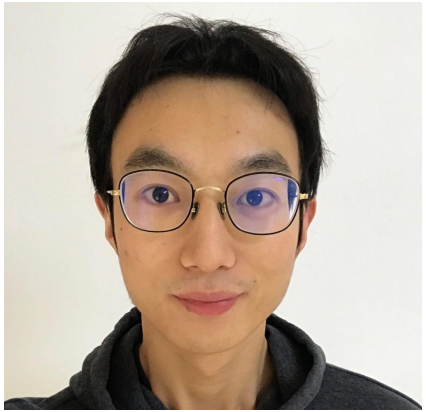
How to Protect the SQL Engine From Running Out of Memory

Presented by Huaiyu Xu/ Song Gao

PingCAP TiDB



Huaiyu Xu
Engineer at PingCAP

Database engineer
Technical lead of TiDB SIG execution
Committer of TiDB SIG planner

Github: @XuHuaiyu

Song Gao
Engineer at PingCAP

Database engineer
Maintainer of Chaos Mesh®
Committer of TiKV SIG scheduling

Github: @Yisaer

Agenda

- Causes of SQL engine OOM
- Solutions
- Implementation in TiDB (TiDB is an open-source, distributed, NewSQL database)
- Q&A

PingCAP

Part I - When will SQL engine run OOM

SQL engine runs out of memory

- statistics cache, table metadata cache, and etc. takes up too much memory space
- hash join builds a large hash table
- in-memory sort on a large dataset
- table scan/index scan reads data fast cause buffers too much data
- ...

Scenario summarization

- Memory resident object consumption
  - statistics cache, table metadata cache, ...
- Large memory consumption during calculating
  - pipeline breaker: hash join, sort, hash aggregation, …
  - pipelining operators: table scan, index scan, ...

PingCAP

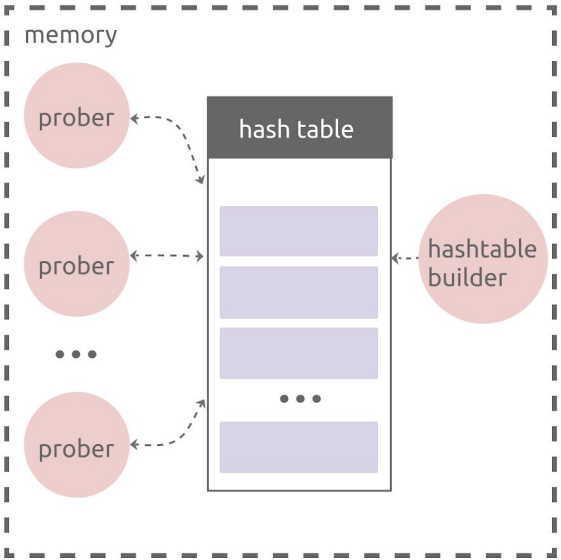
Part II - Solutions

Solution

- Memory resident object consumption
  - In-memory cache with limited size
- Large memory consumption during calculating
  - pipeline breaking operators: spill to disk
  - pipelining operators: adaptive control

