6 Method and Evaluation Strategy

6.1 Method

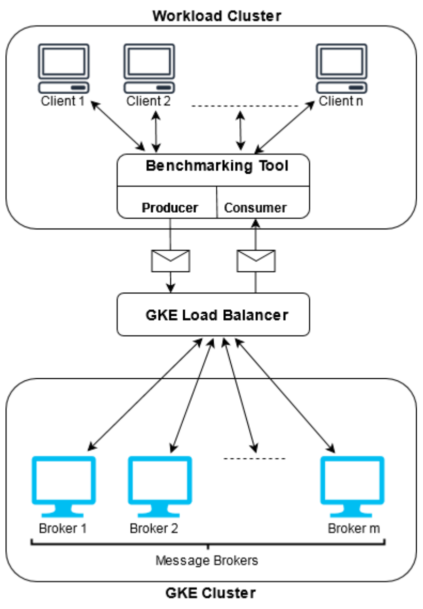

Figure 4: System overview.

6.2 Evaluation Strategy

6.2.1 Test Plan

In this section, different queue settings have been tested in order to create a test model that will be followed for evaluating the message queue systems. Kafka consists of many different parameters that could be tuned for increasing the brokers' performance. RabbitMQ have different have different kinds of queues, namely classic and quorum queues. The quorum queue type uses the Raft consensus algorithm for safely replicating the queue data between the nodes in the cluster, while it is possible to use queue mirroring in classic queue type for replicating the data. Small tests have been conducted in order to create a test model that consists of important parameters for the message brokers.

Kafka Tests

Performed tests for Kafka with a workload of 7 GB (each msg 1000 bytes)

On e2-standard-2 machine type (2 vCPU and 8 GB of memory, backed by 2 physical cores), and maximum Egress bandwidth 4 (Gbps):

1. **1 Broker Cluster:**
    (a) 3 producers/consumers:
    - Producer Throughput: ~ **37 MB/s**
    - Consumer Throughput: ~ **94.5 MB/s**
    - Mean Latency: ~ **37834 ms**
2. **3 Broker Cluster:**
    (a) 3 producers/consumers:
    - Producer Throughput: ~ **81 MB/s**
    - Consumer Throughput: ~ **142 MB/s**
    - Mean Latency: ~ **29052 ms**

On e2-highcpu-16 vCPU and 16 GB of memory, and maximum Egress bandwidth 16 (Gbps):

1. **1 Broker Cluster:**
   (a) 1 producer/consumer:
      - Producer Throughput: ~ **43 MB/s**
      - Consumer Throughput: ~ **43 MB/s**
      - Mean Latency: ~ **71 ms**

   (b) 3 producers/consumers:
      - Producer Throughput: ~ **40 MB/s**
      - Consumer Throughput: ~ **116 MB/s**
      - Mean Latency: ~ **11363 ms**

   (c) 5 producers/consumers:
      - Producer Throughput: ~ **50 MB/s**
      - Consumer Throughput: ~ **236 MB/s**
      - Mean Latency: ~ **16865 ms**

2. **3 Broker Cluster:**
   (a) 1 producer/consumer:
      - Producer Throughput: ~ **66 MB/s**
      - Consumer Throughput: ~ **66 MB/s**
      - Mean Latency: ~ **4080 ms**

   (b) 3 producers/consumers:
      - Producer Throughput: ~ **132 MB/s**
      - Consumer Throughput: ~ **135 MB/s**
      - Mean Latency: ~ **43011 ms**

   (c) 5 producers/consumers:
      - Producer Throughput: ~ **241 MB/s**
      - Consumer Throughput: ~ **209 MB/s**
      - Mean Latency: ~ **59293 ms**

3. **5 Broker Cluster:**
   (a) 1 producer/consumer:
      - Producer Throughput: ~ **60 MB/s**
      - Consumer Throughput: ~ **59 MB/s**
      - Mean Latency: ~ **4403 ms**

   (b) 3 producers/consumers:
      - Producer Throughput: ~ **51 MB/s**
      - Consumer Throughput: ~ **139 MB/s**
      - Mean Latency: ~ **13555 ms**

   (c) 5 producers/consumers:
      - Producer Throughput: ~ **88 MB/s**
      - Consumer Throughput: ~ **323 MB/s**
      - Mean Latency: ~ **27545 ms**

We can e.g. in the last scenario see (5 Broker Cluster) that increasing the replication and partitions in the system, will not necessarily increase performance.

