В Яндексе разработчик – творческое начало,
создающее проекты, отягощенные нагрузкой

О проектах, отягощенных нагрузкой

Банальности, подтвержденные опытом.

Яndex деньги

Какие бывают системы

- Высокая нагрузка
  - 100+К событий в секунду
  - Гигабиты трафика
  - Все ради производительности
- Корпоративные
  - 1000 событий в сутки
  - Сложнозапутанная бизнес-логика
  - Все ради надежности и удовлетворения заказчика

Системы, отягощенные нагрузкой

- 10+ транзакций в секунду
- 300К+ операций в сутки
- Сложная бизнес-логика
- Высокая надежность
- Высокая доступность

Яndex деньги

Характерные особенности

Особенности постановки

- Сложная бизнес-логика уже с первой версии
- Множество различных клиентских мест
- Интеграция с другими системами

Особенности реализации

- 5-30 человеколет
- 4-20 серверов в production
- 10-500 транзакций в секунду в пике

Яndex деньги

Что нужно знать

- Среднесуточную нагрузку
- Максимальную пиковую нагрузку

  Суточная/3600/24*π ³?
- Цена потери транзакции
- Стоимость простоя сервиса
- Возможности технических перерывов
- Допустимые стоимости атаки
  - DDOS
  - Криптография

Яndex деньги

Допустимая цена решения

Сколько денег приносит одна транзакция?

(цифры не имеют отношения к действительности)

Жизнь на проценты :

1 транзакция = 10 центов, 100 К транзакций в сутки, пик – 40 в секунду

3M$ в год. Можно позволить и софт и железо

Жизнь на рекламу:

CPM=5$, 200K страниц в сутки, пик – 80 в секунду

500K $ в год. На софт уже не хватает.

Расходы на инфраструктуру должны составлять не более 10% от общего дохода

Яndex деньги

Со стоимостью определились, теперь можно

НЕМНОЖКО ПРО АРХИТЕКТУРУ

Яndex деньги

Общая архитектура

Проектировать архитектуру можно с точки зрения сисадмина:

SLB, RHEL, nginx, squid, Lustre …

Мы же будем говорить об архитектуре с точки зрения программирования.

