МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	бакалавриат
	(бакалавриат, магистратура, специалитет)
Форма обучения	очная
	(очная, очно-заочная, заочная)
Направление(-я) подготовки	10.05.04 Информационно-аналитические системы безопасности
	(код(-ы) и наименование(-я))
Институт	Кибербезопасности и цифровых технологий (ИКБ)
	(полное и краткое наименование)
Кафедра	КБ-2 «Прикладные информационные технологии»
	(полное и краткое наименование кафедры, реализующей дисциплину (модуль))
Лектор	к.т.н., Селин Андрей Александрович
	(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года 2024/2025

(учебный год цифрами)

Проверено и согласовано «____» _____2024 г. _____ А.А. Бакаев

(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

Технологии хранения в системах кибербезопасности

2024 год

Лекция 7.
MapReduce

Учебные вопросы лекции:

1. Документоориентированные БД
2. Big Data
3. MapReduce

Документоориентированные БД. Альтернативы MongoDB

В предыдущей лекции мы обсудили NoSQL документоориентированную СУБД с открытым исходным кодом MongoDB. Вместо использования таблиц и строк, как в традиционных реляционных базах данных, MongoDB использует коллекции и документы. Документы состоят из пар ключ-значение, которые являются основной единицей данных в MongoDB.

MongoDB использует большой объем данных из-за неиспользования нормализации. Более того, MongoDB имеет строгую схему со сложными модификациями, сложной масштабируемостью, отсутствием поддержки транзакций, использованием большого объема памяти, низкой производительностью и т.д. Проблемы, с которыми трудно справиться. Поэтому, сегодня поговорим об известных альтернативах MongoDB.

NoSQL альтернативы MongoDB:

RethinkDB;
Couchbase;
MarkLogic;
eXist;
Berkeley DB;
CouchDB.

Альтернативы MongoDB

RethinkDB

RethinkDB – это open source БД для real-time приложений. Она располагает встроенной системой уведомления об изменениях, которая непрерывно транслирует обновления для вашего приложения.

RethinkDB является безсхемным хранилищем JSON-документов, но также поддерживает и некоторые особенности реляционных БД. RethinkDB поддерживает кластеризацию, что позволяет очень удобно ее расширять. Реализована возможность настройки шардинга и копирования через встроенный веб-интерфейс. Последняя версия RethinkDB также включает в себя автоматический «fail-over» для кластеров с тремя и более серверами (подразумевается возможность продолжения работы с БД в случае выхода из строя одного из серверов.)

Язык запросов в RethinkDB, который называется ReQL, нативно встраивается в код на том языке, на которым вы пишите своё приложение. Если, например, вы используете ЯП Python, то при написании запросов к БД будете использовать обычный для Python синтаксис. Каждый запрос составляется из функций, который разработчик собирает в цепочку, чтобы точно описать необходимую операцию.

RethinkDB

RethinkDB содержит в себе таблицы, в которых хранятся традиционные JSON-документы. Структура самих JSON объектов может иметь глубокую вложенность.

Каждый документ в RethinkDB имеет свой основной ключ (primary key) – свойство *«id»* с уникальным для таблицы-родителя значением. Ссылаясь на primary key в запросе, можно получить конкретный документ. Написание ReQL-запросов в приложении похоже на использование API-конструктора SQL-запросов. Ниже, на языке JavaScript, представлен простой пример ReQL запроса для определения количества уникальных фамилий в таблице *users*:

```
r.table("users").pluck("last_name").distinct().count()
```

В ReQL запросе каждая функция цепочки работает с данными, полученными из предыдущей функции. Если быть точным, то порядок выполнения данного запроса таков:

table запрашивает определенную таблицу в БД;

pluck достаёт определенное свойство (или несколько свойств) из каждой записи;

disctinct убирает повторяющиеся значения, оставляя только по одному уникальному;

count подсчитывает и возвращает количество полученных элементов.

RethinkDB

Традиционные CRUD операции также просты. ReQL включает в себя функцию ***insert***, которую можно использовать для добавления новых JSON документов в таблицу:

```
r.table("team").insert([
  { name: “Arshavin", species: "forward" },
  { name: “Dzuba", species: "forward" },
  { name: “Kerzhakov", species: "forward" },
  { name: “Kokorin", species: "forward" },
  { name: “Akinfeev", species: "goalkeeper" },
  { name: “Kombarov", species: "defender" },
  { name: “Shirokov", species: "halfback" },
  { name: “Dzagoev", species: "halfback" }
])
```

Функция ***filter*** достаёт документы, которые соответствуют определённым параметрам:

```
r.table("team").filter({species: "forward"})
```

Вы можете добавлять в цепочку такие функции как ***update*** или ***delete***, чтобы выполнить определенные операции над документами, возвращенными из ***filter***:

```
r.table("team").filter({species: "forward"}).update({species: " halfback"})
```

RethinkDB

ReQL включает более 100 функций, которые можно комбинировать для достижения необходимого результата. Есть функции для управления потоками, изменения документов, агрегации, записи и т.д. Также существуют функции, для выполнения стандартных операций со строками, числами, метками времени и гео-координатами.

Существует команда ***http***, которую можно использовать для получения данных из сторонних Web-API. В следующем примере показано как можно использовать ***http*** для получения постов с Reddit:

```
r.http("http://www.reddit.com/r/aww.json")("data")("children")("data").orderBy(r.desc("score")).limit(5).pluck("score", "title", "url")
```

После того как посты получены, они сортируются по очкам и затем отображаются определённые свойства лучшей пятёрки постов. Используя ReQL «на полную мощность», разработчики могут выполнять действительно сложные манипуляции с данными.

