Large-scale Real-time Data-driven Scientific Applications

Junwei Cao and Junwei Li
Research Institute of Information Technology
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing 10084, P. R. China
e-mail: jcao@tsinghua.edu.cn

***Abstract*—Large-scale real-time data processing is becoming common in many scientific disciplines. But processing large amount of data in real-time is still challenging with existing technology. In last few years, the dynamic data driven approach is becoming people's spotlight due to its potential in reducing data intelligently. Enlighten by this concept, a new data-driven framework for large-scale real-time data analysis is proposed in this work and a scientific application under this framework is given in details. By introducing additional information to data analysis processes, large-scale data processing can be achieved with real-time time constraint.**

Keywords-Real-time Computing; Dynamic Data-driven; Scientific Applications; Scalability

I. INTRODUCTION

More and more scientific applications require manipulation of large amount of data, for example, the Large Hadron Collider (LHC) at CERN, the Laser Interferometer Gravitational Wave Observatory (LIGO), and the Sloan Digital Sky Survey (SDSS). These applications are producing up to terabytes of data per day, e.g. LHC are projected to produce several petabytes per year [1]. And in many cases, these large scale data need to be processed in real time. For example, LIGO requires initial data processing to be completed within 30 minutes after data is generated from observatories. Given this scenario, how to construct a framework to process large-scale data in real time becomes major challenges. In the past two decades, people have offered numerous solutions to address this issue.

In 1980s, the approaches used to process large-scale real-time data were not mature. Many researchers, technical managers and government contract monitors even had misconceptions about real-time computing [2]. Since in this period, related applications concentrated more on real-time data management, data processing were less important, so we collectively call this period's applications as massive data management.

In the spring of 1989, the concept, data intensive computing, was originated from a demonstration that would relate remote visualization and networking to prove the impact of high-speed networks by Craig Fields [3]. Purely data intensive applications process multi-terabyte to petabytes-sized datasets which commonly comes in several different formats and is often distributed across multiple locations [4].

In 2000, data grid was proposed [5] to design an integrating layered architecture which provides aggregated resources and common components customized for a set of different large-scale data-intensive applications. In this integrating layered architecture, low layers provide high performance access to basic, policy-independent mechanisms and upper layers define application specific behaviors. By this way, in a data grid, basic mechanisms can be reused while high performance and specialized capabilities are provided to end users and applications.

In recent years, grid computing technologies provide advanced computational capabilities for applications and application simulations. At the same time, measurement infrastructures, from kinds of instruments to sensor systems, to data storage technologies and remote data access have also matured [6]. In such a case, Frederica Darema posted a new term, Dynamic Data Driven Applications System (DDDAS) when he led the organizing effort of the NSF Workshop [7] in March 2000. The DDDAS dynamically integrates both computation aspect and measurement aspect of an application, not considering the two aspects separately. DDDAS can be viewed as a methodology to counterbalance incompleteness in model and capability to enhance the application models by imparting additional information into the model as at runtime additional data are used to selectively enhance or refine the original model [8]. In other words, DDDAS's motivation is to feed additional data from a data archive or collected online measurements into an executing application to give the application ability to control and guide the measurement processes to make measurement processes more efficient and effective by selectively focusing on a subset of the measurement space. Since in many cases, real-time data are generated by measurement systems, for instance, kinds of sensor network, or by systems which can be regarded as measurement systems, e.g. simulations, DDDAS actually provides us a new angle to handle large-scale real-time applications.

From earliest massive data management to recent dynamic data driven applications, concept has changed a lot, but each one has its own drawbacks. Faced with exponentially growing data volumes, data intensive computing is difficult to achieve absolute real time processing to the whole large scale data due to many factors, e.g. wire speed limitations, geographical separation. So a natural idea comes out which is that we can try to only process valuable data in real time. So it requires discriminating data's value, which data intensive computing can not provide. Although DDDAS can meet this requirement by selectively processing some subsets of data with suggestions given by additional data analysis, it mainly focuses on model refining, lacks of universality to be applied to general large-scale real-time applications.

In this work, a new large-scale real-time data-driven framework is proposed. It uses data grid tools, adopts

additional data from DDDAS, and can be applied to general large scale real-time applications. In Section 2, the new framework we propose is introduced in details. A gravitational wave data analysis system is given as a demonstration for this framework in Section 3. Conclusions and future work are included in Section 4.

