LIGO SURF Final Paper: Cutting Edge Computing for the Extraction of Astrophysical Parameters from Gravitational-Wave Observations

Halston Lim*[1], Kent Blackburn†[2], Vivien Raymond‡[2] and Rory Smith§[2]

[1] *California Institute of Technology*
[2]LIGO SURF Advisors, *California Institute of Technology*

Abstract

Detecting gravitational waves emitted by compact binary coalescences can provide important astrophysical information about the progenitor through a process called parameter estimation. Because parameter estimation, which involves finding theoretical waveforms and posterior probability distributions, is computationally expensive, we aim to expedite the process by optimizing on Intel hardware on the Stampede computing cluster, by adapting Intel compatible compilers and functions to improve performance. Our results indicate that the Intel hardware compatible MKL FFT implementation runs an order of magnitude faster than the generic FFTW implementation. Thus, given this improvement, we investigated optimizing the LSC Algorithms Library parameter estimation code as it utilizes FFTs. By reducing the computational cost of parameter estimation via optimization on hardware, longer gravitational wave signals can be analyzed, with the potential of extracting more information about the progenitor. We also compare the performance of the reduced order quadrature method for parameter estimation, which reduced the total runtime by an order of magnitude compared to the regular quadrature method.

1 Background

1.1 Gravitational Waves

The theory of General Relativity predicts that certain astrophysical objects will emit gravitational radiation, in addition to emitting electromagnetic (EM) radiation [1, 2]. Compact binaries coalescences (CBCs), which consist of inspiraling neutron stars and/or black holes, are included in this class of gravitational wave (GW) emitting objects. However, CBC GW sources have only been indirectly detected - the first GW observation from compact binaries was made over the period of several decades (1975-2005+) by measuring the orbital decay of a binary neutron star (BNS), an effect due to energy loss via GW emission [3]. During the inspiral process

*hblim@caltech.edu
†kent@ligo.caltech.edu
‡vivien@caltech.edu
§smith_r@ligo.caltech.edu

of a binary, the emitted GW signal can be detected with advanced ground-based interferometers, which include the LIGO and Virgo detectors [4, 5, 6, 7]. The search for the predicted GW waves from CBCs is ongoing and the upcoming network of GW interferometers (Advanced LIGO, Advanced Virgo) anticipate the first direct observation of CBC GWs with an expected BNS observation rate of 40 events per year, with lower and upper bounds of .4 and 400 events per year, respectively [8].

The problem of determining source physics via GW signal is of great interest and has the potential to provide an alternative to EM astronomy. If detected, a GW signal contains information regarding the binary mass components, spins, location, and distance [9]. Learning more about the mass distribution of such objects can hint at the formation mechanism of binary systems and the nuclear equation of state governing such objects. The spins of binaries can reveal, for instance, information about the common envelope stage or supernova processes. If the location of the progenitor is able to be recovered by the GW signal, the source may be identified with EM counterpart. Alternatively, an EM observation could trigger the detection of a GW signal, given the location of the source. By using both GW and EM signals, the event's location can be better determined, which has great implications on the parameter estimation of the binary [10, 11].

The gravitational waveforms used to model the source signal can only approximate its true form, as exact solutions to Einstein's equations don't exist for such binary system. Waveforms are characterized by several non-spin and spin parameters, as determined by the complexity of the model. For example, we neglect eccentricity of the orbit because any source detected by LIGO likely has a small eccentricity. Although black holes can be described by only a finite number of parameters (charge, mass, spin), neutron stars must be approximated to avoid modeling every individual neutron in the system. Compact binary models can be described with N parameters, where $N = 9$ for the simplest non spin models:

- $\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1+m_2)^{1/5}}$ - the chirp mass of the binary, where m_1 and m_2 are the masses of the individual members of the binary (such that $m_1 > m_2$)
- $\eta = \frac{m_1 m_2}{(m_1+m_2)^2}$ - the symmetric mass ratio
- ι - the inclination angle, which is the angle between the orbital angular momentum vector and the line of sight
- d_L - the luminosity distance
- t_c - the time of binary coalescence
- ϕ_c - the phase of the waveform at time t_c.
- ψ - the polarization, which describes the rotation between the detector and source local coordinates
- (α, δ) - the celestial coordinates of the source given by right ascension and declination

Spin parameters add an extra 6 parameters to describe the spin vectors [12], and precession makes the inclination angle ι time-dependent.

