Clarinet: WAN-Aware Optimization for Analytics Queries

Raajay Viswanathan, Ganesh Ananthanarayanan, Aditya Akella

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Microsoft® Research

Overview

- Web apps hosted on multiple DCs → Low latency access to end-user

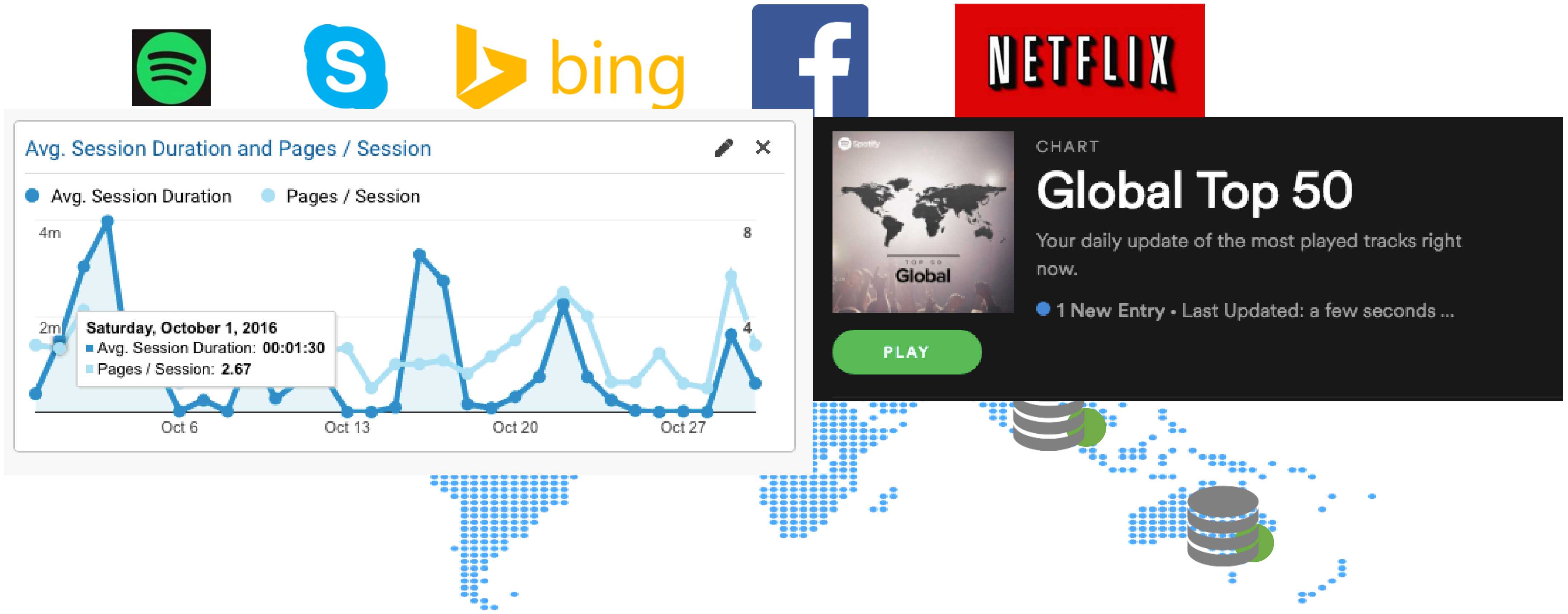
Overview

- Web apps hosted on multiple DCs → Low latency access to end-user

Overview

- Web apps hosted on multiple DCs → Low latency access to end-user

Overview



Avg. Session Duration and Pages / Session

● Avg. Session Duration ● Pages / Session

Saturday, October 1, 2016
Avg. Session Duration: 00:01:30
Pages / Session: 2.67

CHART

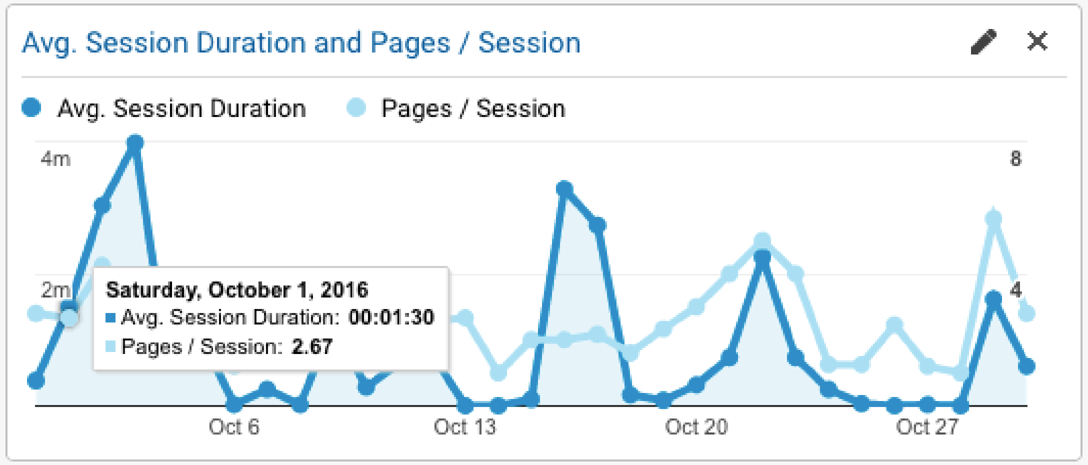
Global Top 50

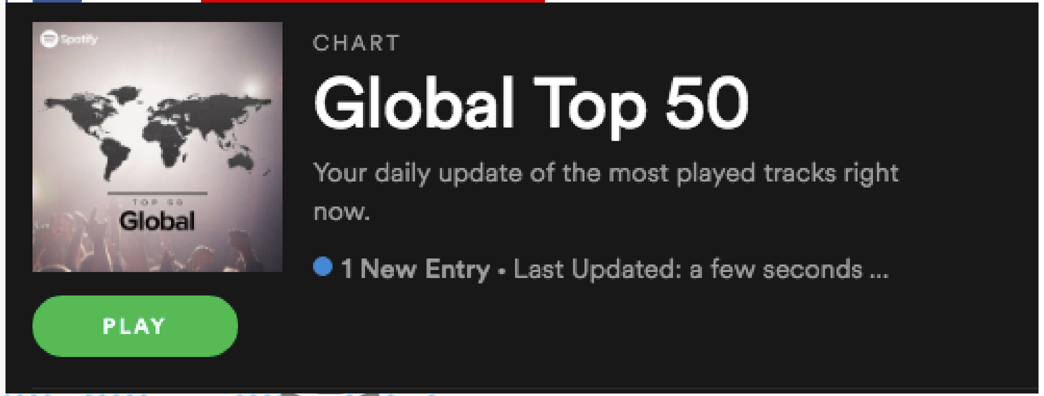
Your daily update of the most played tracks right now.

● 1 New Entry • Last Updated: a few seconds ...

- Web apps hosted on multiple DCs → Low latency access to end-user

Overview





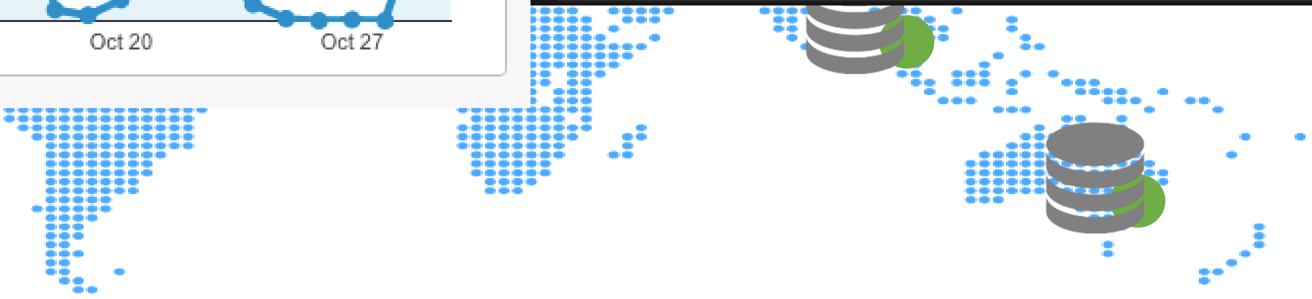
CHART

Global Top 50

Your daily update of the most played tracks right now.

● 1 New Entry • Last Updated: a few seconds ...

PLAY

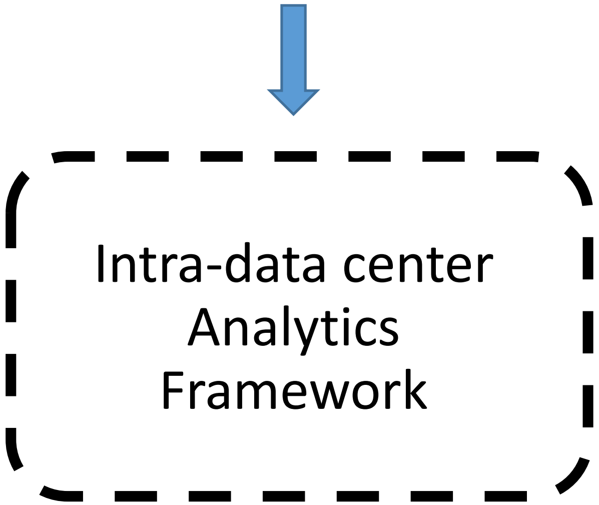


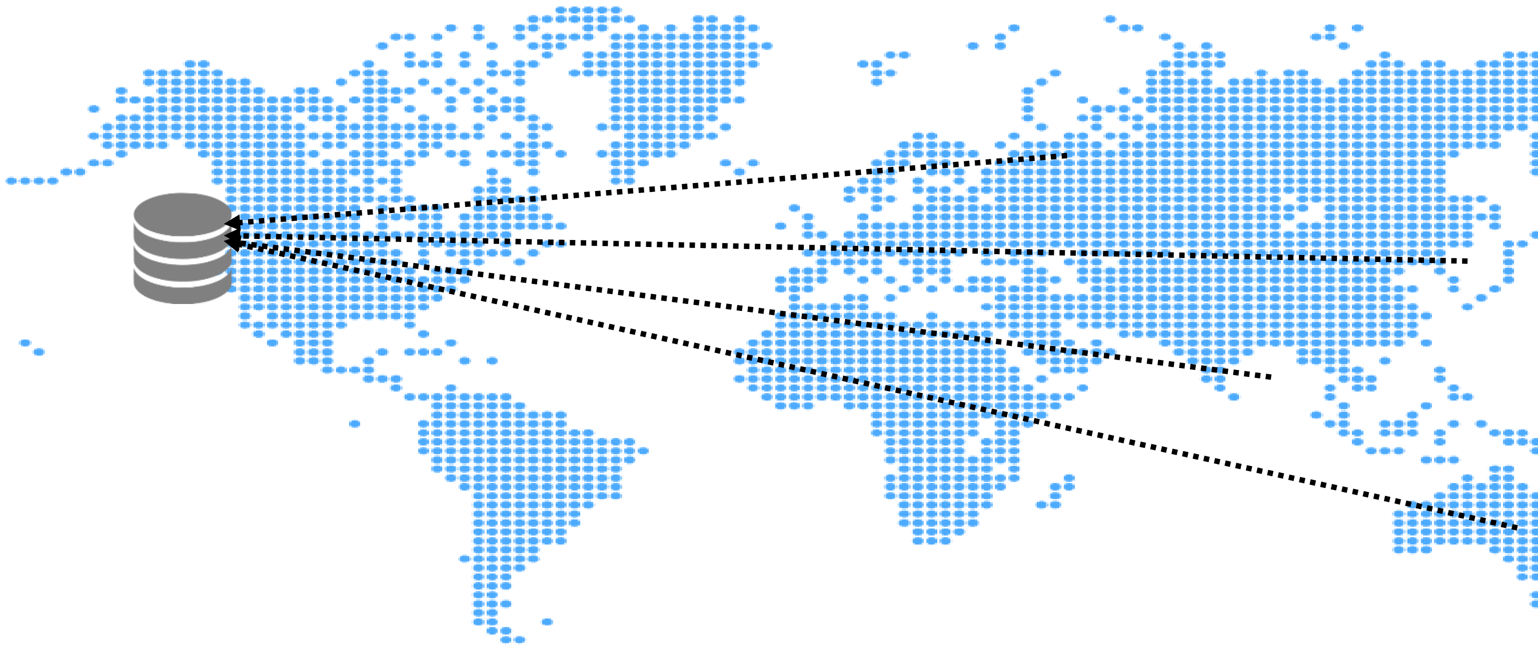
- Web apps hosted on multiple DCs → Low latency access to end-user
- Need efficient methods to analyze data located in **multiple** data centers

Centralized Aggregation is Wasteful

Centralized Aggregation is Wasteful

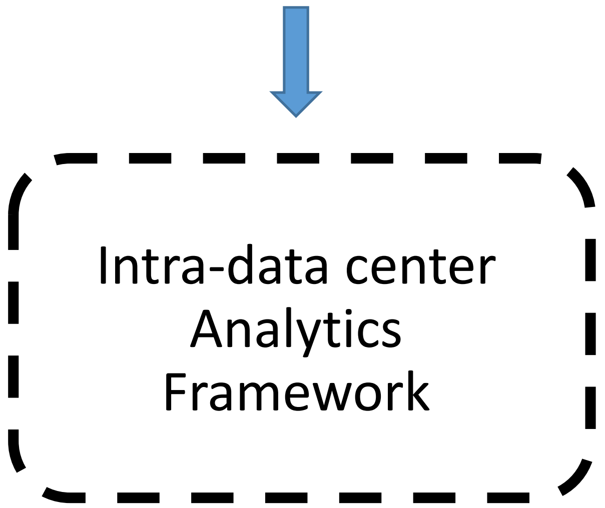
```
SELECT * … FROM .. WHERE .. ;
```

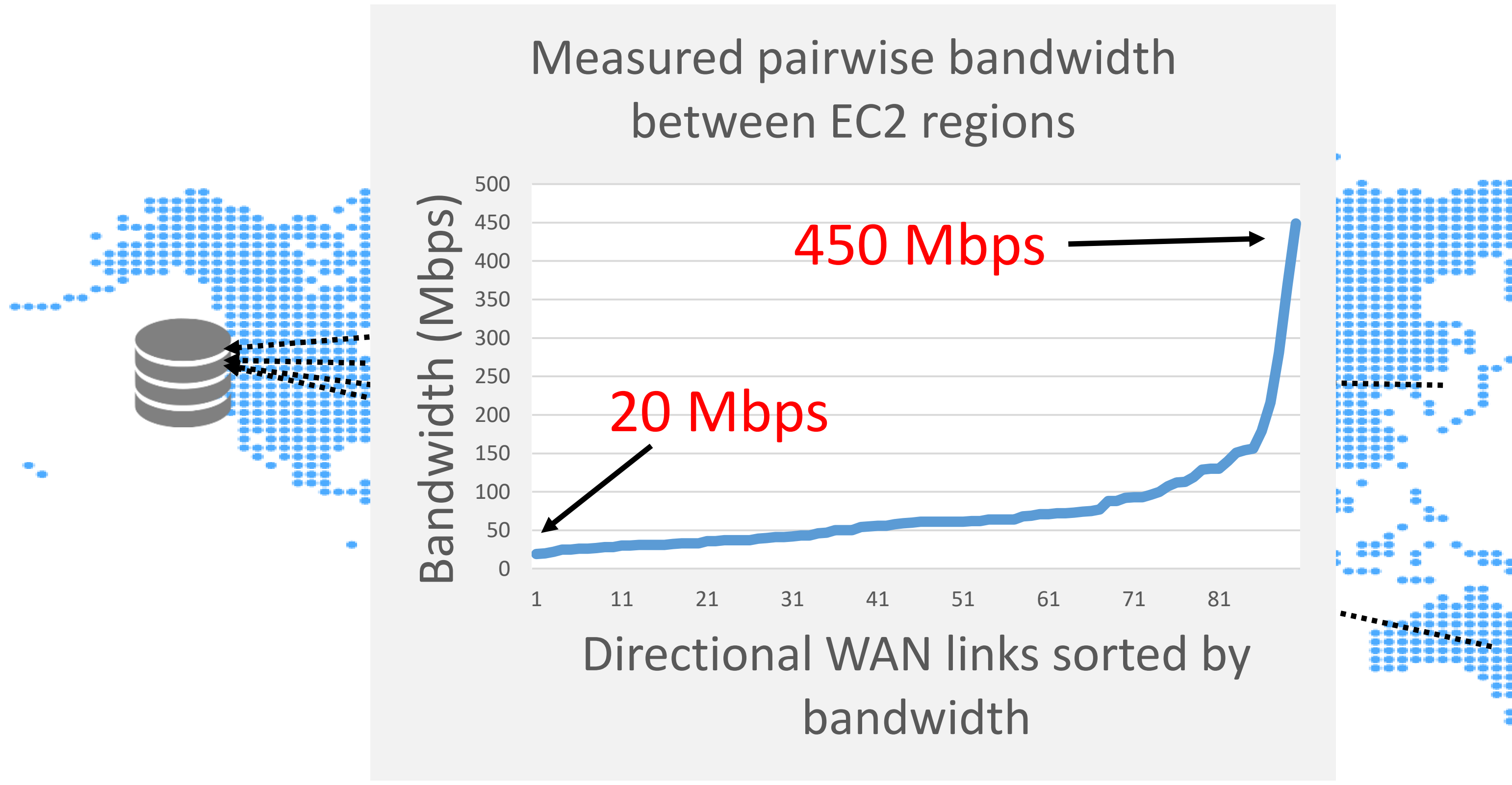




Centralized Aggregation is Wasteful

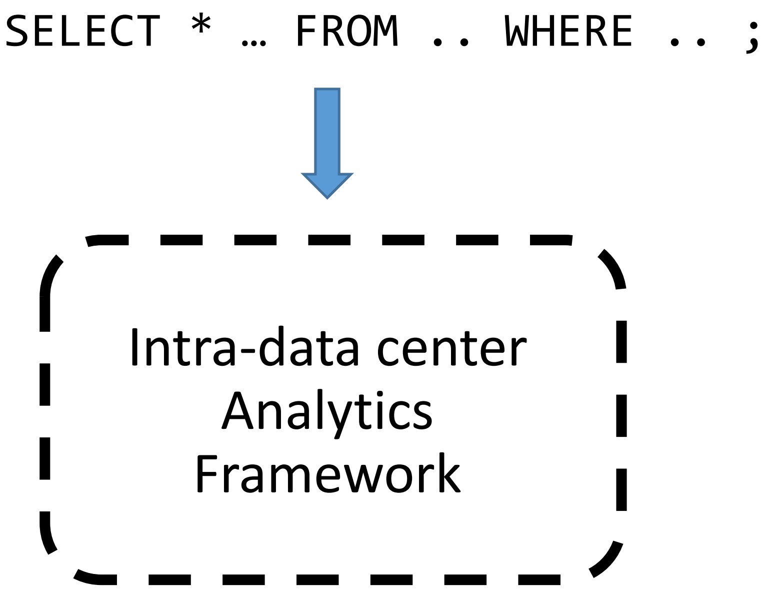
```
SELECT * … FROM .. WHERE .. ;
```

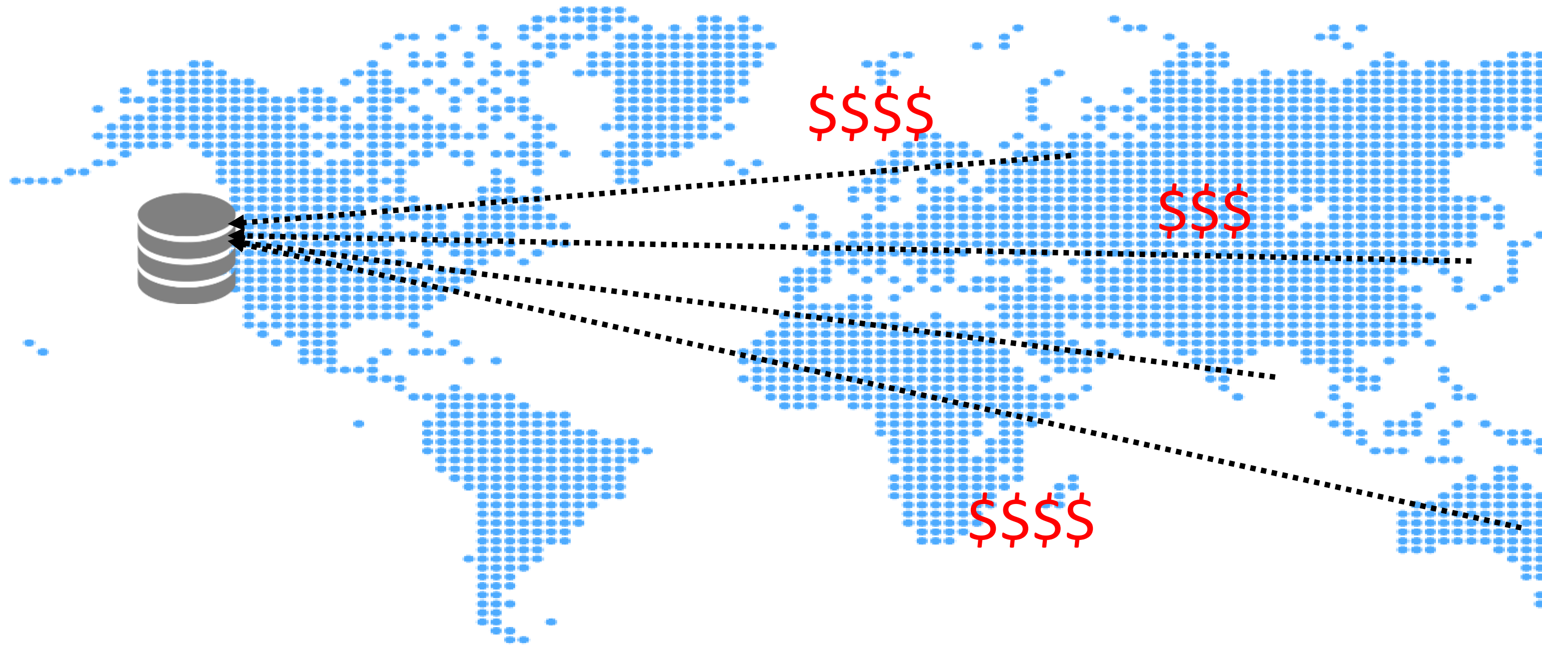




- Available WAN bandwidth is limited → Aggregation latency overhead

Centralized Aggregation is Wasteful

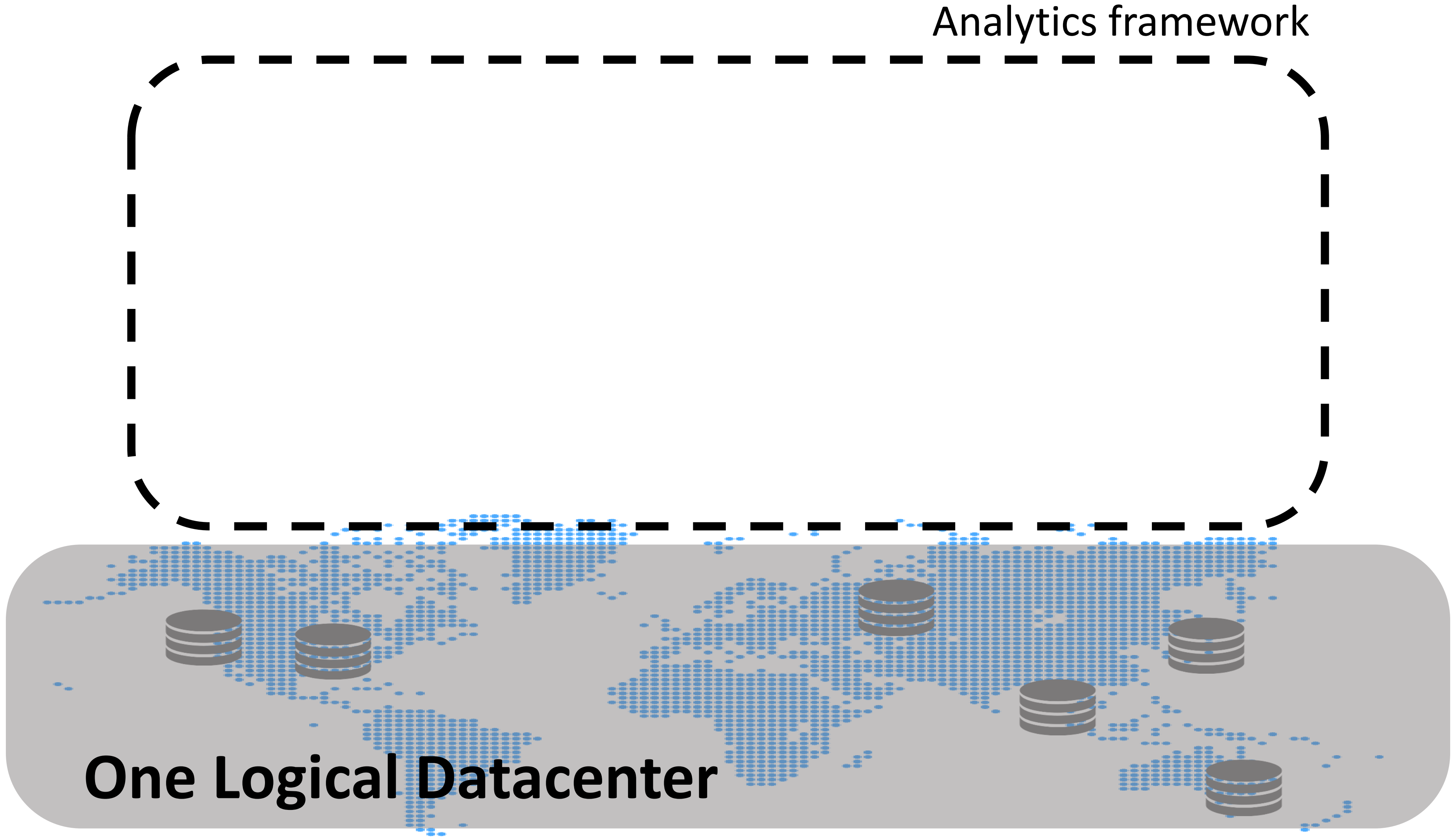

SELECT * … FROM .. WHERE .. ;
Intra-data center Analytics Framework



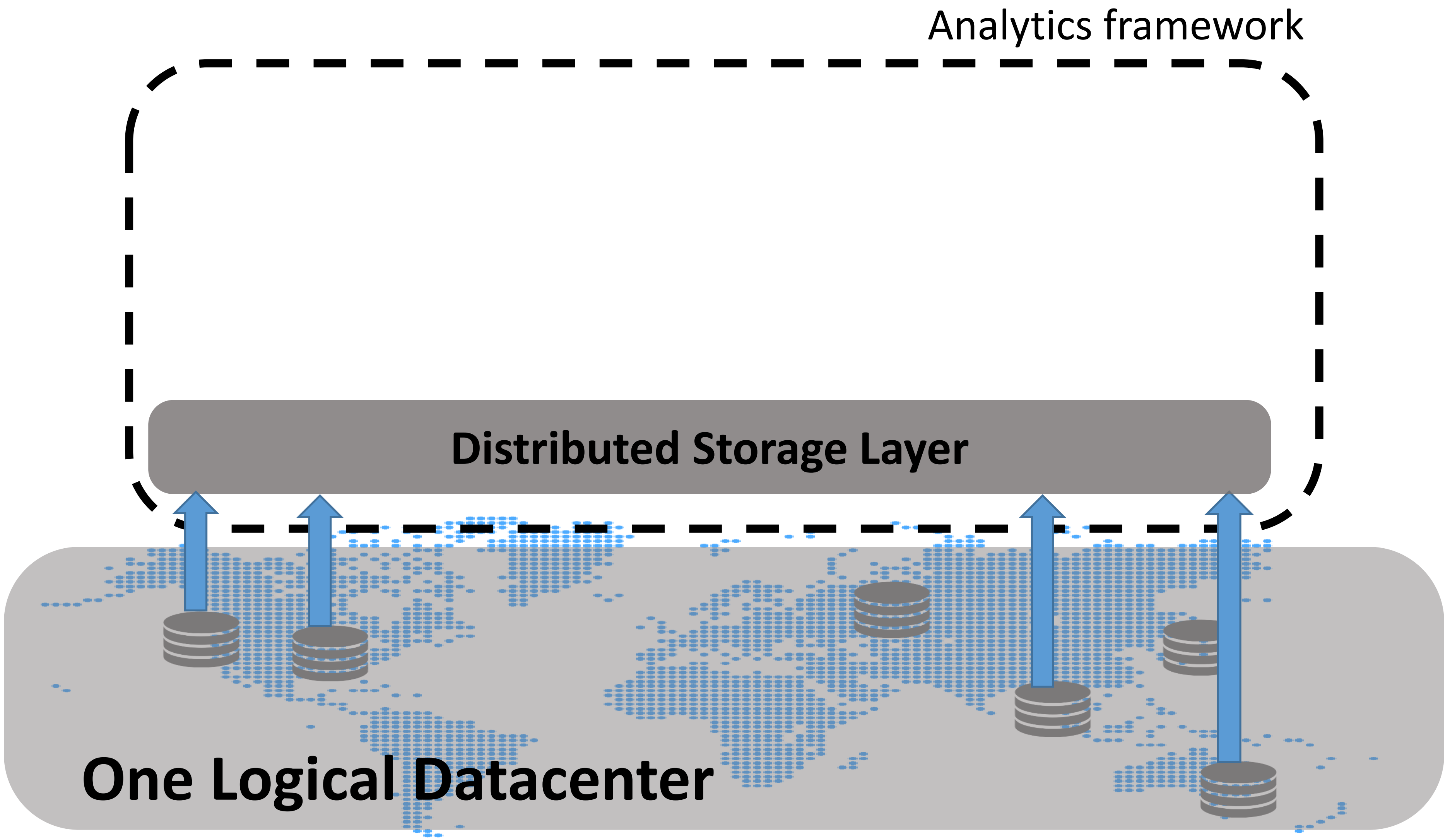
- Available WAN bandwidth is limited → Aggregation latency overhead
- WAN links are expensive → High data transfer cost