II. LARGE-SCALE REAL-TIME DATA-DRIVEN FRAMEWORK

The whole framework consists of 4 layers. In the bottom-up sequence, these are instrument layer, data monitoring layer, data processing layer and application layer. The details are illustrated in Fig. 1.

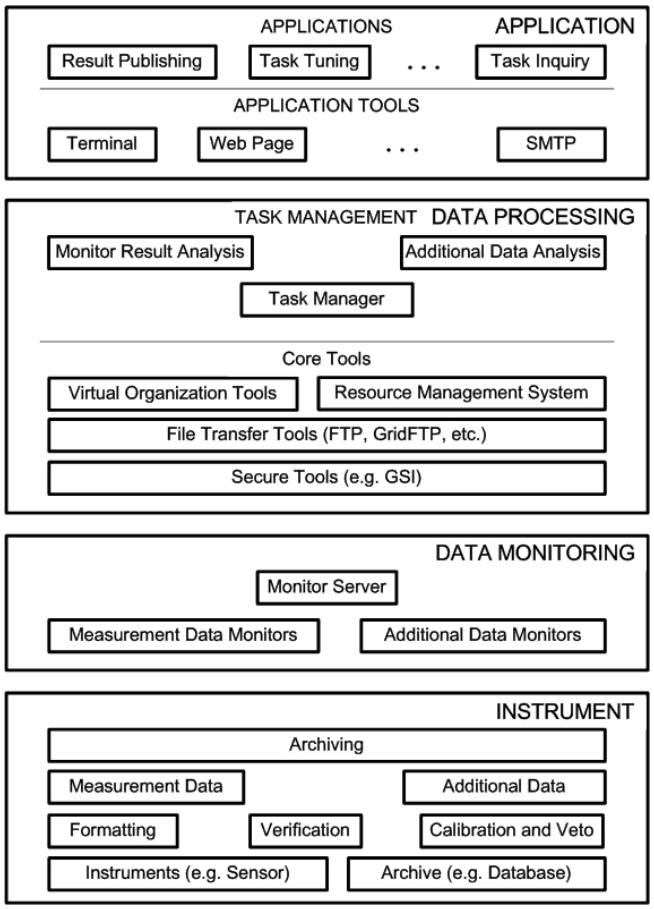

Figure 1. A layered structure.

The instrument layer has two parts, producing measurement data and additional data.

- In measurement data part, measurement data are collected from one or multiple measurement systems, which can be sensor networks or simulation systems and covers most data sources in large-scale real-time applications. Maybe there exist some discrepancies between these multiple measurement systems due to measurement systems' different physical principles, so the measurement data format of these systems can be different. In this situation, formatting these different data into a uniform format is needed, for the consideration of convenience in the following process. Moreover, we can't exclude abnormal measurement's appearance due to measurement equipment's error or some other factors. So verification to measurement data is essential in measurement part. After verification, calibration to altered measurement data and veto to bad measurement data are needed.
- In additional data part, it is analogous to measurement data part that additional data can be collected from one or multiple nodes in different types, e.g. additional data can be collected online or from a data archive. So formatting, verification, calibration and veto are also needed.

An important point is that formatting, verification, calibration and veto should all be done in location where data is generated, for reducing the overhead of transferring data and convenience in the following data processing. Finally, measurement data and additional data should be archived in the instrument layer.

Data monitoring layer's function is to dynamically monitor properties of measurement data and additional data in graphical interfaces. All monitors should register themselves to the monitor server for unified management.

In data processing layer, firstly, additional data is analyzed using the results of monitoring as reference. Then, task manager will start several tasks according to the results of additional data analyzing. In many applications, there are requirements for multi-resolution or multi-group of parameters. In other words, these applications require to be run in multiple instances with different resolutions or parameter values. Moreover, a task can be a single job or a workflow according to its runtime environment, and each task should have its own dedicated computational resources, which could be heterogeneous. For example, a task manager manages two tasks, one is in a single-computer environment and the other is in a grid environment. The one in the single-computer environment can be treated as a single job and the other one in the grid environment which is partitioned into sub-tasks can be treated as a workflow. Except for assigning computational resource, the task manager also handles establishing, monitoring, tuning, result collecting, terminating and other issues about tasks.

