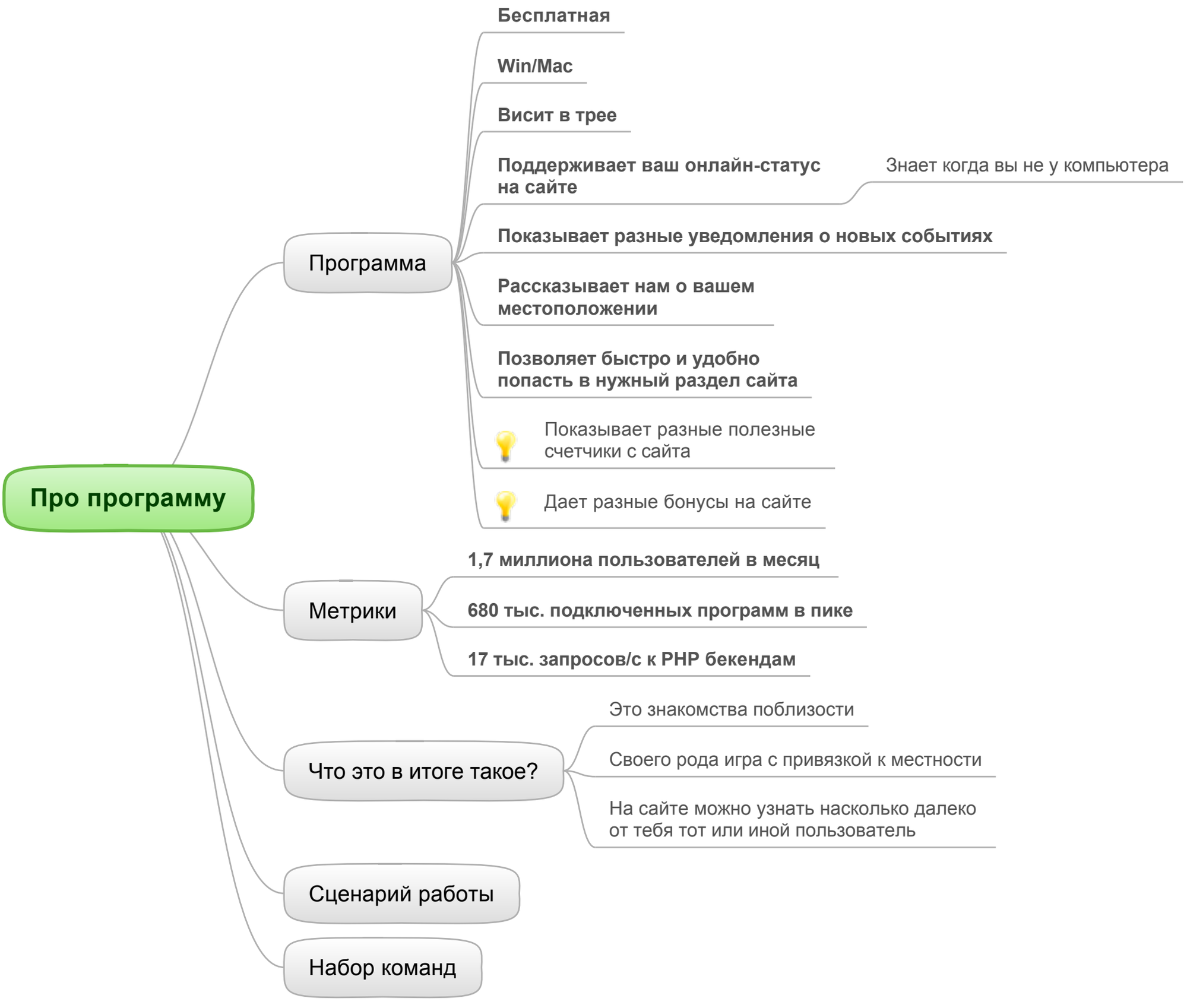
Полмиллиона юзеров в онлайне без падений:
оптимизация высоконагруженного server-side API десктопного приложения


Про Баду
Для юзеров
Лого
Концепция
Социальная сеть для знакомств с новыми людьми
Метрики
В Top-200 alexa с 2007 года
Сейчас на 130-м месте в Alex'e
115 млн пользователей
10 млн пользователей заходят на сайт в течение дня
1,5 млн фотографий загружается каждый день
Пользователей/разработчиков > 1 000 000
Быстрорастущая компания
В январе перешагнули 100 млн
Для разработчиков
Метрики
30 тыс. запросов/с к PHP бекендам в пике
Мы используем
MySQL, PHP, C/C++, Linux, nginx, PHP_FPM, memcached и много своего


