MGM COLLEGE OF ENGINEERING AND TECHNOLOGY

LOTUS SQL

Guide:
Mrs. Deepa P L
Assistant Professor
Department of Computer Science

Presented By:
Bibin P Varghese
HKC18CS021
S7 Computer Science

TABLE OF CONTENTS

- Objective
- Introduction
- BACKGROUND –Lotus
- BACKGROUND–calcite
- WORKFLOW
- Physical Operators
- Query optimization
- Evaluation
- Advantages and Disadvantages
- Conclusion
- References

OBJECTIVE

- An engine to provide SQL support for dataset abstraction on native backend Lotus
- Convenient SQL processing framework to deal with frontend jobs

INTRODUCTION

- Rapid development of information technology has brought significant progress to human society
- The amount of data that computer systems need to deal with has increased accordingly
- SQL is a common choice for data analysis
- To evaluate the execution efficiency of SQL queries TPC defines a benchmark TPC-H is widely used for OLAP performance evaluation.
- SparkSQL is designed for processing structured data on Spark.

INTRODUTION (CONT….)

- Garbage collection and data serialization, are attributable to JVMs
- Lotus is a high performance data-parallel computing engine built with c++
- High performance because of bare-metal runtime environment
- Compact storage strategy,Coarse-grained function call,Memory efficient design
- Uses template usage and automatic type deduction
- Challenges :Semantic gap exists between lotus and SQL and Massive development efforts

BACKGROUND -LOTUS

- Single machine data parallel computing engine
- Low-overhead storage module & Highly efficient compute modue
- Storage module is designed to have low overhead ,combination of buffer caches and compact object models
- Compute model is C++ dataset programming model
- Provides the abstraction of compact collections and efficient operation implementations
- Logically an array of records segmented into multiple partitions

BACKGROUND–LOTUS (CONT….)

- Abstraction is quite similar to spark's RDD for distributed allocation
- Lazy evaluation strategy and supports fault tolerance
- Intermediate result datasets can be cached explicitly
- Employs compact object storage
- Reduce serialization and deserialization overhead
- Supports primary data types
- For string dataset, data are organized into two compact buffers
- Provide a compute engine for LotusSQL

BACKGROUND–CALCITE

- Open-source software framework
- Provides query processing , optimization and query language support
- Perceives developers of specialized systems encounter related problems such as query optimization or the need to support query language
- Minimize the engineering effort
- Unifying and pluggable framework

BACKGROUND–CALCITE (CONT….)

- Logical operators is the primary form of operation and it includes filter , project and join
- Physical operator assigns an implementation method
- Operators compose the relational algebra expression tree, which is the representation of an execution plan
- Execution plan consisting mainly of physical operators called physical plan
- Cost-based dynamic programming search to find the best execution plan
- Employs as a frontend to produce a physical execution plan.

WORKFLOW

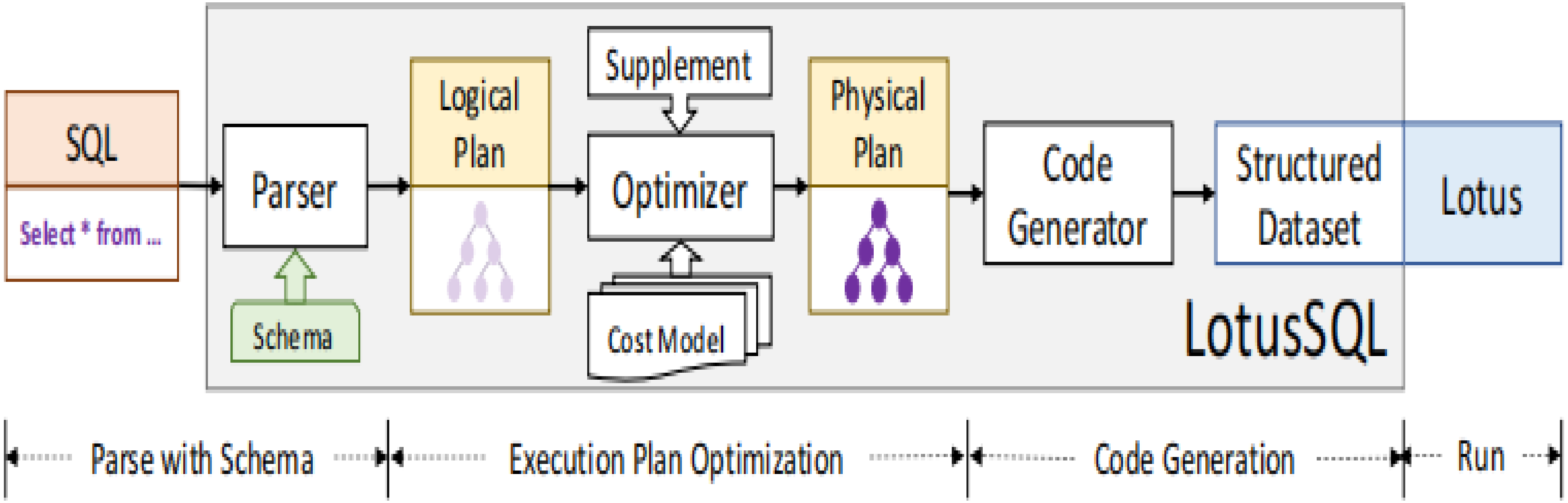

Fig. 1 Workflow overview.

