DataStax

Performance tuning for Apache Pulsar Kubernetes deployments in the cloud

Lari Hotari

October 5, 2022

Lari Hotari

Engineering Coach, Streaming
Customer Reliability Engineering

DataStax

Lari.Hotari@datastax.com

@lhotari

Lari Hotari is an Apache Pulsar committer and PMC member. He has worked on the Java platform since 1997 and has contributed to open source for over 20 years.

Agenda

- Performance Tuning
- Apache Pulsar Kubernetes deployments
- Scaling out and scaling up
- Overview of tunable configuration


15'3
4.65m
DEAD SLOW
LAMPTOR
GASCON
LAMBOURNE CARPETS & FLOORING
SLOW


Grandma C's Kitchen
bakery & espresso
The BASE
CRAFT BEER & WINE BAR
PIZZA SALAD · WINGS
CRAFTS
NO PARKING 4PM - 8PM
Snap-on Racing

Going fast?


Short time-to-recover?


Efficiency?

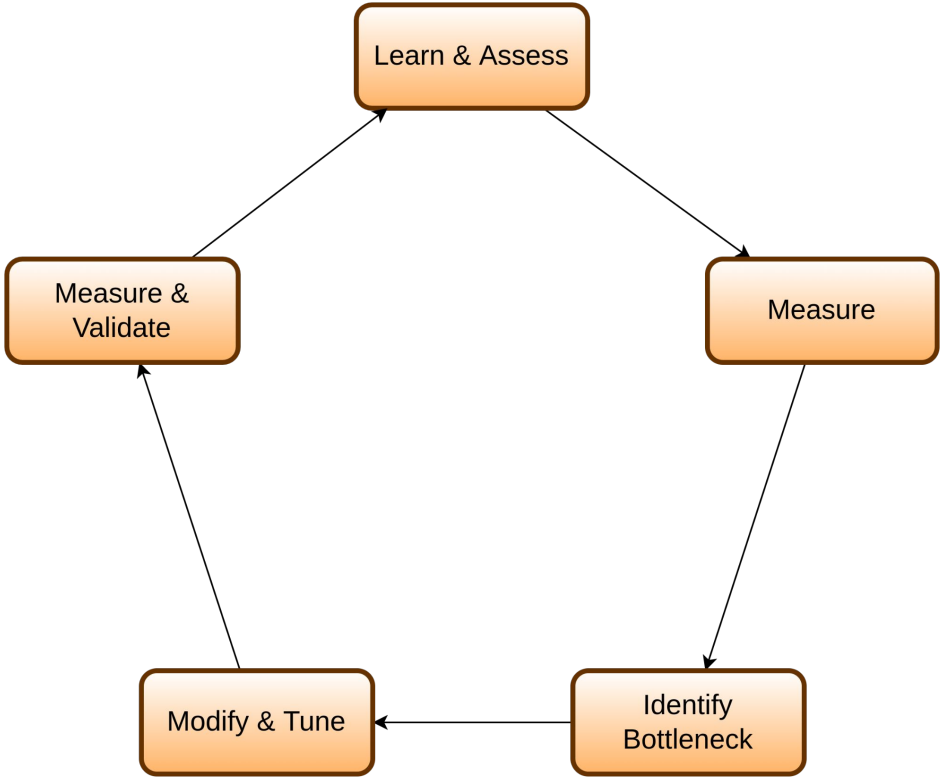
The goal of performance tuning?

- Fulfilling the quality requirements and constraints of your system.
- Improving price/performance to reduce cost

Performance aspects

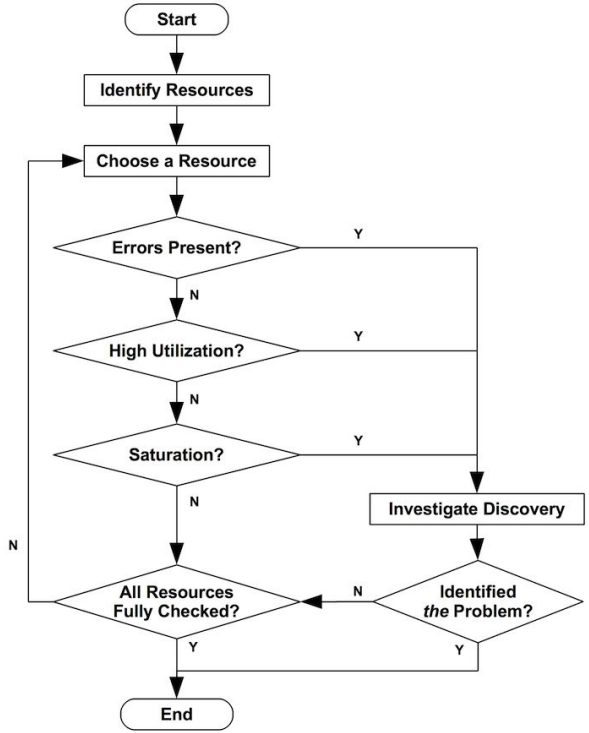
- Latency of operations
- Throughput of operations
- Quality of operations
  - efficiency, usability, responsiveness, correctness, consistency, integrity, reliability, availability, resilience, robustness, recoverability, security, safety, maintainability
    - Impacts value and cost