RethinkDB

RethinkDB client libraries (далее «драйверы») отвечают за интеграцию ReQL в тот язык программирования, на котором ведется разработка приложения. Драйверы внедряют функции для всевозможных запросов, поддерживаемых БД. ReQL-выражения расцениваются как структурированные объекты, похожие на абстрактное синтаксическое дерево. Для того чтобы выполнить запрос, драйверы переводят эти объекты запроса в специальный формат "RethinkDB's JSON wire protocol format", в котором затем передаются в БД.

Функция **run**, замыкающая цепочку, переводит запрос, выполняет его на сервере и возвращает результат. В официальных драйверах работа с соединением выполняется в ручном режиме. Это значит что нужно создавать соединение и закрывать его после выполнения операции.

В следующем примере показано как выполнить запрос в RethinkDB из-под Node.js с установленным драйвером ReQL для JavaScript. Этот запрос достаёт всех полуросликов (halflings) из таблицы fellowship и отображает их в консоли:

```
var r = require("rethinkdb");

r.connect().then(function(conn) {
return r.table("fellowship")
        .filter({species: "halfling"}).run(conn)
   .finally(function() { conn.close(); });
})
.then(function(cursor) {
return cursor.toArray();
})
.then(function(output) {
console.log("Query output:", output);
})
```

RethinkDB

Выводы по разделу:

Эта альтернатива MongoDB помогает упростить создание и масштабирование приложений реального времени;

Можно создавать современные приложения, используя свой любимый веб-фреймворк;

Возможность работать в паре с технологиями реального времени, такими как SignalR и Socket.io;

Позволяет интегрировать последние достижения в технологии БД;

RethinkDB работает под Linux и MacOS X, версия для Windows находится в активной разработке.

Couchbase

Couchbase

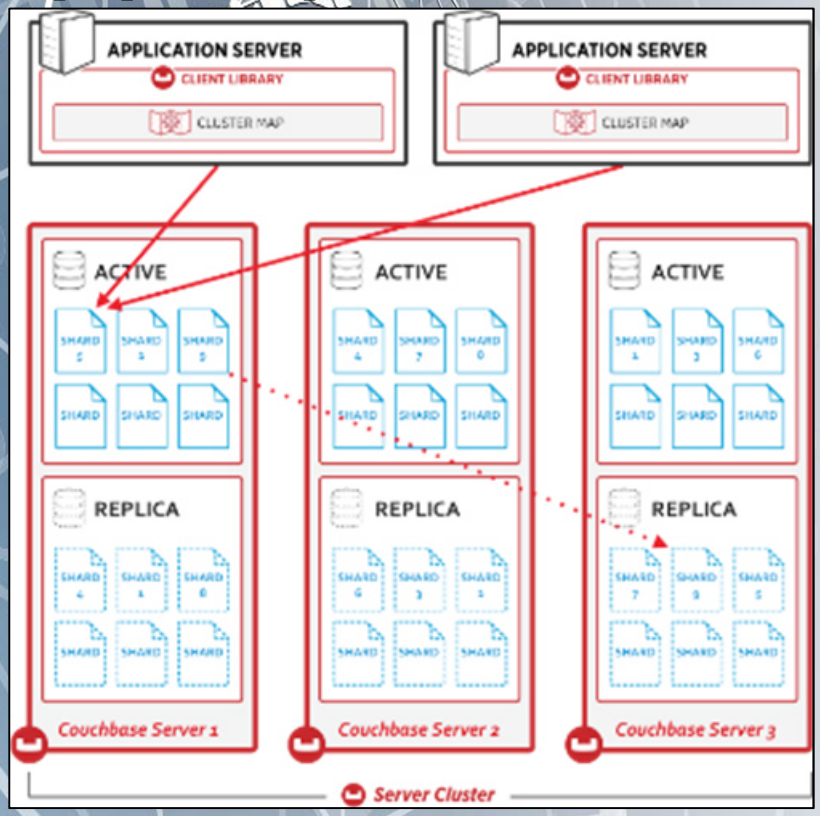
В основе Couchbase лежит распределенное хранилище ключей (KV). Хранилище KV представляет собой чрезвычайно простой подход к управлению данными, который хранит уникальный идентификатор (ключ) вместе с частью произвольной информации. Само хранилище KV может принимать любые данные, будь то бинарный blob или JSON-документ. Благодаря простоте реализации KV доступ к данным обеспечивается с минимальной задержкой. Задержки по сети в 2-3 раза выше, чем предоставление данных по ключу на стороне Couchbase.

Документы хранятся на сервере Couchbase в формате JSON. Формат поддерживает как базовые типы данных, такие как числа, строки и сложные типы, так и встроенные словари и массивы.

Схема данных в Couchbase является логической конструкцией, определяемой приложением и разработчиком. За счет ее гибкости и возможности использовать несколько вариантов, мы можем использовать в документе тег, например, с информацией о версии. Это дает возможность приложению определить, в каком режиме обрабатывать документ, а также обеспечить плавную миграцию базы на новую схему данных.

Couchbase

Одним из составляющих параметров информационной системы является ее доступность. **Couchbase обеспечивает высокую доступность данных** с помощью множества различных функций. Одной из них является репликация данных (распределение нескольких копий данных на разных серверах кластера), что позволяет предоставлять сервис при регламентных работах или же выходе части серверов из строя.



Реплики сервера Couchbase

Второй важной функцией для обеспечения высокой доступности является внутренний протокол DCP (Database Change Protocol). Он обеспечивает высокоскоростную передачу изменений во все копии данных, вторичные индексы (GSI), межкластерную репликацию (XDCR) и внешним потребителям.

