Responses to Reviewers

We update our code link: https://anonymous.4open.science/r/Pontus-3C76/.

Reviewer #1

Many thanks for comments!

O1: In the introduction, or the latest in the discussion of related work, I would appreciate a clearer statement on why burst detection algorithms do not extend to the more general wave pattern.

Author Response O1: Compared with burst, wave is a more comprehensive definition of the fluctuation of data streams. Therefore, tracking waves is more complicated than bursts, requiring more memory and computing power. There are 4 reasons why the previous algorithms cannot extend to wave detection: 1) Previous algorithms structure are not scalable. For example, the design of BurstSketch only support comparison of the frequencies of the item in two adjacent windows to determine whether the item is increasing, while tracking increasing/decreasing the trends of frequencies over multiple windows is a necessity for wave detection. Furthermore, it does not consider the case of "negative" fluctuations. In our experiments, we extended two classic algorithms in the field of data mining, SS and TLF, as baselines. The experimental results show that the performance of these two algorithms is far inferior to that of Pontus, and they are not suitable for small memory scenarios. 2） Existing algorithms face throughput bottlenecks after extension and are unable to support the high-speed processing. For example, BurstSketch traverses all the buckets in the second stage for each insertion,resulting in poor throughput. 3) The design of existing algorithms have some drawbacks, affecting their performance. For example, Burstsketch uses a fixed threshold to remove items with low frequency, and records items with high frequency in the second stage. False negative may occur when an item grows from a low frequency below the fixed threhold to a high frequency, which appears many times in the data stream, seriously affecting the performance of BurstSketch. 4) TopicSketch and BurstSketch couple its insertion and update together, and cannot be deployed in hardware devices (e.g., programmable switches, smart network cards), making them incapable in high-speed data stream scenarios.

Author Actions O1:
Addition in Section 2, fourth paragraph, Page 2:
"Compared with the burst, the wave is a more comprehensive definition of the fluctuation of data streams. Therefore, tracking waves is more complicated than bursts, requiring more memory and computing power. There are four reasons why the previous algorithms cannot extend to wave detection: 1) Previous algorithms' structures are not scalable. For example, the design of BurstSketch only supports comparison of an item's frequencies in two adjacent windows to determine whether the item is increasing. However, tracking the increasing/decreasing trends of frequencies over multiple windows is a necessity for wave detection. 2) Existing algorithms face throughput bottlenecks after extension and are unable to support high-speed processing. For example, BurstSketch traverses all buckets in the second stage for each insertion, resulting in poor throughput. 3) The design of existing algorithms have some drawbacks, affecting their performance. BurstSketch uses a fixed threshold to remove items with low frequency and records items with high frequency in the second stage, which may cause false negatives. 4) TopicSketch and BurstSketch couple their

insertion and update together, and cannot be deployed on hardware devices (e.g., programmable switches, smart network cards), hindering their potential in high-speed data stream scenarios."

O2: In Definition 1, I assume there are some conditions on the value of k (positive, > 1?). Such conditions should be stated explicitely.

Author Response&Actions O2: We have checked the manuscript and fixed the mentioned and other errors to improve the quality of the paper.

Author Actions O2:
Modification in Section 3.1, Definition 1&2, Page 3:
"...Shape condition: f_j \geq k \cdot f_i, f_m \leq \frac{1}{k} \cdot f_n , ***k \in \{Z_+ | k > 1\}****..."*
"...Shape condition: f_j \leq \frac{1}{k} \cdot f_i, f_m \geq k \cdot f_n, ***k \in \{Z_+ | k > 1\}}****..."*

O3: In the description of the Pontus framework, it remains somewhat unclear to me why there are exactly three stages. Can you explain this intuitively before the detailed discussion? Overall, I would appreciate some more discussion on "why" the framework has been designed in this way as opposed to focusing the discussion mostly on "how" components are implemented. Essentially, I am wondering why what you propose is the best thing to do.

Author Response O3: Wave characterizes an abnormal data steam pattern, which usually takes up a very small portion of data stream items.In fact, most items (that is, normal items) in the data stream are in a rather stable or slightly fluctuating state. This motivate our idea of using a three stages"filtering" strategy, where we filter out useless items, i.e., normal/nonsuspicious items, as early as possible, and only maintain and keep track of useful items (suspicious items), so as to save memory overhead and improve throughput. We classify the items in the data stream into three types: 1) useless items, which violate wave conditions already). 2) weak potential waves, items that have started to show increase/decrease trends but haven't satisfied the left shape condition yet. 3) positive/negative waves that have already satisfied the left shape condition. At each time window, we keep most of the incoming items (except those that exist in S2 and S3) in S1. At the end of the each time window, S1 filters out useless items, picks out weak potential waves and sends them to S2. S2 filters out illegal weak potential waves and track the remaining weak potential waves. If weak potential waves meet the definition of potential waves, S2 sends them to S3. S3 records potential waves and filters out illegal potential waves. When a potential wave meets the wave definition, we report it as a wave, otherwise we track it in S3. In our implementation, we set the ratio of S1: S2: S3 to 7:2:1 because # useless item >> # weak potential wave > # potential wave, which can further reduce hash collisions. The experimental results perform well, which verifies the effectiveness of our setup. This ratio can also be fine-tuned in different applications.

