# Обзор инструментов обработки Big Data

**Максим Губин**

Томск

econophysica

# Жизненный цикл данных


1. Собрать данные;
2. Преобразовать данные в формат, подходящий для их хранения;
3. Произвести очистку данных;
4. Проанализировать данные;
5. Использовать результаты анализа;
6. Хранить данные;
7. Уничтожить данные;
8. Вновь начать с шага 1.

# Google Big Table

- Постоянная задержка менее 10 мс.
- Репликация обеспечивает более высокую доступность, долговечность и отказоустойчивость перед лицом зональных сбоев.
- Идеально подходит для рекламных технологий, финансовых приложений и интернета вещей.
- Механизм хранения для приложений машинного обучения
- Простая интеграция с инструментами больших данных с открытым исходным кодом

# Hadoop


- Платформа программного обеспечения с открытым исходным кодом, поддерживающая распределенные приложения с интенсивным использованием данных, лицензированная по лицензии Apache v2
- Высокая масштабируемость и доступность; Может использовать обычное (дешевое!) оборудование с небольшим резервированием;
- Отказоустойчивость;
- Перемещает **вычисления**, а не данные;

# ElasticSearch

*Гибкий и мощный открытый распределенный поисковый и аналитический движок в реальном времени для облака.*

**Возможности:**

Работа с данными в реальном времени, аналитика в реальном времени,

распределенность, высокая доступность, многопользовательский режим,

полнотекстовый поиск, ориентированный на документы, управление конфликтами, схемонезависимый, REST API, персистентность для каждой операции, лицензия с открытым исходным кодом apache 2, сборка поверх Apache Lucene.

**Недостатки:**

Жертвует способностью обрабатывать данные, чтобы получить максимальную производительность для группировки и фильтрации задач.

