Running VMware Tanzu RabbitMQ 1.3 on VMware Tanzu Kubernetes Grid

vmware

Table of Contents

Overview

Note: This solution provides an overview of the performance improvements in new version of VMware Tanzu® RabbitMQ® and guidelines for running Apache Kafka running on VMware Tanzu Kubernetes Grid™. The reference architecture applies to general VMware vSphere® and VMware vSAN™ platforms.

Introduction

VMware Tanzu RabbitMQ for Kubernetes 1.3 contains RabbitMQ 3.10 that was released in May 2022, with many new features and improvements. Additionally, this commercial offering includes all the required dependencies for RabbitMQ as well as some commercial plugins making it easier to deploy this cloud native solution.

Apache Kafka is an open-source distributed event streaming platform for high-performance data pipelines and streaming analytics. It is used as a popular message queue for distributed systems and is commonly used to stream data in the IoT use cases.

This paper provides the performance improvements of Tanzu RabbitMQ for Kubernetes 1.3 on VMware Tanzu Kubernetes Grid. And it covered the general functional testing and guidelines for running Apache Kafka on Tanzu Kubernetes Grid.

Prerequisites

You must first set up a reference architecture environment for Tanzu Kubernetes Grid. And then, the material takes you through the steps to install VMware Tanzu RabbitMQ for Kubernetes.

Technology Overview

The technology components in this solution are:

- VMware Tanzu Kubernetes Grid Multi-Cloud (TKGm)
- Tanzu RabbitMQ for Kubernetes
- Apache Kafka
- Strimzi

VMware Tanzu Kubernetes Grid

VMware Tanzu Kubernetes Grid provides organizations with a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations. You can deploy VMware Tanzu Kubernetes Grid across software-defined datacenters (SDDC) and public cloud environments, including vSphere, Microsoft Azure, and Amazon EC2.

VMware Tanzu Kubernetes Grid provides the services such as networking, authentication, ingress control, and logging that a production Kubernetes environment requires. It can simplify operations of large-scale, multi-cluster Kubernetes environments and keep your workloads properly isolated. Also, it automates lifecycle management to reduce your risk and shift your focus to more strategic work.

VMware Tanzu Kubernetes Grid Multi-Cloud provides a consistent, upstream-compatible, regional substrate that is ready for end-user workloads and ecosystem integrations.

VMware Tanzu RabbitMQ for Kubernetes

VMware Tanzu RabbitMQ for Kubernetes provides the building blocks for a cloud native messaging and streaming service that you can deploy on any Kubernetes cluster.

Streams are a new feature in RabbitMQ for both the open source RabbitMQ and commercial Tanzu RabbitMQ editions. It delivers some of the benefits of log structures, like fast data transfer and the ability to replay and reprocess messages, combined with the best Tanzu RabbitMQ features like flexible routing and command and response, all with a high degree of data safety. Streams support existing protocols as well as a specific new binary protocol for improved speed and throughput.

Streams can be deployed alongside conventional messaging and queues to create a hybrid message topology making it easier for users to migrate from one technology to another.

See VMware Tanzu RabbitMQ 1.3 for detailed information.

Apache Kafka

Apache Kafka is a community distributed event streaming platform capable of handling trillions of events a day. Initially conceived as a messaging queue, Kafka is based on an abstraction of a distributed commit log. Since being created and open sourced by LinkedIn in 2011, Kafka has quickly evolved from messaging queue to a full-fledged event streaming platform.

A cluster of Kafka brokers handles delivery of messages.

A broker uses Apache ZooKeeper for storing configuration data and for cluster coordination. Before running Apache Kafka, an Apache ZooKeeper cluster has to be ready. Apache ZooKeeper is a core dependency for Kafka as it provides a cluster coordination service, storing and tracking the status of brokers and consumers. ZooKeeper is also used for controller election.

Strimzi

Strimzi provides a way to run an Apache Kafka cluster on Kubernetes in various deployment configurations. For development, it is easy to set up a cluster in a few minutes. For production, you can tailor the cluster to your needs, using features such as rack awareness to spread brokers across availability zones, and Kubernetes taints and tolerations to run Kafka on dedicated nodes. You can expose Kafka outside Kubernetes using NodePort, Load balancer, Ingress and OpenShift Routes, depending on your needs, and these are easily secured using TLS. The Operators provided with Strimzi are purpose-built with specialist operational knowledge to effectively manage Kafka.

