Hot Topics and Future Directions in Programming Languages

Guy L. Steele Jr.
Sun Microsystems Laboratories
May 9, 2007

Parallelism

- There's been a fascination for decades
- There was a boom in the 1980s (SIMD and MIMD)
  - Fortran 90, High Performance Fortran
  - ZPL
  - Connection Machine Lisp, C*, NESL
  - Co-array Fortran
  - OpenMP (for C and Fortran)
- Java: platform-independent multithreading
- Big science machines: now “petascale computing”
- Now multichip and multicore desktops and servers

Transactional Memory

- Locks can be hard to use:
  - Maintaining correspondence of data to locks
  - Problems with procedural composition
  - Pessimistic; no read-read concurrency
  - Deadlock, priority inversion
- Transactions
  - Keyed to data actually touched
  - Optimistic; read-read concurrency OK
  - Cannot deadlock (but livelock still possible)
  - Problems with procedural composition
  - Integration with other language features (e.g., exceptions)

Type System Design/Type Theory

- What can you say?
  - Expressive power
  - Generic types, inheritance, mutual exclusion
  - Conditional relationships
  - Problems: soundness, completeness/decidability
- What can you know?
  - Not just data types, but effects and other information
  - Static analysis
    - Type systems are one of many strategies for static analysis
    - Can other kinds of static analysis be described as type systems?

Automatic Storage Management

- Garbage collection, reference counting, other
- Scaling to large memory sizes
- Parallel garbage collection
- Concurrent garbage collection
- Garbage collection with low pause times (“real time”)
- Tuning GC for particular applications
- This continues to be a hot topic after five decades!

Parsing (believe it or not!)

- Fortran syntax was kind of ad hoc
- Parsing theory: lots of algorithms
- LR(k) (especially LR(1)) and recursive descent
- yacc
- Packrat parsers afford new possibilities

Compilation and Optimization

- The hardware keeps changing, so the challenges to compilers keep changing
- Balancing use of multiple resources
  - Processors
  - Memories
  - Caches
  - Networks
  - Latency versus bandwidth; delay versus throughput
- Platform independence versus exploiting features
  - Tradeoffs of delaying versus committing decisions

Dynamic Compilation/Optimization

- Recompilation effort expended while program runs
- Can do better than static compilation
  - Use performance data gathered during execution
  - Tuning is a big issue: how much overhead is worthwhile?
- What changes when you have multiple processors?

Automatic Algorithm Selection

- Higher-level than ordinary optimization
- Sequential versus parallel strategies
- Choices of algorithm (such as sorting or FFT)
- Choices of data structure (such as strings)

Scripting Languages

- JavaScript, Python, Perl, Ruby, . . .
- How to provide a gentle introductory learning curve while maintaining robust scalability?
- How to combine interactiveness and rapid prototyping with good performance?

Programming in the Large

- Managing large modules and interfaces (APIs)
- Interchangeable components
- Version control
- Support for debugging, profiling, automated testing
- Interaction with IDEs

User Studies

- There need to be a lot more research on what actually makes programmers more productive!
  - Need to raise language design from purely an art form (or popularity contest) to a better-understood science
- Need controlled human-factors experiments
  - Which are themselves labor-intensive and time-consuming

My Own Current Research: Fortress

- Growing a language
  - Technical features in the language to aid language growth
  - Emphasis on providing features through library code
  - Emphasis on exposing mechanism to library coders that enables them to hide mechanism from application coders
- Mathematical notation
  - Make programs look more like the pseudocode we use
- Parallelism everywhere
  - Risk upending some previous standards
  - All “loops” are parallel by default
    - Sequential loops are intentionally a little more verbose

A Final Thought

- Moore's Law will probably span my entire career
- It will probably run out during yours!
  - What are the consequences for computer science?
  - What are the consequences for programming languages?
- I may miss out on quantum computing, biocomputing, or whatever is the next major revolution
- Such a revolution may be your greatest challenge!
  - What are the consequences for computer science?
  - What are the consequences for programming languages?

guy.steele@sun.com

http://research.sun.com/projects/plrg
http://fortress.sunsource.net