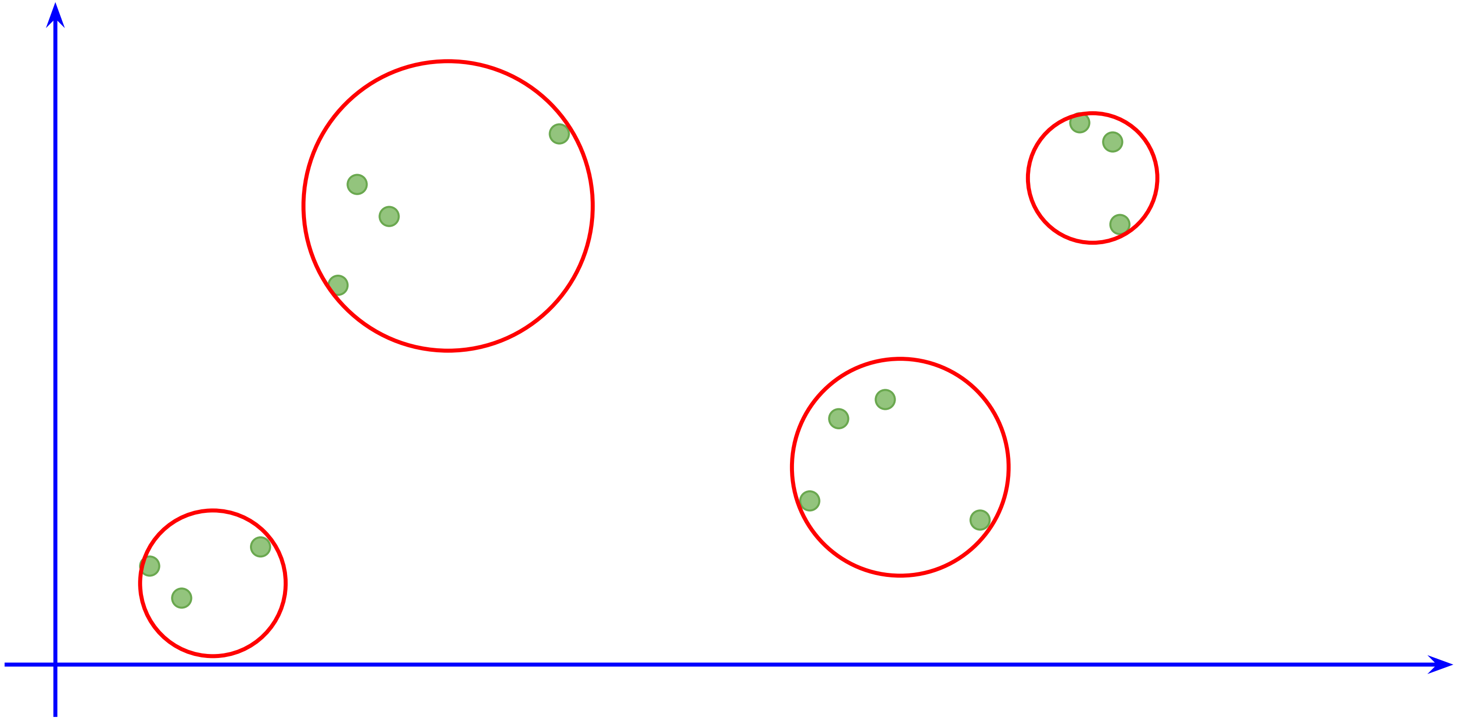
Detailed Analysis and Optimization of CUDA K-means Algorithm

Martin Kruliš, Miroslav Kratochvíl
Department of Software Engineering, Charles University, Prague Czech Republic

ICPP 2020

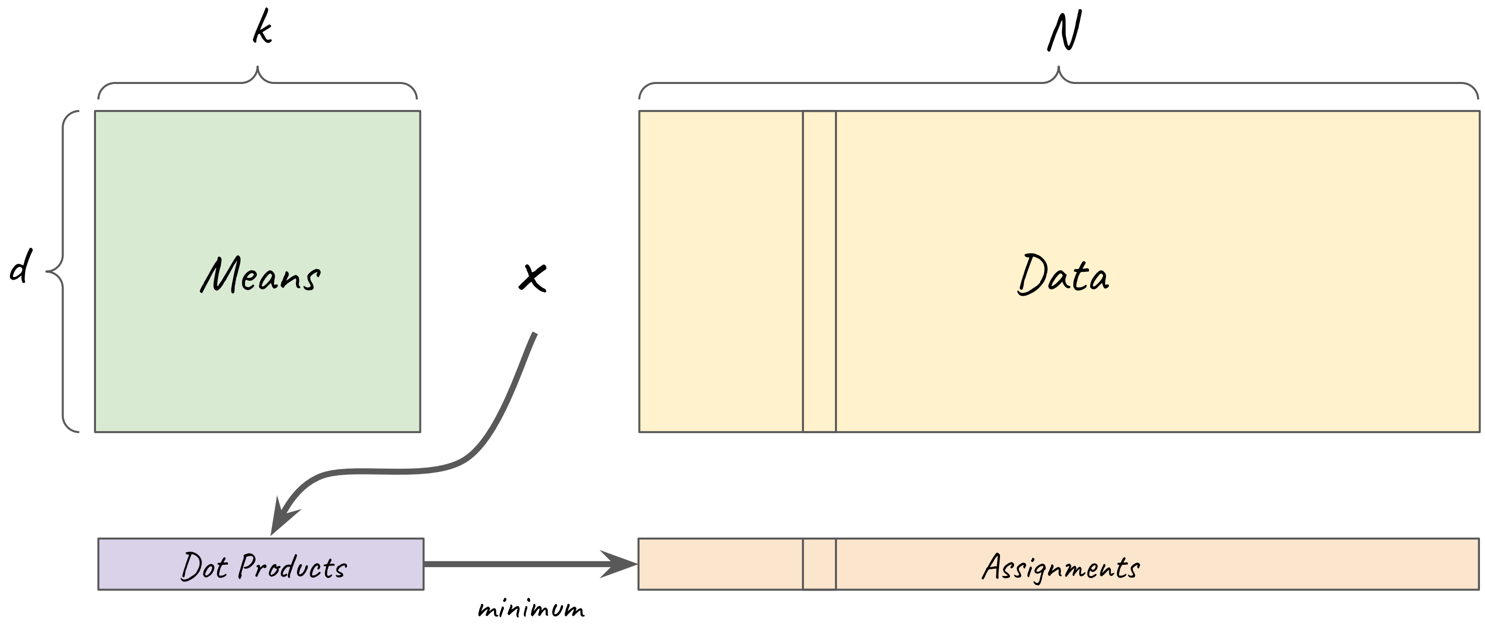
Quick Refresher: K-means Algorithm



Quick Refresher: K-means Algorithm

- Iteratively update set of centroids (means)
  - Compute point assignment
    - Compute Euclidean distance between every point and every mean
    - Find nearest mean (minimum of distances) for each point
  - Update means (per-dimension average)
    - Compute sum of coordinates (per dimension) for each assigned point
    - Divide each sum by the number of points in the corresponding cluster

