SETI@HOME: A Parallel Programming Paradigm

Maximilian Becker and
Gautam Peri

Writers' Comment: This article was originally submitted as our final research report in Professor Norman Matloff's ECS 158 course, "Programming on Parallel Architectures." This assignment called for the analysis of a real-world application in one of three parallel programming architectures we studied, including technical background on the nature of the problem and the significance of parallel processing in its solution, recommendations for potential improvements, and equivalent implementations in alternate architectures. We chose the SETI@Home platform and its utilization of NVIDIA's CUDA GPU language because of their open-source communities and the transparency of the source code as compared to proprietary projects. We were excited about the idea of a network of individuals working together to overcome the obstacles associated with a vast amount of centralized computation, and the power of collaborative efforts as such. We would like to thank Dr. Norm Matloff for his support and infinite wisdom in the subject matter and for inspiring us to really immerse ourselves in the world of parallel programming.

—Maximilian Becker and Gautam Peri

Instructor's Comment: Countless undergraduates seeking my approval of their course plans dismissively speak of General Education courses as "just a GE course," treating any non-tech course as irrelevant. Thus this gem of a term paper by Maxx Becker and Gautam Peri comes as an affirmation of the university's goal to produce broadly-educated graduates. The assignment called for an analysis of a real-world application of our course's material on parallel programming. The students would report on the selected application, explaining the nature of the problem, the type of computation it required, and the parallel programming issues involved: why was parallel computation needed, and what difficulties arose in parallelizing it? Maxx and Gautam did a bang-up job here, explaining the issues clearly, with insight and in a highly engaging manner. I wish to emphasize that not only was theirs the best written report in the class, it was also the top in terms of technical content.

—Norman Matloff, Department of Computer Science

Introduction

Recent innovations in modern technology have made it technically feasible to answer the question that has captured humanity's imagination since we first looked to the skies: are we alone? Since 1985, the nonprofit research organization Search for Extra-Terrestrial Intelligence (SETI) has pooled the resources and brainpower of scientists around the globe to answer this question. With funding and sponsorship from large corporations, scientific foundations, and United States government agencies, SETI employees and volunteers have developed projects to analyze cosmic electromagnetic signals in hopes of finding transmissions from an intelligent alien civilization. One experiment, SETI@Home, has allowed computer clients 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. Clients 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 clients' computational resources without ever inconveniencing them. With SETI@Home, individuals have the satisfaction of aiding in the quest for interstellar companionship while participating in one of the largest parallel computation projects ever designed.

Background/Overview

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. In particular, the Arecibo radio telescope in Puerto Rico was built in 1963 as a cooperative effort between Cornell University and the National Science Foundation. It was designed to study these signals and still stands as the largest radio telescope on Earth. SETI@Home utilizes a fraction of the Arecibo telescope's observational time, passively gathering data while the telescope is not being used for other scientific endeavors.

The scientists at SETI gather and analyze data based on a set of assumptions and restrictions. For instance, it is more feasible to send an intergalactic message over a narrow frequency band. Considering power constraints and noise issues, SETI 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. SETI must also accommodate signal variability in various forms, including frequency changes due to the Doppler shift and digitized, or "chirped," data.

Over the course of two years, the telescope scans its visible portion of the sky three times, generating massive amounts of electromagnetic data. This data is stored on 35 gigabyte tapes, each holding 15.5 hours of data using 2-bit complex samples. 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 clients is simple. The information analyzed from the telescope is centered at the 1420 MHz hydrogen band, within a frequency range that is banned for use by human transmissions. The band collected is 2.5 MHz wide, which is 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 SETI@Home clients are sent 107 seconds of this data, called a work-unit. Paired with additional protocol data, each work-unit ends up being 340 kilobytes. Idle machines running the SETI@Home software are sent work-units from the Berkeley servers to perform the necessary analyses.

Why the need for parallel processing?

The SETI@Home project involves real-time analysis of "mountains of data." To use an old analogy, it would be difficult to find a needle in a haystack if you were working alone. However, thousands of people going through the same haystack are more likely to find the wayward needle. Because of the vast diversity and inherent weakness of potential signals, SETI requires massive amounts of computational resources to accomplish the task of finding an extraterrestrial 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.[1] Clients are free from communicating

1. "In parallel computing, an embarrassingly parallel workload (or

with each other, 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 lies in 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. The following analysis will focus on SETI@Home's use of the CUDA language to increase parallelism, as well as the potential problems in their approach.

