Random Forests over normalized data in CPU-GPU DBMSes

Zachary Huang, Pavan Kalyan Damalapati, Rathijit Sen, Eugene Wu
Data Science Institute, Columbia University, Microsoft Gray Systems Lab

ML in databases is popular

Microsoft Azure

Using Azure Machine Learning with SQL Data Warehouse

Posted on November 24, 2015

Sahaj Saini
Program Manager, SQL Engineering

Azure Machine Learning allows you to build predictive models using data from your Azure SQL Data Warehouse database and other sources.

Azure Machine Learning is a powerful cloud-based predictive analytics service that makes it possible to

Google Cloud

What is BigQuery ML?

BigQuery ML lets you create and execute machine learning models using GoogleSQL queries. BigQuery ML democratizes machine learning by letting SQL practitioners build models using existing SQL tools and skills. BigQuery ML increases development speed by eliminating the need to move data.

aws

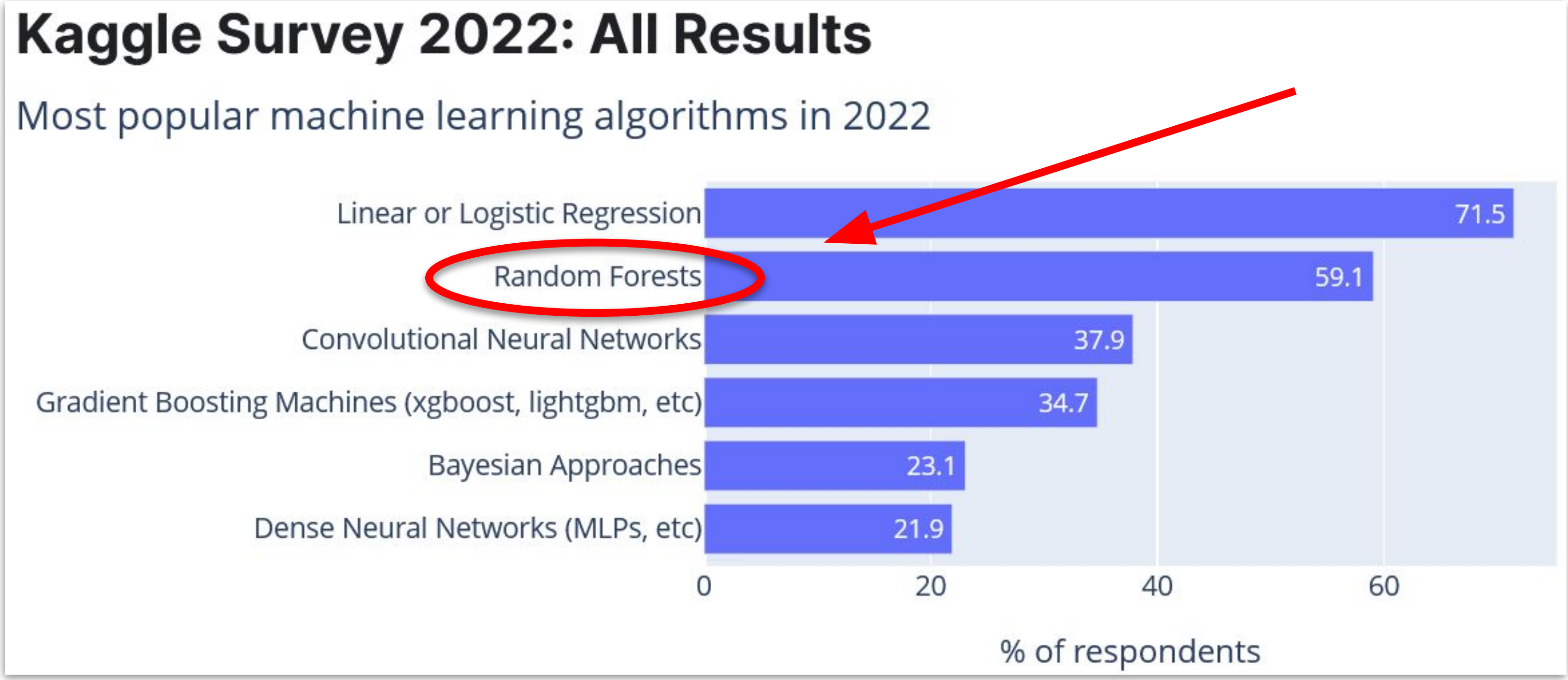
Amazon Redshift ML

Create, train, and deploy machine learning (ML) models using familiar SQL commands

