Optimisation of a Linear Algebra Approach to OLAP

Filipe Oliveira - A57816
a57816@alunos.uminho.pt

Sérgio Caldas - A57779
a57779@alunos.uminho.pt

July 18, 2016

Abstract

Online Analytical processing (OLAP) systems, perform multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modeling. All the referred analysis depends on Relational Algebra, which lack algebraic properties, and qualitative and quantitative proofs for all the relational operator. The proposed solution focus on a typed linear algebra approach, encoding OLAP functionality solely in terms of Linear Algebra operations (matrices).

*It has been argued that linear algebra (LA) is better suited than standard relational algebra for formalizing and implementing queries in on-line multidimensional data analysis [**?**] [**?**]. This can be achieved over a small LA sparse matrix kernel which, further to multiplication and transposition, offers the Kronecker, Khatri-Rao and Hadamard products. We give preliminary experimental results obtained with one cluster of Search6(Ref) using queries of the TPC-H Benchmark(Ref).*

I. Introduction

The design and development of systems that generate, collect, store, process, analyse, and query large sets of data is filled with significant challenges both hardware and software. Combined, these challenges represent a difficult landscape for software engineers.

The relation database is the current solution for big data storage. Data dependencies in databases can be seen as a binary two-way associative relation. Regarding that lema, prior efforts have been made [**?**] [**?**] in the research project "Linear Algebra approach to OLAP", in order to fully represent relational algebra in terms of linear algebra operators.

OLAP is resource-demanding and calls for parallelisation. Regarding the challenge of High Performance Computing, we implemented from the start a typed linear algebra solution given special importance to the data modelling which, if done poorly, limits the attainable efficiency in data-intesive systems like OLAP that tends to access massive amounts of data and is thus time consuming.

With respect to performance evaluation and results validation in a real work scenario, the used datasets were produced with TPC-H Benchmark, in which is workload consists of multiple query runs. In order to obtain realistic and meaningful results large datasets were considered, ranging from 1 to 64GB.

In order to infer conclusions and compare relational and linear algebra the object-relational database management system PostgreSQL version 9.6, with roots in open source community, was chosen in order to represent the relational algebra approach.

Given this proximity between database relations and linear algebra, the question arises: does the linear algebra approach presents performance improvements when compared with the relational one?

The report is organised as follows. Section 2 introduces XXXX. Section 3 presents XXXX. Section 4 gives experimental results. Section 5 presents related work. Section 6 concludes.

II. Linear Algebra Strategy

I. Towards a linear algebra semantics for SQL

Inspired by point-free relational data processing, an alternative roadmap for parallel online analytical processing (OLAP) can be achieved based on encoding data in matrix format and relying thereupon solely on LA operations[**?**].

I.1 Encoding data in matrix format

As example of raw data consider the displayed table 1 where each row records the order key, quantity, return flag, line status and ship date from an given TPC-H benchmark lineitem table. In order to facilitate data association the column number in the corresponding lineitem table was included above each data column.

Table 1: *Collection of raw data (adapted from TPC-H benchmark lineitem table).*

#1 l_orderkey	#5 l_quantity	#9 l_returnflag	#10 l_linestatus	#11 l_shipdate
1	17	N	O	1996-03-13
1	36	N	O	1996-04-12
1	8	N	O	1996-01-29
1	28	N	O	1996-04-21
1	24	N	O	1996-03-30
1	32	N	O	1996-01-30
2	38	N	O	1997-01-28
3	45	R	F	1994-02-02
3	49	R	F	1993-11-09
3	27	A	F	1994-01-16

To obtain useful information from raw data, which in OLAP systems escalated to Terabytes of information, we need to summarise the data by selecting attributes of interest and exhibiting their inter-relationships.

Consider the following simplified TPC-H query 1: "How many items were sold per return flag and line status?". For this particular question, the necessary attributes to answer the query are present on table lineitem, being **return flag**, **line status**, and **quantity**. In relational algebra that question could be easily answered by the following SQL code:

```
SELECT l_returnflag, l_linestatus,  sum(↩
    l_quantity)
FROM LINEITEM_SAMPLE
GROUP BY l_returnflag, l_linestatus;
```

which would produce the result presented on table 2

Table 2: *Simplified query-1 result from the collection of raw data (adapted from TPC-H benchmark lineitem table).*

#9 l_returnflag	#10 l_linestatus	#5 l_quantity
N	O	183
R	F	94
A	F	27

Aggregations like the presented on the prior simplified query occur in all TPC-H queries, hence performance of group-by and aggregation is quite important. That matter will be addressed in the later sections of the report.

