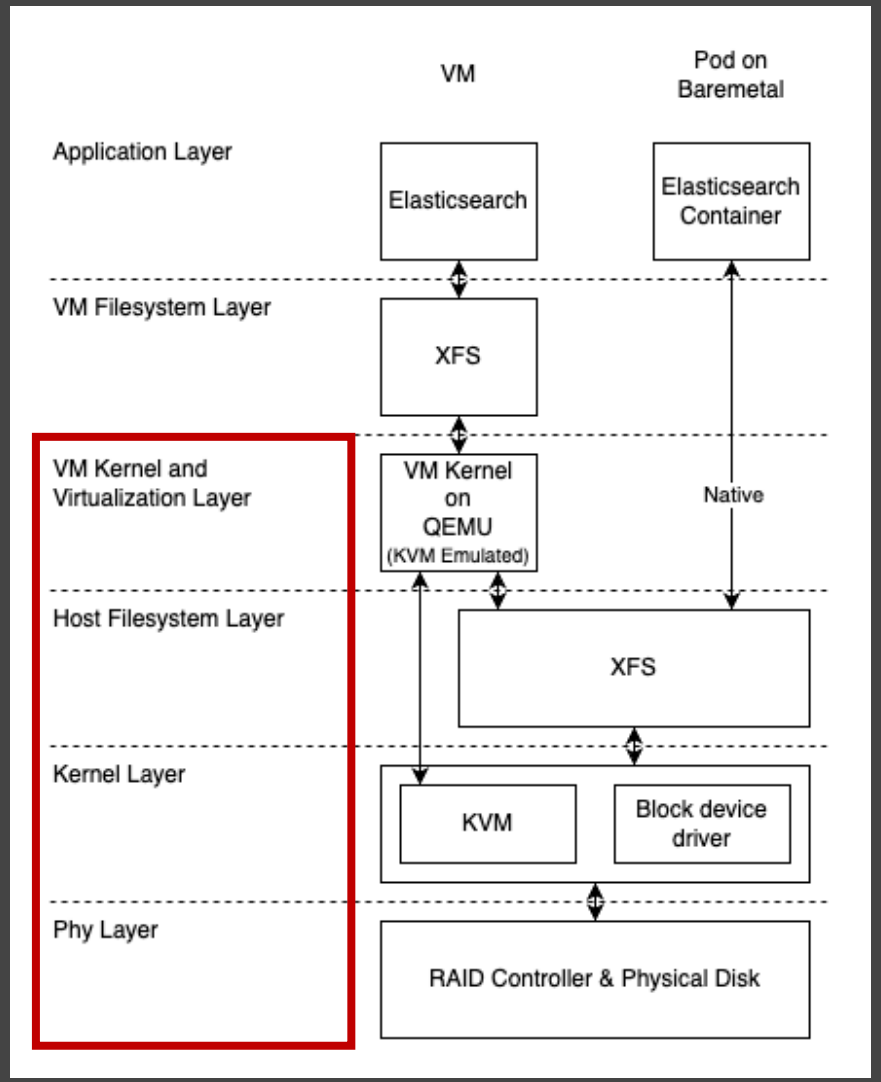
Full-virtualization is really hurt to us

