International Journal of Computer Engineering in Research Trends
Multidisciplinary, Open Access, Peer-Reviewed and fully refereed
Research Paper Volume-10, Issue-4 ,2023 Regular Edition E-ISSN: 2349-7084

SMERAS - State Management with Efficient Resource Allocation and Scheduling in Big Data Stream Processing Systems

B. Gnana Deepthi[1], K. Sandhya Rani[2], P. Venkata Krishna[3]
[1] *Dept. of Computer Science & Engineering, Sri Padmavati Mahila VisvaVidyalayam, Tirupati*
[2] *Rajiv Gandhi University of Knowledge Technologies, RGUKT-RK Valley*
[3] *Dept. of Computer Science & Engineering, Sri Padmavati Mahila VisvaVidyalayam, Tirupati*

e-mail: gdeepthi.bitra@gmail.com, sandhyaranikasireddy@yahoo.co.in, pvk@spmvv.ac.in

**Corresponding Author: gdeepthi.bitra@gmail.com*

https://doi.org/10.22362/ijcert/2023/v10/i04/v10i0401

Received: 12/02/2023, Revised: 17/03/2023, Accepted: 21/04/2023 Published:28/04/2023

Abstract: - Recent advances in big data and distributed computing systems having prominent data management with low cost of storage and efficient resource scheduling strategies leading reliable and scalable system designs. It helps for developing better decision-making systems in the era of Big Data. Speed of data arrival rate demands the speed of data processing rate. Existing scenarios uses complex query execution engines in distributed manner to process real-time and near real-time streaming data. Data Stream processing systems facing challenges with respect to resource management and are looking for the efficient resource scheduling and query execution strategies. In this paper, the SMERAS model is proposed and it uses a state full stream management based on a pipeline with various scheduling queues for managing streams of data. Experimental results show the performance analysis of the proposed system compared with the existing systems.

Keywords: Big Data, Stream Processing, State and Resource Management, Resource allocation and Scheduling

1. Introduction

Today's digital era concentrating on the challenges of data processing of rapid growing big data from various sources like sensors, computer and mobile devices, social media, e-commerce portals, business applications etc., To process real-life applications, real-time and near real-time data is more important than older data [1]. This data most probably considers as streams of volume and gives advances to the business and real-world applications to develop knowledge-based decision-making systems with valuable insights. Big Data Processing (BDP) approaches, batch and stream-oriented are the prominent fields of processing such massive data. MapReduce is the basic batch-oriented approach in BDPs. Map Reduce does not supports high-level programming and re-use of state in iterations in the processing [2]. Stream processing systems can overcome the limitations of batch approach and can provide features like flexibility, scalable, reliable and fault-tolerance.

Processing streams of data in real-time is the major requirement for the data stream applications. Data streams are the basic smallest element or unit in streaming applications and flows through the network as potentially infinite ordered streams of data. Streams are not affordable to store into memory and process from memory as can be deployed from the cloud clusters [3]. To increase the efficiency and performance of the system, multicore system architectures and distributed parallel computing approaches must be incorporated in the system design. Such systems can provide greater computing results like throughput and latency. More researches proposed different frameworks for

data stream processing systems in the field of big data computing.

In this paper, the contributions proposed are the implementation of the proposed architecture for stateful management of the operations and resource allocation of the real-time streams to reduce the starvation situation in the scheduling. The architecture considers distributed clusters for processing data streams in real-time with stateful operations. The remaining sections presented in this paper are as follows. Section 2 describes the preliminaries considered for the work. Section 3 is presenting the literature review. In section 4, proposed system architecture is given and the modules in the workflow of the system are described. In section 5, the results and discussion about the implementation of the system are provided. Section 6 containing the conclusion about the research work done.

2. Background

Most of the business applications are providing data-driven services for the users. The data management architectures need to be more consistent and scalable for the demand of the decision-making systems. Stream processing systems must react faster to the real-world event-driven applications [4]. It is adopted in major business fields and inspires researches for allowing scale-out and with higher throughput and latency. Figure 1 gives the evolutionary stages of data processing systems [5].

