COLLABORATIVE INTERACTIVITY IN PARALLEL HPC APPLICATIONS

Interactive Computational Steering of Grid Applications

M. Riedel[1], W. Frings[1], Th.Eickermann[1], S. Habbinga[1], P. Gibbon[1], A. Streit[1], Felix Wolf[1,2], Th. Lippert[1]

[1] *Forschungszentrum Jülich, Jülich Supercomputing Centre, 52425 Jülich, Germany*
[2] *RWTH Aachen University, Department of Computer Science, 52056 Aachen, Germany*

Abstract Large-scale scientific research often relies on the collaborative use of massive computational power, fast networks, and large storage capacities provided by e-science infrastructures (e.g. DEISA, EGEE, etc.) since the past several years. Especially within e-science infrastructures driven by high-performance computing (HPC) such as DEISA, collaborative online visualization and computational steering (COVS) has become an important technique to enable HPC applications with interactivity and visualized feedback mechanisms. In earlier work we have shown a prototype COVS technique implementation based on the visualization interface toolkit (VISIT) and the Grid middleware of DEISA named as Uniform Interface to Computing Resources (UNICORE). Since then the approach grew to a broader COVS framework. More recently, we investigated the impact of using the computational steering capabilities of the COVS framework implementation in UNICORE on large-scale HPC systems (i.e. IBM BlueGene/P with 65536 processors) and the use of attribute-based authorization. In this paper we emphasize on the improved collaborative features of the COVS framework and present new insights of how we deal with dynamic management of n participants, transparency of Grid resources, and virtualization of hosts of end-users. We also show that our interactive approach to HPC systems fully supports the necessary single sign-on feature required in Grid and e-science infrastructures.

Keywords: Scientific Visualization, Computational Steering, COVS, VISIT, UNICORE

1. INTRODUCTION

Many e-science applications within Grids aim at simulation of a scientific domain-specific problem and *parallel computing* is widely accepted to be an essential tool for the solution of these complex scientific problems. Such simulations of physical, chemical, biological or economical processes provide insights into phenomena that either cannot be addressed by experimental methods or would require experiments that are too expensive or dangerous. Grid applications that use parallel computing techniques for simulations have to use computers with multiple processors that are able to jointly work on one or more specific problems at the same time.

In many parallel simulations, for instance in *computational fluid dynamics (*CFD*)*, astrophysics, or weather and climate research, the tasks of simulation and visualization of the results are usually done as independent steps. First, a researcher performs a simulation for some fixed amount of computational time, and then analyses the computed results in a *separate post-processing step*, for instance by viewing the results in a dedicated visualization application.

On the contrary, several scientists have indicated a higher level of satisfaction by using an *interactive access and control* of their Grid applications during their run-time. That means it is often more efficient and more enlightening, to perform both steps - simulation and visualization - at the same time by also steering certain parameters of applications during run-time. Hence, the scientists are able to observe the intermediate steps during the computation of the simulations (*online visualization*) and can interact with an ongoing parallel simulation to influence its computation (*computational steering*). Some examples where interactive computational steering is used is for parameter space exploration, in engineering applications to kick-start numerous events, calibrations of simulations, and several other use cases.

In this contribution we will highlight certain design concepts of the Collaborative Online Visualization and Steering (COVS) framework and its reference implementation within UNICORE [15], which allows for interactive access to Grid applications. While many work around COVS was published [13, 12, 10, 9, 11], this paper emphasizes on features of how we deal with dynamic management of n participants, Grid transparency in terms of hostnames and ports to satisfy end-user without technical knowledge (e.g. single sign-on).

This paper is structured as follows. Following the introduction the scene is set in Section 2 where we present the evolution of the COVS framework reference implementation in UNICORE and its core building blocks. Section 3 then highlights COVS framework features to seamlessly manage collaborative interactive Grid applications. Section 4 describes two parallel HPC applications that make use of the framework while a survey of related work is described in Section 5. The paper ends with some concluding remarks.

2. COLLABORATIVE ONLINE VISUALIZATION AND STEERING (COVS) FRAMEWORK

The COVS framework enables HPC-based Grid applications with interactivity (i.e. computational steering) and visualized feedback mechanisms. In earlier work [13], we have shown a prototype COVS technique implementation based on the visualization interface toolkit (VISIT) [20] and the Grid middleware UNICORE. VISIT is a communication library that provides a loosely coupled approach between a parallel simulation on the one hand and its visualization on the other.