Increasing latency

Decreasing I/O throughput

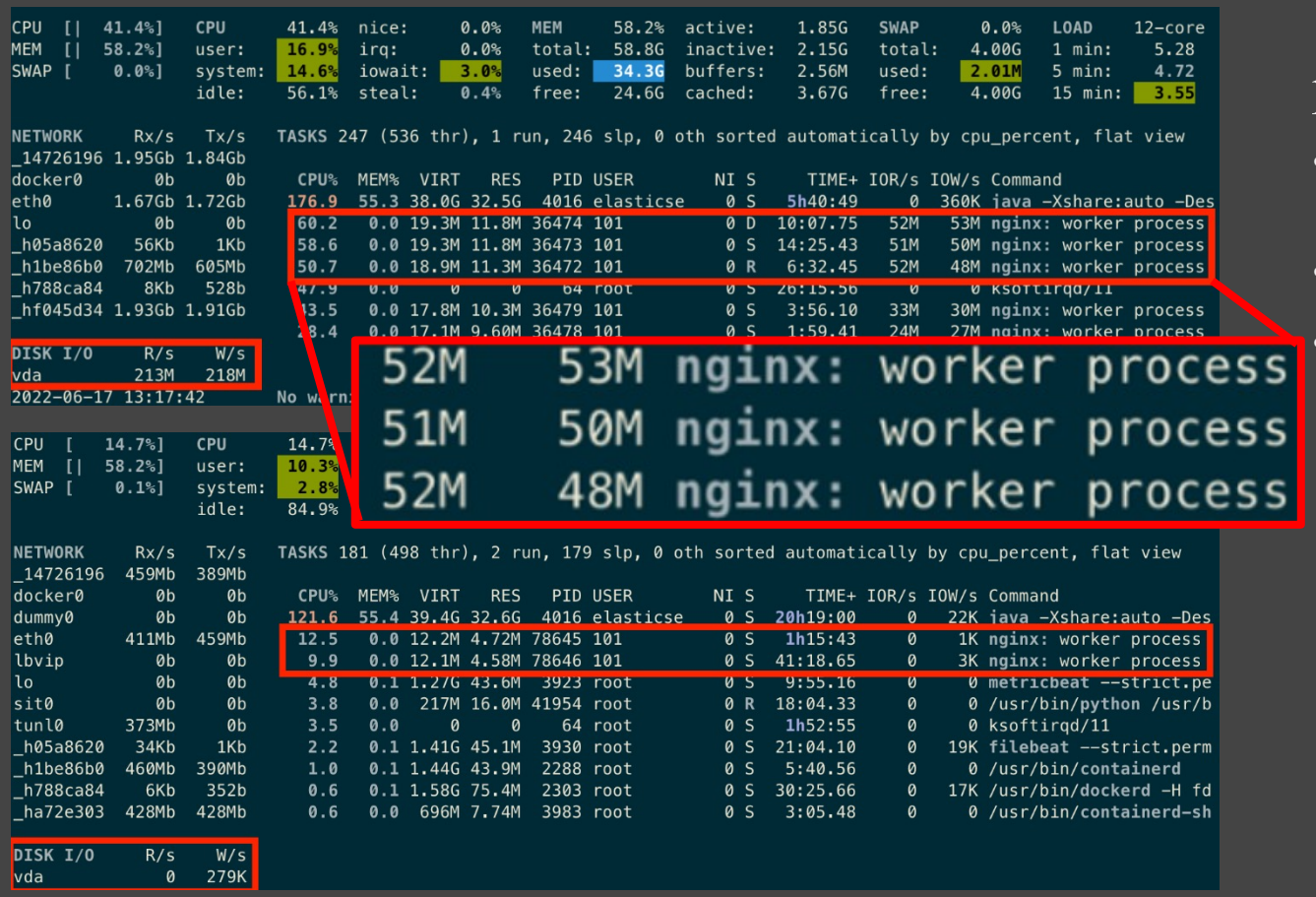
Bad neighborhood problem

Needs skill set to fine tune



Improve Performance (1)

Tuning Coordinator – Enhance Nginx Service Mesh Throughput



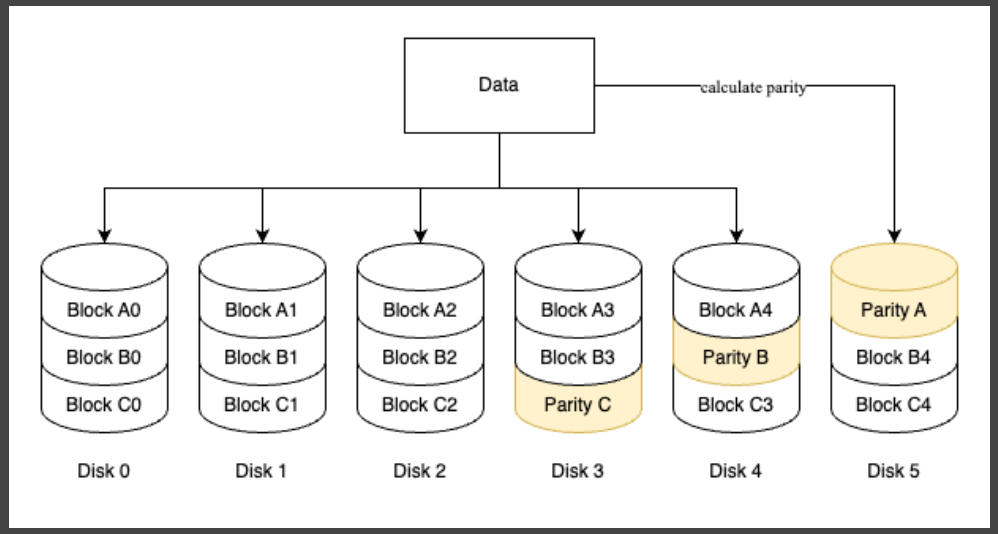
proxy_buffering and proxy_request_buffering

