# Coded MapReduce

Mohammad Ali Maddah-Ali

Bell Labs, Alcatel-Lucent

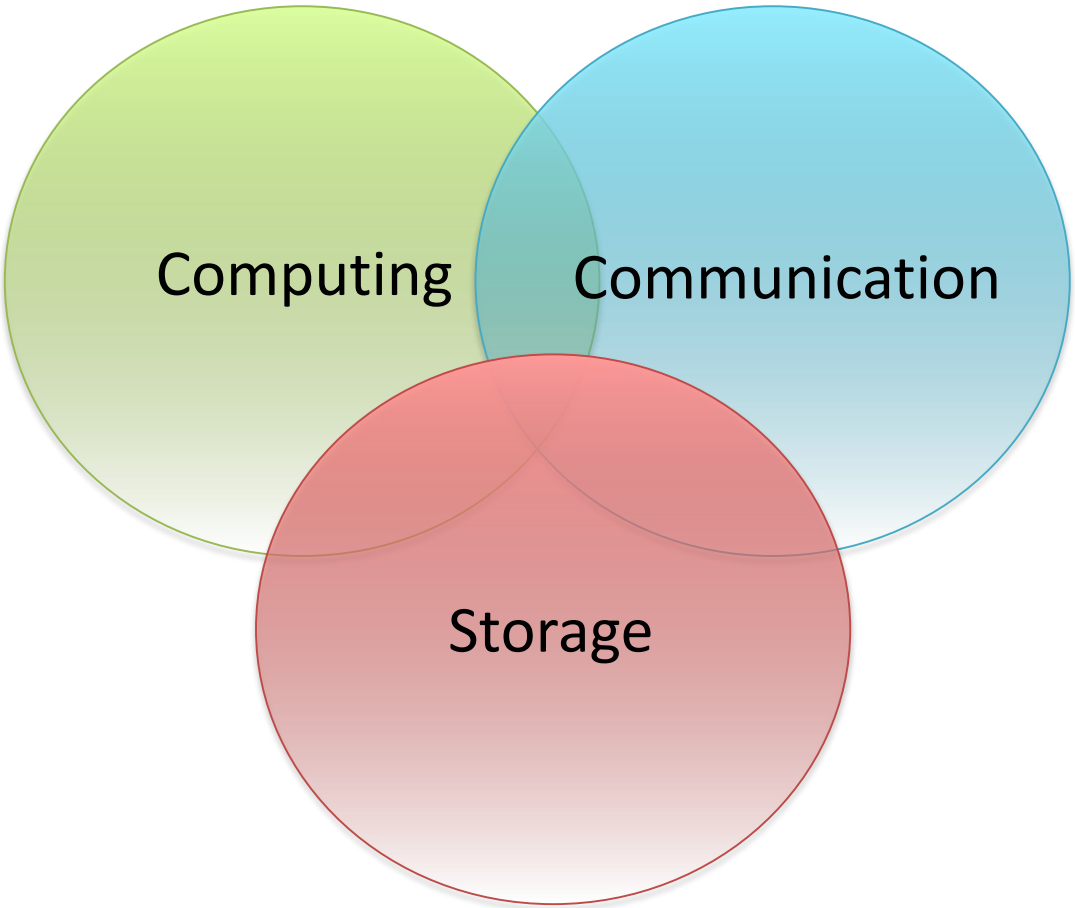
joint work with

Sonze Li (USC) and Salman Avestimehr (USC)

DIMACS

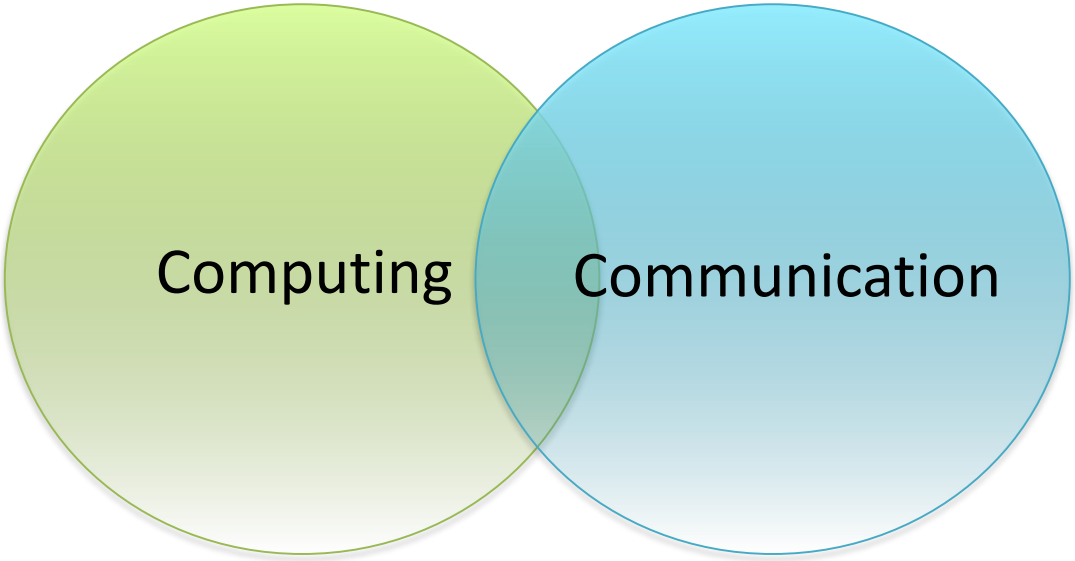
Dec. 2015

# Infrastructure for big data




The interaction among major components is the limiting barrier!