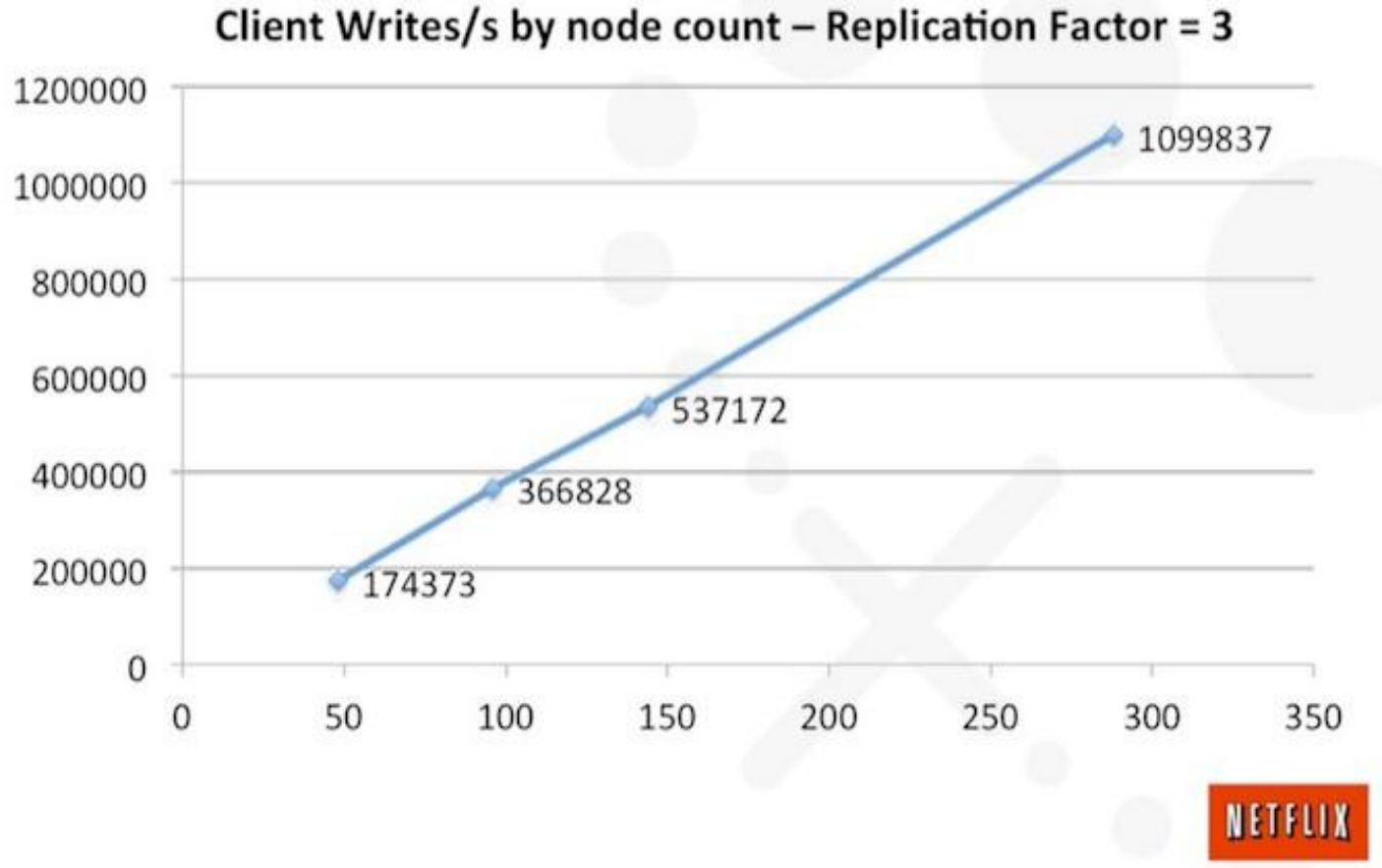
# Cassandra


## Особенности:

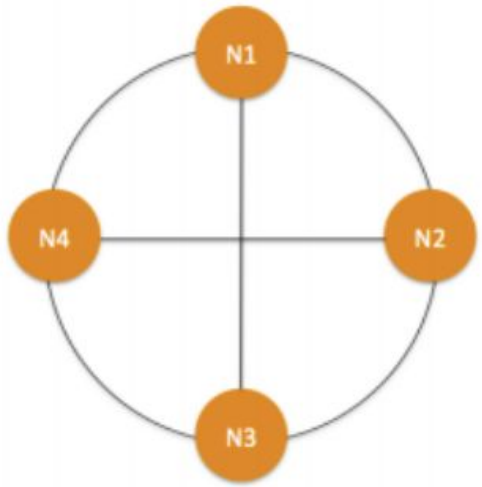
- O (1) поиск узла;
- Хранилище с подходом Ключ – Значение;
- Хранилище данных на основе столбцов;
- Высоко распределенная и децентрализованная (нет главного узла);
- Эластичность;
- Отказоустойчивая - репликации;
- Разреженное хранение;
- Каждый столбец имеет значение и временную метку, актуальными считаются самые свежие данные.

# Cassandra

## Масштабируемость, тест Netflix



# Cassandra


## Архитектура:

- Дробление:

  Как данные распределяются по узлам;

- Репликация:

  Как данные дублируются на узлах;

- Членство в кластере

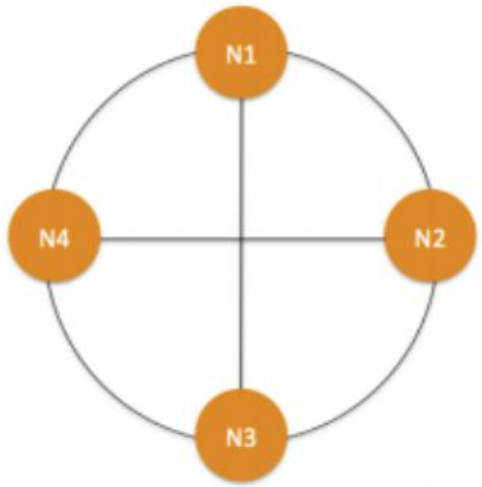
  Как узлы добавляются и удаляются из кластера;





econophysica

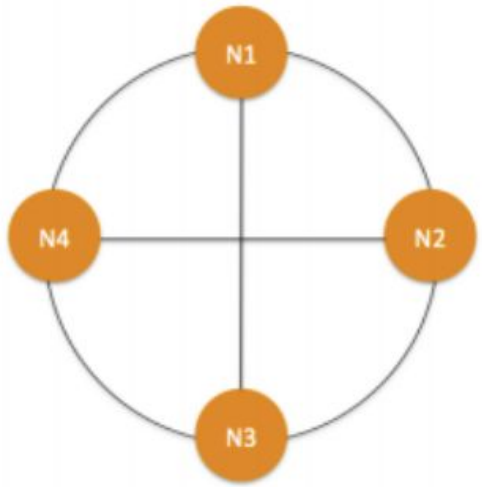
# Cassandra


## Дробление:

- Узлы логически структурированы в кольцевой топологии.
- Хешированное значение ключа, связанного с разделом данных, используется для назначения его узлу в кольце.
- Хеширование округляется после определенного значения для поддержки структуры кольца.
- Слабо загруженные узлы перемещаются, чтобы облегчить нагрузку на сильно загруженные узлы.

