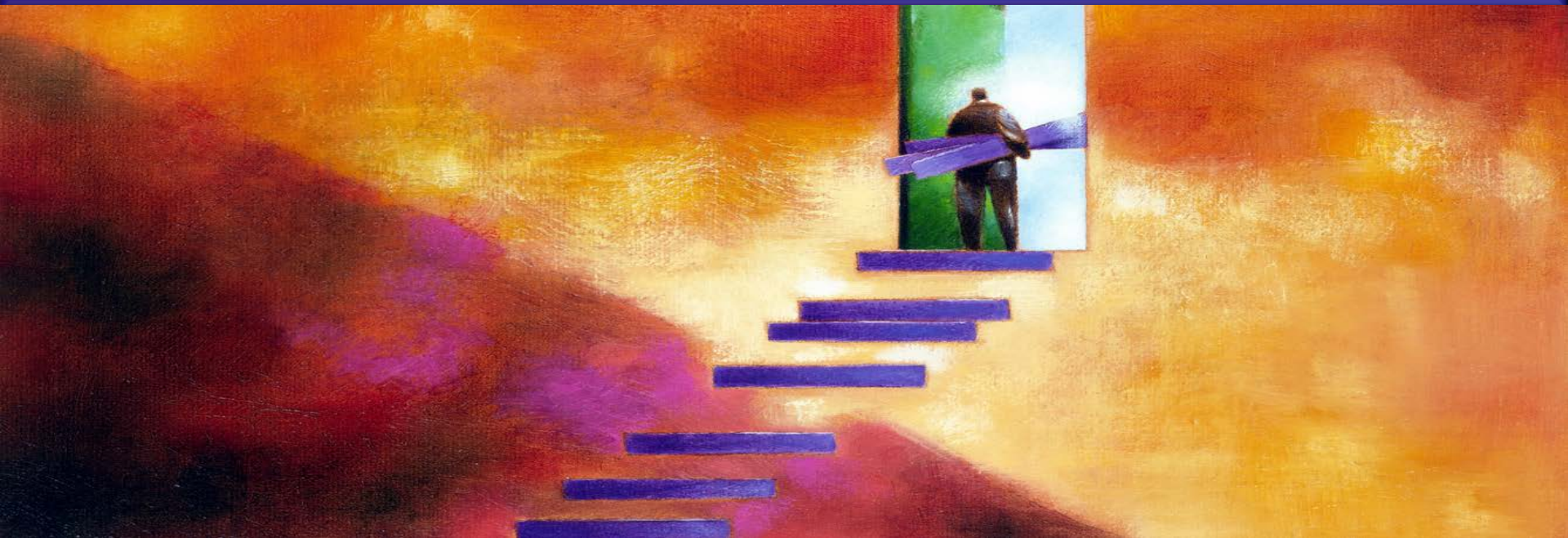


Advanced technologies for breakthrough applications



InterSystems Caché и технологии NoSQL

Оленин Олег,
технический консультант

INTERSYSTEMS

NoSQL и не только за 45 минут



- Обзор NoSQL решений
- InterSystems Caché и NoSQL. Возможно ли сравнение?
- Проект Globals. Глобалы для всех.
- Caché Extreme for Java

Причины появления NoSQL



- Web 2.0. Интернет в режиме Read Write
 - приложения используют информацию, предоставляемую и создаваемую пользователями
 - конкурентное преимущество - пользовательский контент и сообщество
 - воспроизвести функциональность приложения можно, данные и пользователей - нет
- Доставка и развертывание приложений
 - сеть как платформа, приложение как сервис
 - различные устройства доступа
 - готовность к росту
- Процесс разработки - постоянно в beta версии

Причины появления NoSQL



Extreme Transaction Processing

- Поток событий/сообщений
- Большое количество источников данных
- Большие объемы обрабатываемых данных
- Критично время
- Критична доступность данных
- Критична легкость масштабирования

Изменение требований к СУБД



- Природа данных претерпела изменения
- Взаимодействие с приложением - смена приоритетов
- СУБД не должна снижать скорость изменений, вносимых в приложения или делать ее непредсказуемой
- Готовность к большому взрыву в случае лавинообразного роста пользователей
- Постоянная стоимость за ресурс.

One Size Does Not Fit All



- Application Database - одно приложение, одна БД
 - Логика и метаданные целиком в приложении. СУБД не разделяемый ресурс
 - Контроль данных в приложении, СУБД концентрируется на инфраструктурных задачах
 - Доступ всегда через сервисы, API
- Integration Database - много приложений, одна БД
 - Логика и метаданные в БД
 - Контроль данных в СУБД, двойная работа
 - Неэффективность запросов - нет прямого доступа к структурам хранения

NoSQL. Знакомые ценности.



- Простота разработки
- Специализация для конкретного проекта
- Стоимость масштабирования
- Стоимость обработки больших объемов данных

NoSQL. Родовые черты.



- Хорошее горизонтальное масштабирование для “простых операций” на многих серверах
- Распределенное хранение данных (партиционирование) на многих серверах
- Простые API или протоколы (в сравнении с традиционными)
- Отказ от транзакционной целостности
- Эффективное использование распределенных индексов и памяти
- Динамическое добавление атрибутов к записям

Существующие решения



- Hadoop/Hbase
- Cassandra
- MongoDB
- CouchDB
- Riak
- Couchbase
- Neo4J
- SimpleDB
- Azure Table Storage
- Google App Engine Data Storage
- Mark Logic Server
- Infinite Graph
- Riak
- Berkely DB

NoSQL. Как сравнивать?