Application layer mainly contains tools for task result publishing, task parameters tuning and task's dynamic & static information inquiry, e.g. publishing results in a web page or via email, tuning parameters in a terminal with command lines.

The proposed framework requires technical support in many aspects, such as schedule algorithms, application architecture design. Here we enumerate the important technical requirements related to the framework.

- Scalability in data monitoring. Towards large-scale applications, it is common that their users are scattered in a nation or even internationally, and each user's requirement is different. So if scattered users want to monitor same real time data at the same time in different time scales, problems appear. First problem is how to arrange properties of

monitored data in different time scales. Second problem is how to eliminate conflict when a same monitor is used by scattered people with different demands.

- Task scheduling with priority, workflow and time constraint. Running multiple tasks of an application concurrently with real time data streams requires new scheduling algorithm supports. The arrangement of priority to multiple tasks which are only different in parameter values with consideration of time constraint in real time environment is a challenging issue, so the scheduling should be priority and time constraint based. Since a task can be partitioned into sub-tasks, scheduling should be workflow-based, and perform fine-grained scheduling for each workflow. It is also important to assign appropriate computational resources in the heterogeneous computing environment to each task and sub-tasks.
- Reliability in task scheduling. In a real time system, time constraint in task scheduling is obviously important, but reliability is also an important aspect since reliability strongly influence whether time constraint can be meet, especially in a multiple-task environment.
- Task tuning. Automated and manual task parameter tuning are both important. Automated tuning is required to adjust the task model in real time processes. And manual steering is essential since automated steering may not work well. For the convenience of tuning parameters, in application and simulation design we should preserve all the parameters in a plain text file. Then tuning parameters gets simple by editing the parameters file.
- Visualization, publishing results and notification. In visualization, integrating correlative data in a uniform interface at the same time, dynamically visualizing real time data and providing multiple visualization modes (e.g. web pages, software interfaces) are important for users. Besides, publishing results which can be shared to others is also important. Notification of events to special users, e.g. system administrator and research scientists, should be quick and robust enough.

III. AN APPLICATION CASE STUDY

In this section we use gravitational wave data analysis as an application case study of the proposed large-scale real-time data-driven framework.

A. Application Background

The theory of general relativity is a geometric theory of gravitation developed by Albert Einstein in 1916. Gravitational wave is a fluctuation in the curvature of space-time which propagates as wave.

To directly detect gravitational wave, LIGO (Laser Interferometer Gravitational Wave Observatory) is built by California Institute of Technology (Caltech) and the Massachusetts Institute of Technology (MIT). LIGO uses laser interferometers to detect the gravitational wave by measuring the interference of two laser beans whose length is changed when the gravitational wave passes by. LIGO has two observatory sites (H and L) in the United States with a total of three laser interferometers (H1, H2 and L1). In addition to the two observatory sites in United States, LIGO also collaborates the British-German GEO [9] 600m detector located near Hannover, Germany; and shares data with the Italian/French VIRGO [10] detector.

Today only ten percent of all the matter in the universe can be observed in the traditional way. LIGO can give us an insight into the majority of matter which can't be observed in the universe by detecting gravitational waves. So LIGO provides us a new window to observe the universe [11].

B. Gravitational Wave Data Analysis

LIGO searches four classes of gravitational waves: chirp signal, which are caused by the inspiral and collision of binary neutron stars or black holes; unmodeled bursts from various sources including supernovae; periodic signals from pulsars; and stochastic radiation from the Big Bang [12]. In the following content, we will only take burst search as an example, since the searches for the four classes of gravitational waves are almost same in terms of the system structure.

In principle, the whole LIGO data computing system consists of 3 parts: observatory sites, central location and offline system. Now LIGO can use three observatory sites. They are LHO (LIGO Hanford Observatory), LLO (LIGO Livingston Observatory) and VIRGO. In Fig. 2, it shows main software components in one of LIGO's observatory sites.

