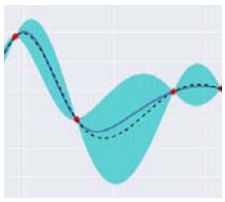
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Large-scale Data Processing and Optimisation



Eiko Yoneki

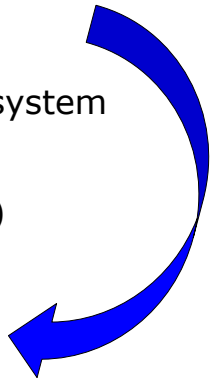
University of Cambridge Computer Laboratory

1

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

2

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Technologies supporting Cluster Computing

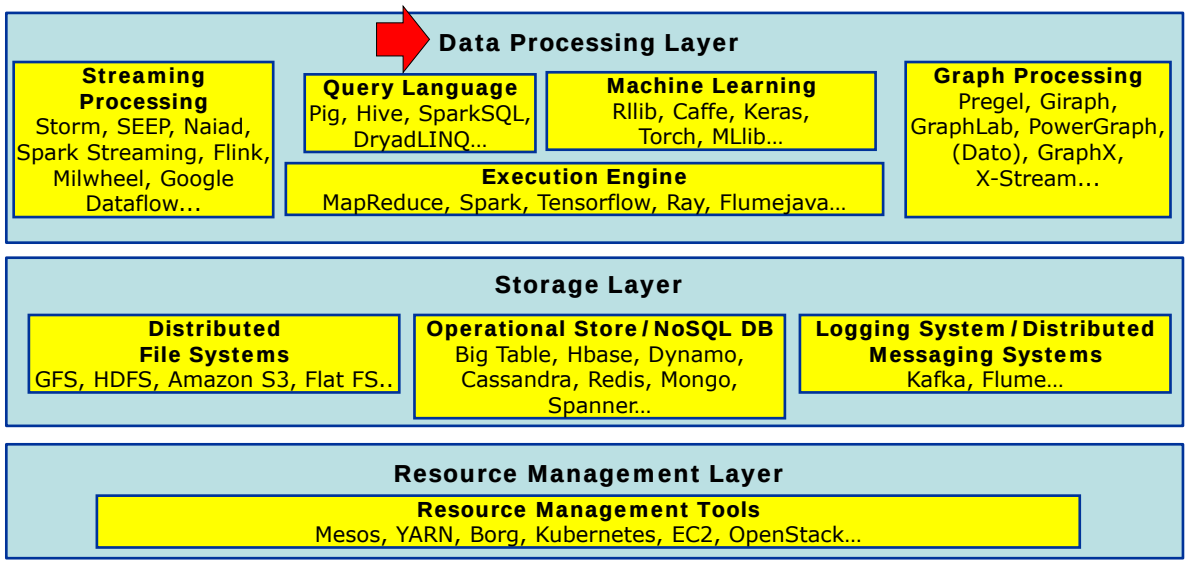
- Distributed infrastructure
  - Cloud (e.g. Infrastructure as a service, Amazon EC2, GCP, Azure)

    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

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Data Processing Stack

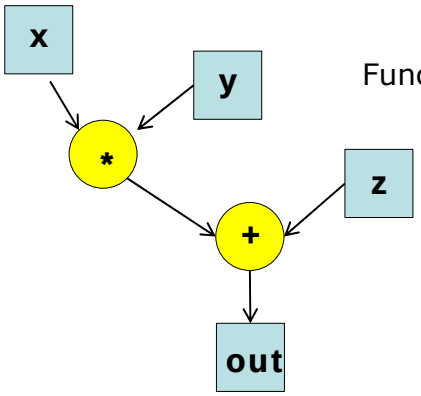


4

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Data Flow Programming

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



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

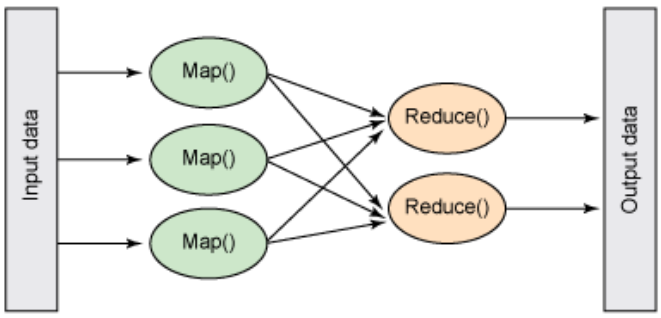
5

5

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)



6

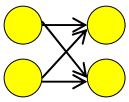
6

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Data Flow Programming Examples

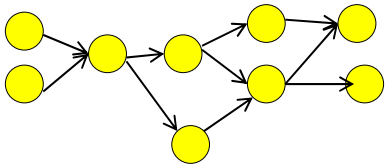
- Data (flow) parallel programming
  - e.g. MapReduce, Dryad/LINQ, NAIAD, Spark, Tensorflow…

MapReduce: Hadoop

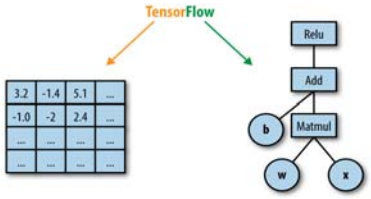


Two-Stage fixed dataflow

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



More flexible dataflow model

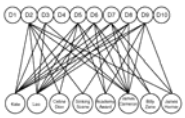


7

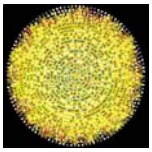
7

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Emerging Massive-Scale Graph Data



Bipartite graph of phrases in documents



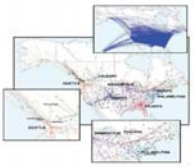
Protein Interactions [genomebiology.com]

Gene expression data

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

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

Web 1.4B pages(6.6B links)



Airline Graphs

Social media data

8

8

UNIVERSITY OF CAMBRIDGE Computer Laboratory

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

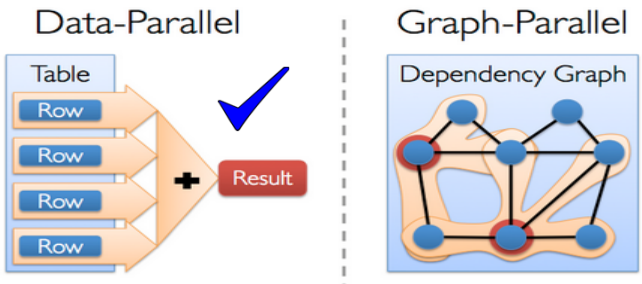
9

9

UNIVERSITY OF CAMBRIDGE Computer Laboratory

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

10

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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..

