Системы хранения и обработки
потоков больших данных

Методы обеспечения устойчивости ко сбоям

Протоколы синхронизации. Хранение состояния.

Самарев Роман Станиславович, 2023 (черновые наброски)

МГТУ им. Н.Э. Баумана
Кафедра Компьютерные системы и сети

План презентации

Основные характеристики: 4V – volume, variety, velocity, and veracity (объем, разнообразие, скорость, достоверность)

Основные проблемы

- Выявление нарушенного порядка сообщений
- Синхронизация сообщений по нескольким каналам
- Подтверждение обработки сообщений
- Надёжная обработка операторов с состоянием

Упорядочивание обработки сообщений

Основные протоколы:

- Slack (ленивое выявление)
- Heartbeat (сердцебиение)
- Low-watermark (минимальный уровень)
- Pointstamps (штампы)

M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos. A survey on the evolution of stream processing systems, 08 2020

Упорядочивание обработки сообщений. Slack

Протокол Slack:

- Сообщения имеют метку, определяющую их порядок.
- Нарушение порядка выявляется по этой метке (порядковый номер, время, и пр.).
- *slack* $= 1$ означает величину "льготного периода"обработки оператором для опаздывающих сообщений.

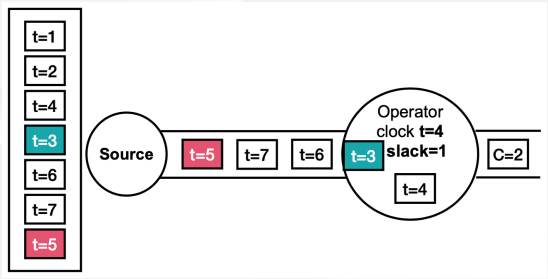


Иллюстрация из [9]

Упорядочивание обработки сообщений. Heartbeat

Протокол Heartbeat [24]:

- Централизованный источник меток.
- В определённые моменты времени в поток добавляется метка времени (или порядковый номер).
- Оператор не может принимать сообщения, имеющие время ранее этой метки.
- В распределённой среде может быть рассогласование локального серверного времени.

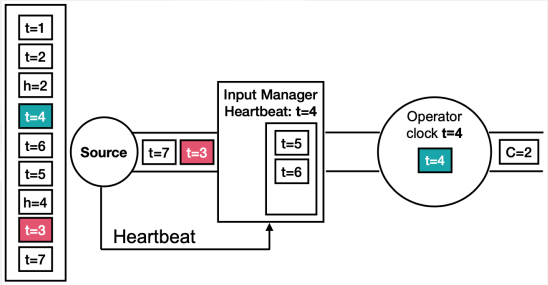


Иллюстрация из [9]

Упорядочивание обработки сообщений. Low watermark

Протокол Low-watermark [15]:

- Отметка минимального уровня определяется для некоторого подмножества сообщений.
- Для оператора метка определяет старейшее сообщение, которое может быть обработано.
- В отличии от Slack и Heartbeat, метки внедряются в поток.
- Метка с метаданными называется "пунктуацией"(punctuations).

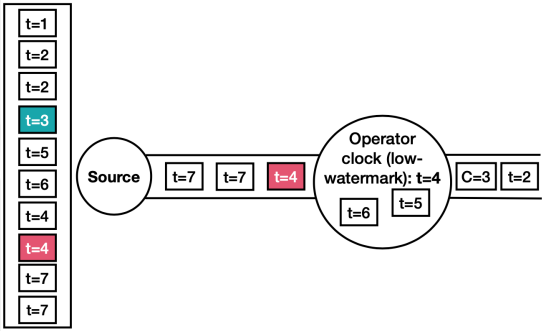


Иллюстрация из [9]

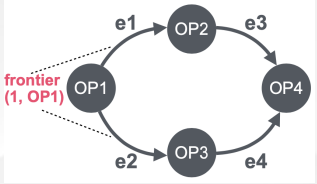
Упорядочивание обработки сообщений. Pointstamps

Протокол Pointstamps [20]:

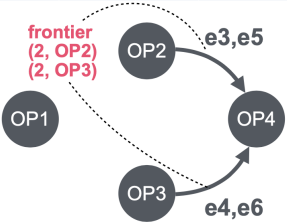
- Каждому сообщению назначается метка времени и координаты в графе операторов обработки.
- На рисунке, e1 и e2 не обработаны. Получают метку по текущему положению $(1, OP1)$. Счетчик вхождения - 2, поскольку узел OP1 - источник для e1 и e2.
- События e3, e4 - обработаны OP2 и OP3.
- Результат обработки $e1 \rightarrow e5$, $e2 \rightarrow e6$. Метки аналогичны e3, e4.
- После обработки OP4 метки удаляются.