Author Actions O3:
Addition in Section 4.2, second paragraph, Page 3&4:
"...The Multi-Stage Progressive Tracking is from the core idea of "filtering", where we filter out useless items, i.e., non-fluctuated items, as early as possible, and only maintain and track useful items (fluctuating items), to save memory and improve throughput. We classify the items in the data stream into three types: 1) useless items, which violate wave conditions already. 2) weak potential waves, items that have started to show increase or decrease trends but haven't satisfied the left shape condition (i.e., f_j \geq k \cdot f_i or f_j \leq \frac{1}{k} \cdot f_i) yet. The \emph{weak

potential wave} is identified by the thresholds \lambda_{in} for a positive wave, i.e., f_j \geq \lambda_{in} \cdot f_i, and \lambda_{de} for a negative wave, i.e., f_j \leq \lambda_{de} \cdot f_i, where 1\le\lambda_{in}<k and 1/k<\lambda_{de}\le1 (e.g., \lambda_{in}=1.3, \lambda_{de} = 0.8), meaning the change of frequency is not strong enough to meet the left shape condition yet, but can be regarded as a possible start of the left shape of a wave. 3) potential waves that have already satisfied the left shape condition...."
"..Particularly, the \emph{Stage 1} (initial stage) is responsible for recording the frequencies of current and \textcolor{red}{previous} windows for the \textcolor{red}{useless} items. The \emph{Stage 2} (weak potential stage) and the \emph{Stage 3} (potential stage) are responsible for recording the frequencies of the weak potential waves and potential waves, respectively..."
"...The \emph{Stages 1} filters out useless items and sends weak potential waves to the \emph{Stages 2} and potential waves to the \emph{Stages 3}. The \emph{Stages 2} filters out illegal weak potential waves among the recorded weak potential waves and sends potential waves to the \emph{Stages 3}. The \emph{Stages 3} filters out illegal potential waves and reports waves...."

O4: In the evaluation, I am wondering how the ground truth has been labeled in the datasets in order to measure, e.g., recall or precision. Please clarify.

Author Response O4: To generate the ground truth label, we assign a bucket to each incoming item, which contains T_{de}+T_l+T_{in} counters to record the frequency of items across windows. At the end of each time window, we iterate through all buckets and check if the frequency change of the counters in the buckets satisfies the definition of wave. If so, we report it as a wave. During experiment, a wave detected by the algorithm is considered to be correct when its type (i.e. pos./neg.), t (timestamp), w_{in}, w_{de}, w_l equals those of the a ground truth wave, which are used to calculate PR and RR. ARE calculations use the formulas in Section 9.1.

Author Actions O4:
Addition in Section 9.1, last paragraph, Page 10&11:
"Ground Truth: To generate ground the truth label, we assign a bucket to each incoming item, which contains T_{de}+T_l+T_{in} counters to record the frequency of items across windows. At the end of each time window, we iterate through all buckets and check if the frequency change of the counters in the buckets satisfies the definition of wave. If so, we report it as a wave. During experiment, a wave detected by the algorithm is considered to be correct when its type (i.e. pos./neg.), \$t\$(timestamp), w_{in}, w_{de}, w_l equals those of the a ground truth wave, which are used to calculate PR and RR. We also assign each bucket four counters to record curve points which are used to calculate ARE."

O5: Furthermore, some statistics on how many waves / bursts each dataset presents, possibly with characteristics of the waves (e.g., most are positive waves, #waves matching a particular application, ...) would be interesting. Maybe you can provide a table with details about the datasets?

Author Response &Actions O5: We will include such statistics in a table.

Author Actions O5:
Addition in Section 9.5, first paragraph & Table 3, Page 11:
"To illustrate the statistics of wave on different applications, we further count the numbers of wave on different applications in previous experiments. Note that for user privacy protection purpose, public datasets do not contain the payload of packets. Therefore we classify applications based on

port numbers. As shown in Table 3, we mainly observe five port numbers and others. The wave of Port 80/443 is reserved by the HTTP protocol for web browsing, which can be adopted in online advertising related applications. The wave of port 53 is for DNS nameservers, which can be used for DNS query related scenarios. The wave of port 23,22,123 may be caused by attacks in the network. For example, Mirai \cite{AntonakakisABBB17} attacks at Port 23, Worm \cite{esec/Spafford89} attacks at Port 123. When deploying Pontus in the real world, a detailed analysis can be done through the payload of a packet."

port / Dataset	80/443	23	53	22	123	others
Data Center	4013	3441	2409	688	139	783
CAIDA	7964	6793	7028	937	539	163
WIDE	4094	3639	363	637	127	238
Synthetic Dataset	4283	3789	4200	2471	1317	414

Table 3: Statistics of Waves on Different Applications.

M1: The abbreviation of DDoS is not introduced. I assume it is distributed denial of service, but readers may not be familiar with it.

Author Response&Actions M1: We have checked the manuscript and fixed the mentioned and other errors to improve the quality of the paper.

Author Actions M1:
Modification in Section 1, fourth&fifth paragraph, Page 1:
"...Distributed Denial of Service (DDoS)..."
"...DNS (Domain Name System)..."

Reviewer #2

Many thanks for comments!

O1: The paper primarily addresses a networking research audience, rather than a data management one. This is problematic because, while there has been significant interest in the use of sketching in SIGMOD/VLDB, it is important to ground the results in the motivating applications. This community is less familiar with recent networking concepts (e.g., P4) and so familiarity with these should not be assumed. It would also help to consider non-networking applications to avoid the perception that this is a paper that has been rejected from networking venues and resubmitted to SIGMOD without significant revision.