Couchbase

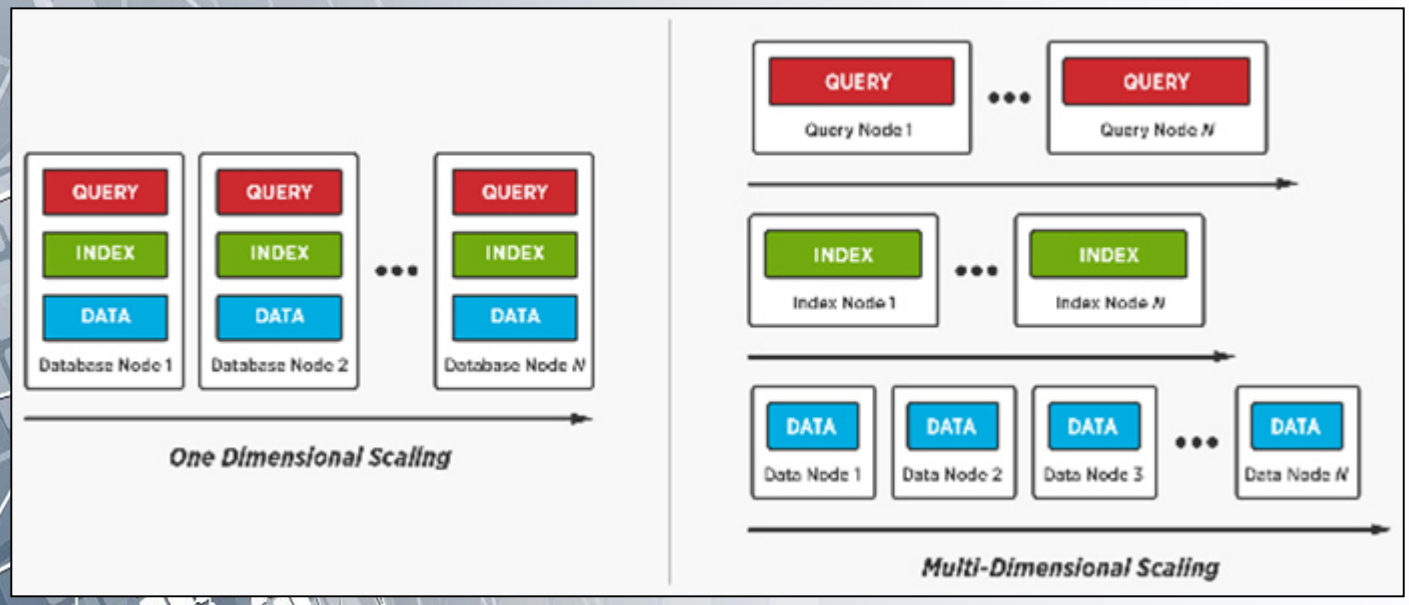
Правильной практикой в организациях является использование резервирования всех бизнес-процессов и оборудования. В идеальном варианте это резервирование в режиме Active-Active (A-A), когда переключение между проблемными узлами происходит автоматически. **Двунаправленная репликация** в Couchbase позволяет обеспечить режим A-A. Однако тестирование репликации показало, что она эффективна только в близко стоящих ЦОД. При разнесении более 100 км появляются конфликты.

У Couchbase есть механизмы решения конфликтов: на основе Timestamp и Sequence Number. Тем неменее, из-за временной задержки на сети, в БД попадают устаревшие данные.

Одной из важных характеристик большинства NoSQL БД является горизонтальное масштабирование. Основным отличием Couchbase является поддержка многомерного масштабирования, когда в кластере наращивается по производительности только нужная служба.

Пример: игра Pokemon GO использует разделение архитектуры. На старте проекта использовалось 5 серверов с совмещенными службами. После увеличения нагрузки ими была применена разнесенная архитектура: 5 серверов с данными и 55 серверов для обработки запросов и индексов. Одним из минусов масштабирования у Couchbase является возникновение проблем у оркестратора, при наличии в кластере свыше 50 дата нод.