Geo-distributed Analytics

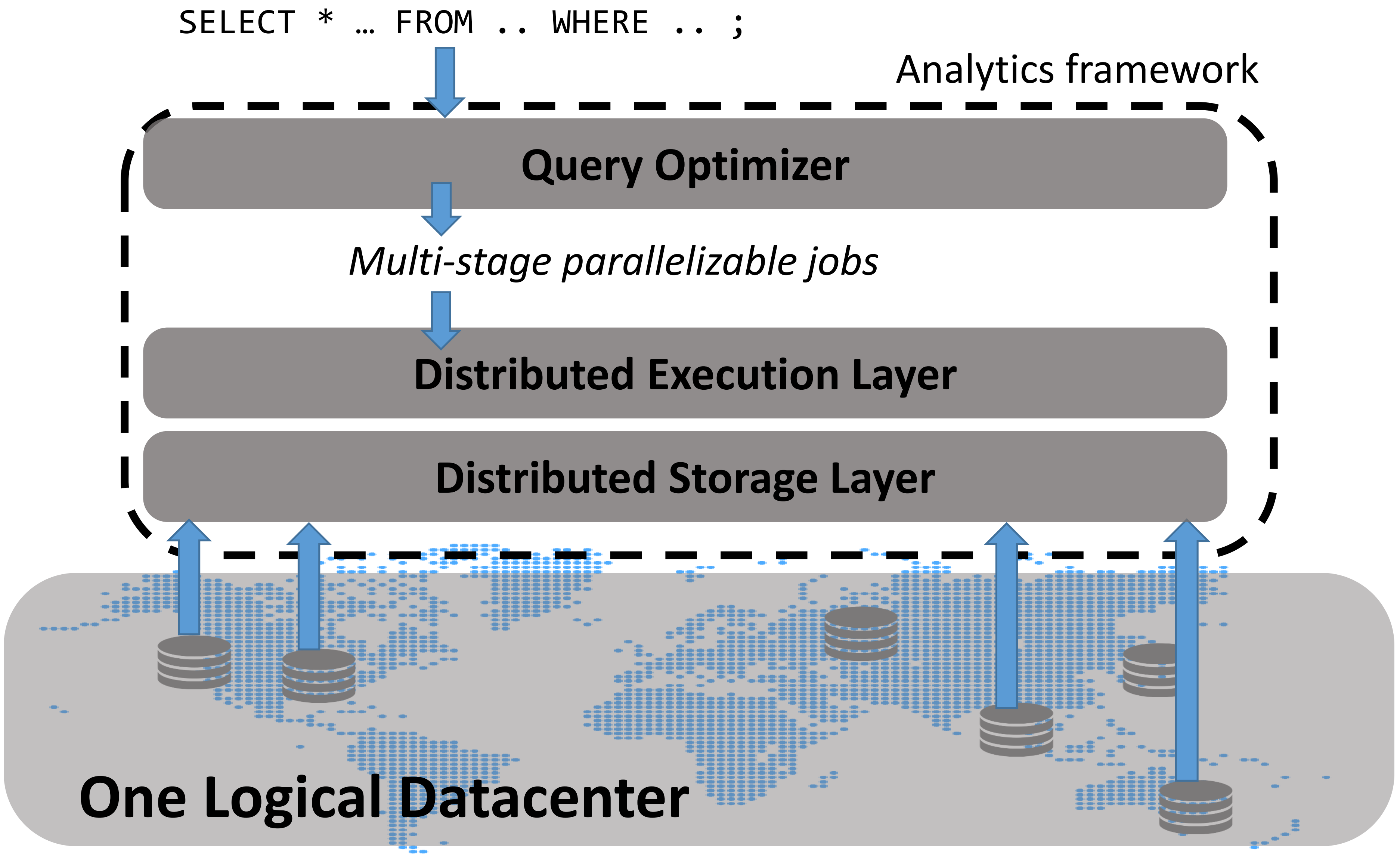
Geo-distributed Analytics



Geo-distributed Analytics

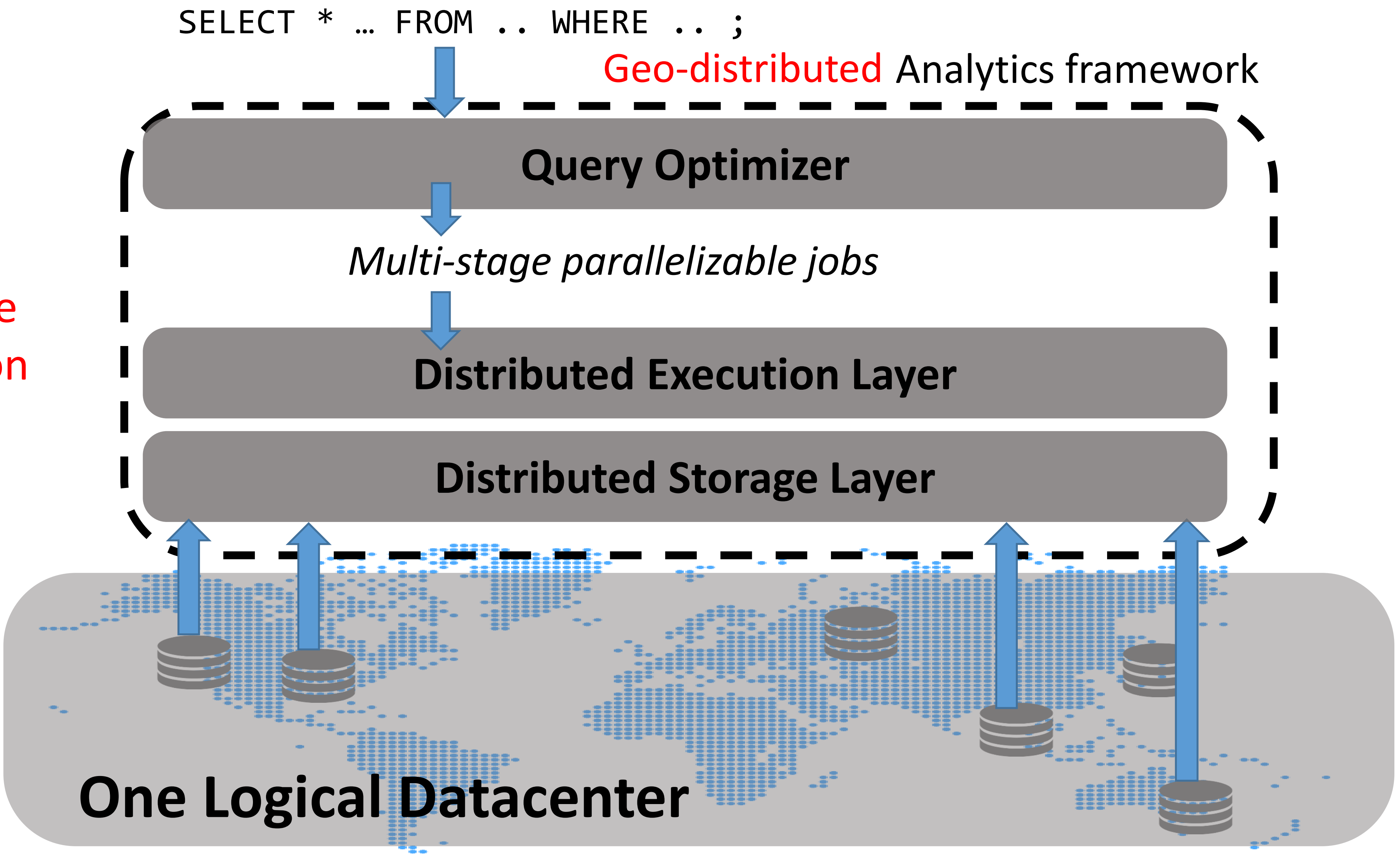


Geo-distributed Analytics


SELECT * … FROM .. WHERE .. ;
Analytics framework
Query Optimizer
Multi-stage parallelizable jobs
Distributed Execution Layer
Distributed Storage Layer
One Logical Datacenter

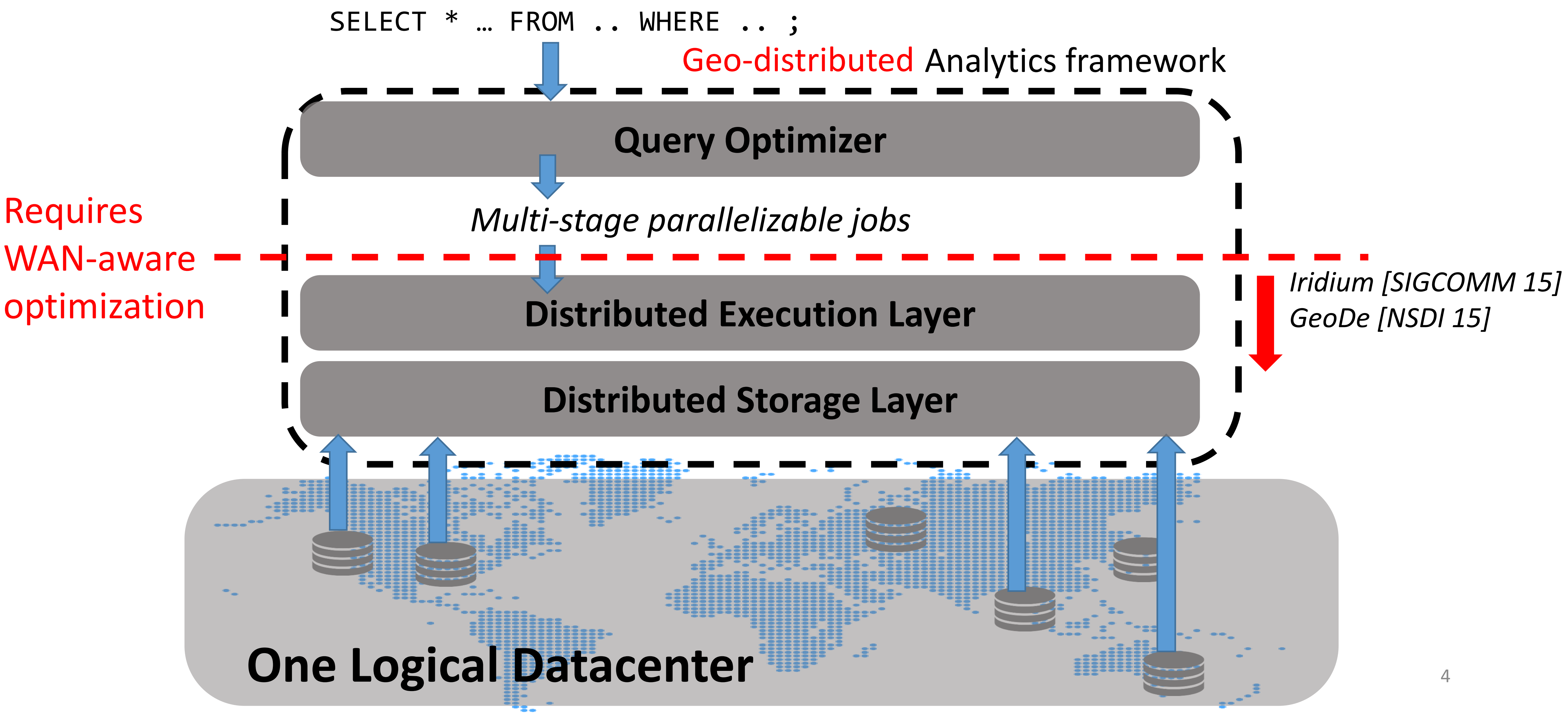
Geo-distributed Analytics

Requires WAN-aware optimization

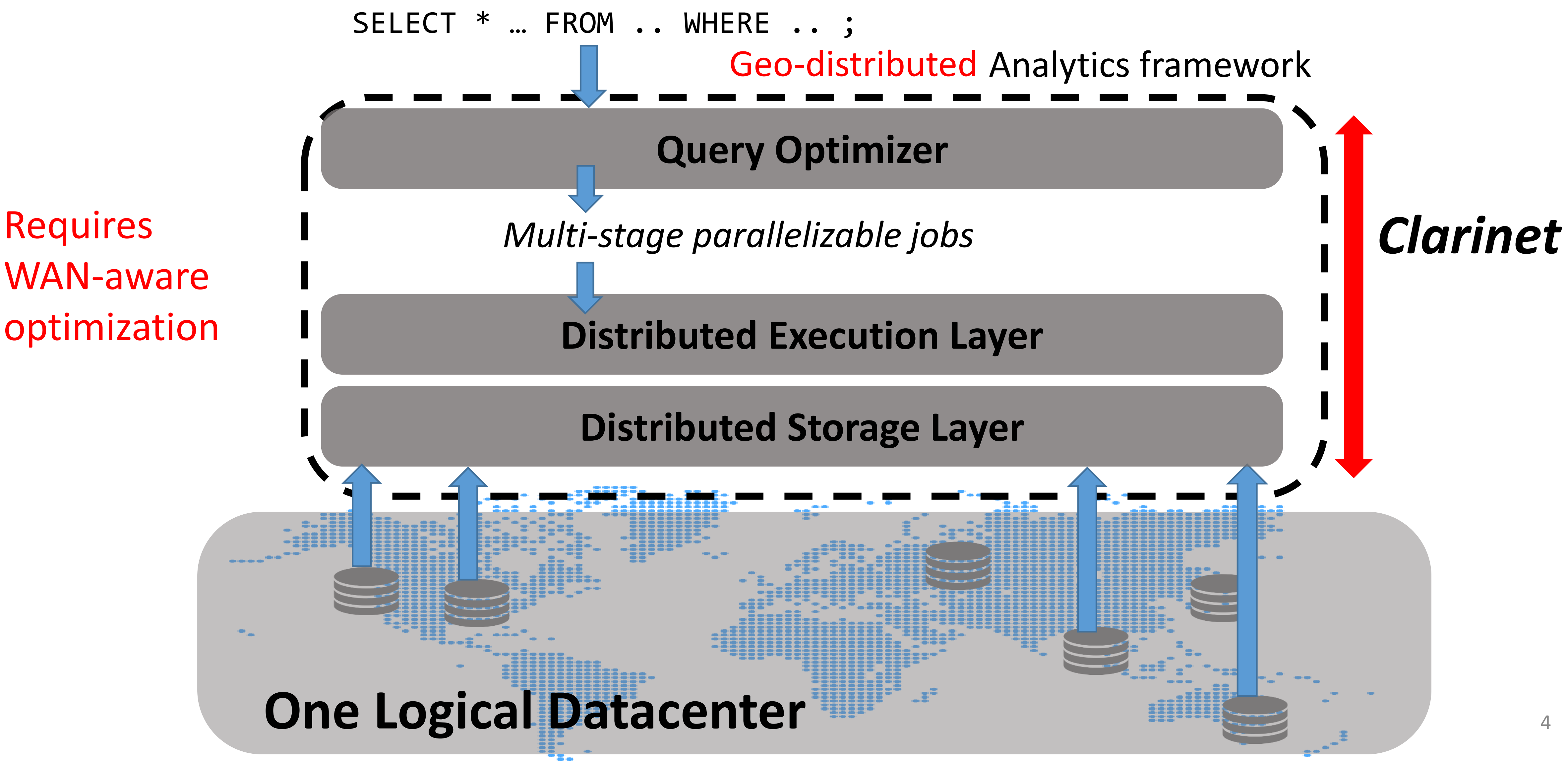


SELECT * … FROM .. WHERE .. ;
Geo-distributed Analytics framework
Query Optimizer
Multi-stage parallelizable jobs
Distributed Execution Layer
Distributed Storage Layer
One Logical Datacenter

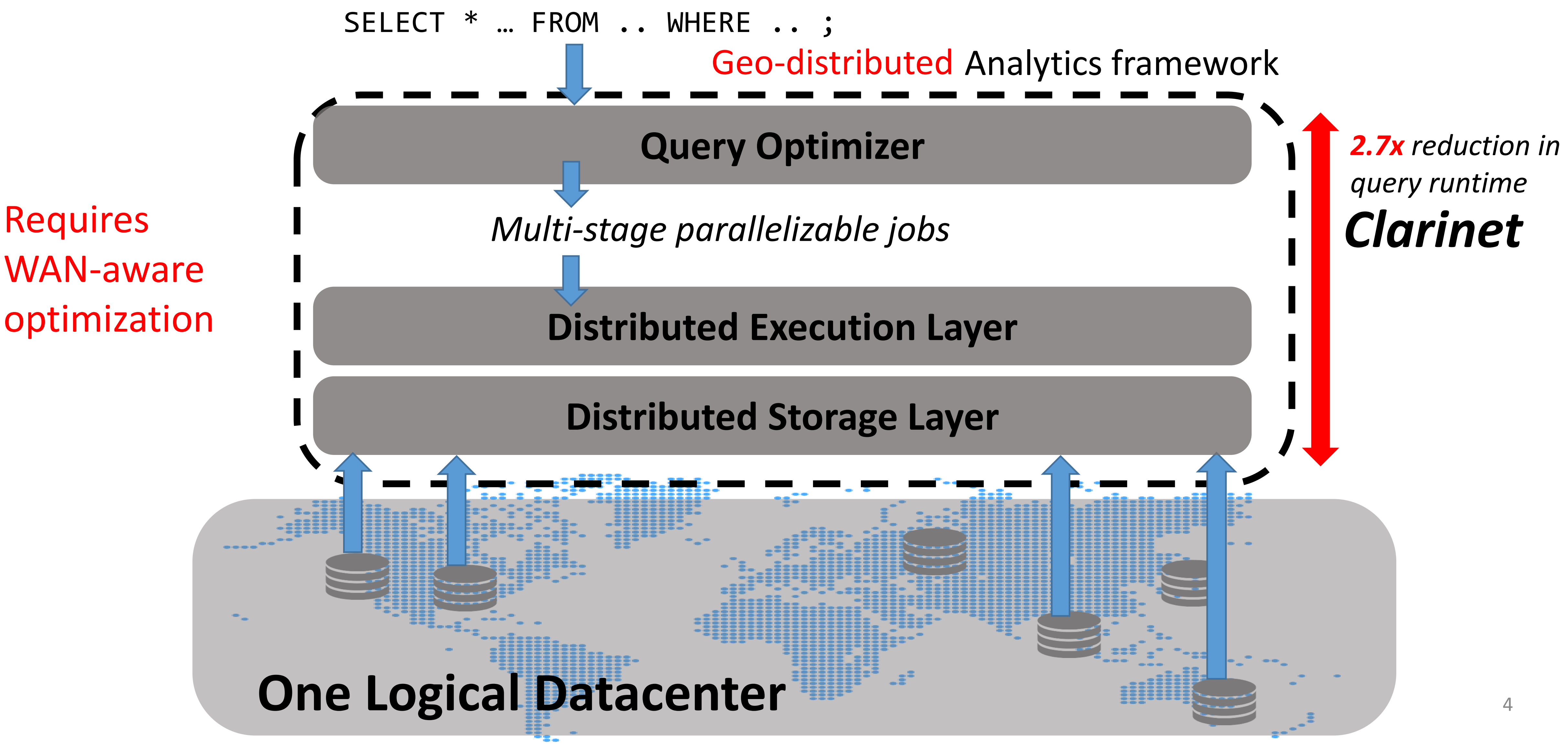
Geo-distributed Analytics


SELECT * … FROM .. WHERE .. ;
Geo-distributed Analytics framework
Query Optimizer
Multi-stage parallelizable jobs
Requires WAN-aware optimization
Distributed Execution Layer
Iridium [SIGCOMM 15]
GeoDe [NSDI 15]
Distributed Storage Layer
One Logical Datacenter

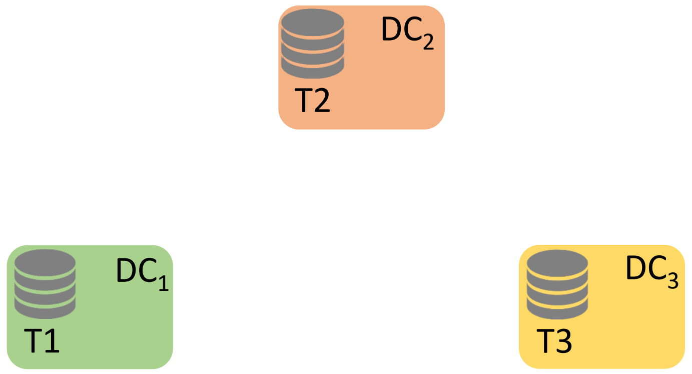
Geo-distributed Analytics


SELECT * … FROM .. WHERE .. ;
Geo-distributed Analytics framework
Query Optimizer
Multi-stage parallelizable jobs
Distributed Execution Layer
Distributed Storage Layer
Clarinet
Requires WAN-aware optimization
One Logical Datacenter

Geo-distributed Analytics

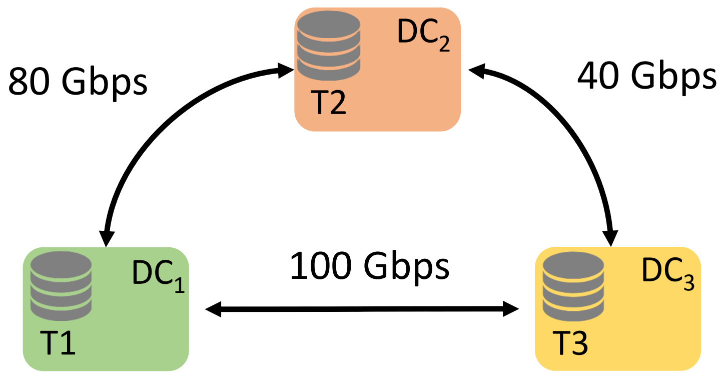

SELECT * … FROM .. WHERE .. ;
Geo-distributed Analytics framework
Query Optimizer
Multi-stage parallelizable jobs
Distributed Execution Layer
Distributed Storage Layer
Requires WAN-aware optimization
2.7x reduction in query runtime
Clarinet
One Logical Datacenter

WAN Aware Query Optimization



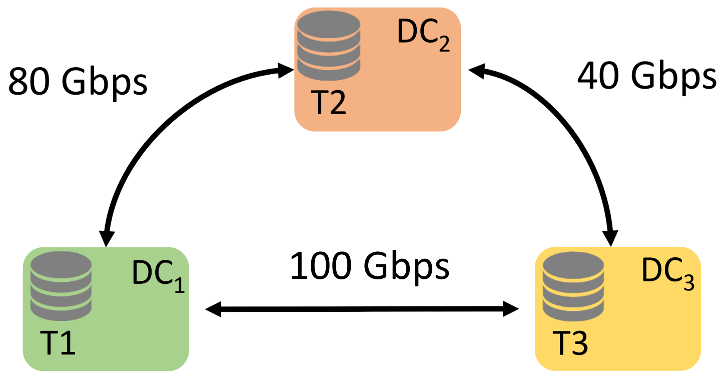
T1, T2, T3: Tables storing click logs

WAN Aware Query Optimization



T1, T2, T3: Tables storing click logs

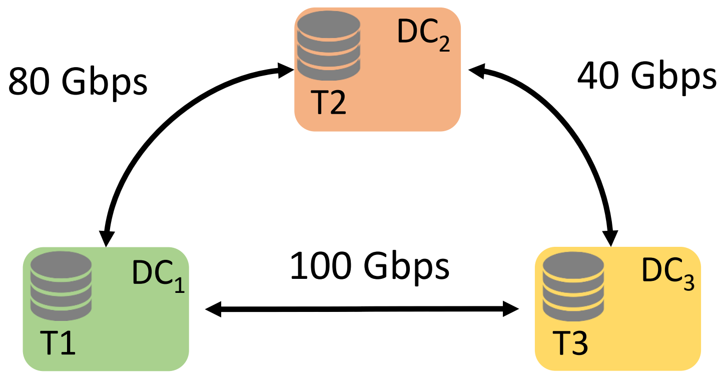
WAN Aware Query Optimization



T1, T2, T3: Tables storing click logs

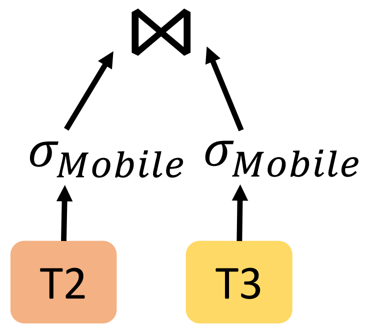
```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

WAN Aware Query Optimization

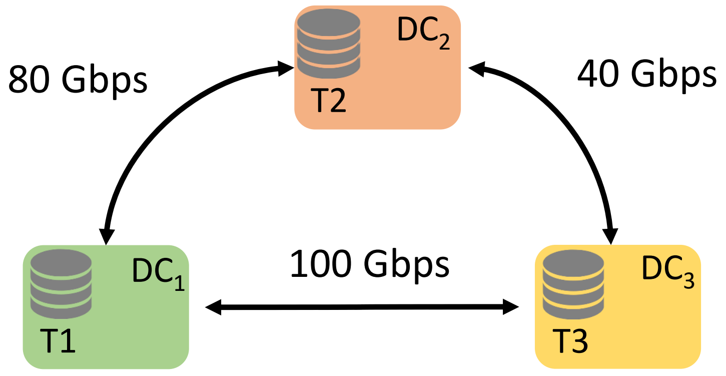


T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

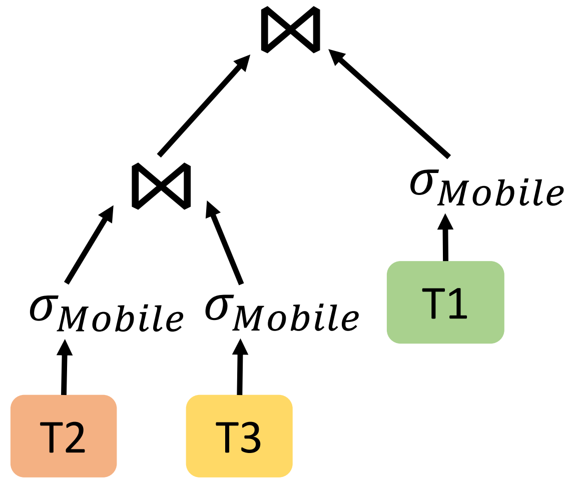


WAN Aware Query Optimization

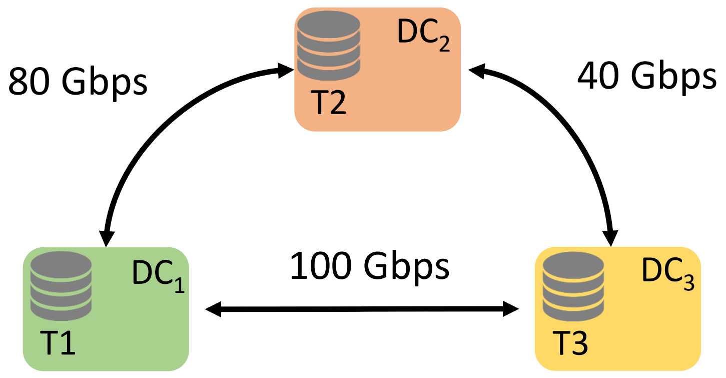


T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

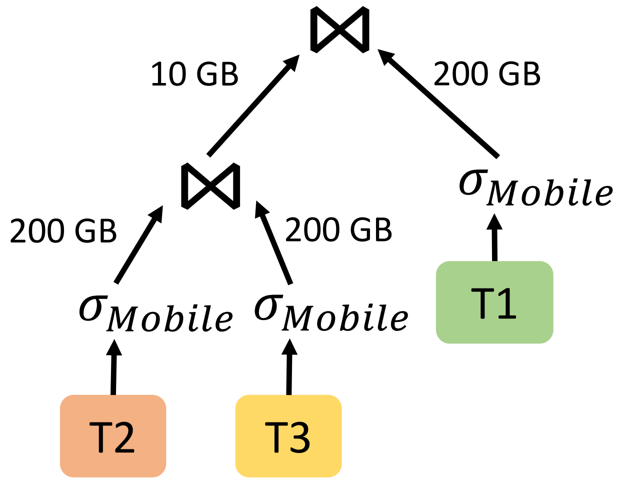


WAN Aware Query Optimization

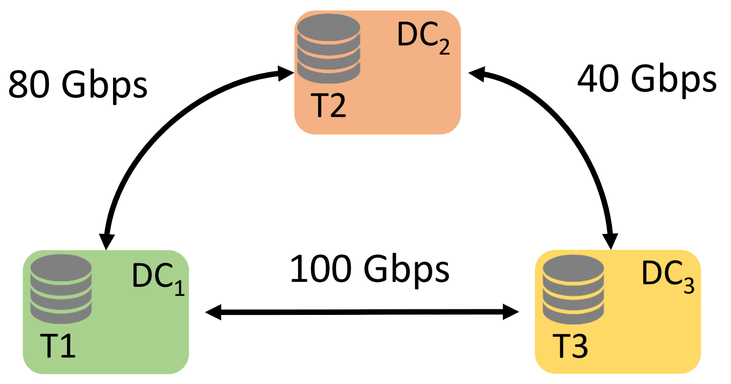


T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == “mobile”;
```



