Taming spiky log volumes:
Maintaining real-time logs using KalDB

Suman Karumuri
SRECon APAC - Singapore
June 2023









Who am I?

Principal Observability engineer @ Airbnb.

LogSearch: ELK, Loglens, KalDB

Tracing: Zipkin, PinTrace, SlackTrace, OpenTracing author.

Large scale distributed systems.

Quiz

Centralized log Search

Logs are widely used to monitor systems.

Centralized log search aggregates data in one central location

- Guaranteed retention.
- Triage issues across services and machines.
- Consistent experience.

Log ingestion pipeline

Motivation: Spiky logs.

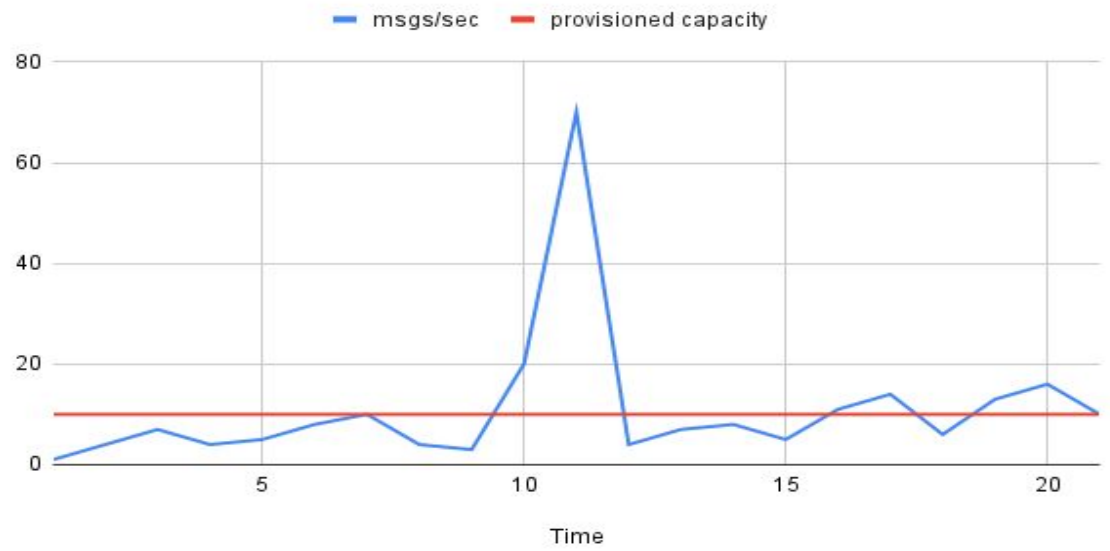
Dealing with Spiky logs.

Intro to Kaldb

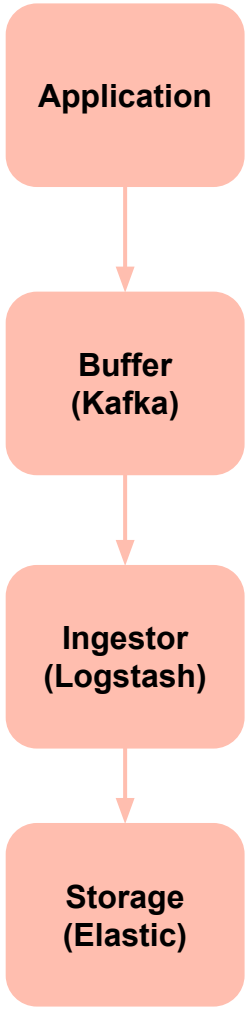
Real time logs with Kaldb

Conclusion

Log spike



Storage sees 10x log volume than usual/provisioned capacity.



Log spike

Storage sees 10x log volume than usual.

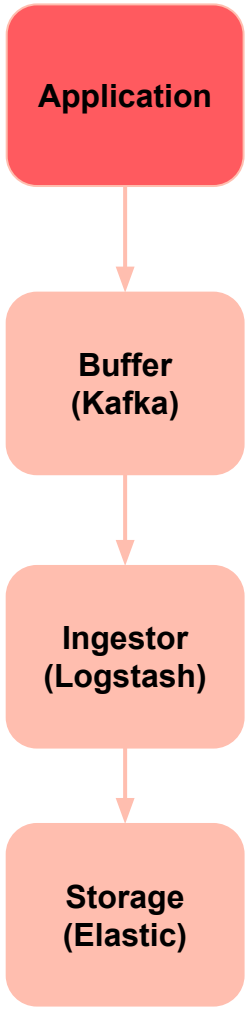
Causes logs to lag: mins to hours.

We lose real-time visibility into our systems.

- Uptime SLA
- Real SLA for freshness - 50-70%
- Perceived SLA: 0%

Increased operational overhead.

Increased infra $$$ for peak provisioning.

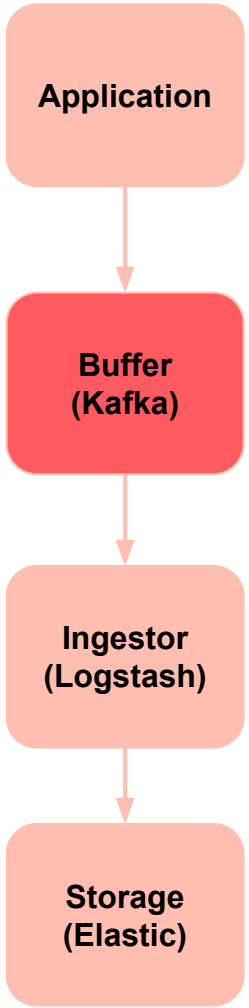


Log spike causes: Misbehaving application

Logging in a tight loop.

Large scale failures of downstream systems like db failures.

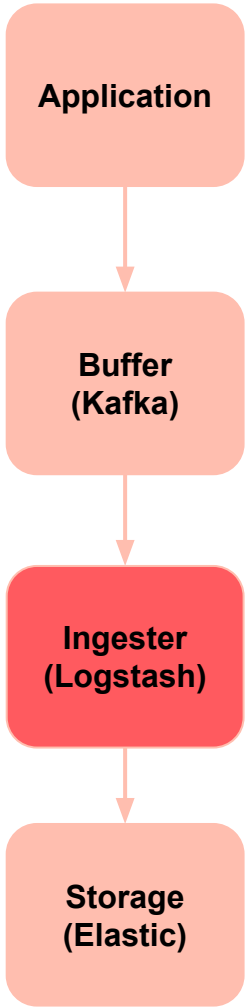
Unexpected request volume to application.



Log spike causes: Buffer issues

Buffer failures cause log accumulation upstream.

Backup of data on the buffer.



Log spike causes: Ingester issues

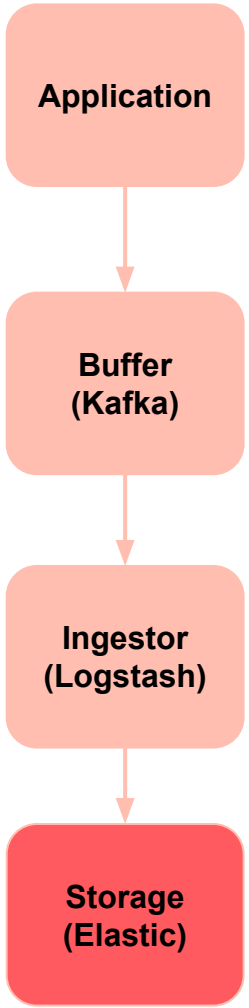
Log ingester is catching up.

Log ingestion is lumpy.

- Large messages.
- Parsing or filtering logs.

Log ingester is mis-configured.

Issues with downstream storage.



Log spike causes: Storage issues.

Storage issues cause log backup.

- When the issue is resolved.
- Causes log spike when storage recovers.

Failure types:

- Node failures
- Reliability/Perf issues.
- Failed writes due to field conflicts.

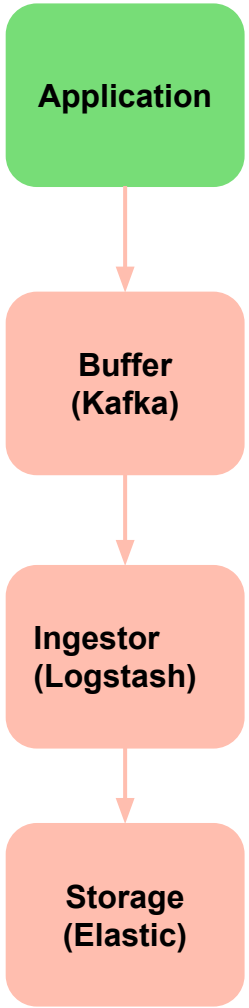
Motivation: Spiky logs.

Dealing with Spiky logs.

Intro to Kaldb

Real time logs with Kaldb

Conclusion



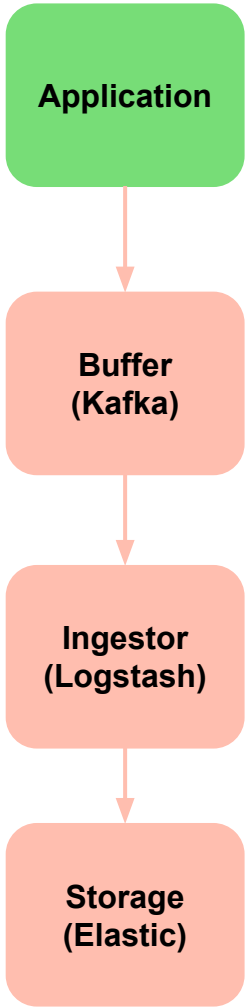
Dealing with Log spikes: Application level

Logging a tight loop or large messages.

Code review.

Code audit critical paths.

```
for(i=0; i< large_value;i++) {
        large_field = ...;
        .....
        log.info("... log..." + large_field);
}
```



Dealing with Log spikes: Log library level

Unexpected log volume from application

Apply rate limits in log reporter.

