

Tech planet 2016

2016.10.17(MON)

Coex Grandballroom

Tech planet 2016

Apache Spark은
어떻게 가장 활발한
빅데이터 프로젝트가 되었나

김상우 @ VCNC (비트윈)



Apache Spark?

**요즘 대세인
인메모리 컴퓨팅
(빅데이터) 프레임워크**



**수많은 데이터를
(수GB ~ 수천 TB)**

**여러대의 컴퓨터를 이용해서
(1대 ~ 수천대)**

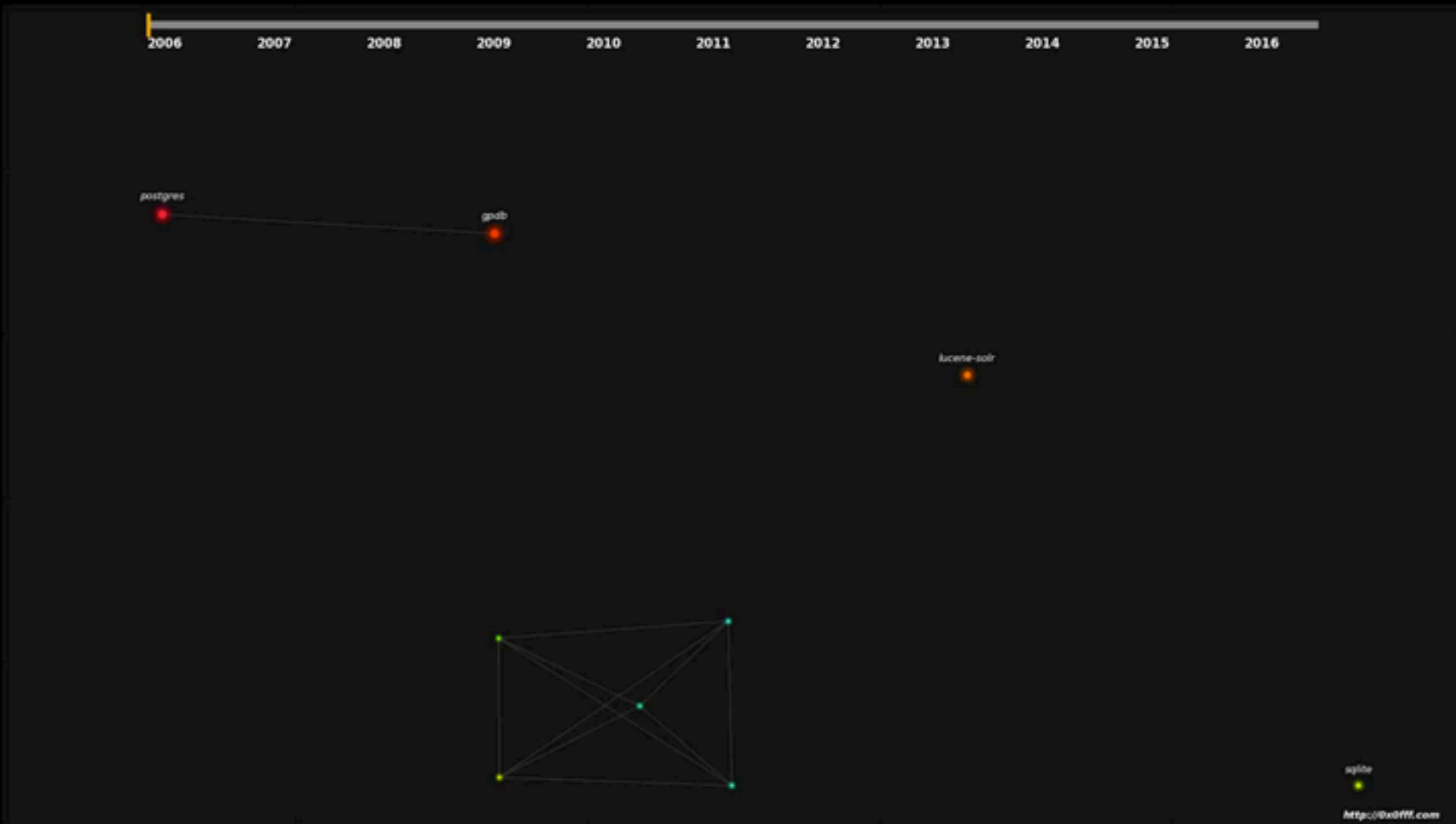
빠르게 분석

발표자: 김상우

- 글로벌 커플 앱 비트윈 개발, 데이터 분석
- 한국 스파크 사용자 모임 (스사모*) 공동 설립 및 운영
- Apache Spark 기반 노트북 프로젝트인 Apache Zeppelin 커미터

