HDF5-FastQuery: An API for Simplifying Access to Data Storage, Retrieval, Indexing and Querying

E. Wes Bethel*, Luke Gosink, John Shalf, Kurt Stockinger, Kesheng Wu
Lawrence Berkeley National Laboratory

Summary

This work focuses on research and development activities that bridge a gap between fundamental data management technology – index, query, storage and retrieval – and use of such technology in computational and computer science algorithms and applications. The work has resulted in a streamlined applications programming interface (API) that simplifies data storage and retrieval using the HDF5 data I/O library, and eases use of the FastBit compressed bitmap indexing software for data indexing/querying. The API, which we call HDF5-FastQuery, will have broad applications in domain sciences as well as associated data analysis and visualization applications.

Large-scale scientific experiments often store data in scientific data formats such as FITS, netCDF and HDF. These data formats provide the ability to store and retrieve multi-dimensional arrays that are often regarded as the building blocks for scientific data exploration. The most recent implementations of these data formats, like HDF5 and parallelNetCDF, have been extended to support parallel data access – a key requirement for data output from simulation codes on massively parallel computing platforms. However, one of the open problems common to all scientific data formats is that they to not have an interface to support semantic indexing. As pointed out by Jim Gray, "scientists need a way to use intelligent indices and data organizations to subset the search." Our work aims to bridge the gap between fundamental Computer Science technologies and their application in production scientific research tools.

Our recent work addresses this fundamental open problem of scientific data formats by providing an interface to support semantic indexing for HDF5 via a simplified query applications programming interface (API). We integrate an efficient searching technology named FastBit with HDF5. The integrated system, named HDF5-FastQuery, allows users to quickly generate potentially complex selections on HDF5 datasets using compound range queries like (*enery > 105*) AND (*70 < pressure < 90*), then retrieves only the subset of data elements that meet the query conditions. LBNL's patented FastBit technology generates compressed bitmap indices that accelerate searches on HDF5 datasets and can be stored together with those datasets in an HDF5 file. Compared with other indexing schemes, compressed bitmap indices are compact and very well suited for searching over multi-dimensional data – even for arbitrarily complex combinations of range conditions.

* (510) 486-7353, ewbethel@lbl.gov

HDF5 supports slab and hyper-slab selections from N-dimensional datasets. HDF5-FastQuery extends the HDF5 selection mechanism to allow arbitrary range conditions on the data values contained in the datasets using the bitmap indices. This allows the HDF5-FastQuery technology to support a fast execution of results for compound queries that span multiple datasets. The API also allows us to seamlessly integrate the FastBit query mechanism for data selection with HDF5's standard hyper-slab selection mechanism. Using the HDF5-FastQuery API, one can quickly select subsets of data from a HDF5 file using text-string queries. Our early performance results show the FastQuery enhancements to HDF5 for processing multi-dimensional queries exhibits consistently better performance over what is possible using only HDF5 and tree-based techniques for fast access to data stored in HDF5 files.

Figure 1 shows the results of one such performance experiment. The objective is to find all data records that match a single criterion. Using only HDF5, a sequential scan of all data records is required since HDF5 provides no intrinsic indexing capability. The result is constant performance proportional to the number of data records. Next, we benchmark an implementation of an R*-tree[1] for use with HDF5 files. The time required to answer queries grows in proportion to the number of data records located by the query and eventually exceeds the time required by the sequential-scan algorithm. Finally, the performance of HDF5-FastQuery increases up to a limit where half the records are returned, then decreases as more and more records are located by the query. This type of performance is characteristic of a specific type of bitmap indexing.

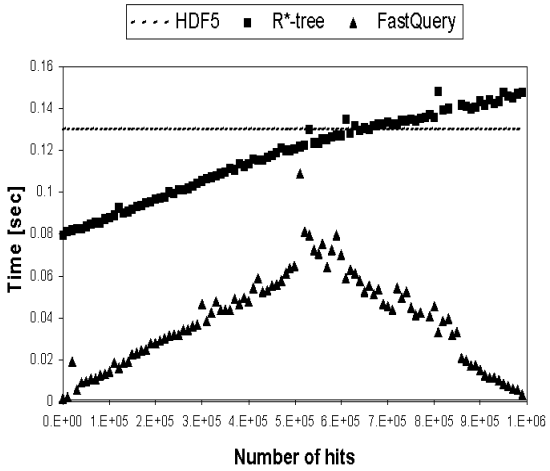

Figure 1. Comparison of query response time of a one-dimensional query with HDF5, HDF5-R*-tree and HDF5-FastQuery.

Our preliminary work shows promise in terms of exceptional performance characteristics and flexibility to its application in many different scientific domains. Future work will include expanding the HDF5-FastQuery API by applying it for use on science stakeholder projects in combustion, fusion and accelerator modeling; extending the API to accommodate multi-grid forms of domains (starting with adaptive mesh refinement problems); and beginning to adapt it for use in visualization and analysis applications. As the technology matures, we will make it available to the broader scientific community through an Open Source licensing mechanism.

For further information on this subject contact:
Name: E. Wes Bethel.
Organization: Lawrence Berkeley National Laboratory.
Email: ewbethel@lbl.gov
Phone: (510) 486-7353

Note: This work is a joint effort by the Visualization and Scientific Data Management programs at Lawrence Berkeley National Laboratory.

[1] An "R-tree" is essentially a multi-dimensional "B-tree" but branching takes into account both spatial as well as data ranges. An R*-tree offers optimizations to the R-tree that accelerate insertions.