window user can set or change settings. Chose preferred language among four languages: Kazakh, English, Russian or Turkish by mark only one of four QRadioButtons. After saving the changes program at all translates into chosen languages. Also in this window user can changes level from first till own level and upload other files to learn to type according to these files. There are settings of interval between closing and appearance of program, time duration of learning. After changing the settings user must press "SAVE" button. Then program install changes and transfer to start window.

After closing program, it does not stop working, it only turn to tray system on the taskbar on bottom side of window. In tray system user can change settings of interval and duration, open program or quit. In the last window of program user can read instruction of program, how to use it, properties of buttons and how does program work at all. Also in this window written how to sit while typing, which distance must be between user and monitor, what does virtual keyboard show and so on.

The project is not finished yet, but it has already implemented two of the four languages, virtual keyboard, automatic appearance, sound, timer, error checking, help on an error and the database.

Reference

1. http://it-event.ru/news/2060/
2. http://www.ixbt.com/soft/typing.shtml

УДК: 004.430

NoSQL: назад в будущее

Shaimardanova M, Bimagambetova R

Сегодня актуальны эффективные технологии создания специализированных решений с гарантированным временем реакции при обработке больших массивов данных, однако потенциал таких давно существующих подходов, как Cache, реализован еще не полностью.

Проекты класса XTP (extreme transaction processing, высоконагруженные приложения) предъявляют новые требования к СУБД — сегодня необходимы технологии, обеспечивающие низкую стоимость масштабирования и обработки больших объемов данных. К таким технологиям относят СУБД класса NoSQL, обещающие разработчикам высокую скорость внесения изменений в приложения, низкие затраты на масштабирование, инструменты обработки и хранения больших объемов данных, а также высокую скорость выполнения на относительно недорогом оборудовании. В базах NoSQL реализуется отличная от традиционной (одинсервер баз данных для нескольких приложений) парадигма, при которой одному приложению или модулю приложения предлагается отдельное решение по работе с данными. По своей сути архитектура решений NoSQL ориентирована на борьбу либо с большим объемом данных, либо с их повышенной сложностью.

Отличительные черты NoSQL — нереляционные модели данных, простые API или протоколы доступа, способность к горизонтальному масштабированию по требованию для некоторого набора операций на многих серверах, распределенное хранение данных, эффективное использование распределенных индексов и памяти для запросов, достаточно свободное обращение с такими незыблемыми для традиционных СУБД вещами, как транзакционная целостность. Общим для NoSQL-проектов являются компромиссы по отношению к взаимно противоречивым требованиям — например, уход от поддержки стандартных правил для обеспечения транзакционной целостности *ACID* (atomicity, consistency, isolation, durability, атомарность, согласованность, изолированность, долговечность) в пользу горизонтальной масштабируемости. Отказ от подобных требований был невозможен и является догмой для традиционных СУБД. Действительно, нельзя быть одновременно надежным, быстрым, распределенным и целостным, однако в ряде конкретных случаев возможны варианты.

Другой источник компромиссов — характер данных решаемой задачи. Именно тут важно требование к технологии, которая должна максимально использовать особенности предметной области. Например, если можно распараллелить обработку данных и использовать принцип *shared nothing* («без разделения ресурсов»), то нужно это эффективно применять как для хранения, так и для исполнения запросов. В этом случае надо строить модель хранения и распределения данных, которая опирается на эту возможность, однако в реляционных базах свободы выбора почти нет и

приходится использовать то, что есть, — скажем, нельзя хранить данные по колонкам на разных серверах (рис. 1). При этом, в отличие от традиционных СУБД, технология NoSQL дает разработчику больше свободы при учете естественных особенностей проектной задачи, которые могут быть использованы, например, для горизонтального масштабирования. Однако разработчик несет больше ответственности за принимаемые архитектурные решения.

