Is the Segment Store on Disk Fast Yet?

Progress Report Meeting
December 12, 2016

Loïc Prieur-Drevon Michel Dagenais

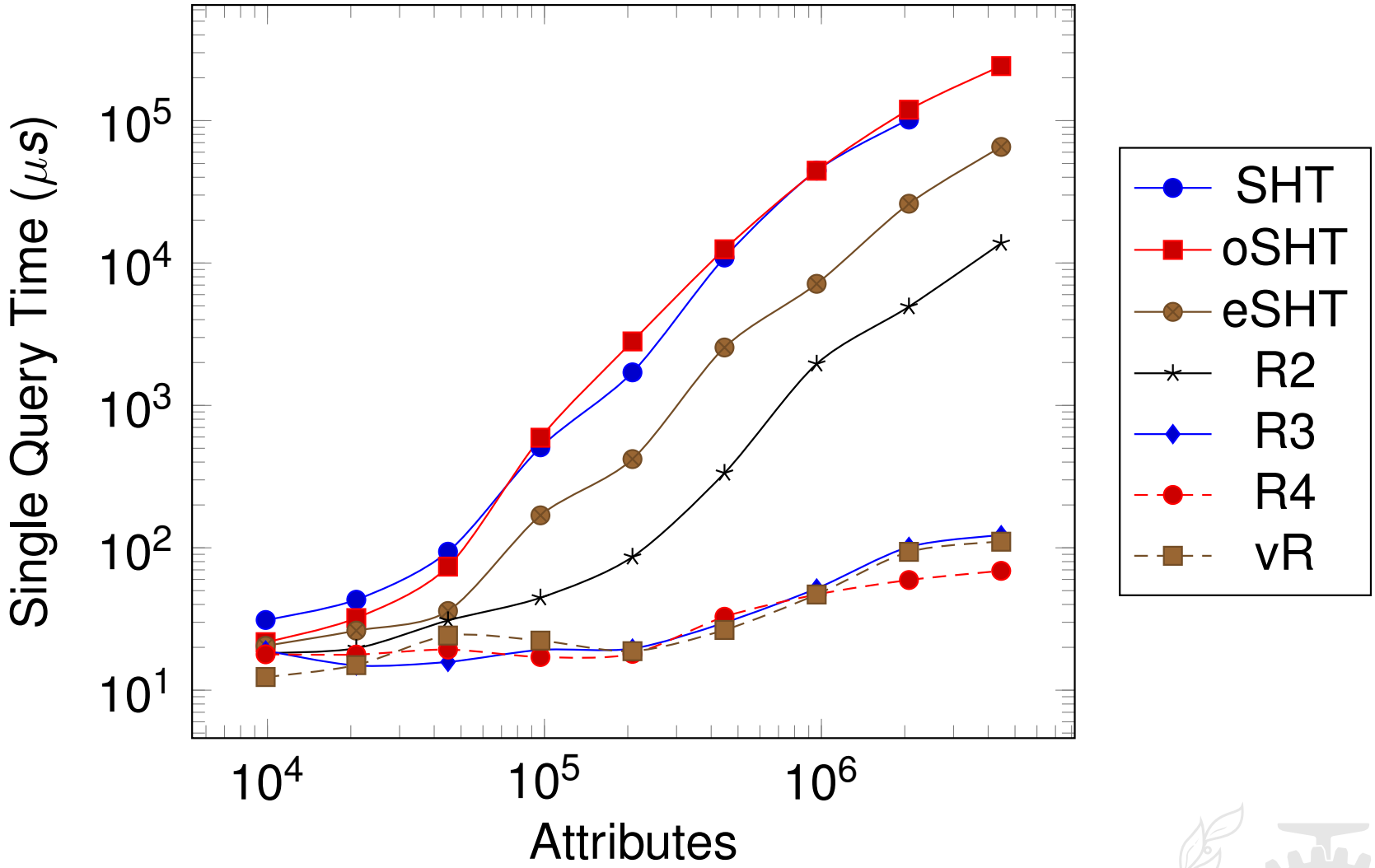
École Polytechnique de Montréal
Laboratoire **DORSAL**