In principle, VISIT transports specific datasets between the visualization and simulation and thus conveys scientific and additional information such as interactive (steering) commands. Following a broader requirement analysis for COVS [12], we observed that one of the most crucial design goals should be to minimize the load on the steered simulation and to prevent failures or slow operations of the visualization from disturbing the simulation progress on supercomputers. This leads to a design of VISIT where the e-science applications (simulations) act as a client which initiates all operations like opening a new connection, sending data to be visualized or receiving new interactive steering parameters. VISIT provides several components and most notably a VISIT server (integrated in visualizations), a VISIT client (integrated in simulations), and additionally a VISIT Collaboration Server and the VISIT Multiplexer to deal with collaborative scenarios.

With the successful integration of VISIT, our approach grew to a mature COVS framework implementation, which evaluation [10] proved that the approach taken is feasible and provides sophisticated performance. Since then, we observed a trend towards higher degrees of parallelism by employing a larger number of moderately fast processor cores. As a result of these development, supercomputer Grids will have to integrate peta-scale high-end computational resources with many cores in the future, which implies that also our COVS framework is used with new kinds of applications. Therefore, we investigated in [9] the impact of using the interactive computational steering capabilities of the COVS framework implementation in UNICORE on large-scale HPC systems (e.g. IBM BlueGene/P JUGENE with 65536 processors).

More recently, we presented in [11] an extension of the COVS framework with attribute-based authorization capabilities. This was particularly motivated by challenges that arise in geographically dispersed visualization sessions, creating a demand for a more fine-grained authorization based on attributes of end-users such as VO/project membership or different roles (e.g. steerer, collaborator, participant, etc.).

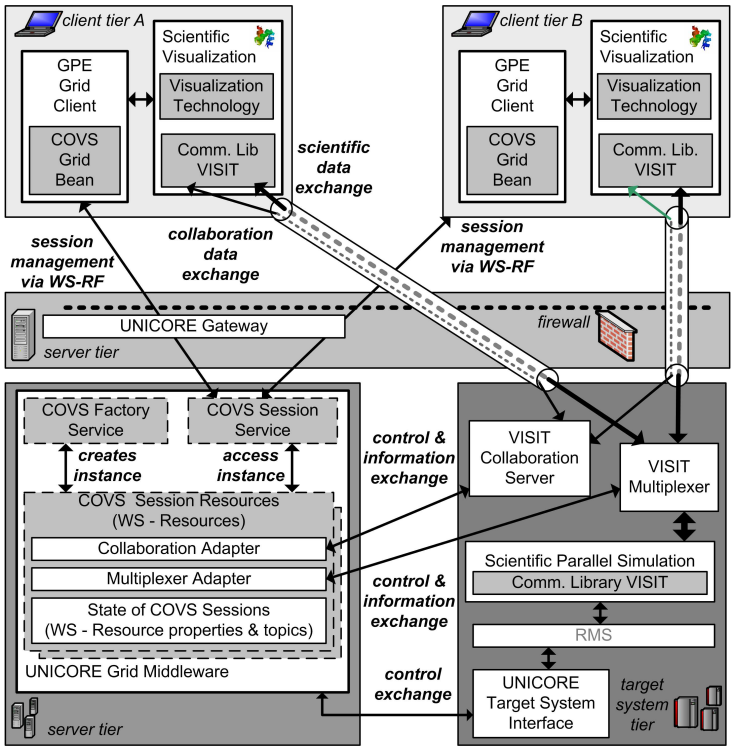

Figure 1. COVS Framework Reference Implementation in the UNICORE Grid middleware.

The current COVS framework reference implementation architecture is shown in Figure 1 [10]. It illustrates a collaborative scenario with two geographically dispersed participants (i.e., client tier A and B). Both run on their machine a scientific visualization that interacts with a COVS GridBean plug-in, which extends the GPE UNICORE Grid client [8].

