View Reviews

Paper ID
52

Paper Title
BriskStream: Scaling Data Stream Processing on Shared-Memory Multicores

Track Name
Research Paper First-Round

Reviewer #1

Questions

1. Overall Evaluation
Weak Accept

2. Reviewer's confidence
Expert

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The paper tackles the problem of operator placement in a parallel computation over a multi-core machine. The context is streaming and the results are applied to Storm, an open source streaming engine. The experimental results demonstrate significant gains when using the proposed techniques to allocate operators to cores.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. Results are convincing, even if somewhat expected. Performance advantages are substantial.

S2. The implementation in Storm is a plus as the authors show not only that the techniques are useful but that they can be deployed in a real system.

S3. The paper does a good job of explaining what is being done and where the performance advantages come from.

7. List 3 or more weak points, labelled W1, W2, W3, etc.
W1. The results are not a surprise. NUMA and multi-core parallelism has been a problem in many systems. Taking a platform like Storm or Flink which have been designed for parallelism at the cluster level and show that they do not do well at the multi-core level is not a groundbreaking discovery. The same can be said of the results. Any strategy that takes into account the interference problems that occur in a multi-core machine will beat a deployment ignoring them.

W2. The servers used for the evaluation are very large and not the usual 2 to 4 socket platforms. It would be important to show from which number of cores the results presented in the paper matter or whether the advantages can be demonstrated only on very large machines. experiments showing the results for a varying number of CPU sockets should be added.

W3. A common problem in these papers is that the baseline does nothing (see point W1). As a result, even the simplest of strategist to optimize the system will win over an un-optimized system. This makes it very difficult to determine whether the strat6egy proposed in the paper is the best or one out of many. Is there any way to come up with some ideal performance target and see how close one gets to it? After all, in some experiments,

the OS placement seems to do a good enough job.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
D1. Overall the paper is a strong engineering effort and the integration in Storm a major plus. I am less sure about the intellectual contribution for lack of a proper baseline. These systems have been designed for distributed execution (mainly to address the I/O problem in big data applications). The paper considers them in a single node scenario. That is fine (it has been proposed before) but still leaves the question open whether it would make more sense to run the same computation in two or three smaller computers. In a single box, no matter how many cores there are, the I/O problem will become very serious. The paper should comment on this and explain the scenarios that they have in mind.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
Yes

10. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc.
Come up with a more convincing explanation that the proposed method is among the best available. Right now the paper compares the method to simply doing nothing.

11. Comments on the revised paper (if appropriate)
I would like to thank the authors for addressing the comments in my review. The paper has now significantly improved.

12. Recommended decision for the revised paper (if appropriate)
Accept

Reviewer #2

Questions

1. Overall Evaluation
Weak Accept

2. Reviewer's confidence
Knowledgeable

3. Originality
Low

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The paper explores the problem of efficient data-stream processing on a single node shared memory multi-socket machine. The goal is to find the optimal execution plan deployment that maximizes the throughput performance. The presented system, BriskStream, uses a NUMA-aware optimization paradigm (titled Relative-Location Aware Scheduling) that considers both operator placement and replication. The optimizer leverages a performance model based on the rate-based optimization framework.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
(S1) The branch and bound algorithm with the three optimizations finds an execution plan deployment (replication + placement) that significantly improves the throughput and is much better than the alternatives.

(S2) The paper shows how one can use a modified rate-based optimization framework to estimate the performance of a stream operator in a multi-socket environment.

(S3) The paper builds upon the findings of a recent ICDE study that analyzed the performance of popular streaming engines when running on multicore environment and applies corresponding optimizations to significantly improve the performance.

7. List 3 or more weak points, labelled W1, W2, W3, etc.

(W1) There is no analysis on the complexity of the branch and bound search algorithm.

(W2) There is no discussion on the theoretical properties of the output of the b&b algorithm. How far from the optimal solution will the results be?

(W3) I am not convinced that the presented performance model required modifications to the RBO framework. It resembles a standard queuing theory where the arrival rate is a function of the output rate of the source operator and the number of hops distance between nodes.

(W4) The paper reads like a follow up on the ICDE study. But even then it is unclear how the optimizations and algorithm explained here relate to the optimizations used there. Hence a better comparison and discussion with respect to state-of-the-art will significantly strengthen the paper.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
Thank you for submitting your work to SIGMOD.

(D1) One of the biggest problems of the submission is the lack of discussion about the complexity and optimality of the deployment algorithm. The first aspect is important to address so that we understand how the algorithm will behave with different properties of the stream query (number of operations, complexity of the operators, complexity of the data-dependency graph, etc.). The second is important because the algorithm gives out an approximate solution, and it is always good to have a theoretical boundaries on how far off the optimal solution we can be.