Обычная трехзвенка

UI layer

- Отображение данных пользователю – тяжелые клиенты или веб-страницы

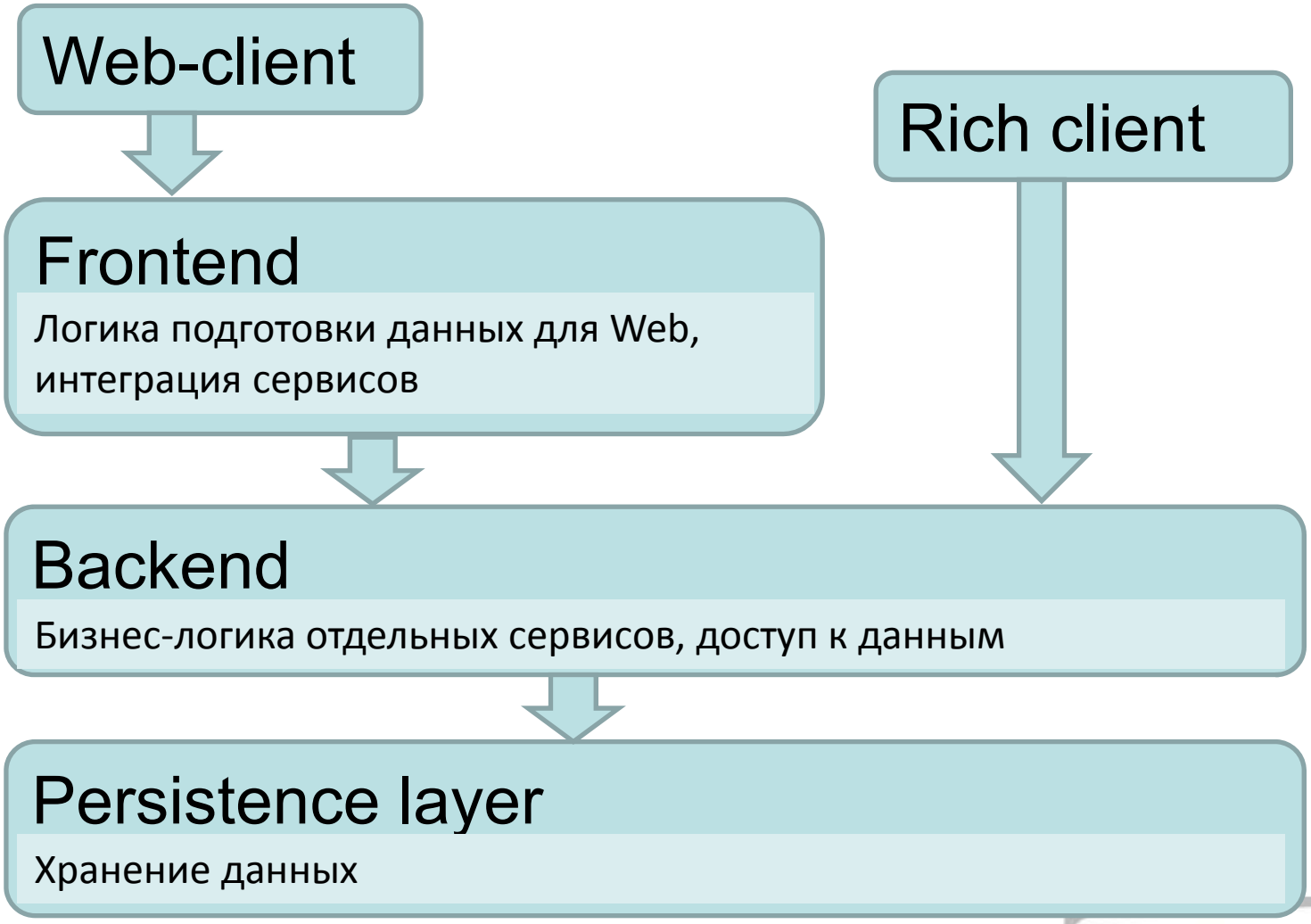
Application Layer

