# Grid Collector: Facilitating Efficient Selective Access from Data Grids

Kesheng Wu, Junmin Gu, Jerome Lauret, Arthur M. Poskanzer, Arie Shoshani, Alexander Sim, and Wei-Ming Zhang

**Abstract** — The Grid Collector is a system that facilitates the effective analysis and spontaneous exploration of scientific data. It combines an efficient indexing technology with a Grid file management technology to speed up common analysis jobs on high-energy physics data and to enable some previously impractical analysis jobs. To analyze a set of high-energy collision events, one typically specifies the files containing the events of interest, reads all the events in the files, and filters out unwanted ones. Since most analysis jobs filter out significant number of events, a considerable amount of time is wasted by reading the unwanted events. The Grid Collector removes this inefficiency by allowing users to specify more precisely what events are of interest and to read only the selected events. This speeds up most analysis jobs. In existing analysis frameworks, the responsibility of bringing files from tertiary storages or remote sites to local disks falls on the users. This forces most of analysis jobs to be performed at centralized computer facilities where commonly used files are kept on large shared file systems. The Grid Collector automates file management tasks and eliminates the labor-intensive manual file transfers. This makes it much easier to perform analyses that require data files on tertiary storages and remote sites. It also makes more computer resources available for analysis jobs since they are no longer bound to the centralized facilities.

**Key words** – Grid, Data Management, Data Analysis, Bitmap Index, Storage Resource Manager

## 1 INTRODUCTION

Modern scientific investigations require the searching over billions of objects and accessing data from various distributed storage systems. The lack of appropriate technology to address this problem has prohibited the effective analysis and spontaneous exploration of scientific data. In this paper, we describe the application of a new theoretically optimal bitmap indexing technology in combination with Grid storage management technology to overcome this data analysis problem. While the technology is generic and can be applied to various application domains, we focus in this paper on applying it to a particular dataset produced from a high-energy and nuclear physics (HENP) experiment called STAR [13]. This presented an additional challenge of incorporating these technologies into the existing analysis framework used by the experiment. This was achieved by making the Grid Collector a plug-in that boosts the performance and capability of STAR analysis framework.

Many on-going and planned HENP physics experiments are capable of producing PetaBytes of raw data per year [1] [2]. Generally, the bulk of the data is composed of snapshots of high-energy collisions, known as events. The recorded data, in its "raw" or digitalized format, usually goes through a reconstruction process to produce event summary data [4]. The event summary data is often further processed into analysis objects [4]. As the name suggests, most user analyses use these analysis objects rather than the raw data or the event summary data. Even though, the analysis objects are much smaller than the raw data, they can still be many terabytes in size [2]. The process of producing these analysis objects and event summary data is time consuming, which may necessitate distributed processing. Since most HENP experiments are large collaborations, participating institutions often desire to replicate certain portions of the event summary data or analysis objects. For these and other reasons, such as reliability and redundancy, high-energy physics data like data from many other sources are distributed across diverse geographical locations. In addition, because the data volume is too large for most disk systems, all of the raw data, the majority of event summary data, and much of the analysis objects are on tertiary storage managed by mass storage systems, which make them inaccessible by analysis jobs designed to work with disk-resident files only.

The event summary data and analysis object data are typically stored as ROOT files [5]. An analysis job is usually performed on a set of selected events. Many of the existing analysis frameworks require these events to be specified as a list of files. These analysis frameworks then read all events in the files and leave the user code to filter out unwanted events. In most cases, only a fraction of the events in the data files are selected, and the time to read the unwanted events could be a significant portion of the total execution time. One objective of the Grid Collector is to eliminate the time spent to read


- *K. Wu, J. Gu, A. M. Poskanzer, A. Shoshani, and A. Sim are with Lawrence Berkeley National Laboratory, Berkeley, CA 94720. E-mail: {KWu, JGu, AMPoskanzer, AShoshani, ASim}@lbl.gov.*
- *W. Zhang is with Department of Physics, Kent State University, Kent, OH 44242. E-mail: zhang@hpacq.kent.edu.*
- *J. Lauret is with Brookhaven National Laboratory, Upton, NY 11973. E-mail: jeromel@bnl.gov.*


This work was supported by the Director, Office of Science, Division of Mathematical, Information and Computational Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

these unwanted events. A second objective is to automate most file and data management tasks, such as determining the event of interest and the respective files containing them, the locations of files, reserving space for the files to be moved, transferring files over the Grid or from mass storage systems, and reclaiming the disk space after analyses.

