On Distributing Symmetric Streaming Computations

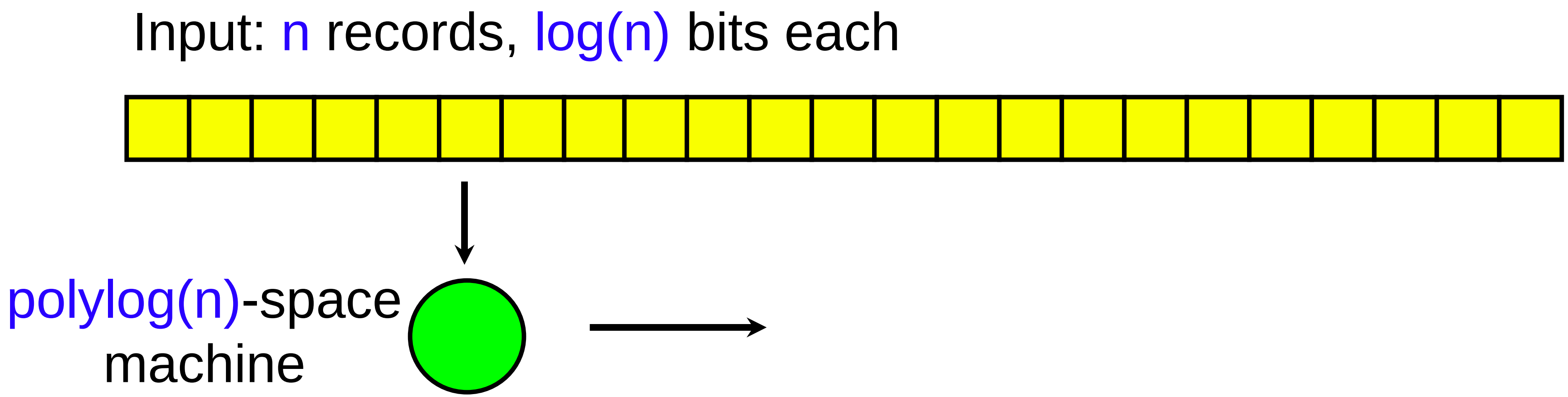
Anastasios Sidiropoulos (MIT)

Joint work with:
Jon Feldman (Google Inc)
Muthu Muthukrishnan (Google Inc)
Cliff Stein (Columbia University)
Zoya Svitkina (Dartmouth College)

Large Data Sets

- How do we deal with large data sets?
- Too much space:
  - Input does not fit into memory
  - Streaming, external memory algorithms, sampling
- Too much time:
  - Single machine cannot handle all the data
  - Parallelization

Streaming



- Simple model
- Easy to program
- Typically approximate
- Efficient computation of (simple) statistics
  [AMS99], [GGIKMS02], [Muthu03]

Parallel Computation

- Typical Model: PRAM [Fortune,Wyllie 78]