Performance tuning



USE method for analyzing system performance

- “For every resource, check **U**tilization, **S**aturation, and **E**rrors.”
- It's intended to be used early in a performance investigation, to identify systemic bottlenecks.



https://www.brendangregg.com/usemethod.html

The Four Golden Signals for monitoring in SRE

- Latency
- Traffic
- Errors
- Saturation

https://sre.google/sre-book/monitoring-distributed-systems/

Apache Pulsar Kubernetes deployment options

- Apache Pulsar Helm chart
  - https://github.com/apache/pulsar-helm-chart
- DataStax Apache Pulsar Helm chart
  - https://github.com/datastax/pulsar-helm-chart
- StreamNative Apache Pulsar Helm chart
  - https://github.com/streamnative/charts/tree/master/charts/pulsar

Observability in Pulsar Kubernetes deployments

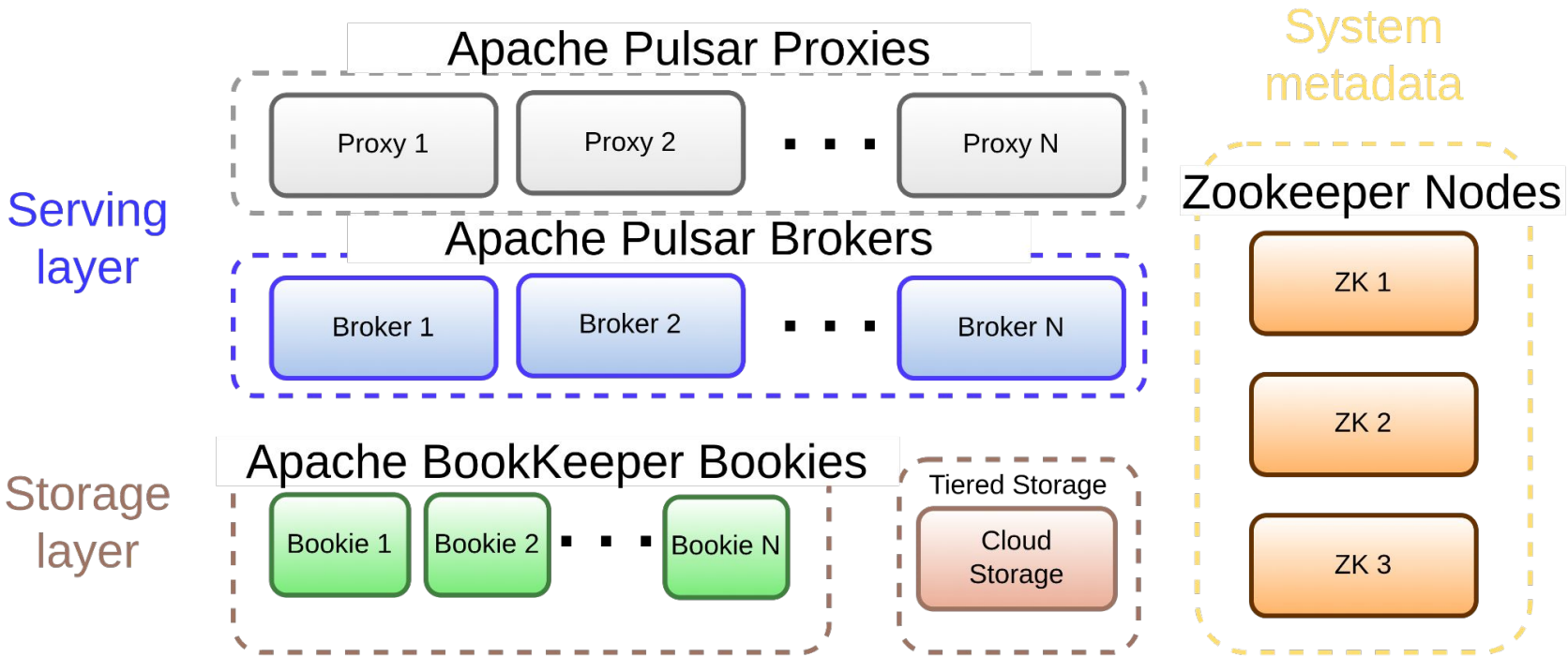
- Prometheus
- Grafana
- Grafana Dashboards
  - Pulsar Broker
  - Bookkeeper
  - Zookeeper

