Multi-Threaded Queries

S20 15-721 Final Presentation

Zecheng He, Yinuo Pan, Yuhong Zhang
Building on Prashanth's work

Project Goals

Adding intra-query parallelism for Sequential Scans in terrier.

75% Goal

Parallel Scan in C++

✓ DataTable support

✓ Dynamic block_range assignment

100% Goal

Parallel Scan Codegen

✓ Built-in Functions

✓ Needed arguments

✓ Parallel operator mode

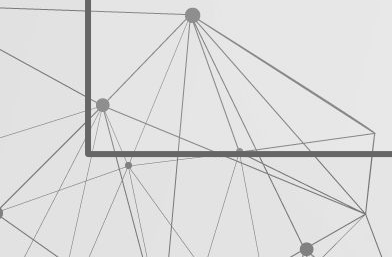
✓ Output buffer partition

125% Goal

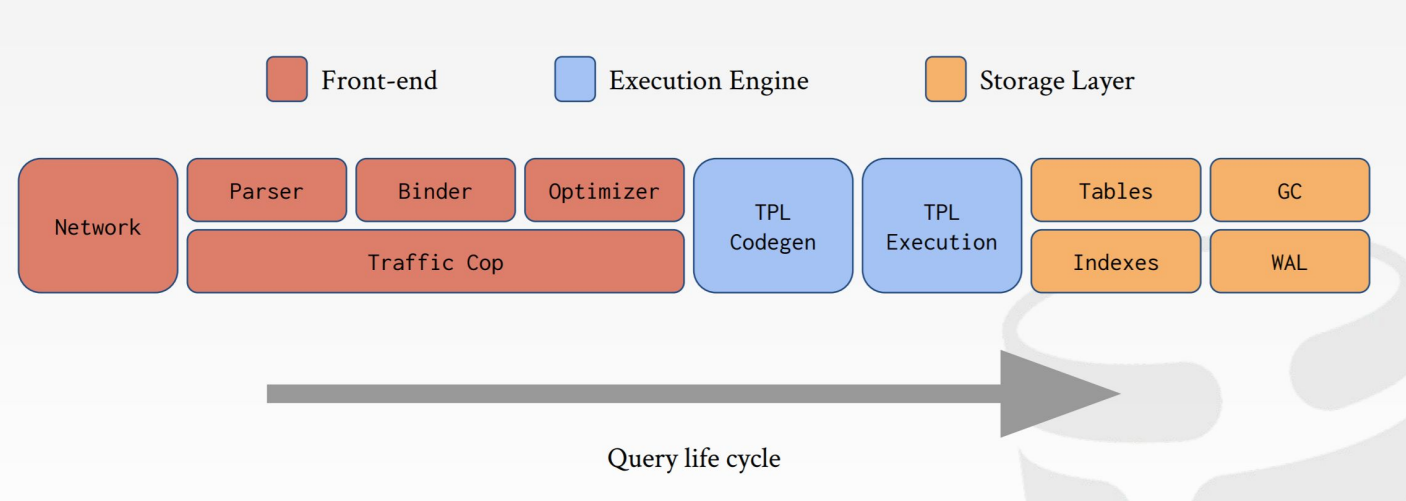
Memory Access & Optimization

