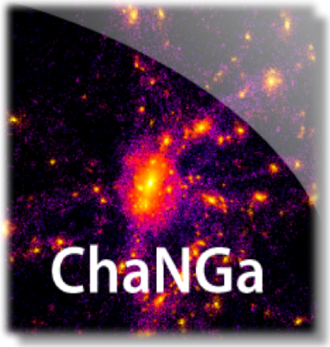
Histogram sort with Sampling (HSS)

Vipul Harsh, Laxmikant Kale

Parallel sorting in the age of Exascale

- Charm N-body GrAvity solver
- Massive Cosmological N-body simulations
- Parallel sorting in every iteration

Parallel sorting in the age of Exascale



- Charm N-body GrAvity solver
- Massive Cosmological N-body simulations
- Parallel sorting in every iteration

- Cosmology code based on Chombo
- Global sorting every step for load balance/locality

CHARM

CHOMBO

UIUC

Parallel sorting: Goals

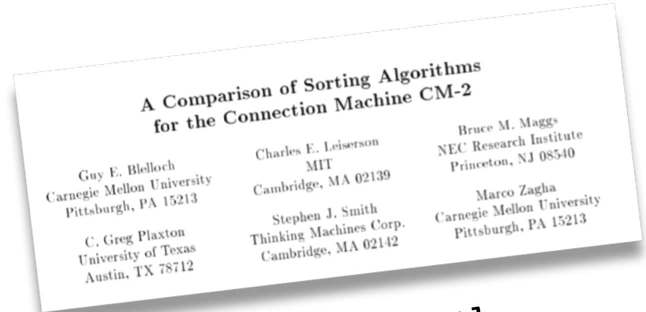
- Load balance across processors
- Optimal data movement
- Generality: robustness to input distributions, duplicates
- Scalability and performance

Parallel sorting: A basic template

- *p* processors, N/*p* keys in each processor
- Determine (*p*-1) splitter keys to partition keys into *p* buckets
- Send all keys to appropriate destination bucket processor
- Eg. Sample sort, Histogram sort

Existing algorithms: Parallel Sample sort

- Samples *s* keys from each processor
- Picks (p-1) splitters from p x s samples


A Comparison of Sorting Algorithms
for the Connection Machine CM-2

Guy E. Blelloch
Carnegie Mellon University
Pittsburgh, PA 15213

Charles E. Leiserson
MIT
Cambridge, MA 02139

Bruce M. Maggs
NEC Research Institute
Princeton, NJ 08540

C. Greg Plaxton
University of Texas
Austin, TX 78712

Stephen J. Smith
Thinking Machines Corp.
Cambridge, MA 02142

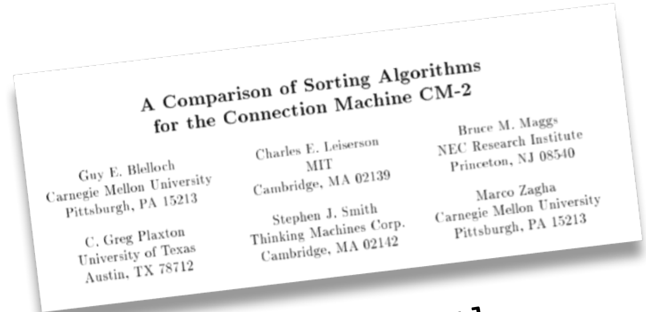
Marco Zagha
Carnegie Mellon University
Pittsburgh, PA 15213

[SPAA' 91]

Problem: Too many samples required for good load balance

Existing algorithms: Parallel Sample sort

- Samples *s* keys from each processor
- Picks (p-1) splitters from p x s samples



A Comparison of Sorting Algorithms
for the Connection Machine CM-2

Guy E. Blelloch
Carnegie Mellon University
Pittsburgh, PA 15213

Charles E. Leiserson
MIT
Cambridge, MA 02139

Bruce M. Maggs
NEC Research Institute
Princeton, NJ 08540

C. Greg Plaxton
University of Texas
Austin, TX 78712

Stephen J. Smith
Thinking Machines Corp.
Cambridge, MA 02142

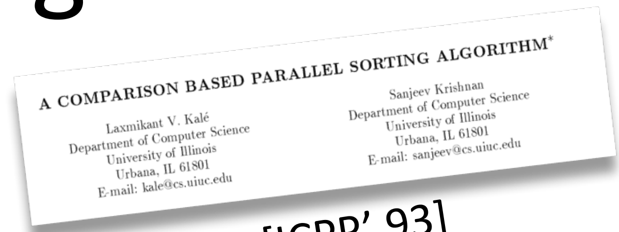
Marco Zagha
Carnegie Mellon University
Pittsburgh, PA 15213

