Scale-out RabbitMQ cluster can improve performance while keeping high availability.

Agenda

- Background
- Goal / Actions
- Ops Side
  - RabbitMQ verification
  - OpenStack messaging inspection
  - Instance increasement load testing
- Dev Side
  - Nova’s messaging problem and solutions against it
  - Implementations
- Conclusion

About us

Ops Side

Dev Side

Mahito Ogura <m.ogura@ntt.com>

NTT Communications

Technology Development

- DevOps Engineer
- Distributed System Architecte
- OpenStack Contributor

Masahito Muroi <muroi.masahito@lab.ntt.co.jp>

NTT

Sofware Innovation Center

- Cloud Architecte
- Blazar PTL
- Congress core reviewer

Background

- About 3 years ago, We got a report that “Message Queue (MQ) became a bottleneck at 10K VMs and it is difficult to create VMs after that”
  - This issue becames topics at past OpenStack Summit & Ops Meetup
  - Solution are known, such as using Nova Cell, tuning MQs and dividing some components into different MQs
  - However, It is difficult for operators to run RabbitMQ clusters.
- NTT Communications is planning to expand its cloud environment
  - 1000+ nodes (as 10K+ VMs) per region.
  - We need to improve performance and operation on MQ

Goal / Actions

Goal:

- To run more than 1000+ nodes with a single RabbitMQ Cluster

Actions

- Research & Verify RabbitMQ’s performance / function
- Analyze messaging inside OpenStack at high load
- Verify RabbitMQ improvement method on OpenStack

Agenda

- Background
- Goal / Actions
- **Ops Side**
  - **RabbitMQ verification**
  - OpenStack messaging inspection
  - Instance increasement load testing
- Dev Side
  - Nova’s messaging problem and solutions against it
  - Implementations
- Conclusion

What is RabbitMQ ?

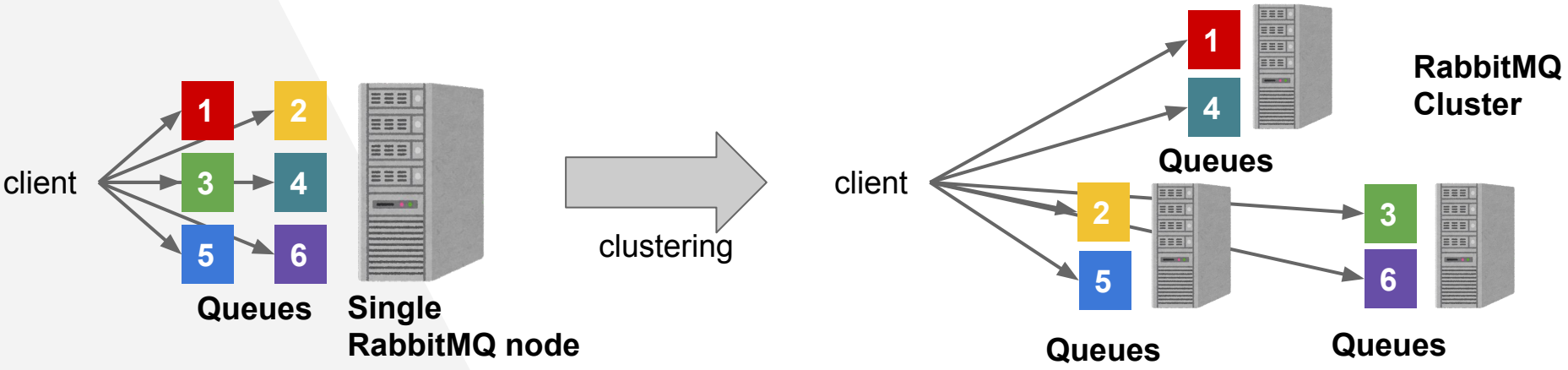
RabbitMQ is

- Widely deployed OSS message broker
- Using Advanced Message Queuing Protocol (a.k.a AMQP)
- Able to build a cluster for load balancing
- Able to use High Availability(HA) mode by mirroring queues [1]

[1] : https://www.rabbitmq.com/ha.html

RabbitMQ cluster can distribute queues

RabbitMQ cluster distributes queues for the Read/Write load distribution.