Solution Configuration

Architecture

In our solution, we deployed the VMware Tanzu RabbitMQ test environment using Tanzu Kubernetes Grid on a 4-node cluster. Firstly, we deployed the Tanzu Kubernetes Grid management cluster with Tanzu CLI by either an installer UI or a configuration file. Refer to Prepare to deploy Management clusters to vSphere for more details. After deploying a management cluster in vSphere, use the Tanzu CLI to deploy the Tanzu Kubernetes Grid workload cluster. The Tanzu Kubernetes cluster was deployed through a configuration file with customized variables for cluster settings and scalability. Refer to Deploy Tanzu Kubernetes Clusters to vSphere for more details. To support the Tanzu RabbitMQ and Apache Kafka deployment, we recommend configuring a Tanzu Kubernetes Grid workload cluster with at least 3 nodes; each node is requested with a minimum value of 24 vCPU, 72GB memory, and 2TB disk space.

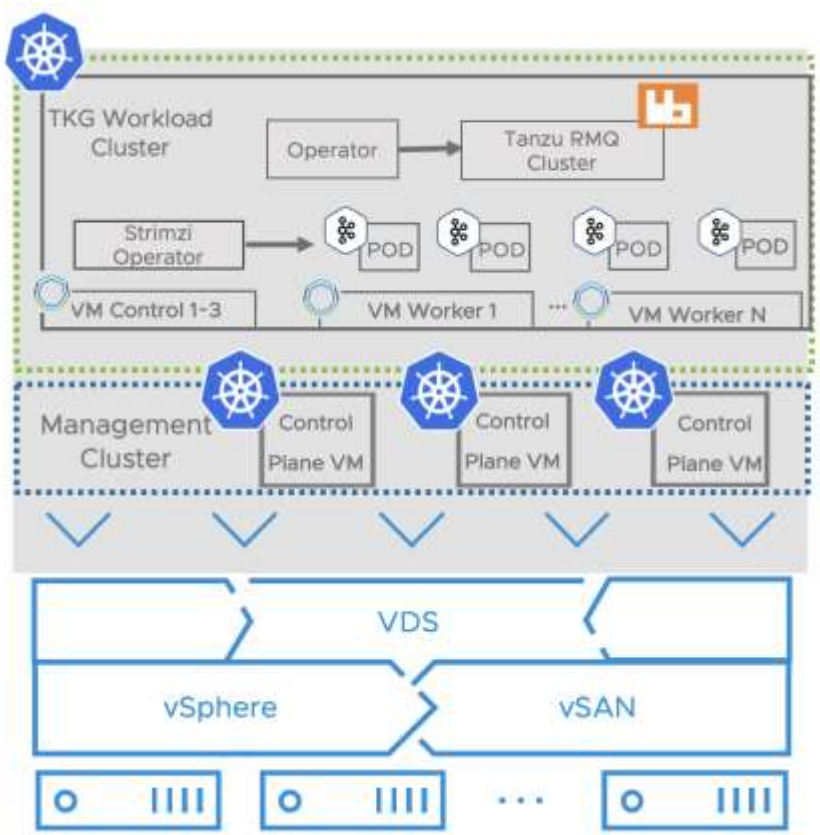


Figure 1: Tanzu RabbitMQ and Apache Kafka Running on Tanzu Kubernetes Grid

Then we deployed VMware Tanzu RabbitMQ for Kubernetes to a Tanzu Kubernetes Grid workload cluster. All the Tanzu RabbitMQ components of the package can be pulled from the registry. Then we installed the Tanzu RabbitMQ Operator to the cluster. After the installation was completed, create a RabbitMQ cluster via the configuration file.

We choose Strimzi operator to run an Apache Kafka cluster on Tanzu Kubernetes Grid in customized deployment configurations. Strimzi provides a Helm chart to deploy the Cluster Operator. Apache Kafka components are provided for deployment to Kubernetes with the Strimzi distribution. The Cluster Operator watches for updates in the namespaces where the Kafka resources are deployed. Then we deployed a Kafka cluster via the configuration files. Strimzi also installs a ZooKeeper cluster and adds the necessary configuration to connect Kafka with ZooKeeper.