[SPAA' 91]

Problem: Too many samples required for good load balance

64 bit keys, p = 100,000 & 5% max load imbalance, sample size ≈ 8 GB

Existing algorithms: Histogram sort


A COMPARISON BASED PARALLEL SORTING ALGORITHM*

Laxmikant V. Kalé
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: kale@cs.uiuc.edu

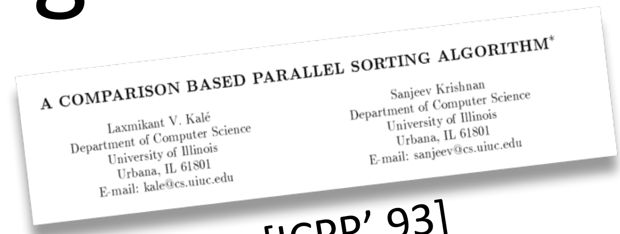
Sanjeev Krishnan
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: sanjeev@cs.uiuc.edu

[ICPP' 93]

- Pick *s x p* candidate keys
- Compute rank of each candidate key (histogram)
- Select splitters from the candidates

PPL
UIUC

Existing algorithms: Histogram sort


A COMPARISON BASED PARALLEL SORTING ALGORITHM*

Laxmikant V. Kalé
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: kale@cs.uiuc.edu

Sanjeev Krishnan
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: sanjeev@cs.uiuc.edu

[ICPP' 93]

- Pick *s x p* candidate keys

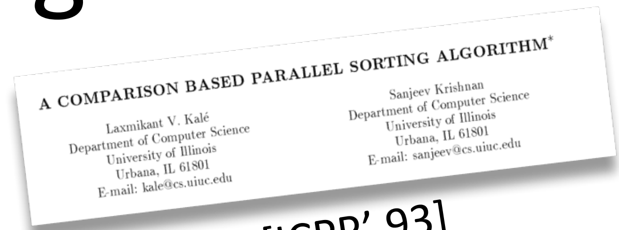
- Compute rank of each candidate key (histogram)

- Select splitters from the candidates

  OR

- Refine the candidates and repeat

Existing algorithms: Histogram sort


A COMPARISON BASED PARALLEL SORTING ALGORITHM*

Laxmikant V. Kalé
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: kale@cs.uiuc.edu

Sanjeev Krishnan
Department of Computer Science
University of Illinois
Urbana, IL 61801
E-mail: sanjeev@cs.uiuc.edu

[ICPP' 93]

- Pick *s x p* candidate keys
- Compute rank of each candidate key (histogram)
- Select splitters from the candidates

OR

- Refine the candidates and repeat

- Works quite well for large *p*
- But can take more iterations if input skewed

Histogram sort with sampling (HSS)

- An adaptation of Histogram sort
- Sample before each histogramming round
    - Sample intelligently
    - Use results from previous rounds
    - Discard wasteful samples at source

PPL
UIUC

Histogram sort with sampling (HSS)

- An adaptation of Histogram sort

- Sample before each histogramming round
    - Sample intelligently
    - Use results from previous rounds
    - Discard wasteful samples at source

- HSS has sound theoretical guarantees

Histogram sort with sampling (HSS)

- An adaptation of Histogram sort
- Sample before each histogramming round
  - Sample intelligently
  - Use results from previous rounds
  - Discard wasteful samples at source
- HSS has sound theoretical guarantees
- Independent of input distribution

PPL
UIUC

Histogram sort with sampling (HSS)

- An adaptation of Histogram sort

- Sample before each histogramming round
    - Sample intelligently
    - Use results from previous rounds
    - Discard wasteful samples at source

