САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Базы данных

Лекция 13
Знакомство с большими данными

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

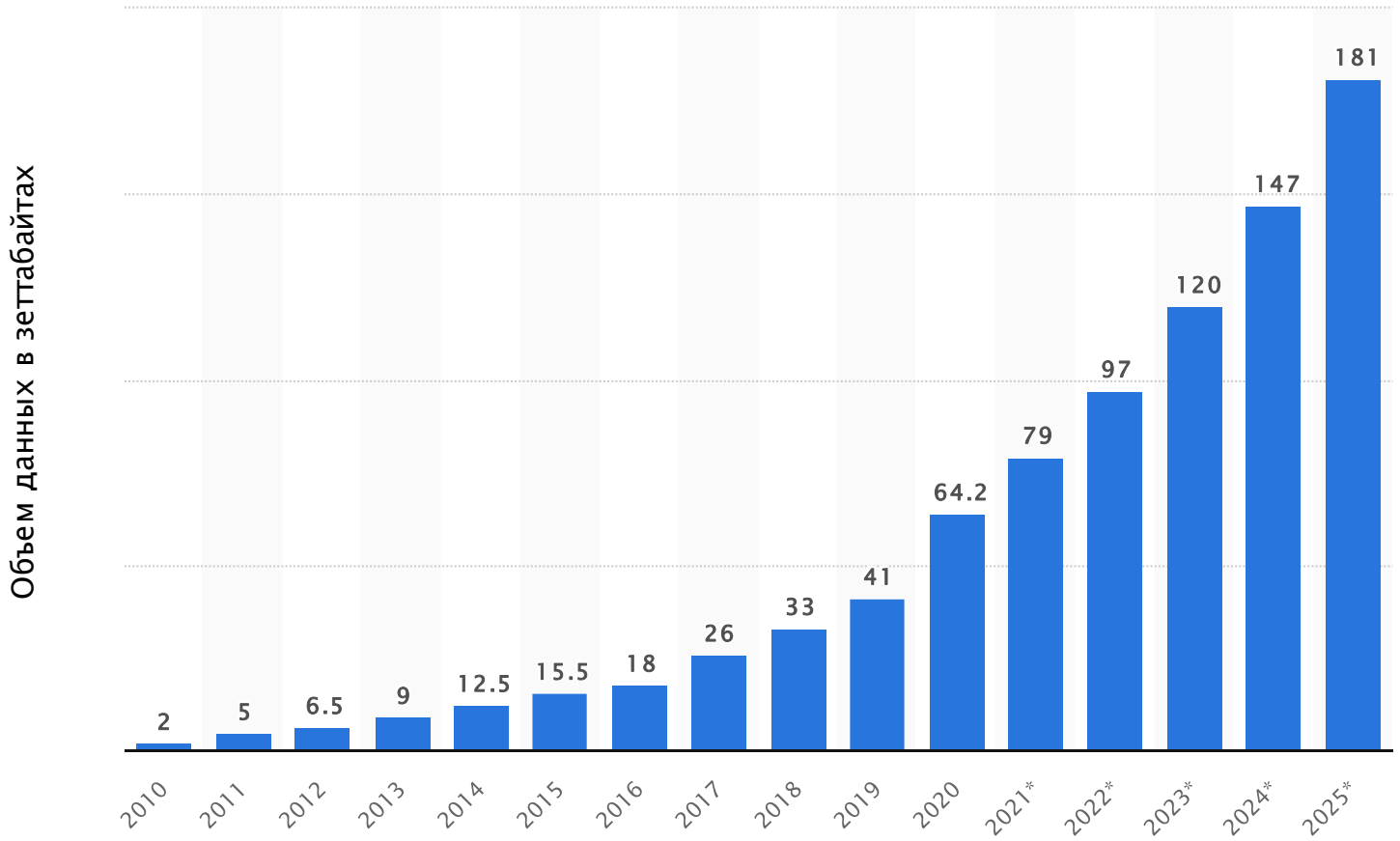
План лекции

- Понятие больших данных (Big Data)
- Принципы работы с большими данными
- Программное обеспечение для работы с большими данными
- Лямбда-архитектура
- Итоги курса

Объемы данных в мире

Источники данных:

- Интернет вещей
- Социальные сети
- Корпоративные данные
- Медицина и биоинформатика
- Астрономические наблюдения



Использование больших данных

- **Государственное управление:** принятие решений в таких областях, как здравоохранение, занятость населения, экономическое регулирование, борьба с преступностью и обеспечение безопасности, реагирование на чрезвычайные ситуации и т.д.;
- **Промышленность:** повышение прозрачности промышленных процессов и внедрение «предиктивное производство», позволяющее более точно прогнозировать спрос на продукцию и, соответственно, планировать расходование ресурсов;
- **Медицина:** разработка новых лекарств, повышение точности определение диагнозов, подборка эффективного лечения, борьба с пандемий;
- **Ретейл:** персонализация ассортимента и маркетинга, снижение стоимости доставки;
- **Интернет вещей:** промышленные и бытовые приборы, подключенные к интернету вещей, собирают огромное количество данных, на основе анализа которых впоследствии регулируется работа этих приборов;
- **Спорт:** анализ перспективных игроков, разработка эффективных стратегий для каждого противника.

Понятие больших данных

Большие данные (Big Data) – это структурированные или неструктурированные данные большого объема и значительного многообразия, эффективно обрабатываемые горизонтально масштабируемыми программными инструментами.

Большие данные (Big Data) – это комплекс технологий, позволяющих обрабатывать как хранящиеся данные, так и поступающий в реальном времени поток разнотипных структурированных и слабоструктурированных данных значительных объемов для получения воспринимаемых человеком результатов.

Технологии обработки больших данных:

- NoSQL
- MapReduce
- Hadoop

Характеристики больших данных

Набор признаков **3V**:

- **Volume** – физический объем данных;
- **Velocity** – скорость прироста данных и скорость быстрой обработки данных с целью получения результатов;
- **Variety** – вариативность, предполагает возможность одновременной обработки различных типов данных;

- **Veracity** – достоверность как самого набора данных, так и результатов его анализа;
- **Variability** – изменчивость форматов, структуры или источников больших данных;
- **Value** – ценность информации, которую можно получить путем обработки и анализа больших наборов данных.

Принципы работы с большими данными

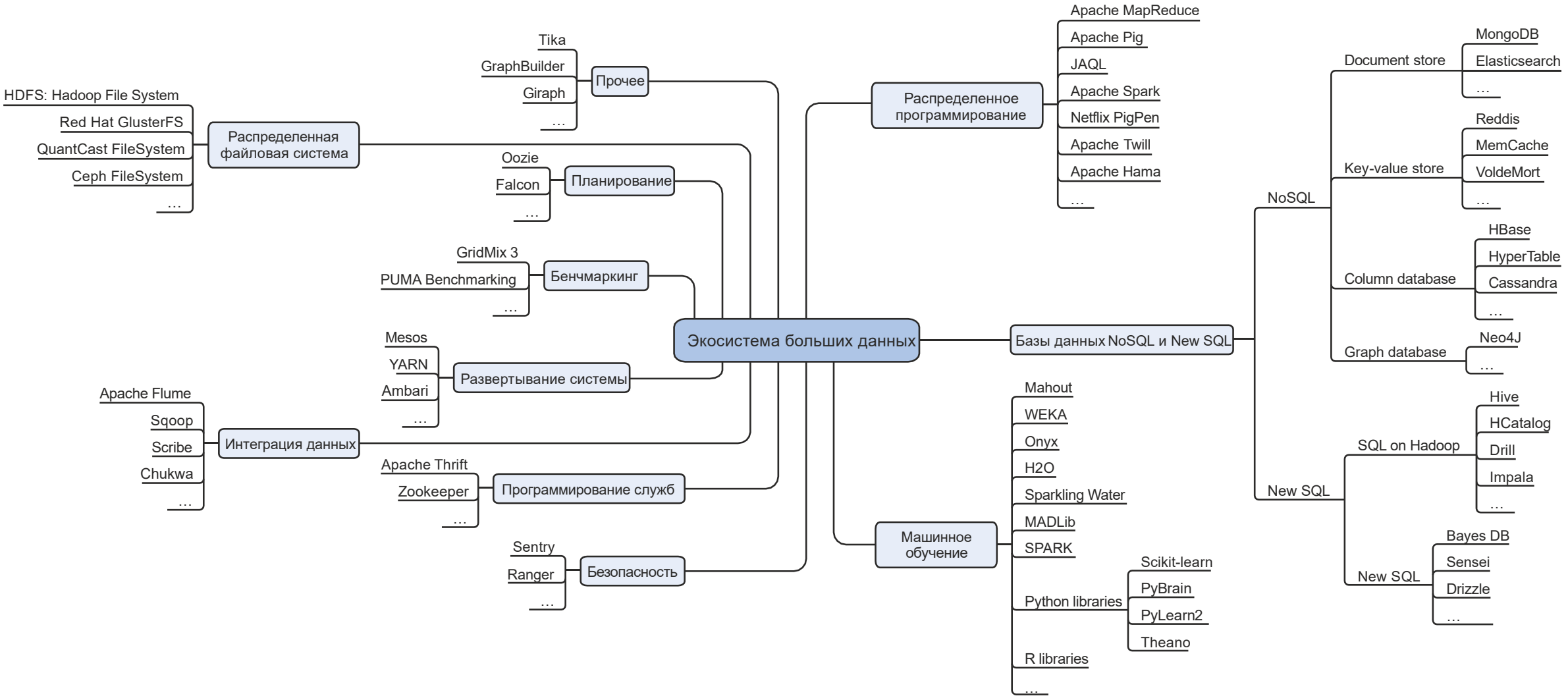
Технология больших данных призвана обеспечить:

1) хранение и управление огромными объемами данных;
2) организацию неструктурированных и слабоструктурированных данных (тексты, изображения, видео и т. д.);
3) получение из больших данных аналитических выводов, которые будут полезны в практической деятельности человека.

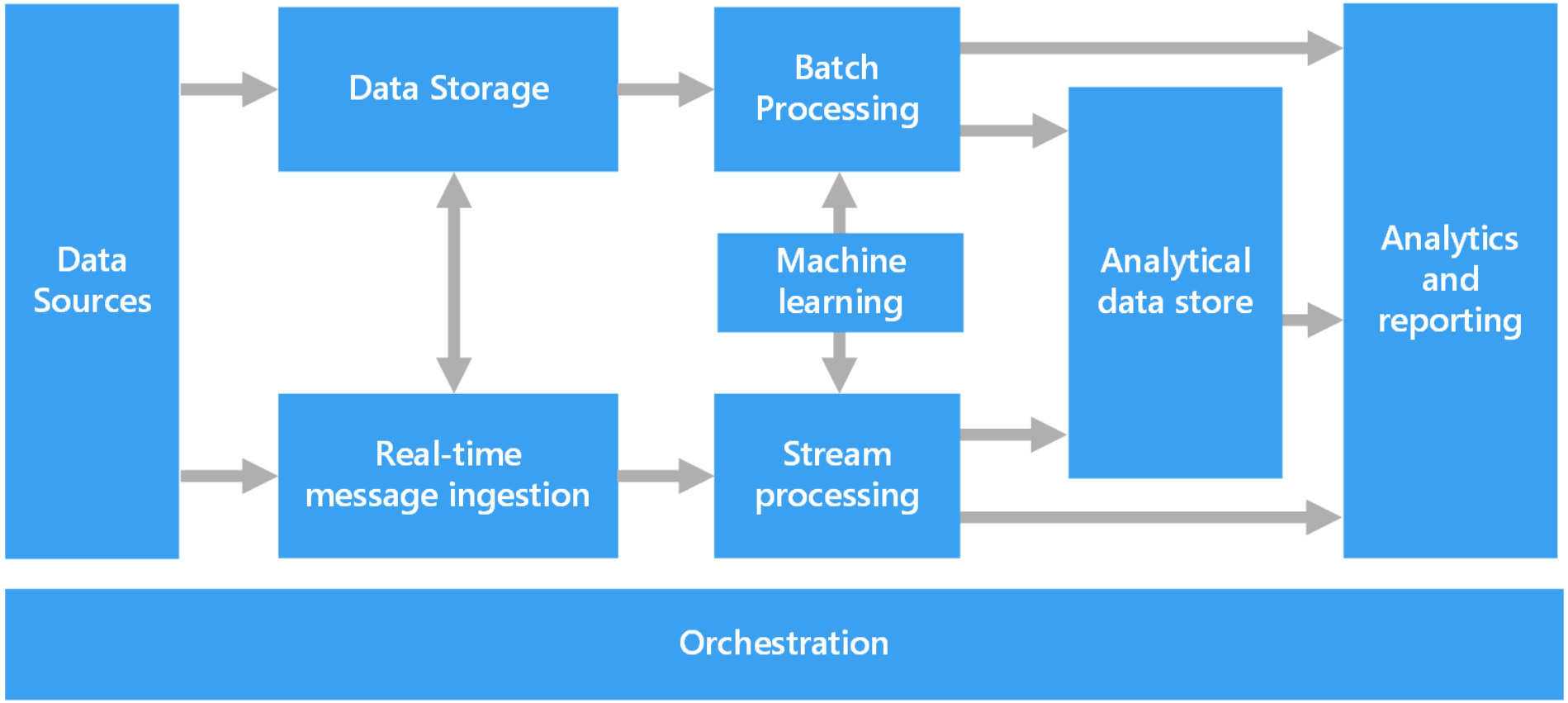
Принципы работы с большими данными:

- горизонтальная масштабируемость, обеспечивающая обработку данных, предполагает, что данные распределены по узлам таким образом, что их обработка не приводит к снижению производительности системы;
- отказоустойчивость, должна позволять свести к минимуму влияние на работу с данными возможные отказы оборудования;
- локальность данных, требует, чтобы по возможности данные обрабатывались на том же компьютере, на котором они и хранятся.

