

# RunBeast - Managing Remote Simulations

Alexander Siemers, Iakov Nakhimovski

August 30, 2001

## Abstract

In many application fields the simulation process is computation extensive and fast computers, e.g. parallel computers or workstation clusters, are needed to gain results in reasonable time. These high performance resources can only be accessed remotely via Intranet or Internet. RunBeast is a simple but effective client server application which simplifies running of remote simulations. It addresses all the major problems related to the data transfers over slow networks, unified access to different remote systems and administration across different organizational domains. The system is actively used at SKF in the context of the BEAST simulation toolbox.

## 1 The Simulation Framework

The RunBeast program described in this article is a part of the BEAST (BEARING Simulation Toolbox) package. The package is developed by SKF to perform dynamic simulations with special focus on detailed contact analysis which is required in simulation of bearings, and many other types of machine elements where contacts are important. BEAST enables studies of internal motions and forces in a bearing under given loading conditions. The model is fully three-dimensional, solving the general differential equations of motion for all components. Typical wall-clock simulation time ranges from several hours on a workstation for small jobs to a few days on a parallel computer for the large ones. More details about the capabilities of the package can be found in [3].

The typical working scenario for the end user follows. First the input data for the simulation has to be prepared using a program called Beauty. This program, which is also a part of the package, provides a GUI to setup various model parameters and boundary conditions. Then the input data can be submitted to the simulation code. The results of the simulation are saved in several output files and can be analyzed with two programs: Beauty and ViewBeast. The Beauty is designed specifically for 3D visualization and animation of simulation data. ViewBeast provides means for detailed analysis of 2D plots.

The RunBeast comes into play when the simulation has to be performed on a remote server in order to get results in reasonable time. RunBeast provides a way to send the input data to a remote server, start simulation, monitor its status and download the output files. In the following sections we will discuss our system design, problems related to the work on a remote server and provide details about our solutions for these problems.

## 2 Remote Simulation System Requirements

Our setup looks as follows. We have a number of engineers using our simulation software and a number of high performance computing sites where the simulations are performed. Most sites are geographically separated and the only possible connection is over the Internet or Intranet. The computing sites belong to different administration domains, run different operating systems, have different performance characteristics. Due to the nature of the Internet the quality of the connection between a user and a computation site depends on their physical location, overall network load, etc.

Such a configuration rises a number of problems. These problems are related to remote network access, usability of the software and some administrative issues. Some of the most crucial problems are discussed below.

- Remote access issues

Having many users potentially accessing the same site over the network results in the need to handle multiple simulation requests simultaneously. That is we need a queue system at every simulation server.

Typical size of a simulation output file is in the order of hundreds of megabytes and so transfer of large files must be guaranteed to work even over low quality network links. Transfer failures must be handled and automatic resume procedure supported. It must also be possible to start download before the simulation finishes.

Security in the system is guaranteed by the firewalls setup at all the sites. The file transfers are safe thanks to the proprietary compression formats, relatively small number of users and need for special tools to interpret the data. However, the remote simulation manager must enforce additional password protection of user data and prevent users from seeing each other work.

Since our users work in different countries it may happen that there is no a single server accessible to all the clients. That is the clients are forced to work on different servers due to the network limitations. This is why we do not want to have a centralized resource manager distributing jobs to all the clients. Any centralized location means one gateway that you have to access before starting your work and it may happen that this particular host is not reachable.

- Usability issues

Different simulation server sites often have different setups and require users to keep the specific of each site in mind. For instance, site policy often results in an automatically assigned login user name different for every site. A procedure for submitting a job can also vary, e.g. due to the differences in the terminal shells.

Remote terminal access over a bad network connection is often very unstable and can be additional discouraging factor stopping users from accessing remote servers and eventually using the simulation package at all.

Lack of a graphical user interface is also a problem, since most engineers are accustomed to work in a Windows environment.

- Administration issues

From the administration point of view we are facing problems due to the different policies and different system administrators working at different sites. First of all creation of a standard user account becomes a problem since every site has a special application procedure and it takes time to make all the applications. On the other hand some users need these accounts just for a short period of time to run a small project and the accounts can be safely removed after that. More convenient situation is to have a single service account setup at every site and let all the users to share this account.

Setup of all the necessary software should be possible with the basic privileges given to the service account user. This eliminates the need to contact administrators of every site just to make a software update.

### 3 Related Projects

The task addressed by the RunBeast is very similar to the problems solved by the batch queue systems and more recently by the Grid technologies. Before going into details of our system we would like to mention these existing technologies and explain why they are not suitable for us.

#### 3.1 Queue Systems

The problem of multiple users access to the same powerful computer is not new. Many high performance computing sites have been using batch queue systems for many years. A comparison of different systems can be found in [5].