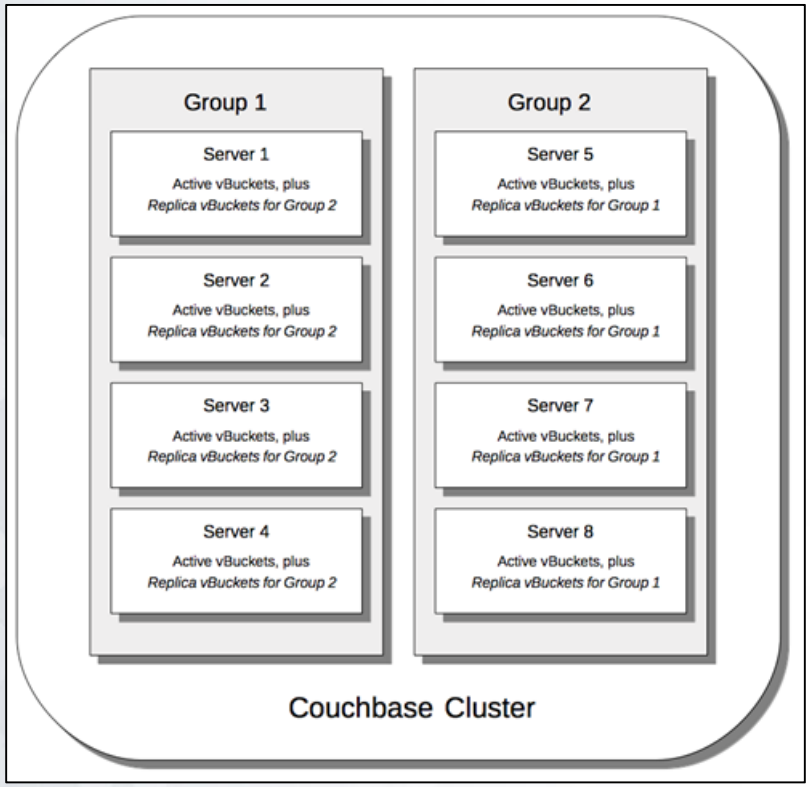
Couchbase



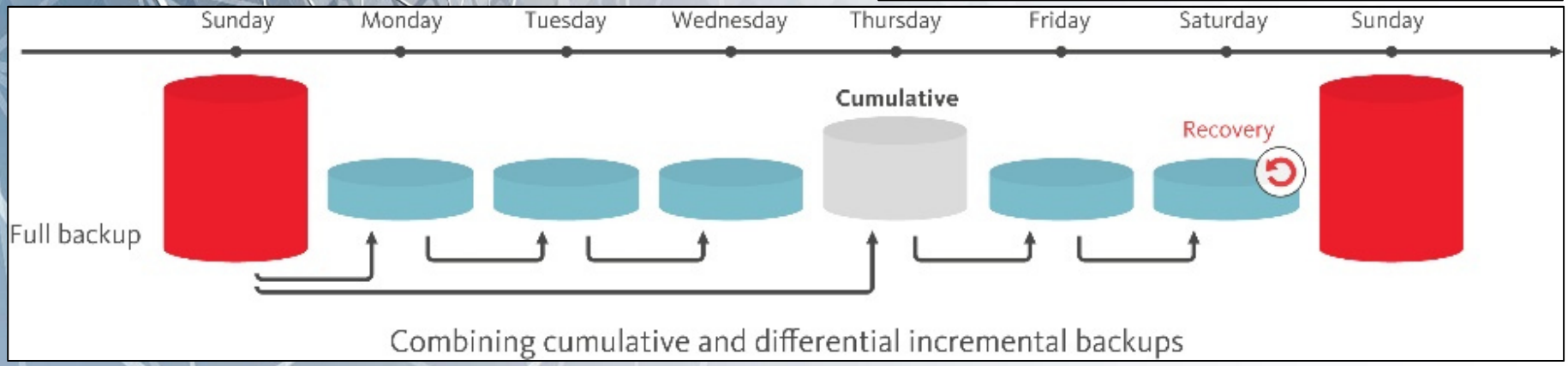
Существенный плюс Couchbase – прозрачный механизм обновления и поддержка API старых версий. На время обновления кластера, он работает в режиме совместимости. Новые механизмы заработают только после полного обновления кластера. Влияния на работающие приложения минимальны за счет поддержки старого API.

Couchbase

Еще одной из интересных возможностей является объединение серверов в кластере в логические группы, с привязкой к ним реплик. Это позволяет распределять полные копии реплик одного кластера по разным группам серверов. Что позволяет при выходе из строя одной группы серверов иметь полную копию данных во второй.



Couchbase содержит в себе готовые инструменты для резервного копирования и восстановления. Процесс бекапа может работать в трёх режимах: полном, дифференциальном и накопительном. Это позволяет в некоторых случаях сэкономить место на дисках и процессорные ресурсы.



Couchbase

Сравнение Couchbase и MongoDB

	Couchbase	MongoDB
Масштабирование	Автоматическое для всего набора данных	Ручной выбор ключа
Распределение данных	Данные всегда равномерно распределены по всем дата нодам	Неправильная разметка может привести к перекосу распределения данных
Добавление/удаление узла или реплики	Добавляется в один шаг через GUI, с ребалансировкой	Довольно сложная задача с расчетами веса для каждой коллекции
Распределение реплик по стойкам/ЦОД	Реализовано через логические группы	Не реализовано
Автоматическое распределение нагрузки	Каждая нода имеет одинаковое количество активных записей доступных на чтение и запись	Не сбалансировано. Вторичные узлы не поддерживают запись
Масштабирование индексов	Гибкое, можно добавлять отдельные индекс ноды за счет разнесенной архитектуры	Жесткое, масштабирование индекса связано с масштабирование данных.
Метаданные кластера	Распределяются по всем узлам кластера	Требуются сервера конфигурации
Интегрированный поиск	N1LQ(SQL++)	JSON запрос

Couchbase

Сравнение репликации

	Couchbase	MongoDB
Архитектура	Межкластерная репликация не имеет зависимостей, кластера независимы друг от друга	Только внутрикластерное расширение
Гибкость настройки	Гибкая(настройка отдельных бакетов, фильтры, тюнинг)	Тюнинг скорости
Топология	Двунаправленая репликация, звезда, цепь и т.д.	Звезда
Режим Active-Active	Поддерживается	Не поддерживается

Big Data

Термин «**Big Data**» начал набирать популярность с 2011 года. С развитием технологий количество данных стало увеличиваться в геометрической прогрессии. Традиционные инструменты перестали покрывать потребность в обработке и хранении информации. Для обработки данных, объем которых превышает сотни терабайт и постоянно увеличивается, были созданы специальные алгоритмы. Их принято называть «big data».

Сегодня информация собирается огромными объемами из разных источников: интернет, контакт-центры, мобильные устройства и т.д. Чаще всего такие данные неструктурированы и неупорядочены, поэтому человек не может использовать их для какой-либо деятельности. Для автоматизации анализа применяют технологии «big data».

Big Data – это крупные массивы разнообразной информации и стек специальных технологий для работы с ней.

Большие данные – обозначение структурированных и неструктурированных данных огромных объёмов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми программными инструментами.

Big Data

Big Data - это данные со следующими характеристиками:

- большой объем;
- высокая скорость поступления;
- разрозненность источников;
- неструктурированность.

