HykSort: A New Variant of Hypercube Quicksort on Distributed Memory Architectures

Hari Sundar
Dhairya Malhotra, George Biros

Institute for Computational Engineering & Sciences

THE UNIVERSITY OF TEXAS AT AUSTIN

International Conference on Supercomputing 2013

Distributed Sorting

problem statement

THE UNIVERSITY OF TEXAS AT AUSTIN

Input

- N unsorted keys, distributed evenly across p tasks
- unknown distribution

Distributed Sorting

problem statement

THE UNIVERSITY OF TEXAS AT AUSTIN

Input

- N unsorted keys, distributed evenly across p tasks
- unknown distribution

Output

- sort according to some specified order
- all keys on task $i \leq$ keys on task $i + 1$

Distributed Sorting

problem statement

THE UNIVERSITY OF TEXAS AT AUSTIN

Input

- N unsorted keys, distributed evenly across p tasks
- unknown distribution

Output

- sort according to some specified order
- all keys on task $i \leq$ keys on task $i + 1$

Challenge

- ensure load balance
- scalable to hundreds of thousands of cores

Distributed Sorting

problem statement

THE UNIVERSITY OF TEXAS AT AUSTIN

Input

- N unsorted keys, distributed evenly across p tasks
- unknown distribution

Output

- sort according to some specified order
- all keys on task $i \leq$ keys on task $i + 1$

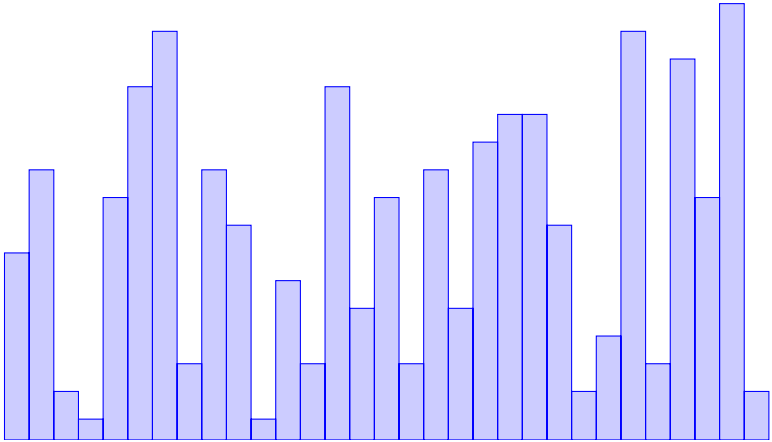
Challenge

- ensure load balance
- scalable to hundreds of thousands of cores

8 trillion keys in 37 secs $\rightarrow$ 0.9TB/s

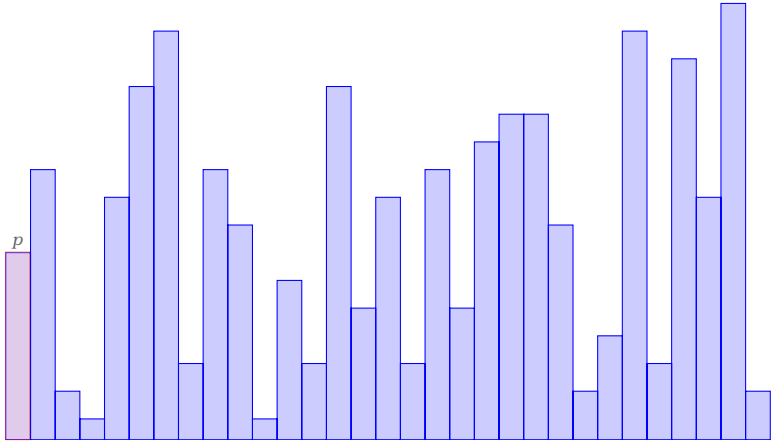
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



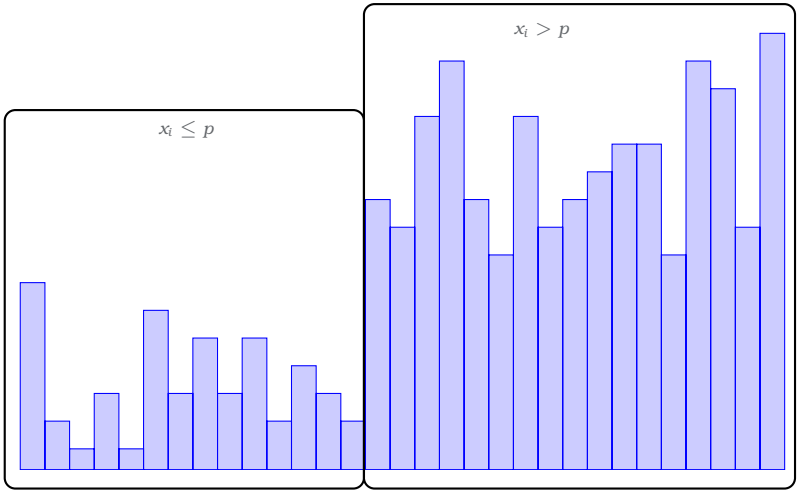
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



