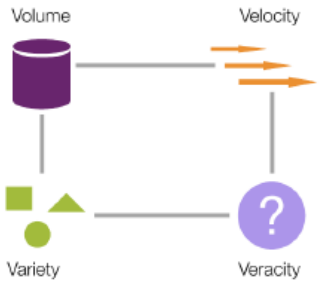
Challenges for Large-scale Data Processing

Eiko Yoneki

University of Cambridge Computer Laboratory

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)

2

Massive Data: Scale-Up vs Scale-Out

- Popular solution for massive data processing
  → scale and build distribution, combine theoretically unlimited number of machines in single distributed storage
- Scale-up: add resources to single node (many cores) in system (e.g. HPC)
- Scale-out: add more nodes to system (e.g. Amazon EC2)

Challenges

- Distribute and shard parts over machines
  - Still fast traversal and read to keep related data together
  - Scale out instead scale up
  - Parallelisable data distribution and processing is key
- 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
  → Stream processing

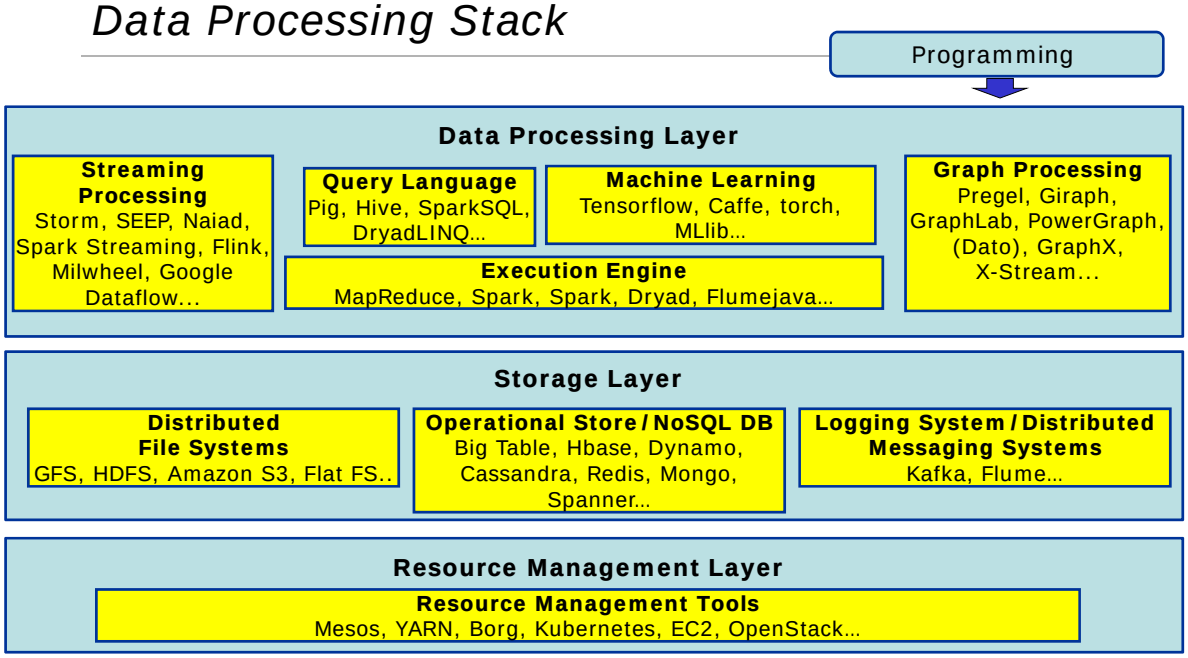
Technologies

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

    cf. Many 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)

5

Data Processing Stack



6

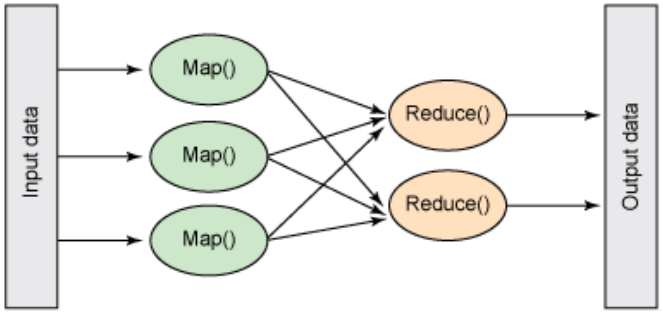
NoSQL (Schema Free) Database

- 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 …

7

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)

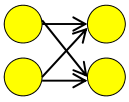


8

Data Flow Programming

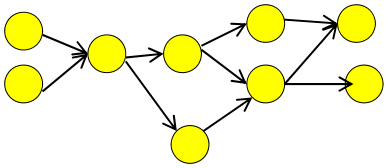
- Non standard programming models
- Data (flow) parallel programming
  - e.g. MapReduce, Dryad/LINQ, NAIAD, Spark

MapReduce: Hadoop



Two-Stage fixed dataflow

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



More flexible dataflow model

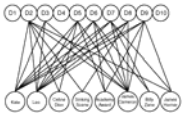
Typical Operation with Big Data

- 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..)

