Initial Results on Hierarchical Topology-aware Load Balancing

Laércio Lima Pilla[1,2], Philippe Olivier Alexandre Navaux[1], Jean-François Méhaut[2]

[1]Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS) – Porto Alegre, Brazil
{laercio.pilla, navaux}@inf.ufrgs.br

[2]LIG Laboratory – CEA / INRIA – Grenoble University – Grenoble, France
jean-francois.mehaut@imag.fr

Abstract

This paper presents initial experiments on the composition of centralized topology-aware load balancing algorithms to form hierarchical ones. This approach tries to benefit from well-tested heuristics while improving their scalability through a hierarchical composition, as the overhead of balancing load in a centralized fashion can overcome its benefit in large scale parallel platforms. Our initial experimental results with a molecular dynamics application running on a parallel platform with 512 processing units indicate performance improvements with hierarchical load balancers of up to 30% over centralized ones.

1. Introduction

High performance computing (HPC) systems composed of multi-core compute nodes have become increasingly hierarchical. This comes from an increase in number of processing units (PUs) per compute nodes that demands an efficient memory and cache hierarchy design, as multiple PUs 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. Additionally, the network interconnection presents its own asymmetries. To attain scalable performance on such parallel systems, in addition to seeking a balanced workload distribution over processing units, taking into account the the communication costs between hardware components becomes a necessity.

In previous work [1, 2], we presented topology-aware load balancing algorithms, named NUCOLB and HWTOPOLB, which try to mitigate load imbalance while improving the communication costs experienced by the application. Both algorithms are centralized, which means that they have a complete view of the application's state running over the whole parallel platform. Although they try to reduce the load balancing overhead by computing their heuristics in polynomial time and avoiding task migrations, their centralized nature can become a hurdle when increasing the scale of both application and machine topology.

In this paper, we discuss a technique to reduce load balancing overheads by composing centralized load balancers in a hierarchical way. This reduction comes from using multiple instances of a load balancer, each limited to a domain of the parallel machine. Task migration between domains in controlled by a centralized load balancer that uses aggregated information in its decisions. We show some initial weak scaling results comparing this technique to the centralized use of HWTOPOLB for a molecular dynamics application in a cluster using up to 512 cores.

This paper is organized as follows. We explain the hierarchical composition of load balancers in Section 2. The experimental setup and results are presented in Section 3. Concluding remarks are discussed in Section 4.

2. Hierarchical LB Composition

Our approach to compose load balancers (LBs) hierarchically using the CHARM++ runtime system is based on the research done by Zheng *et al.* [3]. Load balancers are organized in a n-level tree. Each leaf load balancer aggregates the information from its sub-domain, and communicates it to its parent. This higher level load balancer sees the whole sub-domain as it were only a processing unit. This process is repeated until load balancing information reaches the root of the tree. This LB then computes a new task mapping, and informs its children of the changes in their sub-domains. These children compute their own task mappings, and communicate their decisions to the next tree level. This process is repeated until the leaf load balancers compute their mappings and apply all task migrations.

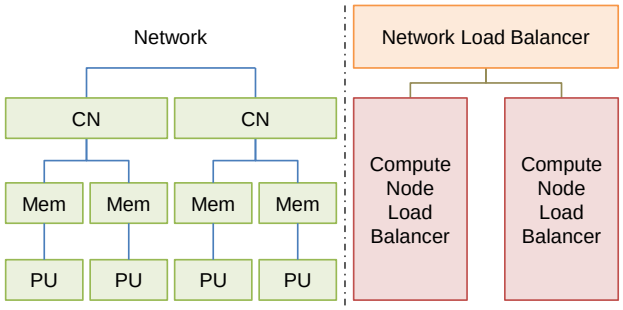

Figure 1. Hierarchical machine topology organization and its equivalent two-level load balancer composition.

We limit our load balancing tree to two levels: a network level and a compute node level. The decision to use two levels is related to results presented by Zheng *et al.*, and to a natural organization of the machine topology in memory and network levels. This hierarchical organization is illustrated in Figure 1, where each compute node (CN) is represented by one load balancer instance.