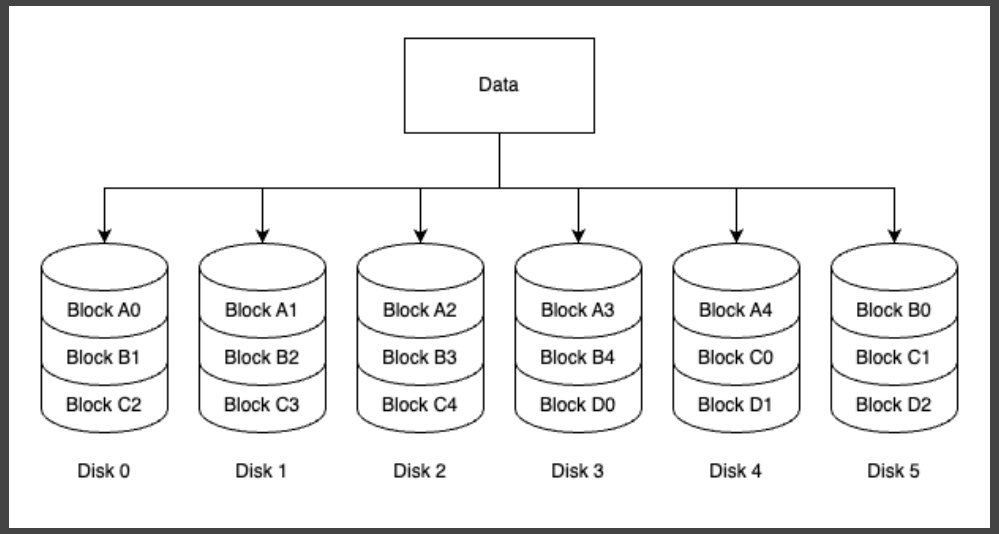
- Larger payloads could be buffered to disk when buffer memory size is smaller than response
- This makes Request-Resp action I/O-intensive
- **Performance largely hurts on high RPS with large request and response**

```
http {
    ...
    proxy_buffering off;
    proxy_request_buffering off;
    ...
}
```

Improve Performance (2)

RAID 0 – Why we don’t care about redundancy





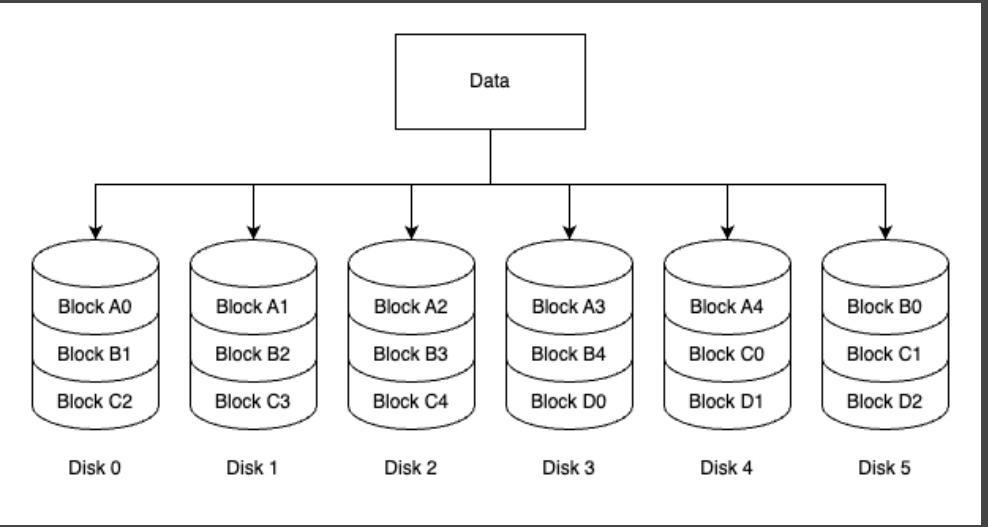
Improve Performance (2)