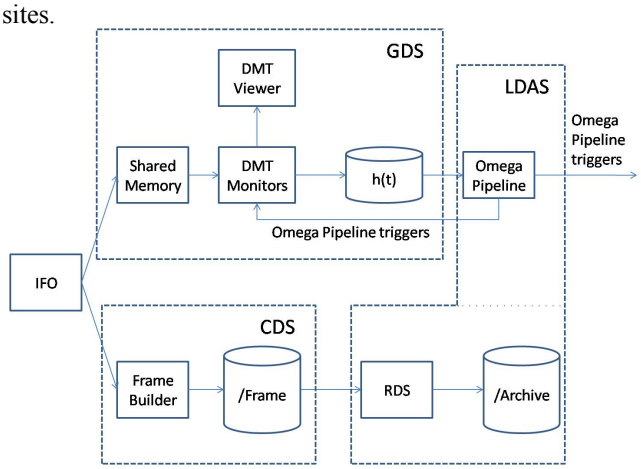

Figure 2. Main components in an observatory site

IFO is the abbreviation of interferometer observatory. Data from IFO is sampled and digitized by the Control and Data System (CDS [13]). These digitized signals are collected at rates as high as 16384 Hz from digital controllers and various analog signals from LIGO sensor channels. Then the frame builder in CDS formats the digitized signals into LIGO standard data frame format and then store that data to disk. A frame contains descriptor information on all of the data contained within a frame and data for all channels

acquired in a given time period. At present, LIGO is producing frames that contain 64 seconds of data. These frames are archived for long term storage. As new frame is generated, it is recognized by LIGO Data Analysis System (LDAS [14]), and then Reduced Data System in LDAS will generate RDS frames as subsets of the raw frame by using some filtering techniques.

As data from IFO is processed by CDS and LDAS, it is also collected by a shared memory. Global Diagnostics System (GDS [15]) will diagnose the interferometer system by monitoring the data. To be more specific, the monitoring work is done by multiple monitors of Data Monitor Tool (DMT [16]) in GDS. By monitoring, types of triggers will be generated. By analyzing these triggers, we can get information about interferometer system, e.g. information about whether this IFO is abnormal or validation of a specific data channel. These triggers are ingested into a LDAS meta-database and will be used to make various cuts on the analysis cycle and data products [14].

After DMT's monitoring, GDS will convert the DARM_ERR channel's data combined with calibration information to calibrated strain data represented by h(t) [17]. It is recorded in a file with special format, named with a timestamp derived from the GPS, the observatory site's name (for example, H1, H2) and other information [18]. Calibrated strain data h(t) is the main data which contains information about gravitational wave, so the following analysis mainly concentrates on h(t). For the convenience of the follow-up analysis, h(t) data is divided into segments. As h(t) is generated, it will be archived and sent to a data analysis pipeline named Omega Pipeline [19] as pipeline's data source. Omega Pipeline will process each h(t) segment, and generate trigger file accordingly as output if gravitational wave's signal intensity is so high that it can be distinguished from the noise. Currently, a trigger has 5 attributes: central GPS time, central frequency, duration, bandwidth and normalized energy. These attributes contain useful information of waves (not only gravitational wave, a lot more are noises) detected by observatory. These triggers will be monitored by DMT, visualized by the DMT viewer and transferred to a central location for follow-up analysis.

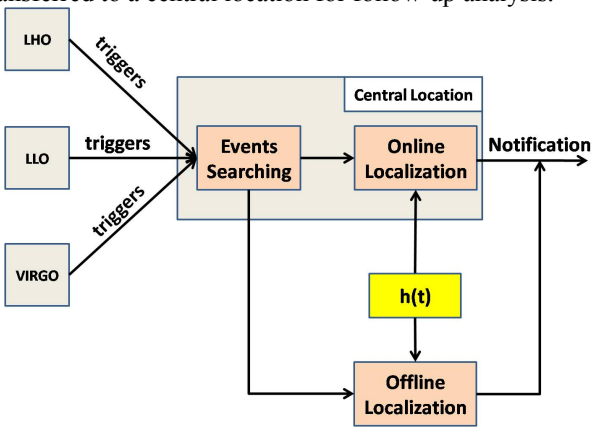

Figure 3. Components in central location and offline system

In Fig. 3, it shows schematic components in the central location and offline system. Trigger files generated at each site will be transferred to this central location, and these three sites' triggers will be analyzed coherently to search events in which gravitational wave may exist. Fig. 4 shows event searching in general.