The Grid Collector project achieves these goals by combining two technologies, an efficient indexing scheme [6] [7] and a distributed file management tool called Storage Resource Manager [8] [9]. The Grid Collector provides a way to specify events of interest using conditions on variables (properties of the events) defined by the physicists. Because the events of interest are explicitly specified, it is possible for the analysis framework to avoid reading unwanted events. The file managements functions mentioned before are performed by the Storage Resource Manager software. The Grid Collector provides a way for an event to be associated with a set of files containing the appropriate event summary data and analysis objects, and get the data files from local or remote systems, including mass storage systems.

There are a number of projects with somewhat similar visions as the Grid Collector; two of the closest ones are STACS [10] and XROOTD [11]. The STACS project can be viewed as the progenitor of the Grid Collector. It demonstrated with one mass storage system and a fixed set of files that it is possible to automate the file management tasks. In contrast, the Grid Collector can support an arbitrary number of storage sites and mass storage systems. It also uses a more efficient indexing scheme. The XROOTD software was developed in the context of a high-energy experiment named BaBar [12]. XROOTD was intended as an extension of the ROOT Daemon ROOTD [5] to provide accesses to ROOT files stored on locally attached storage (any disk or storage attached to a single node), centrally available storage (NFS, NAS, SAN) or on mass storage systems (MSS). One key difference between the Grid Collector and XROOTD is that the Grid Collector provides event-level (i.e. object-level) data access rather than file-level data access. Another difference is that the Grid Collector relies on standard Grid technologies while XROOTD uses its own peer-to-peer protocol.

An overview of the Grid Collector is provided in Section 2. In Sections 3 and 4 we describe the two main components of the Grid Collector: the Storage Resource Manager and the bitmap indexing module. In Sections 5 and 6 we describe how the Grid Collector improves two types of analysis jobs. A short summary is given in Section 7.

## 2 Overview of Grid Collector

The current implementation of the Grid Collector includes a main server we call the Coordinator and a number of other servers to complete its functions. There is a client library that plugs in to the STAR analysis framework [14], which allows the analysis code to make use of the Grid Collector. Figure 1 shows a schematic diagram of the various modules involved. In the next sub-sections we briefly describe the key functions of each module.

### 2.1 Grid Collector Coordinator

The Coordinator is currently implemented as a CORBA server. It consists of four identifiable modules: Index Builder, Event Catalog, File Locator, and File Scheduler.

The Index Builder module ingests the tag files generated by the reconstruction process and uses the information to build a catalog for the events. The catalog contains hundreds of high-level attributes, called tags, about each event. Since the tags

