Collaborative Framework for Scientific Data Analysis

Jaliya Ekanayake [Corresponding Author]
Community Grids Lab, Indiana University
501 N Morton St, Suite 224, Bloomington, IN 47404 USA
Email: jekanaya@indiana.edu
Phone: 1 - 812-606-0561 Fax: 1-812-856-7972

Shrideep Pallickara
Community Grids Lab, Indiana University
501 N Morton St, Suite 224, Bloomington, IN 47404 USA
Email: spallick@indiana.edu
Phone: 1 - 812-856-1311 Fax: 1-812-856-7972

Geoffrey Fox
Community Grids Lab, Indiana University
501 N Morton St, Suite 224, Bloomington, IN 47404 USA
Email: gcf@indiana.edu
Phone: 1 - 812-856-7977 Fax: 1-812-856-7972

ABSTRACT

Higher levels of scientific data analysis are often done by human interpretation. Once the data is processed to a certain level the rest is done by multiple steps of processing and human interpretation. Many large scientific experiments expand over multiple organizations and hence the data, the teams working on these experiments and expertise on the area of research are all distributed across these organizations. Collaboration is a key requirement when the focus of an analysis is to extract new knowledge. Real time or near real-time collaboration of expertise to a scientific data analysis provides a better model of interpretation of the processed data. In this paper we present a collaborative framework for scientific data analysis that is also secure and fault tolerant.

KEYWORDS: Frameworks and Methodologies for Collaboration, Collaborative & Distributed Systems, Architectures and Design of Collaborative Systems

Collaborative Framework for Scientific Data Analysis

Jaliya Ekanayake, Shrideep Pallickara and Geoffrey Fox
Department of Computer Science
Indiana University, Bloomington, IN 47404, USA
{jekanaya,spallick,gcf}@indiana.edu

ABSTRACT

Higher levels of scientific data analysis are often done by human interpretation. Once the data is processed to a certain level the rest is done by multiple steps of processing and human interpretation. Many large scientific experiments expand over multiple organizations and hence the data, the teams working on these experiments and expertise on the area of research are all distributed across these organizations. Collaboration is a key requirement when the focus of an analysis is to extract new knowledge. Real time or near real-time collaboration of expertise to a scientific data analysis provides a better model of interoperation of the processed data. In this paper we present a collaborative framework for scientific data analysis that is also secure and fault tolerant.

KEYWORDS: Collaboration Frameworks, Collaborative Distributed Systems, Scientific Computing.

1. INTRODUCTION

The final step of most scientific data analyses is done by human interpretation. Experts on a particular area of study are located in different parts of the country and in different countries. Collaborative scientific data analysis is a promising field of study as its goal is to bring the expertise scattered all over the world to a single data analysis session. Typical steps of a large scale scientific data analysis are:

- Locate the data from experiments (typically large volume of data).
- Execute various analysis functions on this data.
- Further process the analyzed data to produce graphs or simulations.
- Human interpretation where the collaborative approach has very high benefit to the interpretation process as it provides a framework for different expert minds to participate on the final step of the processing.

Typically these analyses are performed in a server cluster, computation grid or in a group of voluntary participant's machines as in the case of many projects under BOINC [1]. These analyses may produce different number of outputs (results) and in different time frames. These factors decide the way a collaborative session can be setup for the interpretation of data produced by the analyses. If the analysis phase has multiple steps and each phase takes fairly less time, then a synchronous collaboration would be the right model. On the other hand if the analysis takes longer time to complete then it is better to run the analysis first and collaborate on the interpretation of the results in asynchronous manner. In either cases the sharing of results can be done as shard displays or shared events.

Shared display model fits best for simply sharing results so that scientists who simply want to see the results of the ongoing experiment can see them without performing further processing at their sites. Shared event model on the other hand allows scientist to participate on an experiment actively by either performing further processing such as fitting models on the received events or simply merging the results. Both these paradigms have their own merits and demerits which we will discuss extensively when we present the architecture of the proposed system.