Про программу
Программа
Бесплатная
Win/Mac
Висит в трее
Поддерживает ваш онлайн-статус на сайте
Знает когда вы не у компьютера
Показывает разные уведомления о новых событиях
Рассказывает нам о вашем местоположении
Позволяет быстро и удобно попасть в нужный раздел сайта
Показывает разные полезные счетчики с сайта
Дает разные бонусы на сайте
Метрики
1,7 миллиона пользователей в месяц
680 тыс. подключенных программ в пике
17 тыс. запросов/с к PHP бекендам
Что это в итоге такое?
Это знакомства поблизости
Своего рода игра с привязкой к местности
На сайте можно узнать насколько далеко от тебя тот или иной пользователь
Сценарий работы
Набор команд

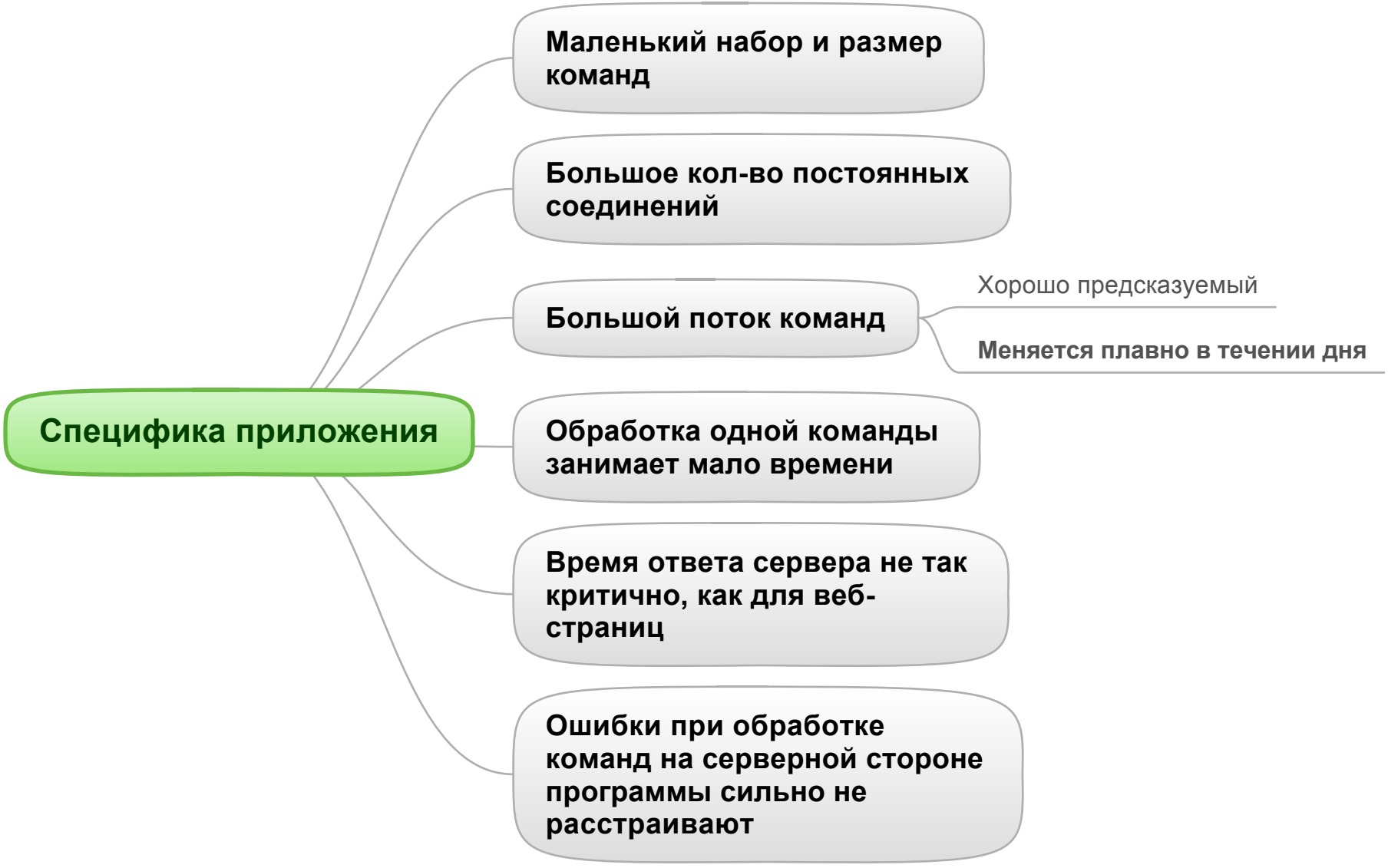

Сценарий работы
Соединяемся с главным фронтендом
Он отправляет нас на нужный фронтенд
Создаем/восстанавливаем сессию
Получаем настройки, перевод и меню
Выполняем в цикле
Показываем уведомления
Посылаем данные о wi-fi и скрин-сейвере

