# GHive: Accelerating Analytical Query Processing in Apache Hive via CPU-GPU Heterogeneous Computing

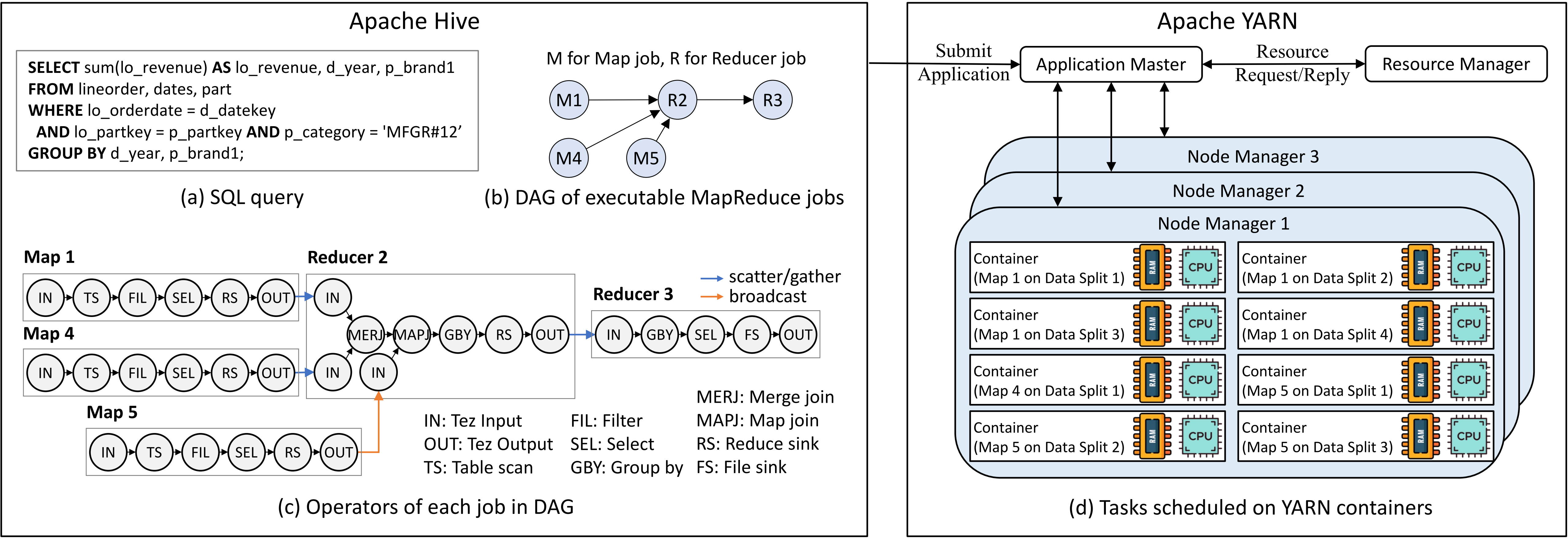
Presenter: Haotian Liu

Email: dbgroup@sustech.edu.cn

# Introduction to Hive

- Support distributed big data analytics on a massive scale
  - Run big data analytical queries with MapReduce paradigm
- Provide a SQL-like interface on top of Hadoop
  - Avoid implementing the details of low-level MapReduce jobs
- Widely used by many organizations
  - Facebook, Google, Huawei, etc.
- Many of our analytical queries are run with Hive.

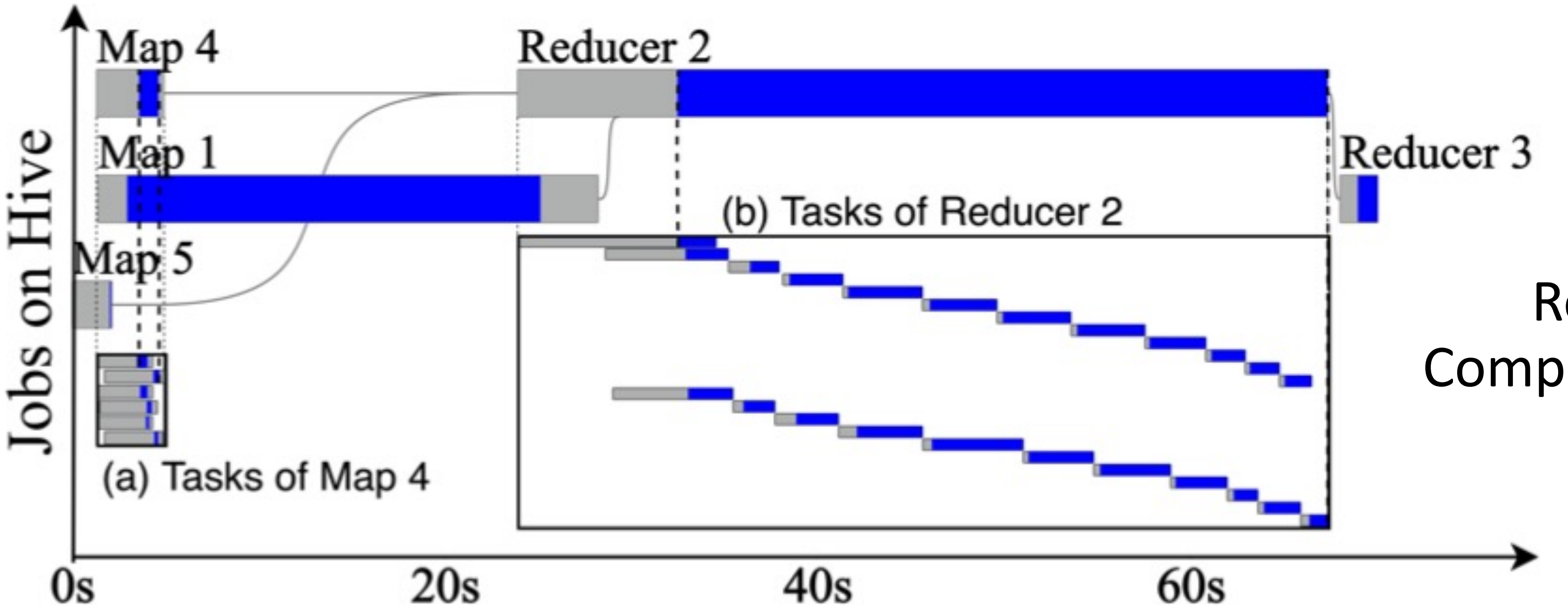
# Query Processing in Apache Hive




- The Map and Reducer **jobs** defines the specific operator sequences.
- Each job contains several **tasks** (according to data size), which will be scheduled.

