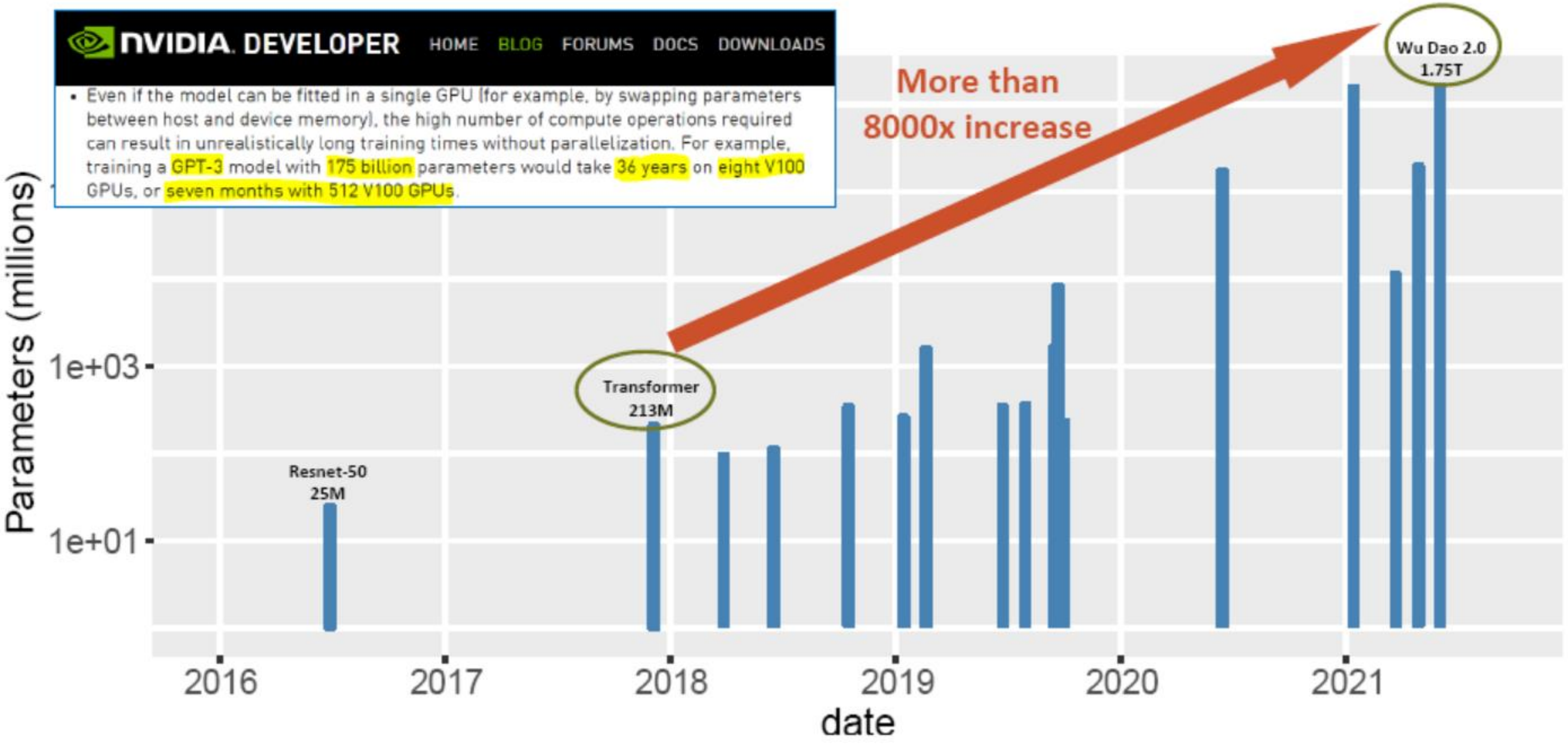
Near-Optimal Sparse Allreduce for Distributed Deep Learning

Shigang Li, Torsten Hoefler
ETH Zurich

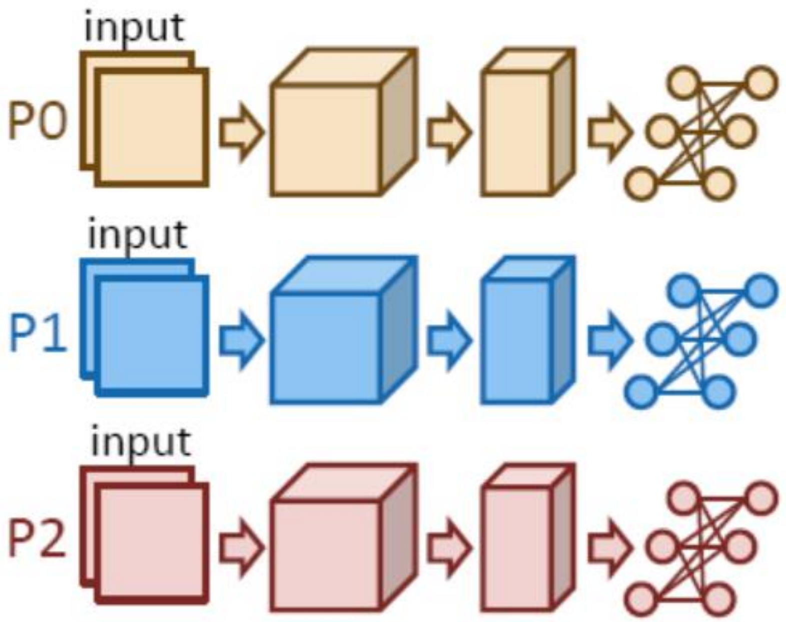
PPoPP'22
April 2-6, 2022
Seoul, Republic of Korea

Model size growing exponentially



Parallel and distributed training

Data parallelism



Pros:

a. Easy to realize

Cons:

a. Using DP alone may NOT work for large models, but works with others **(OP, PP)**

b. High allreduce overhead

This work (***Ok-Topk***) aims to solve

Operator parallelism

Pros:

a. Make large model training feasible

Cons:

b. Communication for each operator (or each layer)

Pipeline parallelism

Pros:

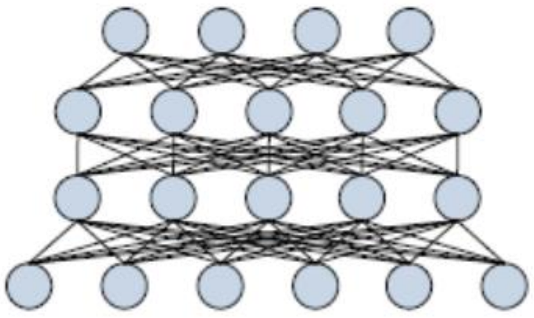
a. Make large model training feasible
b. No collective, only P2P

Cons:

a. Bubbles in pipeline
b. Removing bubbles leads to stale weights

We have **Chimera** (SC'21) to solve the above issues for pipeline parallelism.

