EEF Summer School on Massive Data Sets

Duke University
Center for Geometric & Biological Computing
Department of Computer Science

Jeff Vitter

Distributed Sorting with Multiple Disks
Review of Distribution Sort?

S-way Distribution Sort:

✶ 1. If the input stream (bucket) fits into memory, sort it and quit;

✶ 2. Otherwise

   • [Splitter Selection Phase] Choose $S - 1$ splitters.
   • [Distribution Phase] Read the input and distribute data into buckets as determined by the splitters.
   • Sort each bucket recursively.
If each pass can be done in $O(S \log N) O$ I/Os total, the optimal choice of $S$ is

$$S = \left( \frac{B}{W} \right)^{1/\Theta} \left( \frac{\log N}{B} \right)$$

in internal memory for each bucket. Distribution sort requires a (double) buffer.

- number of independent disks: $D$
- size of disk block: $B$
- size of internal memory: $M$
- problem data size: $N$

[Vitter & Shriver, 90, 94] Parallel Disk Model
Example: $D = 3$ disks, $S = 3$ buckets:

Data streams through internal memory and is partitioned online.

**Challenge**: Each bucket must be distributed among the disks in an online manner. How can we prevent write bottlenecks at the disks? That is, how should we lay out each bucket on the disks?
What is the Challenge?

<table>
<thead>
<tr>
<th>Input Bucket</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Input Bucket" /></td>
<td><img src="image2" alt="Input Bucket" /></td>
<td><img src="image3" alt="Input Bucket" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Bucket</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="Output Bucket" /></td>
<td><img src="image5" alt="Output Bucket" /></td>
<td><img src="image6" alt="Output Bucket" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Bucket</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image7" alt="Output Bucket" /></td>
<td><img src="image8" alt="Output Bucket" /></td>
<td><img src="image9" alt="Output Bucket" /></td>
<td></td>
</tr>
</tbody>
</table>

- Read $D$ blocks (one block per disk) in each input operation.
- Write $D$ blocks (one block per disk) in each write operation.
- Buckets fill at different rates (no problem if only one disk).
Gilbreath Principle

Writing is also no problem if we have only two buckets (streams).

- We can achieve perfect balance for writing two buckets:

<table>
<thead>
<tr>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket 1:</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>J</td>
<td>K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bucket 2: (striped in reverse order)</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>F</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>K</td>
<td>J</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

- Each write of four blocks from the two blocks is guaranteed to be perfectly striped across the disks!
- Reduces necessary buffer space by half.
- Cannot be generalized to $R > 2$. 

Jeff Vitter
The Power of Queueing the Writes

Blocks formed by Distribution process
(labeled by disk destination)

Need pool of $D$ queue buffers (one per disk) in internal memory.

Write cycle: For each nonempty queue, write a block to its disk.

After each write cycle, bring in $(1 - e) \cdot D$ block arrivals.

Problem: If the queues fill up memory, we need to flush them, which takes many I/Os.

The challenge is to show that the total queue space stays small.
individual queues.

The analyses are complicated by dependencies among the sizes of the

use a different random permutation of the disk numbers.

4. RCD: Randomized Cyclic Distribuzione

for each bucket, disk in round-robin order.

choose a new random starting disk and allocate the blocks to

for each consecutive set of blocks allocated to that bucket,

for each bucket,

3. RSD: Randomized Striped Distribuzione

the disks in round-robin order.

select a starting disk then allocate the bucket's blocks to

for each bucket,

2. SRD: Simple Randomized Distribuzione

randomly select a destination disk.

for each block,

1. PRD: Fully Randomized Distribuzione

Randomized Distribuzione on Parallel Disks
Bucket Distribution Variants

Each crosshatched disk block involves a random placement decision.
FRD (Fully Randomized Distribution)
SRD (Simple Randomized Distribution)
RSD (Randomized Striping Distribution)
RCD (Randomized Cycling Distribution)
Previous Work and Our Results

- SRM: Simple Randomized Mergesort [Barve and Vitter].
- Analysis of FRD recently given by [Sanders, Egner, Korst SODA’00] using negative dependence property.
- In this talk we reduce RCD (practical) to FRD (not practical) and thus bound the write I/Os of RCD by that of FRD.
  - (Expected) I/O complexity is optimal.
  - only a constant number of queued blocks per disk are required (on average).
- RCD read complexity is optimal; but not FRD’s.
- RCD is simple to implement.
- Experiments confirm theoretical indications.
Outline