Hardware Resource

Table 1: Hardware Configuration

PROPERTY	SPECIFICATION
CPU	2 x Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz, 28 core each
RAM	480GB
Network adapter	2 x Intel 10Gb Ethernet Controller X550
Storage adapter	1 x Dell HBA330 Mini Adapter
Disks	Cache - 2 x 1.46TB NVMe Capacity - 8 x 1.75TB SSD

Software Resource

Table 2: Software Resources

SOFTWARE	VERSION	PURPOSE
VMware vSphere	7.0 U3c	VMware vSphere is a suite of products: vCenter Server and ESXi.
Tanzu RabbitMQ for Kubernetes	1.3	Based on the widely popular open source RabbitMQ messaging system, Tanzu RabbitMQ for Kubernetes is designed to work seamlessly with Tanzu and also run on any certified Kubernetes distribution.
Tanzu CLI	V0.11.6	Command line interface that allows deploying CNCF conformant Kubernetes clusters to vSphere and other cloud infrastructure.
Kubernetes	V1.22.9	Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

Kubernetes OVA for Tanzu Kubernetes Grid	Ubuntu 20.04 and Kubernetes v1.22.9+vmware.1	A base image template for the Kubernetes Operating System of Tanzu Kubernetes Grid management and workload clusters.
RabbitMQ PerfTest	V2.18.0	A throughput testing tool that is based on the Java client and can be configured to simulate basic workloads and more advanced workloads as well.
RabbitMQ Stream Java Client	V0.7.0	A Java library to communicate with the RabbitMQ Stream Plugin. It contains a performance tool to allow creating and deleting streams as well as publishing to and consuming from these streams to test the performance.
Strimzi	V0.31.0	A way to run an Apache Kafka cluster on Kubernetes in various deployment configurations.
Apache Kafka	V3.2.1	An open-source event streaming platform, it aims to provide the user with a unified, high latency, high throughput platform for handling real-time data.

Table 3. VM Configuration

VM Role	vCPU	Memory (GB)	Storage (GB)	VM Count
Tanzu Kubernetes Grid Management cluster – Control Plane VM	2	4	20	1
Tanzu Kubernetes Grid Management cluster – Worker node VM	2	4	20	1
Tanzu Kubernetes Grid Workload cluster – Control Plane VM	4	32	120	3
Tanzu Kubernetes Grid Workload cluster – Worker node VM	24	72	2000	3

Best Practice

The following recommendations provide the best practices and sizing guidance to run Tanzu RabbitMQ clusters on Tanzu Kubernetes Grid.

- Tanzu Kubernetes Grid:
  - Use the latest ova image to create Tanzu Kubernetes Grid management and workload cluster that runs Tanzu RabbitMQ cluster.
  - Customize and pre-allocate enough CPU and memory resources for the Tanzu Kubernetes Cluster. Refer to Performance Best Practices for Kubernetes with VMware Tanzu for sizing guidance for Tanzu Kubernetes Grid.
- vSAN Storage:
  - vSAN supports the dynamic persistent volume provisioning with Storage Policy Based Management (SPBM).
  - Failures to Tolerate (FTT) is recommended to set to 1 failure - RAID-1 (Mirroring).
  - Enable vSAN Trim/Unmap to allow space reclamation for persistent volumes.
- VMware Tanzu RabbitMQ for Kubernetes:
  - It is recommended to use clusters with an odd number of nodes (3, 5, 7, and so on) so that when one node becomes unavailable, the service remains available, and a clear majority of nodes can be identified. See the "Failure tolerance characteristics of clusters of various size" table in https://www.rabbitmq.com/streams.html for details.
  - It is recommended to use multiple publishers during Benchmark running, a single publisher is unable to fully utilize the queue with very small messages.
  - It is recommended to disable credit flow when running a single queue, otherwise a single publisher will be throttled.
  - As streams persist all data to disks, it is recommended to use the fastest disks possible
  - Stream and queue data structures are not competitors, they are suited for different use cases:
    i. RabbitMQ stream is suited for high throughput, reading simultaneously by multiple applications, and accessing messages from anywhere in the queue. Stream is persistent by default, and the messages are stored directly on disks without minimal memory overhead. Streams are typically more lightweight than queues.
    ii. RabbitMQ queue is suited for long-running tasks or performing reliable background jobs and is used as the middleman between microservices or among legacy applications.
  - Using sub-entry batching can increase throughput at the cost of increased latency and potential duplicated messages. Use sub-entry batching when appropriate in your use cases.
- Apache Kafka
  - It is recommended to use high-performance disks and pod affinity policy avoid putting all the Kafka brokers on a single node.
  - Use Strimzi operator to bootstrap a Kafka cluster almost without any configuration.
  - It is recommended to enable compression, because it improves network and disk utilization.

