SETI at HOME, a Parallel Programming Paradigm

Maximilian Becker and Gautam Peri

June 3 2010

1 Introduction

Recent innovations in modern technology have made it technically feasible to answer the question that has captured the imagination of mankind since we first looked to the skys: are we alone? SETI, the Search for Extra-Terrestrial Intelligence, is a non-profit research organization that has pooled the resources and brainpower of scientists around the globe to answer this question. Since 1985 SETI employees and volunteers have developed projects to analyze cosmic electromagnetic signals in hopes of finding transmissions from an intelligent alien civilization, obtaining sponsorship and funding from large corporations, scientific foundations, and U.S. government agencies. One experiment, SETI@Home, has allowed computer users to donate unused CPU cycles to aid in this analysis. Touted as the largest distributed computation project in existence, SETI@Home spawned the Berkeley Open Infrastructure for Network Computing (BOINC), a platform that is now widely used for many other scientific projects reliant on volunteer resources. Users run BOINC and the SETI@Home project in the background of their normal processing activities and during idle processor time with the SETI@Home screensaver application. This program allows full utilization of the computational resources provided by the user without ever causing an inconvenience. With SETI@Home, individuals have the pleasure of aiding in the quest for interstellar companionship while participating in one of the largest parallel computation projects ever designed.

2 Background/Oveview

In the 20th century, advances in radio technology gave scientists the chance to probe for electromagnetic signals that would further our understanding of the universe. The Arecibo radio telescope in Puerto Rico, built in 1963 as a cooperative effort between Cornell University and the National Science Foundation, was designed to study these signals, and still stands as the largest radio telescope on Earth. SETI@Home utilizes a fraction of the Arecibo telescopes observational time, passively gathering data while the telescope is not used for other scientific endeavors.

The scientists at SETI gather and analyze data based on a set of assumptions and restrictions. Because it is more feasible to send an intergalactic message over a narrow frequency band (considering power constraints and noise issues), scientists postulate that an intelligent species would deliberately concentrate their signal. This means the data can be quantized over specific frequency ranges, and analyzed for signal strength. To account for terrestrial electromagnetic signals, SETI further distinguishes meaningful signals as those that rise and fall in intensity over a 12 second period, or the time it takes for the telescope to scan a portion of the sky. Accommodating for frequency changes due to doppler shifting is also necessary, as is capturing digitized, or chirped data.

Over the course of 2 years, the telescope scans its visible portion of the sky 3 times, generating massive amounts of electromagnetic data. This data is stored on 35 giga-

byte DLT tapes, each holding 15.5 hours of data using 2-bit complex samples (Korpela). These tapes are then sent to Berkeley, where the data is split into fixed-size work units and sent to SETI@Home users over the internet.

Because the data is finite and can be quantized, distributing it to users is simple. The information analyzed from the telescope is centered at the 1420 MHz Hydrogen line, within a frequency range that is banned for use by man-made transmissions. The band collected is 2.5 MHz wide, enough to accommodate for the relative doppler shift of intergalactic bodies. This band is broken up into 256 chunks, each around 10 kHz wide, and users are sent 107 seconds of this data. Paired with additional protocol data, each work unit ends up being 340 kilo-bytes. SETI@Home machines are sent work units from the Berkeley servers when they are idle, perform the necessary analyses, and send the data back.

3 Why the need for parallel processing?

The SETI@Home project involves real-time analysis of mountains of data, becoming much more difficult than finding a needle in a haystack. Because of the vast diversity and inherent weakness of potential signals, SETI would require massive amounts of computational resources to accomplish the task of finding an extra-terrestrial transmission. For this reason, they rely on distributed computation. As described above, the raw data is easily quantized according to the necessary restrictions, and the problem becomes embarrassingly parallel. Clients are free from communicating with eachother, only sending and receiving data from the server when necessary. Redundancy can be implemented to account for malicious or erroneous client results, and the trick becomes systematically gathering and categorizing the data after it has been analyzed.

The SETI@Home project is inherently parallel in nature, and has adapted to changes in consumer computing potential. Users with NVIDIA GPUs can take advantage of their processing power with a recent version that utilizes CUDA to improve computational performance up to 10 times that of a standard CPU. We will focus on SETI@Homes use of the CUDA language to increase parallelism, as well as the potential problems in their approach.

