GPU Accelerated Anomaly Detection of Large Scale Light Curves

Austin Chase Minor, Zhihui Du and Yankui Sun
Dept. Computer Science and Technology
Tsinghua University
Beijing, China
{austin.chase.m,zhihuidu}@gmail.com,syk@tsinghua.edu.cn

David A. Bader
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ, USA
bader@njit.edu

Chao Wu and Jianyan Wei
National Astronomical Observations of China
Chinese Academy of Sciences
Beijing, China
{cwu,wjy}@bao.ac.cn

***Abstract*—Identifying anomalies in millions of stars in real time is a great challenge. In this paper, we develop a matched filtering based algorithm to detect a typical anomaly, microlensing. The algorithm can detect short timescale microlensing events with high accuracy at their early stage with a very low false-positive rate. Furthermore, a GPU accelerated scalable computational framework, which can enable real time follow-up observation, is designed. This framework efficiently divides the algorithm between CPU and GPU, accelerating large scale light curve processing to meet low latency requirements. Experimental results show that the proposed method can process 200,000 stars (the maximum number of stars processed by a single GWAC telescope) in approximately 3.34 seconds with current commodity hardware while achieving an accuracy of 92% and an average detection occurring approximately 14% before the peak of the anomaly with zero false alarm. Working together with the proposed sharding mechanism, the framework is positioned to be extendable to multiple GPUs to improve the performance further for the higher data throughput requirements of next-generation telescopes.**

***Index Terms*—anomaly detection, matched filtering, GPU, performance optimization**

I. Introduction

Microlensing is a unique anomaly that occurs when a lens (or lenses) passes between a light source (star) and an observer (Earth). These lenses are high mass objects that bend the light from the source. As described in [1], this anomaly is useful in the detection of “dark” objects. Therefore, it can be used to detect objects that do not emit light. This is very useful in detecting new planets and black holes [2]. Furthermore, microlensing is a rare astronomical event, especially on the timescale of hours.

Detecting microlensing events requires super large field of view telescopes to observe millions of stars. Furthermore, for short timescale events, fast search methods and high cadence observation are required. The Ground-based Wide Angle Camera (GWAC) [3], which can cover about 5,000 degree^2 (each camera will cover up to 200,000 stars) and take photos every fifteen seconds, is one such system. It is our target optical instrument source for real time light curves used in this research. The GWAC system itself is a comprehensive pipeline. The other pipeline components before detection use roughly five seconds of processing time. Therefore, any detection method must complete within ten seconds to be considered online. Furthermore, detecting these events earlier than their peak is critical for follow-up telescopes to observe the anomaly and capture more data. Thus, an efficient anomaly detection algorithm not only should identify a microlensing event accurately but also trigger an alarm at the early phase of an event. Finally, cost efficiency is crucial to support long time (ten years) observation.

The challenge of this research lies in two aspects. One is the development of an anomaly identification algorithm that can detect short timescale microlensing events accurately from a large number of light curves with high noise. Another is the optimization of the method for detection in under ten seconds to enable real time follow-up observation. We choose to focus on one type of microlensing anomaly, the Paczyński microlensing anomaly [4]. This anomaly represents the situation of a single lens passing between an observer and a single source. This lens causes the brightness from a star to increase until peak and then back to baseline in a symmetric fashion. Considering the shape of Paczyński microlensing can be well defined mathematically, we borrow the successful idea from gravitational-wave detection [5] and design a matched filtering based algorithm because of its advantage in accurately identifying known shape signals from high noise inputs. At the same time, a GPU accelerated scalable framework is developed to handle a substantial amount of light curves in real time. The proposed scalable framework can achieve the performance required when the data generation rate is improved about fifteen times in the next-generation GWAC.

In the rest of this paper, we start with a brief discussion of related work, justifying the need for a new algorithm. We then

This research is supported in part by the Key Research and Development Program of China (No.2016YFB1000602).

describe our method, showing how it satisfies the necessary requirements of our system and achieves its performance gains. To show our method's effectiveness, we give experimental results to demonstrate its accuracy and performance. Finally, we end with a conclusion and emphasize where future work should be invested.

