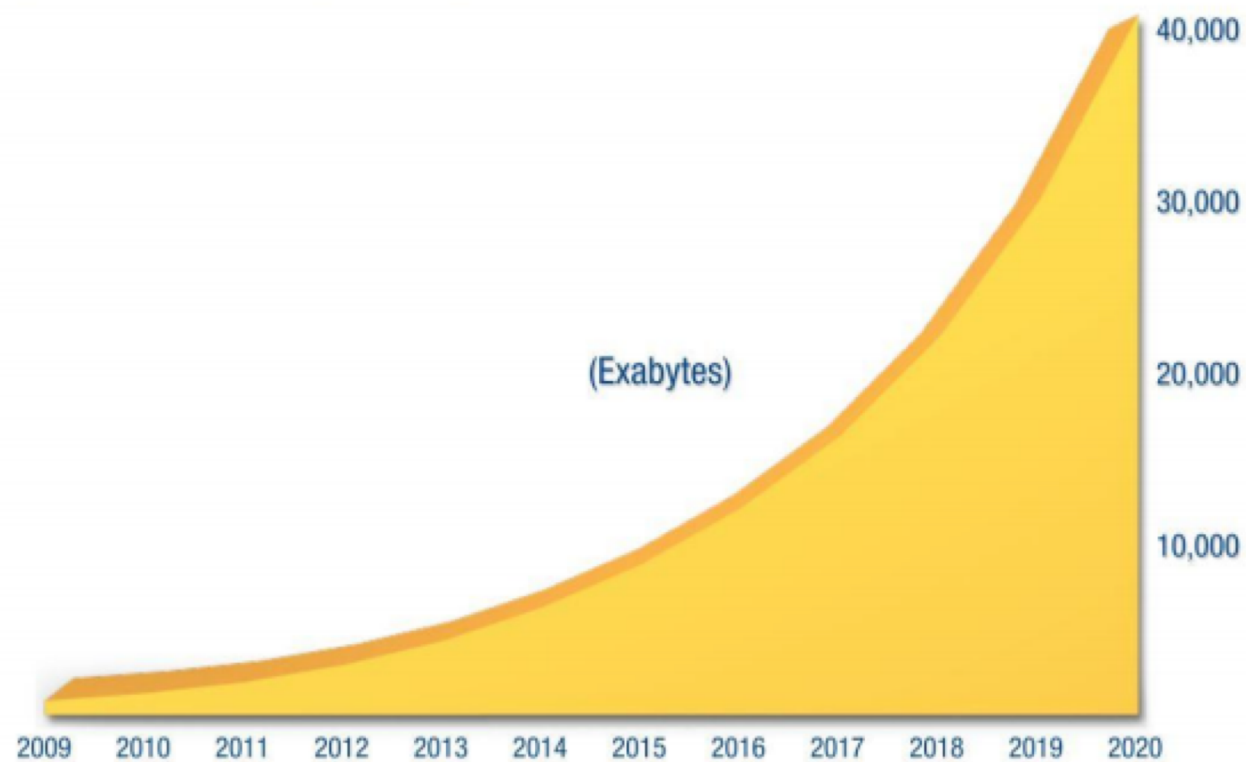


分散コンピューティングへの誤り訂正符号 の応用に関する研究動向

早稲田大学
堀井 俊佑

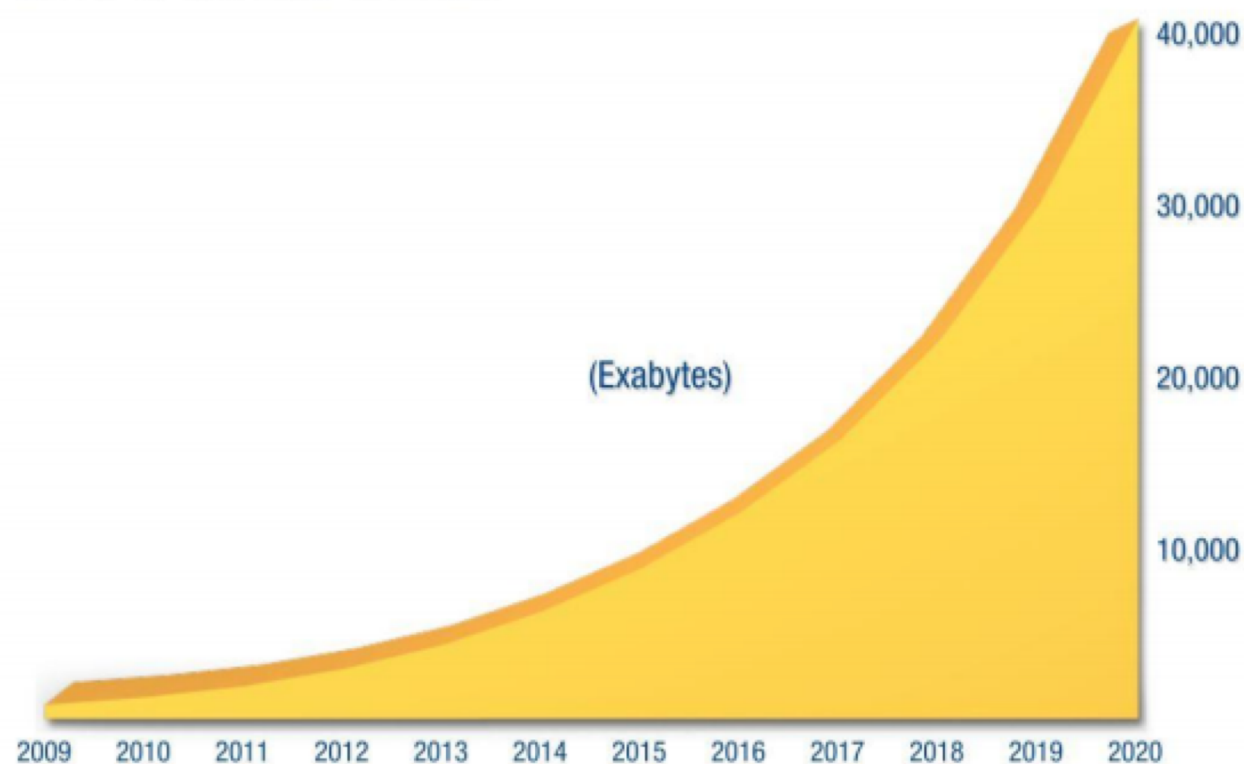
1年間に生成されるデータ量



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

指数関数的に増加

1年間に生成されるデータ量



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

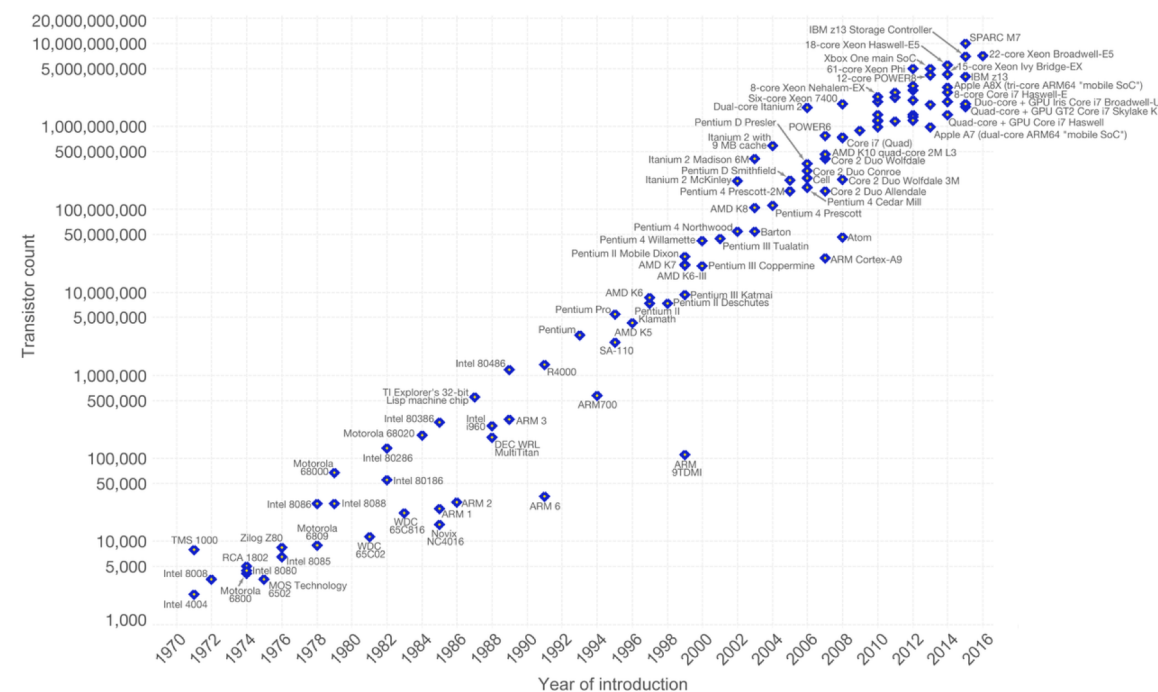
指数関数的に増加

ムーアの法則

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



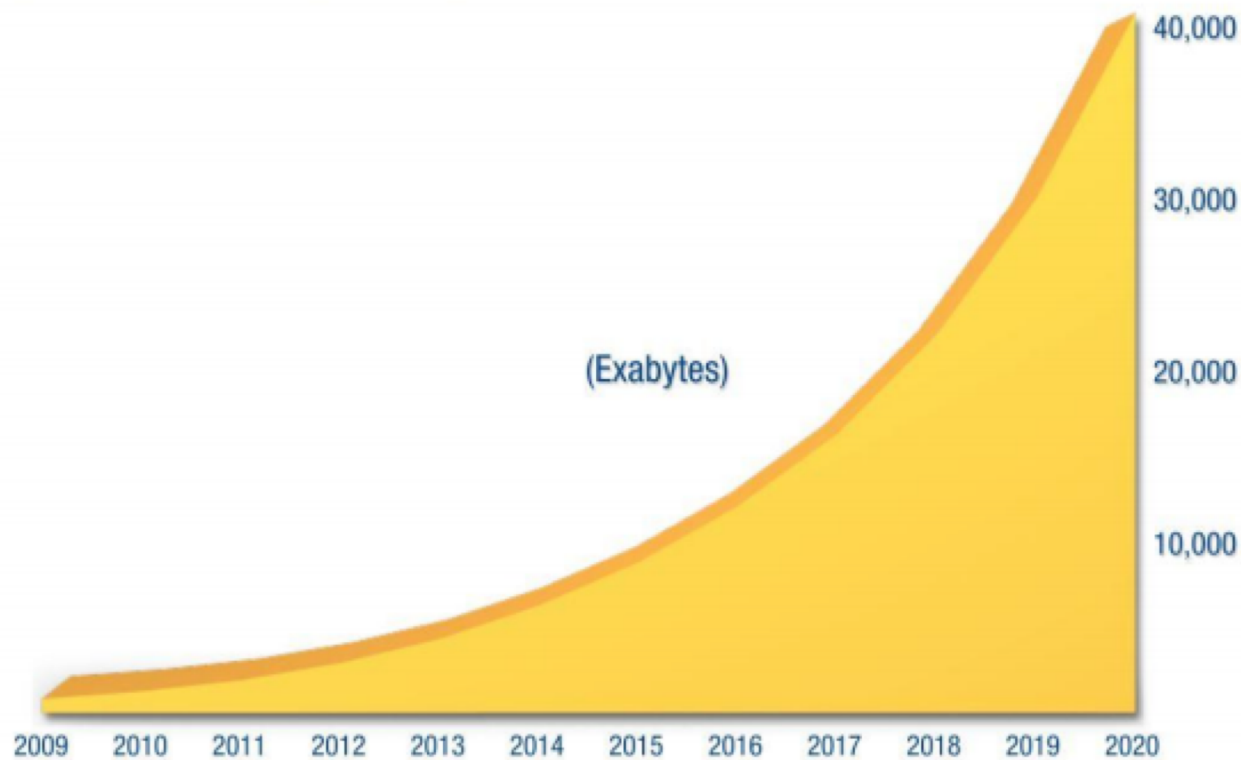
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at [OurWorldinData.org](https://www.ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

2025年頃には収束と予想

1年間に生成されるデータ量



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

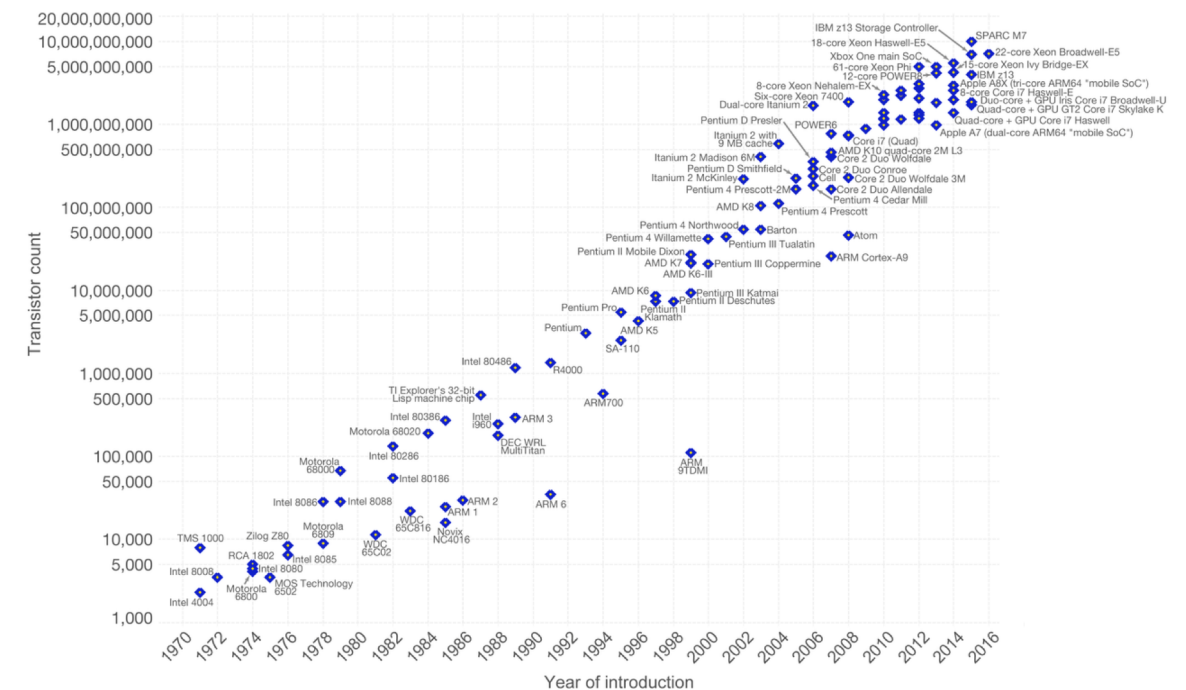
指数関数的に増加

- 複数のコンピュータによる分散計算（並列計算）が必要

ムーアの法則

Moore's Law – The number of transistors on integrated circuit chips (1971-2016) 

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

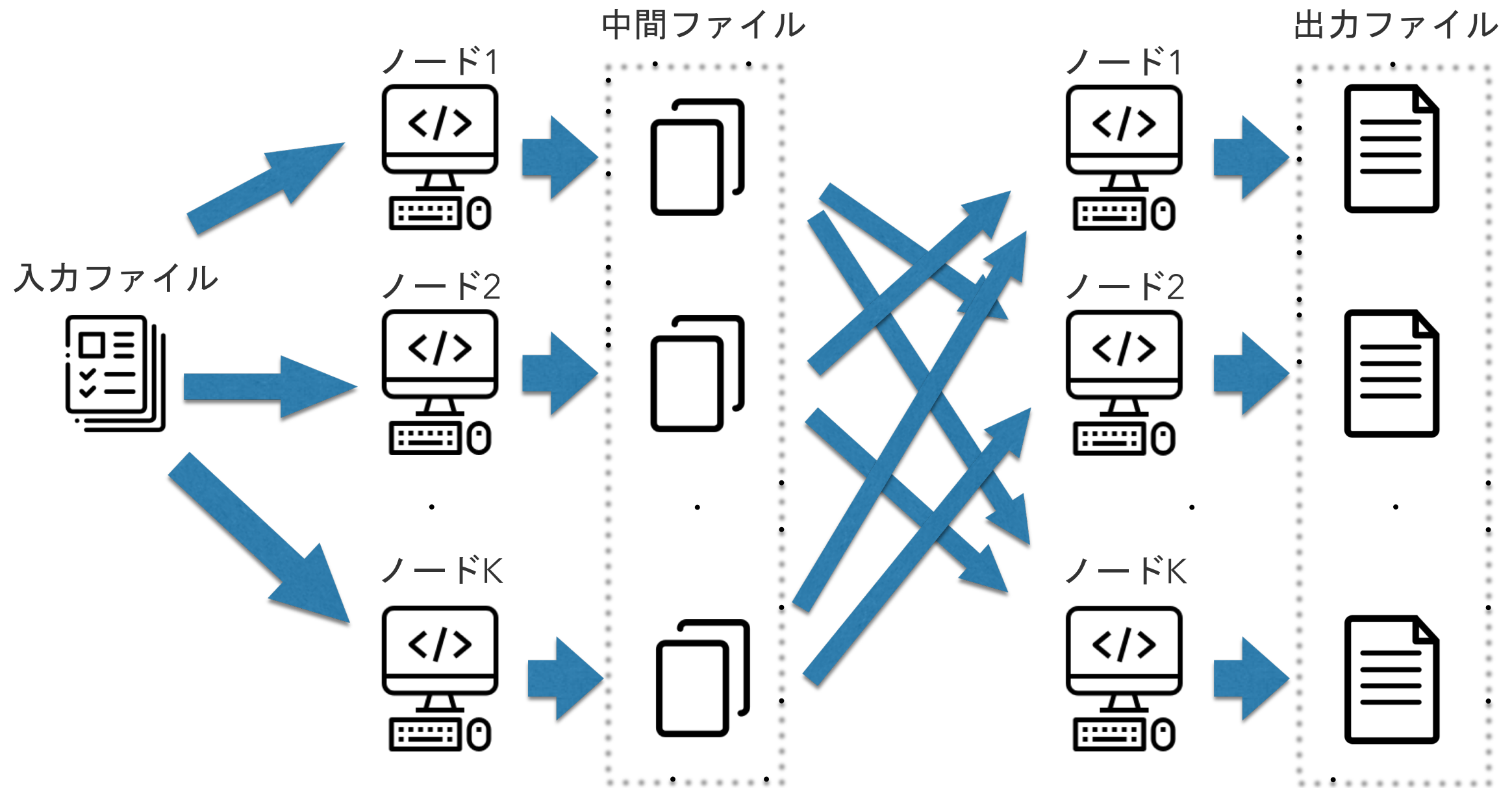
The data visualization is available at [OurWorldinData.org](https://ourworldindata.org/transistor-count). There you find more visualizations and research on this topic.

Licensed under [CC-BY-SA](#) by the author Max Roser.