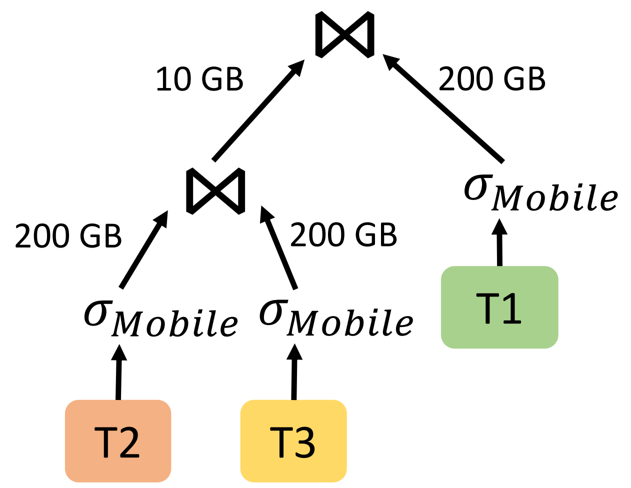
WAN Aware Query Optimization



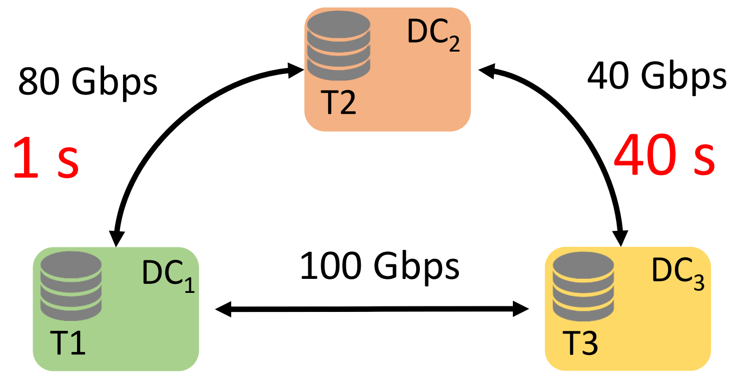
T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

Plan running time: 41 s

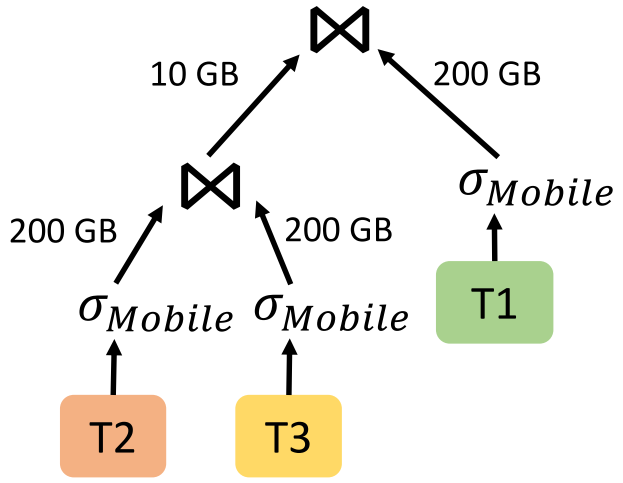


WAN Aware Query Optimization



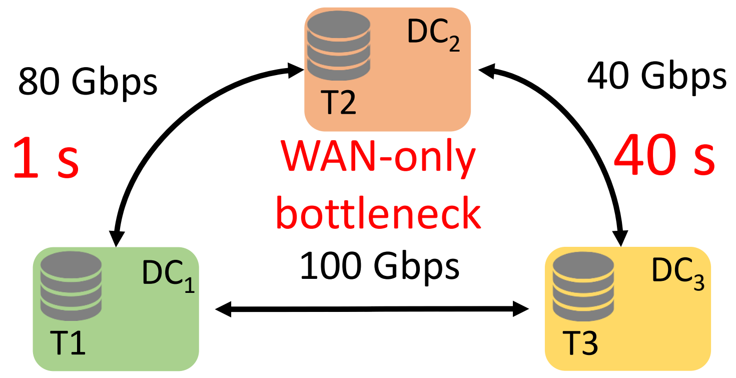
T1, T2, T3: Tables storing click logs

Plan running time: 41 s



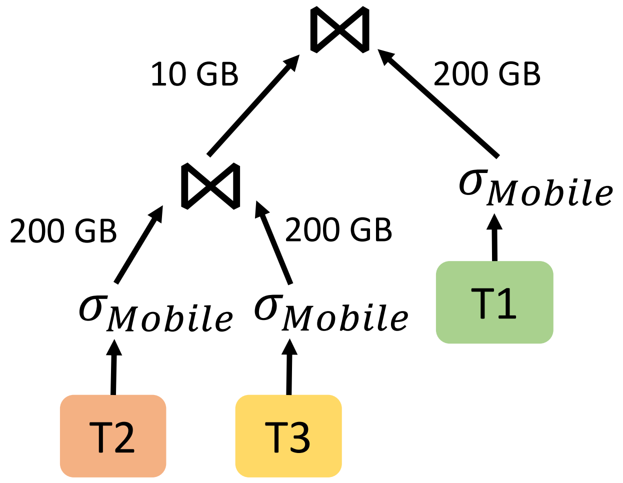
```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

WAN Aware Query Optimization



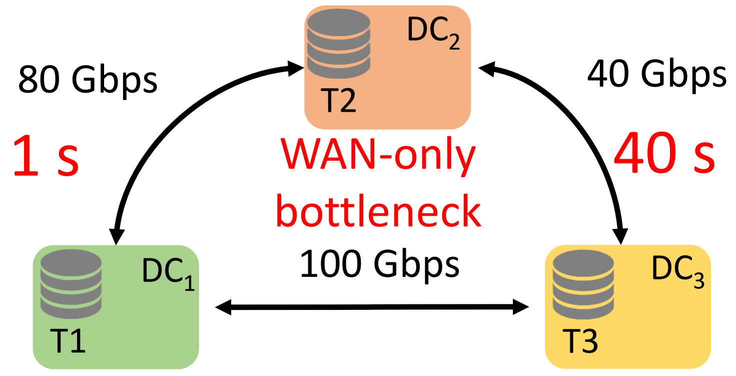
T1, T2, T3: Tables storing click logs

Plan running time: 41 s



```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

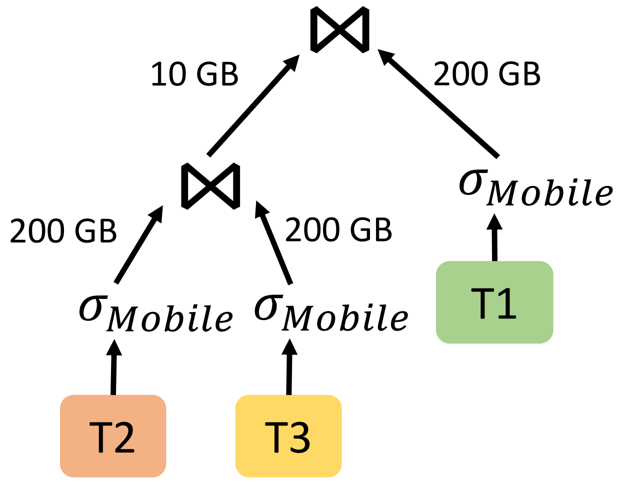
WAN Aware Query Optimization



T1, T2, T3: Tables storing click logs

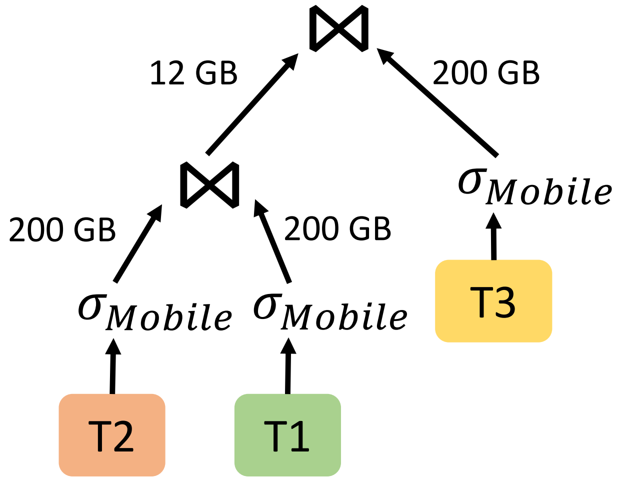
```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM   T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

Plan running time: 41 s



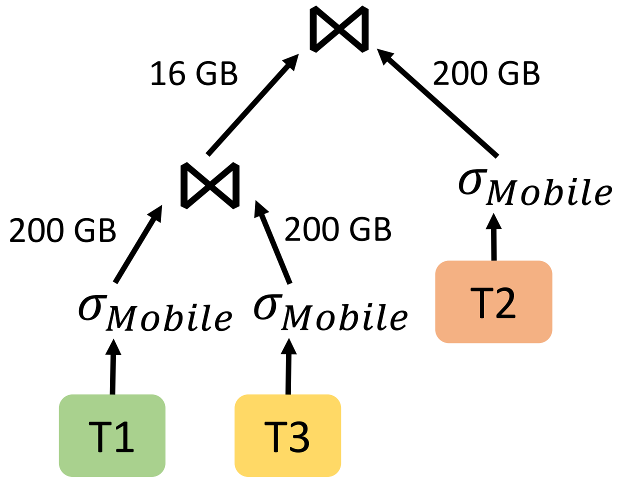
Plan A

Plan running time: 20.96 s



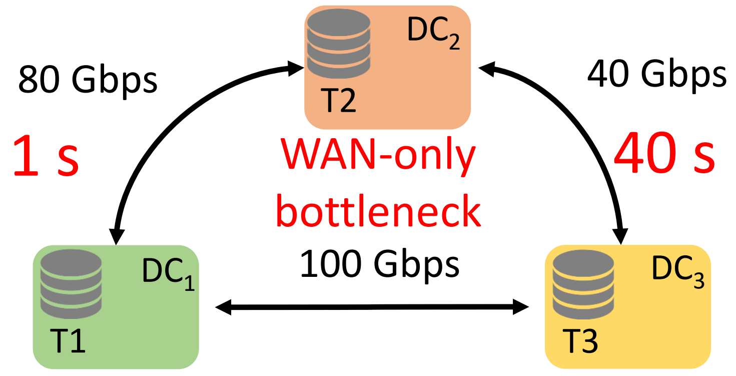
Plan B

Plan running time: 17.6 s



Plan C

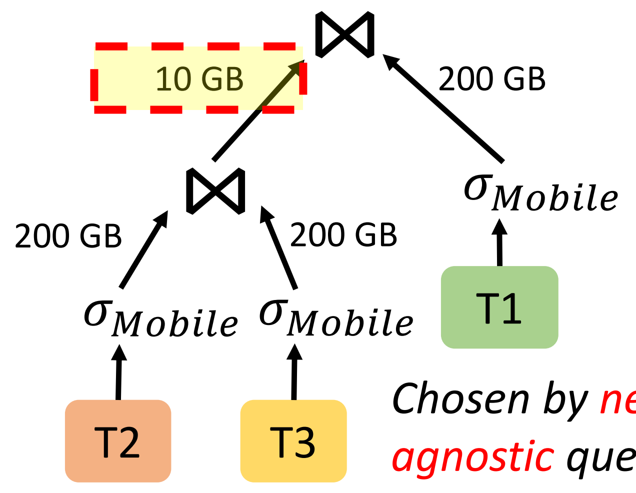
WAN Aware Query Optimization



T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

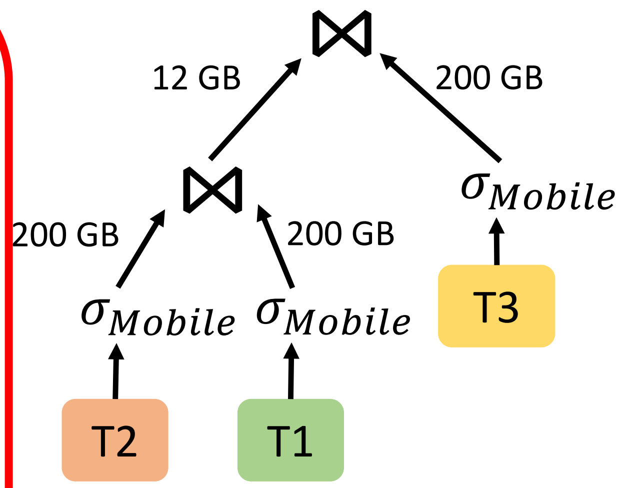
Plan running time: 41 s



Chosen by network agnostic query optimizer

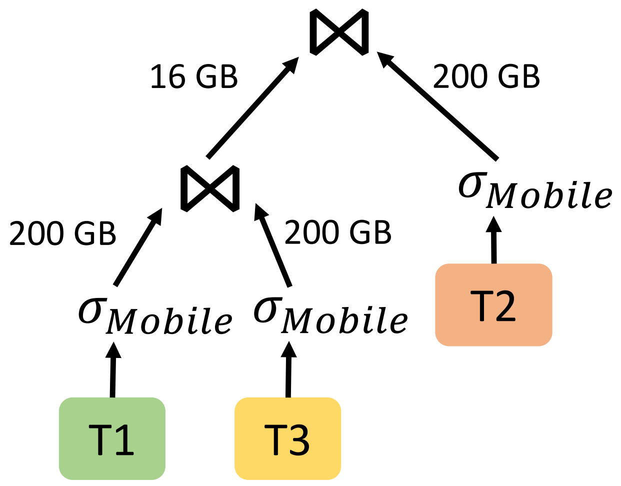
Plan A

Plan running time: 20.96 s



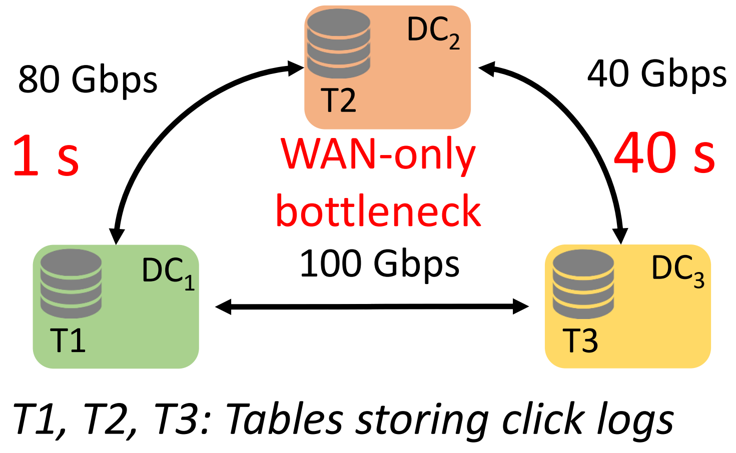
Plan B

Plan running time: 17.6 s



Plan C

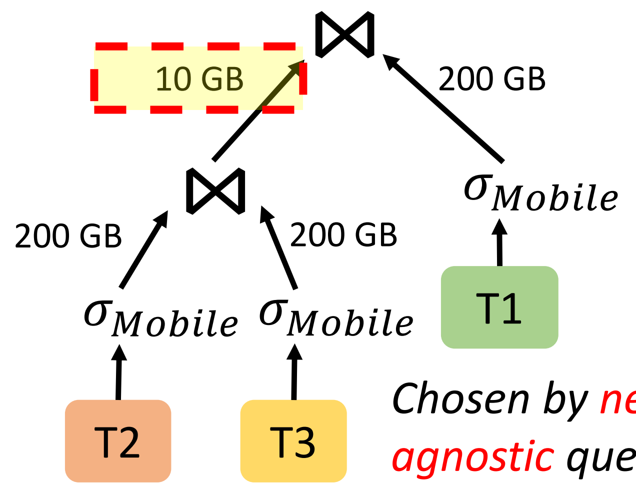
WAN Aware Query Optimization



T1, T2, T3: Tables storing click logs

```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM  T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == "mobile";
```

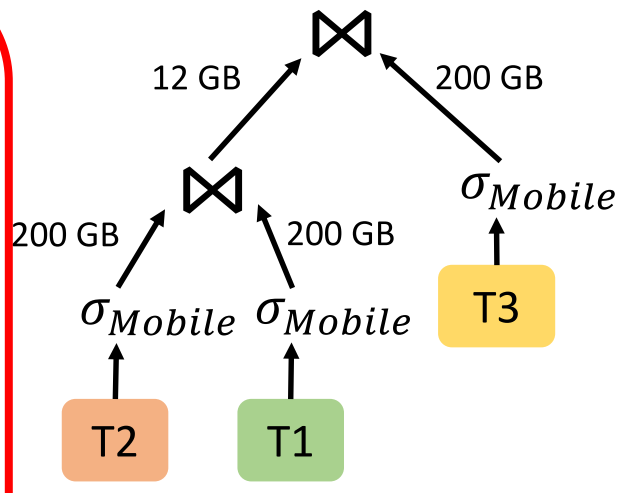
Plan running time: 41 s



Chosen by network agnostic query optimizer

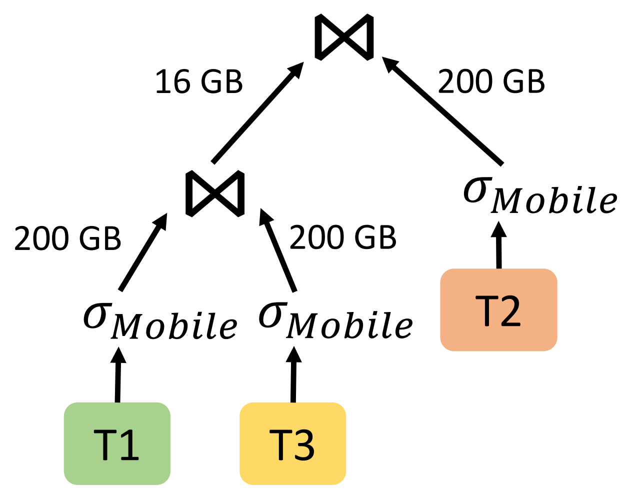
Plan A

Plan running time: 20.96 s



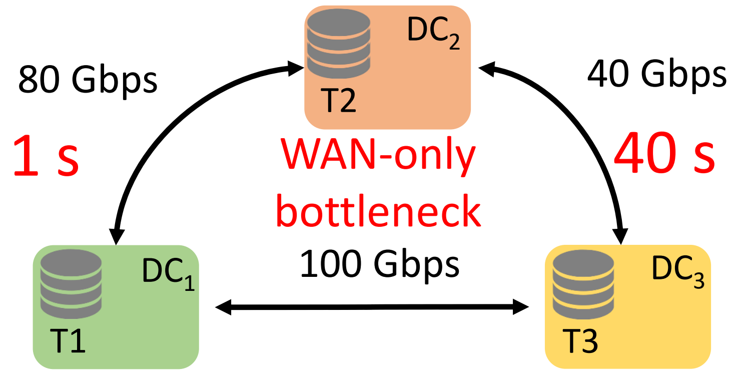
Plan B

Plan running time: 17.6 s



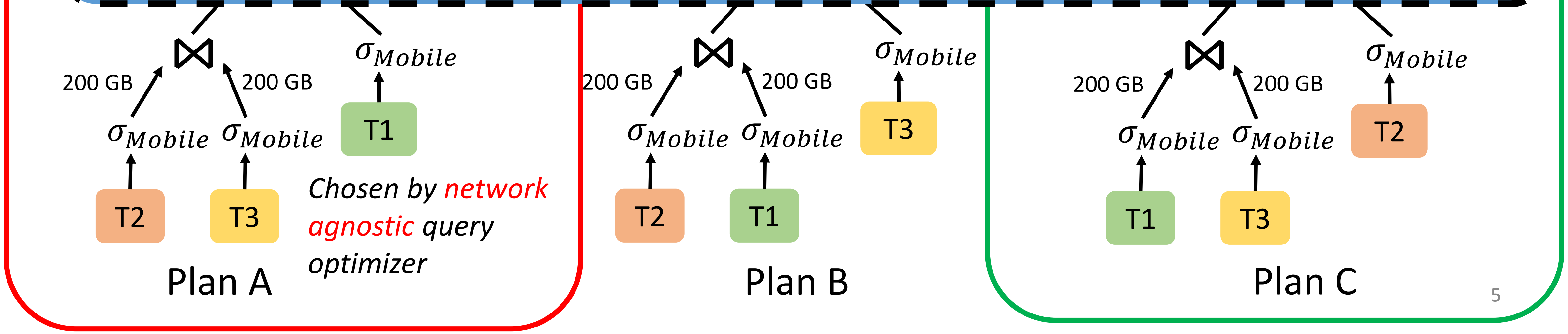
Plan C

WAN Aware Query Optimization