- NoSQL решения реализуют родовые черты в той или иной мере
- Классификация
 - Параллельное масштабирование
 - шардинг, репликации, партиционирование
 - Модель хранения и модель данных
 - key-value, document, extended record
 - Запросы
 - Полнота решения

NoSQL. Незакрытые вопросы.

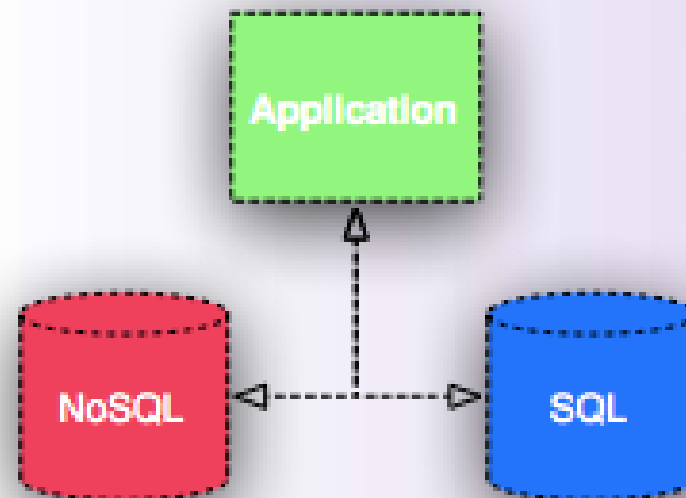


- Какую модель выбрать?
- Какой проект выбрать среди 122?
- Подойдет ли модель запросов? Будут ли поддерживаться “сложные” запросы?
- Насколько будет переносимо решение?
- Зрелость выбранной технологии?

Гибридные подходы



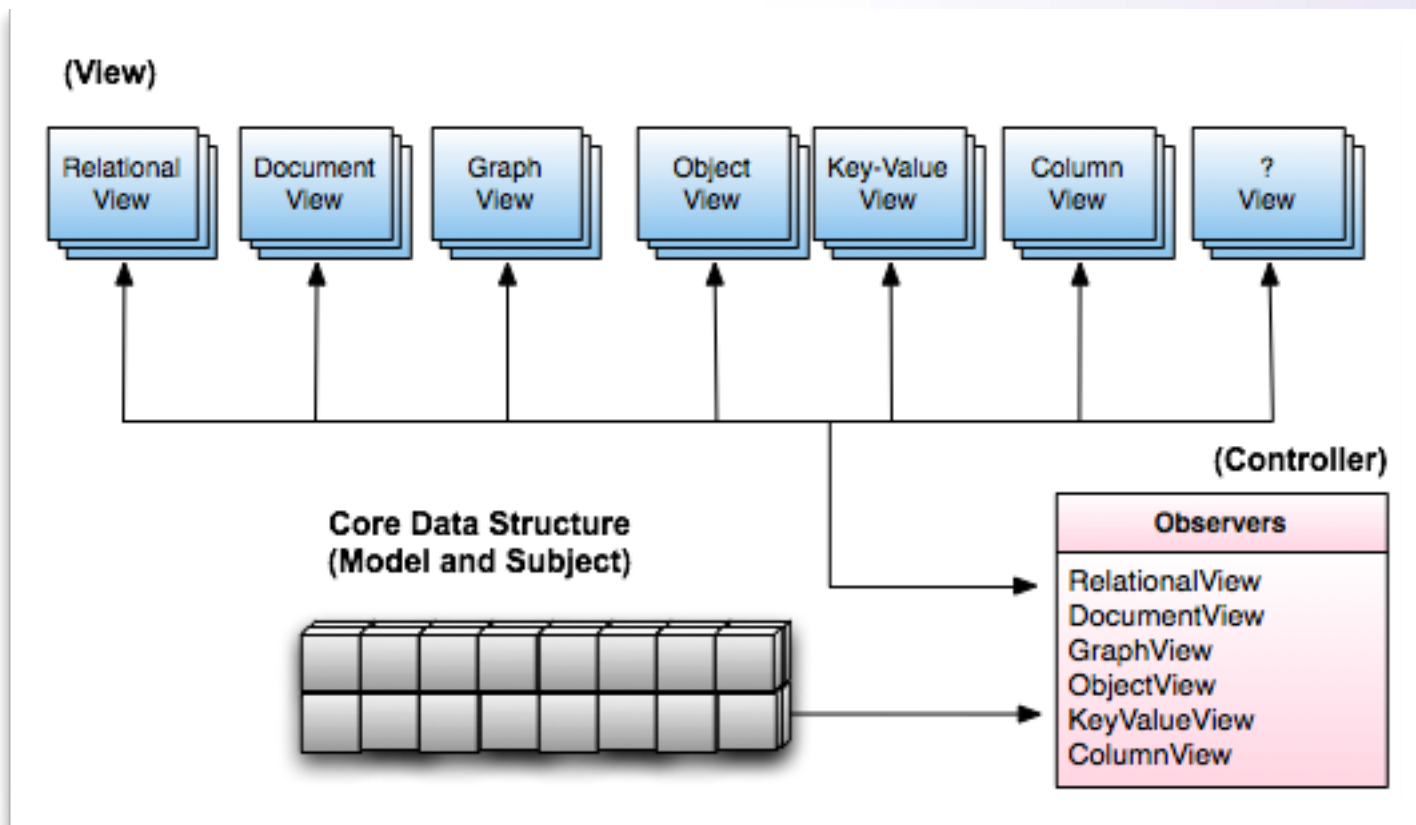
- Две технологии в одном проекте:
 - NoSQL + SQL/RDBMS
 - Распространение изменений
 - в реальном времени
 - асинхронные
 - пакетная обработка



