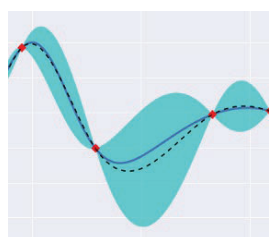


Large-scale Data Processing and Optimisation

Overview

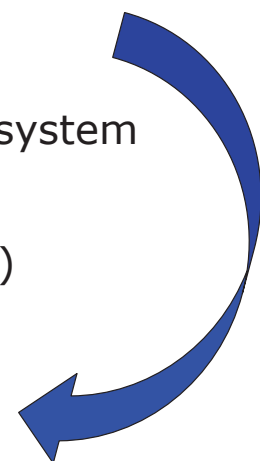
Eiko Yoneki

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)



Initial Technologies supporting Cluster Computing

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

Data Processing Stack

APPLICATION & INTERFACE

SQL, analytics

Trino · Spark SQL · DuckDB · BigQuery
Pig Hvie DryadLINQ

ML / AI & agentic systems

PyTorch · JAX · vLLM · LangGraph ·
AutoGen RLLib

Pipelines & workflow orchestration

Airflow · Dagster · Argo workflows

DATA PROCESSING & COMPUTE

Batch & DAG execution

Spark · Ray Data · Dask
MapReduce (foundational)

Streaming & dataflow

Flink · Kafka Streams · Beam
Managed: Dataflow, Naiad, Flink

Graph & iterative computation

GraphX · GraphScope · Pregel-style GraphLab
Powergraph X-Stream, Giraph
Graph databases / GNN frameworks

DATA & EVENT PLANE

Object & file storage

Amazon S3 · GCS · ADLS · HDFS
Parquet · Arrow

Lakehouse & tables

Iceberg · Delta Lake · Hudi
Catalogs / metadata

Databases & Logging System

Spanner · Cassandra · Redis
Kafka · Pulsar · CDC

INFRASTRUCTURE & RESOURCE MANAGEMENT ORCHESTRATION

Cluster & resource management

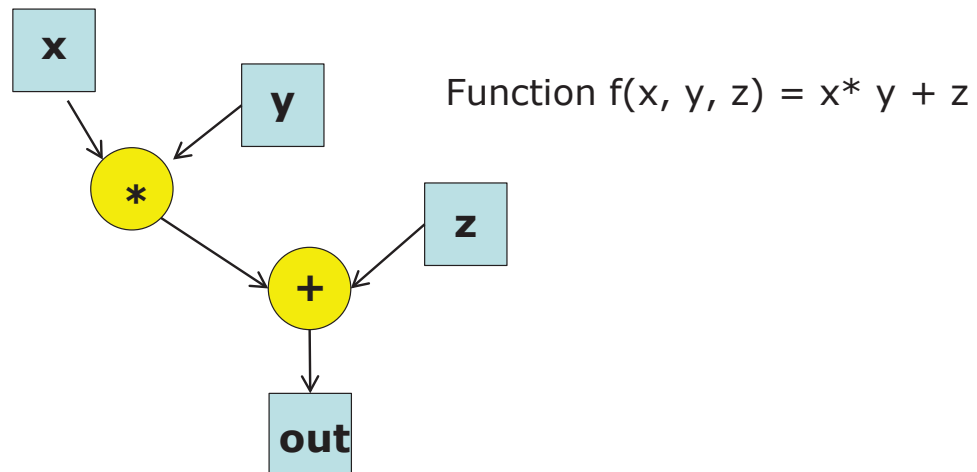
Kubernetes · YARN (legacy) · Slurm · Ray clusters

Cloud & hardware

AWS · Azure · GCP · GPUs / TPUs · networking

Data Flow Programming

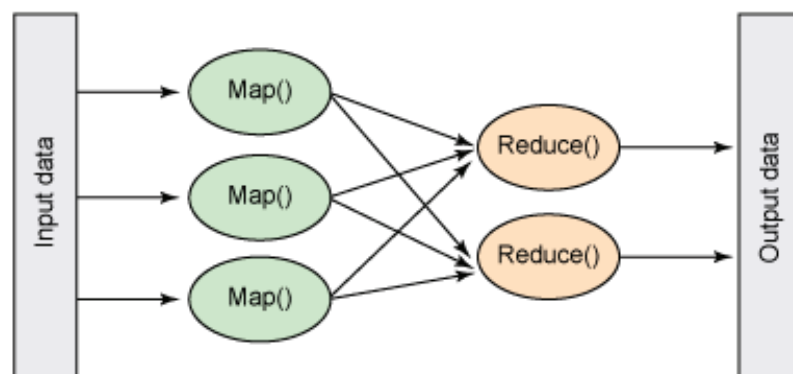
- Non-standard programming models
- Powerful abstraction: mapping computation into dataflow graphs



5

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)