Иллюстрация из [9]

Active Pointstamp	Unprocessed Event(s)	Occurrence Count	Precursor Count
(1, OP1)	e1, e2	2	0
(2, OP2)	e3	1	1 (1, OP1)
(2, OP3)	e4	1	1 (1, OP1)



Active Pointstamp	Unprocessed Event(s)	Occurence Count	Precursor Count
(2, OP2)	e3, e5	1	0
(2, OP3)	e4, e6	1	0



Устойчивость ко сбоям в потоковых системах

System	Processing semantics			Replication			Recovery data			Storage medium			
	Least	Exactly-once		Active	Passive	No	State	Output	No	Resilient store		In-Memory	No
		State	Output							Local	Remote		
Aurora* [48]	✓				✓			✓				✓	
TelegraphCQ [119]		✓		✓			✓	✓		✓			
Borealis [8, 26]	✓			✓			✓	✓				✓	
S4 [111]	✓					✓			✓				✓
Seep [37, 57]			✓		✓		✓	✓			✓		
Naiad [108]		✓			✓		✓	✓			✓		
Timestream [114]			✓		✓		✓	✓			✓		
Millwheel [12]			✓		✓		✓	✓			✓		
Storm [131]	✓					✓			✓				✓
Trident [3]			✓		✓		✓				✓		
S-Store [39, 126]		✓			✓		✓			✓			
Trill [41]		✓			✓		✓			✓			
Heron [90]	✓					✓			✓				✓
Streamscope [97]			✓$_{a,p,r}$	✓$_{a}$	✓$_{p}$	✓$_{r}$	✓$_{p}$		✓$_{a,r}$		✓$_{p}$		✓$_{a,r}$
Streams [77]		✓			✓		✓				✓		
Samza [112]	✓				✓		✓			✓			
Flink [33, 34]		✓			✓		✓				✓		
Spark [20]		✓			✓		✓				✓		

M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos. A survey on the evolution of stream processing systems, 08 2020

Операторы по модели состояния

Операторы потоковой обработки:

- **Stateless operators** - операторы без хранимого состояния, где чистный функциональный подход предполагает вычисление результата, зависимое только от текущих входных данных
- **Stateful operators** - операторы с храненением состояния, где результат вычисления зависит от предудыщих входных данных

Q.-C. To, J. Soto, and V. Markl. A survey of state management in big data processing systems.

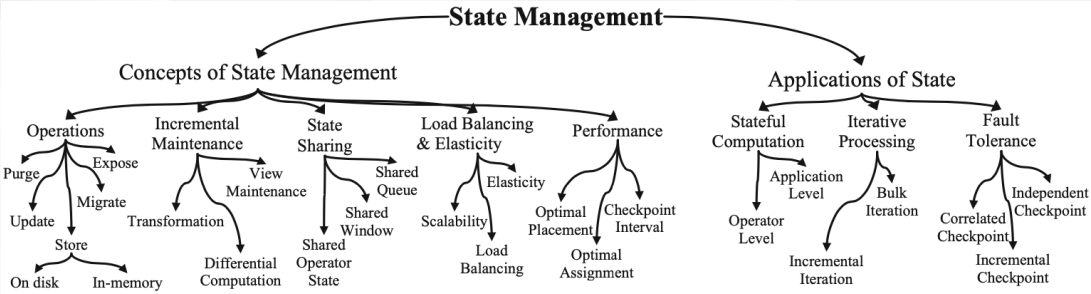
***The VLDB Journal*, 27(6):847–872, Dec. 2018**

Хранение состояния

4V - volume, variety, velocity, and veracity (объем, разнообразие, скорость, достоверность)

Решаемые с помощью хранения состояния задачи:

- Гарантированная обработка
- Агрегация данных, оконные операции
- Обслуживание внешних запросов



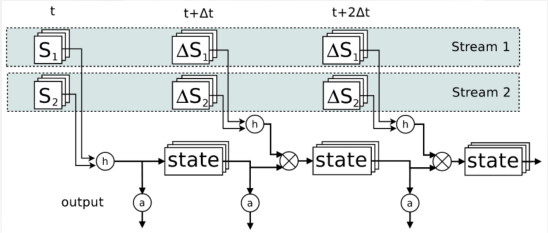
Q.-C. To, J. Soto, and V. Markl. A survey of state management in big data processing systems.