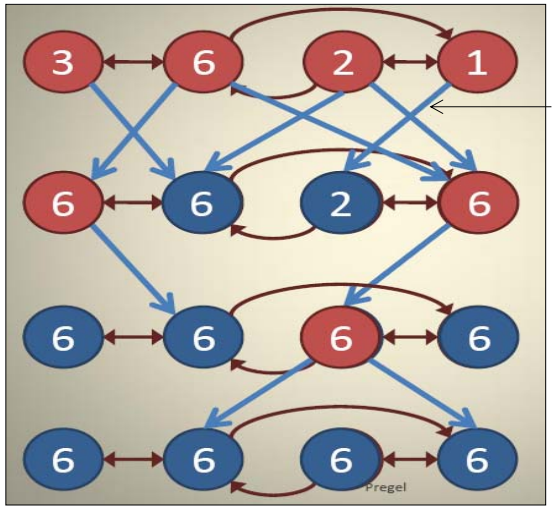
11

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bulk synchronous parallel: Example

- Finding the largest value in a connected graph

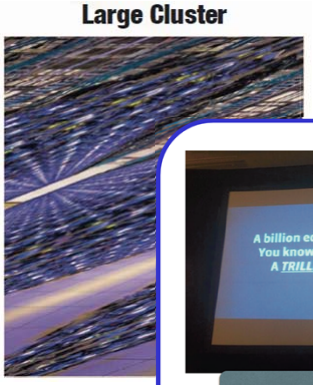
Local Computation
Communication
Local Computation
Communication
...



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



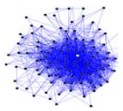
HPC/Graph500 benchmarks (June 2014)

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

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

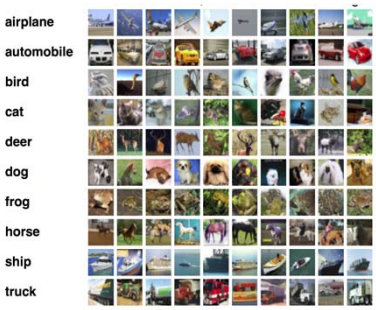
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Data Processing for Neural Networks

- Practicalities of training Neural Networks
- Leveraging heterogeneous hardware

Modern Neural Networks Applications:

Image Classification



Reinforcement Learning

15

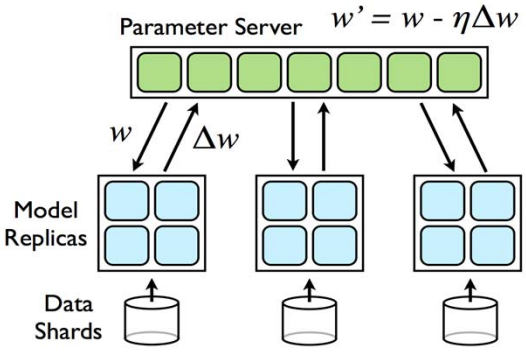
15

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Performance Improvement

- One or more beefy GPUs

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



16

16

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

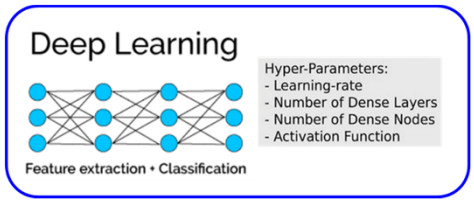
17

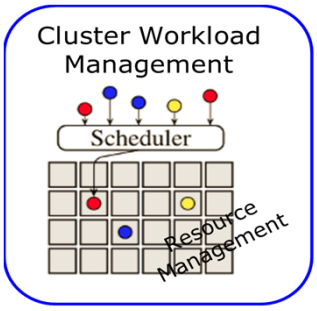
17

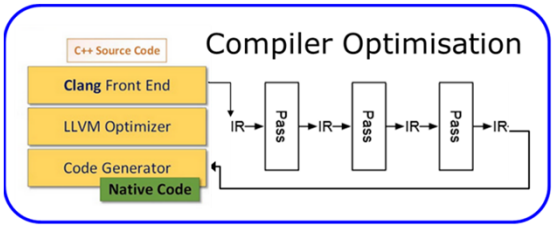
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

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

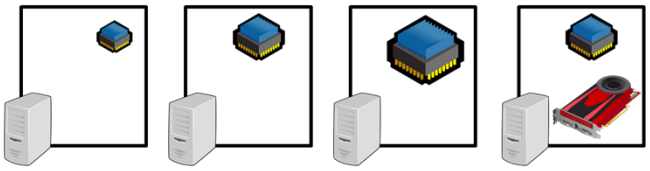
20

20

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

21

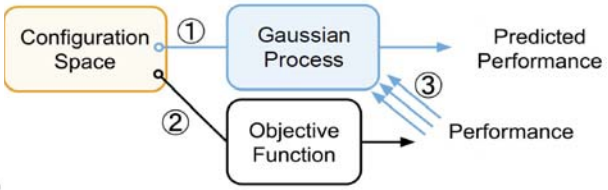
21

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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 $a()$
1: **for** $i = 1, 2, \ldots$ **do**
2: Sample point: $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x}} a(G, \boldsymbol{x})$
3: Evaluate new point: $y_t \leftarrow f(\boldsymbol{x}_t)$
4: Update surrogate distribution: $G \leftarrow G \mid (\boldsymbol{x}_t, y_t)$
5: **end for**



