УДК 004.62

ПРИНЦИПЫ И ПРИЕМЫ ОБРАБОТКИ ОЧЕРЕДЕЙ

Ю.В. ЛАПТЕВ
(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается понятие очередей, их применение в высоконагруженных проектах и принципы обработки. Показаны области применения очередей, инфраструктура обработки очередей.

Очередью в программировании называется упорядоченный набор элементов, которые могут удаляться с ее начала и помещаться в ее конец. Очередь в программировании используется, как и в реальной жизни, когда нужно последовательно совершить какие-то действия в порядке их поступления. Примером может служить организация событий в Windows. Когда пользователь оказывает какое-то действие на приложение, то в приложении не вызывается соответствующая процедура (в этот момент приложение может совершать другие действия), а ему присылается сообщение, содержащее информацию о совершенном действии, это сообщение ставится в очередь, и только когда будут обработаны сообщения, пришедшие ранее, приложение выполнит необходимое действие [1].

Области применения очередей

Самый распространенный способ применения – для асинхронного выполнения операций, порождаемых в процессе обработки запросов пользователя, результаты исполнения которых не требуются немедленно для осуществления обратной связи. Например, отправка почты и уведомлений, предложений дружбы, обновление поисковых индексов, сброс кэшей, постобработка данных, полученных от пользователя (перекодирование видео, нарезка фотографий), отправка данных подсистемам, время реакции/надежность которых не гарантируется. Обладая удобной инфраструктурой для распределенной обработки заданий, спектр применения очередей является еще больше:

- *отложенная обработка действий пользователя*. Не лучшая идея заставлять пользователя ждать, пока будут сохранены его данные в SQL базу или другое хранилище (а зачастую необходимо вносить изменения сразу в несколько систем). Иногда данные могут вовсе не попасть в хранилище, если клиент оборвал соединение, не дождавшись ответа. Хорошей практикой является добавление события о пользовательских действиях в достаточно быстрый сервер очередей, после чего можно отвечать клиенту, что операция прошла успешно. Всю остальную работу надежно и эффективно выполнят обработчики очереди;

- *рассылка сообщений, писем и похожие операции*. Если необходимо отослать большое количество данных, при этом не перегрузив хранилища всплеском запросов, то это делается с помощь варьирования количества обработчиков очереди, размазывая пиковую нагрузку до разумного уровня, чтобы время обработки клиентских запросов не ухудшилось. И самое главное – с помощью очередей легко избежать дублирования сообщений. Крайне неприятно получить два письма об одном и том же событии;

- *транспорт для «надежной» статистики*. Передача важных (все, что связано с деньгами) данных на агрегаторы статистики. Системы агрегации статистики обычно требовательны к ресурсам процессора, и при обработке данных могут не обеспечивать необходимое для frontend-серверов время ответа. Еще одна особенность подобных систем – неравномерная загрузка, обычно связанная с обработкой данных порциями. Передача статистики через серверы очередей позволит избежать проблем с нестабильной задержкой и при этом сохранит гарантию доставки;

- *группировка событий*. Если группа событий будет обращаться к одному и тому же набору данных в других системах, имеет смысл ставить одинаковое время активации. Физический смысл ухищрений в более эффективном использовании кэшей хранилища, в которое пойдут запросы из обработчиков сообщений;

- *каскадные очереди*. Организация конечного автомата из нескольких очередей путем перекладывания данных между очередями по завершении очередного этапа обработки. Часто необходимо в процессе обработки сообщения выполнить ряд действий, сильно различающихся по необходимым ресурсам. В таком случае разнесение «быстрых» и «медленных» действий по разным этапам позволяет эффективно управлять необходимым количеством ресурсов, варьируя число обработчиков для каждой очереди [2].