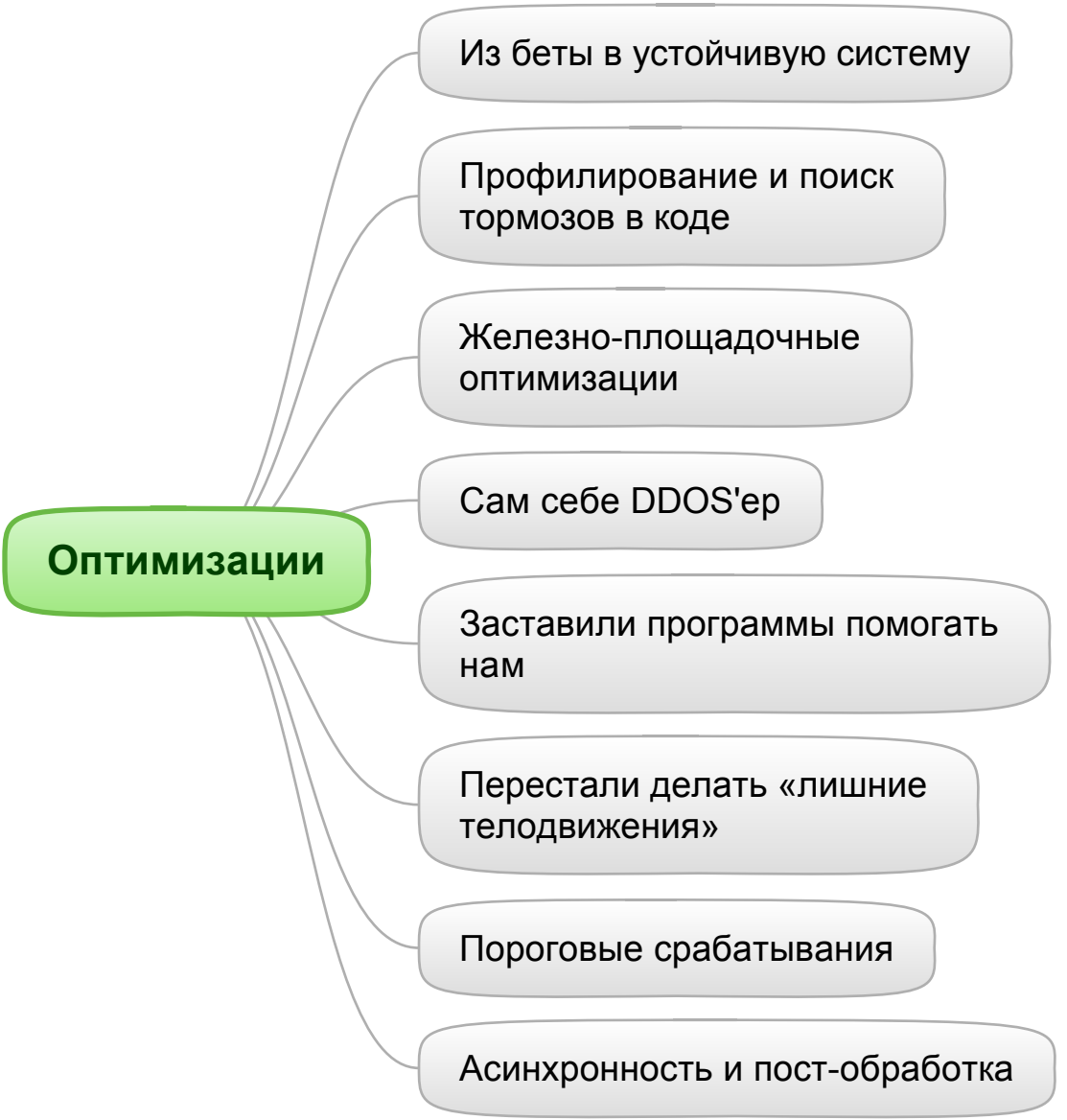

Набор команд
Из программы
Создание/Восстановление сессии
Авторизация
Данные о wi-fi сетях, работе скрин-сейвера
В программу
Ваш id сессии, язык и статус авторизации
Настройки, перевод, меню
Уведомления