Quick Refresher: K-means Algorithm



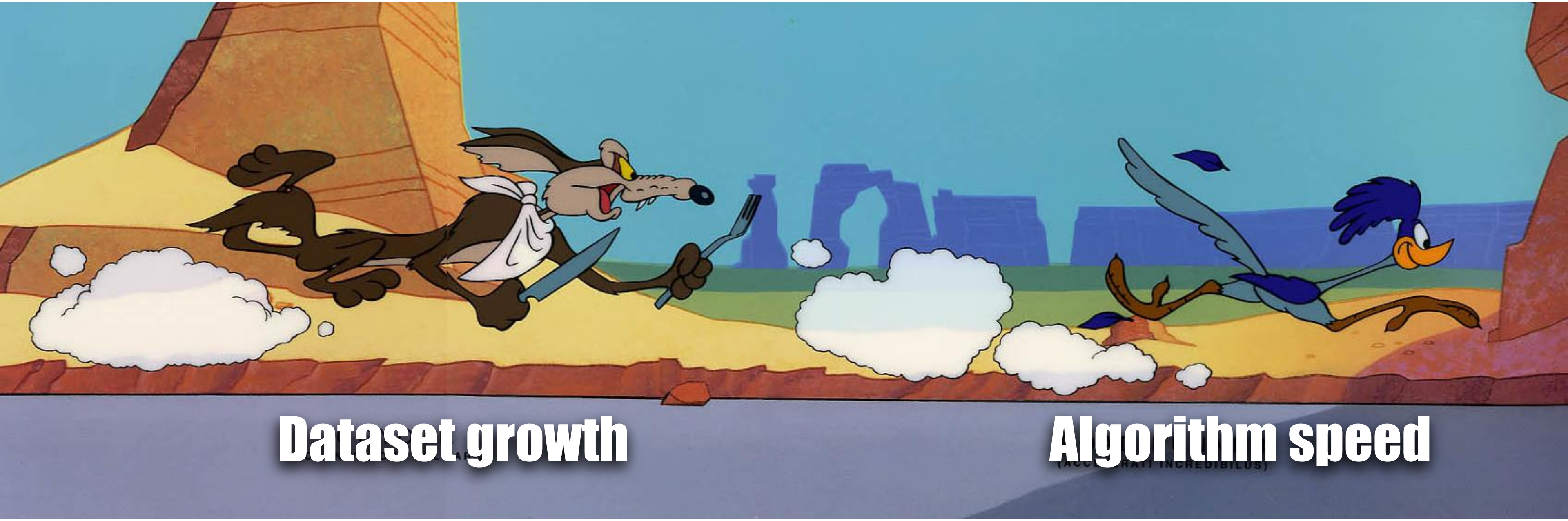
Quick Refresher: K-means Algorithm

- Iteratively update set of centroids (means)
  - Compute point assignment
    - Compute Euclidean distance and the means-wide reduction (minimum)
  - Update means (per-dimension average)
    - Compute sum of coordinates (per dimension) for each assigned point
    - Divide each sum by the number of points in the corresponding cluster

Quick Refresher: K-means Algorithm

- Iteratively update set of centroids (means)
  - Compute point assignment
    - Compute Euclidean distance and the means-wide reduction (minimum)
    - Add point coordinates (per dimension) to its nearest cluster
  - Update means (per-dimension average)
    - Divide each sum by the number of points in the corresponding cluster

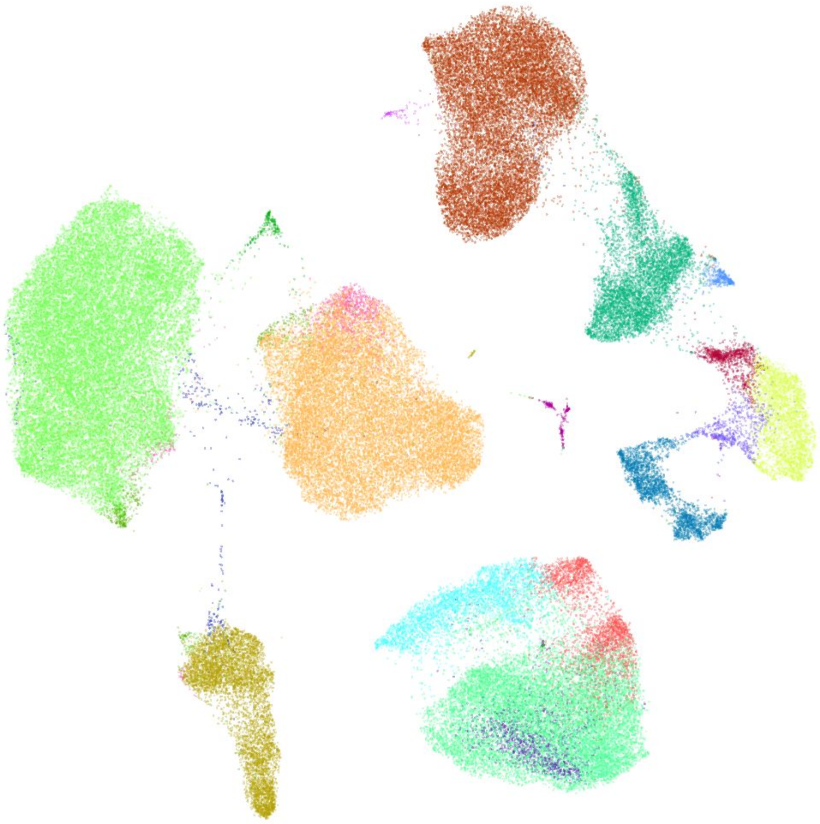
Why k-means again?



High-performance use cases

Meta-clustering single-cell data

- <50 dimensions
- Millions of data points
- Time available: a few seconds (the analysis is interactive)



UMAP projection of 32-dim data to 2D

High-performance use cases
Video browsing & retrieval

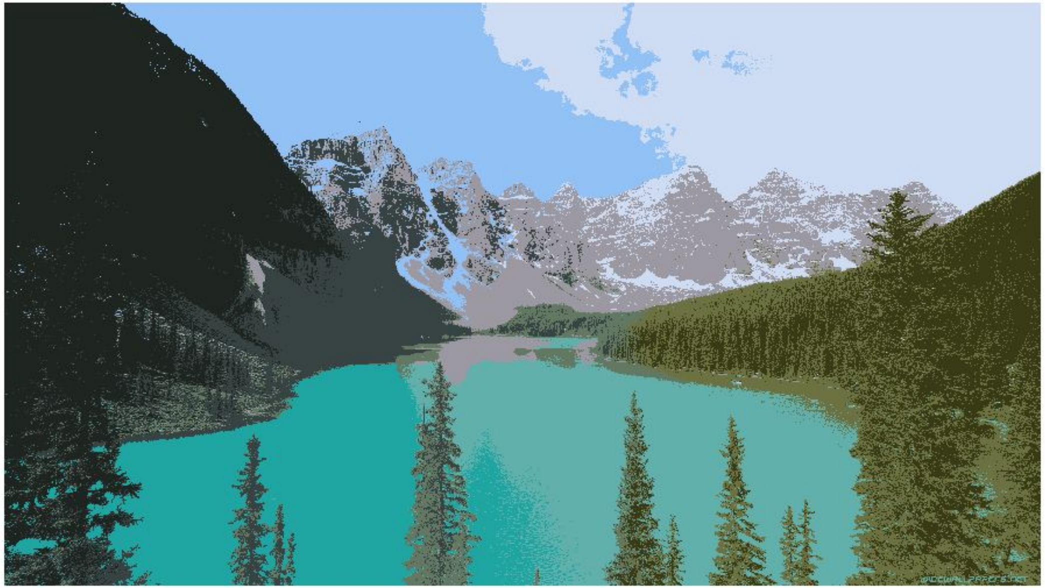
- ~1000 dimensions from a neural net
- Millions of data points
- Time available: <1s

SOMHunter, interactive video retrieval system
(VBS competition winner at MMM2020)

High-performance use cases

Real-time video super-pixel segmentation in Full HD

- <10 dimensions
- ~2 million data points
- Time available: ~50ms



News in CUDA-kmeans: Raw performance

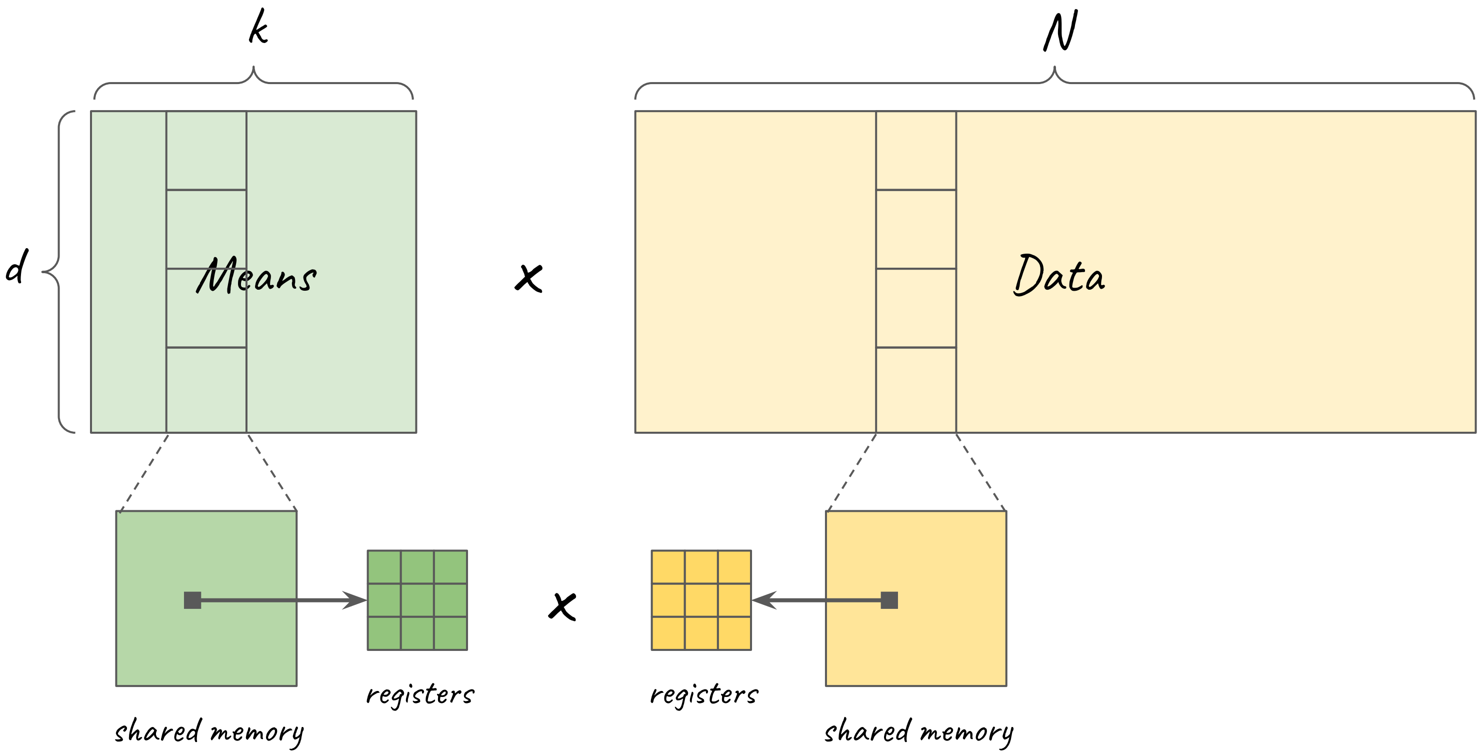
2 million data points, 32 dimensions, time per iteration:

	16 clusters	1024 clusters
nVidia GTX 980	**7.31 ms** = 136 ips	**104.84 ms** = 9 ips
nVidia V100 SXM2	**1.58 ms** = 632 ips	**25.45 ms** = 39 ips

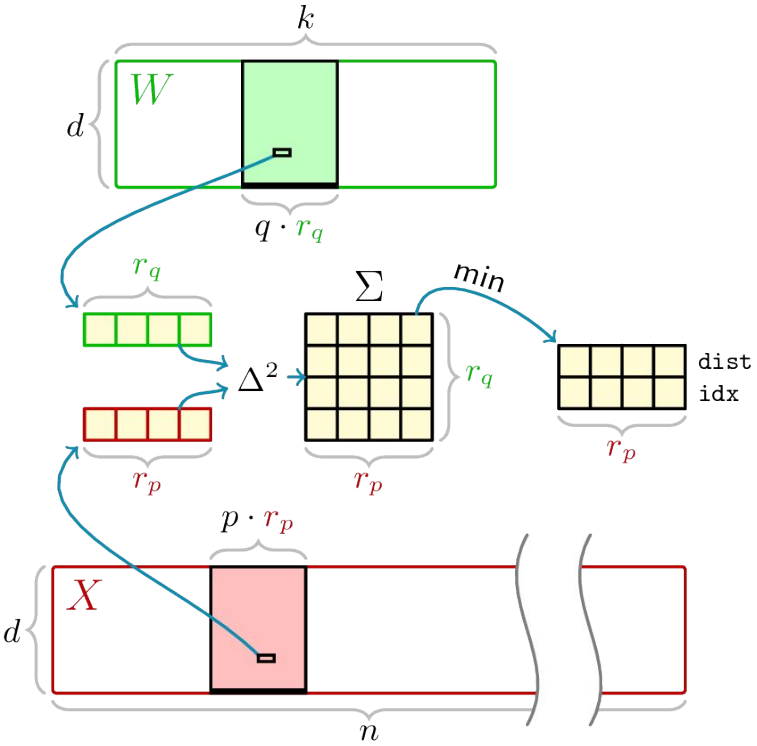
Our Contribution


Cores/memory bandwidth ratio
PCI Express 3.0 Host Interface
GigaThread Engine
GPC
TPC
SM
Memory Controller
HBM2
L2 Cache
High-Speed Hub
NVLink
Lockstep (SIMT) execution

