JOURNAL OF COMPUTATIONAL PHYSICS **139,** 15–34 (1998)
ARTICLE NO. CP975857

A Real-Time Parallel Application:

The Detection of Gravitational Waves by a Network of Heterogeneous Workstations

Stefano Marano,* Mario Medugno,† and Maurizio Longo‡

**Dipartimento di Ingegneria Elettronica, Università degli Studi di Napoli "Federico II," Napoli, Italy;*
†C.N.R., Centro di Ricerche per il Calcolo Parallelo e i Supercalcolatori (CPS), Italy;
‡Dipartimento di Ingegneria dell' Informazione ed Ingegneria Elettrica,
Università degli Studi di Salerno, Salerno, Italy
E-mail: marano@diesun.die.unina.it

Received February 12, 1997; revised September 18, 1997

We deal with the detection of gravitational chirp signals among noisy data, where the reception and the detection are piped and run in parallel. We consider the classical theory of signal detection, which yields a detector with a "bank-of-filters" structure. We investigate distributed network computing in order to implement such a detector by heterogeneous high performance workstations interconnected via an Ethernet network. The goal is to design a distributed detector running on a number of available workstations. The computation is decomposed across the workstations in such a way to minimize communications and to match the acquisition rate. Our approach is general and can be used for networks of workstations different from those used in our experimentation. We point out that the classical performance analysis seems inappropriate if applied to real-time detection by heterogeneous distributed systems, because the execution time requirements are disregarded. To take into account such constraints we characterize the algorithm, evaluate performances on different workstations, and propose a task decomposition strategy assigning the appropriate *Grain* to each workstation. © 1998 Academic Press

Key Words: network computing; real-time; optimal detection.

1. INTRODUCTION

Gravitational wave (GW) detection has become an important experimental research topic. Currently there are several projects in progress aiming to the realization of an interferometric gravitational antenna for the detection of GW. In particular the scope of the Italian–French VIRGO project [27] is the realization of a 3-km baseline interferometric antenna.

In order to detect the emission of GW, we need a suitable filtering algorithm for real-time processing of the data from the interferometric gravitational antenna. Several detection algorithms have been proposed in [24], and a parallel implementation of a bank of filters on a transputer-based computer has been discussed in [21]. More recently, a survey of a number of statistical signal processing approaches to the detection of GW signals has been published in [28]. Finally, the topic of noise level estimation in interferometric antennas has been addressed in [13].

In this paper we consider the processing algorithm of the signals expected from a gravitational interferometer. We discuss an algorithm based on the *likelihood ratio* (LR) test and the complexity of the resulting bank of matched filters; then we determine the memory and the computational power required for processing data with the same acquisition rate. Such hardware resources are not available on a serial computer; therefore we adopt a distributed approach in which several branches of the bank are processed over a single high performance workstation in order to implement a detector prototype; a dedicated parallel architecture is suitable to implement the detection over large parameter ranges and long runs of experiments.

The paper is organized as follows: in Section 2 the problem of the detection of a gravitational signal with chirp-like waveform is mathematically formulated. Section 3 describes the computational approach of the matched-filters bank and some aspects of the algorithm implementation as optimization, complexity analysis, evaluation of the necessary computing resources. Section 4 deals with the design of the distributed algorithm, the performance issues emphasizing the real-time constraints and the domain decomposition function, the model of the software performance on different workstations with UNIX operating system.

2. MATHEMATICAL FORMULATION OF THE PROBLEM

In this section we briefly describe in a mathematical framework the steps involved in the detection of a GW signal. First the signal model, which defines the noise-free antenna response to a gravitational wave, is introduced. Then a hypothesis test is employed to detect the presence of a GW signal in the noisy input; according to the Neyman–Pearson approach, we should compare the likelihood ratio (LR) with a threshold value, but our signal, and consequently the LR, depends upon some unknown parameters. The *generalized likelihood ratio test* (GLRT) approach requires that the unknown parameters be their maximum likelihood estimates, which jointly maximize the LR. We impose necessary conditions for the maximum of the LR over all unknown parameters of the signal. It results that two parameters only can be analytically evaluated so that substitution into the LR can be performed; the maximum over the other two parameters can be found only numerically by an exhaustive search.

2.1. *The Signal Model*

An interferometric gravitational antenna potentially explores a frequency band from about 10 Hz to a few KHz and provides an accurate control of seismic, thermal, and photon noise. Among several types of GW signals, a class of chirp waveforms is expected.

The chirp waveform is emitted by a binary star system. Neglecting tidal effects, the eccentricity of the orbit, the Doppler effect due to earth motion and the relativistic correction, the noise-free antenna response to a GW that is the useful signal can be cast in the form [20, 14]

$$\begin{aligned} s(t) &= IA(t, t_a, \tau)\cos(\beta(t, t_a, \tau) + \phi) \\ t &\in (t_a, t_a + \tau), \end{aligned} \tag{1}$$

where

$$A(t, t_a, \tau) = \tau^{-1} f_0^{-2}\left(1 - \frac{t - t_a}{\tau}\right)^{-1/4},$$
$$\beta(t, t_a, \tau) = \frac{16}{5}\pi f_0 \tau\left[1 - \left(1 - \frac{t - t_a}{\tau}\right)^{5/8}\right].$$

Here I is an amplitude factor depending upon the amplitude and polarization of the wave, and from the geometry of the system, f_0 is the minimum frequency detectable by the antenna (in VIRGO $f_0 = 10$ Hz), ϕ is the initial phase, and t_a is the arrival time. The “sweep time” τ represents the signal duration, and in turn depends upon the “mass parameter” $\mathcal{M}$ of the binary system and upon f_0, through

$$\tau = \frac{3}{100^{-8/3}}\left(\frac{\mathcal{M}}{M_\odot}\right)^{-5/3} f_0{}^{-8/3}\ \mathrm{s}, \tag{2}$$

where $M_\odot$ is the solar mass unit [20, 11].

The chirp signal in the form (1) is an amplitude and frequency modulated signal, the instantaneous amplitude and frequency being monotonically increasing functions. At the instant $t_a + \tau$ in which the system collapses (collapse time), both the instantaneous amplitude and frequency diverge. In fact, the instantaneous frequency of the chirp, which is related to the revolution frequency of the binary system, is [25]

$$f(t) = f_0\left(1 - \frac{t - t_a}{\tau}\right)^{-3/8}\mathrm{Hz}. \tag{3}$$

The GW signal has, therefore, a known shape, but it depends upon the values of the unknown parameters I, ϕ, t_a, τ. Moreover, the signal provided by the interferometric antenna includes noise from several sources in addition to, possibly, the useful signal $s(t)$.

