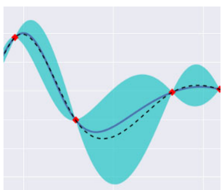
Large-scale Data Processing and Optimisation



Eiko Yoneki

University of Cambridge Computer Laboratory

1

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

2

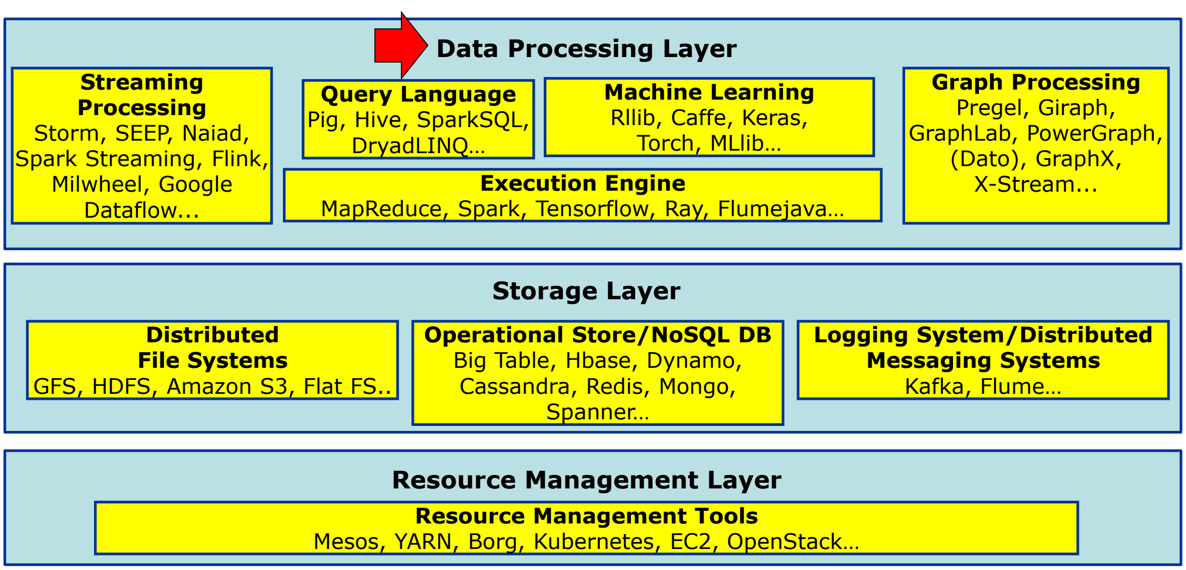
Technologies supporting Cluster Computing

- Distributed infrastructure
  - Cloud (e.g. Infrastructure as a service, Amazon EC2, GCP, 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

3

Data Processing Stack

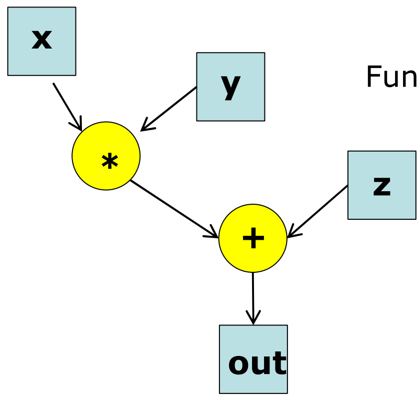


4

4

Data Flow Programming

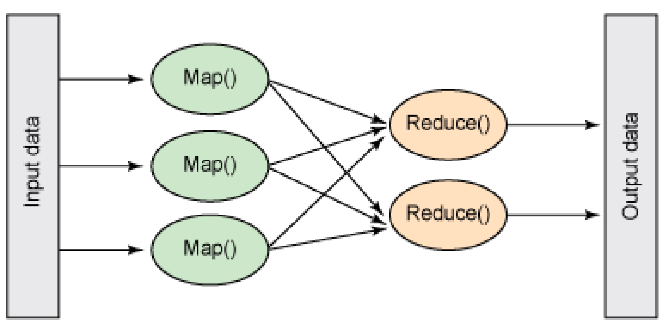
- Non-standard programming models
- Powerful abstraction: mapping computation into dataflow graphs



Function f(x, y, z) = x* y + z

MapReduce Programming

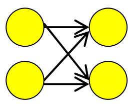
- Target problem needs to be parallelisable
- Split into a set of smaller code (map)
- Next small piece of code executed in parallel
- Results from map operation get synthesised into a result of original problem (reduce)



Data Flow Programming Examples

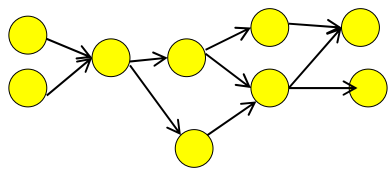
- Data (flow) parallel programming
  - e.g. MapReduce, Dryad/LINQ, NAIAD, Spark, Tensorflow…

MapReduce:
Hadoop

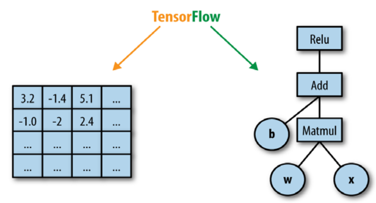


Two-Stage fixed dataflow

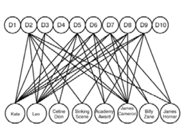
DAG (Directed Acyclic Graph)
based: Dryad/Spark…



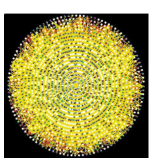
More flexible dataflow model



Emerging Massive-Scale Graph Data



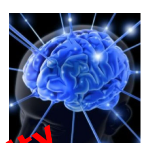
Bipartite graph of phrases in documents



Protein Interactions [genomebiology.com]

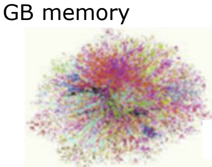
Gene expression data

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

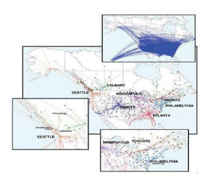


Brain Networks: 100B neurons(700T links) requires 100s GB memory

Social media data



Web 1.4B pages(6.6B links)



Airline Graphs

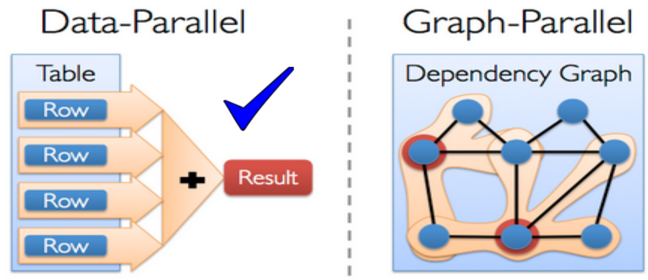
Graph Computation Challenges

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

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



Graph-Parallel

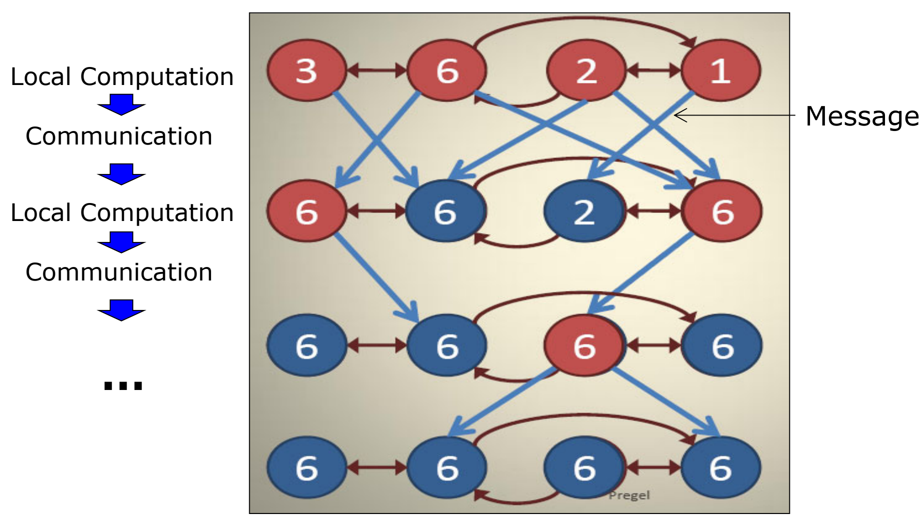
- Graph-Parallel (Graph Specific Data Parallel)
  - Vertex-based iterative computation model
  - Use of iterative Bulk Synchronous Parallel Model
    → Pregel (Google), Giraph (Apache), Graphlab, GraphChi (CMU - Dato)
  - Optimisation over data parallel
    → GraphX/Spark (U.C. Berkeley)
  - Data-flow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