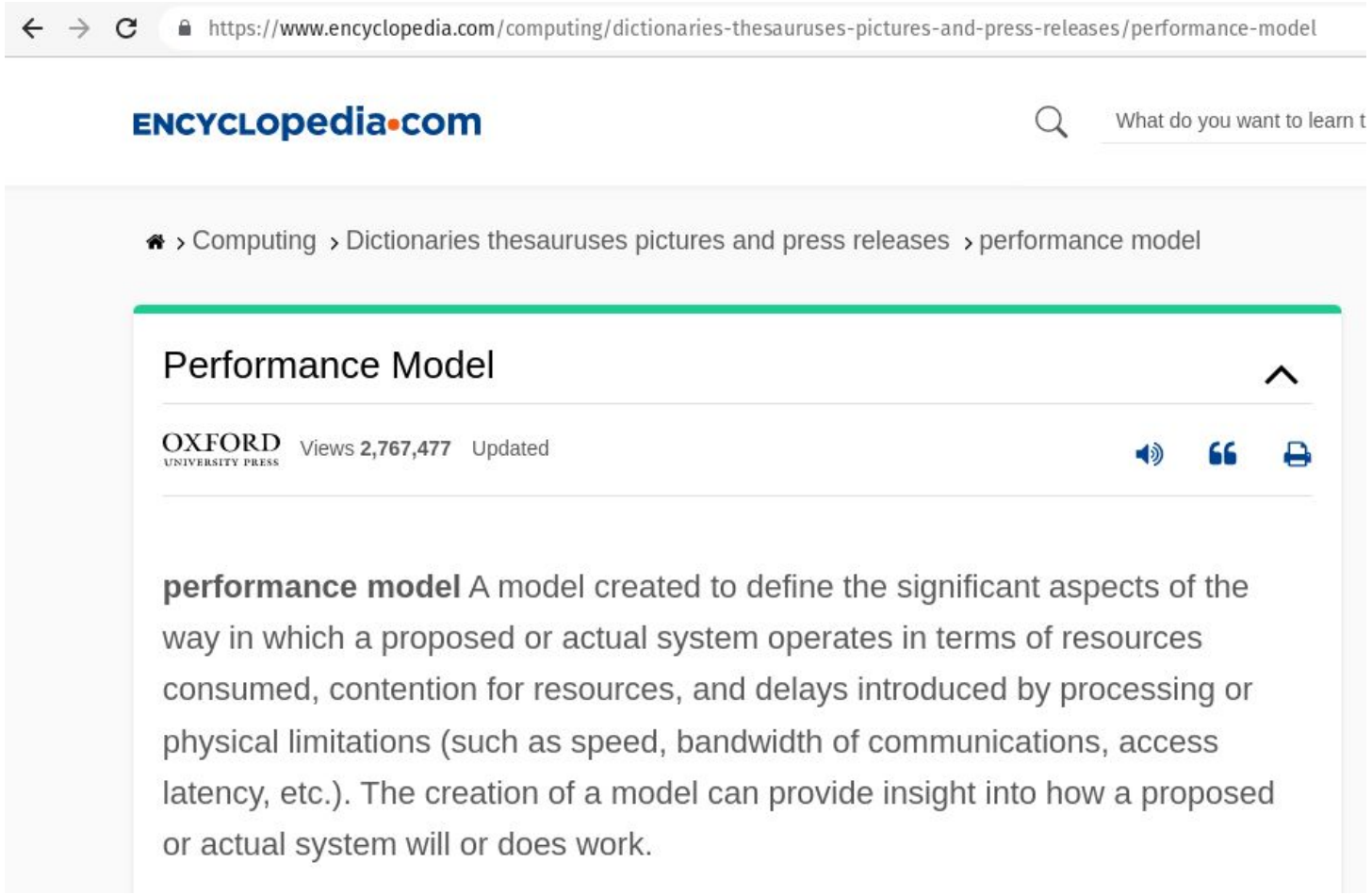

Broker Metrics
Last 15 minutes
1m
Cluster
All
Broker
All
All
Publish rate (msg/s)
200 K
150 K
100 K
50 K
0
01:52
01:54
01:56
01:58
02:00
02:02
02:04
02:06
pulsar-demo-broker-0
pulsar-demo-broker-1
pulsar-demo-broker-2
Dispatch rate (msg / s)
200 kB/s
150 kB/s
100 kB/s
50 kB/s
0 B/s
Publish throughput (MB/s)
20 MB/s
15 MB/s
10 MB/s
5 MB/s
0 B/s
Dispatch throughput (MB/s)
25 MB/s
20 MB/s
15 MB/s
10 MB/s
5 MB/s
0 B/s
Topics count
20
15
10
5
0
Producers count
Consumers count
Messages Backlog
150 K
100 K
50 K
Storage Write Latency
30 K
25 K
20 K
15 K
10 K
5 K
msg / s


Bookie Metrics
Last 15 minutes
1m
pulsar-demo-bookie-0
pulsar-demo-bookie-1
pulsar-demo-bookie-2
Journal Queue length
Journal Force Write Queue length
Journal Callback Queue length
Journal Add latency (p50)
Journal Add latency (p99)
Journal Add latency (Max)
Journal Sync Latency (p50)
Journal Sync Latency (p99)
Journal Sync Latency (Max)
01:50
01:52
01:54
01:56
01:58
02:00
02:02
Storage


ZooKeeper by Prometheus
Last 6 hours
30s
global_sessions
local_sessions
2022-10-04 21:23:05.749
A-series: 27.1
packets_sent
packets_received
connection_drop_count
outstanding_requests
connection_rejected
connection_drop_probability