PHYSICAL OPERATORS

▪PHYSICAL OPERATORS-OPERATION FUSION

- Technique to dataset operation implementation
- Proposed in the main memory database field
- Operation can be fused together to maximize data and code locality
- Leaves tuples in registers and makes the execution cheap

▪ IMPLEMENTATION AND COST MODEL

Table 1 Operator list.

LogicalOp	PhysicalOp	Description
TableScan	LotusTableScan	Read a table (dataset) from the file system.
Filter	LotusFilter	Filter a table by given condition.
Project	LotusSelect	Select some columns from a table.
	LotusMap	Map table rows by given expression.
Aggregate	LotusAggregate	Aggregate all rows by given function.
	LotusHash Aggregate	Aggregate rows by given group and function via HashMap.
Join	LotusCartesian Product	Calculate cartesian product of two tables.
	LotusBroadcast HashJoin	Join two tables via broadcasting one to the other and HashMap.
	LotusShuffle HashJoin	Join two tables via re-partitioning tables and using HashMap.
Sort	LotusSort	Sort all rows by given reference key and direction.
	LotusTopK	Find top-*k* rows by given reference key and direction.

IMPLEMENTATION AND COST MODEL

- Cost model evaluates cost of the implementation
- Generally the cost can be calculated in several aspects , such as CPU usage , memory access and I/O bytes.

IMPLEMENTATION AND COST MODEL

Eg: *LotusBroadcastHashJoin*

$$Cost_{broadcast} = LeftInputRowCount * LeftInputColumnCount * NumRightPartition$$

$$Cost_{hashmap} = (LeftInputRowCount * NumRightPartition + RightInputRowCount) * \log(LeftDistinctRowCount)$$

$$Cost_{output} = OutputRowCount * OutputColumnCount$$

$$Cost = Cost_{broadcast} + Cost_{hashmap} + Cost_{output}$$

QUERY OPTIMIZATION

DECORRELATION OF SUBQUERIES

- Subqueries that do not involve external variables are noncorrelated & parsed into independent subtree
- Correlated subquery appears as a $LogicalCorrelate$ operator in original logical plan
- Behaves like a special type of join, but the right input subtree refers to variables from left input
- Re-executing the right subtree every time hampers performance in most cases
- Thus decorrelation is necessary
- Calcite adopts several methods for decorrelation, but they are not efficient enough

DECORRELATION OF SUBQUERIES (CONT..)