1. Analysis of FRD, RCD
   - FRD guarantees and drawbacks
   - RCD reduction to FRD
   - RCD I/O bounds
2. Experiments
   - FRD, RCD, SRD, RSD
3. Non-sorting Applications
4. Future Work
Analysis of Queue Space Needed

Example: $D = 3$ disks, $S = 3$ buckets

Blocks formed by Distribution process
(labeled by disk destination)

Perform write cycle
every $(1 - \epsilon) \cdot D$
block arrivals.

$Q_i^{(t)} = $ size of queue $i$ (in blocks) at time $t$

$Q^{(t)} = $ total queue space $= \sum_i Q_i^{(t)}$

We use $\hat{Q}_i^{(t)}$ and $\hat{Q}^{(t)}$ as corresponding terms for FRD.
Push the queues (expensive operation).

If ever the total queue size is $M$,

A total of $1 - e$ blocks arrive in queues.

Each nonempty queue writes one block to its disk.

At each read step,

$\text{Total Queue Size is } W = O(D/e)$ blocks.
\[ N \approx \mathcal{D} \]

Pushing the queues at very end may be nonoptimal for \( \mathcal{RD} \), if bounded as \( t \to \infty \).

Let \( \mathcal{D} \) be the bandwidth for writing. It allows the total queue size to stay at the peak arrival rate of \( (1 - \epsilon) \) represents a fraction of the peak.

Then \( \mathbb{P}(\mathcal{D} > 2) \) for all \( \epsilon > 0 \).

Then \( \mathbb{P}(\mathcal{D} < 2 - \epsilon) \) for all \( \epsilon > 0 \).

Then \( \mathbb{E}(\mathcal{D} < 2 - \epsilon) \) for all \( \epsilon > 0 \).

Let \( \nu \) be the number of write steps for the \( t \)th read step.

Then \( \frac{1}{\mu} + (1 - \epsilon) \mathbb{E}(\mathcal{D}) > \mathbb{E}(\mathcal{D}) \) for some \( \epsilon > 0 \).

Then \( \mathbb{E}(\mathcal{D}) = \mathbb{E}(\mathcal{D}) \) for some \( \epsilon > 0 \).

Center for Geometric & Biological Computing

Theorem 1:

Lemma 2:

Lemma 1:
random variable:

\[ p(\theta, \alpha) \]

arrive at each time unit is distributed as a

One block can leave per time unit. The number of blocks that

\[ \gamma \sim \nu(d - 1, \gamma) \]

\[
\begin{pmatrix}
d \\
u
\end{pmatrix} = \{(\gamma, \nu) \}
\]

\[
\text{Prob} \{X \}
\]

with \((d - 1, \theta)\) trials and probability \((\alpha, \theta)\) are independent binomially distributed

\[ X, X, X, X, X \]

\[ Y \]

Let \(Y\) be the number of blocks arriving in the read step.
\[ y \sum_{0 \leq y} \left( \sum_{0 \leq y} \right) \left( x + \sum_{0 \leq y} \right) = (z) X_G \]

Properties

 Variable \( X \) encodes complete information about the distribution of random variable  \( X \).

\[ y \sum_{\{ y = X \} \text{Prob}} \left( \sum_{0 \leq y} \right) = (z) X_G \]
\[ (z)^{(1+\tau)^2}X \mathcal{D} \times (z)^{(1+\tau)^2}X \mathcal{D} = (z)^{(1+\tau)^2}X \mathcal{D} \]

By independence of \( (\lambda)^2 \mathcal{X} \) and \( (\lambda)^2 X \): 

\[
(\lambda)^2 \mathcal{X} \text{ and } (\lambda)^2 X
\]

\[
\# = (1+\tau)^2 X
\]

newly arriving blocks for queue \( \mathcal{Z} \)

\[
\# = (1+\tau)^2 \lambda
\]

blocks still in queue from previous time step, and

\[
(1+\tau)^2 X + (1+\tau)^2 \lambda = (1+\tau)^2 \mathcal{D}
\]