Программное обеспечение для работы с большими данными



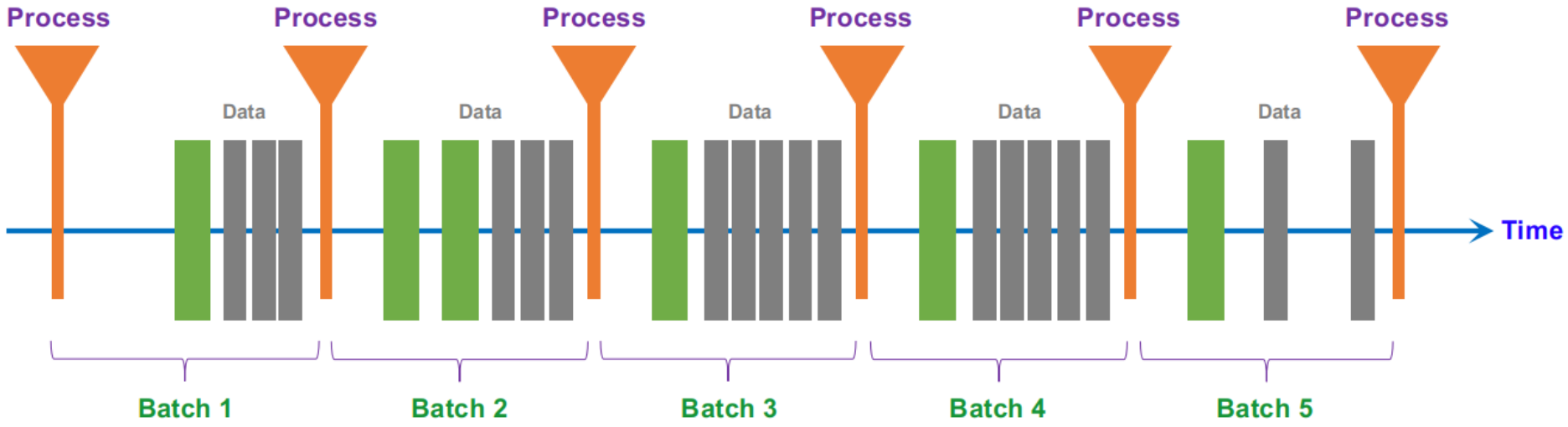
Компоненты архитектуры больших данных



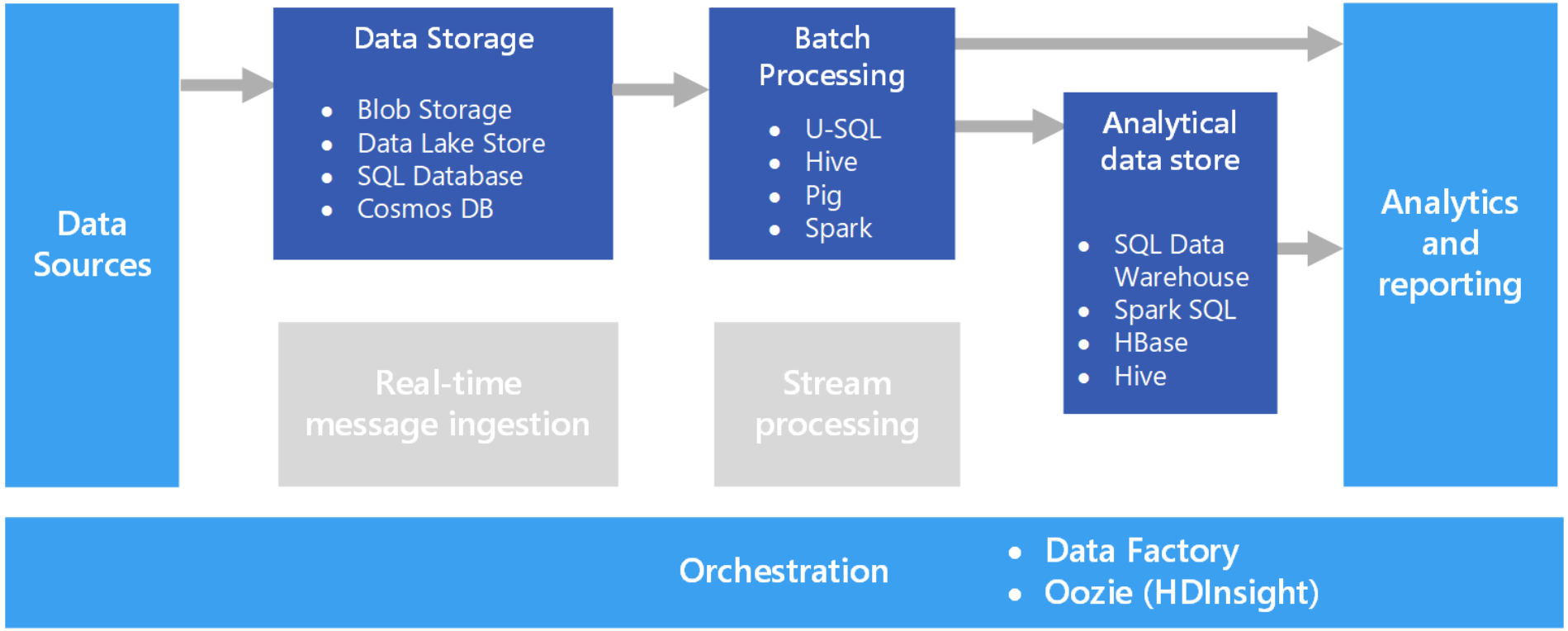
Пакетная обработка

Пакетная обработка (batch processing) – типичный сценарий работы с большими данными. В этом случае исходные данные загружаются в хранилище данных. Затем данные параллельно обрабатываются локально. В рамках обработки может выполняться несколько итеративных шагов до того, как преобразованные результаты будут загружены в хранилище аналитических данных.

- ▲ Высокая эффективность – обрабатывается сразу набор данных.
- ▲ Может выполняться для очень больших наборов данных, которые обрабатываются длительное время.
- ▼ Результаты обработки доставляются с задержкой.



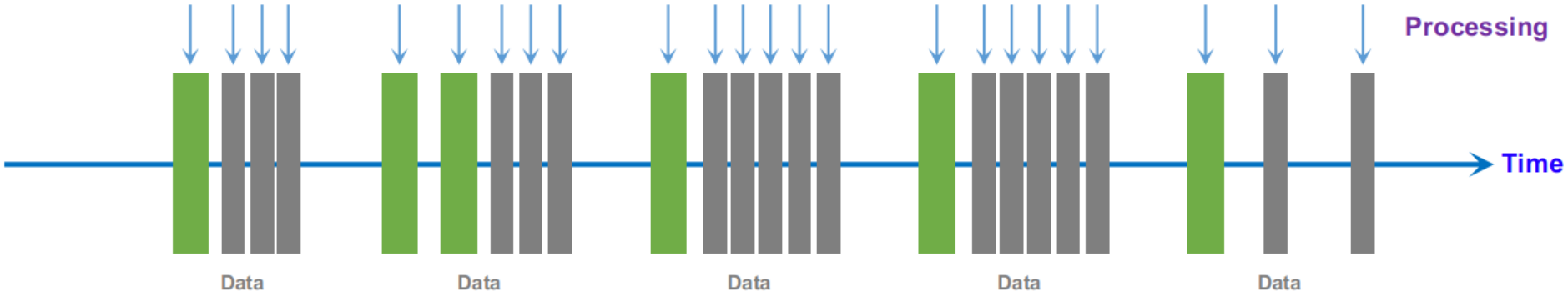
Пакетная обработка



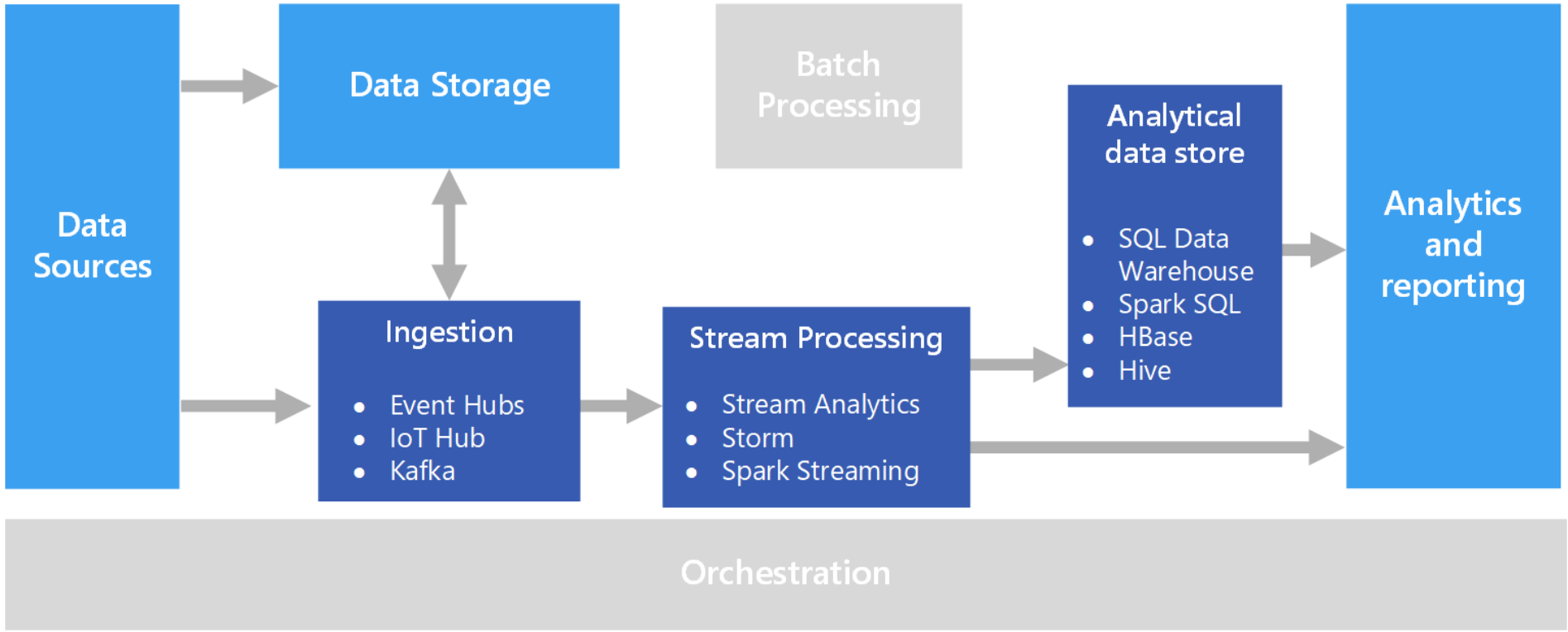
Потоковая обработка

Потоковая обработка (stream processing) выполняется для потоков данных, получаемых в реальном времени и обрабатываемых с минимальной задержкой.

- ▲ Непрерывная обработка входящего потока данных по мере поступления событии.
- ▲ Низкая задержка до доли секунды.
- ▼ Не могут использоваться сложные сценарии обработки данных.
- ▼ Результаты обработки могут быть менее точны или полны, чем при пакетной обработке.

