

Graduate Operating Systems

Spring 2022

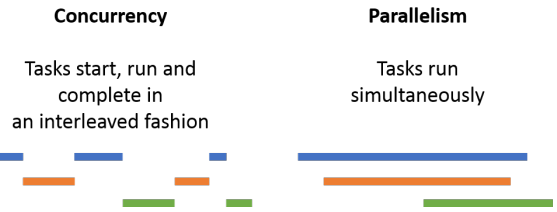
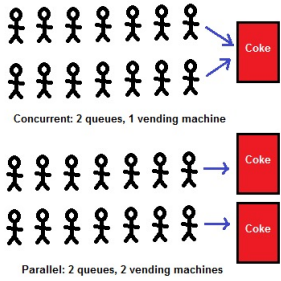
1

Today's Papers

- **[13]** D. Stein and D. Shah, "Implementing Lightweight Threads", Proc. of USENIX, San Antonio, TX, June 1992.
- **[14]** John Ousterhout, "Why Threads are a Bad Idea (for most purposes)", talk given at USENIX Annual Conference, September 1995.

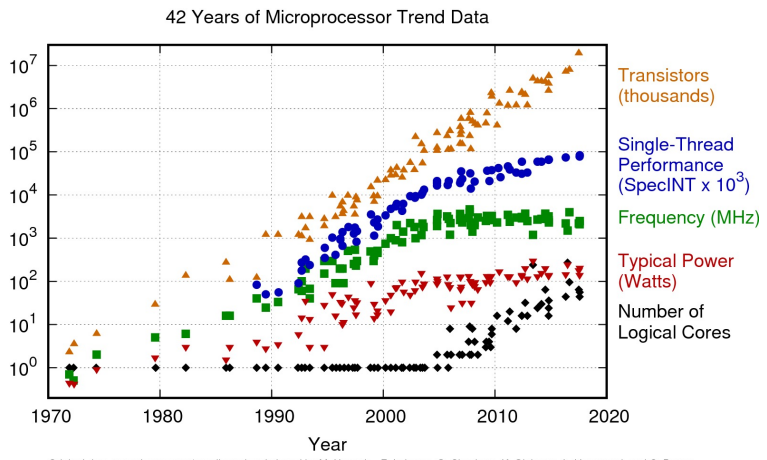
2

Concurrency vs. Parallelism



3

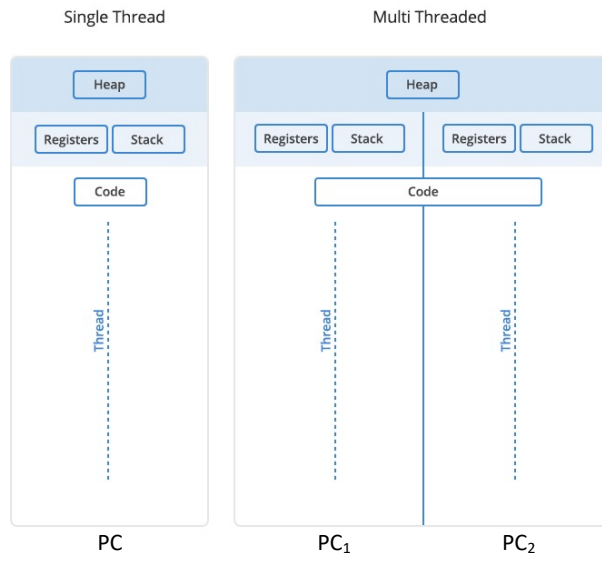
Microprocessor Trends



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

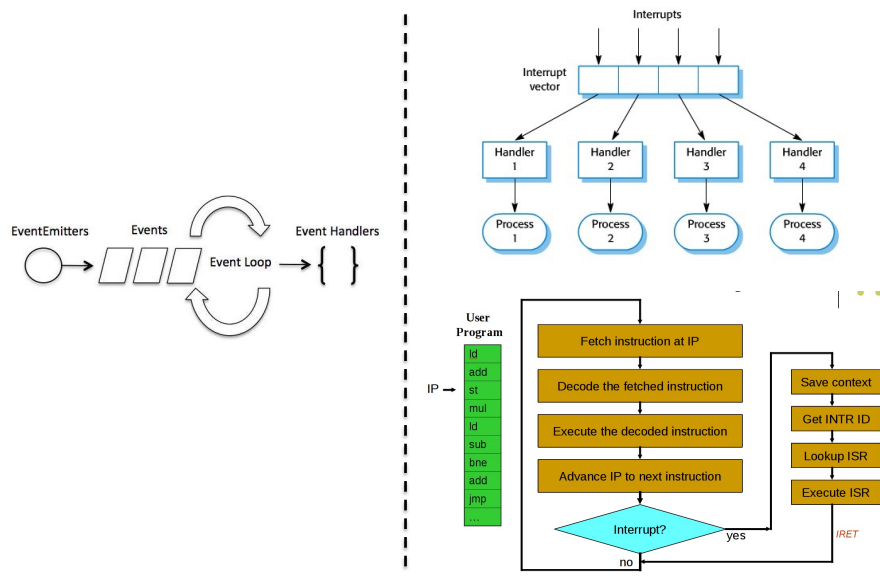
4

Processes vs. Threads



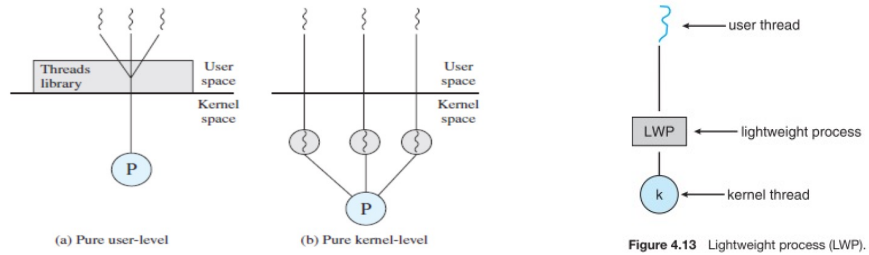
5

Events



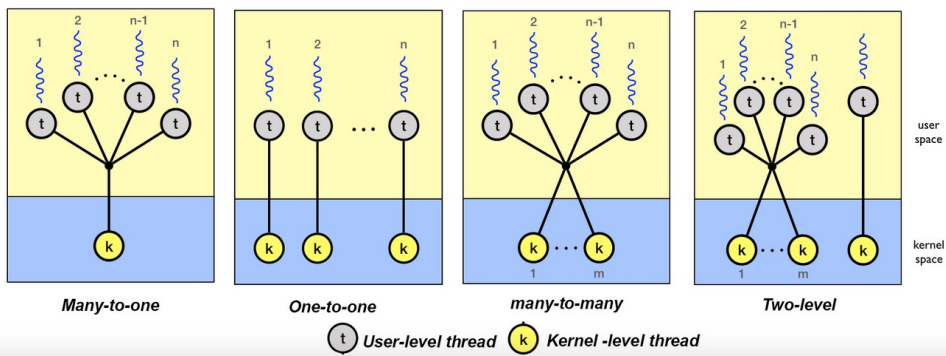
6

Types of Threads



7

Thread Models



8

Paper Discussion

- Why are threads cheaper than processes?
- How is IPC performed using threads?
- Why is synchronization between threads needed?
- Two creation approaches: create ALL threads or create only CALLING thread; difference?
- What is “thread-local storage”?
- What are bound threads and why are they useful?
- Why is signaling challenging?

9

Pthreads (POSIX 1003.1c)

```

#include <stdio.h>
#include <pthread.h>
void printMsg(char* msg) {
    int status = 0;
    printf("%s\n", msg);
