Workflow for web archive indexing and search using limited resources

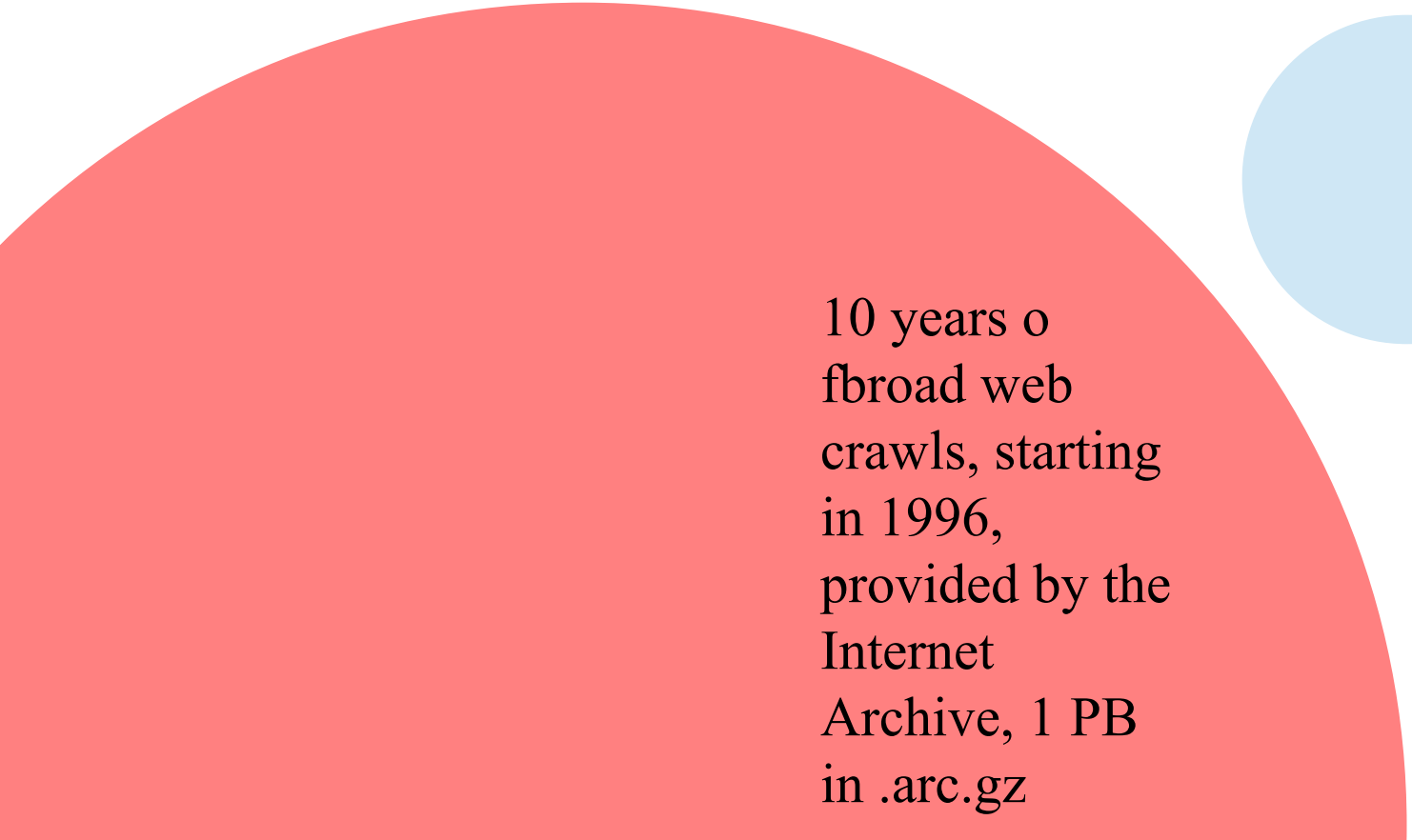
Sara Elshobaky & Youssef Eldakar

BIBLIOTHECA ALEXANDRINA
مكتبة الإسكندرية

BA web archive

- IA collection
  - 1996-2006
  - 1 PB
  - ARC files
- Egyptian collection
  - 2011+
  - 20 TB
  - WARC files