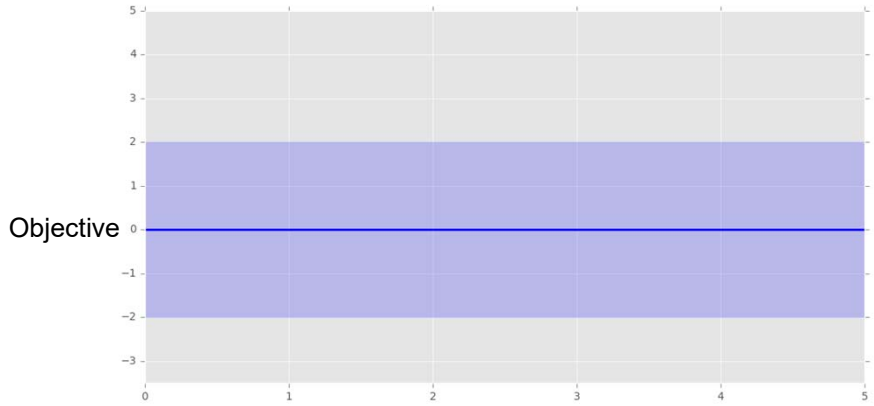
① Find promising point (high performance value in the model)
② Evaluate the objective function at that point
③ Update the model to reflect this new measurement

