Challenges for Data Driven Systems

Eiko Yoneki

University of Cambridge Computer Laboratory

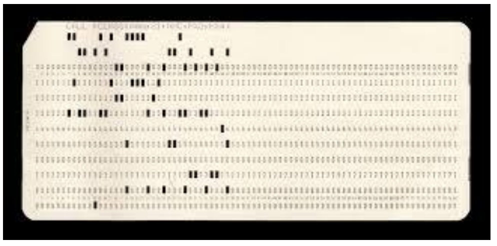
Quick History of Data Management

- 4000 B C Manual recording
- From tablets to papyrus…to paper

A. Payberah'2014

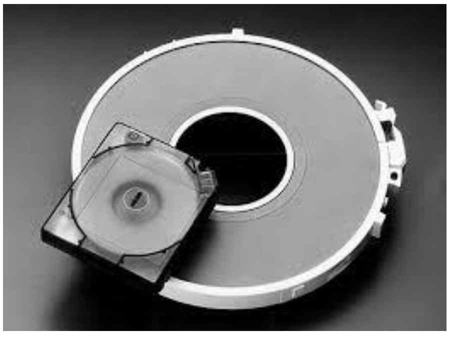
1800's - 1940's

- Punched cards (no fault-tolerance)
- Binary data
- 1911: IBM appeared



1940's - 1970's

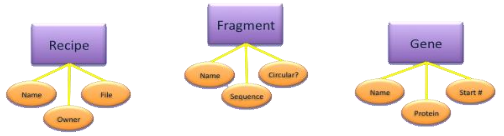
- Magnetic tapes
- Batch transaction processing
- Hierarchical DBMS
- Network DBMS



1980's

- Relational DBMS (tables) and SQL
- ACID (Atomicity Consistency Isolation Durability)
- Client-server computing
- Parallel processing

FRAG	NAME	SEQUENCE	CIRCULAR
F1	r1.p1	ATCG...	TRUE
F2	r1.p2	ATCG...	TRUE
F3	r2.f1	ATCG...	FALSE



RCP	NAME	FILE	OWNER
R1	r1	r1.cr	scooter
R2	r2	r2.cr	ckw

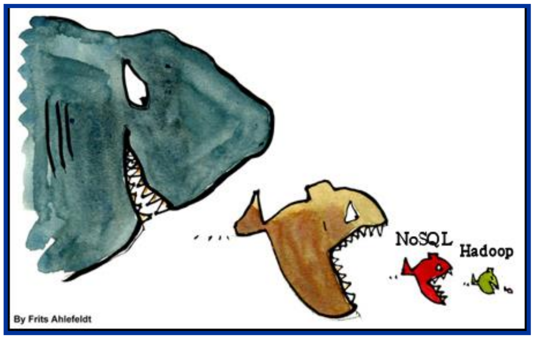
ID	NAME	PROTEIN	START #
G1	AMP	MAKK...	-5
G2	TET	MYAK...	-10
G3	NGF	MYAK...	-1

1990's - 2000's

- The Internet...

2010's

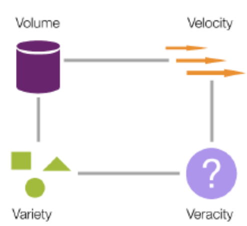
- NoSQL: BASE instead of ACID
  Basic **A**vailability, **S**oft-state, **E**ventual consistency
- Big Data is emerging!



2010s: Big Data

- Why Big Data now?
  - Increase of Storage Capacity
  - Increase of Processing Capacity
  - Availability of Data