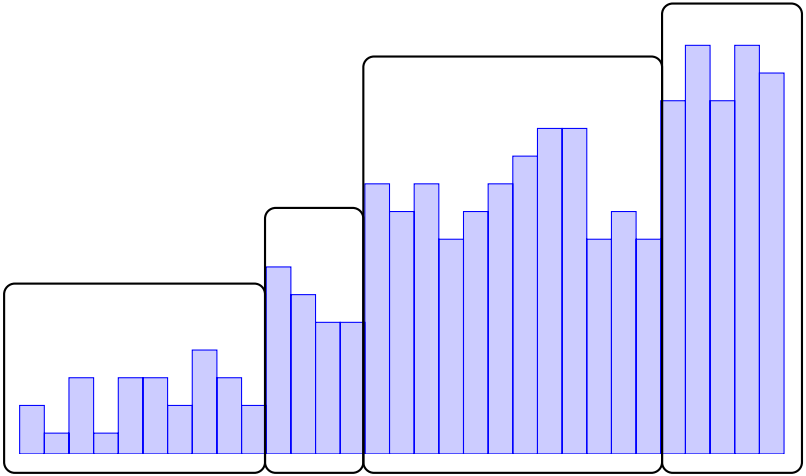
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



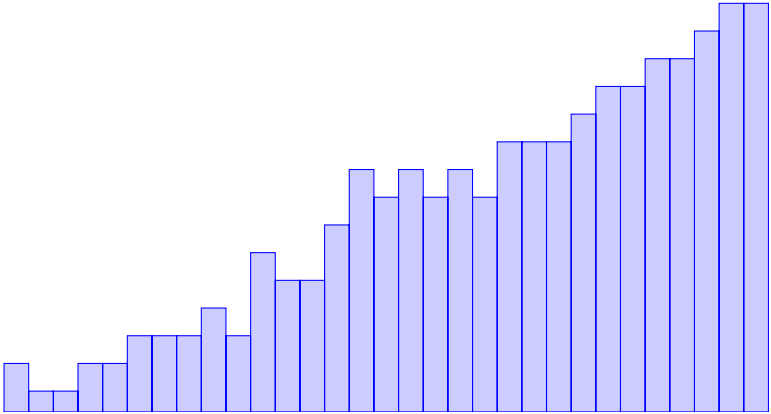
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



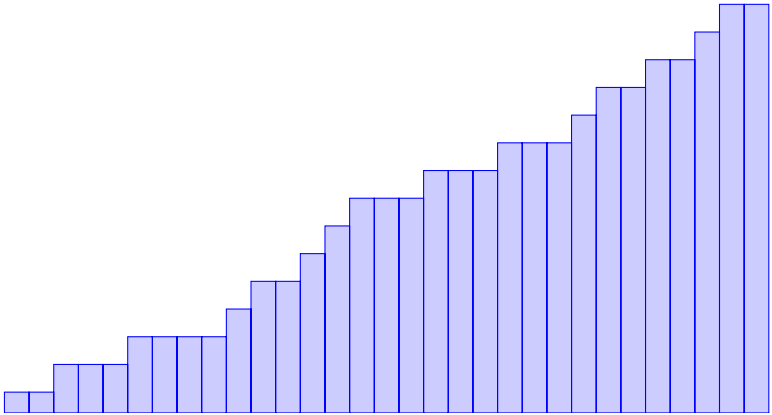
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



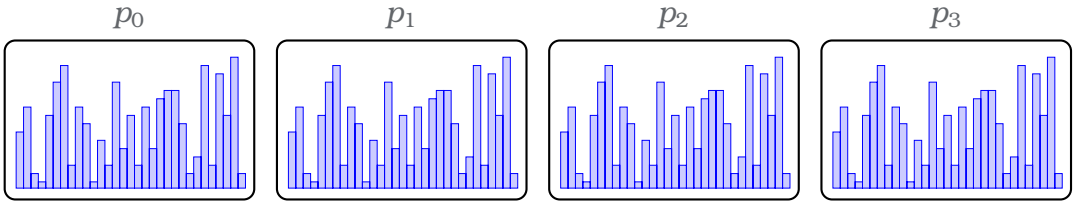
Sequential Sorting: Quicksort

$\mathcal{O}(n \log n)$



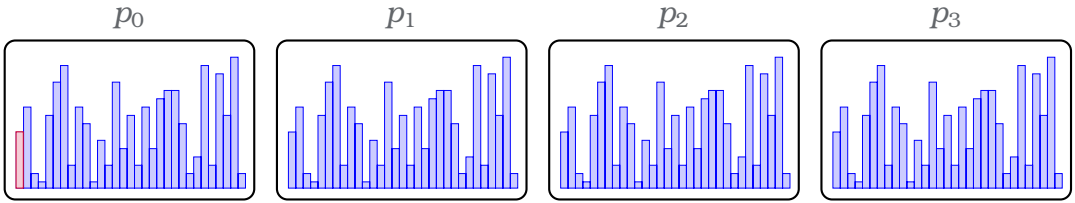
Parallel Sorting

HyperQuickSort (Wagar '87)



