Decoupling Indexing and Search Scalability

- Vigya Sharma, Tue Bui

Who are we?

Vigya Sharma

- Apache Lucene Committer since Aug, ‘22
- Search and Distributed Systems Engineer at Amazon
- vigyasharma@apache.org

Tue Bui

- Search Engineer for about 8 years
- Currently, Sr. Software Engineer at Amazon Search

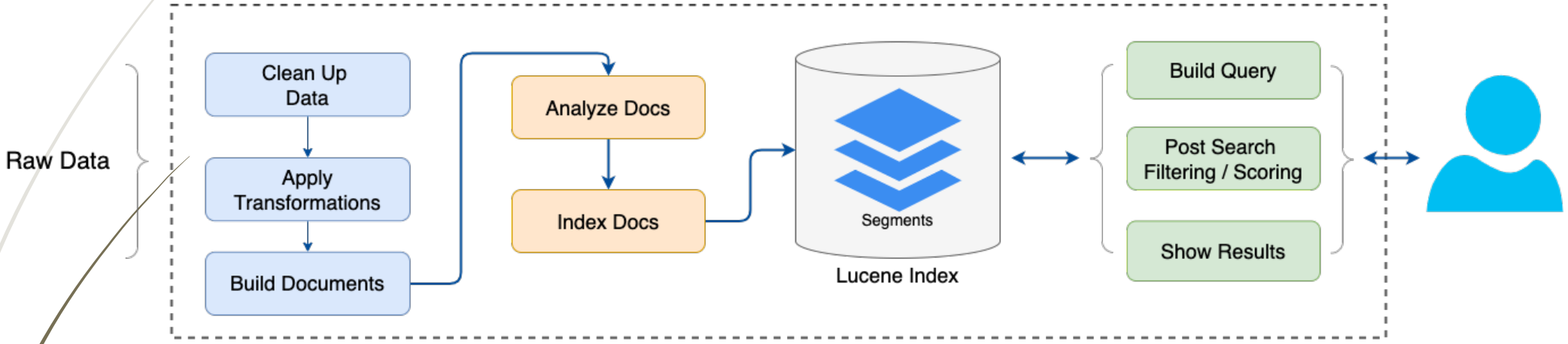
Outline

1. The Problem
2. Scaling Challenges with NRT Updates
3. Solution Approach
4. New in Lucene 9.4.0!
5. Road to Production
6. Early Performance Numbers
7. Q/A

“Let’s start at the very beginning.
A very good place to start…”

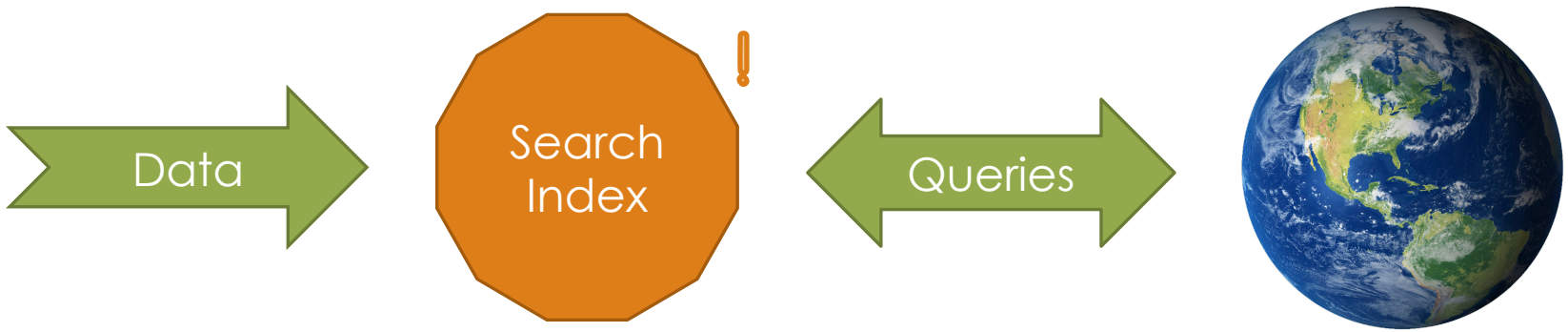
- The Sound of Music

A typical Lucene Search App.