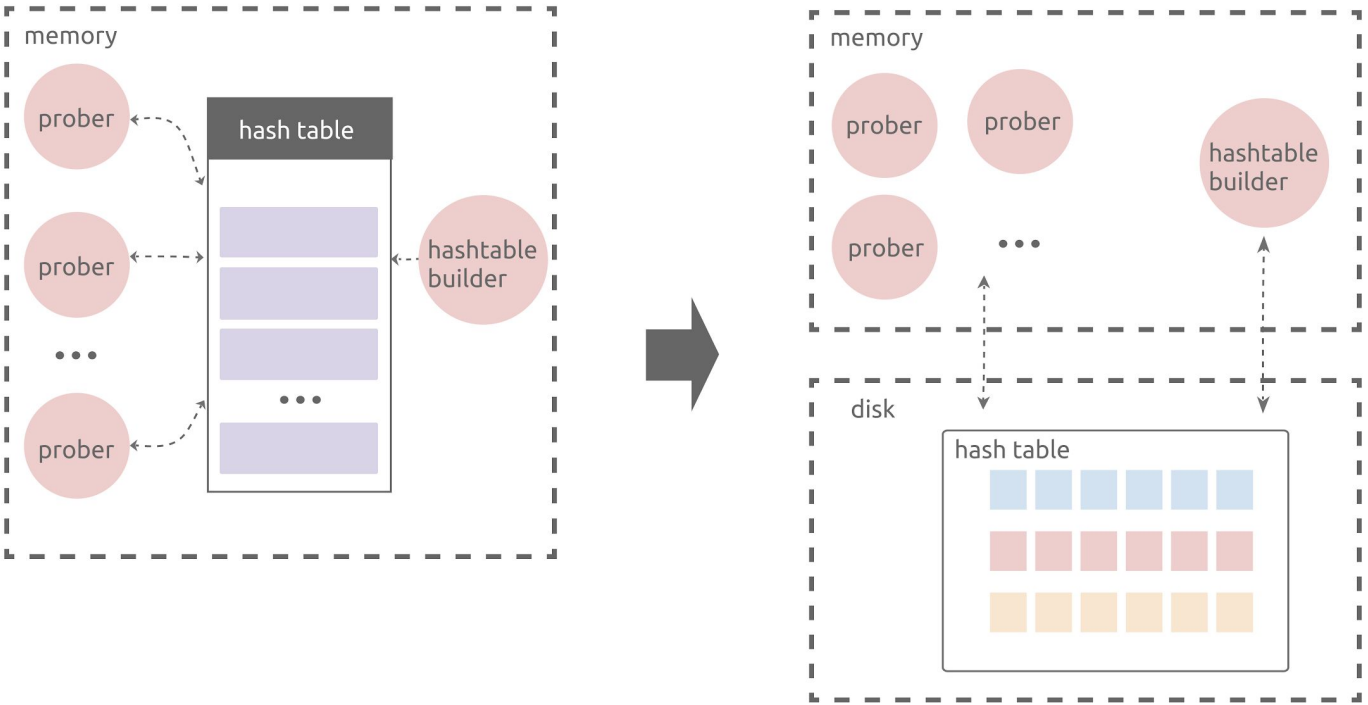
Pipeline breaker

- HashJoin



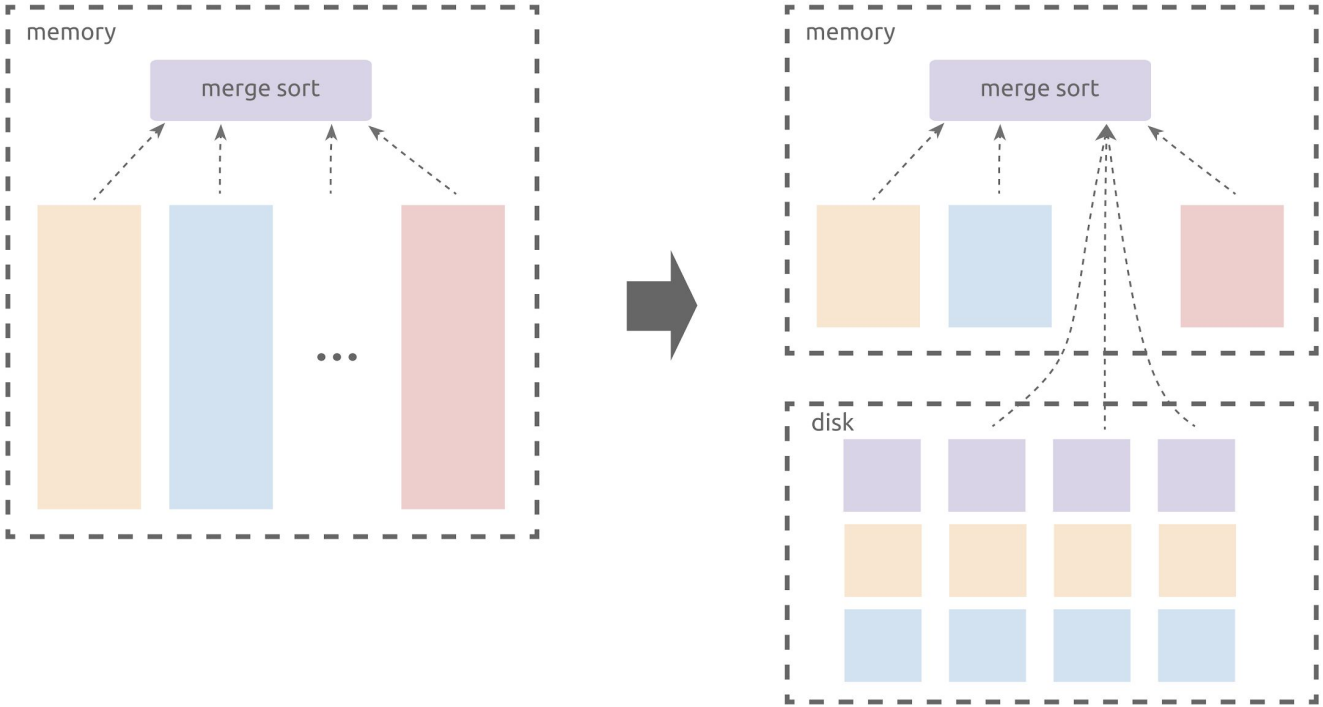
Pipeline breaker

- HashJoin



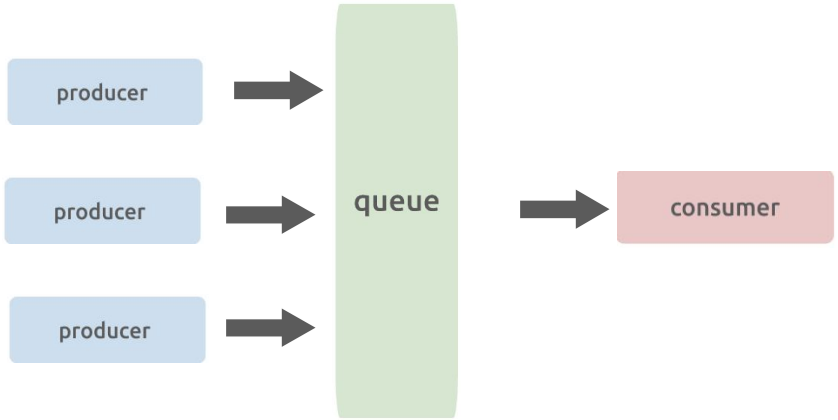
Pipeline breaker

- Sort



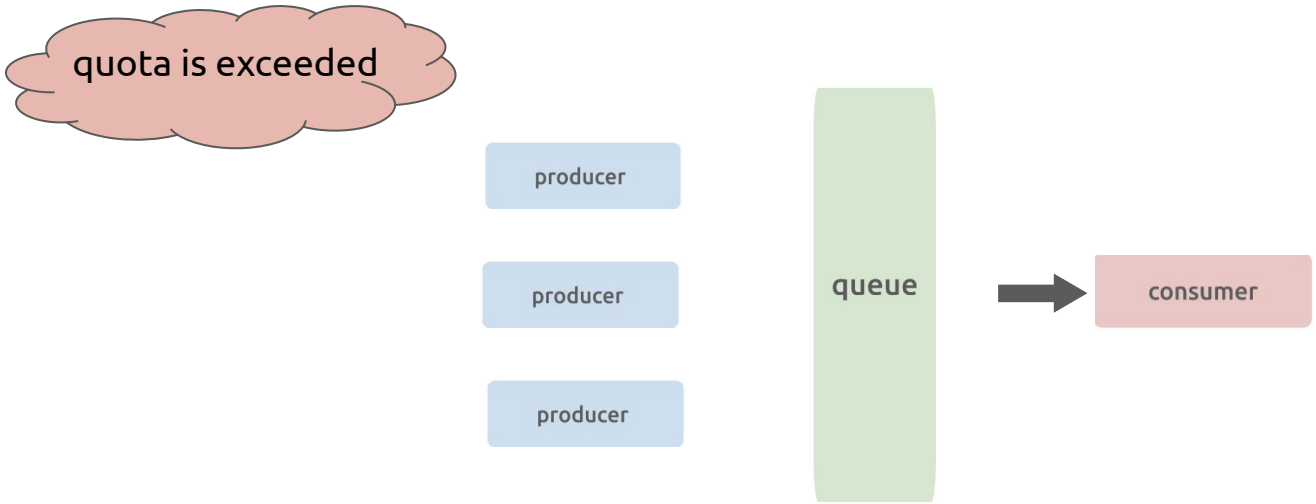
Pipelining operators

- producer-consumer model



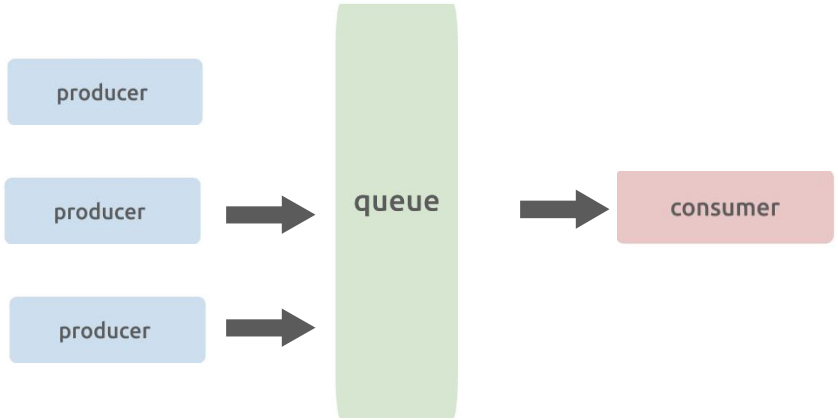
Pipelining operators

- suspend all the producers when the memory quota is exceeded



Pipelining operators

- producers keep creating data
- the rate will be controlled by removing one producer



PingCAP

Part III - Implementation in TiDB

What is TiDB?

