上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

A Coding Framework for Distributed Computing with Two Types of Servers

Jiasheng Xu

2017/5/27

Outline:

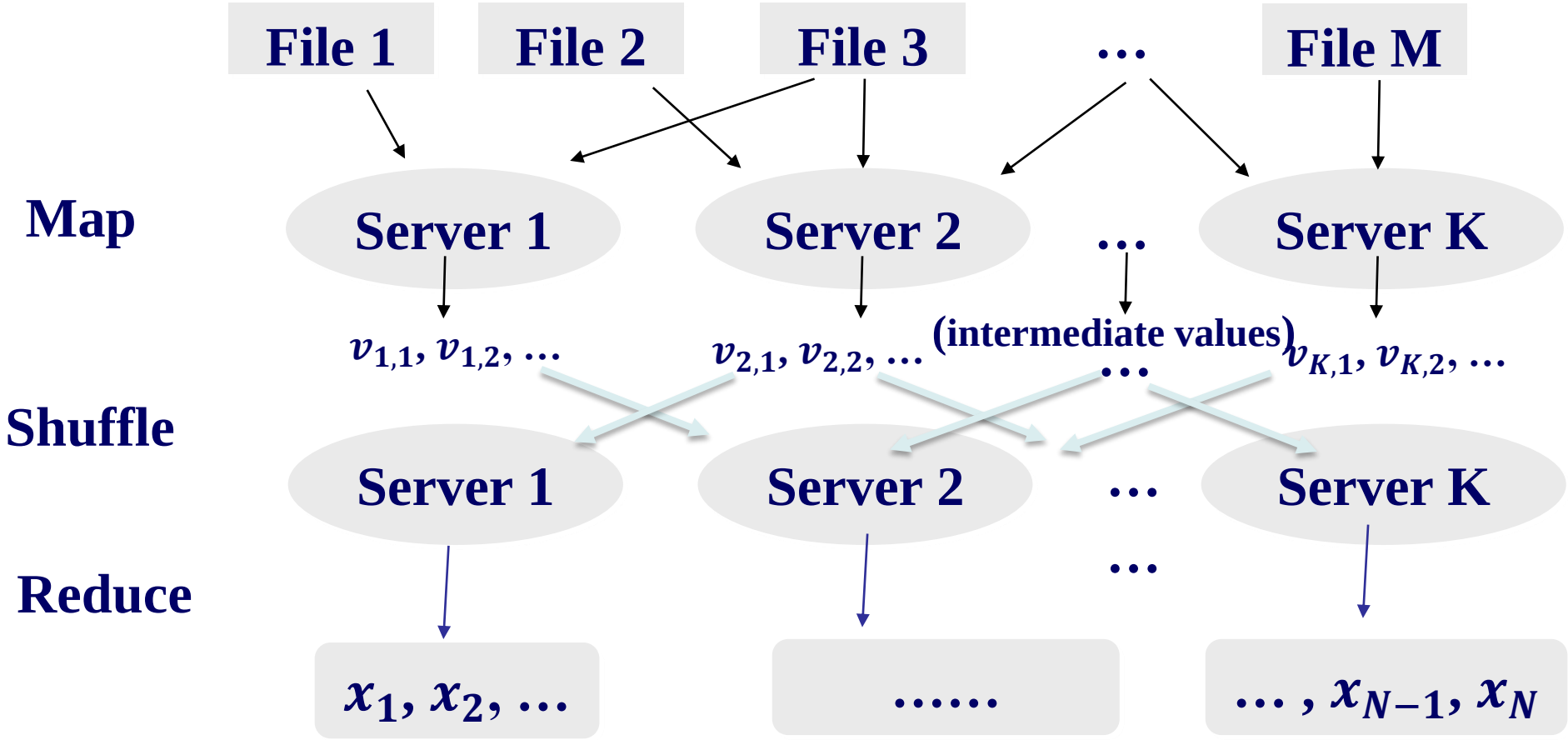
1. Introduction
   - -- Coded Distributed Computing
   - -- MapReduce
2. Existing Problem
   - -- Straggling Servers
3. New Scheme
   - -- Description
   - -- Results
   - -- Analysis
4. Reference

Introduction

Coded Distributed Computing (CDC)

-- Significantly reduce the communication load via coding and some extra computing.