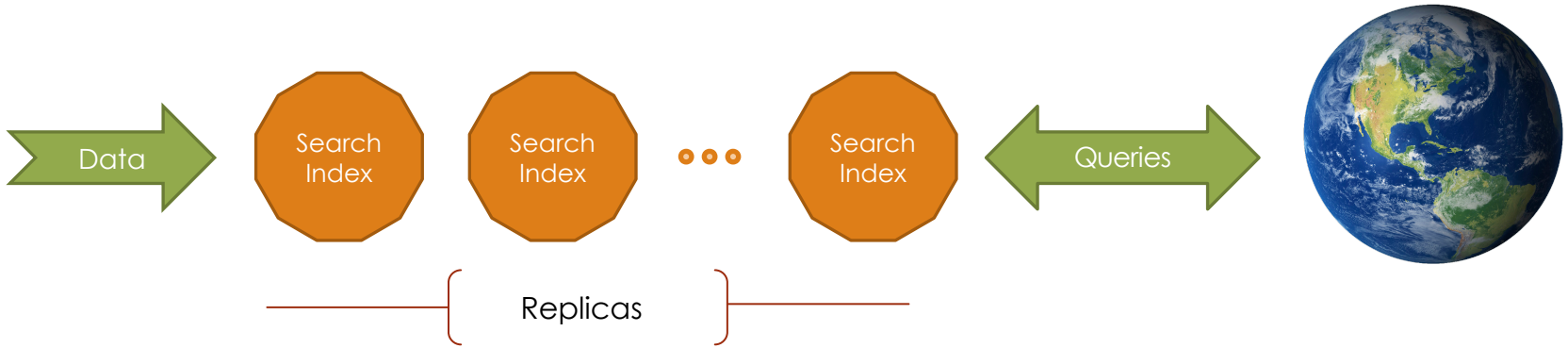


Scaling for Search Traffic

Scaling for Search Traffic

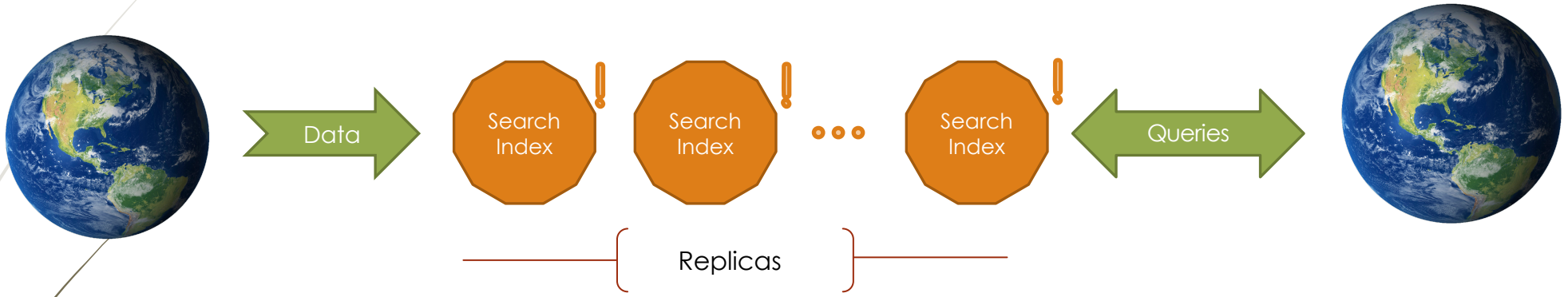


Scaling for High Data Volume



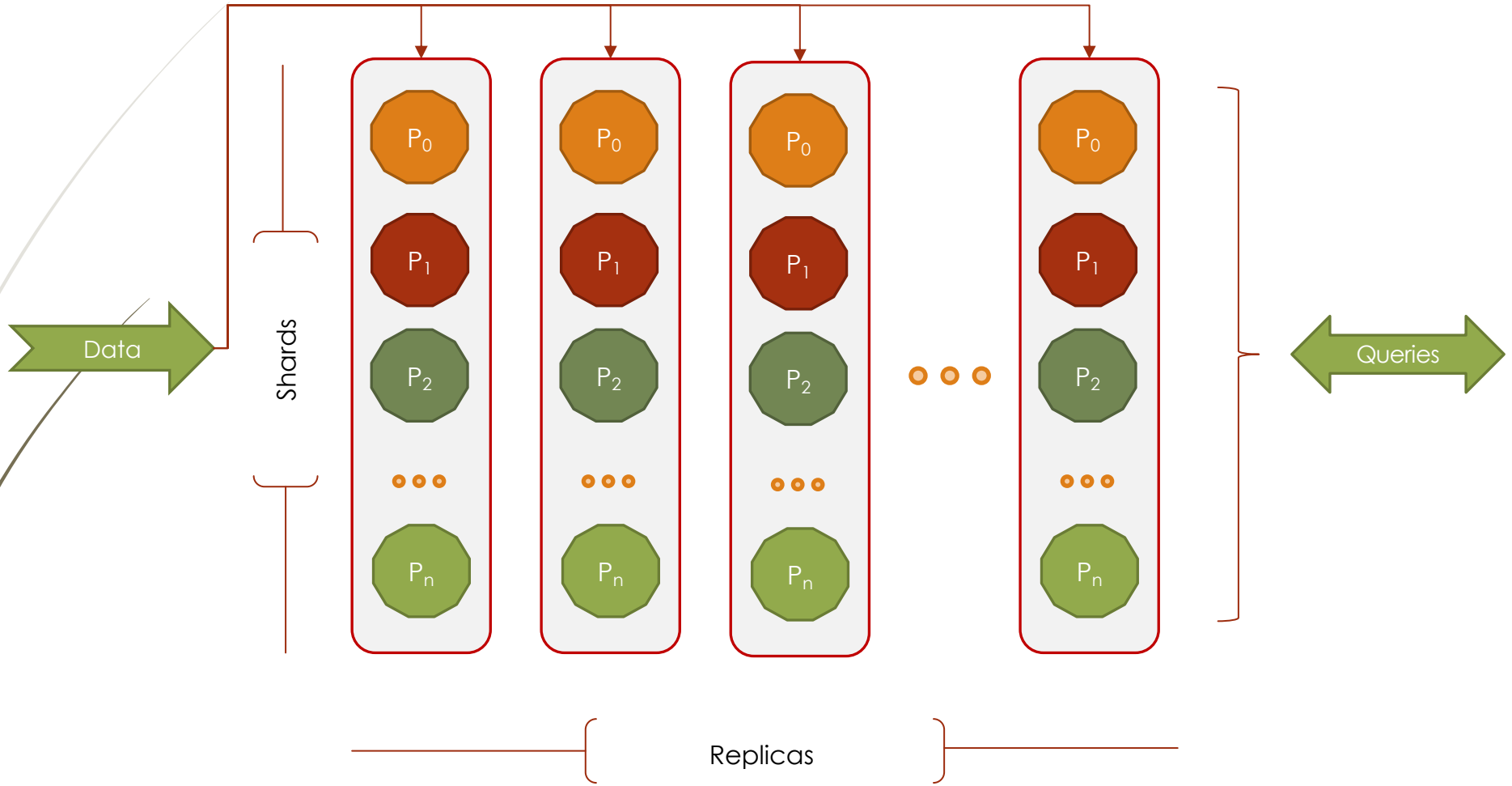
- Add replicas – multiple copies of the same index
- High Search Throughput
- Low Latency – each replica has the entire index