(D2) The second concern I have is related to the performance modelling of the operators. More specifically, when you estimate r_o you mention that the processing rate of an operator depends on the varying input data fetch time. I am not convinced that this is the case. The processing rate of an operator should not depend on the input data stream. The arrival rate (input rate), however, yes. In this case, the arrival rate is a function of (1) the output rate of the upstream operator and (2) the number of hops distance between the nodes where the two operators are placed.

(D3) The third major concern is how this paper relates to the ICDE study by Zhang et al. which revisits the design of DSPs on multicore systems. It reads as a follow-up that fixes the inefficiencies that were discovered for the popular stream engines, but sadly it does not discuss or compare against the optimizations done in that paper. This is clearly the most related work to the submission and deserves a more thorough differentiation and discussion.

(D4) Provide more details for the servers that were used in the experiments. The short description provided in section 2.1. is insufficient. Also make sure to provide references where needed. What is XNC?

(D5) On page 4, where you discuss the inputs to the model you use an unlabeled forward reference to the optimization algorithm, which is not introduced to the reader yet. Please fix it.

(D6) It is OK to say that for the sake of simplicity of presentation the model assumes that the selectivity of the operators is 1, but you need to discuss the implications of this assumption in one of the later sections of the paper.

(D7) Please be more specific when discussing the benefits of the NUMA-awareness that you added to the RBO. It would be nice if in your evaluation you compare against the predictions done by the original framework.

(D8) When you describe the instantiation of the model, you place all the operators on a single socket. Can you fit all of them in a single socket and what happens if you cannot? What happens when you have more complex query plans?
What are non-relevant operators? I am also curious on how you avoid to problem of resource interference when

co-scheduling the operators on a single socket with many shared resources. Also what is a queue blocking time here? Please be more specific and provide more details and clarification in this section.

(D9) Please rename section 3.2 to something more descriptive.

(D10) How do you determine the values of C, B, and Qi,j? Do you perform microbenchmarks using a particular tool? Similarly with M and T? Do you get the from the performance model? I am concerned that this is not a scalable approach given the diversity of operators (exacerbated when supporting user defined functions).

(D11) On page 8, you say that in the implementation of the algorithm you set the ratio to be 5, but you do not elaborate on why that makes a good trade-off?

(D12) How do you measure the time breakdown of an operator for execution time, RMA and others?

(D13) The result discussion/analysis provided in Section 5.2. is not really clear. What do you mean by a more regular address?

(D14) Please proofread the paper for typos and grammar mistakes. There are quite a few of them. For example:
- First sentence of intro: [..] superior performance *for* latency-sensitive applications, with their *increasing computing* capability.
- You use "on the other hand" even though there was no "on the one hand" before that.
- There is no need to use strong adjectives and adverbs like "seriously"

(D15) Also, please write more descriptive captions to the tables and figures and ensure that the results presented in the figures can be read if the paper is printed in black and white. At present, for instance, the legend for Figure 3 is not readable.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
Yes

10. Comments on the revised paper (if appropriate)
I thank the authors for doing the revision and addressing the majority of concerns. The paper has improved a lot.

I still would like to point out issues that would need to be resolved for the camera-ready:
1. You do not seem to discuss the time it takes for the optimization algorithm to compute the optimal replication factor and placement. The complexity analysis was also not convincing, you still do not provide any bounds how far from the optimal solution you can be.
2. Comparison to StreamBox. Adding the comparison in the evaluation section certainly strengthened the paper -- but the reported behavior in your experiments (esp. with respect to scalability beyond a single socket) does not match the experiment evaluation of the original StreamBox paper. Please address the inconsistency!
3. You often reference the work [26] as one of the close related approaches. I am afraid in multiple occasions you are dismissing it wrongly -- that paper does not use cardinality estimation (!) and in fact uses profiling similar to your model instantiation on page 5.
4. Please add a short paragraph discussing the limitations and constraints of your approach.

11. Recommended decision for the revised paper (if appropriate)
Accept

Reviewer #3

Questions

1. Overall Evaluation
Weak Accept

2. Reviewer's confidence
Some Familiarity

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The paper presents Brickstream, a NUMA-aware stream processing system that is targeted at scale-up shared-memory multisocket servers. The authors investigate the operator placement and operator scaling problems for deploying a plan on a multicore and propose a new optimization technique called Relative Location Aware Scheduling that tries to actively minimize the overall impact of remote memory access penalty in modern NUMA servers with the goal of improving overall throughput.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. The paper is well motivated and focuses on the challenging problem of deploying plans on modern NUMA servers.
S2. The paper is well written and easy to read.
S3. A full system implementation on top of Storm shows that existing streaming platforms can be modified to also scale up in addition to scaling out.