In many scientific data analysis the *composability* property, the ability to perform the data analysis as a collection of sub data analyses which can be executed concurrently and by producing the final result by merging the outputs of these sub analysis, is often exploited to achieve scalability. This type of analysis produces considerable amount of sub results and if we enable collaboration for such analysis collaborating participants will be able to see the results getting shaped as when the sub results are available. This will be a very dynamic collaboration if the number of sub results are high and they come at a fairly frequently.

High Energy Physics (HEP) data analysis provides an ideal example for composable data analysis. Typically the aim of such data analysis is to identify particular type of events within a collection of millions of events generated by HEP experiments. The result of such analysis would be a histogram of possible events. The analysis task is to go through all the events available and identify a particular set of events. We can easily break the initial data set into smaller subsets and run the same analysis software on these subsets concurrently. The resulting histograms can then be combined to produce the final result. In a collaborative session, each participant will receive these histograms (under a shared event model) and see them getting merged in real-time. [Please note that the term event in "shared event" has no correlation to the events in the HEP data analysis.]

As explained above, the participating scientists for a collaborative session may not come only from one organization or a single domain of control. It should be a global participation and supporting such collaboration requires careful attention on the security aspects of the framework for collaboration. How to authenticate various participants and authorize them to perform various activities? These are some of the questions that should be answered by the framework for collaboration.

When many scientists participate for a collaborative session for data analysis, it is very important to consider the fault tolerance aspect of the framework as well. A single failure of the processing entities, or software components should not waste the total man hours put into the analysis task. The systems should be able to cater for these failures and proceed with the analysis.

In this paper we present a framework for collaborative data analysis that supports different collaborative models, secure and fault tolerance. Section 2 discusses the core software components that we use in our framework while the section 3 presents the architecture in more detail. Section 4 of the paper discusses the advanced research prototype of the system and the section 5 presents the performance evaluation of our system. Section 6 discusses the related work in this area of research. We present our conclusions and future work in section 7.

2. SOFTWARE SUBSYSTEMS

To support a collaborative data analysis we identify three main types of software sub systems to be integrated. Data analysis subsystem, communication subsystem and control subsystem that control the overall collaborative framework. We use Clarens Server [2] developed at Caltech for the data analysis framework and NaradaBrokering [3] a publish/subscribe messaging substrate as our communication layer.

2.1. Clarens Server

Clarens is a grid enabled web service framework that supports most common web service protocol stacks comprising HTTP, SOAP/XML-RPC and SSL/TLS encryption and X.509 certificate-based authentication implemented in Python. Although the server implementation of Clarens is completely Python based, it provides client libraries for other languages such as Python, Iguana, JavaScript, and most importantly the C++ based interpreted language supported by the ROOT Analysis framework [4]. Support for all the above features as well as the integrated support for ROOT makes Clarens a key framework for various scientific data analyses.

2.1. NaradaBrokering

A content distribution infrastructure based on the publish/subscribe paradigm. The NaradaBrokering substrate itself comprises a distributed network of cooperating broker nodes. It can be used as a message oriented middleware or as a notification framework. Communication within NaradaBrokering is asynchronous and the substrate places no constraints either on the size, rate or scope of the interactions encapsulated within events or the number of entities present in the system. Also, it provides support for wide verity of transport protocols making it an ideal messaging infrastructure for heterogeneous distributed systems.

3. ARCHITECTURE

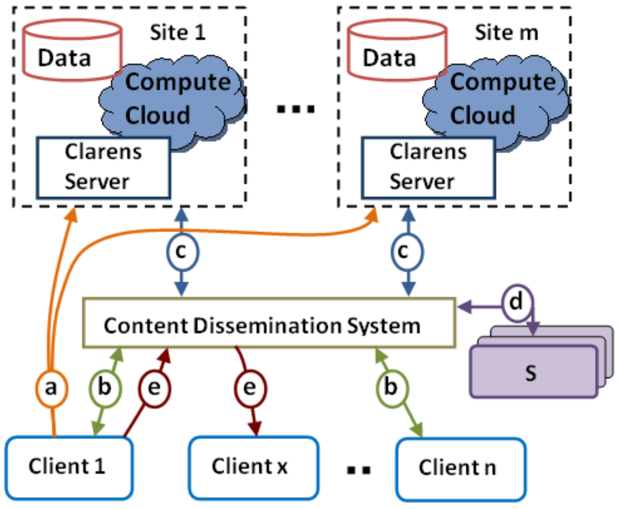
Before presenting the architecture of the system, we would like to identify key feature that we expect from the proposed system. We will use the following terms in our discussion. Master client- the user interface program that a scientist uses to initiate the data analysis task. Client – a user interface program used by other scientists who participate for the collaborative data analysis. The collaborative data analysis framework should support:

1. Functionality to create an experiment workspace to keep track of relevant content such as analysis scripts, configuration files and, if required, resulting data files. It should also support uploading and downloading of these necessary files to and from this space.
2. Execute the analysis in multiple geographic locations where the data is available.
3. Supports multiple models of collaboration such as shared event, shared display, synchronous and asynchronous.
4. Capability to perform different post processing to the received results.
5. Perform the above scenarios in a secure and fault tolerant manner.

A system capable of supporting the above requirements can be architected by incorporating three main software components:

1. A gateway to compute resources i.e. a server capable of operating at a head node of a cluster or grid enabled web service framework. In our implementation we use Clarens server for this purpose.
2. Content distribution framework as the communication layer. Publish/subscribe messaging systems fit best for this purpose as we need to disseminate results and messages to all the participating entities to the collaboration.
3. Software component to keep track of the state of the entire system including the collaborative session and available compute resources. In our implementation we called them agents and we use multiple of them to acquire fault tolerance.

Figure 1 shows the architecture diagram of the system showing the above software components and how they interact to provide a scalable framework for collaborative data analysis.



a. Analysis jobs
b. Discovery of Data and Collaborative Sessions and Shared event messages(Notification of Results)
c. Entity tracking and notification of results
d. Entity tracking, session management and Gossip
e. Shared display messages
s. Redundant set of agents provide services such as session management, entity tracking and data discovery

Figure 1. Architecture of the Proposed Collaborative Data Analysis Framework

Clarens server provides access to data and computation through a web services interface. We use NaradaBrokering for the content dissemination. All the clients use the same user interface program that can operate in multiple settings. Because of this, it should be able to support job submissions to Clarens servers and also support different collaboration models.

At startup time, each Clarens server notifies its availability to the agents using the messaging infrastructure, and the agents start keeping track of the available servers thereafter(not shown in the diagram). We have given complete details of a scalable approach to tracking entities in a distributed system in a secure and authorized manner in Ref [5]. The scientist who initiates the analysis session uses his/her client software (simply master client hereafter) to locate an agent and then uses the agent to locate available Clarens servers. Next she will create an experiment space with the agent specifying the experiment details such as, the data files, the analysis files and other settings related to communication topics. Then she will upload the necessary analysis scripts and configurations files to this experiment space. Both these steps are shown in arrow **b**. Once this is done, other participating scientists can connect to the same experiment using their client software again shown by arrow **b**. After this initialization step the scientist who initiates the analysis (client 1) can start submitting the analysis jobs, shown by arrow **a**, to the available servers which will eventually notify the entities in the collaborative session when the result is available. These notifications are shown by the arrows **c** and **b** in the diagram.

3.1. Collaboration Modes

Above architecture supports most of the typical collaborative modes such as synchronous and asynchronous collaboration each with either shared event or shared display modes.

3.1.1. Synchronous with Shared Event Mode

We support two main approaches to this type of collaboration based on the way the results of the analysis are transferred to the participating entities. One approach is based on the “push paradigm” where the Clarens server publishes the result of an analysis job once it is available. The main advantage of this approach is that the participating clients do not need to connect to the servers to retrieve results. Once they discover the topic over which the results are published the results will be delivered to them as and when they are available at the server. When the results are available at the client it can then perform additional processing on the results if needed or simply save the results. How the security considerations are handled with respect to the above model is discussed in section 3.2.