  - Hardware and software technologies can manage ocean of data



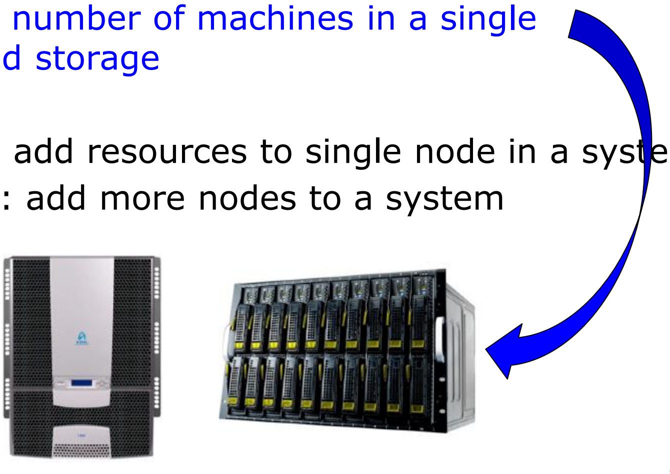
up to 2003 5 exabytes
→ 2012 2.7 zettabytes (500 x more)
→ 2015 ~8 zettabytes (3 x more than 2012)

Examples of Big Data

- Facebook:
  - 300 PB data warehouse (600TB/day)
  - 1 billion users
- Twitter Firehose:
  - 500 million tweet/day
- CERN
  - 15 PB/year - Stored in RDB
- Google:
  - 40000 search queries/second
- ebay
  - 9PB of user data+ >50 TB/day
- Amazon web services
  - Estimated ~450000 servers for AWS
  - S3 450B objects, peak 290K request/sec
- JPMorganChase
  - 150PB on 50K+ servers with 15K apps running

Scale-Up vs. Scale-Out

- Popular solution for big data processing → to scale and build on distribution and combine theoretically unlimited number of machines in a single distributed storage
- Scale up: add resources to single node in a system
- Scale out: add more nodes to a system



Challenges

- Distribute and shard parts over machines
  - Still fast traversal and read to keep related data together
  - Scale out instead scale up
- Avoid naïve hashing for sharding
  - Do not depend on the number of node
  - But difficult add/remove nodes
  - Trade off – data locality, consistency, availability, read/write/search speed, latency etc.
- Analytics requires both real time and post fact analytics – and incremental operation

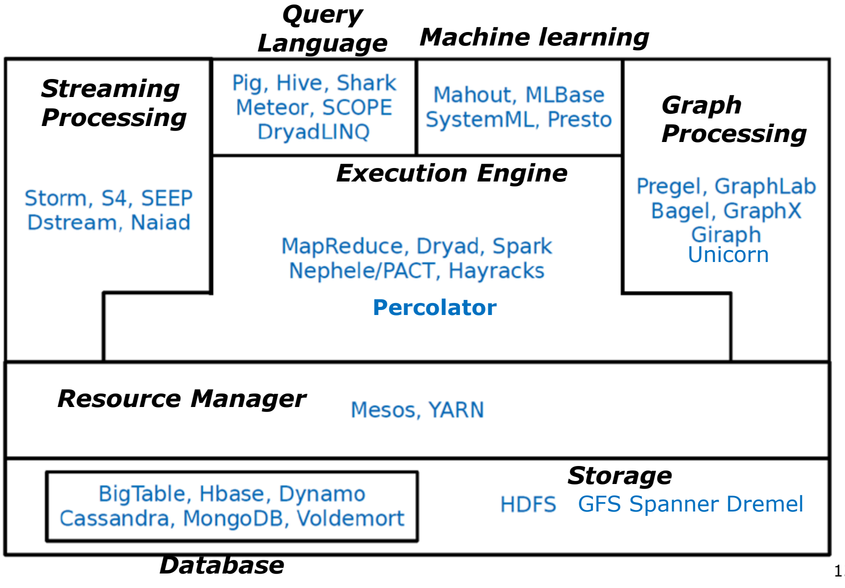
Big Data: Technologies

- Distributed infrastructure
  - Cloud (e.g. Infrastructure as a service, Amazon EC2, Google App Engine, Elastic, Azure)

    cf. Multi-core (parallel computing)
- Storage
  - Distributed storage (e.g. Amazon S3, Hadoop Distributed File System (HDFS), Google File System (GFS))
- Data model/indexing
  - High-performance schema-free database (e.g. NoSQL DB - Redis, BigTable, Hbase, Neo4J)
- Programming model
  - Distributed processing (e.g. MapReduce)

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Big Data Analytics Stack



UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Distributed Infrastructure

- Computing + Storage transparently
  - Cloud computing, Web 2.0
  - Scalability and fault tolerance
- Distributed servers
  - Amazon EC2, Google App Engine, Elastic, Azure
    - System? OS, customisations
    - Sizing? RAM/CPU based on tiered model
    - Storage? Quantity, type
- Distributed storage
  - Amazon S3
  - Hadoop Distributed File System (HDFS)
  - Google File System (GFS), BigTable…

Data Model/Indexing

- Support large data
- Fast and flexible access to data
- Operate on distributed infrastructure

- Is SQL Database sufficient?