```
QUERY
SELECT T1.user, T1.latency, T2.latency, T3.latency
FROM   T1, T2, T3
WHERE T1.user == T2.user AND T1.user == T3.user
AND T1.device == T2.device == T3.device == “mobile”;
```

WAN-aware query optimizer that uses network transfer duration to choose query plans

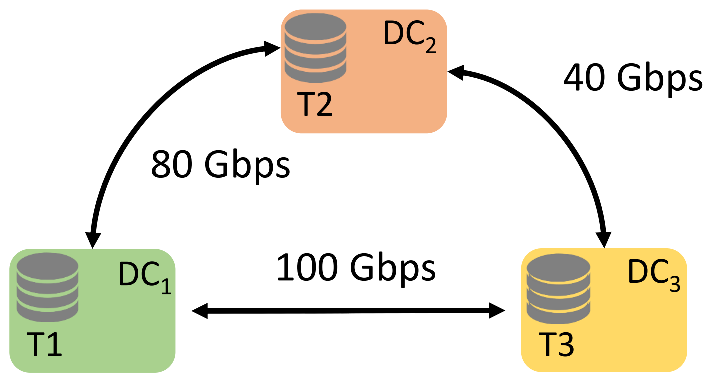


Chosen by network agnostic query optimizer

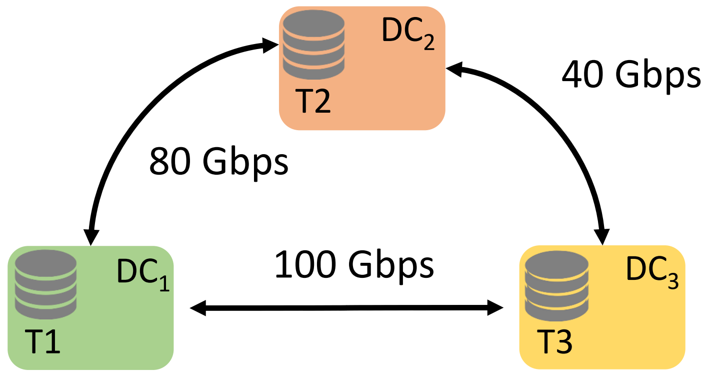
Outline

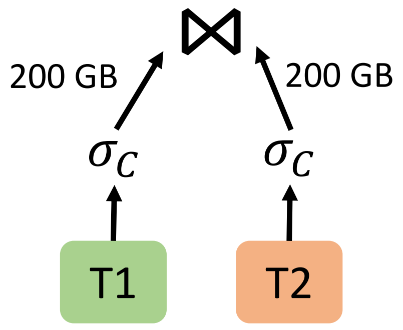
1. Motivation
2. Challenges in choosing query plan based on WAN transfer durations
3. Solution
    - Single query
    - Multiple simultaneous queries
4. Experimental Evaluation

Other factors also affect query plan run time

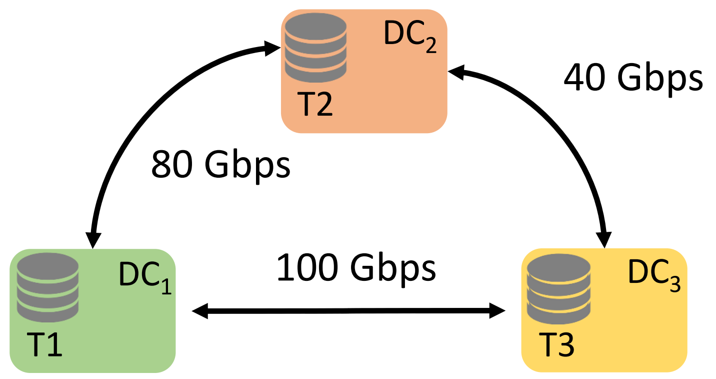


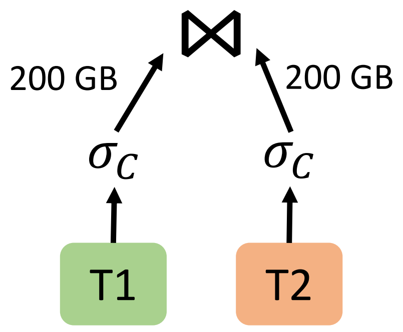
Other factors also affect query plan run time

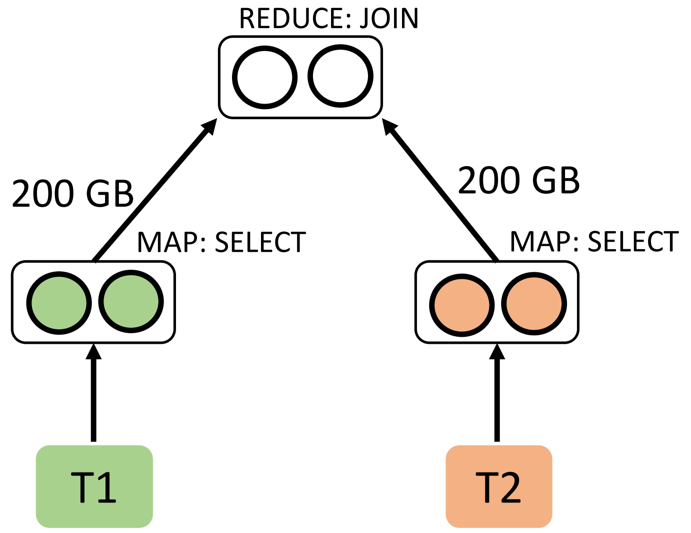




Other factors also affect query plan run time

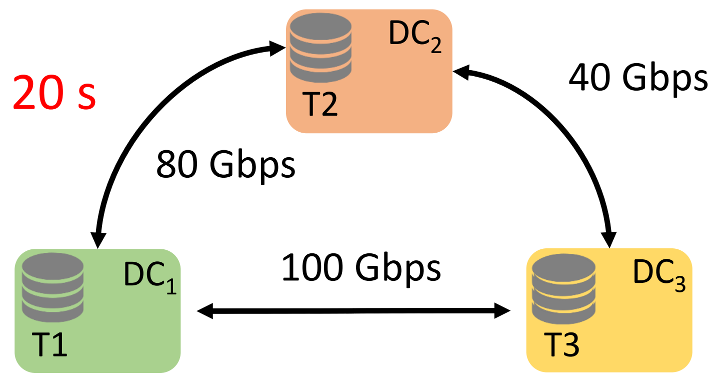


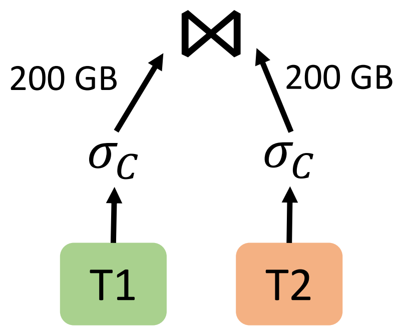


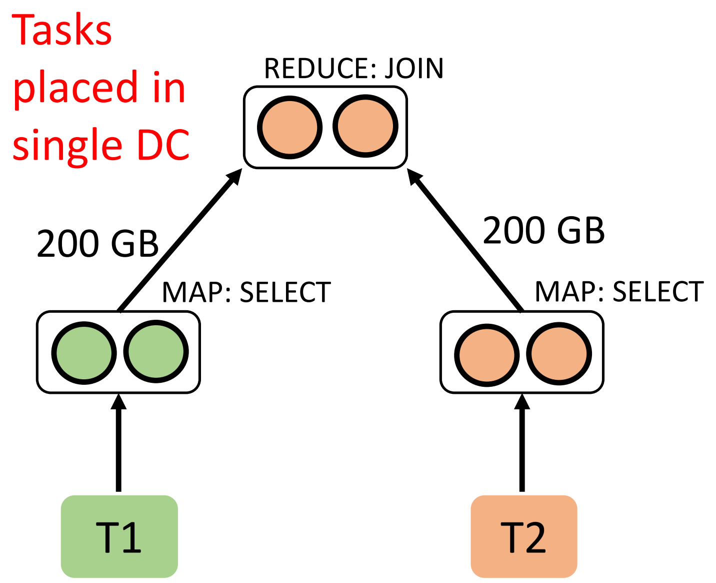


Map Reduce Job

Other factors also affect query plan run time

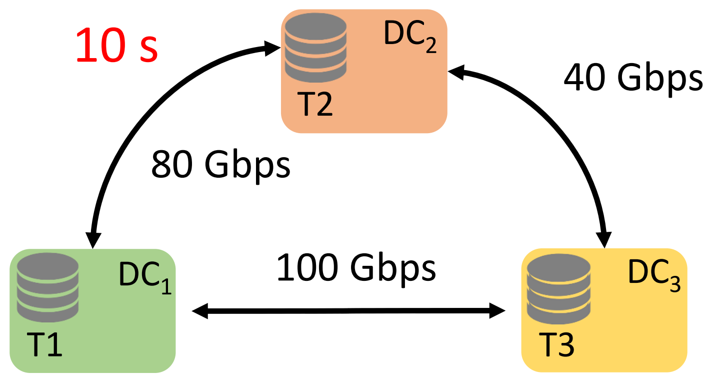


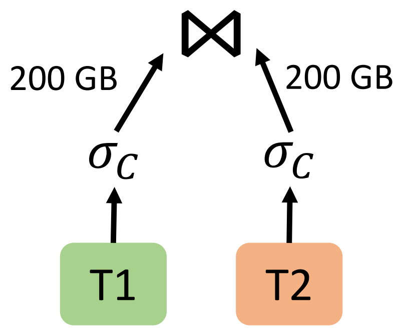




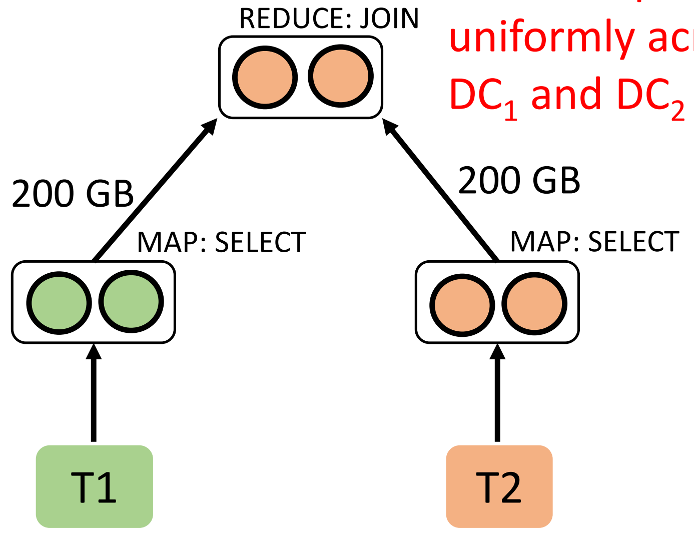
Map Reduce Job

Other factors also affect query plan run time



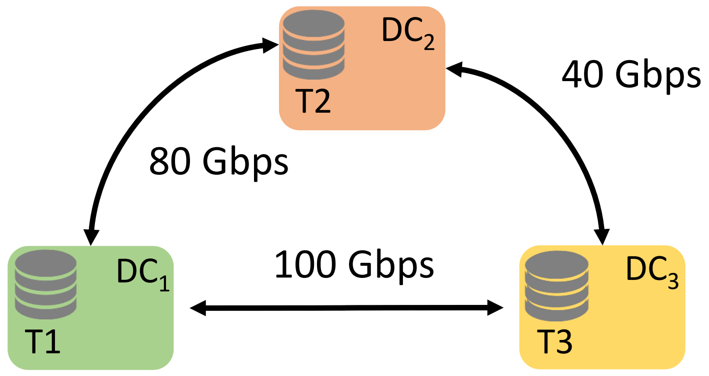


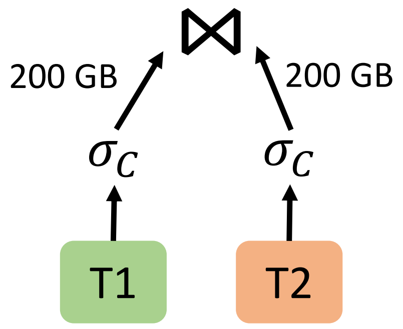
Tasks are placed uniformly across DC$_1$ and DC$_2$

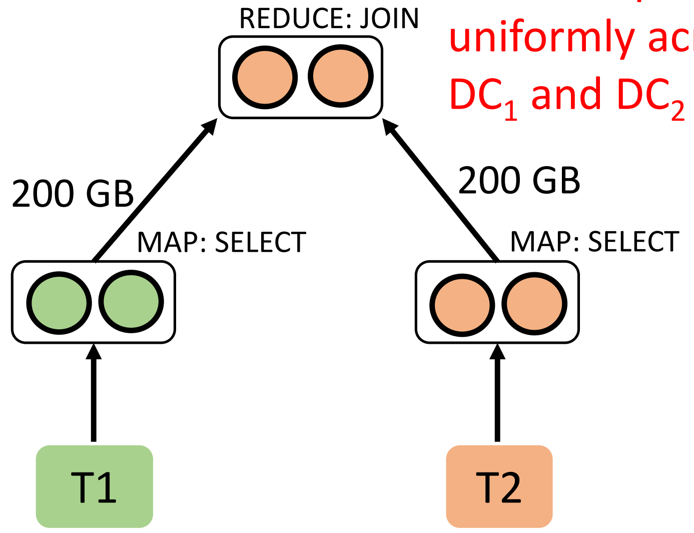


Map Reduce Job

Other factors also affect query plan run time







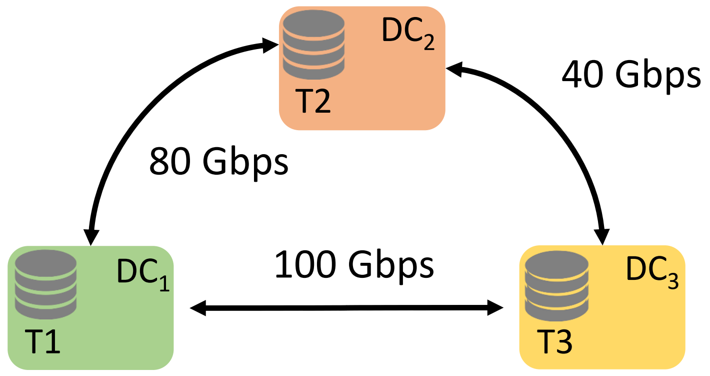
Tasks are placed uniformly across DC$_1$ and DC$_2$

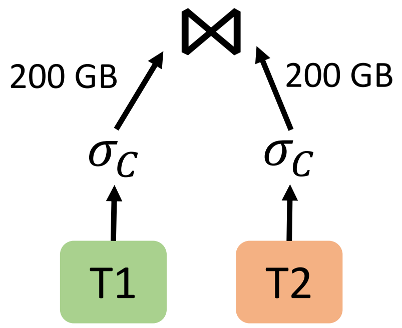
Map Reduce Job

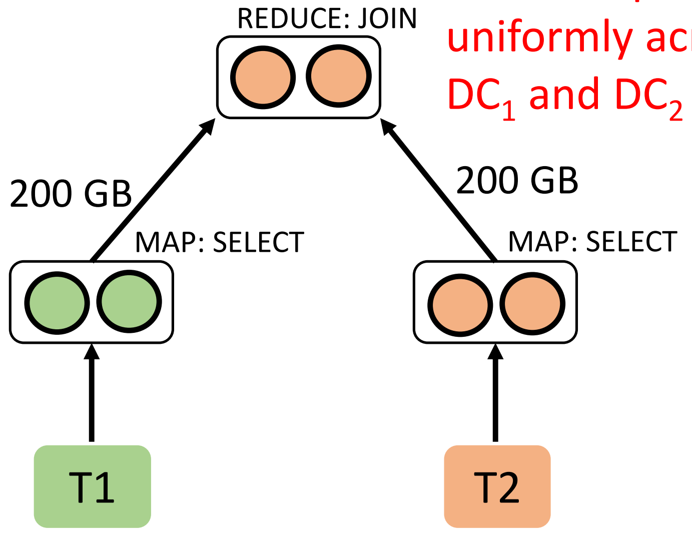
While evaluating different query plans

1. Plan A: 41 s
2. Plan B: 20.96
3. $\boxed{\text{Plan C:}}$ 17.6 s

Other factors also affect query plan run time







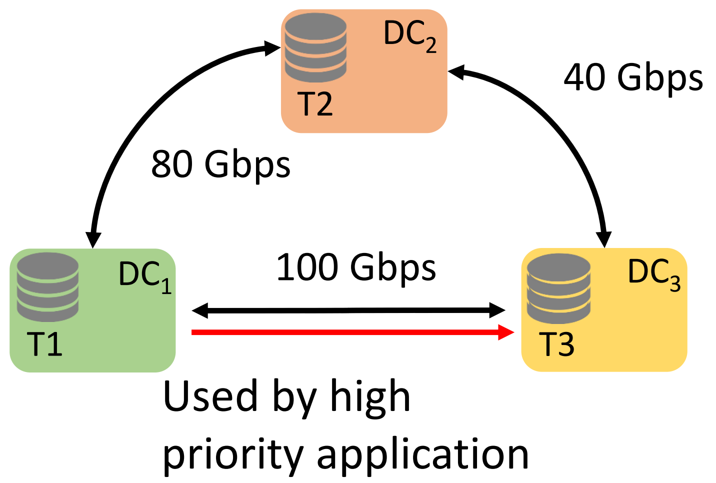
Tasks are placed uniformly across DC$_1$ and DC$_2$

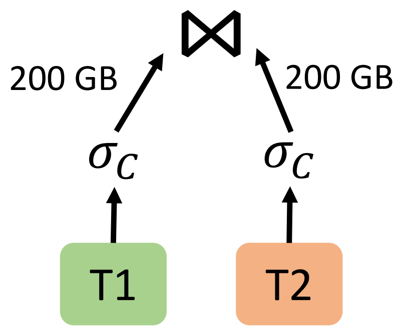
Map Reduce Job

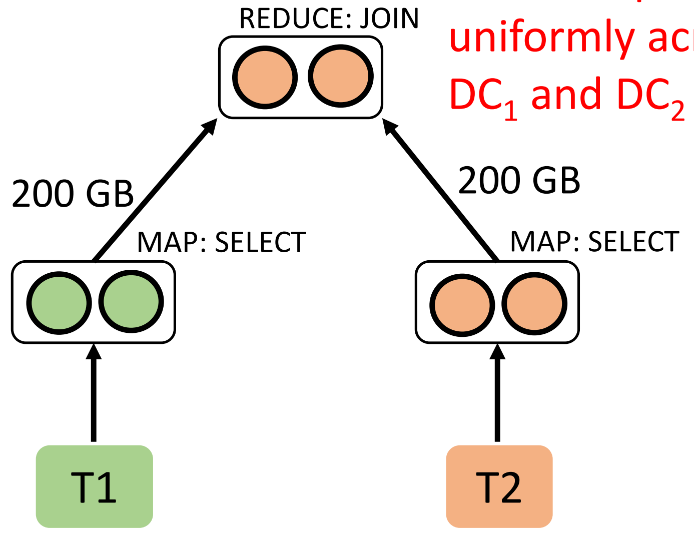
While evaluating different query plans

1. Plan A: ~~41 s~~ 20.5 s
2. Plan B: ~~20.96~~ 11.2 s
3. Plan C: 17.6 s

Other factors also affect query plan run time







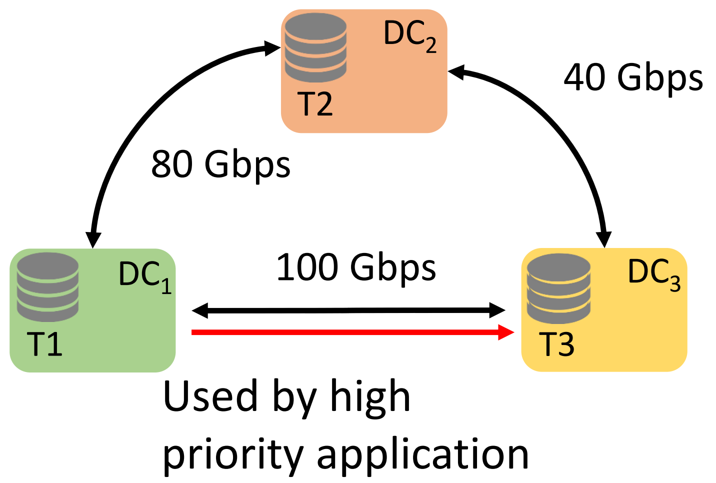
Tasks are placed uniformly across DC$_1$ and DC$_2$

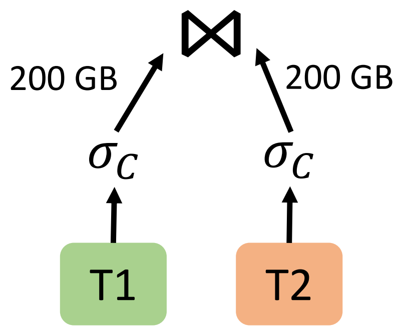
Map Reduce Job

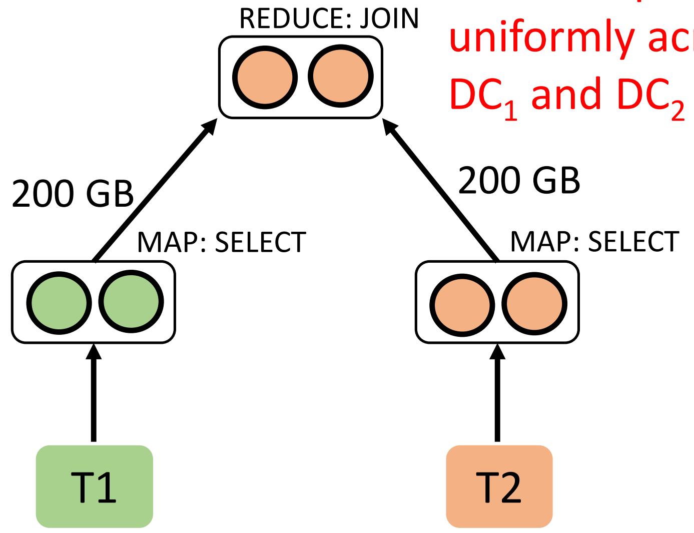
While evaluating different query plans

1. Plan A: ~~41 s~~ 20.5 s
2. Plan B: ~~20.96~~ 11.2 s
3. $\boxed{\text{Plan C:}}$ 17.6 s

Other factors also affect query plan run time







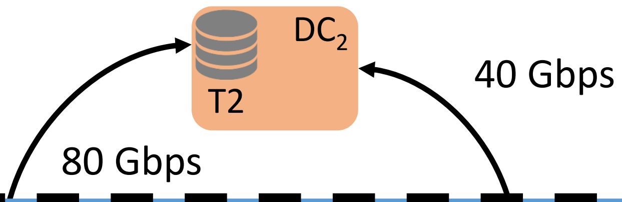
Map Reduce Job

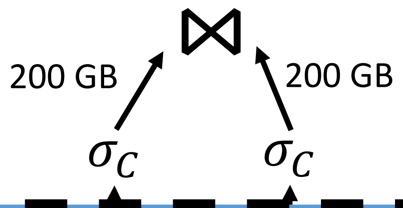
Tasks are placed uniformly across DC$_1$ and DC$_2$

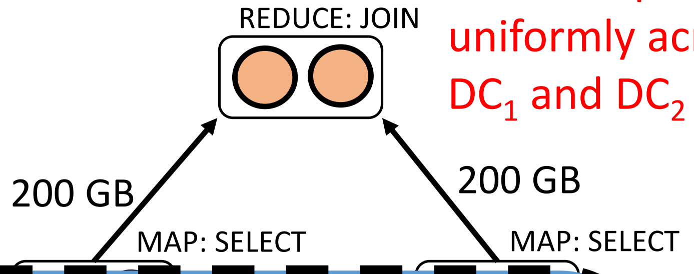
While evaluating different query plans

1. Plan A: ~~41 s~~ 20.5 s
2. Plan B: ~~20.96~~ 11.2 s
3. Plan C: 17.6 s

Other factors also affect query plan run time







Tasks are placed uniformly across DC$_1$ and DC$_2$

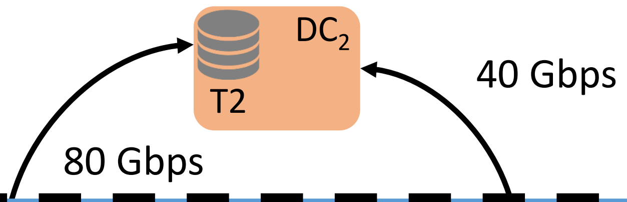
Choose query plan based on:

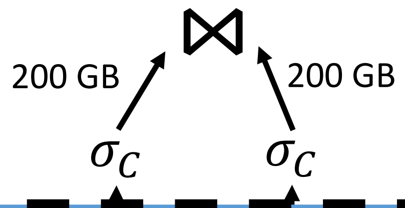
1. Best available task placements

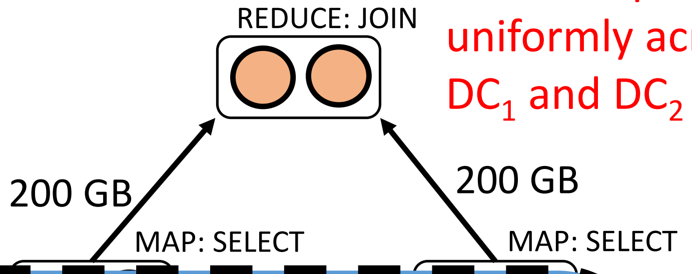
W

1.	Plan A:	~~41 s~~	20.5 s
2.	Plan B:	~~20.96~~	11.2 s
3.	Plan C:	17.6 s	

Other factors also affect query plan run time







Tasks are placed uniformly across DC$_1$ and DC$_2$

Choose query plan based on:

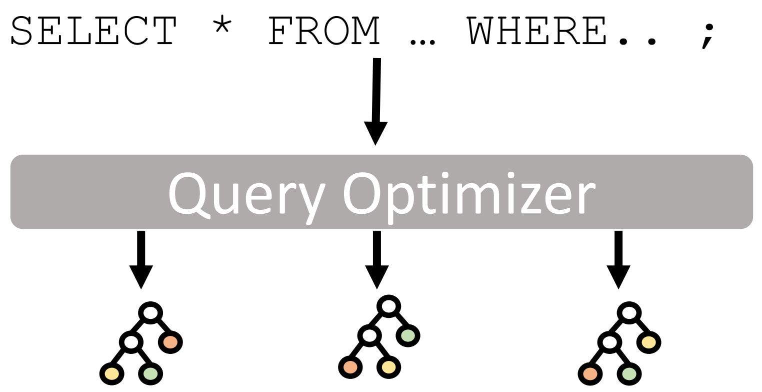
1. Best available task placements
2. Schedule of network transfers

W

1.	Plan A:	~~41 s~~	20.5 s
2.	Plan B:	~~20.96~~	11.2 s
3.	Plan C:	17.6 s	

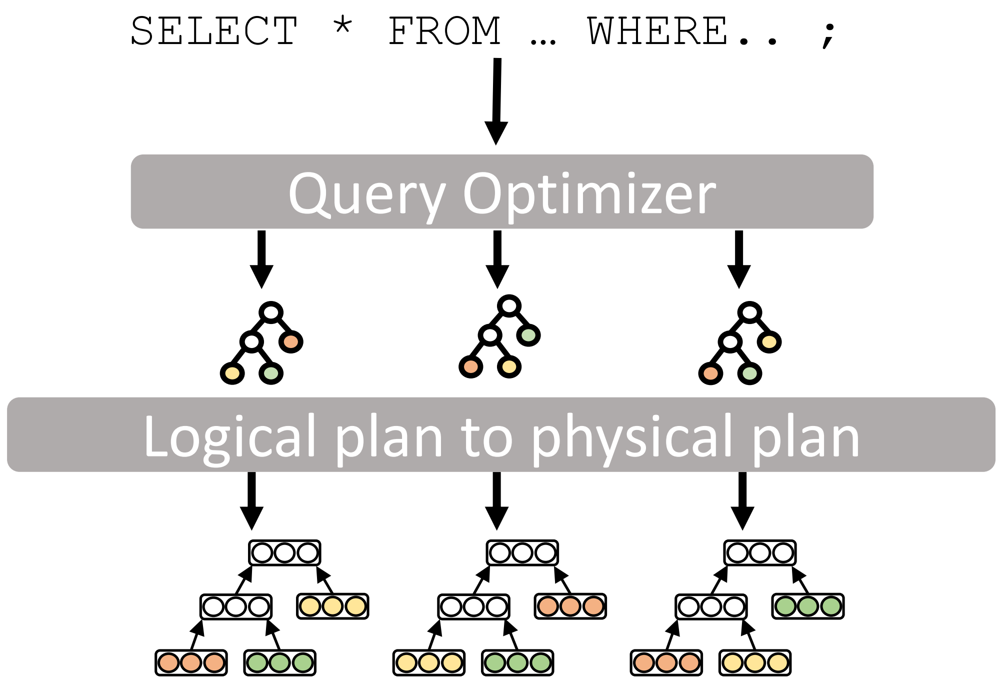
Joint plan selection, placement and scheduling

Joint plan selection, placement and scheduling



Multiple query plans (join orders) per query

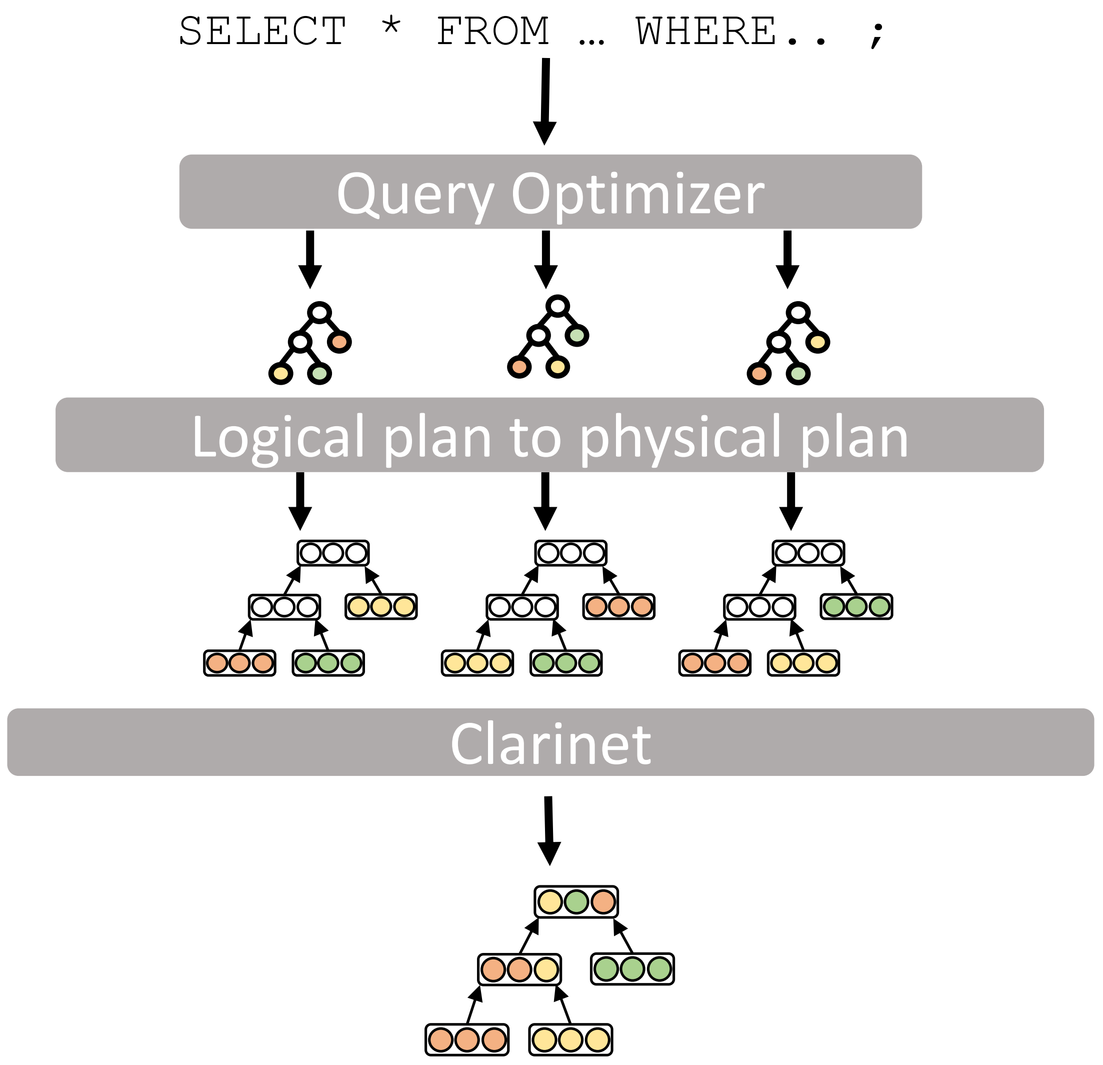
Joint plan selection, placement and scheduling



Multiple query plans (join orders) per query

Assign parallelism for each stage

Joint plan selection, placement and scheduling



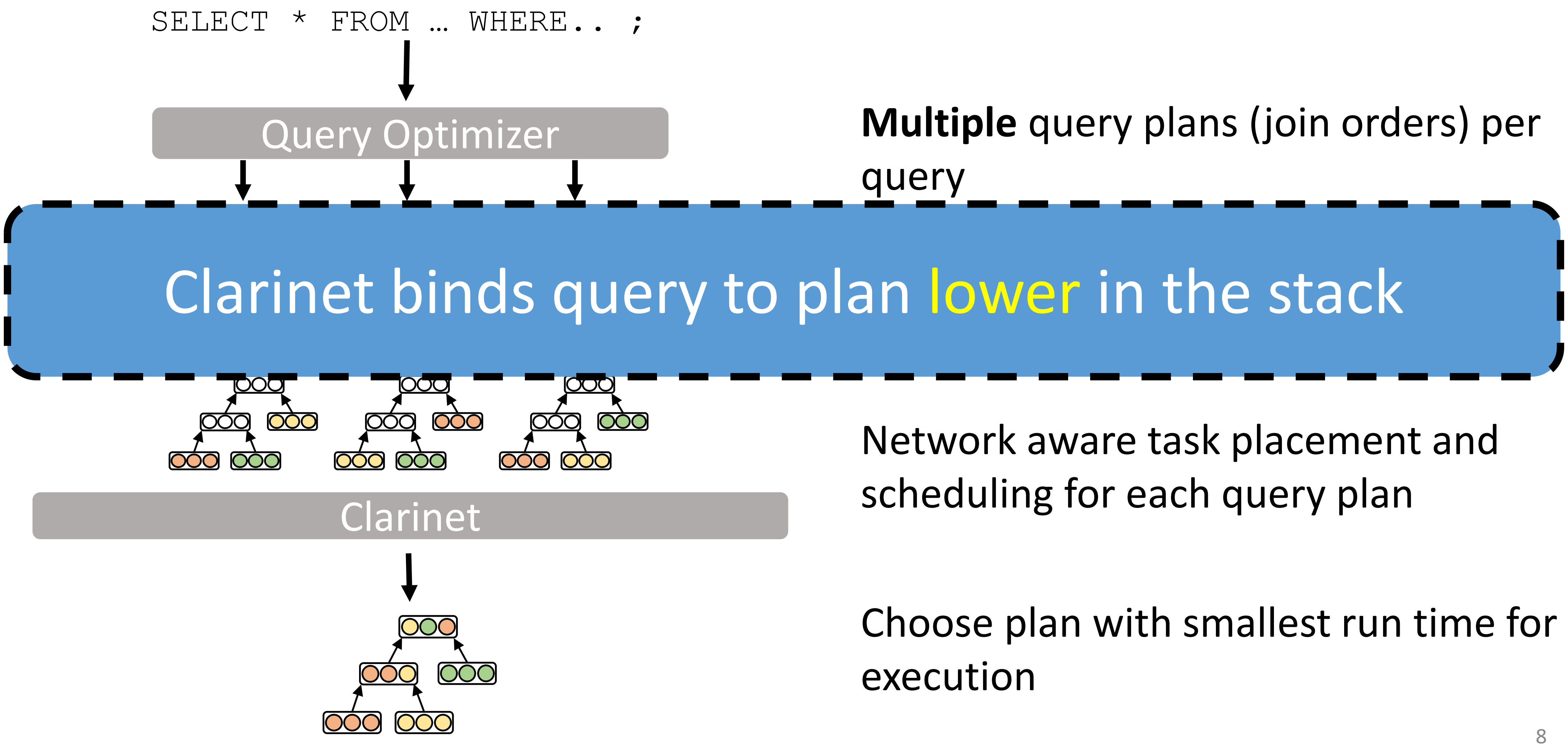
Multiple query plans (join orders) per query

Assign parallelism for each stage

Network aware task placement and scheduling for each query plan

Choose plan with smallest run time for execution

Joint plan selection, placement and scheduling

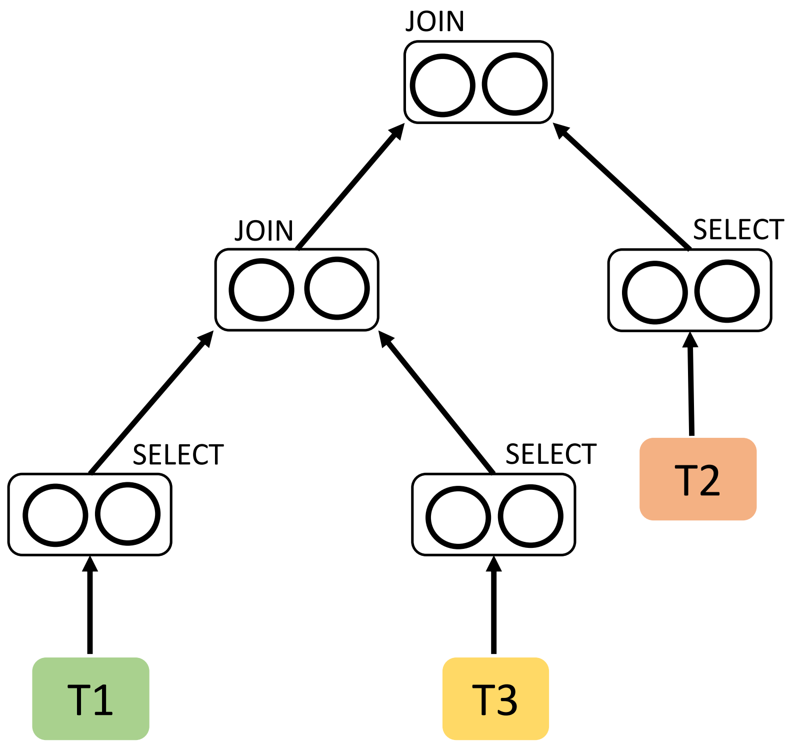

SELECT * FROM … WHERE.. ;
Query Optimizer
Clarinet binds query to plan lower in the stack
Clarinet

Multiple query plans (join orders) per query

Network aware task placement and scheduling for each query plan

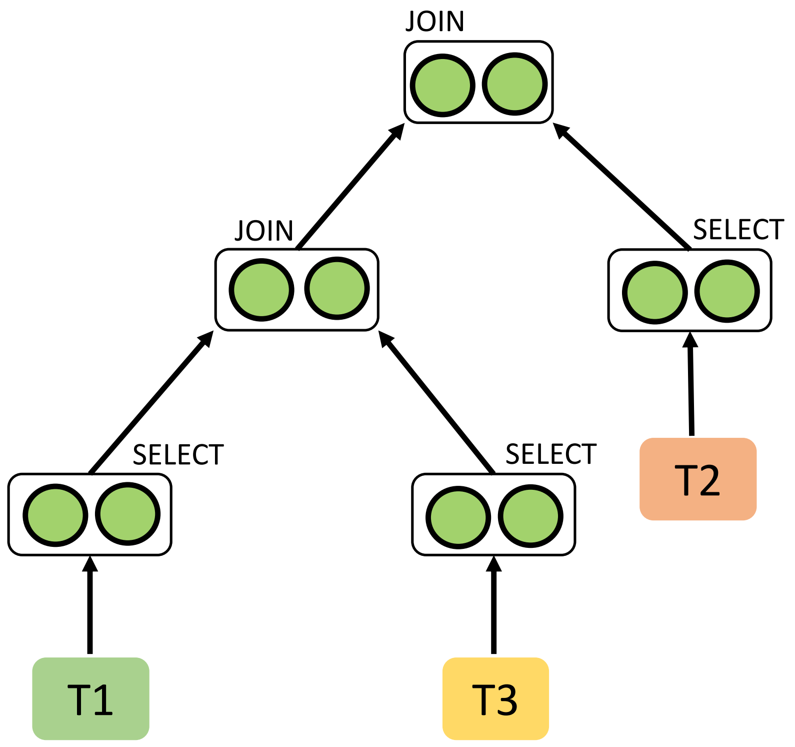
Choose plan with smallest run time for execution

Network aware placement and scheduling



Network aware placement and scheduling

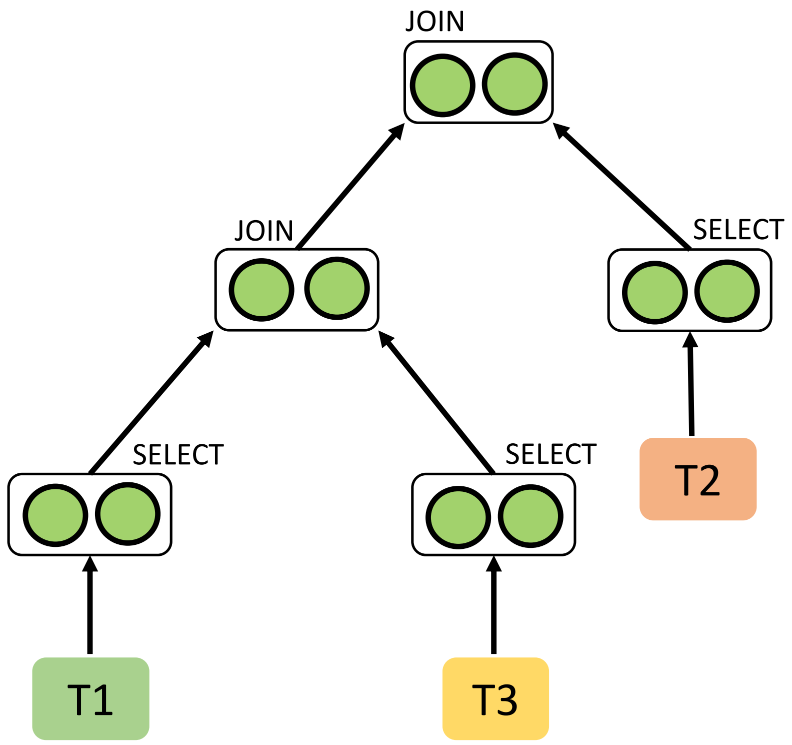
- Task placement decided greedily one stage at a time
  - Minimize per stage run time



Network aware placement and scheduling

- Task placement decided greedily one stage at a time
  - Minimize per stage run time

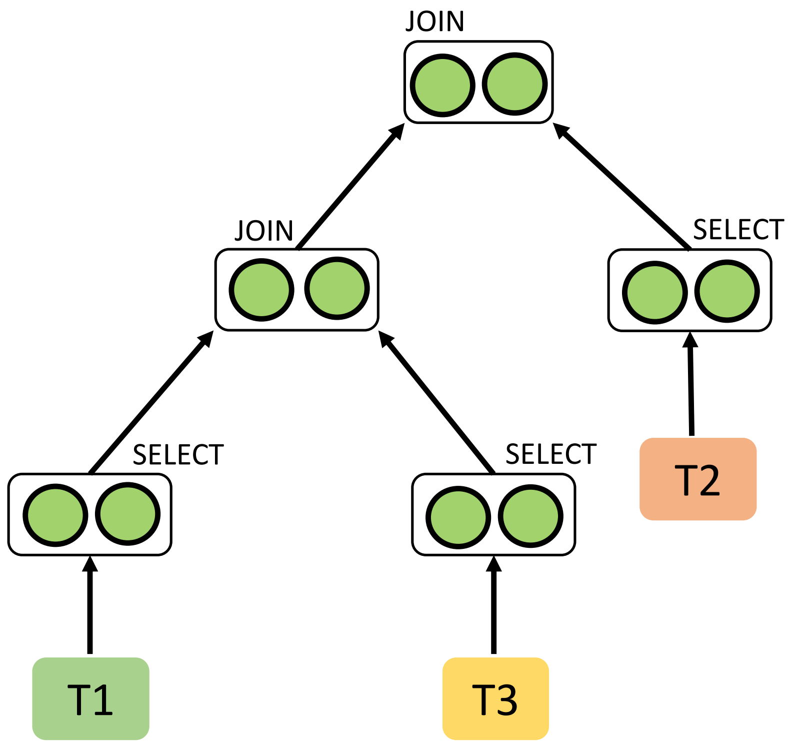
- Scheduling of network transfers
  - Determines start times of inter-DC network transfers



Network aware placement and scheduling

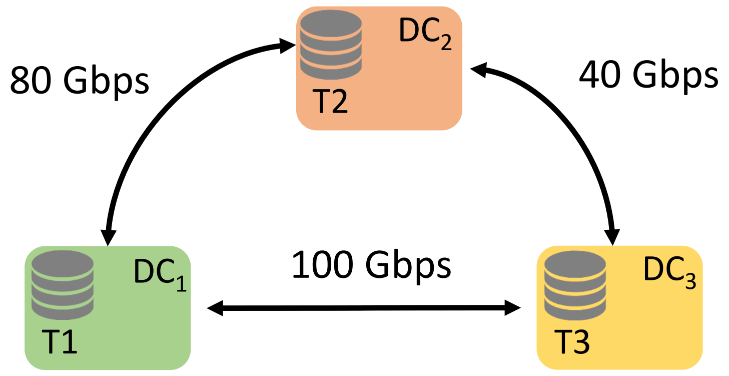
- Task placement decided greedily one stage at a time
  - Minimize per stage run time

- Scheduling of network transfers
  - Determines start times of inter-DC network transfers
  - Formulate a Binary Integer Linear Program to solve scheduling
  - Factors transfer dependencies

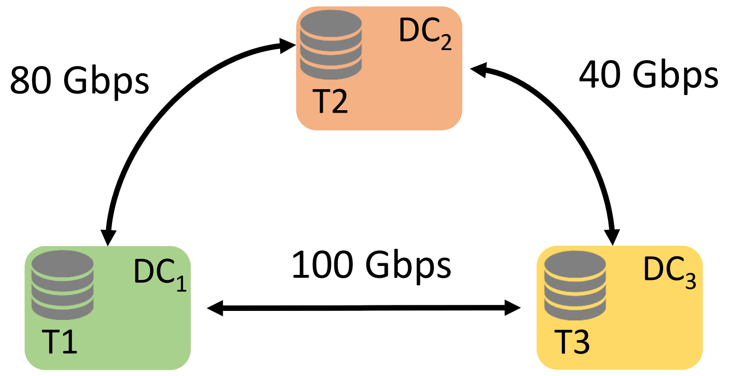


How to extend the late-binding strategy to multiple queries?

Queries affect each others' run time

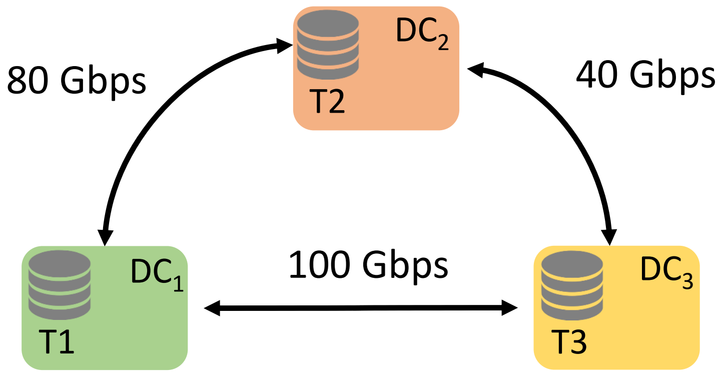


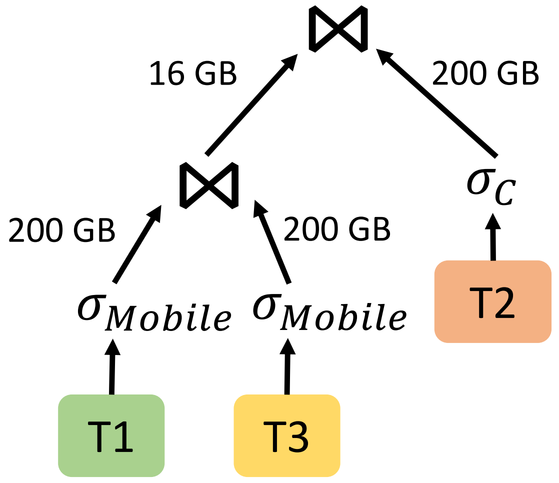
Queries affect each others' run time

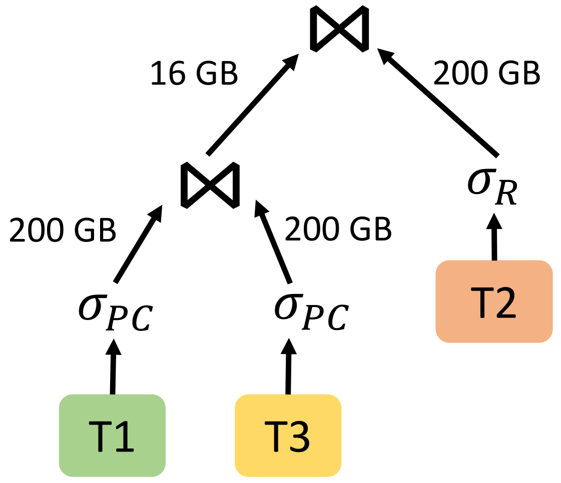