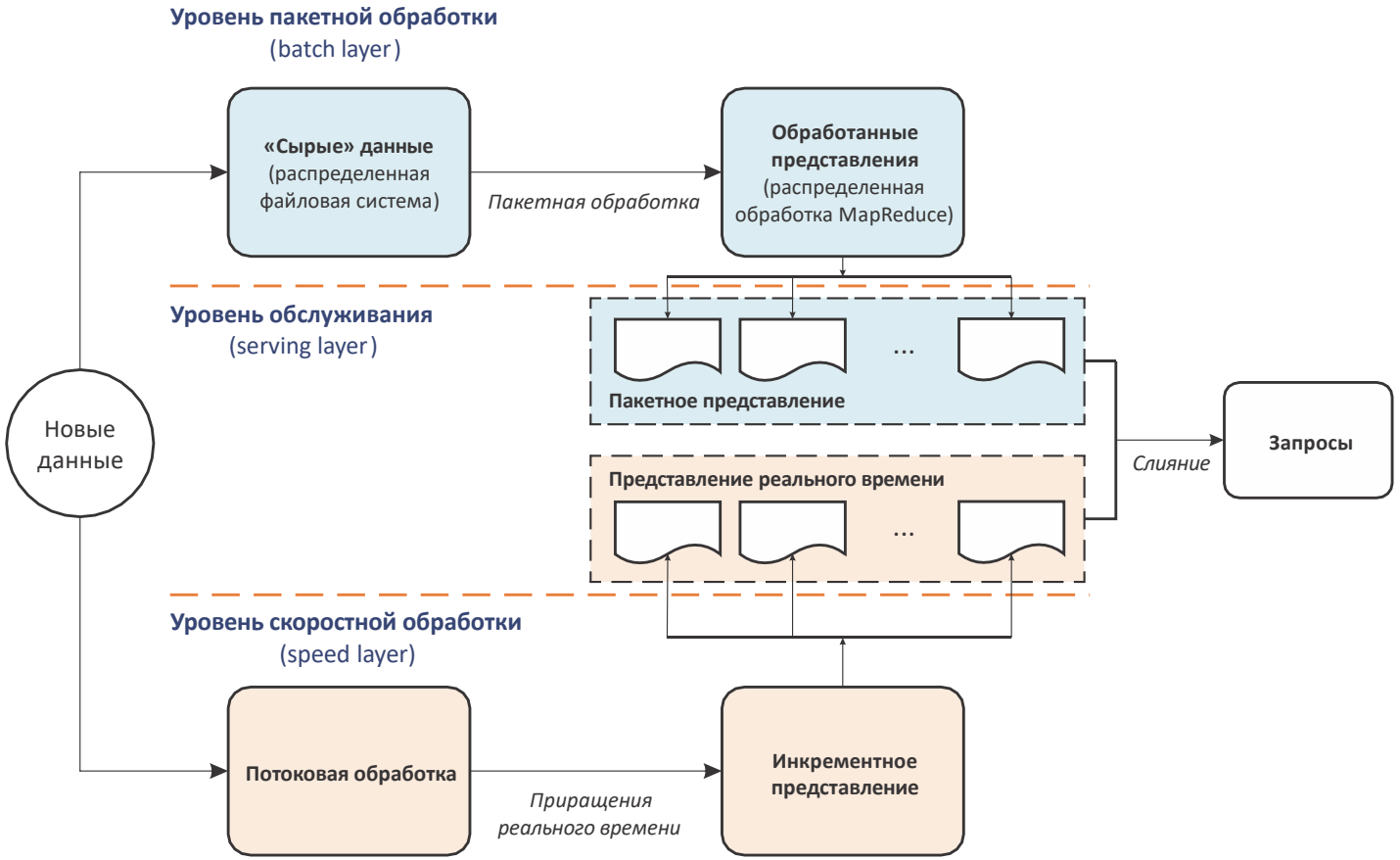


Потоковая обработка



Лямбда-архитектура

Лямбда-архитектура (Lambda Architecture) – это подход к процессингу больших данных, который использует преимущества обработки как пакетных, так и потоковых методов обработки.



Лямбда-архитектура

- Уровень **пакетной обработки** представляет собой хранилище «сырых» данных. Здесь ведется процессинг по расписанию – через заранее заданные интервалы времени отправляются запросы к новым данным. Полученные данные добавляются к накопленному ранее архиву, не изменяя его предыдущие копии. Обычно уровень пакетной обработки реализуется на базе *Apache Hadoop*.

- Уровень **скоростной обработки** осуществляет обработку поступающего в реальном времени потока данных. Работа в реальном времени допускает упрощенную обработку, например можно получить недостаточно точные или неполные данные. Однако эти погрешности с небольшим запозданием компенсируются уровнем пакетной обработки. Используемые инструменты: *Apache Spark, Apache Storm, Apache Samza*.

- Данные, полученные от уровней пакетной и скоростной обработки, сохраняются на **уровне обслуживания**. Слой реагирует на запросы от операторов и возвращает им заранее подготовленные или подготовленные «на лету» представления. На стыке сервисного уровня и уровня скоростной обработки может работать как рассмотренная СУБД *MongoDB*, так и ряд других систем (*Apache Cassandra, Apache HBase*).

Apache Hadoop – инфраструктура, упрощающая работу с компьютерными кластерами.

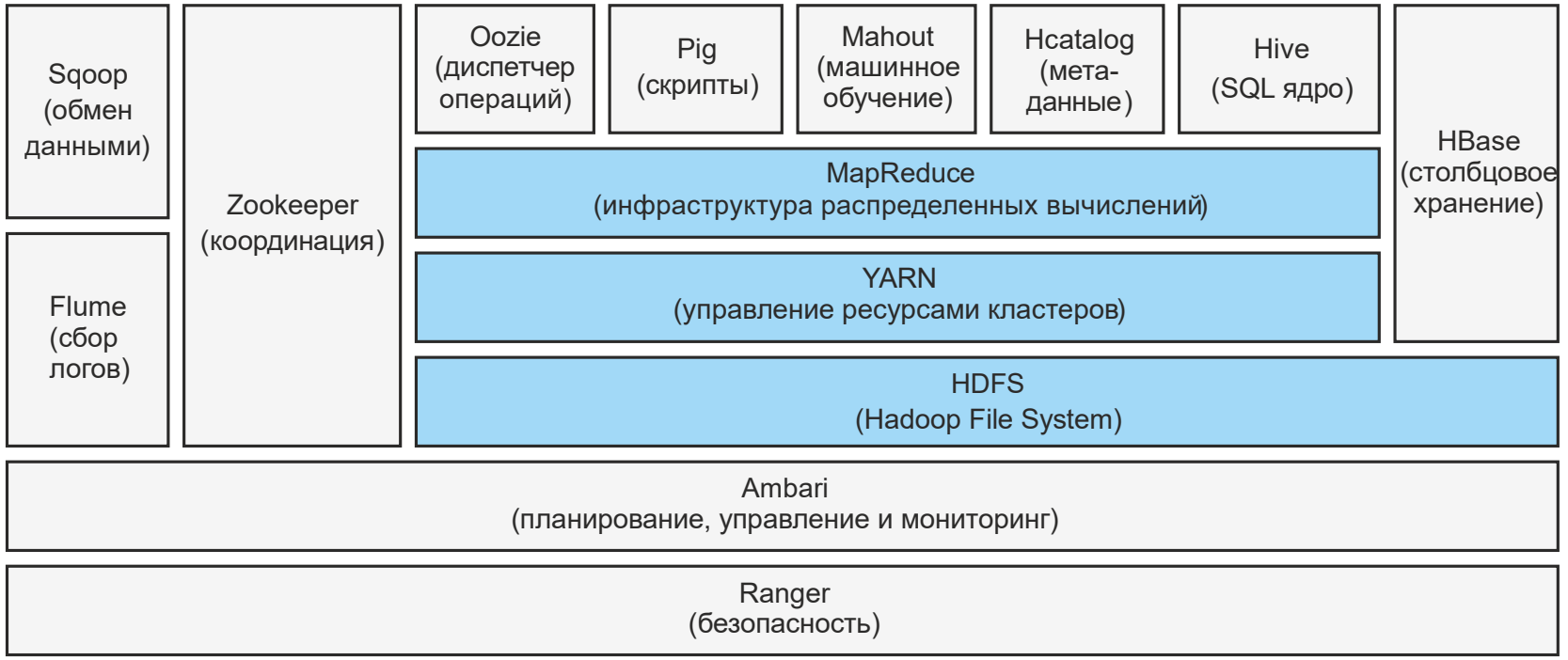
Hadoop пытается достичь следующих целей:

- *Надежность* – достигается путем создания нескольких копий данных и повторного применения логики обработки в случае сбоя;
- *Отказоустойчивость* – обнаружение сбоев и применение автоматического восстановления;
- *Масштабируемость* – данные и их обработка распределяются в кластерах (горизонтальное масштабирование);
- *Портируемость* – возможность установки на различных устройствах и операционных системах.