Scaling for High Data Volume



- Add replicas – multiple copies of the same index
- High Search Throughput
- Low Latency – each replica has the entire index

Shards and Replicas



Thinking about Scaling

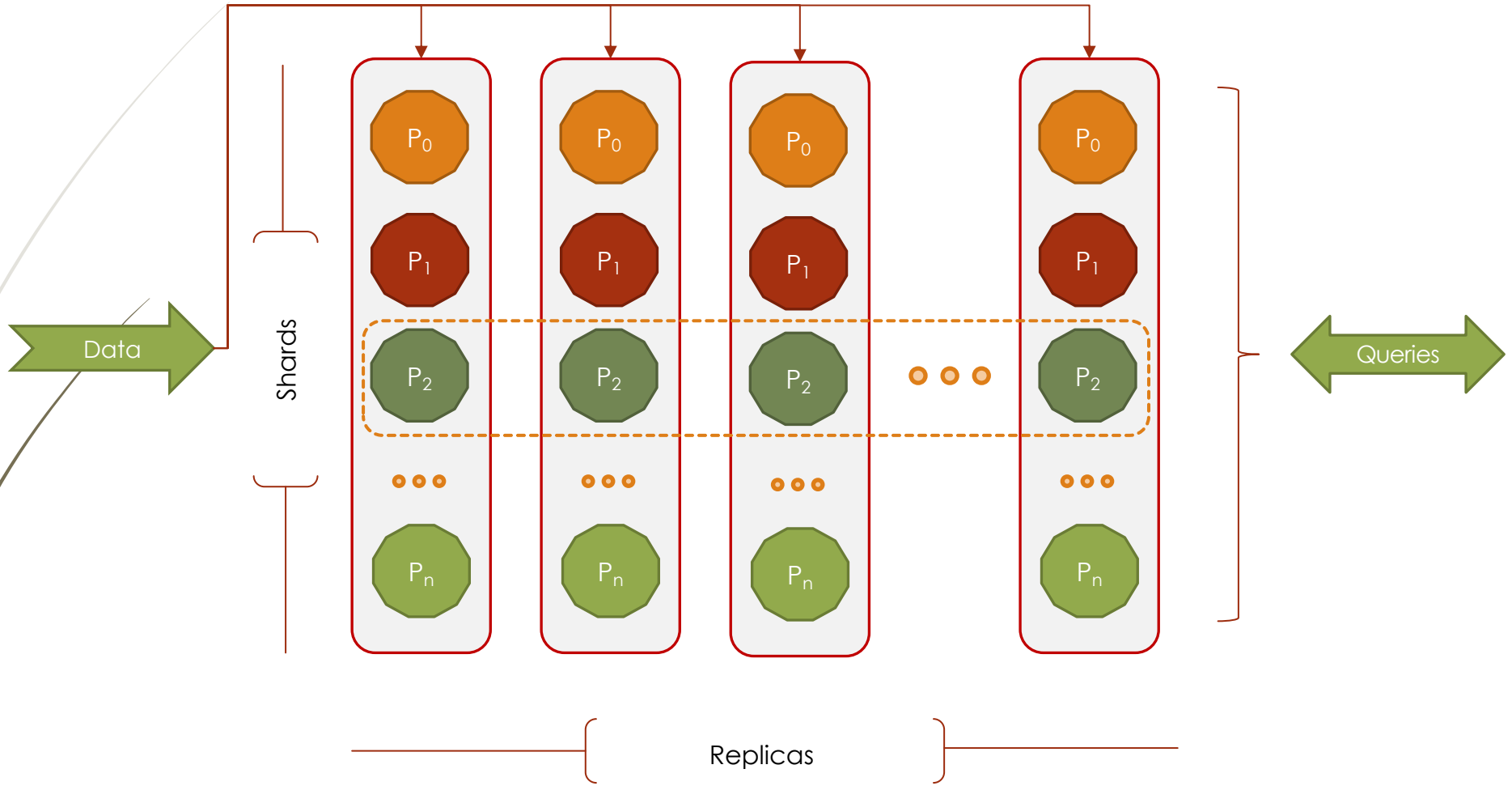
Search

- Search Traffic Throughput
- Tail Latencies
- Matching / Scoring / Inference complexity
- Collation Overhead and Cost

Indexing

- Ingestion Rate
- Data Pipeline Capacity
- Preprocessing Requirements
- Tolerance for staleness

Tight Coupling



The Problem at Amazon Search

- High query throughput
- High indexing rate
- Near real time updates

- To scale indexing by 3x –
  - instead of a 7% increase in h/w, we needed a 200% increase!