Гибридные подходы



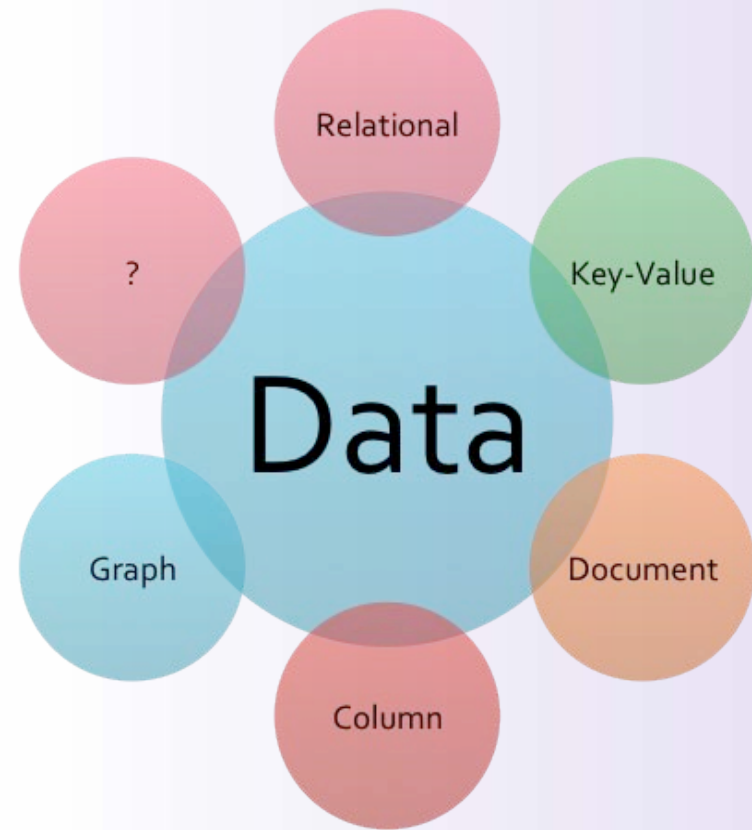
- Использование технологии двойного назначения



Caché - безопасный вход в NoSQL



- InterSystems Caché поддерживает как SQL так и NoSQL
- Зрелая технология NoSQL (20+ лет)
- API для .NET, Java, Perl, Python, ...
- Универсальная технология для специализации под конкретный проект

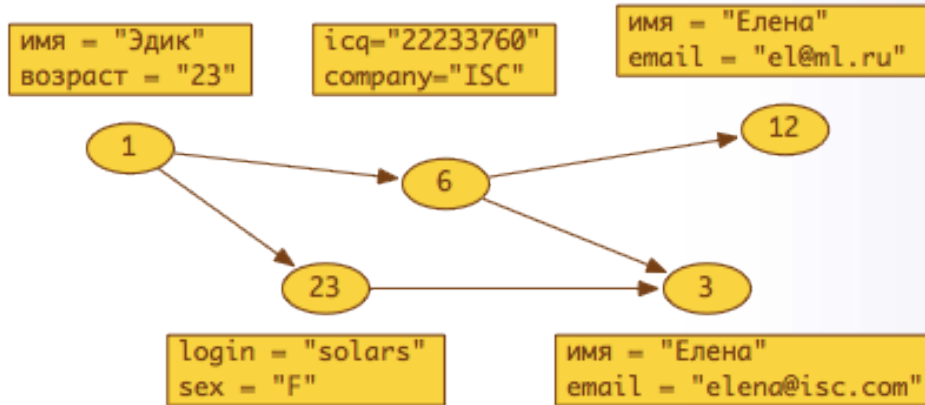


Caché для NoSQL



- Caché - готовая платформа для разработки специализированных хранилищ, устойчивых к нагрузке и масштабируемых
 - позволяет сосредоточиться разработчикам на создании СУБД со своей, характерной, моделью данных
 - при этом обеспечивает полностью все необходимые инфраструктурные элементы для такой заказной СУБД
- Caché - три вида моделей представления
 - Прямой - ассоциативные массивы (глобалы) и узлы
 - Объектный - классы и объекты
 - Реляционный - таблицы и записи
 - Комбинированный и ваш собственный

