Concurrency – a cautionary tale

Grand Central Dispatch (GCD) by Apple
- API for lightweight threads
- Work queues hold functions (or code blocks) to be called
- Queues can manage concurrency, coordinate joining of tasks
- GCD manages scheduling and processor assignment

A simple practical example:
- Generating image thumbnails, for all files in a directory, on a quad-core machine.
- *http://www.mikeash.com/pyblog/friday-qa-2009-09-25-gcd-practicum.html*
- v1: Sequential program – 984 seconds
- v2: Naïve parallelisation – did not run
  - GCD spawns another thread each time one blocks
  - Each task (a) hits the disk, (b) allocates memory to decompress images
  - Then the OS starts swapping, i.e. more disk contention
  - and GCD spawns another thread…
- v3: Read files sequentially in a single task, compress in parallel – 300 seconds
  - This is self-limiting, but can still thrash
  - (Used only 10GB RAM on a machine with 15GB!)
- v4: Limit the number of tasks to 2 x number of CPU cores – 279 seconds

The CPU is not the only resource that threads / processes must share.
Concurrency (and distribution) libraries provide programmer support, not solutions.