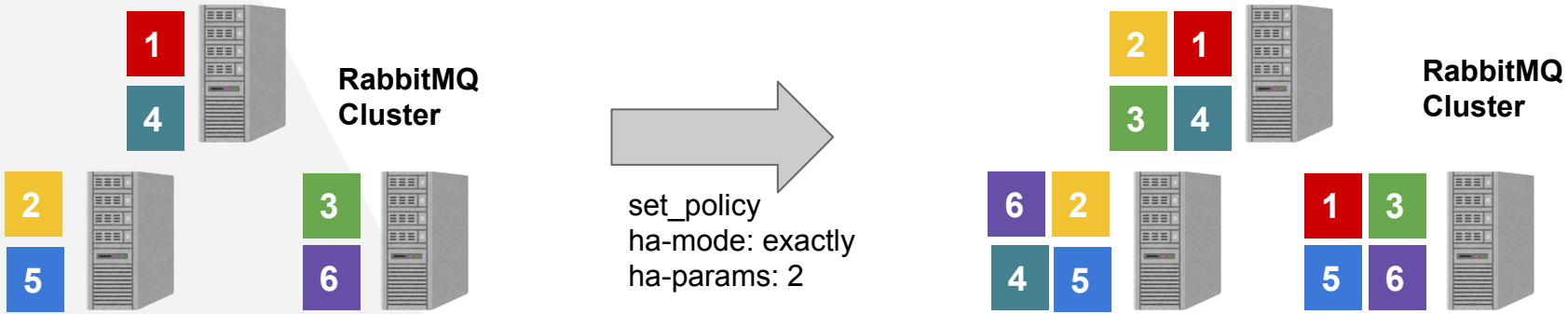


RabbitMQ cluster can make mirrored queues.

RabbitMQ cluster can be set to HA mode

- all: Queue is mirrored across all nodes in the cluster.
- exactly: Queue is mirrored to count nodes in the cluster.
- node: Queue is mirrored to the nodes listed in node names.

At this time, we set “HA = exactly” (mirror = 2) and verified whether it is possible to scale out while keeping HA.



“HA = exactly” can scale out

“HA = exactly” can improve performance by scale out [1]

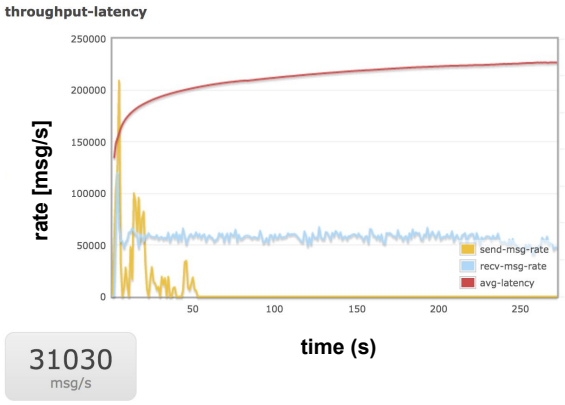


Fig. 1: Node=3

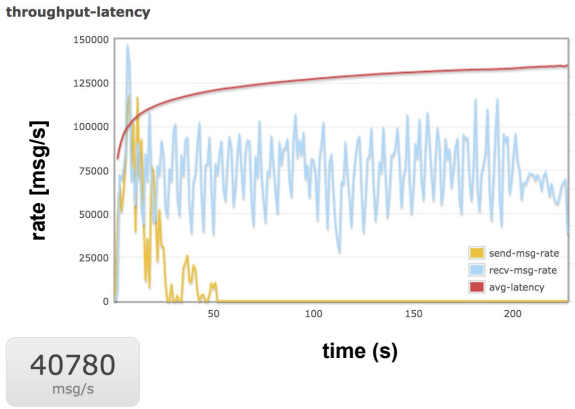


Fig. 2: Node=4

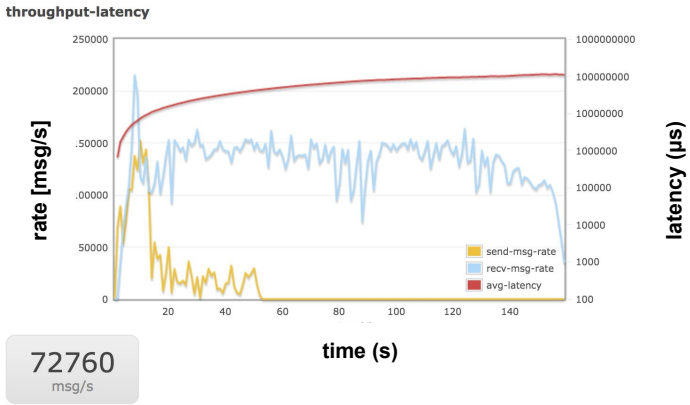


Fig.3: Node=5

[1]: We used HA-Proxy to avoid unbalanced access on specific node

“HA = all” can scale out too but …

“HA = all” can scale out but less than “HA = exactly”[1]

