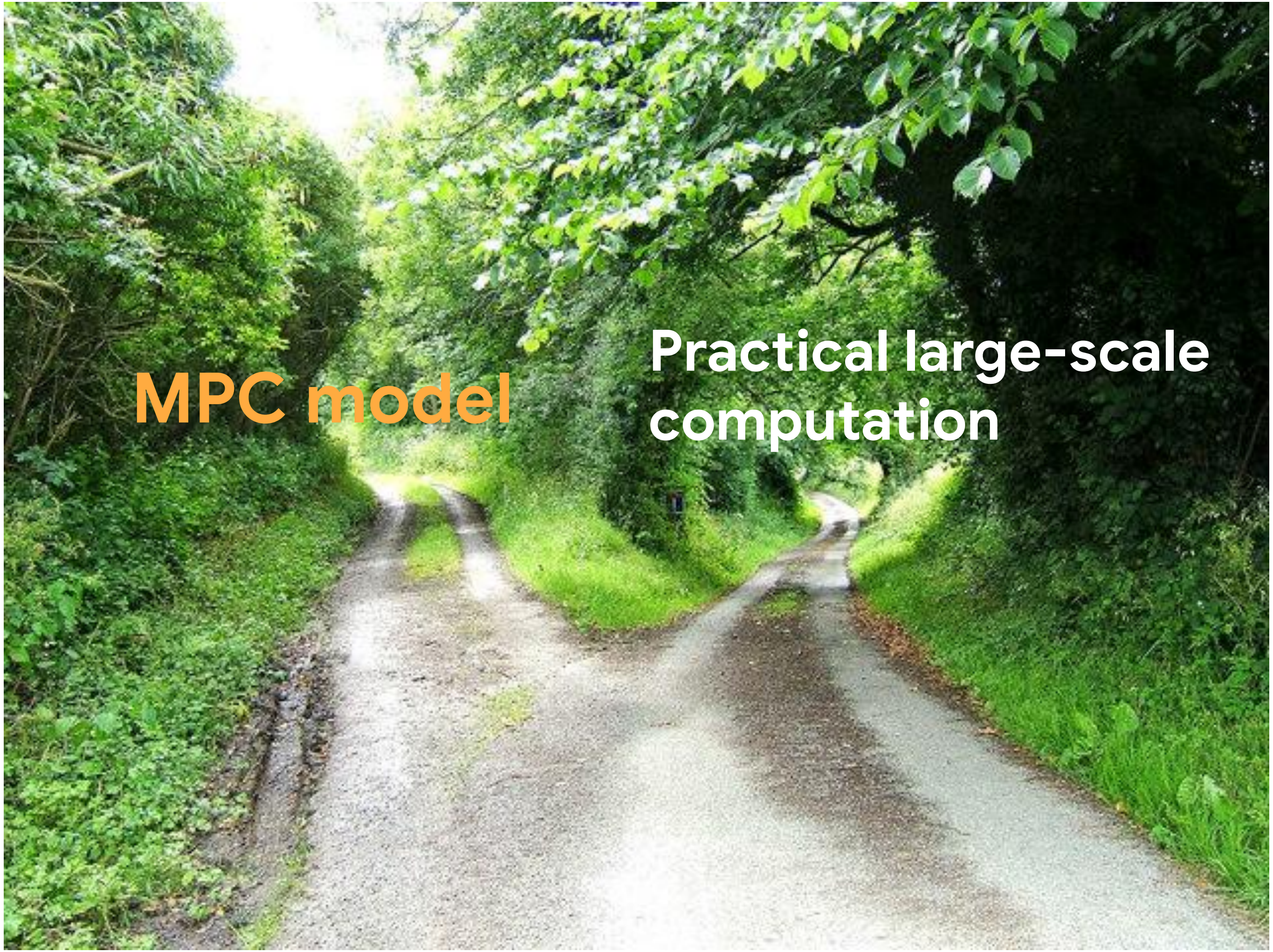
Google Research

Massively Parallel Computation
Theory and Practice

Jakub “Kuba” Łącki
Google Research

AMG workshop, 28.10.2022



https://commons.wikimedia.org/wiki/File:Fork_in_the_road_for_Brunslow_-_geograph.org.uk_-_873046.jpg

Plan of the talk

1. MPC model
   a. Motivation: MapReduce and Pregel
   b. MPC model & algorithmic results
   c. Example: efficient algorithm for finding connected components
2. AMPC model
   a. Definition
   b. Algorithmic results
   c. Empirical evaluation

Google Research

Goal:

Solve large problems fast

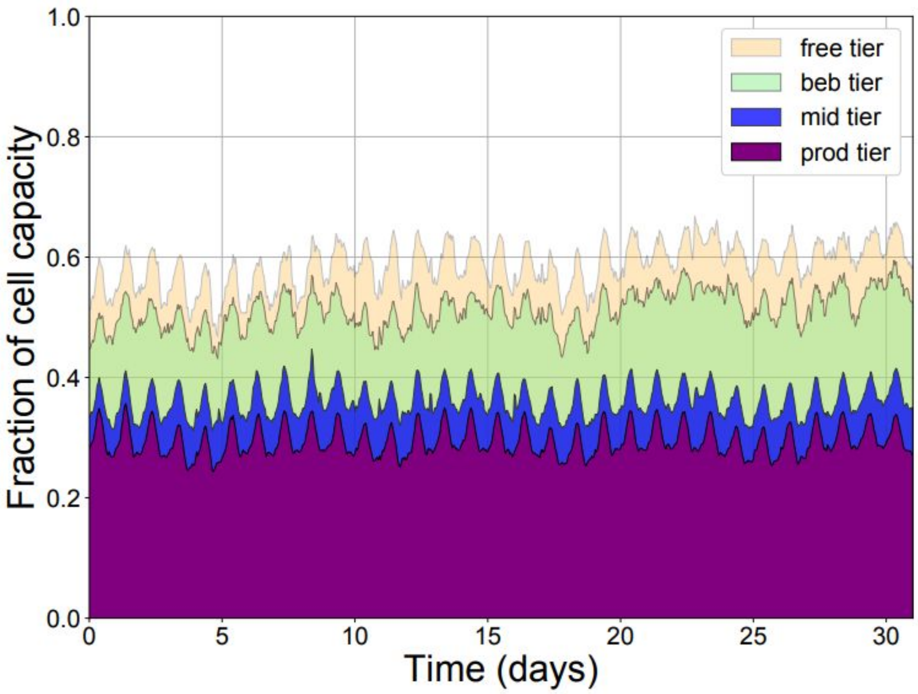
Goal:

