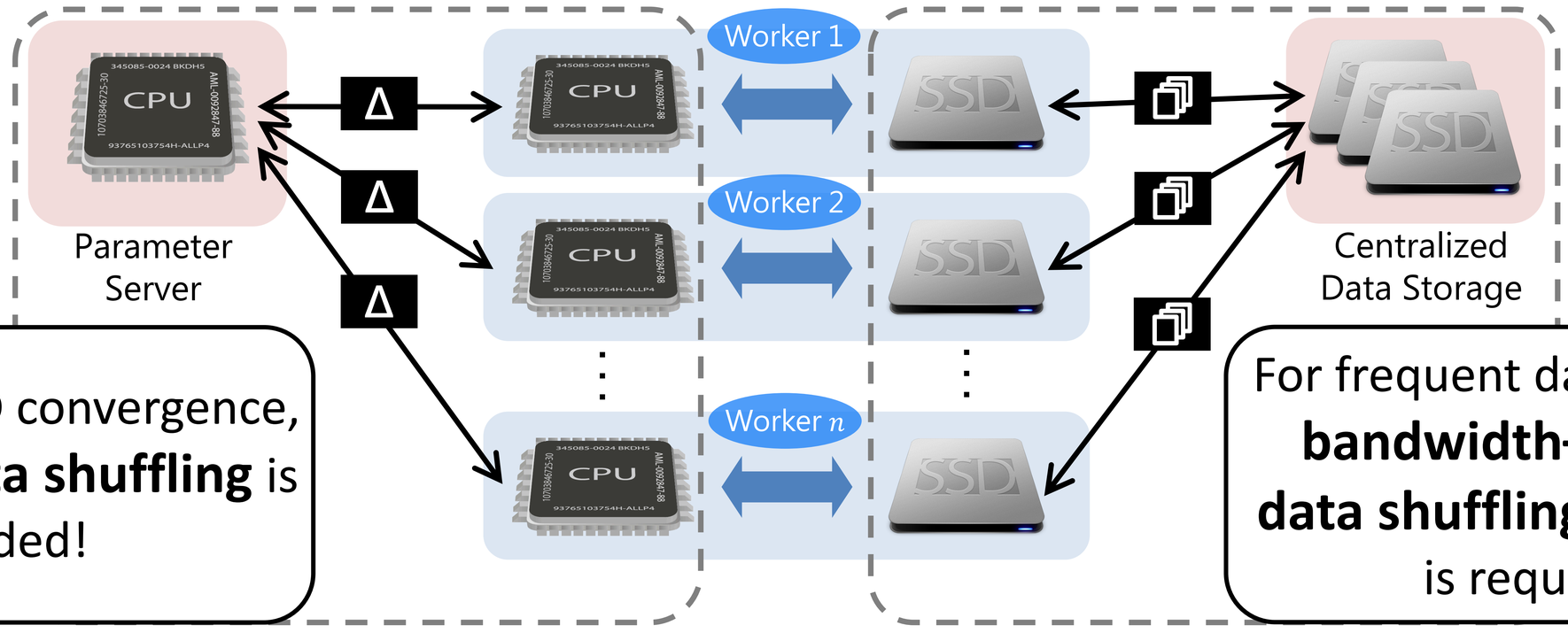
UberShuffle: Faster Distributed Learning via Erasure Coded Data Shuffling

Jichan Chung (KAIST), Kangwook Lee (KAIST), Ramtin Pedarsani (UC Santa Barbara),
Dimitris Papailiopoulos (UW Madison), and Kannan Ramchandran (UC Berkeley)

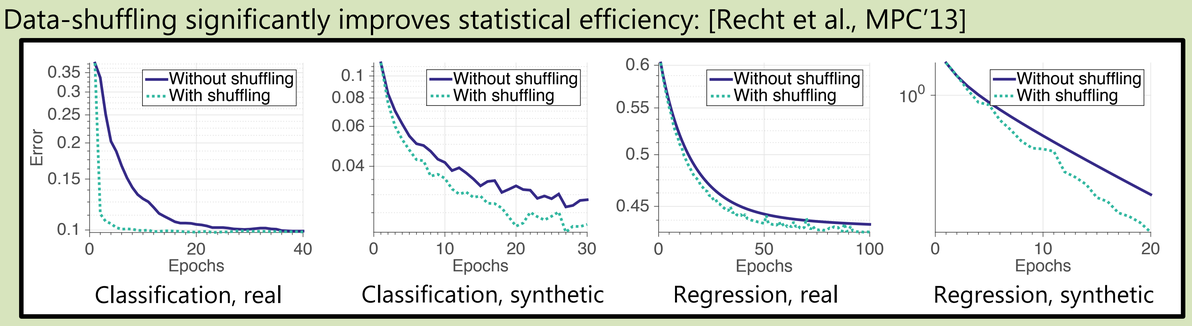


For faster SGD convergence, **frequent data shuffling** is needed!