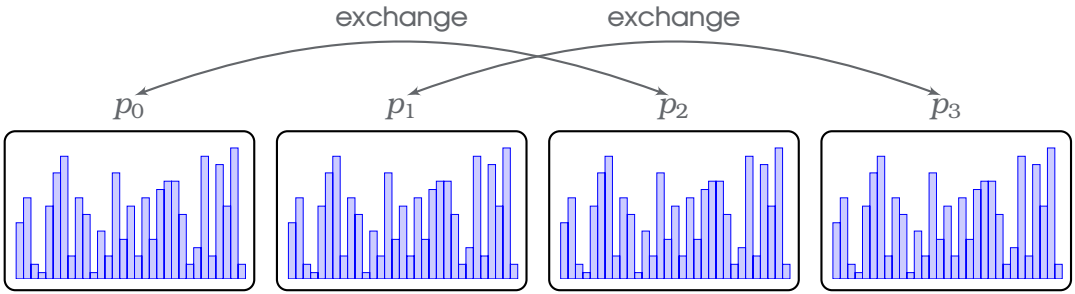
Parallel Sorting

HyperQuickSort (Wagar '87)



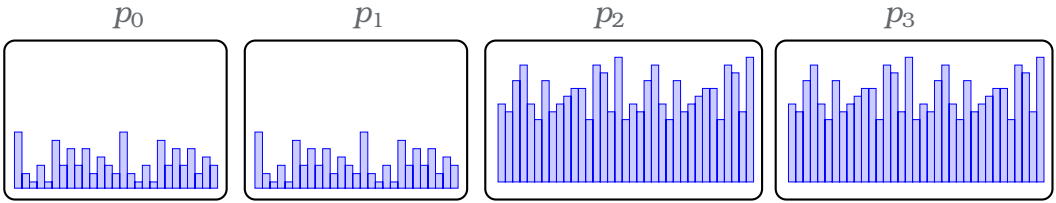
Parallel Sorting

HyperQuickSort (Wagar '87)



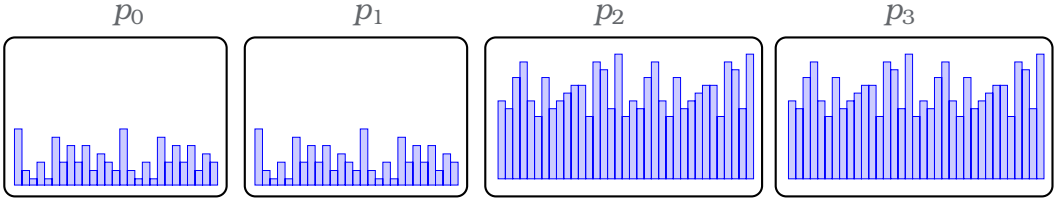
Parallel Sorting

HyperQuickSort (Wagar '87)



Parallel Sorting

HyperQuickSort (Wagar '87)



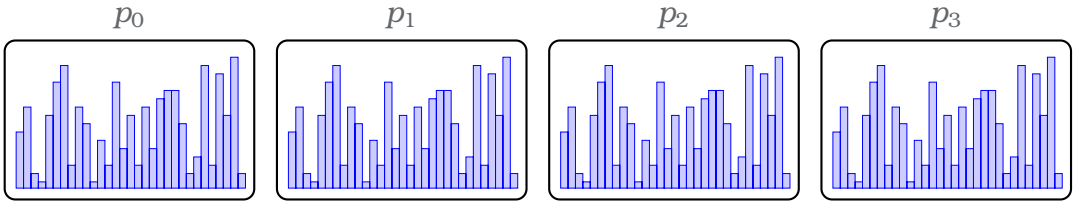
load (im)balance issues $\rightarrow$ pick global median

$\log p$ stages

data movement - $\mathcal{O}(N \log p)$

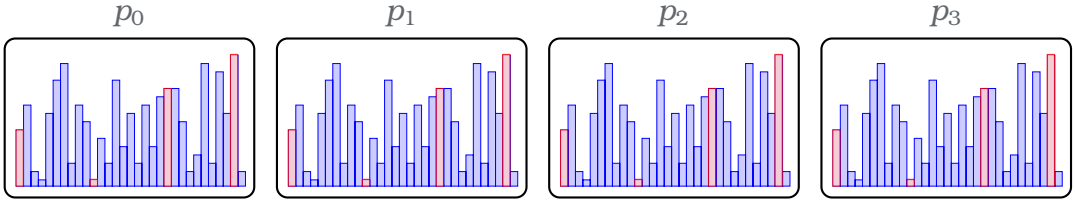
Parallel Sorting

SampleSort (Huang '83)