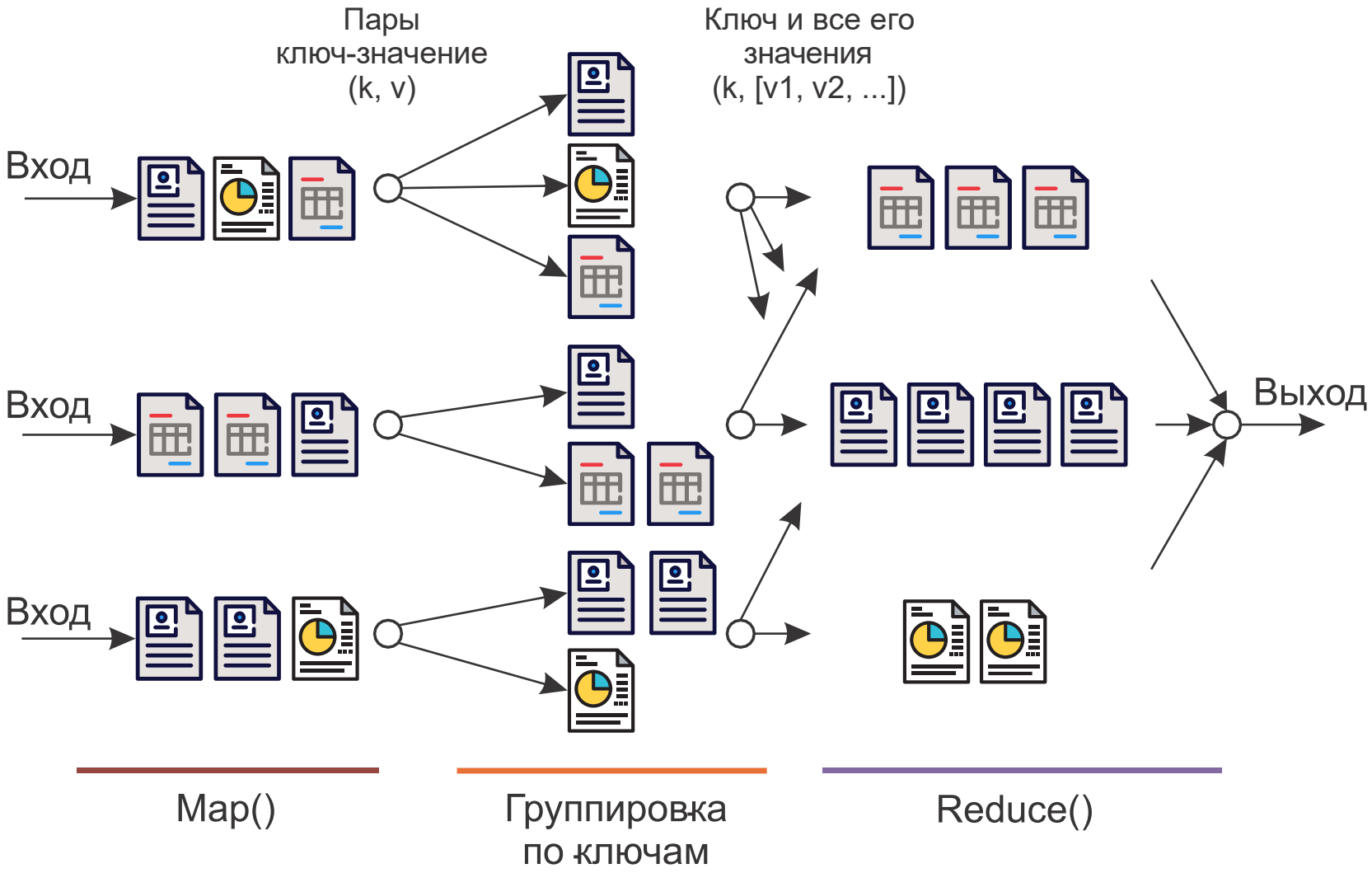
Основные компоненты Apache Hadoop

В Apache Hadoop центральное место занимают три компонента:

- Распределенная файловая система (Hadoop Distributed File System, HDFS);
- Метод распределенных вычислений MapReduce;
- Система управления ресурсами кластера YARN.



Подход MapReduce



Apache Spark

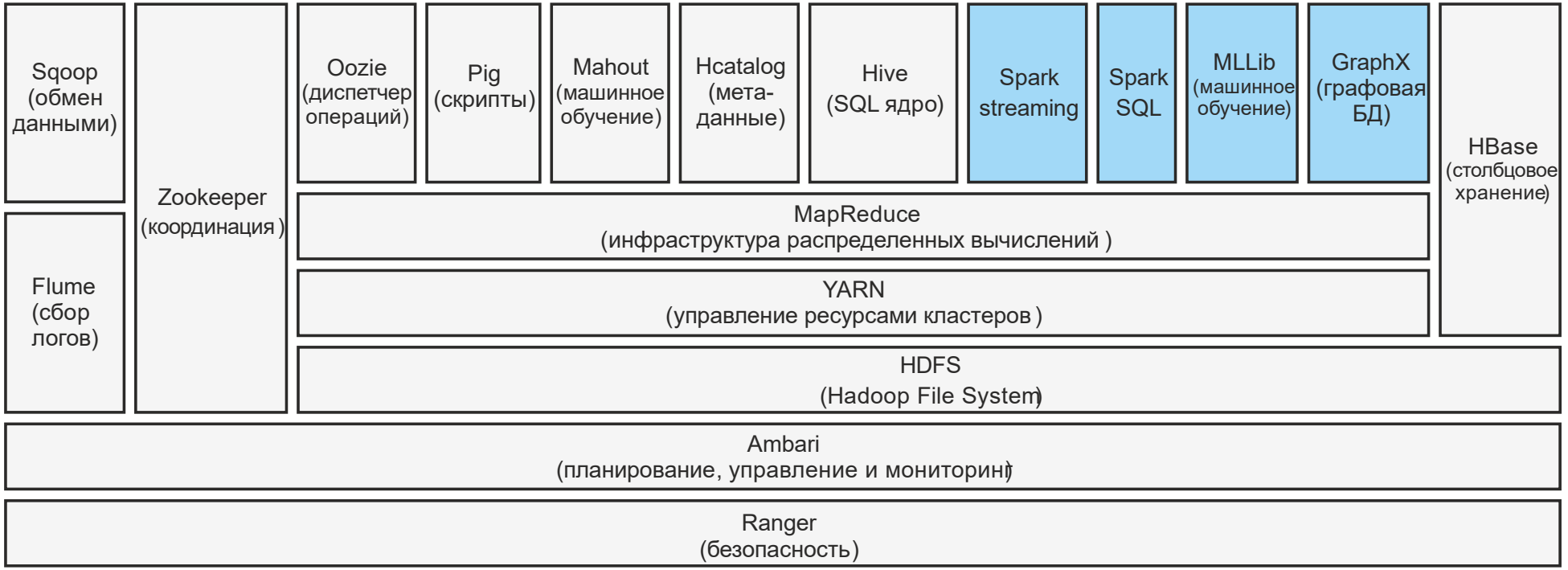
Apache Spark – фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop.

- В отличие от классического обработчика из ядра Hadoop, реализующего двухуровневую концепцию MapReduce с хранением промежуточных данных на базе дискового хранилища, Spark работает в парадигме резидентных вычислений – обрабатывает данные в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы для некоторых классов задач (например, машинного обучения);
- Предоставляет программные интерфейсы для языков Java, Scala, Python, R;
- Поддерживает несколько распределённых систем хранения — HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3.

Основные компоненты Apache Spark

Компоненты Apache Spark:

- Spark Streaming – инструмент анализа потоковых данных в реальном времени;
- Spark SQL – SQL-интерфейс для работы с данными;
- MLLib – инструмент машинного обучения в инфраструктуре Spark;
- GraphX – графовая база данных для распределенной обработки графов.



Apache Kafka – распределенная система обмена сообщениями с высокой пропускной способностью между компонентами программной системы, работающая по принципу «публикация–подписка» (Publish and subscribe). Разрабатывается в рамках фонда Apache.

Особенности системы:

- является распределенной, горизонтально-масштабируемой системой, обеспечивающей наращивание пропускной способности как при росте числа и нагрузки со стороны источников, так и количества систем-подписчиков;
- обеспечивается публикация и подписка на потоки записей;
- поддерживается отказоустойчивый способ хранения потоков записей за счет применения техники, сходной с журналами транзакций;
- поток записей обрабатывается по мере появления записей с возможностью временного хранения данных для последующей пакетной обработки.

