# Asynchronous Distributed Machine Learning

Hagit Attiya and Noa Schiller (Technion)

# Distributed Optimization

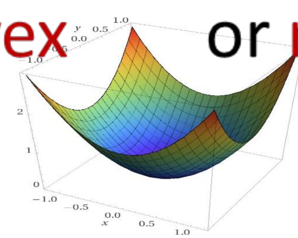
- Processes access the same data distribution $D$ and loss function $\ell: \mathbb{R}^d \times D \to \mathbb{R}$

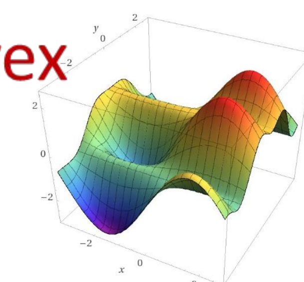
Cost function at $\mathrm{x} \in \mathbb{R}^{\mathrm{d}}$ is $Q(x) \triangleq \mathbb{E}_{z\sim D}[\ell(x,z)]$

☞Minimize $Q$ to find $x^* \in \underset{x \in \mathbb{R}^d}{\operatorname{argmin}}\, Q(x)$

$Q$ is differentiable and smooth

Can be either (strongly) convex or non-convex





# Stochastic Gradient Descent (SGD)

- Stochastic gradient $G(x, z)$ estimates the true gradient $\nabla Q(x)$ using a random data point $z \in D$

For $t = 1, 2, \ldots$

1. Draw a random data point $z \overset{i.i.d}{\sim} D$
2. Update $x_{t+1} = x_t - \eta_t G(x_t, z)$

$\eta_t$: Learning rate

Estimates are unbiased: $\mathbb{E}_{z\sim D}[G(x, z)] = \nabla Q(x)$

with bounded variance: $\mathbb{E}_{z\sim D}[\|G(x, z) - \nabla Q(x)\|_2] \leq \sigma$

# Mini-Batch SGD

Faster convergence by sampling several data points

For $t = 1, 2, \ldots$

1. Draw $M$ random data points $z_1, \ldots, z_M \overset{i.i.d}{\sim} D$
2. Update $x_{t+1} = x_t - \frac{\eta_t}{M} \sum_{i=1}^{M} G(x_t, z_i)$

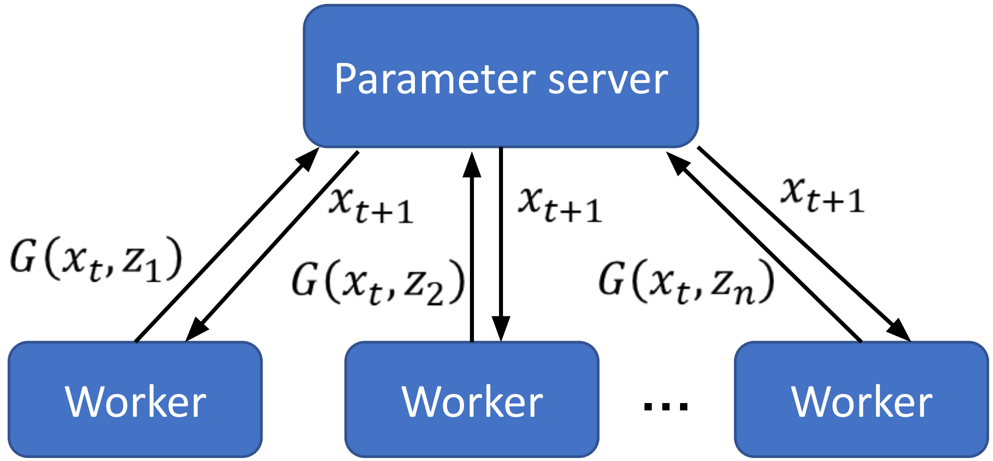
# Simple Distributed Mini-Batch SGD

Centralized scheme requires synchronization

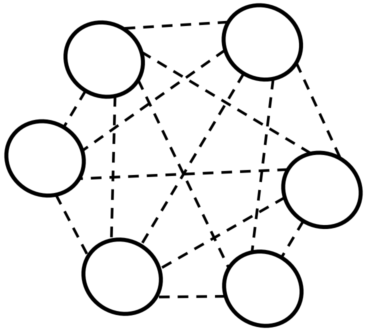
Parameter server is a single point-of-failure

For $t = 1, 2, \ldots$

1. Draw $M$ random data points $z_1, \ldots, z_M \overset{i.i.d}{\sim} D$
2. Update $x_{t+1} = x_t - \frac{\eta_t}{M} \sum_{i=1}^{M} G(x_t, z_i)$



# Decentralized SGD



- Fully-connected set of $n$ nodes

For $t = 1, 2, \ldots$

1. A node draws a random data point $\overset{i.i.d}{\sim} D$
   & computes new gradient
2. Get gradients from M nodes & update

How good is this?