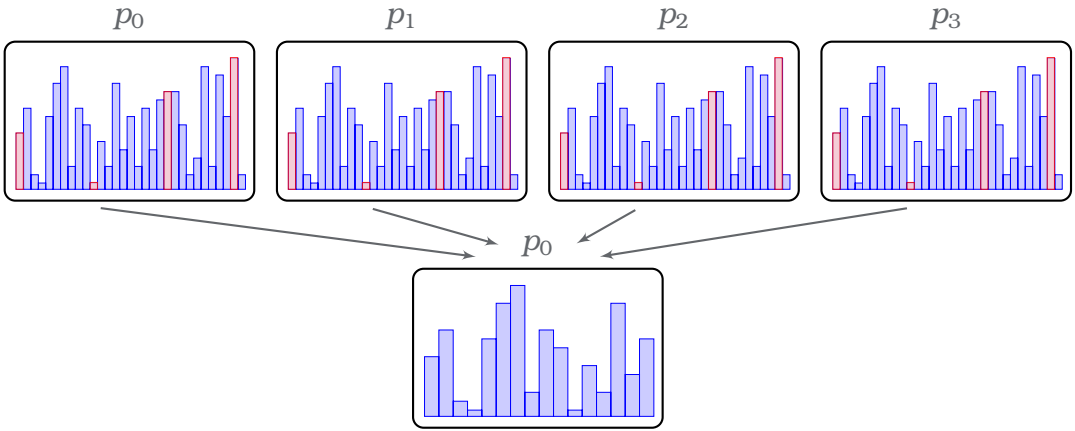
Parallel Sorting

SampleSort (Huang '83)



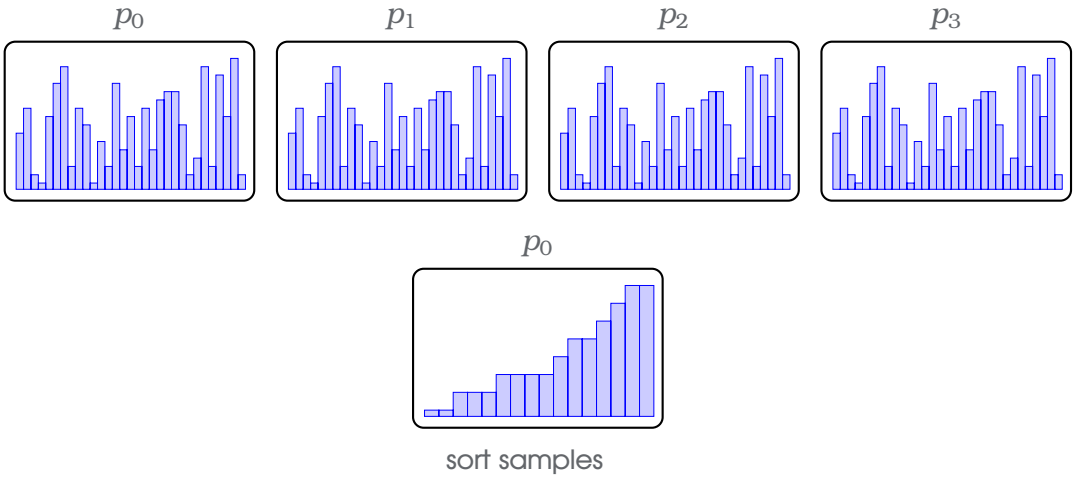
Parallel Sorting

SampleSort (Huang '83)



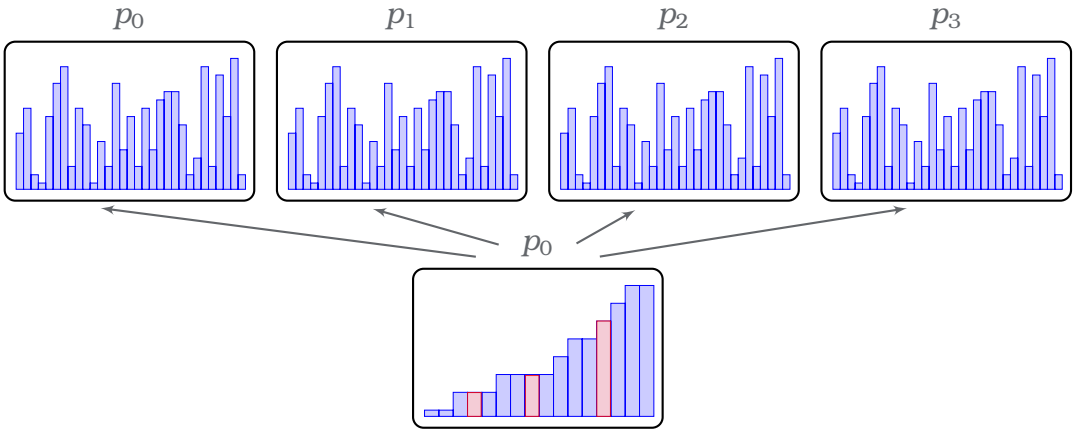
Parallel Sorting

SampleSort (Huang '83)



Parallel Sorting

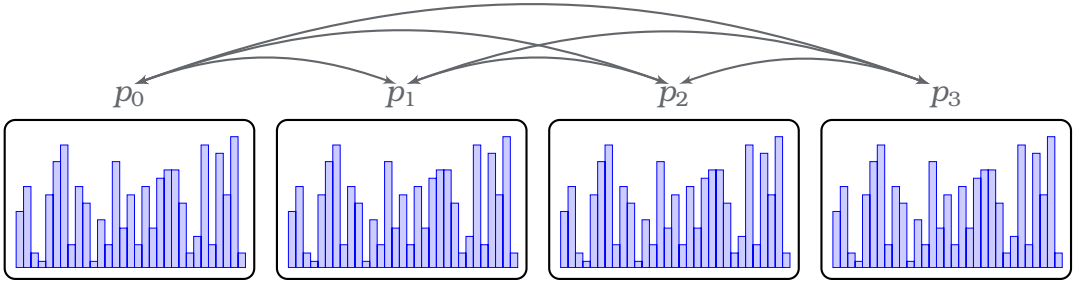
SampleSort (Huang '83)



pick splitters & broadcast

Parallel Sorting

SampleSort (Huang '83)



bucket data & all2all exchange

Parallel Sorting

SampleSort

THE UNIVERSITY OF TEXAS AT AUSTIN

- sort locally, sample evenly, $p - 1$ samples per task
- load balance $\rightarrow 2N/p$
- $\mathcal{O}(p^2)$ samples are a bottleneck for large $p \rightarrow$ BitonicSort
- all2all communication - N keys in p^2 messages