# Performance Profiling

Map 4
Computation ratio:
**28.7%**

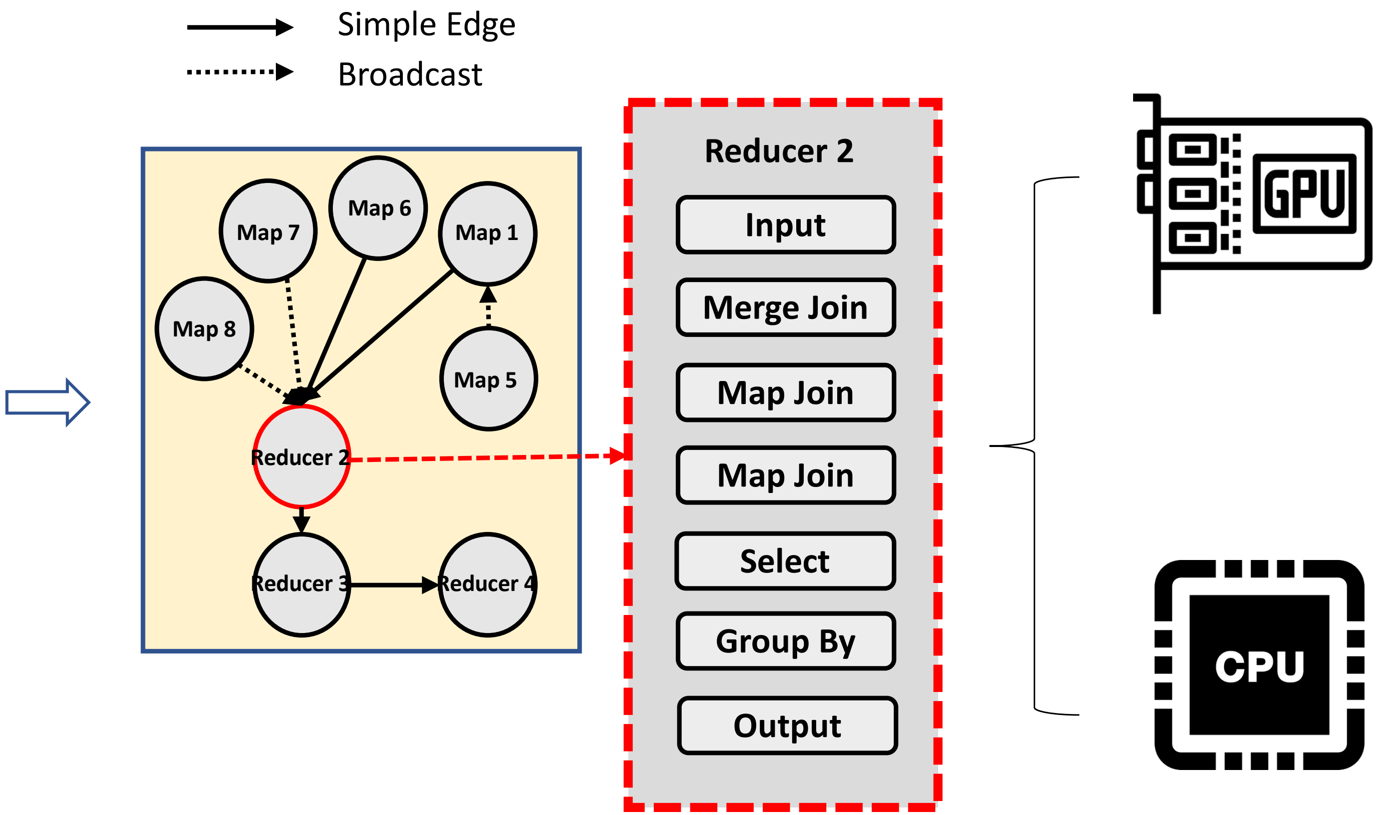



Reducer 2
Computation ratio:
**80.3%**

We can classify the jobs into two categories: **compute-bound** and **I/O-bound**!

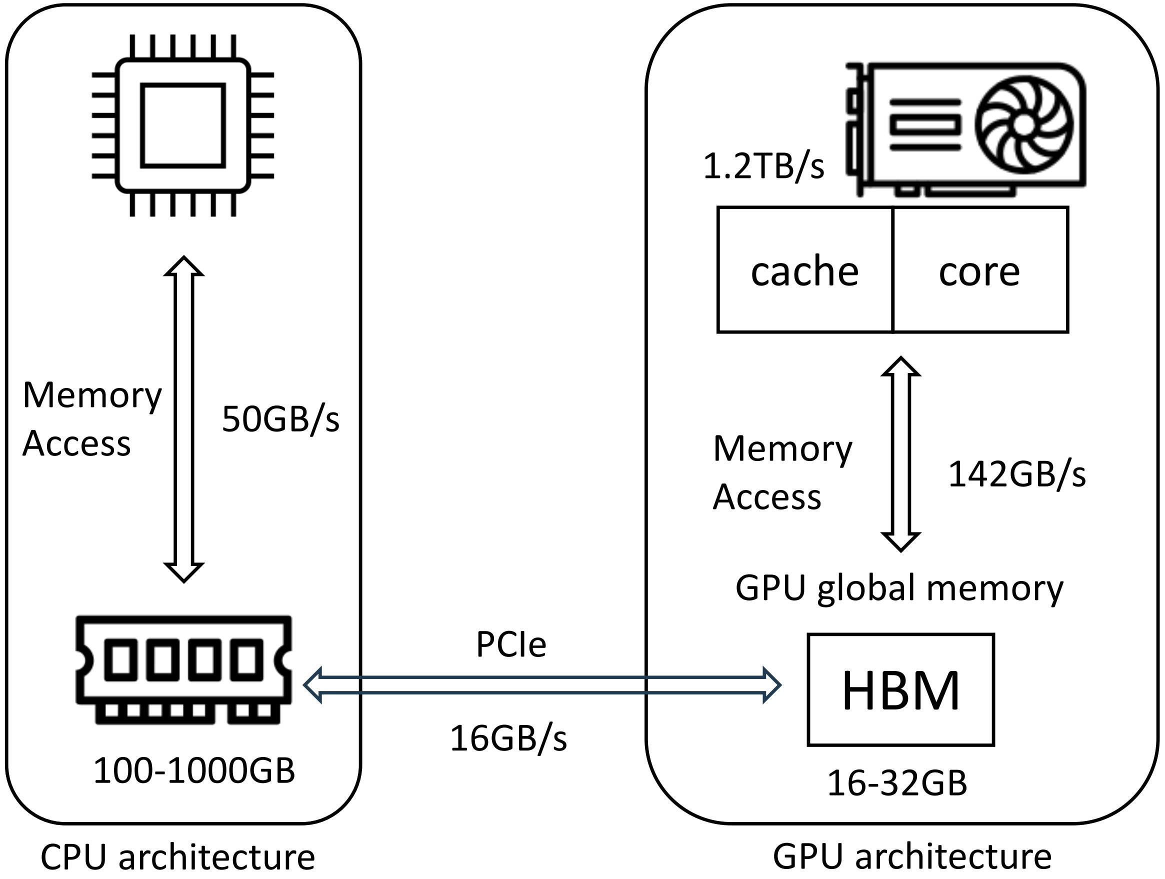
# Motivation of GHive

```sql
select
   d_year, s_city, p_brand1,
   sum(lo_revenue - lo_supplycost) as profit
from
   dates, customer, supplier, part, lineorder
where
   lo_custkey = c_custkey
   and lo_suppkey = s_suppkey
   and lo_partkey = p_partkey
   and lo_orderdate = d_datekey
   and c_region = 'AMERICA'
   and s_nation = 'UNITED STATES'
   and (d_year = 1997 or d_year = 1998)
   and p_category = 'MFGR#14'
group by
   d_year, s_city, p_brand1
order by
   d_year, s_city, p_brand1;
```




- The Hive is deployed on a shared cluster, where GPUs are not fully utilized
- GPU has great compute power to accelerate compute-bound tasks.

