Robust Distributed Learning

El Mahdi El Mhamdi

4th year PhD student,
Distributed Computing Laboratory, EPFL

Machine Learning is Successful

1972: Backprop (Werbos), Hinton & co on text

1989: LeNet neural networks, SVMs…

2012: ImageNet challenge

2016: Human defeat in the game of Go

2017-2019: Poker, Medical diagnosis, DeepFakes…

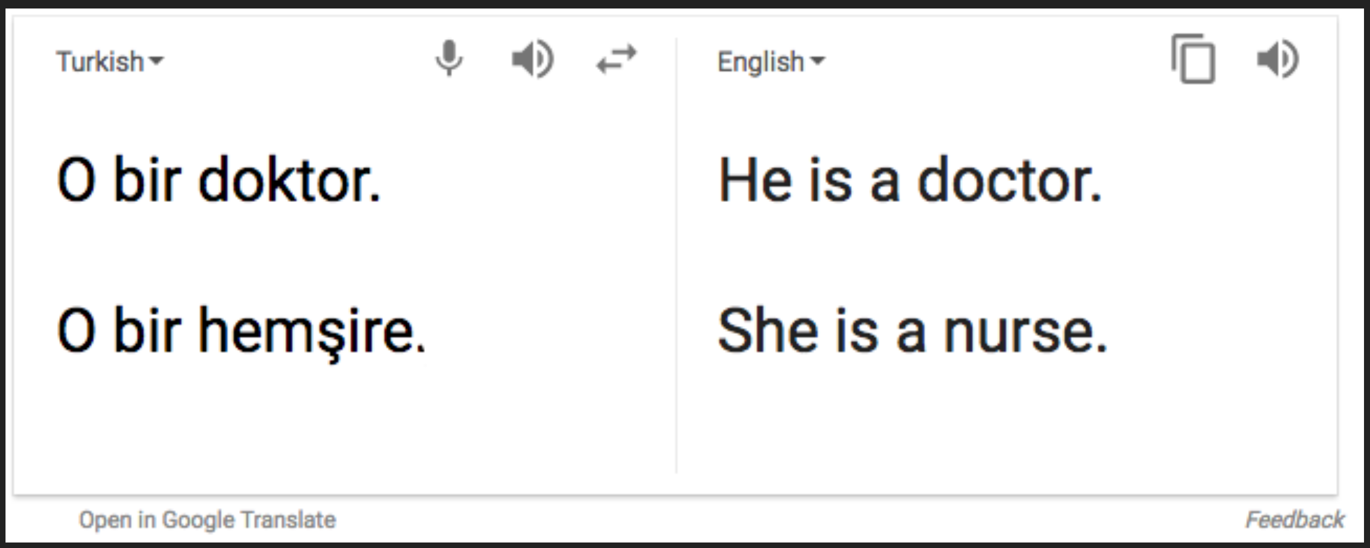
Machine Learning is Distributed

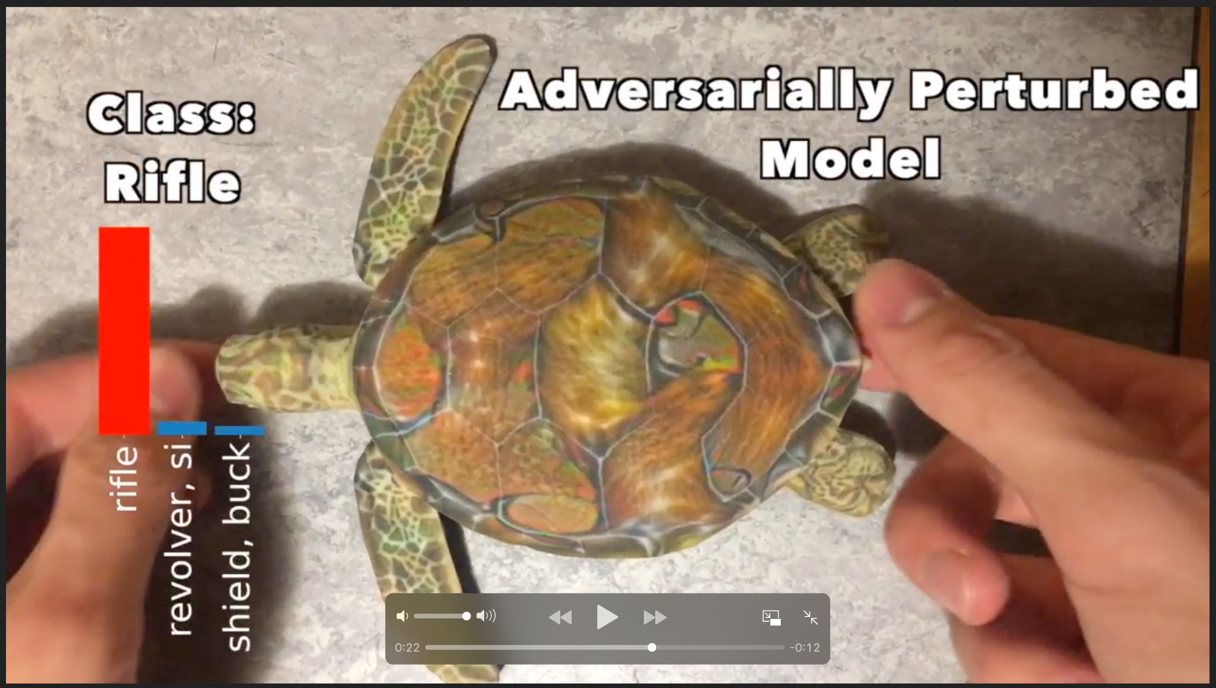
Machines: “*16,000 computers to identify a cat*” (N.Y. Times 2012)

Data: 3.9 billion internet users distributed across the globe

Models: Distributed representations (‘neural’ networks)

Machine Learning is Vulnerable






LATEST OBSESSIONS FEATURED QUARTZ EMAILS EDITIONS BECOME A MEMBER

AI experts are calling on Amazon to stop selling facial recognition to law enforcement

By Dave Gershgorn · April 3, 2019

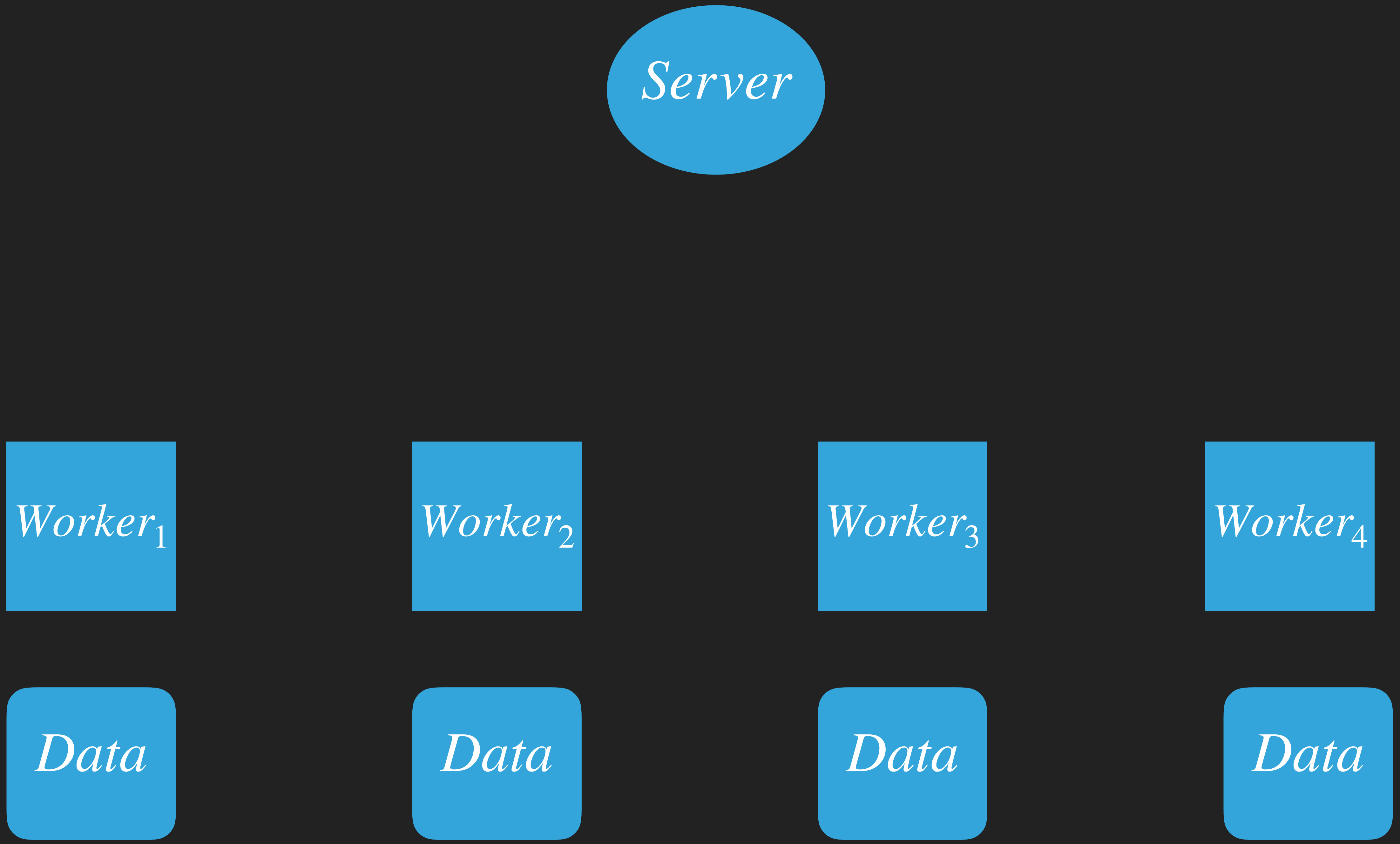
Adversarial Machine Learning

Evasion Attacks (misleading input to trained model, Goodfellow et al. NeurIPS 2014, Madry et al. ICLR 2017, Everyone et al. 201x…)

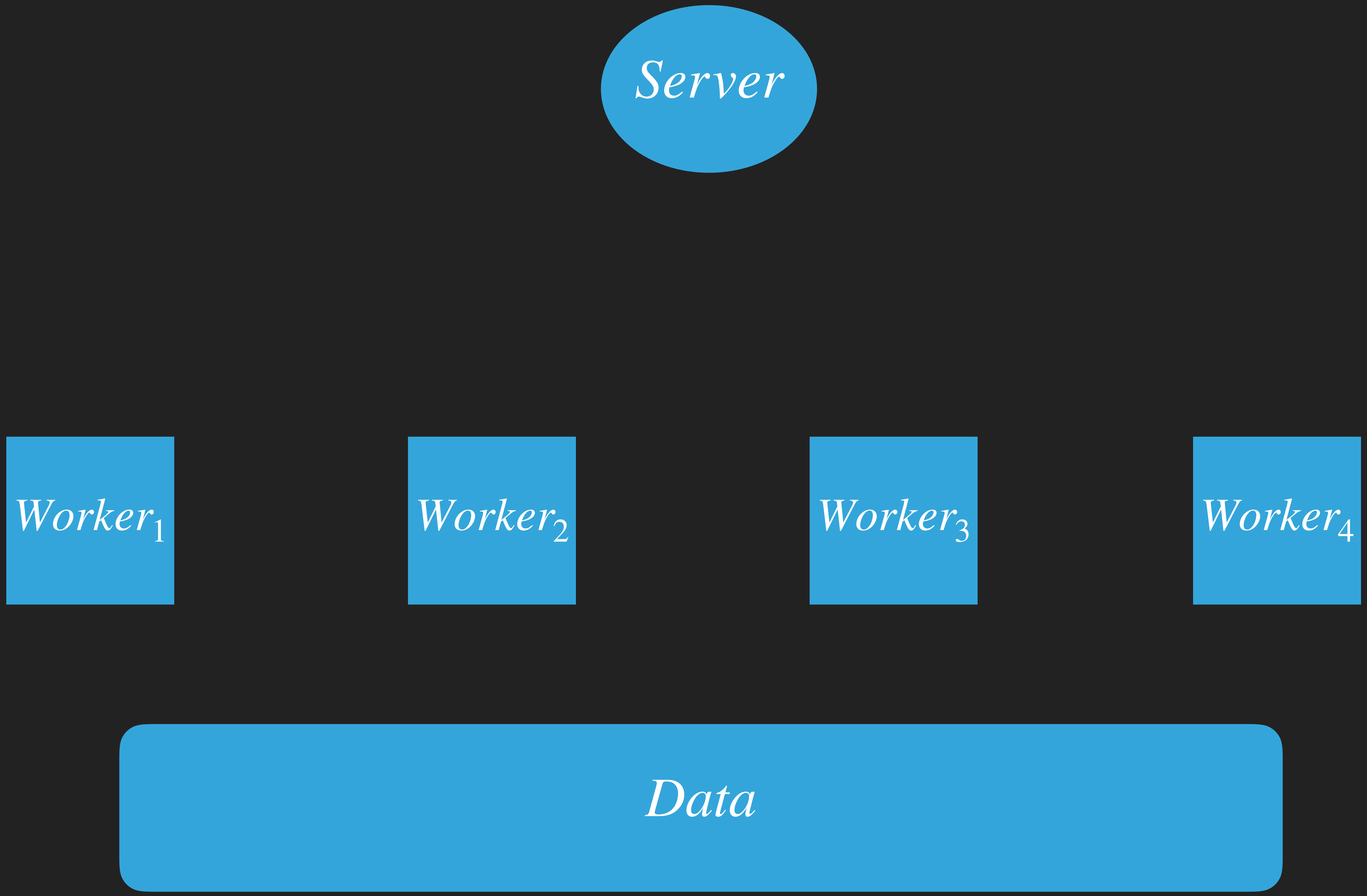
Exploration Attacks (inferring privacy-sensitive info: differential privacy, secure aggregation…)

Poisoning Attacks (Biggio et al. ICML 2012, Stern et al. 2004, **this thesis**)

