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**
- Spark
- TEZ
- Apache Flink
- calcite
- HDFS
- Apache Mesos

**Efforts for Computation over WAN**
- Iridium
- CLARINET
- Tetrium
- Azure Cosmos DB
- Google Spanner
Execution Engine: Core of Big Data Stack

Select *
FROM ...;

SQL Queries

K-means, SVM

AI/ML

WordCount, TopKCount

Stream Processing

...
Execution Engine: Core of Big Data Stack

Select *
FROM …;

SQL Queries

K-means, SVM

AI/ML

Stream Processing

WordCount, TopKCount

...
Execution Engine: Core of Big Data Stack

SQL Queries

K-means, SVM

WordCount, TopKCount

Execution Planner

Typical job execution plans

Job 1

Job 2
Execution Engine: Core of Big Data Stack

Select *
FROM ...;

SQL Queries

K-means, SVM

AI/ML

WordCount, TopKCount

Stream Processing

... ... ...

Coordinator

Worker₁

Worker₂ ...

Workerᴺ
Execution Engine: Core of Big Data Stack

Select * FROM …;

SQL Queries  K-means, SVM  Stream Processing

AI/ML  WordCount, TopKCount

Execution Planner

Execution Engine

Coordinator  Worker_1  Worker_2  …  Worker_N

Resource Scheduler  Storage System
Execution Engine: Core of Big Data Stack

SQL Queries

Execution Planner

Execution Engine

Coordinator

Worker_1

Worker_2

... Worker_N

Resource Scheduler

Storage System

Select * FROM ...;

K-means, SVM

AI/ML

WordCount, TopKCount

Stream Processing

...
Execution Engine: Core of Big Data Stack

Select * FROM ...;

SQL Queries
K-means, SVM
AI/ML
WordCount, TopKCount
Stream Processing

Execution Planner

Coordinator
Worker_1
Worker_2
... Worker_N

Execution Engine

Resource Scheduler
Storage System

Efforts for Computation in LAN

Spark
TEZ
Apache Flink

Efforts for Computation over WAN

Azure Cosmos DB
Tetrium
Google Spanner

CLARINET
iridium
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

CDF across Queries

10 Gbps, $O(1)$ ms

1 Gbps, $O(1)$ ms

10 Gbps, $O(100)$ ms

1 Gbps, $O(100)$ ms

Queries from 100 GB TPC Benchmarks
Impact of Networks on Latency-sensitive Jobs

CDF across Queries

10 Gbps, $O(1)$ ms
1 Gbps, $O(100)$ ms

 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

Problem #1

Slow job execution in high-latency networks

Queries from 100 GB TPC Benchmarks
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

**Problem #1**

Control Plane Inefficiency Due to High Latency

- **Late-binding** of tasks postpones scheduling
- **Slow** job execution in high-latency networks
Impact of Networks on Bandwidth-intensive Jobs

Stage 1  Stage 2  Stage 3

Data transfers over networks

Query25 on 1TB TPC benchmark
Impact of Networks on Bandwidth-intensive Jobs

Stage 1
Stage 2
Stage 3

Data transfers over networks

Query25 on 1TB TPC benchmark

Occupied CPUs

Resource utilization throughout the job
Impact of Networks on Bandwidth-intensive Jobs

Stage 1

Stage 2

Stage 3

Data transfers over networks

Query25 on 1TB TPC benchmark

Resource utilization throughout the job

Stage 1

Stage 2

Stage 3

Time (s)

0 50 100 150 200 250

Percentage of the Total (%)

0 25 50 75 100

Occupied CPUs

CPU Util.

B/w Util.