The recurrence for queue size is

\[
\begin{cases}
  0 & \text{otherwise} \\
  1 - (\lambda)^2 \mathcal{D} & I \leq (\lambda)^2 \mathcal{D} \quad \text{if otherwise}
\end{cases}
\]

\[
= (1+\tau)^2 \lambda
\]
\[
\alpha^{(\omega-1)} \left( \frac{a}{1 - a + z} \right) = (z)^{(1+i)^\omega} \mathcal{D} \quad \iff \\
(1-a+z) \frac{a}{1} = \\
\frac{a}{z} + \frac{a}{1-a} = \\
\frac{1}{z} \frac{a}{1} + 0 \frac{a}{1-a} = (z)^{(1+i)^\omega} \mathcal{D}
\]

\text{(sum of independent RVs)}

\[
(z) \alpha^{(\omega-1)} L + \cdots + (z)^{iL} L + (z)^{1L} = (z)^{(1+i)^\omega} X
\]

\text{Newly Arriving Blocks}
\[
\begin{align*}
\left( \frac{z}{1 - I} \right) \left( 0 \right)_{(i)^2i} + \left( z \right)_{(i)^2i} \frac{z}{1 - I} &= \\
\left( 0 \right)_{(i)^2i} \frac{z}{1 - I} - \left( 0 \right)_{(i)^2i} + \left( z \right)_{(i)^2i} \frac{z}{1 - I} &= \left( z \right)_{(i + i)^2} \frac{1}{I} \\
\left( 0 \right)_{\mathcal{V}} \frac{z}{1 - I} - \left( 0 \right)_{\mathcal{V}} + \left( z \right)_{\mathcal{V}} \frac{z}{1 - I} &= \cdots + \frac{z}{1 - I} + \frac{z}{0} + \frac{z}{0} = \left( z \right)_{\left[ 0 = \mathcal{V} \right] + I - \mathcal{V}} \\
\left( z \right)_{\mathcal{V}} \frac{z}{1 - I} &= \cdots + \frac{z}{1 - I} + \frac{z}{0} + \frac{z}{0} = \left( z \right)_{\left[ 0 = \mathcal{V} \right] + I - \mathcal{V}} \\
\cdots + \frac{z}{1 - I} + \frac{z}{0} + \frac{z}{0} &= \left( z \right)_{\mathcal{V} \frac{z}{1 - I}} \\
\frac{0 = \left( i \right)_{i} + I - \left( i \right)_{i}}{	ext{otherwise}} &\left\{ \begin{array}{l}
0 \\
I \leq \left( i \right)_{i} \frac{I}{I - \left( i \right)_{i}} 
\end{array} \right. \\
&= \left( i + i \right)_{i} \end{align*}
\]
\[
\left. \frac{z}{1} - a^{(p-1)} \frac{d}{1 - a + z} \right/ \frac{z}{1} - (0) \mathcal{C} = (z) \mathcal{C} \quad \Leftarrow \\

\left. a^{(p-1)} \left( \frac{d}{1 - a + z} \right) \times \left( (0) \mathcal{C} \frac{z}{1} - (0) \mathcal{C} + (z) \mathcal{C} \frac{z}{1} \right) \right/ = \\
(z) \mathcal{C} \times (z) \mathcal{C} = (z) \mathcal{C} \\
\left. (z) \mathcal{C} = (z) \mathcal{C} \right/ = (z) (1 + \mathcal{C}) \mathcal{C} \quad \text{In steady state, as } t \to \infty , \ \Leftarrow \\
\]
\[
E(Q(t)) \leq \frac{(1 - z)e}{1 - G(z)}
\]

By L'Hôpital's rule, we know that \( G(1) = 1 \). By L'Hôpital's rule,

\[
\lim_{z \to 1} \left( \frac{G(0)(1 - \frac{1}{z})}{D(1 - \frac{1}{z})} \right) = \frac{G(0)}{e}
\]
Result of Lemma 2

\[ \{ b < (\infty)^t \} \geq \text{the steady-state tail probability} \]

Therefore, the tail probability is \( \{ b < (\infty)^t \} \) is the steady state queue lengths are not empty.

\[ 0 = (\infty)^t \]

The queues are initially empty at time \( t \), i.e., \( 0 = (\infty)^t \).