How ML is Distributed Today

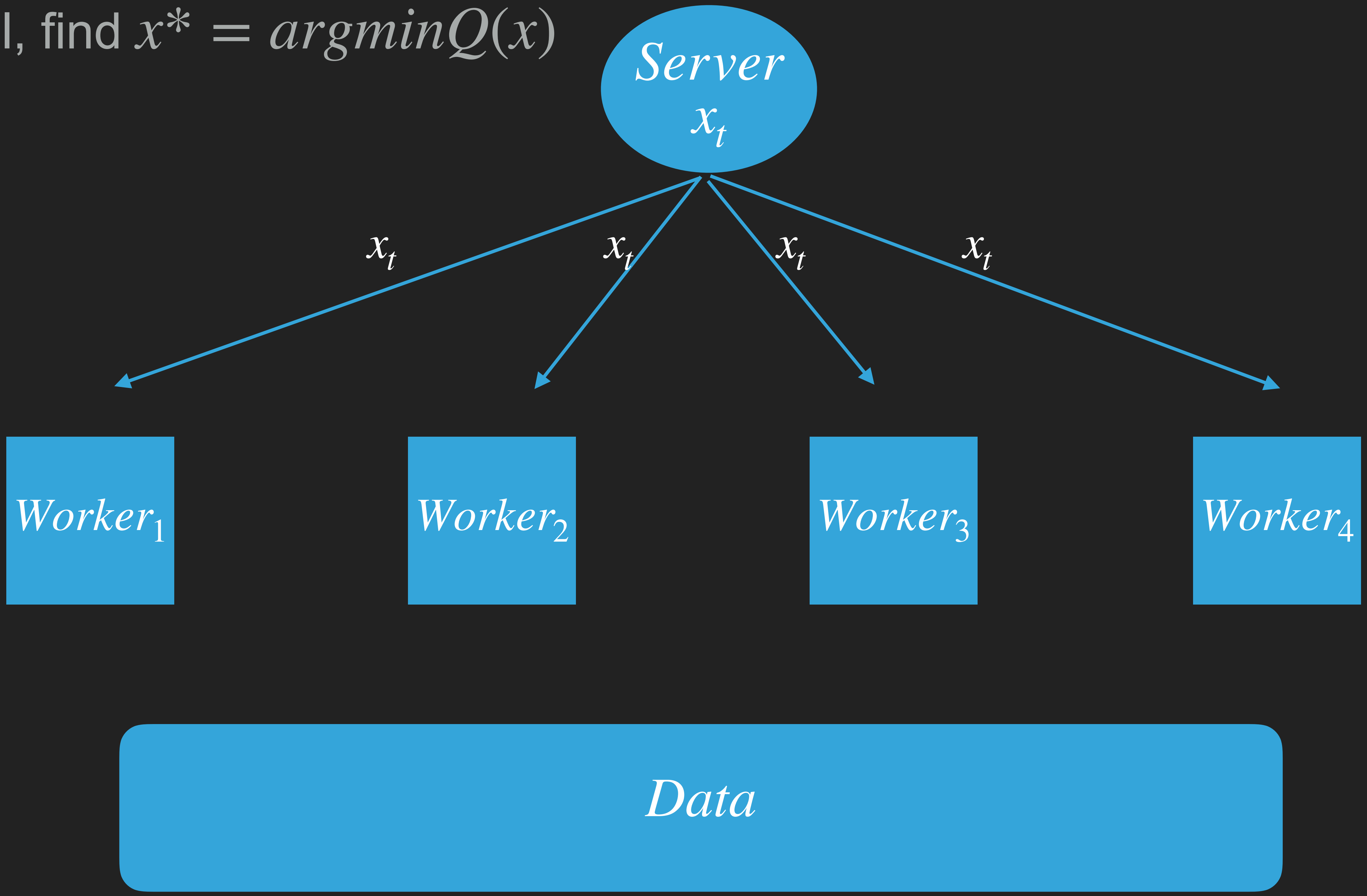


How ML is Distributed Today

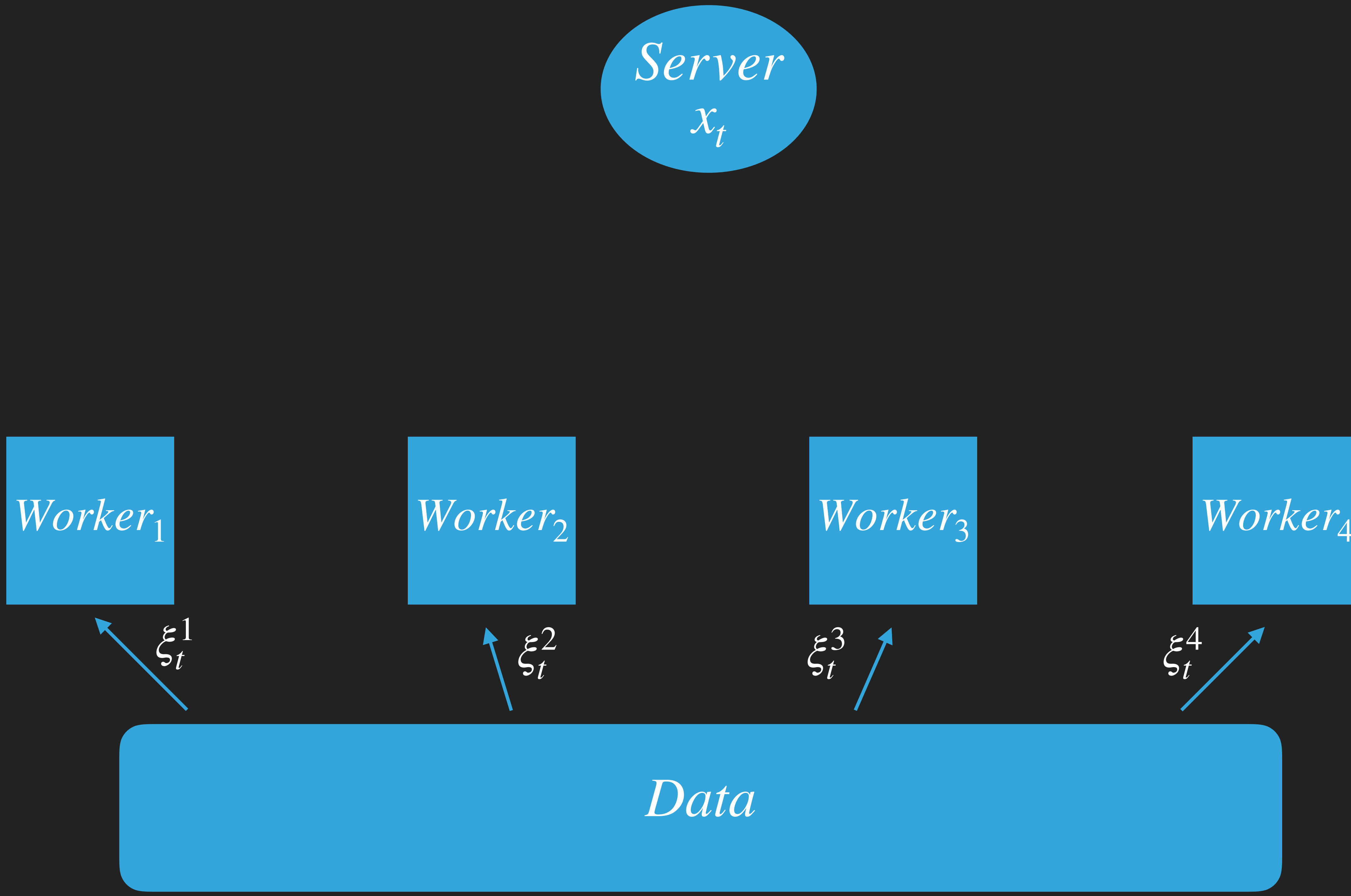


How ML is Distributed Today

goal, find $x^* = argminQ(x)$



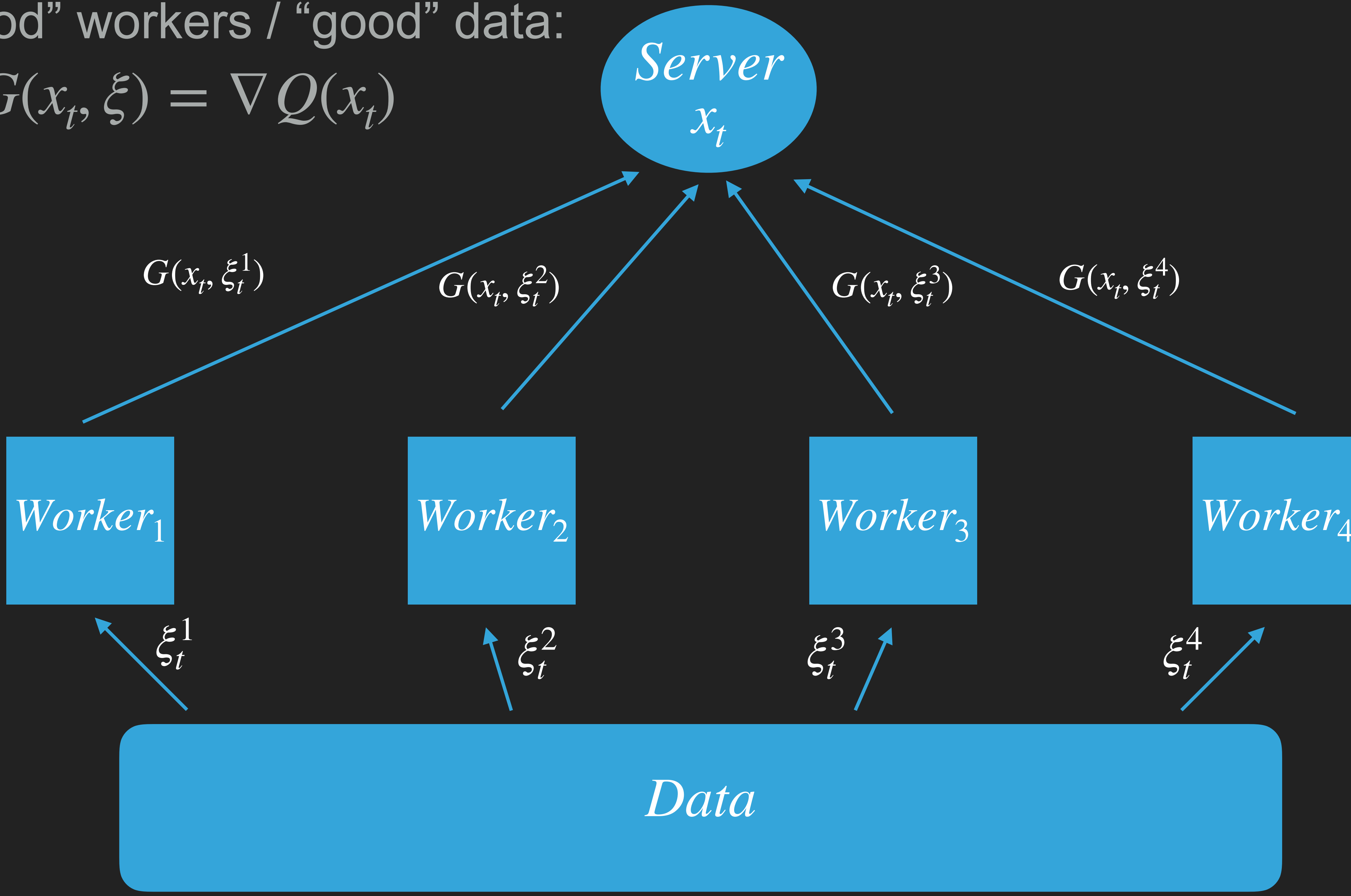
How ML is Distributed Today



How ML is Distributed Today

"good" workers / "good" data:

$$\mathbb{E}_{\xi} G(x_t, \xi) = \nabla Q(x_t)$$



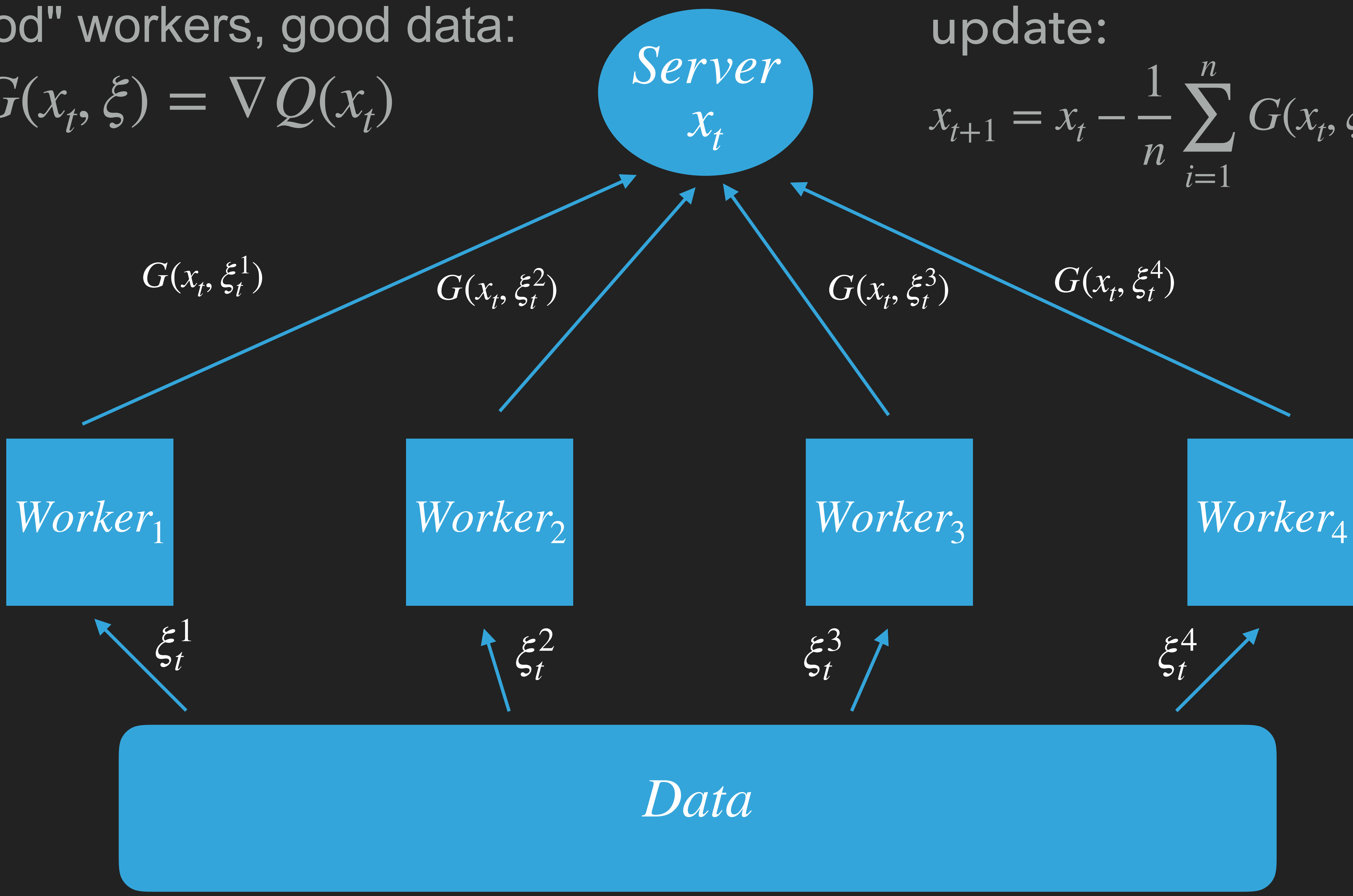
How ML is Distributed Today

“good" workers, good data:

$$\mathbb{E}_{\xi} G(x_t, \xi) = \nabla Q(x_t)$$

update:

$$x_{t+1} = x_t - \frac{1}{n} \sum_{i=1}^{n} G(x_t, \xi_t^i)$$



How ML is Distributed Today

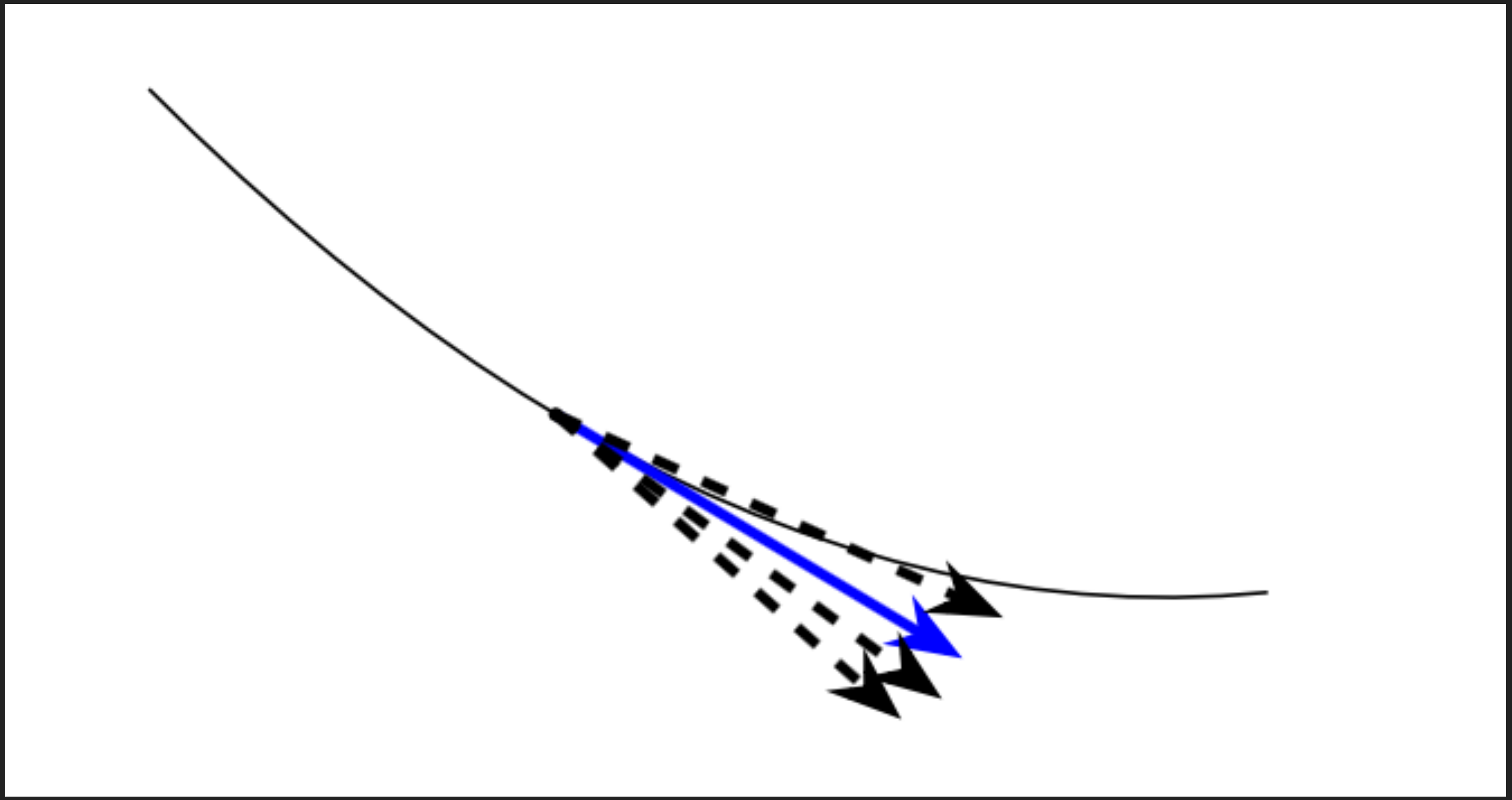
“good" workers, good data:

$$\mathbb{E}_\xi G(x_t, \xi) = \nabla Q(x_t)$$

Server
x_t

update:

$$x_{t+1} = x_t - \frac{1}{n}\sum_{i=1}^{n} G(x_t, \xi_t^i)$$



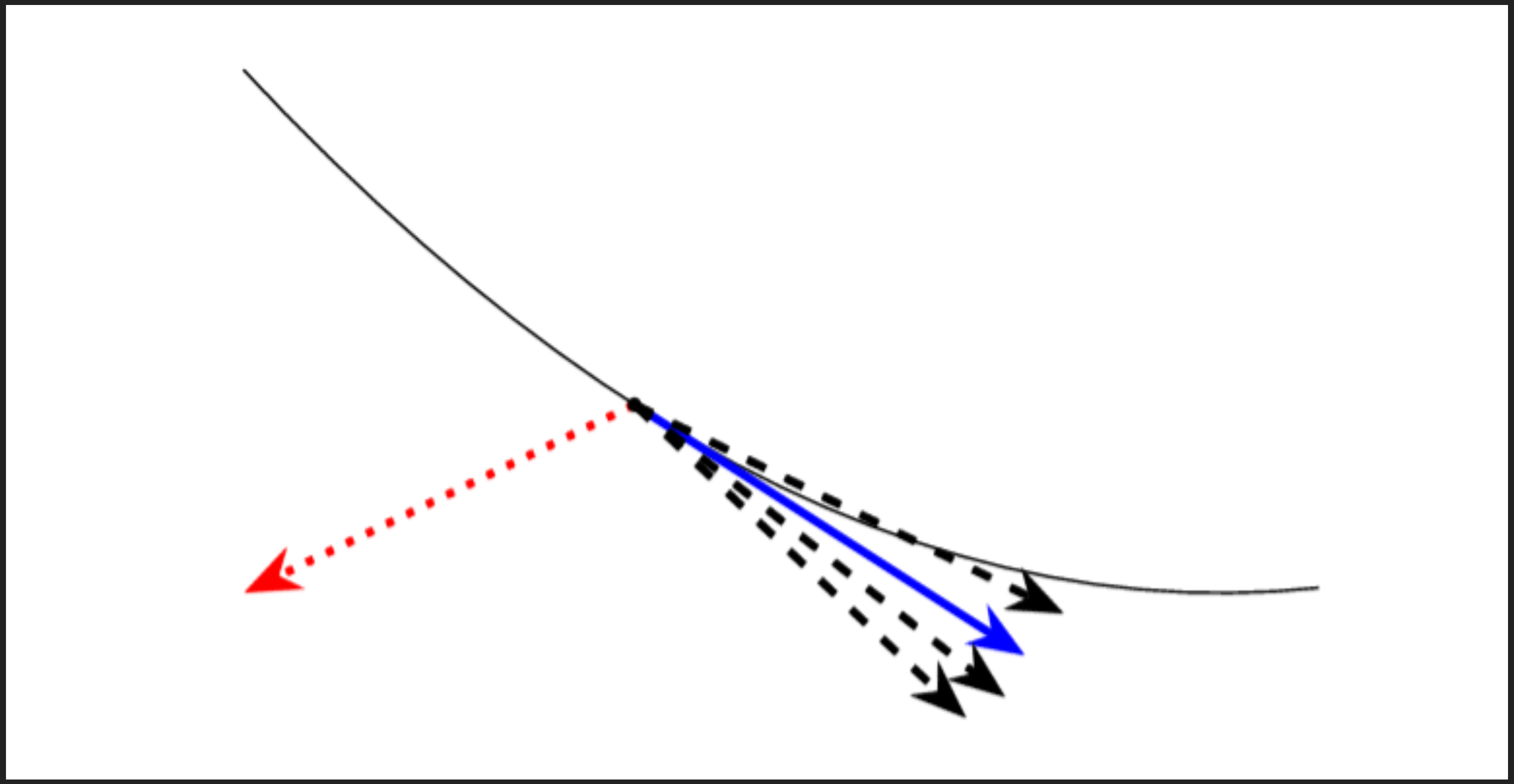
The Problem with How ML is Distributed Today

bad workers / bad data

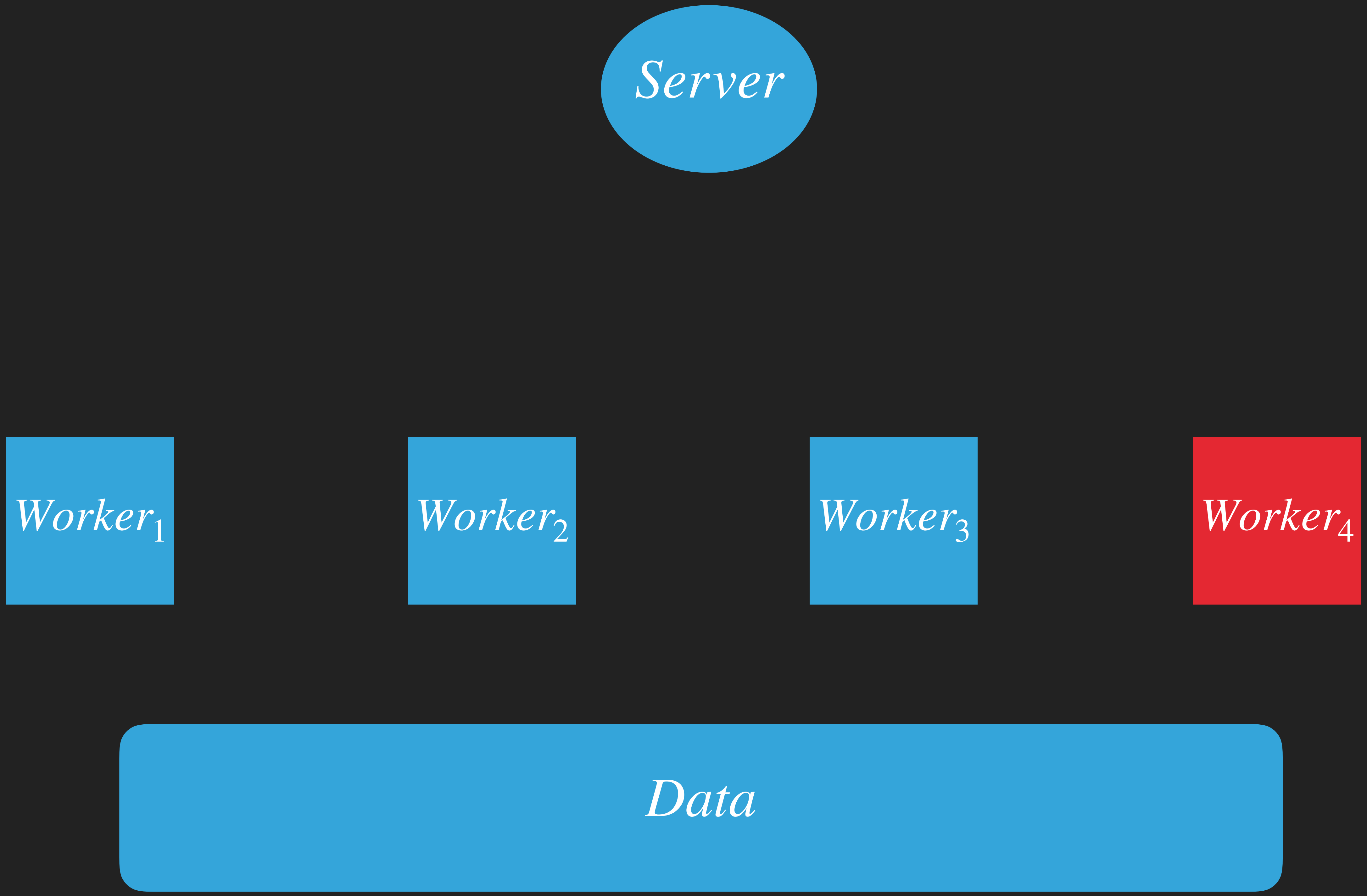
Server
x_t

update:

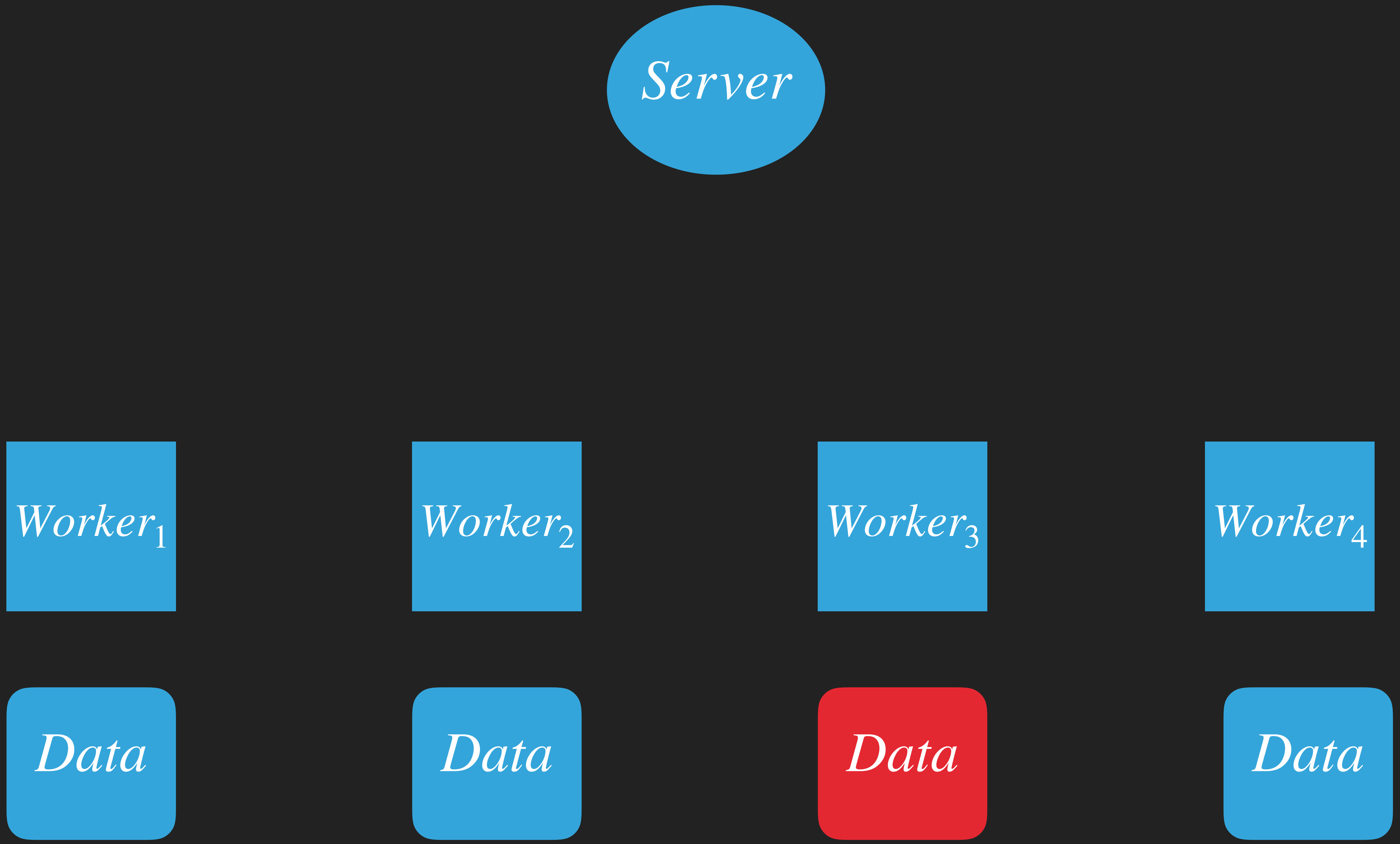
$$x_{t+1} = x_t - \frac{1}{n} \sum_{i=1}^{n} G(x_t, \xi_t^i)$$



