IGGI, A COMPUTING FRAMEWORK FOR LARGE SCALE PARAMETRIC SIMULATIONS: APPLICATION TO UNCERTAINTY ANALYSIS WITH TOUGHREACT

F. Dupros[a], F. Boulahya[a], J. Vairon[a], P. Lombard[c], N. Capit[b], J-F. Méhaut[b]

[a]BRGM, , BP 6009, 45060 Orléans Cedex 2, France(f.dupros@brgm.fr / +33 2 38 64 46 76)
[b]Laboratoire ID-IMAG, INRIA Rhône-Alpes, 38330 Montbonnot Saint Martin, France
[c]ICATIS, 37 avenue du Granier, 38240 Meylan, France

ABSTRACT

IGGI stands for infrastructure for grids, cluster and intranet. This research project partially funded by the French government is aiming at developing technologies allowing the access and the gathering of the whole computing resources spread over the intranet of a company. This could include dedicated computing power or personal computers. The project is collaboration between BRGM, INRIA and Mandriva.
Scientific computing has evolved for last few years and commodity-based components can run a wide range of applications. In this study, our focus will be on the software environment required to carry out large scale parametric simulations, but we will also report on experience of the deployment of such an infrastructure in the day to day life of an intranet.
The IGGI software suite is mainly based on two components: ComputeModeTM which smoothly aggregates idle user machine to a virtual computing cluster. This is done through a transparent switch of the PC to a secondary, protected mode from which it boots from the ComputeMode server taking advantage of the PXE protocol. The second IGGI component is CIGRI which scheduled and executed computing tasks on the idle nodes of clusters.
Special attention has been paid on the end-users ability to perform easily parametric simulations. Transparent access to the batch-scheduler, checkpoint and migration of the application will be exposed for the test-case of the analysis of uncertainty with TOUGHREACT : "Uncertainty in predictions of transfer model response to a thermal and alkaline perturbation in clay" .
The calculations reported were carried out using idle computing capacity on personal computers inside the BRGM. This new architecture is particularly well suited to explore the wide range of perturbations in highly coupled problems.

INTRODUCTION

Grid computing technology enables the collaborative usage of computational resources across different organization. An important part of grid research effort has been focused on this part of grid application. Metacomputing as defined by Larry Smarr (1992), uses a few stable high performance computers with a secured environment through dedicated or non dedicated networks. Experiments with large configurations and real applications have shown impressive results (N.Barberou et al. 2003) but require the design of special family of methodology or algorithm (M.Garbey et al 2000) to overcome the prohibitive latency cost and low bandwidth and latency cost . From the engineering point of view , coupling high performance systems is not feasible in a day to day practice. Netherverless most industrial companies or research institution owned medium scale computing architecture.
On the other hand, personal computer power has rapidly increased these last few years, new architectures have emerged for low cost desktop machines (multicore , hyper threading) . The intranet network of a company often provides more computing power with the CPUs of his engineers than with its specific servers.
A logical way to response to the peak of CPU needs or to the large scale simulation required is to aggregate dedicated resources like supercomputers, cluster of PC and the computing power spread over the intranet. The middleware required to target the resources need to address several issues like heterogeneity, network performance , security, resource volatility etc..
Nowadays the Globus Toolkit represents an industrial standard and his involved in a large number of research projects (http://www.globus.org/.). Even though Globus offer tools to perform grid-based execution, this solution does not offer the tools provided in the framework of IGGI, especially for parametric simulations.

The national French project IGGI is aiming at developing robust technologies to access and gather the computing resources over the intranet. The IGGI components smoothly aggregate resources and provide at the user level and easy-to-use approach to generate and follow large scale parametric simulation. This architecture could also be used to run parallel and communicating applications using for example the message passing interface (MPI). In this paper we will report on experience with the reactive and sequential version of TOUGH2 : ToughReact. In the field of highly coupled problem this is particularly useful to carry out perturbations analysis and understand the global behavior of the phenomena.

An other important point in grid context is the adaptation of the application. A grid infrastructure is highly volatile. In our infrastructure the desktop computers are used during idle period (night, week-end, and vacations) so we need to use fault-tolerance mechanism to guarantee the complete running of the application despite failure or unpredictive come back of PC owners.
Existing checkpoint facilities inside the Toughreact software, these possibilities have been extended and compared to other approaches based on checkpoint library (D.Thain et al. 2003).