# GPU vs CPU



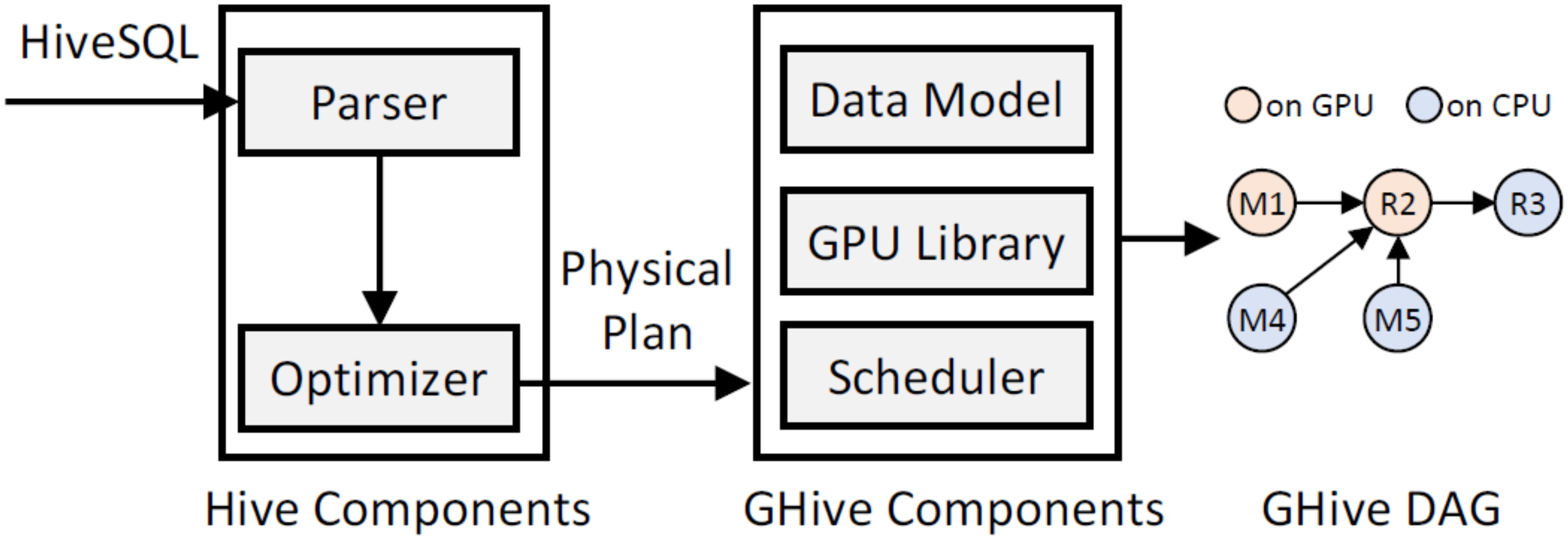

## GPU pros:

- GPU has immense computational power
- GPU memory has high bandwidth

## GPU cons:

- GPU memory has small capacity
- Loading data from main memory is slow

# GHive Architecture




Three key designs:

- Compact data model: ***gTable***
- GPU-based SQL operator library: ***Panda***
- Hardware-aware job placement scheme