In any step the difference in length remains the same or gets

which point they remain the same forever.

Consider two queues processing identical input but with different

\[ 0 < b < (\infty)^t \] for all \( t \):

We now show that...
\[ b_{\varepsilon} - \varepsilon \mathcal{Z} = b_{\varepsilon} - \varepsilon \mathcal{G} > \{ b < (\infty) \mathcal{G} \} \text{prob} \iff b = I \text{ and } 1 < \varepsilon = \varepsilon \mathcal{G} \text{ Substituting:} \]

\[ \cdots + z^{I+\mu}d + z^{\mu}d + \cdots + I + z^{1}d + z^{0}d = \]

\[ I < z \iff \forall I \quad (z)^{X} \mathcal{G}_{\mu = -z} \supseteq \]

\[ \cdots + I+\mu d + \mu d = \{ \mu \leq X \} \text{prob} \]

General Tail Inequality:

\[ \begin{align*}
1 > |x| \quad \text{for } x > \cdots - \frac{\varepsilon}{\varepsilon x} + \frac{\varepsilon}{\varepsilon x} - x = (x + 1) \text{ using the bound in } 1
\end{align*} \]

\[ z > \frac{(I - \varepsilon \mathcal{G})(\varepsilon - I) - \varepsilon}{(\varepsilon \mathcal{G} - I) \varepsilon} \mathcal{G} \Rightarrow \frac{(\varepsilon \mathcal{G})^{z} \mathcal{G}}{(\varepsilon \mathcal{G} - I) \varepsilon} - I = (\varepsilon \mathcal{G}) \mathcal{G} \]

Result of Lemma 2
Lemma 3

\[ a \in \mathbb{R}^+ \implies \frac{1}{b} \geq \exp\left(\lambda - \frac{d}{\lambda} \right) \]

where \( \lambda \) is a parameter that can be set.

\[ \frac{\exp(-d)}{\lambda} < \{a^b < (\lambda)\} \]

Let \( (\lambda) \) be the total queue capacity is \( M \). Let \( (\lambda) \) blocks arrive in queues.

Each nonempty queue writes one block to its disk.

At each read step,
item in one queue affects the other queue negatively. This is, in a sense, better than independence. Placing an item in a queue will then be shorter. The sizes of the other queues will then be equal. If an item is placed in a queue, then it cannot be placed in any of the other queues. Negative Association (NA)
\[
\mathbb{E} \left( \left( \sum_{i} \mathbb{E} \left( \bar{\mathcal{S}}^\alpha \right) \right)^{\mathcal{M}^s - \mathcal{C}} \right) > \mathbb{P} \left( M < \sum_{i} \mathbb{E} \left( \bar{\mathcal{S}}^\alpha \right) \right) \]

Let \( M \) be the allowable total memory space (in blocks):
Choose $s = \varepsilon$.

From Lemma 2, we know $E\{\gamma = (\gamma)\}_{\theta}^{\varepsilon}$

Proof of Lemma 3
\[(a)v^{-\epsilon} + 1 \geq a^g - \epsilon \left( \frac{\epsilon}{d} \right) O + 1 = (d + M)d + 1 \geq \left( (t)u \right) \exists\]

The expected number of write steps at time \( t \) is 

In the worst case it takes \( d + M \) steps to push the queues.

\[a^g - \epsilon \geq \{ M < (t)Q \}\]<br>

Theorem 3 gives the probabilities that the queues are flushed.
Main Theorem

- The Main Theorem states that RCD has the same performance guarantees as does FRD. (In fact, they’re better, because of the final emptying of the queues is guaranteed to be balanced.)

- The challenge is to show that the expected exponential of the total queue space \( E(e^{sQ(t)}) \) in RCD is at most that of FRD:

  \[
  \text{that is, } E(e^{sQ(t)}) \leq E(e^{s\tilde{Q}(t)})
  \]

- We would then inherit the desired tail bound on the total queue size of RCD:

  \[
  \text{Prob}\{Q(t) > W\} = \text{Prob}\{e^{sQ(t)} > e^{sW}\} \\
  < e^{-sW} E(e^{sQ(t)}) \text{ by tail inequality} \\
  < e^{-sW} E(e^{-s\tilde{Q}(t)}) \\
  = e^{-\delta D}
  \]
Key Lemma: Each transformation step causes the quantity $E(\theta_{(i)})$ to increase or stay the same. At the end, it is $E(\theta_{(i)})$.

