A Reliable Replica Mechanism for Stream Processing

Weilong Ding[1(✉)], Zhuofeng Zhao[1], and Yanbo Han[2]

[1] Data Engineering Institute, North China University of Technology, Beijing, China
dingweilong@ncut.edu.cn
[2] Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Beijing, China

Abstract. In Internet of Things, data would be fast generated from massive sensors as real-time data stream, and the replica mechanism is essential to guarantee availability during stream processing. Traditional mechanisms always assume the redundant replicas were exactly correct, but in the practical conditions even slight errors of replica would lead to the calamity for recovery. In this paper, a reliable mechanism is proposed in which space-bounded signature of checkpoint is used for validation during the replica placement. The mechanism has been analyzed theoretically, and also demonstrated by extensive experiments in various conditions.

Keywords: Stream processing · Replica · Availability · Space-bounded · Signature

1 Introduction

In Internet of Things, data is fast generated from massive sensors of many business scenarios, and these real-time, continuous and no-boundary data is termed as data stream. For stream processing, low latency and high throughput is elementary requirements, the failure of any processing element (**PE** for short) not only cuts off the data flow to the downstream, but may also overflow the memory of upstream in chain reaction [1]. Therefore, high availability (**HA** for short) guarantee for stream processing is necessary due to the velocity nature of data stream [2, 3]. As the most favorite HA, the replica of PEs can keep partial data, and would rebuild status of failed PE. Traditional replica mechanisms assume the redundant replicas were exactly correct, but even the slight errors of replica would lead to the calamity for recovery. In this paper, we propose a reliable replica mechanism through space-bounded signature, which is used to validate checkpoint when the replica is placed. We also balance extra overheads during the backup phase, such as bandwidth, memory and CPU. Our contributions conclude as follows. (1) The replicas can be fast validated through space-bounded signature. The validation is efficient in high probability. (2) The optimal tradeoff between overheads is well studied according to physical capacities. The performance holds steady when data scales up.

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 397–407, 2017.
DOI: 10.1007/978-3-319-59288-6_36

This paper is organized as follows. Section 2 shows the background including motivation and related works. Section 3 elaborates our replica mechanism with checkpoint algorithm. Section 4 quantitatively evaluates performance and availability guarantee through extensive experiments in various conditions. Section 5 summarizes the conclusion.

2 Background

2.1 Motivation

Our work comes from the practical project *Grid Transportation* in Chinese city Shenzhen. In this metropolitan city, thousands of cameras are deployed on trunk roads to recognize the passing vehicles for traffic analysis. A basic data unit ***tuple*** would be packaged by a camera in one second, which includes a recognized vehicle-plate, vehicle type, capture time, lane number, camera number, two photos (front and back view) and other 22 attributes. We have developed a backend system in Cloud to do more than 10 business jobs, such as fake-plate detection, copy-plate detection, black-list detection and accompanied-vehicle analysis, etc. It is typical requirement of stream processing that any tuple should respond within no longer than three seconds since its arrival. In our system, more than 40 distributed machines in Cloud are involved, and each processing unit on one machine is termed as ***PE*** [1, 4]. The periodical checkpoint as replica is required for specific PE, because that PE is apt to overload and then fail when the data stream is in high velocity. We found it difficult to guarantee replica's validity especially in unstable network condition. For example, in our system a PE executes quantile aggregation on the tuples in recent 30 s to detect traffic jam. When fails in rush hour, its recovery may not complete from replica due to the network congestion or some errors in replica. However, current replica mechanisms assume the replica is always correct and neglect the slight errors from network in practice. A reliable mechanism for stream processing is required to validate the replica efficiently. That is just our original motivation.