Author Response O1: Wave detection has many important application scenarios outside the network community. In general data mining community, it can be used in online advertising (Scenario 2), trading volumes monitoring (Scenario 3), bursty topic mining (Scenario 3), text stream mining (Scenario 3), and financial market (Scenario 5). So far, the processing speed of the CPU is far from keeping up with the speed of the data stream. How to handle the high-speed data stream is still a key challenge for the data mining community. The network community offers a potential solution to this problem by exploiting the capacity of hardware devices (e.g., programmable switches and smart network interface cards). We believe that the introduction of hardware devices into the data mining community can motivate researchers for data stream processing solutions with high speeds.

Author Actions O1:
Modification in Section 1, fourth&fifth&sixth&seventh&eighth paragraph , Page 1:
“Scenario 1 - Botnet Detection (Network Community)...”
“Scenario 2 - DNS Queries (Network and Data Mining Community)...”
“Scenario 3 - Burst Detection (Data Mining Community)...”
“Scenario 4 - Network Failure Detection (Network Community)...”
“Scenario 5 - Financial Market (Data Mining Community)...”
Addition in Section 2, last paragraph, Page 2:
“Programmable Switches. Programmable switches (e.g., Barefoot Tofino~\cite{Tofino}) are an emerging networking technology that provides hardware programmability and flexibility without compromising performance.A representative programmable switch architecture is Protocol Independent Switch Architecture (PISA)~\cite{pisa}, where the ASIC chip consists of a programmable parser and a number of reconfigurable match-action tables. Operators can implement custom programs in the switch using domain-specific languages (e.g., P4~\cite{p4_language}), allowing the switch to process data traffic at terabits per second. With high line-rate guarantee and flexibility, programmable switches are ideal for wave detection and estimation.”
Modification in Section 7, Page 7&8:
“We implement three stages of Pontus in programmable switches to improve the performance of Pontus. With the Stateful Algorithm and Logical Unit (Stateful ALU) in each stage of the switch pipeline, we can lookup and update the entries in the corresponding register array.
There are three differences between the P4 hardware deployment version and the software version of Pontus. 1) Due to the resource limitation of Stateful ALU, we only store the key and value field in physical registers. For insertion, we need to go back to the register that has the smallest value and reset the key and value when there are hash collisions in S_1. However, this process is not allowed in the P4 language. Hence, we use the resubmit primitive to solve this problem. When a packet is

resubmitted to the beginning of the pipeline, it can maintain up to eight bytes of metadata in the resubmit header. We use the resubmit metadata to record the necessary information (e.g., the counted packet number) for the replace operation. 2) As multiplications, divisions, and floating-point operations are not supported in P4, it is difficult to calculate the probability. In P4, to approximate probabilistic replacement, we generate a 32-bit random number r and replace the smallest bucket in S_1 if (r << L_V) < 2^{32}, in which L_V represents the bit furthest to the left in V. 3) The control plane is deployed on the local CPU of the programmable switch or a remote server. At the end of each window, the programmable data plane sends values in registers of different stages to the control plane. After updating, the control plane sends the updated information back to the data plane through rewriting physical registers in the switch pipeline."

O2: There is a rich and deep algorithmic theory underpinning data stream algorithms, so that for some problems such as heavy hitters matching upper and lower bounds are known. It is an omission that there are no comparable bounds stated for the wave problem: what accuracy can be guaranteed as a function of the space used to track the wave?

Author Response O2: We prove upper bounds of false negative and positive in Section 8.2.

Author Actions O1:
Modification in Section 4.2, second paragraph, Page 3&4:
"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde} further verify that these error bounds are guaranteed to be small with properly selected parameters."
Modification in Section 8.2, Page 8&9:
Please refer to the paper.

O3: The paper is lacking a clear demonstration of the *correctness* of the proposed approach. Definition 1 formally states what it means to be a wave. I would expect that the description of the proposed approach should refer carefully back to this definition to demonstrate (say) that all true waves would be reported, and no false ("illegal" in the terminology of the paper) waves are reported. There are some hints of this in Section 4.2, but not a precise argument, assuming, say, an oracle for the frequency of any item.

Author Response O3: The space complexity of Pontus is O(1) << O(N), which means hash collisions are inevitable in Pontus. Thus Pontus cannot achieve absolute correctness. We will discuss the "correctness" briefly in Section 4.2 and prove upper bounds of false negative and positive in Section 8.2.

Author Actions O3:
Modification in Section 4.2, second paragraph, Page 3&4:

"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde} further verify that these error bounds are guaranteed to be small with properly selected parameters."

Modification in Section 8.2, Page 8&9:

Please refer to the paper.

O4: The theoretical properties that are presented in Section 8 are somewhat simplistic (closely following prior analyses of similar sketching techniques), and unclear whether they fully characterize the important properties of the algorithm.

Author Response O4: We will modify Section 8 and prove upper bounds of false negative and positive in Section 8.2.

Author Actions O4:

Modification in Section 4.2, second paragraph, Page 3&4:

"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde} further verify that these error bounds are guaranteed to be small with properly selected parameters."

Modification in Section 8.2, Page 8&9:

Please refer to the paper.

D1: Defn 1, please clarify that windows are non-overlapping windows that partition the time domain.

Author Actions D1: We will fix it.