Read the overview guide

Amazon Redshift ML makes it easy for data analysts and database developers to create, train, and apply machine learning models using familiar SQL commands in Amazon Redshift data warehouses. With Redshift

Random forests



GPU for Random Forests

NVIDIA DEVELOPER

Join

Data Science

Accelerating Random Forests Up to 45x Using cuML

Feb 25, 2021 0 Like Discuss (1)

By Vishal Mehta

LightGBM

LightGBM GPU Tutorial

The purpose of this document is to give you a quick step-by-step tutorial on GPU training.

For Windows, please see GPU Windows Tutorial.

We will use the GPU instance on Microsoft Azure cloud computing platform for demonstration, but you can use any machine with modern AMD or NVIDIA GPUs.

dmlc XGBoost

XGBoost GPU Support

This page contains information about GPU algorithms supported in XGBoost.

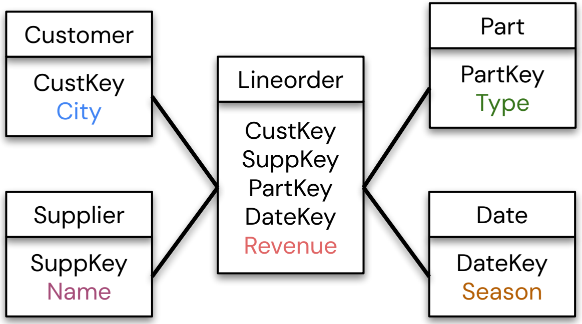
Note

CUDA 11.0, Compute Capability 5.0 required (See this list to look up compute capability of your GPU card.)

CUDA Accelerated Tree Construction Algorithms

Most of the algorithms in XGBoost including training, prediction and evaluation can be accelerated with CUDA-capable GPUs.

Challenge: DB–ML mismatch



City	Type	Season	Name	Revenue
...	...	...	...	...

feature X y

DB: stores **many** tables

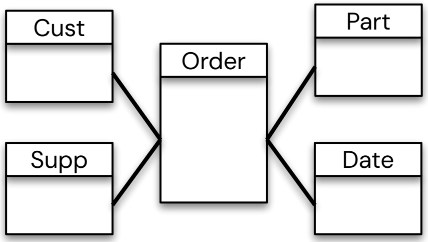
ML: needs **single** table

Google Cloud aws Microsoft Azure

LightGBM dmlc XGBoost

Challenge: DB–ML mismatch

Current solution



DB

Google Cloud aws
Microsoft Azure Azure Databricks

```
CREATE TABLE train AS
SELECT City, Type, Season, Name, Revenue
FROM Customer Join Lineorder ON CustKey
    Join Part ON PartKey
    Join Date ON DateKey
    Join Supplier ON SuppKey;
```

SQL: Join, materialize

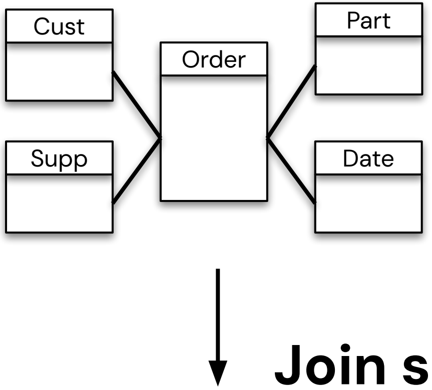
City	...	Revenue
...	...	...

ML

LightGBM dmlc XGBoost

Challenge: Join Results Exceed GPU Memory

- Join significantly increases size



SSB SF=10
DB size = 6GB
after encoding

Join size >15 GB

City	...	Revenue
...	...	...

