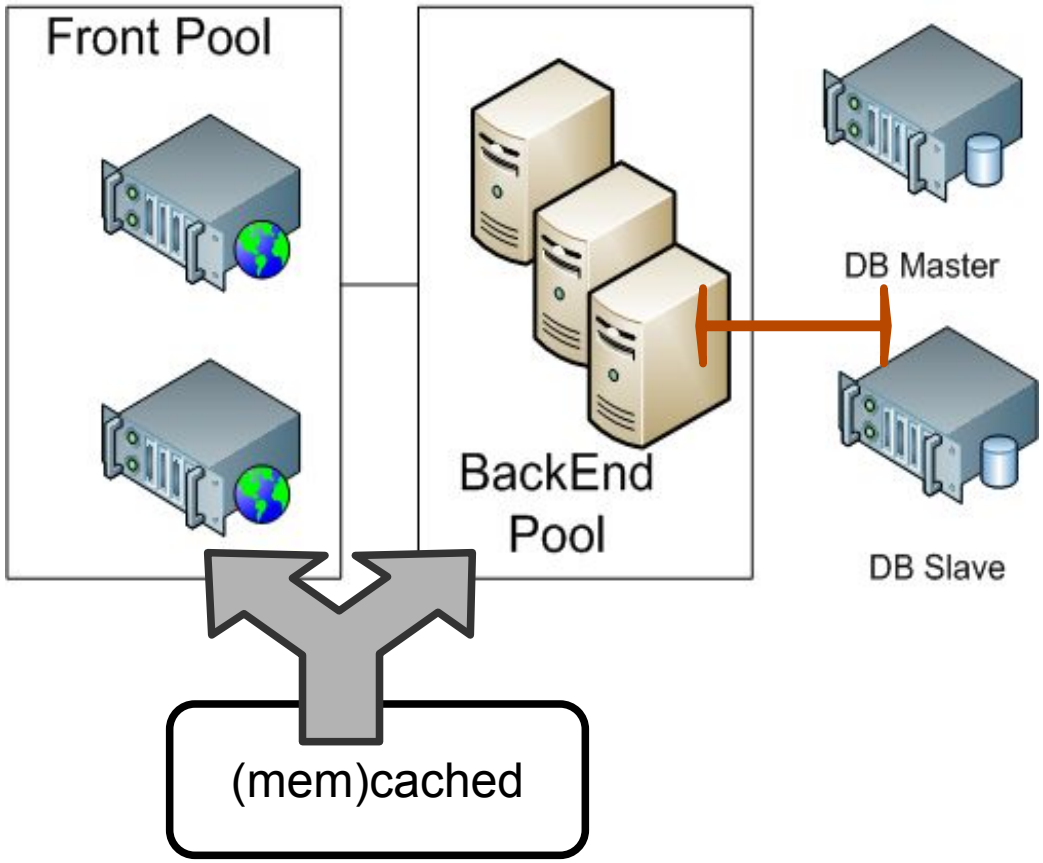
Пора ли отправлять C на свалку истории?

Пишем демонов на PHP с использованием расширения libevent

Кто мы такие?

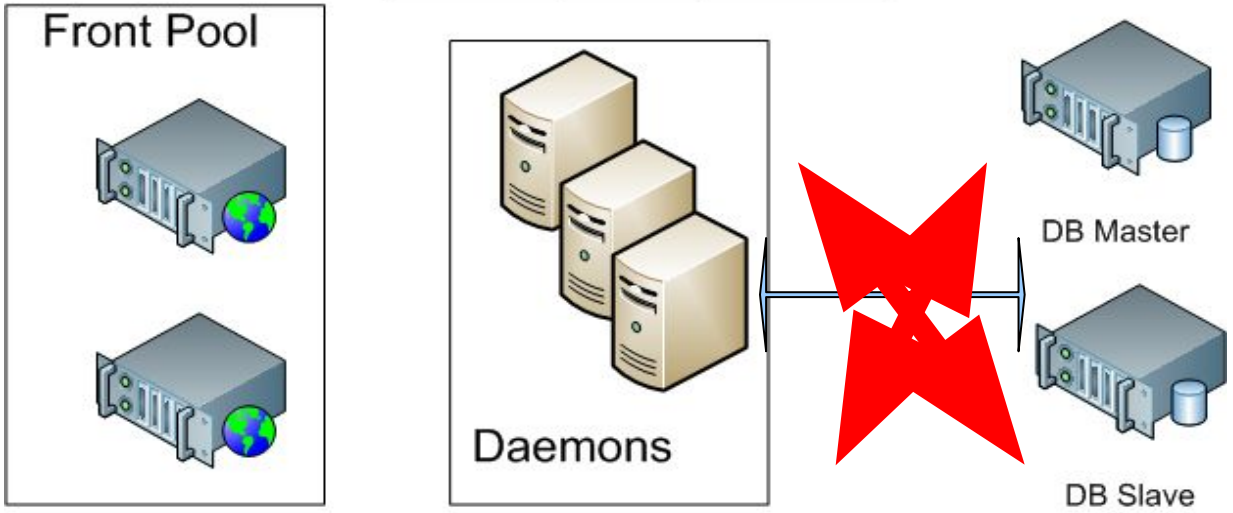
- Вадим Крючков [Long], руководитель группы разработки
- Андрей Голубев [440hz], ведущий разработчик
- Евгений Прудников, ведущий разработчик

