SPARK+AI
SUMMIT EUROPE

Towards a Unified Data Analytics Optimizer

Yanlei Diao

Ecole Polytechnique, France
University of Massachusetts Amherst, USA

#bigdata #deeplearning #optimization

Today's Data Analytics Service

```
SELECT C.uid, avg(P.pagerank)
FROM     Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

ClickStream

Sessionization

Training of a classifier

Predication

Cloud Computing (EC2 ~60 instance types)
-t2.nano –t2.micro –t2.small –t2.medium –t2.large –m4.large –m4.xlarge –m4.2xlarge -m4.4xlarge –m4.10xlarge -m3.medium -m3.large –m3.xlarge –m3.2xlarge –c4.large –c4.2xlarge ...

Today's Data Analytics Service

```
SELECT C.uid, avg(P.pagerank)
FROM     Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

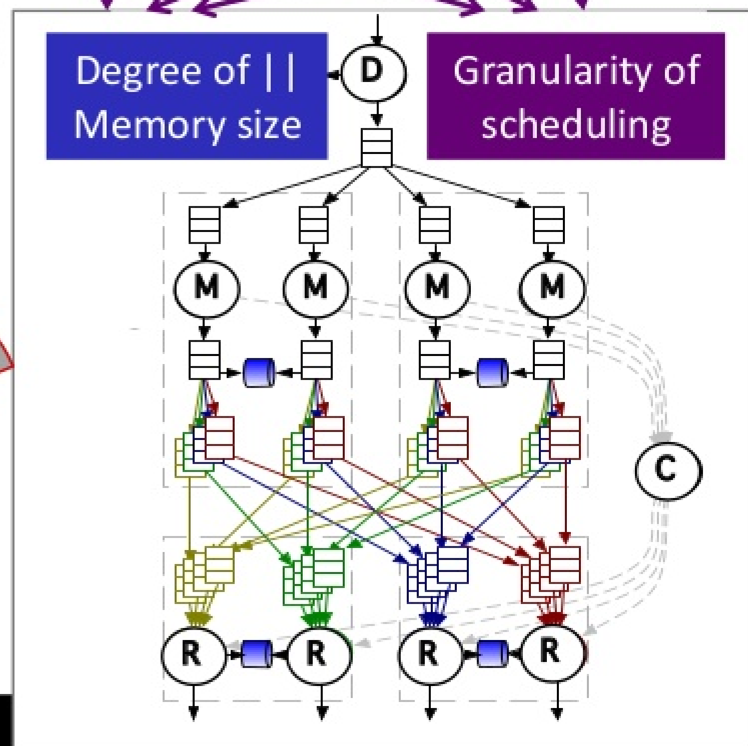
ClickStream

- Sessionization
- Training of a classifier
- Predication

Cloud Computing: ~60 instance types

-t2.nano –t2.micro –t2.small –t2.medium –t2.large –m4.large –m4.xlarge –m4.2xlarge -m4.4xlarge –m4.10xlarge -m3.medium -m3.large –m3.xlarge –m3.2xlarge –c4.large –c4.2xlarge ...

Latency | Throughput | Monetary cost



Today's Data Analytics Service

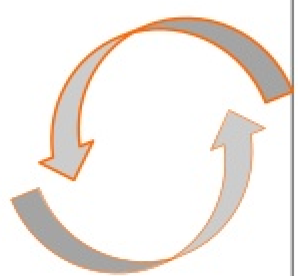
```
SELECT C.uid, avg(P.pagerank)
FROM    Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

ClickStream

Sessionization

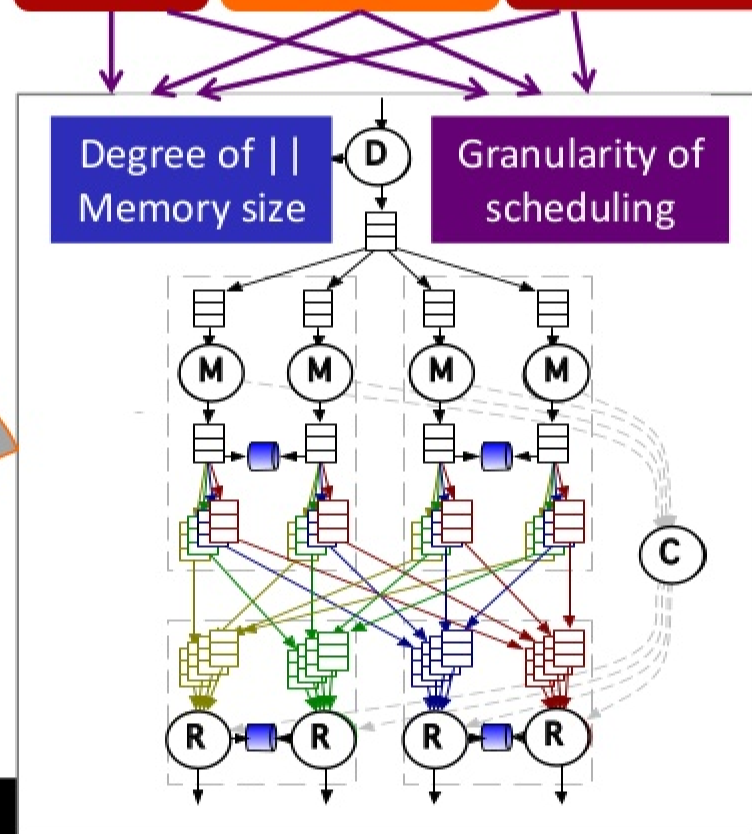
Training of a classifier

Predication



Cloud Computing: ~60 instance types
-t2.nano –t2.micro –t2.small –t2.medium –t2.large –m4.large –m4.xlarge –m4.2xlarge -m4.4xlarge –m4.10xlarge -m3.medium -m3.large –m3.xlarge –m3.2xlarge –c4.large –c4.2xlarge ...

Latency | Throughput | Monetary cost



Today's Data Analytics Service