Table of Contents

Publications

- **Enhanced State History Tree (eSHT) : a Stateful Data Structure for Analysis of Highly Parallel System Traces**
  Presented at IEEE Big Data Congress in San Francisco
- **R-SHT: a State History Tree with R-Tree Properties for Analysis and Visualization of Highly Parallel System Traces**
  Submitted to ACM TOMPECS

R-SHT results: Single Query Scalability



Extract data faster

Before: Single and Full query API.

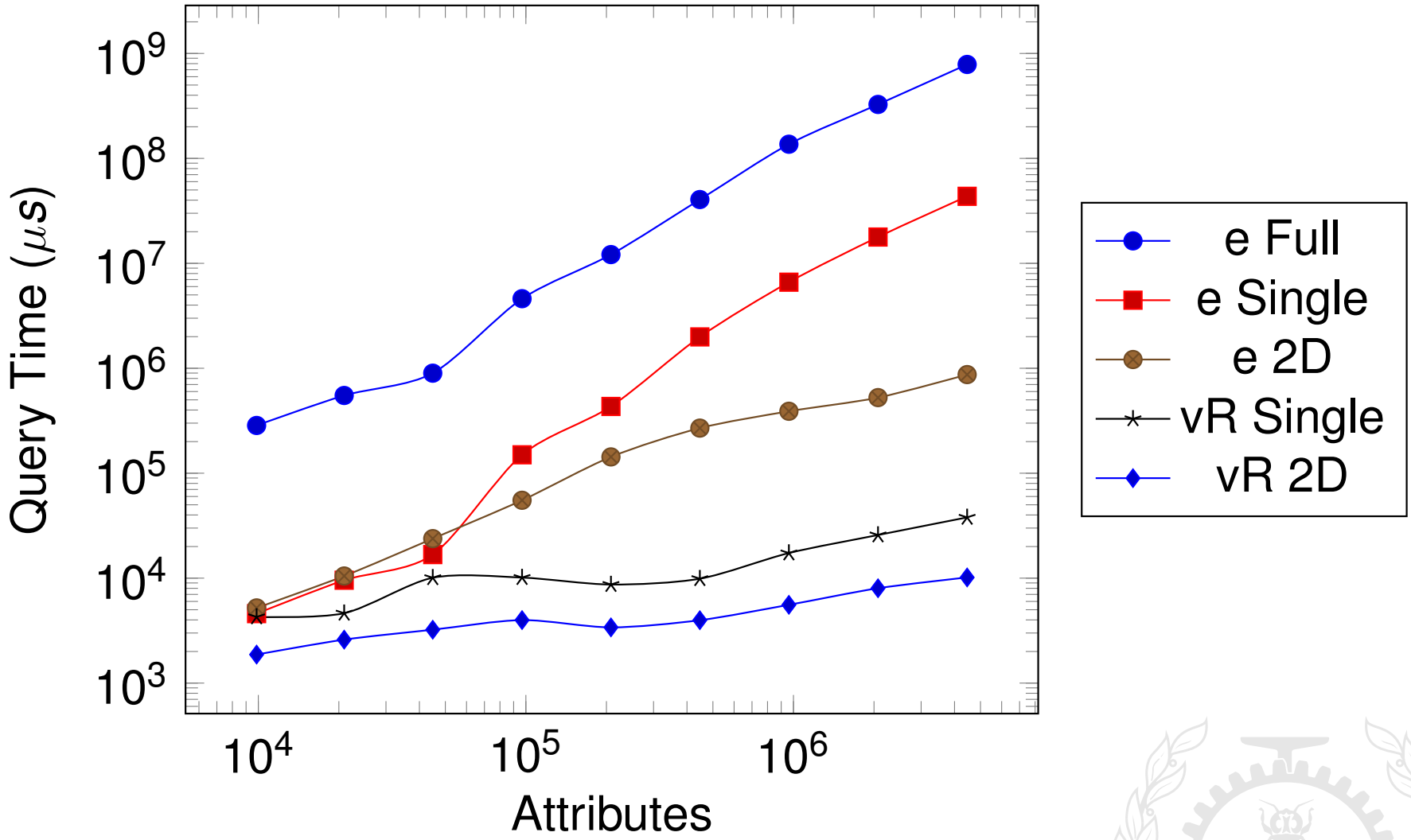
- Full Query
  - returns *all* stati at time *t*
  - requires reading many nodes – $\propto$ *attributes*
- Single Query
  - return status of one attribute at time *t*
  - requires reading half as many nodes

