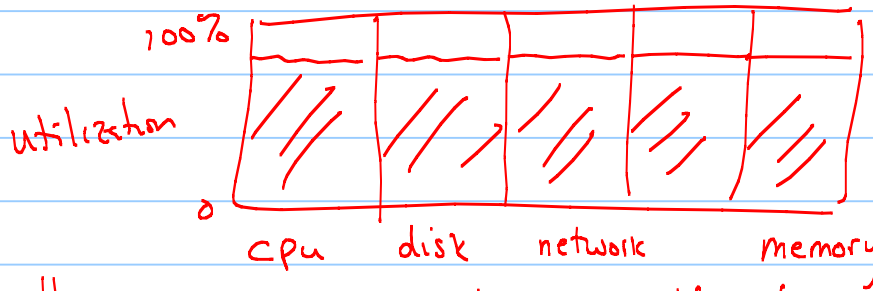
Note Title 2/18/2009

Chapter 6

What are threads used for?

- exploiting parallel resources - processors, I/O, networks.



Threads organize work - mostly for cpu.

Thrashing - high utilization, low useful work

Too many threads — time spent creating / disposing
switching between

Too much memory working set — time spent swapping pages in and out — 100% busy disk

- organize externally concurrent activities
- deferring work to later
- periodic task — a little bit of work every once in a while

→ They use a lot of memory resource doing nothing useful.
rss reader ; top ; task manager
- cron scheduler

Example of balancing resources:

Web server

- while (1) {
  recv request
  read file ←
  send reply
  }
- while (1) {
  recv request
  new Thread (request) . start ( )
  }

read file
send reply
thread ends

Create too many threads — Thrashing on thread switching
— running out of memory

policies

- Want just the right number of threads

* Thread Pool - a collection of threads of given size - that get re-used for different requests.

Task - a unit to be done
- A task is represented in Java by a Runnable or a Callable
- you can directly call the run method.
- you can create a thread and call t.start()
s

Idea: Executor – interface w/
void execute(Runnable cmd)

Different implementation of Executor:

- Run in current thread
- Run in new thread
- Run in thread pool.

— Executor sets the policy about use of threads

Executors class

Executors. newFixedThreadPool (N)

Cached — no maximum

- newSingleThreadExecutor ()

— exactly one thread.

guarantees visibility of earlier tasks memory to later tasks.

- scheduledPool returns ScheduledExecutorService

has add'l methods

schedule (cmd, delay, timeUnit)

scheduleAtFixedRate (cmd, initialDelay, period)

Wonderful unless the tasks depend on each other?

Executors let us tell the system we want to run some tasks.

How do we get results?

Each to call to execute returns a Future.

get() — when result is available it returns it?

get(timeout)

f1 = execute(c1); f2 = execute(c2) ... fn = execute(cn)

f1.get()

f2.get()

fn.get()

To process results in order they are ready:

Executor Completion Service — combines Executor with a completion queue.

- constructed from an Executor, possibly a Blocking Queue.
- it has an execute method and a take() method for getting results

- Listing 6.14 is really confusing - it's intend to Executor Completion Service