Simulating of query processing on multiprocessor database systems with modern coprocessors

Konstantin Y. Besedin*, Pavel S. Kostenetskiy**
* South Ural State University, Chelyabinsk, Russia
besedin.k@gmail.com
** South Ural State University, Chelyabinsk, Russia
kostenetskiy@gmail.com

Abstract - The modern manycore coprocessors and GPUs demonstrate very high performance on certain problems. Recent research has shown that these coprocessors can be used to accelerate database operations. But, to the best of our knowledge, there is only a little prior work on using coprocessors in multiprocessor database systems. This paper focuses on evaluation of database multiprocessor architectures with manycore coprocessors and GPUs. We implemented the emulator of parallel DBMS that uses computing cluster with NVIDIA GPUs or Intel Xeon Phi coprocessors for relational query processing. It allows to simulate simple SELECT and JOIN queries. A number of experiments have been done using this emulator. These experiments have shown that coprocessors are less efficient than CPUs for processing modeled SELECT queries, but more efficient than CPUs for processing INNER JOIN queries.

I. INTRODUCTION

The modern coprocessors like Intel Xeon Phi and NVIDIA CUDA demonstrate very high performance on certain problems. Database community shows a growing interest in applying this power to database systems [1]. Recent research has shown that these coprocessors can be used to accelerate database operations. For example, paper [2] covers relational join algorithms. Architecture-sensitive search tree is presented in [3], [4]. Papers [5], [6] focus on sorting algorithms. Paper [7] describes speculative processing of database operations. To the best of our knowledge, there is only a little prior work on using coprocessors to accelerate database operations in multiprocessor database systems. Modern scientific and engineering problems can imply storing and processing terabytes of data what requires use of high performance database multiprocessors. It is important to determine the appropriateness of using different types of coprocessors when constructing such systems. This paper focuses on evaluation of database multiprocessor architectures with manycore coprocessors and GPUs.

II. IMPLEMENTATION

To evaluate performance of different coprocessors in database multiprocessor systems we implemented the emulator of parallel DBMS. Emulator uses computing cluster with NVIDIA GPUs or Intel Xeon Phi coprocessors for relational query processing. It allows to simulate simple SELECT and JOIN queries. At this stage, for JOIN, only nested loops algorithm is implemented. Implementation was done with OpenMP, MPI and NVIDIA CUDA technologies. To achieve high level of parallelism, we use the principle of partitioned parallelism [8]. Each relation is partitioned into fragments, which are distributed over different worker processes (parallel agents). Worker processes exchange data using MPI technology.

III. EXPERIMENTS

Experiments with CPUs and Intel Xeon Phi were done on Tornado Supercomputer in South Ural State University, Chelyabinsk, Russia [9]. Experiments with GPUs were done on computing cluster in Lobachevsky State University, Nizhniy Novgorod, Russia. Experiments have used two CPUs or one coprocessor on each computing node. Characteristics of hardware are summarized in table I. To simulate SELECT and JOIN queries processing, we ran our emulator on computing clusters. The following simple queries were simulated:

1. SELECT * FROM Relation WHERE Attribute = Value;
2. SELECT * FROM Inner LEFT JOIN Outer ON (Inner.Attribute1 = Outer.Attribute2);

In this paper we consider a select on relation with cardinality of 370M that has size 5.5 GB. Join is done on following two relations: inner relation with cardinality of 33500 and size of 0.5 MB and outer relation that has size of 500 MB and cardinality of 33.5 M. Each relation consists of two integer (int64_t) attributes. Experiments include two steps. The first step is to evaluate query processing performance on a single computing node and determine settings that give maximum performance. In the first step we have run emulator on single computing node and varied its settings. For CPU version of algorithms, we have varied number of MPI processes. For GPU version, we have varied number of CUDA threads. For Manycore

<table>
<thead>
<tr>
<th>Processor</th>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon X5680 3.33 GHz</td>
<td>Intel Xeon L5630 2.13 GHz</td>
<td></td>
</tr>
<tr>
<td>Number of processors/cores</td>
<td>2/12</td>
<td>2/8</td>
</tr>
<tr>
<td>RAM</td>
<td>24GB</td>
<td>24GB</td>
</tr>
<tr>
<td>Coprocessor</td>
<td>Intel Xeon Phi SE10X</td>
<td>NVIDIA Tesla X2070</td>
</tr>
<tr>
<td>Coprocessor memory</td>
<td>8GB</td>
<td>6GB</td>
</tr>
</tbody>
</table>
version, we have varied number of OpenMP threads. Results of the first step are summarized in table II. The second step is to evaluate query processing performance on a computing cluster, using settings from previous step. In the second step we have run emulator on computing cluster and varied it’s the number of nodes from 1 to 8. On each computing node, emulator has been run with settings from previous step. Results of second step are summarized in figure 1 and figure 2. As can be seen from figures, coprocessors are less efficient than CPUs for processing simulated SELECT query. Small speedup is explained by overhead costs for transferring data over the network and over PCI Express bus. In the case of INNER JOIN query, all versions of algorithms have similar speedup. However, coprocessors can process these queries faster, than CPUs. Thus, one can conclude that coprocessors can be effectively used for processing JOIN queries in multiprocessor database systems.

IV. CONCLUSION

This work focuses on evaluating performance of manycore coprocessors and GPUs in multiprocessor database systems. We have implemented the emulator of parallel DBMS. This emulator uses computing cluster with NVIDIA GPUs or Intel Xeon Phi coprocessors for relational query processing. All algorithms have been implemented for CPUs, NVIDIA GPUs and Intel Xeon Phi coprocessors. A number of experiments have been done using this emulator. These experiments have shown that coprocessors are less efficient than CPUs for processing modeled SELECT queries, but more efficient than CPUs for processing INNER JOIN queries. Results of experiments will be used for extending DMM (Database Multiprocessor Model) [8]. Future work directions include:

1. Performance evaluation of heterogeneous computing systems in database applications.
2. Implementation and evaluation of more complicated JOIN algorithms, such as Hash Join for different types of coprocessors.
3. Design and implementation of algorithms that can use CPUs and coprocessor simultaneously for processing database queries.

ACKNOWLEDGMENT

This work was supported in part by the Russian Foundation for Basic Research, project no. 14-07-00420 and the Russian Federation president grant no. MK-3711.2013.9.

TABLE II. BEST EMULATOR SETTINGS FOR A SINGLE COMPUTING NODE

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time(s)</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU JOIN</td>
<td>258</td>
<td>12 MPI processes</td>
</tr>
<tr>
<td>GPU JOIN</td>
<td>90</td>
<td>20/736 CUDA threads</td>
</tr>
<tr>
<td>Manycore JOIN</td>
<td>213</td>
<td>240 OpenMP threads</td>
</tr>
<tr>
<td>CPU SELECT</td>
<td>22.96</td>
<td>12 MPI processes</td>
</tr>
<tr>
<td>GPU SELECT</td>
<td>55.73</td>
<td>5120 CUDA threads</td>
</tr>
<tr>
<td>Manycore SELECT</td>
<td>69.84</td>
<td>140 MPI threads</td>
</tr>
</tbody>
</table>

REFERENCES