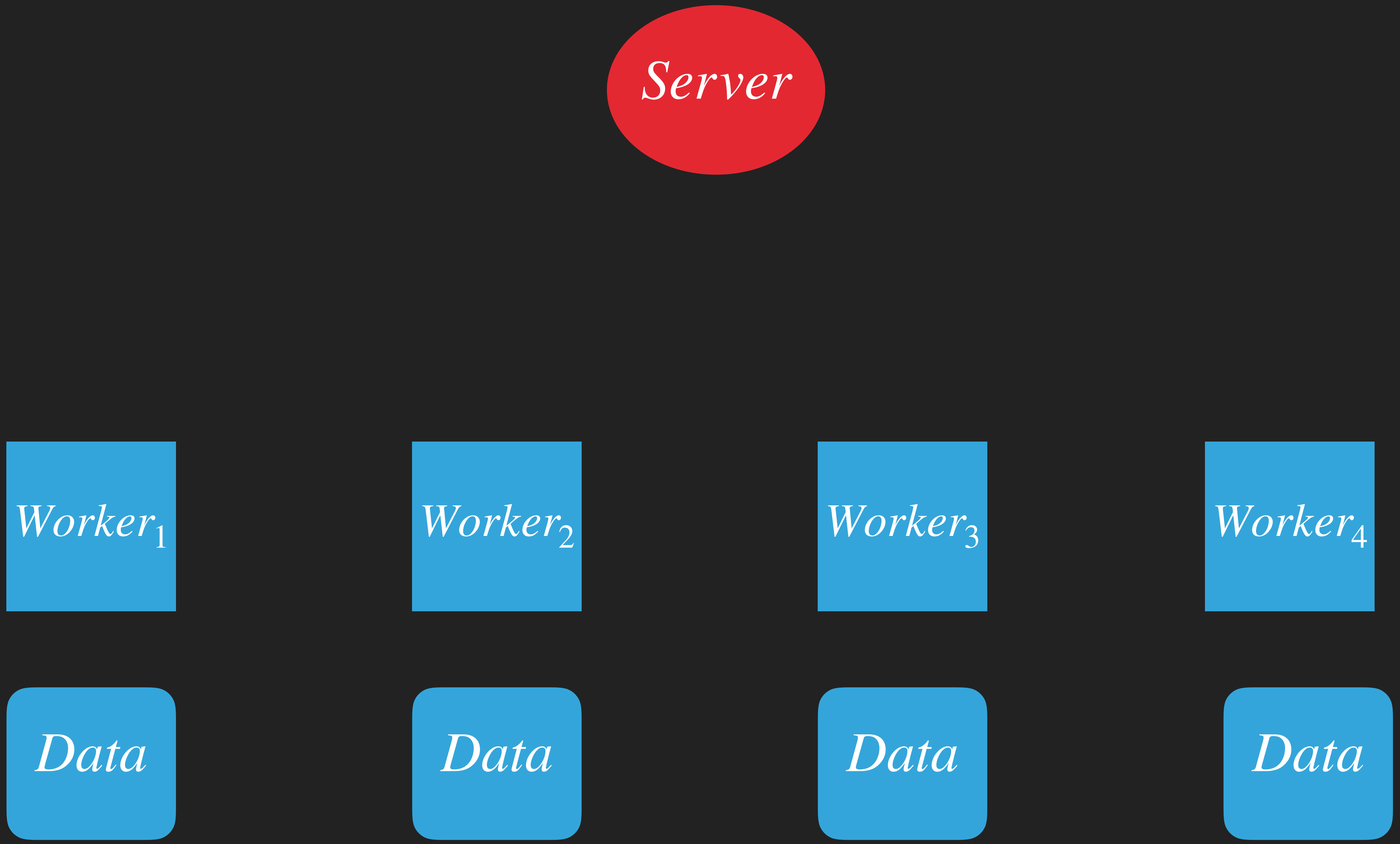
Inevitable failures at all levels



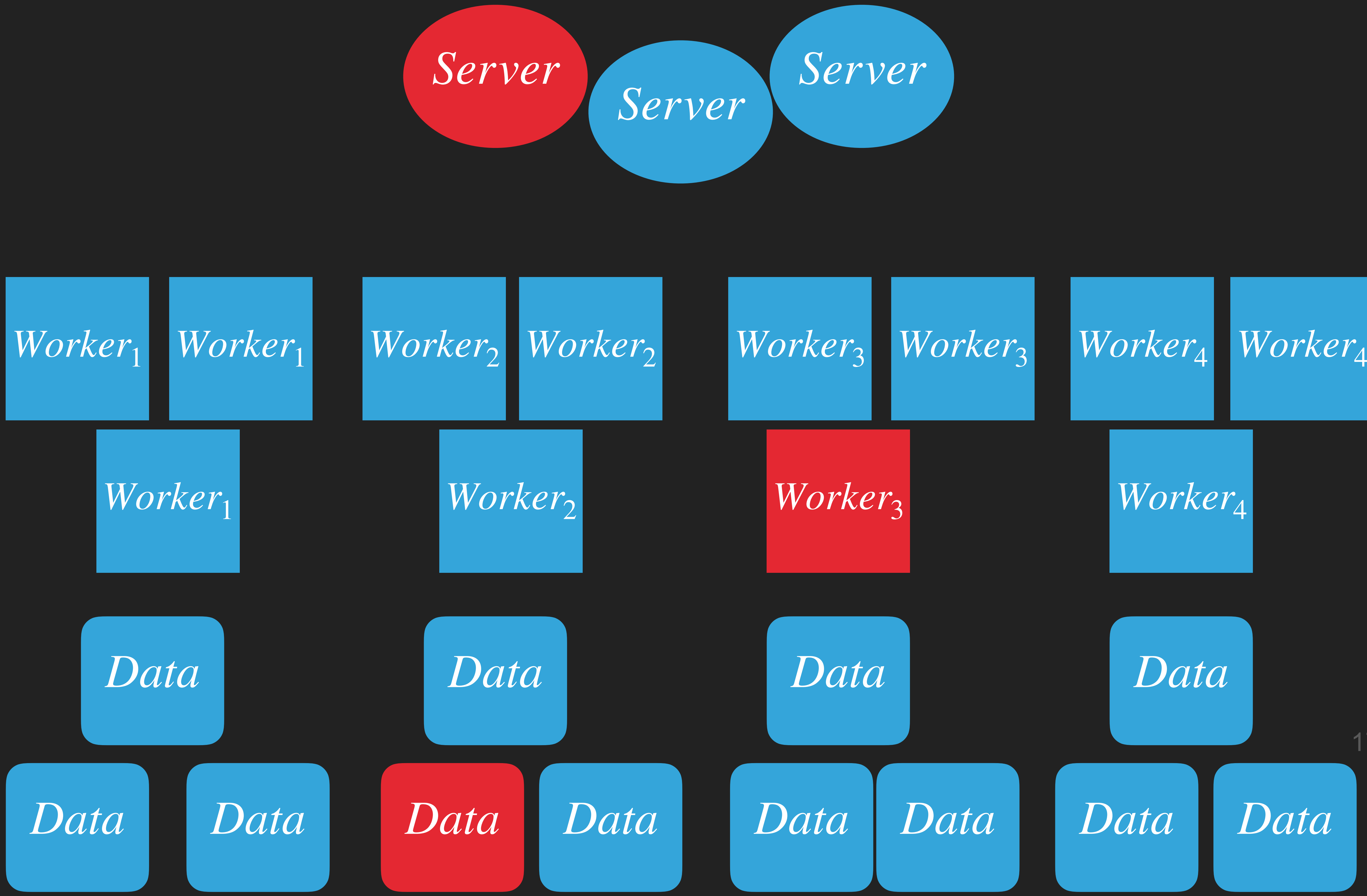
Inevitable failures at all levels



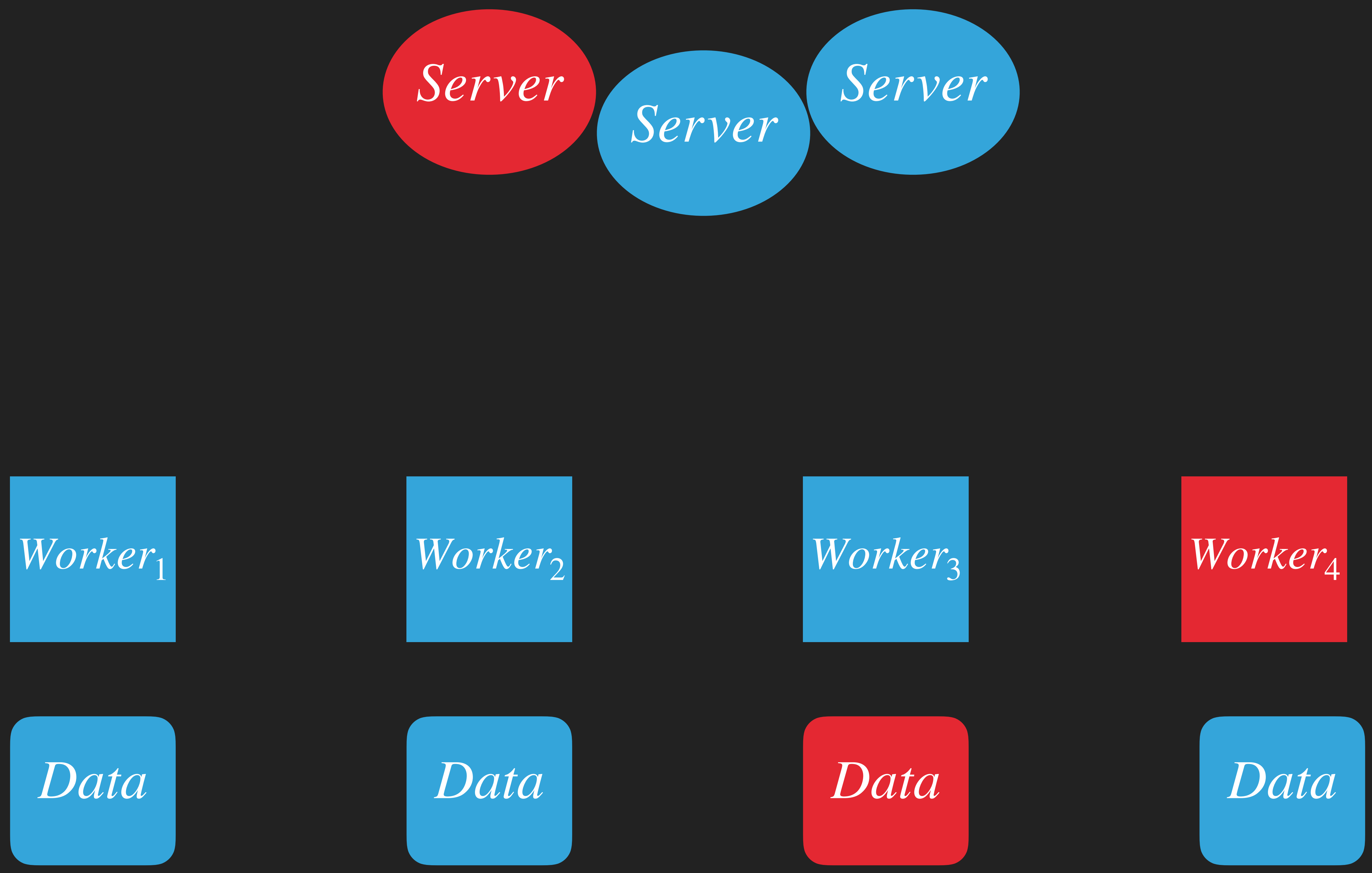
Inevitable failures at all levels



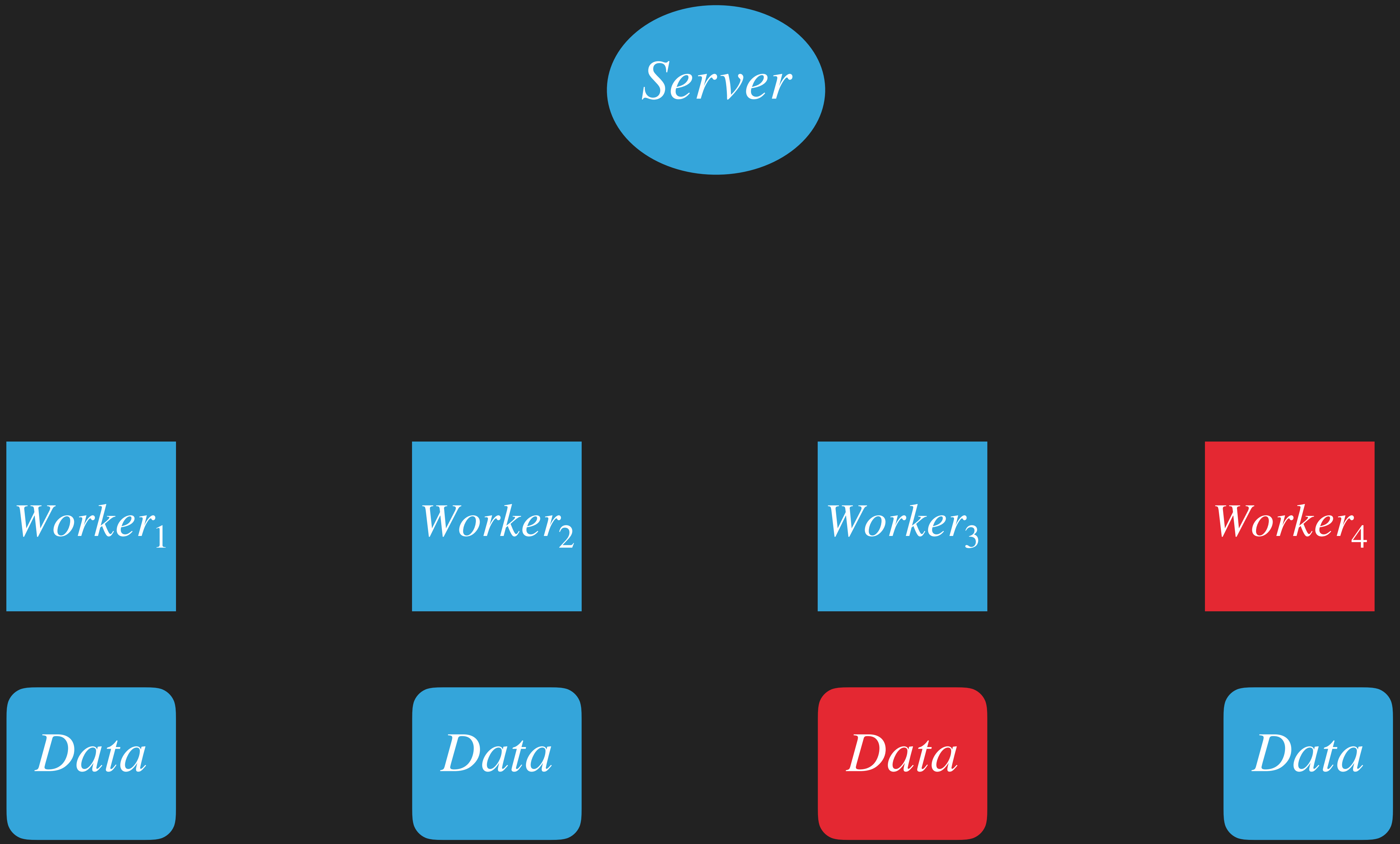
The Overkill of State Machine Replication (SMR)