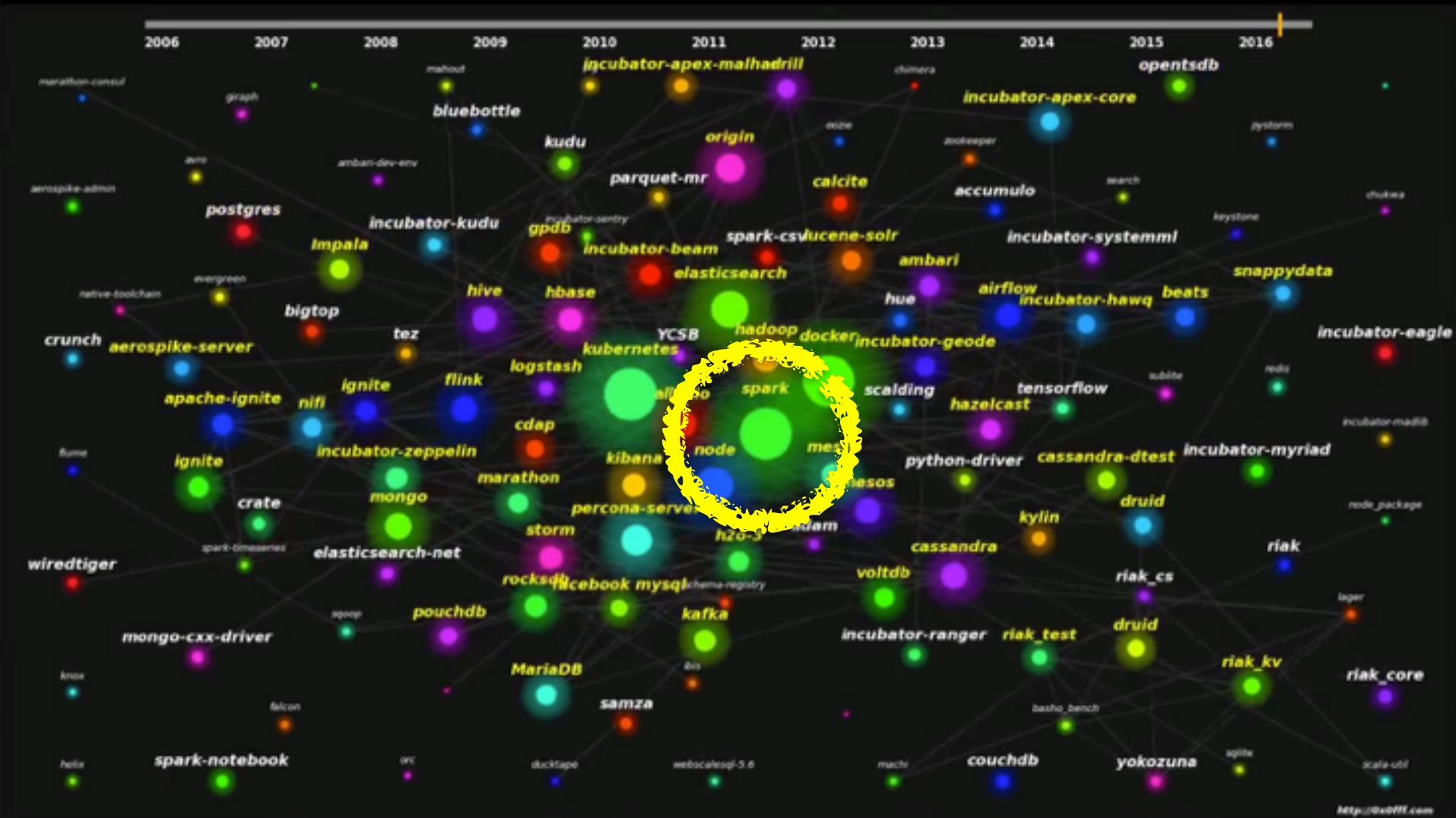


최근 10년 간 빅데이터 오픈소스 프로젝트들의 세력도 (영상)



별의 크기: Contributor 수 (*오픈소스 프로젝트를 평가하는 데 객관적인 하나의 지표가 됨)

최근 10년 간 빅데이터 오픈소스 프로젝트들의 세력도 (영상)



별의 크기: Contributor 수 (*오픈소스 프로젝트를 평가하는 데 객관적인 하나의 지표가 됨)

**Spark, 왜 이렇게
인기가 있을까?**

오픈소스 사용자의 속마음

- Spark 라고 새로 나온 오픈소스 프로젝트가 괜찮다는데.. 한번 써볼까
- 새로나온거 공부도 해야되고 설치나 설정 등 할게 많겠네.
- 좋다는데, 실제로 돌려보면 별로인거 아냐?
- 이미 있는 시스템들 다 바꿔야하는데 잠깐 유행하다가 버려지면 어떻게하지?
- 아직 쓰는 사람도 없고 불안정하겠지?

새로운 기술에 대한

호기심

vs.

의구심

오픈소스 사용자의 속마음