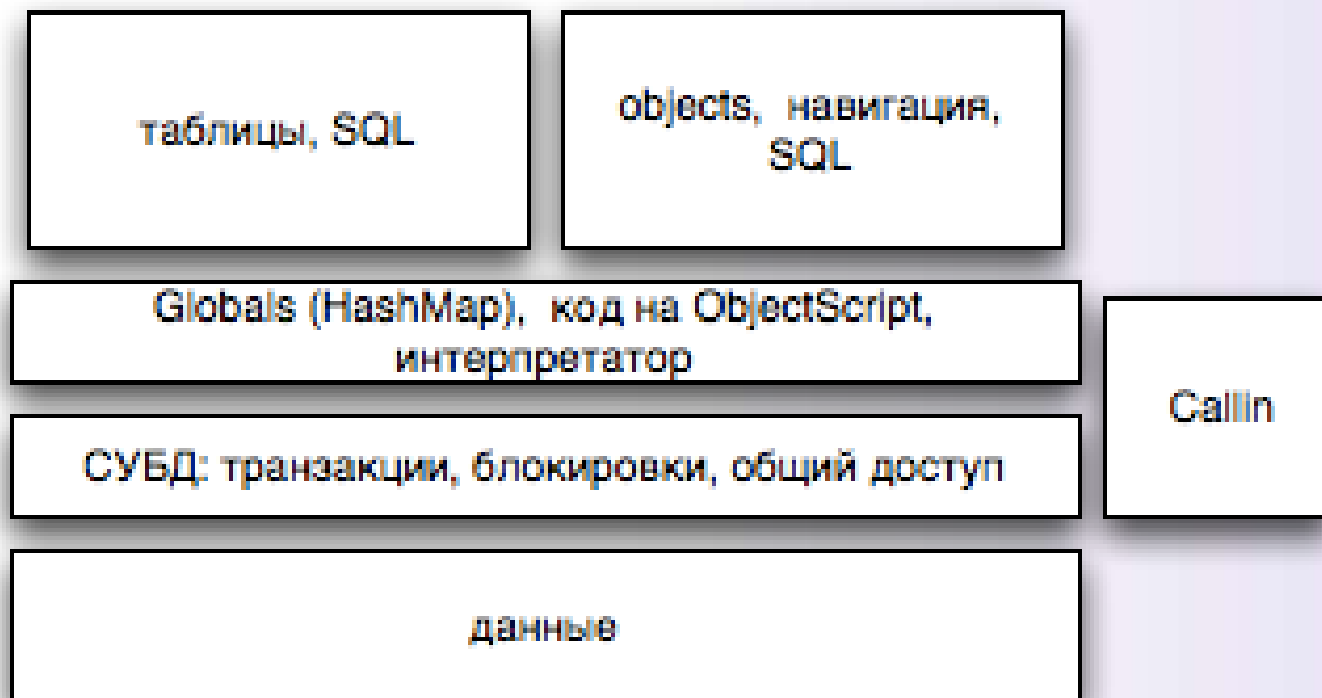
NoSQL модели на глобалах



Представление и сохранение графа и объектов

| Cache | JSON |
|---|--|
| <pre>s ^node(1,"имя") = "Эдик" s ^node(1,"возраст") = 23 s ^node(23,"login")="solars" s ^node(23,"sex")="F" Варианты описания связей: s ^node(1,"link",23) = "земляк" или s ^node(1,"link",23,"login")="solars" s ^node(1,"link",23,"sex")="F" или s ^node(1,"links") = "23,6"</pre> | <pre>{ "node": { "id": 1, "имя": "Эдик", "возраст": 23, "node": { "id": "23", "login": "solars", "sex": "F" } } }</pre> |

Архитектура. Модель хранения.

