Sol

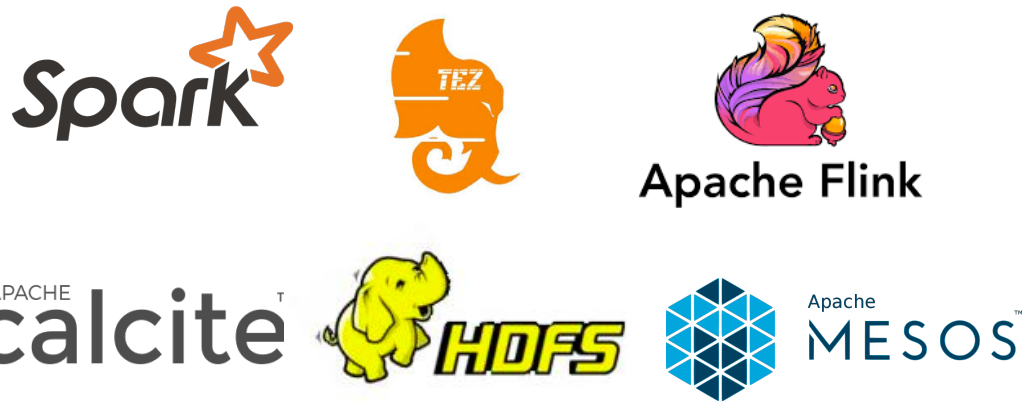
Fast Distributed Computation Over Slow Networks

Fan Lai, Jie You, Xiangfeng Zhu

Harsha V. Madhyastha, Mosharaf Chowdhury

Distributed Data Processing is Ubiquitous

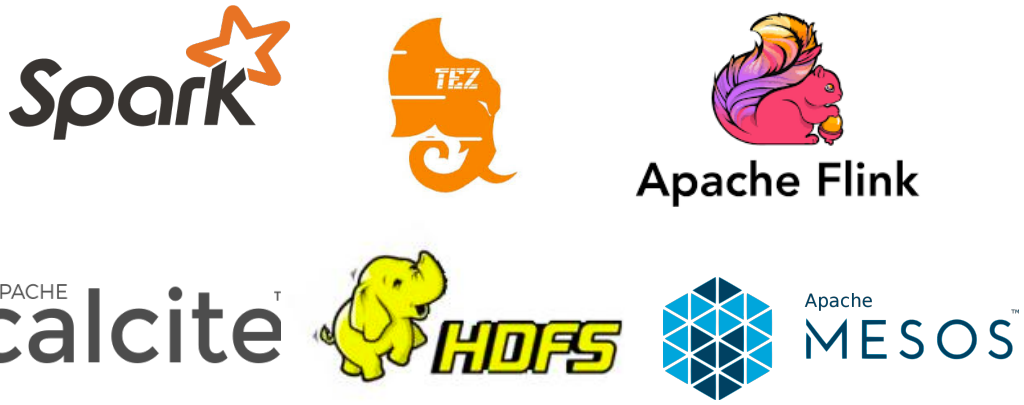
- **Distributed computation in Local-Area Networks (LAN)**
  - To accelerate executions within a single cluster



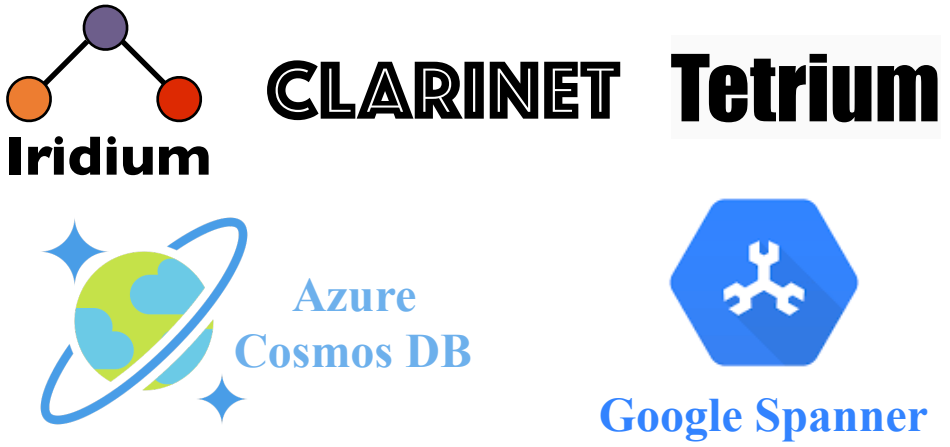
Efforts for Computation in LAN

Distributed Data Processing is Ubiquitous

- **Distributed computation in Local-Area Networks (LAN)**
  - To accelerate executions within a single cluster
- **Computation over Wide-Area Networks (WAN)**
  - To reduce data transfers, mitigate privacy risks

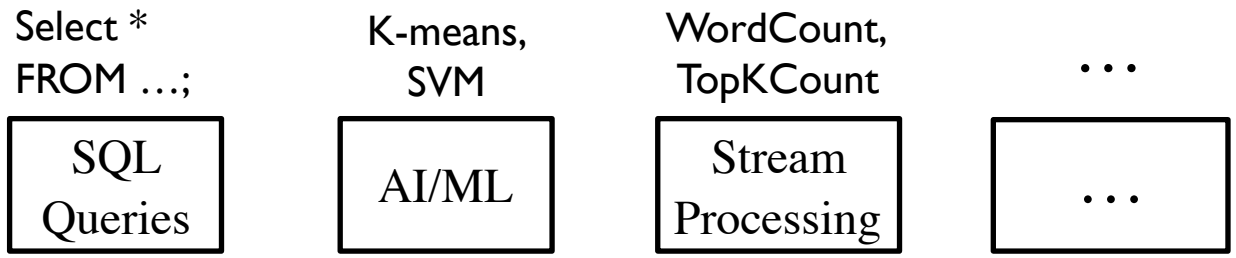


Efforts for Computation in LAN

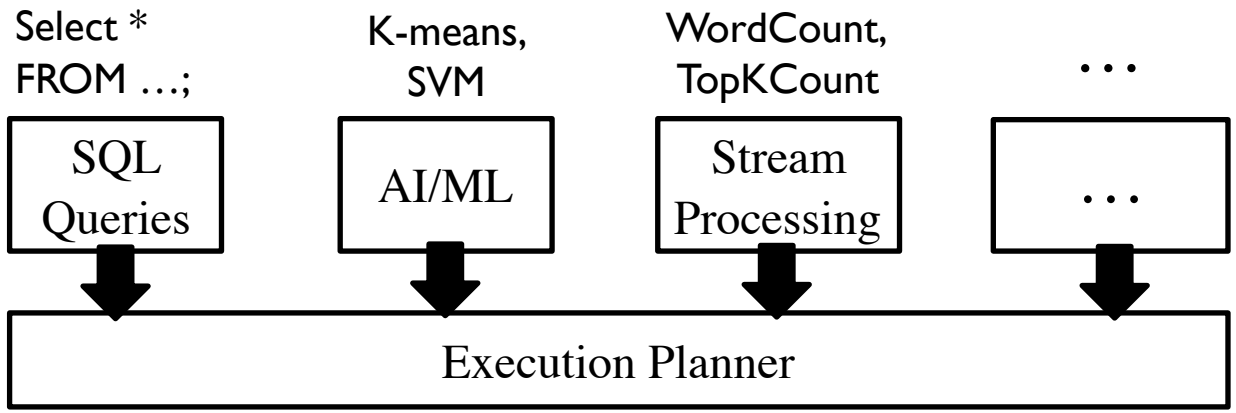


Efforts for Computation over WAN

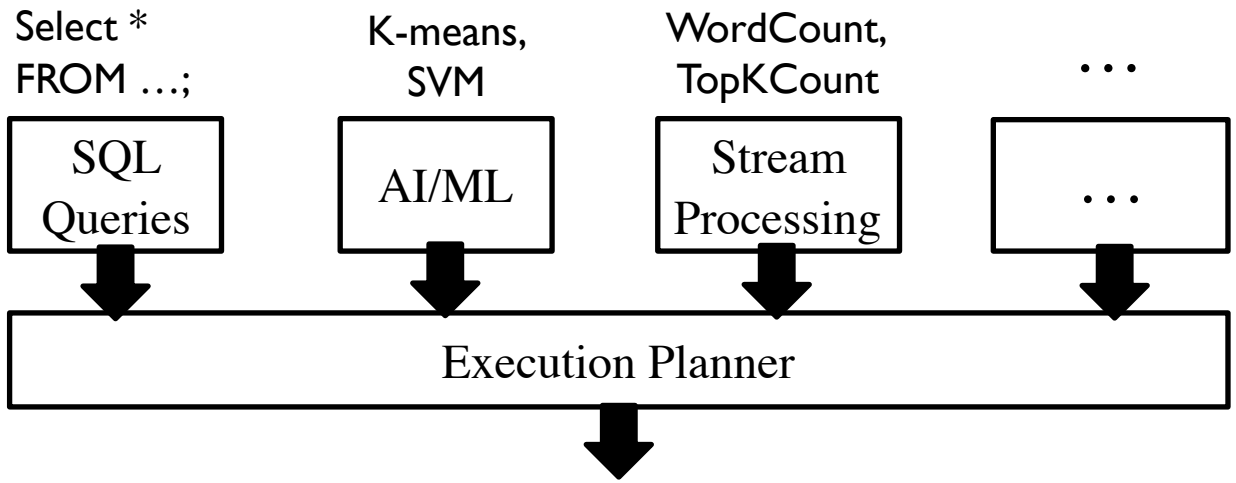
Execution Engine: Core of Big Data Stack

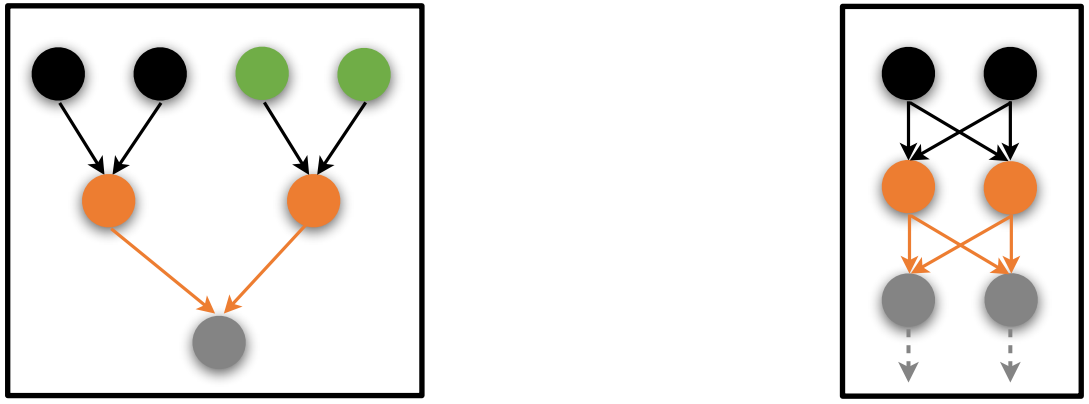


Execution Engine: Core of Big Data Stack



Execution Engine: Core of Big Data Stack



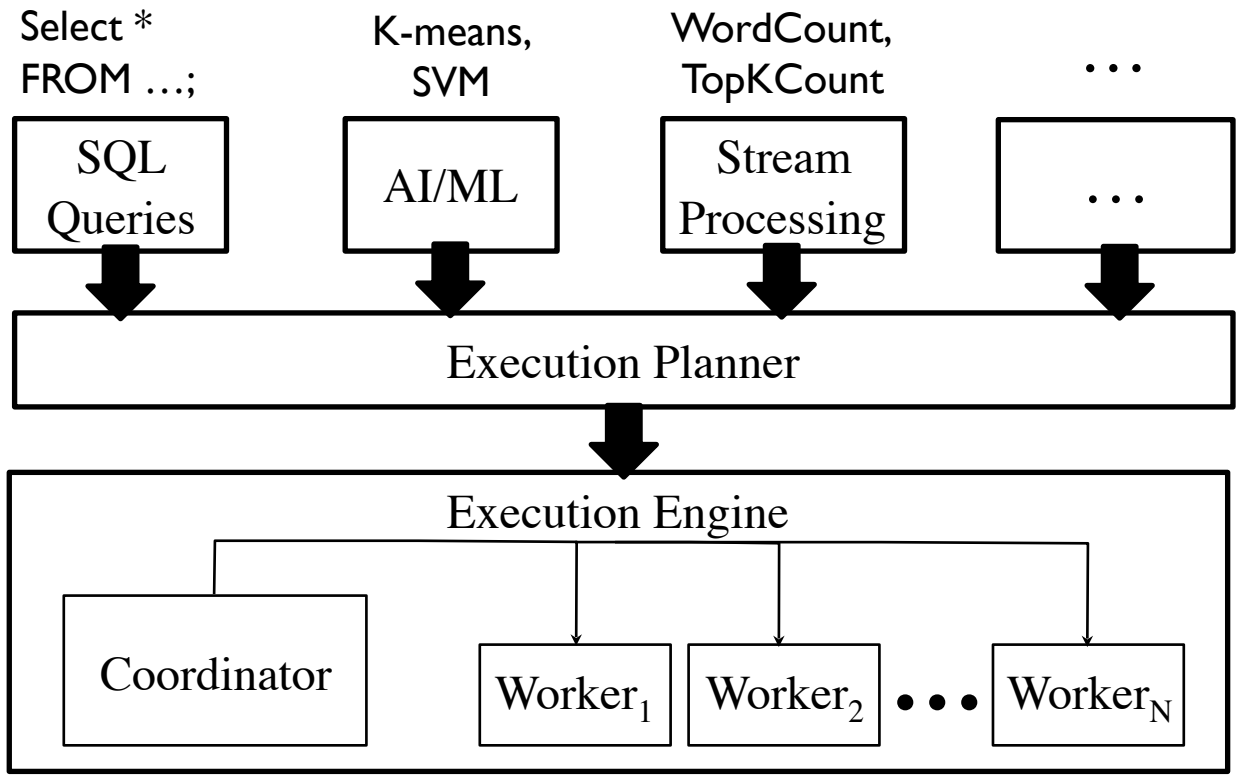


Job 1

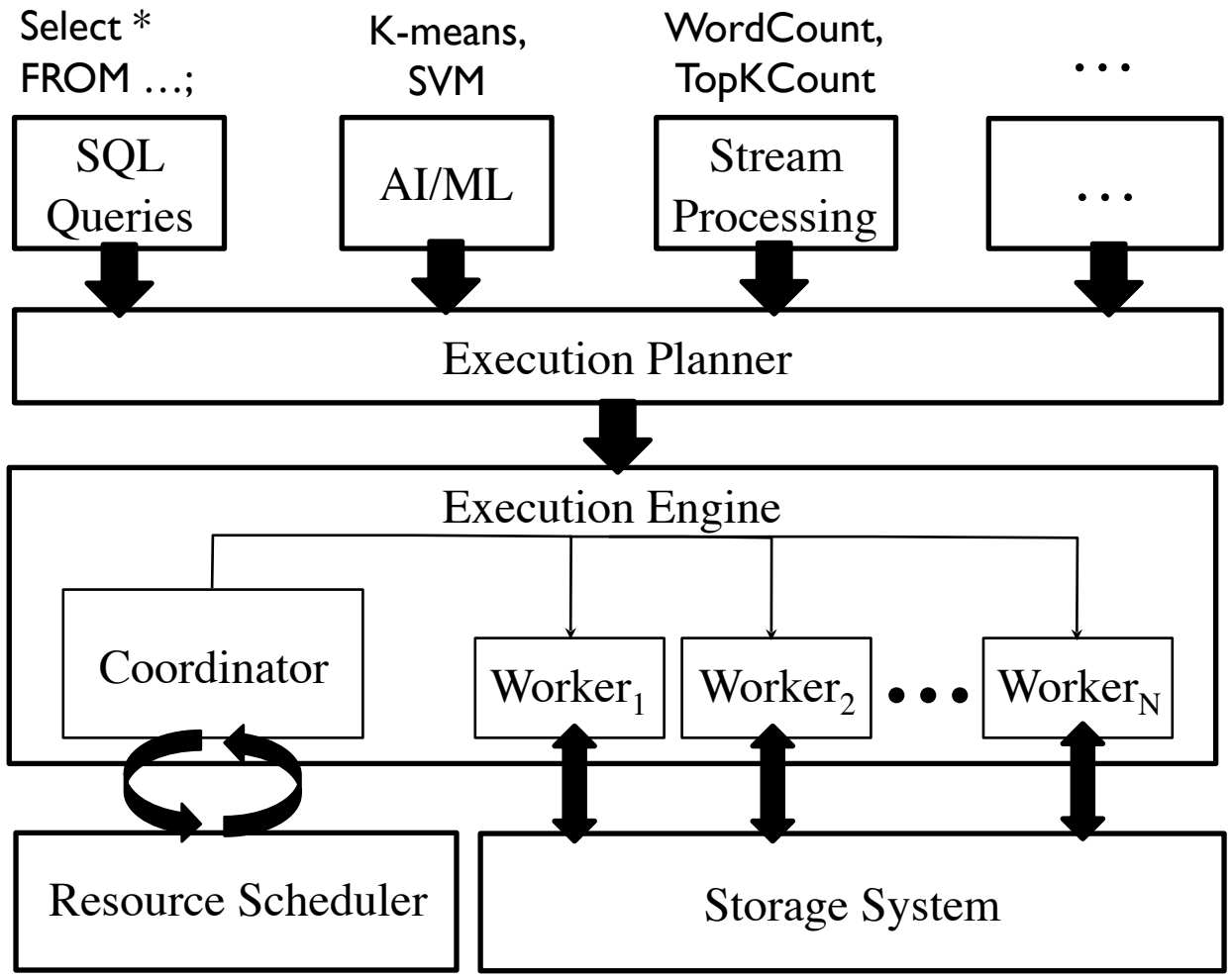
Job 2

Typical job execution plans

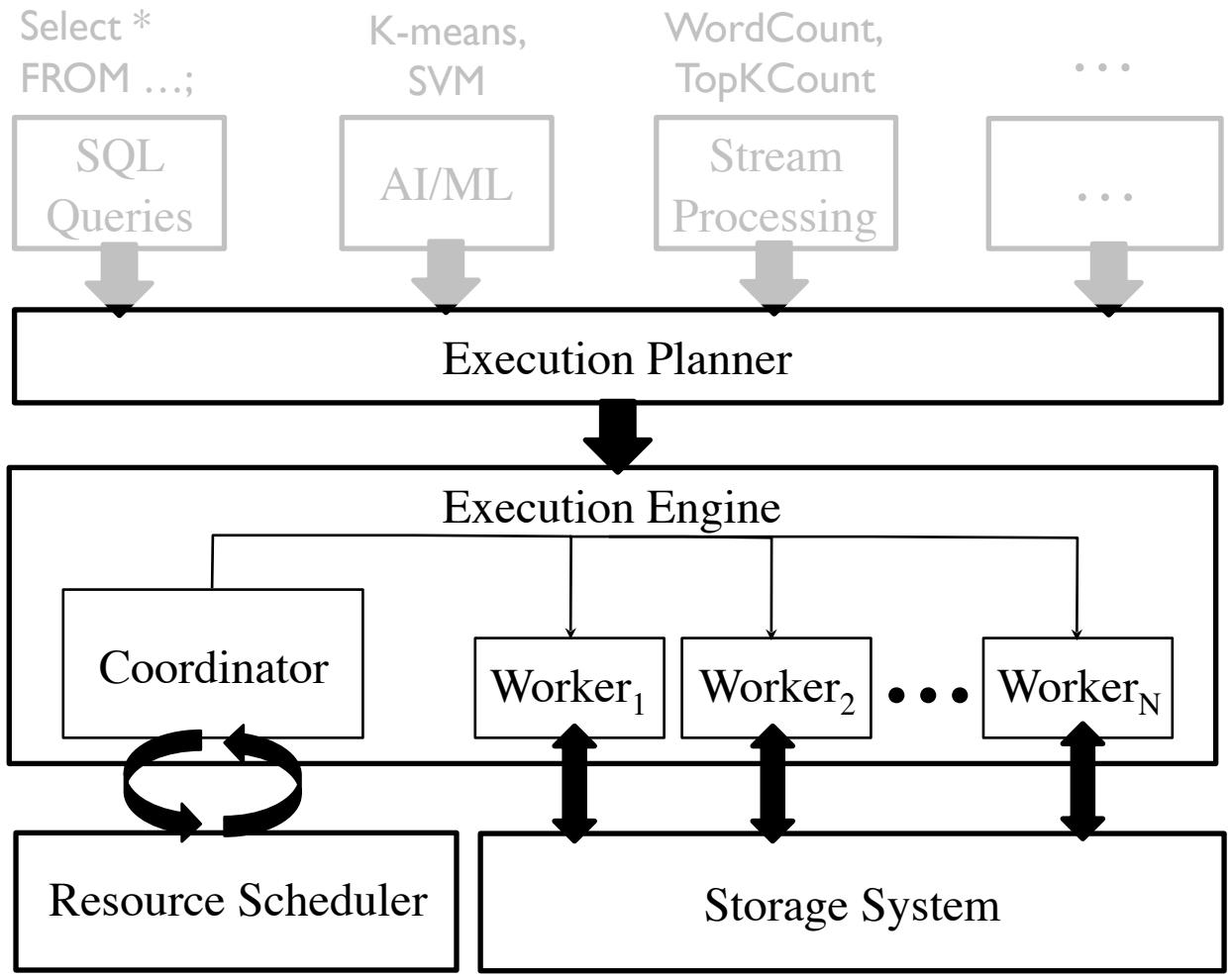
Execution Engine: Core of Big Data Stack