The outline of this paper is as follows.
Section2 describes the components of the computing framework IGGI. Section 3 exposes the deployment of this solution inside a firm or research institute. The test and adaptation performed with Toughreact are detailed in section5.
We conclude on the first results in section 5.

IGGI COMPONENTS

The IGGI middleware is composed of two main components described in this section. The first one is ComputeMode which manages the hardware infrastructure. The second IGGI component is OAR/CIGRI which schedules computing applications on the computing nodes.

ComputeMode

General design

ComputeMode (B. Richard and P. Augerat 2004). is a software suite aiming at providing a seamless Linux cluster infrastructure within an Intranet. To achieve these goals, it uses several system and network technologies such as :

- PXE boot with DHCP and Wake-on-LAN: the configuration is done so that it is transparent with a firm's DHCP server
- NFS accesses: currently provided by the ComputeMode server - an industry NAS may be used too
- PostgreSQL database linked to an Apache server to offer an easy to use web interface

ComputeMode is available free of charge under an Open Source license through http://www.computemode.org/.

Adding a node into ComputeMode

Prior to being a cluster node, a node has to get registered, that is entered into the system. An automatic registration may be carried through if the node happens to boot in PXE. Registration mainly consists in binding a hostname to a MAC address and a schedule.

Schedules, in ComputeMode, are indeed booting schedules, that is some sort of time table which tells when a machine has to go 'Local' (for instance during the day) and when it has to go 'Cluster' (that is at night or on weekends for example).
Now, whenever this machine boots and the time tables tells it to go 'Cluster', then a network Operating System is started and the machine gets registered into a standard job manager, OAR (see next part).

Nodes and labels

If a user arrives when a machine is in Cluster mode, it may choose to halt it by hitting 'Alt-Ctl-Del'. Upon hitting these keys, the server will be informed that the node should be quarantined with regards to ComputeMode. The administrator will have to remove the quarantine by removing the 'Quarantine' flag at a later time.
Administrators also have the possibility to add flags to “glue” properties to nodes. For instance a node may have, at the same time, a model_xyz and room_abc properties. Such properties are exported to the job manager.

Checkpointing and interruption

If a job is currently being run on a node which has to reboot (because of an owner interruption or end of computation time slot), then an asynchronous message is sent to the OAR job manager. Upon reception, OAR then tries to get the application checkpointed by sending it a Unix signal.
After a tunable (currently ~30s) delay, the node is halted or rebooted.

Getting more of users' vacations

Current schedules are based on a standard week's work that is most machines are thought to be available at night and during the week end.
To get some more availability, users may inform the system thanks to a simplified web page that they will be out of office for the next few days. This web page will also provide them with the information on what has currently been done on their machines when they were away.

Computing in an almost homogeneous environment

As every node boots a same diskless Linux flavor, the computations take places in an almost homogeneous environment without library or kernel version hassles.
The only sources of heterogeneity are the difference in CPU powers and the RAM available, which has been shown to be a good trade-off.

Computing safely in security

The diskless OS deployed by ComputeMode offers a safe environment for the host machine as no access whatsoever is ever done to the local disks. As no disk driver is loaded by the diskless Linux kernel, it makes it very difficult for a malicious job to succeed in alter the owner's data.

Besides, ComputeMode uses a network addressing scheme (private B class 176.28.0.0/16) different from the one used by the BRGM (private IANA A class 10.0.0.0/24). Such configuration helps ensuring a complete separation of the standard network and the computation network.

Last but not least, logging in on a node is only allowed through an ssh coming from the ComputeMode server or through an interactive job reservation (with OAR).