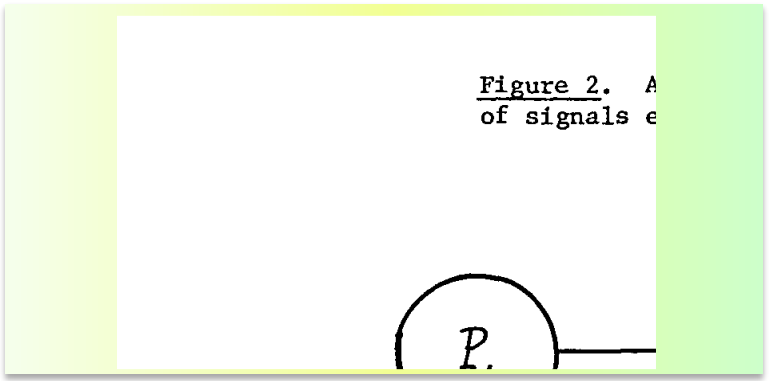
# In this talk



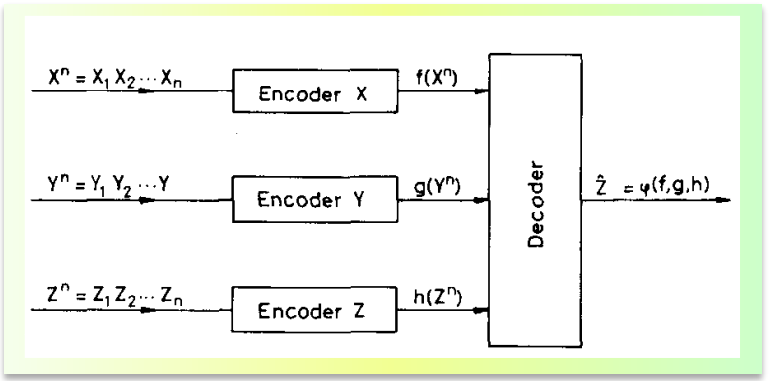

Fundamental tradeoff between Computing and Communication

# Formulation

**Minimum communication** for a specific **computation task**?




**Computer Science** (Yao 1979)




**Information Theory** (Korner and Marton 1979)

**Shortcomings:**

- Problem oriented
- Does not scale

**Need a framework that is**

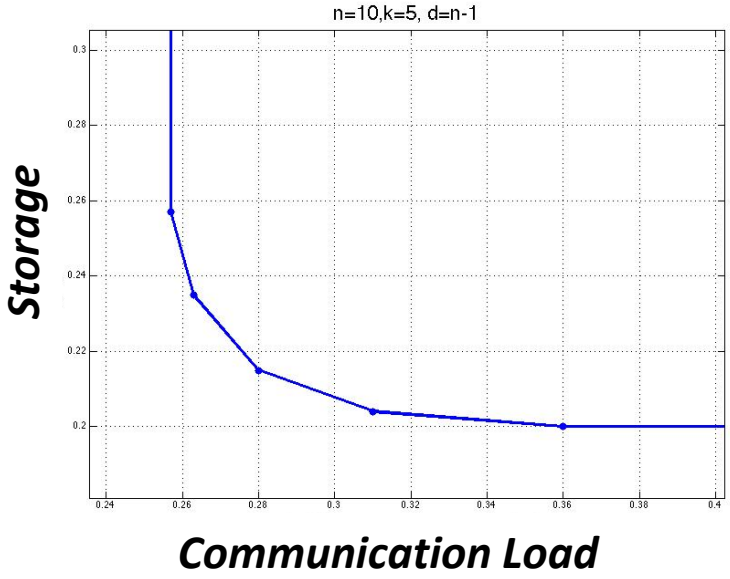
- General
- Scalable

Challenge: right formulation

What does data companies are using?

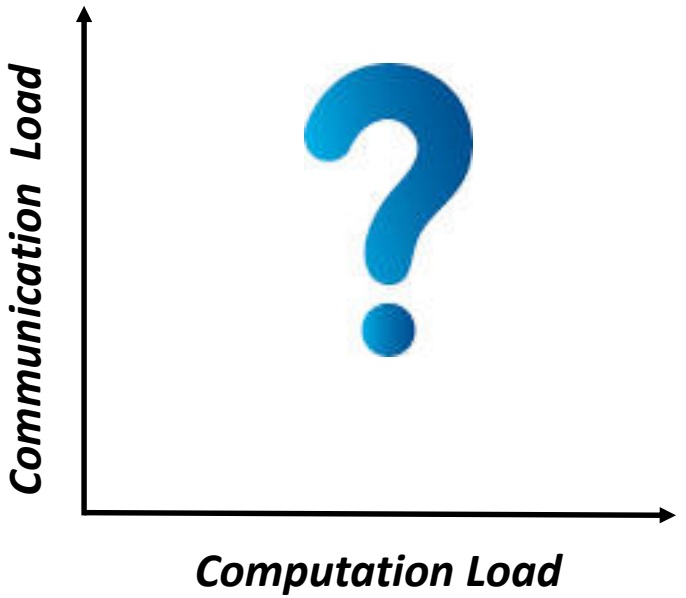

## Storage

**Hadoop Distributed File Systems (HDFS)**




## Computation

**MapReduce**

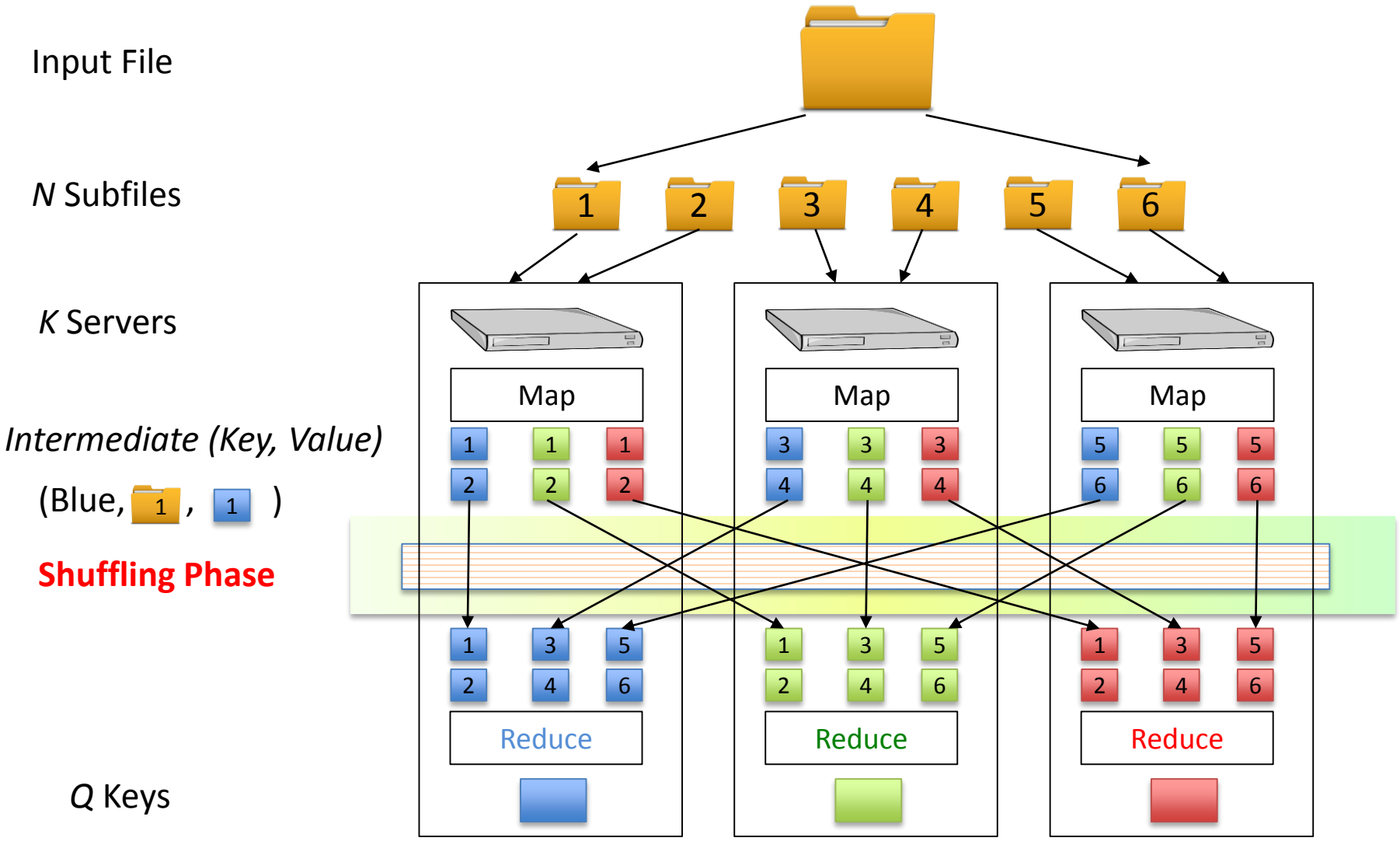



Refer to Yesterdays' Talks:

- Alexander Barg
- Alexander Dimakis

# MapReduce: A General Framework

***N Subfiles, K Servers, Q Keys***



# Example: Word Counting

***N Subfiles, K Servers, Q Keys***

A Books

*N=6* Chapters

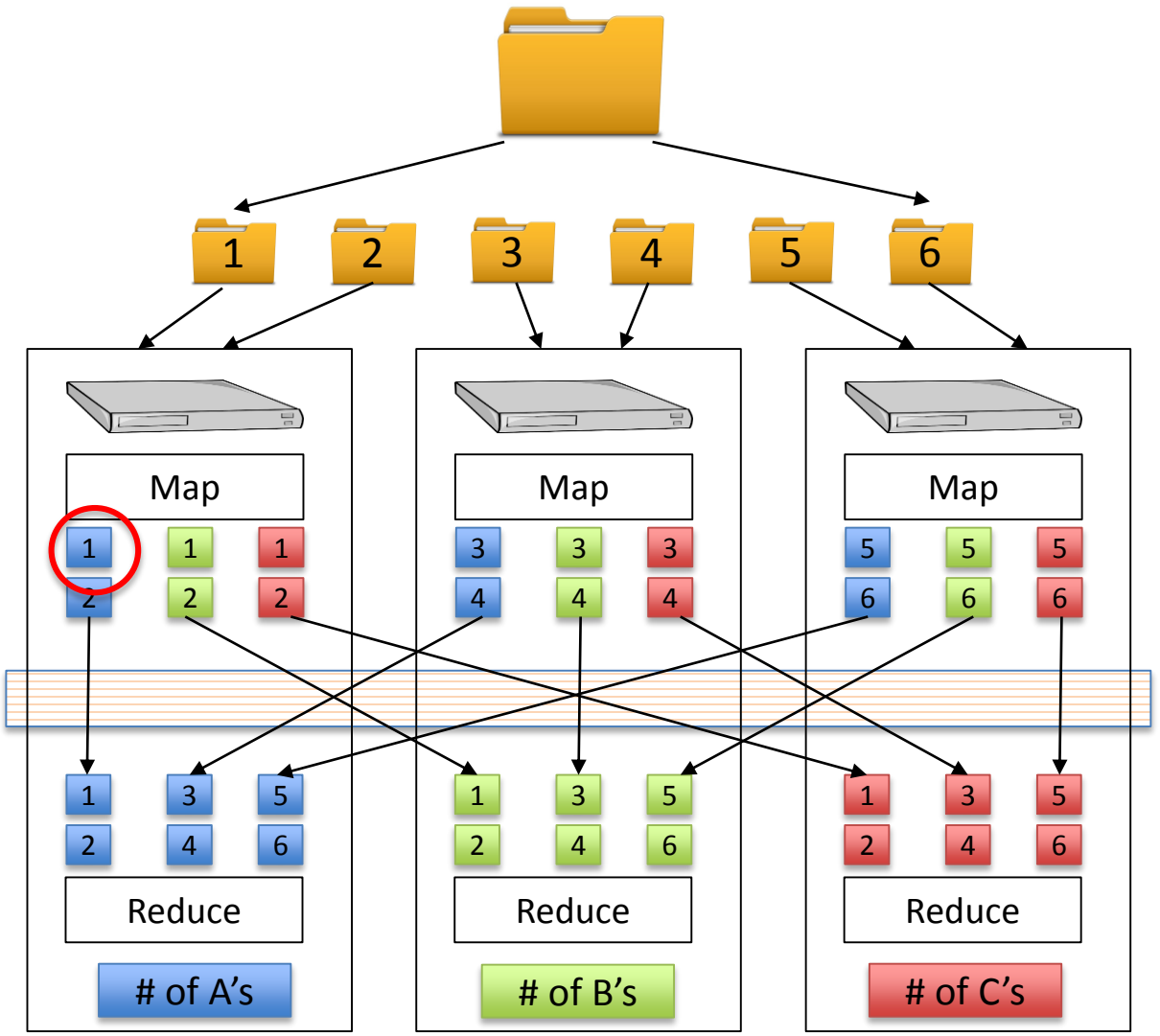
*K=3* Servers

*Intermediate (Key, Value)*

(A, 1, 1)