Open-source distributed NewSQL database for hybrid transactional and analytical processing (HTAP) which speaks MySQL protocol

Horizontal Scalability

Transparent scale-out or scale-in

High Availability

Auto-failover to ensure business continuity

Strongly Consistent

Full ACID transactions at distributed environments

MySQL Compatibility

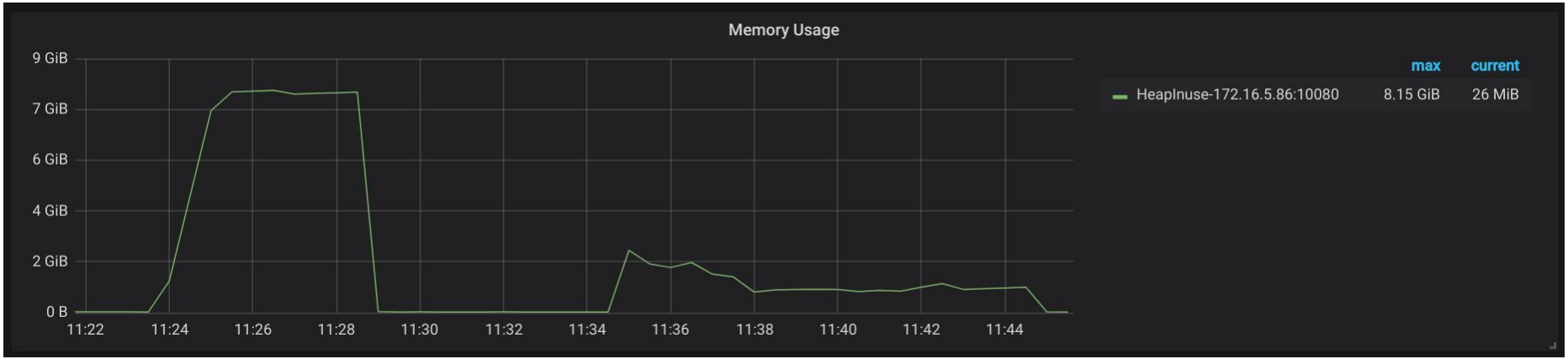
Without changing MySQL application code in most cases

Interfaces

- MemTracker
  - track the memory usage of each element
- OOMAction
  - abstract of different memory management strategies
  - DiskSpillAcion (spill to disk strategy)
  - RateLimitAction (adaptive control strategy)

DiskSpillAction

- select * from partsupp order by PS_AVAILQTY; (TPC-H SF:50)

