Strategies and Principles of Distributed Machine Learning on Big Data

Eric P. Xing et al, “Strategies and Principles of Distributed Machine Learning on Big Data,” Engineering (The Journal of Chinese Academy of Engineering), 2016

Acknowledgement

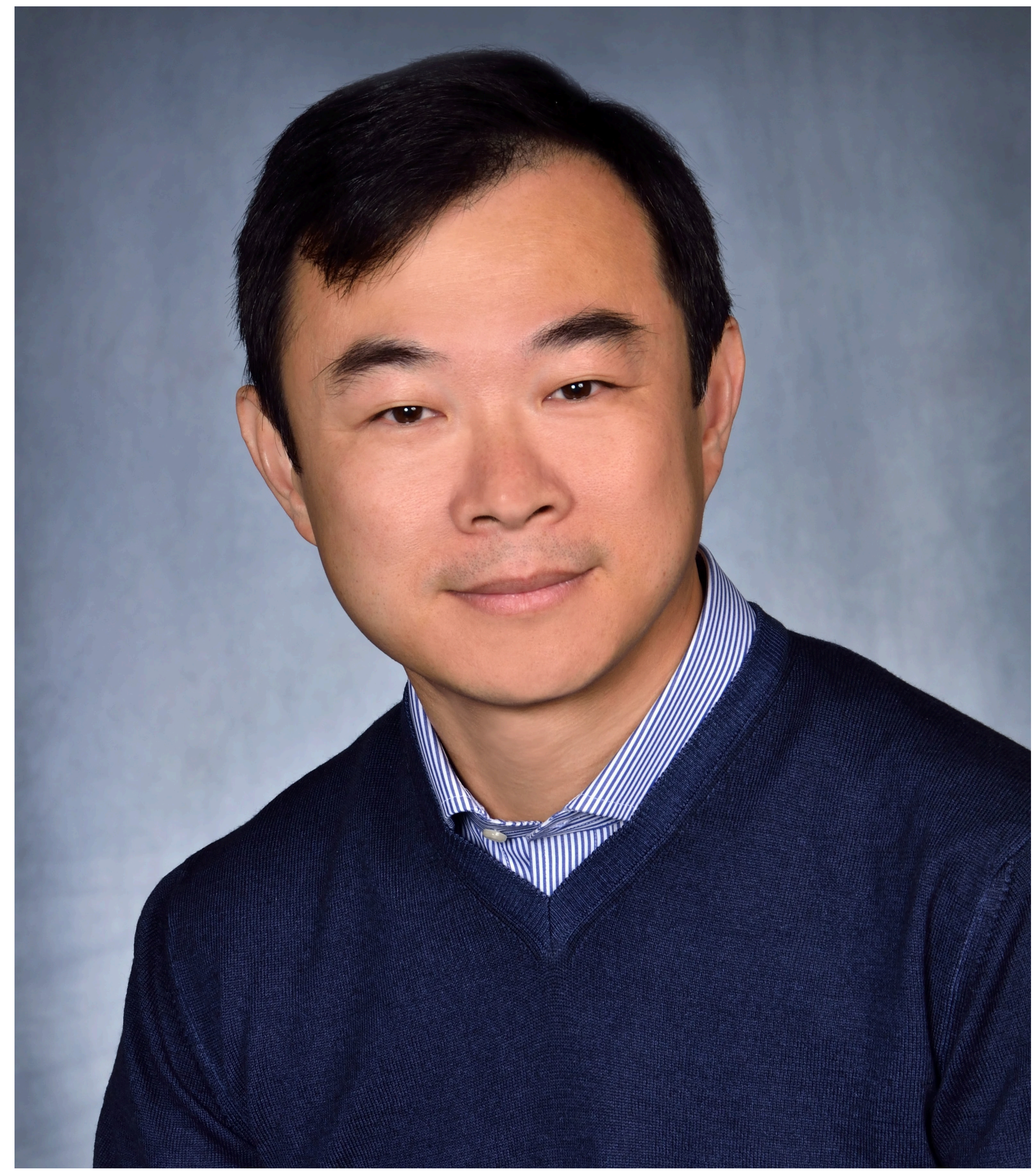
The slides are adapted from the following source materials:

- "Big ML Software for Modern ML Algorithms" with Dr. Qirong Ho at the 2014 IEEE International Conference on Big Data
- "A New Look at the System, Algorithm and Theory Foundations of Distributed Machine Learning" Eric P. Xing and Dr. Qirong Ho at the 21st ACM SIGKDD Conference on knowledge Discovery and Data Mining
- "Distributed machine learning", Stanley Wang
- "Strategies & Principles for Distributed Machine Learning", Eric P. Xing at Allen Institute for Artificial Intelligence

All copyrights belong to the original authors of the materials

About Eric P. Xing

- Professor in the School of Computer Science at CMU
- Ph.D in Molecular Biology from Rutgers University
- Ph.D in Computer Science at U.C. Berkeley
- Research focus on **machine learning and statistical methodology** and **large-scale computational system and architecture**



Challenges for Modern ML

- Massive Data Scale
- Gigantic Model Size
- Inadequate ML library
- ML algorithms iterative convergent

Iterative-Convergent ML Algorithm

$$\arg\max_{\vec{\theta}} \equiv \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

Model **Data** **Parameter**

Solved by an iterative convergent algorithm

```
for (t = 1 to T) {
  doThings()
```

$$\vec{\theta}^{t+1} = g(\vec{\theta}^{t}, \; \Delta_f \vec{\theta}(\mathcal{D}))$$

```
  doOtherThings()
}
```

This computation needs to be parallelized!

A Tale of Two Communities

- ML Communities
  - want correctness, fewer iterations to converge
  - … but assume an ideal system

```
for (t = 1 to T) {
  doThings()
  parallelUpdate(x,θ)
  doOtherThings()
}
```

- Oversimplify systems issues
  - e.g. machines perform consistently
  - e.g. can sync parameters any time

A Tale of Two Communities

- System Communities
  - Want more iterations executed per second
  - … but assume ML also is a black box

- Oversimplify ML issues
  - e.g. assume ML algo “works” without proof
  - e.g. ML algo “easy to rewrite” in chosen abstraction: MapR, vertex program, etc.

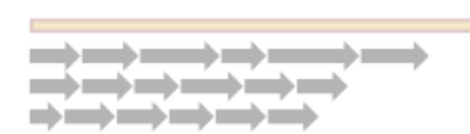
A Tale of Two Communities

- ML Communities
  - want correctness, fe
    iterations to conve
  - … but assume a
    system

```
for (t = 1 to T) {
  doThings()
  parallelUpdate(x,
  doOtherThings()
}
```

- Oversimplify systems issues
  - e.g. machines perform consistentl
  - e.g. can sync parameters any time

- System Communities
  - nt more iterations
    ted per second
  - ssume ML also is a
    x