BA web archive



Focused web crawls, starting in 2011, harvested by the BA, 20 TB in .warc.gz

Problem

- Access is via URL only
- No keyword search
- Probably most wanted feature on users' wishlist since the BA ever had a web archive

Challenge

- Scale: > 1 PB compressed
- Old, low-compute storage hardware
- Need powerful parsing/indexing machines
- Need multi-node environment for hosting the final index

Related work

- UK Web Archive
  - Web Archive Discovery
    https://github.com/ukwa/webarchive-discovery
  - IIPC Technical Training Workshop 2015
    “Introduction to Apache Solr” - Andrew Jackson
- NetArchive.dk
  - IIPC Technical Traiining Workshop 2015
    “Scaling NetArchive Indexing & Search” - Toke Eskildsen

Proposed system

- Storage Cluster
  - Data being migrated to newer hardware, with GlusterFS as storage layer
  - Register unindexed files as they appear on a node
- Index Builders
  - Run on BA’s older High-Performance Computing (HPC) cluster
  - Integrate with the HPC workload manager
- SolrCloud
  - Recycle nodes from an older batch of storage hardware
  - Compensate for modest processing power and RAM with numbers
- Tracking DB
  - Use DB to maintain state information

1: Storage Cluster

- Migrate 1 PB from very low-compute machines to newer GlusterFS storage
- A lot of network traffic
- GlusterFS: 80 nodes, 4x 3-TB drives each
- GlusterFS highlights:
  - No metadata nodes
  - Easy to deploy and manage
  - Files not scrambled on disk
- 1 parser process per drive:
  - Watches for unindexed ARC/WARC files to appear under the mountpoint
  - Runs warc-indexer (UK webarchive-discovery) to extract text
  - Dispatch extracted text to the Index Builders

2: Index Builders

- Run on BA-HPC C1 (older HPC cluster):
  - 130 nodes
  - 8-GB RAM/node
  - 18-TB shared storage
  - 2 “fat” nodes, 64-GB RAM each
- Allocate 10 nodes for indexing:
  - 1 Solr indexer process per node
  - 1-TB maximum index size
- Use fat nodes for post-processing and optimization, 3-TB scratch space
- Monitoring process on each node logs state information in DB

