МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский государственный гуманитарный университет»
(ФГБОУ ВО «РГГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ НАУК И ТЕХНОЛОГИИ БЕЗОПАСНОСТИ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И БЕЗОПАСНОСТИ
Кафедра информационных технологий и систем

ТЕХНОЛОГИИ BIG DATA В ГУМАНИТАРНОЙ СФЕРЕ

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

09.03.03 Прикладная информатика
Код и наименование направления подготовки/специальности

Прикладная информатика в гуманитарной сфере
Наименование направленности (профиля)/ специализации

Уровень высшего образования: *бакалавриат*

Форма обучения: *очная*

РПД адаптирована для лиц
с ограниченными возможностями
здоровья и инвалидов

Москва 2023

ТЕХНОЛОГИИ BIG DATA В ГУМАНИТАРНОЙ СФЕРЕ
Рабочая программа дисциплины

Составители: к.с.-х.н., доц., зав. кафедрой Н.Ш. Шукенбаева

д-р техн. наук, проф. Н.З Султанов

Ответственный редактор
к.с.-х.н., доцент, заведующий кафедрой информационных технологий и систем Н.Ш. Шукенбаева

УТВЕРЖДЕНО
Протокол заседания кафедры
информационных технологий и систем
№ 8 от 15 апреля 2023 г.

ОГЛАВЛЕНИЕ

1 Пояснительная записка

1.1 Цель и задачи дисциплины

Цель дисциплины: изучение методов обработки структурированных и неструктурированных многообразных данных огромных объёмов для получения воспринимаемых человеком результатов.

Задачи:

- изучение основных принципов и методов хранения и управления данными формата Big Data;
- знакомство с методами организации и анализа данных формата Big Data.

1.2 Перечень планируемых результатов обучения по дисциплине, соотнесенных с индикаторами достижения компетенций

Компетенция (код и наименование)	**Индикаторы компетенций** (код и наименование)	**Результаты обучения**
ПК-7 Способен осуществлять разработку и ведение базы данных и поддержку информационного обеспечения решения прикладных задач	ПК-7.1. Знает методологию разработки информационного обеспечения, проектирования, создания и поддержки баз данных. ПК-7.2. Умеет осуществлять разработку и ведение баз данных в зависимости от конкретного назначения. ПК-7.3. Имеет практический опыт разработки и ведения проекта базы данных.	Знать основные принципы и методы хранения, управления, обработки, анализа данных формата Big Data, методологию разработки информационного обеспечения, проектирования, создания и поддержки хранилищ данных. Уметь строить модели для данных, хранящихся в распределенной файловой системе (Hadoop). Владеть методами прогнозного моделирования и анализа данных (алгоритм Map-reduce).
ПК-8 Способен принимать участие в организации ИТ-инфраструктуры и управлении информационной безопасностью	ПК-8.1. Знает способы организации ИТ-инфраструктуры, методы и приемы управления информационной безопасностью. ПК-8.2. Умеет организовывать ИТ-инфраструктуру предприятия и процессы управления информационной безопасностью. ПК-8.3. Владеет навыками организации ИТ-инфраструктуры и управления информационной безопасностью.	Знать способы организации больших данных, организацию их инфраструктуры, основные методы и средства управления информационной безопасностью при работе с большими данными. Уметь организовывать ИТ-инфраструктуру предприятия при работе с большими данными, выбирать методы и разрабатывать средства защиты информации при работе с большими данными Владеть навыками участия в организации ИТ- инфраструктуры и управления информационной безопасности при работе с большими данными

1.3 Место дисциплины структуре основной образовательной программы

Дисциплина «Технологии BIG DATA в гуманитарной сфере» является дисциплиной по выбору вариативной части блока Б1 учебного плана по направлению подготовки «Прикладная информатика».

Пререквизиты дисциплины: для освоения дисциплины необходимы знания, умения и навыки, сформированные в ходе изучения дисциплин: «Информационные технологии», «Информационные системы», «Базы данных», «Программирование», «Математика», «Интеллектуальные информационные системы в гуманитарной сфере» и др.

Постреквизиты дисциплины: в результате освоения дисциплины формируются знания, умения и навыки, необходимые для выполнения выпускной квалификационной работы.

2 Структура дисциплины

Общая трудоёмкость дисциплины составляет 3 з.е., 108 академических часов.
Вид итогового контроля – зачет с оценкой (8 семестр).

Структура дисциплины

Объем дисциплины в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Семестр	Тип учебных занятий	Количество часов
8	Лекции	**14**
8	Практические работы	**28**
	Всего:	**42**

Объем дисциплины (модуля) в форме самостоятельной работы обучающихся составляет 66 академических часов.

Самостоятельная работа включает:

- выполнение индивидуального творческого задания в восьмом семестре (ИТЗ);
- самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, профессиональных баз данных и информационных справочных систем; подготовка к практическим занятиям; подготовка к текущему контролю и промежуточной аттестации).

3 Содержание дисциплины

№	Наименование раздела дисциплины	Содержание
1	Введение в большие данные	1.1 Предметное поле дисциплины. Основная цель изучения дисциплины в структуре ООП ВО. Основные понятия, знания, умения и навыки, получаемые в ходе изучения дисциплины. Структура дисциплины и тематический план. Текущий контроль и промежуточные

		аттестации. Учебно-методическое обеспечение дисциплины (модуля): основная и дополнительная литература. Периодические издания и интернет-ресурсы. Предпосылки формирования тренда больших данных. 1.2 Основные взгляды на большие данные. 1.3 Основные вызовы больших данных (4V). 1.4 Качество исходных данных. 1.5 Драйверы рынка Big Data. 1.6 Структурированность. 1.7 Виртуализация. 1.8 Определение термина «большие данные». 1.9 Базовое представление о Map Reduce и Hadoop. 1.10 Представление о работе аналитика. 1.11 Big Data проекты. 1.12 Открытые проекты.
2	Введение в когнитивный анализ данных	2.1 История искусственного интеллекта. Интеллектуальная система. Задачи интеллектуальной системы. Процесс познания. Эмпирическая гипотеза. Фальсификация гипотез. Характеристики гипотез. 2.2 Классификация задач. Функция конкурентного сходства. Трехвходовая таблица. Задача редактирования. Задача распознавания. Задача кластеризации. Промежуточный случай. Выбор признаков. Заполнение пробелов и прогнозирование. Задачи комбинированного типа. Онтология FRIS-задач. Онтология Data Mining. Методы анализа дан-ных. Меры сходства. Конкурентное сходство. FRIS-функция. Относительные меры сходства. 2.3 Разработка алгоритма на базе FRIS-функции. Функция конкурентного сходства – единый базис для решения различных задач Data Mining. Алгоритм FRIS-Stolp. Классификация методов таксономии. Алгоритм FRIS-Tax. Задача частичного обучения. Требования к обобщенной классификации. Схема алгоритма FRIS-TDR. Эффект от ис-пользования информации из распознаваемой выборки. 2.4 Информативность и выбор признаков. Схема алгоритмов. Фильтрация. Селекция. Жадные алгоритмы. Итеративный алгоритм AdDel. Алгоритм FRiS-GRAD. Критерии информативность. Сравнение критериев. Устойчивость к помехам. 2.5 Обнаружение ошибок и заполнение пробелов в кубах данных. Алгоритм FRIS-ZET. Применение 3D-ZET.
3	Основы языка R	3.1 Общие сведения. Структура языка. Основные конструкции. Синтаксис. Кон-станты. Операторы. Выражения. Управляющие структуры. Структуры данных. Прими-тивные типы. Векторы, списки, матрицы, массивы. Таблицы «объект-свойство».
4	Инструменты DATA MINING	4.1 Weka. Возможности. Преимущества и недостатки. Области применения. 4.2 Визуализация. Цели, возможности, средства.

		Преимущества и недостатки. Области применения. 4.3 R как инструмент Data Mining. Хранение и доступ к данным по средствам Data Frame. Импорт и экспорт. 4.4 R как инструмент Data Mining. Классификация. Кластеризация. Регрессия. R и Hadoop. 4.5 R как инструмент Data Mining. Возможности библиотеки Pandas.
5	Технологии хранения больших данных	5.1 Новые хранилища данных. Свойства больших данных и ограничения RDBMS. ACID требования, CAP-теорема, BASE архитектура. NoSQL. MapReduce. Ключ-значение. Колоночные базы данных. Документо-ориентированные базы данных. Графовые базы данных.

4 Образовательные технологии

Для проведения учебных занятий по дисциплине используются различные образовательные технологии. Для организации учебного процесса может быть использовано электронное обучение и (или) дистанционные образовательные технологии.

5 Оценка планируемых результатов обучения

5.1 Система оценивания

Форма контроля	Макс. количество баллов	
	За одну работу	Всего
Текущий контроль:		
- защита практических работ	12 баллов	60 баллов
Промежуточная аттестация (зачет с оценкой)		40 баллов
Итого за семестр		100 баллов

Полученный совокупный результат конвертируется в традиционную шкалу оценок и в шкалу оценок Европейской системы переноса и накопления кредитов (European Credit Transfer System; далее – ECTS) в соответствии с таблицей:

100-балльная шкала	Традиционная шкала		Шкала ECTS
91 – 100	отлично	зачтено	A
83 – 90	хорошо		B
75 – 82			C
61 – 74	удовлетворительно		D
51 – 60			E
31 – 50			FX

0 – 30	неудовлетворительно	не зачтено	F

5.2 Критерии выставления оценки по дисциплине

Баллы/ Шкала ECTS	**Оценка по дисциплине**	**Критерии оценки результатов обучения по дисциплине**
100-83/ A,B	«отлично»/ «зачтено (отлично)»/ «зачтено»	Выставляется обучающемуся, если он глубоко и прочно усвоил теоретический и практический материал, может продемонстрировать это на занятиях и в ходе промежуточной аттестации. Обучающийся исчерпывающе и логически стройно излагает учебный материал, умеет увязывать теорию с практикой, справляется с решением задач профессиональной направленности высокого уровня сложности, правильно обосновывает принятые решения. Свободно ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «высокий».
82-68/ C	«хорошо»/ «зачтено (хорошо)»/ «зачтено»	Выставляется обучающемуся, если он знает теоретический и практический материал, грамотно и по существу излагает его на занятиях и в ходе промежуточной аттестации, не допуская существенных неточностей. Обучающийся правильно применяет теоретические положения при решении практических задач профессиональной направленности разного уровня сложности, владеет необходимыми для этого навыками и приёмами. Достаточно хорошо ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «хороший».
67-50/ D,E	«удовлетворительно»/ «зачтено (удовлетворительно)»/ «зачтено»	Выставляется обучающемуся, если он знает на базовом уровне теоретический и практический материал, допускает отдельные ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает определённые затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, владеет необходимыми для этого базовыми навыками и приёмами. Демонстрирует достаточный уровень знания учебной литературы по дисциплине.

Баллы/ Шкала ECTS	Оценка по дисциплине	Критерии оценки результатов обучения по дисциплине
		Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «достаточный».
49-0/ F,FX	«неудовлетворительно»/ не зачтено	Выставляется обучающемуся, если он не знает на базовом уровне теоретический и практический материал, допускает грубые ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает серьёзные затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, не владеет необходимыми для этого навыками и приёмами. Демонстрирует фрагментарные знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции на уровне «достаточный», закреплённые за дисциплиной, не сформированы.

При оценивании защиты практической работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 1-5 балла;

- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 6-9 баллов;

- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность -10-12 баллов.

Промежуточная аттестация (зачет с оценкой).

При проведении промежуточной аттестации студент должен ответить на 2 вопроса теоретического характера либо пройти тестирование.

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (1-10 баллов);

- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (11-20 баллов);

- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (21-30 баллов);

- теоретическое содержание освоено полностью, ответ построен по собственному плану (31-40 баллов).