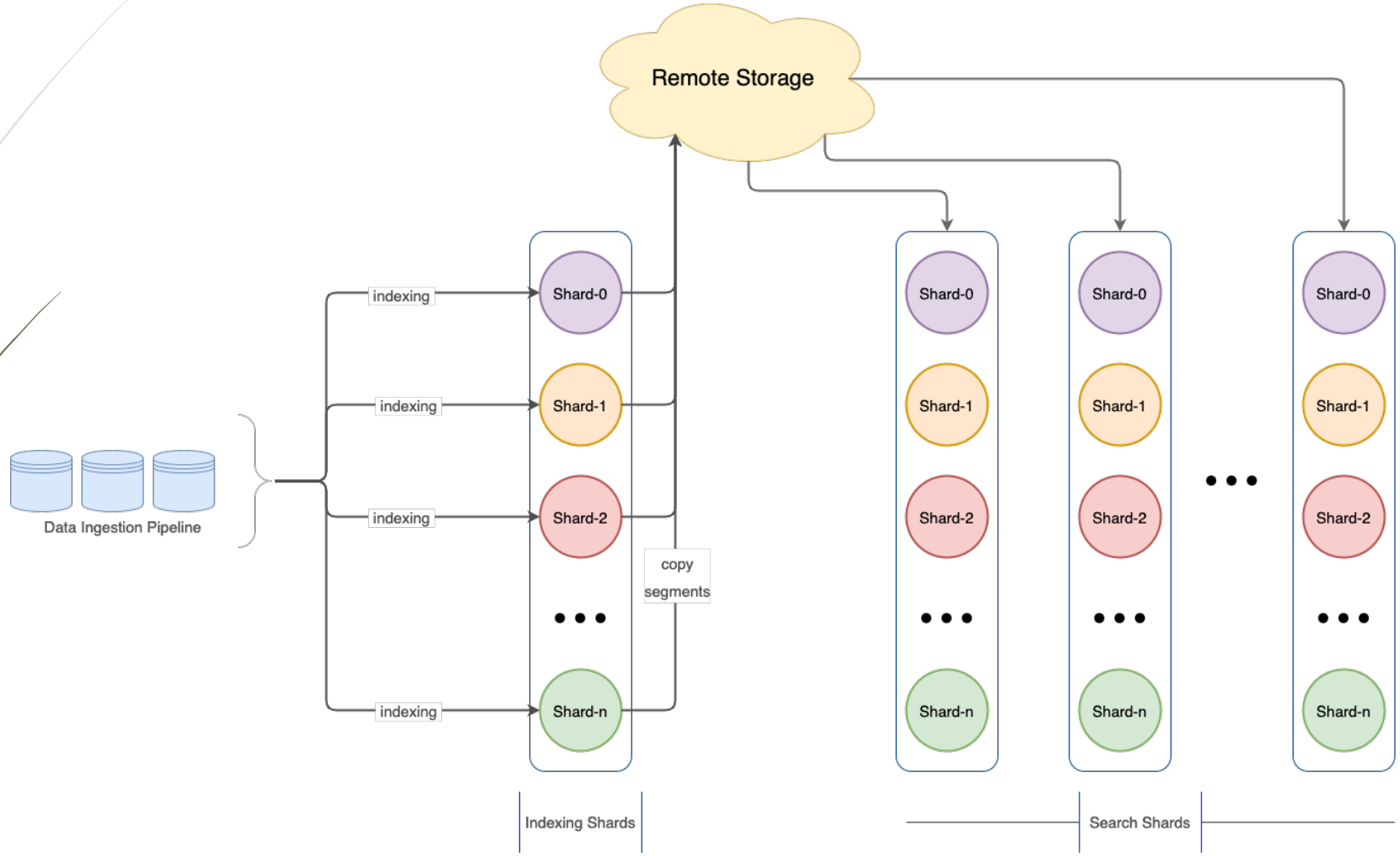
Decoupling Indexing and Search

With Near Real Time Updates

The Recipe

- 1/ Use Segment Replication
- 2/ Separate indexing and search shards
- 3/ Recombine shards after indexing to suit search width.