NoSQL (Schema Free) Database

- NoSQL database
  - Operate on distributed infrastructure
  - Based on key-value pairs (no predefined schema)
  - Fast and flexible

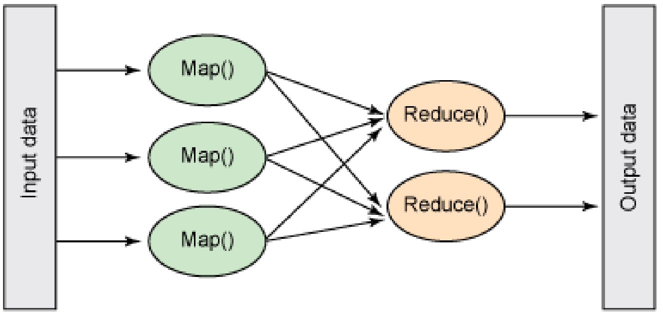
- Pros: Scalable and fast
- Cons: Fewer consistency/concurrency guarantees and weaker queries support

- Implementations
  - MongoDB, CouchDB, Cassandra, Redis, BigTable, Hibase …

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

MapReduce Programming

- Target problem needs to be parallelisable
- Split into a set of smaller code (map)
- Next small piece of code executed in parallel
- Results from map operation get synthesised into a result of original problem (reduce)



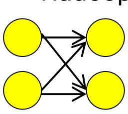
17

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Data Flow Programming

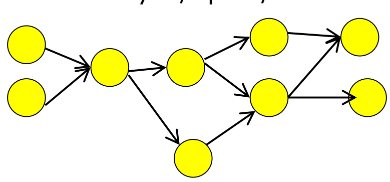
- Non standard programming models
- Data (flow) parallel programming
  - e.g. MapReduce, Dryad/LINQ, NAIAD, Spark

MapReduce:
Hadoop



Two-Stage fixed dataflow

DAG (Directed Acyclic Graph)
based: Dryad/Spark/Tez



More flexible dataflow model

18

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Easy Cases

- Sorting
  - Google 1 trillion items (1PB) sorted in 6 Hours
- Searching
  - Hashing and distributed search

→ Random split of data to feed M/R operation

- BUT Not all algorithms are parallelisable

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Streaming Data

- Departure from traditional static web pages
- New time-sensitive data is generated continuously
- Rich connections between entities

- Challenges:
  - High rate of updates
  - Continuous data mining - Incremental data processing
  - Data consistency

Techniques for Analysis

- Applying these techniques: larger and more diverse datasets can be used to generate more numerous and insightful results than smaller, less diverse ones

- Classification
- Cluster analysis
- Crowd sourcing
- Data fusion/integration
- Data mining
- Ensemble learning
- Genetic algorithms
- Machine learning
- NLP
- Neural networks
- Network analysis
- Optimisation
- Pattern recognition
- Predictive modelling
- Regression
- Sentiment analysis
- Signal processing
- Spatial analysis
- Statistics
- Supervised learning
- Simulation
- Time series analysis
- Unsupervised learning
- Visualisation

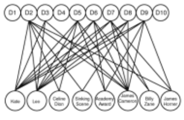
Do we need new types of algorithms?

