HighLoad
для начинающих

highload++ 2014

Конференция разработчиков
высоконагруженных систем

HighLoad для начинающих

Dmitry E. Oboukhov

31 октября 2014

HighLoad, что это?

- Конференция?

HighLoad, что это?

- Конференция?
- Высокая нагрузка?

HighLoad, что это?

- Конференция?
- Высокая нагрузка?
- Миф?!

Высокая нагрузка что это?

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду?

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду? - средний CPU

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду? - средний CPU
- Более реалистично!

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду? - средний CPU
- Более реалистично!
- 1 запрос в секунду?

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду? - средний CPU
- Более реалистично!
- 1 запрос в секунду? - любой веб сервер?...

Высокая нагрузка что это?

- $53.328 * 10^9$ запросов в секунду? - средний CPU
- Более реалистично!
- 1 запрос в секунду? - любой веб сервер?...
  перекодирующий видеоролики? :)

Высокая нагрузка это:

Высокая нагрузка это:

Нагрузка, с которой не справляется железо

Когда это бывает?

Когда это бывает?

Достигнуты технические ограничения

Когда это бывает?

Достигнуты технические ограничения

- Сеть

Когда это бывает?

Достигнуты технические ограничения

- Сеть - за рамками данного доклада

Когда это бывает?

Достигнуты технические ограничения

- Сеть - за рамками данного доклада
- Память

Когда это бывает?

Достигнуты технические ограничения

- Сеть - за рамками данного доклада
- Память
- CPU

Причины

Причины

- Недоиспользование железа

Причины

- Недоиспользование железа
- Трудности масштабирования

Архитектурные проблемы

Недоиспользование железа

Рассмотрим типичный вебсервер.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.
- Формирование ответа (template).

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.
- Формирование ответа (template).
- Отправка ответа клиенту.

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.
- Формирование ответа (template).
- Отправка ответа клиенту.

Традиционная реализация

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.
- Формирование ответа (template).
- Отправка ответа клиенту.

Традиционная реализация

- один процесс на один цикл

Недоиспользование железа

Рассмотрим типичный вебсервер. на Perl, Python, Ruby...

Задачи одного цикла

- Чтение запроса из сети.
- Парсинг запроса http.
- Валидация запроса, выбор контроллера.
- Запрос(ы) к хранилищу данных.
- Формирование ответа (template).
- Отправка ответа клиенту.

Традиционная реализация

- один процесс на один цикл
- один тред на один цикл

Недоиспользование железа

Начались разговоры о HighLoad?

Недоиспользование железа

Начались разговоры о HighLoad?

- Увеличение числа процессов/тредов.

Недоиспользование железа

Начались разговоры о HighLoad?

- Увеличение числа процессов/тредов.
- Увеличение числа серверов.

Недоиспользование железа

Вернемся к рассматриваемому серверу

Недоиспользование железа

Вернемся к рассматриваемому серверу

- Проблемы наступили при ≈100 запросах в секунду.

Недоиспользование железа

Вернемся к рассматриваемому серверу

- Проблемы наступили при $\approx$100 запросах в секунду.
- Увеличили число процессов в работе.

Недоиспользование железа

Вернемся к рассматриваемому серверу

- Проблемы наступили при ≈100 запросах в секунду.
- Увеличили число процессов в работе.
  Помогло.

Недоиспользование железа

Вернемся к рассматриваемому серверу

- Проблемы наступили при $\approx$100 запросах в секунду.
- Увеличили число процессов в работе.
  Помогло.
- Новые проблемы при $\approx$150 запросов в секунду.

Недоиспользование железа

Вернемся к рассматриваемому серверу

- Проблемы наступили при ≈100 запросах в секунду.
- Увеличили число процессов в работе.
  Помогло.
- Новые проблемы при ≈150 запросов в секунду.
- Дальнейшее увеличение числа процессов помогает слабо.

Недоиспользование железа

Что делать?

Недоиспользование железа

Что делать?

- Менять архитектуру?

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!
- Добавлять второй сервер?

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!
- Добавлять второй сервер?
  - Это тоже не просто!
  (бизнеслогика)

Спокойно!

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!
- Добавлять второй сервер?
  - Это тоже не просто!
  (бизнеслогика)

Спокойно!

- Провести анализ архитектуры.

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!
- Добавлять второй сервер?
  - Это тоже не просто!
  (бизнеслогика)

Спокойно!

