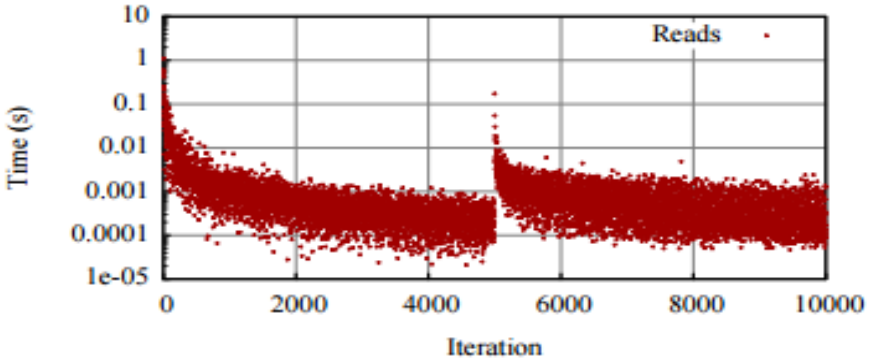
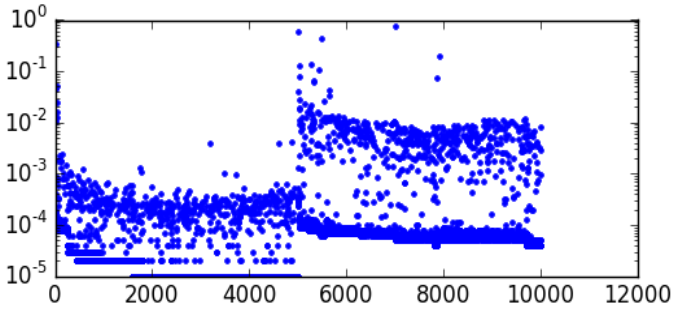
Policy Exploration for JITDs (Java)

By Team Datum

Cracking Results from Paper vs. Observed Results



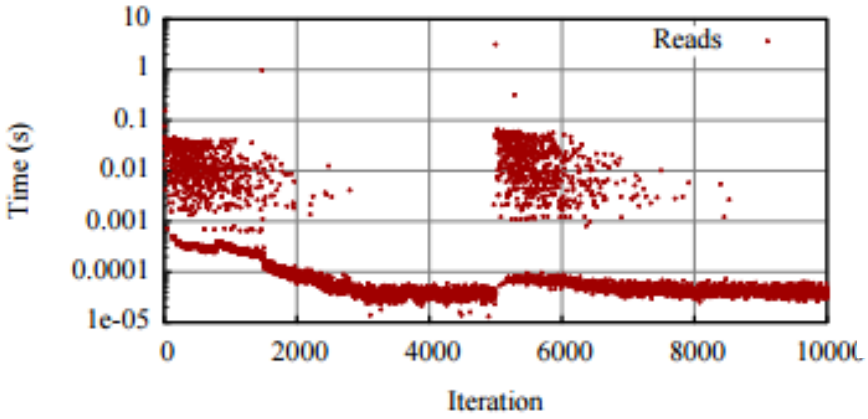
(a) Primitive Policy: Cracking



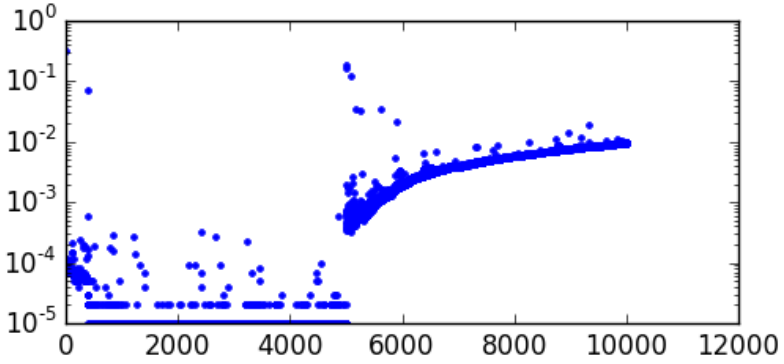
Tested with :

