Partition Balancing in Distributed Nested Data-Parallel Systems

Frank Austin Nothaft
Department of Electrical Engineering and Computer Science, University of California, Berkeley
fnothaft@eecs.berkeley.edu

Michael Linderman
Carl Icahn School of Medicine at Mount Sinai
michael.linderman@mssm.edu

Abstract

Map-reduce frameworks such as Apache Hadoop and Spark provide the abstraction of a large, flat array that is processed in parallel across many machines. While this simple programming model has enabled the broad adoption of data-parallel distributed frameworks, these systems cannot express irregular parallel computation, and their performance is impacted by data imbalance across nodes. In this paper, we present a distributed framework for implementing *nested data parallel* (NDP) computation. Unlike previous NDP systems described by Blelloch and Bergstrom et al. that relied on the use of a *flattening* transformation at compile-time, we present a cost model that is used to select between multiple partitioning strategies at run-time. Through this, we provide a user-tunable means for trading node-to-node imbalance versus communication when executing distributed NDP programs.

1. Introduction

The use of map-reduce as a flat data-parallel (FDP) programming model for distributed systems has grown rapidly since it was introduced in Google's seminal 2004 paper [5]. The development of the open-source Apache Hadoop system enabled the use of this programming model outside of Google. Modern map-reduce systems such as Apache Spark [7] have refined the programming model further by reducing the dependency of the framework on disk via in-memory caching. While this refinement has enabled the use of map-reduce for iterative workloads, the programming model remains confined to computation that can be expressed via flat data-parallel operations. Specifically, Spark presents users with the abstraction of a resilient distributed dataset (RDD), which appears as a flat array that is distributed across compute nodes in a cluster [6].

To expand the algorithms that could be expressed as a data-parallel computation, Blelloch introduced the nested data-parallel (NDP) model [3]. In this programming model, users are provided the abstraction of an array whose elements are a nested level of arrays and data-parallel operators are applied on the nested arrays. A primary complication in the implementation of the NDP model is that the nested arrays frequently do not have uniform size. Several approaches have been suggested for balancing work in NDP programs, including the compile-time vectorization of NDP programs for execution on single-instruction multiple-data (SIMD) machines [3, 4] and flattening for multiple-instruction multiple-data (MIMD) machines [1]. Additionally, dynamic work-stealing approaches have been implemented [2].

In this paper, we introduce a distributed programming framework for NDP algorithms. Our implementation is built on top of Apache Spark. We track the structure of the nested arrays at runtime and choose between two strategies (*uniform* and *segmented*) for partitioning data across machines based on the estimated cost of each strategy. In the segmented partitioning strategy, all values in a single nested segment are co-located on a single compute node, and most computation can proceed without communication. The uniform strategy provides perfect load balance across all nodes, but many operations will need to communicate to execute. To choose between these strategies at runtime, we provide a cost model that evaluates the performance tradeoff of communication overhead versus node-to-node imbalance. Since the partitioning strategy is chosen at runtime, we can leverage knowledge of the nested array structure.

2. Background

2.1 Nested Array Programming Models

2.2 Data Parallel Distributed Computing Models

3. Characterizing and Modeling Imbalance in Nested Arrays

4. Implementation

5. Performance

6. Discussion

7. Conclusion

References

[1] Bergstrom, L., Fluet, M., Rainey, M., Reppy, J., Rosen, S., and Shaw, A. Data-only flattening for nested data parallelism. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '13)* (2013), ACM, pp. 81–92.

[2] Bergstrom, L., Rainey, M., Reppy, J., Shaw, A., and Fluet, M. Lazy tree splitting. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming (ICFP '10)* (2010), ACM, pp. 93–104.

[3] Blelloch, G. E. *Vector models for data-parallel computing*, vol. 356. MIT press Cambridge, 1990.

[4] Blelloch, G. E., and Sabot, G. W. Compiling collection-oriented languages onto massively parallel computers. *Journal of Parallel and Distributed Computing 8*, 2 (1990), 119–134.

[5] Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Sym-*

posium on Operating Systems Design and Implementation (OSDI '04) (2004), USENIX Association, pp. 10–10.

[6] ZAHARIA, M., CHOWDHURY, M., DAS, T., DAVE, A., MA, J., MCCAULEY, M., FRANKLIN, M. J., SHENKER, S., AND STOICA, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI '12)* (2012), USENIX Association, pp. 2–2.

[7] ZAHARIA, M., CHOWDHURY, M., FRANKLIN, M. J., SHENKER, S., AND STOICA, I. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud '10)* (2010), p. 10.