Some Results Related to Synchronization of Data in Distributed Storage Systems

Vitaly Skachek

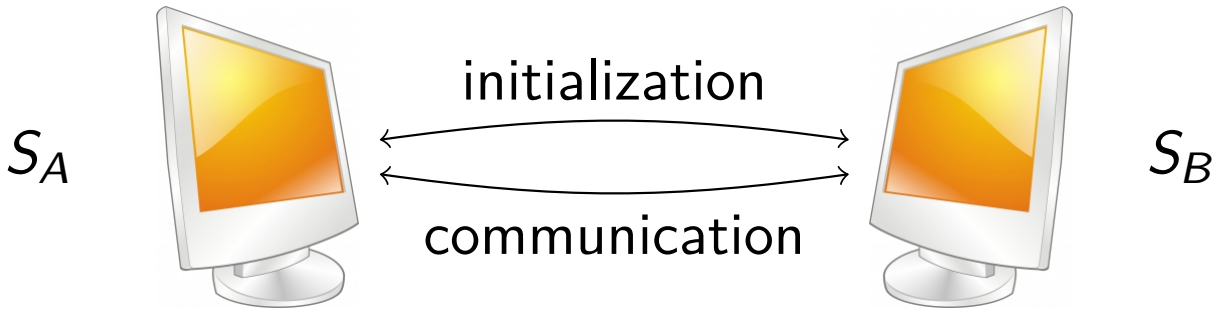
Joint work with **Ivo Kubjas**

Estonian-Latvian Computer Science Theory Days
7 May 2022

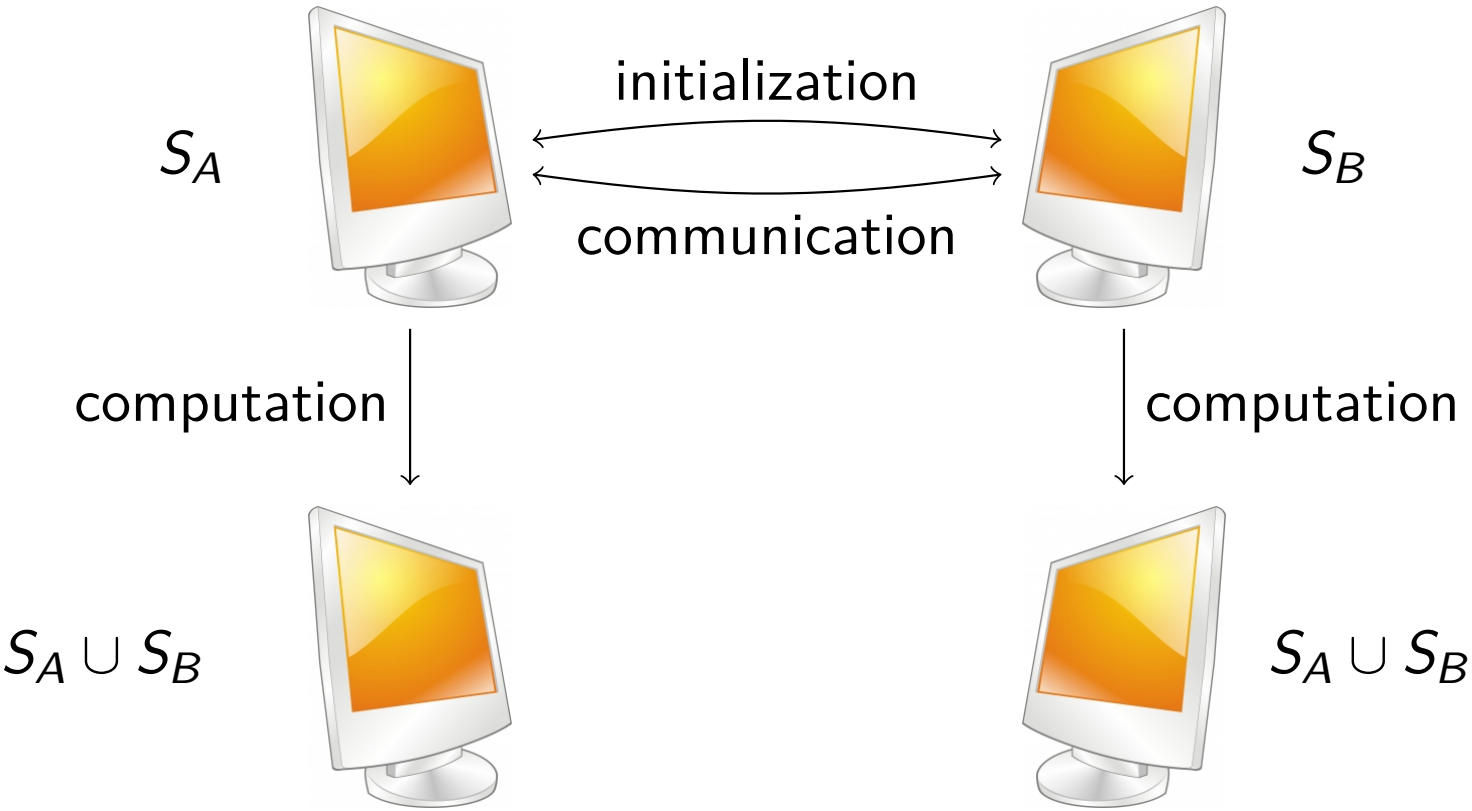
Motivation

S_A   S_B

Motivation



Motivation


S_A
initialization
communication
S_B
computation
computation
$S_A \cup S_B$
$S_A \cup S_B$

Set reconciliation problem

- Party A has set $S_A \subset \mathbb{F}^n$ and party B has set $S_B \subset \mathbb{F}^n$.
- Goal: find $S_A \cup S_B$ with the minimum number of communicated bits.

Set reconciliation problem

- Party A has set $S_A \subset \mathbb{F}^n$ and party B has set $S_B \subset \mathbb{F}^n$.
- Goal: find $S_A \cup S_B$ with the minimum number of communicated bits.

Lemma

Let $S_A, S_B \subset \mathbb{F}^n$ and denote $m = |S_A \triangle S_B|$. Then the number of transmitted bits is at least $O(mn)$.

Set reconciliation problem

- Party A has set $S_A \subset \mathbb{F}^n$ and party B has set $S_B \subset \mathbb{F}^n$.
- Goal: find $S_A \cup S_B$ with the minimum number of communicated bits.

Lemma

Let $S_A, S_B \subset \mathbb{F}^n$ and denote $m = |S_A \triangle S_B|$. Then the number of transmitted bits is at least $O(mn)$.

- Bound achieving protocol by Minsky, Trachtenberg and Zippel in 2003.
- Asymptotically bound achieving and computationally more efficient probabilistic protocol by Goodrich and Mitzenmacher in 2011.
- Multi-party set reconciliation by Boral and Mitzenmacher in 2014.

Invertible Bloom filter

An invertible Bloom filters (IBF) is a data structure that supports operations Insert(), Remove(), Test() and Extract().

Invertible Bloom filter

An invertible Bloom filters (IBF) is a data structure that supports operations Insert(), Remove(), Test() and Extract().

