CONCURRENCY AND THREADING

George Porter
Module 1
Fall 2020

UC San Diego

ATTRIBUTION

- Content from “Java in a Nutshell”, Ivan Vazquez, and Rick Snodgrass

UC San Diego

CONCURRENCY VS PARALLELISM

- Both deal with doing a lot at once, but aren't the same thing
  - Given set of tasks $\{T_1, T_2, \ldots, T_n\}$
- Concurrency:
  - Progress of multiple elements of the set *overlap in time*
- Parallelism:
  - Progress on elements of the set occur *at the same time*

CONCURRENCY

- Might be parallel, might not be parallel

- A single thread of execution can time slice a set of tasks to make partial progress over time
  - Time 0: Work on first 25% of Task 0
  - Time 1: Work on first 25% of Task 1
  - Time 2: Work on first 25% of Task 2
  - Time 3: Work on first 25% of Task 3
  - Time 4: Work on second 25% of Task 0
  - Time 5: Work on second 25% of Task 1
  - …

PARALLELISM

Multiple execution units enable progress to be made simultaneously

Processor 1

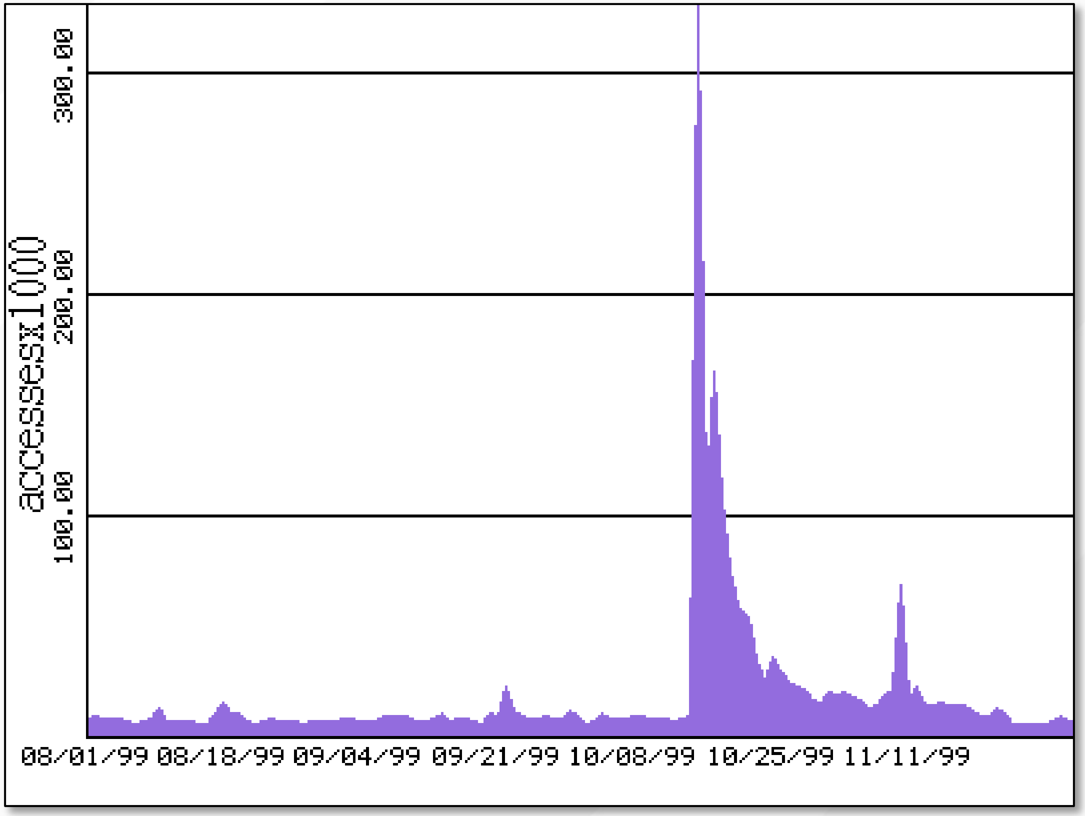
- Time 0: 1st 25% of Task1
- Time 1: 2nd 25% of Task1
- Time 2: 3rd 25% of Task1
- Time 3: 4th 25% of Task1
- Time 4: 1st 25% of Task3