2.2. *Statistical Detection in the Presence of Additive White Gaussian Noise*

We introduce a statistical hypothesis test [8] to detect a gravitational wave in the noisy signal received by the antenna.

Letting $(0, T)$ be the observation interval and $v(t)$ be the signal received by the antenna in the observation interval, the problem can be formalized as the choice between two alternative hypothesis H_0 and H_1 [10]:

$$\begin{aligned} H_1 &: v(t) = s(t) + n(t) \\ H_0 &: v(t) = n(t). \end{aligned} \tag{4}$$

In the following we suppose that $n(t)$ is a sample function from a Gaussian white process with power spectral density $N_0/2$.[1]

[1]A whitening filter can be employed if the receiver is subject to a colored noise (see [10]).

The LR for the problem at hand can be written as [10]

$$\Lambda[v(t), \mathbf{\Theta}] = \exp\left\{ \frac{2}{N_0} \int_0^T s(t, \mathbf{\Theta}) v(t) dt - \frac{1}{N_0} \int_0^T s^2(t, \mathbf{\Theta}) dt \right\}, \tag{5}$$

where $\mathbf{\Theta}$ is the signal parameter vector $\mathbf{\Theta} = (I, \phi, t_a, \tau)$.

Under the Neyman–Pearson criterion, that is to maximize the probability of detection (P_d) for fixed false alarm probability (P_{fa}), the optimum hypothesis test would be, assuming that the useful signal be known,

$$\Lambda \overset{H_1}{\underset{H_0}{\gtrless}} \lambda.$$

As the useful signal $s(t) = s(t, \mathbf{\Theta})$ depends upon unknown parameters $\mathbf{\Theta}$, we use the GLRT approach [26], which consists in substituting the unknown signal parameters with their maximum likelihood estimates over the data $v(t)$. We search for the values of $\hat{\mathbf{\Theta}}$ that jointly maximize the likelihood ratio, that is, $\hat{\mathbf{\Theta}}_{\mathbf{ML}}$ such that:

$$\Lambda(v(t), \hat{\mathbf{\Theta}}_{\mathbf{ML}}) := \max_{I,\phi,t_a,\tau} \Lambda(v(t), I, \phi, t_a, \tau). \tag{6}$$

The hypothesis test then becomes

$$\max_{I,\phi,t_a,\tau} \Lambda(v(t), I, \phi, t_a, \tau) \overset{H_1}{\underset{H_0}{\gtrless}} \lambda. \tag{7}$$

In order to remove the parameters I and ϕ from (7), we impose the necessary condition for the relative maximum, obtaining the system of equations

$$\begin{aligned} \frac{\partial}{\partial I} \Lambda(v(t), I, \phi, t_a, \tau) &= 0 \\ \frac{\partial}{\partial \phi} \Lambda(v(t), I, \phi, t_a, \tau) &= 0 \\ \frac{\partial}{\partial t_a} \Lambda(v(t), I, \phi, t_a, \tau) &= 0 \\ \frac{\partial}{\partial \tau} \Lambda(v(t), I, \phi, t_a, \tau) &= 0 \end{aligned}$$

which in fact can be solved with respect to I and ϕ. On the other hand, the functional (7) is highly multimodal with respect to t_a and τ; hence, a search must be carried on. The test can be recast as

$$\max_{t_a,\tau} \frac{R^2(t_a, \tau)}{N_0 \Omega(\tau)} \overset{H_1}{\underset{H_0}{\gtrless}} \lambda, \tag{8}$$

where

$$\frac{R^2(t_a, \tau)}{\Omega(\tau)} = \left| \int_0^T \frac{A(t, t_a, \tau)}{\sqrt{\Omega(\tau)}} e^{j\beta(t,t_a,\tau)} v(t) dt \right|^2$$

is the square modulus of the cross-correlation of the “raw data” $v(t)$ with the complex and normalized expected signal. The terms $A(t, t_a, \tau)$ and $\beta(t, t_a, \tau)$ were defined in Section 2.1. In the square modulus of the previous expression, we can recognize a convolution between $v(t)$ and a suitable complex filter matched to $s(t)$, having impulse response

$$h(t, \tau) = \frac{A(t_a + \tau - t, t_a, \tau)}{\sqrt{\Omega(\tau)}} e^{j\beta(t_a+\tau-t,t_a,\tau)}, \tag{9}$$

where $t \in (0, \tau)$. Moreover, we get

$$\Omega(\tau) = \int_0^T \frac{s^2(t, \boldsymbol{\Theta})}{I^2} dt \approx \frac{1}{f_0^4 \tau} \tag{10}$$

and the energy of the signal (1) is $\mathcal{E} = I^2\Omega = I^2/(f_0^4\tau)$.

3. THE BANK-OF-FILTERS DESIGN

In this section we describe the design of the algorithm, and the requirements of computational resources for real-time detection.

3.1. *The Computational Approach*

In order to numerically implement the test, we have to discretize the problem. For this purpose we sample the input signal $v(t)$ at a suitable rate $f_c = 1/t_c$. As the spectrum of the GW chirp (in the stationary phase approximation) assumes its maximum at $f = f_0$ and decays like $f^{-7/6}$, $f_c = 1000$ Hz is an appropriate choice.

The values of the sweep time τ depend on the physical values for the mass parameter and f_0. Using the range of mass parameter given in [20], the "full range" in our setup is $\tau \in (5\text{ s}, 1.44 \times 10^4\text{ s})$.

In the discrete time we substitute the arrival time t_a and the sweep time τ with adimensional discrete values $n_a = t_a f_c$ and $w = \tau f_c$, where $n_a \in \sigma = \{0, 1, 2, \ldots, (T-1) f_c\}$, $w \in \alpha = \{w_{\min}, w_{\min} + \Delta w, w_{\min} + 2\Delta w, \ldots, w_{\max}\}$. Moreover, we replace the filter $h(t, \tau)$ with the numerical equivalent

$$h(m, w) = t_c h(m t_c, w t_c). \tag{11}$$

The test (8) can then be written in the form

$$\max_{(n_a \in \sigma, w \in \alpha)} |y(n_a, w)|^2 \underset{H_0}{\overset{H_1}{\gtrless}} \lambda, \tag{12}$$

where

$$y(n_a, w) = \sum_{m=0}^{(T-1) f_c} h(m, w) v(n_a + w - 1 - m)$$

is the convolution at time $n_a + w$, between the input sequence $v(m)$ and the FIR filter $h(m, w)$.

