CSCE 313-200
Introduction to Computer Systems
Spring 2025

Final Notes

Dmitri Loguinov
Texas A&M University

April 22, 2025

Homework #3

- Tested Rabin-Karp performance on enwiki-all.txt
  - FILE_FLAG_NO_BUFFERING, B = 2 MB, 50 slots
  - 8-core Skylake-X server w/RAID-50 @ 4 GB/s

	keywords-B			keywords-D	
	Time	Speed	Found	Speed	Found
1	11.19	2.7 GB/s	319,017,279	27 MB/s	3,374,677,735
2	11.27	2.7 GB/s	319,017,279	40 MB/s	3,374,677,735
3	12.09	2.5 GB/s	319,017,279	47 MB/s	3,374,677,735
4	14.13	2.1 GB/s	319,016,884	50 MB/s	-920,294,583
5	17.28	1.8 GB/s	319,017,279	28 MB/s	/
6	23.09	1.3 GB/s	319,017,279	27 MB/s	1,045,494,283
Ref	7.50	4.0 GB/s	319,017,279	125 MB/s	3,374,677,735

InterlockedMultiply

```
// new function
LONG __cdecl InterlockedMultiply(
  __inout  LONG volatile *Target,
  __in     LONG Value
);
```

- Write code for an interlocked multiply that does not use mutexes (lock-free)
  - Multiple threads might be calling this function at the same time
- Idea: grab the target, multiply locally, then try to swap back into shared space
  - Main caveat is another thread might have changed the target between our read and write

```
// standard functions in Windows
LONG __cdecl InterlockedExchange(
  __inout  LONG volatile *Target,
  __in     LONG Value
);
PVOID __cdecl InterlockedExchangePointer(
  __inout  PVOID volatile *Target,
  __in     PVOID Value
);
LONG __cdecl InterlockedCompareExchange(
  __inout  LONG volatile *Destination,
  __in     LONG Exchange,
  __in     LONG Comparand
);
PVOID __cdecl
InterlockedCompareExchangePointer(
  __inout  PVOID volatile *Destination,
  __in     PVOID Exchange,
  __in     PVOID Comparand
);
```

Scheduling

- Chapters 9-10 (scheduling), 14 (networking) were not covered in this class
- Some of this material discussed in chapters 2-3
  - Ready, blocked, running, suspended process states
  - Dispatcher admitting and swapping processes
- Main algorithms of chapter 9:
  - First-come, first-served (FCFS): same as FIFO, no preemption (i.e., each process executes to completion)
  - Round-robin (RR): assign fixed time slice to each process, preempt after the slice, run the next process in line
  - Weighted RR (WRR): similar to RR, but assign weights to processes based on their type, then set slice time proportional to weights

Scheduling

- Algorithms (cont'd)
  - Strict priority: multiple queues for different priority classes, serve class i only when all higher-priority queues are empty
  - Shortest process next (SPN): run process with the shortest estimated duration of execution D, no preemption
  - Shortest remaining time (SRT): preemptive version of SPN
  - Highest response ratio next (HRRN): response ratio is computed as w / D, where w is the current wait time
- Main issue: difficult to estimate D ahead of time
- Feedback policy: gradually penalize long processes
  - Process starts at highest priority, but after fixed intervals of CPU time, its priority drops by one class
  - Eventually, all long processes are in the idle class

Scheduling

more in CSCE 410

- In user space, process scheduling isn't typically feasible or useful since the OS does it better
- However, many other areas involve similar concepts
  - Amazon gets millions of requests per second, in which order to serve them to minimize response time?
  - Airport gate assignment to minimize wait time, transfer delay
- Chapter 10 deals with multi-CPU scheduling
  - More complex issue related to RAM/cache locality
  - Chapter also covers real-time scheduling to guarantee hard upper bounds on slice duration
- Even more general is distributed system scheduling
  - Jobs running on multiple hosts in parallel

Networking

more in CSCE 463

- Networks use sockets to interface with applications
  - Kernel APIs to open connections, transfer data
- Programming sockets is fairly easy, the interesting aspect are the underlying protocols
  - HTTP, DNS, SMTP, FTP, POP3, P2P: application layer
  - TCP/UDP: transport layer
  - IP: network layer
  - Ethernet, 802.11 wireless: data-link layer
- Homework similar to this class, multi-threaded C++
  - STL is allowed, programming should be simpler than here
  - CSCE 315 isn't needed, although listed as a prereq