II. Related Work

The EWS (Early Warning System) developed for the OGLE (Optical Gravitational Lensing Experiment) [6] project is an early microlensing detection system. The algorithm was designed for detecting microlensing events in non-variable stars. Because the time scale of the target microlensing events can be as long as several weeks to even one year, the proposed method cannot be used in the short timescale (from several hours to dozens of minutes) scenario.

Some machine learning algorithms, such as Hierarchical Temporal Memory (HTM) [7] and Long Short Term Memory (LSTM) Neural Networks [8], were developed to identify anomalies. However, they are too memory and time-consuming to use in real time settings involving large scale data.

In the Korea Microlensing Telescope Network (KMTNet) project [9], a simple and quick algorithm was developed. The algorithm focuses on detecting "rising" events. If three consecutive points are beyond the given threshold, an anomaly alarm will be triggered. However, its accuracy cannot meet the requirement of the GWAC telescope.

NFD (Normalized Feature Deviation) [10] is the current, in-use method on the GWAC project, the telescope system we are targeting. Furthermore, it is a novel method for microlensing anomaly detection that has shown promise. NFD is a very lightweight algorithm so it can handle large scale light curves in real time. However, its false-positive rate is not acceptable for practical observation. Furthermore, its accuracy could be further improved.

With this information, we derive a set of qualities that a successful microlensing detection algorithm must have.

1) High accuracy (low false-positive rate and high true-positive rate).
2) High performance with low (under 10 seconds) latency.
3) Runnable on commodity hardware (cost-efficient).

III. Our Method

We design a new matched filtering based microlensing anomaly detection algorithm named PLAD (Practical Light curve Anomaly Detection). Fundamentally, our system operates in a simple manner, matching our input with the expected shape of a microlensing anomaly.

Our proposed method has two significant features. First, our matched filtering based algorithm can identify anomalies with very high accuracy. Second, our GPU accelerated, scalable framework can provide very high performance (on commodity hardware) to enable real time data processing and follow-up observation.

A. Method Overview

The data processing flow for our system is given in Fig. 1. Analyzing this dataflow, we can dissect our algorithm. First, we have two inputs: light curves and templates. These inputs are first normalized to fit our matched filter operation and increase predictive performance. With our normalized inputs, the templates are used to search the input light curves for microlensing events. The matched filter outputs (which can be seen as a form of confidence values) are input into our detector. Our detector takes these and compares them to its detection function to determine if an anomaly is occurring.

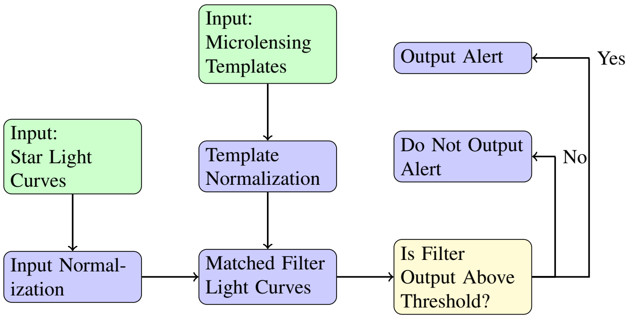

Fig. 1. System Diagram

B. Matched Filtering based Algorithm Design

1) Optimal Matching: Matched filtering is an optimal method for maximizing SNR of a template signal contained in input in the presence of AGWN (Additive Gaussian White Noise). Fundamentally, the matched filter operation needs three inputs: a template (the signal used for searching), an estimate of the AGWN noise level (for SNR scaling), and the input signal. This can be further augmented, as done in gravitational-wave research, to include a noise whitening component to transform certain kinds of non-AGWN to AGWN. With these inputs, the operation consists of computing the convolution between the reversed template and the input signal, scaled by the noise level. This creates a cross-correlation between the input signal and the template. The maximum of this calculation is the SNR of the signal in the input and corresponds to the best time localization of the template in the signal. For more information on the matched filter operation, see [11], [12].