Parallel Sorting

SampleSort

THE UNIVERSITY OF TEXAS AT AUSTIN

- sort locally, sample evenly, $p - 1$ samples per task
- load balance $\rightarrow 2N/p$
- $\mathcal{O}(p^2)$ samples are a bottleneck for large $p \rightarrow$ BitonicSort
- all2all communication - N keys in p^2 messages

Parallel Sorting

Splitter Selection

THE UNIVERSITY OF TEXAS AT AUSTIN

Iteratively choose splitters in parallel - $\mathcal{O}(p)$

- choose guess & broadcast
- compute histogram & reduce
- iterate

HistogramSort (Kale '93, '10)
psort/CloudRAMSort (Chhugani '08, '12)

Parallel Sorting

Splitter Selection

THE UNIVERSITY OF TEXAS AT AUSTIN

Iteratively choose splitters in parallel - $\mathcal{O}(p)$

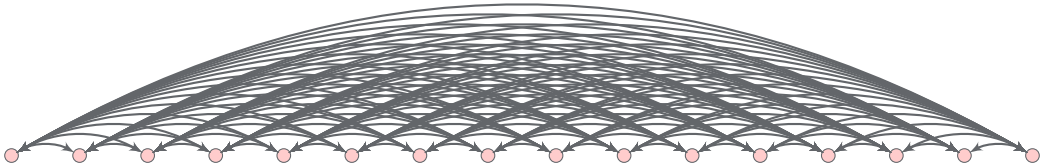
- choose guess & broadcast
- compute histogram & reduce
- iterate

HistogramSort (Kale '93, '10) $\rightarrow$ 1TB in 4.3 secs $\rightarrow$ 0.24TB/s

psort/CloudRAMSort (Chhugani '08, '12) $\rightarrow$ 1TB in 4.6 secs $\rightarrow$ 0.21TB/s

Parallel Sorting

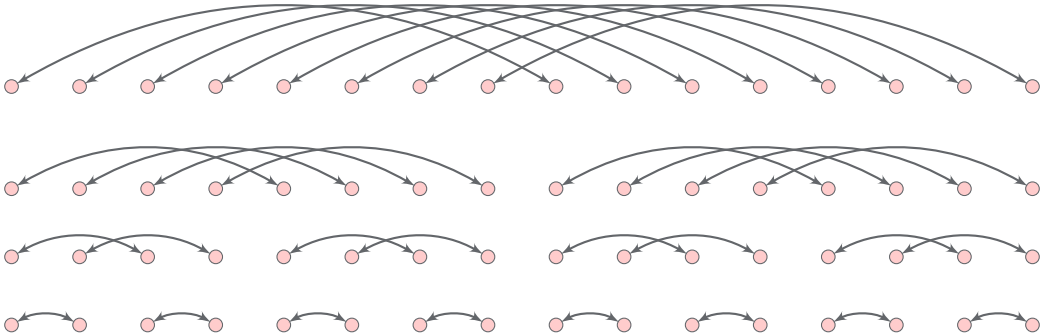
Communication Patterns



SampleSort

Parallel Sorting

Communication Patterns



HyperQuickSort - $\log p$ stages

Parallel Sorting

k-way HyperQuickSort

THE UNIVERSITY OF TEXAS AT AUSTIN

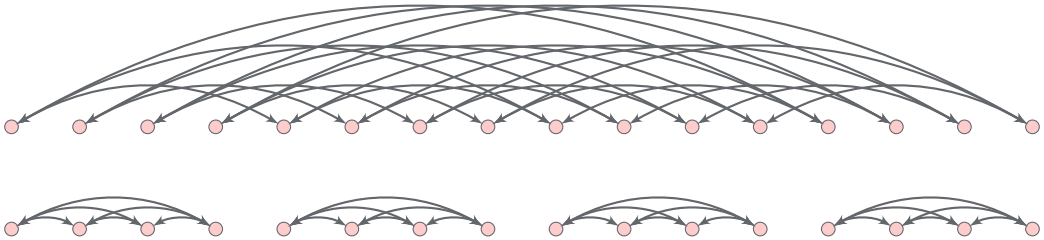
Generalize HyperQuickSort to perform k-way splitting

Parallel Sorting

k-way HyperQuickSort

THE UNIVERSITY OF TEXAS AT AUSTIN

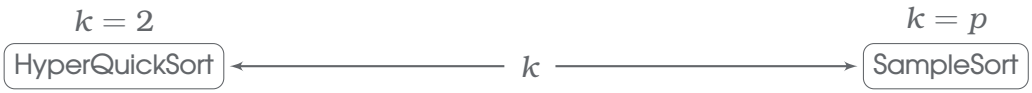
Generalize HyperQuickSort to perform k-way splitting