Data parallelism



$$f(w) = \mathbb{E}_{\xi \sim D}\, F(w;\xi)$$

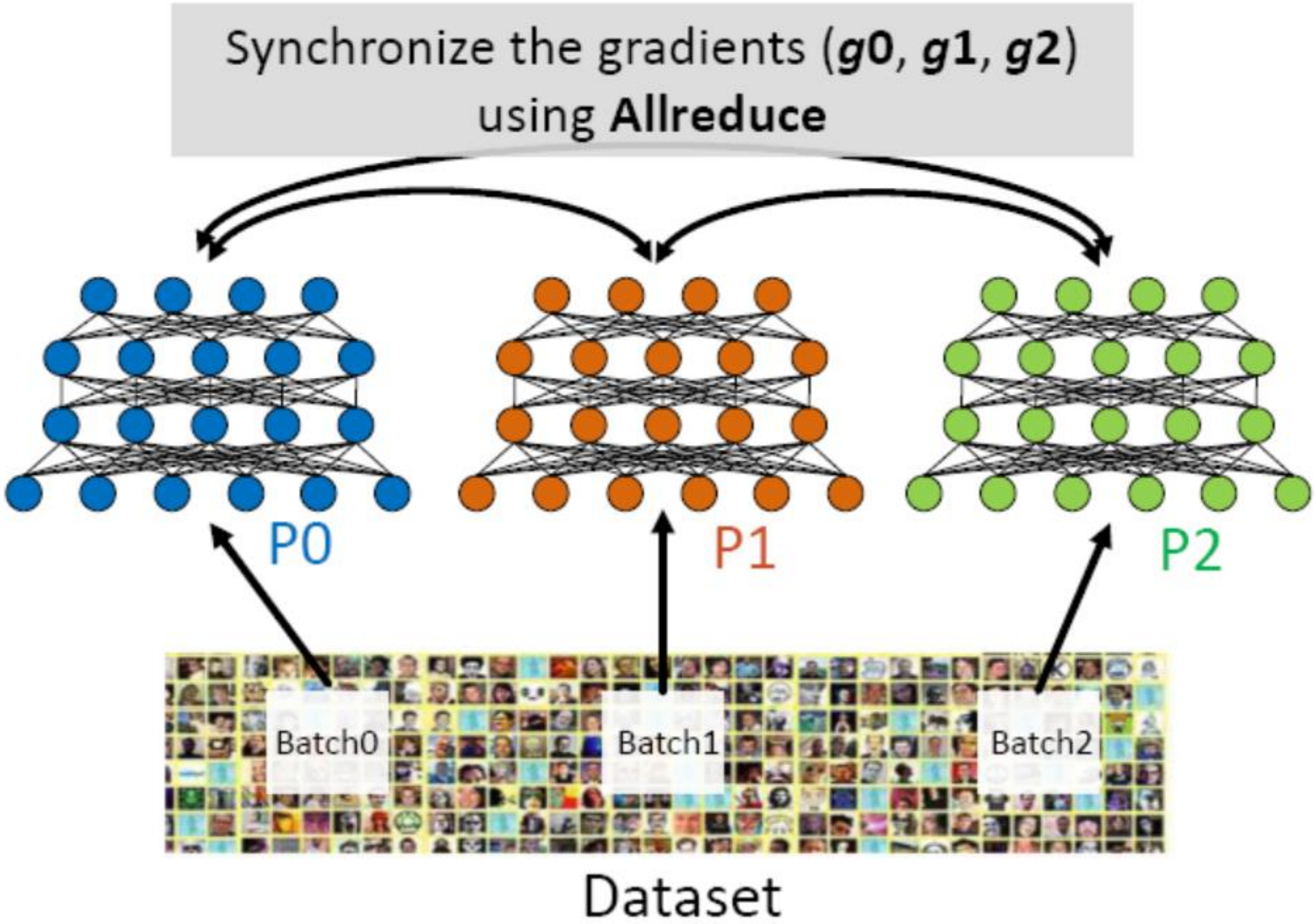
w denotes the model parameters.

F is the loss function.

ξ is a data batch sampled from a distribution ***D***.

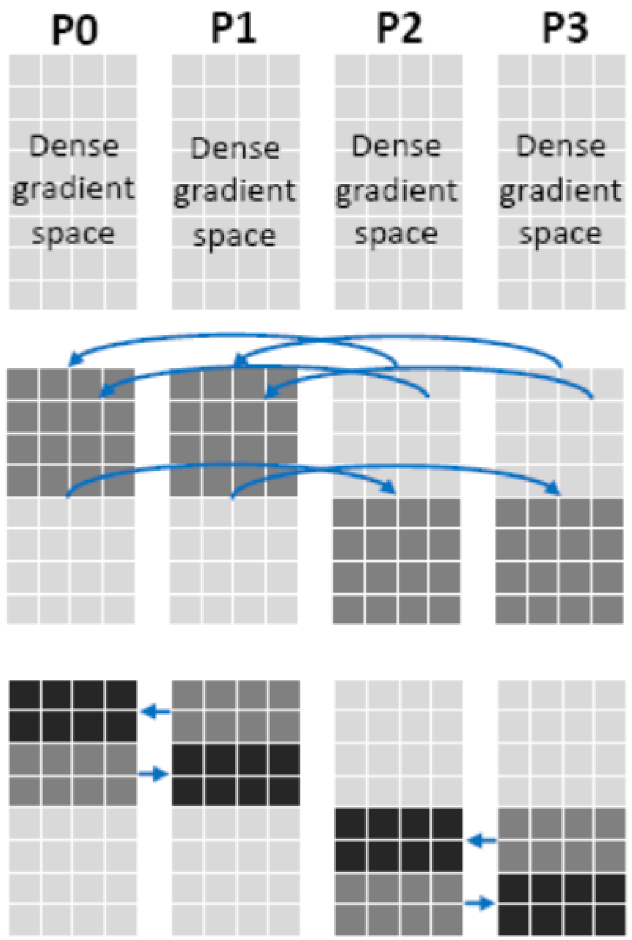
Training: update ***w*** to minimize ***f*** (e.g., SGD).

$$g_t = \frac{1}{b}\sum_{i=0}^{b} \nabla F(w_t, \xi_i) \qquad w_{t+1} = w_t - \eta_t g_t$$

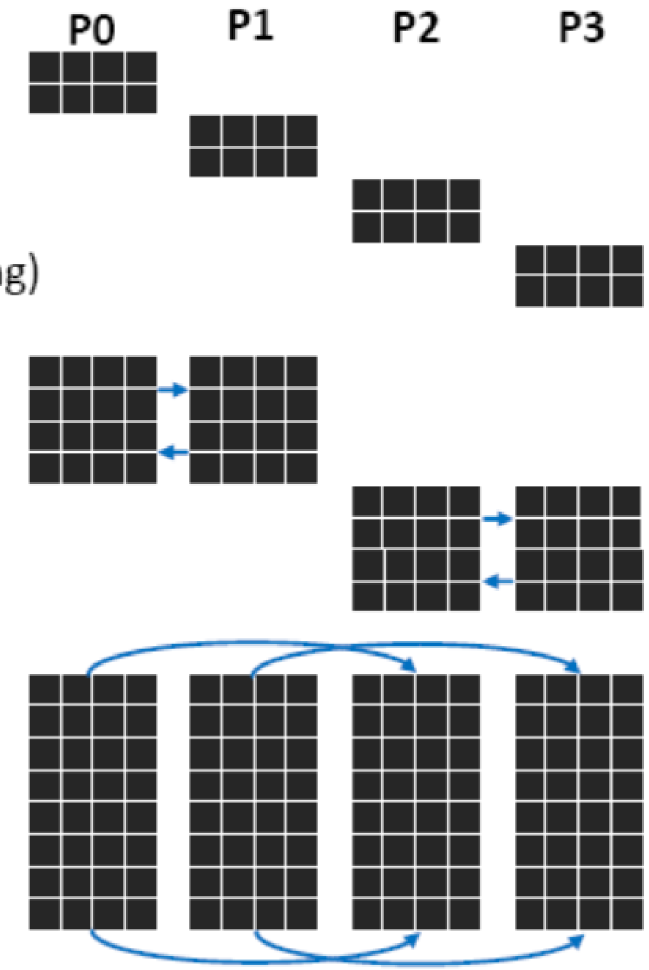


Revisit Dense Allreduce (Rabenseifner's algorithm)

Reduce-Scatter
(Recursive halving)



Allgather
(Recursive-doubling)



Latency-bandwidth model:

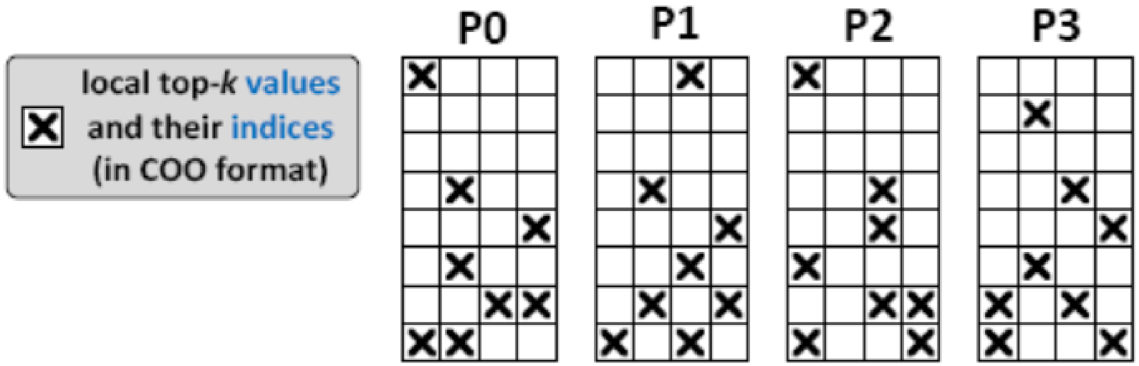
α is the latency; β is the transfer time per word;
p is the number of processes; n is the message size.