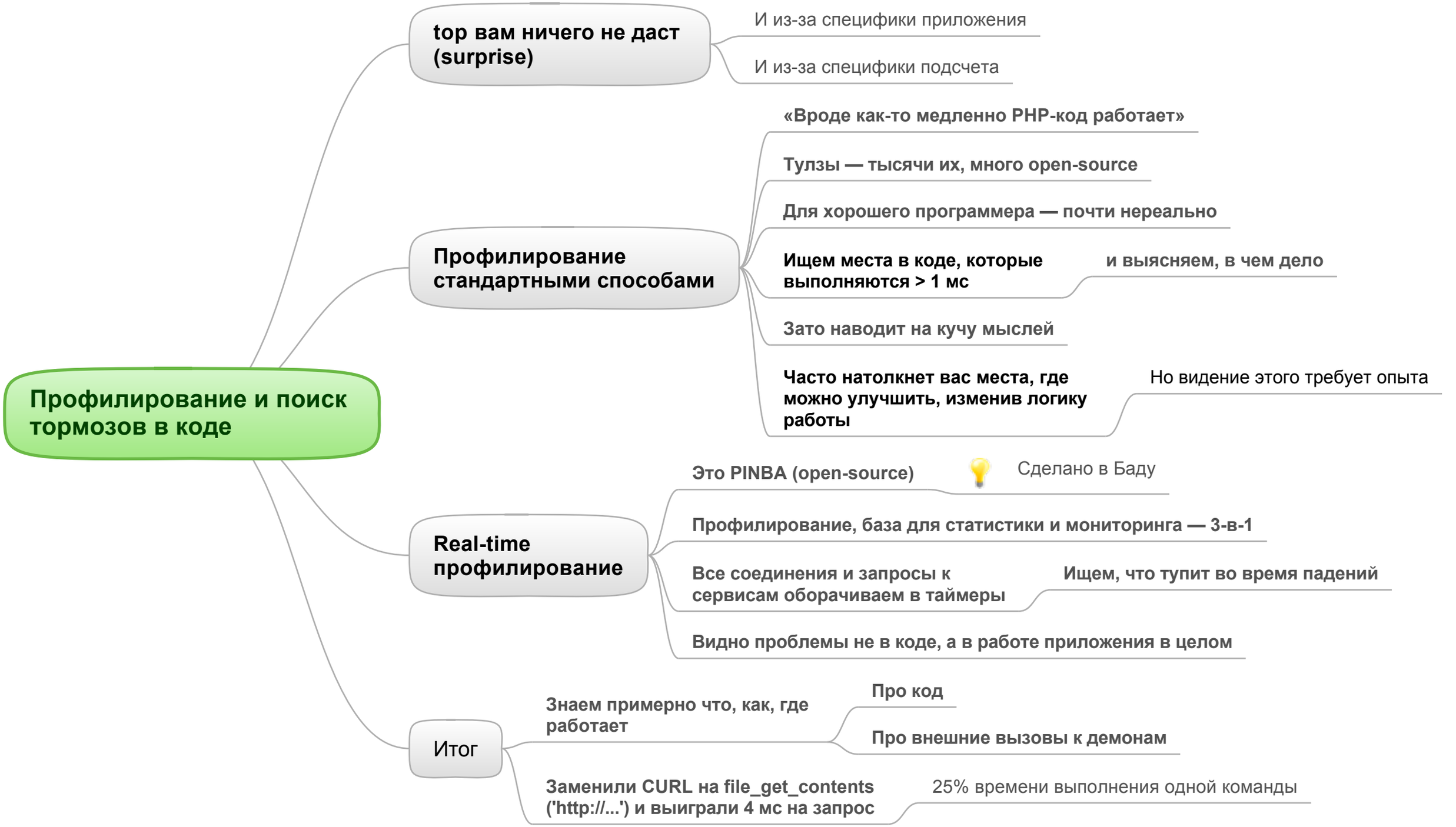
Server-side архитектура

В принципе сильно похоже на обычный веб-сайт


Изначальные архитектурные решения
Протокол асинхронный
Не требует ответа на большинство команд
Для команд, требующих ответа, есть тайм-аут на ответ
Сработал тайм-аут — переподключаемся
Как можно более простые протокол и логика программ
Сложная логика на сервере
При ошибке на сервере не пытаемся восстановиться, а прерываем обработку команды
Один exception handler
Посылаем в программу команду ошибки
При ошибке программа переподключается
Нам не нужна 100% синхронность данных
Переподключение сначала скрыто
Что не можем показать — не показываем


Специфика приложения
Маленький набор и размер команд
Большое кол-во постоянных соединений
Большой поток команд
Хорошо предсказуемый
Меняется плавно в течении дня
Обработка одной команды занимает мало времени
Время ответа сервера не так критично, как для веб-страниц
Ошибки при обработке команд на серверной стороне программы сильно не расстраивают