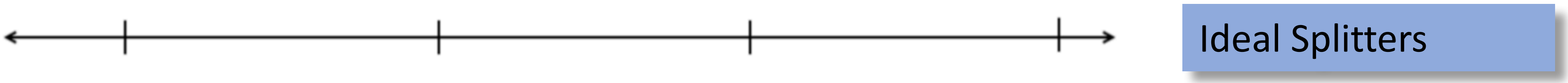
- HSS has sound theoretical guarantees
- Independent of input distribution
- Justifies why Histogram sort does well

HSS: Intelligent Sampling

Find (p-1) splitter keys to partition input into p ranges

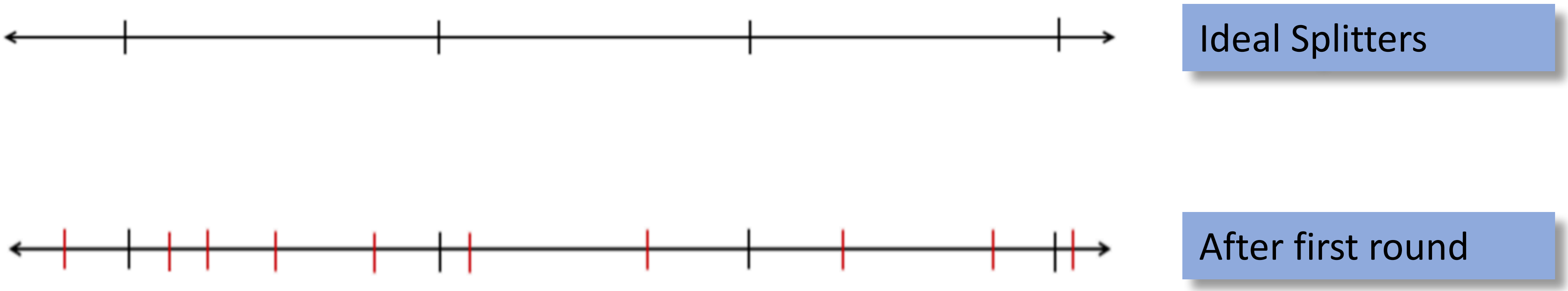
HSS: Intelligent Sampling

Find (p-1) splitter keys to partition input into p ranges



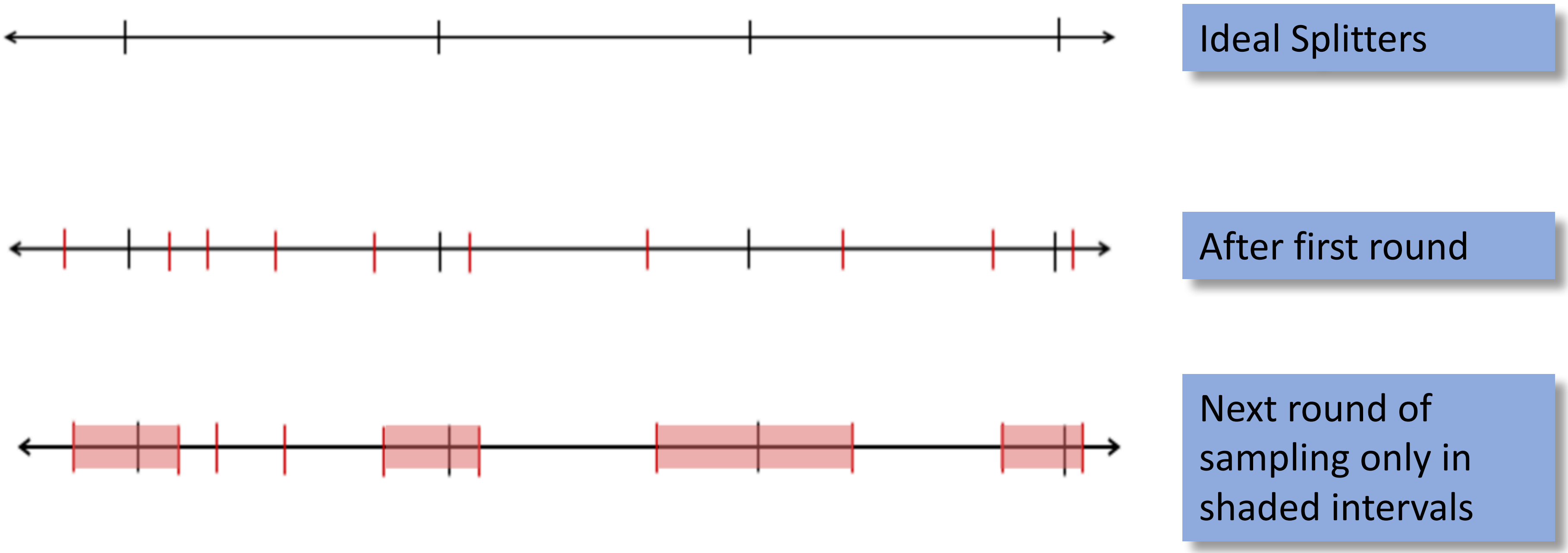
HSS: Intelligent Sampling

Find (p-1) splitter keys to partition input into p ranges