22

22

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

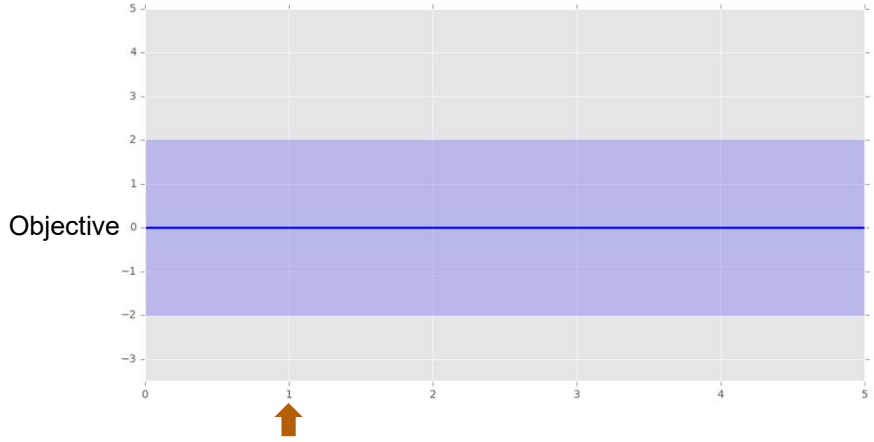
Bayesian Optimisation



23

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

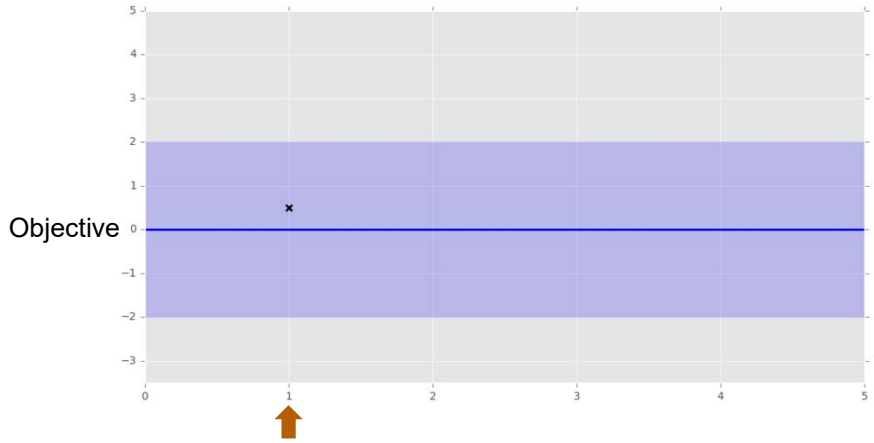
Bayesian Optimisation



24

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

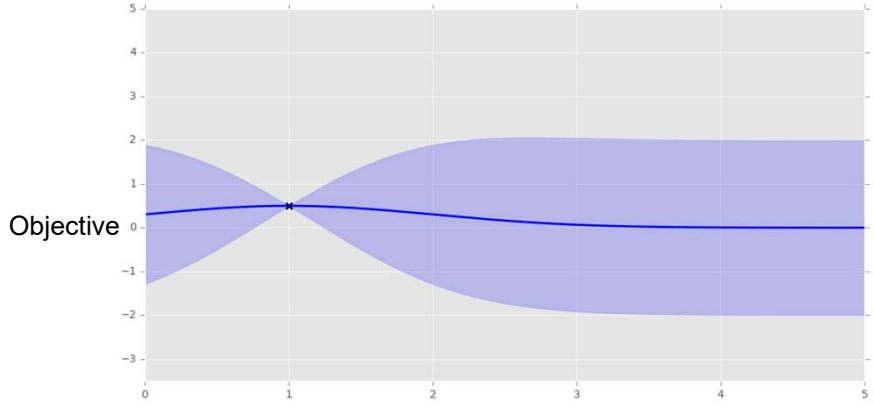
Bayesian Optimisation



25

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

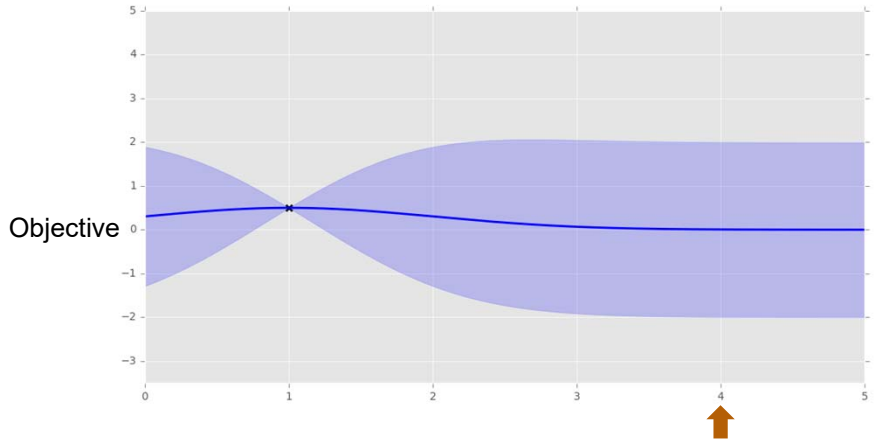
Bayesian Optimisation



26

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

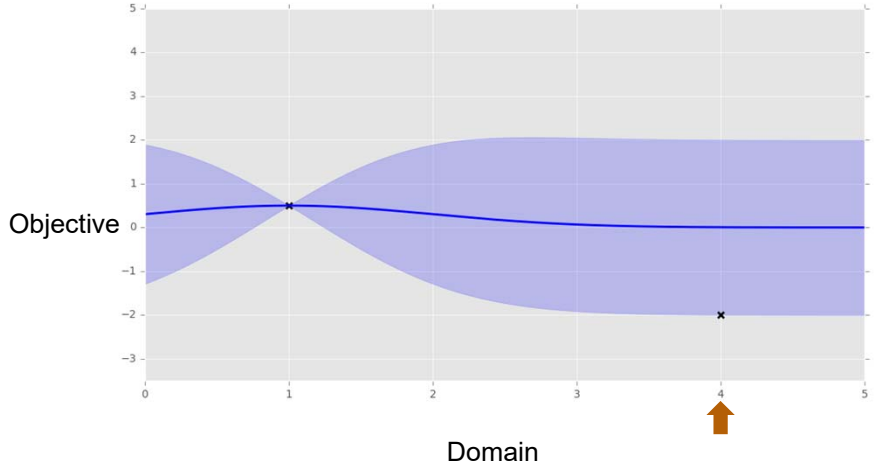
Bayesian Optimisation



27

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

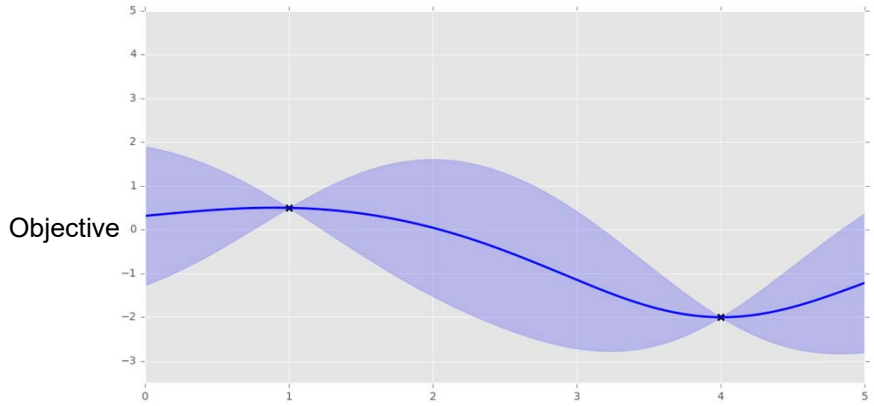
Bayesian Optimisation



28

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

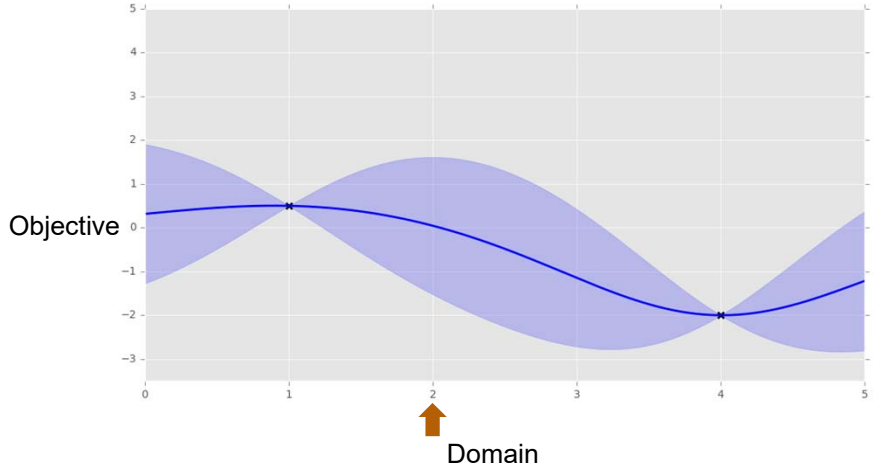
Bayesian Optimisation



29

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

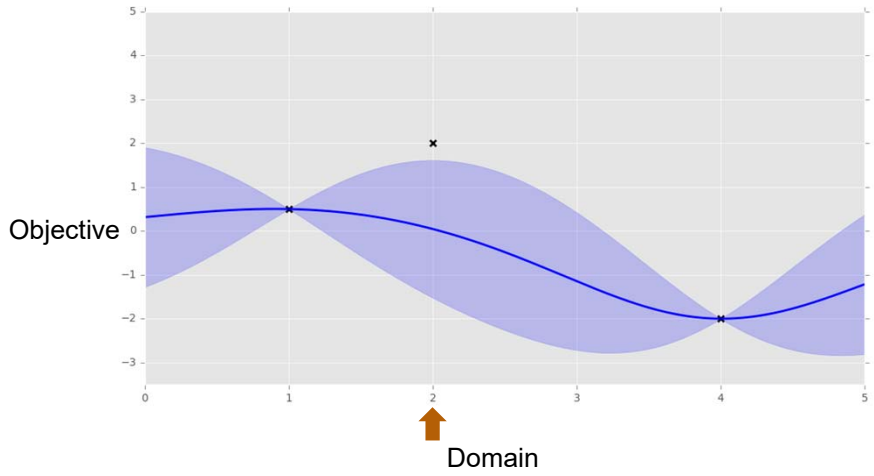
Bayesian Optimisation



30

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