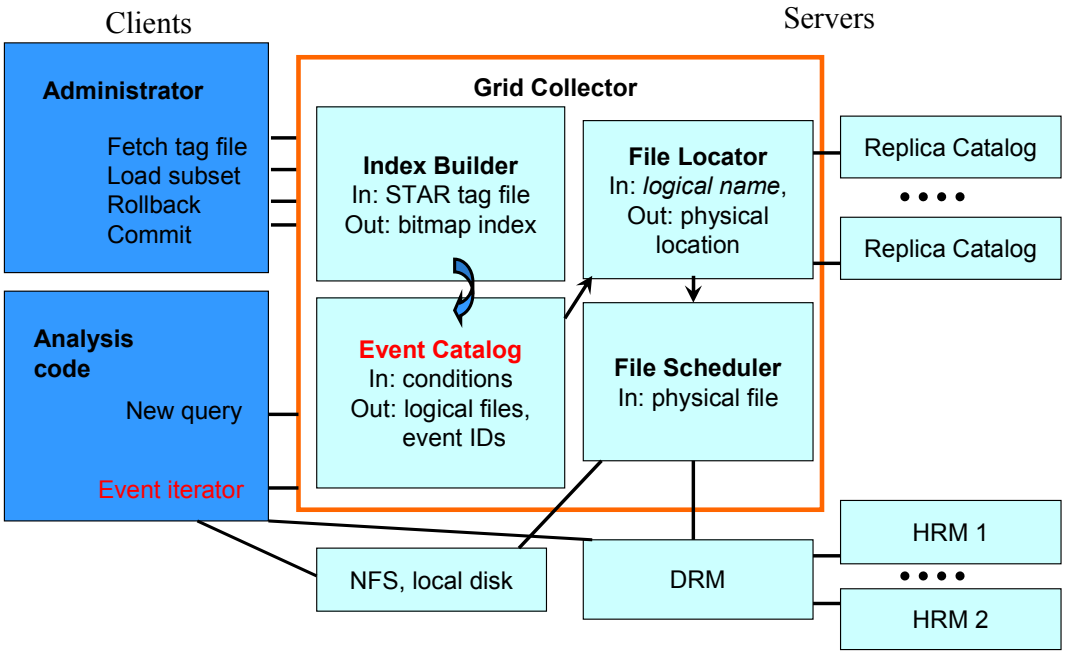


**Figure 1: The architecture of the Grid Collector.**

are defined by the physicists and are relatively small in numbers, they are natural candidates to be indexed and searched.

The core of the Coordinator is an Event Catalog. The data in this catalog is the same as those in the tag files. However, because we build an index for each tag, we can answer questions like "finding all events with chargedMultiplicity between 100 and 200" much more efficiently than using the tag files directly. This module takes queries from users, and outputs the logical names of files containing the events of interest and identifiers of the events in the files.

The File Locator module takes the logical file names and performs the necessary lookup to determine an appropriate copy of the file to use. If a copy is accessible by the client, its name will be returned, otherwise, the "closest" copy will be returned and the file will be scheduled for actual transfer. The File Scheduler decides which files to be transferred next. The actual transfer is performed through Storage Resource Managers. Our design allows the possibility of requesting multiple files at the same time. This is needed for analyses that use two or more different kinds of files simultaneously. In this situation, in order to analyze one event, multiple files must be present at the same time, which creates extra constraints on the file caching mechanism. We refer to the set of files that must be used together as a file bundle. The ability to handle file bundles is also a unique feature of Grid Collector.

### 2.2 External Services

The Grid Collector Coordinator requires two external services, Storage Resource Managers and Replica Catalogs. The replica catalogs contain information about where the replicas (copies) of the files are stored and how to access them. In STAR, we use STAR specific replica catalogs [15]. Potentially other types of replica catalogs can be used as well [16]. The Storage Resource Managers are used to transfer files and cache them for users.

We make use of two different types of Storage Resource Managers, a Hierarchical Resource Manager (HRM) and a Disk Resource Manager (DRM). The HRM is used to transfer files out of mass storage systems and the DRM is used to manage the local cache for the end users and to transfer files from other storage sites. More information about their functions is provided in the next section.

### 2.3 Clients

The clients of the Grid Collector come in two flavors, the Event Iterator that plugs into user analysis code and the stand alone CORBA clients that perform administrative and other functions. The administrative client instructs the Coordinator to build indices, and answers a small number of inquiries such as how many users are active.

The more interesting client is the Event Iterator. The end user code makes use of this module to interact with the Grid Collector Coordinator. It sends the query conditions, receives statistics about the query, such as the number of events selected, and finally, it reads selected events one at a time. The most important function of the Event Iterator is called *next*. If a file bundle is accessible, this function reads the next event of interest. After it is finished reading all events in a file bundle, the Event Iterator requests another file bundle from the Grid Collector. After calling the function *next*, an event is available in memory for analysis computations. The Event Iterator also works with the DRM to ensure that the files in use are pinned in the disk cache (i.e. cannot be removed) and that the pins are released after use, so that the space can be reclaimed.

Because the Event Iterator is designed to be a plug-in to an existing analysis framework, the Grid Collector can easily coexist with another framework. This offers the option for users to try out the Grid Collector with a minimal modification to their analysis programs. This type of a smooth transition path is quite useful to STAR users.

