Plumb: Efficient Processing of Multi-User Pipelines

Abdul Qadeer and John Heidemann USC/ISI {aqadeer, johnh}@isi.edu

As the field of big data analytics matures, workflows are increasingly complex and often include components that are shared by different users. Individual workflows often include multiple stages, and when groups build on each other's work it is easy to lose track of computation that may be shared across different groups.

We propose *Plumb*, a workflow system for multi-stage workflow where parts of computation and output are shared across different groups. Plumb focuses on blocked, streaming workflows, a middle ground between large-file batch processing and small-record streaming. A particular challenge with this problem domain is structural and computational skew, where the computation of different stages and of different blocks in a stage can vary by a factor of ten due to differences in the work or data.

Plumb's first goal is to identify duplication of computation and storage that can occur when different groups share components of a pipeline. When different users are responsible for different portions of the workflow, work done in common stages will be duplicated if each user assumes they begin with raw input, particularly as the workflow evolves over the course of development. This problem of computational sharing was recently recognized at Microsoft [2]; we identify duplication both in computation and in storage of intermediate results. While databases sometimes save and share intermediate results, automated discovery is more challenging in today's loosely structured big-data workflows, where processing modules are largely opaque to the system.

The second problem we take on is *skew*. Prior work has identified data skew, where many data items fall into one processing bin, slowing the overall workflow [1]. We identify and address two new types of skew: computational skew and structural skew. Computational skew occurs when a bin of data takes extra long to process, not necessarily because there is more data, but because the data interacts with the processing algorithms to take extra time. Structural skew occurs when one stage of the processing pipeline is noticeably slower than other stages.

We address both of these types of skew in Plumb by scheduling additional processing elements when one data block or one stage falls behind. Plumb decouples processing for each stage of the workflow, buffering output when required and allowing each stage to be scheduled independently. However, to avoid the cost of data buffering, Plumb also allows stages to run concurrently when they are well matched. This decoupling also addresses computational skew, since additional computation can be brought to bear when specific data inputs take extra time.

Plumb is designed for *large-block, streaming* workloads. Traditional map-reduce has focused on batch processing, and systems such as Spark [3] consider streams of small records. We have identified a class of applications that involve long-term streams of data, but where the processing requires examination of large blocks of data (say, 10 to 1000 megabytes of data at a time) to capture temporal or spatial locality, to integrate with existing tools, and to support fault tolerance and recovery in long-running data processing. Applications that require large-block data preclude use of adaptive sharding schemes to present skew. We have designed Plumb to support large-block, streaming workloads and exploit this "middle ground" where per-data scheduling is possible.

In this poster we propose *Plumb*, a framework for processing of a multi-stage pipeline. Plumb integrates pipelines contributed by multiple users, detecting and eliminating duplication of computation and intermediate storage. It tracks and adjusts computation of each stage, creating more processing instances as required to accommodate both structural and computational skew. Plumb also tracks I/O-boundness of each stage and generate alerts for users for possible merging of I/O bound stage to a CPU-bound stage. Plumb currently uses named large size files as a proxy for large-blocks.

We exercise Plumb with the processing pipeline for B-Root DNS traffic. Compared to the currently operational, hand tuned system, we expect Plumb to provide one-third the latency while utilizing 22% less CPU. Moreover, the Plumb abstractions enable multiple users to contribute to processing with minimal coordination, and it keep latency low during normal conditions, while adapting to cope with dramatic changes to traffic and processing requirements when handling denial-of-service attacks.

The contribution of this poster is to provide an organization-wide processing substrate Plumb that can be used to solve commonly occurring problems and to achieve a common goal. Plumb makes multi-user sharing a first-class concern by providing pipeline-graph abstraction. This abstraction is simple and based on fundamental model of input-processing-output but is powerful to capture processing and data duplication. Plumb then employs best available solutions to tackle problems of large-block processing under structural and computational skew without user intervention.

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-18-2-0280. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

SoCC '18, October 11–13, 2018, Carlsbad, CA, USA

ACM ISBN 978-1-4503-6011-1/18/10.
https://doi.org/10.1145/3267809.3275461

REFERENCES

[1] Emilio Coppa and Irene Finocchi. 2015. On Data Skewness, Stragglers, and MapReduce Progress Indicators. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*. ACM, New York, NY, USA, 139–152. https://doi.org/10.1145/2806777.2806843

[2] Alekh Jindal, Shi Qiao, Hiren Patel, Zhicheng Yin, Jieming Di, Malay Bag, Yifung Lin Marc Friedman, and Sriram Rao Konstantinos Karanasos. 2018. Computation Reuse in Analytics Job Service at Microsoft. In *Proc. of ACM SIGMOD International Conference on Management of Data*. ACM, Houston, TX, USA. https://doi.org/10.1145/3183713.3190656

[3] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. 2013. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 423–438.