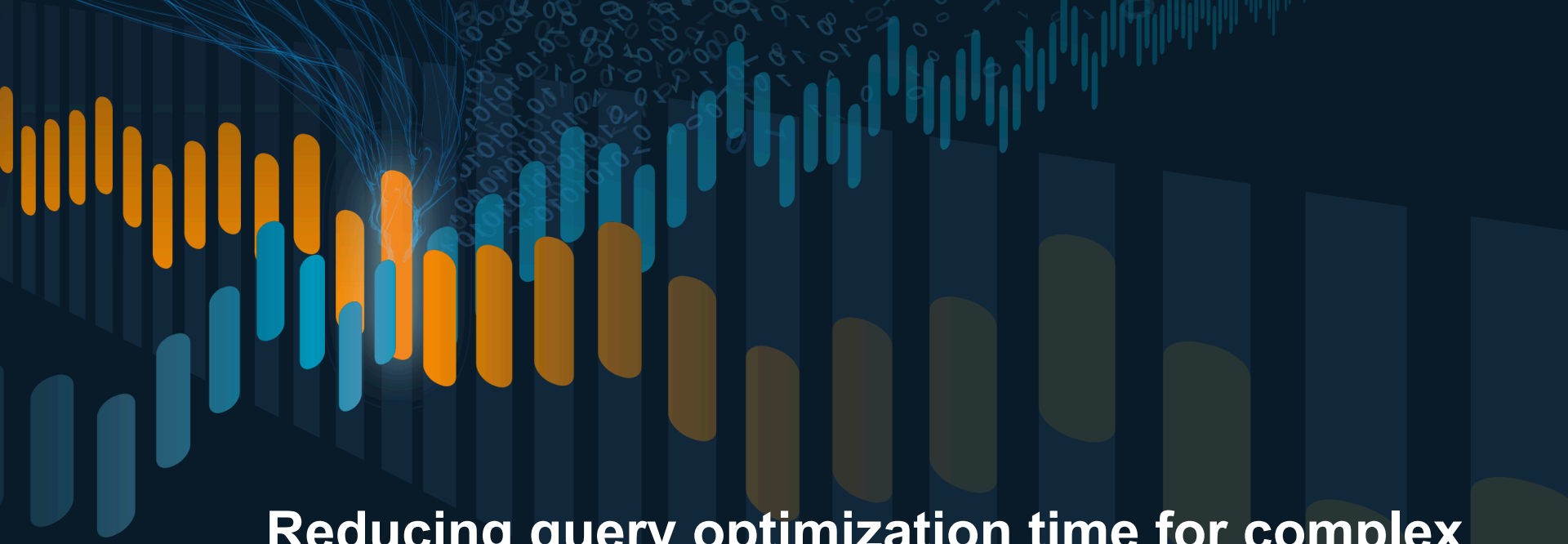

Reducing query optimization time for complex analytical queries in an in-memory distributed database

XLDB 2015

Rajkumar Sen, Jack Chen, Robert Walzer

memsql®

Need for fast query optimization

- Complex analytical queries which need to be answered within sub-second latencies are common
- Very small time budgets for query optimization
- Need extremely fast query optimization for complex analytical queries involving joins and aggregations
- Must choose efficient execution plans with near-optimal runtime performance

Reducing query optimization time

- Challenge: generate near-optimal plan within very limited query optimization time
- Time consuming components in query optimization need to be done efficiently and intelligently
  - Cost-based query rewrites
  - Join enumeration to choose best join order
    - Generating bushy join plans
  - Distributed join optimization
    - Cluster-wide join order and data movement decisions

Global and local optimization

- Global
  - Data movement decisions and global join order
  - Focus on network and distribution method costs and intermediate cardinalities
- Local
  - Choose local per-node join order
  - Join method and access path selection

Efficient join enumeration

- Prune heavily using smart heuristics
- Only consider left-deep join trees
  - Bushy join trees via query rewrites
- Join enumeration per select block
  - Moving joins between select blocks is done via query rewrites

Bushy join plans via query rewrite

- Considering all bushy joins in join enumeration is extremely expensive
- However, bushy joins are critical for execution performance
  - E.g. several TPC-DS queries benefit by 3-10x
- Consider only promising bushy joins instead of all possible cases
- Look for common query shapes that benefit from bushy plans and introduce bushiness via query rewrite


Thank You
www.memsql.com
memsql®