- Spark 라고 새로 나온 오픈소스 프로젝트가 괜찮다는데.. 한번 써볼까
 - 일단 홈페이지가 깔끔하네
- 새로나온거 공부도 해야되고 설치나 설정 등 할게 많겠네.
 - 대부분이 자동으로 설정된다.
- 좋다는데, 실제로 돌려보면 별로인거 아냐?
 - 실제로 돌려보면 굉장히 좋다!
- 이미 있는 시스템들 다 바꿔야하는데 잠깐 유행하다가 버려지면 어떻게하지?
 - Apache 탑레벨 프로젝트. 망하진 않겠지
- 아직 쓰는 사람도 없고 불안정하겠지?
 - 초기엔 불안정했으나 1.0 버전이 나오면서 대부분의 문제가 해결됨



오 이거 대박인데?

공부좀 해볼까

Spark 알아보기

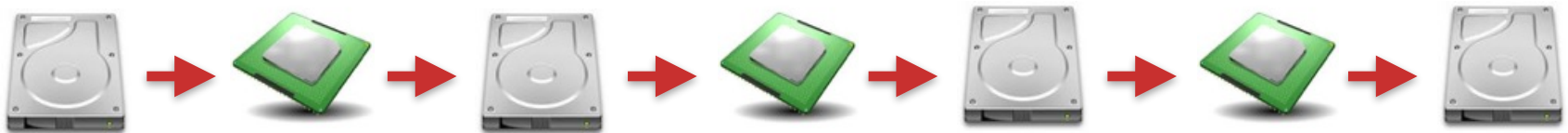
Apache Spark의 3가지 특징

- 인메모리 컴퓨팅 엔진의 압도적인 성능 (x2~x100)
- 쉽고 중독성있는 사용법
- SQL, 스트리밍, 머신러닝, R 등 확장성

(초보자를 위해 매우 쉽게 알아보자)