Solve large problems fast

using simple, fault-tolerant and

cost-effective algorithms

Example: 40% of resources in a production cell are not used



Unused capacity

Low priority jobs,
May be restarted

High priority jobs,
must execute reliably

Natural idea: let's try to use the "unused" resources

Image from *Borg: the Next Generation,* Tirmazi, Barker, Deng, Haque, Hand, Qin, Harchol-Balter, Wilkes. *EuroSys'20*

Working with low-priority resources

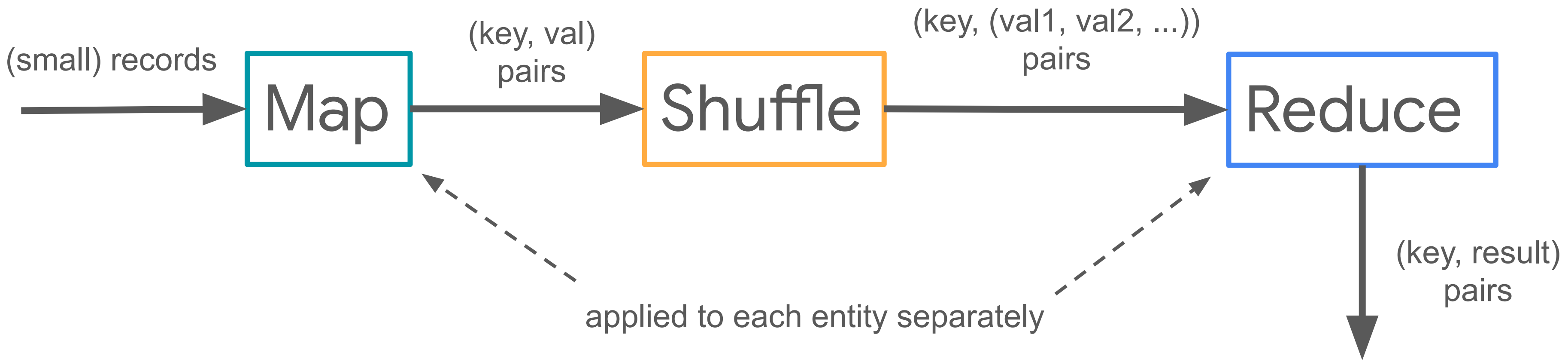
Low cost

Job may be preempted at any time

Need to ensure good fault-tolerance capabilities

MapReduce

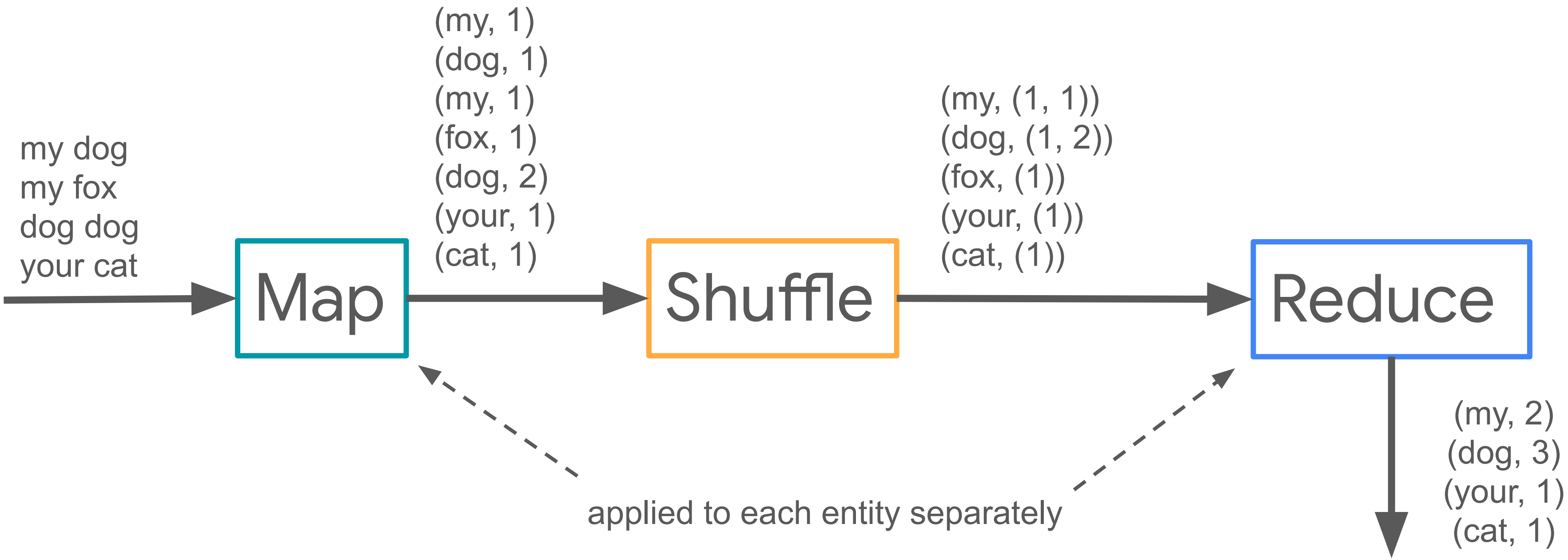
***MapReduce: Simplified Data Processing on Large Clusters*, Dean, Ghemawat, OSDI'04**



