CALIFORNIA INSTITUTE OF TECHNOLOGY
Laser Interferometer Gravitational Wave Observatory (LIGO) Project

To/Mail Code: Distribution
From/Mail Code: Albert Lazzarini
Phone/FAX: 626-395-8444
626-304-9834

Refer to: LIGO-L970579-00-E
Date: October 29, 1997

Subject: LIGO Data Analysis Benchmarks

This note provides background information to the Technical Document T970166 (version 01, dated 10 October 1997), written by Bruce Allen.

1. The benchmark data provided by Prof. Allen (U. Wisconsin-Milwaukee) in collaboration with Caltech/CACR are *preliminary*. The version T970166-01 of the technical memorandum presents results for the IBM SP2 system at CACR which were not optimized using the IBM ESSL FFT routines. That analysis is pending and will likely result in an improvement in the SP2 performance relative to other systems.

2. The non-FFT computations associated with the detection algorithms were not optimized (they were optimized at the level of algorithms and high-level code, not at the assembler level). The proportionate amount of time spent during the benchmark tests in the non-FFT section of the analysis is consequently greater than it should be; however the fraction of time spent in this part of the analysis flow is relatively small (<20%).

3. The large difference (~ 2X) in computational requirements associated with "calculate templates on the fly" versus reading in pre-calculated and stored templates is an upper bound on the difference between these approaches, because it now appears feasible to calculate templates from a relatively small number of "primitives" which need only be rescaled according to chirp mass.

4. There may still be a ~ 10+% discrepancy between measured FFT benchmarks reported here and by Stuart Anderson using the CACR/LIGO components of hardware (SP2, Intel Paragon, SUN).

Al:al
cc:
Data group
Document Control Center
Chronological File

To/Mail Code: Albert Lazzarini/51-33
From/Mail Code: Bruce Allen, U. of Wisconsin – Milwaukee
Phone/Fax 414-229-6439/414-229-5589
Refer to: LIGO-T970166-01-E
Date: October 10, 1997
Subject: Benchmark tests for inspiraling binary searches for LDAS

Benchmark tests for inspiraling binary searches for LDAS

Bruce Allen
Department of Physics
University of Wisconsin - Milwaukee
email: ballen@dirac.phys.uwm.edu

This technical note gives results of a benchmarking study of matched filtering on interferometric data. The data was from the CIT 40-meter prototype, and the filtering was carried out using a template bank consisting of templates from inspiraling binary systems calculated in the 2nd-order post-Newtonian approximation. The results are extrapolated to obtain computer performance requirements for the LIGO's online computational analysis. For each type of system, we estimate the number of slave processors (CPUs) required to keep up with the LIGO data stream in real time using current hardware, and using hardware that might be available when LIGO begins operations in 2 years.

Contents

I. INTRODUCTION

The baseline design for the LIGO Data Analysis System calls for installing, at the detector sites, a large-scale general purpose computer. This computer must be able to carry out a search for inspiraling binary systems in real time, for a reasonable range of the parameters of such a system. While there may be novel and highly-efficient methods of carrying out this search, the only technique that can be described as well understood and well characterized is the classic technique of "matched filtering" (also called "Wiener filtering" or "optimal filtering"). This technique correlates the detector's output with a bank of templates, choosen to represent the range of possible signals with sufficient accuracy over the desired range of parameter space. The correlation is weighted in frequency space to optimize the ratio of filter output in the presence of a signal to signal output in the absence of a signal.

In order to scope the computational requirements of the LDAS design, we have benchmarked a code which carries out this type of search, using data taken in November 1994 with the Caltech 40-meter prototype interferometer. Further details about this data set and the filtering techniques may be found in the manual for the prototype data analysis

system GRASP (Gravitational Radiation Analysis and Simulation Package). The code used for this benchmarking study is the example program `multifilter` in the GRASP package.