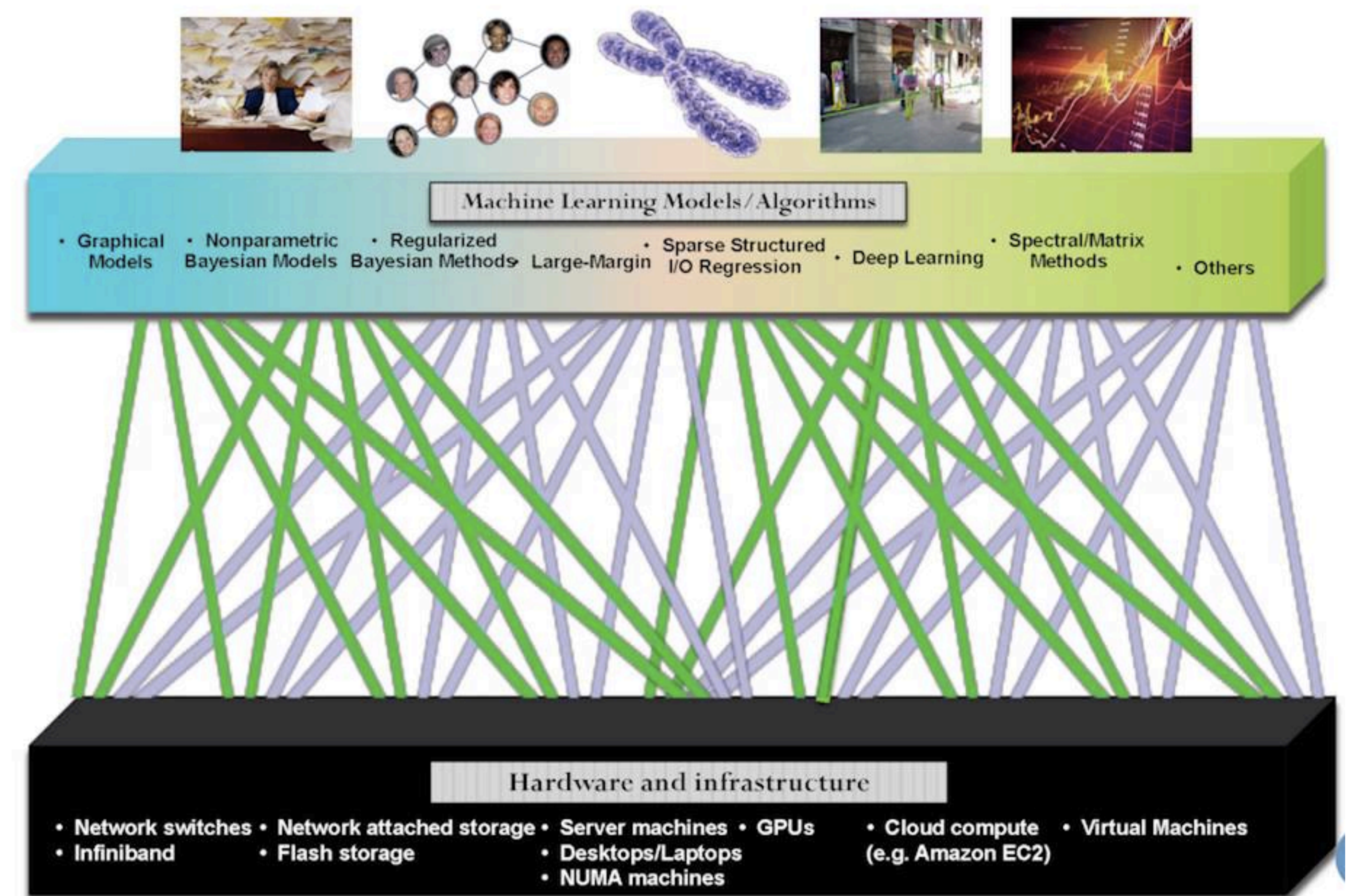
ct
allel

Fast-but-unstable
Asynchronous Parallel

- plify ML issues
  - e.g. assume ML algo "works" without proof
  - e.g. ML algo "easy to rewrite" in chosen abstraction: MapR, vertex program, etc.

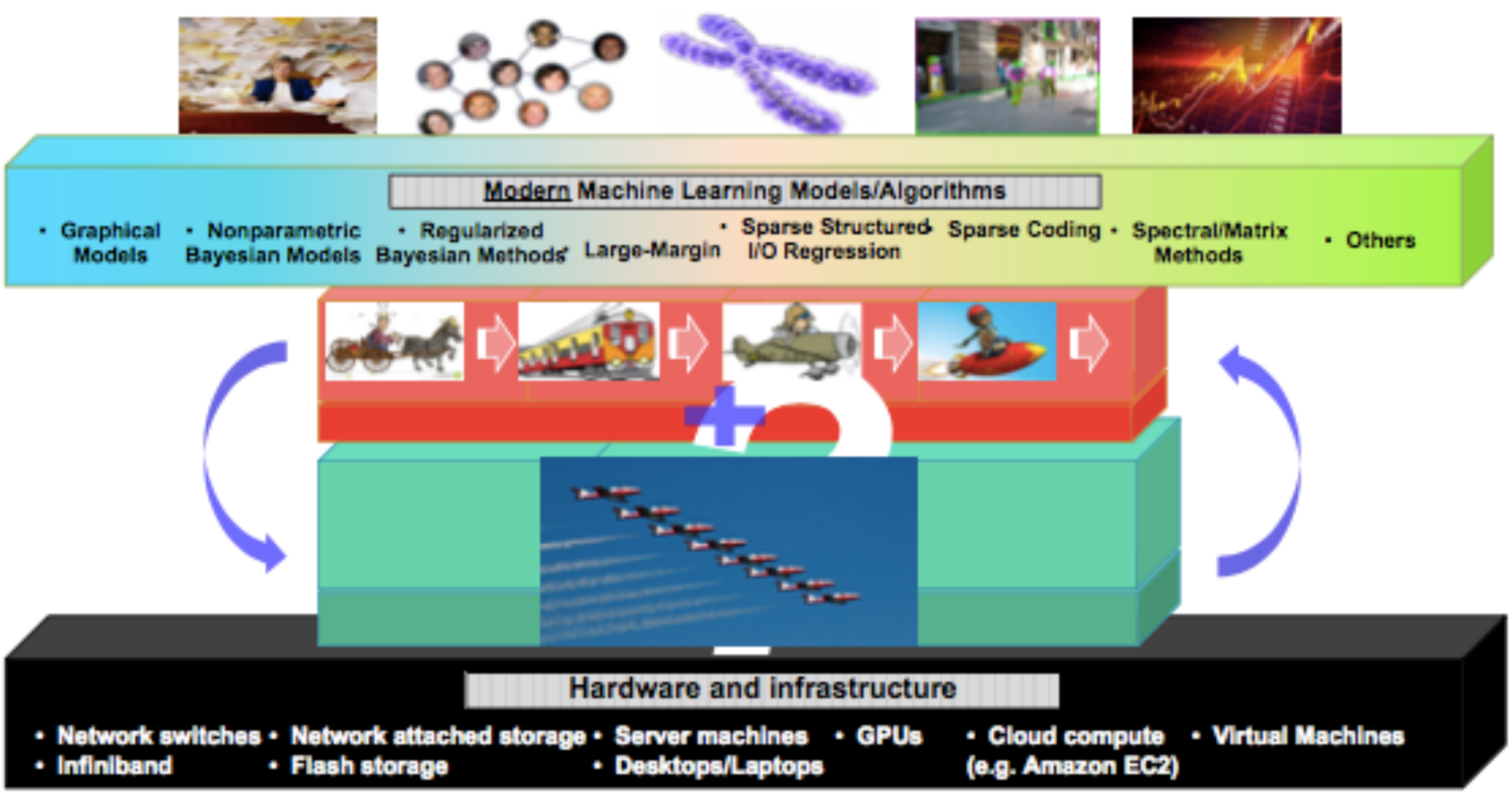
A Tale of Two Communities

Traditional Approach:



A Tale of Two Communities

What they want:



Iterative-Convergent ML Algorithm

$$\arg\max_{\vec{\theta}} \equiv \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

Model **Data** **Parameter**

Solved by an iterative convergent algorithm

```
for (t = 1 to T) {
  doThings()
```

$$\vec{\theta}^{t+1} = g(\vec{\theta}^{t}, \underbrace{\Delta_f \vec{\theta}(\mathcal{D})}_{})$$