RabbitMQ Tests

Performed tests for RabbitMQ (Classic Mirrored Queue vs Quorum Queue) with a workload of 7 GB (each msg 1000 bytes)

On e2-highcpu-16 vCPU and 16 GB of memory, and maximum Egress bandwidth 16 (Gbps):

1. **1 Broker Cluster:**
   (a) Classic Queue (NOTE NO MIRRORING):
      i. 1 producer/consumer:
         - Producer Throughput: ~ **58 MB/s**
         - Consumer Throughput: ~ **58 MB/s**
         - Mean Latency: ~ **51 ms**

      ii. 3 producers/consumers:
         - Producer Throughput: ~ **43 MB/s**
         - Consumer Throughput: ~ **129 MB/s**
         - Mean Latency: ~ **309 ms**

      iii. 5 producers/consumers:
         - Producer Throughput: ~ **32 MB/s**
         - Consumer Throughput: ~ **158 MB/s**
         - Mean Latency: ~ **581 ms**

   (b) Quorum Queue:
      i. 1 producer/consumer:
         - Producer Throughput: ~ **14 MB/s**
         - Consumer Throughput: ~ **14 MB/s**
         - Mean Latency: ~ **315 ms**

      ii. 3 producers/consumers:
         - Producer Throughput: ~ **4.8 MB/s**
         - Consumer Throughput: ~ **14 MB/s**
         - Mean Latency: ~ **2932 ms**

      iii. 5 producers/consumers:
         - Producer Throughput: ~ **3.6 MB/s**
         - Consumer Throughput: ~ **17 MB/s**
         - Mean Latency: ~ **4500 ms**

On e2-highcpu-16 vCPU and 16 GB of memory, and maximum Egress bandwidth 16 (Gbps):

1. **3 Broker Cluster:**
   - (a) Classic Mirrored Queue:
     - i. 1 producer/consumer:
       - Producer Throughput: ~ **20 MB/s**
       - Consumer Throughput: ~ **20 MB/s**
       - Mean Latency: ~ **216 ms**
     - ii. 3 producers/consumers:
       - Producer Throughput: ~ **5.1 MB/s**
       - Consumer Throughput: ~ **12 MB/s**
       - Mean Latency: ~ **54184 ms**
     - iii. 5 producers/consumers:
       - Producer Throughput: ~ **4.5 MB/s**
       - Consumer Throughput: ~ **19 MB/s**
       - Mean Latency: ~ **44160 ms**
   - (b) Quorum Queue:
     - i. 1 producer/consumer:
       - Producer Throughput: ~ **12 MB/s**
       - Consumer Throughput: ~ **12 MB/s**
       - Mean Latency: ~ **398 ms**
     - ii. 3 producers/consumers:
       - Producer Throughput: ~ **5.8 MB/s**
       - Consumer Throughput: ~ **17 MB/s**
       - Mean Latency: ~ **1969 ms**
     - iii. 5 producers/consumers:
       - Producer Throughput: ~ **3 MB/s**
       - Consumer Throughput: ~ **12.5 MB/s**
       - Mean Latency: ~ **6985 ms**

On e2-highcpu-16 vCPU and 16 GB of memory, and maximum Egress bandwidth 16 (Gbps):

1. **5 Broker Cluster:**
   (a) Classic Mirrored Queue:
      i. 1 producer/consumer:
         - Producer Throughput: ~ **18 MB/s**
         - Consumer Throughput: ~ **18 MB/s**
         - Mean Latency: ~ **245 ms**
      ii. 3 producers/consumers:
         - Producer Throughput: ~ **6 MB/s**
         - Consumer Throughput: ~ **17 MB/s**
         - Mean Latency: ~ **1943 ms**
      iii. 5 producers/consumers:
         - Producer Throughput: ~ **4.3 MB/s**
         - Consumer Throughput: ~ **20 MB/s**
         - Mean Latency: ~ **28000 ms**
   (b) Quorum Queue:
      i. 1 producer/consumer:
         - Producer Throughput: ~ **16 MB/s**
         - Consumer Throughput: ~ **16 MB/s**
         - Mean Latency: ~ **309 ms**
      ii. 3 producers/consumers:
         - Producer Throughput: ~ **6 MB/s**
         - Consumer Throughput: ~ **17 MB/s**
         - Mean Latency: ~ **1808 ms**
      iii. 5 producers/consumers:
         - Producer Throughput: ~ **3.1 MB/s**
         - Consumer Throughput: ~ **15 MB/s**
         - Mean Latency: ~ **16941 ms**