Number of A's in chapter one

*Q=3* Keys



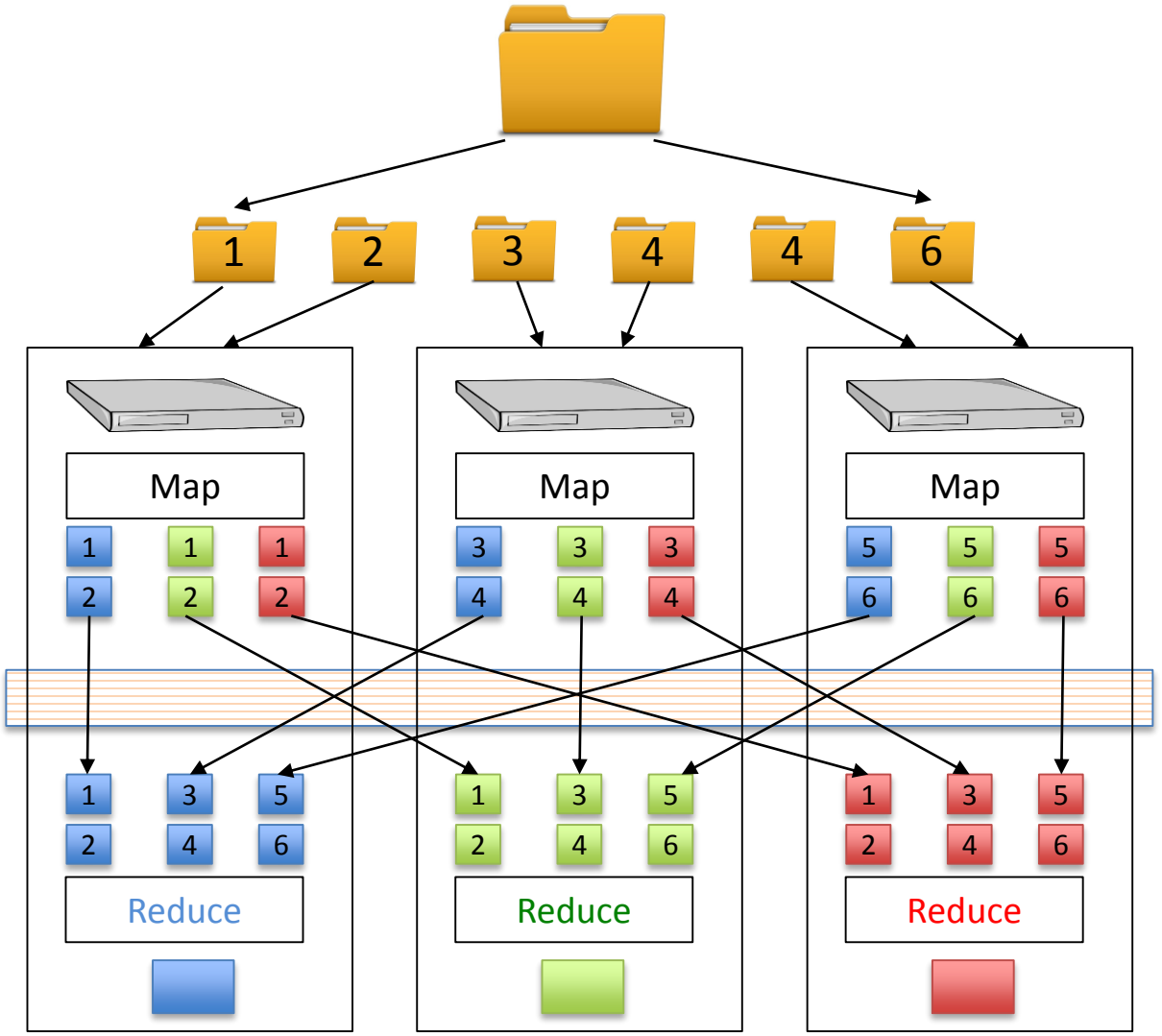
# MapReduce: A General Framework

***N Subfiles, K Servers, Q Keys***

General Framework

- Matrix Multiplication
- Distributed Optimization
- Page Rank
- ….