2025年頃には収束と予想

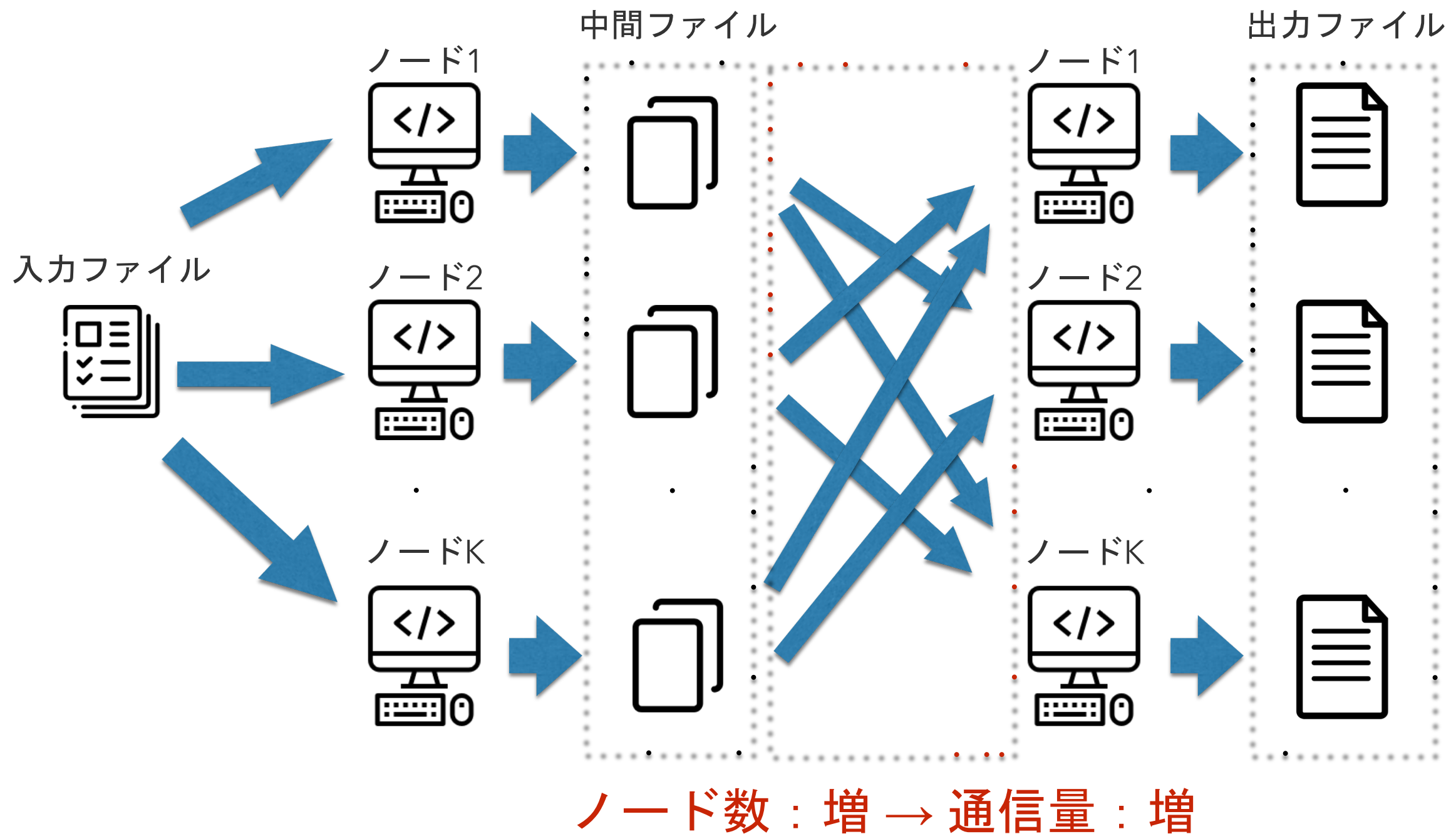
分散計算におけるボトルネック

- Map Reduce モデル



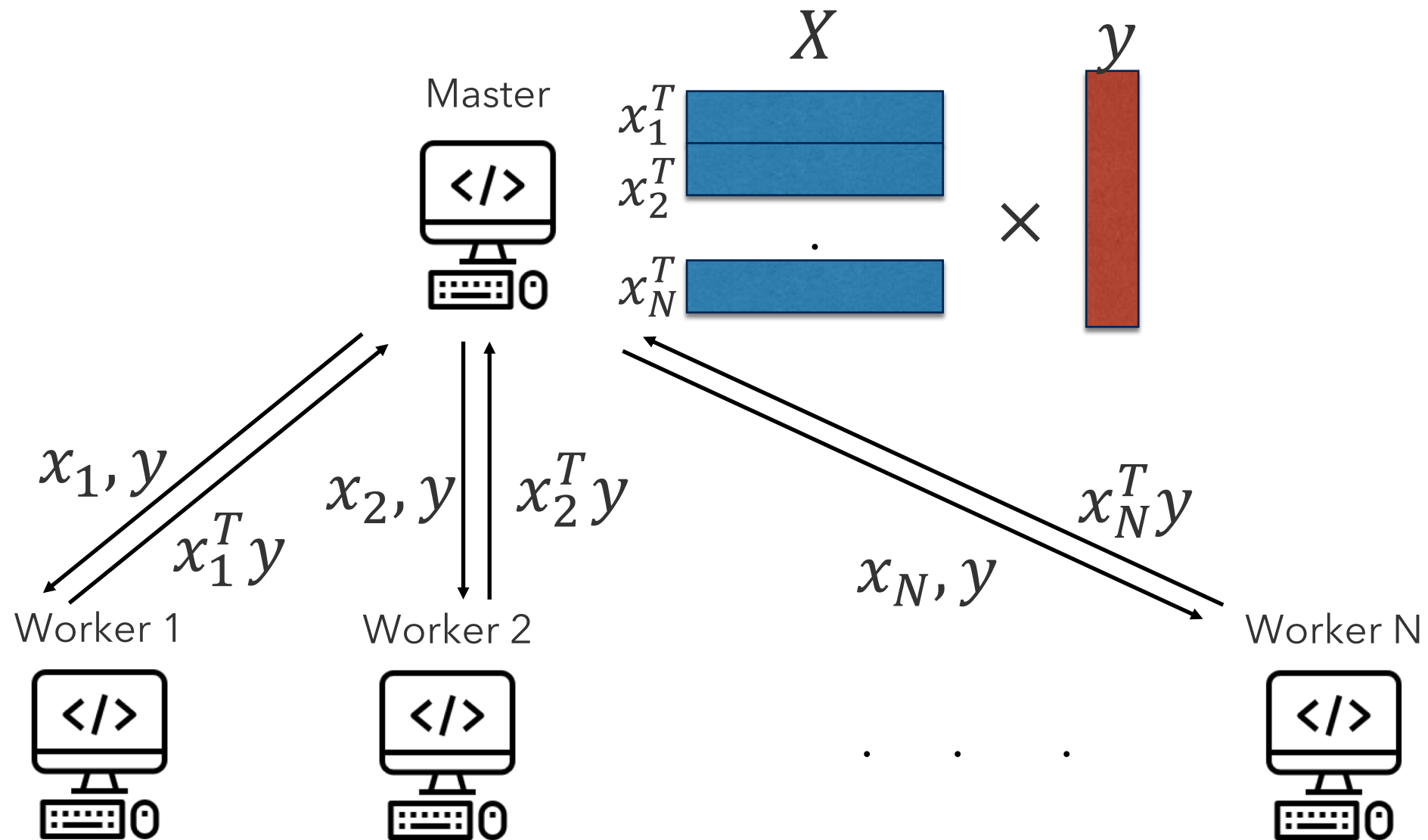
分散計算におけるボトルネック

- Map Reduce モデル



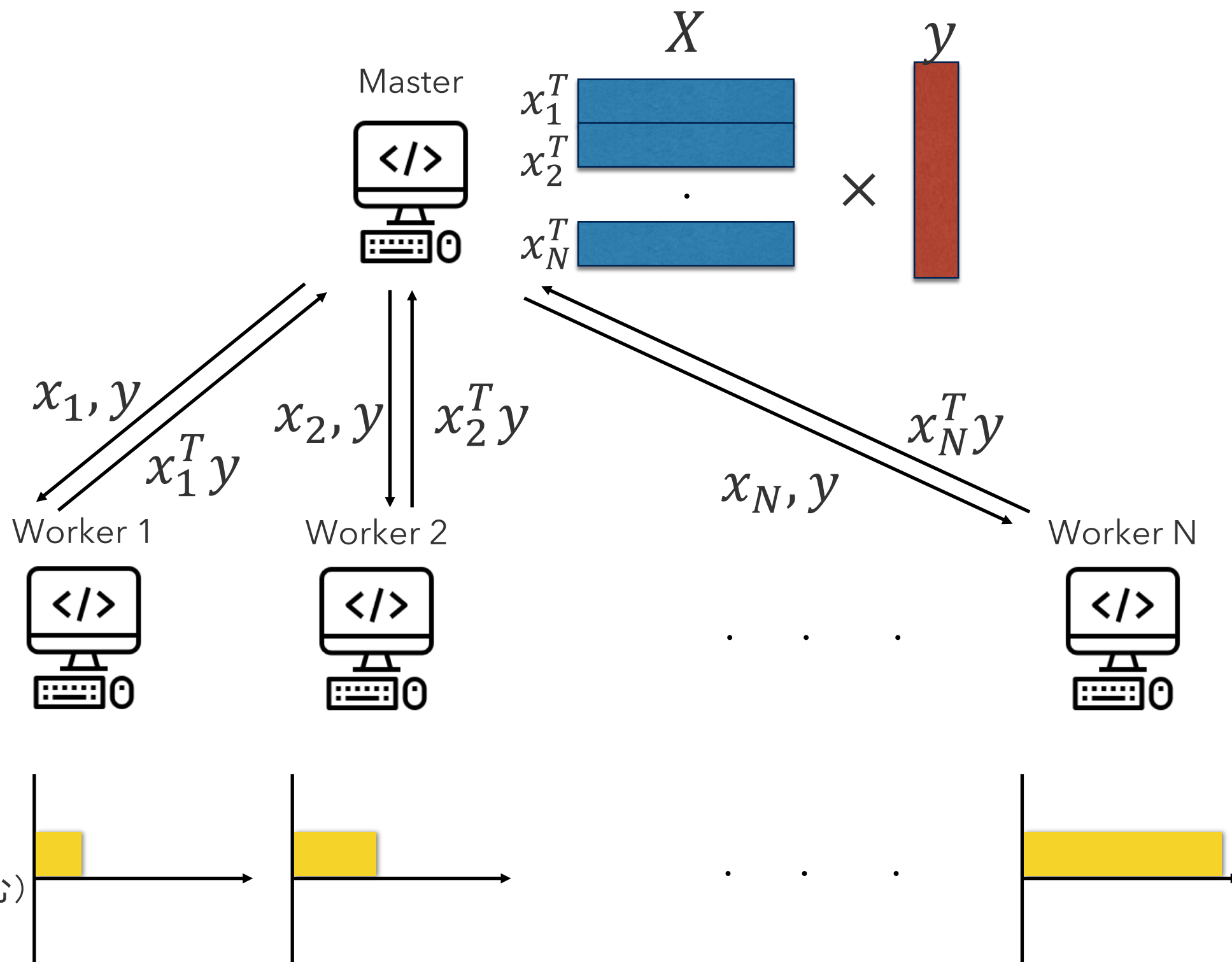
分散計算におけるボトルネック

- Master / Worker モデル



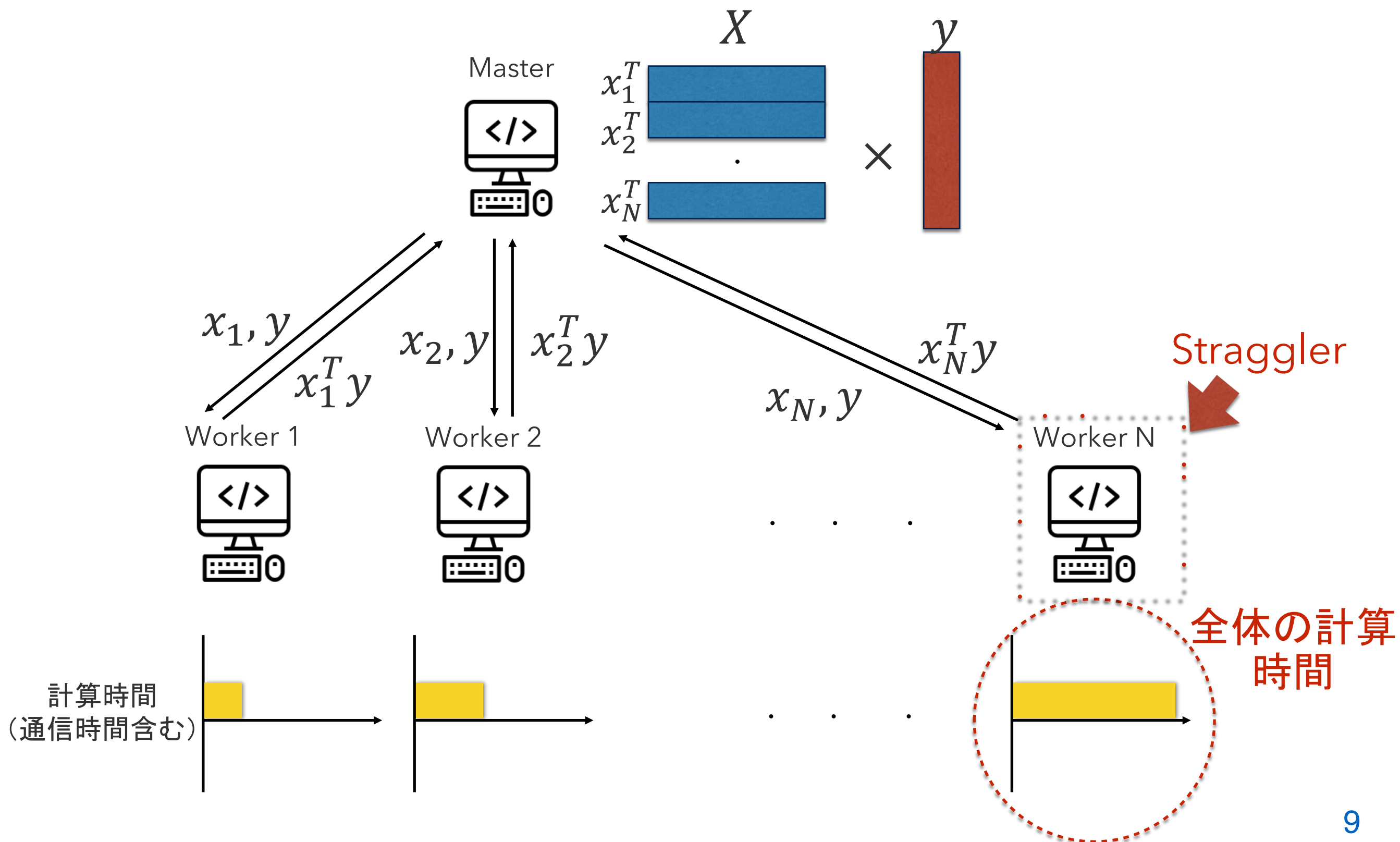
分散計算におけるボトルネック

- Master / Worker モデル



分散計算におけるボトルネック

- Master / Worker モデル



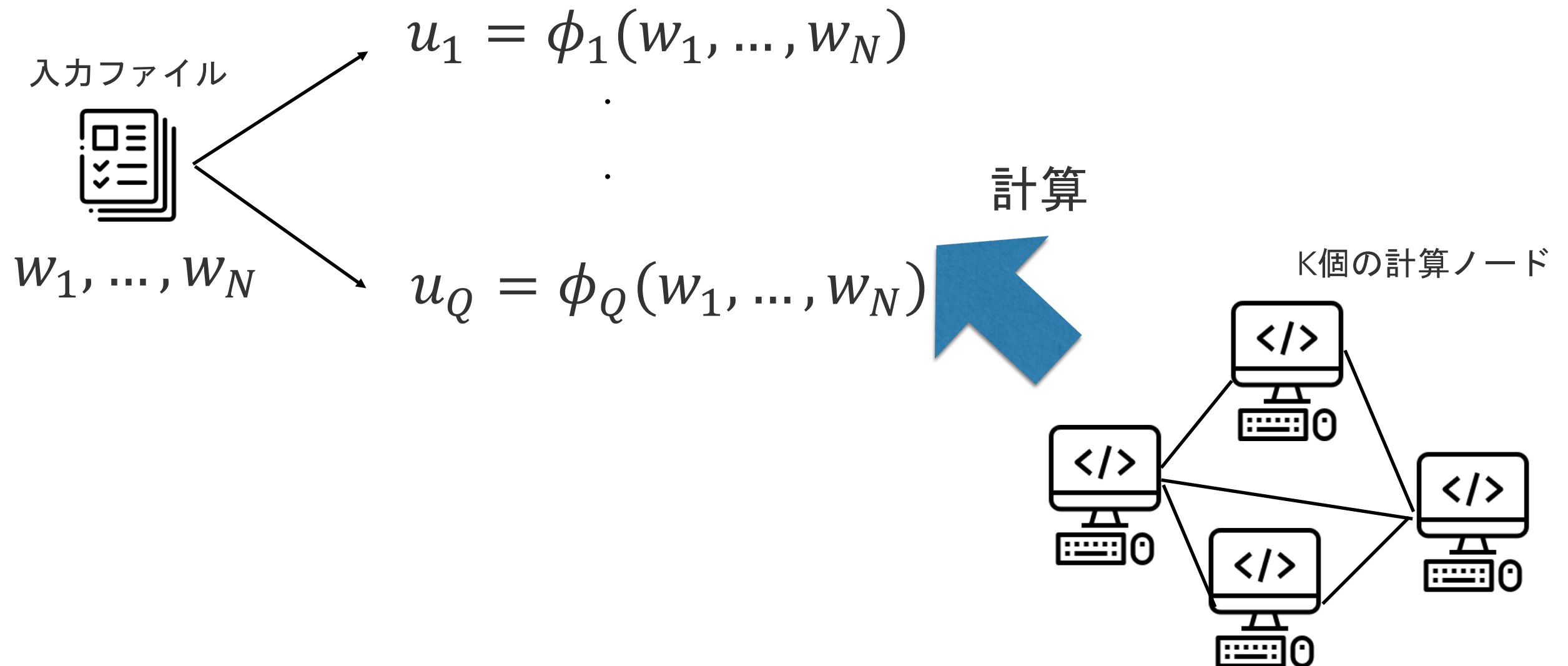
Agenda

- Map-Reduceにおける符号化による通信量削減
- Master / Worker モデルにおける誤り訂正符号を利用したStraggler 効果軽減

Map-Reduceにおける符号化による通信量削減

Map Reduce

- 入力ファイル: $w_1, \dots, w_N \in \mathbb{F}_{2^F}$
- 出力関数: $\phi_1, \dots, \phi_Q, \phi_q: (\mathbb{F}_{2^F})^N \rightarrow \mathbb{F}_{2^B}, q \in \{1, \dots, Q\}$
- 計算ノード数: K



Map Reduce

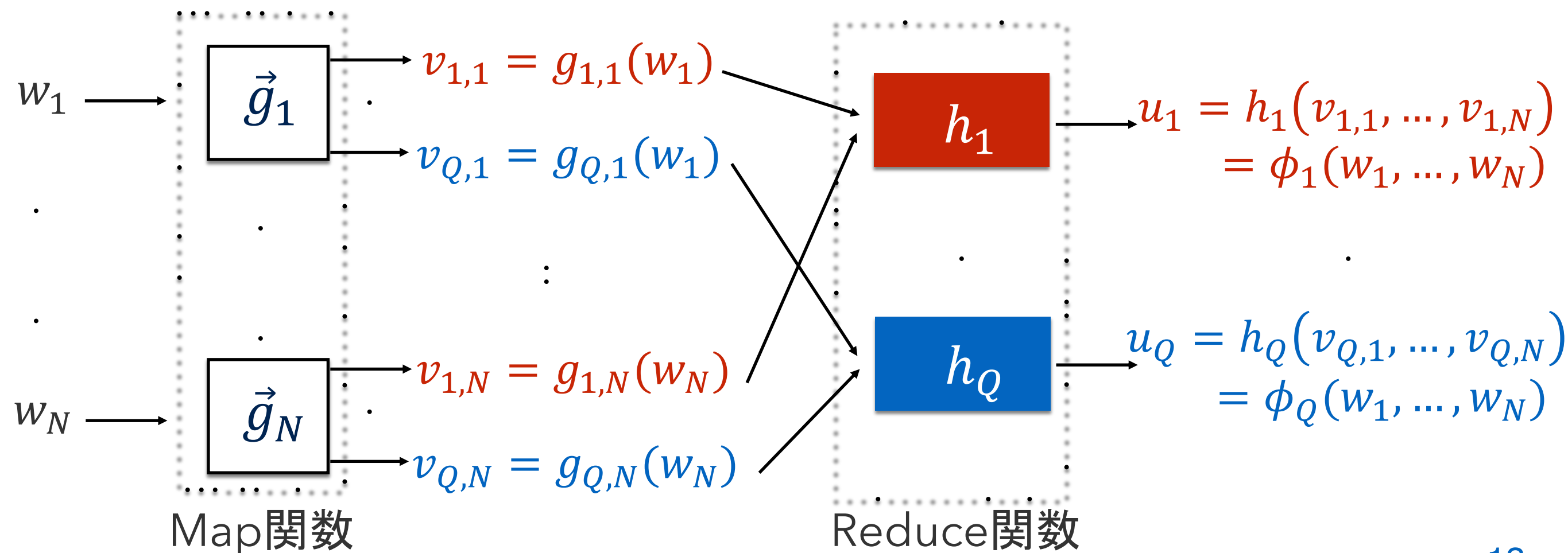
- 仮定 : $\phi_q(w_1, \dots, w_n) = h_q(g_{q,1}(w_1), \dots, g_{q,N}(w_N))$

- Map 関数 : $\vec{g}_n = (g_{1,n}, \dots, g_{q,n}) : \mathbb{F}_{2^F} \rightarrow (\mathbb{F}_{2^T})^Q$

中間ファイル : $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$

- Reduce 関数 : $h_q : (\mathbb{F}_{2^T})^N \rightarrow \mathbb{F}_{2^B}$

$$u_q = h_q(v_{q,1}, \dots, v_{q,N})$$



Map Reduce

- 仮定 : $\phi_q(w_1, \dots, w_n) = h_q(g_{q,1}(w_1), \dots, g_{q,N}(w_N))$

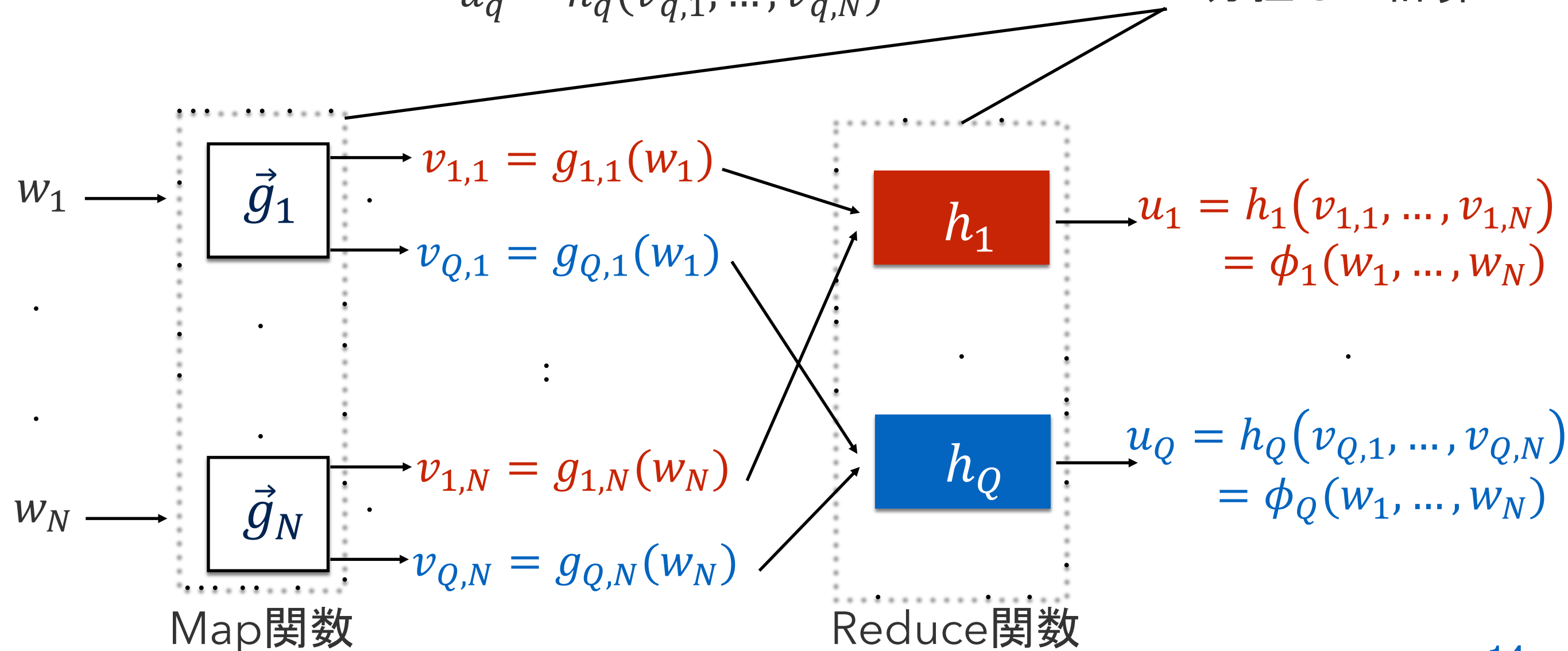
- Map 関数 : $\vec{g}_n = (g_{1,n}, \dots, g_{q,n}) : \mathbb{F}_{2^F} \rightarrow (\mathbb{F}_{2^T})^Q$