5.3 Оценочные средства (материалы) для текущего контроля успеваемости, промежуточной аттестации обучающихся по дисциплине *(модулю)*

Вопросы к текущей аттестации

1. Большинство данных в мире в последние годы содержалось:
    i. +В цифровом виде
    ii. В аналоговом виде
2. В каком веке произошёл перевес объёмов накопленных человечеством данных в сторону цифровых?
    i. 20 (число)
3. Объём накопленных человечеством цифровых данных на 2012 год измеряется:
    i. Петабайтами
    ii. +Зеттабайтами
    iii. Экзабайтами
    iv. Йоттабайтами
4. Сколько Петабайт в Зеттабайте?
    i. 1024 (число)
5. Укажите фактор, способствовавший появлению тренда больших данных
    i. +Маркетинговые кампании крупных корпораций
    ii. +Снижение издержек на хранение данных
    iii. Появление новых технологий обработки потоковых данных
    iv. Выпуск баз данных с обработкой данных в памяти
6. Какие вероятные разочарования тренда больших данных?
    i. Из-за угрозы безопасности личной жизни (privacy) граждан будут усложнены процедуры сбора данных, что приведёт к падению ценности больших данных.
7. Отметьте значимые события, повлиявшие на формирование тренда больших данных:
    i. + Разработка Hadoop
    ii. + Изобретение принципа MapReduce
    iii. Разработка языка Pyhton
    iv. Победа Deepblue в матче с Г.Каспаровым.
8. Выберите верный ответ
    i. Большие данные – это обработка или хранение более 1 Тб информации.
    ii. +Проблема больших данных – это такая проблема, когда при существующих технологиях хранения и обработки сущностная обработка данных затруднена или невозможна.
    iii. Большие данные – это огромная PR-акция крупных вендоров и не более того.
    iv. Большие данные – это явление, когда цифровые данные наиболее полно представляют изучаемый объект.
9. Выберите неверный ответ:
    i. +Большие данные – это данные объёма свыше 1 Тб
    ii. Проблема больших данных – это проблема, когда при существующих технологиях хранения и обработки сущностная обработка данных затруднена или невозможна.
    iii. Большие данные – это тренд в области ИТ, подогреваемый маркетинговыми кампаниями крупных вендоров.
    iv. Большие данные как правило не структурированы.
10. Отметьте те из вариантов, в которых данные структурированы:
    i. Данные о продажах компании, представленные в виде помесячных отчётов в формате MS Word.
    ii. +Таблица с ежедневными показаниями температуры помещения за год в файле формата csv.
    iii. Текст педагогической поэмы А.С. Макаренко, представленный в формате PDF.
    iv. Библиотека фильмов, представленных в формате mpeg4 на одном жестком диске.
11. Перечислите четыре основных характеристики Big Data:

i. Virtualization, Volume, Variability, Vehicle
ii. +Variety, Velocity, Volume, Value
iii. Verification, Volume, Velocity, Visualization
iv. Video, Value, Variety, Volume

12. Выберите неверное высказывание:
i. +Большие объёмы данных приводят к слабой их структуризации, поэтому появляется такое разнообразие данных.
ii. Увеличившаяся производительность телекоммуникационных каналов привела к росту объёмов передаваемой информации.
iii. Удешевление систем хранения на единицу информации привело к росту рынка больших данных.
iv. Большое разнообразие источников данных

13. Отметьте неверное понимание Variety в контексте характеристик Big Data:
i. +Высокая скорость генерирования данных.
ii. +Разные типы данных в колонках таблиц реляционных СУБД.
iii. +Разнообразие отраслей, являющихся источниками данных.
iv. Разнообразие типов данных, включающих в себя структурированные, полуструктурированные и неструктурированные.

14. Принцип MapReduce состоит в том, чтобы
i. +Производить вычисления на узлах, где информация изначально была сохранена
ii. +Использовать вычислительные мощности систем хранения
iii. Использовать функциональное программирование для решения задач массивно-параллельной обработки

15. Выберите одно неверное высказывание про MapReduce:
i. +Интерфейс для массово-параллельной обработки данных, где вычисления производятся на узлах, где информация изначально была сохранена
ii. MapReduce – это две операции: распределения и сборки данных iii. +MapReduce был придуман разработчиками Hadoop
iv. MapReduce был анонсирован разработчиками Google

16. Каков теоретический прирост производительности при подсчёте числа слов в тексте при работе MapReduce при переходе от одного узла к двум:
i. 2

17. Какие из следующих технологий СУБД не используют принцип MapReduce
i. Hadoop
ii. Cassandra
iii. HDInsight
iv. +Redis

18. Какие СУБД полностью полагаются на оперативную память при хранении информации:
i. +Oracle Exalytics
ii. +SAP HANA
iii. BigTable
iv. HBase

19. В чём преимущество колоночно-ориентированных СУБД?
i. Они позволяют выполнять более сложные SQL-запросы по сравнению с реляционными СУБД
ii. +Они позволяют динамически дополнять содержание записей новыми полями
iii. Они имеют более гибкие возможности аналитики.
iv. Они позволяют эффективно делать межколоночные сравнения.

20. Для чего аналитику необходима «песочница»?
i. +Для высокопроизводительной аналитики за счёт использования оперативной памяти и inDB операций.

ii. Для хранения всех полученных от заказчика данных. iii. Для построения отчётов о результатах анализа

iv. +Для снижения затрат, связанных с репликацией данных

21. Какие из следующих средств разумно использовать для анализа данных, представленных единственным csv-файлом размера более 100Гб:

i. +Hadoop

ii. Data Warehouse

iii. +«Песочница»

iv. +Python

22. Выберите верное утверждение:

i. Data Warehouse создаются для проверки гипотез при анализе больших данных. 10

ii. +«Песочница» используется для снижения нагрузки на основной Data Warehouse.

iii. Каждый Data Warehouse должен содержать «песочницу».

iv. «Песочница» необходима для любого процесса аналитики.

23. Расставьте последовательность этапов проекта аналитики в соответствии с CRISP-DM.

i. Понимание бизнеса (Business understanding)

ii. Понимание данных (Data Understanding)

iii. Подготовка данных (Data Preparation)

iv. Моделирование (Modeling)

v. Оценка (Evaluation)

vi. Внедрение (Deployment)

24. На каком из этапов процесса CRISP-DM происходит проверка гипотез?

i. Понимание бизнеса (Business understanding)

ii. Понимание данных (Data Understanding)

iii. +Моделирование (Modeling)

iv. Оценка (Evaluation)

25. Вы являетесь владельцем и аналитиком в компании из 10 человек, в которой требуется проанализировать продажи за 1 год (1 млн. продаж). Какие из этапов CRISP-DM можно опустить:

i. +Понимание бизнеса (Business understanding)

ii. Подготовка данных (Data Preparation)

iii. Моделирование (Modeling)

iv. Оценка (Evaluation)

26. Пример благоразумного использования Hadoop

i. Анализ 10 Гб данных.

ii. Ежедневное сохранение данных температуры, поступающих со всех городов России (по одному показанию на город, всего городов 1100 шт).

iii. +Посекундное сохранение данных температуры, поступающих со всех городов России (по одному показанию на город, всего городов 1100 шт).

iv. Построение графика пульса пациента в реальном времени.

27. Начиная с каких размеров данных обоснованно применение кластера Hadoop для хранения данных?

i. 100Гб

ii. 1Тб

iii. +100Тб

iv. +1Пб

28. Hadoop – это:

i. +Набор утилит, и программный каркас для выполнения распределённых программ, работающих на кластерах.

ii. Распределённая СУБД, позволяющая обрабатывать большие данные.
iii. Язык выполнения заданий в парадигме MapReduce.
iv. Распределённая файловая система, предназначенная для хранения файлов большого объёма.

Вопросы к зачету с оценкой

1. История возникновения термина «Большие данные».
2. Источники больших данных.
3. Развитие больших данных. Цикл Гартнера в развитии информационных технологий.
4. Основы Hadoop. Базовый набор компонентов Hadoop.
5. Базовые модули фреймворка Apache Framework.
6. Файловая система Hadoop Distributed File System (HDFS). HDFS и HDFS2.
7. MapReduce Framework и YARN.
8. Окружение Hadoop: YARN, Tez, Spark, HRS.
9. Базовые приложения Hadoop: Pig, HIVE, HBASE.
10. HBase – распределенная и масштабируемая база данных для работы с большими данными
11. Сравнение HBase и HDFS.
12. Модель данных в HBase.
13. Закономерности формирования BigTable.
14. Особенности отображения временных меток в BigTable.
15. HIVE – хранилище больших данных: архитектура, работа с данными.
16. Анализ данных с помощью PIG. Команды PIG.
17. Логистический анализ с использованием Splunk.
18. Работа с данными с помощью Spark DataFrames и Spark SQL.
19. Основы машинного обучения. «Осмысление данных» для машинного обучения.
20. Основы KNIME.
21. Классификация инструментов, техник и алгоритмов машинного обучения.
22. Ассоциативные правила.
23. Кластерный анализ.
24. Кластеризация в KNIME.
25. Кластеризация в Spank.
26. Социальные сети и базы знаний как источники больших данных.
27. Подходы к обработке больших данных, структурированных в виде графов.
28. Open Source-решения для обработки больших объемов графовых данных.
29. Apache Giraph и GraphLab.
30. MapReduce и обработка крупномасштабных графов.
31. MapReduce против Giraph.

6 Учебно-методическое и информационное обеспечение дисциплины

6.1 Список источников литературы

Основная литература

1. Зарова, Е. В. Методы Data mining в обработке и анализе статистических данных (решения в R) : монография / Е.В. Зарова. — Москва : ИНФРА-М, 2021. — 232 с. : ил. - ISBN 978-5-16-016814-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1240276
2. Форман, Д. Много цифр. Анализ больших данных при помощи Excel / Форман Д.; Пер. с англ. Соколовой А. - Москва :Альпина Пабл., 2016. - 461 с. ISBN 978-5-9614-5032-3. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/551044
3. Блануца, В. И. Социально-экономическое районирование в эпоху больших данных: Монография / Блануца В.И. - Москва :НИЦ ИНФРА-М, 2019. - 194 с.

(Научная мысль) ISBN 978-5-16-013259-4. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/1014727

4. Гуриков, С. Р. Основы алгоритмизации и программирования на Python : учеб. пособие / С.Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2018. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-00091-487-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/924699

Дополнительная литература

1. Кулаичев, А. П. Методы и средства комплексного статистического анализа данных : учеб. пособие / А.П. Кулаичев. — 5-е изд., перераб. и доп. — Москва : ИНФРА-М, 2018. — 484 с. — (Высшее образование: Бакалавриат). — www.dx.doi.org/10.12737/25093. - ISBN 978-5-16-103357-9. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/975598

2. Кузьмич, Р.И. Модификации метода логического анализа данных для задач классификации : монография / Р.И. Кузьмич, И.С. Масич. - Красноярск : Сиб. федер. ун-т, 2018. - 180 с. - ISBN 978-5-7638-3698-1. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/1031829
3. Жуков, Р. А. Язык программирования Python: практикум : учеб. пособие / Р.А. Жуков. — Москва : ИНФРА-М, 2019. — 216 с. + Доп. материалы [Электронный ресурс; Режим доступа: http://znanium.com]. — (Высшее образование: Бакалавриат). — www.dx.doi.org/10.12737/textbook_5cb5ca35aaa7f5.89424805. - ISBN 978-5-16-107207-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1000002
4. Золотарюк, А. В. Язык и среда программирования R : учеб. пособие / А.В. Золотарюк. — Москва : ИНФРА-М, 2019. — 162 с. — (Высшее образование: Бакалавриат). — www.dx.doi.org/10.12737/textbook_5b8fdb0bd795c4.69435980. - ISBN 978-5-16-014388-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/978863

6.2 Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Электронно-библиотечная система «Знаниум» - Режим доступа: http://znanium.com
2. - Информационная система «Единое окно доступа к образовательным ресурсам». - Режим доступа: http://window.edu.ru
3. Онлайн-энциклопедия. - Режим доступа: http://encyclopedia.ru
4. Электронный справочник «Информио» для высших учебных заведений. - Режим доступа: http://www.informio.ru
5. КонсультантПлюс. Правовая поддержка. - Режим доступа: http://www.consultant.ru/
6. Национальный открытый университет «ИНТУИТ». - Режим доступа: https://www.intuit.ru/
7. Сайт Microsoft - Режим доступа: https://msdn.microsoft.com/ru-ru/library/
8. Научная библиотека РГГУ - Режим доступа: http://liber.rsuh.ru/
9. «CITFORUM»: Аналитическая информация в сфере IT. - Режим доступа: http://citforum.ru/

6.3 Профессиональные базы данных и информационно-справочные системы

Доступ к профессиональным базам данных: https://liber.rsuh.ru/ru/bases

Информационные справочные системы:
1. Консультант Плюс
2. Гарант

7 Материально-техническое обеспечение дисциплины *(модуля)*

Для материально-технического обеспечения дисциплины необходимы:

- *для лекций:*
- учебная аудитория,
- доска,
- проектор (стационарный или переносной),
- компьютер или ноутбук,
- программное обеспечение (ПО).

Перечень программного обеспечения (ПО)

№п/п	Наименование ПО	Способ распространения
1	Microsoft Office 2010 Pro	лицензионное
2	Windows 10	лицензионное
3	Kaspersky Endpoint Security	лицензионное
4	Платформа ZOOM	лицензионное

- *для практических занятий:*
- лаборатория,
- доска,
- проектор (стационарный или переносной),
- компьютер или ноутбук для преподавателя,
- компьютеры для обучающихся,
- выход в Интернет,
- программное обеспечение (ПО).