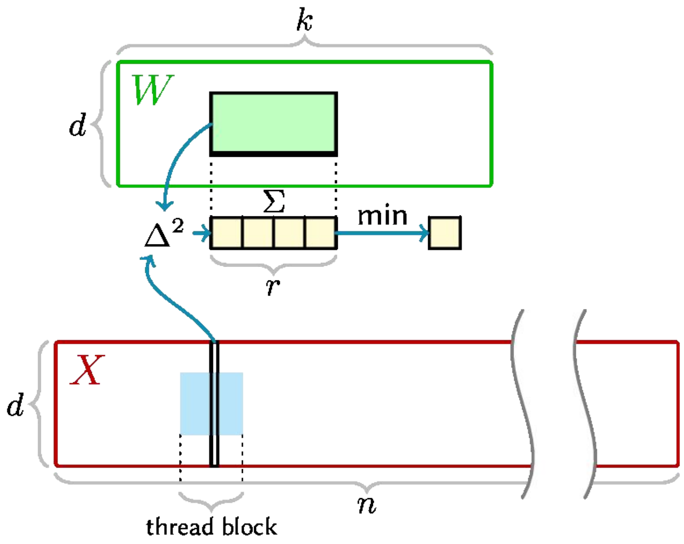
Assignment Step - Clever Caching



Assignment Step - Clever Caching

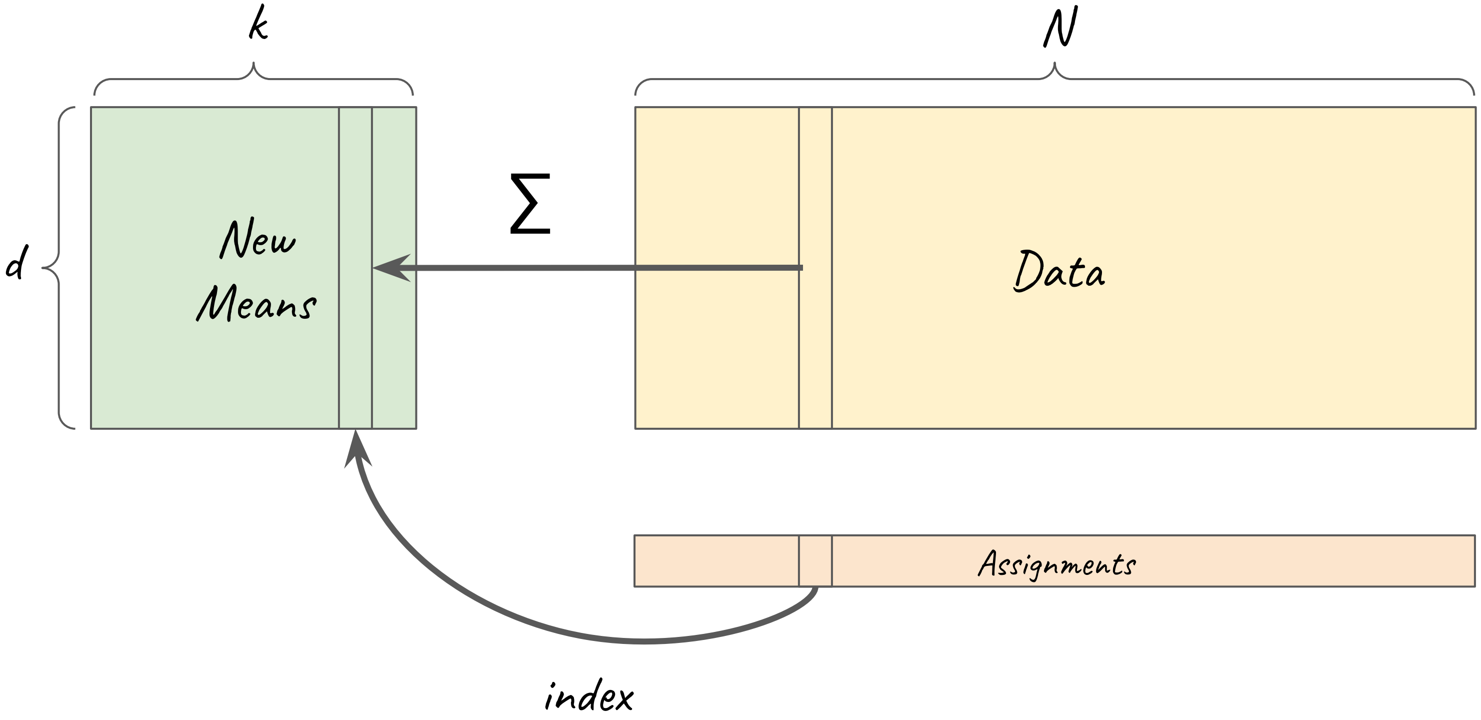


Regs caching strategy



Fixed caching strategy

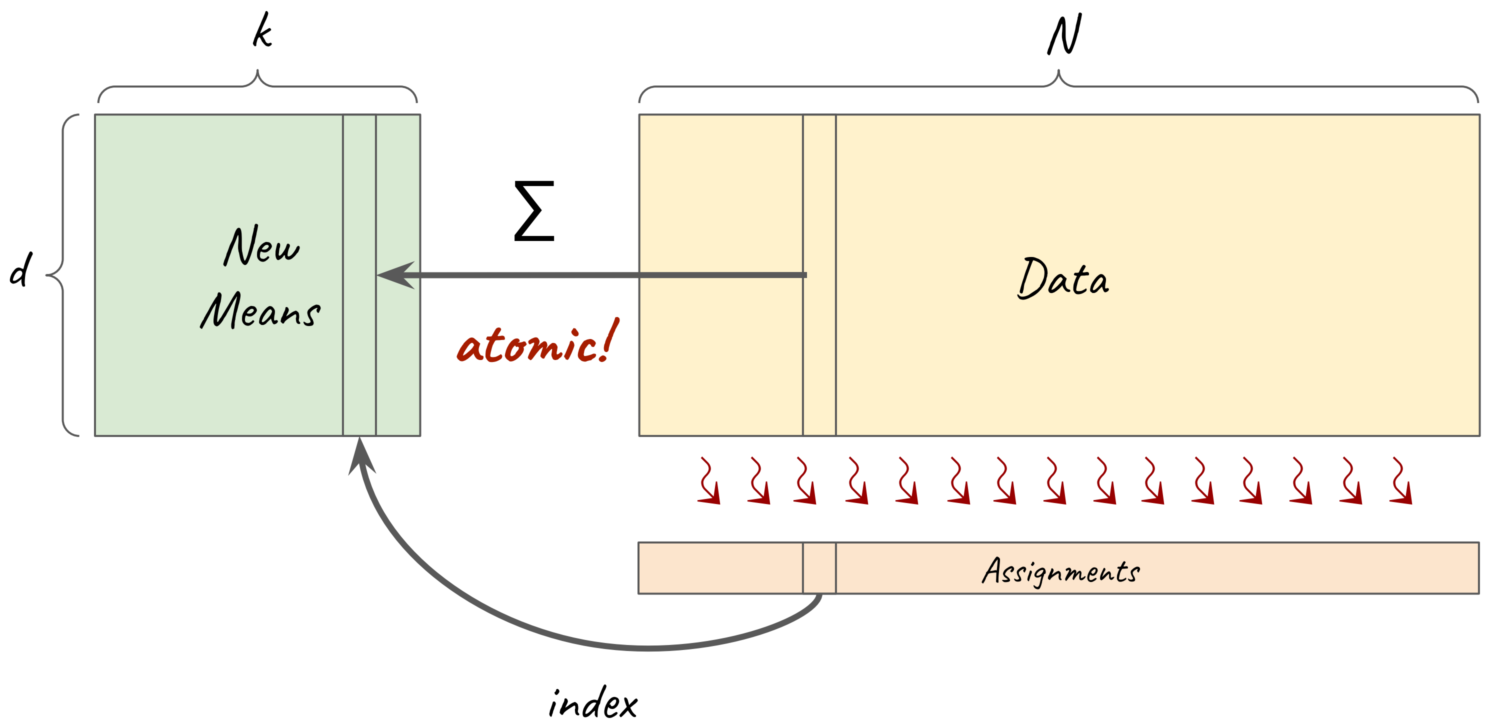
Update Step



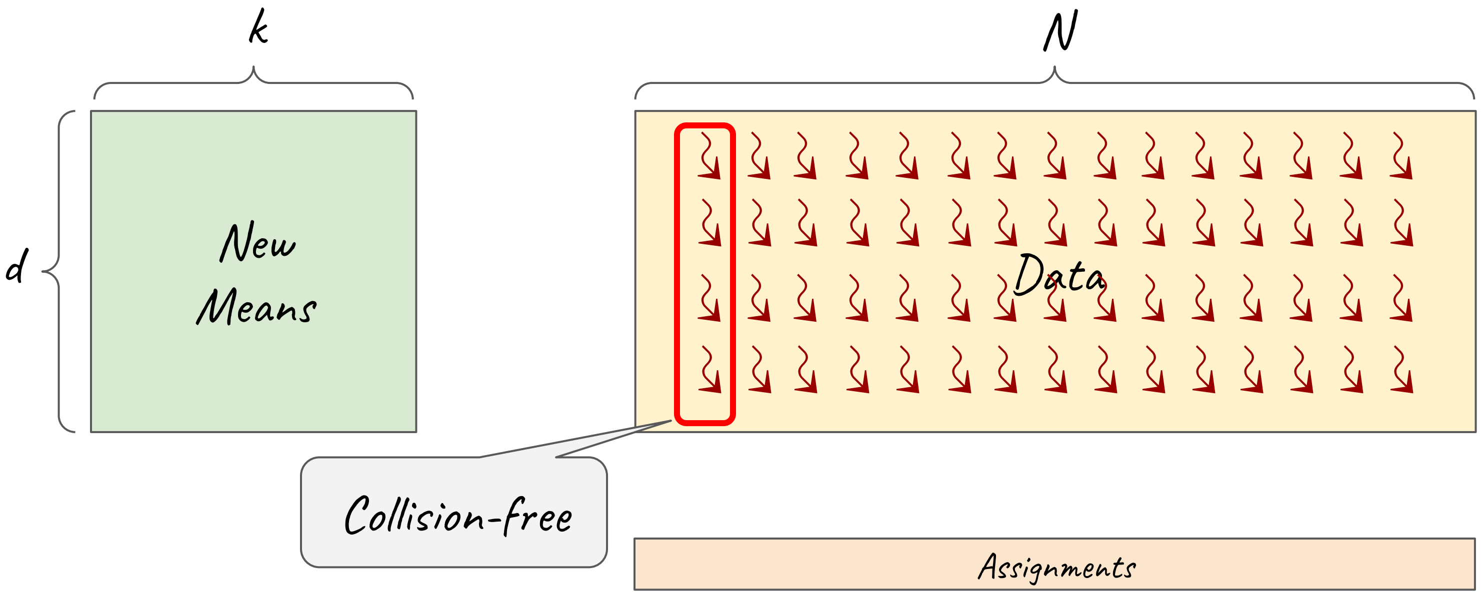
ICPP 2020

Update Step - Thread Allocation