II. STRUCTURE OF THE `MULTIFILTER` CODE

The `multifilter` code implements matched filtering through a template bank. The code uses the Message Passing Interface (MPI) standard for parallel computation, and has proven to be remarkably portable. Testing this code on another machine or platform usually takes just a few hours, most of which is spent in locating and creating the necessary libraries and tools. However the `multifilter` code itself, which is written in ANSI C with MPI calls is *unchanged* from one platform to the next and is completely portable.

The matched filtering process parallelizes trivially over either data or templates. The code that we have tested is parallelized over data only, although it could be easily parallelized over templates as well. Our implementation of this process, the `multifilter` code is structured as a master/slave system. There is one master process, and $K-1$ slave processes, for a total of K processes. Typically each process runs on a different CPU. The master process is the data server. It is responsible for (1) handing data out to the slave processes, and (2) collecting the results and writing them into files. In the `multifilter` code the master carries out the following sequence of steps:

1. Reads the necessary data and calculates the response function of the interferometer.
2. Sets up a grid of templates in parameter space, determining the masses m_1 and m_2 of the systems whose waveforms will be used as templates.
3. Broadcasts this list of templates to the slaves.
4. Acquires and buffers the next N points of gravity-wave data, and FFTs it.
5. Updates a moving (exponentially-decaying in time) average of the noise power spectrum.
6. Broadcasts the first usable segment of calibrated gravity-wave data (N floats, in frequency space) and the average noise power spectrum ($N/2+1$ floats, in frequency space) to the first slave process. The time taken in steps 4-6 is denoted $t_{\rm broad}$ below.
7. Iterates steps 4-6 until all the slaves have gotten data, or no data remains.
8. Listens for a slave to return data (the results of the filtering process). When a slave returns data, the master writes the output to a file, and then either returns to step 4 and gives the slave more data, or gives the slave a termination message if no data remains.
9. When all the slaves have finished work, and no data remains, the master process exits.

The slave carries out the following sequence of actions:

1. Collects the list of template parameters from the master.
2. Listens for $3N/2+1$ points of data and power spectrum, or a termination message. If data arrives then,
3. Calculates 0°- and 90°-phase template (if the templates are not being stored). The time taken for this is denoted $t_{\rm chirp}$ below.
4. FFTs the two different phase templates (if the templates are not being stored). The time taken for this is denoted $2\times t_{\rm fft}$ below (one FFT is required for each phase of the chirp waveform). The templates are then orthonormalized. The time taken for this is denoted $t_{\rm orth}$.
5. Correlates the data with the templates. This requires two inverse FFTs.
6. Searches the output of the correlation for the maximum Signal-to-Noise-Ratio (SNR) point. (The time taken for steps 5 and 6 is denoted $t_{\rm corr}$ below.)
7. If the SNR exceeds a preset threshold, calculate the "splitup" statistic for the alleged chirp.
8. Return a set of 11 (float) signals for each template to the master.
9. Iterate on steps 2-8.

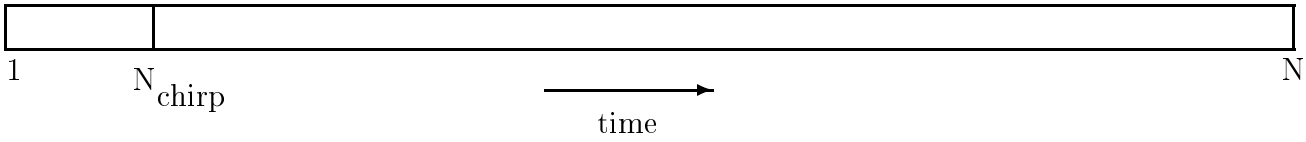
III. PROFILING TECHNIQUES AND SCALING LAWS

The `multifilter` code was instrumented using the Message Passing Extensions (MPE) logging library. Together with the `upshot` and `nupshot` MPE utilities, one can create "time-line" graphs, showing the state of all processes as a function of time. Numerous examples of these diagrams may be found in the GRASP manual. These utilities also make it simple to produce probability distributions of the time spent in any particular state (for example, doing the FFT of N points).