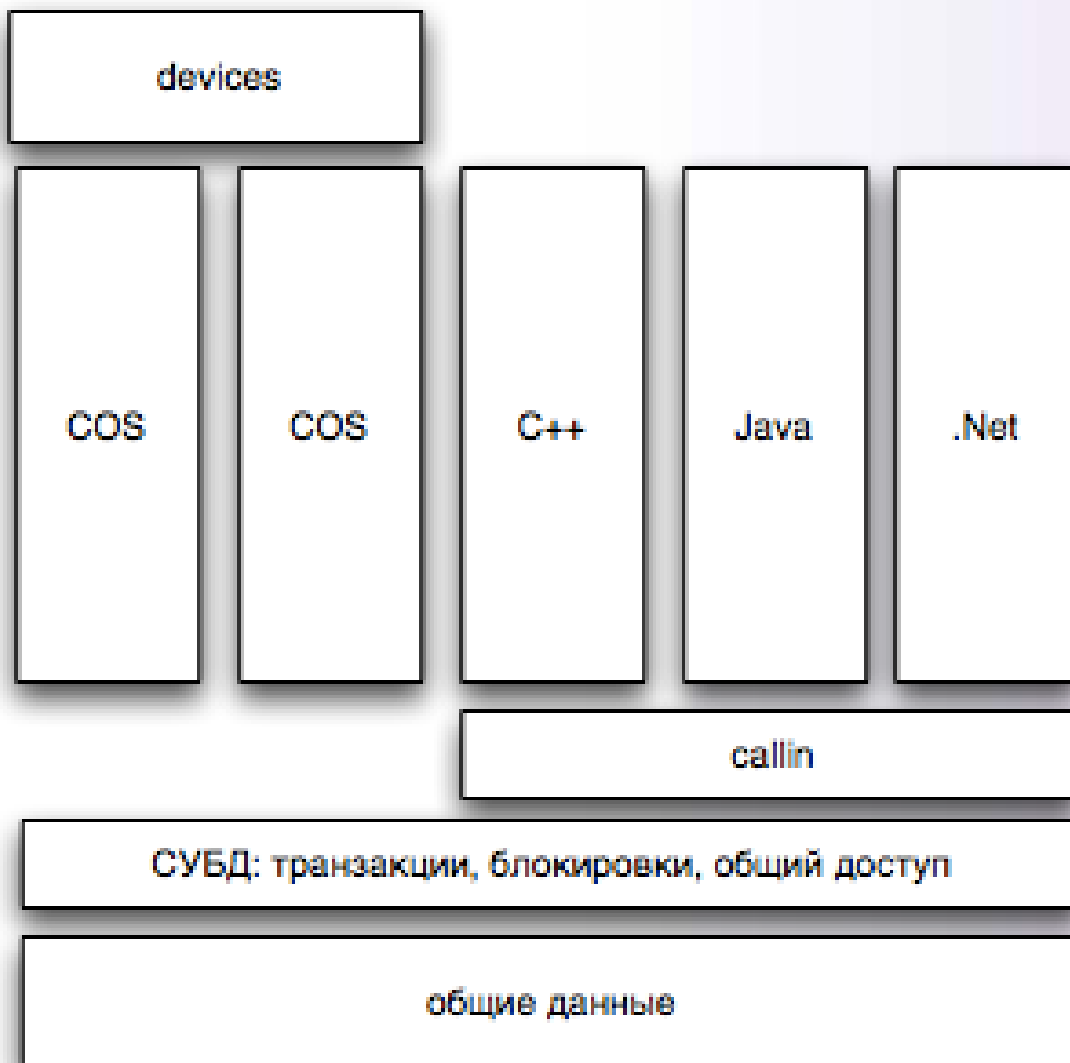


Просто скорость тоже важна



- European Space Agency > 70 000 операций в секунду
- 12% обработки мировых продаж обыкновенных акций происходит с помощью InterSystems Caché
- скорость установки - 15 минут
- скорость настройки кластера - полчаса вместе с тестированием

Архитектура. Модель доступа.