***The VLDB Journal*, 27(6):847–872, Dec. 2018**

Операции. Способы хранения состояния

- сброс состояния на локальный диск [17].
- сброс состояния в распределённую файловую систему типа GFS, HDFS. SGuarf [14].
- Adaptive Incremental Check-pointing (AI-Ckpt) - разбиение состояния по страницам памяти и сброс только изменённых страниц [22].
- координация в географически распределённой системе. Phonon [1]. Используется дополнительный регистрационный слой IdRegistry.

- incremental state update – проблема формирования изменений состояния. Иллюстрация последовательного изменения состояния из [7].



- fine-grained update – проблема контроля влияния обновлений на производительность и задержку.
- consistent update – проблема построения графа зависимостей по внесённым изменениям.
- update semantics – семантика подтверждения операций – at-least-once, at-most-once, and exactly-once.

Операции. Purging state

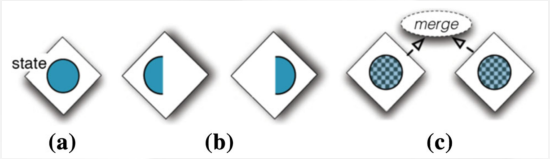
Избавление от ненужных данных

- punctuation based [26] – добавление метаданных и удаление ненужного при выполнении условий с различными модификациями.
- Eager purge/lazy (batch) purge [6] – модификация предыдущего подхода с непосредственным удалением всех данных после получения метки или ожидание окончания пакета.
- out-of-order processing (OOP) [16] – Order-agnostic Aggregation, Join Implementations для части операций.

Миграция состояния. Проблема балансирования нагрузки

- Динамическая миграция для непрерывных планов выполнения запросов – "подвижное состояние параллельные изменения.
- Перенос состояния между узлами в рамках выполнения одного оператора.
- Миграция состояния с явным определением места размещения.
- MigCEP - план миграции, т. е. вероятностная структура данных, описывающая будущие цели и время миграции.
- рандомизированное репликации (на основе Multilevel Counting Bloom Filter) и адаптивная схема репликации.

Раскрытие состояния. Восстановление состояния после сбоя, переразмещение операторов по узлам, выполнение оптимизации.



a - элемент состояния, **b** - состояние, разбитое на два раздела, **c** - множественные экземпляры частичного состояния.

Иллюстрация из [8]

Общее состояние. Использование одного состояния для нескольких операторов.

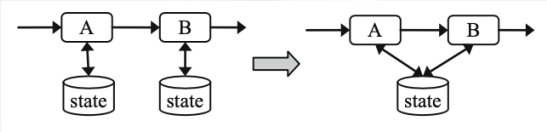


Иллюстрация из [25]

Задачи - уменьшить накладные расходы от точек восстановления и обслуживать состояние последовательно

- differential computation
- incremental view maintenance
- MRQL Streaming
- viewlet transforms

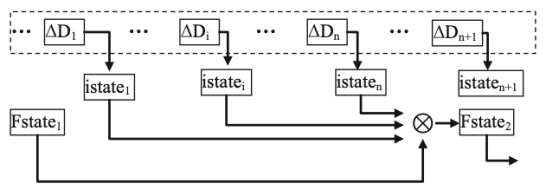
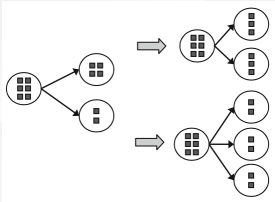


Иллюстрация из [25]

Динамическое перераспределение нагрузки
по доступным узлам
Иллюстрация из [25]



Тип разбиения данных	Система/ работа	Основная задача
Hash based	[23]	Разбиение состояния и маршрутизация потоков данных
Partial-key based	[10]	Функции разбиения
	[12]	Добавлена стоимость агрегации
	[21]	Распределение по ключу и оценка локальной нагрузки
	[2]	Связывание ключа с двумя возможными узлами
Executor centric	[27]	Эластичное исполнение + планировщик на основе модели
Migration based	[28]	Транзакционный протокол миграции и сопоставление потоков-раздел