# Strongly Convex Q ⇨ "External" Convergence

- Single minimum $x^*$ obtained at a unique point

For any $M$, any round $T$, and any node $i$

$$\mathbb{E}\left[\left\|x_T^i - x^*\right\|_2^2\right] \leq O\left(\frac{\left\|x^1 - x^*\right\|_2^2}{MT} + \frac{\sigma^2}{MT}\right)$$

Same convergence rate as sequential mini-batch

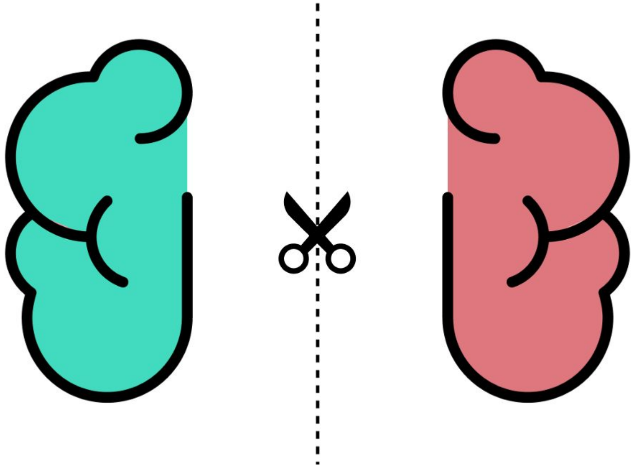
SGD, with batch size M

Implies external convergence $\mathbb{E}\left[\left\|x^i - x^*\right\|_2^2\right] \leq \epsilon$

# Strongly Convex Q
# ⇨ “External” Convergence

- Single minimum $x^*$ obtained at a unique point
- For any $M$ ⇨ the algorithm withstands partitioning
- I.e., converges even when communicating with only a minority

No “split-brain”



# Non Strongly Convex Functions Require a Majority

- For some non-convex cost function Q, algorithm does not converge if a majority of processes fail

$$\max_{i,j} \mathbb{E}\left[\left\|\text{output}^i - \text{output}^j\right\|_2\right] > \delta$$

(No internal convergence)

Proof more complicated than expected and relies on probabilistic indistinguishability

[Goren, Moses & Spiegelman, DISC 2021]

# General Algorithm

Convergence rate similar to sequential algorithm

[Ghadimi,Lan, 2013]

Analysis is a simplified version of

[ElMhamdi,Farhadkhani,Guerraoui,Guirguis,Hoang,Rouault,NeuroIPS 2021]

**For iteration** $t = 1, \dots, T$:

1. **Compute the stochastic gradient** $g_t^i$ **at** $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. **Send** $\langle t, x_t^i \rangle$
4. **Wait to receive** $\geq M$ **iteration-**$t$ **messages**
5. $x_{t+1}^i \leftarrow \mathrm{Avg}($**recieved models**$)$

# General Algorithm

- But with an additive factor of $\Delta$, which is reduced using multi-dimensional approximate agreement

This algorithm needs communication with a majority

For iteration $t = 1, \ldots, T$:

1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq M$ iteration-$t$ messages
5. $x_{t+1}^i \leftarrow \text{Avg}(\text{recieved models})$

# Multi-Dimensional Approximate Agreement

A process starts with input $x^i \in \mathbb{R}^\mathrm{d}$ and returns $y^i \in \mathbb{R}^\mathrm{d}$, such that the outputs are

- In the convex hull of the inputs
- Contracted by a factor of $q$ relative to the inputs

$$\max_{i,j} \left\| y^i - y^j \right\|_2^2 \leq q \max_{i,j} \left\| x^i - x^j \right\|_2^2$$

[Mendes, Herlihy, Vaidya, Garg, DC 2015]
[Fugger, Nowak, DISC 2018]

