CSCE 313-200
Introduction to Computer Systems
Spring 2024

Practice II

Dmitri Loguinov
Texas A&M University

March 6, 2024

Performance

- Example 2: unbounded producer-consumer
- Producer batch = 1
  - Q.size() ≤ 1
- Producer batch = 10
  - Q.size() $\rightarrow \infty$
- PC 1.1
  - Busy spins to enter
  - CPU is high, mostly spent in the kernel
  - Worst method in our comparison

```
int batch;          // PC 1.1
while (true) {
          while (true) {
              WaitForSingleObject(mutex, INFINITE);
              if (Q.size() > 0) {
                    x = Q.pop ();
                    break;
              }
              ReleaseMutex (mutex);
          }
          ReleaseMutex (mutex);
          // now produce
          WaitForSingleObject(mutex, INFINITE);
          for (int i=0; i < batch; i++)
                    Q.add (i+x);
          ReleaseMutex (mutex);
}
```

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
660/sec	187K/sec	**worse**	**worse**

Performance

- PC 1.2 sleeps on semaphore
  - CPU = 20%
- PC 1.4 releases semaphore in bulk
  - Speed-up by 40% over PC 1.2 with batch=10
  - CPU = 20%

```
int batch;          // PC 1.2
while (true) {
        WaitForSingleObject(sema, INFINITE);
        WaitForSingleObject(mutex, INFINITE);
        x = Q.pop ();
        ReleaseMutex (mutex);

        WaitForSingleObject(mutex, INFINITE);
        for (int i=0; i < batch; i++) {
                Q.add (i+x);
                ReleaseSemaphore(sema,1,NULL);
        }
        ReleaseMutex (mutex);
}
```

```
int batch;          // PC 1.4
while (true) {
        WaitForSingleObject(sema, INFINITE);
        WaitForSingleObject(mutex, INFINITE);
        x = Q.pop ();
        ReleaseMutex (mutex);

        WaitForSingleObject(mutex, INFINITE);
        for (int i=0; i < batch; i++)
                Q.add (i+x);
        ReleaseMutex (mutex);
        ReleaseSemaphore(sema,batch,NULL);
}
```

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
275K/s	130K/s	223K/s	112K/s

PC 1.2

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
275K/s	182K/s	223K/s	151K/s

PC 1.4 (hw1)

Performance

- PC 2.1
  - Adds WaitAll
  - CPU = 100%
  - Horrible performance
  - PC 3.2-3.3 similar
- Back to 1.4
  - Over 450% faster than 1.4 for batch=10
  - CPU = 100%

```
HANDLE arr[] = {sema, mutex};     // PC 2.1
while (true) {
          WaitForMultipleObjects(2, arr, true,
                                 INFINITE);
          x = Q.pop ();
          ReleaseMutex (mutex);

          WaitForSingleObject(mutex, INFINITE);
          for (int i=0; i < batch; i++)
                    Q.add (i+x);
          ReleaseMutex (mutex);
          ReleaseSemaphore(sema,batch,NULL);
}
```

```
int batch;          // PC 1.4 with CS
while (true) {
          WaitForSingleObject(sema, INFINITE);
          EnterCriticalSection (&cs);
          x = Q.pop ();
          LeaveCriticalSection (&cs);

          EnterCriticalSection (&cs);
          for (int i=0; i < batch; i++)
                    Q.add (i+x);
          LeaveCriticalSection (&cs);
          ReleaseSemaphore(sema,batch,NULL);
}
```

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
27K/s	**27K/s**	**worse**	**worse**

PC 2.1

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
361K/s	850K/s	280K/s	1.1M/s

PC 1.4 w/CS

Performance

- PC 3.0
  - CPU = 100%
  - Breaks down when Q is persistently small
- PC 3.1
  - Uses kernel events, runs at 450K/s
- PC 3.4
  - CPU = 30%

```
CONDITION_VARIABLE cv;          // PC 3.0
while (true) {
    EnterCriticalSection (&cs);
    while ( Q.size() == 0 )
          SleepConditionVariable (&cv, &cs, ...);
    x = Q.pop ();
    LeaveCriticalSection (&cs);

    EnterCriticalSection (&cs);
    for (int i=0; i < batch; i++)
          Q.add (i+x);
    LeaveCriticalSection (&cs);
    WakeConditionVariable (&cv);
}
```

```
while (true) {      // PC 3.4 (variation)
    EnterCriticalSection (&cs);
    while (Q.size() == 0) {
       LeaveCriticalSection (&cs);
       Sleep (100);   // 100 ms
       EnterCriticalSection (&cs);
    }
    x = Q.pop ();
    LeaveCriticalSection (&cs);

    EnterCriticalSection (&cs);
    for (int i=0; i < batch; i++)
          Q.add (i+x);
    LeaveCriticalSection (&cs);
}
```

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
205K/s	5.9M/s	**78K/s**	7.1M/sec

PC 3.0

k = 600		k = 20K	
batch=1	batch=10	batch=1	batch=10
22M/s	5.9M/s	16.5M/s	7.5M/sec

PC 3.4 (hw2)