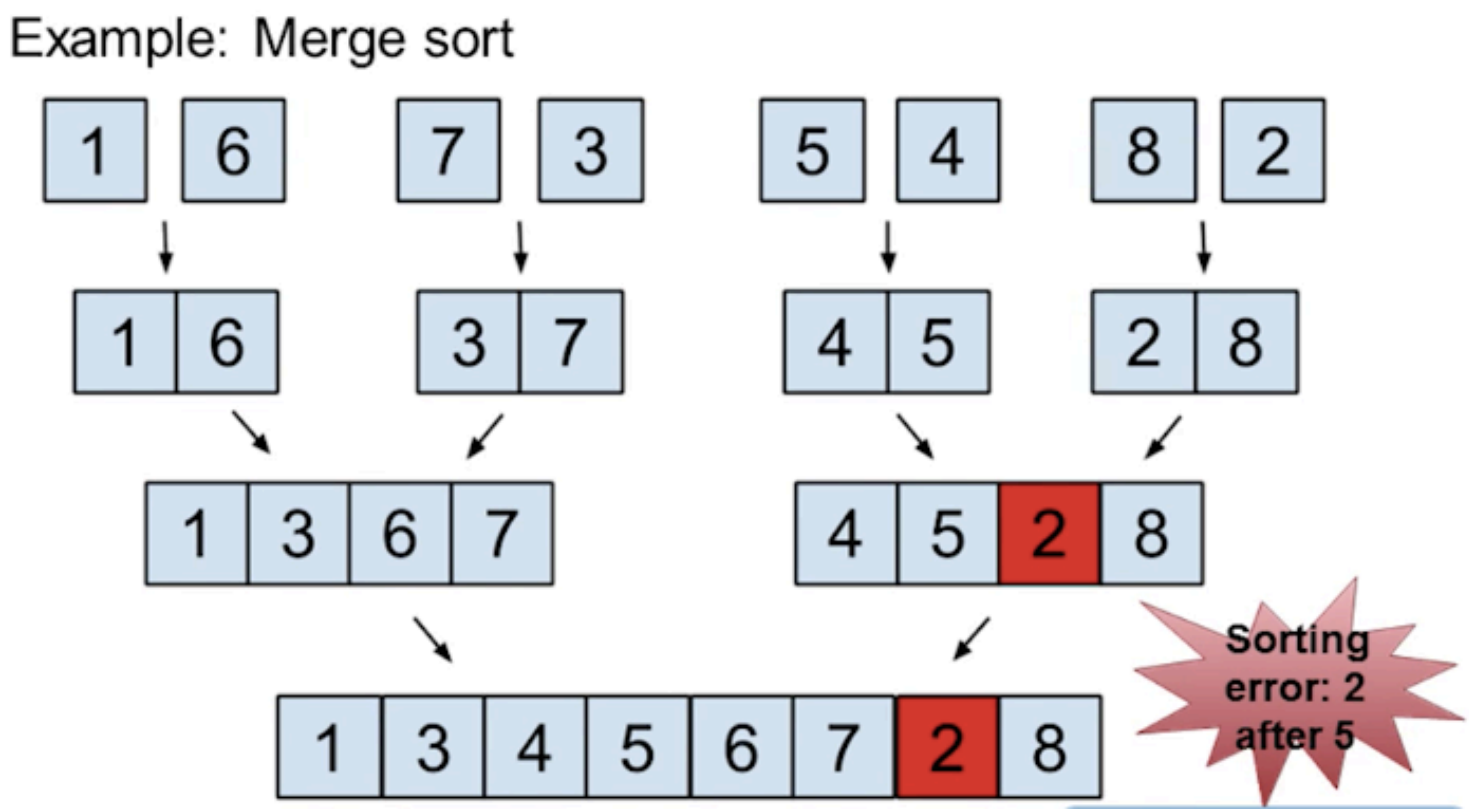
```
  doOtherThings()
}
```

This computation needs to be parallelized!

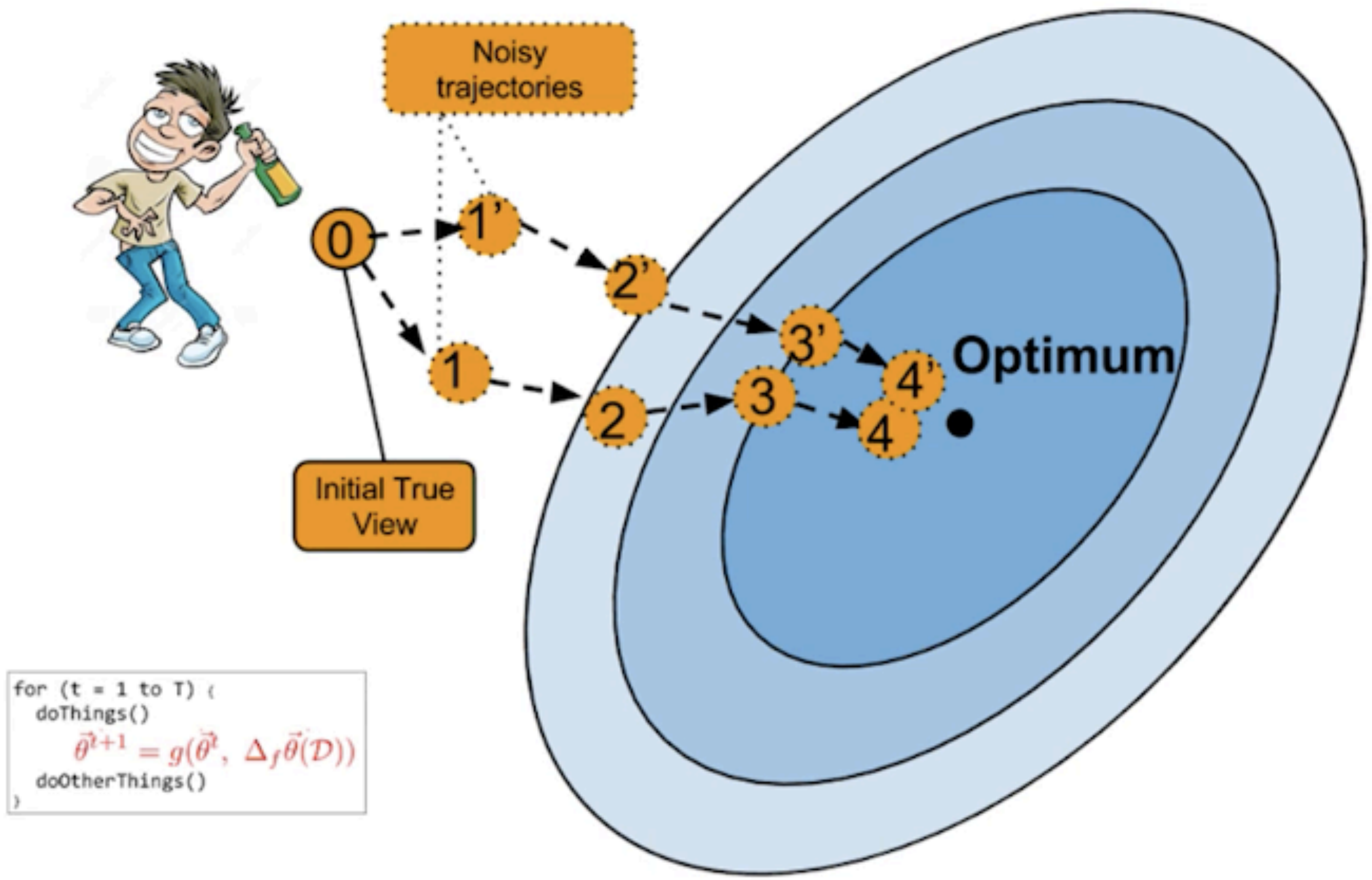
Properties of ML Program

- Error tolerance
- Dependency structure
- Non-uniform convergence
- Compact update