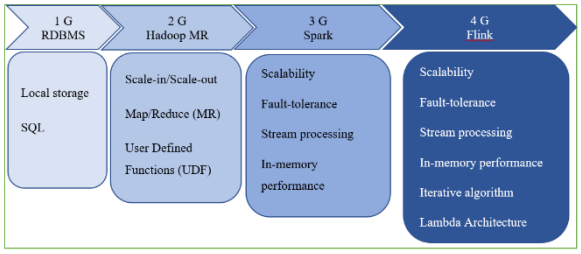

Fig 1. Evolution of Data Processing Systems

Apache Flink's [6] scalability is distributed to thousands of cores and stand-alone clusters can be connected to various open-source resource managers with a full software stack for organizing data streams. Its stateful operations make it as top in the stream processing applications. Data remembered for future computation is state of the application [14]. Data processing pipelines integrated with functional programming API to process continuous data flow events.

2.1 State Handling in Flink

State Management in stream processing systems done in stateless and stateful operations. In stateless operation, output depends on input and in stateful operation, sequence of inputs generates intermediate periodic results maintained in a data structure called state [7]. State management in data processing systems can be considered with respect to several aspects like operations on state, maintenance, sharing of state, performance, load balancing and applications of state.

2.2 Resource Scheduling in Flink

The resource allocation in the streaming applications is the major considerable problem for the performance and computational difficulty in big data computing [8]. In general, the computation done in distributed manner in the stream processing systems [9]. Various approaches are proposed for optimising the performance measures like reducing the computation time, minimising the resources, and efficiently balancing the load on the cluster of nodes. Here jobs are periodical and associated with periodical time, required resources, and scheduled to get optimum result [10].

Through Apache Flink [11], data events not stored in memory and executed or integrated dynamically within single JVM or within clusters of one or multi-nodes. This kind of storage operation is noted as sink means input data streams are exported into databases and no output is returned. User submitted programs compiled, pre-processed, and applied on the datasets as Directed Acyclic Graphs. Default query optimizer processes the graphs and minimizes the query execution time. Special memory data structures used for hashing and sorting leads efficient memory management.

3. Literature Review

Xiang Ni et al. [12] presented a generic strategy for resource allocation, by considered the unobserved data with implementing the reinforcement learning. Author considering the proposed problem as the NP-complete and using graph encoding and graph-aware decoder for getting the optimized results than the existing techniques.

Wasiur R. KhudaBukhsh et. al. [13] studied the heterogenous cloud clusters for understanding the job scheduling in the big data systems. The authors proposed randomised cost-based policy the job scheduling in the big data clusters. They achieved scaling in the several server buffers by implementing the numerical evaluations in the multi-stage systems.

Tiziano De Matteis et. al. [14] propoed algorithmic skeleton for state management in stream processing. The authors merged the structured parallelism for getting the better performance of throughput and latency in the stream processing state management. Multi-core architectures using Fastflow framework in the window aggregations of the state operations. Window farming, key partitioning and pane farming are the modules implemented in this work for doing the window partitioning.

Vasiliki Kalavri et. al. [15] developed workload-aware policy for the state management in the stream processing.

Here authors focussed on the limitations of the LSM-based distributed processing and with experimental results they gain higher throughput and latency.

Hui Zhao et. al. [16] proposed task scheduling algorithm for video transcoding. In three steps like data analysation, pre-processing and proposal of Job Shop Scheduling, authors implemented task scheduling strategy for minimising the transcoding time. The results showing the efficient nature of the algorithm for better load balancing. It combining two existing policies for making a hybrid approach of max-min and min-min algorithms.

Marios Fragkoulis et. al. [17] discussed about the survey taken for understanding the evolution of stream processing from the past 20 years. Major functional features of stream processing are considered for the survey. They are categorized as first generation and 2nd generation. Requirements, various architectures, disorders in the processing, challenges are discussed in their work.

Paris Carbone et. al. [18] discussed about the aggregations on sliding windows. They get feasible results for the records came as streams. Sliding windows used for handling input. Challenges considered are computing large records, and getting throughput.

Gerrit Janßen et. al. [19] developed an algorithm for scheduling the tasks with low latency and feasible bandwidth. Proposed algorithm is the search-based method. Results are almost 50% reduces the latency that obtained by existing algorithms. Fire detection benchmark is considered for the job graphs to schedule. Limited bandwidth is used for the experimental results.