The Grid client is used to submit computational jobs to the Grid and to access two dedicated COVS services that are compliant with the Web Service Resource Framework (WS-RF) standard [21]. According to the factory pattern of WS-RF, the client is used to call a COVS Factory Service, which creates COVS Session resources that are in turn accessible via the COVS Service. Hence, one instance of such a session resource represents a collaborative visualization session. This session resource can be used to manage different participants by controlling the VISIT Multiplexer and the VISIT Collaboration Server. While the VISIT Multiplexer is responsible to distribute the outcome of one parallel simulation (i.e. scientific data) to n participants, the VISIT collaboration server is used to exchange information (e.g. turn angle of viewpoints) between the n participants. Both are using SSH connections using the strong security features of UNICORE 6 more deeply explained in the next Section.

3. COVS FRAMEWORK FEATURES TO MANAGE INTERACTIVE GRID APPLICATIONS

The previous section introduced the core building blocks of the COVS framework implementation and described that the scientific data, steering data, and the collaboration data is transferred via secured dedicated connections with binary wire encoding using VISIT to achieve satisfactory performance. To ensure fast establishment of bi-directional connections as well as satisfactory performance, the Grid middleware of the COVS framework must allow for interactive access to the Grid resources bypassing the typically deployed resource management systems (RMSs) on supercomputers that do scheduling. In addition, the previous section also illustrated that the Grid application (i.e. parallel simulation) submission and management of collaborative sessions use Web service calls that in terms of the overall performance are non-critical. Nevertheless, the COVS framework implementation exposes an overview of the participants and their connection status during a COVS session to all participants. This includes information about the performance and status of each connection to identify bottlenecks.

It is also important to mention that the applicability of interactive steering depends on the type of the Grid application. The time for one computational step should not take too long in order to avoid long-winded online visualizations (no direct feedback) and to see the impact of steering almost immediately. On the other hand, the time for one computational step should not be too short, in order to give the end-user of COVS the chance to steer the parameters based on the current status, before the next data from the simulation is displayed. In order to allow for steering, the parallel Grid simulation must be parameterized in a way that allows the change of values during its run-time.

Since the SSH connections are critical, we give insights in this section how they are created by still providing single sign-on. Hence, the seamless integration into Grids places the requirement on COVS components to remain single sign-on, which avoids specific logins or tedious password requests.

Also, this section will describe in detail how transparency of Grid resources in terms of hostnames and ports are realized within the COVS framework. Hence, the COVS framework implementation is typically well embedded in a broad Grid or e-science infrastructure by the meaning of hiding the fact that resources are physically distributed. This transparency includes differences in security policies, data representation, and how a resource is accessed when using services of the COVS framework.

Both described fundamental features of COVS are necessary in order to provide high-levels of usability for end-users that have no technical Grid understanding like many e-scientists that are experts in their scientific domain, but are no experts in distributed system technologies.

3.1 Secure Data Transfer and Interactive Access

The COVS framework relies on the usage of a bi-directional connection for secure data transfer between the parallel simulation and the visualizations with low latency. Lessons learned from earlier work [2] in the context of COVS revealed that if interactive steering data is transferred through several Grid components (e.g. Grid services) the overall latency will considerably increase compared to a direct connection. Therefore, the implementation of the COVS framework is based on SSH connections for the transfer of scientific and interactive steering data, also because firewalls often allow access via SSH to the protected systems (i.e. supercomputers) running parallel simulations.

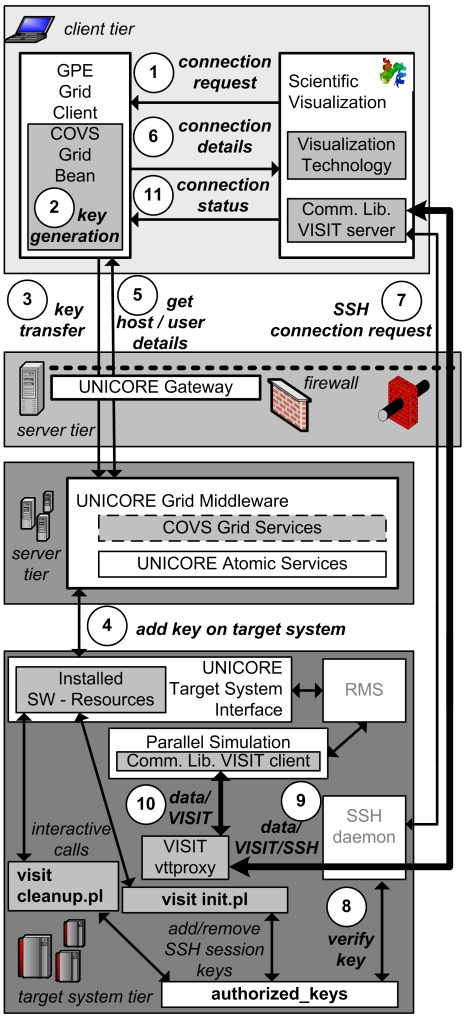

Figure 2. Using interactive calls for the establishment of SSH connections.