Modification in Section 3.1, Definition 1&2, Page 3:

*"...For Item e, a positive wave, as Figure 1(a) shows, is identified if there exist four **non-overlapping** windows W_i,W_j,W_n,W_m,..."*

*"...For Item e, a negative wave, as Figure 1(b) shows, is identified if there exist four **non-overlapping** windows W_i,W_j,W_n,W_m,..."*

D2: Please give more discussion of the drawbacks of the strawman solution. If sufficient space is available, what are the other limitations of keeping multiple sketches -- say more about how filtering items could lead to improved accuracy.

Author Response D2: The strawman solution has three main drawbacks. 1) When the model parameters are changed (e.g. T_{in}, T_l, T_{de}), the number of sketches in the strawman needs to be changed, necessitating the redeployment of the model. As such, the strawman algorithm has poor flexibility. 2) The throughput of strawman is very limited, since each insertion of strawman needs to traverse the entire hash table. 3) Strawman is memory-intensive because it requires T_{in}+T_l+T_{de} sketches to store the frequencies.4) Strawman stores a large amount of useless item information in each time window, which cannot efficiently use space, resulting in excessive hash collisions and poor performance. Wave characterizes an abnormal data steam pattern, which usually takes up a very small portion of data stream items. In fact, most items (that is, normal items) in the data stream are in a rather stable or slightly fluctuating state. This motivates our idea of using a three stages "filtering" strategy, where we filter out useless items, i.e., normal/nonsuspicious items, as early as possible, and only maintain and keep track of useful items (suspicious items), so as to save memory overhead and improve throughput.

Author Actions D2:
Addition in Section 4.1, third paragraph, Page 3:
"The strawman solution has four main drawbacks. 1) When the model parameters (e.g., T_{in}, T_l, T_{de}) change, the number of sketches in the strawman needs to be changed, necessitating the redeployment of the model. As such, the strawman algorithm has poor flexibility. 2) The throughput of strawman is very limited, since each insertion of strawman needs to traverse the entire hash table. 3) Strawman is memory-intensive because it requires T_{in}+T_l+T_{de} sketches to store the frequencies. 4) Strawman stores a large amount of useless item information in each time window, which cannot efficiently utilize the memory, resulting in excessive hash collisions and poor performance. 5) Strawman couples its insertion and update together, and cannot be deployed on hardware devices."

D3: "current and last windows" do you mean current and previous?

Author Response&Actions D3: We have rechecked the manuscript and fixed the mentioned and other errors to improve the quality of the paper.

Author Actions D3:
Modification in Section 4.2, first&second paragraph, Page 3&4:
*"...Therefore, in Pontus, we first only record the item's frequencies of the current and **previous** windows...."*
*" ...\emph{Stage 1} (initial stage) is responsible for recording the frequencies of current and **previous** windows for the useless items...."*

D4:"the left shape of a wave may span multiple windows". Please say more about how the proposed approach will ensure to catch all waves and not introduce false positives while only keeping two sketches. It was unclear to me how this could be achieved without more detail.

Author Response D4: The left shape condition means f_j \geq k \cdot f_i / f_j \leq \frac{1}{k} \cdot f_i for positive wave and negative wave, respectively. The space complexity of Pontus is O(1) << O(N), which means that hash collisions are inevitable in Pontus. Therefore, Pontus may see false positives and false negatives. We use four curve points f_i,f_j,f_m,f_n to approximate the shape of the entire wave.In the control plane, for each item in S1, there are V_c (implying the frequency of the previous window item) and V_d (implying the frequency of the current window item). We compare V_c and V_d to determine whether the frequency of the item increases or decreases. If V_c and V_d satisfy the left shape condition (Situation1 or Situation2), i.e., the frequency of the item increases by k times or decreases by 1/k times within a time window, we record V_c and V_d as curve points f_i, f_j, and then send the item to S3. If not, we will continue monitoring it in S2 until it satisfies or becomes illegal. Details are in Section 6.

Author Actions D4:
Modification in Section 4.2, second paragraph, Page 3&4:
"...2) weak potential waves, items that have started to show increase/decrease trends but haven't satisfied the left shape condition ***(i.e., f_j \geq k \cdot f_i / f_j \leq \frac{1}{k} \cdot f_i)*** *yet...."*
"...Note that although Multi-Stage Progressive Tracking strategy can save memory overhead, the space complexity of Pontus is O(1) << O(N), which means hash collisions are inevitable in Pontus. Thus Pontus cannot achieve absolute correctness. We proof the upper bounds on the probability of false positives and false negatives for Pontus. Experiments further verify that they are guaranteed to be small under appropriate parameters."

D5: "Stage variance maximization" is referred to many times during the description of the algorithm, but not described until the very end. Even then, I found the description unclear to me. Please consider how to present the technique more clearly, without these long range forward pointers.

Author Response D5: The purpose of "Stage Variance Maximization" is to save the waves with large fluctuations, because the waves with large fluctuations are strong indicators for anomalies of the items in the data streams, which is inspired by Top-k and Conservative Update technique in sketching.

Author Actions D5:
Modification in Section 4.2, sixth paragraph, Page 4:
"...To solve hash collisions in the control plane, we use the Stage Variance Maximization technique to maintain waves with larger variances and reduce the estimation error. The purpose of Stage Variance Maximization is that the waves with large fluctuations are strong indicators for anomalies of the items in the data streams...."

D6: The motivation for the three stages was not very clear, even given the overview in Section 5. Is the intent to achieve greater accuracy by storing progressively fewer items in each stage? Some guidance on how to set the size of each hash table, and whether the aim is to collect exact or approximate counts would be helpful.