4. SMERAS Architecture

Data flow in the Flink's processing pipeline is denoted as a directed acyclic graph. Pipeline's primary building block is state since it contains the complete status of the computation at any given point. Operator-state and Keyed-state are the two states maintained in the state management and that are reflected as exactly once in the operation update. In the keyed state, MapState and ReduceState API's [20] used for doing the mapping and reducing operations with key-value operations put and get.

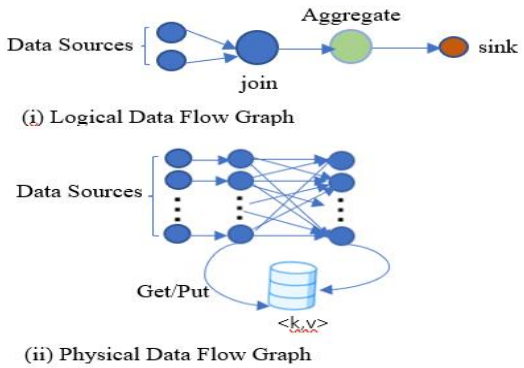

Fig 2. Data Flow Graphs in Stateful operations

The logical data flow graph G = {T, E} in figure (i), is mapped as G* = {T*, E*}, the physical data distribution graph showed in figure (ii) where the logical task t in G belongs to T (t ϵ T) is mapped as set of physical tasks in G* belongs to T* (t1, t2,…tn ϵ T).

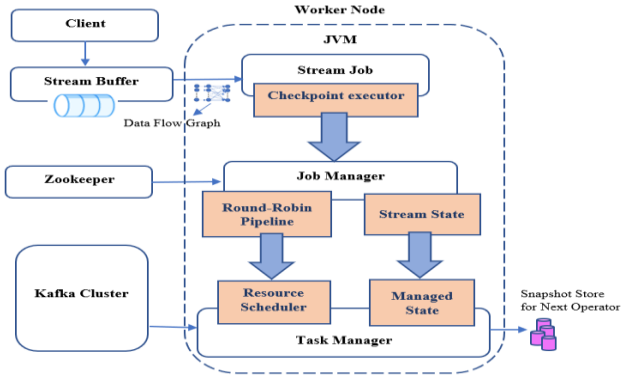

Fig 3. SMERAS Architecture for State management and Resource Allocation

In this architecture, the input streams are buffered into the stream buffer, and further converted to directed acyclic graph and mapped as number of tasks streams into the data pipeline. Session Window aggregation [18] is used which dynamically set by the dynamic change in the input stream and each logical task graph is redeployed as per the structure of the input. Zookeeper [21] maintains the metadata at the checkpoints. Job Manager do the task scheduling and maintains the state information.

Algorithm to schedule stream with SMERAS

Input ← configured stream jobs

$$J_s = \{ j_1, j_2, j_3, \ldots, j_n \}$$

1. # Set all the worker nodes based on resources availability.
2. # Load jobs into the stream buffer
3. # Set priorities to the jobs and time quantum T_q

4. # Set burst time B_t according to stream speed
5. # Build a pipeline to schedule the jobs J_s
6. foreach Job in J_s
7. if ($B_t < T_q$)
8. run job from the pipeline
9. state is updated
10. else
11. run job until T_q expires
12. repeat step 6
13. end if
14. end for

Round robin scheduling pipeline processes the sub tasks with in specific time intervals [22]. This can moderate the execution speed and waiting time of the tasks in the stream buffer dynamically with elastic pipeline. It reduces the starvation problem of large tasks with minimum waiting time. Snapshot information of all tasks is passed to next operator. Here the task scheduler is customizable and optimizes the resource management in a guaranteed scale.

The scheduling tasks queued into the stream pipeline as Js. On each worker node jobs loaded according to the load on the stream buffer that changes dynamically. Priorities and time quantum Tq assigned to each job in the queue. For each job, the burst time Bt compared with Tq, later went for processing and state updated. It will be iterated on the worker nodes.

Flink's ProcessFunctions [3] control state and time of each processing event by storing the state of the new arrival event and for the future event specific timestamp is registered. Here, past input is remembered and is used for influencing the future event processing. Current state is maintained here for the computing event and result is passed to the next events. State stored locally and distributed across cluster instead storing in external storage that guarantees consistency. Even in failure, state restored to the last checkpoint and efficient scalability can be achieved.