## 3 The Storage Resource Managers

Storage Resource Managers (SRMs) are Grid middleware components whose function is to provide dynamic space allocation and file management on shared storage components on the Grid [8]. They are designed to provide effective sharing of files, by monitoring the activities, and making dynamic decisions on which files to replace when space is needed.

Managing shared storage resources on the Grid is a necessary but complex task because of the diversity of the storage resources present. Storage resources can vary in complexity: a single disk under a UNIX file system, large sets of disk caches or disk RAIDs, or mass storage systems (such as HPSS). To simplify the access to these diverse resources, SRM provides a uniform interface for the clients. SRM supports many transfer protocols such as FTP, HTTP/S, GridFTP, or BBFTP, by automatic negotiation between the client and the server. It makes file transfers robust and reliable by automatically recovering from transient errors and retry.

SRMs also simplify the Grid client's interaction with storage systems. For example, it is a lot simpler for an application client to request one thousand files in a single request from an SRM regardless of their location on the Grid, rather than having to get each file from its source location. An SRM accepts a multi-file request, queues each file requested, retrieves the files from the source locations (using a file transport service such as GridFTP [22] from the Globus project [23]) based on space availability, and streams the files to the client. If files are found locally, they are pinned for a certain lifetime. The File Scheduler of the Grid Collector takes advantage of this function. An SRM can enable sharing of files between clients and between requests, makes storage usage more effective, and avoiding unnecessary file transfers over the network.

SRMs also enable a more effective use of the large shared storage space by automatic garbage collection. A common problem of managing shared storage resources is that files are deposited in such systems and often not removed. Much of the storage space may be occupied by files that are not needed any longer. SRMs reduce this problem by associating a lifetime with temporary files. This makes it easy for SRMs to reuse the space occupied by inactive files. SRMs support the "pinning" of files for the duration of a lifetime, as well as explicit "release" of files.

As depicted in Figure 2, an SRM which is placed in front of a mass storage system, such as HPSS [24], is referred to as an HRM). It manages a disk cache of its own, and interacts with HPSS to stage and archive files. When a request for files

