Multicore OSes
Looking Forward from 1991, er, 2011

David A. Holland, Margo I. Seltzer
{dholland, margo}@eecs.harvard.edu

May 11, 2011
The Scenario

• Physical limits have been reached.

• Hardware isn’t getting faster any more.

• To go faster we’re going to have to run in parallel.
The Scenario

• Physical limits have been reached.
• Hardware isn’t getting faster any more.
• To go faster we’re going to have to run in parallel.

Parallel code is hard!
The Scenario

- Physical limits have been reached.
- Hardware isn’t getting faster any more.
- To go faster we’re going to have to run in parallel.

Parallel code is hard!

Scalable parallel code is harder!
The Scenario

- Physical limits have been reached.
- Hardware isn’t getting faster any more.
- To go faster we’re going to have to run in parallel.

Parallel code is hard!

Scalable parallel code is harder!

CRISIS!!!!!!
Meh.
Ok, it’s not quite 1991.

From software, multiprocessor $\approx$ multicore.

Lessons from the past twenty years:

- Shared-memory code with locks doesn’t scale.
- Hardware will end up shared-nothing.
- Programming will involve message passing.
Ok, it’s not quite 1991.

From software, multiprocessor \(\approx\) multicore.

Lessons from the past twenty years:

- Shared-memory code with locks doesn’t scale.
- Hardware will end up shared-nothing.
- Programming will involve message passing.

Let’s skip the bankruptcy filings and go straight to messages.
Lightweight messages and channels

Different programming paradigm.

Has some chance of scaling.

Not actually new:

- Communicating Sequential Processes
- pi calculus
- Erlang
- goroutines
What It Looks Like (in “C”)

```c
chan <- value; /* send on channel */
value <- chan; /* receive from channel */
```

Comparable to procedure calls.

```c
choose {
    option x <- c1: foo(x); break;
    option x <- c2: bar(x); break;
}
```

Like `select()`.

```c
start { baz(); }
```

Makes a new thread.
The Way Forward

We need whole systems built this way:

language...

and kernel...

and applications.

Not just one research system, either.
The Way Forward

We need whole systems built this way:
language...
and kernel...
and applications.

Not just one research system, either.

Let’s talk about kernels.
(But don’t worry; I’m not advocating Erlang.)
Channel OS Architecture

- System calls will be messages.
- This enables new OS structures.
- Also need a whole new kernel based on channels...
Foreseeable Issues...

- Implementing choice.
- Waiting for channels to become ready.
- What does virtual memory look like?
- Too much parallelism?
- Partial failure.
- Scheduling.

(and of course scaling is still hard)
Project State

→ hot air
vapourware
slideware
demoware
software
abandonware
Multicore OSes
Looking Forward from 1991, er, 2011

David A. Holland, Margo I. Seltzer
{d holland, margo}@eecs.harvard.edu