Scalable In-Situ Exploration over Raw Data

Florin Rusu
University of California Merced
frusu@ucmerced.edu

Application. The Palomar Transient Factory[1] (PTF) project aims to identify and automatically classify transient astrophysical objects such as variable stars and supernovae in real-time. A list of candidates is extracted from the images taken by the telescope during a night. They are stored as a table in one or more FITS[2] files. The initial stage in the identification process is to execute a series of aggregate queries over the batch of extracted candidates. This corresponds to data exploration. The general `SQL` form of the queries is:

```
SELECT AGGREGATE(expression) AS agg
FROM candidate
WHERE predicate
HAVING agg < threshold
```

where `AGGREGATE` is `SUM`, `COUNT`, or `AVERAGE` and `threshold` is a verification parameter. These queries check certain statistical properties of the entire batch and are executed in sequence—a query is executed only if all the previous queries are satisfied. If the candidate batch passes the verification criteria, an in-depth analysis is performed for individual candidates. The entire process – verification and in-depth analysis – is executed by querying a PostgreSQL[3] database—only after the candidates are loaded from the original FITS files. This workflow is highly inefficient for two reasons. First, the verification cannot start until data are loaded. Second, if the batch does not pass the verification, both the time spent for loading and the storage used for data replication are wasted.

Problem. Our objective is to optimally execute exploration over raw data in a shared-memory multi-core environment where I/O operations are overlapped with extraction. In our concrete example, this corresponds to minimum execution time for the verification process—with minimal loading. Since data exploration does not have to be exact, as

---

[1] www.astro.caltech.edu/ptf/
[2] http://heasarc.gsfc.nasa.gov/fitsio/
[3] http://www.postgresql.org/

long as accurate estimates that guide the decision process are generated, its goal is achieved. However, if the detailed analysis is triggered, the work performed during exploration should allow for incremental extensions.

Parallel in-situ online aggregation on raw data. In-situ processing (RAW) has been proposed as a novel data exploration solution in many domains generating massive amounts of raw data since it provides immediate `SQL` querying over raw files. The performance of in-situ processing across a query workload is, however, limited by the speed of full scan, tokenizing, and parsing of the entire data. Online aggregation (OLA) has been introduced as an efficient method for data exploration that identifies uninteresting patterns faster by continuously estimating the result of a computation during the actual processing—the computation can be stopped as early as the estimate is accurate enough to be deemed uninteresting. However, existing OLA solutions have a high upfront cost of randomly shuffling and/or sampling the data.

Our approach is to seamlessly integrate online aggregation into raw data processing such that we cumulate their benefits. To this end, we introduce OLA-RAW, a bi-level sampling scheme for parallel online aggregation over raw data. Similar to RAW, OLA-RAW distributes data loading across the query workload. Notice, though, that loading in RAW – and, by extension, in OLA-RAW – corresponds to caching data in memory, not necessarily materializing on secondary storage. The same idea is extended to shuffling. Instead of randomly permuting all the data before performing online aggregation, OLA-RAW partitions shuffling across the queries in the workload. Moreover, loading and shuffling are combined incrementally such that loaded data do not require further shuffling. Sampling in OLA-RAW is query-driven and performed exclusively in-situ during the runtime query execution, without data reorganization. This is realized by a novel resource-aware bi-level sampling algorithm that processes data in random chunks concurrently and determines adaptively the number of sampled tuples inside a chunk. In order to avoid the cost of repetitive conversion from raw data, OLA-RAW builds and maintains incrementally a memory-resident bi-level sample synopsis. Essentially, OLA-RAW provides a resource-aware parallel mechanism to adaptively extract and incrementally maintain samples from raw data. Since OLA-RAW chooses the sampling plan that minimizes the execution time and guarantees the required accuracy for each query in a given workload, the end result is a focused data exploration process that avoids unnecessary work and discards uninteresting data.