Processor 2

- Time 0: 1st 25% of Task2
- Time 1: 2nd 25% of Task2
- Time 2: 3rd 25% of Task2
- Time 3: 4th 25% of Task2
- Time 4: 1st 25% of Task4

FLASH TRAFFIC

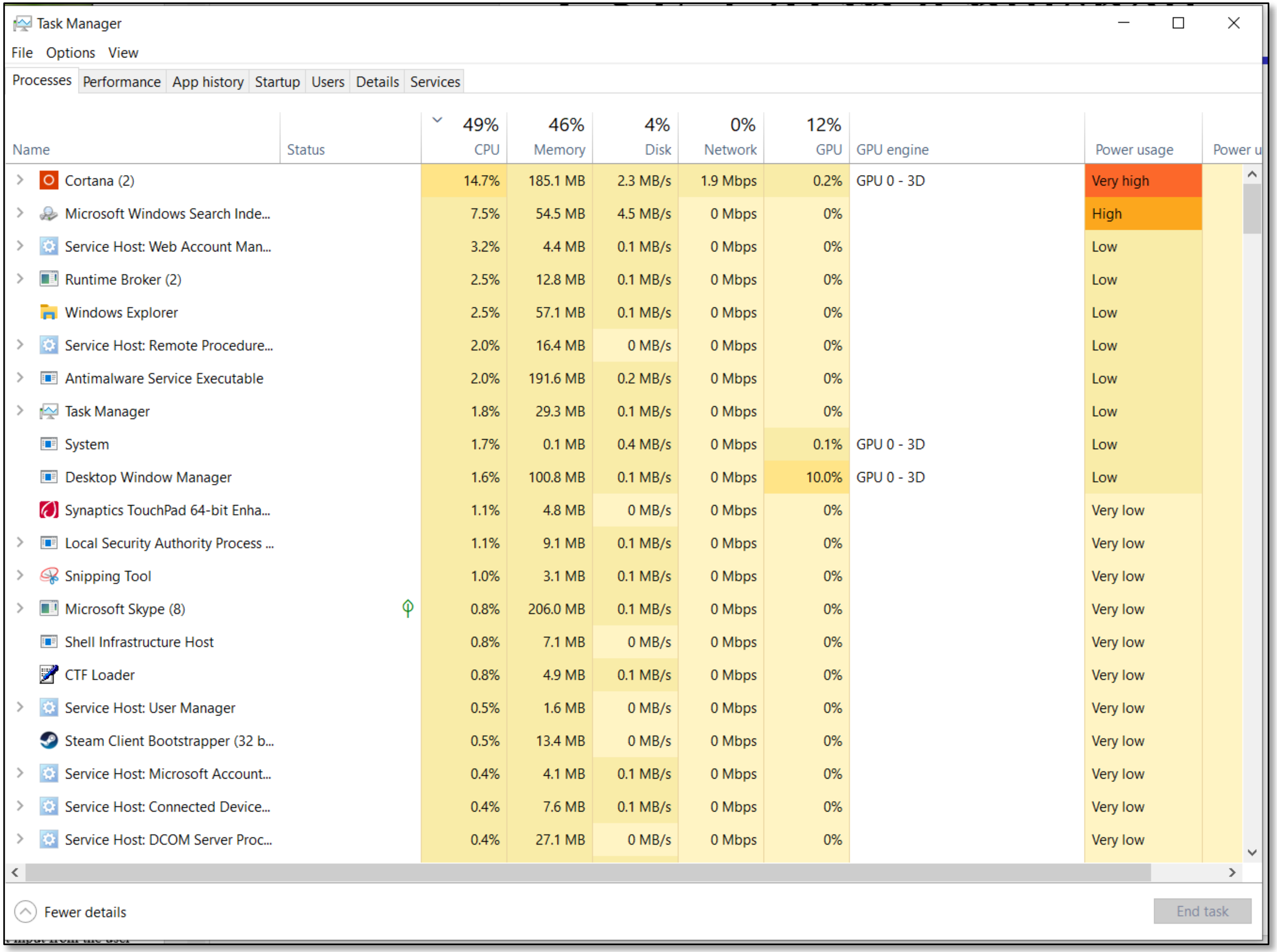
- USGS Pasadena, CA office Earthquake site
- Oct 16, 1999 earthquake



OS in a “nutshell”