# Convergence of General Algorithm

- With an appropriate q, we get internal convergence

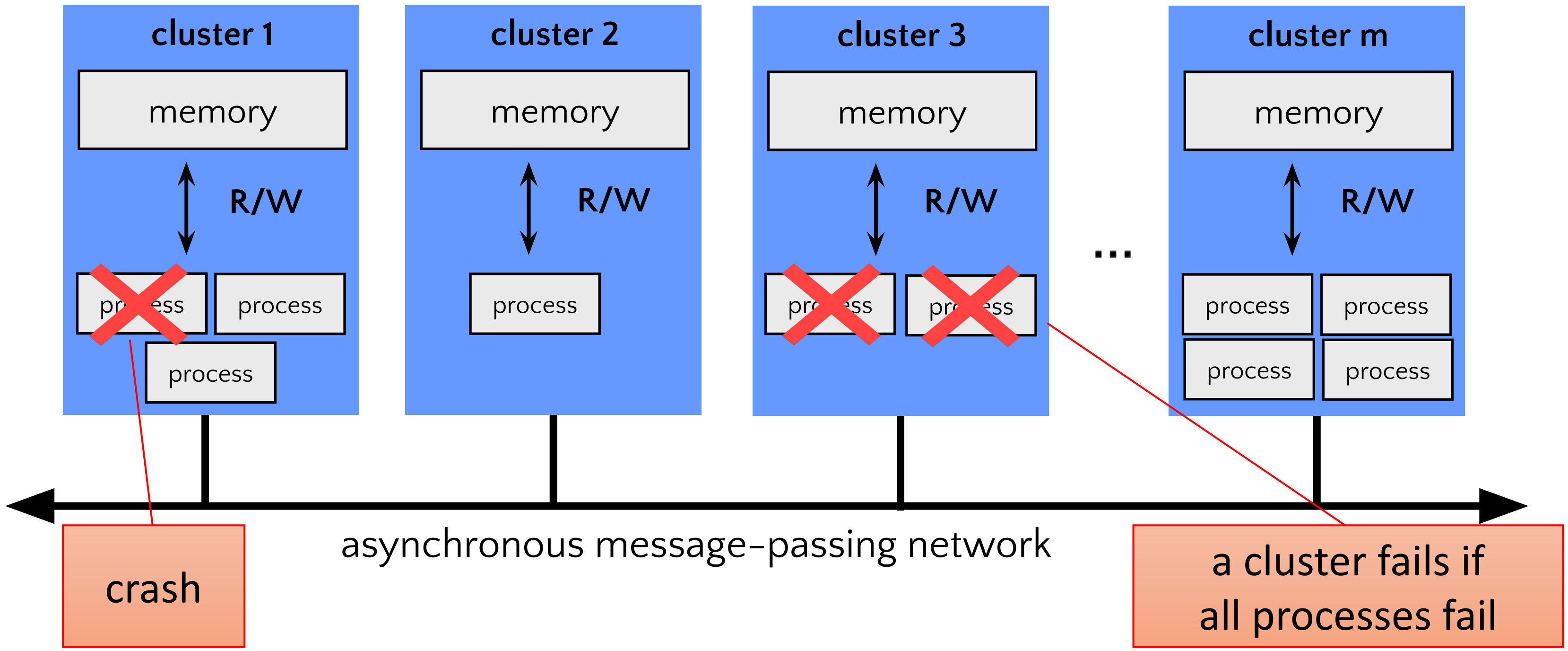
$$\max_{i,j} \mathbb{E}\left\|x^i - x^j\right\|_2 < \delta$$

Can also show external convergence

☝ Better (1-dimension) AA when shared-memory is used

[A,Kumari,Schiller,OPODIS 2020]