Another approach that we support is to let the Clarens server publishes the location of the result of a particular analysis job once it is completed. Once this location is received by the participating entities they should connect to Clarens servers and retrieve the results. This approach is similar to "pull paradigm" except it does not require a busy waiting at the client side. This method is more effective when the results of analysis jobs are fairly large in size. The server does not need to partition the result and publish them to the clients; instead client programs can retrieve them from the servers and perform further processing if necessary.

The main benefit of the shared event mode of collaboration is that the participating clients can perform additional processing to the received results without limiting to the post processing functions defined by the master client at the setting up of the experiment.

3.1.2. Synchronous with Shared Display Mode

The control of the shared display mode of collaboration is done mainly by the master client. Once a master client receives a result from the server it performs the necessary post processing such as fitting and plotting functions. If there is a change in his user interface screen then the client program capture a screenshot of it an publishes it using the NaradaBrokering. These notifications are received by the collaborating entities who are only interested in seeing the results of an experiment without actively participating on further processing them (shown by arrows **e**). The client program that these scientists use will simply render the images sent to them by the master client. This approach has many advantages as it simply expand mechanisms on distributing the results of a particular analysis. The post processing necessary at the client program is simply rendering the images. Another possibility is to upload the images to a web server so that they are publically available.

3.1.3. Asynchronous Collaboration

Although a collaborating session is more interactive when all the participants are participating to it in real-time (in synchronous fashion), getting all the interested participants to participate in a session at right time will be a hard problem. Especially when they are participating from around the globe locations those are in different time zones. Therefore, we should expect to support participants who come late to a collaborative session or participants who needs only to see the results of an already ended collaborative session. Our architecture supports both the above type of collaboration in the following manner. When a master client creates a collaborative session with an agent the agent starts keeping track of the collaboration session. It keeps the book keeping information such as the communication topics under which various events related the experiment are disseminated. Nardabrokering's reliable delivery feature [6, 7] guarantees that once a client is subscribed to a particular topic he can retrieve all the events that have been published to the same topic from the creation of the topic. To eliminate the possibilities of overlapping topic names master client appends a 128 bit long universally unique identifier to each topic it creates for an data analysis session .If a scientist joins collaborative session after it has been started or after it is finished, she can simply retrieve the events related to that experiment. Here also, she can either retrieve results events as they were and perform further processing of her own or simply see the screenshots published by the master client. With the former method she will be able to see the results as if it is been conducted now.

3.2. Security

The above collaborative infrastructure spans across multiple domains of control and hence the security aspect of the framework is very important. The Clarens Server provides authentication [8] via the Public Key Infrastructure, which is based on the X509 certificates issued by a trusted Certificate Authority, and authorization via a *gridmap* file similar to other grid frameworks such as the GlobusToolkit [9]. Communication between the clients and different Clarens Servers uses transport layer security using SSL. This ensures that only the authorized scientists can perform analysis jobs using Clarens. However, since the system uses publish/subscribe messaging as the main medium of communication we need to guarantee that only the authorized entities will be able to discover topics and publish data to those topics. In a companion paper S. Pallickara et al. [10] have given a detailed description of NaradaBrokering's secure and authorized end-to-end delivery capability of streams. This will ensure that only the authorized entities can create experiment spaces in an agent and publish or subscribe to topics used.

3.3. Fault Tolerance

The proposed architecture does not have any central control point such as a "portal" and hence poses no threat of a single-point-of-failure. The messaging infrastructure can be configured as a set of broker network which supports fault tolerance [7]. Agents keep track of the ongoing experiments and other bookkeeping information relevant to the experiments. We incorporate a discovery mechanism for agents wherein the client is allowed to discover an available agent based on certain criteria such as response time. Once such an agent is discovered, the client uses that agent to run the experiments. Agents also use a gossip protocol to synchronize the experiment's metadata between them: this allows the system to tolerate individual agent failures. Clarens server failures during the execution of a data analysis task can cause an incomplete results set. However, the composability nature of the data analysis allows users to restart the analysis in a different Clarens server for the data files that have not been analyzed and combine the results from that point on provided that the same data is available in the new location as well. In our current architecture user intervention is required in selecting a new Clarens server and also for restarting the analysis.

