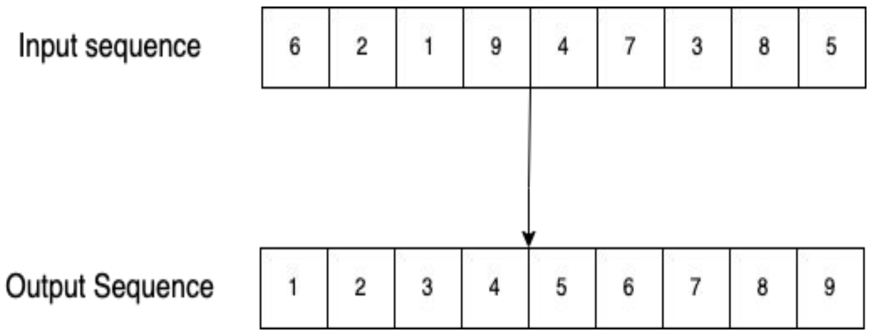
Histogram Sort with Sampling

Course: COMP 5704
Parallel Algorithms and Application in Data Science
Megha Agarwal

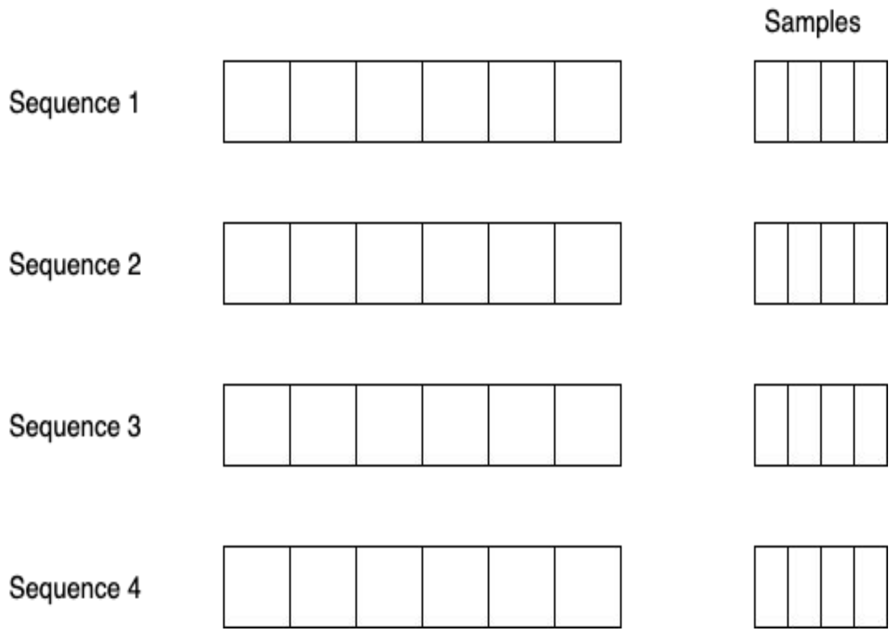
Introduction

- What is Sorting?
- Why do we need it?
- Types of Sorting:
  - Sequential
  - Parallel
- Chief goal of parallel sorting