Производительность(Performance)

- Влияние частоты сохранения состояний
- Сложность оптимального размещения состояния
- Сложность оптимального назначения состояния

Характеристика	System	Основной механизм	Цель
Batch processing	[19]	Indexing	Избежать последоствального перебора
	[3]	State as explicit input	Минимизировать движение данных
Stream processing	[18]	Partitioned stateful operators	Балансировать нагрузку
	[4]	Parallel patterns	Улучшить параллельное выполнение

Таблица из [25] `Implementations of state and limitations`

Устойчивость ко сбоям (Fault tolerance)

- passive standby,
- active standby,
- upstream backup

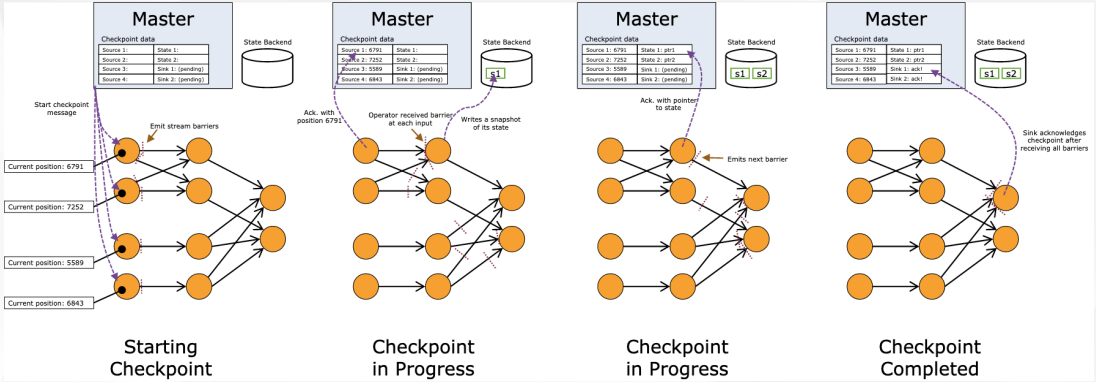
J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing.

In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, page 779–790, USA, 2005. IEEE Computer Society

Common characteristics	System	Main mechanism	Objective
Divide-and-conquer strategy	[85]	Checkpoint a small fragment of query graph	Efficient checkpointing and failure recovery
	[57]	Split operator state into control tuples	Balance recovery and running time
	[18]	Split states into slice units	Fast recovery
Adapts to computing environments	[26]	Utilize the similarity of access patterns	Adapt to scarce memory
	[96]	multi-level checkpointing with delta compression	Adapt to I/O and network bandwidth
N/A, only one system	[56]	Compute the optimal number of incremental checkpoints	Reduce checkpointing overhead
N/A, only one system	[95]	Inject barriers into data	Minimize space requirements

Таблица из [25]

Flink. Snapshotting Operator State



https://ci.apache.org/projects/flink/flink-docs-release-1.12/concepts/stateful-stream-processing.html

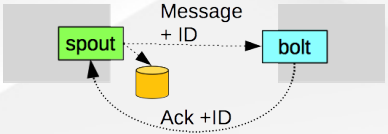
Подтверждение для каждого сообщения

Недостатки

- В распределённой среде из-за задержки подтверждения появляются дубликаты
- Низкая производительность

Достоинства

- Простота реализации



spout – источник сообщений
bolt – получатель сообщений

http://storm.apache.org/releases/current/Guaranteeing-message-processing.html

http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

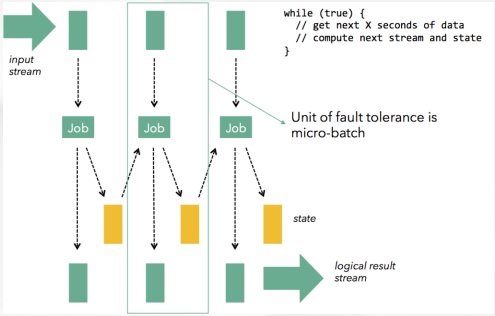
Подтверждение для каждого пакета сообщений (microbatching)

Недостатки

- Высокая задержка времени обработки и затруднён контроль обработки окна
- Возможно бездействие операторов при ожидании сохранения данных

Достоинства

- Теоретически высокая скорость



http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

https://databricks.com/blog/2015/07/30/diving-into-apache-spark-streamings-execution-model.html