Low CPU util.
Data Plane Inefficiency Due to Low Bandwidth

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

Site 1

- Site 1
  - Site 2
  - Site 3

WAN

Task Arrivals

O(100) ms

Sol Coordinator

O(100) ms
Sol: A Federated Execution Engine

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

- **Site Manager**
  - Coordinate local workers
  - Manage queued tasks

**Sol Architecture**

- Task Arrivals
- LAN
- WAN
- Site Manager
- Site 2
- Site 3

O(100) ms
Sol: A Federated Execution Engine

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

- **Site Manager**
  - Coordinate local workers
  - Manage queued tasks

---

**Sol Architecture**

- **Sol Coordinator**
- **Site Manager**
- **Site 2**
- **Site 3**

*Task Arrivals*
Sol: A Federated Execution Engine

- Central Coordinator
  - Coordinate inter-site executions

- Site Manager
  - Coordinate local workers
  - Manage queued tasks

- Task Manager
  - Manage worker resource

```
<table>
<thead>
<tr>
<th>Site</th>
<th>Manager</th>
<th>Task Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Sol Coordinator

```
Task Arrivals
```

LAN

```
O(100) ms
```

WAN

```
O(100) ms
```

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

Coordinator

Worker

Time

Tasks

Launch

Complete

Busy

Idle

O(100) ms
Task Early-binding in Control Plane

Coordinator  Site Manager  Worker

Time

Tasks

O(100) ms  O(1) ms
Task Early-binding in Control Plane

- Coordinator
- Site Manager
- Worker

- O(100) ms
- O(1) ms

Launch(● □)
Task Early-binding in Control Plane

Coordinator

Worker

Site Manager

Launch

Complete

Busy

Time

Tasks

O(100) ms

O(1) ms

Tasks

Launch

Complete

Busy

O(100) ms

O(1) ms

15
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)
    - Sol 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**

- $T_1$ and $T_2$ depend on $T_3$
- $T_1$ and $T_2$ can be pushed together if $T_3$ is available in advance

**Design in Existing Engines**

- $S_1$ and $S_2$ are separate tasks
- $C$ is a communication task between $S_1$ and $S_2$
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
Challenge 1.2: How to Push Tasks w/ Dependencies?

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

W/o full knowledge, pushing leads to tradeoff
I. Sol improves **utilization** by pushing with speculation
   • E.g., historical information

Design in Existing Engines
I. Sol improves **utilization** by pushing with speculation

- E.g., historical information

---

**Design in Existing Engines**

**Push w/ Correct Speculations**
I. Sol improves **utilization** by pushing with speculation

- E.g., historical information

---

**Design in Existing Engines**

---

**Push w/ Correct Speculations**

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

Design in Existing Engines
2. In case of mistakes, Sol retains good scheduling by recovering

- With worker-initiated re-scheduling
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
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 $\rightarrow$ CPU underutilization

Decouple resource provisioning to improve CPU utilization
• Decouple the resource provisioning *internally* with
  • *Communication task*: prepare data over networks
• 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 \textit{internally} with
  - \textit{Communication task}: prepare data over networks
  - \textit{Computation task}: perform computation on input

\textit{Sol} scales down \textit{CPU requirements} and \textit{reclaims} unused \textit{CPUs}
Challenge 2: How to Manage Jobs?
Challenge 2: How to Manage Jobs?

For bandwidth-intensive task

Incoming tasks

Large remote read?

Create comm. task

Y

Control flow of decoupling
Challenge 2: How to Manage Jobs?

For bandwidth-intensive task

Incoming tasks

Large remote read?

Create comm. task

Complete comm.?

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

For bandwidth-intensive task
Challenge 2: How to Manage Jobs?

• **How many communication tasks to create?**
  - Too few $\rightarrow$ Network is not saturated
  - Too many $\rightarrow$ CPUs are not saturated

  \[\text{Adapt to available bandwidth}\]

---

Incoming tasks

- Large remote read?

Y

Create comm. task

\[\text{Complete comm.?}\]

Control flow of decoupling

For bandwidth-intensive task
Challenge 2: How to Manage Jobs?

• How to manage the computation tasks?
Challenge 2: How to Manage Jobs?

- How to manage the computation tasks?
Challenge 2: How to Manage Jobs?

- How to manage the computation tasks?
Challenge 2: How to Manage Jobs?

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

For bandwidth-intensive task

[Diagram showing control flow of decoupling]
Challenge 2: How to Manage Jobs?

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

How does Sol perform:
1. compared to existing engines?
2. across design space?
3. under uncertainties?

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

Deployment over 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*

**Control-plane benefits**
(2.6x on avg.)

**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)

- Spark
- Sol

3.9x improvement on average

High-bandwidth setting (10 Gbps)

- Spark
- Sol

1.3x improvement on average
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