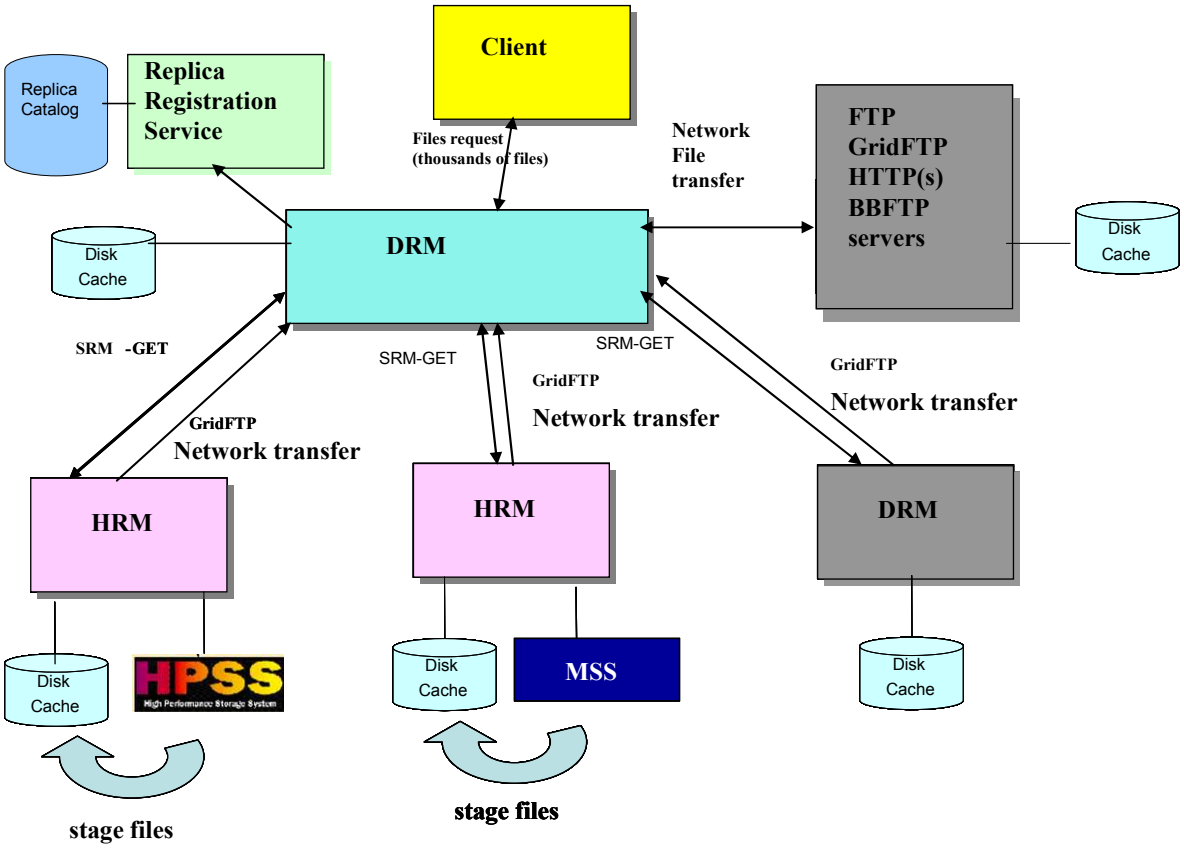


**Figure 2: A schematic drawing of Storage Resource Management.**

comes to a DRM of a client, it communicates with other HRMs, DRMs or file repository with FTP, GridFTP, BBFTP, or HTTP servers using the source URL. The source URL is usually the physical address of a file, or the address used by SRMs to access the file from the storage systems they access. The DRM then brings each file to its local disk to be accessed by the client. In the STAR experiment, the DRM also registers the files into the File Replica Catalogs upon a successful transfer. This enables the Grid Collector to make use of the new copies without any user intervention.

## 4 THE BITMAP INDICES

A key component of Grid Collector is the Event Catalog. It holds searchable attributes of events (called tags by the physicists), information about how to locate the file, and the location of each event within a file. Using it, one can specify conditions on any number of searchable attributes and get back enough information 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 because of the complexity of the problem of searching over millions or billions of event properties. An obvious way of building an event catalog is to put all the data into a commercial DBMS. However, this approach is not practical, since even commercial DBMS systems are not able to search efficiently the large volume of data produced from a typical HENP experiment. Frequently, the experiments use an open-source DBMS system to implement the file catalogs and replica catalogs. However, an event catalog would have 100 to 1000 times more records than the file catalogs. Because of their size, the challenge we had to address is how to search event catalogs efficiently.

The general technique for speeding up searching of large datasets is indexing. Recently, we implemented an efficient compressed bitmap index scheme. Complexity analyses show that our compressed bitmap indices are 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 [7]. Only few especially efficient indexing

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

**Table 1: The sizes and average query processing time on a subset of STAR data with 2.2 million events and 12 commonly used attributes.**

schemes, such as B-tree, have this optimality property. 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 or other indexing methods, including multi-dimensional indexes. 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 [6].

Table 1 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. We compare our bitmap indices against B-tree and the projection index because the B-tree is the most commonly used indexing scheme and the projection index is known to be the most efficient scheme for multi-dimensional range queries. Each average query processing time reported in Table 1 is an average over 1000 range queries with randomly generated range conditions. The same queries are answered with all three indexing schemes. A client program generates the queries and measures the query response time. For 5-dimensional range queries, our compressed bitmap indices use about one quarter of the time required by the projection indices and less than one tenth the time consumed by the B-tree indices. We have conducted a number of tests on different datasets, and our compressed bitmap indexing scheme consistently outperforms other indexing schemes [6] [7]. Our bitmap indices can also be built in much less time than the typical B-tree indices.

## 5 Speeding up Common Analysis Jobs

One of the main reasons for developing the Grid Collector is to speed up the analysis jobs. In a previous paper, we reported the performance of reading selected events without doing any useful analysis work [20], where the performance improvement on typical analysis tasks was between 2 and 100 depending on the number of events selected. However, the previous measurements assumed that the whole content of an event had to be read into memory before the filter function can be applied. In this section, we report some timing results using a flow analysis program [21]. The time measured includes both the I/O time and the analysis computation time. More recently, the STAR experiment has reorganized the analysis object data files to make them more efficient for the commonly used filtering mechanism. It was important to us to measure against this improved filtering mechanism. Next, we describe these results.

