SoCC'18, Carlsbad, CA, USA

Dynamic and Decentralized Global Analytics via Machine Learning

Hao Wang[1], Di Niu[2], Baochun Li[1]
[1]University of Toronto, [2]University of Alberta

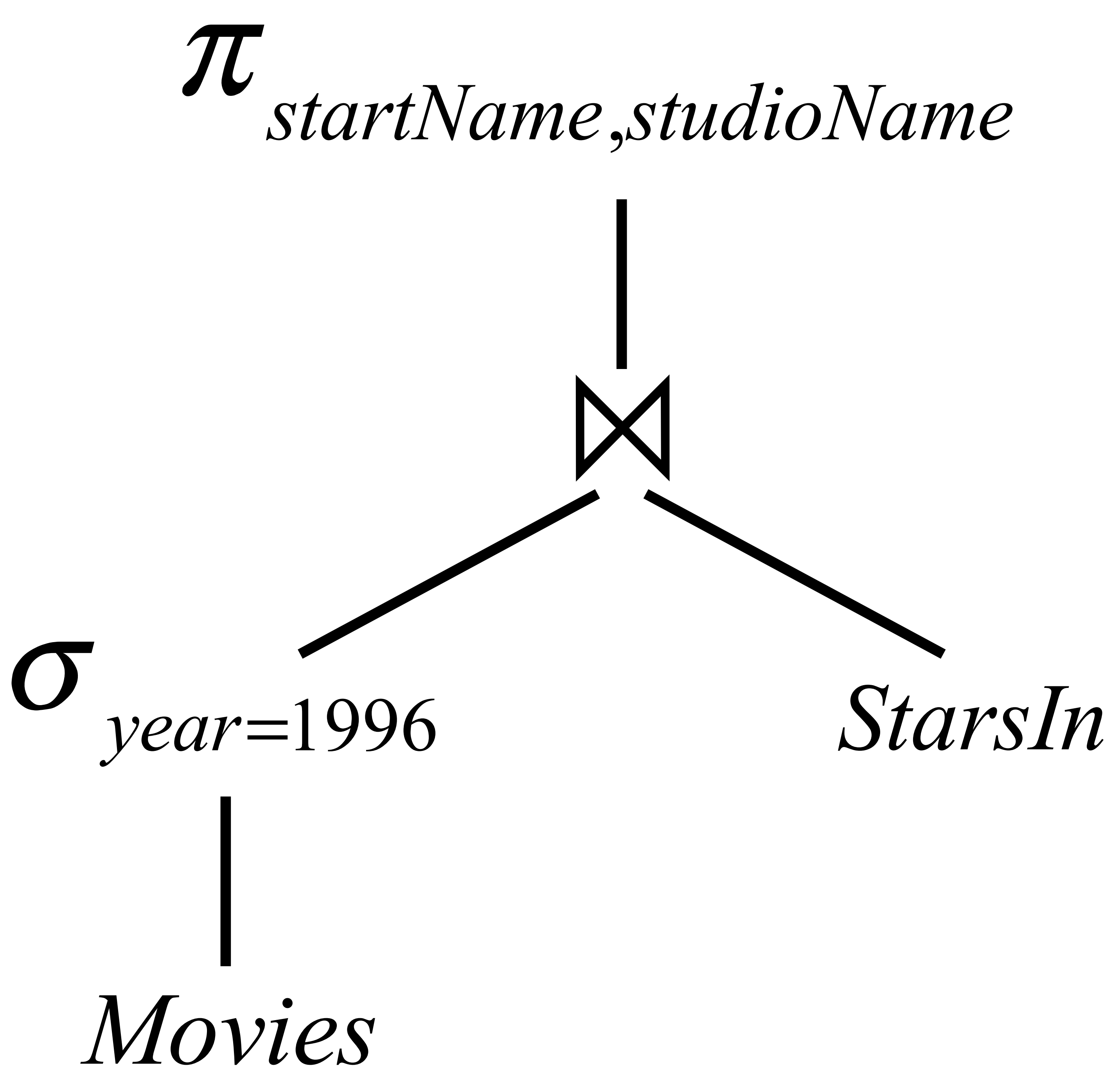
UNIVERSITY OF TORONTO

Query Processing

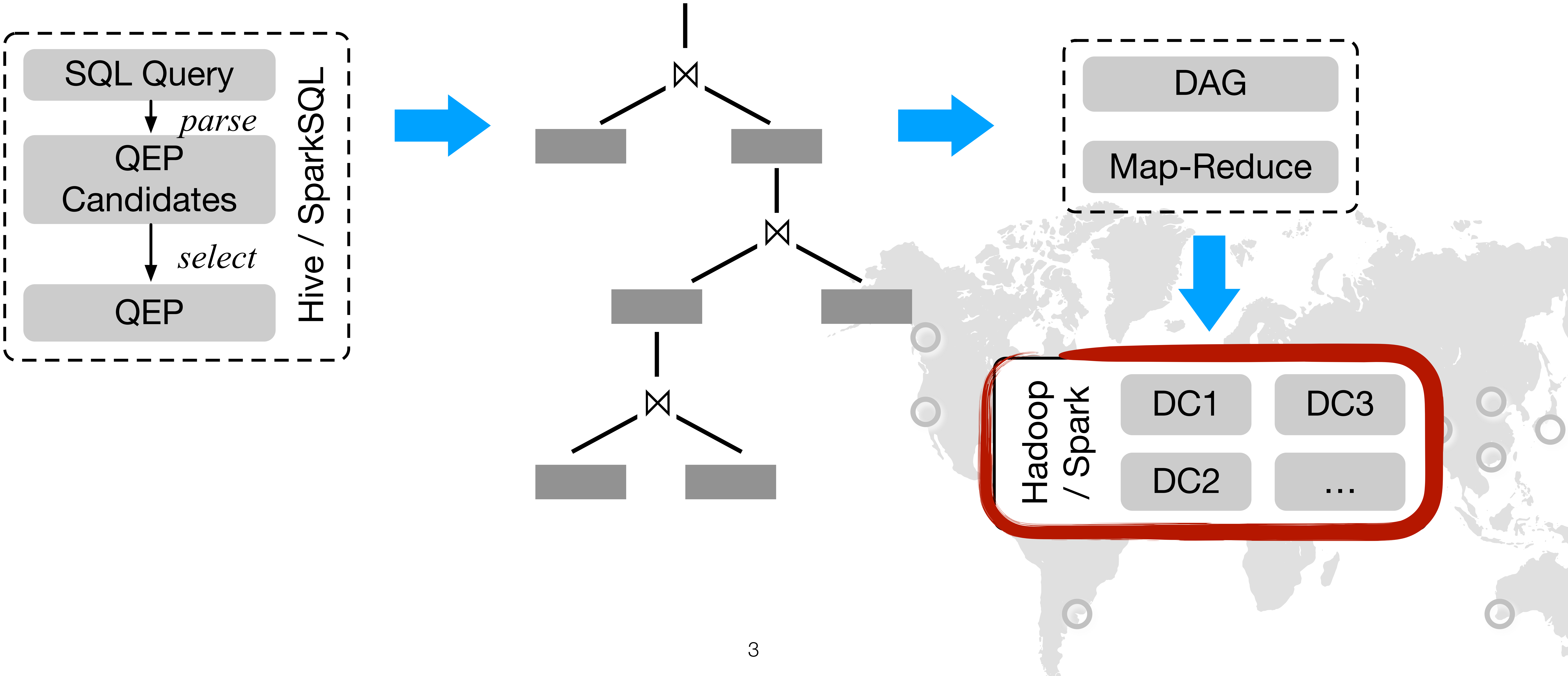
```sql
CREATE VIEW MoviesOf1996 AS
      SELECT *
      FROM Movies
      WHERE year = 1996;

SELECT starName, studioName
FROM MoviesOf1996 JOIN StarsIN;
```