Refer to the next two chapters for solution validation performance.

Application Validation

This solution testing showcases VMware Tanzu RabbitMQ running on Tanzu Kubernetes Grid for performance improvement and Apache Kafka running Tanzu Kubernetes Grid for performance validation.

VMware Tanzu RabbitMQ Performance Testing Scope and Result

VMware Tanzu RabbitMQ for Kubernetes 1.3 includes RabbitMQ 3.10 with many new features and improvements on Quorum Queue.

We used RabbitMQ PerfTest to generate loads for the RabbitMQ cluster to compare quorum queues with the previous version. In the performance testing, we provisioned a cluster of three Tanzu RabbitMQ pods, each with 8 vCPUs, 16GB of RAM, and 2000GB storage.

Scenario 1: We used just 1 quorum queue with fast publisher and consumer. Compare with the previous version, VMware Tanzu RabbitMQ 1.3 improved 30% throughput. And we tested with message size of 10, 100, 1000, and 5000 bytes.

Figure 1. Messages Published/s of 10, 100, 1000 and 5000 Bytes

We ran the test with different message sizes:

- Each quorum queue published rate average at 55,634 message/s, consumed average at 55,628 message/s at small message size.
- Each quorum queue published rate average at 15,334 message/s, consumed average at 15,207 message/s at large message size.

Scenario 2: We published large million messages and then consumed all of them. It took ~130s to publish 10 million messages

and ~170s to consume the messages.

