Alek Westover

Old Brainstorming

1 chill

Algorithm 1 Alg greed

```
procedure GREED0
    while True do
        if all processors are idle then
            if If there are more than p/k queued tasks then
                schedule all (or p) in serial
            else
                Schedule one in parallel
            end if
        end if
    end while
end procedure
```

Algorithm 2 Alg greed

```
procedure GREED1
    while True do
        if all processors are idle then
            if If there are more than p/k queued tasks then
                schedule all (or p) in serial
            else
                Schedule all in parallel
            end if
        end if
    end while
end procedure
```

Algorithm 3 Alg greed

```
procedure GREED2
    while True do
        if all processors are idle then
            q ← number of queued tasks
            schedule ⌊q/p⌋ tasks in serial to each processor
            q' ← q mod p
            if q' > p/k then
                Schedule q' tasks in serial
            else
                Schedule q' tasks in parallel (distributing work equally)
            end if
        end if
    end while
end procedure
```

You can probably do a similar thing for the non-symmetric case. The main idea is locally be greedy, but also lazy about scheduling. So in the non-symmetric case you'd probably be like "if all processors are idle, schedule in a good way."

Algorithm 4 Alg chill

procedure CHILL
  while True **do**
    if all processors are idle **then**
      if If there are more than p/k queued tasks **then**
        schedule all (or p) in serial
      else
        Schedule one in parallel
      end if
    end if
  end while
end procedure

2 Alg X

Algorithm 5 Alg X

This procedure is continuously running.
procedure X
  if there are more than p/k queued tasks $\bmod p$ **then**
    for each task running in parallel **do**
      if it has more than 1 total work left **and** the number of queued tasks $\bmod p$ is at most $p-1$ **then**
        kill this task it (i.e. put it on the queue)
      end if
    end for
    Schedule as many queued tasks as possible to processros that have less than 1 work assigned to them. (scheduling to minimize backlog, i.e. scheduling in ascending order of backlog)
  end if
  if There is an idle processor **then**
    if There is a queued task **then**
      if backlog ≥ 1 **then**
        schedule tasks in serial on any idle processors
      end if
      if backlog < 1 **then**
        schedule a task in parallel, scheduling as balancedly as possible
      end if
    end if
    if There is no queued task **and** there is a serial task that could be cancelled and then redistributed that would result in all cups that are getting the redistribution stuff end up with less work than the thing that was cancelled from **then**
      Cancel the serial task and reschedule as specified
    end if
  end if
end procedure

Claim 1. *Alg X is good.*

Proof. This seems basically impossible to prove, it's a super complicated algorithm with so much branching. □

3 some more strategies

- randomized (smoothed analysis)

- look at discrete version?

Algorithm 6 Randomized alg

```
This procedure is continuously running.
procedure RANDOMIZEDSTRATEGY

    if backlog would be made smaller by cancelling everything and swapping the mode, and scheduling everything according ot the new mode then
        do it!
    end if
end procedure
```

Algorithm 7 Alg binary

```
This procedure is continuously running.
procedure BINARY(mode)
    when you get a new task schedule it to minimize backlog in the current mode (mode is serial or parallel)
    if backlog would be made smaller by cancelling everything and swapping the mode, and scheduling everything according ot the new mode then
        do it!
    end if
end procedure
```