- Common GPU memory is limited

GPU	memory
V100	16/32 GB
P100	16 GB
K80	12 GB
P4	8 GB

In contrast, CPUs can easily support 100GB to several TBs memory

Our Approach

Algorithm:

- Translate Random Forests Training as Join Aggregations
- Push down Aggregation before Join

System:

- Explore design choices on CPU-GPU DBMSes

We will start with the background of Random Forests Training

Background: Random Forests Training

Feature Target

R

F	Y
1	1
1	1
2	1
2	2
3	3
3	4

Random Forests are ensemble of trees. For each tree:

Tree-split Problem: Find the best split point F to minimize the sum of variance of Y.

var([1,1,1,2,3,4])
= 1.6

Background: Random Forests Training

R

F	Y
1	1
1	1
2	1
2	2
3	3
3	4

Tree-split Problem: Find the best split point F to minimize the sum of variance of Y.

F > 1

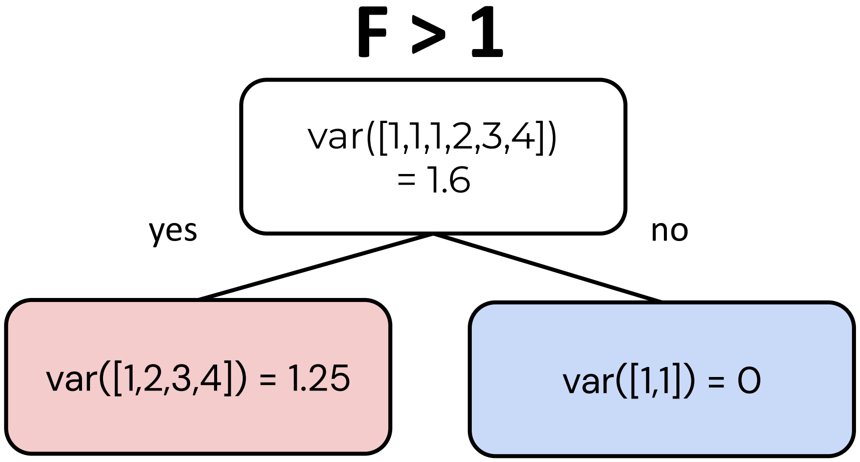
var([1,1,1,2,3,4])
= 1.6

Background: Random Forests Training

R

F	Y
1	1
1	1
2	1
2	2
3	3
3	4

Tree-split Problem: Find the best split point F to minimize the sum of variance of Y.



Sum of variance = 0 + 1.25 =<u>1.25</u>

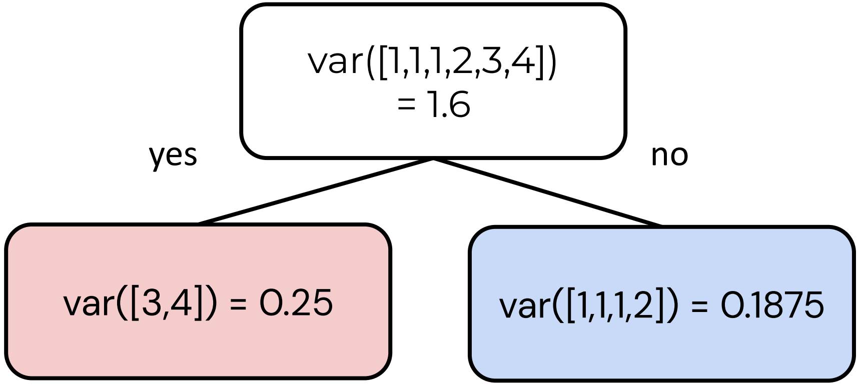
Background: Random Forests Training

R

F	Y
1	1
1	1
2	1
2	2
3	3
3	4

Tree-split Problem: Find the best split point F to minimize the sum of variance of Y.