11

11

Bulk synchronous parallel: Example

- Finding the largest value in a connected graph



12

12

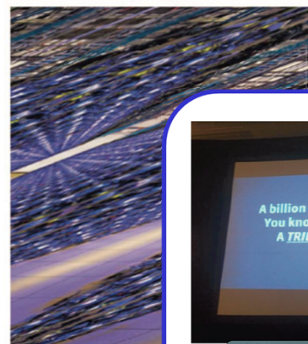
UNIVERSITY OF CAMBRIDGE Computer Laboratory

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

Avery Ching, Facebook @Strata, 2/13/2014

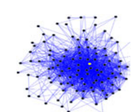
Yes, using 3940 machines

13

13

UNIVERSITY OF CAMBRIDGE Computer Laboratory

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

14

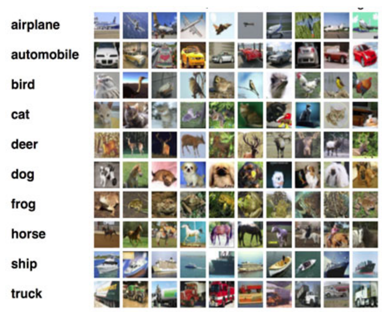
14

Data Processing for Neural Networks

- Practicalities of training Neural Networks
- Leveraging heterogeneous hardware

Modern Neural Networks Applications:

Image Classification



Reinforcement Learning

Image courtesy: The Guardian

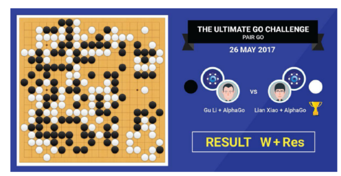


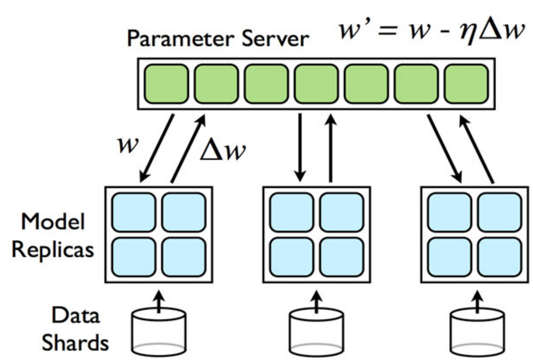
Image courtesy: Twitter - Deep Mind AI

15

Performance Improvement

- One or more beefy GPUs

- Parameter Architecture: exploit both Data Parallelism and Model Parallelism (by Google)



16

Computer Systems Optimisation

- How do we improve performance:
  - Manual tuning
  - Auto-tuning
- What is performance? – objective function of optimisation
  - Resource usage (e.g. time, power)
  - Computational properties (e.g. accuracy, fairness, latency)
- What is Optimisation Model?
  - Short-term dynamic control (e.g. stream processing: distinct workload or dynamic workload)
  - Combinatorial optimisation (e.g. indexing DB, device assignment)

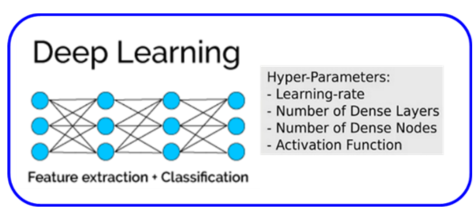
Many systems problems are combinatorial in nature

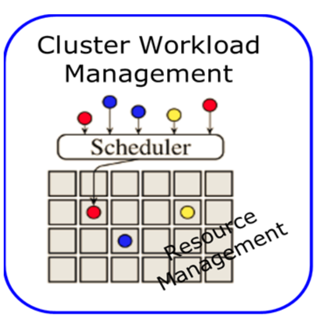
17

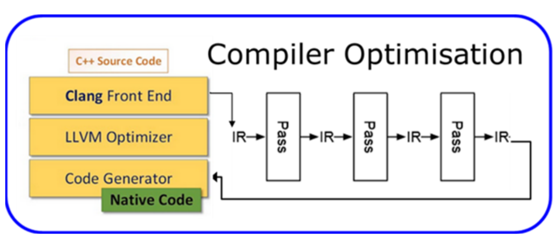
17

Turing Computer System is Complex Task

- Increasing data volumes and high-dimension parameter space
- Expensive Objective Functions
- Hand-crafted solutions impractical, often left static or configured through extensive offline analysis
- Not well-tuned system's performance does not scale







18

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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 θ ∈ [1, 2, 3, …]
- Random search
- 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

19

19

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Search Parameter Space

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

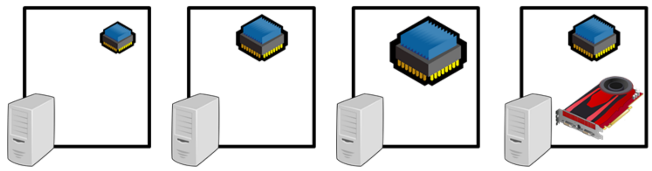
Random Search	Genetic algorithm / Simulated annealing	Bayesian Optimisation
No overhead	Slight overhead	High overhead
High #evaluation	Medium-high #evaluation	Low #evaluation

20

20

Parameter Space of Task Scheduler

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

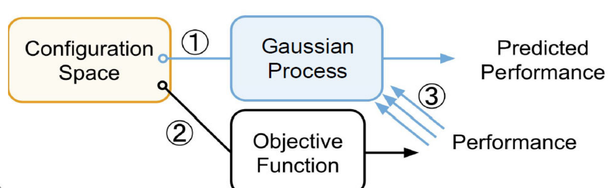


Parameter server	✗	✗	✓	✓
Worker	✗	✓	✓	✓
# Inputs	0	10	16	10 - 28

Bayesian Optimisation

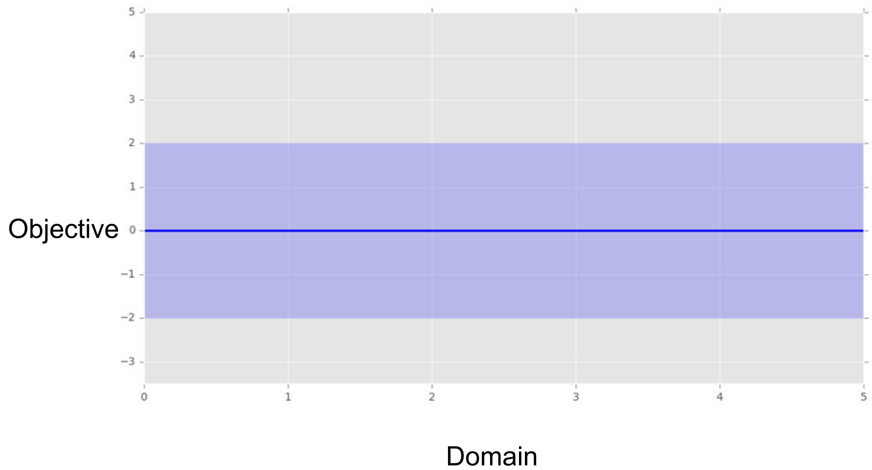
- Iteratively builds probabilistic model of objective function
- Typically Gaussian process as probabilistic model
- Data efficient: converges quickly

Input: Objective function $f()$
Input: Surrogate function initial distribution G
Input: Acquisition function $a()$
for $i = 1, 2, \dots$ **do**
  Sample point: $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x}} a(G, \boldsymbol{x})$
  Evaluate new point: $y_t \leftarrow f(\boldsymbol{x}_t)$
  Update surrogate distribution: $G \leftarrow G \mid (\boldsymbol{x}_t, y_t)$
end for