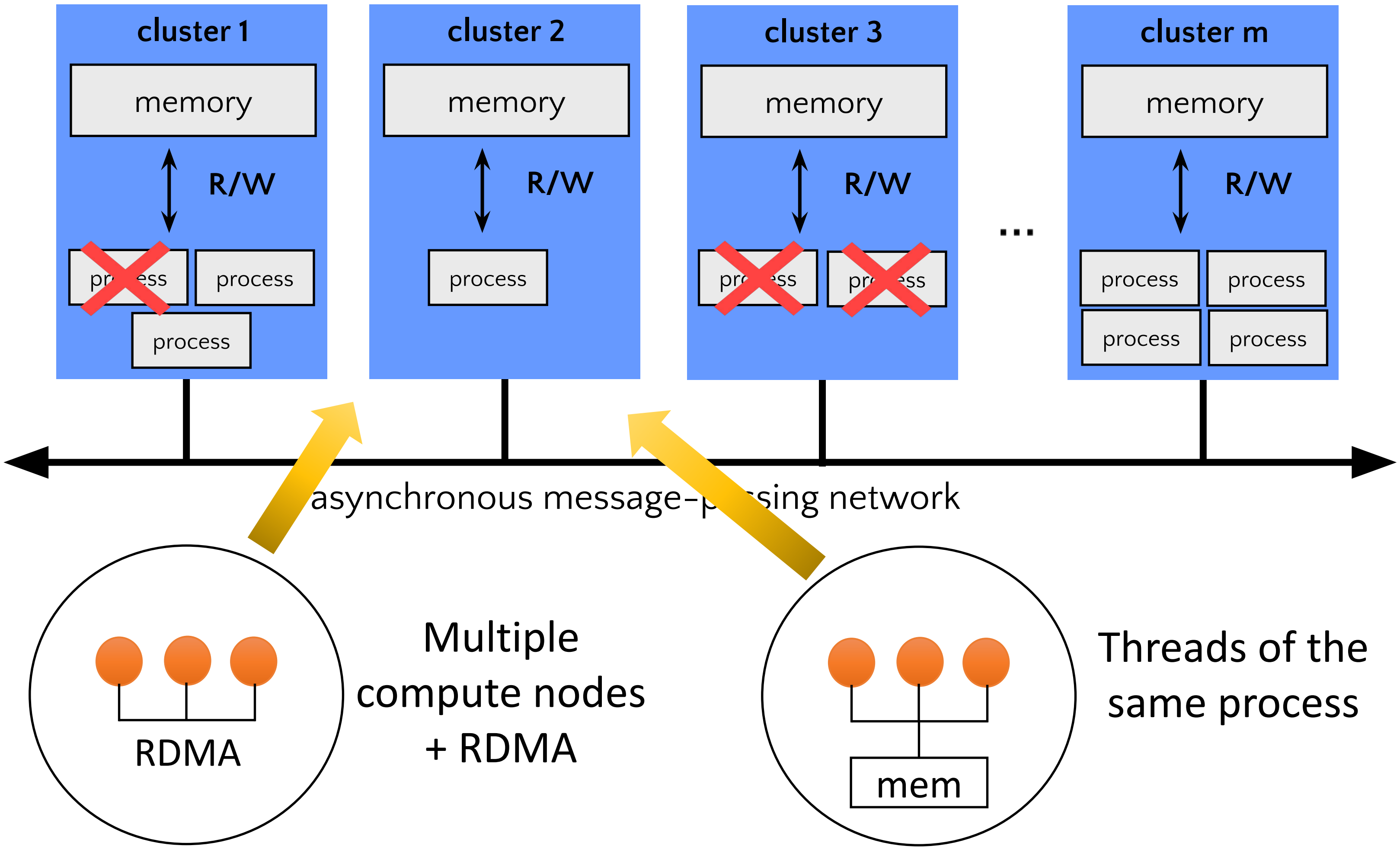
# Cluster-Based Model




- Disjoint clusters, each with shared read/write registers
- All processes can send asynchronous messages to each other

[Raynal, Cao,ICDCS 2019]

# Cluster-Based Model for HPC



# Multi-Dimensional AA in the Cluster-Based Model

```
For round r = 1 ... R:
1. z_r = ClusterApproximateAgreement(y_r)
2. Send ⟨r, z_r⟩ to all processes
3. Wait to receive round r messages representing
              a majority of processes
4. y_{r+1} = Aggregate(received values)
```

# Multi-Dimensional AA in the Cluster-Based Model

A process represents all processes in its cluster

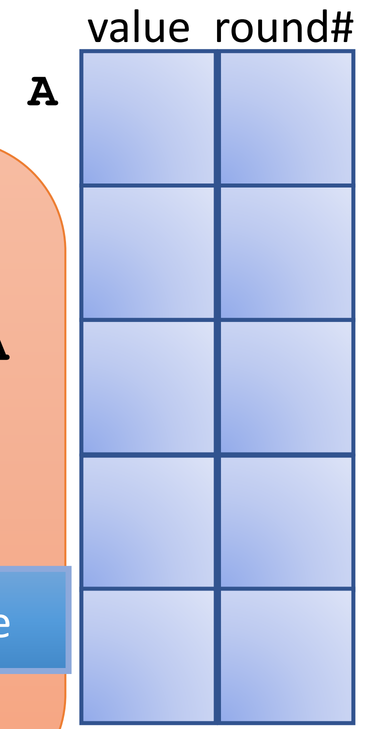
```
For round r = 1 ... R:
1. z_r = ClusterApproximateAgreement(y_r)
2. Send ⟨r, z_r⟩ to all processes
3. Wait to receive round r messages representing
               a majority of processes
4. y_{r+1} = Aggregate(received values)
```