Execution Engine: Core of Big Data Stack



Execution Engine: Core of Big Data Stack



Spark TEZ Apache Flink

APACHE calcite HDFS Apache MESOS

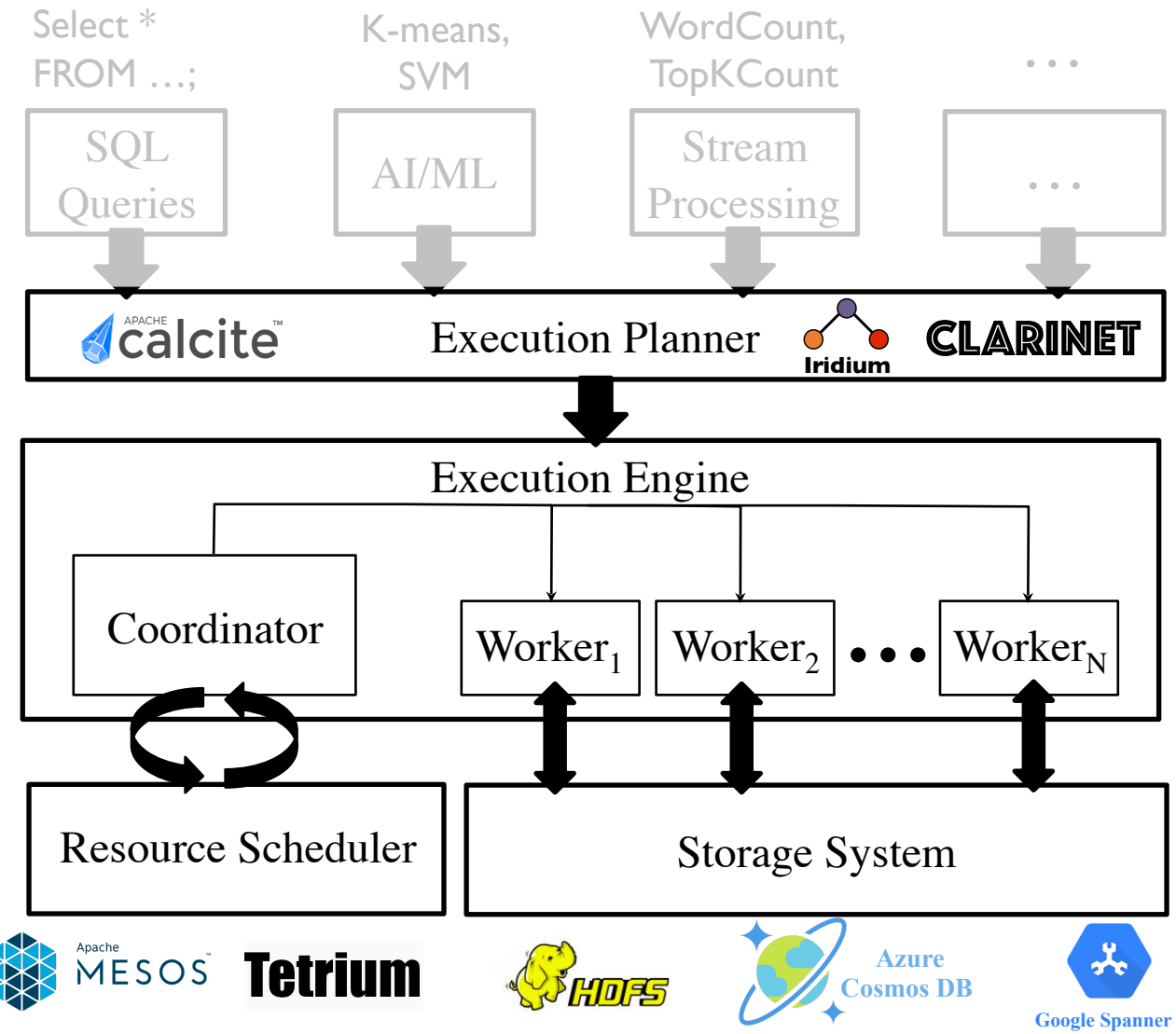
Efforts for Computation in LAN

Iridium CLARINET Tetrium

Azure Cosmos DB Google Spanner

Efforts for Computation over WAN

Execution Engine: Core of Big Data Stack



Spark

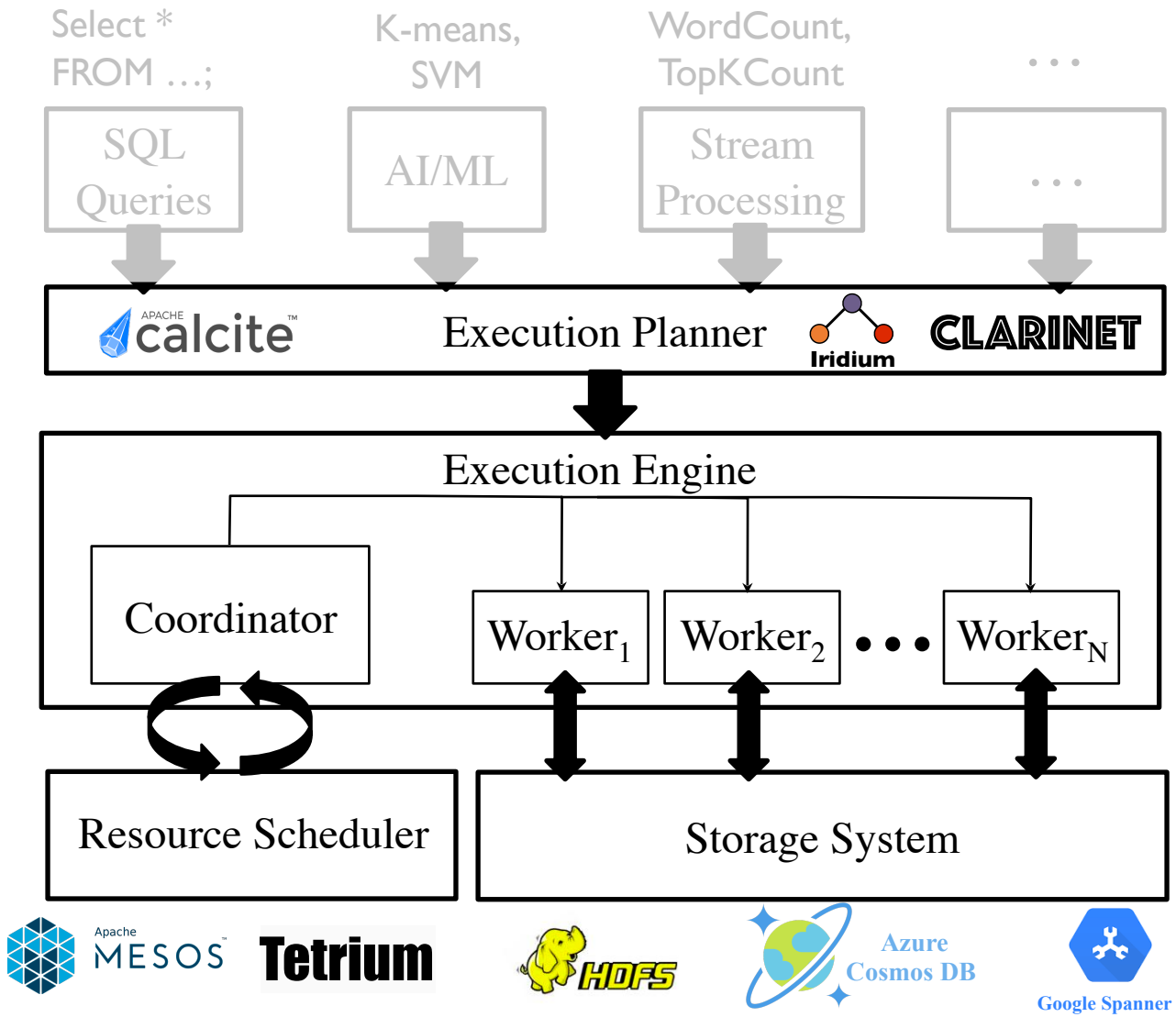
TEZ

Apache Flink

Efforts for Computation in LAN

Efforts for Computation over WAN

Execution Engine: Core of Big Data Stack

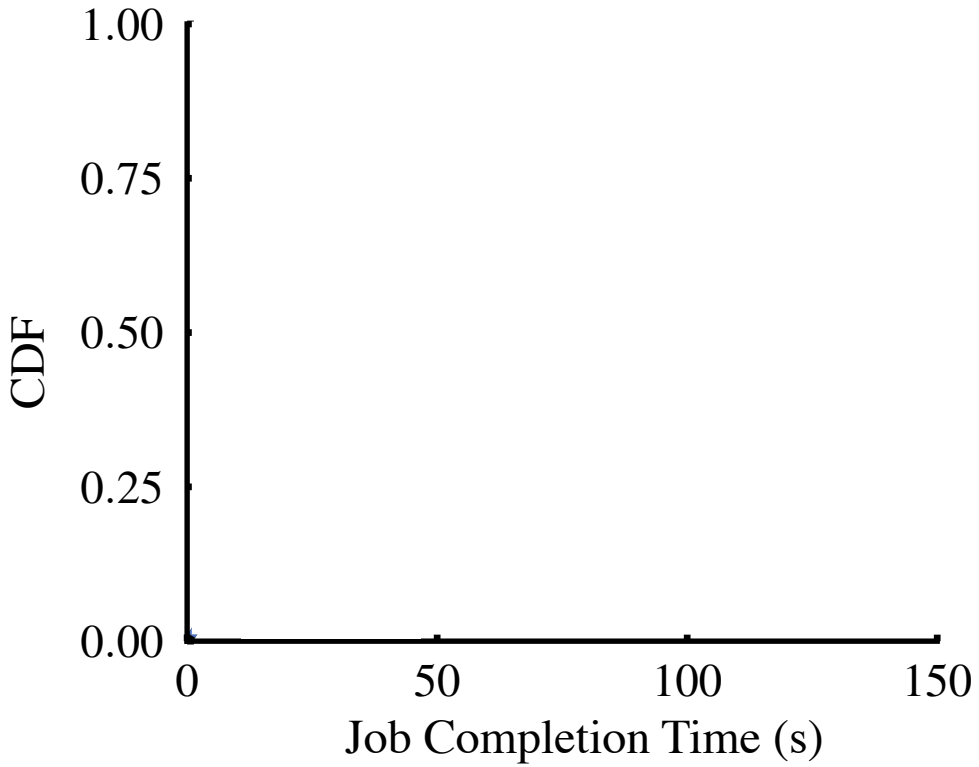


While network conditions are diverse in real, execution engines remain the same

Outline

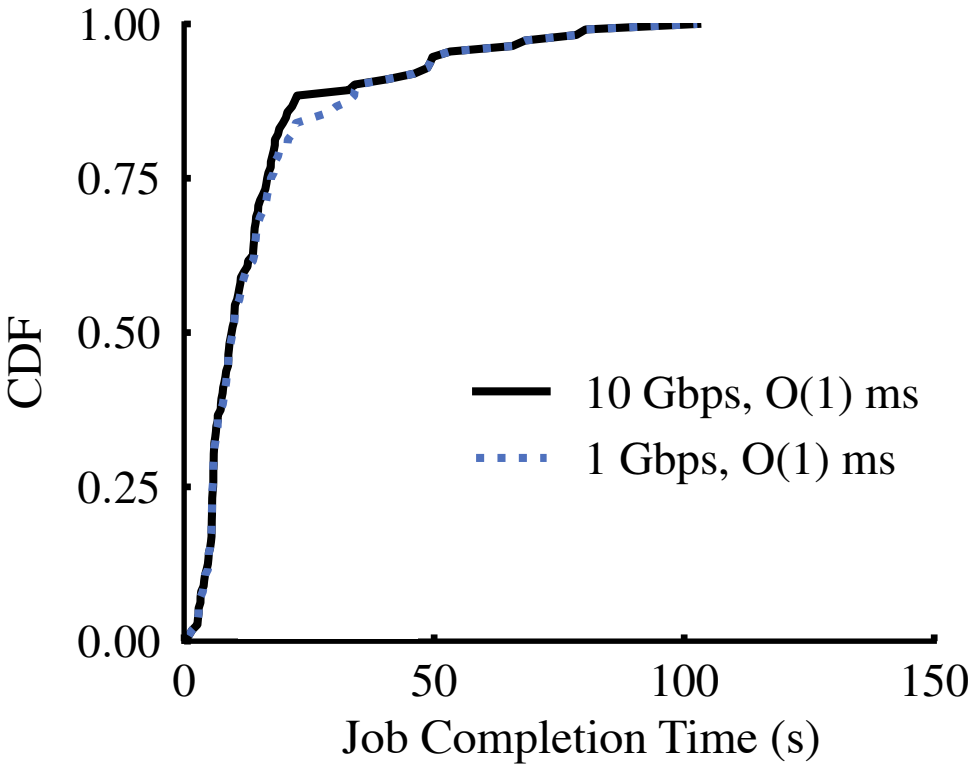
- Today's Execution Engines
- Sol Architecture
- Control Plane Design
- Data Plane Design
- Evaluation

Impact of Networks on Latency-sensitive Jobs



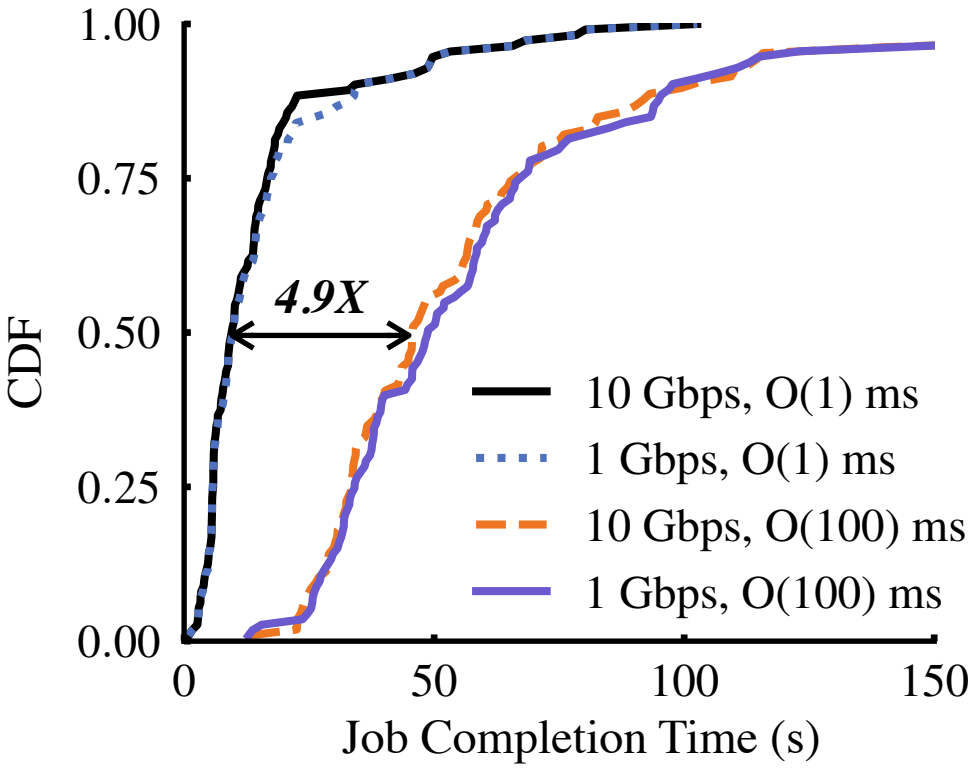
Queries from 100 GB TPC Benchmarks

Impact of Networks on Latency-sensitive Jobs



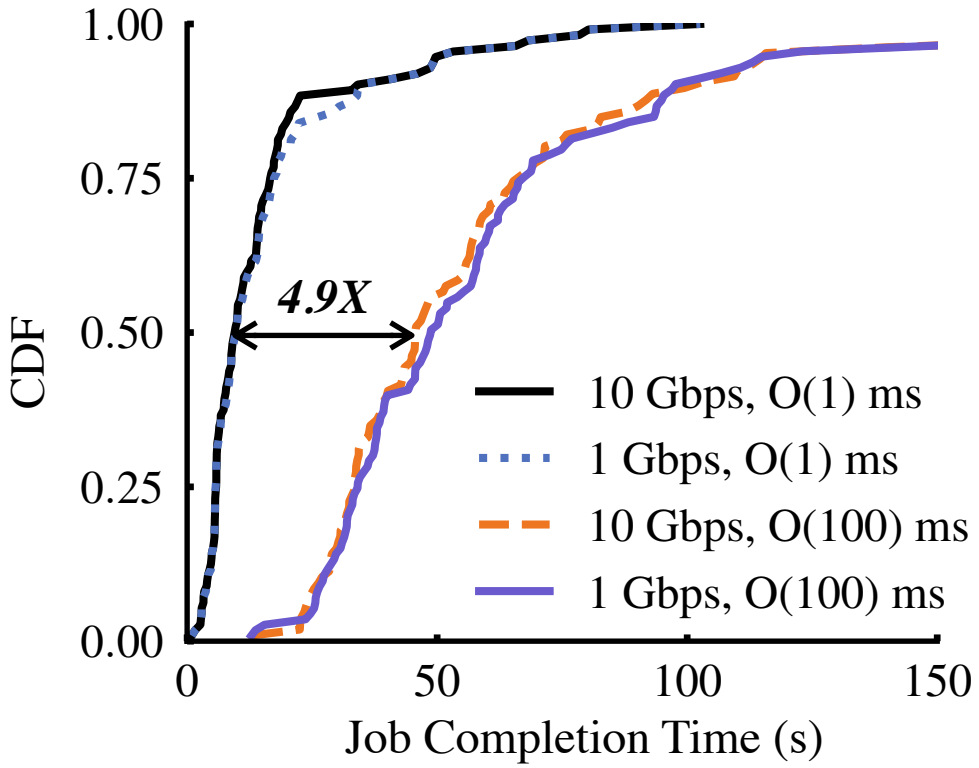
Queries from 100 GB TPC Benchmarks

Impact of Networks on Latency-sensitive Jobs



Queries from 100 GB TPC Benchmarks

Impact of Networks on Latency-sensitive Jobs

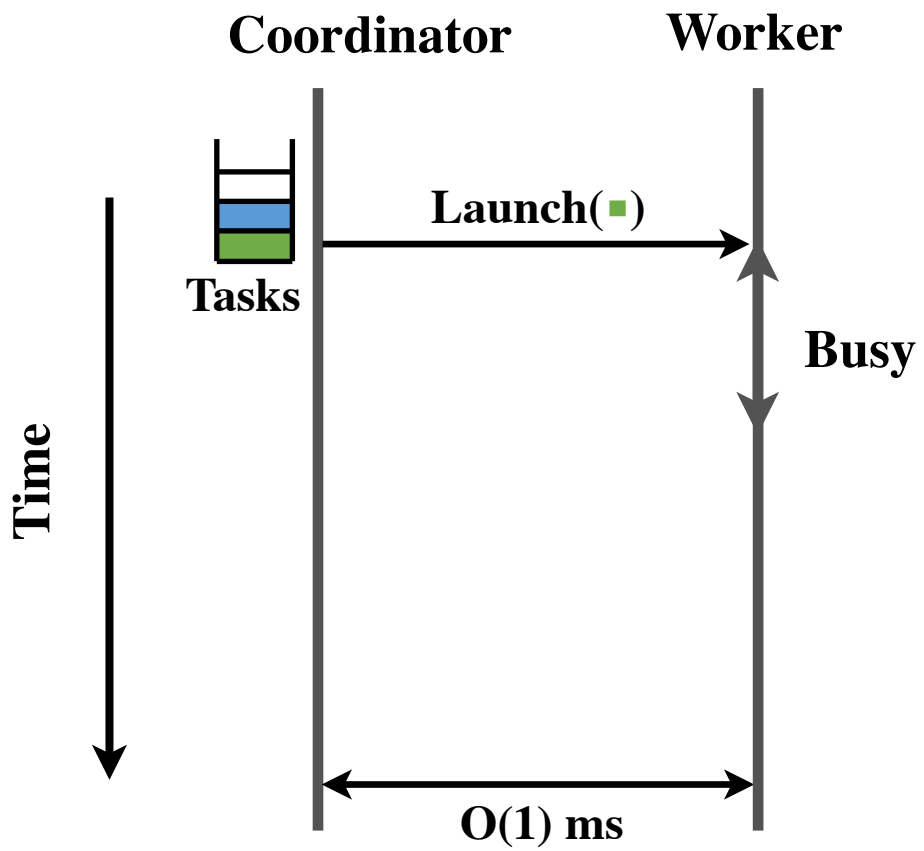


Queries from 100 GB TPC Benchmarks

Problem #1

Slow job execution in *high-latency networks*

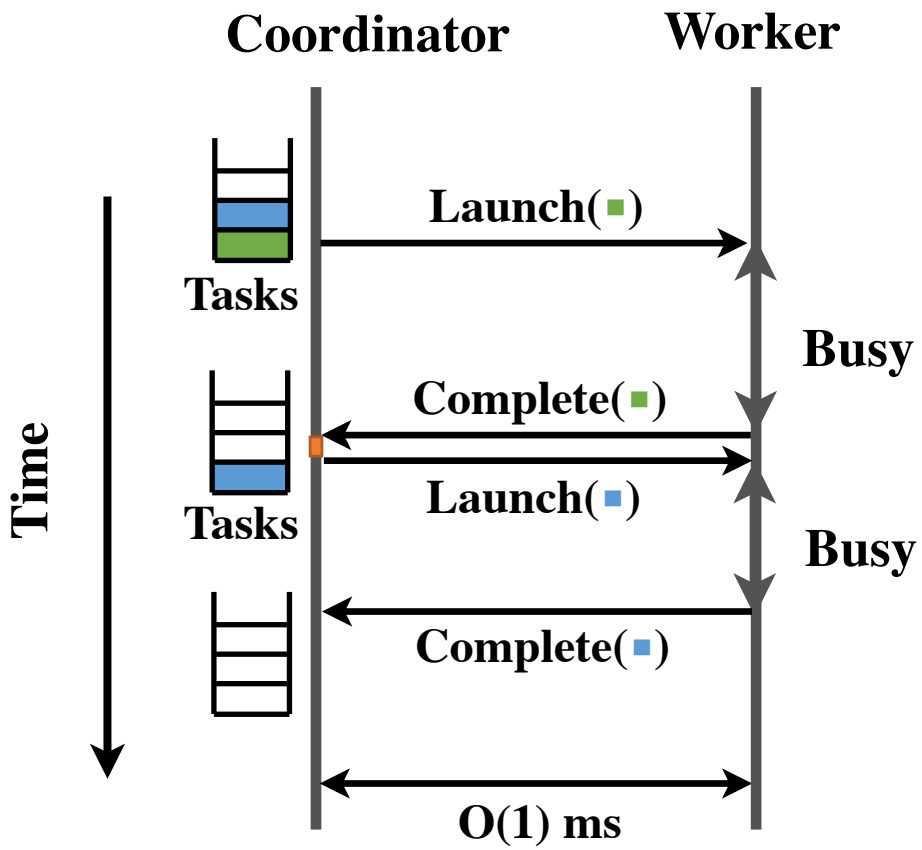
Control Plane Inefficiency Due to High Latency