4 How the problem is parallelized

As discussed above, SETI analyzes a frequency range of 2.5 MHz. SETI@Home begins by splitting up that band into 256 manageable chunks of 9766 Hz (or approximately 10 KHz), each of which amounts to about 107 seconds of data. Sampling at the Nyquist rate of 20 kbps, each chunk occupies about 0.25 megabytes of memory. Each of these chunks is called a "work-unit" that is then sent to the participating users for processing. Along

with the work-unit is transmitted information about the work-unit and the necessary processing to be performed. Each user will receive about 340 kbytes of data in total for each work-unit they work on. As the Arecibo telescope remains fixed, the time it takes for a target to cross the beam is about 12 seconds. Thus, the expected signal that the program is looking for is a Gaussian that peaks around the six-second mark. The work-units also overlap by 20 to 30 seconds as to accomodate a 12-second margin that is in the transition.

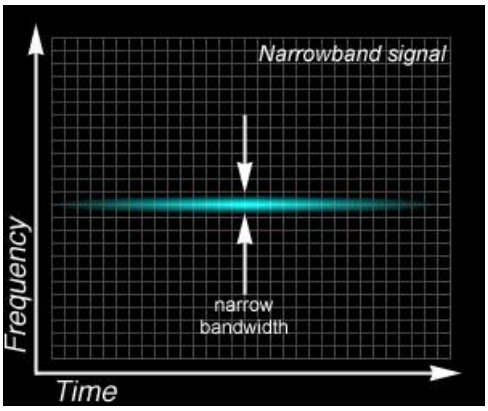
After receiving a work-unit, a user performs various tests on the data sample to find any possible signals that fit SETI's search criterion of continuous or discrete (pulsed) Gaussians (Figures 1a and 1b). As the signals are transmitted across vast distances, they are subject to the Doppler effect, or "chirping" as SETI calls it (Figures 1c and 1d). The program begins by "dechriping" the data, that is to negate the skewing of the signal from the Doppler effect. SETI neatly describes this process as follows.

> *At the finest resolution, we have to do this a total of 20,000 times, from -10 Hz/sec to +10 Hz/sec in steps of .002 Hz/sec. At each chirp-rate, the 107 seconds of data is de-chirped and then divided into 8 blocks of 13.375 seconds each. Each 13.375 second block is then examined with a bandwidth of .07 Hz for peaks*

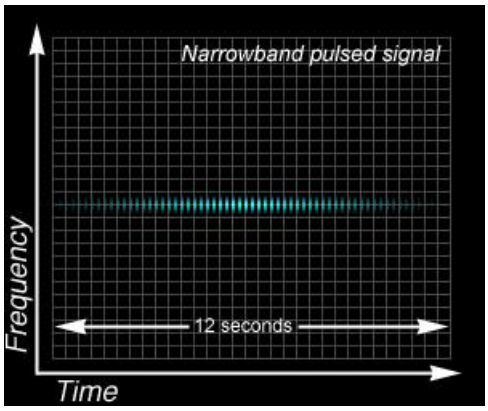
These dechriping tests are then performed in the range of ± 10 Hz/s to $\pm$ 50 Hz/s to ensure a clean signal. Once the data has been dechirped, tests are then performed at wider bandwidths at 0.15, 0.3, 0.6, 1.2, 2.5, 5, 10, 20, 40, 75, 150, 300, 600, and 1200 Hz.

SETI uses two algorithms to find pulsed signals in the data. The first, called the triplet test, looks for two pulses that are above a threshold value, and looks to find a similar pulse exactly in between the two. The second, called the "fast folding" algorithm," is a rather clever solution for finding pulses. As these pulses may be very weak, SETI breaks up the data into chunks that are analyzed with respect to time and power. Given the right period in a time-slice, if all of the slices are summed, the resulting summed power will grow and be distinct from the background noise.

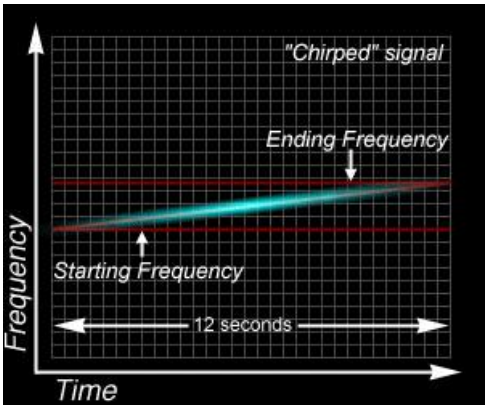
Given the nature of the problem of analyzing a band of frequencies, it is seen that the frequency ranges are independent of one another when split up into slightly overlapping chunks. This embarrassingly parallel problem is then perfect to split up into the work-units as SETI has. Given the details of the calculations that each user performs (10 to 50 hours of work, as SETI estimates), it is boldly apparent that even within work-units there is an exceeding amount of parallelism to be acheived. Parallelism is acheived at most points in the calculations, during dechirping the data, in processing FFT calculations, in fitting Gaussians to the selected data, and in finding pulses. Indeed, the CUDA version of SETI@Home acheives all of this.