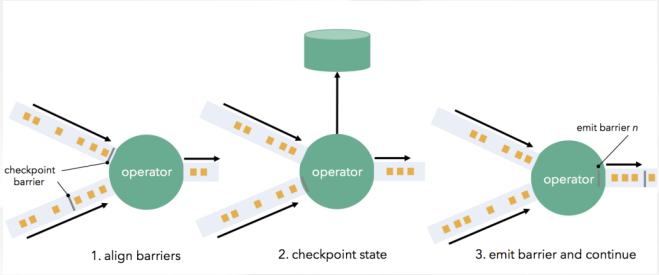
Asynchronous Barrier Snapshotting (ABS) [5]. Происходит выравнивание состояния по меткам "checkpoint barrier".

Недостатки

- Сложность реализации

Достоинства

- Высокая скорость сочетается с отсутствием простоя операторов



http://data-artisans.com/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink/

- Сигналы от источников направляются агенту (Tracker)
- Tracker агрегирует информацию про элементы в потоке от всех операций
- Tracker рассылает уведомления об окончании подпотока
- Каждая операция присваивает каждому выходному элементу случайное число и код операции, после отправляет в Tracker
- Операция, которая принимает элемент, отправляет то же число в Tracker
- Tracker агрегирует числа по предикату и применяет к ним XOR, если он стал 0 - подпоток закончился

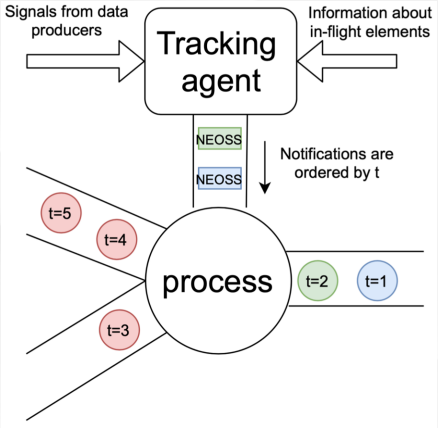


Иллюстрация из [13]

Сохраняемые состояния, оконные операции

- Реализации операторов агрегации требует сохранения и восстановления предыдущего состояния (Apache Flink, Apache Spark, Apache Storm/Trident)
- Внутреннее накопление данных необходимо для реалиации операций с окном данных. Обработка активируется триггером

Виды окон

- С перекрытием (по времени или количеству сообщений)
- Последовательные

Виды триггеров

- Интервальные
  - по времени поступления
  - по времени в обработке
  - по времени пользователя
- По количеству сообщений

https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/windows.html

Общая оптимизация (Integrative optimization)

Common characteristics	System	Main mechanism	Objective
Divide-and-conquer strategy	[85]	Checkpoint a small fragment of query graph	Efficient checkpointing and failure recovery
	[57]	Split operator state into control tuples	Balance recovery and running time
	[18]	Split states into slice units	Fast recovery
Adapts to computing environments	[26]	Utilize the similarity of access patterns	Adapt to scarce memory
	[96]	multi-level checkpointing with delta compression	Adapt to I/O and network bandwidth
N/A, only one system	[56]	Compute the optimal number of incremental checkpoints	Reduce checkpointing overhead
N/A, only one system	[95]	Inject barriers into data	Minimize space requirements

Таблица из [25]

Реализации хранимых состояний и их ограничения

Systems	State management	Fault tolerance	Guarantees
Storm	Not native	Tuples acknowledge	At least once
Storm + Trident	Specific operators	Tuples acknowledge	Exactly once
Heron	Stateful topologies	Tuples acknowledge	At least once
Samza	Stateful operators	Log of updates	At least once
Spark	State DStream	RDD lineage	Exactly once
Flink	Stateful operators	State checkpoint	Exactly once

Таблица из [25]

References I

[1] R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman. Photon: Fault-tolerant and scalable joining of continuous data streams. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, page 577–588, New York, NY, USA, 2013. Association for Computing Machinery.

[2] M. Anis Uddin Nasir, G. De Francisci Morales, N. Kourtellis, and M. Serafini. When two choices are not enough: Balancing at scale in distributed stream processing. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 589–600, 2016.

[3] A. Brito, C. Fetzer, H. Sturzrehm, and P. Felber. Speculative out-of-order event processing with software transaction memory. In *Proceedings of the Second International Conference on Distributed Event-Based Systems*, DEBS '08, page 265–275, New York, NY, USA, 2008. Association for Computing Machinery.