7. List 3 or more weak points, labelled W1, W2, W3, etc.
W1. Lack of comparison with other optimized scale-up systems. The only competors are Storm and Flink which are not really built for scaling up on multicores. A comparison with StreamBox would have be very useful.

W2. More targeted evaluation of RLAS with more sophisticated alternatives in addition to just FF and round robin.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
D1. In general, the paper is very well written. The fact that the authors chose to optimize Storm and build the whole system on it is much appreciated. However, this reviewer is not entirely convinced about the key argument behind the central contribution of this work (RLAS).

D2. Prior studies have looked at the problem of deploying plans on multicores like Giceva et al pointed to by authors. Given the similarity between relational query plans and stream processing plans, it should be possible to use similar techniques here. The authors of BriskStream argue that such is not possible due to one important reason: processing capability of each operator varies depending on remote memory access. So they argue that it is necessary to accurately measure remote access latency and combine scaling with placement. However, there is no empirical evidence that such a technique is the ideal one in the paper. There is no evaluation of the impact of the three heuristics, the degree of parallelism choice made by the scaling algorithm, a comparison a baseline RLAS that does not use remote access latency, or with the approach presented by Giceva et al as applied to stream processing. The evaluation of RLAS in Section 5.4 cursorily compares RLAS placement with OS, FF, and RR. Even OS, which is completely agnostic to the application, seems to be as good as, or better than FF and RR. While this clearly shows that FF and RR are bad, this doesn't necessarily prove that RLAS is optimal.

D3. In relational plan deployment, query parallelism is decoupled from placement. Parallelism is often determined by the optimizer in a rewrite pass. Once determined, placement algorithm then determines the right location for each parallel replica. In BrickStream's case, as far as this reviewer observed, identification of "bottleneck" operators can be done during preanalysis. Thus, the degree of parallelism for such operators can be done separately before placement similar to database engines. The authors argue that each such parallel DAG would result in operators having a different cost and so scaling and placement should be combined. But no empirical evidence is offered. Can the scaling algorithm is used separately to deteermine degree of parallelism? In such a case, only the placement would have to deal with remote access latency.

D4. Instead of placement dealing with remote access latency via a cost model, an alternative to this is the

Morsels approach from Hyper, where a thread pool is used to adaptively decide units of work. Streambox uses intel TBB with buffer annotation to ensure that data from a producer is passed to a consumer in the same node. So they take a different approach to parallelizing the plan and support NUMA awareness without explicitly using a cost model that considers remote memory access. There is no evaluation to show why the suggested approach of combining scaling and placement is better than such an alternative. If what the authors claim is the case, stream processing that are optimized for scale-up multicores like Streambox that do not explicitly address remote access latency should suffer from poor scalability. This is not demonstrated. Instead, they show that Storm and Flink, two systems that are optimized for scale out, do not scale well on multisockets.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
Yes

10. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc.
R1. Expand section 5.4 by adding a better alternative to RLAS in addition to OS, FF, and RR. At the very least, one could consider removing the remote memory access latency component from the cost model. If so, what performance does RLAS provide without it? What impact do the heuristics have?

R2. Present a comparison with StreamBox or other alternatives that are optimized for shared memory multicores. Demonstrate that these systems suffer from NUMA overhead due to remote memory access.

11. Comments on the revised paper (if appropriate)
The reviewer appreciates the authors effort in revising the paper. An evaluation of the RLAS algorithm with an alternative that does not include remote cost is a welcome addition.

The paper also includes a comparison with StreamBox. But the results seem to contradict the original StreamBox work. The original StreamBox paper demonstrated good scalability for word count on a 4-socket, 56-core server. However, in this work, the default StreamBox barely scales. The authors describe that they modified Streambox to disable ordering overhead and the resulting system scales well only within a socket. Even with their change, StreamBox fails to scale across sockets for WC. A reduced throughput due to remote access latency is understandable, but a completely lack of scalability indicates a deep rooted problem with thread synchronization that limits scalability across sockets.

The authors suggest centralized task scheduling to be one such source of synchronization. But they do not clearly quantify the overhead. Thus, it is not clear as to what extent the NUMA impact of remote data access contributes to the projected performance loss as compared to thread synchronization. Any attempt at quantifying this better would make the insights more useful. The reviewer also strongly recommends the authors to add to the final version of the paper a comparison with more recent, scalable stream processing systems, like Trill or SStore that does not suffer from this overhead.

