# GRID COLLECTOR: USING AN EVENT CATALOG TO SPEED UP USER ANALYSIS IN DISTRIBUTED ENVIRONMENT

Kesheng Wu, Arie Shoshani, *Lawrence Berkeley National Lab*, Wei-Ming Zhang, *Kent State University*, Jerome Lauret, Victor Perevoztchikov, *Brookhaven National Lab*

*Abstract*

Nuclear and High Energy Physics experiments such as STAR at BNL are generating millions of files with PetaBytes of data each year. In most cases, analysis programs have to read all events in a file in order to find the interesting ones. Since the interesting events may be a small fraction of events in the file, a significant portion of the computer time is wasted on reading the unwanted events. To address this issue, we developed a software system called Grid Collector. The core of Grid Collector is an Event Catalog. This catalog can be efficiently searched with compressed bitmap indices. Tests show that Grid Collector can index and search STAR event data much faster than database systems. It is fully integrated with an existing analysis framework so that a minimal effort is required to use Grid Collector. In addition, by taking advantage of existing file catalogs, Storage Resource Managers (SRMs) and GridFTP, Grid Collector automatically downloads the needed files anywhere on the Grid without user intervention.

Grid Collector can significantly improve user productivity. For a user that typically performs computation on 50% of the events, using Grid Collector could reduce the turn around time by 30%. The improvement is more significant when searching for rare events, because only a small number of events with appropriate properties are read into memory and the necessary files are automatically located and downloaded through the best available route.

## INTRODUCTION

High-Energy and Nuclear physics experiments are producing a large amount of data. For example, the experiment STAR at Brookhaven National Laboratory is collecting millions of collision events and PetaBytes of data every year [1]. Analyzing these large volumes of data efficiently is a significant challenge.

In these datasets, the basic unit of data is a collision event. The events are usually collected into files and files are organized into directories according to experiment parameters, such as time, trigger setup and magnetic field scale. Because of the large data volume, a majority of the data is stored on mass storage (tape) systems. Because manually retrieving any significant number of files from mass storage system is labor-intensive, error-prone and time-consuming. These files are as inaccessible as files distributed many miles away. This leaves only a small amount of the data accessible for analyses. Many users work around this limitation by designing their analyses to use only data on disk. This limits analysis options and limits user productivity. One major objective of Grid Collector is to automate the access to data files on mass storage systems and distributed over the Grid. The first two success stories involving Grid Collector both make heavy use of this particular feature.

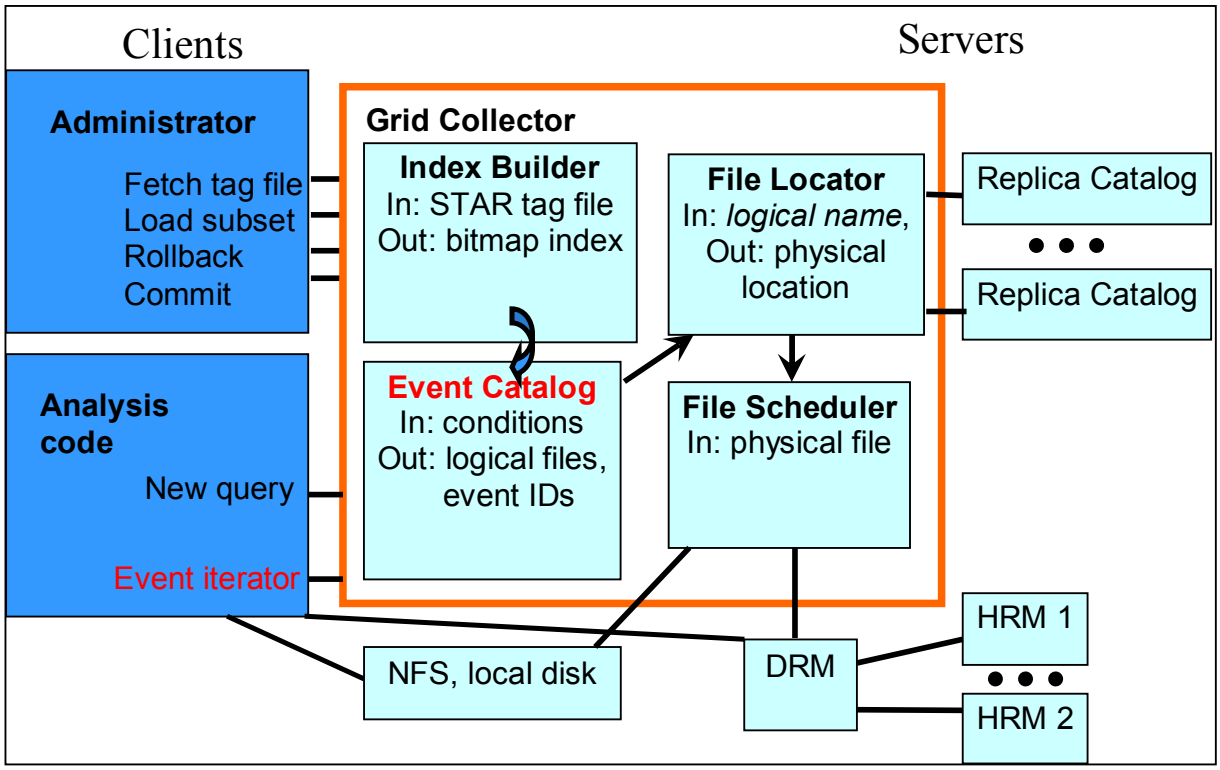
Most of analysis jobs select a modest subset of events and perform its computation only on these events of interest. Users usually specify the events of interest as a list of data files, a list of directories, or the name of a predefined subset. Properly managing numerous subsets is a very challenging problem, which is being addressed now in the research community. Existing analysis frameworks typically read all events in the named files and directories. The user analysis code is required to perform its own filtering to select the events of interest. On many computer systems, especially those with many users, the time to read events from disk is more than 95% of the total execution time. Clearly, allowing users to more precisely specify the events of interest and reading only those selected events would reduce the total execution time. This is the second major objective of Grid Collector.