Specifically, we use the alternative FFT-based formalism discussed in [13] and [14]. This formalism reduces the time complexity from $O(N^2)$, for naive convolution, to $O(Nlog(N))$. However, it is important to note that the $O(Nlog(N))$ has many hidden constants from the FFT operation used that can slow down processing for "small" dataset sizes. Furthermore, the formalism includes the potential to address certain types of non-AGWN. However, we modify it to only consider AGWN of $S_n(f) = 2 * \Delta f$ (where Δf is the bin width used in the FFT operation) and relax the constraint of outputting the SNR. Furthermore, we discretize it for use on data samples. Our formalism used can be seen in (1), where $a(t)$ is the input, $b(t)$ is the template, $\tilde{a}(f)$ denotes the Fourier transform of $a(t)$, and $\tilde{a}^*(f)$ denotes the complex conjugation of $\tilde{a}(f)$.

$$\langle a(t)|b(t)\rangle = \Re\left[\sum_{0}^{\lceil N/2\rceil} \tilde{a}(f)\tilde{b}^*(f) + \tilde{a}^*(f)\tilde{b}(f)\right] \qquad (1)$$

2) Template Design: There are two major design questions when designing a template system: "What templates are we looking for?" and "How do we space these templates?" Both questions are fundamental. We chose to go with the Paczyński microlensing anomaly model. This model describes microlensing anomalies in terms of two parameters: u_0 and Einstein time (t_E). u_0 determines the peak, and t_E controls the length/timescale of the microlensing event. Since the number of templates can significantly affect the total detection time, astronomy related domain knowledge is used to limit the parameter searching space without missing the target microlensing events.

With regards to "How do we space these templates?", we evenly space templates along a range of values defined by a high and low threshold and the density (total number of spacings) desired. We wrote a Python script to generate all the necessary templates. Through initial experimentation, the total number of templates necessary is much less than those generated by a straightforward method that covers every single desired event in the parameter search space. This is accomplished in two ways. First, we reduced the dimensions of our problem. As mentioned, Paczyński microlensing anomalies consist of two parameters. However, similar to gravitational-wave research [13] [14] in our initial testing, one dimension (the length dimension) conveyed most of the appropriate information. This greatly reduces the total number of templates needed. Second, we intentionally mismatch the templates by only generating one template for a group of events instead of one for each event desired. For more treatment of template spacing, see [14] where an optimal template spacing solution is developed for use in gravitational-wave detection.

As a final note, one small, additional optimization is used in generating templates. As we are focusing on detecting microlensing events before their peak (online case), we can bias our templates for detecting early. We do this by only generating half of the Paczyński microlensing anomaly. This focuses detection on detecting an anomaly before its peak. Furthermore, it reduces the amount of data needed to store and process templates by two.

3) Input: Essential to every matched filtering method are inputs. Currently, our method handles two different types of inputs: online and offline. The online input is built to interact with the GWAC system. The offline input reads from various different formats. All of our inputs go through a multi-stage process. First, they are read. In the case of online inputs, this is done while running. In the case of offline inputs, this is done at the beginning of the program. Next, each star input is windowed to include only a subset of current data. We use this to minimize the amount of data we must process. This speeds up our program while allowing it to execute in GPU RAM (approximately 1,920 samples per star per night and at most 200,000 stars per-frame).

4) Normalization Methods: We normalize our templates in three ways. First, we ensure our templates start and end at approximately a magnitude of zero. We do this by subtracting the minimum value from the microlensing anomaly. This is to ensure that we do not impart an arbitrary DC term to different templates, which could impart bias. Second, we normalize them to unity under the matched filter operation used. This ensures each template is balanced so that no template is more powerful than another. Third, we normalize the templates based on their length. Due to the nature of our GPU optimization, all templates in our code are converted to the same length through the FFT. Increasing the length of a smaller template might change its energy. Therefore, to fit our optimization, we modify the template normalization process to use FFT lengths of the maximum length used in our program for a given set of templates (the maximum length template rounded to the next power of two).

For the input processing, the principal normalization we focus on is outlier removal. Outliers in the data could cause spurious outputs of the matched filter operation. Therefore, we implemented a simple outlier removal process to remove non-consecutive outliers that lie three standard deviations above the average. This corresponds to removing points that are not within the 0.27% of points, assuming our data follows a *Normal* distribution.

5) Detector Design: Our detector is designed around issues in the matched filter process. As seen in gravitational-wave research, matched filter output is complicated by the presence of "glitches", which can cause outputs similar to true anomaly detections [15] and [16]. There are many ways to handle this. We chose to focus on a simple and efficient method that should minimize the effect of very short transient glitches. Our method relies on the assumption that if an input contains a microlensing anomaly, then it will contain it for multiple windows of data in a row. Using this assumption, we only consider an anomaly to be a valid anomaly if the matched filter output (for any template, as determined by the maximum operation over all the templates' outputs) is above the detector threshold for three windows of data. This should help reduce the effect of very short transient anomalies.