4. IMPLEMENTATION

We selected a particle physics data analysis task as our main use case and built a proof of concept implementation to see how the overall system works. Physicists at Caltech were using Clarens server and ROOT to perform various data analysis tasks relating to particle physics. ROOT was the desired data analysis framework for particle physicist and the Clarens server also support execution of data analysis functions written in ROOT. However, their use of these technologies did not help them to collaborate over an analysis task or process data available at different geographic locations. We were able to burrow their analysis scripts and demonstrated that our system can perform the same data analysis in collaborative and much more scalable manner then how they use it earlier.

ROOT also comes with a C++ interpreter named CINT [11] which can be used for rapid prototyping. Thus we also decided to use CINT as our user interface programming language. Therefore our implementation uses ROOT in two distinct tasks:

1. Object oriented data analysis framework of which the analysis functions are written.
2. Language for developing the user interfaces.

With the decision of implementing the user interfaces using ROOT we faced the major challenge in integrating three software systems that are completely heterogeneous with respect to the implementation details. Integrating NaradaBrokering written in Java with user interfaces written in ROOT and Clarens written in Python is one of the major challenges we faced. The communication medium of our architecture is a publish/subscribe messaging infrastructure and hence it should be accessible to all the entities in the system irrespective of the language they were written. We implemented a publish/subscribe client in C++ to cater the above and implemented the necessary changes to Clarens in Python. This allow both Clarens and the user interfaces written in ROOT to access the publish/subscribe capabilities of NaradaBrokering. During the first phase of the work we exclude the implementation of agents and we are currently working on completing the implementation to comply with the proposed architecture. Figure 2 shows the user interface of the client program that a physicist can use for collaborative data analysis.

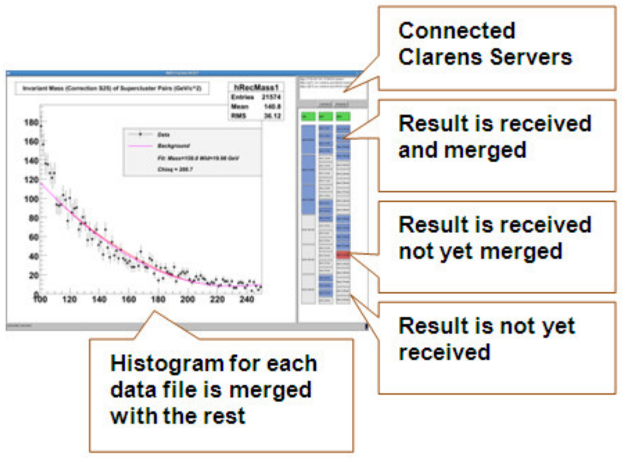

Connected Clarens Servers
Result is received and merged
Result is received not yet merged
Result is not yet received
Histogram for each data file is merged with the rest
hRecMass1

Figure 2. User Interface of Proof of Concept Implementation

Top right most corner of the image shows the available Clarens servers for the data analysis tasks. The compute capabilities used by these servers are completely depend on the way they have configured at each location. Once user selects servers and presses a button to connect to them, the user interface retrieves names of the available data files and shows them to the user. In our current implementation these names are retrieved directly from Clarens servers. Once user selects an appropriate number for grouping the data files to computing jobs (we call them rootlets) the user interface starts submitting these jobs to the appropriate servers. This step happens only in the master client. For the rest of the participants once connected to a collaboration session they simply wait till the arrival of results.

At the end of processing each rootlet the respective Clarens servers notify the clients publishing the results. Once this is received the user interface performs the necessary post processing. The cells shown at the right hand side of the screenshot (Figure 2) are the individual analysis jobs. The cells are color coded in the GUI to reflect the different stages of the processing of the corresponding analysis jobs. In our use case the size of a single data file is 33MB and we group multiple of those files for a single execution unit (rootlet) for analysis. The resulting histogram of an analysis job is nearly 9KB in size. Once a histogram is received by the user interface it merges the histogram with the existing data and executes a “fitting” function to fit a curve to the available data and finally it updates the current result displayed in the canvas.