Problem #1

Slow job execution in *high-latency networks*

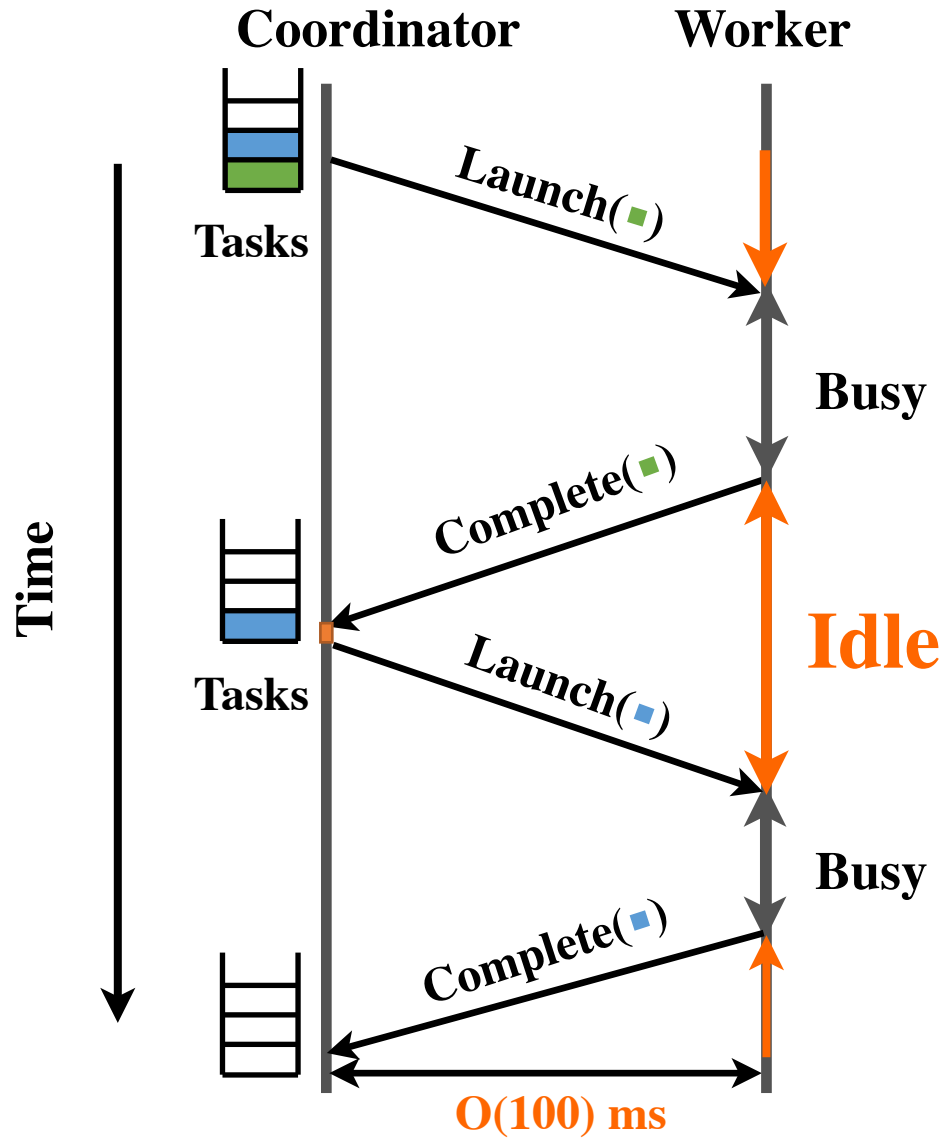
Control Plane Inefficiency Due to High Latency



Problem #1

Slow job execution in *high-latency networks*

Control Plane Inefficiency Due to High Latency



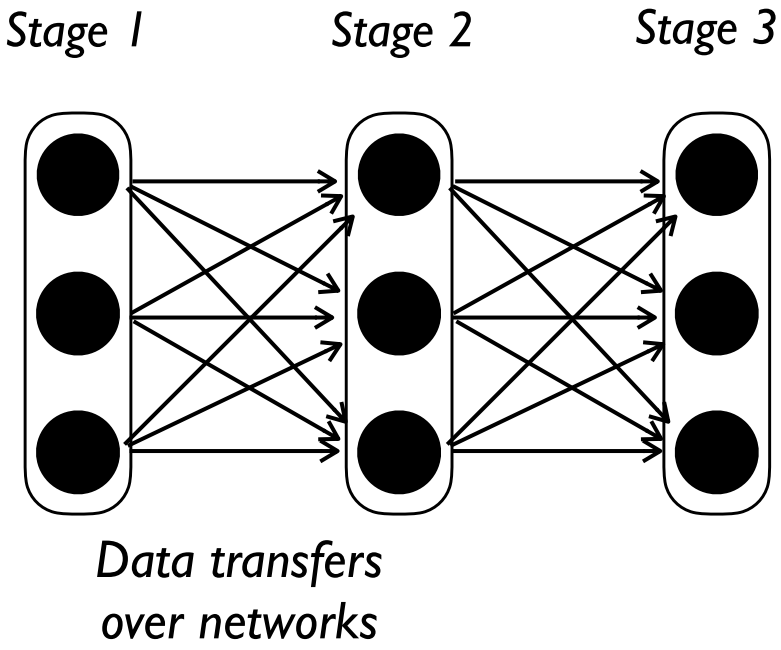
Late-binding of tasks postpones scheduling

↓

Problem #1

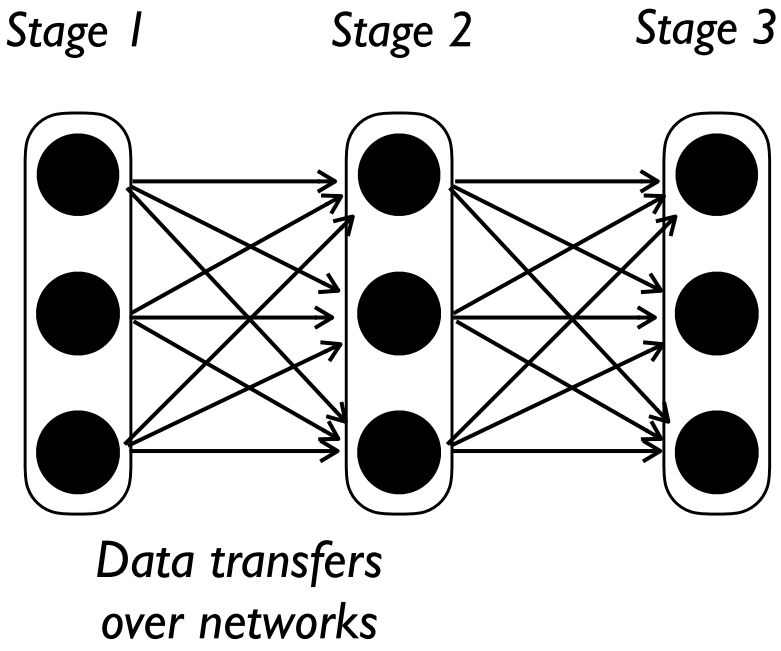
Slow job execution in *high-latency networks*

Impact of Networks on Bandwidth-intensive Jobs

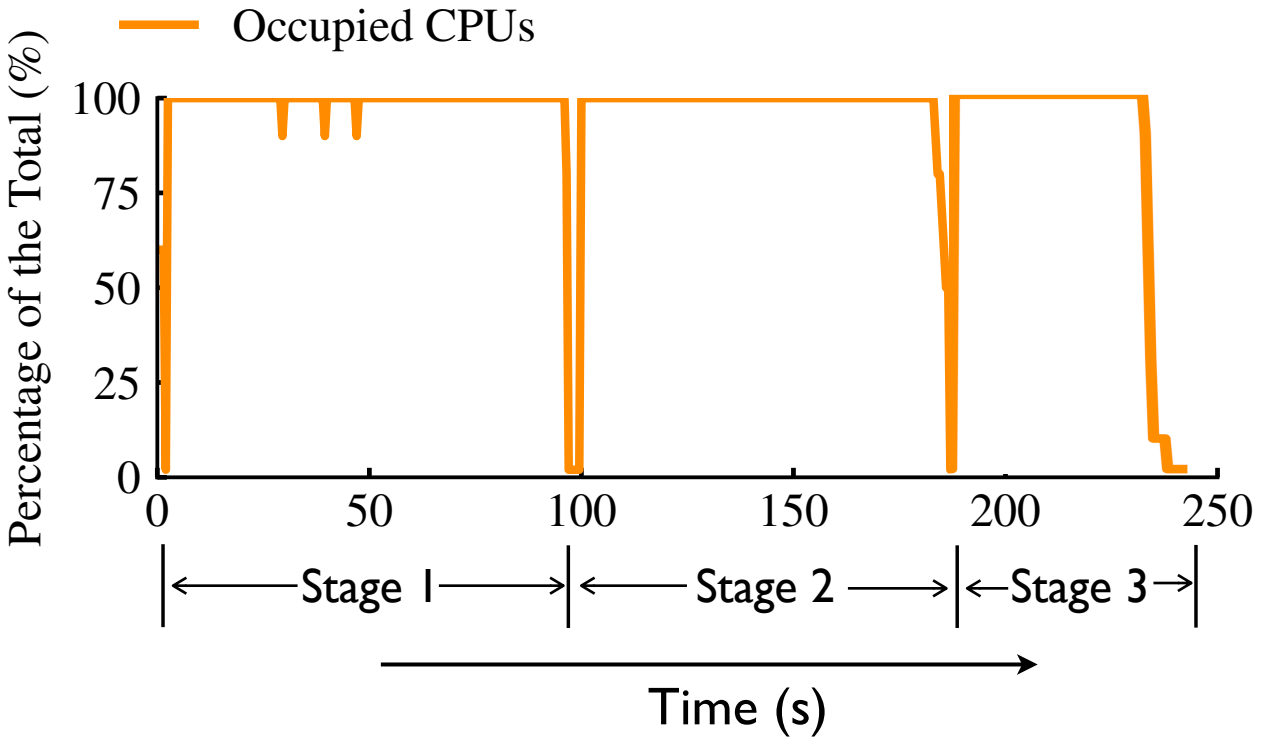


Query25 on 1TB TPC benchmark

Impact of Networks on Bandwidth-intensive Jobs

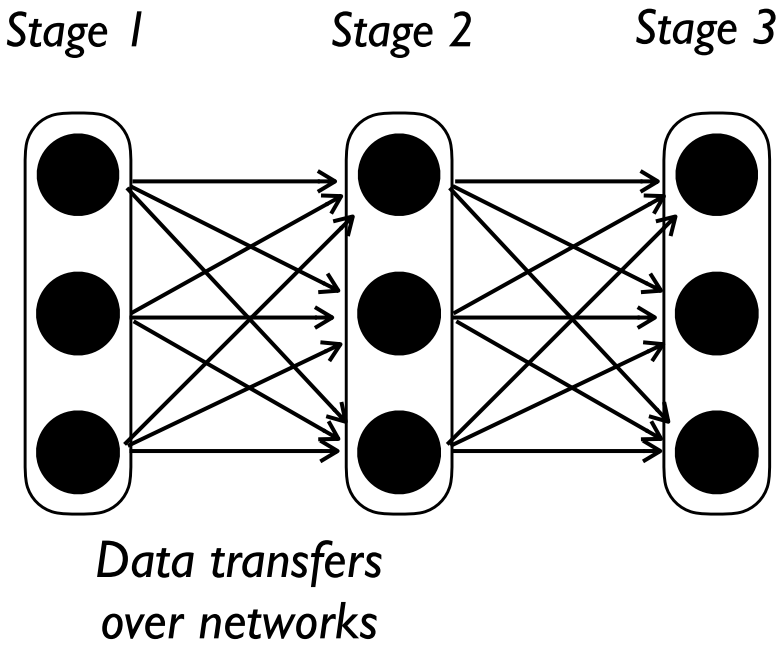


Query25 on 1TB TPC benchmark

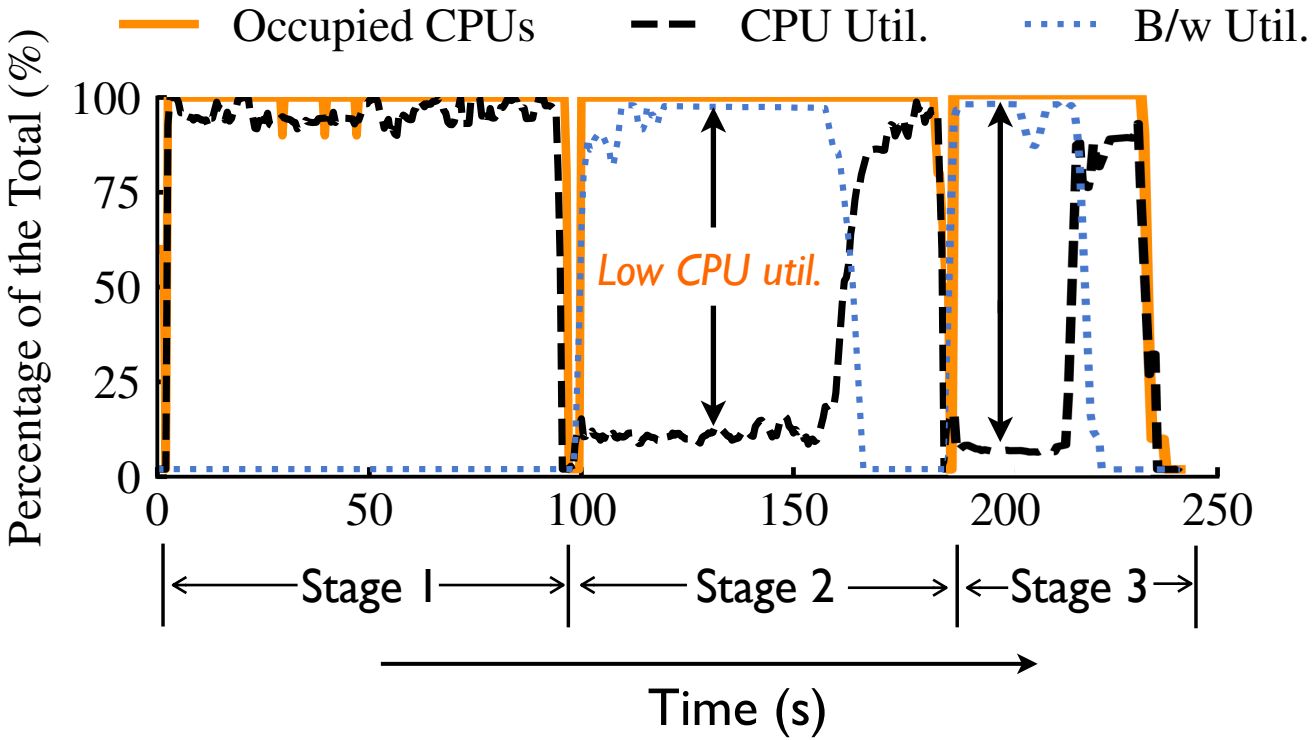


Resource utilization throughout the job

Impact of Networks on Bandwidth-intensive Jobs

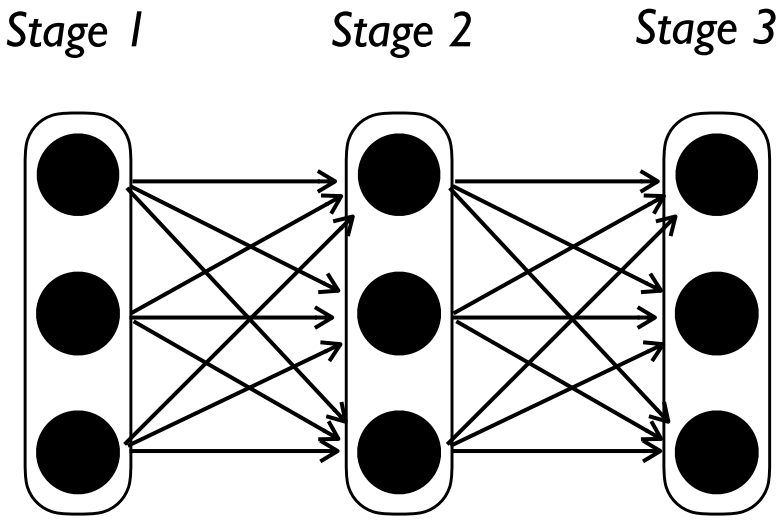


Query25 on 1TB TPC benchmark



Resource utilization throughout the job

Data Plane Inefficiency Due to Low Bandwidth



Data transfers over networks

Query25 on 1TB TPC benchmark

Tasks hog CPUs throughout the lifespan

↓

Problem #2

CPU *underutilization* in *low-bandwidth networks*

Outline

- Today's Execution Engines
- Sol Architecture
- Control Plane Design
- Data Plane Design
- Evaluation

Problem #1

High latency → *Idleness of workers*

Problem #2

Low b/w → *CPU underutilization*

Outline

- Today's Execution Engines
- Sol Architecture
- Control Plane Design
- Data Plane Design
- Evaluation

