A Novel Multi-CPU/GPU Collaborative Computing Framework for SGD-based Matrix Factorization

Yizhi Huang*, Yinyan Long†, Yan Liu*,
Shuibing He‡, Yang Bai*, Renfa Li*

* HUNAN UNIVERSITY  † ZHEJIANG LAB  ‡ ZHEJIANG UNIVERSITY
Outline

• Background and Motivation

• Design and Implementation

• Evaluation
Background

- Matrix Factorization: can help recommender systems predicted user’s preferences to products.

- SGD-based MF

\[ \theta(p_i, q_j) = \frac{1}{2}(r_{i,j} - p_i \cdot q_j)^2 + \lambda_1 \|P\|^2 + \lambda_2 \|Q\|^2 \]

\[ p_i \leftarrow p_i - \gamma \frac{\partial \theta(p_i, q_j)}{\partial p_i} \]

\[ q_j \leftarrow q_j - \gamma \frac{\partial \theta(p_i, q_j)}{\partial q_j} \]

Iteration

Each score \( r \) will be used to update two \( k \)-dimensional vectors \( p, q \).

Need to accelerate SGD-based MF
Observation: the Under-utilized CPUs

- Many computing nodes have multi-CPUs/GPUs
- Existing researches more willing to manage the GPUs for computing
- CPUs’ computing power is easily overlooked
- Is it possible to cooperate with the CPUs to accelerate SGD-based MF?
• The performance of high-end GPUs does not increase linearly with price

• Cooperative computing of CPU and GPU may bring a good price/performance ratio
Challenges

- How to uniformly manage and transparently use heterogeneous CPUs and GPUs?
- How to design appropriate data distribution?
- How to optimize communication inter-CPUs/GPUs?

Unbalanced load leads to short board effect

Naïve Communication Cost: \( R_{m \times n} = P_{m \times k} \times Q_{k \times n} \)

Netlix: \( m = 480190, n = 17771, k = 128, \text{iterations} = 20, \text{cost} = 0.4s \)
Outline

• Background and Motivation

• Design and Implementation

• Evaluation
Our solution: HCC-MF

Problem 1
How to transparentize heterogeneous CPUs and GPUs

A general framework that unifies the abstraction and workflow

Problem 2
How to distribute data to each heterogeneous CPU/GPU to make the whole system more efficient?

- A time cost model for guiding data distribution.
- Two data partition strategies to deal with different synchronization overhead conditions

Problem 3
How to optimize communication Inter-CPUs/GPUs?

Communication optimization strategies that reduce the amount of data transmission and use computation to overlap communication
HCC-MF

- Heterogeneous CPUs/GPUs are abstracted into worker processes
- Use shared memory as a COMM channel between processes
- Server assigns data to workers, workers asynchronously calculate SGD-based MF
- Workers: Pull -> Computing -> Push
- Servers: Synchronization $\sum_{i=1}^{p} (P_i + Q_i)/p$
Our solution: HCC-MF

Problem 1
How to transparentize heterogeneous CPUs and GPUs

A general framework that unifies the abstraction and workflow

Problem 2
How to distribute data to each heterogeneous CPU/GPU to make the whole system more efficient?

- A time cost model for guiding data distribution.
- Two data partition strategies to deal with different synchronization overhead conditions

Problem 3
How to optimize communication Inter-CPUs/GPUs?

Communication optimization strategies that reduce the amount of data transmission and use computation to overlap communication
Time Cost Model

\[ T = \max \{ T_i \} + T_{\text{sync}} \]

- \( T_i \gg B_i \)

Omit performance-related components

\[ T = \max \left\{ \frac{x_i \text{nnz} (16k + 4)}{B_i} \right\} + \frac{2k(m + n)}{B_{\text{bus}_i}} + \frac{3tk(m + n)}{B_{\text{server}}} \]

Can sync be ignored?

Worker 0
- Pull
- Computing
- Push

Worker 1
- Pull
- Computing
- Push

Worker 2
- Pull
- Computing
- Push

Worker 3
- Pull
- Computing
- Push

Worker 4
- Pull
- Computing
- Push

Worker \( i \)
- Computational complexity: \( 7kx_i \text{nnz} \)
- Memory access complexity: \( (16k + 4)x_i \text{nnz} \)
- Transmission complexity: \( 2k(m + n) \)
Data partition for load balance

\[
\theta(x) = \min\{T\} = \min \left\{ \max \left\{ \frac{x_i \cdot \text{nnz}(16k + 4)}{B_i} \right\} + \frac{2k(m + n)}{B_{bus_i}} \right\}
\]

\[
\theta(x) = \min\{\max(Ax + B)\}
\]

Assuming \(B_i\) is a constant function of \(x_i\)

Can DP0 really guarantee load balance?

\[
a_1x_1 + b_1 = a_2x_2 + b_2 = \cdots = a_nx_n + b_n, \theta \text{ is the minimum}
\]

\[
b_1 \approx b_2 \approx \cdots \approx b_n
\]

\[
\text{DP}_0: \ x_i = \frac{1}{\sum_{j=1}^{p} a_j} = \frac{1}{\sum_{j=1}^{p} \frac{T_i e}{T_{j e}}}
\]
Data partition for load balance