中間ファイル : $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$

- Reduce 関数 : $h_q : (\mathbb{F}_{2^T})^N \rightarrow \mathbb{F}_{2^B}$

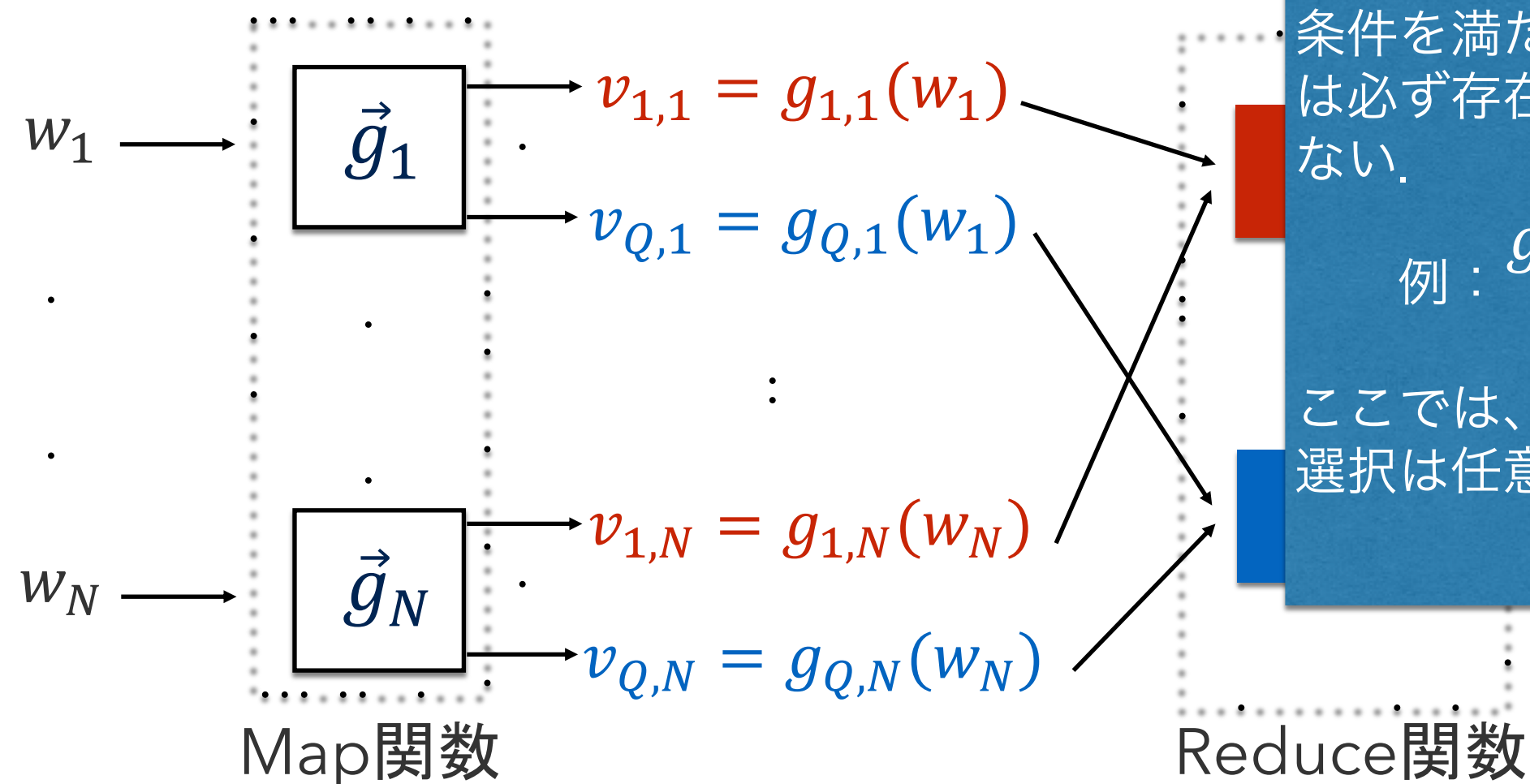
$$u_q = h_q(v_{q,1}, \dots, v_{q,N})$$

K個のノードで
分担して計算



Map Reduce

- 仮定 : $\phi_q(w_1, \dots, w_n) = h_q(g_{q,1}(w_1), \dots, g_{q,N}(w_N))$
 - Map 関数 : $\vec{g}_n = (g_{1,n}, \dots, g_{q,n}) : \mathbb{F}_{2^F} \rightarrow (\mathbb{F}_{2^T})^Q$
中間ファイル : $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$
 - Reduce 関数 : $h_q : (\mathbb{F}_{2^T})^N \rightarrow \mathbb{F}_{2^B}$
 $u_q = h_q(v_{q,1}, \dots, v_{q,N})$



注：
条件を満たすMap関数、Reduce関数は必ず存在し、一意に決まるとは限らない。

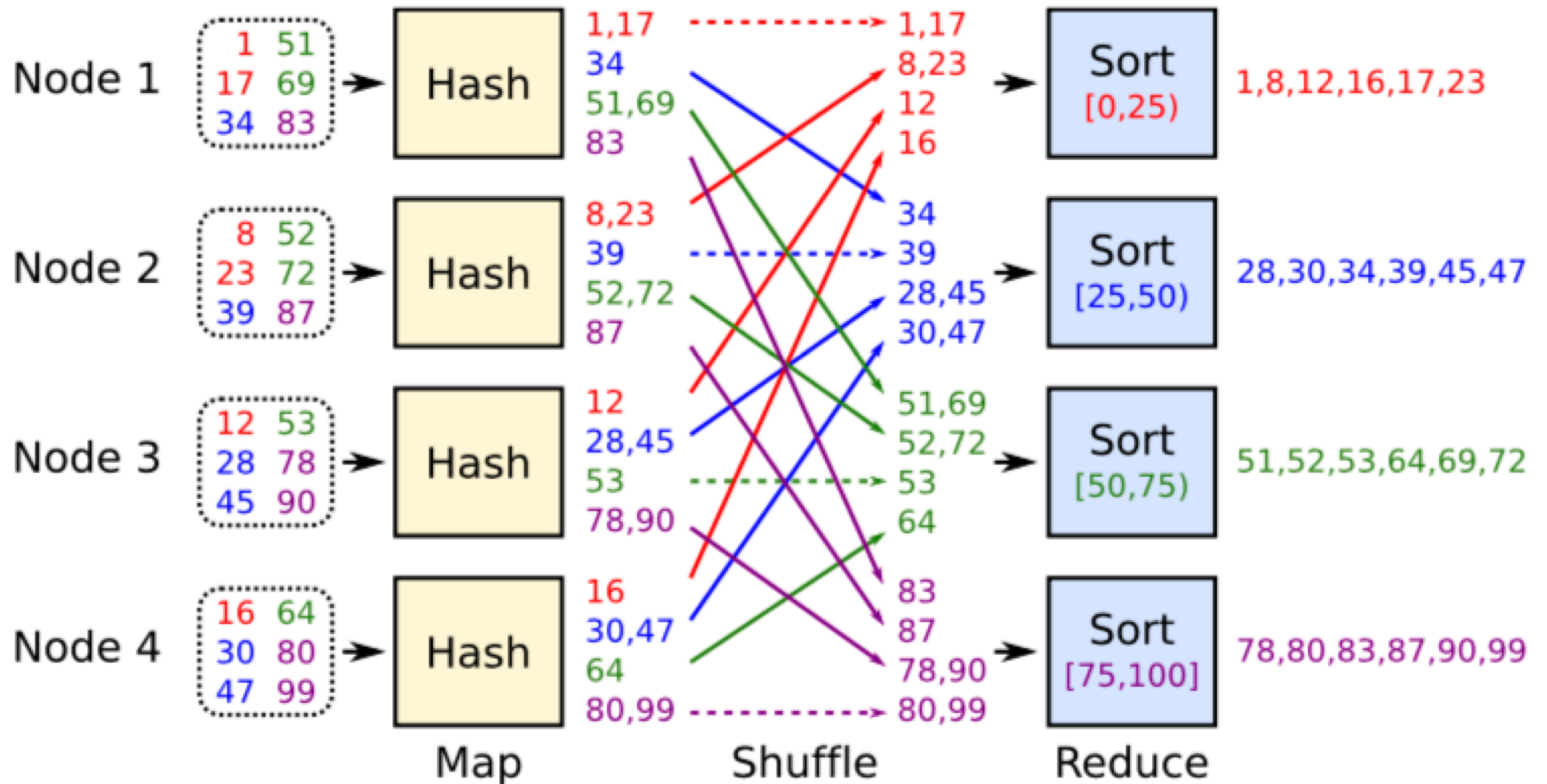
例 : $g_{q,n}(w_n) = w_n$

$\phi_q = h_q$

ここでは、Map関数、Reduce関数の選択は任意のものとする。

Map Reduce

- 例：TeraSort

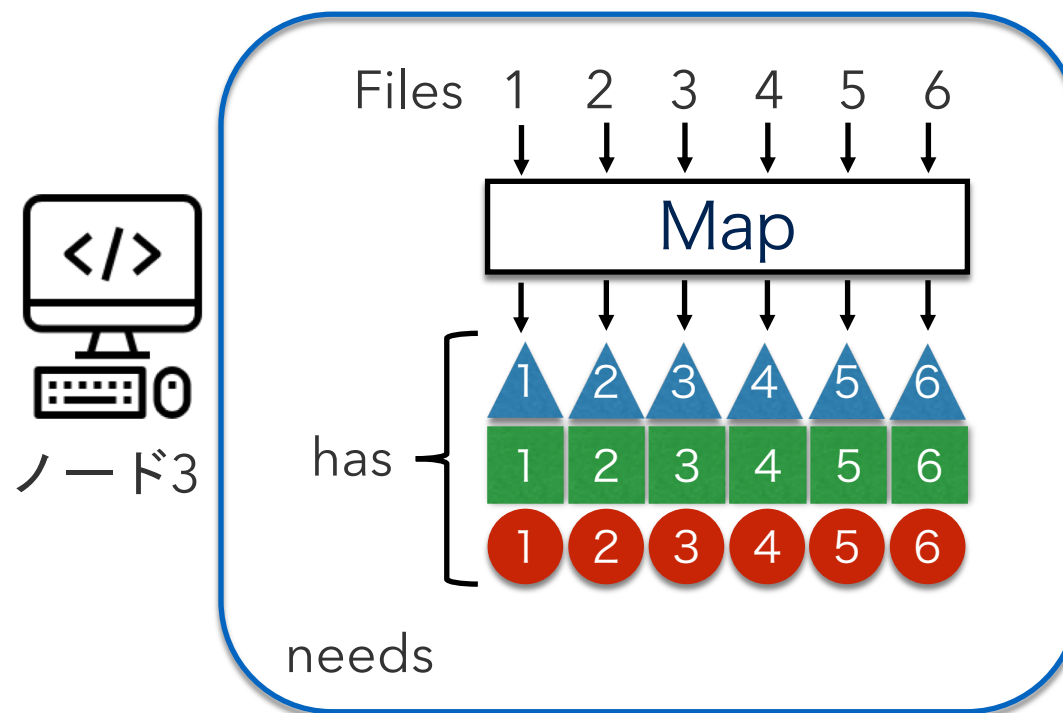


引用：

Li et al.: Fundamental Tradeoff Between Computation and Communication in Distributed Computing

計算負荷と通信負荷のトレードオフ

- 例 : $N = 6, Q = 3, K = 3$



計算負荷

$$= \frac{\text{Map関数の計算回数}}{N}$$

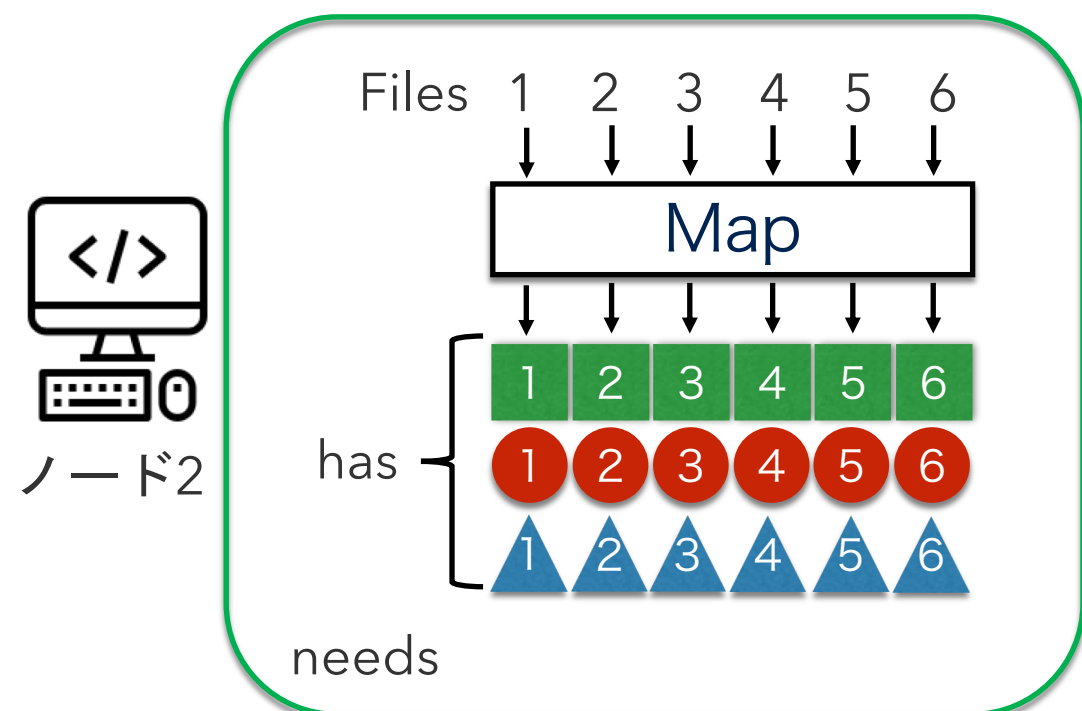
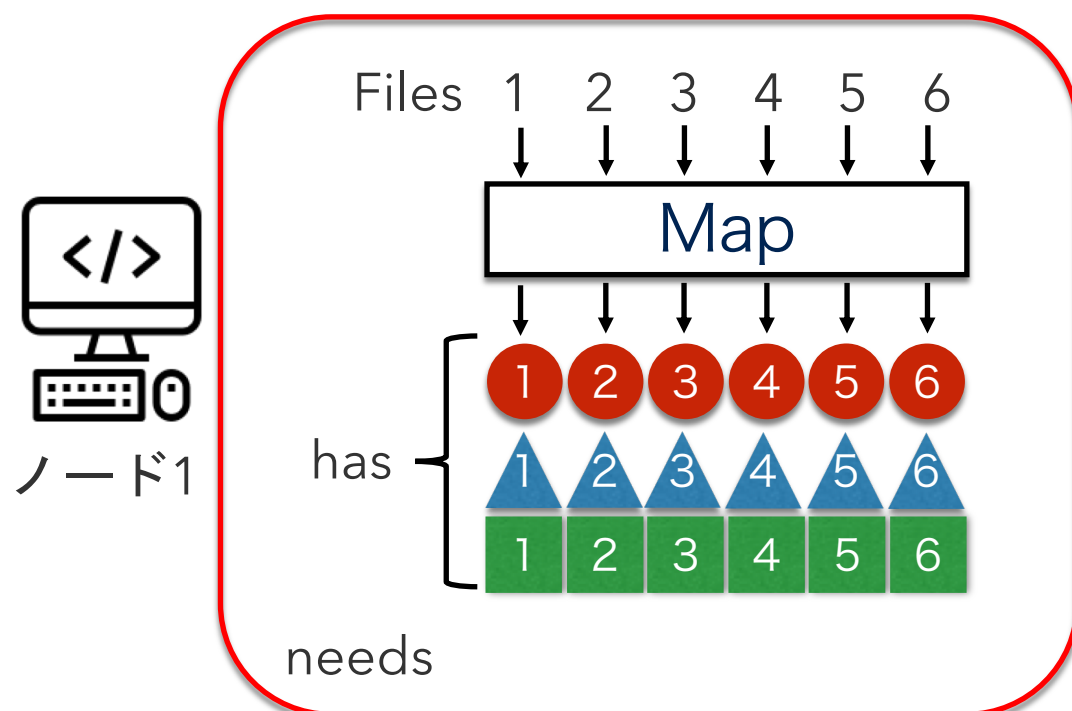
$$= \frac{18}{6} = 3$$

