Large-scale Data Processing and Optimisation

Eiko Yoneki

University of Cambridge Computer Laboratory

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
  → Parallelisable data distribution and processing is key
- 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)

2

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)

3

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 …

4

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, 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...

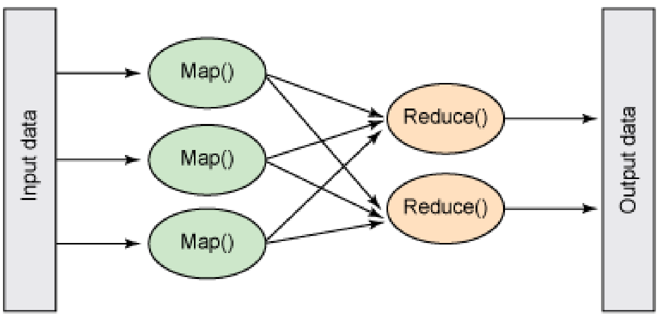
Logging System / Distributed Messaging Systems
Kafka, Flume...

Resource Management Layer

Resource Management Tools
Mesos, YARN, Borg, Kubernetes, EC2, OpenStack...

MapReduce Programming

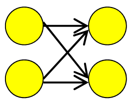
- Target problem needs to be parallelisable
- Split into a set of smaller code (map)
- Next small piece of code executed in parallel
- Results from map operation get synthesised into a result of original problem (reduce)



Data Flow Programming

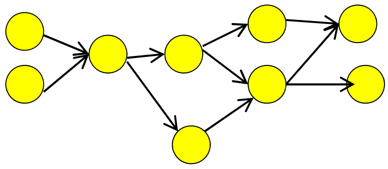
- Non standard programming models
- Data (flow) parallel programming
  - e.g. MapReduce, Dryad/LINQ, NAIAD, Spark, Tensorflow…

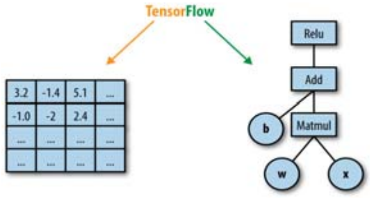
MapReduce: Hadoop



Two-Stage fixed dataflow

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



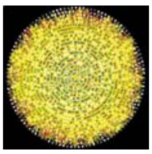


More flexible dataflow model

7

Emerging Massive-Scale Graph Data

Bipartite graph of phrases in documents



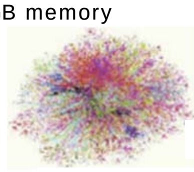
Protein Interactions [genomebiology.com]

Gene expression data

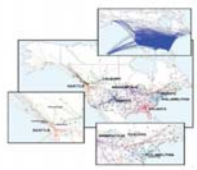
BFS DFS CC SCC SSSP ASP A* Community Centrality Diameter Page Rank MIS SALSA…

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

Social media data



Web 1.4B pages(6.6B links)



Airline Graphs

8

Graph Computation Challenges

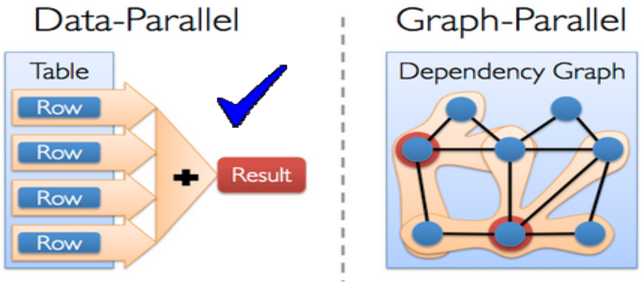
1. Graph algorithms (BFS, Shortest path)
2. Query on connectivity (Triangle, Pattern)
3. Structure (Community, Centrality)
4. ML & Optimisation (Regression, SGD)