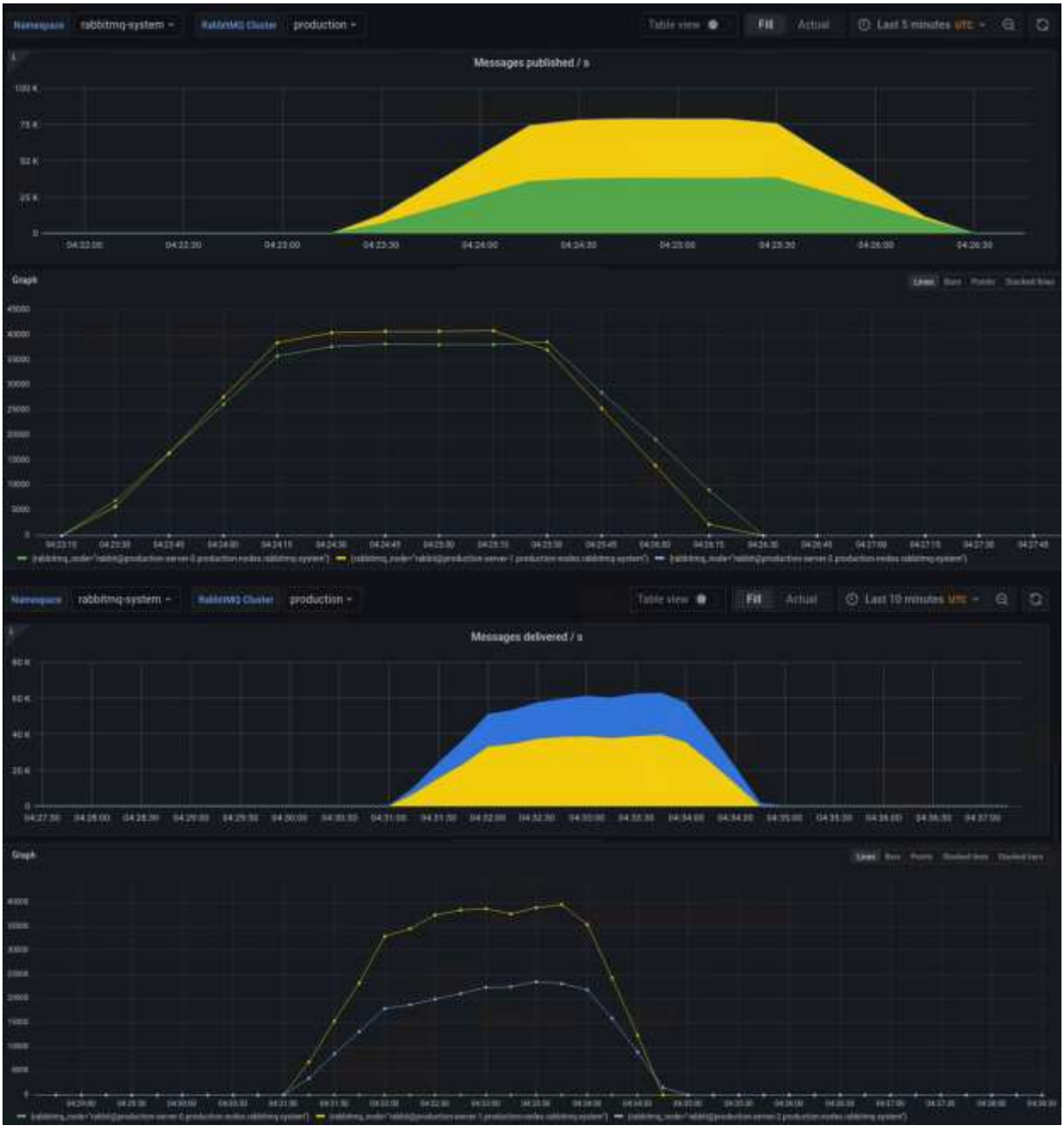


Figure 2. 10 million Message Published and Consumed

Figure 3. Memory Available before Publishers Blocked

During the process of publishing and consuming, Quorum queue used ~3GB memory. But when we published 50 million messages, the cluster hit a memory alarm and stopped due to waiting for publisher confirmation for too long.

Scenario 3: We tested VMware Tanzu RabbitMQ streams with different number of streams. It did not improve much than the previous version. The detailed published and consumer results were shown below:

- Single RabbitMQ streams published 976,589 message/s, consumed 973,953 msg/s
- Three (3) streams published 2,475,177 msg/s, consumed 2,473,692 msg/s

Figure 4. Stream Test Results

Kafka Performance Testing Scope and Result

This solution testing showcases Apache Kafka running on Tanzu Kubernetes Grid for performance validation.