Most of the systems are designed for Unix-like operating systems. Although the systems are portable not all the features are available on different platforms. For example forced checkpoint restart requires special support from the operating system and is often not supported. Some of the well known queue systems are LSF (Load Sharing Facility) often used on SGI Origin systems, NQE (Network Queuing Environment) used on Cray T3E and PBS (Portable Batch System) ported to many platforms. These systems are typically oriented toward work on a single site with all computing nodes (CPUs or cluster machines) connected to a designated front-end machine.

The tight coupling between the queue system and the operating system becomes a drawback for the application services such as ours.

First, we normally have access to only one Unix account at every site. All the users of our service are using a single Unix login name but different higher level user or project names. These higher level names are known to the service but the operating system has no notion of them. It means that the queue system relying on the Unix user names cannot distinguish between two different users of the service. Even if there is an accounting system for different project names the selection of a project ID by the user is not protected.

Second problem concerns with the use of different queue systems at different sites. Each system has its own script notation and the user would need to provide the same information in a different style every time he or she decides to use a different site.

Next problem is the Unix environment itself. Many industrial users are accustomed to one of the Microsoft Windows operating systems. The necessity to write a script and

use command line to perform a simulation creates an extra barrier for the use of the simulations.

The last problem that we would like to outline is the output files copying process. As we have mentioned earlier the amount of output data forces us to support download during the simulation and resumed download. The standard tools lack the support for these tasks.

### 3.2 Grid Technologies

The scope of the Grid project and related research problems is described in several articles, books and on many web sites. See for example [2, 4]. The Computing Portals group [1], which is a member of the Global Grid Forum (Global GF), is aimed at contributing to the coherence and interoperability of frameworks, portals, Problem Solving Environments, and other Grid-based computing environments and Grid services. Some participants of the group are working on the problems that are similar to our.

There are however some conceptual differences between the approach taken in the Grid community and our intentions. One important distinction is in the representation of the resources to the users. The Grid technology suggests that the end user just connects to the Grid without knowing the properties of different sites. We instead just want to provide unified access to all the sites but keep the information about the computational power of every separate site open. So, in our system the user can choose his preferred site and the system does not try to alter this decision.

Grid technologies assume collaboration between sites that was not available for our project for the administration reasons mention earlier. Many of the Grid projects are still under development and as such can be problematic to use in an industrial system.

## 4 RunBeast System Architecture

RunBeast implements a client server architecture for the remote control over a simulation server. It is used to start and monitor BEAST simulations on remote machines. One or many clients can connect to one or many simulation servers, one at a time, and submit simulations into the simulation queue, see Figure 1.

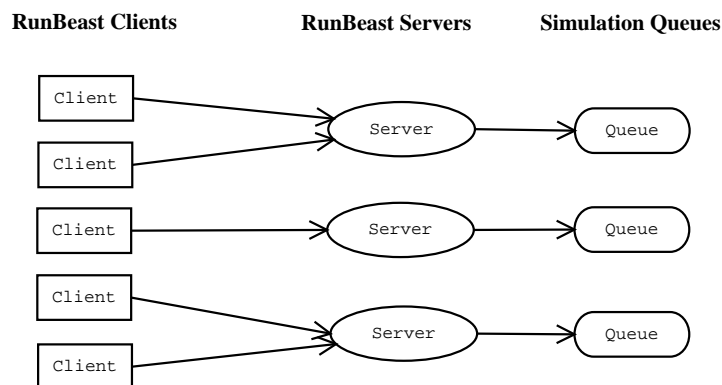


Figure 1: RunBeast clients connect to RunBeast servers to submit simulation jobs to the simulation queue.

It addresses all the multi-client multi-server problems mentioned in section 2.

- Reliable remote access
- Usability
- Simple administration

## **4.1 Remote Access**

### **4.1.1 Multiple Simulation Requests**

Several users can connect to a server at the same time. To handle multiple user access a simple queuing system has been implemented on the server side. Each server has exactly one queue. Different queues (and servers) for different priorities can be setup for a single cluster. Higher priority queues are emptied first then lower priority queues start their jobs.

Jobs can be removed from a queue at all times. Currently running jobs will be killed first then removed from the queue. Only the authorized users are allowed to remove jobs from the queue. Authorized users are server administrator and the user who queued the job.

A client can connect to one server at a time but can switch to another server at run-time. Thus several simulations can be submitted to different simulation queues but only one simulation can be monitored at a time. Each server is configured for a certain cluster of machines or multiprocessor machine. Servers might be installed on the master node of a cluster or any other machine with connection to the master node. Communications are always initiated by the client with a request to the server.