Create a new singleton bucket with its block at time step $t$.

Remove one block from bucket $q$ at time step $t$.

Do the following transformation step.

Issues at least one block at time step $t$.

While there is a non-singleton bucket $q$ that

For $r = 1$ to $t$ do

(each block is randomly assigned to a disk)

$RCD$ in which all buckets are singletons

$\equiv$ FRD

bucket that contains a total of one block

singleton bucket

Reduction of RCD to FRD
(that the disks are arranged in cycle order.)

Assume WLOG that the other blocks can stay where they are and

What is the effect of removing the first block issued by bucket b at

disk 2, 3, 4, 5, 6, 7, 8,

time step t.

Layout of Bucket b on the Disks
Analogy with a Lake

☆ Suppose each day the sun removes a gallon of water from the lake.

☆ Then, later in the day, it may or may not rain. If it rains, the lake gets some added water.

☆ If the lake always has at least two gallons at the start of each day, then if we remove a gallon of water in April, it will have a gallon less in September.

☆ If on the other hand, the lake has only one gallon at the start of June 28, then the sun will empty the lake. Therefore, if we remove a gallon in April, there will be no change in September.
A Critical Queue (a Sufficiently Full Lake)

<table>
<thead>
<tr>
<th>$t'$</th>
<th>$r$</th>
<th>$r + 1$</th>
<th>...</th>
<th>$t - 1$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_{i}^{(t')}$</td>
<td>$\geq 2$</td>
<td>$\geq 2$</td>
<td>...</td>
<td>$\geq 2$</td>
<td>$\geq 2$</td>
</tr>
<tr>
<td>Item Arrivals</td>
<td>•</td>
<td>...</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
</tbody>
</table>

- The size of the $i$th queue $Q_{i}^{(t')}$ is at least 2 for $r \leq t' < t$.
- $Q_{i}^{(t')}$ will remain at least 1 even without the arrival at time step $r$, and a block will continue to be consumed at each time step.
- Hence, if there is no arrival of a block into the $i$th queue at time $r$ (keeping all other block arrivals the same), the final size $Q_{i}^{(t)}$ of the queue will be one less than before.
Proof

★ \( Q^{(t)} \) is the sum of queues at time \( t \).
★ \( Q^{(t)}' \) is the sum of queues at time \( t \) after the block is removed.
★ \( Q^{(t)}'' \) is the sum of queues at time \( t \) after bucket \( b \) has been transformed.

Then

\[
Q^{(t)}'' = Q^{(t)}' + \text{[new bucket increases queue size]}
\]

We want to show

\[
E(f(Q^{(t)}'')) \geq E(f(Q^{(t)})),
\]

where \( f(x) = e^{sx} \).
Proof

Suppose that $c$ of the $D$ possible starting points for bucket $b$ are critical with respect to time step $t$.

**Case 1: Starting Point is Critical**

$Q''(t)$ is either $Q(t) - 1$ or $Q(t)$

$$E( f(Q''(t)) \mid \text{the starting point is critical} )$$

$$\geq \left( 1 - \frac{c}{D} \right) E( f(Q(t) - 1) \mid \text{starting point is critical} )$$

$$+ \frac{c}{D} E( f(Q(t)) \mid \text{starting point is critical} )$$

$$= \left( \left( 1 - \frac{c}{D} \right) \frac{1}{f(1)} + \frac{c}{D} \right) E( f(Q(t)) \mid \text{starting point is critical} ),$$

since $f(x) = e^{sx}$ and thus $f(Q(t) - 1) = \frac{1}{f(1)} f(Q(t))$. 

