Dynamic resource adaptation for coordinating runtime systems

Extended Abstract

Stuart Gordon
Heriot-Watt University
sg315@hw.ac.uk

Sven-Bodo Scholz
Heriot-Watt University
S.Scholz@hw.ac.uk

Abstract

In this paper we propose a new approach towards negotiating resource distributions between several parallel applications running on a single multi-core machine. Typically, this negotiation process is delegated to the operating system or a common managed runtime layer. Alternatively, we have formulated a modest extension for arbitrary runtime systems that enables dynamic resource adaptations to be triggered from the outside, i.e., through a separate coordination application.

We demonstrate the effectiveness of the approach in the context of SaC. The paper delineates the required extensions of SaC's runtime system and it discusses how the functional setting substantially eases the process. Furthermore, we demonstrate the effectiveness of our approach when it comes to maximising the overall performance of several parallel applications that share a single multi-core system.

1. Introduction & motivation

As multi-core architectures have now become the norm, an application must not only be able to perform well, it must also have the ability to scale well over parallel architectures and operate in harmony alongside other parallel applications. One might hope that such a negotiation of shared resources can be delegated to the operating system as this traditionally happens when several single-threaded applications share a machine. Although in principle this is possible, the joint performance of several parallel applications on a shared multi-core system typically is rather unsatisfactory. This effect has various technical reasons; in the end these boil down to the fact that the operating system has little if any knowledge about the side conditions that exist in the individual parallel applications: As soon as the parallel applications jointly request more resources than available this over-subscription is resolved by the operating system without taking possible interdependencies into account. To make matters worse, different runtime systems typically have different interdependencies.

An ideal solution would be to express all parallelism as a high-level abstraction open to all languages. Attractive in principle, it has so far proven to be an elusive goal. This is evident by the rapid onset of parallel language implementations and programming models, with no consensus as to which is best. Expressing parallelism may require specific domain, or application knowledge in order to be expressed in an optimal form. Alternatively, language implementations, such as Lithe (Pan et al. 2010), attempt to provide low-level abstractions, with an emphasize efficient parallelism implemented using a standard interface, to implement runtime systems. But however affective this maybe, it requires codes to be heavily modified and reimplemented. A technique also offered by Callisto (Harris et al. 2014), a resource management layer for parallel runtimes, that relies on heavy adaptation in order to be implemented and used as a basis for all runtime systems.

In this paper we propose a radically different approach. Instead of coordinating low-level threads bottom up we propose to coordinate them top-down. The whole approach builds on the idea that it suffices to enable runtime systems to dynamically adapt the number of resources used in order to avoid an over-subscription of resources. That way, the parallel applications can be executed almost entirely independently guaranteeing efficient performance of each individual application.

The contributions of this paper are as follows:

- We propose a generic programming interface to facilitate the negotiation of shared resources.
- A web based application that serves as the user interface.
- We provide a detailed analysis of:
  - SaC's extended runtime system, providing details of the implementation, outlining the relevant benefits of a functional setting.

- we demonstrate the effectiveness of our approach detailing the the overall performance variations of several parallel applications that share a single multi-core system.

Acknowledgments

This work was supported in part by grant EP/L00058X/1 from the UK Engineering and Physical Sciences Research Council (EPSRC).

References

T. Harris, M. Maas, and V. J. Marathe. Callisto: Co-scheduling parallel runtime systems. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pages 24:1–24:14, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2704-6. . URL `http://doi.acm.org/10.1145/2592798.2592807`.

H. Pan, B. Hindman, and K. Asanović. Composing parallel software efficiently with lithe. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 376–387, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0019-3. . URL `http://doi.acm.org/10.1145/1806596.1806639`.