Segment Replication

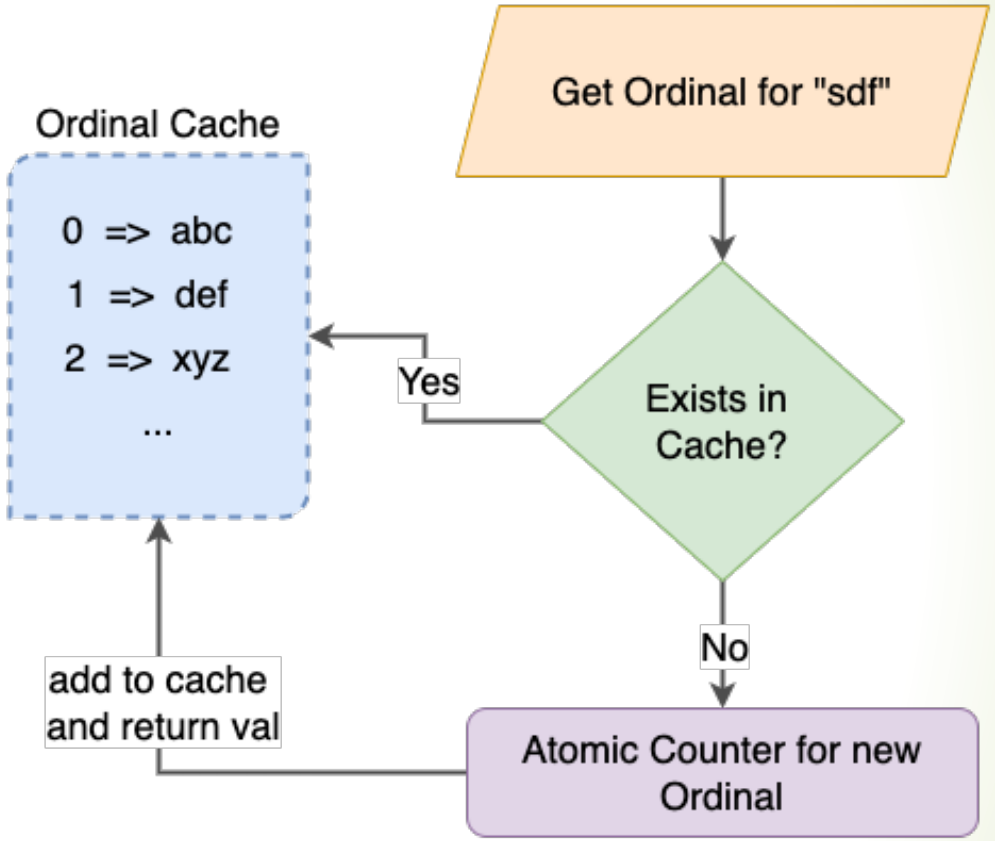


Combine Indexes for Search

- IndexWriter#addIndexes (Directory…)
  - \+ Fast – No need to reindex
  - \+ Transactional
  - \+ Copies segment files from provided directories
  - – Cannot modify incoming segments
  - – Append only. Does not apply previous deletes.

Ordinals in Lucene

- Unique Identifiers
- <Integer Ordinal> → <String or Binary Value>
- Helps count and refer to unique values



Taxonomy Ordinals in Lucene

Deletes / Updates in Lucene

- High performance, lock free, per thread delete queue
- Updates are buffered and applied at flush
- Live docs – bitset for docs still alive
- At flush,
  - Buffered updates applied to previous commits
  - Live docs updated to reflect previous generation deletes

Cross Shard Deletes / Updates

- To apply deletes to old segments, you need access to old segments.

- Single shard: no problem.
- Works for segment replication for a single shard –
  - As long as replica doesn't do merges, live docs can be replicated.

- Cross Shards: Segment geometry has changed!
- Live docs are no longer valid.
- Need to handle it manually

Carrying deletes forward

- At Amazon Search, deletes are on a primary key
- Create special marker doc for deleted documents.
- Can reapply deletes when combining index.

- Future Work?
  - Carry forward frozen deletes for the general case?

Modifying the incoming indexes

- IndexWriter#addIndexes (CodecReader…)

1. Create FilteredLeafReaders on incoming segments
2. Apply transformations like ordinal remapping
3. Process any pending deletes
4. Wrap as a CodecReader and add to index

“addIndexes”: Under the Hood

- Transactional API to add readers into the index.
- Readers are only “views” on actual index segments
- Need to be written down as segments

- Merges all readers into a single segment… in a single thread.
- Blocking
- Slow
- High add to search latency.

“

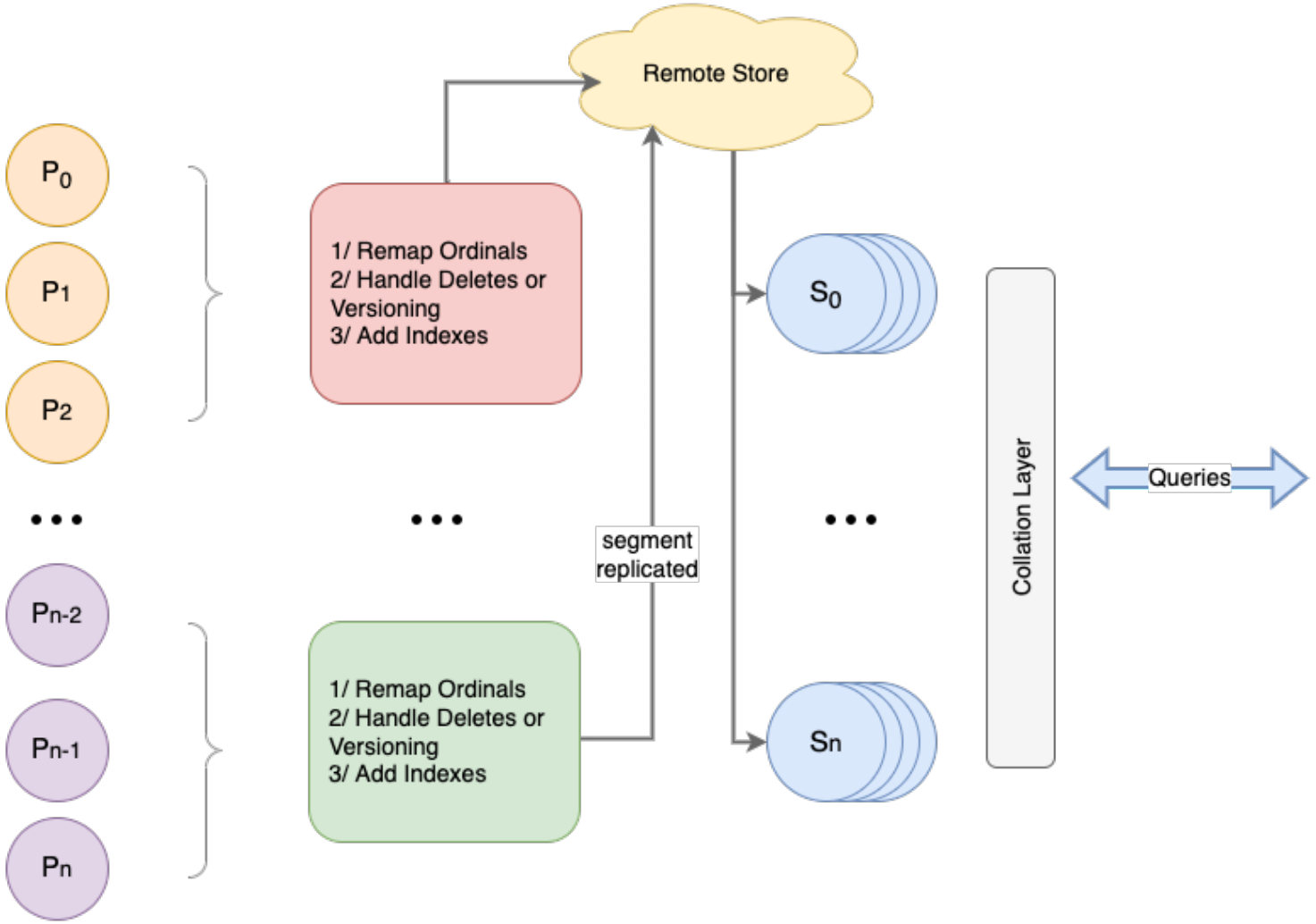
But we want near real time updates !