通信負荷

$$= \frac{\text{各ノードからの送信}bit\text{数の和}}{QNT}$$

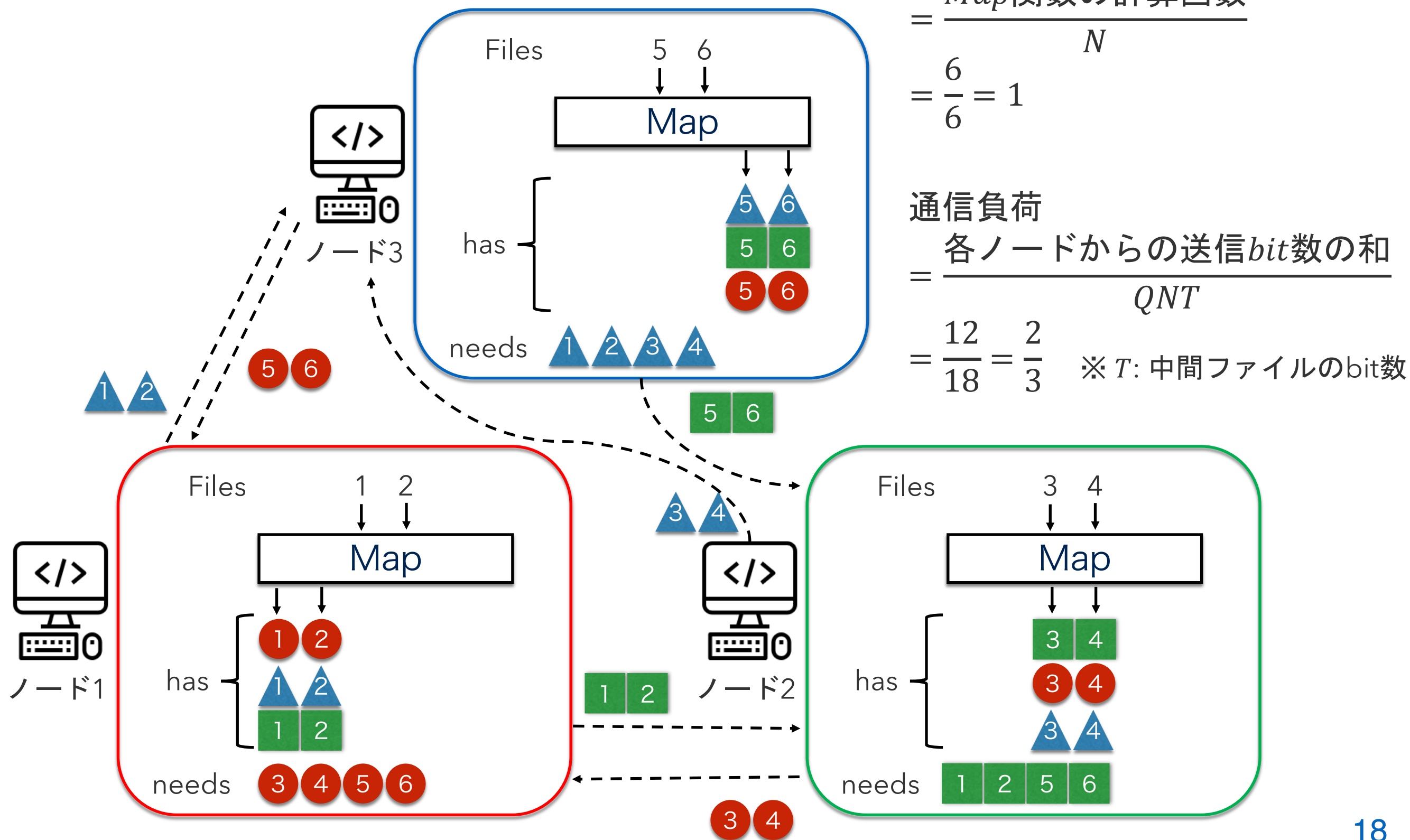
$$= 0$$

※ T : 中間ファイルのbit数



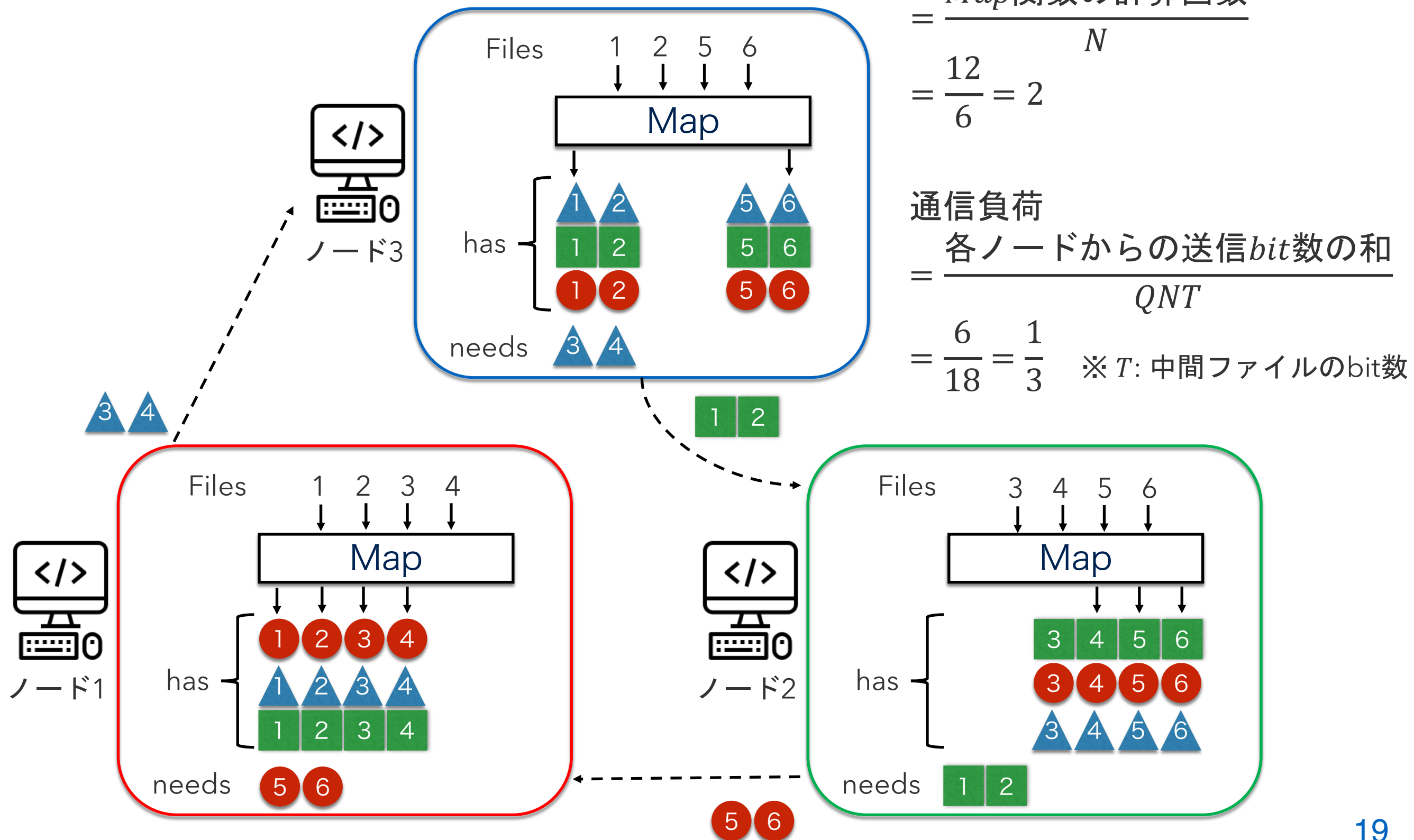
計算負荷と通信負荷のトレードオフ

- 例 : $N = 6, Q = 3, K = 3$



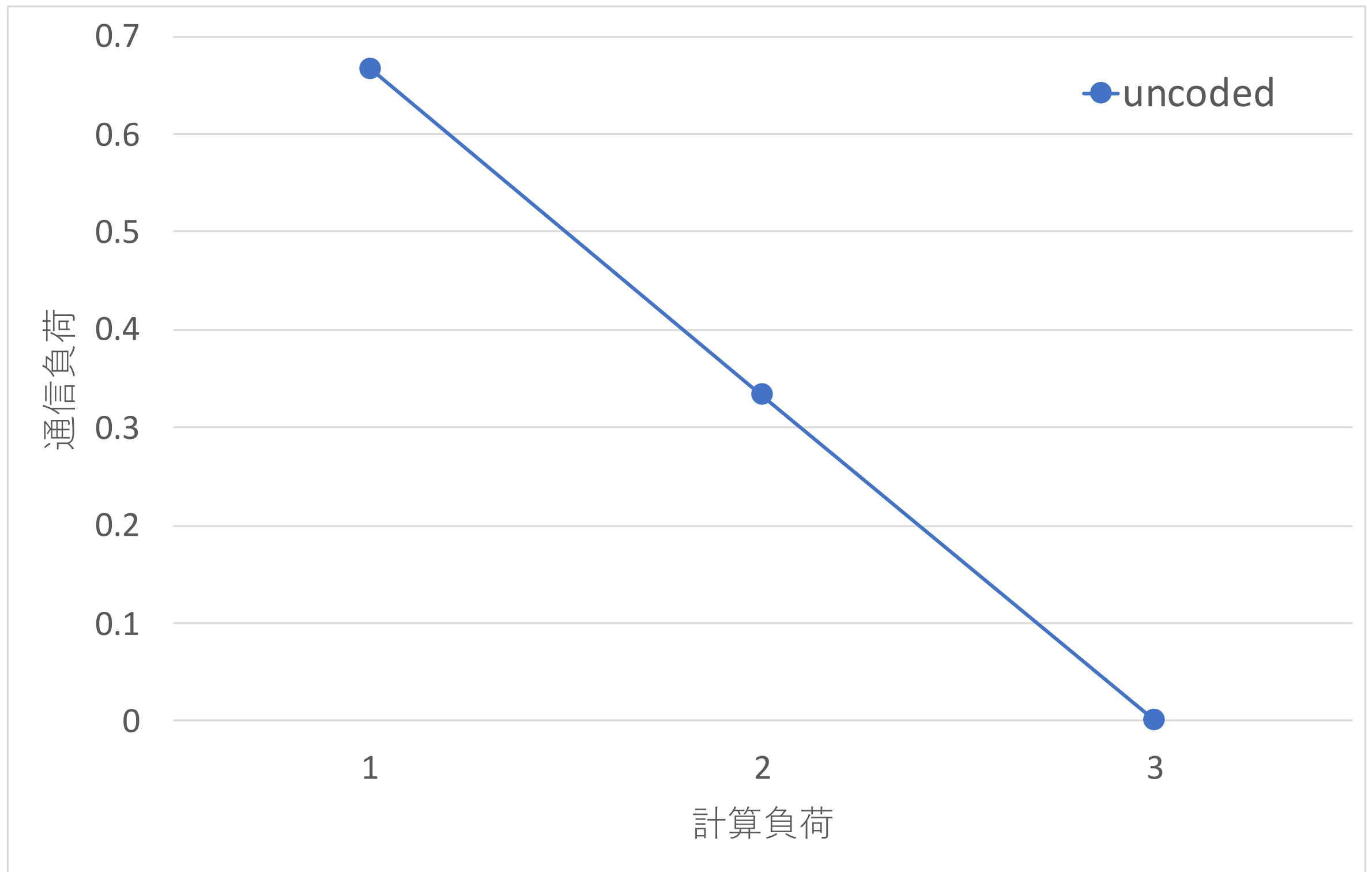
計算負荷と通信負荷のトレードオフ

- 例 : $N = 6, Q = 3, K = 3$



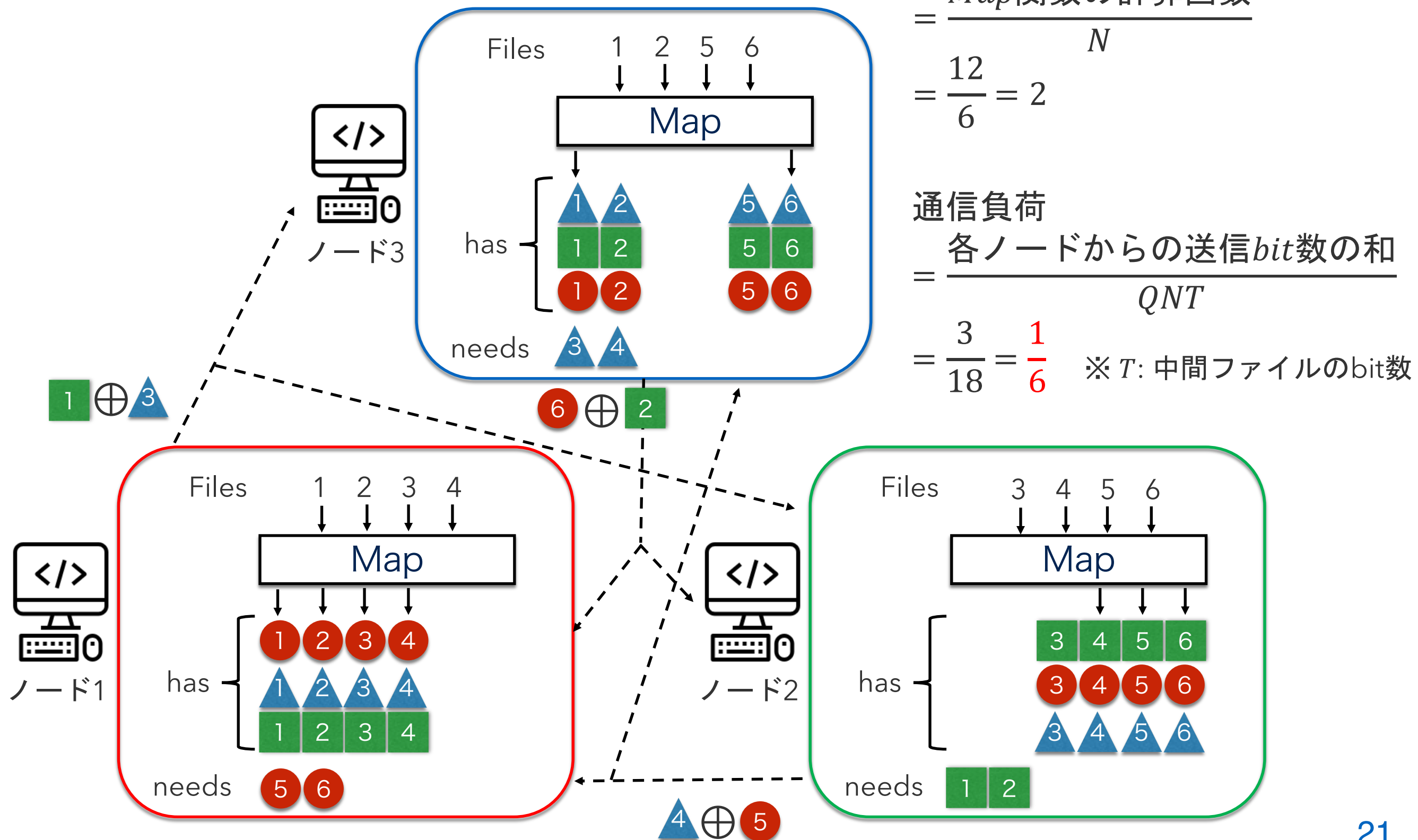
計算負荷と通信負荷のトレードオフ

- 符号化無しの場合



符号化による通信負荷低減

- 例 : $N = 6, Q = 3, K = 3$



計算負荷

$$= \frac{\text{Map関数の計算回数}}{N}$$

$$= \frac{12}{6} = 2$$

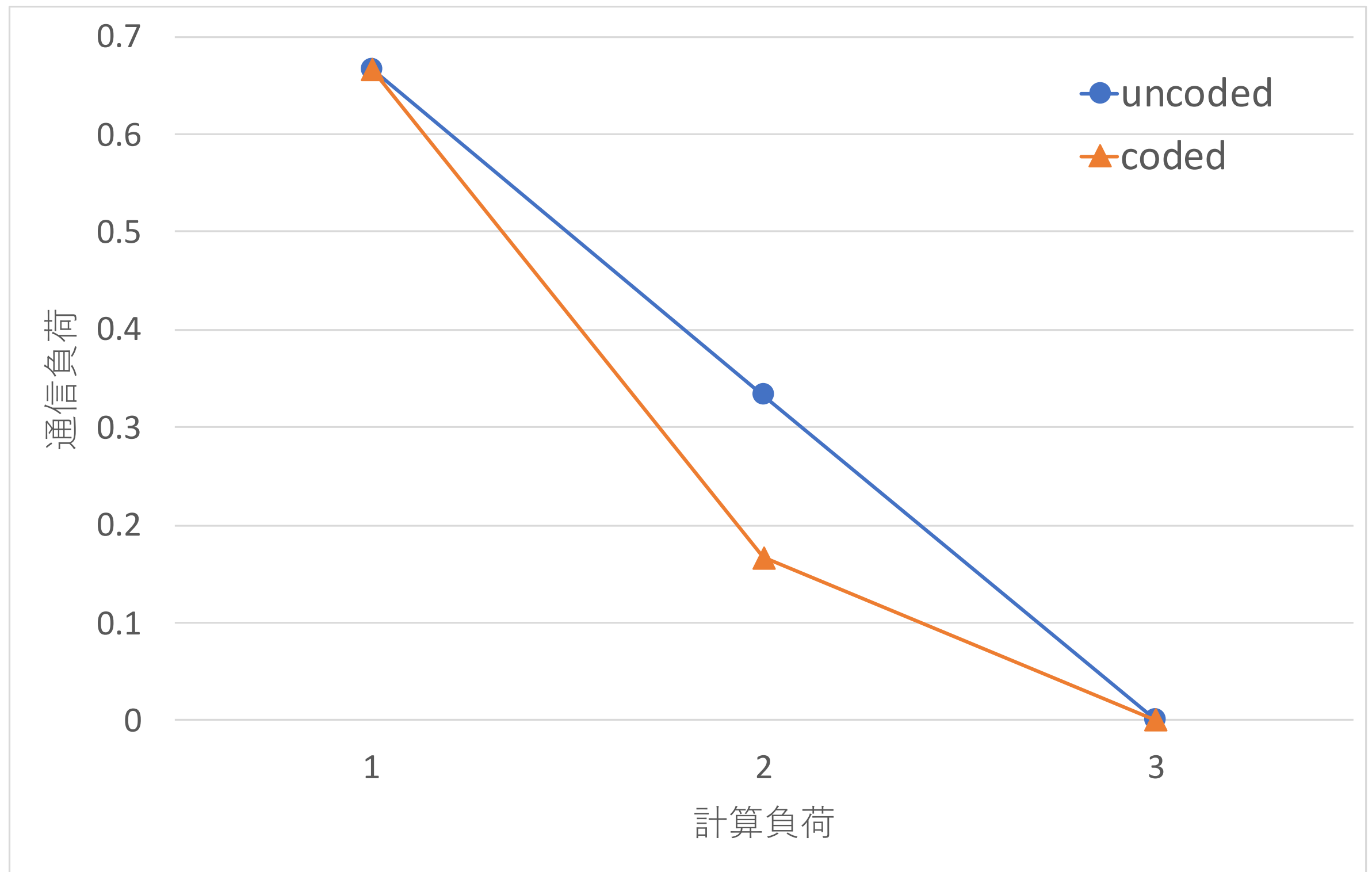
通信負荷

$$= \frac{\text{各ノードからの送信} \textit{bit} \text{数の和}}{QNT}$$

$$= \frac{3}{18} = \frac{1}{6} \quad \text{※ } T: \text{中間ファイルの} \textit{bit} \text{数}$$

符号化による通信負荷低減

- 符号化有りの場合



符号化による通信負荷低減

- 定義

- (r, L) がfeasible

$\Leftrightarrow$ 任意の $\delta > 0$ と十分大きな N に対して, $|r - \tilde{r}| \leq \delta$, $|L - \tilde{L}| \leq \delta$ で, すべての $\phi_1, \dots, \phi_Q$ が正しく計算可能となるような計算負荷が $\tilde{r}$ で通信負荷が $\tilde{L}$ の符号化方式が存在する.

- computation-communication function $L^*(r)$:

$$L^*(r) = \inf\{L : (r, L) \text{ is feasible}\}$$

定理 [Li et al. 2018]

T を十分大きく取るとき, $L^*(r)$ は以下で与えられる.

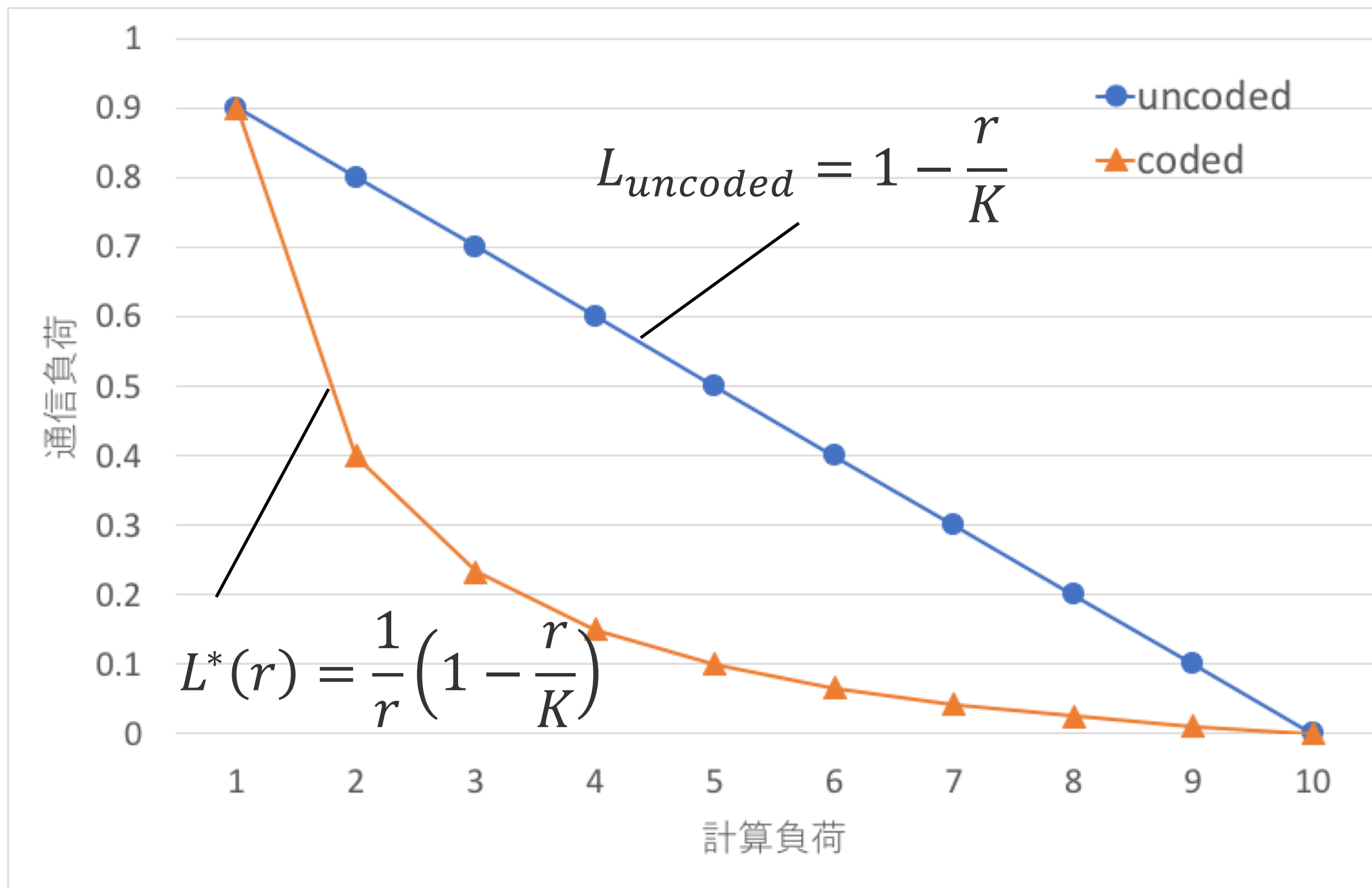
$$L^*(r) = \frac{1}{r} \left(1 - \frac{r}{K}\right), \quad r \in \{1, \dots, K\}$$

整数以外の r については, $\left\{ \left(r, \frac{1}{r} \left(1 - \frac{r}{K}\right) \right) : r \in \{1, \dots, K\} \right\}$ のlower convex envelopeで与えられる.

上記を達成可能な符号化が存在し(Achievability), 上記より良い符号化は存在しない(Converse).

