Проекта Предиктивное уведомление клиентов boost::beast

Мельников Лев Александрович СКБ 161

2019-12-11

1 Используемые понятия

- Уведомление клиентов - мероприятие в котором, ограниченное число операторов совершают звонки большому числу клиентов
- Метод уведомления клиентов - набор правил, согласно которым определяется стратегия уведомления клиентов
- Прогрессивный метод уведомления клиентов - метод, по которому новый звонок совершается, только если количество свободных операторов больше чем количество вызовов, ожидающих ответа клиента
- Брошенный вызов - звонок, который был принят клиентом, но не был отвечен оператором
- Предиктивный метод уведомления клиентов - метод согласно, которому новые звонки могут совершаться, даже если свободных операторов меньше чем вызовов, ожидающих ответа. Предиктивные методы с помощью предсказаний минимизируют количество брошенных вызовов
- АТС - Автоматическая телефонная станция
- ВАТС - Виртуальная АТС

2 Постановка задачи

Разработать программу для управления уведомлением клиентов по прогрессивному методу. Программа должна работать по протоколоу http и поддерживать следующие методы.

- begin - создать сессию обзвона. Параметр session - название сессии. Дополнительный параметр stat_size - размер статистики, которую будет принимать эта сессия.

- end - удалить сессию. Параметр session - название сессии.

- add - добавить в статистику сессии вызов. Параметр session - название сессии. setup - время до поднятия трубки. talk - время разговора.

- get - Получить количество вызовов, которое нужно совершить для максимальной эффективности. Параметр session - название сессии. free - количество свободных операторов. incoming - количество вызовов, которые еще не приняты.

Реализация должна в методе get содержать механизм для рассчета по прогрессивному методу, но быть легко расширяемой на предиктивный метод. Реализация также должна иметь возможность обрабатывать несколько сессий одновременно.

3 Обзор технологии

В реализации используется boost::asio и boost::beast::http. boost::asio используется для многопоточной ассинхронной обработки запросов. Функция accept принимает соединение(здесь и далее асинхронно) и в обработчике новго соединения создает объект класса worker, выделяя его в динамическую память. После этого обработчик заново вызвает функцию accept. Таким образом достигается непрерывное и ассинхронное принятие соединений. Объект класса worker - в конструкторе создает shared_ptr, ссылающийся на себя. И далее начинает обрабатывать соединение. Полученные данные передаются для обработки в объект класса session_manager - в котором, реализована логика обзвона. После того как запрос обработан и ответ отправлен (а также в случае ошибки и невозможности обработать запрос), shared_ptr зануляется, что приводит к освобождению памяти выделенной под worker.

4 Использование boost::beast

Из boost::beast::http используются parser и request(response). В parser делегируется работа по обработке http запроса, он передается в качестве аргумента в функцию async_read. request(response) используются для чтения(составления) запроса(ответа) в удобном виде.

5 Трудности в разработке

Узким местом в производительности являются блокировки общих данных, а особенно блокировки в методе get, тк он может требовать времени сильно больше чем вся остальная часть программы. Было предложено следующее решение. В реализации есть хранилище сессий и его мьютекс - storage_lock. У каждой сессии есть свой мьютекс. Порядок захвата мьютексов определен следующим образом:

1. begin
   (a) master(lock)
   (b) Добавление сессии в storage(Быстрая операция)
   (c) master(unlock)
2. add
   (a) master(lock)
   (b) Поиск нужной сессии(Быстрая операция)
   (c) session(lock)
   (d) master(unlock)
   (e) Добавление статистики(Быстрая операция)
   (f) session(unlock)
3. get
   (a) master(lock)
   (b) Поиск нужной сессии(Быстрая операция)
   (c) session(lock)
   (d) master(unlock)
   (e) Предиктивный метод(Медленная операция операция)
   (f) session(unlock)
4. end
   (a) master(lock)
   (b) Поиск нужной сессии(Быстрая операция)
   (c) session(lock)
   (d) session(unlock)

(e) Удаление сесси(Быстрая операция)

(f) master(unlock)

Блокировке 4(c) - 4(d) нужны, чтобы убедиться, что мьютексом сессии никто не владеет. Так как порядок захвата мьютексов одинаковый - deadlock невозможен.

6 Материалы

Ссылка на репозиторий: https://github.com/markgrin/pdc_daemon . В коде также присутствует документация, которая не приведена здесь. Также в репозитории есть файл test.py для тестирования работоспособности программы.