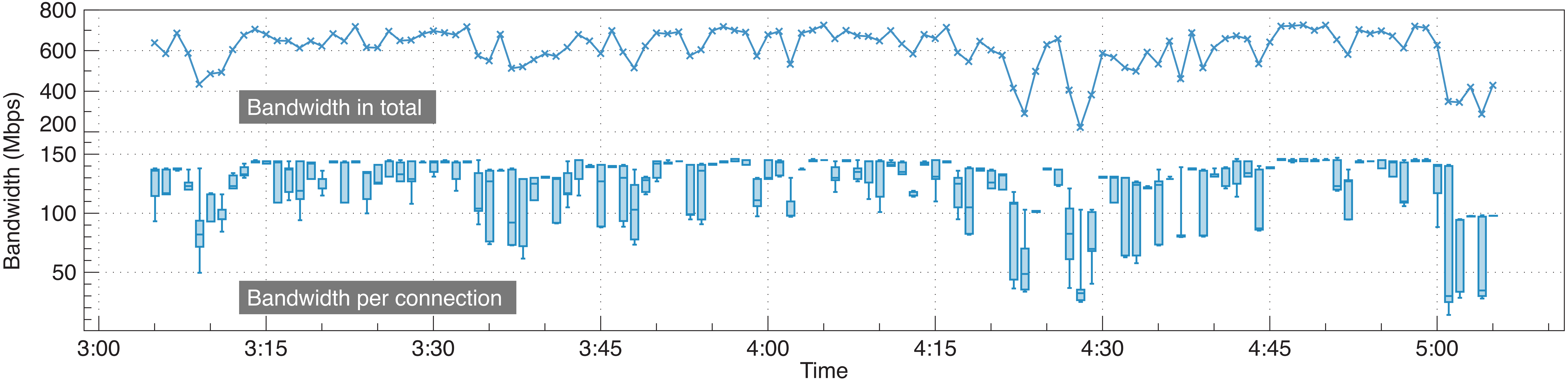
Query Plan



Decentralized Global Analytics



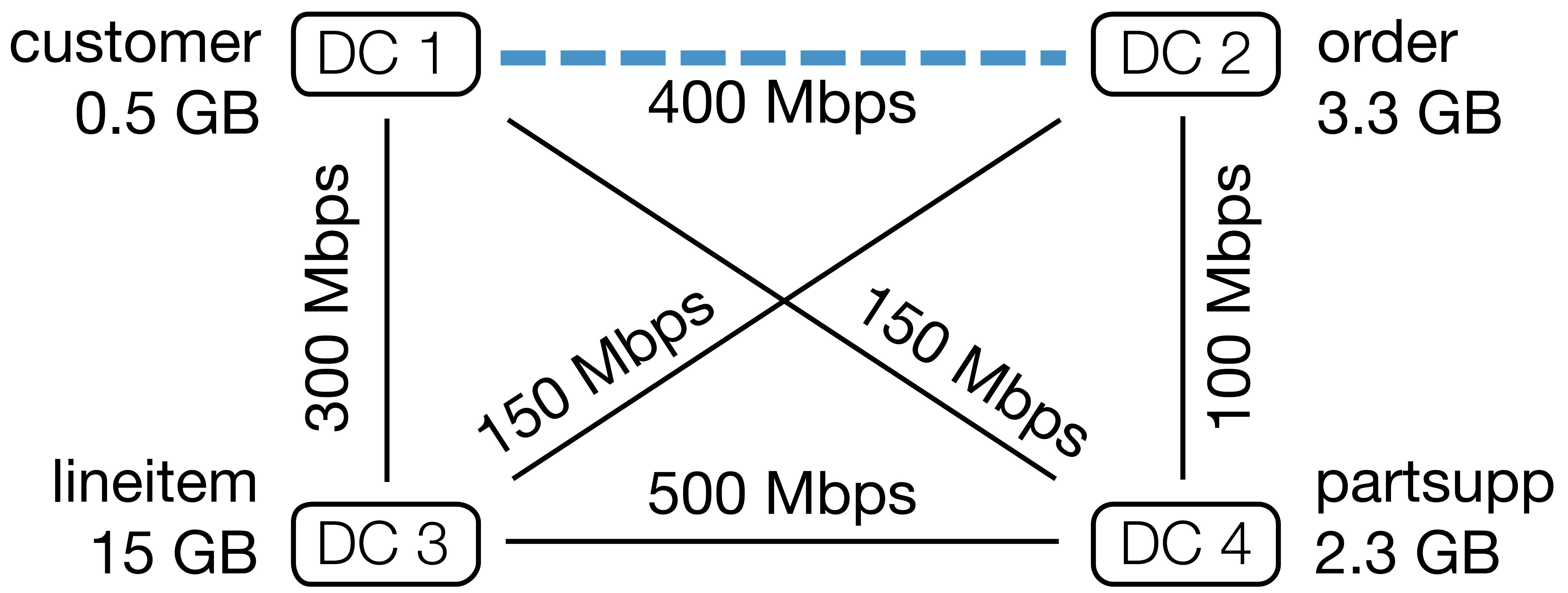
Fluctuating WAN



`iperf -t 10 -P 5` **Google Cloud** Taiwan ⟷ Iowa

A Toy Example

```
SELECT
  C.name, O.orderstatus,
  L.discount, PS.availqty
FROM
  customer as C,
  order as O,
  lineitem as L,
  partsupp as PS
WHERE O.orderkey == L.orderkey,
  AND PS.partkey == L.partkey,
  AND PS.suppkey == L.suppkey,
  AND C.custkey  == O.custkey
```



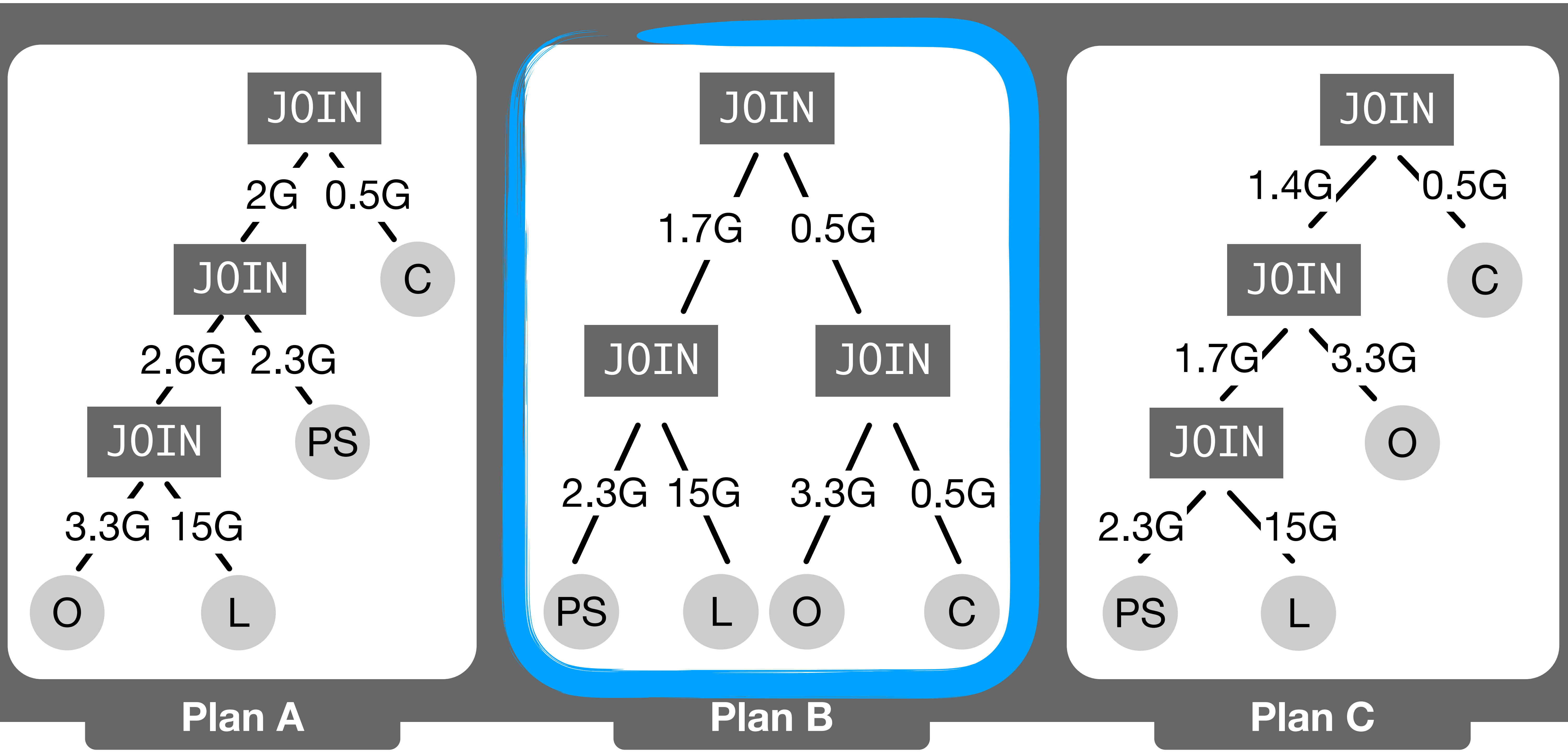
Query Plan Candidates

Plan A

- The worst plan
- The baseline

Plan B

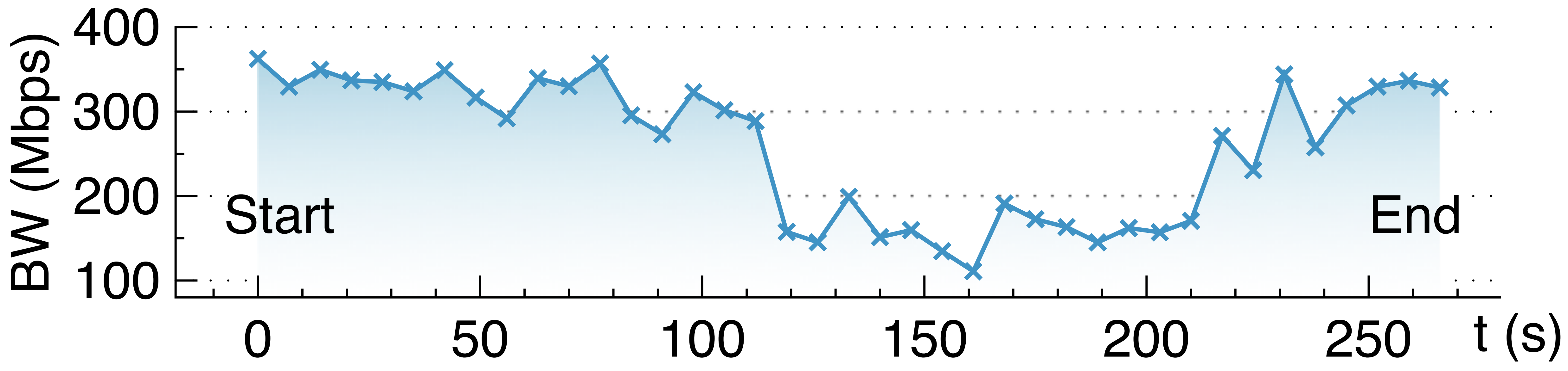
- The initial optimal plan
- Selected by Clarinet

