Разработка высоконагруженных проектов, Что нового?

Олег Бунин

Тенденции 2010 года

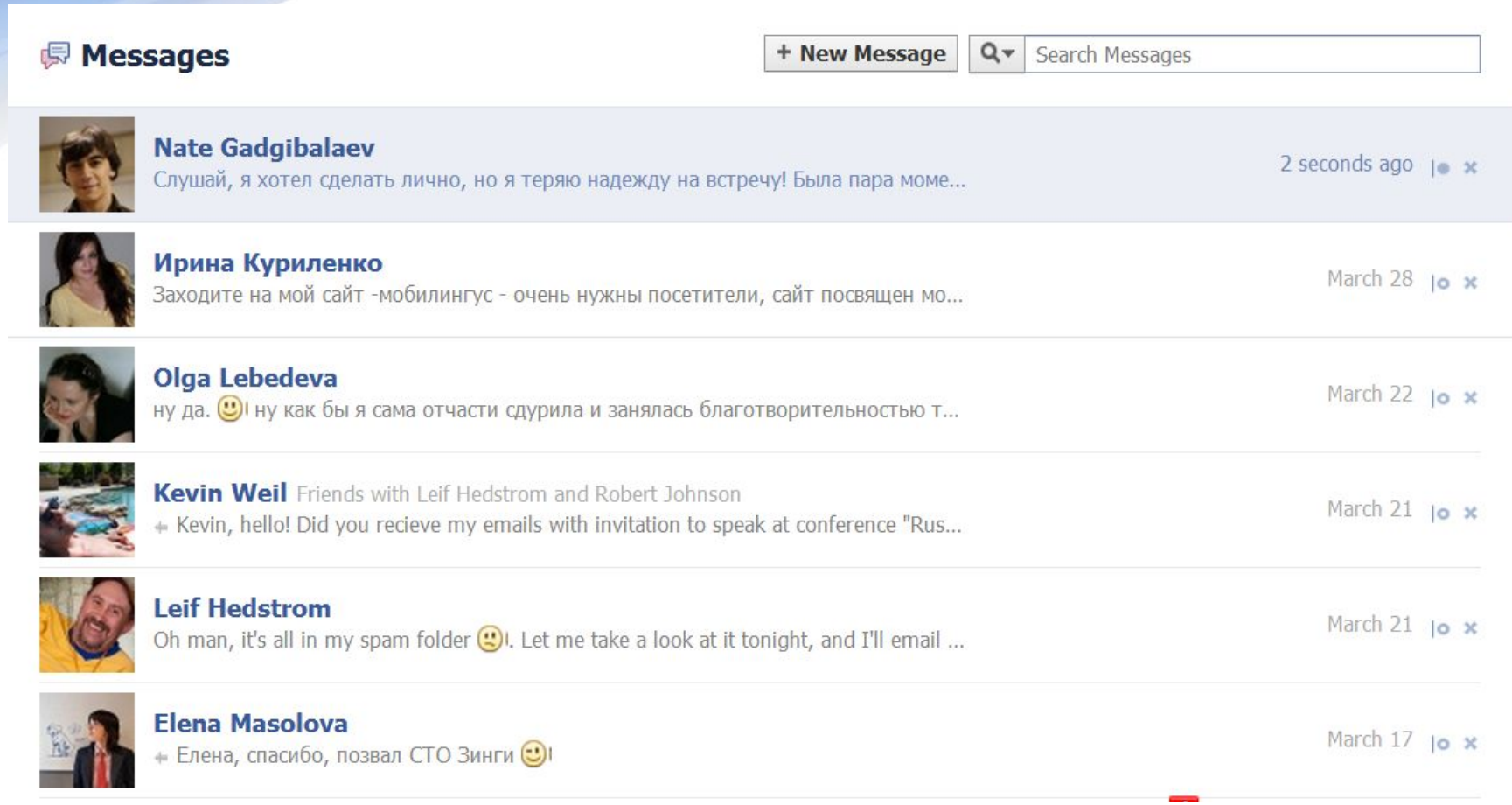
1. Облачная инфраструктура;
2. NoSQL базы данных;
3. Новые языки, Erlang;
4. nginx в железе.

Практический пример NoSQL в высоконагруженном проекте

Модели данных

- **Реляционная модель**: данные в базе данных представляют собой набор отношений;
- **Иерархическая модель**: база данных состоит из объектов с указанием отношений родитель ⇔ ребенок;
- **Сетевая модель**: база данных со структурой в виде графа;
- **Объектно-ориентированная модель**: база данных, в которой данные представлены в виде моделей объектов.

