# A Collaborative Framework for Scientific Data Analysis and Visualization

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

## ABSTRACT

*Human interpretation is a common practice in many scientific data analyses. After the data is processed to a certain extent, the remainder of the analyses is performed as a series of steps of processing and human interpretation. Many large scientific experiments span multiple organizations, therefore, both the data and the teams involved in these experiments, are distributed across these organizations. When the focus of an analysis is to extract new knowledge, collaboration is a key requirement. Real time or near real-time collaboration of expertise, on scientific data analyses, 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:** Collaboration Frameworks, Collaborative Distributed Systems, Scientific Computing.

## 1. INTRODUCTION

The final step of most scientific data analyses is human interpretation. Experts in 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, since its goal is to bring the expertise scattered all over the world to a single data analysis session. The typical steps of a large-scale scientific data analysis are: (i) Locate the data from experiments (typically a large volume of data). (ii) Execute various analysis functions on this data. (iii) Further process the analyzed data to produce graphs or simulations. (iv) Interpretation by scientists.

The collaborative approach yields tremendous benefits to the final interpretation, since it provides a framework for different experts to participate in the final step of the processing. Typically, these analyses are performed in a server cluster, computational grid, or in a group of voluntary participants' computers, as in the case of many projects under Berkeley Open Infrastructure for Network Computing[2]. These analyses may produce a different number of outputs (results), and these outputs may be generated at different time intervals. These factors decide the way a collaborative session can be set up for the interpretation of data produced by an analysis. If the analysis phase has multiple steps, and each phase takes fairly less time, then synchronous collaboration would be the right model. On the other hand, if the analysis takes a longer time to complete, then it is better to run the analysis first, and collaborate on the interpretation of the results in an asynchronous manner. In either case, the sharing of results can be done as shared displays or shared events.

The shared display model fits best for simply sharing results. Therefore, the scientists who only wanted to see the results of an ongoing experiment can see them without further processing at their sites using this model. The shared event model, on the other hand, permits the scientists to participate in an experiment actively. They can either, perform further processing such as fitting models on the received events or simply merge 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 analyses, the composability property, which is the ability to perform the data analysis as a collection of sub data analyses that can be executed concurrently, and thereby producing the result by merging the outputs of these sub analyses, is often exploited to achieve scalability. This type of analysis produces a considerable amount of sub results, and by enabling collaboration for such analysis, the collaborating participants will be able to see the results getting shaped as and when the sub results are available. This will be a very dynamic collaboration if the number of sub results is high, and if they are produced at higher rates.

High Energy Physics (HEP) data analysis is an example of composable data analysis. Typically, the aim of such data

analysis is to identify a particular type of events within the millions of events, generated by HEP experiments. The result of such analysis would be a histogram of possible events. The task of the analysis is to go through all the events available, and identify a particular set of events. We can easily break down 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 being 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 scientists participating in a collaborative session will not be from only one organization or a single administrative domain. It will be a global participation. Therefore, supporting such collaboration requires careful attention to the security aspects of the framework for collaboration. How to authenticate various participants and how to authorize them to perform various activities are some of the questions that should be answered by the collaborative framework.

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

Typical collaborative frameworks in the scientific community fall into two categories. First is the framework for sharing data and heterogeneous distributed resources and, second is the framework for supporting collaboration among multiple participants. Most grid portal frameworks such as OGCE[9] and GridSphere[8] fall into the first category, and form a more static version of collaboration. Sharing data and resources among the participants is a key strength in these technologies. Software such as NEESGrid[6] extends the goal of sharing data and resources into near real-time tele-observation of sensor and experimental data and also to tele-operation of remote equipment control systems, allowing a dynamic form of collaboration among the participants. The collaborative frameworks such as Enabling Virtual Organizations(EVO)[4] and Anabas[1] fall into the second category. They focus on developing tools for collaboration, such as shared white boards, audio video conferencing, recording, playback etc. Although the above systems support different aspects of collaborative data analysis, there is no support for real-time (synchronous) collaborative data analysis in either major Grid or Web 2.0 systems.