Large log messages

Size limits on log messages: <O(10k)/msg

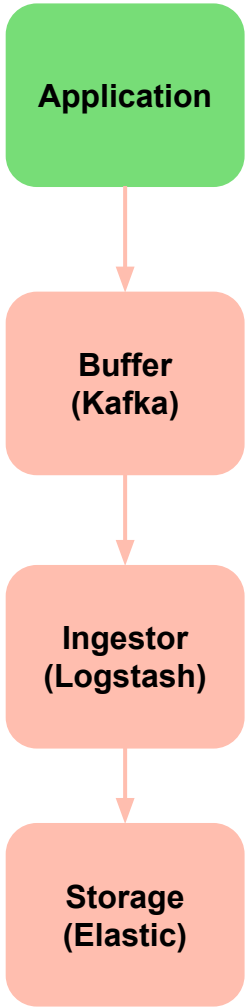
Limits on field truncation size: <1-2k/field

Buffering logs in application?

Smooth log reporting:

small batches & limits.

<1MB per batch



Dealing with Log spikes: Application level

Log sampling

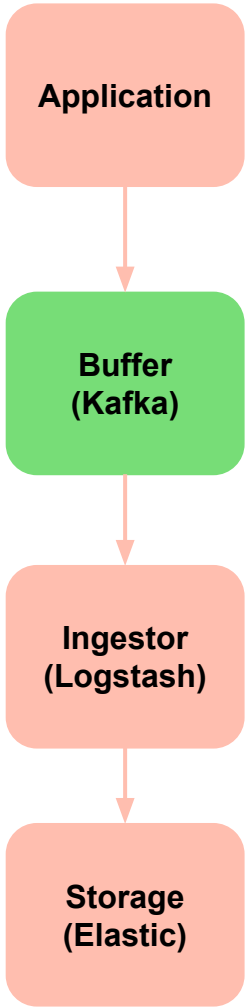
Every log location should have a sampling rate.

```
log.info(0.05, "log ....")
```

Log message prioritization

Only log interesting logs.

One man's trash is another's treasure.



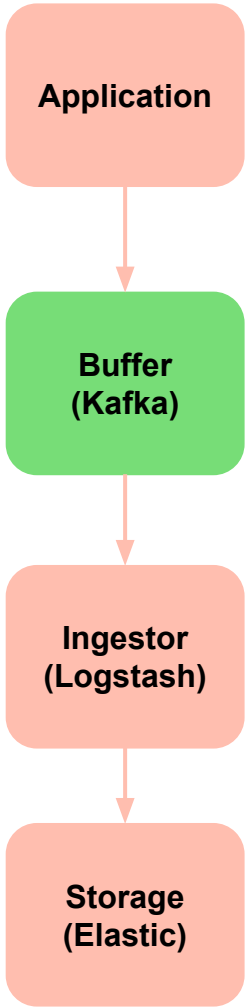
Dealing with Log spikes: Buffer level

Rate limits

Apply rate limits per stream.

Apply message size limits per stream.

Limit messages ingested per second.



Dealing with Log spikes: Buffer level

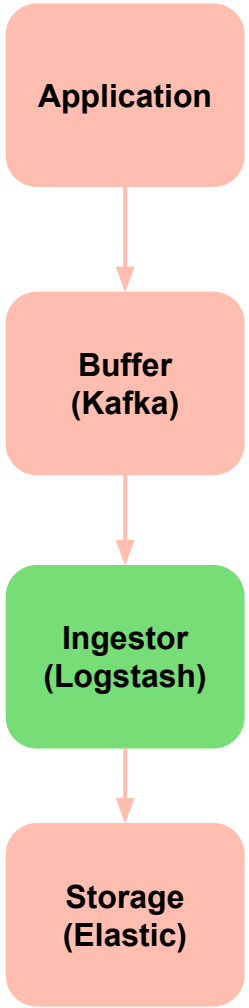
Manage the buffer better

[OpenRunbook]

Open source runbooks for OSS systems.
Real production runbooks.
Don't reinvent runbooks!

[Kafka runbook: OpenRunbook]

Please contribute!



Dealing with Log spikes: Ingester level

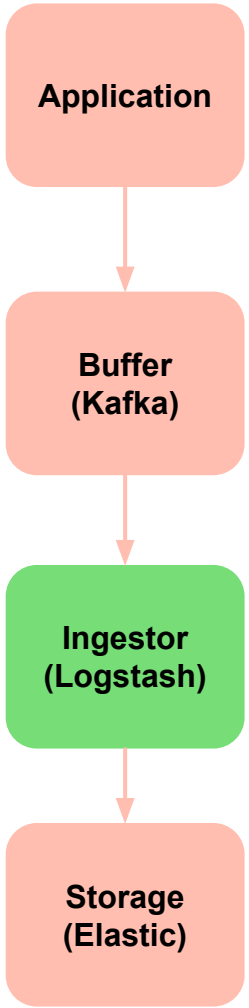
Rate limits

Limit number of messages in/out.

Quotas

Assign quotas per service to isolate noisy neighbors.

Separate streams for services.



Dealing with Log spikes: Ingester level

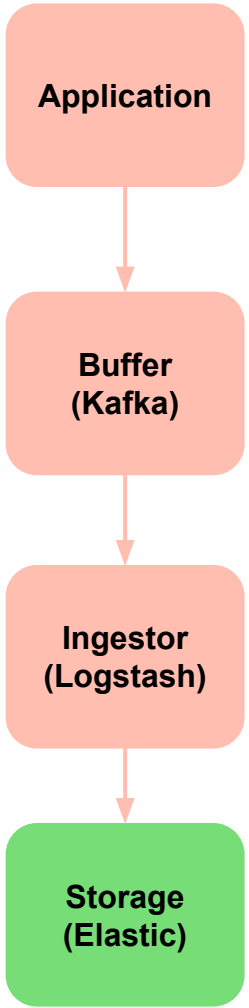
Log sampling

Sample logs in the telemetry pipeline.
Use uniform sampling rate when possible.
Keeps logs useful.

Drop logs

If lag is very high(hours), drop logs.

Risk: Data loss.



Dealing with Log spikes: Storage

Rate limits

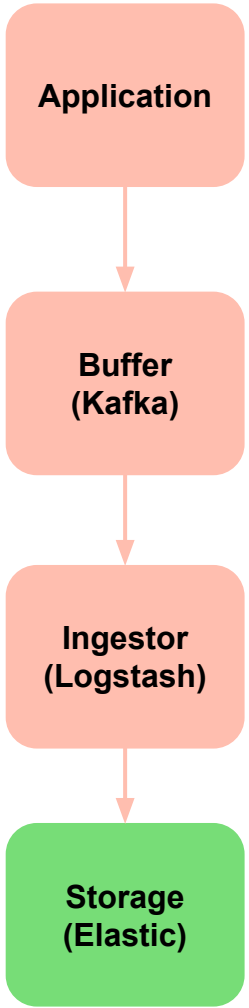
Limit messages written per second.

Fixed limit per node allows better prediction.

Protect storage from excessive reads.

Strict timeouts on reads.

Limit number of parallel read queries.



Dealing with Log spikes: Storage

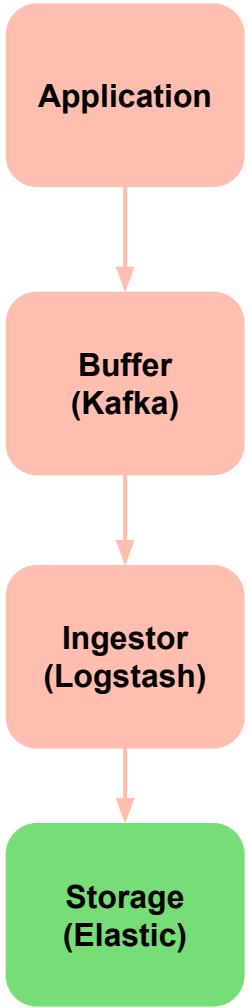
Isolation

Separate clusters for large tenants.

Separate tables for each tenant.

Quotas

Enforce quotas for each tenant.



Root causing of log spikes

Querying storage

Count messages grouped by a field(s).

Pick top 10.

Plot a chart over the last N minutes.

Often the culprit message shows up as an anomaly in the chart.

Can also be applied in the ingester as a stream processor.

Less flexible.

Summary: Log spike

Log spike is a 10x increase in volume of logs.

Log spikes lead to lag => loss of real time visibility into our systems.

Application issues or failures in log ingestion pipelines cause log spikes.

Better management, rate limiting, sampling, quotas etc minimize impact of log spikes.

Prevention still results in data loss/lag + toil.

Yet,
when
problems
happen
humans are
paged.

What if storage can adapt to handle a log spike?

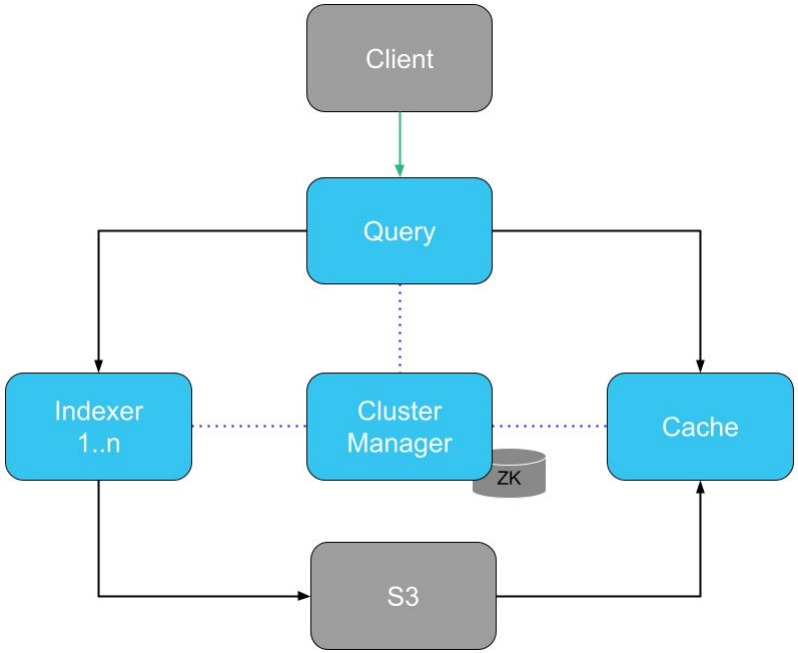
Motivation: Spiky logs.

Dealing with Spiky logs.

Intro to Kaldb

Real time logs with Kaldb

Conclusion



KalDB

KalDB is the ONLY lucene based cloud native observability database.