Only SMR on the server is acceptable

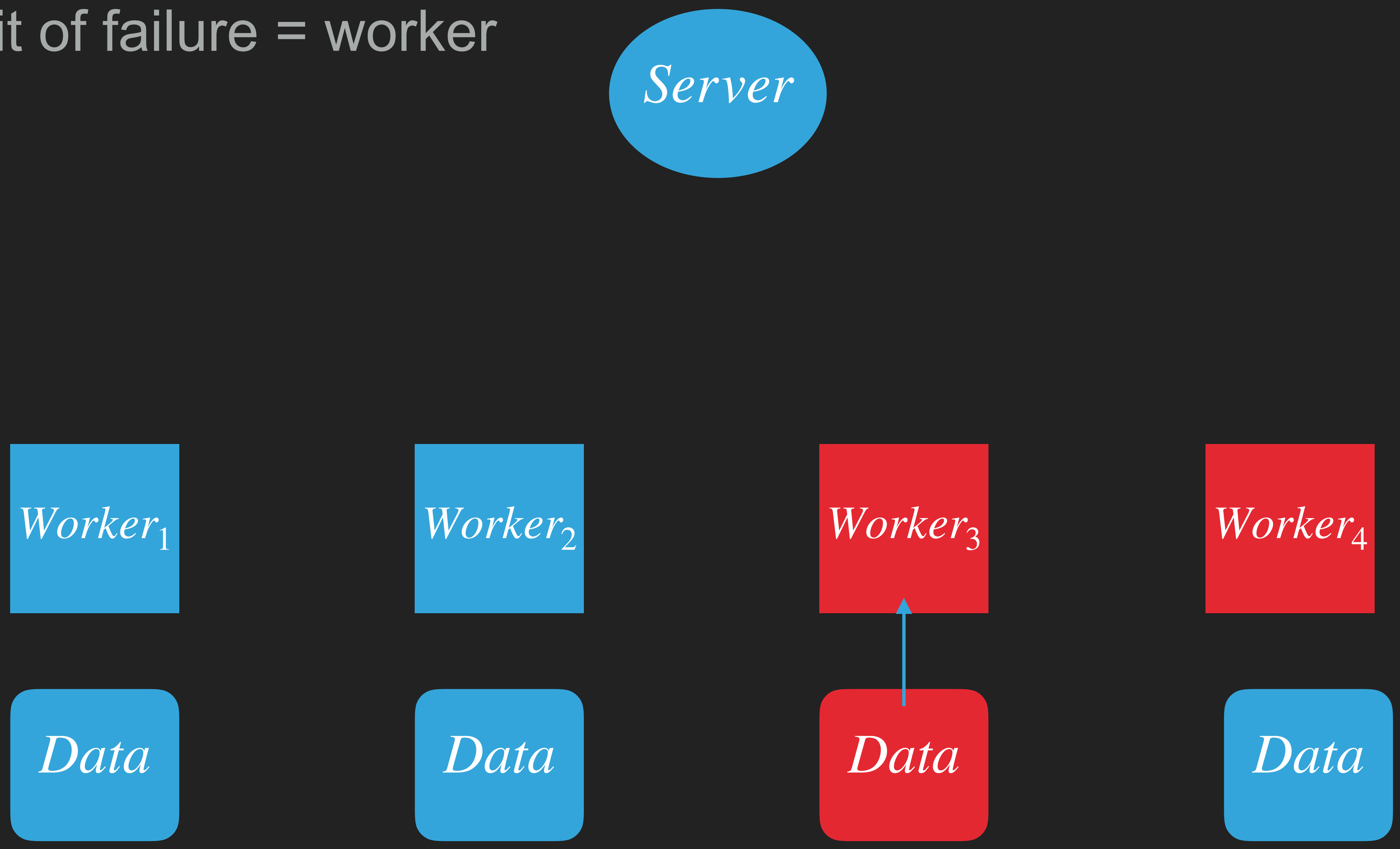


Only SMR on the server is acceptable



Bad data → Bad "worker"

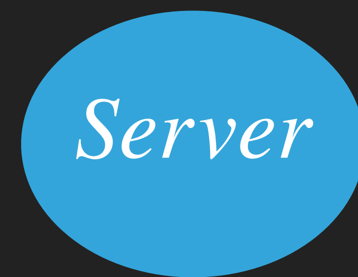
Unit of failure = worker



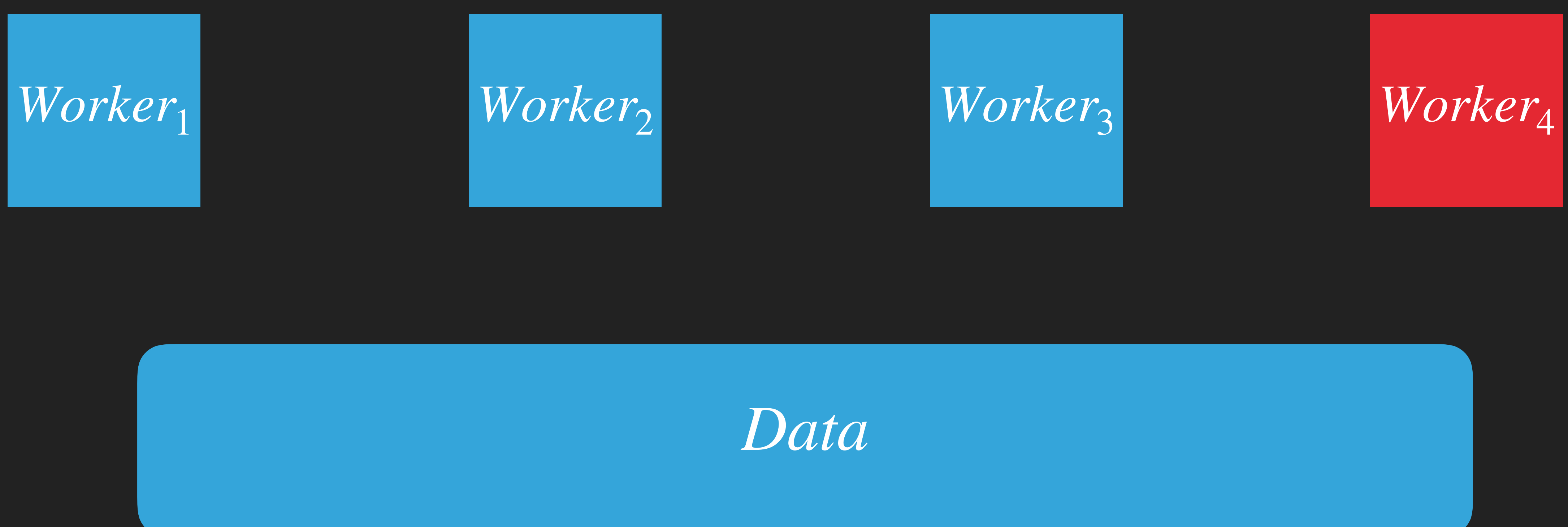
any "unit" of gradient generation can be abstracted as a "worker" (e.g. a social media account)

Setting

1 server
n workers
f Byzantine



Threat model: omniscient ≠ omnipotent



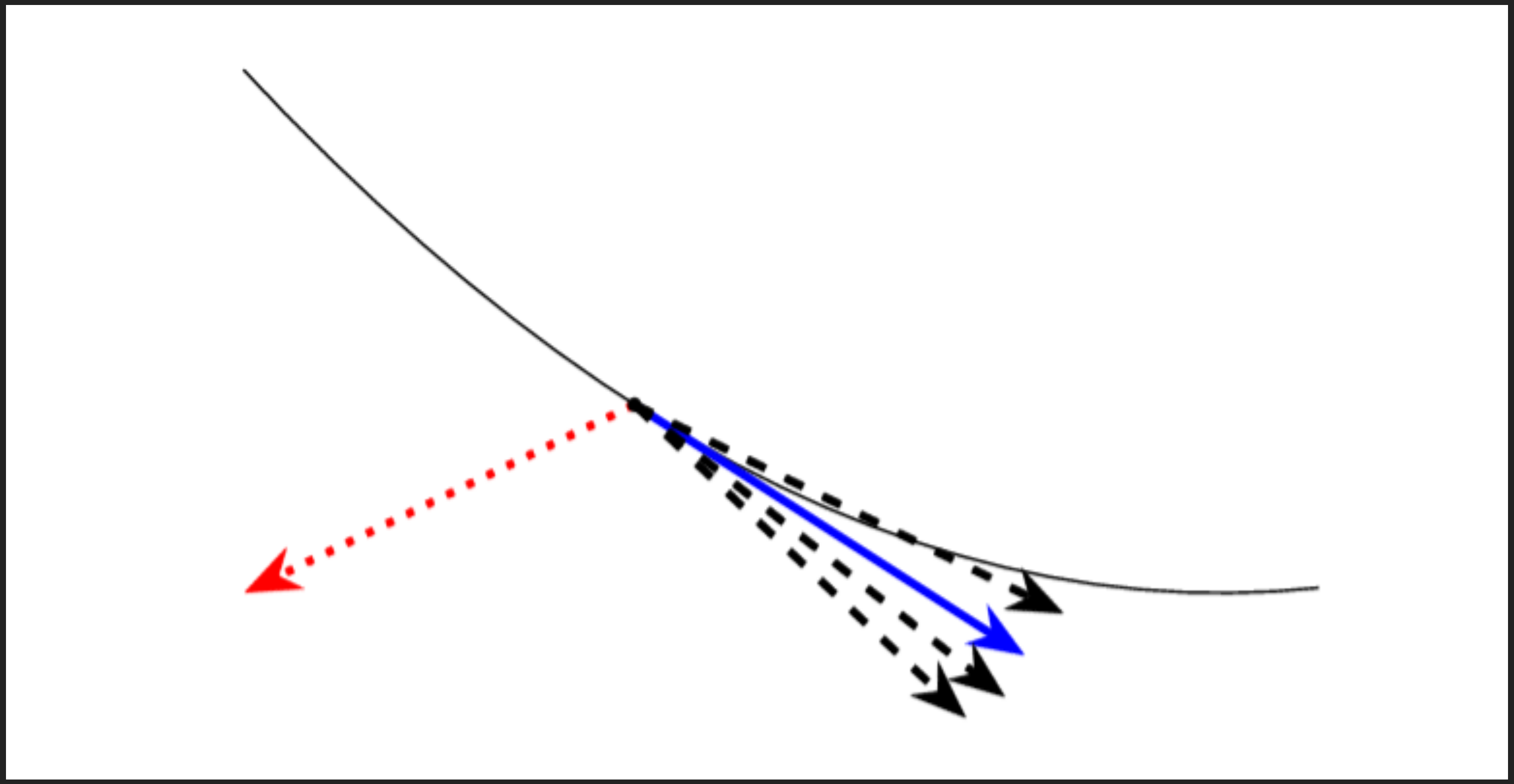
The Problem with How ML is Distributed Today

bad workers / bad data

Server
x_t

update:

$$x_{t+1} = x_t - \frac{1}{n}\sum_{i=1}^{n} G(x_t, \xi_t^i)$$



The Obvious Vulnerability of (distributed) Learning

Our adopted view:

Learning ~ Aggregating Knowledge from Data Points
Learning ~ Some sort of “statistical agreement”
data points “agree” on a model that minimises their loss

Closer problem: Byzantine approximate agreement

Proposed values => agreement on a value within some ϵ

Solution proposed by Mendes and Herlihy (STOC 2013)

However: incompatible with ML requirements:

- $\mathcal{O}(n^d)$ to compute a safe area

- requires $n = \Omega(f.d)$ workers

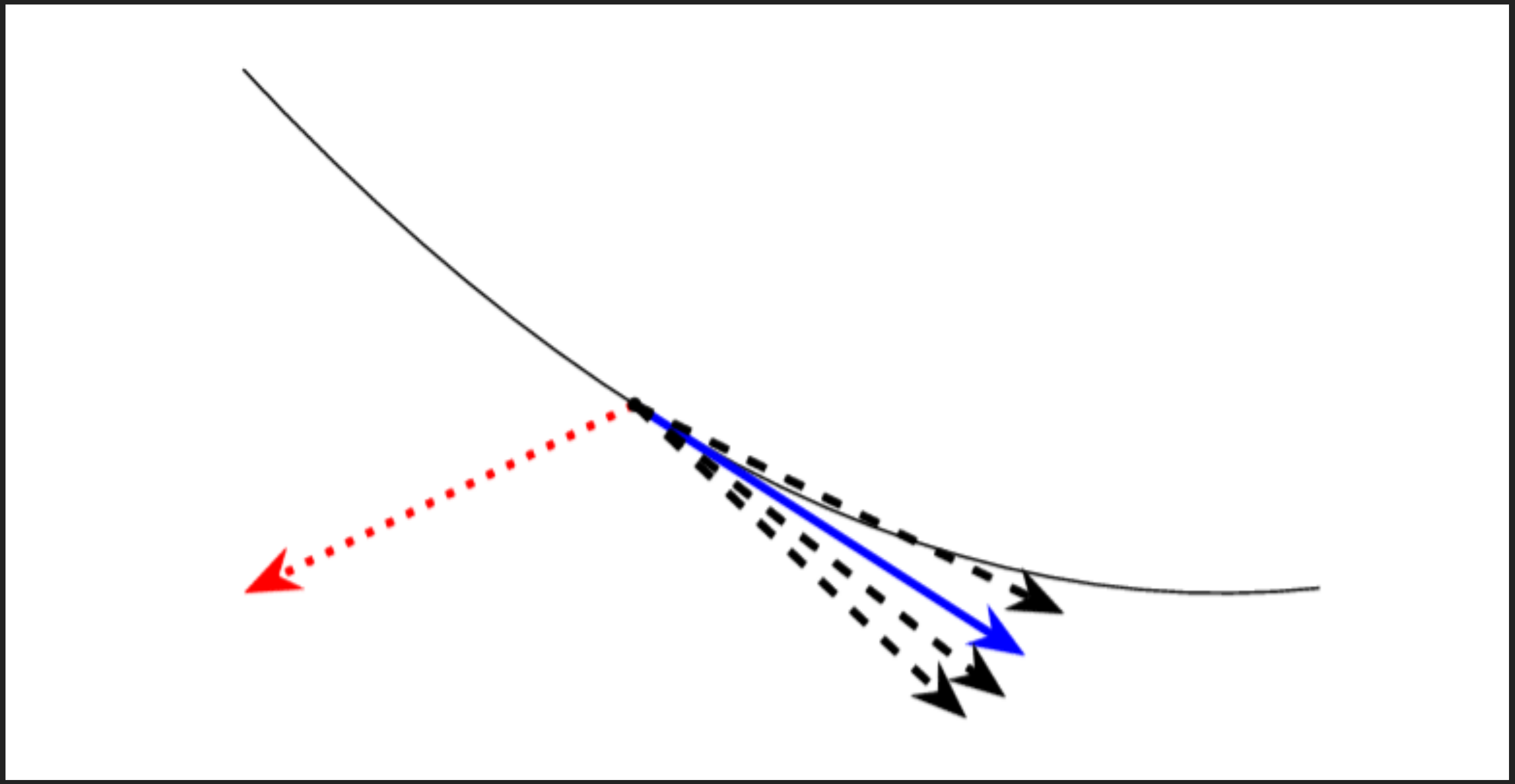
"we think of d as a constant, and note that $d \leq 3$ in many practical applications."

$\rightarrow$ true for mobile agents agreeing on a meeting point (problem in mind of the authors in Mendes and Herlihy?), not true for modern ML (d can reach up to 10^{10} and typically $d > n$ not the other way)

The (solution to) the Obvious Vulnerability of (distributed) Learning

*Stay in the "**correct cone**" !*

(i.e in the half space decreasing the cost function: the half-space requirement of Bottou 1998)



The (solution to) the Obvious Vulnerability of (distributed) Learning

*Stay in the "**correct cone**" → [(α, f)-Byzantine Resilience]:*

Let $V_1, \ldots, V_n$ be any i.i.d random vectors in $\mathbb{R}^d$, $V_i \sim G$, with $\mathbb{E}G = g$.