- 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

9

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



10

Graph-Parallel

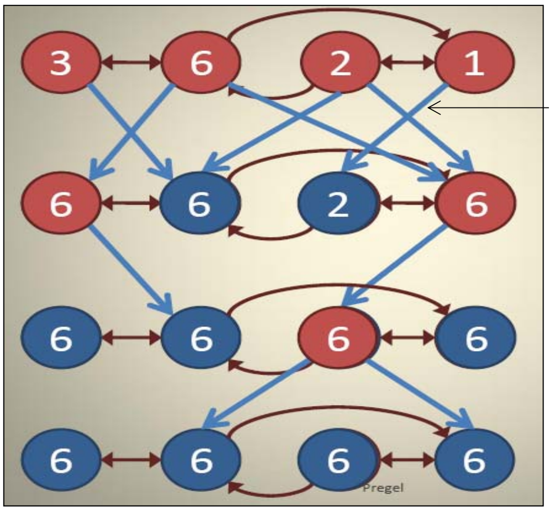
- Graph-Parallel (Graph Specific Data Parallel)
  - Vertex-based iterative computation model
  - Use of iterative Bulk Synchronous Parallel Model
    ➔ Pregel (Google), Giraph (Apache), Graphlab, GraphChi (CMU - Dato)
  - Optimisation over data parallel
    ➔ GraphX/Spark (U.C. Berkeley)
  - Data-flow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

11

Bulk synchronous parallel: Example

- Finding the largest value in a connected graph

Local Computation
Communication
Local Computation
Communication
...



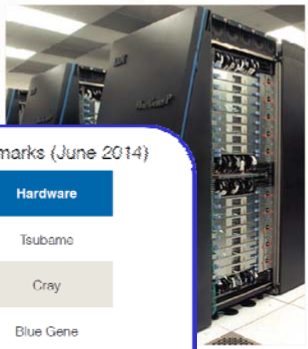
Message

12

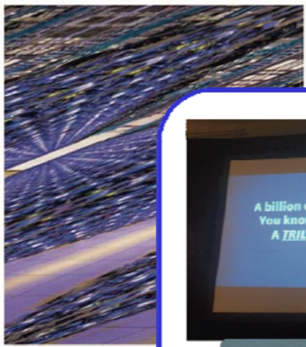
Are Large Clusters and Many cores Efficient?

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

Big Iron

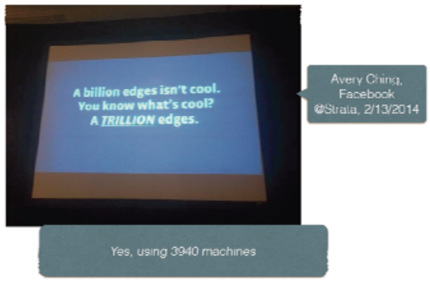


Large Cluster

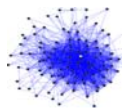


HPC/Graph500 benchmarks (June 2014)

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 for Neural Networks

- Practicalities of training Neural Networks
- Leveraging heterogeneous hardware

Modern Neural Networks Applications:

Image Classification

Reinforcement Learning

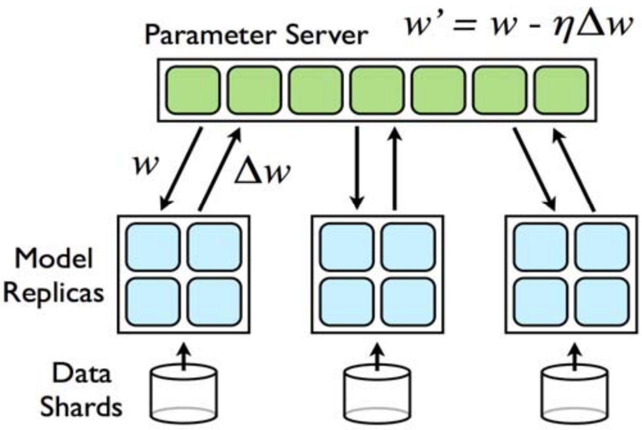
15

Single Machine Setup

- One or more beefy GPUs

16

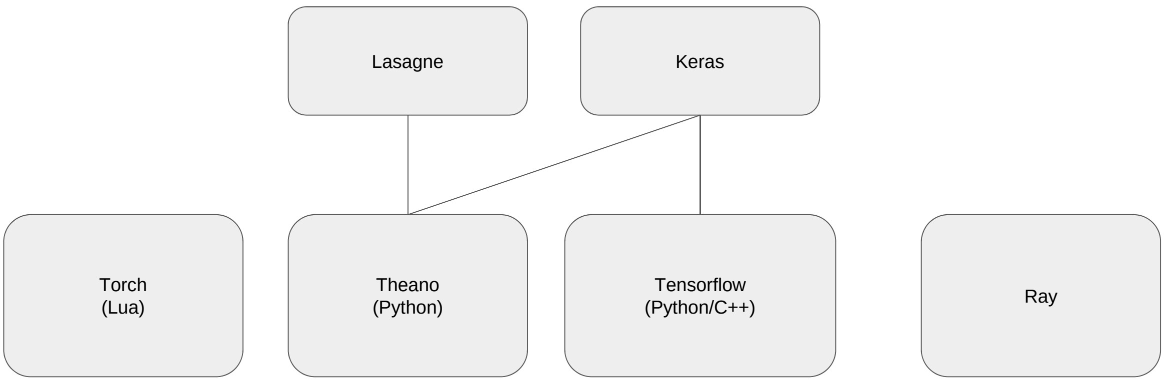
Distribution: Parameter Server Architecture



- Can exploit both Data Parallelism and Model Parallelism

Source: Dean et al.: Large Scale Distributed Deep Networks

Software Platform for ML Applications



RLgraph: Dataflow Composition

- Our group's work

API, Component configuration			Prebuilt models, inference
RLgraph component graph			Model design, dataflow composition
TensorFlow	PyTorch	...	Local backends variables/operations
Distributed TF	Horovod	Ray	Distributed execution engine
Hardware: CPU, GPU, TPU, FPGAs...			Execution, orchestration

19

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, 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...

20

Computer Systems Optimisation

- What is performance?
  - Resource usage (e.g. time, power)
  - Computational properties (e.g. accuracy, fairness, latency)
- How do we improve it:
  - Manual tuning
  - Runtime autotuning
  - Static time autotuning

Manual Tuning: Profiling

- Always the first step
- Simplest case: Poor man's profiler
  - Debugger + Pause
- Higher level tools
  - Perf, Vtune, Gprof...
- Distributed profiling: a difficult active research area
  - No clock synchronisation guarantee
  - Many resources to consider
  - System logs can be leveraged

→ tune implementation based on profiling (never captures all interactions)

Auto-tuning Complex Systems

- Many dimensions
- Expensive objective function
- Hand-crafted solutions impractical (e.g. extensive offline analysis)

Blackbox Optimisation

✓ can surpass human expert-level tuning

- Grid search $\theta \in [1, 2, 3, \ldots]$
- Evolutionary approaches (e.g. PetaBricks)
- Hill-climbing (e.g. OpenTuner)
- Bayesian optimisation (e.g. SPEARMINT)

1000s of evaluations of objective function

Computation more expensive

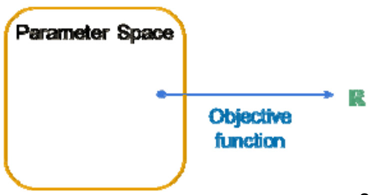
Fewer samples

23

Static time Autotuning

Especially useful when:

- There is a variety of environments (hardware, input distributions)
- The parameter space is difficult to explore manually
- Defining a parameter space
  - e.g. Petabricks: A language and compiler for algorithmic choice (2009)
    - BNF-like language for parameter space
    - Uses an evolutionary algorithm for optimisation
    - Applied to Sort, matrix multiplication



24

Ways to do an Optimisation

Random search: No risk of 'getting stuck'
potentially many samples required

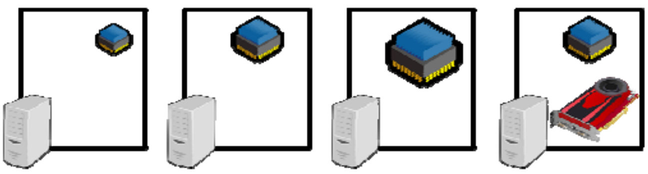
Evolution strategies: Evaluate permutations against fitness function