The detector decides about the presence of a gravitational wave at intervals of T seconds (decision interval), until the processes are stopped at an undefined time. This implies that in each decision interval, a double maximization over n_a and w must be performed on a wide grid of values. We choose to search first for $\max_{n_a} |y(n_a, w)|^2 = |y(\hat{n}_a, w)|^2$, and then to search for $\max_w |y(\hat{n}_a, w)|^2 = |y(\hat{n}_a, \hat{w})|^2$. In such a way the detector has been decomposed as a *bank of filters* structure, each filter being tuned to a specific value of w (hence of τ) according to (11).

The distance between the values of w of two adjacent filters is $\Delta w = \Delta\tau f_c$; an important result is that such spacing can be chosen approximately constant [20, 19]. If Δw were equal to 1 (the ideal minimum spacing), searching over the full range of τ would require about 10^7 filters. As this complexity is unaffordable a compromise must be taken. In the following we assume $\Delta w = 240$; such a value reasonably bounds the number of filters and assures a sufficient detector sensitivity [19].

The data relevant to this investigation must be analyzed without interruption over a substantial amount of time, possibly of the order of several months. Moreover, there is no natural windowing of data as data acquisition is not interleaved with data processing; this requires that all processes to be performed run in parallel. In particular, such is the case of the processes performed at each detector branch, consisting of FIR filtering of the nonfinite data stream. Such processing can be conveniently carried out in the frequency domain by cyclic convolution. We focus on the fast convolution method known as the overlap-save algorithm (OVS) [17, 15], instead of the overlap-add algorithm, which requires a greater computational effort.

With reference to Fig. 1, the input data sequence is segmented into blocks of length l; the block Mseg with length $n = l + w - 1$ is formed by adding $w - 1$ points of the

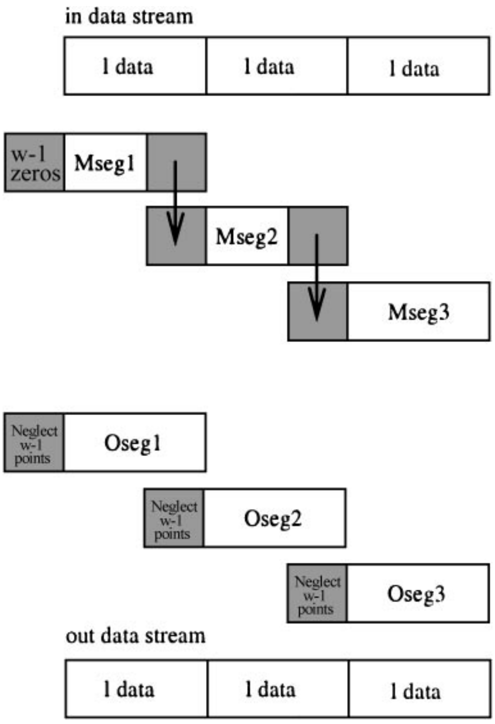

FIG. 1. Data segmentation in the overlap and save method.

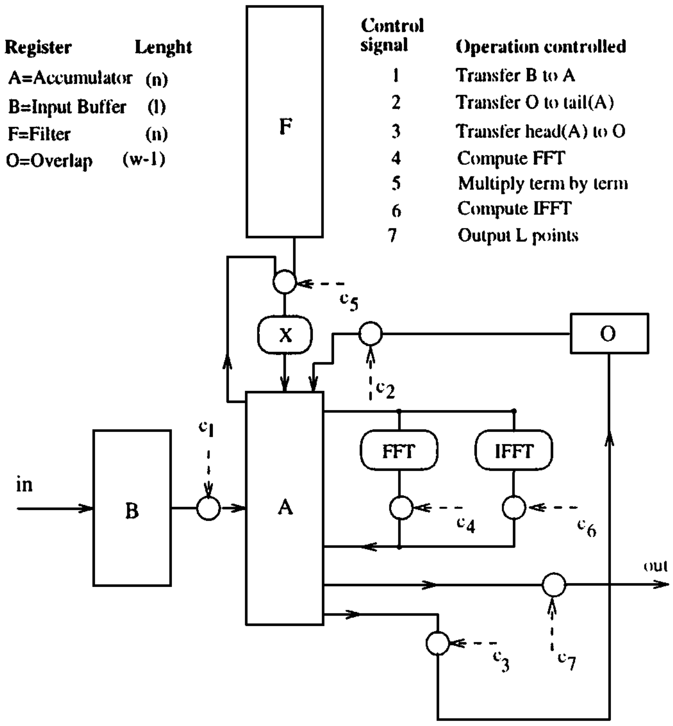

FIG. 2. Hardwired control unit for the convolution by overlap-save.

previous block to the current one. The block Oseg is the IDFT of the sequence obtained by pointwise multiplying the DFT of Mseg and the DFT of the filter sequence $h(m, w)$, which is computed once and for all. The last l samples of the block Oseg represent the convolution output data for the current cycle.

A modular design technique is applied for the bank of filters realization; a possible scheme of a hardwired control unit for the convolution is depicted in Fig. 2. Note that the sizes l, w, and n of the memory registers depend on the branch we consider; the registers A and B contain temporary data, while F and O contain respectively the samples of the filter transform and the data to be overlapped.

By replicating the registers F and O, several convolutions can be performed by a single control unit. This is the modular approach we exploit in a distributed detector algorithm to be detailed later.

To measure the time complexity of a method, we consider the number of operations involved in the processing of a single input sample—we call it ISC (input sample complexity)—instead of the classic one based on RAM [2],

The basic floating-point operation (FLOP) count is that referred to an FFT performed on n complex points, or n-FFT for short. Although the exact count changes slightly, depending on the specific FFT routine, it is, in particular, for the radix-2 algorithm (that we assume from now on) in the order $O(n \log_2 n)$[1].

Letting $n = l + w - 1$, the ISC of the OVS algorithm is

$$\mathcal{C}(n) = (2n \log_2 2n)/l \tag{13}$$

[17] and if we assume $l = kw$ (k integer), so that $n \simeq w(k+1)$, the resulting ISC is $O(\log_2 w)$. With such assumptions the frequency domain convolution is more efficient than in the time domain, whose ISC is obviously $O(w)$.

3.2. *Optimization over the Number of Operations*

Summing up, each branch of the filter bank performs a convolution, a square modulus operation, and a search for a maximum. The main contribution to the complexity comes from the convolution.

We now deal with the optimization of the OVS algorithm, that is with the minimization of the ISC with respect to n.

Let us consider a certain value of w, hence a given branch in the bank, and assume $l \geq w$ for the sake of efficiency. Let us also assume $n = 2^b$ with b an integer to allow for a radix-2 FFT algorithm.