Jeff Vitter
Proof

Case 2: Starting Point is Non-critical

\[ Q''(t) \text{ is either } Q(t) \text{ or } Q(t) + 1 \]

\[
E\left( f(Q''(t)) \mid \text{starting point is non-critical} \right) \\
\geq (1 - \frac{c}{D}) E\left( f(Q(t)) \mid \text{starting point is non-critical} \right) \\
\quad + \frac{c}{D} E\left( f(Q(t) + 1) \mid \text{starting point is non-critical} \right) \\
= \left( (1 - \frac{c}{D}) + \frac{c}{D} f(1) \right) E\left( f(Q(t)) \mid \text{starting point is non-critical} \right),
\]

since \( f(x) = e^{sx} \) and thus \( f(Q(t) + 1) = \frac{1}{f(1)} f(Q(t)) \).
\( (2) \quad \text{Starting point is non-critical}) \quad \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} + \left( \frac{a}{\epsilon} - 1 \right) \left( \frac{a}{\epsilon} - 1 \right) + \left( \frac{a}{\epsilon} + \frac{1}{\epsilon} \left( \frac{a}{\epsilon} - 1 \right) \right) \frac{a}{\epsilon} \right) < \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} \right) = \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} \right)

\text{After Transformation}

\( (1) \quad \text{Starting point is non-critical}) \quad \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} - 1 \right) + \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} \right) = \mathcal{E} \left( (\partial f) \frac{a}{\epsilon} \right)

\text{Before Transformation}
Starting point.

Starting point and the cycle orders with a non-critical
proof uses a mapping between the critical orders with a critical

Intuition: Let $1 \geq c \geq d$ be the number of critical starting points. Then

$$
\left( \frac{1}{f} (\Omega f) \right) \mathbb{E} \frac{(1)f}{I} = \\
\left( \frac{1}{f} (\Omega f) \right) \mathbb{E} \left( \left( 1 - (1)\Omega f \right) \frac{a}{z} \right) + \\
\left( \frac{1}{f} (\Omega f) \right) \mathbb{E} \left( \left( 1 - (1)\Omega f \right) \frac{a}{z} \right) \frac{(1)f}{I} \\
\left( \frac{1}{f} (\Omega f) \right) \mathbb{E} - \left( \frac{1}{f} (\Omega f) \right) \mathbb{E}
$$

Lemma 5:

Proof

Combining (1) and (2) from the previous slide we get

Combining (1) and (2) from the previous slide we get
Experimental Results

Testing with smaller numbers of disks shows that even non-asymptotic behavior is attractive. Test parameters:

☆ Block arrival regimes:
  1. “Random input”: next bucket to receive a block is chosen randomly.

☆ Small and large $\epsilon$. Can $\epsilon = 0$? (That is, can we write out a full $(1 - \epsilon)D = D$ items in each write cycle?)

☆ Wait for steady state and then record the histogram of the total queue space (i.e., total memory space) used.
Random Issue, $\epsilon = 0.3$

Buckets issue Blocks in Random Order
N=2000000 D=10 S=50 epsilon=0.3

Frequency

Total Queue Space (blocks)
Round-Robin Issue, $\epsilon = 0.3$

Buckets Issue Blocks in Round-Robin Order
$N=2000000$ $S=50$ $D=10$ $\epsilon=0.3$

Center for Geometric & Biological Computing

Jeff Vitter
Random Issue, $\epsilon = 0.1$

Buckets issue Blocks in Random Order
N=2000000 D=10 S=50 epsilon=0.1

Graph showing the frequency distribution of total queue space (blocks) for different methods: RCD, SRD, RSD, FRD.
Round-Robin Issue, $\epsilon = 0.1$

Buckets Issue Blocks in Round-Robin Order
N=2000000 S=50 D=10 epsilon=0.1

Center for Geometric & Biological Computing
Random Issue, $\epsilon = 0$

Buckets issue Blocks in Random Order
N=2000000 D=10 S=50 epsilon=0

RCD
SRD
RSD
Round-Robin Issue, $\epsilon = 0$

Buckets Issue Blocks in Round-Robin Order
$N=2000000 \ S=50 \ D=10 \ \epsilon=0$

- RCD
- SRD
- RSD
Conclusions and Future Work

- RCD is a simple, practical, and provably good method for sorting with parallel disks.
- We conjecture that SRD and RSD perform similarly to RCD.
- Randomized cycling can be applied to merge sort to get a practical and theoretically optimal sorting algorithm.
- RCD can be used in distribution sweeping applications.
- We are starting practical implementation/study.