# Cassandra


## Репликация:

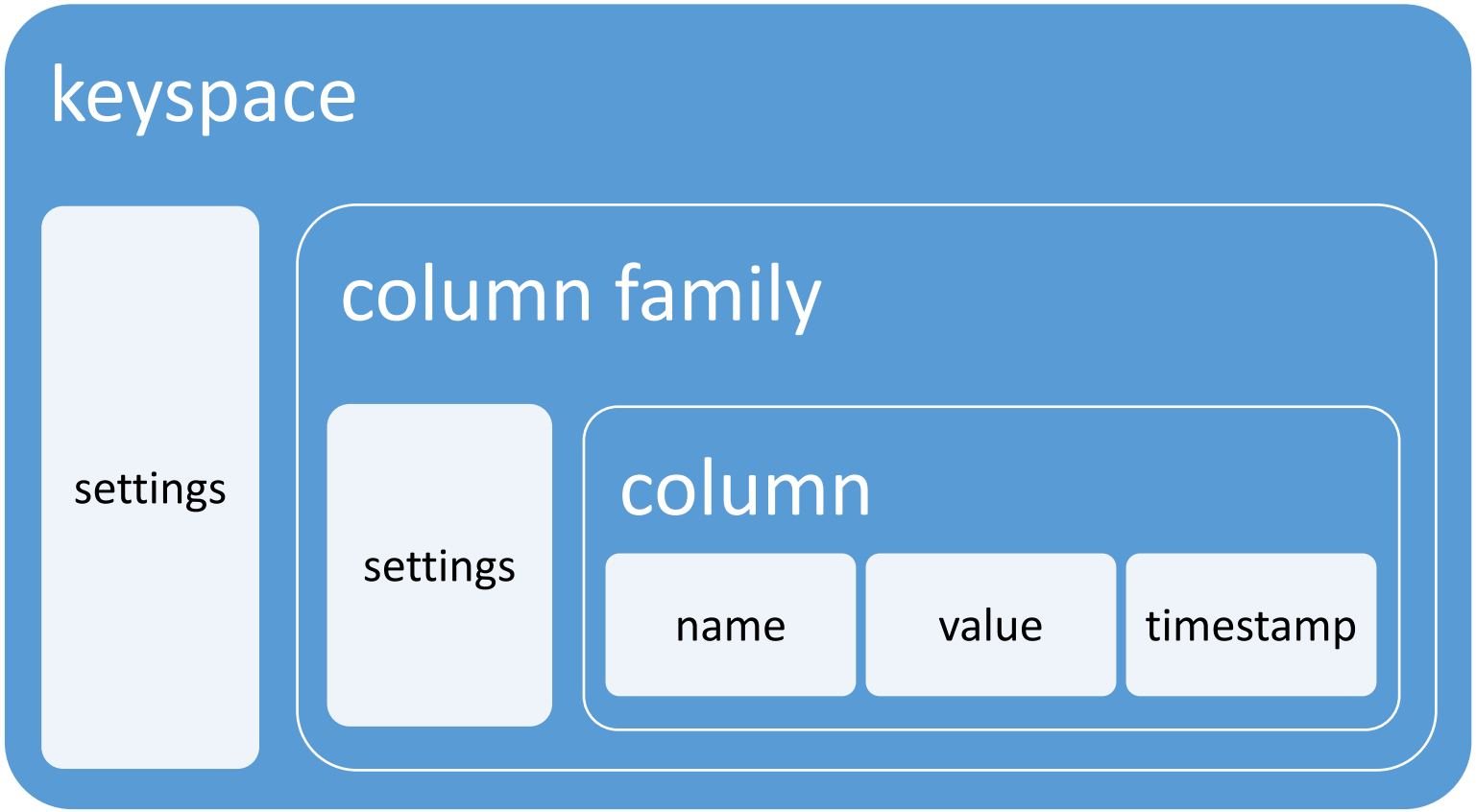
Каждый элемент данных реплицируется в N (фактор репликации) узлах.

**Различные политики репликации**

- **Rack Unaware** - реплицируйте данные на N-1 последовательных узлах после своего координатора
- **Rack Aware** - использует «Zookeeper» для выбора лидера, который сообщает узлам диапазон, для которого они являются репликами
- **Datacenter Aware** - аналогично Rack Aware, но лидер выбирается на уровне Datacenter, а не на уровне Rack.