Our architecture provides a solution to both data analysis and real-time or near real-time sharing of results for a set of collaborative participants. The support for executing the data analyses in a close network proximity to the data minimizes the overall data movement, and thereby favors scientific data analyses involving large volumes of distributed data. The multiple modes of collaboration, supported by the proposed framework enable the scientists to join a collaborative data analysis session using a model of their choice. For example, a scientist with limited computer power can decide to join a collaborative session using the shared display mode. In addition, a scientist who joins in late to a collaborative session can retrieve the results (events) as shared events or shared displays. This form of collaboration is best suited for the final steps of most scientific data analyses, which focus on extracting knowledge from the processed data. The goal of the proposed framework is to support this type of collaboration with secure and fault tolerant manner.

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

## 2. ARCHITECTURE

Before presenting the architecture of the proposed framework, it is important to identify the key features, which are expected from the proposed framework. 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 in the collaborative data analysis. The collaborative data analysis framework should support:

- Functionality to create an experimental session to keep track of relevant content such as analysis scripts, configuration files, and communication topics.
- Execution of the analysis in multiple geographic locations where the data is available.
- Multiple modes of collaboration such as shared events, shared displays, synchronous, and asynchronous collaboration.
- Capability to perform different post processing on the results received.
- Performing the above scenarios in a secure and fault tolerant manner.

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

1. A gateway to compute resources i.e. a server capable of operating at the head node of a cluster or as a grid enabled web service framework.
2. Content distribution framework as the communication layer. Publish/subscribe messaging systems are best suited for this purpose, since we need to disseminate results and messages to all the participating entities within the collaborative session.
3. Software components to keep track of the state of the entire system, including the collaborative sessions and the available compute resources. In our implementation, we have called these agents, and we use multiple instances of these to sustain failures.

## 2.1. Software Subsystems

We use Clarens server[7], developed at Caltech, for the data analysis framework and NaradaBrokering[10], a publish/subscribe messaging substrate, as our communication layer.

### 2.1.1 Clarens Server

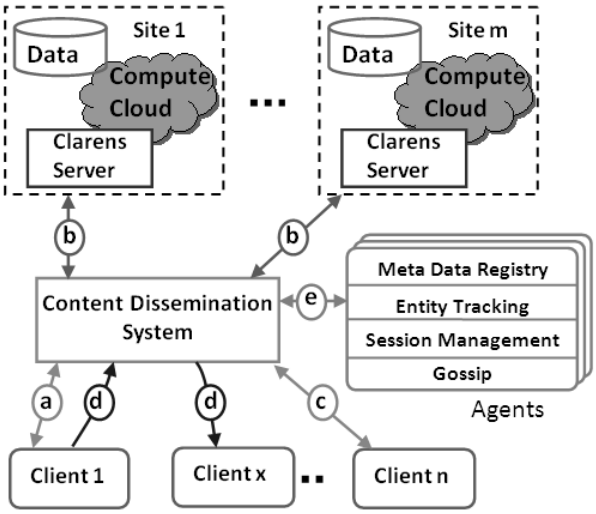
Clarens is a grid enabled web service framework, implemented in Python, which supports most common web service protocol stacks comprising HTTP and SOAP/XML-RPC with SSL/TLS encryption and X.509 certificate-based authentication. Although the server implementation of Clarens is Python-based, it provides client libraries for other languages such as Python, Iguana, JavaScript, and most importantly, the C++ based interpreted language supported by the ROOT Analysis framework[17]. Support for all the above features as well as the integrated support for ROOT makes Clarens a key framework for various scientific data analyses.

### 2.1.2. NaradaBrokering

NaradaBrokering is a content distribution infrastructure. 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 on the size, rate, or scope of the interactions encapsulated within events or the number of entities present in the system. In addition, it incorporates support for wide verity of transport protocols making it an ideal communications infrastructure for heterogeneous distributed systems.