```sql
SELECT C.uid, avg(P.pagerank)
FROM    Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

Today's Cloud Service guarantees availability (SLA)

But too many configuration issues are left to the user…

Sessionization

Training of a classifier

Predication

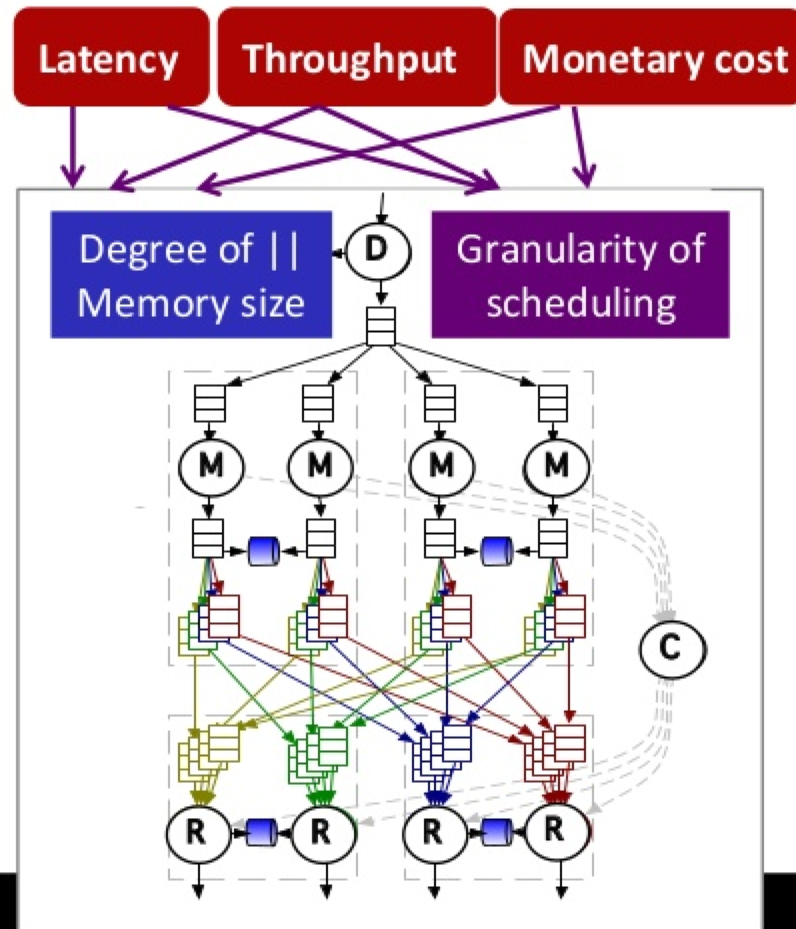
A New Data Analytics Service

```
SELECT C.uid, avg(P.pagerank)
FROM     Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

- Sessionization
- Training of a classifier
- Predication

Cloud Computing: ~60 instance types

-t2.nano –t2.micro –t2.small –t2.medium – t2.large –m4.large –m4.xlarge –m4.2xlarge - m4.4xlarge –m4.10xlarge -m3.medium - m3.large –m3.xlarge –m3.2xlarge –c4.large – c4.2xlarge …



A New Data Analytics Service