Author Response D6: Wave characterizes an abnormal data steam pattern, which usually takes up a very small portion of data stream items.In fact, most items (that is, normal items) in the data stream are in a rather stable or slightly fluctuating state. This motivate our idea of using a three stages"filtering" strategy, where we filter out useless items, i.e., normal/nonsuspicious items, as early as possible, and only maintain and keep track of useful items (suspicious items), so as to save memory overhead and improve throughput. We classify the items in the data stream into three types: 1) useless items, which violate wave conditions already). 2) weak potential waves, items that have started to show increase/decrease trends but haven't satisfied the left shape condition yet. 3) positive/negative waves that have already satisfied the left shape condition. At each time window, we keep most of the incoming items (except those that exist in S2 and S3) in S1. At the end of the each time window, S1 filters out useless items, picks out weak potential waves and sends them to S2. S2 filters out illegal weak potential waves and track the remaining weak potential waves. If weak potential waves meet the definition of potential waves, S2 sends them to S3. S3 records potential waves and filters out illegal potential waves. When a potential wave meets the wave definition, we report it as a wave, otherwise we track it in S3. In our implementation, we set the ratio of S1: S2: S3 to 7:2:1 because # useless item >> # weak potential wave > # potential wave, which can further reduce hash collisions. The experimental results perform well, which verifies the effectiveness of our setup. This ratio can also be fine-tuned in different applications. The approximate count is used in S1. For S2 and S3, we use "Stage variance maximization" to replace the item with the smallest variance, without changing the value. So the exact count is used in S2 and S3.

Author Actions D6:
Addition in Section 4.2, second paragraph, Page 3&4:
"...The Multi-Stage Progressive Tracking is from the core idea of "filtering", where we filter out useless items, i.e., non-fluctuated items, as early as possible, and only maintain and track useful items (fluctuating items), to save memory and improve throughput. We classify the items in the data stream into three types: 1) useless items, which violate wave conditions already. 2) weak potential waves, items that have started to show increase or decrease trends but haven't satisfied the left shape condition (i.e., f_j \geq k \cdot f_i or f_j \leq \frac{1}{k} \cdot f_i) yet. The \emph{weak potential wave} is identified by the thresholds \lambda_{in} for a positive wave, i.e., f_j \geq \lambda_{in} \cdot f_i, and \lambda_{de} for a negative wave, i.e., f_j \leq \lambda_{de} \cdot f_i, where 1\le\lambda_{in}<k and 1/k<\lambda_{de}\le1 (e.g., \lambda_{in}=1.3, \lambda_{de} = 0.8), meaning the change of frequency is not strong enough to meet the left shape condition yet, but can be regarded as a possible start of the left shape of a wave. 3) potential waves that have already satisfied the left shape condition...."
"..Particularly, the \emph{Stage 1} (initial stage) is responsible for recording the frequencies of current and \textcolor{red}{previous} windows for the \textcolor{red}{useless} items. The \emph{Stage 2} (weak potential stage) and the \emph{Stage 3} (potential stage) are responsible for recording the frequencies of the weak potential waves and potential waves, respectively..."
"...The \emph{Stages 1} filters out useless items and sends weak potential waves to the \emph{Stages 2} and potential waves to the \emph{Stages 3}. The \emph{Stages 2} filters out illegal weak potential waves among the recorded weak potential waves and sends potential waves to the \emph{Stages 3}. The \emph{Stages 3} filters out illegal potential waves and reports waves...."

D7: The description of the data plane algorithm (Alg 1) is a bit confusing, since there is no way for an item to get into stage 2 or stage 3. It should be explained that this will happen periodically at another stage of the algorithm.

Author Response D7: The data plane is only responsible for recording item frequencies in the current window. The control plane is responsible for recording all the other information of the three stages, identifying waves, removing illegal ones and moving items between stages. Note that the items in the three stages of the data plane match those in the control plane. The control plane sends item update information to the data plane at the end of each time window, and the data plane updates accordingly. In this way, the data plane structure is simplified and shrunk to an extremely small size, which can be cached by CPU multi-level caches (X86-based software version) or deployed to programmable switches (P4-based hardware version) for acceleration of the highly frequent item-by-item recording.

Author Actions D7:
Addition in Section 4.2, third paragraph, Page 4:
"...The data plane is only responsible for simply recording item frequencies in the current window. The control plane is responsible for recording all the other information and conducting more

complex operations of the three stages, including identifying waves, removing illegal ones and moving items between stages (e.g., S_1 -> S_2, S_1 -> S_3 and S_2 \rightarrow S_3)...."

D8: The description of the 4 situations in Section 6 should be related carefully to Definition 1, to help confirm if the algorithm is exactly matching the wave definition.

Author Actions D8:
Modification in Section 6, second paragraph, Page 5:
"...Situation 1: V_{d} \geq k \cdot V_{c} \land V_{d} \geq \mathcal{T}, indicating that an item increases k times. Situation 2: V_{d} \leq 1/k \cdot V_{c} \land V_{c} \geq \mathcal{T}, indicating that an item decreases k times. Situation 3: V_{d} > \lambda_{in} \cdot V_{c} \land V_{d} \geq \mathcal{T}/T_{in}, indicating that an item starts to increase. Situation 4: V_{d} < \lambda_{de} \cdot V_{c} \land V_{c} \geq \mathcal{T}, indicating that an item starts to decrease."