The measurements were performed on a Linux computer with two Intel Xeon 2.8 GHz processors and 2 GB of main memory. The tests use a subset of recently produced analysis objects totaling about 8 GB. The files used in the tests reside on an active disk vault; the programs run on a lightly loaded machine. We ran two equivalent programs with the same settings for analysis computations. The main difference between the two is that one uses the existing filtering mechanism while the other program relies on the Grid Collector to perform the same selection. The filtering mechanism requires part of an analysis object to be read into memory before a decision is made on whether to continue the analysis computation. Using the Grid Collector, only the events required for the analysis computation are read, and therefore the program using the Grid Collector should take less time. Because the ROOT files storing the analysis objects are compressed in baskets (blocks) and a basket has to be decompressed before any data in the basket can be accessed [5], the exact amount of time can be saved is not obvious.

The speedup values shown in Figure 3 are ratios of time used by the analysis program with the existing filtering mechanism to the one with Grid Collector. The horizontal axis is the fraction of events selected for analysis computation. The dots are the actual measurements, where each dot is an average of 4 different runs with the same settings. The solid lines are trend lines produced by the plotting program (Microsoft Excel). Since the analysis portion of the programs performs the same computations, the one with Grid Collector uses less time because it reads fewer events. Because the data files are compressed ROOT files [5], reading an analysis object actually require a significant amount of CPU time. In Figure 3, the speedup values for elapsed time is less than those for CPU time because both programs have to open the files through the same NFS system, which have the same file system delays, network delays and other random delays. In most test cases, these delays are longer than the CPU time required.

The test show that the program using the Grid Collector never takes more time than the one without Grid Collector. However, the speedup is less than the inverse of selectivity because the ROOT files are compressed in baskets. Selectivity is defined as the number of events selected for the analysis relative to the total number of events. When the selectivity is large, say, > 0.1, almost all baskets are read and decompressed in order to access the selected events. If fewer baskets are read or fewer files are opened, then the speedup is more significant.

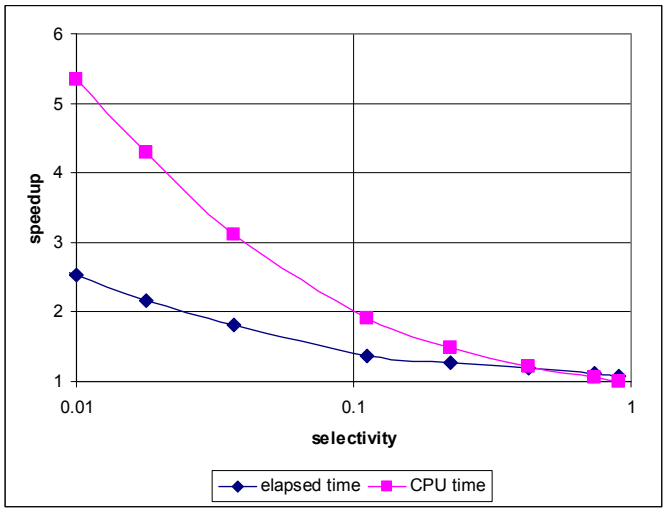


**Figure 3: Using Grid Collector reduces both CPU time and elapsed time, and speeds up analysis jobs.**

There is a large class of flow analysis jobs with selectivity of about 0.1 [21]. For these jobs, using the Grid Collector reduces the CPU time about 50% and reduces the elapsed time about 30%. If every job uses 30% less time than before, the same computer center can accommodate 40% more analysis jobs. Clearly, using the Grid Collector is beneficial even compared against the improved filtering mechanism in STAR.

## 6 Enabling "Exotic" Analysis Jobs

Next, we briefly describe two cases that are very expensive in the current analysis frameworks, but are relatively inexpensive using the Grid Collector, namely analyses of rare events and analyses outside of the computer centers. Physicists often refer to such analysis jobs as "exotic". Unlike the common analysis jobs where the files are on disk, these jobs typically involve files that are not already on disks. Because of the excessive amount of manpower and disk resource required to complete these jobs, they are practically prohibitive, and are usually put on the backburner for years, which may hinder critical scientific discoveries.