Apache Kafka is a community distributed event streaming platform capable of handling trillions of events a day. Initially conceived as a messaging queue, Kafka is based on an abstraction of a distributed commit log. Strimzi provides a way to run an Apache Kafka cluster on Kubernetes in various deployment configurations.

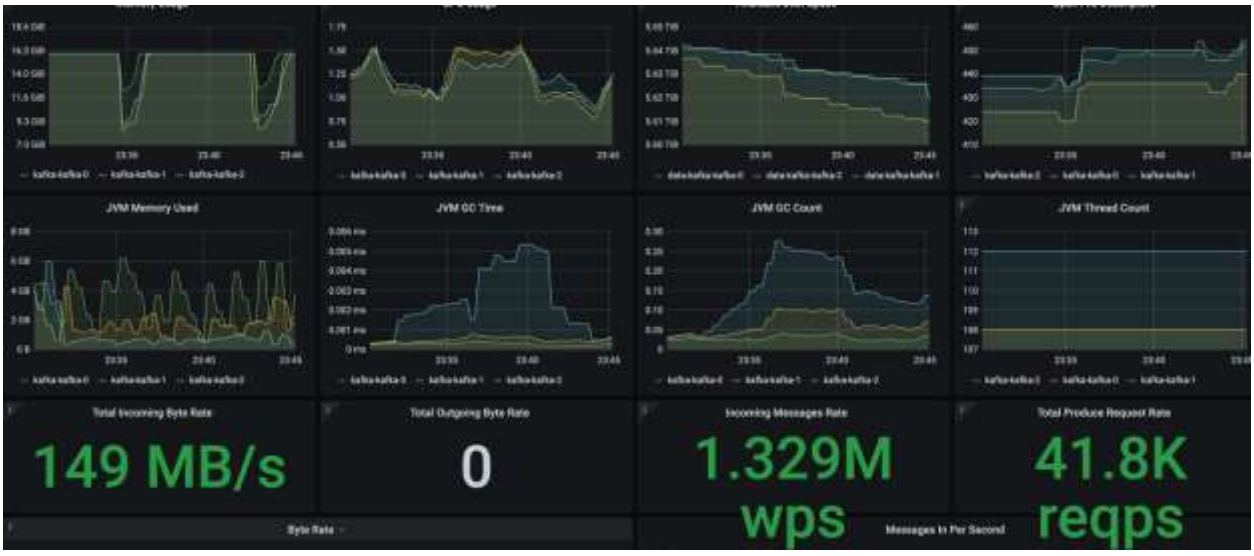
Strimzi provides a Helm chart to deploy the Cluster Operator. The Cluster Operator watches for updates in the namespaces where the Kafka resources are deployed. You can now deploy a Kafka cluster via the configuration files. Strimzi also installs a ZooKeeper cluster and adds the necessary configuration to connect Kafka with ZooKeeper. In configuration file, it can export the metrics that we can get the performance through Prometheus and Grafana.