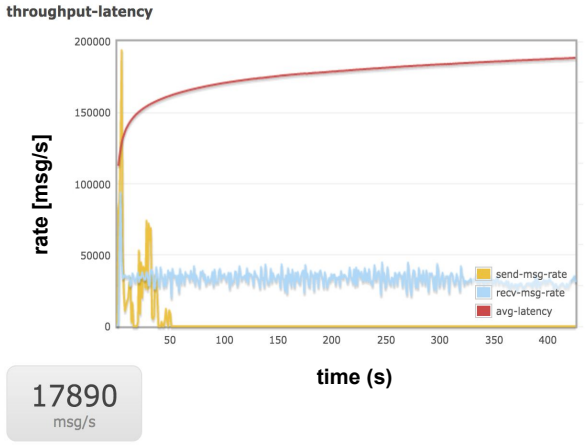


Fig. 1: Node=3

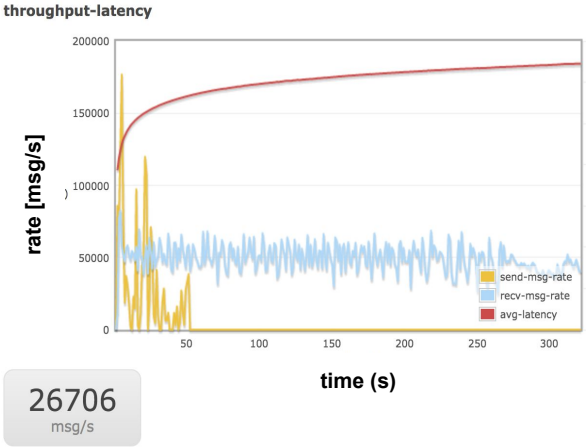


Fig. 2: Node=4

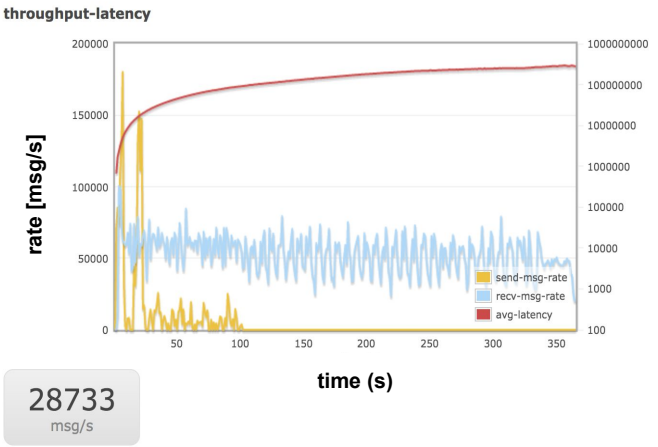


Fig. 3: Node=5

[1]: We used HA-Proxy to avoid unbalanced access on specific node

Results of RabbitMQ verification 1/2

- RabbitMQ cluster can Scale-out
  - “HA = exactly” scales performance better than “HA = all”
- RabbitMQ behavior when adding node to cluster
  - A new queue may be placed in a new node
  - Existing queues don’t rebalance and aren’t mirrored in the new node
    - Rebalance requires manual operation or cluster rebuild
    - Rebalancing large capacity queues may cause high load
  - RabbitMQ clients need config update & restart
    - when using Load Balancer, you have to update config to access the new node

Results of RabbitMQ verification 2/2

- Behavior when removing node from cluster
  - Existing queues are mirrored properly and no queue is lost
  - Error increases for a while on the client side
    - Clients should consider error handling
- Behavior at failure
  - Similar to “removing node from cluster”
    - Existing queues are rebalanced because of node decrease
    - Rebalance didn't occur even after node are added for recovery
      - Rebalance requires manual rebalance or cluster rebuild

Agenda

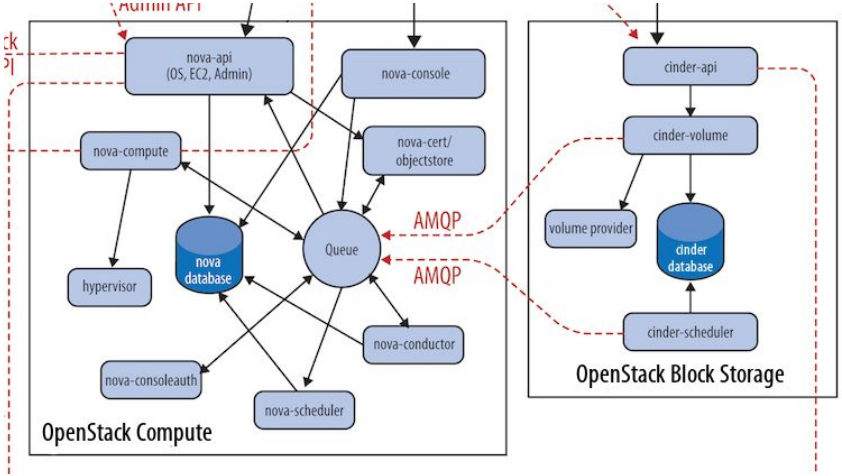
- Background
- Goal / Actions
- **Ops Side**
  - RabbitMQ verification
  - **OpenStack messaging inspection**
  - Instance increasement load testing
- Dev Side
  - Nova’s messaging problem and solutions against it
  - Implementation
- Conclusion

