



Addressing the Shortcomings of Hadoop

하둡 vs 헤이즐캐스트

본 자료는 헤이즐캐스트 기술파트너사
Numatica Corporation
Jacek Kruszelnicki씨의 웨비나 발표 자료에 기초
하여 제작하였습니다.

Big Data, Simple and Fast

“Any intelligent fool can make things bigger and more complex... It takes a touch of genius - and **a lot of courage** to move in the opposite direction.”

- E.F. Schumacher

- Hadoop이 빅데이터 플랫폼 관점에서 범용적 기준도 아니며 비용적인 측면에서 만족하지 못할 뿐 아니라 이름 그대로 크게 접근해야 한다.
- 기술적인 선결 요건은 유연하고 빠르게 빅데이터 접근이 필요.
- 하둡을 대체할 수 있는 솔루션?
- 인메모리 컴퓨팅은 왜 Big/Fast Data에 적합한가?
- 헤이즐캐스트는 위의 선결 요구 조건에 적합한가?
- 위 사항에 충족하는 시스템 아키텍처에 포커스를 맞춘 코딩 작업과 프로그램을 쉽게 지원할 수 있어야 한다.



“논외로 두고 가는 주제들 ”

- 빅데이터에 관한 심도있는 주제
- 하둡과 그 외의 기술에 대한 주제
- 빅데이터 시장에 관한 전반적인 고찰
- 빅데이터 분석과 시장 사업성에 대한 인사이트



즉각적인 운영 정보 대응:

- 비즈니스 흐름을 분석하고 대외적인 요소에 즉각적인 대응 체계 요구 증가
- 즉각적인 대응 가능한 IT 인프라 구축 필요
- 실시간 데이터 처리 과정의 중요성 부각 및 낮은 대응시에 따르는 비즈니스 가치 손실에 대한 인식 부족.

실시간 빅- 메가 패스트 데이터 분석 예제:

- 변동이 심한 상품 가격 테이블 / Dynamic pricing (e-commerce)
- 인기 상품 매매 / High-frequency trading
- 네트워크 보안 위협 / Network security threats
- 신용카드 정보 보안 / Credit card fraud prevention
- 제조시설 실시간 생산 정보 수집 / Factory floor data collection, RFID
- 모바일 환경과 M2M 어플리케이션 / Mobile infrastructure, machine to machine (M2M) applications
- 예약 실행 혹은 위치기반 어플리케이션 / Prescriptive or Location-based applications
- 실시간 대시보드, 경고 및 보고체계 / Real-time dashboards, alerts, and reports

빅데이터 요소인 3V 재 설정



Common definition:

데이터셋은 너무 크고 복잡해서 기존의 DB 툴과 프로그램으로는 처리할 수 없다. (다른 말로 'MS 엑셀'에서 구동할 수 없다)

Variety

- 정형 / 비정형 데이터
- 정적 혹은 동적 데이터
- 복잡 다양한 데이터 유형

Velocity

- 초당 수백만 데이터 이동 발생
- 실시간 이벤트를 반영하기 위한 빈도 수 높은 데이터 분석 요구
- 수시로 변화하는 정적 / 동적 데이터
- 방대한 데이터속에서 가치 발굴의 어려움
- 오프라인 연산이 필요 (기계학습)

Volume

- 구글이나 페이스북 같은 방대한 데이터 처리를 요구하는 회사가 얼마나 되나?
- 평균 일반 기업 데이터는 < 100 TB 임.

동적 데이터 / Data In-Motion:

- 수시로 값이 변환하면서 이동하는 데이터 (이벤트)
- 복수의 데이터 소스와 유형
- 시스템이 작동하는 동안 데이터도 동시 처리 요구 증가
- 데이터 이벤트 3V 보다는 비즈니스 가치에 초점을 뒀야함

정적 데이터

Data At-Rest:

- 데이터가 이미 정적이며 변환될 때 알림 기능.
- 복수의 유형
- 데이터 값을 재설정 혹은 재 처리해야함

Software stacks to support “In-Motion” and “At-Rest” Big Data are needed.

"Big Elephant in the room..."



The Hadoop Stack:

- Currently the large scale data analysis tool of choice
- *De facto* standard, almost synonymous with Big Data
- "Nobody ever got fired for using Hadoop"

Problems:

