Future Generation Computer Systems 30 (2014) 191–201

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A topology-aware load balancing algorithm for clustered hierarchical multi-core machines

Laércio L. Pilla[a,b,*], Christiane P. Ribeiro[b], Pierre Coucheney[b], François Broquedis[b], Bruno Gaujal[b], Philippe O.A. Navaux[a], Jean-François Méhaut[b]

[a] Instituto de Informática – Universidade Federal do Rio Grande do Sul, Avenida Bento Gonçalves 9500. 91501-970, Porto Alegre, Brazil

[b] Université de Grenoble – Laboratoire d'Informatique de Grenoble – UJF – CNRS – INRIA – INP – CEA, 51 avenue Jean Kuntzmann, 38330, Montbonnot-Saint-Martin, France

HIGHLIGHTS

- We propose a topology-aware load balancing algorithm for multi-core machines.
- The algorithm is demonstrated to converge asymptotically to the optimal solution.
- We model distances among hardware components in terms of latency and bandwidth.
- Topology-aware load balancing algorithm shows scalable performance over 256 cores.

ARTICLE INFO

Article history:
Received 30 December 2012
Received in revised form
16 May 2013
Accepted 17 June 2013
Available online 3 July 2013

Keywords:
Load balancing
Hierarchical architectures
Hardware topology
Proof of optimality
Benchmarking

ABSTRACT

In this paper, we present a topology-aware load balancing algorithm for parallel multi-core machines and its proof of asymptotic convergence to an optimal solution. The algorithm, named HwTopoLB, aims to improve the application performance by reducing core idleness and communication delays. HwTopoLB was designed taking into account the properties of current parallel systems composed of multi-core compute nodes, namely their network interconnection, and their complex and hierarchical core topology. The latter comprises multiple levels of cache, and a memory subsystem with NUMA design. These systems provide high processing power at the expense of asymmetric communication costs, which can hamper the performance of parallel applications depending on their communication patterns if ignored. Our load balancing algorithm models asymmetries in terms of latencies and bandwidths, representing the distances and communication costs among hardware components. We have implemented HwTopoLB using the Charm++ Parallel Runtime System and evaluated its performance with two different benchmarks and one application. Our experimental results with HwTopoLB exhibit scalability over clustered multi-core compute nodes, and average performance improvements of 23% over execution without load balancers and 19% over the existing load balancing strategies on different multi-core systems.

1. Introduction

Parallel systems composed of multi-core compute nodes require the use of parallel programming languages and environments to explore their processing power. Such programming languages provide a way to decompose an application into tasks, threads, and processes, and to distribute them over available cores. Nevertheless, an initial work distribution may not achieve the expected performance and efficiency due to applications' dynamic and irregular characteristics.

In these scientific applications, the load of each task and their number may depend on input data. Their load may also vary between timesteps. Examples are: molecular dynamics simulations, where atoms move in space through simulated timesteps; and weather forecasting models, where incidence of rain can increase the computational load of some tasks. The resulting load imbalance leads to sub-optimal performance.

Load balancing strategies can be used during the application execution to improve the distribution of tasks. Such strategies try to find a new task distribution that maximizes core usage by taking into account the tasks' execution times. Still, this work distribution

* Corresponding author at: Instituto de Informática – Universidade Federal do Rio Grande do Sul, Avenida Bento Gonçalves 9500. 91501-970, Porto Alegre, Brazil. Tel.: +55 5499736313.
E-mail addresses: llpilla@inf.ufrgs.br, pilla@imag.fr (L.L. Pilla).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.06.023

may not provide optimal performance due to communication overheads from the design of current multi-core systems.

The increasing number of cores in parallel compute nodes demands an efficient memory and cache hierarchy design, as multiple cores compete for resources to access shared memory. In this context, current architectures use multiple levels of shared cache memories and a non-uniform memory access (NUMA) design. This introduces a complex topology and hierarchical memory sub-system with asymmetric latencies and bandwidths. These affect application communication and its performance [1]. Additionally, the network interconnection presents its own asymmetries. Therefore, to attain scalable performance on parallel systems, it becomes necessary to take into account the communication costs between hardware components when load balancing.

In this scenario, we propose a novel load balancing approach for parallel multi-core systems. We combine information about the machine topology with statistics about the application (*i.e.* workload and communication patterns) to ensure that no core will be underutilized due to unbalance of the tasks being executed. Moreover, this approach is used to reduce the costs of communication through memory and network. Based on this approach, we designed a centralized topology-aware load balancing algorithm named Hardware Topology Load Balancer, or HwTopoLB. This algorithm is proven to asymptotically converge to an optimal mapping, as we demonstrate in Theorem 1. HwTopoLB was first presented in our previous work [2], where its machine topology model was limited to shared memory machines and memory latency only, and its performance evaluation was based only on the benchmarks. The main contributions of the current paper are as follows:

- we propose a machine topology model for clustered multi-core compute nodes that represents communication costs as latencies and bandwidths.
- we present in details a load balancing algorithm that combines our machine topology model with profiled information about the application.
- we provide the proof of its asymptotic convergence to the optimal solution.
- we evaluate its effectiveness with benchmarks and one application over different parallel machines. Its performance and scalability are compared to other load balancers.