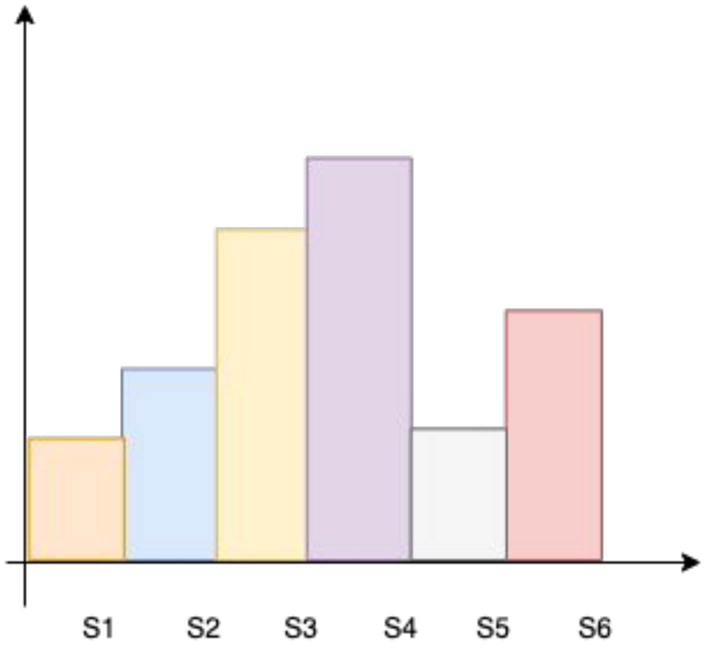
Sample Sort

1. Sample: *ps* keys
2. Split: *p-1* keys are selected
3. Exchange data: p^{th} bucket to p^{th} processor



Histogram Sort

1. Broadcast a probe
2. Create local histogram
3. Sum to form global histogram
4. Finalize splitters



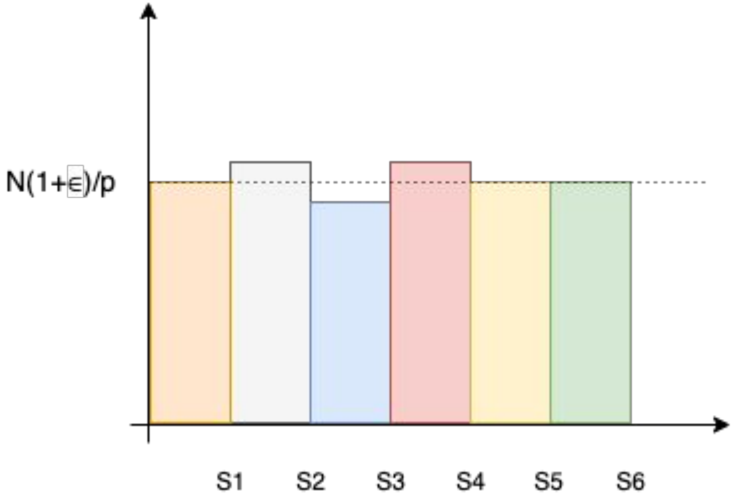
Histogram Sort with Sampling

- Goal: Approximate Splitting => $N(1+\varepsilon)/p$
- Processor *i* owns all keys greater than equal to *S(i)* and less than *S(i+1)* := Global balancing
- Two Major steps:
    - Histogramming
    - Sampling
- Input keys: A(0),...,A(N-1)
- Redistributed to: I(0),...,I(N-1), where if A(j)=I(k), it has rank *k*.

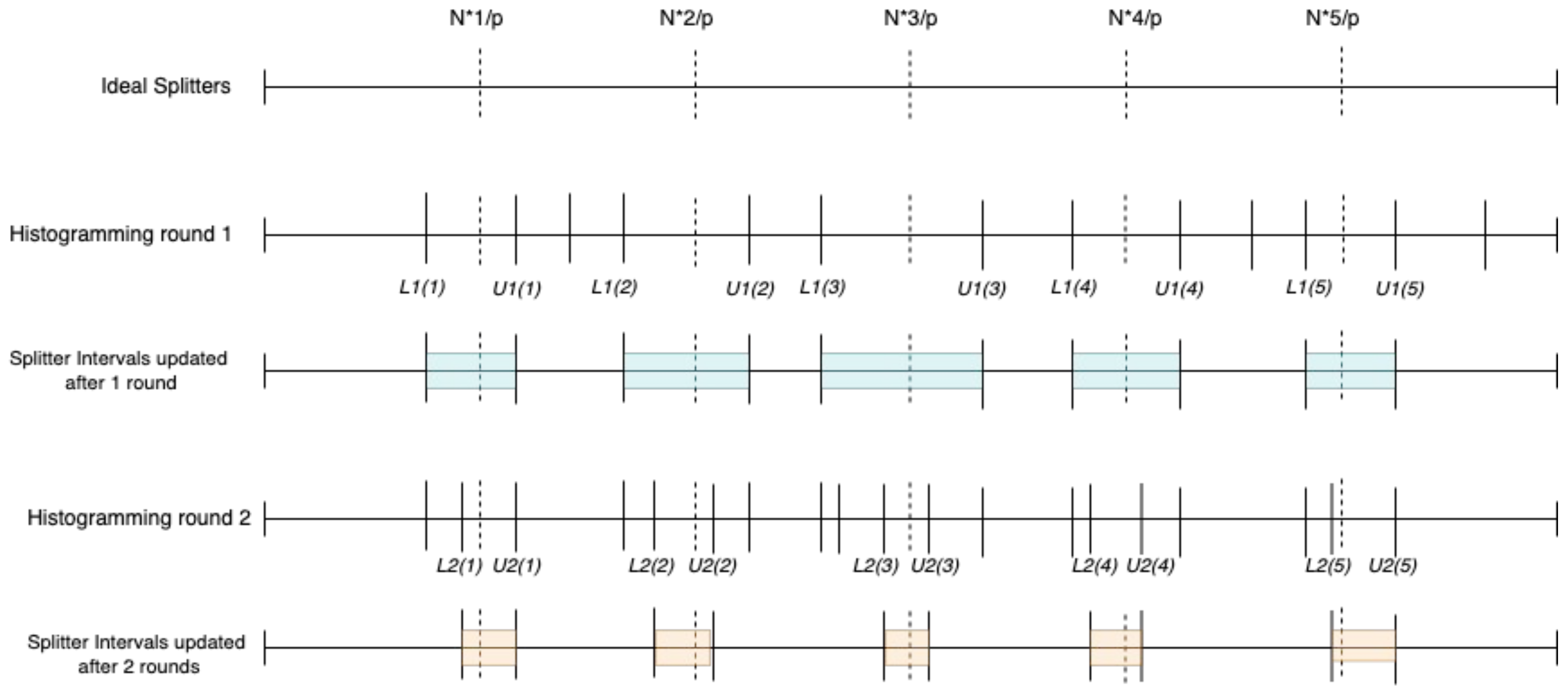
- Satisfactory splitter: S(i) = I(χ(i)), where χ(i) $\in \mathcal{T}_i = \left[\frac{Ni}{p} - \frac{N\epsilon}{2p}, \frac{Ni}{p} + \frac{N\epsilon}{2p}\right]$