2.2 Related Work

For stream processing, the replica mechanisms can be classified by two perspectives [5], which imply different tradeoffs between performance and overheads. (1) In backup phase at run-time, replica mechanism has three types: active standby, passive standby and upstream backup. The replicas in those three types have the decreasing independence, while the more independent the replica is, the higher overheads and lower latency would cost. (2) In recovery phase at fail-time, replica mechanism could have three types either: gap recovery, rollback recovery and precise recovery. Those three types have increasing precision after recovery, while the more precise the recovery guarantees, the higher overheads and longer latency would suffer. Each type has its own advantages and limitations, and both perspectives have to be considered in current systems. For example, Borealis [6] adopts active standby with precise recovery; SAND [7] uses upstream backup and passive-standby to guarantee precise recovery; MillWheel [8] provides

passive standby with gap recovery; Flume [9] and Storm [10] uses upstream backup to implement rollback or precise recovery. Our work employs the passive standby at run-time to guarantee gap recovery at fail-time.

Due to the flexible tradeoff, the PE's checkpoint as replica is the most commonly used mechanism in current systems [2], and current works focus on the optimization in certain conditions. However, the assumptions that the replica would be correctly transferred through network may not always hold in practice, because real network condition is unpredictable. There is no validation for the replicas' correctness. The replica mechanism in SAND [7] can solve the uncertainty from the disordered tuples by new type PE *coordinator*; but its strong consistency guarantee restricts the throughput of stream processing. In our work, limited extra bits as signature are included in the replica, by which the checkpoint can be validated fast and correctly enough. Moreover, this configurable signature can balance between the correctness and the bandwidth overhead during the backup phase.

3 Replica with Space-Bounded Signature

3.1 PACK Protocol for Passive Standby

We propose the PACK (Passive replica with check) protocol to provide reliable replica, with the help of a monitor module to manage status and the configurations. For passive standby, traditional replica is a synonym for checkpoint, but the replica through PACK includes checkpoint and its signature. The sequence diagram of PACK is showed in Fig. 1. (1) In Step 1, a checkpoint algorithm is employed to build replica with signature, which would be discussed in the next section. (2) In Steps 2–6, replica is placed among machines; in Steps 4–6, the checkpoint would be validated by monitor module. Therefore at fail-time, after detecting a PE's failure, the monitor selects one of its replicas on another machine, downloads the program package of the failed PE and transfers it to that machine. The PE would be initiated from the program package and then rebuilt from the replica.

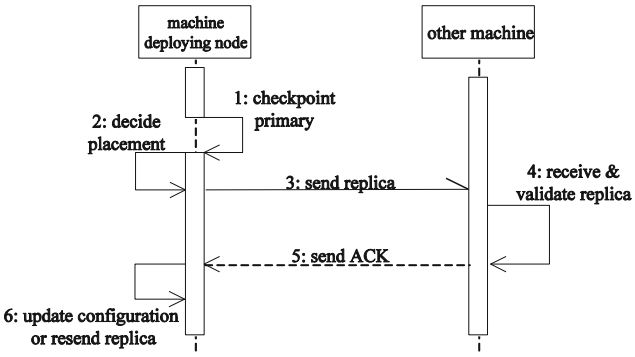

Fig. 1. The PACK protocol at run-time

PACK extends the Sweeping Checkpointing protocol (**SC** for short) [11] in IBM CLASP whose validity has been proved formally. Compared with SC, PACK has the advantages below. First, the checkpoint is programmable, and a signature of checkpoint is attached in the replica. The content of replica can be tuned via the signature. Second, the checkpoint can be validated from its signature after the replica is placed. If unsuccessful, the replica is abandoned and the monitor would resend a replica again after the notification. Third, the replicas are built asynchronously with their own timer and configurable intervals. It is more suitable for stream processing especially in high velocity.

There are two assumptions in our work. (a) All PEs are in clock-synchronization. It can be easily ensured through NTP (Network Time Protocol) servers. (b) The PEs' failure comes from machines' fail-stop. It is sensible because we focus on the replicas among distributed machines. One PE cannot be recovered only if all its replicas are inaccessible in the system.