- Classic Mirrored Queues use its recommended replication factor of N/2 + 1, while the quorum group size is the size same as number nodes in the cluster
- Quorum queues focuses on data safety/recovery, where the declared queues can be durable. However, in classic mirrored queues queues can not be declared as durable and are therefore lost on broker shutdown/restarts.
  - Quorum queues trade safety over performance compared to classic queues
    - However, higher availability and reliability of queues can be gained by using mirroring for classic queues

6.2.2 Conclusions

- **Overall**:

  - After testing Kafka (since it is believed to have most throughtput) it showed that 7GB data takes about 2 min time:
    - Data to send to the cluster should not be less than 7GB, otherwise insignificant data, not reaching its peak
    - Different machine types has shown to affect the brokers' performance in the systems
    - The size of the messages affect the brokers' performance, e.g. optimal to have 1kB messages for Kafka
  - **RabbitMQ**:
    - Have tested RabbitMQs quorum queues against classic mirrored queues:
      - If only classic queues are used (no replication/mirroring), then classic have much higher performance than quorum queues but when testing classic mirrored queues against quorum queues, the classic mirrored queues have slightly better performance in terms of both throughput and latency
      - Classic mirrored queues have higher throughput than quorum queues at the cost of less reliability
  - **Kafka**:
    - Important parameters such as default.replication and num.partitions was shown to be really important. A test showed that replicating the data with multiple partitions (being spread across the nodes) influence both the producer and consumer throughput a lot. In order to be able to reach higher throughput, it is important to use multiple producers/consumers for reaching higher level of parallelism in the system.
    - If compression techniques are used by the producers, the throughput could be boosted up to 2-3 times (using Snappy), however, this would result in unfair results
    - Also tested to disable batching for producers -¿ resulted in very low throughput since producers in Kafka uses a batch buffer which send records that have been processed within a certain time. If the batching is disabled, then the producer will send messages in somewhat synchronized manner (send messages on after one)
    - The consumers are set to consume from all available partitions for being able to reach higher throughput
  - **Nats Streaming** - Didn't find any broker specific parameters in STAN for tuning the system

- **Different Settings:**
  - Message sizes:
    - 100 bytes(B) (0.1 kB)
    - 500 bytes(B) (0.5 kB)
    - 1000 bytes(B) (1 kB)
  - Number of messages (total data to send 7 GB):
    - 7 000 000 000 B / 100 B = 70 000 000 messages
    - 7 000 000 000 B / 500 B = 14 000 000 messages
    - 7 000 000 000 B / 1000 B = 7 000 000 messages

- GCE Machine types:
    - E2 high-CPU machine type (e2-highcpu-8) : High-CPU machine type with 8 vCPU and 8 GB of memory, and maximum Egress bandwidth 16 (Gbps).
    - E2 high-CPU machine type (e2-highcpu-16): High-CPU machine type with 16 vCPU and 16 GB of memory, and maximum Egress bandwidth 16 (Gbps).

Algorithm 1: Message queue testing procedure - Total number of test-runs = 486 tests

```
Input: Parameters: broker setup={1,3,5}, machine type={e2-highcpu-8, e2-highcpu-16},
       message queue={Kafka, RabbitMQ, Nats Streaming}, producer/consumer
       mode={1 prod/cons, 3 prod/cons, 5 prod/cons}, message setting={0.1 kB/msg,
       0.5 kB/msg, 1 kB/msg}
Output: Metrics for comparing the systems
/* 1, 3 or 5 brokers setup */
foreach broker setup b do
    /* One of the two VM machine types */
    foreach machine type m do
        /* For all message queues (Kafka, RabbitMQ, NATS Streaming) */
        foreach message queue q do
            /* Decides producer/consumer modes depending on nr of brokers */
            foreach producer/consumer set $s_i$ do
                /* Run each particular test 3 times for statistically significant data */
                foreach message setting $m_s$ do
                    for $i \leftarrow 1, 3$ do
                        Test system q with benchmarking tool publishing $m_s$ messages
                    end
                end
            end
        end
    end
end
```

6.2.3 Questions

- There are many parameters that affect the performance of the systems, what can be (if possible) done to reduce number of tests further (current nr tests = 486)?
- Should I use quorum queues over classic mirrored queues despite its bad performance?
    - Things to consider:
        - Classic Mirrored Queues will be removed in future RabbitMQ versions (quorum or classic queues remaining in future)
        - Cannot replicate using regular classic queues
- If we e.g. in a test run 3 producers and 3 consumers, then total 7 GB data are sent from the producers to the brokers. The total egress in this test will be 7 * 3 = 21 GB data. 1 GB costs 0.01 USD, in this example, totally 21 cents.
    - In the created test-model above, the total cost for only egress will approximately be 102 USD. Is this okay?