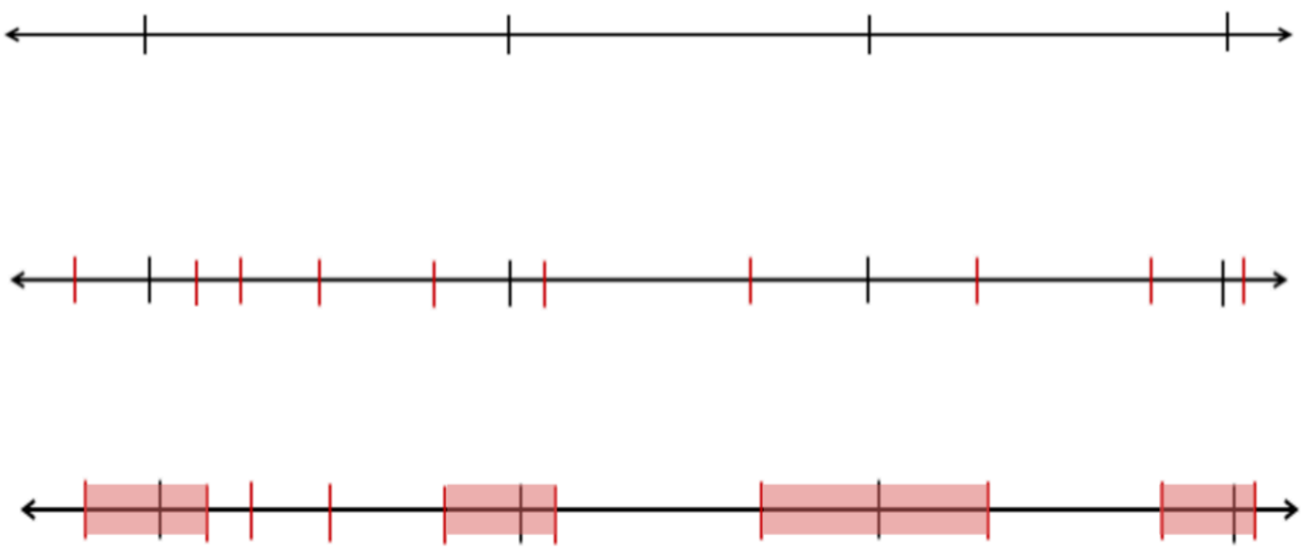
HSS: Intelligent Sampling

Find (p-1) splitter keys to partition input into p ranges



HSS: Intelligent Sampling

Find (p-1) splitter keys to partition input into p ranges



Ideal Splitters

After first round

Next round of sampling only in shaded intervals

Samples outside the shaded intervals are wasteful

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

PPL
UIUC

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp \sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