3.2 Checkpoint Algorithm

To validate the checkpoint, some extra bits as the signature are required. But how many bits are adequate is the very problem, which implies how much the additional bandwidth overhead would cost. In the following, we assume that the checkpoint own m binary bits and its size is M. That is, $M = 2^m$ and $m = \ln M$ where $\ln x$ expresses the logarithmic value based on 2. A lemma is proved below to show that problem is hard.

Lemma 1. To correctly validate a checkpoint containing m bits, at least m bits have to be checked through a deterministic algorithm.

Proof. The least erroneous case is that only the rth bit in the checkpoint is reversed during the transfer. $1 <= r <= m$. In this case, r uniformly and randomly lies in $[1..m]$ and could not be determined in advance. Therefore, to definitely and correctly locate r in a deterministic algorithm, at least m bits have to be checked in this replica. ■

In deterministic algorithm, the extra bits for validation have the same magnitude as the checkpoint itself. It is impossible and impractical due to the doubled the bandwidth. Therefore, we propose a checkpoint algorithm to build replica using space-bounded bits as signature to balance the validation correctness and bandwidth overhead. We name this algorithm as SCC (Sweeping Checkpoint with Check).

```
Input: a natural number i bigger than 2
Output: a replica including checkpoint with signature

SCC algorithm
1 begin
2   if (checkpoint needs inputQueue)
3    checkpoint += inputQueue;
4   end if
5   while (state && (checkpoint needs state))
6     stateMap.add(state);
7   end while
8   checkpoint += stateMap;
9   checkpoint += outputQueue;
10  p = randomPrime(i);
11  rem = checkpoint mod p;
12  replica = [checkpoint, p, rem];
13  return replica;
14 end
```

The parameter i of a natural number bigger than 2 is termed as ***check index***. The ***signature*** of given checkpoint is composed by p and *rem*, and the replica include checkpoint with its signature. As the line 9, the output queue is always included in the checkpoint, while the input queue or internal states are optional on demand. The signature of checkpoint is generated at lines 10–11: p is random prime and *rem* is the remainder of *checkpoint* as binary values divided p. The function *randomPrime*(i) generates $p <= 2^{i*\ln m}$, where m is the bits of the checkpoint and i restricts the range of p. To get this random prime, it is the straightforward method to generate a random number and test whether it is a prime or not; if not repeat until the test is true. However, the classical prime test algorithms like Miller-Rabin [12, 13] or AKS [14] have high time complexity. For that reason, a trick is adopted here to fast find a random prime: we maintain enough primes [15] beforehand as a table in memory, and select a required prime (less than $2^{i*\ln m}$) randomly in that table.

Accordingly, the replica validation can be deduced as these steps. First, the replica is unpacked as three pieces *checkpoint*, p and *rem*. Second, if the remainder of *checkpoint* dividing p equals *rem*, the validation is claimed to be true; otherwise is false. Due to the random prime p, there is the opportunity to get a false claim for the validation, when the remainder equals *rem* but the checkpoint is not identical than the original one. We then prove that error probability is bounded and small enough.

Theorem 1. Through the replica built by SCC algorithm, the error probability of validation is less than i/m^{i-2}. Here, i is check index, m is the bits of the checkpoint, and the random prime $p <= 2^{i*\ln m}$.

Proof. Assume an original checkpoint C becomes the C' after the transfer through network, # is the set size, *Prime* is the set of primes, and *length*(p) returns the bit-length of random prime p. The error probability $\Pr_{err}$ of validation can be deducted as:

$$\begin{aligned}\Pr\ err &= \Pr((C - C') \bmod p = 0 | C \neq C') \\ &= \frac{\#\{p | p \in Prime, \text{length}(p) <= i * \ln m, (C - C') \bmod p = 0\}}{\#\{p | p \in Prime, \text{length}(p) <= i * \ln m\}} \\ &< \frac{m}{\Pi(m^i)} < \frac{i * m}{m^{i-1}} = \frac{i}{m^{i-2}}\end{aligned}$$