F > 2



Sum of variance = 0.1875 + 0.25 = <u>0.4375</u>

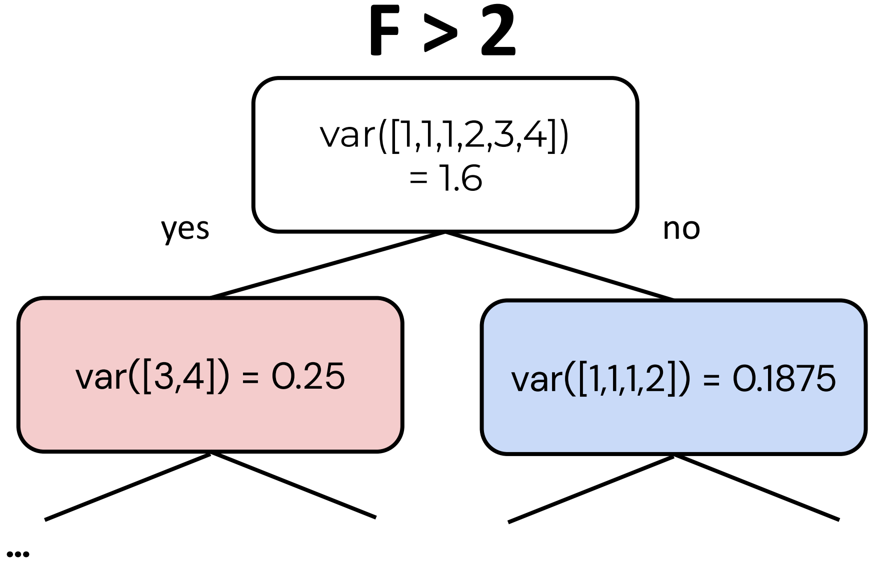
F > 2 is the best split point!

Background: Random Forests Training

We have splitted the base relation.

Recursively split the splitted relation and find the best splits.

F	Y
3	3
3	4



F	Y
1	1
1	1
2	1
2	2

Core Computation: Variance

Variance(Y) = $\Sigma Y^2 - (\Sigma Y)^2 / \Sigma 1$
Based on sum of squares, sum, count

Lightweight Postprocessing
To find the feature with the best split

```
SELECT F, – S * S / C AS variance
FROM (
    SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
    FROM R
    GROUP BY F
    )
GROUP BY F
ORDER BY variance DESC LIMIT 1;
```

Core Aggregation

*The query for tree is more complex by computing the **sum** of variances. Please refer to the paper for the detailed queries.

Core Computation: Variance

$\text{Variance}(Y) = \Sigma Y^2 - (\Sigma Y)^2 / \Sigma 1$

Based on sum of squares, sum, count

Core Aggregation

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R
GROUP BY F
```

*The query for tree is more complex by computing the **sum** of variances. Please refer to the paper for the detailed queries.

Core Computation: Variance

Variance(Y) = ΣY^2 - $(\Sigma Y)^2$ / $\Sigma 1$

Based on sum of squares, sum, count

How to speed up Core Aggregation computation over join?

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R JOIN F ON J
GROUP BY F
```

*The query for tree is more complex by computing the **sum** of variances. Please refer to the paper for the detailed queries.

Our Solution: Aggregation Pushdown

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R JOIN F ON J
GROUP BY F
```

F (fact)

Y	J
1	1
2	1
3	2
4	3

R (dim)

F	J
1	1
1	2
2	3

Our Solution: Aggregation Pushdown

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R JOIN F ON J
GROUP BY F
```

As large as the fact table

F (fact)

Y	J
1	1
2	1
3	2
4	3

R (dim)

F	J
1	1
1	2
2	3

$R \bowtie F$

Y	J	F
1	1	1
2	1	1
3	2	1
4	3	2

Our Solution: Aggregation Pushdown

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R JOIN F ON J
GROUP BY F
```

F (fact)

Y	J
1	1
2	1
3	2
4	3

R (dim)

F	J
1	1
1	2
2	3

Our Solution: Aggregation Pushdown

```
CREATE TABLE msg AS
SELECT J, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM F GROUP BY J;
```

F (fact)

Y	J
1	1
2	1
3	2
4	3

Msg (partial agg)

J	(C,S,Q)
1	(2,3,5)
2	(1,3,9)
3	(1,4,16)

R (dim)

F	J
1	1
1	2
2	3

Our Solution: Aggregation Pushdown

```
SELECT F, SUM(C) AS C, SUM(S) AS S, SUM(Q) AS Q
FROM R JOIN msg ON J
GROUP BY F
```

Bounded by dimension table

F (fact)

Y	J
1	1
2	1
3	2
4	3

Msg (partial agg)

J	(C,S,Q)
1	(2,3,5)
2	(1,3,9)
3	(1,4,16)

R (dim)

F	J
1	1
1	2
2	3

$R \bowtie Msg$

F	(C,S,Q)
1	(3,6,14)
2	(1,4,16)

Our Solution: Aggregation Pushdown