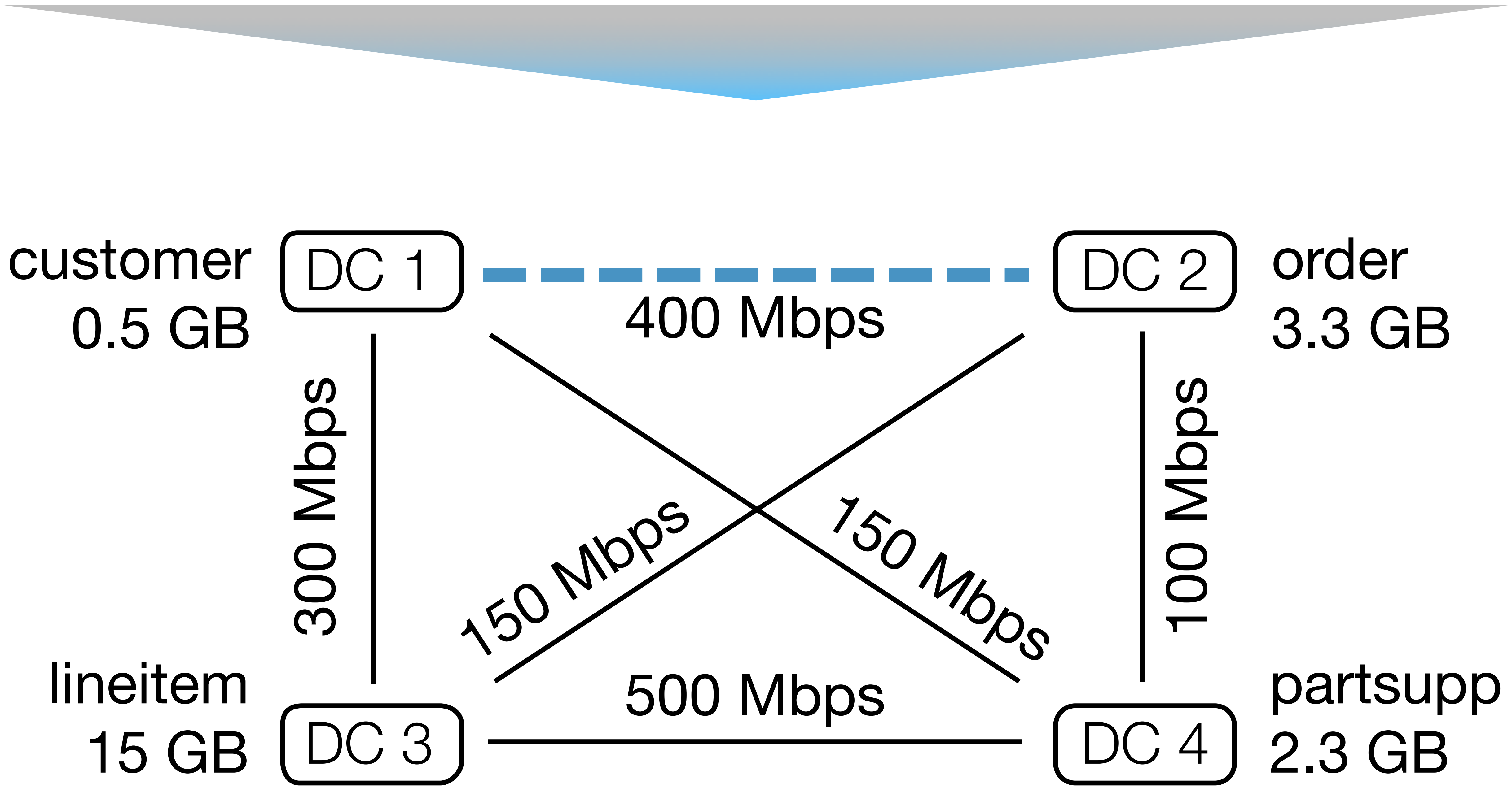


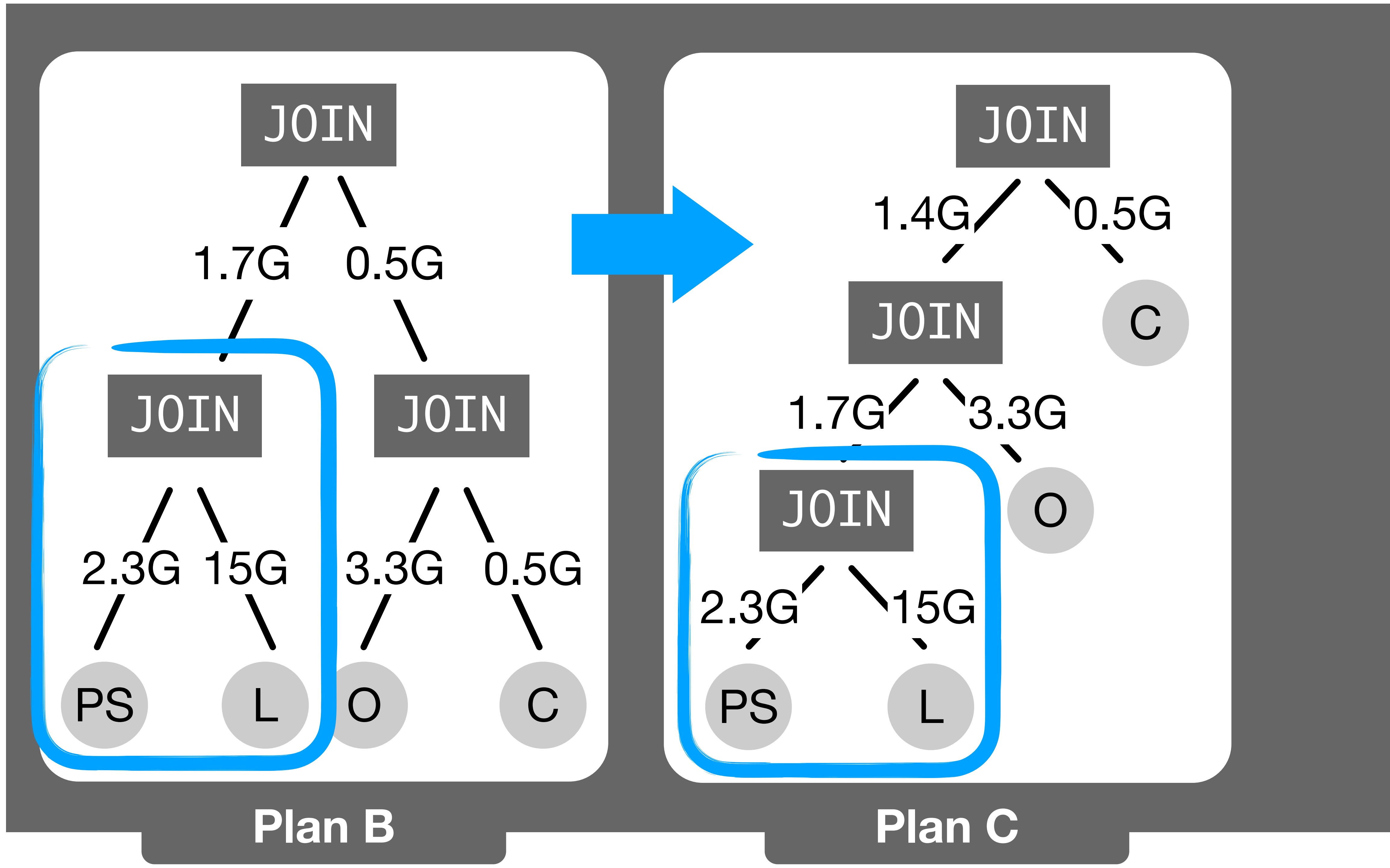
Plan C

- …

A Toy Example







Plan C

- The adjusted plan
- Adapt to bandwidth fluctuation

Query Completion Time

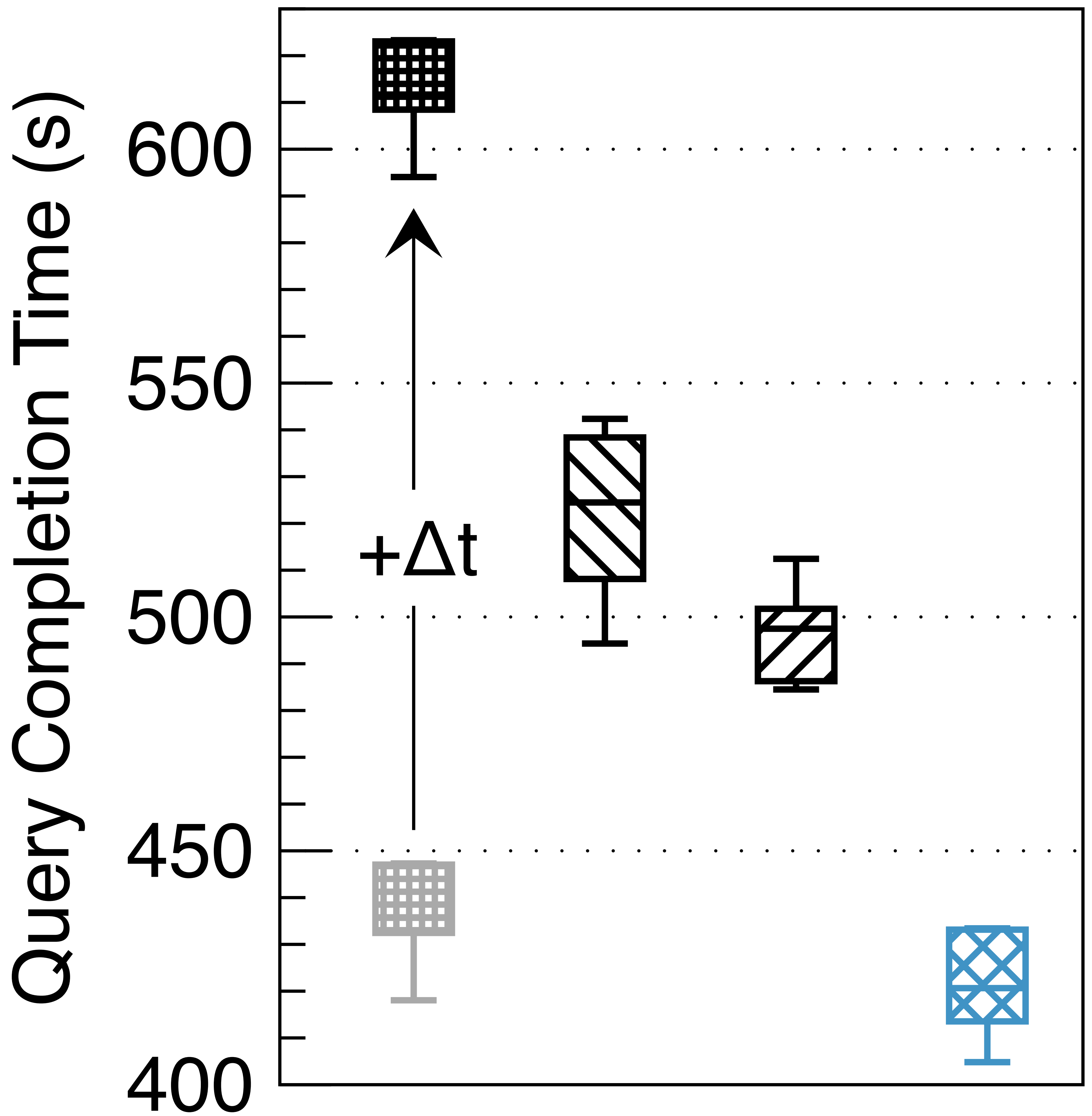
Centralized plan

Baseline (Plan A)

Plan selected by Clarinet (Plan B)

Dynamic adjusted plan (Plan C)