### 6.1 Analyses of rare events

Rare events in high-energy collisions could reveal important new physics insights. For example, one of the main missions of the STAR experiment is to search for the quark-gluon plasma, however, out of the many millions of collision events already collected there are very few that display unambiguous signatures of quark-gluon plasma, or other new forms of matter such as strangelets, or new phenomenon such as jet quenching [17] [18] [19]. Clearly, analyses of rare events are important to the overall goals of HENP experiments. Frequently, there is an initial search for rare events using the analysis objects on disk followed by one or more analyses on progressively fewer but more promising events. Because the follow-up analyses often require files not on disk, they can not be easily performed which forces the scientists to device new analysis strategies to analyze the data on disk to reveal the same information. The Grid Collector can remove most bottlenecks and make the follow-up analyses relatively straightforward.

To illustrate the use of the Grid Collector in these cases, we give one example of searching for the evidence of jet quenching. An initial search has been performed on a large number of analysis objects, and a small number of events (about 80) were found to have unusual jet distributions, which could indicate jet quenching. To further investigate these 80 events, they have to be extracted from the files containing them. However, because the events are scattered in many large files and most of the files are on tertiary storage systems, the analysts are not willing to spend the disk resource and manpower to transfer the files themselves. For this reason, the follow-up analyses were delayed for three years. However, once the analysts learned of the Grid Collector, they were able to extract these 80 events within 15 minutes. In STAR, the Grid Collector has also helped in searching for anti-Helium-3 ($^3\overline{He}$) and potential evidences of strangelets.

### 6.2 Analyses outside of computer centers

Many physics groups have access to their own computing resources outside of the main computer centers. Often the aggregate computing power of these scattered resources rivals or surpasses those of the computer centers. So far, these scattered resources have not been put into productive analyses of HENP data. One of the main difficulties is the need to manage distributed files and disk spaces.

The Grid Collector makes this type of analysis much easier than before. On the analysis machine, the user only needs the Event Iterator library and an installation of DRM. The Event Iterator can be configured to use one of the Grid Collector Coordinators running in the computer centers. At this time, there are two Coordinators running, one at Brookhaven National Laboratory and one at Lawrence Berkeley National Laboratory. Users can specify events of interests and have the Grid Collector transfer all the necessary files to the local DRM from the "closest" sources. This should make more computer power available for high-energy data analyses – boosting the analysis capability of HENP data.

## 7 Summary

The Grid Collector combines an indexing technology and a Grid file management technology to make analysis of high-energy physics data considerably faster and easier than using the existing analysis frameworks. For common analysis jobs where the required files are on disk, the Grid Collector can speed up the executions because it avoids reading unwanted events. For analyses that involve files not already on disk, the Grid Collector automatically transfers the necessary files and avoids the tedious manual file management tasks. With the same computer resources, more analysis jobs can be performed with the Grid Collector. The Grid Collector can also make more computer resources available for analysis by making it easier to use the scattered computer resources outside of the computer centers. The Grid Collector is designed as a plug-in for an existing analysis framework. This allows an on-going HENP experiment to easily boost their analysis capacity with a minimal amount of investment.

## Acknowledgment

The authors gratefully acknowledge comments and suggestions received from our users, in particular, those from Drs. Lee Barnby, Markus Oldenburg, and Aihong Tang.

## References

[1]. The GriPhyN Collaboration. "Petascale Virtual-Data Grids for Data Intensive Sciences." http://www.phys.ufl.edu/~avery/mre/white_paper.html. 2000.

[2]. J. M. Landgraf, M. J. LeVine, A. Ljubicic, Jr., J. M. Nelson, D. Padrazo and M. W. Schulz, and for the STAR Collaboration, "An Overview of the STAR DAQ System." *Nucl. Instrum. Meth. A* 499, pp. 762-767, 2003.

[3]. D. G. York, J. Adelman, J.E. Anderson, Jr., et al., "The Sloan Digital Sky Survey: Technical Summary." *The Astronomical Journal*, 120:3, pp. 1579-1587, 2000.