Based on these graphs and on the operation of the code, the performance of each tested machine may be described by the following equations. The length of the data stream searched by each iteration of the filter is

$$T_{\rm search} = \frac{N - N_{\rm chirp}}{f_{\rm samp}}. \tag{3.1}$$

Here $f_{\rm samp}$ is the instrument's sample rate (or the rate to which the data has been decimated) and $N_{\rm chirp}$ is the length of the longest chirp. The quantity $N/N_{\rm chirp}$ is sometimes called the *overlap*. Schematically:



If the templates are stored (rather than computed on the fly) then the time required to filter the set of templates is

$$T_{\rm temp-bank} = N_{\rm filter} t_{\rm corr}. \tag{3.2}$$

Here $t_{\rm corr}$ is the time required to carry out the correlation described in step 6 of the slave task-list above, and $N_{\rm filter}$ is the number of templates. If the templates are computed on the fly (rather than stored) then the time required to filter the set of templates is

$$T_{\rm temp-bank} = N_{\rm filter}(t_{\rm corr} + t_{\rm chirp} + 2t_{\rm fft} + t_{\rm orth}). \tag{3.3}$$

Here $t_{\rm chirp}$ is the time required to calculate an inspiral waveform, $t_{\rm fft}$ is the time required to FFT a set of N points, and $t_{\rm orth}$ is the time required to orthonormalize the templates (an order N operation). The number of slave processors required to keep up with the data in real time is then given by

$$N_{\rm slave} = \frac{T_{\rm temp-bank}}{T_{\rm search}}. \tag{3.4}$$

In order that a single master can broadcast data to all the slaves, the computation time required to compute the average power spectrum (one FFT) and transmit the data to the slaves must be less than the compute time of the slave:

$$t_{\rm broad} + t_{\rm fft} < T_{\rm search}. \tag{3.5}$$

Here $t_{\rm broad}$ is the time required for the master to perform any computations on the data that it is handling, and to broadcast that data to one slave. This requirement is easily satisfied. If necessary, a number of different masters can communicate data to sets of slaves which work on different banks of templates.

The tables of results given in the following section show the number of slave processors required to keep up with the data stream in real time, when the templates are (a) stored and (b) computed "on the fly". For case (a) we *assume* that the bandwidth of the template storage medium is large and that its access-time latency is small. This means specifically that the bandwidth in bytes/unit-time is greater than

$$\frac{2 \text{ phases } \times \frac{N \text{ floats}}{\text{phase}} \times \frac{4 \text{ bytes}}{\text{float}}}{t_{\rm corr}} = 8N \text{ bytes } / t_{\rm corr} \tag{3.6}$$

and that the latency is much smaller than $t_{\rm corr}$. For example, for the Beowulf system profiled in Table XIV, the storage $\Longleftrightarrow$ cpu bandwidth would need to be greater than 3 Mbytes/sec and the latency would need to be much less than 2.7 sec.

IV. PERFORMANCE ON DIFFERENT HARDWARE PLATFORMS

In this section, we show the best performance obtained on different hardware platforms, and extrapolate/scale to the initial LIGO requirements. We also comment on any apparent idiosyncrasies associated with that platform.

1. Intel Paragon

These tests were performed on the Intel Paragon at the Center for Advanced Computing Research (CACR) at Caltech. This is a 533-node machine; each node is an Intel I860 processor with a theoretical peak speed of about 70 Mflops. The majority of the nodes have 32 Mbytes of memory; some nodes have additional memory, serve as disk controllers, or provide other services.

