Parallel Computing 51 (2016) 67–78

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Local search to improve coordinate-based task mapping

Evan Balzuweit [a], David P. Bunde [b,*], Vitus J. Leung [c], Austin Finley [b], Alan C.S. Lee [b]

[a] Department of Computer Science & Engineering, Washington University in St. Louis, St Louis, MO, USA
[b] Department of Computer Science, Knox College, Galesburg, IL, USA
[c] Sandia National Laboratories, Albuquerque, NM, USA

ARTICLE INFO

Article history:
Available online 31 October 2015

Keywords:
Task mapping
Stencil communication pattern
Non-contiguous allocation
Local search

ABSTRACT

We present a local search strategy to improve the coordinate-based mapping of a parallel job's tasks to the MPI ranks of its parallel allocation in order to reduce network congestion and the job's communication time. The goal is to reduce the number of network hops between communicating pairs of ranks. Our target is applications with a nearest-neighbor stencil communication pattern running on mesh systems with non-contiguous processor allocation, such as Cray XE and XK Systems. Using the miniGhost mini-app, which models the shock physics application CTH, we demonstrate that our strategy reduces application running time while also reducing the runtime variability. We further show that mapping quality can vary based on the selected allocation algorithm, even between allocation algorithms of similar apparent quality.

1. Introduction

The resource management pipeline for parallel batch jobs has three stages, scheduling, allocation, and task mapping. Scheduling decides when a job will run, allocation determines which nodes it is assigned, and task mapping matches the job's tasks to individual computational elements (nodes or cores). Scheduling and allocation are typically done at the system level, while task mapping is left to the application. In MPI programs, task mapping is the decision of which MPI rank performs each part of the computation, even if done in a "default" manner.

This paper focuses on improving task mapping, which has been shown to significantly reduce job running times for a variety of scientific applications (e.g. [1–5]), including achieving a speedup of 1.64 on a quantum system simulation [3]. Better task mapping reduces the number of hops between communicating tasks and hence the amount of system bandwidth consumed by each message. As systems grow larger and processor performance continues to improve faster than network performance, its importance will continue to grow.

In addition to performance, we consider the effect of task mapping on run-time predictability, which has recently become the subject of active research (e.g. [6]). Predictability is desirable in HPC systems because users must estimate job running time for the scheduler.

The job's communication pattern and the system's network topology are both important for task mapping. Here, we focus on jobs communicating in a nearest-neighbor stencil pattern, a very common pattern in computational science applications. In this pattern, the tasks correspond to integer points in a grid and communicate with their nearest neighbors, the 4 closest points in

* Corresponding author. Tel.: +1 3093417479; fax.: +1 3093417601.
E-mail addresses: ebalzuwe@knox.edu (E. Balzuweit), dbunde@knox.edu (D.P. Bunde), vjleung@sandia.gov (V.J. Leung), afinley@knox.edu (A. Finley), cslee@knox.edu (A.C.S. Lee).

http://dx.doi.org/10.1016/j.parco.2015.10.012
0167-8191/

the $+x$, $-x$, $+y$, and $-y$ directions for 2D or the 6 closest points for 3D (add the $+z$ and $-z$ directions). This pattern arises naturally from spatial decompositions into rectangular[1] regions.

In this paper, we target machines whose network topology is a 3D mesh and allow the possibility that a job is allocated to a non-contiguous set of nodes. This is appropriate for the Cray XT, XE, and XK series of systems, including the Cray XE6 (Cielo) we use for our experiments. Two kinds of allocators, linear and center-based, have been studied for this type of machine. We study the interaction of our task mapping algorithms with both types of allocators using experiments (Cray uses a linear allocator) and simulations.

Although jobs are allocated nodes on our target machines, our task mapping algorithms work in terms of MPI ranks rather than compute nodes allocated to that job. Each MPI rank is a single process in a distributed memory program; we will refer to them simply as ranks. In general, each compute node may support many ranks depending on its number of cores and the mix of distributed- and shared-memory programming models (e.g. MPI and OpenMP) used.

A recent task mapping algorithm developed for our setting is Geometric (Geom) [7], which operates by finding corresponding decompositions of the job tasks and allocated ranks. This algorithm was shown to outperform a wide variety of other algorithms, reducing application running time by around 30% [7,8].

Contribution. The main contributions of this paper are as follows:

- We present a new algorithm, Geom-Based Local Search (GSearch), that tries to improve on Geom using the observation (e.g. [7–10]) that job running time correlates with the average number of hops between communicating tasks (*average hops metric*). Specifically, GSearch uses a local search to improve the average hops metric by swapping pairs of tasks when doing so improves the average hops metric.
- We demonstrate our algorithm in a proxy application and show that it improves total application running time. Furthermore, it does so while reducing the running time variability.
- We show that the benefit of GSearch depends on the algorithm used to allocate nodes to jobs. While the benefit is fairly modest with linear allocation algorithms, it is much greater with a center-based algorithm. This is the first time such a dependence has been seen.
- We examine the number of swaps made by GSearch, showing it is reasonable in practice. We also show that it can be large, but suggest using a cutoff to avoid pathological cases.

At a high level, our results again demonstrate that Geom is a good task mapping algorithm for practical settings, but show that local search (represented by GSearch) can improve upon it.

This paper builds on a preliminary version [11], with the addition of center-based allocation, more analysis of the variability of mapping quality, and improved presentation.

The rest of this paper is organized as follows. Section 2 describes our algorithms. Section 3 describes the setup for our experiments and simulations. Section 4 describes our results. Section 5 summarizes related work. Section 6 concludes and discusses future work.

2. Algorithms

We begin by reviewing allocation algorithms and then showing how GSearch builds on Geom.

2.1. Allocation

To identify the nodes allocated for each job, we use two different allocation algorithms. The first of these is a linear allocation algorithm that combines ideas of Lo et al. [12] and Leung et al. [9]. This algorithm organizes the nodes in a linear order along an "s-curve", which curves back and forth along the machine's shortest dimension (z). The free nodes are grouped into intervals by their position along the curve and the algorithm allocates nodes from the smallest interval with enough nodes (best fit). If no interval is large enough, nodes are chosen to minimize the maximum distance along the curve between chosen nodes. This scheme is fast and generates good allocations [13]. It is also similar to the algorithm used in practice on Cray systems [14].

Our second allocation algorithm is MC1x1 [15], which builds a candidate allocation around each available node. The node being built around is the *center* and its candidate allocation is the closest nodes to the center (measured in hops). The candidate allocation minimizing the sum of pairwise distances is selected. Because of its selection criteria, in some sense MC1x1 is optimizing all-to-all communication, essentially being pessimistic in the absence of job-specific information. MC1x1 is based on a similar algorithm MC [16] and several variations have been studied in the literature (e.g. [17,18]). It is not used in practice, but represents an approach which avoids the 1-dimensional reduction used by the linear allocation algorithms.

2.2. Task mapping

We now describe our task mapping algorithms. Geom first rotates the job so that its dimension lengths have the same order as the bounding box of the ranks (i.e. if the bounding box of ranks is largest in the x dimension, then the job's largest dimension will also be x and so on).

[1] We use "rectangular" to include rectangular prisms in 3D.