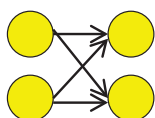
6

Data Flow Programming Examples

- Data (flow) parallel programming

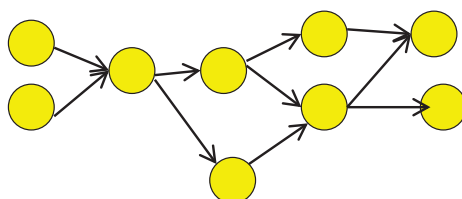
- e.g. MapReduce, Dryad/LINQ, NAIAD, Spark, Tensorflow...
- e.g. ML compiler: computation graphs

MapReduce:
Hadoop

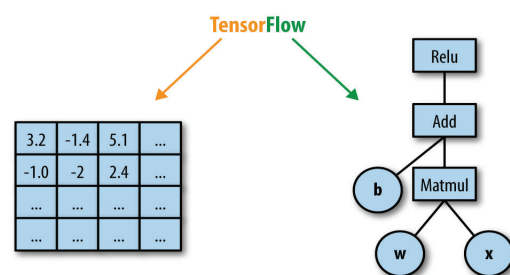


Two-Stage fixed dataflow

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



More flexible dataflow model



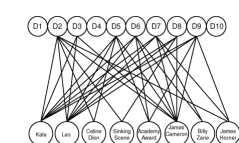
7

Data Flow Programming's Usefulness for ML

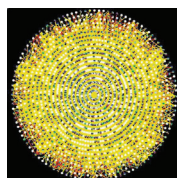
- Data flow programming holds the key to natural, modular, streamlined ML specification
- Same language for layer, network, and integration specification
 - For complex NN classes, RL with complex reward, gated networks, attention/transformers, mix of NN and data processing
- Same language for ML algorithmics and compiler design
- Well-established analysis and compilation techniques integrated within existing ML compilers (e.g. MLIR)
 - MLIR (Multi-Level Intermediate Representation) is intended to be a hybrid IR which can support multiple different requirements in a unified infrastructure (<https://mlir.llvm.org/>)

8

Emerging Massive-Scale Graph Data



Bipartite graph of
phrases in
documents



Protein Interactions
[genomebiology.com]



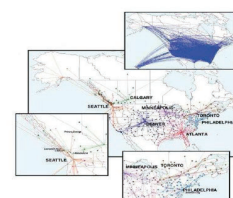
Gene expression
data



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



Web 1.4B
pages(6.6B
links)



Airline Graphs



Social media data

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

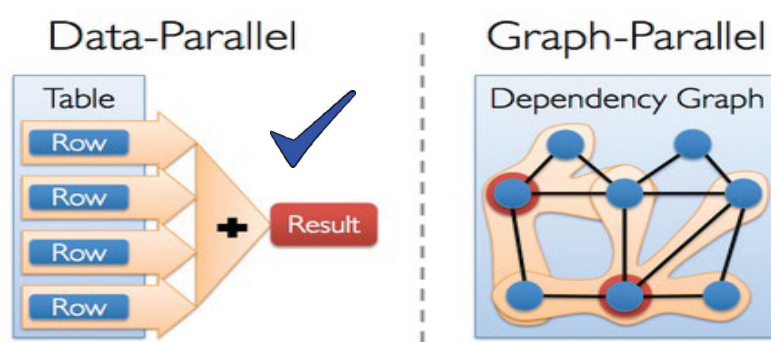
Graph Computation Challenges

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



11

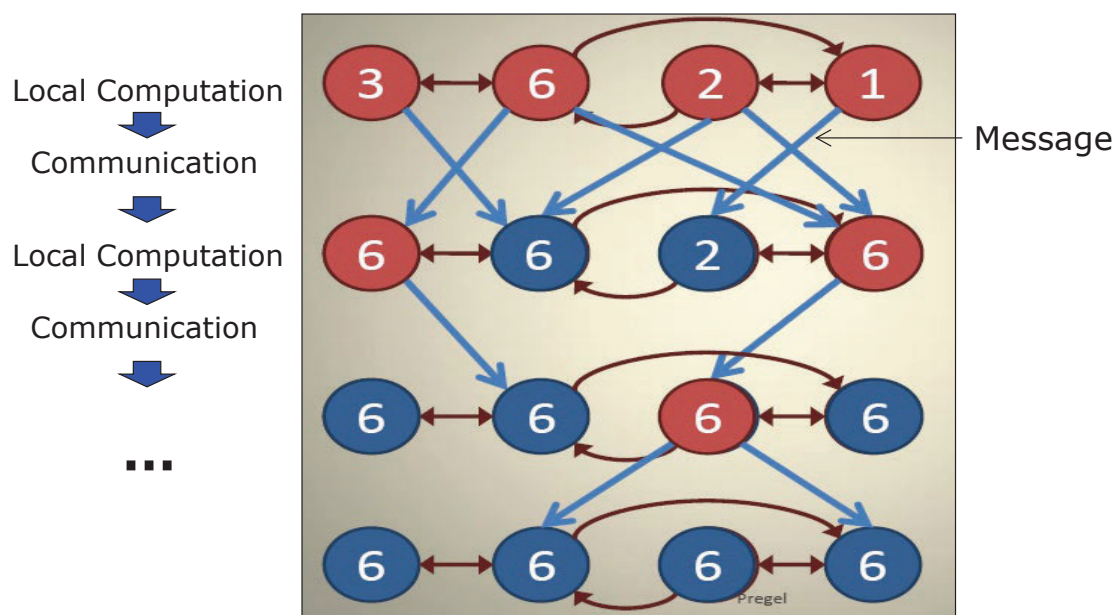
Graph-Parallel

- 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...