PPL
UIUC

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

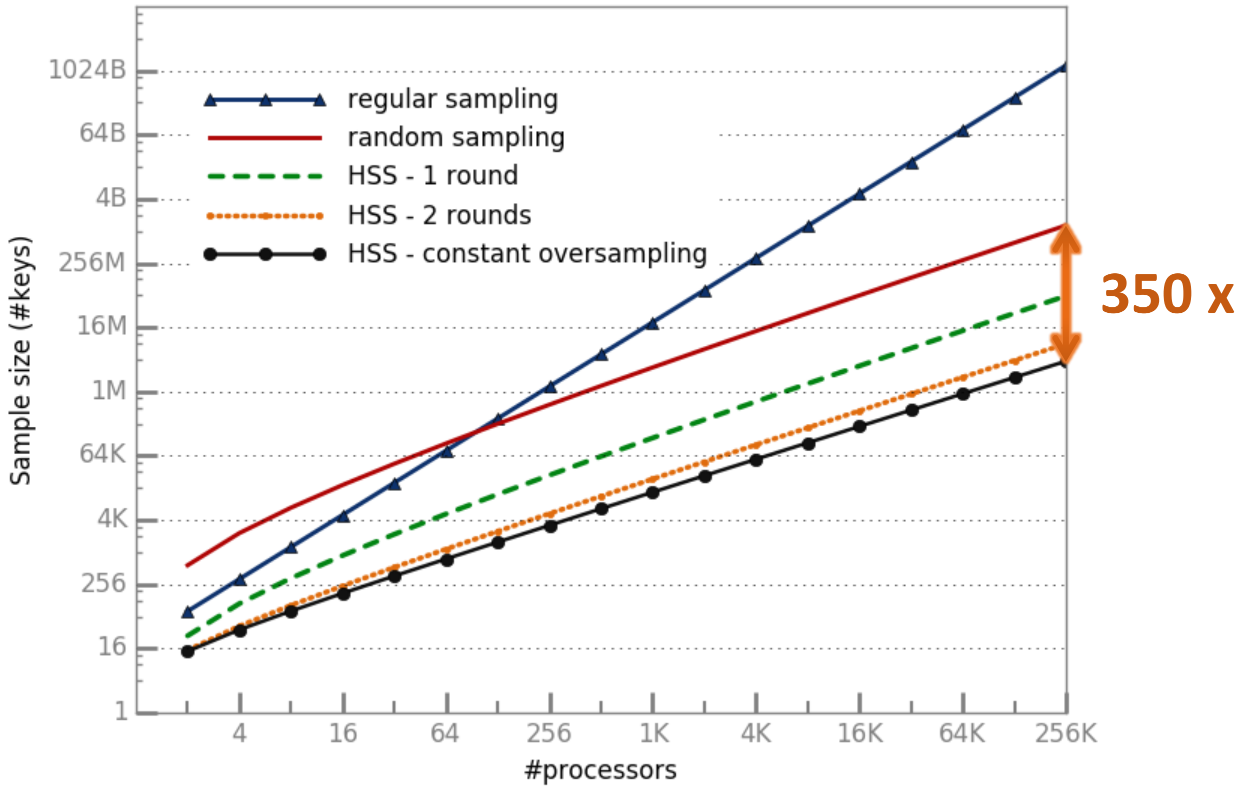
HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

HSS: Sample size

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB
Sample sort with random sampling	$\mathcal{O}(\frac{p \log N}{\epsilon^2})$	8.1 GB
HSS with one round	$\mathcal{O}(\frac{p \log p}{\epsilon})$	184 MB
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-
HSS with $\mathcal{O}(\log \frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p \log \frac{\log p}{\epsilon})$	10 MB

HSS: Sample size



64 bit keys, 5% load imbalance

Number of histogram rounds

p (x 1000)	sample size/round (x p)	Number of rounds	Number of rounds (Theoretical)
4	5	4	8
8	5	4	8
16	5	4	8
32	5	4	8

Number of rounds hardly increases with p → log (log p) complexity

Optimizing for shared memory

- Modern machines are highly multicore
  - BG/Q: 64 hardware threads/node
  - Stampede KNL(2.0): 272 hardware threads/node

- How to take advantage of within-node parallelism?

Final All-to-all data exchange

- In the final step, each processor sends a data message to every other processor

- O(p^2) fine grained messages in the network

Final All-to-all data exchange

- In the final step, each processor sends a data message to every other processor