”

Changes in Lucene – 9.4

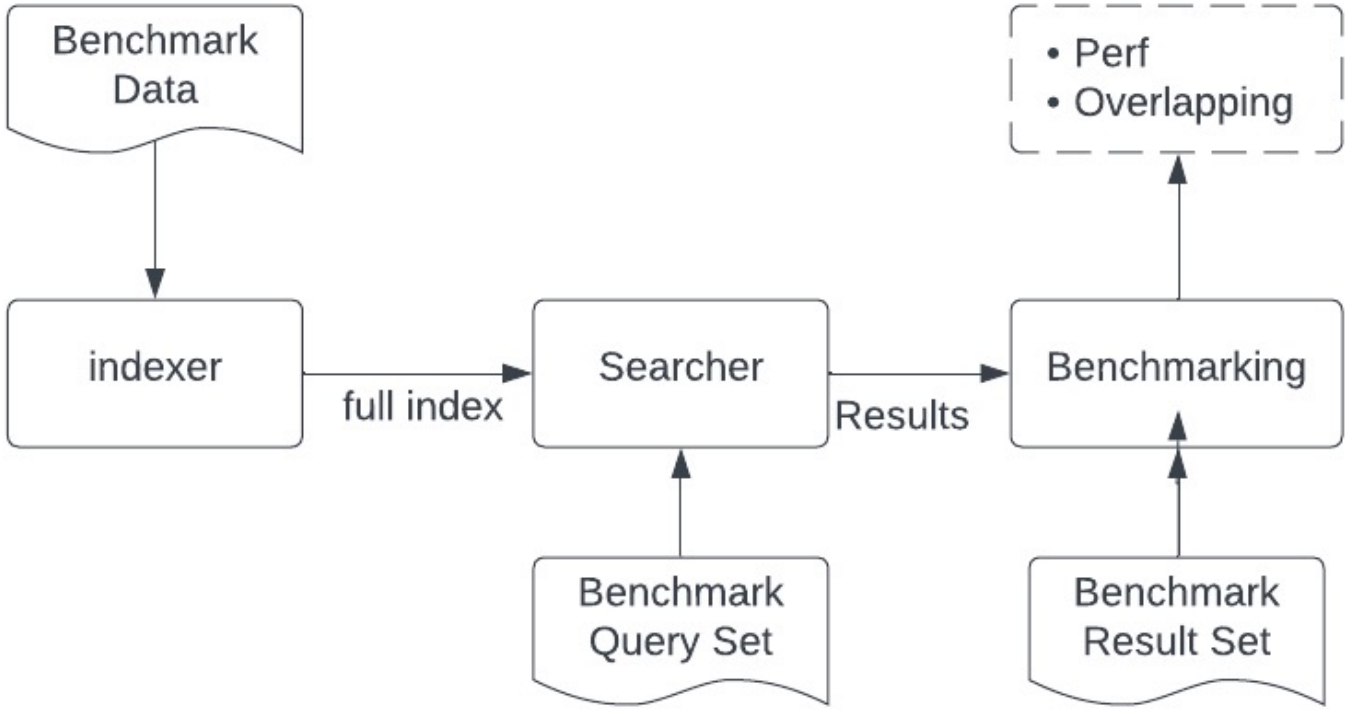
- addIndexes (CodecReader…) is now concurrent
- One thread per segment
- Configurable via MergePolicy#findMerges()
- Still Transactional
- Eventually merged in background using concurrent merge scheduler
- Low add-document to search latency