Histogram Sort with Sampling

1. Each processor picks samples with probability ps_1/N and broadcast.
2. Create a local histogram at each processor and summed at central processor
3. Maintain a lower and upper bound $L_j(i)$ and $U_j(i)$ and update splitter intervals
4. Sample using new intervals for j+1th round.
5. If $j=k$,
   a. Histogramming phase is complete and move to next step.
   b. Else, if $j<k$, samples are collected at central processor, and move towards next round of histogramming.
6. Finally, the key closest to Ni/p is chosen as the ith splitter.



Ideal Situation


N*1/p
N*2/p
N*3/p
N*4/p
N*5/p
Ideal Splitters
Histogramming round 1
L1(1)
U1(1)
L1(2)
U1(2)
L1(3)
U1(3)
L1(4)
U1(4)
L1(5)
U1(5)
Splitter Intervals updated after 1 round
Histogramming round 2
L2(1)
U2(1)
L2(2)
U2(2)
L2(3)
U2(3)
L2(4)
U2(4)
L2(5)
U2(5)
Splitter Intervals updated after 2 rounds

Experimental setup

Charm++

- C++ based
- Supports MPI communication protocol
- Divides into processor elements called *chares*.
- Steps:
  - Local Sorting
  - Splitter Determination
  - Data exchange

Expected results of HSS

- Sampling ratio s= $O\left(p\sqrt[k]{\frac{\log p}{\epsilon}}\right)$ and *k= log(log p/ϵ)*
- *O(p)* samples can achieve global sorting in *O(log N/p + log log p)* rounds
- Costs:
  - Computation in local Sorting: *O((N/p) log N/p)*
  - Sampling: *O(S)* at local processor and *O(S log p)* for sorting at central processor
  - Computing local histogram: *O(S log N/p)*
  - Computation of sampling and histogramming per stage: *O(r log((log r)/ϵ)) log N)*
- Communication overhead due to multiple stage sorting.
- Overall histogramming rounds: Θ(*log log p*)
- Overall sample size: Θ*(p log log p)*

HSS compared to other algorithms

AMS-sort

- Better for one round of histogramming by $\Theta(min(log\ p,\ 1/\varepsilon))$
- HSS achieves a globally-balanced splitting, making it easily generalizable
- Takes approximately *3x* time for splitting phase than HSS

HykSort

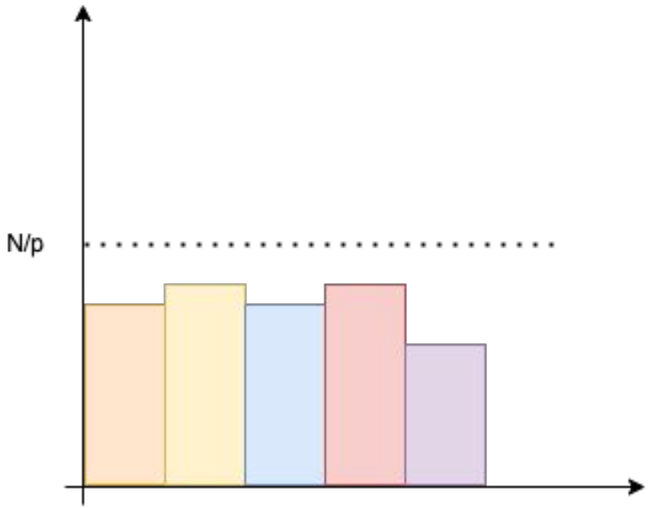
- Requires at least $\mathbf{\Omega}(log(p)/log^2 log(p))$ times more samples.
- Slower convergence of splitters

Questions

1. Do more processors mean better performance?

2. Is it possible to have the resulting histogram look like this?

3. What would the worst case be?



Thank you!