- Cannot always store all data
  - Online/streaming algorithms
    - Have we seen x before?
    - Rolling average of previous K items
  - Incremental updating
- Memory vs. disk becomes critical
  - Algorithms with limited passes
- N^2 is impossible and fast data processing
  - Approximate algorithms, sampling
- Iterative operation (e.g. machine learning)
- Data has different relations to other data
  - Algorithms for high-dimensional data (efficient multidimensional indexing)

Typical Operation with Big Data

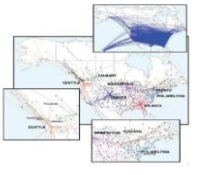
- Scalable clustering for parallel execution
- Smart sampling of data
- Find similar items → efficient multidimensional indexing
- Incremental updating of models → support streaming
- Distributed linear algebra → dealing with large sparse matrices
- Plus usual data mining, machine learning and statistics
  - Supervised (e.g. classification, regression)
  - Non-supervised (e.g. clustering..)

How about Graph (Network) Data?

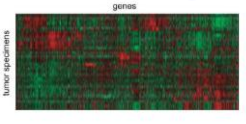


Bipartite graph of phrases in documents

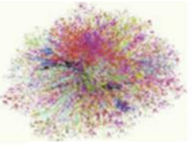
Brain Networks: 100B neurons(700T links) requires 100s GB memory



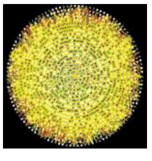
Airline Graphs



Gene expression data



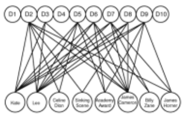
Web 1.4B pages(6.6B links)



Protein Interactions [genomebiology.com]

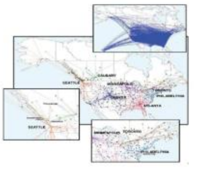
Social media data

How about Graph Data?

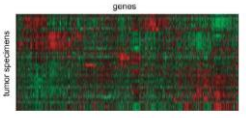


Bipartite graph of phrases in documents

Brain Networks: 100B neurons(700T links) requires 100s GB memory

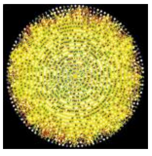


Airline Graphs



Gene expression data

Web 1.4B pages(6.6B links)



Protein Interactions [genomebiology.com]

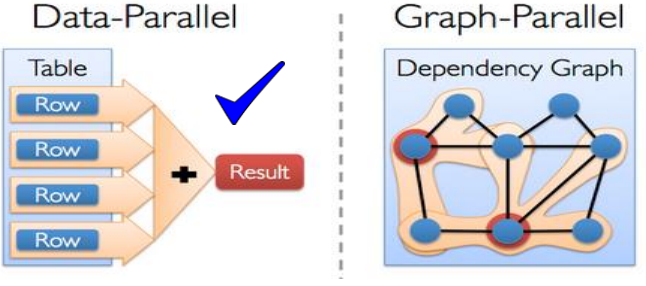
Social media data

BFS
DFS
CC
SCC
SSSP
ASP
A*
Community
Centrality
Diameter
Page Rank
MIS
SALSA…

25

Data-Parallel vs. Graph-Parallel

- *Data-Parallel* for all? *Graph-Parallel* is hard!
  - Data-Parallel (sort/search - randomly split data to feed MapReduce)
  - Not every graph algorithm is parallelisable (interdependent computation)
  - Not much data access locality
  - High data access to computation ratio



26

Graph-Parallel

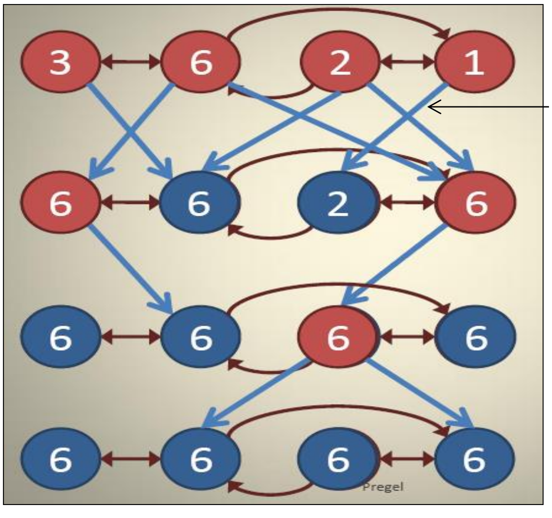
- Graph-Parallel (Graph Specific Data Parallel)
  - Vertex-based iterative computation model
  - Use of iterative Bulk Synchronous Parallel Model
    → Pregel (Google), Giraph (Apache), Graphlab, GraphChi (CMU)
  - Optimisation over data parallel
    → GraphX/Spark (U.C. Berkeley)
  - Data-flow programming – more general framework
    → NAIAD (MSR)

27

BSP Example

- Finding the largest value in a connected graph

Local Computation
Communication
Local Computation
Communication
...