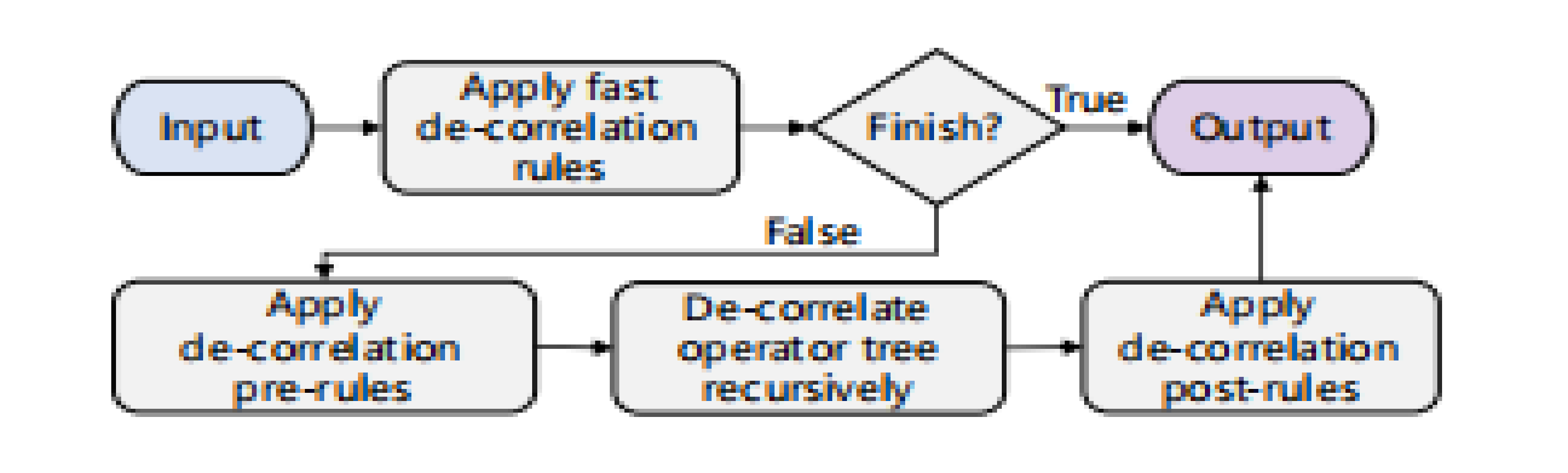

Fig.1 Calcite decorrelation

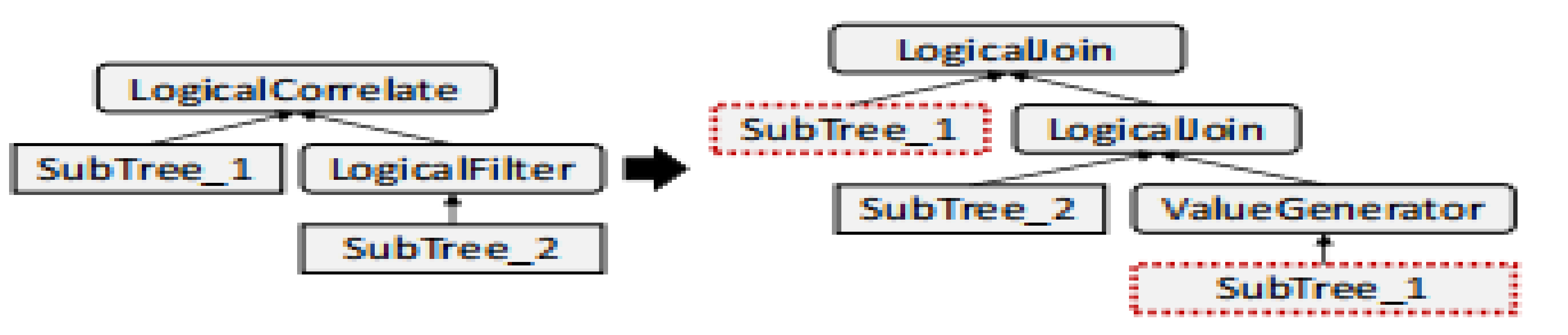

Fig.2 Decorrelation Example

ROWCOUNT ESTIMATION

- Calcite users invoke $getRowCount()$ to estimate the no: of output rows of an operator
- Estimation is based on Calcite's mechanism that provides metadata
- This estimates RowCount and condition selectivity
- Also tracks inherited properties- unique keys and column origins

ROWCOUNT ESTIMATION

- Eg: estimation of the output RowCount of the simple query

$$select * from\ TableA, Table\ B \text{ where } TableA.x = TableB.y$$

its estimated number of rows is

$$RowCount = TableA.RowCount * TableB.RowCount * Selectivity(TableA.x = TableB.y)$$

selectivity is a simple guess that returns a value between 0.5 and 1.0

EVALUATION

- Workloads and environment
- Query translation analysis
- Computing time
- Memory usage

ADVANTAGES

- Big data processing system
- Takes less memory

DISADVANTAGES

- Complex
- Long queries

CONCLUSION

- Lotus is a big data processing system developed with native programming language
- To boost lotus we present LotusSQL
- Uses Calcite to compile and optimize queries with the guidance of a physical cost model
- Dependencies are resolved as a whole and compressed during C++ compilation time
- With all the about strategies, LotusSQL outperforms SparkSQL in TPC-H queries by more than twice on average

REFERENCES

- [1] Apache Hadoop, Apache hadoop, http://hadoop.apache.org, 2021.
- [2] J. Ekanayake, H. Li, B. J. Zhang, T. Gunarathne, S. H. Bae, J. Qiu, and G. Fox, Twister: A runtime for

  iterative mapreduce, in *Proc. 19*th *ACM Int. Symp. on High Performance Distributed Computing*,

  Chicago, IL, USA,2010, pp. 810–818.
- [3] Y. Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, HaLoop: Effificient iterative data processing on large clusters, *Proc. VLDB Endowm.*, vol. 3, nos. 1&2, pp. 285–296, 2010.
- [4] G. M. Essertel, R. Y. Tahboub, J. M. Decker, K. J. Brown, K. Olukotun, and T. Rompf, Flare: Optimizing apache spark with native compilation for scale-up architectures and medium-size data, in *Proc. of the 13*th *USENIX Conf. on Operating Systems Design and Implementation*, Berkeley, CA, USA, 2018, pp. 799–815.
- [5] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica, Shark: SQL and rich analytics at scale, in *Proc. 2013 ACM SIGMOD Int. Conf. on Management of Data*, New York, NY, USA, 2013, pp. 13–24

THANK YOU