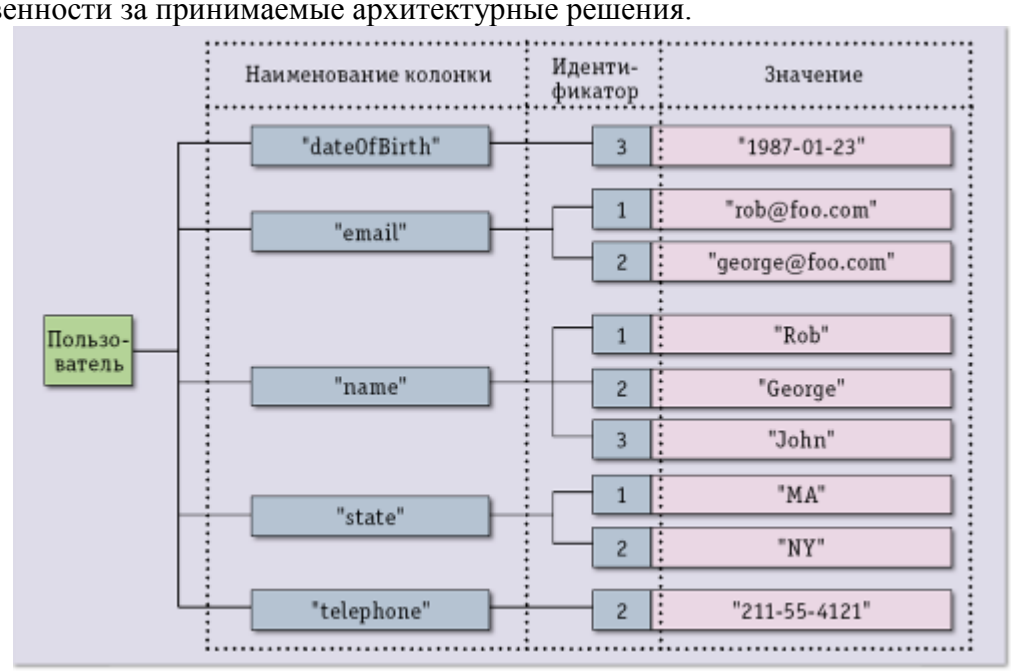

Рис. 1.Реализация поколоночного хранения

Несмотря на многообразие NoSQL-проектов, до сих пор нет ни одного, который можно было бы назвать универсальной и всеобъемлющей NoSQL-платформой. Поэтому, если у разработчиков возникла идея воспользоваться NoSQL-подходами в следующем проекте, то прежде придется ответить на целый ряд вопросов и разрешить множество рисков, например: какую модель данных выбрать; насколько стабильна и отработана выбранная технология; насколько серьезны будут изменения в коде в случае попытки смены технологии, использующей NoSQL-решения, на другую; будет ли язык запросов достаточно полным и технологичным для удовлетворения проектных требований. Отдельно стоит отметить, что многие NoSQL-технологии были созданы специально в рамках конкретного проекта и есть вероятность того, что, закрывая требования и задачи первоначального проекта, они могут оказаться не очень подходящими для другого.

Для начального знакомства с NoSQL с минимальными рисками целесообразно выбрать гибридную архитектуру для подсистемы управления хранимыми данными: одновременно использовать технологию NoSQL и обычную СУБД; использовать технологию, поддерживающую концепции обоих миров, например InterSystems Cache.

В основе Caché лежит реализация более простой, чем реляционная, модели, называемой по имени своих атомарных элементов *глобалами* (global — от Global Persistent Variable), которые не имеют схемы, допускают динамическое добавление столбцов и используют разреженное хранение значений столбцов. На уровне глобалов можно использовать блокировки, транзакции, организовать распределенное хранение и партиционирование. Допуская некоторую неточность в определении, можно говорить о глобалах как о структуре данных, схожей с существующими в языках программирования коллекциями с доступом к элементам по ключу, например ассоциативным массивом в PHP.

Модель глобалов ближе всего к одной из самых простых и распространенных моделей данных в NoSQL — модели *ключ-значение* (key-value). В качестве значений могут храниться скалярные типы, длинные строки и списки, а хранение данных происходит без заранее определенной схемы, причем за интерпретацию данных отвечает само приложение. В том или ином виде все прочие модели являются в некотором смысле наследниками модели ключ-значение. Глобалы

также предоставляют базис для построения нереляционных моделей, используемых в NoSQL: расширяемые записи, хранение по колонкам, документо-ориентированные и графовые модели.

На уровне глобалов не существует привычного для реляционных баз данных декларативного языка запросов — запросы определяются алгоритмически, а их выполнение сводится к исполнению кода на языке Cache Object Script, который предоставляет набор простых операций для работы с данными, хранимыми в глобалах. Это, пожалуй, единственный язык, в синтаксис которого явно введена конструкция для определения персистентности переменной — указания области ее существования — в памяти или в базе данных. Если бы в Java или .NET была такая возможность, то не существовало бы проблем с преодолением среды между программой и базой данных. Отсутствие такой конструкции в универсальных языках программирования кажется странным — ведь естественно предположить, что код работает не только с переменными в памяти, но и с хранимыми переменными. При этом не надо заранее определять структуры в базе данных, а можно просто работать с ними так же, как с переменными в языках с нестрогой типизацией.