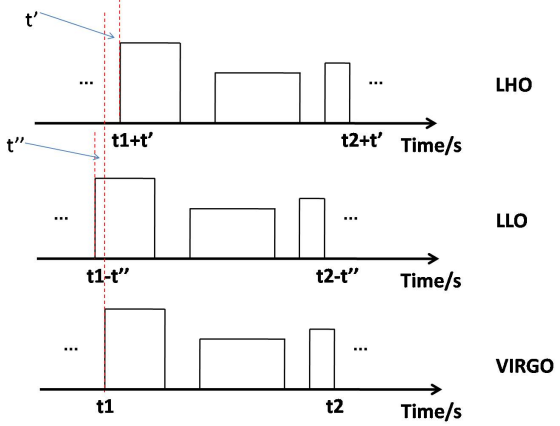

Figure 4. Event searching principle

In Fig. 4, there are three time axes; each represents one observatory site's trigger series. Trigger performs in the square waveform. One square waveform represents a trigger.

As we know, a gravitational wave reaches 3 observatory sites in different time because of different distances. So if three observatory sites all receive this gravitational wave, then there exist time shifts among the time when gravitational wave is received in three sites. From Fig. 4, it can be inferred that the three triggers in LHO each has a *t'* time shift compared to VIRGO's, and three triggers in LLO each has a *t''* time shift. So if the three triggers in three sites are similar after the time shift is eliminated, the time interval which covers the three triggers may contain a gravitational wave, which is called an event.

To find the sky position where an event sources, the central location will analyze h(t) data corresponding to the triggers, and inform astronomers to observe these events immediately. But this type of reconstruction can use infinite computing power, since it requires to search a huge parameter space which mainly contains the position in the sky, gravitational wave frequency, and rate of change of frequency [18]. And this parameter space can be larger without any limitation. This is because position in the sky is a continuous variable, so here sky is cut into multiple areas; position in the sky actually means an area in the sky. The area can be smaller, so sky position resolution is unlimited.

Since LIGO's science run has a real time requirement of signal reconstruction, which can use many computing resources, central location has to restrict the position resolution in the sky. But the following scenario may occur with resolution constraint. If the astronomers don't observe events in a given position by the central location, a more precise position is required. This can be carried out by the offline mode.

The relationship between central location and offline system is as follows. After an event is discovered by central

location, central location will start online localization on a cluster and offline localization on a grid (e.g. Open Science Grid [20]) at the same time. Astronomers can be notified in time by online localization with sacrificed position resolution. In a period of time after astronomers are notified by online localization, offline localization will provide a more precise position to astronomers.

C. *Application Requirements*

LIGO interferometers generate about 10 megabytes of data per second, which is almost one terabyte data per day [21]. And LIGO interferometers are highly distributed in thousands of kilometers away. To handle such large-scale real-time data, LIGO gravitational wave data analysis system has the following characteristics.

- Data reduction. For example, in LIGO Hanford Observatory, raw data rate is 9.063 megabytes per second, but after processing of Reduced Data System mentioned before, data rate can be reduced to 0.029 megabytes per second [22]. Another example is about trigger files. Trigger files can be treated as indexes to h(t) data segments. A common h(t) data segment is at one megabyte level, but a trigger file is only at one kilobyte order.
- Adoption of data grid tools. For example, to locate data and get high-speed, reliable access to data from the central location to observatory site, LIGO developed tools like LSCdataFind and LIGO Data Replicator [23] by using and customizing many data grid tools, e.g. Globus RLS (Replica Location Service [24]) and GridFTP [25].
- Dynamic data driven. In LIGO gravitational wave data analysis system, h(t) can be regarded as measurement data, and triggers can be regarded as additional data. Through trigger analysis, the system can distinguish which subsets of h(t) possibly contain information about gravitational waves, and only analyzes the subsets in localization.