Перечень программного обеспечения (ПО)

Наименование ПО	**Способ распространения**
Windows 10	лицензионное
Microsoft Office 2010 Pro	лицензионное
Mozilla Firefox	свободно распространяемое
Kaspersky Endpoint Security	лицензионное
Cloudera Manager Express	бесплатная свободно распространяемая версия
Платформа ZOOM	лицензионное

Профессиональные полнотекстовые базы данных:
1. Национальная электронная библиотека (НЭБ) www.rusneb.ru
2. ELibrary.ru Научная электронная библиотека www.elibrary.ru
3. Электронная библиотека Grebennikon.ru www.grebennikon.ru
4. Cambridge University Press
5. ProQuest Dissertation & Theses Global

6. SAGE Journals
7. Taylor and Francis
8. JSTOR

Информационные справочные системы:
1. Консультант Плюс
2. Гарант

8 Обеспечение образовательного процесса для лиц с ограниченными возможностями здоровья и инвалидов

В ходе реализации дисциплины используются следующие дополнительные методы обучения, текущего контроля успеваемости и промежуточной аттестации обучающихся в зависимости от их индивидуальных особенностей:

- для слепых и слабовидящих: лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением; письменные задания выполняются на компьютере со специализированным программным обеспечением или могут быть заменены устным ответом; обеспечивается индивидуальное равномерное освещение не менее 300 люкс; для выполнения задания при необходимости предоставляется увеличивающее устройство; возможно также использование собственных увеличивающих устройств; письменные задания оформляются увеличенным шрифтом; экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.
- для глухих и слабослышащих: лекции оформляются в виде электронного документа, либо предоставляется звукоусиливающая аппаратура индивидуального пользования; письменные задания выполняются на компьютере в письменной форме; экзамен и зачёт проводятся в письменной форме на компьютере; возможно проведение в форме тестирования.
- для лиц с нарушениями опорно-двигательного аппарата: лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением; письменные задания выполняются на компьютере со специализированным программным обеспечением; экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

При необходимости предусматривается увеличение времени для подготовки ответа.

Процедура проведения промежуточной аттестации для обучающихся устанавливается с учётом их индивидуальных психофизических особенностей. Промежуточная аттестация может проводиться в несколько этапов.

При проведении процедуры оценивания результатов обучения предусматривается использование технических средств, необходимых в связи с индивидуальными особенностями обучающихся. Эти средства могут быть предоставлены университетом, или могут использоваться собственные технические средства.

Проведение процедуры оценивания результатов обучения допускается с использованием дистанционных образовательных технологий.

Обеспечивается доступ к информационным и библиографическим ресурсам в сети Интернет для каждого обучающегося в формах, адаптированных к ограничениям их здоровья и восприятия информации:

- для слепых и слабовидящих: в печатной форме увеличенным шрифтом, в форме электронного документа, в форме аудиофайла.
- для глухих и слабослышащих: в печатной форме, в форме электронного документа.

• для обучающихся с нарушениями опорно-двигательного аппарата: в печатной форме, в форме электронного документа, в форме аудиофайла.

Учебные аудитории для всех видов контактной и самостоятельной работы, научная библиотека и иные помещения для обучения оснащены специальным оборудованием и учебными местами с техническими средствами обучения:

• для слепых и слабовидящих: устройством для сканирования и чтения с камерой SARA CE; дисплеем Брайля PAC Mate 20; принтером Брайля EmBraille ViewPlus;

• для глухих и слабослышащих: автоматизированным рабочим местом для людей с нарушением слуха и слабослышащих; акустический усилитель и колонки;

• для обучающихся с нарушениями опорно-двигательного аппарата: передвижными, регулируемыми эргономическими партами СИ-1; компьютерной техникой со специальным программным обеспечением.

В ходе реализации дисциплины используются следующие дополнительные методы обучения, текущего контроля успеваемости и промежуточной аттестации обучающихся в зависимости от их индивидуальных особенностей:

• для слепых и слабовидящих:

- лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением;
- письменные задания выполняются на компьютере со специализированным программным обеспечением, или могут быть заменены устным ответом;
- обеспечивается индивидуальное равномерное освещение не менее 300 люкс;
- для выполнения задания при необходимости предоставляется увеличивающее устройство; возможно также использование собственных увеличивающих устройств;
- письменные задания оформляются увеличенным шрифтом;
- экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

• для глухих и слабослышащих:

- лекции оформляются в виде электронного документа, либо предоставляется звукоусиливающая аппаратура индивидуального пользования;
- письменные задания выполняются на компьютере в письменной форме;
- экзамен и зачёт проводятся в письменной форме на компьютере; возможно проведение в форме тестирования.

• для лиц с нарушениями опорно-двигательного аппарата:

- лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением;
- письменные задания выполняются на компьютере со специализированным программным обеспечением;
- экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

При необходимости предусматривается увеличение времени для подготовки ответа.

Процедура проведения промежуточной аттестации для обучающихся устанавливается с учётом их индивидуальных психофизических особенностей. Промежуточная аттестация может проводиться в несколько этапов.

При проведении процедуры оценивания результатов обучения предусматривается использование технических средств, необходимых в связи с индивидуальными особенностями обучающихся. Эти средства могут быть предоставлены университетом, или могут использоваться собственные технические

средства.

Проведение процедуры оценивания результатов обучения допускается с использованием дистанционных образовательных технологий.

Обеспечивается доступ к информационным и библиографическим ресурсам в сети Интернет для каждого обучающегося в формах, адаптированных к ограничениям их здоровья и восприятия информации:

- для слепых и слабовидящих:
- в печатной форме увеличенным шрифтом;
- в форме электронного документа;
- в форме аудиофайла.
- для глухих и слабослышащих:
- в печатной форме;
- в форме электронного документа.
- для обучающихся с нарушениями опорно-двигательного аппарата:
- в печатной форме;
- в форме электронного документа;
- в форме аудиофайла.

Учебные аудитории для всех видов контактной и самостоятельной работы, научная библиотека и иные помещения для обучения оснащены специальным оборудованием и учебными местами с техническими средствами обучения:

- для слепых и слабовидящих:
  - устройством для сканирования и чтения с камерой SARA CE;
  - дисплеем Брайля PAC Mate 20;
  - принтером Брайля EmBraille ViewPlus;
- для глухих и слабослышащих:
  - автоматизированным рабочим местом для людей с нарушением слуха и слабослышащих;
  - акустический усилитель и колонки;
- для обучающихся с нарушениями опорно-двигательного аппарата:
  - передвижными, регулируемыми эргономическими партами СИ-1;
  - компьютерной техникой со специальным программным обеспечением.

9 Методические материалы

9.1 Планы практических занятий

Практическая работа № 1 Принципы работы с большими данными, парадигма MapReduce

Цель работы

Рассмотреть принципы работы с большими данными и познакомиться с парадигмой mapReduce.

Задачи

- Изучить теоретические сведения о Big Data;
- Рассмотреть основные принципы, которым следуют все средства и парадигмы работы с большими данными;
- Рассмотреть парадигму MapReduce и разобрать несколько задач, в которой она может быть применена.

Ход Работы

Теоретические сведения

История вопроса и определение термина

Термин Big Data появился сравнительно недавно. Google Trends показывает начало активного роста употребления словосочетания начиная с 2011 года :

При этом уже сейчас термин используется очень часто. Особенно часто не по делу термин используют маркетологи. Так что же такое Big Data на самом деле?

Встречаются разные определения:

· Big Data – это когда данных больше, чем 100Гб (500Гб, 1ТБ, кому что нравится)

· Big Data – это такие данные, которые невозможно обрабатывать в Excel

· Big Data – это такие данные, которые невозможно обработать на одном компьютере

И даже такие:

· Big Data – это вообще любые данные.

· Big Data не существует, ее придумали маркетологи.

Определение с wikipedia:

Большие данные (англ. big data) — серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста, распределения по многочисленным узлам вычислительной сети, сформировавшихся в конце 2000-х годов, альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence.

Таким образом под Big Data понимается не какой-то конкретный объём данных и даже не сами данные, а методы их обработки, которые позволяют распредёлено обрабатывать информацию. Эти методы можно применить как к огромным массивам данных (таким как содержание всех страниц в интернете), так и к маленьким.

Несколько примеров того, что может быть источником данных, для которых необходимы методы работы с большими данными:

· Логи поведения пользователей в интернете

· GPS-сигналы от автомобилей для транспортной компании

· Данные, снимаемые с датчиков в большом адронном коллайдере

· Оцифрованные книги в Российской Государственной Библиотеке

· Информация о транзакциях всех клиентов банка

· Информация о всех покупках в крупной ритейл сети и т.д.

Количество источников данных стремительно растёт, а значит технологии их обработки становятся всё более востребованными.

Принципы работы с большими данными

Исходя из определения Big Data, можно сформулировать основные принципы работы с такими данными:

1. Горизонтальная масштабируемость. Поскольку данных может быть сколь угодно много – любая система, которая подразумевает обработку больших данных, должна быть расширяемой. В два раза вырос объём данных – в два раза увеличили возможности аппаратной части в кластере и система продолжила работать.

2. Отказоустойчивость. Принцип горизонтальной масштабируемости подразумевает, что машин в кластере может быть много. Например, Hadoop-кластер Yahoo имеет более 42000 машин (по этой ссылке можно посмотреть размеры кластера в разных организациях). Это означает, что часть этих машин будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий.

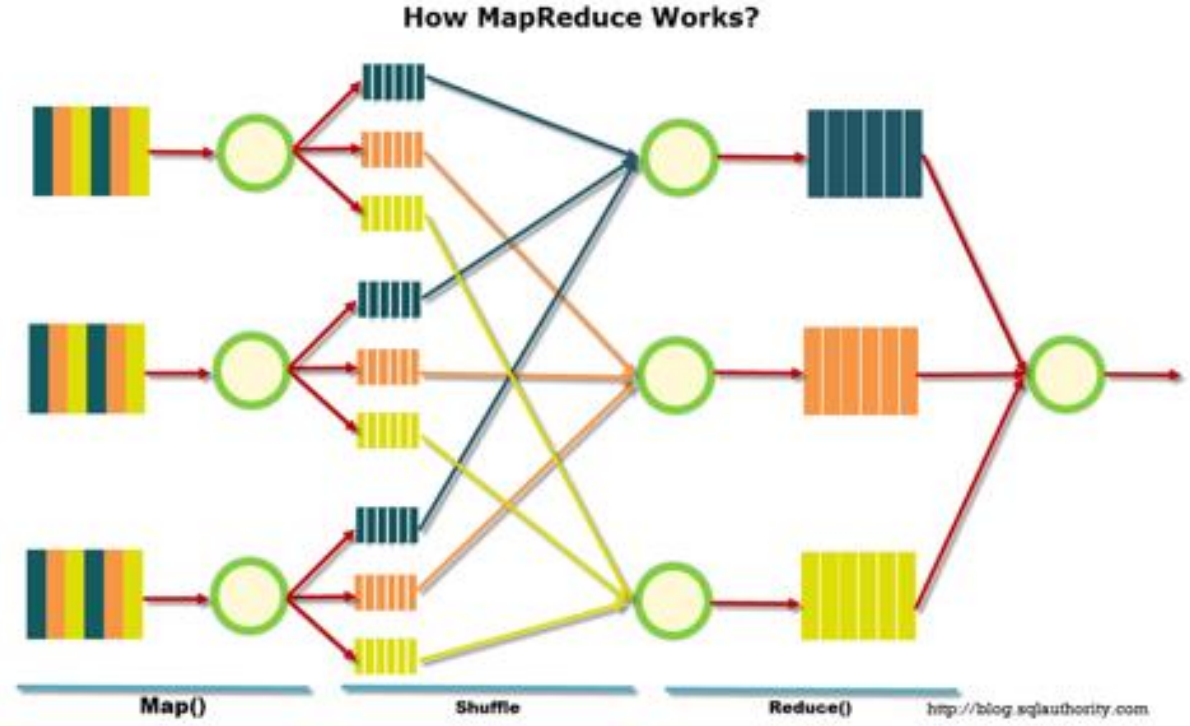
3. Локальность данных. В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном

сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования BigData-решений является принцип локальности данных – по возможности обрабатываем данные на той же машине, на которой их храним.

Все современные средства работы с большими данными так или иначе следуют этим трём принципам. Для того, чтобы им следовать – необходимо придумывать какие-то методы, способы и парадигмы разработки средств разработки данных. Один из самых классических методов и разберем в этой работе.

MapReduce

MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объёмов данных на компьютерных кластерах. MapReduce неплохо иллюстрируется следующей картинкой :



MapReduce предполагает, что данные организованы в виде некоторых записей. Обработка данных происходит в три стадии:

1. Стадия Map. На этой стадии данные предобрабатываются при помощи функции map, которую определяет пользователь. Работа этой стадии заключается в предобработке и фильтрации данных. Работа очень похожа на операцию map в функциональных языках программирования – пользовательская функция применяется к каждой входной записи.

Функция map, примененная к одной входной записи и выдаёт множество пар ключ-значение. Множество – т.е. может выдать только одну запись, может не выдать ничего, а может выдать несколько пар ключ-значение. Что будет находится в ключе и в значении – решать пользователю, но ключ – очень важная вещь, так как данные с одним ключом в будущем попадут в один экземпляр функции reduce.

2. Стадия Shuffle. Проходит незаметно для пользователя. В этой стадии вывод функции map «разбирается по корзинам» – каждая корзина соответствует одному ключу вывода стадии map. В дальнейшем эти корзины послужат входом для reduce.

3. Стадия Reduce. Каждая «корзина» со значениями, сформированная на стадии shuffle, попадает на вход функции reduce.

Функция reduce задаётся пользователем и вычисляет финальный результат для отдельной «корзины». Множество всех значений, возвращённых функцией reduce(), является финальным результатом MapReduce-задачи.

Несколько дополнительных фактов про 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 внутри себя представляет параллельную сортировку, 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).

5) MapReduce – это всегда полное сканирование данных, никаких индексов нет. Это означает, что MapReduce плохо применим, когда ответ требуется очень быстро.