- IBFs are constructed using a set of hash functions $H_i \in \mathcal{H}$, $i \in [h]$, where $H_i : \mathcal{X} \to [N]$.
- Assume that all H_i can be described efficiently, and that they can be enumerated.
- Assume that the values $H_i(x)$ are distributed uniformly, i.e. for a uniform selection $x \in \mathcal{X}$ the probability of $H_i(x)$ to take any value in $[N]$ is exactly $1/N$.
- A uniform hash function $\mathfrak{B}$ is used for checksum, which is $\mathfrak{B} : \mathcal{X} \to [C]$ for a large constant $C \in \mathbb{N}$.
- We choose hash functions H_i such that

$$\forall x \in \mathcal{X} \ : \ H_i(x) \neq H_j(x) \text{ for every } i, j \in [h], \ i \neq j \ .$$

Invertible Bloom filter (cont.)

- IBF is defined as an array of cells of the form $(\texttt{count}, \texttt{val}, \texttt{ch})$.
- `count` contains an integer or a finite field element.
- `val` contains an element in $\mathcal{X}$
- `ch` contains an element in $[C]$.

Invertible Bloom filter (cont.)

- IBF is defined as an array of cells of the form $(\texttt{count}, \texttt{val}, \texttt{ch})$.
- `count` contains an integer or a finite field element.
- `val` contains an element in $\mathcal{X}$
- `ch` contains an element in $[C]$.

If we denote the i-th cell as c_i, then an IBF $\mathcal{F}$ is an array $(c_1, \ldots c_N)$.

Insertion into IBF

In order to insert an element x into an IBF $\mathcal{F}$:

Insertion into IBF

In order to insert an element x into an IBF $\mathcal{F}$:

- Insert() computes the index $j_i = H_i(x)$ for each $i \in [h]$;
- The cell c_{j_i} is updated by incrementing the field `count` by one, by adding x to the field `val`, and by adding $\mathfrak{B}(x)$ to the field `ch`.
- The total number of inserted elements into $\mathcal{F}$ is denoted as $f \triangleq |\mathcal{F}|$.