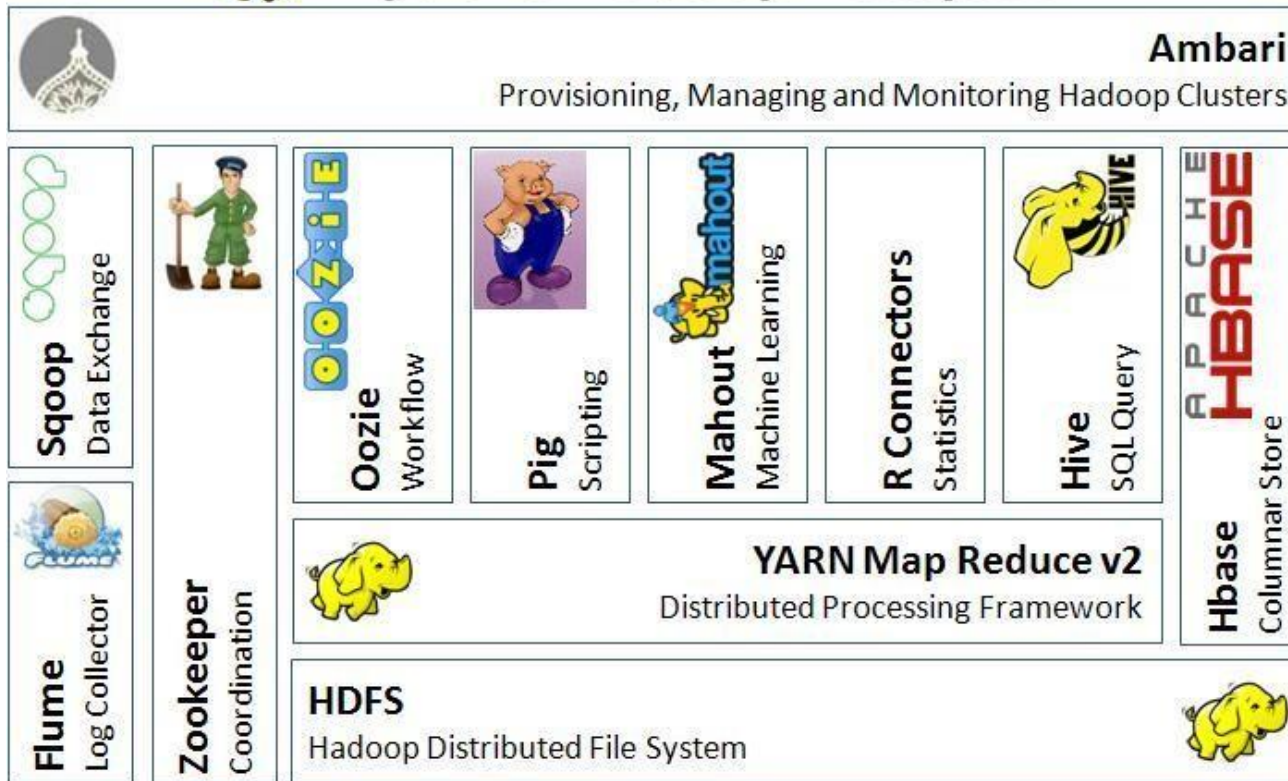
- 👎 Very complex/expensive to deploy and run -> high TCO
- 👎 MapReduce slow, **batch-oriented**, "intrusive"
- 👎 No stream processing
- 👎 No support for in-memory processing
- 👎 Closely coupled with HDFS (third-party solutions of varying quality)
- 👎 SQL-on-Hadoop limited and slow

<http://hortonworks.com/blog/install-hadoop-windows-hortonworks-data-platform-2-0/>
http://www.chrisstucchio.com/blog/2013/hadoop_hatred.html/

Hadoop... and the kitchen sink



Apache Hadoop Ecosystem



Not shown:

JobTracker.
TaskTracker,...
Avro (Serialization)
Chukwa
(logs, incremental)
EMR
BigTop
Spark
Impala (vs Hive)

19 shown, up to 24

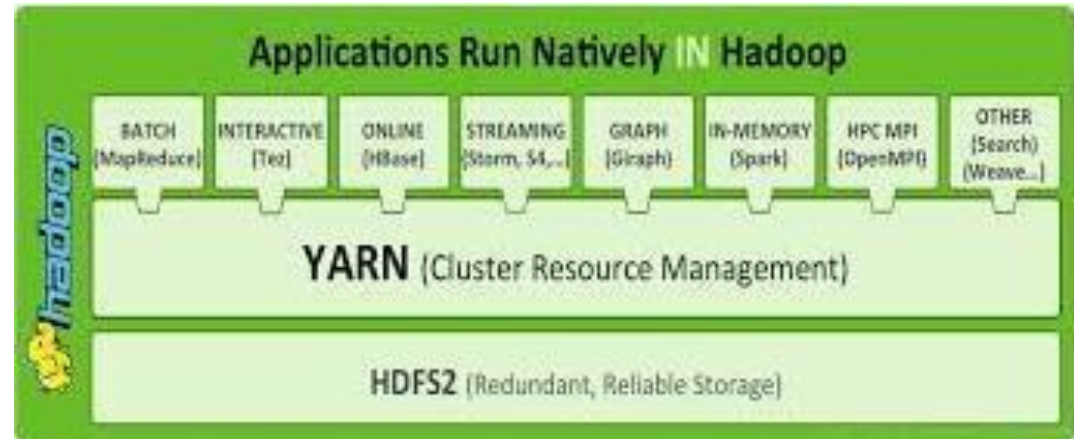
분산 컴퓨팅의 첫 번째 규칙에서 벗어난 아키텍처
DO NOT DISTRIBUTE (unnecessarily).

Hadoop 2.0 – 높은 총 소유 비용 (TCO) 문제 상존



성능 향상:

- 👍 YARN, MapReduce 분리됨
- 👍 Removed Name Node/ Job Tracker as SPOF?

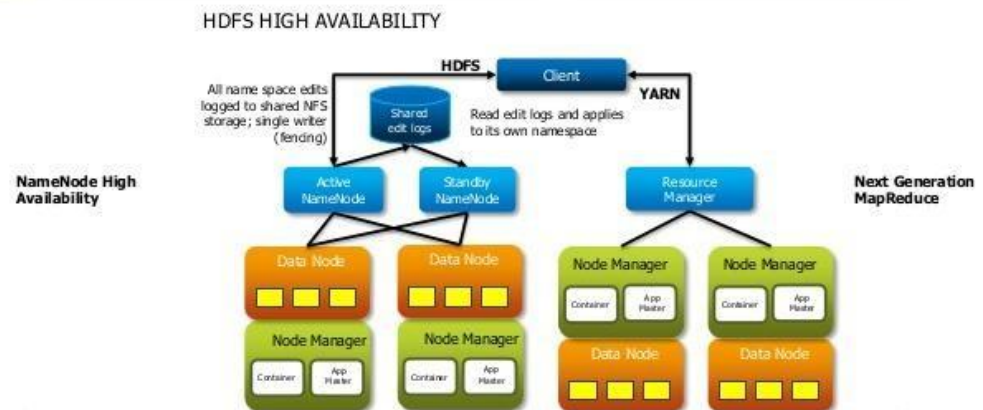


Unchanged:

- 👎 더 복잡해진 아키텍처
- 👎 Low-level 아키텍처
- 👎 Master/Slave 구조 유지

Hadoop 2.0 Cluster Architecture - HA

edureka!