OpenStack messaging inspection

OpenStack uses messaging via MQ

Messaging patterns

- API call
  - User Request
  - Call from Periodic task
  - etc…
- Periodic task
  - Status check/update
  - Heal instance info cache
  - etc...



Messaging between Nova and Cinder

Messaging Kinds in OpenStack

3 Messaging Kinds used in OpenStack

Messaging Kinds	Messaging Targets	Reply
cast	1 : 1	No
call	1 : 1	Yes
fanout	1 : n	No

Kinds of “queue name” that OpenStack creates

- “ServiceName”
- “ServiceName_fanout_<<uuid>>” for fanout
- “ServiceName.HostName”
- ”reply_<<uuid>>” for call reply

Pattern 1: API call messaging

User Request or call from periodic task using API

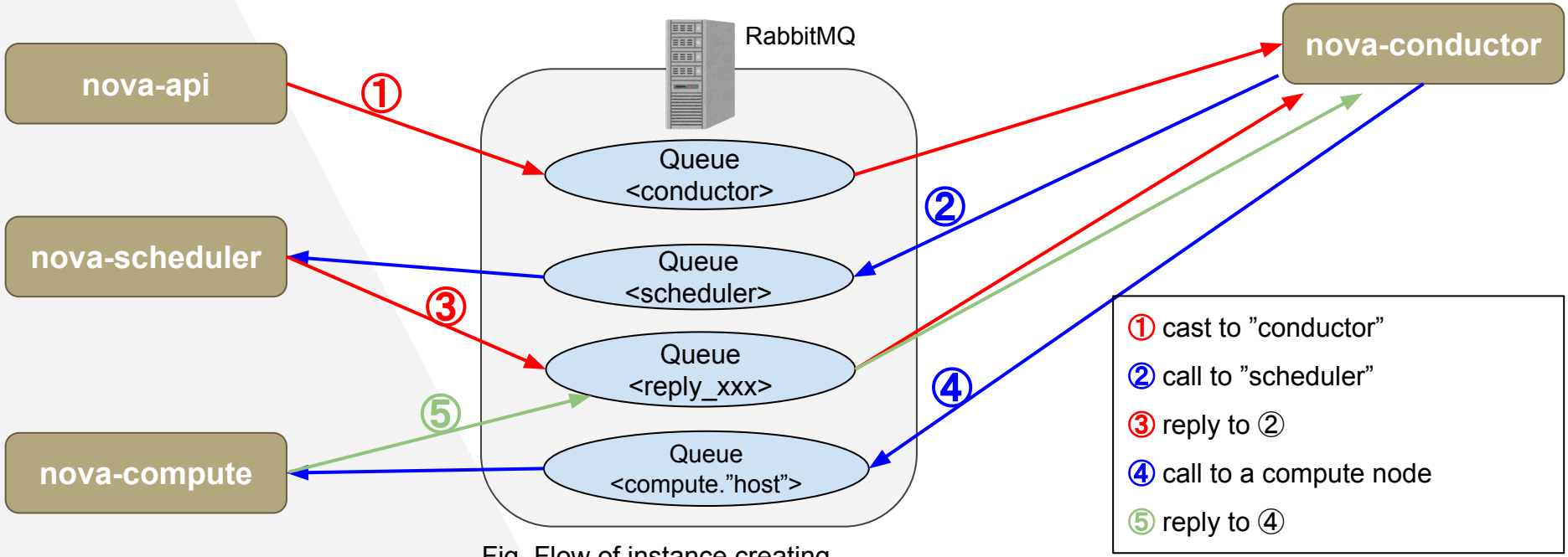


Fig. Flow of instance creating

Pattern 2: Periodic task messaging

There are tasks that are executed periodically in OpenStack service. Some of them send messages to other service by RPC (Remote Procedure Call) or API

The number of periodic tasks in main components

- Nova: 24（Messaging 23, non-Messaging 1）
- Neutron: 2（Messaging 2）
- Cinder: 2（Messaging 1，non-Messaging 1）
- Glance: None
- Keystone: None

Pattern 2: Periodic task messaging

Peiodic task regulary executes API calls and / or messaging to other service.