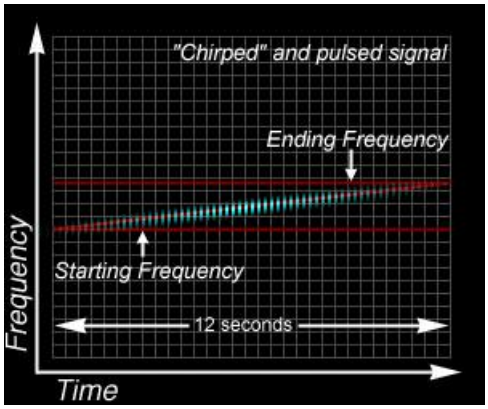
(a) Continuous Gaussian Signal



(b) Discrete Gaussian Signal



(c) Continuous Chirped Gaussian Signal



(d) Discrete Chirped Gaussian Signal

Figure 1: Types of singals the program looks for

5 Problems and Solutions

The first apparent problem with SETI's implementation of parallelizing for the CUDA architecture is in their utilization of the entirety of the GPU's resources. As shown below, all CUDA kernel calls are made with the configuration of 64 threads per block in one dimension. Similarly, the grid structure also only utilizes a one-dimensional block arrangement. Assigning 64 threads to a block indicates a close tie with the structure of CUDA's execution of 32 threads per warp. It can be seen that the grid structure is configured for a set of data points to be assigned to a block in which two warps work on the sampled set. Recommendations for improving performance in dechriping, as well as in other portions of computations, are given in the section below.

```
dim3 block(64, 1, 1);
dim3 grid((cudaAcc_NumDataPoints + block.x - 1) / block.x, 1, 1);
```

If we consider the dechirping algorithm. Each thread is used to analyze a time slice to figure out the chirp angle with respect to that slice. There is a potential for bank conflicts

in this implementation, as each 13.375-second block is accessed by many threads in the process of computing chirp angles.

```
__global__ void cudaAcc_CalcChirpData_kernel(int NumDataPoints, float chirp_rate,
float recip_sample_rate, float2* cx_DataArray, float2* cx_ChirpDataArray) {
const int i = blockIdx.x * blockDim.x + threadIdx.x;

if (i < NumDataPoints) {
float2 cx = cx_DataArray[i];
float c, d, real, imag;

float time= i * recip_sample_rate;
// since ang is getting moded by 2pi, we calculate "ang mod 2pi"
// before the call to sincos() inorder to reduce roundoff error.
// (Bug submitted by Tetsuji "Maverick" Rai)
float ang  = chirp_rate*time*time;
```

However, the issue of bank conflicts and how to avoid them are already prediagnosed by the developers at SETI in other portions of the computations. The programmers have implemented an algorithm that does the task of scanning the datapoints in $O(lg(n))$ with an option to explicitly avoid all bank conflicts (see the comments in `https://setisvn.ssl.berkeley.edu/svn/branches/seti_cuda/seti_boinc/client/cuda/cudaAcc_scanLargeArrakernel.cu`). The source code proves to be incredibly complex, utilizing advanced CUDA techniques to achieve superior parallelism.

6 Recommendations

One can look at the breakdown of the algorithms used to analyze data and find that some computations could potentially utilize more of the GPU resources. Looking at the de-chirping algorithm, we find that it is broken into 3 distinct layers of analysis: chrip-rate, granularity and bandwidth. These can easily translate into the 3 dimensions of a block, with one thread per calculation. Overall, there are 200 billion de-chirping calculations to be performed in a work unit, but further dividing these calculations among blocks could prove to maximize the GPU resource utilization and efficiency. Expanding block usage into two dimensions, we could assign multiple time-slices to individual block rows to analyze time-slices concurrently.