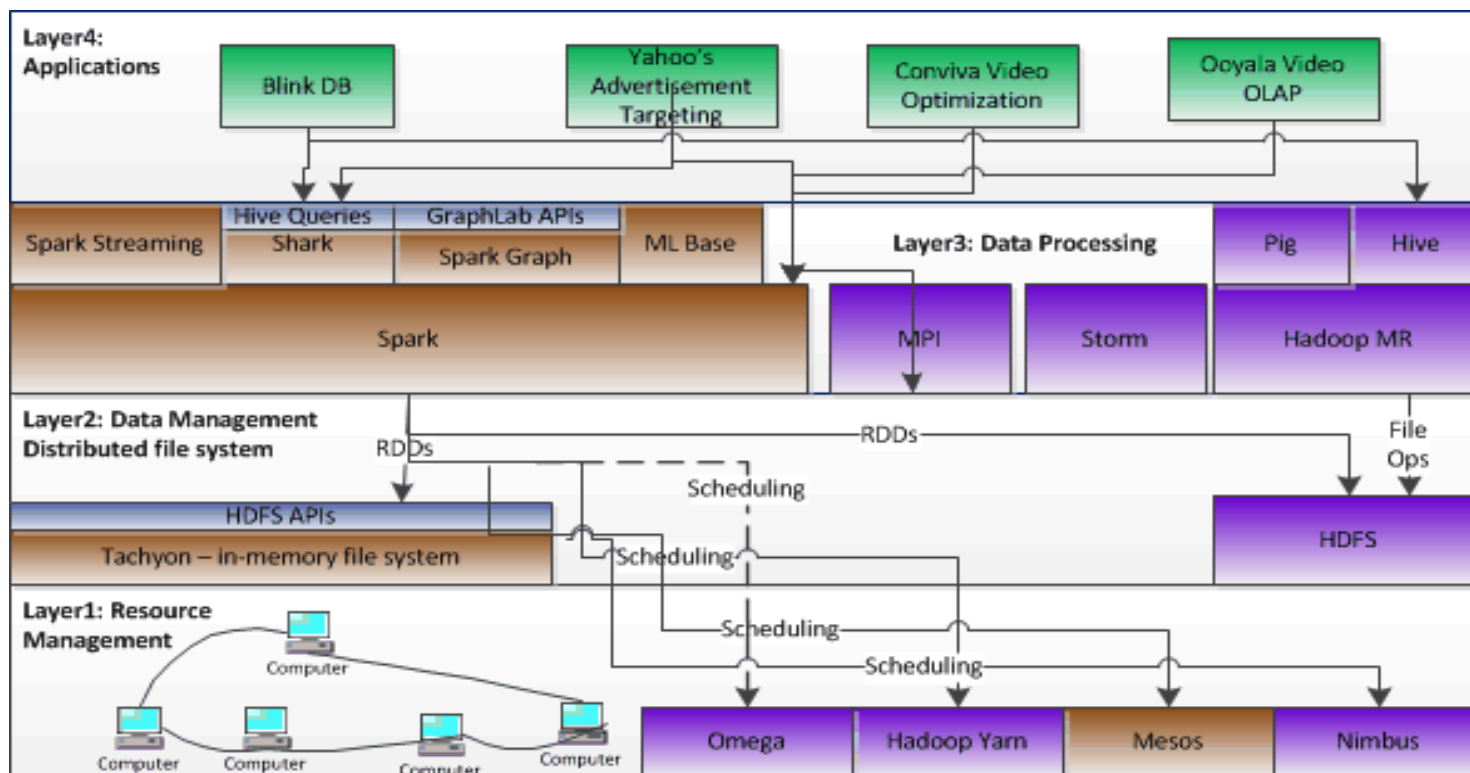


<http://hadoop.apache.org/docs/stable2/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailabilityWithNFS.html>

Hadoop 2.0 – 높은 총 소유 비용 (TCO) 문제 상존



Hadoop-based stack example from the Web. 더욱 더 복잡해진 구조적 문제



Berkeley Big-data Analytics Stack (BDAS)