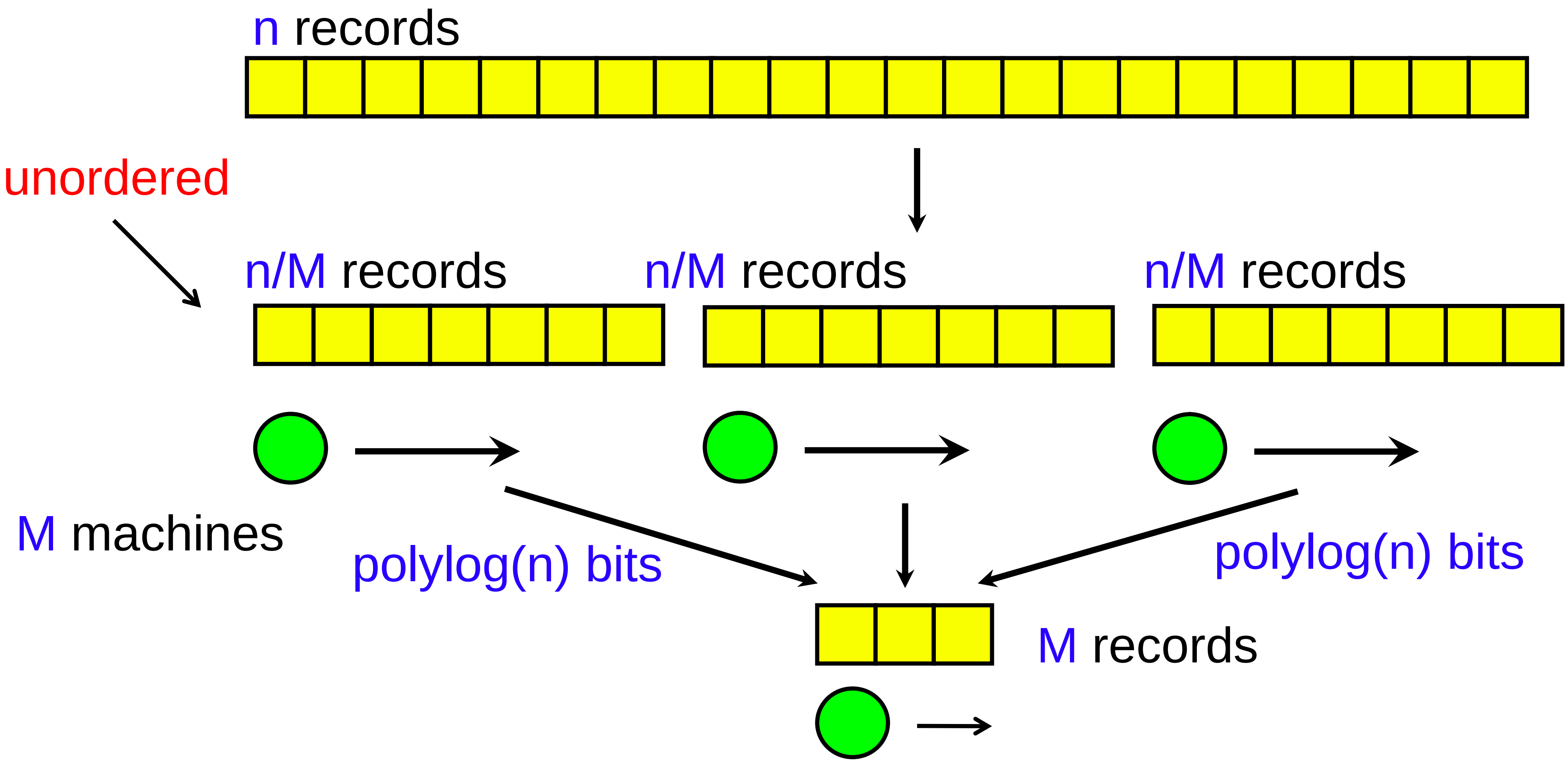
Prohibitively large communication overhead

- Also true for other models: LogP [CKPS+93], [PaYa 88], etc

Modern Distributed Computation for Large Data Sets

- Data spread arbitrarily in 1000’s of chunks.
- Many loosely coordinated machines work independently on the chunks.
- Process can iterate.
- Example: MapReduce (Google), Hadoop (Apache, Yahoo!)

MUD (Massive Unordered Distributed)



MUD vs. Streaming

How powerful is MUD?

- Streaming can simulate MUD
- Can MUD simulate Streaming?
  - YES if we make the comparison fair

MUD vs. Streaming

What is not fair?

- if we want to solve a problem that depends critically on the ordering of the input.
- E.g. “How many times does the first odd number appear in the input?”