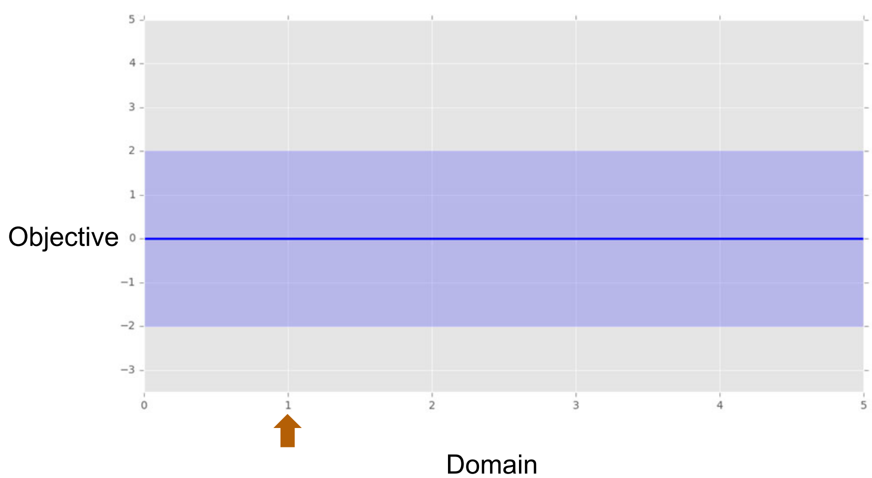
① Find promising point (high performance value in the model)
② Evaluate the objective function at that point
③ Update the model to reflect this new measurement