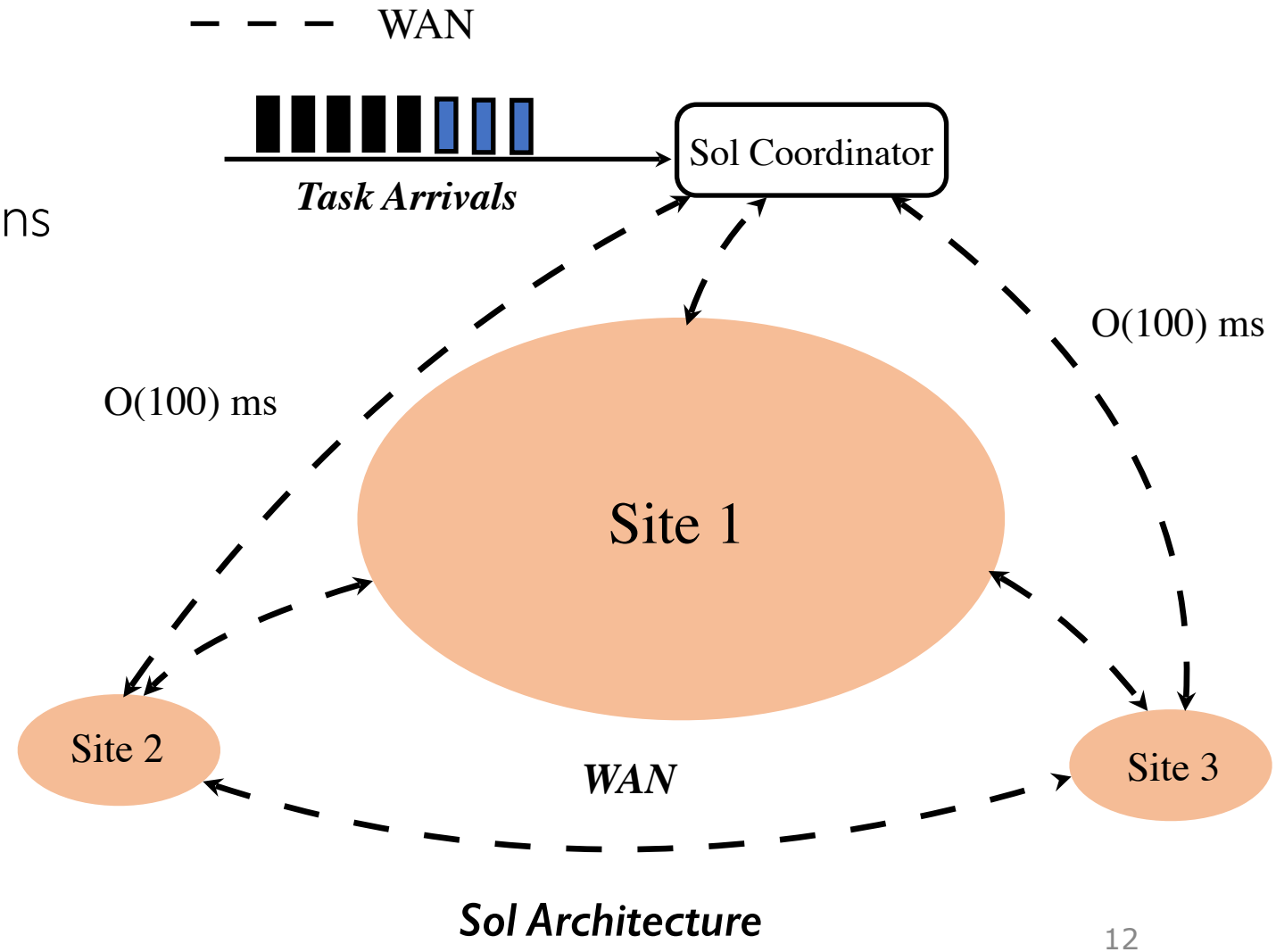
Sol

A *federated* execution engine for *diverse* network conditions w/

- faster job execution
- higher resource utilization

Sol: A Federated Execution Engine

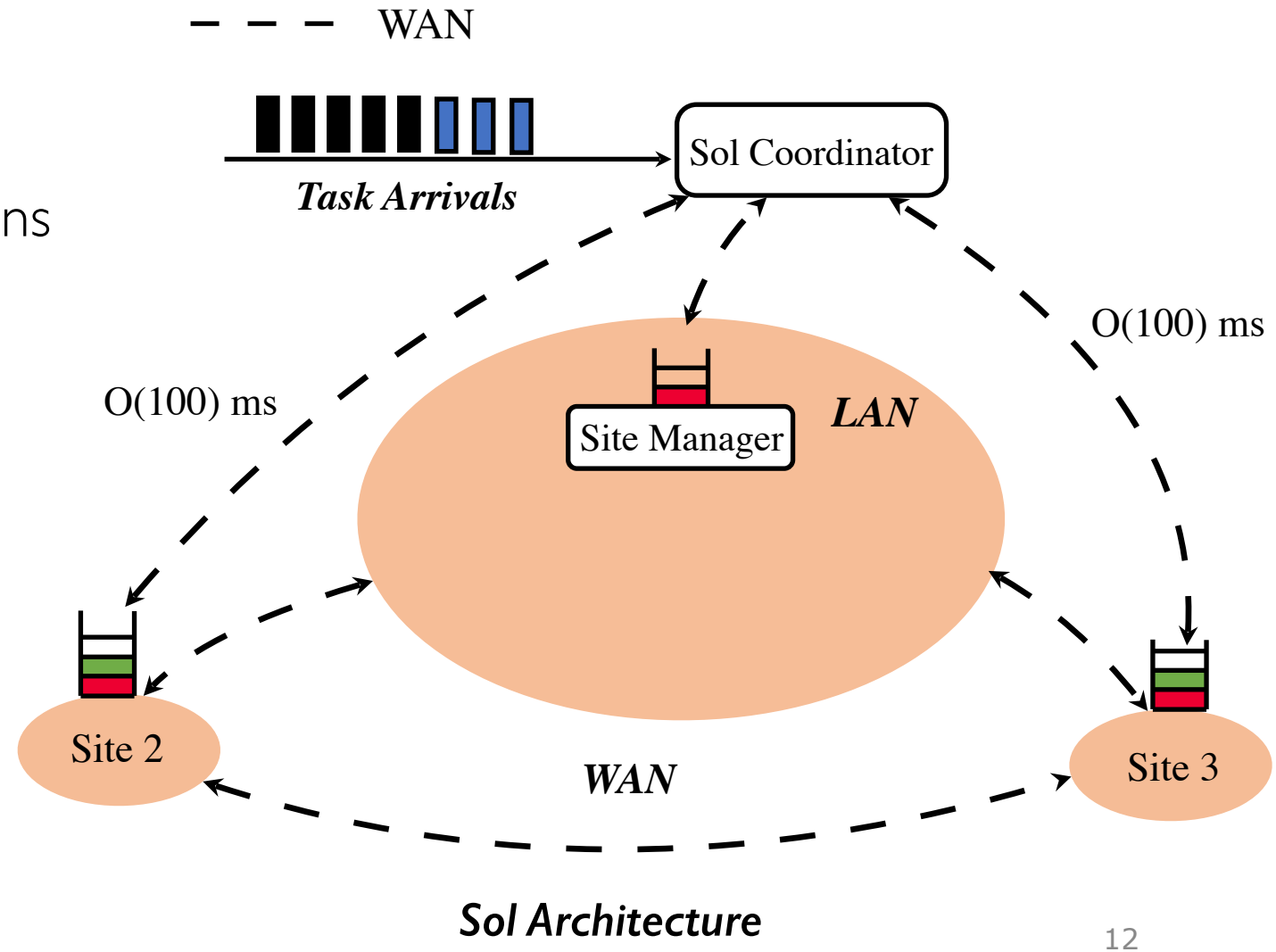
- **Central Coordinator**
  - Coordinate inter-site executions



Sol Architecture

Sol: A Federated Execution Engine

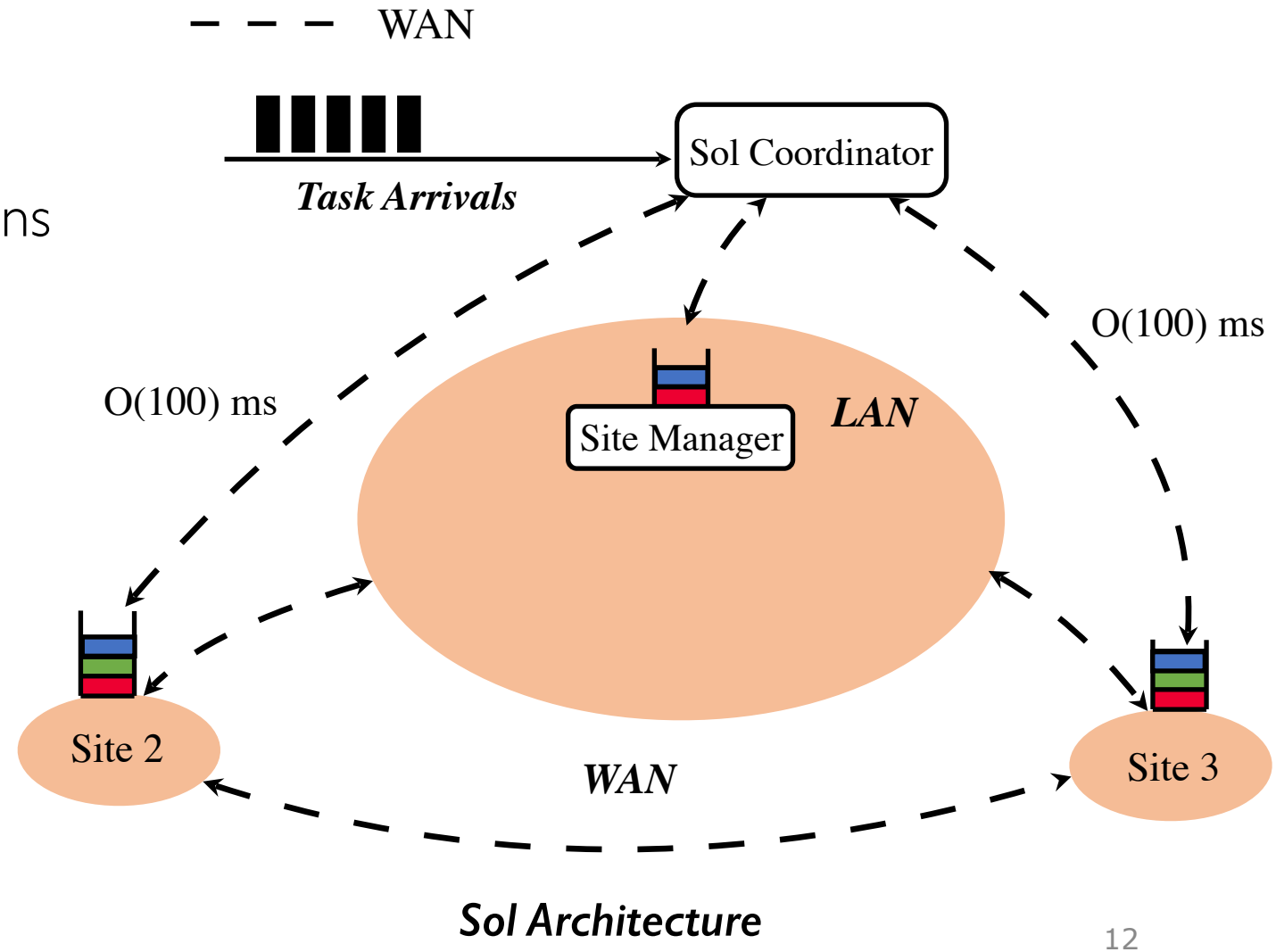
- **Central Coordinator**
  - Coordinate inter-site executions
- **Site Manager**
  - Coordinate local workers
  - Manage queued tasks



Sol Architecture

Sol: A Federated Execution Engine

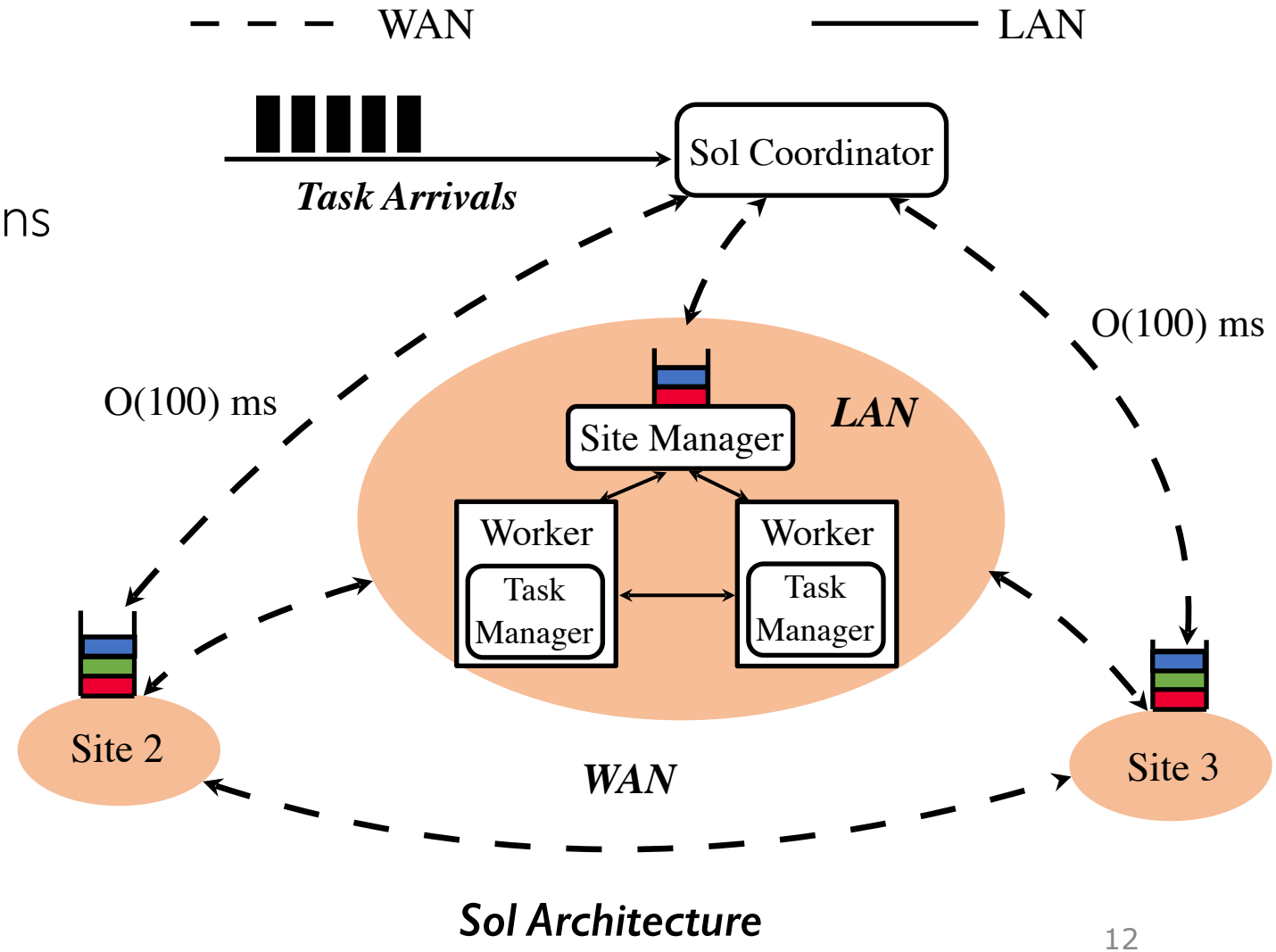
- **Central Coordinator**
  - Coordinate inter-site executions
- **Site Manager**
  - Coordinate local workers
  - Manage queued tasks



Sol Architecture

Sol: A Federated Execution Engine

- **Central Coordinator**
  - Coordinate inter-site executions
- **Site Manager**
  - Coordinate local workers
  - Manage queued tasks
- **Task Manager**
  - Manage worker resource



Sol Architecture

Outline

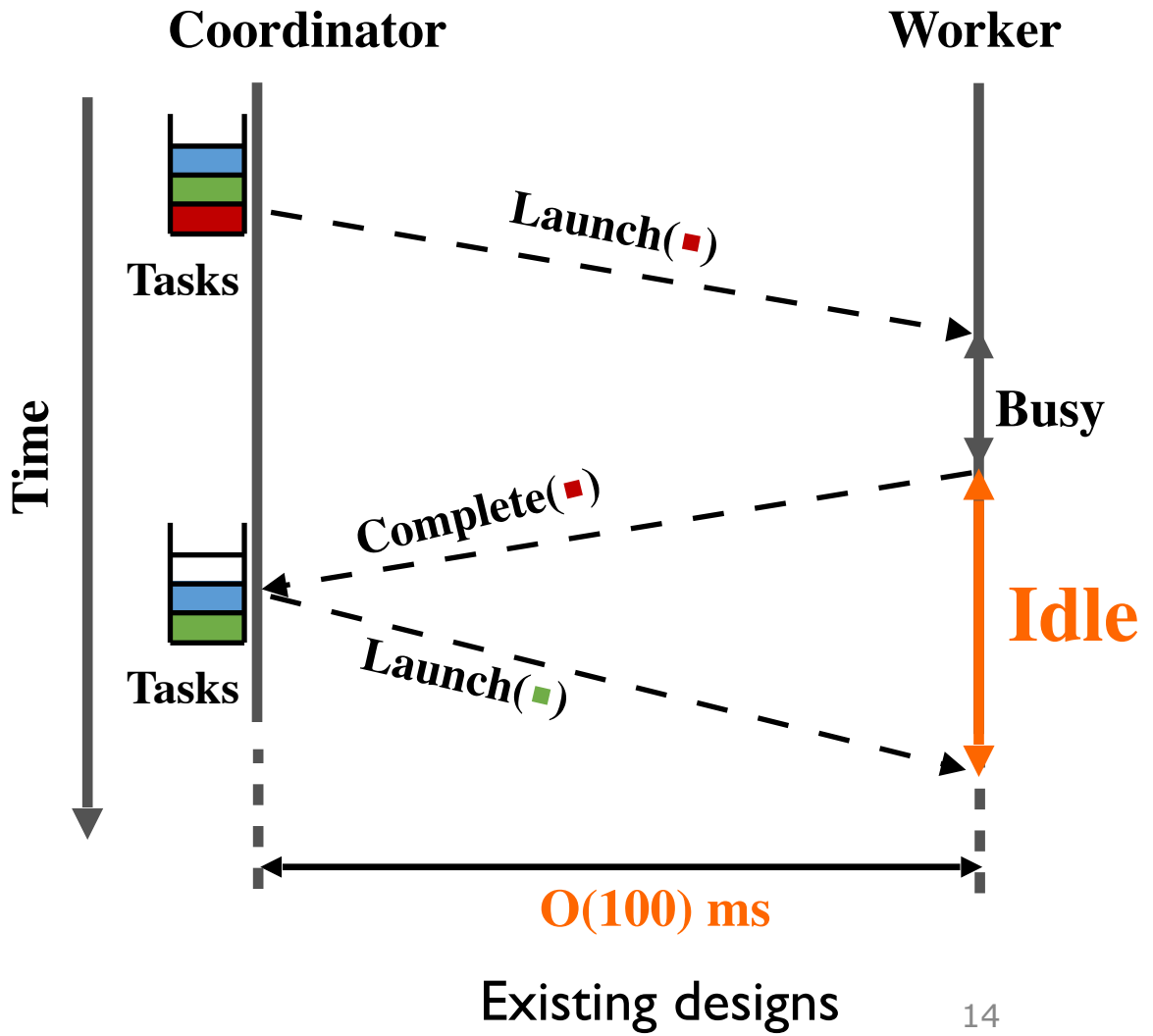
- Today's Execution Engines
- Sol Architecture
- Control Plane Design
- Data Plane Design
- Evaluation

Problem #1

High latency → Idleness of workers

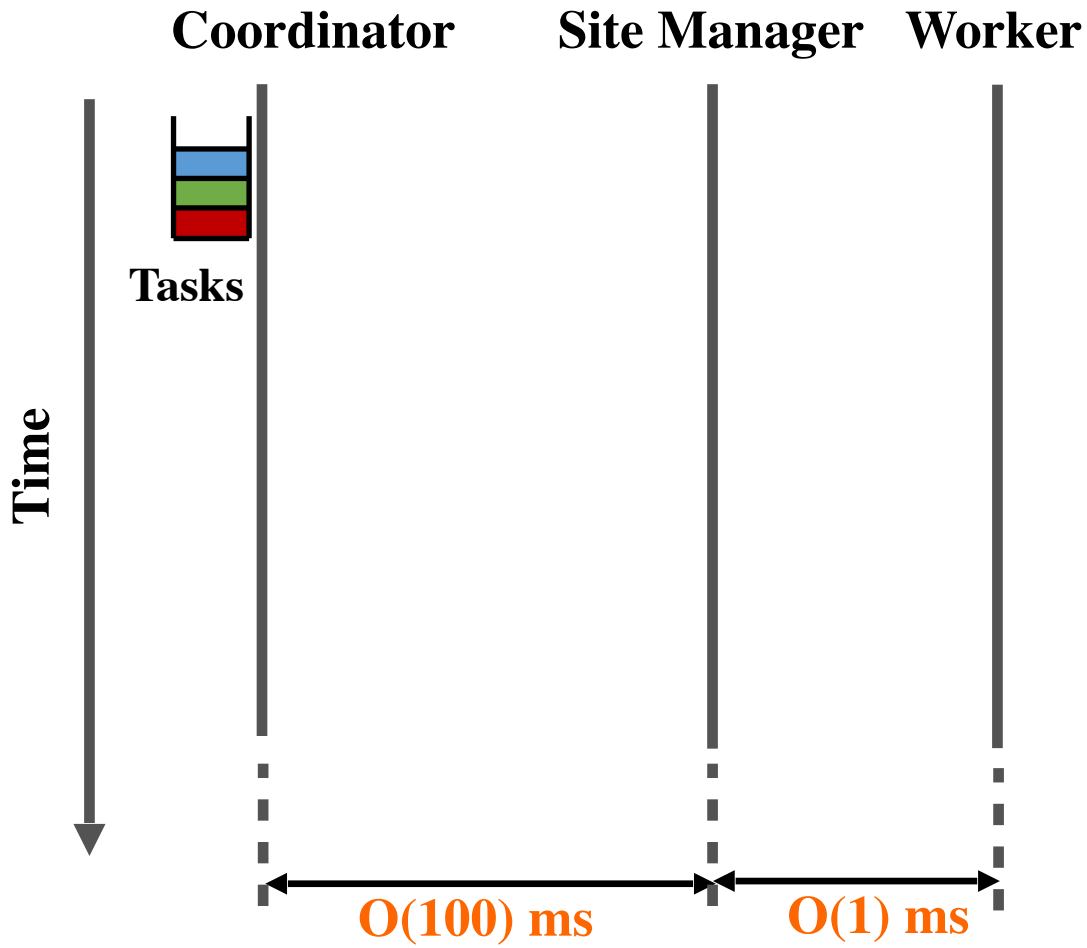
Push tasks proactively to reduce worker idle time