- Провести анализ архитектуры.
- Провести измерения.

Недоиспользование железа

Что делать?

- Менять архитектуру?
  - Мы над этим 3 года работали!
- Добавлять второй сервер?
  - Это тоже не просто!
  (бизнеслогика)

Спокойно!

- Провести анализ архитектуры.
- Провести измерения.
- Найти слабые места.

Недоиспользование железа

Измерения

Недоиспользование железа

Измерения

Чтение запроса из сети.	15K RPS

Недоиспользование железа

Измерения

Чтение запроса из сети.	15K RPS
Парсинг запроса, валидация, выбор контроллера.	150K RPS/CPU

Недоиспользование железа

Измерения

Чтение запроса из сети.	15K RPS
Парсинг запроса, валидация, выбор контроллера.	150K RPS/CPU
Запросы к хранилищу.	60K RPS

Недоиспользование железа

Измерения

Чтение запроса из сети.	15K RPS
Парсинг запроса, валидация, выбор контроллера.	150K RPS/CPU
Запросы к хранилищу.	60K RPS
Формирование ответа (Соединение данных с template)	100K RPS/CPU

Недоиспользование железа

Измерения

Чтение запроса из сети.	15K RPS
Парсинг запроса, валидация, выбор контроллера.	150K RPS/CPU
Запросы к хранилищу.	60K RPS
Формирование ответа (Соединение данных с template)	100K RPS/CPU
Отправка ответа клиенту.	15K RPS

Недоиспользование железа

Итого

$$\frac{1}{\frac{1}{15\ \ 10^3}+\frac{1}{150\ \ 10^3}+\frac{1}{60\ \ 10^3}+\frac{1}{100\ \ 10^3}+\frac{1}{15\ \ 10^3}} = 6000\,RPS$$

Недоиспользование железа

Итого

$$\frac{1}{\frac{1}{15\ \ 10^3}+\frac{1}{150\ \ 10^3}+\frac{1}{60\ \ 10^3}+\frac{1}{100\ \ 10^3}+\frac{1}{15\ \ 10^3}} = 6000\,RPS$$

Но, позвольте!

Недоиспользование железа

Итого

$$\frac{1}{\frac{1}{15\ \ 10^3}+\frac{1}{150\ \ 10^3}+\frac{1}{60\ \ 10^3}+\frac{1}{100\ \ 10^3}+\frac{1}{15\ \ 10^3}} = 6000\,RPS$$

Но, позвольте!

- У нас проблемы на 150 RPS!

Недоиспользование железа

Итого

$$\frac{1}{\frac{1}{15\ \ 10^3}+\frac{1}{150\ \ 10^3}+\frac{1}{60\ \ 10^3}+\frac{1}{100\ \ 10^3}+\frac{1}{15\ \ 10^3}} = 6000\,RPS$$

Но, позвольте!

- У нас проблемы на 150 RPS!
- Тут что-то не так!

Недоиспользование железа

Начинаем разбираться

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.
- То же самое и с взаимодействием с клиентом.

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.
- То же самое и с взаимодействием с клиентом.

Что делать?

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.
- То же самое и с взаимодействием с клиентом.

Что делать?

- Попробуем еще увеличить число процессов?

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.
- То же самое и с взаимодействием с клиентом.

Что делать?

- Попробуем еще увеличить число процессов?
- Проблемы стали больше!

Недоиспользование железа

Начинаем разбираться

- Хранилище выходит на свои RPS при достаточно большом числе соединений к нему.
- Либо хранилище надо располагать локально.

  неприемлемо.
- То же самое и с взаимодействием с клиентом.

Что делать?

- Попробуем еще увеличить число процессов?
- Проблемы стали больше!
- Почему?!

Недоиспользование железа

Резюме ситуации, еще раз

Недоиспользование железа

Резюме ситуации, еще раз

- Имеется 100500 строк кода, над которым работали несколько лет.

Недоиспользование железа

Резюме ситуации, еще раз

- Имеется 100500 строк кода, над которым работали несколько лет.
- Этот код AS IS по результатам измерений может выдавать гораздо больше RPS чем в реальности.

