MULTI-PROCESS DESIGN IN ERLANG

Curt Clifton
Rose-Hulman Institute of Technology

SVN Update *ErlangInClass*

PROGRAMMING FOR MULTICORE

- Use lots of processes
- Avoid side effects
- Eliminate sequential bottlenecks
- Write “small messages, big computations” code

USE LOTS OF PROCESSES

- Want all CPUs busy all the time
- Most easily achieved # of processes >> # of CPUs
- Want processes to do comparable amounts of work

Q1

AVOID SIDE EFFECTS

- Use process loop arguments to maintain "state"
- Don't use process dictionary if you can avoid it
- Be very careful with ETS tables
    - Avoid *public* tables
    - Be careful with *protected* tables
    - Favor *private* tables

Q2

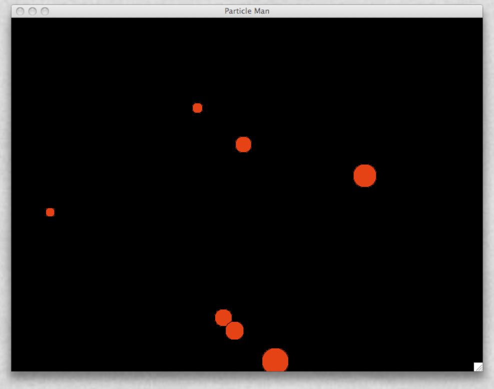
WATCH OUT FOR SEQUENTIAL BOTTLENECKS

- Some things are intrinsically sequential
  - Like disk I/O
- Registered processes are a warning sign
- Often need to find a distributed algorithm

Q3

EXERCISE

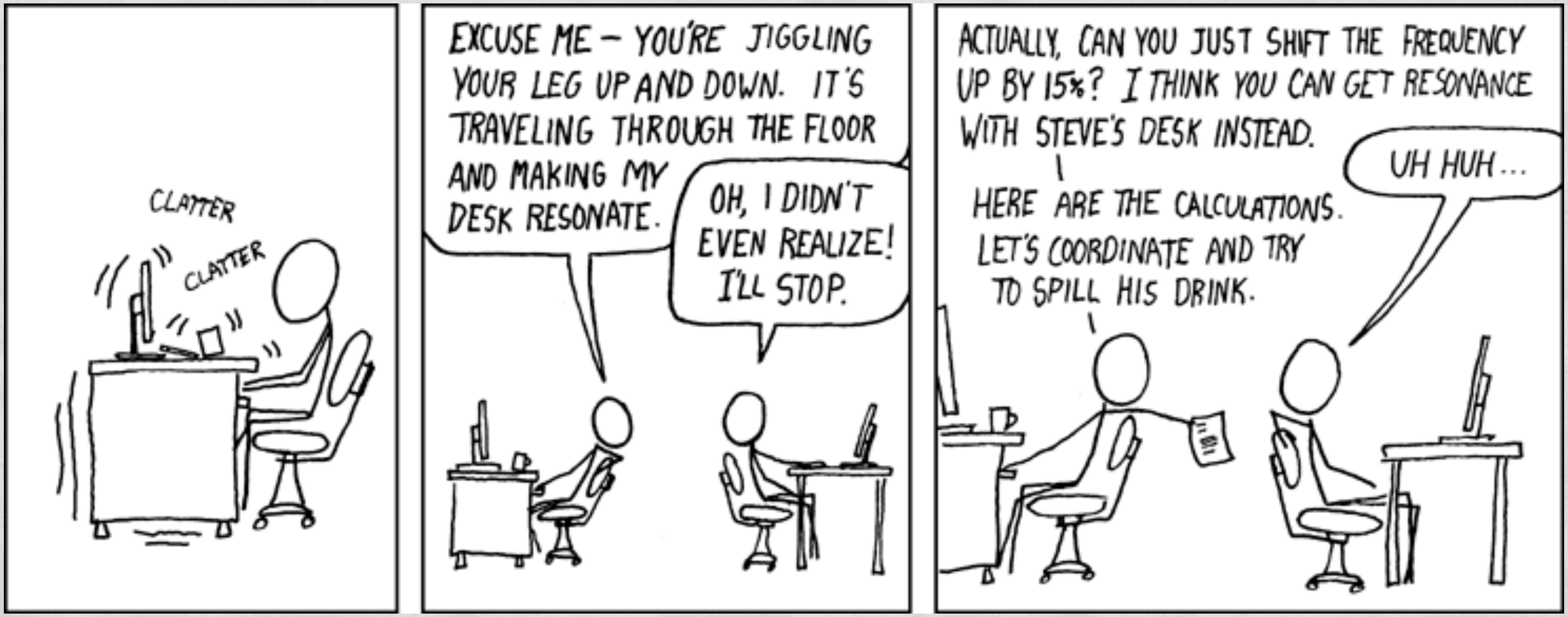
- Design an *n*-body simulation
  - What processes do we need?
  - What messages should they respond to?