```
SELECT C.uid, avg(P.pagerank)
FROM     Clicks C, Pages P
WHERE   C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)
```

Sessionization

Training of a classifier

Predication

Latency | Throughput | Monetary cost

Dataflow Optimizer

Bring database optimization to a broader class of analytics

① job configuration

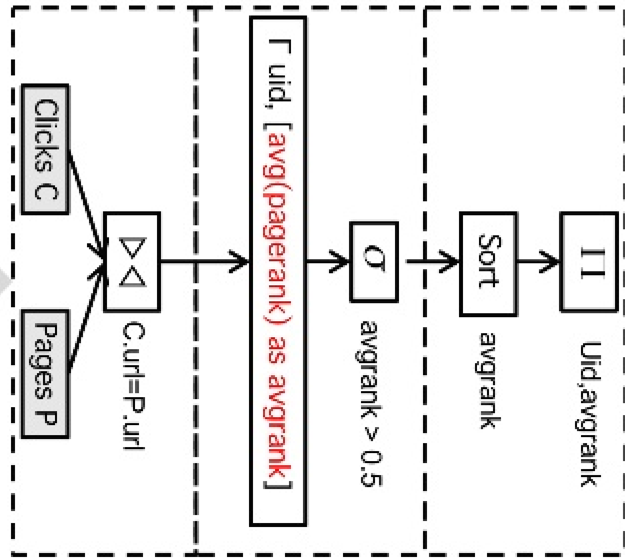
② cloud instance

Dataflow Optimizer: dataflow programs

Latency | **Throughput** | **Monetary cost** | **...**

SELECT C.uid, avg(P.pagerank)
FROM Clicks C, Pages P
WHERE C.url = P.url
GROUP BY C.uid
HAVING avg(P.pagerank) > 0.5
ORDER BY avg(P.pagerank)

logical optimization of the query plan



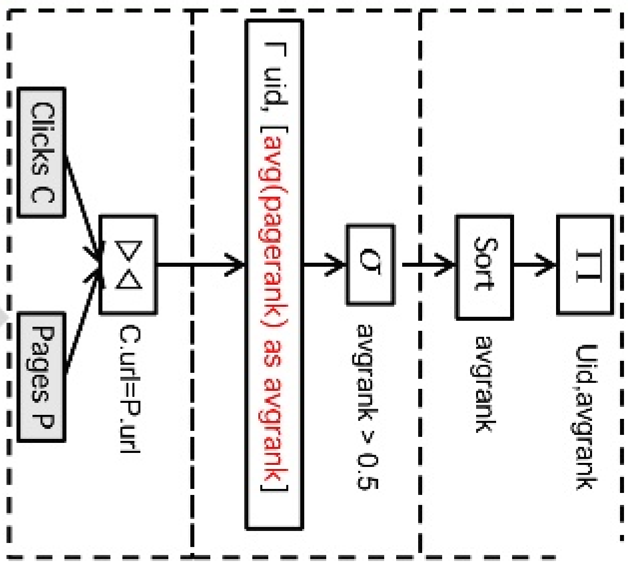
Sessionization

Training of a classifier

Predication

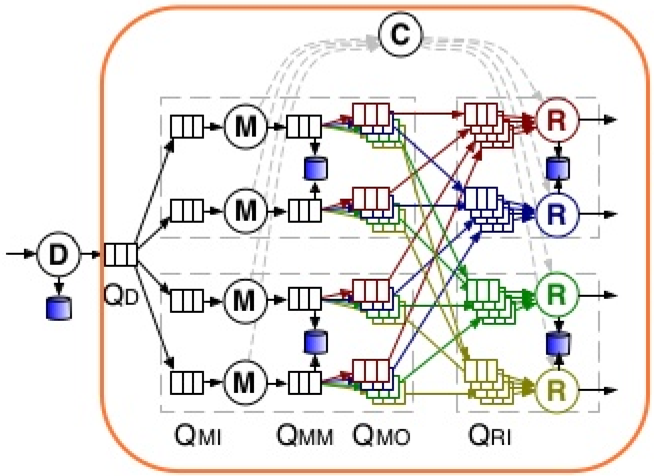
Dataflow Optimizer: from Logical to Physical Plans

Logical Plans



Phyiscal Plans

Hadoop Streaming [Li et al 2011,2012,2015]

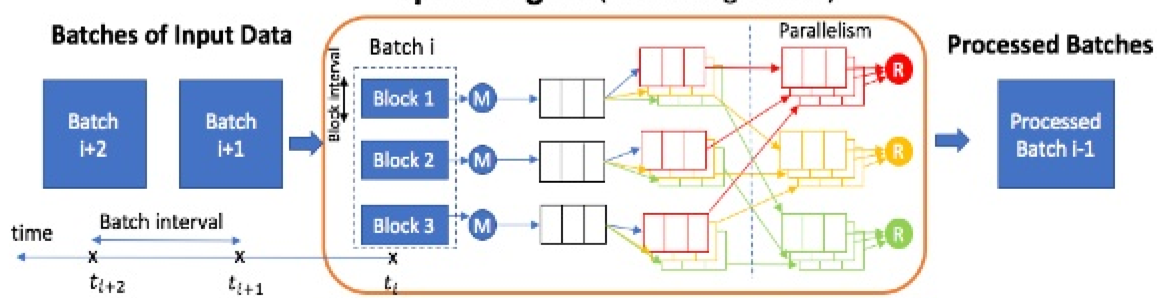


Degree of parallelism θp

Granularity of scheduling θs

Memory per executor…

Spark Engine (Processing Batch i)



Challenge 1: Cost Models for the Dataflow Optimizer

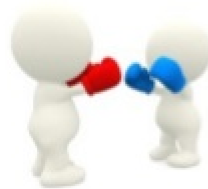