Extraction from IBF

In order to extract an element from an IBF $\mathcal{F}$:

- Extract() iterates over all cells until. g
- When it finds a cell c_j with `count` field value ± 1, and $\texttt{ch} = \mathfrak{B}(\texttt{val})$ when $\texttt{count} = 1$, and $-\texttt{ch} = \mathfrak{B}(-\texttt{val})$ when $\texttt{count} = -1$, then the value `val` is inserted into the set $\mathcal{S}_{\mathcal{F}}$, and Remove() is called.
- Remove() does the opposite of what Insert() does.
- The extraction procedure proceeds with the rest of the cells in $\mathcal{F}$. If no element is extracted while looping over the cells, then $\mathcal{S}_{\mathcal{F}}$ is returned and the procedure halts.

IBF example

i	`count`	`val`	`ch`
1			
2			
3			
4			
5			
6			
⋮	⋮	⋮	⋮
25			
⋮	⋮	⋮	⋮
N			

IBF example

i	`count`	`val`	`ch`
1			
2	1	17	$\mathfrak{B}(17)$
3			
4	1	17	$\mathfrak{B}(17)$
5			
6			
⋮	⋮	⋮	⋮
25	1	17	$\mathfrak{B}(17)$
⋮	⋮	⋮	⋮
N			

IBF example

i	`count`	`val`	`ch`
1			
2	1	17	$\mathfrak{B}(17)$
3			
4	1	17	$\mathfrak{B}(17)$
5	1	19	$\mathfrak{B}(19)$
6			
$\vdots$	$\vdots$	$\vdots$	$\vdots$
25	2	$17 + 19$	$\mathfrak{B}(17)+\mathfrak{B}(19)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	1	19	$\mathfrak{B}(19)$

Protocol by Goodrich and Mitzenmacher '2011

Protocol

1. A and B initialize $\mathcal{F}_A$ and $\mathcal{F}_B$ of size N, respectively.
2. For all $x \in \mathcal{S}_A$ and $y \in \mathcal{S}_B$, do $\mathcal{F}_A \leftarrow \text{Insert}(\mathcal{F}_A, x)$ and $\mathcal{F}_B \leftarrow \text{Insert}(\mathcal{F}_B, y)$.
3. A and B exchange $\mathcal{F}_A$ and $\mathcal{F}_B$.
4. A and B compute $\mathcal{F}_\Delta \leftarrow \text{Add}(\mathcal{F}_A, -\mathcal{F}_B)$.
5. A and B obtain $\mathcal{S}_\Delta \leftarrow \text{Extract}(\mathcal{F}_\Delta)$.
6. A and B add elements from $\mathcal{S}_\Delta$ to the sets $\mathcal{S}_A$ and $\mathcal{S}_B$, respectively.

Cooperative function computation: example

Example

- Two wireless temperature measuring sensors, which take measurements in random times.
- Each data vector has form (t, τ), where t is the time of measurement and τ is the value value.
- The average temperature is

$$f(S) = \frac{\sum_{(t,\tau)\in S} \tau}{|S|},$$

where S_A are the measurements of the first sensor, S_B are the measurements of the second sensor and $S = S_A \cup S_B$.

Cooperative function computation

- Party A has $S_A \subset \mathbb{F}^n$ and B has $S_B \subset \mathbb{F}^n$.
- Parties A and B want to cooperatively compute the function $f(S_A, S_B) = \phi(S_A \cup S_B)$.
- Generalization of set reconciliation: $\phi(S) = S$.
- Naive approach: reconcile sets and then compute the value of the function.

Cooperative function computation

- Party A has $S_A \subset \mathbb{F}^n$ and B has $S_B \subset \mathbb{F}^n$.
- Parties A and B want to cooperatively compute the function $f(S_A, S_B) = \phi(S_A \cup S_B)$.
- Generalization of set reconciliation: $\phi(S) = S$.
- Naive approach: reconcile sets and then compute the value of the function.

Example

Assume that A and B are interested in computing $\Phi_{\max}(S_A \cup S_B) = \max\{S_A \cup S_B\}$. $2n$-bit communication protocol:

1. The users A and B compute $x_A = \max\{S_A\}$ and $x_B = \max\{S_B\}$, respectively.
2. The users A and B exchange the values of x_A and x_B.
3. Each user computes $\max\{x_A, x_B\}$.

