Herding cats IV

Jade Alglave

Arm and University College London

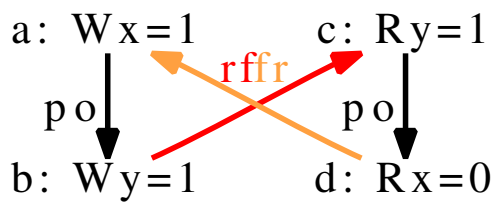
March 2019

Recall the Message Passing example

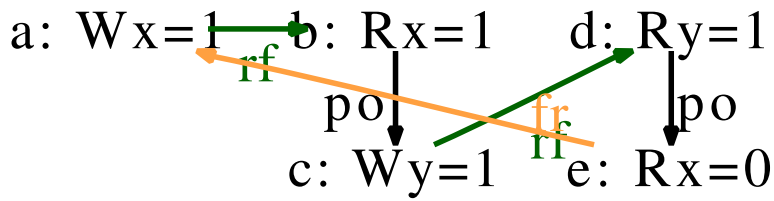
initially x=y=0

P_0	P_1
(a) x $\leftarrow$ 1	(c) r3 $\leftarrow$ y
(b) y $\leftarrow$ 1	(d) r4 $\leftarrow$ x

r3=1 and r4=0 allowed?



And now consider its sibling WRC



Multi-copy atomicity

- SC, TSO and Armv8 are multi-copy atomic
- IBM Power isn't

A-cumulativity

```
let A-cumul(r) = rfe? ; r
let cumul-fence = A-cumul(fence)
```

Linux: a common denominator of hardware models? [Torvalds, 2016a]

> *Weak memory ordering is [...] hard to think about [...] So the memory ordering rules should [...] absolutely be as tight as at all humanly possible, given real hardware constraints.*

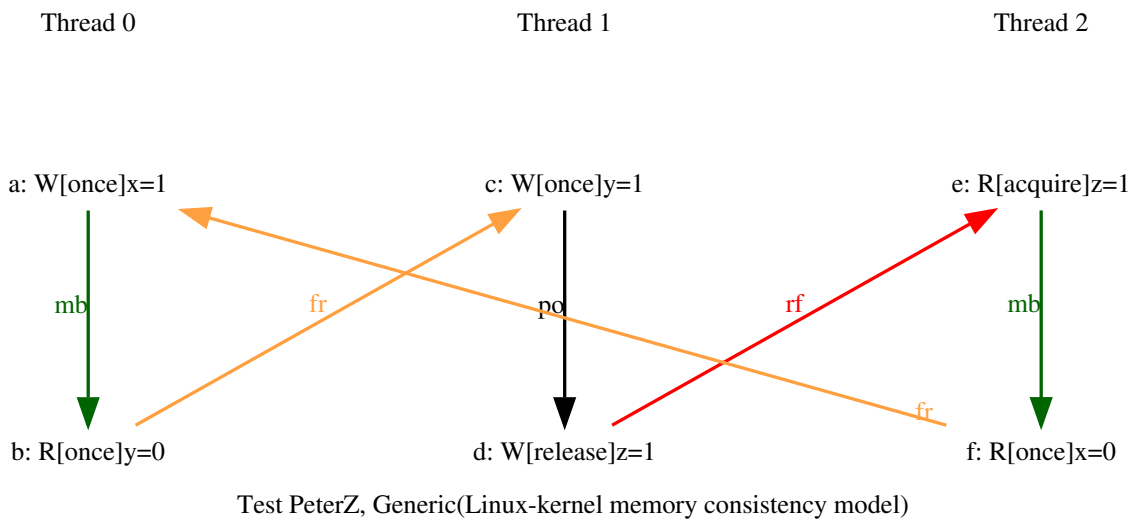
Not an envelope for the architectures it supports? [Molnar, 2013]

> *it's not true that Linux has to offer a barrier and locking model that panders to the weakest (and craziest!) memory ordering model amongst all the possible Linux platforms—theoretical or real metal. Instead what we want to do is to consciously, intelligently pick a sane, maintainable memory model and offer primitives for that—at least as far as generic code is concerned. Each architecture can map those primitives to the best of its abilities.*

An example from PeterZ

https://www.spinics.net/lists/kernel/msg2421883.html

PeterZ: Forbidden



Test PeterZ, Generic(Linux-kernel memory consistency model)

References

Here are pointers to some of the material we have studied:

- the herd+diy toolsuite: `diy.inria.fr`
- the Armv8 model:

  `https://github.com/herd/herdtools7/blob/master/herd/libdir/aarch64.cat`
- the Linux model:

  `https://github.com/torvalds/linux/blob/master/tools/memory-model/linux-kernel.cat`

There you go!