Оптимизации
Из беты в устойчивую систему
Профилирование и поиск тормозов в коде
Железно-площадочные оптимизации
Сам себе DDOS'ер
Заставили программы помогать нам
Перестали делать «лишние телодвижения»
Пороговые срабатывания
Асинхронность и пост-обработка

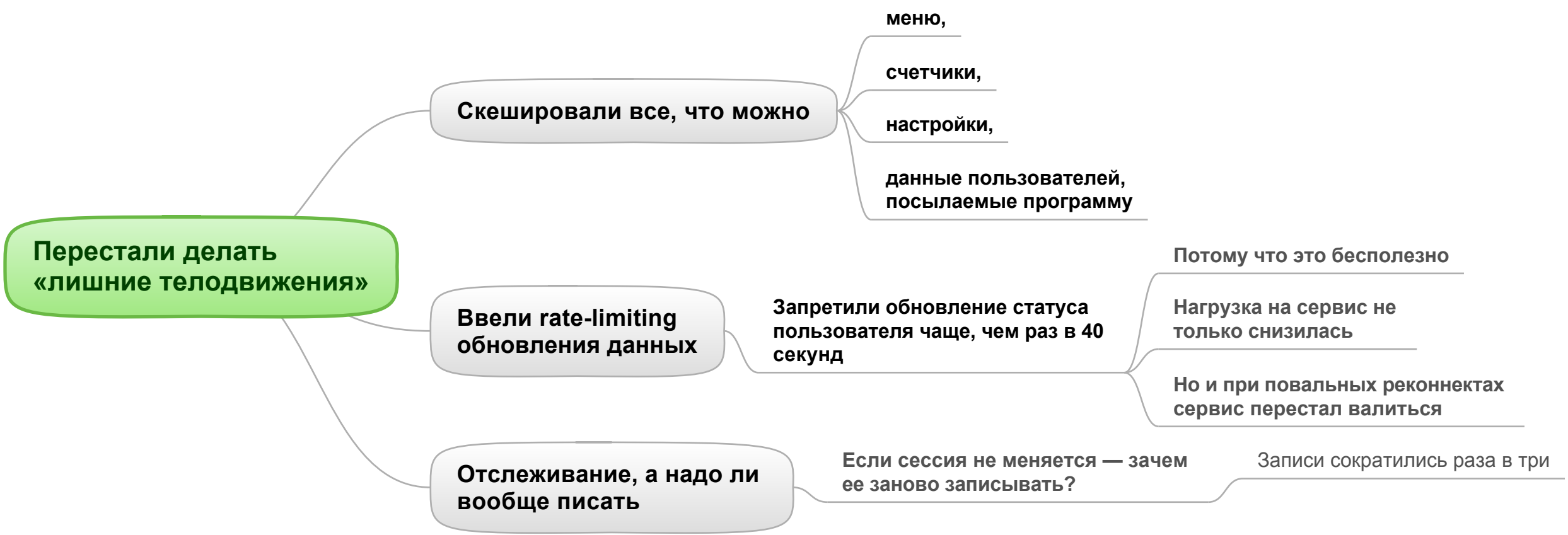

Профилирование и поиск тормозов в коде
top вам ничего не даст (surprise)
И из-за специфики приложения
И из-за специфики подсчета
Профилирование стандартными способами
«Вроде как-то медленно PHP-код работает»
Тулзы — тысячи их, много open-source
Для хорошего программера — почти нереально
Ищем места в коде, которые выполняются > 1 мс
и выясняем, в чем дело
Зато наводит на кучу мыслей
Часто натолкнет вас места, где можно улучшить, изменив логику работы
Но видение этого требует опыта
Real-time профилирование
Это PINBA (open-source)
Сделано в Баду
Профилирование, база для статистики и мониторинга — 3-в-1
Все соединения и запросы к сервисам оборачиваем в таймеры
Ищем, что тупит во время падений
Видно проблемы не в коде, а в работе приложения в целом
Итог
Знаем примерно что, как, где работает
Про код
Про внешние вызовы к демонам
Заменили CURL на file_get_contents ('http://...') и выиграли 4 мс на запрос
25% времени выполнения одной команды