12

Bulk synchronous parallel: Example

- Finding the largest value in a connected graph



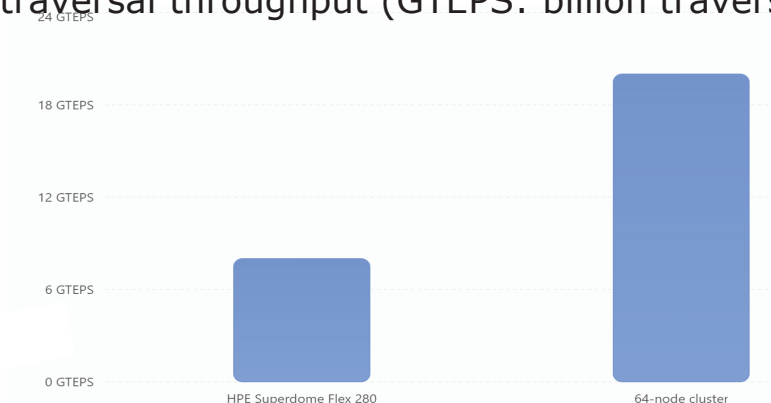
13

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

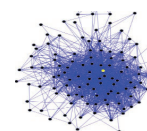
HPE Superdome Flex 280 vs. 64-node commodity cluster

(Graph500 BFS, graph traversal throughput (GTEPS: billion traversed edges per second))



14

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

Fixed-point iteration:

All vertices active in each iteration
(50% computation, 50% communication)

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

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

from Frank McSherry HotOS 2015

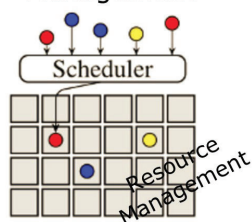
15

Optimisation/Tuning Computer Systems is Complex

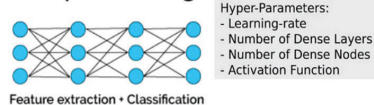
- Complex and High Dimension Parameter Space
- Expensive Objective Functions
- Hand-crafted solutions impractical, often left static or configured through extensive offline analysis
- Not well-tuned system's performance does not scale

→ Necessity of scalable Auto-tuning

Cluster Workload Management



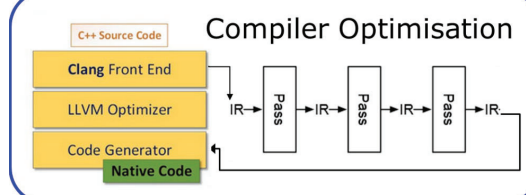
Deep Learning



SoC/ASIC SW/HW Co-Design



Compiler Optimisation

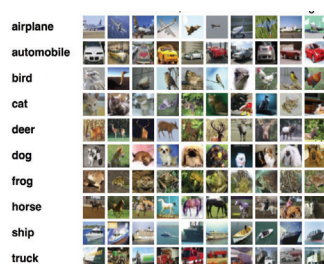


16

Data Processing Paradigm Change

- Emergence of modern Neural Networks Applications
- Practicalities of training Neural Networks
- Leveraging heterogeneous hardware
- Traditional dataflow programming does not deal with mathematical objects (no deep learning back then), now control flow requires to be numerically differentiable (i.e. TensorFlow)
- Now LLM, Agentic systems...

Image Classification



Reinforcement Learning



17

Challenging: 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)
 - Large number of parameters
 - Evaluation is slow and expensive
- **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)
- **Integrate power of Hardware (accelerator, GPU)**

Many systems problems are combinatorial in nature

18

Search Parameter Space

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



Evolution strategies: Evaluate
permutations against fitness function

SPEARMINT

Hill Climbing

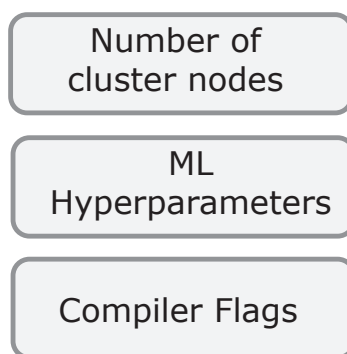


Bayes Opt: Sample efficient, requires
continuous function, some configuration

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

19

Massive Parameter Space in Large-scale Systems



Coarse

Optimise parameters θ ?

Minimise some cost function $f(\theta)$

...where cost is runtime,
memory, I/O, or combination of
all...

