Beyond Expert-Only Parallel Programming?
In 2006, Linus Torvalds noted that since 2003, the Linux kernel community's grasp of concurrency had improved to the point that patches were often correct at first submission.

Why the improvement?
- Not programming language: C before, during, and after
- Not synchronization primitives: Locking before, during, and after
- Not a change in personnel: Relatively low turnover
- Not born parallel programmers: Remember Big Kernel Lock!

So what was it?
Sea Change In Linux-Kernel Parallel Programming: A Virtuous Cycle

Acculturation

Tooling

Economics
Sea Change In Linux-Kernel Parallel Programming: A Virtuous Cycle

What can be done and how things are done.

Acculturation

Tooling

Development investments to ease future development

Economics

“Time is money,” hardware availability

“Time is money,” hardware availability
Sea Change In Linux-Kernel Parallel Programming: A Virtuous Cycle

- What can be done and how things are done.
- More developers, productivity
- Better ROI for additional tooling
- Development investments to ease future development
- "Time is money," hardware availability

- Acculturation
- Tooling
- Economics
Sea Change In Linux-Kernel Parallel Programming: A Virtuous Cycle

Sample code → What can be done and how things are done.

Acculturation

Faster & easier development

Tooling

Better ROI for additional tooling

Economics

More developers, productivity

Development investments to ease future development

“Time is money,” hardware availability

sparse, lockdep, and coccinelle

Better ROI for additional tooling

More developers, productivity

“Time is money,” hardware availability

sparse, lockdep, and coccinelle

Development investments to ease future development

Faster & easier development

Sample code → What can be done and how things are done.
Linux-Kernel Economics

- **1990s:** Carrying 5 Model-D CPU boards to Sequent Benchmarking center
  - Three times the purchase price of my house!

- **2004:** Grad student bought dual-core Mac for the sole purpose of giving a parallel-processing talk using a parallel processor
  - From three times the purchase price of a house to a fraction of a used car in less than 15 years

- **Now:** Nearly free, especially if you lock into a 2-year plan
**Sample Linux-Kernel Tooling**

- **sparse**: Motivated by user/kernel pointer errors
  - And by ca-2004 excitement over static analysis of software
  - Extended to concurrency (locking and RCU): approximate analysis

- **lockdep**: Motivated by ca. 2004-5 real-time work
  - Preemptibility greatly increases the probability of concurrency bugs
  - Real-time developers got tired of fixing others' deadlocks
  - The lockdep tool further increases the probability of detecting deadlock
    - Forcing developers to learn to avoid deadlocks

- **coccinelle**: Academic project! (“sed” that understands C)
  - Tested project on Linux kernel, submitted bug reports and fixes
    - Fixes can be automatically generated
  - Patches from coccinelle have been in the top-20 developers
    - (See contributions from Julia Lawall)
What is Left to Work On?

- SMP systems with many hundreds (or thousands) of CPUs
- Special-purpose hardware accelerators (GPGPUs, FPGAs)
- Parallel systems with real-time/energy-efficiency constraints
- Parallelize difficult-to-parallelize applications
- Unit volumes of multicore embedded systems: >100 million
  - Extreme reliability required: With the proper requirements set out!
- Codifying current expert-only techniques for general use
- Rigorous theoretical models of current expert-only techniques
- [Your idea here]
Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM and IBM (logo) are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
Discussion