Задание «асинхронный HTTP сервер»

* Реализовать две протокольные команды:
  + POST “PING <userId>”, на которую сервер должен отвечать “PONG N”, где N количество раз, сколько этот клиент запрашивал PING. Параметр userId передается в теле запроса в произвольном виде.
  + GET "STAT", на которую сервер должен ответить списком объектов [{"userId": "user-id-value-here", "times": N}] (постраничный вывод результатов).
* Требования:
  + executable jar, java –jar your-service-name-here.jar `any-parameters-here`
  + В архитектуру заложить то, что типов команд в будущем будет много (не только PING и STAT).
  + НЕ использовать популярные http-фреймворки (Spring, Jersey, etc).
  + Первый уровень сложности (чуть сложнее):
    - Решить задачу на pure Java.
    - В качестве асинхронного TCP/HTTP движка использовать Netty/Grizzly/Mina/Akka.
  + Второй уровень сложности (чуть легче):
    - Решить задачу на pure Java and servlets.
    - В качестве реализации Servlet API использовать Jetty/Tomcat/Undertow/Resin.
  + Сервис должен быть легко скалируемым горизонтально.
  + Хранилище данных выбрать соответственно задаче самостоятельно, учитывая, что сервис работает под высокой нагрузкой. Написать обоснование своего выбора.

Комментарии:

* В планах:
  + Постраничный вывод
  + Проверка на масштабирование горизонтально
  + Покрыть тестами
  + Нагрузочное тестирование рандомными запросами через SoapUI + мониторинг и профайлинг через VisualVM, Mission Control, Flight Recorder, NetBeans Profiler. + мониторинг диска, БД, памяти, cpu.
  + Дроп и создание таблицы при старте
  + JSON в /STAT надо сделать, пока text/plain + версионирование добавить
  + Подумать над схемой таблиц, индексы. Пока самое простое.
  + Подумать надо постраничным гетом. Выходит страницу надо указывать в квери параметрах, так они могут попасть под кэши, если их потом включат. Лучше бы след страницу указывать в теле запроса.
* Postges, т. к. реляционные базы быстрее NoSQL. В то же время postgres поддерживает масштабирование, работает у многих в проде и бесплатная. И если все же захочется перейти на хранение JSON, то postgres это тоже поддерживает.
* UserId сделал int, чтобы не заморачиваться с SQL-инъекциями. В методе STAT вывожу его как строку
* Делаю на Netty/Grizzly/Mina/Akka, т. к. с ними никогда не работал. Так для себя полезнее.