Bayesian Optimisation



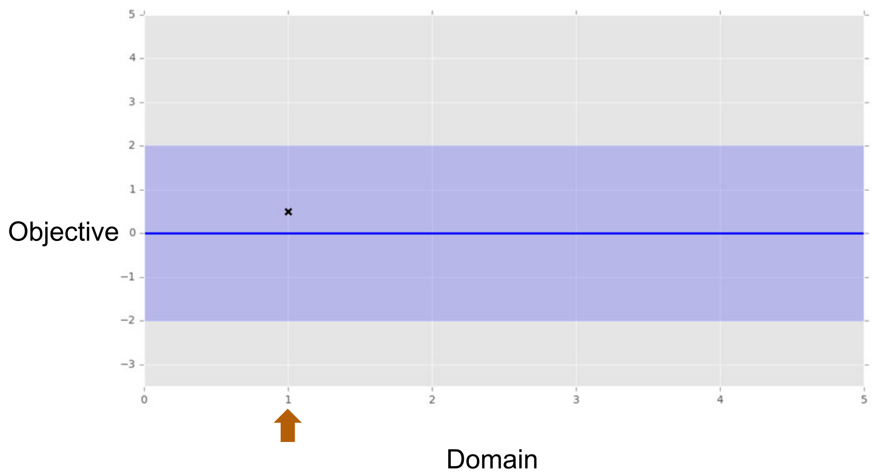
23

Bayesian Optimisation



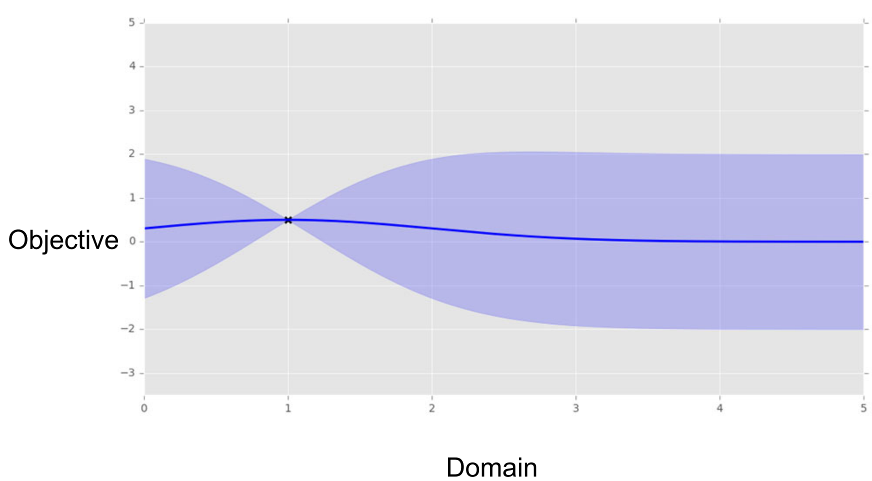
24

Bayesian Optimisation



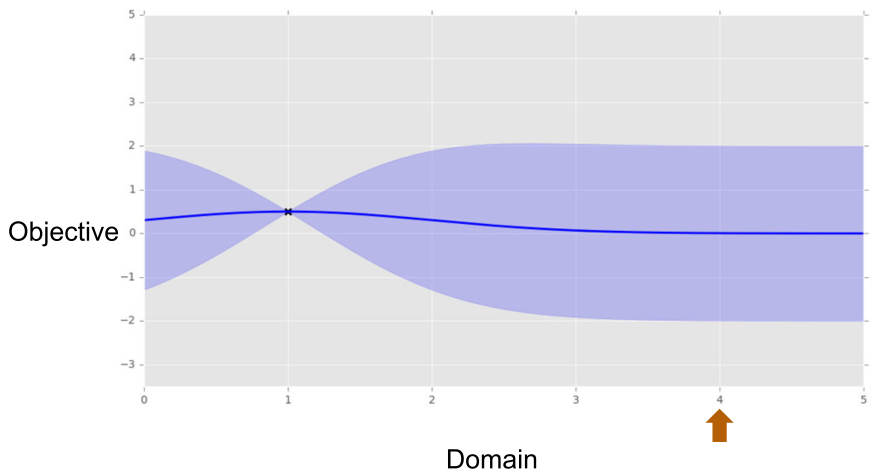
25

Bayesian Optimisation



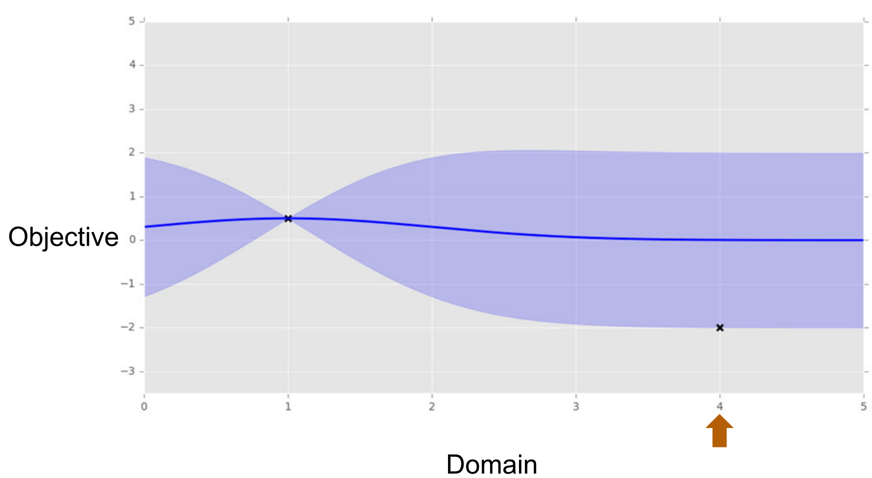
26

Bayesian Optimisation



27

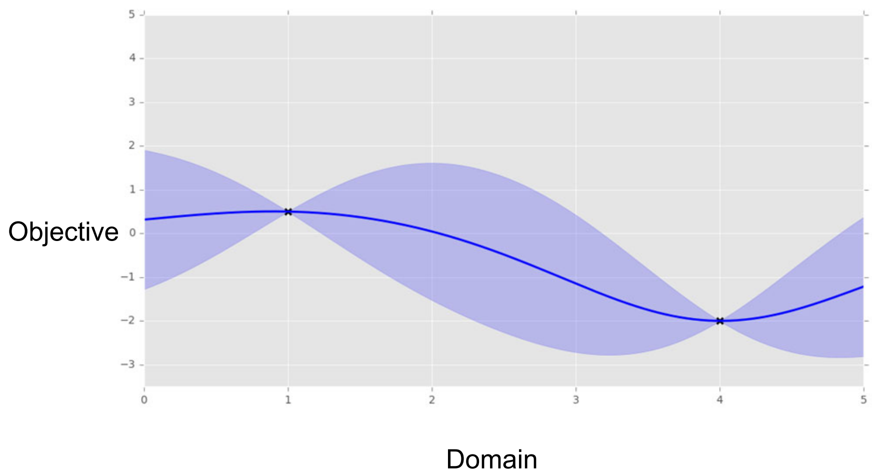
Bayesian Optimisation



28

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

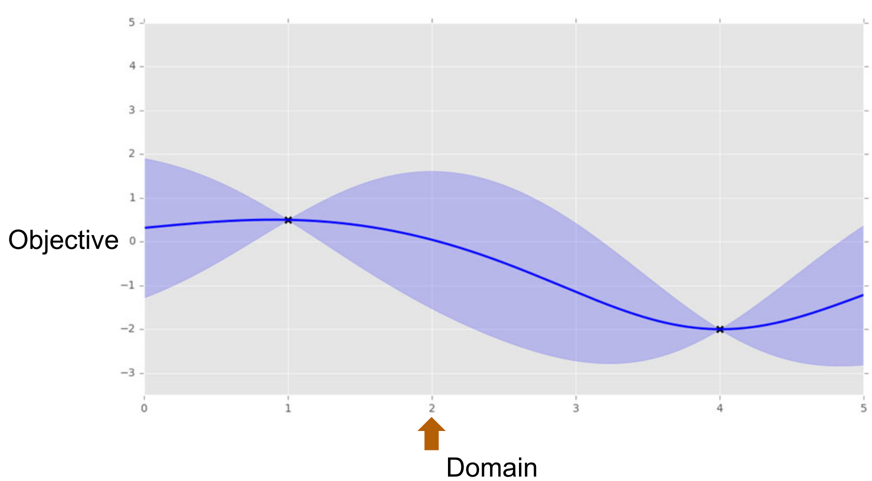
Bayesian Optimisation



29

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

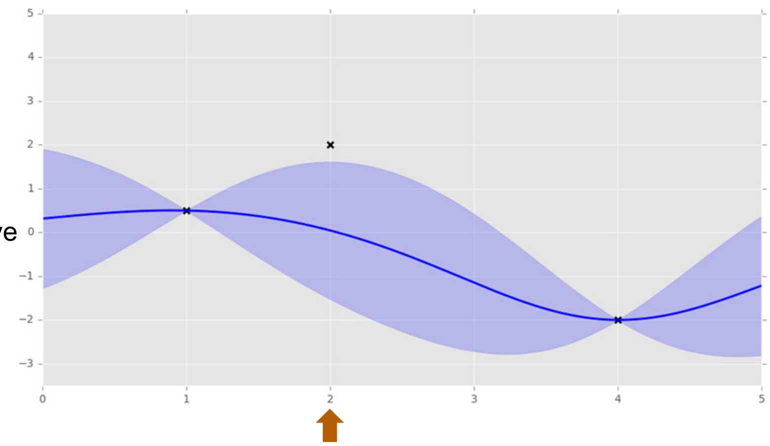
Bayesian Optimisation



30

Bayesian Optimisation

Objective



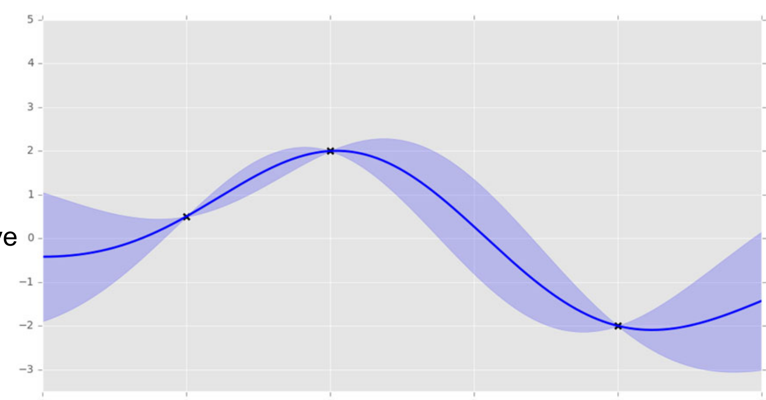
Domain

31

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bayesian Optimisation

Objective

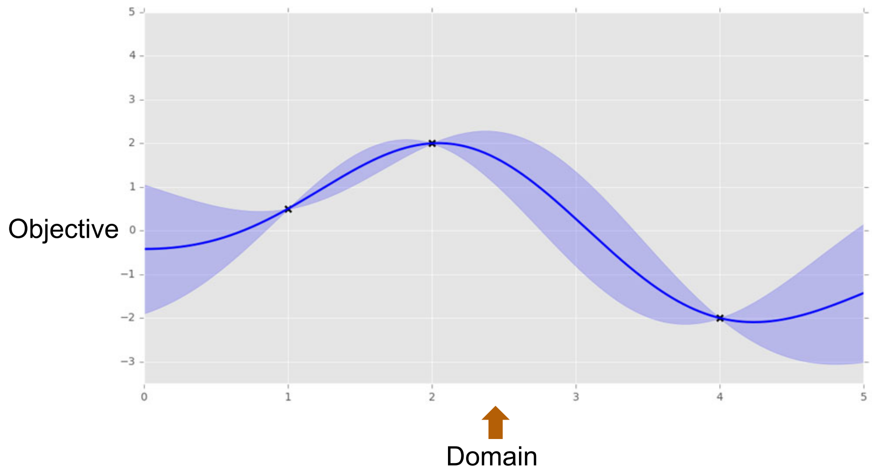


Domain

32

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

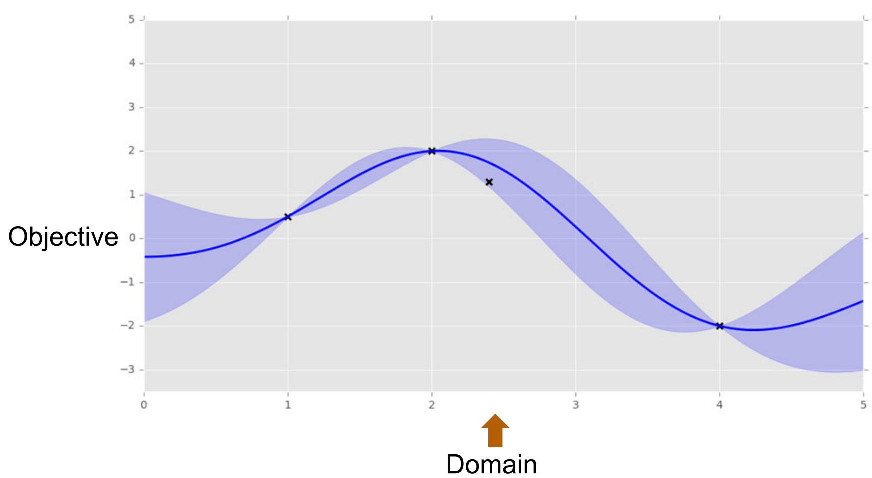
Bayesian Optimisation



33

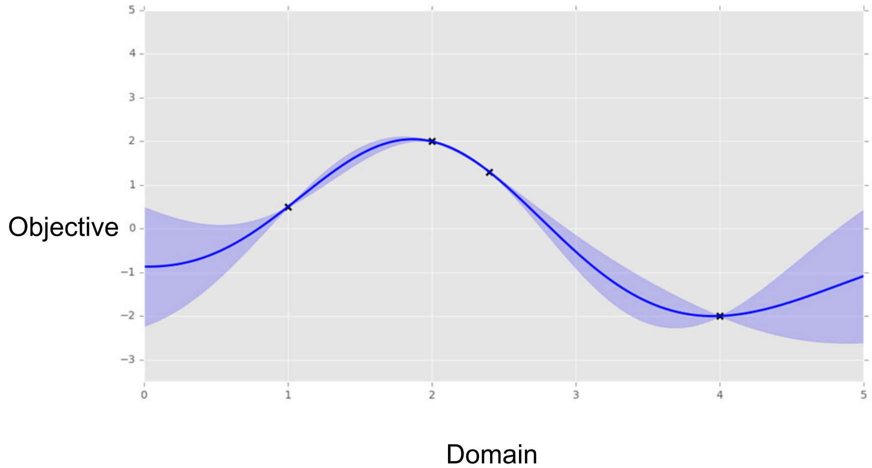
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bayesian Optimisation