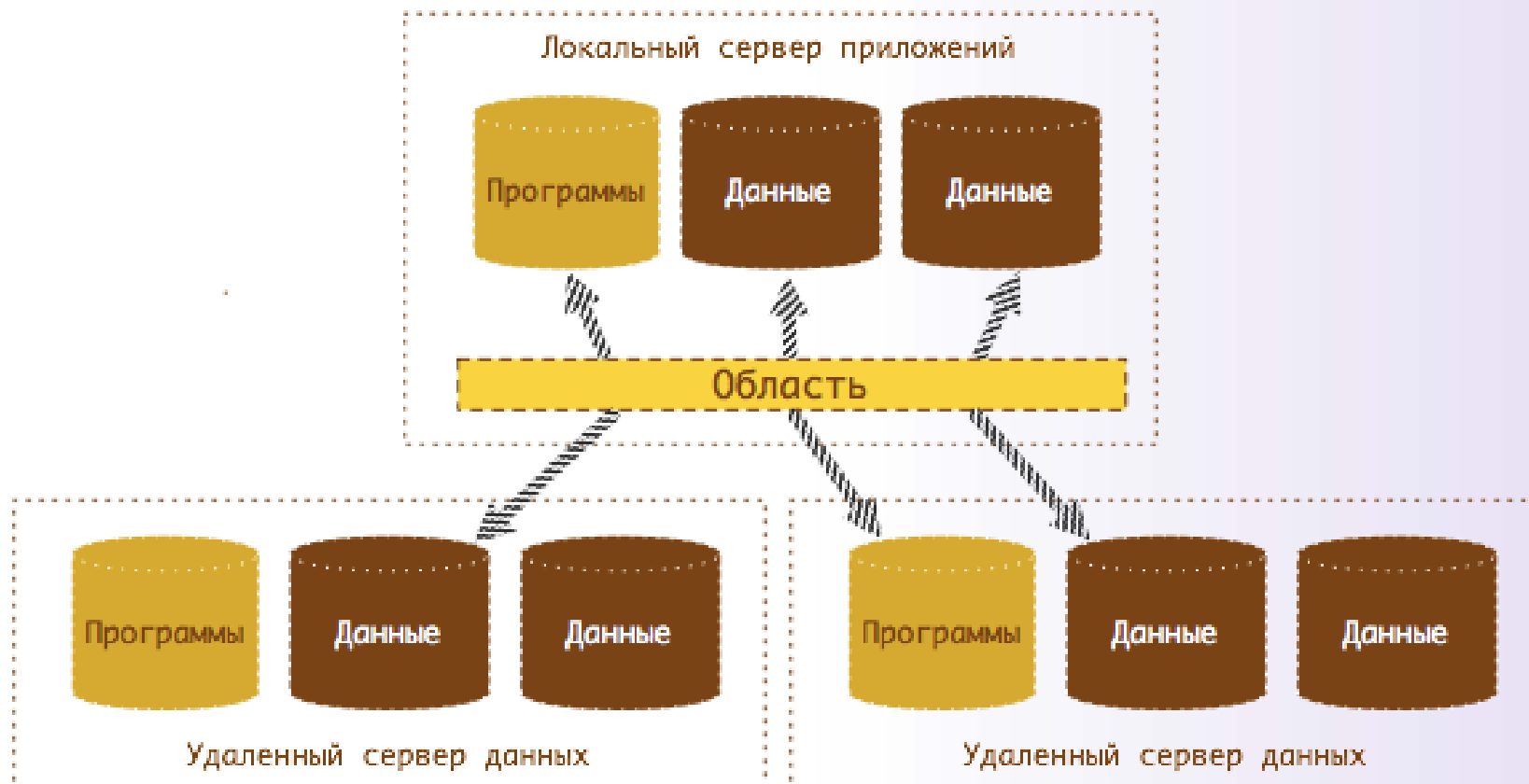


Построение распределенных баз

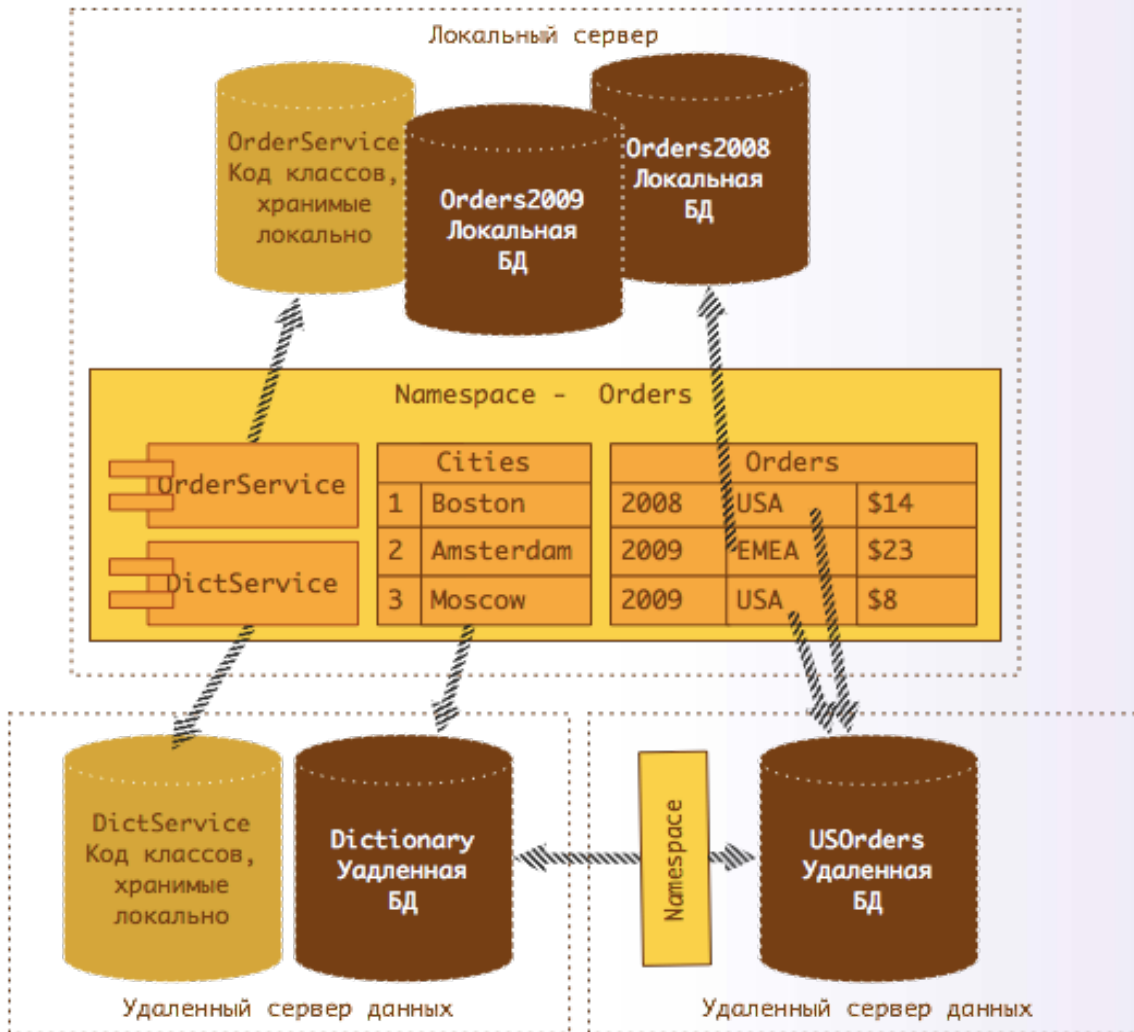


- Легкость конфигурации достигается с помощью использования принципа отделения логического доступа к программам и данным от физического их расположения
- Особенность Cache - отсутствие различия между функционированием сервера Cache как сервера приложений и сервером данных
- Subscript Level Mapping - распределение записей по разным базам данных в зависимости от значения полей записи
 - партиционирование данных
 - сохранять данные в зависимости от контекста распределенно
- Enterprise Cache Protocol

Абстракция доступа



ЕСР и SLM вместе



InterSystems Globals



- Globals is a fast, proven, simple, flexible and free
- Цели проекта
 - Повысить уровень информированности о базовой технологии компании
 - Увеличение круга разработчиков, знакомых с преимуществами технологии InterSystems для успешного создания приложений
- Сравнение с InterSystems Caché
 - Caché предоставляет весь спектр технологий
 - Globals предоставляет доступ к базовой технологии
 - Приложение под Globals работает на Caché