D. *Data monitoring*

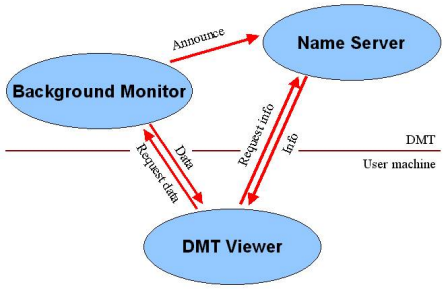

Figure 5. Mechanism of the DMT system

Fig. 5 [26] explains how the whole DMT system works at runtime. In general monitors run at background. In the gravitational wave data analysis system, there can be many DMT monitors. Monitor doesn't display any graphics by itself, but serving data to the DMT viewer. The procedure is as follows. Firstly, user starts a monitor, and monitor announces its name and port information to the name server. The name server is the monitor server which keeps track of which monitor is available. Then users use DMT viewer to connect to the name server and makes a request for the list of monitors currently running and these monitors' port information. User will select monitor in the list, DMT viewer will then request a list of data objects (Data objects namely are properties of data monitored) directly from the monitor by using port information. Finally, after the user has selected one or more data objects, the data itself is downloaded from the monitor, transferred to DMT viewer and shown in the graphical pad of DMT viewer. Since name server's port is pre-fixed, so monitors and the DMT viewer can always find name server and establish communication.

It can be inferred that monitor, name server and DMT viewer can be at totally different locations since they are connected by Internet. This is the basis of scalability in data monitoring. Scattered users can start and shutdown their own DMT viewer in their local host at anytime without influence to each other. DMT monitors are built on a set of C++ libraries. One typical monitor should have the following four functions [27].

- Constructor. This function initializes the monitor.
- Destructor. This function cleans up after the monitor is complete. It typically flushes and pending output.
- ProcessData. This is the primary function of the monitor that is called repeatedly for processing each new block of data.
- Attention. This function handles interrupt request. It typically calls the interrupt service handlers for services such as DMT viewer requests.

E. *Data processing*

In LIGO gravitational wave data analysis system, reconstruction of an event is done via scanning the whole sky by Omega Pipeline. In Omega Pipeline, a variable named skyPosition sets the entire sky to be analyzed by default. If skyPosition is set as 2-element cell of 2-element vectors (similar to {[aMin aMax] [bMin bMax]}), skyPosition specifies a range of the sky to be analyzed [28]. To partition the localization task into sub tasks, the entire sky should be partitioned into multiple non-overlap ranges.

In Fig. 6, it shows the structure of task management. Task manager handles two tasks, online and offline localization. Online localization is based on clusters and offline localization is based on grids. Both tasks are a workflow of sub-tasks. Online localization task's priority is higher than offline localization task forever since online localization needs to provide faster notification.

Since online localization task and offline localization task are similar, we next only take the procedure of offline localization for instance. Firstly, triggers are analyzed to find events. Task manager generates and sends an offline localization task to RMS according to an event. RMS will partition the task into multiple sub tasks according to the analysis of both performance of grid and reliability of resources. Then RMS will generate a task description file for each sub task. A task description file contains the range of sky a sub task should scan, the assigned resource information, the required h(t) data information, etc. All task

description file will be sent to corresponding resources by a daemon. The daemon requires each sub task information until sub task is finished and result returned. Integrator combines all sub task results together and send to task manager. Task manager then pushes the combined result to astronomers.

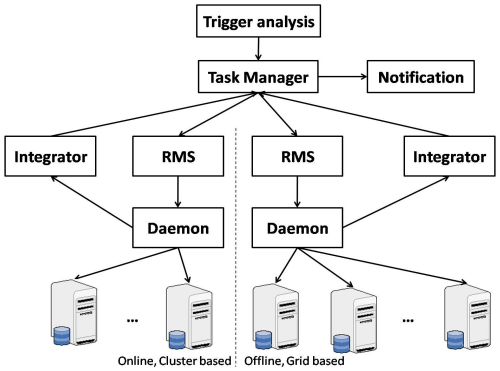

Figure 6. Structure of task management

IV. CONCLUSIONS

The concept of dynamic data driven system is mainly driven by simulation applications. In simulation applications, it is essential and important to optimize parameters to simulate real systems and additional data can provide additional information to support better parameter optimizing.

Whether we can use additional data's information to better reduce data volume to be processed in common large-scale real-time applications is a challenging issue. The framework presented in this work provides an environment to address this issue. LIGO gravitational wave data analysis system is a typical case study of the proposed framework.

In the near future, scales of data will undoubtedly increase in an exponential order. Under such circumstances, large-scale real-time data-driven framework will obtain more opportunities to reveal its potential with more emerging scientific applications like LIGO. Our future work includes the application of the framework to more domains.

ACKNOWLEDGMENT

This work was supported by National Science Foundation of China (grant No. 60803017) and Ministry of Science and Technology of China under National 973 Basic Research Program (grants No. 2011CB302505 and No. 2011CB302805).

