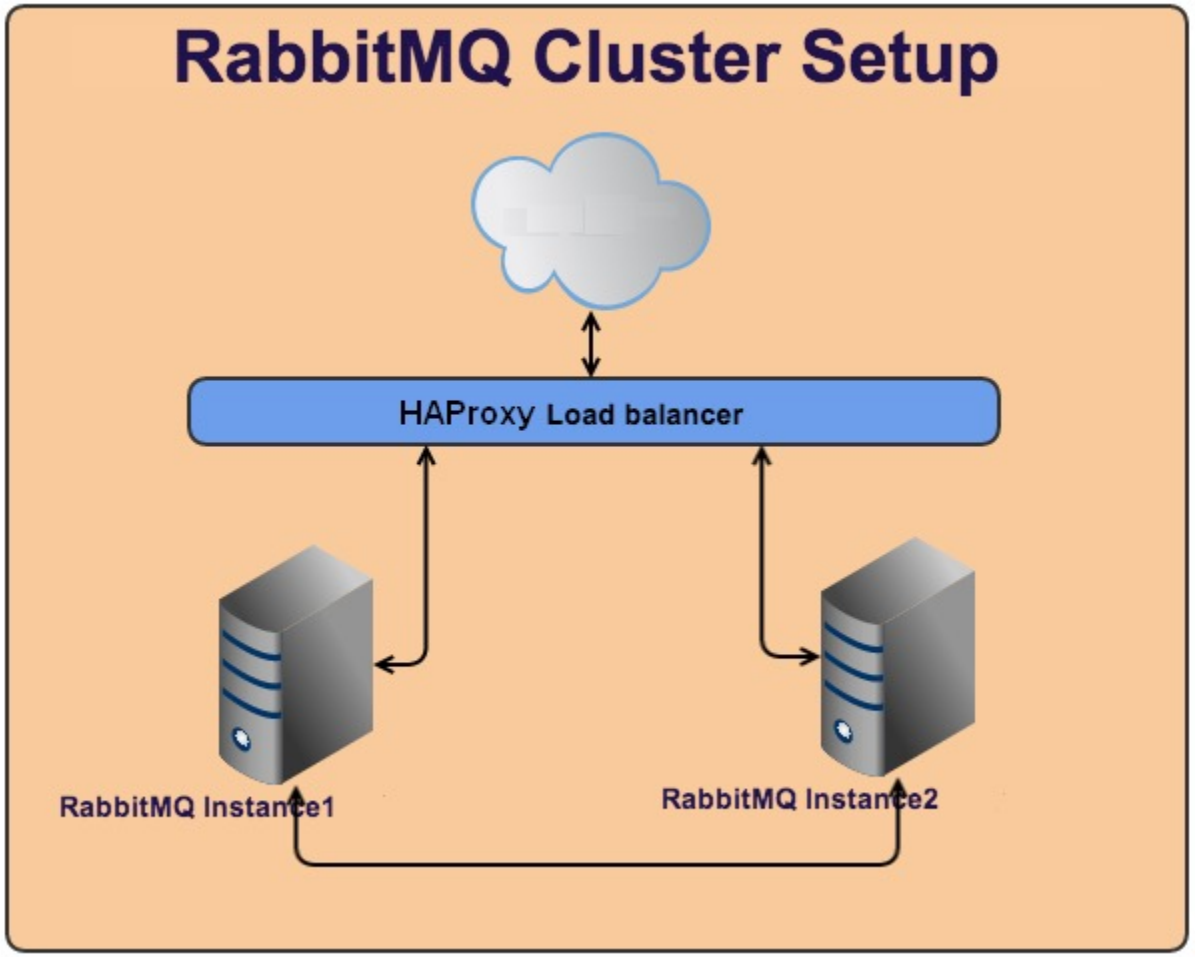
HA - Proxy

HA - Proxy


RabbitMQ Cluster Setup
HAProxy Load balancer
RabbitMQ Instance1
RabbitMQ Instance2

HP

HA - Proxy

- A free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications.
- Define a load balance in front of it and map the backend RabbitMQ instance.