Another simplification of the true waveform comes from using only a finite number of terms from the post-Newtonian expansion of Einstein's equations [12]. The relatively weak effect of

gravitational disturbances amidst detector noise makes the inference of such parameters quite challenging. In order to study how GW signals are processed by the network of interferometers, it is highly useful to utilize artifact signals, or injections [13]. An injection is a simulated GW wave based on a waveform approximation that is inserted into the detector data either through the hardware stage (simulated detector strain) or software stage (simulated into processed data) of detection. Injections can provide information of the efficiency, background, and other fundamental statistics of the analysis pipeline.

1.2 Parameter Estimation and Bayes' Theorem

After the injection is incorporated into the detector data (with either real or simulated noise), every segment that contained an injection is analyzed using the *LALInference* pipeline, software part of the LSC Algorithm Library, in order to generate the posterior probability distributions (PDFs) for each parameter in $\vec{\theta}$ for given waveform H [14]. The speed of this process has been improved greatly with more efficient computational methods, for instance, Reduced Order Modeling [15, 16]. The Reduced Order Quadrature (ROQ) MCMC used in this method has a reported speed up by a factor of 30.

The PDFs are calculated using Bayes' Theorem, which finds the probability that the injection value is $\vec{\theta}$ for a given model H,

$$p(\vec{\theta}|H,\{d\}) = \frac{p(\vec{\theta}|H)p(\{d\}|\vec{\theta},H)}{p(\{d\}|H)}. \tag{1}$$

Here $p(\vec{\theta}|H,\{d\})$, the PDF, is the probability that the injection value is $\vec{\theta}$ given the data $\{d\}$ and model H; $p(\vec{\theta}|H)$ is the prior distribution, which is the probability of $\vec{theta}$ before any observations and depends on assumptions regarding the source. The likelihood function $p(\{d\}|\vec{\theta},H)$ determines the probability of observing the data given a model H and parameter values $\vec{\theta}$. This is just the probability that the residuals, after subtracting the model, are pure Gaussian noise.

While it is extremely computationally expensive to use Bayes' Theorem to calculate every PDF manually for each value of each parameter, statistical methods are used to approximate the true underlying distribution. In order to find the PDF for a given parameter, we stochastically sample various values for the parameter and bin the values into a histogram. To sample the distributions more effectively, we implement the Markov Chain Monte Carlo method (MCMC), which has been proven to converge to the true distribution quickly as the number of samples increases [17].

1.3 Computational Aspects

In 2012, Intel began to design multiprocessor units with a new computer architecture, called the Many Integrated Core Architecture, or MIC (pronounced Mike) [18] . These processors, known as Xeon Phi, have the capability of combining the processing power of many CPUs into one chip. The MIC architecture is especially efficient at parallel processing in a computer cluster environment, and have the ability to greatly exceed current computational limits. Parameter estimation of CBCs is an example of a computationally demanding task that is well suited for parallel computing.

The implementation of MIC processors has already begun. The LIGO Laboratory at the California Institute of Technology currently utilizes the MIC architecture on several clusters. A

much larger scale implementation has been adopted by computing resources managed by the NSF funded Extreme Science and Engineering Discovery Environment (XSEDE) [19]. One of its most powerful computational resources, the Stampede computer cluster, is operated under XSEDE and utilizes a large scale implementation of the Xeon Phi processors (61 cores) [20].

Current computational limits prevent excessive sampling of the parameter space when generating GW waveforms, even when excluding spin effects. Thus, our current ability to answer astrophysical questions relating population distributions greatly relies on intelligent sampling of the parameter space. However, by modifying the post-processing pipeline to take advantage of more efficient architectures, like that of Stampede, more complex waveforms could be calculated which would increase the breadth and content of current studies. We also aim to quantify the efficiency advantage of MIC clusters over current computational limits.