Практическая часть

Примеры задач, эффективно решаемых при помощи MapReduce

Word Count

Классическая задача – Word Count. Задача формулируется следующим образом: имеется большой корпус документов. Задача – для каждого слова, хотя бы один раз встречающегося в корпусе, посчитать суммарное количество раз, которое оно встретилось в корпусе.

Решение:

Раз имеем большой корпус документов – пусть один документ будет одной входной записью для MapRreduce–задачи. В MapReduce мы можем только задавать пользовательские функции, что мы и сделаем (будем использовать python-like псевдокод):

```
def map(doc):
for word in doc:
        yield word, 1
def reduce(word, values):
yield word, sum(values)
```

Функция map превращает входной документ в набор пар (слово, 1), shuffle прозрачно для нас превращает это в пары (слово, [1,1,1,1,1,1]), reduce суммирует эти единички, возвращая финальный ответ для слова.

Обработка логов рекламной системы

Второй пример - из реальной практики Data-Centric Alliance.

Задача: имеется csv-лог рекламной системы вида:

```
<user_id>,<country>,<city>,<campaign_id>,<creative_id>,<payment></p>
11111,RU,Moscow,2,4,0.3
22222,RU,Voronezh,2,3,0.2
13413,UA,Kiev,4,11,0.7
…
```

Необходимо рассчитать среднюю стоимость показа рекламы по городам России.

Решение:

```
def map(record):
user_id, country, city, campaign_id, creative_id, payment = record.split(",")
payment=float(payment)
if country == "RU":
        yield city, payment
```

```
def reduce(city, payments):
yield city, sum(payments)/len(payments)
```

Функция map проверяет, нужна ли нам данная запись – и если нужна, оставляет только нужную информацию (город и размер платежа). Функция reduce вычисляет финальный ответ по городу, имея список всех платежей в этом городе.

Практическая работа № 2-3 Hadoop

Цель работы

Познакомиться с фреймворком Hadoop и MapReduce

Задачи

- Изучить теоретические сведения о Hadoop и MapReduce;
- Установить и настроить одноузловой кластер Hadoop;
- Познакомиться с приложением на основе MapReduce;
- Научиться запускать программы MapReduce на Hadoop.

Ход Работы

Теоретические сведения

Как известно парадигму MapReduce предложила компания Google в 2004 году в своей статье MapReduce: Simplified Data Processing on Large Clusters. Поскольку предложенная статья содержала описание парадигмы, но реализация отсутствовала – несколько программистов из Yahoo предложили свою реализацию в рамках работ над web-краулером nutch.

Изначально Hadoop был, в первую очередь, инструментом для хранения данных и запуска MapReduce-задач, сейчас же Hadoop представляет собой большой стек технологий, так или иначе связанных с обработкой больших данных (не только при помощи MapReduce).

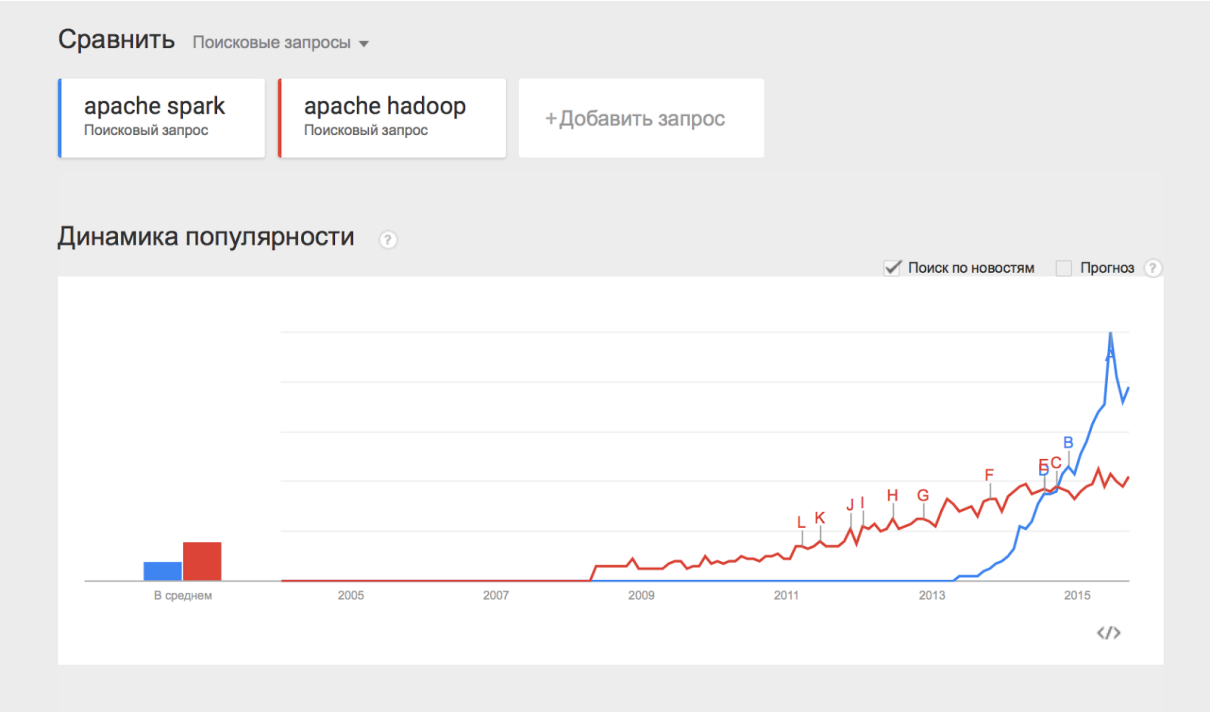
Основными (core) компонентами Hadoop являются:

- Hadoop Distributed File System (HDFS) – распределённая файловая система, позволяющая хранить информацию практически неограниченного объёма.
- Hadoop YARN – фреймворк для управления ресурсами кластера и менеджмента задач, в том числе включает фреймворк MapReduce.
- Hadoop common

Также существует большое количество проектов непосредственно связанных с Hadoop, но не входящих в Hadoop core:

- Hive – инструмент для SQL-like запросов над большими данными (превращает SQL-запросы в серию MapReduce–задач);
- Pig – язык программирования для анализа данных на высоком уровне. Одна строчка кода на этом языке может превратиться в последовательность MapReduce-задач;
- Hbase – колоночная база данных, реализующая парадигму BigTable;
- Cassandra – высокопроизводительная распределенная key-value база данных;
- ZooKeeper – сервис для распределённого хранения конфигурации и синхронизации изменений этой конфигурации;
- Mahout – библиотека и движок машинного обучения на больших данных.

Отдельно хотелось бы отметить проект Apache Spark, который представляет собой движок для распределённой обработки данных. Apache Spark обычно использует компоненты Hadoop, такие как HDFS и YARN для своей работы, при этом сам в последнее время стал популярнее, чем Hadoop:



Практическая часть

Установка Hadoop на кластер при помощи Cloudera Manager

Раньше установка Hadoop представляла собой достаточно тяжёлое занятие – нужно было по отдельности конфигурировать каждую машину в кластере, следить за тем, что ничего не забыто, аккуратно настраивать мониторинги. С ростом популярности Hadoop появились компании (такие как Cloudera, Hortonworks, MapR), которые предоставляют собственные сборки Hadoop и мощные средства для управления Hadoop-кластером. В нашем цикле материалов мы будем пользоваться сборкой Hadoop от компании Cloudera.

Для того чтобы установить Hadoop на свой кластер, нужно проделать несколько простых шагов:

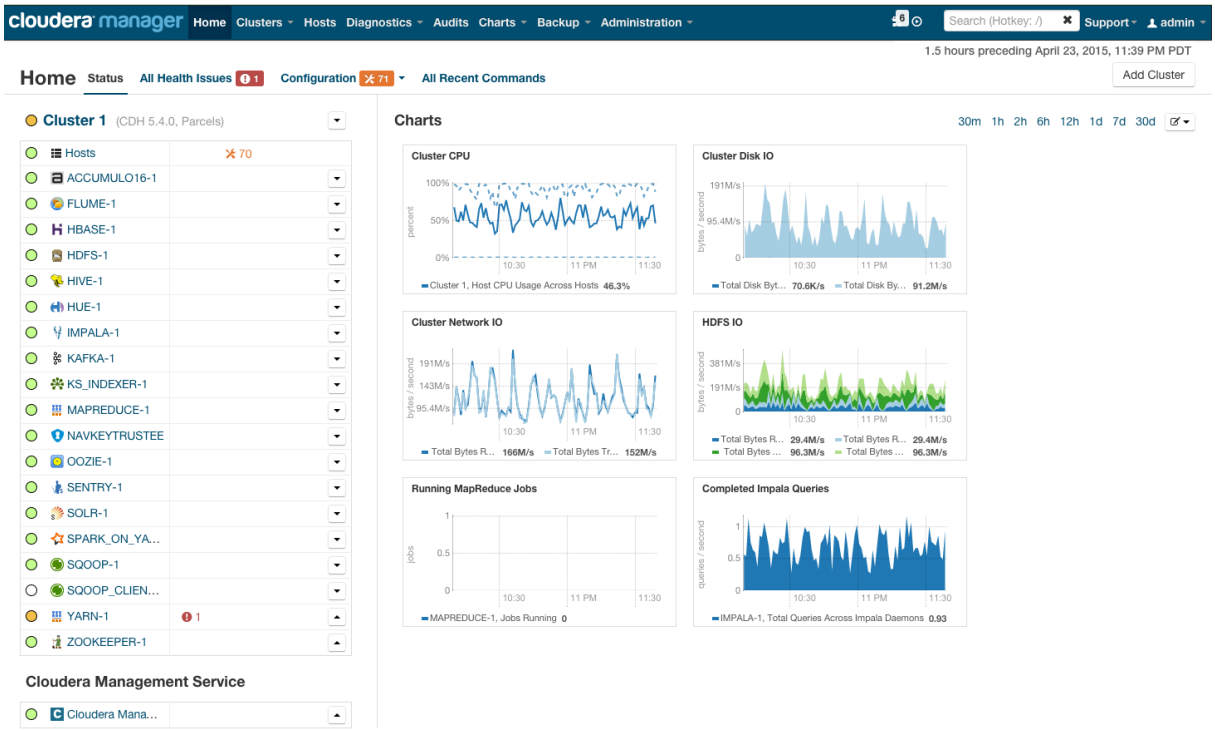
1 Скачать Cloudera Manager Express на одну из машин своего кластера отсюда;

2 Присвоить права на выполнение и запустить;

3 Следовать инструкциям установки.

Кластер должен работать на одной из поддерживаемых операционных систем семейства linux: RHEL, Oracle Enterprise linux, SLES, Debian, Ubuntu.

После установки вы получите консоль управления кластером, где можно смотреть установленные сервисы, добавлять/удалять сервисы, следить за состоянием кластера, редактировать конфигурацию кластера:



Более подробно с процессом установки Hadoop на кластер при помощи cloudera manager можно ознакомиться по ссылке в разделе Quick Start.

Если же Hadoop планируется использовать для «попробовать» – можно не заморачиваться с приобретением дорогого железа и настройкой Hadoop на нём, а просто скачать преднастроенную виртуальную машину по ссылке и пользоваться настроенным hadoop'ом.

Запуск MapReduce программ на Hadoop

Теперь покажем как запустить MapReduce-задачу на Hadoop. В качестве задачи воспользуемся классическим примером WordCount, который был разобран в предыдущей работе. Для того, чтобы экспериментировать на реальных данных, подготоим архив из случайных новостей с сайта lenta.ru. Скачать архив можно по ссылке.

Напомним формулировку задачи: имеется набор документов. Необходимо для каждого слова, встречающегося в наборе документов, посчитать, сколько раз встречается слово в наборе.

Решение:

Map разбивает документ на слова и возвращает множество пар (word, 1).

Reduce суммирует вхождения каждого слова:

```
def map(doc):
for word in doc.split():
    yield word, 1
def reduce(word, values):
    yield word, sum(values)
```

Теперь задача запрограммировать это решение в виде кода, который можно будет исполнить на Hadoop и запустить.

Способ №1. Hadoop Streaming

Самый простой способ запустить MapReduce-программу на Hadoop – воспользоваться streaming-интерфейсом Hadoop. Streaming-интерфейс предполагает, что map и reduce реализованы в виде программ, которые принимают данные с stdin и выдают результат на stdout.

Программа, которая исполняет функцию map называется mapper. Программа, которая выполняет reduce, называется, соответственно, reducer.

Streaming интерфейс предполагает по умолчанию, что одна входящая строка в mapper или reducer соответствует одной входящей записи для map.

Вывод mapper'а попадает на вход reducer'у в виде пар (ключ, значение), при этом все пары соответствующие одному ключу:

- Гарантированно будут обработаны одним запуском reducer'а;
- Будут поданы на вход подряд (то есть если один reducer обрабатывает несколько разных ключей – вход будет сгруппирован по ключу).