RAID 0 – Why we don't care about redundancy

Elasticsearch itself has features for high availability

- Elasticsearch has a replica shards
- Lost all VMs disk since RAID 0 has no fault tolerance

shards: 12 * 2| docs: 53,630 | size: 102.11MB	shards: 12 * 2| docs: 113,845 | size: 145.08MB	shards: 12 * 2| docs: 65,689 | size: 104.95MB
10 11	10 11	10 11
9 10	9 10	9 10
2 3	2 3	2 3



Improve Performance (2)

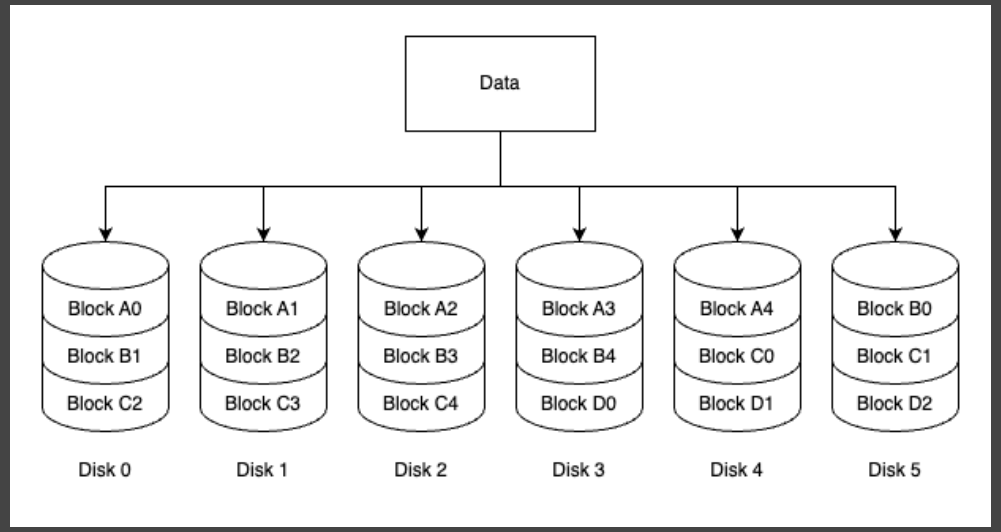
RAID 0 – Why we don't care about redundancy

Elasticsearch itself has features for high availability

- Elasticsearch has a replica shards
- Lost all VMs disk since RAID 0 has no fault tolerance

Some clusters need high indexing performance

- Heavy indexing requires high Disk I/O
- Ensure 34,000 IOPS for read and 22,500 IOPS for write in worst case



Improve Performance (2)

RAID 0 – Why we don't care about redundancy

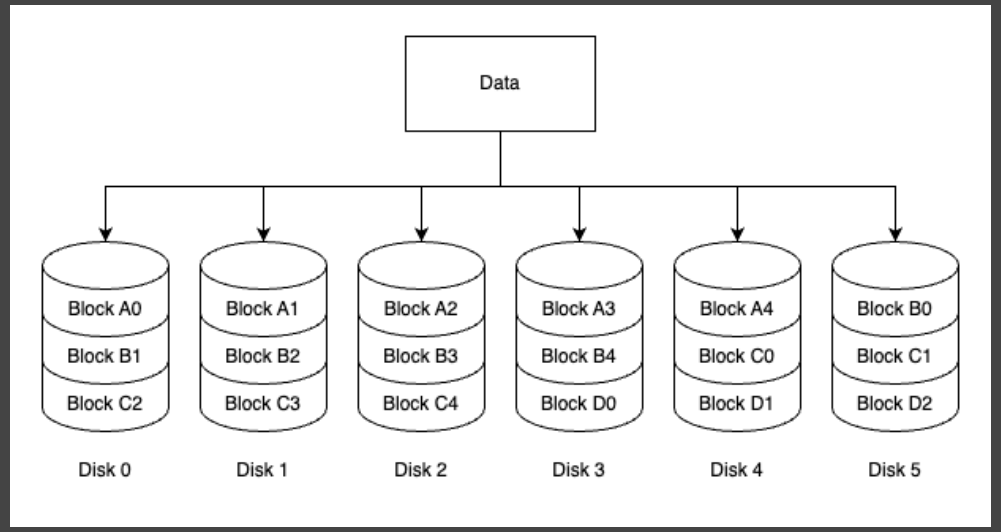
Elasticsearch itself has features for high availability