- Бизнес-логика, логика предметной области, доступ к данным, кэширование

Persistence Layer

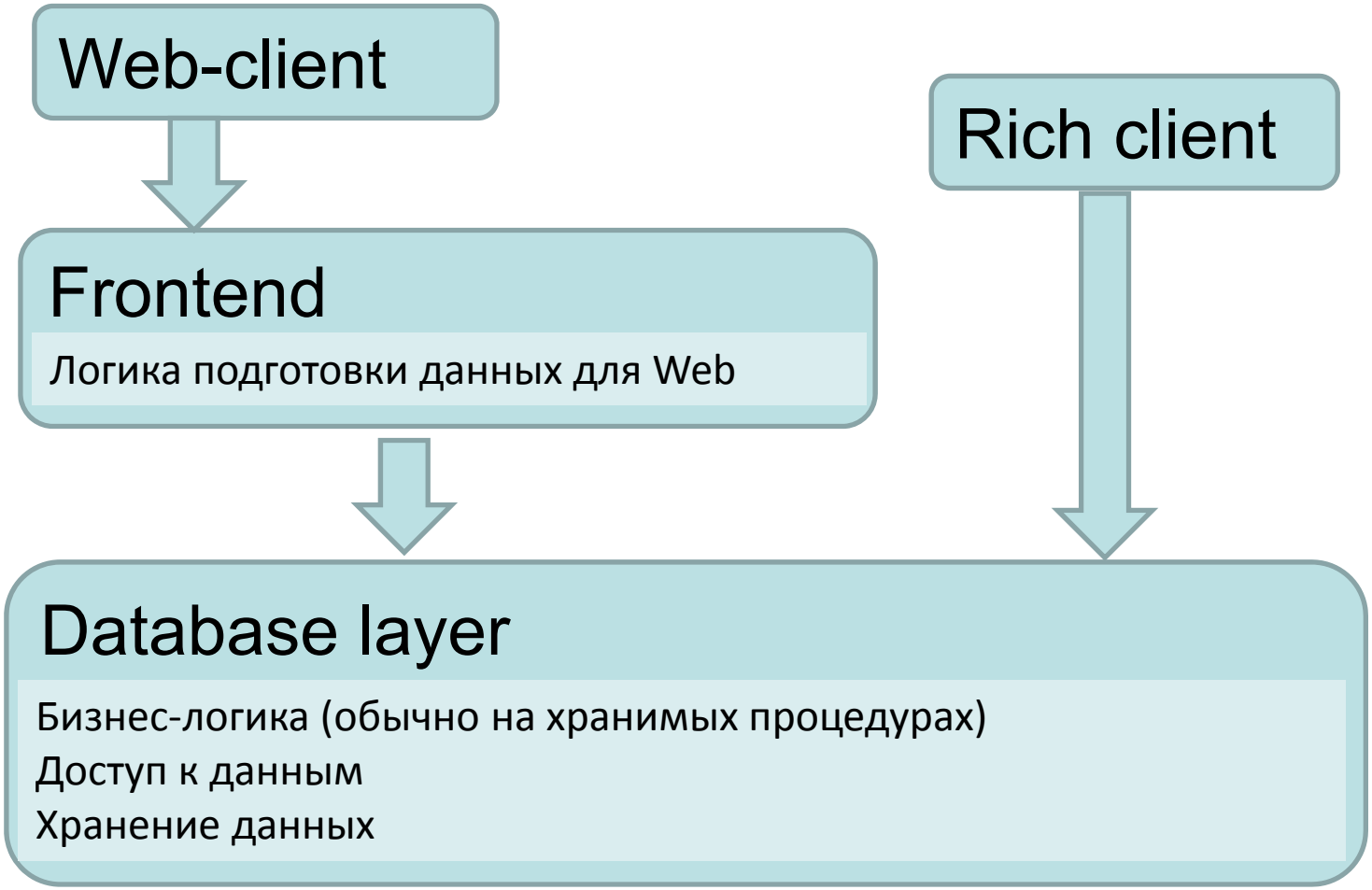
- Постоянное хранение данных (база данных)