MapReduce:



Concepts of Coded MapReduce

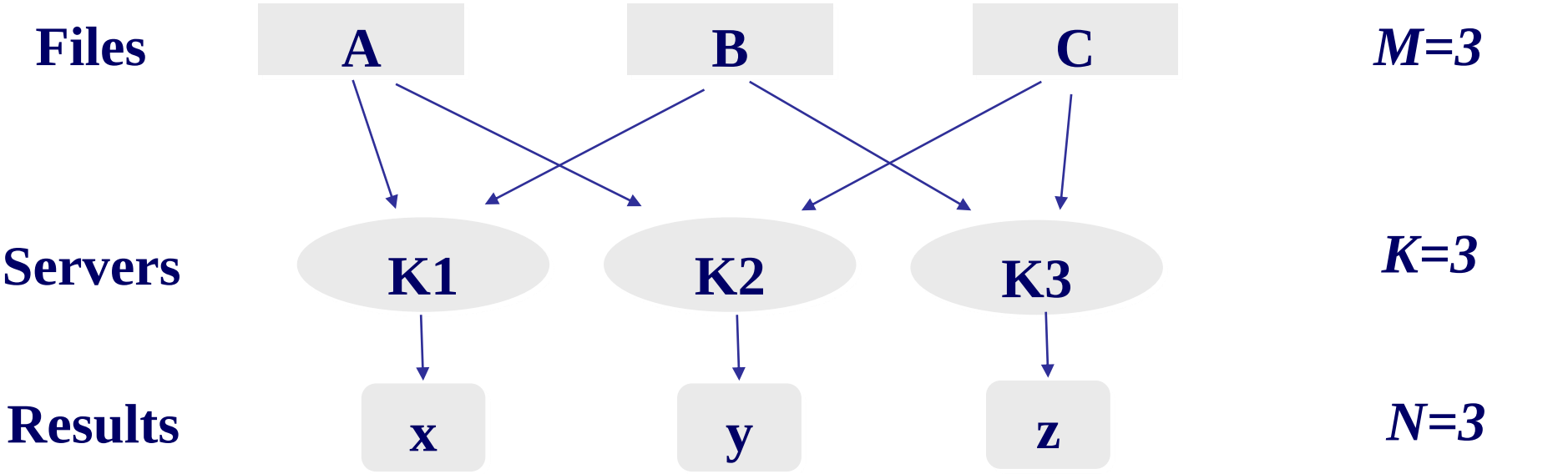
Settings: M input files $\xrightarrow{\text{K servers}}$ N output results

Definitions:

***Computation load (r)*: the total number of files computed across the K servers in the Map phase, normalized by the number of files M.**
(It can be interpreted as the average number of servers that map each file)

***Communication load (L)*: the number of bits communicated by the K servers during the Shuffle phase, normalized the total number of bits in all intermediate values**

Example for Coded MapReduce



K1

$v_{A,x}$, $v_{A,y}$, $v_{A,z}$,
$v_{B,x}$, $v_{B,y}$, $v_{B,z}$

$v_{C,x}$

K2

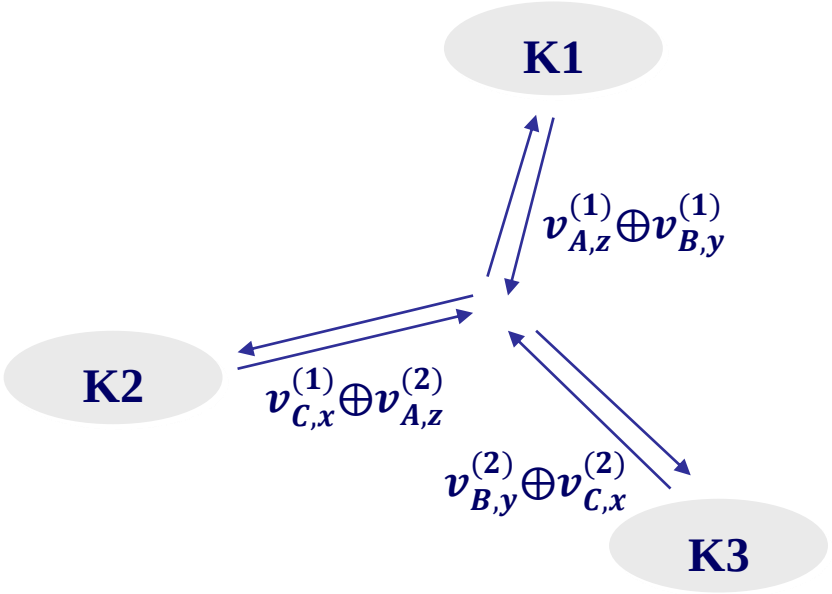
$v_{A,x}$, $v_{A,y}$, $v_{A,z}$,
$v_{C,x}$, $v_{C,y}$, $v_{C,z}$