The core of Grid Collector is an event catalog. Similar to a file catalog, an event catalog is a system of software and data that associates each collision event with a number of user-defined searchable attributes, such as number of particles produced by the collision, total charge of all particles, and number of primary tracks. In addition to those attributes, our event catalog also contains the necessary information to read the events from files. Since the event catalog is expected to contain billions of records, the efficiency of searching the catalog is potentially a concern. Through our research, we have identified a compressed bitmap indexing technique that is capable of processing common filtering conditions very efficiently. It allows us to answer common conditions in a few seconds or less, which is insignificant compared to the typical run time of analysis jobs.

Grid Collector builds an event catalog from existing files, without requiring all data to be placed under a single database management system (DBMS). It can be used with a minimal amount of additional space and it requires users to make only minimal changes to their analysis codes to utilize the new functionalities. Compared with the conventional approach of using DBMS, our approach

requires a minimal amount of change to the existing analysis framework, while providing significant new functionalities. This approach is well suited for on-going experiments and experiments that have already decided on their basic analysis frameworks. Future experiments may consider building an event catalog using similar underlying technology.

**Figure 1. Schematics of Grid Collector.**




## OVERVIEW OF GRID COLLECTOR

Grid Collector is a set of software modules that integrates an event catalog with an existing High-energy analysis framework and a number of Grid software packages to fulfil its design goals. It has two primary design goals, one to automate accesses to remote files and one to read only the selected events.

To retrieve any remote file, one has to reserve enough disk space for the file, monitor the file transfers, recover from any recoverable errors, retry an alternative source in case of an unrecoverable error, and finally, remove the file after use. If thousands of files were needed, these file management tasks would be very tedious and time consuming. A number of software packages have been designed to address some of these file management tasks. For example, most large experiments have their own file catalogs and replica catalogs to keep track of files and their copies distributed on different storage systems. These catalogs are important for automating the file accesses. A system that we use extensively is the Storage Resource Manager (SRM), which composes of a Disk Resource Manager (DRM) and a Hierarchical Resource Manager (HRM). These software packages can manage and retrieve files any where on the Grid including on mass storage systems. In addition, DRM can cache remote files and automatically reclaim the disk space after use. This can enable streamlined analysis of a large number of files that may not fit on disk at the same time.

To read only the events of interest, the event catalog contains information necessary to access any particular event in a data file. At this point, this feature uses run numbers and event numbers, and is only available for ROOT files [2]. Grid Collector works in a client-server model. The server side manages the files and a client library called the Event Iterator take the information from the server and reads the selected events. The events are passed to the analysis framework so that the user code can proceed as usual.

At the high level, Grid Collector has two sets of disjoint functions, one for administration of the event catalog and one for user analysis. The files needed for building the event catalog and those needed for user analyses are managed through DRMs. The user analysis code invokes the Event Iterator, which allows the user to access transparently the events of interest, without explicitly knowing the names or the locations of the files involved. The files used are not required to be under the management of any DRM. If a file is on a NFS system that is accessible to the client program or on a local file

system on the client machine, Grid Collector will instruct the Event Iterator to use the file. For a file that is not available locally, Grid Collector can contact multiple file catalogs and replica catalogs to determine an appropriate location to retrieve the file.