When written in terms of b, the ISC (13) becomes

$$\mathcal{C}(b, w) = \frac{2^{(b+1)}(b+1)}{2^b - w + 1}. \tag{14}$$

In Fig. 3 we present several plots of $\mathcal{C}(b, w)$ versus b, each plot corresponding to a different value of w. Each plot starts at the lowest value of b for that w, b_{low} say, which is

$$b_{low}(w) = \lceil \log_2(2w - 1) \rceil, \tag{15}$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. Let us denote by $b_{opt}(w)$ the values of b for which $\mathcal{C}(b, w)$ achieves a minimum. By comparing $\mathcal{C}(b_{opt}, w)$ (denoted by stars in Fig. 3) with $\mathcal{C}(b_{low}+2, w)$ (denoted by circles in Fig. 3), we see that a reasonable approximation for b_{opt}, assumed below in the paper, is

$$b_{opt}(w) = b_{low}(w) + 2 = \lceil \log_2(2w - 1) \rceil + 2 \tag{16}$$

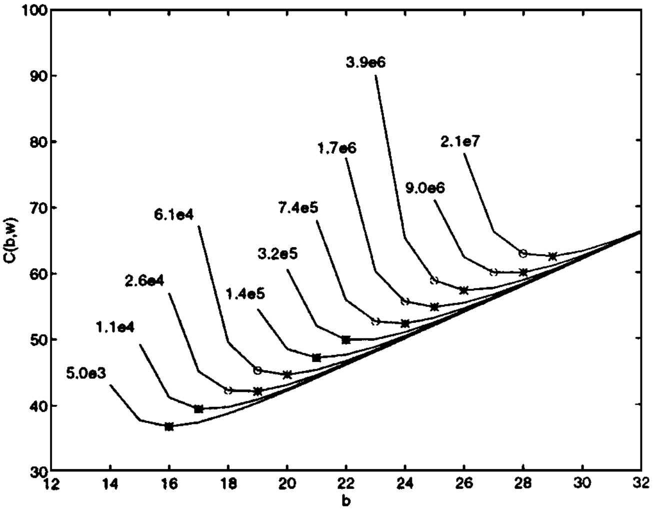

FIG. 3. $\mathcal{C}(b, w)$ versus b for several values of the parameter w.

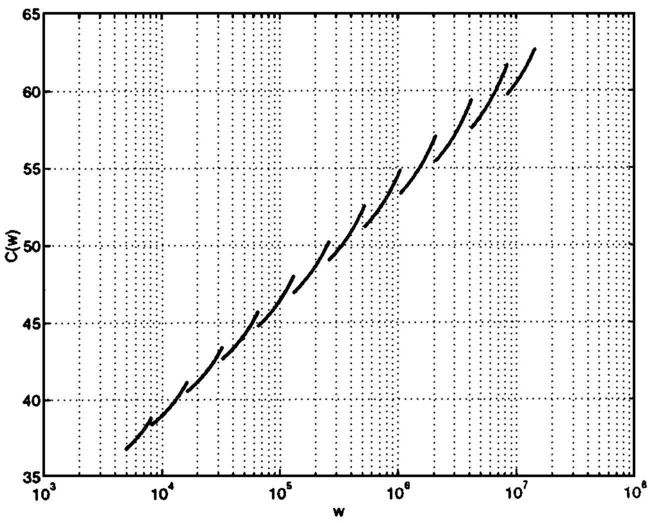

FIG. 4. $\mathcal{C}(w)$ versus w.

and, hence, with the obvious notation

$$\mathcal{C}(w) = \mathcal{C}(b_{opt}, w) = \frac{2^{\lceil \log_2(2w-1)\rceil+3}(\lceil \log_2(2w-1)\rceil + 3)}{2^{\lceil \log_2(2w-1)\rceil+2} - w + 1}. \tag{17}$$

The function $\mathcal{C}(w)$ is shown in Fig. 4; it increases with w in all the intervals in which it is continuous, while the discontinuities are due to the ceiling operators.

Notice also that the space complexity is minimum for $b = b_{low}$ and that at each increase of b by one unit, the buffer size doubles.

Now we consider the numerical round-off error deriving from the FFT algorithm for large values of n; it can be estimated by Theorem 1 in [18]. For larger values of w we have FFTs in the order of $n \simeq 10^8$ points, and the relative error increases as $O(\log_2 n)$. Using single precision arithmetic, the relative error is less than 10^{-5}, while using double precision it is less than 10^{-14}. These error values seem acceptable.

3.3. *Resource Requirements for Real-Time Detection*

We already stressed that the FFT of the filter $h(m, w)$ can be performed off-line. The time $\mathcal{T}(w)$ required to process a single data point on a single branch is then inversely proportional to F, the maximum number of floating operations per second that the machine can achieve (FLOPS), and directly proportional to the complexity of the algorithm $\mathcal{C}(w)$. Thus, introducing a factor μ to take into account the effects arising from the program–computer coupling (neglected in the previous theoretical analysis), we have

$$\mathcal{T}(w) = \frac{\mu\mathcal{C}(w)}{F}.$$

In Appendix A the following bounds are derived:

$$\mathcal{T}(w) \geq \frac{2.1\mu}{F}(4 + \log_2 w) \tag{18}$$

$$\mathcal{T}(w) \leq \frac{2.3\mu}{F}(5 + \log_2 w). \tag{19}$$

In a subsequent section we show how the value of μ changes over different machines, as well as for different values of w.

Suppose now that our bank of filters contains $NF = (w_{\max} - w_{\min})/\Delta w$ branches and that the computation is performed sequentially. Let us denote by $\mathcal{T}_{NF}$ the time required for the overall processing of a single input point; hence, $\mathcal{T}_{NF} = \sum_w \mathcal{T}(w)$, where the summation runs over the *NF* values of w. Then the real-time requirement is

$$\mathcal{T}_{NF} \le t_c \tag{20}$$

and, in turn, using the bound (19), it implies

$$F \ge 2.3\, f_c \sum \mu (5 + \log_2 w). \tag{21}$$

With $w \in [5 \times 10^3, 1.44 \times 10^7]$, $\Delta w = 240$ hence $NF \simeq 60000$, and $\mu = 1$, $f_c = 1$ KHz, we get $F = 3.8$ GFLOPS (the lower bound (18) yields $F = 3.3$ GFLOPS).