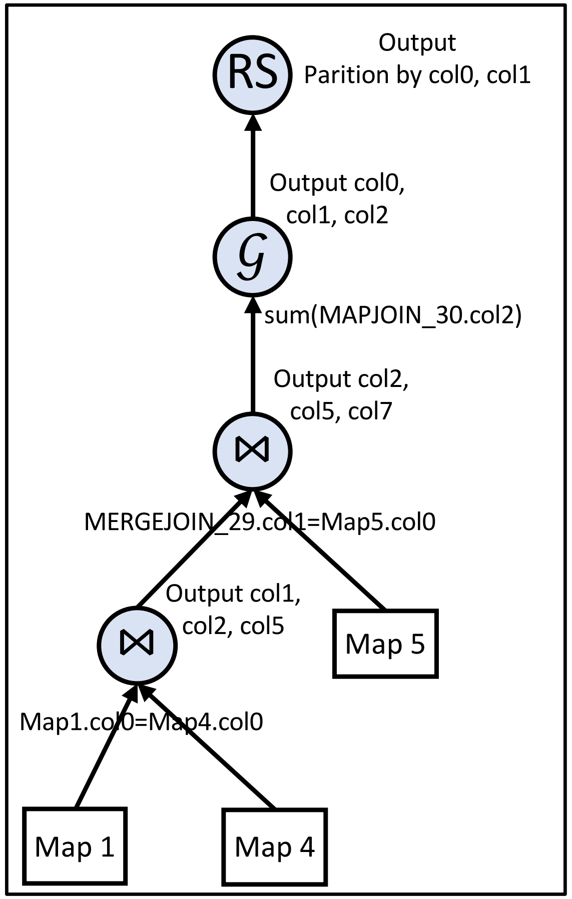
# Plan Parser

The Plan Parser extracts

1. Cardinality information for hardware-aware placement.
2. Operator (type, order) information for moving them to GPU.

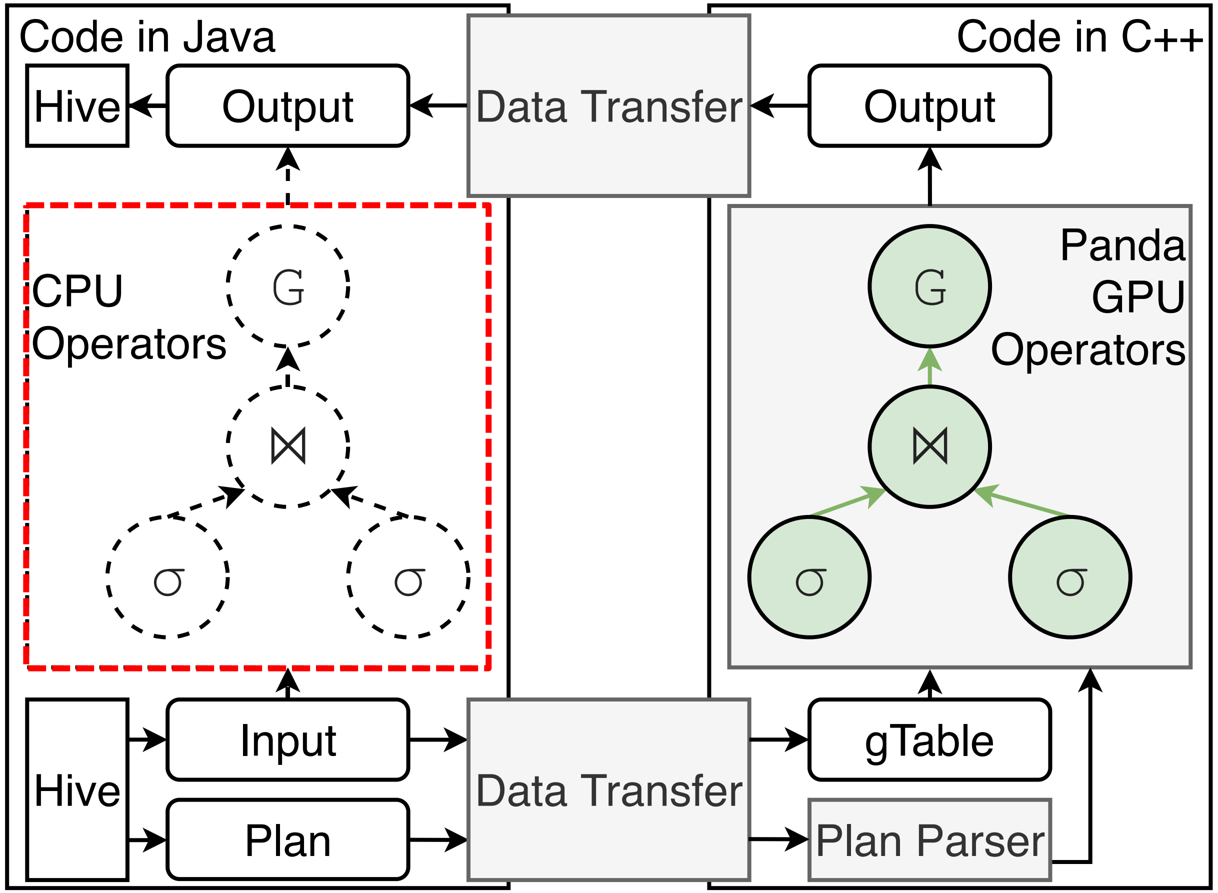
```
Reducer 2 [SIMPLE_EDGE]
 SHUFFLE [RS_17]
  PartitionCols:_col0, _col1
  Group By Operator [GBY_16]
  (rows=14517574 width=373)
Output:["_col0","_col1","_col2"],
aggregations:["sum(_col2)"],keys:_col7,
_col5
    Map Join Operator [MAPJOIN_30]
     (rows=14517574 width=373)
Conds:MERGEJOIN_29._col1=RS_40._col0
(Inner),
    HybridGraceHashJoin:true,
Output:["_col2","_col5","_col7"]
    <-Map 5 [BROADCAST_EDGE]
vectorized
    <-Merge Join Operator
[MERGEJOIN_29]
    (rows=13197795 width=373)
Conds:RS_34._col0=RS_37._col0(Inner),
      Output:["_col1","_col2","_col5"]
     <-Map 1 [SIMPLE_EDGE] vectorized
     <-Map 4 [SIMPLE_EDGE] vectorized
```