Better contraction inside a cluster

Tune to get $q$-contraction in O(log $q$) rounds

# MDAA within a Cluster

An array **A** of ⟨value, round#⟩ for each cluster



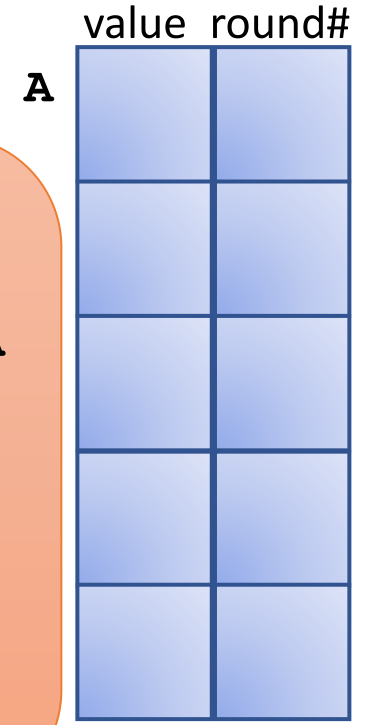

```
r ← 1 ; A[i] ← ⟨x,r⟩
while r < constant do
   let r_max be the largest round number in A
   if r = r_max then
      X ← values in A with round r_max
      A[i] ← ⟨MidExtremes(X),r+1⟩
      r ← r + 1
   else r ← r_max
return some x_j s.t. A[j] = ⟨x_j,r+1⟩
```

Aggregation rule

**MidExtremes** returns the average of the two values realizing the maximum Euclidean distance

# MDAA within a Cluster

An array **A** of ⟨value, round#⟩ for each cluster




```
r ← 1 ; A[i] ← ⟨x,r⟩
while r < constant do
    let r_max be the largest round number in A
    if r = r_max then
        X ← values in A with round r_max
        A[i] ← ⟨MidExtremes(X),r+1⟩
        r ← r + 1
    else r ← r_max
return some x_j s.t. A[j] = ⟨x_j,r+1⟩
```

Ensures constant contraction within O(1) rounds

# Skipping

```
r ← 1 ; A[i] ← ⟨x,r⟩
while r < constant do
   let r_max be the largest round number in A
   if r = r_max then
      X ← values in A with round r_max
      A[i] ← ⟨MidExtremes(X),r+1⟩
      r ← r + 1
   else r ← r_max
return some x_j s.t. A[j] = ⟨x_j,r+1⟩
```

skipping

# Skipping

A process can skip to the most advanced iteration instead of going through intermediate iterations

```
For round r = 1 … R:
1. z_r = ClusterApproximateAgreement(y_r)
2. Send ⟨r, z_r⟩ to all processes
3. Wait to receive round r message from
              a majority of clusters
4. y_{r+1} = AggregationRule(received values)
5. If received round r′ messages, r′ > r ,
              then skip to round r′
```

# Recovery through Skipping

Allows recovering process to rejoin the computation

Non-volatile memory can be used to checkpoint the current status

```
For round r = 1 ... R:
1. z_r = ClusterApproximateAgreement(y_r)
2. Send ⟨r, z_r⟩ to all processes
3. Wait to receive round r message from
            a majority of clusters
4. y_{r+1} = AggregationRule(received values)
5. If received round r' messages, r' > r ,
            then skip to round r'
```

# Some Related Work

Other work does not ignore (stale) parameters from previous iterations

[Li,Ben-Nun,Di Girolamo,Alistarh,Torsten Hoefler,PPoPP 2020]

[Li,Ben-Nun,Di Girolamo, Dryden,Alistarh,Torsten Hoefler,TPDS 2021]

Elastic consistency bounds the staleness

[Nadiradze,Markov,Chatterjee,Kungurtsev,Alistarh, AAAI 2021]