MapReduce computation is a sequence of Map/Shuffle/Reduce steps (synchronous rounds)

MapReduce example: counting words

***MapReduce: Simplified Data Processing on Large Clusters*, Dean, Ghemawat, OSDI'04**



Fault tolerance provided by the framework:

- All intermediate results replicated & saved to disk
- Each Map and Reduce runs independently; preemptions handled by restarting the computation

Google Research

Pregel

***Pregel: A System for Large-Scale Graph Processing*, Malewicz, Austern, Bik, Dehnert, Horn, Leiser, Czajkowski, SIGMOD'10**

There is a collection of **vertices**, each having:

- Its internal state
- A list of neighbors

The algorithm runs in **supersteps**. In each superstep each vertex:

- Receives messages sent in the previous superstep
- Updates its state/set of outgoing edges
- Sends messages to other vertices (to be delivered in the next step)

Goal:

Solve large problems fast

using simple, fault-tolerant and

cost-effective algorithms

Very easy to scale horizontally

Simple API

Provided for free

Thanks to fault-tolerance we can use low-priority resources

Google Research

Is the problem solved?

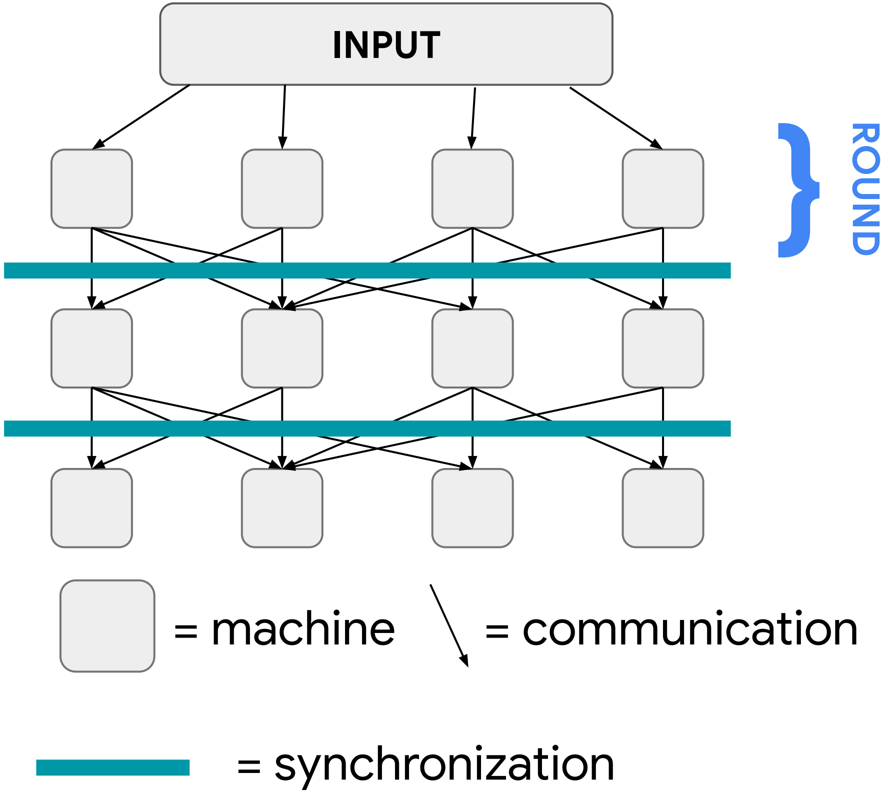
Well, we also need **algorithms**

How do we know an algorithm is good?

- Low number of steps
- Low amount of data shuffled
- No reducer is overloaded with data

MPC model (Massively parallel computation) [KSV'10, GSZ'11, BKS'17]

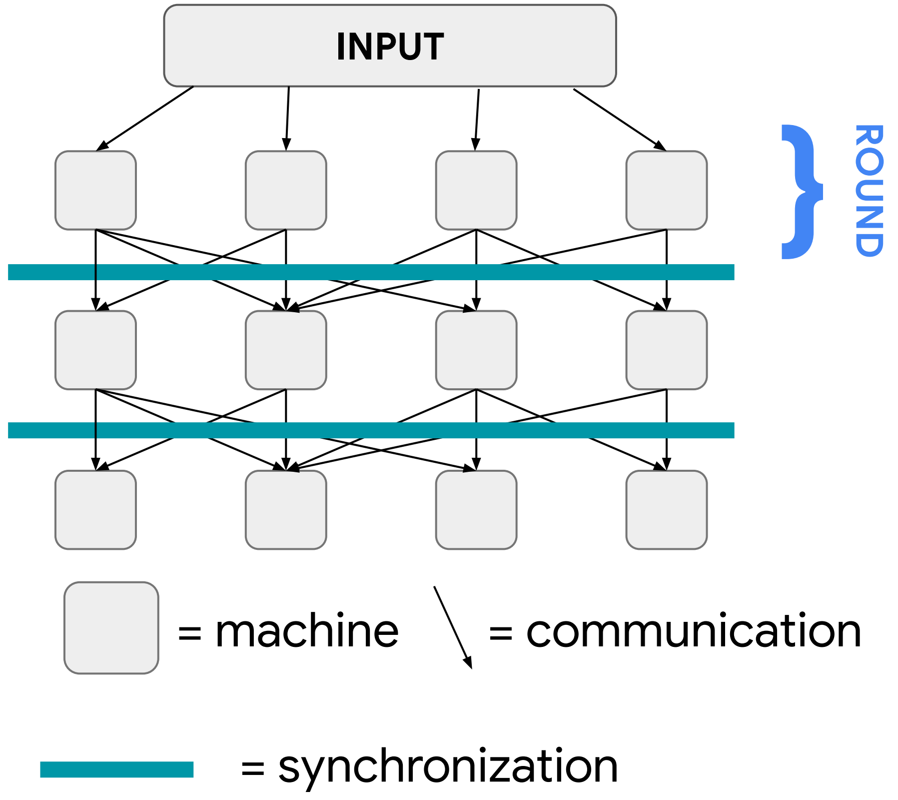
- Computation in **synchronous** rounds
- In each round, a machine:
    1. **Receives** messages from the previous round
    2. **Performs** arbitrary computation
    3. **Sends** messages to other machines