인메모리 컴퓨팅 엔진 (1)

기존의 빅데이터 분석 방법의 무식함



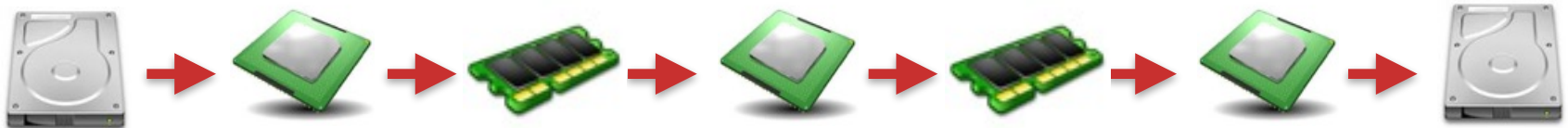
단순한 연산인 경우 나쁘지 않지만
반복연산일수록 효율이 매우 떨어짐



만든사람이 바보라서가 아니고, 분산 처리 및 Fault tolerance(장애 복구)를 위한 어쩔 수 없는 설계상의 선택

* 수십~수백대의 컴퓨터를 연결하여 연산을 하는 클러스터 컴퓨팅에서는
일부 컴퓨터가 고장나는 경우가 매우 흔하며, 시스템이 이에 대비하도록 설계되어 있다

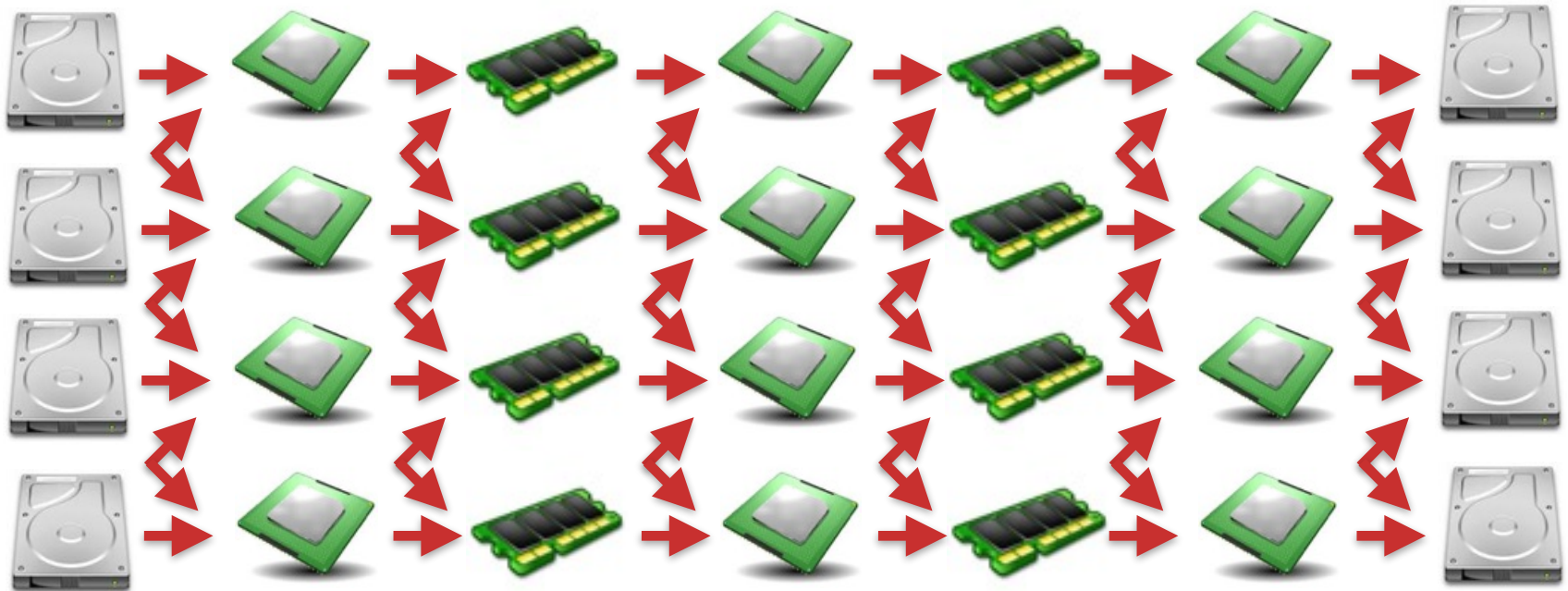
인메모리 컴퓨팅 엔진 (2)