Comm. Cost $= 2(\lg p)\alpha + 2((p-1)/p)n\beta$

Bandwidth optimal and scalable, but w.r.t. n

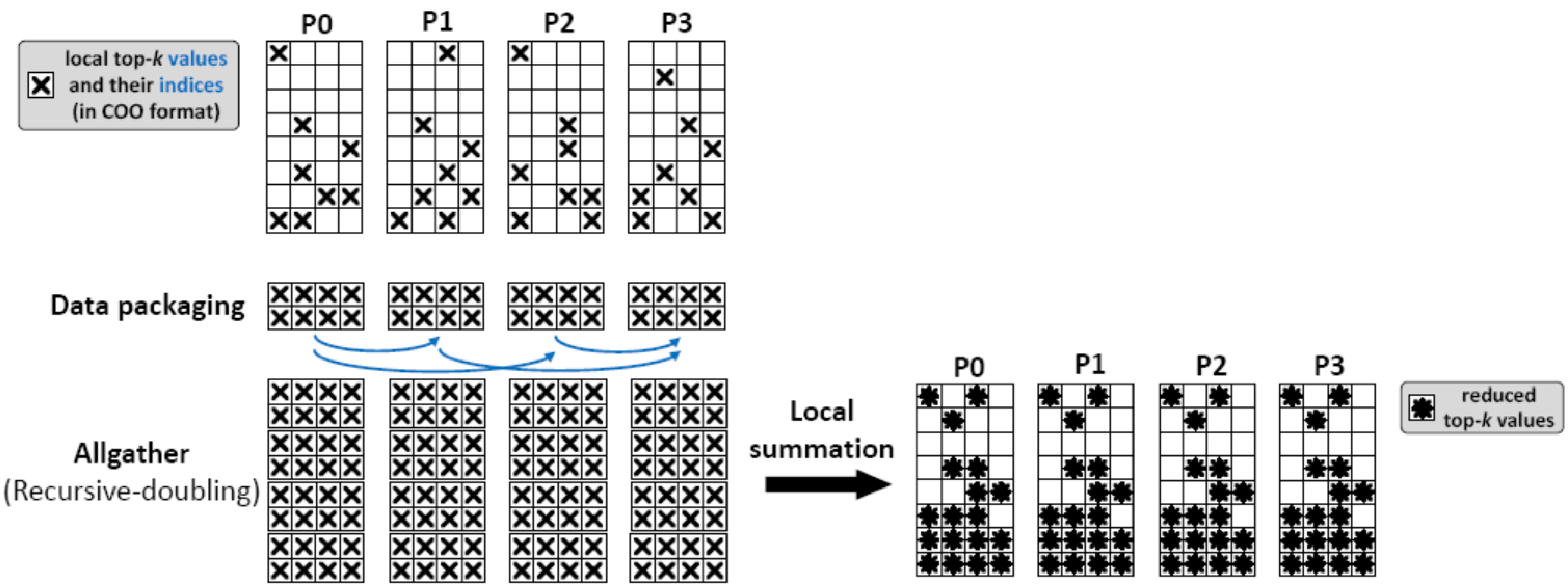
Gradient sparsification (Top$k$ SGD)

Top$k$ SGD: each process only selects **the largest** (absolute value) k of n components from the gradients, and usually the **density k/n is around 1% or less**.



How to **Allreduce** these **sparse** gradients?

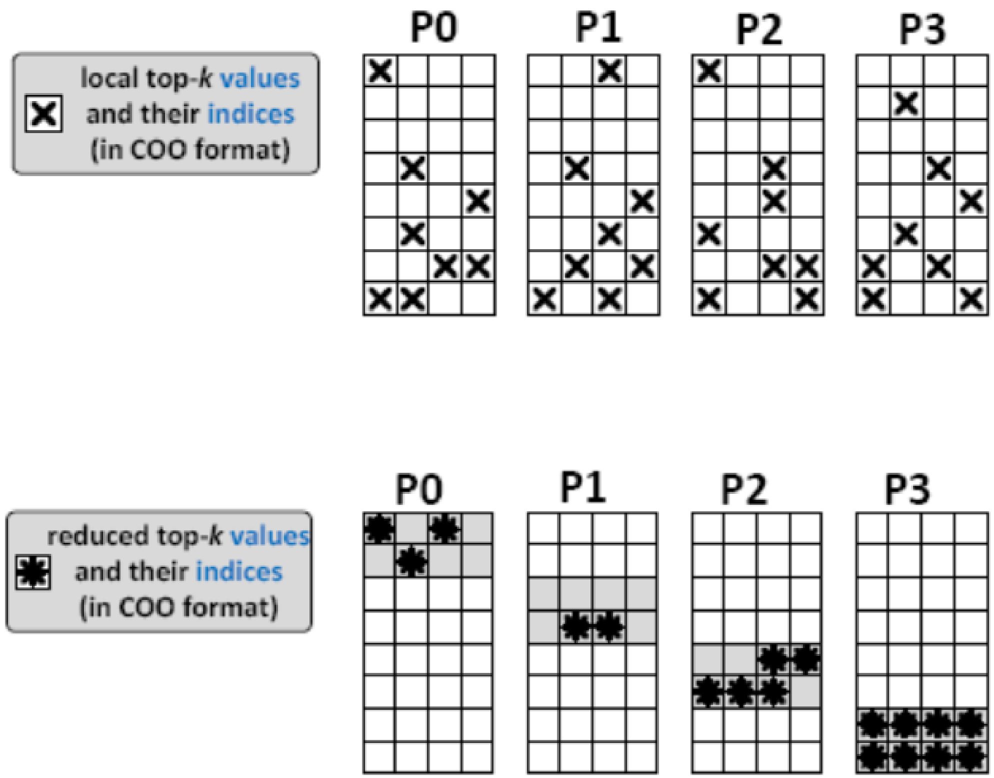
Algorithm 1: **Sparse Allreduce** based on **Allgather** (**Top*k*A**)