Pulsar Architecture for Kubernetes deployments



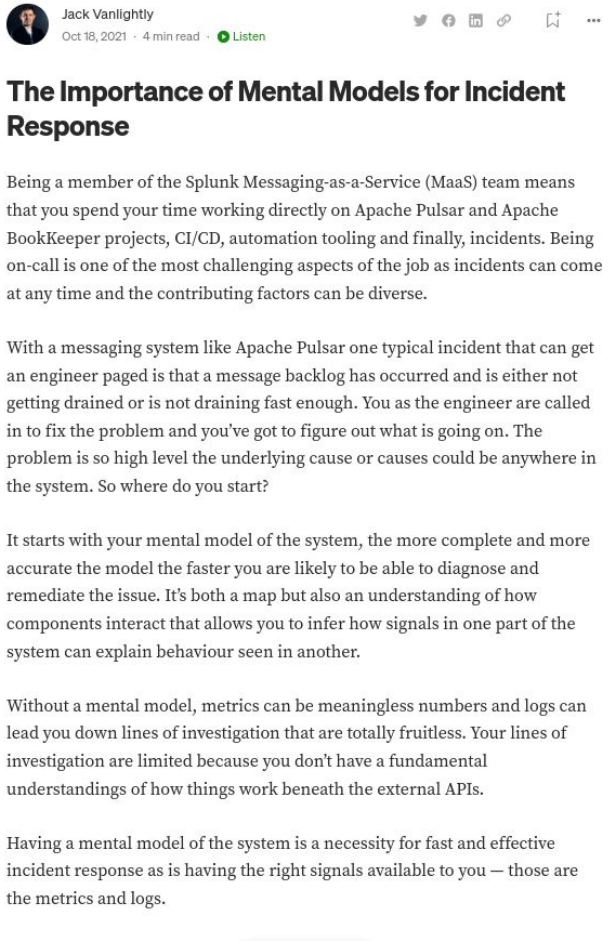


https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/performance-model

“The Importance of Mental Models for Incident Response” by Jack Vanlightly

”It starts with your mental model of the system, the more complete and more accurate the model the faster you are likely to be able to diagnose and remediate the issue. It’s both a map but also an understanding of how components interact that allows you to infer how signals in one part of the system can explain behaviour seen in another.”

https://medium.com/splunk-maas/the-importance-of-mental-models-for-incident-response-da1bfcfdcac



Jack Vanlightly
Oct 18, 2021 · 4 min read · Listen

The Importance of Mental Models for Incident Response

Being a member of the Splunk Messaging-as-a-Service (MaaS) team means that you spend your time working directly on Apache Pulsar and Apache BookKeeper projects, CI/CD, automation tooling and finally, incidents. Being on-call is one of the most challenging aspects of the job as incidents can come at any time and the contributing factors can be diverse.

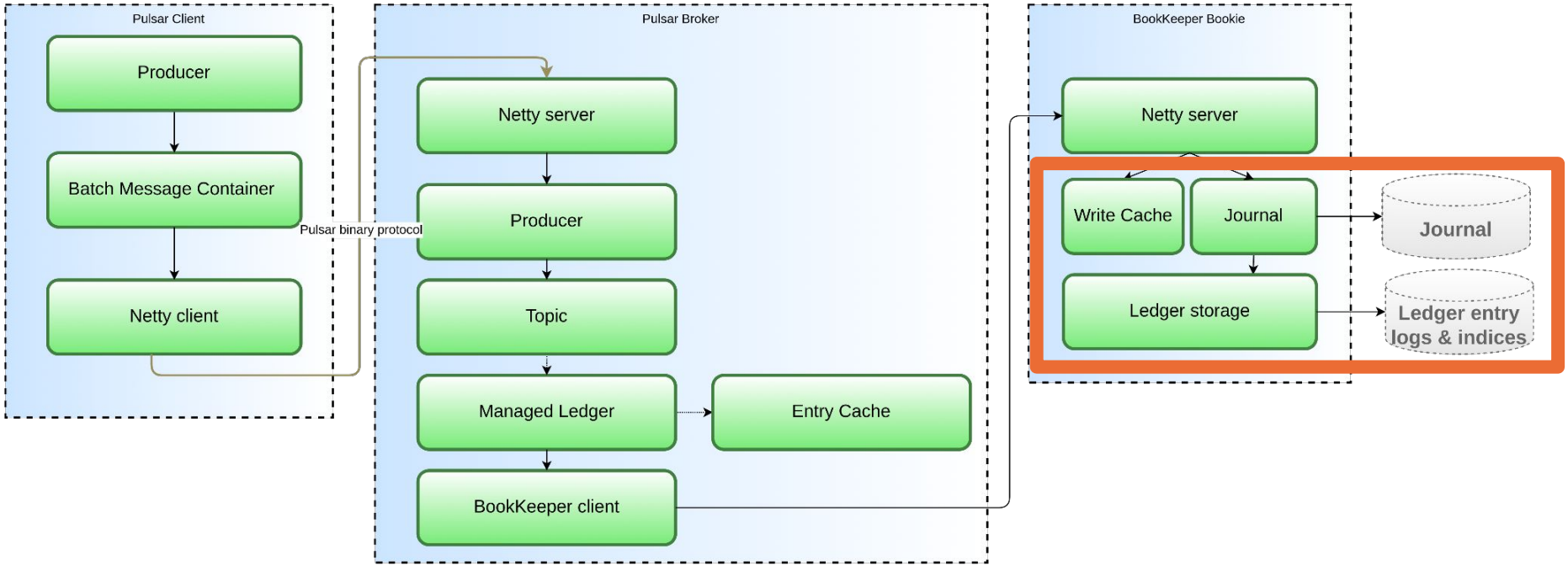
With a messaging system like Apache Pulsar one typical incident that can get an engineer paged is that a message backlog has occurred and is either not getting drained or is not draining fast enough. You as the engineer are called in to fix the problem and you’ve got to figure out what is going on. The problem is so high level the underlying cause or causes could be anywhere in the system. So where do you start?

