Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds

Kevin Hsieh
Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, Onur Mutlu†

Carnegie Mellon University  
†ETH Zürich
Machine learning is widely used to derive useful information from large-scale data.
Big Data is Geo-Distributed

• A large amount of data is generated *rapidly*, all over the world
Centralizing Data is Infeasible \([1, 2, 3]\)

- Moving data over wide-area networks (WANs) can be extremely slow
- It is also subject to data sovereignty laws

1. Vulimiri et al., NSDI’15
2. Pu et al., SIGCOMM’15
3. Viswanathan et al., OSDI’16
Geo-distributed ML is Challenging

• No ML system is designed to run across data centers (up to 53X slowdown in our study)
Our Goal

• Develop a geo-distributed ML system
  • Minimize communication over wide-area networks
• Retain the accuracy and correctness of ML algorithms
• Without requiring changes to the algorithms

Key Result: 1.8-53.5X speedup over state-of-the-art ML systems on WANs
Outline

• Problem & Goal
• Background & Motivation
• Gaia System Overview
• Approximate Synchronous Parallel
• System Implementation
• Evaluation
• Conclusion
The parameter server architecture has been widely adopted in many ML systems.
The parameter server architecture has been widely adopted in many ML systems. Synchronization is critical to the accuracy and correctness of ML algorithms.
Deploy Parameter Servers on WANs

- Deploying parameter servers across data centers requires **a lot of communication** over WANs
WAN: Low Bandwidth and High Cost

- WAN bandwidth is **15X smaller** than LAN bandwidth on average, and up to **60X smaller**
- In Amazon EC2, the **monetary cost** of WAN communication is up to **38X** the cost of renting machines
ML System Performance on WANs

Matrix Factorization

<table>
<thead>
<tr>
<th>Environment</th>
<th>IterStore</th>
<th>Bösen</th>
<th>Normalized Execution Time until Convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>3.7X</td>
<td>5.9X</td>
<td>3.7X 5.9X</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>3.5X</td>
<td>4.4X</td>
<td>3.5X 4.4X</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>23.8X</td>
<td>24.2X</td>
<td>23.8X 24.2X</td>
</tr>
</tbody>
</table>

1) Cui et al., “Exploiting Iterative-ness for Parallel ML Computations”, SoCC’14
2) Wei et al., “Managed Communication and Consistency for Fast Data-Parallel Iterative Analytics”, SoCC’15
Running ML systems on WANs can seriously slow down ML applications

1) Cui et al., “Exploiting Iterative-ness for Parallel ML Computations”, SoCC'14
2) Wei et al., “Managed Communication and Consistency for Fast Data-Parallel Iterative Analytics”, SoCC'15

11 EC2 Regions

Singapore / São Paulo

Virginia / California

Matrix Factorization

IterStore

Bösen

23.8X

24.2X
Outline

- Problem & Goal
- Background & Motivation
- Gaia System Overview
- Approximate Synchronous Parallel
- System Implementation
- Evaluation
- Conclusion
Gaia System Overview

• **Key idea:** Decouple the synchronization model *within* the data center from the synchronization model *between* data centers.
• **Key idea**: Decouple the synchronization model *within* the data center from the synchronization model *between* data centers.
Key Finding: Study of Update Significance

The vast majority of updates are insignificant
• Problem & Goal
• Background & Motivation
• Gaia System Overview
• Approximate Synchronous Parallel
• System Implementation
• Evaluation
• Conclusion
The significance filter

- Filter updates based on their significance

ASP selective barrier

- Ensure significant updates are read in time

Mirror clock

- Safe guard for pathological cases
The Significance Filter

Worker Machine

Update ($\Delta_2$) on X

Parameter Server

Parameter X

Value

Aggregated Update

Other Parameters

Significance Function

$\frac{\text{Agg. Update}}{\text{Value}}$

Significance Threshold

$\frac{10\%}{\sqrt{T}}$

Update ($\Delta_1$)

Aggregated Update

$\Delta_1 + \Delta_2$
Approximate Synchronous Parallel

The significance filter
- Filter updates based on their significance

ASP selective barrier
- Ensure significant updates are read in time

Mirror clock
- Safeguard for pathological cases
ASP Selective Barrier

Only workers that depend on these parameters are blocked
Outline

• Problem & Goal
• Background & Motivation
• Gaia System Overview
• Approximate Synchronous Parallel
  • System Implementation
• Evaluation
• Conclusion
Put it All Together: The Gaia System
Control messages (barriers, etc.) are always prioritized

