Matrix Chain Multiplication via Multi-Way Join Algorithms in MapReduce

Jaeseok Myung

Intelligent Data Systems Lab
School of Computer Science & Engineering
Seoul National University, Seoul, Korea
Introduction

- Matrix multiplication is an important operation for many applications
  - Scientific Computing, Social Network Analysis, ...
  - Finding \( n \)-hop neighbors from a social network

![Diagram of a social network with nodes a, b, and c, and edges between a and b, b and c, and a and c.]

\[
\begin{array}{c|c|c}
 & a & b & c \\
\hline
a & 0 & 1 & 1 \\
b & 1 & 0 & 0 \\
c & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
 & a & b & c \\
\hline
a & 1 & 1 & 0 \\
b & 0 & 1 & 1 \\
c & 1 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
 & a & b & c \\
\hline
a & 1 & 1 & 1 \\
b & 1 & 1 & 1 \\
c & 1 & 1 & 0 \\
\end{array}
\]

- MapReduce has emerged as a scalable data processing framework
  - The size of data is rapidly increasing
  - MapReduce is a new programming model so it requires new algorithms

- In this paper, we focus on MR algorithms for the matrix multiplication
Matrix Multiplication using Join Operation

- Matrix multiplication is often translated into a join operation between two relations.
- Especially, join operation is useful for sparse matrices.
  - There are a number of sparse matrices in real-world.
- In this paper, we deal with the matrix multiplication via join algorithms in MapReduce.

```sql
SELECT M1.row, M2.col, sum(M1.val*M2.val) FROM M1, M2 WHERE M1.col=M2.row GROUP BY M1.row, M2.col
```

- Especially, join operation is useful for sparse matrices.
  - There are a number of sparse matrices in real-world.
- In this paper, we deal with the matrix multiplication via join algorithms in MapReduce.
Our Approach: Inter-Operation Parallelism

• Several studies have been published to improve performance of *multiplication between two matrices*

• However, we focus on *multiplication of several matrices*
  – Thus, we focus on the entire equation rather than an operation
    • The notion of inter-operation parallelism
    – Fortunately, matrix multiplication is associative

\[
ABCD = (((AB)C)D) = (AB)(CD) = (ABC)D = \cdots = (ABCD)
\]

• We believe that the notion of inter-operation parallelism is helpful for MapReduce and matrix chain multiplication
Iterative MapReduce

- Multiplication of several matrices requires MapReduce iteration
  - It is important to reduce the number of MapReduce jobs because MR iteration is usually inefficient
Three different methods for matrix chain multiplication

1. Serial 2-way (S2) \( (n-1) \) iterations
2. Parallel 2-way (P2) \( \log_2 n \) iterations
3. Parallel M-way (PM) \( \log_m n \) iterations

Figure 2: An example of parallel 3-way join. The number of MR iterations is \( \lfloor \log_3 5 \rfloor \)
Experiments – 1

- Amazon EC2 (4 units)
  - M (10,000 X 10,000)

- The result shows the importance of the inter-operation parallelism
Experiments – 2

- Amazon EC2
  - M (1,000,000 X 1,000,000)

- PM shows the worst performance
- S2 has iteration overhead, but it can take advantage of multiple machines
- P2 shows the best performance

<table>
<thead>
<tr>
<th></th>
<th>4 units</th>
<th>8 units</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>710</td>
<td>373</td>
</tr>
<tr>
<td>P2</td>
<td>310</td>
<td>304</td>
</tr>
<tr>
<td>PM</td>
<td>788</td>
<td>1030</td>
</tr>
</tbody>
</table>
Discussion

- Different MR implementations for 2-way and m-way join
  - MR programming model is based on the key-value model
  - Join operation should be implemented like a hash join algorithm
    - Mapper makes a partition of records that shares the same join key
    - Reducer actually makes a join result for the partition
Overhead in Hadoop's MR Implementation - Sorting

- Mappers always do sorting for every partition even if records in the partition will be discarded
  - Only thing we expect to the mapper is just transfer a record to the right reducer

![Diagram showing the process of Mappers and Reducers in Hadoop's MR Implementation with overhead in sorting.]
Discussion

- Serial 2-way Join
  - Several iterations
  - No duplication

- Parallel 2-way Join
  - No duplication
  - Reduced iterations

- Parallel m-way join
  - No iterations
  - Duplication

- Parallel 2-way join balances the intra-operation parallelism and the inter-operation parallelism, so it shows the best performance
Conclusion

• Contribution
  – The first MR approach to the matrix chain multiplication
  – Applying the MR multi-way join algorithm to the matrix chain multiplication
    • Implementation of S2 / P2 / PM
  – Experiments
    • Analysis of limitations

• Two Types of Overhead for Matrix Chain Multiplication with MR
  – Overhead between MR jobs -> this paper
    • Experimental results shows that P2 shows the best performance
  – Overhead within a MR job (Sorting) -> Future work
    • There have been several studies for efficient MR job execution
      – HaLoop, HOP, ...
Appendix
What is MapReduce?

- MapReduce a parallel programming model based on the notion of functional programming
  - Map \((k_1, v_1)\)  \(\rightarrow\) list\((k_2, v_2)\)
  - Reduce \((k_2, \text{list}(v_2))\)  \(\rightarrow\) list \((k_3, v_3)\)
Related Work

• Several studies exist to optimize multiplication of two matrices
  – A MapReduce Algorithm for Matrix Multiplication, 2009
    • http://homepage.mac.com/j.norstad/matrix-multiply/index.html
  – PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations, ICDM 2009
  – SystemML: Declarative Machine Learning on MapReduce, ICDE 2011

• Multiplication of several matrices requires MapReduce iteration
  – It is important to reduce the number of MapReduce jobs because MR iteration is inefficient
A Multi-way Join Algorithm in MapReduce

\[ R(A, B) \bowtie S(B, C) \bowtie T(C, D) \]

- Let \( h \) be a hash function with range 1, 2, ..., \( m \)
  - \( S(b, c) \rightarrow (h(b), h(c)) \)
  - \( R(a, b) \rightarrow (h(b), \text{all}) \)
  - \( T(c, d) \rightarrow (\text{all}, h(c)) \)

- [EDBT 10] minimize the expression
  - \( r|h(c)| + s + t|h(b)| \)
  - where \(|h(c)| \times |h(b)| = k\)

Reduce processes
(# of Reduce processes: \(4^2 = 16\))
m=4, k=16