5. Results & Discussion

The proposed architecture is implemented for the stateful resource allocation strategy with the window aggregate based partition protocol. The dataset used here is the open accessible vehicle detection dataset available in Kaggle. In this section, the results of experiments are discussed and compared with the existing scheduling strategies.

The implementation takes place with the YARN cluster of 10 machines on the Amazon EC2 cloud and on 64-bit Cent OS with each having 16 CPU cores, 16 GB RAM and 512 GB SSD and HDD both. For state backend store, RocksDB out of core locality is used. Stream snapshotting done in Hadoop HDFS.

The experimental results are showed in the below figures 4-6. The results specifying the overall performance of the proposed SMERAS architecture is higher and resource utilization is improved with 7.8%, 7.6% and 9.4% of the CPU core and throughput ratio. The latency reduced for each set of window aggregations. The performance gain is in an average of 9.6% than the default system.

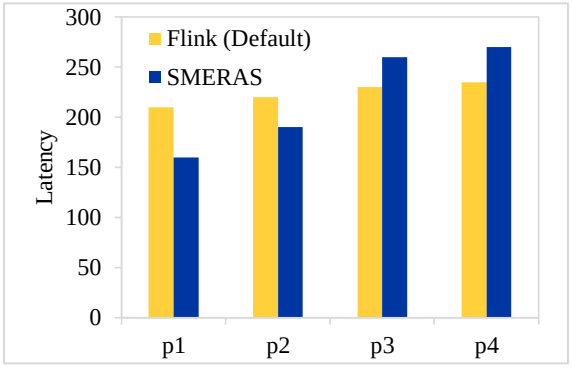

Fig 4. Latency measured for SMERAS and Flink (default) with respect to the incoming stream rates.

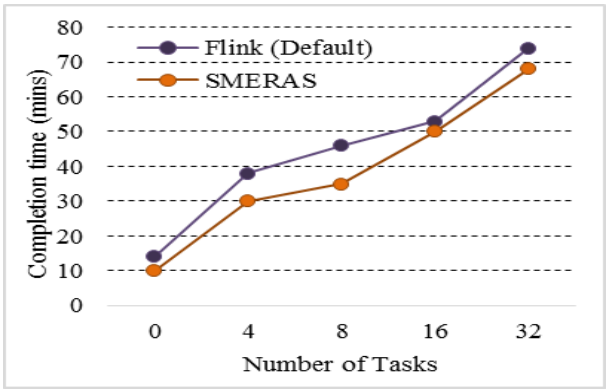

Fig 5. Throughput is measured with respect to the number of jobs per each core and the job completion time in mins.

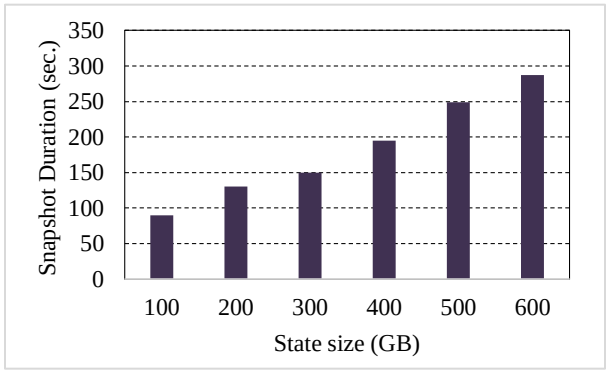

Fig 6. Snapshot Duration measurement in State size.

Overall results of this experiment showing that the proposed SMERAS architecture for the stateful resource management compared with the default behavior of the Flink [19] performance. The elastic window partition for the stream pipeline getting more accurate results for throughput

and latency. High resource utilization is observed for getting high throughput. And the SMERAS algorithm getting less completion time for the number of jobs processed.

6. Conclusion

In this work, the stateful operations and resource scheduling in the data stream processing systems analyzed efficiently with the proposed architecture SMERAS. It using the dynamic elastic data pipelines to stream the incoming events in the application. It producing better results than the existing researches with high throughput, low latency, and better resource utilization.

References

[1] Tian, L., & Chandy, K. M. (2006, September). Resource allocation in streaming environments. In 2006 7th IEEE/ACM International Conference on Grid Computing (pp. 270-277). IEEE.

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

[3] Asyabi, S. E. Toward Workload-Aware State Management in Streaming Systems.