Finally, the amount of memory locations MEM required for the problem at hand is evaluated. As shown in Appendix B, the following bounds can be obtained:

$$\mathrm{MEM} \ge 12 \frac{w_{\max}^2 - w_{\min}^2}{\Delta w}$$
$$\mathrm{MEM} \le 24 \frac{w_{\max}^2 - w_{\min}^2}{\Delta w}.$$

Under the assumptions listed below Eq. (21), the memory resources required are estimated in the range $8.32 \times 10^{13} \le \mathrm{MEM} \le 1.66 \times 10^{14}$ bytes.

The current market already offers several massively parallel processors [6] achieving the theoretical power of 3.8 GFLOPS derived above for the detection over the full mass range, aggregate main memory of several GBytes, and RAID (redundant array of inexpensive disks, used for parallel file systems) mass storage of tens of TBytes. However, acquiring one such machine would require at the moment an enormous investment. This consideration motivates our further investigation of parallel implementation.

4. DESIGN AND ANALYSIS OF A PARALLEL DETECTION ALGORITHM

In this section we describe the steps involved in the design of the detection algorithm. First the reasons for the choice of the network of workstations and then the parallel algorithm are introduced; several constraints arising from such architecture are examined. The computation in (12) is decomposed across the workstations in such a way to minimize communications and to meet the requirement (20). To take into account such a constraint we characterize the parallel algorithm, evaluate the performance on different workstations, and then propose a suitable domain decomposition strategy. The performance of the parallel machine has to be measured before execution. This is a general approach which can be used both for a network of workstations and of parallel computers different from that used in our experimentation.

4.1. *The Parallel Detection Algorithm*

Our approach, motivated by the current availability of different high performance workstations connected with an Ethernet network, is to distribute the computation over several

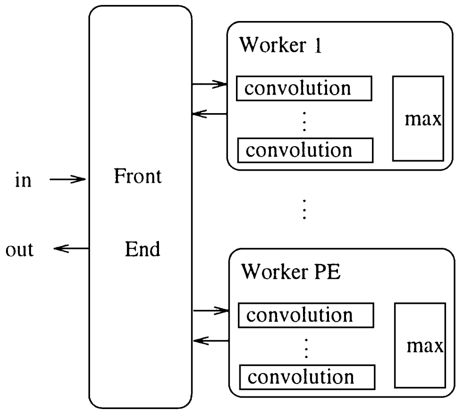

FIG. 5. Processes of the bank of filters.

such workstations. The price to be paid is the lesser mass range to be explored. Given the values of $w_{\min}$ and Δw of the set α in Eq. (12), the value of $w_{\max}$ can be chosen depending on the aggregate computational power of the workstations. Such an approach would better resist obsolescence when more efficient software and hardware tools for FFT computation are developed.

The sequential method for the detection of GW we have outlined computes a multiple kernel convolution of a nonfinite data stream. The filtering computations are independent of each other and can evolve concurrently. In order to implement it on a network of computer systems interconnected with the Ethernet network, we have to "parallelize" the algorithm. The parallel algorithm is designed for a distributed memory parallel architecture and consists of several computational processes communicating by message-passing [7]. It is obtained by the decomposition of the w-domain in Eq. (12), which just describes concurrent filtering tasks with different values of w.

Figure 5 shows the structure of the processes for the bank of matched filters. Adjacent branches of the bank can be grouped in a single process called the *worker process*, which cyclically performs the computation of a branch as the data are available. Let *PE* be the number of heterogeneous workstations, on each of which we allocate a worker process. We replicate on a worker, according to the modular approach described in Section 3.1, the filter and the overlap buffers for each branch and share the other data structures for saving memory space. The algorithm requires that data be exchanged between each filter task and a front-end process, which sends blocks of the input data to all of the computers, collects the data of local maxima, and performs the statistical test.

PVM (parallel virtual machine) [9] has been used as the message-passing environment for implementing the detector algorithm because of its stable interface with high-level languages and portability across different computers. In PVM the tasks communicate by an efficient low-level infrastructure that uses standard network protocols and the data representation XDR. Fault tolerant capability [12] and process migration [3] can be provided to PVM, augmenting the application features. PVM, as nearly all message-passing environments, does not guarantee the timing of message-passing operations.

Ethernet is appropriate for interconnecting tens of workstations for such applications, but several factors (e.g., access collisions) degrade the theoretical Ethernet performance.

Due to the limits of the network, a single filtering task cannot be further decomposed but must be executed on a single computer in order to minimize the communication needs and to guarantee the necessary bit-rate. Adjacent branches of the bank can be computed by a worker process using the same data received from the front process.

The bit-rate involved in forwarding the input data (4 bytes/sample) to all worker processes is 32 $PE\, f_c$. For $f_c = 1$ KHz, such rate is consistent with the Ethernet capacity (nominally 10^7 bps) for values of PE up to a few hundreds. Moreover, according to [23], describing the communication overheads introduced by PVM over Ethernet, the point-to-point communication time of a message Q bytes long is $t_{co}(Q) = s + \Gamma Q$, where s is a *startup* term and Γ is a *cost per byte* term. Then the front-end process has to send the data packets (of $Q/4$ samples) to all worker processes with a rate according to f_c, so $PE\, t_{co}(Q) \leq t_c Q/4$. By choosing Q such that $s \ll \Gamma Q$, we get $PE \leq 1(4\Gamma f_c)$, or we can use a virtual machine with no more than 100 workstations.

In each worker, the delay introduced by the network is made up by a queue, which buffers the data incoming from the front-end process. In such way the computation on the available data is completely overlapped with the communication of new data upon the Ethernet network.

The timing correctness of the real-time detection requires that, given a number of input samples, the processing time has to be not greater than the acquisition period. To this end, we set the execution time of a worker process, or stated otherwise, we determine how many branches of the bank a single machine is able to process under the real-time constraint. We obtain a *decomposition* function which maps the computer executing a worker process to the number of filters the worker can execute; this issue is related to the algorithm performance discussed in the next section. Now we give a pseudocode description of the parallel algorithm.

```
BANK_OF_FILTER (int PE+1, int T, float wmin, float Deltaw).
1. for $i = 2$ to PE+1 do
    (a) spawn on the $i$th processor a queue process
    (b) spawn on the $i$th processor a worker process
    (c) send to the $i$th worker process an info record with the range of values of $w$ to be
        processed
2. while (true)
    (a) while ($k \leq T f_c$)
        i. generate the $k$th data point
        ii. broadcast a packet of $Q$ data to queue processes
    (b) receive the current maximum from the workers
    (c) test the maximum over $n_a$ and $w$ with the value of $\lambda$
```

The worker algorithm can be described by the following pseudolanguage:

```
WORKER( )
1. receive the info record with the range of values of $w \in (w_1, w_2)$ to be processed
2. while (true)
    (a) for $w = w_1$ to $w_2$ step $\Delta w$ do
        i. receive $l$ data points of a segment to be processed
        ii. overlap $w - 1$ data points from the previous segment
```

iii. compute *FFT*
iv. multiply the transformed data with the template filter
v. compute FFT^{-1}
vi. on the first l data compute the module and find the maximum over n_a
vii. each T f_c points *send* the current maximum to the process BANK_OF_FILTER

4.2. *The Algorithm Performance*

The performance of a parallel algorithm for distributed memory systems with p identical processors is generally analyzed by comparing the execution times of the program derived from the algorithm to its sequential version, having fixed the size of the input data and the number of processors used [2].