Here, the first inequality holds due to these facts. $C, C' \in \{1, 2, 2^2, \ldots, 2^m\}$, so $(C - C') \in \{1, 2, 2^2, \ldots, 2^m\}$. $(C - C') \bmod p = 0$ implies $(C - C')$ is a composite number and could be expressed as the multiplication of a series of primes. That is, $(C - C') = 2 * 3 * 5 \ldots * p' < 2^m$. Here, p' is at most the m^{th} prime, which makes the numerator less than m. Meanwhile, the denominator is the number of all the primes whose bits are less than $i*\ln m$, and can be expressed as $\Pi(2^{i * \ln m}) = \Pi(m^i)$, where $\Pi(x)$ is the number of the prime which is less than x.

The second inequality is holds due to these facts. According to prime number theorem [16], $\Pi(x) \approx x/\ln x$. Therefore, $\Pi(m^i) \approx m^i/(i*\ln m) > m^i/(i*m) = m^{i-1}/i$. Then, considering the numerator, we conclude $\Pr_{err} < m/\Pi(m^i) < i/m^{i-2}$. ■

Therefore the upper bound i/m^{i-2} of error probability correlates with the check index i and m like Fig. 2. The larger i or larger checkpoint size (larger m) would make this bound tighter exponentially, which implies lower error probability of validation.

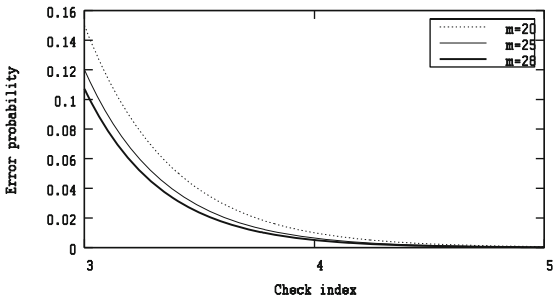

Fig. 2. Theoretical upper bound of error probability for validation.

Moreover, we prove that the extra required bandwidth overhead from the signature is also bounded as the theorem below.

Theorem 2. The extra bandwidth from the signature generated by algorithm SCC in the replica is less than $2^{1+i*\ln m}$. Here, m is the bits of checkpoint and i is check index.

Proof. The bandwidth increase comes from the signature in replica which includes two pieces p and rem. As the remainder on the divisor p, rem must be smaller than p. Therefore, the extra bandwidth is $p + rem < 2*p <= 2*2^{i*\ln m} = 2^{1+i*\ln m}$. ■

This upper bound of bandwidth overhead correlates with the check index i and the checkpoint size either. The larger i or larger checkpoint size (as larger m) would make this bound looser, which implies larger bandwidth consumption. In fact, we can tune

the tradeoff through the check index: larger i is required when checkpoint correctness is the emphasis; otherwise smaller i should be set due to the economical bandwidth.

4 Experiment and Evaluation

For the quantitative and qualitative evaluation, a distributed data stream processing system VINCA-CCS [1, 4, 17] is used for experiments. Any machine in this system owns 4 core CPU, 8 GB RAM, 100 GB storage and CentOS 5.5 × 86 64 bit, which is the virtualized one via VMware vSphere™ 5.1 in our private Cloud on eight Dell PowerEdge™ rack servers in local area network. A data generator [1, 4, 17] is employed to simulate streams from the offline data in our practical project. By default, the generator simulates 1000 connections, and sends data as the speed 1 tuple/s per connection. The tuple as data unit includes 22 attributes including vehicle-plate and photos. Both the PACK protocol and original SC protocol are implemented in VINCA-CCS for comparison, and we would evaluate the tradeoff between the correctness and extra bandwidth from SCC algorithm.

In the topology as Fig. 3, the PE *communicator* and its downstream PE *mq* are deployed in different machines, where *communicator* receives the raw data and packs them as structural tuples; *mq* receives tuples and keeps the payload in given JMS (Java Message Service) topics. Moreover, an endpoint-to-endpoint connector HAC-node [17] is used as the monitor module, which run as PACK or original SC respectively to build replica *cp* for *communicator* periodically. Here, the check index of SCC algorithm of PACK is set 3 by default.