Properties of ML Program



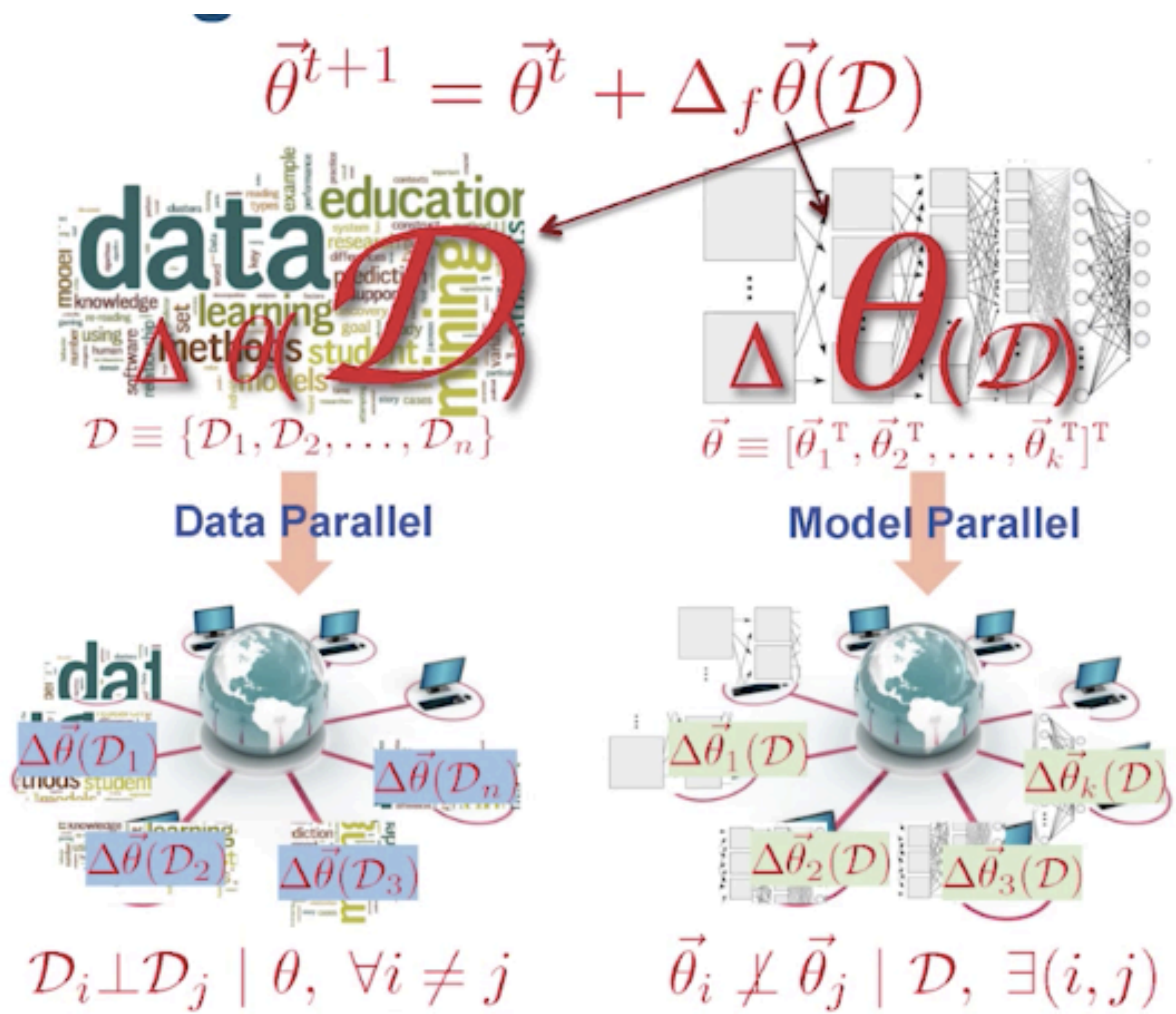
Properties of ML Program



Properties of ML Program

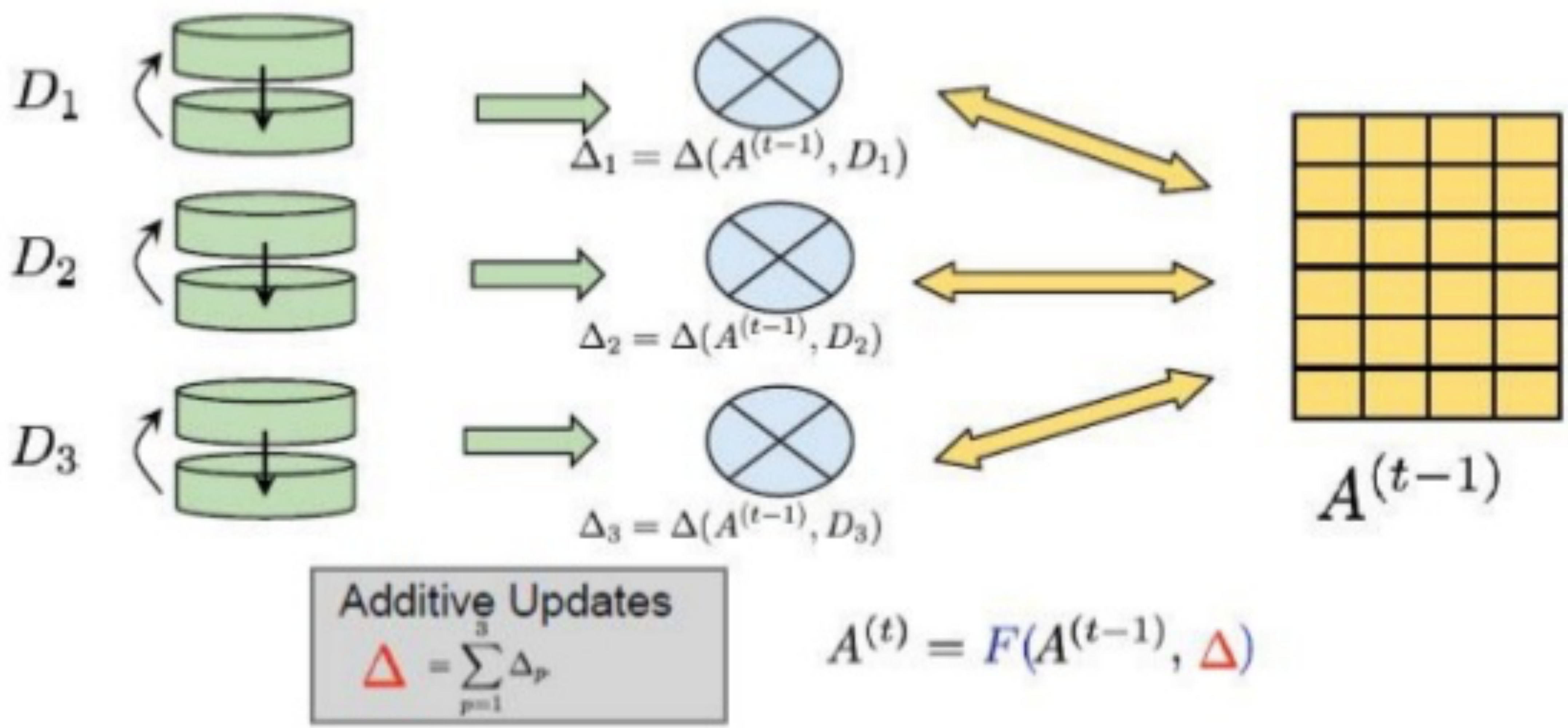
- Error tolerance
- Dependency structure
- Non-uniform convergence
- Compact update

