Class 37:
How to Find Aliens (and Factors)

Finding the Aliens

for signal in signals:
  power = findPowerSpectrum (signal)
  if (isAlien (power)):
    print “Found an alien!” + signal

Finding Aliens

Arecibo Antenna, Puerto Rico
From http://www.adl.gatech.edu/research/tff/final_radps.html

Processing Signals

- Power spectrum
- Find patterns in signal
- Eliminate natural and human-made signals
- Today:
  - BlueGene, 280Tflops/s
  - ~$200 per Giga flop
- No success finding aliens

Finding the Aliens Cheaper

par
for signal in signals:
  power = findPowerSpectrum (signal)
  if (isAlien (power)):
    print “Found an alien!” + signal

Parallel for: instead of doing each element sequentially in order, we can do each element in parallel on a different machine.

Note: python does not actually have parfor, but other languages do.

Public Distributed Computing
Incentives are Necessary

- People will cheat*
- How to cheat?
  - Respond with the "there are no aliens" message without actually doing all the work
- Chances of getting caught?
  - 0 (Assumes all jobs have no aliens. So far this is true.)

* Only applies in real world, not at UVa.

Incentives are Dangerous

- People will cheat*
- How to cheat?
  - Respond with the "there are no aliens" message without actually doing all the work
- Chances of getting caught?
  - 0 (Assumes all jobs have no aliens. So far this is true.)

* Only applies in real world, not at UVa.

Why is finding aliens so "easy"?

- Can be broken into many tasks
- Each task can be done completely independently
  - No shared memory between the tasks
- The data to describe a task and response is small compared to the computing
  - SETI@home jobs are 350KB data download, 1KB upload, 3.9 Trillion operations (several hours on PC)

Doug Szajda and colleagues at University of Richmond work on these problems (see link on notes)
“Harder” Task

- Dividing your PS8 project work among your team

PS8 Update: you have 1 week left!
If you do not have basic functionality working by tomorrow, arrange to meet with me

Scheduling Meetings

Alice wants to schedule a meeting with Bob and Colleen

Partial Ordering of Events

- Sequential programs give use a total ordering of events: everything happens in a determined order
- Concurrency gives us a partial ordering of events: we know some things happen before other things, but not total order

Race Condition

Preventing Race Conditions

- Use locks to impose ordering constraints
- After responding to Alice, Bob reserves all the times in his response until he hears back (and then frees the other times)
### Deadlocks

**Bob**

"When can you meet Friday?"

**Alice**

"When can you meet Friday?"

"9, 11am or 3pm"

Locks calendar for Doug, can’t respond to Alice

**Colleen**

"When can you meet Friday?"

Can’t schedule meeting, no response from Bob

### Why multiple processors is hard?

- Too few ordering constraints: race conditions
- Too many ordering constraints: deadlocks
- Hard/impossible to reason modularly
  - If an object is accessible to multiple threads, need to think about what any of those threads could do at any time!

### Charge

- The easiest way to solve distributed scheduling problems is to “undistribute” them:
  - Find your teammates now and make sure you know what you are doing next week
- Wednesday: Google
  - Read the paper distributed today
- Friday: Review
  - Send me your questions and topic requests
- Monday: PS8 presentations/demos