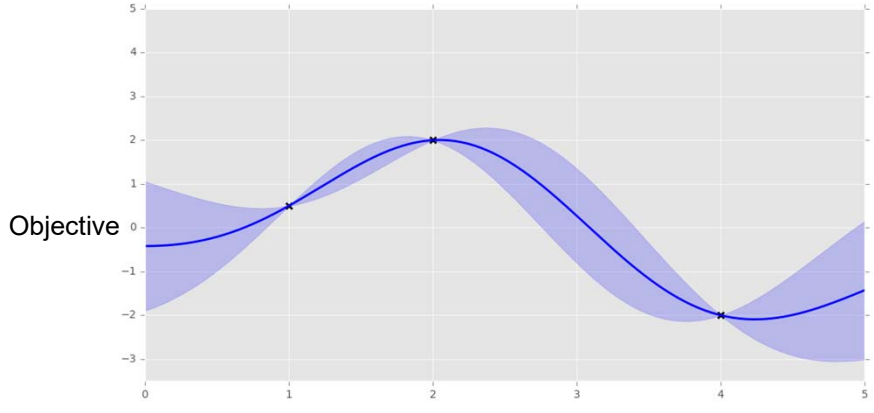
Bayesian Optimisation



31

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

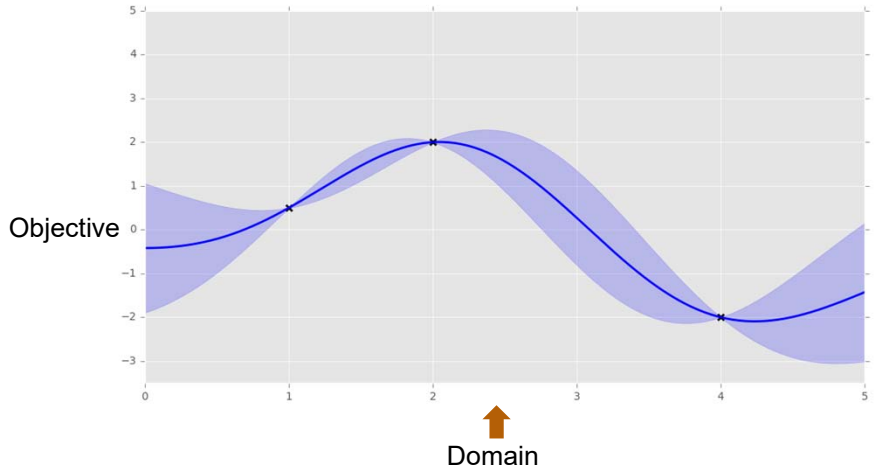
Bayesian Optimisation



32

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

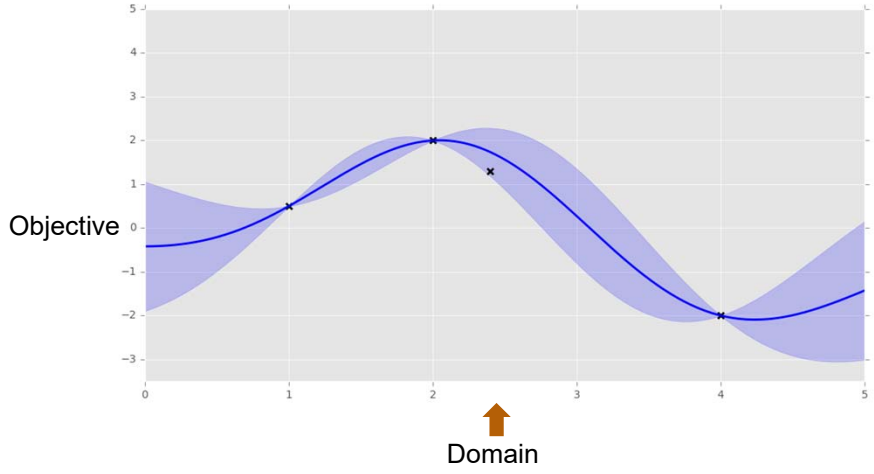
Bayesian Optimisation



33

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

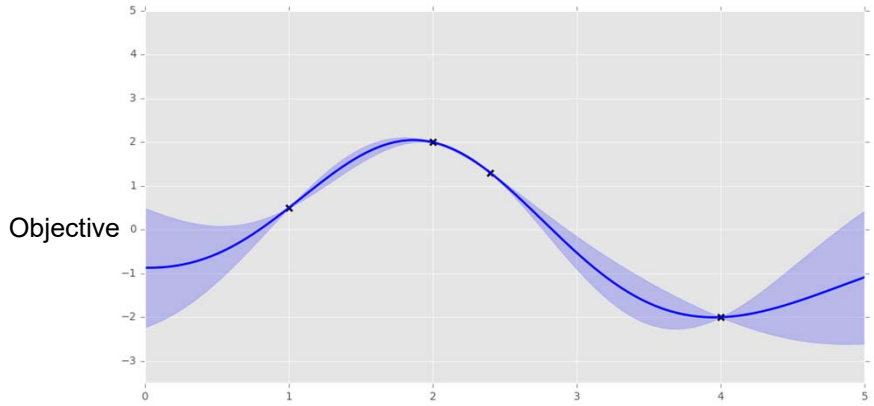
Bayesian Optimisation



34

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

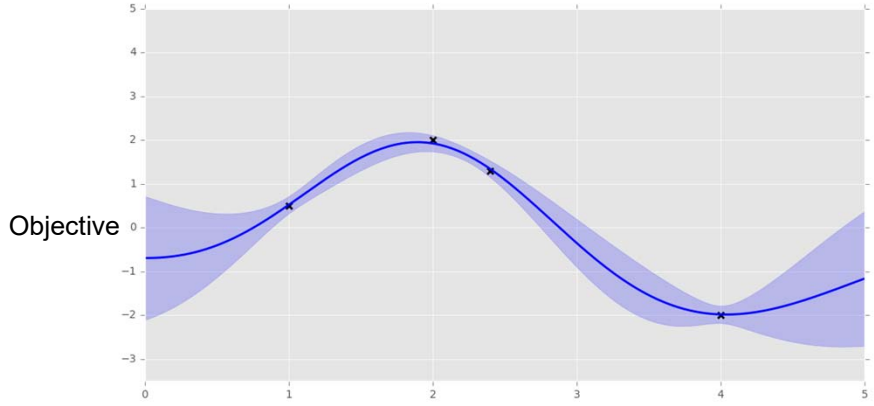
Bayesian Optimisation



35

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bayesian Optimisation



36

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Further Bayesian Optimisation…

- BO overview/Tutorial
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2020_2021/aid/BO_overview_Archambeau.pdf
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2020_2021/aid/BO_overview_adams.pdf
  - https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2020_2021/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.