[4] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. *Information Retrieval*, 9:123–138, 03 2006.

[5] P. Carbone, G. Fóra, S. Ewen, S. Haridi, and K. Tzoumas. Lightweight asynchronous snapshots for distributed dataflows. *CoRR*, abs/1506.08603, 2015.

[6] L. Ding and E. A. Rundensteiner. Evaluating window joins over punctuated streams. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, page 98–107, New York, NY, USA, 2004. Association for Computing Machinery.

[7] L. Fegaras. Incremental query processing on big data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2998–3012, 2016.

[8] R. C. Fernandez, M. Migliavacca, E. Kalyvianaki, and P. Pietzuch. Making state explicit for imperative big data processing. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 49–60, Philadelphia, PA, June 2014. USENIX Association.

[9] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos. A survey on the evolution of stream processing systems, 08 2020.

[10] B. Gedik. Partitioning functions for stateful data parallelism in stream processing. *The VLDB Journal*, 23(4):517–539, Aug. 2014.

[11] J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, page 779–790, USA, 2005. IEEE Computer Society.

[12] N. R. Katsipoulakis, A. Labrinidis, and P. K. Chrysanthis. A holistic view of stream partitioning costs. *Proc. VLDB Endow.*, 10(11):1286–1297, aug 2017.

[13] I. E. Kuralenok, A. Trofimov, N. Marshalkin, and B. Novikov. Deterministic model for distributed speculative stream processing. In *ADBIS*, 2018.

[14] Y. Kwon, M. Balazinska, and A. Greenberg. Fault-tolerant stream processing using a distributed, replicated file system. *Proc. VLDB Endow.*, 1(1):574–585, Aug. 2008.

[15] J. Li, K. Tufte, V. Shkapenyuk, V. Papadimos, T. Johnson, and D. Maier. Out-of-order processing: a new architecture for high-performance stream systems. *PVLDB*, 1:274–288, 01 2008.

[16] J. Li, K. Tufte, V. Shkapenyuk, V. Papadimos, T. Johnson, and D. Maier. Out-of-order processing: a new architecture for high-performance stream systems. *PVLDB*, 1:274–288, 01 2008.

[17] B. Liu, Y. Zhu, and E. Rundensteiner. Run-time operator state spilling for memory intensive long-running queries. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, page 347–358, New York, NY, USA, 2006. Association for Computing Machinery.

[18] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum. Stateful bulk processing for incremental analytics. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 51–62, New York, NY, USA, 2010. Association for Computing Machinery.

[19] J. Meehan, N. Tatbul, S. Zdonik, C. Aslantas, U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, and H. Wang. S-store: Streaming meets transaction processing. *Proc. VLDB Endow.*, 8(13):2134–2145, sep 2015.

[20] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi. Naiad: A timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, page 439–455, New York, NY, USA, 2013. Association for Computing Machinery.

[21] M. A. U. Nasir, G. De Francisci Morales, D. García-Soriano, N. Kourtellis, and M. Serafini. The power of both choices: Practical load balancing for distributed stream processing engines. In *2015 IEEE 31st International Conference on Data Engineering*, pages 137–148, 2015.

[22] B. Nicolae and F. Cappello. Ai-ckpt: Leveraging memory access patterns for adaptive asynchronous incremental checkpointing. In *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '13, page 155–166, New York, NY, USA, 2013. Association for Computing Machinery.

[23] M. Shah, J. Hellerstein, S. Chandrasekaran, and M. Franklin. Flux: an adaptive partitioning operator for continuous query systems. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 25–36, 2003.

[24] U. Srivastava and J. Widom. Flexible time management in data stream systems. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, page 263–274, New York, NY, USA, 2004. Association for Computing Machinery.

[25] Q.-C. To, J. Soto, and V. Markl. A survey of state management in big data processing systems. *The VLDB Journal*, 27(6):847–872, Dec. 2018.

[26] P. Tucker, D. Maier, T. Sheard, and L. Fegaras. Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):555–568, 2003.

[27] L. Wang, T. Z. J. Fu, R. T. B. Ma, M. Winslett, and Z. Zhang. Elasticutor: Rapid elasticity for realtime stateful stream processing, 2017.

[28] W. Y. and T. K. Chronostream: elastic stateful stream computation in the cloud. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, page 723–734, 2015.