34

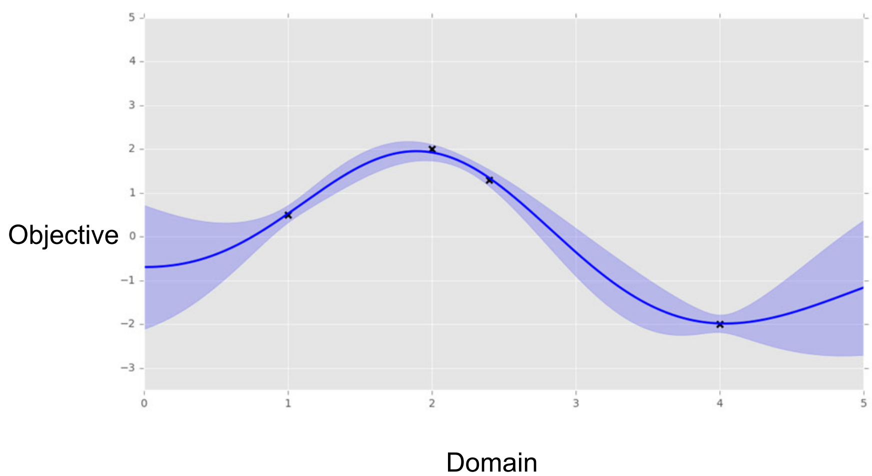
Bayesian Optimisation



35

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bayesian Optimisation



36

Further Bayesian Optimisation…

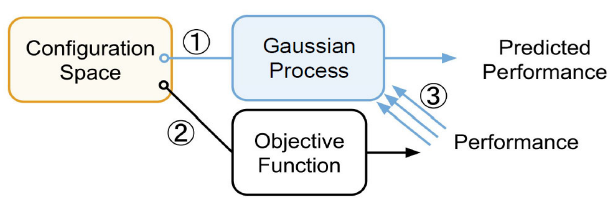
- BO overview/Tutorial
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2021_2022/aid/BO_overview_Archambeau.pdf
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2021_2022/aid/BO_overview_adams.pdf
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2021_2022/aid/BO_overview_gonzalez.pdf
- Papers
  - Review paper by Shahriari, et al. (2016): Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE 104(1):148-175, 2016.
  - Slides by Ryan Adams (2014): A Tutorial on Bayesian Optimization for Machine Learning. CIFAR NCAP Summer School.
  - Slides by Peter Frazier (2010): Tutorial: Bayesian Methods for Global and Simulation Optimization. INFORMS Annual Meeting.

37

37

Bayesian Optimisation

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



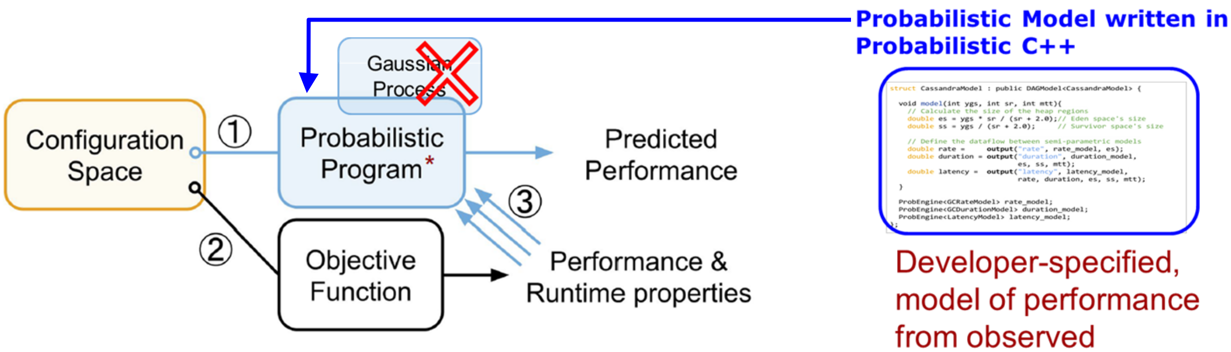
Pros:
✓ Data efficient: converges in few iterations
✓ Able to deal with noisy observations

Cons:
✗ In many dimensions, model does not converge to the objective function

38

38

Structured Bayesian Optimisation (SBO)



Developer-specified, model of performance from observed performance + arbitrary runtime characteristics

✓ Better convergence
✓ Use all measurements

BOAT: a framework to build **B**esp**O**ke **A**uto-**T**uners

39

39

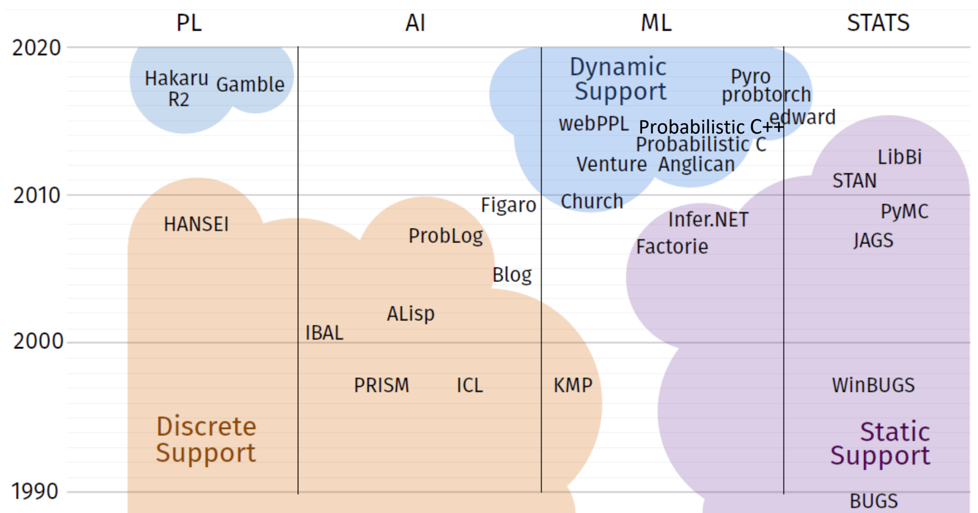
Probabilistic Model

- 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…)

Tutorial: Session 6 – Guest Lecture by Brooks Paige

40

Probabilistic Programming

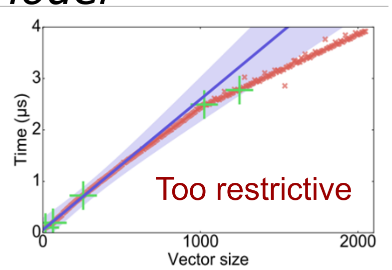


B. Paige

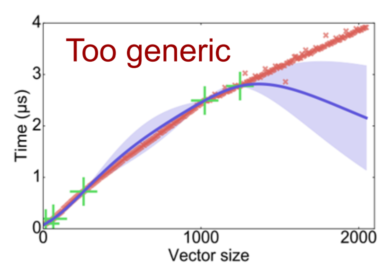
41

Semi-parametric Model

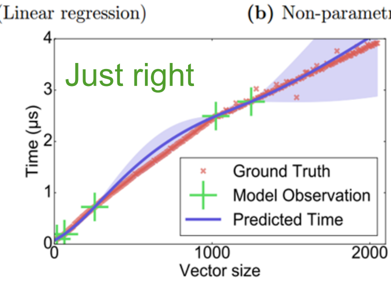
- Easy to use and well suited to SBO
  - Understand general trend of Objective function
  - High precision in region of optimum for finding highest performance



(a) Parametric (Linear regression)



(b) Non-parametric (Gaussian process)



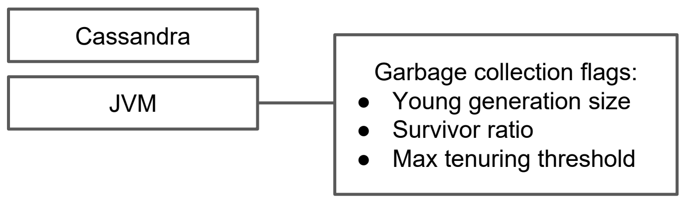
(c) Semi-parametric (Combination)

42

42

Example: JVM Garbage Collection

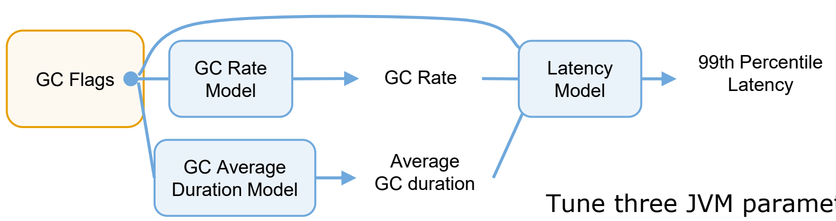
- Cassandra's garbage collection



- Minimise 99th percentile latency of Cassandra

Performance Improvement from Structure

User-given probabilistic model structured in semi-parametric model using Directed Acyclic Graph



Tune three JVM parameters of database (Cassandra) to minimise latency

DAG model in BOAT

```
struct CassandraModel : public DAGModel<CassandraModel> {

  void model(int ygs, int sr, int mtt){
    // Calculate the size of the heap regions
    double es = ygs * sr / (sr + 2.0);// Eden space's size
    double ss = ygs / (sr + 2.0);     // Survivor space's size

    // Define the dataflow between semi-parametric models
    double rate =     output("rate", rate_model, es);
    double duration = output("duration", duration_model,
                             es, ss, mtt);
    double latency =  output("latency", latency_model,
                             rate, duration, es, ss, mtt);
  }

  ProbEngine<GCRateModel> rate_model;
  ProbEngine<GCDurationModel> duration_model;
  ProbEngine<LatencyModel> latency_model;
};
```


