Graduated Locality-Aware Scheduling for Search Engines

Jaimie Kelley and Christopher Stewart

The Big Idea

- For search engines, answering user queries is like... Pulling a fish out of a fish bowl
- Each user only wants a certain breed and color
- Don’t check every fish in the bowl. Sample
- Similarly search engines do not check every item in their data archives; they sample

Background: Search Engines

- Modern search engines construct an aggregator tree
- Aggregators create parallelism
- Processes intermediate results
- Acts as a per-query barrier
- May be nested
- Bottom level: Networked, low latency storage—such as, key-value stores that keep raw data

Front-End Interface (Response Time Limited)

Aggregator(s)

KV 1

KV N

Graduated Locality-Aware Scheduling

- Based on an assumption that the accesses are not ordered by usefulness—that is, a uniform distribution of usefulness exists over all requested keys.
- A stage \( s \) is a set of \( k \) keys with similar access times, loosely influenced by the underlying memory hierarchy.
- Keys with the fastest access times are accessed first, and only if more data is needed will the next stage be accessed.
- Using this method of scheduling, stages must be accessed in monotonically increasing order to keep the data filters as accurate as possible.
- Kneebends in the graph exist between L1 and L2, and more prominently between memory and secondary storage.

Naive Order-based Scheduling

- This method is based on the assumption that the order in which keys are requested matters.
- A stage \( s \) is a set of \( k \) keys accessed in order that they are produces by an aggregator, first the first \( k \) keys, next the second \( k \) keys if time will allow, etc.
- \( \text{keys available} = (k \ast s \text{ to } k \ast s + k) \)

Problem: Scheduling Partial Execution

- Networked storage can respond to data accesses in two ways
  1. With the actual data
  2. By eliding the data in favor of partial execution
- Challenge: Which accesses should be elided for low response time?
- Our preliminary study compared two competing approaches

Graduated Locality-Aware Scheduling

- The system was tested with OpenEphyra, a question-answering system like IBM Watson, to model the search engine application.
- The workload used a portion of the TREC 2007 QA main task data stored on Zookeeper nodes. Sample questions were taken randomly from this set of news documents. This is a sample workload that both OpenEphyra and IBM Watson have used as a benchmark for full execution, which made easy the decision to use it as a benchmark for partial execution in our research.

GLAS System on OpenEphyra

- GLAS checks the “confidence” score of intermediate answers
- Intercepts incoming key lookups. Returns only results in from stage
- Uses Bloom Filters to determine stage membership
- The system was tested with OpenEphyra, a question-answering system like IBM Watson, to model the search engine application.

Results for this benchmark were mixed, with a few high points but no clear case for locality-based scheduling.

- Future work is focusing more clearly on implications of the large increase in access time between memory and secondary storage for use in partial execution.
- Other future work will involve further modeling on the parameters of quality, response time boundary, and the number of each kind of node necessary to achieve these goals within a framework of partial execution.

Query: What city does Batman protect?

Front-End Interface (Response Time Limited)

Aggregator(s)

KV 1

KV N

Solution Checker

Open

GLAS

GLAS

Check out my demo!

Intermediate answer good enough? If no, graduate to next stage

Uses Bloom Filters to determine stage membership

Graduated Locality-Aware Scheduling

- Based on an assumption that the accesses are not ordered by usefulness—that is, a uniform distribution of usefulness exists over all requested keys.
- A stage \( s \) is a set of \( k \) keys with similar access times, loosely influenced by the underlying memory hierarchy.
- Keys with the fastest access times are accessed first, and only if more data is needed will the next stage be accessed.
- Using this method of scheduling, stages must be accessed in monotonically increasing order to keep the data filters as accurate as possible.
- Kneebends in the graph exist between L1 and L2, and more prominently between memory and secondary storage.

Time for Set Number of Accesses

- 3.3 billion accesses per test

Graph shows Time (s) vs. Array Size for different cache and memory access times.