УДК 004.9

СРАВНИТЕЛЬНЫЙ АНАЛИЗ БРОКЕРОВ СООБЩЕНИЙ

Тойбеков Айдын Темирболатұлы
aidyn_work@mail.ru
ЕНУ им. Л.Н.Гумилева, Нур-Султан, Казахстан
Научный руководитель – Т.К. Жукабаева

В работе рассмотрен современные брокеры сообщений, которые можно разделить на две группы: безброкерные и брокерные. Безброкерные очереди сообщений являются одноранговыми, так что посредник не участвует в передаче сообщений, в то время как брокерные очереди имеют какой-то сервер между конечными точками. Системы, которые мы будем анализировать, это: безброкерные (Nanomsg, ZeroMQ), брокерные (ActiveMQ, NATS, Kafka, Kestrel, NSQ, RabbitMQ, Redis, ruby-nats).

1.1. Анализ пропускной способности

Здесь мы наблюдаем пропускную способность «отправителя» и пропускную способность «приемника», то есть количество сообщений, которые могут быть отправлены в секунду, и количество сообщений, которые могут быть получены в секунду.

Этот тест был выполнен путем отправки 1 000 000 сообщений размером на 1 КБ и измерения времени для отправки и получения с каждой стороны. Все тесты проводились на MacBook Pro 2,6 ГГц i7, с 16 ГБ оперативной памяти, то есть на одном и том же персональном компьютере.

Многие тесты производительности обычно используют более мелкие сообщения в диапазоне от 100 до 500 байтов. Выбрано 1 КБ, потому что он более репрезентативен тем, что вы можете видеть в производственной среде, хотя это зависит от конкретного случая. Для ориентированных на сообщения систем промежуточного программного обеспечения использовался только один брокер. В большинстве случаев кластерная среда дает гораздо лучшие результаты.

Неудивительно, что на стороне отправки есть более высокая пропускная способность. В ZeroMQ большая разница в количестве сообщений отправителя и получателя. Он способен отправлять более пяти миллионов сообщений в секунду, но может получать только 600 000

сообщений в секунду. Напротив, nanomsg посылает три миллиона сообщений в секунду, но может получить почти два миллиона(рис.1).

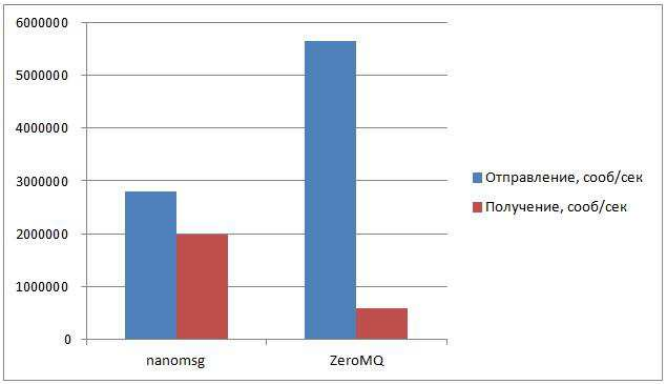

Рисунок 1. График пропускной способности безброкерных очередей

Касательно брокерных очередов сообщений мы наблюдаем, что брокерские очереди сообщений имеют значительно меньшую пропускную способность, чем их безброкерные аналоги на пару порядков. Половина брокерских очередей имеет пропускную способность ниже 25 000 сообщений в секунду.

Показатели для Redis могут быть вменяиемым. Несмотря на предоставление возможностей *публикация / подписки*, он не предназначен для работы в качестве надежной очереди сообщений. Так как весь набор данных всегда находится в оперативной памяти, это может привести к нехватке или фрагментации памяти. В момент, когда на серверах Redis появляется дисковая активность, время ответа растет, наблюдается ухудшение производительности. Если снизить частоту сохранения данных на диск, тогда возникает опасность потерять данных. Таким образом, мы не рассматриваем его в качестве кандидата.

Kafka, NATs и ruby-nats имеют аналогичные характеристики как Redis, но они смогли надежно обрабатывать объем сообщений без прерывистых сбоев(рис.2).

А брокеры сообщении ActiveMQ, Kestrel, NSQ и RabbitMQ показали низкую пропускную способность, который не удовлетворяет нашим требованием(рис.2).

Есть еще один важный момент, брокерские очереди имеют довольно равномерную пропускную способность. В отличие от безброкерных библиотек, брокерских очередях объем посылаемых в секунду и объем принимаемых в секунду сообщений близки друг к другу.