С раздельным frontend и backend

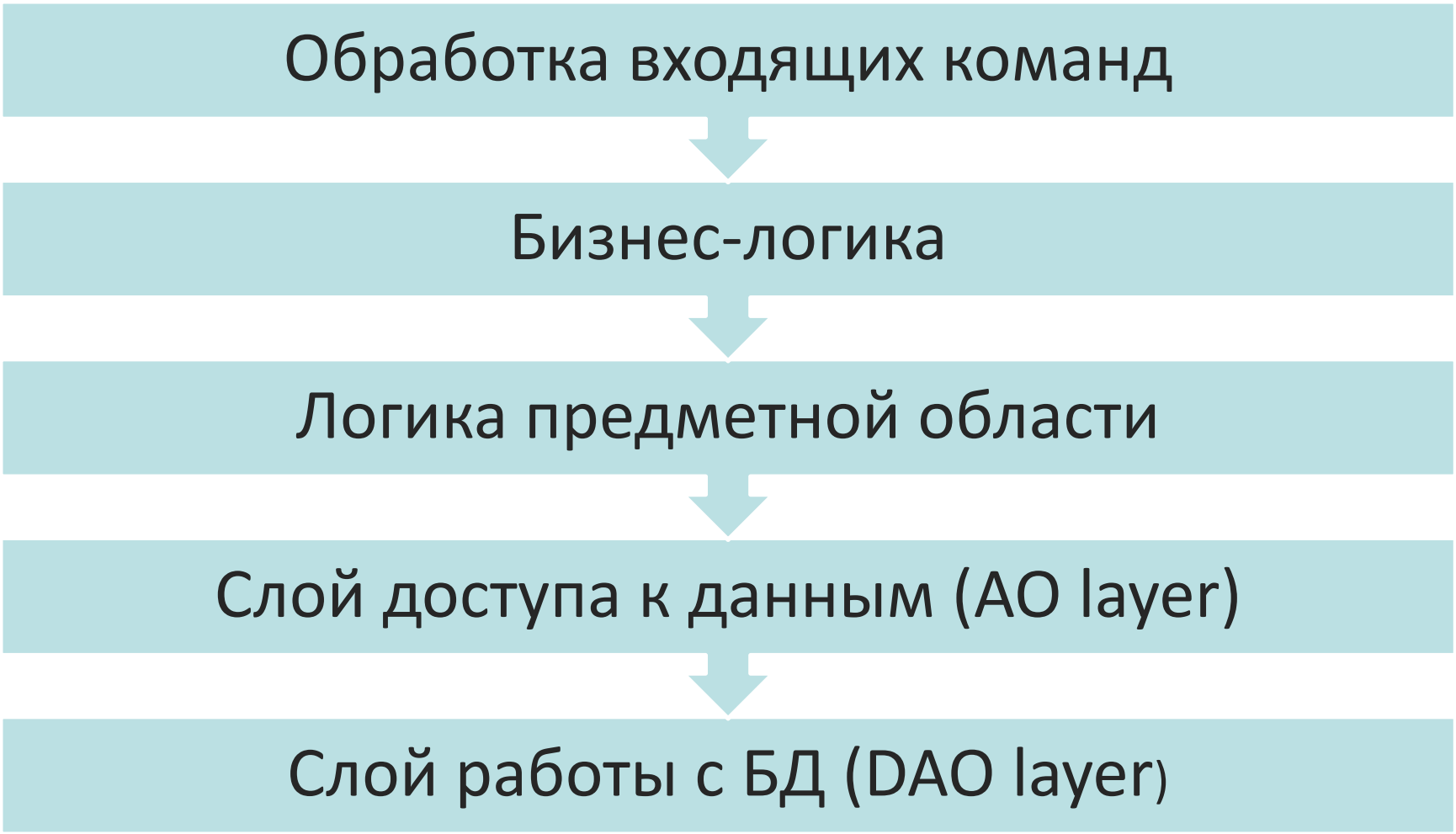


Яndex деньги

С интеграцией через БД



Как устроен backend?



Зачем нужны команды

Единство описания

Общий подход к управлению транзакциями

Общий подход к логгированию

Общий подход к правам доступа

Историчность вызовов

Простота тестирования

Популярные мифы

Надежда – глупое чувство

БД – умная, сама все сделает (двухзвенка)

Дорого, плохо масштабируемо.

EJB – умные, сами все сделают

Стоит узнать, как происходит кластеризация

ORM – умный, сам разберется

Стоит понять, а как именно хранит и кэширует.

Теперь пройдемся по тем вызовам, на которые приходится отвечать:

ОСНОВНЫЕ ПРОБЛЕМЫ

Яndex деньги

Высокая доступность

- Датацентров желательно иметь хотя бы два. А лучше три.
  - И каждый должен выдерживать годовой пик нагрузки.
- Архивные копии обеспечивают не доступность сервиса, а выживание бизнеса.
  - И нужно иметь два независимых механизма для резервирования пользовательских данных (backup + logging)
- Стандартные реализации Active/Standby все равно требуют доработки.
  - Автоматически сложно избежать возможного мерцания
  - В большинстве реализаций нет автоматического возврата к нормальному состоянию с синхронизацией данных
- Не следует верить всем обещаниям производителей.
- И не забывайте менять батарейки в RAIDax

Выпуск новых версий

- Любую версию надо сначала протестировать.
- Любой компонент системы можно безболезненно откатить на предыдущую версию.
- Все компоненты системы должны поддерживать предыдущие версии всех протоколов (т.е. работать с предыдущими версиями компонент)
- Выпуск версии должен производиться с минимальным напряжением головного мозга (лучше – одной кнопкой)
- Код клиентских приложений должен обновляться автоматически
- Структура баз данных может изменяться только в сторону увеличения
- Нужно тестировать как процедуру обновления версии, так и процедуру отката выпущенных версий
- Нужно знать время реакции своих пользователей на проблемы