```
SELECT F, SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM R JOIN F ON J
GROUP BY F
```

Aggregation over Join

↓ Query rewrite

```
CREATE TABLE msg AS
SELECT SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM F GROUP BY J;
```

Partial Aggregate from Fact Table

```
SELECT F, SUM(C) AS C, SUM(S) AS S, SUM(Q) AS Q
FROM R JOIN msg ON J
GROUP BY F
```

Sum Partial Aggregate by Dim Table

System Design

Problem: Train random forests directly in CPU-GPU DBMSes.
We can use CPU memory when data doesn't fit in GPU.

<u>Where to place data and execute query for CPU & GPU?</u>

Aggregation for random forests

Msg from Fact table

```
CREATE TABLE msg AS
SELECT SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM F GROUP BY J;
```

Simple query over big data
Better on GPU

Lightweight Postprocessing by Dim table

```
SELECT F, Q - S * S / C AS variance
FROM (

        SELECT F, SUM(C) AS C, SUM(S) AS S, SUM(Q) AS Q
        FROM R JOIN msg ON J
        GROUP BY F

     )
GROUP BY F
GROUP BY variance DESC LIMIT 1;
```

Complex query over small data
Better on CPU
Inter-query Parallelism

Aggregation for random forests

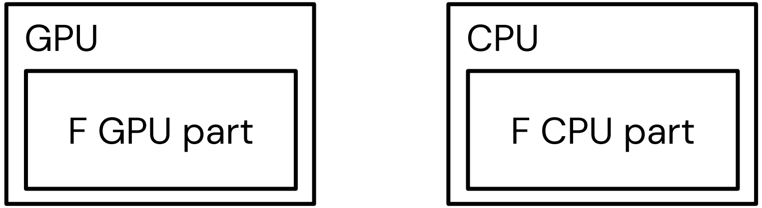
```sql
CREATE TABLE msg AS
SELECT SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM F GROUP BY J;
```

Msg from Fact table

Simple query over big data
Better on GPU

What if F doesn't fit in GPU memory?

Store only a horizontal partition of F fit in GPU.



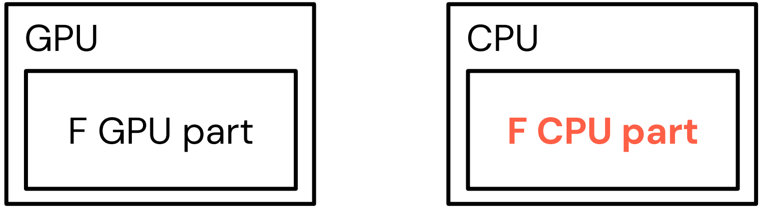
Aggregation for random forests