12. Recommended decision for the revised paper (if appropriate)
Accept

Reviewer #4

Questions

1. Overall Evaluation
Weak Reject

2. Reviewer's confidence
Expert

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)

This paper presents BrinkStream, which is designed for stream processing on shared-memory multicore systems. The authors first identify that the allocation of operators in multicore systems can affect system throughput because of the NUMA effect. Therefore, the authors propose a rate-based optimization method to produce an allocation plan to maximize system throughput. Some implementation improvements are also proposed, including reducing instruction footprint and combining tuples into a JumboTuple. The experiments show that these techniques can achieve a better system throughput in comparing to existing open-source distribute stream processing systems and simple operator allocation plans.

6. List 3 or more strong points, labelled S1, S2, S3, etc.

S1. Improving system throughput of stream processing in multi-core system is a very relevant problem.
S2. The paper verifies that operator allocation can affect system throughput because of the NUMA effect.
S3. With an implementation of proposed techniques over the Storm codebase, the authors run an extensive experimental study that verifies the advantage of the proposed improvement.

7. List 3 or more weak points, labelled W1, W2, W3, etc.

W1. The operator allocation problem is not really defined as to maximize the system throughput as claimed in the paper.
W2. The paper completely ignores a common stream processing metric: latency.
W3. The presentation is a bit imprecise in various places. The experimental configuration is vaguely presented.
W4. Only compared with naive alternatives.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)

D1. The operator allocation problem is defined to maximize the expected output rate. The expected output rate of an operator is equal to the sum of the input rate if it is under supplied. This means that if the system is under-supplied (when statistics are being collected), the optimizer would only look for a feasible solution that fulfills the constraints without actually optimizing the system throughput. This is problematic. Because in a DSP, stream rates can vary over time. How BrinkStream can use the right parameter to optimize operator allocation in practical scenarios is unclear. BrinkStream does not perform load balancing. Hence if the operator allocation is based on statistics with under-supplied situations, the system could suffer from severe throughput problems in comparing to a balanced allocation if the input rates increase significantly during runtime.

In the experiments, only the cases with over-supplied situations are used. This further strengthens my concern.

D2. The paper completely ignore latency, a usual metric for DSP systems. One may argue that latency may not be a significant issue in some DSP application, which I completely agree. However, as a scientific study, it would be nice to understand the latency performance of BrinkStream. What the trade-offs are and what can be improved for designing a system for applications where latency is critical.

D3. The authors only compare LRAS with some naive allocation algorithms. There are plenty algorithms for operator allocation in distributed DSP systems that attempts to collocate operators, e.g. "Rohit Khandekar, Kirsten Hildrum, Sujay Parekh, Deepak Rajan, Joel L. Wolf, Kun-Lung Wu, Henrique Andrade, Bugra Gedik: COLA: Optimizing Stream Processing Applications via Graph Partitioning. Middleware 2009: 308-327". Why not compare with them? You can argue that the objectives in those methods are different. But this is also true for the naive algorithms, which do not specifically optimize the expected output rate.

D4. In Fig 15, it seems the benefit brought by RLAS is not particularly high. But it is acceptable. Why only reporting the figures for LR? It seems BrinkStream can achieve a higher overall performance improvement on the other applications. It would be nice to see the breakdown of the figures for the other applications too. There is still plenty of space for the appendix to add more figures.

D5. Some details of the experimental configuration are missing. For example, what are the input rates of the different streams in LR? It could be hard to repeat the experiments. The reported throughput in many experiments are normalized and hence it is even impossible to reverse engineer the input rates.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
No

10. Comments on the revised paper (if appropriate)
I appreciate the authors' efforts of revising the paper. The paper is indeed improved in some aspects.

However, I am quite disappointed that some of my major concerns are not addressed sufficiently: 1) the assumption and objective of the optimization algorithm, 2) latency.

I don't see how the over-supplied assumption can be used in practice. The proposed algorithm can only be used if the system is under a constantly over-supplied workload, which is often not true. A more usual scenario is that the average workload is lower than the system capacity, but the instant workload can fluctuate and be very high. Since the input contains multiple streams with different temporal characteristics, one cannot model the instant workload accurately. Therefore, most existing approaches maximize throughput by load balancing and cost minimization. A load balancing operator allocation can better tolerate load fluctuations. I don't think the authors' arguments and descriptions are sufficient to justify their significantly different approach.

The latency experiment is also not very insightful, partly due to their assumption of being over-supplied. It seems that the high latency numbers of Flink and Storm are mainly due to the high queueing latency caused by overloading. The experiments do not really compare the processing latency when all the systems are sufficiently provisioned, which is much more interesting than queuing latency.