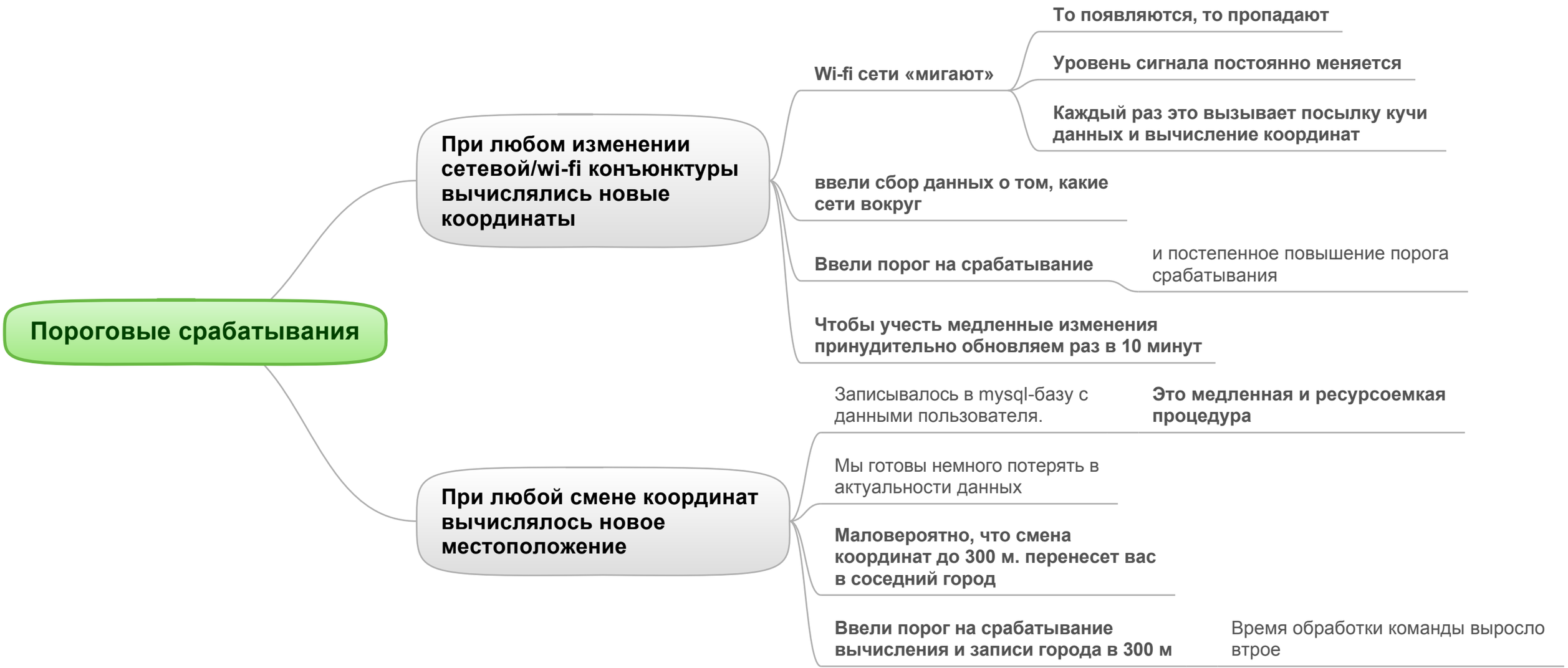
Железно-площадочные оптимизации

- **на серверах, получающих основную массу коннектов задействовали все 4 сетевые карты**
  - по 200 тыс. одновременных коннектов на сервер
  - Лучше производительность
- **отдельный мемкеш**
  - Минимум конкуренции
  - **меньше зависишь от соседей — спокойнее живешь**
- Меняли способ хранения сессий
  - **переехали с MemCacheDB** — Не использовать никогда
  - **на Redis**
    - оказалось так себе
    - Большие тормоза при дампах
    - 💡 Append-only режима не было
  - **Затем на MySQL+HandlerSocket**
  - Вывод:
    - **read и write-оптимизированные системы**
    - все БД нужно тестировать под большой нагрузкой
- Долгие тайм-ауты при торможении площадки занимают ресурсы
  - **Вместо оптимистичных тайм-аутов ответов демонов устанавливаем жесткие (0,5—1с)** — Больше свободных процессов php-fpm
- **Демоны, ответ которых нам все-таки важен (например, загрузку сессий программ, без которых ни одну команду не обработать)**
  - Есть сервисы, которые «забывают» вам ответить/теряются пакеты — memcached — отличный пример
  - **пытаемся опрашивать несколько раз с паузой между попытками**
    - умеем бороться с единичными ошибками без посылки ошибок в программу
    - Снижает кол-во показываемых ошибок
    - Снижает кол-во реконнектов
- **Коннект — дорогая операция**
  - **Где можем, используем pconnect**
  - 💡 Какая была интересная история, как всех не туда перезалогинило
  - Вывод: отличная шутка, большой выигрыш — 💡 но надо «уметь правильно готовить»


