Massive Cycle Harvesting for Computational Chemistry

Brian Vinter, Rasmus Andersen, Jonas Bardino and Henrik H Karlsen, University of Copenhagen, Copenhagen, Denmark

Martin Rehr, University of Southern Denmark, Odense Denmark

Cycle harvesting, or Screen Saver Science, has become very popular with some fields of science because the computational power that may be harnessed through desktop scavenging is enormous. Unfortunately it is non-trivial to write applications that can run in these environments since a number of rules must be obliged in order to ensure the safety of the computer that performs the calculations. Porting large code-bases to the existing screen-saver-science systems is usually not feasible, and maintaining the code in two versions, a standard version and one for screen-savers, is highly undesirable. In this text we present work on porting a standard applications from computational chemistry, NAMD[1], to a generic sandboxed screensaver. To further improve things the sandbox computing model of our Minimum intrusion Grid, MiG[2], actually provides real scheduling so that a single job is never left waiting a long time for CPU cycles once it has been started. To show that the same model may be used for internal Grids the setup have been repeated using a commercial Grid system, OfficeGRID[3], where a number of Linux virtual machines have been hosted on a network of Windows PCs.

Portability and Security

A classic problem with Grid computing is the need to have a software base on each resource that contributes to a calculation. In the case of computational chemistry, it is hard to imagine thousands of PC owners installing and managing NAMD distributions. In addition the same PC owners would be putting a lot of trust in the software they download since it could, whether intentional or not, raise any number of security concerns. Thus what is needed is a so-called sandbox in which the application is isolated from the host machine and vice-versa. Such a sandbox usually takes the shape of a virtual machine, which is exactly the approach that we have taken in this work. Using such a virtual machine we have managed to eliminate the need for installing any chemistry software on the host computer, only the Screen Saver is needed.

Scheduling

Existing Screen-Saver-Science systems all target problems that has many, usually millions, of independent tasks that often run for tens or hundreds of hours. Once a task has been assigned to a computer it will be processed while the computer is in screen-saver-mode. Processing is suspended if the screensaver is suspended and respectively resumed again along with the screensaver. An artefact of this model is that one never knows when the result of a given task is ready, and it is very hard to determine if the task has been lost or if it is simply only allowed to proceed very slowly.

For applications such as computational chemistry this model is very poorly suited. The number of tasks is usually in the tens or hundreds and it is often the case that analyses of the results can only start once the result of every task is in. Thus we cannot use the luxury of having millions of tasks to execute or even the ability to use the results before all the tasks have completed.

To address this problem we have adopted two models; expected runtime and low priority processing. The MiG system provides the means to guess the length of time a screensaver will be active once it activates, these guesses are very good, and only if the guess is too optimistic, will it be necessary to kill the execution and restart the task on another machine. More details on this model may be found in [4].

The alternative approach, low priority processing, works under the assumption that there will at any time be an abundance of processing power available on a modern PC, even when it is being used for interactive applications. Thus by specifying the process to be low priority we should be able to have a computational task running in the background without annoying the user, since only spare CPU cycles will be given to the low priority

task. Unfortunately this has previously been an unsuccessful approach for generic applications because while CPU is only given to the process when there are cycles to spare, memory is allocated with the same priority as any other process. This means that a computational task could allocate enough memory to inhibit efficient usage of the PC for the interactive user. However, using the virtual machine, we are able to limit the amount of memory that is visible to the virtual machine and thus we can set a bound on how much memory a computational task is allowed to allocate and we can thus protect the user against interference from the background job.

An experiment

To verify our model we have run a number of NAMD jobs on a LAN of Windows machines. Since the authors are all computer scientists we know very little of computational chemistry and we have thus opted to take an example NAMD job for our experiments (par_all27_prot_lipid). As nobody would want to run the exact same job a number of times we have taken the assumption that we cannot expect the execution time for individual tasks to be the same. Rather than attempting to change the chemical properties of the simulation to obtain this end we have chosen to vary the 'minimization' option in NAMD, which effectively makes the simulation run for different periods.

The experiment is run on a LAN with 8 identical Windows computers. Each computer is the host of a virtual Linux machine that runs the Grid client code and the NAMD simulation itself, one of the machines also runs the master Grid software, this is run in native Windows mode.

Since we have 8 machines we have chosen to execute 25 jobs, this means that there is 3 jobs per machine while one last machine needs to execute one additional job before the experiment is completed. The minimization number and the runtime of the 25 jobs can be seen in figure 1.

When submitted to the Grid, the 25 jobs were completed in 21 minutes and 5 seconds, with an accumulated runtime of 2 hours, 29 minutes and 23 seconds. Repeating the experiment on a single machine resulted in a total runtime that was 4 minutes and 1 second lower, translating to an overhead of 2.76% or 10 seconds per job for being run in the background and through the Grid system. Such overhead is hardly noticeable and the model must be deemed successful.

Conclusions

Unused PC's represent a huge computational resource for science, including computational chemistry. However, the complexity of installing, managing and ensuring security of a large computational chemistry package is prohibiting widespread acceptance. Using secure sandbox technology the Minimum intrusion Grid has successfully eliminated all of these limitations. Performance findings and further details will be available in the extended abstract.

References

[1] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26:1781-1802, **2005**.

[2] Brian Vinter, The Architecture of the Minimum intrusion Grid: MiG, *Communicating Process Architectures 2005*, 189-202, 2005.

[3] OfficeGRID, *www.meshtechnologies.com*

[4] Harvesting Idle Windows CPU Cycles for Grid Computing, Rasmus Andersen and Brian Vinter, *in proc of GCA'06 (to appear)*, **2006**

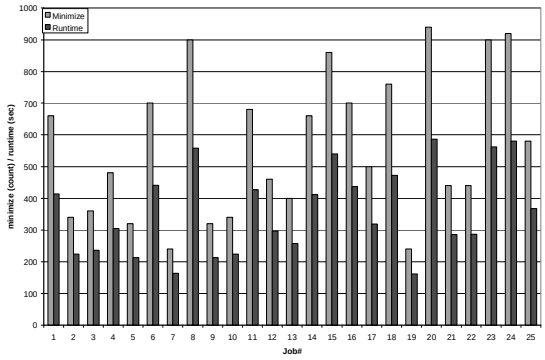

Figure 1. Distribution of job length and runtime