(a) Plan text generated by Hive



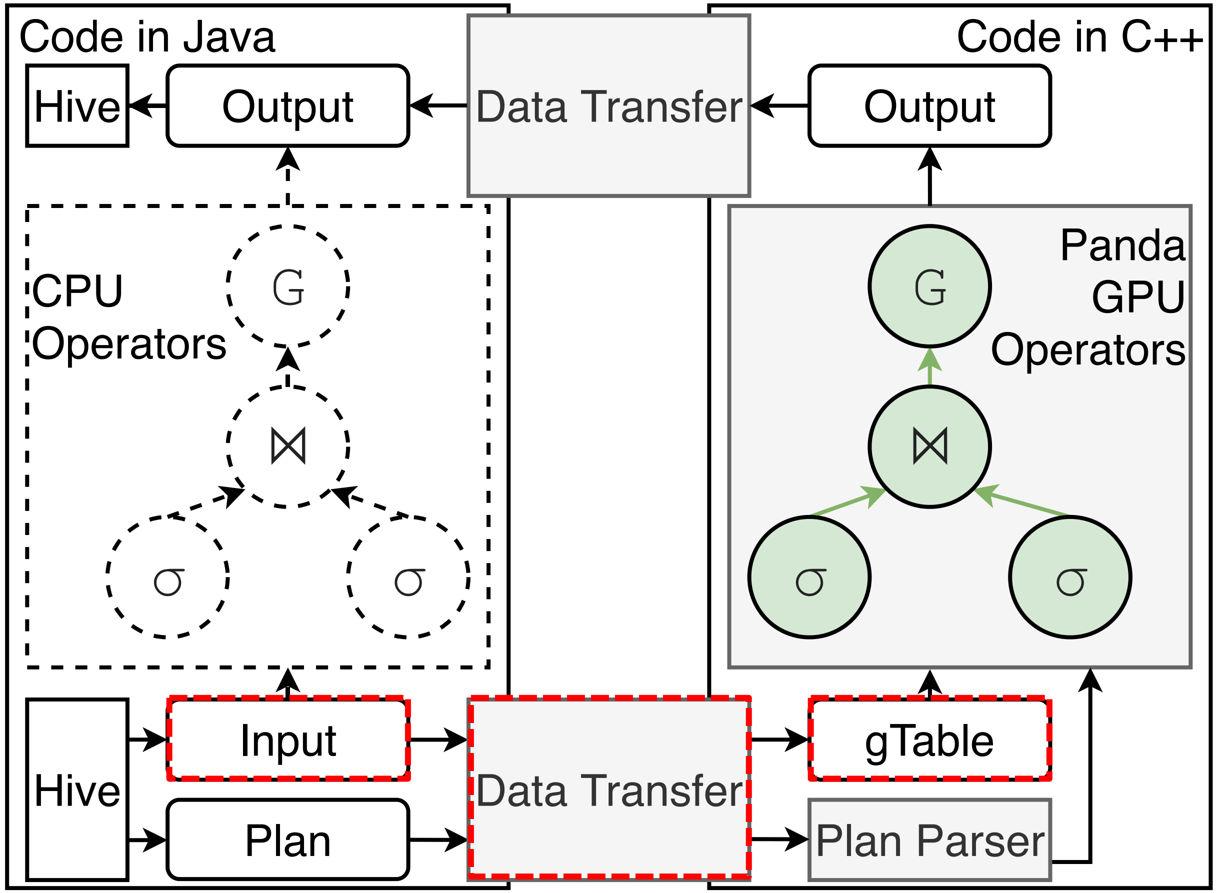

(b) C++-based execution plan tree

# Moving Jobs from CPU to GPU



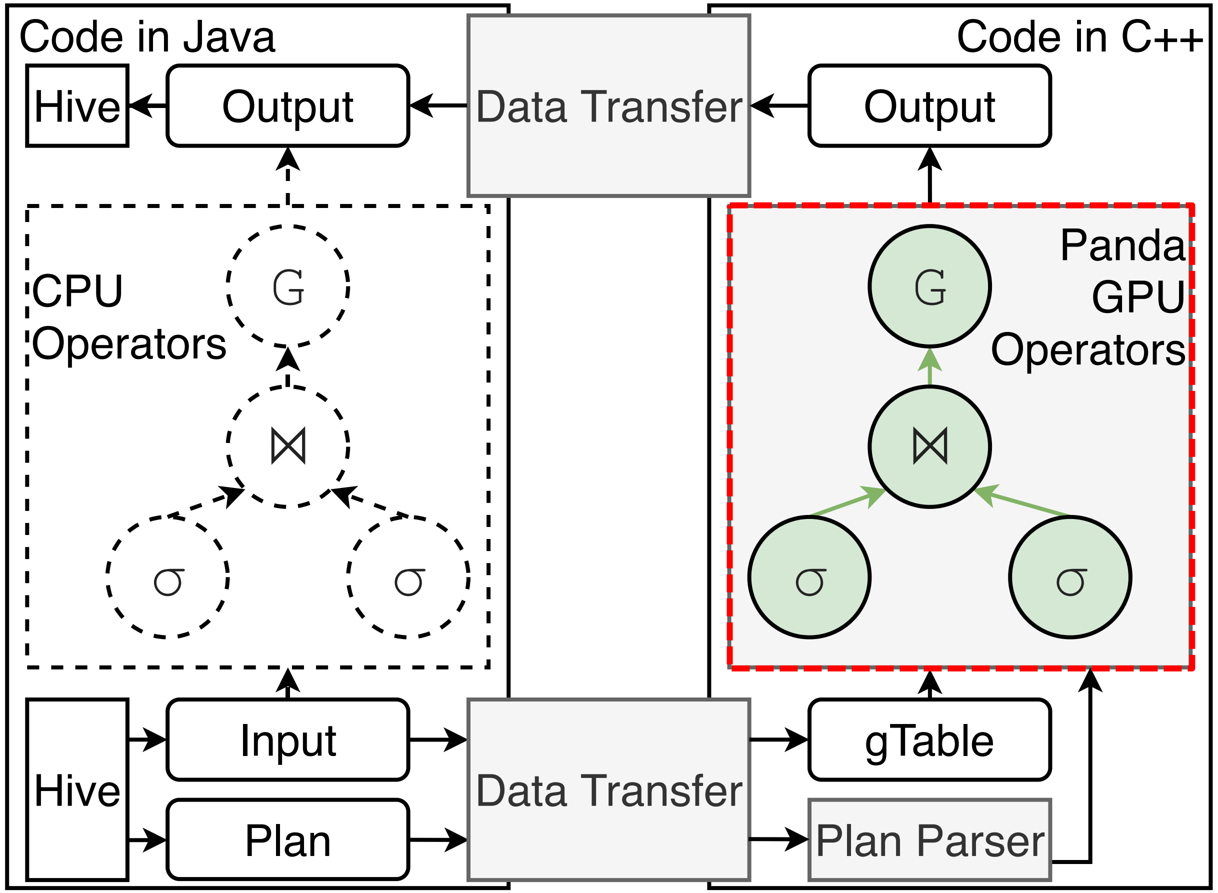

- ➢ Hardware-aware job placement

# Moving Jobs from CPU to GPU




- Hardware-aware job placement
- Data model: gTable.

# Moving Jobs from CPU to GPU



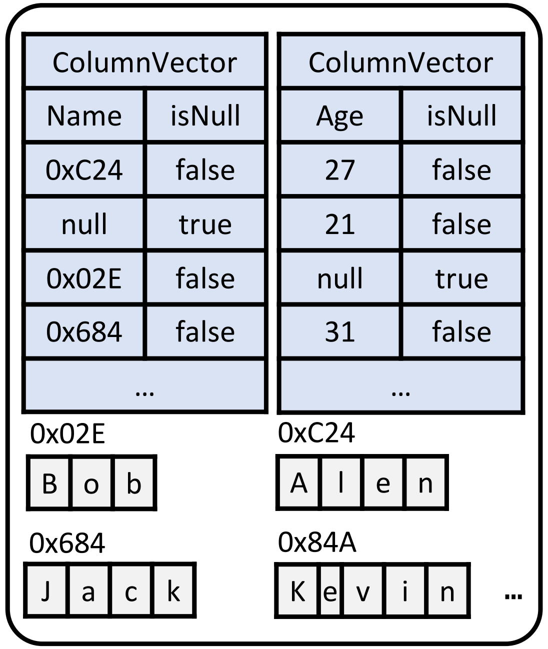

- Hardware-aware job placement
- Data model: gTable.
- GPU operator library: Panda.

# A compact data model: *gTable*

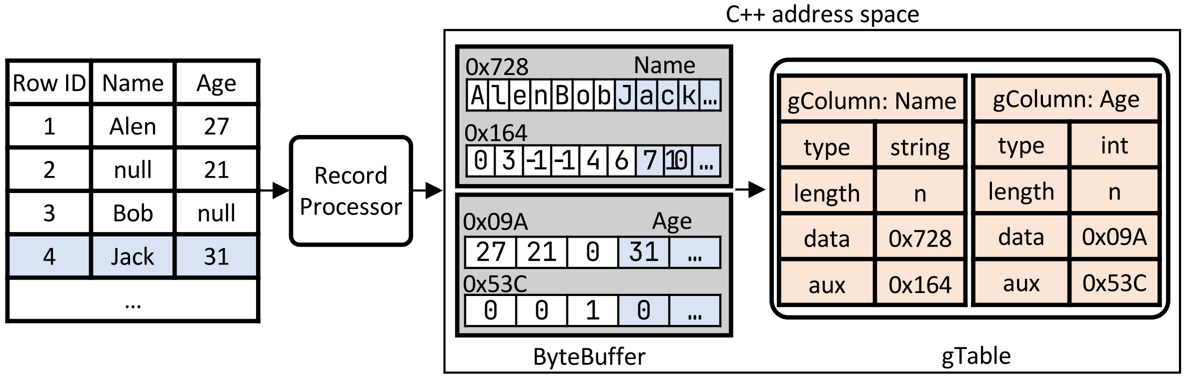
| Row ID | Name | Age | Row ID | Name | Age |
|---|---|---|---|---|---|
| 1 | Alen | 27 | 5 | Kevin | 40 |
| 2 | null | 21 | 6 | Luke | 22 |
| 3 | Bob | null | 7 | Mike | 25 |
| 4 | Jack | 31 | … | | |

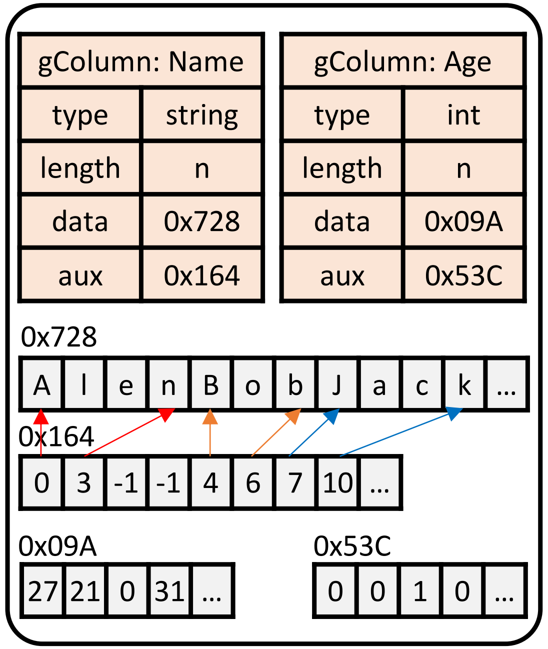
(a) Table T

- Larger batch for higher data parallelism.
- Compact design for variable-length values.
- Direct ByteBuffer to avoid extra copying.