**Active Research Area:**
How to fit different jobs into this framework.

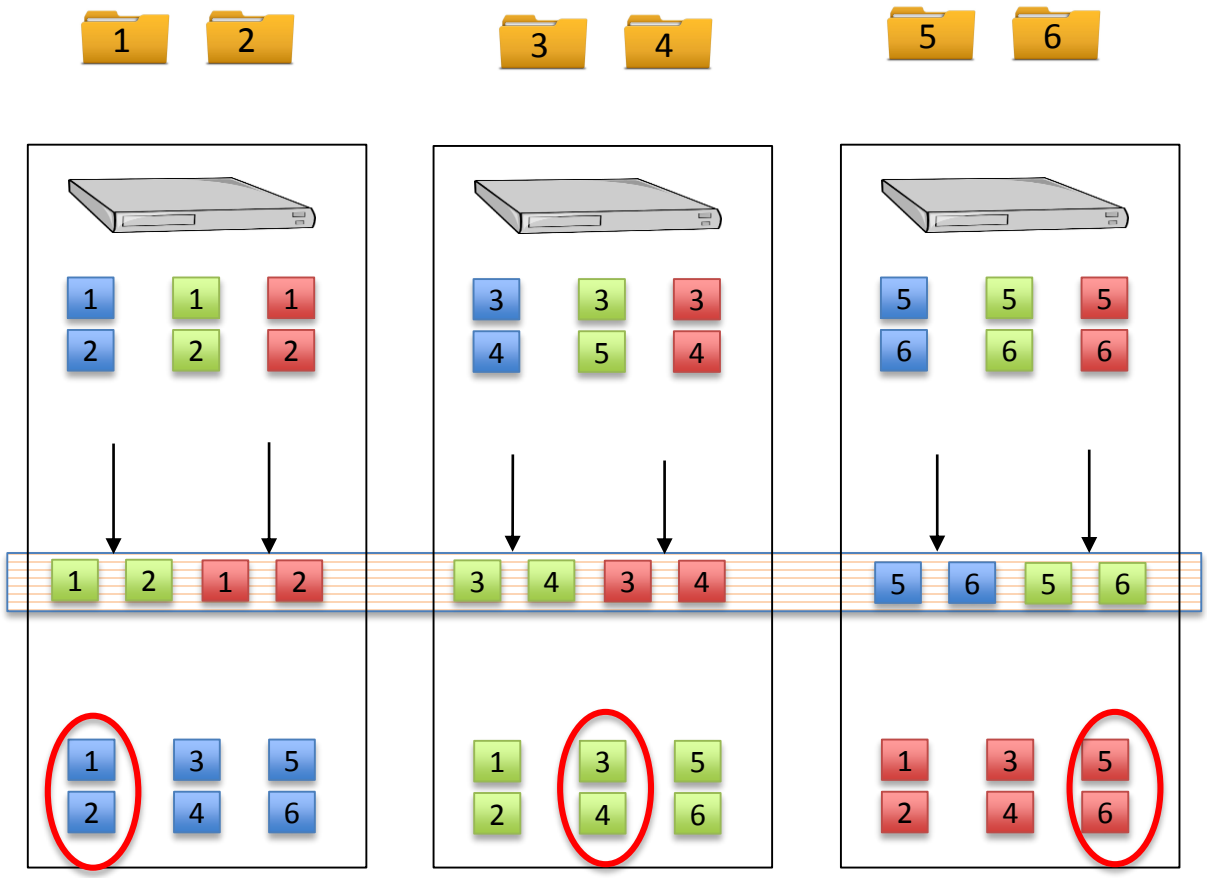


# Communication Load

***N=6 Subfiles, K=3 Servers, Q=3 Keys***

**Communication Load (MR)**

$$L_{MR} = QN(1 - \frac{1}{K})$$

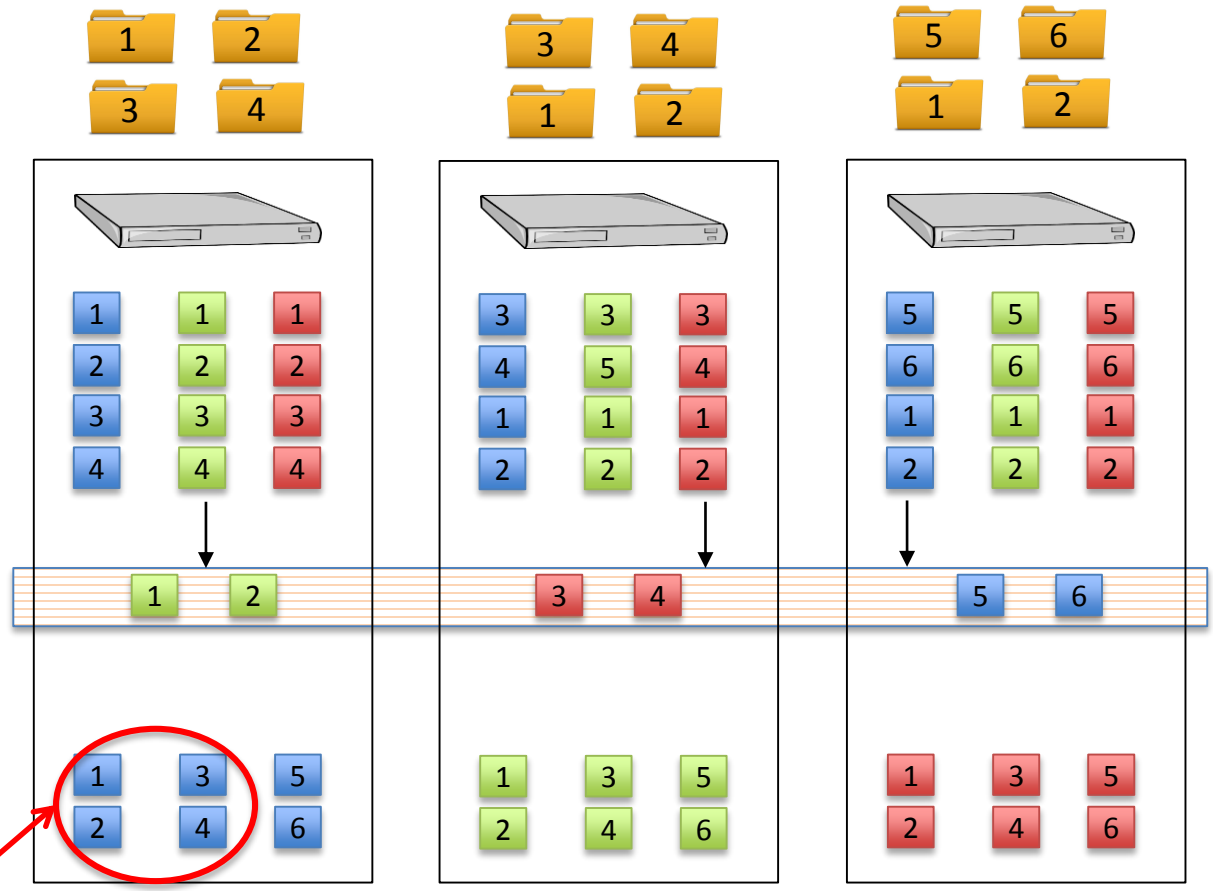



**Communication is a bottleneck!**

**Can we reduce communication load at the cost of computation?**

# Communication Load

***N Subfiles, K Servers, Q Keys, Comp. Load r***




**Comm. Load (Uncoded)**

$$L_U(r) = QN(1 - \frac{r}{K})$$

**Locally available**

# Communication Load

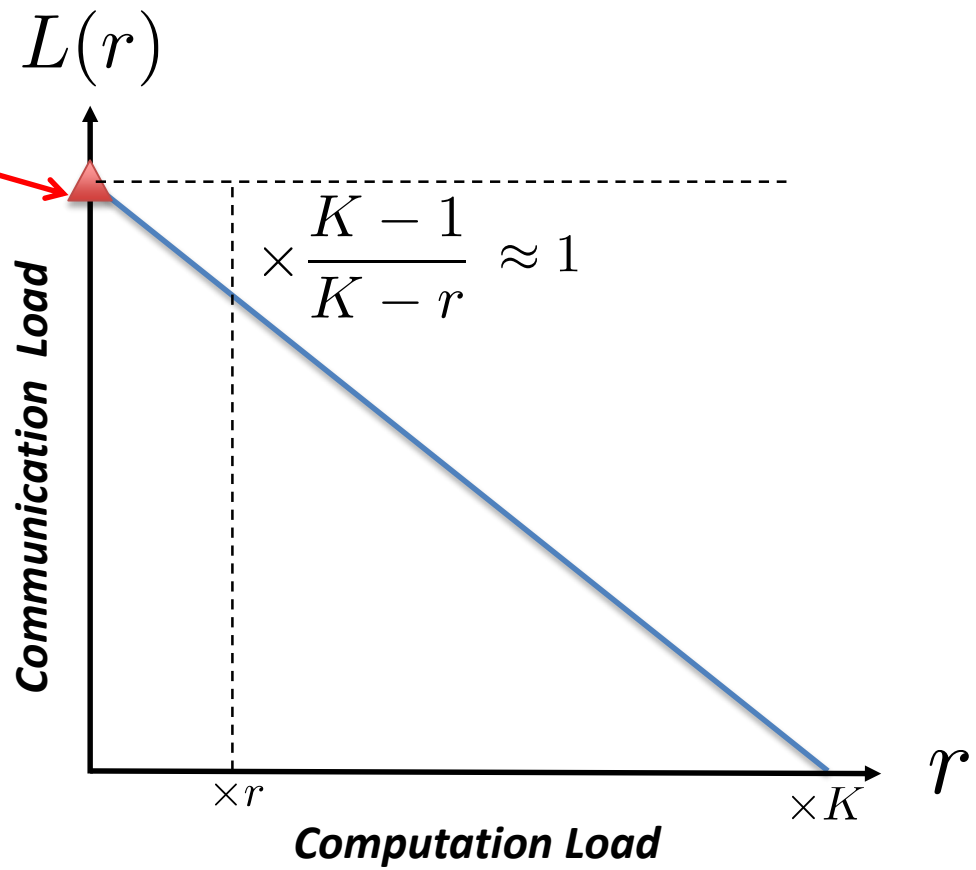
***N Subfiles, K Servers, Q Keys, Comp. Load r***