- Remember:
  - Use lots of processes
  - Avoid side effects
  - Eliminate sequential bottlenecks
  - Write "small messages, big computations" code

Q4

RESONANCE

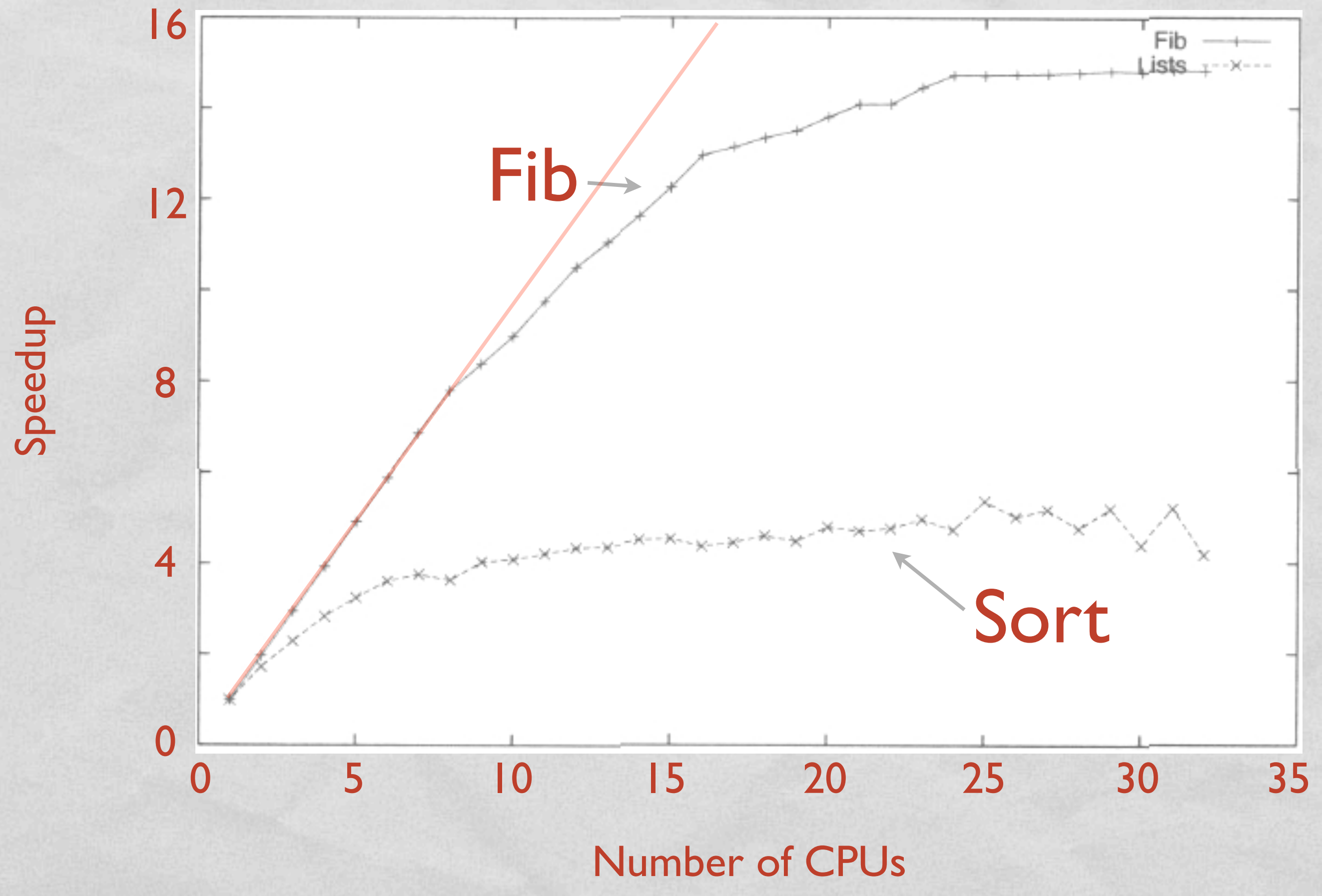

CLATTER
CLATTER
EXCUSE ME – YOU'RE JIGGLING YOUR LEG UP AND DOWN. IT'S TRAVELING THROUGH THE FLOOR AND MAKING MY DESK RESONATE.
OH, I DIDN'T EVEN REALIZE! I'LL STOP.
ACTUALLY, CAN YOU JUST SHIFT THE FREQUENCY UP BY 15%? I THINK YOU CAN GET RESONANCE WITH STEVE'S DESK INSTEAD.
HERE ARE THE CALCULATIONS. LET'S COORDINATE AND TRY TO SPILL HIS DRINK.
UH HUH...

It's really hard to control the frequency, actually.

WRITE "SMALL MESSAGES, BIG COMPUTATIONS" CODE

- Example: open *ErlangInClass/pmap.erl*
    - Study *pmap* implementation
    - Look at various sample tests
    - If you've got multiple cores, use ***erl +S 2*** to use two of them.

PMAP WITH MULTIPLE CORES



Programming Erlang, Fig. 20.1