```
QUERY 1
SELECT …
device == "mobile"
…;

QUERY 2
SELECT …
genre == "pc"
…;
```

Queries affect each others' run time





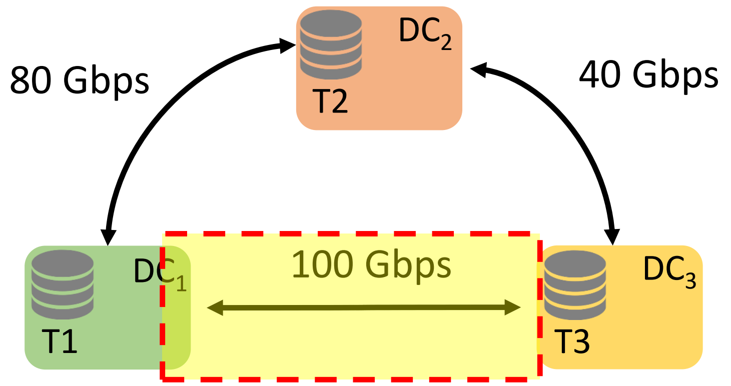


```
QUERY 1
SELECT …
device == "mobile"
…;
```

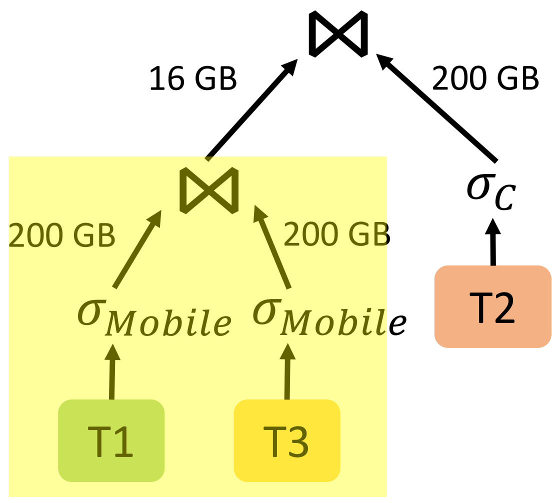
Same query plan (Plan C) for Query 1 and Query 2

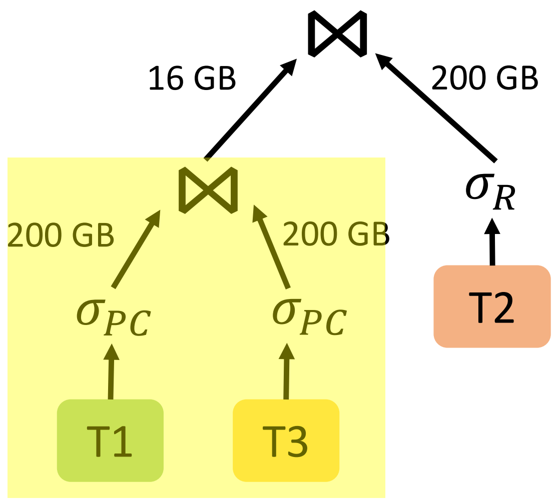
```
QUERY 2
SELECT …
genre == "pc"
…;
```

Queries affect each others' run time



Contention increases query run time

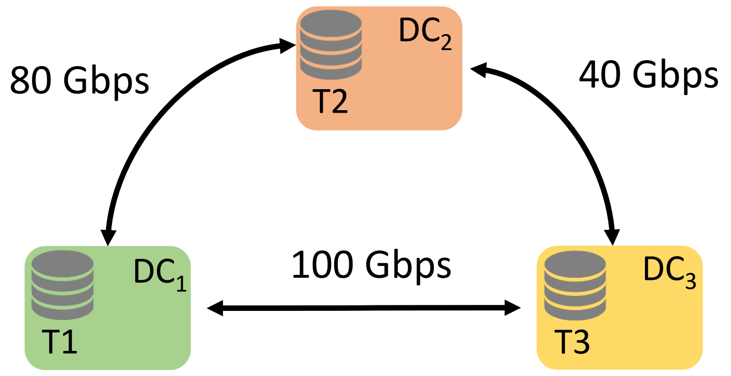




Same query plan (Plan C) for Query 1 and Query 2

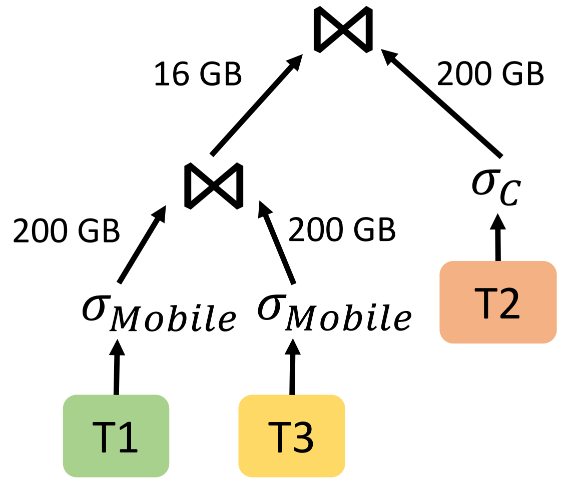
```
QUERY 1
SELECT …
device == "mobile"
…;

QUERY 2
SELECT …
genre == "pc"
…;
```

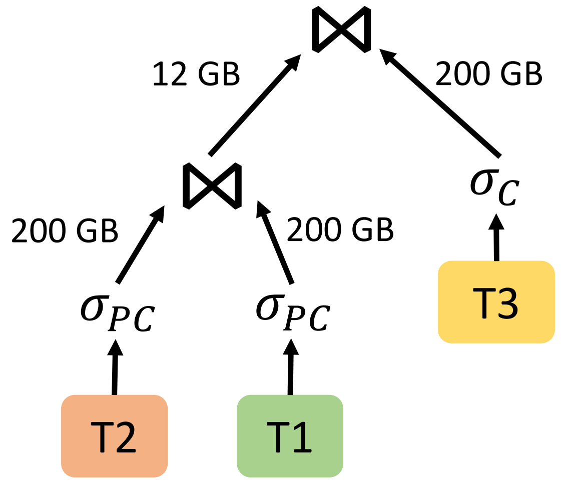
Queries affect each others' run time



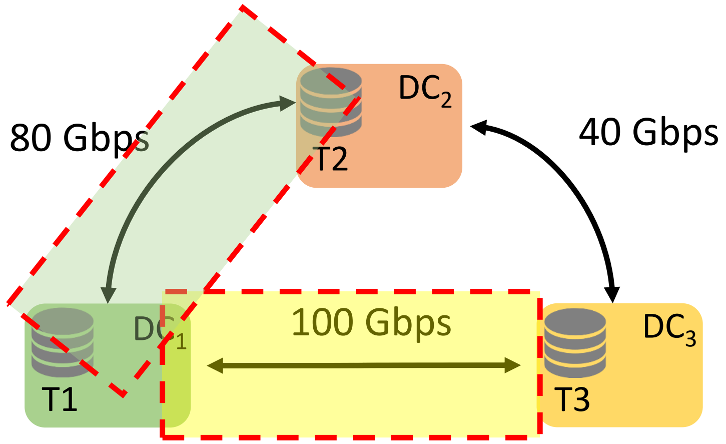
```
QUERY 1
SELECT …
device == "mobile"
…;

QUERY 2
SELECT …
genre == "pc"
…;
```

Different query plans for Query 1 (Plan C) and Query 2 (Plan B)





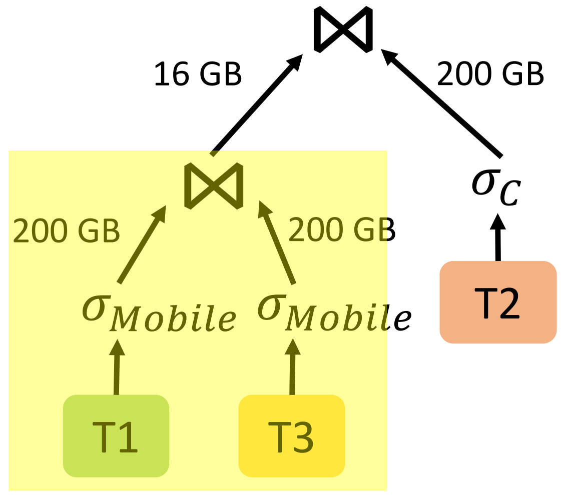
Queries affect each others' run time

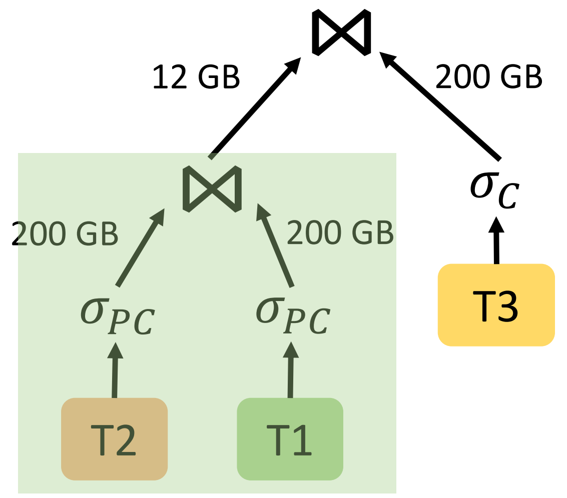


No contention of network links

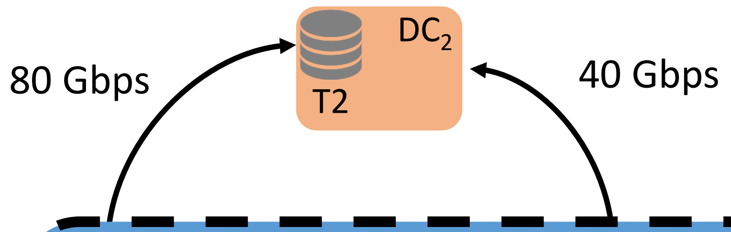
```
QUERY 1
SELECT …
device == "mobile"
…;

