Задание для допуска к зачету.
Java 2018, 2МІТ СПбАУ

Задание состоит в том, чтобы написать графическое приложение, которое будет позволять тестировать производительность той или иной архитектуры сервера.

Суть сетевого взаимодействия в данной задаче сводится к тому, что клиент просит сервер отсортировать некий массив (для определенности какой-нибудь квадратичной сортировкой), сервер его сортирует и возвращает клиенту отсортированный массив.

Клиент и сервер обмениваются в данной задаче сообщениями одинакового вида - число N и массив из N чисел типа int.

Протокол взаимодействия должен быть реализован с помощью Google Protocol Buffers сообщений. Чтобы в потоке данных отделять одно сообщение от другого, договоримся, что в сетевом пакете хранится размер сообщения Protobuf и далее само сообщения Protobuf.

В данной задаче существуют следующие константы, оказывающие влияние на производительность:

- N - количество элементов в сортируемых массивах
- M - количество одновременно работающих клиентов
- Δ - временной промежуток от получения ответа от сервера на одно сообщение клиента до начала отправки следующего сообщения клиента.
- X - суммарное количество запросов, отправляемых каждым клиентом. Имеется в виду, что все клиенты работают по одному принципу - включаются, отсылают X запросов (с учетом Δ между ними) и заканчивают свою работу.

Метрики, по которым мы будем тестировать приложение:

- Время обработки запроса (сортировки в данном случае) на сервере, ms (считается с момента начала обработки до момента окончания обработки)
- Время обработки клиента на сервере, ms (считается с момента получения запроса от клиента до момента отсылки результата клиенту)
- Среднее время одного запроса на клиенте, ms. Считаем время от старта клиента до конца его работы, делим на X. Усредняем по всем клиентам.

Варианты архитектуры сервера:
TCP:__

- Клиент устанавливает постоянное соединение. Сервер создает отдельный поток на общение (прием запроса, выполнение запроса и отправку ответа) с конкретным клиентом.
- Клиент устанавливает постоянное соединение. Сервер создает по отдельному потоку на каждого клиента для приема от него данных + по одному SingleThreadExecutor для отсылки данных клиенту. Полученные от клиента запросы попадают в общий пул потоков фиксированного размера. После

обработки ответ клиенту отправляется через соответствующий SingleThreadExecutor.
- Клиент устанавливает постоянное соединение. Сервер производит неблокирующую обработку. Каждый запрос обрабатывается пуле потоков фиксированного размера. Сервер работает с сокетами в однопоточном режиме (один поток и селектор на прием всех сообщений и один поток и селектор на отправку всех сообщений).
- (опционально +0.1) Сервер обрабатывает запросы в асинхронном режиме.

При старте приложение должна позволять

1. Выбрать архитекту, которую мы тестируем.
2. Задать число X.
3. Выбрать параметр, который будет меняться в ходе тестирования (N, M или Δ)
4. Задать в каких пределах и с каким шагом он будет изменяться
5. Задать постоянные значения других параметров

Результатом работы программы должны быть три графика - по графику на каждую метрику. И три файла с полученными значениями. В начале каждого файла (или в “соседнем” файле с описанием) должно быть описаны все установленные значения из пунктов 1-5.

Для сдачи работы необходимо:

- Написать программу и загрузить ее в систему контроля версий
- Построить в отдельной программе 9 графиков - как меняется каждая из метрик при изменении каждого из параметров. На каждом графике должны быть отображены кривые соответствующие всем архитектурам
- Выложить графики в систему контроля версий
- Выложить в систему контроля версий файлы с результатами, по которым построены итоговые графики
- Подумать и осознать, почему получились такие результаты
- Прислать на адрес anton.m.kuznetsov@gmail.com ссылку на репозиторий. Или добавить Антона Михайловича к своему репозиторию.
- Получить одобрение АМ

P.S. Замечания, комментарии и т.д. Всячески приветствуются - пишите на почту.
P.P.S. За списывания буду карать. Сильно.