Optimization of LA in OLAP

2nd general debriefing meeting

Filipe Oliveira Sérgio Caldas

Advisors: Alberto Proença and José Nuno Oliveira

Table of Contents

Actual Progress

Progress Until First Meeting

- LA Operations translated into our DSL
- Profiled the given dataset
- Our approach decisions
  - Use Sparse MKL Library
  - Use Quarks structure from Glib library
  - Use sparse matrices formats for matrices representations

Progress Until Now

- Sequential version of LA Operations implemented and optimized
- Querie-1 translated into our DSL
- Querie-1 First Results in our HPC test environment

Translating TPC-H Query-1

TPC-H Query-1 (simplified)

```
SELECT l_returnflag, l_linestatus, sum(l_quantity)
FROM LINEITEM_1
WHERE l_shipdate >= '1998-08-28' AND l_shipdate <= '1998-12-01'
GROUP BY l_returnflag, l_linestatus;
```

Translates into LA Operations:

$$\underbrace{(L_{ReturnFlag} \triangledown L_{LineStatus})}_{\text{projection}} \cdot \underbrace{[L]^{Shipdate>=1998-08-28}_{Shipdate} \cdot [L]^{Shipdate<=1998-12-01}_{Shipdate}}_{\text{selection}} \cdot \underbrace{[\![L]\!]_{Quantity} \cdot !^{\circ}}_{\text{aggregation}}$$

Translating TPC-H Query-1

Translates into our DSL:

```
BEGIN
bitmap returnFlag, lineStatus, shipdate_gt, shipdate_lt;
matrix quantity;

returnFlag = tbl_read('lineitem.tbl', 8);
lineStatus = tbl_read('lineitem.tbl', 9);
quantity = tbl_read('lineitem.tbl', 4);
shipdate_gt = tbl_filter('lineitem.tbl', 10, >=, '1998-08-28' );
shipdate_lt = tbl_filter('lineitem.tbl', 10, <=, '1998-12-01' );


bitmap projection, selection;
matrix aggregiation, intermediate_result, final_result;

vector bng;
bng = bang(6);

selection = shipdate_gt * shipdate_lt;
projection = returnFlag krao lineStatus;
aggregation = quantity * bng;
intermediate_result = projection * selection;
final_result = itermediate_result * aggregation;

tbl_write(final_result, 'final_result.tbl');
END
```

Implementing and optimizing the LA operations

- Several LA approaches were considered
- La operations required: dot product, Kronecker Product, Hadamard Product, Khatri-Rao Product
- Based on the special matrices property **(1 element per column)** all 4 products were simplified **(not enough to auto-vectorize)**
  - All produced matrices have to be squared (if not pad with 0s)
- Opportunity to use SSE/AVX vector instructions in all 3 products
  - Enforced vectorization of loops
  - All Data is aligned during creation accordingly to cache line size in order to assist vectorization **(Allocation alignment)**
  - Instructed the compiler to assume that all CSR arrays are aligned on an 32-byte boundary **(Access alignment)**

Implementing and optimizing the LA operations

- Khatri-Rao (used in query-1) optimization report:

```
Begin optimization report for: csr_krao(float *, int *, int *, int, int, int, float *, int *, int *, int, int,

    Report from: Vector optimizations [vec]


LOOP BEGIN at olap_search.c(538,13)
   remark #15388: vectorization support: reference C_IA1 has aligned access
   remark #15388: vectorization support: reference A_IA1 has aligned access
   remark #15388: vectorization support: reference C_csc_values has aligned access   [ olap_search.c(540,20) ]
   remark #15388: vectorization support: reference B_csc_values has aligned access   [ olap_search.c(540,20) ]
   remark #15388: vectorization support: reference A_csc_values has aligned access   [ olap_search.c(540,20) ]
   remark #15388: vectorization support: reference C_JA1 has aligned access   [ olap_search.c(541,13) ]
   remark #15388: vectorization support: reference B_JA1 has aligned access   [ olap_search.c(541,13) ]
   remark #15388: vectorization support: reference A_JA1 has aligned access   [ olap_search.c(541,13) ]
   remark #15305: vectorization support: vector length 8
   remark #15309: vectorization support: normalized vectorization overhead 0.200
   remark #15301: FUSED LOOP WAS VECTORIZED
   remark #15448: unmasked aligned unit stride loads: 5.
   remark #15449: unmasked aligned unit stride stores: 3.
   remark #15475: --- begin vector loop cost summary ---
   remark #15476: scalar loop cost: 18.
   remark #15477: vector loop cost: 3.120.
   remark #15478: estimated potential speedup: 5.380.
   remark #15488: --- end vector loop cost summary ---
LOOP END
```