Two Strategies of ML System

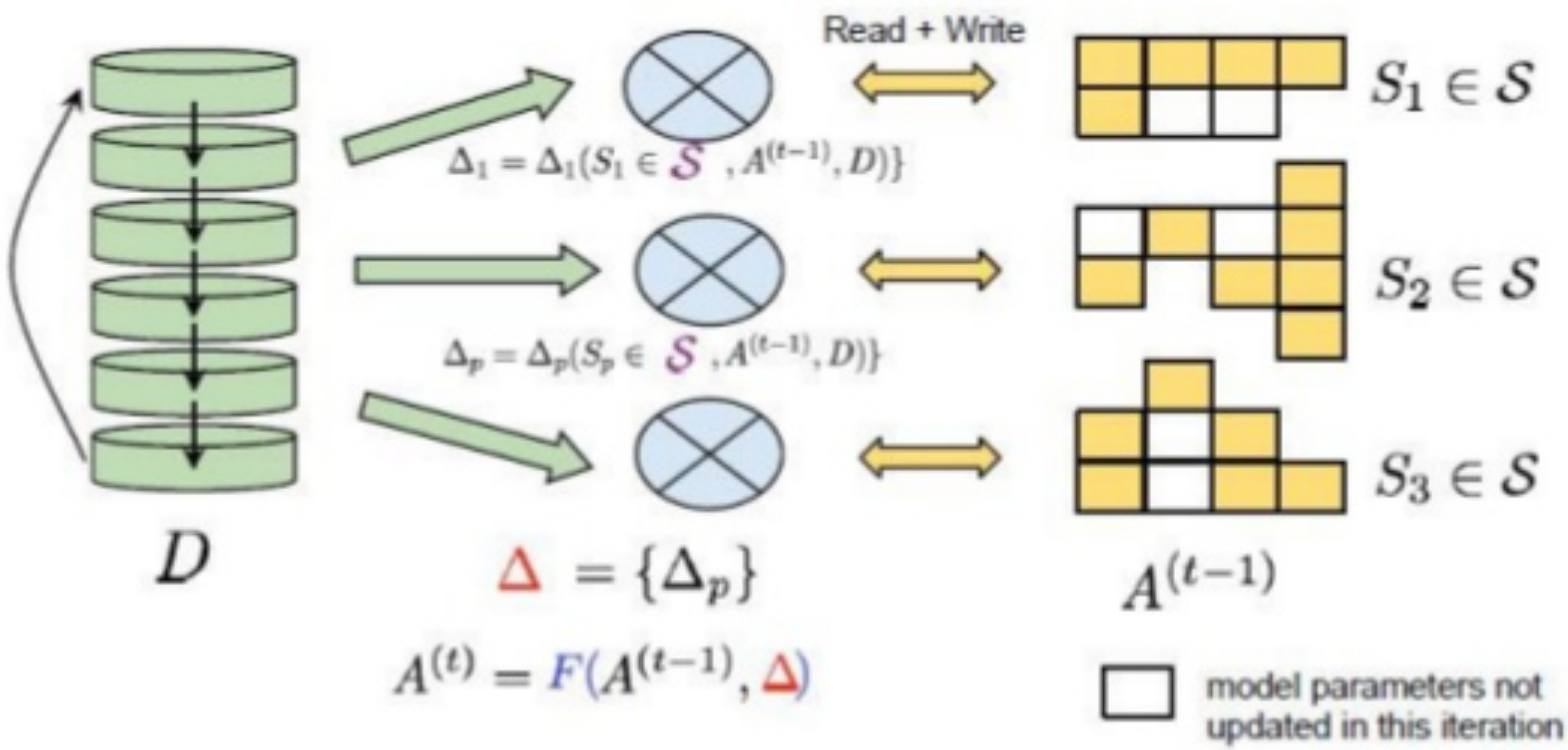


Data parallel & Model parallel

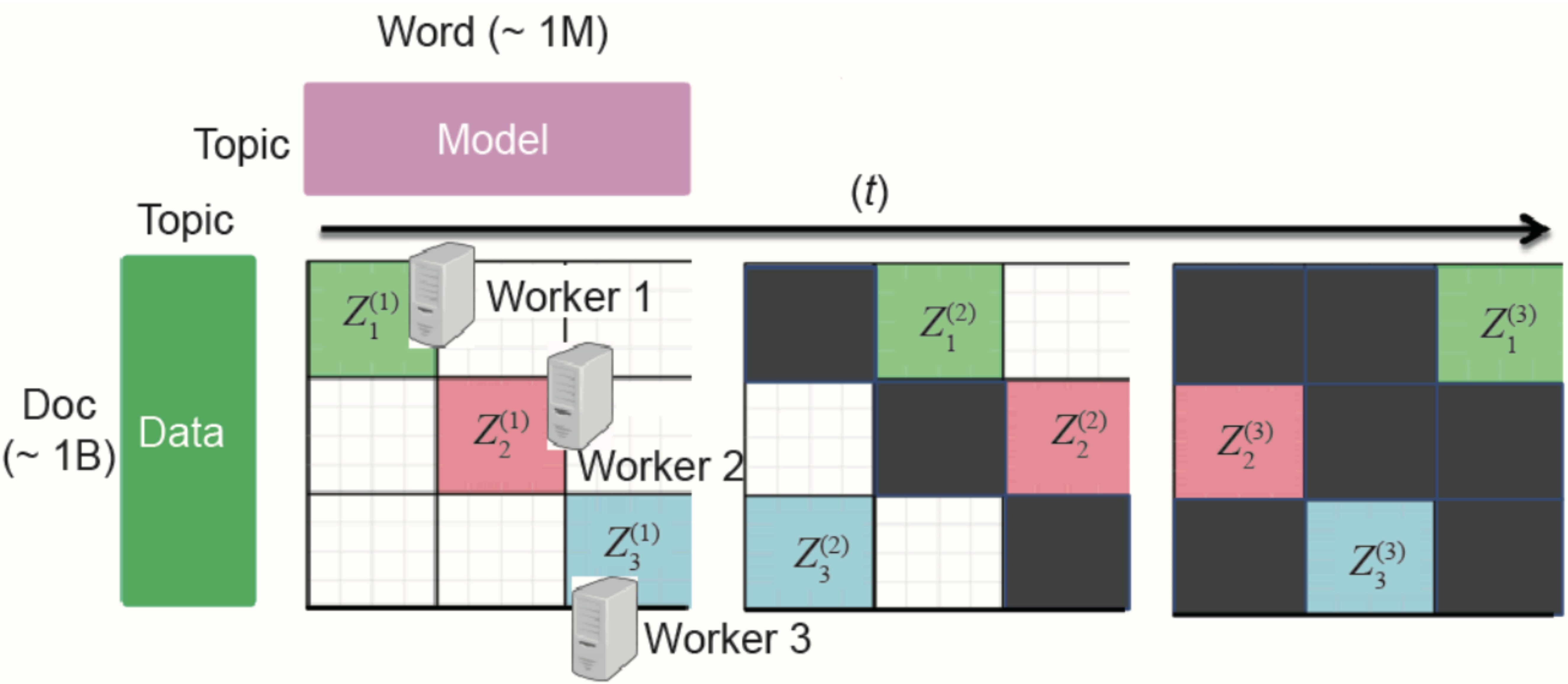
Data Parallelism



Model Parallelism



Data + Model Parallelism



High-level illustration of simultaneous data and model parallelism in LDA top-ic modeling.

Four Principles of ML System

- How to Distribute the Computation?
- How to Bridge Computation and Communication?
- How to Communicate?
- What to Communicate?

How to Distribute?

Structure Aware Parallelisation:

- schedule(): a small number of parameter are prioritised, and dependency checks;
- push(): perform update computation in parallel on worker machines
- pull(): perform F computation

How to Distribute?

Slow-worker agnosticism:

- A solution to straggler problem in ML program

- Faster machine repeat their updates while waiting for the stragglers to catch up.

How to Distribute?

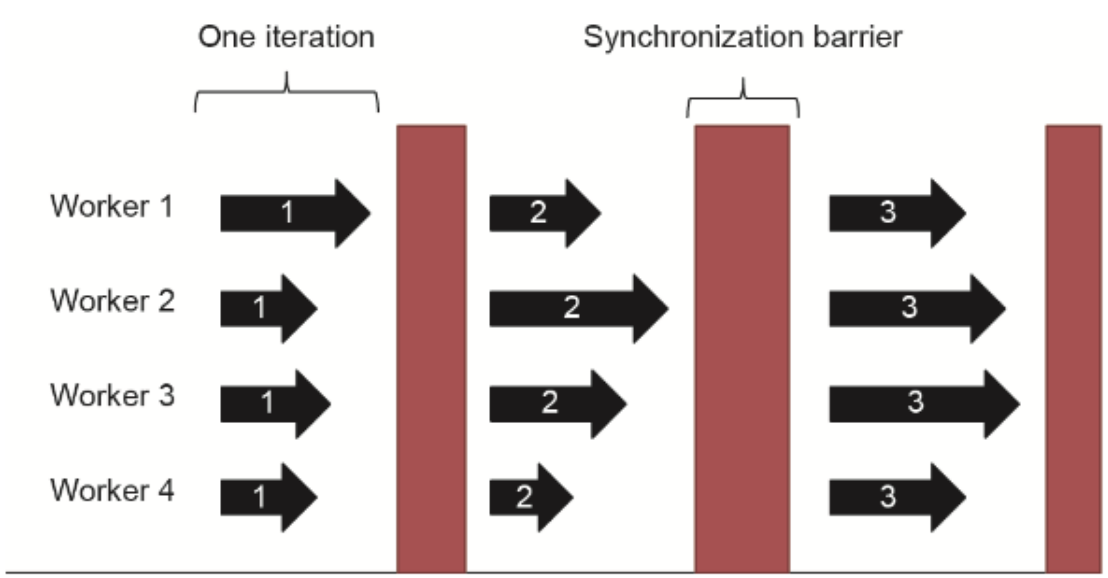
Theorem 1: SAP execution

$$\mathbb{E}\left[\left|A_{\text{ideal}}^{(t)} - A_{\text{SAP}}^{(t)}\right|\right] \leq \frac{2dPm}{(t+1)^2 \hat{P}} L^2 X^{\mathrm{T}} XC$$