11. Recommended decision for the revised paper (if appropriate)
Reject

Reviewer #5

Questions

1. Overall Evaluation
Weak Reject

2. Reviewer's confidence
Knowledgeable

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The submission revolves around stream processing systems, and how such a system can be deployed efficiently on a shared-memory multicore machine. In a nutshell, the authors propose an optimization algorithm that aims to enable NUMA-aware (stream) operator placement. The resulting system, dubbed BriskStream, extends Apache Storm with the aforementioned optimization phase, and refactors its implementation to better fit a single-node scenario.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
S1. Query optimization for stream processing systems is an oft-overlooked, yet important research area.
S2. Principled approach towards an optimization algorithm.
S3. Elaborate discussion of a cost model that can be used in stream processing apps, and honest presentation of its current shortcomings in terms of hardware modeling.

7. List 3 or more weak points, labelled W1, W2, W3, etc.
W1. Unclear why the discussed problem is a stream-specific problem, instead of a query-processing-on-multicores one.
W2. Experimental evaluation omits systems optimized for single-node performance, contains few workloads,

and lacks details.
W3. System implementation section proposes already established techniques.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
D1. Discussion of the "three heuristics" is clearer in the intro section that in Section 3. Consider rewriting the latter part for clarity.
D2. Unclear why the optimization phases proposed by the authors are stream-processing-specific. The two main concerns, namely i) operator placement and ii) operator replication factor seem directly mappable to NUMA-aware analytical processing and determining the degree of parallelism for each operator; there are numerous research papers addressing those concerns, some of them being data morsels from Leis et al., and NUMA-aware data and task placement by Psaroudakis et al. among others.
D3. The authors mention in passing in Appendix B.2 that operator fusion can impact their design. Pipelined query processing needs to be explicitly mentioned in the main part of the paper, and the authors need to explain whether their contributions are still applicable in the cases where maximizing pipelining in the query plan relaxes resource constraints on the NUMA machine.
D4. The optimization process that the authors propose comprises two phases, repeated iteratively. Does keeping the two phases separate mean that some potentially optimal configurations will not be considered?
D5. Figures 4 and 6 should ideally be placed on the page that contains their text description. Also, it is unclear to me what the dotted arrows of Figure 4 refer to.
D6. Is there a specific reason that the authors opted to extend Storm instead of Heron, which arguably is the evolved version of Storm?
D7. The optimizations mentioned in Sections 4.1 and 4.2 are straightforward code refactorings that one would perform to turn a distributed system into a single-node solution, and in some cases explicitly address current shortcomings of Storm. In addition, Figure 12 indicates that the gains of BriskStream stem, to a great extent, from these optimizations.
D8. The description of the experimental setup is rather limited, and points to the appendix and to a previous paper for more details. The main body of the current paper needs to be self-contained.
D9. The queries that the authors use are rather limited. Worse, most of the plots end up revolving around two queries: 'WC' or 'LR'.
D10. Figure 7 offers little insight, given that it revolves around (only) 20 random plans that were actually randomly generated. Given the number of operators in the LR query, the choice of 20 plans seems arbitrary and small, while the reader is also unable to figure out why the plan chosen by BriskStream is of better quality.
D11. Through the entire paper, the authors offer little comparison to existing stream processing systems that target single-box deployment (e.g., Trill), as well as streaming systems that evolved from OLTP solutions, such as S-Stream, and thus are prone to be very CPU-efficient.
D12. Likewise, the evaluation section compares BriskStream against essentially distributed systems, which naturally pay a
"(de)serialization and distributed execution tax".
D13. Should Figure 12 have bars for Storm that also include RMA?
D14. In Section 6, the authors indicate that "the pipeline execution model of DSP systems requires different cost models". Why is that, and what would be the difference when comparing the DSP paradigm to i) operator-at-a-time OLAP (i.e., a la Vectorwise / C-Store), and to ii) pipelined OLAP (i.e., a la HyPer)?

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
Yes

10. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc.
Address weak points and detailed evaluation comments.

11. Comments on the revised paper (if appropriate)
The authors placed significant effort in revising the paper.
My comments were, to a sufficient extent, addressed.
Still, I would have liked to see a less hand-waiving response to C5-3 & C5-10: My comment concerned the fact that Section 4 mainly discusses implementation issues of Storm, careful zero-copy data movement, etc.
Finally, I still believe that a comparison with Trill / S-Store would have been more appropriate to highlight pros and cons of the authors' approach.

12. Recommended decision for the revised paper (if appropriate)

Accept