- Low operational overhead and k8s native.
- Open Source

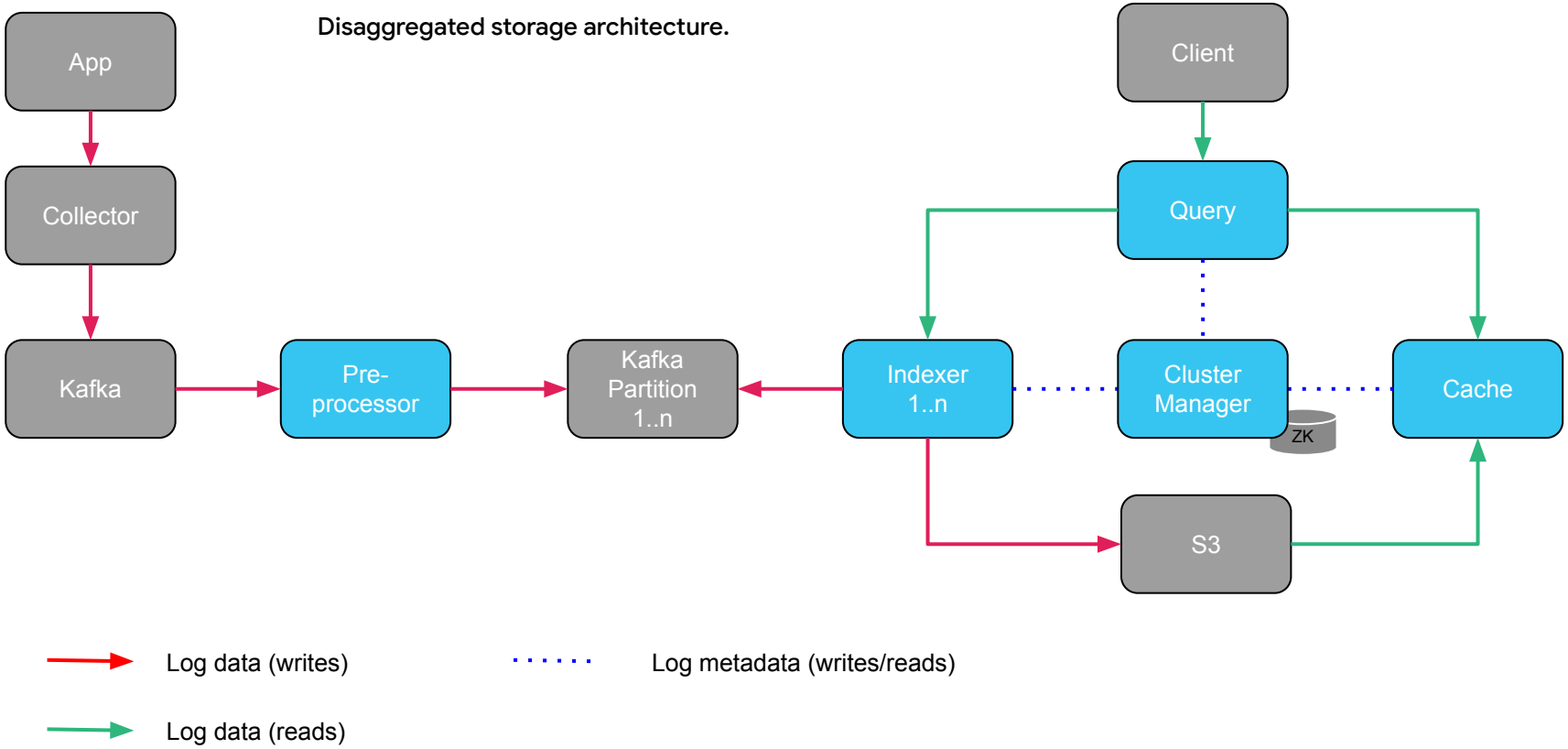
Drop in replacement for OpenSearch.

Designed for PB scale workloads.

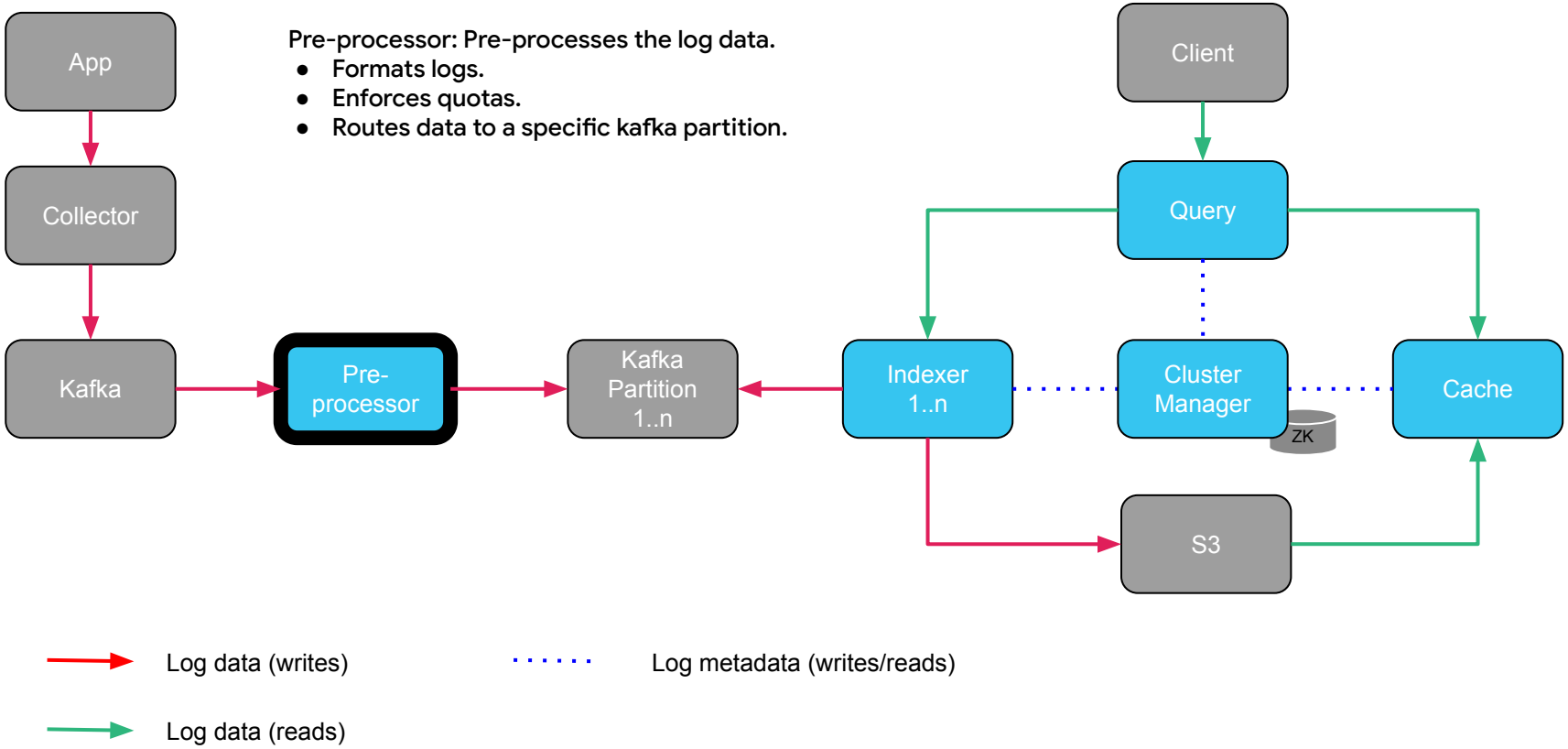
Handles field conflicts automatically.

Faster and up to 10x cheaper than OpenSearch.