Our MDAA algorithm “beats” a *f(d + 2)*-redundancy lower bound for Byzantine failures

[Mendes,Herlihy,Vaidya,Garg, DC 2015]

*2f*-redundancy is a necessary and sufficient condition for f-resilient Byzantine optimization

[Su,Vaidya,PODC 2016]

# Thank!
# Questions?

# General Algorithm

Proceed to the next iteration, only after receiving current-iteration messages from a majority

For iteration $t = 1, \ldots, T$:

1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq \lceil n/2 \rceil$ iteration-$t$ messages
5. $x_{t+1}^i \leftarrow \text{Avg}(\text{recieved models})$

# Distributed SGD Algorithm

For iteration $t = 1 \dots T$:

1. Compute stochastic gradient $g_t$ w.r.t $x_t$
2. Update $y_t = x_t - \eta_t g_t$
3. Send $\langle t, y_t \rangle$ to all processes
4. Collect enough round $t$ models and average them $\bar{y}_t$
5. $x_{t+1} = \text{ApproximateAgreement}(\bar{y}_t)$
6. **If** received enough messages for round $t' > t$ **then** skip to round $t'$

# Correctness Proof – Definitions

- Given a family of vectors, $\vec{x} = (x^1, \dots, x^k)$ such that $x^i \in \mathbb{R}^d$, define the ***coordinate-wise diameter***:

$$\Delta^{cw}(\vec{x}) \triangleq \sum_{i=1}^{d} \max_{j,l \in [k]} \left| x^j[i] - x^l[i] \right|$$

and the **average** of the vectors in the family:

$$\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x^i$$

# Main Correctness Claims

For every iteration $t \geq 1$, for a constant learning rate,

$$\eta_j = \eta \leq \frac{1}{2Ld(\lfloor m/2 \rfloor + 1)}$$

Smoothness constant

Parameter dimension

Constant

$$\mathbb{E}[\Delta^{cw}(\vec{x}_t)] \leq \boxed{\eta C}$$

Where:

- $\Delta^{cw}(\vec{x}_t) \triangleq \sum_{i=1}^{d} \max_{c_1, c_2 \in [m]} \left| x_t^{c_1}[i] - x_t^{c_2}[i] \right| \geq \Delta_2(\vec{x})$
- $C \triangleq 4d \lfloor m/2 \rfloor \sigma + 2\epsilon_{max}(\lfloor m/2 \rfloor + 1)$

# Main Correctness Claims

For every iteration $t \geq 1$, for decreasing learning rate, $\eta_j \leq \frac{1}{2Ld(\lfloor m/2 \rfloor + 1)}$

$$\mathbb{E}[\Delta^{cw}(\vec{x}_t)] \leq \eta_{\lfloor t/2 \rfloor} C + \eta_1 C q^{\lfloor t/2 \rfloor}$$

Where

- $q <$
- $C \triangleq$

If $\lim_{t \to \infty} \eta_t = 0$

then $\lim_{t \to \infty} \mathbb{E}[\Delta^{cw}(\vec{x}_t)] = 0$

# Correctness Proof

- Let $C \triangleq 4d\lfloor m/2 \rfloor \sigma + 2\epsilon_{max}(\lfloor m/2 \rfloor + 1)$ be a constant

For every iteration $t \geq 1$, assuming that the learning rate sequence is decreasing, such that $\eta_1 \leq \frac{1}{2Ld(\lfloor m/2 \rfloor + 1)}$, then

$$\mathbb{E}[\Delta^{cw}(\vec{x}_t)] \leq \eta_{\lfloor t/2 \rfloor} C + \eta_1 C \left( \frac{2\lfloor m/2 \rfloor + 1}{2\lfloor m/2 \rfloor + 2} \right)^{\lfloor t/2 \rfloor}$$

**Conclusion:** $\lim_{t \to \infty} \mathbb{E}[\Delta^{cw}(\vec{x}_t)] = 0$

# Main Correctness Claims

- For iteration $t \geq 1$ and cluster $c$, define the ***effective gradient***, $G_t^c = \left(x_t^c - x_t^{c+1}\right)/\eta_t$

