INSTITUTE OF PHYSICS PUBLISHING CLASSICAL AND QUANTUM GRAVITY

Class. Quantum Grav. **21** (2004) S847–S851 PII: S0264-9381(04)67766-5

A computational test facility for distributed analysis of gravitational wave signals

P Amico[1,2], **L Bosi**[1,2], **C Cattuto**[1,2], **L Gammaitoni**[1,2], **M Punturo**[1], **F Travasso**[1,2] **and H Vocca**[1,2]

[1] Istituto Nazionale di Fisica Nucleare, Sezione di Perugia, Virgo Project, I-06100 Perugia, Italy
[2] Dipartimento di Fisica, Università di Perugia, I-06100 Perugia, Italy

E-mail: michele.punturo@pg.infn.it

Received 19 August 2003
Published 10 February 2004
Online at stacks.iop.org/CQG/21/S847 (DOI: 10.1088/0264-9381/21/5/070)

Abstract
In the gravitational wave detector Virgo, the *in-time* detection of a gravitational wave signal from a coalescing binary stellar system is an intensive computational task. A parallel computing scheme using the message passing interface (MPI) is described. Performance results on a small-scale cluster are reported.

PACS numbers: 04.80.N, 95.55.Ym, 07.05.−t, 89.80.+h

1. Introduction

The gravitational signal coming from a system of two coalescing neutron stars (or black holes) is the best candidate for detection in a terrestrial gravitational wave detector. If the signal is reproduced well by its second-order post-Newtonian (PN) approximation, it is possible to implement a Wiener filtering strategy for the detection. The complete case study is realized in [1]; here we just recall the final conclusions. To perform a complete coalescing binary search in the Virgo data stream, looking at the neutron star mass range of 0.9–2.0 solar masses, with a SNR loss lower than 3%, it is necessary to build a 45 000 template grid (taking into account the two polarizations of the gravitational wave). The good seismic attenuation in the Virgo detector permits taking into account also the low-frequency part of the signal coming from such a stellar system, relatively far from the coalescence. The template cut-in frequency is about 30 Hz (100 s templates) and the cut-off frequency is 1 kHz (equivalent to 2 kHz of sampling frequency). The in-memory size of the data buffer and of each template is about 4 MB. To keep in memory (RAM) the complete template grid needed is about $45\,000 \times 4$ MB $= 180$ GB of data. In an in-time processing scheme the analysis system must compute about 300 templates per second. It is clear that the required computational power is quite significant; hence, a distributed computing philosophy, based on a Beowulf cluster of PCs, is adopted.

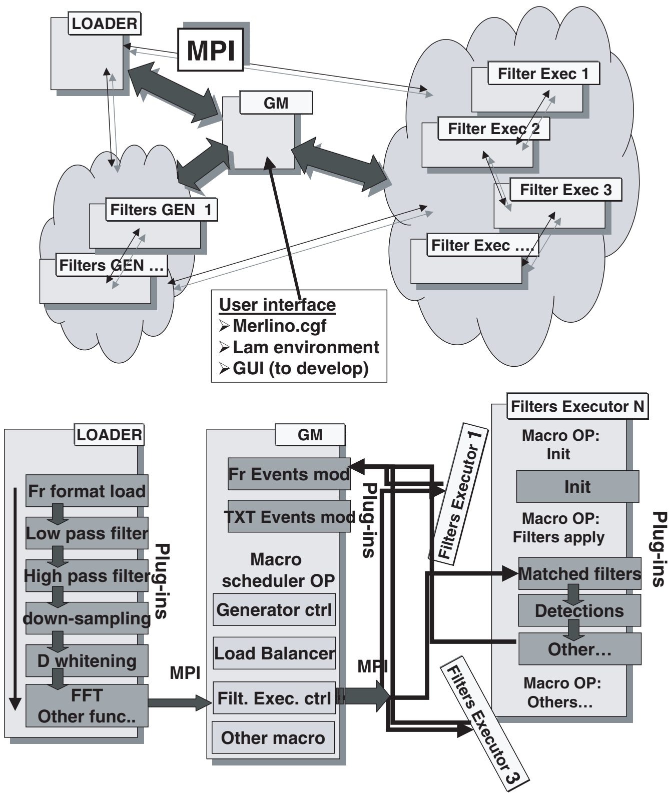

Figure 1. Distribuited program (Merlino) schemes.

2. Hardware and software environment

Two Beowulf clusters have been implemented, the older one based on dual processors Pentium III 1 GHz Intel computers (133 MHz bus), and the newer cluster based on dual Pentium Xeon 1.7 GHz Intel computers (400 MHz bus). The operating system environment and the software libraries used for this study are given below.