```
CREATE TABLE msg AS
SELECT SUM(1) AS C, SUM(Y) AS S, SUM(Y * Y) AS Q
FROM F GROUP BY J;
```

Msg from Fact table

Simple query over big data
Better on GPU

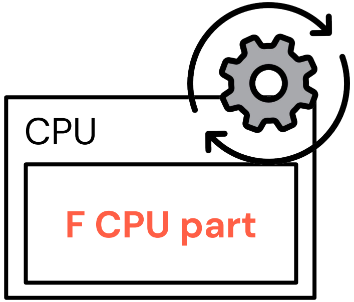
How to execute query over CPU part of F?



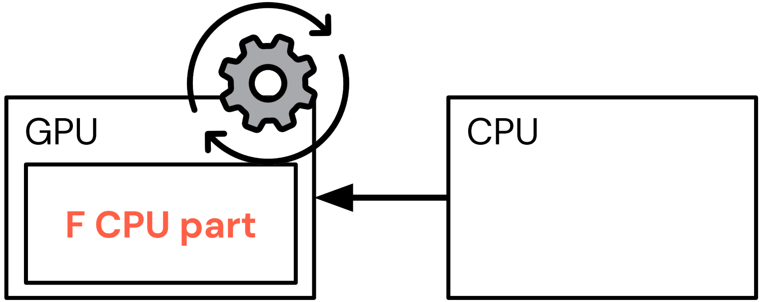
Aggregation for random forests

How to execute query over CPU part of F?

1. Directly use the CPU to execute query



2. <u>Transfer</u> through PCIe from CPU to GPU, then execute query using GPU