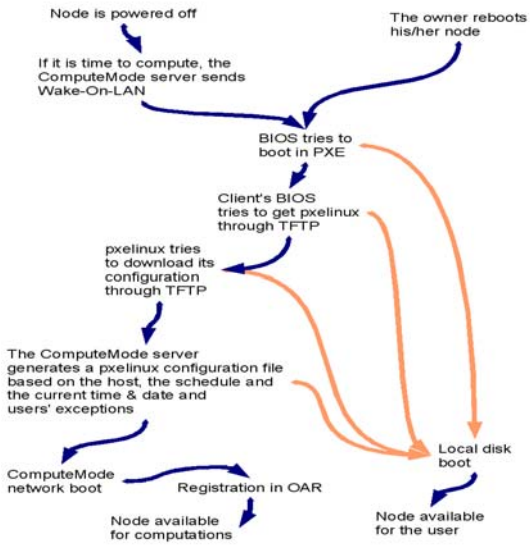

Figure 1. Workflow in ComputeMode

Scheduling : OAR - CIGRI

OAR

A batch scheduler is a system that manages resources, which are the nodes of the cluster. It is in charge of accepting jobs submissions (which are sequential or parallel computation required by the users of the cluster) and schedule the execution of these jobs on resources it manages.
Thus, the objective of a batch scheduler is to allow users to use resources easily : users should not have to worry neither about the availability of the nodes nor about the interference of their job with some other job.
There are plenty of available batch schedulers for clusters and parallel machines. Among the most known we can quote PBS/OpenPBS (http ://www.openpbs.org/overview.html.), LSF, Condor (D.Thain et al. 2003).
The two studies (M. Baker et al 1995) and (http://www.crpc.rice.edu/NHSEreview, 1996) list these major systems and their features.
These systems are generally developed using the C language and implement by themselves the storage, management and logging of job submissions (because of performance issues).
Although they all provide an interface to use and extend them, none of them fullfil all of our needs.
Some issues in most of these systems are the lack of support for job that might be automatically cancelled when its resources are needed (which is required to implement efficiently a support for large multi-parametric applications) and the lack of convenient exploitation of logging information.

OAR (N.Capit et al 2005) is a job manager developed by the ID-IMAG laboratory. This batch scheduler is based upon an original design that emphasizes on low software complexity by using high level tools. The global architecture is built upon the scripting language Perl and the relational database engine MySQL.

The goal of the project OAR is to prove that it is possible today to build a complex system for resource management using such tools without sacrificing efficiency and scalability. Currently, our system offers most of the important features implemented by other batch schedulers such as priority scheduling (by queues), reservations, backfilling and some global computing support.

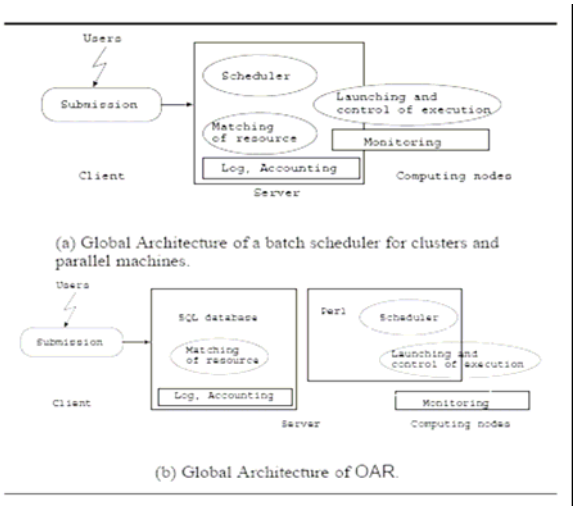

Figure 2. Architecture of a batch scheduler

Despite the use of high level tools, our experiments show that our system has performances close to other systems. Furthermore, OAR is currently exploited for

the management of 700 nodes (a metropolitan GRID) and has shown good efficiency and robustness.

The IGGI project uses OAR specially the node state switch feature and properties. These one describe each computers with values like CPU speed, memory capacity, etc.

The command line utilities provide the user with basic (and more evolved) node reservations in an interactive or batch mode and functions to get statistics, or the list of current jobs or current nodes.

OAR is available free of charge under an Open Source license http://oar.imag.fr/

CIGRI

CIGRI is a campaign manager for parametric job which was written first to gather and ease the use of several clusters of the CIMENT Grid project http://ciment.ujf-grenoble.fr.

Similarly to the other key components, it is based on a MySQL database and a web interface to provide a front-end to the users as well as graphic charts and feedback. Back-end and command line tools are written in Perl and may be used to launch and cancel campaigns.