The code was compiled with:
icc -O4 -Mvect -Minfo
and linked to the *Numerical Recipes in C* library, compiled with the same options. The public-domain MPICH implementation of the MPI standard was used. The optimized real FFT routines `scfft1d()` and `csfft1d()` from the CLASSPACK math library were used as replacements for the real FFT routine in the *Numerical Recipes* library.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$65536 = 2^{16}$	
Broadcast time/segment	$t_{\rm broad}$	538 msec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	266-395 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15800	
Real FFT time/segment	$t_{\rm fft}$	78 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	57 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	282 msec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	5.04 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	18.6 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	55 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	4	
Number of slaves (on the fly)	$N_{\rm slave}$	11	

TABLE I. Measured performance of Intel Paragon on 40-meter data set. In this table, the segment length is 2^{16} points. The data files are ligo.caltech.edu:~ballen/LOGS/PARAGON/multifilter.5.16.log and multifilter.33.96.log. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$262144 = 2^{18}$	
Broadcast time/segment	$t_{\rm broad}$	1.4 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	287-657 msec	($3.0 \rightarrow 1.0$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	38000	
Real FFT time/segment	$t_{\rm fft}$	370 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$		not done
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	1.23 sec	
Number of filters	$N_{\rm filter}$	608	covers mass range $1.0 \rightarrow 3.0$ solar masses
Search length	$T_{\rm search}$	22.7 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	747 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	1675 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	33	
Number of slaves (on the fly)	$N_{\rm slave}$	74	

TABLE II. Measured performance of Intel Paragon on 40-meter data set. In this table, the segment length is 2^{18} points. The data files are ligo.caltech.edu:~ballen/LOGS/PARAGON/binary_search.16.30.log. The number of slaves is the number of CPUs required to keep up with the data set in real time.

We can extrapolate these results to the initial LIGO requirement. In this extrapolation, we have (1) used the results of Blackburn's model for the number of templates and sample rates and (2) assumed that the FFT times scale as $N \log_2 N$.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	1024 Hz	
Segment length	N	$2^{20} = 1048576$	
Broadcast time/segment	$t_{\rm broad}$	8.6 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	1.52 sec	(1.2 solar masses)
Samples in longest chirp	$N_{\rm chirp}$	73000	
Real FFT time/segment	$t_{\rm fft}$	1.56 sec	
Orthonormalize (2 phases)	$t_{\rm orth}$	912 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	5.64 sec	
Number of filters	$N_{\rm filter}$	34323	covers mass range $1.2 \rightarrow$?? solar masses
Search length	$T_{\rm search}$	953 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	193581 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	627149 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	205	
Number of slaves (on the fly)	$N_{\rm slave}$	658	

TABLE III. Extrapolated performance of the Intel Paragon on initial LIGO data. The sample rates and segment lengths are different than the actual test. The number of slaves is the number of CPUs required to keep up with the data set in real time.

The Intel Paragon is a very powerful type of computer for this kind of problem. In particular, although its processors are only rated at 70 Mflops peak, the I860 was designed as a Digital Signal Processing (DSP) chip and has a very efficient optimized FFT routine. For example applying the standard rule that a real FFT requires $3N \log_2 N = 3.1 \times 10^6$ operations, dividing this by $t_{\rm fft}$ yields an FFT performance of 40.3 Mflops, or about 57% of peak. This is extremely efficient.

The I860 has a very slow divide operation, and the function library for the `pow()`, `sin()` and `cos()` functions is also extremely inefficient. In order to obtain the performance figures given here, the chirp generation routines were written to include in-line Chebyshev polynomial approximations for calculating trig functions, and a Chebyshev/Newton-Raphson cube-root function.

2. SGI Origin2000

These tests were performed on an SGI Origin2000 located at SGI's Houston Benchmarking and Performance Center. The machine had 8 nodes with 2 cpus each, for a total of 16 cpus. Each cpu has 4 Mbytes of cache. The total memory of the machine was 4 Gbytes, distributed as 512 Mbytes/node. Each CPU is an R10000 chip with a theoretical (peak) performance of about 390 Mflops.

