CrystalPerf: Learning to Characterize the Performance of Dataflow Computation through Code Analysis

Huangshi Tian, Minchen Yu, Wei Wang

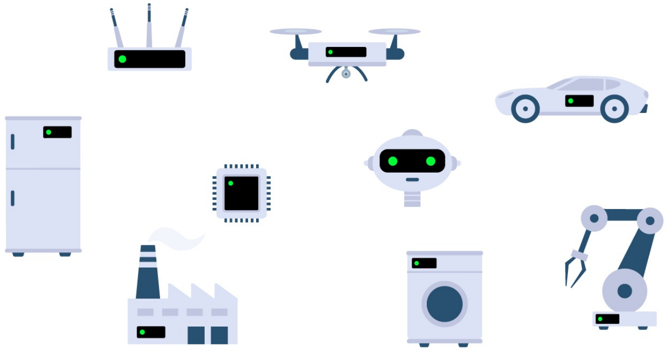
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HUAWEI

Business Analytics

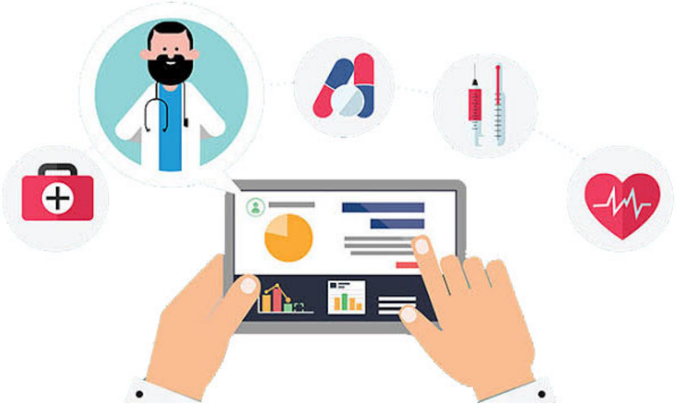
APACHE Spark TEZ

Real-Time Monitoring



Flink HERON

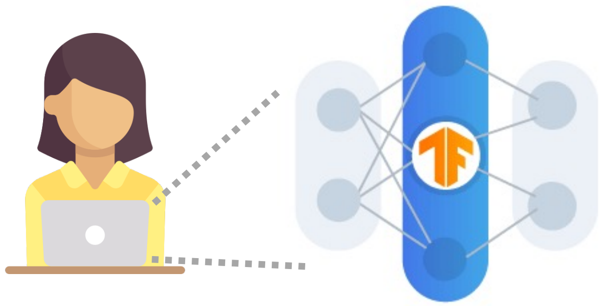
Health Diagnostics

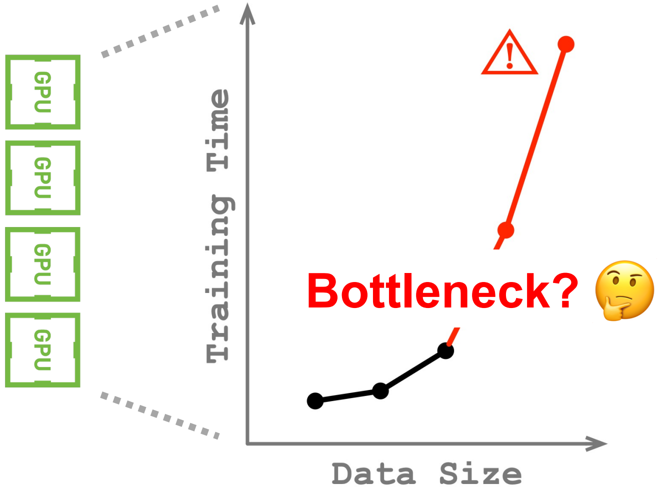


TensorFlow mxnet

Dataflow computation is *prevailing* and *diverse*.

The Troubles of Data Analyst Jane[1]



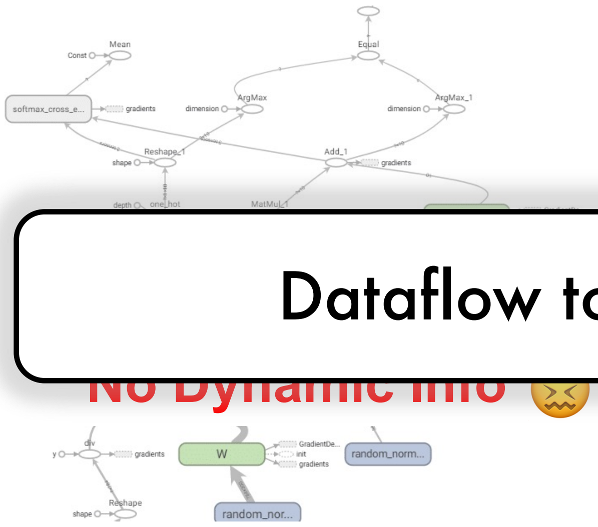


1. Jane is developing a **TensorFlow** model for her medical project.

2. She gets confused when the program **cannot scale** even with **multiple GPUs**.

1. Based on a real question from Stack Overflow: https:// bit.ly/2kiT2dD

The Troubles of Data Analyst Jane



Dataflow toolchain is *lagging behind*.

