

Offset	Topic
00:17	<ul style="list-style-type: none"> • Intro <ul style="list-style-type: none"> • RIP, Arthur C. Clarke <ul style="list-style-type: none"> • http://www.latimes.com/news/local/la-me-clarke19mar19,0,393161.story
02:14	<ul style="list-style-type: none"> • Word of the Week: cyberpunk <ul style="list-style-type: none"> • http://catb.org/jargon/html/C/cyberpunk.html
04:07	<ul style="list-style-type: none"> • Hacking 101: Parallel Computing, Programming <ul style="list-style-type: none"> • Difference between serial and parallel <ul style="list-style-type: none"> • Serial <ul style="list-style-type: none"> • Each step must complete before the next • Specific order • Must get in the car first, then fill the tank at the station, then run errands • Parallel <ul style="list-style-type: none"> • Order independent • If you have enough resources, can do at the same time • Can shop at stores in any order • Can divide the list with a friend, do errands in half the time • Both old concepts with computers <ul style="list-style-type: none"> • Applies to serial, parallel ports • Describe how data is sent on the connected cable • Of necessity, some pieces of low level computing are parallel • The CPU itself has been traditionally serial <ul style="list-style-type: none"> • Up through the 80s and 90s, CPUs simply ran faster • Governed by a clock, can do only so many things each clock tick • There are clock-less systems, uncommon, still pretty much just research • Ticking clock helps with coordination, few parallel components • Problem of frequency scaling <ul style="list-style-type: none"> • Power consumption rises as a function of number of transistors • Also as frequency of clock • Moore's law driven by number of transistors per chip increasing • Plateau in clock speeds, but more transistors • Do more work per tick • This is what multiple core computing does • Good overview of parallel computing <ul style="list-style-type: none"> • http://en.wikipedia.org/wiki/Parallel_computing • Identifies problems • Speed up is limited by slowest purely serial task

Offset

Topic

- Amdahl's Law, Gustafson's law
- Also, there is some non-obvious scaling
- A huge multiple improvement for smaller independent part beaten by modest multiple for much larger independent part
- Parallel programming predates multiple core
 - Had multiple CPUs, symmetric multiprocessing
 - More than one separate CPU per motherboard
 - Uncommon but been around for some time
 - Usually requires special motherboard, special operating systems code
 - Clusters, distributed
 - Parallel scaled way up
 - Problems with interconnects, data sharing
 - Do you use standard networking?
 - Custom hardware?
 - Multiple cores pushes those down to a single component
- Multi threading
 - Even with only one processor, cheats to make it appear like multiple tasks run at the same time
 - Takes advantage of processor running far faster than we perceive
 - Switches between tasks many times per second
 - Looks like each task runs independently
 - Each task is associated with a thread of execution, may also be called processes but same idea
 - Only one thread can be running, others are suspended but don't know it
 - Think of the first time you used Windows
 - Hints at problems that will get worse with multiple cores
 - Order of operations cannot be just random
 - Tasks that share resources have to be coordinated by the operating system
- Challenges
 - Data dependency
 - The input to task B is the output of task A
 - That makes task A dependent on task B
 - Limits parallel execution, task B has to wait
 - Longest chain of dependent task is known as critical path
 - Whole program cannot run faster than critical path
 - Race conditions, mutual exclusion
 - Race condition is when one thread interferes with the expected state in another thread
 - Each of multiple threads does the same thing
 - Read a variable, increment that variable, write the new value

- For multiple threads, final value of the variable may be unpredictable
- Some threads read after others have written, some read the same value
- Mutual exclusion is a way of preventing threads from interfering
 - Simplest example, associate a lock, or monitor, with a variable
 - A thread must get the lock
 - Only the thread that has the lock can alter the variable
 - All other threads will wait, or block, until the lock becomes available
- Many different kinds of locks, outside the scope of this discussion
- Can still get into trouble
 - Dining philosophers problem
 - Each philosopher must get the chopsticks to either side at a roundtable
 - Eventually, all diners will be blocked waiting on someone else, and so on
 - Referred to as a deadlock
- Software approaches based on different memory models
 - Shared memory
 - All the threads access the same memory
 - Multiple core systems are shared memory
 - Have to deal with race conditions, mutual exclusion directly
 - Distributed memory
 - Each thread has its own memory, separate from all other threads
 - How do you coordinate work?
 - Threads pass messages to each other
 - Clusters, grids do this of necessity
 - Ultimate goal is automatic parallelization
 - What if your compiler could handle locking bits of memory?
 - Would have to be able to model out all the thread interactions
 - Non-trivial problem
 - Would save programmers having to understand, all programs would benefit
- With multiple cores on the rise
 - More programmers need to be aware of multi threaded
 - If keep programs single threaded, won't benefit since clock frequencies have stabilized
 - Program will be relegated to a single core
 - Thinking in threads is difficult, harder to predict what will happen at what time
 - Naive assumptions can lead to disastrous results
 - No easy answers, this is an active area of research, still

Offset

Topic

- For example, Java has had threading all along
 - Still got some huge improvements, very recently
 - Based on work of Doug Lea at Oswego, others
- 2 and 4 cores are common, now
- Like frequency scaling, we'll just keep getting more cores
- Desire for automatic parallelization tools will only grow

26:29

• **Outro**

- Contact me
 - Email to feedback@thecommandline.net
 - Web site at <http://thecommandline.net/>
 - IM to command.line@skype
 - Listener comment line is 240-949-2638
 - del.icio.us tag is "for:cmdln"
 - <http://twitter.com/cmdln>
- I'd like to thank libsyn.com for AAC hosting and Wouter de Bie for MP3 hosting
- These notes and the show audio and music are covered by a Creative Commons license
 - <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>
 - Attribution, non-commercial, share alike