(b) VectorizedRowBatch model

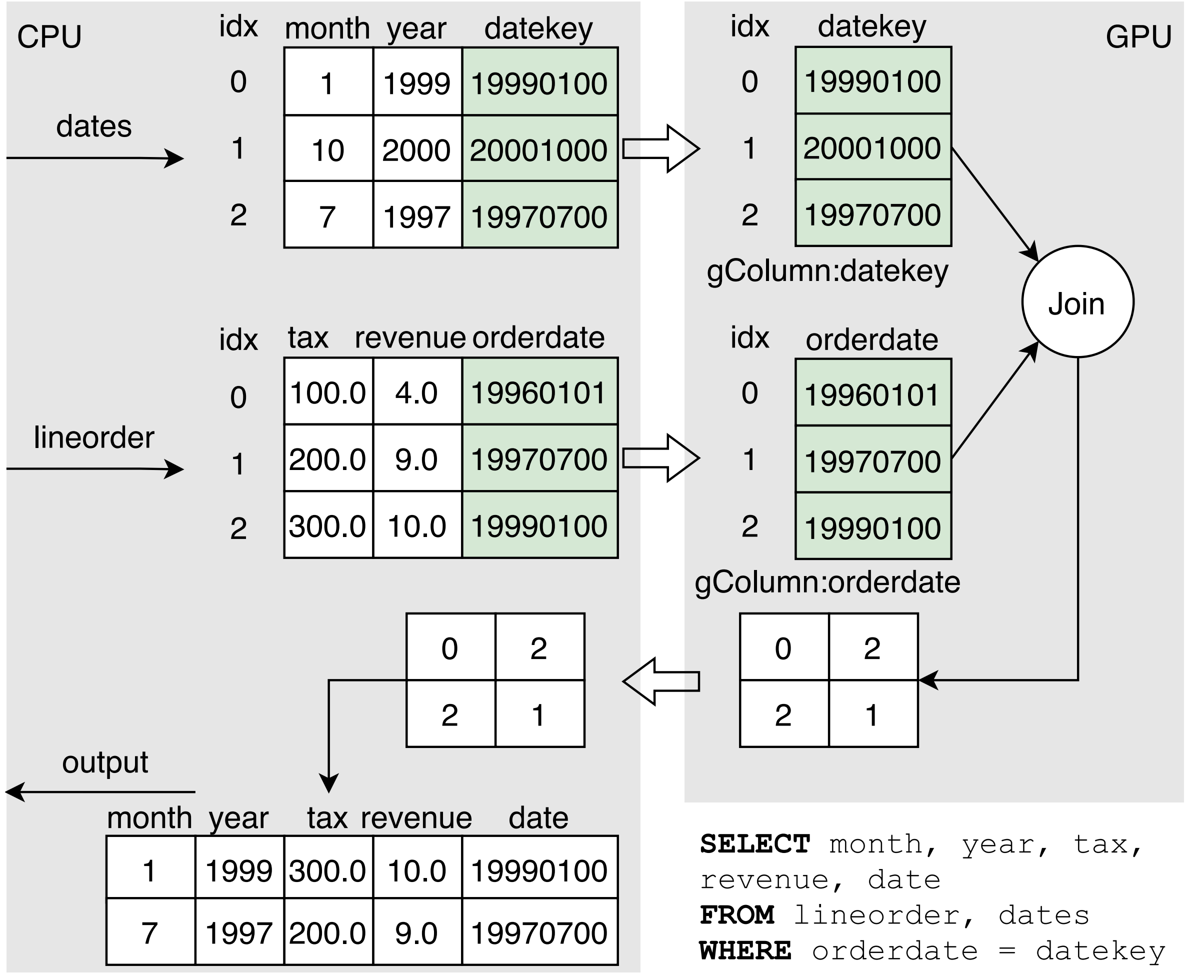


(c) gTable model

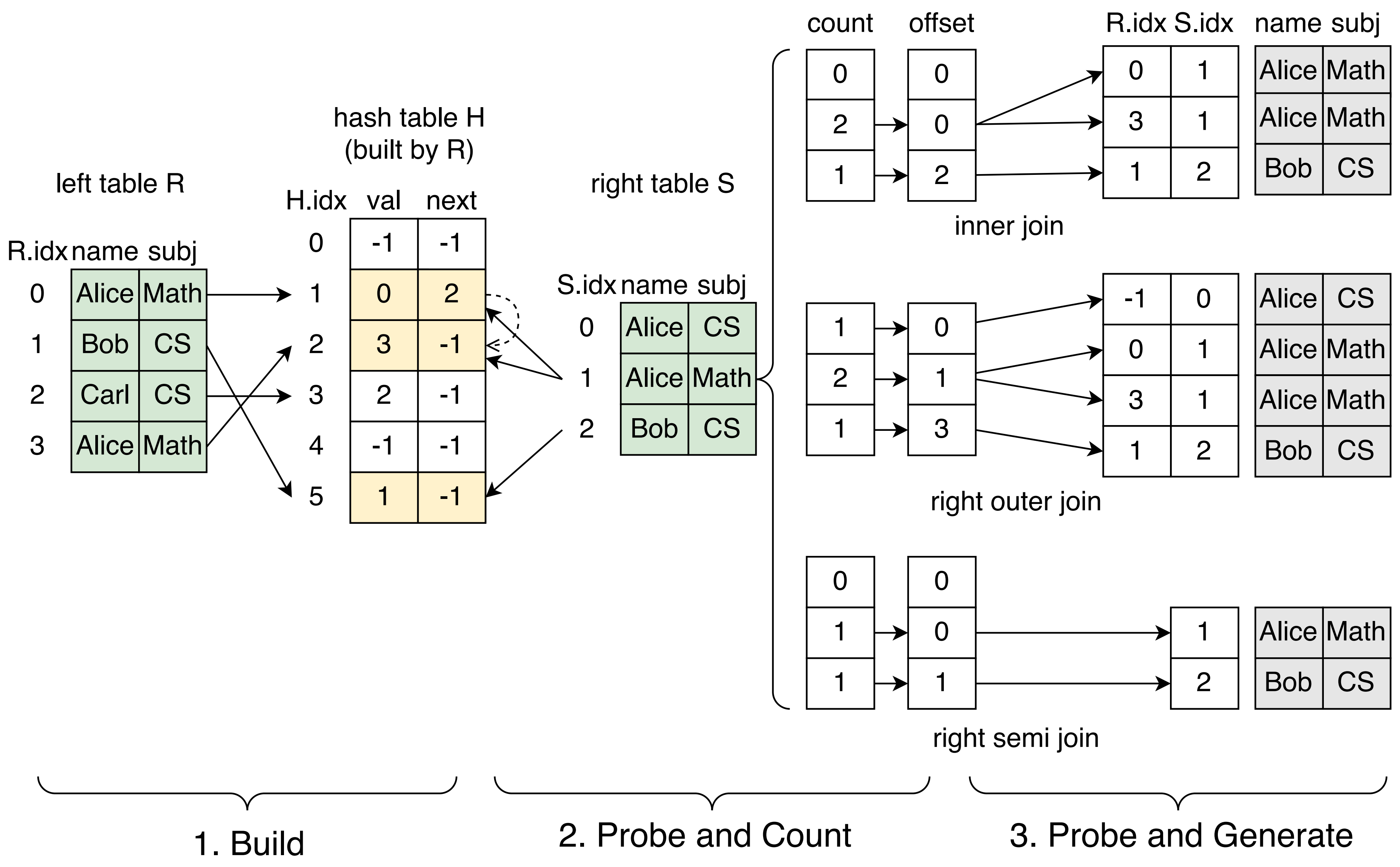
# GPU-based SQL operator library: ***Panda***