3. She tries to visualizes the model, but the **static** graph does not tell much about the execution.

4. The built-in profiler gives over-whelmingly much **low-level infor-mation**.

Resource Problems

Performance Debugging
examine malfunctioning resources

Performance Reasoning
what if more resources are allocated

Program Diagnosis
detect bottleneck and inefficiency

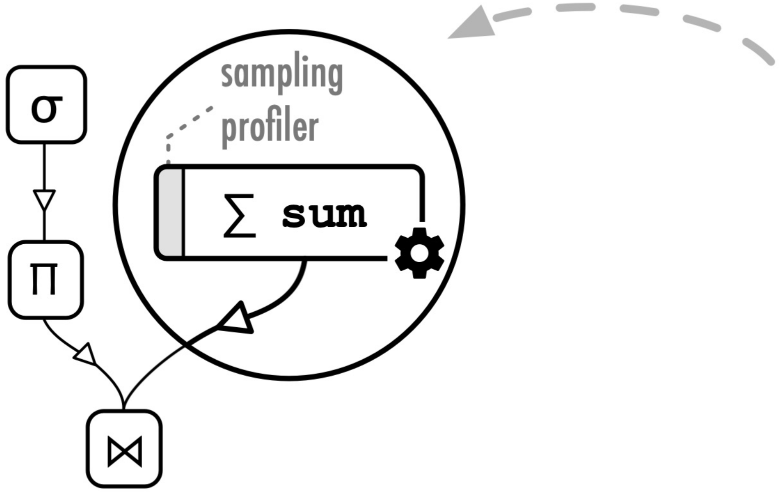
Existing Solutions

Framework-Specific
Starfish (VLDB'11)
Ernest (NSDI'16)

Requires Instrumentation
Monotasks (SOSP'17)
SnailTrail (NSDI'18)

Objective: General Performance Characterization without Instrumentation

Overview: Finding Resource-Time Relationship

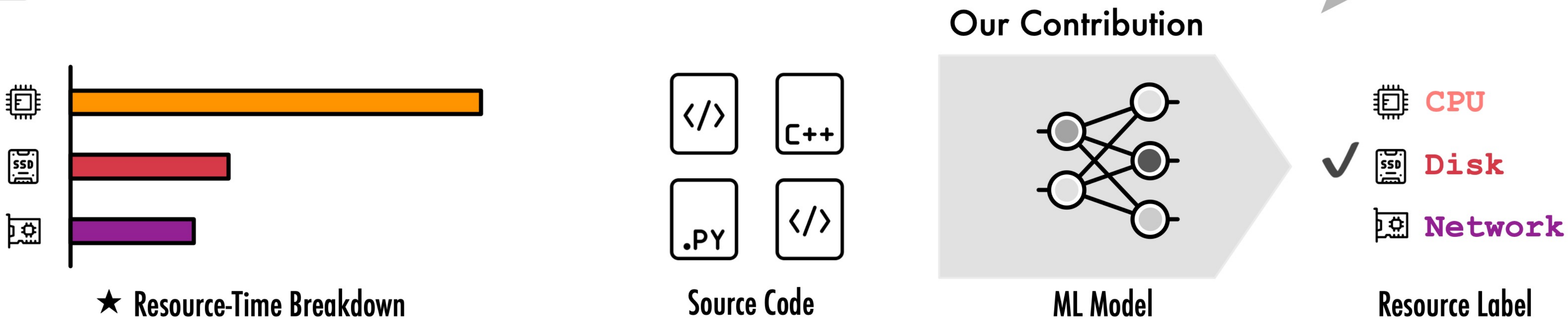


Instrumentation-Free

Collecting runtime information *non-intrusively*.

Framework-Independent

Exploiting resource information in source code.