The code was compiled with:
-g3 -O3 -OPT:alias=restrict -lfastm
and linked to the *Numerical Recipes in C* library, compiled with the same options. The SGI implementation of MPI was used. The optimized real FFT routines `scfft1du()` and `csfft1du()` from the SGI/Cray math library were used as replacements for the real FFT routine in the *Numerical Recipes* library.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$65536 = 2^{16}$	
Broadcast time/segment	$t_{\rm broad}$	47 msec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	27-38 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15800	
Real FFT time/segment	$t_{\rm fft}$	29 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	7 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	96 msec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	5.04 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	7.3 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	13.3 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	1.4	2 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	2.6	3 slave processors needed

TABLE IV. Measured performance of SGI Origin2000 on 40-meter data set. In this table, the segment length is $N = 2^{16}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SGI/logfile5. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$262144 = 2^{18}$	
Broadcast time/segment	$t_{\rm broad}$	366 msec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	57-80 msec	($1.6 \rightarrow 1.1$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	33000	
Real FFT time/segment	$t_{\rm fft}$	154 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	32 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	562 msec	
Number of filters	$N_{\rm filter}$	117	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	23 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	76 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	113 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	3.3	4 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	4.9	5 slave processors needed

TABLE V. Measured performance of SGI Origin2000 on 40-meter data set. In this table, the segment length is $N = 2^{18}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SGI/logfile6. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Note that although the R10000 has very fast floating point hardware, its FFT routines are not as efficient as those on the I860. In this case, carrying out $3N \log_2 N = 14.1 \times 10^6$ operations in 154 msec, the performance is 92 Mflops, or about 24% of theoretical peak performance.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	1024 Hz	
Segment length	N	$2^{20} = 1048576$	
Broadcast time/segment	$t_{\rm broad}$	1.46 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	176 msec	($1.6 \rightarrow 1.1$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	73000	
Real FFT time/segment	$t_{\rm fft}$	684 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	128 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	2.5 sec	
Number of filters	$N_{\rm filter}$	34323	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	953 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	85807 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	143195 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	90	45 nodes
Number of slaves (on the fly)	$N_{\rm slave}$	151	76 nodes

TABLE VI. Extrapolated performance of the SGI Origin2000 on initial LIGO data. The sample rates and segment lengths are different than the actual test. The number of slaves is the number of CPUs required to keep up with the data set in real time.

It is at first glance rather surprising that the performance of the Origin2000, which is a supercomputer-class machine (priced at ≈$80K/node) is not considerably faster per node in our testing than "garden variety" workstations or even Pentium Pro machines (priced at ≈$2K/node). Under more detailed examination, the reason for this is not hard to discern. The majority of the cost in the Origin2000 has gone into a memory subsystem that permits the memory assicoated with any node to be accessed at random, with small latency time, by any other node of the system. This feature is *not* shared by the networks of workstations, which must communicate by fast ethernet or ATM networks, with comparatively lower bandwidth and higher latency. The point is that the matched filtering process (at least, as implemented in `multifilter`) does not make use of the high memory bandwidth and low latency of the Origin2000, because the communication overhead is quite low. The number of bytes of data being passed to each node and the results being returned by it are quite small in comparison to the number of computations being performed by each node on that data, and there is no need for the nodes to share data with each other. It may be possible to implement the matched filtering process in a way which could take advantage of the shared-memory parallel-processing capability of the Origin2000, however there appears to be no clear advantage to this, and no obvious significant speedup. One reason that we have not done this is that there is as yet no industry standard (equivalent, say, to the MPI standard) for shared-memory parallel computation. Although there is a proposed standard emerging, this means that the form of any test code can not be the same for different shared-memory parallel-processing platforms.

3. IBM SP2

These tests were performed on an IBM SP2 located at the Center for Advanced Computing Research at Caltech. At the time that the benchmarking was performed (late March 1997), the machine had 12 CPUs of different types. Six of these nodes were faster than the other 6. In the tables given below, we have assumed that ALL the nodes of the machine are the fastest type of nodes (spin03).