- GNU/Linux OS (Debian and RedHat-based)
- GNU compiler toolchain (gcc–3.2)
- LAM-MPI 6.5.9
- FFTW 2.1.3 – 3.0
- Libraries Siglib 6, Fr v6r07, Frv v4r02

2.1. The matching engine

The detection software environment (Merlino, see figure 1) is composed of four main processes communicating through message passing interface (MPI) primitives, in a master–slave configuration: the *Controller*, a process for the whole cluster; the *Loader*, a process for the whole cluster; the *Group Manager*, one for each active master in the cluster; the *Filter Executor*, one or more processes for each Group Manager.

Data are read from the Virgo data stream (currently from disk storage) by the Loader process. This process is also responsible for signal conditioning (whitening [2], downsampling

Table 1. Total computation time and single contributions in the matched filter implementation; the overall time has been evaluated in a realistic detection scheme (about 100 s of template length, several templates to be processed) and each contribution has been measured by check-pointing in the code the different phases and measuring the effective time spent in the computation.

	Total time (ms)	Array product	rFFT	Max detection
PIII	238	13.4%	75.3%	11.3%
Xeon	181	4.1%	94.8%	1.1%
Xeon (FFTW 3.0)	106	8.0%	88.0%	4.0%

to 2 kHz), pre-analysis (i.e., data quality vetoing) and transforming into the frequency domain by means of a fast Fourier transform. The output of the Loader module is sent to the Group Manager, which dispatches it to the Filter Executor (FE) processes belonging to the same MPI communicator. Each FE process is responsible for a predetermined set of templates, kept in physical memory (RAM), against which the signal is matched. The output of the Wiener filters computed by the FE processes is sent to the Controller process. This process regulates the data fluxes and provides continuous access to the results of the above signal processing chain. In the case that the matched filter output is higher than a defined threshold, the FE also executes a χ^2 reconstruction over several frequency bands [3]. This algorithm is used as a second-level trigger in the coalescing binaries signal detection.

It is important to note that the Merlino matching engine, thanks to the *plug-in* philosophy, can easily accomodate different filtering algorithms to detect different signals. In the second Virgo Mock Data Challenge [3], Merlino has been used to implement the mean filter for the impulsive signal search.

3. Performances of the system

For evaluating the difference in performance of the old and new clusters it is necessary to discriminate the different contributions to the CPU load. The matched filter can be seen as composed of three main steps: array product, inverse real Fourier transform (rFFT) and detection of the largest peak in an array of data (*max. detection*). In the first two data rows of table 1, the contributions of each of the above steps to the overall computation time are reported for the two node types. An overall performance gain of only 1.3 (=238/181) can be observed, together with an abnormal weight of the FFT computation time. In the Xeon architecture, this is due to the lack of specific Xeon optimizations in the FFTW 2.1.3 code. With the P4 (FFTW 3.0) specific version of the library the CPU load sharing between the different parts of the matching filter is recovered.

In both the clusters we realized, dual CPU nodes have been used to afford a higher density. This choice can have memory bottleneck issues: concurrent memory access by the two CPUs places high demands on the memory bus, effectively starving the older Pentium-III architecture. As shown in figure 2, Xeon-based nodes, in contrast, are able to gracefully handle the load.

3.1. Dimensioning the Beowulf cluster

From figure 3, it is possible to extract the number of templates that each Xeon-based node is able to compute in the current configuration: with a data buffer of 4 MB, a single processor machine is able to compute about 5.5 templates per second using FFTW 2.1.3 and about 10

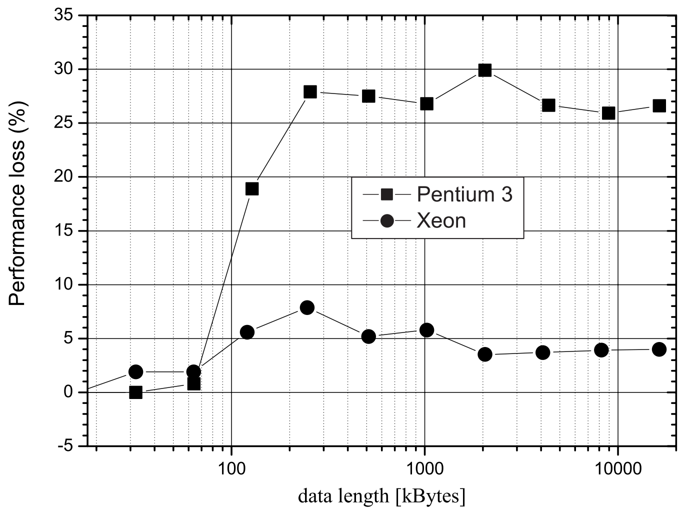

Figure 2. Loss of performance (in percentage), per CPU, for FFT computation in a dual processor machine with respect to a single processor machine. Intel Pentium III is indicated with a full square and Intel Xeon with a full circle.