Fine

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

➔ **Blackbox Optimisation**



Optimisation: Iterative Operation

- Common to use an iterative global search algorithm for optimisation problem
- e.g. Bayesian optimisation algorithm that is capable of simultaneously approximating the target function that is being optimised while optimising it
- Automated machine learning (AutoML) algorithms being used to choose an algorithm, an algorithm and hyperparameters, or data preparation, algorithm and hyperparameters, with very little user intervention

21

Machine Learning and Optimisation

- Function Optimisation
 - Find the set of inputs to a target objective function that result in the minimum or maximum of the function
- Function Approximation:
 - Generalise from specific examples to a reusable mapping function for making predictions on new examples
 - ML can be described as function approximation as approximating the unknown underlying function that maps examples of inputs to outputs in order to make predictions on new data
 - Function approximation often uses function optimisation
- At the core of many ML algorithms is an optimisation algorithm!

22

Bayesian Optimisation

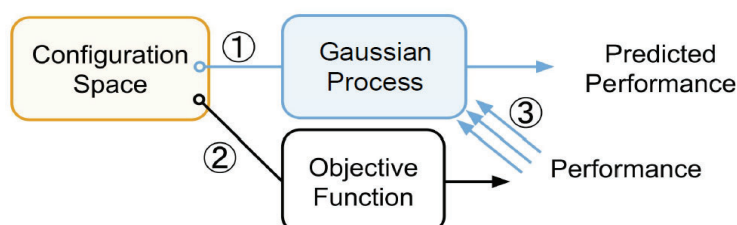
- Iteratively builds probabilistic model of objective function
- Typically Gaussian process as probabilistic model
- Data efficient: converges quickly

Input: Objective function $f()$
Input: Surrogate function initial distribution G
Input: Acquisition function $a()$

```

1: for  $i = 1, 2, \dots$  do
2:   Sample point:  $\mathbf{x}_i \leftarrow \arg \max_{\mathbf{x}} a(G, \mathbf{x})$ 
3:   Evaluate new point:  $y_i \leftarrow f(\mathbf{x}_i)$ 
4:   Update surrogate distribution:  $G \leftarrow G | (\mathbf{x}_i, y_i)$ 
5: end for

```



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

- 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

23

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	• Expressive • Flexible	• Fitting is $O(n^3)$ in train-data size • Continuous, non-hierarchical configuration space only
Tree-Parzen estimators	• Fitting is $O(n)$ in train-data size • Categorical and hierarchical configuration space supported	• Less sample efficient than GP
Random forests	• Computationally very cheap • Categorical and hierarchical configuration space supported	• Inaccurately extrapolates uncertainty

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

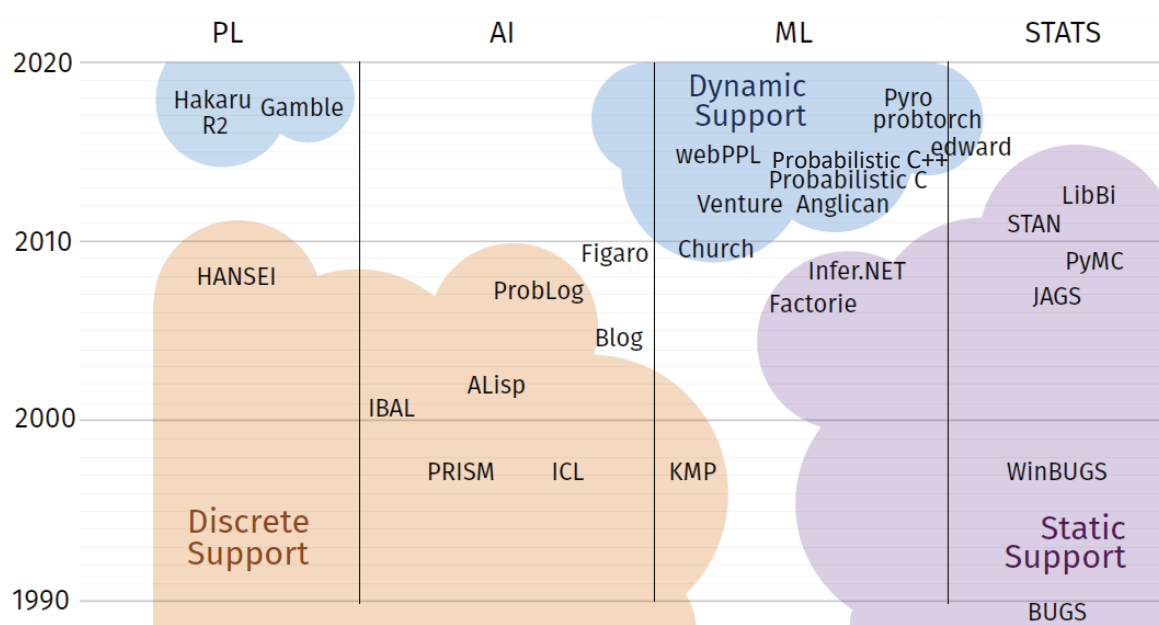
24

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...)