Extract data faster

After: 2D query API.

- **@param** range or set of attributes
- **@param** range or set of time stamps
- **@return** Iterable<StateInterval>
  - lazy
  - low memory footprint
  - unordered

R-SHT results: 2D Query Scalability



Query Type		PsTree (ms)	Zoom (ms)
SHT			
	full	5 360	29 880
eSHT			
	full	6 168	30 310
	2D	226,8	4 408
vR			
	full	17 100	40 970
	2D	135,7	4 041

Why?

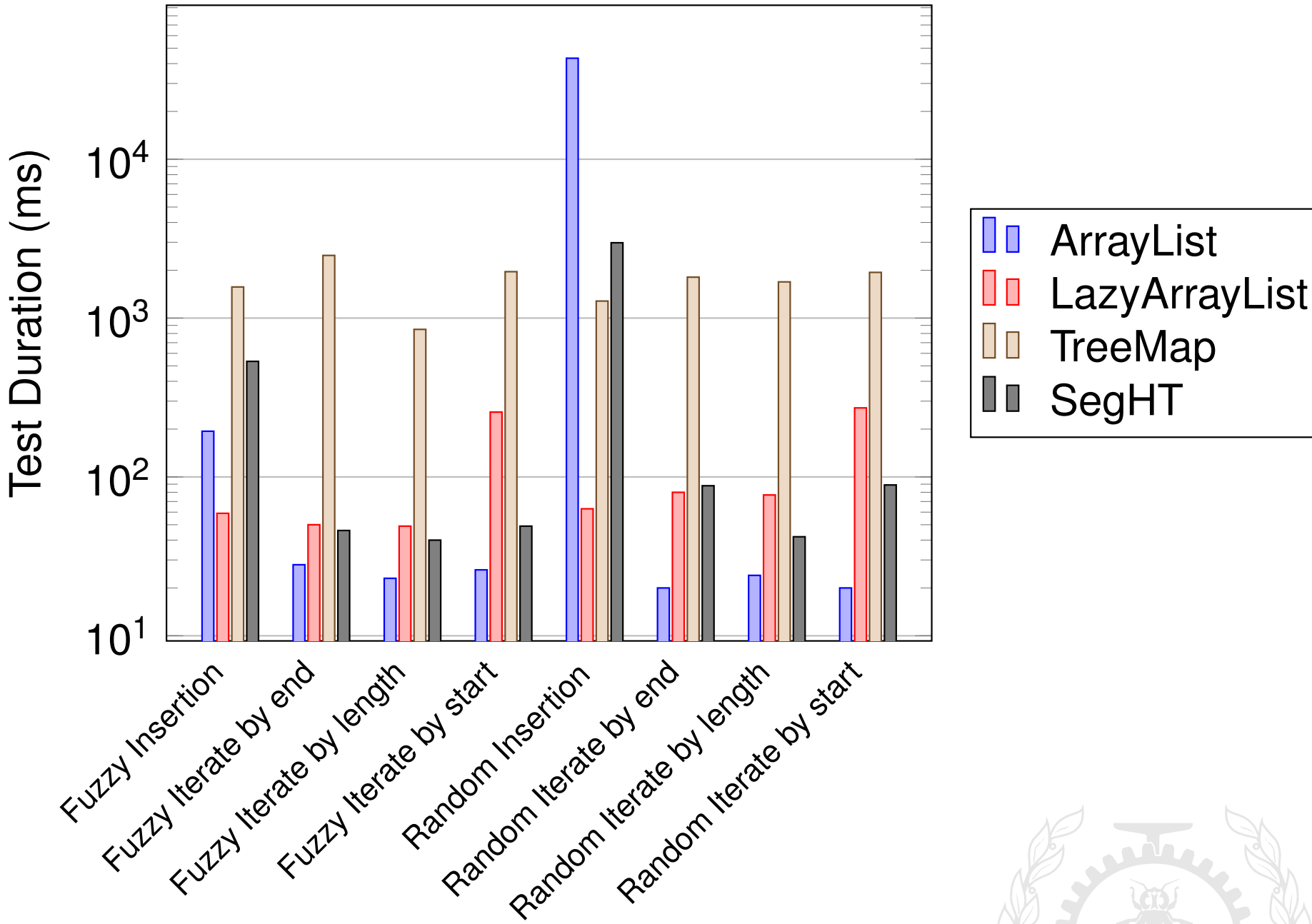
Previous implementations were based upon **TreeMap** and **ArrayList**.