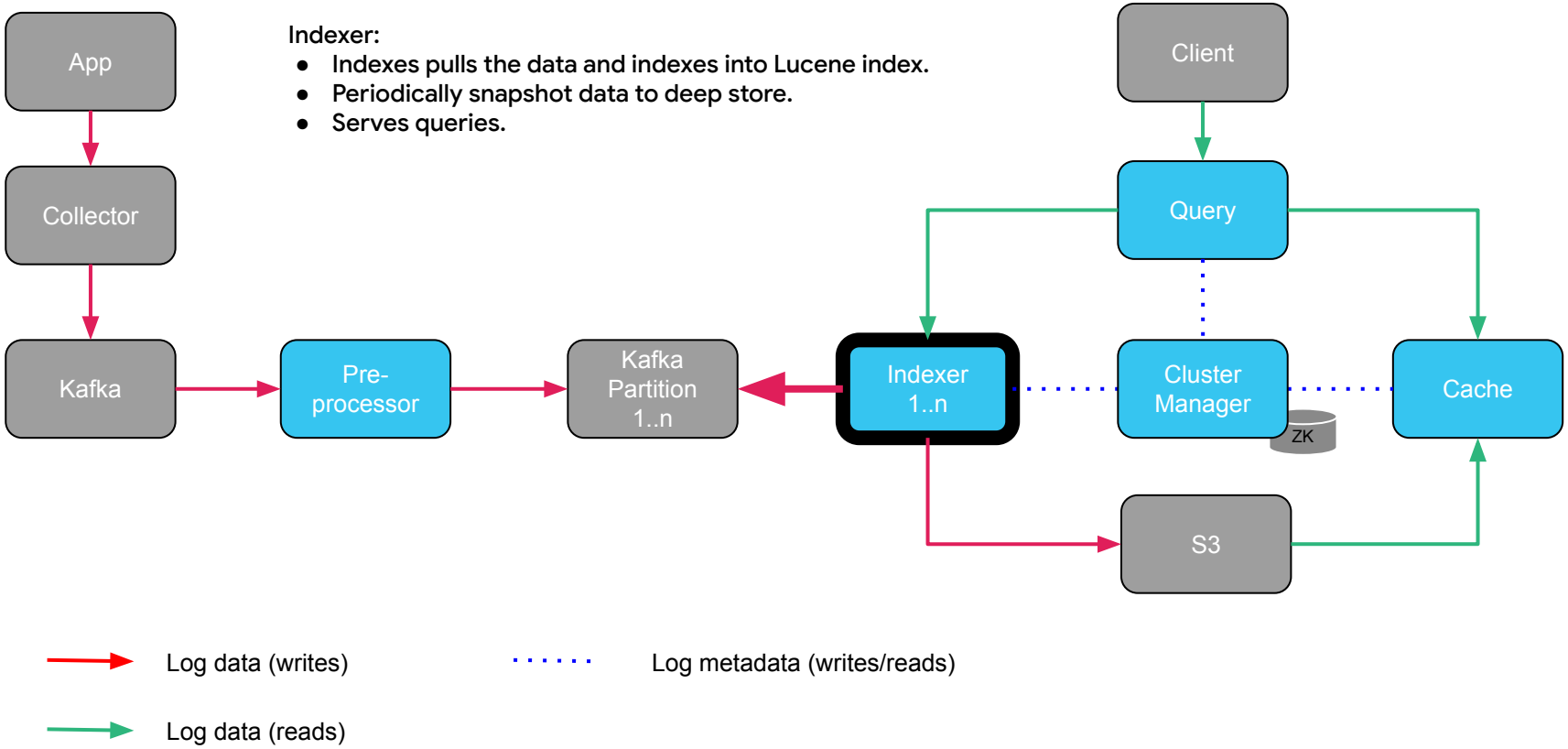
KalDB Architecture: Cloud native



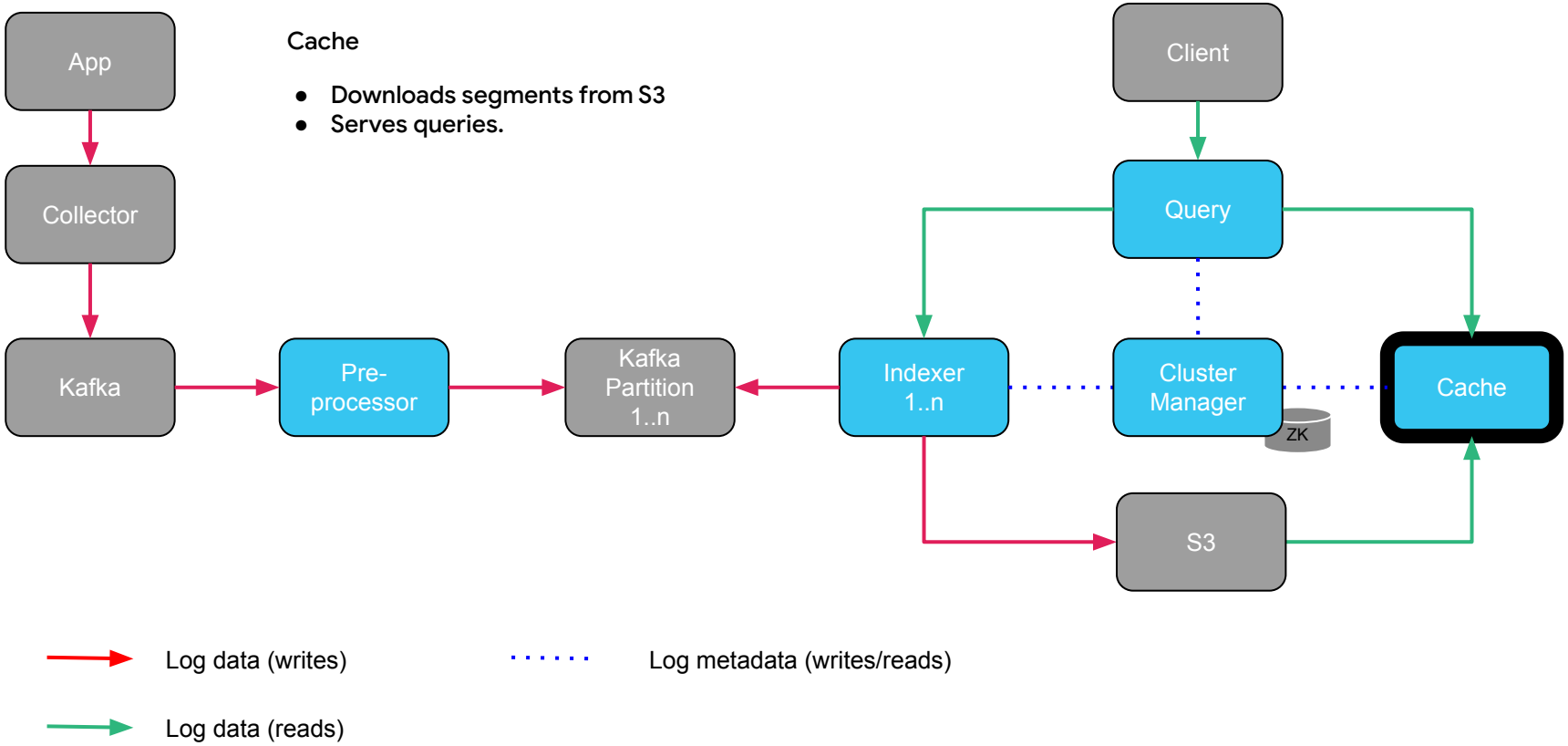
KalDB Architecture: Cloud native



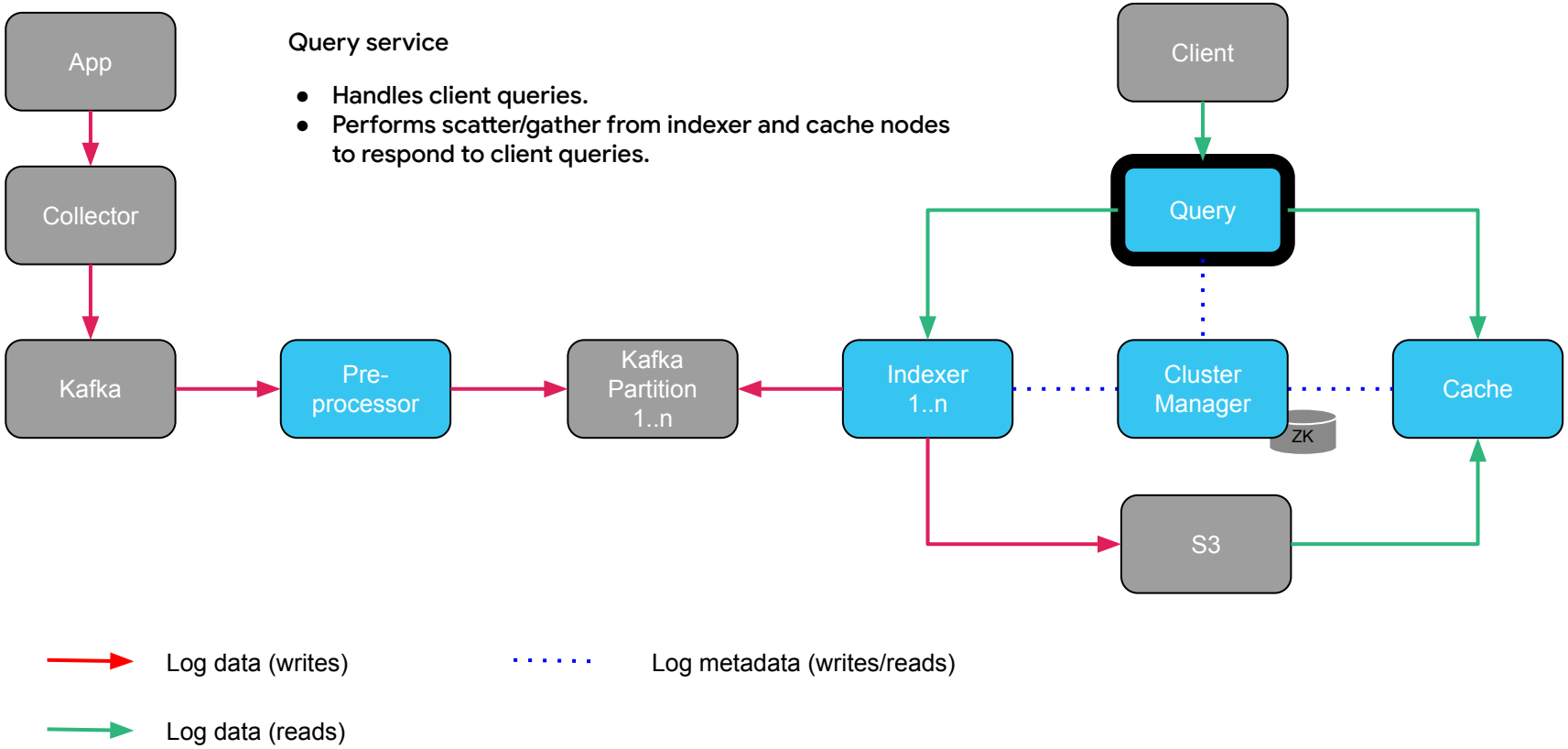
KalDB Architecture: Cloud native



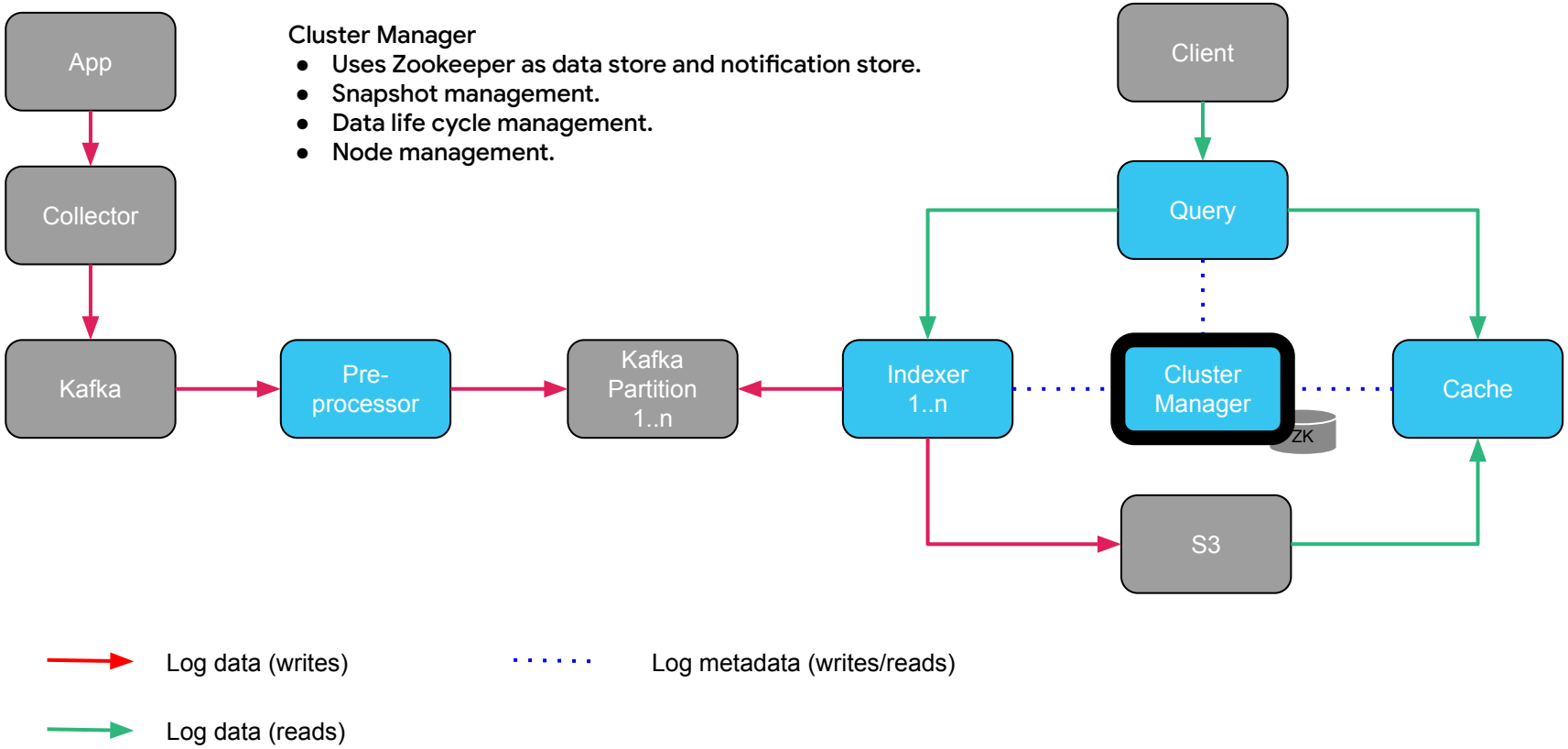
KalDB Architecture: Cloud native



KalDB Architecture: Cloud native

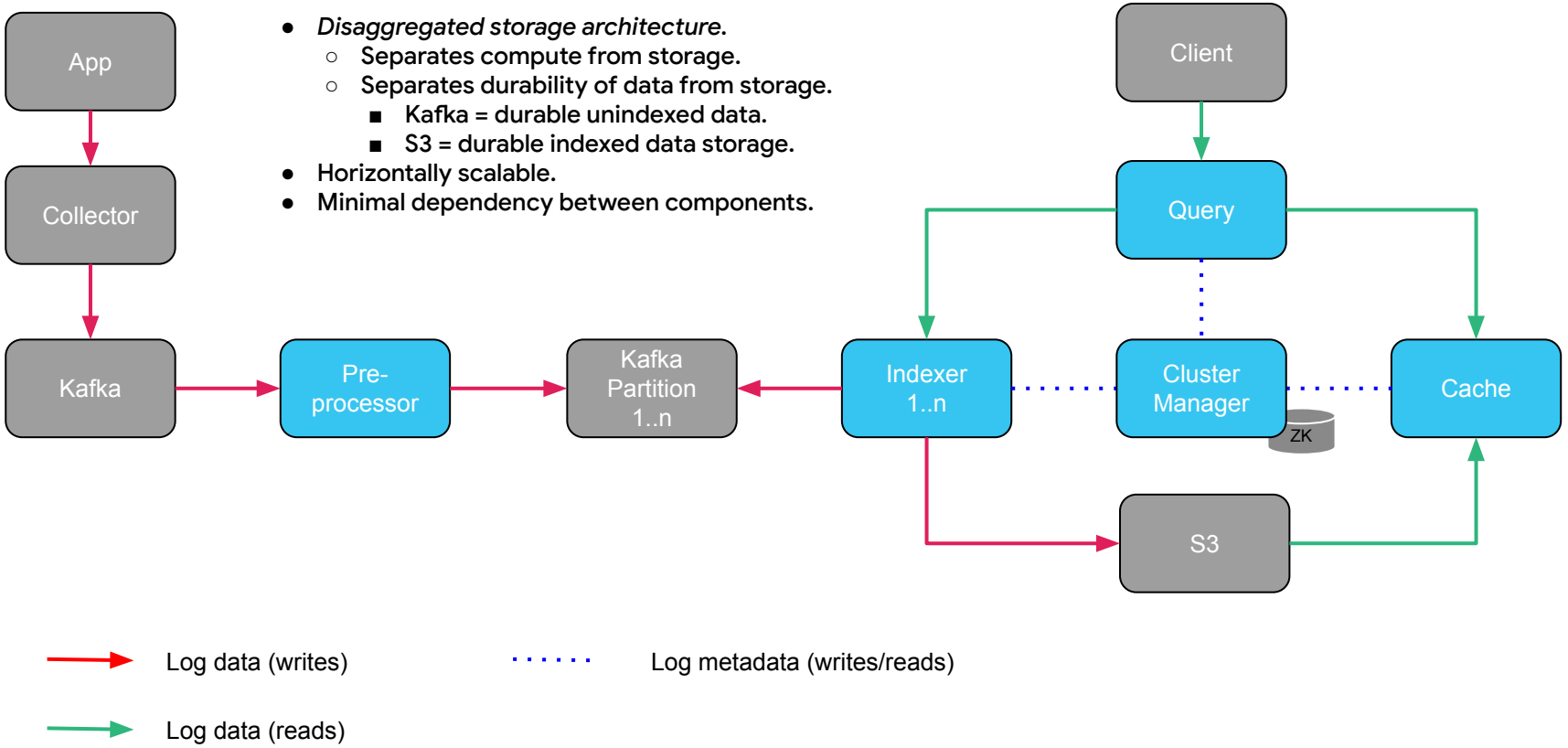


KalDB Architecture: Cloud native



KalDB Architecture: Cloud native

- *Disaggregated storage architecture.*
  - Separates compute from storage.