25

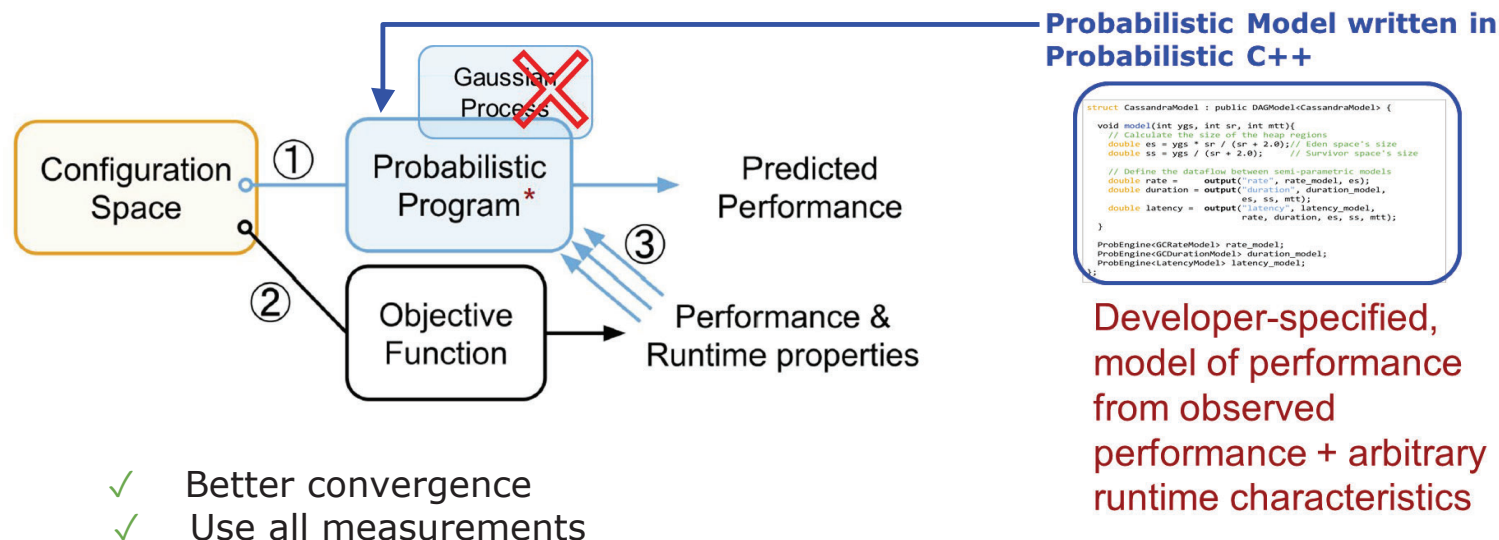
Probabilistic Programming



B. Paige

26

Structured Bayesian Optimisation (SBO)



27

Further Bayesian Optimisation...

■ BO overview/Tutorial

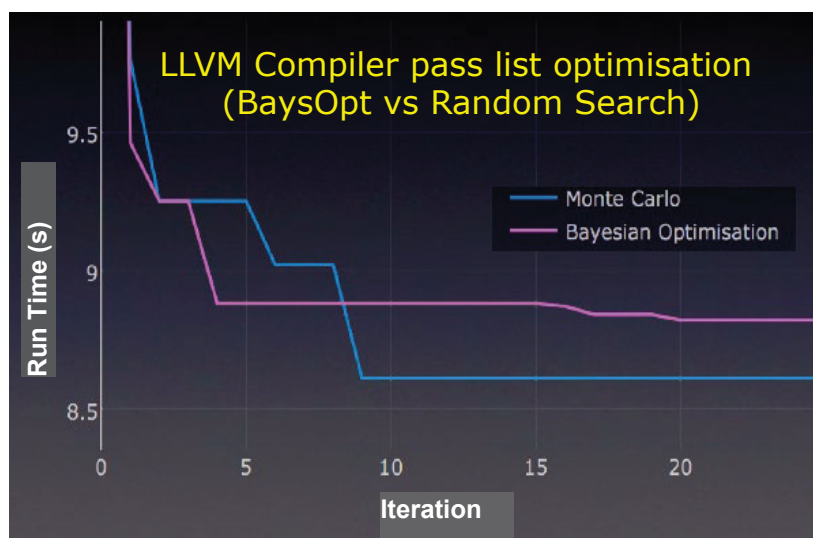
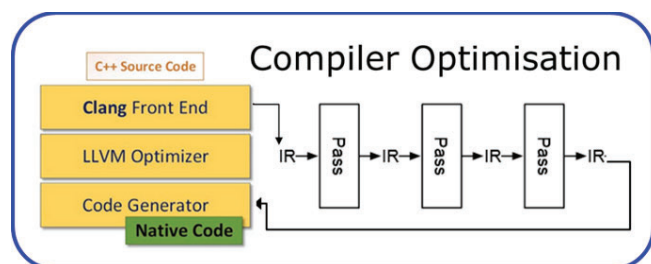
- https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2024_2025/aid/BO_overview_Archambeau.pdf
- https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2024_2025/aid/BO_overview_adams.pdf
- https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2024_2025/aid/BO_overview_gonzalez.pdf

■ 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.