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. This work-unit contains the actual data and information about the necessary processing to be performed. Each client will receive about 340 kilobytes of data in total for each work-unit they analyze. 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 6-second mark, about halfway through the process. The work-units also overlap by 20 to 30 seconds to accommodate a 12-second margin that is in the transition.

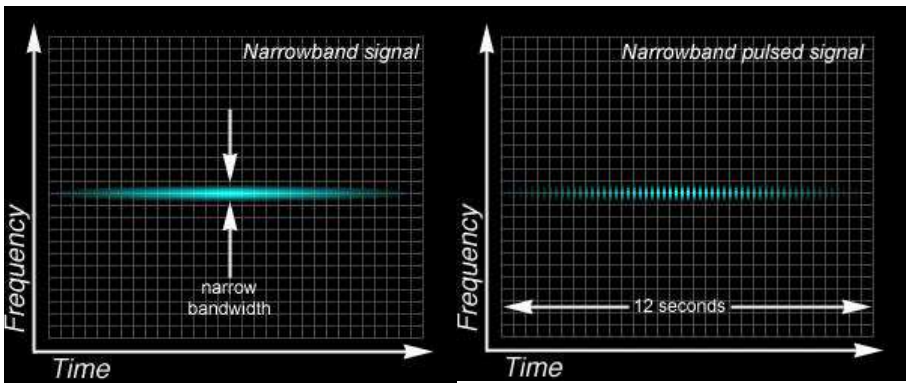
After receiving a work-unit, a client performs various tests on the data sample to find any possible signals that fit SETI's search criteria of either continuous or discrete (pulsed) Gaussians (Figures a and b). As any potential signals would be transmitted across vast distances, they are subject to the Doppler effect, or "chirping" as SETI calls it (Figures c and d). The program begins by "de-chirping" the data, negating the skew of

embarrassingly parallel problem) is one for which little or no effort is required to separate the problem into a number of parallel tasks. This is often the case where there exists no dependency (or communication) between those parallel tasks" (*Wikipedia*, s.v. "Embarrassingly parallel," http://en.wikipedia.org/wiki/Embarrassingly_parallel#cite_note-dbpp-0).

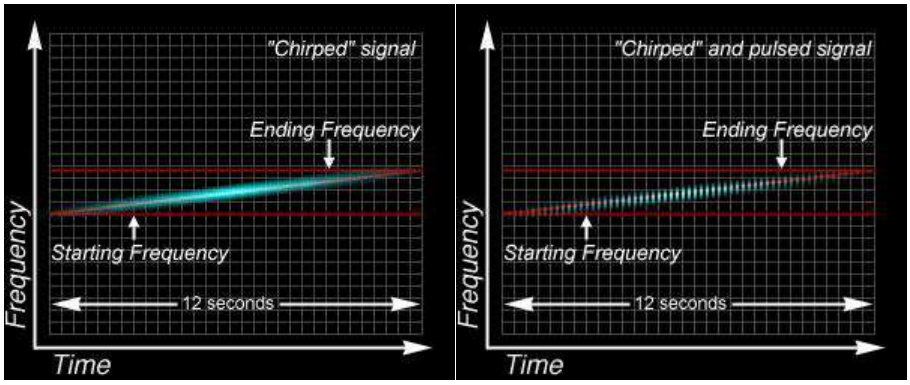
the signal caused by the Doppler effect. SETI describes this process as follows:

> 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.[2]

These de-chirping tests are then performed in the range of ±10 Hz/sec to ±50 Hz/sec to ensure a clean signal. Once the data has been de-



(a) Continuous Gaussian Signal (b) Discrete Gaussian Signal



(c) Continuous Chirped Gaussian Signal (d) Discrete Chirped Gaussian Signal

Figure 1: Types of signals the program looks for

Images courtesy Dr. David Anderson, SETI @ Home website:
http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_4.html

2. Hipschman R, 2003, http://seticlassic.ssl.berkeley.edu/about_seti_at_home_4.html

chirped, similar tests are conducted 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 seeks a similar pulse situated 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, then the resulting summed power will grow and be distinct from the background noise.

Due to the natural independence of frequencies in the electromagnetic spectrum, the frequency ranges in a certain band are mutually exclusive when split into chunks. Considering the details of the calculations that each client performs (10 to 50 hours of work, as SETI estimates), it is apparent that a significant amount of parallelism can be achieved within work-units. De-chirping the data, processing Fast Fourier Transform (FFT) calculations, fitting Gaussians to the selected data, and finding pulses all yield opportunities for parallelizing computation. Indeed, the CUDA version of SETI@Home takes advantage of these opportunities, though there are a few potential improvements to be considered.