CIGRI acts as a meta-scheduler and uses OAR and ssh to successfully handle the campaigns (task and job scheduling, multi-cluster support).
CIGRI enables to manipulate huge parametric campaigns (several thousand parameters) and indicates the user where jobs are executed at what time.
Moreover this middleware uses a specific job type which is implemented in the batch scheduler named “besteffort”. Indeed this feature permits to execute a task and when an other user wants to use the cluster it is killed to let him work. So CIGRI jobs don't disturb “normal” user of the cluster, it just optimize the computing time.
Users can see what happens on the grid platform and what are errors related to their campaigns. Thus, for example, they can decide to ignore or repair a job in error.

In the BRGM context, CIGRI can launch tasks on both the dynamic ComputeMode cluster and the dedicated computing cluster.

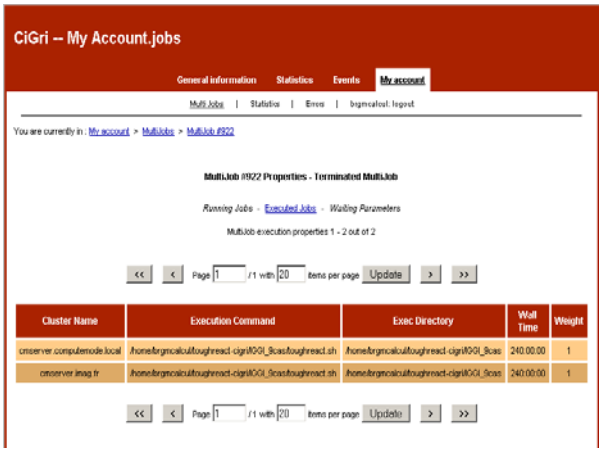

Figure 3. Cigri interface

CIGRI is available free of charge under an Open Source license http://cigri.imag.fr/

DEPLOYMENT

Computational needs in geosciences

The computing power of the French geological survey is essentially based on a 12 nodes bi-xeon cluster interconnected by a gigabit network. This architecture is rather classical and can handle day to day simulations, especially for seismic wave propagation, geochemistry, soil mechanics etc..
Like dedicated computing center the computing load is quite irregular on this kind of machine.
Large scale FEM simulations could require the reservation of the entire cluster to understand more precisely a phenomena. On the other hand long term predictive geochemical simulations could be running during a couple of days and not optimize the high performance components of the computing system (bandwidth – RAM memory etc...)
An other important point is the growing need for parametric simulations, especially in environmental modelisation. A good illustration could be the ongoing European project footprint.
FOOTPRINT is a research project of the 6th European Framework Program which aims at developing a suite of three pesticide risk prediction and management tools, for use by three different end-user communities: farmers and extension advisors at the farm scale, water managers at the catchment scale and policy makers/registration authorities at the national/EU scale http://www.eu-footprint.org/

This research project is based on the evaluation of different scenarios (climate, soil types, crops etc...) and the first estimation of the CPU power required leads to several hundreds of PC running pesticides models during few months. National scale computing facilities could not be totally devoted for such kind of parametric simulations and grid infrastructure is the only solution, especially for institution like BRGM.

Resources available

The French geological survey is composed of roughly 600 researchers and 250 employees working in the administrative staff. So the immerged computing power spread over the intranet is very important.
Table 1 gives some fingers on the numbers of machines available, the first approach of the IGGI infrastructure is focused on desktop computers.
Assuming an average of 2Gflops per machine, a simple aggregation gives a computing grid of more than 1 Tflops !

Table 1. Computers repartition in Orleans.

Desktop computer	350
Engineers computer	300
Laptop	250

Those computers are spread over 27 hectares, and 25 buildings (figure4)

Figure 4. BRGM Campus.

The first challenge of IGGI is to gather different resources and make them easily available for the users. The components listed in the previous part are used to schedule in a transparent way the different kind of jobs. Parallel jobs could have a high priority on the dedicated computing cluster to take full benefit of the quality of the network and the CPU. Parametric campaign only uses the computing resources when their availability is reported by the scheduler OAR.

Figure 5 gives an overview of the global architecture with the aggregation of the intranet resources and of a computing cluster.

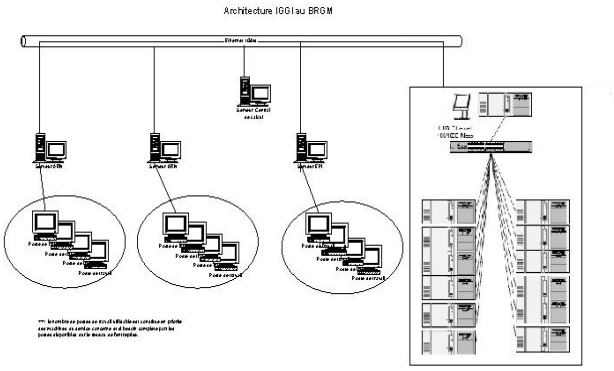

Figure 5. IGGI Global architecture.