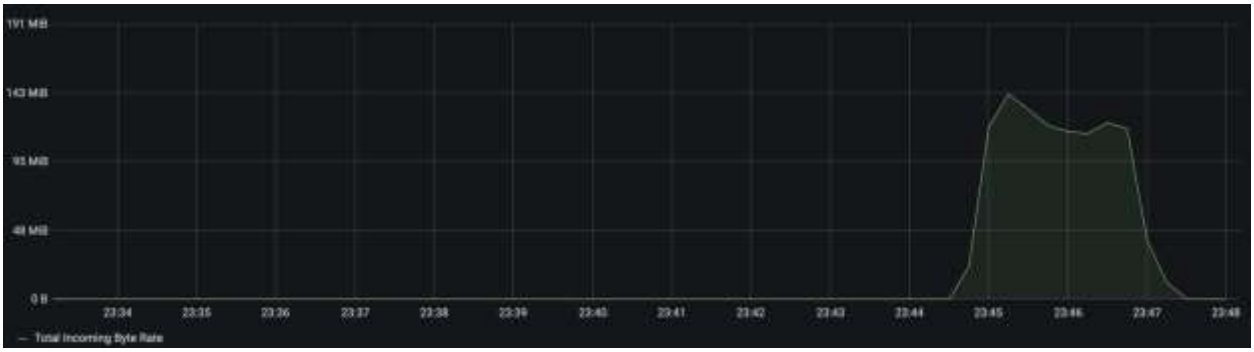
In the solution testing, we allocated 3 broker pods with 8 vCPU and 32 GB RAM in container resource spec. We created the topic with 12 partitions and replication factor of 1 or 3.

Scenario 1: We created a job with the configuration to achieve the maximum throughput. For single Producer throughput with no replication and 3 replicas, each broker runs the following producer-perf test command below:

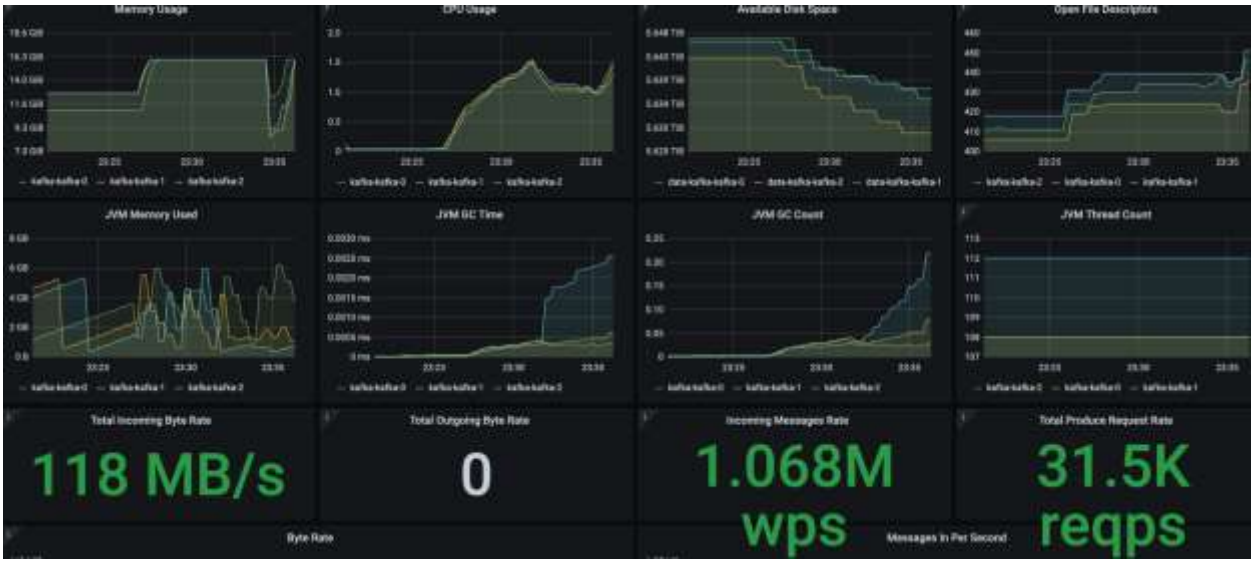
./kafka-producer-perf-test.sh --topic test_p12_r1 --producer-props bootstrap.servers=bootstrap.kafka.svc:9092 buffer.memory=67108864 batch.size=64000 acks=1 --record-size 100 --throughput -1 --num-records 50000000