Пример системы с лямбда-архитектурой

Система для обработки данных о движении «подключенных» транспортных средств:

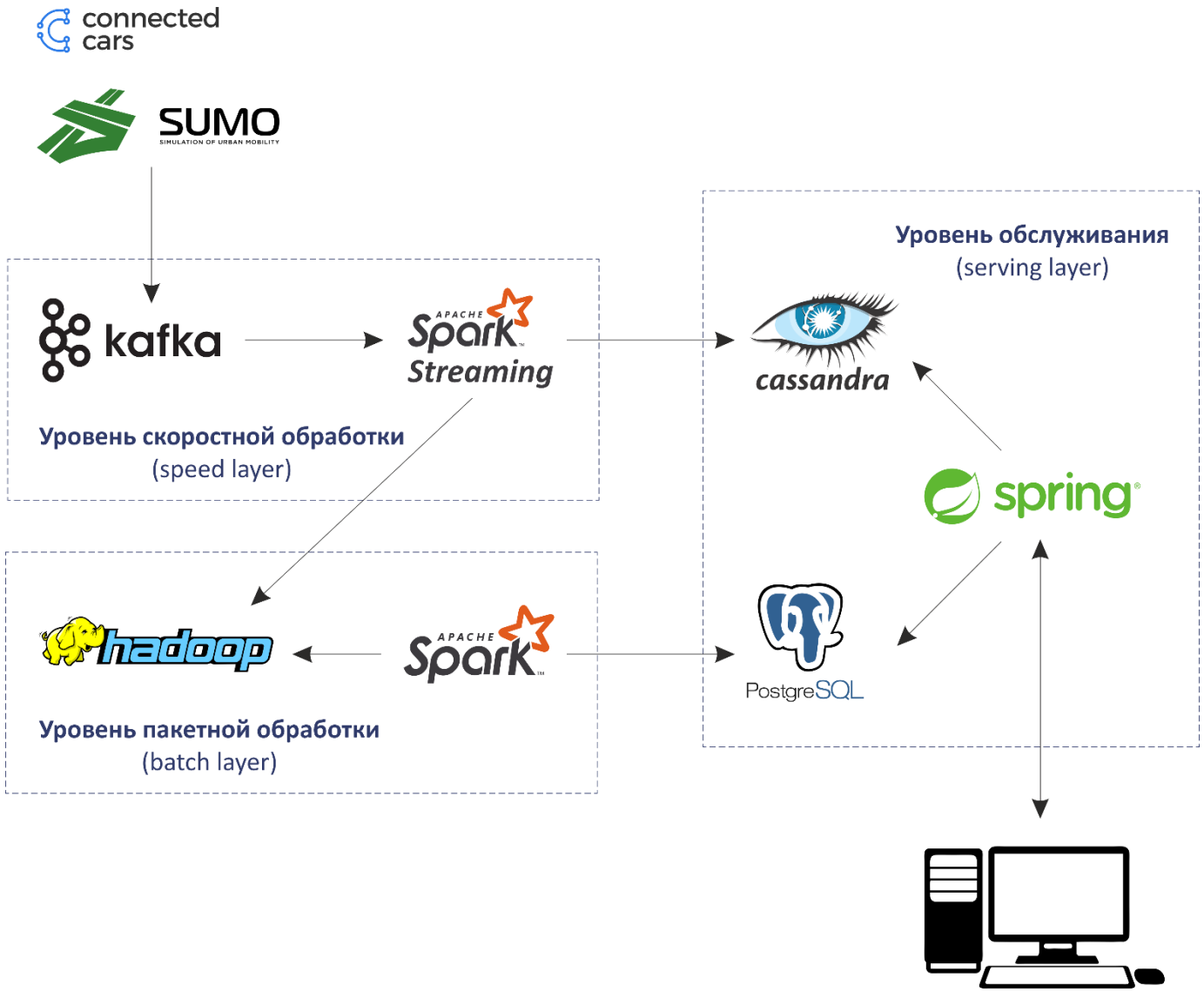
- Транспортные средства передают (каждую секунду) информацию о своем положении и мгновенной скорости движения;
- Скорости движения по каждому сегменту сети усредняются (за каждый 5-минутный интервал движения) – текущая скорость;
- На основе текущей скорости и архивных данных выполняется прогноз скорости движения на следующий интервал;
- Исходные данные о положения транспортных средств и результат обработки должны быть доступны пользователю посредством веб-приложения.

Пример системы с лямбда-архитектурой

Решаемые системой задачи по уровням обработки лямбда-архитектуры можно распределить следующим образом:

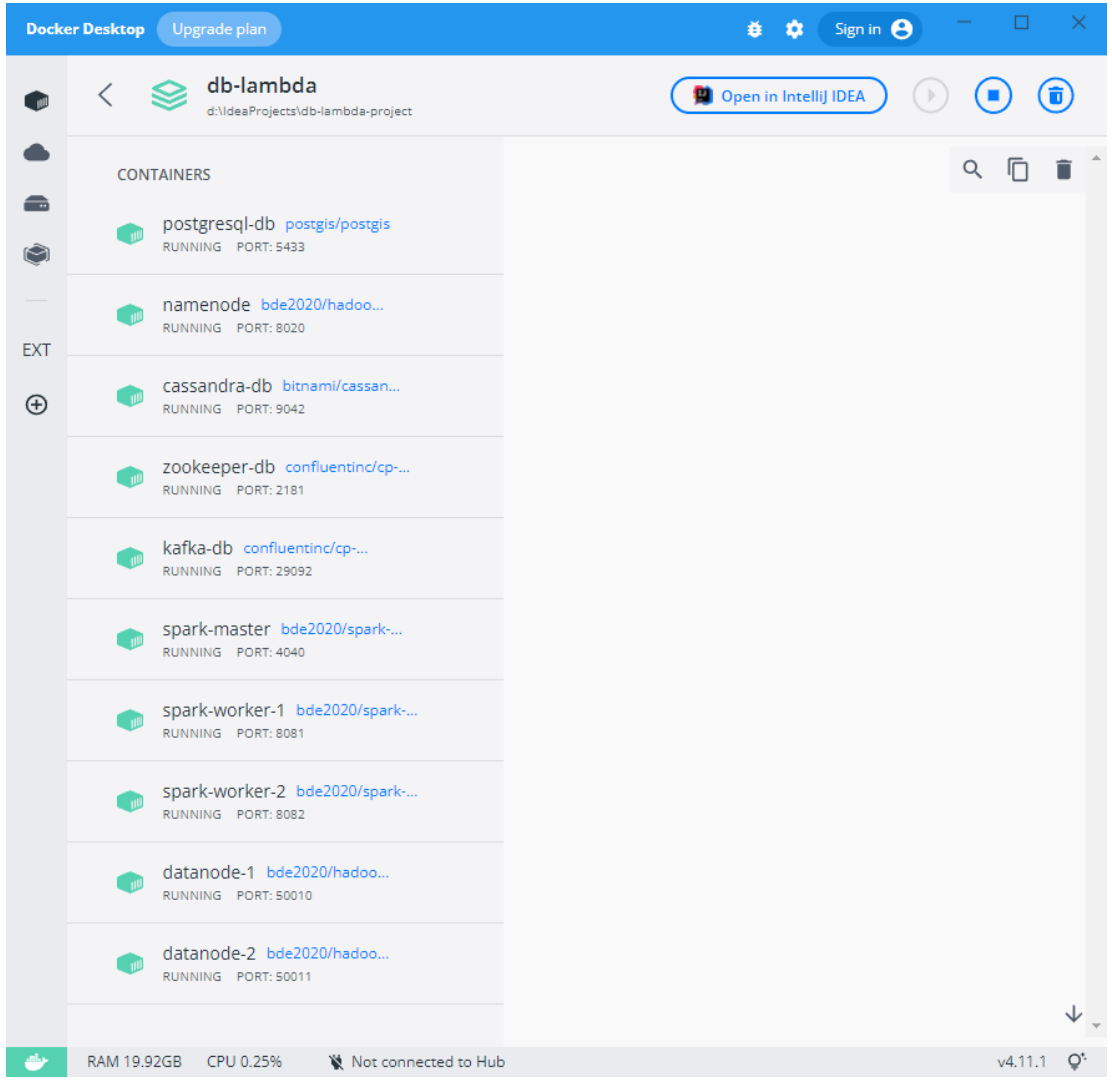
1. *Уровень скоростной обработки* (speed layer) – получение новых сообщений о параметрах движения транспортных средств, сохранение данных на диск для последующей пакетной обработки и в БД для выдачи пользователям.
2. *Уровень пакетной обработки* (batch layer) – чтение информации с диска и
   a) усреднение мгновенной скорости за 5-минутный интервал;
   b) обучение модели прогнозирования скорости движения на основе накопленных данных;
   c) прогнозирование скорости движения с использованием обученной модели прогнозирования и архивных/текущих данных о скорости движения;
   d) сохранение результатов анализа в базу данных;
3. *Уровень обслуживания* (service layer) – обработка http-запросов пользователей и выдача исходных и обработанных данных.

