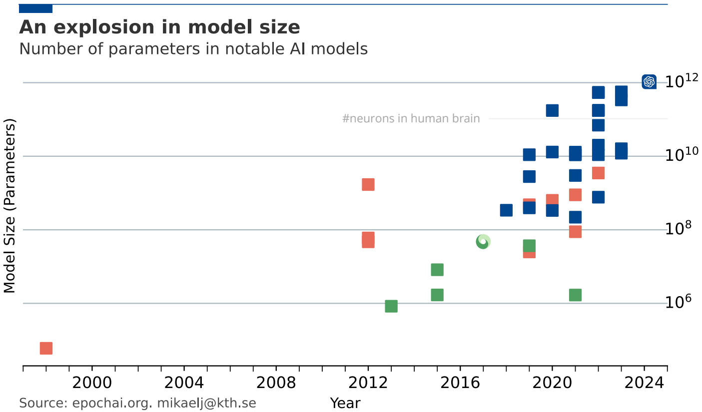
From Promise to Practice: Realizing High-performance Decentralized Training

International Conference on Learning Representations 2025

Zesen Wang[1], Jiaojiao Zhang[1], Xuyang Wu[2], and Mikael Johansson[1]
[1]KTH Royal Institute of Technology [2]Southern University of Science and Technology

The challenge of large-scale model training



Implies similar trends in training data (Chinchilla scaling: 20 tokens/parameter)

Decentralized training is promising, but with complex design space...

Advantages:

- Reduce communication overhead
- Enjoy same convergence rates as synchronous algorithms

Decentralized training is promising, but with complex design space...

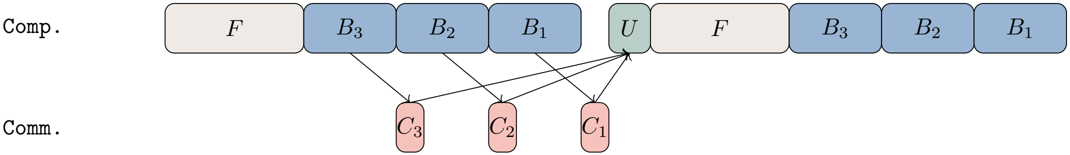
Complex design space:

- Decentralized communication topology
  Convergence rate in terms of the number of iterations or practical runtime
- Decentralized Algorithm
  Communication patterns, additional buffers to compensate the decentralized updates, etc..
- Orthogonal techniques
  local updates and communication compression

Problem: Complex design space leads to difficulties in practical implementation and underexplored potential speedup.

Concern: AllReduce training can be hard to beat...

AllReduce training is hard to beat with fast communication and homogeneous hardware and workloads.



Can happen with few workers, or in the very high-end of dedicated HPC clusters.

Important Aspects of Multi-node Decentralized Training

Decentralized training can enable:

- More overlapping between communication and computation
- Better utilization of heterogeneous communication links
- Better resilience to stragglers

1. More overlapping between communication and computation

Communicate-while-adapt decentralized algorithm:

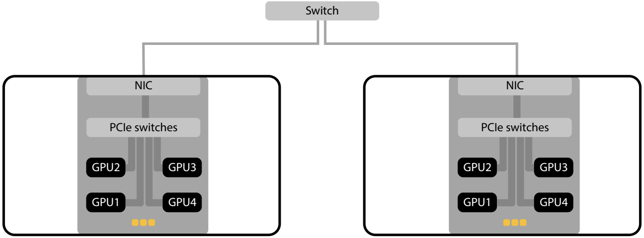
$$x_i^{(t+1)} = d_i\left(\nabla f(x_i^{(t)}; \xi_i^{(t)})\right) + \sum_{j \in n_i^{(t)}} w_{ij}^{(t)} x_j^{(t)} \tag{1}$$

- Red: local computation.
- Blue: decentralized communication with neighbors.

Decentralized communication can be launched in the last iteration and further overlap with the forward pass in the current iteration.

2. Heterogeneous communication links

Mixture of fast (NVLinks) and slow communication (inter-node Ethernet) links.



2. Heterogeneous communication links

Decentralized communication topologies could be tailored to utilize the heterogeneous communication environment.