Компания InterSystems реализовала на основе глобалов объектно-ориентированную и реляционную модели, но можно на них реализовать собственную, уникальную модель данных, построенную на принципе Unified Data Model (рис. 2). Суть принципа заключается в том, что работа с одними и теми же хранимыми данными происходит через разные модели представления, в зависимости от контекста конкретной задачи. К примеру, для быстрой вставки и чтения используется модель ключ-значение, а для запросов используются возможности реляционной модели. При построении своего языка запросов для NoSQL-решения можно использовать язык Cache Object Script, предоставляющий набор простых операций для работы с данными, хранимыми в глобалах.

Рис. 2. Концепция Unified Data Model основана на принципе «данные одни — моделей представления много»

Оценивая возможности Cache в таких важных для NoSQL категориях, как горизонтальное масштабирование и организация распределенного хранения данных, необходимо отметить, что в этой СУБД изначально на базовом уровне были заложены технологии для партиционирования данных и их хранения в распределенных базах. Разделение данных на основе правил разрешения логических имен структур хранения (глобалов) и правил для ключей, в том числе составных, записей происходит с использованием Subscript Level Mapping (SLM) и областей (namespace). Фактически SLM и области абстрагируют физический адрес хранения данных от логических имен и ссылок на записи, которые использует приложение. Конфигурация правил хранения происходит без изменения приложения. Правила SLM допускают как локальное распределение данных, так и распределение данных по разным серверам Cache. Для организации распределенной работы (рис. 3) используется технология ECP (Enterprise Cache Protocol).

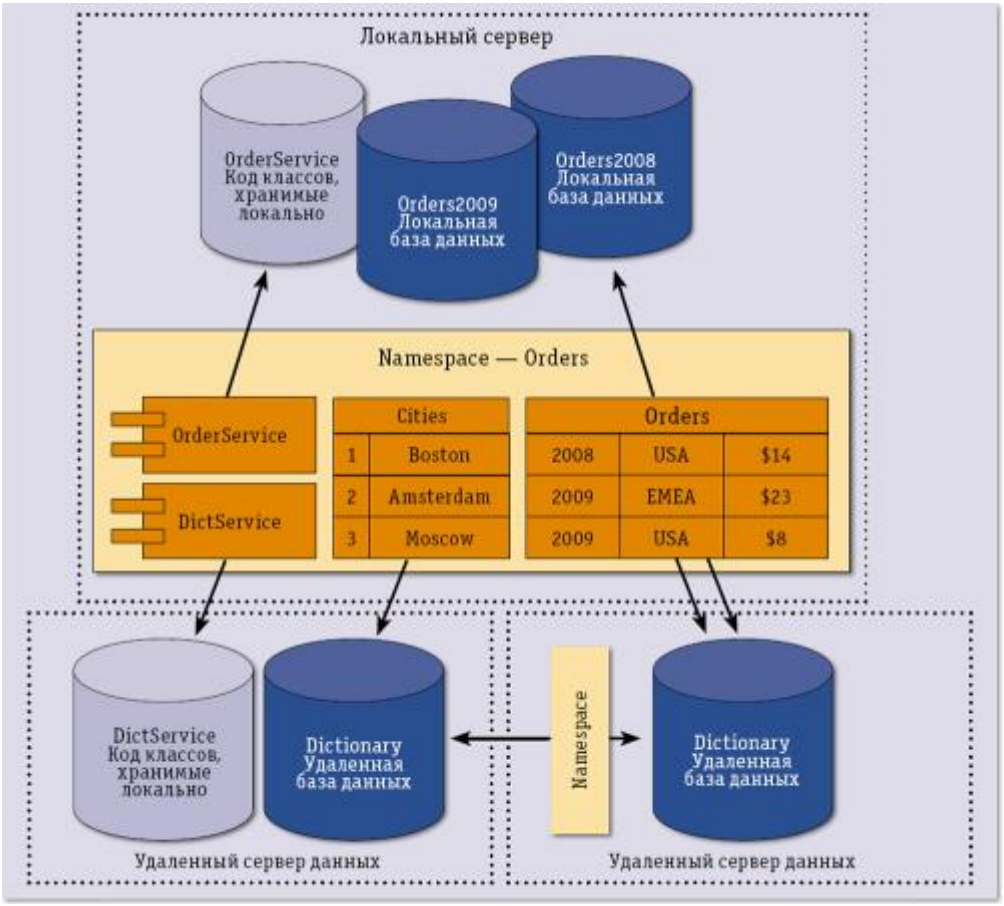

Рис. 3. Партиционирование и распределенное хранение с помощью ECP и SLM

Проиллюстрировать возможности Cache по распределенному хранению можно примером реализации подхода, который ввели в обиход инженеры Google при создании BigTable и который предполагает *шардинг* (sharding) — разбиение логически замкнутых частей базы данных на части для хранения на разных серверах. Цель шардинга — при увеличении нагрузки распределить данные. При использовании шардинга разные записи могут распределяться по серверам на основе попадания первичного ключа в определенный диапазон или построения хэша по первичному ключу. Данные одной записи также могут храниться, согласно правилам для группы, на разных серверах — например, группа атрибутов с адресной информацией хранится отдельно от группы атрибутов с финансовой информацией. В случае Cache правила SLM позволяют по ключу или его части распределять записи (узлы) глобалов и при определенной организации модели хранения обеспечивать хранение значений разных атрибутов одной записи на разных серверах. ECP при этом обеспечивает доступ к удаленным базам и согласованную работу с распределенными данными.