Globals как NoSQL решение



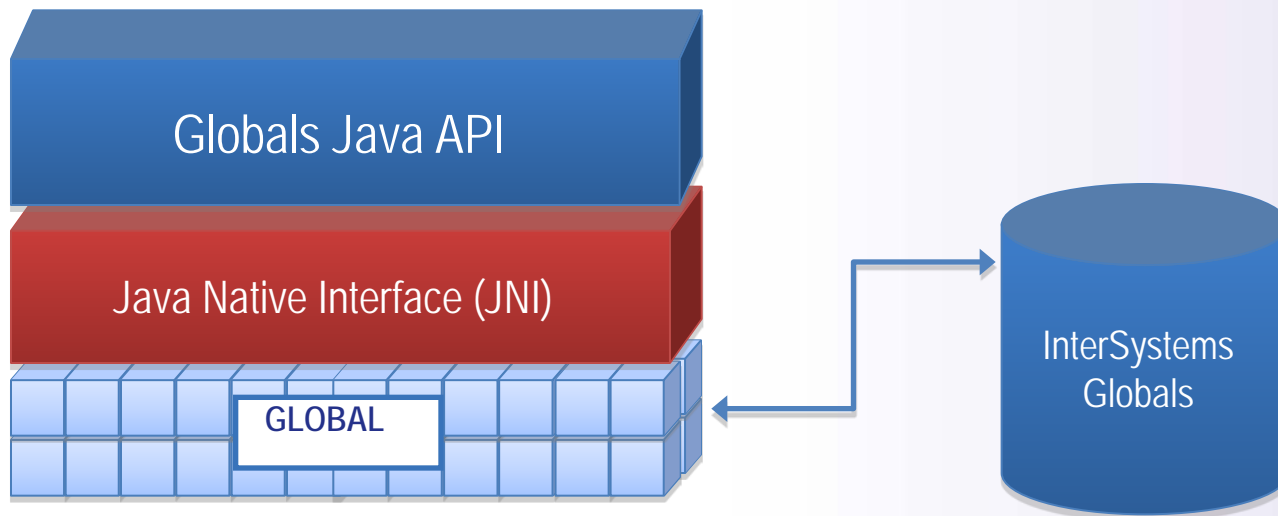
- Свободное использование для разработки и распространения
- Нереляционные модели могут быть реализованы в Globals. Модель данных может как использовать так и не использовать схему
- Globals Document Store (GDS) API
- Индексирование опционально
- Скорость и производительность (порядок - 100,000 записей в секунду)
- Простые API. Java, Node.js, (.Net в процессе разработки)

Отличия Globals от NoSQL



- Нет ограничений на конкретную модель данных (Column/Wide Column, Key-Value, Graph, Document)
- Есть блокировки и транзакции
- Эффективная работа с данными в памяти и целостность данных на диске
- Нет готового механизма для шардинга и партиционирования
- Globals используют стабильную базовую технологию, которая будет гарантированно развиваться и поддерживаться
- При развитии проекта всегда есть возможность перейти на Caché

Общая архитектура



- возможность использовать высокопроизводительное многомерное хранилище Cache в Java приложениях
- отказ от ТСП, использование in-memory доступа
- процесс JVM является процессом Cache