General setup

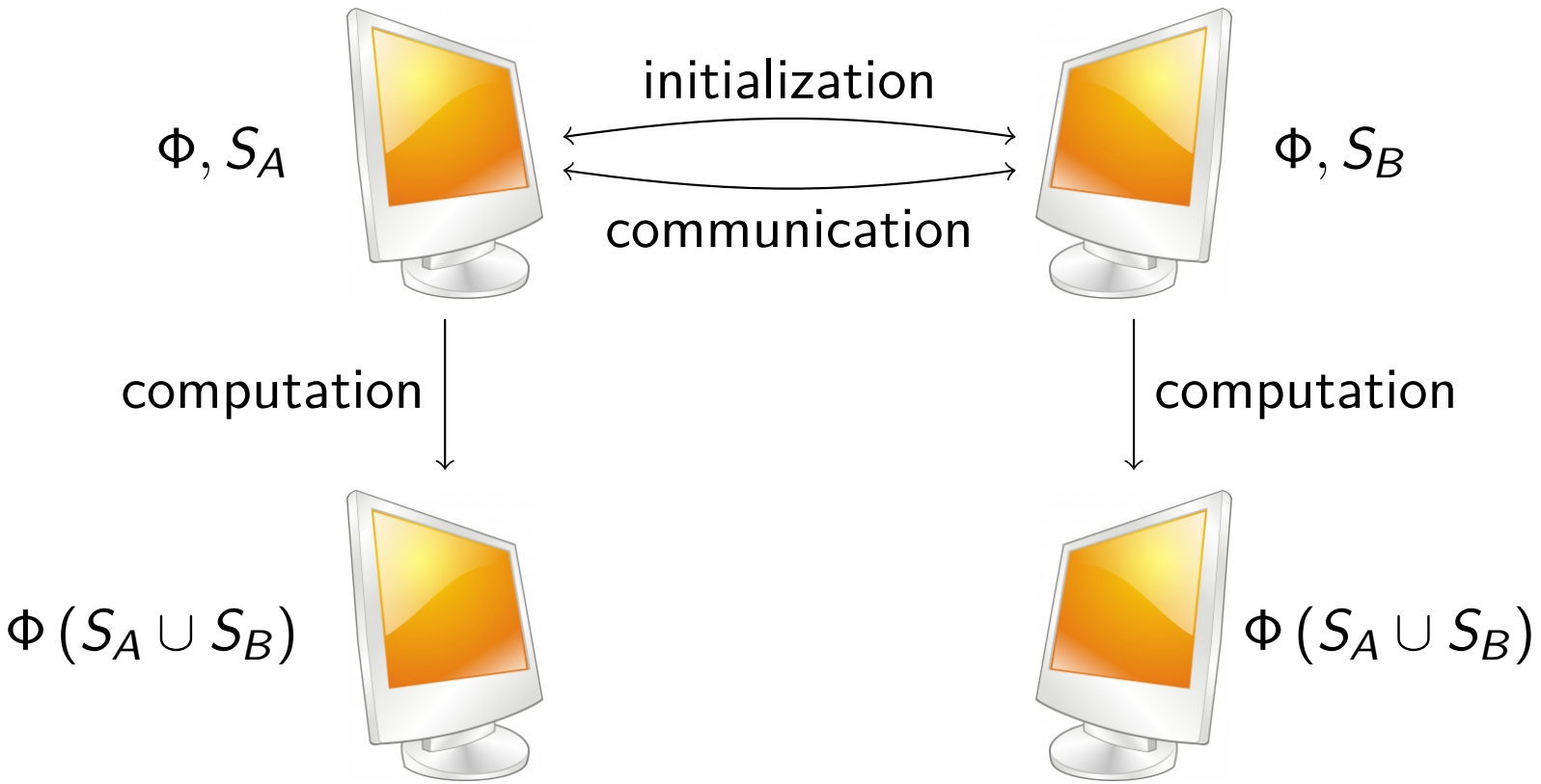


Figure: Cooperative function computation

Yao's deterministic model

- Parties A and B have $i \in M$ and $j \in N$, respectively. They want to determine the value of $f(i,j)$ by alternately sending a message at a time.
- The protocol Π consists of the messages over binary field $\mathbb{F}$:

$$S_k = (w_{k,1}, w_{k,2}, \ldots, w_{k,p_k}) \in \mathbb{F}^{p_k}, \quad k = 1, 2, \ldots, r \ .$$

- The protocol terminates if either A or B has determined $f(i,j)$.

Definition

Cost of the protocol Π is the maximum number of transmitted bits $\sum_{k=1}^{r} p_k$ over all inputs $i \in M$ and $j \in N$.

