Exploring the scalability of RPC with oslo.messaging
- Request-response
- Each request goes to one of the servers
- Worker includes a client and a server
- Clients make requests concurrently
- Clients record throughput and latency
- Test fails if response not received within 2 seconds of request
Disclaimer

There are lots of aspects of scale and lots of different use cases or variations that could be explored.

This is an initial experiment that I hope provides some food for thought. It most certainly should not be considered comprehensive or conclusive.
Code for test: https://github.com/grs/ombt
• 2 machines for servers, both with 4 cores, both running Fedora 19
• 4 machines for clients, 2 with 12 cores, 2 with 16 cores, all running RHEL7
• RabbitMQ 3.1.5
• Qpidd 0.28
• Qpid Dispatch 0.2 (with patch)
• Oslo.messaging from https://github.com/FlaPer87/oslo.messaging/tree/gordon/
This graph shows how the average rate of requests – on the vertical axis - varies as we increase the number of workers for different configurations – i.e. As we move right along the horizontal axis.

The cut-off in each case is the point at which the response time of any request is above 2 seconds.
The next graph just focuses in on a smaller 'slice' of the horizontal axis...
The point at which we start on the vertical axis is the maximum request-response rate and is latency sensitive.

As we increase the workers, the rate of each client drops off.

The rate of degradation and the point at it starts are key measures of scalability.
For the two configurations using the AMQP 1.0 driver, there is minimal degradation until we get to about 20 workers.

For the rabbit driver, the degradation is more immediate.

For the qpid driver, degradation is somewhere in between that for the other two drivers, but we start significantly lower. Why?
The configuration these two lines represent are using the exact **same broker**.

The only thing that is different is the driver (and the client library it uses).

In fact the cutoff first observed for the qpid driver was due to this extra latency. On the same machine as the broker, with reduced latency but also reduced CPU available, the cutoff happens much later. Increasing the timeout marginally shows a more accurate picture of degradation from a bad starting point. (The same increase doesn't affect the cutoff for the other drivers).

The qpid driver has very poor latency due to extra (unnecessary) synchronous roundtrips arising from:

(a) creating a sender for every message (and querying the node type in each case)

(b) using a synchronous send (both for request and response)
This graph shows the growth in aggregate request rate – i.e. The overall rate of requests through the broker from all the clients together.

Initially the aggregate rate increases as we add workers.

Eventually we reach a point beyond which the rate cannot increase. Additional workers then tends to reduce the overall rate.

The maximum aggregate rate and the point at which it is reached are also key aspects of scalability.
Again, we will focus in on a smaller 'slice' of the horizontal axis to better see some of the detail...
The configurations using the AMQP 1.0 driver, increase fairly linearly with the number of workers up to about 20 workers, tailing off after 30 workers or so. The maximum aggregate rate is significantly higher than either of the other two drivers.

The rabbit driver shows a linear increase in aggregate rate up to about 6 or 7 workers, and flattens out at about 10 workers.

The qpid driver shows much slower growth than the other drivers, but the increase continues to about 30 workers.
What are the limits, and can we get round them?
1. limited number of workers we can support at the maximum rate of requests

2. limited achievable aggregate rate

3. limited number of workers we can support while staying within the defined maximum response time
1. Can we delay the point at which the average rate seen by each worker begins to degrade?

2. Can we keep increasing the aggregate rate?

3. Can we extend the scale at which we can keep within a given maximum timeout?
Need to go beyond the limits of a single intermediating process
The point of significant degradation was postponed with the AMQP 1.0 driver and a Qpid Dispatch Router pair, though the line is not quite flat.

With the rabbit driver an A RabbitMQ cluster pair the curve was shifted right a little, but there was no alteration in basic shape.
Extended the maximum achievable aggregate rate both with rabbit driver and clustered RabbitMQ and for AMQP 1.0 driver and Qpid Dispatch Router network.
Extended the number of workers we can support while staying within the maximum allowed response time both with rabbit driver and clustered RabbitMQ and for AMQP 1.0 driver and Qpid Dispatch Router network.