**Comm. Load (Map Reduce)**

$$L_{MR} = QN(1 - \frac{1}{K})$$

**Comm. Load (Uncoded)**

$$L_U(r) = QN(1 - \frac{r}{K})$$

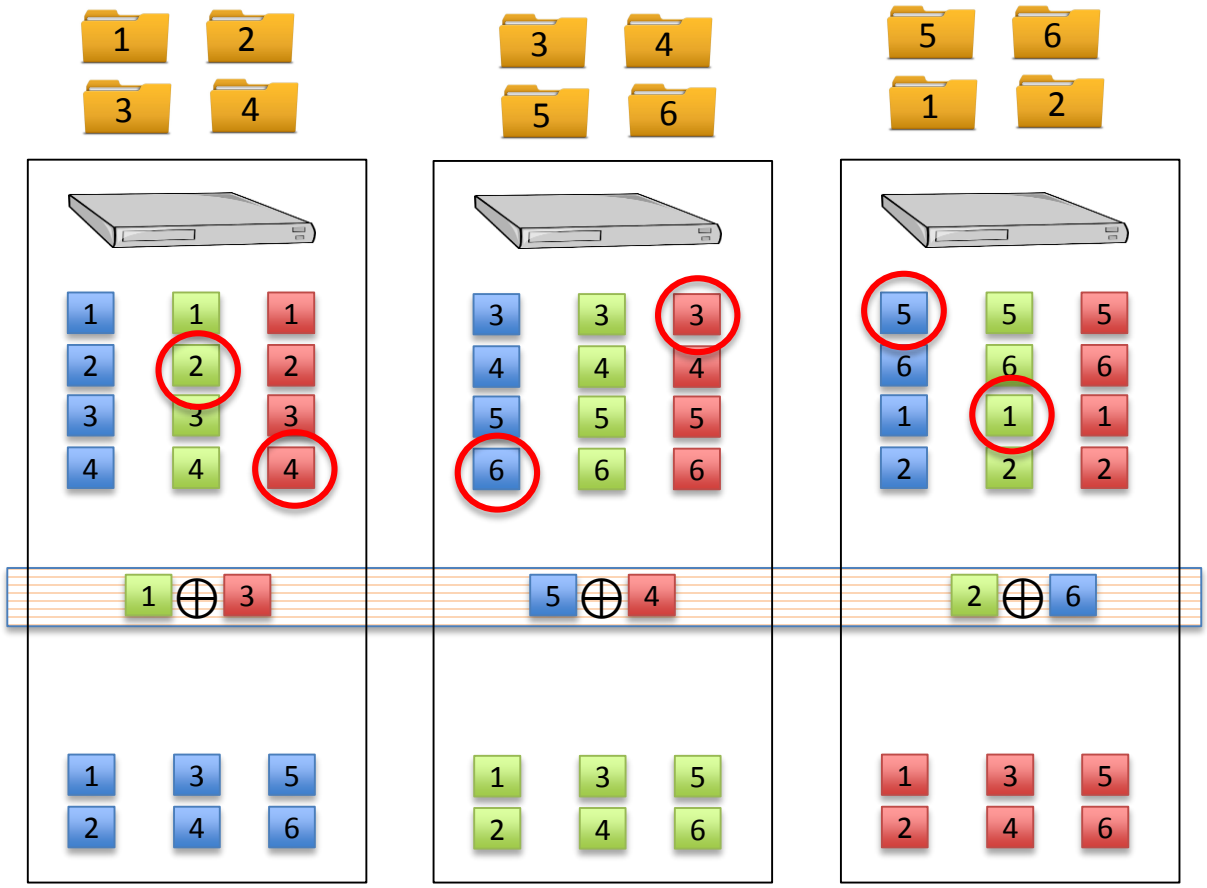



**Can we do better?**

**Can we get a non-vanishing gain for large K?**

# Coded MapReduce

***N Subfiles, K Servers, Q Keys, Comp. Load r***




**Comm. Load (Uncoded)**

$$L_U(r) = QN(1 - \frac{r}{K})$$

**Comm. Load (Coded)**

$$L_C(r) = QN(1 - \frac{r}{K})\frac{1}{r}$$

**Each Coded (key,value) pairs are useful for two servers**

# Communication Load

***N Subfiles, K Servers, Q Keys, Comp. Load r***

**Comm. Load (Map Reduce)**

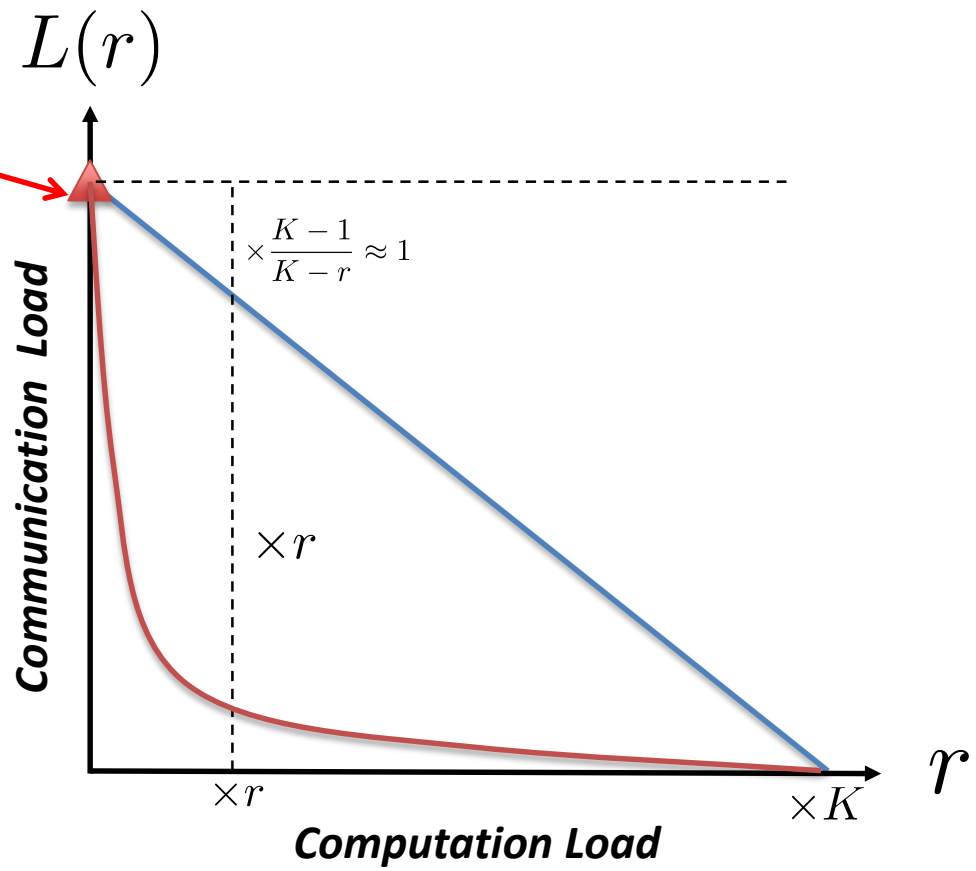
$$L_{MR} = QN(1 - \frac{1}{K})$$

**Comm. Load (Uncoded)**

$$L_U(r) = QN(1 - \frac{r}{K})$$

**Comm. Load (Coded)**

$$L_C(r) = QN(1 - \frac{r}{K})\frac{1}{r}$$




**Communication Load x Computation Load ~ constant**

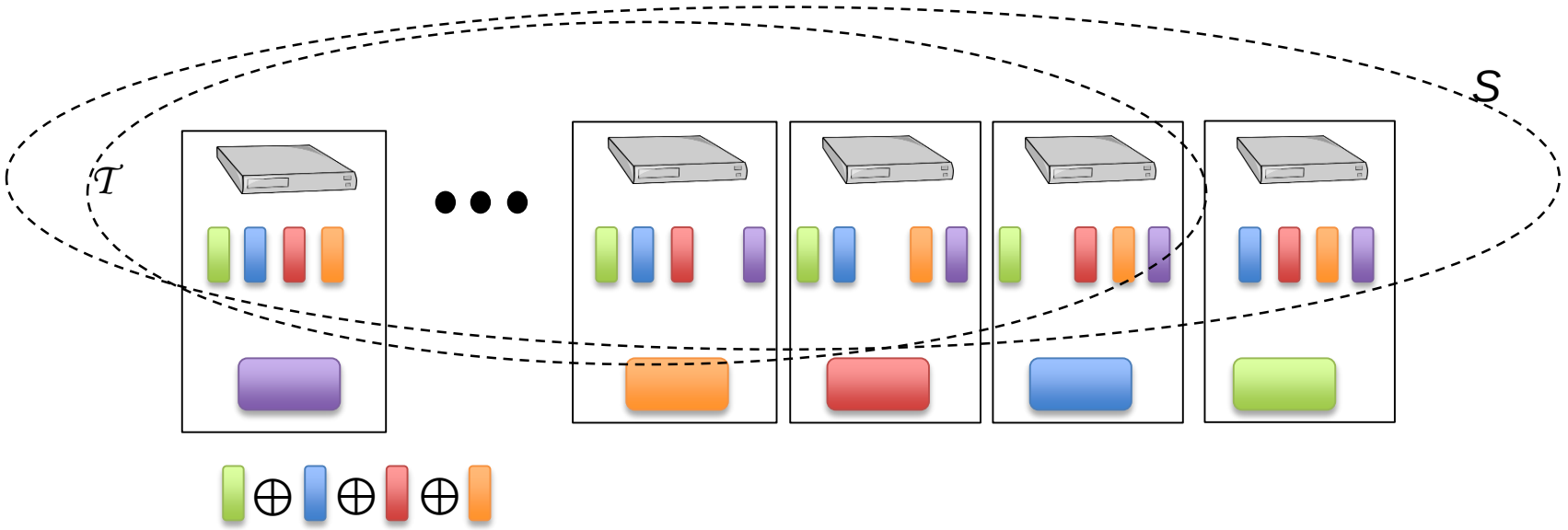
# Proposed Scheme

***N Subfiles, K Servers, Q Keys, Comp. Load r***

**Objective:** Each server can coded intermediate (Key, Value) pairs that are useful for *r* other servers

Need to assign the sub-files such that:

- for every subset $S$ of *r+1* servers,
- and for every subset $\mathcal{T}$ of $S$ with *r* servers,
- Servers in $\mathcal{T}$ share an intermediate (Key, Value) pairs useful for server $S\backslash\mathcal{T}$



# Proposed Scheme

***N Subfiles, K Servers, Q Keys, Comp. Load r***

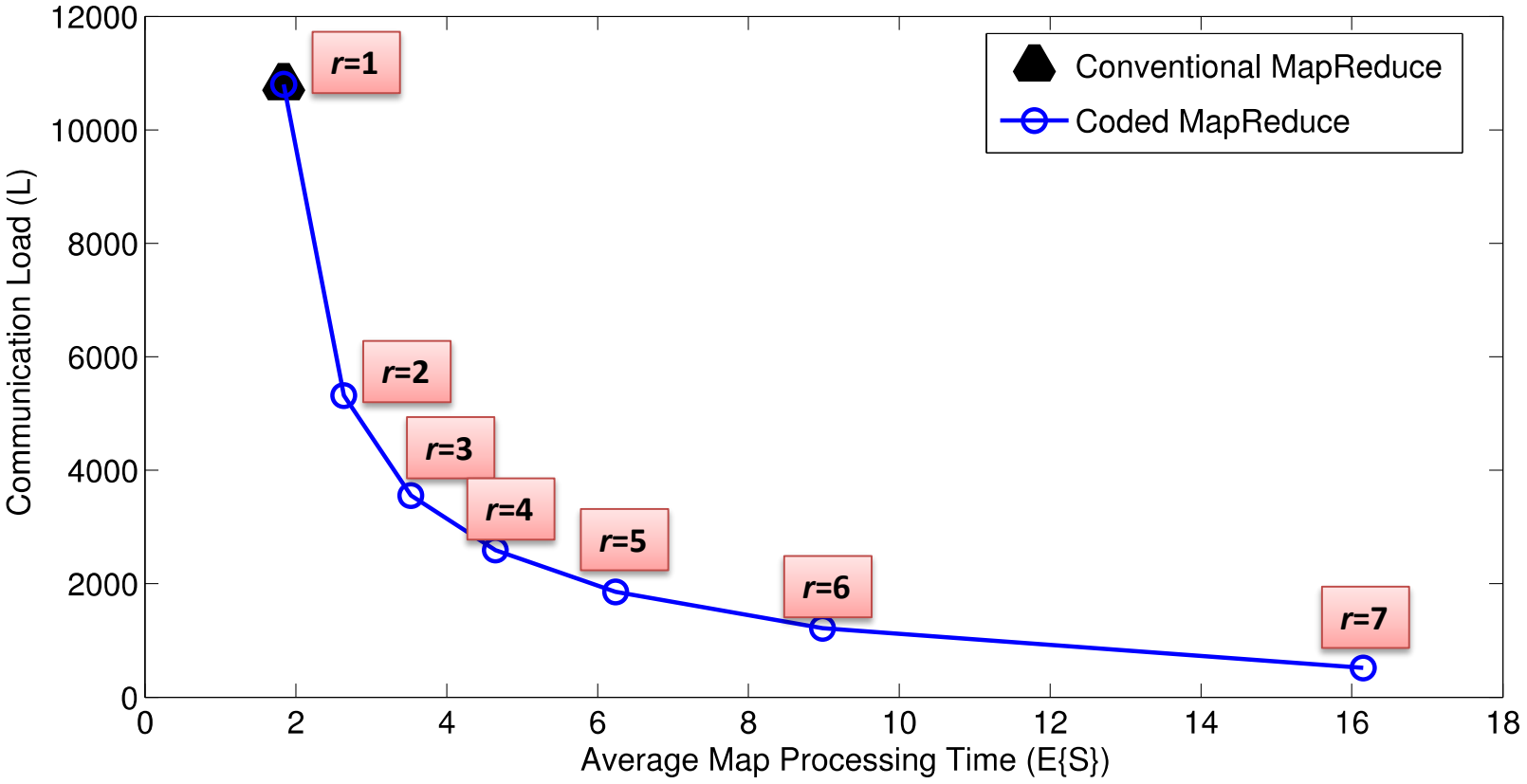
-*N* sub-files: $W_1, W_2, \ldots, W_N$

- Split the set of subfiles to $\binom{N}{r}$ batch of subfiles.

- Each subset of size r of the servers takes a unique batch of subfiles.

# Coded MapReduce-Delay Profile

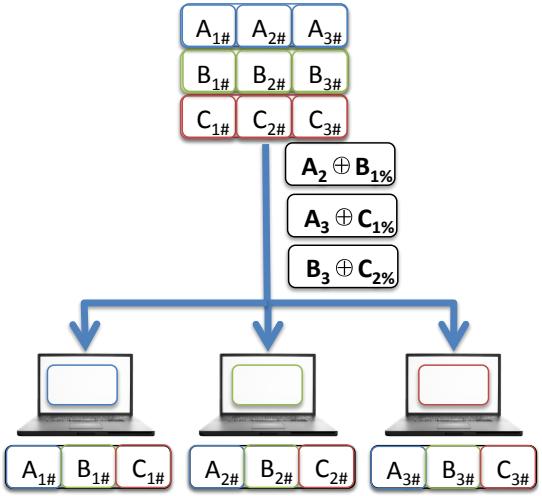
***N=1200 Subfiles, K=10 Servers, Q=10 Keys***



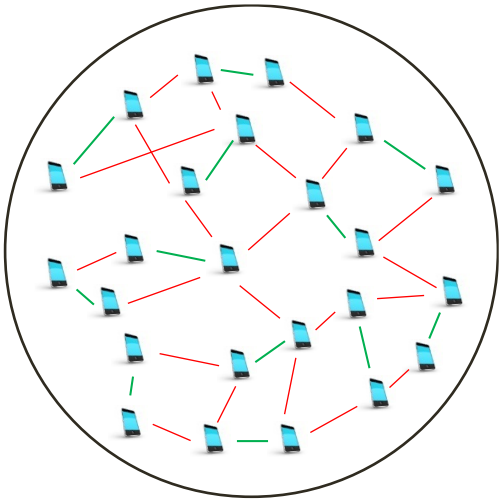

As soon as ***r*** copes of a mapping is done, kills that mapping on other servers.

Map time duration: Exponential random variable

# Connection with Coded Caching




Maddah-Ali-Niessen, 2012



Ji-Caire-Molisch, 2014

- In coded caching, in placement phase, the demand of the each user is not known
- In coded MapReduce, in job assignment, the server which reduces a key is known!

# Why it works!

***N Subfiles, K Servers, Q Keys, Comp. Load r***

Key Idea:

- When a subfile is assigned to a server, that server computes ***all*** (key,value) pairs for that subfiles.
- This imposes a **symmetry** to the problem.

# Can We Do Better?

**Theorem:**

The proposed scheme is optimum within a constant factor in rate.