# Cassandra


## Модель данных








econophysica

# Cassandra


PACELC

в случае разделения сети (P) в распределённой компьютерной системе необходимо выбирать между доступностью (A) и согласованностью (C) (согласно теореме CAP), но в любом случае, даже если система работает нормально в отсутствии разделения, нужно выбирать между задержками (L) и согласованностью (C).

Cassandra позволяет настраивать уровень согласованности чтения и записи.





econophysica

# Cassandra: использование






econophysica

# Cassandra


## Подойдет ли Cassandra для моей задачи?

- Вам требуется очень высокая скорость записи;
- Вам нужно хранить данные долгое время;
- У вас много данных:
  - > ГБ
  - > = трех серверов
- ваше приложение развивается
  - режим стартапа, структура данных меняется
  - свободные данные предметной области, "точки интереса"

**Ваши программисты справятся с:**

- документацией;
- сложностью;
- согласованностью модели;
- изменениями;
- инструментами видимости данных;

**Ваших ресурсов хватит на:**

- аппаратные требования;
- необходимость перемещать данные;
- JMX-мониторинг.





econophysica

# MongoDB


Описание:

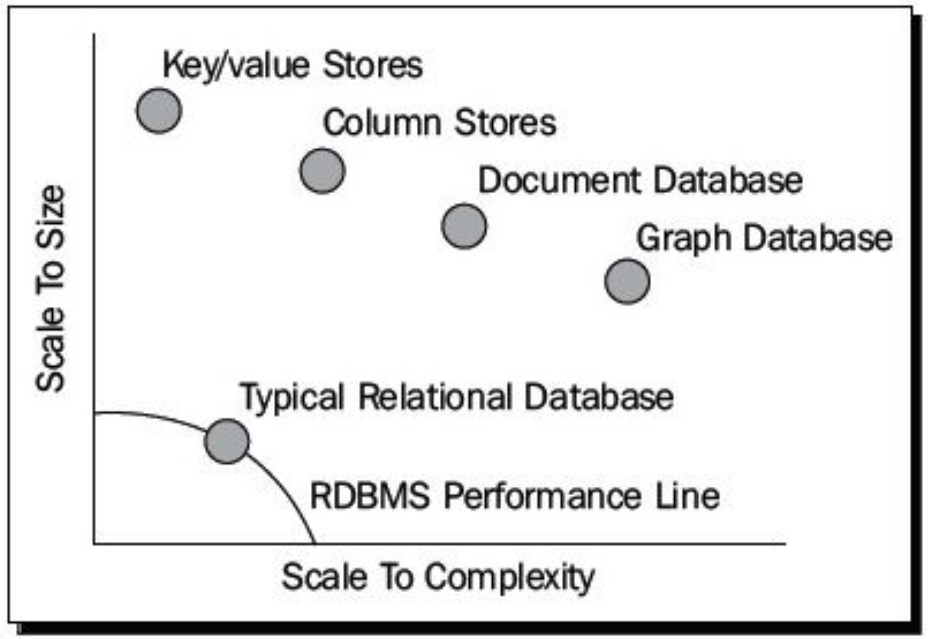
- Open-source;
- Основана на документах – объектах в формате BSON (Binary JSON);
- “High performance, high availability”;
- Автоматическое масштабирование;
- C-P по теореме CAP;
- Eventually consistent
  Данные в конце концов попадут на все узлы, но нет требования, чтобы все узлы всегда содержали самые свежие данные.

# MongoDB


## Возможности:

- Авто-Sharding (горизонтальное масштабирование);
- Большие наборы данных могут быть разделены и распределены по нескольким шардам;
- Быстрые обновления на месте;
- Операции обновления являются атомарными для производительности без конкуренции;
- Интегрированный Map-Reduce;

# MongoDB




## Использование:

- Высокопроизводительные и масштабируемые приложения;
- Большинство веб-приложений, в которых вы ранее использовали SQL;

Не используйте для:

- Приложений, критически чувствительных к выполнению транзакций.

# MongoDB


## Пользователи:






econophysica

# CouchDB


## Возможности:

- CouchDB - это документно-ориентированная СУБД, не реляционная: без схемы базы данных;
- Модель ключ-значение;
- Распределенная и отказоустойчивая;
- Данные моделируются как автономные документы: документ представлен структурой **JSON** с атрибутами любого типа.
- Запросы выполняются с помощью JavaScript.

