Please mind the gap between intra- and inter-chassis parallelism!

Malte Schwarzkopf † Derek G. Murray ‡ Steven Hand †
†University of Cambridge Computer Laboratory ‡Microsoft Research Silicon Valley

ABSTRACT

Programming abstractions to simplify distributed parallel computing have been widely adopted. Yet, intra-chassis parallelism remains regarded as challenging, despite its often compelling performance advantages over distribution.

We believe that—especially in the face of increasing architectural diversity inside the chassis [3]—benefits are to be had from programming a single machine using abstractions similar to those used for programming distributed systems. Multi-core operating systems have already realized this vision by using message-passing as a core primitive [1], but we argue that in large-scale parallel data processing, we can go further: in particular, only requiring the application programmer to write straight-line serial task code, and auto-parallelizing it transparently using the execution framework, which internally uses optimizations available inside a chassis, such as shared memory. The particular appeal of this approach lies in its generality: the same application code seamlessly scales out to clusters of machines.

Recent work on task-parallel programming models has introduced abstractions permitting dynamic adaptation of task-parallel programs to their execution environments [2]. Based on this, we conjecture that expressing programs as dynamic task graphs achieves the generality we seek. Efficient support for such a model, however, requires an integration of OS-level and runtime resource management for task placement—which is feasible in a data-center environment.

Our prototype system achieves comparable performance with a shared-memory implementation of the *k*-means clustering algorithm when running inside a multi-core machine, while also scaling beyond the computational capacity of a single machine, facilitating *multi-scale* parallelism for the algorithm.

BODY

Existing, easy-to-use distributed programming models are sufficient to enable multi-scale parallel computation inside and across machines.

REFERENCES

[1] A. Baumann, S. Peter, A. Schüpbach, A. Singhania, T. Roscoe, P. Barham, and R. Isaacs. Your computer is already a distributed system. Why isn't your OS? In *Proceedings of HotOS 2009*, pages 12–12. USENIX Association, 2009.

[2] D. Murray and S. Hand. Non-deterministic parallelism considered useful. In *Proceedings of HotOS 2011*, 2011.

[3] S. Smith, A. Madhavapeddy, C. Smowton, M. Schwarzkopf, R. Mortier, R. N. M. Watson, and S. Hand. The Case for Reconfigurable I/O Channels. In *Proceedings of RESoLVE*, 2012.

Volume 1 of Tiny Transactions on Computer Science