For frequent data shuffling, **bandwidth–efficient data shuffling algorithm** is required!

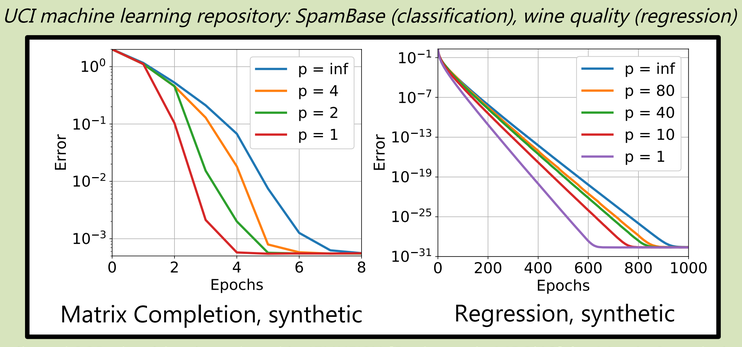
1. Data shuffling for statistical efficiency

Data-shuffling significantly improves statistical efficiency: [Recht et al., MPC'13]



UCI machine learning repository: SpamBase (classification), wine quality (regression)

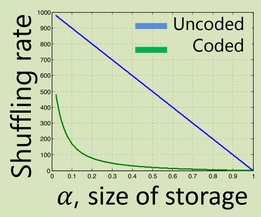
PSGD converges faster ***when data is shuffled more frequently.***



We use PSGD for Classification and Regression, and distributed SGD algorithm introduced in JELLYFISH [Recht et al., MPC'13] for Matrix Completion

2. Coded shuffling [1] Lee et al., NIPS MLSYS 2015, [2] Li et al., 2015

[1] Additional storage and **broadcast channel**
➔ Lower bandwidth for data shuffling!
([2] Additional computation ➔ Lower shuffle bandwidth for MR)

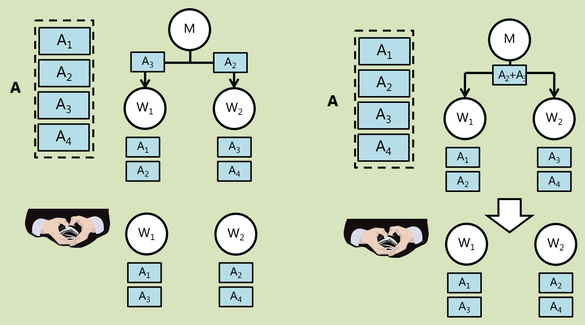


Theorem:
q = *the number of data points*
n = *the number of workers*
α = *the fraction of the data matrix that can be cached at each worker*
➔ *Coded shuffling achieves:*

$$R_c \to \frac{q(1-\alpha)}{\alpha n}$$

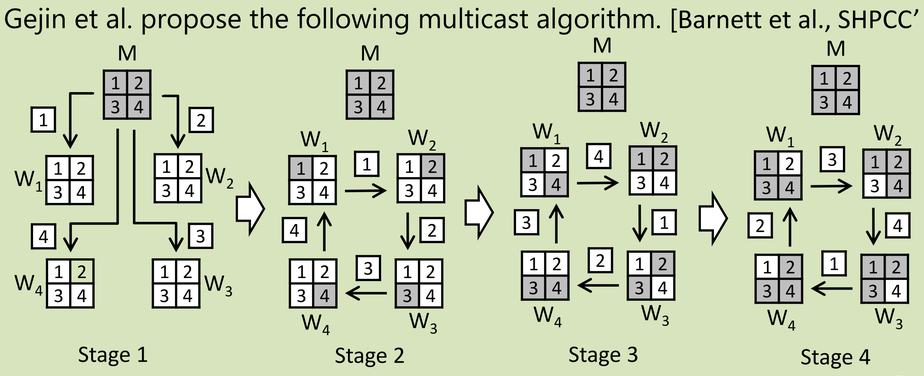
i.e., the coded shuffling can reduce the communication overhead by a factor of αn.