다단계의 연산이나 반복 연산의 경우
중간 결과를 메모리에 저장하면
매번 디스크에 쓰는것 보다 훨씬 빠르다

(데이터가 전부 들어갈 정도의 메모리가 필요!)

인메모리 컴퓨팅 엔진 (2)

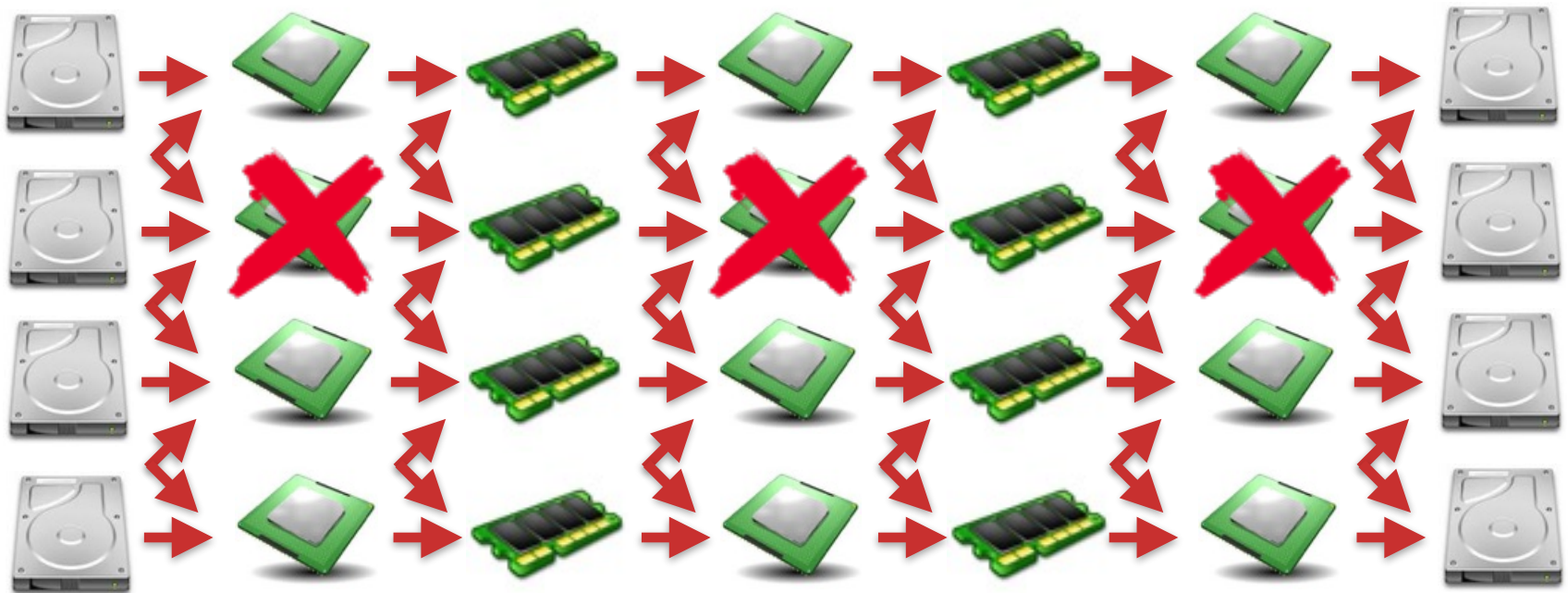


실제로는 데이터도 분산저장되어있고
연산도 분산처리를 한다



그럼 중간 복구지점이 없다면,
Fault tolerance(장애 복구)는 어떻게?