Repo Demo

Implementation & Experiment

We use off-the-shelf dataframes to integrate with the Python ecosystem.
Aggregation pushdown achieved through query rewriting.

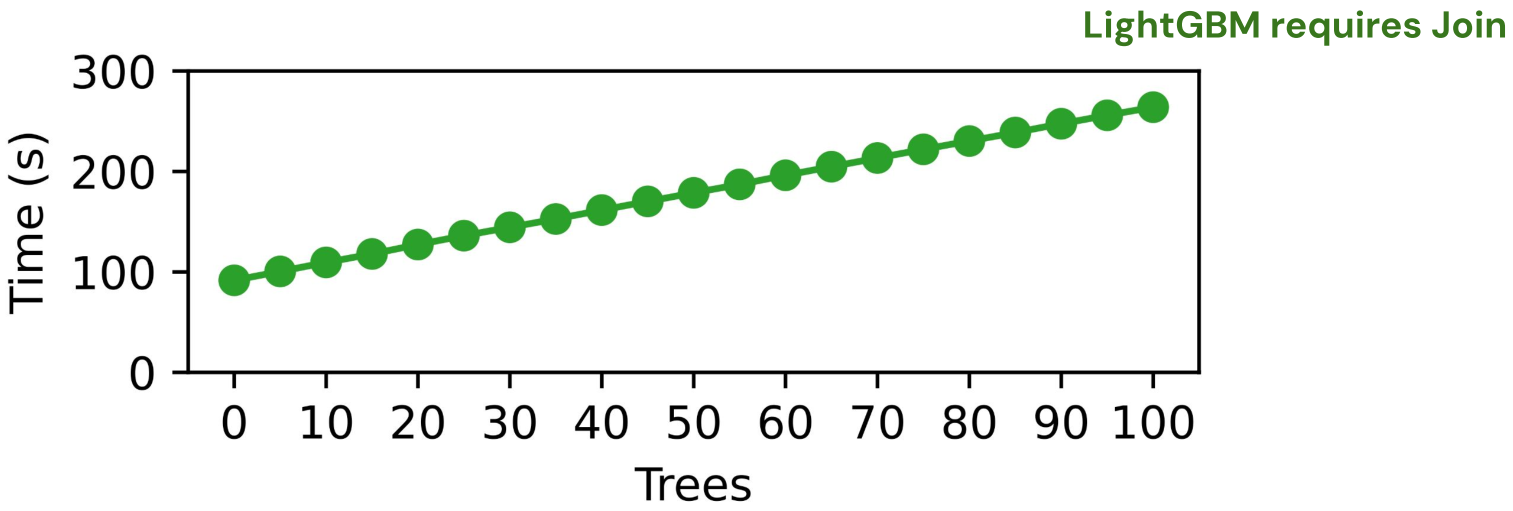
	Data	Query Execution
CPU	Pandas DF	DuckDB
GPU	CuDF	CuDF

*Pandas DF is not efficient (single thread), so use DuckDB.
* cuDF is not efficient as it materializes intermediates, and our results can be improved using backends like CUDA based on Crystal

Hardware: V100 GPU (16 GB) connected via PCIe3 to CPU (112 GB)