Инфраструктура обработки очередей

По мере увеличения количества типов очередей уследить за возрастающим количеством скриптов и демонов (cron) становится все сложнее. Нужно следить за нагрузкой, запускать достаточное количество процессов, не забывать мониторить их, а при выпадении серверов своевременно поднимать процессы на других машинах. Для решения всех этих проблем нужна инфраструктура, которой нужно предоставить

кластер серверов-обработчиков и серверы очередей, а она сама уже будет управлять всем этим. При проектировании инфраструктуры для обработки очередей необходимо учитывать следующие характеристики:

- **распределенность** – задания одного типа могут выполняться на разных серверах;
- **отказоустойчивость** – выпадение 10-20 процентов серверов кластера не должно приводить к деградации важных очередей;
- **гомогенность** – отсутствие какой-либо специализации серверов, из-за чего любое задание может выполняться на любом сервере;
- **приоритизация** – есть критичные очереди, а есть те, которые могут и подождать;
- **автоматизация** – минимум ручного вмешательства: балансировка – перераспределение обработчиков каждого типа в зависимости от потребностей; адаптация к серверу – определение количества обработчиков на каждом сервере в зависимости от его характеристик, отслеживание потребления памяти и процессора;
- **пакетная обработка** – задания одного типа должны группироваться для возможности применения оптимизаций;
- **статистика** – сбор данных о поведении очередей, потоке заданий, количестве, наличии ошибок;
- **мониторинг** – оповещение о нештатных ситуациях и изменении стандартного поведения очередей.

Схема инфраструктуры обработки очередей представлена на рисунке 1.

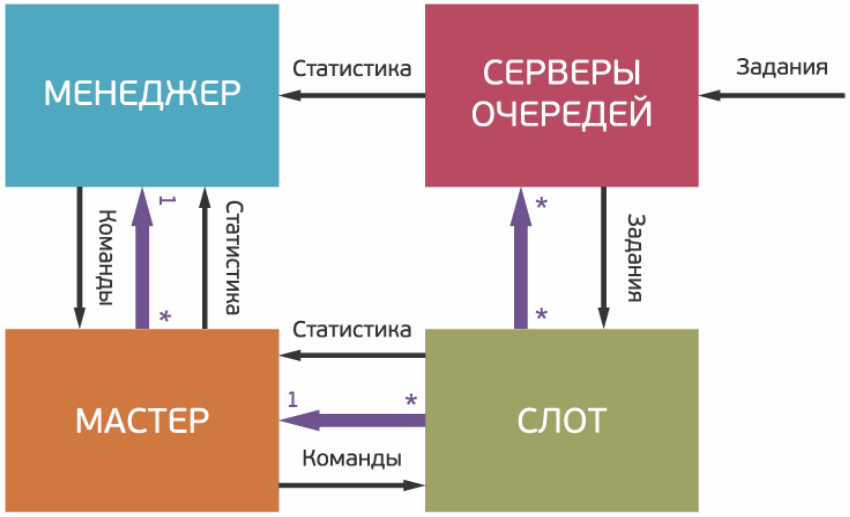

Рисунок 1. – Схема инфраструктуры обработки очередей

На данной схеме можно выделить 3 базовые сущности: слот, мастер и менеджер.

Слот – это процесс-обработчик очереди (воркер). Слотов на каждом сервере может быть запущено множество, каждый слот характеризуется типом очереди, которую он в данный момент обрабатывает. Специализация слотов на очередях определенных типов повышает локальность соединений к базам данных и внешним сервисам, а также упрощает отладку.

Мастер представляет собою процесс, ответственный за работу слотов на определенном сервере. В его обязанности входит запуск и остановка слотов, поддержание оптимального количества слотов на сервере в зависимости от его конфигурации, доступной памяти, а также мониторинг жизнедеятельности слотов, уничтожение зависших и использовавших слишком много системных ресурсов слотов.

Сущность *менеджер* выступает в роли процесса-руководителя кластера, который запускается в одном экземпляре и отвечает за весь кластер. Данный процесс на основе данных от сервера очередей и статистических данных от мастеров определяет необходимое количество слотов каждого типа и решает, какой конкретный слот будет заниматься заданиями какого типа [3].

ЛИТЕРАТУРА

1. Структуры данных, Очередь [Электронный источник]. – 2017. – Режим доступа: https://prog-cpp.ru/data-queue/. – Дата доступа: 27.09.2017.
2. Сервер очередей [Электронный источник]. – 2017. – Режим доступа: https://habrahabr.ru/company/mailru/blog/216363/. – Дата доступа: 27.09.2017.
3. Инфраструктура обработки очередей с социальной сети Мой Мир [Электронный источник]. – 2017. – Режим доступа: https://habrahabr.ru/company/mailru/blog/228131/. – Дата доступа: 27.09.2017.