Task Early-binding in Control Plane

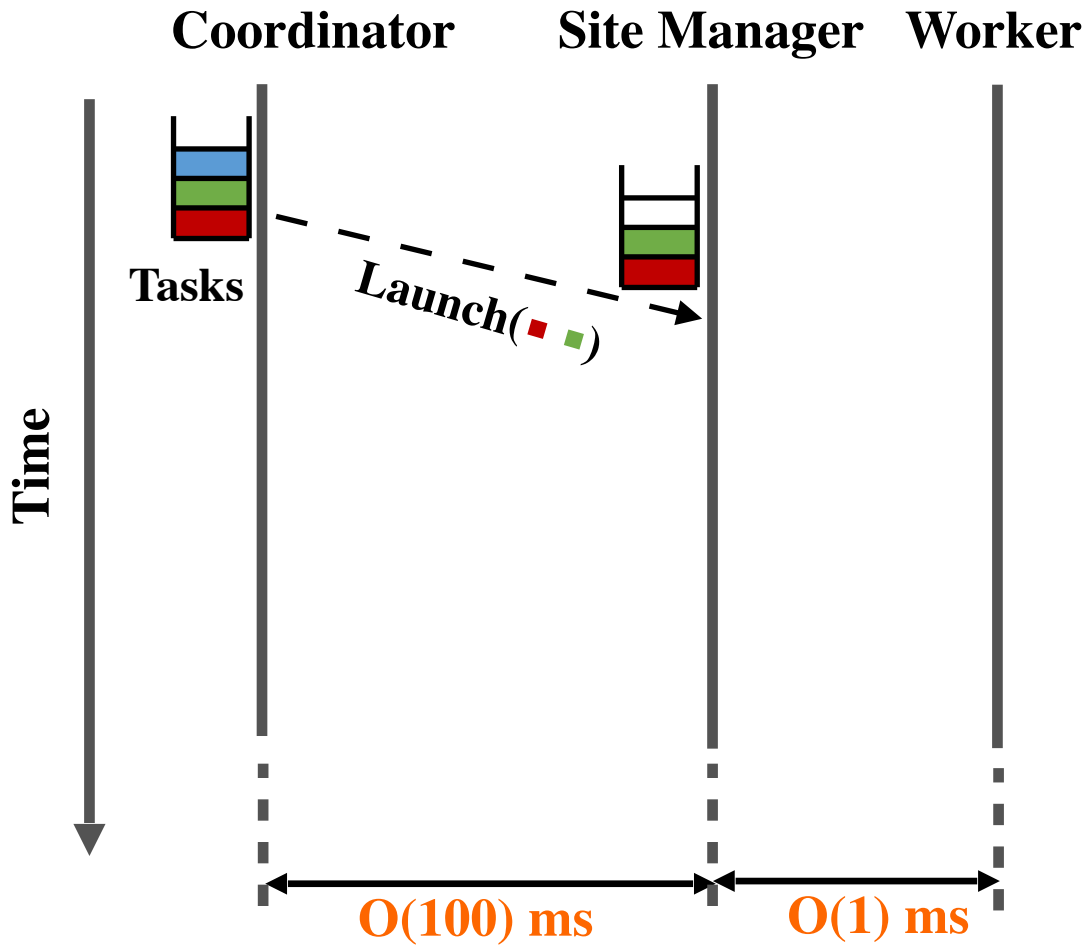


Existing designs

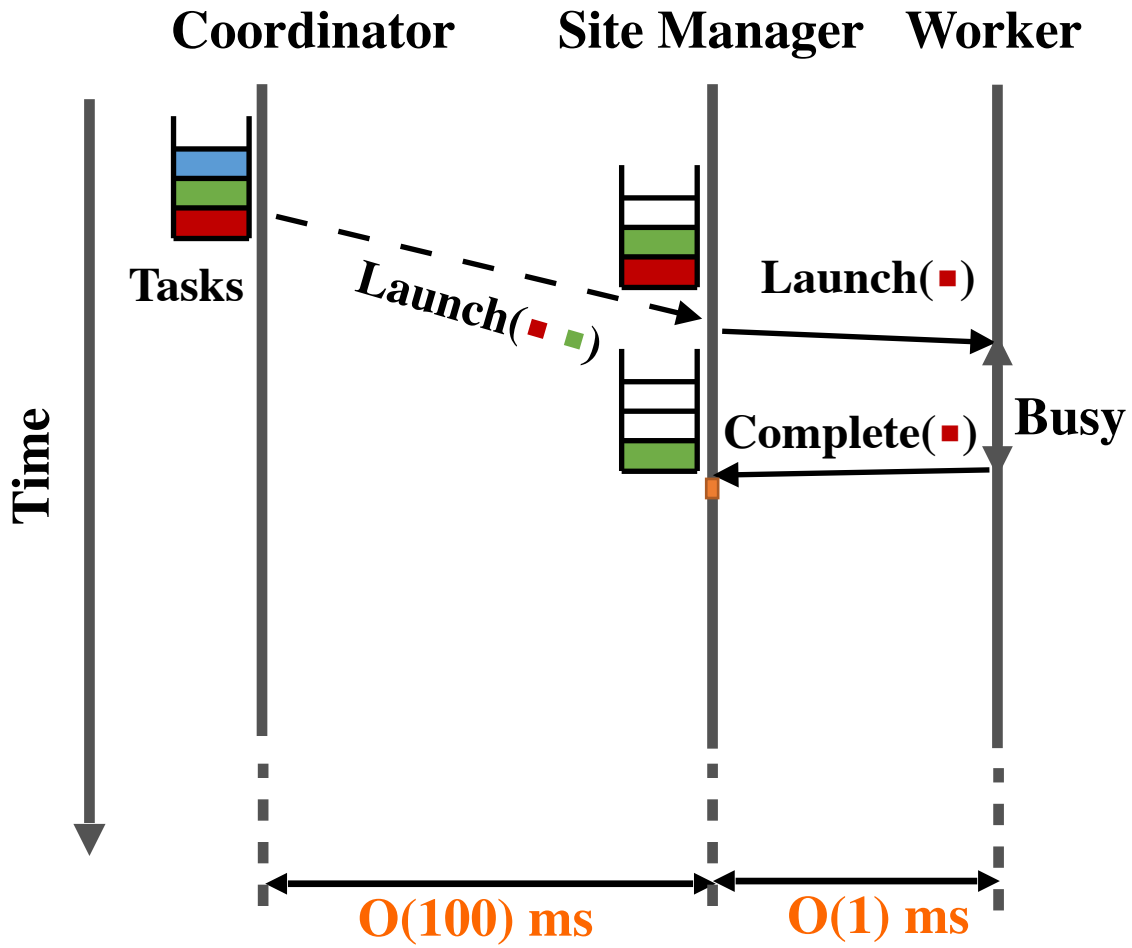
Task Early-binding in Control Plane



Task Early-binding in Control Plane



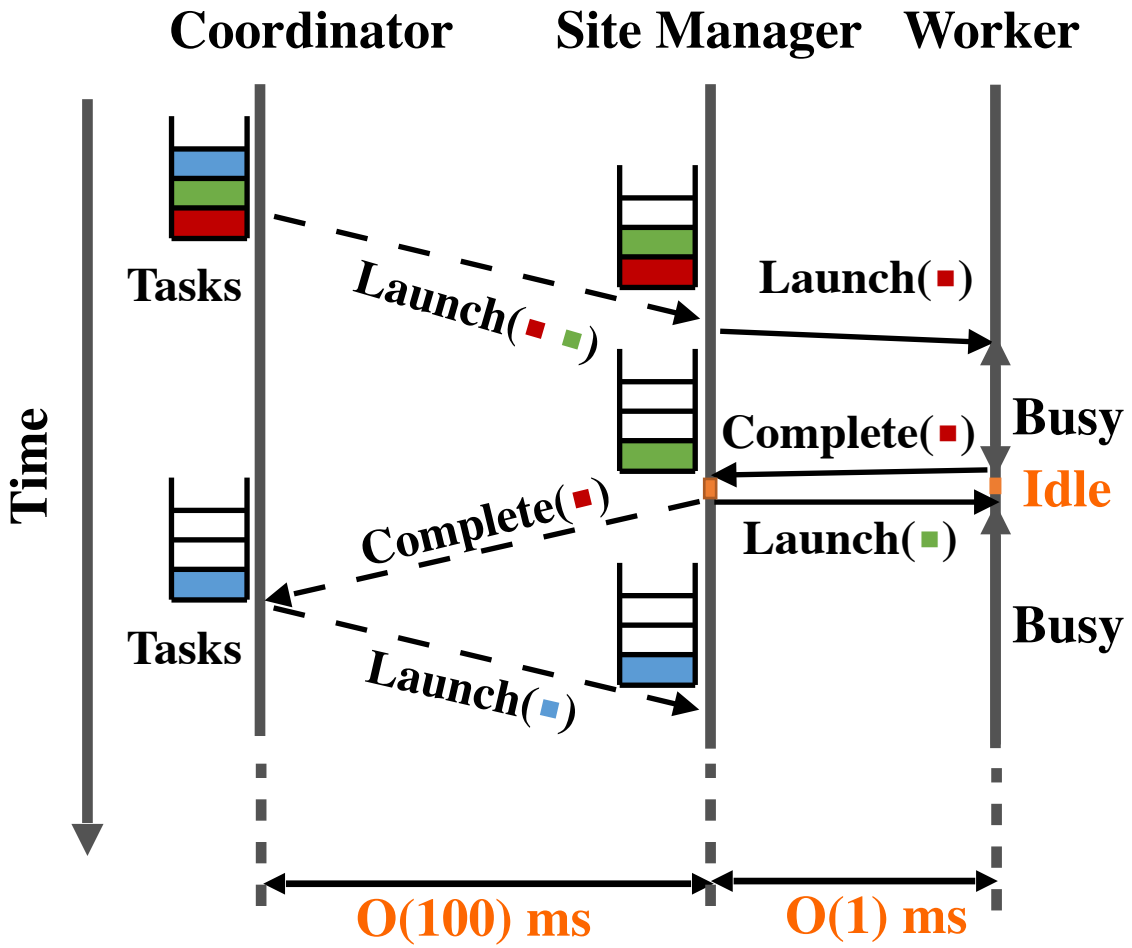
Task Early-binding in Control Plane



Task Early-binding in Control Plane

- **Coordinator ⟷ Site Manager**
  - Inter-site operations are *early-binding*

  → Guarantee high utilization



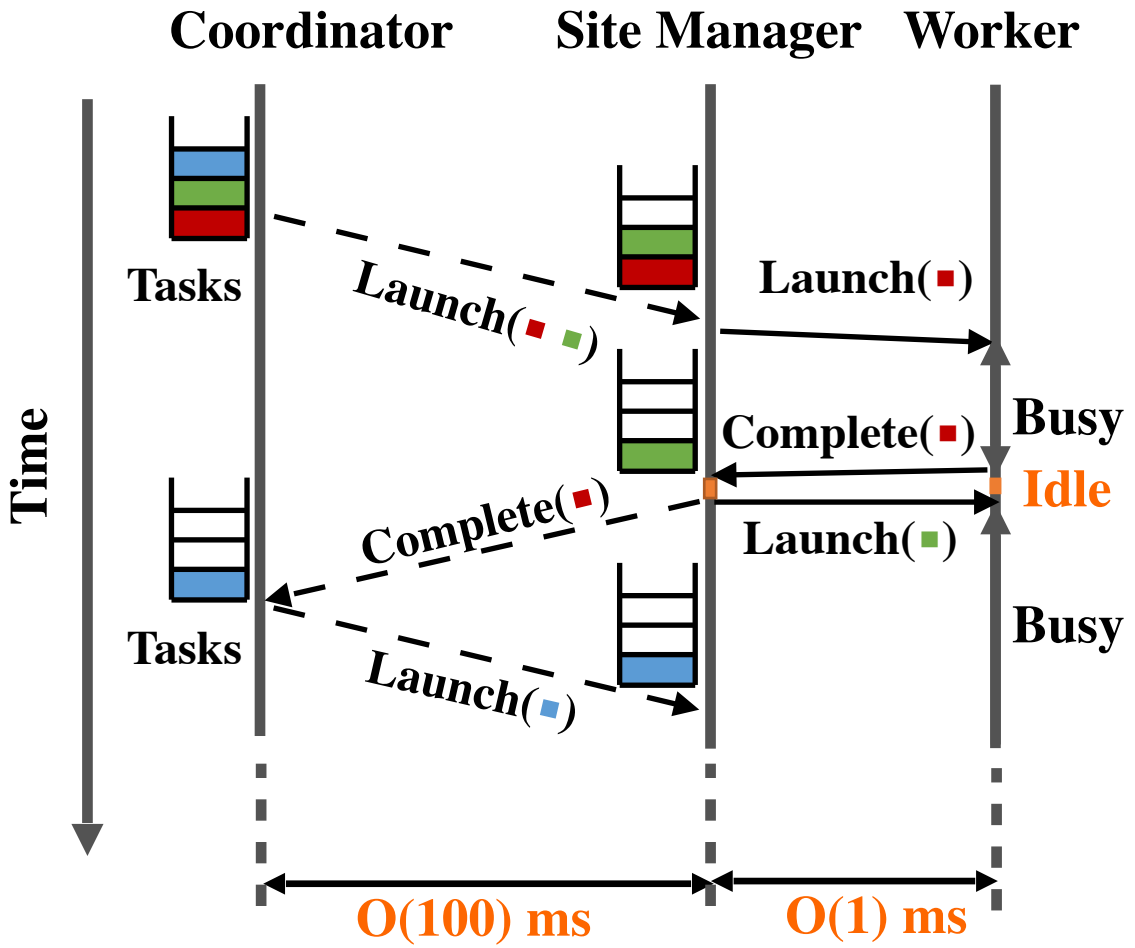
Task Early-binding in Control Plane

- **Coordinator ⟷ Site Manager**
  - Inter-site operations are *early-binding*

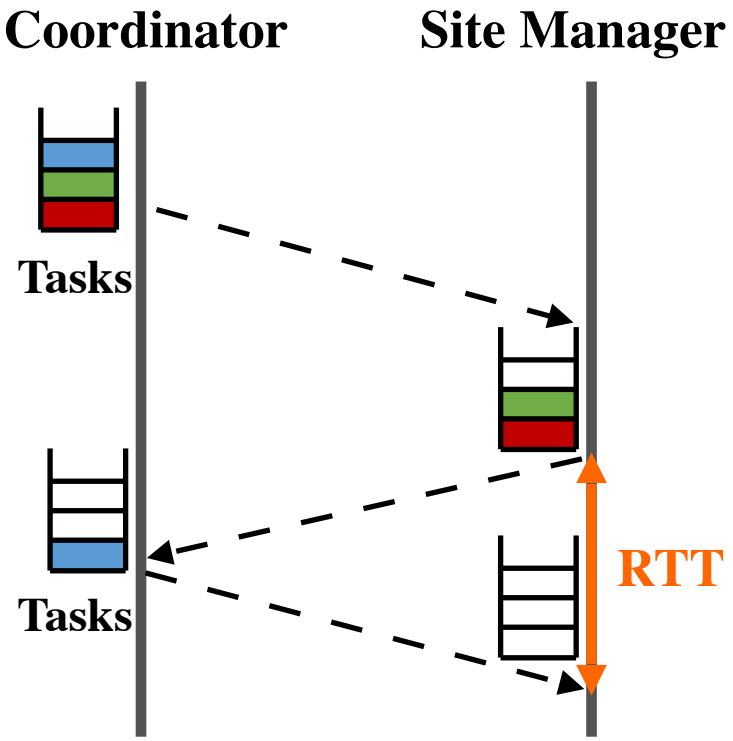
  → Guarantee high utilization

- **Site Manager ⟷ Worker**
  - Intra-site operations are *late-binding*

  → Retain precise views

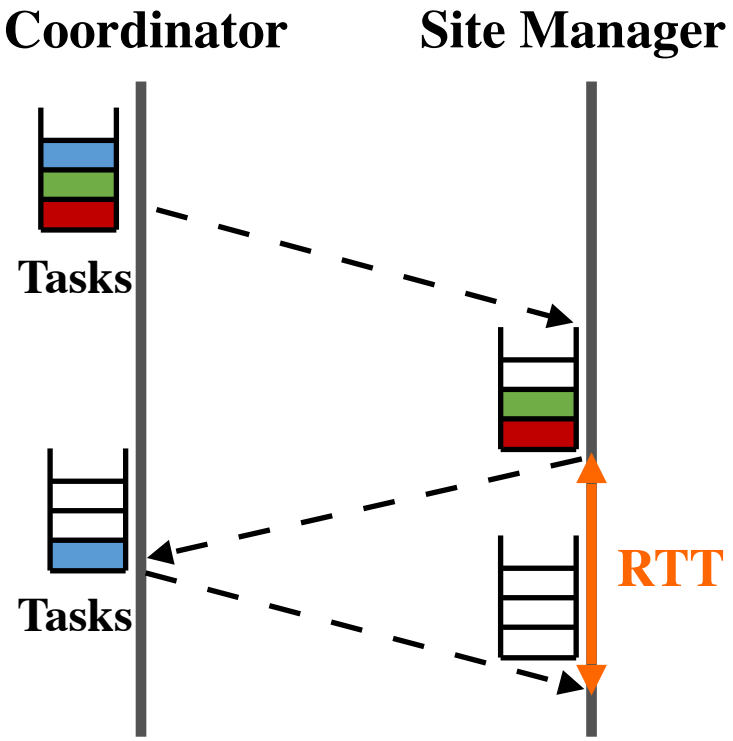


Challenge 1.1: How Many Tasks to Push?



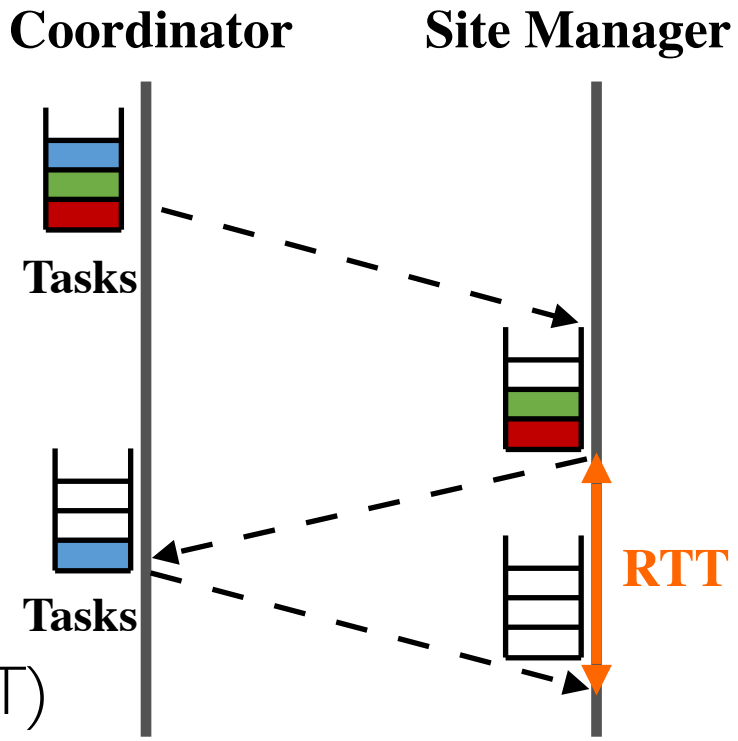
Challenge 1.1: How Many Tasks to Push?

- **Queue up too few**
  - Not enough work → *Underutilization*
- **Queue up too many**
  - Scheduling too early → *Suboptimal placement*



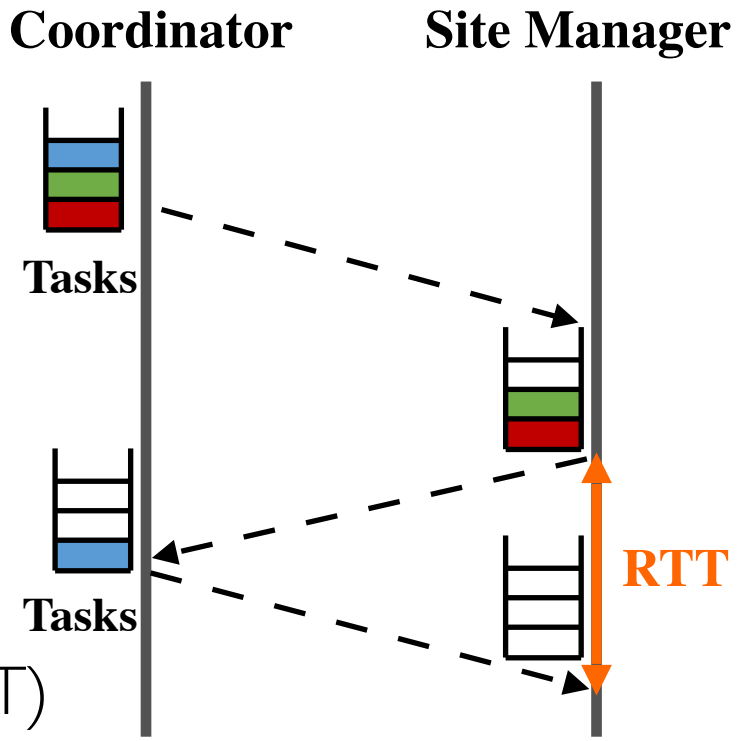
Challenge 1.1: How Many Tasks to Push?

- **Queue up too few**
  - Not enough work → *Underutilization*
- **Queue up too many**
  - Scheduling too early → *Suboptimal placement*
- **Target:**
  - Total duration of queued tasks ≃ Round-Trip Time(RTT)



Challenge 1.1: How Many Tasks to Push?