GC Rate Semi-parametric model

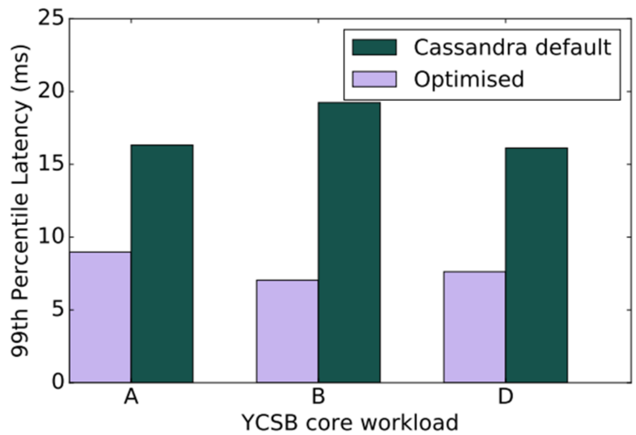
```
struct GCRateModel : public SemiParametricModel<GCRateModel> {

  GCRateModel() {
    allocated_mbs_per_sec =
     std::uniform_real_distribution<>(0.0, 5000.0)(generator);
    // set the GP parameters here
  }

  double parametric(double eden_size) const {
    // Model the rate as inversely proportional to Eden's size
    return allocated_mbs_per_sec / eden_size;
  }

  double allocated_mbs_per_sec;
};
```

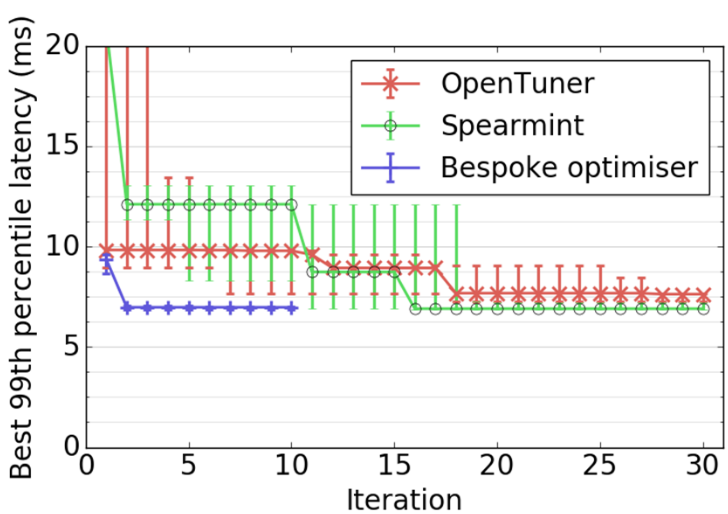
Evaluation: Garbage collection



47

47

Evaluation: Garbage collection

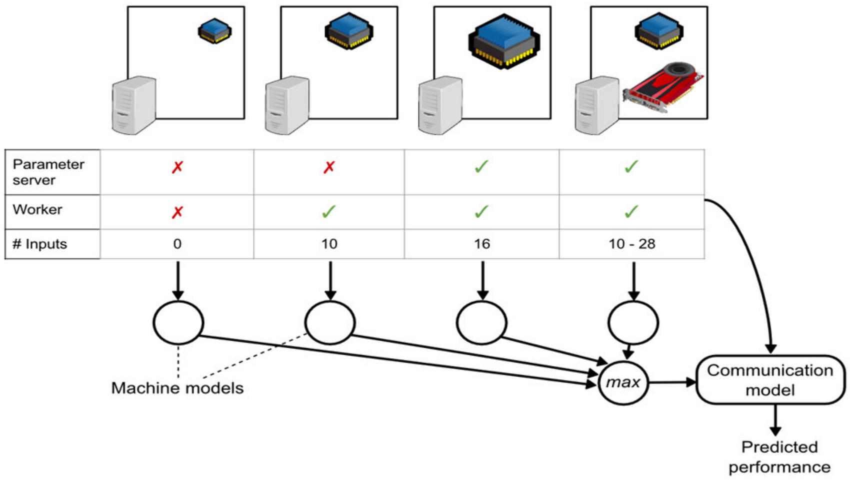


48

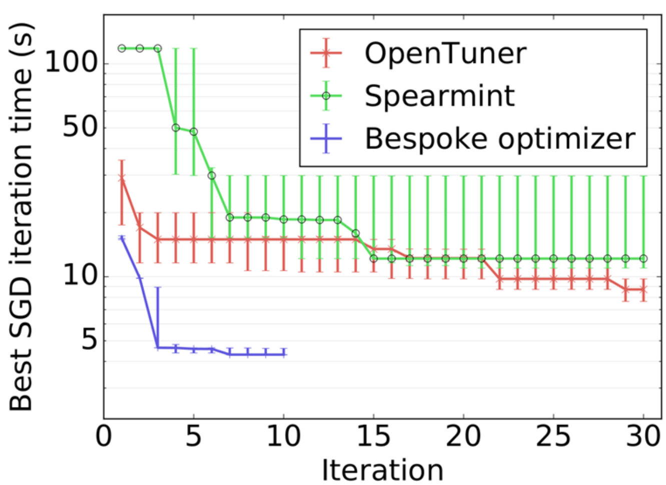
48

Distributed Scheduling of Neural Networks (SGD)

- Tune scheduling over 10 machines, setting ~30 parameters (e.g. ~10^{53} possible valid configurations)



Evaluation: Neural network scheduling



Default configuration: 9.82s
OpenTuner: 8.71s
BOAT: **4.31s**

Existing systems don't converge!

Generic Auto-Tuning with DAG Models

- Manual Tuning
  - User to learn expert knowledge and not transferable
  - e.g. Ottertune (manually selects limited number of parameters then use BO)
- Automated Tuning
  - Divide-and-diverge sampling to explore the configuration space, and recursive-bound-and-search to exploit the most promising areas
  - Use of Gaussian processes, but show that it struggles to make accurate performance predictions because of Spark's high dimensionality

→ Generic Auto-Tuning with DAG models
  - Use of DAG models for surrogate model, which mitigates the curse of dimensionality while also retaining all configurable variables
  - Exploit data analysis to identify parameter dependencies
  - Automatic building of DAG models: use of Bayesian Networks
  - Integration to BoTorch (i.e. support in Pyro as PPL)

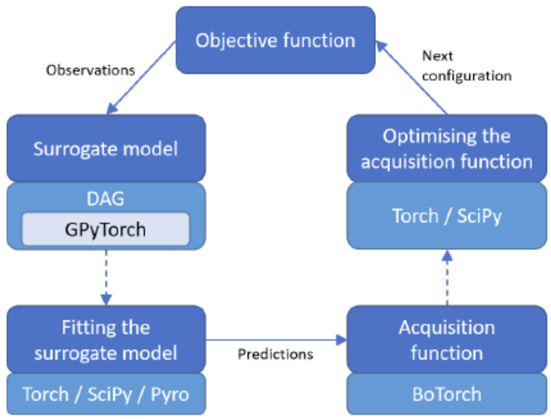
Surrogate Model in Bayesian Optimisation

Table 2.1: Comparison of surrogate models for BO

Model	Advantages	Disadvantages
Parametric models	• Quickly fit long-distance trends	• Require known structure of f
Gaussian processes [38]	• Expressive • Flexible	• Fitting is $O(n^3)$ in train-data size [40] • Continuous, non-hierarchical configuration space only
Tree-Parzen estimators [7]	• Fitting is $O(n)$ in train-data size • Categorical and hierarchical configuration space supported	• Less sample efficient than GP [41]
Random forests [29]	• Computationally very cheap • Categorical and hierarchical configuration space supported	• Inaccurately extrapolates uncertainty [40]

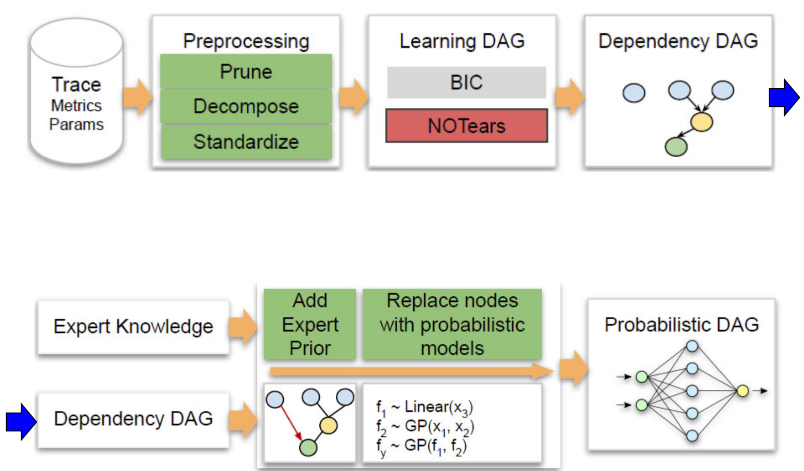
- Structural information (e.g. DAG model) improves Optimisation.