Dataset Characterization

TPC-H Benchmark Tables

- Used in TPC-H Querie-1:
  - **lineitem.tbl**
- Other TPC-H Benchmark Tables:
  - customer.tbl
  - nation.tbl
  - part.tbl
  - region.tbl
  - orders.tbl
  - partsupp.tbl
  - supplier.tbl

Datasets

- Generated data for different sizes: 1GB, 2GB, 4GB, 8GB, 16GB and 32 GB, producing 6 distinct LINEITEM DB tables.
- To speed up queries, indexes were created for all DB Tables:
  - f.e.: CREATE INDEX l_orderkey_idx8 on LINEITEM_8 (l_orderkey);

Environment Test Platform (Hardware)

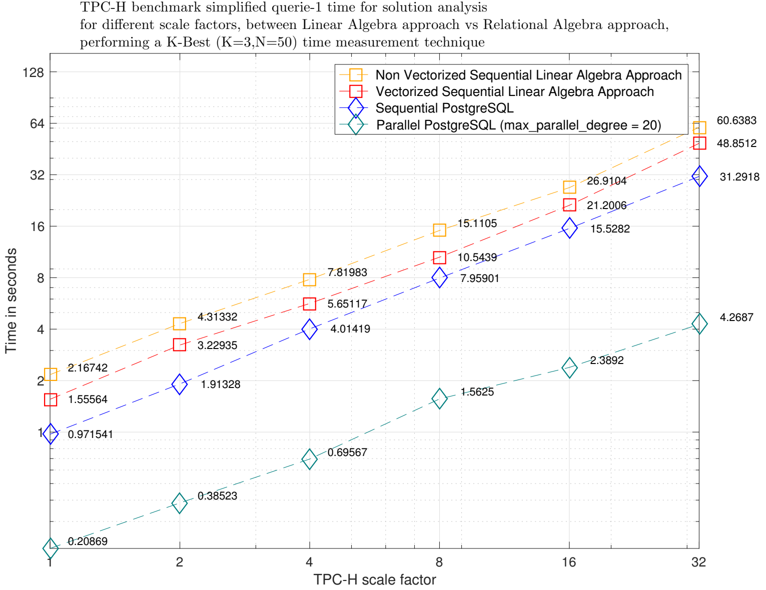
System	compute-652-1
# CPUs	2
CPU	Intel® Xeon® E5-2670v2
Architecture	Ivy Bridge
# Cores per CPU	10
# Threads per CPU	20
Clock Freq.	2.5 GHz
L1 Cache	10 x 32 KB instruction caches 10 x 32 KB data caches
L2 Cache	2,560 KB 256 KB per core
L3 Cache	25 MB
Inst. Set Ext.	AVX, SIMD
#Memory Channels	4
Vendors Announced Peak Memory BW	59.7 GB/s
Measured Peak Memory BW	57.18 GB/s

Environment Test Platform (Software)

- LA OLAP:
  - Compiler: ICC version 16.0.0 (GCC version 4.4.6 compatibility)
    - no vectorization: -O3 -std=c99 -no-vec -farray-notation
    - vectorization: -O3 -std=c99 -farray-notation -xAVX -vec-report7
  - Intel® MKL Version 11.3
    - Link line: -lmkl_intel_lp64 -lmkl_core -lmkl_sequential -lpthread -lm
- PostgreSQL: version 9.6+
  - Built with the following dependencies:
    - GCC version 4.9.0
    - Python 2.6.6
  - Why PostgreSQL 9.6+ ?
    - Open-Source
    - PostgreSQL has Parallel-Query available[1]
- Benchmarking:
  - 22 TPC-H Benchmark queries

[1] http://www.postgresql.org/docs/9.6/static/runtime-config-resource.html

Query-1 First Results in our HPC test environment



Planning our next steps

- Multithreaded parallelism
  - Expected high speedup values
- Improve data locality
  - Implement all products in BSR format
  - Irregular access patterns decreases performance
- Analyse Gather/Scatter analysis for distributed memory parallelism
  - Detect possible latency or bandwidth issues

Optimization of LA in OLAP

2nd general debriefing meeting

Filipe Oliveira   Sérgio Caldas

Advisors: Alberto Proença and José Nuno Oliveira