Outline

- ❑ Background and Overview
- ❑ **Resource Classifiers**
  how we infer resource usage from source code
- ❑ **Execution Profile**
  how we represent job execution and debug performance
- ❑ **Resource Models**
  how we model resource behavior and predict performance
- ❑ **Evaluation Highlights**

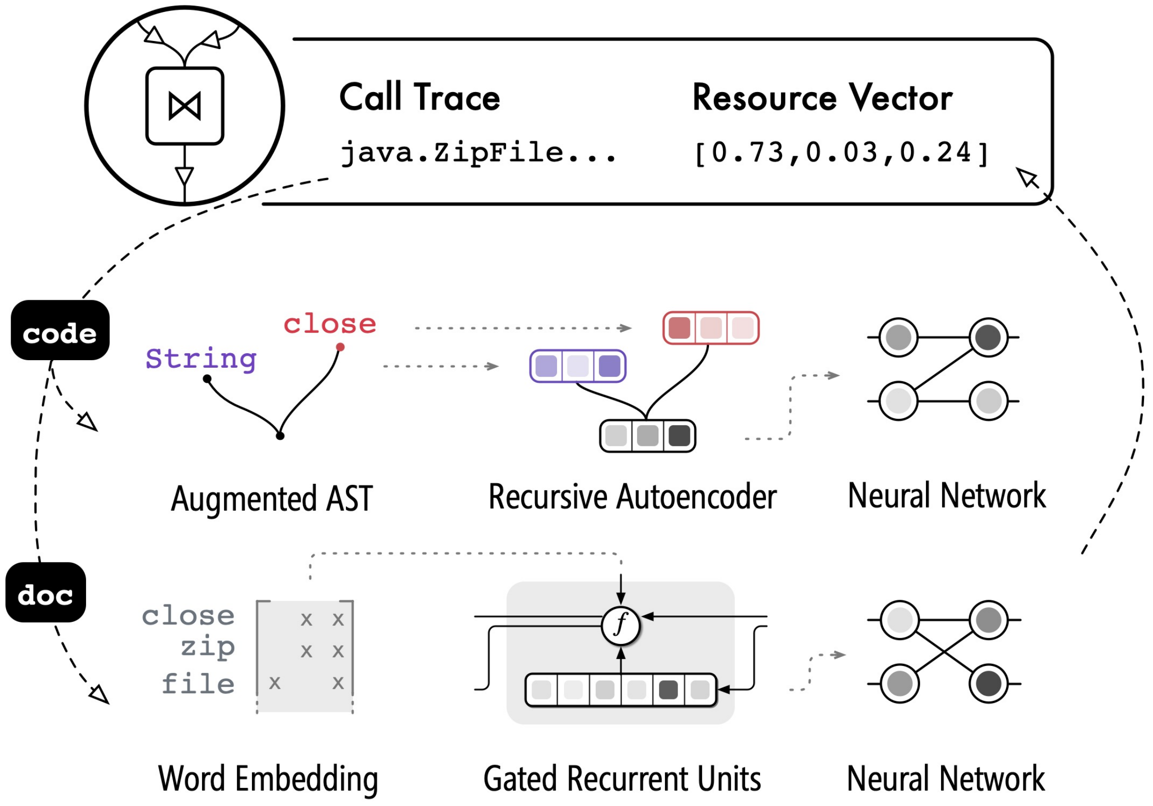
Resource Classifiers

Resource Vector

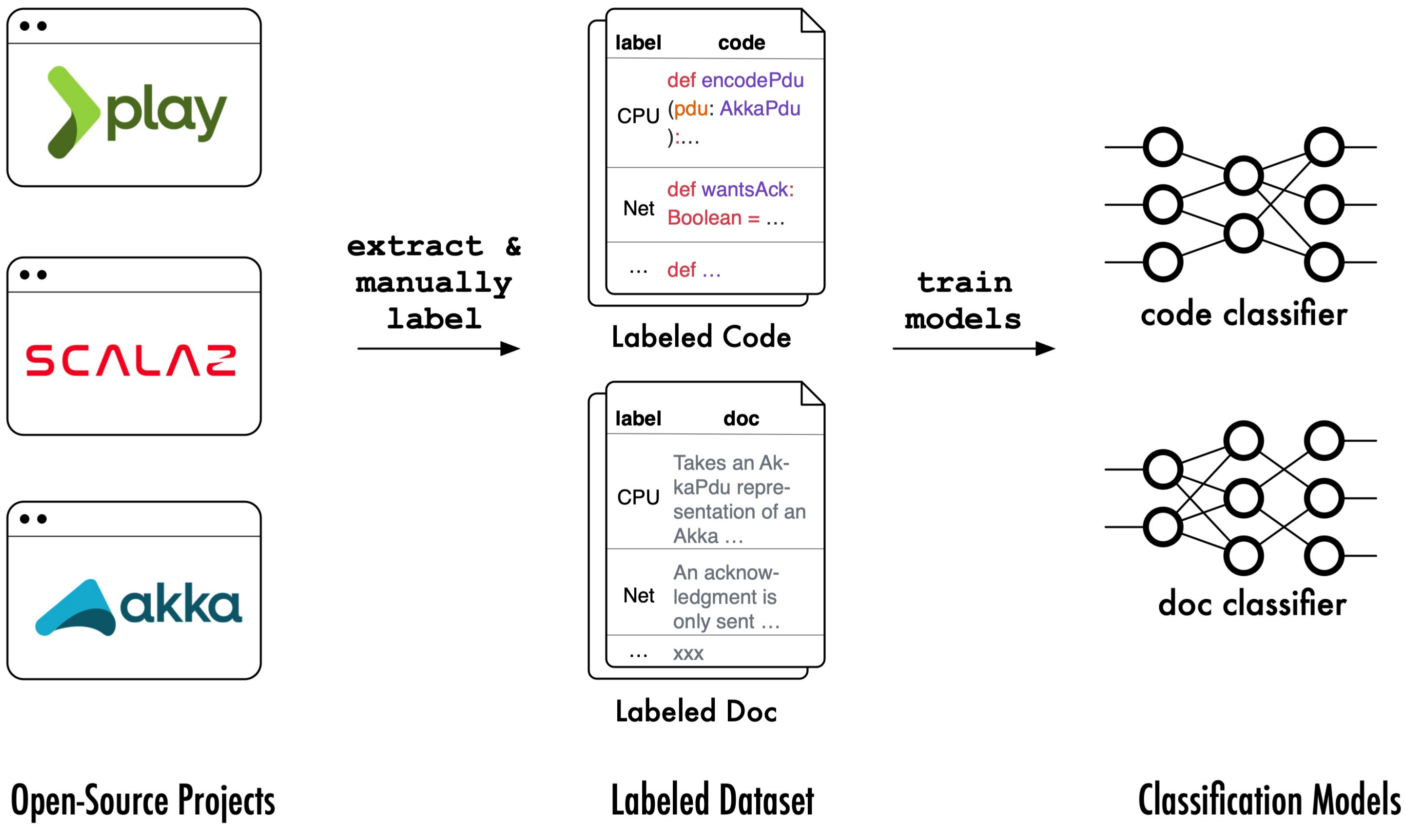
- each component is the *probability* of using certain resources
- in the form of $(p_{cpu}, p_{disk}, p_{net})$

Classification Models

- two *separate* models for code and documentation
- model details in the paper



Model Training



Why Classifiers Work?

Explanatory Framework: LIME[1]

- explains why a model makes certain prediction

Procedure

- Collect a manually verified dataset.
- Train two classifiers.
- Explain their predictions.

Table 1: Top 5 words that cause the classifier predict a certain resource.

Resource	**Words**
CPU	engine, entry, stream, key, certificate
Disk	file, error, tar, info, name
Network	socket, sock, send, result, address

1. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, SIGKDD'16

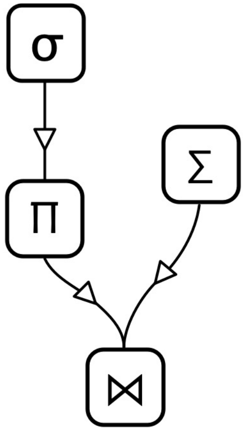
Execution Profile

Key Information

- *runtime* of the operator from logs
- *resource vector* inferred by the classifiers

Performance Debugging

- estimate resource-time from resource vector



Dataflow

Execution Profiles

op	runtime	res. vec.
σ	25	[0.6,0.2,0.2]
Σ	100	[0.8,0.1,0.1]
Π	15	[0.6,0.1,0.3]
⋈	200	[0.7,0.1,0.2]

bottleneck: compute

Performance Prediction

Key Question

- How runtime would change given a resource variation?