However, the theorem assumes:
(1) q grows to infinity
(2) perfect broadcast channel



3. Coded Shuffling for Multicast

Van de Gejin et al. propose the following multicast algorithm. [Barnett et al., SHPCC' 1994]



(For n= 4)

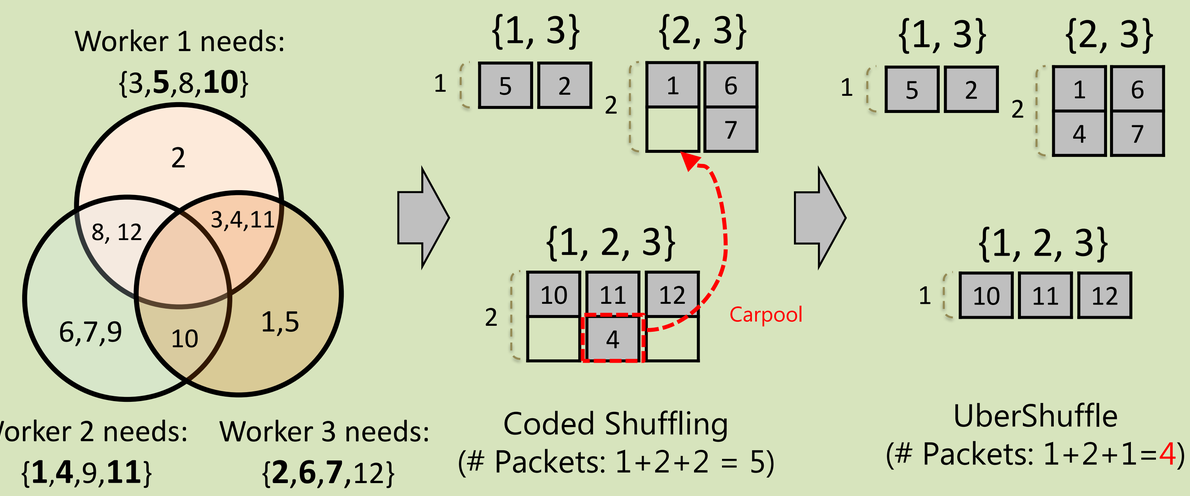
$$T_{multicast} = \left(\frac{2n-1}{n}\right) T_{unicast} \approx 2T_{unicast}$$

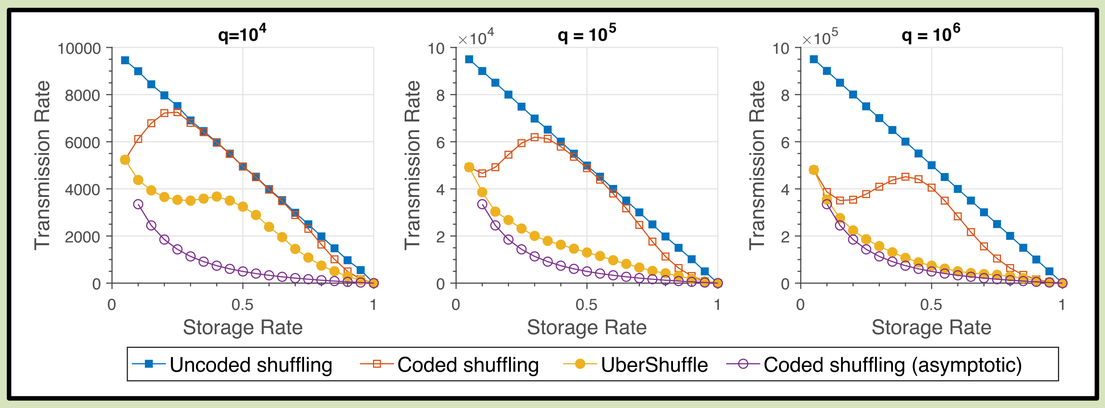
Hence, shuffling time of coded shuffling algorithm under multicast channel is:

$$T_{shuffle,multicast} \approx \frac{2q}{B}\left(\frac{1-\alpha}{\alpha n}\right) T_{unicast}$$

4. UberShuffle

Coded Shuffling's **strict asymptotic requirement** ➔ not optimal in practice!





➔ UberShuffle approaches the asymptotic result much faster!

5. Experimental Results

- Implemented using OpenMPI with C
- Runs Coded Shuffling / UberShuffle using **Multicast**
- EC2: 20 m3.xlarge workers + 1 m3.2xlarge master

Shuffling time: up to ***47.2%*** *↓*
Training time: up to ***32.1%*** *↓*