Communication complexity

Definition

Deterministic communication complexity of function f is the minimum cost over all deterministic protocols Π that compute f. We denote it as $D(f)$.

Definition

Randomized communication complexity of function f is the minimum cost over all randomized protocols Π that compute f with error ε. For shared randomness case, we denote it as $R_{\varepsilon}^{\text{pub}}(f)$, and for private randomness case, we denote it as $R_{\varepsilon}^{\text{priv}}(f)$.

Definition

Randomized communication complexity with zero errors of function f is the minimum average communication cost over all randomized protocols Π that compute f with zero error. We denote it as $R_0(f)$.

Yao's deterministic model (cont.)

- The leaves ℓ in the decision tree of the protocol Π partition the input set $M \times N$ into disjoint partitions $\{R_\ell\}$.
- A set $R = U \times V$ with $U \subset M$ and $V \subset N$ is a *(combinatorial) rectangle*.
- If f is fixed on R, then R is called f-monochromatic rectangle.

Example

	000	001	010	011	100	101	110	111
000	0	1	1	0	1	0	0	0
001	1	0	0	0	0	0	0	1
010	1	0	0	0	1	0	0	0
011	0	0	1	0	0	0	0	1
100	1	0	0	0	1	0	0	1
101	1	1	1	0	0	0	1	1
110	0	0	0	0	1	0	0	0
111	0	1	1	0	1	1	0	1

f-monochromatic rectangles

Lemma

Any deterministic protocol Π for a function f induces a partition of $M \times N$ into f-monochromatic rectangles. The number of rectangles is the number of leaves of Π.

Corollary

If any partition of $M \times N$ into f-monochromatic rectangles has at least t rectangles, then $D(f) \geq \log_2 t$.

Lower bound for sum using monochromatic rectangles

- Consider $\Phi_\Sigma(S) = \sum_{x \in S} x$ over integers.

Theorem

The number of bits communicated between A and B in any deterministic protocol Π that computes the sum function Φ_Σ is at least $D(\Phi_\Sigma) \geq 2^n + n - 1$.

	$\emptyset$	1	2	3	1,2	1,3	2,3	1,2,3
1,2,3	6	6	6	6	6	6	6	6
2,3	5	6	5	5	6	6	5	6
1,3	4	4	6	4	6	4	6	6
1,2	3	3	3	6	3	6	6	6
3	3	4	5	3	6	4	5	6
2	2	3	2	5	3	6	5	6
1	1	1	3	4	3	4	6	6
$\emptyset$	0	1	2	3	3	4	5	6

- Main diagonal 2^{2^n-1}
- Side diagonal 2^{2^n-2}
- Number of side diagonals $2^n - 1$
- At least 2^{2^n+n-2} rectangles
- Thus, the deterministic communication complexity is $\geq 2^n + n - 1$.

Other results for summation

- Upper bound $2^n + 2n - 2$:
  - $2^n - 1$ bits to represent the set
  - $2n - 1$ bits to represent the sum

Other results for summation

- Upper bound $2^n + 2n - 2$:
  - $2^n - 1$ bits to represent the set
  - $2n - 1$ bits to represent the sum
- Reducing set disjointness to sum:
  - Set disjointness function:

$$\mathrm{DISJ}(S_A, S_B) = \begin{cases} 1 & \text{if } S_A \cap S_B = \varnothing \\ 0 & \text{otherwise} \end{cases} .$$

  - Determine if sum of the union equals to the sum of the parties' sets.
  - Reduction overhead $2n$ bits.
  - Lower bound of $2^n + 1$ in the literature for deterministic protocols.
  - Asymptotically tight bound $\Theta(2^n)$ for randomized protocols.

Bounds for the summation function (overview)

Communication Complexity	Protocol Type	Comments
$\Theta(d \cdot n)$	Deterministic	Reconciliation first, difference size is d
$\geq 2^n + n - 1$	Deterministic	
$\leq 2^n + 2n - 2$	Deterministic	
$\geq 2^n - 2n + 1$	Deterministic	Reduction to set disjointness
$\Theta(2^n)$	Shared randomness	Reduction to set disjointness
$O(\kappa) + 4n$	Shared randomness	Reduction to finding the intersection, set sizes are κ
$O(\kappa) + 4n + O(\log n)$	Private randomness	Reduction to finding the intersection, set sizes are κ
$O(\kappa \cdot \log d_A + n)$	Shared randomness with zero error	Set sizes are κ, $d_A = \lvert S_A \setminus S_B \rvert$