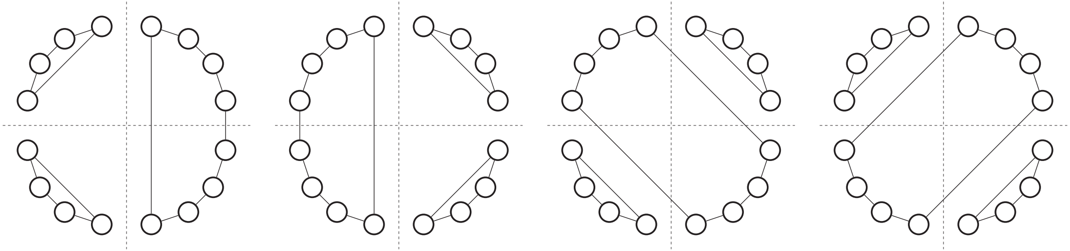
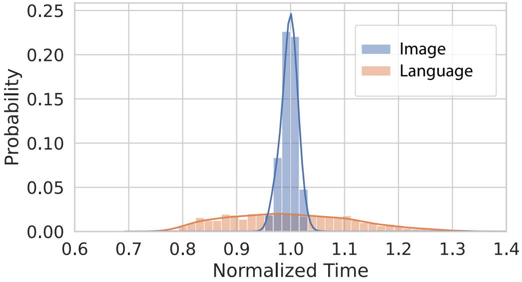


Figure: Alternating Exponential Ring (AER). Circles: workers (GPUs). Dashed lines: boundaries of the nodes. Solid lines: communication operations.

3. Stragglers caused by imbalanced Workloads and system noise

Specific tasks (machine translation in natural language, for example) may introduce imbalanced workloads or random stragglers caused by system noise.



Decentralized communication and asynchronous updates pose looser synchronization across workers, which provides better resilience to stragglers.

Timeline Comparisons

Significant speedup could be achieved when taking all aspects into consideration.

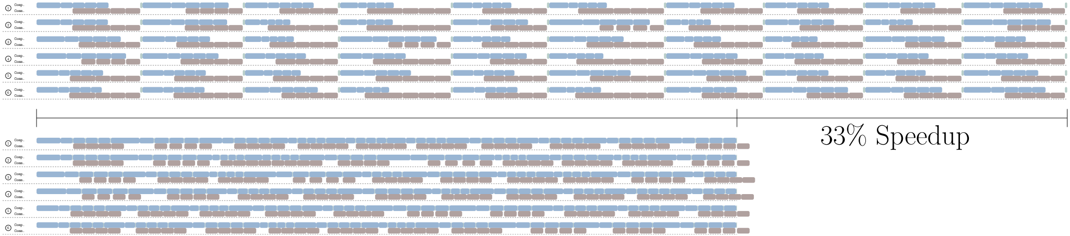
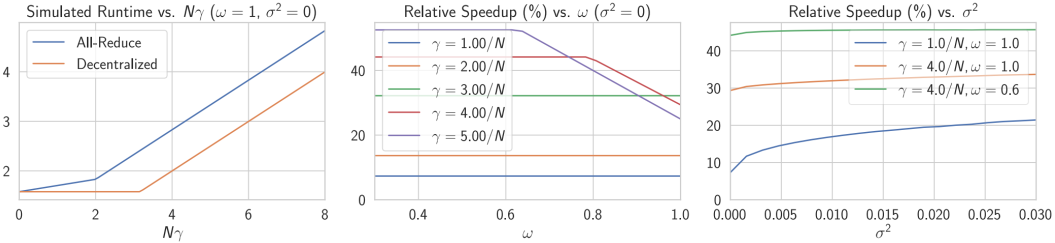


Figure: Timelines of GPU activities of 6 workers for 10 iterations. Decentralized training achieves around 33% speedup. Blue: Computation activities (forward, backward, update). Gray: Communication activities. Top: AllReduce training. Bottom: Decentralized training.

Runtime model

The proposed runtime model quantifies the key factors and estimates the speedup brought by decentralized training.

- γ: Large γ means slower communication (relative to computation time).
- ω: Smaller ω means faster decentralized communication (relative to AllReduce).
- σ^2: Large σ^2 means larger variance in computation time.



A decentralized Adam algorithm

Algorithm 1 Decentralized Adam on worker i

$m_i^{(0)}, v_i^{(0)} \leftarrow 0;\ \theta_i^{(0)} \leftarrow \theta^{(0)}$

for $t = 1, 2, \ldots, T$ **do**

$\quad g_i^{(t)} \leftarrow \nabla \ell\big(\theta_i^{(t)}; \xi_i^{(t)}\big)$

$\quad m_i^{(t)} \leftarrow \beta_1 m_i^{(t-1)} + (1-\beta_1) g_i^{(t)}$

$\quad v_i^{(t)} \leftarrow \beta_2 v_i^{(t-1)} + (1-\beta_2) \big[g_i^{(t)}\big]^2$

$\quad \theta_i^{(t+1)} \leftarrow -\alpha \frac{m_i^{(t)}/(1-\beta_1^t)}{\sqrt{v_i^{(t)}/(1-\beta_2^t)}+\epsilon} + \sum_{j \in n_i} w_{ij} \theta_j^{(t)}$

end for

A decentralized Adam algorithm

Theorem. If Algorithm 1 uses $0 < \beta_1 < \beta_2 < 1$, then

$$\left[\left\|\nabla \ell(\bar{\theta}^{(\tau)})\right\|_2^2\right] \leq \frac{4R}{\alpha \tilde{T}}\left(\ell(\bar{\theta}^{(0)}) - \ell^*\right) + E\left[\frac{1}{\tilde{T}} \ln\left(1 + \frac{R}{\epsilon(1-\beta_2)}\right) - \frac{T}{\tilde{T}} \ln(\beta_2)\right], \quad (2)$$

where $\bar{\theta}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \theta_i^{(t)}$, τ is a random stopping time, and $E \sim \alpha^2$.