We got an average of 492K records/s with 100 bytes per record, 46.9MB/s throughput and 1.13ms latency. For 3 producers, we got the total peak throughput at 149MB/s.

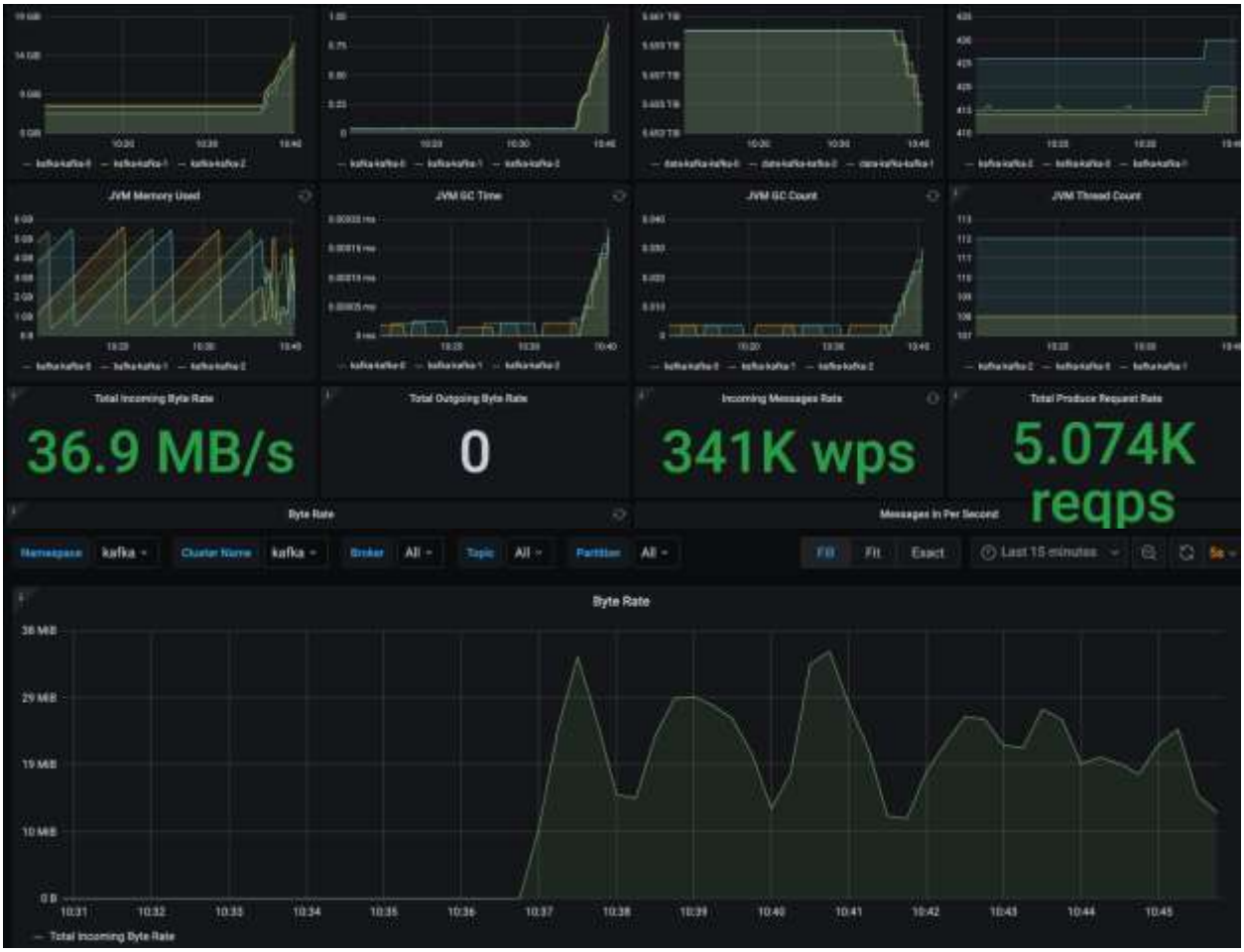




Then we created a topic with 3x replication and got the replica async. It achieved the total peak throughput at 118MB/s.



Next, we created a topic with 3x replication and got the replica sync. It means all the replicas have been persisted before acknowledging the producer. We can see a little bit more than 3x performance reduction. That is related to the overhead of sending and receiving acknowledgements by the brokers.



We ran the commands to consume all the messages:

```
./kafka-consumer-perf-test.sh --bootstrap-server localhost:9092 --messages 50000000 --topic
test_p12_r3 --threads 1
```

We can see the throughput was bigger than the producer with no replica.

Apache Kafka is a stream processing and event-driven architecture, end-to-end latency measures the time a message traverses the pipeline from the producer through the system to the consumer. To test end-to-end latency, run the below command:

```
sh kafka-run-class.sh kafka.tools.EndToEndLatency localhost:9092 test_p12_r3 10000 all 1000
```

```
175000  6.354546999999999
176000  1.458957
177000  1.4203610000000002
178000  1.42405
179000  1.115937
180000  1.518577
181000  1.516256
182000  1.540252
183000  1.47108
184000  1.920327
185000  1.540208
186000  1.459633
187000  1.9301
188000  1.088621
189000  1.572921
190000  1.260847
191000  1.466928
192000  1.562683
193000  1.396028
194000  1.452801
195000  1.471385
196000  1.594881
197000  1.101534
198000  1.463369
199000  1.527193
Avg latency: 2.7601 ms

Percentiles: 50th = 1, 99th = 14, 99.9th = 209
```

The topic test_p12_r3 was created with 12 partitions and RF=3, producer acknowledgements were set to all, the highest durability. We got the average of 3ms latency at 1KB message size.

Conclusion

Apache Kafka and RabbitMQ are two pub/sub systems, RabbitMQ is a general-purpose message system that supports protocols including AMQP and MQTT. Now it is also being used for streaming use cases. VMware Tanzu RabbitMQ provides all the messaging and streaming capabilities of RabbitMQ in an enterprise-grade supported distribution. It includes a validated, production-ready container image, warm standby replication to keep their schemas and message in-sync. Kafka is a durable message broker that enables applications to process, persist and re-process streamed data. While the Venn diagram of use cases, these two technologies could fulfill was very tight, you would analyze their differences and also consider the skills needed to operate the services and the developer communities and then decide which of the two best fits your use case.

Additional Resources

For more information about Tanzu RabbitMQ, you can explore the following resources:

- VMware vSphere
- VMware vSAN
- VMware Tanzu Kubernetes Grid
- VMware Tanzu RabbitMQ
- Running VMware Tanzu RabbitMQ on VMware Tanzu Kubernetes Grid
- RabbitMQ Stream Java Client
- Installing VMware Tanzu RabbitMQ for Kubernetes
- Solution Deployment and Configuration Files

About the Author and Contributors

Yimeng Liu, Senior Technical Marketing Manager in the Workload Technical Marketing Team of the Cloud Infrastructure Big Group, wrote the original version of this paper. The following reviewers also contributed to the paper contents:

- Catherine Xu, Senior Manager of the Workload Technical Marketing Team in VMware
- Howard Twine, Senior Product Manager of Tanzu RabbitMQ in VMware


vmware

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.

vmware