✓ Thread-local state access

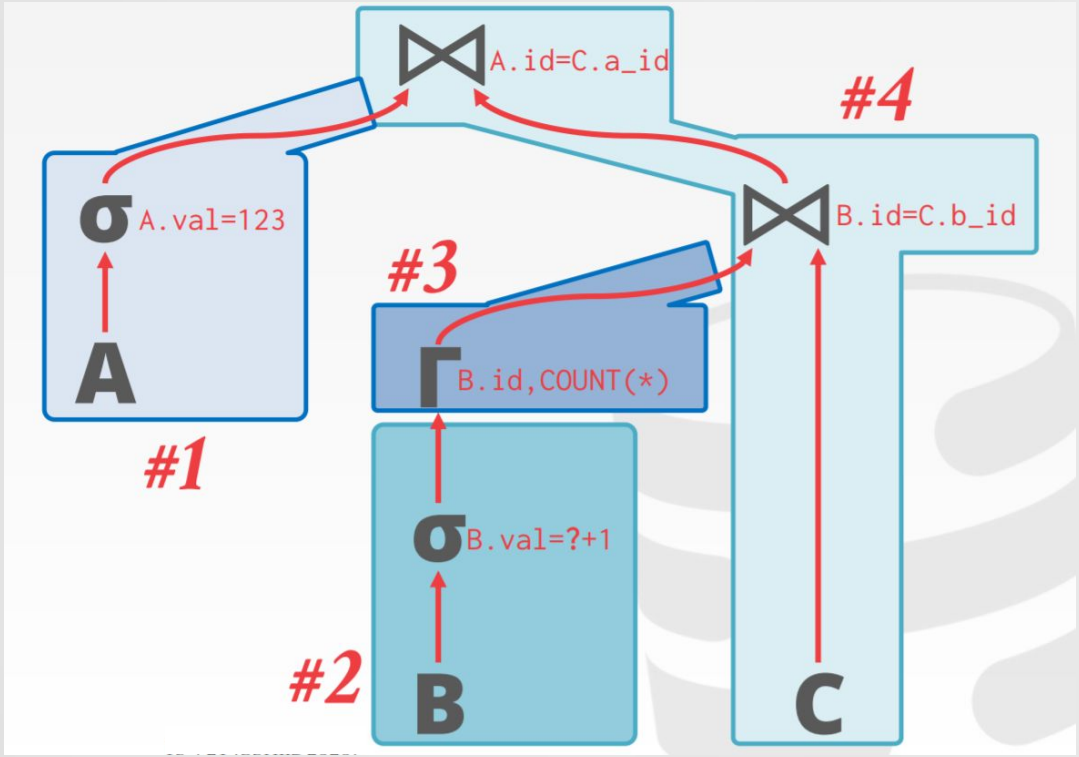
Performance optimization (In progress)



System Design



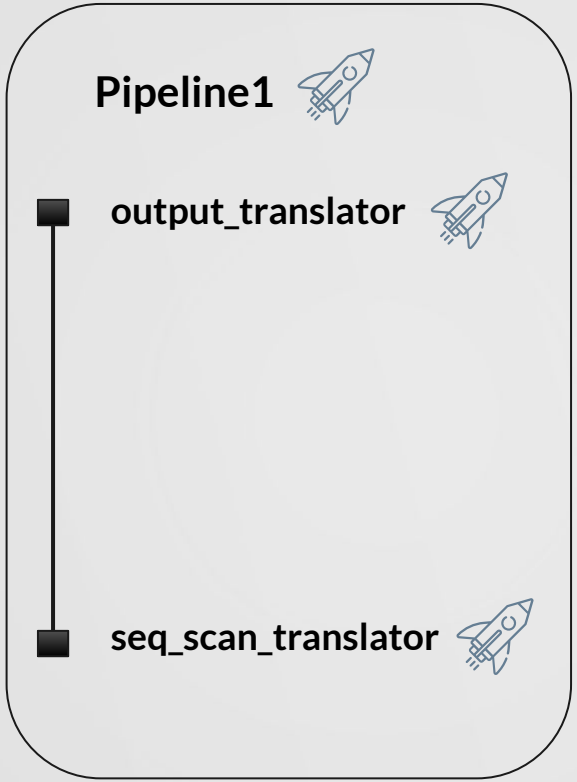
System Design



Pictures from CMU 15-721 Course

Step 1: Breaking the physical plan into multiple pipelines

System Design



Step 2: Deciding the execution mode of the whole pipeline:

- Parallel if every operator is parallel
- Serial otherwise

System Design

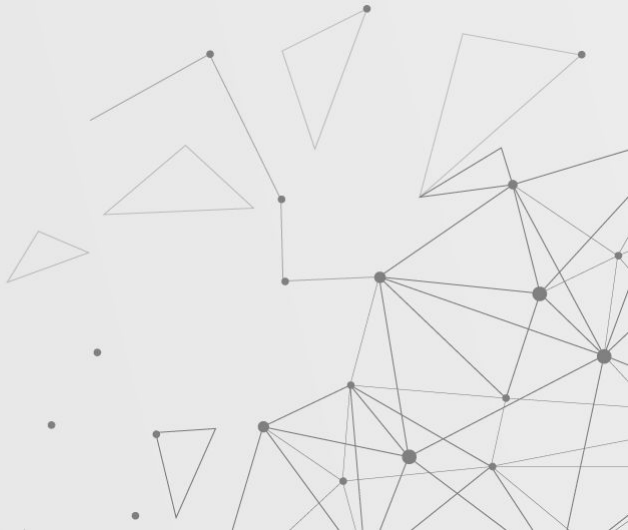
Step 3: Generating corresponding function with correct arguments