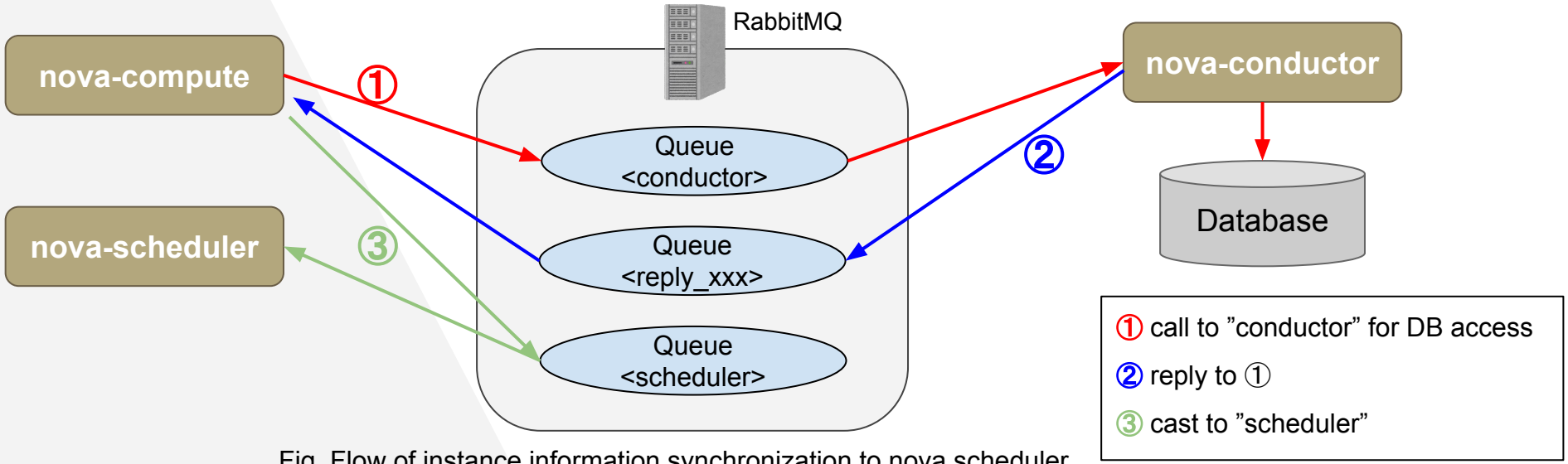


Fig. Flow of instance information synchronization to nova scheduler.

Results of OpenStack messaging inspection

- OpenStack has 2 messaging patterns, "API call" and "Periodic Task". The two uses a common set of queues.
- The **number of queues** does not increase, unless nodes or services are added.
- The **amount of messages** depends on the number of API calls from users, the number of resources such as instances, block devices and virtual routers, and the number of service nodes.
- The **message size** depends on the number of resources such as instances, block devices and virtual routers.

Agenda

- Background
- Goal / Actions
- **Ops Side**
  - RabbitMQ verification
  - OpenStack messaging inspection
  - **Instance increasement load testing**
- Dev Side
  - Nova’s messaging problem and solutions against it
  - Implementation
- Conclusion