5. RESULTS

We performed several benchmarks using our proof of concept implementation. First we measured the time for various execution tasks in our framework. Figure 3 highlights the key matrix related our use case.

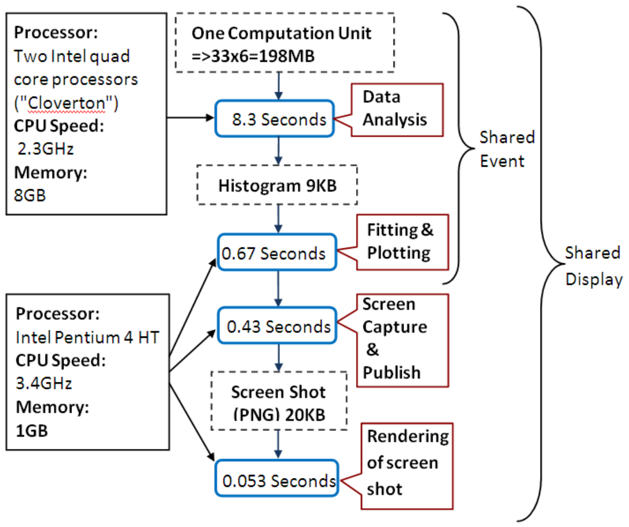
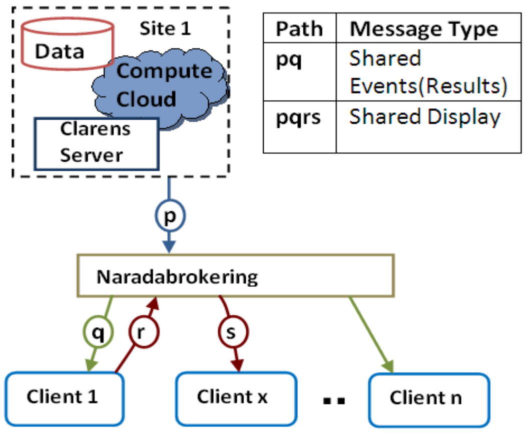


Figure 3. Main Processing Tasks and their Execution Times

After identifying key execution times we measured the propagation time of events from the end of the data analysis phase till it reaches the shared event clients and then to the shared display clients. We measure this by varying the number of participants to the collaboration session to see the effect of the number of participants to the event propagation time. Figure 4 shows the experimental setup that we used for the above measurements.



Path	Message Type
pq	Shared Events(Results)
pqrs	Shared Display

Figure 4. Benchmark Setup

The results obtained by the above benchmark are shown in the Figure 5 where the event propagation time is shown in log scale. In our architecture at least one participant, typically the master client, should publish screenshots of his client software to support shared display mode of collaboration. The additional cost in capturing the screen and publishing it adds a delay to the shared display events. This is the reason for having a higher event propagation time for shared display type collaboration in the graph. From these results it is evident that the number of collaborators does not cause any performance implications on the event propagation time. Typically these numbers of participants are significantly lower than the scalability and the peak throughputs that can be sustained by NaradaBrokering messaging substrate.

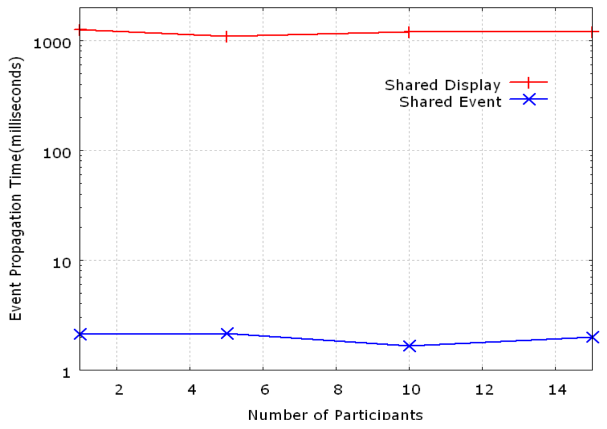


Figure 5. Latencies Involved for Interactions in Different Collaboration Modes

Next we performed a benchmark to see how the network overhead varies with the increase in rate at which the results are generated from the analysis. In this benchmark although, it is meaningless to generate higher number of shared display events such as 200 images per second we simulated it to see the network overhead caused by such higher event rates. Figure 6 shows our results.