HykSort - $\log p / \log k$ stages

Parallel Sorting

HykSort & k-way SampleSort



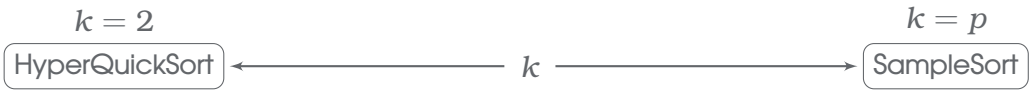
Same idea can be used to implement a k-way `MPI_Alltoallv`

k-way SampleSort scales better than standard SampleSort

Is the performance the same as HykSort ?

Parallel Sorting

HykSort & k-way SampleSort



Same idea can be used to implement a k-way `MPI_Alltoallv`

k-way SampleSort scales better than standard SampleSort

Is the performance the same as HykSort ?

HykSort estimates $\mathcal{O}(k)$ samples as opposed to $\mathcal{O}(p)$ samples for SampleSort

Parallel Sorting

Other Features

THE UNIVERSITY OF TEXAS AT AUSTIN

- Implemented using only non-blocking point-to-point exchanges
- Overlap computation with communication
- `C++, MPI, OpenMP`
- Vectorized multithreaded MergeSort $\rightarrow$ local sort
- available from `http://padas.ices.utexas.edu/`

Results

Evaluation Setup

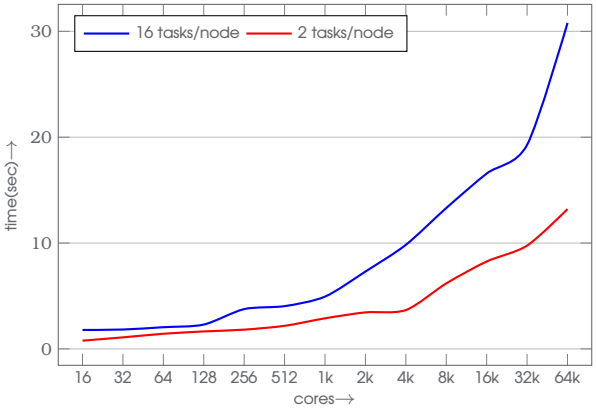
- *Titan*, Cray XK7 supercomputer at Oak Ridge National Laboratory (ORNL)
- $18,688$ nodes $\times$ 16 cores, 32GB RAM
- $299,008$ total cores

Results

MPI vs Hybrid Code

key type	tasks : threads 16:1	2:8	1:16
int 128mb	0.51	0.61	0.63
int 2GB	9.09	10.65	10.96
treenode 128mb	0.39	0.76	1.13
treenode 2GB	6.4	12.78	18.61
100-byte 128mb	0.33	0.62	1.15
100-byte 2GB	5.54	10.52	15.79

Single-node performance on Titan.

