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: We will add it to our related work.

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.

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: We will add it to Section 4.2.

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 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: We will add it to Section 9.1.

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.

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.

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: We will rewrite the scenarios in the introduction to emphasize wave's application in data mining scenarios. We will describe the knowledge of hardware devices in detail in related work to help DB and DM readers have a better perspective.

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: . The classic heavy hitter detection method is CM Sketch or C Sketch combined with heap and detect within a time window. In Pontus, we need to consider the hash collision of the T_{in}+T_l+T_{de} (>10) time windows. For an item, we need to consider at least 2^10 cases which is not practical. Besides, TopicSketch and BurstSketch are also unable to prove error bounds. Therefore, we prove the error bound for S1 and the hash collision probability for S2 and S3, which implies the probability of false positive and false negative (as mentioned in **Author Response O3**). We verify that the ARE of Pontus is small and PR,RR is high through sufficient experiments.

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. False positives happen only when 3 conditions are met: 1) the item has a hash collision in S1, and it is sent to S2 or S3 due to the hash collision, 2) the item does not suffer a hash collision in S2 and S3, 3)the item finally satisfies the right shape conditions and is reported. E.G.: the freq. of item e in the previous time window is 100, and k is 2. Its freq. in current time window is 150. Due to hash collision, its freq. is thought to be 500. Therefore, e satisfies $f_j>k\cdot f_i$ of the left shape condition in the pos. wave (Alg. 4 line 3) and is sent to S3. Its freq. is 50 in the next time window. e satisfies the $f_m<1/k\cdot f_n$ of the right shape condition in the pos. wave (Alg. 4 line 8-9) and is falsely reported as a positive wave. False negatives happen when: 1) the item satisfies the left shape condition and is sent to S2 or S3 (not because of hash collisions), 2) the item is filtered out at S2 or S3 because of hash collisions. 3) In the next few time window, the item will satisfy the right shape condition. We find that the probability of false positive and false negative are related to the error bounds of S1 and the hash collision probability of S2 and S3.

Author Actions O3: We will add it to Section 4.2.

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: In Section 8.1, we prove the error bound of S1 on a case-by-case basis, which is similar to [5]. Note that our probabilistic replacement is different from [5]. The difference are in Section 5.The error bound of [5] is tighter than ours. However, we insert an item much faster than [5] because [5] needs to traverse all buckets. In Thm1, we relax the error bound of S1 to be consistent with the error bound of CM Sketch. In general, it is much tighter than CM Sketch. The error bound of Pontus is intractable since we need to excessive number of cases. We find that the probability of false positive and false negative are 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. The experimental results further verify this claim.

[5] Ran Ben-Basat and Gil Einziger. 2017. Randomized admission policy for efficient top-k and frequency estimation. In INFOCOM. 1–9.

Author Actions O4: We will illustrate the difference from [1] in Section 8.1. We do not relax our error bounds in thm1.

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

Author Actions D1: We will fix it.

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: We will describe this in detail in Section 4.1.

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.

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 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: We will modify it in Section 4.2.

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: Please refer to **Author Response D10**.

Author Actions D5: We will describe "Stage variance maximization" in detail in Section 4.2 to help readers get a better perspective.

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: We will add it to Section 4.2.

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: We will add it to Section 4.2.

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: We will carefully check the description of the 4 situations in Section 6.

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: We will describe it in detail in Section 6.

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: We will describe it in detail in Section 6.

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. We place the detailed discussion in the appendix for interested readers.

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 & Actions D12: Please refer to **Author Response O2&O4**.

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 & Actions D13: Please refer to **Author Response O2&O4**.

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&Actions 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.

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: We will discuss ARE in detail in Section 9.1.

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 & Actions 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.

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

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 & Actions O3: We will shorten detailed descriptions and add more context information in Sections 5 & 6.

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: We will give a brief description of SS and TLF in Section 9.2.

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

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

D2: Avoid adjacent section headings such as 9/9.1

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