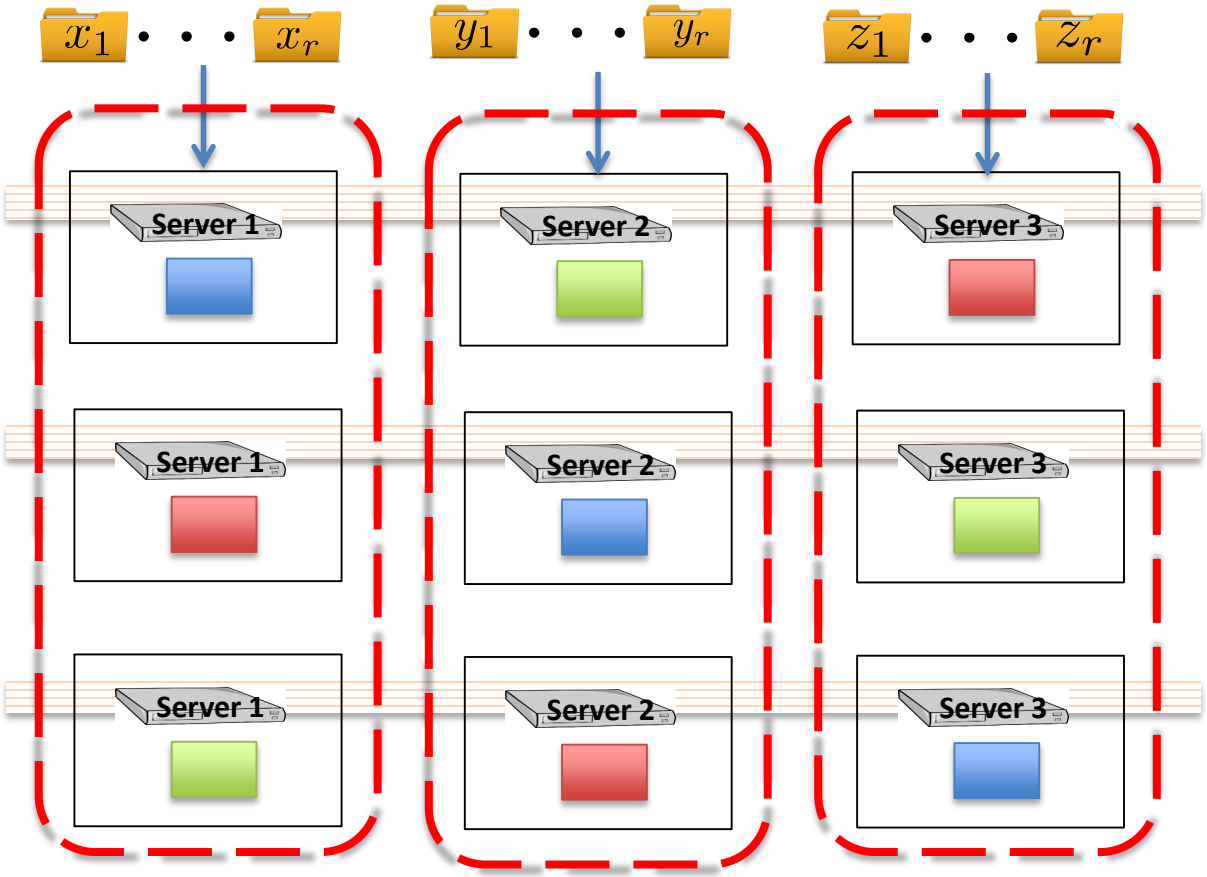
$$\alpha L_C(r) \leq L^*(r) \leq L_C(r)$$

**Comm. Load (Coded)**

$$L_C(r) = QN(1 - \frac{r}{K})\frac{1}{r}$$

# Outer Bound

***N=3 Subfiles, K=3 Servers, Q=3 Keys, Comp. Load r***




$$L^*(r) \geq \frac{1}{2}Q(N-r)$$

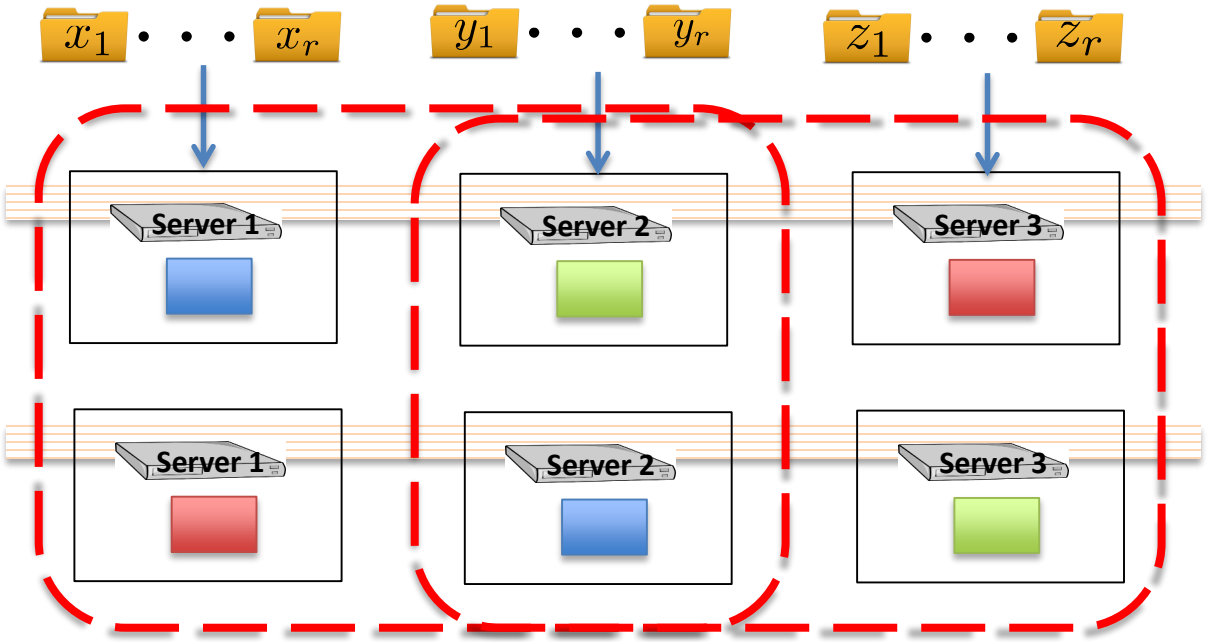
$$R_2 + R_3 + R_2' + R_3' + R_2'' + R_3'' + rQ \geq NQ$$

$$R_1 + R_3 + R_1' + R_3' + R_1'' + R_3'' + rQ \geq NQ$$

$$R_1 + R_2 + R_1' + R_2' + R_1'' + R_2'' + rQ \geq NQ$$

# Outer Bound

***N=3 Subfiles, K=3 Servers, Q=3 Keys, Comp. Load r***




$$L^*(r) \geq \frac{1}{2} Q(N - r)$$

$$L^*(r) \geq \frac{3}{2} Q(N - 2r)$$

$$R_3 + R_3' + 2rQ \geq NQ$$

$$R_1 + R_1' + 2rQ \geq NQ$$

$$R_2 + R_2' + 2rQ \geq NQ$$

# Conclusion

- Communication-Computation tradeoff is of great interests and challenging
- Coded MapReduce provides a near optimal framework for trading "computing" with "communication" in distributed computing
- Communication load x Computation load is approximately constant
- Many future directions:
  - Impact of Coded MapReduce on the overall run-time of MapReduce
  - General server topologies
  - Applications to wireless distributed computing ("wireless Hadoop")
- Papers available on arxiv.