- The assumption of $B_i$ is not true
- The Runtime performance may not be ignored

**Differential**  

if $\Delta x$ is small, $\Delta T$ can be regarded as linear

Few iterations

DP0  ->  Algorithm 1  ->  DP1

Algorithm 1 Compensation algorithm

Input: Old data partition $\{x_1, x_2, ..., x_p\}$; The computing time $\{t_1, t_2, ..., t_p\}$

Output: New data partition $\{x_1, x_2, ..., x_p\}$

1. $T_{\text{prev}} = \frac{1}{p} \sum_{i=1}^{p} t_i$
2. $T_{\text{prev}} = \frac{1}{p} \sum_{i=1}^{p} (x_i + \Delta t_i)$
3. while $\min\{T_{\text{prev}} - T_{\text{prev}}\} > 0.1$ do
   4. $T_{\text{prev}} = \frac{1}{p} \sum_{i=1}^{p} (x_i + \Delta t_i)$
   5. for $i = 1, c$ do
      6. $x_i_{\text{new}} = \frac{x_i_{\text{prev}} + x_i_{\text{new}}(x_i - x_i_{\text{new}})}{x_i_{\text{prev}}}$
   end for
   7. for $j = 1, g$ do
      8. $x_j_{\text{new}} = \frac{x_j_{\text{prev}} + x_j_{\text{new}}}{x_j_{\text{prev}}}$
   end for
10. $\{x_1, x_2, ..., x_p\} \leftarrow \{x_1_{\text{new}}, x_2_{\text{new}}, ..., x_p_{\text{new}}\}$
11. $T_{\text{prev}} = \frac{1}{p} \sum_{i=1}^{p} (x_i_{\text{new}} + \Delta T)$
12. return $\{x_1, x_2, ..., x_p\}$
Data partition: hiding synchronization

\[ T = \max \left\{ \frac{x_i \text{nnz}(16k + 4)}{B_i} + \frac{2k(m + n)}{B_{bus_i}} \right\} + \frac{3tk(m + n)}{B_{server}} \]

\( t \) is a nonlinear function of \( x \)

Difficult to solve the objective function

Use DP1 to balance the computational overhead of each worker

\[ T_1 = T_2 = \cdots = T_n \]

Use calculation to hide synchronization overhead

\[ T_{(i+n)} = T_i \pm nT_{i,\text{sync}} \]

Server

Worker 0

Worker 1

Worker 2

Worker 3

Worker 4
Our solution: HCC-MF

Problem 1
How to transparentize heterogeneous CPUs and GPUs

A general framework that unifies the abstraction and workflow

Problem 2
How to distribute data to each heterogeneous CPU/GPU to make the whole system more efficient?

- A time cost model for guiding data Distribution.
- Two data partition strategies to deal with different synchronization overhead conditions

Problem 3
How to optimize communication Inter-CPUs/GPUs?

Communication optimization strategies that reduce the amount of data transmission and use computation to overlap communication
Reduce data transmission

Rows (columns) are independent of each other

Transmitting Q matrix only

The data range of the rating matrix is limited

Transmitting FP16 Data

Rating Matrix $R$

User Matrix $P$

Item Matrix $Q$

\[ R \approx \hat{R} = P \times Q \]

\[ k \]
Overleap communication

Multiple Asynchronous computing-transmission streams in worker

GPU: copy engine

CPU: multithreads and free bandwidth

SoC: copy engine in iGPU
Outline

• Background and Motivation

• Design and Implementation

• Evaluation
Evaluation Setup

<table>
<thead>
<tr>
<th>Item</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>2 Intel(R) Xeon(R) Gold 6242, Nvidia RTX 2080S, Nvidia Rtx 2080</td>
</tr>
<tr>
<td>DataSet</td>
<td>Netflix, Yahoo Music R1, R2, R1*, Movielens-20m</td>
</tr>
<tr>
<td>Baseline</td>
<td>FPSGD and cuMF_SGD we implemented</td>
</tr>
</tbody>
</table>

- We do not change the core idea of the baseline algorithm in our implementation
- We optimized the code to make the baseline execute faster
- We use baseline as the kernel running on the worker
Overall performance

---

Same convergence rate

Faster training speed

Netflix

R1

R2
Data partition evaluation

DP0 can only guarantee load balancing on similar processors

DP1 can guarantee load balance on all processors
- Netflix-4workers: -12.2%
- R2-4workers: -10%

DP2 can hide synchronization overhead
- R1*-4workers: -12.1%
Without any communication optimization, the communication overhead will offset the benefits brought by parallelism.

Q can achieve better optimization results, but the effectiveness depends on the shape of the rating matrix.

The transmission performance of half-q is more than twice that of Q.
Conclusion

HCC-MF: A heterogeneous multi-CPU/GPU collaborative computing framework for SGD-based matrix factorization

- Unified workflow and transparent heterogeneous CPUs/GPUs usage
- Data distribution algorithm for different synchronization conditions
- Optimal inter-CPUs/GPUs communication

Limitation (Under study):

- Communication overhead can be further optimized
- Server bottleneck
Thank you

Yizhi Huang
huangyizhi @hnu.edu.cn