符号化による通信負荷低減

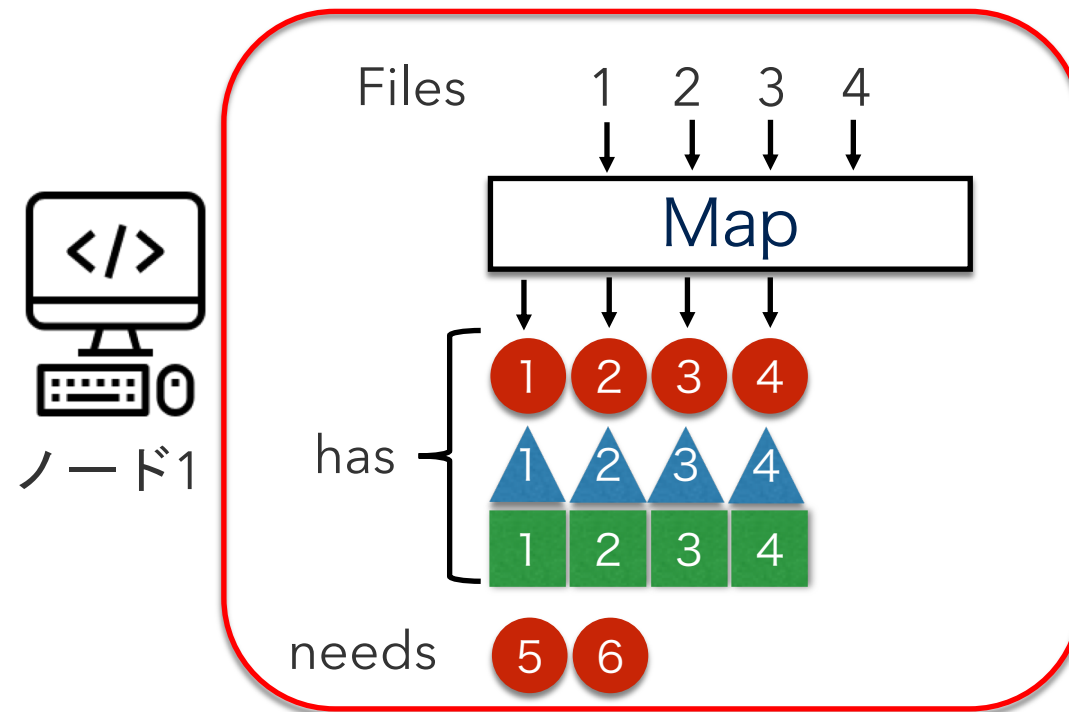
- $K = 10$ の場合



Achievability

- 準備

- $\mathcal{M}_k \subseteq \{w_1, \dots, w_N\}$: ノード k が担当するMap関数の入力ファイル集合
- $\mathcal{W}_k \subseteq \{1, \dots, Q\}$: ノード k が担当するReduce関数のインデックス集合



$$\mathcal{M}_1 = \{w_1, w_2, w_3, w_4\}$$

$$\mathcal{W}_1 = \{1\}$$

仮定 :

$$\bigcup_{k=1, \dots, K} \mathcal{M}_k = \{w_1, \dots, w_N\}$$

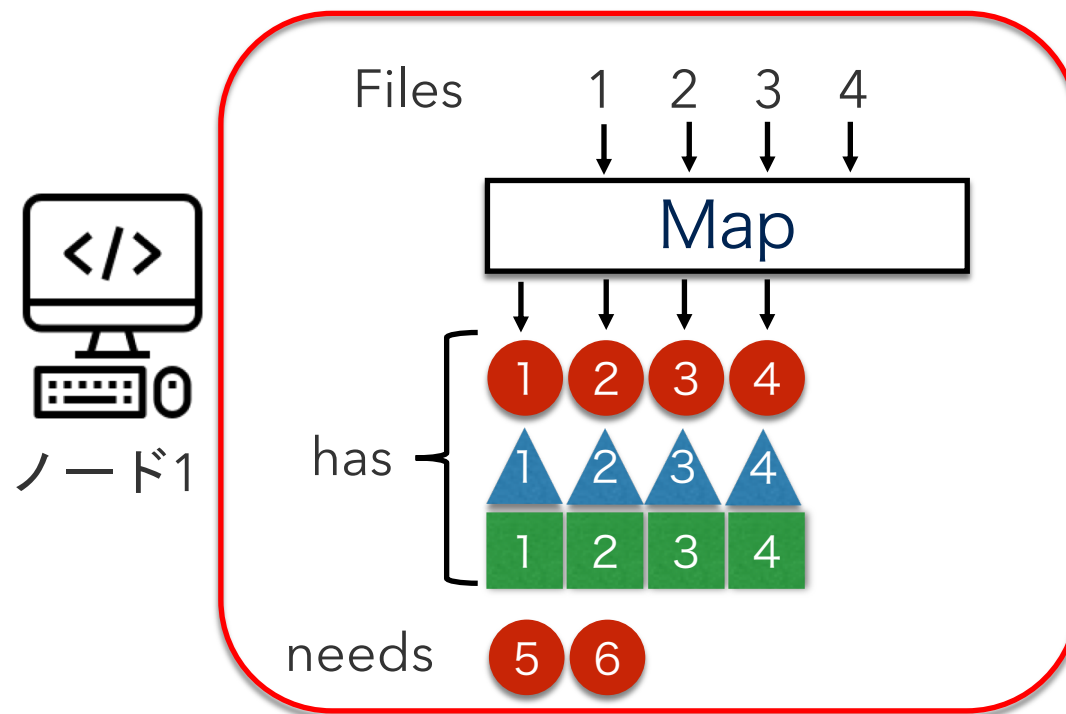
$$|\mathcal{W}_1| = \dots = |\mathcal{W}_K| = \frac{Q}{K}, \mathcal{W}_j \cap \mathcal{W}_k = \emptyset$$

※ 非対称な割当に対するcomputation-communicationトレードオフの導出は open problem

Achievability

- 準備(Cont'd)

- $\mathcal{M}_k \subseteq \{w_1, \dots, w_N\}$: ノード k が担当するMap関数の入力ファイル集合
- $\mathcal{W}_k \subseteq \{1, \dots, Q\}$: ノード k が担当するReduce関数のインデックス集合



$$\mathcal{M}_1 = \{w_1, w_2, w_3, w_4\}$$

$$\mathcal{W}_1 = \{1\}$$

- $\psi_k: (\mathbb{F}_{2^T})^{Q|\mathcal{M}_k|} \rightarrow \mathbb{F}_{2^{\ell_k}}$: ノード k の符号化関数

$$X_k = \psi_k(\{v_{q,n} : q \in \{1, \dots, Q\}, w_n \in \mathcal{M}_k\})$$

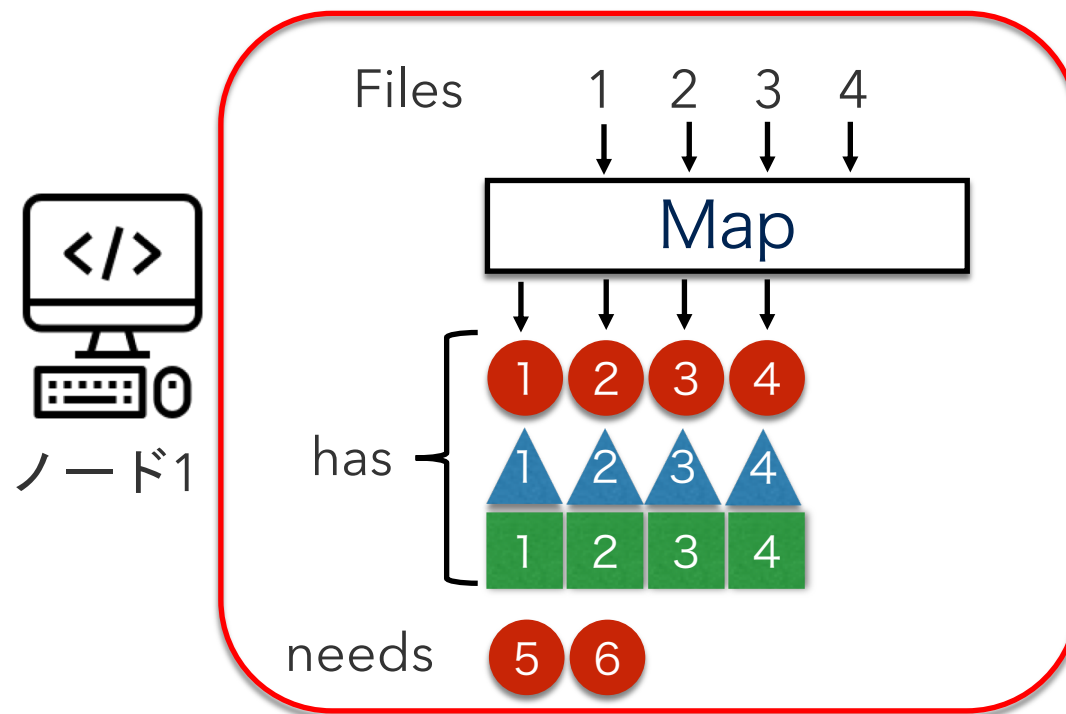
- $\chi_k^q: \mathbb{F}_{2^{\ell_1}} \times \dots \times \mathbb{F}_{2^{\ell_K}} \times (\mathbb{F}_{2^T})^{Q|\mathcal{M}_k|} \rightarrow (\mathbb{F}_{2^T})^N$

$$(v_{q,1}, \dots, v_{q,N}) = \chi_k^q(X_1, \dots, X_K, \{v_{q,n} : q \in \{1, \dots, Q\}, w_n \in \mathcal{M}_k\})$$

Achievability

- 準備(Cont'd)

- $\mathcal{M}_k \subseteq \{w_1, \dots, w_N\}$: ノード k が担当するMap関数の入力ファイル集合
- $\mathcal{W}_k \subseteq \{1, \dots, Q\}$: ノード k が担当するReduce関数のインデックス集合



計算負荷：

$$r = \frac{\sum_{k=1}^K |\mathcal{M}_k|}{N}$$

通信負荷：

$$L = \frac{\sum_{k=1}^K \ell_k}{QNT}$$

- $\psi_k: (\mathbb{F}_{2^T})^{Q|\mathcal{M}_k|} \rightarrow \mathbb{F}_{2^{\ell_k}}$: ノード k の符号化関数

$$X_k = \psi_k(\{v_{q,n} : q \in \{1, \dots, Q\}, w_n \in \mathcal{M}_k\})$$

- $\chi_k^q: \mathbb{F}_{2^{\ell_1}} \times \dots \times \mathbb{F}_{2^{\ell_K}} \times (\mathbb{F}_{2^T})^{Q|\mathcal{M}_k|} \rightarrow (\mathbb{F}_{2^T})^N$

$$(v_{q,1}, \dots, v_{q,N}) = \chi_k^q(X_1, \dots, X_K, \{v_{q,n} : q \in \{1, \dots, Q\}, w_n \in \mathcal{M}_k\})$$

Achievability

- Map Phase Design

- 簡単のため $\frac{N}{\binom{K}{r}} = \eta_1 \in \mathbb{N}$ とする.
- N 個のファイル $\{w_1, \dots, w_N\}$ を η_1 ごとに分割する.

$$\{w_1, \dots, w_N\} = \bigcup_{\mathcal{T} \subset \{1, \dots, K\}, |\mathcal{T}|=r} \mathcal{B}_{\mathcal{T}}$$

- 例 : $N = 6, K = 3, r = 2$ の場合

$$\mathcal{B}_{\{1,2\}} = \{w_3, w_4\}, \mathcal{B}_{\{1,3\}} = \{w_1, w_2\}, \mathcal{B}_{\{2,3\}} = \{w_5, w_6\}$$

- ノード k は, $k \in \mathcal{T}$ ならば $\mathcal{B}_{\mathcal{T}}$ のファイルを Map.
 - 例 : ノード 1 は $1 \in \{1, 2\}, 1 \in \{1, 3\}$ なので, $\mathcal{B}_{\{1,2\}}$ と $\mathcal{B}_{\{1,3\}}$ のファイルを Map.

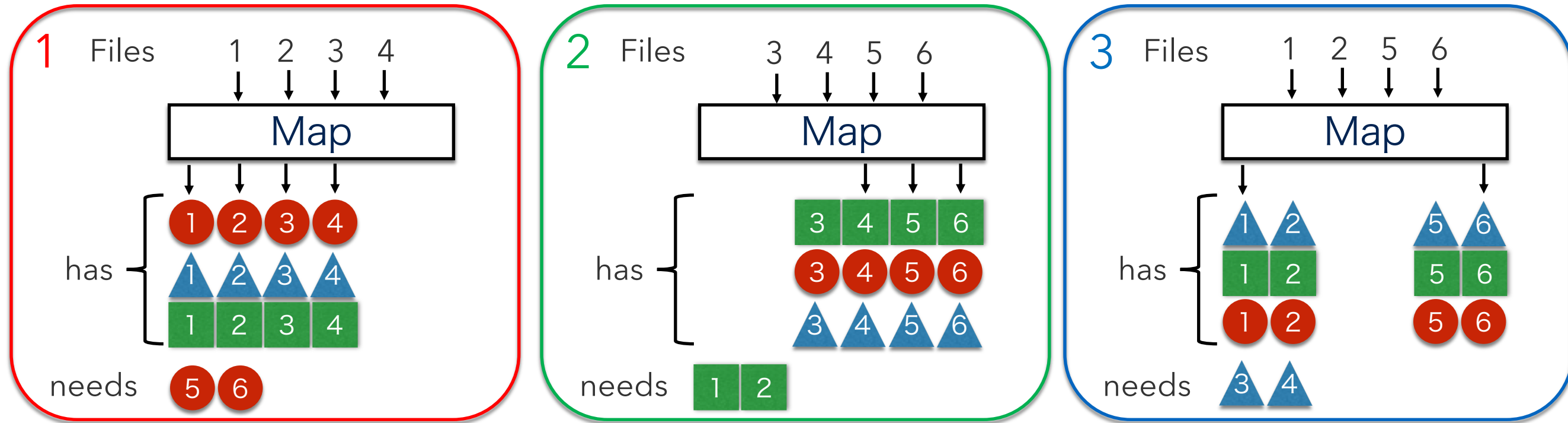
- Reduce Phase Design

- $\frac{Q}{K} = \eta_2$ とし, $|\mathcal{W}_1| = \dots = |\mathcal{W}_K| = \frac{Q}{K}$ とする.

Achievability

- 符号化

- $|S| = r + 1$ となるようなノードの部分集合 S を選ぶ.



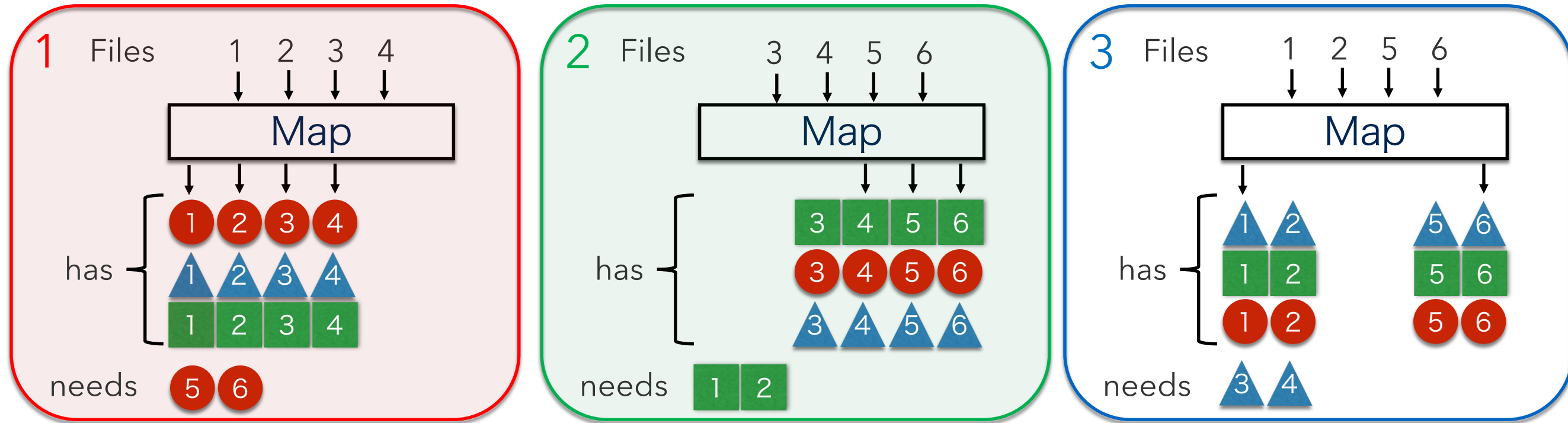
- $|S_1| = r$ であるような $S_1 \subseteq S$ に対して, 以下を定義.

$$\mathcal{V}_{S_1}^{S \setminus S_1} = \left\{ v_{q,n} : q \in \bigcap_{k \in S \setminus S_1} \mathcal{W}_k, q \notin \bigcup_{k \notin S} \mathcal{W}_k, w_n \in \bigcap_{k \in S_1} \mathcal{M}_k, w_n \notin \bigcup_{k \notin S_1} \mathcal{M}_k \right\}$$

Achievability

- 符号化

- $|S| = r + 1$ となるようなノードの部分集合 S を選ぶ.



- $|S_1| = r$ であるような $S_1 \subseteq S$ に対して, 以下を定義.

$$\mathcal{V}_{S_1}^{S \setminus S_1} = \left\{ v_{q,n} : q \in \bigcap_{k \in S \setminus S_1} \mathcal{W}_k, q \notin \bigcup_{k \notin S} \mathcal{W}_k, w_n \in \bigcap_{k \in S_1} \mathcal{M}_k, w_n \notin \bigcup_{k \notin S_1} \mathcal{M}_k \right\}$$

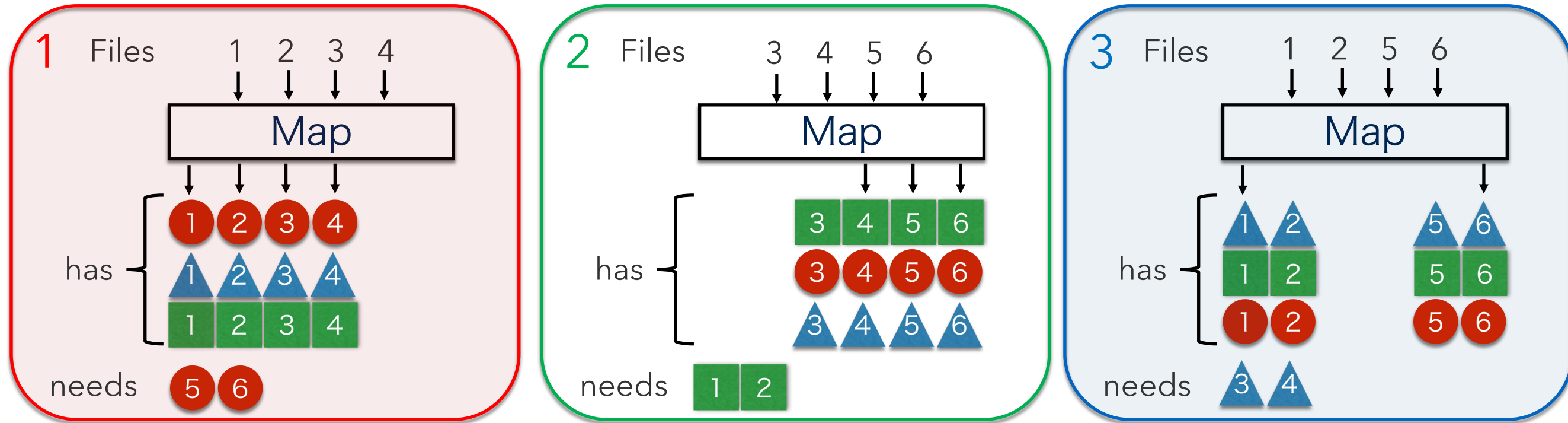
$S_1 = \{1, 2\}$ のとき :

$$\mathcal{V}_{\{1,2\}}^{\{3\}} = \{v_{3,3}, v_{3,4}\}$$

Achievability

- 符号化

- $|S| = r + 1$ となるようなノードの部分集合 S を選ぶ.



- $|S_1| = r$ であるような $S_1 \subseteq S$ に対して, 以下を定義.

$$\mathcal{V}_{S_1}^{S \setminus S_1} = \left\{ v_{q,n} : q \in \bigcap_{k \in S \setminus S_1} \mathcal{W}_k, q \notin \bigcup_{k \notin S} \mathcal{W}_k, w_n \in \bigcap_{k \in S_1} \mathcal{M}_k, w_n \notin \bigcup_{k \notin S_1} \mathcal{M}_k \right\}$$

$S_1 = \{1, 2\}$ のとき :

$$\mathcal{V}_{\{1,2\}}^{\{3\}} = \{v_{3,3}, v_{3,4}\}$$

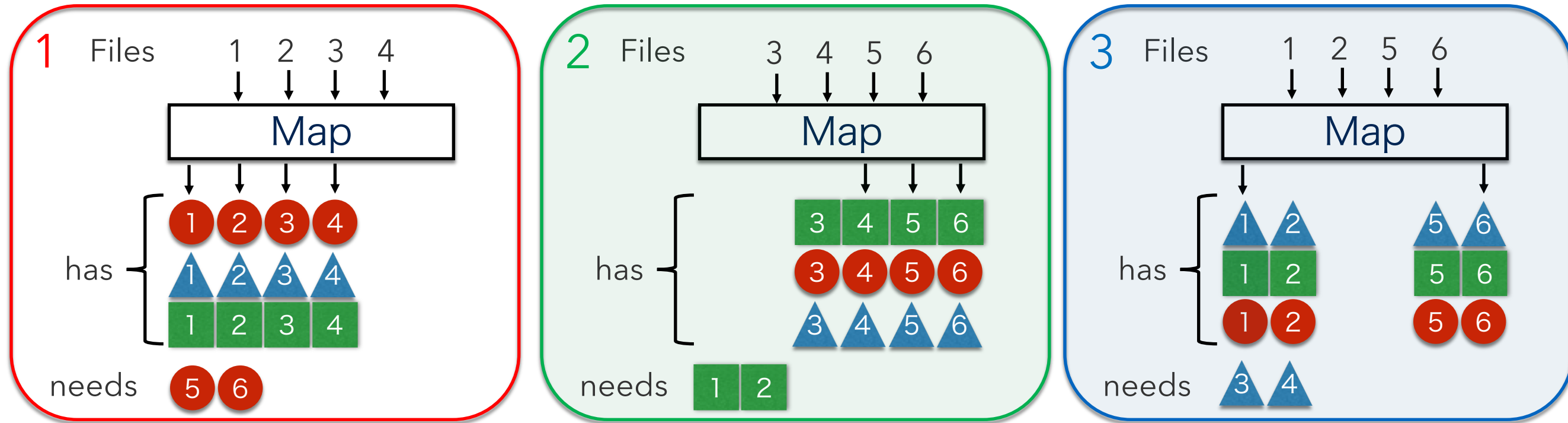
$S_1 = \{1, 3\}$ のとき :

$$\mathcal{V}_{\{1,3\}}^{\{2\}} = \{v_{2,1}, v_{2,2}\}$$

Achievability

- 符号化

- $|S| = r + 1$ となるようなノードの部分集合 S を選ぶ.