REFERENCES

[1] B. Allcock, A. Chervenak, I. Foster, C. Kesselman, M. Livny. Data Grid tools: enabling science on big distributed data. J. Physics: Conference Series 16, 571-575 (2005).

[2] J. A. Stankovic. Misconceptions about real-time computing: a serious problem for next-generation systems. IEEE Computer, 21(10), pp. 10-19 (1988).

[3] W. E. Johnston. High-speed, wide area, data intensive computing: A ten year retrospective. Seventh IEEE Int. Symp. on High Performance Distributed Computing, pp. 280 (1998).

[4] I. Gorton, P. Greenfield, A. Szalay, R. Williams. Data-intensive computing in the 21st century. Computer, 41 (4). pp. 30-32 (2008).

[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. J. Network and Computer Applications, 23, 187-200 (2000).

[6] F. Darema. Dynamic data driven applications systems: New capabilities for application simulations and measurements. ICCS 2005, LNCS 3515, pp. 610-615 (2005).

[7] NSF Workshop, March 2000. www.cise.nsf.gov/dddas

[8] F. Darema. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. ICCS 2004, LNCS 3038, pp. 662-669 (2004).

[9] B Willke, P Ajith. The GEO-HF project. Classical and Quantum Gravity 23, S207-S214 (2006).

[10] F Acernese, M Alshourbagy. Virgo status. Classical and Quantum Gravity 25 (2008).

[11] Overview of LIGO. http://www.ligo-la.caltech.edu/contents/overviewsci.htm

[12] K. Chan. A correlation technique to identify coincident bursts in data from LIGO interferometer pairs. http://www.ligo.caltech.edu/~ajw/bursts/KChan_finalpaper6.pdf

[13] R. Bork, R. Abbott, D. Barker, J. Heefner. An Overview of the LIGO Control and Data Acquisition Systems. 8th International Conference on Accelerator & Large Experimental Physics Control Systems (2001).

[14] S. Anderson, K. Blackburn, A. Lazzarini, W. Majid, T. Prince, R. Williams. The LIGO Data Analysis System. Gravitational Waves and Experimental Gravity (1999).

[15] GDS- Global Diagnostics system. http://www.ligo-wa.caltech.edu/gds/dtt/gdsOverview.html

[16] Data Monitor Tool Project. http://www.ligo.caltech.edu/~jzweizig/dmt/DMTProject/

[17] A. Dietz, J. Garofoli, G. Gonzalez, M. Landry, B. O's Reilly, M. Sung. Calibration of the LIGO detectors for S4. http://www.ligo.caltech.edu/docs/T/T050262-00.pdf

[18] B. Abbott, R..Abbott, R. Adhikari. Search for gravitational-wave bursts in LIGO data from the fourth science run. Classical and Quantum Gravity 24, pp. 5343-5369 (2007).

[19] Omega Pipeline. https://geco.phys.columbia.edu/omega

[20] OSG – Open Science Grid. http://www.opensciencegrid.org/

[21] S. Koranda. Cataloging, Replicating, and Managing LIGO Data on the Grid. http://www.mardigrasconference.org/conf_2005/2005/Presentations/Koranda.pdf

[22] G. Mendell. LIGO S5 Reduced Data Set Generation. http://www.ligo.caltech.edu/docs/G/G070127-00/G070127-00.pdf

[23] Kevin Flasch. Data Replication in LIGO. http://www.ligo.caltech.edu/docs/G/G070905-00.pdf

[24] RLS - Replica Location Service. http://www.globus.org/toolkit/data/rls/

[25] The Globus Project White Paper. GridFTP: Universal Data Transfer for the Grid. http://www.globus.org/toolkit/docs/3.0/gridftp/C2WPdraft3.pdf

[26] DMT system diagram. http://www.ldas-sw.ligo.caltech.edu/cgi-bin/cvsweb.cgi/~checkout~/gds/doc/quickstart/MonitorAPI/monserv1.jpg?rev=1.1;content-type=image%2Fjpeg;cvsroot=GDS

[27] DMT. http://emvogil-3.mit.edu/~shourov/fan/dmt/update_20080822.html

[28] Sky position specification in Omega Pipeline. https://geco.phys.columbia.edu/omega/wiki/Documentation/omega/skyPosition