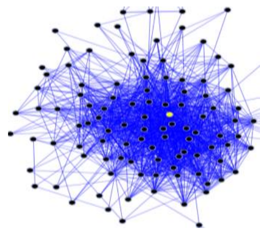
Message

28

UNIVERSITY OF CAMBRIDGE

Graph Computation

- Two characteristic patterns: traversal and fixed-point iteration
- Breadth-first search (weakly-connected components)
  - Search proceeds in a frontier
  - 90% computation, 10% communication
- PageRank
  - All vertices active in each iteration
  - 50% computation, 50% communication

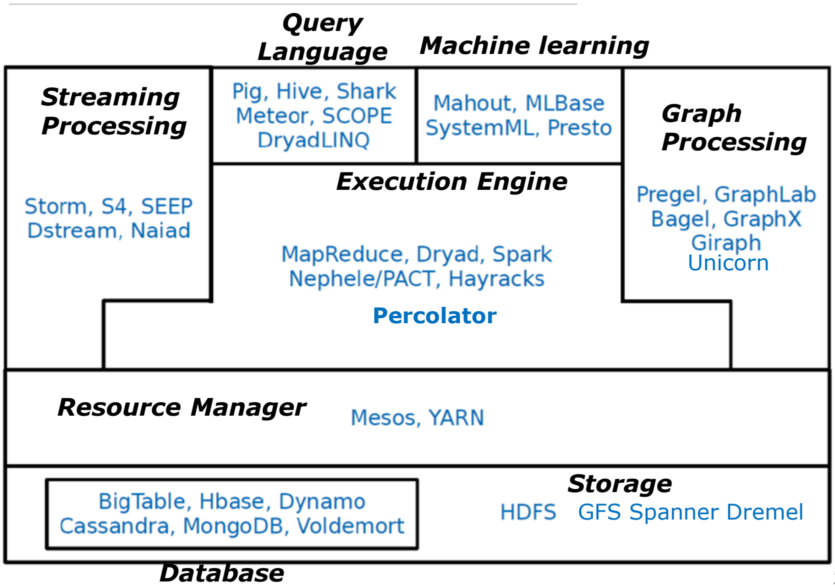


(* based on Pannotia benchmark suite)

29

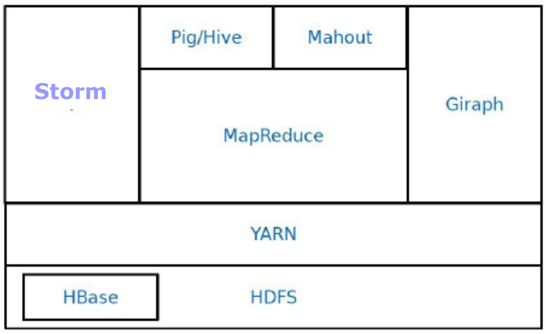
UNIVERSITY OF CAMBRIDGE

Big Data Analytics Stack

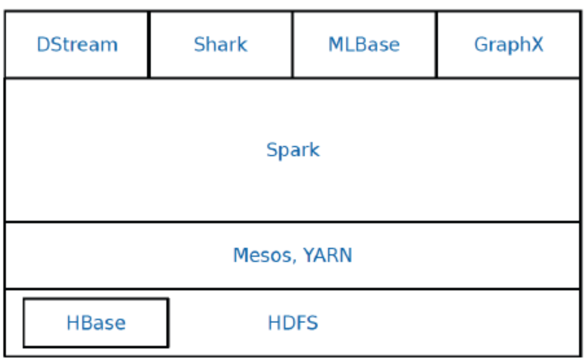


30

Hadoop Big Data Analytics Stack



Spark Big Data Analytics Stack



Do we really need a large cluster?