MPC model (Massively parallel computation) [KSV’10, GSZ’11, BKS’17]

- Input of size N
- P machines with space S
- $N = \Theta(P^*S)$
- **$S = N^{\varepsilon}$ for some $\varepsilon \in (0, 1)$**
- **Each machine sends / receives data of size S in a round**

Goal: minimize #rounds



MPC model - discussion

- Can you “cheat” by performing arbitrary computation?
  - Most algorithms use near-linear time
  - Still arbitrary computation is useful e.g. in derandomization
- Are machines stateful?
  - Stateful & stateless are equivalent

How powerful is MPC?

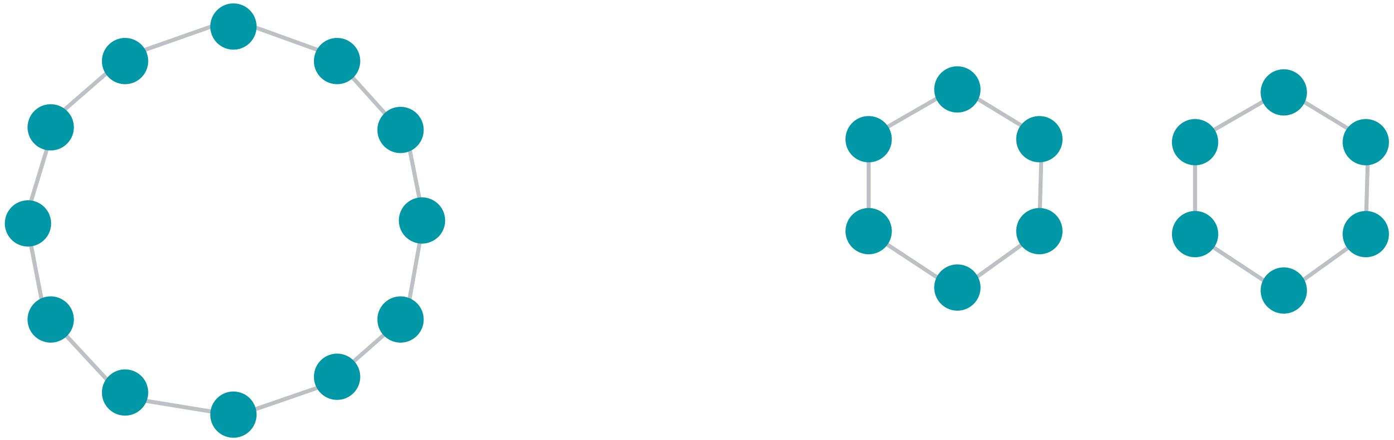
MPC vs PRAM

- MPC can often simulate PRAM
- MPC can be more powerful than PRAM
  - Computing XOR requires $\Omega(\log n)$ depth in PRAM but only $O(1/\varepsilon)$ MPC rounds

MPC lower bounds

- Computing OR requires $\Omega(1/\varepsilon)$ rounds
- Most commonly used: $\Omega(\log n)$ conditional lower bound

MPC model - hardness (1-vs-2-cycle problem)



Distinguish between a

- cycle on 2n nodes and
- two cycles on n nodes

Conjecture: this requires $\Omega(\log n)$ rounds in MPC model

Google Research

MPC: three classes of graph algorithms

$S = O(n^{1+\varepsilon})$

for some constant $0 < \varepsilon < 1$

$S = O(n)$

$S = O(n^{\varepsilon})$

for some constant $0 < \varepsilon < 1$

Only make sense
when $m = n^{1+\Omega(1)}$

Very similar to
CONGESTED-CLIQUE

Best scalability
Most challenging

Graph algorithms in MPC - three different regimes

Problem	$O(n^{1+\varepsilon})$	$\tilde{O}(n)$	$O(n^{\varepsilon})$
Connected components	$O(1)$	$O(1)$	$O(\log D + \log\log n)$
Minimum spanning tree	$O(1)$	$O(1)$	$O(\log n)$
Maximal matching	$O(1)$	$O(\log\log n)$	$\tilde{O}(\mathrm{sqrt}(\log n))$
Maximal independent set	$O(1)$	$O(\log\log n)$	$\tilde{O}(\mathrm{sqrt}(\log n))$
$(\Delta+1)$-coloring	$O(1)$	$O(1)$	$O(\log\log\log n)$
PageRank	?	?	$O((\log\log n)^2)$

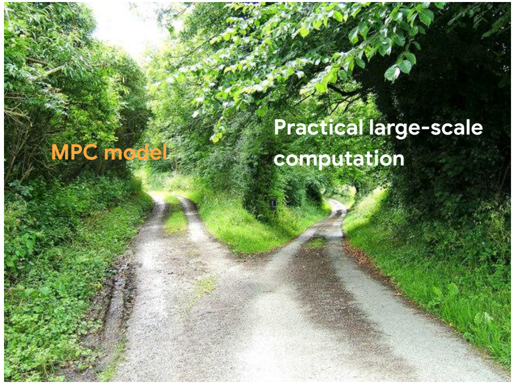
D = graph diameter

Can we now solve large problems fast?