- Elasticsearch has a replica shards
- Lost all VMs disk since RAID 0 has no fault tolerance

Some clusters need high indexing performance

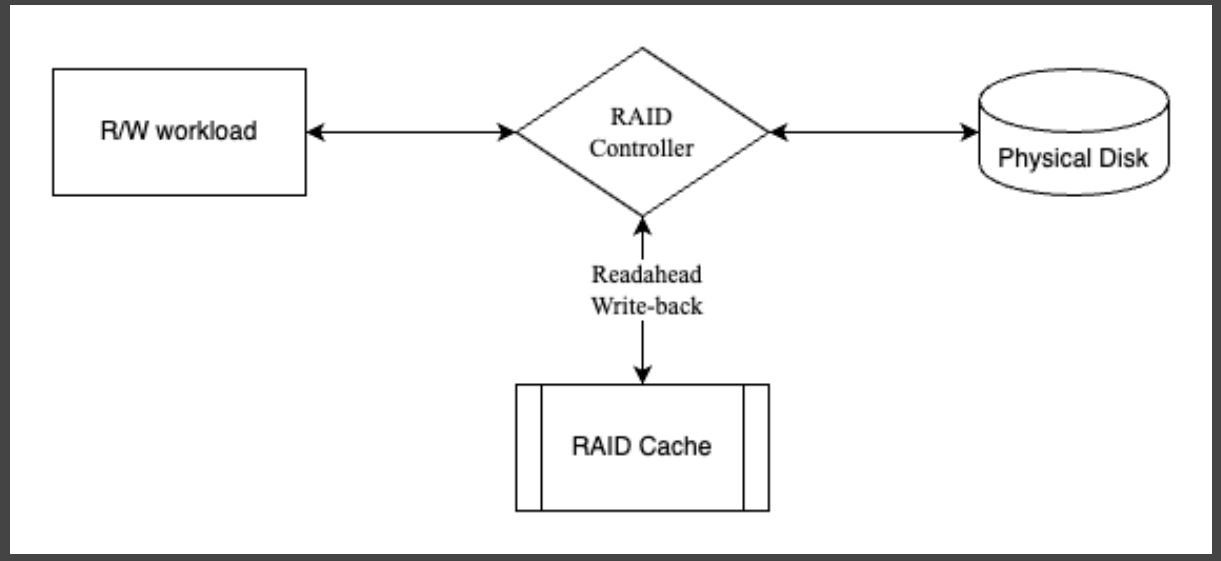
- Heavy indexing requires high Disk I/O
- Ensure 34,000 IOPS for read and 22,500 IOPS for write in worst case