Производительность

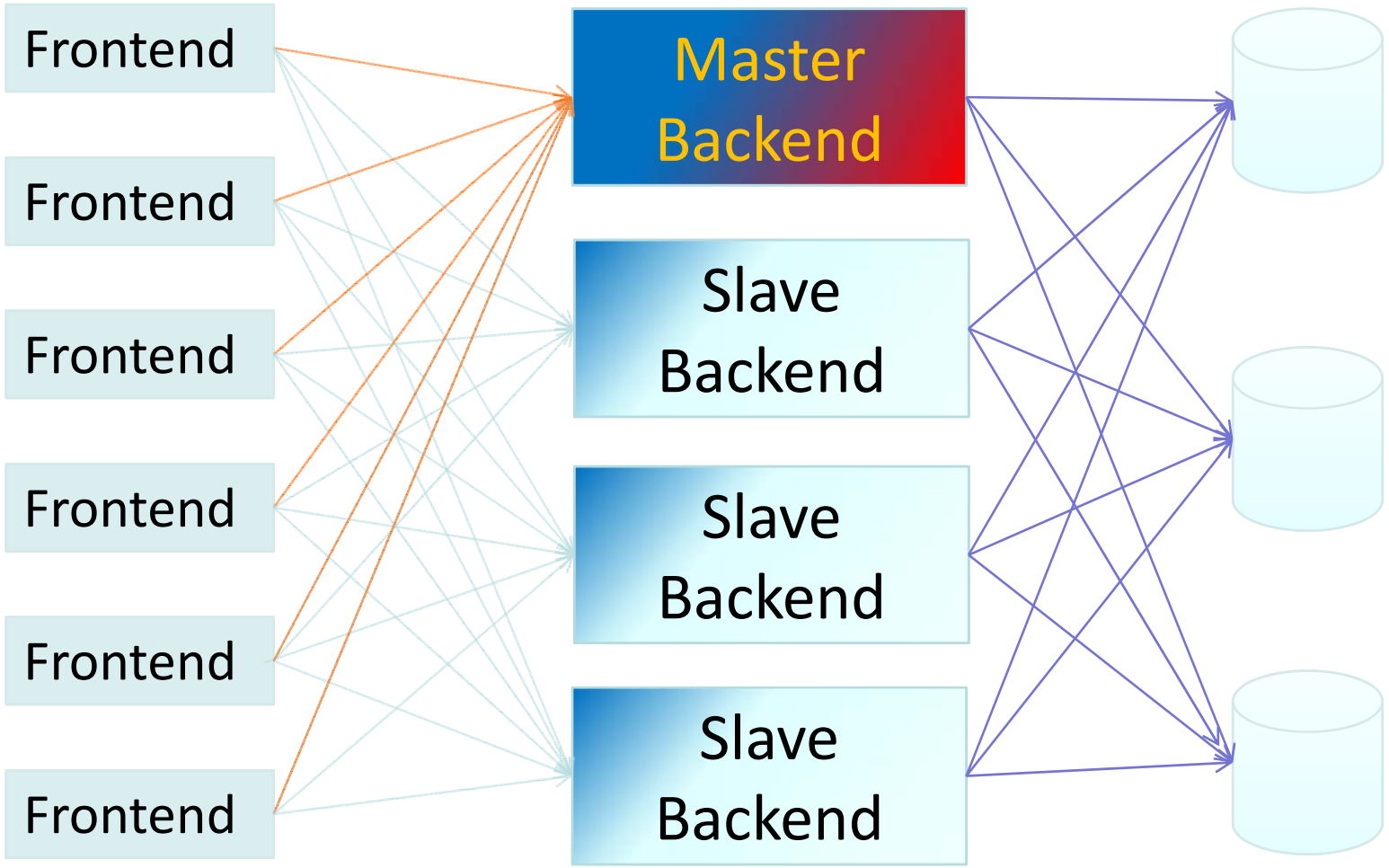
Кэширование – это наше все.

Горизонтально масштабировать лучше, чем вертикально. Но сложнее.

Много точек чтения, одна – изменения (1W+nR)