Regarding expressing the OLAP in therms of LA, the key resides in expressing operations in the form of matrix algebra expressions. In this particular example, we should be able to build three matrices, one for each attribute, with each matrix being correlated to relational algebra as the row storage of columns #5, #9, and #10. However, in order to do so, we need to find a two-way association between the presented string on the rows #9 and #10 and an unique integer identifier. Our proposed solution encodes strings recurring to Gnome **Gquarks** [**?**] - a two-way association between a non-zero unsigned int and a char * - based on a thread safe hashtable.

By order of appearance, each unique string will be associated to an unique unsigned integer. Repeated strings will be associated to the prior corresponding unsigned integer. The resulting unsigned integer value range will start in the number 1. The value 0 in GQuarks is associated to NULL. Since in the proposed solution row and column numbers will respect C-style arrays notation both column and row numbering will start on 0. The GQuark unsigned integer value decremented by 1 will represent the row position on the matrix, and the register number, starting at 0, will represent the column position on the matrix.

We can now verify that there will be only one element per column of the matrix in order to maintain the two-way association between an register and its corresponding string. There is also the possibility of direct associating an register number with a value, wether integer or floating point. The referred matrices will be diagonal matrices, in which the element value is directly associated with the register.

Two types of matrices have now been presented:

1. Projection Matrices - which recur to Gnome Gquarks, in which the Gquark decremented by 1, represents the row position of the matrix, and the register number, starting at 0, will represent the column position on the matrix.

2. Measure Matrices - direct associating an register number with a value, wether integer or floating point, which the element value is directly associated with the register number, and consequently the column and row position.

We can now build the necessary matrices for the given examples. The two-way association between char* and unsigned integer is also presented in table 3 in order to aid the example comprehension.

Table 3: *Two-Way association between GQuarks and Row Number, for rows #9 and #10 presented in the collection of raw data (adapted from TPC-H benchmark lineitem table).*

String	GQuark	Row #
N	1	0
R	2	1
A	3	2
O	4	3
F	5	4

III. Hardware Characterization

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[1] Peak Memory BW	58.5GB/s

Table 4: *Architectural characteristics of the evaluation platform.*

IV. Hardware Characterization

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[2] Peak Memory BW	58.5GB/s

Table 5: *Architectural characteristics of the two evaluation platforms.*

V. Hardware Characterization

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[3] Peak Memory BW	58.5GB/s

Table 6: *Architectural characteristics of the two evaluation platforms.*

VI. Hardware Characterization

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[4] Peak Memory BW	58.5GB/s

Table 7: *Architectural characteristics of the two evaluation platforms.*

VII. Hardware Characterization

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[5] Peak Memory BW	58.5GB/s

Table 8: *Architectural characteristics of the two evaluation platforms.*

VIII. Acknowledgements

[**?**] Aggregation Performance.

Database, application, and storage servers ship with a large number of configu- ration parameters like buffer cache sizes, number of I/O daemons, and parameters input to the database query opti- mizer?s cost model. Finding good settings for these param- eters is a challenging task because of the complex ways in which parameter settings can affect performance

The platform used by us for our study at Search6 is a dual-socket system equipped with two Intel® Ivy Bridge processors. The system, referenced as compute node 652-1, has two Intel® Xeon® E5-2670v2 (Ivy Bridge architecture) and features 64 GB of DDR3 RAM, supported at a frequency of XXXX MHz divided in 4 memory channels.

System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	320KB 32KB per core
L2 Cache	2560KB 256KB per core
L3 Cache	25600KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured[6] Peak Memory BW	58.5GB/s

Table 9: *Architectural characteristics of the two evaluation platforms.*

References

@articlemacedo2015linear, title=A linear algebra approach to OLAP, author=Macedo, Hugo Daniel and Oliveira, José Nuno, journal=Formal Aspects of Computing, volume=27, number=2, pages=283–307, year=2015, publisher=Springer @articleda2015benchmarking, title=Benchmarking a Linear Algebra Approach to OLAP Master Thesis, author=da Costa Pontes, Rogério António, year=2015 @articlemacedo10matrices, title=Matrices as arrows! a biproduct approach to typed linear algebra, 2010, author=Macedo, HD and Oliveira, JN, journal=Submitted to MPC, volume=10 @techreportmacedo2011middle, title=Do the middle letters of ?OLAP? stand for linear algebra (?LA?), author=Macedo, Hugo Daniel and Oliveira, José Nuno, year=2011, institu-

tion=Technical Report TR-HASLab: 04: 2011, INESC TEC and University of Minho, Gualtar Campus, Braga