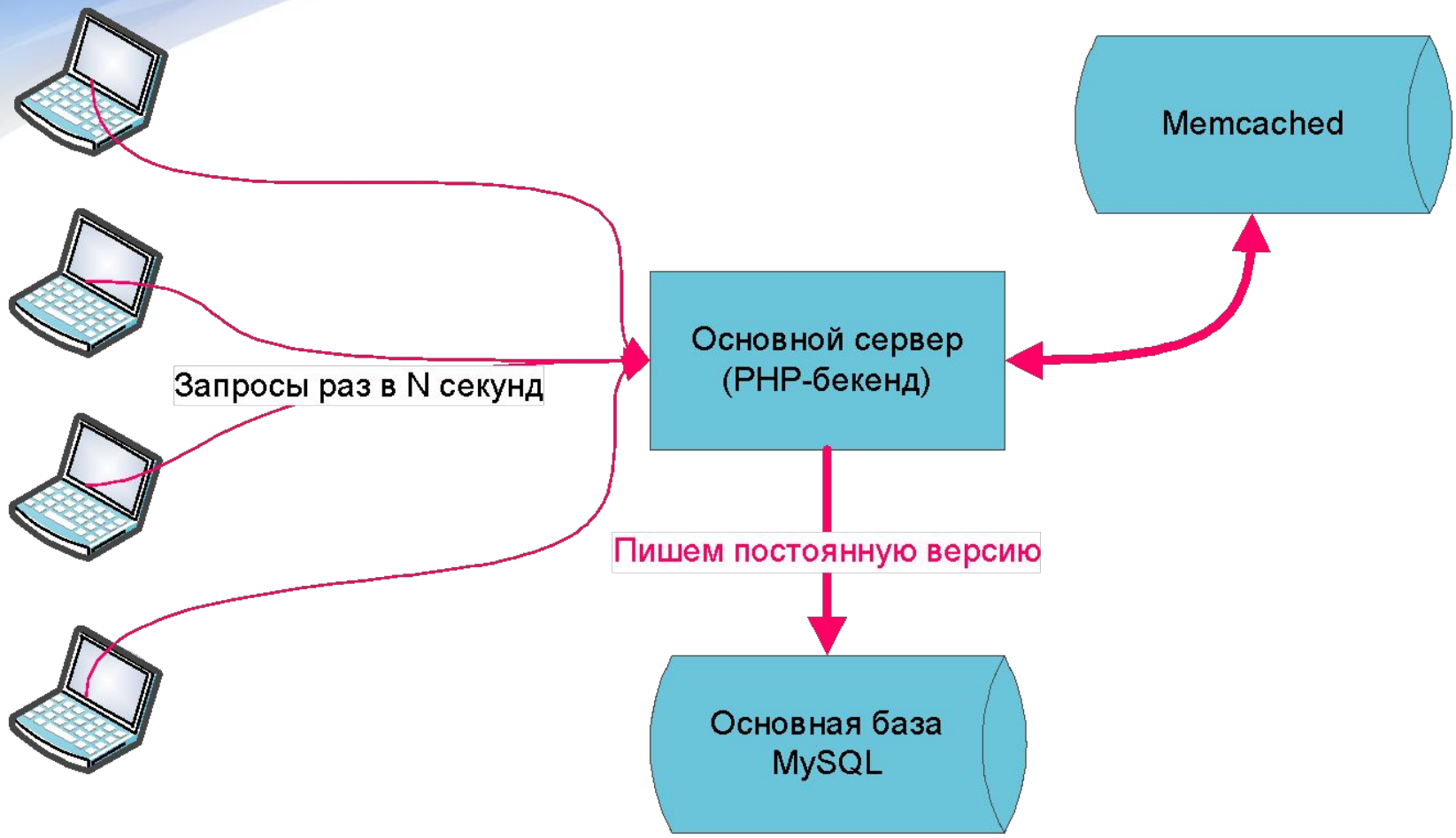
Разработка чата



Постановка задачи

- Около десяти тысяч одновременных подключений;
- Доставка сообщения в реальном времени (чат);
- Сохранение истории сообщений (переписка);
- Одновременная переписка с несколькими участниками.

Обычная схема



Вопросы к обычной схеме

Кэширование переписок

Обновление информации

Мгновенная доставка

Мгновенность доставки

Как нам сообщать клиенту о том, что пришло сообщение?

Нам нужен event-driven!