Объем. Из названия «большие данные» становится понятно, что они содержат в себе много информации. И это действительно так, «большие данные» не были бы таковыми без объема.

Скорость. Большие данные поступают и обрабатываются из разных источников с высокой скоростью. При отсутствии этого свойства информацию уже нельзя будет назвать «big data». А еще они генерируются без остановки.

Неструктурированность (разнообразие). Большие данные содержат в себе информацию, относящуюся к разным типам. Это одно из главных отличий от простых данных, которые всегда структурированы и могут быть сразу сохранены в базе данных.

Big Data

Существует шесть основных критериев (шесть «V»), которые определяют **Big Data**:

Volume (объем) – V поступающей информации более 150 Гб в сутки.

Velocity (скорость) – для работы с массивами информации в режиме реального времени требуются повышенные вычислительные мощности.

Variety (разнообразие) – поступающая информация имеет разные форматы или степень структурированности. Например, контент социальных сетей может сильно различаться даже в пределах одной страницы.

Veracity (достоверность) – источникам данных можно доверять, а результат их обработки обладает достоверностью, достаточной для принятия решений.

Variability (вариативность) – поток данных изменчив, на него может влиять множество факторов . Например, в час пик приходит больше данных от таксистов.

Value (ценность) – данные могут иметь разное значение для компании. Например, сделки с крупными покупателями имеют большее значение, чем с мелкими.

Big Data

Примеры типов данных

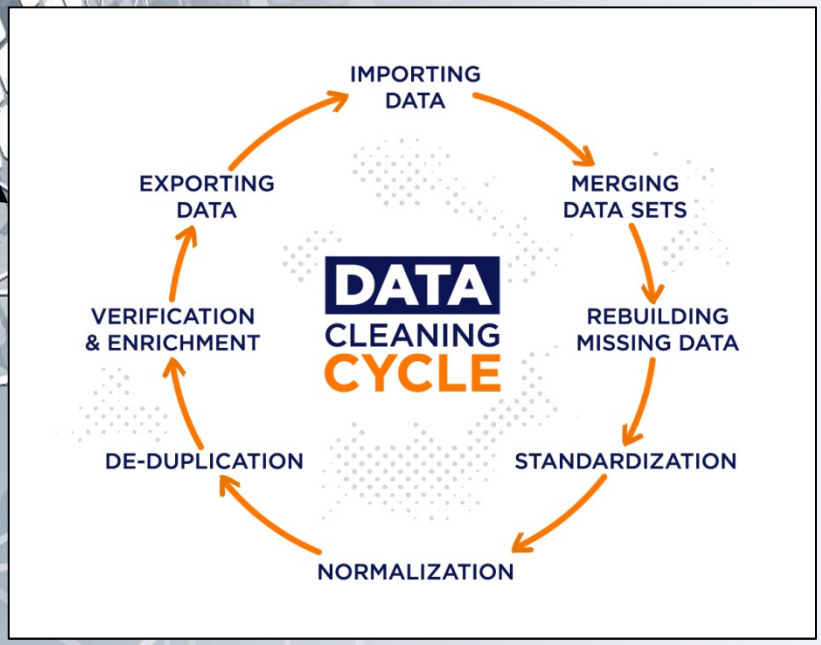
Большие данные	Обычные данные
Записи всех звонков сотрудников крупного Call-центра	Бухгалтерские отчеты компании в Excel
Поисковые запросы, переходы по ссылкам, движения и нажатия мыши всех пользователей Яндекса	ФИО и возраст всех пользователей сервиса Яндекс Лавка
Сведения о перемещениях таксистов, трафик и спрос на поездки	Расписание маршрутов всего общественного транспорта области
Информация о покупках клиентов банка и снятии ими наличных в терминалах и отделениях	Список клиентов с просроченными задолженностями

Big Data

Этапы работы с Big Data :

- сбор;
- хранение;
- обработка;
- анализ.

Сбор. Все начинается с интеграции технологий сбора информации, определения ее источников и необходимой обработки. Это могут быть действия пользователей сайта, отчеты о продажах, статистические, медицинские и любые другие данные, которые ценны для компании. К процессу также подключаются специалисты по **Data Cleaning,** которые настраивают фильтры для будущего анализа.



Big Data

Хранение. Для таких объемов информации недостаточно будет даже нескольких компьютеров, поэтому компании прибегают к услугам облачных провайдеров и задействуют распределенные вычислительные мощности. Примеры технологий, которые используются для хранения:

Data Warehouse – единое корпоративное хранилище с обработанной и структурированной информацией. Хранилище упрощает анализ полученных данных, но требует структурированности.

Data Vault – одна из моделей хранилища Data Warehouse с временными отметками размещения данных, которые позволяют проследить изменение хранимой информации во времени.

Data Lake – данные в хранилище поступают непрерывно в неструктурированном или, наоборот, структурированном или слабоструктурированном виде. Используется для сбора данных из разных источников в режиме реального времени.

Data Mart – хранилище данных, предназначенных для повседневного использования. Поступающую информацию необходимо тщательно обрабатывать, но после этого к ней проще регулярно обращаться.

Big Data

Обработка. Для обработки крупных объемов информации используется технология MapReduce. Массивы распределяются на разных узлах, которые могут обрабатывать их параллельно, даже если на одном узле случилась ошибка. На MapReduce, например, работают кластеры Apache Spark, Apache Hadoop.