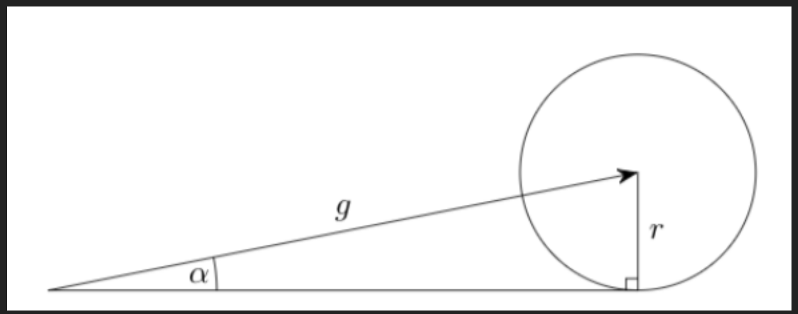
$B_1, \ldots, B_f$ any random vectors in $\mathbb{R}^d$, (possibly dependent on the V_i's).

Gradient Aggregation Rule F is said to be (α, f)-Byzantine resilient if, for any $1 \leq j_1 < \ldots < j_f \leq n$, the vector $F = F(V_1, \ldots, \underbrace{B_1}_{j_1}, \ldots, \underbrace{B_f}_{j_f}, \ldots, V_n)$ satisfies

(i) $\langle \mathbb{E}F, g \rangle \geq (1 - \sin\alpha) \cdot \|g\|^2 > 0$ and

(ii) for $r = 2,3,4,$ $\mathbb{E}\,\|F\|^r$ is bounded above by a linear combination of terms $\mathbb{E}\,\|G\|^{r_1} \ldots \mathbb{E}\,\|G\|^{r_{n-1}}$

with $r_1 + \ldots + r_{n-1} = r$.



The (solution to) the Obvious Vulnerability of (distributed) Learning

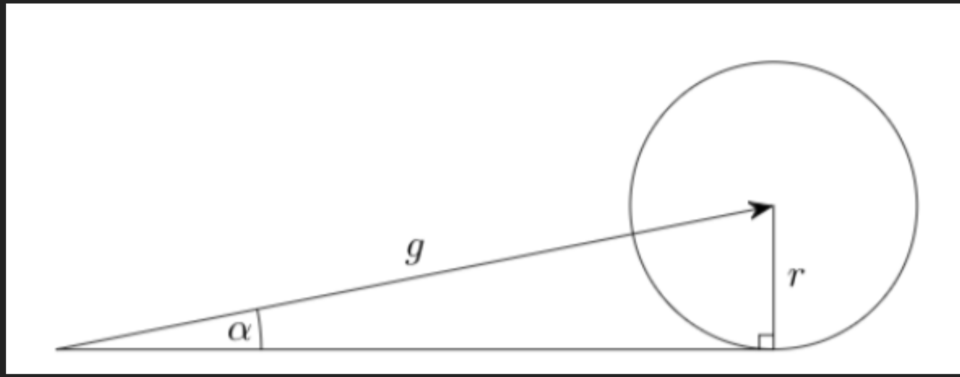
Krum ! (NeurIPS 2017)

$$(m)\ \underset{i\in\{1,\ldots,n\}}{\arg\min} \sum_{i\to j} \|\boldsymbol{G}_i - \boldsymbol{G}_j\|^2$$

$i \to j$: stands for "G_j is among the $n-f-2$ closest vectors to G_i" as long as $2f+2<n$ and $r = \eta(n,f)\sqrt{d}\cdot\sigma < \|g\|$,

then Krum is (α, f)-Byzantine resilient where $0 \le \alpha < \pi/2$ is defined by

$$\sin\alpha = \frac{\eta(n,f)\cdot\sqrt{d}\cdot\sigma}{\|g\|} \quad \text{and} \quad \eta(n,f) = \mathcal{O}(n \text{ or } \sqrt{n})$$

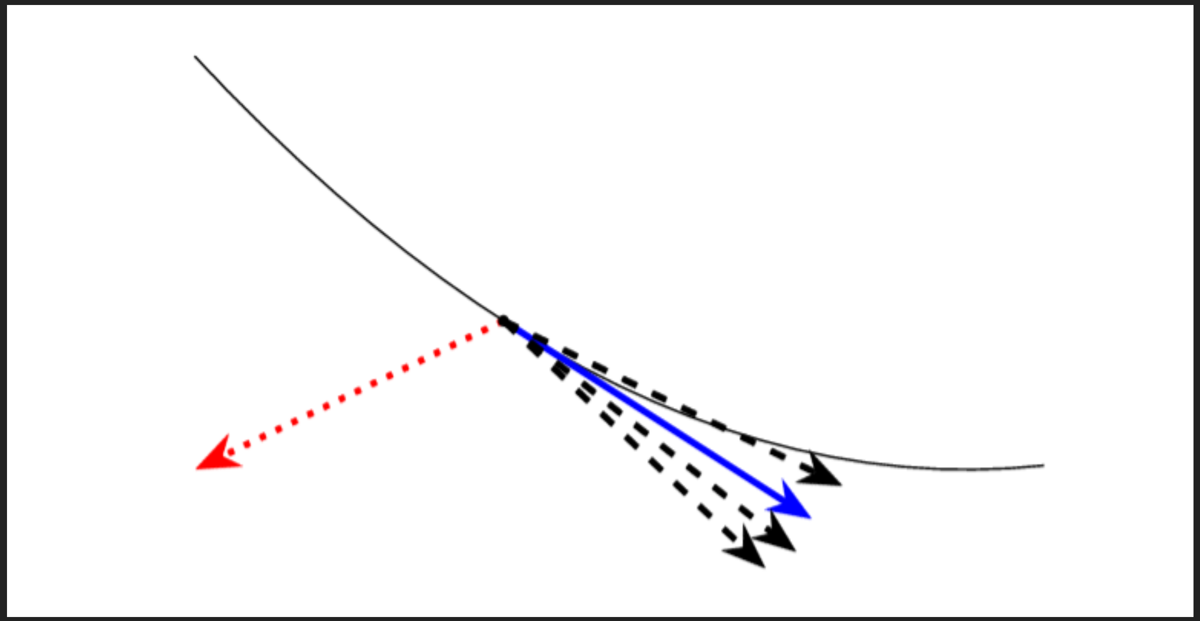


many other groups provided related solutions/improvements since then: Yin et al. ICML 2018, Chen et al. Sigmetrics 2018, Draco etc.

The **Hidden** Vulnerability of (distributed) Learning

The correct cone is ok, but poses a few problems in very high dimension (due to the condition $\eta(n, f)\sqrt{d} \cdot \sigma < \|g\|$*)*

- *In very high dimension, a cone is (extremely) wide !*
- *things get worse with highly non convex loss functions*

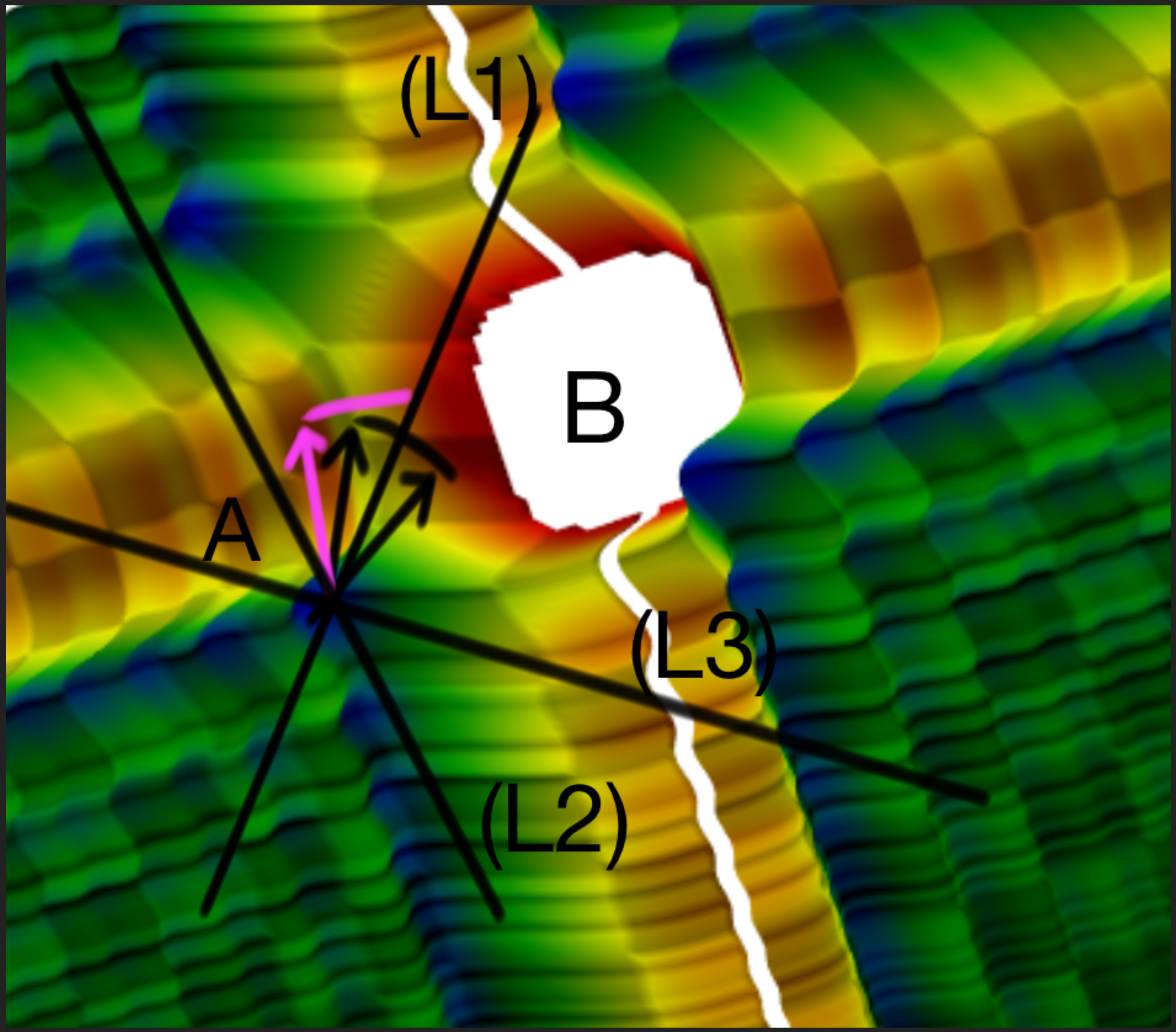


The **Hidden** Vulnerability of (distributed) Learning

The correct cone is ok, but poses a few problems in very high dimension ($\sqrt{d}$ dependance)

- *In very high dimension,*
  a cone is (extremely) wide !
- *things get worse with highly*
  non convex loss functions

Attack as simple as linear regression on correct gradients
(breaks the median also, pink vector is
In the cone)

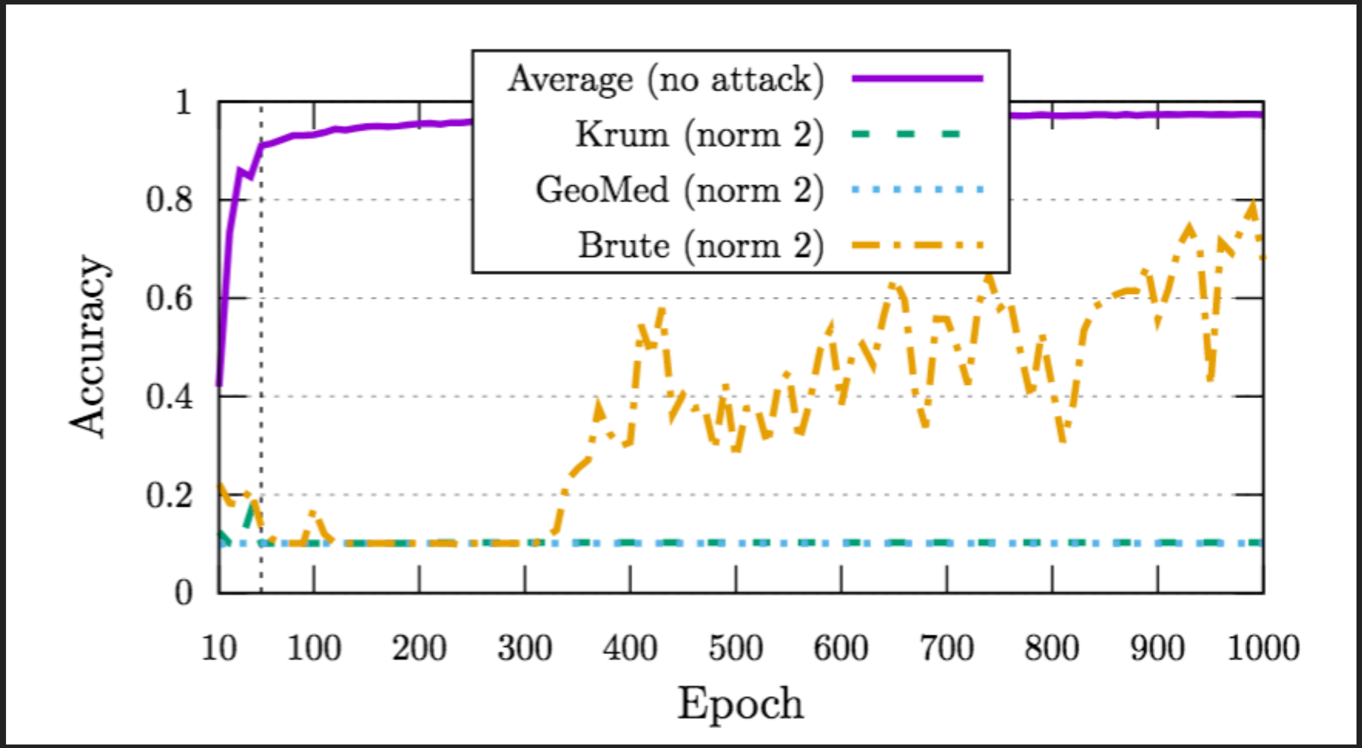


The **Hidden** Vulnerability of (distributed) Learning

The correct cone is ok, but poses a few problems in very high dimension (hinted by Krum: $\sqrt{d}$ dependance

- *In very high dimension, a cone is (extremely) wide !*
- *things get worse with highly non convex loss functions*

Attack as simple as linear regression on correct gradients (breaks the median also)



Attack on CIFAR-10 with a CNN

The (solution to the) **Hidden** Vulnerability of (distributed) Learning

Bulyan:

1) *Take any Byzantine aggregation rule in the "geometric median" family (e.g. Krum)*

2) *Iterate it 2f times to produce a "soup" of vectors in the correct cone* *(at iteration 2f, you still have "n"= n-2f > 2f , Bulyan requires n>4f)*

3) *Looks at the component-wise medians of that soup, produce an artificial vector with it*

4) *The attacker's leeway has been divided by* $\sqrt{d}$ *!*

remarks

1) *If **f** is way smaller than n/4 → more workers involved per epoch → less variance*

2) *Higher batch-size→ less variance → more robustness (narrower correct cone)*

aSynchronous SGD

- Real time recommendations
- Stale workers (Federated learning, on-device ML , low band-with in some areas)

Asynchrony is (extremely) hard in distributed computing

Agreement is impossible in asynchrony with one single crashed worker (not even Byzantine).