It starts with your mental model of the system, the more complete and more accurate the model the faster you are likely to be able to diagnose and remediate the issue. It’s both a map but also an understanding of how components interact that allows you to infer how signals in one part of the system can explain behaviour seen in another.

Without a mental model, metrics can be meaningless numbers and logs can lead you down lines of investigation that are totally fruitless. Your lines of investigation are limited because you don’t have a fundamental understandings of how things work beneath the external APIs.

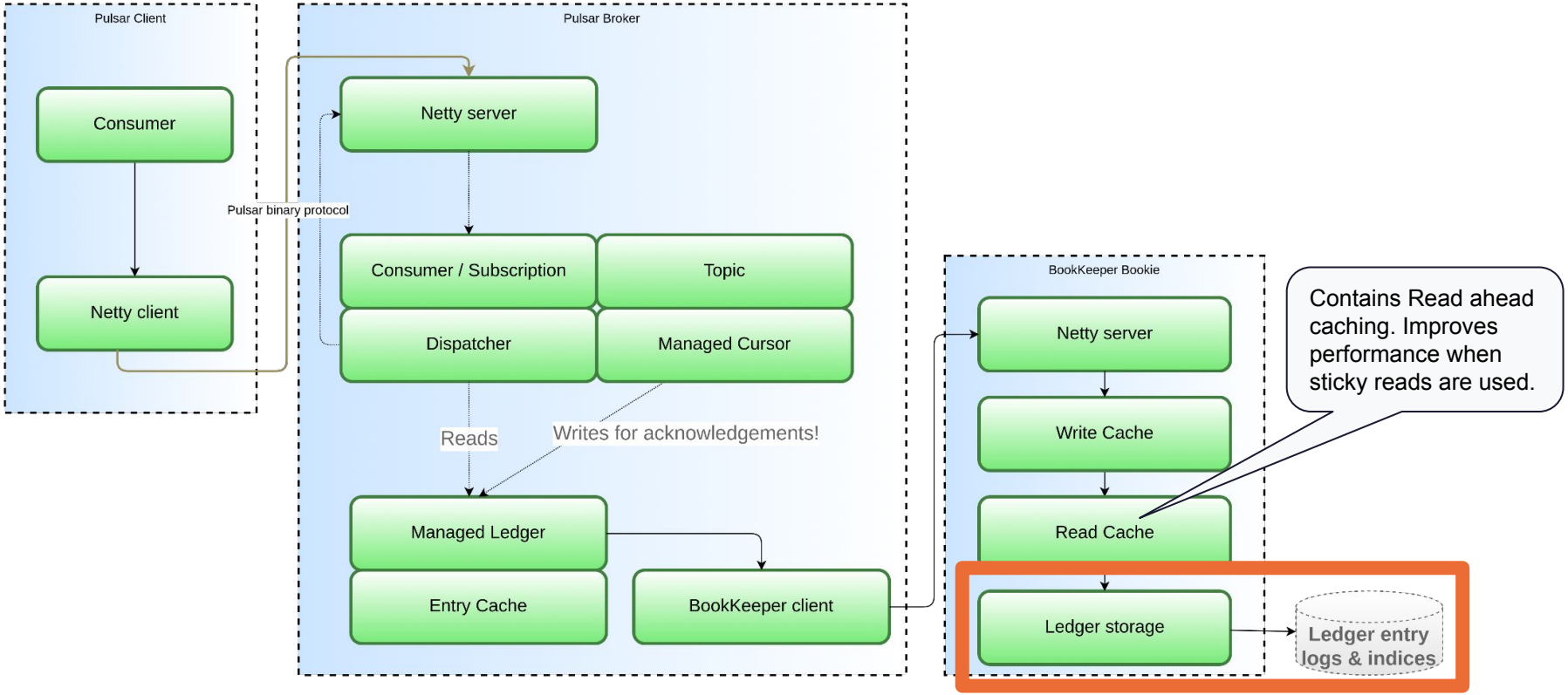
Having a mental model of the system is a necessity for fast and effective incident response as is having the right signals available to you — those are the metrics and logs.