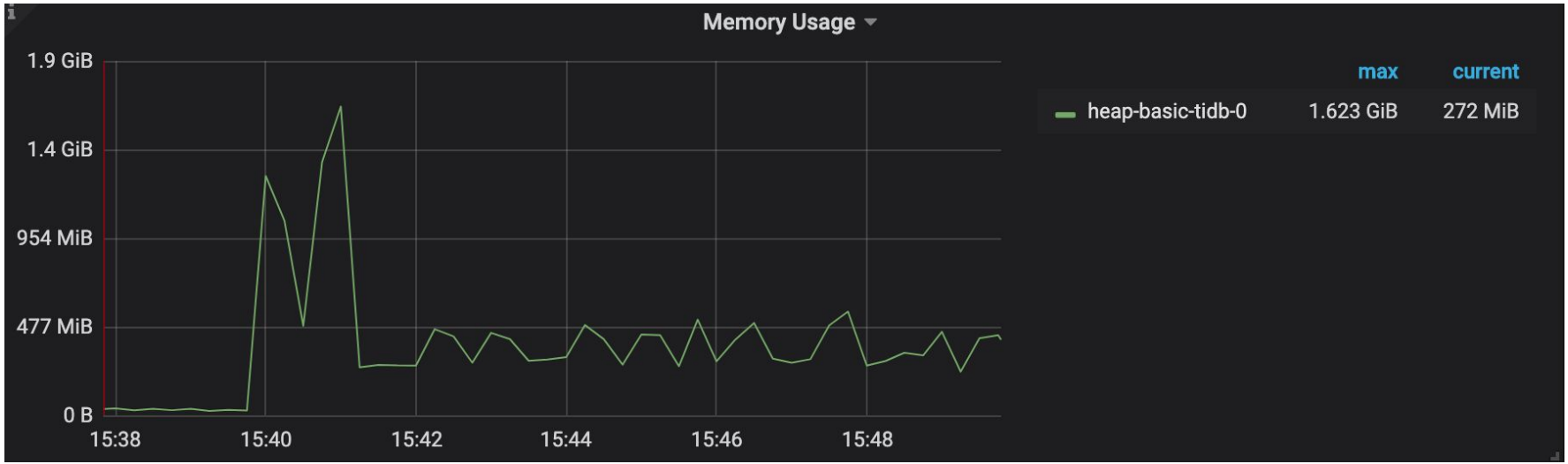


mem_quota is unlimited

mem_quota is set to 1GB
DiskSpillAction is triggered

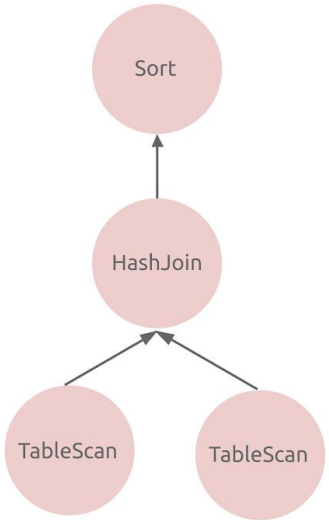
RateLimitAction

- dump 200GB data
    - OOM will happen when mem_quota is unlimited



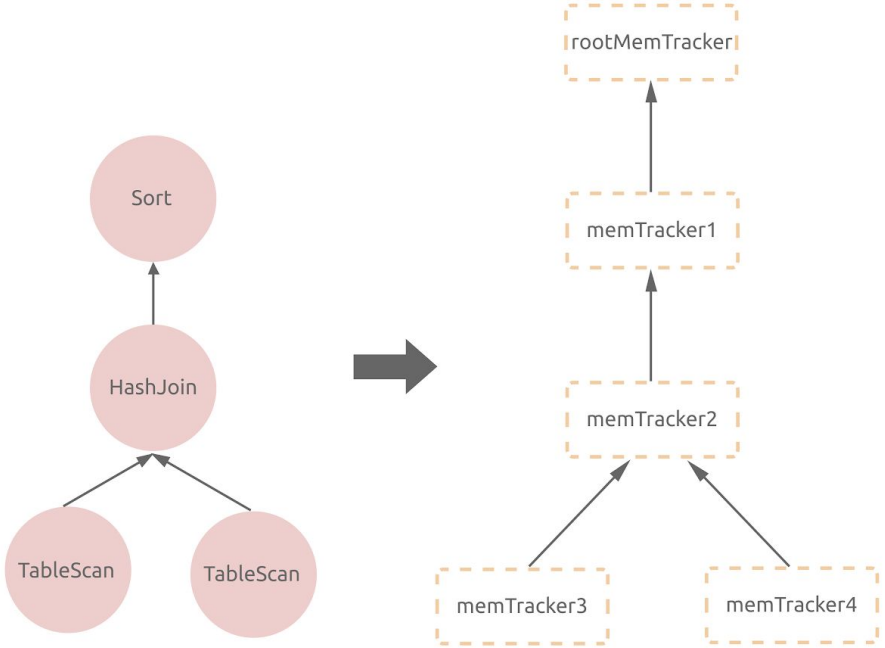
Implementation in TiDB

- select s.b from t join s on t.a = s.a order by s.b

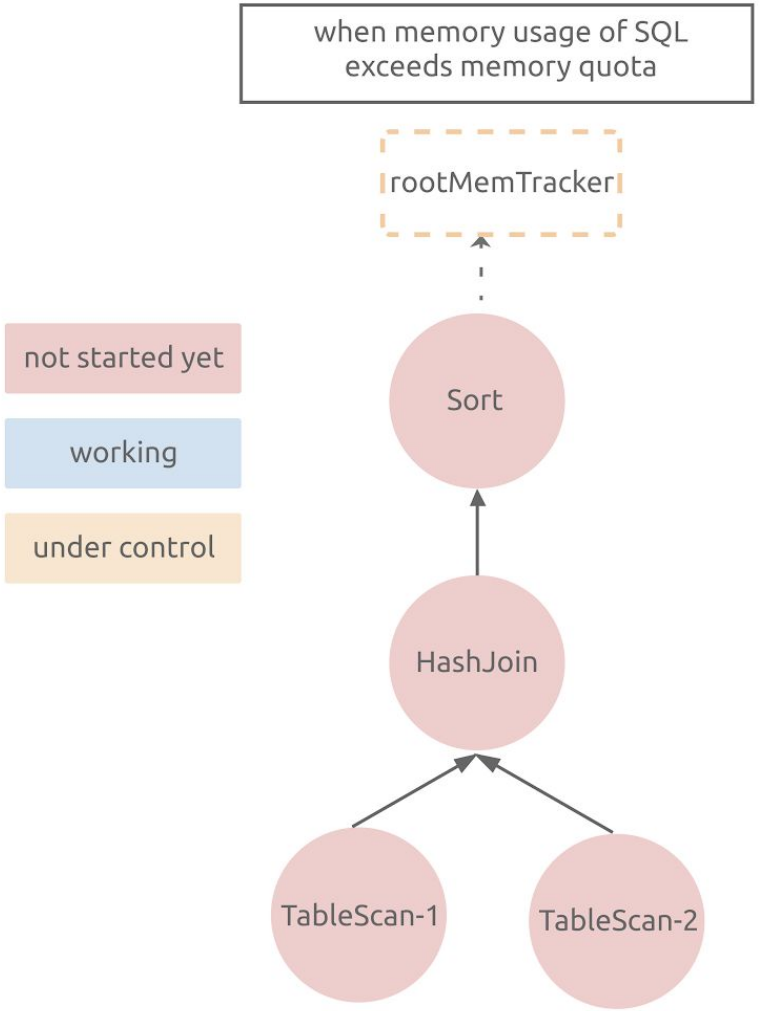


Implementation in TiDB

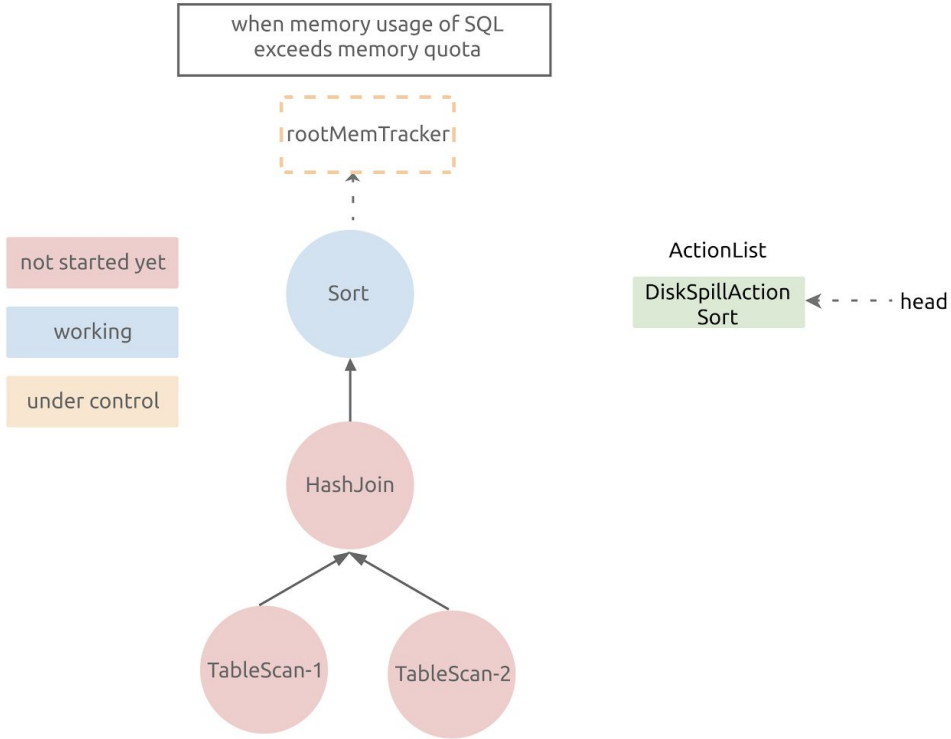
- select s.b from t join s on t.a = s.a order by s.b