Сам себе DDOS'ер
При проблемах на площадке ложилось все резко падало и плохо поднималось
Мы устраивали сами себе DDOS сервиса
программы отваливались и переподсоединялись скопом за короткий промежуток времени
Даже если все сервис могли работать нормально, бесконечно поднималось и падало волнами
Положительная обратная связь
Можно ли ее избежать?
Увеличивание пауз посылки команд при ошибках
при получении ошибки
которая означает либо ошибку при выполнении команды либо таймаут backed'a
программа притормаживает посылку следующей команды
при получении нормальных ответов притормаживание плавно снимается
То же самое для реконнектов
Если при сбое все программы отваливаются
то очень скоро попытки подключения пойдут со значительными интервалами
что даст площадке свободные ресурсы
для восстановления производительности
Улучшили алгоритм реконнекта
вместо схемы «сходи на роутер-сервер, потом иди куда он скажет»
переподключение стало отличаться от первого подключения после запуска программы
При переподключении мы пытаемся несколько раз подключиться туда, где мы были
и только после того, как все попытки оказались неуспешными
идем на роутер-сервер (полный цикл реконнекта)


Заставили программы помогать нам
Для отслеживания времени последнего обновления статуса нужно это время куда-то писать
пишем в мемкеш
почему бы не заставить программу отслеживать и сообщать нам это?
Снимает большую часть нагрузки на мемкеш
Пользователь может быть не за компьютером
Нам такие пользователи неинтересны
на сайте они оффлайн
Сделали снижение частоты посылки уведомлений
Экономит серверные мощности
Медленные соединения (GPRS)
Бывало вообще не приконнекчивалась
Ввели интеллектуальное увеличение и уменьшение времени тайм-аута на ответ сервера
Меньше реконнектов, больше подсоединенных программ