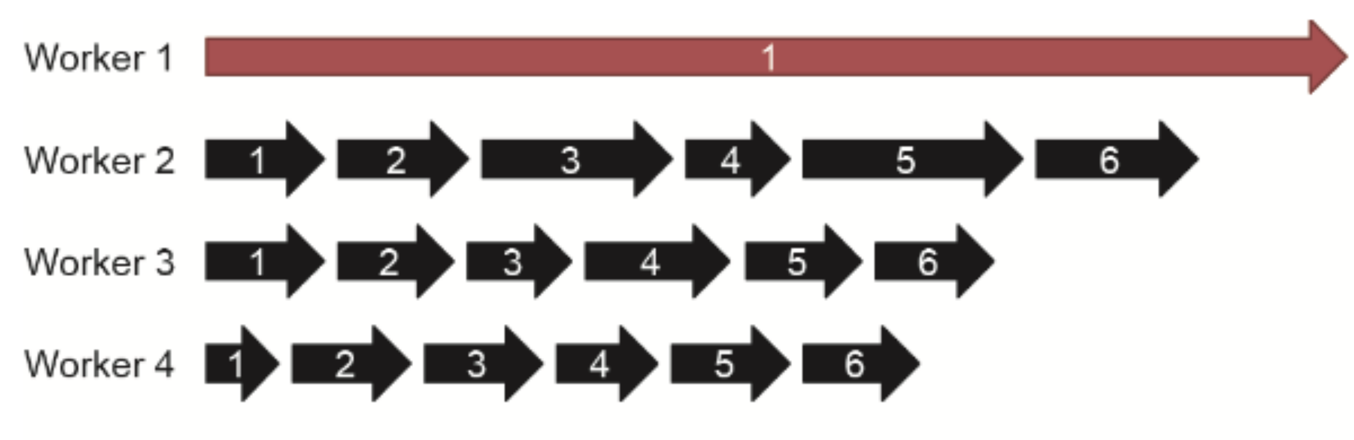
Theorem 2: SAP slow-worker agnosticism

$$\begin{aligned}\mathrm{Var}\left(A^{+n_p}\right) = &\mathrm{Var}(A) - c_1 \eta_t n_p \mathrm{Var}(A) - c_2 \eta_t n_p \mathrm{CoVar}(A, \nabla \mathcal{L}) + \\ &c_3 \eta_t^2 n_p + O(\text{cubic})\end{aligned}$$

How to Bridge?

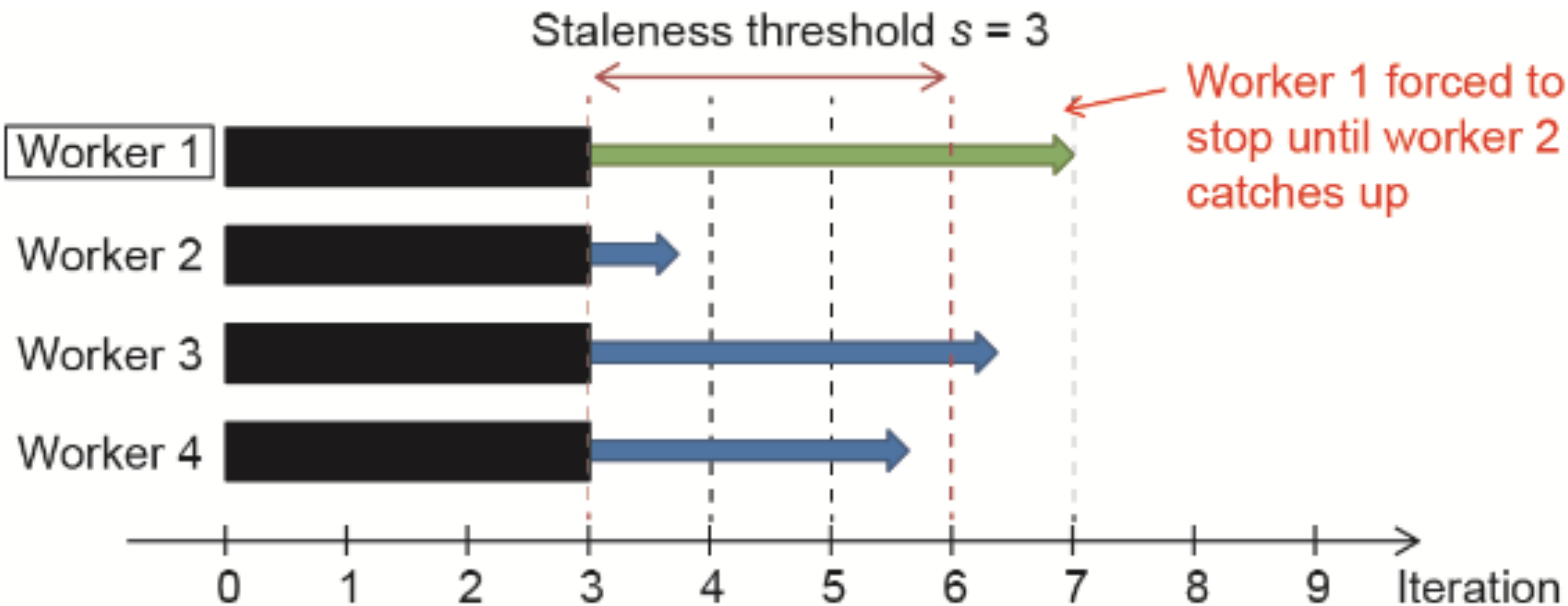


Bulk synchronous parallel



Asynchronous parallel execution

How to Bridge?



Stale Synchronous Parallel

How to Bridge?

Theorem 3: SSP data parallel

$$P\left[\frac{R[A]}{T} - \frac{1}{\sqrt{T}}\left(\eta L^2 + \frac{F^2}{\eta} + 2\eta L^2 \mu_\gamma\right) \geq \tau\right]$$

$$\leq \exp\left\{\frac{-T\tau^2}{2\bar{\eta}_T \sigma_\gamma + \frac{2}{3}\eta L^2 (2s+1) P\tau}\right\}$$

Theorem 4: SSP model parallel

(1) $\sum_{t=0}^{\infty} \|A(t+1) - A(t)\|^2 < \infty$;

(2) $\lim_{t\to\infty} \|A(t+1) - A(t)\| = 0$, *and for all p,* $\lim_{t\to\infty} \|A(t) - A^p(t)\| = 0$;

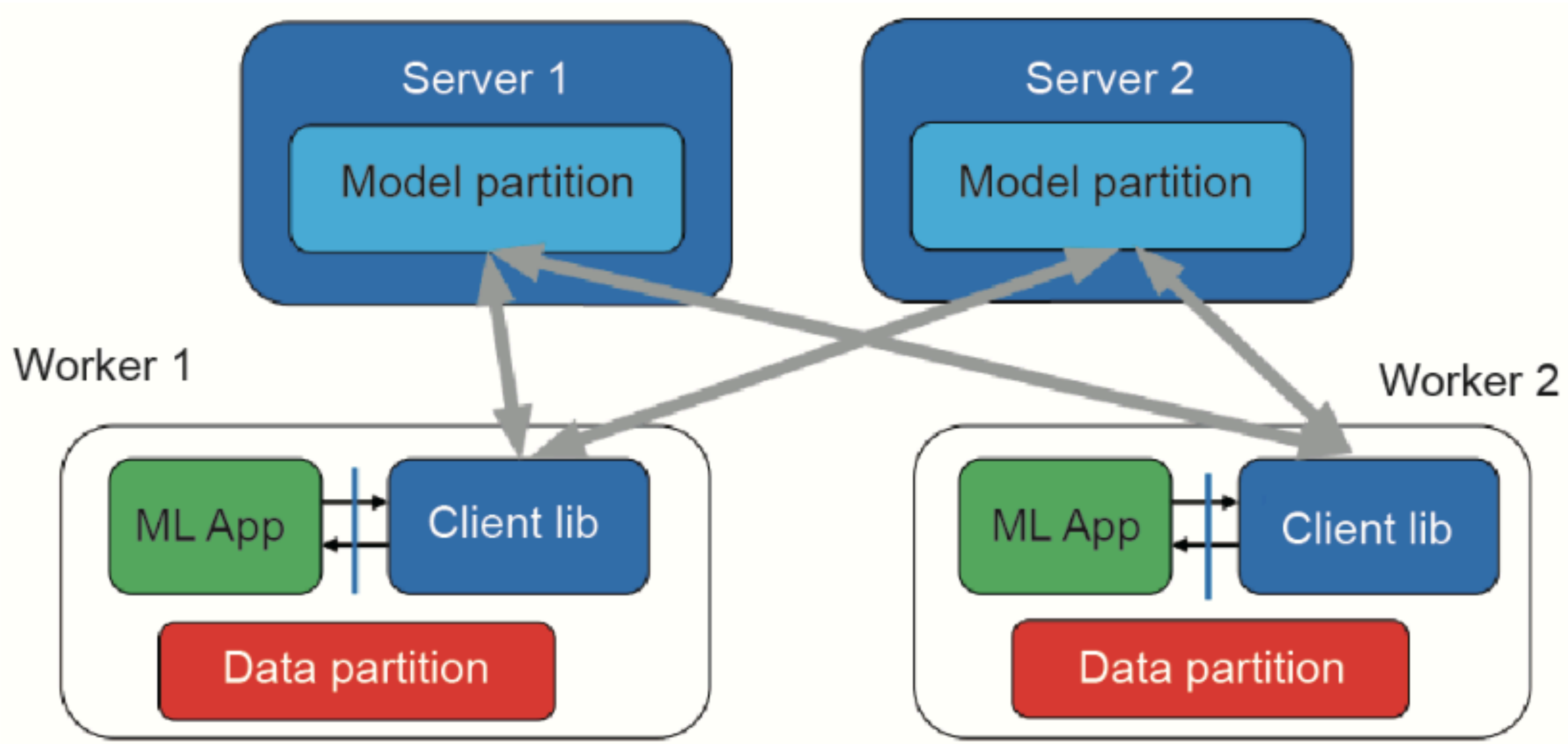
(3) The limit points of $\{A(t)\}$ *coincide with those of* $\{A^p(t)\}$, *and both are critical points of* $\mathcal{L}$.