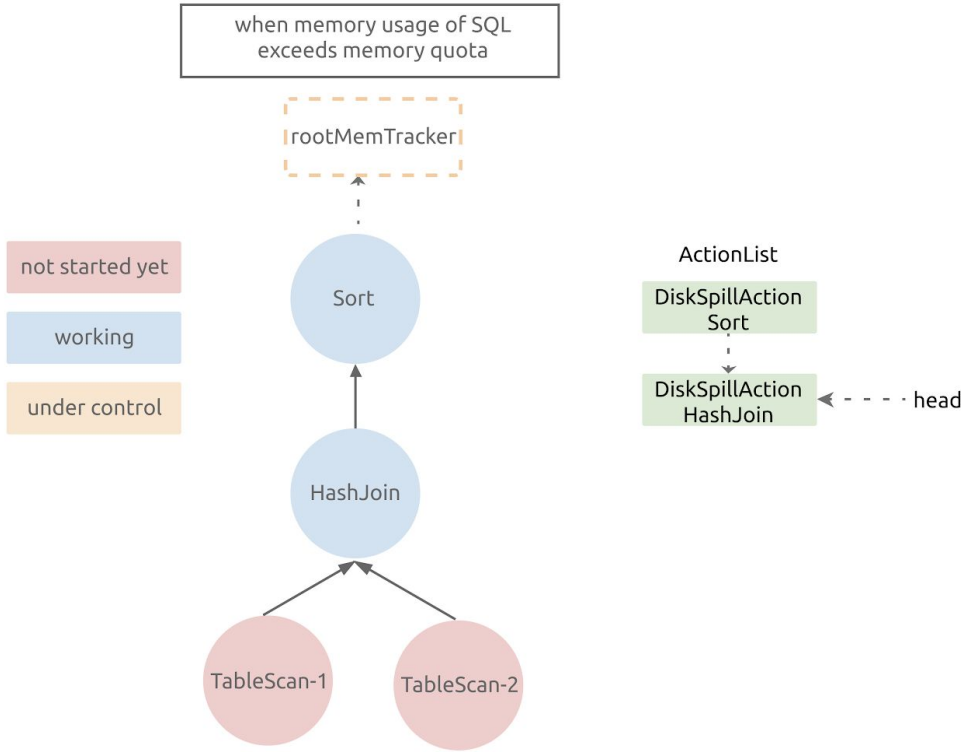
1. SQL engine builds a query plan tree.



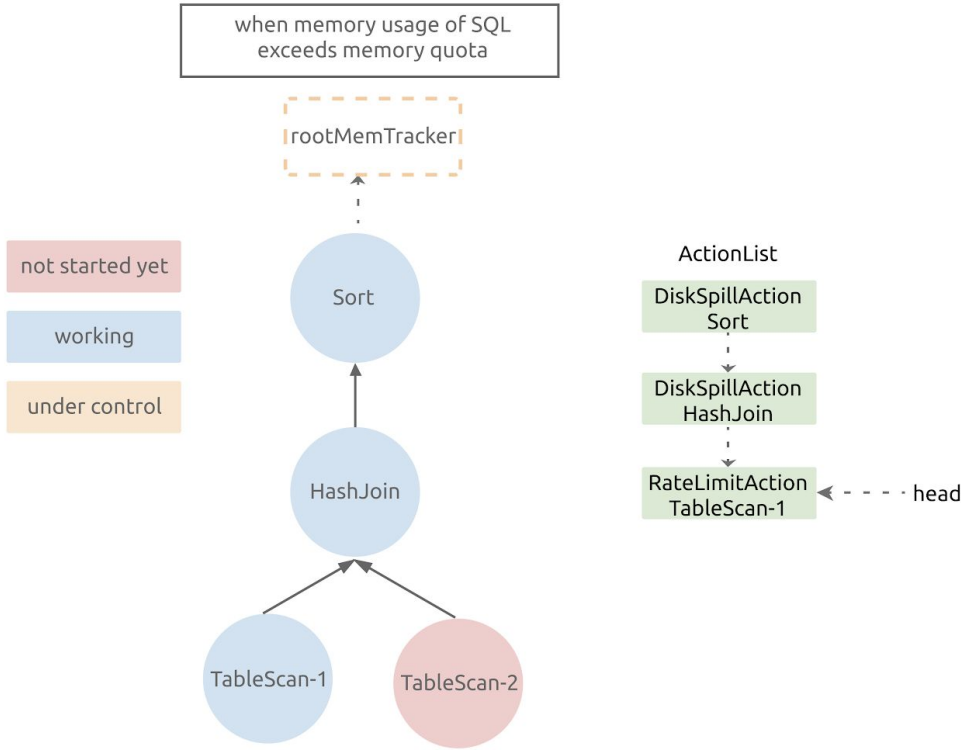
1. SQL engine builds a query plan tree.
2. Sort starts to work.



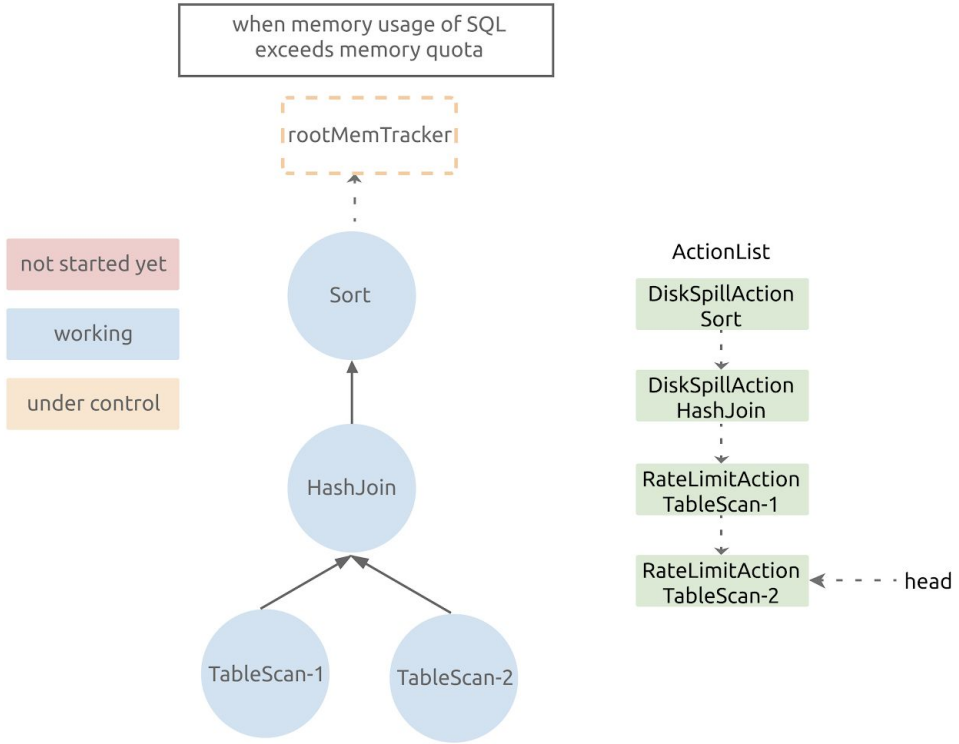
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.