No change is required for ML algorithms and ML programs
Communication overhead is proportional to the number of data centers
Save communication on WANs by aggregating the updates at hubs
Outline

• Problem & Goal
• Background & Motivation
• Gaia System Overview
• Approximate Synchronous Parallel
• System Implementation
• Evaluation
• Conclusion
Methodology

• **Applications**
  • Matrix Factorization with the Netflix dataset
  • Topic Modeling with the Nytimes dataset
  • Image Classification with the ILSVRC12 dataset

• **Hardware platform**
  • 22 machines with emulated EC2 WAN bandwidth
  • We validated the performance with a real EC2 deployment

• **Baseline**
  • IterStore (Cui et al., SoCC’14) and GeePS (Cui et al., EuroSys’16) on WAN

• **Performance metrics**
  • Execution time until algorithm convergence
  • Monetary cost of algorithm convergence
Gaia achieves 3.7-6.0X speedup over Baseline
Gaia is at most 1.40X of LAN speeds
Performance and WAN Bandwidth

Gaia achieves 3.7-53.5X speedup over Baseline
Gaia is at most 1.23X of LAN speeds
Results – EC2 Monetary Cost

Gaia is 2.6-59.0X cheaper than Baseline

Matrix Factorization
Topic Modeling
Image Classification
More in the Paper

• **Convergence proof** of Approximate Synchronous Parallel (ASP)

• ASP vs. fully asynchronous

• Gaia vs. centralizing data approach
Key Takeaways

• The Problem: How to perform ML on geo-distributed data?
  • Centralizing data is infeasible. Geo-distributed ML is very slow

• Our Gaia Approach
  • Decouple the synchronization model within the data center from that across data centers
    • Eliminate insignificant updates across data centers
    • A new synchronization model: Approximate Synchronous Parallel
      • Retain the correctness and accuracy of ML algorithms

• Key Results:
  • 1.8-53.5X speedup over state-of-the-art ML systems on WANs
  • at most 1.40X of LAN speeds
  • without requiring changes to algorithms
Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds

Kevin Hsieh
Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, Onur Mutlu†

Carnegie Mellon University †ETH Zürich
Executive Summary

• The Problem: **How to perform ML on geo-distributed data?**
  • Centralizing data is infeasible. Geo-distributed ML is very slow

• Our Goal
  • **Minimize communication** over WANs
  • Retain the **correctness and accuracy** of ML algorithms
  • **Without requiring changes** to ML algorithms

• Our **Gaia** Approach
  • **Decouple the synchronization model** within the data center from that across data centers: Eliminate insignificant updates on WANs
  • A **new synchronization model**: Approximate Synchronous Parallel

• Key Results:
  • **1.8-53.5X speedup** over state-of-the-art ML systems on WANs
  • **within 1.40X** of LAN speeds
Approximate Synchronous Parallel

The significance filter
- Filter updates based their significance

ASP selective barrier
- Ensure significant updates are read in time

Mirror clock
- Safeguard for pathological cases
Mirror Clock

Data Center 1

Data Center 2

Parameter Server

Parameter Server

Clock N

Clock N + DS

 Guarantees all significant updates are seen after DS clocks

No guarantee under extreme network conditions
Effect of Synchronization Mechanisms

Matrix Factorization

Topic Modeling
Methodology Details

• Hardware
  • A 22-node cluster. Each has a 16-core Intel Xeon CPU (E5-2698), a NVIDIA Titan X GPU, 64GB RAM, and a 40GbE NIC

• Application details
  • Matrix Factorization: SGD algorithm, 500 ranks
  • Topic Modeling: Gibbs sampling, 500 topics

• Convergence criteria
  • The value of the objective function changes less than 2% over the course of 10 iterations

• Significance Threshold
  • 1% and shrinks over time $\left(\frac{1%}{\sqrt{T}}\right)$
IterStore [Cui et al. SoCC’15] shows 10X performance improvement over PowerGraph [Gonzalez et al., OSDI’12] for Matrix Factorization.