28

Bayesian Optimisation not for Combinatorial Model



29

Reinforcement Learning in Computer Systems

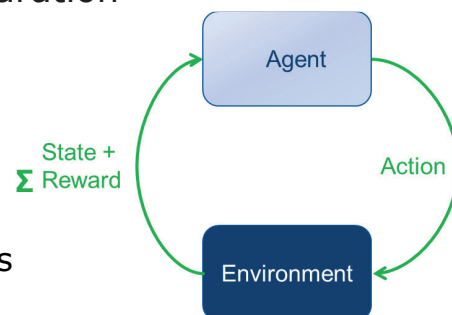
- Agent interacts with **Dynamic** environment
- **Goal:** Maximise expectations over rewards in agent's lifetime
- Notion of **Planning/Control**, not single static configuration

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

Practical Consideration:

- Action spaces do not scale
- Exploration in production system not a good idea
- Simulations can oversimplify problem (Expensive to build)
- **Online steps take too long**



30

Reinforcement Learning for Optimisation

Many problems in systems are sequential decision making and/or combinatorial problems

- Compiler Optimisation
- Chip placement
- Datacentre resource allocation
- Network congestion control with multiple connections
- Wide range of signals to make decisions (e.g., VM allocation)
- Database: Query optimiser, Dynamic indexing...

31

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

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

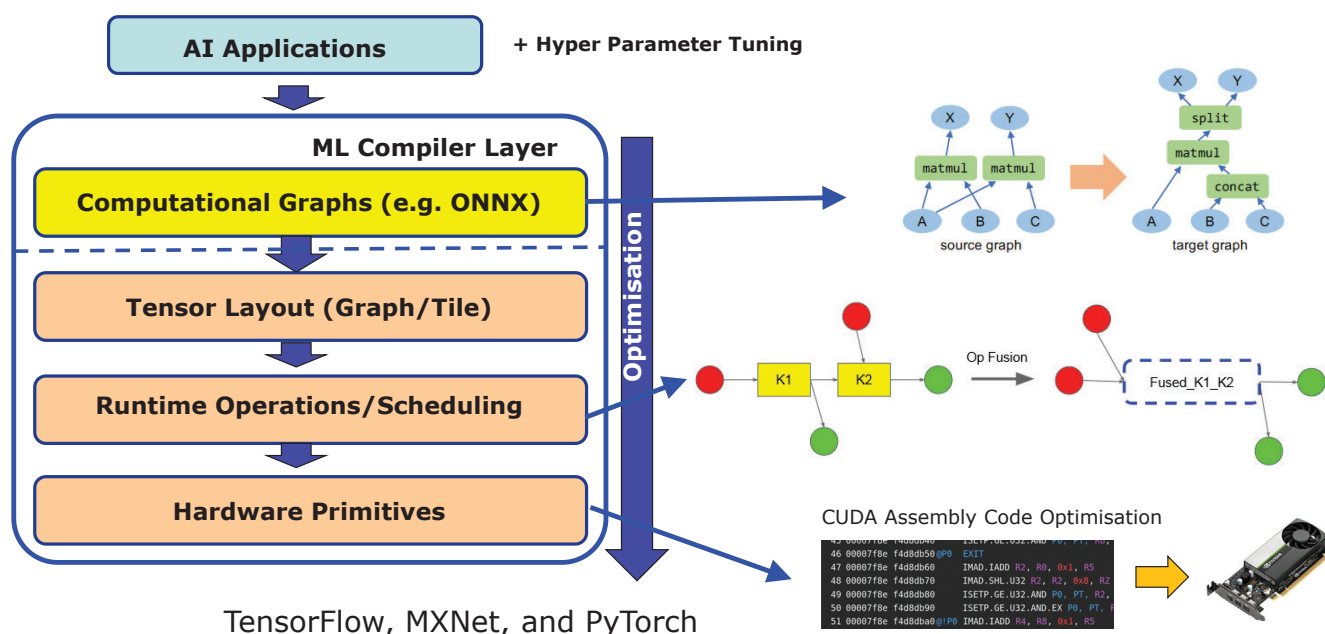
32

ML Compiler Challenges

- Fast tensor operation capable compilers: Complex optimisation plays a crucial role for LLM processing
 - Use of ML to optimise ML Compiler
 - Superoptimisation: Deal with massive parameter space
 - use of Equality Saturation (ES), MCTS
 - use of reinforcement Learning
 - Many compiler optimisations are solving phase ordering problem
- Challenge to bridge the gap between ML models and HW
- Existing ML Compiler
 - Apache TVM
 - NVIDIA TensorRT - CuDNN
 - ONNX runtime
 - LLVM
 - Google MLIR
 - TensorFlow XLA
 - Meta Glow
 - PyTorch nvFuser
 - INTEL pLAIDml
 - Open VINO