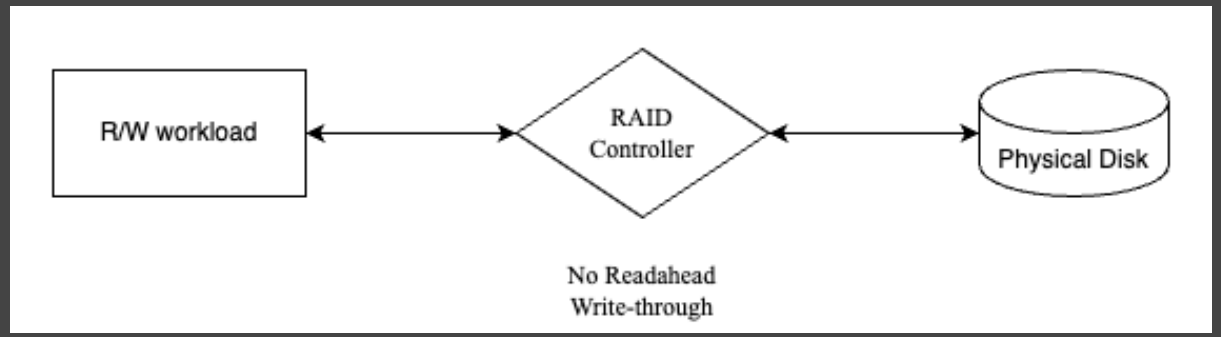
A way to keep redundancy with RAID 0 performance?



Improve Performance (3)

RAID 0 – Maximizing performance of RAID 0 SSD array



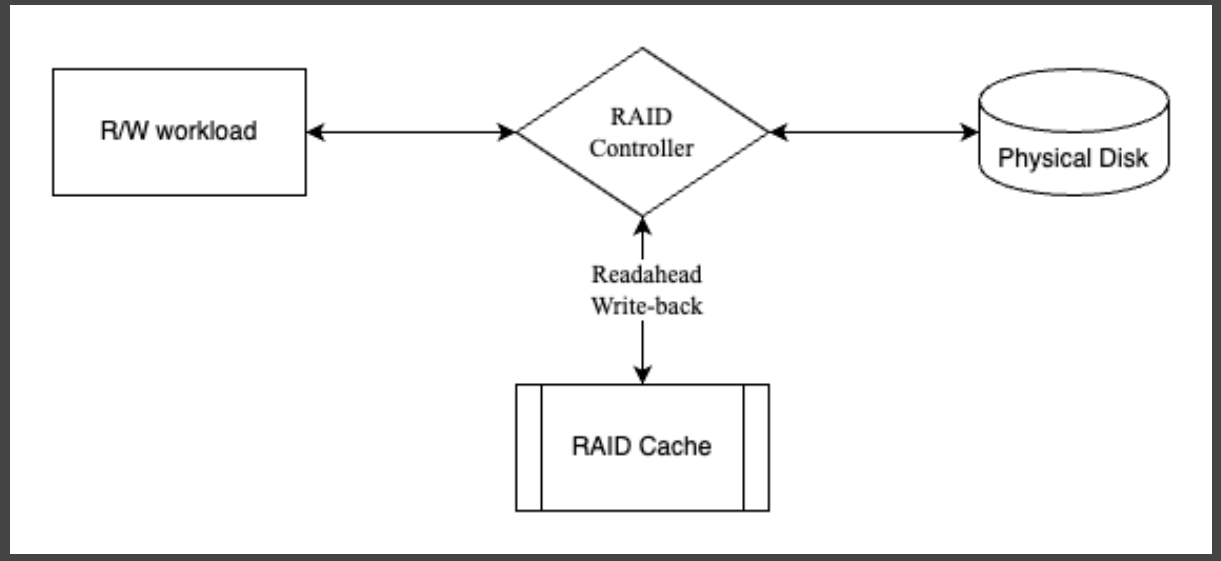


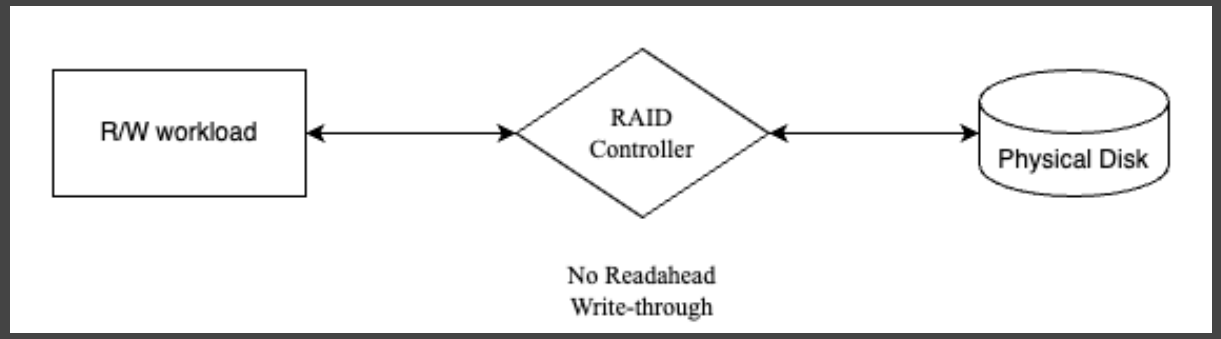
Write-back cache is not always effective