Выбранные инструменты

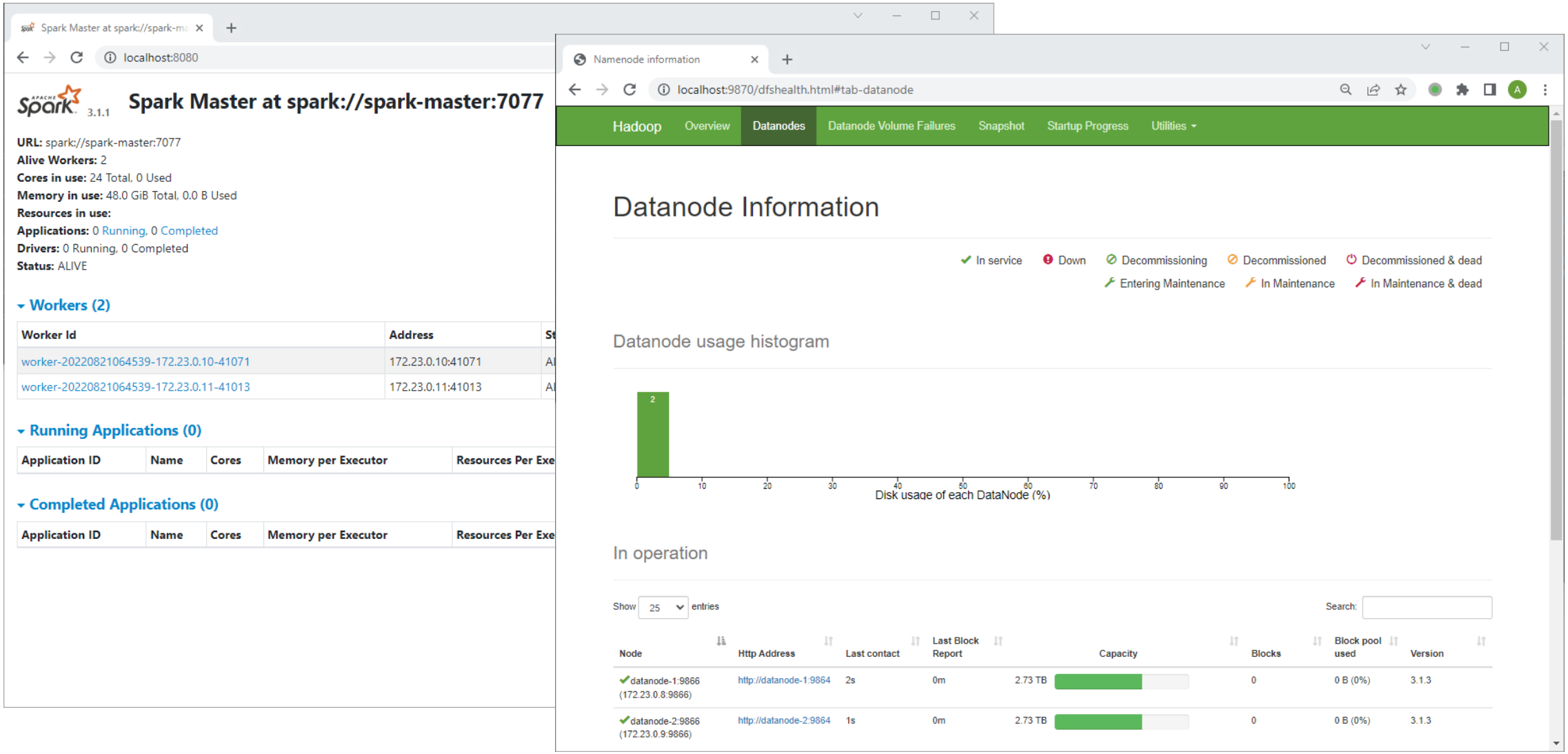


Развертывание компонентов системы. Docker

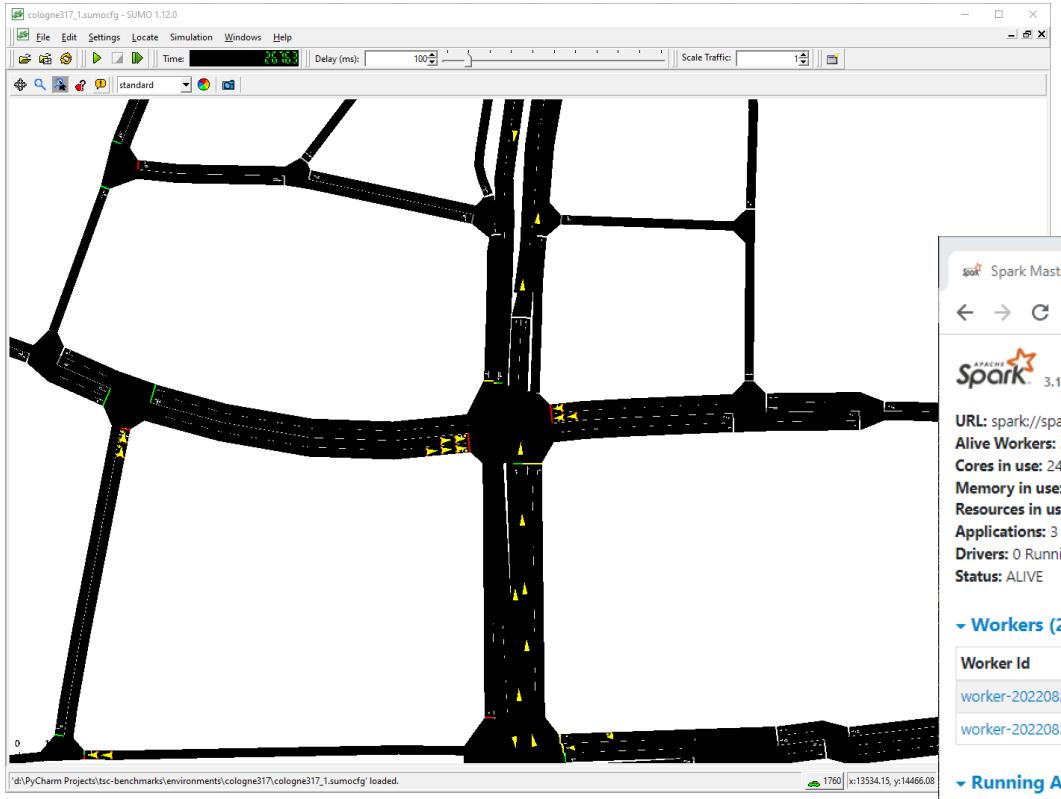
```
services:
  zookeeper:
      image: confluentinc/cp-zookeeper:7.2.1
      container_name: zookeeper-db
      ...
  kafka:
      image: confluentinc/cp-kafka:7.2.1
      container_name: kafka-db
      ...
  cassandra:
      image: 'bitnami/cassandra:latest'
      container_name: cassandra-db
      ...
  postgresql:
      image: postgis/postgis
      container_name: postgresql-db
      ...
  spark-master:
      image: bde2020/spark-master:3.1.1-hadoop3.2-java11
      container_name: spark-master
      ...
  spark-worker-1:
      image: bde2020/spark-worker:3.1.1-hadoop3.2-java11
      container_name: spark-worker-1
      ...
  namenode:
      image: bde2020/hadoop-namenode:2.0.0-hadoop3.1.3-java8
      container_name: namenode
      ...
  datanode-1:
      image: bde2020/hadoop-datanode:2.0.0-hadoop3.1.3-java8
      container_name: datanode-1
      ...
```



Проверка компонентов системы. Spark. Hadoop



Обработка данных в Spark




```
docker exec spark-master /spark/bin/spark-submit --class
edu.db.lambda.streaming.StreamingProcessor  --master
spark://localhost:7077 /opt/spark-data/db-spark-
processor-1.0.0.jar
```

Spark Master at spark://spark-ma

localhost:8080

Spark 3.1.1

Spark Master at spark://spark-master:7077

URL: spark://spark-master:7077
Alive Workers: 2
Cores in use: 24 Total, 18 Used
Memory in use: 48.0 GiB Total, 6.0 GiB Used
Resources in use:
Applications: 3 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220821113711-172.23.0.10-33007	172.23.0.10:33007	ALIVE	12 (9 Used)	24.0 GiB (3.0 GiB Used)	
worker-20220821113712-172.23.0.11-36159	172.23.0.11:36159	ALIVE	12 (9 Used)	24.0 GiB (3.0 GiB Used)	