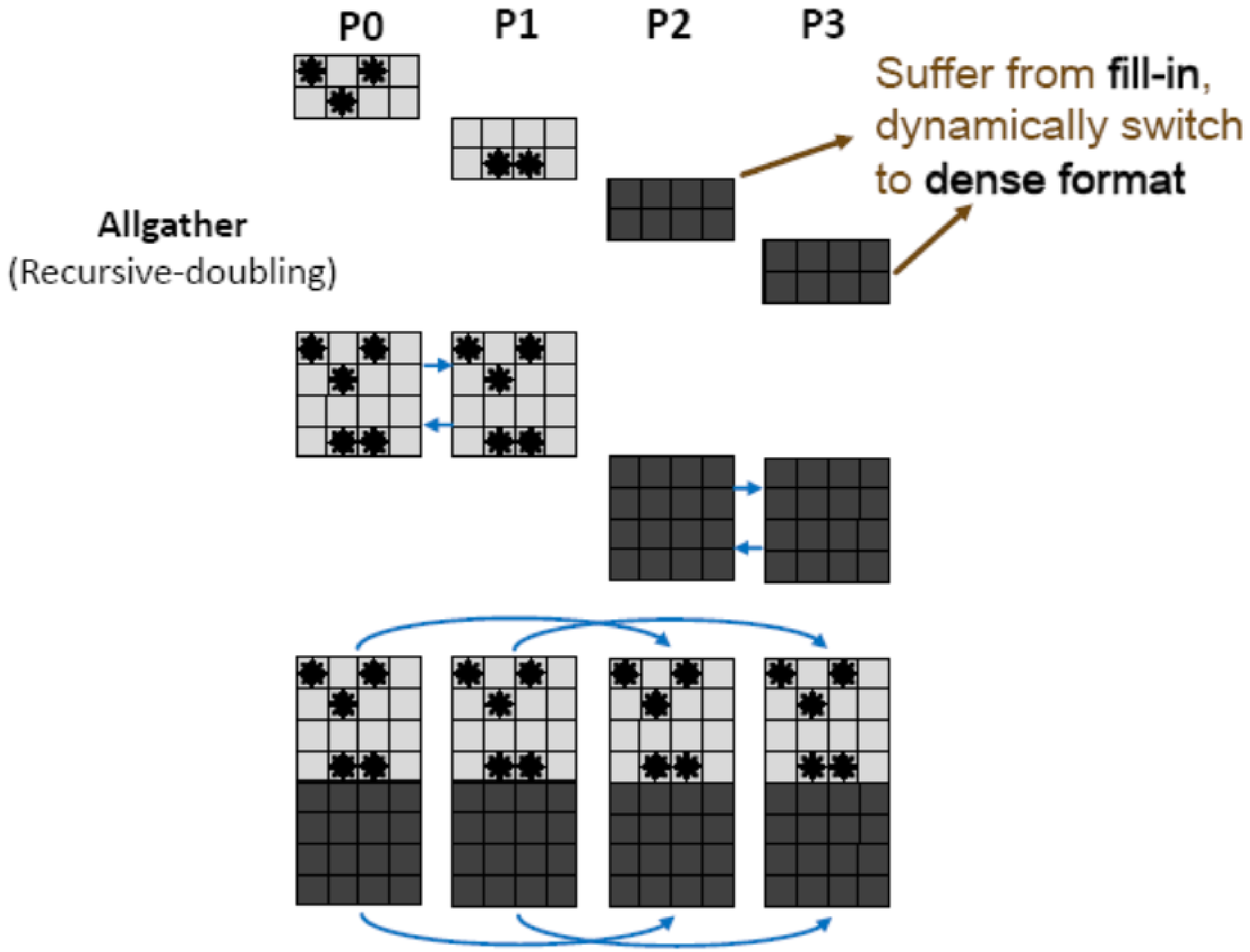


Comm. Cost = $(\lg p)\alpha + 2k(p-1)\beta$ **Not scalable!**

Algorithm 2: Dynamic Sparse Allreduce (**Top*k*DSA, SC'19**)

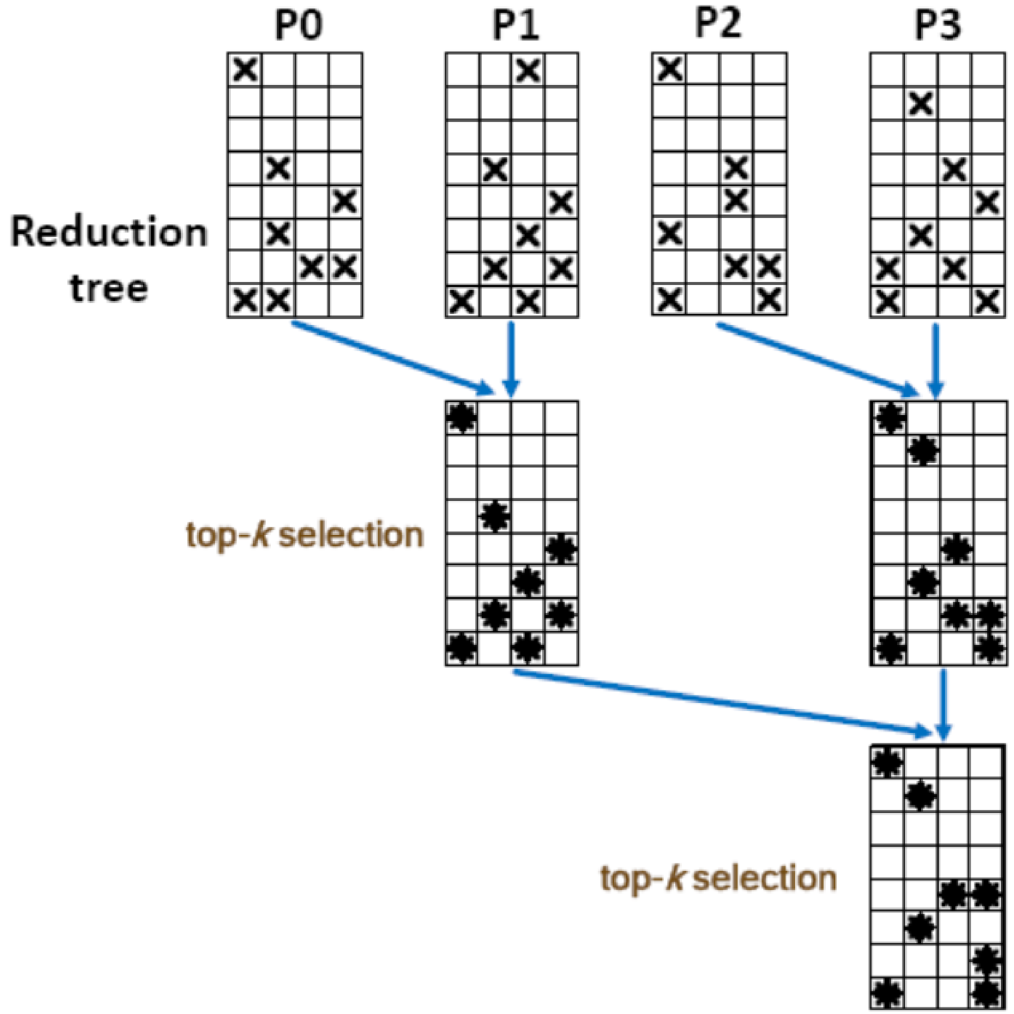
Inspired by dense Rabenseifner's algorithm



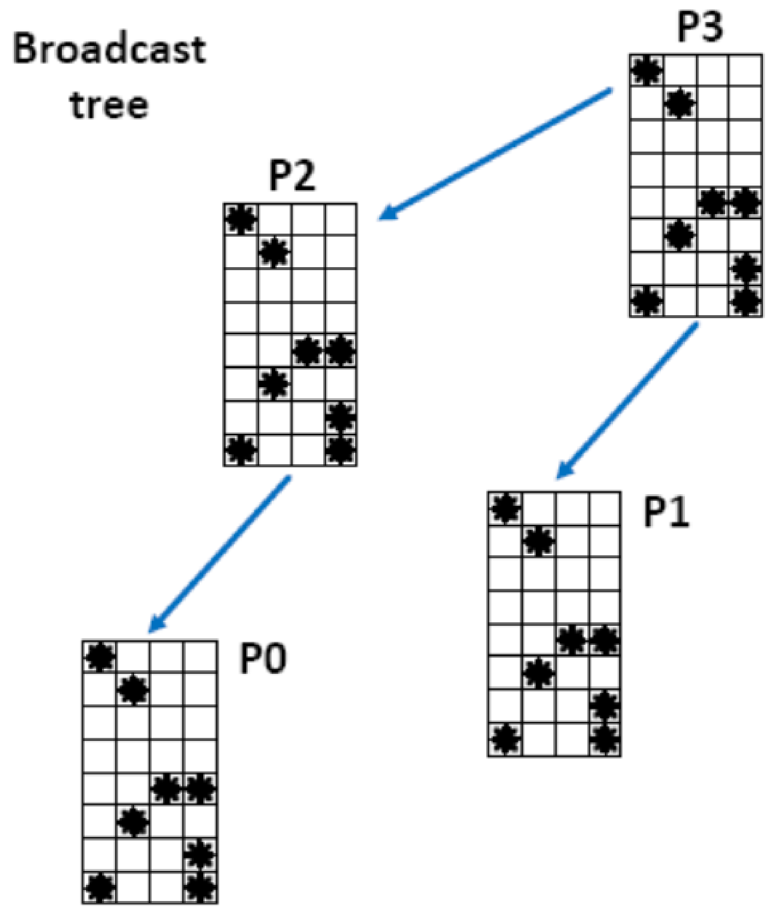


Comm. Cost = $(p + 2\lg p)\alpha + [(4k(p-1)/p)\,\beta,\ ((2k+n)(p-1)/p)\,\beta]$ Suffer from **fill-in issue** when scaling up!