Bayes Opt: Sample efficient, requires continuous function, some configuration

Random Search	Genetic algorithm / Simulated annealing	Bayesian Optimisation
No overhead	Slight overhead	High overhead
High #evaluation	Medium-high #evaluation	Low #evaluation

Parameter Space of Task Scheduler

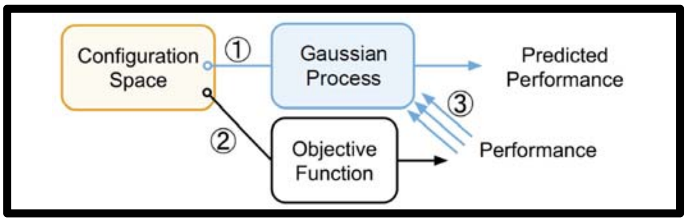
- Tuning distributed SGD scheduler over TensorFlow
- 10 heterogeneous machines with ~32 parameters
  - **$\sim 10^{53}$** possible valid configurations
- Objective function: minimise distributed SGD iteration time



Parameter server	✗	✗	✓	✓
Worker	✗	✓	✓	✓
# Inputs	0	10	16	10 - 28

Bayesian Optimisation

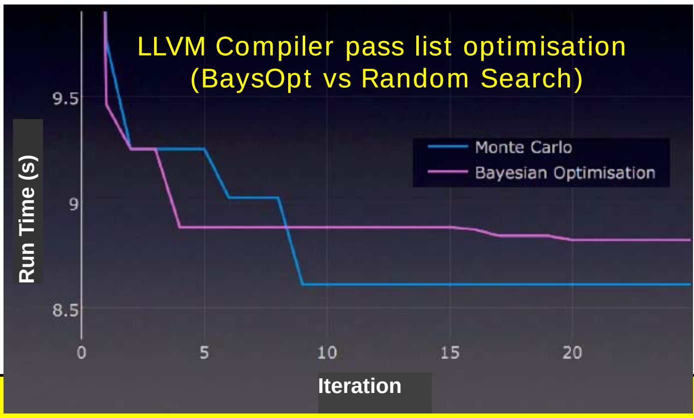
- Iteratively builds probabilistic model of objective function
- Typically Gaussian process as probabilistic model
- Data efficient: converges quickly



Limitations:
- In high dimensional parameter space, model does not converge to objective function
- Not efficient to model dynamic and/or combinatorial model

27

Bayesian Optimisation



Limitations:
- In high dimensional parameter space, model does not converge to objective function
- Not efficient to model dynamic and/or combinatorial model

28

Computer Systems Optimisation Models

- ***Short-term dynamic control:*** major system components are under dynamic load, such as resource allocation and stream processing, where the future load is not statistically dependent on the current load.
  BaysOpt is sufficient to optimise distinct workloads. For dynamic workload, Reinforcement Learning would perform better.

- ***Combinatorial optimisation:*** a set of options to be selected from a larger set under potential rules of combination. There is no straightforward similarity between different combinations. Many problems in device assignment, indexing, compiler optimisation fall in this category.
  BaysOpt cannot be easily applied. Either learning online if the task is cheap via random sampling, or via RL + pre-training if the task is expensive, or massively parallel online training if the resources are available.

Many systems problems are combinatorial in nature

AutoML: Neural Architecture Search

Current: ML expertise + Data + Computation

AutoML aims turning into: Data + 100 x Computation

- Use of Reinforcement Learning, Evolutionary Algorithms

- ..and tune network model?
  - Graph transformation
  - Compression
  - + Hyper parameter tuning