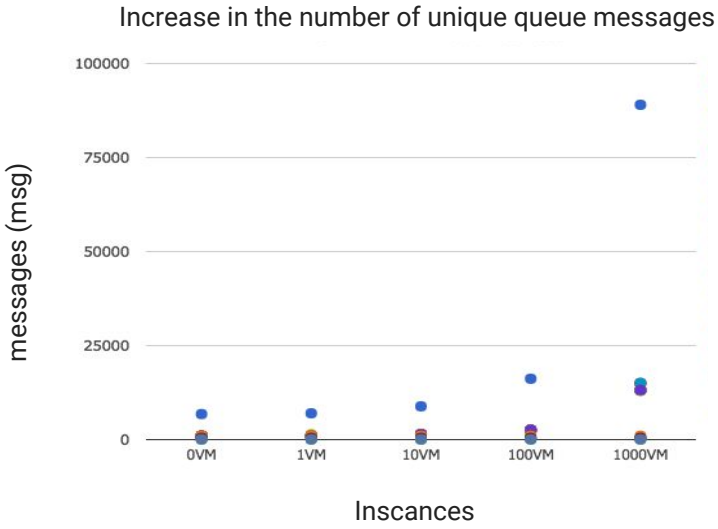
Environment

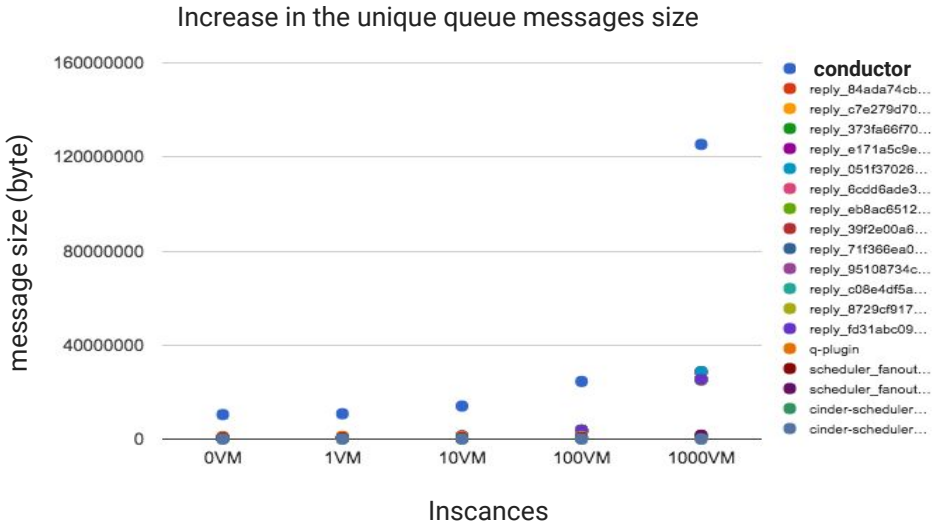
We created 1 to 1,000 instances to measure the messaging count and message size of each queue in test environment.

- nova-compute : 13 node
- RabbitMQ v3.2.4 : 3 node (HA : ALL)

Number of Messages & Message size

The number of messages & message size (in one hour) increase when number of instances increases



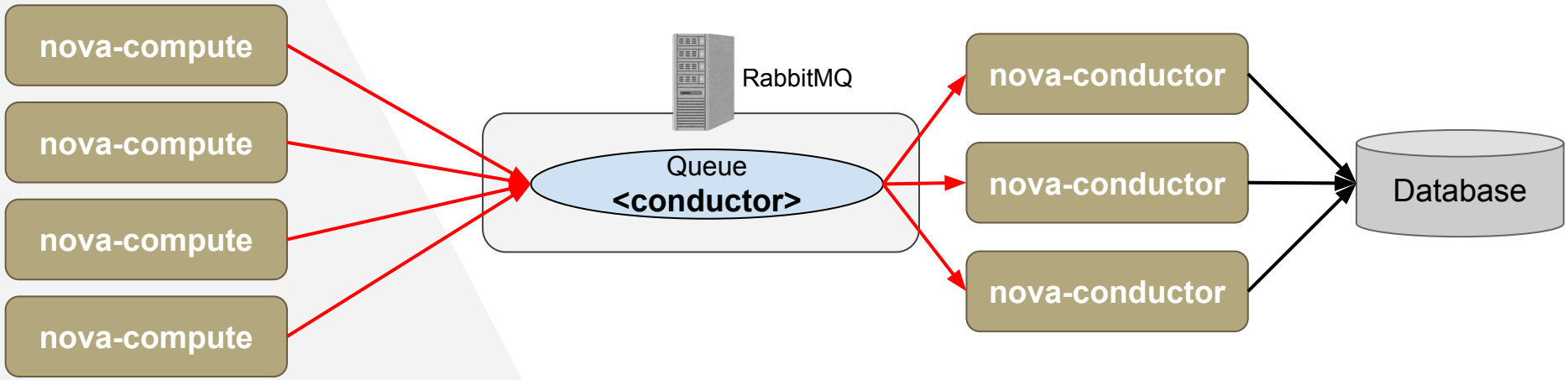


Agenda

- Background
- Goal / Actions
- Ops Side
  - RabbitMQ verification
  - OpenStack messaging inspection
  - Instance increasement load testing
- **Dev Side**
  - Nova’s messaging problem and solutions against it
  - Implementation
- Conclusion