- **Queue up too few**
  - Not enough work → *Underutilization*
- **Queue up too many**
  - Scheduling too early → *Suboptimal placement*
- **Target:**
  - Total duration of queued tasks ≃ Round-Trip Time(RTT)
    - SoI works well *w/o* precise knowledge of task duration
      - Hoeffding-bound (details in paper)

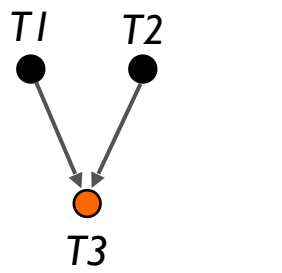


Challenge 1.2: How to Push Tasks w/ Dependencies?

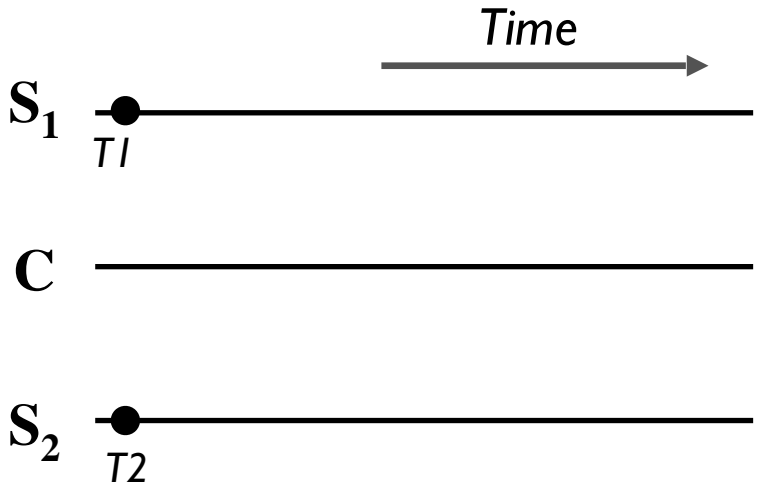
- **Task placements depend on upstream outputs**
  - In order to reduce data transfers over networks

Challenge 1.2: How to Push Tasks w/ Dependencies?

- **Task placements depend on upstream outputs**
  - In order to reduce data transfers over networks



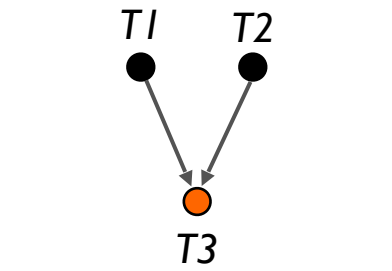
Task Dependencies



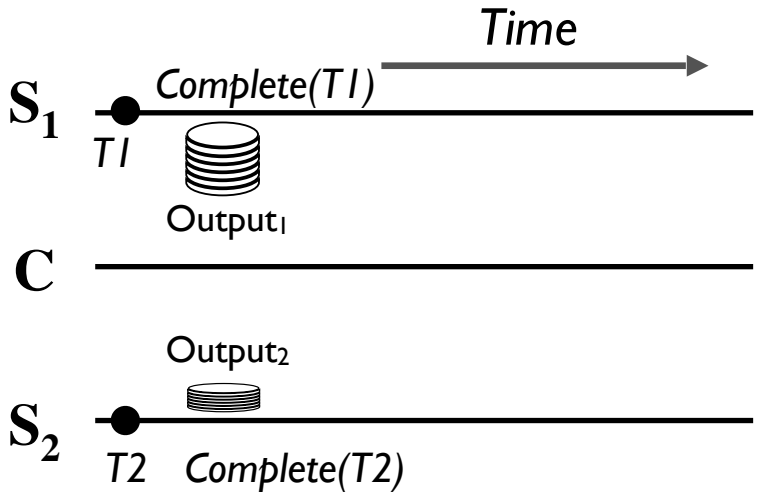
Design in Existing Engines

Challenge 1.2: How to Push Tasks w/ Dependencies?

- **Task placements depend on upstream outputs**
  - In order to reduce data transfers over networks



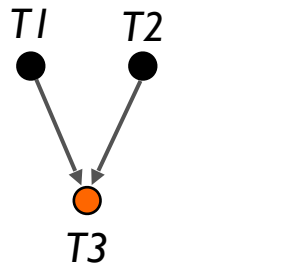
Task Dependencies



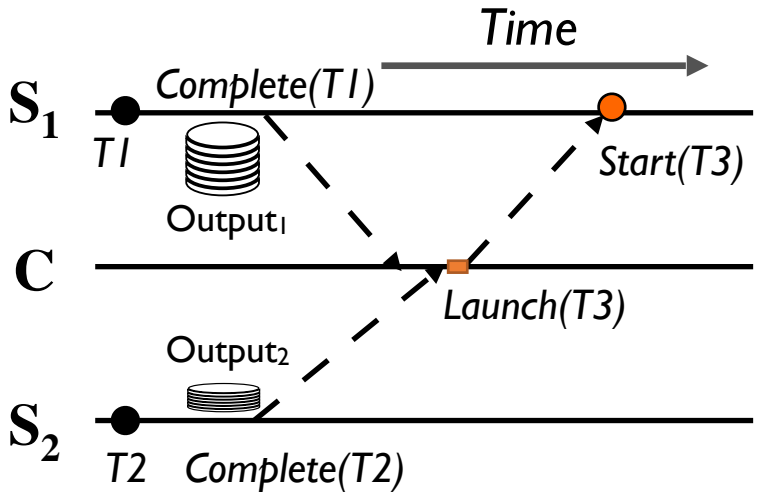
Design in Existing Engines

Challenge 1.2: How to Push Tasks w/ Dependencies?

- **Task placements depend on upstream outputs**
  - In order to reduce data transfers over networks



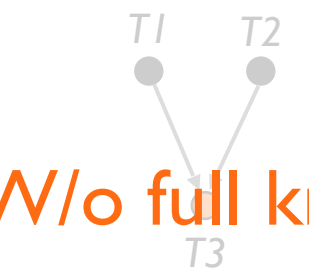
Task Dependencies



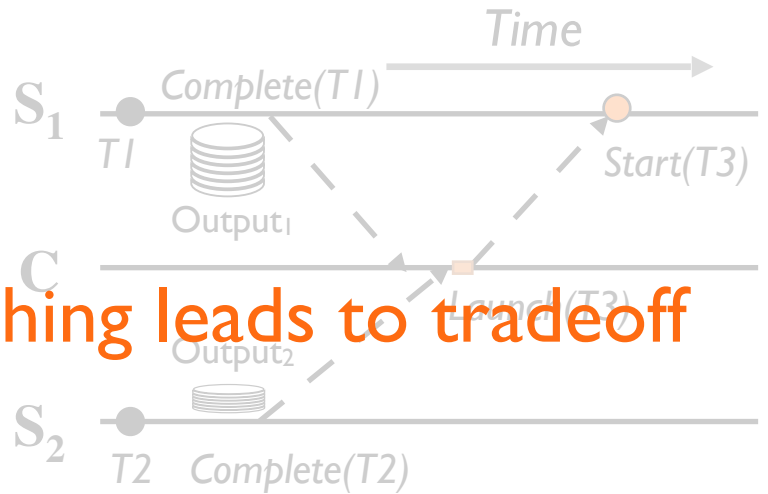
Design in Existing Engines

Challenge 1.2: How to Push Tasks w/ Dependencies?

- **Task placements depend on upstream outputs**
  - In order to reduce data transfers over networks



Task Dependencies

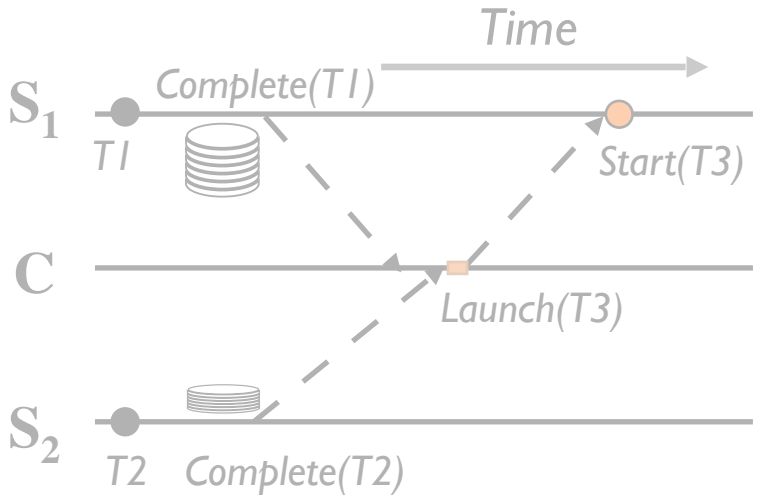


Design in Existing Engines

W/o full knowledge, pushing leads to tradeoff

Challenge 1.2: How to Push Tasks w/ Dependencies?

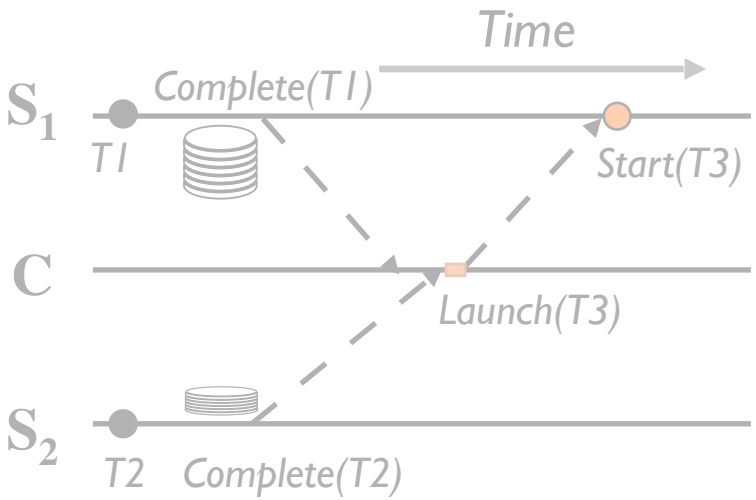
1. Sol improves utilization by pushing with speculation
   - E.g., historical information



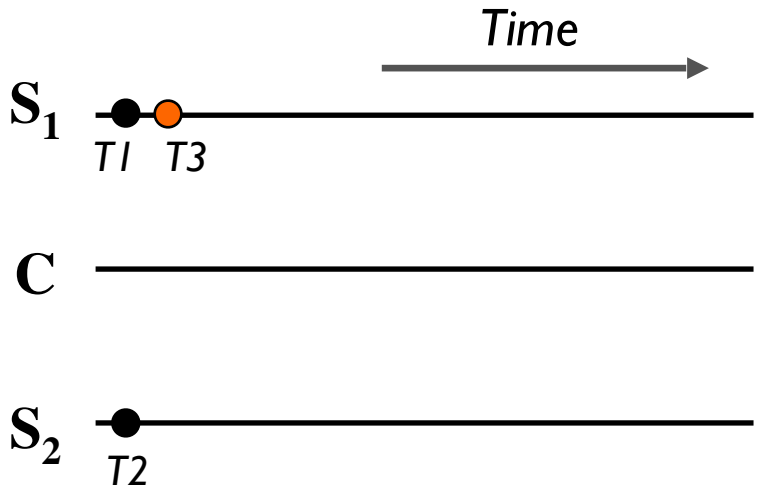
Design in Existing Engines

Challenge 1.2: How to Push Tasks w/ Dependencies?

1. Sol improves utilization by pushing with speculation
   - E.g., historical information



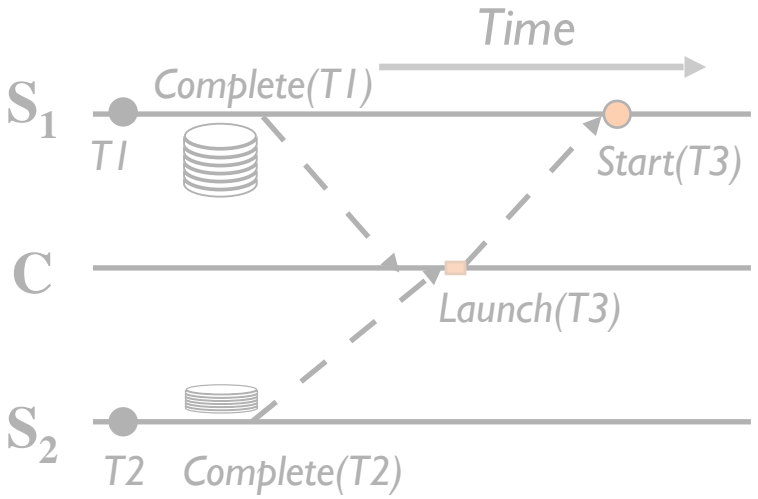
Design in Existing Engines



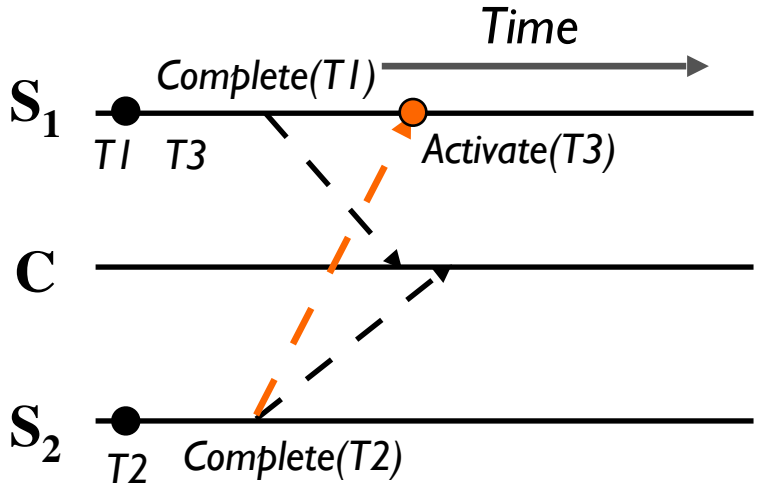
Push w/ Correct Speculations

Challenge 1.2: How to Push Tasks w/ Dependencies?

1. Sol improves utilization by pushing with speculation
   - E.g., historical information



Design in Existing Engines

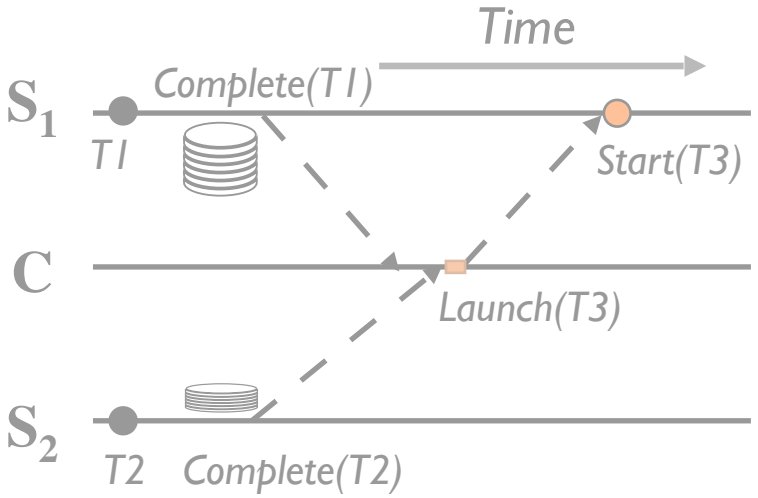


Push w/ Correct Speculations

Sol saves RTTs

Challenge 1.2: How to Push Tasks w/ Dependencies?

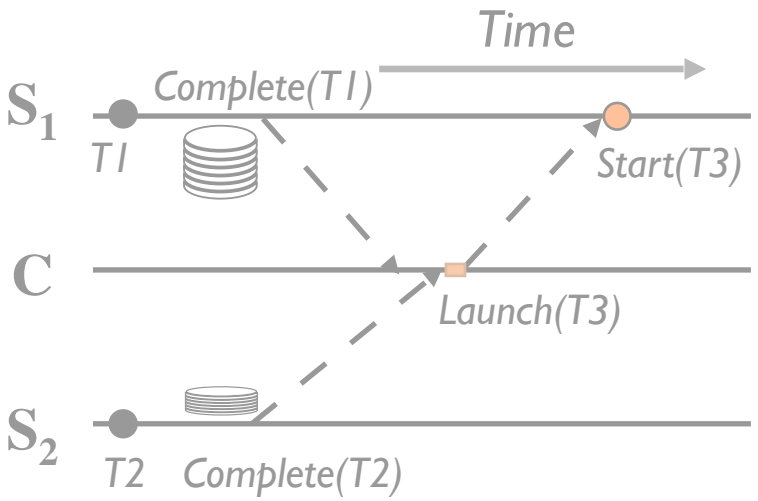
2. In case of mistakes, Sol retains good scheduling by recovering
   - With worker-initiated re-scheduling



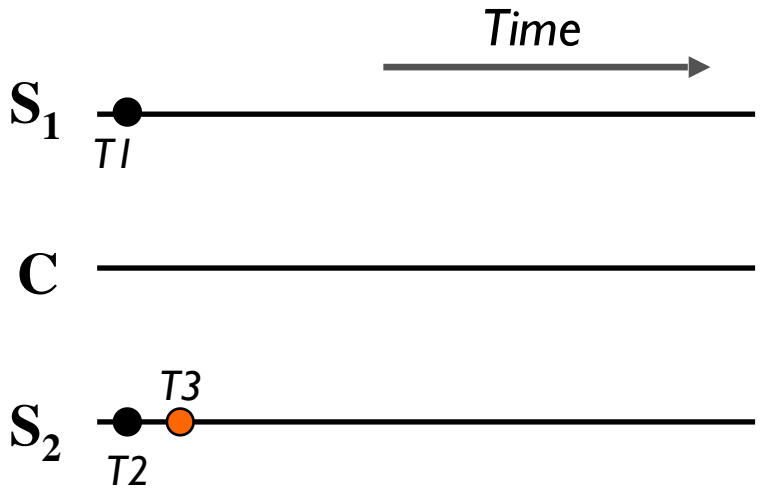
Design in Existing Engines

Challenge 1.2: How to Push Tasks w/ Dependencies?

2. In case of mistakes, Sol retains good scheduling by recovering
   - With worker-initiated re-scheduling



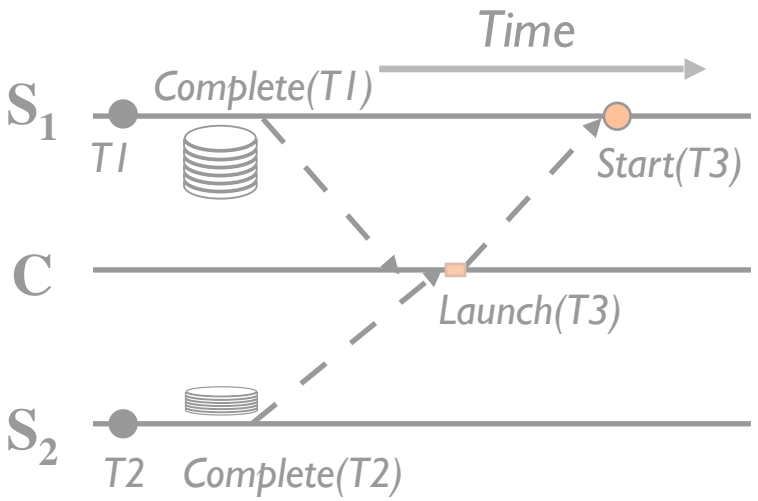
Design in Existing Engines



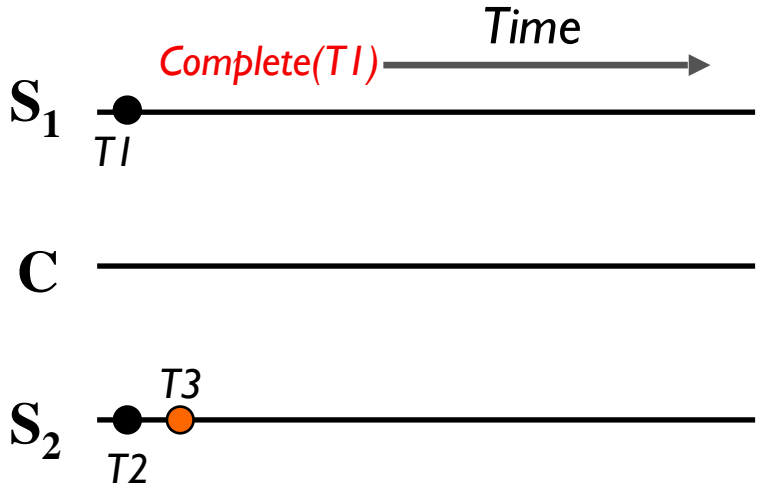
Push under Mispredictions

Challenge 1.2: How to Push Tasks w/ Dependencies?