Integration

Contrary to other grid project, a reboot the PC is performed for the IGGI infrastructure. This choice guarantees optimal usage of the hardware resources, security through private class of address, modularity in the choice of the flavor of Linux for the computation. In the context of a research institution we have choosen a progressive approach for the deployment. The first target was the IT department representing approximately 50 PC.

The strategy is based on voluntaries: employees who want to contribute switch off there computer when leaving. This solution avoids the reboot of PC required for specific tasks during the night (long compilation, special benchmark …). When coming back in the morning, the computer is available in Windows mode. Including vacations, nights and week-end a single PC could be used 4.5 days in a week for grid computations, IGGI solution enhances the real exploitation of IT resources.

The default period of computing is from 6pm until 8pm during business days and the whole Saturday and Sunday. This is defined through a user-friendly interface **(**figure 6), one can easily define special schedule period for certain class of PC (training classroom depending of availability for example).

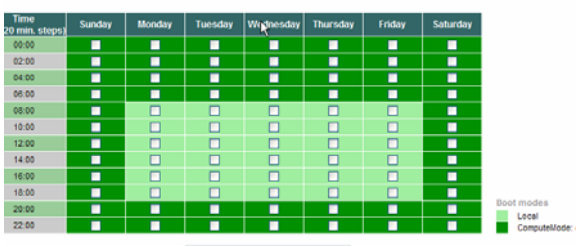

Figure 6. Schedule

The schedule interface is only available for the administrators of grid. An over interface is designed to allow any employee to define is vacations period (consider like an exceptions in the ComputeMode system).
Each year a third of the computers is renewed, during this phase; all new BRGM computers will integrate the virtual computing cluster. 3 years plus 2 categories (desktop/engineers) imply 6 configurations of computers: this demonstrate the heterogeneity of resources and the ability of IGGI to integrate them smoothly.

ADAPTATION

Checkpointing

Checkpointing is taking a snapshot of the current state of a program in such a way that the program can be restarted from that state at a later time
Additionally, periodic checkpointing provides fault tolerance. Snapshots are taken periodically, and after an interruption in service the program can continue from the most recent snapshot.
In the framework of IGGI architecture, the computing period on a single machine could not be contiguous and a simulation could be checkpoint on a first machine and restart on an other. For example the average elapse time for one simulation in *"Uncertainty in predictions of transfer model response to a thermal and alkaline perturbation in clay"* is approximately 80 hours. We have to checkpoint the application at least two times considering a complete week-end (58 Hours) and 3 nights (30 Hours) of computing availability on a desktop machine.

Application Level

The Toughreact software provides facilities to checkpoint and restart simulations. One can use the SAVE and savechem files to stop an ongoing simulation and resume it later using the latest computed state. To integrate this feature in the complete process of IGGI capaign we need the ability to decide when writing with this two checkpoint files and stop the application. Two strategies have been studied due to the small size of these files:

– 23 Kb for SAVE
– 423 Kb for savechem

First of all, we have included in the Toughreact source code an extension that allow the application to catch specific Unix Signal. By sending SIGUSR1 signal, the application stops and generates the relevant files for future restart. This signal could be sent by the user during the running for debugging purpose but the main purpose is to make OAR use this commodity when the computing period for the desktop computers is over.
We have also used the NWTI control variables of Toughreact to force the checkpoint at fixed period. This mode could be seen as fault tolerance execution and is transparent in terms of global CPU performance in comparison with classical simulations.
A light script included in the Cigri campaign management perform the required files manipulation for the restart (rename – modification of the flow.inp etc...), this script also glue the different results file generated during several runs of the same test case.

Condor

An other possibility to checkpoint application, especially sequential application is to use condor library.
Condor provides checkpointing services to single process jobs on a number of UNIX platforms. To enable checkpointing, the user must link the program with the Condor system call library (libcondorsyscall.a), using the condor_ compile command.