$v_{B,y}$

K3

$v_{C,x}$, $v_{C,y}$, $v_{C,z}$,
$v_{B,x}$, $v_{B,y}$, $v_{B,z}$

$v_{A,z}$



The information transmitted in the Shuffle phase is

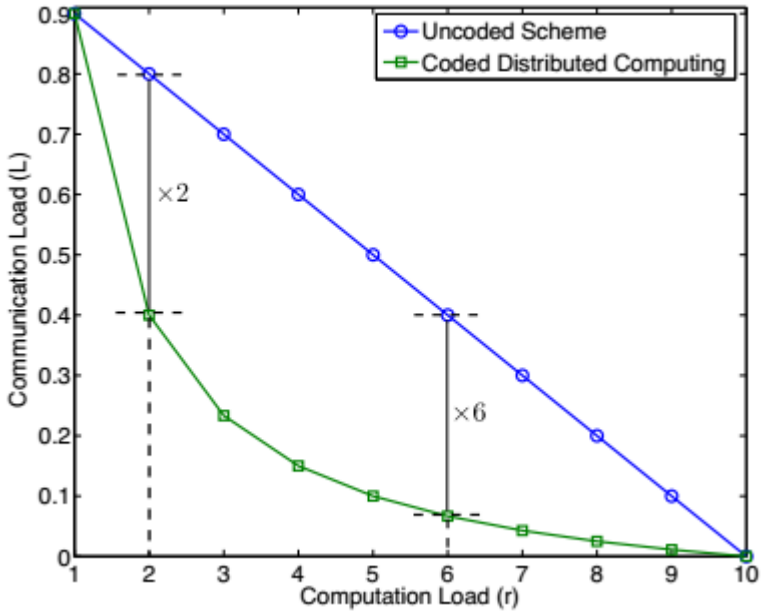
$3 \times \frac{1}{2} = 1.5$, $L = \frac{1.5}{9} = \frac{1}{6}$

For uncoded scheme, transmitted information is 3,

$L = \frac{3}{9} = \frac{1}{3}$

Results for general cases:

$$L_{coded}(r) = \frac{1}{r} \cdot \left(1 - \frac{r}{K}\right)$$



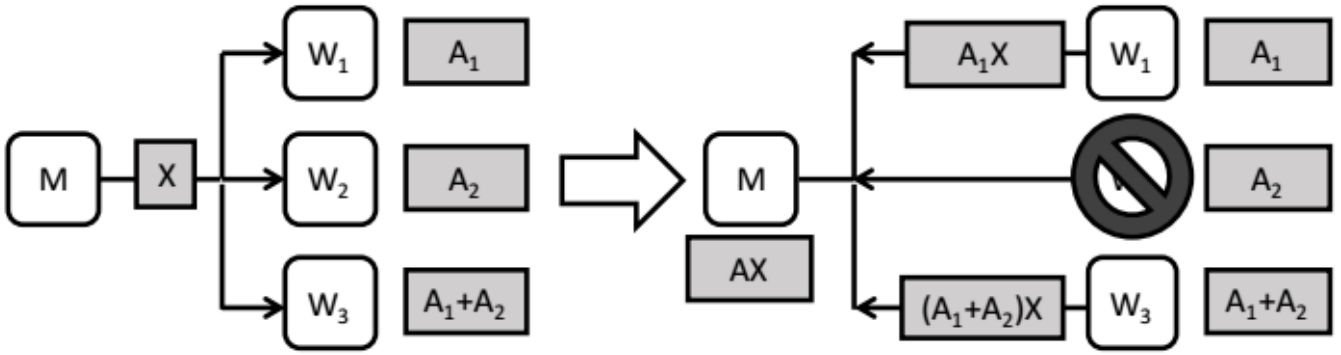
1. **The system achieves a reduction in L through repetitive computation.**

2. Repetitive computation **and** *coding* **enable multicast opportunity, allowing the system to achieve a coding gain.**

Existing Problem

Straggling Servers: Some servers in the system may encounter unpredictable failure and perform badly in computing.