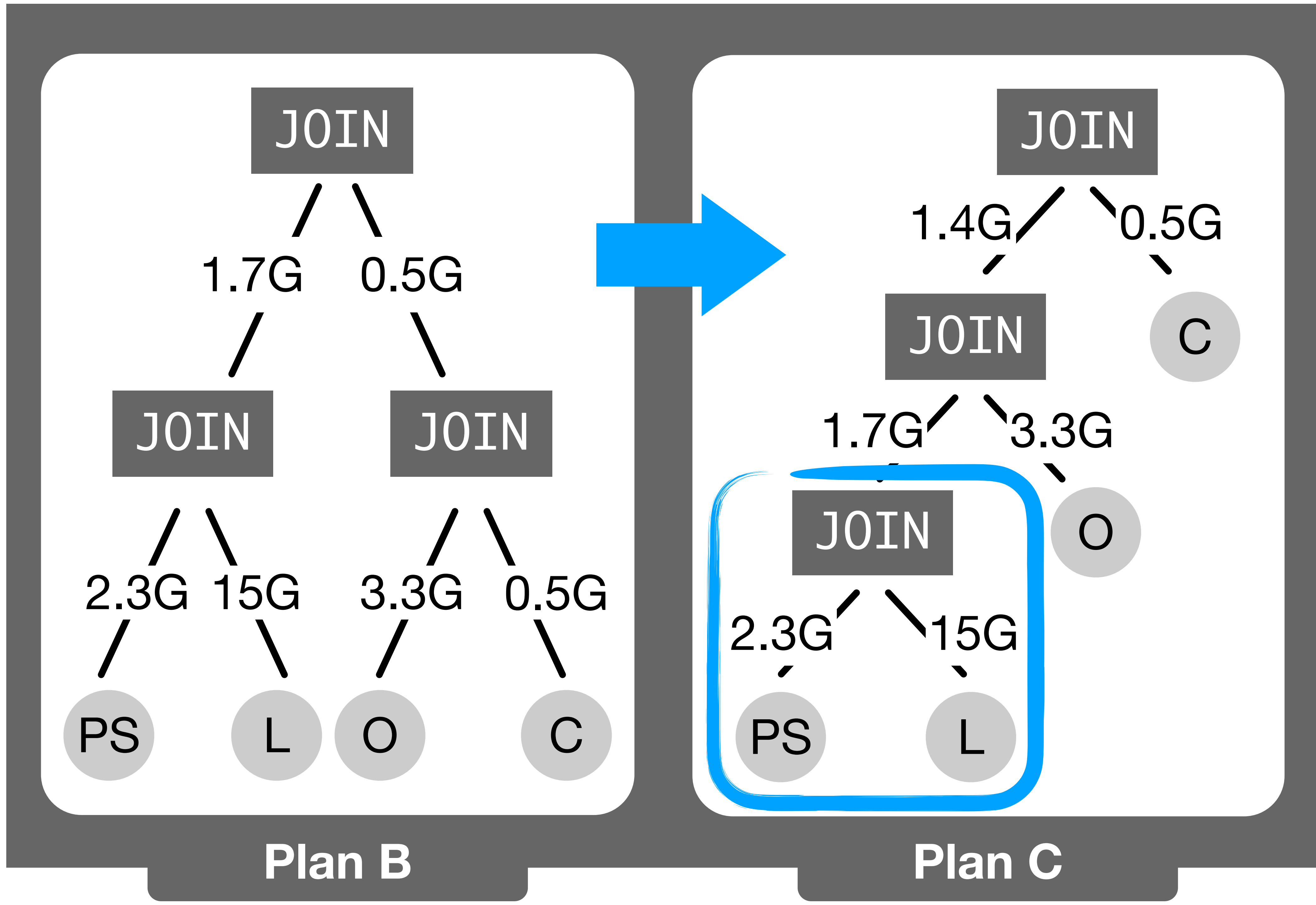
Δt The data movement time



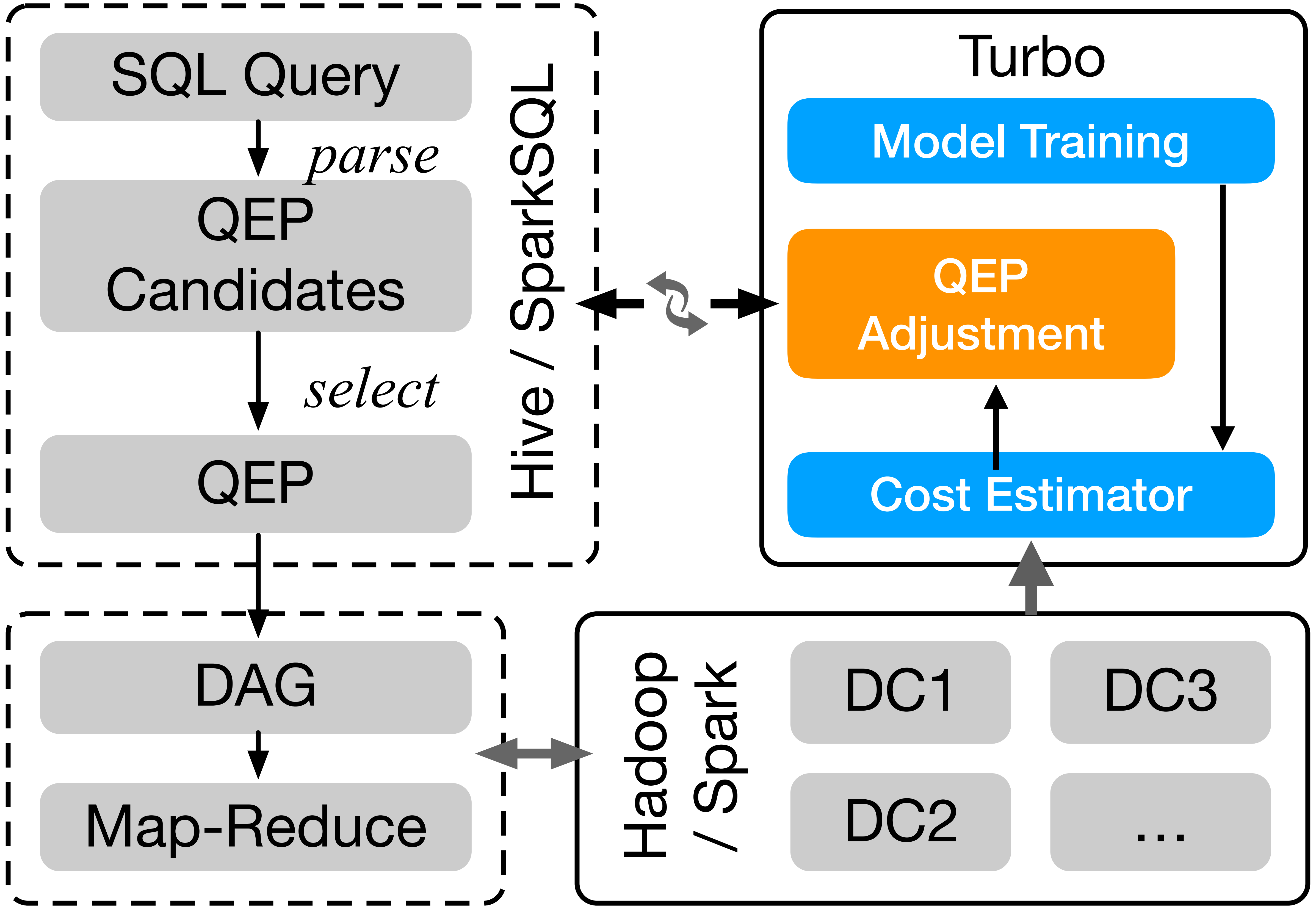
Dynamic Query Planning

Challenges:

- Accurately estimating runtime cost of query plans.
- Minimize overall completion time of queries.