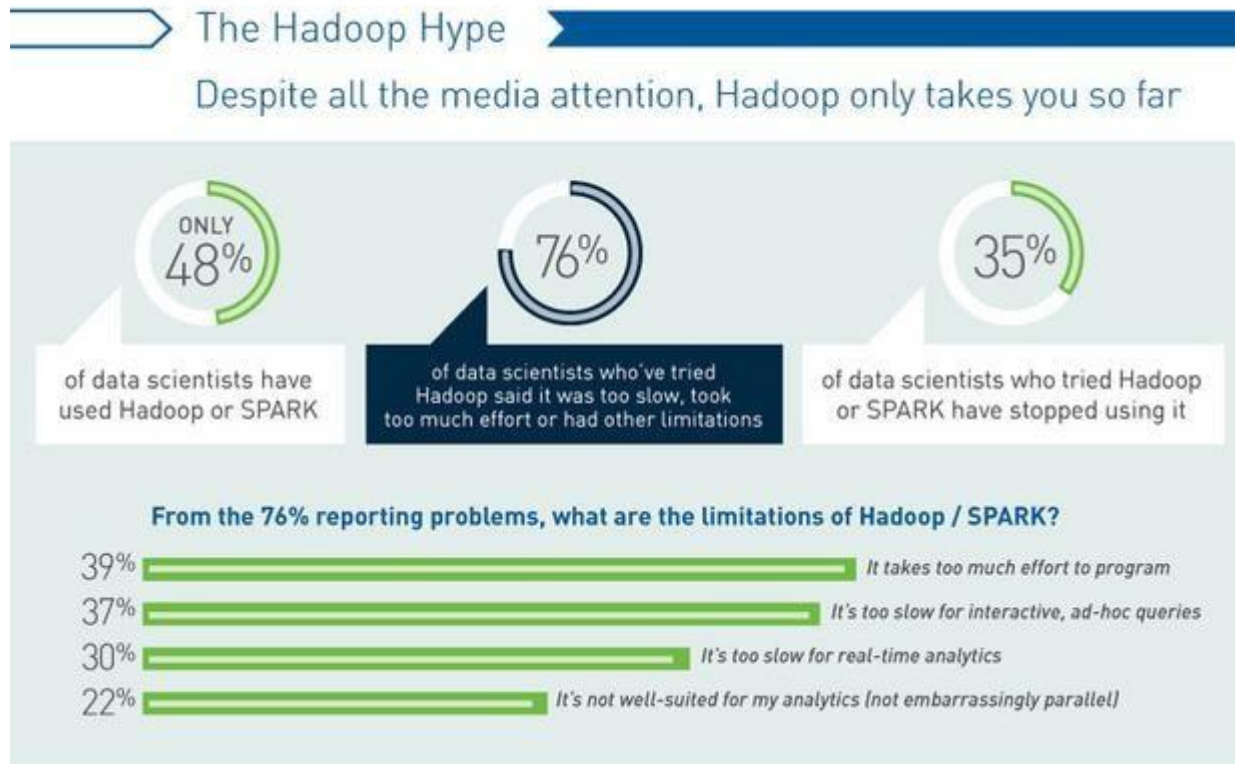
Legend

- Part of BDAS
- Optional frameworks with which BDAS interacts
- Applications built over BDAS

Hadoop의 환상과 현실적 문제



Hadoop이 빅데이터 기본이라며 열광했으나 오히려 빅데이터 처리에 방해?



Data Scientists: 76 %는 너무 느리고 프로그래밍에 너무 많은 투자를 해야한다고 응답함.

<http://www.cio.com/article/2449814/big-data/data-scientists-frustrated-by-data-variety-find-hadoop-limiting.html>

MapReduce를 쓰지 않기로 결정한 구글 / MR is on its way out at Google.

<http://www.datacenterknowledge.com/archives/2014/06/25/google-dumps-mapreduce-favor-new-hyper-scale-analytics-system/>

하둡 대체 기술 / Some Hadoop Alternatives



Hadoop “extensions”

- Spark
- Shark (Spark on Hive)
- Storm
- Kafka

하둡 모듈 기반: HDFS, YARN, Pig, Hive, HBase, JobTracker, TaskTracker,...

MPP DBs

- Teradata
- Vertica
- Greenplum

상용 솔루션, 복잡한 아키텍처, 높은 비용, 쿼리 프로그래밍 완성도 없음.

NoSQL DBs

Column: Cassandra, Hbase, etc.

Document/K-V: MongoDB, CouchDB, Riak, etc.

In-Memory: VoltDB, etc.

No ACID/Referential integrity,

No triggers, foreign keys

Mongo/Couch -> JSON-centered, Cassandra -complex

Query language proprietary, subset of SQL

VoltDB – precoded stored procs, no ad-hoc

Not Turing complete

In-Memory DataGrids

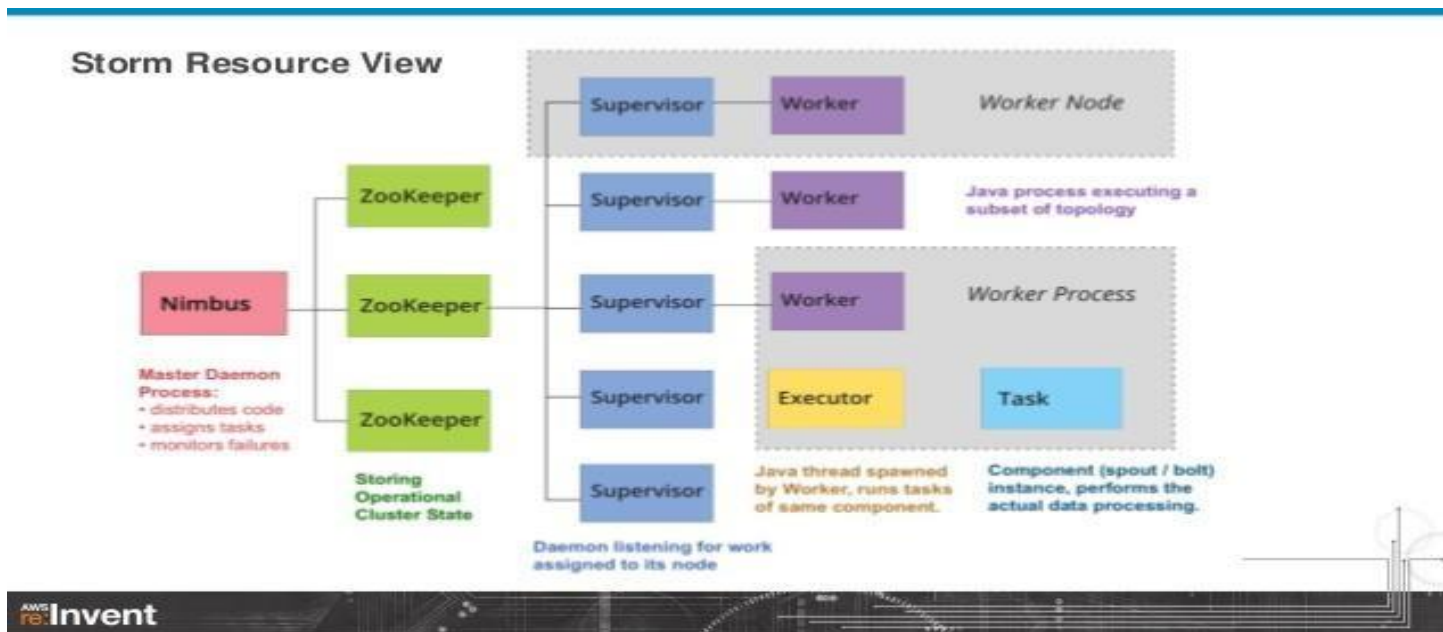
- Coherence (commercial)
- Hazelcast (OSS)
- Terracotta (commercial)
- Gemfire (commercial)
- GridGain (OSS/commercial)
- Gigaspaces XAP (OSS)

Big + Fast 대체 솔루션



Storm/Kafka

- Intrusive (introduces exotic abstractions): Streams, Spouts, Bolts, Tasks, Workers, Stream Groups, Topologies
- Low-level: Shell scripting, 곳곳에 산재한 Clojure
- Complex Admin: ZooKeeper 의존 100%, Nimbus is a SPOF (Single Point of Failure)