MUD vs. Streaming

- What about symmetric problems?
- NO in general. E.g.: Symmetric-Index

Input: $(j,x_1),(j,x_2),\ldots,(j,x_n),(i,y_1),(i,y_2),\ldots,(i,y_n)$

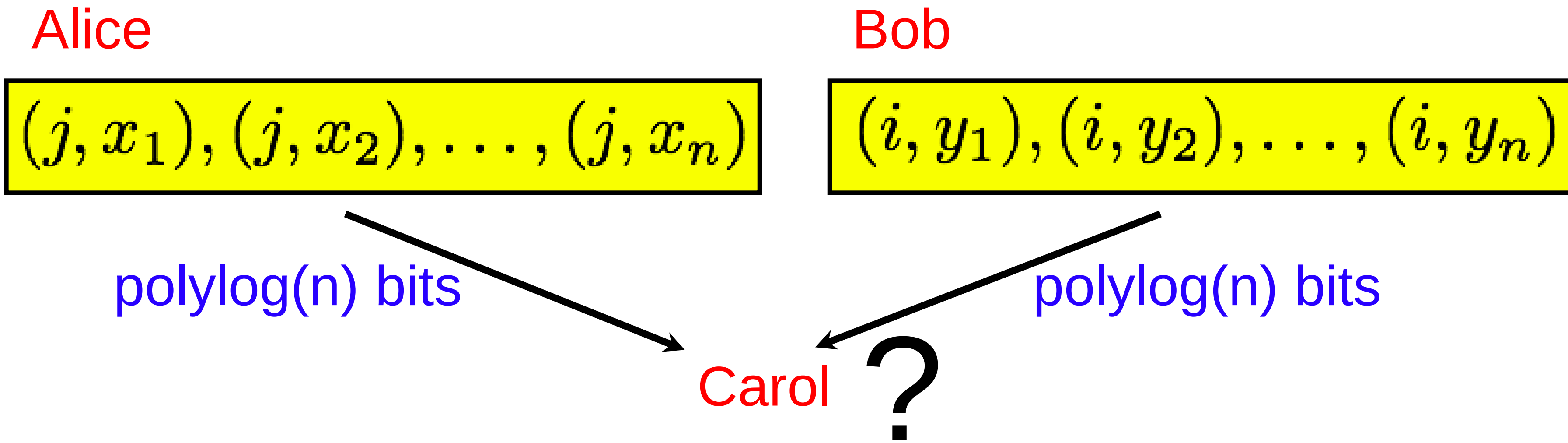
$i,j \in [n], x_k, y_l \in \{0,1\}$

s.t.: $x_j = y_i = \alpha$

Output: α

MUD vs. Streaming

- Streaming easy: Read first record (j,x), wait until you read y_j
- MUD hard: Bad instance:

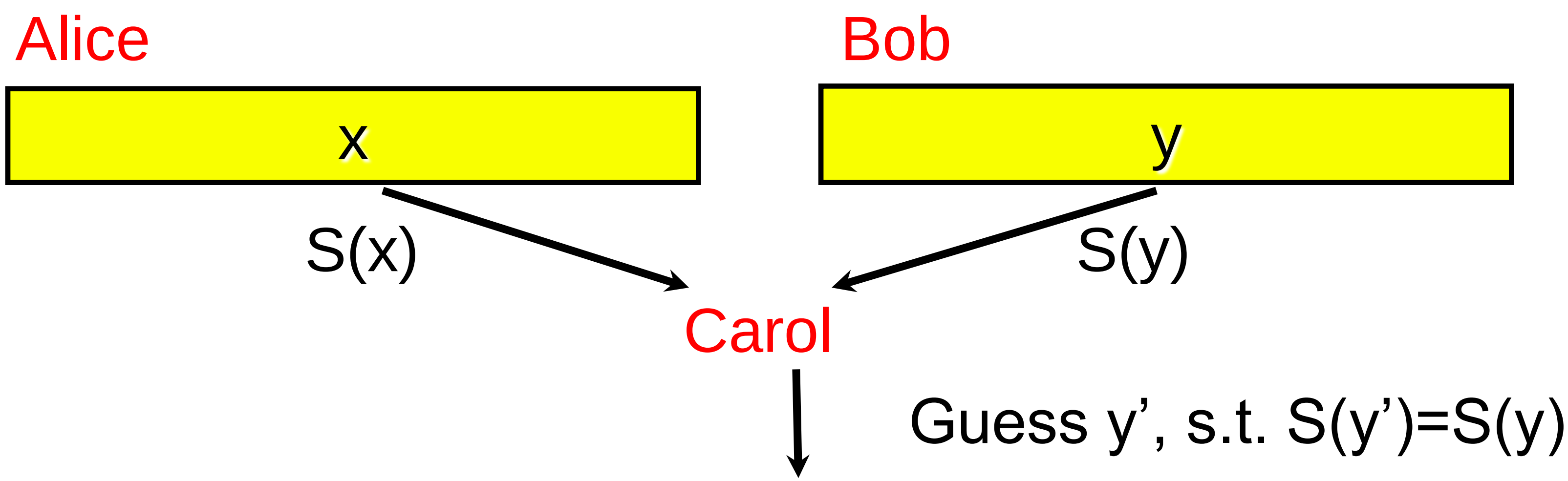


MUD vs. Streaming

- What about symmetric total single-value problems?
- YES! In this case, MUD = Streaming

MUD vs. Streaming

Streaming algorithm S



Run S with memory S(x), on input y’

Non-deterministically: Guess y’ bit-by-bit, simulate S starting with memory S(x), and S with empty memory.
If S with empty memory does not yield memory S(y), then reject.
By Savitch’s theorem, there exists polylog-space algorithm.

Correctness:

$$S^{S(x)}(y') = S(x, y') = S(y', x) = S^{S(y')}(x) = S^{S(y)}(x) = S(y, x) = S(x, y)$$

Summary

- MUD = Streaming on symmetric total functions (deterministic case)
- Also true for randomized algorithms that compute symmetric functions for any fixed randomness
- Not true for randomized algorithms, if MUD has private randomness
- Not true for partial functions
- Not true for indeterminate functions

Conclusions and Open Problems

- Can we capture more realistic scenarios?
  - E.g. different communication patterns
  - Multiple parallel instantiations / multiple labels of output
- k-round MUD vs k-pass Streaming?
- Time bounds?
- Approximation algorithms?