Bringing it all together

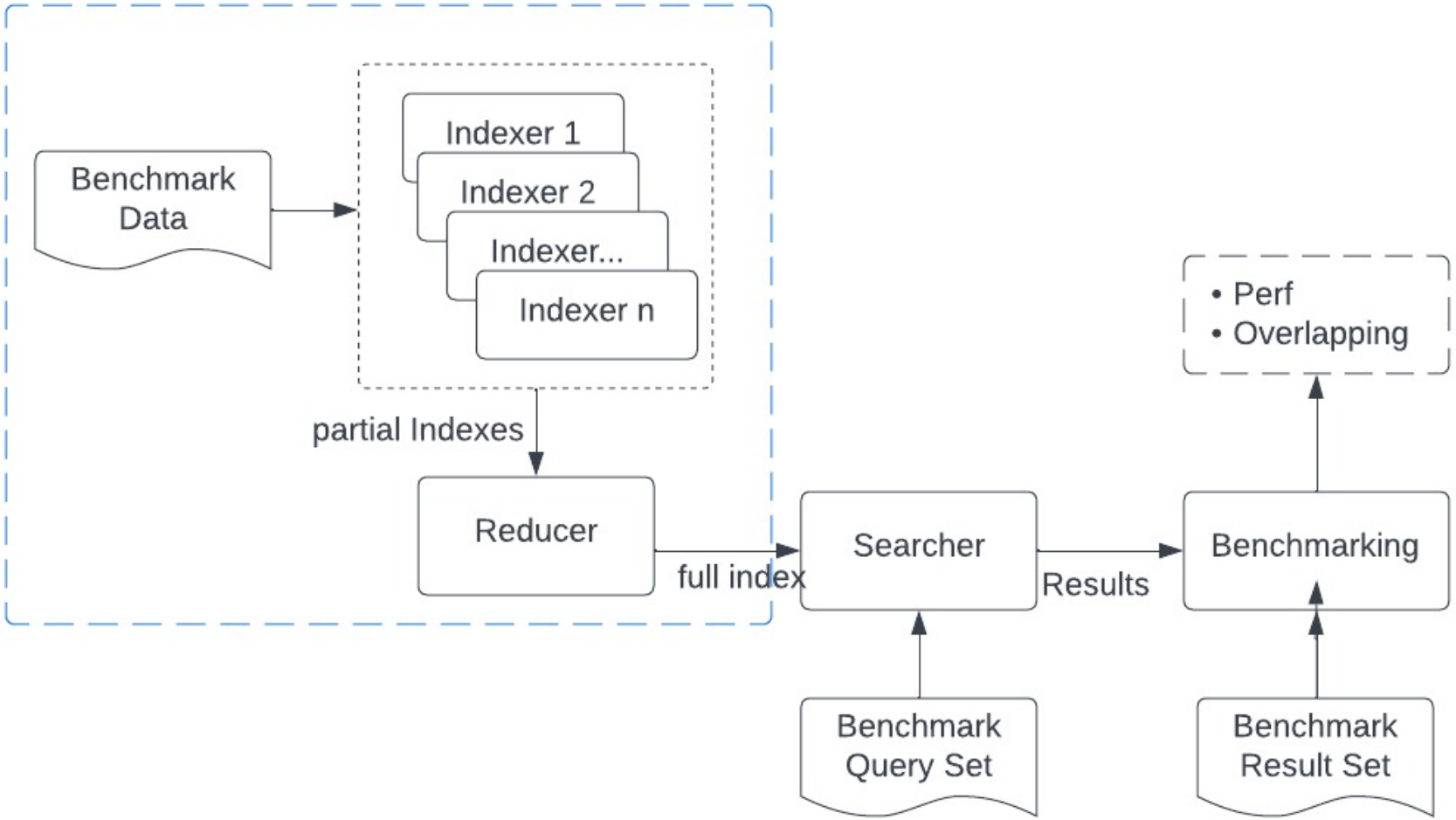


Test Architecture

Overlap Testing Architecture

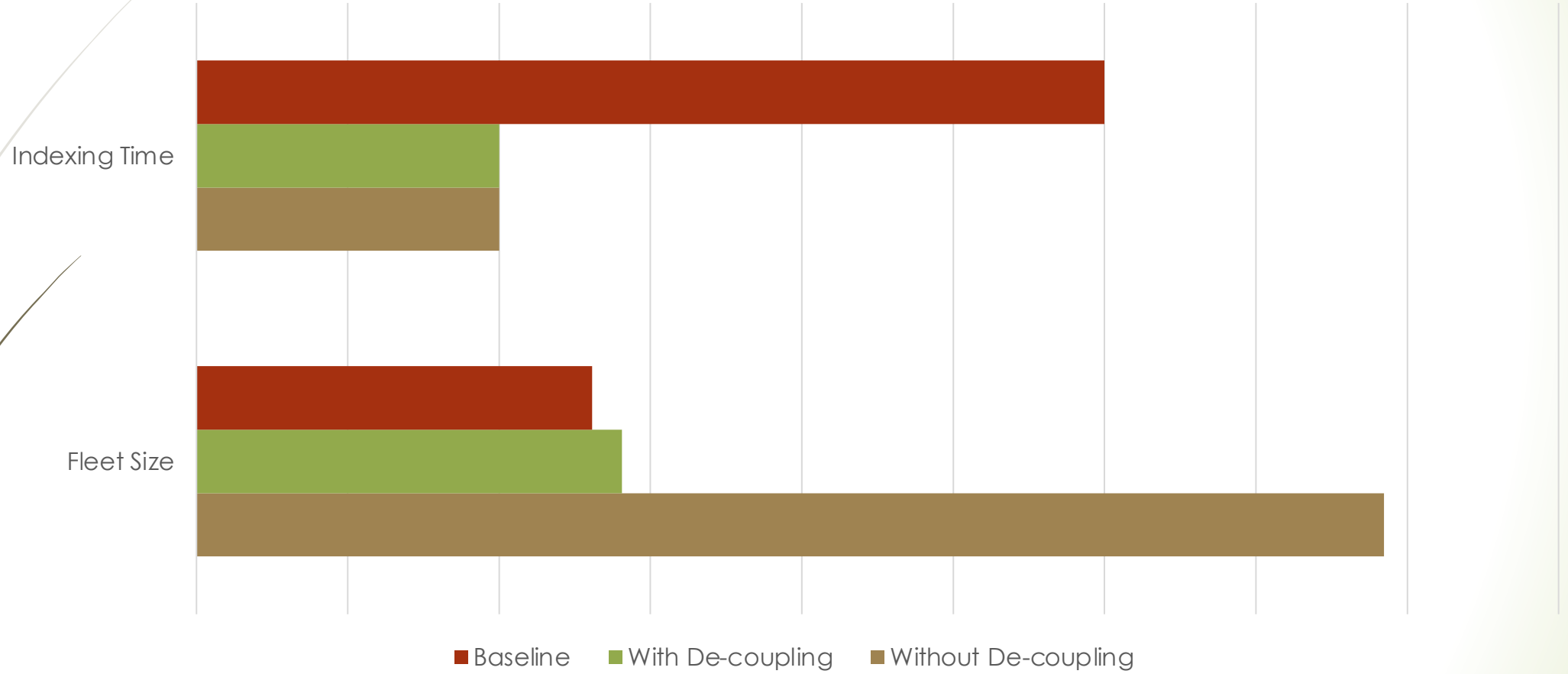


Extended setup for decoupled indexing / search

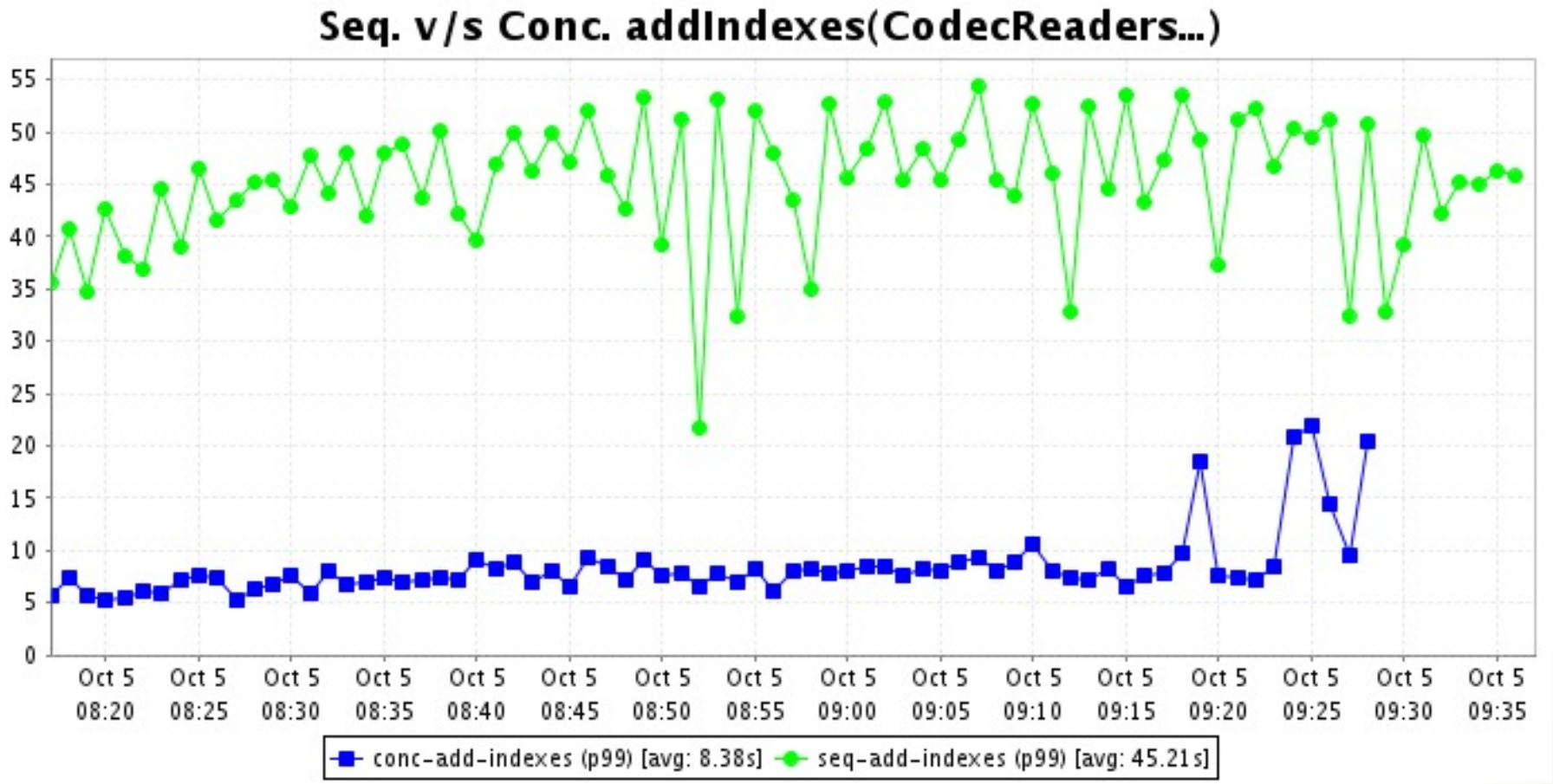


Early Performance Numbers

Indexing Time v/s Hardware Needed



Seq. v/s Conc. Add Indexes



Applications

- Unlock your indexing layer to do more
- Narrower search fleet for better cost and performance
- Fewer offline indexing processes
- Scale data intensive near real time applications –
  - Log analytics
  - Incident Response
  - Generating metrics and aggregate dashboards

Thank you… Questions?