Turbo



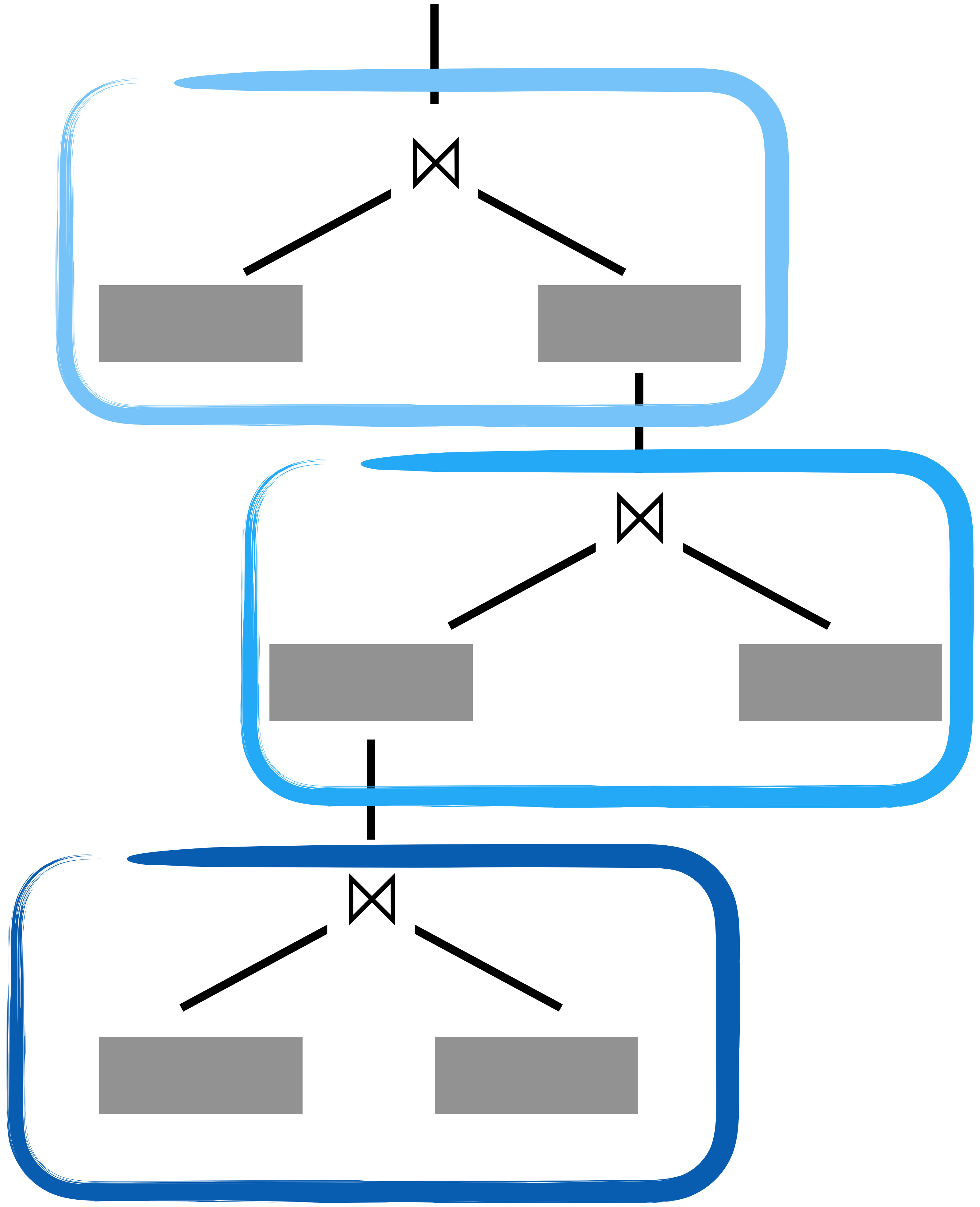
Data

Planning

Model

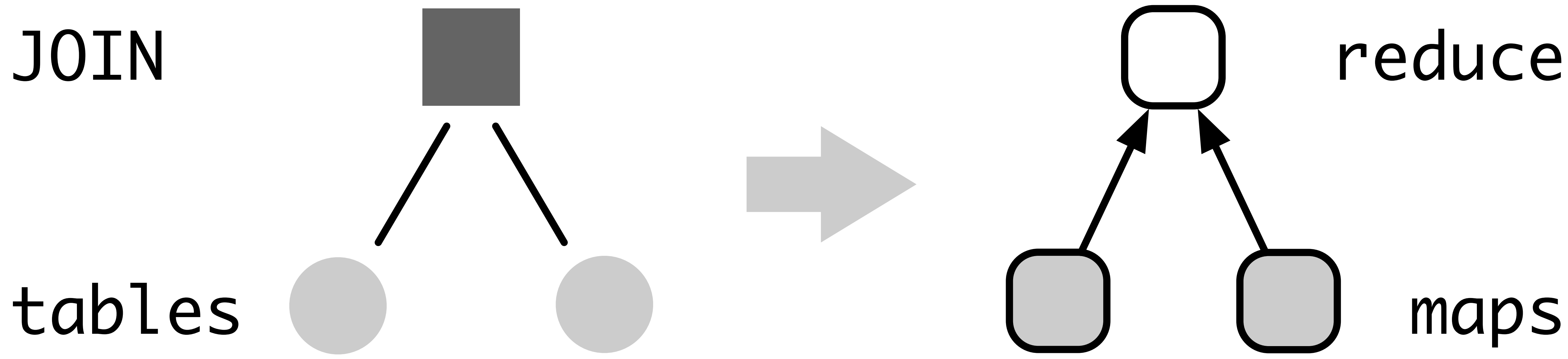
Evaluation

Prediction Target



`(duration, output size)`

Data Generation



1.Operator —> Map stage

$\sigma_{price>100}$ ➡ `filter(order o=>(o.price>100))`

2. Operator —> MapReduce stages

$customer \bowtie orders$ ➡

```
map(customer c=>(c.custkey, c.values))

map(order o=>(o.custkey, o.values))

reduce(custkey, values)
```

Data Generation

15K records

Raw Features	Range
total_exec_num	1 – 16
cpu_core_num	1 – 8 per executor
mem_size	1 – 4 GB per executor
avail_bw	5 – 1000 Mbps per link
tbl1_size, tbl2_size	0.3 – 12 GB per table
hdfs_block_num	1 – 90

Data Preprocessing

1. Handcrafting features

2. Polynomial feature crossing

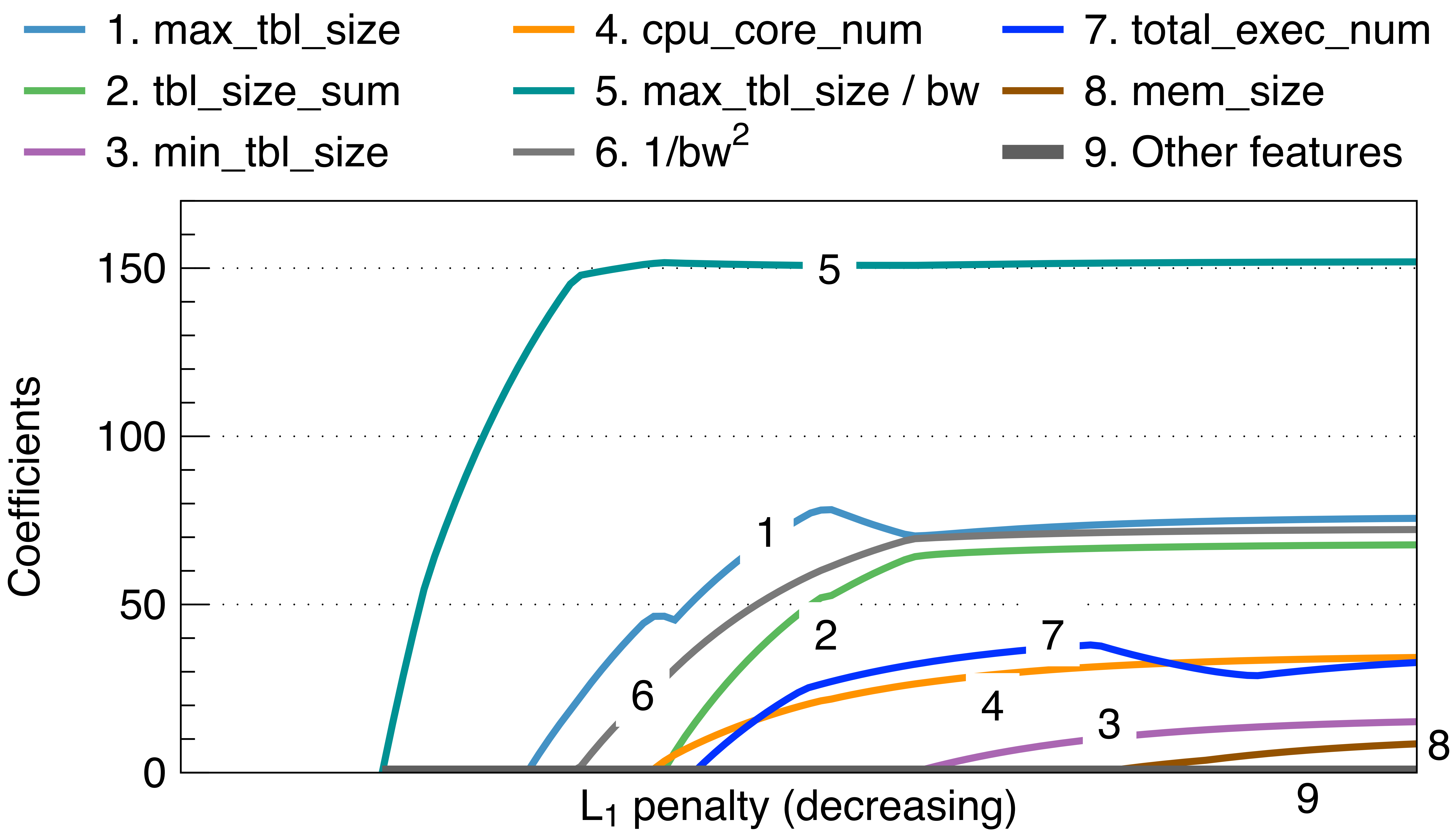
3. Feature selection by LASSO path

$$[a, b, c] \rightarrow [1, a, b, c, a^2, ab, ac, b^2, bc, c^2]$$

Handcrafted Features
tbl_size_sum = sum(tbl1_size, tbl2_size)
max_tbl_size = max(tbl1_size, tbl2_size)
min_tbl_size = min(tbl1_size, tbl2_size)
1/avail_bw, 1/total_exec_num, 1/cpu_core_num