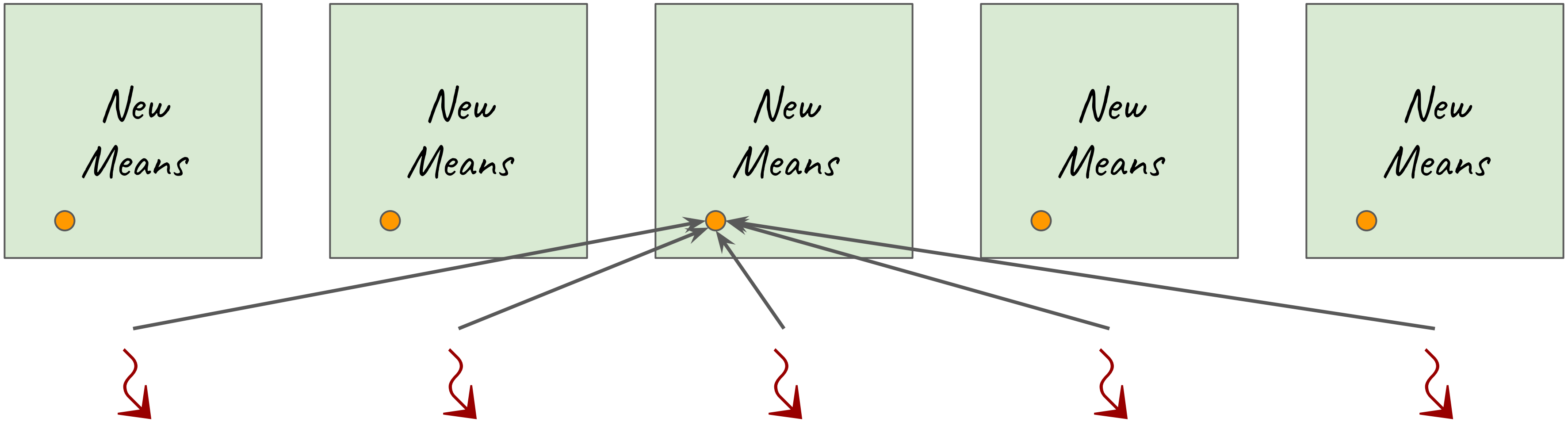
Update Step - Thread Allocation



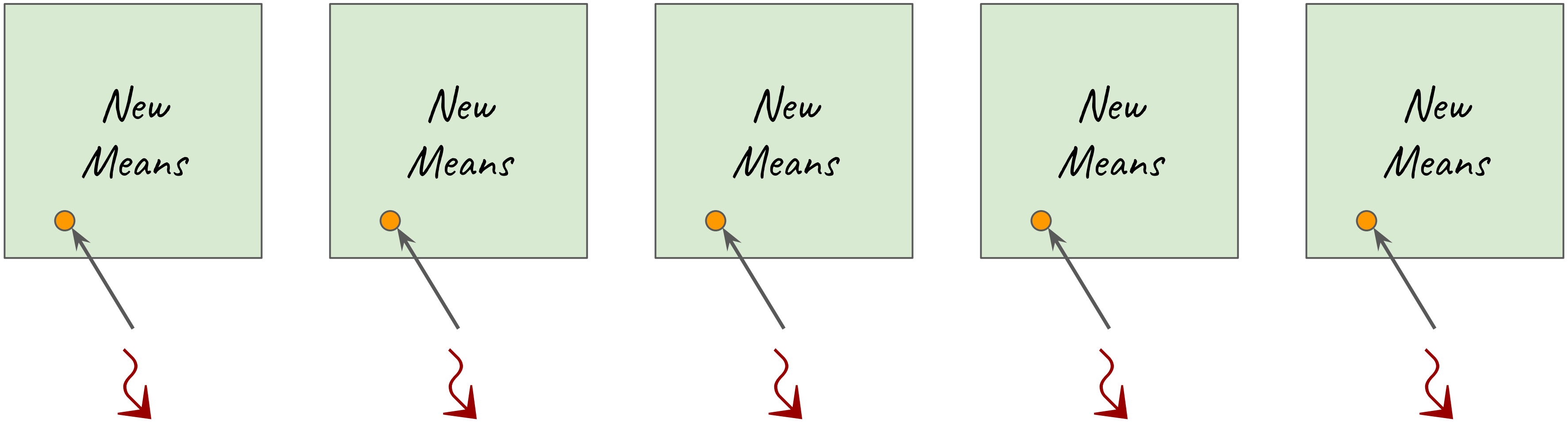
Update Step - Thread Allocation



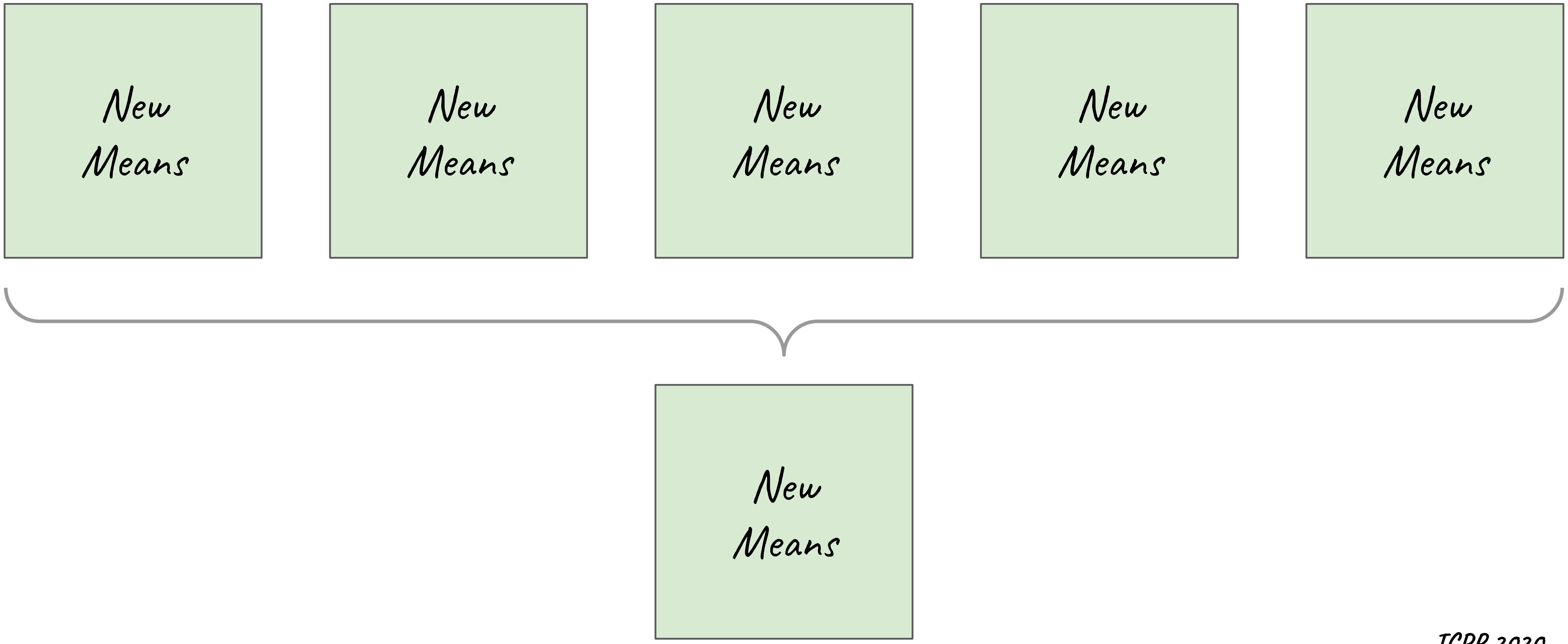
Increasing Atomic Throughput: Privatization



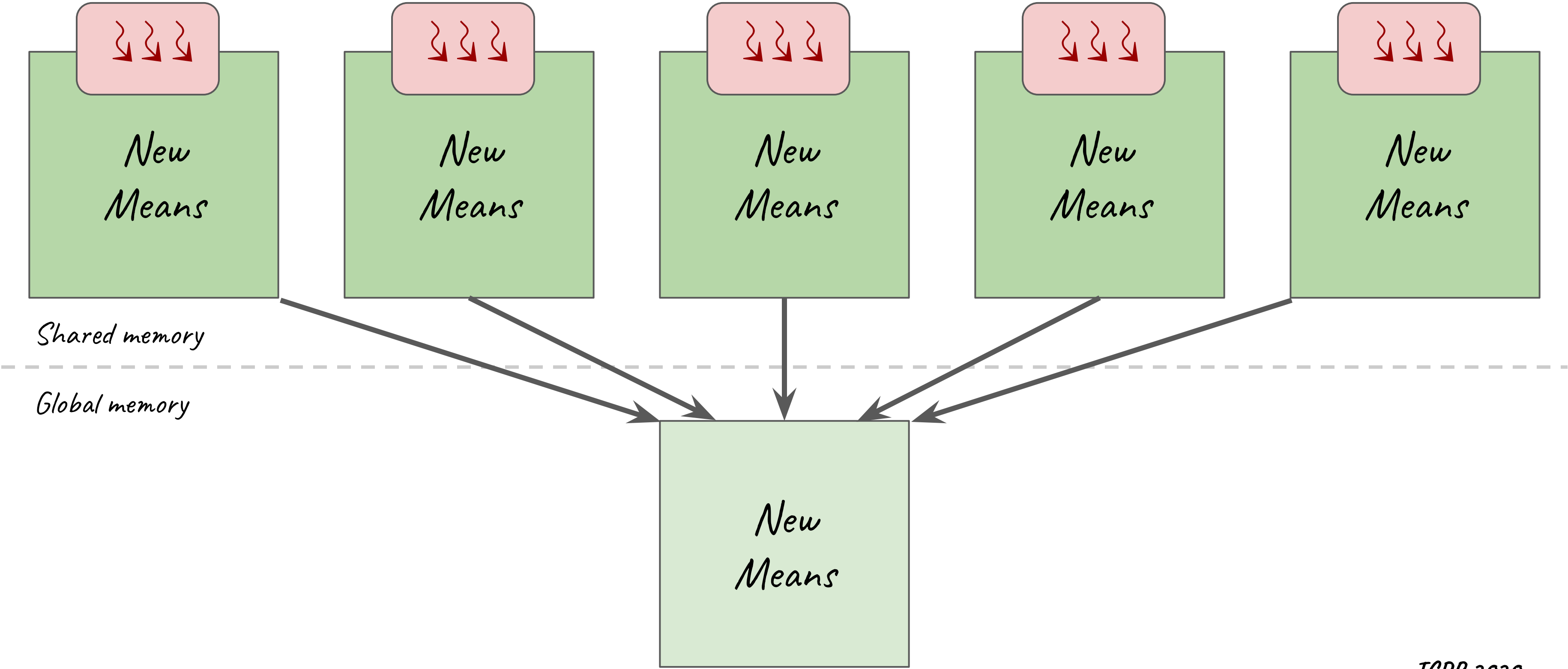
Increasing Atomic Throughput: Privatization