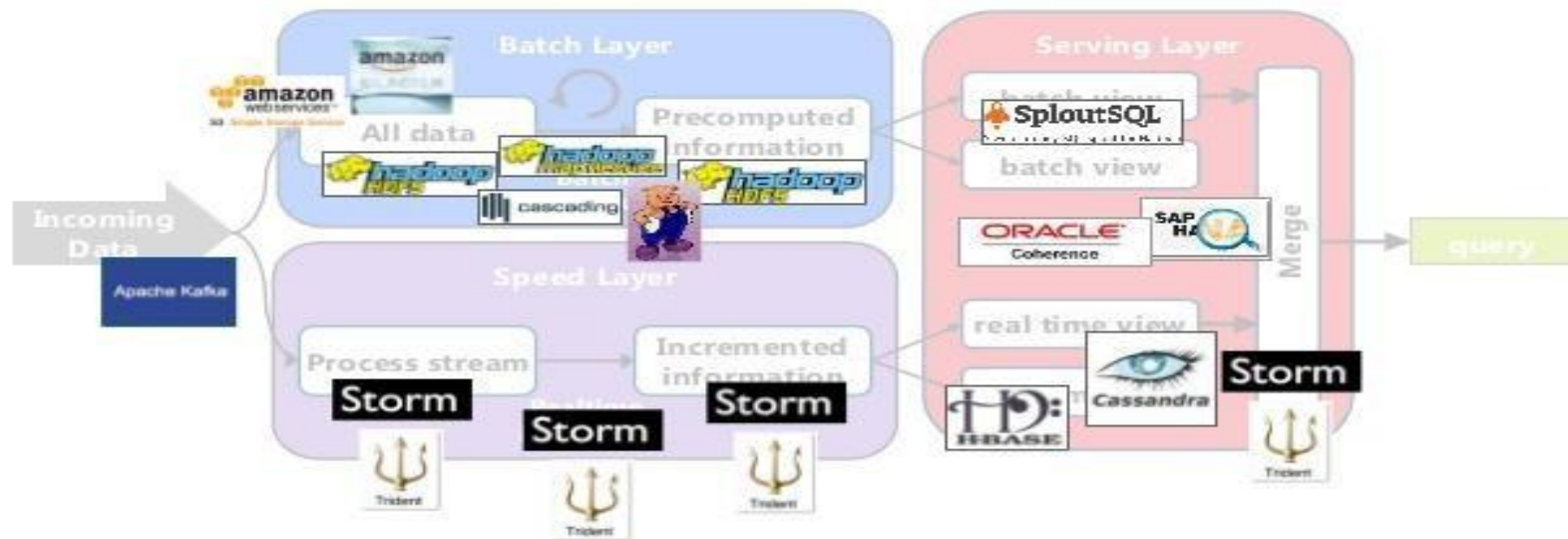
<https://news.ycombinator.com/item?id=8416455>

Big + Fast 대체 솔루션



Lambda Architecture. 너무 많은 연동 솔루션

Lambda Architecture
Big Data and Fast Data combined
one possible product/framework mapping



43

2013 © Trivadis
Kafka and Storm - event processing in realtime
22.10.2013

trivadis
makes IT easier.



성공적인 Big + Fast Data stack 구축을 위한 요구 조건:

- 👍 간결하고 정밀한 아키텍처 [낮은 총소유 비용 Total Cost of Ownership (TCO)]
- 👍 초고속 트랜잭셔널한 데이터 처리 [반응속도 <1s, “fresh” 혹은 실시간 데이터]
- 👍 대용량 처리, 전체적으로 성능 향상 과 확장성을 가진 아키텍처
- 👍 대용량 데이터 처리 Stream/Complex Event Processing
- 👍 SQL 같이 querying, ACID, TxS(including XA)
- 👍 분산 실행 프레임워크 Distributed Execution Framework
- 👍 고가용성 및 낮은 시스템 장애 효율 / High Availability and Fault Tolerance.
- 👍 분산환경, 탈 중앙집중적, 탄력적인 클러스터 관리 구조.

In-Memory Grids for Big And Fast Data



- ❑ Databases (RDBMS, NoSQL) are not enough
 - SQL or SQL-like languages are not Turing complete
 - Object-oriented/functional/parallel programming abstractions needed

- ❑ 진일보한 기술 요소 (6x sequential, 100K x random):
 - RAM – 나노 초 (ns) 단위 속도의 디스크
 - DISK - 마이크로 초 (ms) 단위 속도 테이프 저장매체

- ❑ In-memory data 는 안정적이지 않고 휘발성 강함?
 - In-memory data는 이미 안정적임.
 - No need to achieve archival durability.
 - 대부분의 Big Data는 이미 다른 곳에 저장되어 있는 경우 많음.
 - Stream data는 장기간 저장할 필요 없거나 제한적 시간동안만 저장함.

Why Hazelcast?



- 기술적으로 가장 앞섰고 간결한 아키텍처로 분산 인메모리 컴퓨팅 가능.
- Minimalism as design aesthetic:
 - Non-intrusive,
 - No dependencies
 - 3.7MB single jar library.
- Implements Java APIs (Map, List, Set, Queue, Lock) in a distributed manner
- Distributed Execution Framework (extension of Java's Executor Service)
 - Distributed Queries (SQL/Predicate), Data affinity (execution on specific node execution)
- Peer-peer architecture, no single point of failure
- Elastic cluster management Java API included
 - Auto-discovery of nodes and re-balancing
- Apache License 2, commercial extensions + support

Hazelcast vs Hadoop



Java8 + Hazelcast (single 3.7 MB jar)






Pluggable persistence (HDFS, MapR, RDBMS)
MapReduce
Data manipulation & querying
In-memory parallel processing
Stream processing
Messaging
Scalable and Elastic
Cluster Management
Elastic, simple (automatic cluster re-balancing)

Java + Hadoop

HDFS
MapReduce
Hive (not OLTP)
Spark
Storm
Kafka
ZooKeeper
ZooKeeper, YARN, Mesos N/A