Google Research

We have good frameworks, model and algorithms, but:

- Are the algorithms easy to implement?
- What is the hidden constant? (50 log log n vs log n rounds)
- What is the amount of communication and local computation?



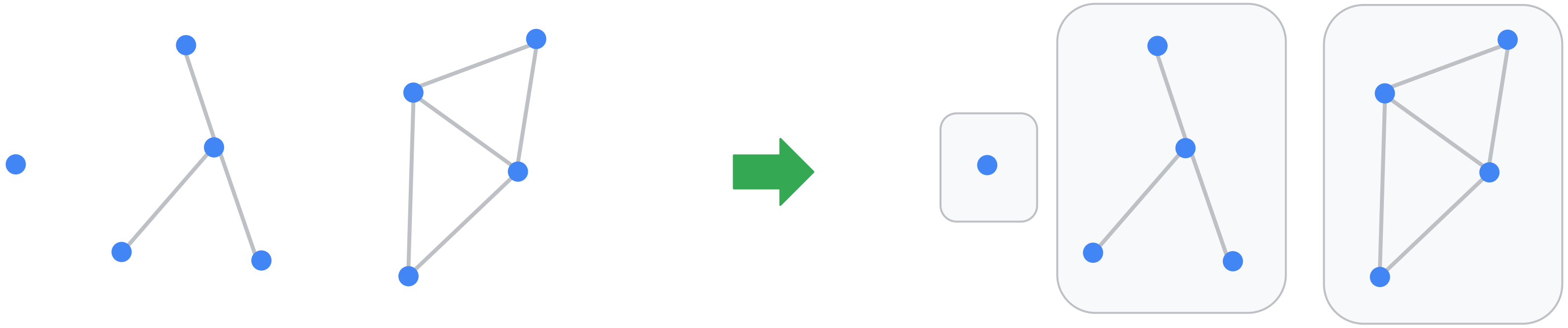
Many MPC algorithms are impractical.
Still, MPC is a great model to develop practical algorithms

Example: connected components

Google Research

Input: a graph $G = (V, E)$

Output: component(v) for each $v \in V$



Example: connected components

```
while |E(G)| > 0
    for v ∈ V(G)
        label(v) := U[0, 1]
        best(v) := 2-hop neighbor of v minimizing label(w)
    group nodes by best(v) and merge together
```

Claim

Let d := minimum degree in G^2

Then, the number of nodes shrinks by a factor of $\tilde{\Omega}(d)$ in expectation

Performance on random graphs

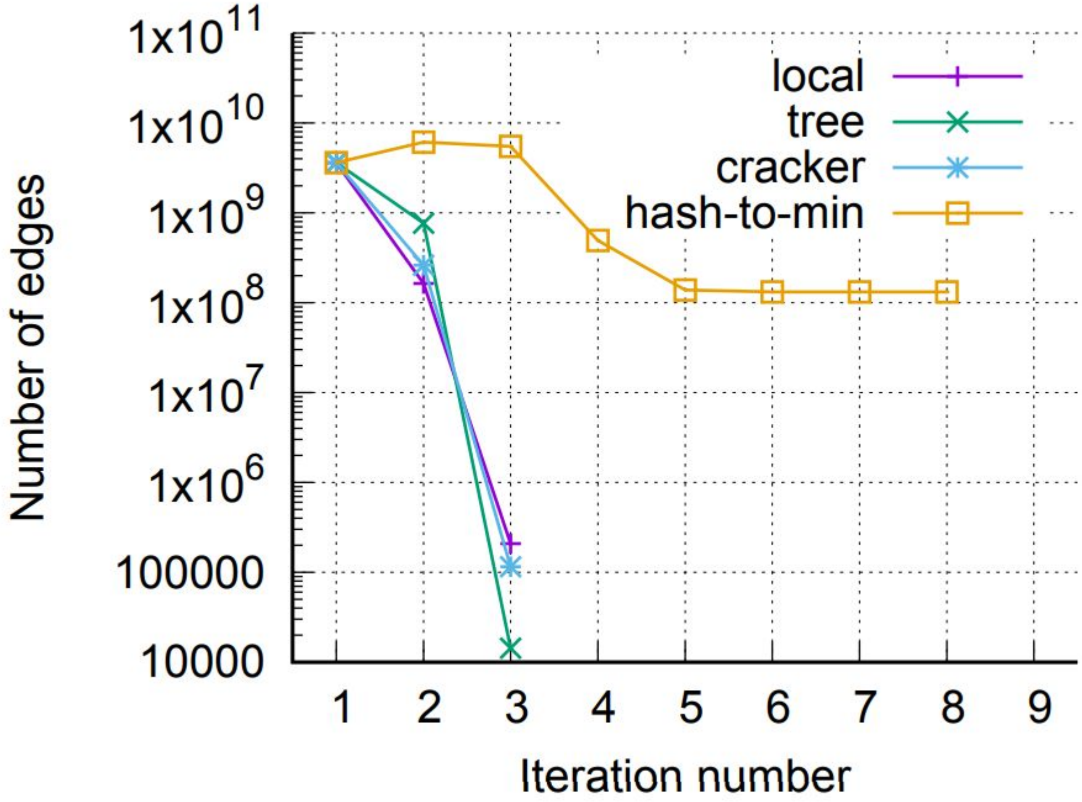
Theorem [Ł.MW'19]

Let H ~ ErdosRenyi(n, c log n / n). Assume that H $\subseteq$ G. Then the (modified) algorithm finds connected components in H in O(log log n) rounds.