Note. Last term new compared to single-machine setting, but vanishes quickly with α.

The vanishing mini-batch problem: AccumAdam

For good generalization performance, the batch size should not be too large.

- with fixed batch-size, local mini-batches vanish in size as worker count increases
- small-batches gives high-variance of momentum parameter updates

Proposed fix: accumulate gradients, update momentum parameters every s iterations.

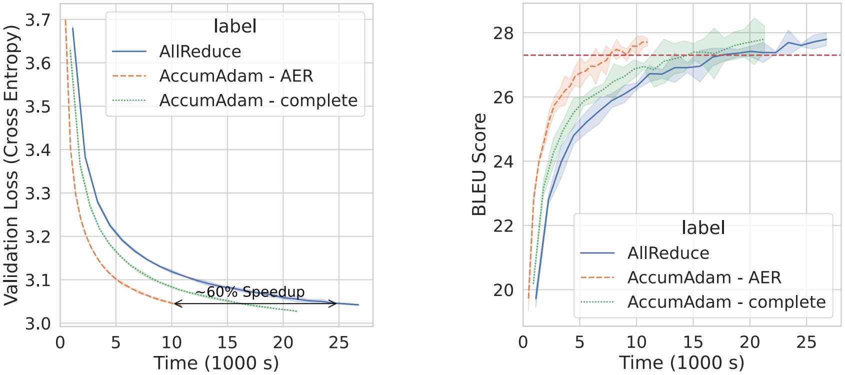
$$
\begin{aligned}
g_i^{(t)} &\leftarrow \nabla \ell\big(\theta_i^{(t)}; \xi_i^{(t)}\big) \\
m_i^{(t)} &\leftarrow \beta_1 m_i^{(t-1)} + (1-\beta_1) g_i^{(t)} \\
v_i^{(t)} &\leftarrow \beta_2 v_i^{(t-1)} + (1-\beta_2) \big[g_i^{(t)}\big]^2
\end{aligned}
$$

$\Rightarrow$

$m_i^{(t)} \leftarrow \beta_1 \bar{m}_i^{(t-1)} + (1-\beta_1) g_i^{(t)}$
$v_i^{(t)} \leftarrow \beta_2 \bar{v}_i^{(t-1)} + (1-\beta_2) \big[g_i^{(t)}\big]^2$
$b_i^{(t+1)} \leftarrow b_i^{(t)} + g_i^{(t)} / s$
if $t \bmod s == 0$ **then**
    update $\bar{m}_i, \bar{v}_i$ based on b_i
end if

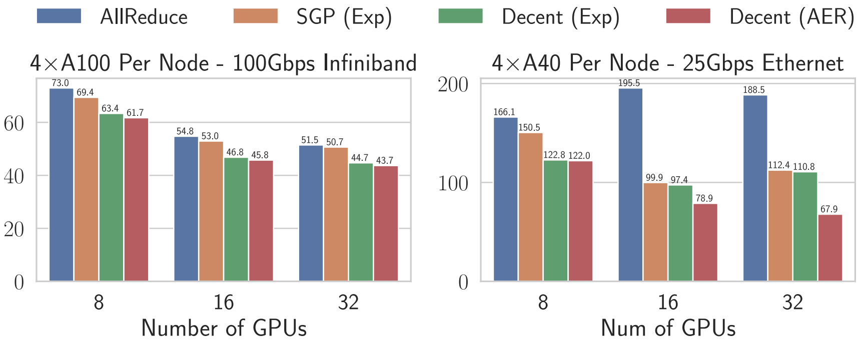
Numerical experiment results

Training of transformer for English-to-German translation task (65M parameters) with 4 nodes (4×A40 each, inter-connected by 25Gbps Ethernet) **with the same iteration budget**.



Numerical experiment results

Consistent speedups in computation-bound and communication-bound scenarios (transformer training)



Open-source repositories

Experiment

PyTorch Extension (Decent-DP)

[1] [2]

[1] https://github.com/WangZesen/Decentralized-Training-Exp

[2] https://github.com/WangZesen/Decent-DP

Summary

Contributions:

1. **High-Performance Decentralized DNN Training Framework**: Validated by extensive experiments on typical ML workloads.
2. **Simple yet Accurate Runtime Model**: Quantifies key environmental impacts and estimates potential speedups.
3. **DAdam Optimizer and Its Variant (AccumAdam)**: Enhances runtime performance and shows promising experimental results.