Resource Models

- *CPU*: simulated rescheduling with warmup effect
- *Memory*: reverse-roofline model
- *I/O*: buffered transmission model
- detailed models in the paper

Execution Profiles

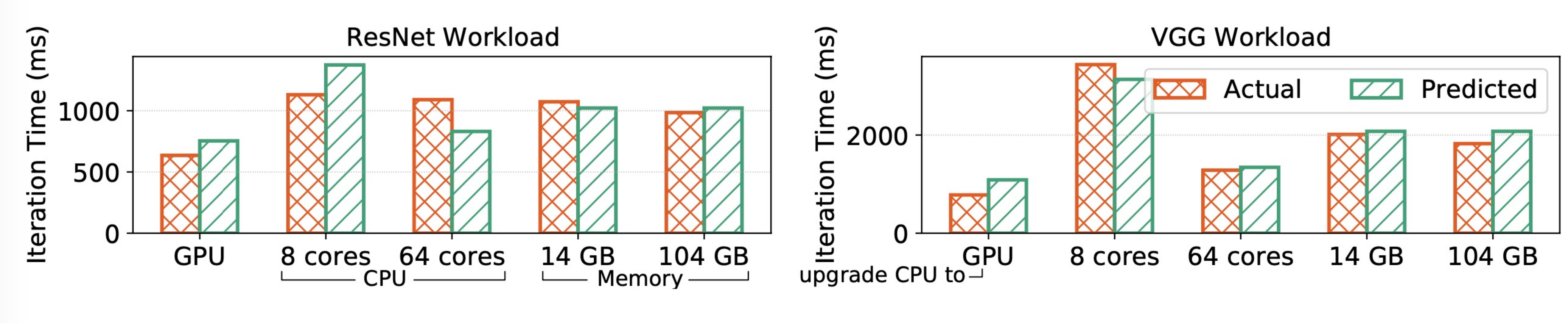
op	runtime	res. vec.	pred. (2x CPU)
σ	25	[0.6,0.2,0.2]	-7.5
Σ	100	[0.8,0.1,0.1]	-40
Π	15	[0.6,0.1,0.3]	-4.5
$\bowtie$	200	[0.7,0.1,0.2]	-70

pred. runtime: 190

Approach implemented as a CLI tool.

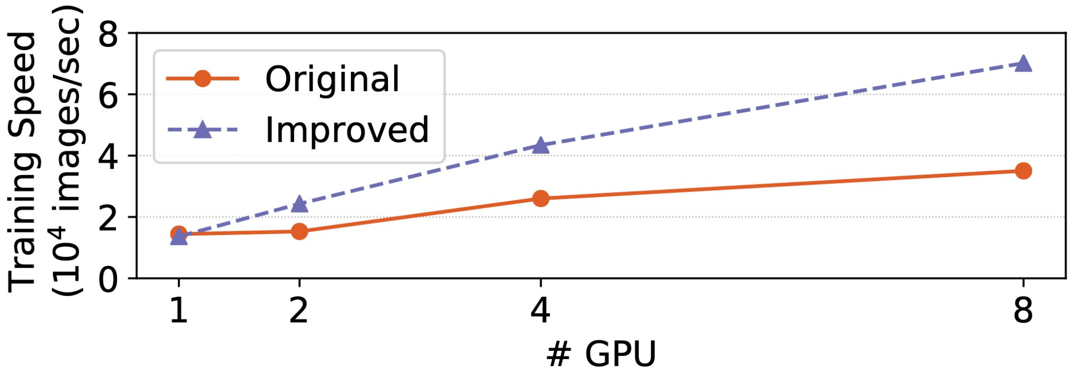
Evaluation: Prediction Accuracy

Framework	Workload	Varied Resources	Error
Spark v2.4.3	two queries from TPC-H with scale factor 100	# CPU cores / memory / network bandwidth	13.49% ± 8.57%
Flink v1.7.2	Yahoo Streaming Benchmark and Dhalion Benchmark	CPU share / network bandwidth	12.70% ± 10.11%
TensorFlow v1.13	ResNet and VGG on a flower image dataset	computing devices / # CPU cores / memory	14.22% ± 11.77%



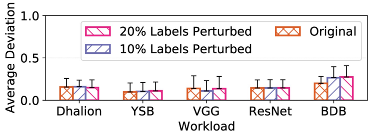
Case Study: Identifying Bottleneck

- In Jane's case, CrystalPerf identifies the bottleneck as I/O.
- We enable NCCL, an optimized library for inter-GPU I/O, and find the scalability improved.



More Results in the Paper

- Robustness against labeling inaccuracy.



- Generalizability to other frameworks.

	Code Cls.	Doc. Cls.
Spark	81.0%/.797	76.5%/.673
Flink	73.3%/.664	74.8%/.589
TensorFlow	79.9%/.752	77.8%/.652

- More real-world case studies.

7.5 CrystalPerf in Action: Case Studies

We further conduct three real-world case studies to demonstrate how CrystalPerf could help users identify resource bottleneck and address performance issues with ease.

Diagnosing Slow Operation A user implemented a Spark

Thank You for Your Attention!

For more Q&A, please contact

Huangshi Tian, htianaa@cse.ust.hk