C. GPU Accelerated Scalable Framework

1) CPU-GPU Collaborative Design: Our complete algorithm functionality is divided into two segments, a CPU segment and a GPU segment. In the main split of operations, outlier removal, loading data, and windowing are assigned to the CPU. The computation-intensive matched filtering operation is assigned to the GPU. Through initial testing, we saw a marked improvement through GPU-ization of the matched filter operation. Furthermore, the loading of the data and the processing of the data is orchestrated so that new data can be loaded while old data is being processed. This is accomplished through locking and copying the current data. Afterward, the current data is unlocked so that new data can be added.

2) Multi-source/sink Matched Filter Optimization: Another way of viewing the problem of processing multiple inputs in a matched filter is as a multi-source/sink matched filter. With this framework, our problem becomes, "How do we compute multiple sources to multiple templates in an efficient manner?" This is an analogue to the matrix multiplication operation. In this operation, we have multiple rows being applied (element-by-element multiplied and added) to many columns. If we assign the rows of one matrix to be stars and the columns of another matrix to be templates or vise-versa, we can compute multiple stars against multiple templates. As an additional added benefit, matrix multiplication, on the GPU, is a heavily optimized operation. Compared to a naive GPU threading design, we can have confidence of great performance using matrix multiplication on the GPU. The relation and setup of a matrix multiplication to compute our inner product definition can be found in Fig. 2. (Note, we must first use FFT on the individual inputs.)

We further implement one more matched filter optimization. We batch our stars and templates into groups before computing the matrix multiplication based matched filter. Thus, our matched filter matrix multiplication is computed multiple times for the combinations of the different groups. There are three benefits to this. First, if all of the data does not fit into the GPU's RAM, then we can group it differently to fit. Second, we can determine through testing the optimal grouping for maximum throughput for any given GPU. Third, in future work, we can extend our method to use multiple GPUs by assigning different group combinations to different GPUs.

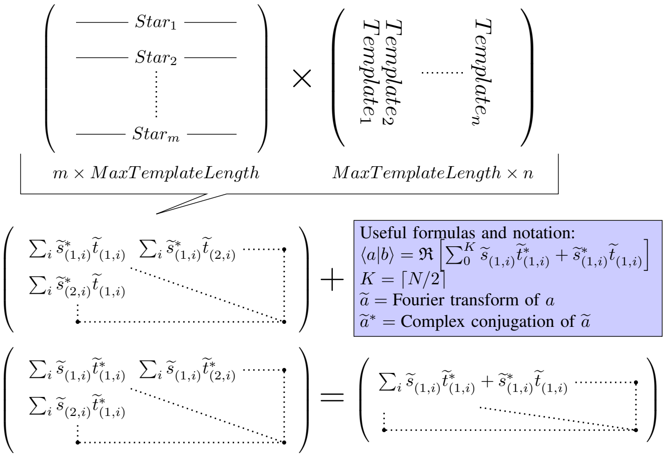
$Star_1$
$Star_2$
$Star_m$
×
$Template_1$
$Template_2$
$Template_n$
$m \times MaxTemplateLength$
$MaxTemplateLength \times n$
$\sum_i \widetilde{s}^*_{(1,i)}\widetilde{t}_{(1,i)}$
$\sum_i \widetilde{s}^*_{(1,i)}\widetilde{t}_{(2,i)}$
$\sum_i \widetilde{s}^*_{(2,i)}\widetilde{t}_{(1,i)}$
+
Useful formulas and notation:
$\langle a|b\rangle = \Re\left[\sum_0^K \widetilde{s}_{(1,i)}\widetilde{t}^*_{(1,i)} + \widetilde{s}^*_{(1,i)}\widetilde{t}_{(1,i)}\right]$
$K = \lceil N/2\rceil$
$\widetilde{a}$ = Fourier transform of a
$\widetilde{a}^*$ = Complex conjugation of $\widetilde{a}$
$\sum_i \widetilde{s}_{(1,i)}\widetilde{t}^*_{(1,i)}$
$\sum_i \widetilde{s}_{(1,i)}\widetilde{t}^*_{(2,i)}$
$\sum_i \widetilde{s}_{(2,i)}\widetilde{t}^*_{(1,i)}$
=
$\sum_i \widetilde{s}_{(1,i)}\widetilde{t}^*_{(1,i)} + \widetilde{s}^*_{(1,i)}\widetilde{t}_{(1,i)}$