- $|S_1| = r$ であるような $S_1 \subseteq S$ に対して, 以下を定義.

$$\mathcal{V}_{S_1}^{S \setminus S_1} = \left\{ v_{q,n} : q \in \bigcap_{k \in S \setminus S_1} \mathcal{W}_k, q \notin \bigcup_{k \notin S} \mathcal{W}_k, w_n \in \bigcap_{k \in S_1} \mathcal{M}_k, w_n \notin \bigcup_{k \notin S_1} \mathcal{M}_k \right\}$$

$S_1 = \{1, 2\}$ のとき :

$$\mathcal{V}_{\{1,2\}}^{\{3\}} = \{v_{3,3}, v_{3,4}\}$$

$S_1 = \{1, 3\}$ のとき :

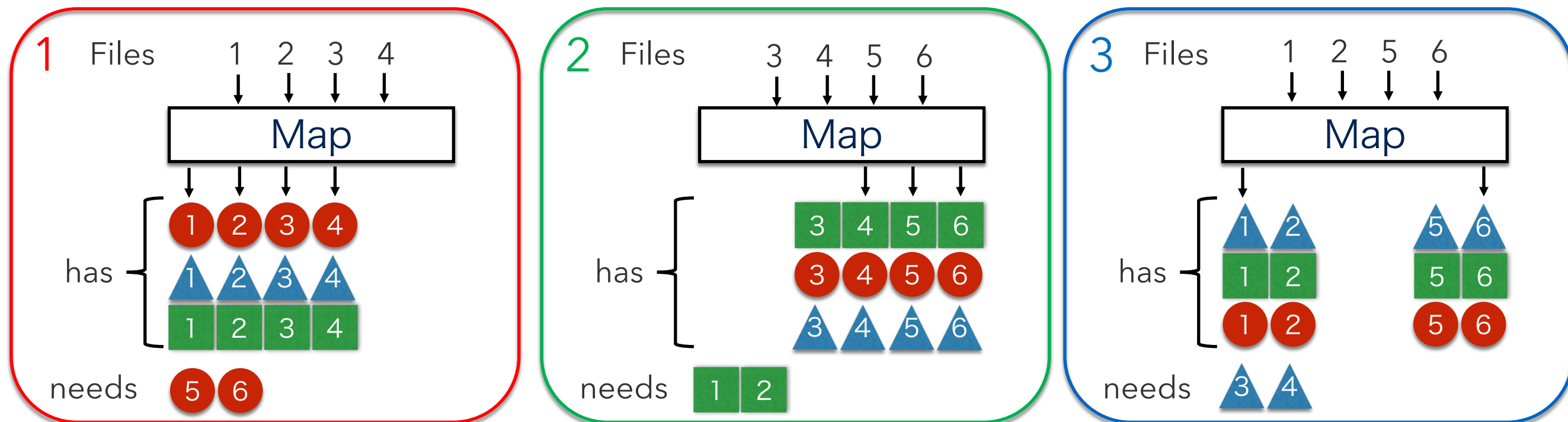
$$\mathcal{V}_{\{1,3\}}^{\{2\}} = \{v_{2,1}, v_{2,2}\}$$

$S_1 = \{2, 3\}$ のとき :

$$\mathcal{V}_{\{2,3\}}^{\{1\}} = \{v_{1,5}, v_{1,6}\}$$

Achievability

• 符号化 (Cont'd)



- $\mathcal{V}_{S_1}^{S \setminus S_1}$ の要素を連結したものを $U_{S_1}^{S \setminus S_1}$ とし, それを r 個のパケットに分割する.

$$\mathcal{V}_{\{1,2\}}^{\{3\}} = \{v_{3,3}, v_{3,4}\} \Rightarrow U_{\{1,2\},1}^{\{3\}} = \{v_{3,3}\}, U_{\{1,2\},2}^{\{3\}} = \{v_{3,4}\}$$

$$\mathcal{V}_{\{1,3\}}^{\{2\}} = \{v_{2,1}, v_{2,2}\} \Rightarrow U_{\{1,3\},1}^{\{2\}} = \{v_{2,1}\}, U_{\{1,3\},3}^{\{2\}} = \{v_{2,2}\}$$

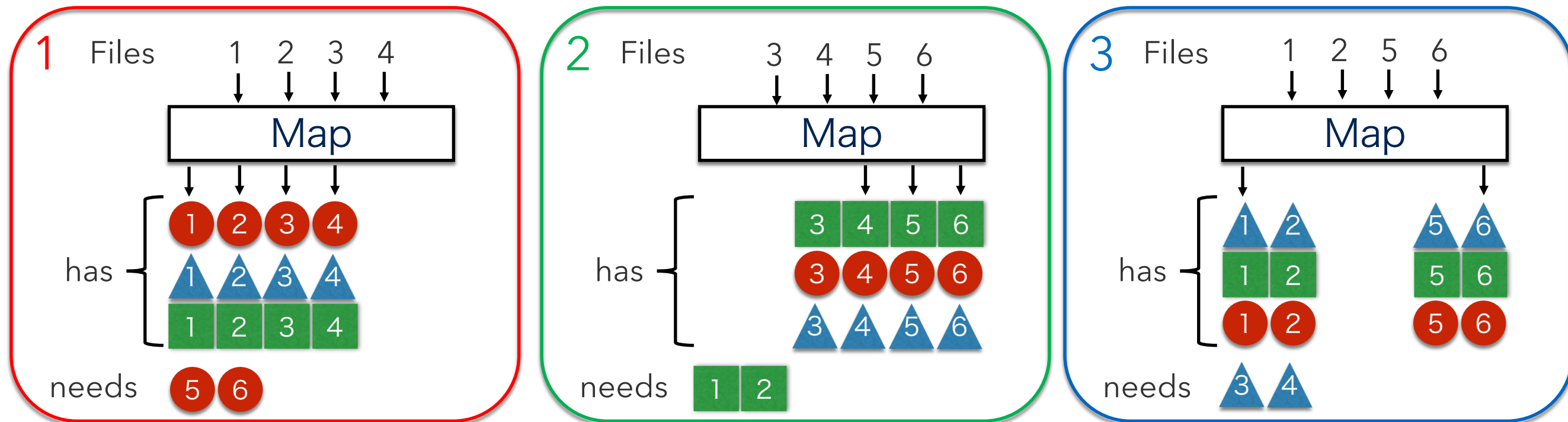
$$\mathcal{V}_{\{2,3\}}^{\{1\}} = \{v_{1,5}, v_{1,6}\} \Rightarrow U_{\{2,3\},2}^{\{1\}} = \{v_{1,5}\}, U_{\{2,3\},3}^{\{1\}} = \{v_{1,6}\}$$

構成法からノード k が $U_{S_1,k}^{S \setminus S_1}$ を持っている点に注意.

※ この例では, $\eta_1 \eta_2 = r = 2$ なので, パケットを中間ファイルと同一視できる.

Achievability

• 符号化 (Cont'd)

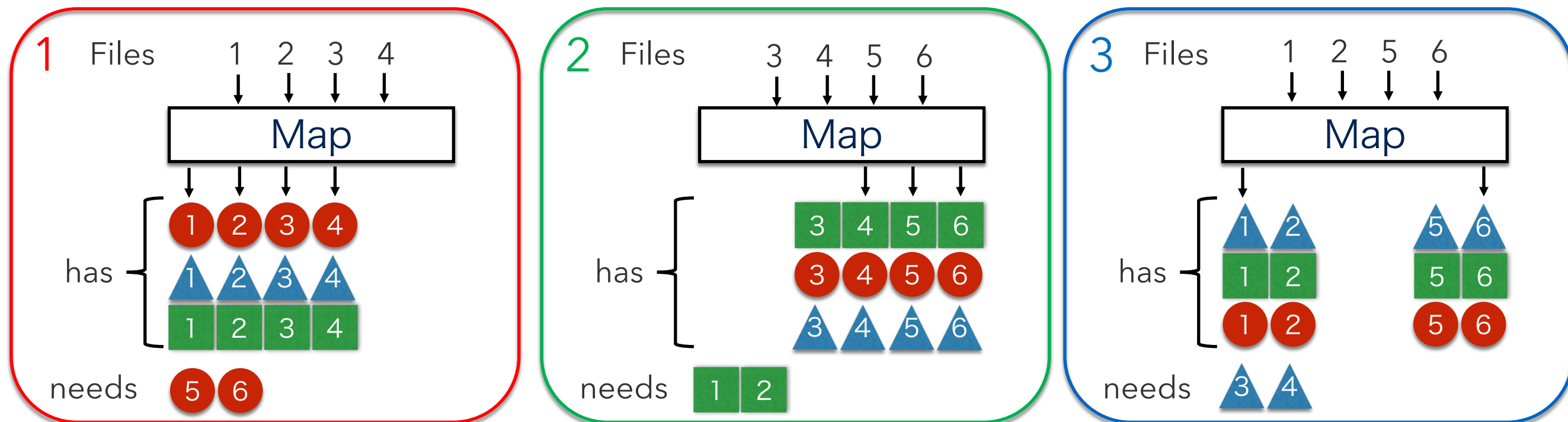


- 各 $k \in S$ に対して, k を含むサイズ r の S の部分集合が $n_1 = \binom{|S|-1}{r-1}$ 個存在.
これを $S_{(k)}[1], \dots, S_{(k)}[n_1]$ とする.
 - 例: $k = 1$ に対して, $\{1, 2\}$ と $\{1, 3\}$
- ノード k は $\bigoplus_{j=1}^{n_2} U_{S_{(k)}[j], k}^{S \setminus S_{(k)}[j]}$ を S の他のノードに対してマルチキャスト.
 - 例: $k = 1$ は, $U_{\{1, 2\}, 1}^{\{3\}} \oplus U_{\{1, 3\}, 1}^{\{2\}} = v_{3, 3} \oplus v_{2, 1}$ をマルチキャスト

以上をサイズが $r + 1$ であるような全ての S に対して行う.

Achievability

復号



- $j \in S, j \neq k$ に対して, j, k を両方含むサイズ r の S の部分集合は, $\binom{|S|-2}{r-2}$ 個存在. $\Rightarrow$ ノード j は, ノード k の送信語の元となる n_1 個のパケット

$U_{S(k)[1],k}^{S \setminus S(k)[1]}, \dots, U_{S(k)[n_1],k}^{S \setminus S(k)[n_1]}$ のうち, $\binom{|S|-2}{r-2}$ 個は既知.

- $\binom{|S|-1}{r-1} - \binom{|S|-2}{r-2} = \binom{|S|-2}{r-1} = \binom{r-1}{r-1} = 1$ 個のパケットが復元可能.
- S 内の自身以外からの r 個のメッセージから r 個のパケットを復元.
 $\Rightarrow \eta_1 \eta_2$ 個の中間ファイルを復元.

Achievability

- Correctness

- ノード k はMapフェーズで1つの関数につき $\frac{rN}{K}$ 個の中間ファイルを計算.
他のノードから $N - \frac{rN}{K} = \binom{K-1}{r} \eta_1$ 個の中間ファイルを得る必要がある.
- K 個のノード集合からノード k を含むサイズ $r+1$ の集合 S の選び方は $\binom{K-1}{r}$ 通り.

$\Rightarrow$ ノード k が他のノードから得られる中間ファイルの数は $\binom{K-1}{r} \eta_1$

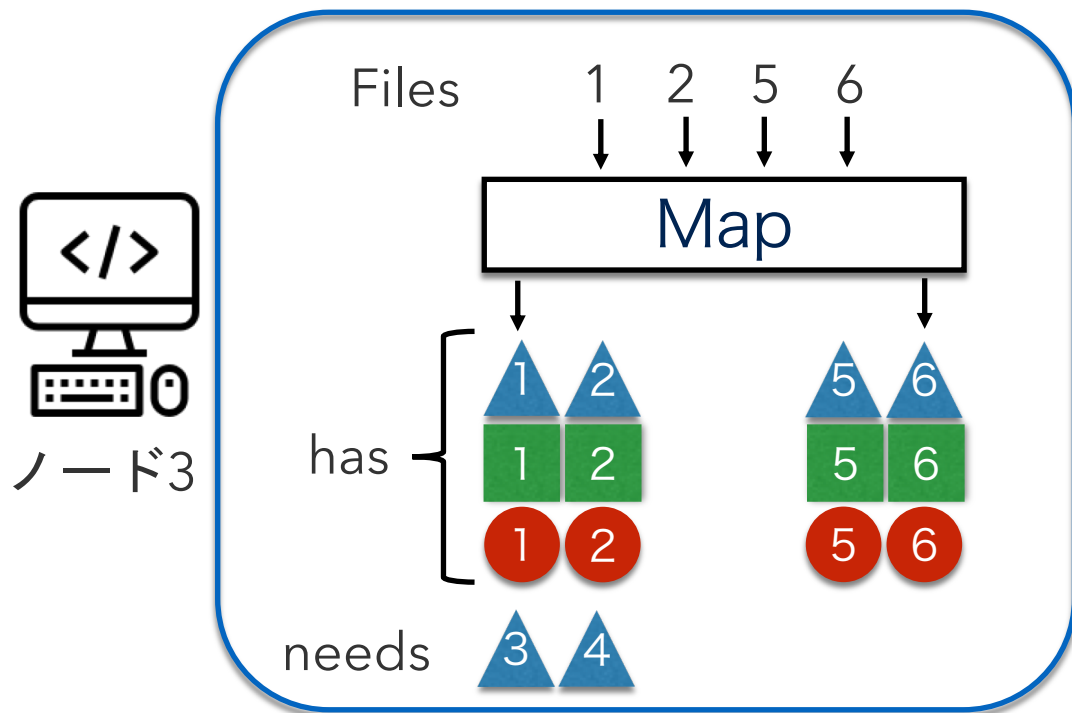
- 通信負荷

- 各 S に対し, あるノードが送信するメッセージは $\frac{\eta_1 \eta_2 T}{r}$ ビット

$$\begin{aligned} L_{\text{coded}}(r, s) &= \frac{\binom{K}{r+1} (r+1) \frac{\eta_1 \eta_2 T}{r}}{QNT} \\ &= \frac{\binom{K}{r+1} (r+1) \frac{\eta_1 \eta_2 T}{r}}{\eta_2 K \eta_1 \binom{K}{r} T} = \frac{1}{r} \left(1 - \frac{r}{K}\right) \end{aligned}$$

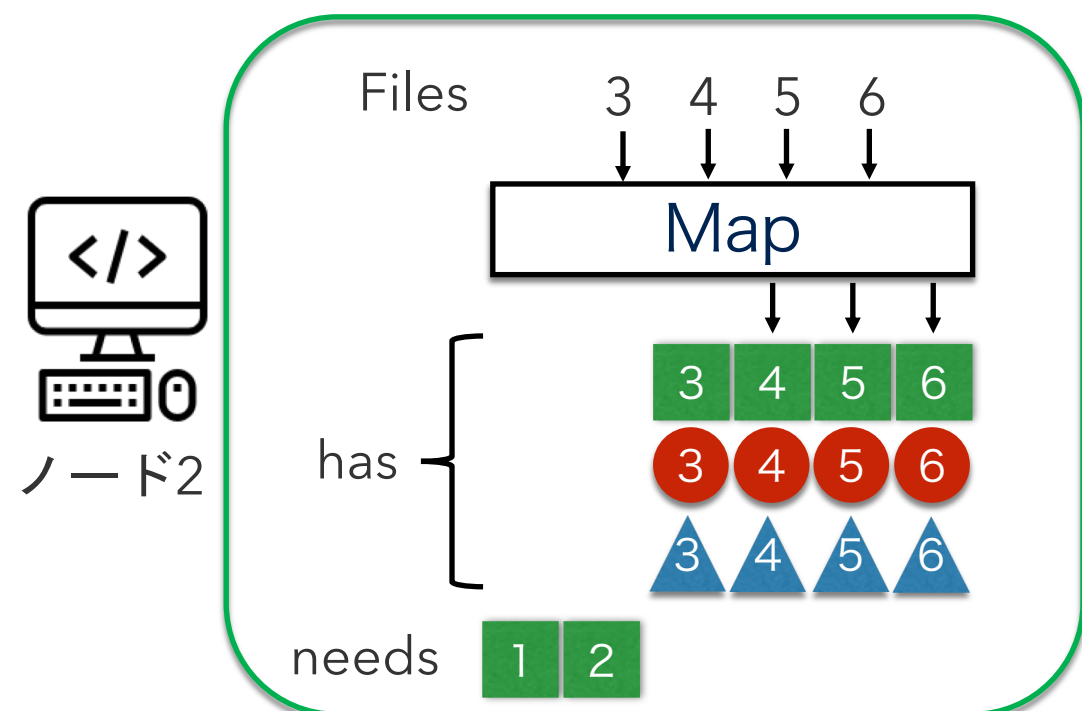
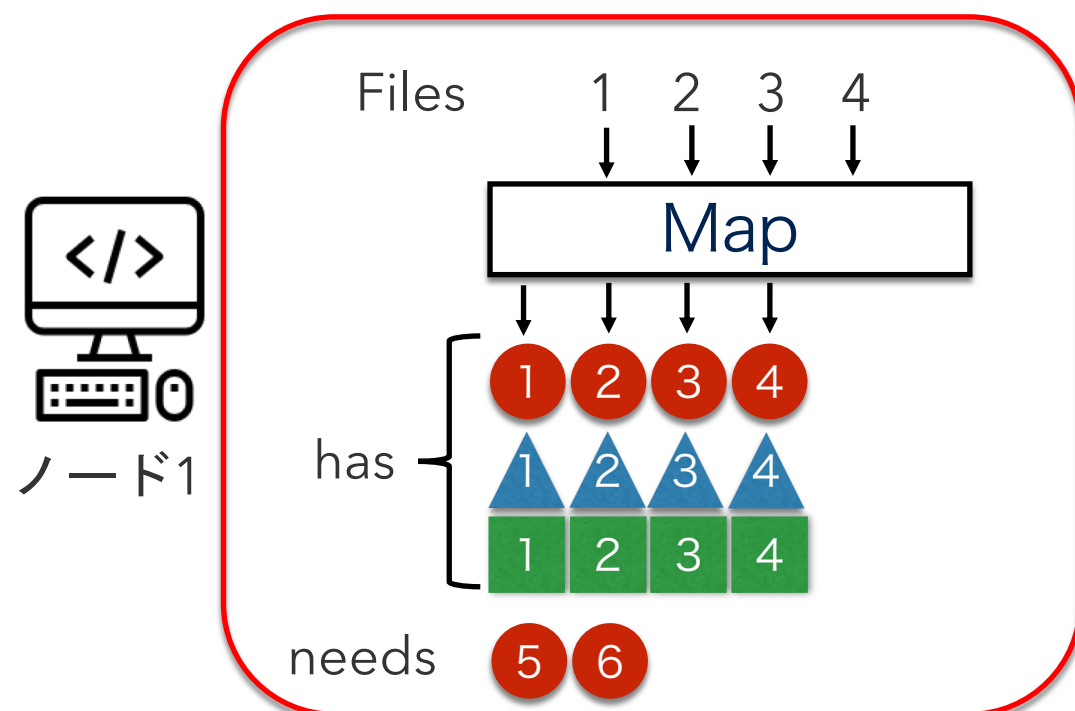
Index coding との関係

- 通信部分はIndex coding の問題と見る事が出来る



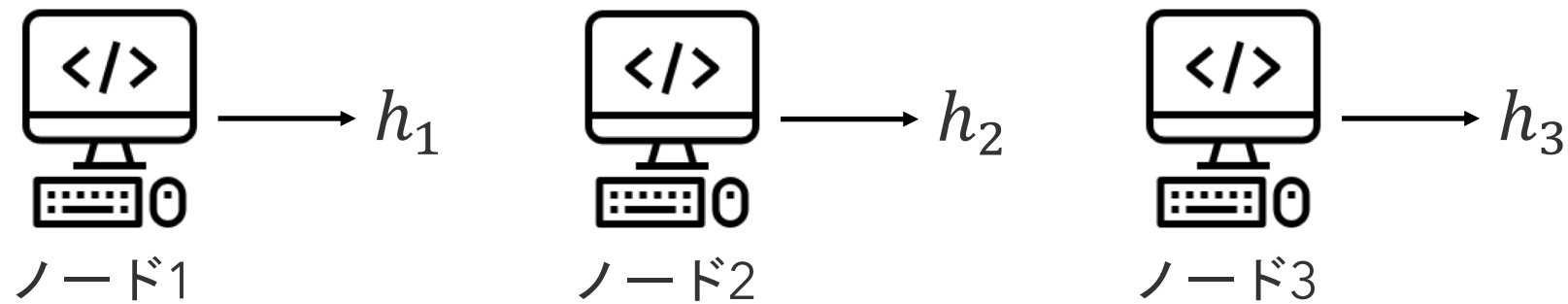
- それぞれのノードが要求する情報が異なる場合は、一般的には未解決.
- この問題では、Mapフェーズの設計により、Side Information が設計できるため、タイトな下界が導出可能.