Serial

```
Pipeline0_SerialWork(query_state, exec_ctx) {
    // Initialize tableVectorIterator to scan over the whole table
}
```

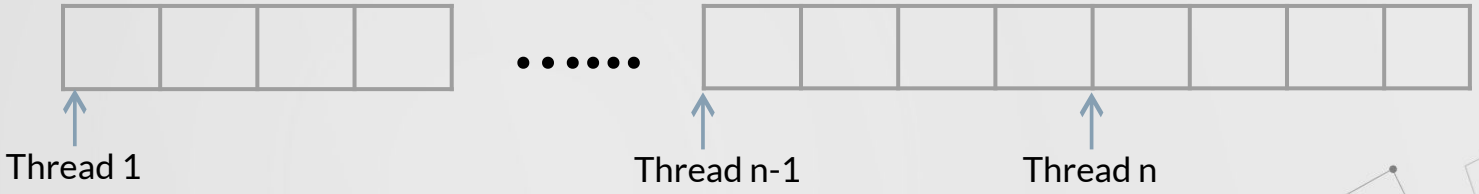
Parallel

```
Pipeline0_ParallelWork(query_state, exec_ctx, table_vector_iterator) {
    // table_vector_iterator is initialized to iterate its own block range
}
```



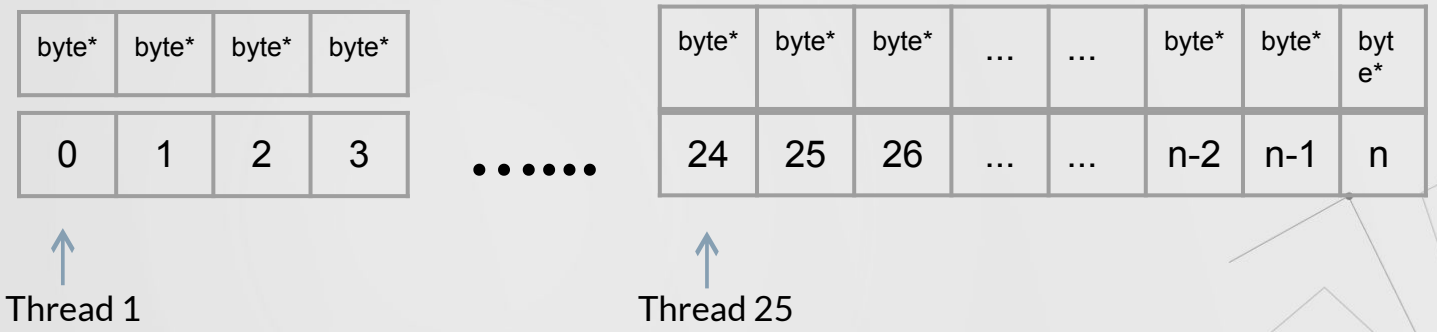
System Design

Step 4: Parallel Scan on different block ranges



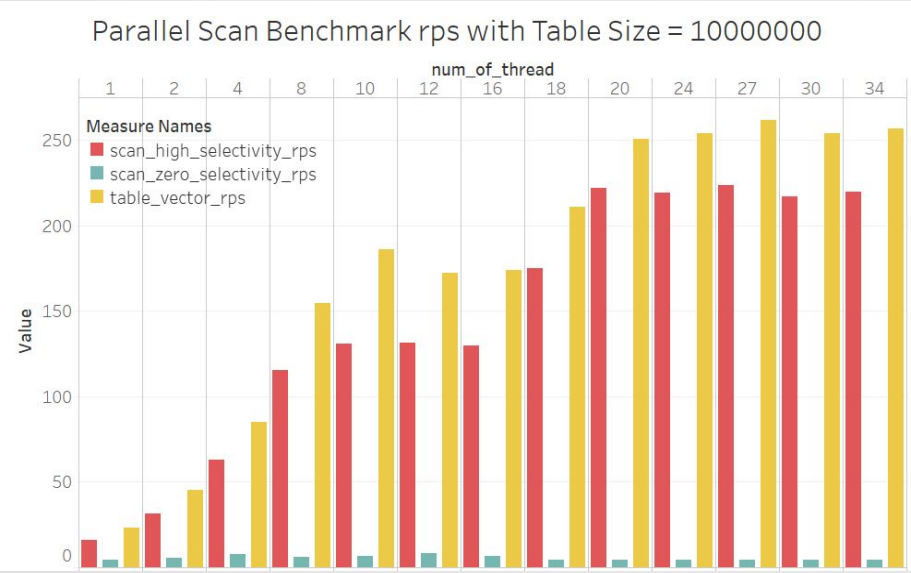
System Design

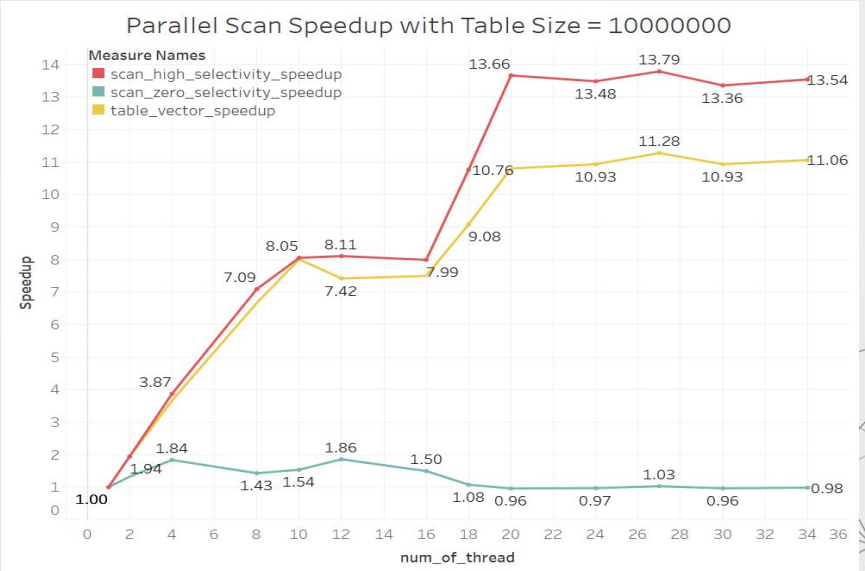
Step 5: Concurrently writing to output buffer



Benchmark

- TableVectorBenchmark: scan on c++ side with multiple worker
- ParallelScanBenchmark: execute whole sequential scan query (including the output buffer)





Why the whole execution is so slow?

1. Latch on call back function

   Callback functions invoked in output buffer are not thread-safe

2. Even with a thread-safe callback - the maximum speed up was still ~4.4x.

   Comparing the results between high and low selectivity, we think it might be the step copying data into output buffer.

