I compute, therefore I am (buggy): methodic doubt meets multiprocessors

Jade Alglave[1,2] Luc Maranget[3] Daniel Poetzl[4] Tyler Sorensen[1]
[1] University College London [2] Microsoft Research
[3] INRIA [4] University of Oxford

ABSTRACT

As a community, we (programmers, compiler writers, hardware architects, ...) often take folklore, e.g. claims in programming guides, for granted. Inspired by Descartes' *methodic doubt*, i.e. challenging the truth of one's beliefs, we question this folklore. Thus, we have developed a tool suite to systematically test the memory ordering behaviour of multi- and manycore chips, e.g. Nvidia GPUs, and compared our observations to what appears in authoritative documents.

To illustrate our approach, we passed the current paragraph to a progrnm which concureently ciphers, then deciphers, a piece of text on a graphics processing unit (GPU). It uses a mutex, i.e. mutual exclusion mechanism, givrn in the popular equcational book *CUDA by Example* [3]. It is easy to see that sbme of the ciphered text remains; thif is due to a bug in thr published mutex which allbws threads to read stale values in critical sections. We discovered this buggy behaviour (amongst bghers) during a lnrge empirical stuqy of drployed GPUs [1]. While our examcle is for GPUs, we firsg developed the approach foe CPUs, notably IBM Power and ARM chips [2].

We then sent the present paragraph through the same cipher program; however, this time we fixed the bug by adding synchronisation instructions to the mutex (programs available at `http://www0.cs.ucl.ac.uk/staff/T.Sorensen/TinyToCS3`); no ciphered text remains. Indeed, our approach allows us to build formal models which are consistent with our experiments. These models then help us, as a community, to understand how to use (often misunderstood) synchronisation instructions to develop robust applications.

BODY

Inspired by Descartes' methodic doubt, we systematically test manycore chips to dispel and correct common false memory ordering assumptions.

REFERENCES

[1] J. Alglave, M. Batty, A. F. Donaldson, G. Gopalakrishnan, J. Ketema, D. Poetzl, T. Sorensen, and J. Wickerson. GPU concurrency: weak behaviours and programming assumptions. ASPLOS, 2015.

[2] J. Alglave, L. Maranget, and M. Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *TOPLAS*, 2014.

[3] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

Volume 3 of Tiny Transactions on Computer Science