“Why Events are a Bad Idea (For high-concurrency servers)”

Paper by Rob von Behren, Jeremy Condit and Eric Brewer, May 2003

Presentation by Loren Davis, August 2011
What’s this Paper About?

- Who are the authors?
  - Rob von Behren, Jeremy Condit and Eric Brewer were CS professors at UC Berkeley
  - Eric Brewer was a co-author of SEDA

- What did they do?
  - They wrote a threading library that scaled well up to 100,000 threads
  - They “refined” the duality argument in Lauer and Needham’s paper

- Why is that required reading in this course?
  - Glad you asked
Their Use of Previous Work

- The key result of this paper is not that they put threads ahead of events in the tug-of-war.
- Based on the same papers we read previously, they predicted that threads and events should have similar performance.
- And they were right.
A Review of Threads and Events
Lauer and Needham

- Contrasted “Message-Oriented systems” (few, static processes and message-passing) with “Procedure-Oriented systems” (many, dynamic processes and shared memory)
- Described message port/message queue library and monitor library based on C.A.R. Hoare
- Argued that neither had any inherent advantage over the other
Lauer-Needham Duality

**Event-Driven**
- Message Channels
- Dispatch message
- Delayed send, await
- SendReply
- Wait in message loop
- Message handlers
- Wait for messages

**Threaded**
- Procedure names
- Procedure call
- Create, join threads
- Procedure return
- Wait for lock
- Procedure bodies
- Condition variables, signals
What von Behren Took

- Ignored the definitions given for “Message-oriented” and “Procedure-oriented”
  - Which were not well-defined
  - Or relevant to current operating systems
  - Or even used in the duality argument
- Recast the argument as about control flow through a blocking graph
- Applied the results to events and threads
- Concluded that the two should, in theory, be as fast
- Provided actual data
The title, “Why Events are a Bad Idea (For high-concurrency servers),” is a riff on Osterhout’s paper (5 April)

- “Why Threads Are a Bad Idea (For most purposes)”
- Yet, “his arguments do not conflict with ours.”
- In particular, Osterhout supported threads for true concurrency
- But believed that threads are hard to program
Adya et al.

- Li Lei will present this paper
- Von Behren et al. use one result in particular:
  - Events had been thought to be easier to program
  - In reality, it’s *cooperative multitasking* that’s easier to program
  - It’s possible to get cooperative multitasking with threads
Welsh, Culler and Brewer

- Wrote the Staged Event-Driven Architecture (SEDA)
- Applications included a web server, Haboob
- One of the few event-driven frameworks to rigorously use Lauer-Needham messaging
- Believed that messages had inherent advantages over threads
SEDA Versus Threads in Two Graphs

- The graph on the left repeats the SEDA benchmark.
- The graph on the right compares SEDA to a new threading system with similar scheduling (green) and one that prioritizes new connections (blue).
What Happened

● The authors of the 2001 SEDA paper believed that threads were the problem
  – “[C]ache and TLB misses, scheduling overhead, and lock contention”
● The authors of this paper, in 2003, believed that that couldn’t be true
  – How could it be, when every event-driven program has a threaded dual with the same flow? (Half-true.)
● They found the real problem in the threading library
  – Non-scalable, $O(n)$ operations
  – Too many context switches
● Haboob becomes CPU-limited at 512 connections
Future Research

- Some of the points they raised, they dealt with by saying that solutions *should* exist, thanks to Lauer-Needham duality, but didn’t actually write
  - These include introspection and modularity

- They give examples of cooperative threading packages, but not ones designed for large numbers of blocking threads

- They call for improved compiler support for threading, including explicit parallel blocks, static analysis of them, dynamic stack growth, and live-state management
Conclusions

- Threads perform similarly to, or even better than, events
- They have a simpler programming model
- Static analysis of them is easier
- Compilers should tightly integrate them