IN MEMORY

Didn't scale.

Quark-less oSHT

The State History Tree stores Intervals:
Interval : $< key, start, end, value >$
Segment : $< start, end, value >$
Why not reuse the work from the State History Tree instead of starting from scratch?

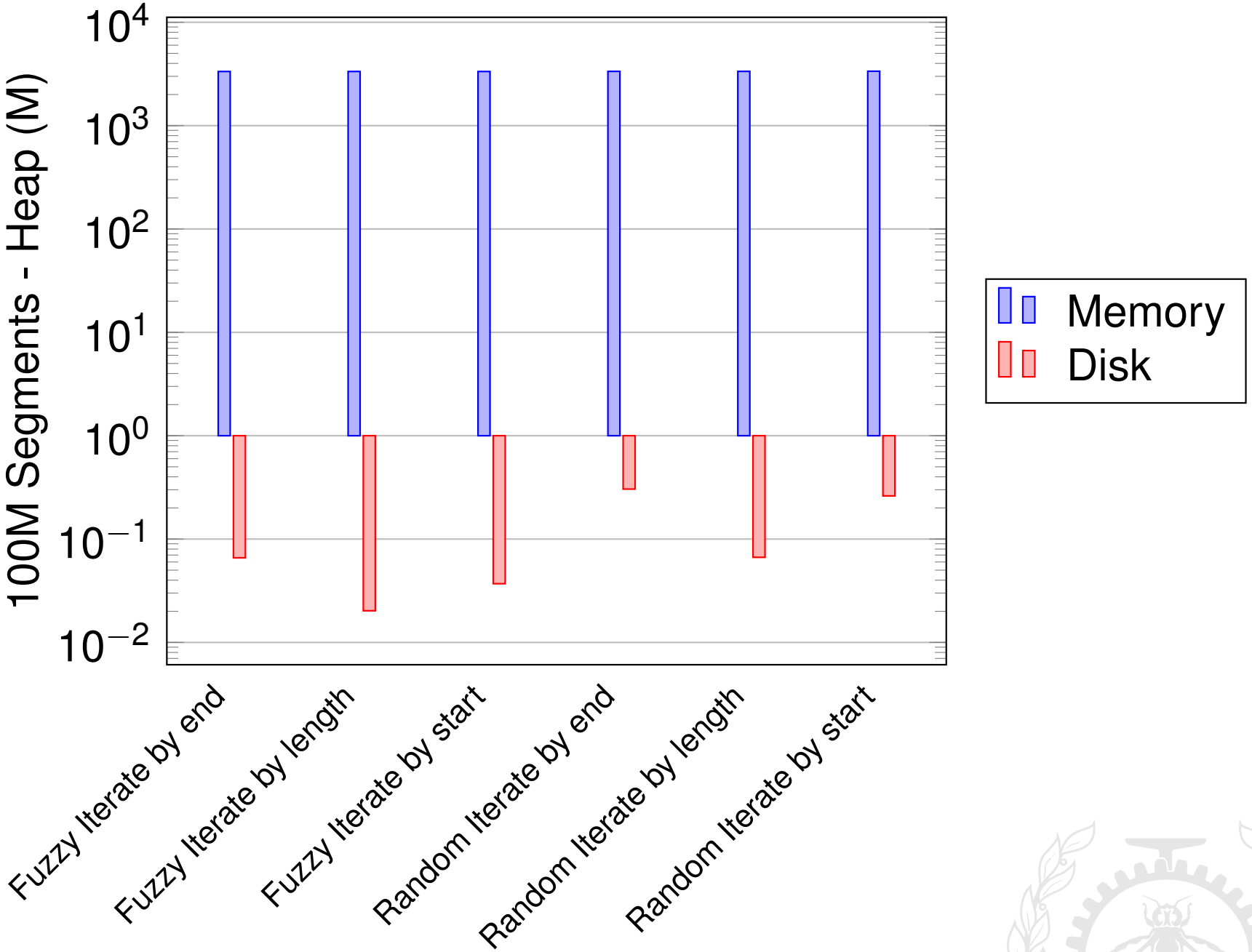

Test Duration (ms)
10^4
10^3
10^2
10^1
Fuzzy Insertion
Fuzzy Iterate by end
Fuzzy Iterate by length
Fuzzy Iterate by start
Random Insertion
Random Iterate by end
Random Iterate by length
Random Iterate by start
ArrayList
LazyArrayList
TreeMap
SegHT