2. In case of mistakes, Sol retains good scheduling by recovering
   - With worker-initiated re-scheduling



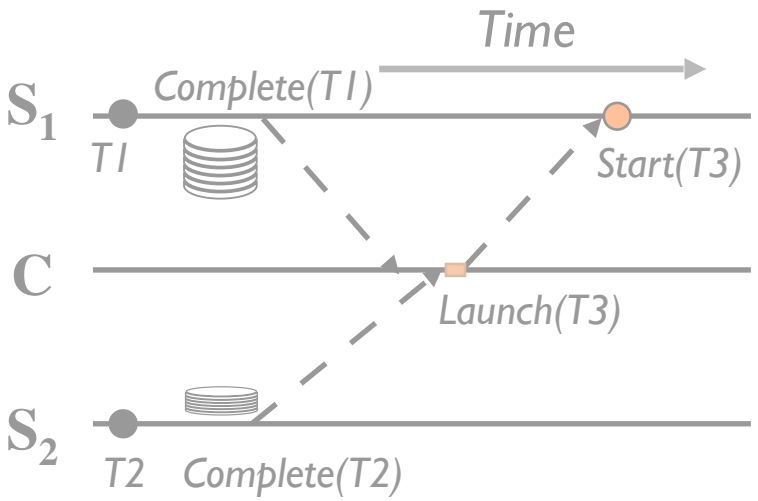
Design in Existing Engines



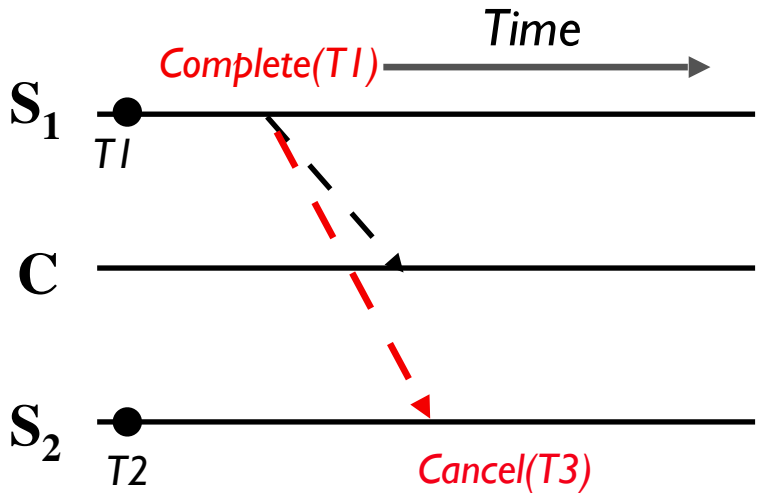
Push under Mispredictions

Challenge 1.2: How to Push Tasks w/ Dependencies?

2. In case of mistakes, Sol retains good scheduling by recovering
   - With worker-initiated re-scheduling



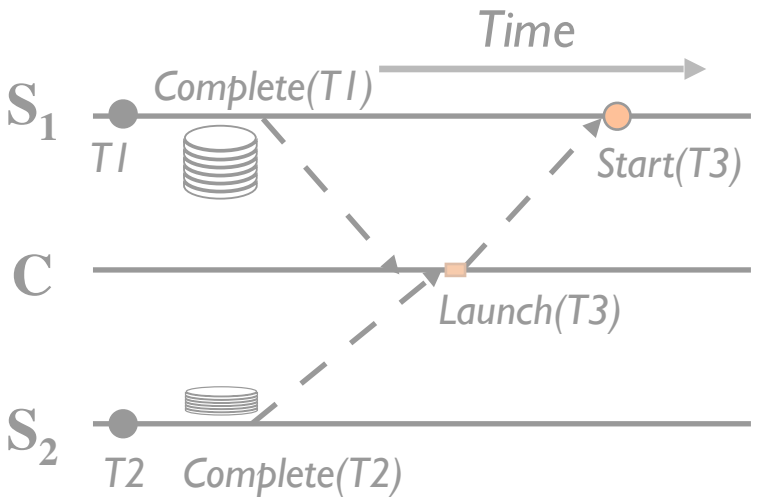
Design in Existing Engines



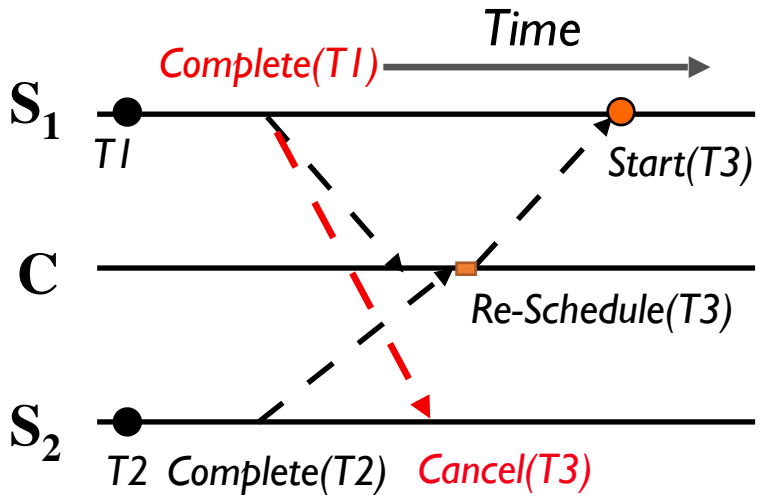
Push under Mispredictions

Challenge 1.2: How to Push Tasks w/ Dependencies?

2. In case of mistakes, Sol retains good scheduling by recovering
   - With worker-initiated re-scheduling



Design in Existing Engines

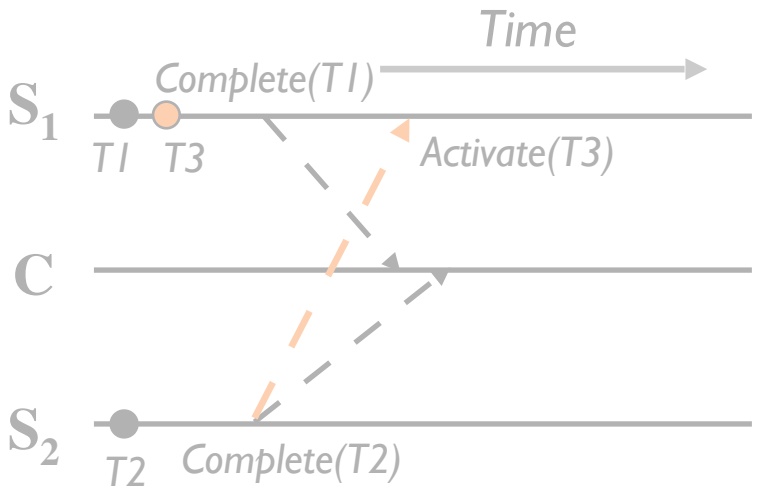


Push under Mispredictions

Sol does not make things worse

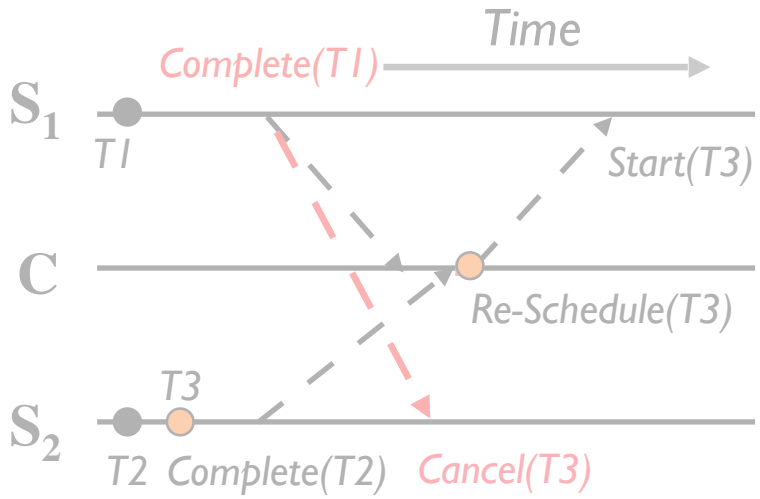
Task Early-binding in Control Plane

- Sol improves *utilization* while retaining good scheduling quality



Push w/ Correct Speculations

Sol improves utilization



Push under Mispredictions

Sol retains good scheduling quality

Outline

- Today's Execution Engines
- Sol Architecture
- Control Plane Design
- Data Plane Design
- Evaluation

Problem #2

Low b/w → CPU underutilization

Decouple resource provisioning to improve CPU utilization

Resource Decoupling in Data Plane

- **Decouple the resource provisioning *internally* with**
  - ***Communication task***: prepare data over networks

Resource Decoupling in Data Plane

- **Decouple the resource provisioning *internally* with**
  - ***Communication task***: prepare data over networks
  - ***Computation task***: perform computation on input

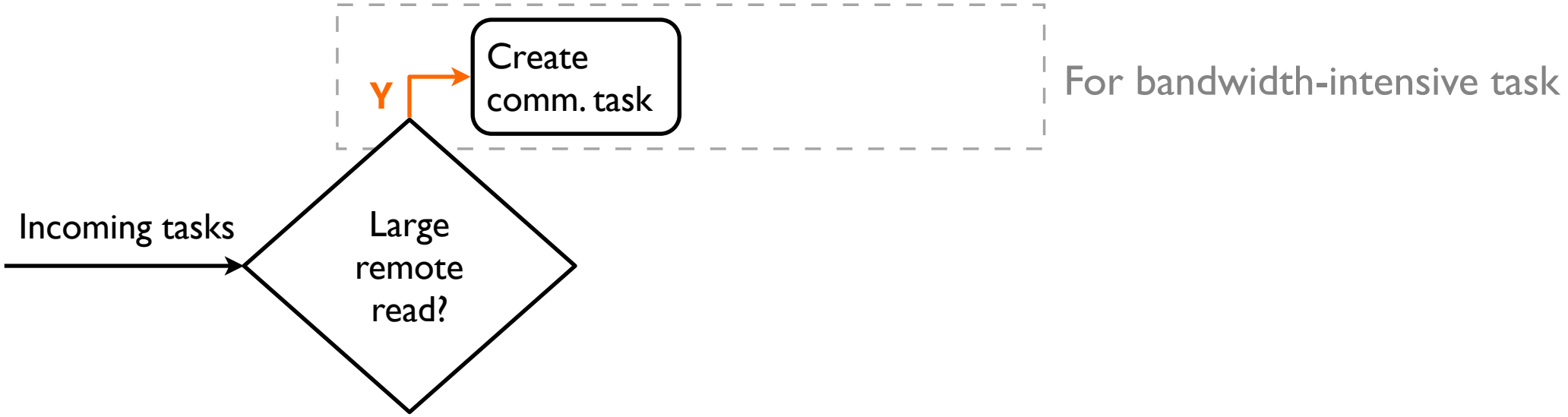
Resource Decoupling in Data Plane

- **Decouple the resource provisioning *internally* with**
  - ***Communication task***: prepare data over networks
  - ***Computation task***: perform computation on input

Sol scales down CPU requirements and reclaims unused CPUs

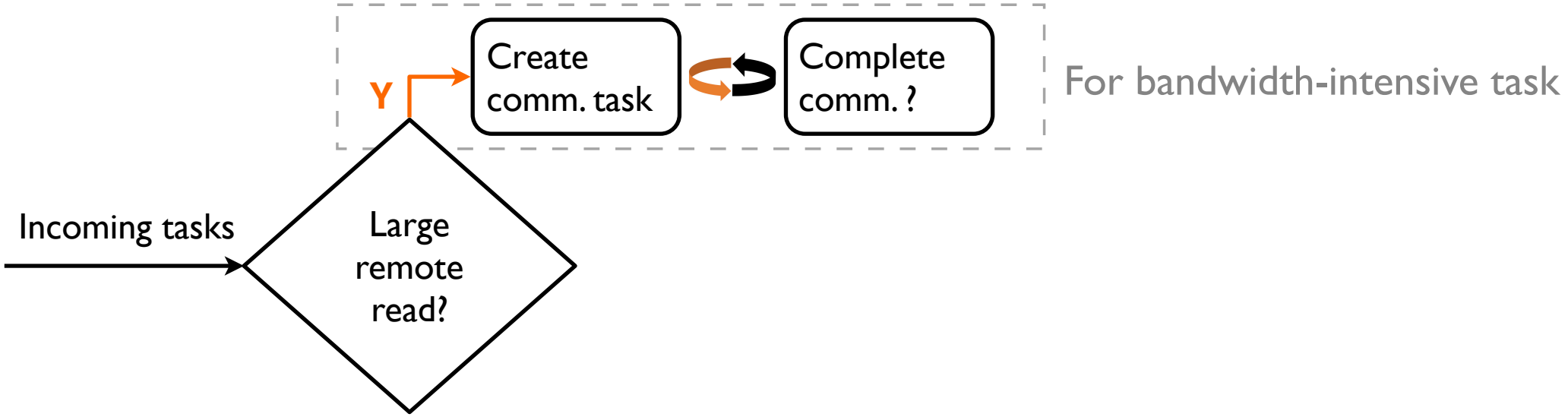
Challenge 2: How to Manage Jobs?

Challenge 2: How to Manage Jobs?



Control flow of decoupling

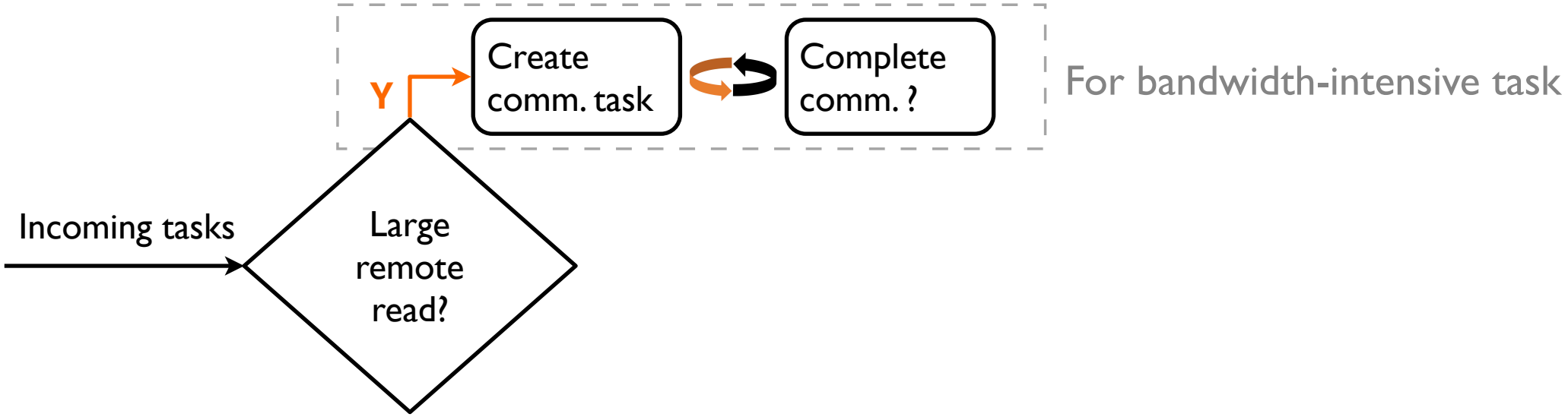
Challenge 2: How to Manage Jobs?



Control flow of decoupling

Challenge 2: How to Manage Jobs?

- **How many communication tasks to create?**
  - Too few → Network is not saturated
  - Too many → CPUs are not saturated

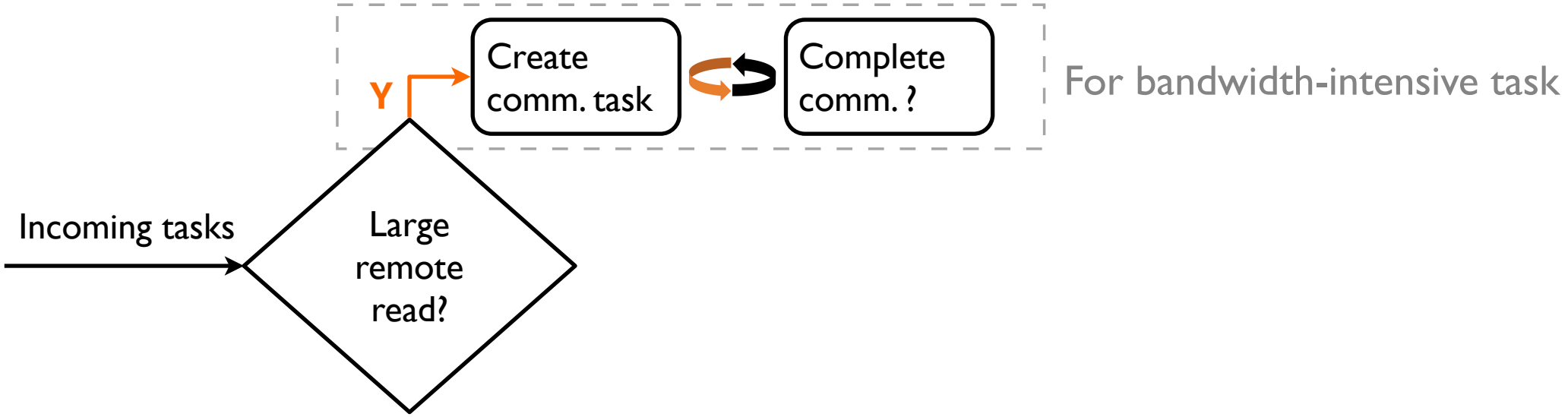


Control flow of decoupling

Challenge 2: How to Manage Jobs?

- **How many communication tasks to create?**
  - Too few → Network is not saturated
  - Too many → CPUs are not saturated

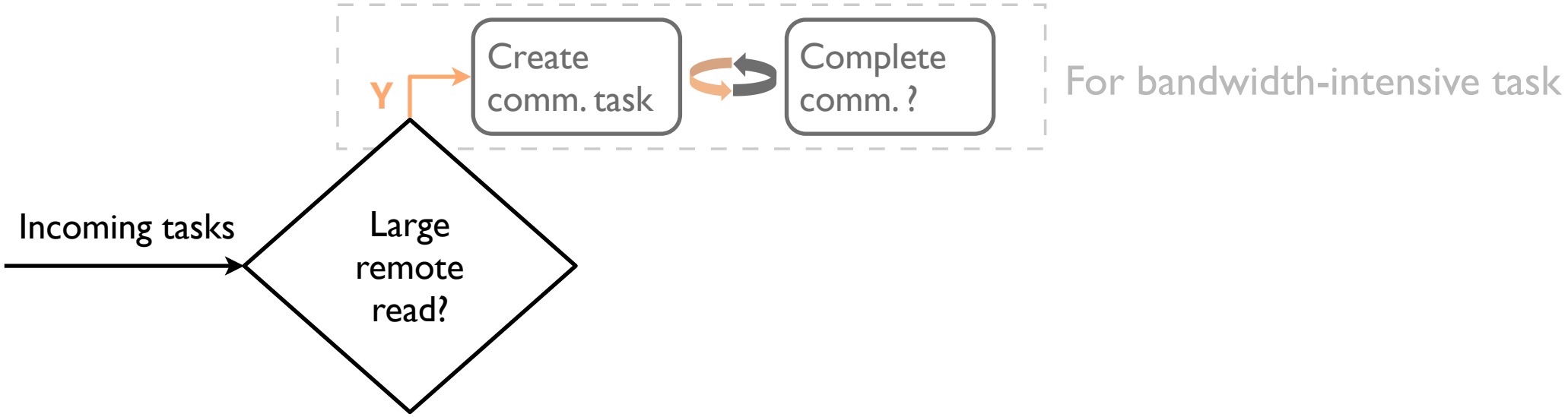
} *Adapt to available bandwidth*



Control flow of decoupling

Challenge 2: How to Manage Jobs?

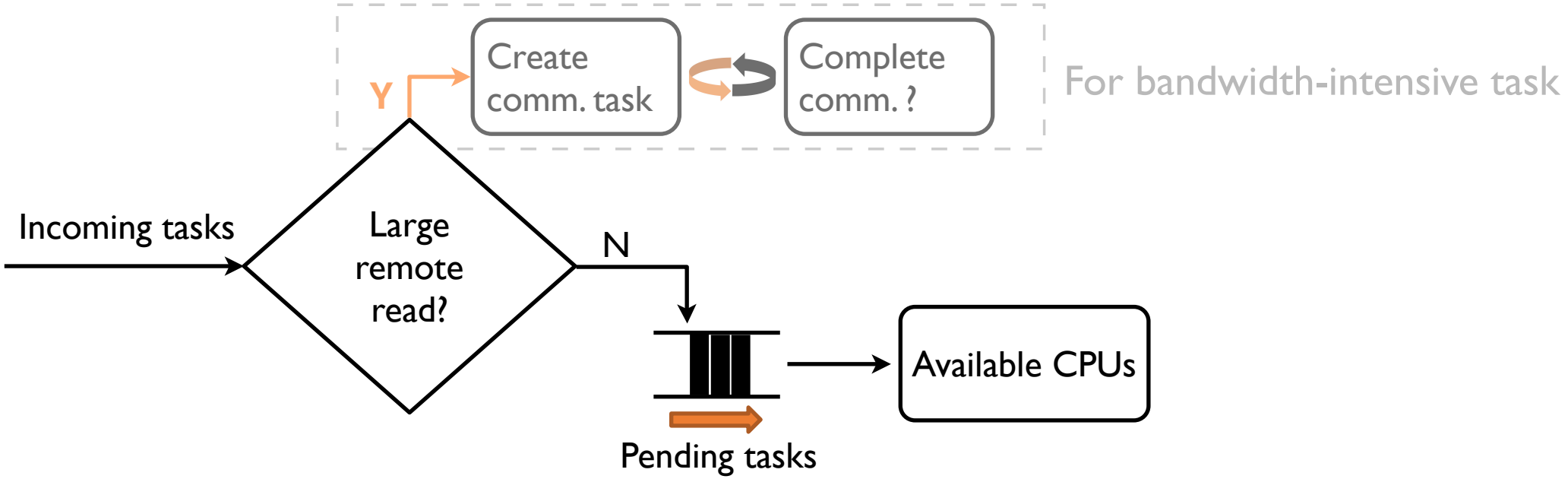
- How to manage the computation tasks?