In some prior works, codes are applied to alleviate the effects of stragglers by using more servers to compute.



Example for matrix multiplication

New Scheme

A different scheme for distributed computing which solves straggling problem.

Features:

1. *Asymmetric* design for all three phases of MapReduce

2. A reduction of communication load by factor r

Settings:

M input files, N output results,

K servers (λK fast servers, others are slow servers)

Definitions for L and r remain the same

Central idea: assign different amount of computing tasks to different types of servers

(Fast servers and slow servers have different functions)

Computation load for *fast* and *slow* servers: r^*, 1 ($\mathbf{r = r^*+1}$)

Map phase

The fraction of files computed in fast servers:

$$\mu_1 = \frac{r^*}{\lambda K}$$

slow servers:

$$\mu_2 = \frac{1}{K - \lambda K}$$

We aim to achieve:

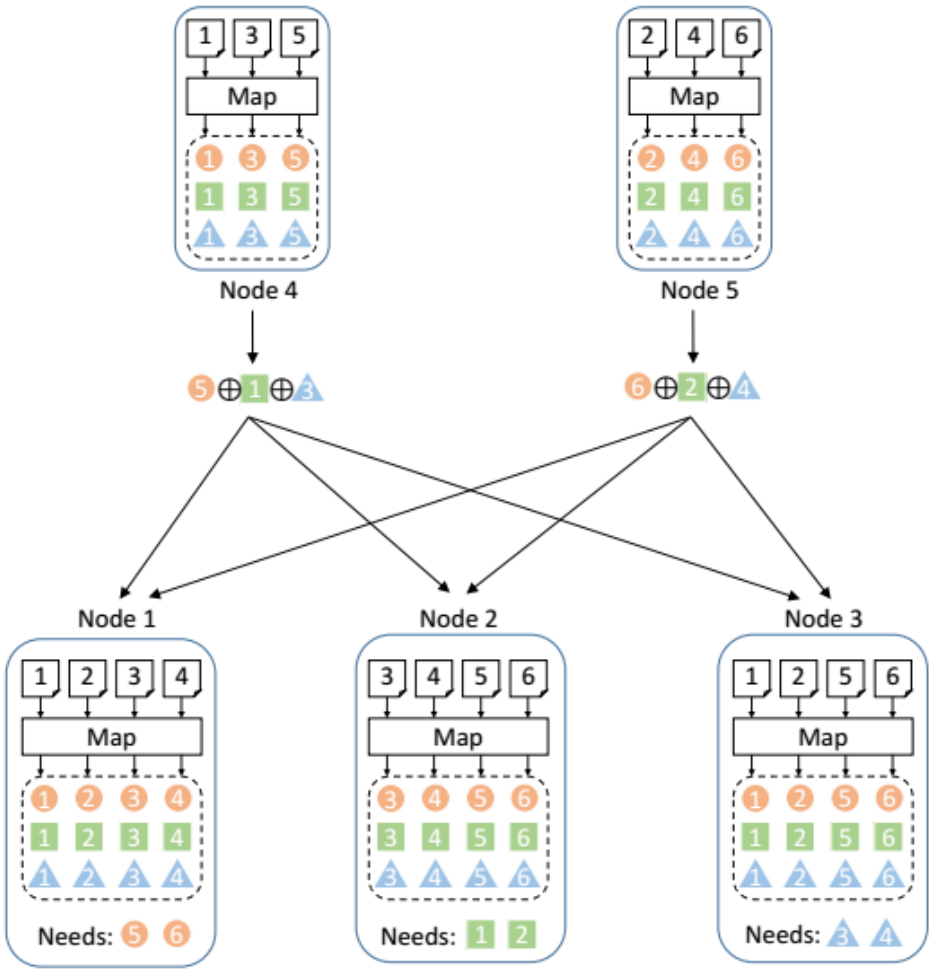
$$\frac{\mu_1}{\mu_2} = \beta$$

$$r^* = \frac{\lambda}{1-\lambda}\beta$$

Shuffle phase **Only *slow servers* are responsible for sending messages, fast servers receive the messages.**