The code was compiled with:
-O -lm
and linked to the *Numerical Recipes in C* library, compiled with the same options. The MPICH implementation of MPI was used. At the time that this benchmarking work was done, there was no optimized FFT routine available on the SP2, so that the FFT routine used here is the `realft()` routine from *Numerical Recipes*.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$65536 = 2^{16}$	
Broadcast time/segment	$t_{\rm broad}$	1.05 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	113-167 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15800	
Real FFT time/segment	$t_{\rm fft}$	77 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	21 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	270 msec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	5.04 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	20.7 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	49.7 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	9.8	10 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	4.1	5 slave processors needed

TABLE VII. Measured performance of IBM SP2 on 40-meter data set. In this table, the segment length is $N = 2^{16}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/IBM/multifilter.12.25.log.alog. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	1024 Hz	
Segment length	N	$2^{20} = 1048576$	
Broadcast time/segment	$t_{\rm broad}$	16.8 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	646 msec	($1.6 \rightarrow 1.1$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	73000	
Real FFT time/segment	$t_{\rm fft}$	1.54 sec	
Orthonormalize (2 phases)	$t_{\rm orth}$	336 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	5.4 sec	
Number of filters	$N_{\rm filter}$	34323	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	953 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	185344 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	445000 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	195	
Number of slaves (on the fly)	$N_{\rm slave}$	467	

TABLE VIII. Extrapolated performance of the IBM SP2 on initial LIGO data. The sample rates and segment lengths are different than the actual test. The number of slaves is the number of CPUs required to keep up with the data set in real time.

4. Sun Ultra 2 Workstations

These tests were performed on the LIGO network of Ultra 2 168 and 200 MHz machines, connected by an ATM network. These are mostly two processor machines; both processors were used on each machine. The results given in these tables are for the 200 MHz machines.

The code was compiled with:
-fast -xO4 -dalign -xarch=v8plusa -xpg
and linked to the *Numerical Recipes in C* library, compiled with the same options. The MPICH implementation of MPI was used. The optimized FFT routines `rfftf()` and `rfftb()` from the Sun Performance Library were used for the FFTs.

Description	Symbol	Value	Comments
CPUs used in testing		12	mostly 2 cpu machines
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$262144 = 2^{18}$	
Broadcast time/segment	$t_{\rm broad}$	3.1 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	206-253 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15800	
Real FFT time/segment	$t_{\rm fft}$	390 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	260 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	1.02 sec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	24.7 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	68 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	140.0 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	2.8	3 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	5.7	6 slave processors needed

TABLE IX. Measured performance of Sun Ultra 2 200 MHz workstations on 40-meter data set. In this table, the segment length is $N = 2^{18}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SUN/ULTRA2/multifilter.12.25.log. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
CPUs used in testing		12	mostly 2 cpu machines
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$1048576 = 2^{20}$	
Broadcast time/segment	$t_{\rm broad}$	11.5 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	600 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	131072	
Real FFT time/segment	$t_{\rm fft}$	1.2 sec	
Orthonormalize (2 phases)	$t_{\rm orth}$	858 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	3.24 sec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	93.0 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	214 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	491 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	2.3	3 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	5.3	6 slave processors needed

TABLE X. Measured performance of Sun Ultra 2 200 MHz workstations on 40-meter data set. In this table, the segment length is $N = 2^{20}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SUN/ULTRA2/multifilter.12.25.longlog. Apart from the number of filters, the parameters here are choosen to match the initial LIGO requirements. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	1024 Hz	
Segment length	N	$1048576 = 2^{20}$	
Broadcast time/segment	$t_{\rm broad}$	11.5 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	1.2 sec	($1.6 \rightarrow 1.1$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	73000	
Real FFT time/segment	$t_{\rm fft}$	1.2 sec	
Orthonormalize (2 phases)	$t_{\rm orth}$	858 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	3.24 sec	
Number of filters	$N_{\rm filter}$	34323	
Search length	$T_{\rm search}$	953.0 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	111289 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	255342 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	116	
Number of slaves (on the fly)	$N_{\rm slave}$	267	

TABLE XI. Extrapolated performance of Sun Ultra 2 200 MHz workstations on Initial LIGO data set. The parameters here are very minor modifications from the previous (measured) data set, because all that has changed is the number of filters. The number of slaves is the number of CPUs required to keep up with the data set in real time.

