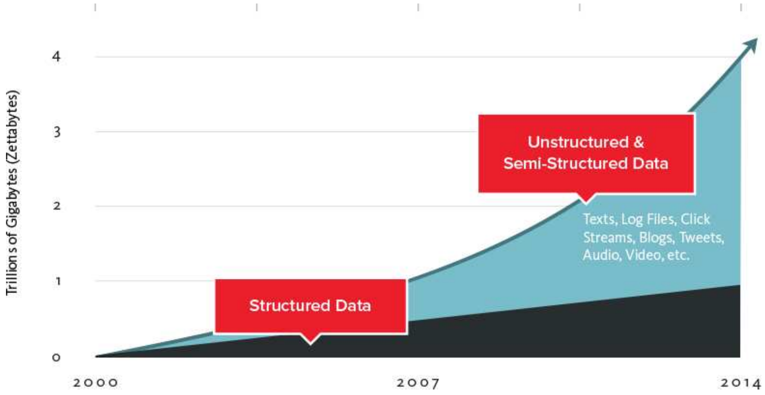
From Big Data towards Fast Data

Vojtěch Juránek

JBoss - a division by Red Hat

6. 2. 2016, DEVCONF.CZ, Brno

Data today



Source: `http://www.couchbase.com/nosql-resources/what-is-no-sql`

How big are Big data?

How big are Big data?

Source: `https://twitter.com/DEVOPS_BORAT/status/288698056470315008`

How big are Big data?

Source: `https://twitter.com/DEVOPS_BORAT/status/288698056470315008`

- You can scale up, but sooner or later you'll probably have to scale out
- Need for highly scalable solution also because of cost effectiveness

Big data - challenges and approaches

- Analysis run on top of the huge amount of data
- Ability to store huge amount of unstructured data (often for performance reasons)
- But also ability to talk to RDBMS or query structured data is often needed as well
- Scalable solution
- Cloud architecture - everything is ephemeral

Big data - challenges and approaches

- Analysis run on top of the huge amount of data
- Ability to store huge amount of unstructured data (often for performance reasons)
- But also ability to talk to RDBMS or query structured data is often needed as well
- Scalable solution
- Cloud architecture - everything is ephemeral

How these challenges are usually addressed:

- Data replication
- Map-reduce model

Big data - challenges and approaches

- Analysis run on top of the huge amount of data
- Ability to store huge amount of unstructured data (often for performance reasons)
- But also ability to talk to RDBMS or query structured data is often needed as well
- Scalable solution
- Cloud architecture - everything is ephemeral

How these challenges are usually addressed:

- Data replication
- Map-reduce model

Probably the most popular implementation:

- hadoop

Speeding up! I.

Speeding up! I.

Keep computation intensive data in memory

Speeding up! I.

Keep computation intensive data in memory

Don’t replicate every single change of the data

Apache Spark

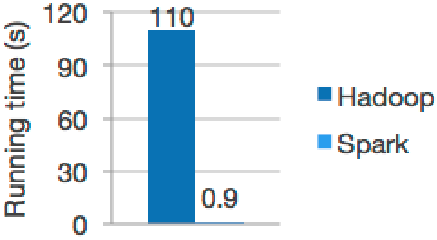
Resilient Distributed Dataset (RDD)

- Immutable distributed collection of data
- RDD is split into multiple partitions - can be located on different nodes
- Generated by a set of deterministic operations applied on a data source or other RDDs
- Provides 2 types of operations:
  - **transformation** creates new RDD (e.g. `map()` or `filter()`) - return type is always RDD
  - **action** computes a result from RDD (e.g. `count()` or `first()`)
- Lazy evaluation - only upon calling action on RDD
- RDD contains enough information (its linage) to be (re)created from a stable source

See M. Zaharia et al., NSDI, 2012.

Apache Spark

- For some type of jobs (e.g. iterative algorithms) substantial speed up
- Speed up of one, sometimes even two orders of magnitude



Logistic regression (an ML algorithm for classification) in Hadoop and Spark
Source: `http://spark.apache.org/`

Speeding up! II.

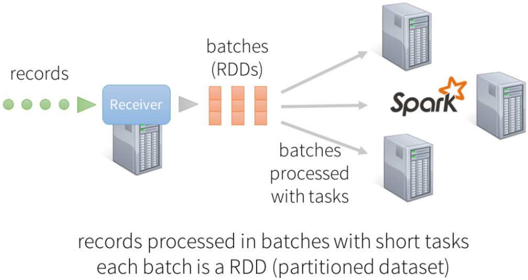
Speeding up! II.