mode cracker
init 100000000
seqread 5000
write 10000000
seqread 5000

Adaptive Merge Results from Paper vs. Observed Results



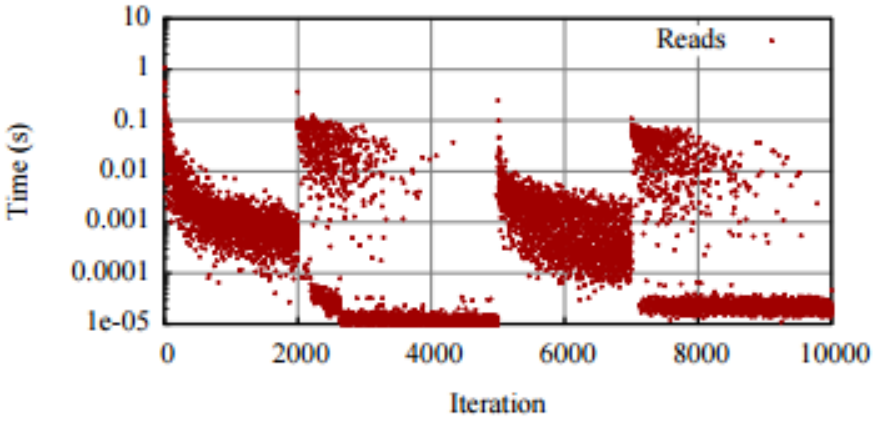
(b) Primitive Policy: Adaptive Merge



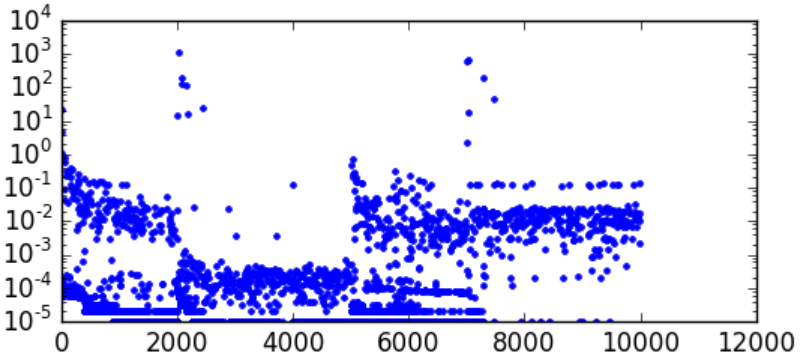
Tested with :

mode merge
init 100000000
seqread 5000
write 10000000
seqread 5000

Comparison of Swapping Results from Paper vs. Observed Results



(c) Hybrid Policy: Swap



Tested with :

mode cracker
init 100000000
seqread 2000
mode merge
seqread 3000
write 10000000
mode cracker
seqread 2000
mode merge
seqread 3000

Past : Uniform(Random) Workload

- Currently, all the graphs are plotted using RandomIterator where the Lower bound of range query is selected at random.
- All the Data values have equal probability of Selection.
- Is this the Correct way for evaluation?