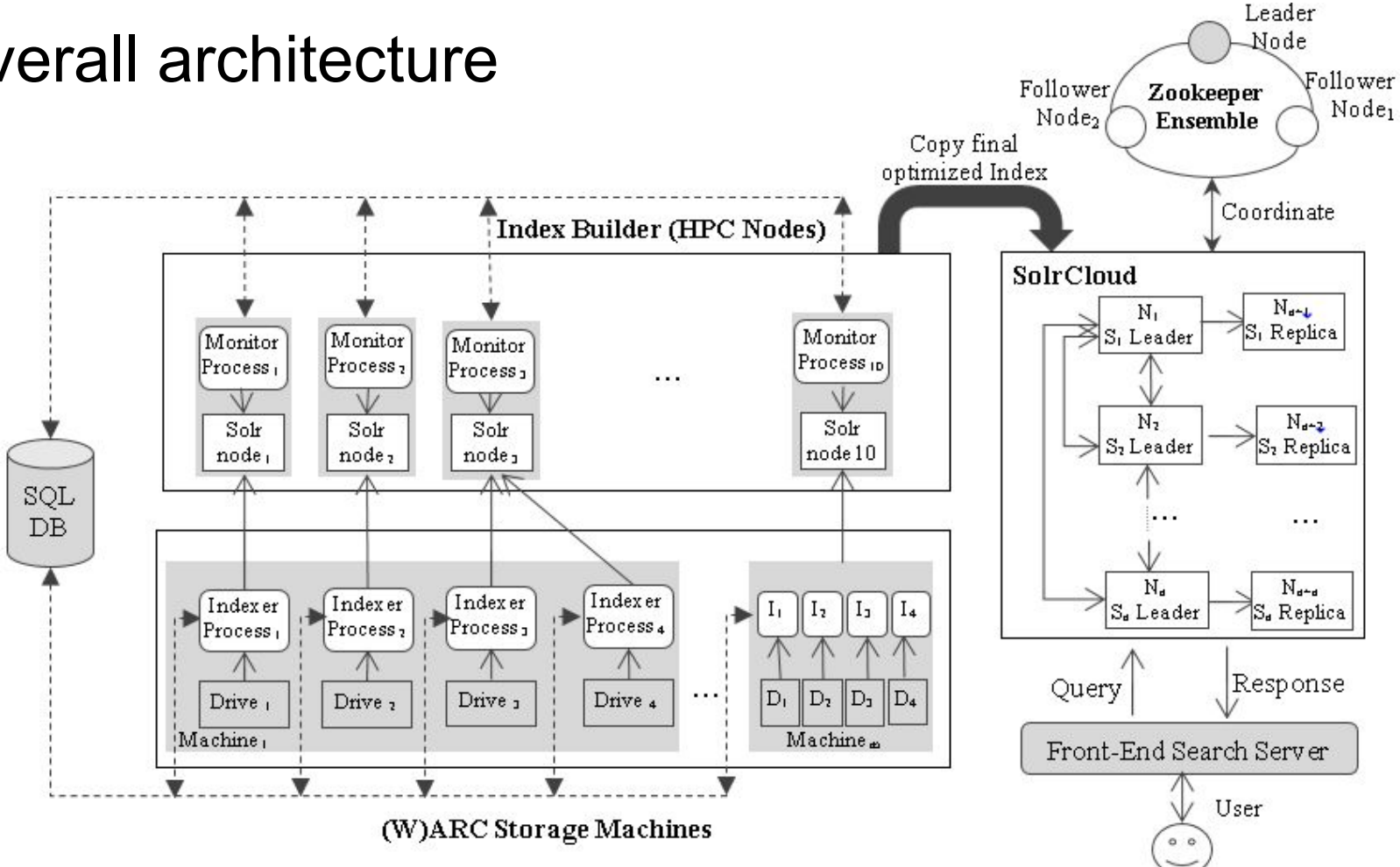
3: SolrCloud

- Estimating 100 TB of index per 1 PB
- Divide the index into 100 shards, 1-TB each
- Re-assemble 100 nodes from scrap parts
  - 2-GB RAM/node
  - Intel Core2 Duo
  - Each node to serve a 1-TB index shard
  - Each shard has a unique ID
- ZooKeeper ensemble:
  - Handles naming, configuration, and synchronization
  - Each node is a leader, can be replicated given hardware exists

4: Tracking DB

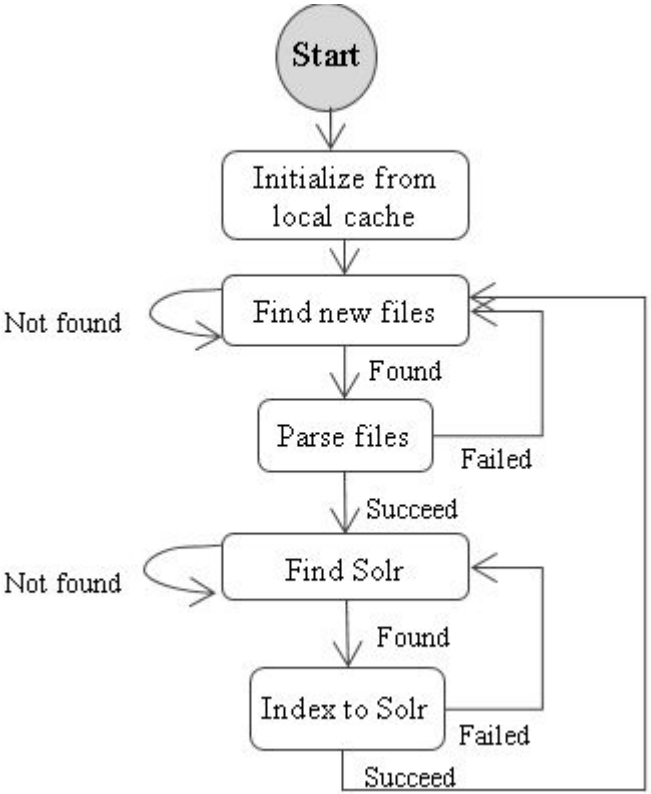
- Coordination of the workflow is relies on information stored in a MySQL DB
- Holds state information related to:
  - Data drives and data files in the Storage Cluster
  - Index Builder jobs on the HPC
  - Generated Solr index shards and where they exist in the SolrCloud
- Logs changes of state
- Maintains record of the set of ARC/WARC files that make up an index shard