- ➢Indexing-based processing model (Late Materialization)




```
SELECT month, year, tax,
revenue, date
FROM lineorder, dates
WHERE orderdate = datekey
```

# The Generality of Panda: Hash join example



# Hardware-aware job placement

- CPU-based cost estimation model

$$T_C(J) = \sum_{\forall op_i \in J} f(op_i, n_i)$$

- CPU-based cost estimation model

$$T_G(J) = T_{pre}(J) + T_{exec}(J) + T_{post}(J).$$

- The policy to place a job to GPU

$$\frac{T_C(J) - T_G(J)}{T_C(J)} \geqslant \theta$$

# Experiment setting

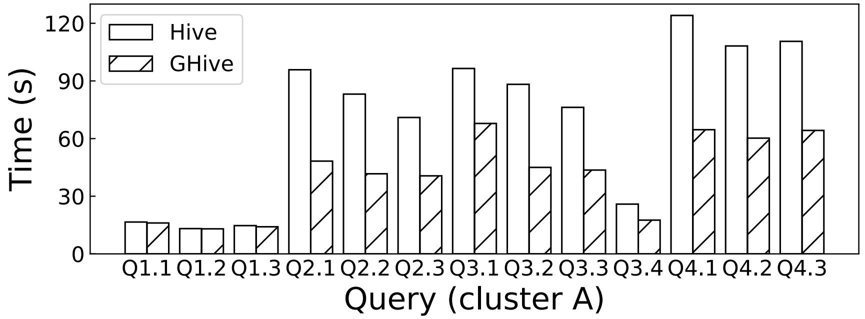
| Hardware | Cluster A | Cluster B |
|---|---|---|
| CPU | Intel Xeon E5-2640 v4 | Intel Xeon Gold 5122 |
| CPU number | 2 | 2 |
| Core number | 20 | 8 |
| CPU memory | 64GB | 512GB |
| GPU | NVIDIA Tesla T4 | NVIDIA TITAN Xp |
| GPU memory | 16GB | 12GB |

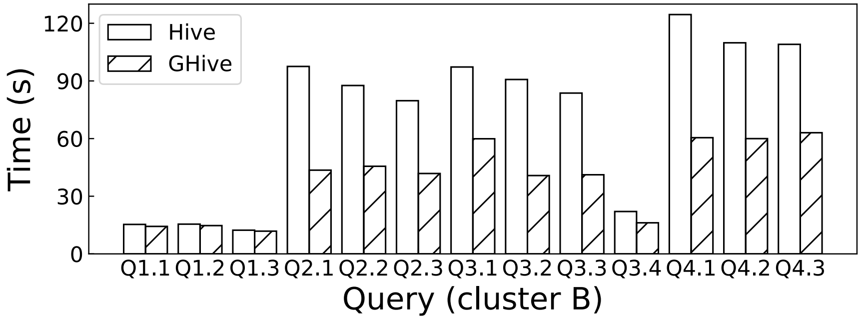
## Evaluation Metrics (Benchmark: Star Schema Benchmark)

- End-to-end time
- GPU Utilization: corresp the potential of collocating with other workloads
- Cost ratio: corresp the operating cost $\mu = \frac{T_{\text{GHive}} \cdot (P^{CPU}_{\text{GHive}} + P^{GPU}_{\text{GHive}})}{T_{\text{Hive}} \cdot P^{CPU}_{\text{Hive}}}$