Considerations



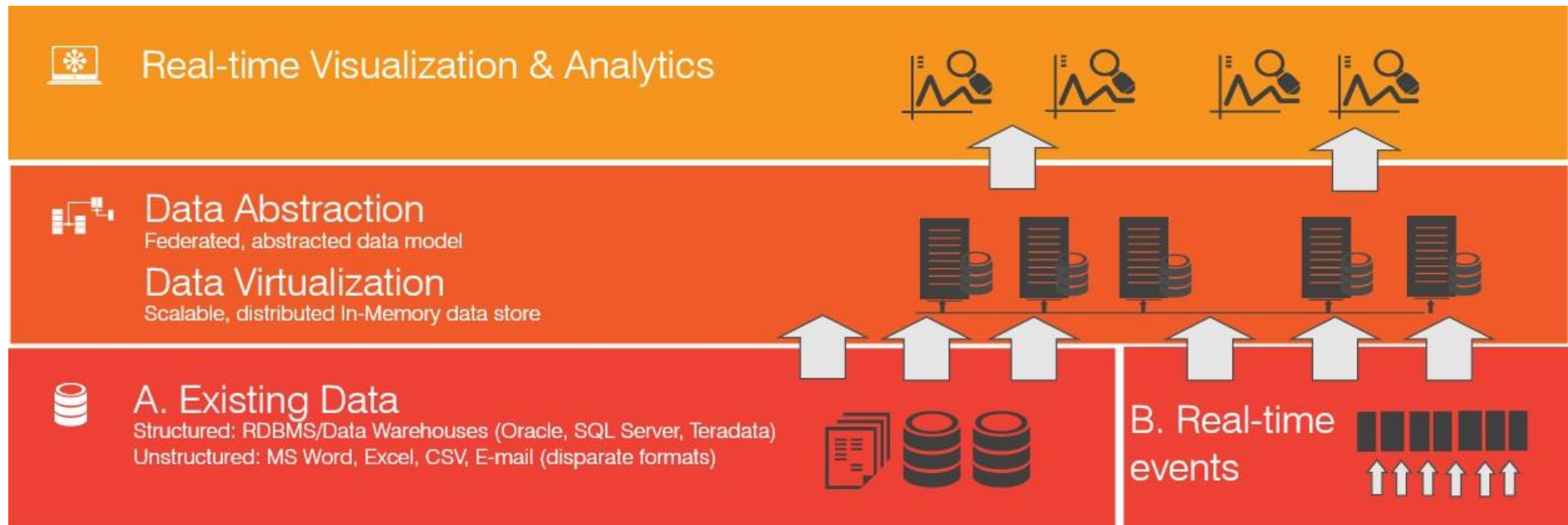
-  RAM currently maxes out at ~ 640GB/server
-  RAM still more expensive than SSDs and HDDs
-  Garbage collection
-  Cost and capacity limitations will disappear over time
-  Off-heap memory and specialized JVMs (Azul, etc.)

From Technologies to Platform



In-Memory Data/Compute Grids 고려할 점:

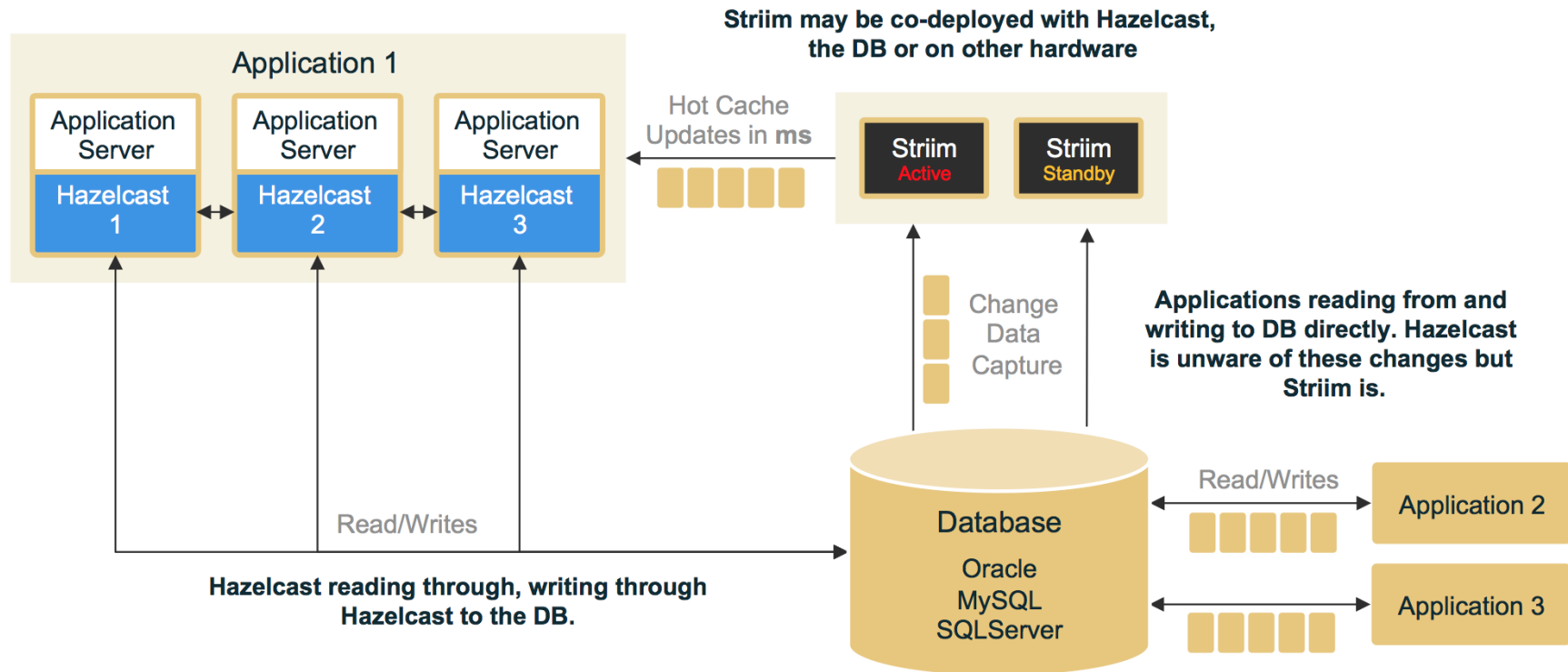
- 백업 저장소: RDBMS, NoSQL, HDFS GlusterFs, XFS
- Enterprise Message Store(s) – 헤이즐캐스트 HD 메모리 기능으로 해결
- Multi-Source Data Harvesting, Ingestion, Transformation – Striim 플러그인으로 해결
- Data Abstraction, Modeling, Querying, Visualization – 헤이즐캐스트 JET으로 해결