Let T_p be the time required to execute the program with p identical processors. We can introduce two classic performance parameters, the speedup S and the efficiency E, where

$$S = T_1/T_p, \quad E = S/p. \tag{22}$$

Let $f_c T$ be the amount of data (the same for each filter) to be processed in $(0, T)$; then the sequential execution time is

$$T_1 = f_c T \sum_w \mathcal{T}(w).$$

We assume now that a decomposition of the w-domain, such that each worker processes $f_c T$ data in the same time, can be made. In the following we show the approach aimed to realize such domain-decomposition for our real-time problem. As the communication of the data over Ethernet overlaps with the computation, the parallel execution time is

$$T_p = T_1/p + (s_c + \Gamma_c f_c T) NF_w,$$

where s_c and Γ_c are respectively the startup and the cost per byte of communication between two processes coallocated on the same processor, and NF_w is the maximum number of filters in a worker. The speedup can be expressed as

$$S = \frac{p}{1 + [(s_c + \Gamma_c f_c T) NF_w p/T_1]}, \tag{23}$$

where in the square brackets there is a penalty factor to the linear speedup.

If we consider now a heterogeneous system [5] with PE different processors (or computers as in our case), the speedup can be defined as

$$S = \frac{T_{seq}}{\max(T^{(1)}, .., T^{(PE)})}, \tag{24}$$

where T_{seq} is the lowest time of the complete serial execution on a single processor of the heterogeneous system and $T^{(i)}$ is the time of the concurrent execution of the ith decomposition. The speedup and efficiency in (22, 24) are classical performance parameters suitable when the goal is to solve a large parallel application by several cooperating processors.

In some applications, such as real-time detection, the parallel execution time is a problem-specific requirement. The performance parameters defined above seem inappropriate in such a case, because the time constraints of the problem are disregarded. These parameters are in fact significant, provided that there exists a single processor among the processing elements performing the complete serial computation over $f_c T$ data in time less than T and a task decomposition compatible with the same time constraint.

The definition (24) can nevertheless be used as a starting point for a different approach to performance evaluation, which provides also a task decomposition complying with the time constraint.

4.3. *A Different Approach to Performance Evaluation: The Grain Parameter*

Usually the term *grain* refers to the parts resulting from the decomposition of the domain associated with a problem [7]. We introduce in this section a Grain parameter representing the number of branches of the bank that a single processing element is able to process, given the real time requirement (20).

The optimal load balance in (24) is obtained when the $T^{(i)}$'s are all equal. This involves the solution of an optimal assignment problem in which the maximum workload be assigned each worker matching the real-time constraints. The resulting values of w assigned to a worker define its Grain. In our case this ideal condition can be well approximated assigning each worker a *suitable* set of branches (i.e., values of w) which define the Grain.

However, we adopt a simplified setup. Let us consider the ratio

$$G(w) = \frac{T_c}{\mathcal{T}(w)} = \frac{F t_c}{\mu \mathcal{C}(w)}. \tag{25}$$

As $\mathcal{T}(w)$ is approximately constant when w ranges upon several Δw, we can use $G(w)$ as the number of adjacent branches in the neighbour of w that a single workstation is able to process in real time.

We remark that in such a setup the Grain of a worker depends upon the value of w and the power of the computer on which the worker is spawned. This approach requires that the performances of all machines in the network be measured prior to execution (see Eq. (26) in the next section).

Notice that if the number of branches assigned to any of the workers is larger than (25) the real time constraint is not satisfied. On the other hand, the lower the number of branches assigned to the workers, the larger the idle times.

Using the results in Section 3.3, we get (Appendix A)

$$G(w) \geq \frac{F t_c}{2.3\mu(5 + \log_2 w)}$$
$$G(w) \leq \frac{F t_c}{2.1\mu(4 + \log_2 w)}.$$

Figure 6 shows the values of G versus w, assuming as nominal values $F = 10^7$ FLOPS and $\mu = 1$.

The following section is devoted to an experimental assessment of the analytical results just derived.

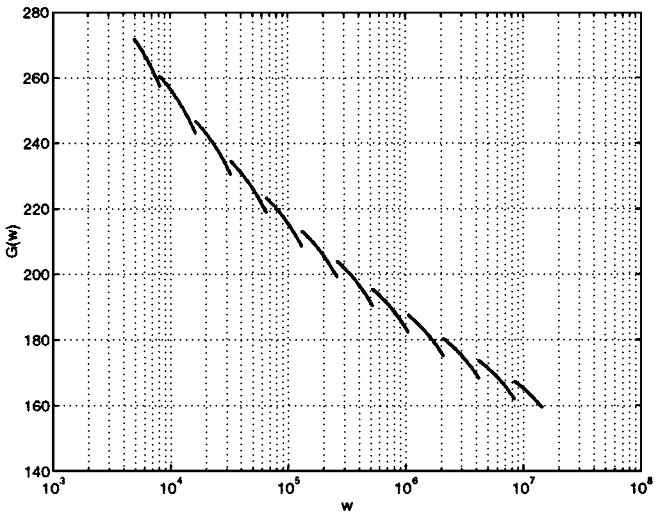

FIG. 6. Theoretical value of Grain versus w.

4.4. *Experimental Results on Unix Workstations*

The testbed used is a network of high performance workstations, SUN, DEC, and IBM, running under a Unix operating system. Our tests consist of measuring on the FFT routine the actual value of

$$\mathcal{L} = \mu \frac{10^7 \text{ FLOPS}}{F}.$$

We define such a global parameter, including the value of F and μ, since different benchmarks and FFT routines gave too different values to be useful at the design stage. Letting $G_{act} = G/\mathcal{L}$ we get the value to be used for the grain size assignment for each machine configuration.