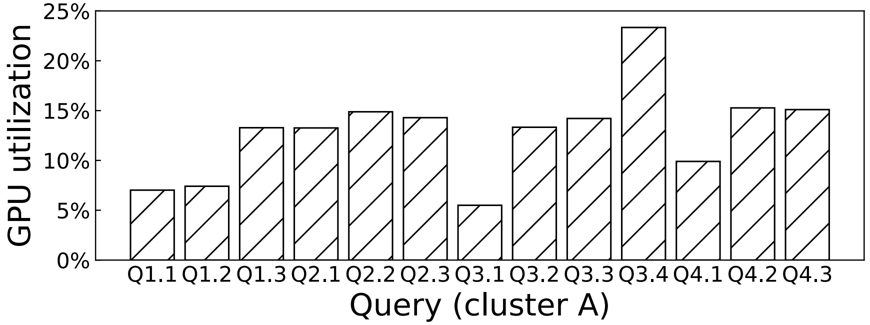
# Experiment result

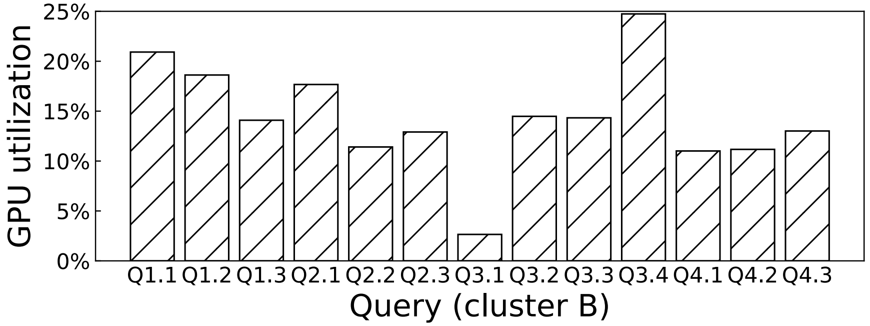
End-to-end time



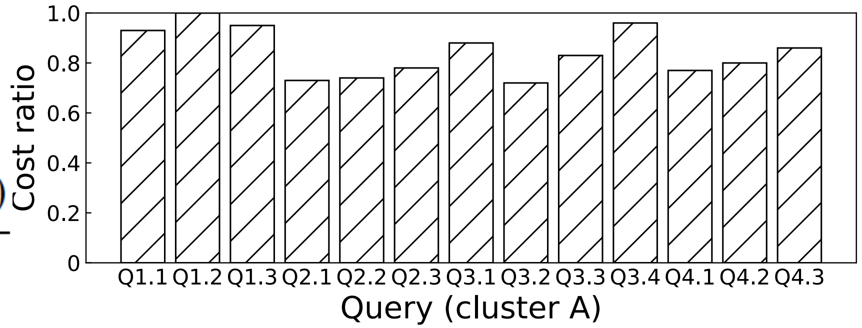

GPU utilization



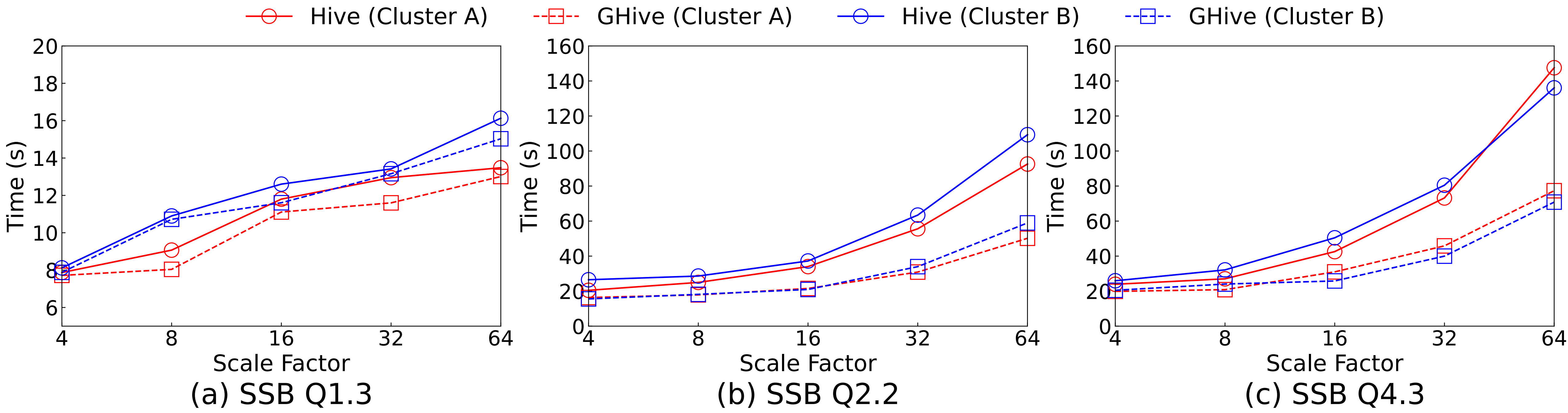




Cost ratio

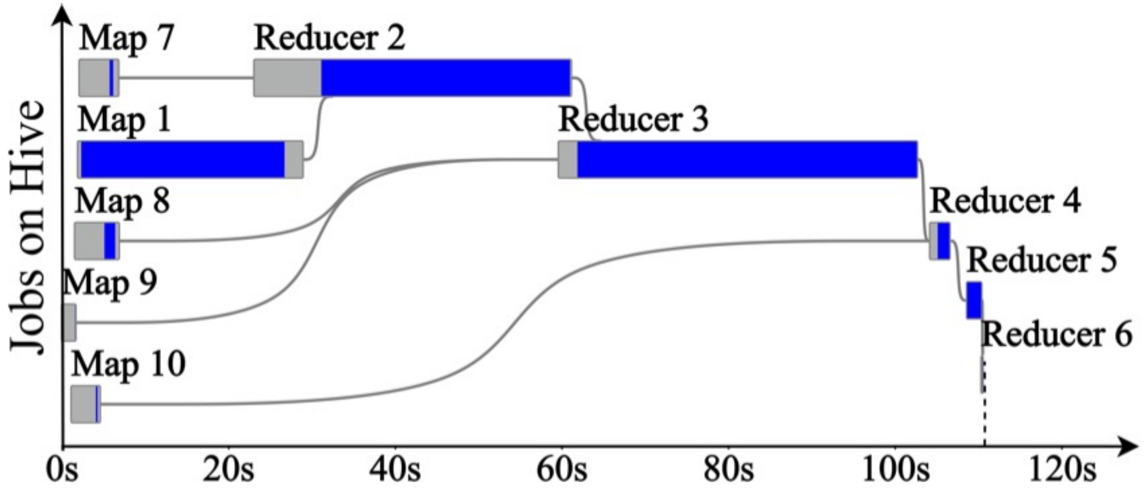
$$\mu = \frac{T_{\text{GHive}} \cdot (P_{\text{GHive}}^{CPU} + P_{\text{GHive}}^{GPU})}{T_{\text{Hive}} \cdot P_{\text{Hive}}^{CPU}}$$

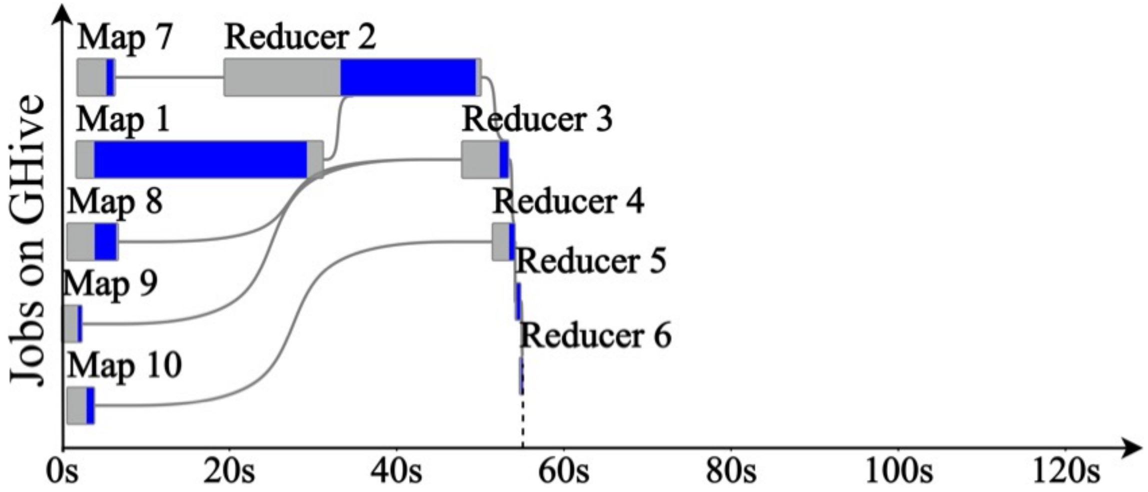



# Effect of Scale Factor

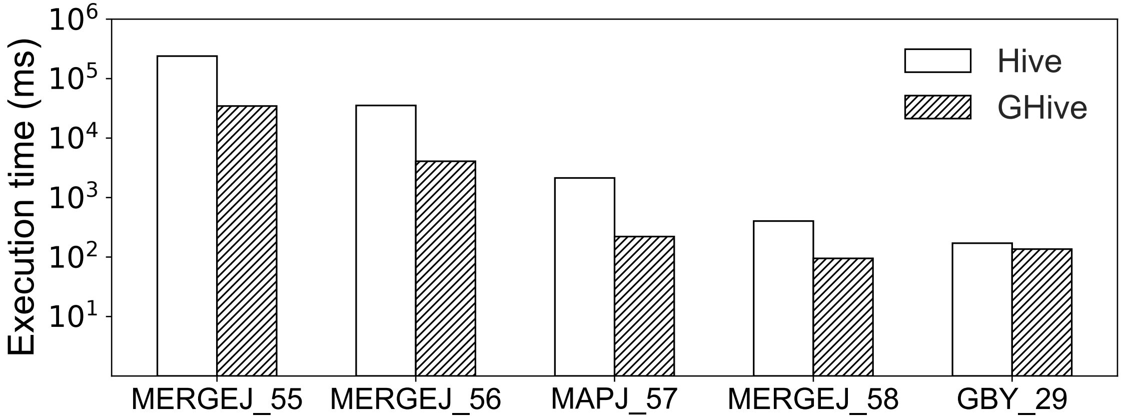


# Case Study




- Operator-level profiling:



ACM Symposium on Cloud Computing

# Thanks

Presenter: Haotian Liu

Email: dbgroup@sustech.edu.cn

SUSTech Southern University of Science and Technology

HUAWEI

The Hong Kong Polytechnic University 香港理工大學

Boston University

Aalborg University