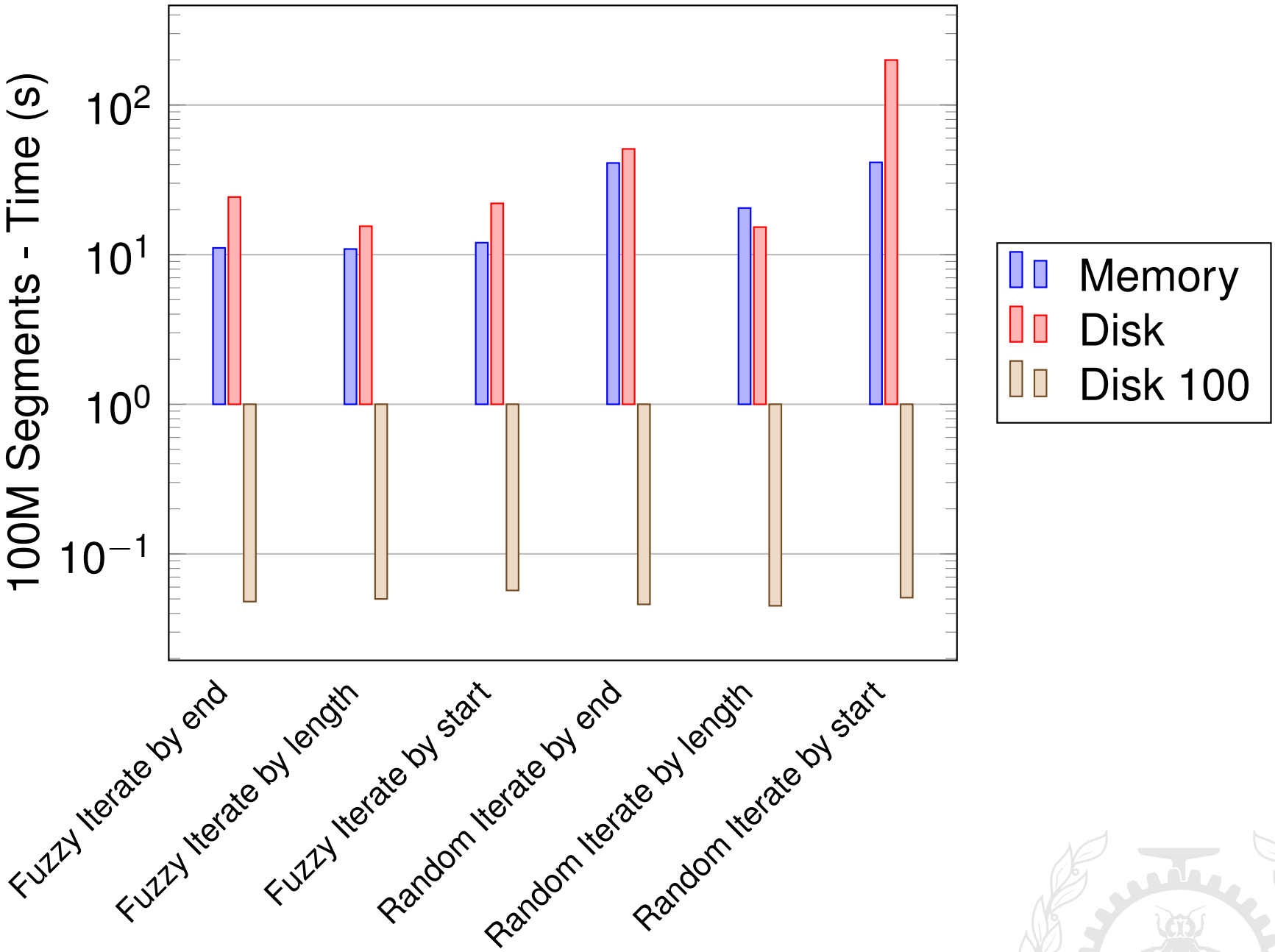
Issue: Sorting

Many views, analysis required a sorted list of segments.
Easy when the Segment Store can fit into memory.
But when they don't?
ISegmentStore.getIntersectingElements(start, end, order)

Sorted Iteration on Disk Segment Store

```
class iterator(time, order)
    nodes ← PriorityQueue(order, rootNode);
    segments ← PriorityQueue(order);
    function next()
        return segments.remove();
    end
    function hasNext()
        while segments = ∅ ∨ segments.head() > nodes.head() ∧
        nodes ≠ ∅ do
            node ← tree.readNode(nodes.remove());
            nodes.add(node.intersectingChildren(time));
            segments.add(node.intersectingSegments(time));
        end
        return segments ≠ ∅;
    end
end
```


100M Segments - Heap (M)
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
Memory
Disk
Fuzzy Iterate by end
Fuzzy Iterate by length
Fuzzy Iterate by start
Random Iterate by end
Random Iterate by length
Random Iterate by start


100M Segments - Time (s)
10^2
10^1
10^0
10^{-1}
Fuzzy Iterate by end
Fuzzy Iterate by length
Fuzzy Iterate by start
Random Iterate by end
Random Iterate by length
Random Iterate by start
Memory
Disk
Disk 100