Write path, Sending messages (simplified!)



“All models are wrong and some are useful”. This diagram doesn’t describe the sequence of actions and isn’t accurate. The purpose is to get a high idea of the write path.

Read path, Consuming messages (simplified)



"All models are wrong and some are useful". This diagram doesn't describe the sequence of actions and isn't accurate. The purpose is to get a high idea of the write path.

Messaging workload edge cases to cover in workload simulation

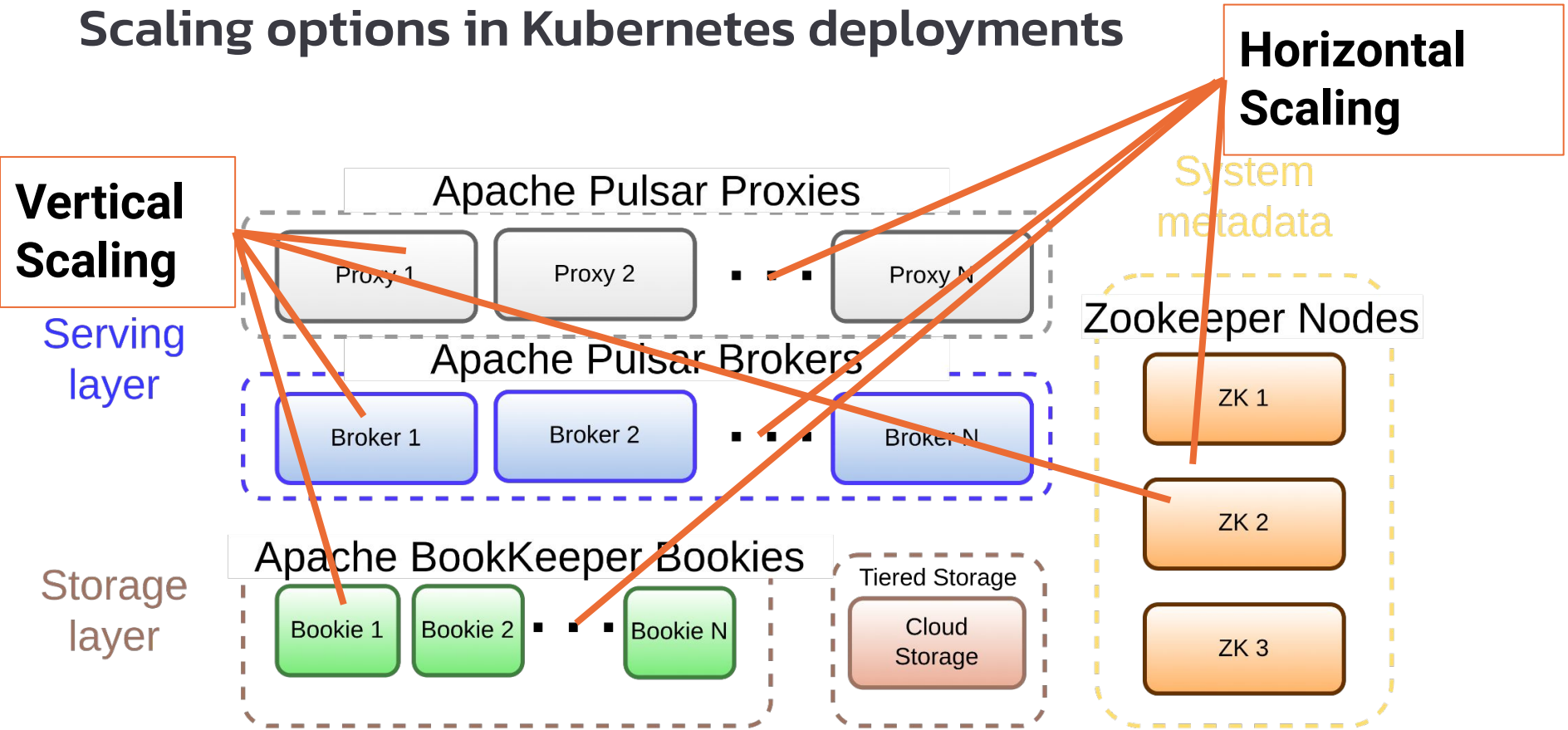
- Catch-up read / cold read
- Tailing read / hot read

- High fan-out (many subscriptions / consumers)
- Slow consumer & Fast producer

“Modify & Tune” in Apache Pulsar Kubernetes deployments

- Scaling out in Kubernetes deployments
- Scaling up in Kubernetes deployments
  - Kubernetes resource requests (and limits)
  - JVM settings for components
  - Changing Cloud VM types
- Apache Broker configuration
  - Broker cache
  - Pulsar Load balancer
  - Apache Bookkeeper client configuration
- Apache Bookkeeper configuration

Scaling options in Kubernetes deployments



Horizontal scaling in Kubernetes

- Horizontal scaling by defining the amount of Broker, Bookie and Zookeeper nodes.