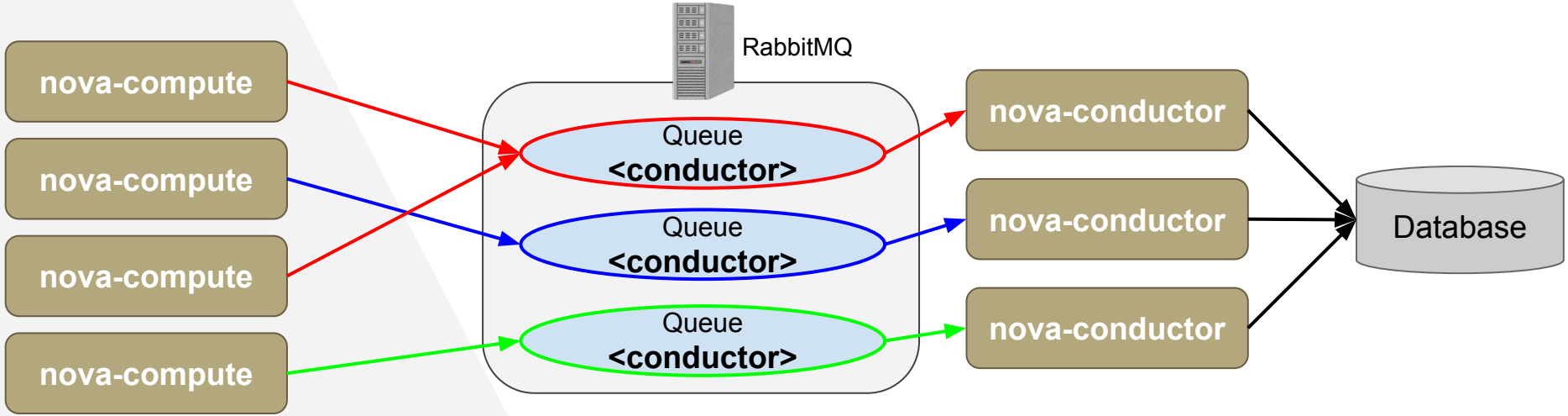
Problems in Nova's messaging

- heavily concentrated in 'conductor' queue
  - Becaues of scale-out reason of nova-conductor and nova-compute



Basic ideas against the problem

- Dividing the messaging workloads into multi queues
  - connected between nova-conductors and nova-computes with mesh architecture