[ASW, PODC'19] showed that O(log log n) rounds are possible if spectral gap >= 1/polylog n

Number of edges decreases by ~10x in each iteration

Google Research



Empirical Performance - relative running times

Google Research

Graph (#edges)	Orkut (117M)	Friendster (1.8B)	Clueweb (37.3B)	videos (626B)	webpages (6.5T)
New	1.0	1.0	1.0	1.03	1.0
Cracker	1.38	1.16	2.65	1.0	~3.0
Two-phase	5.77	1.73	1.77		
Hash-to-min	5.84	20.27			

Google Research

AMPC model

Adaptive massively parallel computation

Massively Parallel Computation via Remote Memory Access, Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Łącki, Vahab Mirrokni, Warren Schudy, SPAA'19

Parallel Graph Algorithms in Constant Adaptive Rounds: Theory meets Practice. Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Łącki, Vahab Mirrokni, Warren Schudy, VLDB'20.

AMPC = a combination of MPC and a distributed hash table

Google Research

Distributed hash table (DHT, a.k.a. Key-value store)

- Service storing (key, value) pairs
- Query provides a key and returns the corresponding value(s)

Lookup latency as low as 1-3 µs (~20x slower than RAM)

Previous applications of DHT

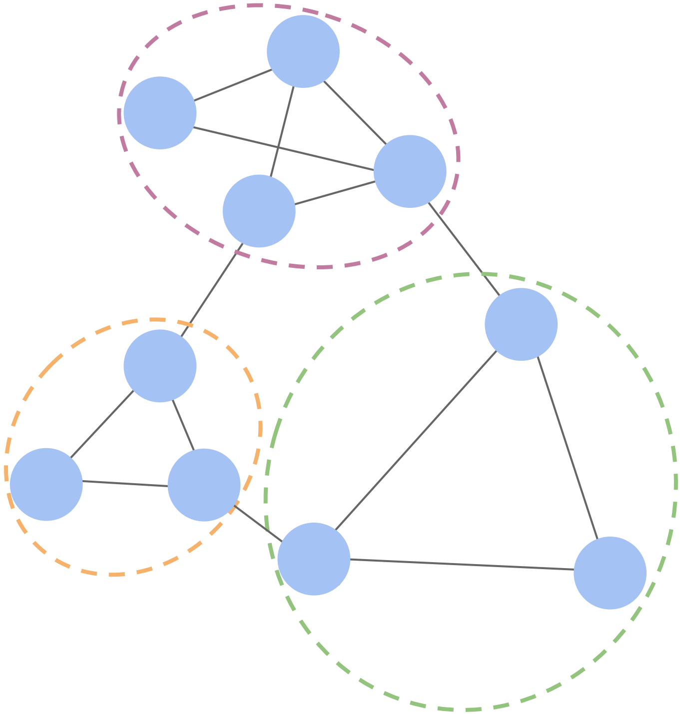
Affinity clustering [BBDHKLM, NeurIPS'17]

- Allows O(1)-round implementation

Connected components [KLMRV, SOCC'14]

- Used in a previous SOTA implementation

The applications rely on the input being "nice"



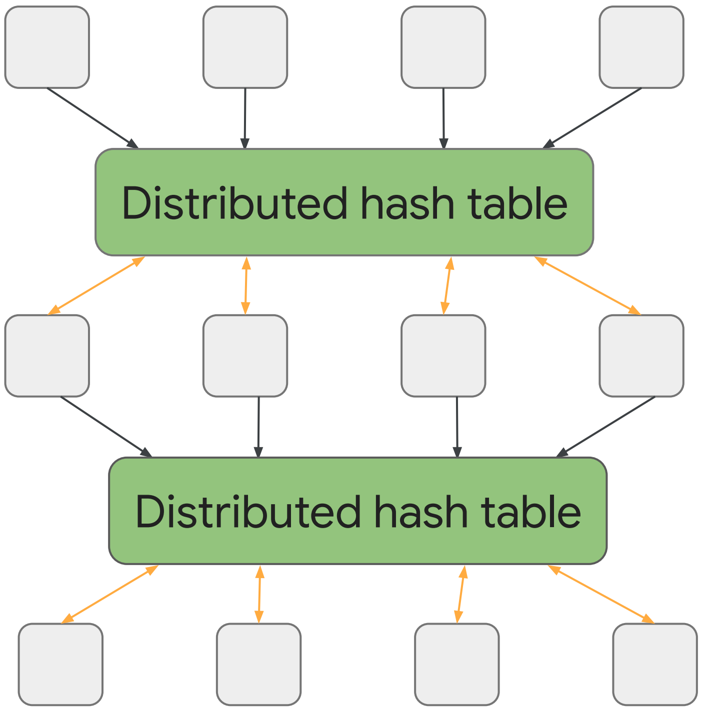
Adaptive Massively Parallel Computation (AMPC) - definition

N, P, S defined as in MPC

Differences

- All messages saved to a distributed hash table (DHT)
- In the following round each machine can *adaptively* read S values from the DHT