[4]. F. Carminati, P. Cerello, C. Grandi, E. Van Herwijnen, O. Smirnova, J. Templon, "Common User Cases For A HEP Common Application Layer -- HEPCAL," http://www.cern.ch/fca/HEPCAL-prime.doc. 2003.
[5]. Rene Brun and Fons Rademarkers, "ROOT user's guide," http://root.cern.ch/root/doc/RootDoc.html. 2004.
[6]. K. Wu, E. Otoo and A. Shoshani, "Compressing Bitmap Indexes for Faster Search Operations," *Proc. SSDBM'02*, pp. 99-108, 2002.
[7]. K. Wu, E. Otoo and A. Shoshani, "On the performance of bitmap indices for high cardinality attributes," *Proc. VLDB'2004*, pp. 24-35, 2004.
[8]. A. Shoshani, A. Sim and J. Gu, "Storage Resource Managers: Essential Components for the Grid," In *Grid Resource Management: State of the Art and Future Trends*, pp. 321-340, Kluwer Academic Publishers, 2003.
[9]. A. Sim, J. Gu, A. Shoshani and V. Natarajan, "DataMover: Robust Terabyte-Scale Multi-file Replication over Wide-Area Networks," *Proc. SSDBM'04*, pp. 403-412, 2004.
[10]. L. M. Bernardo, A. Shoshani, A. Sim and H. Nordberg, "Access Coordination of Tertiary Storage for High-energy Physics Applications," *IEEE Symposium on Mass Storage Systems 2000*, pp. 105-118, 2000.
[11]. A. Hanushevsky and A. Dorigo and F. Furano, "The Next Generation Root File Server," *Proc. CHEP 2004*, 2004. Software available at http://xrootd.slac.stanford.edu.
[12]. BaBar experiment's main web site is at http://www.slac.stanford.edu/BFROOT/.
[13]. STAR experiment's main web site is at http://www.star.bnl.gov/.
[14]. D.L. Olson, C.E. Tull and D. Prindle, "STAR Analysis Framework", Technical Report LBNL-39764. Lawrence Berkeley National Laboratory. 1997.
[15]. STAR File Catalog is fully described at http://www.star.bnl.gov/STAR/comp/Grid/fileCatalog/index.html.
[16]. A. L. Chervenak, E. Deelman, I. Foster, A. Iamnitchi, C. Kesselman, W. Hoschek, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services," *Proc. of the IEEE Supercomputing Conference (SC 2002)*, November 2002. IEEE Computer Society Press.
[17]. STAR Collaboration (J. Adams et al.). "Azimuthal anisotropy and correlations at large transverse momenta in p+p and Au+Au collisions at S(NN)**(1/2) = 200-GeV".
*Phys. Rev. Lett.* 93 pp. 252301-6, 2004.
[18]. STAR Collaboration (J. Adams et al.), "Multistrange baryon production in Au-Au collisions at S(NN)**1/2 = 130 GeV," *Phys. Rev. Lett.* 92, pp. 182301-6, 2004.
[19]. STAR Collaboration (J. Adams et al.), "Evidence from d + Au measurements for final state suppression of high Pt hadrons in Au+Au collisions at RHIC" *Phys. Rev. Lett.* 91, pp. 072304-9. 2003.
[20]. K. Wu, W.-M. Zhang, V. Perevoztchikov, J. Lauret, and A. Shoshani. "The Grid Collector: Using an Event Catalog To Speed up User Analysis in Distributed Environment," *Proc. CHEP 2004*, 2004. Document available at http://chep2004.web.cern.ch/.
[21]. A. M. Poskanzer and S. A. Voloshin, "Methods for Analyzing Anisotropic Flow in Relativistic Nuclear Collisions," *Phys. Rev. C.* 58, 1671-1678. 1998.
[22]. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in high performance computational Grid environments. *Parallel Computing Journal*, 28(5):749–771, 2002.
[23]. The Globus Alliance, http://globus.org/.
[24]. HPSS. High Performance Storage System, http://www.sdsc.edu/HPSS, San Diego Supercomputer Center, La Jolla, CA, 1997.