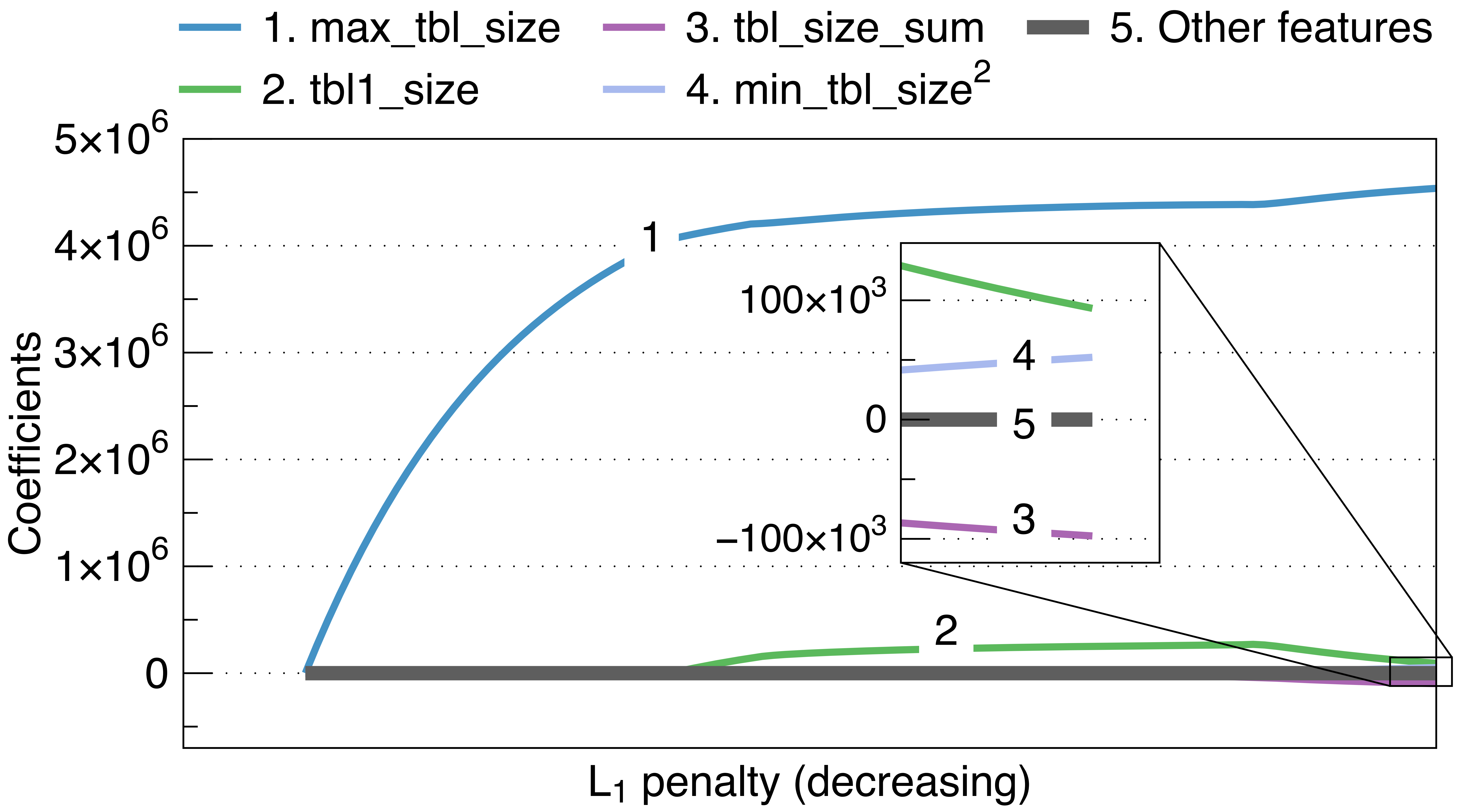
Feature Selection

duration



Feature Selection

output size



Training

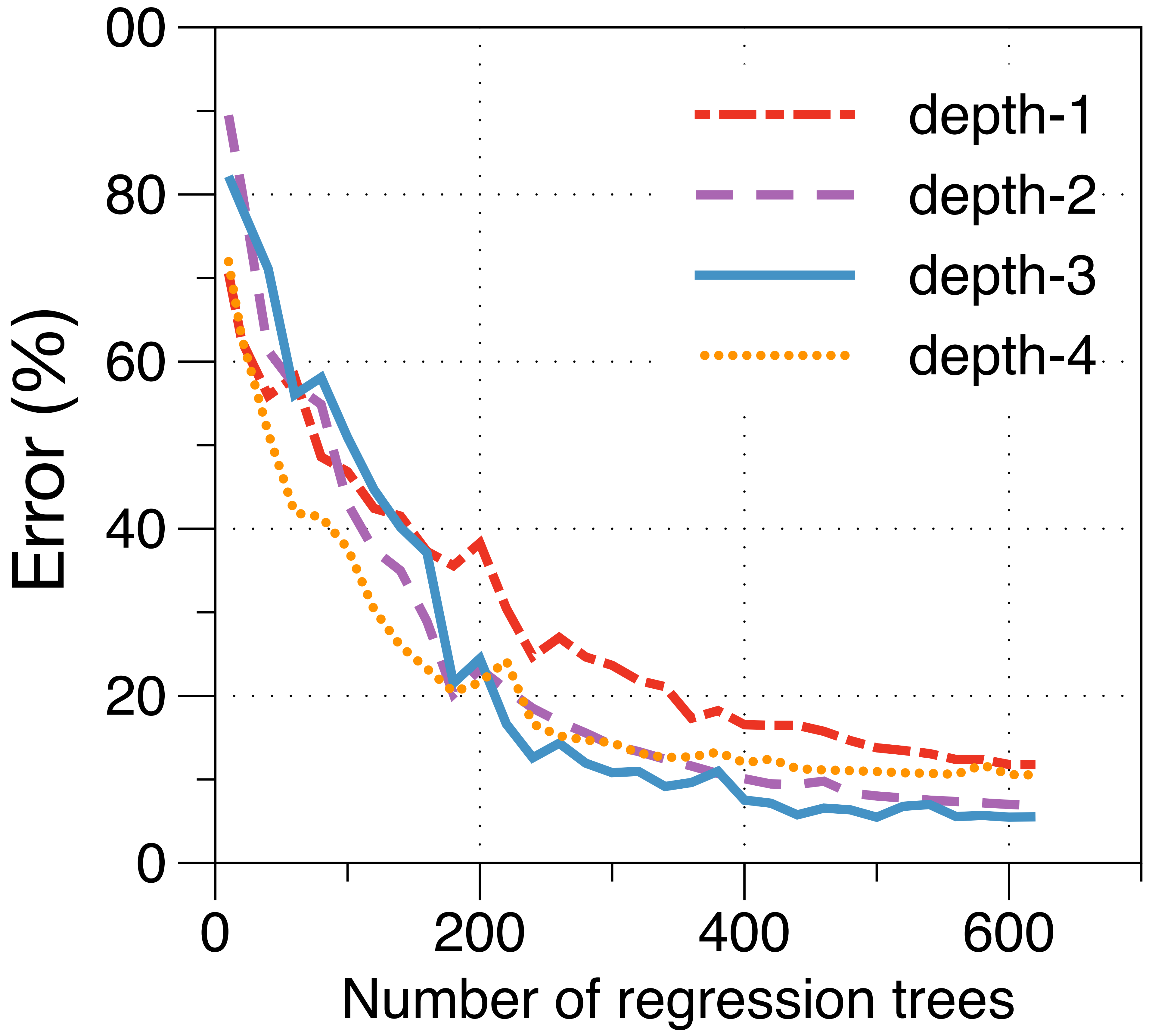
LASSO Regression

Linear Regression with L1 penalty

GBRT

Gradient Boosting Regression Tree

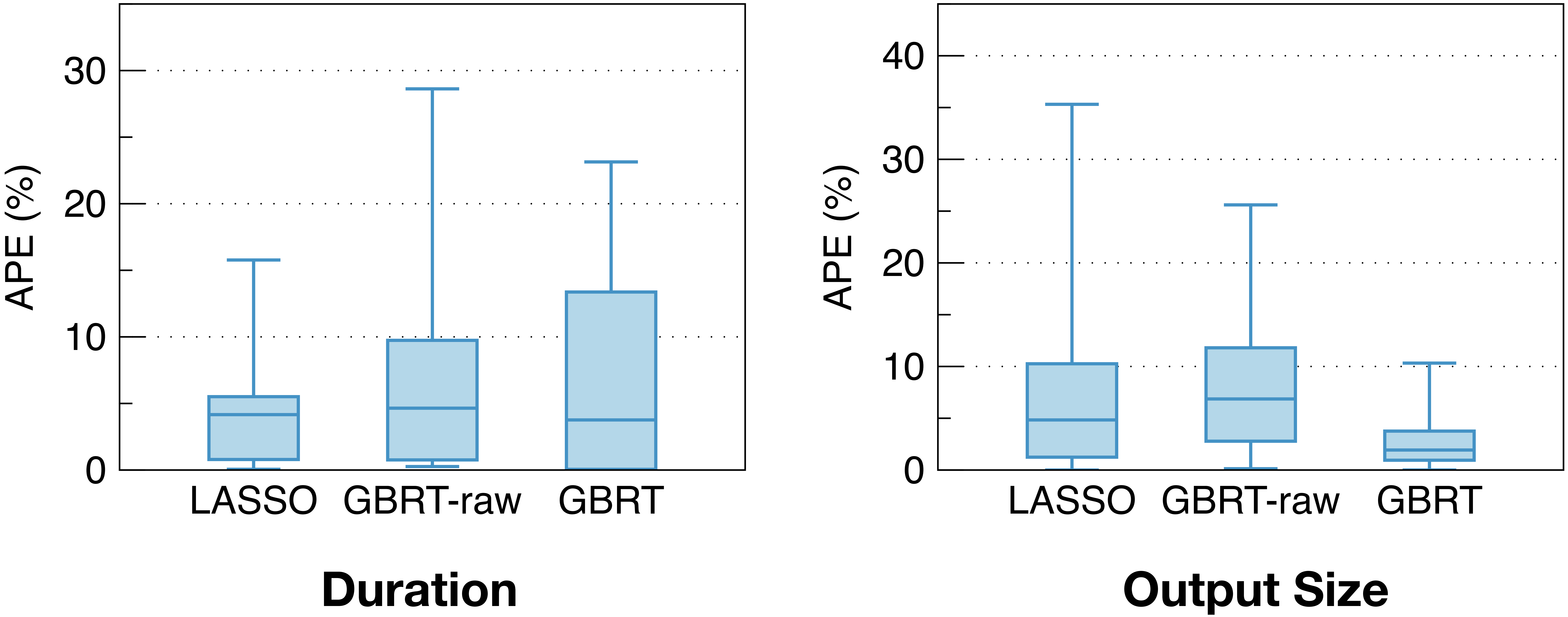
500 ternary regression trees of depth 3



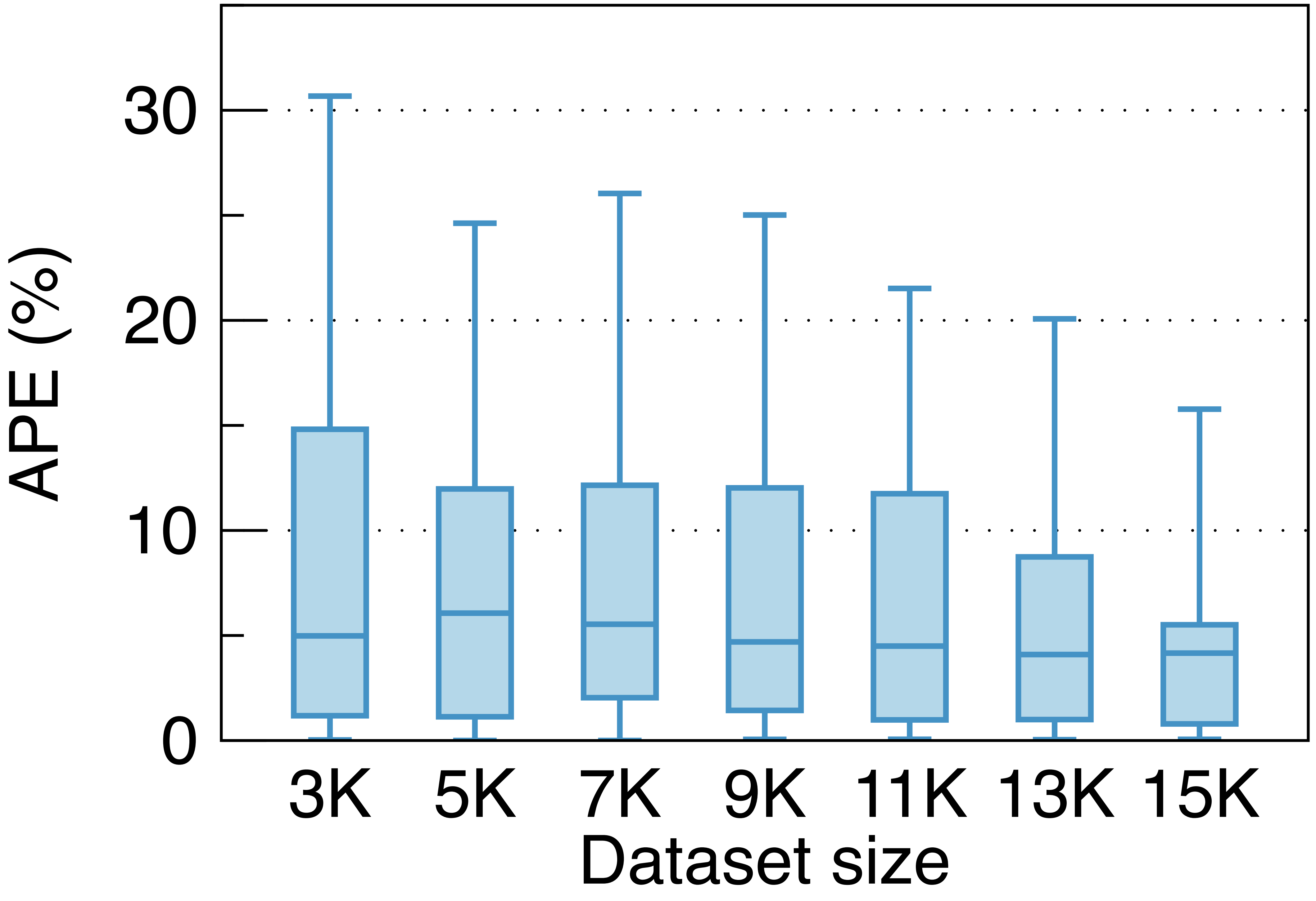
Model Test

Absolute Percentage Error:

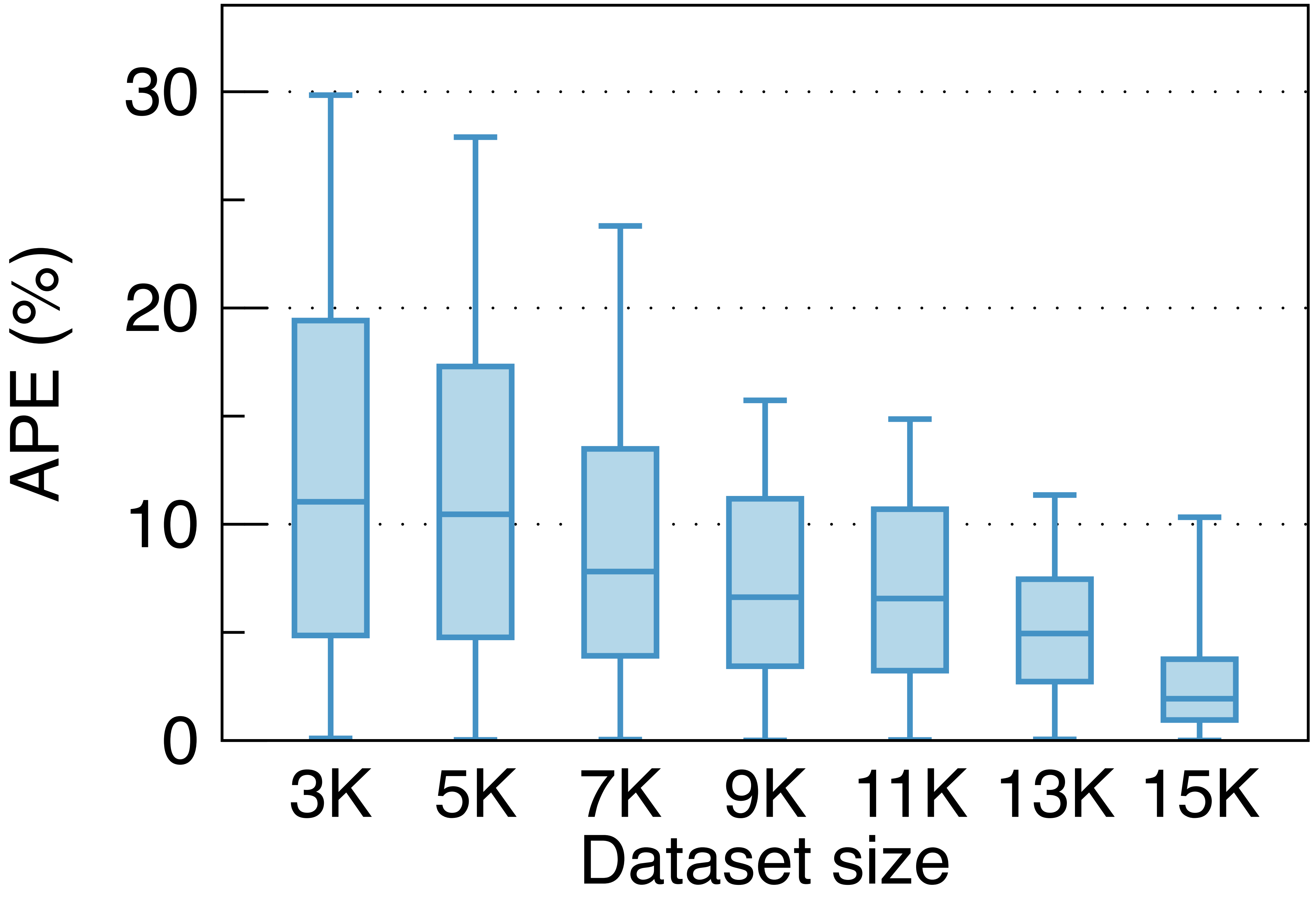
$$APE_i = \frac{|y_i - h(x_i)|}{y_i} \times 100\%.$$



Model Test



Duration



Output Size

Dynamic Planning Strategies

- **Shortest Completion Time First (SCTF)**

  `duration`

- **Maximum Data Reduction First (MDRF)**

  `data_reduction`

- **Maximum Data Reduction Rate First (MDRRF)**

  `data_reduction / duration`

Evaluation Setup

- TPC-H benchmark
- Google Cloud
  - 33 instances across 8 regions

Table	**Location**	**Table**	**Location**
`lineitem`	Taiwan	`customer`	Frankfurt
`region`	Singapore	`orders`	Sao Paulo
`supplier`	Sydney	`nation`	Northern Virginia
`part`	Belgium	`partsupp`	Oregon