Итак, реализуем mapper и reducer на python:

```
#mapper.py
import sys

def do_map(doc):
for word in doc.split():
   yield word.lower(), 1

for line in sys.stdin:
   for key, value in do_map(line):
      print(key + "\t" + str(value))
#reducer.py
import sys

def do_reduce(word, values):
   return word, sum(values)

prev_key = None
values = []

for line in sys.stdin:
   key, value = line.split("\t")
   if key != prev_key and prev_key is not None:
      result_key, result_value = do_reduce(prev_key, values)
      print(result_key + "\t" + str(result_value))
      values = []
   prev_key = key
   values.append(int(value))

if prev_key is not None:
   result_key, result_value = do_reduce(prev_key, values)
   print(result_key + "\t" + str(result_value))
```

Данные, которые будет обрабатывать Hadoop должны храниться на HDFS. Загрузим наши статьи и положим на HDFS. Для этого нужно воспользоваться командой hadoop fs:

```
wget https://www.dropbox.com/s/opp5psid1x3jt41/lenta_articles.tar.gz
tar xzvf lenta_articles.tar.gz
hadoop fs -put lenta_articles
```

Утилита hadoop fs поддерживает большое количество методов для манипуляций с файловой системой, многие из которых один в один повторяют стандартные утилиты linux. Подробнее с её возможностями можно ознакомиться по ссылке.

Теперь запустим streaming-задачу:

```
yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar\
```

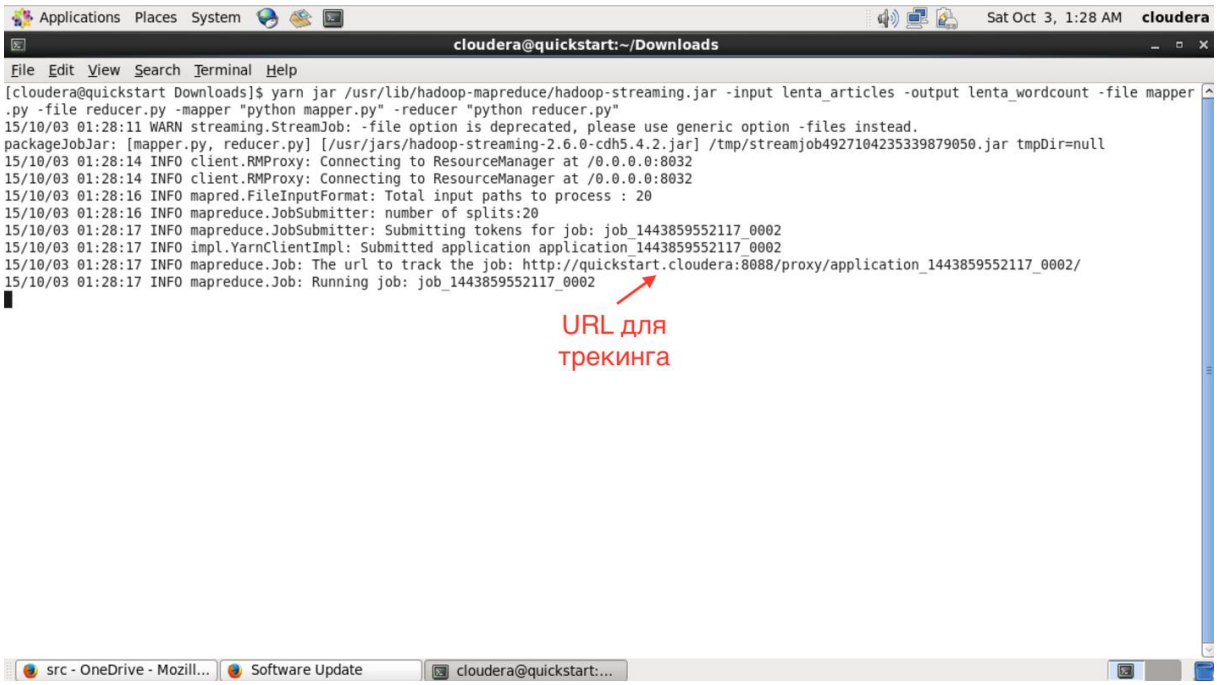
```
-input lenta_articles\
-output lenta_wordcount\
-file mapper.py\
-file reducer.py\
-mapper "python mapper.py"\
-reducer "python reducer.py"
```

Утилита yarn служит для запуска и управления различными приложениями (в том числе map-reduce based) на кластере. Hadoop-streaming.jar – это как раз один из примеров такого yarn-приложения.

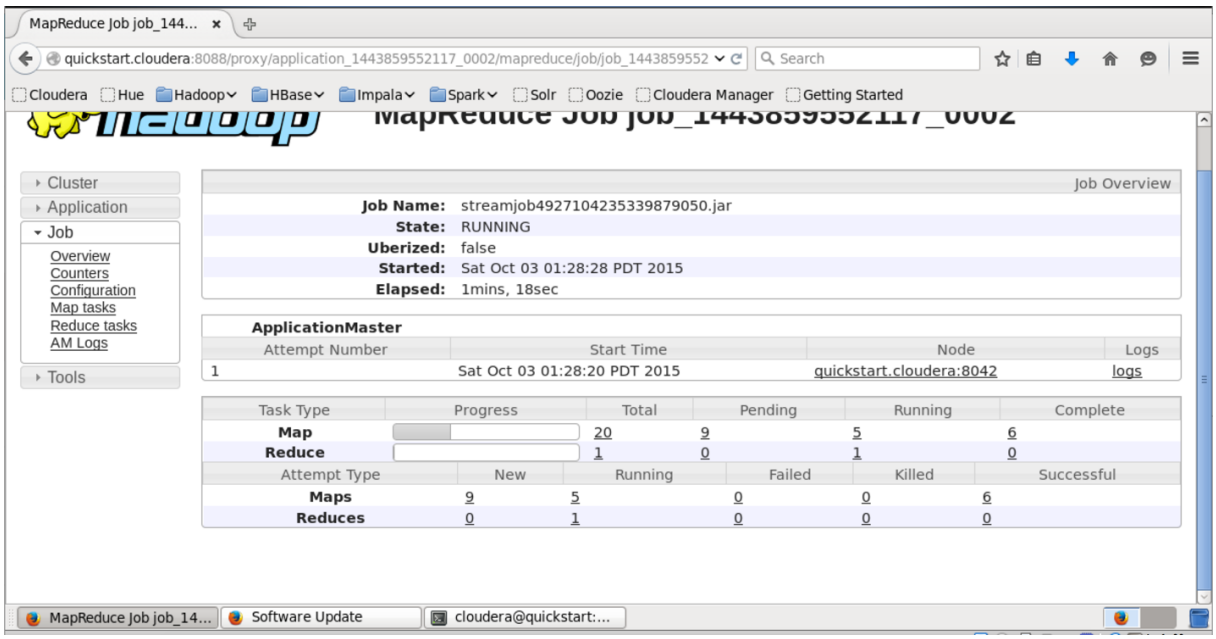
Дальше идут параметры запуска:

- input – папка с исходными данными на hdfs;
- output – папка на hdfs, куда нужно положить результат;
- file – файлы, которые нужны в процессе работы map-reduce задачи;
- mapper – консольная команда, которая будет использоваться для map-стадии;
- reduce – консольная команда которая будет использоваться для reduce-стадии.

После запуска в консоли можно будет увидеть прогресс выполнения задачи и URL для просмотра более детальной информации о задаче.



В интерфейсе доступном по этому URL можно узнать более детальный статус выполнения задачи, посмотреть логи каждого маппера и редьюсера (что очень полезно в случае упавших задач).



Результат работы после успешного выполнения складывается на HDFS в папку, которую мы указали в поле output. Просмотреть её содержание можно при помощи команды «hadoop fs -ls lenta_wordcount».

Сам результат можно получить следующим образом:

```
hadoop fs -text lenta_wordcount/* | sort -n -k2,2 | tail -n5
с	41
что	43
на	82
и	111
в	194
```

Команда «hadoop fs -text» выдаёт содержимое папки в текстовом виде. Отсортируем результат по количеству вхождений слов. Как и ожидалось, самые частые слова в языке – предлоги.

Способ №2

Сам по себе hadoop написан на java, и нативный интерфейс у hadoop-а тоже java-based. Покажем, как выглядит нативное java-приложение для wordcount:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
```

```
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
          StringTokenizer itr = new StringTokenizer(value.toString());
          while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
          }
        }
      }

      public static class IntSumReducer
          extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) throws IOException, InterruptedException {
          int sum = 0;
          for (IntWritable val : values) {
            sum += val.get();
          }
          result.set(sum);
          context.write(key, result);
        }
      }

      public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new
Path("hdfs://localhost/user/cloudera/lenta_articles"));
        FileOutputFormat.setOutputPath(job, new
Path("hdfs://localhost/user/cloudera/lenta_wordcount"));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
      }
    }
```

Этот класс делает абсолютно то же самое, что наш пример на Python. Мы создаём классы TokenizerMapper и IntSumReducer, наследуя их от классов Mapper и Reducer соответсвенно. Классы, передаваемые в качестве параметров шаблона, указывают типы входных и выходных значений. Нативный API подразумевает, что функции map на вход подаётся пара ключ-значение. Поскольку в нашем случае ключ пустой – в качестве типа ключа мы определяем просто Object.

В методе Main мы заводим mapreduce-задачу и определяем её параметры – имя, mapper и reducer, путь в HDFS, где находятся входные данные и куда положить результат.

Для компиляции нам потребуются hadoop-овские библиотеки. Я использую для сборки Maven, для которого у cloudera есть репозиторий. Инструкции по его настройке можно найти по ссылке. В итоге файл pom.xmp (который используется maven'ом для описания сборки проекта) у меня получился следующий):

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>2.6.0-cdh5.4.2</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-auth</artifactId>
      <version>2.6.0-cdh5.4.2</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-hdfs</artifactId>
      <version>2.6.0-cdh5.4.2</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-mapreduce-client-app</artifactId>
      <version>2.6.0-cdh5.4.2</version>
    </dependency>

  </dependencies>

  <groupId>org.dca.examples</groupId>
  <artifactId>wordcount</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Соберём проект в jar-пакет:

```
mvn clean package
```

После сборки проекта в jar-файл запуск происходит похожим образом, как и в случае streaming-интерфейса:

```
yarn jar wordcount-1.0-SNAPSHOT.jar  WordCount
```

Дожидаемся выполнения и проверяем результат:

```
hadoop fs -text lenta_wordcount/* | sort -n -k2,2 | tail -n5
с	41
что	43
на	82
и	111
в	194
```

Как нетрудно догадаться, результат выполнения нашего нативного приложения совпадает с результатом streaming-приложения, которое мы запустили предыдущим способом.

Резюме

Мы рассмотрели Hadoop – программный стек для работы с большими данными, описали процесс установки Hadoop на примере дистрибутива cloudera, показали, как писать mapreduce-программы, используя streaming-интерфейс и нативный API Hadoop'а.

Практическая работа № 4 Приемы и стратегии разработки MapReduce-приложений

Цель работы

Рассмотреть различные приёмы, которые позволяют эффективно использовать MapReduce для решения практических задач, а также изучить некоторые особенности Hadoop, которые позволяют упростить разработку или существенно ускорить выполнение MapReduce-задачи на кластере.

Задачи

- рассмотреть паттерны и приемы решения задач при помощи MapReduce,
- изучить, как объединять MapReduce-задачи в цепочки и join-ить логи по ключу.

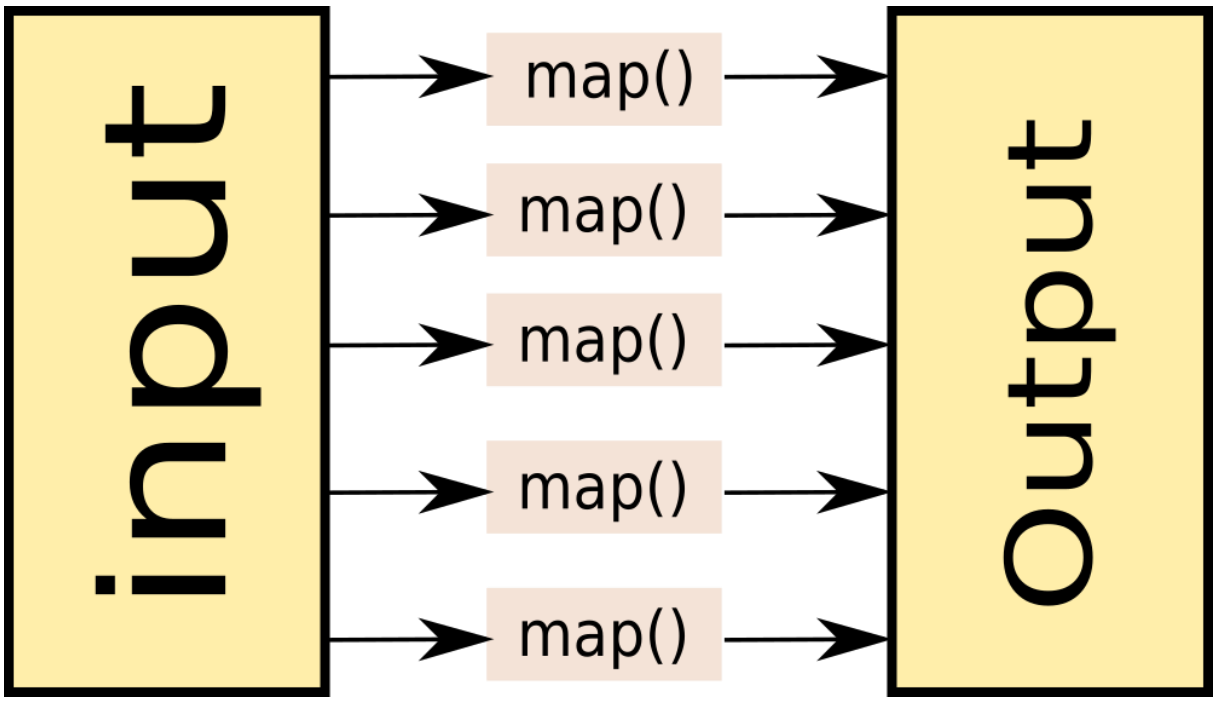
Ход Работы

Map only job

Как мы помним, MapReduce состоит из стадий Map, Shuffle и Reduce. Как правило, в практических задачах самой тяжёлой оказывается стадия Shuffle, так как на этой стадии происходит сортировка данных. На самом деле существует ряд задач, в которых можно обойтись только стадией Map. Вот примеры таких задач:

- Фильтрация данных (например, «Найти все записи с IP-адреса 123.123.123.123» в логах web-сервера);
- Преобразование данных («Удалить колонку в csv-логах»);
- Загрузка и выгрузка данных из внешнего источника («Вставить все записи из лога в базу данных»).

Такие задачи решаются при помощи Map-Only. При создании Map-Only задачи в Hadoop нужно указать нулевое количество reducer'ов:



Map Only Job

Пример конфигурации map-only задачи на hadoop:

Native interface

Hadoop Streaming Interface

Указать нулевое количество редьюсеров при конфигурации job’a:

```
job.setNumReduceTasks(0);
```

Более развёрнутый пример по ссылке.

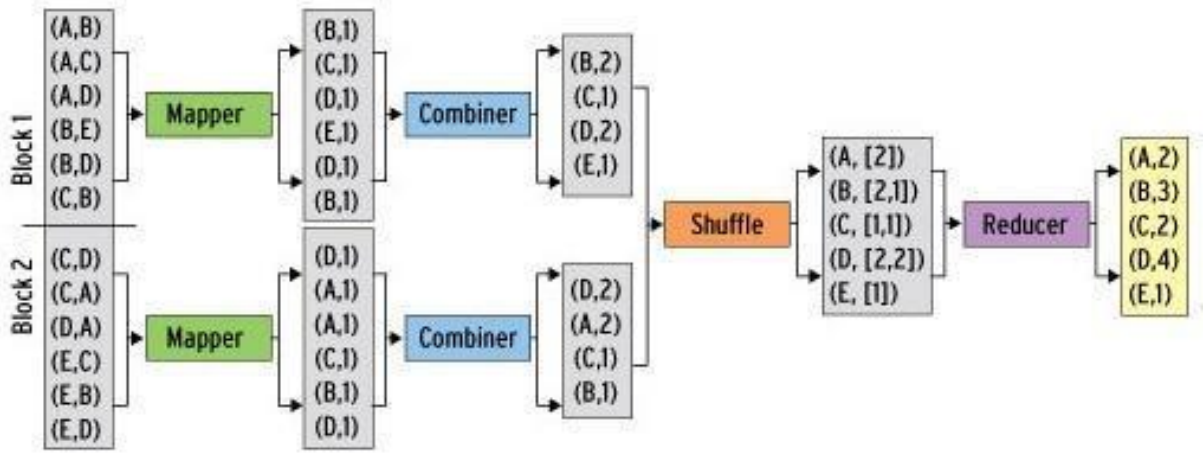
Не указываем редьюсер и указываем нулевое количество редьюсеров Пример:

```
hadoop jar hadoop-streaming.jar \
 -D mapred.reduce.tasks=0\
-input input_dir\
-output output_dir\
-mapper "python mapper.py"\
-file "mapper.py"
```

Map Only jobs на самом деле могут быть очень полезными. Например, в платформе Facetz.DCA для выявления характеристик пользователей по их поведению используется именно один большой map-only, каждый маппер которого принимает на вход пользователя и на выход отдаёт его характеристики.

Combine

Обычно самая тяжёлая стадия при выполнении Map-Reduce задачи – это стадия shuffle. Происходит это потому, что промежуточные результаты (выход mapper’a) записываются на диск, сортируются и передаются по сети. Однако существуют задачи, в которых такое поведение кажется не очень разумным. Например, в той же задаче подсчёта слов в документах можно предварительно предагрегировать результаты выходов нескольких mapper’ов на одном узле map-reduce задачи, и передавать на reducer уже просуммированные значения по каждой машине.



Combine.

В hadoop для этого можно определить комбинирующую функцию, которая будет обрабатывать выход части mapper-ов. Комбинирующая функция очень похожа на reduce – она принимает на вход выход части mapper'ов и выдаёт агрегированный результат для этих mapper'ов, поэтому очень часто reducer используют и как combiner. Важное отличие от reduce – на комбинирующую функцию попадают не все значения, соответствующие одному ключу.

Более того, hadoop не гарантирует того, что комбинирующая функция вообще будет выполнена для выхода mapper'а. Поэтому комбинирующая функция не всегда применима, например, в случае поиска медианного значения по ключу. Тем не менее, в тех задачах, где комбинирующая функция применима, её использование позволяет добиться существенного прироста к скорости выполнения MapReduce-задачи.

Использование Combiner'а на hadoop:

Native Interface

Hadoop streaming

При конфигурации job-а указать класс-Combiner. Как правило, он совпадает с Reducer:

```
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
```

В параметрах командной строки указать команду -combiner. Как правило, эта команда совпадает с командой reducer'а. Пример:

```
hadoop jar hadoop-streaming.jar \
-input input_dir\
-output output_dir\
-mapper "python mapper.py"\
-reducer "python reducer.py"\
-combiner "python reducer.py"\
-file "mapper.py"\
-file "reducer.py"\
```

Цепочки MapReduce-задач

Бывают ситуации, когда для решения задачи одним MapReduce не обойтись. Например, рассмотрим немного видоизмененную задачу WordCount: имеется набор текстовых документов, необходимо посчитать, сколько слов встретилось от 1 до 1000 раз в наборе, сколько слов от 1001 до 2000, сколько от 2001 до 3000 и так далее.

Для решения нам потребуется 2 MapReduce job'а:

1 Видоизменённый wordcount, который для каждого слова рассчитает, в какой из интервалов оно попало;

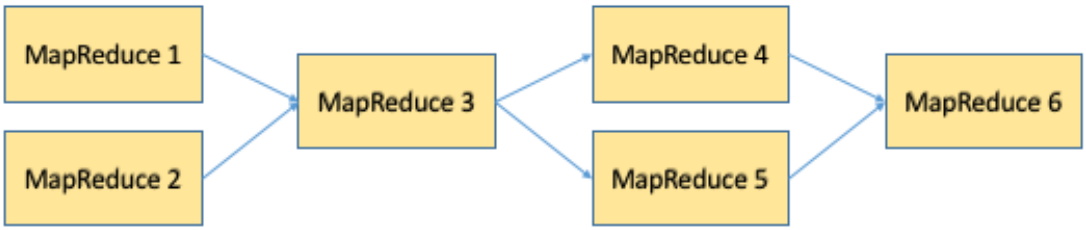
2 MapReduce, подсчитывающий, сколько раз в выходе первого MapReduce встретился каждый из интервалов.

Решение на псевдокоде:

```
#map1
def map(doc):
for word in doc:
        yield word, 1
#reduce1
def reduce(word, values):
yield int(sum(values)/1000), 1
#map2
def map(doc):
interval, cnt = doc.split()
yield interval, cnt
#reduce2
def reduce(interval, values):
yield interval*1000, sum(values)
```

Для того, чтобы выполнить последовательность MapReduce-задач на hadoop, достаточно просто в качестве входных данных для второй задачи указать папку, которая была указана в качестве output для первой и запустить их по очереди.

На практике цепочки MapReduce-задач могут представлять собой достаточно сложные последовательности, в которых MapReduce-задачи могут быть подключены как последовательно, так и параллельно друг другу. Для упрощения управления такими планами выполнения задач существуют отдельные инструменты типа oozie и luigi.



Пример цепочки MapReduce-задач.

Distributed cache

Важным механизмом в Hadoop является Distributed Cache. Distributed Cache позволяет добавлять файлы (например, текстовые файлы, архивы, jar-файлы) к окружению, в котором выполняется MapReduce-задача.

Можно добавлять файлы, хранящиеся на HDFS, локальные файлы (локальные для той машины, с которой выполняется запуск задачи). Уже было неявно показано, как использовать Distributed Cache вместе с hadoop streaming: добавляя через опцию -file файлы mapper.py и reducer.py. На самом деле можно добавлять не только mapper.py и reducer.py, а вообще произвольные файлы, и потом пользоваться ими как будто они находятся в локальной папке.

Использование Distributed Cache:

Native API

```
//конфигурация Job'а
JobConf job = new JobConf();
DistributedCache.addCacheFile(new URI("/myapp/lookup.dat#lookup.dat"),
                              job);
DistributedCache.addCacheArchive(new URI("/myapp/map.zip", job);
```

```
DistributedCache.addFileToClassPath(new Path("/myapp/mylib.jar"), job);
DistributedCache.addCacheArchive(new URI("/myapp/mytar.tar", job);
DistributedCache.addCacheArchive(new URI("/myapp/mytgz.tgz", job);
//пример использования в mapper-е:
public static class MapClass extends MapReduceBase
implements Mapper<K, V, K, V> {
 private Path[] localArchives;
 private Path[] localFiles;

 public void configure(JobConf job) {
  // получаем кэшированные данные из архивов
  File f = new File("./map.zip/some/file/in/zip.txt");
 }

 public void map(K key, V value,
            OutputCollector<K, V> output, Reporter reporter)
 throws IOException {
  // используем данные тут
  // ...
  // ...
  output.collect(k, v);
 }
}
```

Hadoop Streaming

#перечисляем файлы, которые необходимо добавить в distributed cache в параметре –files. Параметр –files должен идти перед другими параметрами.

```
yarn  hadoop-streaming.jar\
-files mapper.py,reducer.py,some_cached_data.txt\
-input '/some/input/path' \
-output '/some/output/path' \
-mapper 'python mapper.py' \
-reducer 'python reducer.py' \
```

пример использования:

```
import sys
#просто читаем файл из локальной папки
data = open('some_cached_data.txt').read()
for line in sys.stdin()
#processing input
#use data here
```

Reduce Join

Те, кто привык работать с реляционными базами, часто пользуются очень удобной операцией Join, позволяющей совместно обработать содержание некоторых таблиц, объединив их по некоторому ключу. При работе с большими данными такая задача тоже иногда возникает. Рассмотрим следующий пример:

Имеются логи двух web-серверов, каждый лог имеет следующий вид:

\t\t.

Пример кусочка лога:

```
1446792139    178.78.82.1     /sphingosine/unhurrying.css
1446792139    126.31.163.222          /accentually.js
1446792139    154.164.149.83          /pyroacid/unkemptly.jpg
1446792139    202.27.13.181 /Chawia.js
1446792139    67.123.248.174          /morphographical/dismain.css
```

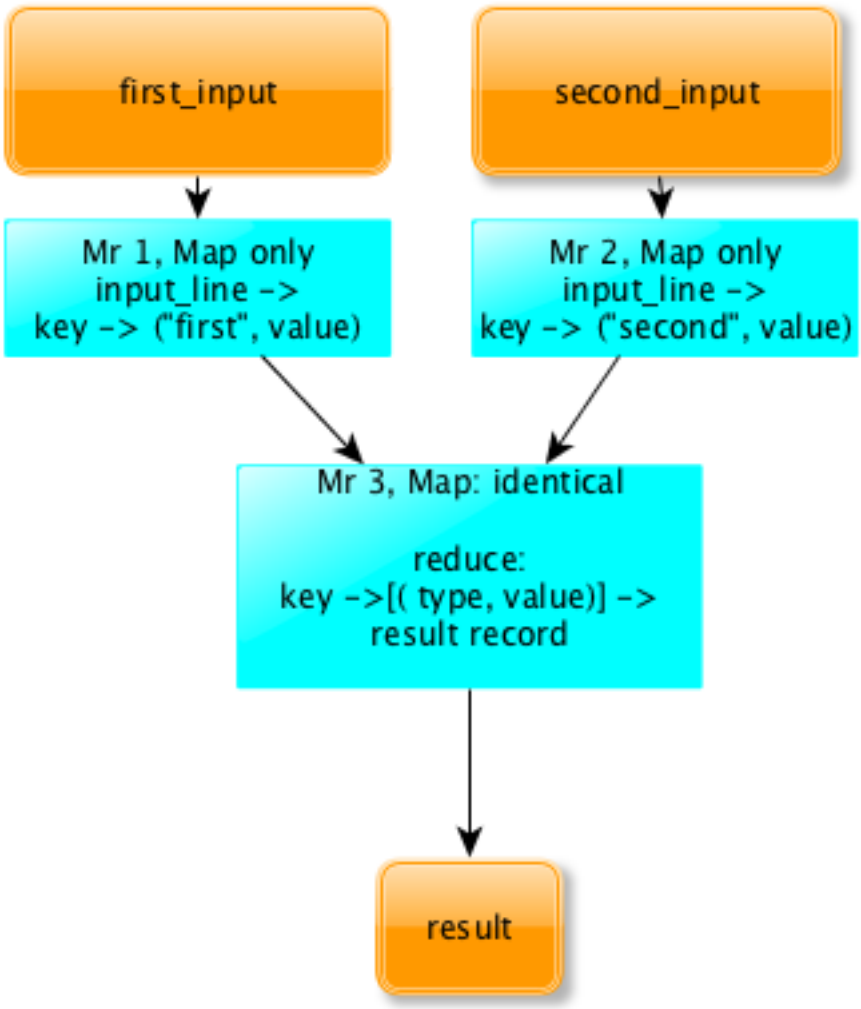
1446792139 226.74.123.135 /phanerite.php
1446792139 157.109.106.104 /bisonant.css

Необходимо посчитать для каждого IP-адреса на какой из 2-х серверов он чаще заходил. Результат должен быть представлен в виде:

\t. Пример части результата:

178.78.82.1 first
126.31.163.222 second
154.164.149.83 second
226.74.123.135 first

К сожалению, в отличие от реляционных баз данных, в общем случае объединение двух логов по ключу (в данном случае – по IP-адресу) представляет собой достаточно тяжёлую операцию и решается при помощи 3-х MapReduce и паттерна Reduce Join:



Общая схема ReduceJoin

ReduceJoin работает следующим образом:

На каждый из входных логов запускается отдельный MapReduce (Map only), преобразующий входные данные к следующему виду:

key -> (type, value)

Где *key* – это ключ, по которому нужно объединять таблицы, *Type* – тип таблицы (first или second в нашем случае), а *Value* – это любые дополнительные данные, привязанные к ключу.

Выходы обоих MapReduce подаются на вход 3-му MapReduce, который, собственно, и выполняет объединение. Этот MapReduce содержит пустой Mapper, который про-

сто копирует входные данные. Дальше shuffle раскладывает данные по ключам и подаёт на вход редьюсеру в виде:

key -> [(type, value)]

Важно, что в этот момент на редьюсер попадают записи из обоих логов и при этом по полю type можно идентифицировать, из какого из двух логов попало конкретное значение. Значит данных достаточно, чтобы решить исходную задачу. В нашем случае reducere просто должен посчитать для каждого ключа записей, с каким type встретилось больше и вывести этот type.

MapJoin

Паттерн ReduceJoin описывает общий случай объединения двух логов по ключу. Однако есть частный случай, при котором задачу можно существенно упростить и ускорить. Это случай, при котором один из логов имеет размер существенно меньшего размера, чем другой. Рассмотрим следующую задачу:

Имеются 2 лога. Первый лог содержит лог web-сервера (такой же как в предыдущей задаче), второй файл (размером в 100кб) содержит соответствие URL-> Тематика. Пример 2-го файла:

/toyota.php auto
/football/spartak.html sport
/cars auto
/finances/money business

Для каждого IP-адреса необходимо рассчитать страницы какой категории с данного IP-адреса загружались чаще всего.

В этом случае нам тоже необходимо выполнить Join 2-х логов по URL. Однако в этом случае нам не обязательно запускать 3 MapReduce, так как второй лог полностью влезет в память. Для того, чтобы решить задачу при помощи 1-го MapReduce, мы можем загрузить второй лог в Distributed Cache, а при инициализации Mapper'а просто считать его в память, положив его в словарь -> topic.

Далее задача решается следующим образом:

Map:

находим тематику каждой из страниц первого лога

input_line -> [ip, topic]

Reduce:

Ip -> [topics] -> [ip, most_popular_topic]

Reduce получает на вход ip и список всех тематик, просто вычисляет, какая из тематик встретилась чаще всего. Таким образом задача решена при помощи 1-го MapReduce, а собственно Join вообще происходит внутри map (поэтому если бы не нужна была дополнительная агрегация по ключу – можно было бы обойтись MapOnly job-ом):

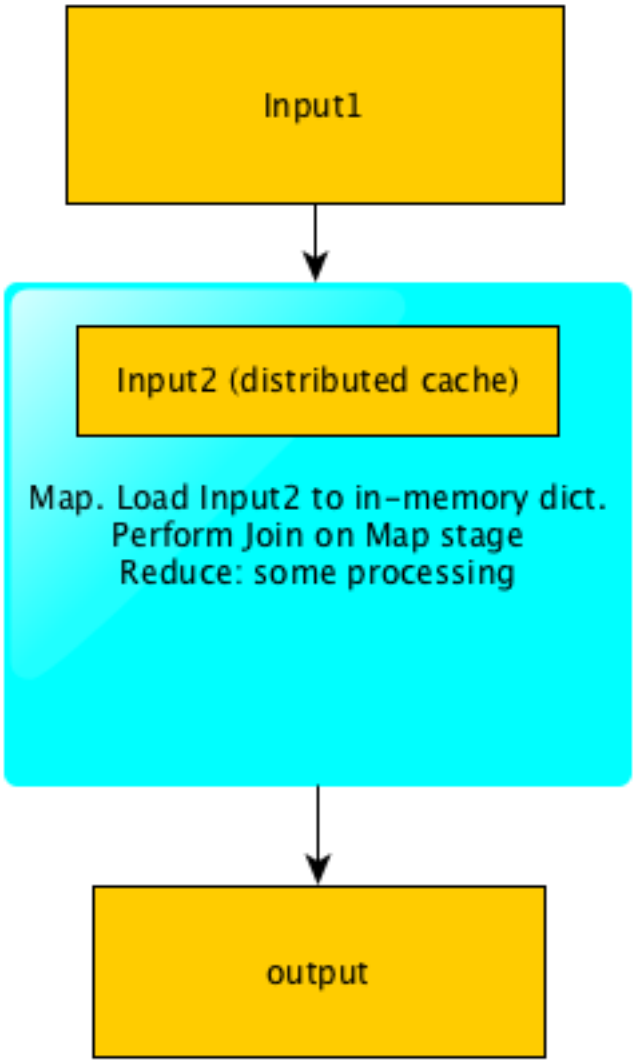


Схема работы MapJoin

Резюме

Мы рассмотрели несколько паттернов и приемов решения задач при помощи MapReduce, показали, как объединять MapReduce-задачи в цепочки и join-ить логи по ключу.

Практическая работа № 5 Hbase

Цель работы

Изучить инструмент Hbase и использовать его в качестве основного хранилища сырых данных.

Задачи

- Изучить модель данных Hbase;
- Рассмотреть поддерживаемые операции и архитектуру
- Изучить концепцию BigTable.

Ход Работы

Кто и зачем придумал Hbase ?

Как и многие другие проекты из области Big Data, Hbase зародилась из концепции, которая была разработана в компании Google.

Обычные файлы довольно неплохо подходят для пакетной обработки данных, с использованием парадигмы MapReduce.

С другой стороны информацию хранящуюся в файлах довольно неудобно обновлять; Файлы также лишены возможности произвольного доступа. Для быстрой и удобной работы с произвольным доступом есть класс nosql-систем типа key-value storage, таких как Aerospike, Redis, Couchbase, Memcached. Однако в обычно в этих системах очень неудобна пакетная обработка данных. Hbase представляет из себя попытку объединения удобства пакетной обработки и удобства обновления и произвольного доступа.

Модель данных

Hbase — это распределенная, колоночно-ориентированная, мультиверсионная база типа «ключ-значение».

Данные организованы в таблицы, проиндексированные первичным ключом, который в Hbase называется RowKey.

Для каждого RowKey ключа может храниться неограниченны набор атрибутов (или колонок).

Колонки организованны в группы колонок, называемые Column Family. Как правило в одну Column Family объединяют колонки, для которых одинаковы паттерн использования и хранения.

Для каждого аттрибута может храниться несколько различных версий. Разные версии имеют разный timestamp.

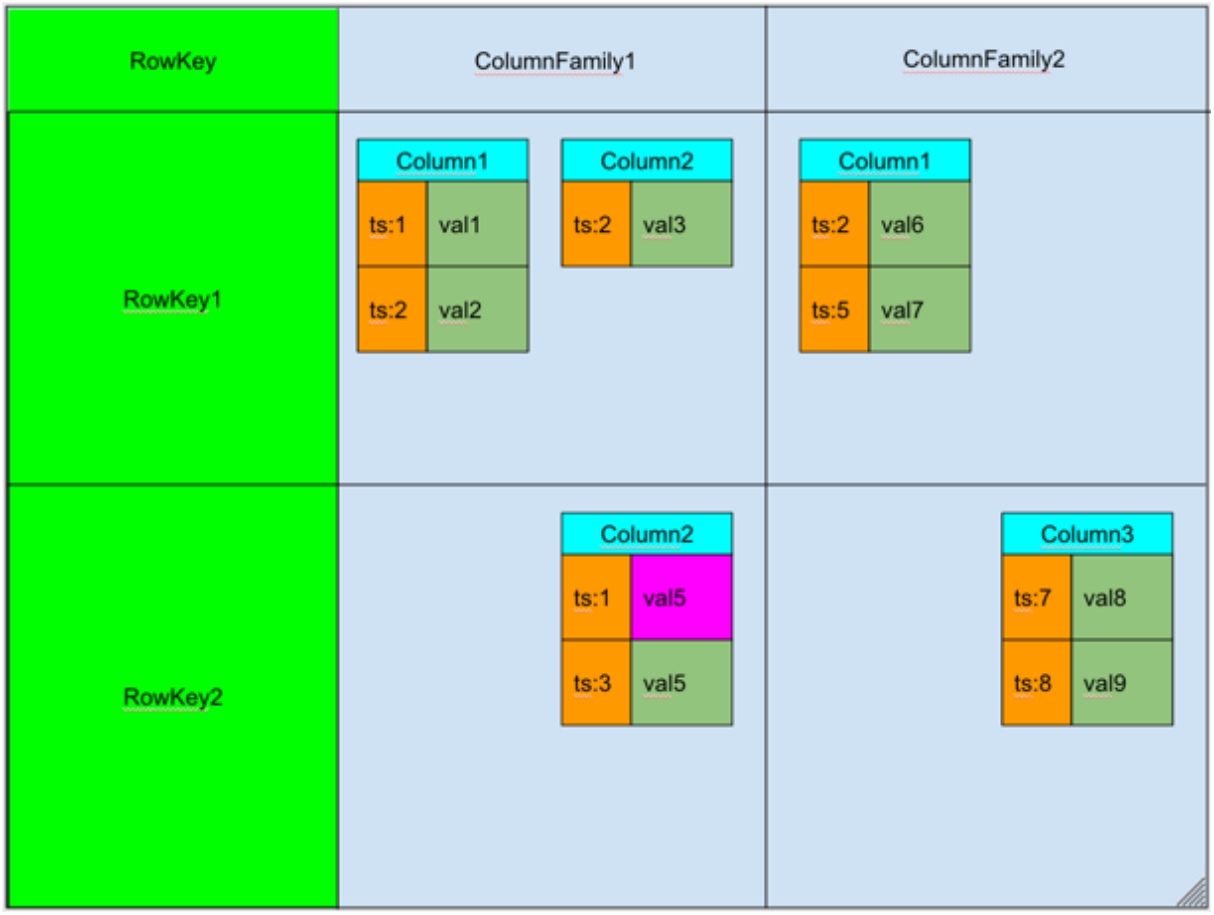
Записи физически хранятся в отсортированном по RowKey порядке. При этом данные соответствующие разным Column Family хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок.

При удалении определённого атрибута физически он сразу не удаляется, а лишь маркируется специальным флажком tombstone. Физическое удаление данных произойдет позже, при выполнении операции Major Compaction.

Атрибуты, принадлежащие одной группе колонок и соответствующие одному ключу физически хранятся как отсортированный список. Любой атрибут может отсут-

ствовать или присутствовать для каждого ключа, при этом если атрибут отсутствует — это не вызывает накладных расходов на хранение пустых значений.

Список и названия групп колонок фиксирован и имеет четкую схему. На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых версий. Если разница между timestamp для определенно версии и текущим временем больше TTL — запись помечается к удалению. Если количество версий для определённого атрибута превысило максимальное количество версий — запись также помечается к удалению.



Модель данных Hbase можно запомнить как соответствие ключ значение:
<table, RowKey, Column Family, Column, timestamp> -> Value

Поддерживаемые операции

Список поддерживаемых операций в hbase весьма прост. Поддерживаются 4 основные операции:

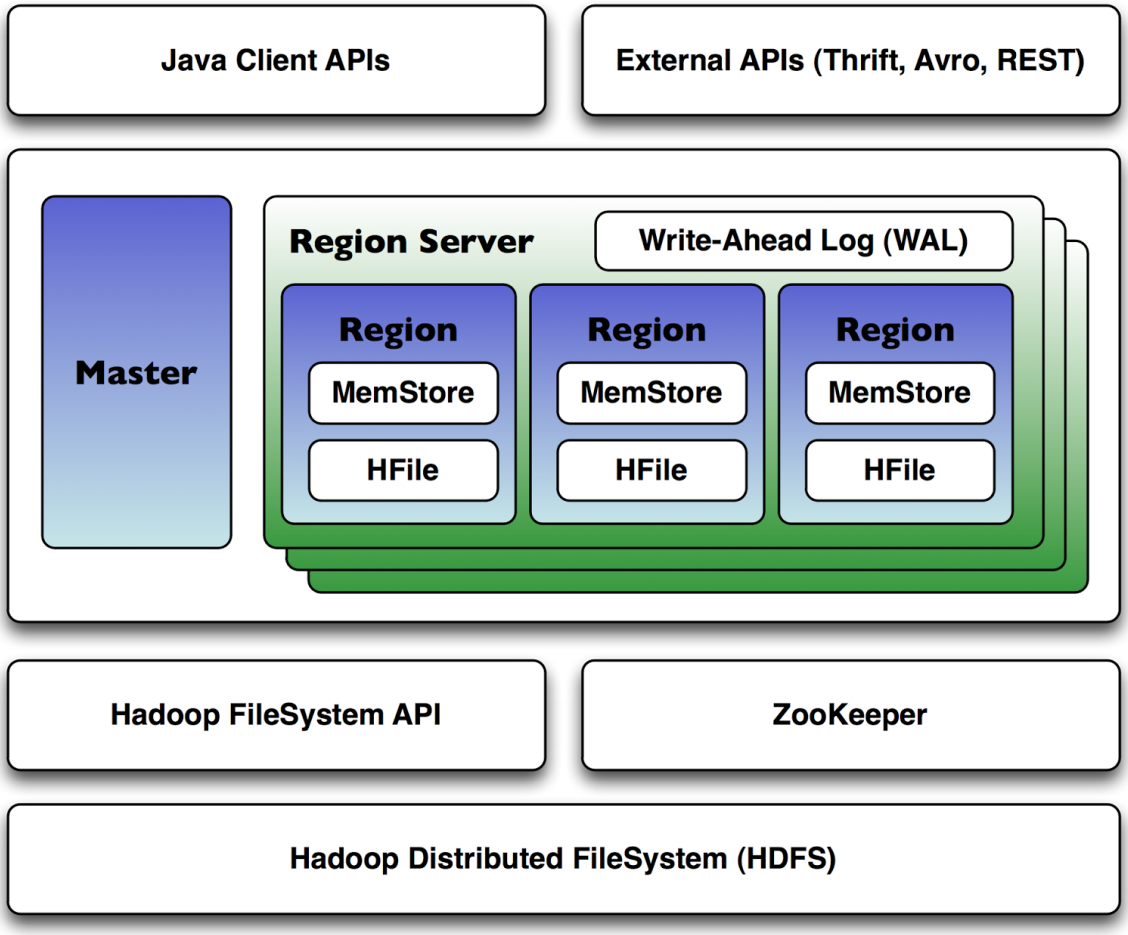
— Put: добавить новую запись в hbase. Timestamp этой записи может быть задан руками, в противном случае он будет установлен автоматически как текущее время.

— Get: получить данные по определенному RowKey. Можно указать Column Family, из которой будем брать данные и количество версий которые хотим прочитать.

— Scan: читать записи по очереди. Можно указать запись с которой начинаем читать, запись до которой читать, количество записей которые необходимо считать, Column Family из которой будет производиться чтение и максимальное количество версий для каждой записи.

— Delete: пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction (см. ниже).

Архитектура



Hbase является распределенной базой данных, которая может работать на десятках и сотнях физических серверов, обеспечивая бесперебойную работу даже при выходе из строя некоторых из них. Поэтому архитектура hbase довольна сложна по сравнению с классическими реляционными базами данных.

Hbase для своей работы использует два основных процесса:

1. Region Server — обслуживает один или несколько регионов. Регион — это диапазон записей соответствующих определенному диапазону подряд идущих RowKey. Каждый регион содержит:

- Persistent Storage — основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате HFile. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle.
-
- MemStore — буфер на запись. Так как данные хранятся в HFile d отсортированном порядке — обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении MemStore до некоторого критического значения данные записываются в новый HFile.
-
- BlockCache — кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.
-
- Write Ahead Log(WAL). Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не про-

изошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.

2. Master Server — главный сервер в кластере hbase. Master управляет распределением регионов по Region Server'ам, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.

Для координации действий между сервисами Hbase использует Apache ZooKeeper, специальный сервис предназначенный для управления конфигурациями и синхронизацией сервисов.

При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.

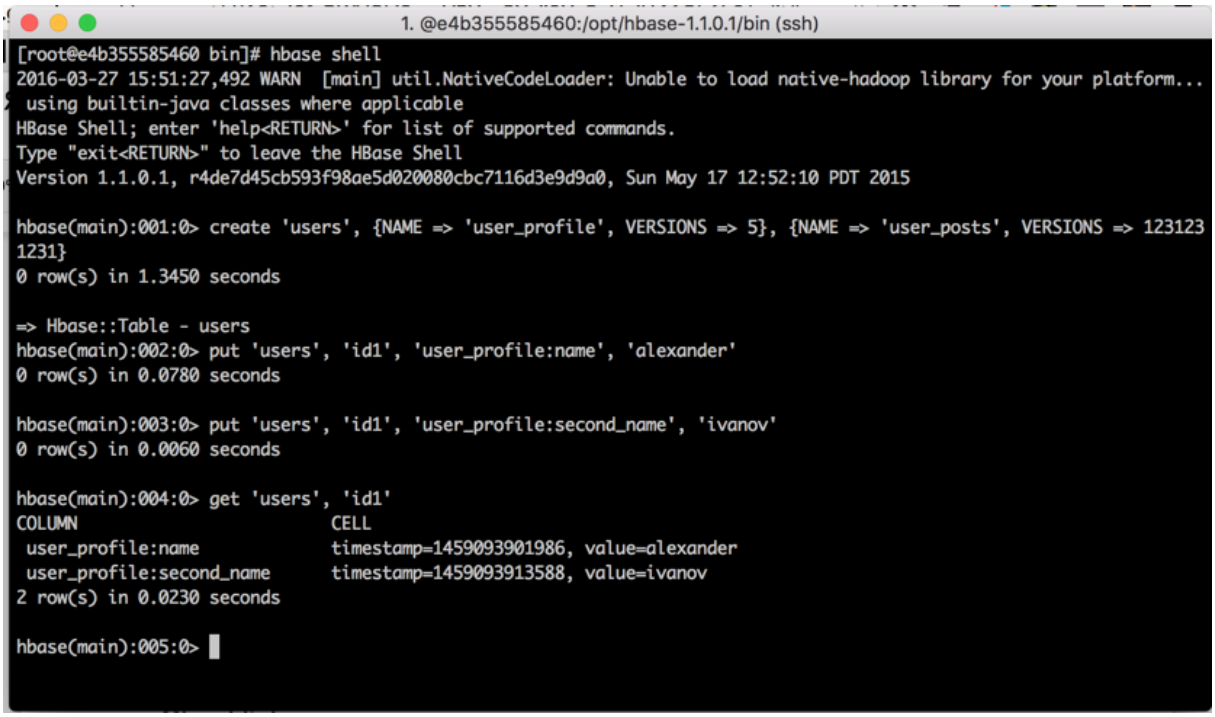
Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется compaction. Compaction'ы бывают двух видов:

- Minor Compaction. Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.
-
- Major Compaction. Запускается руками или по наступлению срабатыванию определенных триггеров(например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера. Major Compaction'ы лучше делать во время когда нагрузка на кластер небольшая. Во время Major Compaction также происходит физическое удаление данных, ране помеченных меткой tombstone.

Способы работы с Hbase

Hbase Shell

Самый простой способ начать работу с Hbase — воспользоваться утилитой hbase shell. Она доступна сразу после установки hbase на любой ноде кластера hbase.



Hbase shell представляет из себя jruby-консоль с встроенной поддержкой всех основных операций по работе с Hbase. Ниже приведён пример создания таблицы users с двумя column family, выполнения некоторых манипуляций с ней и удаление таблицы в конце на языке hbase shell:

Простыня кода

Native Api

Как и большинство других hadoop-related проектов hbase реализован на языке java, поэтому и нативный api доступен для языке java. Native API довольно неплохо задокументирован на официальном сайте. Вот пример использования Hbase API взятый оттуда же:

Простыня кода

Thrift, REST и поддержка других языков программирования.

Для работы из других языков программирования Hbase предоставляет Thrift API и Rest API. На базе них построены клиенты для всех основных языков программирования: python, PHP, Java Script и тд.

Некоторые особенности работы с HBase

1. Hbase «из коробки» интегрируется с MapReduce, и может быть использована в качестве входных и выходных данных с помощью специальных TableInputFormat и TableOutputFormat.

2. Очень важно правильно выбрать RowKey. RowKey должен обеспечивать хорошее равномерное распределение по регионам, в противном случае есть риск возникновения так называемых «горячих регионов» — регионов которые используются гораздо чаще остальных, что приводит к неэффективному использованию ресурсов системы.

3. Если данные заливаются не единично, а сразу большими пачками — Hbase поддерживает специальный механизм BulkLoad, который позволяет заливать данные намного быстрее чем используя единичные Put'ы. BulkLoad по сути представляет из себя двухшаговую операцию:

— Формирование HFile без участия put'ов при помощи специального MapReduce job'a

— Подкладывание этих файликов напрямую в Hbase.

4. Hbase поддерживает вывод своих метрик в сервер мониторинга Ganglia. Это может быть очень полезно при администрировании Hbase для понимания сути происходящих с hbase проблем.

Пример

В качестве примера можем рассмотреть основную таблицу с данными, которая у нас в Data-Centric Aliance используется для хранения информации о поведении пользователей в интернете.

RowKey

В качестве RowKey используется идентификатор пользователя, в качестве которого используется GUUID, строчка специально генерируемая таким образом, чтобы быть уникальной во всем мире. GUUID'ы распределены равномерно, что дает хорошее распределение данных по серверам.

Column Family

В нашем хранилище используются две column family:

— Data. В этой группе колонок хранятся данные, которые теряют свою актуальность для рекламных целей, такие как факты посещения пользователем определенных URL. TTL на эту Column Family установлен в размере 2 месяца, ограничение по количеству версий — 2000.

— LongData. В этой группе колонок хранятся данные, которые не теряют свою актуальность в течение долгого времени, такие как пол, дата рождения и другие «вечные» характеристики пользователя

Колонки

Каждый тип фактов о пользователе хранится в отдельной колонке. Например в колонке Data:_v хранятся URL, посещенные пользователем, а в колонке LongData:gender — пол пользователя.

В качестве timestamp хранится время регистрации этого факта. Например в колонке Data:_v — в качестве timestamp используется время захода пользователем на определенный URL.

Такая структура хранения пользовательских данных очень хорошо ложится на наш паттерн использования и позволяет быстро обновлять данные о пользователях, быстро доставать всю необходимую информацию о пользователях, и, используя MapReduce, быстро обрабатывать данные о всех пользователях сразу.

Альтернативы

Hbase довольно сложна в администрировании и использовании, поэтому прежде чем использовать hbase есть смысл обратить внимание на альтернативы:

- Реляционные базы данных. Очень неплохая альтернатива, особенно в случае когда данные влезают на одну машину. Также в первую очередь о реляционных базах данных стоит подумать в случае когда важны транзакции индексы отличные от первичного.
- Key-Value хранилища. Такие хранилища как Redis и Aerospike лучше подходят когда необходима минимизация latency и менее важна пакетная обработка данных.
- Файлы и их обработка при помощи MapReduce. Если данные только добавляются, и редко обновляются/изменяются, то лучше не использовать Hbase, а просто хранить данные в файлах. Для упрощения работы с файлами можно воспользоваться такими инструментами как Hive, Pig и Impala.

Использование Hbase оправдано когда:

— Данных много и они не влезают на один компьютер

— Данные часто обновляются и удаляются

— В данных присутствует явный «ключ» по к которому удобно привязывать все остальное

— Нужна пакетная обработка данных

— Нужен произвольный доступ к данным по определенным ключам

Заключение

В данной работе рассмотрели Hbase — мощное средство для хранения и обновления данных в экосистеме hadoop, показали модель данных Hbase, её архитектуру и особенности работы с ней.

Приложение 1. Аннотация
рабочей программы дисциплины

АННОТАЦИЯ РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ

Цель дисциплины: изучение методов обработки структурированных и неструктурированных многообразных данных огромных объёмов для получения воспринимаемых человеком результатов.

Задачи:

- изучение основных принципов и методов хранения и управления данными формата Big Data;
- знакомство с методами организации и анализа данных формата Big Data.
-

В результате освоения дисциплины обучающийся должен:

Знать основные принципы и методы хранения, управления, обработки, анализа данных формата Big Data, методологию разработки информационного обеспечения, проектирования, создания и поддержки хранилищ данных, способы организации больших данных, организацию их инфраструктуры, основные методы и средства управления информационной безопасностью при работе с большими данными.

Уметь строить модели для данных, хранящихся в распределенной файловой системе (Hadoop), организовывать ИТ-инфраструктуру предприятия при работе с большими данными, выбирать методы и разрабатывать средства защиты информации при работе с большими данными.

Владеть методами прогнозного моделирования и анализа данных (алгоритм Map-reduce), навыками участия в организации ИТ- инфраструктуры и управления информационной безопасности при работе с большими данными.