Figure 2 illustrates that the visualization makes a connection request to the COVS Grid Bean (1). Afterwards, the COVS Grid Bean generates an SSH session key pair on the fly (2) and transports the public key to the UNICORE Grid site (3). The UNICORE Grid middleware in turn adds the key into the *$HOME/.ssh/authorized_keys* file (4) using a previously installed UNICORE software resource (sw-resource) [15] *visit_init.pl*.

The Grid client gets the resource specific details such as the username, hostname or the location of the VISIT *vttproxy* from this script as well (5). These pieces of information and the key pair are then transferred to the scientific visualization application (6). In particular, this application integrates the VISIT server of the communication library VISIT, which uses the transferred details to make a SSH connection request to the SSH daemon of the UNICORE Grid site (7). The daemon in turn checks whether the used session key is present in the *$HOME/.ssh/authorized_keys* file (8). After approval, the *vttproxy component* of VISIT is started and thus a bi-directional connection can be established (9).

Finally, the parallel simulation uses the VISIT client that sends the data to the *vttproxy* (10), which in turn forwards the data over the secure SSH connection to the visualization using the VISIT protocol. The *vttproxy* acts as a proxy for the visualization located in the same security domain as the simulation. The simulation can still decide when to connect to the *vttproxy*.

After the establishment of the connection, the sw-resource *visit_cleanup.pl* is used to remove the session key from the *$HOME/.ssh/authorized_keys* file. The key transfer and its addition to this file raise the demand for interactive access so that the establishment of the SSH connection is not scheduled via a Resource Management System (RMS) typically deployed on Grid resources. This is realized by using the INTERACTIVE label within the UNICORE configuration for the two sw-resources *visit_init.pl* and *visit_cleanup.pl*.

Furthermore, the usage of the Grid middleware UNICORE ensures that the implementation provides authentication and authorization of users. In particular, the Grid job from the GPE Client, which includes the execution of sw-resources, can only be executed if the user of the Grid client is correctly authenticated and authorized at the enhanced NJS [19] backend of UNICORE. Hence, the implementation gains the benefit of the UNICORE security infrastructure since a non-authorized user is not able to establish the SSH connection and thus can not use it for visualization.

Also, the installation is very lightweight, because both sw-resources are simple perl-scripts that run directly on the supercomputer. Note that the same principle of establishment is also used in collaborative visualization sessions between the multiplexer and several visualizations also using the *vttproxy* component in between.

3.2 Naming Service for Decoupling COVS Components

One of the fundamental ideas of the COVS framework is to hide the complexity of the Grid from its end-users. That leads to the demand that they do not have to deal with hostnames, usernames, or other environment-specific details. This section provides details of how exactly the connections on the network-level between all the different components are realized by still hiding the necessity for hostnames, usernames and ports from end-users. Hence, after the successful establishment of the SSH tunnel and startup of the vttproxies as described in the previous section, the used VISIT components including the VISIT multiplexer, VISIT collaboration server, VISIT servers and VISIT client must exactly know where to send data. In other words, these components require exact hostname and port information form each other.

The key to the understanding of this internal information exchange is a simple naming services provided by the VISIT toolkit named as the Service Announcement Protocol (SEAP) server [20]. This server allows for a decoupling of the parallel simulation and their visualizations. It basically maps a *seapservice:seappassword* to a set of information that comprises a hostname and port. Here, the *seapservice* is used for the identification of a client tier or simulation, while *seappassword* is used for the identification of a COVS session. Hence, it does not deal with authorization based on passwords.