- O(p^2) fine grained messages in the network

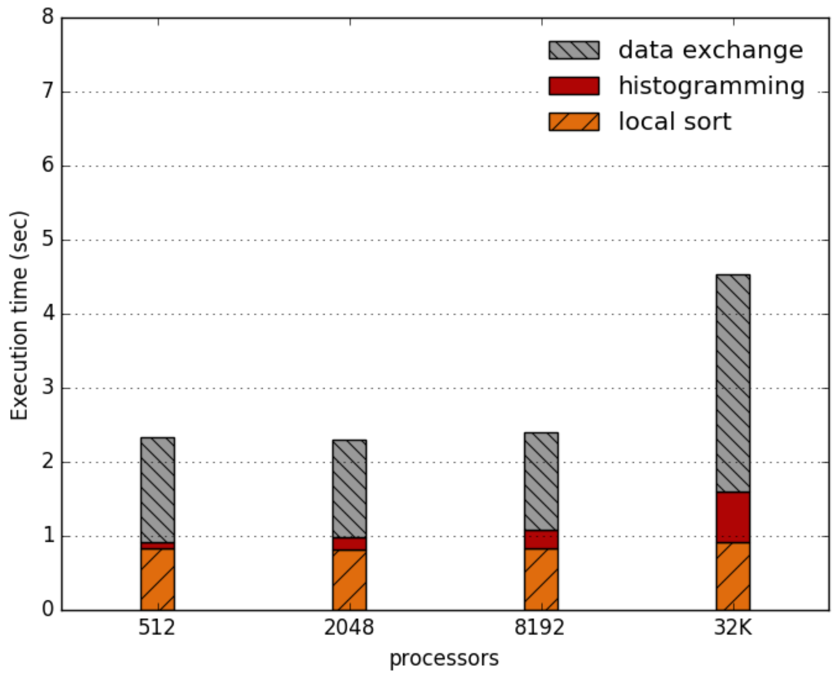
- What if all messages having the same source, destination node are combined into one?

- Messages in the network: O(n^2)
    - Two orders of magnitude less!

What about splitting?…

- We really need splitting across nodes rather than individual processors
- (n-1) splitters needed instead of (p-1)
  - An order of magnitude less
  - Reduces sample size even more
- Add a final within node sorting step to the algorithm

Execution time breakdown



Very little time is spent on histogramming!

Weak Scaling experiments on BG/Q Mira with 1 million 8 byte keys and 4 byte payload per key on each processor, with 4 ranks/node

Conclusion

- HSS combines sampling and histogramming to accomplish fast splitter determination
- HSS provides sound theoretical guarantees
- Most of the running time spent in local sorting & data exchange (unavoidable)

PPL
UIUC

Future work

- Integration in HPC applications (e.g. ChaNGa)

Future work

- Integration in HPC applications (e.g. ChaNGa)

Acknowledgements

- Edgar Solomnik
- Omkar Thakoor
- ALCF

Thank You!

Thank You!

Backup slides

HSS: Computation/Communication complexity

Algorithm	Overall sample size	Overall sample size for $p = 10^5, \epsilon = 5\%$	Computation complexity	Communication complexity
Sample sort with regular sampling	$\mathcal{O}(\frac{p^2}{\epsilon})$	1600 GB	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + \frac{p^2}{\epsilon}\log p + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(\frac{p^2}{\epsilon} + p + \frac{N}{p}\right)$
Sample sort with random sampling	$\mathcal{O}(\frac{p\log N}{\epsilon^2})$	8.1 GB	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + \frac{p\log N\log p}{\epsilon^2} + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(\frac{p\log N}{\epsilon^2} + p + \frac{N}{p}\right)$
HSS with one round	$\mathcal{O}(\frac{p\log p}{\epsilon})$	184 MB	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + \frac{p\log p}{\epsilon}\log N + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(\frac{p\log p}{\epsilon} + p + \frac{N}{p}\right)$
HSS with two rounds	$\mathcal{O}(p\sqrt{\frac{\log p}{\epsilon}})$	24 MB	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + p\sqrt{\frac{\log p}{\epsilon}}\log N + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(p\sqrt{\frac{\log p}{\epsilon}} + p + \frac{N}{p}\right)$
HSS with k rounds	$\mathcal{O}(kp\sqrt[k]{\frac{\log p}{\epsilon}})$	-	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + kp\sqrt[k]{\frac{\log p}{\epsilon}}\log N + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(kp\sqrt[k]{\frac{\log p}{\epsilon}} + p + \frac{N}{p}\right)$
HSS with $\mathcal{O}(\log\frac{\log p}{\epsilon})$ rounds	$\mathcal{O}(p\log\frac{\log p}{\epsilon})$	10 MB	$\mathcal{O}\left(\frac{N}{p}\log\frac{N}{p} + p\log\frac{\log p}{\epsilon}\log N + \frac{N}{p}\log p\right)$	$\mathcal{O}\left(p\log\frac{\log p}{\epsilon} + p + \frac{N}{p}\right)$

PPL
UIUC