Fischer Lynch Paterson (1983)

Asynchrony is (extremely) hard in distributed computing

Good news:

We know what we should agree on (the gradient of a cost function)

→ we can exploit its mathematical regularities

Setting (asynchrony)

n : workers

f : Byzantine

Server

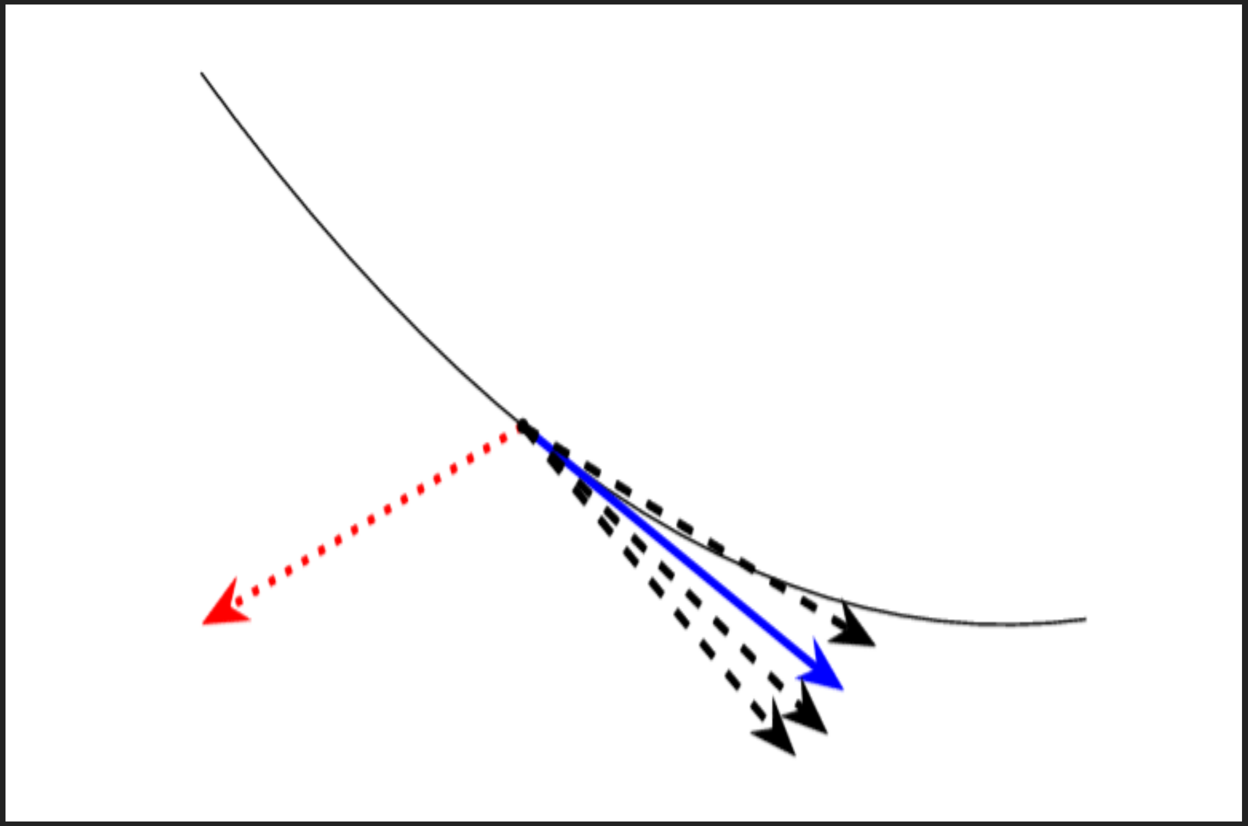
Does not wait (no aggregation)

Threat model: omniscient ≠ omnipotent

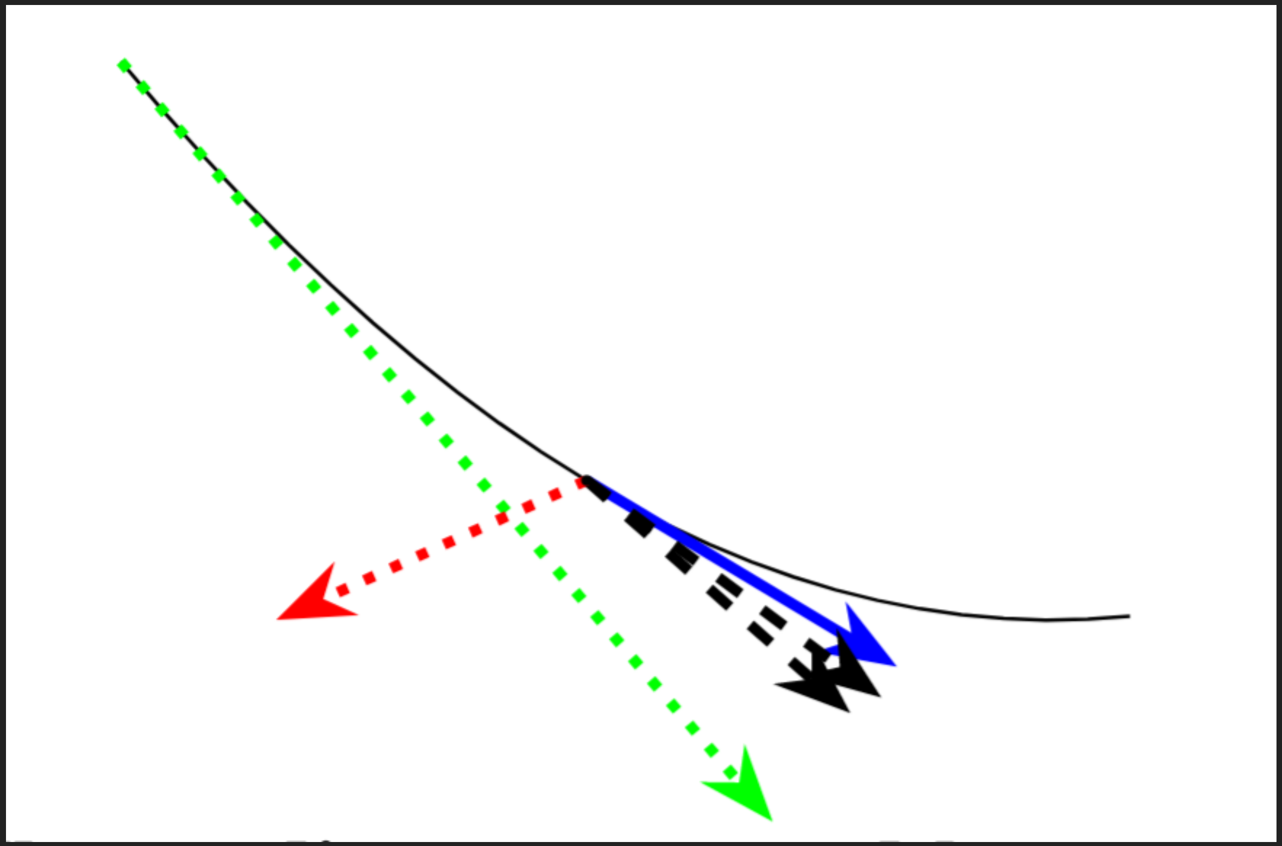
$Worker_1$ $Worker_2$ $Worker_3$ $Worker_4$

Data

What changed from Synchronous Settings ?



Synchronous

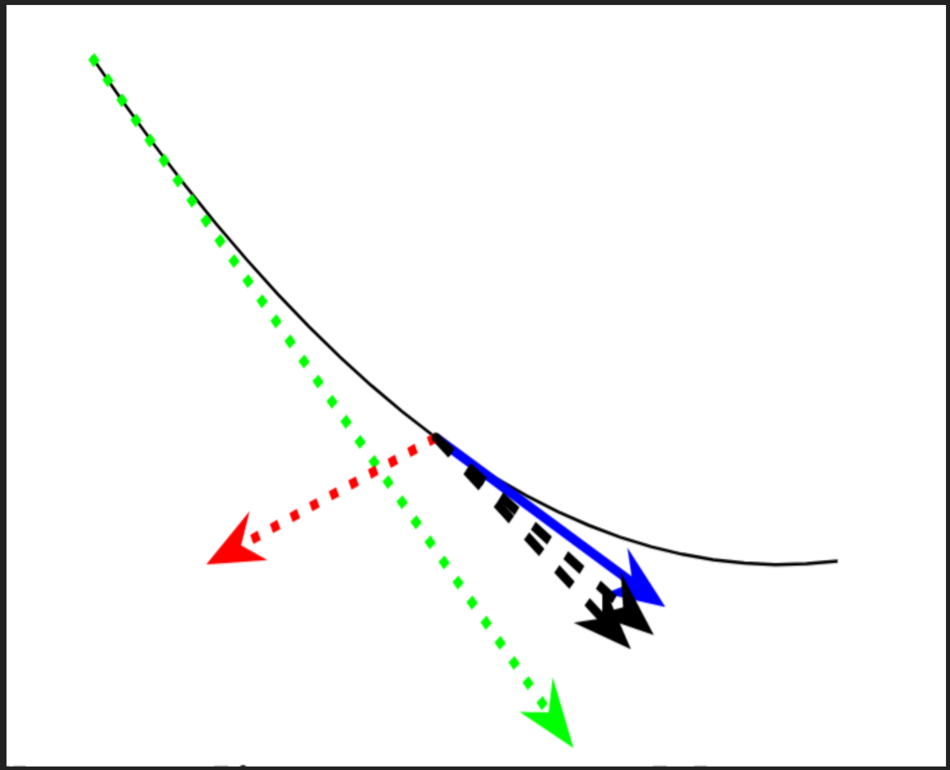


Asynchronous

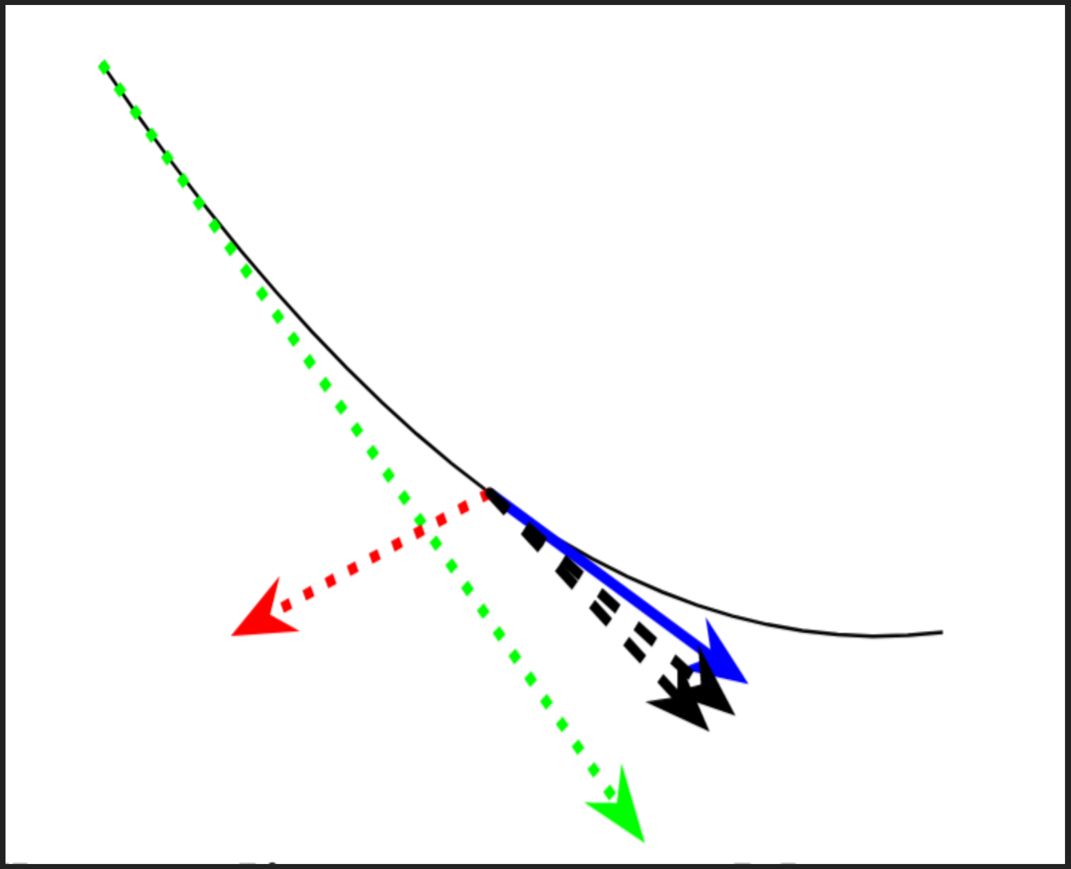
What changed from Synchronous Settings ?

Bad news: a new impossibility result

Theorem (informal): no Asynchronous SGD algorithm tolerates a **single** Byzantine worker, if both infinite participation and unbounded delays are allowed.



What changed from Synchronous Settings ?



Good news:

Theorem: (a minimalistic relaxation is) unbounded delays with finite successive participation via a filtering scheme

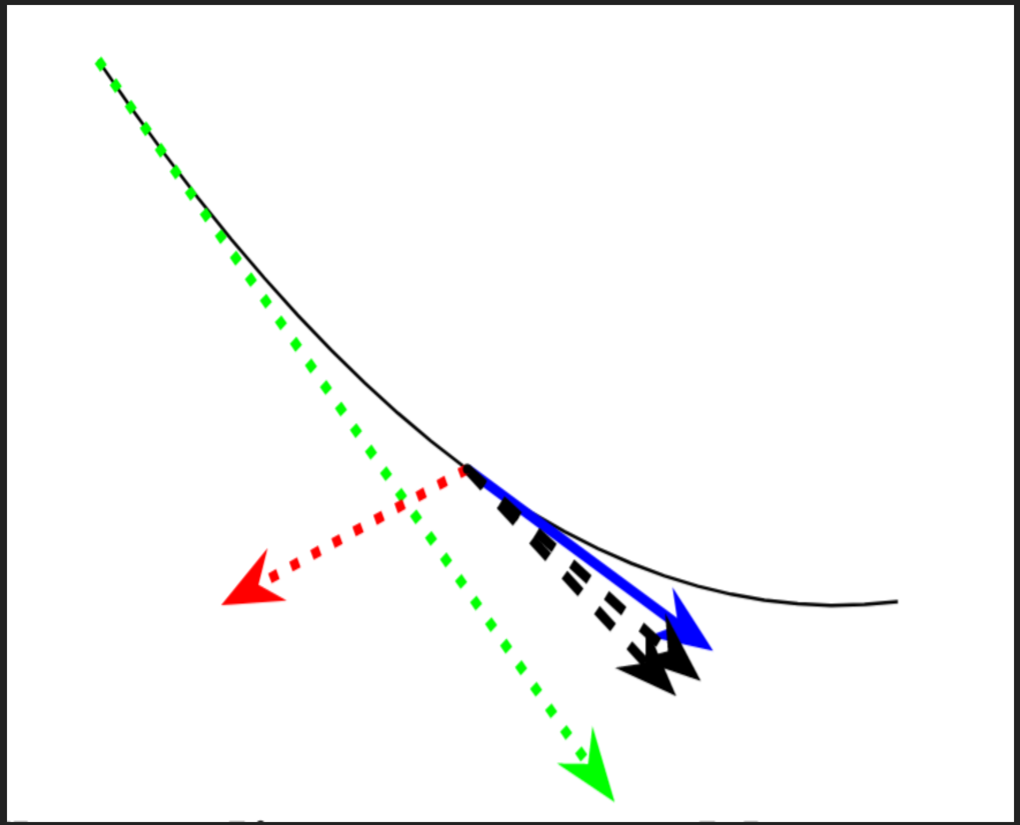
(result is relevant to asynchrony beyond Byzantine faults)

Remember "The (solution to) the Obvious Vulnerability of (distributed) Learning" ?