Очень похожа на ElasticSearch, но с более широким набором запросов, и медленнее.

# CouchDB


## Возможности:

- Различные типы данных поддерживаются как дополнительные документы (видео, аудио, изображения и т. Д.)
- Связь с приложениями и пользователями осуществляется через RESTful сервисы : «Передача репрезентативного состояния» - программная модель клиент-серверной архитектуры, используемая для распределенных систем
- Протокол связи HTTP:
  - методы HTTP используются явно;
  - Stateless
  - Выставляет структуру через URI
  - Данные передаются в формате XML или JSON (для CouchDB).

# CouchDB


## HTTP:

**Протокол:**

- **GET** извлекает ресурс, на который ссылается URI.
- **PUT** создает ресурс по указанному URI.
- **POST** отправляет сообщение (вместе с некоторыми данными) существующему
  ресурс.
- **DELETE** удаляет ресурс.

Очень удобно в веб-среде: нет необходимости использовать клиентскую библиотеку - Документы могут быть легко включены в веб-интерфейс.

# CouchDB


## Примеры запросов:

Отправить запрос HTTP, получить ответ.

$ curl -X GET http://mycouch.org

{"couchdb":"Welcome","version":"1.0.1"}

Создать базу данных.

$ curl -X PUT http://mycouch.org/myDB

{"ok":true}

Создать документ – поместить в БД ресурс.

$ curl -X PUT http://mycouch.org/myDB/myDoc \

-d '{"key": "value"}'

{"ok":true,"id":"myDoc","rev":"1-25eca"}

Получить документ по его URI:

$ curl -X GET http://mycouch.org/myDB/myDoc

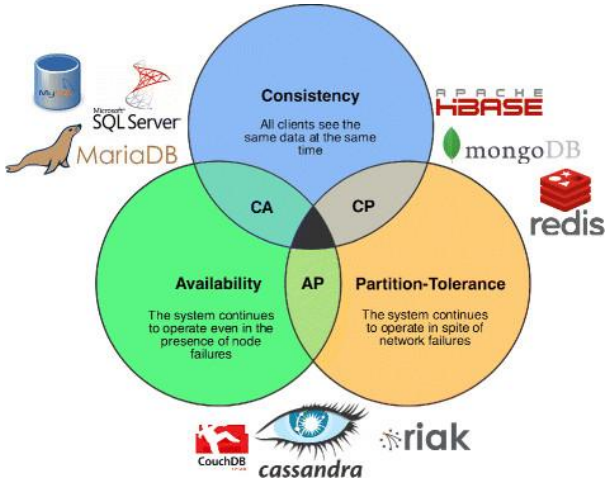
{"_id":"myDoc","_rev":"1-25eca","key":"value"}





econophysica

# Подводя итог:




- Кассандра и CouchDB предлагают доступность.
- Hadoop и MongoDB предлагают согласованность.
- ElasticSearch следует примеру реляционных баз данных и предлагает и то, и другое по цене отказа в работе в случае разбиения кластера на части.

# Flume


## Возможности:

- Сбор, агрегация потоковых данных о событиях;
- Обычно используется для данных журналов событий;
- Значительные преимущества перед специальными решениями;
- Надежный, масштабируемый, управляемый, настраиваемый и высокопроизводительный;
- Декларативная, динамическая конфигурация;
- Контекстная маршрутизация;
- Многофункциональный;
- Полностью расширяемый;

# Flume


## Событие:

***Событие - это основная единица данных, транспортируемых Flume от пункта отправления до конечного пункта назначения. Событие - это полезная нагрузка в виде байтового массива, сопровождаемая необязательными заголовками.***

- Полезная нагрузка непрозрачна для Flume
- Заголовки указываются как неупорядоченная коллекция пар строк (ключ-значение), причем ключи являются уникальными для всей коллекции.
- Заголовки могут быть использованы для контекстной маршрутизации

# Flume


## Клиент:

*Сущность, которая генерирует **события** и отправляет их одному или нескольким **агентам**.*

Примеры:

- Flume log4j Appender
- Пользовательский клиент, использующий Client SDK (org.apache.flume.api)

Отделяет Flume от системы, из которой поступают данные о событиях;

Требуется не всегда.