QUERY 2
SELECT …
genre == "pc"
…;
```

Different query plans for Query 1 (Plan C) and Query 2 (Plan B)





Queries affect each others' run time



Choosing execution plans jointly for multiple queries improves performance

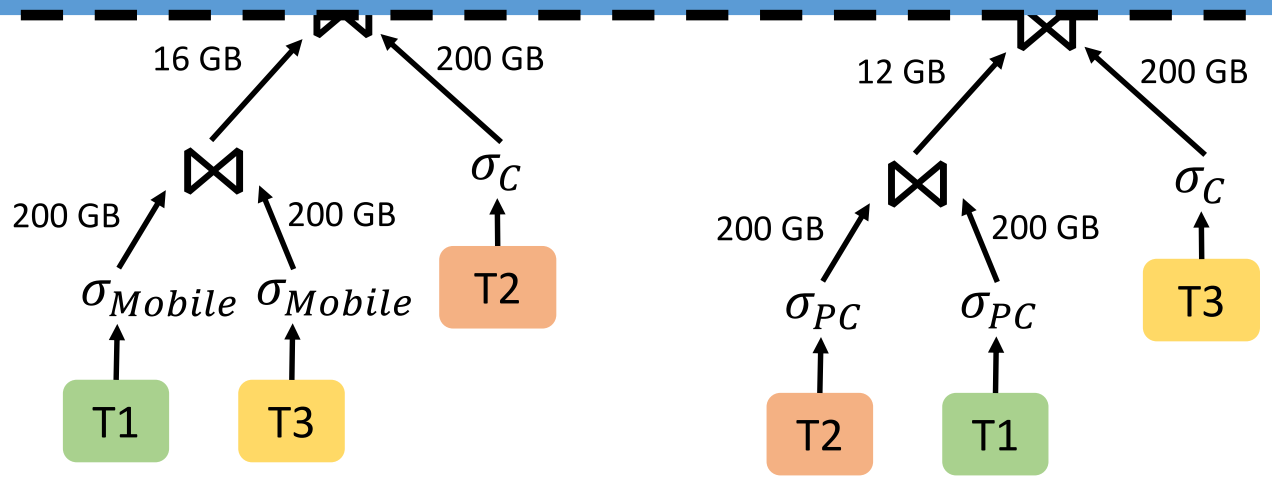
N

Qu

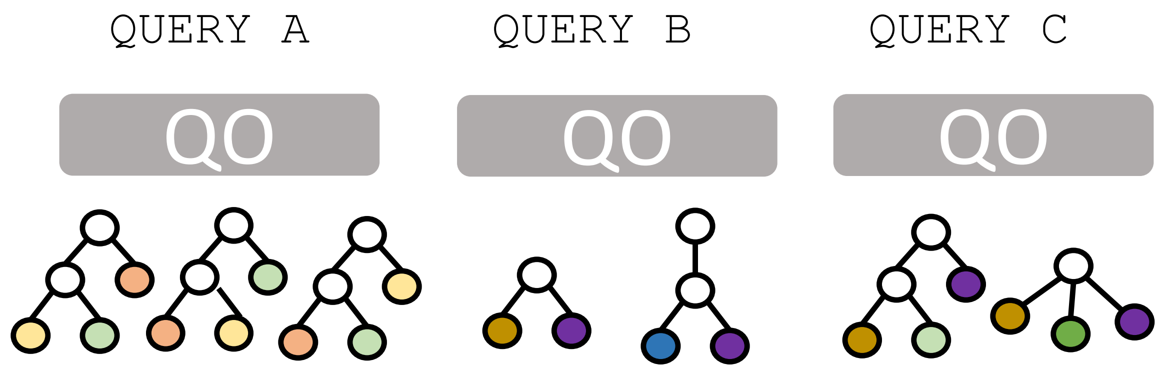
```
SELECT …
device == "mobile"
…;
```

QUERY 2