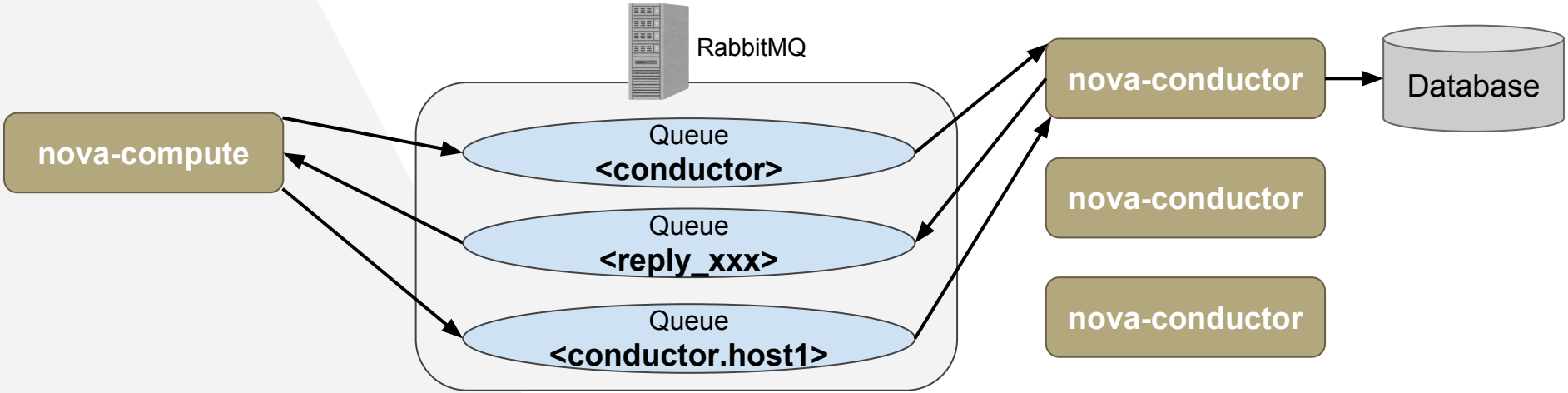


Agenda

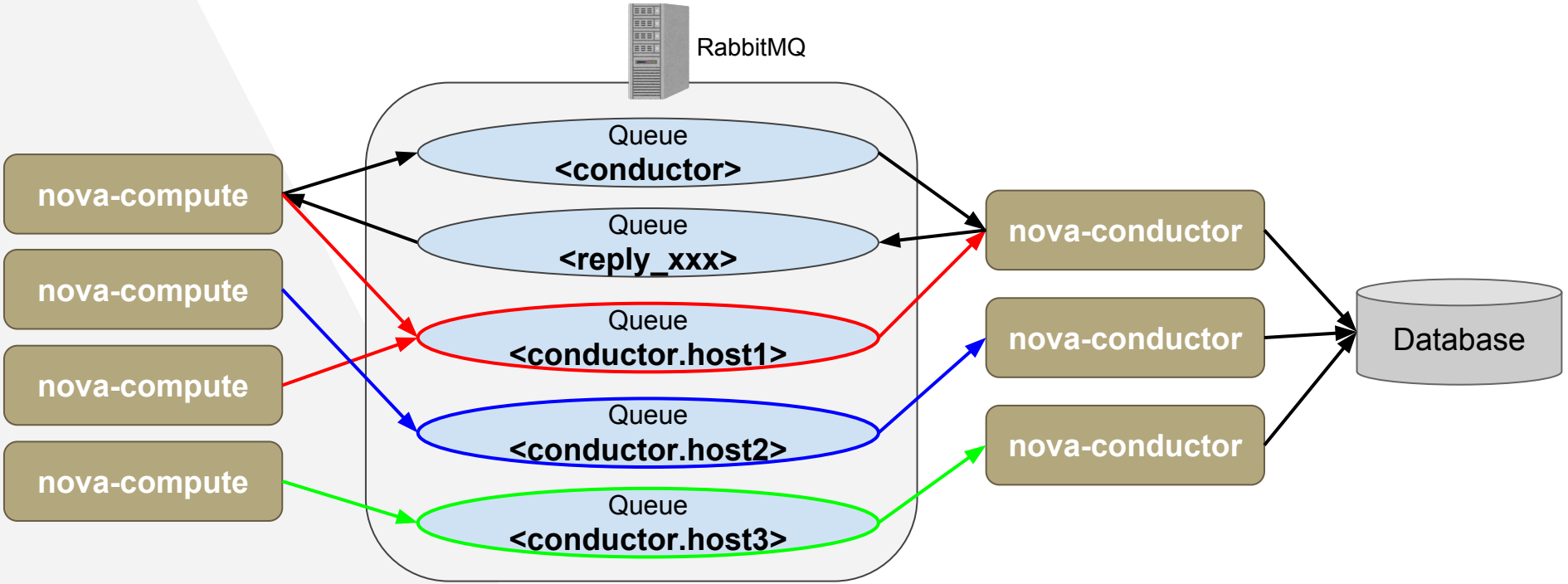
- Background
- Goal / Actions
- Ops Side
  - RabbitMQ verification
  - OpenStack messaging inspection
  - Instance increasement load testing
- **Dev Side**
  - Nova's messaging problem and solutions against it
  - Implementation
- Conclusion

Implementation

- Each nova-compute locally selects a leader nova-conductor
  - Basing on a receiver of a previous RPC
  - calling all RPC to the leader nova-conductor directly



Implementation



Advantages of the locally selected leader RPC

- Non central manager
- Automatic Load Balancing among nova-conductors
- Automatic rebalancing in case of down in nova-conductor

Agenda

- Background
- Goal / Actions
- Ops Side
  - RabbitMQ verification
  - OpenStack messaging inspection
  - Instance increasement load testing
- Dev Side
  - Nova’s messaging problem and solutions against it
  - Implementation
- **Conclusion**

Conclusion 1/2

- Scale-out RabbitMQ cluster can improve performance while keeping high availability
  - Load balancing to distribute Read / Write to queues within a cluster
  - When the performance limit for a specific queue is reached, it is necessary to respond with scale-up
- The number of messages & message size increase when instances & services increase.
  - Especially the messages which nova-compute throw to nova-conductor due to DB access, will increase fast.

Conclusion 2/2

- The stuck of messages in ‘conductor’ queue can be resolved by changing messaging style.
  - sample codes:
    https://github.com/muroi/nova/commits/conductor-message-bus
  - Not yet proposed to Nova community
- There is not much information on “HA mode exactly”
  - It is necessary to confirm that there are no problems due to large-scale and long-term stabilization tests in the future

Thank you for your attention.

This presantation is provided by

Agenda

- Background
- Goal / Actions
- Ops Side
- Dev Side
- Conclusion
- **Reference**

Reference : OpenStack Summit Barcelona 1/2

- RabbitMQ at Scale, Lessons Learned
  - Cisco reported that they have run 800+ compute nodes by using single RabbitMQ cluster
    - 3 RabbitMQ nodes and HA-mode is ALL
    - It is possible to run 800+ compute nodes by tuning Linux Kernel, Erlang VM and RabbitMQ

Reference : OpenStack Summit Barcelona 2/2

- Chasing 1000 Nodes Scale
  - Default number of API/RPC workers in OpenStack services wouldn't work for us if it tightened up to number of cores
  - When CPU and RAM are set enough in OpenStack services
    - MySQL and RAbbitMQ aren't bottoleneck at all.
    - Scheduler performance/scalability issues

Thank you!!