VIKA TEAM

THAI SON NGUYEN – 2ND JUN

INTRODUCTION

- About Us?

CONTENTS

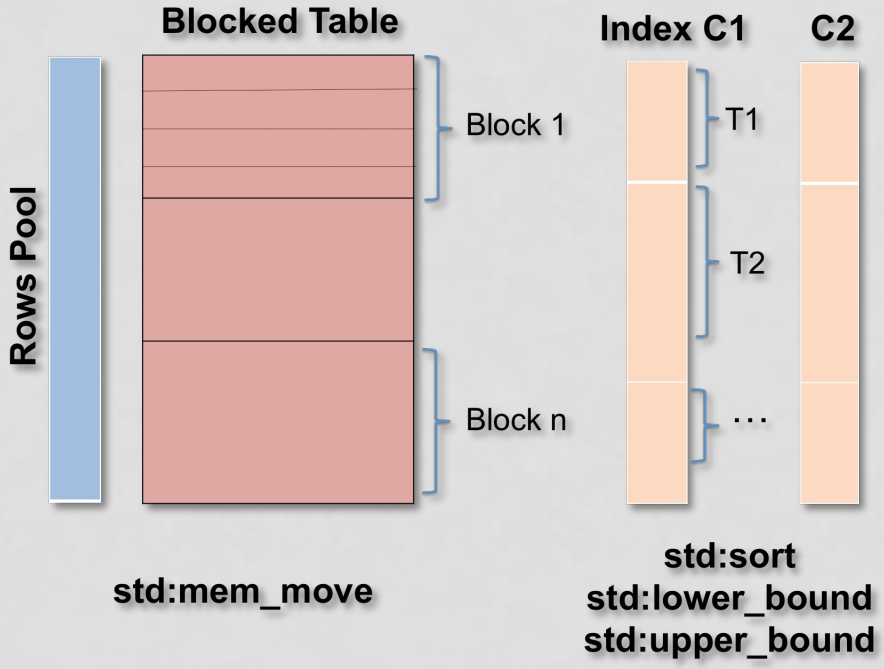
OUR APPROACH

- To build a program that:
  - Handles an **unlimited number** of transactions & queries
  - Runs **as fast as possible**

- Our approach:
  - Minimize memory allocations / de-allocations (avoid memory leaks)
  - Use indexes as much as possible
  - Parallelize every step

DATA STRUCTURES

- **Tables** & **Column Indexes**
- Tables:
  - Storing permanent data
  - Row oriented
- Column Indexes:
  - Indexing data belonged to live transactions
  - Sorted arrays of value-row_id pairs
  - Predictable size



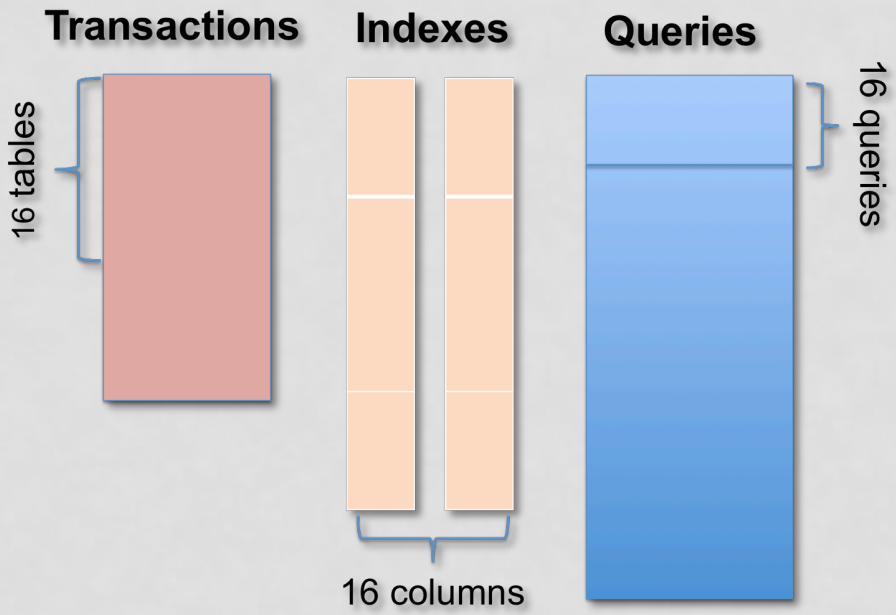
ALGORITHM

- **Step 1**: Read and put transactions and queries into queues until seeing **Flush** operation
- **Step 2**: *Concurrently* process transactions
- **Step 3**: *Concurrently* build indexes
- **Step 4**: *Concurrently* validate queries and corresponding transactions
- **Step 5**: Write results & clean dead transactions

- Then repeat these steps

EXECUTION & PARALLELISM

- Parallelizing steps **#2,#3,#4** in different manners
- Processing transactions **per table** (not so efficient)
- Building indexes by **table columns** (horizontal)
- Parallelizing **queries** (vertical)



IMPROVEMENTS FOR DATA SPECIFICATION

- Use **max-min index** for low-cardinality columns
- Index tuples that are belonged to a **range of small transactions** instead of indexing tuples of each such transaction
- Try to find a good condition which **leads to a very small number of matched tuples** instead of combining query conditions using set intersection

CONCLUSION

- Not yet taking advantages of new computer architecture (**SIMD instructions**)
- **Many thanks** to the ACM Contests

THANK YOU