Помимо распределенной работы с данными, хранимыми на разных серверах, ECP дает возможность осуществлять горизонтальное масштабирование вычислительной нагрузки при одновременной работе большого количества пользователей за счет создания распределенного кэша данных.

Технологии NoSQL сейчас очень популярны, поэтому InterSystems выпустила бесплатную СУБД InterSystems Globals, позволяющую оценить возможности технологии работы с глобалами. Эта СУБД предоставляет простые API для работы из Java, Node.js и .Net. В отличие от Cache Object Script, в Globals операции по работе с глобалами доступны из внешнего по отношению к СУБД языка программирования. При этом процесс работающего с Globals приложения (например, виртуальной машины Java) становится фактически одним из процессов СУБД. В Globals могут быть реализованы нереляционные модели.

В случае с Java технология Globals позволяет быстро реализовывать свои собственные структуры хранимых данных, которые при этом естественны для языка. Например, можно быстро создать аналог HashMap, данные которого будут храниться в СУБД. При таком подходе, так же как и в случае Cache Objects Script, различия между переменной в памяти и на диске начинают исчезать, что делает работу с хранимыми данными более естественной.

Для Node.js доступ к Globals позволяет работать с естественными для JavaScript типами данных — массивы и объекты JavaScript можно без дополнительных накладных расходов на разработку сразу сохранять, читать и изменять, что упрощает проблему персистентности данных. Вдобавок к этому Globals в связке с Node.js дает высокую скорость работы — например, Globals работает быстрее, чем Redis (один из распространенных проектов NoSQL).

Globals позиционируется как база данных NoSQL, но при этом отличается от основного потока реализаций NoSQL: нет ограничений на конкретную модель данных; имеется возможность использования блокировок и транзакций; обеспечена одновременно эффективная работа с данными в памяти и на диске; отсутствует технология для распределенной работы с хранимыми данными. Globals предоставляет ядро по работе с глобалами без объектного и реляционного доступов, но при развитии проекта всегда есть возможность перейти на Caché без изменения кода приложения — Globals API является подмножеством технологии Caché Extreme.

***

InterSystems Cache вполне можно отнести к СУБД NoSQL с поддержкой объектной и реляционной моделей, которая не уступает по своим эксплуатационным качествам традиционным СУБД. Несмотря на то что СУБД NoSQL стали популярны недавно, идеология СУБД InterSystems Cache неплохо приспособлена для работы с современными высоконагруженными приложениями, обеспечивающими гарантированное время реакции при обработке больших массивов данных.

УДК: 004.430

Parallel computing and OpenMPI

Kurmanova Zhanna

Parallel Computing is evolved from serial computing that attempts to emulate what has always been the state of affairs in natural World. We can say many complex irrelevant events happening at the same time sequentionally. For instance; planetary movements, automobile assembly, galaxy formation, weather and ocean patterns.

Historically, it is considered to be “the high end of computing” and has been used to model difficult scientific, computational and engineering problems.

In computational field technique which is used for solving the computational tasks by using different type multiple resources simultaneously is called as parallel computing. It breaks down large problem into smaller ones, which are solved concurrently [1]**.**

The another term related to parallel computing is parallel processing. It describes simultaneous use of more than one central processing units in order to execute a program.

There are many possible ways of designing a parallel computer, and Michael Flynn in 1966 developed a **taxonomy** for parallel processors, a way of thinking about these alternatives. Flynn categorized them based on two parameters: the stream of instructions (the **algorithm**) and the stream of data (the input). The instructions can be carried out one at a time or concurrently, and the data can be processed one at a time or in multiples. In Flynn's scheme, SISD is "Single Instruction stream, Single Data stream," and refers to a traditional sequential computer in which a single operation can be carried out on a single data item at a time.

The two main categories of parallel processor are SIMD and MIMD. In a SIMD (Single Instruction, Multiple Data) machine, many processors operate simultaneously, carrying out the same operation on many different pieces of data. In a MIMD (Multiple Instruction, Multiple Data) machine, the number of processors may be fewer but they are capable of acting independently on different pieces of data. The remaining category, MISD (Multiple Instruction, Single Data) is rarely used since its meaning is not