  - Separates durability of data from storage.
    - Kafka = durable unindexed data.
    - S3 = durable indexed data storage.
- Horizontally scalable.
- Minimal dependency between components.



Motivation: Spiky logs.

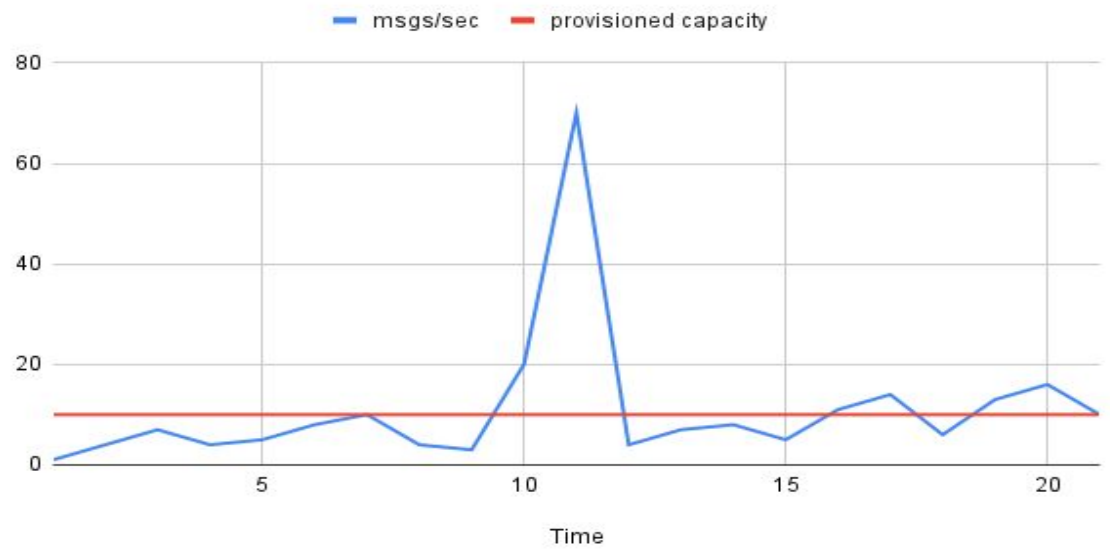
Dealing with Spiky logs.

Intro to Kaldb

Real time logs with Kaldb

Conclusion

Log spike



Storage sees 10x log volume than usual/provisioned capacity.

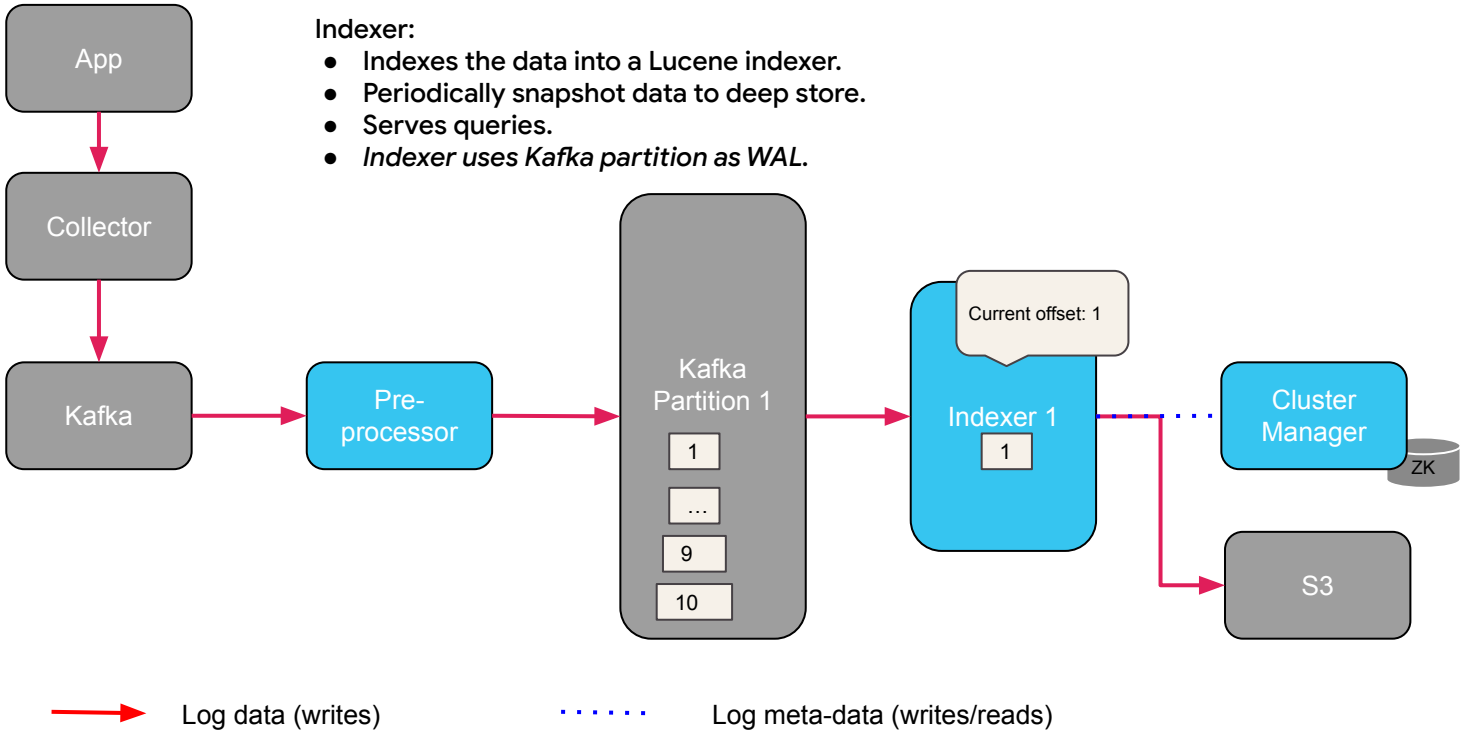
To ingest log spike while being real-time:

Prioritize ingesting fresh logs over older logs.