## 2.2. Collaborative Data Analysis Framework

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




| Arrow | Messages |
|---|---|
| a | Data Discovery , Analysis Job Submission, and Results Notification |
| b | Entity Tracking, Analysis Job Retrieval, and Results Notification |
| c | Session Discovery and Results Notification |
| d | Shared Display |
| e | Entity Tracking, Session Management, and Gossip |

**Figure 1. Architecture of the Proposed Framework**

In the above architecture, the Clarens server acts as the gateway to data and compute resources at a particular site. Entities in our architecture, clients, servers, and agents, communicate using NaradaBrokering. At startup, each Clarens server notifies its availability to the agents, and the agents start keeping track of the available servers thereafter. We have given complete details of a scalable approach to tracking entities in a distributed system in a secure and authorized manner in Ref[11]. The scientist who initiates the analysis session uses the client software (simply master client hereafter) to locate an agent and then uses the agent to locate available Clarens servers and data. Next, the scientist proceeds by creating a session for the experiment with the agent, specifying the experiment details such as the data files, the analysis files, and other settings related to communication topics. These steps are shown by the arrow **a**. Once a session is created, other participating scientists can connect to the same session, shown by arrow **c**, using their client software. After this initialization step, the scientist who initiates the analysis (client 1 in Figure 1) informs the agent to start the analysis. The Agent notifies the Clarens servers about the availability of the analysis jobs, and the Clarens servers starts processing them (shown by the arrow **b**). This approach decouples the communication logics associated

with the computational servers from the client implementation. Once an analysis job is completed, the Clarens servers notify the results to the collaborating clients (shown by the arrows **a** and **c**).

## 2.3. Collaboration Modes

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

### 2.3.1. Synchronous with Shared Event Mode

We support two main approaches within 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 servers. When the results are available at the client it can then perform additional processing on the results, which is a main benefit of this approach, if needed or simply save the results.

Another approach that we support is to let the Clarens servers publish the location of the result of a particular analysis job once it is completed. These can be the inputs for another level of processing (if required). Although we have not yet explored this, the above feature enables a hierarchical data analysis using our framework.

### 2.3.2. Synchronous with Shared Display Mode

In the shared display mode of collaboration, the master client handles the control of the collaborative session. When the master client receives a result from a Clarens server, it performs the necessary post processing such as fitting and plotting functions. If there is a change in the user interface, then the client program captures it, and publishes it using the NaradaBrokering. Collaborating entities who are only interested in seeing the results of an experiment without actively participating in processing these results (shown by the arrow **d**) receive these events. The client programs, used by scientists, will simply render the images sent to them by the master client. This approach has many advantages, since it simply expands the number of ways of distributing the results of a particular analysis. The only post processing required at the client program is simply rendering these images. Another possibility is to upload the images to a web server so that they can be publicly available.

### 2.3.3. Asynchronous Collaboration

Although a collaborative session is more interactive when all the participants are participating in real-time (in synchronous fashion), getting all the interested participants to participate in a session at the right time will be hard. This is especially true when they are participating from locations around the globe that are in different time zones. Therefore, we should expect to support participants who come late into a collaborative session or participants who need to only see the results of an already terminated collaborative session.

Our architecture supports both these types of collaborations in the following manner. When a master client creates a collaborative session with an agent, the agent keeps track of the collaborative session. It stores information such as the communication topics under which various events related the experiment are disseminated, analysis functions, and the location of the data files. Nardabrokering's reliable delivery feature[12,13] guarantees that once a client is subscribed to a particular topic it can retrieve all the events that have been published to the same topic from the creation of that topic. To eliminate the possibilities of overlapping topic names, master client appends a 128 bit long universally unique identifier to each topic that it creates for a data analysis session. If a scientist joins a collaborative session, after it has been started, or after it has finished, the scientist can simply retrieve the events related to that experiment. Here also, the scientists can retrieve results events either as they were and perform further processing of their own or simply see the screenshots published by the master client. With the former method, the scientists will be able to see the results as if it is being performed in real-time.