2 Methods

In optimizing the GW parameter estimation code, we first investigated the theoretical improvements in speed by creating a simple benchmark program. Specifically, the program calculated the forward FFT of the box function for a given number of points in the time series (we tested only with 2^N points) (figure 1). Although the FFT algorithm is only part of the parameter estimation calculation, it can be used as an indicator of the relative speed improvements expected. The wall clock times of the FFT algorithm were measured.

Using this benchmark program, quantitative comparisons between the CPU run times of the XLAL (Ligo Algorithms Library) and MKL (Intel Math Kernel Library) FFT implementations were made, and took into account several factors, including:

- Using the Intel Math Kernel Library (MKL) (directly or with a wrapper to call from XLAL functions) vs. XLAL (LIGO Algorithm Library)
- Intel compiler vs. generic GNU compiler
- Running jobs in the login node vs. submitting to cluster via SLURM
- Using MICs vs. no MICs
- Standard threaded vs. sequential version of MKL

After determining the theoretical speed up with the FFT alone, we proceeded to test the actual parameter estimation code. The parameter estimation utilized simulated data in the Advanced LIGO-Virgo detectors and analyzed 50 injected BNS singles uniformly oriented with fixed component masses of $1.4M_{\odot}$ with SNR from 5 to 20. Taylor F2 3.5 PN (no spin model) was used to fit the waveforms. For each implementation of the parameter estimation code (i.e. the factors listed above) the LIGO Algorithms Library was rebuilt accordingly. As each parameter estimation build each analyzed the same 50 injections, the runtime distribution between builds were compared.

Finally, the reduced order quadrature method was compared to the normal quadrature method by comparing the runtimes of the parameter estimation described above.

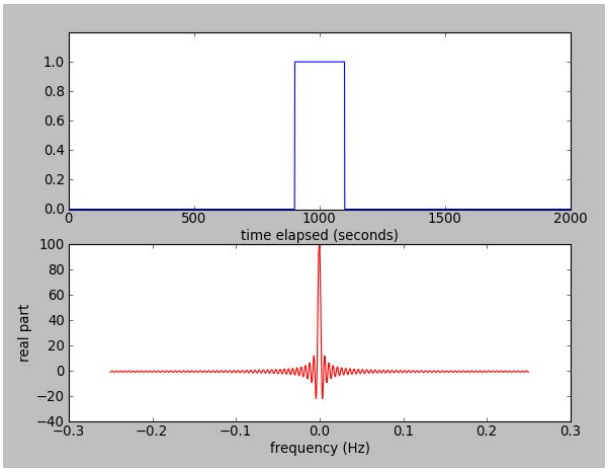

Figure 1: The fast Fourier transform algorithm in the Intel Math Kernel Library implemented on the box function and returned its Fourier transform pair, the distinctive *sinc* function in the frequency domain (top). The performance of the XLAL and MKL FFT algorithms without MIC implementation as function of number of points.

3 Results

We compared the wall clock time of calculating the forward FFT of the box function, shown in figure 2. Each series describes one implementation of the FFT, compiled with with either the Intel Compiler (icc) or the generic GNU compiler (gcc) used with either XLAL or MKL. In addition, MKL supports a parallel and sequential options, where MKL parallel uses threaded libraries and MKL sequential uses non-threaded libraries. It can be seen that all of the implementations (except the parallel MKL, mic offloaded) had runtimes that approximately scaled according to $n\log(n)$. This agrees with the theoretical scalability curve of FFTs [21].

Also, very basic MIC offloading was implemented. While the MIC implementation proved to be by far the most scalable, the overhead to simultaneously utilize the Xeon Phi processors proved significant. However, an application in which there are opportunities to compute in parallel (and not just the FFT itself) can be expected to outperform non-MIC implementations.

MKL outperformed XLAL by an order of magnitude for all N. The MKL utilizes its own discrete fourier transform function, while XLAL utilizes the FFTW implementation. Lastly, when XLAL was built with the FFT linked to the MKL libraries, a speed up was also seen. At low N, the inefficiencies in the MKL wrapper mask the speed up in the FFT.