SQL Optimizer	Dataflow Optimizer
• Cost model for **relational algebra**	• Cost model for arbitrary **dataflow programs** (in java, scala, python, …)
• Cost model for **resource consumption** (CPU and IO counters)	• Cost models for any **user-defined objectives** (latency, throughput, cost, …)
• System behaviors are often centered on CPU and IO	• Modeling **diverse system behaviors**, including CPU, IO, shuffling, queuing, data skew, stragglers, failures…
• Tend to run **homogenous hardware**	• Cloud computing may employ **heterogeneous hardware**

Building a general cost model for any user objective is hard!

Challenge 2: Need A New Multi-Objective Optimizer

Latency versus Throughput

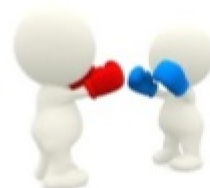
1M tuples/sec with 1 sec latency



1K tuple/sec with 0.1 sec latency

Latency versus Monetary Cost

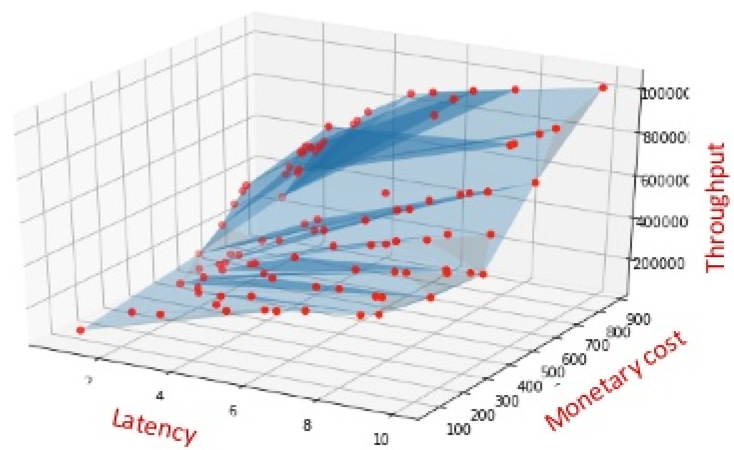
1 sec latency at $800/day



5 sec latency at $300/day

Multi-Objective Optimization

- Solution is a (Pareto) set
- Reveals interesting tradeoffs



<θ*p*, θ*s*, λ, …>

Overview: Cost Modeling for Dataflow Optimization

Dataflow Optimizer

- Cost model for arbitrary **data dataflow programs** (in java, scala, ...)