- Kubernetes StatefulSets are used for Brokers, Zookeepers and Bookies
  - StatefulSet size can be increased for Brokers and Bookies to increase capacity without taking down the cluster
  - Scaling in Bookies (reducing cluster size) is currently a manual operation, which should be done with care (bookie node decommissioning).
  - Zookeeper cluster size modifications require more careful design. Zookeeper doesn't scale linearly, and it's usually a better option to stick with a size of 3 or 5 nodes and instead scale the nodes vertically.
- ReplicaSet is used for Proxies
  - Size can be increased and decreased "online"

Vertical scaling in Kubernetes

- In Kubernetes, the scheduler will take care of running the workload (pods in most cases) on the nodes with the requested CPU resources and memory amount
  - More memory or CPU resouces can be allocated
  - Disk volume sizes, types and amounts can be adjusted
  - In a statefulset, all pods will get the same amount of resources

- Running an isolated pod on a single node is an additional option for vertical scaling
  - Addresses the “noisy neighbor” problem caused by other Kubernetes pods
  - Might be needed to achieve stable high network or disk IO throughput.
  - This could also be a cost-effective option when an optimally sized VM type is used.

Java VM options tuning: heap size and direct memory

- Applies to Pulsar components
  - Broker pods
  - Proxy pods
  - Bookie pods
  - Zookeeper pods
- Environment variables to configure Java VM options
  - **PULSAR_MEM**
    - Default (2.10): -Xms2g -Xmx2g -XX:MaxDirectMemorySize=4g
    - Recommendation: add -XX:+ExitOnOutOfMemoryError option to prevent leaving the process in inconsistent state after there's a possible OutOfMemoryError.
  - **PULSAR_GC**
    - Default (2.10): -XX:+UseG1GC -XX:MaxGCPauseMillis=10 -XX:+ParallelRefProcEnabled -XX:+UnlockExperimentalVMOptions -XX:+DoEscapeAnalysis -XX:ParallelGCThreads=32 -XX:ConcGCThreads=32 -XX:G1NewSizePercent=50 -XX:+DisableExplicitGC
  - **PULSAR_EXTRA_OPTS**
    - Default (2.10): -Dpulsar.allocator.exit_on_oom=true …
    - Be aware of https://github.com/apache/pulsar/pull/13563 and https://github.com/apache/pulsar/issues/13382

Java process memory allocation & Linux OOM killer

- **JVM processes consume more memory than max heap size + max direct memory**
  - Total memory = Heap + Code Cache + Metaspace + Symbol tables +
    Other JVM structures + Thread stacks +
    Direct buffers + Mapped files +
    Native Libraries + Malloc overhead + …
    - See https://stackoverflow.com/a/53624438
- For Pulsar & Bookkeeper, allocate about 20-25% more RAM than max heap size + max direct memory
  - For example for -Xmx2g -XX:MaxDirectMemorySize=4g, request (2+4) * 1.25 = 7.5GB of RAM
  - Prefer Kubernetes resource requests instead of limits

Vertical scaling on cloud VMs

- Vertical scaling by defining / changing the cloud VM type for a **Kubernetes node**.
  - The VM type applies to a complete Kubernetes node that will run multiple types of pods by default.
- CPU and Memory are tied to a cloud VM type
  - VM types vary on AWS, Azure and GCP
    - AWS ec2 instance types: https://aws.amazon.com/ec2/instance-types/
    - Azure VM sizes: https://learn.microsoft.com/en-us/azure/virtual-machines/sizes
    - GCP machine types: https://cloud.google.com/compute/docs/machine-types
- Categorized
  - General purpose
  - Compute optimized
  - Memory optimized
  - Storage optimized
  - Accelerated computing (GPU)
- Beside CPU and Memory, VM type usually also impacts options for storage performance, storage options and available network bandwidth options

Selecting a specific VM type for a Pulsar component type

- In GCP GKE, there are “node pools” that can be created with a specified “Machine type”
- Kubernetes “taints and tolerations” are used to ensure that only selected pods can be run on the specific node pool
  - Node pool contains a NoSchedule taint for a specific key-value pair
  - Pods contain a toleration for the NoSchedule taint with the matching key-value pair
- Kubernetes “Pod affinity/anti-affinity” or “Pod Topology Spread Constraints” rules can be used to ensure a desired spread of components across the nodes. For example, it’s possible to have a rule that ensures that only one of a component type is run on a node. Rules can also be used to ensure spread across multiple availability zones.

- For example, it’s possible to make 1 bookie run on a 1 specially configured Kubernetes node type.
  - The benefit of this is that it’s possible to select an optimal VM type for running bookies.
    - This might be necessary to achieve high IOPS and consistent IO performance.