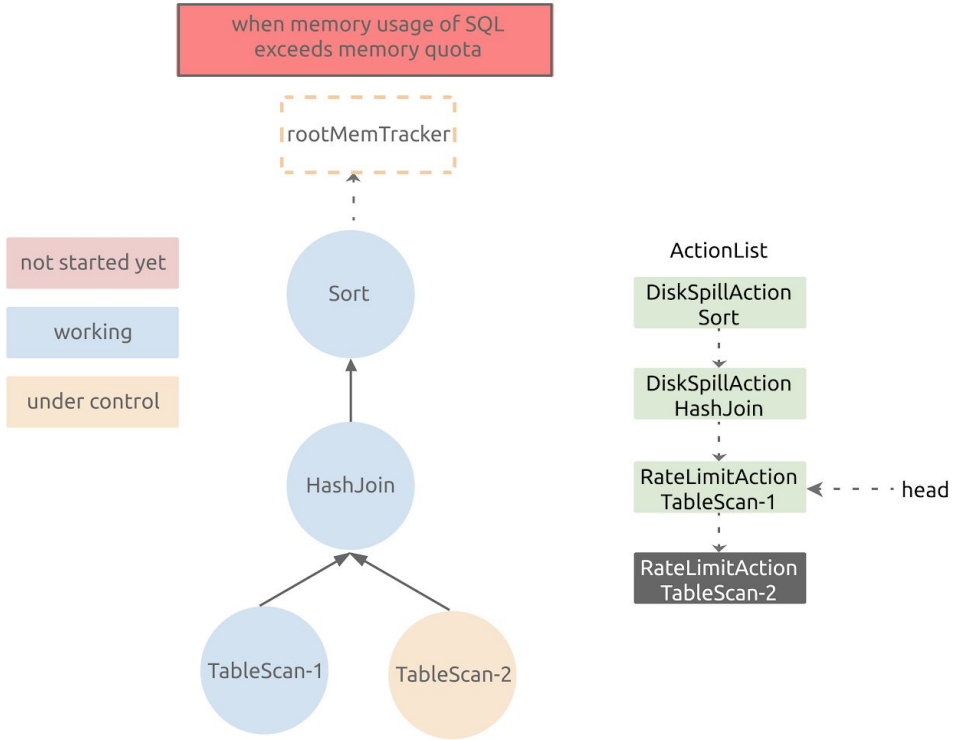
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.



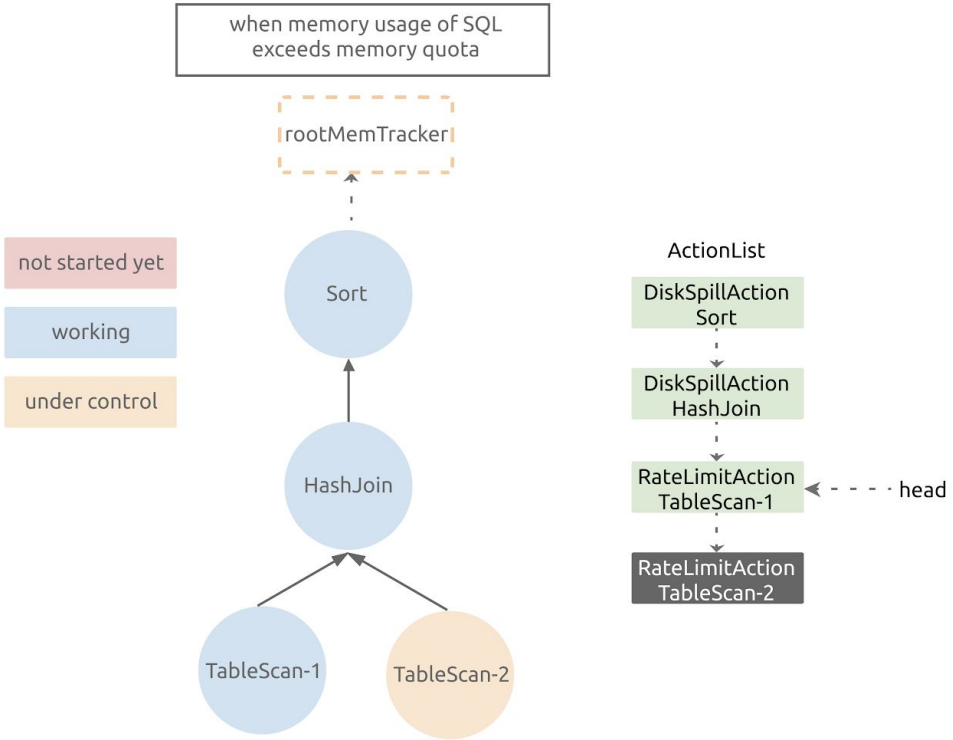
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.



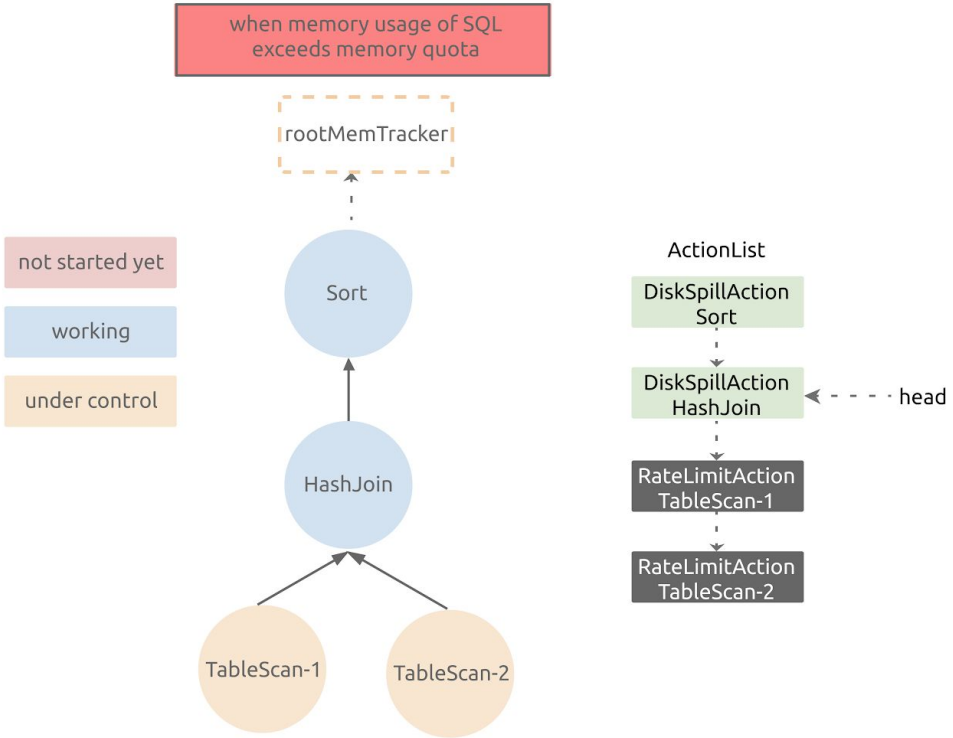
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.