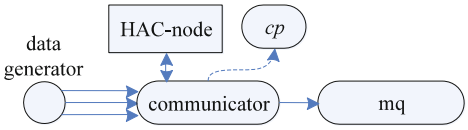

Fig. 3. Topology sample for experiments

Experiment 1. In data generator, the concurrency is set as 1000 TCP connections and the data speed is kept as 1 tuple/s/connection. The checkpoint interval for *communicator* is set 500 ms. Through both protocols, the memory of *communicator* is monitored at run-time.

Experiment 2. In data generator, the concurrency is set 30 and the data speed is kept as 1 tuple/s/connection in data generator. Record the memory used by *communicator* through two protocols respectively.

Experiment 3. In data generator, the concurrency is set 10000 and the data speed is kept as 1 tuple/s/connection. Record the memory used by *communicator* through two protocols respectively.

The replica size is the same magnitude of the memory consumed, which is also the bandwidth overhead. In experiment 1, we find the memory used is about 4 MB and

deduce the bits of checkpoint $m = \lceil \ln 4\ \text{MB} \rceil = 25$. Accordingly in the experiments 2 and 3, we find the used memory is about 100 KB and 30 MB and deduce the m is 20 and 28 respectively. For those experiments, the Fig. 4 presents the theoretical worst of bandwidth overhead.

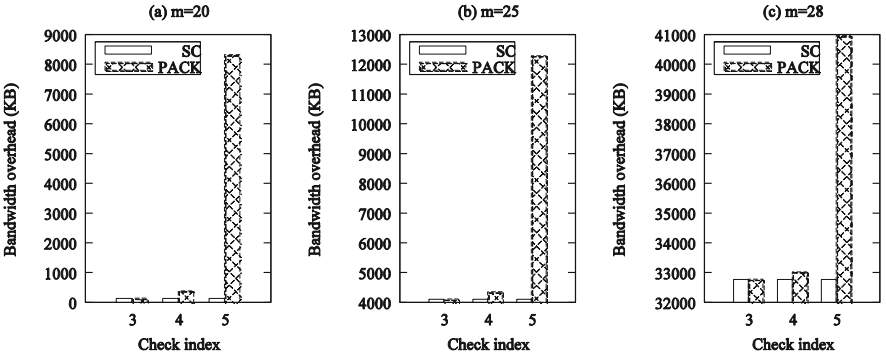

Fig. 4. Upper bound of bandwidth overhead

We can discuss the tradeoff theoretically. (1) As the Theorem 1, the upper bound decreases exponentially when i increase, and the larger checkpoint has tighter upper bound for the same check index. Take the given checkpoint in experiment 1 ($m = 25$) as an example. When i is set 3, the size of rem is $\lceil 3 * \ln m \rceil = 3 * 5 = 15$ bits. According to the Theorem 2, the extra bandwidth is less than $2 * 2^{15} = 8$ KB. Meanwhile, according to the Theorem 1, the upper bound of error probability for validation is less than $3/25 \approx 12\%$. While if i is set 4, the size of rem is $\lceil 4 * \ln m \rceil = 4 * 5 = 20$ bits. The extra bandwidth is less than $2 * 2^{20} = 256$ KB and the upper bound of error probability for validation is less than $4/25^2 \approx 0.64\%$. Then, compare the checkpoints in experiment 1 ($m = 25$) and 3 ($m = 28$). When i is set 4 in both experiment, the checkpoint in experiment 3 would be validated correctly in higher possibility, because it is larger (30 MB > 4 MB) with the tighter error bound. (2) Larger check index implies less error but more bandwidth overhead, which is tunable in practice. Note that check index cannot be set too large either. In those experiments, when i is set 3, 4 or 5, theoretical worst extra bandwidth through PACK are 8 KB, 256 KB and 8 MB respectively. However, when $i = 4$, 256 KB signature is larger than the 100 KB checkpoint ($m = 20$) itself; when $i = 5$, 8 MB signature is much larger than 100 KB checkpoint ($m = 20$) or 4 MB checkpoint ($m = 25$). It seems that huge checkpoint has more options on check index and more flexible tradeoff.