- Cost models for any **user-defined objectives**
- Modeling **diverse system behaviors** including CPU, IO, shuffling, queuing, data skew, stragglers, failures…
- Cloud computing may employ **heterogeneous hardware**

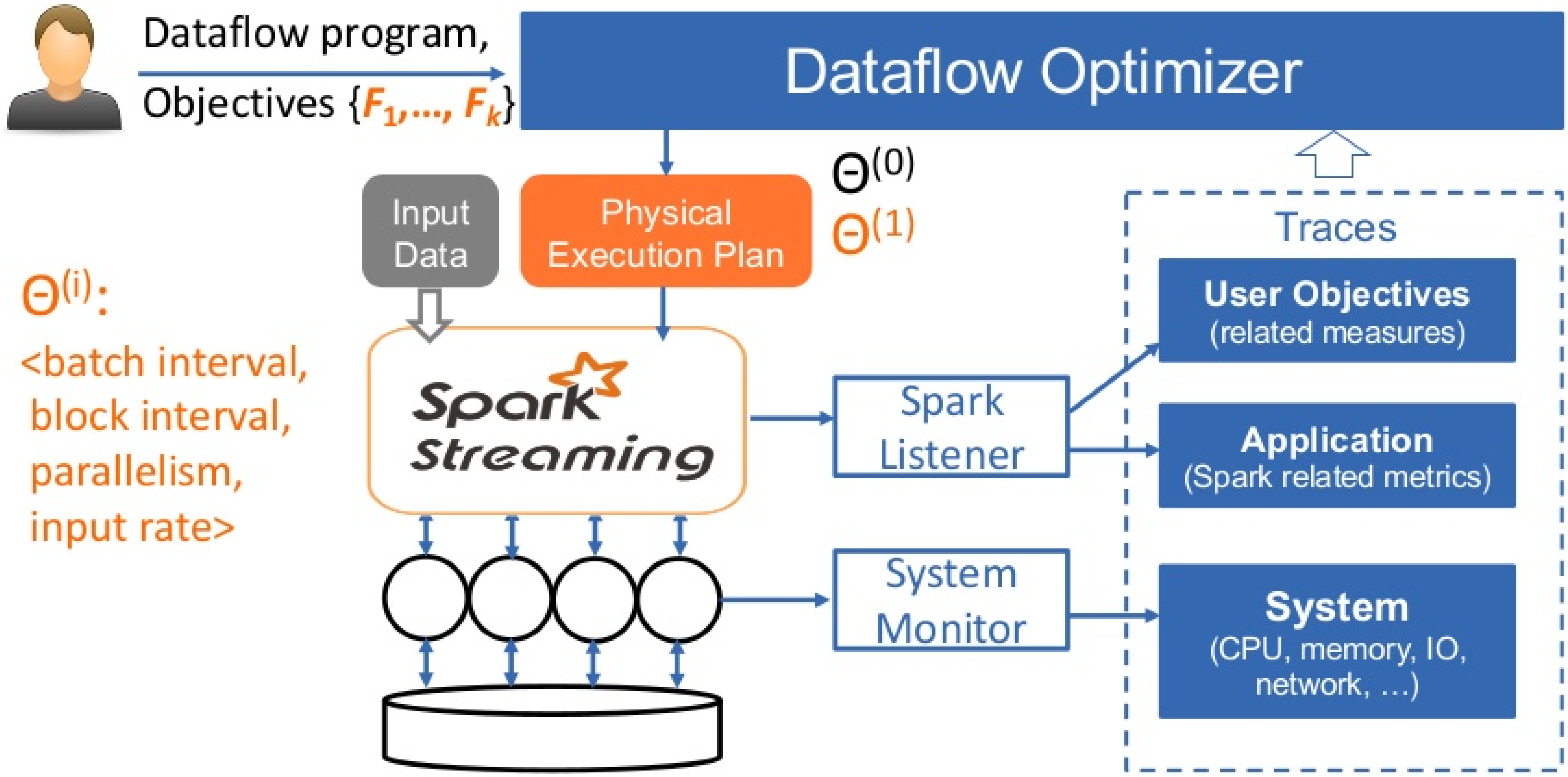
❑ In-situ modeling

- Learn a model as complex as necessary for the current computing environment

❑ Deep Learning

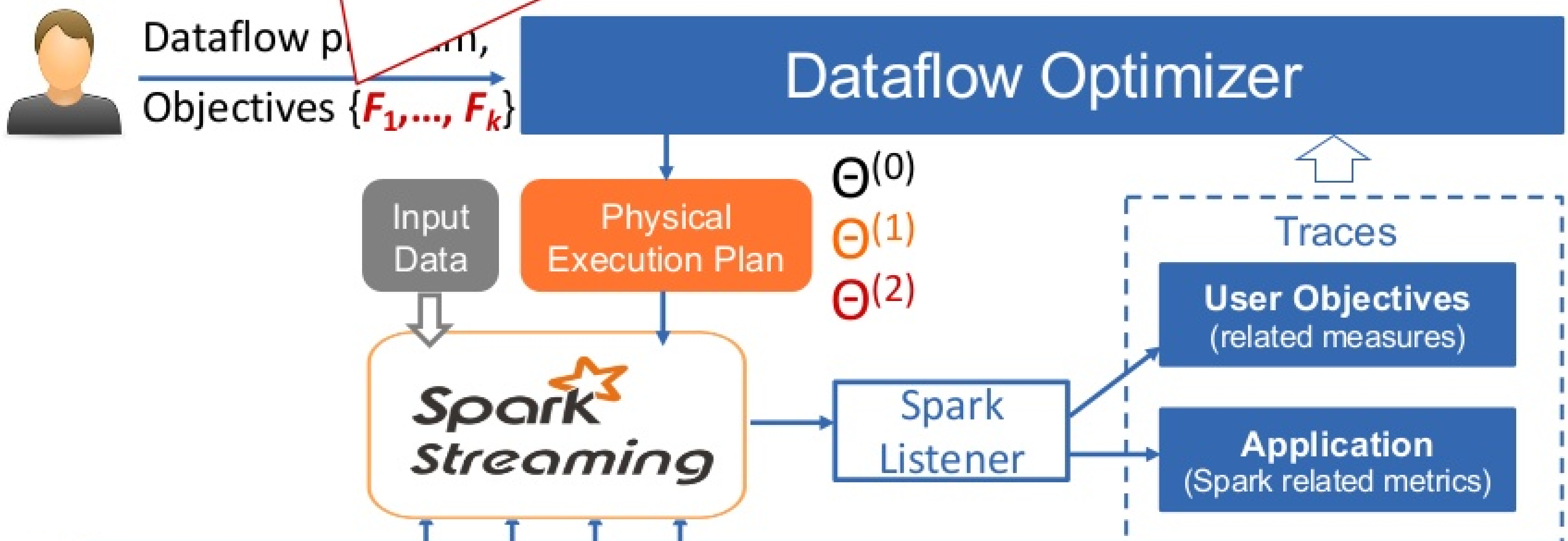
- Representation learning for an arbitrary program
- A (non-linear) function for any objective

Trace Collection for Cost Modeling & Optimization



Trace Collection for Cost Modeling & Optimization

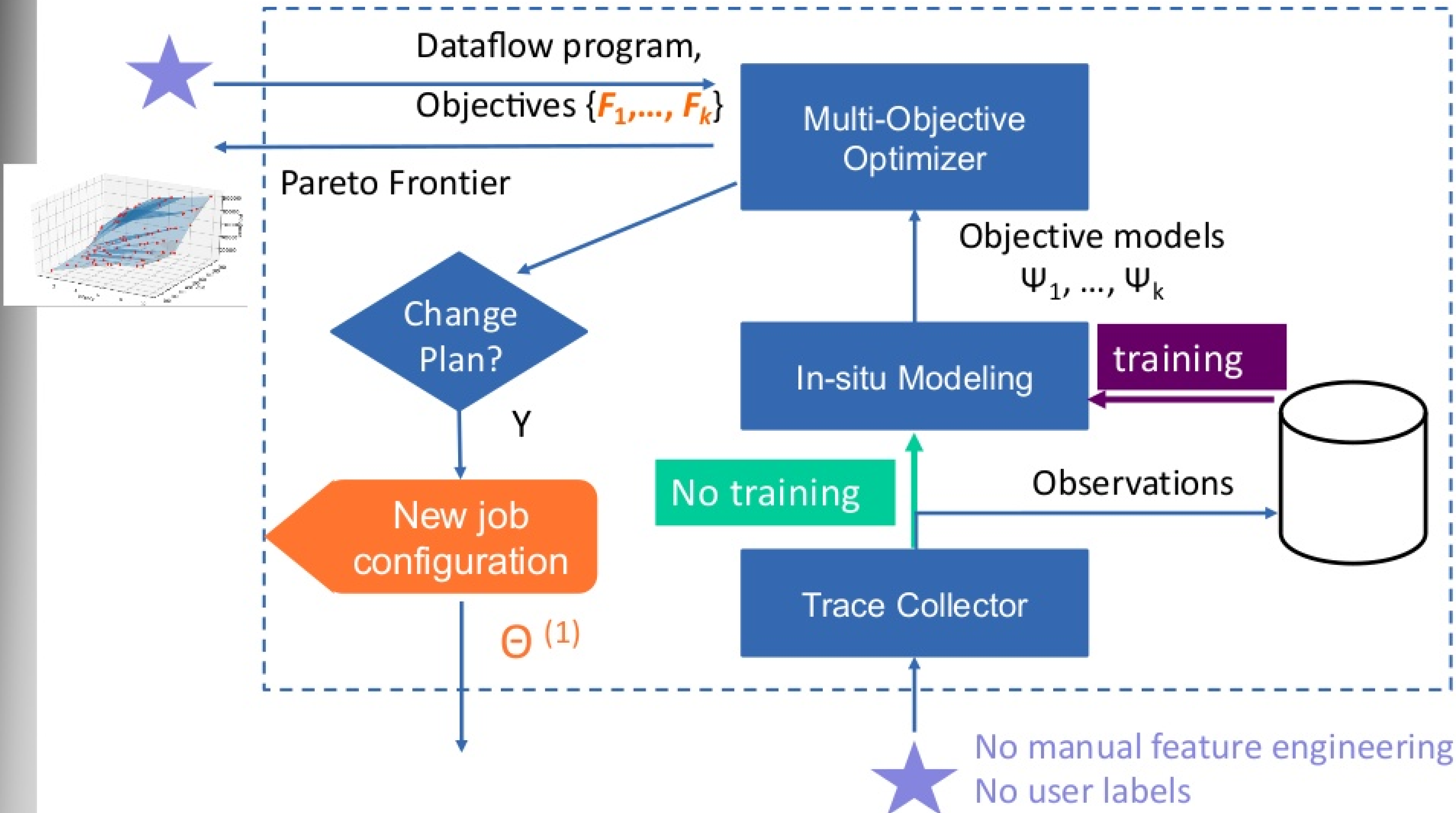
Significant change of user preference, e.g. switching to the **replay** mode



Optimization time: $T_{\Theta^{(i)}} - T_{ith\_preference\_change}$

- $T_{\Theta^{(1)}}$: job first seen, workload encoding and optimization
- $T_{\Theta^{(i), i>1}}$: job already seen, optimization

Inside the Dataflow Optimizer



1. Building a Cost Model for each User Objective

Given a dataflow, we assume that there is a job-specific encoding W_j (of its key characteristics) which is invariant to time and runtime parameters.