Query

Turbo-SCTF

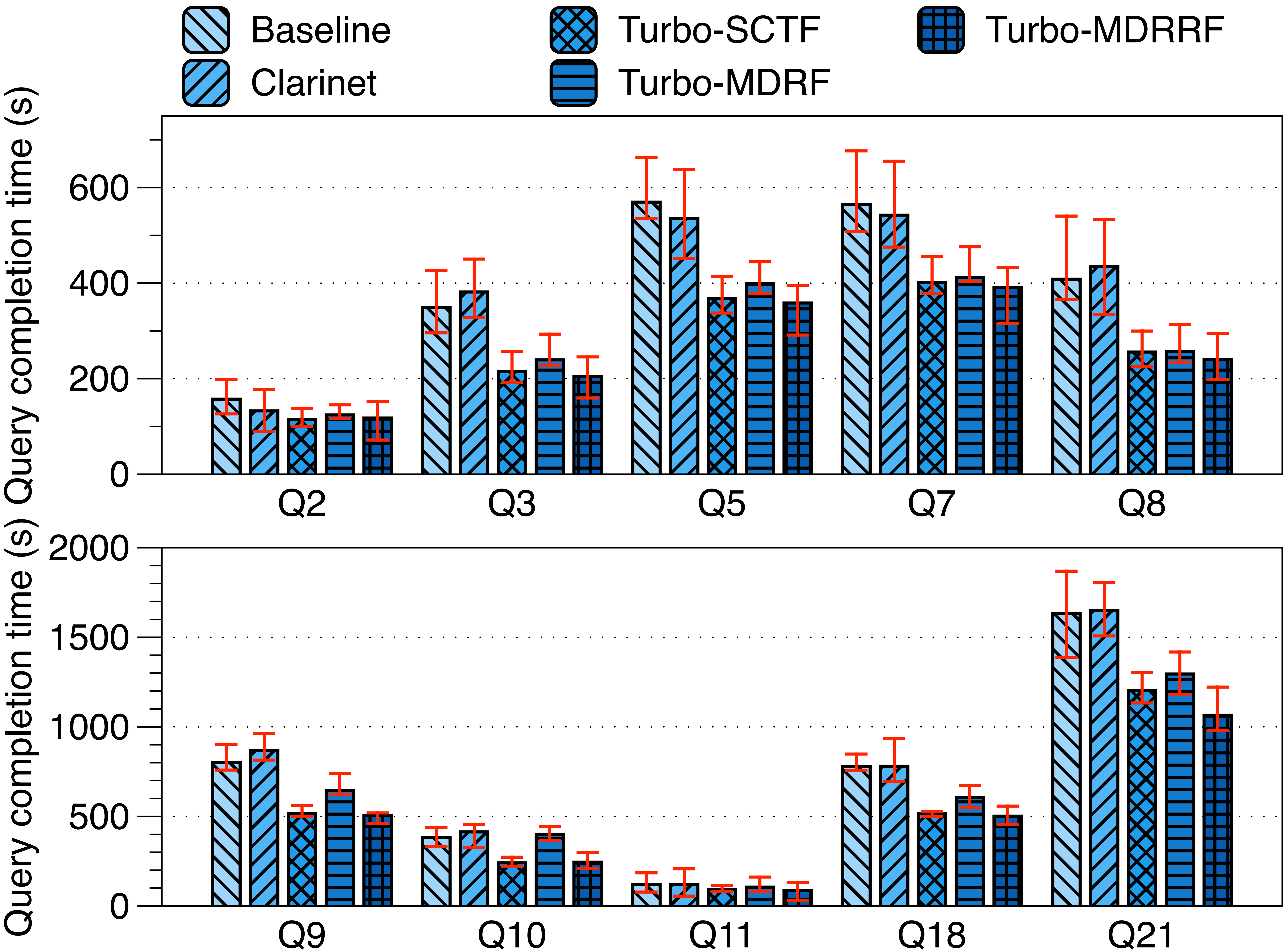
- 25.1-38.5%

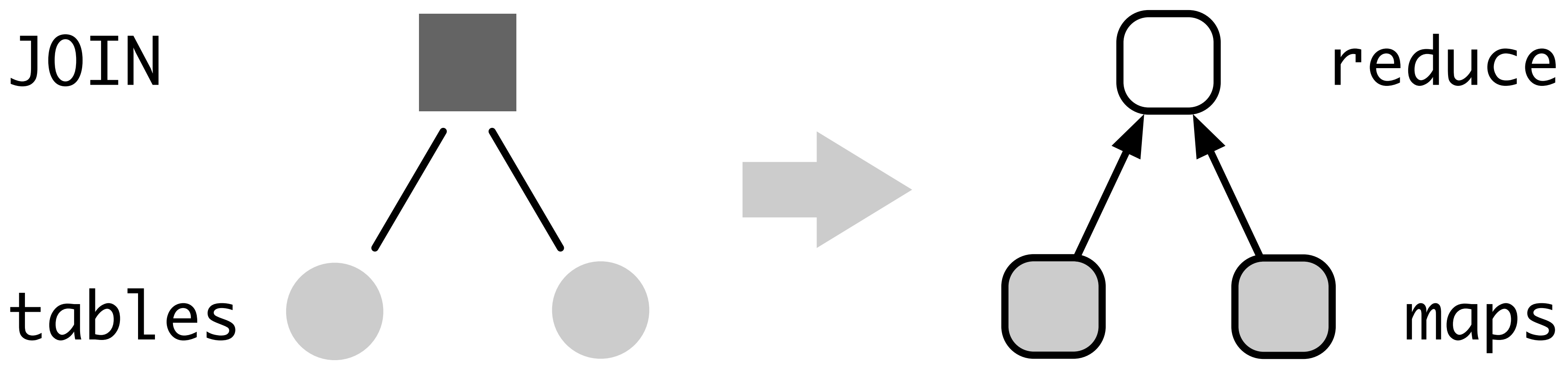
Turbo-MDRF

- 12.6-37.1%

Turbo-MDRRF

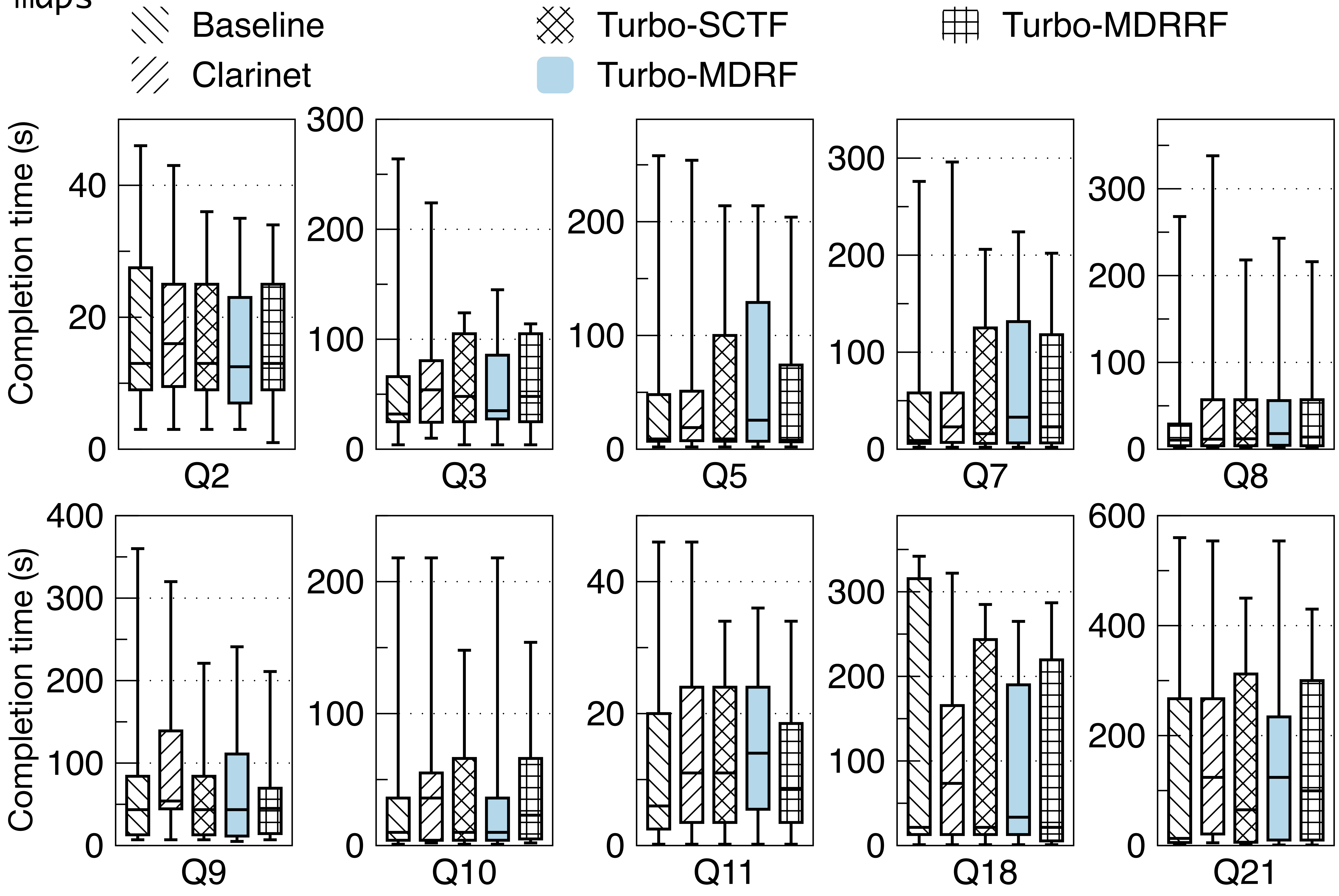
- 25.2-41.4%





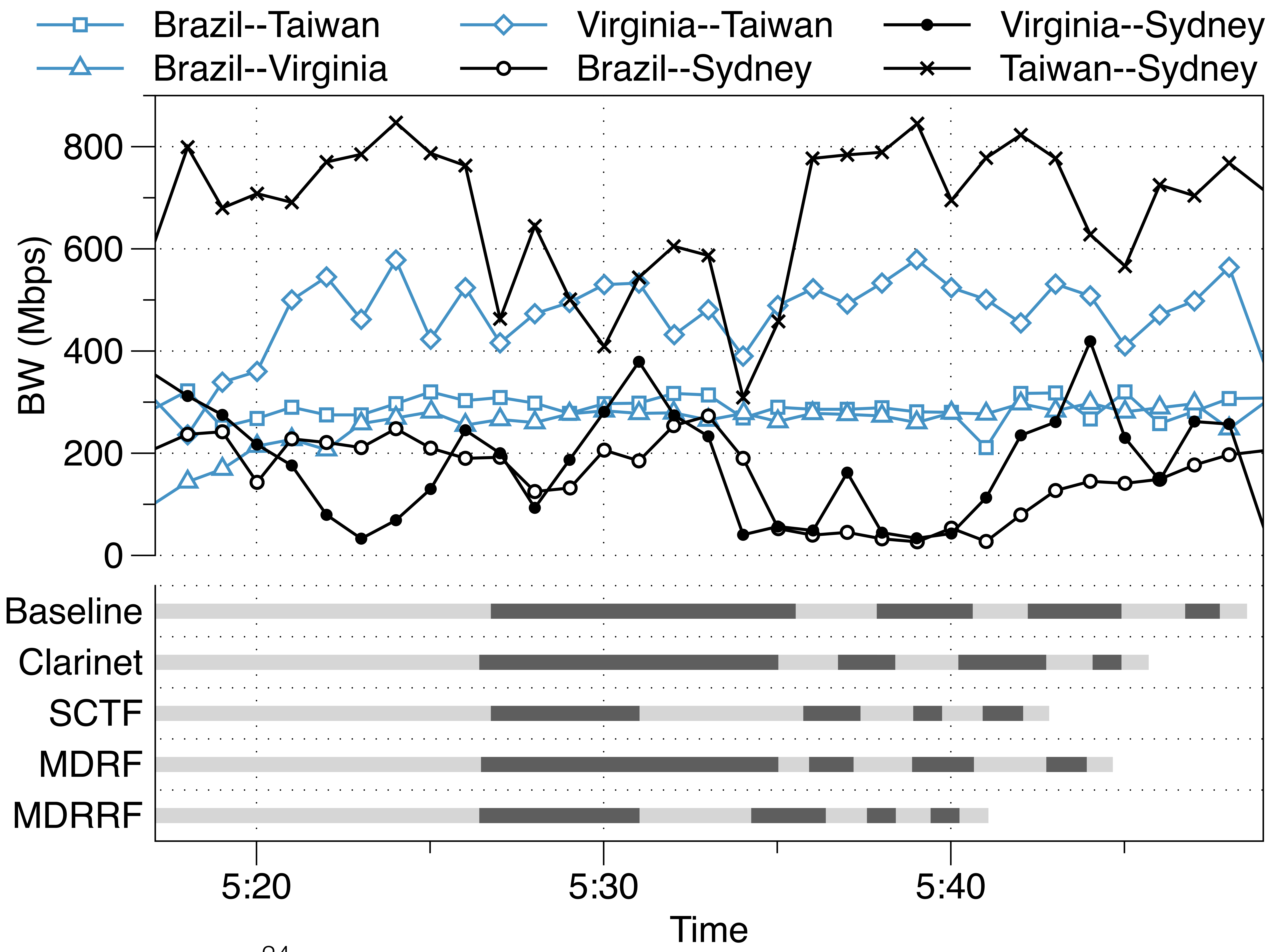
Pairwise Join

The completion time distributions of pairwise joins.



Case Study

The Gantt chart of the query Q21



Related Work

Work	Data Placement	Task Scheduling	Plan Optimization	Working Mode
Geode [26]		√	√	static
WANanalytics [27]	√	√		static
Iridium [20]	√	√		static
SWAG [16]		√		static
JetSteam [21]	√			static
Clarinet [25]	√	√		static
Lube [15]		√		dynamic
Graphene [14]		√		static
Turbo			√	dynamic

Conclusion

- Turbo: dynamic query planning with awareness of WAN bandwidths
- Data-driven cost estimation of pairwise join with accuracy over 95%
- Greedy strategies that reduces the query completion times by up to 41% based on the TPC-H benchmark


The End
Thank You