DAG model integration to BoTorch



53

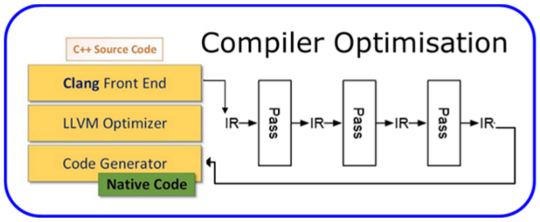
Automation of DAG model building

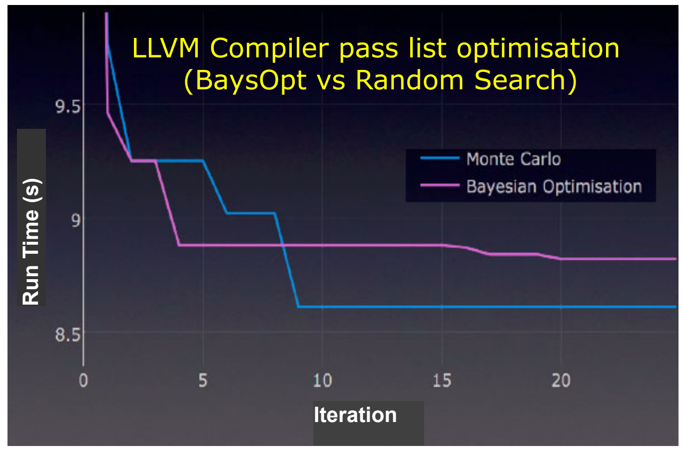


- System's inherent structure builds dependency graphs
- Decomposability allows tuning a larger number of parameters, providing interpretable optimisation suggestions
- Natural way to encode expert knowledge in its graph

54

Bayesian Optimisation not for Combinatorial Model





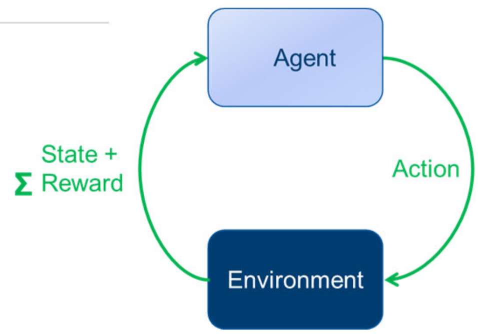
Reinforcement Learning for Optimisation

Many problems in systems are sequential Decision Making and/or Combinatorial Problems on graph data

- Compiler Optimisation
  - Input: XLA/HLO graph
  - Objective: Scheduling fusion of ops
- Chip placement
  - Input: Chip netlist graph
  - Objective: placement on 2D of ND grids
- Datacentre resource allocation
  - Input: job - workload graph
  - Objective: Placement on datacentre cells and racks
- Packet Classification
  - Input: network packets
  - Objective: minimise the classification time and memory footprint
- Network congestion control with multiple connections
- Wide range of signals to make decisions (e.g., VM allocation)
- Database: Query optimiser, Dynamic indexing…

Reinforcement Learning

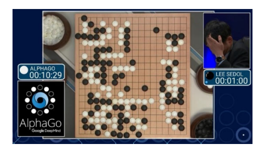
- Agent interacts with Dynamic environment
- Goal: Maximise expectations over rewards over agent's lifetime
- Notion of Planning/Control, not single static configuration



What makes RL different from other ML paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential)
- Agent's actions affect the subsequent data it receives

Model-free and Model-based RL



57

57

A brief history of Deep Reinforcement Learning Tools

Gen (2014-16): Loose research scripts (e.g. DQN), high expertise required, only specific simulators

Gen (2016-17): OpenAI gym gives unified task interface, reference implementations

- Good results on some environments (e.g. game), difficult to retool to new domains and execution modes
- Abstractions/Libraries: not fully reusable, customised towards game simulators
- High implementation risk: lack of systematic testing, performance strongly impacted by noisy heuristics

Gen (2017-18): Generic declarative APIs, distributed abstractions (Ray Rllib, RLGraph), some standard *flavours* emerge

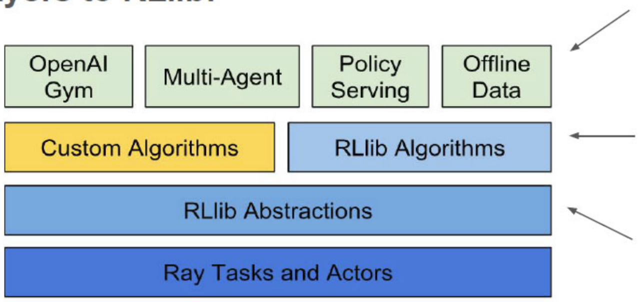
Still Problems... Tightly coupled execution/logic, testing, reuse...

58

58

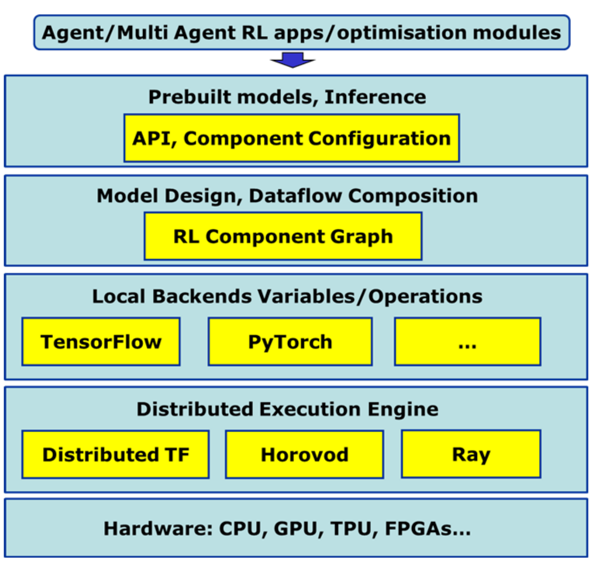
RLlib (UC Berkeley) Architecture

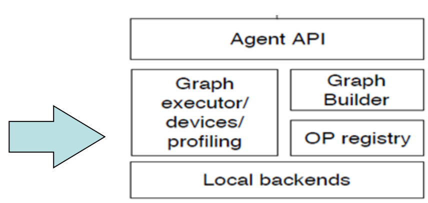
User perspective: three main layers to RLlib:



1. APIs that make RL accessible to a variety of applications
2. Collection of best-in-class reference algorithms
3. Primitives for implementing new RL algorithms

RLgraph: Modular Dataflow Composition





- ... is a programming model to design and execute RL algorithms across execution paradigms
- ... generates incrementally testable, transparently configurable code through a staged build process

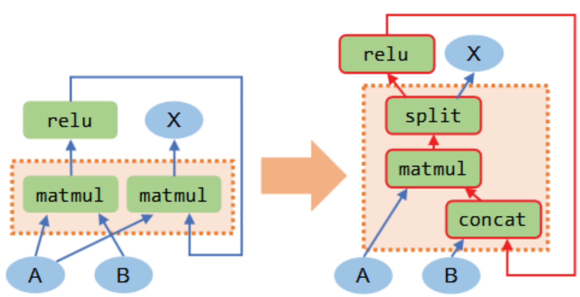
RL in Computer Systems: Practical Considerations

- Action spaces do not scale well:
  - Systems problems often combinatorial
- Exploration in production system not a good idea
  - Unstable, unpredictable
- Simulations can oversimplify problem
  - Expensive to build, not justified versus gain
- Unlike supervised learning: Not single dominant execution pattern
- Algorithms highly sensitive to hyper-parameters

- Online steps take too long

Optimising DNN Computation with Graph Substitutions

- TASO (SOSP, 2019): Performance improvement by transformation of computation graphs
- In progress: use of Reinforcement Learning

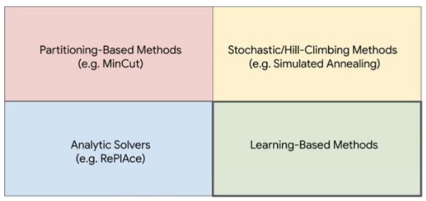


(a) before substitution (b) after substitution

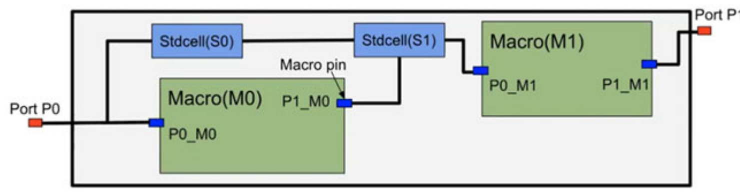
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Chip Placement with Reinforcement Learning

- A. Mirhoseini and A. Goldie: Chip Placement with Deep Reinforcement Learning, ISPD, 2020.