Algorithm 3: Global Top*k* (**gTop*k*, ICDCS'19**)

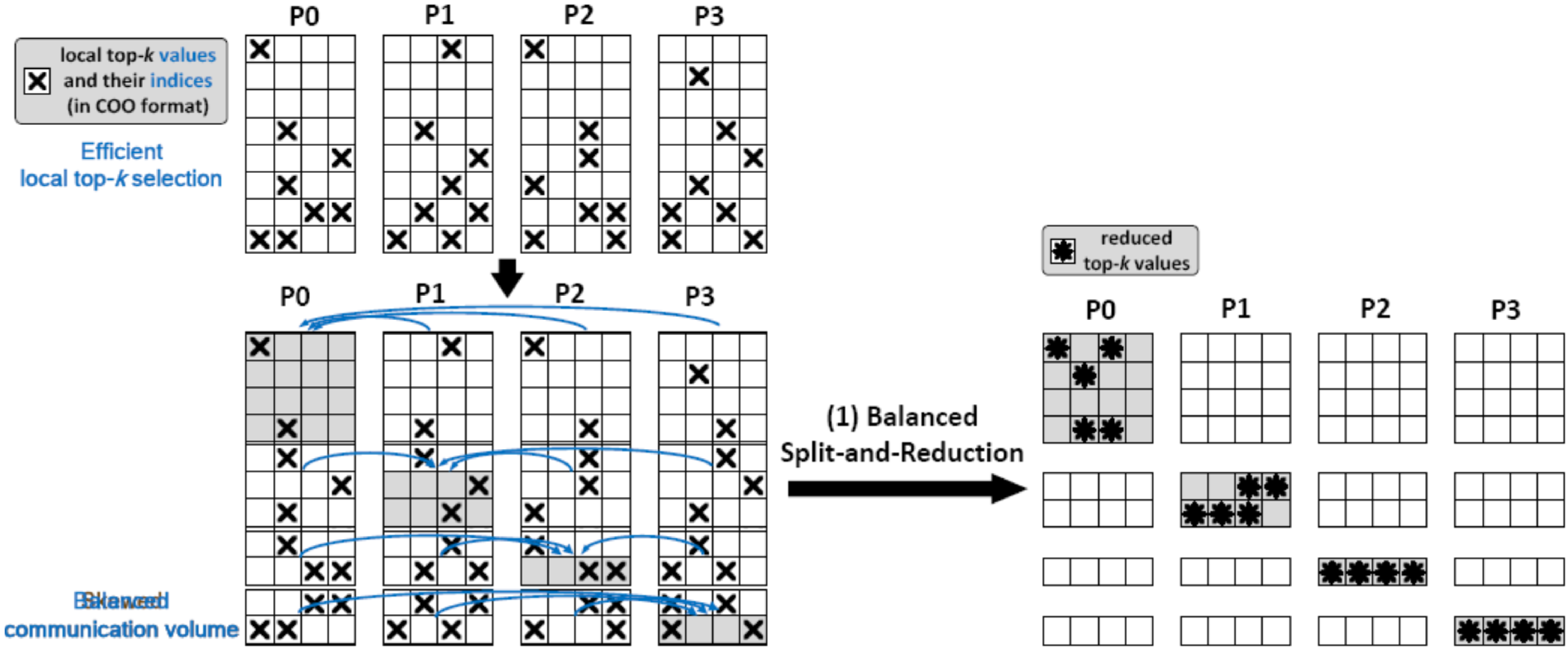


Comm. Cost = $(2\lg p)\alpha + 4k(\log p)\beta$



Solve the fill-in issue, but
(1) high top-*k* selection cost, and
(2) suboptimal bandwidth cost.