Task: Random forests (100 trees, 8 max leaves, 0.5 sample) for SSB dataset

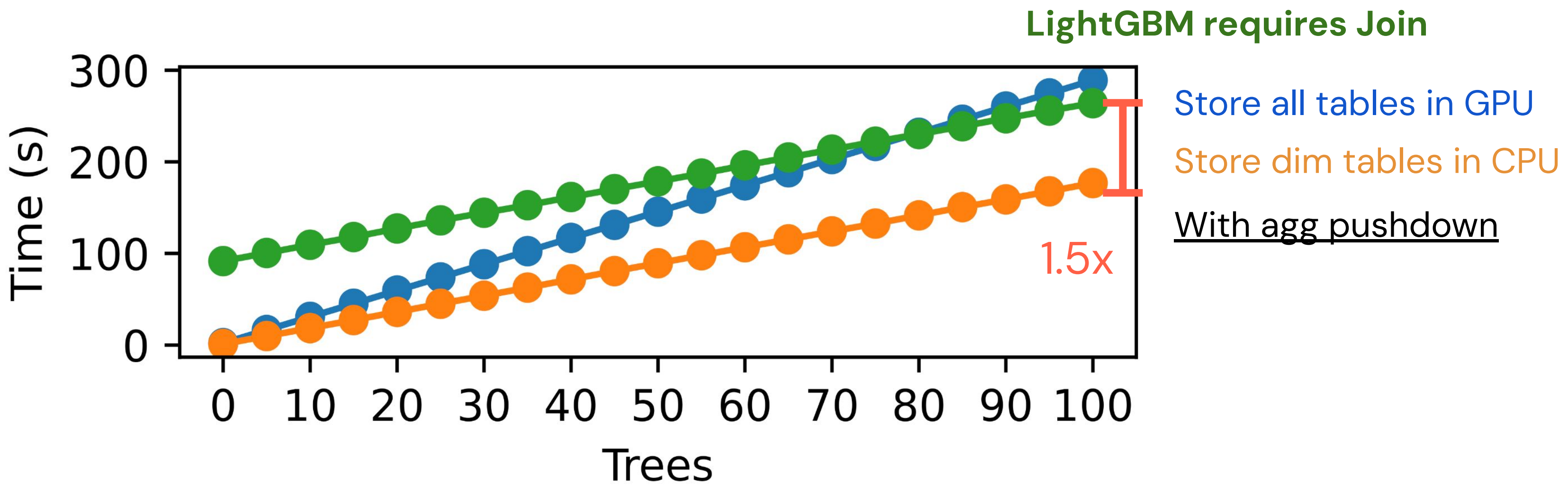
When the database fit in GPU

SSB SF=10 (6.5GB after encoding)



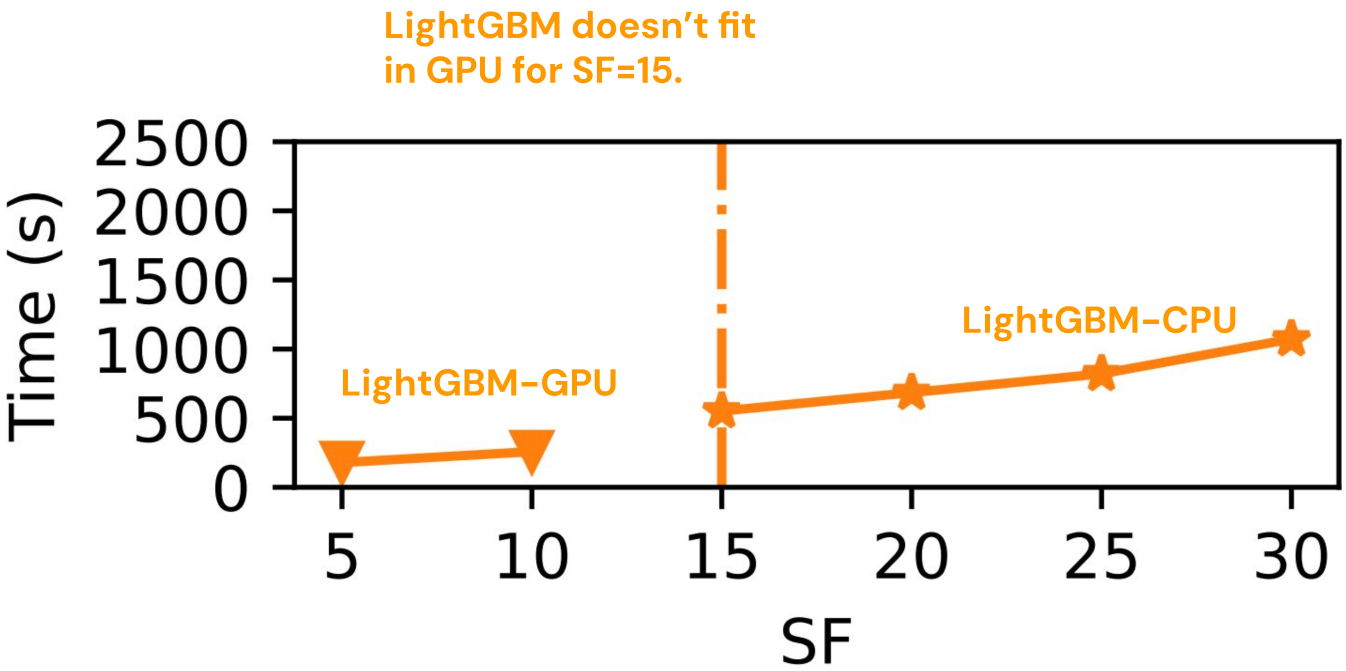
When the database fit in GPU

SSB SF=10 (6.5GB after encoding)



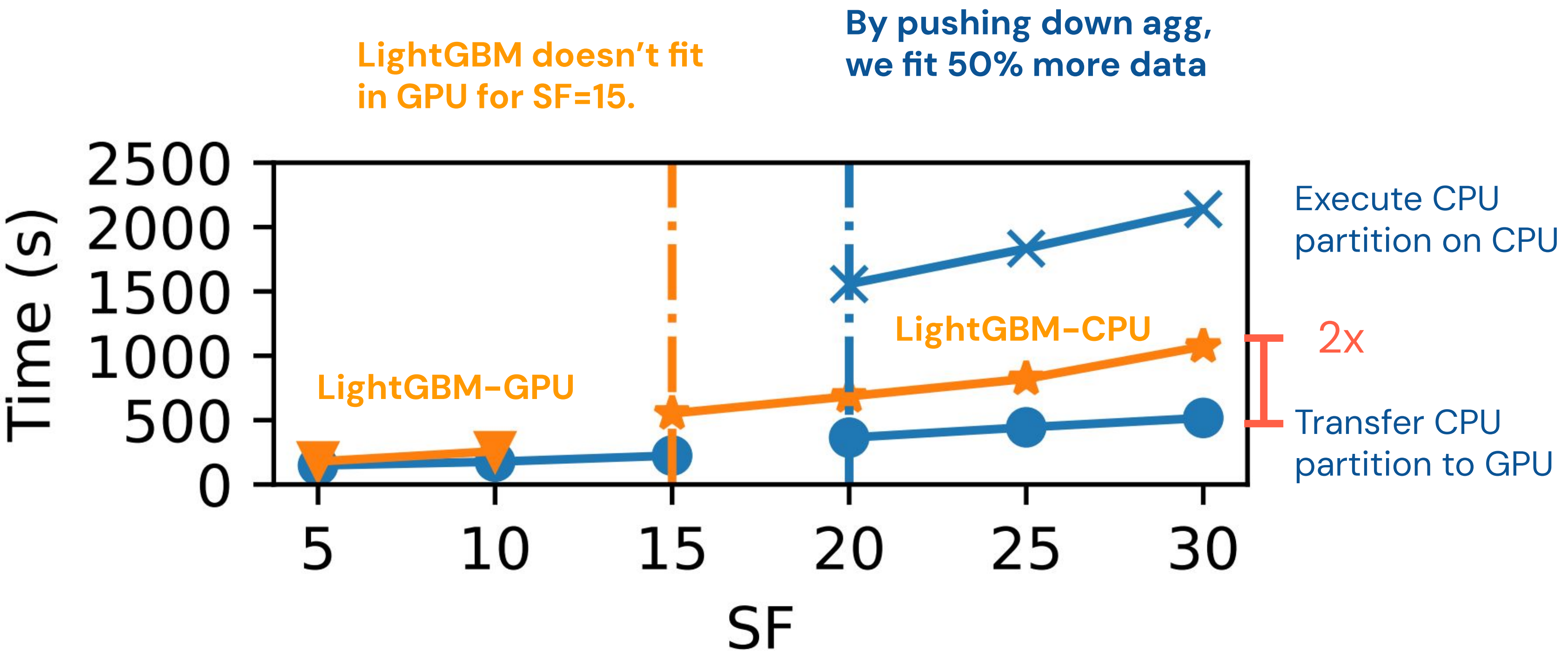
When the database doesn’t fit in GPU

Vary SSB SF



When the database doesn't fit in GPU

Vary SSB SF



Thank you!

Repo

Demo