5. Beowulf (LINUX) system

"Beowulf" is the name of a project to create inexpensive supercomputer class machines by connecting together commodity hardware systems. These tests were performed on naegling, a system at the CACR, which consists of 62 Pentium Pro 200 MHz personal computers running LINUX, networked together by a pair of 32 port fast ethernet hubs.

The code was compiled with:
-O3
and linked to the *Numerical Recipes in C* library, compiled with the same options. The MPICH implementation of MPI was used. The optimized FFT routines from the FFTW package were used, with "preferences" set for a Pentium Pro chip.

Description	Symbol	Value	Comments
CPUs used in testing		16	single cpu machines
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$65536 = 2^{16}$	
Broadcast time/segment	$t_{\rm broad}$	151 msec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	41-63 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15300	
Real FFT time/segment	$t_{\rm fft}$	47 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	14 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	131 msec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	5.04 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	8.7 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	19.0 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	1.7	2 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	3.7	4 slave processors needed

TABLE XII. Measured performance of Beowulf system (networked Pentium Pro PCs running LINUX). In this table, the segment length is $N = 2^{16}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SUN/LINUX/multifilter.16.30.log.alog. The number of slaves is the number of CPUs required to keep up with the data set in real time.

Description	Symbol	Value	Comments
CPUs used in testing		16	single cpu machines
Sample rate	$f_{\rm samp}$	9868 Hz	
Segment length	N	$1048576 = 2^{20}$	
Broadcast time/segment	$t_{\rm broad}$	2.77 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	217-237 msec	($1.6 \rightarrow 1.2$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	15300	
Real FFT time/segment	$t_{\rm fft}$	966 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	226 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	2.69 sec	
Number of filters	$N_{\rm filter}$	66	covers mass range $1.6 \rightarrow 1.2$ solar masses
Search length	$T_{\rm search}$	104 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	178 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	346.0 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	1.7	2 slave processors needed
Number of slaves (on the fly)	$N_{\rm slave}$	3.4	4 slave processors needed

TABLE XIII. Measured performance of Beowulf system (networked Pentium Pro PCs running LINUX). In this table, the segment length is $N = 2^{20}$ points. The data file is ligo.caltech.edu:~ballen/LOGS/SUN/LINUX/multifilter.log.16.1048576.30.alog. The number of slaves is the number of CPUs required to keep up with the data set in real time. The parameters are comparable to those required for initial LIGO.

Description	Symbol	Value	Comments
Sample rate	$f_{\rm samp}$	1024 Hz	
Segment length	N	$1048576 = 2^{20}$	
Broadcast time/segment	$t_{\rm broad}$	2.77 sec	(data + power spectrum)
Chirp Generation (2 phases)	$t_{\rm chirp}$	230 msec	($1.6 \rightarrow 1.1$ solar masses)
Samples in longest chirp	$N_{\rm chirp}$	73000	
Real FFT time/segment	$t_{\rm fft}$	966 msec	
Orthonormalize (2 phases)	$t_{\rm orth}$	226 msec	
Correlate/search (includes 2 FFTs)	$t_{\rm corr}$	2.69 sec	
Number of filters	$N_{\rm filter}$	34323	
Search length	$T_{\rm search}$	953.0 sec	
Time to filter one segment through bank	$T_{\rm temp-bank}$	92329 sec	Stored templates
Time to filter one segment through bank	$T_{\rm temp-bank}$	179470 sec	On the fly
Number of slaves (store templates)	$N_{\rm slave}$	96	
Number of slaves (on the fly)	$N_{\rm slave}$	189	

TABLE XIV. Extrapolated performance of Pentium Pro/LINUX workstations on Initial LIGO data set. The parameters here are very minor modifications from the previous (measured) data set, because all that has changed is the number of filters. The number of slaves is the number of CPUs required to keep up with the data set in real time.