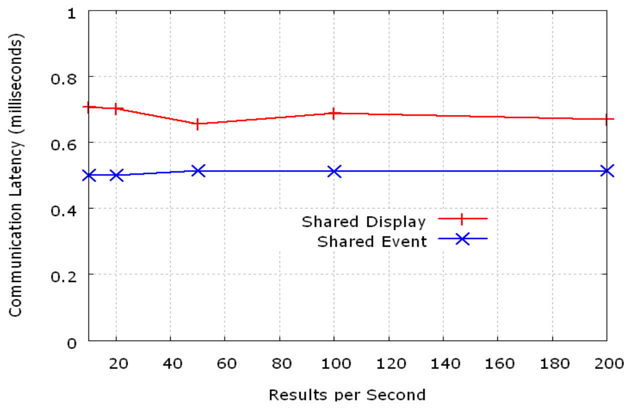


Figure 6. Network Overhead for Collaborative Interactions under Varying Results Stream Rates

In our use case shared display events (20KB) are larger than the resulting histograms (9KB) from sub analysis and hence transferring images incur slightly higher network overhead. However, this can change drastically depending on the analysis. If the size of the results (in its raw format) is larger than the screenshot then these values may change. Also converting output to an image may reduce the resolution of the results and hence the scientists may prefer raw data (share events) over a screenshots (shared display). Therefore which collaborating mode we should use entirely depends on the characteristics of the underlying data analysis task.

6. RELATED WORK

Collaboration among various entities is a well studied field in both industry and the academic communities. Many of proprietary collaborative frameworks such as WebEx [12], Windows Meeting Space [13] and Anabas [14] are all focused on sharing multimedia content over a collaborative sessions. These include audio, video streams, desktop sharing, online meetings, collaborative whiteboards and presentations. Academia also has similar set of applications focusing on data sharing and multimedia collaborations. EVO [15] (the successor of VRVS) is a tool from Caltech which supports instant messaging, video conferencing and desktop sharing functionalities. These functionalities make them essential software components for interactive collaborative sessions where group of people can exchange data, ideas and suggestions. However, use of these systems for sharing real-time results of scientific data analysis is not straight forward where the focus is more towards dissemination of content irrespective of the format of the content.

The Data Trubine[16] presented by S. Tilak et.al. is a content dissemination framework for scientific and engineering applications. They claim that the many content dissemination systems based on messaging architectures do not suit well for scientific data dissemination. However, we argue that the content dissemination should be decoupled from the complexities of the specific data type used by the scientific application. In our implementation we used NaradaBrokering, a publish/subscribe messaging framework, to transfer information relating to the collaborative sessions as well as data files generated during the analyses. Higher level APIs provide the specific level of abstractions for the data that the system transfers. For example, when an analysis job is completed we can either publish the location of the output or the output itself using Naradabroking. Real-time Data Viewer (RDV) [17] is an interface for viewing real-time, synchronized, streaming data from an equipment site. RDV can also be configured to receive streams from the Data Trubine server. This setup is the only close implantation that we found similar to our work. However, RDV supports only the time series data from equipments where as in our architecture we pose no limitation on the way the scientist can process the received data. Collaborating participants can receive any form of data depending on the analyses task and they can perform various posts processing on these data as well.

Portal frameworks such as OGCE[18], uPortal[19] and GridSphere[20] support collaboration by providing a coherent framework for sharing data, compute resources and services to its participants. The web browser based interface allows minimal software for accessing data and services provided by portals and also they supports portlets for monitoring and visualizing data streams generated by scientific applications or sensors. Our user interface can be made into a portlet so that the participants can see the result on the portal.