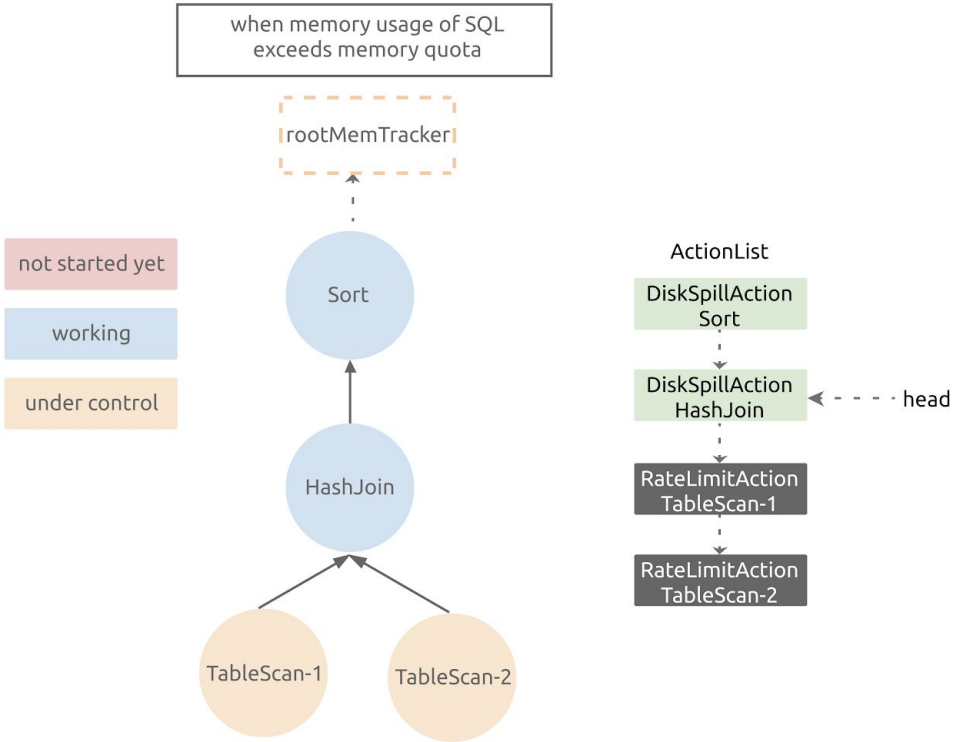
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.



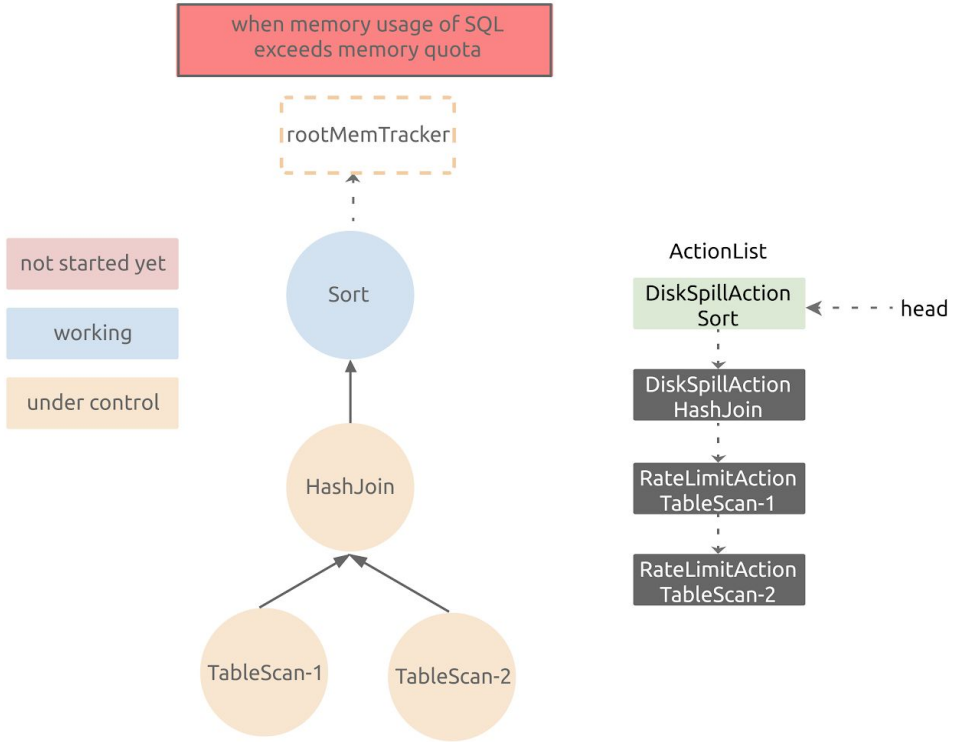
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.