One of the major requirements of the COVS framework is to prevent disturbances of the simulation progress by any online visualizations that are connected to it. Therefore, the parallel simulation itself connects to the VISIT multiplexer component, which in turn needs to connect to multiple visualizations using the vttproxies of each visualization as shown in Figure 3. Hence, the parallel simulation needs to know the hostname and port of the vttproxies that forward the data through the SSH connections to the visualizations.

Another key to the understanding relies in the COVS Grid Bean implementation that interacts with the COVS Grid service, which exposes ongoing COVS session information via WS-Resource properties [21]. In the context of the SEAP server, the COVS Grid Bean is used to set up a COVS session, which leads to the creation of a *seappassword (e.g. svc1)* for this session. This identifier is kept internally within the implementation, instead the end-users see a human readable name for this session which is mapped on the internal identifier.

In the context of Figure 3, some end-user (e.g. client tier C) has already created such as session that is internally identified by *svc1* as the *seappassword*. The human readable name related to this identifier is exposed via WS-Resource properties to the client tier A and B. Therefore, the COVS Grid Bean shows this particular session and both client tier A and B are able to request for participation in this particular session as follows.

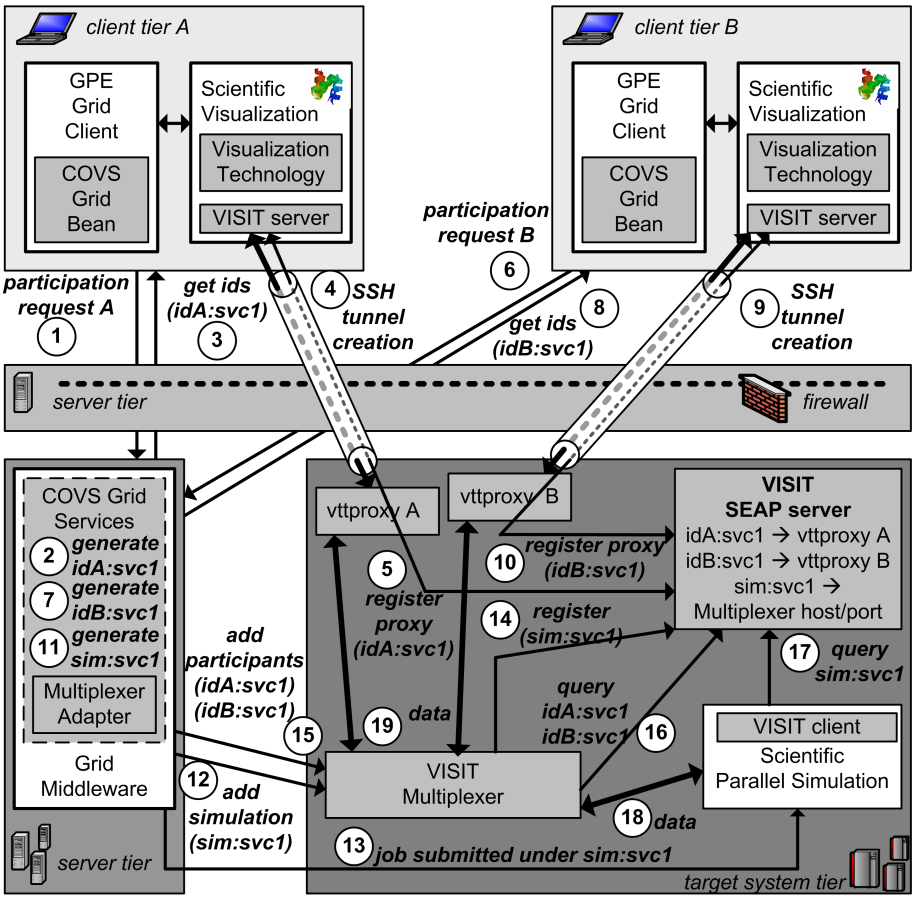

Figure 3. Using the VISIT SEAP server as naming service. Two geographically dispersed participants are dynamically connected using seapservice:seappassword abstractions. SEAP information exchanges are also tunneled via SSH were necessary.

As illustrated in Figure 3, client tier A makes a participation request for a specific session at the COVS Grid service using Web service message exchanges (1). Internally, this session is identified by *svc1*. Next, the COVS Grid service generates a *seapservice idA* for this participant (2). Hence, the combination *idA:svc1* identifies a particular user that participates in a specific COVS session. This combination is send back to the COVS Grid Bean (3). Afterwards, the COVS Grid Bean creates the SSH tunnel using the mechanisms described in the previous sections (4). In order to publish the location of the started vttproxy A, the combination *idA:svc1* is registered at the SEAP server (5) and maps to the specific hostname and port of the vttproxy A.