```
SELECT …
genre == "pc"
…;
```

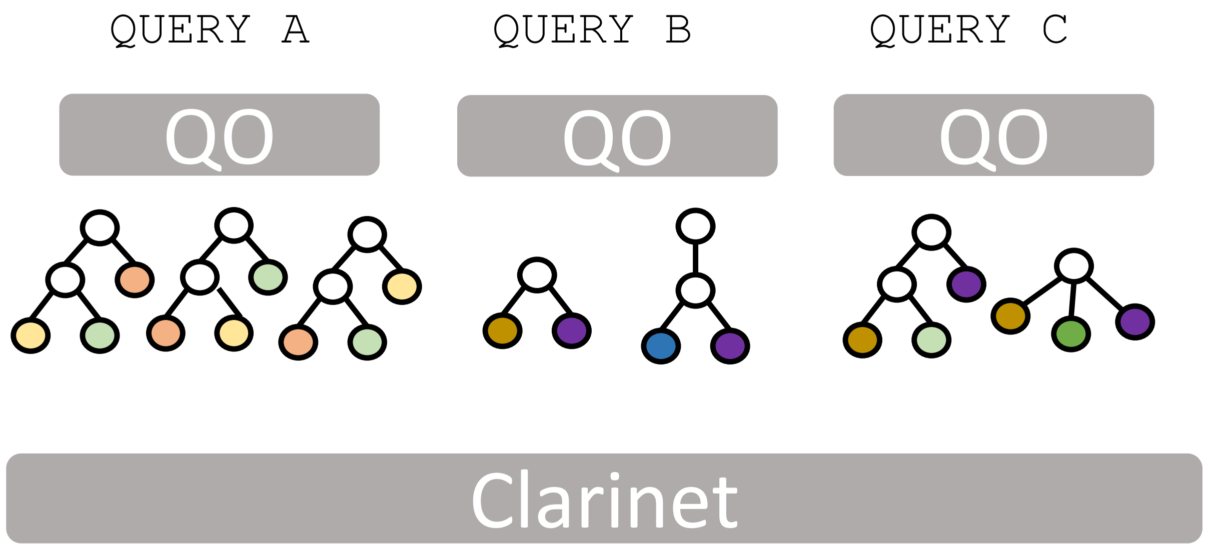


Iterative Shortest Job First



- Best combination → minimize average completion
  - Computationally intractable

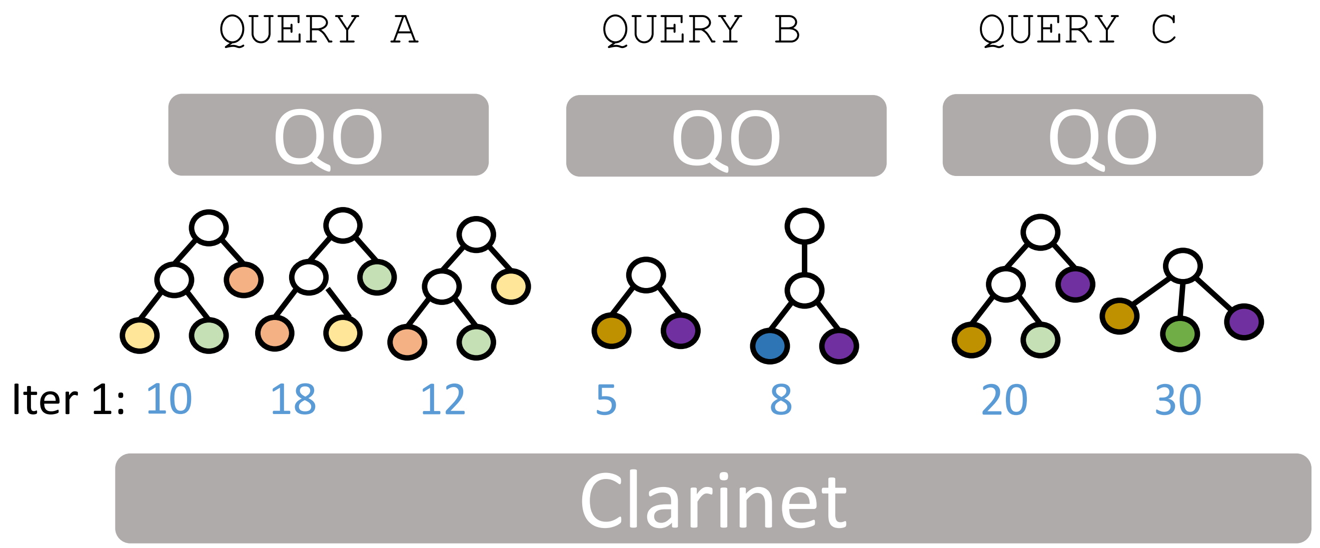
Iterative Shortest Job First



- Best combination → minimize average completion
  - Computationally intractable

- Iterative Shortest Job First (SJF) scheduling heuristic
  1. Pick shortest physical query plan in each iteration

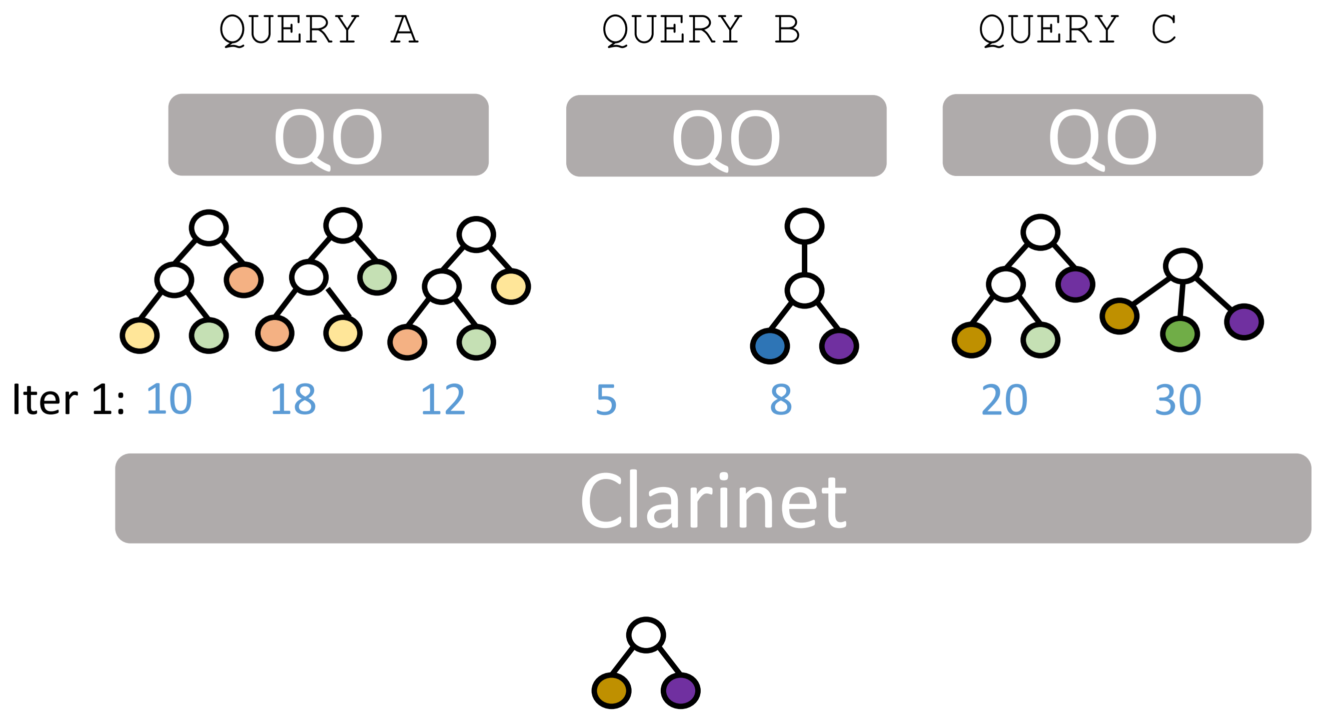
Iterative Shortest Job First



- Best combination → minimize average completion
    - Computationally intractable

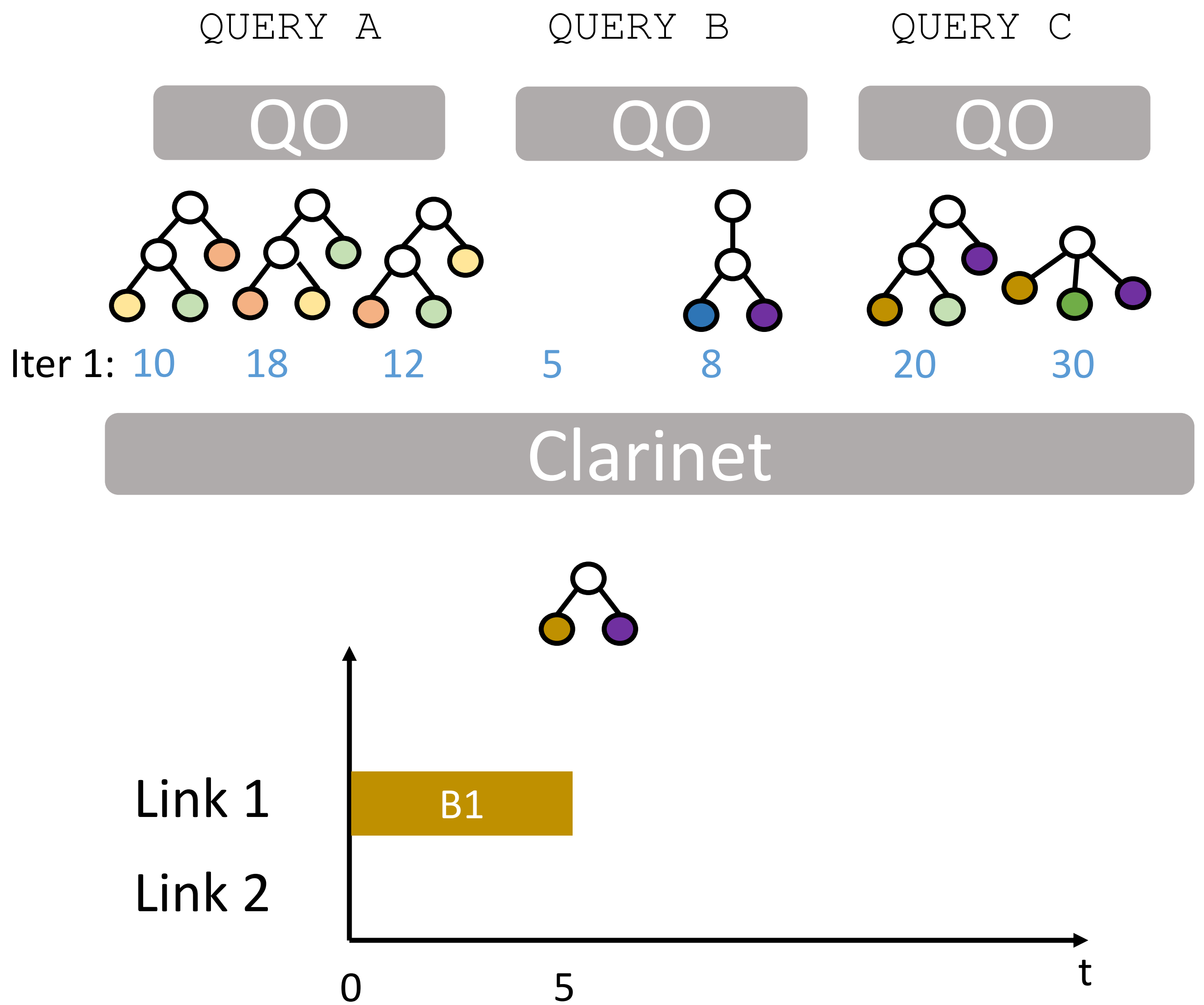
- Iterative Shortest Job First (SJF) scheduling heuristic
    1. Pick shortest physical query plan in each iteration

Iterative Shortest Job First



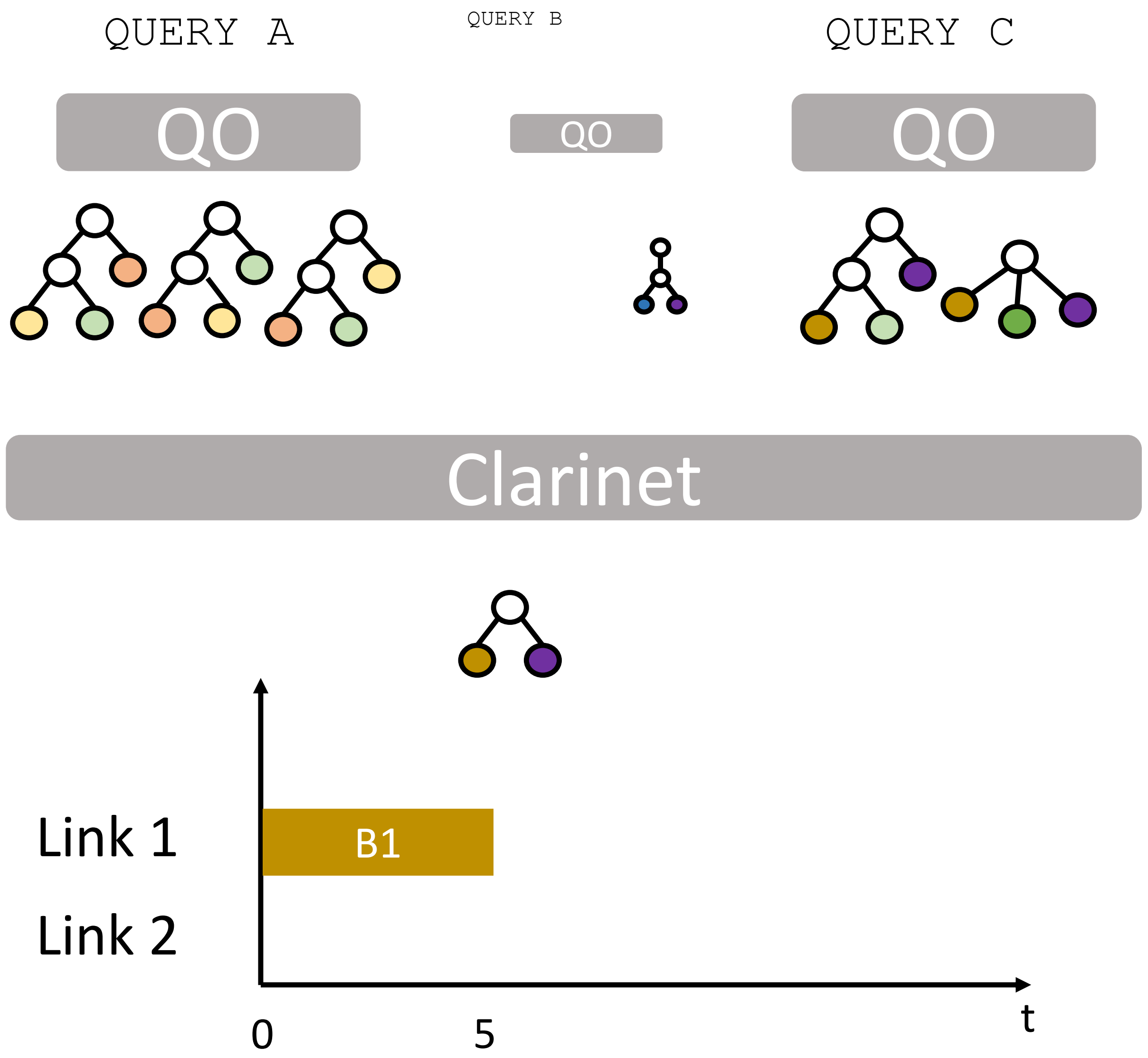
- Best combination → minimize average completion
    - Computationally intractable
- Iterative Shortest Job First (SJF) scheduling heuristic
    1. Pick shortest physical query plan in each iteration

Iterative Shortest Job First



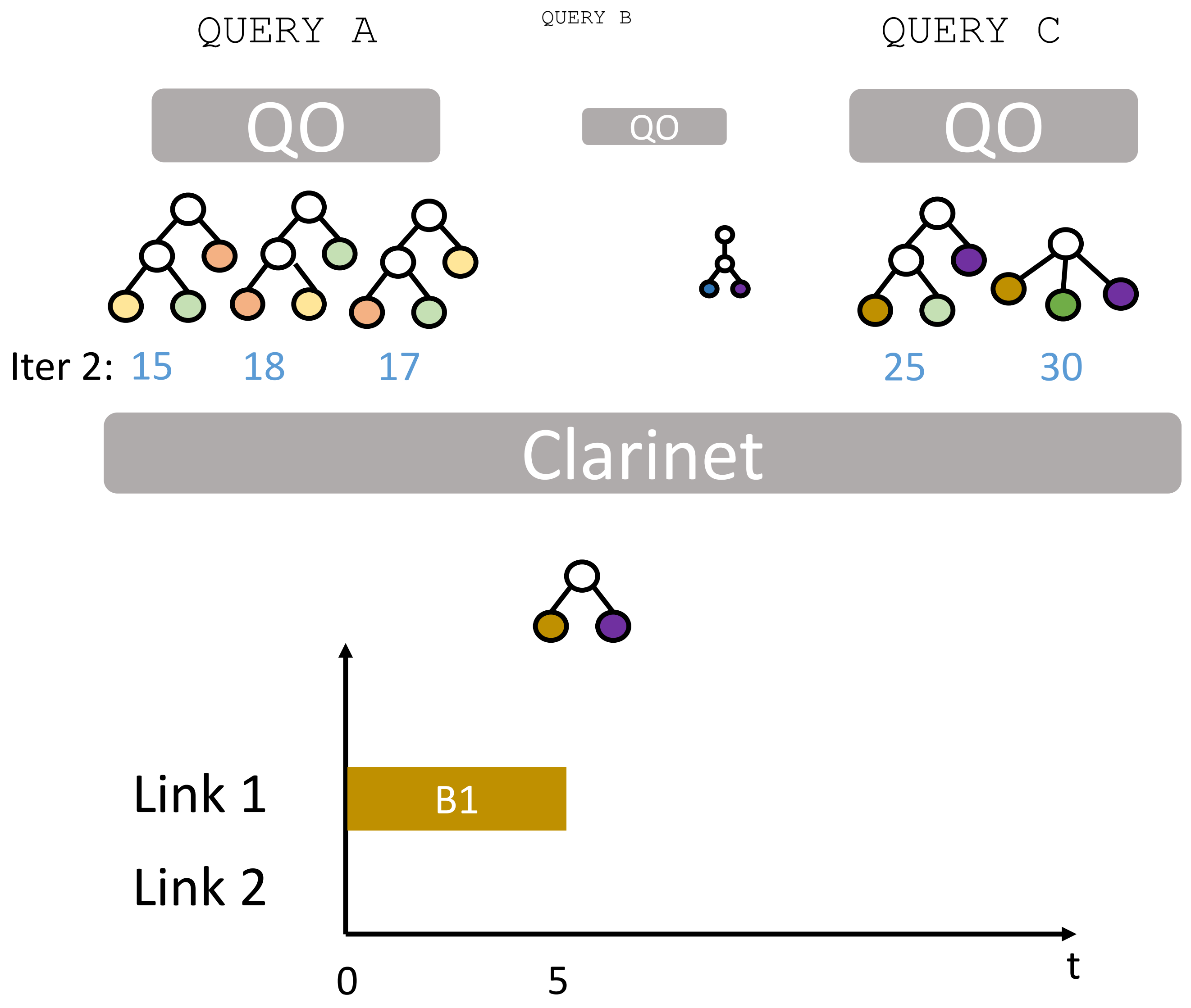
- Best combination → minimize average completion
  - Computationally intractable
- Iterative Shortest Job First (SJF) scheduling heuristic
  1. Pick shortest physical query plan in each iteration
- Reserve bandwidth to guarantee completion time

Iterative Shortest Job First



- Best combination → minimize average completion
    - Computationally intractable
- Iterative Shortest Job First (SJF) scheduling heuristic
    1. Pick shortest physical query plan in each iteration
- Reserve bandwidth to guarantee completion time

Iterative Shortest Job First

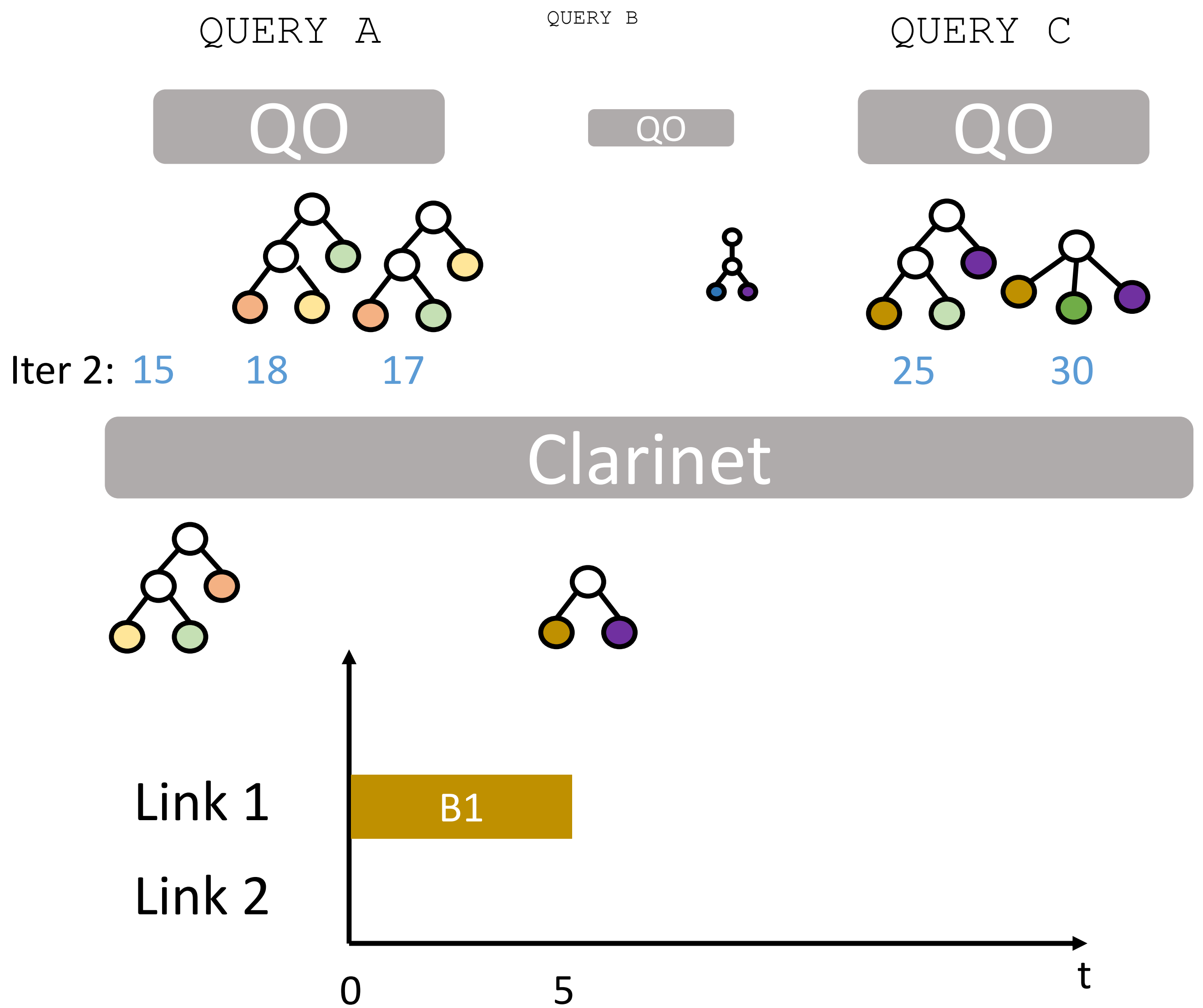


- Best combination → minimize average completion
  - Computationally intractable

- Iterative Shortest Job First (SJF) scheduling heuristic
  1. Pick shortest physical query plan in each iteration

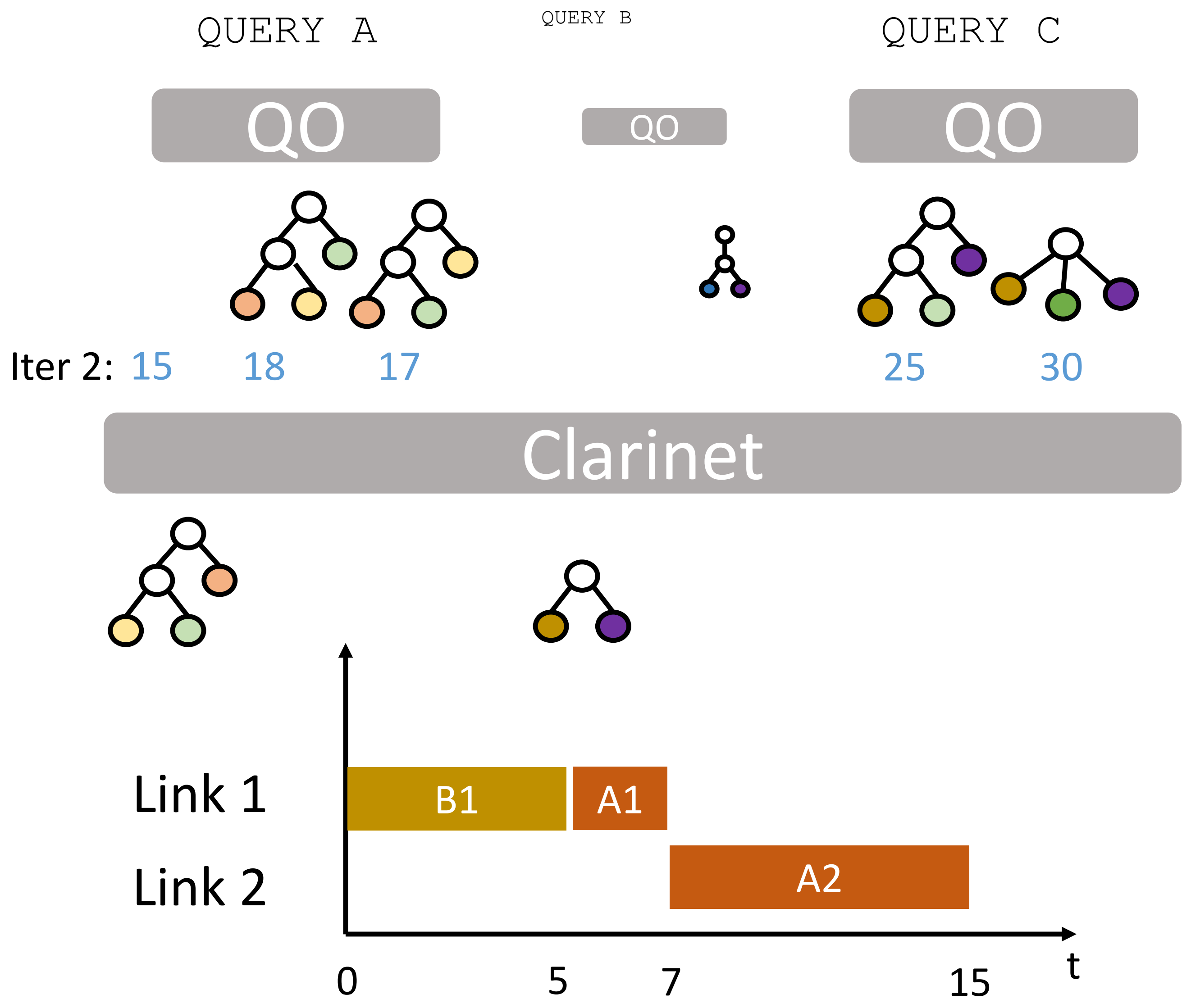
- Reserve bandwidth to guarantee completion time

Iterative Shortest Job First



- Best combination → minimize average completion
  - Computationally intractable
- Iterative Shortest Job First (SJF) scheduling heuristic
  1. Pick shortest physical query plan in each iteration
- Reserve bandwidth to guarantee completion time

Iterative Shortest Job First



- Best combination → minimize average completion
    - Computationally intractable
- Iterative Shortest Job First (SJF) scheduling heuristic
    1. Pick shortest physical query plan in each iteration
- Reserve bandwidth to guarantee completion time

Avoid fragmentation and improve completion time

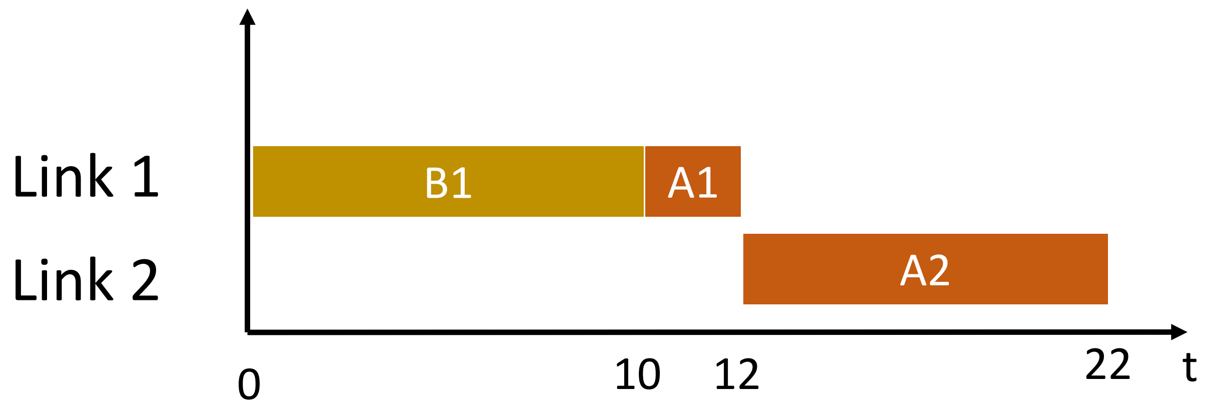
Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

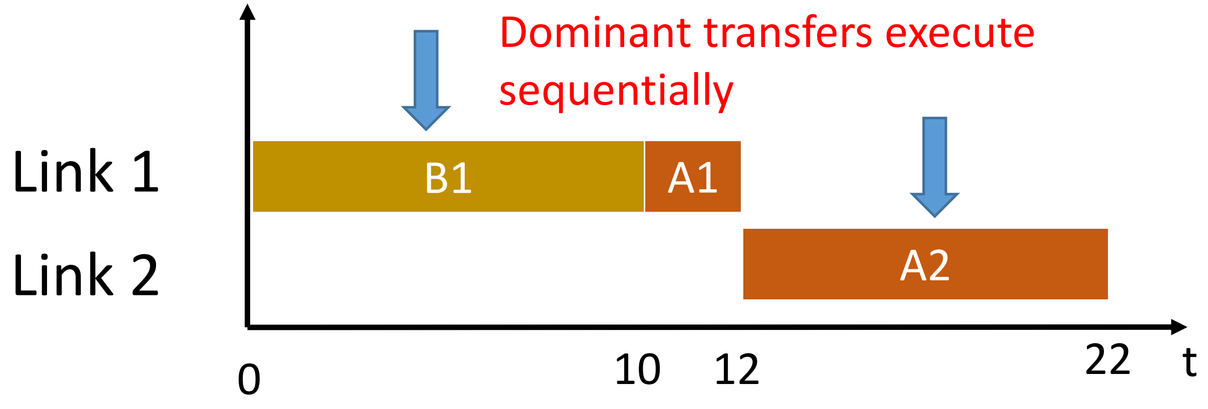
Scheduled in SJF order



Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

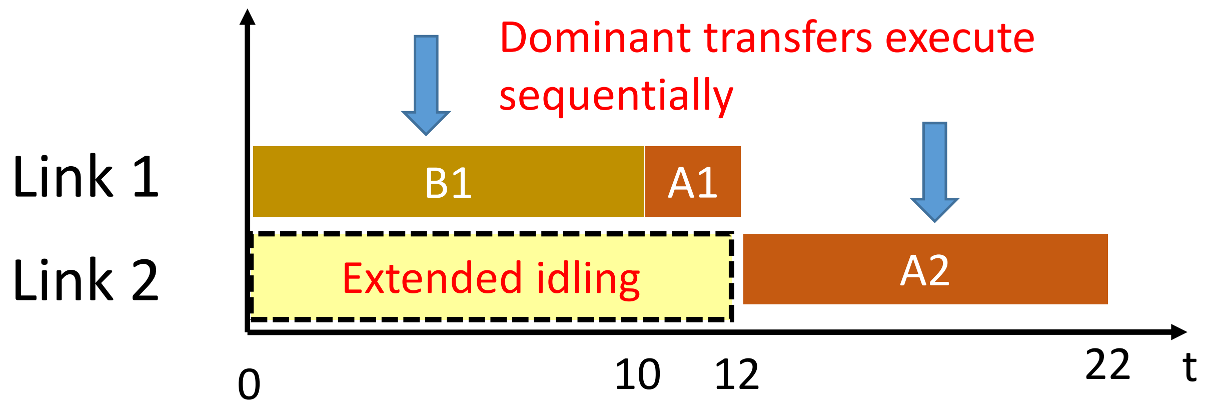
Scheduled in SJF order



Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

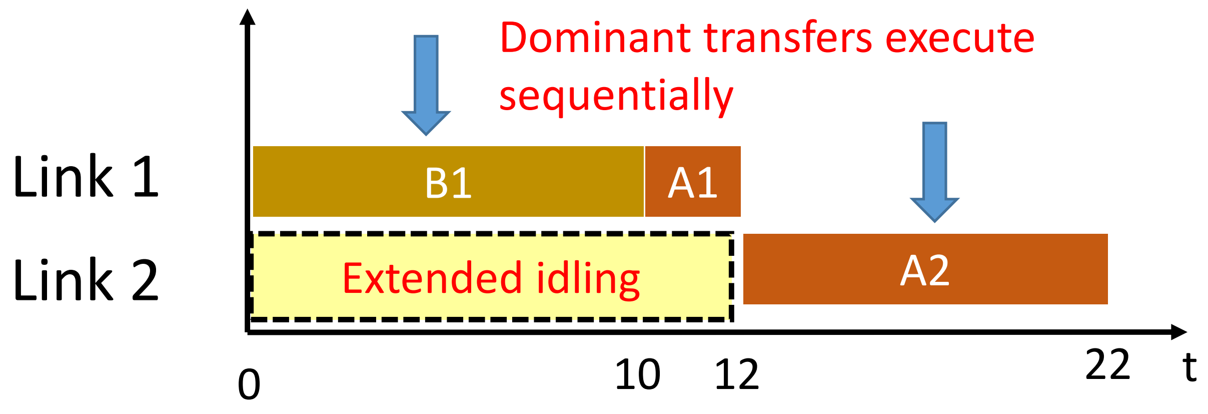
Scheduled in SJF order



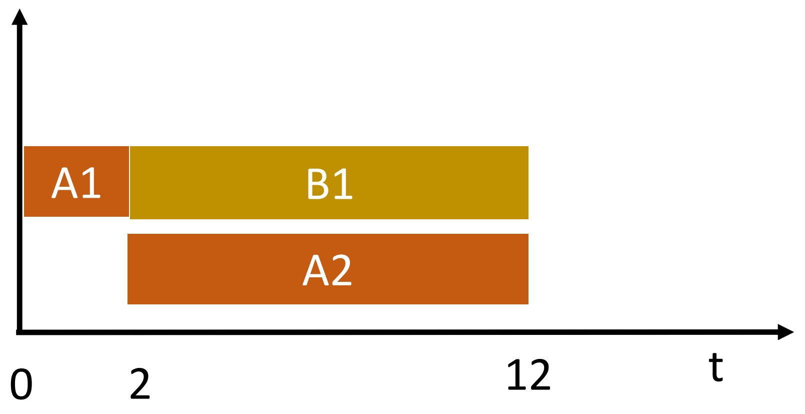
Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

Scheduled in SJF order

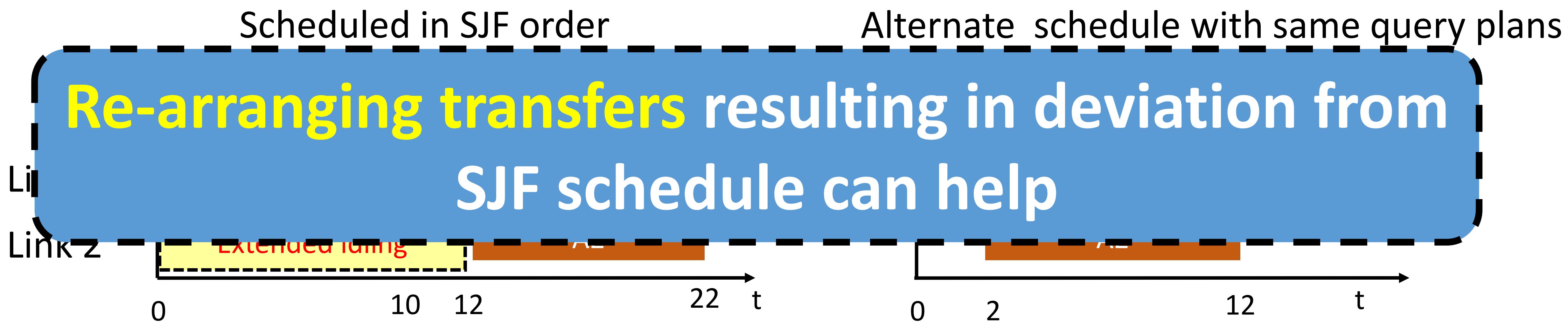


Alternate schedule with same query plans



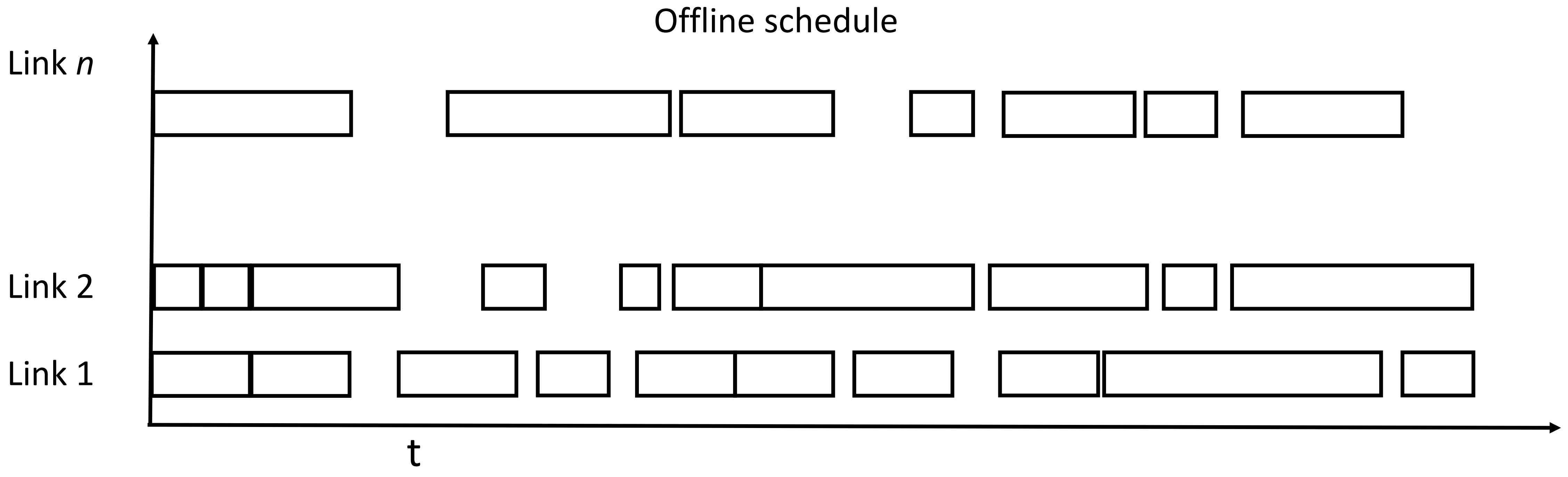
Avoid fragmentation and improve completion time

- SJF & reservation leads to bandwidth fragmentation

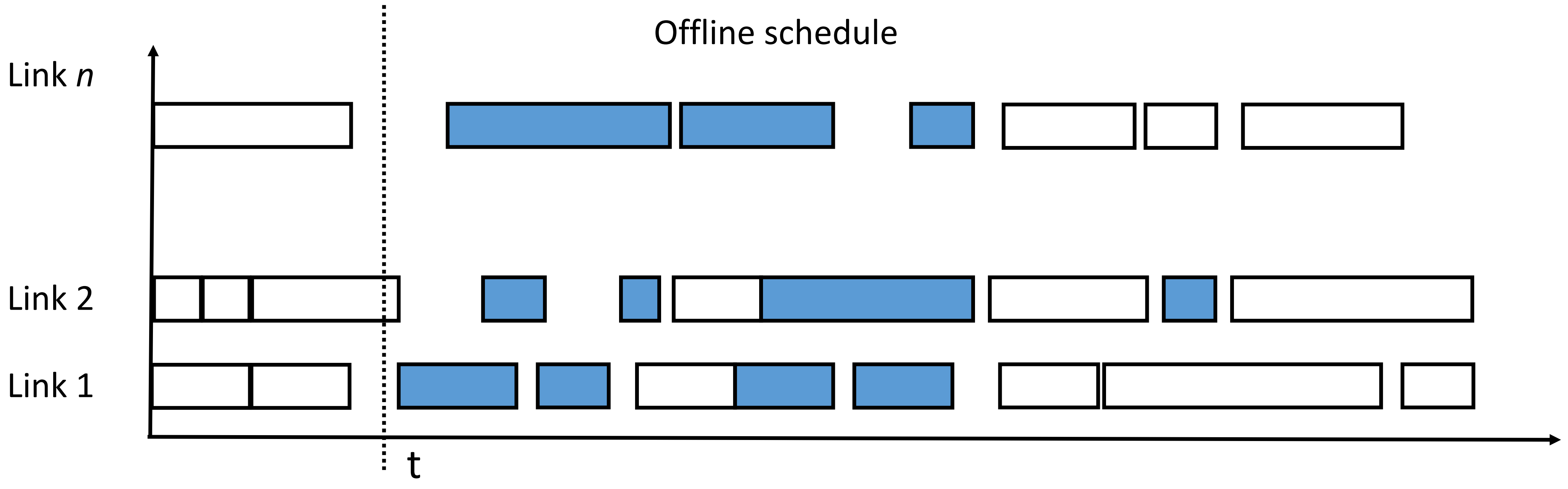


Re-arranging transfers resulting in deviation from SJF schedule can help

k-Shortest Jobs First Heuristic

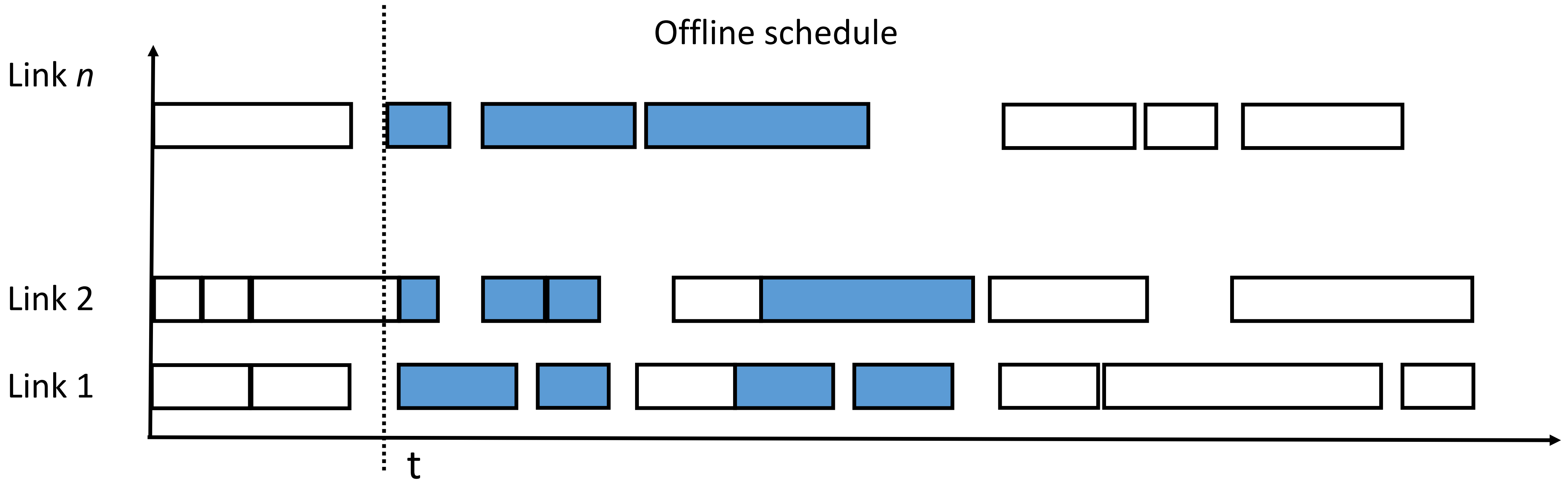


k-Shortest Jobs First Heuristic



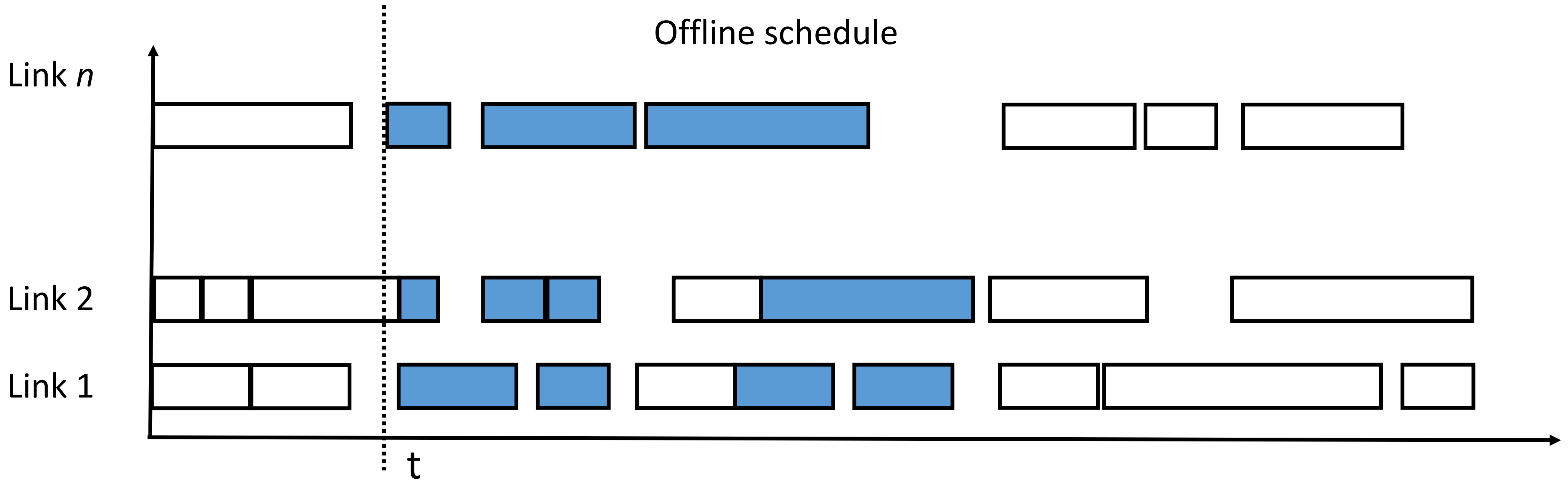
- Identify transfers of k-shortest yet incomplete jobs

k-Shortest Jobs First Heuristic



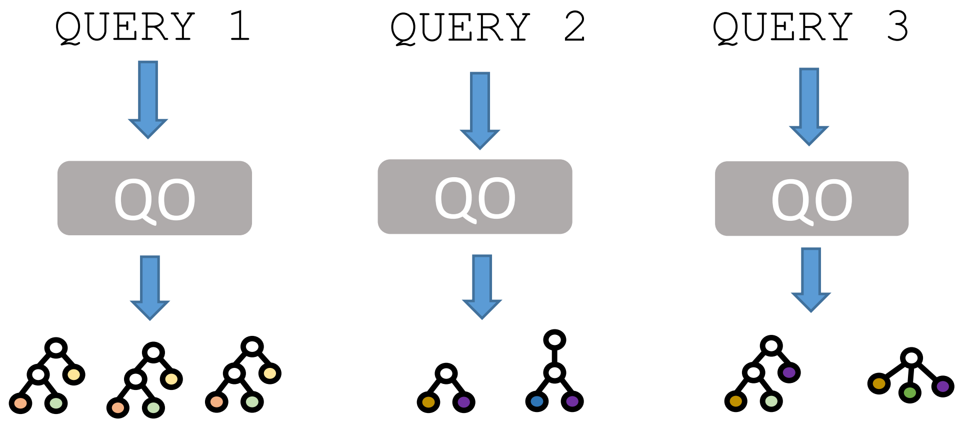
- Identify transfers of k-shortest yet incomplete jobs
- Relax transfer schedule → Start as soon as link is free and task is available

k-Shortest Jobs First Heuristic



- Identify transfers of k-shortest yet incomplete jobs
- Relax transfer schedule → Start as soon as link is free and task is available
- Best 'k' ←Prior observations (or) through offline simulations

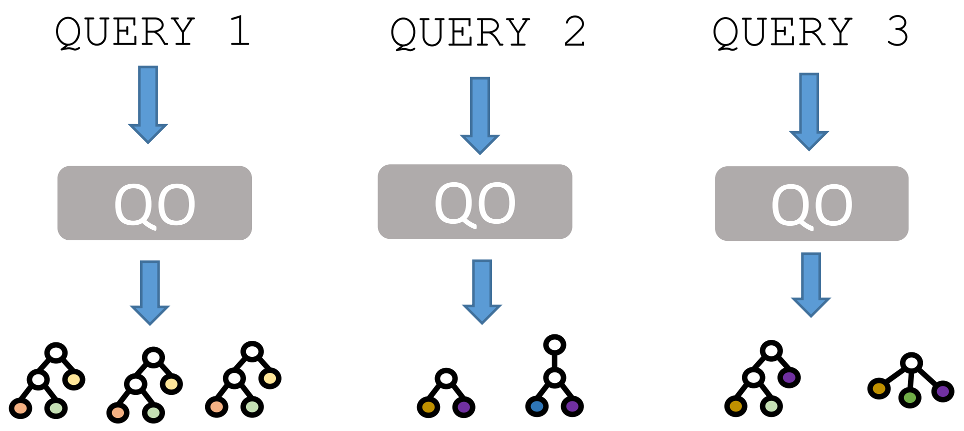
Clarinet Implementation



Batch of queries

Existing Query Optimizers

Clarinet Implementation

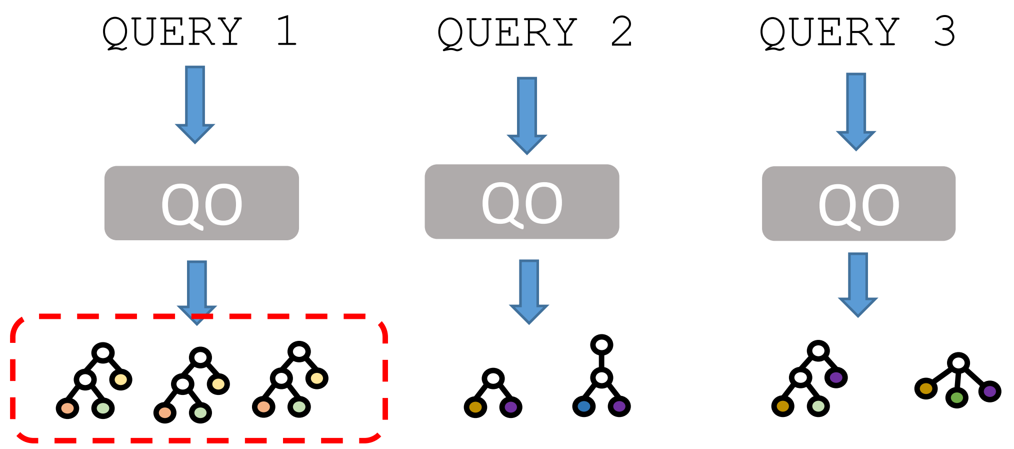


Batch of queries

Existing Query Optimizers

- Modified Hive to generate multiple plans

Clarinet Implementation

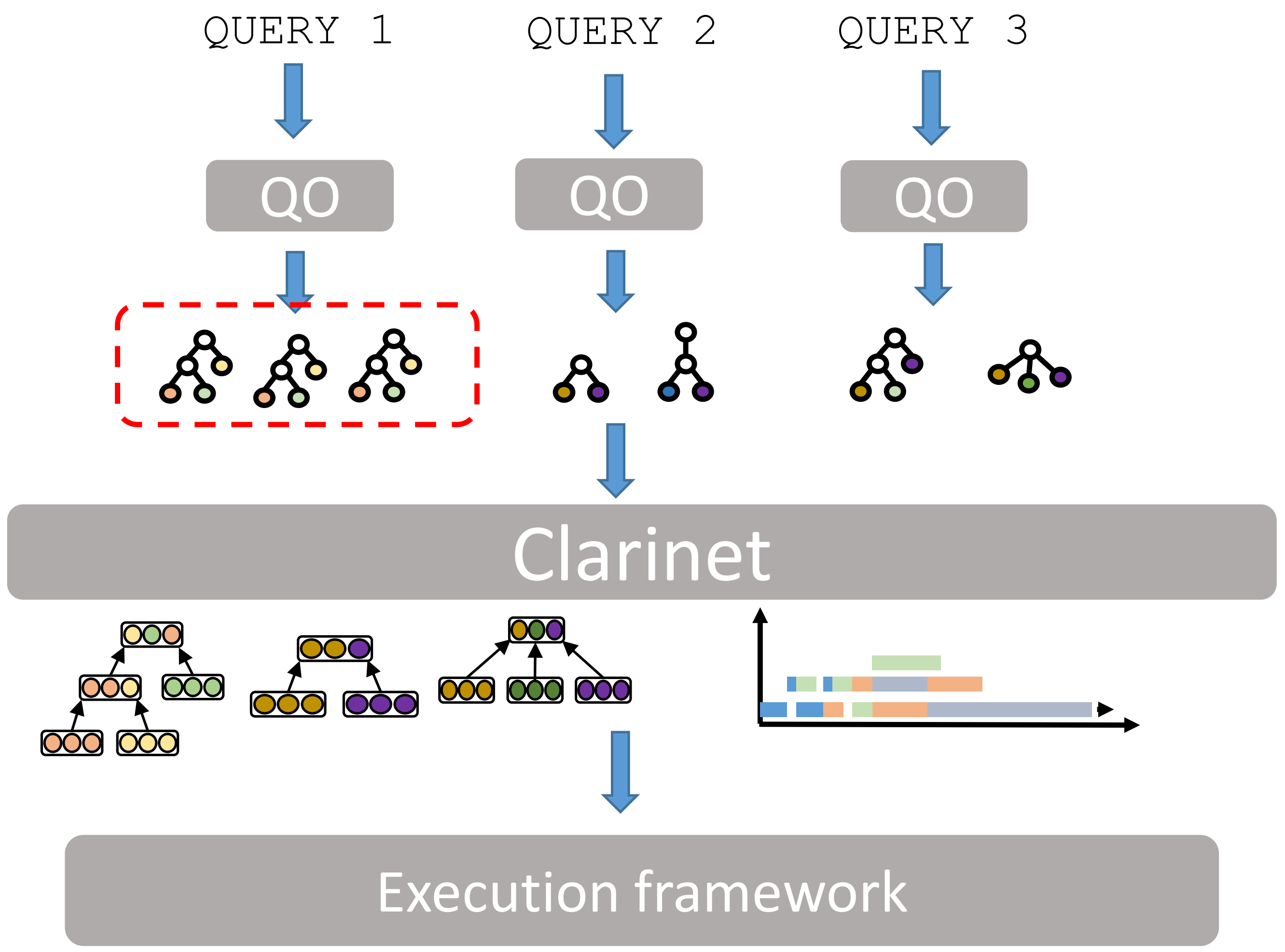


Batch of queries

Existing Query Optimizers

- Modified Hive to generate multiple plans
- QOs control set of generated plans
- Existing optimizations are applied
  - Push down Select
  - Partition pruning

Clarinet Implementation



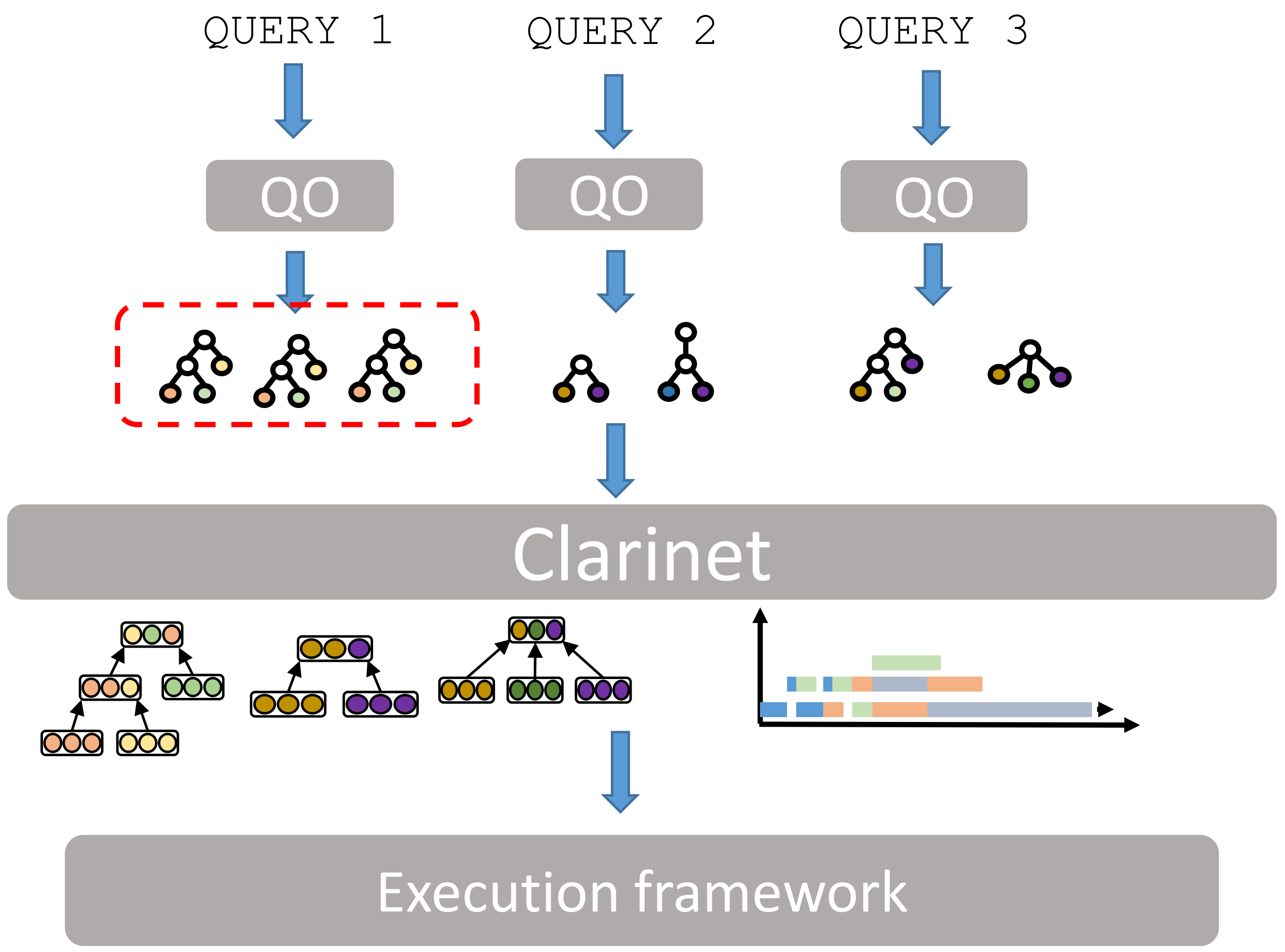
Batch of queries

Existing Query Optimizers