We implemented HwTopoLB using the Charm++ parallel runtime system for this evaluation. Charm++ provides a load balancing framework with statistics of the application profiled during runtime [3,4]. The experiments show that the proposed load balancing algorithm achieves average performance improvements of 23% over the execution without load balancers and of 19% over other load balancers. HwTopoLB also exhibits a better scalability on experiments with up to 256 cores distributed on 8 interconnected multi-core compute nodes.

The remainder of this paper is organized as follows: Section 2 discusses related work. We describe the proposed load balancing algorithm in Section 3. Its implementation is discussed in Section 4. In Section 5, we outline the experimental methodology used in the evaluation presented in Section 6. Concluding remarks and future work are presented in Section 7.

2. Related work

The hierarchical design of current parallel systems and the complexity of parallel applications have demanded efficient task mapping algorithms. For that purpose, extensive research has been done on schedulers, load balancers, and work-stealing algorithms. Efforts usually focus on one of two contexts: (i) among compute nodes, or (ii) inside multi-core compute nodes. We split the state of the art in these two categories.

2.1. Network topology optimizations

Research in a network level focuses on reducing network contention, distributing work among compute nodes (but not directly among their cores), and avoiding task migration overheads by moving tasks among nearby compute nodes, where the locality is related to the number of hops between two nodes.

Zheng et al. [4] studied the performance and scalability of different periodic load balancers on Charm++ [3]. The authors presented two greedy algorithms, a refinement-based algorithm named RefineLB, and a two-level hierarchical algorithm. Among them, only one greedy algorithm considers the communication patterns of the application, and none considers the machine topology. Their results show that: (i) centralized algorithms tend to reduce the most the duration of application timesteps; (ii) refinement-based and hierarchical algorithms provide better scalability by reducing load balancing overheads; and (iii) the hierarchical approach significantly reduces the memory footprint of a load balancer.

Lifflander, Krishnamoorthy and Kale [5] demonstrated the use of periodic load balancing on MPI applications. They also applied work stealing [6,7] in the same scenario. The authors developed a threaded active message library over MPI which enables the use of medium grain tasks. They presented two load balancing algorithms: one similar to RefineLB [4], and one hierarchical algorithm. In the latter, cores are organized in a tree and send their smallest tasks to their immediate parents in case of imbalance. Their vision of locality is related to the tree organization only, as it does not consider metrics such as latency and bandwidth. Additionally, both algorithms do not consider the application's communication graph. Their work on distributed work stealing involved stealing tasks from random cores, and keeping one work queue per core. Since the applications considered are iterative, tasks stolen by a core are kept by it in an effort to reduce the migration overhead.

Bhatele et al. [8] researched the influence of topology-aware load balancing algorithms on NAMD [9], a molecular dynamics application. Their study focused on static and dynamic topology-aware mappings on large scale parallel machines with 3D mesh and torus topologies. Their results showed performance improvements of up to 10% on NAMD. The metric used for the evaluation was *hop-bytes*, which is based on the total number of bytes exchanged between processors weighted by the distance between them.

Hoefler and Snir [10] studied the problem of mapping applications with irregular communication patterns to the network topology. They evaluated the effect of different heuristics, and proposed a library to provide automated topology mapping. The communication pattern of the MPI application and the network topology are statically gathered. Topology mapping is done by re-numbering processes in the MPI communicator. Results with the simulation of real parallel systems showed reductions of network congestion by up to 80% and average dilation by up to 50%, while results on 512 compute nodes of a BlueGene/P system showed performance improvements between 10% and 18%. The network bandwidth is considered by their algorithms, but the bandwidth inside one compute one is accounted as unbound.

Catalyurek et al. [11] use hypergraph partitioning to balance the load of dynamic applications using the Zoltan toolkit [12]. They focus on improving communication performance while load balancing by reducing the communication volume among cores and avoiding task migrations. The application is represented as a hypergraph, where vertices are tasks, and nets (hyperedges) represent communication and migrations costs. Communication and migration costs are computed using the amount of bytes a task communicates and contains, respectively. Still, these costs do not take into consideration the machine topology. Their approach involves a graph coarsening phase, a recursive bisection phase,