Process data immediately once it arrives

Spark streaming

- Discretized Streams (DStreams) - RDD micro-batches
- User defined (time) size of the batch

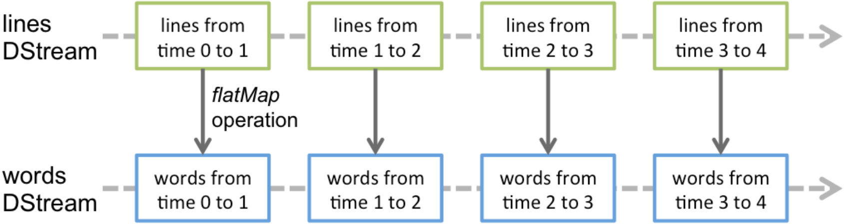
```
val conf = new SparkConf().setAppName("WordCount")
val ssc = new StreamingContext(conf, Seconds(1))
```



Source: `https://databricks.com/blog/2015/07/30/diving-into-spark-streamings-execution-model.html`

Spark streaming

```scala
// Split each line into words
val words = lines.flatMap(_.split(" "))
```



Source: `http://spark.apache.org/docs/latest/streaming-programming-guide.html`

Homework for you: Real-time stream processing frameworks

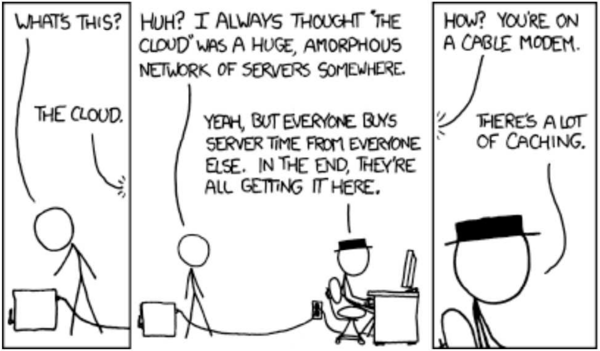
Apache Storm

Apache Flink

Apache Samza

Speeding up! III.

Keep the data in memory all the time



Source: Part of xkcd #908

In-memory data grid: Infinispan

http://infinispan.org/

- Data grid platform, written in Java
- In-memory No-SQL key-value data store, (optionally) schema-less
- Distributed cache - offers massive memory
- Elastic and scalable - can run on hundreds of nodes
- Highly available - no SPOF, resilient to node failures
- Transactional
- Supports indexing and searching
- Many other features

Infinispan integration with Spark

- Connector enables read ISPN data from Spark or write Spark data to ISPN
- Spark partitions contain only cache segments owned by the associated ISPN server

Creating RDD from data in ISPN cache

```
val config = new Properties
config.put("infinispan.rdd.cacheName", "my-cache")
config.put("infinispan.client.hotrod.server_list", "
    127.0.0.1:11222")
val ispnRDD = new InfinispanRDD[String, String](sc,
    configuration = config) //for String String pairs
```

Creating DStream from data in ISPN cache

```
//same config as in previous example
val ispnStream = new InfinispanInputDStream[String, Double
    ](ssc, StorageLevel.MEMORY_ONLY, config)
```

Infinispan integration with Spark

- ISPN server side filters and converters can be used for adjusting RDDs when created
- ISPN queries can be applied to RDDs

Creating RDD/DStream by querying ISPN cache

```
val query = Search.getQueryFactory(cache).from(classOf[User
    ]).having("name").equal("Vojtech").toBuilder[RemoteQuery
    ].build
val filteredRDD = rdd.filterByQuery(query, classOf[User])
```

Writing RDD/DStream to ISPN cache

```
//same config as in previous examples
InfinispanDStream[String, Double](temperatureStream).
    writeToInfinispan(config)
```

Few other ISPN highlights wrt. fast data processing

- Event listeners

```java
@ClientListener
public static class MyListener {
    @ClientCacheEntryCreated
    @ClientCacheEntryModified
    public void onEntryChange(ClientCacheEntryModifiedEvent<String> event) {
       //TODO some action when entry is created or modified
    }

    @ClientCacheEntryRemoved
    @ClientCacheEntryExpired
    public void entryRemove(ClientCacheEntryRemovedEvent<String> event) {
      //TODO some action when entry is removed or expired
    }
}
```

- Continuous query

```java
QueryFactory qf = Search.getQueryFactory(myCache);
Query query = qf.from(User.class).select("name").having("age").lte(30).toBuilder().build();
ContinuousQueryListener<Object, Object> listener = new MyListenerI<Object, Object>();
ContinuousQuery<Object, Object> cq = new ContinuousQuery<>(cache);
cq.addContinuousQueryListener(listener, query);
```

- Distributed streams - implementation of `java.util.stream.Stream` over (distributed!) cache data

Where do we get from Big data?