인메모리 컴퓨팅 엔진 (3)



중간에 컴퓨터 한대가 고장이나 오류가 난 경우,
누락된 데이터의 연산과정을 기록한 Lineage (계보)
를 이용해서, 해당 부분을 처음부터 다시 계산함

(게임 아님 주의)

뛰어난 성능과 편리한 인터페이스의 조합이

Spark의 핵심이라 할 수 있는

RDD (Resilient Distributed Dataset) !

(탄력있는 분산 데이터셋 - 분산 처리 및 장애복구가 가능하다는 뜻을 내포)

쉽고 편리한 인터페이스 (1)

- Scala, Python, Java를 지원 (Scala 권장)
- 기존의 Hadoop MapReduce에 비해 극적으로 간단해짐
- Spark Shell 을 제공하여 탐색적인 데이터 분석이 가능
- Apache Zeppelin같은 시각화 노트북 툴 활용 가능

쉽고 편리한 인터페이스 (2)

- Scala - Java와 호환되며, 강력한 문법으로 간결한 표현이 가능한 언어

```
public class WordCount {

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
    IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable,
    IntWritable> {
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<
        output, Reporter reporter) throws IOException {
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);


        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Word Count Example



```
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                  .map(word => (word, 1))
                  .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

수십 라인의 Java 코드를 단 3줄로 만들어 버림

데이터분석의 괴로움 해결

뛰어난 확장성

- **Spark SQL**: 코드 대신 SQL로 데이터를 분석
- **Spark Streaming**: 코드베이스를 많이 바꾸지 않고도 실시간 분석이 가능
- **MLLib**: 대용량 머신러닝
- **GraphX**: 대용량 그래프 분석
- **SparkR**: Spark기반으로 돌아가는 R
- **Apache Zeppelin**: Spark을 위한 노트북, 시각화 도구
- 그 밖의 수많은 확장 프로젝트

Apache Zeppelin 광고

- 대화형 Shell에서 더 발전하여, 코드 및 결과 기록, Visualization까지 책임져 주는 도구



Spark History

- 2009년: UC Berkeley AMPLab에서 시작됨 (Matei Zaharia)
- 2010년: 오픈소스화 됨
- 2012년: 0.5버전 릴리즈
- 2013년: Apache 프로젝트가 됨
- 2014년 2월: Apache Top Level 프로젝트가 됨
- 2014년 2월: 0.9버전 릴리즈, 안정성이 대폭 향상됨
- 2014년 5월: 1.0버전 릴리즈, 대부분의 문제가 해결됨
- 2014년 11월: Databricks가 Spark을 이용, 대용량 정렬 세계 신기록 세움
- 2015년: 1.2~1.5버전 4번의 릴리즈, 많은 개선이 됨, 누적 contributor 1000명 돌파
- 2016년 7월: 2.0버전 릴리즈, 앞으로의 방향성 제시



만든사람

이렇게 되기까지..

- 좋은 프로젝트를 향한 순수한 열정
- JIRA(이슈 관리 시스템)에서 항상 토론이 이루어지고, 창립자인 Matei Zaharia의 이름도 항상 볼 수 있음
- 성능향상을 노렸던 프로젝트 & 인터페이스를 개선한 프로젝트 - Spark은 둘. 다.
- 창립자들이 만든 회사인 Databricks는 Andreessen Horowitz 등으로부터 \$47M (약 500억 원) 이상을 투자받기도 함

Spark 관련 질문들



Q. Apache Spark이 하둡을 대체하게 될까요?

A: 네-니오

Spark이 하둡 (MapReduce, Hive)가 하는 일을 잘 할 수 있고,
Spark을 Hadoop과 혼합하지 않고 단독으로 설치하는 경우가 많은
것은 맞습니다.