Анализ. Заключительным этапом работы является анализ – получение самого ценного из всего хранилища данных. С помощью СУБД, нейросетей и других инструментов массивы информации преобразуются в таблицы, диаграммы, графики и другое

Области применения Big Data:

– **телекоммуникации;**
– **торговля;**
– **банковская сфера;**
– **транспорт;**
– **подбор персонала;**
– **автомобилестроение;**
– **и др.**

Big Data

Инструменты для работы с большими данными (пример):

Назначение	Технологии
Обработка транзакций	Managed Service for PostgreSQL Managed Service for MongoDB Managed Service for MySQL Managed Service for SQL Server
Запросы и отчеты	Data Proc Managed Service for Greenplum Managed Service for ClickHouse
New SQL	Managed Service for YDB
Документоориентированная СУБД	Managed Service for MongoDB
Резидентная СУБД	Managed Service for Redis
БД «ключ — значение»	Data Proc Managed Service for Greenplum Managed Service for MongoDB
БД временных рядов	Managed Service for ClickHouse
Потоковая обработка	Data Proc Managed Service for Apache Kafka
Полнотекстовый поиск	Managed Service for OpenSearch
Очередь сообщений	Message Queue

MapReduce

MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объемов данных на компьютерных кластерах. Изначально название MapReduce было запатентовано Google, но по мере развития технологий Big Data стало общим понятием мира больших данных. Сегодня множество различных коммерческих, так и свободных продуктов, использующих эту модель распределенных вычислений: Apache Hadoop, Apache CouchDB, MongoDB, MySpace Qizmt и прочие Big Data фреймворки и библиотеки, написанные на разных языках программирования.

Среди других наиболее известных реализаций MapReduce стоит отметить следующие:

– Greenplum – коммерческая реализация с поддержкой языков Python, Perl, SQL и пр.;
– GridGain – бесплатная реализация с открытым исходным кодом на языке Java;
– Phoenix – реализация на языке C с использованием разделяемой памяти;
– MapReduce реализована в графических процессорах NVIDIA с использованием CUDA;
– Qt Concurrent – упрощённая версия фреймворка, реализованная на C++, для распределения задачи между несколькими ядрами одного компьютера;
– CouchDB использует MapReduce для определения представлений поверх распределённых документов;
– Skynet – реализация с открытым исходным кодом на языке Ruby;
– Disco – реализация от компании Nokia, ядро которой написано на языке Erlang, а приложения можно разрабатывать на Python;
– Hive framework – надстройка с открытым исходным кодом от Facebook, позволяющая комбинировать подход MapReduce и доступ к данным на SQL-подобном языке;
– Qizmt – реализация с открытым исходным кодом от MySpace, написанная на C#;
– DryadLINQ – реализация от Microsoft Research на основе PLINQ и Dryad.

MapReduce

MapReduce выполняет распределенные вычисления на больших данных с помощью параллельного распределенного алгоритма, использующего большое количество обрабатывающих узлов. Каждое задание связано с двумя наборами задач, Map и Reduce, которые в основном используются для запроса и выбора данных в распределенной файловой системе Hadoop (HDFS).

MapReduce можно использовать почти везде, реализации уже имеются в *C#*, *Ruby*, *Java*, *Python*.