Definition: For a job *j*, the workload encoding W_j is a real-valued vector satisfying three conditions:

1. **Invariance**: For the same logical dataflow program, W_j should be (approximately) an invariant during a period of time
2. **Reconstruction**: W_j should carry all the information for reconstructing the observations (traces), given a specific job configuration Θ_j^i
3. **Similarity Preserving**: For similar programs *i* and *j*, their encodings W_i and W_j should also be similar

Formal Description of the Predictive Model

Under our assumption, the model for a user objective (e.g., latency) can be abstracted as a deterministic function:

$$\Psi(\theta_j^i, W_j) = l_j^i$$

Interpreted as an optimization problem, our goal is to find a function s.t.

$$\Psi^* = \arg\min_{\Psi} \quad \text{avg}(\text{distance}(\Psi(\theta_j^i, W_j), l_j^i)$$

where average is taken among all training instances (j, W_j).

Challenge: Building the Ψ function is easy if both W_j and Θ_j^i are known, but the workload encoding W_j is <u>unknown</u> in practice!

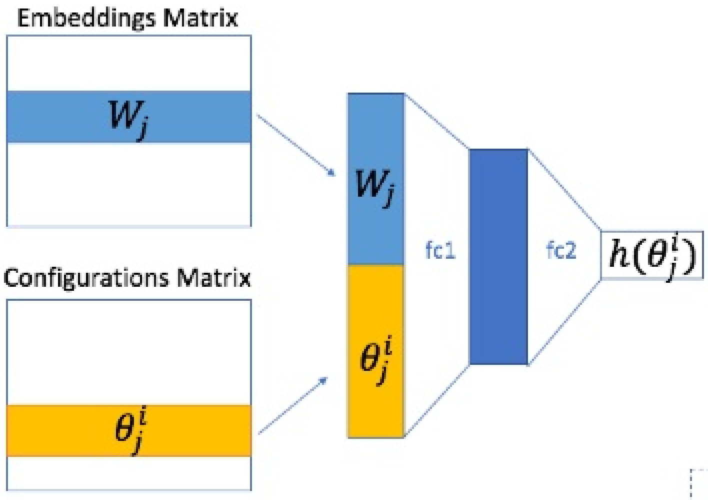
<u>**Neutral network architectures**</u> that can

(1) extract the workload encoding for each new job;