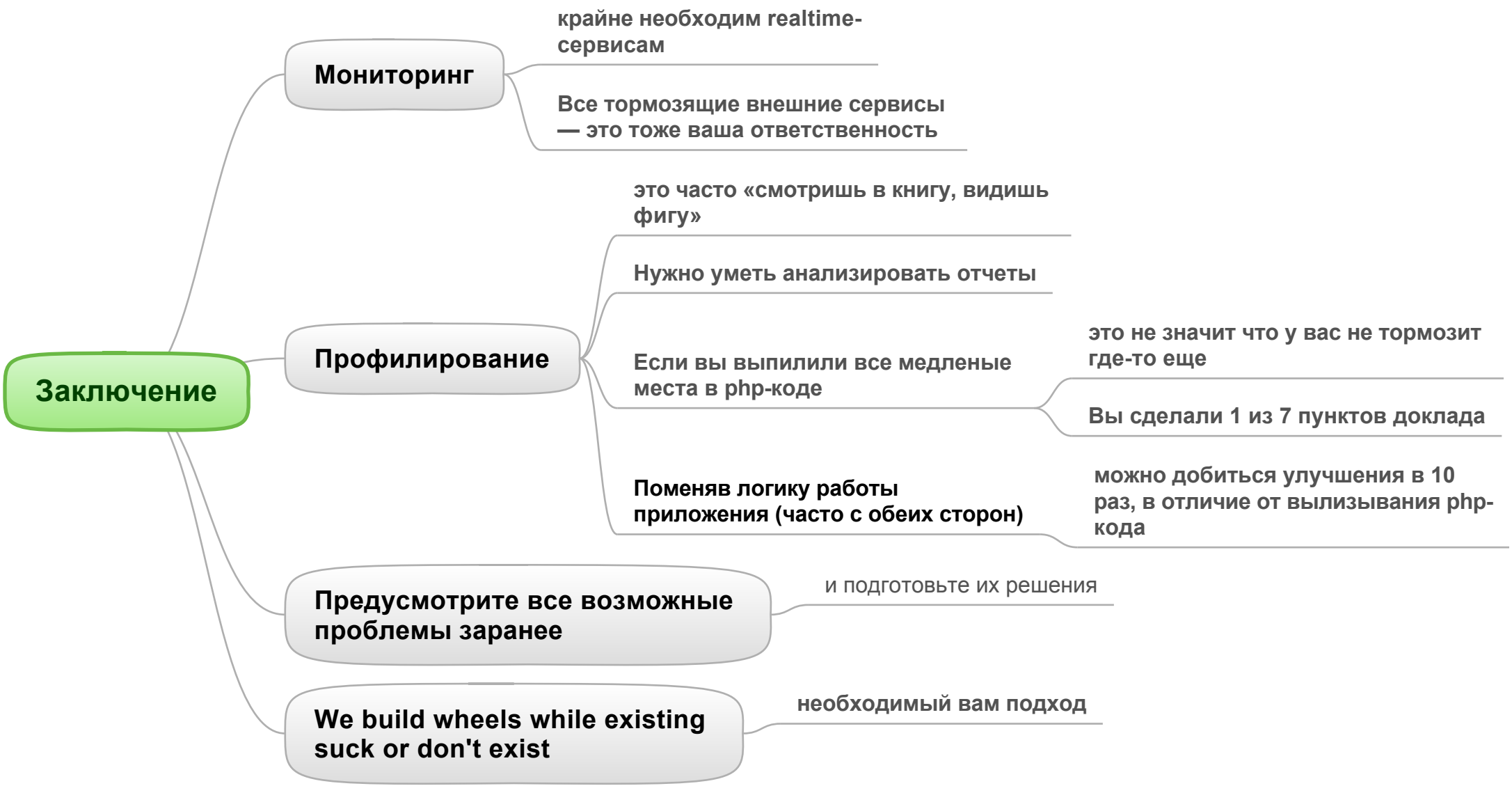

Перестали делать «лишние телодвижения»
Скешировали все, что можно
меню,
счетчики,
настройки,
данные пользователей, посылаемые программу
Ввели rate-limiting обновления данных
Запретили обновление статуса пользователя чаще, чем раз в 40 секунд
Потому что это бесполезно
Нагрузка на сервис не только снизилась
Но и при повальных реконнектах сервис перестал валиться
Отслеживание, а надо ли вообще писать
Если сессия не меняется — зачем ее заново записывать?
Записи сократились раза в три


Пороговые срабатывания
При любом изменении сетевой/wi-fi конъюнктуры вычислялись новые координаты
Wi-fi сети «мигают»
То появляются, то пропадают
Уровень сигнала постоянно меняется
Каждый раз это вызывает посылку кучи данных и вычисление координат
ввели сбор данных о том, какие сети вокруг
Ввели порог на срабатывание
и постепенное повышение порога срабатывания
Чтобы учесть медленные изменения принудительно обновляем раз в 10 минут
При любой смене координат вычислялось новое местоположение
Записывалось в mysql-базу с данными пользователя.
Это медленная и ресурсоемкая процедура
Мы готовы немного потерять в актуальности данных
Маловероятно, что смена координат до 300 м. перенесет вас в соседний город
Ввели порог на срабатывание вычисления и записи города в 300 м
Время обработки команды выросло втрое


Асинхронность и пост-обработка
Ответ на команду как можно раньше
быстрее получит — меньше таймаутов
Стараемся сервисные задачи выносить из прямой обработки команд
Быстрее ответ
Меньше ошибок валится в программы
Пакетная обработка всегда быстрее
Синхронизация использования ресурсов
Многие сервисные задачи на стороне сервера можно выполнять без привязки к конкретному времени
Если несколько таких задач используют один ресурс
почему бы не синхронизировать их, уменьшив нагрузку на этот ресурс?
Например все вместе раз в сутки для одного user_id
Ввели временное окно на выполнение каждой сервисной задачи
Если начала выполняться сервисная задача, которая пишет в сессию
автоматически запускаются все остальные, которые тоже будут писать в сессию
Если задача так и не была выполнена, а временное окно истекает
запускаем ее принудительно
Снизило кол-во записей сессий на 20%

Суровая реальность

- От падений **никто** не защищен
  - Limited CPU
  - Ограниченная производительность всех внешних сервисов
  - Ограниченное кол-во одновременно обрабатываемых запросов
    - то есть процессов php-fpm
- Мониторинг, мониторинг и еще раз мониторинг
  - на всех возможных уровнях
    - забыли поставить на мониторинг всего один сервер
  - с профилированием
  - **realtime**
    - Не график с отметкой раз в 5—15 минут, а **в реальном времени**
    - Вы работаете в совершенно другой среде, в отличие от веба
      - Много SPOF
      - Одна ошибка и вы заваливаете сами себя
      - Большая взаимозависимость между отдельными элементами системы (командами, внеш. сервисами)
      - Вы должны реагировать быстро
      - и видеть результат своих действий сразу
      - мониторинг должен быть сразу с профилировкой
      - PINBA, да
- Failover
  - выкладка старого кода
    - ручная
    - автоматическая
  - выключение части функционала
    - Ручное точно необходимо
  - выпадение отдельного сервера из кластера и возврат в строй


Заключение
Мониторинг
крайне необходим realtime-сервисам
Все тормозящие внешние сервисы — это тоже ваша ответственность
Профилирование
это часто «смотришь в книгу, видишь фигу»
Нужно уметь анализировать отчеты
Если вы выпилили все медленые места в php-коде
это не значит что у вас не тормозит где-то еще
Вы сделали 1 из 7 пунктов доклада
Поменяв логику работы приложения (часто с обеих сторон)
можно добиться улучшения в 10 раз, в отличие от вылизывания php-кода
Предусмотрите все возможные проблемы заранее
и подготовьте их решения
We build wheels while existing suck or don't exist
необходимый вам подход