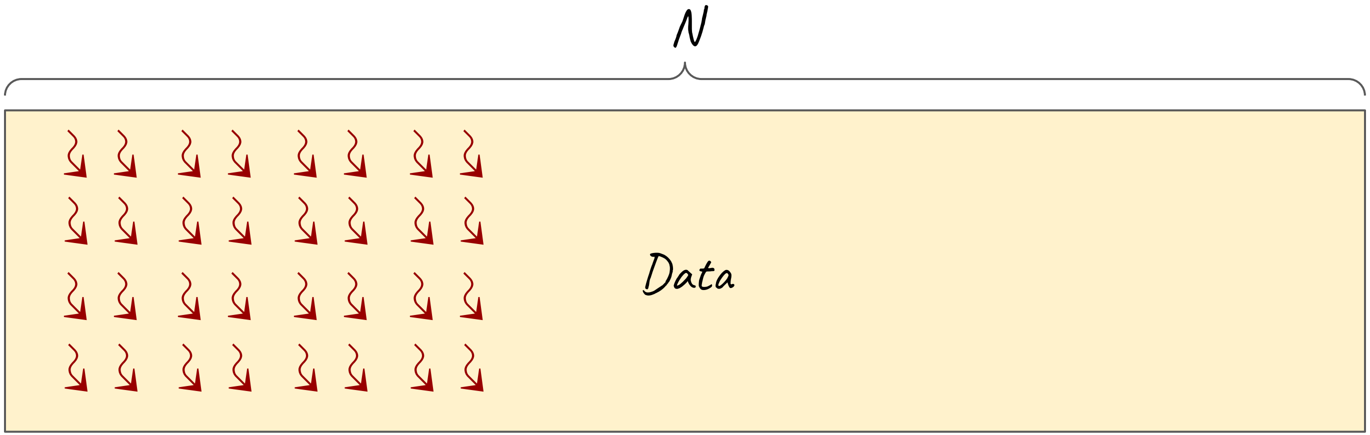
Increasing Atomic Throughput: Privatization