In brief, by the check index, it is the tunable tradeoff between the bandwidth and the replica correctness. Next we evaluate the performance through both protocols.

Experiment 4. In data generator, the data rate is set from 1 to 10 tuples/s/connection and the data speed is kept as 1 tuple/s/connection. In HAC-node through PACK, the check index is set 4. Repeat tests to get the average of the system response time under each data rate.

The result is showed in Fig. 5(a).

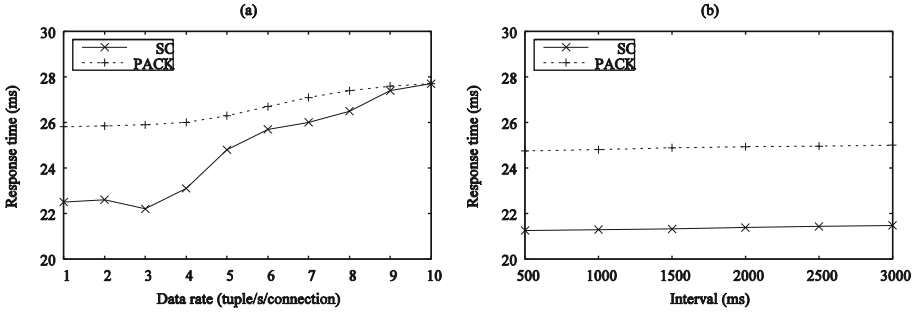

Fig. 5. Performance through both protocols

Experiment 5. In data generator, the concurrency is set as 1000 and the data speed is kept as 1 tuple/s/connection. In HAC-node, the checkpoint interval is set from 500 ms to 3 s and the check index through PACK is set 4. Repeat tests to get the average of the system response time under each interval.

The result is showed in Fig. 5(b).

We found the response time grows as data rate increases through both protocols, but the relative increment shrinks through PACK than SC under the same data rate. The growth of data rate makes larger checkpoint at given backup interval, but the relative increment through PACK becomes smaller because the proportion of signature in replica relatively decreases for larger checkpoint. Moreover, we found the response time is steady when backup interval increases through both protocols; the difference between PACK and SC is no more than 3 ms under the same interval. It is the reason that the CPU consumption for checkpoint through PACK is small enough due to the space-bounded signature. It also demonstrates the Theorem 2 in Sect. 3.2.

In brief, through PACK, the performance holds in multiple conditions. Next we simulate the errors of replica to evaluate the availability guarantee.

Experiment 6. In data generator, the concurrency is set as 1000 and the data speed is kept as 1 tuple/s/connection. In HAC-node, the checkpoint interval is set as 500 ms and the check index through PACK is set 3. In HAC-node, we simulate errors by flipping random bits of checkpoint before the replica is sent to another machine. The size of flipped bits is set 2, 5 and 10 respectively. After replica placement, we can compare the replica with the original one to find whether error appears. Repeat tests hundreds of times to get the average ratio of errorless replica under each size of flipped bits.

The result is showed in Fig. 6. The ratio here reflects the reliability in the unstable network condition. When flipped bit grows, we found the errorless ratio through PACK remains above 95%, but that of SC drops dramatically. The difference between them comes from the replica validation through PACK. In PACK, there are three possibilities for the flipped bits: first, the flipped bits entirely lie in the checkpoint; second, flipped ones entirely lie in the signature; third, those ones span the checkpoint and the signature. By our SCC algorithm, the signature is far small than the other, so that the first case remains relatively high possibility. In the first case the validation must be unsuccessful, and the replica has to be resent until the validation is successful. PACK makes the correct

replica in high possibility; contrarily, SC has to appear errors as long as at least one bit is flipped.

