Runtime Support for Adaptive Resource Provisioning in MPI Applications

Gonzalo Martín, David E. Singh,
Maria-Cristina Marinescu, and Jesús Carretero

Computer Science Department, Universidad Carlos III de Madrid
{gmcruz,desingh,mcristina,jcarrete}@arcos.inf.uc3m.es

1 Introduction

The work we present in this paper focuses on dynamic provisioning of computational resources depending on the performance requirements of the application and the characteristics of the cluster available for execution. We target applications which exhibit variable performance over time. The idea is to dynamically optimize the cost (total CPU-time) / performance (program execution time) ratio of parallel applications by (1) reducing the number of processors when the computation requirements decrease enough to justify it, and (2) moving computation onto those processors that can compute faster but don't require extensive remapping of the data. This approach adapts well to time-shared platforms in which many applications may need to execute on the same cluster at the same time, and allows users to implement different cost / performance tradeoffs.

The approach we are proposing differs from AMPI [1] in that they exploit process virtualization while we employ non-virtualized MPI processes. It also differs from DynMPI [4], which drops those nodes from computation which most degrade the performance of the application. In contrast, our approach removes processes only when the computation requirements of the application decrease.

2 The Basic Architecture

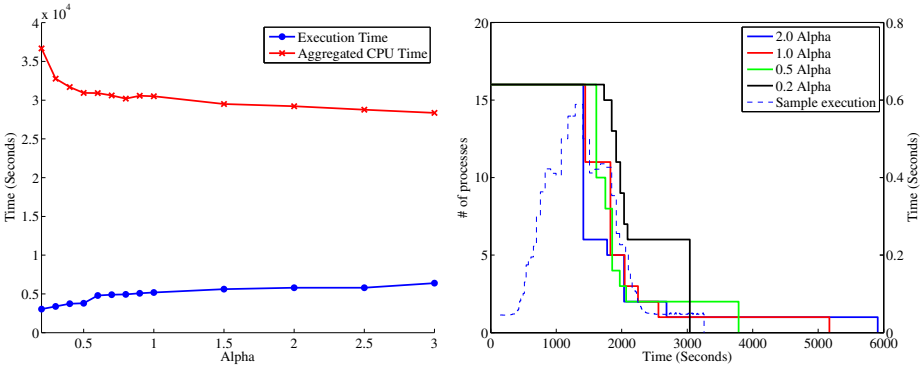
The main components of our runtime environment are the **decision module**, the **scheduler**, and the **monitoring layer** which tracks the performance of the application and feeds this data to the decision module. The decision module implements heuristics to establish how many resources to assign to the application at different points during its execution. This decision is communicated to the scheduler, which elects the set of processes that will continue executing such that they are located on the compute nodes with fastest execution time and which involve minimum data remapping. For space reasons we skip most details of the runtime environment [2] and focus on the decision module.

We have evaluated our framework for EpiGraph [3], an iterative, distributed simulator for infectious diseases which exhibits a significant variability in the iteration cost during its execution. Our heuristic Throughput-Based algorithm (TB) starts from the assumption that the user provisions a maximum number of

J.L. Träff, S. Benkner, and J. Dongarra (Eds.): EuroMPI 2012, LNCS 7490, pp. 294–295, 2012.
© Springer-Verlag Berlin Heidelberg 2012

processors (P_{max}) for executing an application. Using this amount of resources, as the iteration time increases the throughput decreases and reaches its minimum value. This is the point where the application reaches maximum speedup, after which this decreases. A low speedup implies using many resources for a low performance improvement. The idea of TB is to reduce the number of processors to meet the cost/performance requirements.

The TB algorithm takes as input from the monitoring layer the current execution time of the program t_s. Based on the previous (t_{s-1}) and current execution times, TB predicts the execution time for the next iteration interval (t_{s+1}). The number of processes that will execute during this interval (P_{s+1}) is computed by the formula $P_{s+1} = P_{max} \frac{t_{s+1}}{\alpha * t_s}$. α controls the program throughput rate such that larger values of α imply fewer processes and, as a result, achieve smaller throughputs. Fig.1(a) shows the aggregated CPU time and the overall program execution time for TB on a cluster of 16 compute nodes. For α=0.2 we reduce the aggregated CPU time by 19% with a degradation in overall execution time of 11% when compared to executing EpiGraph on 16 processes without the support to our runtime environment. For the same setup, Fig.1(b) illustrates the progression of the number of resources used over time and the iteration times.



(a) (b)

Fig. 1. (a) Impact of α on execution time, (b) number of resources for TB

Acknowledgments. This work has been partially funded by the Spanish Ministry of Science and Technology under the grant TIN2010-16497.

References

1. Huang, C., Lawlor, O., Kale, L.: Adaptive MPI. In: Rauchwerger, L. (ed.) LCPC 2003. LNCS, vol. 2958, pp. 306–322. Springer, Heidelberg (2004)
2. Martín, G., et al.: Runtime support for elastic execution of epigraph. Technical report (2012)
3. Martín, G., Marinescu, M., Singh, D., Carretero, J.: Leveraging social networks for understanding the evolution of epidemics. BMC Syst. Biol. 5(3) (2011)
4. Weatherly, D., Lowenthal, D., Nakazawa, M., Lowenthal, F.: Dyn-mpi: Supporting mpi on medium-scale, non-dedicated clusters. JPDC 2006 66(6), 822–838 (2006)