- A form of graph resource optimization
- Place the chip components to minimize the latency of computation, power consumption, chip area and cost, while adhering to constraints, such as congestion, cell utilization, heat profile, etc.



63

63

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Summary: Massive Data Processing and Optimisation

→Dataflow is key to improve performance

→Parameter space is complex, large and dynamic/combinatorial

- Systems are nonlinear and difficult to model manually → Exploit ML
- Reinforcement Learning to optimise dynamic combinatorial problem
- Key concept behind is Dataflow (~=Graph) structural transformation/Decomposition

→Exploit structural information for model decomposition to accelerate optimisation process

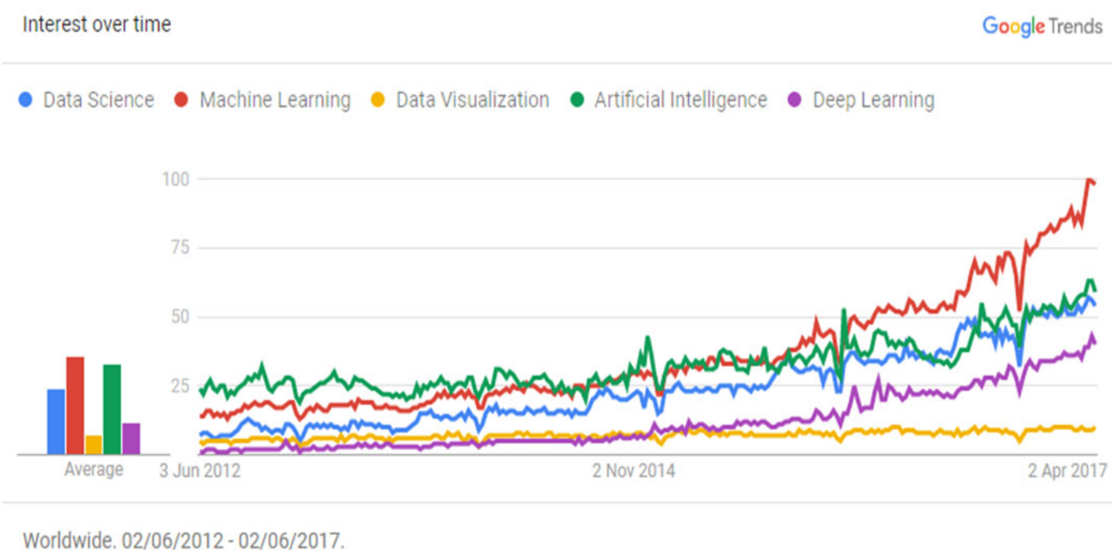
→Bayesian Optimisation and Reinforcement Learning are key

64

64

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Rise of Data Science

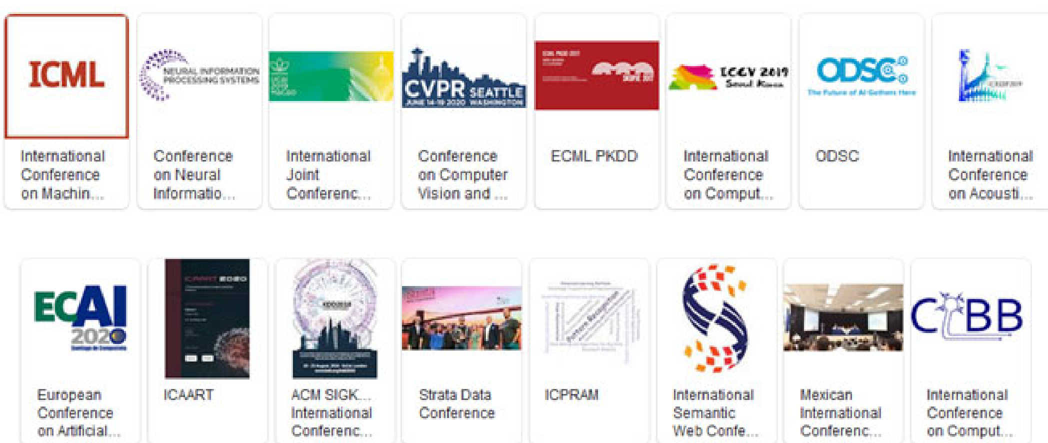


65

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Machine Learning Conferences

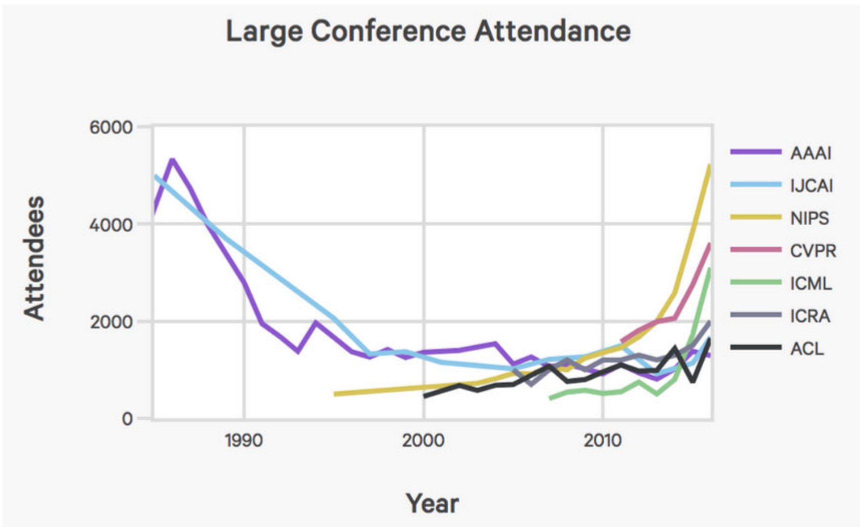
- Large ML research community runs large conferences



66

UNIVERSITY OF CAMBRIDGE Computer Laboratory

Scale of Community Size in ML/AI



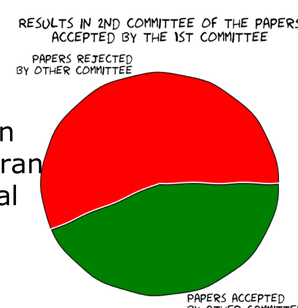
67

UNIVERSITY OF CAMBRIDGE Computer Laboratory

NIPS/NEURIPS: 8000 Attendees in 2017

Randomness of Paper acceptance?

- 2016: 2,406 submissions and 568 acceptance (24% acceptance rate)
- 2017: 3,240 submissions and 679 acceptance (21% acceptance rate)
- 2020: 9,467 submissions and 1,990 acceptance (20% acceptance rate)
- In 2014, Corinna Cortes and Neil Lawrence ran the NIPS experiment where 1/10th of papers submitted to NIPS went through the NIPS review process twice, and then the accept/reject decision was compared. http://blog.mrtz.org/2014/12/15/the-nips-experiment.html
- In particular, about 57% of the papers accepted by the first committee were rejected by the second one and vice versa. In other words, most papers at NIPS would be rejected if one reran the conference review process (with a 95% confidence interval of 40-75%).
- Check out newer paper on this topic: https://arxiv.org/pdf/2109.09774.pdf



68

UNIVERSITY OF CAMBRIDGE Computer Laboratory

MLSys Conference spawn in 2018-2019

- MLSys (originally SysML) is a conference targeting research at the intersection of systems and machine learning

  https://mlsys.org
- 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

Attendance Question!

Steering Committee

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

UNIVERSITY OF CAMBRIDGE Computer Laboratory

Gap between Research and Practice

Device Placement Optimization with Reinforcement Learning

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]

20H with 80GPUs!

70