# Flume


## Агент:

*Контейнер для размещения **источников**, **каналов**, **приемников** и других компонентов, которые позволяют переносить **события** из одного места в другое.*

- Фундаментальная часть потока Flume;
- Обеспечивает поддержку конфигурации, управления жизненным циклом и мониторинга размещенных компонентов.

# Flume


## Источник:

*Активный компонент, который получает **события** из специализированного местоположения или механизма и размещает его на одном или нескольких **каналах**.*

Различные типы источников:

- Специализированные источники для интеграции с известными системами. Пример: системный журнал, Netcat
- Автогенерация источников: Exec, SEQ
- Источники IPC для связи между агентами: Avro

Требуется хотя бы один канал для работы.

# Flume


## Канал:

*Пассивный компонент, который буферизует входящие **события** до тех пор, пока они не будут взяты из канала **приемниками**.*

Различные каналы предлагают разные уровни постоянства хранения:

- Канал памяти: энергозависимый
- Файловый канал: поддерживается реализацией WAL
- Канал JDBC: поддерживается встроенной базой данных

- Канал полностью транзакционный;
- Обеспечивает слабые гарантии сохранения порядка;
- Может работать с любым количеством источников и приемников.

# Flume


## Приемник:

*Активный компонент, который забирает **события** из **канала** и передает их в пункт назначения следующего перехода.*

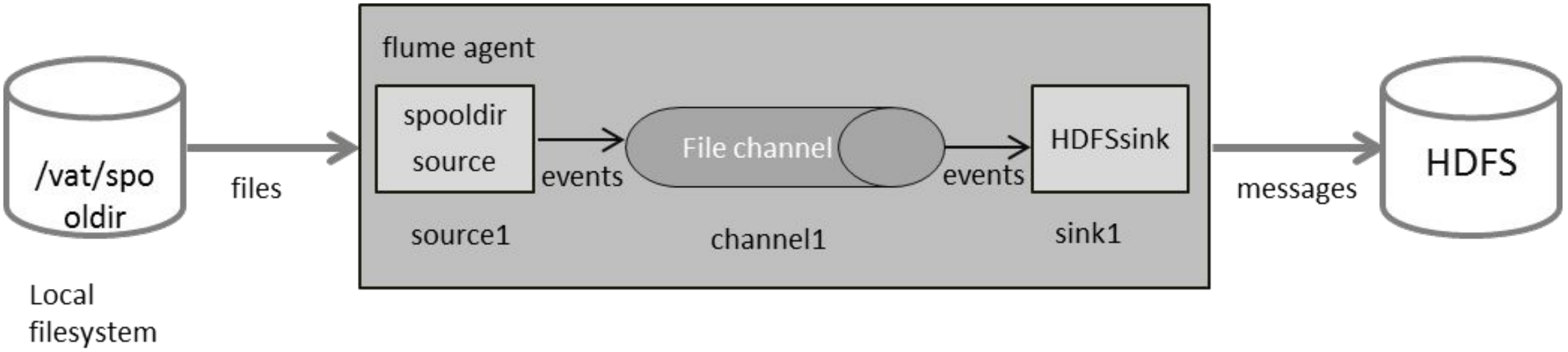
Различные типы приемников:

- Терминальные приемники, которые вносят события в их конечный пункт назначения. Например: HDFS, Hbase
- Авто-потребляющие приемники. Например: null sink.
- Приемник IPC для связи между агентами: Avro

Требуется ровно один канал для работы.