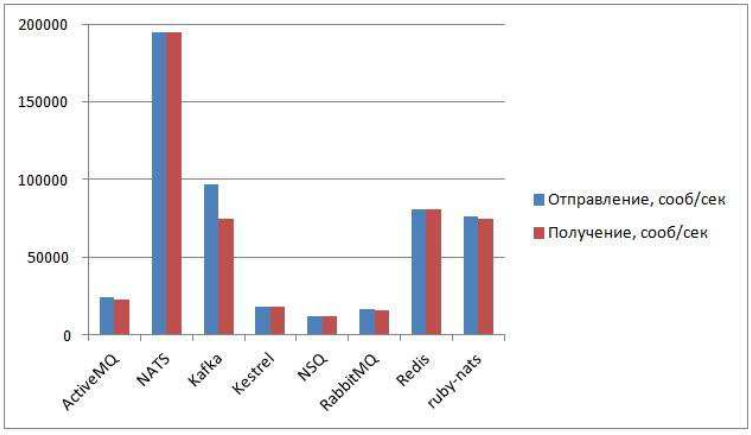

Рисунок 2. График пропускной способности брокеров сообщении

1.2. Тесты на латентность

Реальность такова, что латентность сообщения, отправленного по сети, не является однородной. Он может сильно различаться для каждого сообщения. В отличие от пропускной способности, латентность не измеряется у отправителя или получателя, а скорее в целом. Но поскольку каждое сообщение имеет свою собственную задержку, мы рассмотрим средние из них. Средняя латентность сообщения колеблется в зависимости от количества отправленных сообщений.

В случае безброкерных систем (ZeroMQ, nanomsg), чем больше сообщений отправляется через систему, латентность каждого сообщения увеличивается. Латентность до 500 000 сообщении резко возрастает, а по мере приближения к 1 000 000 сообщений, увеличивается с меньшей скоростью (рис.3). Другим особенностям является первоначальный всплеск латентности между 1000 и 5000 сообщениями, что более выражено с помощью nanomsg. Трудно определить причину, но эти изменения могут указывать на то, как в каждой библиотеке реализованы пакетная обработка сообщений и другие оптимизации обхода сетевого стека.

Брокерные очереди показывает аналогичные характеристики, а также следующие некоторые особенности. Когда количество сообщении доходит до 5000, Redis показывает скачок значении латентности до 1/100 (миллисекунд/сообщения), после этого задержка становится практически постоянным сразу после 5000 сообщений (рис. 4).

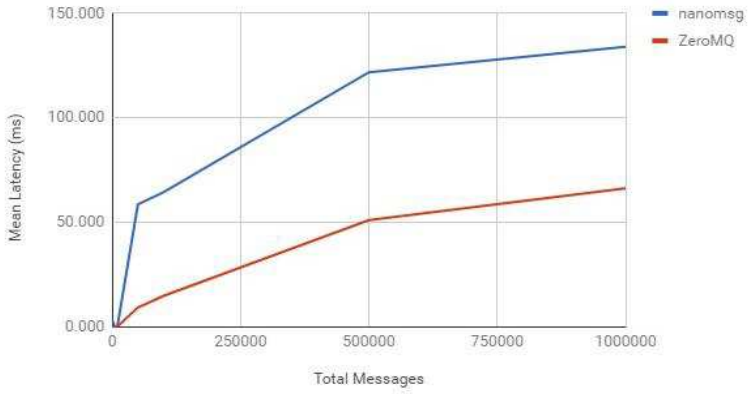

Рисунок 3. График латентности безброкерных очередей

NSQ не проявляет такого же всплеска в латентности и ведет себя, более или менее, линейно. Задержка остальных на порядок выше, чем в других брокерных очередях сообщений(рис. 4).