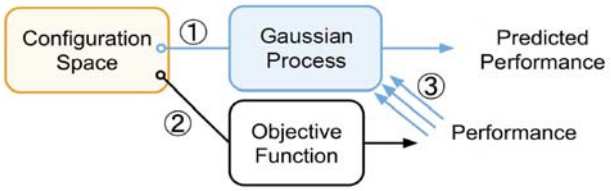
37

37

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

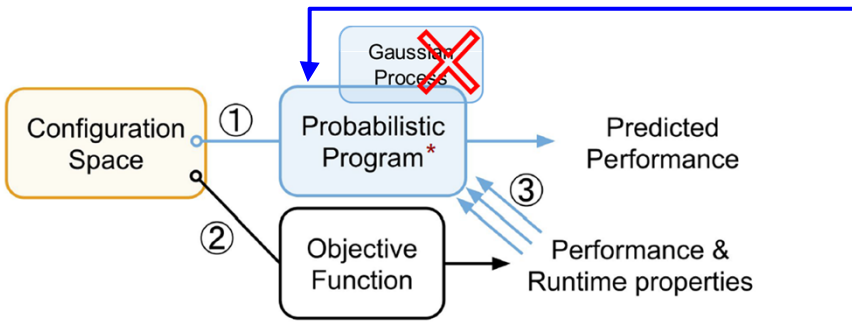
Cons:
- ✗ In many dimensions, model does not converge to the objective function

38

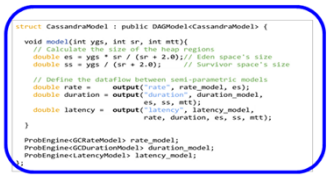
38

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Structured Bayesian Optimisation (SBO)



Probabilistic Model written in Probabilistic C++



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

✓ Better convergence
✓ Use all measurements

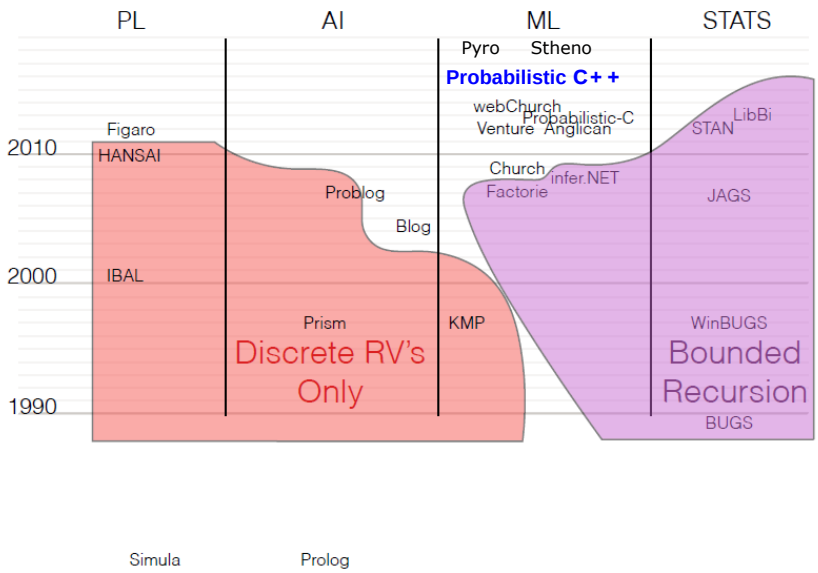
BOAT: a framework to build **B**esp**O**ke **A**uto-**T**uners

39

39

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Probabilistic Programming: Probabilistic C++

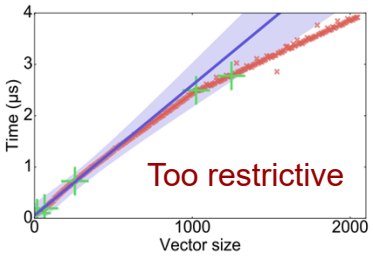


40

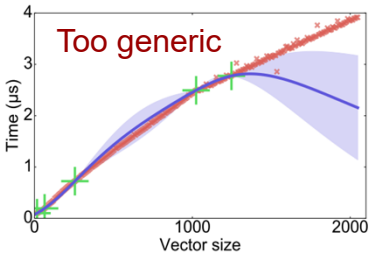
40

Semi-parametric Model

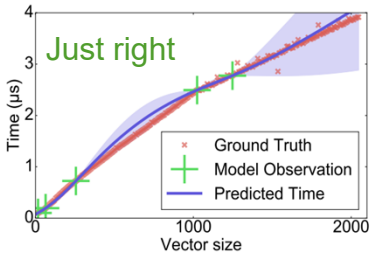
- Easy to use and well suited to SBO
  - Understand general trend of Objective function
  - High precision in region of optimum for finding highest performance



(a) Parametric (Linear regression)



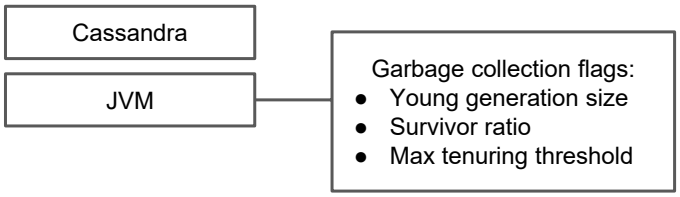
(b) Non-parametric (Gaussian process)



(c) Semi-parametric (Combination)