### **4.1.2 File Transfer**

Several files are transferred between client and server. When a new job is submitted into the servers queue the preprocessed input file is sent from the client to the server. To analyze simulation data simulation files are fetched from the server to the client machine.

File transfer is implemented via sockets. The data is splitted into equal sized blocks. To assure correctness of the data a checksum is added to each block by the server before it is sent to the client machine. The other party is accumulating these blocks and restores the data.

Two techniques are used to handle broken connections. Simulation input files are relatively small and are transferred once again. Dynamic output data files are the big ones, in the range of hundreds of megabyte up to two gigabyte. When a dynamic output file transfer has been interrupted it can simply be resumed when the connection is stable again. Client and server compare file sizes and resume transfer after the last transfered block.

The file resuming technique is used to implement another transfer method, which we call “fetch-during”. Here the user can choose to fetch simulation files in blocks while the simulation is still running. This is done to avoid large file transfers after the simulation is finished. A user adjustable timer is used on the client side to trigger fetch-during’s. Each time a fetch-during is triggered the client and server negotiate local and remote file sizes and transfer the data difference.

Simulation output files are written in blocks by the simulation program. If a block write is completed the block is marked as valid. RunBeast ensures that only valid blocks are transferred to the client. Even the fetch-during method transfers valid blocks only. This way the user can start analyzing data before the simulation is actually finished and kill it if the results are not satisfying.

### **4.1.3 User Access**

Each user has its own simulation directory on the server and can see and work with this directory only. This prevents users from seeing each others work. Subdirectories might be created by the user to keep simulations ordered. If simulation files are downloaded a copy will be kept on the server. Simulation files stay on the server until the user decides to delete them.

Running jobs can be monitored by users. The owner of the current simulation will get detailed job information. This includes progress of the simulation, solver information, and some statistics about parallelization. Other users only get information about job owner, and simulation progress.

### **4.1.4 Resource Management**

RunBeast uses a decentralized resource management system to assure server availability. Centralized management systems often use a single gateway for several servers, thus it is often transparent to the user which resource is actually used. If this gateway is down no resources are available any longer. To avoid this RunBeast implements one gateway for each server. The user decides which server to use for the simulation. If a server goes down only a single gateway is unavailable and all others are still open.

## **4.2 Usability**

Usability is granted in two ways. First, user accounting and simulation startup is handled by the RunBeast server. Thus the user does not care on which machine the server is running. It might be a Linux cluster or a multiprocessor Alpha machine, this is transparent to the user. Second, users are provided with a Java based user interface which looks the same on all machines. Since standard Java is used for the user interface no platform specific development is necessary.

No matter if the user is sitting on a Microsoft Windows driven machine or a SUN workstation submitting jobs to a simulation server is always the same. Thus users need to be trained only once and will be able to run the application on different platforms and start simulations on different servers.

## **4.3 Administration**

A single service account system is used by RunBeast. One RunBeast server account is needed per simulation server machine. The servers implement a higher level accounting system for RunBeast users. Thus users do not need a special account on each simulation server machine but use the same username and password for all RunBeast servers. New

users can be added to all RunBeast servers by the RunBeast administrator without the need to notify the system administrators of the simulation machines.

Each server keeps an user account file which contains username, e-mail, and password. Each client request to the server, e.g. getting status information or submitting a job, is initiated with a login procedure. This is transparent to the user since the client stores username and password. The user is asked for the username and password only once, when connecting to a new server.

## 5 Conclusion and Future Work

The RunBeast system has been in use for three years now and has proved to be efficient and useful. From the user point of view it has allowed the access to a number of remote servers via the same intuitive GUI. From the developer point it has simplified the introduction of new simulation servers to the users since no extra tutoring is required thanks to the single client interface.

The future development of the RunBeast system will be directed toward integration with the standard tools described in section 3. We are exploring the opportunities to build the specialized simulation server queue on top of the general queue systems of the different sites. Rapid development of the Grid-related projects can make an implementation based on these technologies feasible in the future.

We believe that the setup described above is quite general and that it will be used more and more in the industry as the situation when many servers are providing the same service over the Internet to many users become more popular.

## References

- [1] Grid Computing Environments working group.  
On-line: <http://www.computingportals.org>.
- [2] I. Foster, C. Kesselman (Eds), and Morgan Kaufmann. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.
- [3] D. Fritzson, L-E. Stacke, and P. Nordling. BEAST - a rolling bearing simulation tool. *Proc. Instn Mech Engrs, Part K*, 213, 1999.
- [4] The Globus Project. On-line: <http://www.globus.org>.
- [5] Michael L. Nelson Joseph A. Kaplan. A comparison of queueing, cluster and distributed computing systems. Technical Report NASA TM 109025, NASA Langley Research Center, 1994.