How to Communicate?

Communication management:

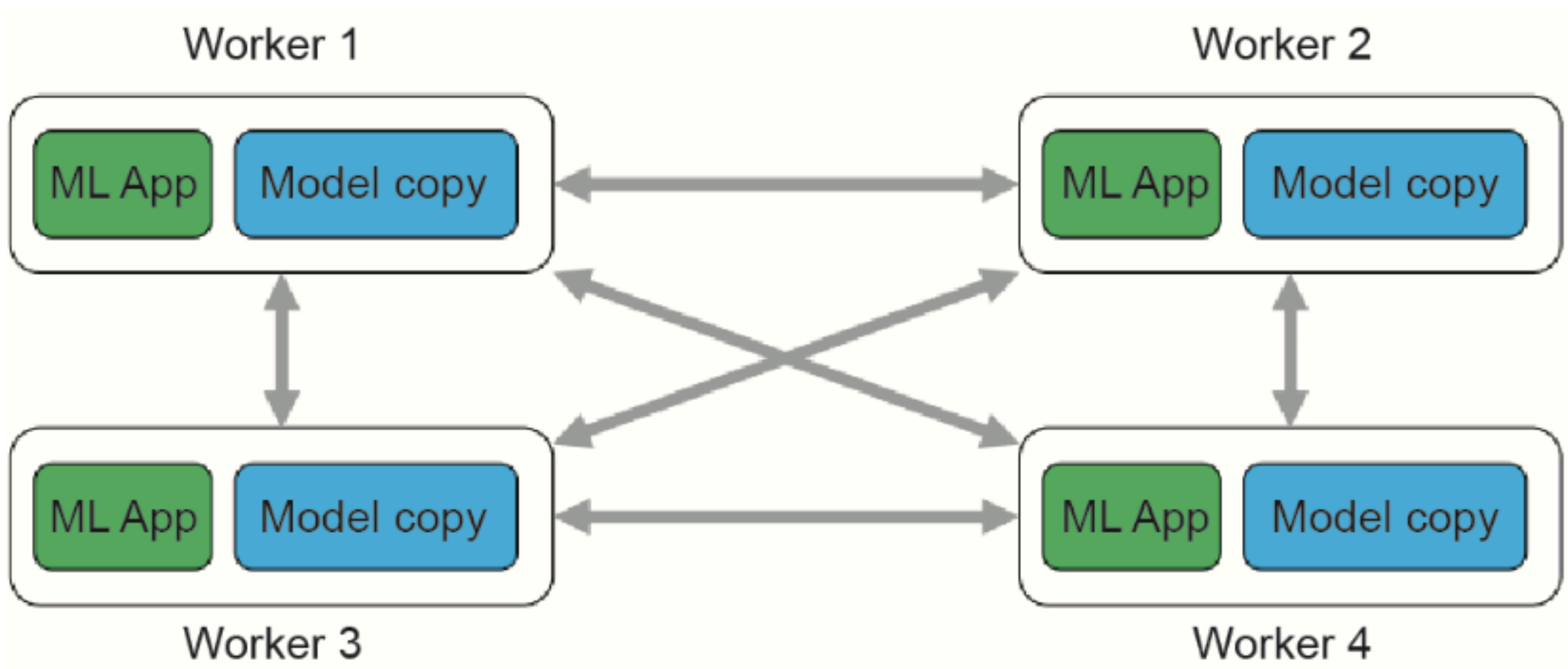
- Continuous communication
- Update Prioritisation
- Parameter Storage and Communication Topologies

How to Communicate?



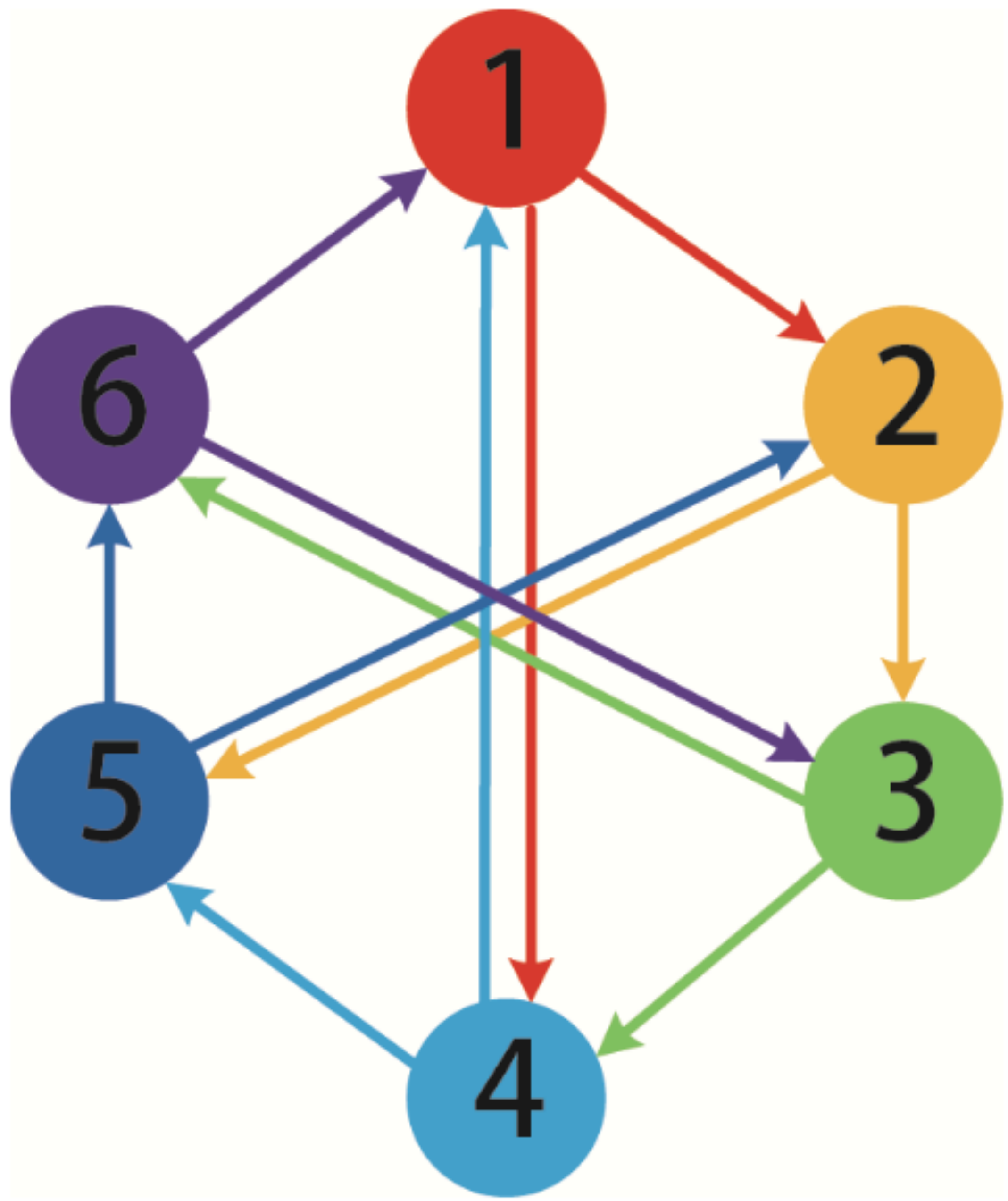
Master-Slave network topology

How to Communicate?



Peer-to-peer network topology

How to Communicate?



Halton Sequence network topology

What to Communicate?