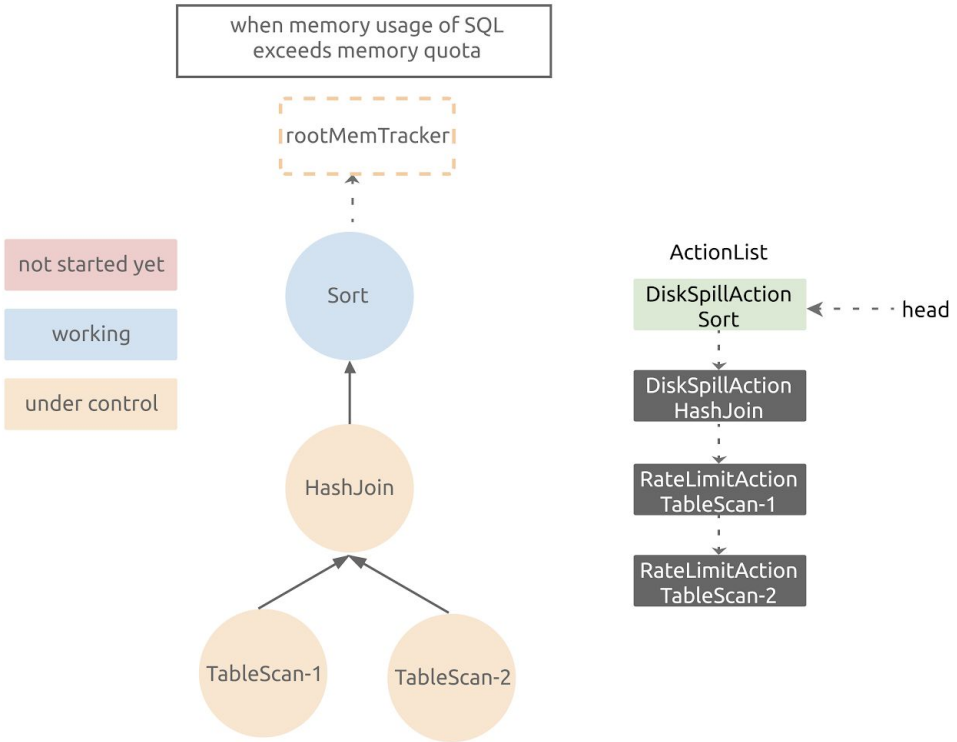
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.
9. Memory usage is under control.



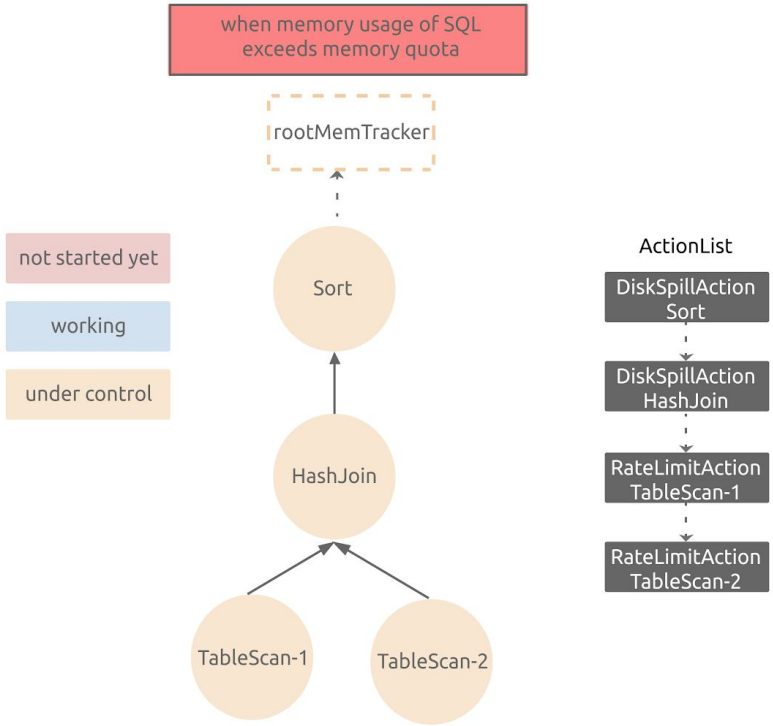
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.
9. Memory usage is under control.
10. Quota is exceeded. HashJoin is spilled to disk.



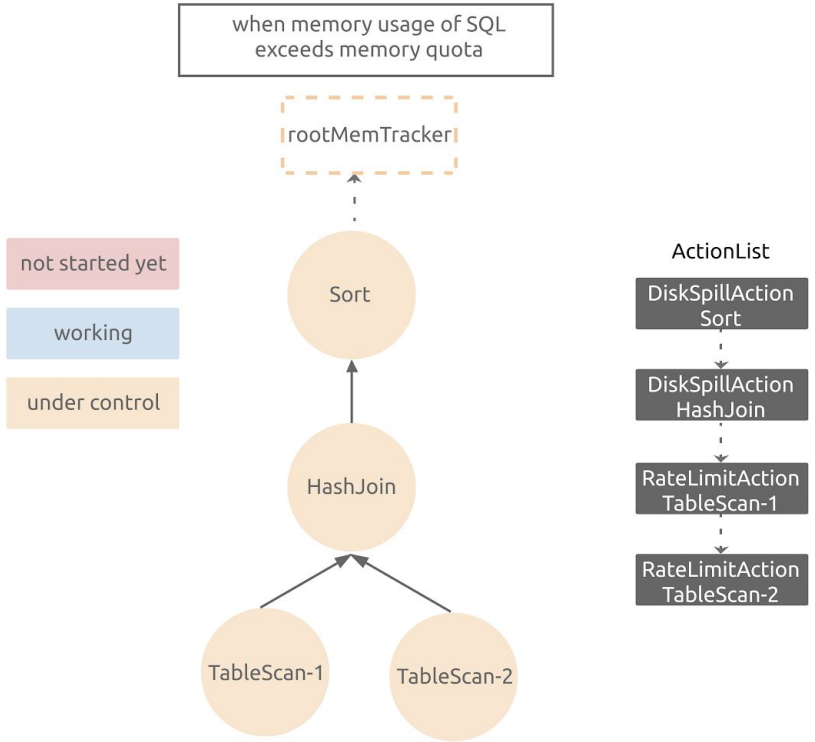
1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.
9. Memory usage is under control.
10. Quota is exceeded. HashJoin is spilled to disk.
11. Memory usage is under control.



1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.
9. Memory usage is under control.
10. Quota is exceeded. HashJoin is spilled to disk.
11. Memory usage is under control.
12. Limit is exceeded. Sort is spilled to disk.



1. SQL engine builds a query plan tree.
2. Sort starts to work.
3. HashJoin starts to work.
4. TableScan-1 starts to work.
5. TableScan-2 starts to work.
6. Quota is exceeded. TableScan-2 is adaptively controlled.
7. Memory usage is under control.
8. Quota is exceeded. TableScan-1 is adaptively controlled.
9. Memory usage is under control.
10. Quota is exceeded. HashJoin is spilled to disk.
11. Memory usage is under control.
12. Limit is exceeded. Sort is spilled to disk.
13. Memory usage is under control.



Future work

- Support priority for different actions
- Support more adaptive memory control strategy
- Support a server-level memory control strategy

Join us

- GitHub: https://github.com/pingcap/tidb
- Website: https://www.pingcap.com/
- Twitter: @PingCAP
- Slack: #everyone in Slack

Q&A

Thank You !