- Define: a process, a thread, parallelism, concurrency
- How to use threads for concurrency (this class)
  - Not how to implement threads (CSE 120/221)
- Not covered: Sharing state between threads

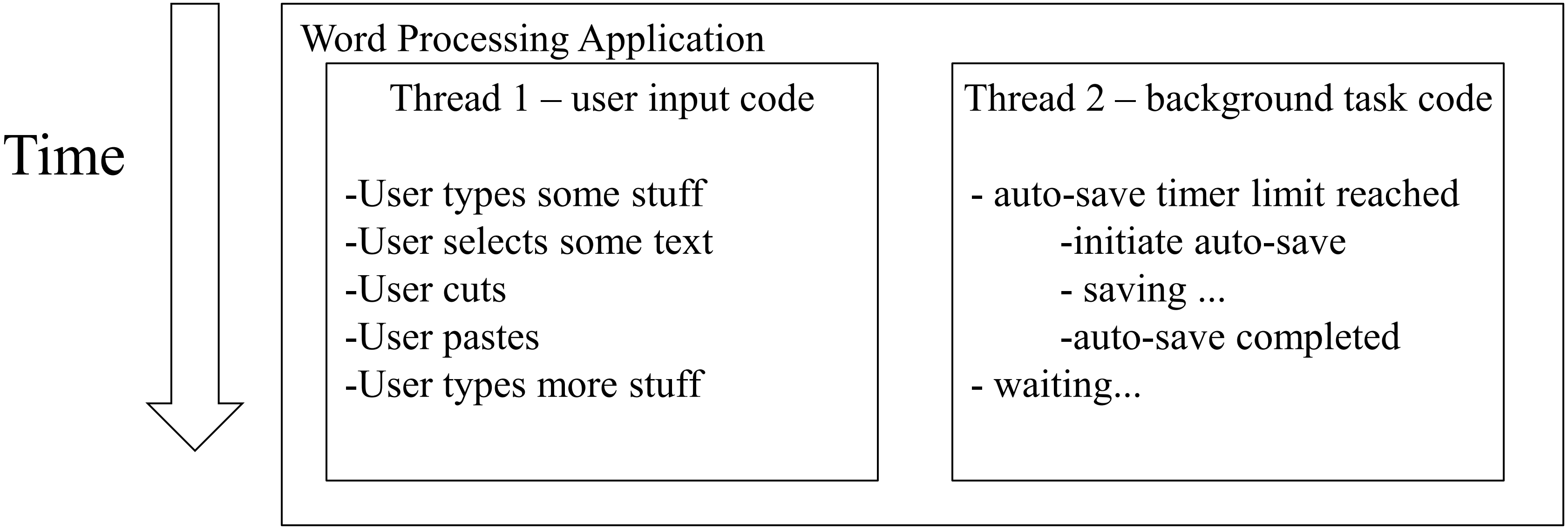
Many processes running at a time


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

Name	Status	49% CPU	46% Memory	4% Disk	0% Network	12% GPU	GPU engine	Power usage	Power u
Cortana (2)		14.7%	185.1 MB	2.3 MB/s	1.9 Mbps	0.2%	GPU 0 - 3D	Very high	
Microsoft Windows Search Inde...		7.5%	54.5 MB	4.5 MB/s	0 Mbps	0%		High	
Service Host: Web Account Man...		3.2%	4.4 MB	0.1 MB/s	0 Mbps	0%		Low	
Runtime Broker (2)		2.5%	12.8 MB	0.1 MB/s	0 Mbps	0%		Low	
Windows Explorer		2.5%	57.1 MB	0.1 MB/s	0 Mbps	0%		Low	
Service Host: Remote Procedure...		2.0%	16.4 MB	0 MB/s	0 Mbps	0%		Low	
Antimalware Service Executable		2.0%	191.6 MB	0.2 MB/s	0 Mbps	0%		Low	
Task Manager		1.8%	29.3 MB	0.1 MB/s	0 Mbps	0%		Low	
System		1.7%	0.1 MB	0.4 MB/s	0 Mbps	0.1%	GPU 0 - 3D	Low	
Desktop Window Manager		1.6%	100.8 MB	0.1 MB/s	0 Mbps	10.0%	GPU 0 - 3D	Low	
Synaptics TouchPad 64-bit Enha...		1.1%	4.8 MB	0 MB/s	0 Mbps	0%		Very low	
Local Security Authority Process ...		1.1%	9.1 MB	0.1 MB/s	0 Mbps	0%		Very low	
Snipping Tool		1.0%	3.1 MB	0.1 MB/s	0 Mbps	0%		Very low	
Microsoft Skype (8)		0.8%	206.0 MB	0.1 MB/s	0 Mbps	0%		Very low	
Shell Infrastructure Host		0.8%	7.1 MB	0 MB/s	0 Mbps	0%		Very low	
CTF Loader		0.8%	4.9 MB	0.1 MB/s	0 Mbps	0%		Very low	
Service Host: User Manager		0.5%	1.6 MB	0 MB/s	0 Mbps	0%		Very low	
Steam Client Bootstrapper (32 b...		0.5%	13.4 MB	0 MB/s	0 Mbps	0%		Very low	
Service Host: Microsoft Account...		0.4%	4.1 MB	0.1 MB/s	0 Mbps	0%		Very low	
Service Host: Connected Device...		0.4%	7.6 MB	0.1 MB/s	0 Mbps	0%		Very low	
Service Host: DCOM Server Proc...		0.4%	27.1 MB	0 MB/s	0 Mbps	0%		Very low	