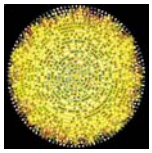
Do we need new types of algorithms?

- 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)

Emerging Massive-Scale Graph Data



Bipartite graph of phrases in documents



Protein Interactions [genomebiology.com]

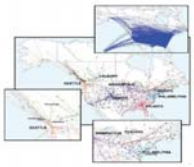
Gene expression data

BFS
DFS
CC
SCC
SSSP
APSP
A*
Community
Centrality
Diameter
Page Rank
MIS
SALSA...

Social media data

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

Web 1.4B pages(6.6B links)



Airline Graphs

12

Graph Computation Challenges

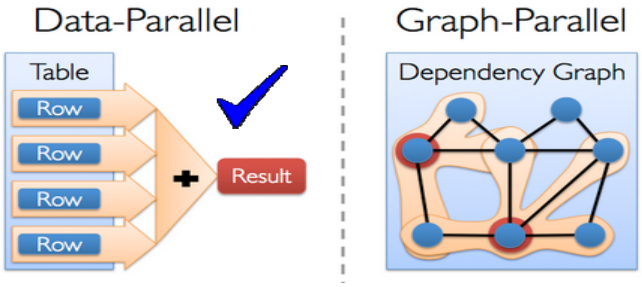
1. Graph algorithms (BFS, Shortest path)
2. Query on connectivity (Triangle, Pattern)
3. Structure (Community, Centrality)
4. ML & Optimisation (Regression, SGD)

- Data driven computation: dictated by graph's structure and parallelism based on partitioning is difficult
- Poor locality: graph can represent relationships between irregular entries and access patterns tend to have little locality
- High data access to computation ratio: graph algorithms are often based on exploring graph structure leading to a large access rate to computation ratio

13

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

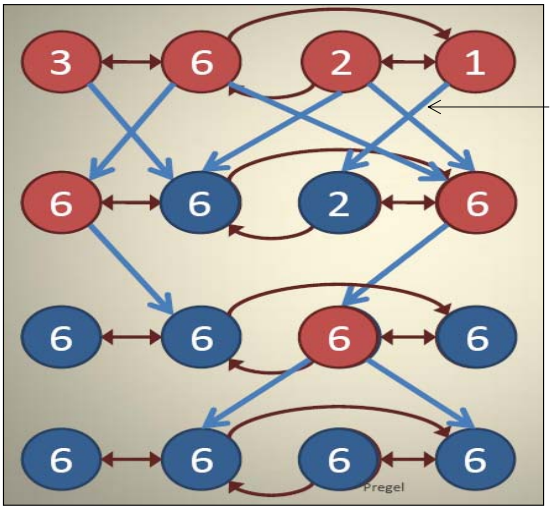


14

BSP Example

- Finding the largest value in a connected graph

Local Computation
Communication
Local Computation
Communication
...



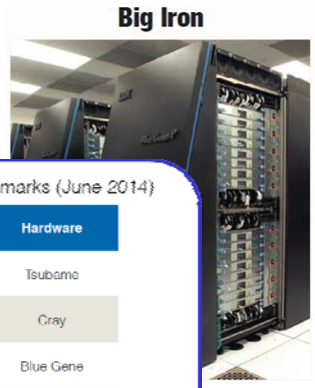
Message

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 - Dato)
  - Optimisation over data parallel
    → GraphX/Spark (U.C. Berkeley)
  - Data-flow programming – more general framework
    → NAIAD (MSR)

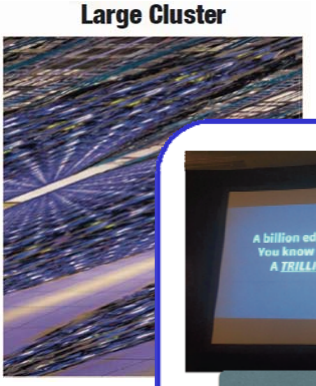
Are Large Clusters and Many cores Efficient?