PowerGraph matches the performance of GraphX [Gonzalez et al., OSDI’14], a Spark-based system.
Matrix Factorization (1/3)

- Matrix factorization (also known as collaborative filtering) is a technique commonly used in recommender systems.

<table>
<thead>
<tr>
<th>Movie</th>
<th>Alice (1)</th>
<th>Bob (2)</th>
<th>Carol (3)</th>
<th>Dave (4)</th>
<th>$x_1$ (romance)</th>
<th>$x_2$ (action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Love at last</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Romance forever</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Cute puppies of love</td>
<td>?</td>
<td>4</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Nonstop car chases</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Swords vs. karate</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Matrix Factorization (2/3)

Movie

User

$\rightarrow$

Rank (User Preference Parameters) ($\theta$)

Rank (Movie Parameters) (x)
Matrix Factorization (3/3)

• Objective function (L2 regularization)

\[
J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2
\]

• Solve with stochastic gradient decent (SGD)
Background – BSP

• BSP (Bulk Synchronous Parallel)
  • All machines need to receive all updates before proceeding to the next iteration

Always sends and reads all updates
Accurate but slow

Worker 3
Clock

Worker 1
0 1 2 3
Background – SSP

- **SSP (Stale Synchronous Parallel)**
  - Allows the fastest worker ahead of the slowest worker by a bounded number of iterations

![Diagram](image)

Alleviates the network bandwidth requirement, but still sends all updates

- Staleness = 1
## Compare Against Centralizing Approach

<table>
<thead>
<tr>
<th></th>
<th>Gaia Speedup over Centralize</th>
<th>Gaia to Centralize Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix Factorization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>1.11</td>
<td>3.54</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>2.13</td>
<td>1.17</td>
</tr>
<tr>
<td><strong>Topic Modeling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>0.80</td>
<td>6.14</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.02</td>
<td>1.26</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>1.25</td>
<td>1.92</td>
</tr>
<tr>
<td><strong>Image Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC2-ALL</td>
<td>0.76</td>
<td>3.33</td>
</tr>
<tr>
<td>V/C WAN</td>
<td>1.12</td>
<td>1.07</td>
</tr>
<tr>
<td>S/S WAN</td>
<td>1.86</td>
<td>1.08</td>
</tr>
</tbody>
</table>
SSP Performance – 11 Data Centers

Matrix Factorization

Normalized Execution Time

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
<td>2.0X 1.8X</td>
<td>2.0X</td>
<td>1.8X</td>
<td>1.5X 1.3X</td>
<td>3.0X</td>
<td>1.3X</td>
</tr>
<tr>
<td>SSP</td>
<td>3.8X</td>
<td>3.7X</td>
<td></td>
<td>1.5X</td>
<td>3.0X</td>
<td>2.7X</td>
</tr>
</tbody>
</table>

- Amazon-EC2
- Emulation-EC2
- Emulation-Full-Speed

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
SSP Performance – 11 Data Centers

Topic Modeling

Normalized Execution Time

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Emulation-EC2</td>
<td>2.0X</td>
<td>3.7X</td>
<td>4.8X</td>
<td>1.5X</td>
<td>3.5X</td>
</tr>
<tr>
<td></td>
<td>Emulation-Full-Speed</td>
<td>2.5X</td>
<td>2.0X</td>
<td>1.7X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Baseline, Gaia, LAN, BSP, SSP
SSP Performance – V/C WAN

Matrix Factorization

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
<td>3.7X</td>
<td>3.5X</td>
<td>2.6X</td>
</tr>
<tr>
<td>SSP</td>
<td>2.6X</td>
<td>2.3X</td>
<td>2.3X</td>
</tr>
</tbody>
</table>

Topic Modeling

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Gaia</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
<td>3.7X</td>
<td>3.9X</td>
<td>3.1X</td>
</tr>
<tr>
<td>SSP</td>
<td>3.1X</td>
<td>3.2X</td>
<td>3.2X</td>
</tr>
</tbody>
</table>
SSP Performance – S/S WAN

Matrix Factorization

- Baseline
- Gaia
- LAN

SSP:
- 16X 14X

BSP:
- 25X 24X

Topic Modeling

- Baseline
- Gaia
- LAN

SSP:
- 17X 21X

BSP:
- 14X 17X

Graphs showing performance comparison between Baseline, Gaia, and LAN for Matrix Factorization and Topic Modeling with respective performance metrics.