Недоиспользование железа

Резюме ситуации, еще раз

- Имеется 100500 строк кода, над которым работали несколько лет.
- Этот код AS IS по результатам измерений может выдавать гораздо больше RPS чем в реальности.
- Проблемы начинаются на уровне RPS на порядок меньших, нежели расчетные.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.
- Формирование запроса (запросов) в БД.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.
- Формирование запроса (запросов) в БД.
- Ожидание ответа (ответов) из БД.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.
- Формирование запроса (запросов) в БД.
- Ожидание ответа (ответов) из БД.
- Соединение данных из БД с шаблоном.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.
- Формирование запроса (запросов) в БД.
- Ожидание ответа (ответов) из БД.
- Соединение данных из БД с шаблоном.
- Ожидание отправки данных клиенту.

Недоиспользование железа

Еще раз рассмотрим цикл обработки

- Ожидание запроса (данных) от пользователя.
- Парсинг запроса, валидация.
- Формирование запроса (запросов) в БД.
- Ожидание ответа (ответов) из БД.
- Соединение данных из БД с шаблоном.
- Ожидание отправки данных клиенту.
- Следующий клиент!

Недоиспользование железа

Измеряем

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя. 70 мкс

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя.	70 мкс
Парсинг запроса, валидация.	6 мкс

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя.	70 мкс
Парсинг запроса, валидация.	6 мкс
Формирование запроса (запросов) в БД.	1 мкс

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя.	70 мкс
Парсинг запроса, валидация.	6 мкс
Формирование запроса (запросов) в БД.	1 мкс
Ожидание ответа (ответов) из БД.	16 мкс

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя.	70 мкс
Парсинг запроса, валидация.	6 мкс
Формирование запроса (запросов) в БД.	1 мкс
Ожидание ответа (ответов) из БД.	16 мкс
Соединение данных из БД с шаблоном.	10 мкс

Недоиспользование железа

Измеряем

Ожидание запроса (данных) от пользователя.	70 мкс
Парсинг запроса, валидация.	6 мкс
Формирование запроса (запросов) в БД.	1 мкс
Ожидание ответа (ответов) из БД.	16 мкс
Соединение данных из БД с шаблоном.	10 мкс
Ожидание отправки данных клиенту.	70 мкс

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 =$

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 = 17$ мкс

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 = 17$ мкс
- Чего-либо ожидали: $70 + 16 + 70 =$

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 = 17$ мкс
- Чего-либо ожидали: $70 + 16 + 70 = 156$ мкс

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 = 17$ мкс
- Чего-либо ожидали: $70 + 16 + 70 = 156$ мкс
- Код выполняется только 10% времени!

Недоиспользование железа

Итого

- Код выполнялся: $6 + 1 + 10 = 17$ мкс
- Чего-либо ожидали: $70 + 16 + 70 = 156$ мкс
- Код выполняется только 10% времени!
- И при этом тормозит!

Недоиспользование железа

```
#include <unistd.h>

int main(int argc, char **argv) {
        int i;
        for (;;) {
                usleep(70);     usleep(7);
                usleep(16);     usleep(10);
                usleep(70);
        }
}
```

Недоиспользование железа

Итого

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU - по моим измерениям - где-то 15% загрузки на CPU

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Вернемся к нашему серверу

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Вернемся к нашему серверу

- Каждая отдельная часть имеет хорошую производительность

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Вернемся к нашему серверу

- Каждая отдельная часть имеет хорошую производительность достаточную для развития проекта еще на несколько лет вперед.

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Вернемся к нашему серверу

- Каждая отдельная часть имеет хорошую производительность достаточную для развития проекта еще на несколько лет вперед.
- Большую часть времени (90%) наш код проводит в ожидании.

Недоиспользование железа

Итого

- Код, делающий только sleep в цикле неплохо грузит CPU
  - по моим измерениям - где-то 15% загрузки на CPU
- Запустив десяток таких “воркеров”, получаем примерно такую же нагрузку как на проблемном сервере.
- Понятно что пример синтетический (есть вопросы к реализации usleep).

Вернемся к нашему серверу

- Каждая отдельная часть имеет хорошую производительность достаточную для развития проекта еще на несколько лет вперед.
- Большую часть времени (90%) наш код проводит в ожидании.
- Что делать?

Недоиспользование железа

Просто реорганизовать код

Событийно-ориентированное программирование

Компьютер — это конечный автомат. Треды для тех людей, которые не умеют программировать конечные автоматы.

Алан Кокс

Избавимся от тредов!

Избавимся от тредов!

- и процессов.