To measure the nonideal factor we assume

$$\mathcal{L} = \frac{\hat{T}_r(w)10^7}{\mathcal{C}(w)}, \tag{26}$$

where $\hat{T}_r(w)$ is the experimentally measured value of the processing time for a single data point over the w-value branch. The experimental data consists of measuring $\mathcal{L}$ for 16 values of w in the range [5, 170] s. We implemented the bank and the measures of $\hat{T}_r(w)$ using the four1 C routine from [16] over the machines A, B, C, whose characteristics are summarized in Appendix C.

It turns out that the value of $\mathcal{L}$ deviates sensibly from a constant value. The reasons for such different behavior reside in the use of the UNIX multitasking operating system, the memory caching management, the memory paging, functional hardware units, compiler optimization features, and so on. We adopt in our investigation the linear approximation $\mathcal{L}(w) = \alpha + \gamma w$, according to the minimum square criterion. Table 1 shows the numerical values of α and γ.

Figure 7 shows for each machine the 16 experimental plots of $\mathcal{L}(w)$ and the related approximating straight line. We can see that the machines with small memory size present

TABLE 1
Values of Intercept and Slope Coefficient for the Linear Approximation of $\mathcal{L}(w)$

Machine	α	γ
A	19.49	3.27×10^{-5}
B	5.75	7.7×10^{-6}
C	4.88	6.0×10^{-5}

a step discontinuity in correspondence with a value of w for which the execution of the computational process requires virtual memory. Then a better approximation could be obtained by adopting two different linear approximations before and after the use of the virtual memory.

The plots related to small values of w of the machine C (having the larger cache memory) denote a significant caching effect that decreases the value of $\mathcal{L}$ with respect to the approximating straight line, which otherwise fits all other data plots. In fact, the computational process uses the large main memory, avoiding the paging times introduced by the virtual memory management.

Using the constant spacing of $\Delta w = 0.24$ s and a network of 10 workstations for each type A, B, and C for a total of 30 machines, the range $w \in (5, 179.7)$ s is covered. In Table 2 is summarized the number of branches allocated for each workstation type. The branches are allocated first over the C workstations, then over the A type and then over the B type.

As a final remark, we note that the parallel application is *scalable* in the sense that a larger problem size (range of w) can be addressed by a larger number of processors. However, this number cannot exceed the bound due to Ethernet network issued in Section 4.1. When no workstation is able to provide $G_{act} \geq 1$, a single convolution has to be performed in parallel

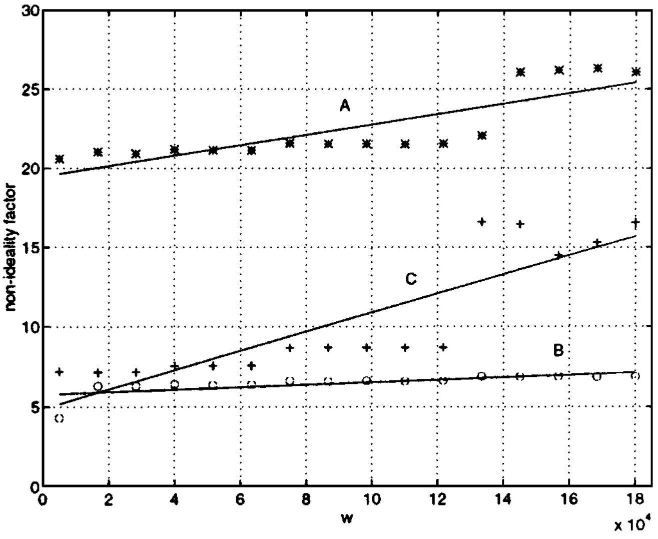

FIG. 7. Experimental values of the parameter $\mathcal{L}(w)$ versus w for several machines.

TABLE 2
Branch Allocation over a Network of 30 Workstations

Workstations	No. of branches
10 type A	100
10 type B	313
10 type C	316
Whole network	729

(using a dedicated interconnection network), because the real-time constraint cannot be satified.

5. CONCLUSIONS

In this paper the problem of detecting gravitational waves of chirp type is addressed. The classical statistical theory of signal detection yields a bank-of-filters structure for the detector, but the number of filters in the bank turns out to be very large (in the order of 10^5 filters). This task is unachievable by a single serial machine.

Our goal is to design a run-time parallel detector algorithm based on the bank-of-filters. We introduce, after algorithm characterization and performance evaluation, the so-called Grain parameter representing the number of branches of the bank that a single workstation is able to process, given real-time requirements. The parallel application scales over a network of workstations connected via an Ethernet network in the sense that the larger the aggregate computational power of the parallel machine, the larger is the mass range.

We fix the lower value of mass and the spacing among filters, as more filters are located in the lower part of the mass range. Although the number of workstations for full mass range seems to be too large for practical implementation, we stress that our results represent a worst case for (at least) two reasons. First we deal with the white noise case, in which the Δw is the smallest. In fact, our first investigations in the colored (VIRGO-like) noise indicates that such spacing increases approximately by an order of magnitude (a decrease of an order of magnitude in the number of filters). This is in agreement with the results of [22] for their different setup.

The second reason is our assumption that the range of physically relevant mass parameters is (0.25, 30) solar mass. The smaller the range the smaller is the number of allowable sweep times and, hence, the number of filters to be implemented.

As the presented experimental results are bounded to values of τ in the range of (5, 180) s, a whole range test must be performed to validate (or probably increase) the rough adopted linear approximation using a larger number of workstations. We have shown that such approximation is true also for large values of τ if the workstation memory is large.

Moreover, we carried out only an optimization over the number of operations. When the value of w becomes too large, another approach is to perform a joint optimization with respect to the number of operations and the memory requirements. To this end the work recently suggested in [4] should be of some interest. This is a topic of current work.

Finally, we stress that the bank of filters is a real-time computation intensive application whose nature is appropriate for testing the next generation of message-passing environments and real-time distributed operating systems for parallel architectures.

APPENDIX A: THE SINGLE-BRANCH COMPLEXITY

In this section we give simple bounds for the value of the complexity of a single branch of the filters bank. For

$$\begin{aligned} b_{low} &= \lceil \log_2(2w-1) \rceil \\ n &= 2^{b_{low}+2} = 4\,2^{b_{low}} \\ l &= n - w + 1 \\ \mathcal{C}(w) &= \frac{2n}{l}\log_2 2n \end{aligned}$$

we get

$$b_{low} = \lceil \log_2(2w-1) \rceil = \log_2(2w-1) + \delta \tag{27}$$

with $0 \le \delta < 1$. With simple algebraic developments we get

$$\begin{aligned} \mathcal{C}(w) &= \frac{8\,2^{\delta}(2w-1)[3+\delta+\log_2(2w-1)]}{4\,2^{\delta}(2w-1) - w + 1} \\ &\approx \frac{16}{8-2^{-\delta}}(4+\delta+\log_2 w), \end{aligned} \tag{28}$$

where the condition $w \gg 1$ has been exploited. Moreover, as $16/(8-2^{-\delta}) \in (2.1, 2.3)$ it results that

$$2.1(4+\log_2 w) \le \mathcal{C}(w) \le 2.3(5+\log_2 w). \tag{29}$$

APPENDIX B: THE SPACE COMPLEXITY

In this appendix we investigate the memory requirements for the implementation of the bank of filters. To simplify the calculations we suppose that the structure of Fig. 2 is fully replicated for each branch and assume that each memory location requires the same number of bytes.