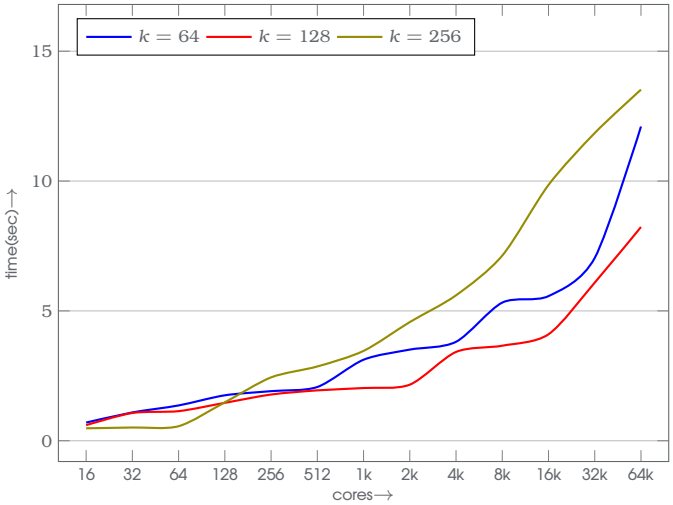


128−way `all2all`

512M `int` keys per node

Results

Choosing k



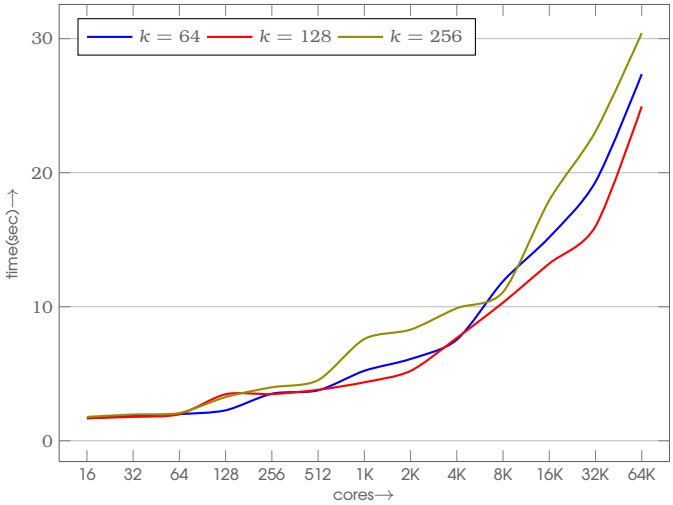
HykSort

`exchange data` stage

`2GB` `treenode` keys per node

Results

Choosing k



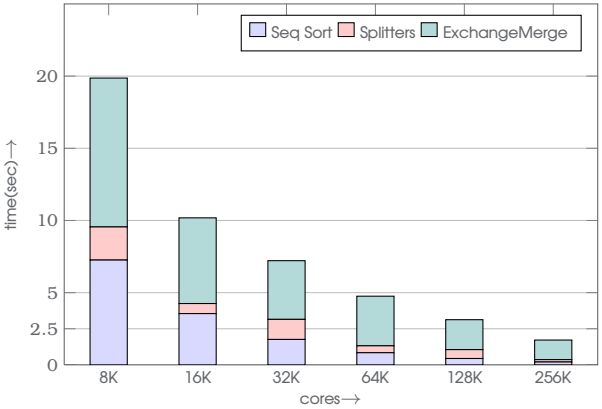
SAMPLESORT

`all2all` stage

`2GB int` keys per node

Results: Scalability

strong scalability



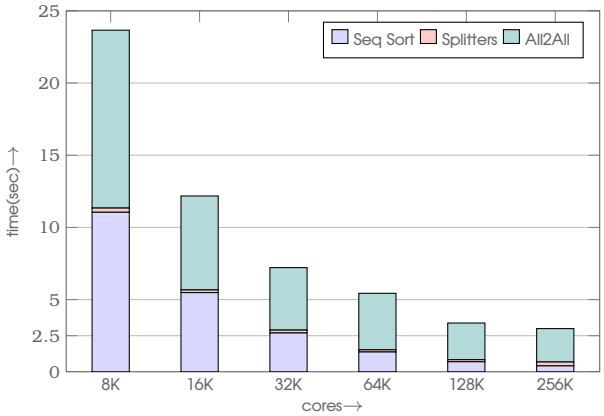
128-way HykSort

1TB integer keys

12x speedup

Results: Scalability

strong scalability



128-way SAMPLESORT

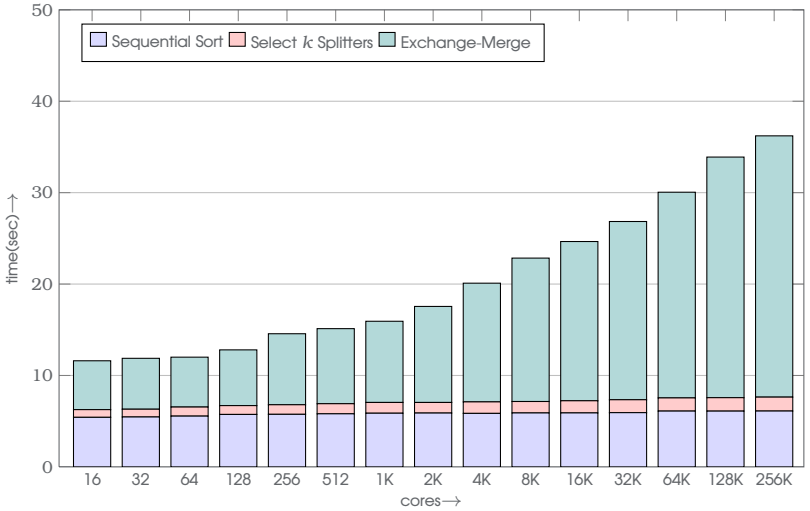
1TB integer keys

8x speedup

Results: Scalability

THE UNIVERSITY OF TEXAS AT AUSTIN

weak scalability

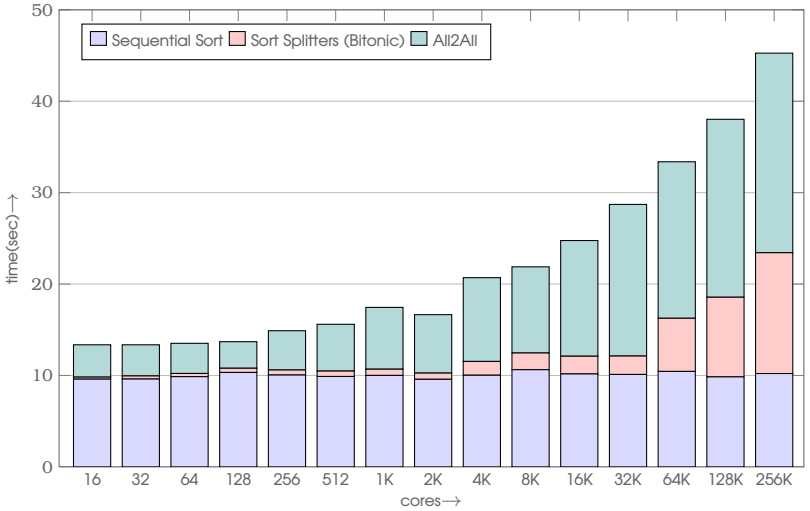


128-way HykSort
2GB → 32TB (16,384x)
128MB per core -- INT
throughput: 0.9TB/s

Results: Scalability