Stay in the "correct cone" !

(i.e in the half space decreasing the cost function)

The correct cone idea will be crucial (again), **but without synchronous aggregation !**



Solution: Kardam

1) Kardam relies on an online filtering scheme (no aggregation or waiting):

   workers are judged by

   (a) their gradients' "empirical Lipschitzness", i.e. the growth rate of their gradient relative to their model change,

   (b) their "talkativeness"

2) Kardam uses a dampening scheme on stale-gradients, (practicality: scale down correct but stale gradients)

Kardam: the filtering scheme (Lipschitz)

1) Worker p sends gradient and declares Kp, its empirical gradient rate of growth

   Kp (declared by worker, considered by server to estimate quantiles): $\frac{\|\boldsymbol{g}_p - \boldsymbol{g}_p^{prev}\|}{\|\boldsymbol{x}_{l_p} - \boldsymbol{x}_{l_p}^{prev}\|}$

2) Server does not trust the worker and assigns $\hat{K}_t^p$ to worker p, where

   $$\hat{K}_t^p = \frac{\|\boldsymbol{g}_p - \boldsymbol{g}_q\|}{\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|}$$

   (considered by server to filter)

3) Kp are still useful (since a majority will not cheating) to evaluate the quantiles of declared Kp.

4) Workers whose $\hat{K}_t^p$ falls into the n-f/n quantile are trusted, others are ignored (potential loss of work).

Kardam: Byzantine resilience

Passing the filter guarantees:

$$\langle \mathbb{E}_\xi \boldsymbol{G}(\boldsymbol{x};\xi), \boldsymbol{\nabla} Q(\boldsymbol{x})\rangle > \Omega\left((\|\boldsymbol{\nabla} Q(x_t)\| - \sqrt{d}\sigma)\|\boldsymbol{\nabla} Q(x_t)\|\right)$$

As long as gradient estimators make sense, i.e

- As long as gradients are large
- As long as variance is small (batch-size is high enough)

Kardam makes provable progress (cone is smaller than π/2)

remarks

- Computing the **empirical** growth rate of gradients, is still a very naive way of estimating the curvature

- Frequency filter probably still too restrictive (no optimality was proven on number of participations)

Summary

This presentation:

- Formulating the Byzantine SGD question
- Initial solution in the synchronous case

  NeurIPS 2017a

- Strengthening synchronous solutions for high dimension / non convexity

  ICML 2018a

- Initial (and so far only) solution to the asynchronous case

  ICML 2018b

Remaining challenges:

A better understanding of the optimisation tool-box,
Leverage recent results in neural nets theory to reduce model-dependant / data-dependent unknowns,
Server-side, decentralised SGD without SMR…

Summary

This presentation:

- Formulating the Byzantine SGD question
- Initial solution in the synchronous case — NeurIPS 2017a

- Strengthening synchronous solutions for high dimension / non convexity — ICML 2018a

- Initial (and so far only) solution to the asynchronous case — ICML 2018b

During this PhD:

- Robustness of Neural Networks as Distributed System — IPDPS 2017, SRDS 2017
- Robustness in Systems Biology — BDA 2017, Biorxiv 487348
- Gathering behaviours with reinforcement learning — Bulletin of the EATCS 2019
- Testing the Byzantine SGD algorithms on a systems level — SysML 2019
- Safety in reinforcement learning — NeurIPS 2017b, arXiv:1805.11447

Ongoing:

- Byzantine servers without SMR
- More on the robustness of Neural Nets
- Asynchrony and curvature…

is a little *really* enough?

- Gradient norm is small close to convergence (early stopping is enough)
- $(average + 1.5\sigma)_{coordinate-wise}$ proposed f times
- We tested it: Bulyan never prevented from convergence, same accuracy as without the proposed attack
- out of topic: black box (choose a loss, propose a gradient from it)

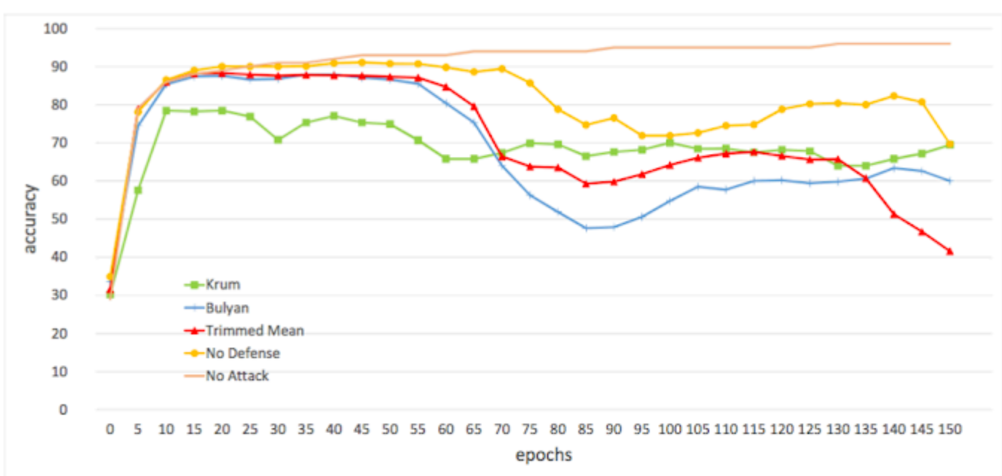
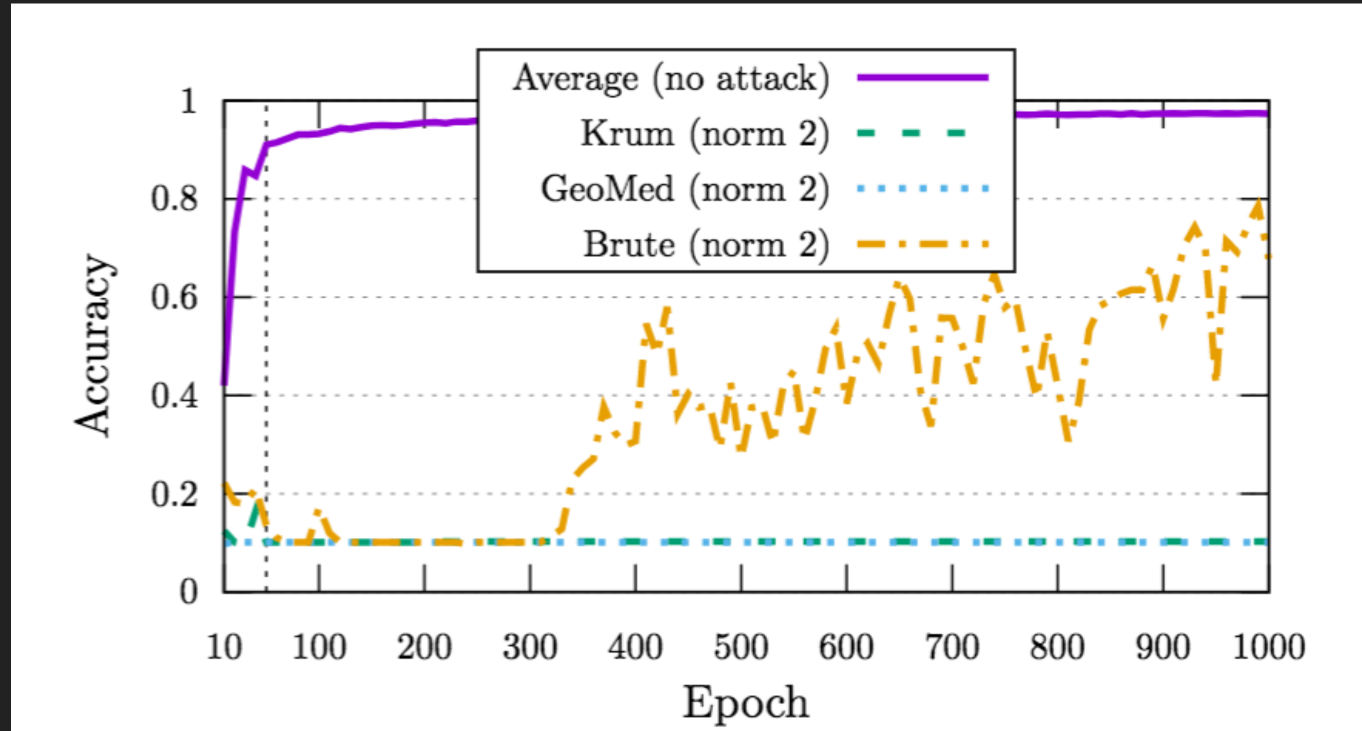


Figure 2: Model accuracy on MNIST. $m = 24\%$ and $z = 1.5$. *No Attack* is plotted for reference.

(a little is enough)



Bulyan (our attack) for comparison

server-side

- Ongoing work with (1) SR and (2) AG
  - 1: n workers (f Byzantine) and N servers (F Byzantine) on asynchronous network problem is drift of honest parameters, tackled with an anti-drift mechanism (a median on $N - F$ parameters that a received at a server).
    - Issue: still strong hypothesis on alignment of parameters on correct workers
    - but: verified empirically (but no reason it applies to all models and all datasets)
  - 2: Kardam-like filter on models, synchronous training, Krum on workers, norm growth filter instead of growth rate filter

convex, model-dependence, etc

- Hypothesis hide a few model/data-dependent variables
- need to go down to more convex settings for exact hypothesis (ours are indicative (e.g. how much exactly the batch-size))

Kardam: the filtering scheme (frequency)

Specification:

The frequency filter ensures that any sequence of length ~ 2f consequently accepted gradients contains at least ~ f gradients computed by honest workers.

How:

$$\sum_{p \in \theta} n_p \leq f$$

L: list of ids of the workers that contributed the last 2f gradients,
accept p (if passed the Lipschitz filter) as long as

(we can do better)

Kardam: the staleness-aware component

Update takes "declared" staleness of the worker into account and s $\Lambda(\tau_{tl})$ down with

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \gamma_t \boldsymbol{Kar}_t \\ &= \boldsymbol{x}_t - \gamma_t \sum_{[\boldsymbol{G}(\boldsymbol{x}_l;\xi_m),l] \in \mathcal{G}_t} \Lambda(\tau_{tl}) \cdot \boldsymbol{G}(\boldsymbol{x}_l;\xi_m) \end{aligned}$$

$$|\mathcal{G}_t| = M$$

With M=1 in "pure" asynchrony (one gradient per update)

Generic SAware scheme, interesting in its own right - makes Kardam work practically, outperforms alternatives (that are actually subcases of it)

Kardam: the staleness-aware scheme

Update takes “declared” staleness of the worker into account and s $\Lambda(\tau_{tl})$ down with

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \gamma_t \boldsymbol{Kar}_t \\ &= \boldsymbol{x}_t - \gamma_t \sum_{[\boldsymbol{G}(\boldsymbol{x}_l;\xi_m),l]\in\mathcal{G}_t} \Lambda(\tau_{tl}) \cdot \boldsymbol{G}(\boldsymbol{x}_l;\xi_m)\end{aligned}$$

$$\gamma_t = \underbrace{\sqrt{\frac{Q(\boldsymbol{x}_1) - Q(\boldsymbol{x}^*)}{KTMd\sigma^2}}}_{\gamma} \cdot \underbrace{\frac{M}{\sum_{\lambda\in\boldsymbol{\Lambda}_t} \lambda|\mathcal{G}_{t\lambda}|}}_{\mu_t}$$

Adaptive learning rate.
γ: baseline learning rate
μ_t: incorporate total staleness at epoch t

Kardam: convergence guarantee

Replacing unbiasedness by our correct cone alternative, the ergodic (Zhang et al. 2015, Jiang et al. 2017) convergence proof guarantees that:

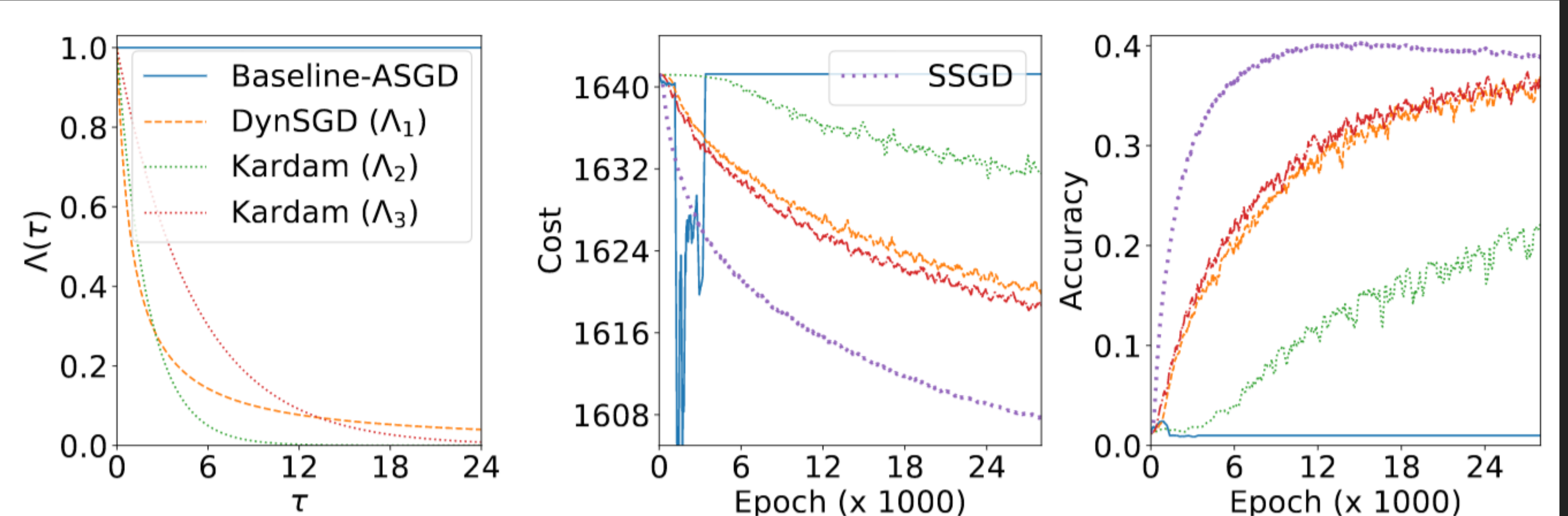
$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla Q(\boldsymbol{x}_t)\|^2 \leq (2+\mu_{\max}+\gamma K M\chi\mu_{\max})\cdot\gamma K\cdot d\sigma^2+d\cdot\sigma^2+2DK\sigma\sqrt{d}+K^2D^2$$

D: the global confinement of the cost function
K: the global lipschitz bound
And parameters of the adaptive learning rate (details in the paper)

Experiments

Same remark as first talk: no Byzantine resilience is proven with experiments, only vulnerability can be proven with an attack

However… Kardam has some merits *besides* Byzantine resilience since “pure” asynchrony ~ Byzantine
(+ knowledge of f$\longrightarrow$ less lost gradients)



(a) Dampening functions (b) Cost (c) Accuracy