Matrix-Parameterized Models (MPMs)

Matrix parameter W

$$\min_{W} \underbrace{\frac{1}{N}\sum_{i=1}^{N} f_i(\overline{W}a_i;b_i)}_{\text{Loss function}} + \underbrace{h(W)}_{\text{Regularizer}}$$

Loss function

Regularizer

Distance Metric Learning, Sparse Coding, Distance Metric Learning, Group Lasso, Neural Network, etc.

What to Communicate?

Sufficient Factor (SF) Updates

[Xie et al., 2015]

- **Full parameter matrix update** ΔW can be computed as **outer product of two vectors** uv^T (called sufficient factors)
  - Primal stochastic gradient descent (SGD)

$$\min_W \frac{1}{N}\sum_{i=1}^{N} f_i(Wa_i;b_i) + h(W)$$

$$\boxed{\Delta W = uv^{\mathrm{T}} \quad u = \frac{\partial f(Wa_i,b_i)}{\partial (Wa_i)} \quad v = a_i}$$

  - Stochastic dual coordinate ascent (SDCA)

$$\min_Z \frac{1}{N}\sum_{i=1}^{N} f_i^*(-z_i) + h^*(\frac{1}{N}ZA^{\mathrm{T}})$$

$$\boxed{\Delta W = uv^{\mathrm{T}} \quad u = \Delta z_i \quad v = a_i}$$

- Send the lightweight SF updates (u,v), instead of the expensive full-matrix ΔW updates!

What to Communicate?

Theorem 5 (adapted from Ref. [55]): **SFB under SSP, convergence theorem.** *Let $\mathbf{A}_p(t)$, $p = 1,\ldots, P$, and $\mathbf{A}(t)$ be the local worker views and a "reference" view respectively, for the ML objective function $\mathcal{L}$ in Eq.* (16) *(assuming $r \equiv 0$) being solved by SFB under the SSP bridging model with staleness s. Under mild assumptions, we have*

(1) $\lim_{t\to\infty} \max_p \left\|\mathbf{A}(t) - \mathbf{A}_p(t)\right\| = 0$, *that is, the local worker views converge to the reference view, implying that all worker views will be the same after sufficient iterations t.*

(2) There exists a common subsequence of $\mathbf{A}_p(t)$ and $\mathbf{A}(t)$ that converges almost surely to a stationary point of $\mathcal{L}$, with rate $O\left(\frac{Ps\log(t)}{\sqrt{t}}\right)$.

Theorem to show convergent rate of SFB

What to Communicate?

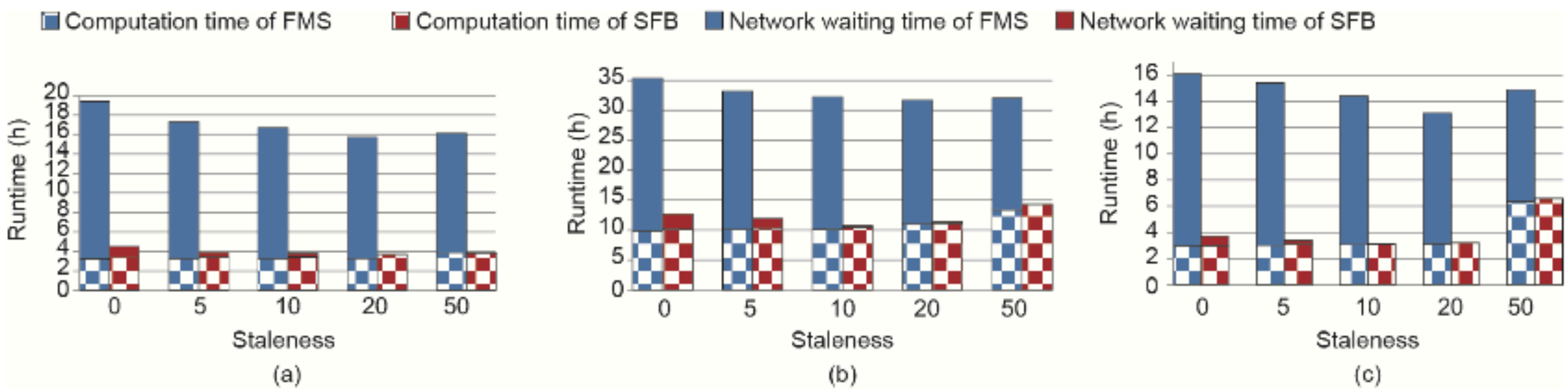
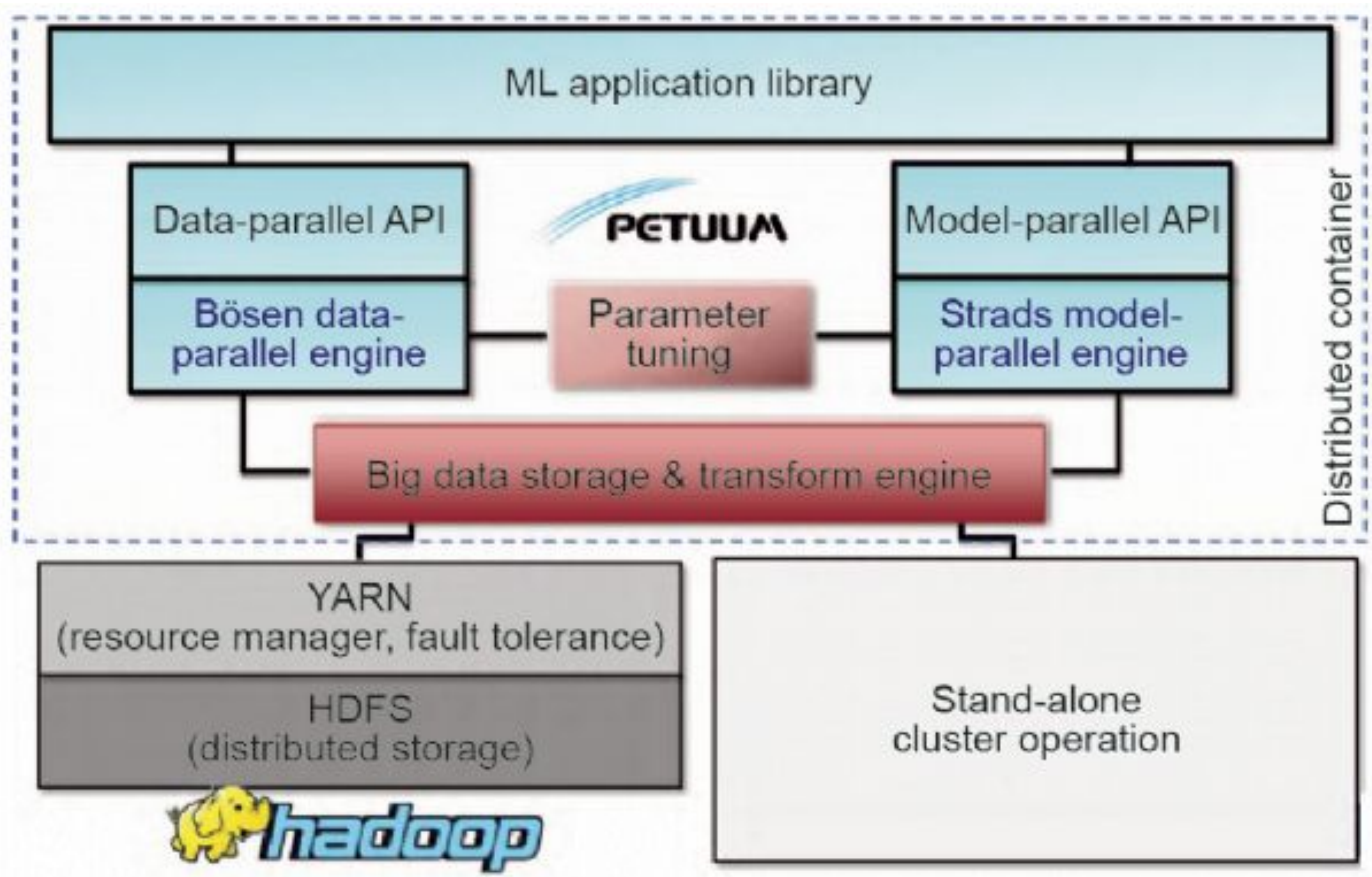


Fig. 18. Computation time versus network waiting time for (a) MLR, (b) DML, and (c) L2-MLR.

Empirically SFB is more efficient than FMB.

Petuum



Architecture of Petuum

Summary

- Machine Learning is different from traditional big data programming.
- Data parallelism and mode parallelism.
- Principles on distribution and communication.