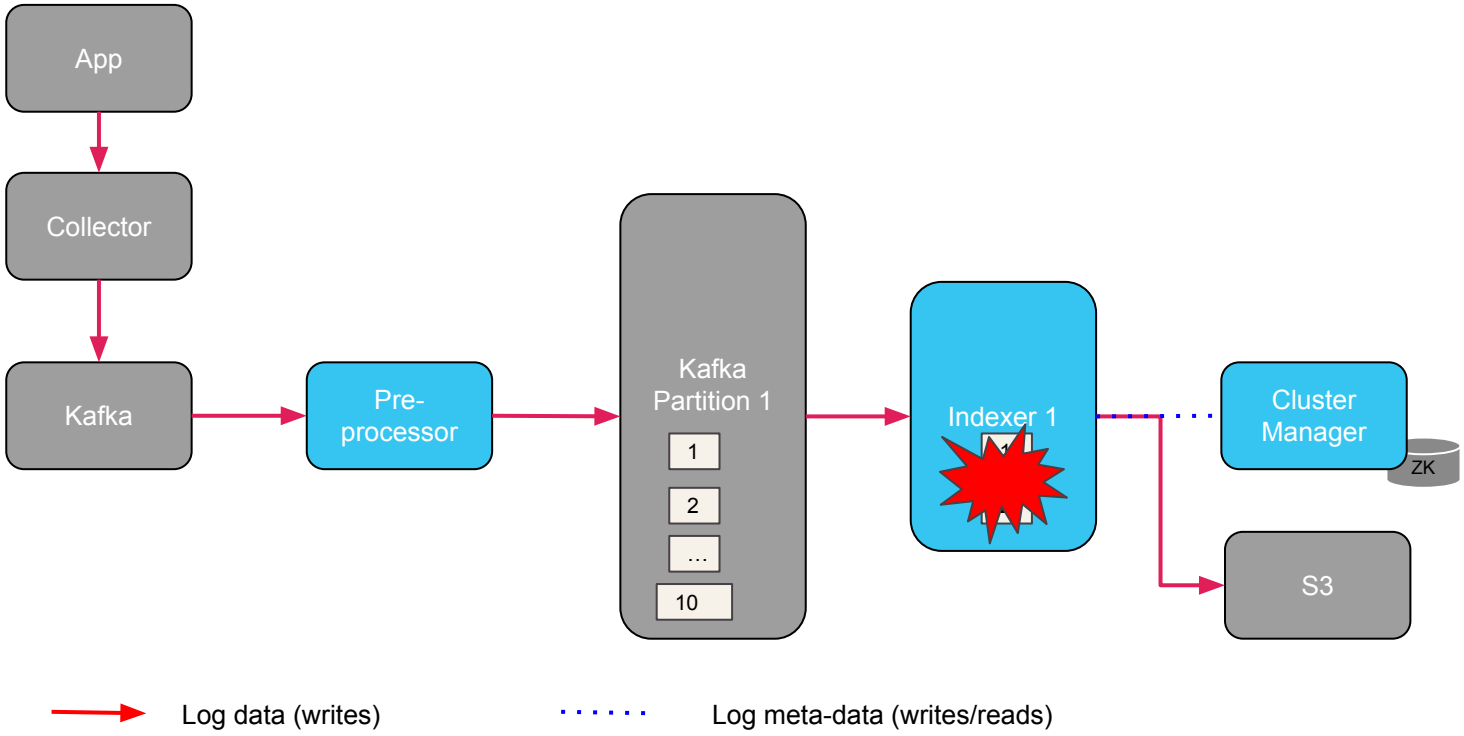
Quiz

What does ES do when you add more nodes to it during a log spike?

KalDB: Prioritize fresh logs over older logs.

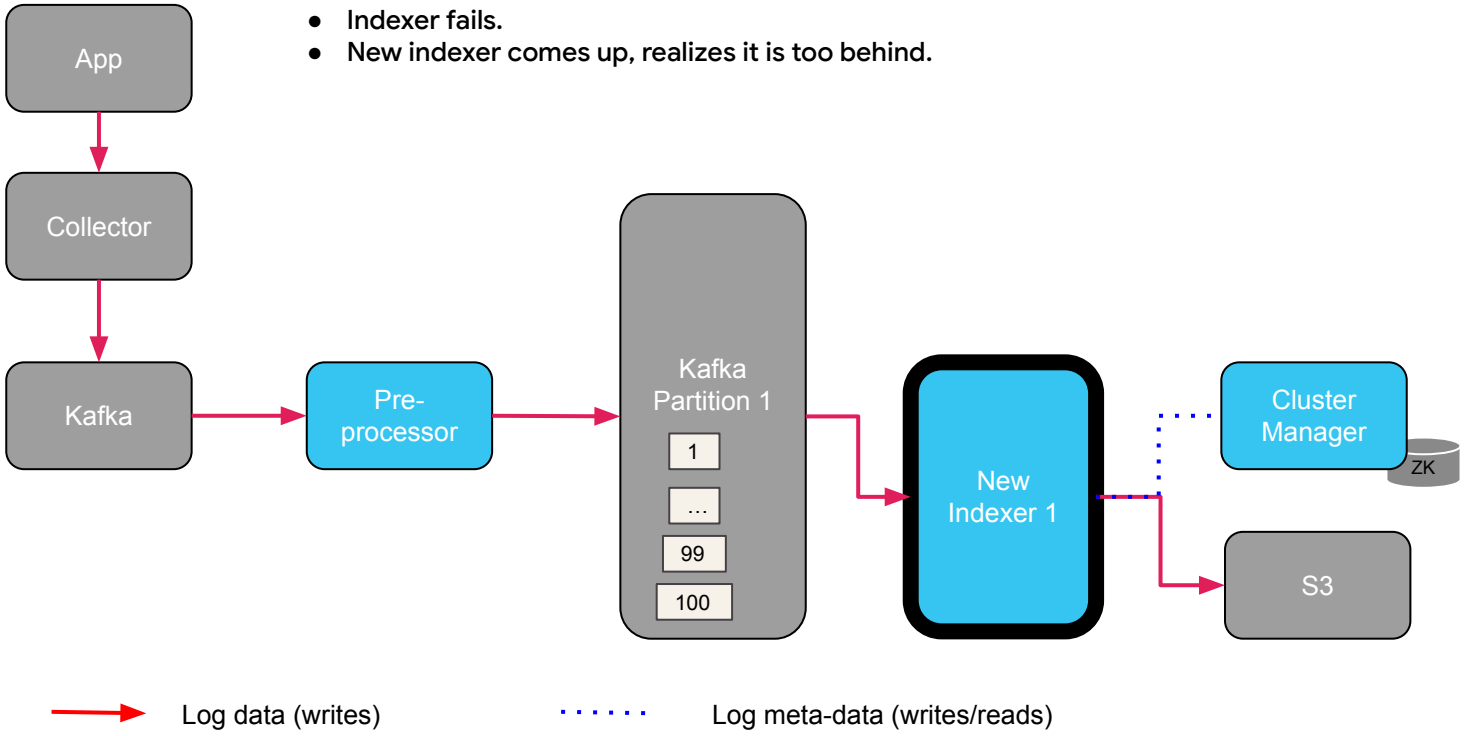


KalDB: Prioritize fresh logs over older logs.



KalDB: Prioritize fresh logs over older logs.