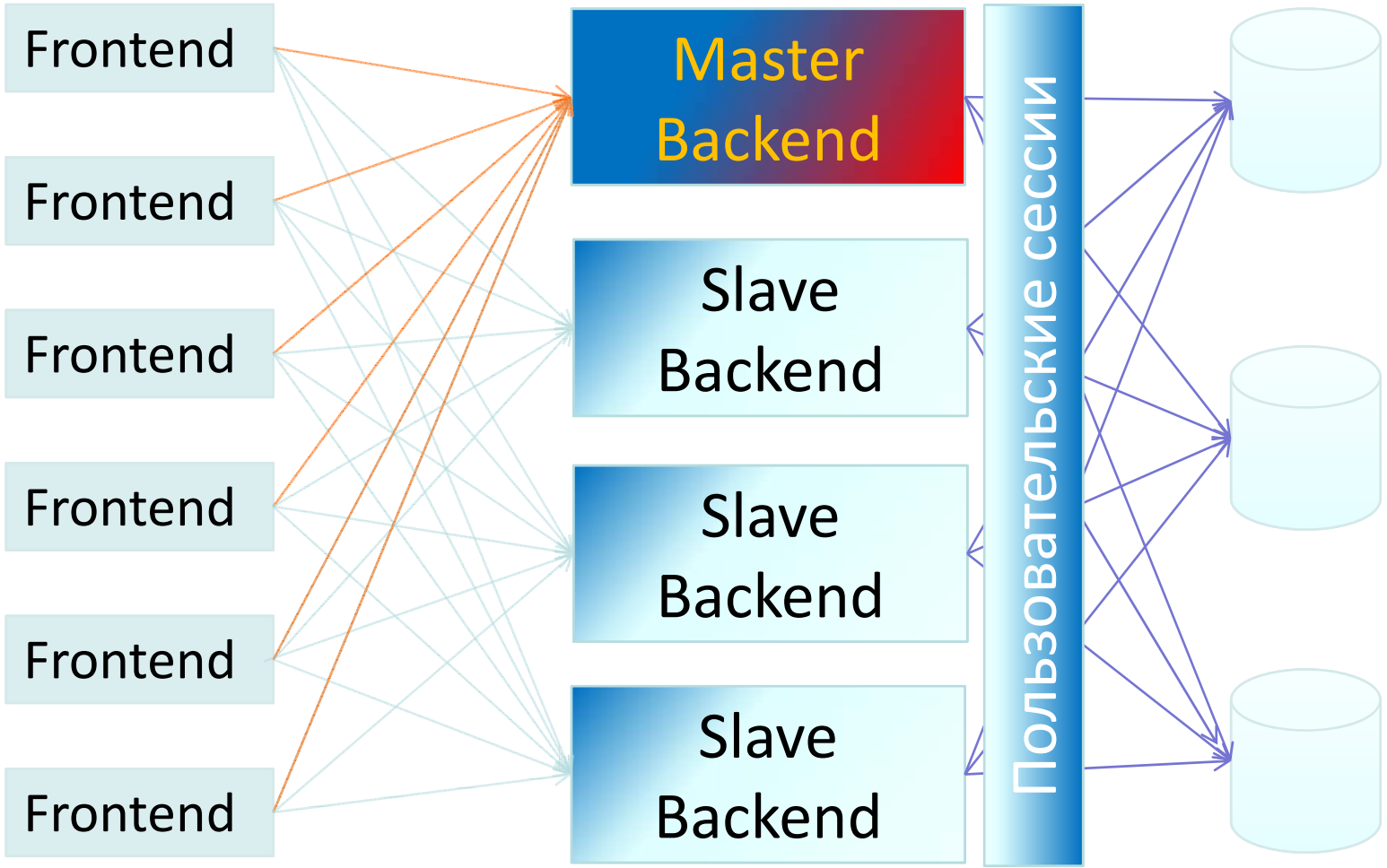
Если есть возможность делать stateless – то нужно делать stateless. Если нет – то придумывать, как синхронизировать сессии.

1W * nR, stateless



Яndex деньги

1W * nR



Яndex деньги

Кэширование

Где живет

Локально, внутри ДЦ, между ДЦ

Блокировки

С транзакциями, атомарный, без блокировок

Надежность

Пять девяток, реплицированный, не думаем

Как очищать

По времени использования, по переполнению, комбинированные схемы

Синхронизация кэшей

Как передавать изменения

- Синхронно, асинхронно

Как узнавать о изменениях

- Push – инициатор изменений дергает кэш
- Poll – кэш регулярно проверяет актуальность
- CheckOnRequest – проверка версии при чтении

Что делать при изменениях

- Очищать, перечитывать, заменять

Масштабирование баз данных

Лучше базу данных не масштабировать.

Но если уж пришлось, то:

Partitioning:

Просто и дорого, часто есть прямо в «коробке»

Sharding:

Нужно делать самому

Есть неочевидные сложности с балансировкой

С архитектурой и основными проблемами разобрались, появились

ПРОБЛЕМЫ ВЫБОРА

Яndex деньги

Выбираем язык

Основные критерии:

- Надежность кода
- распространенность среди разработчиков
- масштабируемость
- готовые библиотеки

Выбор БД

Oracle

Дорого. Очень дорого. С приключениями, но круто.

DB2

Дорого и мало специалистов, но есть Express C

MySQL/Postgress

Дешево. Но с приключениями.

Прочие

А вы уверены? Вы действительно уверены?

Как показывать странички

Логика отображения прямо на клиенте

AJAX, Applets, Flash, JavaFX , Silverlight, ActiveX..

Логика отображения через шаблоны:

Smarty, CTPP, Velocity, JSP, ASP, XSLT

С различным уровнем наворотов:

JSF, Struts, Tapestry, ASP.Net, ….