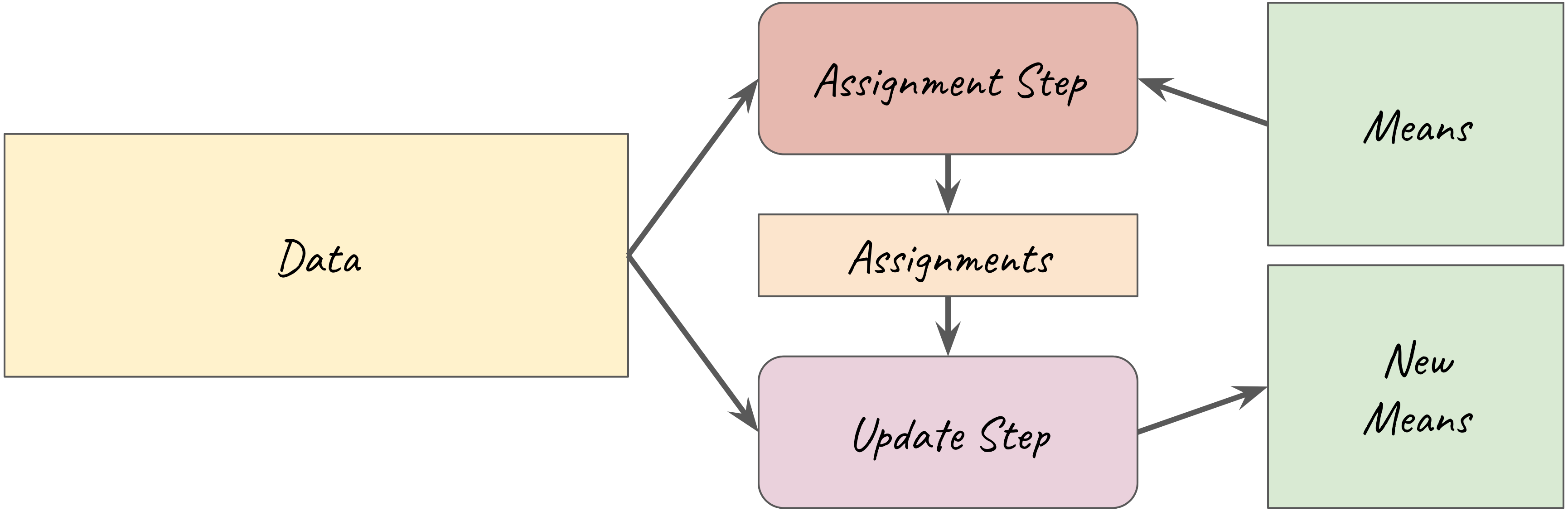
Privatization in Shared Memory



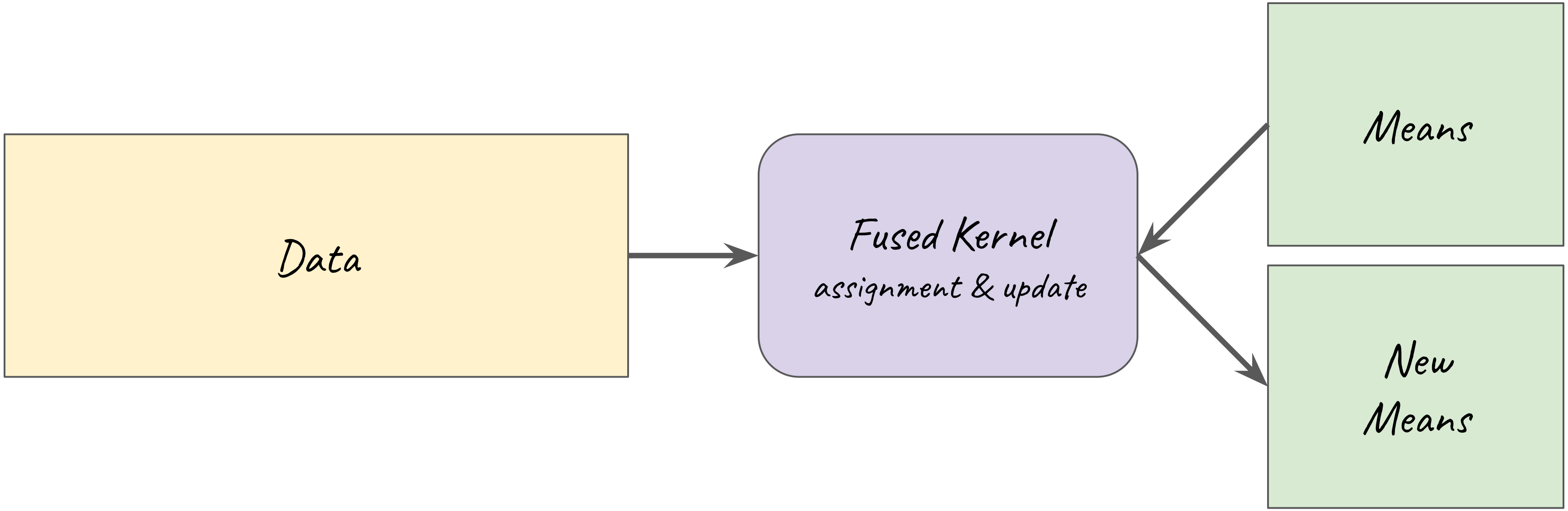
Privatization in Shared Memory



Complete Solution



Complete Solution - Fused Kernels

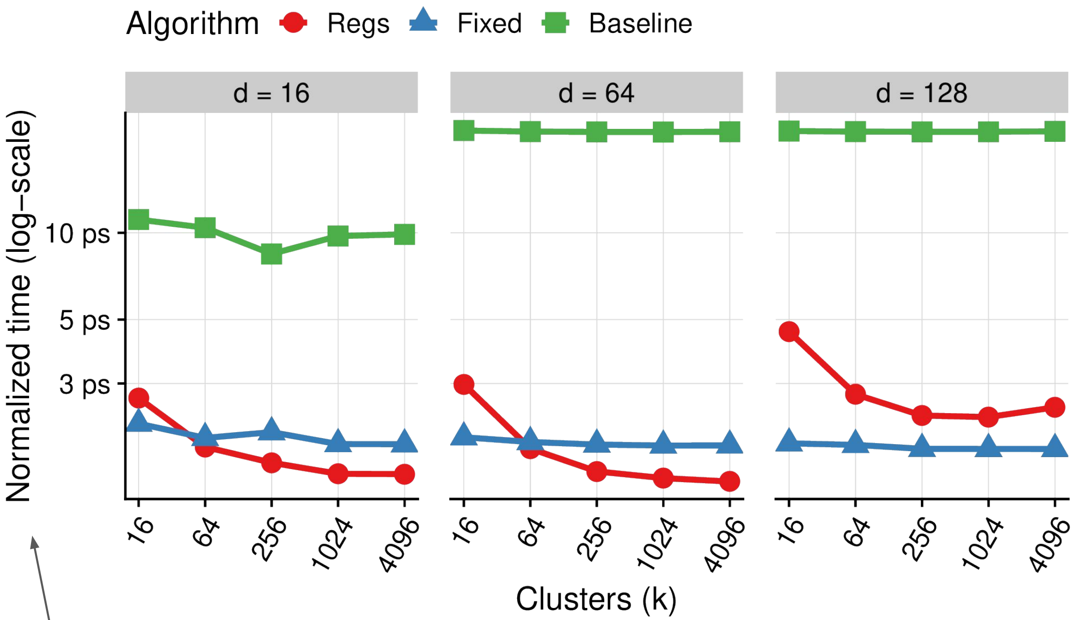


Technical Details

- Best data layout
- Atomic addition implementation
- Update step actually comprise two kernels - sum and division
- Code templating and loop unrolling
- Thread block shape and size
- ...