Overall architecture



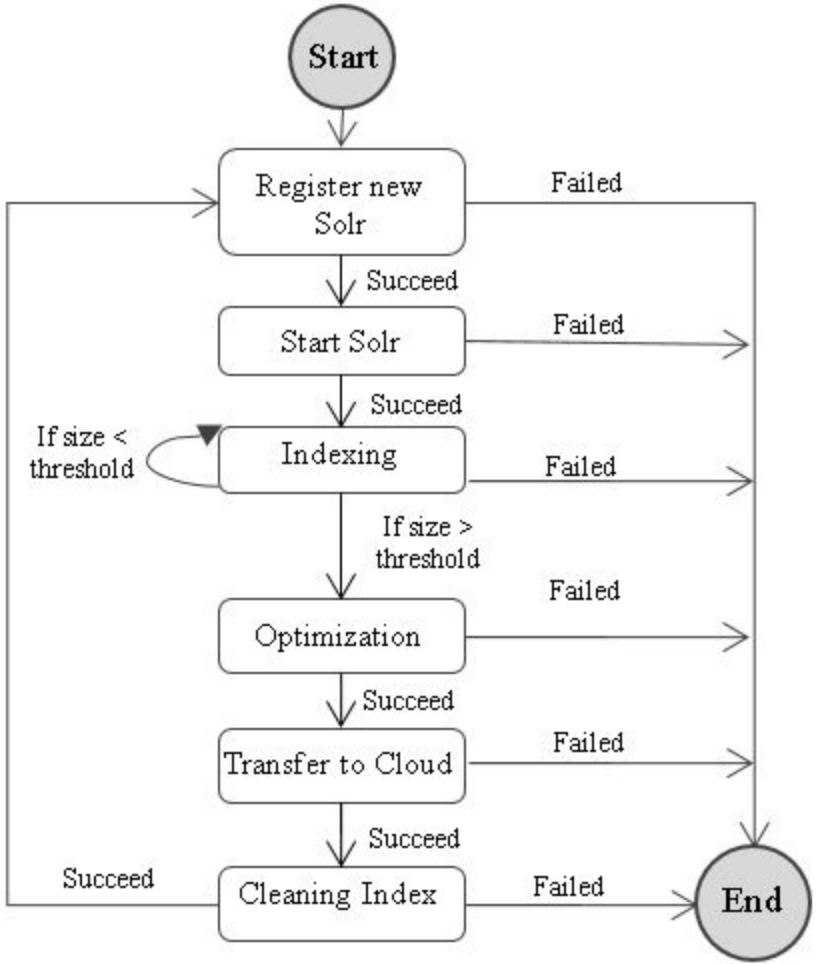
Workflow

- Wait for unindexedfiles to appearon a drive:
  - Extract text using warc-indexer
  - Find least loaded Solrprocess among Index Builders and dispatch extracted text to it
- If indexing fails, mark Solrprocess inactive and try another
- Maintain a list of indexed files in DB to ensure a file is not processed twice (maintain local cache per drive to reduce load on DB)