- Data is kept in memory all the time and thus processing and exchanging the data is much faster
- Data is processed once it arrives
- Results of the analysis can be pushed to user by various means, e.g. using continuous queries

Where do we get from Big data?

- Data is kept in memory all the time and thus processing and exchanging the data is much faster
- Data is processed once it arrives
- Results of the analysis can be pushed to user by various means, e.g. using continuous queries

Where do we get from Big data?

- Data is kept in memory all the time and thus processing and exchanging the data is much faster
- Data is processed once it arrives
- Results of the analysis can be pushed to user by various means, e.g. using continuous queries

Where do we get from Big data?

- Data is kept in memory all the time and thus processing and exchanging the data is much faster
- Data is processed once it arrives
- Results of the analysis can be pushed to user by various means, e.g. using continuous queries

Where do we get from Big data?

- Data is kept in memory all the time and thus processing and exchanging the data is much faster
- Data is processed once it arrives
- Results of the analysis can be pushed to user by various means, e.g. using continuous queries

== Fast data?

”Hello world” Demo

Infinispan integration with Apache Spark:

Temperature average

- Stream of temperature measurements from different places stored into Infinispan
- Average temperature is continually recomputed for each place in Spark
- Results are stored back in Infinispan

Sources available on

https://github.com/vjuranek/presentations/tree/master/DevConf_Brno2016

"Hello world" Demo

Source: `https://twitter.com/DEVOPS_BORAT/status/222837225921060864`

"Hello world" Demo

Source: `https://twitter.com/DEVOPS_BORAT/status/222837225921060864`

- One Infinispan server for storing incoming data and results
- One app randomly generating place and temperature (simulating e.g. network of temperature sensors)
- Spark streaming for computing the average temperature at given place
- Client app showing result data when they arrive, using Infinispan cache listener

"Hello world" Demo

Source: `https://twitter.com/DEVOPS_BORAT/status/222837225921060864`

- One Infinispan server for storing incoming data and results
- One app randomly generating place and temperature (simulating e.g. network of temperature sensors)
- Spark streaming for computing the average temperature at given place
- Client app showing result data when they arrive, using Infinispan cache listener

"Hello world" Demo

Source: `https://twitter.com/DEVOPS_BORAT/status/222837225921060864`

- One Infinispan server for storing incoming data and results
- One app randomly generating place and temperature (simulating e.g. network of temperature sensors)
- Spark streaming for computing the average temperature at given place
- Client app showing result data when they arrive, using Infinispan cache listener

"Hello world" Demo

Source: `https://twitter.com/DEVOPS_BORAT/status/222837225921060864`

- One Infinispan server for storing incoming data and results
- One app randomly generating place and temperature (simulating e.g. network of temperature sensors)
- Spark streaming for computing the average temperature at given place
- Client app showing result data when they arrive, using Infinispan cache listener

Summary

Do the data analysis with frameworks which keep the data in memory during processing

Summary

Process data once it arrives

Summary

If possible, keep data in memory during whole application stack

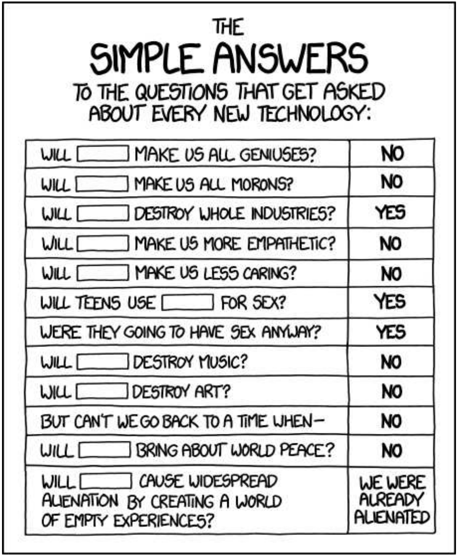
Summary

Infinispan provides many useful features like integration with Apache Spark, continuous query, cache listeners and many others

Summary

- Do the data analysis with frameworks which keep the data in memory during processing
- Process data once it arrives
- If possible, keep data in memory during whole application stack
- Infinispan provides many useful features like integration with Apache Spark, continuous query, cache listeners and many others

Question?



Source: `https://xkcd.com/1289/`

http://infinispan.org/

Thank you for your attention!