HA - Proxy

- yum install haproxy

/etc/haproxy/haproxy.cfg

```
global
daemon

defaults
mode tcp
maxconn 10000
timeout connect 5s
timeout client 100s
timeout server 100s

listen rabbitmq 192.168.1.188:5670
mode tcp
balance roundrobin
server rabbitmaster 192.168.1.182:5672 check inter 5s rise 2 fall 3
server rabbitslave 192.168.1.185:5672 check inter 5s rise 2 fall 3
```

3 failed checks will put the server into the *DOWN* state:

2 successful health checks are needed before the server will return to the load-balancing rotation:

Performance

Introduction

- Very often the application needs to be optimized on the client side:
  - CPU-intensive applications can be optimized by running one thread for each CPU core
  - I/O-intensive applications can be optimized by running many threads per core in order to hide implicit latencies

HP

Multithreading and queues

- Using threads can help the application's performance.
- The queue is faster when empty and you should design your application, in order to always have a queue as empty as possible.
- The queue capacity comes in handy every time your application needs to deal with load spikes.
- By using queues, messages can eventually be buffered and handled without losing any information.
- If the consumer is slower than the producer and you have to consume the messages quickly, you can try to add more threads or more consumers.

HP

System tuning

- Set ***vm_memory_high_watermark*** configuration
- At fifty percent of vm_memory_high_watermark, RabbitMQ will start to move messages from memory to on-disk paging space.
- If neither this paging mechanism, nor the consumers are able to keep pace with the producers, the limit will be reached, and then RabbitMQ will block the producers.

System tuning

- Configure the watermark using

```
rabbitmqctl set_vm_memory_high_watermark 0.6
Or directly in the rabbitmq.config file using:
[{rabbit, [{vm_memory_high_watermark, 0.6}]}].
```

HP

Improving bandwidth

- Using noAck flag and managing the prefetch parameter is another client-side way to improve the performance and the bandwidth.
- Both noAck and prefetch are used by the consumers.

HP

Improving bandwidth

- Prefetch
  - To set the prefetch, use basicQos(prefetch_count)
  - The prefetch count is the maximum number of unacknowledged messages: a large value will let the client prefetch many messages in advance without waiting for the acks of the messages being processed.
  - the prefetch count can make the difference, especially when you have more consumers bound to the same queue.

HP

Improving bandwidth

- NoAck :
  - To set noAck, use basicConsume(Constants.queue, true).
  - The parameter is useful when you have a stream data or when it doesn't matter to send the acks manually.
- Optimizing messaging operations, by acting on application-side optimizations. This is feasible both for "small" and "large" messages:
  - If the size of your messages is too small, you can aggregate them manually before sending them and unpack them at the receiver side
  - If the size of the messages is too large, you can try to compress the message before sending it and decompress it at the consumer side

Flow Control

- Two flow control
  1. A per-connection credit flow - initially.
  2. A global mechanism - Memory-Based

- Credit Flow

reader -> channel -> queue process -> message store.