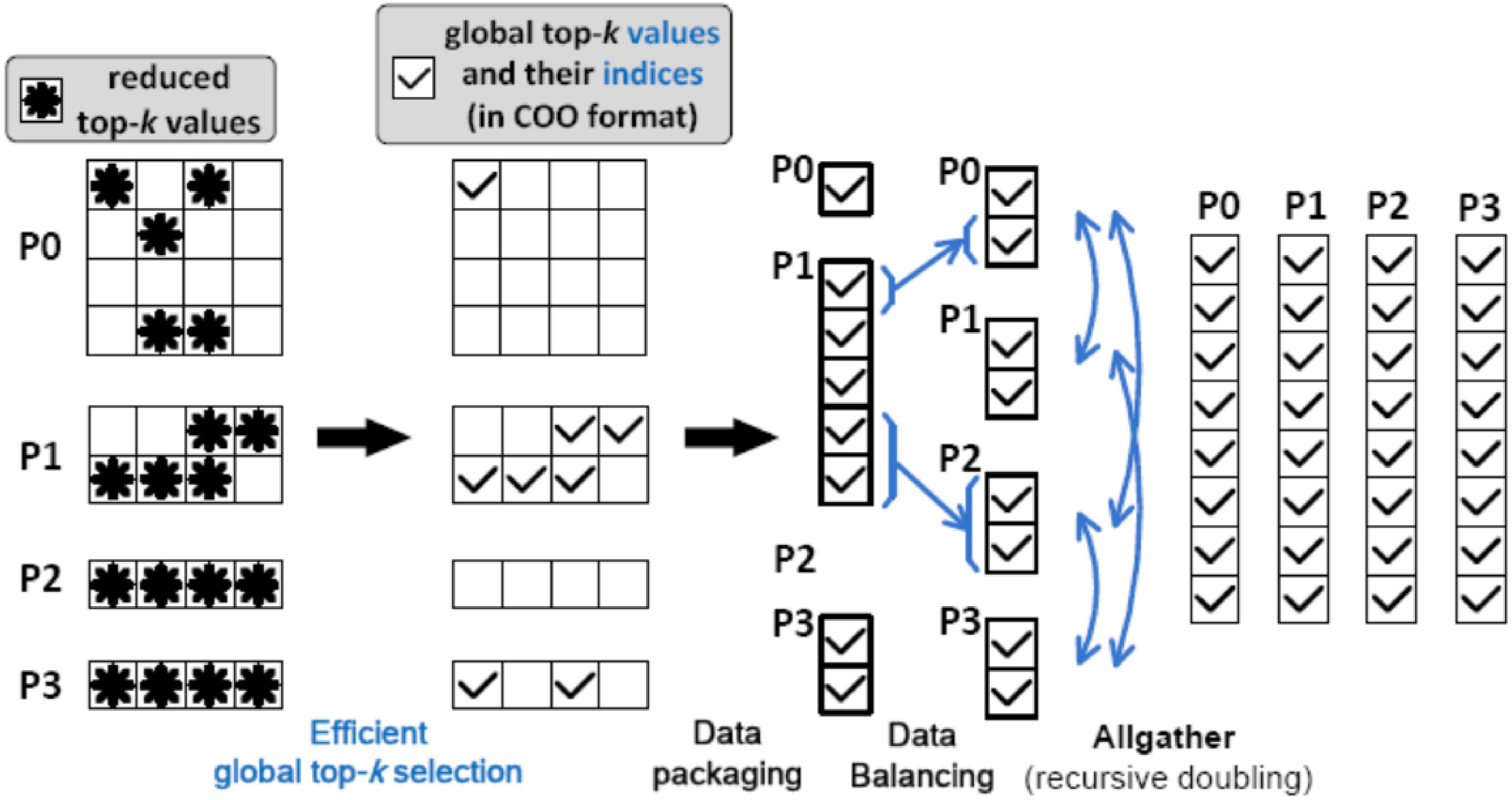
O(*k*) sparse Allreduce



The cost of balanced *split_and_reduce* = $(P-1)\alpha + 2k\,((P-1)/P)\,\beta$

O(k) sparse Allreduce

(2) Global top-k selection & balanced Allgather

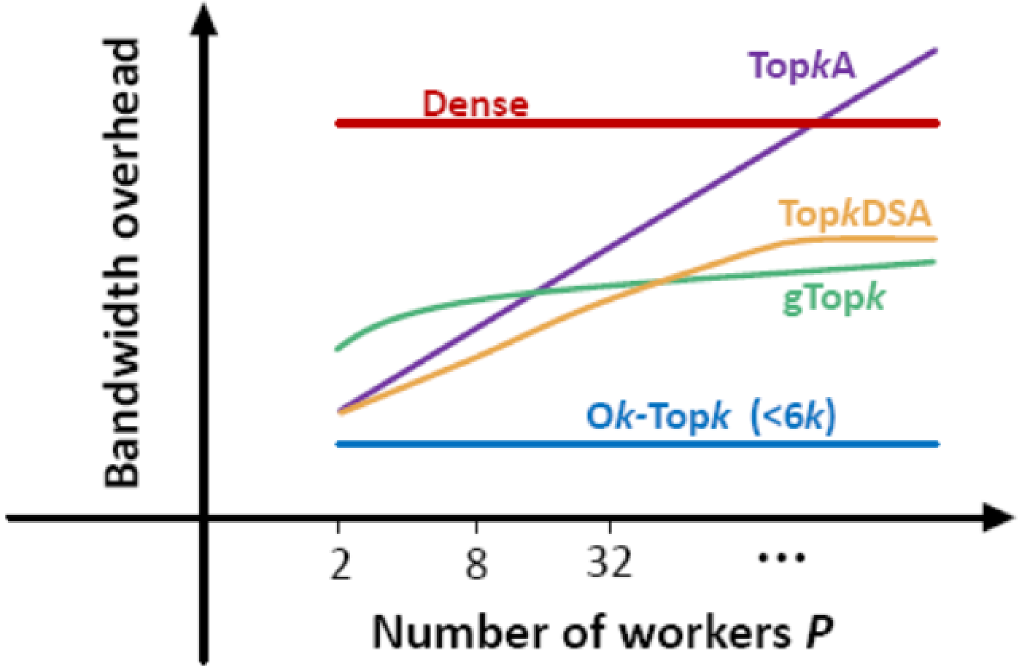


Total Cost <= $(2p + 2\lg p)\alpha + 6k((p-1)/p)\beta$

Less than $6k$, asymptotically optimal

Scalability analysis for dense/sparse Allreduce algorithms

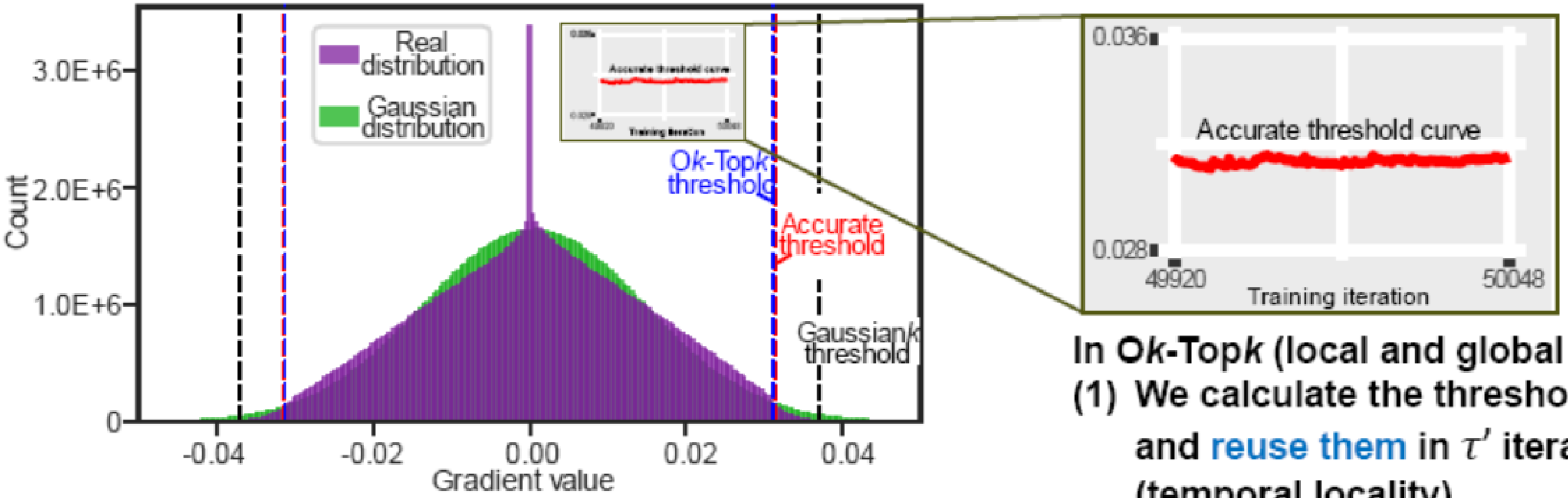
Algorithms	Bandwidth	Scalability
Dense [12]	$2n\frac{P-1}{P}\beta$	
Top$k$A [36, 47]	$2k(P\text{-}1)\beta$	👎👎
Top$k$DSA [36]	$[4k\frac{P-1}{P}\beta, (2k+n)\frac{P-1}{P}\beta]^1$	👎
gTop$k$ [42]	$4k(\log P)\beta$	👎
Gaussian$k$ [41]	$2k(P\text{-}1)\beta$	👎👎
Ok-Top$k$ (ours)	$[2k\frac{P-1}{P}\beta, 6k\frac{P-1}{P}\beta]^1$	👍



Existing sparse Allreduce algorithms suffer from scalability issue.
Ok-Topk solves the issue!

Efficient **local** and **global top-k selection**

BERT on Wikipedia with density = 1.0%



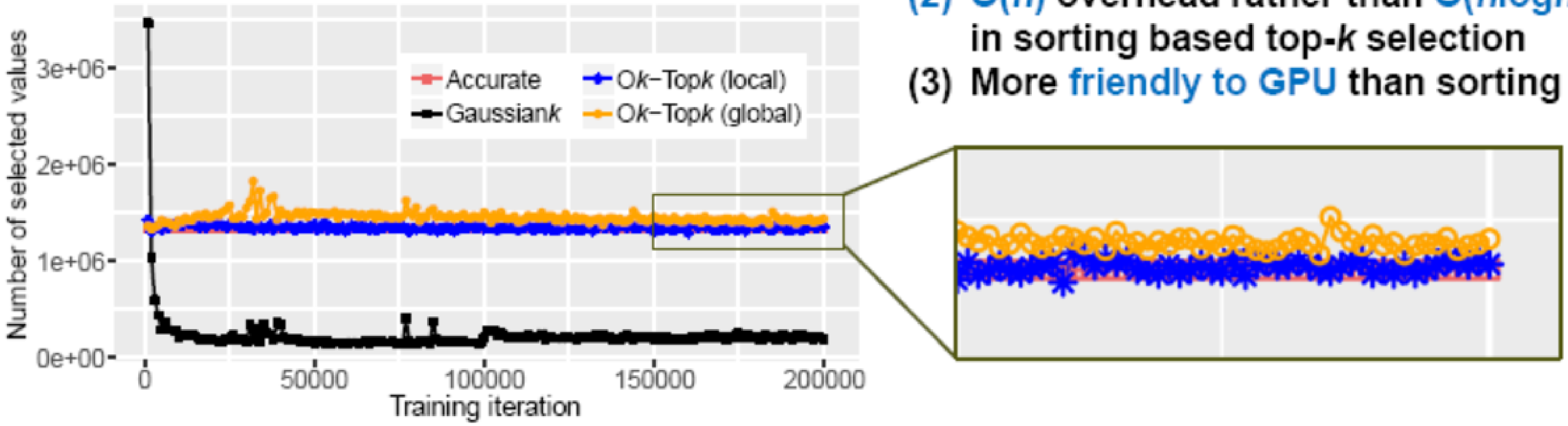
In **Ok-Top*k* (local and global top-*k*)**:

(1) **We calculate the thresholds once and reuse them in τ' iterations (temporal locality)**

(2) **O(n) overhead rather than O($n$log$n$) in sorting based top-*k* selection**

(3) **More friendly to GPU than sorting**

BERT on Wikipedia on 32 GPUs (density=1.0%, τ' =128)



Ok-Top$k$ parallel SGD algorithm

Inputs: stochastic gradient $G^i(\cdot)$ at worker i, value k, learning rate α.

Initialize $\epsilon_0^i = 0, G_0^i = 0$

for $t = 1$ **to** T **do**

  $\boxed{acc_t^i = \epsilon_{t-1}^i + \alpha G_{t-1}^i(w_{t-1})}$ ▷ Accumulate residuals

  $u_t, \boxed{indexes} = Ok\_sparse\_allreduce(acc_t^i, t, k)$

  $\boxed{\epsilon_t^i = acc_t^i - acc_t^i(indexes)}$ ▷ Update residuals

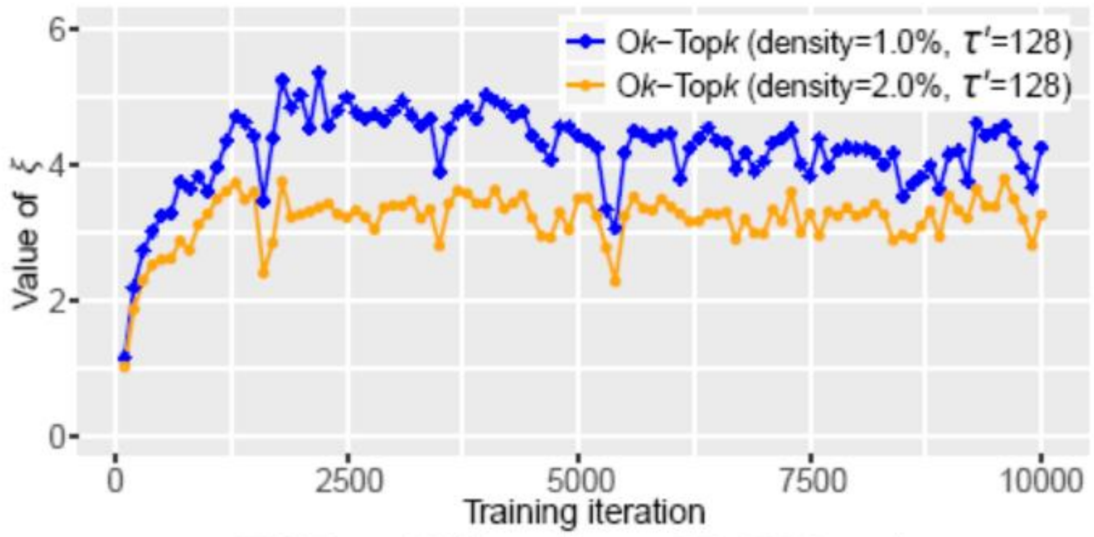
  $w_t = w_{t-1} - \frac{1}{P} u_t$ ▷ Apply the model update