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 utilizes a one-dimensional block arrangement. This arrangement of 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, where two warps work on the sampled set:

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

Each thread in the de-chirping algorithm 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.

Such an instance is seen in the following code, where the global variable `chirp_rate` is accessed by every thread performing this computation:

```
__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 diagnosed by the developers at SETI in other portions of the computations. The programmers have implemented an algorithm that does the task of scanning the data points in O(lg(n)) with an option to explicitly avoid all bank conflicts.[3] The source code proves to be incredibly complex, utilizing advanced CUDA techniques to achieve superior parallelism.

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. Considering the de-chirping algorithm, we find that it is broken into three distinct layers of analysis: chirp-rate, granularity, and bandwidth. These can easily translate into the three dimensions of a block, with one thread per calculation. Overall, there are 200 billion de-chirping calculations to be performed in a work-unit, and further dividing these calculations among blocks could 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 in the utilization of shared memory. Use of shared memory is rare, and we observed cases (such as the de-chirping computations) in which it might be appropriate. In the de-

[3]See the comments in the source code at: https://setisvn.ssl.berkeley.edu/svn/branches/seti_cuda/seti_boinc/client/cuda/cudaAcc_scanLargeArray_kernel.cu

chirping code, calculations access a global matrix in each thread, potentially causing a significant slowdown. With the aforementioned block configuration, we could utilize shared memory in our 3D structure to perform concurrent calculations by copying necessary portions of the global matrix into shared memory. Threads would then concurrently work on different chirp-rates, granularities, and bandwidths on the portion of the matrix that is shared. Subsequent accesses to the shared portion, once it is copied from the global memory, should yield performance boosts.

However, 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.

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. To begin with, the BOINC system communicates through the HTTP protocol, as the programmers wanted to avoid conflicts with firewalls or session interruptions. While they use different application-layer protocols, both paradigms use TCP to send and receive data. However, BOINC is designed to handle dynamic nodes and distributes the data as needed; there is no run-time accommodation for new or failed nodes in MPI. Therefore, one could potentially use MPI with a set of clients, developing a complex protocol to serve and collect data on a distributed network of workstations.

In such an implementation of MPI, it is apparent that the server (node 0) would distribute and gather the data. Node failure can be accommodated 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 client for computation, and the client then returns the results to the server. There is no time limit, nor is there 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 a potential MPI implementation is given here:

```
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 actual MPI code would become increasingly complex to handle failed nodes and the distribution and gathering of data.

Because of the nature of the computation required, it is much more intuitive to imagine SETI@Home in a distributed setting than with shared memory. With shared memory comes the necessity for massive amounts of data storage. However, the real bottleneck is in the amount of processing power available. In a multi-core machine, the number of truly parallel threads is limited by the number of processors (four or eight, given today's technology). Instead of having hundreds of machines performing calculations concurrently, the computations must be split between individual CPUs, 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. However, considering technological constraints, this problem is naturally placed in a distributed platform.

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 to creating an efficient and necessarily complex program, and the user community is continuously involved in improving the platform and in aiding with computational power. The only limiting factor, it seems, is in the progress of consumer technology available to process the data ever faster. 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 to date? Will this large-scale search for extraterrestrial intelligence prove fruitful? Are we alone? The truth is out there . . . .

Sources

Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D. SETI@home: an experiment in public-resource computing. *Communications of the ACM,* 2002, 45(11): pp. 56–61. http://setiathome.berkeley.edu/sah_papers/cacm.php

Hipschman R, How SETI@home works. SETI@home: the search for extraterrestrial intelligence, 2003, University of California, Berkeley. http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_1.html

Korpela E, Werthimer D, Anderson D, Cobb J, Lebofsky M. SETI@Home—Massively distributed computing for SETI. *Computing in Science and Engineering,* Jan/Feb 2001, pp. 78–83. http://setiathome.berkeley.edu/sah_papers/CISE.pdf

SETI@HOME, 2010, University of California, Berkeley. http://setiathome.berkeley.edu/

Source code for the SETI@Home project. set_boinc - the SETI@Home client program, 2006. https://setisvn.ssl.berkeley.edu/svn/branches/seti_cuda/seti_boinc/client/cuda/