Example: JVM Garbage Collection

- Cassandra's garbage collection

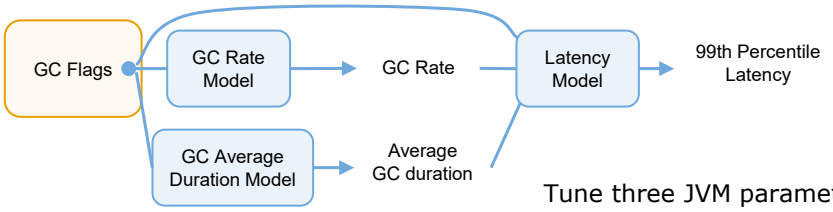


- Minimise 99th percentile latency of Cassandra

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

43

43

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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;
};
```

44

44

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

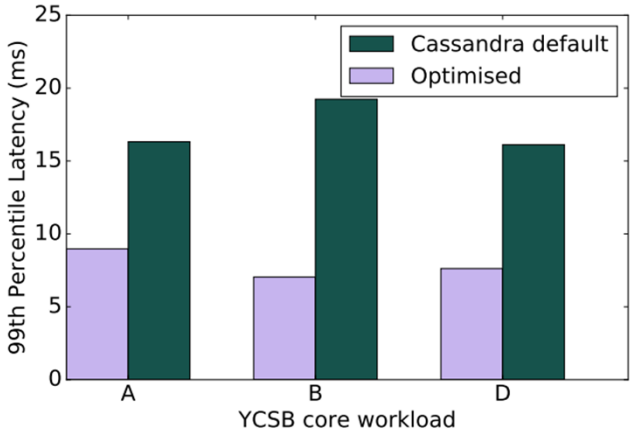
GC Rate Semi-parametric model