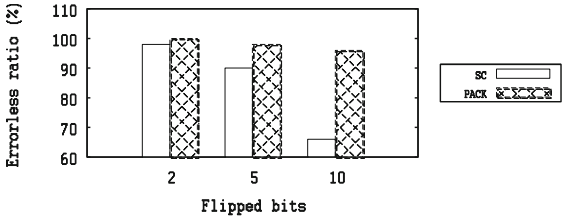

Fig. 6. Errorless ratio through both protocols

In brief, through PACK high availability is guaranteed.

5 Conclusion

We propose a reliable replica mechanism for stream processing to guarantee the correctness of the checkpoint. The replicas can be validated using space-bounded signature fast and correctly with high probability. In various conditions, the performance can hold by the tunable tradeoff and high availability is guaranteed.

Acknowledgment. This work was supported by the R&D General Program of Beijing Education Commission (No. KM2015_10009007), the Key Young Scholars Foundation for the Excellent Talents of Beijing (No. 2014000020124G011) and Foundation for the Excellent Youth Scholars of North China University of Technology (XN072-006).

References

1. Ding, W., Han, Y., Wang, J., Zhao, Z.: Feature-based high-availability mechanism for quantile tasks in real-time data stream processing. Softw. Pract. Experience **44**, 855–871 (2014)
2. Bockermann, C.: A Survey of the Stream Processing Landscape. Lehrstuhl furk unstliche Intelligenz Technische Universität Dortmund (2014)
3. Barlow, M.: Real-Time Big Data Analytics: Emerging Architecture. O'Reilly Media Inc., Sebastopol (2013)
4. Ding, W., Han, Y., Wang, J., Zhao, Z.: Feature-based high availability mechanism for extreme aggregation tasks in real-time data stream processing. J. Internet Technol. **14**, 327–340 (2013)
5. Hwang, J.H., Balazinska, M., Rasin, A., Cetintemel, U., Michael, S., Stan, Z.: High-availability algorithms for distributed stream processing. In: The 21st International Conference on Data Engineering, pp. 779–790 (2005)
6. Balazinska, M., Balakrishnan, H., Madden, S.R., Stonebraker, M.: Fault-tolerance in the borealis distributed stream processing system. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 13–24. ACM (2005)
7. Liu, Q., Lui, J.C., He, C., Pan, L., Fan, W., Shi, Y.: SAND: a fault-tolerant streaming architecture for network traffic analytics. In: The 44th Annual IEEE/IFIP International

Conference on Dependable Systems and Networks (DSN 2014), Atlanta, Georgia, USA, pp. 80–87 (2014)
8. Akidau, T., Balikov, A., Bekiroglu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., Whittle, S.: MillWheel: fault-tolerant stream processing at internet scale. Proc. VLDB Endow. **6**, 1033–1044 (2013)
9. http://archive.cloudera.com/cdh/3/flume/
10. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J.M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., Ryaboy, D.: Storm@twitter. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 147–156. ACM, Snowbird (2014)
11. Gu, Y., Zhang, Z., Ye, F., Yang, H., Kim, M., Lei, H., Liu, Z.: An empirical study of high availability in stream processing systems. In: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, pp. 1–9. Springer, New York (2009)
12. Miller, G.L.: Riemann's hypothesis and tests for primality. J. Comput. Syst. Sci. **13**, 300–317 (1976)
13. Rabin, M.O.: Probabilistic algorithm for testing primality. J. Number Theory **12**, 128–138 (1980)
14. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Ann. Math. **160**, 781–793 (2004)
15. http://primes.utm.edu/lists/small/millions/
16. http://en.wikipedia.org/wiki/Prime_number_theorem
17. Ding, W., Zhao, Z., Han, Y.: A framework to improve the availability of stream computing. In: 23rd IEEE International Conference on Web Services (ICWS 2016), pp. 594–601. IEEE (2016)