The same steps occur when client tier B makes a participation request (6). This leads to the generation of the combination *idB:svc1* (7). Furthermore,

this combination is send back to the client tier B (8). Afterwards, the COVS Grid Bean on client tier B creates a SSH tunnel (9) and registers the location of vttproxy B under the combination *idB:svc1* at the seap server (10). Note that all these pieces of information are transferred via secure Web service message exchanges or using the secure SSH tunnel.

While the end-users on client tier A and B wait for connection to the simulation, the parallel simulation job is submitted through the Grid middleware UNICORE for instance by an end-user at client tier C (not in the figure). This leads to a generation of the *sim:svc1* combination at the COVS Grid services (11). Hence, this combination identifies a particular simulation that provides the data for a specific interactive COVS session. This information is given as an input to the VISIT multiplexer (12).

Furthermore, this particular simulation is submitted to the underlying Grid resource by providing the *sim:svc1* combination for identification of the simulation (13). The multiplexer registers the combination *sim:svc1* at the SEAP server that maps on the location of the multiplexer (hostname and port) (14). This is necessary to provide the simulation with information where to send the data. But before any data is send, the combinations *idA:svc1* and *idB:svc1* are added to the multiplexer (15). Therefore the multiplexer is able to query for the exact location of vttproxy A and vttproxy B using these combinations (16) and to establish a connection to them.

In the meanwhile, the parallel simulation queries the SEAP server using its identification combination *sim:svc1* in order to know the exact location of the multiplexer (17). This information is then used to send the scientific data of the parallel simulation to the multiplexer (18). Next, the multiplexer sends this data via the connections to the vttproxies (19) that in turn forwards the data to the corresponding visualizations through the SSH tunnel. Of course, this established mechanism works also for the interactive steering control information that is using the same bi-directional channels and identification combinations.

Also, Figure 3 reveals the unique design of VISIT since the VISIT servers are integrated into the visualizations at the client-side, while the VISIT client is integrated into the parallel simulation. This makes the *parallel simulation act as a client* that initiates all operations like opening a connection, or sending the data to be visualized or receiving new interactive steering parameters. Finally, the exact hostname and port of the SEAP server itself are typically configures within an environment variable at the sites that run VISIT-enabled components. To sum up, by using the SEAP server within the COVS framework implementation all hostnames and ports of COVS session participants are dynamically configured at run-time which allows for maximum flexibility for ad-hoc participating end-users.

3.3 Enable Collaboration with Naming Service

Similar to the decoupled communication with the VISIT multiplexer and vttproxies as described in the previous section, the VISIT collaboration server uses also the SEAP naming service for decoupled communication. The collaboration server is used to exchange collaboration data between the multiple visualizations that participate in a COVS session. In particular, the CollaborationAdapter within the COVS Grid service starts the VISIT collaboration server that in turn register itself under a combination of *collaseapservice:collaseappassword* at the SEAP server. This means the combination maps to the correct hostname and port of the collaboration server on the target system tier.

In addition, the *collaseapservice:collaseappassword* combination is also exposed via WS-Resource properties of the COVS Grid service. Therefore, the COVS Grid Bean gets this information via WS message exchanges and transfers it to the scientific visualization via the COVS GridBean. After this transfer is done, the scientific visualization uses this information to query the SEAP server through the SSH tunnel in order to get the correct port and hostname of the collaboration server. Hence, the scientific visualization establishes a connection to the collaboration server through the SSH tunnel using the SEAP server. This allows for dynamic configurations of the VISIT collaboration server, for instance, it can be deployed on another host than the VISIT multiplexer (decrease load on host) and thus making it necessary to establish another SSH tunnel to this particular host.

To sum up, all collaboration data that is exchanged between the collaboration server and multiple visualizations is securely tunneled via SSH tunnels, including the SEAP information exchange.