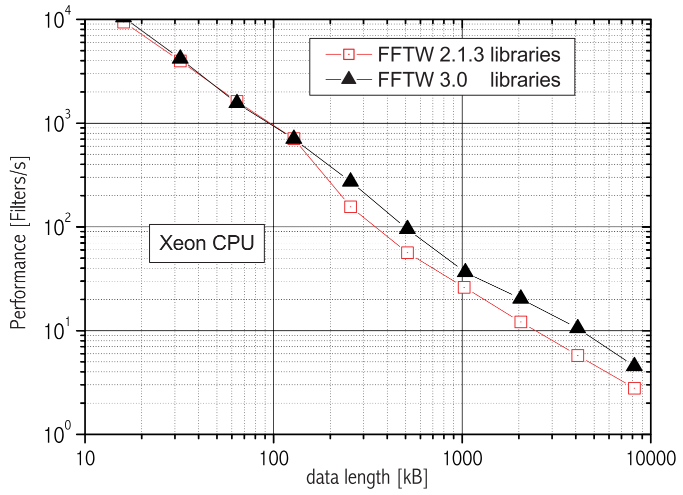

Figure 3. Number of computed templates per second, in a *single* Xeon processor machine, using the older FTTW 2.1.3 version (empty square symbol) or newer P4 optimized FTTW 3.0 (full triangle symbol).

templates using the FFTW 3.0, Xeon optimized. In a dual CPU configuration a relative loss per CPU of about 12% is expected. Using FFTW 3.0, keeping the detection *in time* with the data acquisition, each node is able to evaluate about 1500 templates (825 using the older FFTW 2.1.3, 1300 templates per CPU in a dual configuration) so that about 30–35 Xeon CPUs are needed to evaluate the whole template grid. Each node must have at least 7 GB of RAM (14 GB in the case of dual processor nodes). Since it is necessary to evaluate, for each over-threshold matched filter, the χ^2 filter on several frequency bands, an additional computational load is present. If the same cluster is used to perform this second-level filter, essentially a doubling of the machine number is expected.

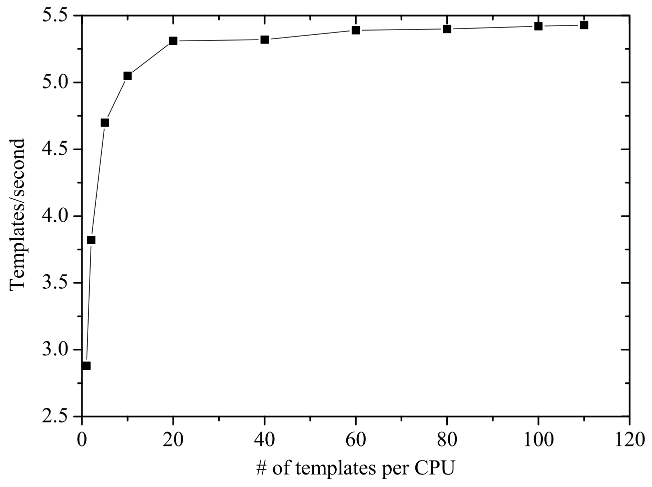

Figure 4. Intensive matched filtering test on a small-scale prototype based on Xeon CPUs and standard FFTW 2.1.3 libraries.

4. Small-scale prototype

A small-scale software detection prototype was implemented using eight dual-processor Xeon machines connected through a fully-switched Gigabit network. To avoid bus contention and to make the performance analysis easier, one of the two processors in each machine has been left idle while the other was performing matched filtering. Standard FFTW 2.1.3 libraries have been used. Results are reported in figure 4, where it is clear that a plateau of 5.5 templates per second is already reached with the number of templates per CPU lower than the 825 required in the full-scale computation. The performance loss shown for few templates per CPU is due to the increasing impact of network communication overheads in those conditions.

5. Conclusions

The feasibility of an *in-time* detection system for coalescing binaries signals in the Virgo interferometric gravitational wave detector has been demonstrated. A small-scale Beowulf cluster of Intel Pentium Xeon processors, running MPI-based parallel matched filtering software, has been realized and tested on the Virgo (CITF) engineering runs data [3]. In this configuration, for detecting stars whose mass ranges from 0.9 to 2.0 solar masses, using a PN 2 spinless template approximation and χ^2 reconstruction for the matched templates, a larger cluster with about 70 Xeon CPUs is required.

Acknowledgment

This research has been supported by the Istituto Nazionale di Fisica Nucleare (INFN) under the Virgo Project.

References

[1] Amico P *et al* 2003 *Comput. Phys. Commun.* **153** 179–89

[2] Cuoco E *et al* 2001 *Class. Quantum Grav.* **18** 1727

[3] Acernese F *et al* 2004 Search for inspiralling binaries events in the Virgo Engineering Run data *Class. Quantum Grav.* **21** S709