Event-driven

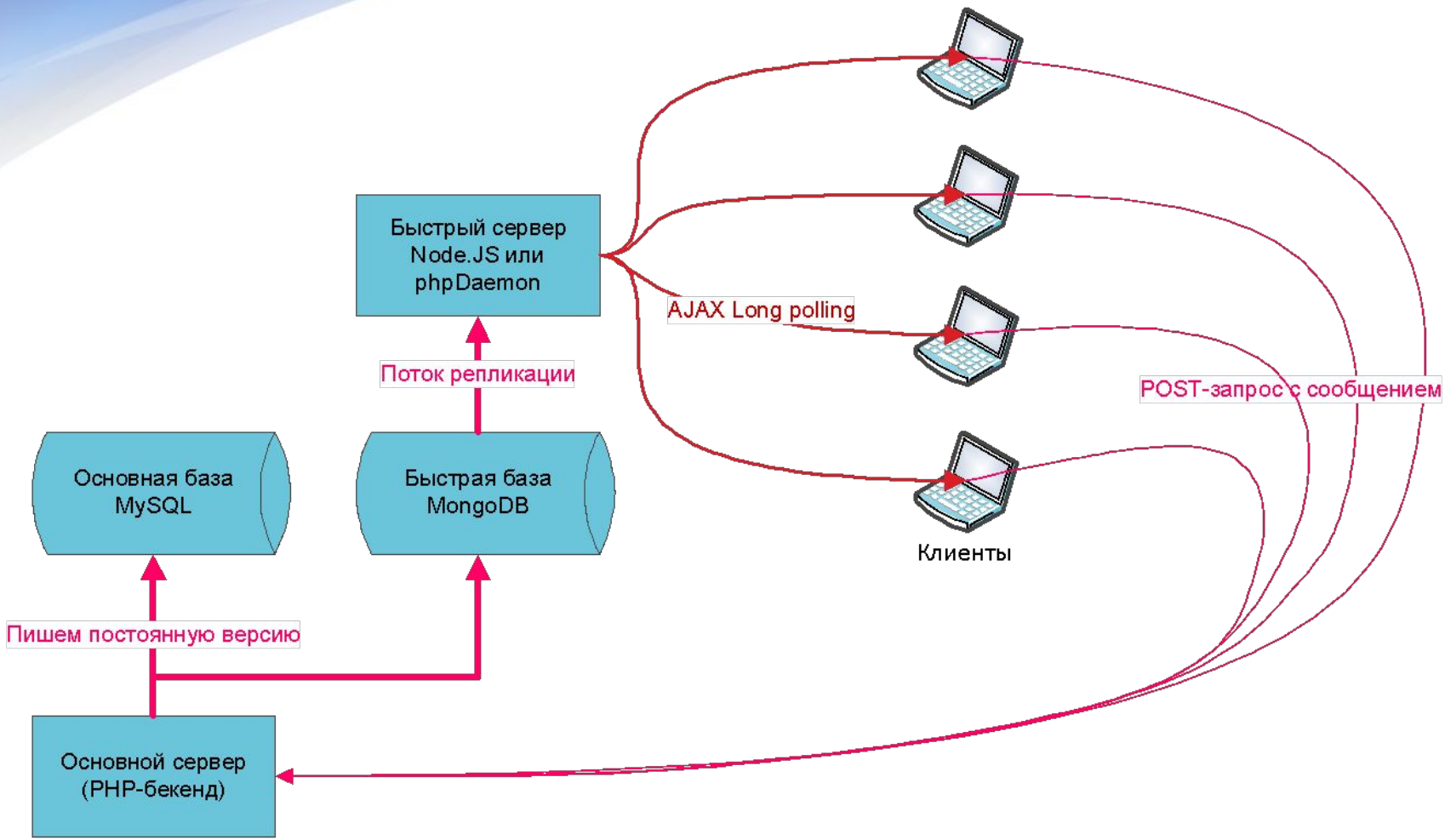
Нам нужно сообщать в реальном времени клиенту о новых сообщениях!

Мгновенную доставку можно решить некоей имитацией длинных незакрытых соединений (WebSockets, FlashSocket, AJAX Long polling, Iframe).

Event-driven чат

- Каждый новый клиент, входящий в переписку устанавливает соединение с одним из «быстрых» серверов (**phpDaemon** или **Node.js**) и от него же будет получать сообщения.
- При отправке на основной сервер нового сообщения, оно записывается в постоянную базу данных (**MySQL**), а так же в некую «быструю» базу данных (в данном случае **MongoDB**), где будет храниться только актуальная переписка, например за последние сутки.
- Master MongoDB записывает изменения в коллекцию **local.oplog.$main** для репликации.
- «Быстрые сервера» в это время ведут опрос «быстрой» базы на предмет наличия в ней новых сообщений, и, если такие есть, отправляют их тому клиенту, которому они адресованы.

Event-driven чат



Тестирование скоростей

Демон	Complete requests	Requests per second	Time per request, ms (across all concurrent requests)	Min request, ms	Max Request, ms
10 keep-alive соединений					
phpDaemon	1147	54.36	18.440	21	7116
Node.JS	1285	64	15.618	16	9064
100 keep-alive соединений					
phpDaemon	1543	75.15	13.313	22	12889
Node.JS	1284	64.12	15.765	17	13347
500 keep-alive соединений					
phpDaemon	1365	67.73	14.824	15	15174
Node.JS	1236	61.71	16.611	23	16078
1000 keep-alive соединений					
phpDaemon	1159	56.99	17.680	19	13103
Node.JS	1528	75.02	13.354	17	16785

Красным цветом отмечены «лучшие показатели»

oleg.bunin@ontico.ru