Fig. 2. Matrix Multiplication based Inner Product

3) Work Sharding: Instead of only requiring high throughput, our problem is challenging in that it also requires strict latency bounds. As mentioned, we must finish processing each set of images every ten seconds. Combined with windowing, this opens up interesting avenues for speed improvement. Noticing that windowing implies that we process some data points more than once, we implemented a system of work sharding where we skip some data points while still processing the overall signal. This sharding relies on the microlensing anomaly being detectable for more than a very small number of points. Furthermore, we are still able to process the overall signal due to the high cadence of the GWAC telescope. The two optimizations that allow work sharding to work are skip-deltas and fragments. These both describe different but related functionalities.

First, skip-deltas answer the questions of "When a window of data is ready for processing?" Intuitively, a skip-delta "skips" over a certain number of points before allowing the algorithm to process a star's window again. For example, a skip-delta of 15 would cause a star that has recently been processed to wait 15 data points (15 seconds times 15 samples) before being processed again. In this time period, the window of the star is still updated. Thus, at the next processing time, our algorithm processes the freshest data for the star. Furthermore, skip-deltas are guaranteed to not increase false-positives. To prove this, consider that we use no skip-delta and set our threshold so that no false-positives are detected. The set of matched filter outputs produced by using skip-deltas is a subset of this data. Therefore, it is impossible that it would have additional false-positives. The same cannot be said about decreasing false-positives. It is possible that skip-deltas will skip over bad windows of data leading to a reduction in false-positives, if they exist. Finally, as to its effect on true-positives, if the anomaly is detectable over many points, it should not affect things significantly. Furthermore, any reduction should be concentrated in a delay to the detection of the anomaly.

Second, fragments answer the question of "What set of frames is each star computed on?" This is crucial. Skip-deltas by themselves only increase the throughput. However, if we can shard the stars into groups, then we can process different groups at different times, reducing per-frame processing time and helping with latency. For example, if we notice that it takes twenty seconds to process all of the stars, we can assign the number of fragments to 2 and use a skip-delta of 2 to shard the computation into two ten-second operations, processing half of the stars on one frame and half of the stars on the other frame. Our program uses fragments to establish frame sets to compute the different shards on.

IV. Experimental Results

A. Experiments Setup and Design

For testing, we implemented our algorithm in Rust using ArrayFire for the GPU operations. Unless mentioned or clear from context, each of our tests ran our algorithm with outlier removal enabled. Furthermore, our window size was set to be 60 samples. This corresponds to 15 minutes of data. For detection, we used our detect-after-three scheme. For templates, we used 600 half-width templates with a u_0 of 1 and t_E ranging between 1,800 and 87,616 seconds. Optimal thresholds were calculated using a binary search across the threshold space down to a precision of 10^{-3}.

Regarding our performance tests, all of our results were timed from the start of the program (including data loading) using the POSIX shell time utility. Furthermore, the performance dataset was loaded locally using the offline data path discussed earlier. To reduce loading time for the massive number of stars, each star was stored in a space-efficient format inside of an SQLite3 database. Our templates were kept in GPU memory throughout the entire run. The default batching of stars and templates (grouping each into groups of 1,024) was used. Finally, our program was compiled with the release flag and ran.

All performance tests were run on a commodity hardware machine running Ubuntu 18.04.4 (Kernel 5.6.15-050615-generic). The CPU used was an AMD Ryzen 5 3600 consisting of 6 cores and 12 threads running at 3.6 GHz with a max boost of 4.2 GHz. The storage used was an Intel 660p Series M.2 512 GB PCI-e x4 NAND SSD. The RAM used was a Corsair Vengeance LPX 32GB DDR4 3000 SDRAM stick. Finally, the GPU used was the Gigabyte Radeon RX 5700 XT OC 8G graphics card with 8GB of GDDR6 RAM running with the AMDGPU PRO 20.10 OpenCL driver.