D9: Please indicate what st or vr refers to: are these abbreviations?

Author Response D9: Yes. V_{st}=V_{start} indicates f_i , V_{vr}=V_{variance} indicates f_j-f_i.

Author Actions D9:
Modification in Section 6, ninth paragraph, Page 7:
"...V_{st} indicates f_i which is the start curve point of a wave. V_{vr} indicates |f_j-f_i| which is the variance of a wave...."

D10: Please give more motivation for the Stage Variance Maximization step: what is the justification for this choice? Why is it called Stage Variance Maximization? It seems somewhat similar to the Conservative Update technique in sketching, or other techniques from hashing.

Author Response D10: The purpose of "Stage Variance Maximization" is to save the waves with large fluctuations, because the waves with large fluctuations are strong indicators for anomalies of the items in the data streams, which is inspired by Top-k and Conservative Update technique in sketching.

Author Actions D10:
Modification in Section 4.2, sixth paragraph, Page 4:
"...To solve hash collisions in the control plane, we use the Stage Variance Maximization technique to maintain waves with larger variances and reduce the estimation error. The purpose of Stage Variance Maximization is that the waves with large fluctuations are strong indicators for anomalies of the items in the data streams...."

D11: Section 7 will be quite unfamiliar to a DB audience. I would think it could be removed or moved to a technical report without any issue.

Author Response&Actions D11: As mentioned in **Author Response O1,**We believe that the introduction of hardware devices into the data mining community can motivate researchers for data stream processing solutions with high speeds.Considering that the DB audience does not know much about programmable switches, we will avoid the details of how programmable switches are deployed, shorten the length of Section 7, and only give a brief hardware solution.

Addition in Section 2, last paragraph, Page 2:
"Programmable Switches. Programmable switches (e.g., Barefoot Tofino~\cite{Tofino}) are an emerging networking technology that provides hardware programmability and flexibility without compromising performance.A representative programmable switch architecture is Protocol Independent Switch Architecture (PISA)~\cite{pisa}, where the ASIC chip consists of a programmable parser and a number of reconfigurable match-action tables. Operators can implement custom programs in the switch using domain-specific languages (e.g., P4~\cite{p4_language}), allowing the switch to process data traffic at terabits per second. With high line-rate guarantee and flexibility, programmable switches are ideal for wave detection and estimation."
Modification in Section 7, Page 7&8:
"We implement three stages of Pontus in programmable switches to improve the performance of Pontus. With the Stateful Algorithm and Logical Unit (Stateful ALU) in each stage of the switch pipeline, we can lookup and update the entries in the corresponding register array.
There are three differences between the P4 hardware deployment version and the software version of Pontus. 1) Due to the resource limitation of Stateful ALU, we only store the key and value field in physical registers. For insertion, we need to go back to the register that has the smallest value and reset the key and value when there are hash collisions in S_1. However, this process is not allowed in the P4 language. Hence, we use the resubmit primitive to solve this problem. When a packet is resubmitted to the beginning of the pipeline, it can maintain up to eight bytes of metadata in the resubmit header. We use the resubmit metadata to record the necessary information (e.g., the counted packet number) for the replace operation. 2) As multiplications, divisions, and floating-point operations are not supported in P4, it is difficult to calculate the probability. In P4, to approximate probabilistic replacement, we generate a 32-bit random number r and replace the smallest bucket in S_1 if (r << L_V) < 2^{32}, in which L_V represents the bit furthest to the left in V. 3) The control plane is deployed on the local CPU of the programmable switch or a remote server. At the end of each window, the programmable data plane sends values in registers of different stages to the control plane. After updating, the control plane sends the updated information back to the data plane through rewriting physical registers in the switch pipeline."

D12: The results of section 8 seem too separated from the algorithms that they apply to. I would suggest to integrate them into the relevant section.

Author Response D12: We will modify Section 8 and prove upper bounds of false negative and positive in Section 8.2.

Author Actions D12:
Modification in Section 4.2, second paragraph, Page 3&4:
"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde}

further verify that these error bounds are guaranteed to be small with properly selected parameters."

Modification in Section 8.2, Page 8&9:

Please refer to the paper.

D13: As mentioned, Lemma 8.1 is quite similar to previous results in sketching. Thm 8.1 follows the classical Count-min sketch result.

Author Response D13: We will modify Section 8 and prove upper bounds of false negative and positive in Section 8.2.

Author Actions D13:

Modification in Section 4.2, second paragraph, Page 3&4:

"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde} further verify that these error bounds are guaranteed to be small with properly selected parameters."

Modification in Section 8.2, Page 8&9:

Please refer to the paper.

D14: Theorem 8.2 analyzes the probability of hash collisions in S2 -- why is this an important function to analyze, and how should we interpret the bound? The description is somewhat unclear -- random variable Y is for a single bucket, so the event of Y >= 2 only applies to one bucket, not to the whole structure. So I think this analysis needs to be patched up. Equation (3) seems it should have a negative in the exponent. The claim "in practice, the potential waves inserted into S2 are less than..." makes it sound like some assumption is being made, when in fact (4) is trivially true if the previous bound holds.