Reduce phase **Only *fast servers* compute Reduce functions and get final results.**

Example for New Scheme



M=6, K=5, N=3
(λ=0.6, β=1.33)
r=2*

$$L = \frac{2}{18} = \frac{1}{9}$$

Uncoded scheme:

$$L = \frac{6}{18} = \frac{1}{3}$$

reduced by a factor
r = r*+1 = 3

Coding **enables multicast, so that a reduction of L is achieved**

General description for our proposed scheme

Map Phase Assignment. For simplicity, we denote $a = \lambda K$ and $b = K - \lambda K$ as the number of fast and slow servers, respectively. Each server assigned with Reduce tasks reduces h results in the Reduce phase, so we have N=ha. For large N, h is an integer.

Assume M is large, we evenly partition all files into $b\binom{a}{r^*}$ disjoint sets and map each set to a tuple of subset consisting of r^* fast servers and a slow server, which can be marked as (i, A), where i is one of the index of slow servers, and A is a cardinality-r^* subset of the set consisting of all fast servers' indices. We denote the set of files that is mapped to (i, A) by $\mathcal{B}_{i,A}$.

We let every server belonging to (i, A) maps all files in $\mathcal{B}_{i,A}$ in the Map phase, for all possible sets (i, A) and $\mathcal{B}_{i,A}$.

Shuffle Phase Assignment. We denote $V_{i,A,n}$ as a variable that contains all the intermediate values for reduce output function n from all files in $\mathcal{B}_{i,A}$. In this phase, each slow server will multicast the following messages: Find a subset of fast servers $\mathcal{S}$ whose cardinality is r^*+1, slow server i will multicast $Y_{i,\mathcal{S}} \triangleq \oplus_{k\in\mathcal{S}} V_{i,\mathcal{S}\backslash\{k\},k_j}$, where k_j is the j_{th} reduce function that server k has to compute, to all the fast servers in $\mathcal{S}$, for every $j \in [h]$. This procedure continues until all possible subsets $\mathcal{S}$ have been found.

Reduce Phase Assignment. Each fast server will compute their assigned Reduce tasks, the intermediate values needed for reduction are derived from either the Map phase or the Shuffle phase.

Results and Analysis

The communication load under our proposed scheme is given as follows:

$$\mathrm{L} = \frac{1}{r^*+1}\left(1 - \frac{r^*}{\lambda K}\right)$$

Results for Conventional Coded MapReduce

$$\mathbf{L}(\boldsymbol{r}) = \frac{\mathbf{1}}{\boldsymbol{r}} \cdot \left(\mathbf{1} - \frac{\boldsymbol{r}}{\boldsymbol{K}}\right)$$

1. **Not only alleviate the *straggling effect*, but also enable *coding opportunities* like the prior schemes.**
2. **The scheme is valid for *arbitrary* system parameters, e.g. λ, β.**
3. **When the proportion of fast servers λ is small, a *lower* L is achieved compared with the conventional schemes.**

Reference

[1] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," e-print arXiv:1604.07086, Apr. 2016, submitted to IEEE Trans. Inf. Theory.

[2] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded MapReduce," 53rd Allerton Conference, Sept. 2015.

[3] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," arXiv preprint arXiv:1609.01690, 2016.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," CoRR, abs/1512.02673, 2015.

[5] R. Tandon, Q. Lei, A. Dimakis, and N. Karampatziakis, "Gradient Coding," arXiv preprint arXiv:1612.03301, 2016.

[6] Q. Yu, S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "How to Optimally Allocate Resources for Coded Distributed Computing?" arXiv preprint arXiv:1702.07297, 2017.

[7] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," IEEE Transactions on Information Theory, vol. 60, no. 5, pp. 2856–2867, Mar. 2014.

Thanks!