THE UNIVERSITY OF TEXAS AT AUSTIN

weak scalability

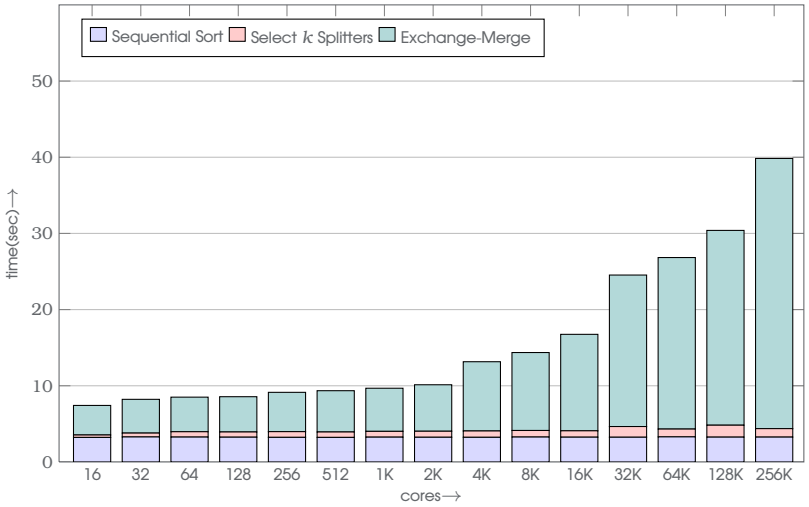


128-way SAMPLESORT
2GB → 32TB (16,384x)
128MB per core -- INT
throughput: 0.75TB/s

Results: Scalability

THE UNIVERSITY OF TEXAS AT AUSTIN

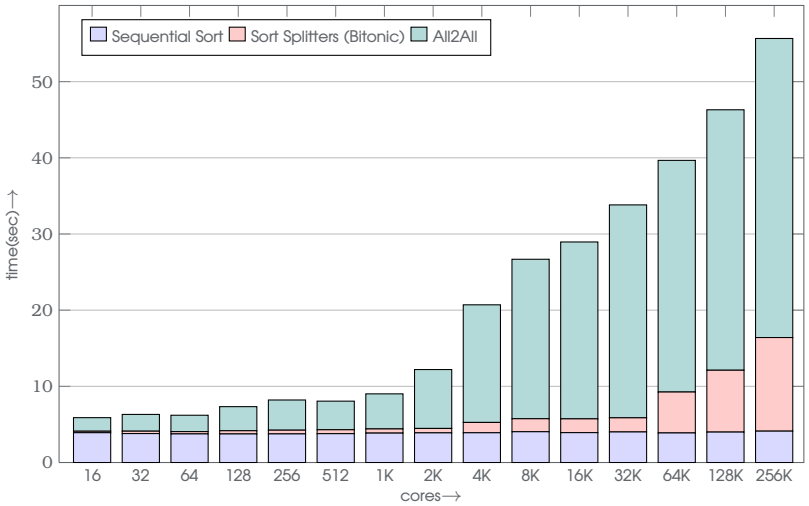
weak scalability



128-way HykSort
2GB → 32TB (16,384x)
128MB per core --
100-BYTE
throughput: 0.85TB/s

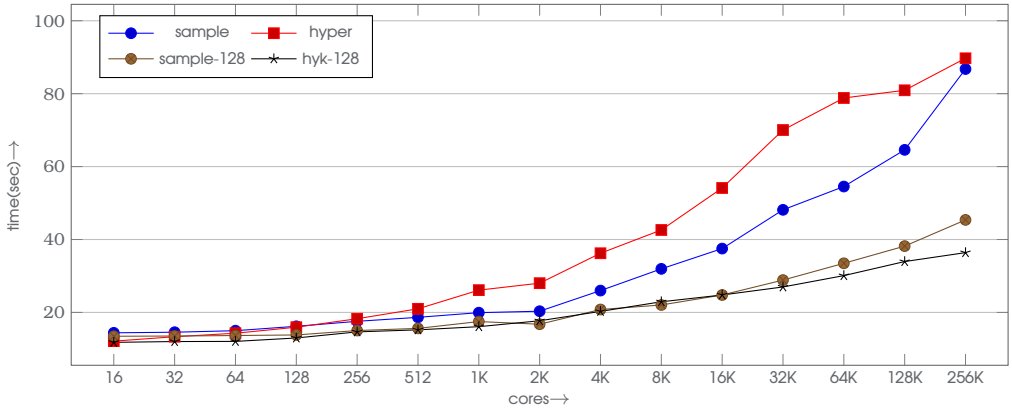
Results: Scalability

weak scalability



128-way SAMPLESORT
2GB → 32TB (16,384x)
128MB per core --
100-BYTE
throughput: 0.63TB/s

Conclusions



- distributed sort that scales to hundreds of thousands of cores
- staged all2all to avoid network congestion
- efficient implementation using non-blocking communication
- effective sort throughput of 0.9TB/s on *Titan*

Thank you

THE UNIVERSITY OF TEXAS AT AUSTIN

Questions ?