- Dramatically good effect on HDD, but not on SSD

Improve Performance (3)

RAID 0 – Maximizing performance of RAID 0 SSD array





Write-back cache is not always effective

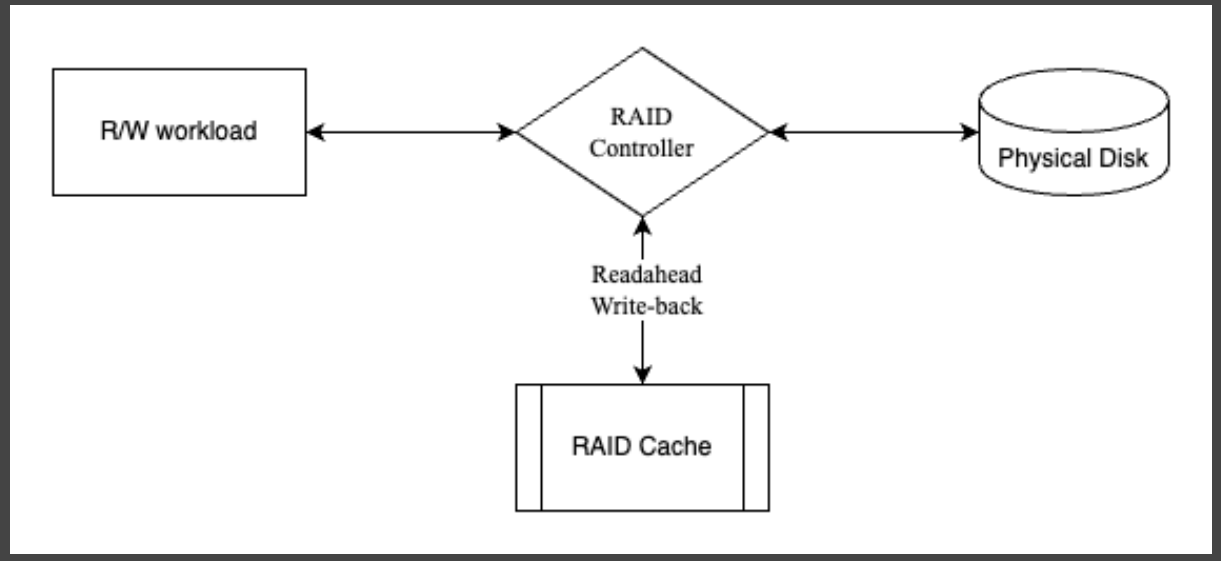
- Dramatically good effect on HDD, but not on SSD

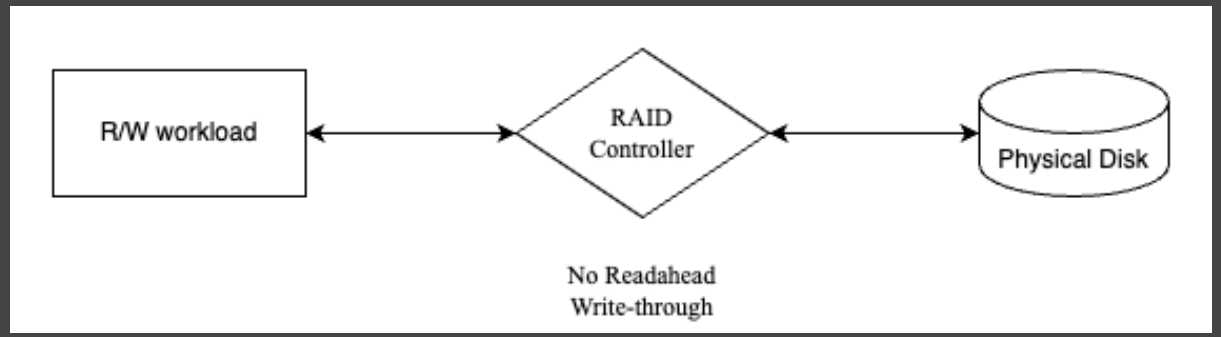
Readahead is not required to SSD

- Locality of data
- Most effective on HDD too

Improve Performance (3)