## EVENT CATALOG

A key component of Grid Collector is the event catalog. It holds searchable attributes of events, information about how to locate the file, and how to read the event within the file. With the information, one can specify conditions on the searchable attributes and get back information on how to retrieve the selected events. A number of experiments have previously explored the idea of building event catalogs; however, most have given up on this approach. An obvious way of building an event catalog is to put all the data into a commercial DBMS. However, this approach is expensive. Most freely available DBMS systems are not able to handle the large volume of data produced from a typical experiment. Frequently, the experiments use a free DBMS system to implement their file catalogs and replica catalogs. An event catalog would have 100 to 1000 times more records than the file catalogs. Because of their large sizes, a significant challenge in implementing an event catalog, commercial or otherwise, is how to search the catalog efficiently.

The general technique for speeding up searching of large datasets is indexing. Recently, we implemented an efficient compressed bitmap index for searching large datasets. Complexity analyses show that our compressed bitmap indices are in fact optimal for one-dimensional range queries. The time complexity of answering these one-dimensional range queries is a linear function of the number of hits [4]. Only few especially efficient indexing schemes, such as B+-tree, have this optimality. Since the results of one-dimensional queries can be efficiently combined to answer multi-dimensional queries, this optimality implies that compressed bitmap indices are also efficient for multi-dimensional range queries. The same is not true for B+-trees. Performance measurements on a variety of datasets demonstrated that the compressed bitmap indices are significantly more efficient than other indices not only on one-dimensional range queries but also on multi-dimensional range queries.

| | | B-tree | Projection | Bitmap |
|---|---|---|---|---|
| Size (MB) | | 408 | 113 | 186 |
| Query processing (seconds) | 1-dim | 0.95 | 0.51 | 0.02 |
| | 2-dim | 2.15 | 0.56 | 0.04 |
| | 5-dim | 2.23 | 0.67 | 0.17 |

**Figure 2. The sizes and average query processing time on a subset of STAR data with 2.2 million events and 12 commonly used attributes.**

The table in Figure 2 is a summary of the performance of three indexing schemes, a B-tree from a commercial DBMS, a projection index and our compressed bitmap index. In most applications, the projection index is the most efficient one for multi-dimensional range queries. The average query processing time reported in the table is an average over 1000 range queries with randomly generated range conditions. The same queries are answered with all three indexing scheme. A client program generates the queries and measure the query response time of the different indexing schemes.

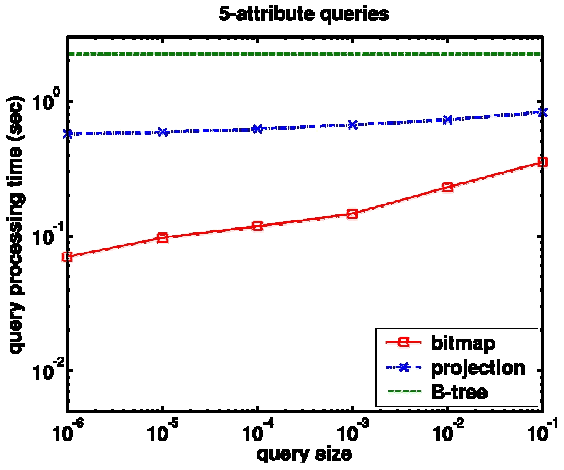



**Figure 3. The average time used to answer 5-dimensional queries with different indexing schemes.**

Figure 3 shows the query processing time using the same three indexing schemes on some 5-dimensional range queries. The horizontal axis in the plot is the query size, which is measured as the fraction of the domain of the attributes selected by the range conditions. For example, let the number of primary tracks have a domain of 0 to 10,000 and zdc1Energy have a domain of 0 to 200, then the size of query 'number of primary tracks > 5,000 and 10 < zdc1Energy < 20' is 0.025.

From Figure 2 and Figure 3, we see that the compressed bitmap indices are at least three times as fast as the projection indices, and are at least ten times as fast as the B-tree in the particular commercial DBMS. Though the particular dataset used in the performance test is relatively small, the relative performance differences are typical of many performance tests [4].