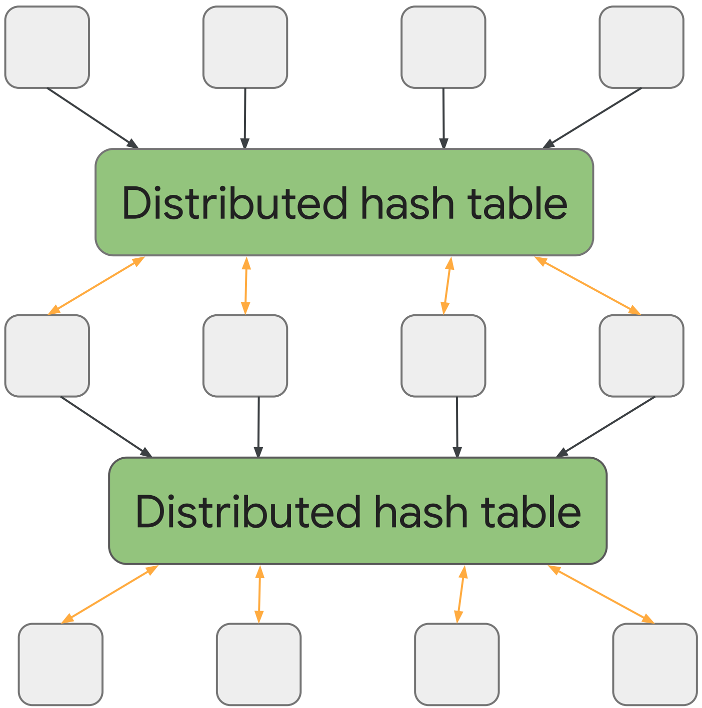
Same bounds on communication



AMPC - properties

- Fault tolerance
  - Use a fault tolerant DHT
  - A failing machine can just restart

- Allowing writes?
  - Technically possible
  - Ensuring fault-tolerance becomes much more challenging

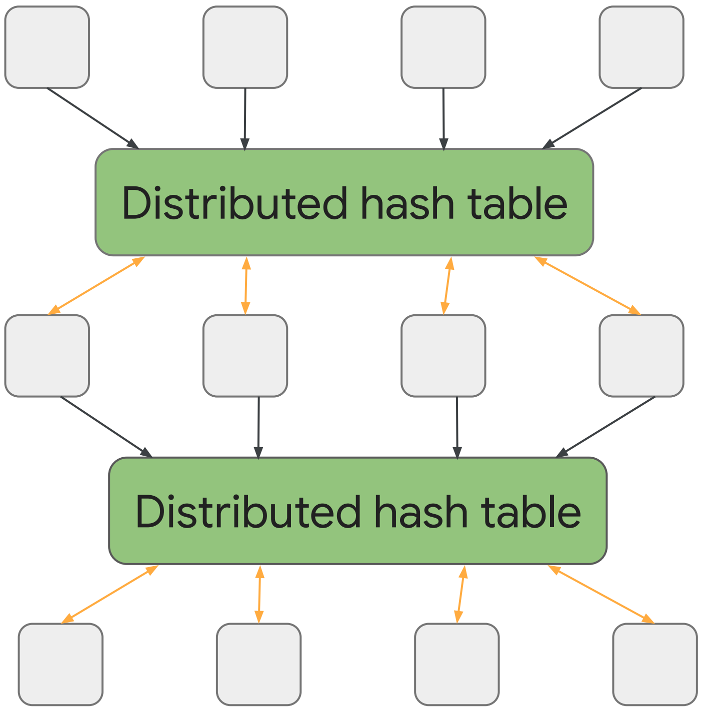


Google Research

AMPC - realism

- Slow "chains" of reads?
  - Very low read latency (1-3 µs)

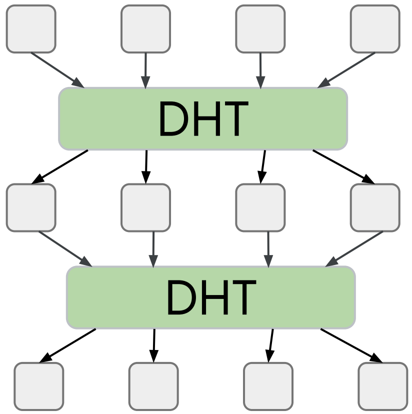
- Read contention
  - No contention under natural assumptions:
    - $P = O(N^{0.5})$
    - Random sharding
    - Caching of lookup results



AMPC - 1-vs-2-cycle problem

Algorithm idea

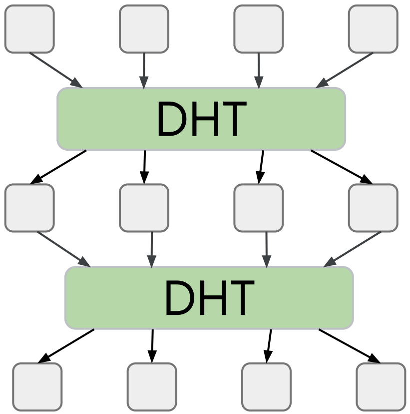
- Repeatedly shrink each cycle by a factor of $n^{\Omega(1)}$ by contracting edges
- After O(1) rounds, the graph fits on a single machine

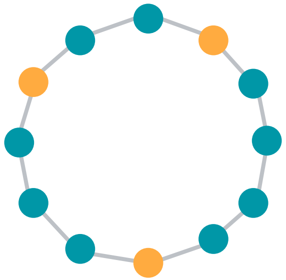


AMPC - 1-vs-2-cycle problem

How to shrink the cycle?

- Write node -> (neighbor1, neighbor2) entries to the DHT
- Sample each node w.p. $n^{-\Omega(1)}$
- **Each sampled vertex can find its two nearest sampled neighbors in 1 round**





1-vs-2-cycle problem solvable in O(1) rounds

AMPC is strictly stronger than (the model of) MapReduce, Hadoop, Pregel, Giraph, ...