## 2.4. Security

The above collaborative infrastructure spans multiple domains of control, and therefore the security aspect of the framework is a very important consideration. The Clarens server provides authentication[18] via a 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[5]. Since our framework 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.[14] have given a detailed description of NaradaBrokering's framework for the secure and authorized end-to-end delivery of streams. Clarens server uses the user credentials available in the publish/subscribe messages for the authentication and authorization. This ensures that only the authorized scientists can perform

analysis jobs using Clarens. This will ensure that only the authorized entities can create experimental sessions with an agent, and publish or subscribe to topics used.

### 2.5. 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[13]. 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 result set. However, the composability property of the data analysis allows users to restart the analysis in a different Clarens server for the data files that have not been analyzed if 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 for restarting the analysis.

## 3. 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 the Clarens server and ROOT to perform various data analyses tasks relating to particle physics experiments. ROOT was the data analysis framework for the particle physicist, and the Clarens server also supports 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 to process data available at different geographic locations. We have access to their analysis scripts, and demonstrated that our system can perform the data analysis in a collaborative and much more scalable manner than before.

ROOT also comes with a C++ interpreter named CINT[3], which can be used for rapid prototyping. Thus, we also decided to use CINT as our user interface programming language. With this decision, our implementation uses ROOT as an object oriented data analysis framework, of which the analysis functions are written, and as a language for developing user interfaces.

NaradaBrokering is the main communication medium in our architecture, and hence, it should be accessible to all the entities in the system. Integrating NaradaBrokering written in Java with user interfaces written in ROOT and with Clarens server written in Python is one of the main challenges that we encounter. We implemented a publish/subscribe client in C++, and incorporated the necessary changes into Clarens in the Python language. This allows 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.

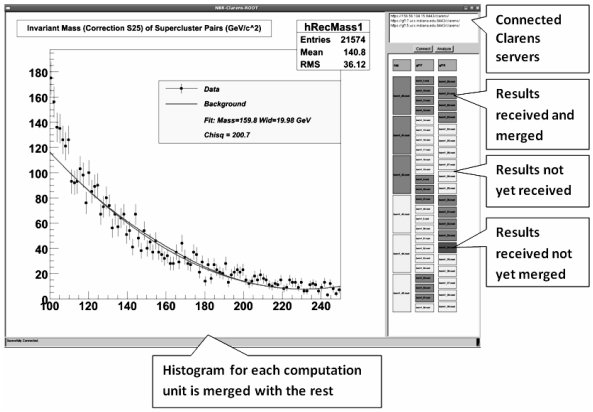



**Figure 2. User Interface of Proof of Concept Implementation**

The top right corner of Figure 2 shows the available Clarens servers for the data analysis tasks. The compute capabilities used by these servers depend on the way they have configured at each location. Once a 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. The rest of the participants, once connected to a collaboration session, simply await the results.

At the end of processing each rootlet, the respective Clarens servers notify the clients publishing the results. Once an event is received, the user interface performs the necessary post-processing. In our use-case, the size of a single data file is 33MB, and we group multiple of these files for a single execution unit (*rootlet*) for analysis. The resulting histogram of an analysis job is nearly 9KB in size. The cells shown at the right hand side of the screenshot (Figure 2) are the individual analysis jobs. Once a histogram is received by the user interface, it merges the histogram with the existing histograms and

executes a "fitting" function to fit a curve to the available data, and finally, it updates the current result displayed in the canvas.

## 4. 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 shows the experiential setup that was used, while Figure 4 highlights the key metrics that we obtained.