## GRID COLLECTOR PERFORMANCE

One of the main reasons for developing Grid Collector is to speed to the analysis of events of interest. In this section, we measure how it actually performs on a set of STAR data [1]. We selected three runs from a recent 62 GeV production. Without Grid Collector, users usually have to form events of interest by naming files or directories. In STAR, users may also use the Scheduler to form events of interest based on runs. Since this option is the closest to the option provided Grid Collector, we use the time required to read all events in a run as the base line for measuring the speedup of Grid Collector. Since the number of events in each run varies widely, we have chosen only to show the speedup numbers rather than actual time. Figure 4 shows the speedup measurements

on a Linux box with Xeon 2.8 GHz processor and 1 GB of main memory.

If a small fraction of events is selected, the speed up is large. For example, when one in 1000 events is select, the speedup can be from 20 to 50. Many analyses select 10% to 50% of events of a run. In such a case, Grid Collector also speeds up the analyses. If 50% of events are selected, the speedup is more than 1.5, and if 10% of events are selected, the speedup is more than two.

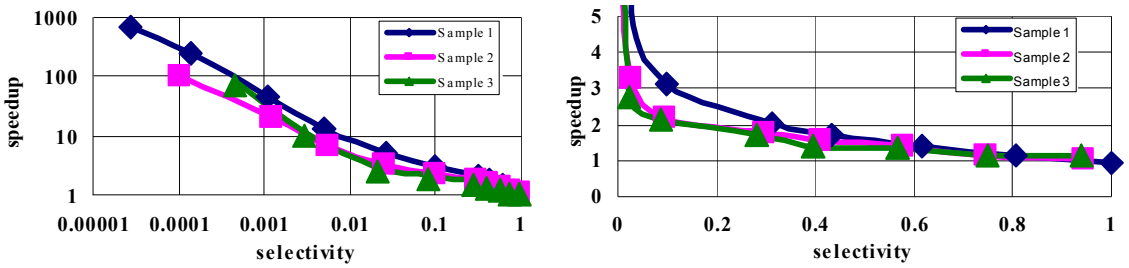



**Figure 4. Using Grid Collector can significantly speed up the reading of events of interest.**

Clearly, many factors can affect the actual speedup observed in a particular analysis. First among which is the organization of the files involved. In our tests, the data files are compressed ROOT files. When accessing a random event in a file, other events are read as well because the compression is applied on a group of events at a time. In this case, a redesign of the structure of the ROOT files could make it significantly easier to read a random event, and therefore make Grid Collector even more efficient.

## SUCCESS STORIES

Our first two users are two STAR users who have done some preliminary analyses on a large number of events to search for anti-$_3$He (Lee Barnby) and indications of stranglets (Aihong Tang). In both cases, the preliminary analyses have produced a few hundreds of interesting events. The preliminary analyses were performed on higher-level summary data files. The next step of analyses requires the event data rather than the summary data. Each event data file is much larger than the corresponding summary data file, and these event files are on the mass storage system at Brookhaven Lab. In addition, there is no easy way to identify the particular files that contain the interesting events. Most likely, the event files for all runs containing the events of interest will have to be retrieved from the mass storage system. For each analysis to continue, many thousand of large event files will have to be retrieved. It is hard for the users to find enough disk space to store all the files. In addition, the process of retrieving these files and manually recover from any error is very tedious and time consuming. For these reasons, the second step of the analyses was put on hold until Grid Collector become available. Using Grid Collector, the process of reading the events of interest and writing them out for further analyses took only a few hours. In these two cases, Grid Collector enabled user to perform analyses that was previously nearly impossible to do. Most other cases are not as dramatic as these two; however, the reduction in turn-around time for analyses is significant.

## SUMMARY

Grid Collector was designed to address two major needs in analyses of high-energy physics experiment data, namely the need to work with remote files transparently and the need to access only the events of interest. The key technology that enables Grid Collector to fulfil these needs is the event catalog and the ability to search for events of interest efficiently. Our event catalog uses an efficient bitmap index that is ten times faster than commonly used indexing techniques and at least three times faster than the best-known technique. Because the searching time is usually small, using Grid Collector to read any subset of events is aster than reading all events in the files involved. Grid Collector can significantly improve user productivities by making hard analyses practical and making routine analyses faster.

## ACKNOWLEDGEMENTS

This work was supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No.~DE-AC03-76SF00098. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

## REFERENCES

[1] http://www.star.bnl.gov/.
[2] http://root.cern.ch/.
[3] D. Zimmerman. The design and implementation of the STAR tag database, CHEP 98, 1998.
[4] K. Wu, E. Otoo and A. Shoshani. On the performance of high cardinality attributes. VLDB 2004.