(assuming 1-vs-2-cycles conjecture)

AMPC algorithms

Parallel Graph Algorithms in Constant Adaptive Rounds: Theory meets Practice. Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Kuba Łącki, Vahab Mirrokni, Warren Schudy, VLDB'20.

AMPC - Graph Algorithm in *constant* rounds

Problem	**MPC**	**AMPC**
Maximal Independent Set	$\tilde{O}(\text{sqrt}(\log n))$	O(1)
Connected Components	O(log D)	O(1)
Minimum Spanning Tree (MST)	O(log n)	O(1)
Approximate matching	$\tilde{O}(\text{sqrt}(\log n))$	O(1)
1-vs-2-cycles	O(log n)	O(1)

D = graph diameter
Assumption: graph has at least $n^{1+\varepsilon}$ edges

AMPC - results for graphs with O(n) edges

Problem	#rounds	Total space
Connected components	O(log log n)	O(n)
Connected components	O(1)	O(n log n)
Minimum Spanning Tree (MST)	O(log log n)	O(n)
Forest connectivity	O(1)	O(n log log n)
Approximate matching	O(1)	$O(n^{1+\varepsilon})$

AMPC model - hardness

Unconditional Lower Bounds for Adaptive Massively Parallel Computation. Moses Charikar, Weiyun Ma, Li-Yang Tan, SPAA'20

Theorem

The 1-vs-2-cycle problem requires $\mathbf{\Omega(1/\varepsilon)}$ rounds in the AMPC model with n^{ε} space per machine.

Google Research

AMPC model - new algorithms

Problem	**#rounds**	**Reference**
Maximum independent set, maximum matching, isomorphism testing **on trees**	O(1)	[HKSS, ITCS'22]
(2+ε)-approximate min cut	O(log log n)	[HKOS, SPAA'22]
Maximal matching	O(1), optimal total space	[B, FOCS'21]

Implementing the AMPC model

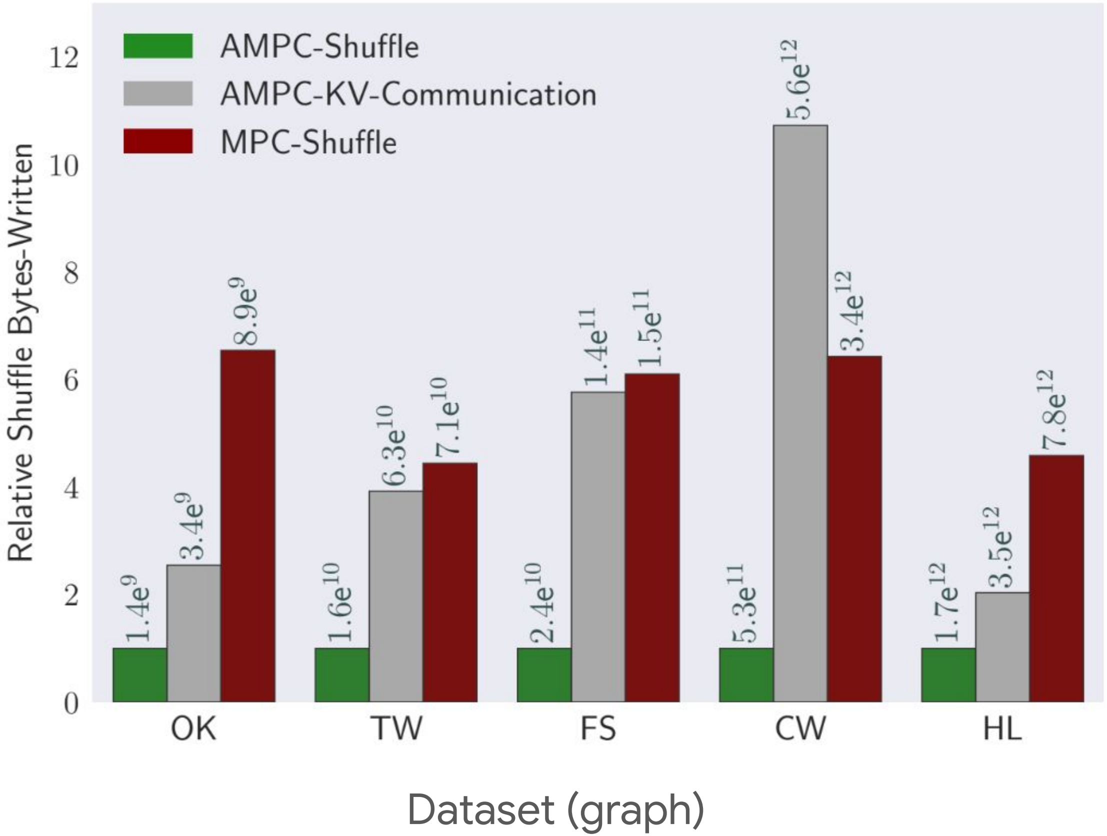
- Starting point: Flume-C++ (MapReduce - like framework)
- Existing distributed hash table implementation
  - Uses RDMA
- Bulk of communication is using shuffles (“the regular way”)
  - DHT used when needed

Adaptive MPC - empirical results

Problem	MPC rounds	AMPC rounds	AMPC Speedup
Minimum spanning forest	33-84	5	2.6x - 7.2x
Maximal independent set	8-14	1	2.3x - 3x
Maximal matching	8-16	1	1.16x - 1.7x

5 graphs of up to 225B edges

AMPC - communication in the MIS implementation



Google Research

Summary

- MPC
  - theory model of modern large-scale computation
  - became one of widely accepted theory models
  - very helpful in designing practical algorithms

- AMPC
  - AMPC := MPC + a distributed hash table
  - Many graph problems are solvable in $O(1)$ rounds using simple algorithms