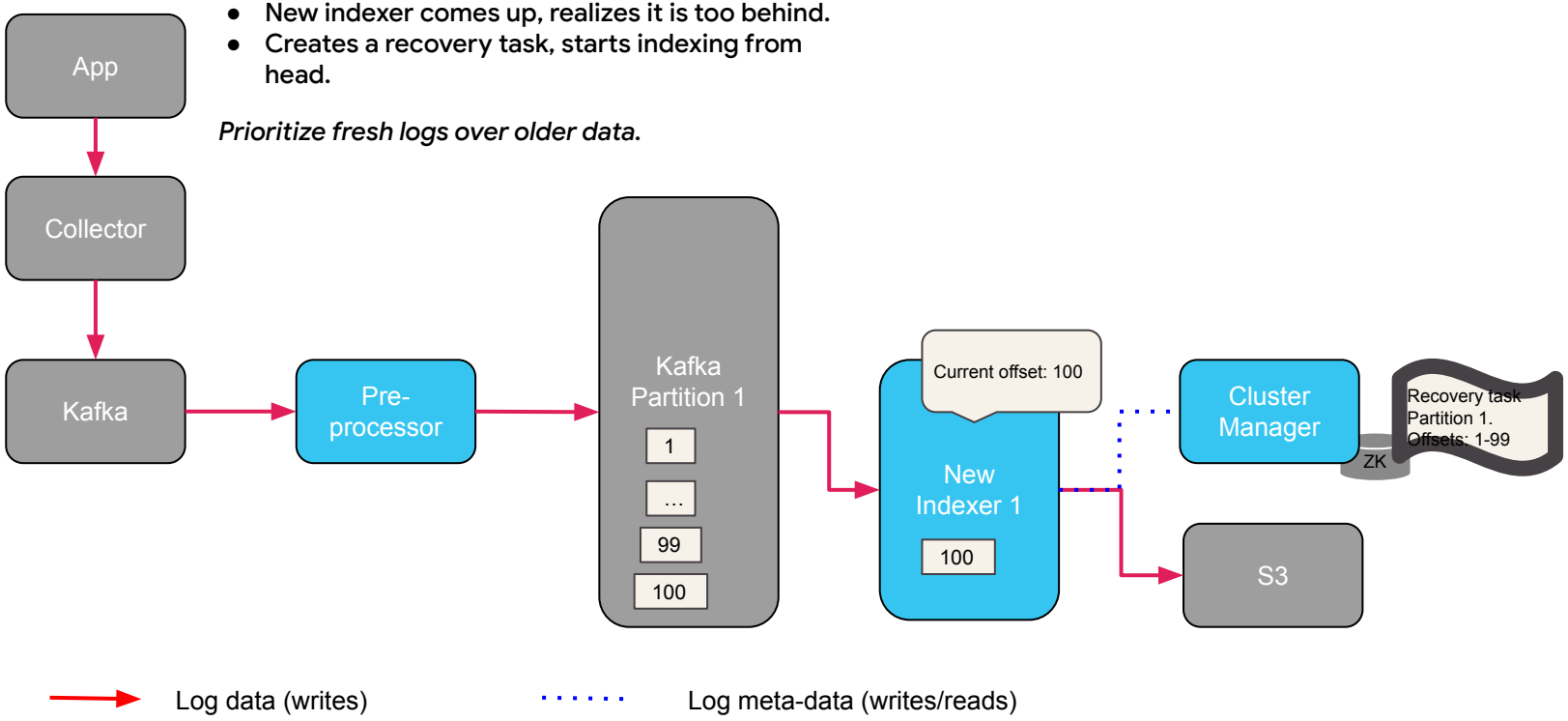
- Indexer fails.
- New indexer comes up, realizes it is too behind.



KalDB: Prioritize fresh logs over older logs.

- Indexer fails.
- New indexer comes up, realizes it is too behind.
- Creates a recovery task, starts indexing from head.

Prioritize fresh logs over older data.



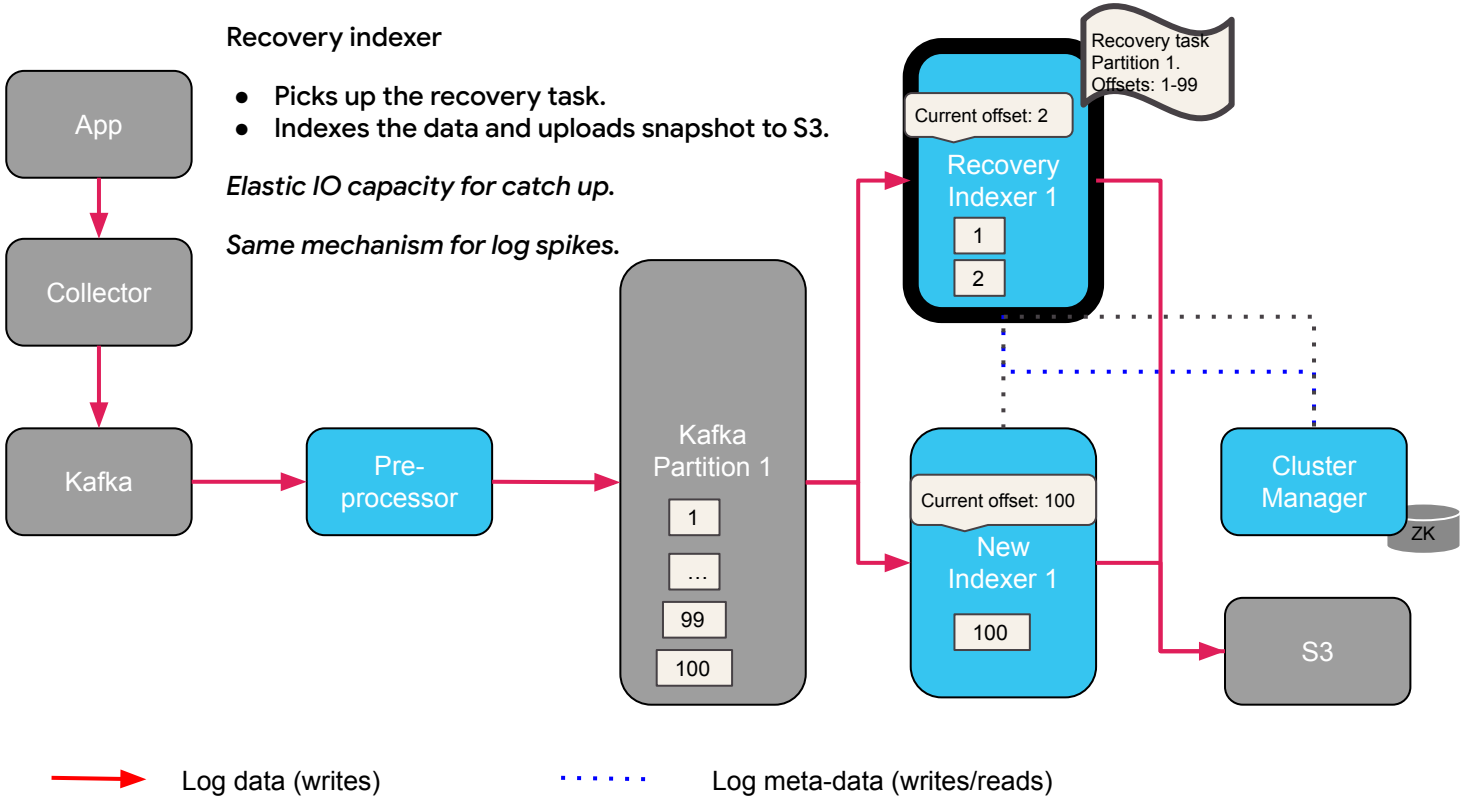
KalDB: Prioritize fresh logs over older logs.

Recovery indexer

- Picks up the recovery task.
- Indexes the data and uploads snapshot to S3.

Elastic IO capacity for catch up.

Same mechanism for log spikes.



Dealing with noisy neighbours

Isolation

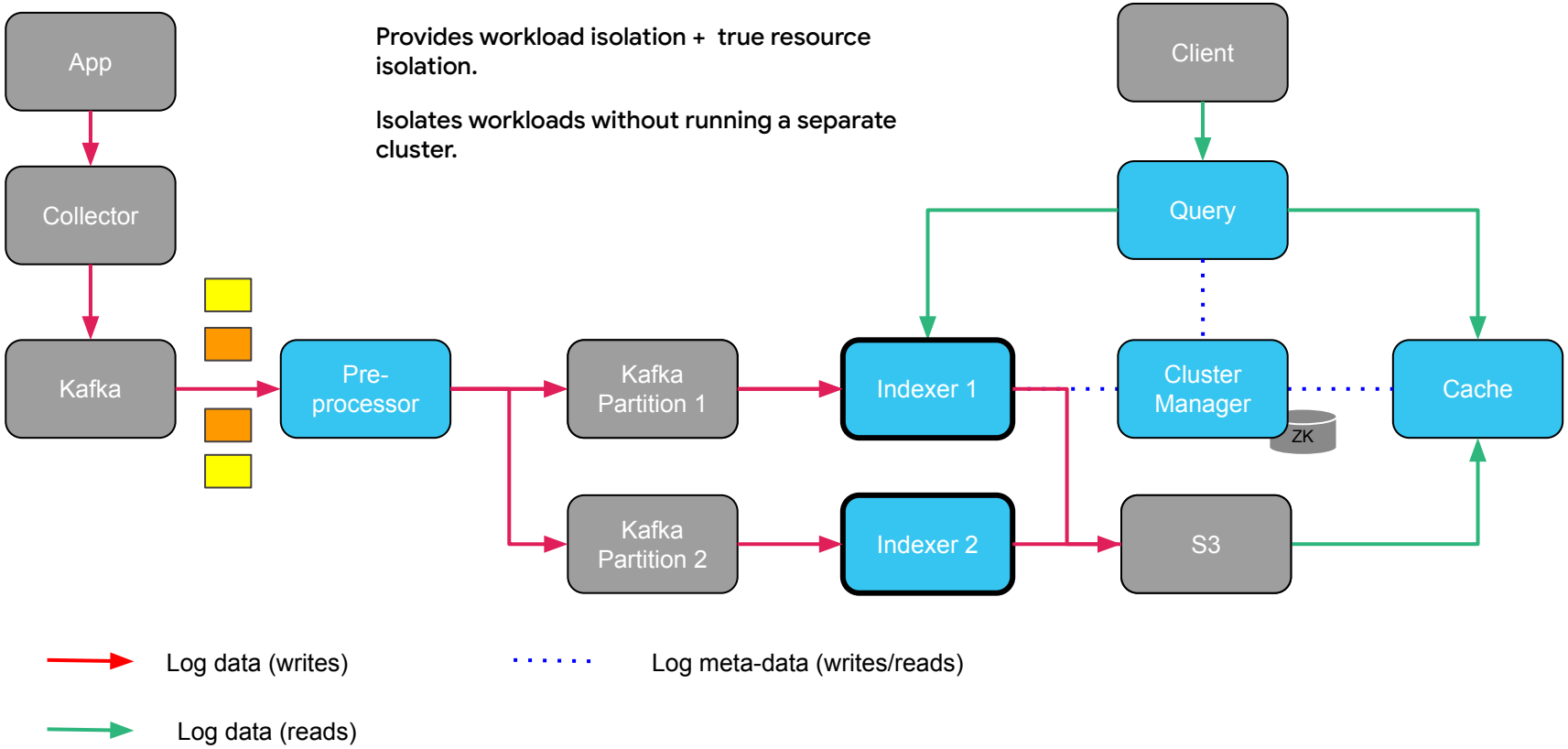
Separate clusters for large tenants.
Separate tables for each tenant.

Quotas

Enforce quotas for each tenant.

Managing multiple(100+) clusters is tedious and error prone.