Обычная архитектура



Наша архитектура — включаем демоны

- Данные в памяти
- Готовые структуры
- Нет затрат на
  - компиляцитю
  - инициализацию



Демонизация. Что есть такое libevent?

- Предоставляет простой механизм для запуска callback функций, при наступлении определенного события на дескрипторе:
  - READ
  - WRITE
  - TIMEOUT
  - SIGNAL
- http://www.monkey.org/~provos/libevent/
- http://ru.php.net/manual/en/intro.libevent.php

Пишем демона, работающего с сокетом

```
// Создаем сокет - event вешается на дескриптор
$rSocket = stream_socket_server (
    'tcp://127.0.0.1:666',
    $errno, $errstr,
STREAM_SERVER_BIND | STREAM_SERVER_LISTEN );

// далем его не блокирующим, что бы позволить принимать другие
    коннекты
stream_set_blocking ( $rSocket, 0 );
```

Пишем демона — подключаем libevent

```
// создаем событийную базу
$rBaseEvent = event_base_new ( );
// создаем новое событие для сокета
$rSocketEvent = event_new ( );
/**
 * ловим события "чтение" и после операции чтения возвращаем
     событие в базу
 * EV_READ - чтение
 * EV_PERSIST - вернуть событие в базу после выполнения
 */
event_set ( $rSocketEvent, $rSocket, EV_READ | EV_PERSIST,
     'onAcceptEvent' );
// устанавливаем событие в базу событий
event_base_set ( $rSocketEvent, $rBaseEvent );
// запускаем отслеживание
```

Метод обработки

```
function onAcceptEvent ( $rSocket, $rEvent, $args ) {
    global $rBaseEvent; // удобнее сделать через объект ;)

    static $iConnect = 0; // идентификатор конекта
    $iConnect++;
    // Примем коннект
    $rConnection = stream_socket_accept ( $rSocket );
    // далем коннект не блокирующим, что бы позволить принимать
     еще коннекты
    stream_set_blocking ( $rConnection, 0 );
    // создадим буфер обмена данными
    $buf = event_buffer_new ( $iConnect, 'onReadEvent', 'onWriteEvent',
     'onFailureEvent', $iConnect);
    // подключаем буфер к базе событий
    event_buffer_base_set ( $buf, $rBaseEvent );
```

Метод чтения

```
$iBufferReadLenght = 1024; // размер буфера чтения
function onReadEvent($rStream, $args) {
    global $iBufferReadLenght;
    $tmp = '';
    do {
        $tmp .= event_buffer_read ( $hBuffer, $this->iBufferReadLenght );

        if( $iBufferReadLenght > strlen($tmp) ) {
            break;
        }
    } while ( true );

    return $tmp;
}
```

Превращаем демона в ...