What Are Threads?

- As an example program using threads, a word processor should be able to accept input from the user and at the same time, auto-save the document.
- The word processing application contains two threads:
  - One to handle user-input
  - Another to process background tasks (like auto-saving).



Time

Word Processing Application

Thread 1 – user input code

-User types some stuff
-User selects some text
-User cuts
-User pastes
-User types more stuff

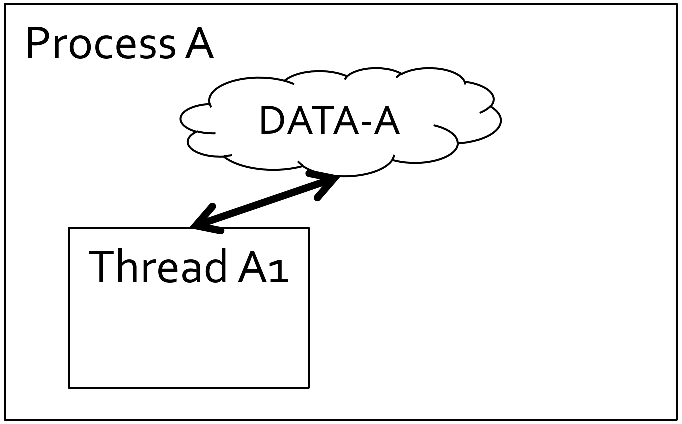
Thread 2 – background task code

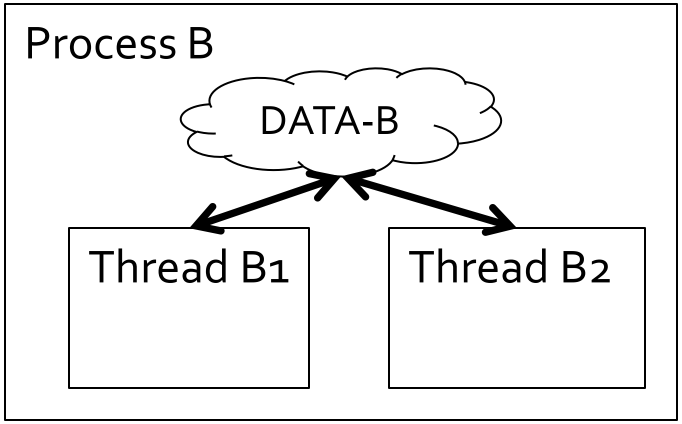
- auto-save timer limit reached
    -initiate auto-save
    - saving ...
    -auto-save completed
- waiting...

Programming Perspective

- Every process on your server/machine has:
  - A virtual memory address space
    - Your program’s heap, code, global variables
  - One or more “threads of control” (or just “threads”)
    - Each one consists of:
      - Its own local program counter
      - Its own local stack

- Threads run on a CPU (or CPU “core”)
- The OS schedules threads (puts them on the CPU)
  - And deschedules them (takes them off the CPU)
- Your “main” function runs in a thread
  - You’ve already been programming using threads!





UC San Diego