    pthread_exit(&status);
}
int main(int argc, char** argv) {
    pthread_t thrdID;
    int* status = (int*)malloc(sizeof(int));
    printf("creating a new thread\n");
    pthread_create(&thrdID, NULL, (void*)printMsg, argv[1]);
    printf("created thread %d\n", thrdID);
    pthread_join(thrdID, &status);
    printf("Thread %d exited with status %d\n", thrdID, *status);
    return 0;
}

```

10

Common Programming Models

Multi-threaded programs tend to be structured as:

- **Producer/consumer**
Multiple producer threads create data (or work) that is handled by one of the multiple consumer threads
- **Pipeline**
Task is divided into series of subtasks, each of which is handled in series by a different thread
- **Defer work with background thread**
One thread performs non-critical work in the background (when CPU idle)

11

Threads vs. Events

- *What is biggest problem with threads (in reading assignment)?*
- Threads:
 - Independent execution streams
 - Preemptive scheduling
 - Synchronization
 - Deadlocks
 - Debugging
 - “Threads break abstraction”
 - Getting good performance
 - OS support of threads

12

Threads vs. Events

- Events:
 - No CPU concurrency
 - Callbacks; event handlers
 - No preemption
 - Long-running handlers
 - State across handler invocations
 - Debugging
 - Overheads
 - Portability