V. COMPARISON WITH ANDERSON'S FFT TESTS

The most time-consuming part of the matched filtering process is FFTs. In order to affirm the reliability of our conclusions, we have checked the processing time for real FFTs against a study done by Stuart Anderson. Anderson gave the computation speed as a FLOP rating (floating-point operations/second). The relation between computation time for real FFT and this rate is

$$t_{\rm fft} = \frac{\frac{5}{2} N \log_2(N/2) + 10N}{FLOP} \tag{5.1}$$

In two cases of interest (N=2^{16} and $N = 2^{20}$ points) this gives:

$$t_{\rm fft} = \begin{cases} \frac{3.1}{MFLOP} \text{ sec} & \text{for} N = 2^{16} \\ \frac{58}{MFLOP} \text{ sec} & \text{for} N = 2^{20}. \end{cases} \tag{5.2}$$

For example, a 116 MFLOP machine could do a real FFT of $N = 2^{20}$ points in 500 msec. Together with Anderson's measurements for the FLOP ratings of different machines with different values of N, we can use this formula to infer the FFT time and compare them with what has been found in this study. The results are summarized below:

Machine	N	Package	Anderson time	Package	This study
168 MHz Sparc Ultra	2^{20}	FFTW	950 msec	Sun Performance Lib	1200 msec (200 MHz)
200 MHz Pentium Pro	2^{16}	FFTW	57 msec	FFTW	47 msec
200 MHz Pentium Pro	2^{20}	FFTW	1137 msec	FFTW	966 msec

TABLE XV. Tables of real-FFT times measured or inferred from Anderson's work, compared with those found in this study. The times to compute FFTs are very similar.

VI. CONCLUSION

The initial LIGO data search requirements can be easily met with a large-scale parallel computer or network of workstations. The figures given here represent an honest effort to estimate the scale of computation required. I believe that they represent an upper bound on the computational power required. It may be possible to reduce the CPU power needed by further optimization of the code. However based on my profiling and optimization experience to date, I think that further gains in efficiency of perhaps 25% may be possible through low-level hand-coding of certain routines (in particular, the correlation output searching and orthonormalization modules). Larger gains in efficiency are probably not practical. Generally speaking, the computation is FFT-bound. Thus a machine with hardware that is efficient at performing FFTs will have an advantage over a machine that is not efficient at this process.

One of the reasons to put computational machinery on-site is to be able to search the data stream in real time, both to discover instrumental quirks and problems quickly, and to be able to alert the scientific community in the event of a strong signal. We note however that if the filtering is carried out in the naive way described by the extrapolations given here, the response time will be quite slow. In particular, since the search length T_{search} is about 15 minutes, the "group delay" of the filtering process is at least 15 minutes. In fact, if the filtering is parallelized only over data, then the "group delay" is typically of order 10^5 seconds, or more than 24 hours. If the filtering is parallelized over both data and templates, it should be practical to reduce this by perhaps an order of magnitude. However it will be difficult to get response times that are less than hours.

The predictions of initial LIGO's computing requirements become somewhat less daunting when we take account of the ever-increasing speed and capability of computer hardware. Some rough fits show that over the past decade the computation speed of computer hardware has scaled as

$$\text{floating point ops /second} \propto 2^{\left(\frac{t}{1.6\ \text{years}}\right)}. \tag{6.1}$$

Given that about two years remain before initial LIGO will need to have a computing engine in place, this means that we can expect the compute performance/node of the available hardware to increase by a factor of about 2.4. Much of this performance increase will probably come about through straightforward increase in CPU clock frequency, with a consequent decrease in computation time. This should reduce the number of nodes needed in two years by about the same factor ≈ 2.4.