We used our centralized load balancing algorithms NucoLB and HwTopoLB to generate two hierarchical heuristics. The first one (HwTopo+HwTopo) uses HwTopoLB as both the network and the compute node load balancers, while the second one (HwTopo+Nuco) uses NucoLB as the compute node algorithm. NucoLB was not applied as a network load balancer because its heuristic takes more time to compute than HwTopoLB, which could limit the scalability of the new heuristic. We evaluate the performance of both hierarchical load balancers in the following section.

3. Experimental Evaluation

We assess the performance of our heuristics by evaluating the weak scalability of the molecular dynamics application *LeanMD* on up to 16 Cray XE6 CNs. Each compute node is composed of 32 processing units (PUs). The application runs for 300 iterations, and load balancing calls are done after iterations 40, 140, and 240. *LeanMD*'s cell dimensions start as $8 \times 11 \times 5$ for 2 CNs, and go up to $64 \times 11 \times 5$ for 16 CNs.

The total execution time of *LeanMD* with no load balancer, HwTopoLB, and the two proposed LBs can be seen in Figure 2. All load balancing algorithms start by providing performance improvements of 31% over the baseline (no load balancer). Still, as the platform and application grow, HwTopoLB's improvements decrease. Meanwhile, HwTopo+Nuco is able to maintain improvements between 12% and 27% over the baseline, and HwTopo+HwTopo keeps it between 25% and 31%.

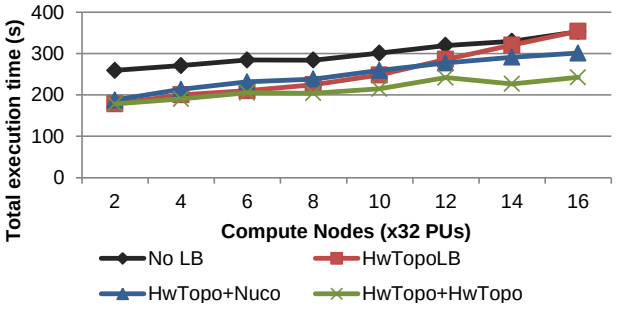

Figure 2. Total execution time of *LeanMD* with different load balancing strategies.

HwTopoLB's performance decreases as we increase the parallel platform because its load balancing time increases. Its average load balancing time for 16 CNs is 50 s, which practically overcomes its iteration time benefits. This does not affect the hierarchical load balancers as much. Still, HwTopo+HwTopo outperforms HwTopo+Nuco because HwTopoLB is a faster algorithm than NucoLB, and migrates less tasks, which results in a smaller overhead.

4. Conclusion

The overhead of centralized load balancing algorithms can overcome their performance improvements in large scale parallel machines. To avoid this problem, we propose the composition of topology-aware load balancing algorithms as a way to improve application performance. Experimental results showed improvements of up to 30% over our centralized algorithm. These improvements come mainly from reducing the time spent load balancing.

Future work includes additional performance evaluations with different benchmarks and applications, and strong scalability tests.

References

[1] L. L. Pilla, C. P. Ribeiro, D. Cordeiro, C. Mei, A. Bhatele, Navaux, F. Broquedis, J. Mehaut, and L. V. Kale. A Hierarchical Approach for Load Balancing on Parallel Multi-core Systems. In *Parallel Processing (ICPP), 2012 41st International Conference on*, pages 118–127, 2012.

[2] L. L. Pilla, C. P. Ribeiro, P. Coucheney, F. Broquedis, B. Gaujal, P. O. A. Navaux, and J.-F. Méaut. A Topology-Aware Load Balancing Algorithm for Clustered Hierarchical Multi-Core Machines. *Future Generation Computer Systems*, 2013.

[3] G. Zheng, A. Bhatele, E. Meneses, and L. V. Kale. Periodic Hierarchical Load Balancing for Large Supercomputers. *International Journal of High Performance Computing Applications (IJHPCA)*, Mar. 2011.