GPU time is calculated by subtracting the real time of a run from the system time and user time. The system time and user represent the total processor time [17]. Thus, the remaining time is time spent waiting. We assume that this time is time spent waiting on the GPU as our RAM is fast, and we are not reading or writing data to the SSD after loading the templates and stars.

We tested our accuracy against the GWAC Variable Star dataset from [10] generated using their test dataset generator. (All NFD results used are from [10]. For details on the parameters under which NFD was tested on the GWAC Variable Star dataset, see [10].) We tested our performance against a custom dataset consisting of 200,000 stars each of 1,922 samples (8.01 hours).

The GWAC Variable Star dataset used consists of 3,240 stars. Each star is made up of one Paczyński microlensing event at the end of the signal and a single sine-wave background source with noise. Furthermore, the background sine-wave in each star has a DC term of 0. Regarding the length of the data, each star signal consists of 24 (approximately eight-hour) days. Each day corresponds to a typical observation period for the GWAC telescope, which is eight hours. Furthermore, to match real world conditions, the phase of the background sine-wave is changed every eight-hour period to simulate waiting a new day before another night of observation. For further details on this dataset, see [10].

To test the effect of work sharding on accuracy and performance, we used two different strategies. For the accuracy dataset, a combination of skip-delta and fragments would have been inefficient to use. This is due to our accuracy dataset having very few stars (small amounts of data are parallelized poorly on the GPU). Therefore, we only used skip-deltas to approximate sharding. This processes the same amount of data as using fragments and skip-deltas together. It only processes different windows which should not, in general, change the accuracy profile. For the performance dataset, we set fragments and skip-deltas to be equal to the same number. This creates the proper work sharding setup we described earlier.

The source code for PLAD's implementation is available on GitLab[1].

B. Accuracy Results

Accuracy is measured in regards to two quantities: SPR and ADP. Sample precision rate (SPR) is a measure of the predictive power. It is the rate of true-positives given there are no false-positives. Average detecting position (ADP) is a measure of detecting position. It computes the position of the detection within an anomaly. Since Paczyński microlensing anomalies are symmetric, ADP is a percentage value from -50% to 50%, with 0% representing the peak of the anomaly.

As seen in Table I, the proposed PLAD method surpasses the predictive performance of NFD both when and when not using work sharding. Furthermore, with a near 20% increase in detection performance (SPR), our good results are not the result of Monte-Carlo differences in small parts of the data. They are an improvement in the detection of the data. Furthermore, this is further confirmed by the analysis of the ability to detect anomalies early (before peak). PLAD (for a skip-delta of 1) achieved an approximately 17% increase in early detection time compared to NFD. Therefore, PLAD not only can detect more anomalies, but it can also allow for the gathering of more data on on-going anomalies through early detection, on average -14% before the peak of the anomaly.

Regarding the effect of work sharding on predictive performance, we see minimal effects on accuracy (SPR). By increasing the number of shards, some windows of data are skipped. These windows might consist of more or less false/true-positives. Though fixed and deterministic windows are chosen at each time, it is practically impossible to determine which set of windows contains more or less false/true-positive windows. However, due to the high cadence of the telescope, these skipped windows should not affect the final result significantly.

Concerning the effect of work sharding on ADP, our ADP only decreases slightly. Under the assumption that a microlensing anomaly is detectable for a number of windows in a row (number of detectable windows greater than or equal to the number of shards) near the peak, this slight decrease is in line with our observation that the worst-case scenario in this situation is that the current window being analyzed is the final window before true-positive windows appear. Thus, the proposed PLAD algorithm would need to wait the number of shards before detection.

TABLE I
ACCURACY AND EARLY ALARM RESULTS

Method	SPR	ADP
NFD	73.2%	3.0%
PLAD (Skip-delta = 1)	92.28%	-14.24%
PLAD (Skip-delta = 15)	92.16%	-11.75%

As we can see in Table II, PLAD is generally resistant to false-positives. There exists a range of approximately 20

[1] Source code located at gitlab.com/acminor/plad-rust

threshold units where our algorithm only looses 479 stars (drops accuracy to 77.4%). This is still greater than the NFD's performance on our dataset. Furthermore, in this range, loss in ADP does not result in after peak prediction (on average). Therefore, we still can detect before the peak of the anomaly, and PLAD still beats NFD's performance on our dataset. The most important thing about these findings is that we can sacrifice very little true-positives to improve resistance to false-positives.This is key in balancing the proper usage of shared follow-up telescopes.