또한 기존의 하둡 작업들을 Spark으로 옮기기에도 어렵지 않습니다.

단 Shuffling이 많이 일어나는 연산들은 (예: 수십TB의 데이터를 아
이디별로 그룹만들기) Spark에서는 잘 안되는 경우도 종종 있습니
다. 물론 이런 작업들은 하둡을 사용해도 굉장히 오래걸리거나 비효율
적인 경우가 많기는 합니다.



Q. 데이터 개발자입니다. Spark을 공부해야 할까요?

A: 네. 바로 시작하세요

Hadoop을 사용하시고 계시던, 처음 시도해보는 것이던 바로 시작해보시는것을 권해드립니다.

고성과와 편리한 인터페이스로
시간과 비용이 많이 아껴지고
더 많은 일들을 할 수 있습니다.

또 Spark은 지금도 점점 더 많이 쓰이고 있으므로
커리어 면에서도 지금 시작하는것은 좋다고 생각합니다.



**Q. 데이터 분석을 처음 해보려는 회사입니다.
Spark으로 해보는게 좋을까요?**

A: 네. 하지만 그 전에..

Google Analytics, Firebase Analytics, Facebook Analytics 같은 무료 분석 툴을 사용해 보세요.

어떤 것들을 할 수 있는지, 어떤 것들을 해야 할 지 감을 잡을 수 있을겁니다.

이런 툴들에서 부족함을 느끼면 그때부터 Spark을 활용해보면 좋을겁니다.



Q. 기존에 하둡으로 분석하고 있습니다.
Spark으로 옮겨야 할까요?

A: 저희는 그렇게 했습니다.

Spark과 Hadoop을 동시에 운영하면서,
새로 만드는 부분은 Spark으로 하고,
기존의 Hadoop MapReduce코드를
조금씩 Spark으로 옮기는 작업을 하였습니다.
약 6개월 정도만에 하둡 클러스터를 전부 내렸습니다.

Spark을 활용하면 일반적으로
2배~5배 정도의 성능 향상과 함께
코드가 간단해지므로 생산성이 크게 향상됩니다.
분명히 시간과 노력을 투자할 가치가 있을겁니다.



Q. 현재 사용하고 있는 클러스터에 메모리가 부족합니다.
그래도 Spark을 쓸 수 있나요?

A: 네 가능합니다.

반복 작업이 아닌 단순한 ETL작업 같은 경우는 메모리에 올려야 할 필요가 없기에, 데이터에 비해 메모리가 부족하더라도 문제가 없습니다. 물론 이 경우는 하둡에 비해 큰 성능 향상은 없습니다. (잘하면 x2 정도)

반면 Grouping같은 연산 (Shuffling이 많은) 을 하려면 메모리가 넉넉해야 합니다. 하둡과는 다르게 메모리가 부족한 경우 작업이 끝나지 않는 경우도 있어, 주의가 필요합니다.

Bonus:

Apache Spark의 미래

Apache Spark의 미래 ⁽¹⁾

- Spark SQL이 점점 좋아지고, 중요해짐
- 더 효율적으로 데이터를 처리하기 위한 Spark 2.0의 DataSet API
- 자유도 높은 데이터 처리를 위한 현재의 RDD API 도 함께 공존
- Tensorflow 등 새롭게 조명받는 프로젝트의 고성능 엔진 역할

Apache Spark의 미래 (2)

- 기존의 SQL기반 분석, BI툴들과 정면 경쟁
- 점점 더 많이 쓰이고, 널리 쓰일 것
- 빅데이터 분석 엔진에서 General 컴퓨팅 엔진으로 확산

Tech planet 2016

Thank you