Monitoring Solr indexing jobs

- Each of the Index Builders on the HPC is driven by a monitoring process that:
  - Registers the new index shard, assigning it a unique ID in the DB
  - Starts indexing, regularly polling to ensure indexing process did not die out and shard size did not exceed the set limit
- If Solr not responding, mark job as failed in DB (for operator to examine)
- If shard size limit exceeded:
  - Start optimization on a fat node
  - Dispatch final index shard to SolrCloud
  - Clean up



Limited resources accommodation in shard nodes

Limit index size

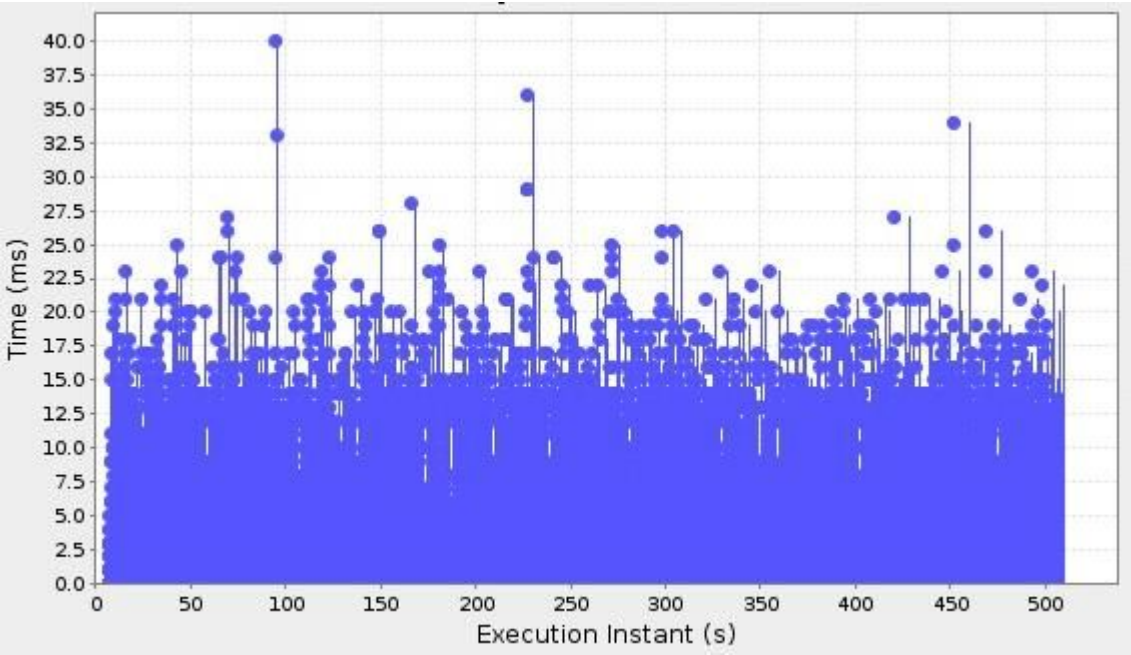
- Index page content but do not store it:
  - Decreases index size by 60%
  - Downside: no snippets, no highlighting
- Remove some fields from the schema (schema initially based on the UKWA's)
- Do not use copy fields (collections of other fields)

Limit Solr memory usage

- No faceting, suggestions, auto-completion
- No sorting by string, only by relevance or date
- Return only the first 1,000 matches (same as Google)

Stress test with limited memory

- Using SolrMeter to benchmark a single Solr instance
- 700-GB index shard
- 1-GB RAM dedicated to Solr
- 500 queries/second
- Results are returned in less than 40 milliseconds



Conclusion

- We designed an automated workflow to enable full-text search of large-scale web archive content
- The proposed system makes use of the limited hardware resources already at our disposal at the Bibliotheca Alexandrina
- The current result is a usable full-text search interface with a single index shard covering the Egyptian collection, with the set of search features tuned for low memory and storage
- We hope to later on report on results and lessons learned after indexing the full 1 PB using this workflow