Linux Symmetric Multiprocessing

Stephen Tweedie
Red Hat Software
sct@redhat.com

Contents

- What's all the fuss about SMP?
- Why SMP is *hard*
- History of SMP on Linux
- State of the art and cool stuff in 2.3

So why does anybody care about SMP?

- SMP is easy to program. Very true!
- SMP is scalable. True, but not as true as some vendors would have you believe.
- SMP is easy to administer. Absolutely: you don't get better Single System Image than this.
- SMP lets hardware vendors sell more expensive boxes. Don't underestimate the marketing department.

What makes it harder than UniProcessor computing?

- Correctness
  - We suddenly have data structures which can be accessed from multiple CPUs at once! We need extra locking
  - Interrupts get complex...
- Performance
  - Locking granularity
  - Cache coherency is expensive
  - Need to propagate some updates between CPUs

A Brief History of Linux SMP

- In the beginning there wasn't any.
- 2.0: First support of any kind, on Intel ("give me a motherboard and 6 months...")
  - Single global kernel lock (ugh), but it worked
- 2.2: Much better support, more platforms working
  - Key fine-grained locking for device interrupts, scheduling, signals
  - Support for Sparc and Alpha SMP

Current state of SMP in 2.3

- New primitives: wake-one, spin_lock_bh()
  - Major areas where the global kernel lock has gone:
  - Separate read/write locks to protect fib, dev list and protocol list
  - WIP: Multi-thread the network "bottom-half"
  - The page cache has per-page locks
- Microsoft are Linus's new test labs, identifying performance bottlenecks for us

Does it work?

- Yes! But we still hit scaling limits in places. We're working on it...
- You can buy 8-way Intel boxes today
- Linux has run on 16-cpu or larger Sparcs
- ```
  time make -j4 "MAKE=make -j4" bzImage
  252.39user 22.32system 1:12.79elapsed 377%CPU
  (0avgtext+0avgdata 0maxresident)k
  ```
- You can count on it getting better!
  ```