The number of memory locations for each branch is $2n + l + (w-1) = 3n$, i.e,

$$12\,2^{b_{low}} = 12\,2^{\lceil \log_2(2w-1) \rceil}. \tag{30}$$

Assuming the bounds to b_l in Appendix A, we get the lower and upper bounds of MEM

$$24\,2^{\delta} \sum_{i=0}^{NF-1} (w_{\min} + i\,\Delta w) \tag{31}$$

TABLE 3
Characteristics of the Machine in the Testbed

Machine	A
Model	SPARCstation 10
Cpu/Arch.	SPARC/sun4
Clock (Mhz)	100
Main Mem (MBy)	32
O.S. version	SunOs 4.1.3
Manifacturer	Sun
Cache (ist./d)	64K
Machine	**B**
Model	DEC 3000-600
Cpu/Arch.	DEC 21064/AXP
Clock (Mhz)	175
Main Mem (MBy)	64
O.S. version	DEC OSF/1 3.2
Manifacturer	Digital
Cache (ist./d)	2M
Machine	**C**
Model	RS/6000-3AT
Cpu/Arch.	Power2
Clock (Mhz)	59
Main Mem (MBy)	32
O.S. version	AIX 3.2.5
Manifacturer	IBM
Cache (ist./d)	32k, 64k

respectively for $\delta = 0$ and $\delta = 1$. If $NF \gg 1$ we easily get

$$\mathrm{MEM} \geq 12\frac{w_{\max}^2 - w_{\min}^2}{\Delta w}$$
$$\mathrm{MEM} \leq 24\frac{w_{\max}^2 - w_{\min}^2}{\Delta w}.$$

APPENDIX C: CHARACTERISTICS OF THE TESTED MACHINES

We summarize in Table 3 the main characteristics of the three machines named A, B, C.

REFERENCES

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison–Wesley, Reading, MA, 1982).
2. G. S. Almasi and A. Gottlieb, *Highly Parallel Computing* (Benjamin/Cummings, Redwood City, CA, 1989).
3. J. Casas and R. Konuru, Adaptive load migration systems for PVM, in *Proceedings, Supercomputing 94, 1994* (IEEE Comput. Soc. Press, Los Alamitos, CA, 1994), p. 390.

4. H. C. Chiang and J. C. Liu, Fast algorithm for FIR filtering in the transform domain, *IEEE Trans. Signal Process.* **44**(1), 126 (1996).
5. V. Donaldson, F. Berman, and R. Paturi, Program speedup in a heterogeneous computing network, *J. Parallel Distrib. Comput.* **21**(3), 301 (1994).
6. J. J. Dongarra, H. W. Meuer, and E. Strohmeier, *TOP500 Supercomputer Sites*, Technical Report RUM33/93, Mennhaimer University, 1994.
7. G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors* (Prentice-Hall, Englewood Cliffs, NJ, 1988).
8. J. E. Freund and R. E. Walpole, *Mathematical Statistics*, 3rd ed. (Prentice-Hall, Englewood Cliffs, NJ, 1980).
9. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine a Users' Guide and Tutorial for Networked Parallel Computing* (MIT Press, Cambridge, MA, 1994).
10. C. W. Helstrom, *Statistical Theory of Signal Detection*, 2nd ed. (Pergamon, Oxford, 1968).
11. A. Krolak, J. A. Lobo, and B. J. Meers, Estimation of the parameters of the gravitational-wave signal of a coalescing binary system, *Phys. Rev. D* **48**, 8 (1993).
12. J. León, A. Fisher, and P. Steenkiste, *Fail-safe PVM: A Portable Package for Distributed Programming with Transparent Recovery*, Technical Report CMU-CS-93-124, Carnegie-Mellon University, 1993.
13. M. Longo, M. Lops, and S. Marano, CFAR detection of chirp gravitational wave signals, *Signal Process.*, submitted.
14. M. Longo and S. Marano, Locally optimum detector of chirp gravitational waves, University of Naples, Italy, Dept. of Electronic Engineering, 1995.
15. H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms* (Springer-Verlag, New York, 1982).
16. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C* (Cambridge Univ. Press, Cambridge, 1988).
17. J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing* (Macmillan Co., New York, 1986).
18. G. U. Ramos, Roundoff error analysis of the fast Fourier transform, *Math. Comput.* **25**, 116 (1971).
19. L. Salvatore, *Analisi dei ricevitori di onde gravitazionali provenienti da coalescenti binarie*, thesis, University of Naples, Italy, Dept. of Electronic Engineering, 1995.
20. B. S. Sathyaprakash and S. V. Dhurandhar, Choice of filters for the detection of gravitational waves from coalescing binaries, *Phys. Rev. D* **44**, 12 (1991).
21. B. S. Sathyaprakash and S. V. Dhurandhar, A parallel algorithm for filtering gravitational waves from coalescing binaries, *J. Comput. Phys.* **109**(2), 215 (1993).
22. B. S. Sathyaprakash and S. V. Dhurandhar, Choice of filters for the detection of gravitational waves from coalescing binaries. II. Detection in colored noise, *Phys. Rev. D* **49**, 4 (1994).
23. B. Schmidt and V. Sunderam, Empirical analysis of overhead in cluster environment, *Concurrency Practice and Experience* **6**, 1 (1994).
24. B. F. Schutz, *Gravitational Wave Data Analysis* (NATO-ASI series, 1988).
25. S. Smith, Algorithm to search for gravitational radiation from coalescing binaries, *Phys. Rev. D* **36**, 10 (1987).
26. H. L. Van Trees, *Detection, Modulation and Estimation Theory*, Vol. 1 (Wiley, New York, 1971).
27. Virgo Group, Final conceptual design, 1989.
28. P. Willett, M. Lops, and S. Marano, Detection of a long and weak signal with unknown parameter, in *CISS, 30th Conference on Information Science and Systems, Princeton, NJ, 1996.*