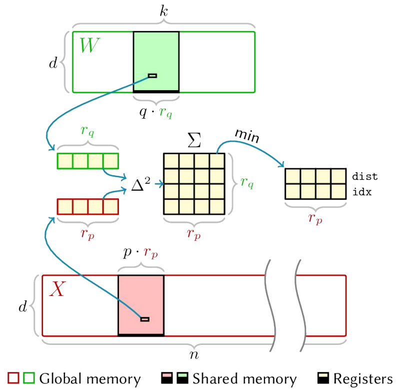
Experimental Results

Assignment step

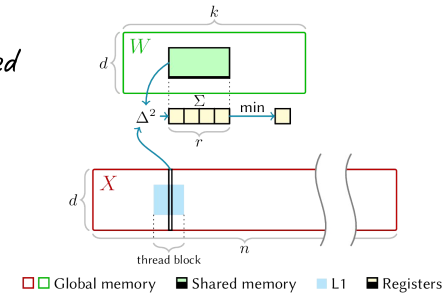


Actual time divided by N*D*k

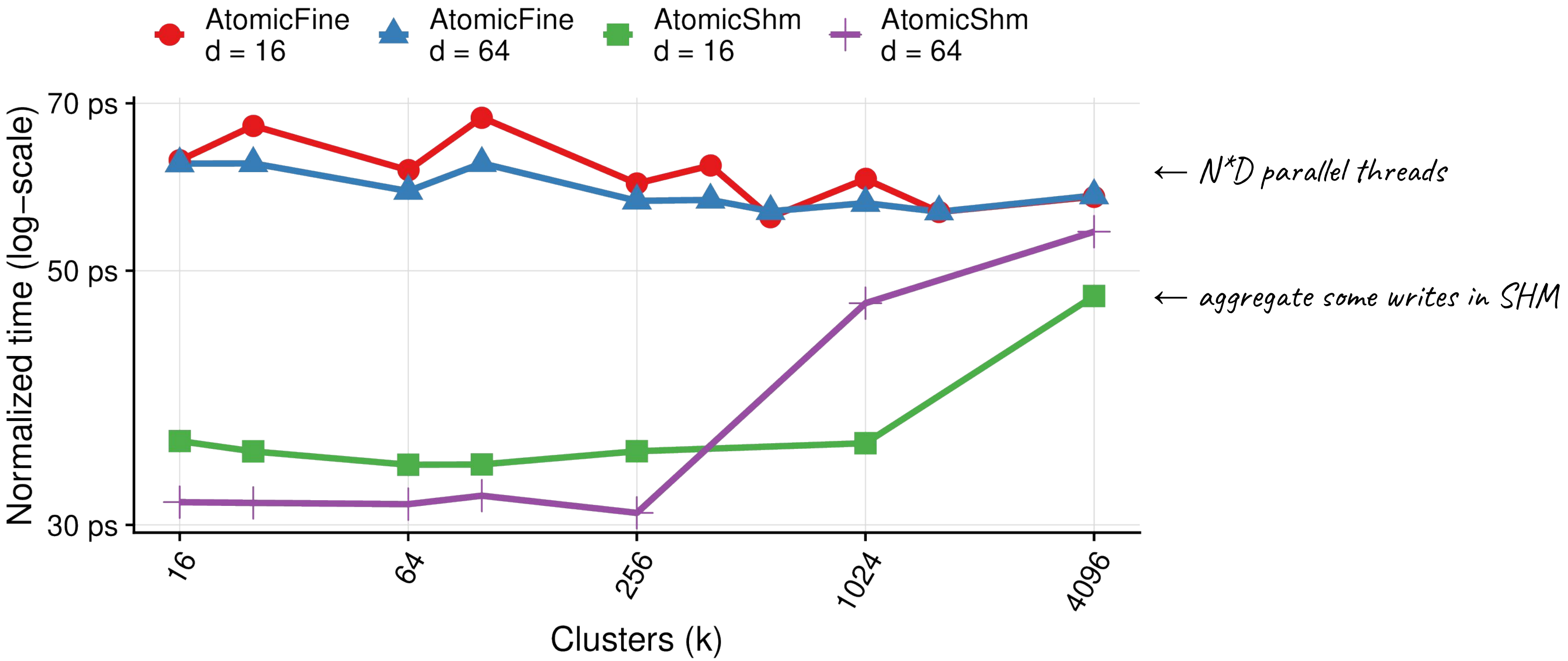
Regs



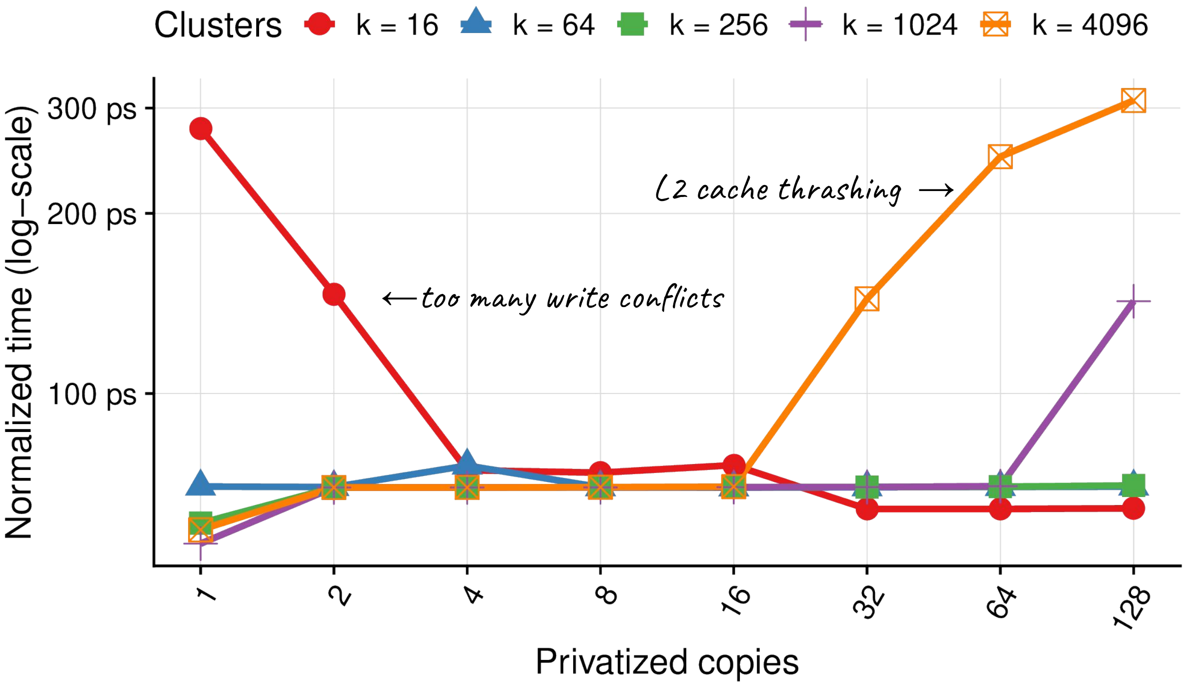
Fixed



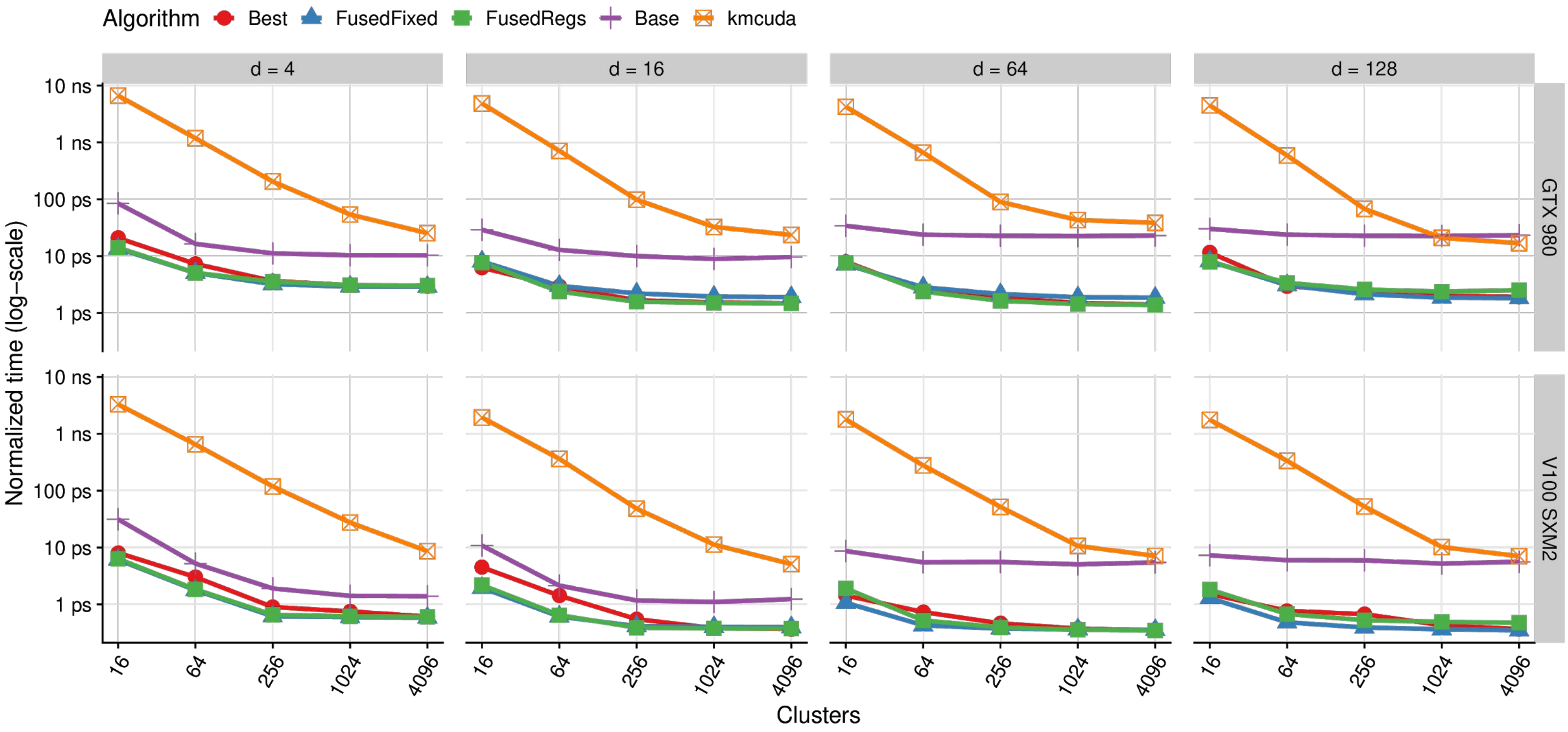
Update step - avoiding write conflicts

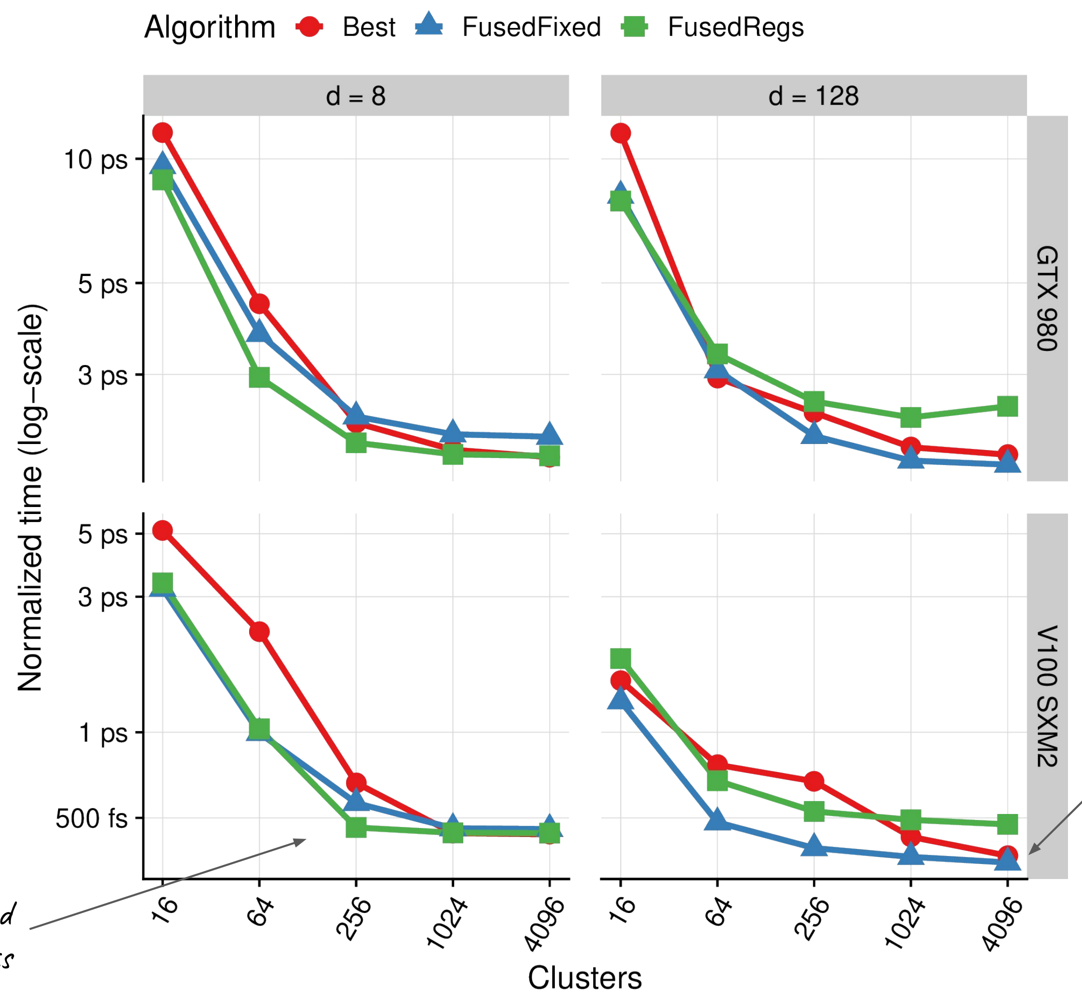


Update step - write conflicts of N*D threads



Combined results





1000 iterations per second
on 1 million points

5 iterations per second
on 1 million points

ICPP 2020

https://github.com/krulis-martin/cuda-kmeans

Thank you for watching!

Martin Kruliš, Ph.D.

Department of software engineering, Charles University
Prague, Czech Republic

krulis@ksi.mff.cuni.cz
http://www.ksi.mff.cuni.cz/~krulis
ORCID: 0000-0002-0985-8949

Miroslav Kratochvíl, M.Sc.

Department of software engineering, Charles University
Prague, Czech Republic

kratochvil@ksi.mff.cuni.cz
https://www.ksi.mff.cuni.cz/~kratochvil
ORCID: 0000-0001-7356-4075

matfyz