4. INTERACTIVE COMPUTATIONAL USE CASES

In this section we review how the COVS framework implementation is used in interactive use cases with parallel HPC applications that solve n-body problems. N-body problems appear in many scientific domains such as astrophysics, plasma-physics, molecular dynamics, or fluid dynamics. N-body problems are concerned with determining the effect of forces between bodies (i.e. particles) [16]. In the domain of astrophysics several scientists use the COVS framework implementation with the Nbody6++ [14] code, which is a parallel variant of the Aarseth-type N-body code nbody6 suitable for N-body simulations on massively parallel resources.

The scientists use the framework with different subversions of the main code Nbody6++ for different areas of astrophysical research. These areas are dynamics of star clusters in galaxies and their centres, formation of planet

systems and dynamical evolution of planetesimal. In this context, the interactivity enabled via the framework allows for the exploration of parameter spaces through computational steering. In addition, the collaborative session support allows new forms of collaboration and experience exchange with geographically dispersed e-scientists within the AstroGrid-D, which is the astrophysical VO within the German National Grid D-Grid [17].

Another interactive use case in the area of e-science applications that represent N-body problems are several routines of the Pretty Efficient Parallel Coulomb solver (PEPC) [7]. Plasma-physics scientists use the COVS framework implementation with these massively parallel codes that use a hierarchical tree algorithm to perform potential and force summation of n charged particles in a time O(n log n). This allows for mesh-free particle simulation on length- and time-scales usually possible only with particle-in-cell or hydrodynamic techniques.

One particular use case of this code is for the simulation of a particle accelerator via laser pulses and the interactive approach of COVS realizes an interactive control of the laser target configuration. This in turn is very useful to verify start-up parameters such as the initial alignment of laser and target, or to perform quick test runs with a small set of particles as a prelude to full-blown production runs on large-scale systems.

In the context of large-scale systems (e.g. peta-scale systems) we foresee that any many-body code which is attempting to model macroscopic systems will benefit from higher particle numbers. This is because on the one hand it improves the overall statistics by better reproducing the mathematical set of equations used to describe the system. On the other hand the large scale systems permit simulation of larger, often more realistic systems (i.e. a galaxy in astrophysics).

Basically, we learn from both our COVS framework use cases that there is no upper limit on the number of simulation particles, but some codes are still in the process of getting scalable towards high amount of cores. This is particularly the case for codes that keep local copies of certain global data and are thus rather memory bound. But this could be overcome with some code restructuring which should permit scaling up to high amount of processors. In this context, also computational steering methods of the COVS framework have to be revised, especially in terms of the well-known master-slave pattern. This means the master collects the data from all processors and sends it to the visualizations. The current trend is that more than one masters are used to collect the data and to send the data out but this is still work in progress and has to be researched more deeply in terms of scalability towards peta-scale systems.

5. RELATED WORK

In the context of our work and the related work in the field interactivity mostly refers to the capability of a framework for computational steering. The UK RealityGrid project provides a steering library [1] that enable calls which can be embedded into its three components that are simulation, visualization, and a steering client. While the COVS framework is loosely coupled to the Grid middleware UNICORE, and could be in principle used with any other WS-RF compliant Grid middleware (e.g. Globus Toolkit [3]), the RealityGrid steering library is much more tighter integrated with the Imperial College e-Science Networked Infrastructure (ICENI) [6].

Related work in the field can be also found in the NAREGI project [18] offers a visualization system that consists of a visualization library and a Grid visualization API [4]. Even if this framework is also WS-RF compliant and thus similar to our work, the steering and interactive capabilities are rather limited (e.g. turn views on the simulation) that is not directly computational steering.

Finally, another well-known work with interactive behavior is the Grid Enabled Visualization Pipeline (GVID) [5], which offers interactive capabilities via Event-Encoders that run on thin clients and sends steering commands to the simulation. The major difference to our approach is that it is not seamlessly integrated as a higher-level service into a common Grid middleware.

6. CONCLUSIONS

This paper describes how the computational steering and online visualization with the COVS framework implementation can be used within todays Grid and e-science infrastructures. The fundamental benefit of our approach is that the COVS framework implementation can be used by any parallel application that is instrumented with the VISIT toolkit to gain interactive functionality in terms of computational steering.

According to our evaluations we proved that the SSH connections and the VISIT protocol that is used between all VISIT components provides reasonable low latency to interactively steer a HPC application conveniently. This also implies the real-time feedback per visualizations to observe changes in the application during run-time according to the steered parameters.