For every iteration $t \geq 1$:

$$\mathbb{E}\left[\|G_t^c - \nabla Q(x_t^c)\|_2^2\right] \leq$$

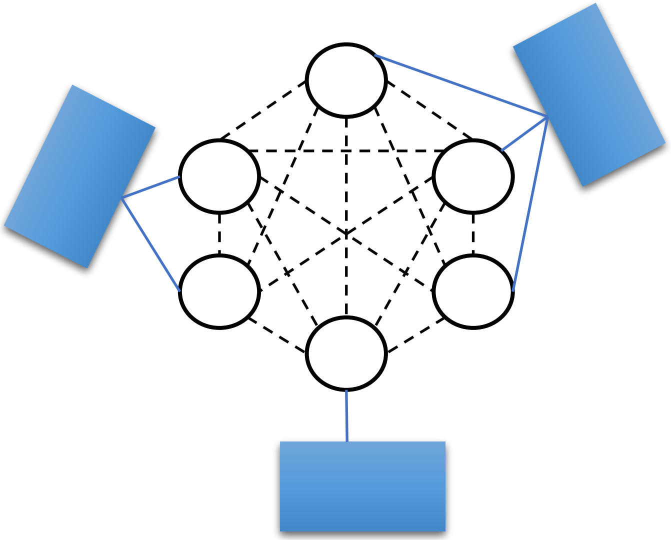
$$(24nd + 4)\sigma^2 + \left(\frac{4}{\eta_t^2} + 4L^2 + 12L^2 d\right)\mathbb{E}[(\Delta^{cw}(\vec{x}_t))^2]$$

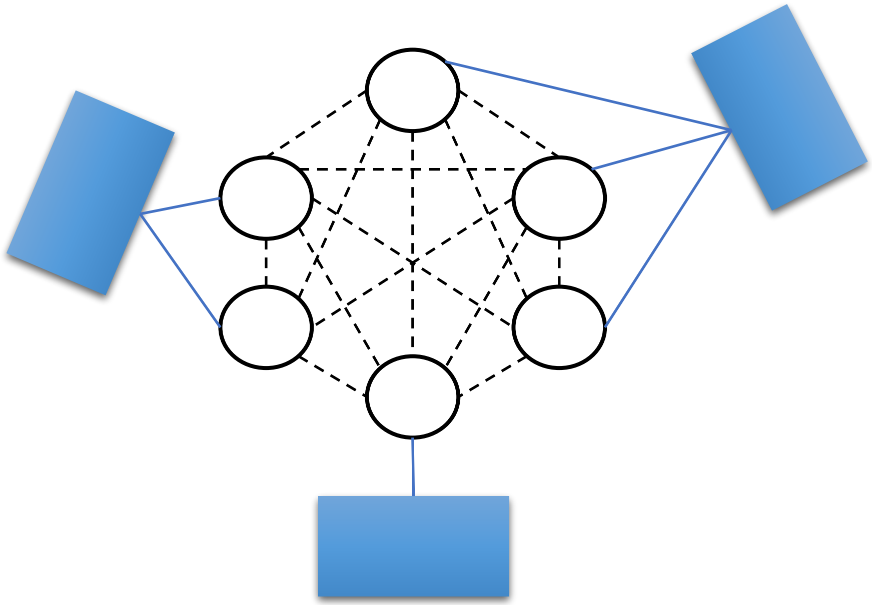
Using this lemma, we can prove convergence for **smooth convex** and **non-convex** cost functions