V. Watson presents [21] design criteria for scientific collaborative applications where he discusses different collaborative modes such as synchronous and asynchronous collaboration.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented a framework for collaborative scientific data analysis. The architecture differs from typical collaborative supports provided by applications focusing on delivering multimedia content to the participants. Our architecture integrates processing entities, content dissemination sub system, session management entities and end users into a single framework. Participants can collaborate over data analyses where the results of a particular analysis is relayed to all the participants either in its raw format or as screenshots, produced after post processing, in real-time. We have successfully converted a non collaborative application used by particle physicist researchers at Caltech into a collaborating application using our architecture.

Currently the user interfaces of our application is implemented in ROOT's interpreted language CINT which is the de-facto standard for data analysis toolkit in particle physics research. However, R[22] is an another data analysis toolkits used by many scientific data analyses. Supporting multiple data analysis toolkits to expand the usability of our implementation is one of the main future works. Although our framework can withstand failures of agents or individual collaborative client applications, failure of Clarens servers results an incomplete results set. The composability feature of the data analysis allows restarting of the experiment only for the failed server as the analysis does not depend on the order in which the results of sub analyses are combined. The framework requires user intervention in selecting the same data set in a different site (if available). As future work we would like to explore the possibilities of using the agents to automatically identify the available Clarens servers attached to the same data set and restart the analysis without user intervention.

REFERENCES

[1]BOINC – Berkeley Open Infrastructure for Network Computing, http://boinc.berkeley.edu/

[2] The Clarens Grid-Enabled Web Services Framework, http://clarens.sourceforge.net/

[3] The NaradaBrokering Project @ Indiana University, http://www.naradabrokering.org/

[4] ROOT - An Object Oriented Data Analysis Framework, http://root.cern.ch/

[5] S Pallickara, J Ekanayake, G Fox: A Scalable Approach for the Secure and Authorized Tracking of the Availability of Entities in Distributed Systems, IPDPS 2007: 1-10.

[6] Shrideep Pallickara, Geoffrey Fox, Beytullah Yildiz, Sangmi Lee Pallickara, Sima Patel and Damodar Yemme, On the Costs for Reliable Messaging in Web/Grid Service Environments, Proceedings of the 2005 IEEE International Conference on e-Science & Grid Computing. Melbourne, Australia.pp 344-351.

[7] Shrideep Pallickara, Hasan Bulut, Geoffrey Fox, Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems, Proceedings of the 4th IEEE International Conference on Autonomic Computing.

[8] Steenberg et al .The Clarens Web Services Architecture., Proceedings of CHEP2003, La Jolla, Paper MONT008, 2003.

[9] The Globus Toolkit, http://www.globus.org/toolkit/

[10] S Pallickara et al: A Framework for Secure End-to-End Delivery of Messages in publish/Subscribe Systems, GRID 2006: 215-222.

[11] CINT – The CINT C/C++ Interpreter, http://root.cern.ch/twiki/bin/view/ROOT/CINT

[12]WebEx - Cisco Web Meetings and Collaboration Solutions, http://www.webex.com/

[13]Windows Meeting Space, http://www.microsoft.com/windows/products/windowsvista/features/details/meetingspace.mspx

[14]Anabas Inc., http://www.anabas.com/

[15]EVO Collaboration Network, http://evo.caltech.edu/

[16] Sameer Tilaky, Paul Hubbardy, Matt Millerz,Tony Fountainy ,The Ring Buffer Network Bus (RBNB) DataTurbine Streaming Data Middleware for Environmental Observing Systems, at third IEEE International Conference on E-Science and Grid Computing, Bangalore, India (December 2007).

[17] RDV-Real-time Data Viewer http://it.nees.org/software/rdv/

[18]OGCE -he Open Grid Computing Environments Portal and Gateway Toolkit, http://www.collab-ogce.org

[19]uPortal, http://www.uportal.org/

[20]Gridsphere- Gridshpere Portal Framework, http://www.gridsphere.org

[21] V. Watson, Supporting Scientific Analysis within Collaborative Problem Solving Environments, Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9, p.9032, January 03-06, 2001

[22]R – R Project for Statistical Computing, http://www.r-project.org/