The parameter estimation code was conducted on 50 injected BNS events to get a distribution of runtimes. In figure 3 shows the comparison between 1) the Intel compiled and GNU compiled parameter estimation on the Stampede compute cluster and 2) parameter estimation run on Stampede vs. CIT computer clusters. MIC offload and MKL implementations of the parameter estimation code were not tested, due to time limitations. Different compilers had little impact, which is an agreement with the FFT results. However, between the Stampede and CIT compute nodes, a drastic different was seen. The Stampede system allows a user to continuously utilize a node for a certain time limit (usually 2 days). This contrasts with the

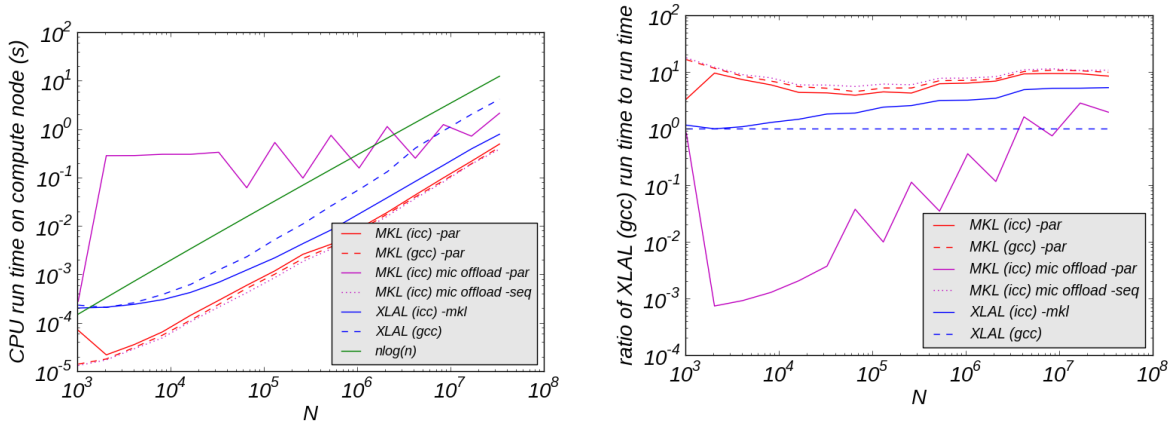

Figure 2: The CPU run time of XLAL and MKL with the Intel and GNU compilers on the compute node. While the MKL functions are still an order of magnitude faster than the XLAL functions, the compute node speeds up all processes by about a factor of 5. Also, XLAL supports a wrapper to call MKL functions (which is shown in dark, undotted blue) and has a speed increase over generic XLAL. The MIC offload runtime with parallel mode MKL is the most scalable but memory overhead dominates until $N = 10^7$. MIC offload with sequential mode provided the fastest results.

CIT system, where a user is allowed infinite time, but not guaranteed continuous node usage. In addition to faster processors, this would explain the large discrepancy.