```
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;
};
```

45

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

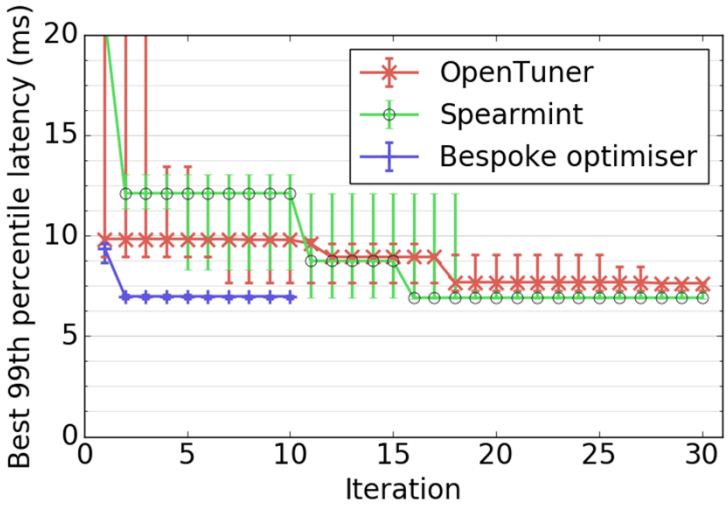
Evaluation: Garbage collection



46

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Evaluation: Garbage collection



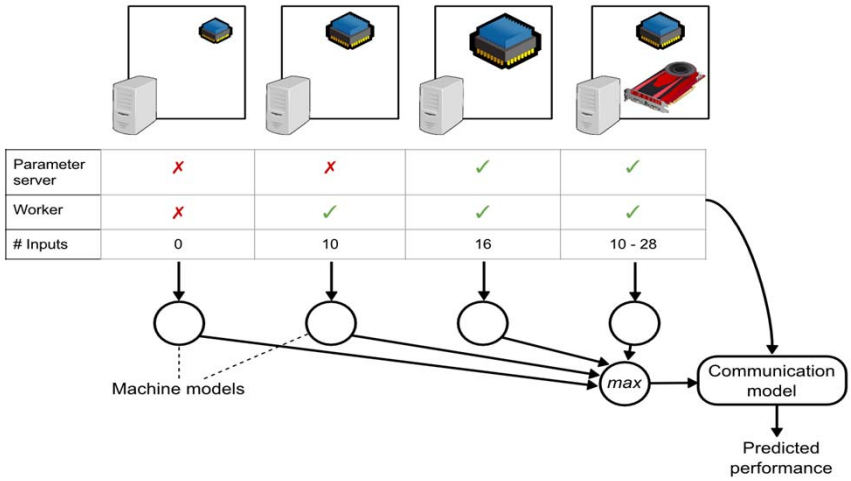
47

47

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Distributed Scheduling of Neural Networks (SGD)

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

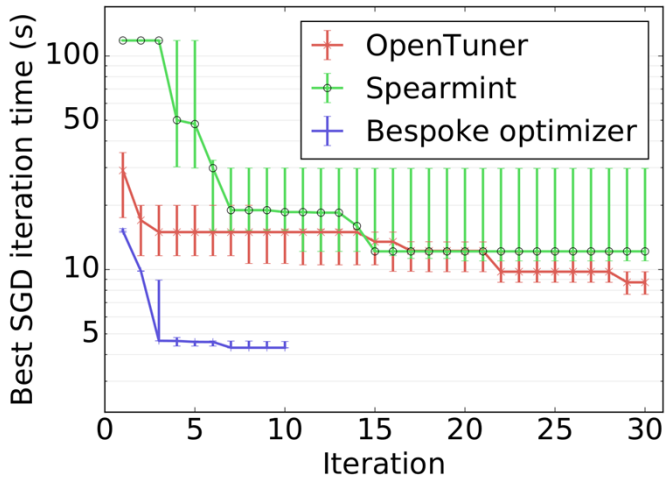


48

48

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Evaluation: Neural network scheduling



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

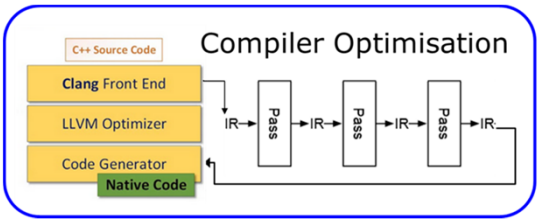
Existing systems don't converge!

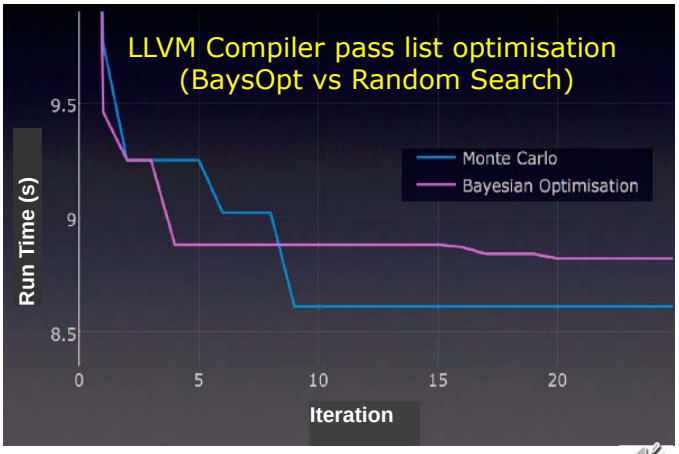
49

49

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Bayesian Optimisation not for Combinatorial Model





50

50

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Reinforcement Learning for Optimisation

Many problems in systems are ssequential 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…

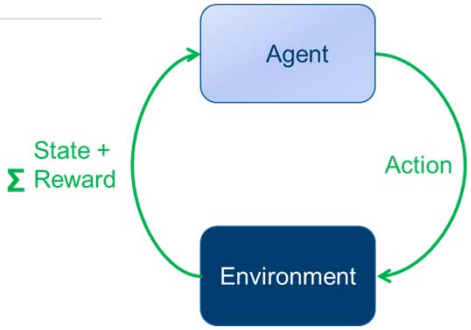
51

51

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

52

52

UNIVERSITY OF CAMBRIDGE Computer Laboratory

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

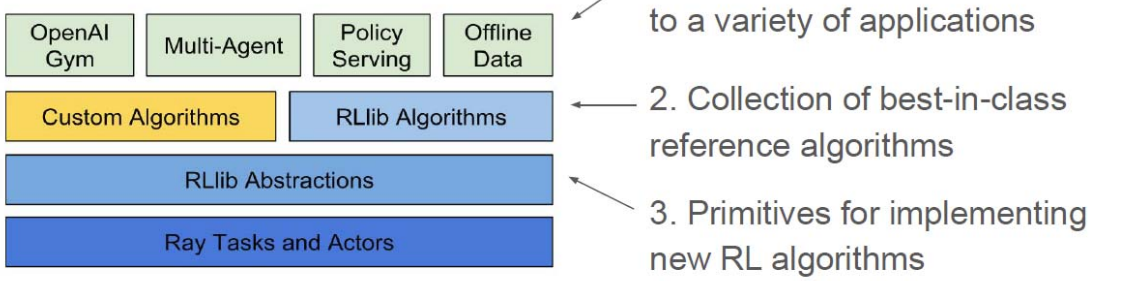
Still Problems… Tightly coupled execution/logic, testing, reuse...

53

UNIVERSITY OF CAMBRIDGE Computer Laboratory

RLlib (UC Berkeley) Architecture

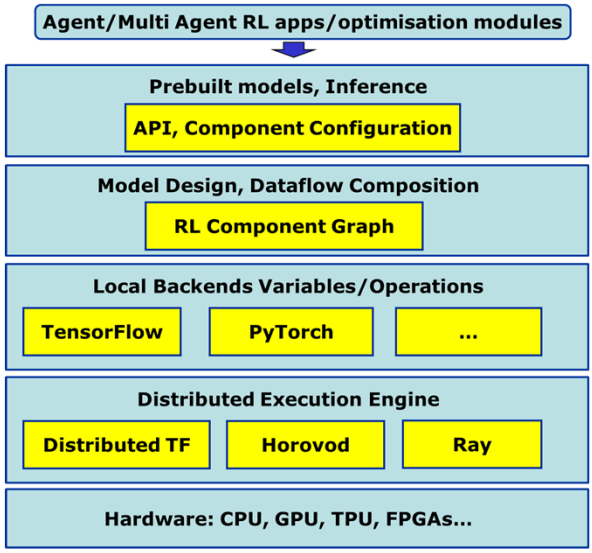
User perspective: three main layers to RLlib:

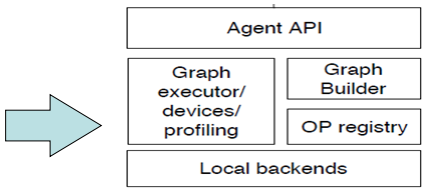


54

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

55

55

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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

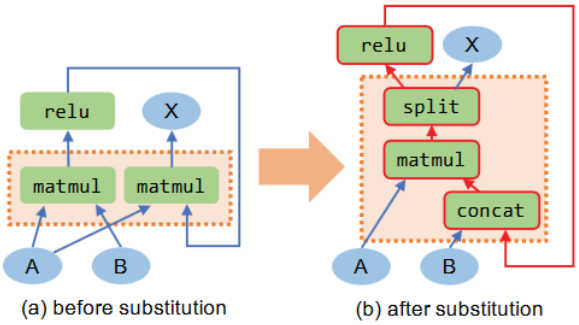
56

56

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Optimising DNN Computation with Graph Substitutions

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



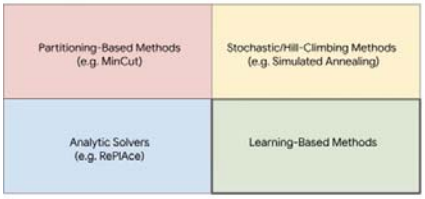
57

57

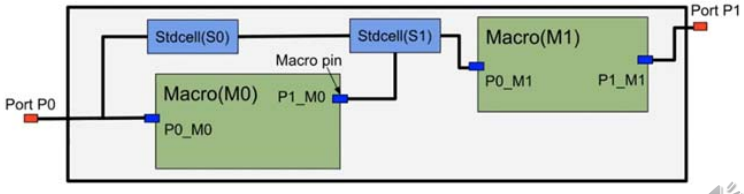
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

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.



58

58

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

59

59

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Gap between Research and Practice

Device Placement Optimization with Reinforcement Learning

20H with 80GPUs!

Azalia Mirhoseini [*12] Hieu Pham [*12] 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]

60

60