Control flow of decoupling

Challenge 2: How to Manage Jobs?

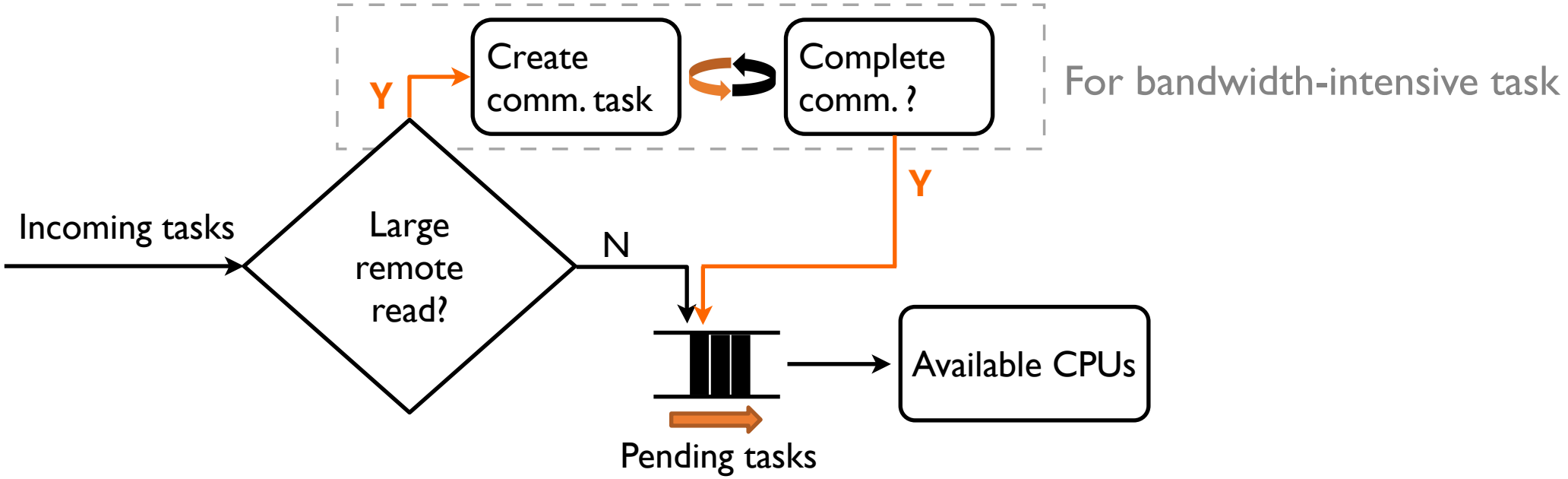
- How to manage the computation tasks?



Control flow of decoupling

Challenge 2: How to Manage Jobs?

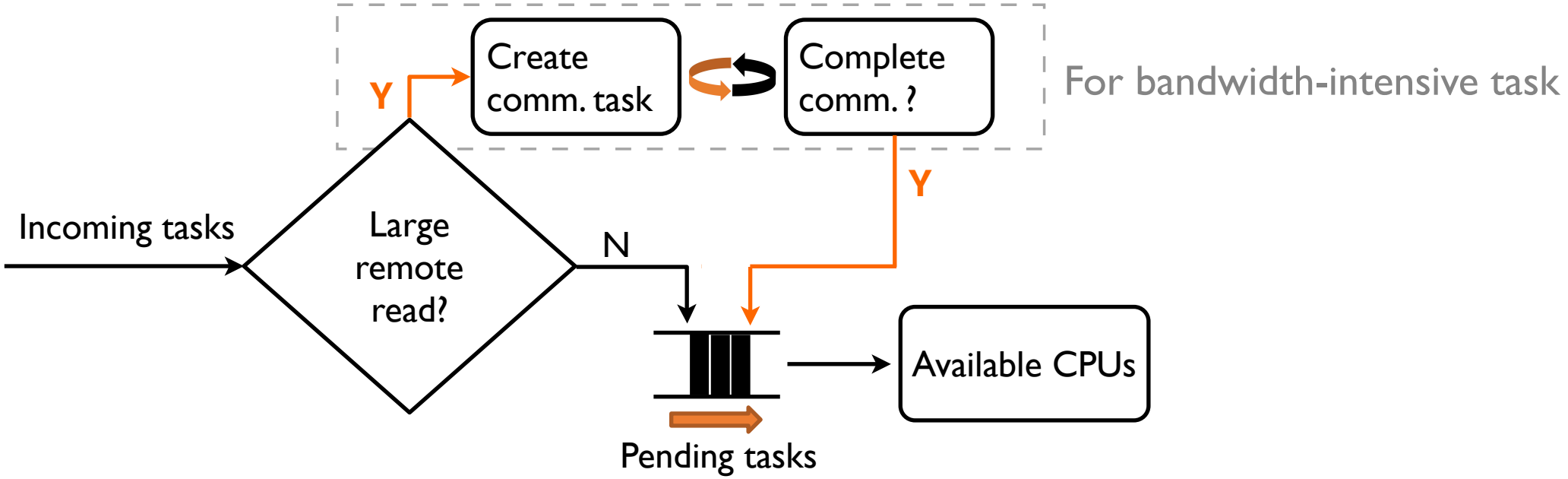
- How to manage the computation tasks?



Control flow of decoupling

Challenge 2: How to Manage Jobs?

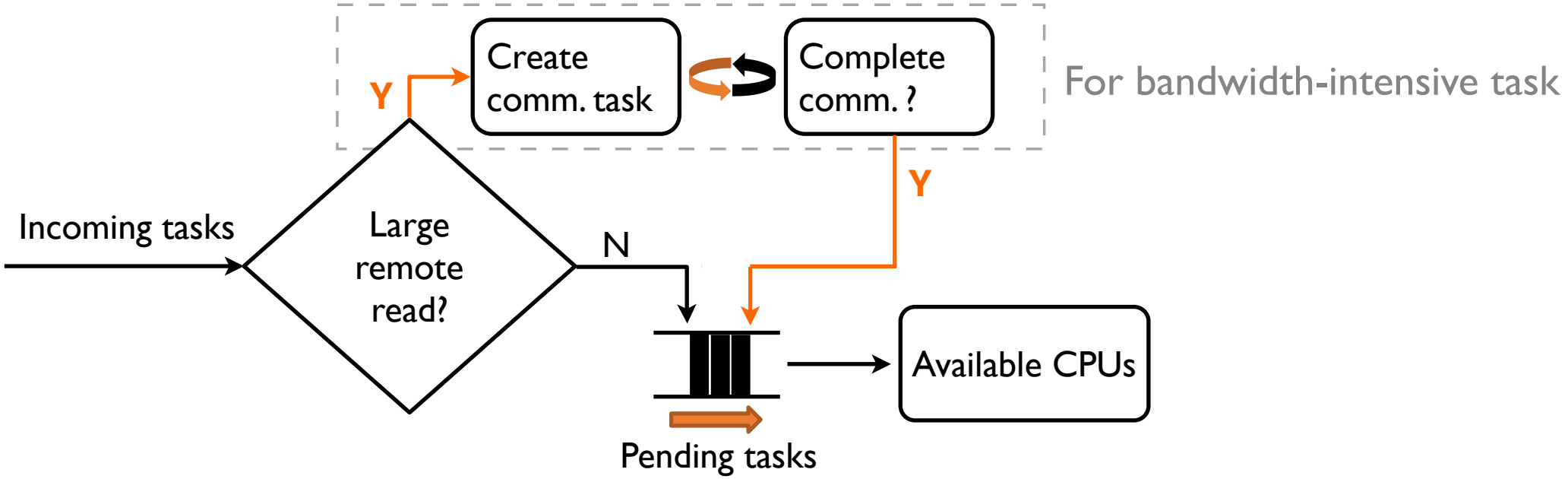
- **How to manage the computation tasks?**
  - ***Prioritize*** them when data is ready



Control flow of decoupling

Challenge 2: How to Manage Jobs?

- **How to manage the computation tasks?**
  - ***Prioritize*** them when data is ready



Control flow of decoupling

Evaluation

With a prototype supporting generic data processing

- **Environment**
  - 10-site deployment in EC2
  - 4 *m4.4xlarge* VMs in each site

Deployment over WAN

Evaluation

With a prototype supporting generic data processing

- **Environment**
  - 10-site deployment in EC2
  - 4 *m4.4xlarge* VMs in each site

Deployment over WAN

How does Sol perform:

1. compared to existing engines?
2. across design space?
3. under uncertainties?

Sol Improves Job Performance and Resource Util. (WAN)

Benchmark — multi-job execution

- Latency-sensitive TPC queries
- Bandwidth-intensive TeraSort

Baseline

- Apache Spark

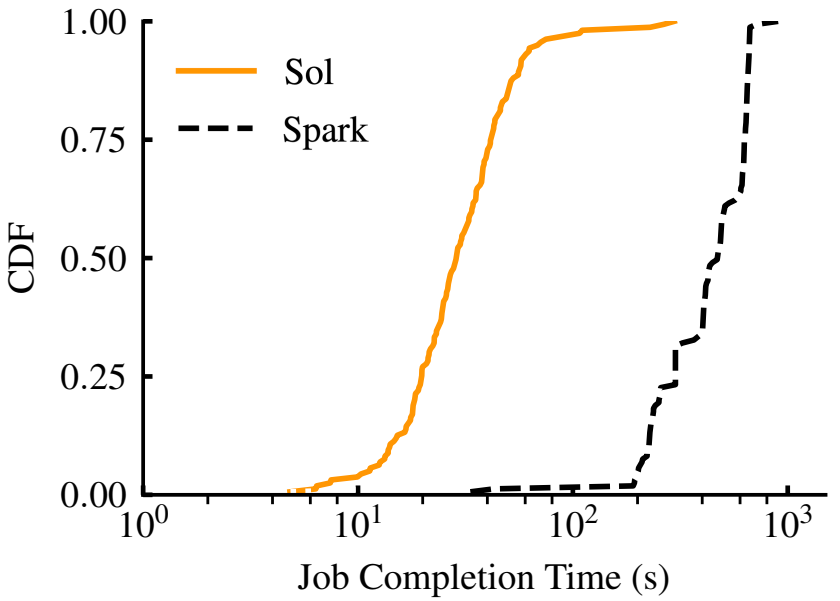
Sol Improves Job Performance and Resource Util. (WAN)

Benchmark — multi-job execution

- Latency-sensitive TPC queries
- Bandwidth-intensive TeraSort

Baseline

- Apache Spark

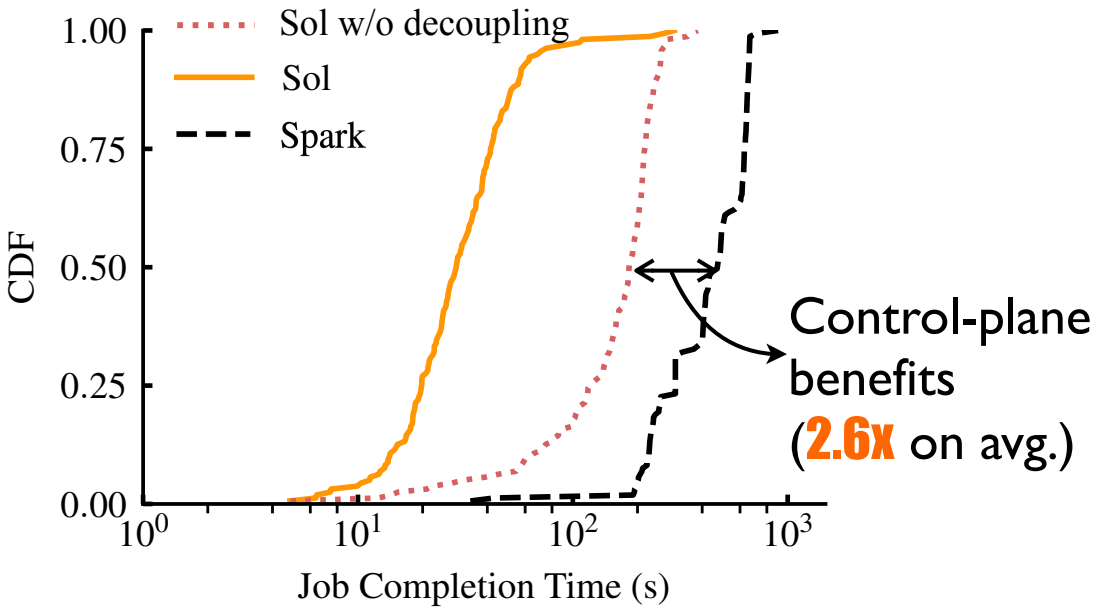


16.4x improvement on average

Sol Improves Job Performance and Resource Util. (WAN)

Control Plane:

Early-binding → Less idle time

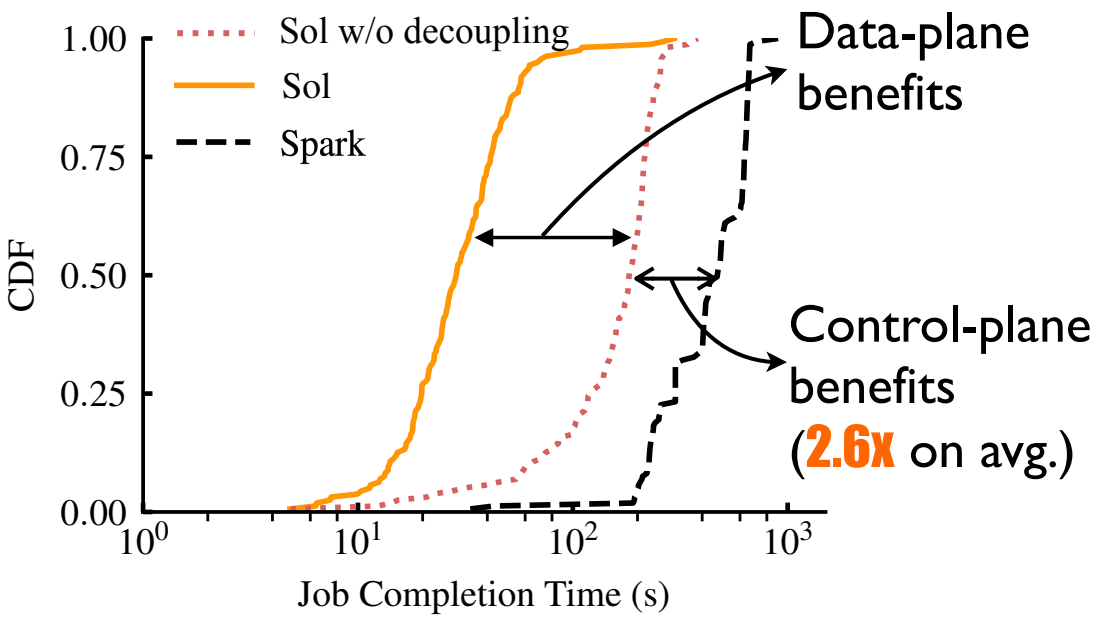


16.4x improvement on average

Sol Improves Job Performance and Resource Util. (WAN)

Control Plane:
Early-binding → Less idle time

Data Plane:
Decoupling → Less under-util.



16.4x improvement on average

Sol Improves Job Performance and Resource Util. (WAN)

Control Plane:

Early-binding → Less idle time

+

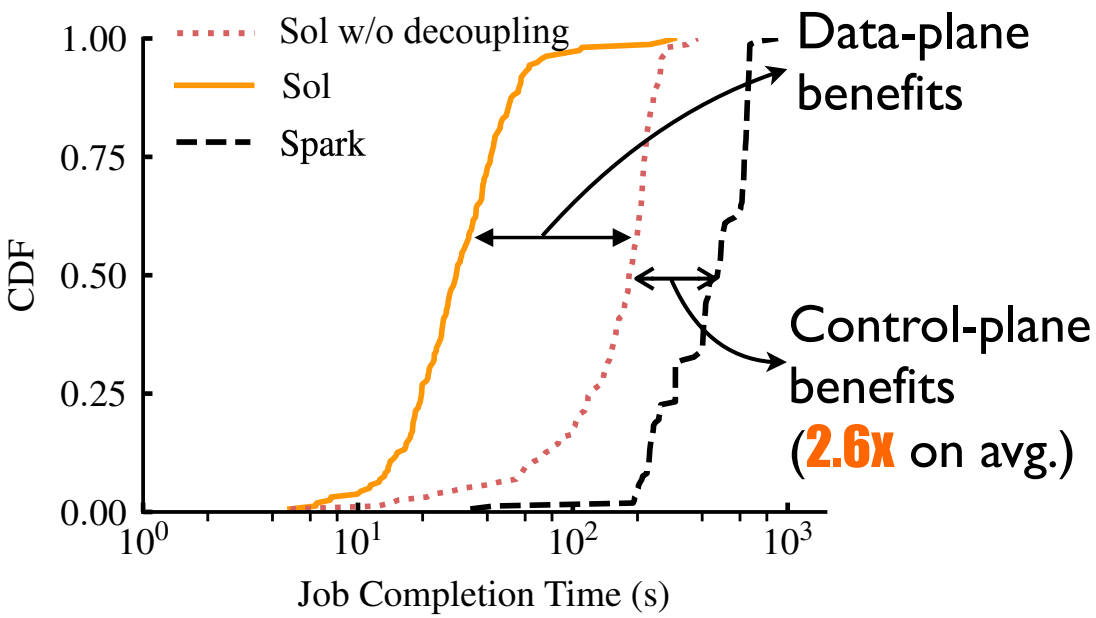
Data Plane:

Decoupling → Less under-util.

↓

16.4x *better job completion*

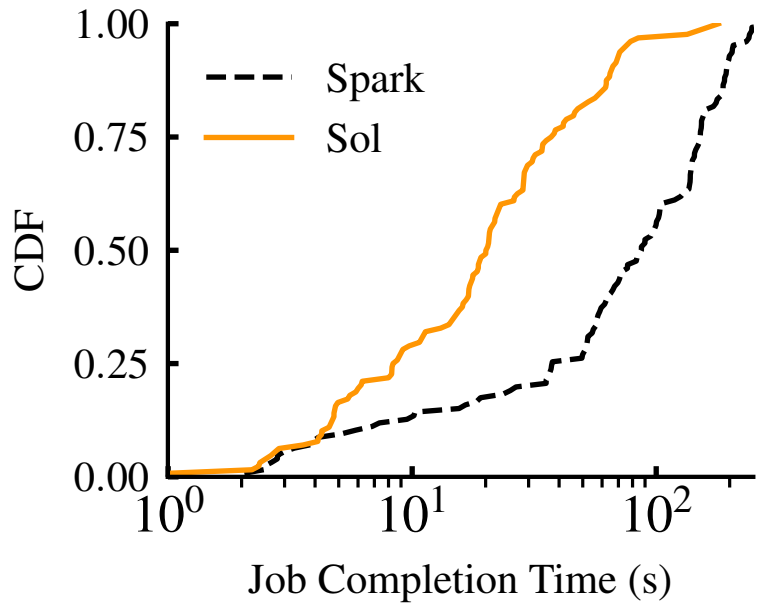
1.8x *better CPU util.*



16.4x improvement on average

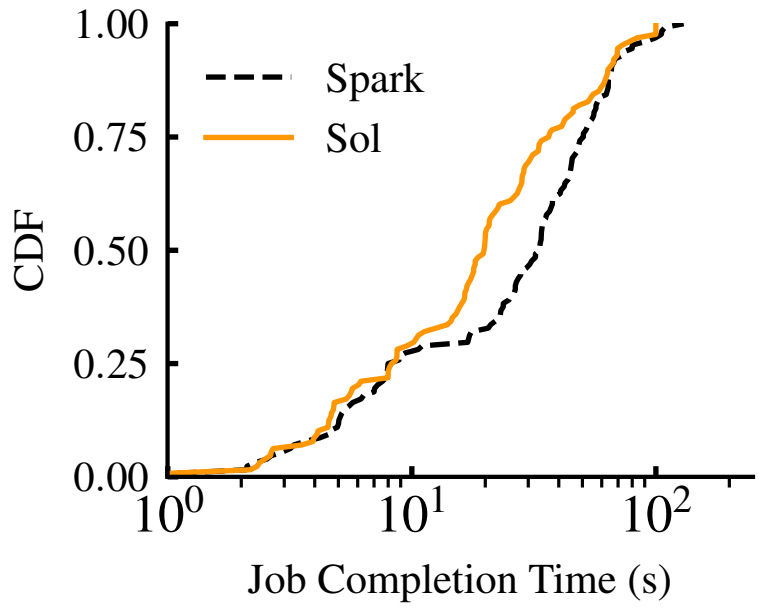
Sol Performs Well Across Design Space (LAN)

Low-bandwidth setting (1 Gbps)



3.9x improvement on average

High-bandwidth setting (10 Gbps)



1.3x improvement on average

Sol

https://github.com/SymbioticLab/Sol

A *federated* execution engine for ***diverse*** network conditions with

- *Faster job execution*
- *Higher resource utilization*

Improve CPU util. {
- *before* task executions → *Early-binding* of tasks
- *during* task executions → *Decoupling* of resource provisioning

UNIVERSITY OF MICHIGAN