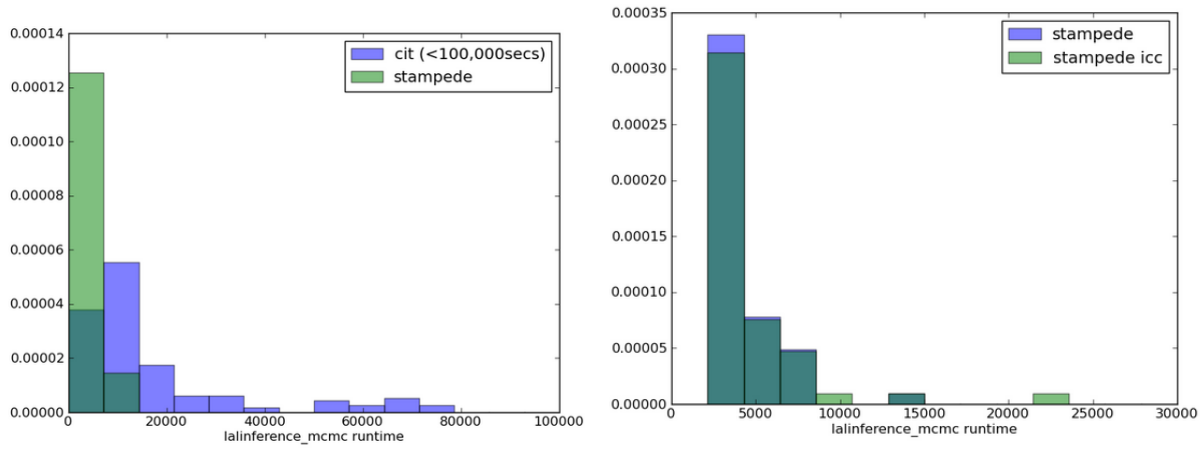

Figure 3: The parameter estimation normalized runtime distribution for 50 injected BNS events. On the left is the comparison between runtimes of CIT and Stampede and on the right is the comparison between Intel and GNU compilers. The (mean, standard deviation, median) for Stampede (icc) was (6680,13260,3702) seconds; for Stampede (gcc) was (4190,2040,3500) seconds; for CIT (gcc), for events that took less than 10^5 seconds was (17600,17800,9730) seconds.

The reduced order quadrature method was tested against the regular quadrature method in a similar fashion. Figure 4 shows the normalized histograms of runtimes on the CIT and Stampede compute nodes. An order of magnitude speed up was seen on both Stampede, and two orders of magnitude speed up was seen on the Caltech cluster.

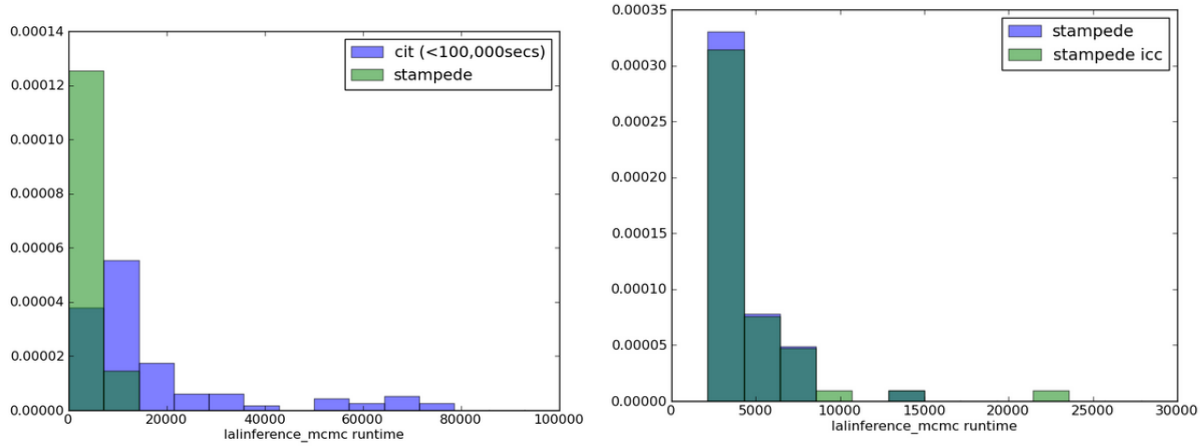

Figure 4: The parameter estimation normalized runtime distribution for 50 injected BNS events. On the left is the comparison between runtimes of CIT non roq and Stampede roq and on the right is the comparison between the Stampede non roq and roq runtimes. The (mean, standard deviation, median) for Stampede (non roq) was (4190,2040,3500) seconds; for Stampede (roq) was (330,449,218) seconds; for CIT (non roq), for events that took less than 10^5 seconds was (17600,17800,9730) seconds.

4 Conclusion

Current analyses involving calculating gravitational waveforms can be limited by both the analytical approximations and computational cost. In this project we studied how the parameter estimation code used in the LIGO analysis pipeline can be optimized.