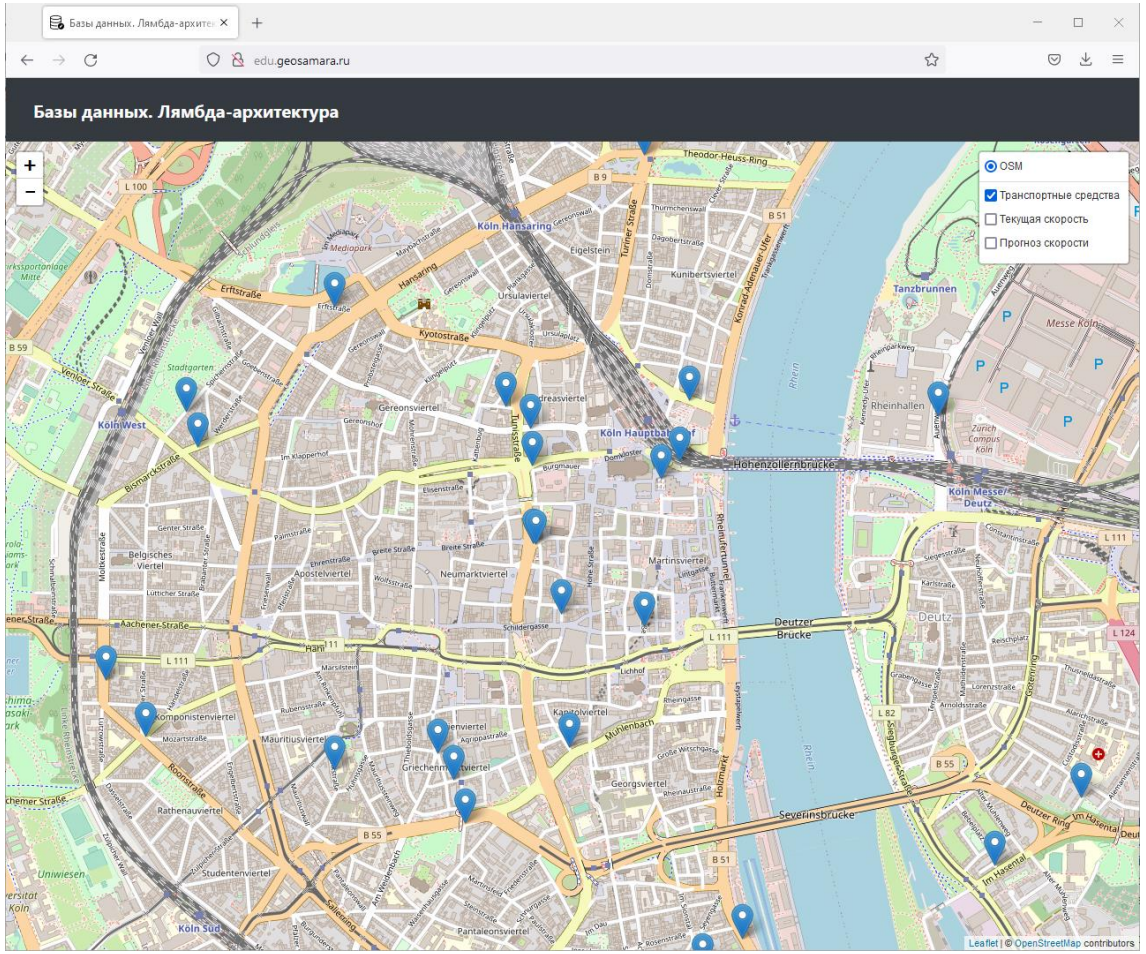
Running Applications (3)

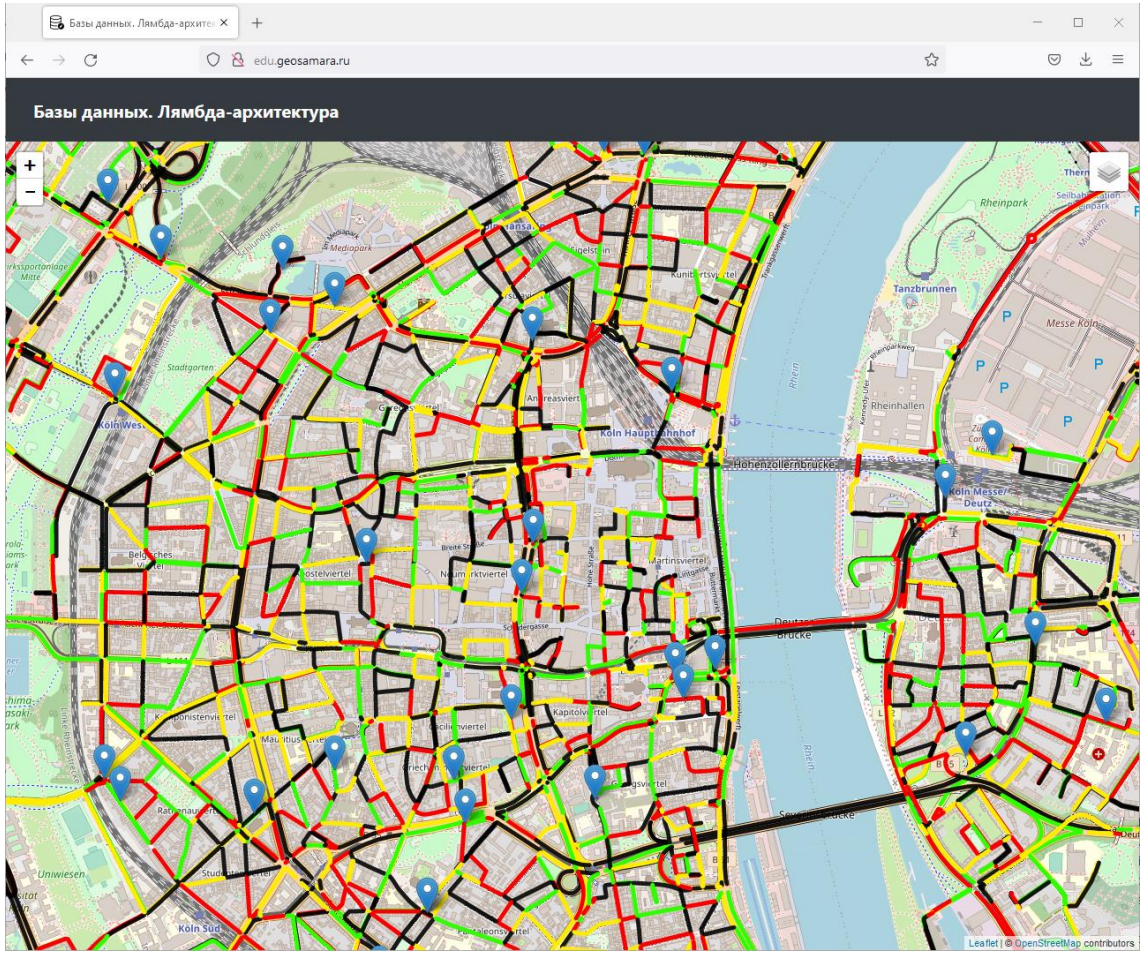
Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20220821113859-0003	(kill)	speed-prediction	6	1024.0 MiB		2022/08/21 11:38:59	root	RUNNING	13 s
app-20220821113851-0002	(kill)	batch-processor	6	1024.0 MiB		2022/08/21 11:38:51	root	RUNNING	21 s
app-20220821113732-0000	(kill)	streaming-processor	6	1024.0 MiB		2022/08/21 11:37:32	root	RUNNING	1.7 min

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20220821113819-0001	ml-training	6	1024.0 MiB		2022/08/21 11:38:19	root	FINISHED	19 s

Отображение результатов анализа





Заключение. Рассмотренные темы

- Реляционные базы данных
  - Проектирование баз данных
  - Язык SQL
  - Процедурные расширения языка SQL
  - Транзакции
  - Работа со слабоструктурированными данными
- NoSQL базы данных
  - Типы NoSQL решений
  - Документоориентированная БД MongoDB
- Большие данные

Заключение. Что осталось за кадром

Вопросы безопасности в СУБД:

- Резервное копирование данных
- Репликация данных
- Сегментирование (шардинг) баз данных
- Администрирование прав доступа
- Шифрование критических данных

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