Событийно-ориентированное программирование

Машина событий

Событийно-ориентированное программирование

Машина событий

- Вся работа делается в обработчике события.

Событийно-ориентированное программирование

Машина событий

- Вся работа делается в обработчике события.
  - в общем случае - callback.

Событийно-ориентированное программирование

Машина событий

- Вся работа делается в обработчике события.
  - в общем случае - callback.
- Когда программе нечего делать (например она ждет события), то управление возвращается машине событий.

Событийно-ориентированное программирование

Машина событий

- Вся работа делается в обработчике события.
  - в общем случае - callback.
- Когда программе нечего делать (например она ждет события), то управление возвращается машине событий.
  в общем случае - return из callback.

Событийно-ориентированное программирование

Машина событий

- Вся работа делается в обработчике события.
  - в общем случае - callback.
- Когда программе нечего делать (например она ждет события), то управление возвращается машине событий.
  в общем случае - return из callback.
- Обработчик события может генерировать другие события и устанавливать другие обработчики.

Событийно-ориентированное программирование

Перестроим наш сервер

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.
  - заменится обработчиком события "пришел ответ из БД"

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.
  - заменится обработчиком события "пришел ответ из БД"
- Соединение данных из БД с шаблоном.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.
  - заменится обработчиком события "пришел ответ из БД"
- Соединение данных из БД с шаблоном.
  - не изменится

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.
  - заменится обработчиком события "пришел ответ из БД"
- Соединение данных из БД с шаблоном.
  - не изменится
- Ожидание отправки данных клиенту.

Событийно-ориентированное программирование

Перестроим наш сервер

- Ожидание запроса от пользователя.
  - заменится обработчиком события "пришел запрос от пользователя"
- Парсинг запроса, валидация.
  - не изменится
- Формирование запроса (запросов) в БД.
  - не изменится
- Ожидание ответа (ответов) из БД.
  - заменится обработчиком события "пришел ответ из БД"
- Соединение данных из БД с шаблоном.
  - не изменится
- Ожидание отправки данных клиенту.
  - заменится обработчиком события "данные пользователю отправлены"

Событийно-ориентированное программирование

Итого

Событийно-ориентированное программирование

Итого

- Производительность одного сервера выросла в 10 раз

Событийно-ориентированное программирование

Итого

- Производительность одного сервера выросла в 10 раз
- Одного CPU/коннекта к БД достаточно для еще нескольких лет роста нагрузки.

Событийно-ориентированное программирование

Итого

- Производительность одного сервера выросла в 10 раз
- Одного CPU/коннекта к БД достаточно для еще нескольких лет роста нагрузки.
- Но слишком много переделок!

Событийно-ориентированное программирование

Что затронуто изменениями

Событийно-ориентированное программирование

Что затронуто изменениями

- Интерфейс с вебсервером (получение параметров запроса итп)

Событийно-ориентированное программирование

Что затронуто изменениями

- Интерфейс с вебсервером (получение параметров запроса итп)
  - некритично. В крайнем случае обходится написанием "врапперов". В большинстве случаев вообще незаметно.

Событийно-ориентированное программирование

Что затронуто изменениями

- Интерфейс с вебсервером (получение параметров запроса итп)
  - некритично. В крайнем случае обходится написанием "врапперов". В большинстве случаев вообще незаметно.
- Интерфейс с БД.

Событийно-ориентированное программирование

Что затронуто изменениями

- Интерфейс с вебсервером (получение параметров запроса итп)
  - некритично. В крайнем случае обходится написанием "врапперов". В большинстве случаев вообще незаметно.
- Интерфейс с БД.
  - критично. Много кода бизнеслогики поехало в callbacks. Сложную логику практически невозможно реализовать. Требуется переписывание 90% проекта.

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”
- Простое управление

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”
- Простое управление
  Три основных метода

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”
- Простое управление
  Три основных метода
  - Создать процесс (create, async)

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”
- Простое управление
  Три основных метода
  - Создать процесс (create, async)
  - Передать управление планировщику (yield, cede)

Событийно-ориентированное программирование

Планировщик

Поскольку планировщик OS - очень тяжелый, необходим планировщик userspace.