RAID 0 – Maximizing performance of RAID 0 SSD array





Write-back cache is not always effective

- Dramatically good effect on HDD, but not on SSD

Readahead is not required to SSD

- Locality of data
- Most effective on HDD too

Enabling SSD Array Optimization

- Supported by RAID controller

Results

Random Read

128,000 IOPS → 307,000 IOPS

🚀 240% Improved

Random Write

39,600 IOPS → 204,000 IOPS

🚀 515% Improved

Improved Disk I/O Performance

*Higher is better
fio 3.7 - 4K randread and randwrite

Results

Indexing

Total Time

5h 32m → 2h 41min

🏎️ 51% Reduced

Searching

Latency

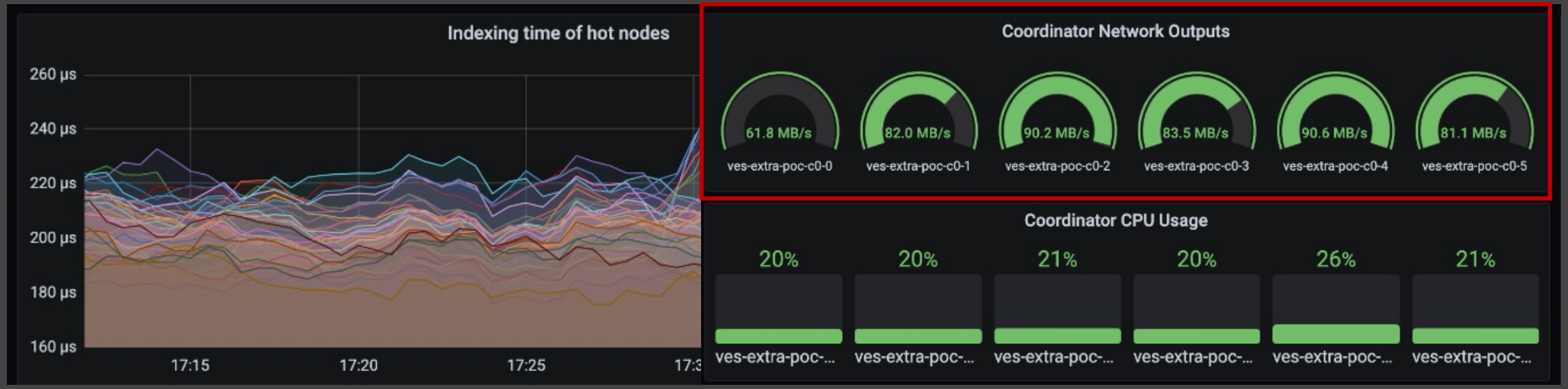
45.65ms → 19.17ms

🏎️ 58% Reduced

Improved esrally performance

*Lower is better; Compared with newly designed node type esrally benchmark set - http_logs and pmc

Results



Metric of proof-of-concept cluster after performance tuning

FUTURE PLANS

OpenSearch

License problems for higher version

Support both Elasticsearch and OpenSearch

Migrate Elasticsearch to OpenSearch without downtime

Performance

Bring the largest legacy cluster in LINE to Private Cloud

Support the multiple container for extreme workloads

Support NVMe-oF to search-intensive clusters

Auto-healing

Replace abnormal nodes automatically

Need different strategy by node role

Node Pool Architecture

- Tolerance for outages of IaaS control plane
- Support baremetal to provide native performance

High Availability

Support multiple availability zone for IaaS resources

Support fault tolerance against datacenter failure

- Cross-Cluster Replication
- Place role-set to every availability zone (DC)

🚀 We are hiring 😄

No need for Elasticsearch / OpenSearch Experience

Thank you
Q&A

LINE