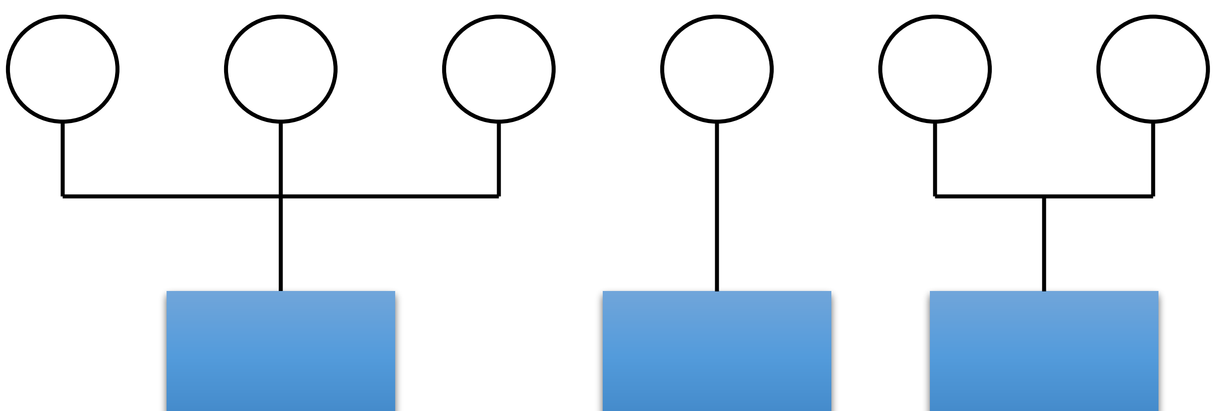
# To Conclude

- Less intra and inter-cluster synchronization
- Can this framework can be used for other optimization problems?
- Can the algorithm be made practical?
- Can we show theoretical or empirical speedup compared to the sequential and other distributed algorithms?

# Model







# Back to Overall MDAA

- Can pick ε to ensure $q$-contraction in O(log $q$) rounds

For round $r = 1 \dots R$:
1. $z_r =$ `ClusterApproximateAgreement(`$y_r$`)`
2. Send $\langle r, z_r \rangle$ to all processes
3. Wait to receive round $r$ message from a majority of clusters
4. $y_{r+1} =$ `Aggregate(received values)`

# General Algorithm

Proceed to the next iteration after receiving current-iteration messages from a majority

For iteration $t = 1, \ldots, T$:

1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$ → **Perturbation phase**
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq \lceil n/2 \rceil$ iteration-$t$ messages
5. $y_t^i \leftarrow \text{Avg}(\texttt{recieved models})$
6. $x_{t+1}^i \leftarrow \text{MultiDimApproxAgree}(y_t^j)$

(Steps 5–6) → **Contraction phase**

# General Algorithm

Prove $\mathbb{E}\left[\left\|x_{t+1}^i - x^*\right\|_2^2\right] \leq ??\ \mathbb{E}\left[\left\|x_t^i - x^*\right\|_2^2\right] + T\Delta$

Analysis is a simplified version of

[ElMhamdi,Farhadkhani,Guerraoui,Guirguis,Hoang,Rouault,NeuroIPS 2021]

**For iteration** $t = 1, \dots, T$:

1. **Compute the stochastic gradient** $g_t^i$ **at** $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. **Send** $\langle t, x_t^i \rangle$
4. **Wait to receive** $\geq \lceil n/2 \rceil$ **iteration-**$t$ **messages**
5. $x_{t+1}^i \leftarrow \mathrm{Avg}(\text{recieved models})$

# General Algorithm

Prove $\mathbb{E}\left[\left\|x_{t+1}^i - x^*\right\|_2^2\right] \leq ??\ \mathbb{E}\left[\left\|x_t^i - x^*\right\|_2^2\right] + \Delta$

Matches the sequential algorithm

[Ghadimi,Lan, 2013]

**For iteration** $t = 1, \dots, T$:

1. **Compute the stochastic gradient** $g_t^i$ **at** $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. **Send** $\langle t, x_t^i \rangle$
4. **Wait to receive** $\geq \lceil n/2 \rceil$ **iteration-**$t$ **messages**
5. $x_{t+1}^i \leftarrow \mathrm{Avg}(\text{recieved models})$

# General Algorithm

- Prove $\mathbb{E}\left[\left\|x_{t+1}^i - x^*\right\|_2^2\right] \leq ??\ \mathbb{E}\left[\left\|x_t^i - x^*\right\|_2^2\right] + \Delta$

Reduce $\Delta$ to bring the values together with multi-dimensional approximate agreement

For iteration $t = 1, \ldots, T$:
1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq \lceil n/2 \rceil$ iteration-$t$ messages
5. $x_{t+1}^i \leftarrow \text{Avg}(\text{recieved models})$

# General Algorithm

Prove $\mathbb{E}\left[\left\|x_{t+1}^{i} - x^{*}\right\|_2^2\right] \leq ??\ \mathbb{E}\left[\left\|x_{t}^{i} - x^{*}\right\|_2^2\right] + \Delta$

Reduce $\Delta$ to bring the values together with multi-dimensional approximate agreement

For iteration $t = 1, \ldots, T$:

1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq \lceil n/2 \rceil$ iteration-$t$ messages
5. $y_t^i \leftarrow \mathrm{Avg}(\text{recieved models})$
6. $x_{t+1}^i \leftarrow \mathrm{MultiDimApproxAgree}(y_t^j)$