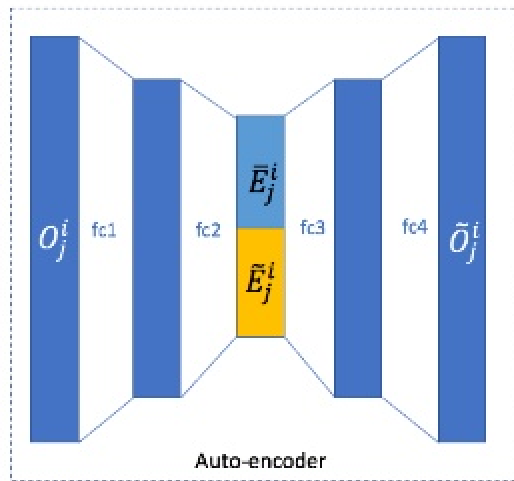
(2) predict on a user objective given a new job configuration.

Two Types of Architecture

1. Embedding



2. Autoencoder



→ Regressor

Results on Modeling

52 SQL-like jobs from 5 parameterized templates for click stream analysis, which use different group-by's, joins, global or windowed aggregates, and user-defined functions. 6 intensive jobs (each with 455 conf.), 46 regular jobs (each with 25 conf.).

12 ML jobs based on binary classification using Stochastic Gradient Descent. Each has 25 configurations.

	SQL-like Jobs		ML Jobs	
	T1	**T2**	**T1**	**T2**
Embedding	9.7%	33.3%	16.5%	16.2%
Autoencoder	11.6 %	22.0%	24.9%	53.9%
Autoencoder + Opt2	11.6 %	13.4%	24.9%	25.0%
Autoencoder + Opt1	8.5%	10.3%	2.9%	2.6%
Autoencoder + Opt1/2	8.5%	8.4%	2.9%	3.6%
HadoopStreaming [17]	31.9%	31.9%	84.6%	84.6%

Results on Modeling

Predication accuracy on **familiar** jobs

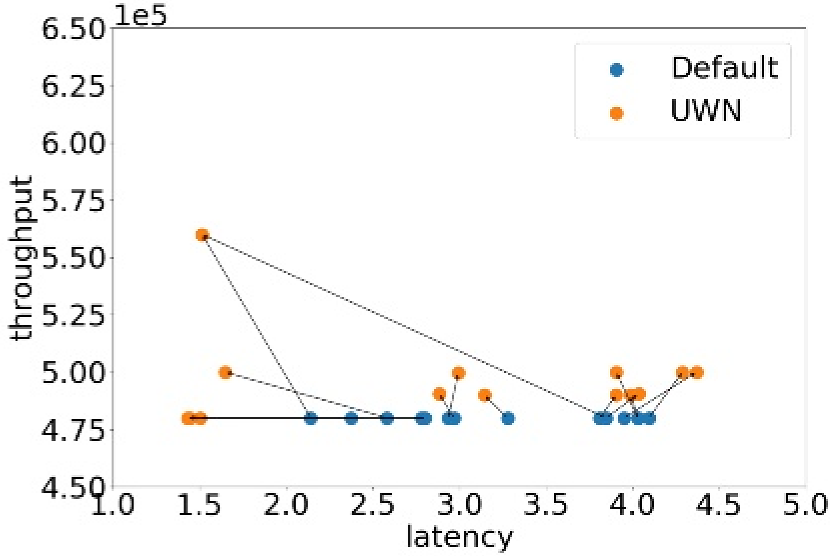
Prediction accuracy on **new (seen first time)** jobs

	SQL-like Job		ML Jobs	
	T1	**T2**	**T1**	**T2**
Embedding	9.7%	33.3%	16.5%	16.2%
Autoencoder	11.6 %	22.0%	24.9%	53.9%
Autoencoder + Opt2	11.6 %	13.4%	24.9%	25.0%