9

ML Compiler Optimisation

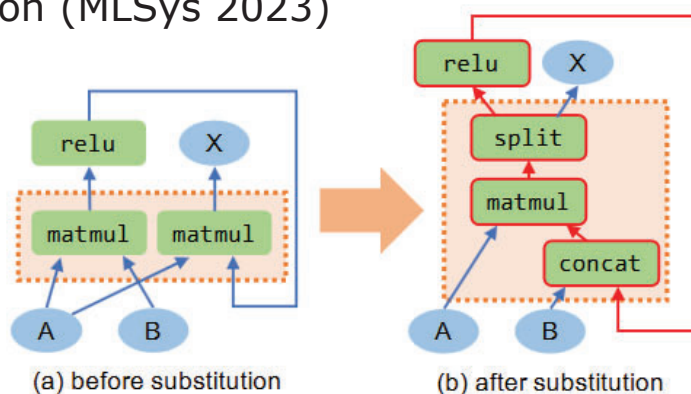


See ML Compiler Tutorial: <https://mlc.ai/summer22/schedule>

Also Survey: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&number=9222299>

Optimising DNN Computation with Graph Substitutions

- TASO (SOSP, 2019): Performance improvement by transformation of computation graphs
- PET (OSDI, 2021): Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections
- Equality Saturation for Tensor Graph Superoptimization (MLSys 2021)
- X-RLflow: Graph Reinforcement Learning for Neural Network Subgraph Transformation (MLSys 2023)



35

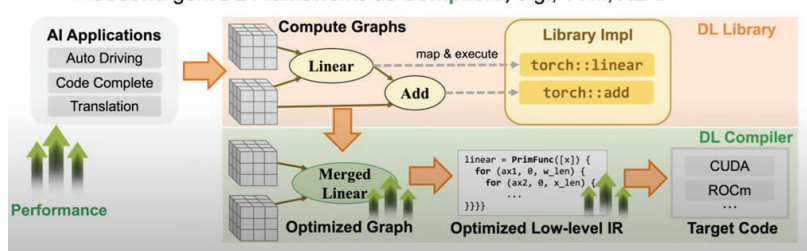
TVM: Deep Learning Compiler

- TVM: end-to-end deep learning compilers
- Operator/expression-level optimisation operates on Tensor Operation Abstraction(TIR)
- Core: Optimising programs through a sequence of transformation
- Automatic optimisation

DL Systems getting **Faster** & more **Complex**

❖ **First** gen. DL Frameworks as **Libraries**, e.g., PyTorch, TensorFlow.

❖ **Second** gen. DL Frameworks as **Compilers**, e.g., TVM, XLA.



36

CuAsmRL: Optimising CUDA GPU SASS schedules

- Optimising GPU SASS schedules via Deep RL

CUDA kernels compilation pipeline:

C++/CUDA → PTX → SASS → Cubin

SASS: GPU native instructions. GPU architectures-dependent. (SIP optimizes at this level)

- Search space:
 - Original: full permutation of instructions; computationally intractable
 - Pruning: only global read and write memory instructions, e.g. LDG, STG etc.
- Mutation:
 - Randomly select an instruction and reorder an instruction above or below
- Feedback signal:
 - Assemble the mutated SASS and run on GPUs

37

2026: What is next in ML/AI?

- Further larger models ?
- Complex reasoning of LLM - remains a challenge
- Instead of being solely focused on traditional tasks, they will need to integrate AI tools into their workflows
- Generative API to Agentic AI
 - 2023 AI: Generate content (write email, create image)
 - 2026 AI: Execute workflows (send email, book meeting, complete task)

➡ AI Agents ?

AI Workflow = Task runner

AI Agent = Goal solver

MCP = Infrastructure backbone

38

AI Agent?

- Multi-agent systems with separate agents dedicated to planning and evaluation → Models trained specifically to improve the model's built-in reasoning
- Build systems with separate agents dedicated to planning and evaluation process
- Models are trained to **think** before answering and have a self-contained reasoning process allowing them to break down tasks into simpler steps, work iteratively on the steps, recognise and correct mistakes before returning a final answer

39

Agentic Systems in the Modern Data Processing Stack

APPLICATION / CONTROL

Goals · planning · tool use
Multi-agent coordination
LangGraph · AutoGen · CrewAI

WORKFLOW EXECUTION

Durable tasks · DAGs · retries
Event-driven orchestration
Temporal · Ray · Prefect

STATE & CONTEXT

Session / workflow state
Memory · retrieval · vector DB
KV-cache reuse · checkpoints

MODEL & TOOL RUNTIME

LLM serving · inference
Tool / API execution
vLLM · SGLang · Triton

SHARED DATA PLANE · Kafka / Pulsar · object stores / lakehouse · databases · vector indexes

INFRASTRUCTURE & RESOURCE MANAGEMENT · Kubernetes · Ray clusters · GPUs / TPUs · autoscaling · scheduling

Optimisation focus: end-to-end latency, token / tool cost, placement, concurrency, fault tolerance, state locality and KV-cache lifecycle. Agentic systems are a cross-layer workload, not simply another processing engine.