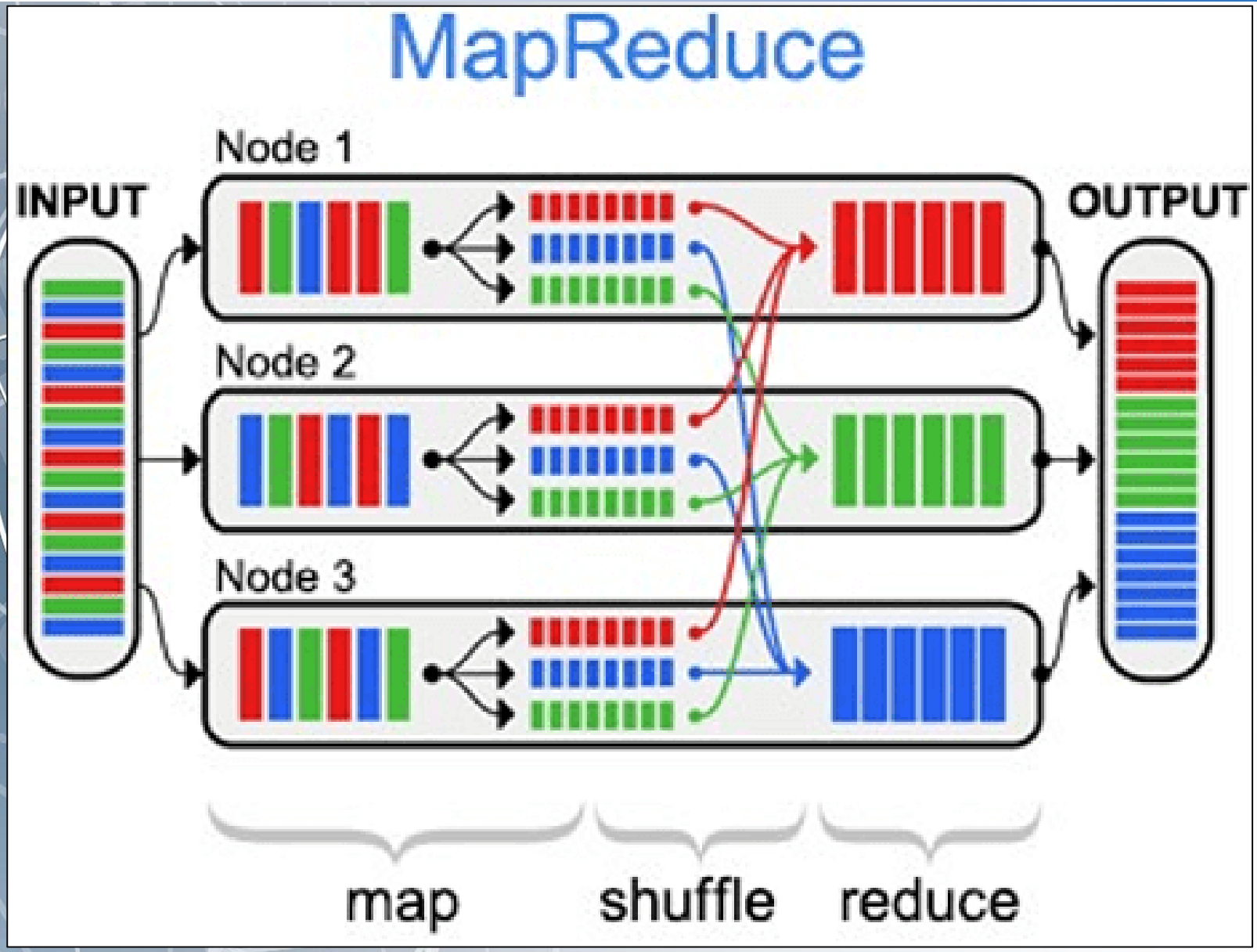
ПРИНЦИП РАБОТЫ MapReduce

map принимает на вход список значений и некую функцию, которую затем применяет к каждому элементу списка и возвращает новый список;

reduce (свёртка) — преобразует список к единственному атомарному значению при помощи заданной функции, которой на каждой итерации передаются новый элемент списка и промежуточный результат.

Для обработки данных в соответствии с вычислительной моделью MapReduce следует определить обе эти функции, указать имена входных и выходных файлов, а также параметры обработки.

MapReduce


MapReduce
Node 1
INPUT
OUTPUT
Node 2
Node 3
map
shuffle
reduce

MapReduce

Сама вычислительная модель состоит из трёхшаговой комбинации вышеприведенных функций:

Map – функция предназначена для выполнения указанной операции над каждым элементом концептуального списка, состоящего из нескольких независимых элементов, поэтому каждый элемент работает независимо, а исходный список не изменяется Потому что новый список создается здесь, чтобы сохранить новые ответы. То есть операция Map представляет собой высокопараллельную инфраструктуру MapReduce, функции Map и Reduce определяются в соответствии со структурой данных в форме (ключ, значение). Map получает пару ключ-значение в Data Domain, а затем возвращает список пар ключ-значение:

Map(k1,v1) → list(k2,v2)

Функция Map будет вызываться параллельно и применяться к каждой паре ключ-значение (с ключом K1) во входном наборе данных. Каждый вызов затем возвращает список ключей по парам K2. После этого платформа MapReduce собирает все пары ключ-значение с одним и тем же ключом (здесь k2) из всех списков и объединяет их для создания группы для каждого ключа. На этапе Map также, происходит операция разделения на группы.

Shuffle – рабочие узлы перераспределяют данные на основе ключей, ранее созданных функцией Map, таким образом, чтобы все данные одного ключа лежали на одном рабочем узле.

MapReduce

Reduce – это объединение элементов списка соответствующим образом. Хотя она не так параллельна, как функция Map, поскольку упрощение всегда дает простой ответ, а крупномасштабные операции относительно независимы, функция упрощения также полезна в высокопараллельной среде. Функция Reduce применяется к каждой группе параллельно, тем самым генерируя набор значений в одной и той же области данных:

Reduce(k2, list (v2)) → list(v3)

Reduce() выполнит соответствующую операцию уменьшения в соответствии с кодовой логикой, написанной программой, такую как подсчет и сложение в соответствии с одной и той же парой ключ-значение. Если объем данных, полученных стороной «Reduce», достаточно мал, он сохраняется непосредственно в памяти. Если объем данных превышает определенный процент от размера буфера, данные объединяются и записываются на диск.

MapReduce

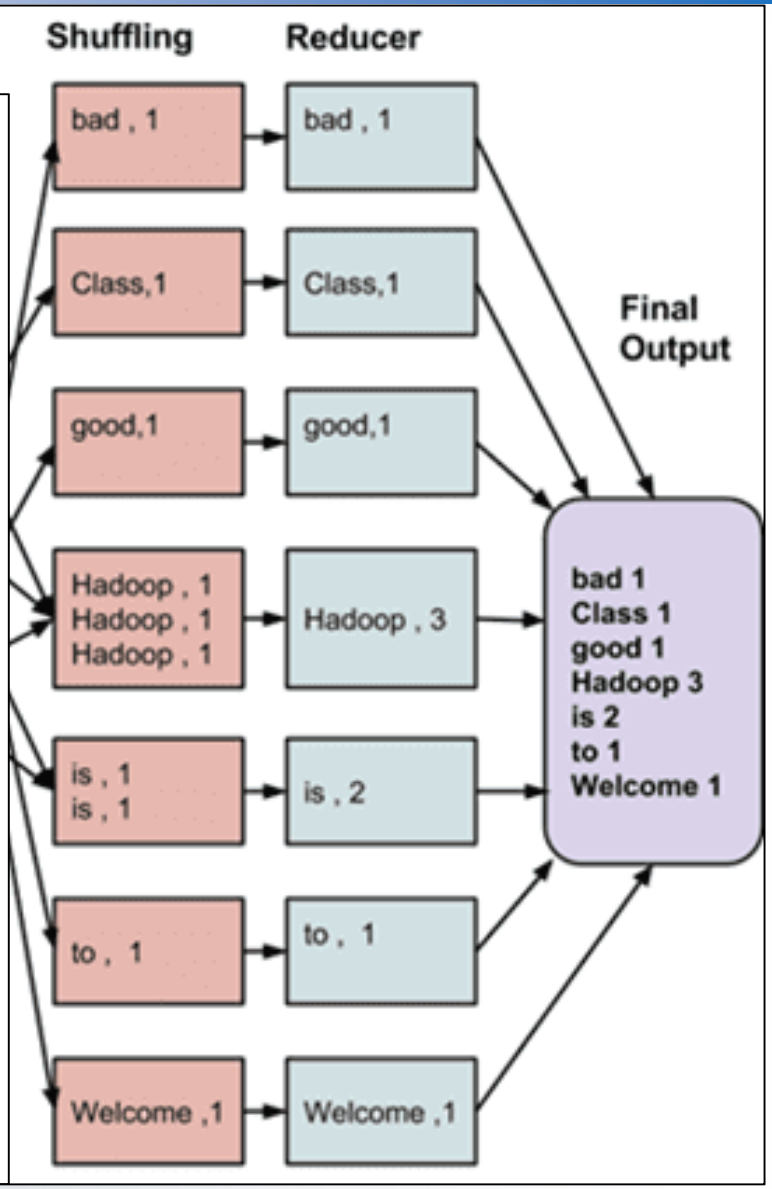
Дополнительные факты о MapReduce:

1) Все запуски функции **map** работают независимо и могут работать параллельно, в том числе на разных машинах кластера.

2) Все запуски функции **reduce** работают независимо и могут работать параллельно, в том числе на разных машинах кластера.

3) Shuffle - это по сути своей параллельная сортировка, поэтому также может работать на разных машинах кластера. **Пункты 1-3 позволяют выполнить принцип горизонтальной масштабируемости.**

4) Функция map, как правило, применяется на той же машине, на которой хранятся данные – это позволяет снизить передачу данных по сети (принцип локальности данных).



MapReduce

MapReduce можно назвать основой Big Data, т.к. именно данная технология позволяет обрабатывать огромные массивы информации параллельно в распределенных кластерах.

Эту вычислительную модель поддерживают множество различных коммерческих и свободных продуктов: Apache Hadoop, Spark, Greenplum, Hive, MongoDB, Phoenix, DryadLINQ и прочие Big Data фреймворки и библиотеки, написанные на разных языках программирования.

Достоинства MapReduce:

Возможность распределенного выполнения операций предварительной обработки (map) и свертки (reduce) большого объема данных. На практике количество одновременно исполняемых функций map ограничивается источником входных данных и числом используемых процессоров. Аналогичным образом множество узлов производят свертку (reduce) после того, как каждый из них обработал все результаты функции map с одним конкретным значением ключа.

Быстрота обработки больших объёмов данных за счет распределения операций по вышеописанному принципу. В частности, всего за пару часов MapReduce может отсортировать целый петабайт данных.

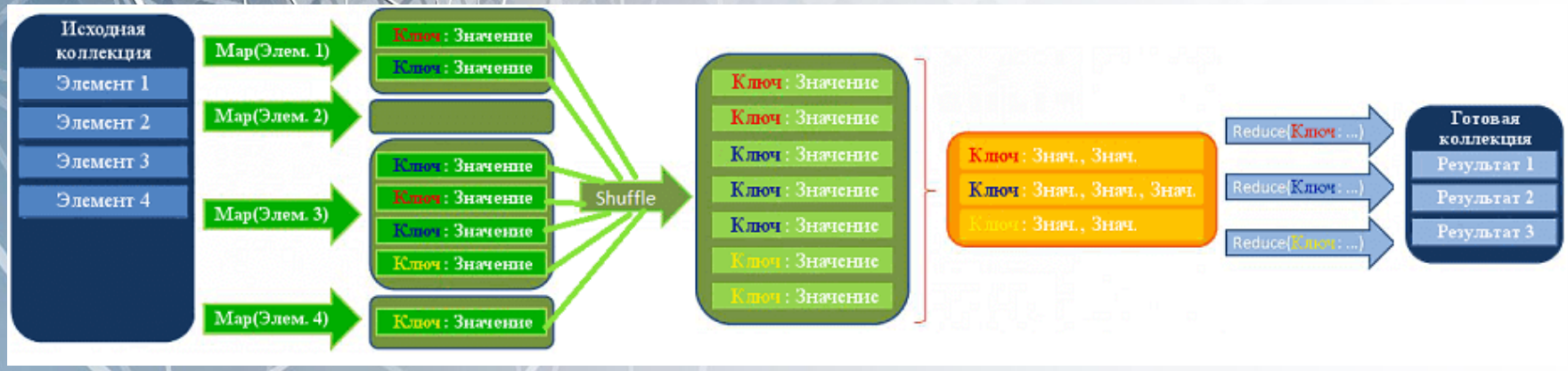
Отказоустойчивость и оперативное восстановления после сбоев: при отказе рабочего узла, производящего операцию map или reduce, его работа автоматически передается другому рабочему узлу в случае доступности входных данных для проводимой операции.

MapReduce

Недостатки MapReduce, обусловленные архитектурными особенностями этой вычислительной модели:

Недостаточно высокая производительность – классическая технология, в частности, реализованная в ядре Apache Hadoop, обрабатывает данные ациклично в пакетном режиме. При этом функции Reduce не запустятся до завершения всех процессов Map. Все операции проходят по циклу чтение-запись с жесткого диска, что влечет задержки в обработке информации.

Ограниченность применения – высокие задержки распределенных вычислений, приемлемые в пакетном режиме обработки, не позволяют использовать классический MapReduce для потоковой обработки в режиме реального времени, повторяющихся запросов и итеративных алгоритмов на одном и том же датасете, как в задачах машинного обучения. Для решения этой проблемы, свойственной Apache Hadoop, были созданы другие Big Data фреймворки, в частности, Apache Spark и Flink.



СПАСИБО ЗА ВНИМАНИЕ!