※ 今回は下界の導出は省略.

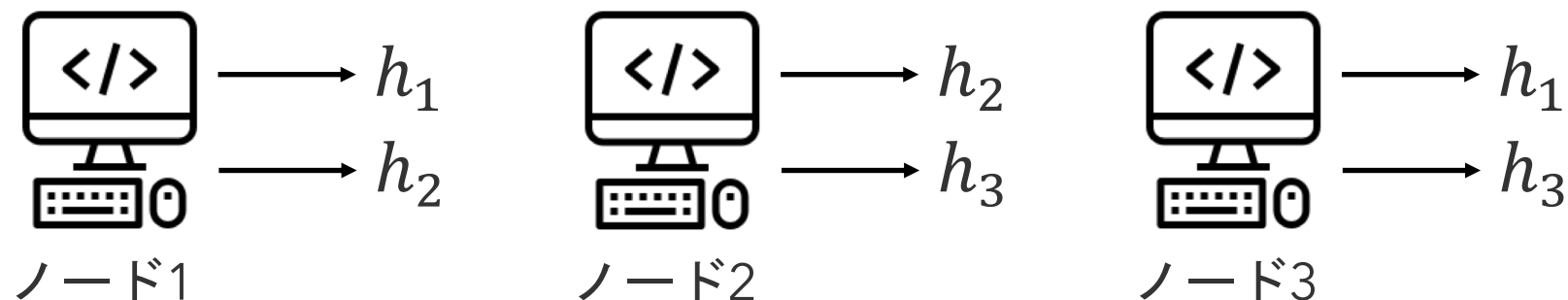


Cascaded distributed computing

- ここまで：各 $h_q, q = 1, \dots, Q$ に対して， h_q を計算するノードは1つだけ.



- 各 $h_q, q = 1, \dots, Q$ を複数個(s 個)のノードが計算する問題を考える.



- Practical には， ある計算の出力が次の計算の入力となる場合がある.
- 出力に冗長性を持たせることで， fault-toleranceを向上.

Cascaded distributed computing

- (r, s, L) のfeasibilityを先ほどと同様に定義し, $L^*(r, s)$ を以下のように定義.

$$L^*(r, s) = \inf\{L : (r, s, L) \text{ is feasible}\}$$

定理 [Li et al. 2018]

T を十分大きく取るとき, $L^*(r, s)$ は以下で与えられる.

$$L^*(r, s) = \sum_{l=\max\{r+1, s\}}^{\min\{r+s, K\}} \frac{l \binom{K}{l} \binom{l-2}{r-1} \binom{r}{l-s}}{r \binom{K}{r} \binom{K}{s}}, \quad r \in \{1, \dots, K\}$$

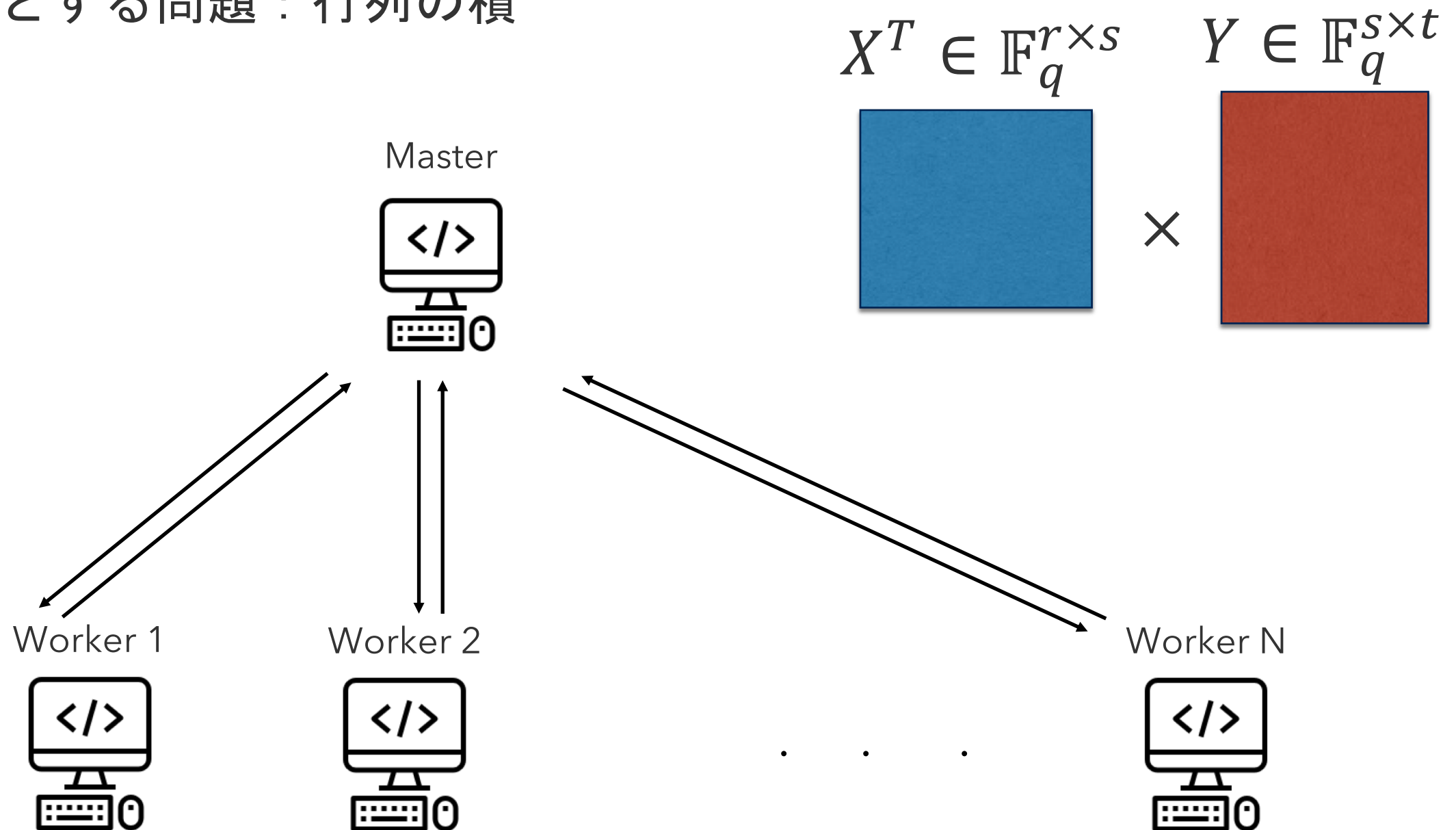
整数以外の r については, $\{(r, L^*(r, s)) : r \in \{1, \dots, K\}\}$ のlower convex envelopeで与えられる.

- 符号化・復号化の変更点
 - サイズが $r+1$ 以上の全ての部分集合 S の間で通信を行う.
 - 符号化では線形変換をし, 復号化では連立方程式を解く.

Master / Worker モデルにおける誤り訂正符号
を利用したStraggler 効果軽減

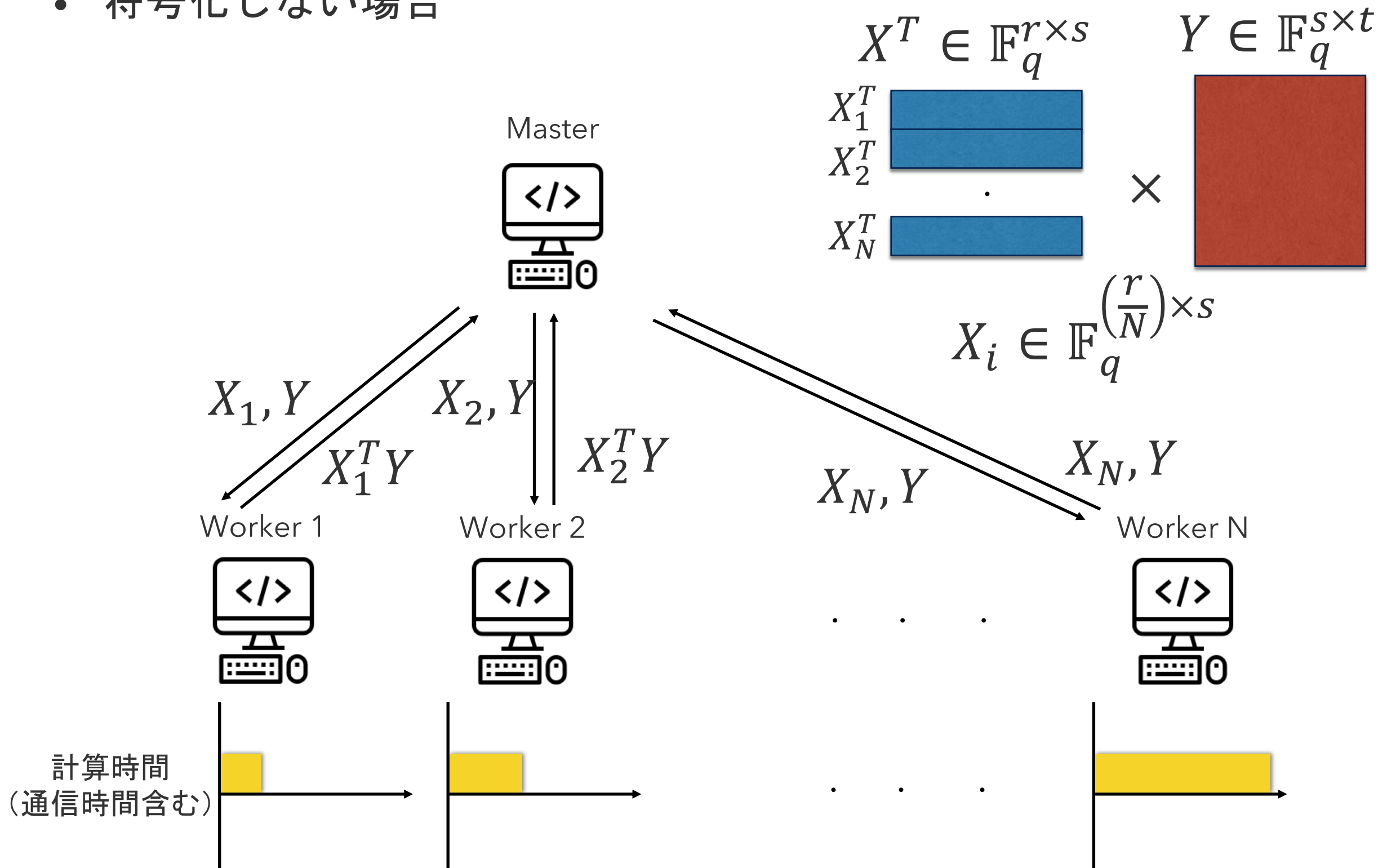
Master / Worker モデル

- 対象とする問題：行列の積



Master / Worker モデル

- 符号化しない場合



- 全体の処理時間 = 一番遅いWorkerの処理時間

Master / Worker モデル

- 反復符号を利用した場合 ($N = 4$ の場合)

$$\begin{array}{|c|} \hline X_1^T \in \mathbb{F}_q^{\left(\frac{r}{2}\right) \times s} \\ \hline X_2^T \in \mathbb{F}_q^{\left(\frac{r}{2}\right) \times s} \\ \hline \end{array} \quad Y \in \mathbb{F}_q^{s \times t}$$

Worker 1



$$X_1^T Y$$

Worker 2



$$X_1^T Y$$

Worker 3



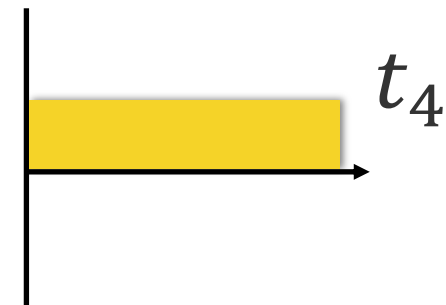
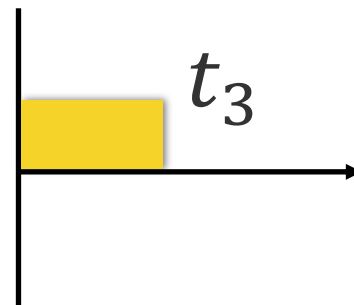
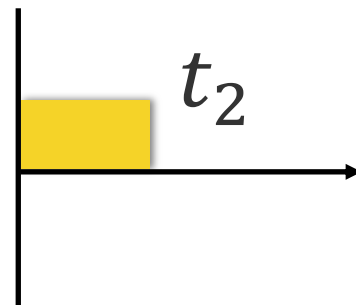
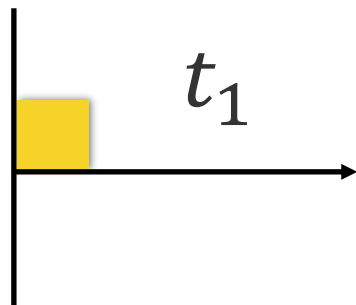
$$X_2^T Y$$

Worker 4



$$X_2^T Y$$

計算時間
(通信時間含む)



- 全体の処理時間 = $\max(\min(t_1, t_2), \min(t_3, t_4))$

Master / Worker モデル

- 反復符号を利用した場合（繰り返し回数 N/K ）

$$X_1^T \in \mathbb{F}_q^{\left(\frac{r}{K}\right) \times s}$$

:

$$X_K^T \in \mathbb{F}_q^{\left(\frac{r}{K}\right) \times s}$$

$$Y \in \mathbb{F}_q^{s \times t}$$

- Worker i の計算時間 : t_i
- 全体の処理時間 :

$$\max_{i \in \{1, \dots, K\}} \min_{j \in \{1, \dots, \frac{N}{K}\}} \left\{ t_{(i-1)\frac{N}{K} + j} \right\}$$

Master / Worker モデル

- (3, 2) MDS符号を利用した場合 ($N = 3$ の場合)

$$X_1^T \in \mathbb{F}_q^{\left(\frac{r}{2}\right) \times s}$$
$$X_2^T \in \mathbb{F}_q^{\left(\frac{r}{2}\right) \times s}$$

$$Y \in \mathbb{F}_q^{s \times t}$$

Master



Worker 1



$$X_1^T Y$$

Worker 2



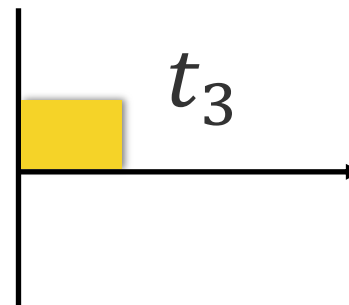
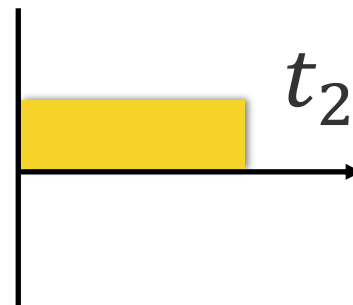
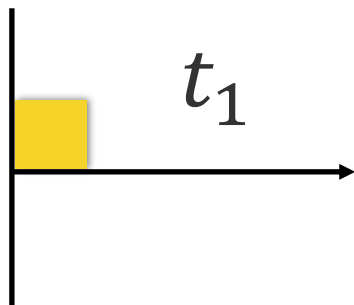
$$X_2^T Y$$

Worker 3



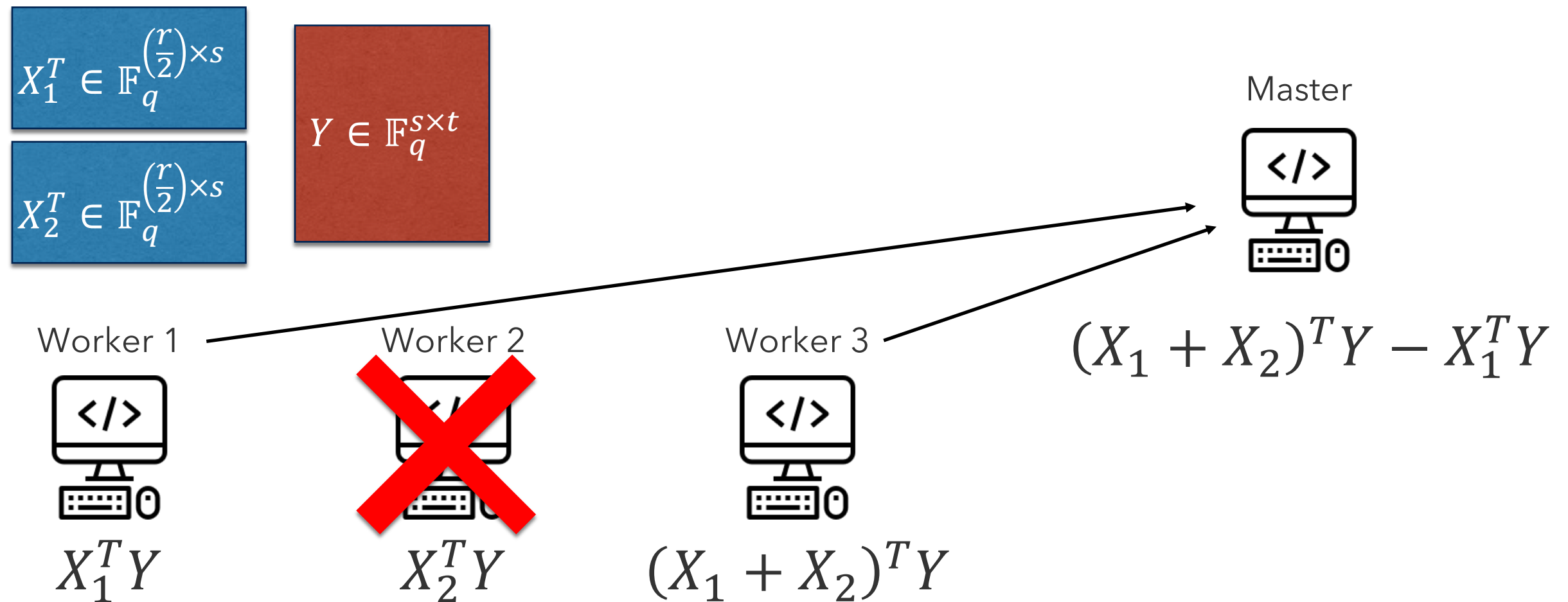
$$(X_1 + X_2)^T Y$$

計算時間
(通信時間含む)

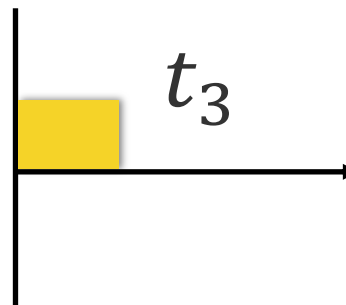
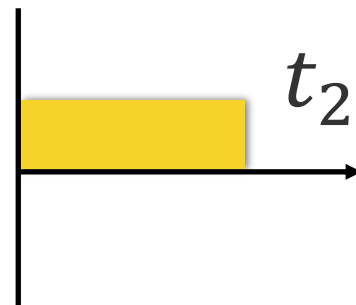
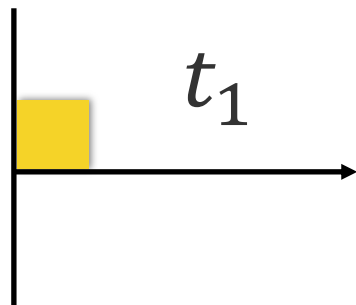


Master / Worker モデル

- (3, 2) MDS符号を利用した場合 ($N = 3$ の場合)

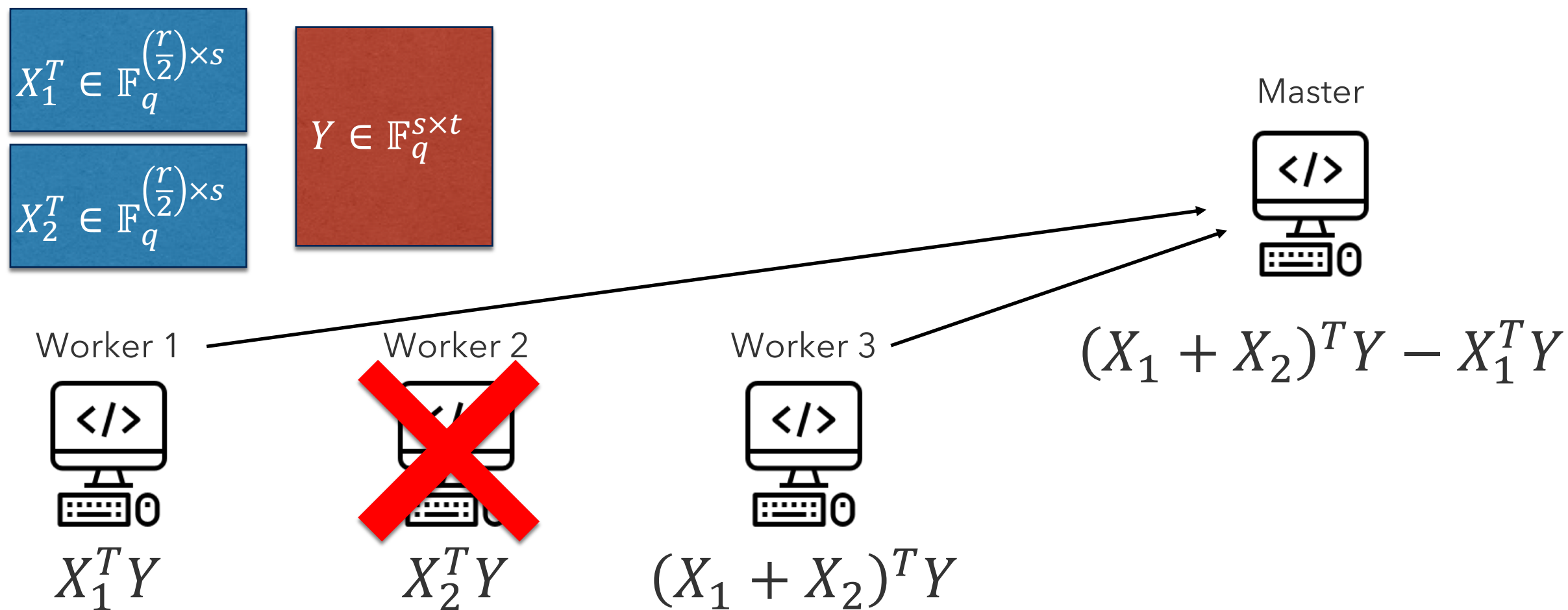


計算時間
(通信時間含む)