With the FFT algorithm benchmarking results, we investigated optimization of the *lalinference_mcmc* parameter estimation application. The FFT results generated provided information of the relative speed ups expected with the parameter estimation code. As there was no speed up in FFT algorithm in switching compilers, there was similarly no speed up in the parameter estimation code. Because the MKL FFT was an order of magnitude faster than the XLAL FFT, we expect the MKL linked parameter estimation code to be faster (in accordance to FFT results). However, data wasn't collected due to time limitations. The parameter estimation code ran an order of magnitude faster on the Stampede compute node compared with the CIT compute node, demonstrating the impact of faster processors and different job schedulers. Lastly we tested the reduced order quadrature method and found drastic speed ups - order of magnitude faster than the Stampede non reduced order quadrature, and two orders of magnitude faster than the CIT non reduced order quadrature method.

One application of optimizing the parameter estimation code is that it might justify lowering the threshold frequency at which signals are detected. While the computational cost of lower the threshold detection frequency from 40 Hz to 20 Hz is high, the additional information

gained from having a longer signal may help improve parameter estimations (reduce Bayesian uncertainties). The amount by which the computational cost is reduced will can be explored by comparing the *lalinference_mcmc* code results from injection with 40 Hz versus 20 Hz threshold frequency. This remains to be done for future work.

In conclusion, our results provide insight into more optimal approaches to take for computational demanding algorithms like the discrete Fourier transform and GW parameter estimation.

References

[1] A. Einstein, Preuss. Akad. Wiss. Berlin pp. 154-167 (1918).

[2] P. C. Peters and J. Mathews, Phys. Rev. **131**, 435 (1963).

[3] R. A. Hulse and J. H. Taylor, Astrophys. J. **253**, L51 (1975)

[4] G. M. Harry and the LIGO Scientific Collaboration Class. Quantum Grav. **27**, 084006 (2010), URL http://iopscience.iop.org/0264-9381/27/8/084006/

[5] The LSC-Virgo Collaboration (In preparation), LIGO P1200087.

[6] *Advanced Virgo Baseline Design* (2009), URL https://pub3.ego-gw.it/itf/tds/file.php?callFile=VIR-0027A-09.pdf.

[7] J. Veitch, I. Mandel, B. Aylott, B. Farr, V. Raymond, C. Rodriguez, M. van der Sluys and V. Kalogera *et al.*, Phys. Rev. D **85**, 104045 (2012) arXiv:1201.1195.

[8] J. Abadie *et al.* [LIGO Scientific and Virgo Collaborations], Class. Quant. Grav. **27**, 173001 (2010) arXiv:1003.2480.

[9] J. Aasi *et al.* [LIGO and Virgo Collaborations], Phys. Rev. D **88**, 062001 (2013), arXiv:1304.1775.

[10] J. S. Bloom et al. (2009), 0902.1527.

[11] H. B. Lim and V. Raymond. (2014). *Improving gravitational-wave astronomy with electromagnetic observations of binary neutron stars*. In preparation.

[12] J. D. E. Creighton and W. G. Anderson, *Gravitational-Wave Physics and Astronomy* (Wiley-VCH, Weinheim)

[13] (2011), LIGO Scientific Collaboration and Virgo Collaboration, The LIGO / Virgo Blind Injection GW100916 (2011), http://www.ligo.org/science/GW100916/, http://www.ligo.org/news/blind-injection.php.

[14] LSC Algorithm Library, URL http://www.lsc-group.phys.uwm.edu/lal.

[15] P. Canizares, S. E. Field, J. R. Gair and M. Tiglio, Phys. Rev. D **87**, no. 12, 124005 (2013) 1304.0462.

[16] P. Canizares, S. E. Field, J. Gair, V. Raymond, R. Smith and M. Tiglio, 1404.6284.

[17] M. van der Sluys, V. Raymond, I. Mandel, C. Rover, N. Christensen, V. Kalogera, R. Meyer and A. Vecchio, Class. Quant. Grav. **25**, 184011 (2008) arXiv:0805.1689.

[18] *Xeon Phi Product Family* URL http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html

[19] *Extreme Science and Engineering Discovery Environment* URL https://www.xsede.org/

[20] *Texas Advanced Computing Center Stampede* URL https://www.tacc.utexas.edu/stampede/

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. *Numerical Recipes* (Cambridge University Press)