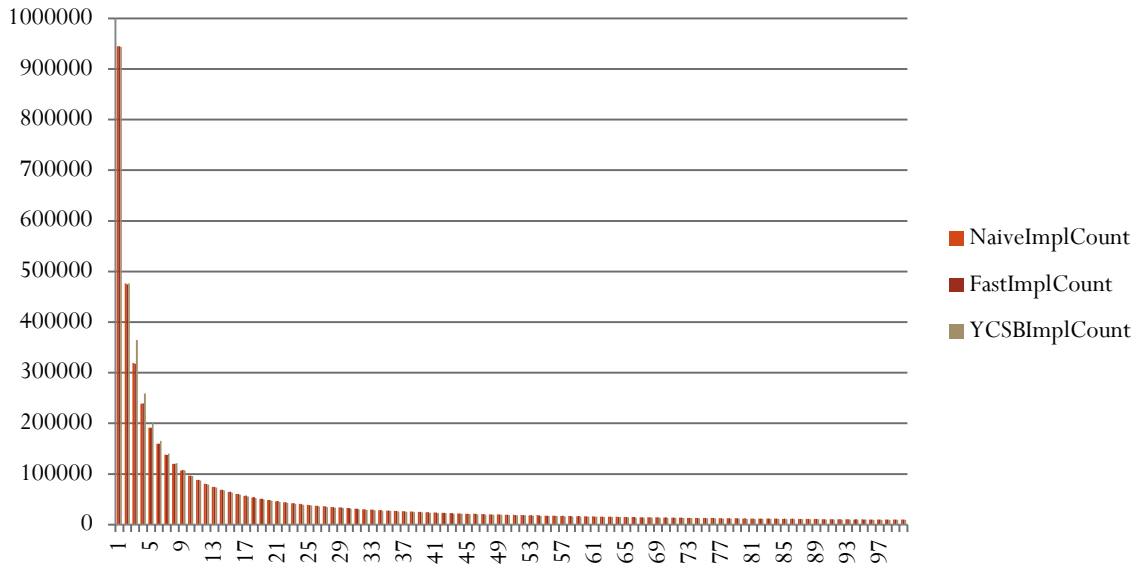
Current : Zipfian Workload

- Zipfian distribution Vs uniform distribution
- Added new Iterator that extends current KeyValueIterator.

```
public static class ZipfianIterator extends KeyValueIterator {
    Integer max;
    long key, value;
    ZipfianGenerator zipfRand = new ZipfianGenerator();
```

- Considered 3 different implementations for Zipfian Distribution Generation.
  - Naïve Zipfian Generator
    (Uses basic implementation of Zipfian distribution)
  - Fast Zipfian Generator
    (Stores values in a NavigableMap prior to the iterator's next() call)
  - YCSB's Zipfian Generator
    (Implements Zipfian distribution fully using the standard distribution form)

Distribution Stats



Total duration (in millisecs) :

NaiveImpl : 67212.710357

FastImpl : 540.022825

YCSBImpl : 950.114582

Progress

- Basic implementation of Splaying is done without the concept of Cogs. Should find the policies that fits the current implementation.
- Should find how current implementation works against Workloads following Zipfian Distribution.

JITDs on Disk

Team Warp

Animesh, Archit, Rishabh, Rohit

UPDATED FILE FORMATS TO INCLUDE NEW METADATA

Data , Separator, Data

Data,2,Data	Null,5,Null	Data,6,Data

File Pointer, Separator, File Pointer

File,2,File	Null,5,Null	File,6,File

COGS TO SUPPORT PAGING

- PageCog - deals with pages
- FileCog - deals with data in files

PAGING DATA IN AND OUT

- Basic implementation for saving index trees in pages
- Basic implementation for restoring index trees from pages
- Policies on when and what to page out
- Researching on the ideal page size

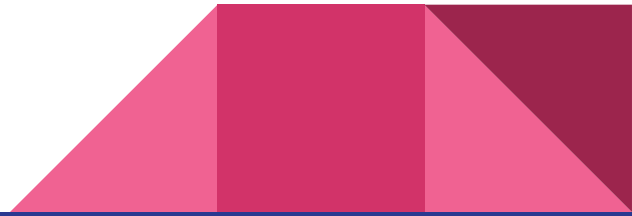
QUESTIONS?

Policy Exploration of JITDs (C)

Team Twinkle

Today’s Presentation

- Splay tree policy exploration.
- Policy implementation details.
- Tests and Test Results.