или не документированные возможности

Таймеры (thnx 440hz)

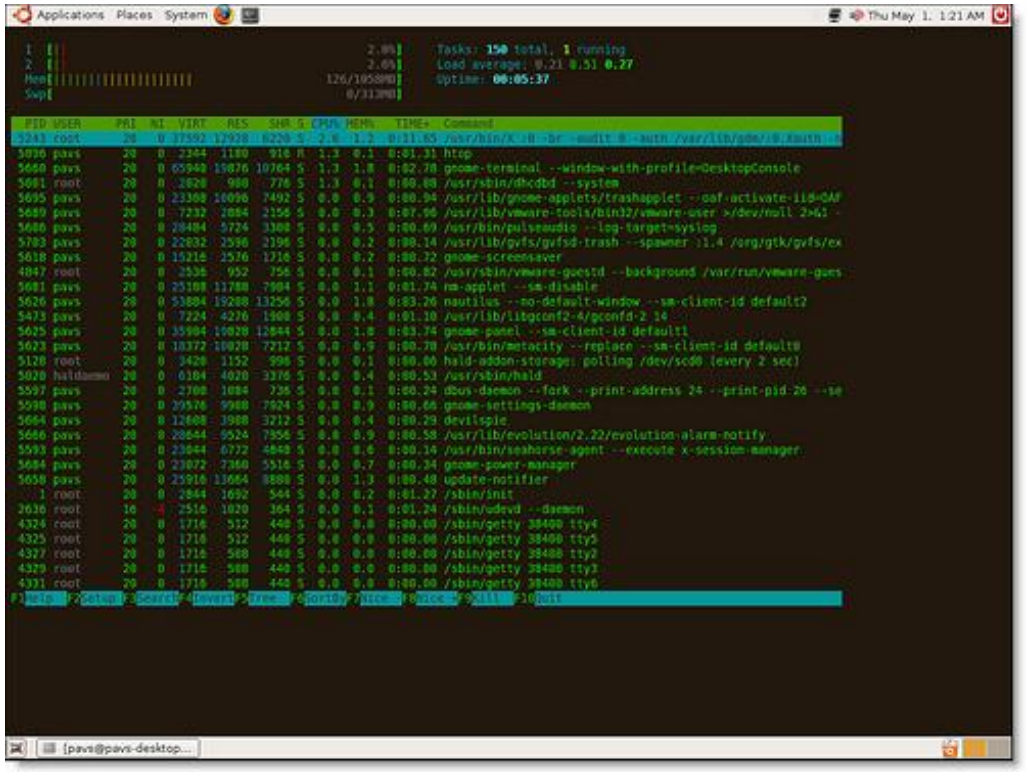
- Стандартный таймер libevent'a не работает :(
- **Выход есть!**
  - событие можно повесить на «любой» дескриптор
  - event_add (resource $event, int **$timeout**)

Таймеры - решение

- tmpfile() - открываем новый временный файл
- «вешаем» на этот дескриптор отложенное событие

```
event_set(
        $rTimers,
        $rTtmpFile,
        0,
        'onTimer',
);
```

Демонстрация



http://cyberdot.ru/src/socket.php
s

Подводные камни

- Очень мало информации и примеров
- Следить за ресурсами, не забываем их освобождать
- Хитрости при чтении данных, превышающих размер буфера
- Входных данных — много и они бывают «чужие» :)
- Проблемы с отслеживанием сигналов (EV_SIGNAL)

Даем нагрузку

Тестирование ботами

Имитируем … пользователей в on-line:

- Воспользовались API
- Написали приложение, генерирующее ботов

Результаты

Сервер Xeon 8x2.66GHz, RAM 8Gb:

- Около 2.5 тысяч запросов в секунду (не Hello, World)
- На 1 пользователя в online расходуется около 1Мб памяти
- Приложение (пока) не подвергалось жесткой оптимизации

Результаты

Сервер Xeon 8x2.66GHz, RAM 8Gb:

- Около 2.5 тысяч запросов в секунду (не Hello, World)
- На 1 пользователя в online расходуется около 1Мб памяти
- Приложение (пока) не подвергалось жесткой оптимизации
- Память не «течет» (1 месяц публичного бета-теста не выявили)

Советы

- **Научитесь «мыслить параллельно»**
  - Процесс не завершается
  - Чужие данные
- **Читайте исходники —** в них много полезного
- **Если демон будет не один —** напишите простенький фреймворк
- **Документируйте код + протокол взаимодействия**
- **Напишите хороший логгер** - без него отлаживать приложение будет сложно
  - Сделайте несколько уровней логгирования

Выводы

Выводы пока делать рано

:)

Выводы (серьезно)

- Можно рекомендовать к использованию на продакшене
- Позволяет держать хорошие нагрузки (при этом оставляя LA в разумных пределах)
- Реальные тесты — придется подождать :(

Вопросы?