Probabilistic Model

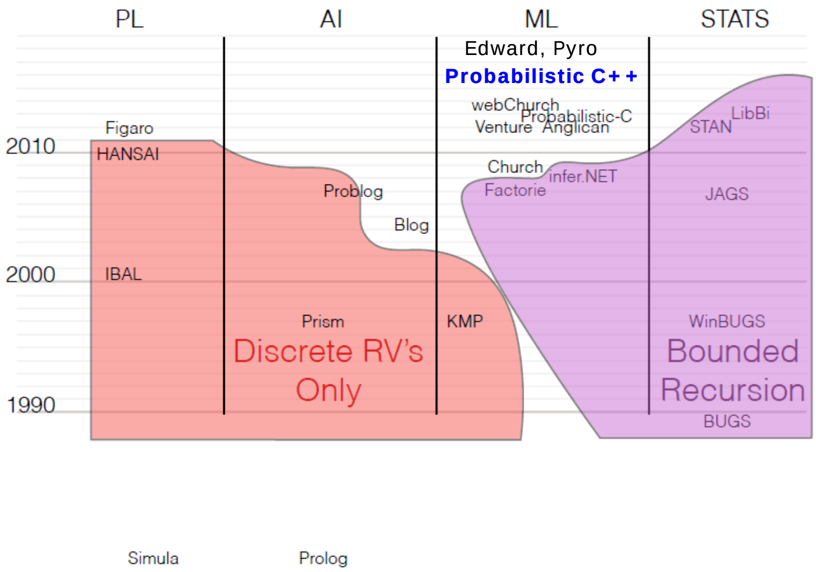
- Probabilistic models incorporate random variables and probability distributions into the model
  - Deterministic model gives a single possible outcome
  - Probabilistic model gives a probability distribution
- Used for various probabilistic logic inference (e.g. MCMC-based inference, Bayesian inference…)

Python based PP:

- Pyro: https://pyro.ai/examples
- Edward: http://edwardlib.org

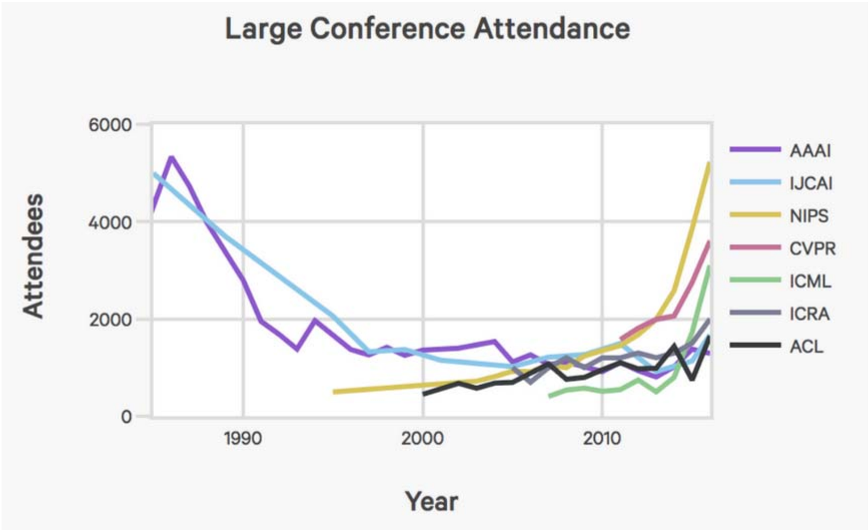
31

Probabilistic Programming



32

Scale of Community Size in ML/AI



33

SysML Conference spawn in 2018-2019

- SysML is a conference targeting research at the intersection of systems and machine learning
- Aims to elicit new connections amongst these fields, including identifying best practices and design principles for learning systems, as well as developing novel learning methods and theory tailored to practical machine learning workflows

Steering Committee

Jennifer Chayes
Bill Dally
Jeff Dean
Michael I. Jordan
Yann LeCun
Fei-Fei Li
Alex Smola
Dawn Song
Eric Xing

34

Gap between Research and Practice

Device Placement Optimization with Reinforcement Learning

20H with 80GPUs!

Azalia Mirhoseini [*1 2] Hieu Pham [*1 2] Quoc V. Le [1] Benoit Steiner [1] Rasmus Larsen [1] Yuefeng Zhou [1]
Naveen Kumar [3] Mohammad Norouzi [1] Samy Bengio [1] Jeff Dean [1]

35

Summary

- R244 course web page:

www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2019_2020

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: Probabilistic Programming + Guest lecture (Brooks Paige)
Session 6: Exploring ML for optimisation in computer systems
Session 7: ML based Optimisation examples in Computer Systems
Session 8: Project Study Presentation (2019.12.12 @11:00)

36