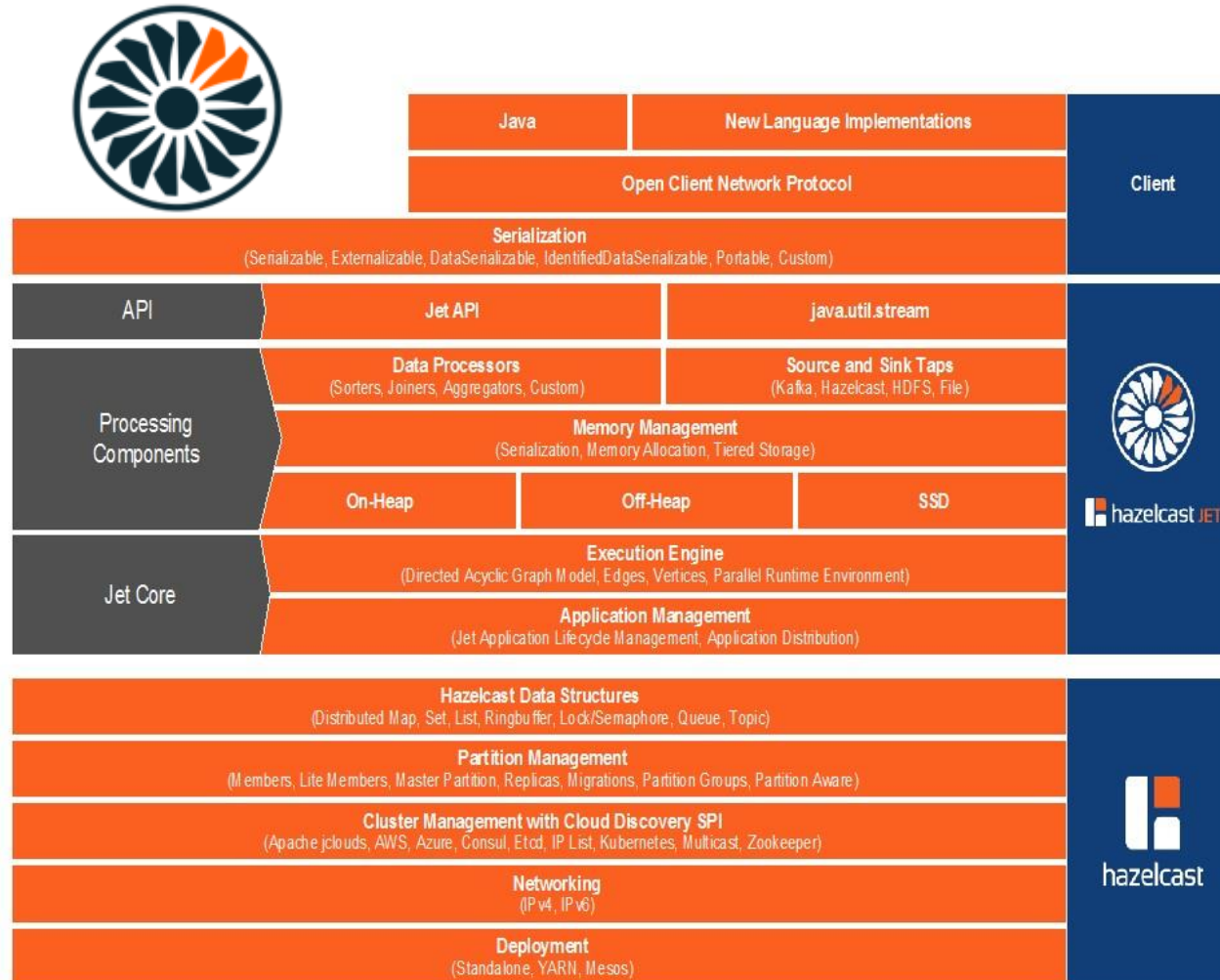
Hazelcast Striim Hot Cache



CDC (change Data Capture)기능



Hazelcast JET

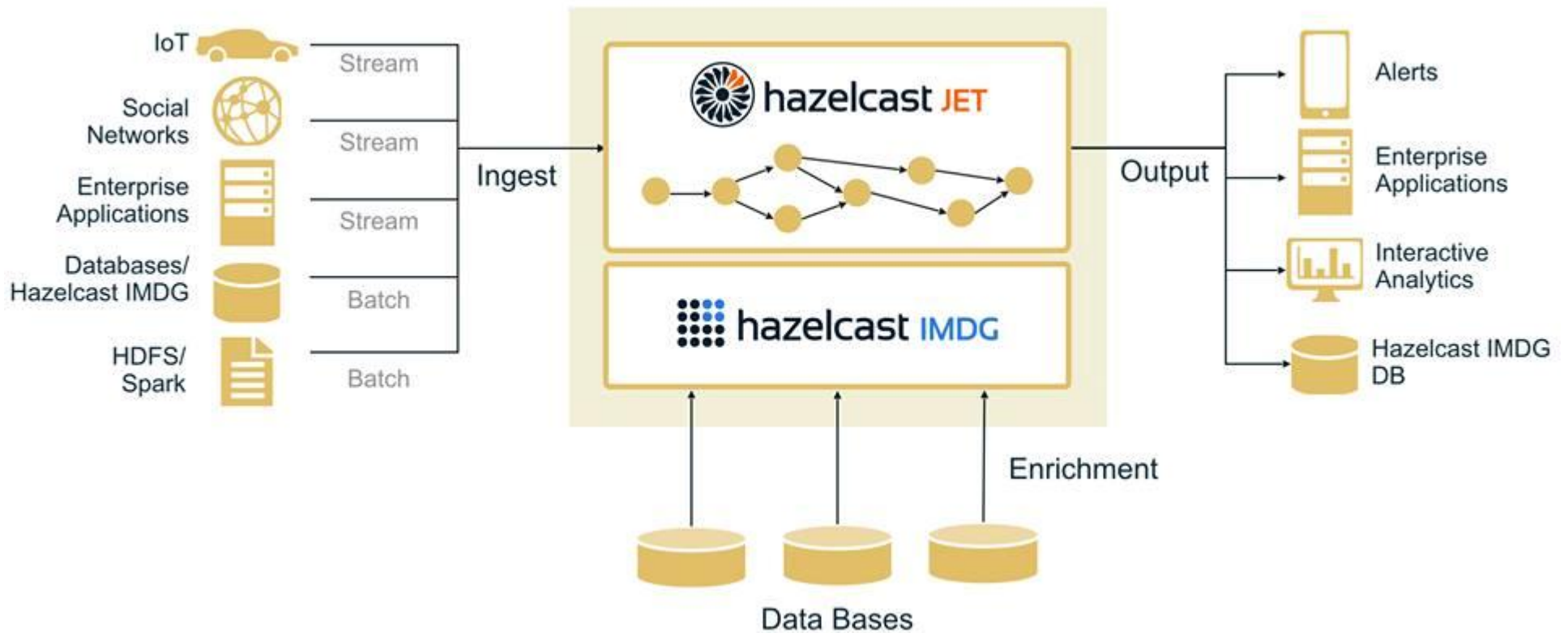


초고속 빅데이터 플랫폼

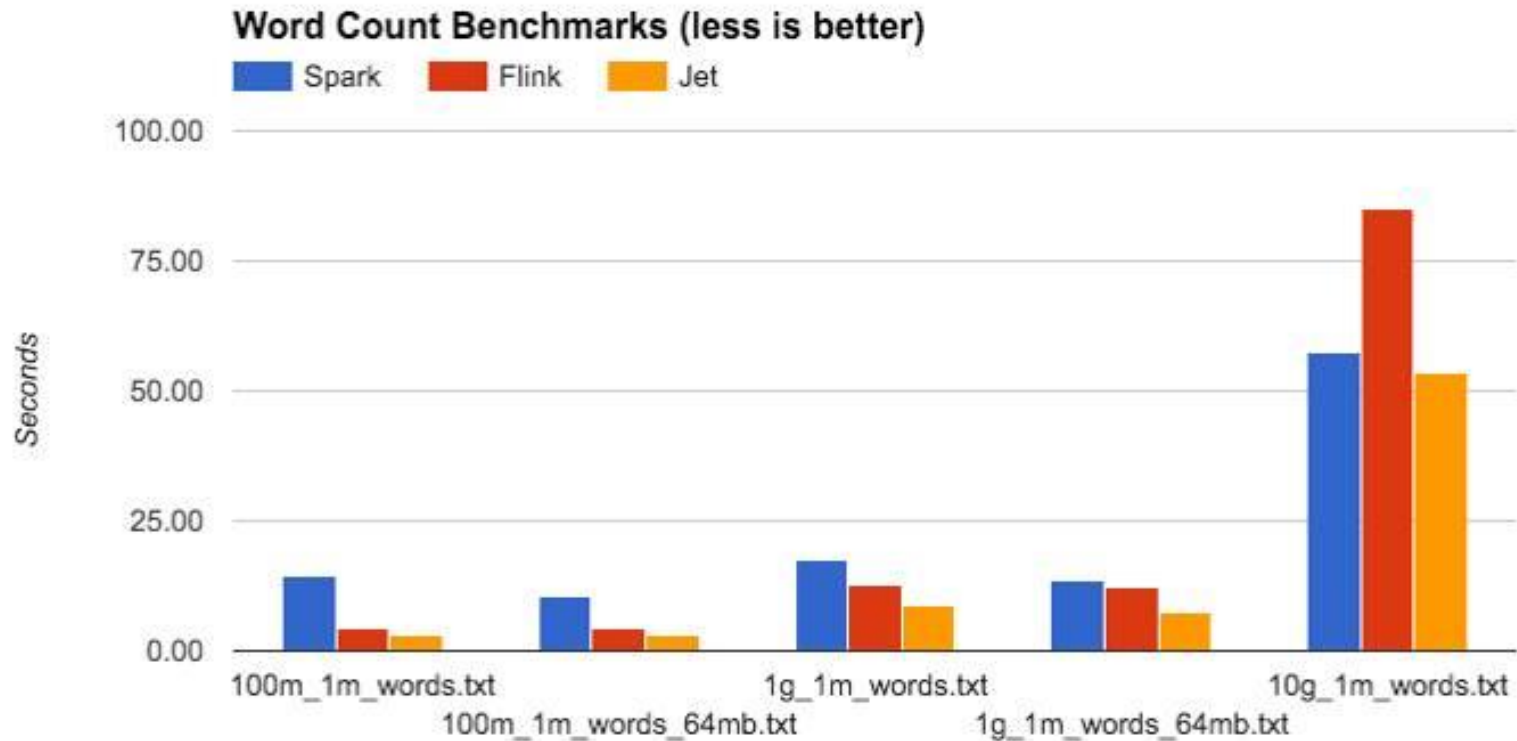
간결한 아키텍처
구축시간 감소
헤이즐캐스트 연동

Hadoop 대비 20배 빠름

Hazelcast JET



Hazelcast JET 벤치마크 시험 결과





**“Intellectuals solve problems.
Geniuses prevent them.”**

-- Albert Einstein

End of Document