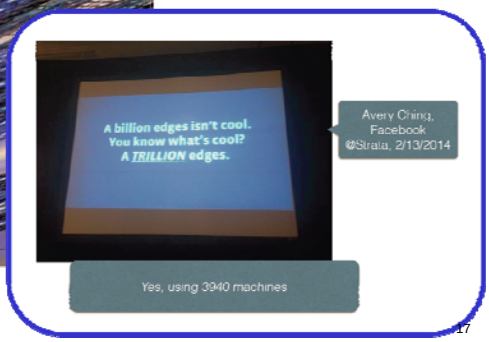
- Brute force approach really efficiently works?
  - Increase of number of cores (including use of GPU)
  - Increase of nodes in clusters



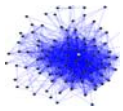
HPC/Graph500 benchmarks (June 2014)

Graph Edges	Hardware
1 trillion	Tsubame
1 trillion	Cray
1 trillion	Blue Gene
1 trillion	NEC





Do we really need large clusters?



- Laptops are sufficient?

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

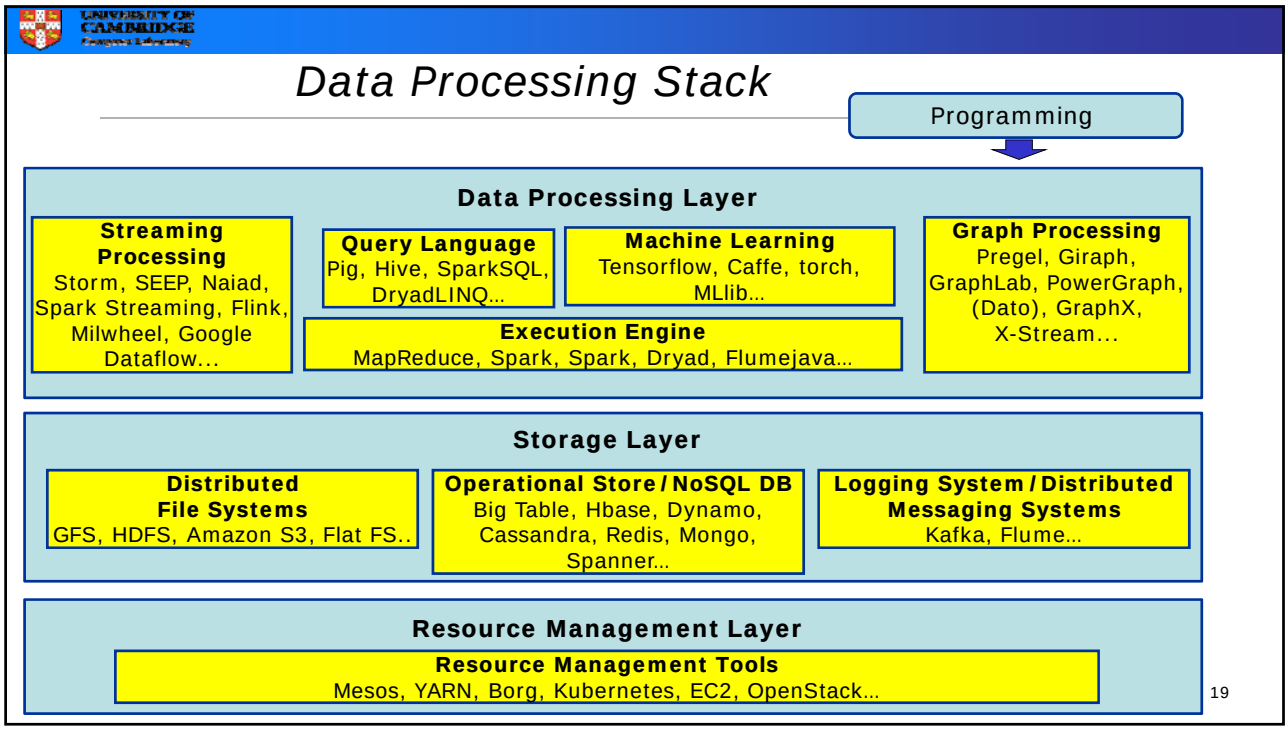
Label propagation to fixed-point (graph connectivity)