   (More profiling and optimization)

Correctness

- ParallelScan test on DataTable
  - DataTableTest: RangeScanTest
  - TableVectorIteratorTest: ParallelScanTest
- Codegen
  - CompilerTest: Generate tpl code from physcial plan
  - Other unit tests on sequential scan

Code Assessment

1. Virtual method

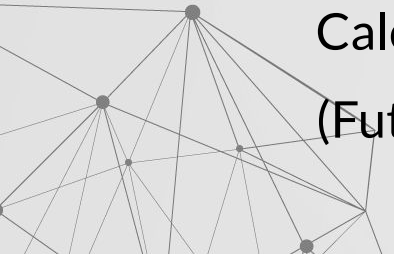
   LaunchWork() is not implemented in most operators. Temporarily set to virtual but not abstract method.

2. Magic constant
   - sema_builtin.cpp

     Adding more comments to explain the arguments being checked.
   - table_vector_iterator.cpp

     Calculating the block range using: *# of blocks / # of cores*

     (Future: ask thread pool to provide available threads)



Future Works

- Profile current implementation to identify the bottleneck
- Integrate with Numa thread pool
- Infrastructures for other operators' parallelism (pipeline states, compilation context, etc.)
- Add support to other operators (hash join, hash aggregation, etc.)

THANKS

Does anyone have any questions?