Cloud deployment storage options

- Cloud disks
  - SSD is needed. Bookkeeper and Zookeeper are not optimized for spinning disks.
  - IOPS vs. Throughput characteristics
    - Bookkeeper tuning for reducing IOPS
      - increase journalMaxGroupWaitMSec from 1 to 2. Also increase journalBufferedWritesThreshold if that limit is reached with the workload.
    - Increase the size of the journal size since larger cloud disks come with more IOPS
- Impact of disk size for performance on major cloud providers
  - GCP: https://cloud.google.com/compute/docs/disks/performance

Other BookKeeper storage options

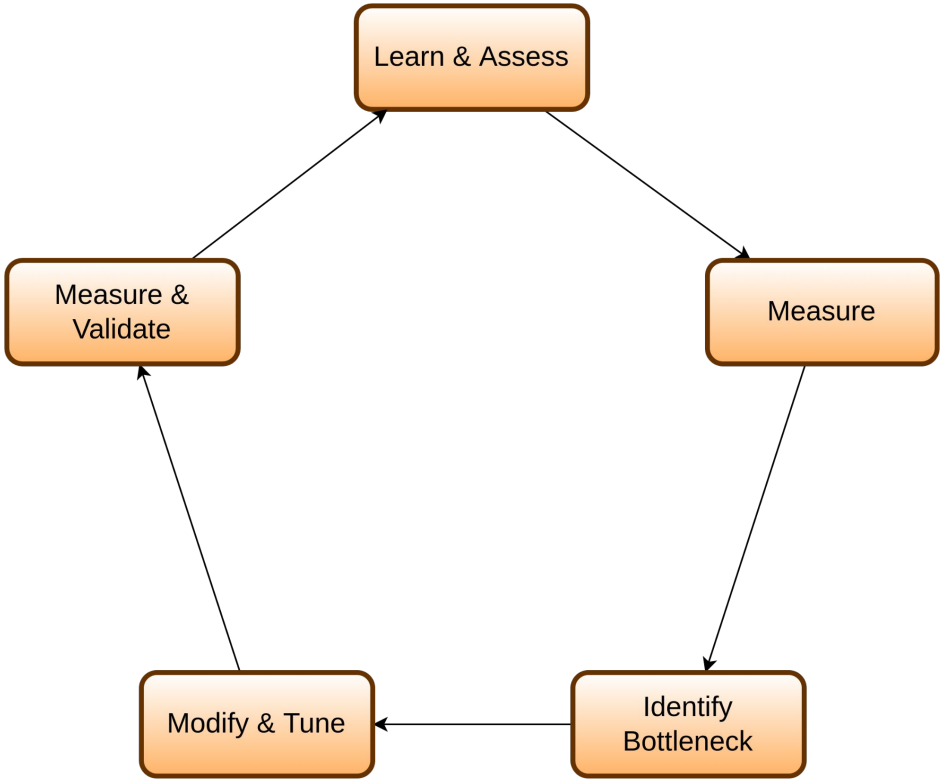
- Bookkeeper BP-46: Running without the journal
  - https://bookkeeper.apache.org/bps/BP-46-run-without-journal/
  - Available since 4.15.0 .
- Replication Factor, specified with write quorum, Q_w
  - Defaults to 2 in Pulsar
  - In Bookkeeper, $Q_w \geq Q_A$. Q_A- 1 nodes can be lost safely.
    - With the default $E=Q_W=Q_A=2$, it's safe to lose 1 node at a time in a particular ensemble.
    - With $E=Q_W=Q_A=3$, it's safe to lose 2 nodes at a time in a particular ensemble.
  - Sticky reads aren't used unless $E=Q_w$. That's why ledger "striping" ($E>Q_w$) will have a negative impact on reads.
- Multiple journal disks, Multiple ledger disks
- Rackawareness / Availability Zone awareness

Pulsar Broker configuration tuning

- Broker cache tuning so that catch-up reads can be handled efficiently
  - Some important improvements for broker cache have been contributed to Apache Pulsar recently and aren't yet released
    - Support caching to drain backlog consumers: https://github.com/apache/pulsar/pull/12258
    - Bug fix for cache eviction: https://github.com/apache/pulsar/pull/17273 , workaround is to set cacheEvictionByMarkDeletedPosition=true
    - Do not send duplicate reads to BK/offloaders: https://github.com/apache/pulsar/pull/17241
  - Adjust settings based on metrics in dashboards. It's currently an evolving topic since there are multiple important improvements that haven't been released. New dashboard is necessary to follow metrics.
- Bookkeeper client tuning for backpressure handling
  - Evolving topic: https://github.com/apache/pulsar/issues/10439 ,
- Pulsar load balancer tuning
  - Ensuring correct resource size for network and CPU
    - Detecting network bandwidth doesn't work in all cases and must be configured manually with the use of loadBalancerOverrideBrokerNicSpeedGbps config option
    - The usage of "-XX:ActiveProcessorCount=n" when Kubernetes resource requests are used for CPU because of bug https://github.com/apache/pulsar/issues/17815

Key takeaways

- **Follow a systematic performance tuning method**
- Create performance models to help analysis and bottleneck identification
- Incrementally learn how to tune the deployment and brokers, bookkeepers and zookeepers in a way that meet **your** requirements and use case
- Contribute to Apache Pulsar!
  - Mailing list, GitHub Discussions, Slack



We serve real-time applications with an open data stack that just works