Should We Fear Concurrency?

James Larus
eXtreme Computing Group (XCG)
Microsoft Research

Workshop on Advancing Computer Architecture Research (ACAR)
February 22, 2010
• 2005: Intel, AMD, Sun, IBM, … announce future processors will be parallel
  • “multicore”, “manycore”, “CMP”
• Explicitly parallel architecture
  • replicate processor and connect through shared memory
  • simplify processor microarchitecture to increase number per chip
  • fixed (lower) clock frequency

• Potentially 2x, 4x, 8x, … computation/tick
  • performance again tracks Moore’s law
  • e.g. DSP and graphics
  • all is good, again

New Era of Concurrency
• Make programmers feel your pain
  • parallel programming now **necessary** to improve performance
• Parallelism is painful & disruptive
  • Transistor $\rightarrow$ mainframe
  IC $\rightarrow$ minicomputer
  VLSI $\rightarrow$ microcomputer
  CMP $\rightarrow$ ?
• What is it like to live when computers do not run existing programs faster every year?

Architects’ Revenge!
Can We Program Parallel Computers?

- Advent of explicitly parallel machines
  - only parallel code benefits from CMPs
  - existing code runs slower on new machines
- Huge challenge in effectively using parallelism
  - no parallel analogue to Von Neumann model
  - difficult to write correct, efficient parallel programs
  - parallelism only successful in narrow domains
  - little insight from research
    - no good ideas lying around for industry to pick up
• Tried Transactional Memory
  • Didn’t work
• Trying deterministic parallelism
  • Won’t work (definition of determinism isn’t deterministic)
• GP-GPUs
  • Works well (for some problems)
• Map-reduce (Hadoop)
  • Works well (for some problems)
• Servers continue as the parallel success story
  • Cloud computing

5 Years Later
- Few non-technical, non-game applications
  - Where is demand for higher performance?
  - And, they are data-parallel (Photoshop, Excel, Video)
- Not enough CPU parallelism to rewrite applications
  - Too much pain, too little gain for 4x improvement
- Correctness and performance are large barriers
  - New skills, new models, new tools
- No clear guidance to developers
  - Programming model, language, tools?
- Shift to web and other platforms
David Callahan’s View

Distributed Systems

Unstructured Parallelism

Data Parallelism
• Successes are data parallel
  • Operation applied across aggregate of data
  • Independent computations
  • Sequential code

• What’s not to like?
  • Simple programming model
    • Sometime hard to use (cf Hillis PhD)
  • Errors are not subtle

• Not appropriate for all problems
  • Performance still a challenge

Data Parallelism
• Unstructured parallel programming errors are synchronization errors
  • Data races, deadlock, atomicity, …
• What would life be like without synchronization?
  • (Functional)
• Independent parallelism does not require synchronization
  • Data parallelism
  • Bulk Synchronous Parallelism (BSP)
  • Transactional
  • Functional

Synchronization is **the Problem**
(Not Parallelism)
What is Cloud Computing?

• More than Amazon Web Services / Microsoft Azure / Google AppEngine
  • Platform as a service
    • On-demand, internet computing resources hosted by someone else
  • Commodification of distributed computing
• New software model enabled by new platform
  • Inherently distributed
    • Range of clients (single purpose → rich)
    • Multiple computers in the cloud
  • Ubiquitous access to information and computation
Cloud Computing
The Cloud is Inherently Parallel

- Client-cloud
  - Distributed system

- Cloud infrastructure
  - Distributed system and independently parallel servers

- Clients
  - Parallel to reduce power

- Distributed systems built on message passing
  - Long, variable latency communication
  - Scalability to millions (billions) of servers
  - Varying communications patterns
  - Failure isolation
• Eliminate shared-memory programming
  • (In most cases)
• Error-prone programming model
  • Does anyone understand memory consistency models (except Sarita)?
  • Has anyone ever written a non-trivial, properly synchronized program?
  • Tools cannot fix flawed model
• Why should communications be implicit?
  • Count instructions, memory references, communications
• Not scalable
  • Large systems
  • Distributed systems
• No failure isolation

Bite the Bullet, Accept the Inevitable
• Opportunity to separate pointer semantics from communications
  • Explicitly transfer pointer-rich data between processors
• Enormous advantage in uniform pointer semantics across machines
  • Pass rich data structures without serialization
• Appropriate for tightly coupled processors (same failure domain)
  • How to recover from losing part of address space?
  • Not DSM
  • Serialize data otherwise
• Opportunities for optimization
  • Pointer passing, rather than copying

Shared Address Space ≠ Shared Memory
- Channels are strongly typed (value & behavior), bidirectional communications ports
  - Messages passing with extensive language support
- Messages live outside processes, in exchange heap
  - Only a single reference to a message
- “Mailbox” semantics enforced by linear types
  - Copying and pointer passing are semantically indistinguishable
- Channel buffers pre-allocated according to contract
Intel Single-Chip Cloud Computer
• Get the software out of the path
  • Infiniband vs Ethernet
• Inexpensive, low-latency, high-bandwidth networks
  • Chip, board, rack, container, data center, world
  • VL2 work at MSR
• Low-latency message delivery and processing
• Channel, not packet
  • Think flows, not individual messages
• Security at channel setup, not communications
• Failure notification
• Low-power

Architectural Support for Message Passing
• Verifiable channel contracts
  • Specification of process’s interface
  • Mechanically verifiable
• Language integration
  • Message passing is asynchronous
    • Procedure call is synchronous
    • Threads or events or ?
  • Integration of pattern match and messaging
    • Erlang
  • Error propagation and handling

Programming Support for Message Passing
• “Insanity: doing the same thing over and over again and expecting different results” – Einstein
• Why so much attention to unstructured, shared-memory parallelism?
  • Impossible to use well
  • Not scalable
• Why so little focus on synchronization?
• Why so little focus on message passing?
• Look at what works, build on it
  • Data parallelism
  • Message passing
• Structure unstructured parallelism
  • Functional, BSP, ???
• Parallel programming models should drive architecture

Conclusion

Disclaimer: These are my views, not Microsoft’s (though, they should be).