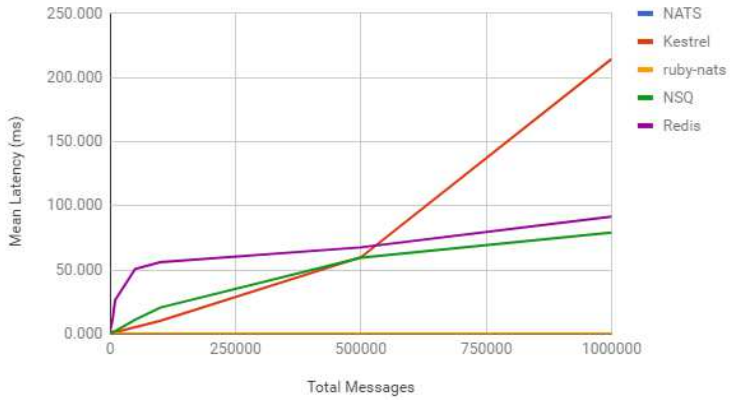

Рисунок 4. График латентности брокеров сообщении 1

Задержка RabbitMQ является постоянной, а ActiveMQ и Kafka являются линейными(рис. 5). А ruby-nats и NATS практически не регистрируются на графике. Они

демонстрировали удивительно низкие задержки и неожиданные отношения с количеством сообщений(рис. 6).

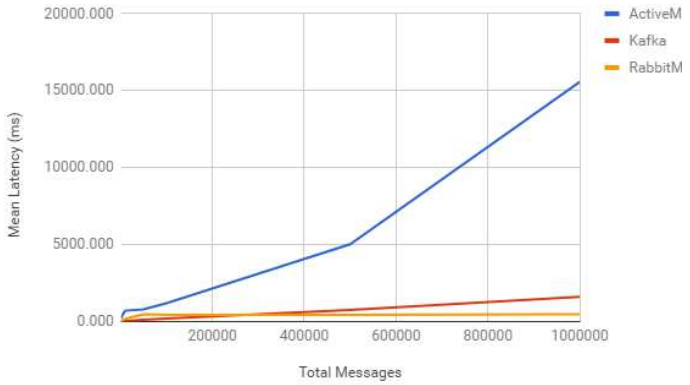

Рисунок 5. График латентности брокеров сообщении 2

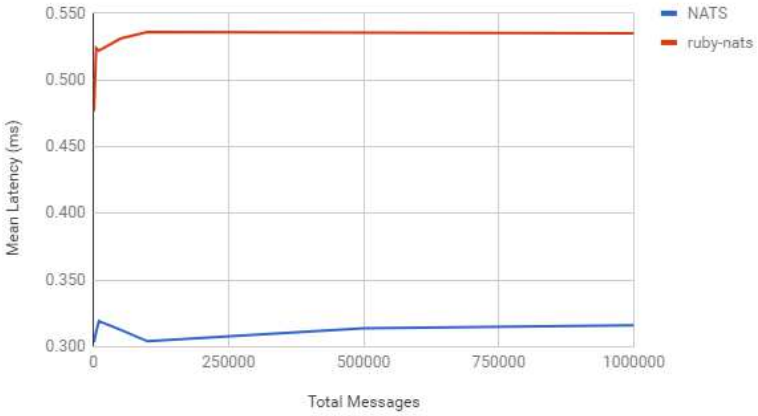

Рисунок 6. График латентности брокеров сообщении 3

В ходе работы был проведен сравнительный анализ брокеров сообщений. Приведен подробный анализ пропускной способности, проведены тесты на латентность, а также качественный анализ брокеров сообщений. На основе проведенных исследований для дальнейшей работы выбран Kafka. Первоначально разработанный LinkedIn, Kafka реализует обмен сообщениями-подписками через распределенный журнал фиксации. Он предназначен для работы в качестве кластера, который может потребляться большим количеством клиентов. Горизонтальное масштабирование выполняется без особых усилий с помощью ZooKeeper, чтобы можно было легко добавлять дополнительных потребителей и брокеров. Он также прозрачно заботится о перебалансировке кластера. Kafka использует постоянный журнал фиксации для хранения сообщений в брокере. В отличие от других надежных очередей, которые обычно удаляют сохраняющиеся сообщения при потреблении, Kafka сохраняет их в течение определенного периода времени. Это означает, что сообщения могут быть «воспроизведены» в случае сбоя потребителя. Кафка, является более «универсальным» решением, среди этих системы. Рассмотрев все преимущества и недостатки остальных систем было обнаружено, что Kafka обладает наилучшим набором свойств для выполнения поставленной задачи, в результате чего она была выбрана в качестве системы переноса больших данных.

Список использованных источников

1. Линев Ф. А. Обзор систем обмена сообщениями // Молодой ученый. — 2017. — №19. — С. 29-32. — URL https://moluch.ru/archive/153/43351/ (дата обращения: 02.04.2018).
2. Сервисы AWS для обмена сообщениями. URL https://aws.amazon.com/ru/messaging/ (дата обращения: 02.04.2018).
3. Сервисы AWS для обмена сообщениями