Author Response D14: As mentioned in **Author Response O3**,the probability of false positive and false negative is related to the error bounds of S1 and the hash collision probability of S2 and S3. Therefore, we prove them in Section 8 and by setting the parameters of Pontus reasonably, the probability of false positives and false negatives can be guaranteed to be small. Y is a random variable that indicates the number of distinct items in a single bucket. The hash collision happens on a bucket when the number of distinct items in this bucket is greater than or equal to 2, i.e. Y>=2. For the insertion of an item, it is hashed d^{\hat} times. When all buckets of d^{\hat} hashes have hash collisions, the insertion of this item into S2 or S3 is unsuccessful. Thus, thm 8.2 proves an upper bound (i.e., an upper bound on hash collisions) for the failure probability of item insertion into S2 or S3. The exponent in Equation (3) should be negative, we will fix this. Thm 8.2 gives an upper bound on hash collisions in S2 and S3, but in general the hash collision probability is less than this bound.

Author Actions D14:

Modification in Section 4.2, second paragraph, Page 3&4:

"...The Multi-Stage Progressive Tracking strategy can significantly save memory overhead. As the space complexity of Pontus is O(1) << O(N), hash collisions are inevitable in Pontus. Though Pontus cannot achieve absolute correctness, we proof the upper bounds on the probability of false positives and false negatives for Pontus in Section \ref{MA} and experiments in Section \ref{wde} further verify that these error bounds are guaranteed to be small with properly selected parameters."

Modification in Section 8.2, Page 8&9:

Please refer to the paper.

D15: The absolute ARE values achieved of 0.2 upwards seem quite high -- it would be useful to clarify what these mean in practice for applications.

Author Response D15: As mentioned in Section 9.1, the ARE is calculated from 4 curve points, so the ARE in our paper is usually about 4 times larger than the previous. In our paper, ARE is used to judge whether an algorithm's characterization of the wave shape is accurate. A lower ARE means that the detected wave shape is closer to the ground truth. For example, in the optical network failure scenario, the fluctuation of the measured wave is much larger than the ground truth, which will erroneously report to the enterprise and allow experts to solve the problem, resulting in additional economic costs.

Author Actions D15:

Modification in Section 9.1, third paragraph, Page 10:

"Average Relative Error (ARE): \frac{1}{\Psi} \sum_{e \in \Psi} \sum_{a \in \mathcal{C}}\frac{\lvert a - \hat{a}\rvert}{a}, where a is the true curve point of item e, \hat{a} is the estimated curve point, \mathcal{C}=\{a_i,a_j,a_n,a_m\} is true curve points set of item e, and \Psi is the true waves set. We use ARE to evaluate error in wave estimation."

Reviewer #3

Many thanks for comments!

O1: As someone not particularly familiar with the field, it was not immediately clear to me what it means to *estimate* a wave or a burst. It wasn't until the evaluation described calculating estimation error that this seemed to be clearly specified. Clarifying this upfront would be helpful.

Author Response O1: We will describe *estimate* in detail in Section 3.2 and add more sketch background and related works to help readers get a better view.

Author Actions O1:
Addition in Section 3.2, third paragraph, Page 3:
"Wave Estimation. For each detected wave, the wave curve is estimated by its frequencies during \{t,t+1,...,t+(w_{in}+w_{l}+w_{de})\} windows, which is hard to realize within limited memory. Thus, we use the curve points f_i,f_j,f_n, and f_m as shown in Figure 1, to approximately estimate a wave curve instead. Note that wave estimation can be turned off if it is unnecessary for specific scenarios or the memory is insufficient. "

O2: The explanation of the algorithm is very dense. While the detail is appreciate, I found it difficult to track the state maintained during each stage through the process. I think simple changes such as using neg/pos instead of n/p using h[k] = v instead of "set h[k].V to v" would make things much easier to read. The notation also seems inconsistent.

Author Response: We have rechecked the manuscript and fixed the mentioned and other errors to improve the quality of the paper.

Author Actions O2:
Addition in Section 5&6, Page 5-7.

O3: Although the detailed descriptions are helpful, I find they sometimes simply mirror the algorithms in words without providing much useful context. Some of these explanations could be significantly shortened.

Author Response O3: We will shorten detailed descriptions and add more context information in Sections 5 & 6. We will give more useful context in Section 4.2.

Author Actions O3:
Addition in Section 5&6, Page 5-7.
Modification in Section 4.2, second&third&fifth paragraph, Page 3&4:
"...The Multi-Stage Progressive Tracking is from the core idea of "filtering", where we filter out useless items, i.e., non-fluctuated items, as early as possible, and only maintain and track useful items (fluctuating items), to save memory and improve throughput. We classify the items in the data stream into three types: 1) useless items, which violate wave conditions already. 2) weak potential waves, items that have started to show increase or decrease trends but haven't satisfied the left shape condition (i.e., f_j \geq k \cdot f_i or f_j \leq \frac{1}{k} \cdot f_i) yet. The \emph{weak

potential wave} is identified by the thresholds \lambda_{in} for a positive wave, i.e., f_j \geq \lambda_{in} \cdot f_i, and \lambda_{de} for a negative wave, i.e., f_j \leq \lambda_{de} \cdot f_i, where 1\le\lambda_{in}<k and 1/k<\lambda_{de}\le1 (e.g., \lambda_{in}=1.3, \lambda_{de} = 0.8), meaning the change of frequency is not strong enough to meet the left shape condition yet, but can be regarded as a possible start of the left shape of a wave. 3) potential waves that have already satisfied the left shape condition...."

"..Particularly, the \emph{Stage 1} (initial stage) is responsible for recording the frequencies of current and \textcolor{red}{previous} windows for the \textcolor{red}{useless} items. The \emph{Stage 2} (weak potential stage) and the \emph{Stage 3} (potential stage) are responsible for recording the frequencies of the weak potential waves and potential waves, respectively..."