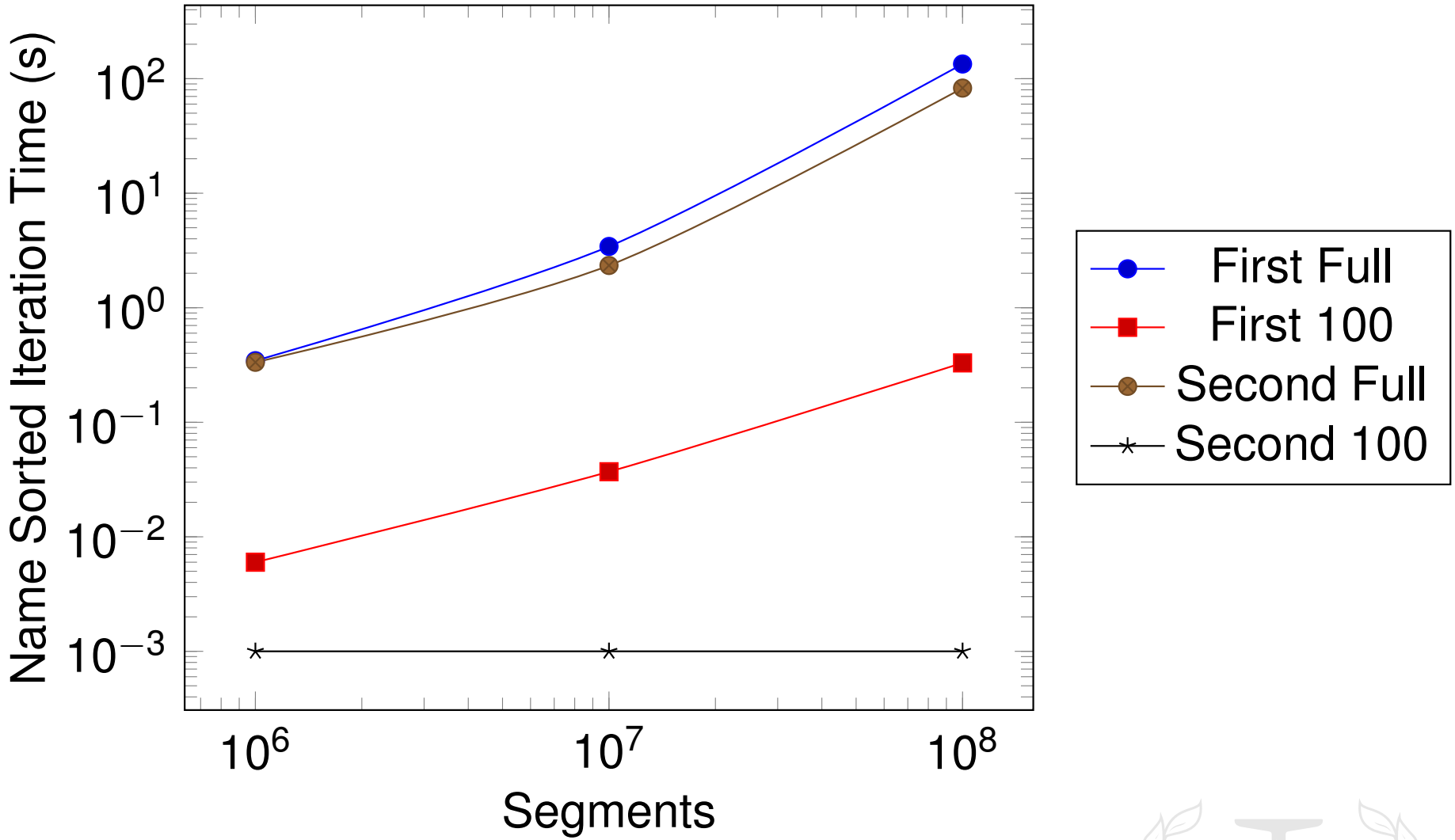
What about custom aspects?

Segments can carry a number of aspects:

- a name
- a value
- both
- anything else...

We cannot index all possible aspects.
Build an index on the side?

Before and After on the side Name index



Summary of Master's results

Scale	Before	After
Size on Disk	$O(Max(A^2, I))$	$O(I)$
Depth (D)	$O(Max(A, log(I)))$	$O(log(I))$
CFE-entryList	$O(W \times D)$	$O(I)$
CFE-zoom	$O(W \times D)$	$O(log(W \times I))$
Mem CFE-SS	$O(W \times D)$	$O(1)$
Max SegStore	RAM	Disk
SegStore – RAM	segments	log(segments)

W = screen width

I = nb Intervals

What's next?

- Finish Thesis
- Get patches into Trace Compass
- Get a job $\vee$ Do a PhD

loic.prieur-drevon@polymtl.ca