[4] Langhi, S., Tommasini, R., & Della Valle, E. (2020, December). Extending kafka streams for complex event recognition. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 2190-2197). IEEE.

[5] Fragkoulis, M., Carbone, P., Kalavri, V., & Katsifodimos, A. (2020). A survey on the evolution of stream processing systems. arXiv preprint arXiv:2008.00842.

[6] Li, Z., Yu, J., Bian, C., Pu, Y., Wang, Y., Zhang, Y., & Guo, B. (2020). Flink-er: an elastic resource-scheduling strategy for processing fluctuating mobile stream data on flink. Mobile Information Systems, 2020, 1-17.

[7] de Souza, P. R., Matteussi, K. J., dos Anjos, J. C., Dos Santos, J. D., Geyer, C. F. R., & da Silva Veith, A. (2018, July). Aten: A dispatcher for big data applications in heterogeneous systems. In 2018 International Conference on High Performance Computing & Simulation (HPCS) (pp. 585-592). IEEE.

[8] Jiang, Y., Huang, Z., & Tsang, D. H. (2016). Towards max-min fair resource allocation for stream big data analytics in shared clouds. IEEE Transactions on Big Data, 4(1), 130-137.

[9] Kiruthiga, R., & Akila, D. (2021, January). Heterogeneous fair resource allocation and scheduling for big data streams in cloud environments. In 2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM) (pp. 128-132). IEEE.

[10] Ahmad, W., Alam, B., & Atman, A. (2021). An energy-efficient big data workflow scheduling algorithm under budget constraints for heterogeneous cloud environment. The Journal of Supercomputing, 77, 11946-11985.

[11] Tang, Z., Du, L., Zhang, X., Yang, L., & Li, K. (2021). AEML: An acceleration engine for multi-GPU load-balancing in distributed heterogeneous environment. IEEE Transactions on Computers, 71(6), 1344-1357.

[12] Ni, X., Li, J., Yu, M., Zhou, W., & Wu, K. L. (2020, April). Generalizable resource allocation in stream processing via deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 01, pp. 857-864).

[13] KhudaBukhsh, W. R., Kar, S., Alt, B., Rizk, A., & Koeppl, H. (2020). Generalized cost-based job scheduling in very large heterogeneous cluster systems. IEEE Transactions on Parallel and Distributed Systems, 31(11), 2594-2604.

[14] De Matteis, T., & Mencagli, G. (2017). Parallel patterns for window-based stateful operators on data streams: an algorithmic skeleton approach. International Journal of Parallel Programming, 45(2), 382-401.

[15] Kalavri, V., & Liagouris, J. (2020, July). In support of workload-aware streaming state management. In Proceedings of the 12th USENIX Conference on Hot Topics in Storage and File Systems (pp. 19-19).

[16] Zhao, H., Zheng, Q., Zhang, W., & Wang, J. (2016). Prediction-based and locality-aware task scheduling for parallelizing video transcoding over heterogeneous mapreduce cluster. IEEE Transactions on Circuits and Systems for Video Technology, 28(4), 1009-1020.

[17] Carbone, P., Fragkoulis, M., Kalavri, V., & Katsifodimos, A. (2020, June). Beyond analytics: The evolution of stream processing systems. In Proceedings of the 2020 ACM SIGMOD international conference on Management of data (pp. 2651-2658).

[18] Carbone, P., Katsifodimos, A., & Haridi, S. (2019). Stream Window Aggregation Semantics and Optimization.

[19] Janßen, G., Verbitskiy, I., Renner, T., & Thamsen, L. (2018, December). Scheduling stream processing tasks on geo-distributed heterogeneous resources. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 5159-5164). IEEE.

[20] Akil, B., Zhou, Y., & Röhm, U. (2017, December). On the usability of Hadoop MapReduce, Apache Spark & Apache flink for data science. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 303-310). IEEE.

[21] Tang, S., He, B., Liu, H., & Lee, B. S. (2016). 9 Resource Management in Big Data Processing Systems. Big Data Principles and Paradigm.

[22] Stein, O., Blamey, B., Karlsson, J., Sabirsh, A., Spjuth, O., Hellander, A., & Toor, S. (2020, December). Smart Resource Management for Data Streaming using an Online Bin-packing Strategy. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 2207-2216). IEEE.