"...\emph{Stages 1} filters out useless items and sends weak potential waves to \emph{Stages 2} and potential waves to \emph{Stages 3}. \emph{Stages 2} filters out illegal weak potential waves and sends potential waves to \emph{Stages 3}. \emph{Stages 3} filters out illegal potential waves and reports waves..."

"...The data plane is only responsible for recording item frequencies in the current window. The control plane is responsible for recording all the other information of the three stages, identifying waves, removing illegal ones and moving items between stages (e.g., S_1 -> S_2, S_1 -> S_3 and S_2 -> S_3)...."

"...To solve hash collisions in the control plane, we use the Stage Variance Maximization technique to maintain waves with larger variances and reduce the estimation error. The purpose of Stage Variance Maximization is that the waves with large fluctuations are strong indicators for anomalies of the items in the data streams...."

Modification in Section 5, third paragraph, Page 4&5:

"Example: Figure 2 shows an example of insertion. We set d=1,\hat{d} =1,\mathcal{T}=10,k=2 and item frequency f = 1 for simplicity. 1) To insert e_3, we use hash function g(\cdot) to map it to bucket (e_3, 4) in S_3 and increment V by 1 (from 4 to 5). 2) To insert e_9, we map it to bucket (e_6, 1),(e_3,9) in S_3, S_2, sequentially. Both of the K are not equal to e_9. Then we map it to bucket (e_7,8) in S_1. The K is also not equal to e_9. We replace this bucket to (e_9,9) with probability \frac{1}{9}.}"

Modification in Section 6, eighth paragraph, Page 7:

"Example: Figure 3 shows an example of control plane update. Estimated frequencies in S_2,S_3 and K-V pairs in S_1 collected by the data plane are highlighted in green. We set T_{in}=T_{de}=3,T_l=10,d=1,\hat{d} =1,\mathcal{T}=10,k=2,W_t=8,\lambda_{in} = (k - 1) / T_{in} + 1 and \lambda_{de} = 1 - (1 - 1 / k) / T_{de}. For S_3, 1) To update e_1, since it matches Situation4, we check w_{de}=2 \geq T_{de}-1, thus e_1 is an illegal wave and we clear it to empty. 2) To update e_4, since it matches Situation2, we do w_{de}+=1 (from 1 to 2) and report it as a positive wave. 3) To update e_7, since it matches Situation2, we do w_{de}+=1. We successfully insert it into S_3 using Stage Variance Minimization. 4) To update e_{15}, since it matches Situation3, we use a temporary bucket tmp and set K=e_{15},t=8,V=6,w_{in}=1,type=p. We successfully insert tmp into S_2 using Stage Variance Minimization. Then we clear the K-V pair in S_1 to empty."

O4: A very brief description of SS and TLF which are used for comparison would be helpful for readers not already familiar with these algorithms.

Author Response O3: SpaceSaving (SS) maintains a hash table with k buckets, and each bucket contains a K-V pair. When we insert an item, 1) if the item is in the hash table, we increment its corresponding counter (V) by 1. 2) If the item is not in the hash table and the hash table is not full, we add the item to the hash table, and its counter is set to 1. 3) If the item is not in the hash table and the hash table is full, we remove the item with the smallest counter in the hash table and insert the new item in its place. We add 1 to the previous counter. Here we use the heap to find the item with the smallest count. Two-Level Filtering (TLF) uses two hash tables: the first-level filter and the second-level filter, which contain k_1 and k_2 buckets, respectively. Each bucket maintains a K-V pair. When we insert an item, 1) if the item is in the second-level filter, we increase its corresponding counter by 1. 2) If it is in the first-level filter, the insertion is the same as SS. After the insertion, if the counter exceeds the threshold T, we send it to the second-level filter, and clear its bucket of the first-level filter.

Author Actions O4:
Addition in Section 9.2, first paragraph, Page 10:
"...SpaceSaving \cite{MetwallyAA05} (SS) maintains a hash table with k buckets, and each bucket contains a K-V pair. Two-Level Filtering \cite{VenkataramanSGB05} (TLF) uses two hash tables: the first-level filter and the second-level filter, which contain k_1 and k_2 buckets, respectively. We extend the bucket in SS and TLF to maintain more information for tracing waves and construct baseline methods based on them for comparison in Section 9.2-9.3."

D1: It would be helpful to mention briefly what a CM sketch is.

Author Response D1: We will discuss CM sketch in detail in related work.

Author Actions D1:
Addition in Section 2, second paragraph, Page 2:
"...For example, the CM sketch consists of d arrays, each consisting of l counters. The d arrays are associated with d pairwise independent hash functions. To insert an incoming item e, the CM sketch first calculates d hash functions and insert e to the mapping counter in each array. To query an item e, the CM sketch returns the minimum estimated frequency in the d mapping counters. The C sketch is similar to the CM sketch. The difference lies in that the C sketch uses different d hash functions to get an unbiased estimation of an item and return the medium estimated frequency in d mapping counters."

D2: Avoid adjacent section headings such as 9/9.1

Author Response D2: We have rechecked the manuscript and fixed the mentioned and other errors to improve the quality of the paper.

Author Actions D2:
Addition in Section 9, first paragraph, Page 9:
"In this section, we first introduce the experiment setup and metrics. Then we conduct experiments to evaluate Pontus."