TABLE II
SELECTING THRESHOLD VALUE TO EFFICIENTLY REDUCE FALSE-POSITIVES WITH LOW TRUE-POSITIVE AND EARLY ALARM LOSS, SKIP-DELTA = 15

Threshold	False-positives	True-positives	ADP
31.250	3,240	0	N/a
39.063	871	2,229	-16.510%
41.016	2	2,994	-12.570%
41.504	1	2,988	-12.023%
41.748	1	2,987	-11.741%
41.763	1	2,986	-11.737%
41.767	1	2,985	-11.749%
41.768	0	2,986	-11.748%
41.769	0	2,986	-11.748%
41.771	0	2,986	-11.742%
41.779	0	2,986	-11.729%
41.809	0	2,986	-11.685%
41.870	0	2,984	-11.632%
41.992	0	2,983	-11.454%
42.969	0	2,952	-10.633%
46.875	0	2,868	-8.208%
62.500	0	2,507	-4.465%
125.000	0	1,441	-1.471%
250.000	0	594	-0.028%
500.000	0	386	1.027%
1000.000	0	0	N/a

C. *Performance Results*

As seen in Table III, the proposed PLAD's performance is consistent. With the consistency property satisfied, we can discuss our algorithm's performance generally. First, we notice that most of our time is GPU computation time. This fits with our program architecture and run environment. Only a small fraction of the program reads from disk and only at the beginning. Furthermore, our standard output is piped to /dev/null. Therefore, most of our time is spent calculating on the GPU. If we desire further efficiency gains, then we must focus on improving GPU speed, efficiency, or count. Second, sharding our work into 15 shards gains a roughly 13.5x performance improvement demonstrating the power of sharding in regard to scalability.

As seen in Table IV, PLAD operates within the performance constraints of the GWAC system, processing 200,000 stars in 3.34 seconds. Furthermore, using our method's work sharding functionality, we can process 200,000 stars in less than second. When the GWAC system upgrades to processing an image every second, PLAD can continue to work with no changes to the hardware running it. Work sharding demonstrates our algorithms future adaptability and scalability, which is crucial for a project with a ten-year run.

TABLE III
EXECUTION TIME FOR 200,000 STAR DATASET (TIME IN SECONDS), SHARDS=SKIP-DELTA=FRAGMENTS

					CPU	GPU
Run	Shards	Real	User	System	Time	Time
1	1	6,231.922	585.412	146.425	731.837	5,500.085
2	1	6,218.907	571.965	148.946	720.911	5,497.996
3	1	6,211.460	568.147	153.670	721.817	5,489.643
Average	1	6,220.763	575.175	149.680	724.855	5,495.908
1	15	461.038	95.967	11.807	107.774	353.264
2	15	459.095	94.275	11.259	105.534	353.561
3	15	460.220	94.894	12.000	106.894	353.326
Average	15	460.118	95.045	11.689	106.734	353.384

TABLE IV
AVERAGE EXECUTION TIME TO PROCESS 200,000 STARS FOR 200,000 STAR DATASET, SHARDS=SKIP-DELTA=FRAGMENTS

Method	Time (seconds)
PLAD (1 Shard)	3.34
PLAD (15 Shards)	0.25

V. CONCLUSION

Low latency, high throughput, and scalable parallel algorithms are crucial for accelerating scientific research based on big data. In this paper, we proposed an accurate matched filtering based microlensing anomaly detection algorithm with a very low false-positive rate. Furthermore, our algorithm can raise an alarm approximately 14% before the peak of a microlensing event on average. This is important because more information about the event can be captured by follow-up observation. Finally, with minimal effect on our detection goals, we can increase our resistance to potential future false-positives with very little loss in true-positives. This is key to properly share and efficiently use follow-up telescopes.