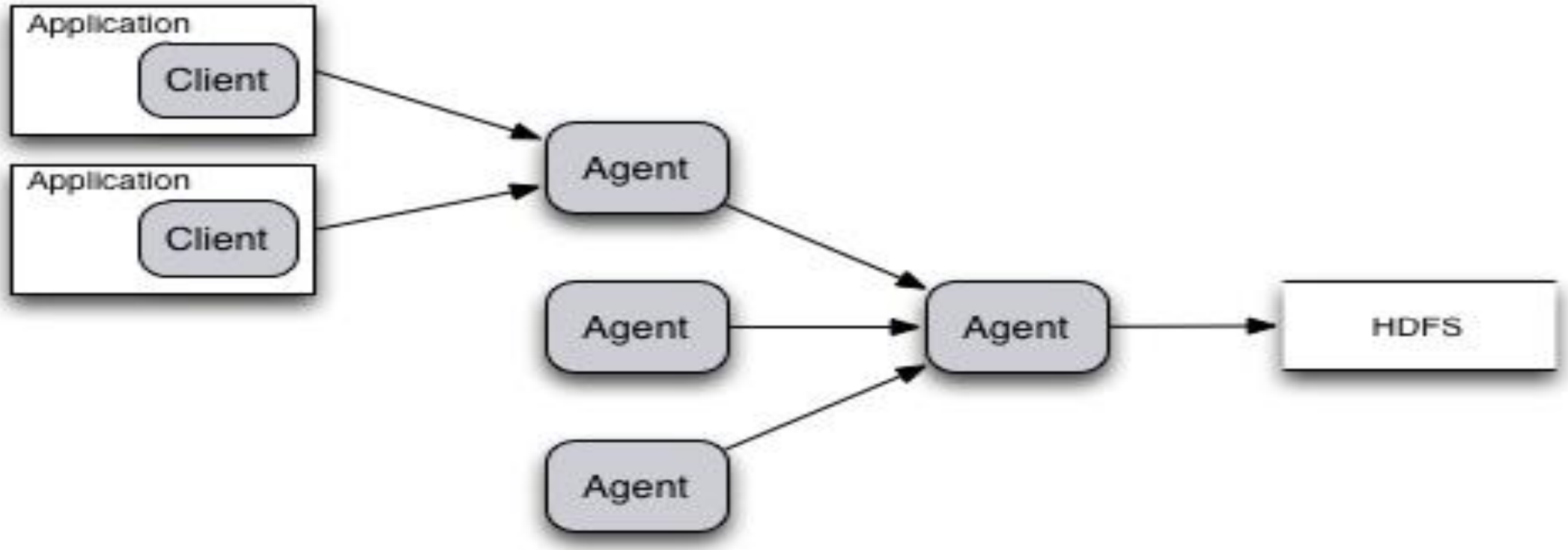
# Flume

## Архитектура:



# Flume


## Архитектура:





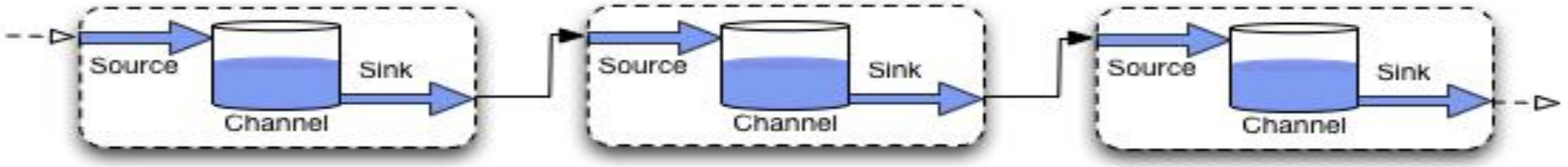



econophysica

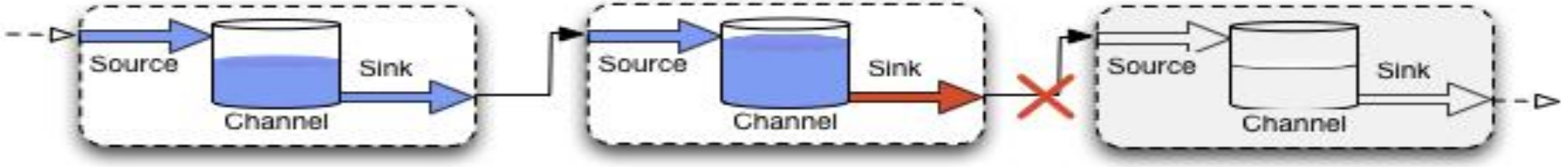
# Flume

## Архитектура:

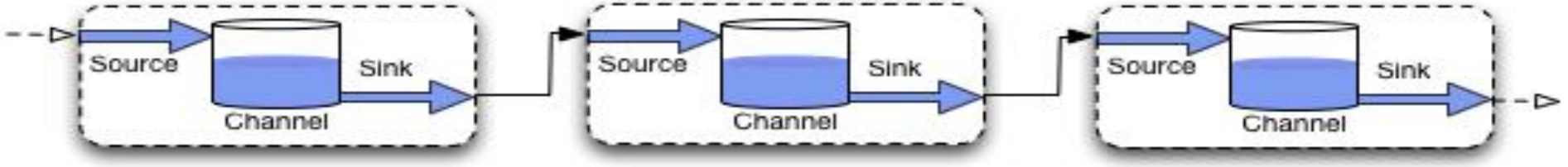
Обычный поток данных




Сбой передачи




Возобновление передачи








econophysica

# Logstash


## Особенности:

Logstash - это приложение, которое собирает файлы журналов с серверов приложений, анализирует их, форматирует и отправляет в Elastic Search. Elastic Search хранит и индексирует данные, которые предоставляет Kibana. Конечные пользователи получают доступ к веб-интерфейсу Kibana для просмотра данных.