- Modified Hive to generate multiple plans
- QOs control set of generated plans
- Existing optimizations are applied
    - Push down Select
    - Partition pruning

Enforces Clarinet's schedule

Clarinet Implementation



Batch of queries

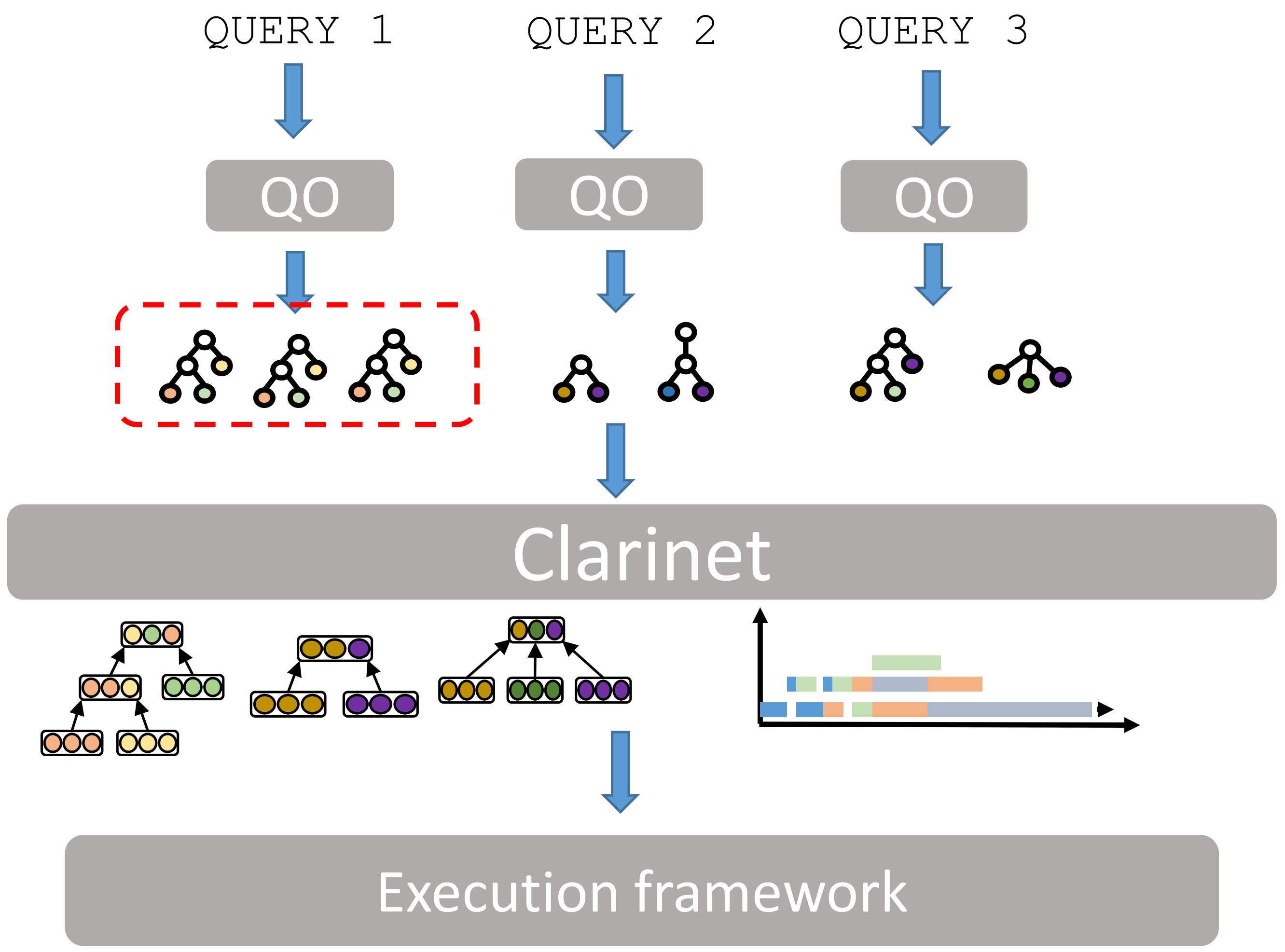
Existing Query Optimizers

- Modified Hive to generate multiple plans
- QOs control set of generated plans
- Existing optimizations are applied
    - Push down Select
    - Partition pruning

Enforces Clarinet's schedule

- Modified Tez's DAGScheduler

Clarinet Implementation



Batch of queries

Online query arrivals

Existing Query Optimizers

- Modified Hive to generate multiple plans
- QOs control set of generated plans
- Existing optimizations are applied
    - Push down Select
    - Partition pruning

Enforces Clarinet's schedule

- Modified Tez's DAGScheduler
- Fairness guarantees

Evaluation

Compare Clarinet with following GDA approaches:

Evaluation

Compare Clarinet with following GDA approaches:

1. Hive
2. Hive + Iridium
3. Hive + Reducers in single DC

Evaluation

Compare Clarinet with following GDA approaches:

1. Hive : WAN agnostic task placement + scheduling
2. Hive + Iridium
3. Hive + Reducers in single DC

Evaluation

Compare Clarinet with following GDA approaches:

1. Hive : WAN agnostic task placement + scheduling
2. Hive + Iridium : WAN aware task placement across DCs
3. Hive + Reducers in single DC

Evaluation

Compare Clarinet with following GDA approaches:

1.	Hive	: WAN agnostic task placement + scheduling
2.	Hive + Iridium	: WAN aware task placement across DCs
3.	Hive + Reducers in single DC	: Distributed filtering + central aggregation

Evaluation

Compare Clarinet with following GDA approaches:

1.	Hive	: WAN agnostic task placement + scheduling
2.	Hive + Iridium	: WAN aware task placement across DCs
3.	Hive + Reducers in single DC	: Distributed filtering + central aggregation

- Geo-Distributed Analytics stack across 10 EC2 regions

Evaluation

Compare Clarinet with following GDA approaches:

1. Hive	: WAN agnostic task placement + scheduling
2. Hive + Iridium	: WAN aware task placement across DCs
3. Hive + Reducers in single DC	: Distributed filtering + central aggregation

- Geo-Distributed Analytics stack across 10 EC2 regions
- Workload:
    - 30 batches of 12 randomly chosen TPC-DS queries

Evaluation: Reduction in average completion time

GDA Approach Vs. Hive	Average Gains
Clarinet	2.7x
Hive + Iridium	1.5x

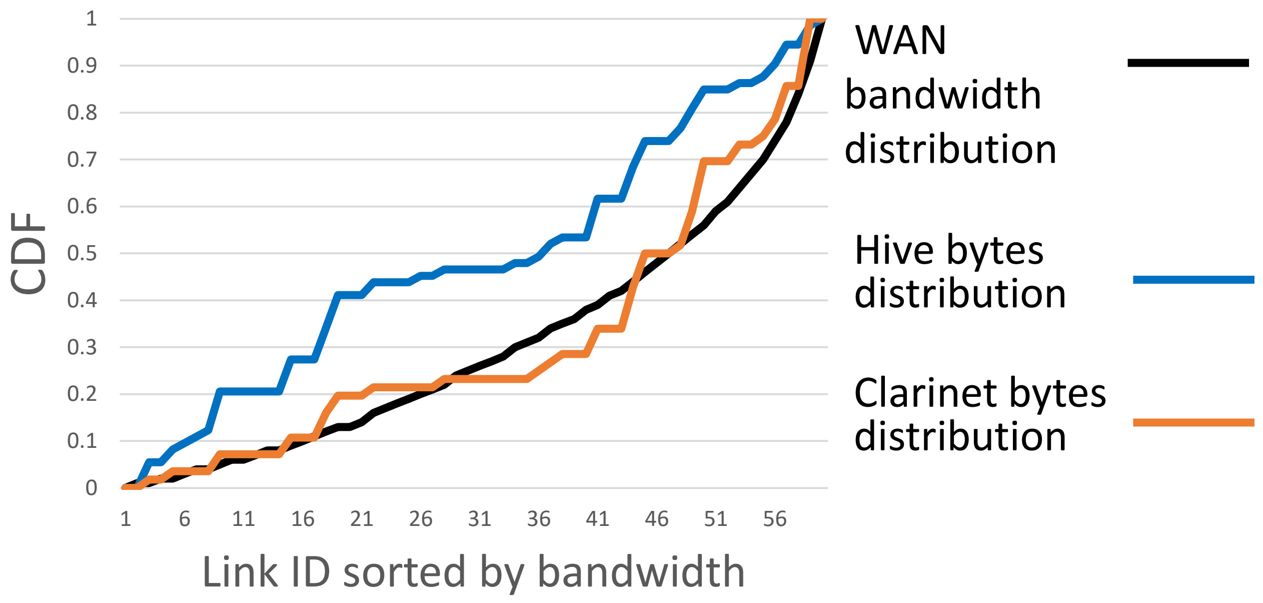
Evaluation: Reduction in average completion time

GDA Approach Vs. Hive	Average Gains
Clarinet	2.7x
Hive + Iridium	1.5x

Clarinet chooses a different plan for 75% of queries

Evaluation: Reduction in average completion time

GDA Approach Vs. Hive	Average Gains
Clarinet	2.7x
Hive + Iridium	1.5x

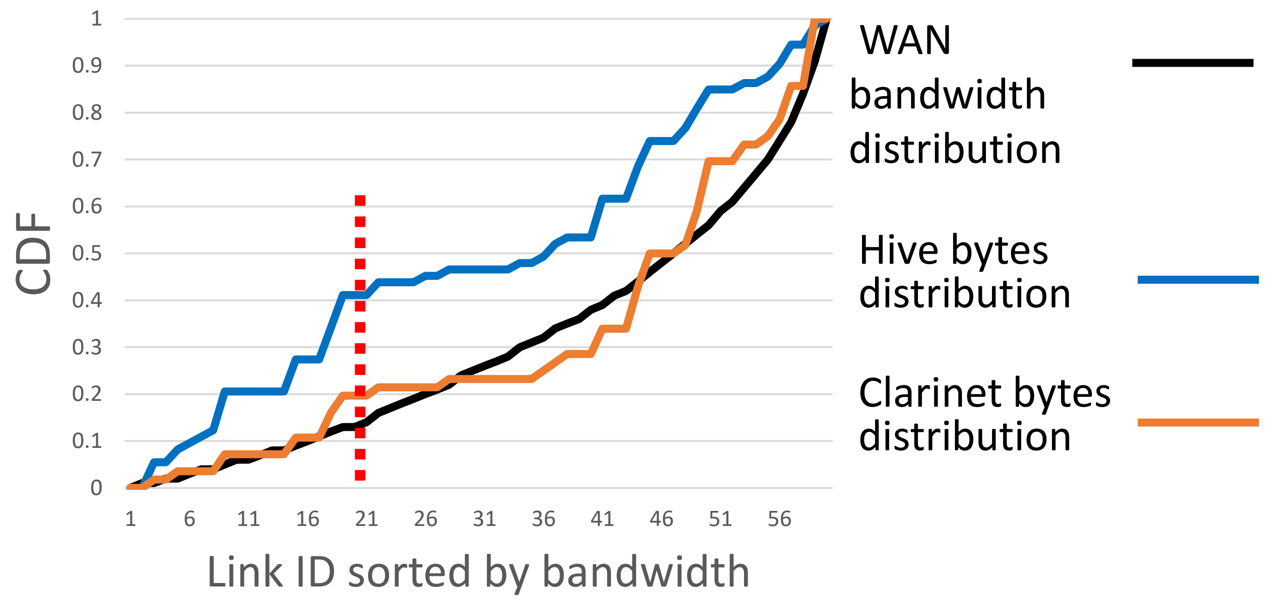


Data from a single batch 12 queries

Clarinet chooses a different plan for 75% of queries

Evaluation: Reduction in average completion time

GDA Approach Vs. Hive	Average Gains
Clarinet	2.7x
Hive + Iridium	1.5x



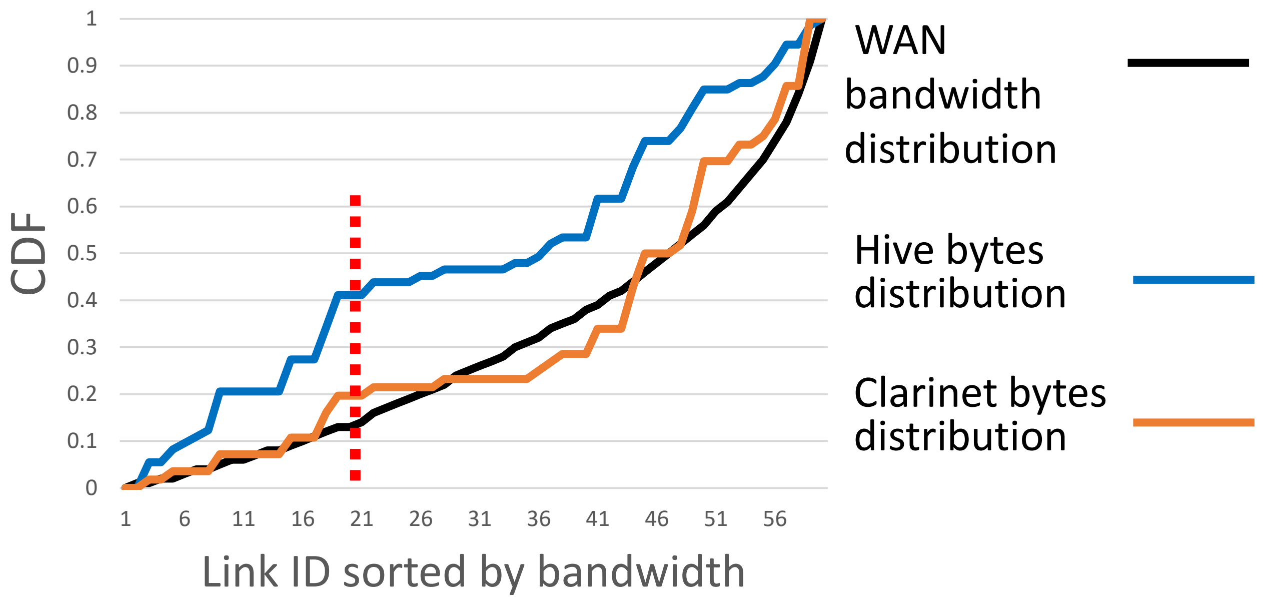
Data from a single batch 12 queries

Clarinet chooses a different plan for 75% of queries

Evaluation: Reduction in average completion time

GDA Approach Vs. Hive	Average Gains
Clarinet	2.7x
Hive + Iridium	1.5x
Hive + Reducers in single DC	0.6x

Clarinet chooses a different plan for 75% of queries



Data from a single batch 12 queries

Evaluation: Optimization overhead

Evaluation: Optimization overhead

1. Generate multiple query plans

2. Iterative multi-query plan selection

Evaluation: Optimization overhead

1. Generate multiple query plans
    - Up to 64 plans in less than 5 s

2. Iterative multi-query plan selection

Evaluation: Optimization overhead

1. Generate multiple query plans
   - Up to 64 plans in less than 5 s

2. Iterative multi-query plan selection
   - Max. 15 s for batches with 12 queries

Evaluation: Optimization overhead

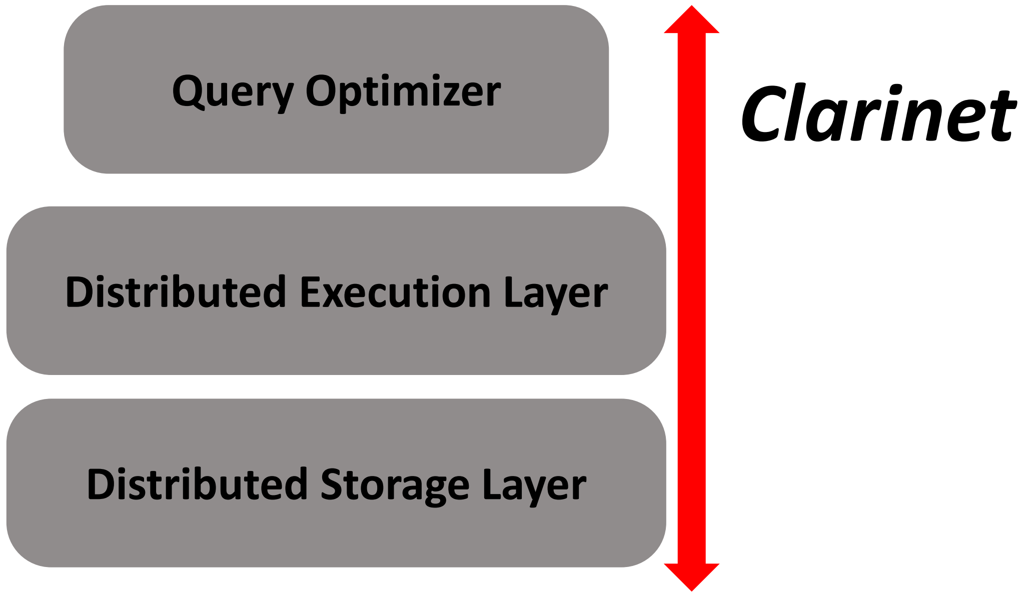
1. Generate multiple query plans
   - Up to 64 plans in less than 5 s

2. Iterative multi-query plan selection
   - Max. 15 s for batches with 12 queries

Insignificant w.r.t. query running times (order of 10’s of minutes)

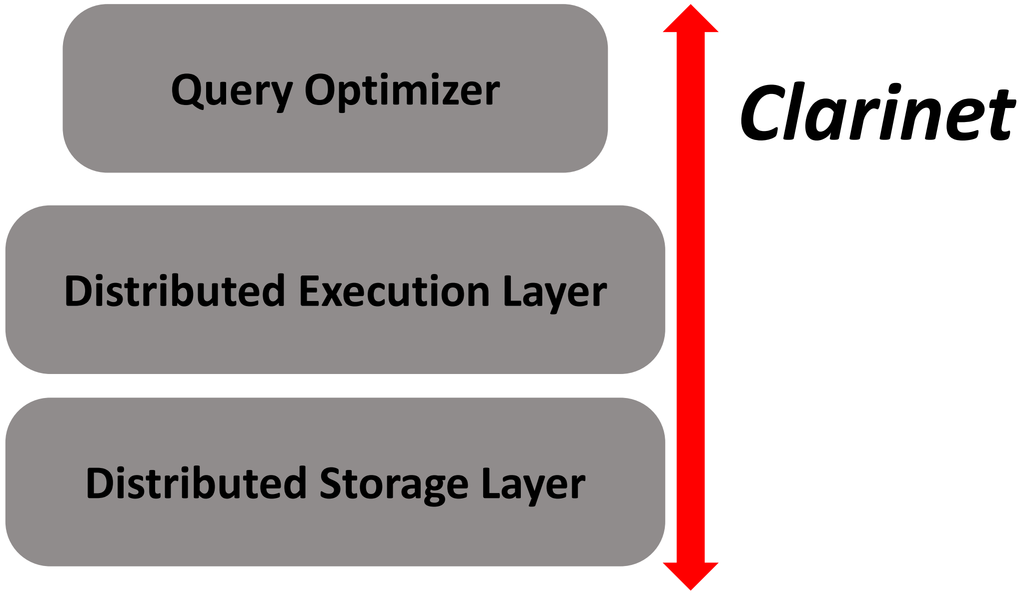
Summary

- WAN-awareness in QO + cross-layer optimization



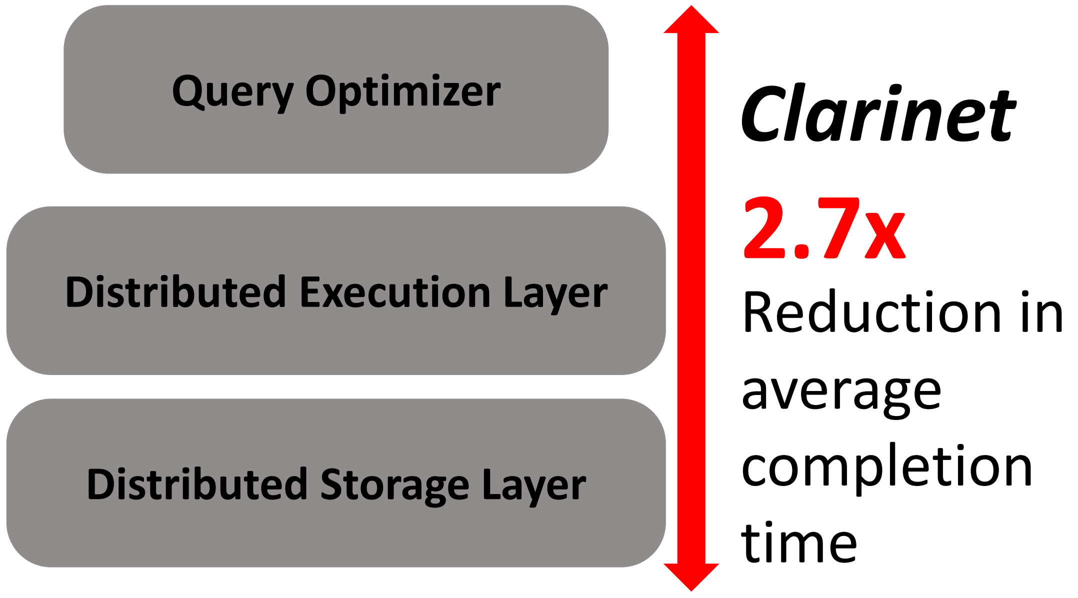
Summary

- WAN-awareness in QO + cross-layer optimization
- Presented a **scalable** way to implement **multi-query optimization** with minimal overhead



Summary

- WAN-awareness in QO + cross-layer optimization
- Presented a **scalable** way to implement **multi-query optimization** with minimal overhead


Query Optimizer
Distributed Execution Layer
Distributed Storage Layer
Clarinet
2.7x
Reduction in average completion time