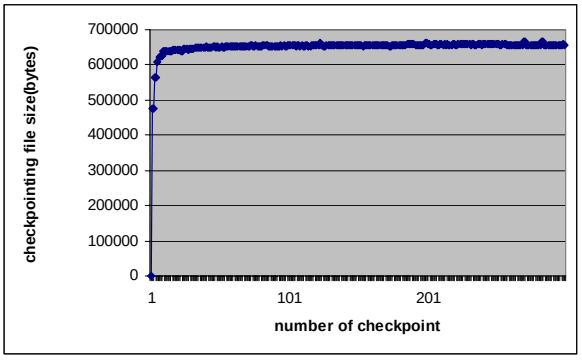

Figure 6. Checkpointing file size.

The bench has been performed on the reference test case and for a quarter of the 2000-years of simulation. The checkpointing time is negligible and the size of the condor restart file is quasi equivalent during the simulation (approximately 630 Kb).

Restart

The previous methodology to checkpoint Toughreact application could appear equivalent, one need also to consider the restart part of the mechanism.
Condor uses -_condor_restart command to perform execution of the code from the checkpoint file. The simple use of pmap of ps facility to trace the the memory consumption shows that a quite simple execution of Toughreact could become very costly with condor (table 2)

Table 2. Memory distribution.

	Resident memory	Virtual memory
Before checkpoint	4 816 Kb	152 909 Kb
After checkpoint application level	4 819 Kb	152 900 Kb
After checkpoint condor	153 025 Kb	153 752 Kb

First explanations of this behavior could be the massive use of common fortran feature. This instructions induced a special management of the memory and condor probably restart the application considering all the memory included in the common is really used. These firsts results need some more investigations. A workaround is probably to pay special attention to the default configuration of the common size in Toughreact source code.

CONCLUSION

A grid computing architecture is presented, the different components of the IGGI project allow the aggreagtion of heterogeneous computing resources spread over the intranet : desktop PC, cluster.
Thanks to the special design of the middleware, especially Cigri and OAR, this infrastructure is particularly well adapted to multi-parametric simulations and experiences have been carried out with the software Toughreact.
An important point in our infrastructure is the ability to checkpoint application, Berkeley Lab Checkpoint Restart (Duell et al.) will be evaluate to complete our current implementation.
During the deployment phase, one of the botlleneck could be the manipulations required on the PC to allow network boot.
The evolution planned for our infrastructure could be the use of virtualization technologies (Qemu, Xen or WMware) instead of performing a complete reboot of the machines.
This would allow to collect small period of inactivities. Some preliminary results with the Toughreact software show a degradation of less than 15 % of the performance between a reboot of the PC and the deployment of a VMware client running Toughreact. Due to the low memory consumption of Toughreact, virtualisation also allows the PC owner to handle his day to day activities during running of the code without any interactions.

REFERENCES

L. Smarr and C. E. Catlett, Metacomputing, Communications of the ACM 35-6 (1992) 45-52.

N. Barberou, M. Garbey, M. Hess, M. Resch, T. Rossi, J. Toivanen and D. Tromeur-Dervout, E±cient meta-computing of elliptic linear and non-linear problems, *J. of Parallel and Distributing Computing*, 63 (2003) 564-577.

M. Garbey and D. Tromeur Dervout, A Parallel Adaptive Coupling Al-gorithm for Systems of Differential Equations, *J. Comput. Phys*., 161(2) (2000) 401-427.

D. Thain et M. Livny. Condor and the grid. In Fran Berman, Anthony J.G. Hey, et Geoffrey Fox, éditeurs, Grid Computing : Making The Global Infrastructure a Reality. John Wiley, 2003.

B. Richard and P. Augerat , Effective Aggregation of Idle Computing Resources for Cluster Computing, , *Journal of European Research Consortium for Informatics and Mathematics (ERCIM*), special issue on Grids, next generation, October 2004

M. Baker, G. Fox, and H. Yau. Cluster computing review.Technical report, Northeast Parallel Architectures Center,Syracuse University, 1995.

N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, O. Richard, A batch scheduler with high level components, Conference on Cluster Computing and Grids (CCGrid'2005), Cardiff, UK, May 2005

Duell, J., Hargrove, P., and Roman., E. The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart. Berkeley Lab Technical Report (publication LBNL-54941)