Also, we have shown how the SEAP server is used within our framework to solve the Grid transparency issues. By using the SEAP naming service within the COVS framework implementation all hostnames and ports of COVS session participants are dynamically configured at run-time which allows for maximum flexibility for ad-hoc participating end-users that have to technical knowledge about the distributed nature of the Grid.

To conclude, our interactive approach with a dedicated data and steering channel provides much better performance than using XML-based Web service message exchanges for large data transfer, while Web services are well suited for the management of the connections and the control of collaborative COVS sessions.

REFERENCES

[1] Cohen, J. et al., "RealityGrid: an integrated approach to middleware through ICENI", in *Philosophical Transactions of The Royal Society A*, 363, pages 1817–1827, 2005

[2] Eickermann, Th. et al, "Steering UNICORE Applications with VISIT", in *Philosophical Transactions of The Royal Society Journal*, London, 2005

[3] Foster, I., "Globus Toolkit version 4: Software for Service-Oriented Science", in *Proceedings of IFIP International Conference on Network and Parallel Computing*, LNCS 3779, pages 213–223, Springer-Verlag, 2005

[4] Kleijer, P. et al., "API for Grid Based Visualization Systems", in *GGF 12 Workshop on Grid Application Programming Interfaces*, 2004

[5] Köckerbauer, Th. et al.,"GVid - Video Coding and Encryption for Advanced Grid Visualization", in *Proceedings of the first Austrian Grid Symposium*, Linz, 2005

[6] Mayer, A. et al., "ICENI: an integrated Grid middleware to support e-Science", in *Component models and systems for Grid applications*, pages 109–124, Springer Verlag, 2005

[7] Pfalzner, S. et al., *Many-body Tree Methods in Physics*, Cambridge University Press, 1996, ISBN-10: 0521019168

[8] Ratering R. et al, "GridBeans: Supporting e-Science and Grid Applications", in *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, Amsterdam, 2006

[9] Riedel M. et al., "Computational Steering and Online Visualization of Scientific Applications on Large-Scale HPC Systems", in *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, Bangalore, India, 2007

[10] Riedel, M. et al., "Design and Evaluation of a Collaborative Online Visualization and Steering Framework Implementation for Computational Grids", in *Proceedings of the 8th IEEE/ACM Int. Conf. on Grid Computing*, Austin, USA, 2007

[11] Riedel, M. et al., "Extending the Collaborative Online Visualization and Steering Framework for Computational Grids with Attribute-based Authorization", in *Proceedings of the 9th IEEE/ACM Int. Conf. on Grid Computing*, Tsukuba, Japan, 2008, to be published

[12] Riedel M. et al., "Requirements and Design of a Collaborative Online Visualization and Steering Framework for Grid and e-Science Infrastructures", in *Online Proceedings of German e-Science Conference*, Baden-Baden, Online: http://edoc.mpg.de/display.epl?mode=doc &id=316630&col=100&grp=1414, 2007

[13] Riedel, M. et al., "VISIT/GS: Higher Level Grid Services for Scientific Collaborative Online Visualization and Steering in UNICORE Grids",in *Proceedings of 6th International Symposium on Parallel and Distributed Computing 2007 (ISPDC2007)*, Linz, Austria, ISBN 0-7695-2936-4, 2007

[14] Spurzem, R. et al., *Nbody6 Features of the Computer Code*, Online: ftp://ftp.ari.uni-heidelberg.de/pub/staff/spurzem/nb6mpi/nbdoc.tar.gz, 2003

[15] Streit A. et al., "UNICORE - From Project Results to Production Grids", in *Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing*, 14, pages 357–376, Elsevier, 2005

[16] Wilkinson, B. et al., *Parallel Programming*, Prentice Hall, 1999, ISBN 0-13-671710-1

[17] German National Grid Initiative D-Grid, Online: http://www.d-grid.de

[18] NAREGI Project, Online: http://www.naregi.org

[19] UNICORE Website, Online: http://www.unicore.eu

[20] Visualization Interface Toolkit (VISIT), Online: http://www.fz-juelich.de/zam/visit

[21] Web Services Resource Framework (WSRF) Technical Committee (OASIS), Online: http://www.oasis-open.org/committees/wsrf