- A laptop can perform sufficiently

Twenty pagerank iterations

System	cores	twitter_rv	uk_2007_05
Spark	128	857s	1759s
Giraph	128	596s	1235s
GraphLab	128	249s	833s
GraphX	128	419s	462s
Single thread	1	300s	651s

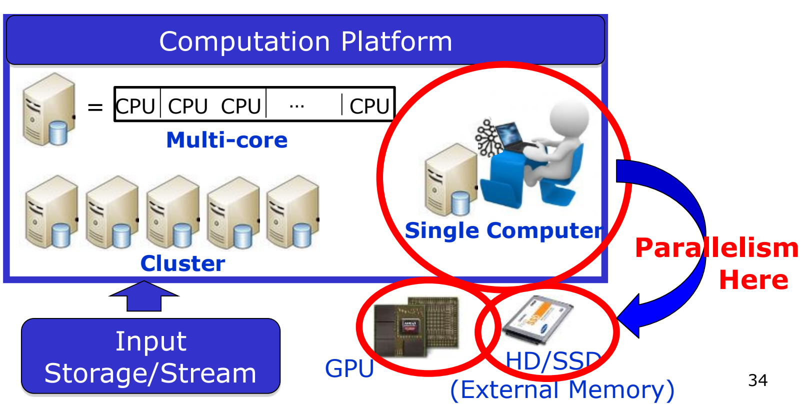
Label propagation to fixed-point (graph connectivity)

System	cores	twitter_rv	uk_2007_05
Spark	128	1784s	8000s+
Giraph	128	200s	8000s+
GraphLab	128	242s	714s
GraphX	128	251s	800s
Single thread	1	153s	417s

from blog by Frank McSherry 33

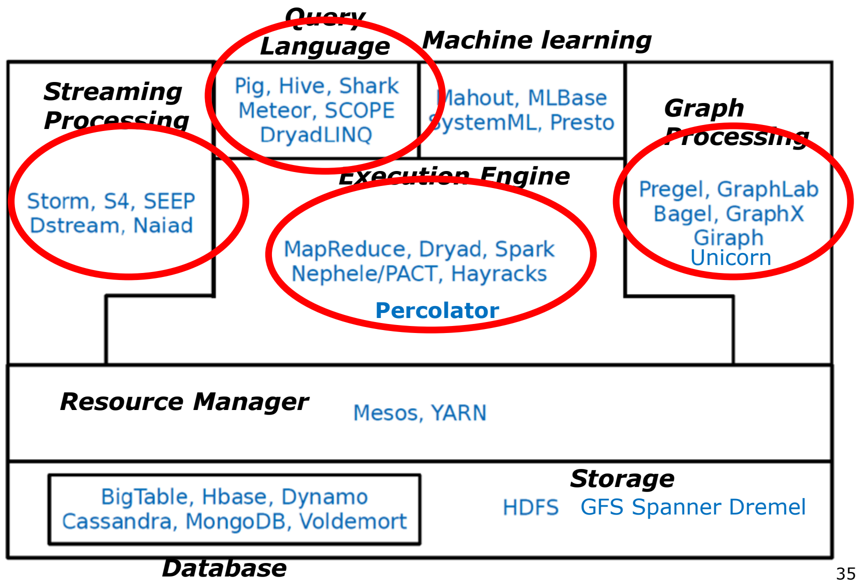
Single Computer?

- Use of powerful HW/SW parallelism
  - SSDs as external memory
  - GPU for massive parallelism
- Exploit graph structure/algorithm for processing



34

Big Data Analytics Stack



35

Topic Areas

Session 1: Introduction

Session 2: Programming in Data Centric Environment

Session 3: Processing Models of Large-Scale Graph Data

Session 4: Data Flow Programming Hands-on Tutorial with EC2

Session 6: Stream Data Processing + Guest lecture

Session 5: Optimisation in Data Processing

Session 7: Machine Learning for Computer System's Optimisation

Session 8: Project Study Presentation

36

Summary

- R212 course web page:

www.cl.cam.ac.uk/~ey204/teaching/ACS/R212_2015_2016

- Enjoy the course!