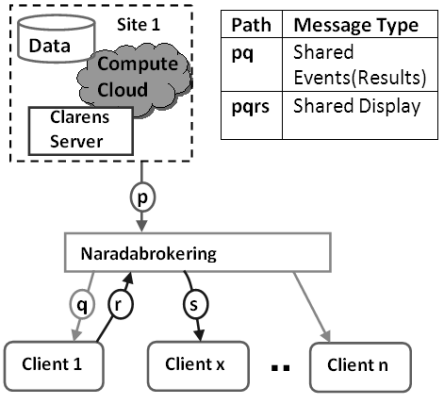



**Figure 3. Benchmark Setup**

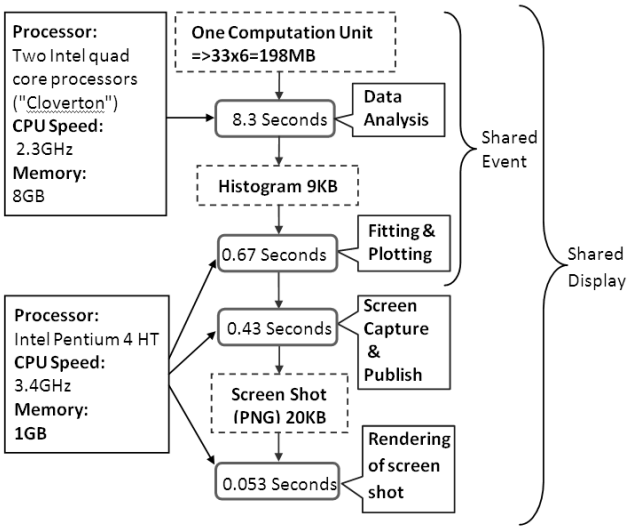



**Figure 4. Main Processing Tasks and their Execution Times**

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 from there on to the shared display clients. We measured this by varying the number of participants within the collaboration session to see the effect of the number of participants on the event propagation latencies.

The results obtained by the above benchmark are shown in the Figure 5 in which the event propagation time is shown in a logarithmic scale. In our architecture, at least one participant, typically, the master client should publish screenshots of the client software to support the 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.

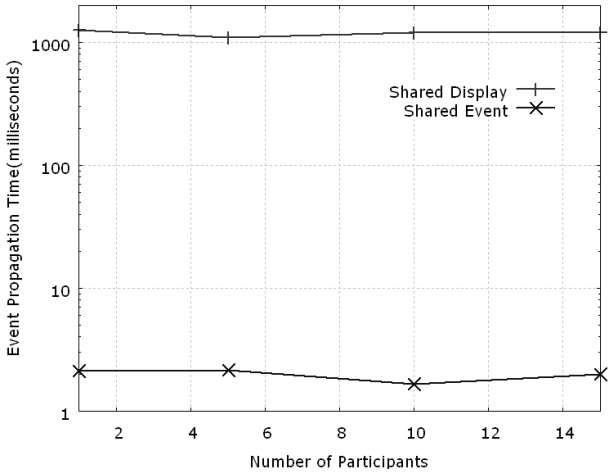



**Figure 5. Latencies Involved for Interactions in Different Collaboration Modes**

Next, we performed a benchmark to see how the network overhead varies with the increase in the rate at which the results are generated from the analysis. We also went on to generate a high number of shared display events such as 200 images per second to see the network overhead caused by the higher rates. Figure 6 shows our results.

In our use case, shared display events ( 20KB) are larger than the resulting histograms ( 9KB) from sub analysis and hence transferring images incur a 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 screenshots then these values may change. Converting the output to an image may reduce the resolution of the results and hence scientists may prefer raw data (share events) to screenshots (shared display). Therefore, the collaboration mode we should use depends entirely on the characteristics of the underlying data analysis task.