end for

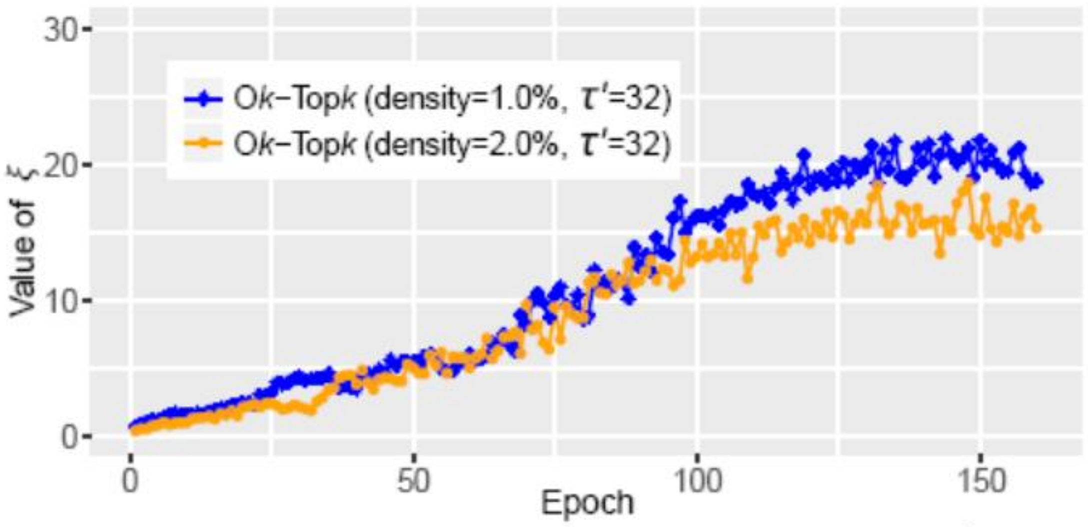
Convergence analysis for O*k*-Top*k* SGD

$$\left\|\underbrace{\mathrm{Top}k\left(\frac{1}{P}\sum_{i=0}^{P-1}\left(\alpha G_t^i(w_t)+\epsilon_t^i\right)\right)}_{\text{True global top-}k\text{ gradient}} - \underbrace{\mathrm{Top}k\left(\frac{1}{P}\sum_{i=0}^{P-1}\mathrm{Top}k\left(\alpha G_t^i(w_t)+\epsilon_t^i\right)\right)}_{\text{O}k\text{-top}k\text{ gradient}}\right\| \le \xi \underbrace{\|\alpha G_t(w_t)\|}_{\text{Dense gradient}}$$

For full proof refer to:
Dan Alistarh, et al., The convergence of sparsified gradient methods, NeurIPS'18



BERT on Wikipedia on 32 GPU nodes



LSTM on AN4 running on 32 GPU nodes

The effect of ξ is dampened by both small learning rates and P.

Evaluation

- **CSCS Piz Daint** supercomputer
  - Each node contains an Intel Xeon E5-2690 CPU, and one NVIDIA Tesla **P100 GPU**
  - **Cray Aries** interconnected network
  - **mpi4py** as the communication library, built against Cray-MPICH 7.7.16

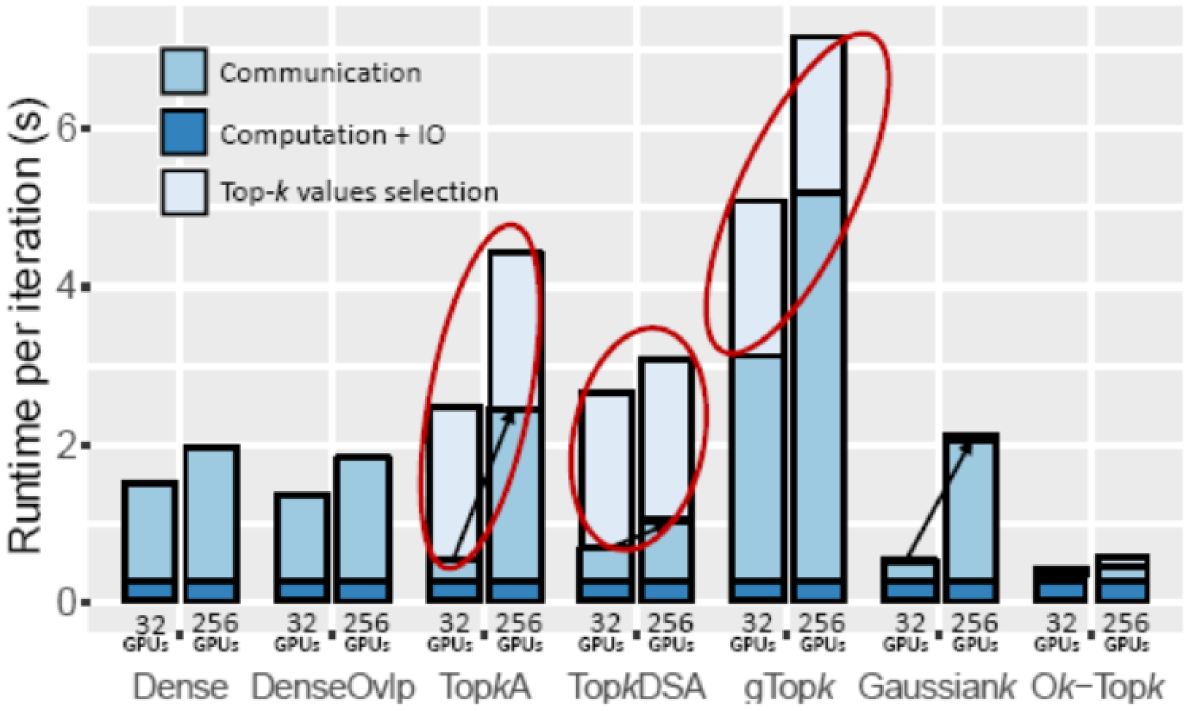
Dense/Sparse algorithms used in evaluation

Algorithms	Bandwidth
Dense [12]	$2n\frac{P-1}{P}\beta$
Top$k$A [36, 47]	$2k(P\text{-}1)\beta$
Top$k$DSA [36]	$[4k\frac{P-1}{P}\beta, (2k+n)\frac{P-1}{P}\beta]^1$
gTop$k$ [42]	$4k(\log P)\beta$
Gaussian$k$ [41]	$2k(P\text{-}1)\beta$
Ok-Top$k$ (ours)	$[2k\frac{P-1}{P}\beta, 6k\frac{P-1}{P}\beta]^1$

Neural networks used for evaluation

Tasks	Models	Parameters	Dataset
Image classification	VGG-16 [44]	14,728,266	Cifar-10
Speech recognition	LSTM [21]	27,569,568	AN4 [1]
Language processing	BERT [13]	133,547,324	Wikipedia [13]