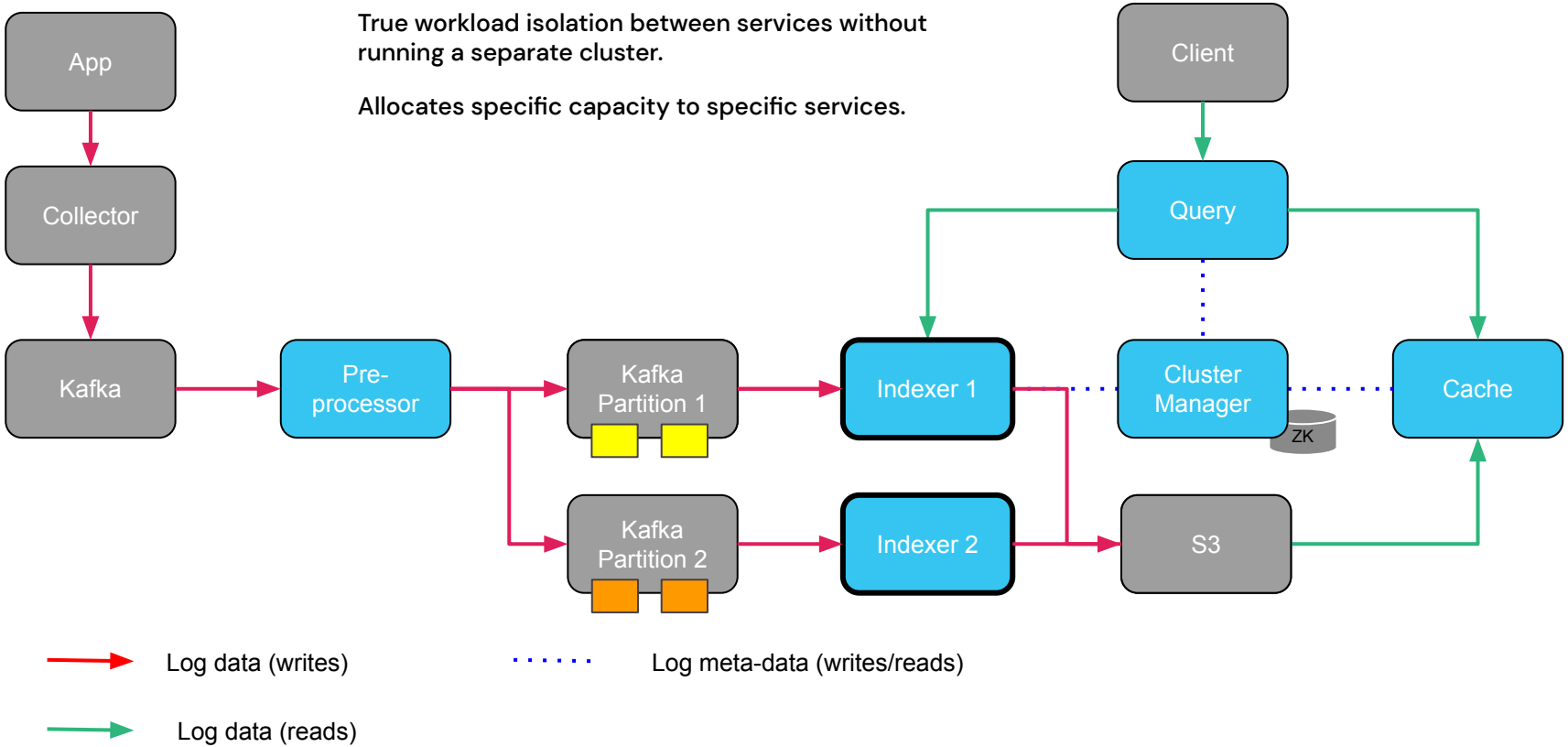
KalDB: Multi-tenancy



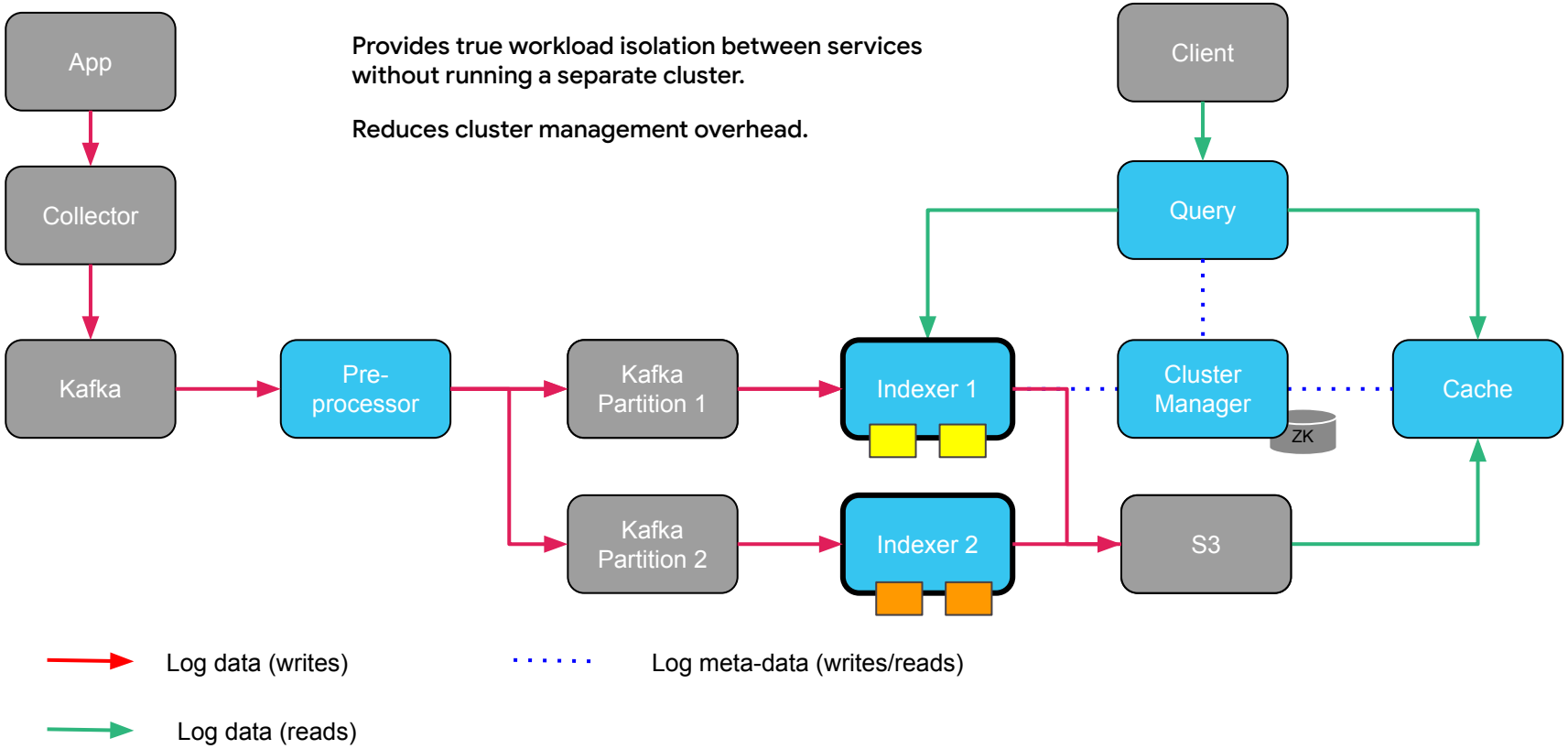
KalDB: Multi-tenancy

True workload isolation between services without running a separate cluster.

Allocates specific capacity to specific services.



KalDB: Multi-tenancy



Motivation: Spiky logs.

Dealing with Spiky logs.

Intro to Kaldb

Real time logs with Kaldb

Conclusion

Summary: Log spike

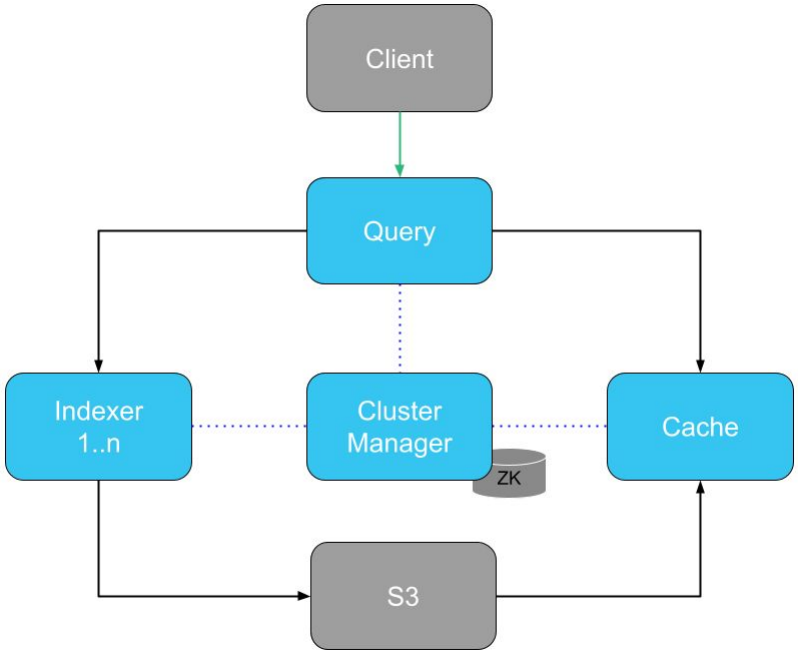
Log spike is a 10x increase in volume of logs.

Log spikes lead to lag => loss of real time visibility into our systems.

Application issues or failures in log ingestion pipelines cause log spikes.

Better management, rate limiting, sampling, quotas etc minimize impact of log spikes.

Prevention still results in data loss/lag + toil.



Summary: KalDB

KalDB is an open source petabyte scale lucene based log search engine.

KalDB has built in back-fill that prioritizes ingesting fresh logs over older logs.

KalDB features like multi-tenancy, automatic field conflict resolution simplify log pipeline maintenance.

Thank you!

Suman Karumuri @ LinkedIn.

mansu @ twitter

Kaldb @ https://github.com/slackhq/kaldb

OpenRunbook

All product names, logos, and brands are property of their respective owners. All company, product and service names used in this presentation are for identification purposes only. Use of these names, logos, and brands does not imply endorsement.

Q & A

Handling Field Conflicts

Kaldb

Real time logs with Kaldb

All techniques result in Data lag or Data loss.