- Невытесняющая многозадачность
- Простое порождение “процессов”
- Простое управление
  Три основных метода
  - Создать процесс (create, async)
  - Передать управление планировщику (yield, cede)
  - Разбудить выбранный процесс (wakeup, ready)

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

- Регистрация события в машине событий

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

- Регистрация события в машине событий
- Передача управления планировщику

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

- Регистрация события в машине событий
- Передача управления планировщику
- Событие будит текущий процесс (файбер)

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

- Регистрация события в машине событий
- Передача управления планировщику
- Событие будит текущий процесс (файбер)
- Программа продолжает работу с данными от события

Событийно-ориентированное программирование

Интегрируем с машиной событий

Структура кода теперь выглядит так:

- Регистрация события в машине событий
- Передача управления планировщику
- Событие будит текущий процесс (файбер)
- Программа продолжает работу с данными от события

Итого

Вернулись к (почти) традиционному виду программы.

Событийно-ориентированное программирование

Вернемся к нашему серверу

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.
- Переписываем интерфейс с БД

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.
- Переписываем интерфейс с БД
  - относительно трудоемко, но решается враппером.

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.
- Переписываем интерфейс с БД
  - относительно трудоемко, но решается враппером.
- Переписываем другие сетевые обращения (если есть)

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.
- Переписываем интерфейс с БД
  - относительно трудоемко, но решается враппером.
- Переписываем другие сетевые обращения (если есть)
  - врапперы

Событийно-ориентированное программирование

Вернемся к нашему серверу

- Добавляем машину событий
- Добавляем библиотеку fibers
- Переписываем интерфейс с вебсервером
  - некритично, решается враппером.
- Переписываем интерфейс с БД
  - относительно трудоемко, но решается враппером.
- Переписываем другие сетевые обращения (если есть)
  - врапперы
- Итого: переписываем около 5% кода.

Библиотеки и языки

Библиотеки и языки

- Perl

Библиотеки и языки

- Perl
  Coro + AnyEvent

Библиотеки и языки

- Perl
  Coro + AnyEvent
- Python

Библиотеки и языки

- Perl
  Coro + AnyEvent
- Python
  fibers + twisted

Библиотеки и языки

- Perl
  Coro + AnyEvent
- Python
  fibers + twisted
- PHP5

Библиотеки и языки

- Perl
  Coro + AnyEvent
- Python
  fibers + twisted
- PHP5
  появился оператор yield, fiber

Что дальше?

Что дальше?

- Используем fiber'ы/event-машины в том языке к которому привыкли

Что дальше?

- Используем fiber'ы/event-машины в том языке к которому привыкли
- Рассматриваем существующие варианты

Что дальше?

- Используем fiber'ы/event-машины в том языке к которому привыкли
- Рассматриваем существующие варианты
  - Node.JS

Что дальше?

- Используем fiber'ы/event-машины в том языке к которому привыкли
- Рассматриваем существующие варианты
  - Node.JS
    - отказались от парадигмы fibers

Что дальше?

- Используем fiber'ы/event-машины в том языке к которому привыкли
- Рассматриваем существующие варианты
  - Node.JS
    - отказались от парадигмы fibers
  - Tarantool...

Tarantool

Tarantool

- Полноценный app-сервер

Tarantool

- Полноценный app-сервер
- БД на борту

Tarantool

- Полноценный app-сервер
- БД на борту
  - in-memory

Tarantool

- Полноценный app-сервер
- БД на борту
  - in-memory
  - disk

Tarantool

- Полноценный app-сервер
- БД на борту
  - in-memory
  - disk
- Сокеты, диск, http-сервер, очереди

Недостатки

Недостатки

- Для больших проектов одного CPU все-таки маловато

Недостатки

- Для больших проектов одного CPU все-таки маловато
- Реализации fiber'ов для традиционных ЯП плохо масштабируются по CPU/хостам.

Перспектива

Перспектива

- Erlang

Перспектива

- Erlang
  - хорошее масштабирование по CPU и хостам

Перспектива

- Erlang
  - хорошее масштабирование по CPU и хостам
  - очень качественное решение

Перспектива

- Erlang
  - хорошее масштабирование по CPU и хостам
  - очень качественное решение
  - высокий порог вхождения

Перспектива

- Erlang
  - хорошее масштабирование по CPU и хостам
  - очень качественное решение
  - высокий порог вхождения
- Go

Перспектива

- Erlang
  - хорошее масштабирование по CPU и хостам
  - очень качественное решение
  - высокий порог вхождения
- Go
  - более низкий порог вхождения