Java. Globals API

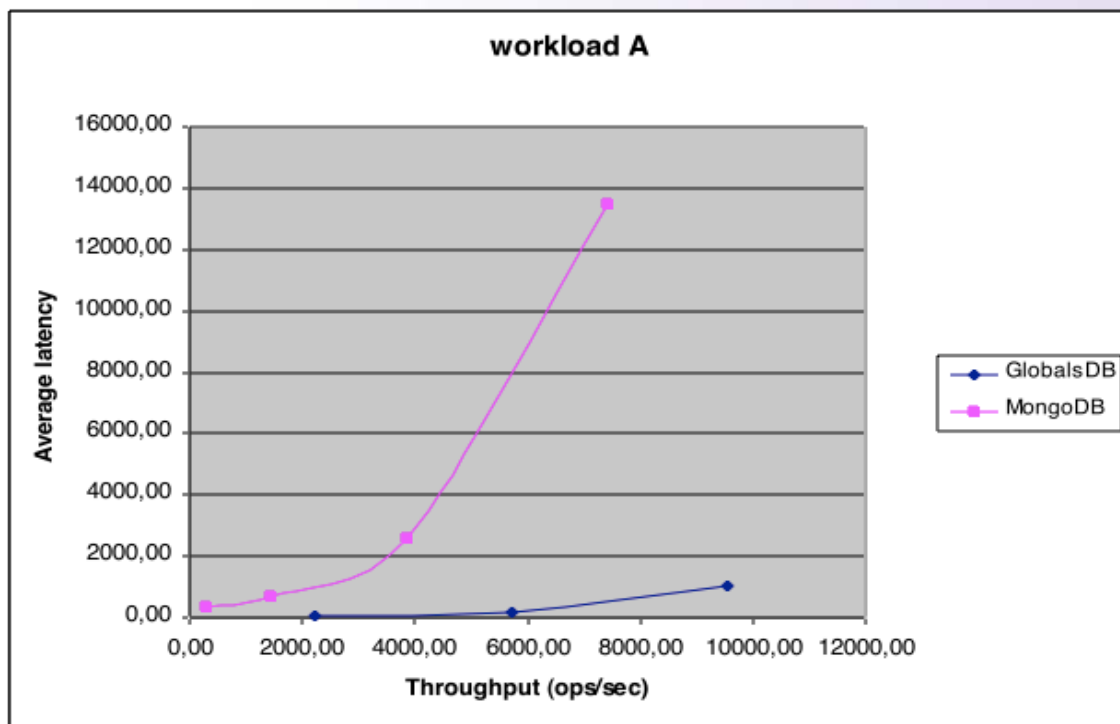


- Работа с глобалами из Java
 - Создание узлов глобалов, навигация по глобалам
 - Преобразование типов данных
 - Транзакции, блокировки
- Основные классы
 - Connection
 - NodeReference
 - ValueList

Yahoo Cloud Serving Benchmark



- Идет процесс тестирования
- Команда МИФИ
- Результаты от 14.09.2011
- GlobalsDB вместе с MongoDB, Cassandra 7.0 и Hbase
- Linux x64, 2x Xeon E5504, 2ГЦRAM 8 ГБ, SATA II 500 ГБ



Node.js



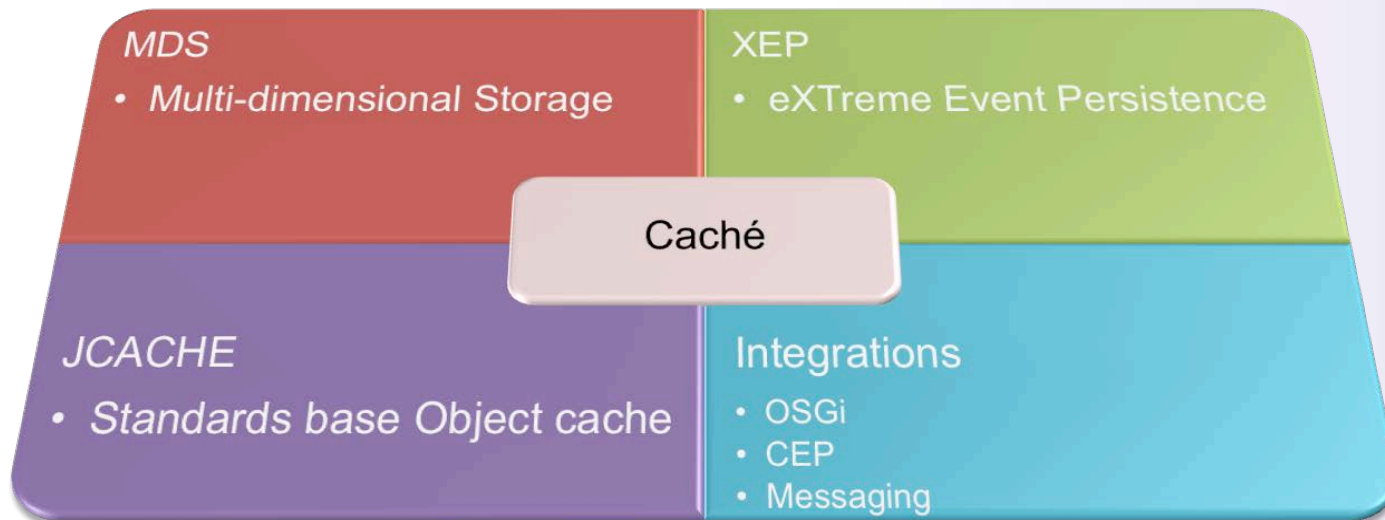
- Серверный javascript с использованием асинхронной работы для разработки высокопроизводительных веб приложений на базе движка V8
- API предоставляет работать с естественными для javascript типами данных
 - примитивные типы
 - массивы
 - объекты

```
myData.open({path: pathToGlobalsMGR, namespace: namespace});  
  
a[n ++] = {global: "Contact", subscripts: [1], data: "Michael Pantaleo"};  
a[n ++] = {global: "Contact", subscripts: [1, "address", 1], data: "San Diego, CA"};  
  
var result = myData.update(a, "array");
```

Caché eXTreme for Java



- КОМПОНЕНТЫ
 - IN-Process JDBC (JNI)
 - Globals API (MDS)
 - eXTreme Event Persistence
 - Dynamic Object API



eXTreme Dynamic Object



- Работа с хранимыми объектами Caché Object Script
- Динамические объекты property-value
- XTDatabaseConnection
 - in-memory соединение
 - транзакции
 - построение индексов
- XTDynamicObject
 - attached/detached
 - `obj.set("property", "value"), obj.getString("property")`
 - `createNew(), openId(), insert(), save(), update(), delete()`

Caché eXtreme Event Persistence



- проекция объектов Java в глобалы (создается хранимый класс %Persistent)
- использование аннотаций
- фильтрация полей
- индексация в отдельном процессе
- EventPersister, Event, EventQueryIterator
- EventQuery или JNI-JDBC

Advanced technologies for breakthrough applications



Вопросы?

Олег Оленин
oleg.olenin@intersystems.com

INTERSYSTEMS