Policy 1 : Splaying

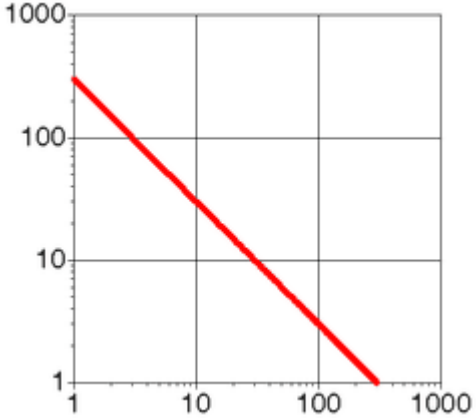
- When to Splay ?
  - Test Scenario: Splay after every 10 reads.
  - Performance benefit is summarized in the following slides.
  - It is yet to be determined the optimal time to Splay.
- How to Splay ?
  - Test Scenario: Splay on the Tree Median Btree-cog
  - Other possible splays:
    - Most recently accessed data.
    - Most frequently accessed data prior to splaying
    - Random splaying

Performance comparison of cracking with splaying vs without Splaying

For a random array of size 100000 and key range of 1000

random 10 read key range	Without Splaying (in msec.)	With Splaying (in msec.)
1000	83	78
100	6	5
10	0	1

Why Zipfian Distribution?

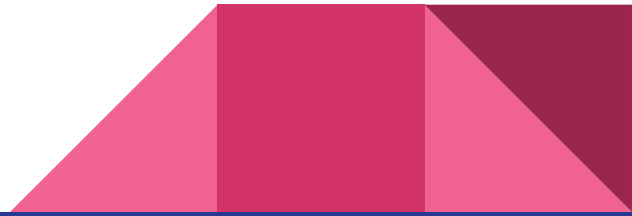


Distribution of Data Points
on Logarithmic Scale

- Real life workload.
- More selective distribution.
- Part of major benchmarking softwares like YCSB

Testing Base Setup

- **One million records** of random data created using mk_random_array() function.
- Same distribution values for the test run on both the splaying and un-splayed data-points.
- Cracking performed on the range-scan operations.
- Splaying performed after 100 reads.
- Total of 1000 reads performed on each test.
- Selectivity or range scan width changed for each test.



Results for the test

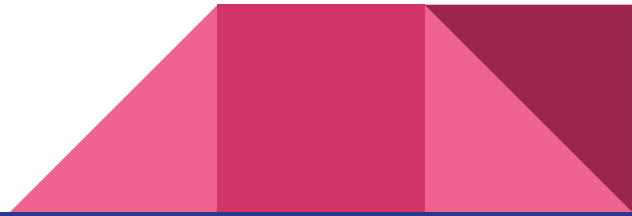
Selectivity for range scan changed.

Time in milliseconds

	Selectivity(10)	Selectivity(50)	Selectivity(100)	Selectivity(1000)
Test ran without splaying	5333	5325	5419	5319
Test ran with splaying	5142	5172	5151	5138

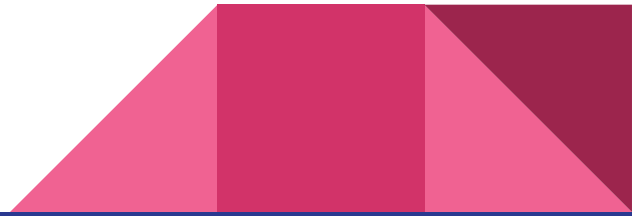
- Test ran with splaying varying splay interval
- Range scan 1000

Splaying after 5 reads	Splaying after 10 reads	Splaying after 100 reads	Splaying after 200 reads
5174	5296	5337	5239



Future Work

- Perform more testing b changing the parameters taking into consideration more factors.
- Perform read and write simultaneously into the cog and check how the performance is impacting.
- Explore other self balancing data-structures like AVL tree,Red-Black tree and perform the same workload operations.



Questions?