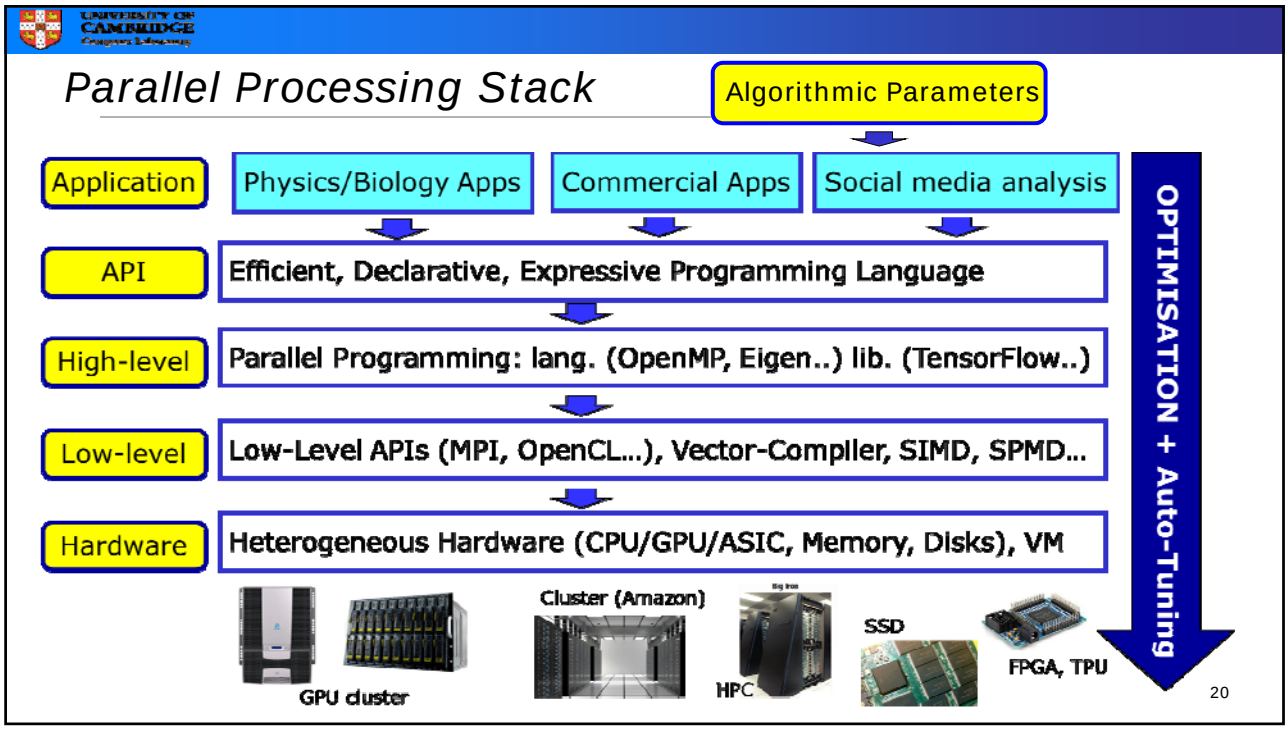
System	cores	twitter_rv	uk_2007_05
Spark	128	1784s	8000s+
Giraph	128	200s	8000s+
GraphLab	128	242s	714s
GraphX	128	251s	800s
Single thread	1	153s	417s

Fixed-point iteration: All vertices active in each iteration (50% computation, 50% communication)

Traversal: Search proceeds in a frontier (90% computation, 10% communication)

from Frank McSherry HotOS 2015


Data Processing Stack
Programming
Data Processing Layer
Streaming Processing
Storm, SEEP, Naiad, Spark Streaming, Flink, Milwheel, Google Dataflow...
Query Language
Pig, Hive, SparkSQL, DryadLINQ...
Machine Learning
Tensorflow, Caffe, torch, MLlib...
Execution Engine
MapReduce, Spark, Spark, Dryad, Flumejava...
Graph Processing
Pregel, Giraph, GraphLab, PowerGraph, (Dato), GraphX, X-Stream...
Storage Layer
Distributed File Systems
GFS, HDFS, Amazon S3, Flat FS..
Operational Store / NoSQL DB
Big Table, Hbase, Dynamo, Cassandra, Redis, Mongo, Spanner...
Logging System / Distributed Messaging Systems
Kafka, Flume...
Resource Management Layer
Resource Management Tools
Mesos, YARN, Borg, Kubernetes, EC2, OpenStack...
19


Parallel Processing Stack
Algorithmic Parameters
Application
Physics/Biology Apps
Commercial Apps
Social media analysis
API
Efficient, Declarative, Expressive Programming Language
High-level
Parallel Programming: lang. (OpenMP, Eigen..) lib. (TensorFlow..)
Low-level
Low-Level APIs (MPI, OpenCL...), Vector-Compiler, SIMD, SPMD...
Hardware
Heterogeneous Hardware (CPU/GPU/ASIC, Memory, Disks), VM
OPTIMISATION + Auto-Tuning
GPU cluster
Cluster (Amazon)
HPC
SSD
FPGA, TPU
20

Topic Areas

Session 1: Introduction

Session 2: Data flow programming: Map/Reduce to TensorFlow

Session 3: Large-scale graph data processing

Session 4: Hands-on Tutorial: Map/Reduce and Deep Neural Network

Session 5: Stream Data Processing + Guest lecture

Session 6: Machine Learning for Optimisation of Computer Systems

Session 7: Task scheduling, Performance, and Resource Optimisation

Session 8: Project Study Presentation

Summary

- R244 course web page:

www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2017_2018

- Enjoy the course!