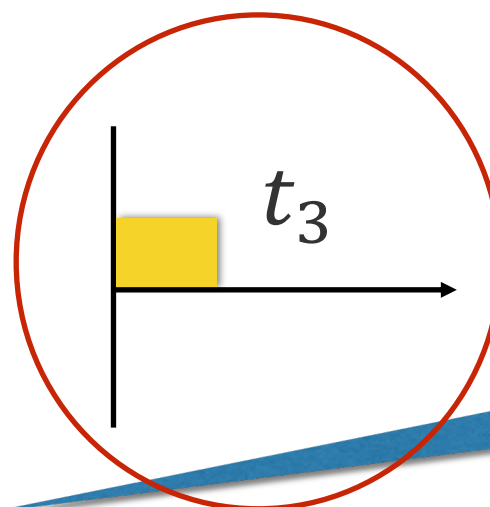
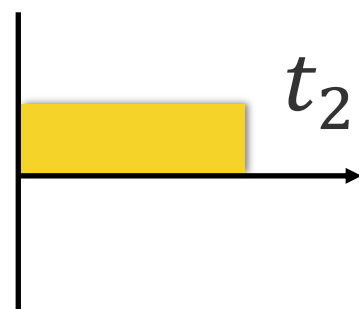
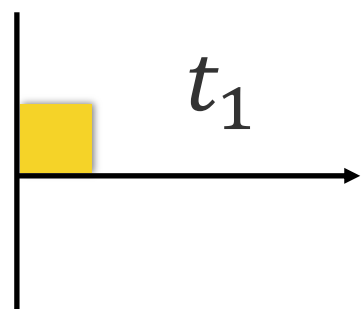


Master / Worker モデル

- (3, 2) MDS符号を利用した場合 ($N = 3$ の場合)



計算時間
(通信時間含む)



$\{t_1, t_2, t_3\}$ を小さい順に並べたときの2番目の値

- Worker 全体の処理時間 = $t_{(2)}$

Master / Worker モデル

- (N, K) MDS符号を利用した場合

$$\begin{array}{c} X_1^T \in \mathbb{F}_q^{\left(\frac{r}{K}\right) \times s} \\ \vdots \\ X_K^T \in \mathbb{F}_q^{\left(\frac{r}{K}\right) \times s} \end{array} \qquad Y \in \mathbb{F}_q^{s \times t}$$

- MDS符号により $\tilde{X}_1, \dots, \tilde{X}_N$ を生成
- Master は最初に計算の終わった K 個の Worker からの出力から $X_1, \dots, X_K$ を復元
- Worker i の計算時間 : t_i
- Worker 全体の処理時間 : $t_{(K)}$

$\{t_1, \dots, t_N\}$ を小さい順に並べたときの K 番目の値

計算時間の解析 [Lee et al., 2018]

- 1つのWorkerでタスクを計算するのにかかる時間：確率変数 T_0
 - $\Pr(T_0 \leq t) = F(t) = 1 - e^{-\mu(t-1)}$, μ : straggling parameter
- タスクを ℓ 個のサブタスクに分割した場合に各サブタスクを計算するのにかかる時間：確率変数 $T_i, i = 1, \dots, \ell$
 - $\Pr(T_i \leq t) = F(\ell t) = 1 - e^{-\mu(\ell t-1)}$
 - 各Workerの計算時間は**独立**と仮定.

計算時間の解析 [Lee et al., 2018]

- 各手法のWorker全体の計算時間の分布

- 符号化なし : $F_{overall}^{uncoded}(t) = [F(Nt)]^N$

- 反復符号 : $F_{overall}^{repetition}(t) = \left[1 - [1 - F(Kt)]^{\frac{N}{K}}\right]^K$

- (N, K) MDS符号 :

$$F_{overall}^{(N,K)MDS}(t) = \int_{\tau=0}^t NKf(K\tau) \binom{N-1}{K-1} F(K\tau)^{K-1} [1 - F(K\tau)]^{N-K} d\tau$$

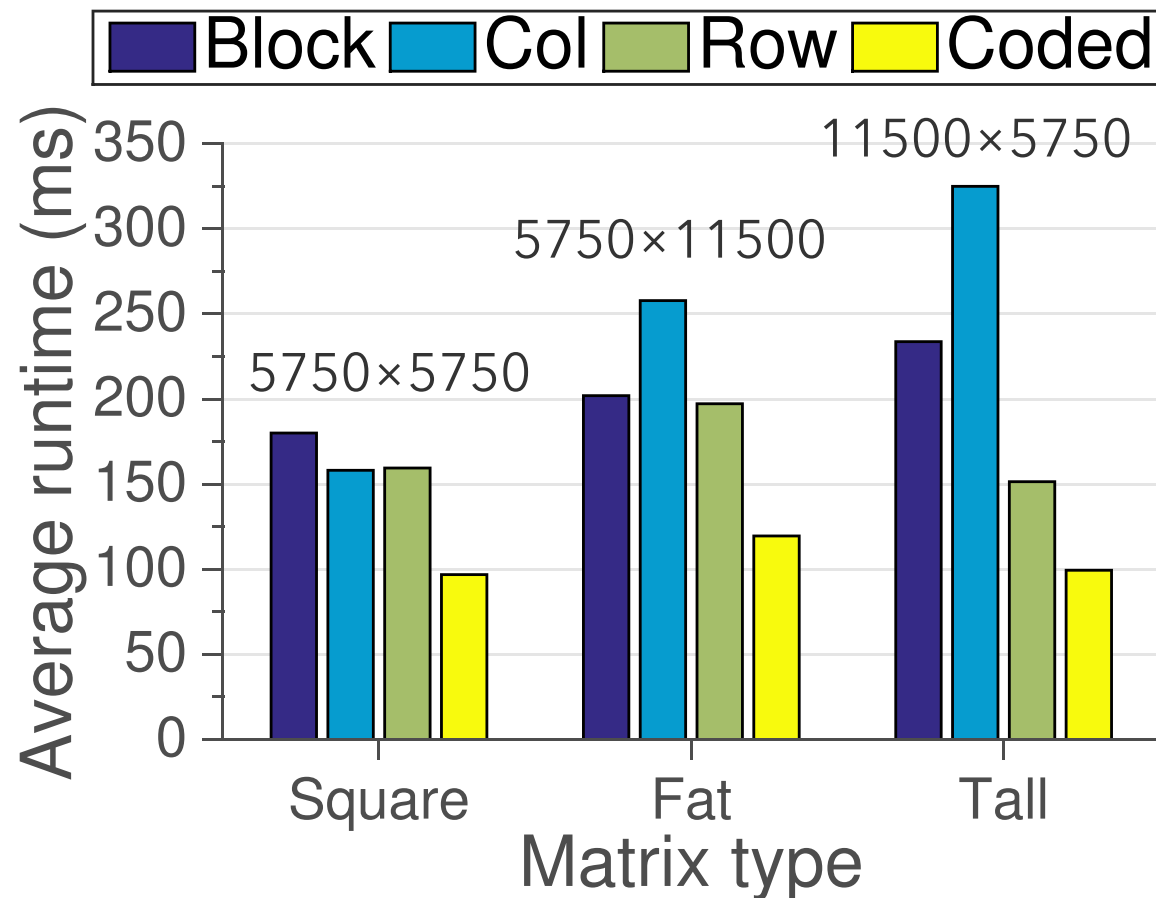
- 計算時間の期待値

- 符号化なし : $E[T_{overall}^{uncoded}] \cong \frac{1}{N} \left(1 + \frac{1}{\mu} \log N\right) = \Theta\left(\frac{\log N}{N}\right)$

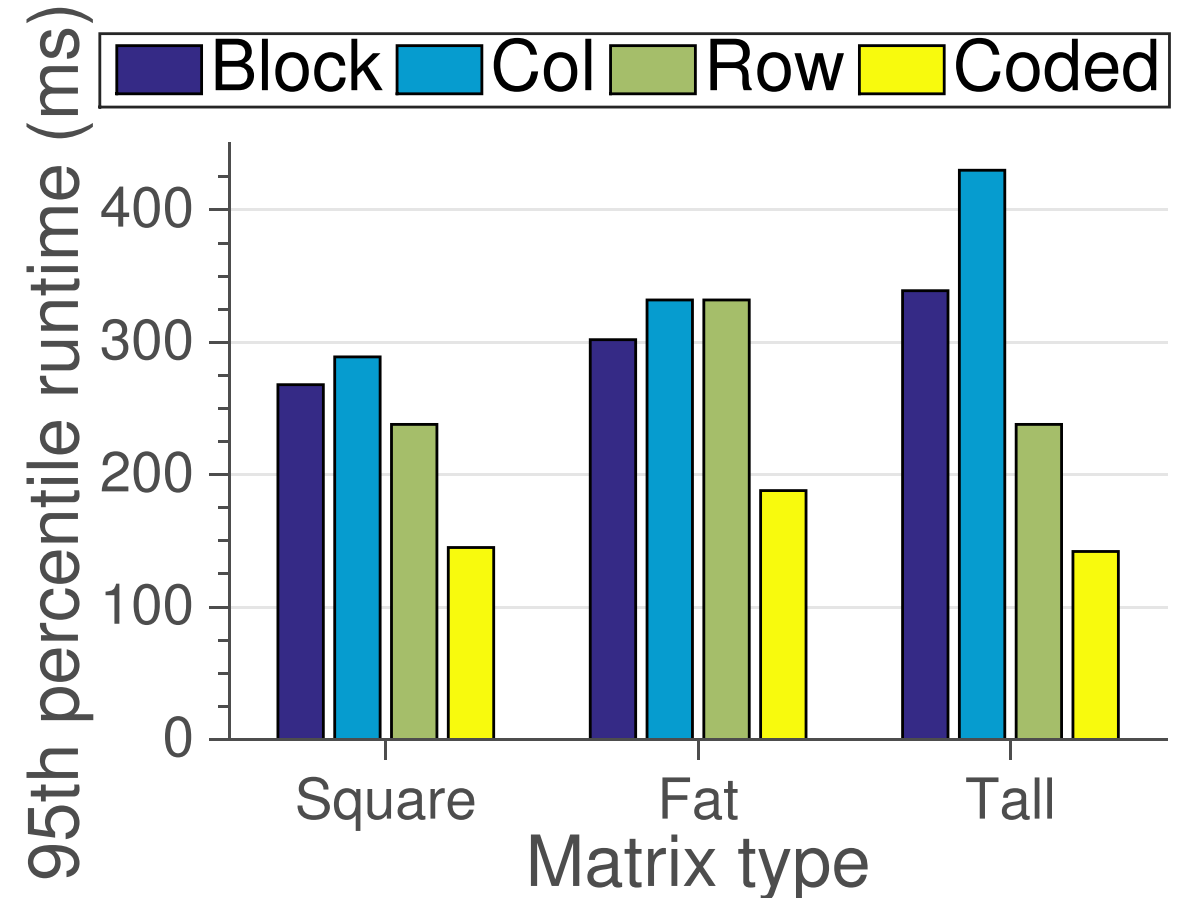
- 反復符号 : $E[T_{overall}^{repetition}] \cong \frac{1}{K} \left(1 + \frac{K}{N\mu} \log K\right) = \Theta\left(\frac{\log N}{N}\right)$

- (N, K) MDS符号 : $E[T_{overall}^{(N,K)MDS}] \cong \frac{1}{K} \left(1 + \frac{1}{\mu} \log \left(\frac{N}{N-K}\right)\right) = \Theta\left(\frac{1}{N}\right)$

Amazon EC2 上での実験



(a) m1-small, average runtime



(b) m1-small, tail runtime

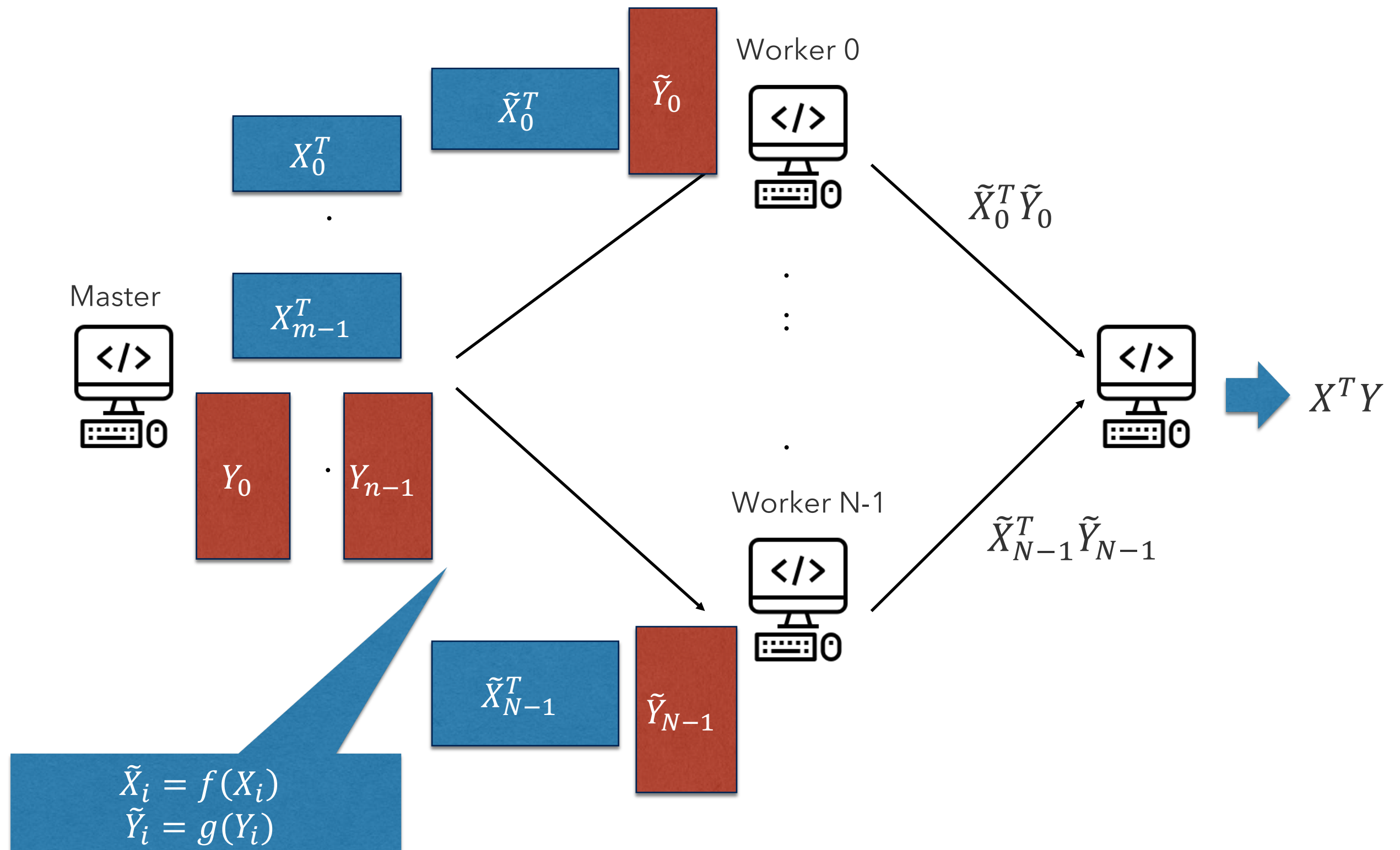
引用：

Lee et al.: Speeding Up Distributed Machine Learning Using Codes

- 行列とベクトルの積を $N = 25$ の Worker と Master で計算
- (25, 23) MDS符号を利用

Polynomial 符号 (Reed-Solomon 符号)

- X と Y の両方を分割



Polynomial 符号 (Reed-Solomon 符号)

- 例 : $N = 5, m = n = 2$ の場合

$$X^T = \begin{bmatrix} X_0^T \\ X_1^T \end{bmatrix} \quad Y = [Y_0 \quad Y_1] \quad \rightarrow \quad X^T Y = \begin{bmatrix} X_0^T Y_0 & X_0^T Y_1 \\ X_1^T Y_0 & X_1^T Y_1 \end{bmatrix}$$

Worker i

$$\begin{aligned} \tilde{X}_i &= X_0 + i \cdot X_1 \\ \tilde{Y}_i &= Y_0 + i^2 \cdot Y_1 \end{aligned} \quad \rightarrow \quad \begin{array}{c} \text{Worker } i \\ \text{Computer Icon} \end{array} \quad \rightarrow \quad \tilde{Z}_i = X_0^T Y_0 + i \cdot X_1^T Y_0 + i^2 \cdot X_0^T Y_1 + i^3 \cdot X_1^T Y_1$$

- Worker 1 ~ 4 の出力 :

$$\begin{bmatrix} \tilde{Z}_1 \\ \tilde{Z}_2 \\ \tilde{Z}_3 \\ \tilde{Z}_4 \end{bmatrix} = \begin{bmatrix} 1^0 & 1^1 & 1^2 & 1^3 \\ 2^0 & 2^1 & 2^2 & 2^3 \\ 3^0 & 3^1 & 3^2 & 3^3 \\ 4^0 & 4^1 & 4^2 & 4^3 \end{bmatrix} \begin{bmatrix} X_0^T Y_0 \\ X_1^T Y_0 \\ X_0^T Y_1 \\ X_1^T Y_1 \end{bmatrix}$$

Polynomial 符号 (Reed-Solomon 符号)

- 例 : $N = 5, m = n = 2$ の場合

$$X^T = \begin{bmatrix} X_0^T \\ X_1^T \end{bmatrix} \quad Y = [Y_0 \quad Y_1] \quad \rightarrow \quad X^T Y = \begin{bmatrix} X_0^T Y_0 & X_0^T Y_1 \\ X_1^T Y_0 & X_1^T Y_1 \end{bmatrix}$$

Worker i

$$\begin{aligned} \tilde{X}_i &= X_0 + i \cdot X_1 \\ \tilde{Y}_i &= Y_0 + i^2 \cdot Y_1 \end{aligned} \rightarrow \begin{array}{c} \text{Worker } i \\ \text{Computer icon} \end{array} \rightarrow \tilde{Z}_i = X_0^T Y_0 + i \cdot X_1^T Y_0 + i^2 \cdot X_0^T Y_1 + i^3 \cdot X_1^T Y_1$$

- Worker 1 ~ 4 の出力 :

Vandermonde 行列

$$\begin{bmatrix} \tilde{Z}_1 \\ \tilde{Z}_2 \\ \tilde{Z}_3 \\ \tilde{Z}_4 \end{bmatrix} = \begin{bmatrix} 1^0 & 1^1 & 1^2 & 1^3 \\ 2^0 & 2^1 & 2^2 & 2^3 \\ 3^0 & 3^1 & 3^2 & 3^3 \\ 4^0 & 4^1 & 4^2 & 4^3 \end{bmatrix} \begin{bmatrix} X_0^T Y_0 \\ X_1^T Y_0 \\ X_0^T Y_1 \\ X_1^T Y_1 \end{bmatrix}$$

多項式補間で復号可能

Polynomial 符号 (Reed-Solomon 符号)

- Master の事前計算
 - $X = [X_1 \ \cdots \ X_m], Y = [Y_1 \ \cdots \ Y_n]$ と分割.
 - 各Workerに相異なる $x_i \in \mathbb{F}_q$ を割り当てる
 - $\tilde{X}_i = \sum_{j=0}^{m-1} X_j x_i^j, \ \tilde{Y}_i = \sum_{j=0}^{n-1} Y_j x_i^{jm}$ を計算し, $\tilde{X}_i, \tilde{Y}_i$ をWorker i に送る
- Worker i の計算
 - $\tilde{X}_i^T \tilde{Y}_i = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} X_j^T Y_k x_i^{j+km}$
 - 次数が $mn - 1$ の多項式 $h(x) = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} X_j^T Y_k x^{j+km}$ を $x = x_i$ で評価した値
- Master の復号計算
 - mn 個のWorkerの出力を元に多項式補間 (Reed-Solomon符号の復号) アルゴリズムを用いて $h(x)$ を復元
 - $X_j^T Y_k, j = 0, \dots, m-1, k = 0, \dots, n-1$ を得る

- MapReduceへの応用
 - S. Li et al., "Fundamental Tradeoff Between Computation and Communication in Distributed Computing," IEEE Trans. on IT, Vol. 64, No. 1, Jan. 2018, pp. 109–28.
- Straggler Mitigation
 - K. Lee et al., "Speeding Up Distributed Machine Learning Using Codes," IEEE Trans. on IT, Vol. 64, No. 3, Mar. 2018, pp. 1514-1529.
 - Q. Yu et al., "Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication," Proc. of Advances in Neural Information Processing Systems, (NIPS2017), pp. 4403-4413.