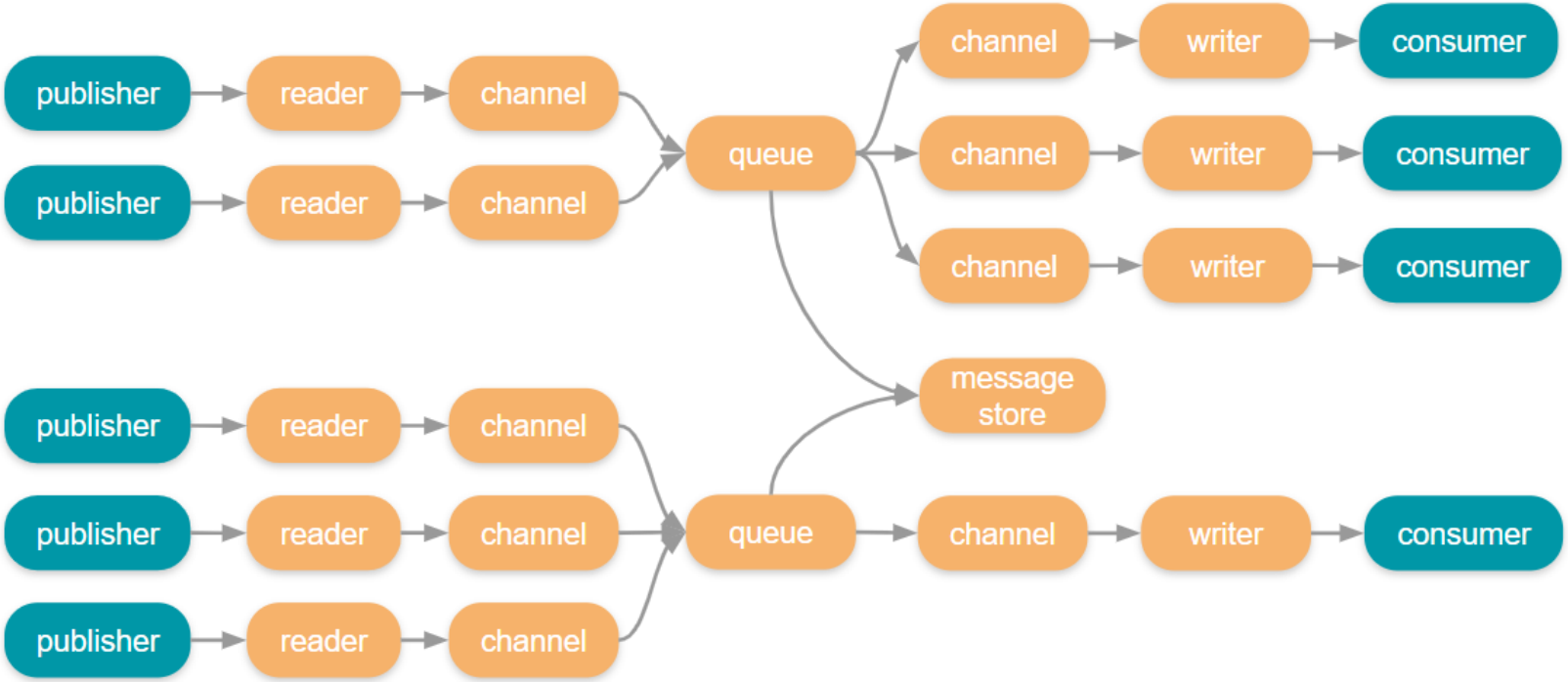
(credit_flow_default_credit under rabbitmq.config (200/50)

200 messages of initial credit, and after 50 messages processed by the receiving process, the process that sent the messages will be granted 50 more credits.

`msg_store_credit_disc_bound` (2000/500)

If the credit based flow control was unable to put the brakes on enough, or memory usage has grown to critical levels for another reason, memory alarms kick in as a last resort to protect the broker from crashing (or being killed by the OS) due to running out of memory.

Message flow




```
[root@rabbitmq0 /]# rabbitmqctl list_connections
Listing connections ...
user    peer_host       peer_port       state
guest   172.18.0.3      50720   running
[root@rabbitmq0 /]# rabbitmqctl list_connections
Listing connections ...
user    peer_host       peer_port       state
guest   172.18.0.3      50720   running
guest   172.18.0.2      58172   running
guest   172.18.0.2      58174   flow
[root@rabbitmq0 /]#
```

HP

Memory-Based Flow Control

1. Detects the total amount of RAM on startup
2. ***set_vm_memory_high_watermark*** *fraction* is executed.
3. Raises a memory alarm and blocks all connections.

   40%
4. Once the memory alarm has cleared normal service resumes.

```
[{rabbit, [{vm_memory_high_watermark, 0.4}]}].
```

Lab: Load Balancing- HA Proxy & Performance Tuning