40

■ *AI Is Reshaping the Landscape of Systems Research*

AI-driven evolutionary search could substantially accelerate systems research, but its effectiveness depends on the quality of the search space, evaluation framework, and human-defined research objectives (from Barbarian paper by UC Berkeley)

The availability of a reliable evaluator as a prerequisite for AI-driven research. However developing AI-based methods capable of reasoning, searching, and optimising when reliable evaluators are the important research challenge

41

Suitable for AI based approach

Scheduling	Clear objectives such as latency, throughput, makespan, and cost
Resource allocation and placement	Measurable performance and well-defined constraints
Load balancing	Algorithms can be evaluated using workload traces and simulators
Database optimisation	Query performance, transaction scheduling, and cache efficiency are measurable
LLM inference optimisation	Expert placement, KV-cache reuse, and inference efficiency can be benchmarked
Networking optimisation	Routing, congestion control, and related performance metrics can be evaluated

Candidate solutions can be generated, executed, and evaluated automatically using a reliable verifier

42

Less-suitable for AI based approach

Complex distributed protocols	Consensus protocols require coordinated changes across state management, networking, and failure detection
Semantic correctness problems	Arbitrary database query-plan transformations can be difficult or impossible to verify for equivalence
Expensive GPU optimisation	Weight compression and large-scale distributed training may require prohibitively expensive evaluations
Problems solvable by existing solvers	ILP and other established optimisation methods may solve the problem more directly and efficiently

Optimisation of an isolated KV-cache eviction policy is a good candidate. But jointly redesigning KV-cache management, distributed scheduling, state consistency, and agent orchestration is considerably harder.

43

Static and Dynamic Optimisation

Aspect	Static optimisation	Dynamic optimisation
Workload	Known or fixed	Time-varying, often unpredictable
Decisions	Made before execution	Continuously revised during execution
Search space	Relatively stable	State-dependent and evolving
Objective	Optimise a fixed configuration	Optimise long-term performance under changing conditions
Evaluation	Repeatable benchmarks	Workload-dependent, potentially stochastic
Verification	Relatively straightforward	Requires testing behaviour over time and across conditions
Examples	Static GPU placement, kernel scheduling	Online LLM scheduling, KV-cache management, adaptive resource allocation

44

Comparison to BO/RL/LLM

Bayesian optimisation (BO)	Reinforcement learning (RL)	LLM-based optimisation	
Primary goal	Find optimal configurations	Learn decision-making policies	Generate algorithms, programs, and optimisation strategies
Search space	Usually predefined parameters	Predefined states, actions, policy representations	Potentially open-ended program structures
Feedback	Objective measurements	Rewards and state transitions	Performance measurements, test results, semantic feedback
Static optimisation	Excellent for expensive evaluations	Possible, but often unnecessary	Promising for algorithm discovery
Dynamic optimisation	Possible through repeated optimisation	Particularly suitable for sequential decisions	Requires additional mechanisms for online adaptation
Main advantage	Sample efficiency	Long-term adaptive decision-making	Semantic reasoning and structural exploration
Main weakness	Limited by search-space design	Training cost and generalisation	Unreliable generation and expensive verification

45

Optimisation Current and Future

- First generation: BO and RL automate optimisation within human-designed search spaces and environments.
- Second generation: LLMs automate parts of algorithm generation and search-space construction.
- Next generation: LLMs, BO, and RL jointly support continuous algorithm evolution and adaptive system optimisation.
- The important research opportunity is not merely LLM versus BO versus RL, but how to develop a system that can automatically construct its **optimisation space**, discover effective algorithms, learn adaptive policies, and revise these as the workload evolves.

46

LLM-Based Optimisation: Where Is the Added Value?

- Traditional optimisation (mathematical programming, heuristics, search, BO, RL, compilers) already provides powerful methods for systems optimisation
- Use of semantic knowledge of code, algorithms, and system architecture for optimisation strategies
- Search-space construction: LLMs can generate new transformations, heuristics, and algorithm structures rather than exploring predefined choices
- Cross-layer reasoning: LLMs may identify optimisation opportunities spanning components that traditional methods usually optimise separately
- Integration: LLMs can generate candidate algorithms, while traditional optimisers efficiently tune parameters, select configurations, or learn runtime policies
- Dynamic systems: LLMs could periodically redesign policies, while lightweight conventional algorithms handle time-critical decisions
- Main limitation: LLM-generated solutions still **require rigorous correctness** verification, performance evaluation, and generalisation testing

47

Summary: Data Processing and Optimisation

- Computation graph (dataflow) is key element used in optimisation
- 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 ($\sim$ Computational Graph) structural transformation/Decomposition
- LLM-Based Optimisation: A New Frontier with Much to Explore
- Exploit structural information for model decomposition to accelerate optimisation process and/or transform the structure

48