Task Fusion: Improving Utilization of Multi-user Clusters

Robert Dyer
Iowa State University
rdyer@iastate.edu

Abstract
Researchers use shared computing clusters to ask interesting questions and wish to maximize their utilization. Currently, optimizations focus on individual programs. We present task fusion to automatically merge multiple tasks into a single task. An example implementation shows fused tasks take 14–90% less time than running the tasks individually.

Categories and Subject Descriptors D.1.3 [PROGRAMMING TECHNIQUES]: Concurrent Programming

Keywords MapReduce; optimization; task fusion

1. Introduction
Big data techniques allow researchers to propose and answer many interesting hypotheses on very large sets of data. Analyzing big data is accomplished using a distributed cluster, which is typically shared by several users. Increasing the utilization of such shared resources is important in order to keep costs low and allow for more users.

Many techniques have previously been proposed to optimize programs that run on such clusters, however these optimizations typically focus on single programs. Assuming we have already used existing single-program optimization techniques to increase utilization of shared clusters, important research questions we wish to answer are: Can we further increase utilization by automatically optimizing groups of programs? If so, what pre-conditions must hold to apply such cross-program optimizations?

In this research we answer these questions and propose a new optimization technique called task fusion, which is inspired by a compiler optimization called loop fusion that increases data locality by fusing multiple loops together. Task fusion takes multiple, individual tasks and fuses them into a single task. This process creates a single program that reads the input data only once and performs multiple computations on it, thus avoiding the overhead of reading the data multiple times.

We prove the feasibility of our optimization by implementing it in the Boa framework for software repository mining [4, 6]. Our early evaluation shows that task fusion in Boa provides performance benefits of up to ten times faster execution when compared to running tasks individually.

2. Background and Related Works
Dean and Ghemawat described a MapReduce framework [3] for distributed computations. In this framework, data is represented as key/value pairs. Each pair is given to a map (or mapper) function, which transforms it into zero or more output key/value pairs. This output is then fed into a second user-defined function called a reducer, which aggregates all values for a given key and produces a final result. The framework automatically sorts the data between the map and reduce phases. Hadoop [1] is an open-source implementation of MapReduce written in Java.

Chain folding and job merging [5] are two optimizations for MapReduce programs. Chain folding aims to take multiple map or reduce steps in a single program and fold them together into fewer steps. Job merging is when two un-related programs that operate over the same input are manually merged into a single program by merging the mappers and reducers together.

FlumeJava [2] is a library for Java that abstracts the details of MapReduce from the user, instead providing a set of high-level features such as parallel collections and functions. The library generates a query plan which may include multiple MapReduce phases. This optimization is for a single FlumeJava program.

3. Task Fusion
The previously mentioned optimizations typically focus on individual programs. The only optimization that looks at multiple programs, job merging, assumes the programs are all by the same author and requires manually merging them together. In this section we propose a fully automated solu-
The first column shows the time (in seconds) to run the tasks in a given workload individually. The next column shows the execution time when the tasks are fused together. The last column shows the speedup achieved by task fusion.

In the best case, task fusion is over ten times faster than running the tasks individually. Even in the worst case, task fusion runs in 14% less time than individually. These early results clearly demonstrate the potential benefit of using task fusion in shared clusters.

5. Conclusions and Future Work

Maximizing the utilization of shared clusters is important for researchers interested in answering questions with big data techniques. Task fusion is an optimization that allows multiple programs from different users to be fused and executed together, requiring 14%–90% less time than executing the tasks individually.

In the future we plan to investigate ways to relax some of the assumptions currently required for task fusion, allowing application of the technique to more infrastructures.

Acknowledgments

This work funded in part by NSF grants CCF-10-17334 and CCF-11-17937.

References