Weak scaling evaluation

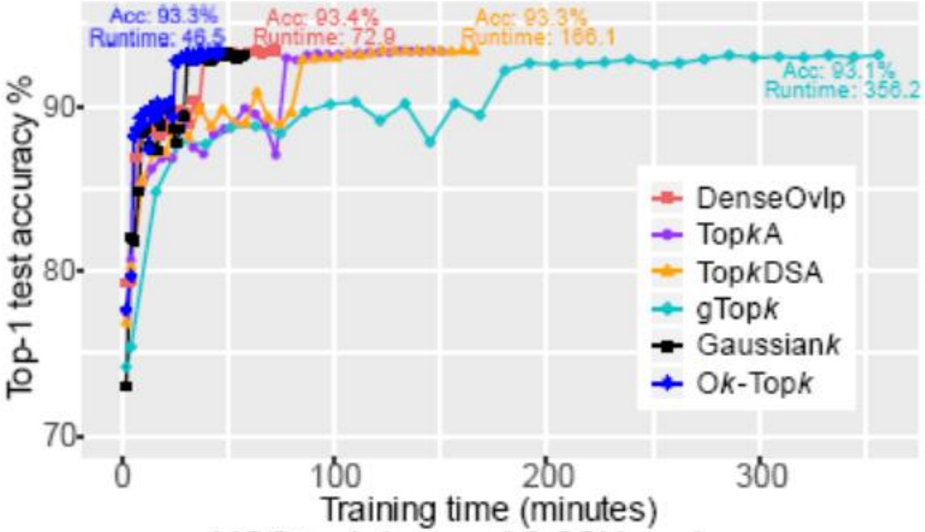


BERT training time,
scaling from 32 GPUs to 256 GPUs

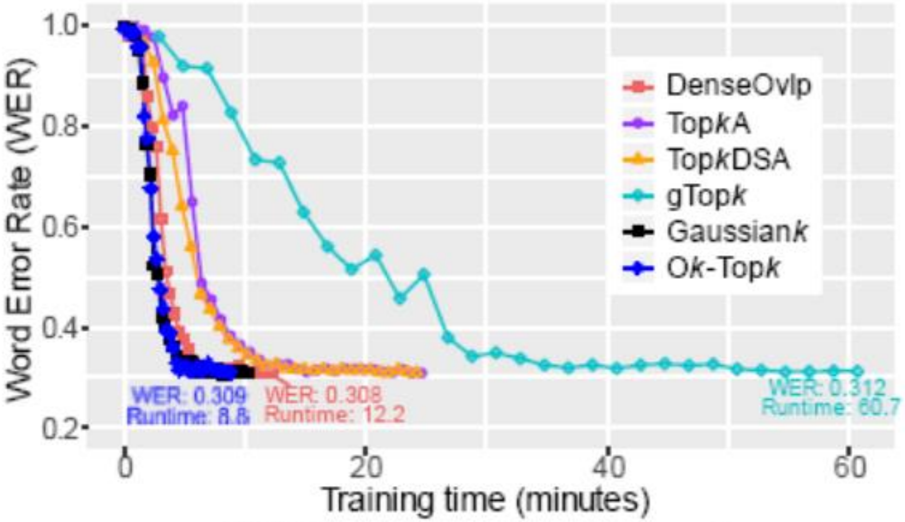
For Ok-Topk :

(1) Much better scalability for the communication overhead than the others.
(2) Threshold reuse strategy for top-*k* selection is very effective.

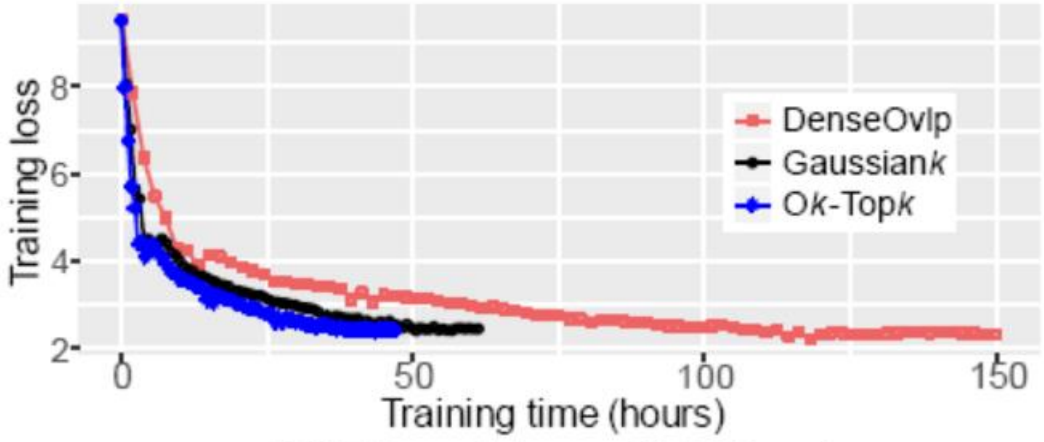
Model accuracy evaluation



VGG training on 16 GPU nodes



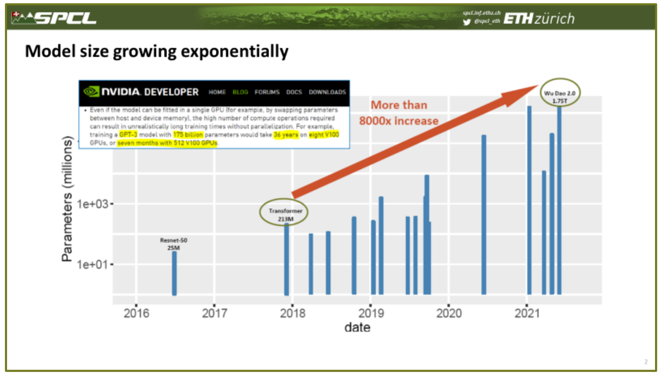
LSTM training on 32 GPU nodes



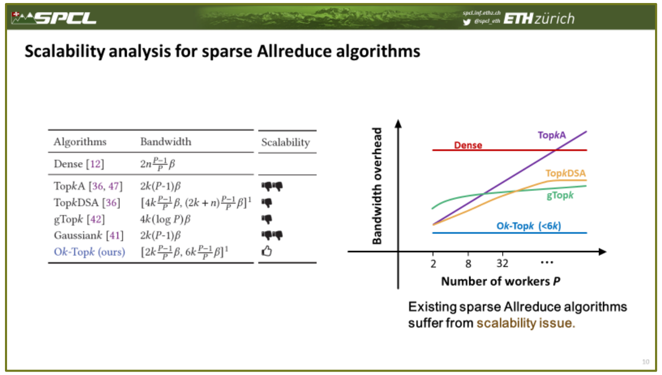
BERT Pretraining on 32 GPU nodes

Conclusion

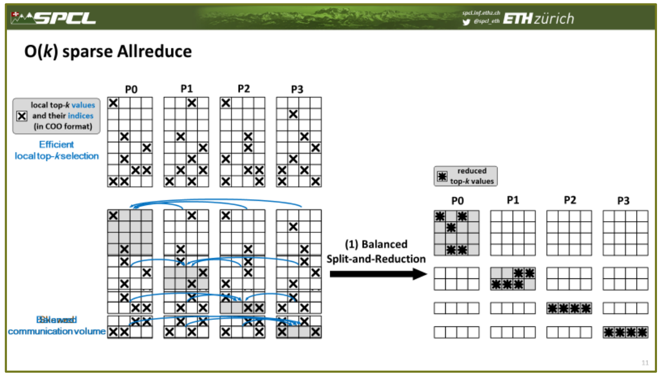
1. *Model size rapidly grows*



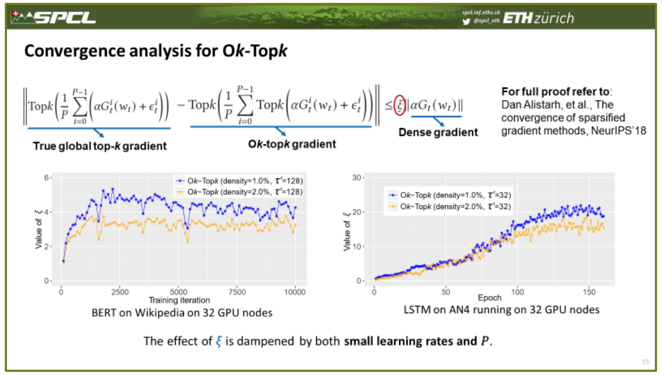
2. *Sparse algorithms suffer from scalability issue*



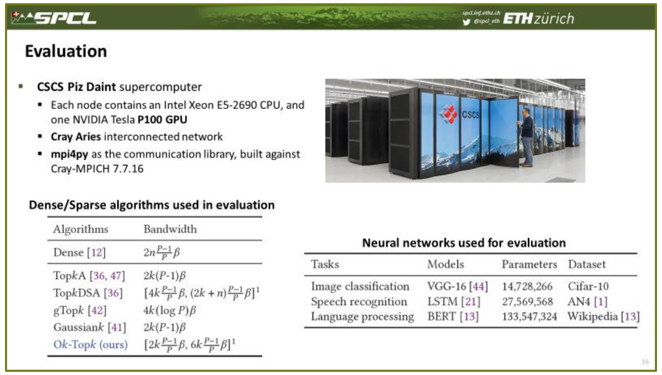
3. *O(k) sparse Allreduce*



4. *Ok-Topk SGD and convergence analysis*



5. *Evaluation on supercomputer*



6. ***For the future work**, we will study how to use **Ok-Topk** with **a hybrid data and pipeline parallelism.***

For more questions: shigangli.cs@gmail.com