Очень быстрое решение, но менее функциональное чем Flume, и ориентированное прежде всего на работу со стеком ELK.

# Kafka

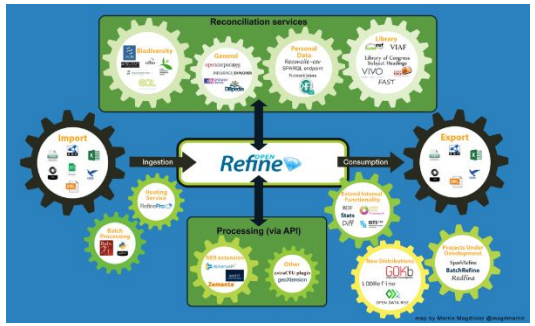

## Особенности:

- Очень высокая производительность;
- Эластически масштабируемая;
- Низкие эксплуатационные расходы;
- Надежная, высокодоступная;

Гарантирует:

- Проверку целостности данных;
- Доставку данных минимум один раз;
- Доставку данных с сохранением порядка, в пределах раздела (partition).

# OpenRefine




## Особенности:

- Открытый инструмент для очистки больших данных, предназначенный для очистки сырых данных.
- Удобный;
- Хорошее сообщество;
- Требуются некоторые знания о подходах к очистке данных.

# DataCleaner

## Особенности:

- Удобный инструмент для очистки больших данных, предназначенный для очистки сырых данных.
- Закрытый исходный код, платный по подписке.





econophysica

# Kibana

## Особенности:

- Веб-плагин для ElasticSearch, который позволяет осуществлять полную визуализацию данных кластера.
- Гибкая платформа для аналитики и визуализации.
- Сводные данные и графики потоковой передачи данных в реальном времени.
- Интуитивно понятный интерфейс для различных пользователей.
- Мгновенный обмен и встраивание панелей мониторинга.

# Matplotlib


## Особенности:

Matplotlib - это библиотека Python 2D для построения графиков, которая генерирует графики уровня публикаций в различных печатных форматах и интерактивных средах на разных платформах.

Matplotlib может использоваться в скриптах Python, оболочках Python и IPython, записной книжке Jupyter, серверах веб-приложений и четырех наборах инструментов графического интерфейса пользователя.

# Tableau

## Особенности:

Tableau - это инструмент визуализации данных, в котором основное внимание уделяется бизнес-аналитике. Вы можете создавать карты, гистограммы, точечные диаграммы и многое другое *без необходимости программирования*. Недавно вышел веб-коннектор, который позволяет подключаться к базе данных или API, что дает возможность получать живые данные в визуализации.

Tableau Public бесплатен, остальные версии платные.

# Python


## Платформы:

- Jupyter Notebooks;
- Matplotlib;
- Hadoop;
- Spark;
- PanDA.

NumPy, SciPy, Scikit-Learn, и т. д.





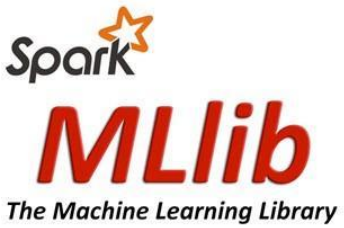
econophysica

# R

## Особенности:

R - это свободная программная среда для статистических вычислений и графики. R предоставляет широкий спектр статистических (линейное и нелинейное моделирование, классические статистические тесты, анализ временных рядов, классификация, кластеризация и т. Д.) И графические методы и обладает широкими возможностями расширения. Язык S часто является средством выбора для исследования в области статистической методологии, а R предоставляет открытый исходный код для участия в этой деятельности.

# MLLib




## Особенности:

- Используется в Java, Scala, Python и R.
- Высококачественные алгоритмы, в 100 раз быстрее, чем MapReduce.
- Работает везде, где работает Spark, на Hadoop, Apache Mesos, Kubernetes, в автономном режиме или в облаке, с различными источниками данных.

# Заключение

При выборе технологий обратите пристальное внимание на сильные и слабые стороны конкретных реализаций, а также на характер ваших данных и компромиссы, на которые вы можете и не можете пойти.

# Спасибо за внимание!

mgubin@tpu.ru

econophysica