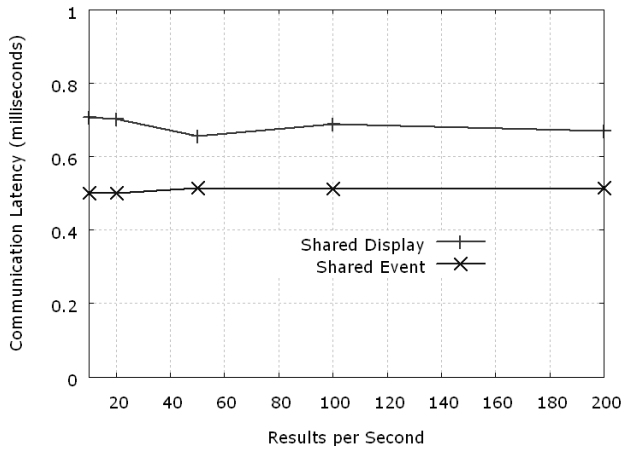



**Figure 6. Network Overhead for Collaborative Interactions under Varying Results Stream Rates**

## 5. 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[22], Windows Meeting Space[23] and Anabas[1] are all focused on sharing multimedia content over a collaborative session. 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[4] (the successor of the Virtual Room Videoconferencing System) 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 a 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 Turbine[19] 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)[16] 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 Turbine 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, uPortal[20] and GridSphere 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 support 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.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a framework for collaborative scientific data analysis. The architecture differs from typical collaboration support 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 stand-alone application used by particle physicist researchers at Caltech into a collaborative application using our architecture.

Although our framework can withstand failures of agents and individual collaborative client applications, failure of Clarens servers results in an incomplete result 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 data set at a different site (if available). In our future work, we will explore the possibility of using the agents to automatically identify the available Clarens servers attached to the same data set and restart the analysis without user interventions. 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[15] 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 also one of our research goals.

## ACKNOWLEDGEMENT

This research is supported by a grant from the National Science Foundation's Division of Earth Sciences project number EAR-0446610, and from a Department of Energy STTR grant (grant number DE-FG02-07ER86301).

## REFERENCES

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

[2] Anderson, D.P., "BOINC: A System for Public-Resource Computing and Storage," GRID 2004, pp. 4-10.

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

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

[5] Foster, I. and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing, SIAM, Lyon, France, August 1996.

[6] Gullapalli, S., S. Dyke, P. Hubbard, D. Marcusiu, L. Pearlman, and C. Severance, "Showcasing the Features and Capabilities of NEESgrid: A Grid Based System for the Earthquake Engineering Domain," Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004.

[7] Lingen, F.V., C. Steenberg, A. Anjum, F. Khan,H. Newman, A. Ali, J. Bunn, and I. Legrand, "The Clarens Web service framework for distributed scientific analysis in grid projects," Proceedings of the 2005 International Conference on Parallel Processing Workshops ( ICPPW'05), pp. 45 – 52.

[8] Novotny, J., M. Russell, and O. Wehrens, "GridSphere: a portal framework for building collaborations," Concurrency and Computation: Practice & Experience, Vol. 16, No. 5, 2004, pp.503-513.

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

[10] Pallickara, S., G. Fox, "NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids," Middleware 2003, pp. 41-61.

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

[12] Pallickara, S., G. Fox, B. Yildiz, S.L. Pallickara, S. Patel, and D 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.

[13] Pallickara, S., H. Bulut, and G. Fox, "Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems," Proceedings of the 4th IEEE International Conference on Autonomic Computing.

[14] Pallickara, S., M. Pierce, H. Gadgil, G. Fox, Y. Yan, and Y. Huang, "A Framework for Secure End-to-End Delivery of Messages in publish/Subscribe Systems," GRID 2006, pp. 215-222.

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

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

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

[18] Steenberg, C.D., E. Aslakson, J.J. Bunn, H.B. Newman, M.Thomas, and F.V. Lingen, "The Clarens Web Services Architecture," Proceedings of CHEP2003, La Jolla, Paper MONT008, 2003.

[19] Tilak, S., P. Hubbard, M. Miller, and T. Fountain, "The Ring Buffer Network Bus (RBNB) DataTurbine Streaming Data Middleware for Environmental Observing Systems," Third IEEE International Conference on E-Science and Grid Computing, India, 2007.

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

[21] Watson, V. "Supporting Scientific Analysis within Collaborative Problem Solving Environments," Proceedings of the 34th Annual Hawaii International Conference on System Sciences HICSS-34, 2001.

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

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