Критерии выбора:

– знакомый
– простой
– масштабируемый

Я выбираю XScript ☺

Как показывать формочки

Три основных пожелания

- Возможность автоматических тестов
- Простота интеграции с сервером
- Простота разработки

Три основных выбора

- Win Only/Run anywhere
- Intranet/Internet
- Web/Standalone

Java/RCP, Java/Swing, WinForms/C#

Я обещал банальности? Сейчас самое время

БАНАЛЬНОСТИ

Яndex деньги

Про базы данных

- Insert быстрее update и гораздо лучше масштабируется
- Blobы иногда бывают полезны
- Длинные транзакции противопоказаны даже версионникам
- Генерация отчетов – не дело для оперативной системы
- БД должна быть простой

Сериализация

Существует много разных способов сериализации:

- XML
- Reflection
- Binary formats
- Через единый контейнер

Главное, не забывать соблюдать совместимость между различными версиями сериализации одного объекта.

Асинхронная обработка

Не всегда можно успеть обработать все полученные события. Но часто достаточно их только получить, обработать можно и потом.

Подробности – в докладе Яши Сироткина

Обработка ошибок

Ошибки бывают:

- Пользовательские
- Свои
- Чужие
- Устранимые силами пользователя
- Неустранимые
- Фатальные
  - Нарисованные тончайшей кистью из верблюжьей шерсти
  - Похожие издали на мух

Логи

Последний рубеж защиты данных

Первый рубеж troubleshooting

Нужно записывать все, что пришло в систему и все, что ушло из системы

Необходимы для организации тестирования

Для любителей log4* - помните о MDC

Мониторинг

Главное – он должен быть.

Оперативный:

- Nagios, Zabbix

Профилактический

- Статический анализ логов. Хотя бы ежедневный

Быть самураем

Не бывает сервисов, которые не падают.

- Надо знать заранее, кому звонить при падении сервиса
- И описание действий по подъему сервиса лучше написать заранее
- Ноутбук и карточка SkyLink стоят немного
- Если сисадмину часто звонят ночью – то точно нужен еще один сисадмин.

Яndex деньги

Код

KISS!!! (KeepItSimpleStupid)

Нужно уметь пользоваться системой контроля версий

Легче поставить еще один сервер, чем сломать голову, разбирая запутанный код

Изобретая велосипед, задумайся, зачем

А теперь еще несколько банальностей о самом главном, о процессе

ОРГАНИЗАЦИЯ

Главное при разработке

Кадры решают все.

Любое архитектурное решение или сделанный выбор в первую очередь зависит от имеющихся людей.

Роли в команде

- Системное администрирование
- Разработка БД
- Общение с заказчиком
- Общение с начальством
- Тестирование
- Удержание архитектурной рамки
- Интеграция
- И, наконец, разработчик

Яndex деньги

Идеальная команда

“Бойцовый Кот есть боевая единица сама в себе, способная справиться с любой мыслимой и немыслимой неожиданностью, так?”

- Самостоятельность
- Здравый смысл
- Свободная коммуникация
- Делегирование ответственности и полномочий
- Взаимное доверие

Какие документы нужны

- Описание архитектуры
- Требования к системе
- Запросы на изменения
- Недельный план
- План итерации

А какие документы желательны

- Коллекция наработанных Best Practices
- Всевозможные описания «how to» для популярных задач
- Список всех допущенных грязных хаков
- Мечты по рефакторингу

Инструменты

- База знаний - Wiki (Confluence)
- Учет работ и ошибок – Jira, Bugzilla
- Оперативное планирование - ProjectCards, GreenHopper
- Контроль версий – Subversion

Яndex деньги

Методологии

Каждому проекту – свою методологию!

- Нужен общий проект системы
- Agile-методологии на грани применимости
  - Вряд ли XP, но вполне SCRUM
- Не все практики легко применимы
- Рефакторинг необходим
- Предвидеть пожелания заказчика на один шаг

Тестирование

- Автоматическая сборка
- Покомпонентное автоматическое тестирование
- Интеграционное автоматическое тестирование
- Автоматическое нагрузочное тестирование

Ну вот и все

Все события и цифры в докладе вымышлены, все совпадения случайны.

При подготовке доклада ни один сервер не пострадал.

Спасибо за внимание, с вопросами и предложениями –

PhillipDelgiado@yandex.ru

Яndex деньги