Another potential improvement is the use of shared memory. We observed close to no use of shared memory, and found cases (such as the de-chirping computations) where it might be appropriate. In the dechirping code, calculations access a global matrix in each thread, potentially causing a significant slowdown. Using our idea noted above, we could utilize shared memory in our 3-dimensional structure to perform concurrent calculations by copying necessary portions of the globa matrix into shared memory. Threads would then concurrently work on different chirp-rates, bandwidths, and granularities on that

portion of the matrix that is shared. Subsequent accesses to the shared portion, once it is copied from the global memory, will yield performance boosts.

Considering the limitations of shared memory in CUDA, it becomes apparent why SETI developers are reluctant to use it. With large floating point data sets, the overhead of moving data between global and shared memory reduces the benefits of shared memory use.

7 Different Architectures

One of the intriguing factors of SETI@Home is that it runs on top of BOINC, a distributed parallel platform. MPI is an alternative distributed platform that operates in a similar fashion, but they are quite different. For one, the BOINC system communicates through the HTTP protocol, as the programmers wanted no conflicts with firewalls or session interruptions. While they use different application-layer protocols, both paradigms use TCP to send and receive data. BOINC is designed to handle dynamic nodes, and distribute the data as needed. In MPI (as far as we know), you cannot accomodate for new or failed nodes at run time. You could potentially use MPI on a set of clients, developing a complex protocol to serve data and collect it on a distributed network of workstations.

In a MPI implementation, it is apparent that the server (node 0) would distribute and gather the data. We could accomodate node failure by reassigning chunks to nodes that are currently active and have finished their assigned computation. Another issue to consider is the gathering and monitoring of finished work-units. In the BOINC implementation, a work-unit is simply shipped out to a user for computation, who then returns the results to Berkeley. There is no time limit, nor there is any issue of monitoring the "end" of all computation. However, this is quite the opposite in MPI, as one would need to actively monitor the current state of all data gathered to effectively "end" the program. Pseudocode for an MPI implementation is given as follows:

```
node 0:
    q = current queue of work units
    n = # of active nodes
    for(i = 0 to n)
        send node i work unit from q
    while(q !empty || n != 0)
        for(i = 0 to n)
            receive node i state
            if(failed)
                put node i work unit back in q
                n = n - 1
```

```
            if(done)
                send request for data gather
                receive completed work unit data
                send next item from q to node i

other nodes:
    receive work unit
    during every portion of computation, send state information
    send work unit data when requested
```

Of course, this is a very simple illustration, and the MPI code would become increasingly complex to handle node failure appropriately and distribution and gathering of data.

Because of the nature of the computation required, it is much more intuitive to imagine this problem in a distributed setting than with shared memory. With shared memory, you would need massive amounts of storage for the data, but the real bottleneck is the amount of processing power as you're only limited to 4 or 8 cores (at the most), and therefore 4 or 8 concurrent threads. Instead of having hundreds of machines performing calculations concurrently, you could only split the computation between the individual CPUs at one time, yielding much less parallelism. One can imagine that with massive amounts of storage and hundreds of CPUs, SETI@Home could be implemented in a shared memory setting, but it is much more natural to place the computation in a distributed setting.

8 Conclusion

The SETI@Home project is an impressive parallel platform, utilizing distributed workstations and GPU architectures simultaneously to perform computations on a massive and ever-increasing data set. The great minds at SETI have devoted much time and effort into creating an efficient and necessarily complex program, and the user community is continuously involved in improving the platform and in aiding with computational power. SETI@Home has been an active project for more than a decade, with the intellectual community at Berkeley ever improving upon the software. The only limiting factor, it seems, is in the progress of consumer technology available to process the data ever faster. As with the advent of nVidia's CUDA, the SETI@Home team quickly incorporated the platform into their program. Further developments as such are sure to engage the community in expanding the project to allow computation on the latest and fastest technologies that enter the market. Will there be a limit to the computational power of the largest distributed computational project in existence? Will this large-scale search for extraterrestrial intelligence prove fruitful? Are we alone? The truth is out there...

9 Sources

- The CUDA source code for the SETI@Home project:
`https://setisvn.ssl.berkeley.edu/svn/branches/seti_cuda/seti_boinc/client/cuda/`

- Articles and resources that we consulted:
`http://setiathome.berkeley.edu/sah_papers/CISE.pdf`
`http://setiathome.berkeley.edu/sah_papers/cacm.php`

- SETI resources:
`http://setiathome.berkeley.edu/`
`http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_1.html`