Autoencoder + Opt1	8.5%	10.3%	2.9%	2.6%
Autoencoder + Opt1/2	8.5%	8.4%	2.9%	3.6%
HadoopStreaming [17]	31.9%	31.9%	84.6%	84.6%

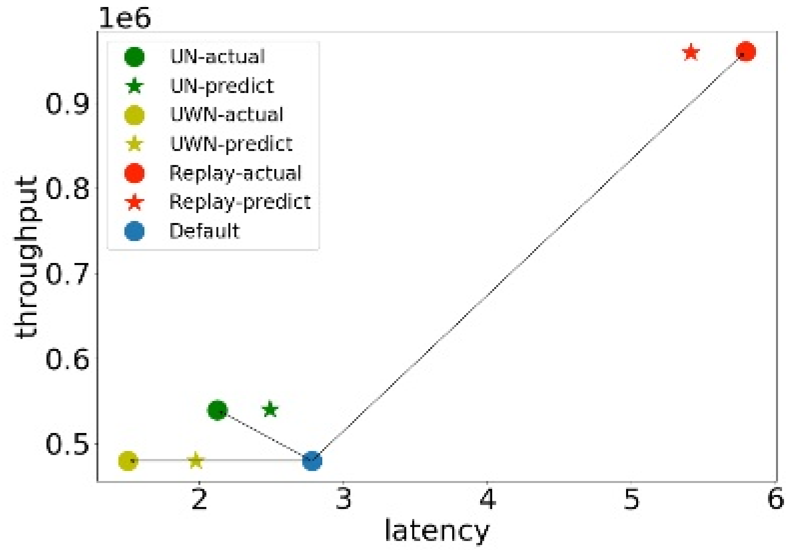
2. A New Multi-Objective Optimizer

- Given a set of user objectives, a Multi-Objective Optimizer uses the predicted performance measures to construct a Pareto optimal set (frontier).
- The skyline offers insights on tradeoffs:
  - e.g., two configurations consume the same amount of resources, but one achieves 20% higher throughput with only 1% loss of latency
- It finally chooses one optimal configuration to set the system parameters (e.g., degree of parallelism, granularity of scheduling, input rate) for execution.

Integration Results



Improvement over configurations set by engineers, dominance in 10/15, tradeoffs in 5/15



Improving throughput in the replay mode, reducing running time by 50%

Messages

✧ **In-situ modeling based on deep learning** has the potential to model any user objective in a given computing environment

✧ **Multi-objective optimization** has the potential to explore tradeoffs and move in direction of better performance automatically

Acknowledgements

Your comments are very welcome. Thank you!