Since microlensing is infrequent and short timescale microlensing events are even scarcer, a large number of stars must be monitored constantly. This is a great challenge for high performance data processing. We develop a systematic framework that employs the GPU to accelerate the total performance. Our framework can achieve both high throughput and low latency using commodity hardware. Furthermore, through the use of work sharding, we can process 200,000 stars in under a second with minimal effect on detecting position and accuracy. Thus, we can scale our method along with future upgrades to the GWAC telescope system.

There are two next steps. The first step is to use our 200,000 Star dataset to determine optimal group sizes (for modern GPUs) for batching stars and templates. Furthermore, this should be done under two different constraints: paging the templates in and out of GPU memory and keeping them in GPU memory. The second step is to extend our framework to run on multiple GPUs to further improve the performance for next-generation sky survey telescopes, allowing for performance gains while skipping less or no data.

REFERENCES

[1] B. Paczynski, “Gravitational microlensing by the galactic halo,” *The Astrophysical Journal*, vol. 304, pp. 1–5, 5 1986.

[2] A. Gould and A. Loeb, "Discovering planetary systems through gravitational microlenses," *The Astrophysical Journal*, vol. 396, pp. 104–114, 9 1992.

[3] J. Wei, B. Cordier, S. Antier, P. Antilogus, J.-L. Atteia, A. Bajat, S. Basa, V. Beckmann, M. Bernardini, S. Boissier *et al.*, "The deep and transient universe in the svom era: new challenges and opportunities-scientific prospects of the svom mission," *arXiv preprint arXiv:1610.06892*, 2016.

[4] B. Paczynski, "Gravitational microlensing by the galactic halo," *The Astrophysical Journal*, vol. 304, pp. 1–5, 1986.

[5] B. P. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari *et al.*, "Observation of gravitational waves from a binary black hole merger," *Physical review letters*, vol. 116, no. 6, p. 061102, 2016.

[6] A. Udalski, M. Szymanski, J. Kaluzny, M. Kubiak, M. Mateo, W. Krzeminski, and B. Paczynski, "The optical gravitational lensing experiment. the early warning system," *arXiv preprint astro-ph/9408026*, 1994.

[7] S. Ahmad and S. Purdy, "Real-time anomaly detection for streaming analytics," *arXiv preprint arXiv:1607.02480*, 2016.

[8] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015.

[9] H.-W. Kim, K.-H. Hwang, Y. Shvartzvald, J. C. Yee, M. D. Albrow, S.-M. Cha, S.-J. Chung, A. Gould, C. Han, Y. K. Jung *et al.*, "The Korea Microlensing Telescope Network (KMTNet) Alert Algorithm and Alert System," *arXiv preprint arXiv:1806.07545*, 2018.

[10] J. Qiu, Y. Sun, C. Wu, Z. Du, and J. Wei, "NFD: Toward real-time mining of short-timescale gravitational microlensing events," *Publications of the Astronomical Society of the Pacific*, vol. 130, no. 992, p. 104504, 2018.

[11] G. Turin, "An introduction to matched filters," *IRE Transactions on Information Theory*, vol. 6, no. 3, pp. 311–329, June 1960.

[12] L. A. Wainstein and V. D. Zubakov, *Extraction of Signals from Noise*. Englewood Cliffs, N.J.: Prentice-Hall, 1962.

[13] B. J. Owen, "Search templates for gravitational waves from inspiraling binaries: Choice of template spacing," *Physical Review D*, vol. 53, pp. 6749–6761, Jun 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.53.6749

[14] B. J. Owen and B. S. Sathyaprakash, "Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement," *Physical Review D*, vol. 60, no. 2, p. 022002, 1999.

[15] C. Messick, K. Blackburn, P. Brady, P. Brockill, K. Cannon, R. Cariou, S. Caudill, S. J. Chamberlin, J. D. Creighton, R. Everett, and e. al., "Analysis framework for the prompt discovery of compact binary mergers in gravitational-wave data," *Physical Review D*, vol. 95, no. 4, Feb 2017. [Online]. Available: http://dx.doi.org/10.1103/PhysRevD.95.042001

[16] Q. Chu, "Low-latency detection and localization of gravitational waves from compact binary coalescences," phdthesis, The University of Western Australia, 2017.

[17] T. Cokelaer, "Meaning of Real, User and Sys time statistics," 02 2018. [Online]. Available: https://thomas-cokelaer.info/blog/2018/02/meaning-of-real-user-and-sys-time-statistics/