What is the coolest thing you have done with threads? [closed]

Asked 16 years, 3 months ago Modified 11 years, 9 months ago

Viewed 2k times

2

Just wondering

multithreading

Share

Improve this question

Follow

edited Sep 21, 2008 at 15:41

mmattax

27.6k ● 42 ● 117 ● 151

asked Sep 21, 2008 at 14:36

Nazgul

2,004 ● 4 ● 24 ● 34

25 Answers

Sorted by: Highest score (default)

30

I made them work as I wanted once ! That was cool!

Share Improve this answer Follow

answered Sep 21, 2008 at 14:38

Fernando Barrocal
13.1k ● 9 ● 45 ● 51

13

I don't know if this counts, but for me a working multithreaded software is fascinating in itself, not so much the purpose they achieve. You have like 10, 20, 100 workers working in your program with the same infrastructure (Singletons, files etc.). Having everything work in harmony with mutexes, semaphores, context switches etc. is wonderful to observe, like being a manager and your team is working perfectly together. You read the application log, see the threads cooperate for a common goal, and it's just great. Can anybody relate to this feeling?

Share Improve this answer Follow

answered Sep 21, 2008 at 15:00

Thorsten79
10.1k ● 6 ● 40 ● 54

Totally agree with you! It's an addictive feeling! – Martin08
Sep 22, 2008 at 2:20

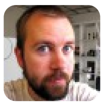
2 It's like looking at an ant farm that you programmed, just the most addictive thing one can have in terms of programming !!! – Gustavo Carreno Oct 2, 2008 at 13:02

6

I'm so useless with threads, I have to get my girlfriend to sew my buttons back on.

Share Improve this answer Follow

answered Sep 21, 2008 at 15:06



Dave Rutledge
5,535 ● 7 ● 29 ● 24

6

I'd love to say that I've cleverly parallelized an algorithm using lock-free data structures in order to get n-fold performance increase on an n-core processor. But I've never had a practical need, especially since most of my professional code has been for single-core systems.

Almost every time I've used more than one thread, in any language, it has been one of two reasons:

- the system (or a third party) offers a blocking API and I need an asynchronous one (or at least to let several ops run at once).
- to take advantage of pre-emptive priority-based scheduling to keep everything nice and responsive without having to chop all my slow operations into tiny pieces by hand.

Necessary, but not what you'd call glamorous.

Share Improve this answer Follow

edited Sep 22, 2008 at 22:42

answered Sep 21, 2008 at 14:47

Steve Jessop

279k ● 40 ● 469 ● 709

3

We once wrote a multithreaded application that essentially read a file line by line and did a lookup in an internal database to see if there was a match, appended some data and moved to the next line. The complexity though was that there could be multiple files processed at the same time, and multiple records per file could be searched. There was a manager class that knew how many threads were available and was responsible for divding out available worker threads to each file (if there was only one file to be processed it would recieve all 40 threads, if there were 5 files, depending on priority, each would recieve a fraction of those 40). We used Async delegates at first but noticed it would be hard to catch any exceptions that may occur in the async threads so we used the traditional thread start in .net.

The key to this was having a collection of ManualResetEvents in the manager class that was comprised of ManualResentEvents that were public proprties in the worker classes(threads). When a worker thread would finish it would signal it's ManualResetEvent which would be picked up by the .WaitAny() on the Manager class. The manager would then know that one of the threads was finished and would start a new one. In reality it was a little more complex than this but this was the core of what it did.

The hard part was unit testing this to ensure that at any given times the correct number of threads were running. We had tests that would act as if there was only one file in queue (gets all 40 threads), then another file was introduced and the allocation of threads would have to cycle down to 20 a piece for the two files. We had "Mock Objects" that essentially had a thread sleep parameter that we would pass a value for (in ms) to control how long each thread would take to process so we would have a good idea of when to do our assertion or interrogate the file processor to see how many threads, records, it was currently processing. There were also tests that would have two files running with 20 threads a piece, then one file would finish and as all the record threads would finish on the first file they would be reallocated to the second file to help it finish faster.

I'm sure this isn't the clearest explanation of what we actually did, I really need to write a blog post about it. If anyone needs more information on it please contact me, I'll try to answer as best I can.

edited Sep 21, 2008 at 15:28

answered Sep 21, 2008 at 15:22

Marcus King
1,677 ● 3 ● 15 ● 20

3

I have recently been hired to help with a quite large and complicated multithreaded application running on Microsoft Windows systems, with reader/writer locking objects. That made it difficult to search for deadlocks, so I wrote a deadlock detection object executing in its own thread that was sent information messages (with PostThreadMessage) from the locking objects whenever they were attempting to lock, succeeded or failed to lock, and unlocked.

By looking up the different threads and the shared locks and their state in a truth table it was then possible to without any doubt pinpoint the cause and location of the deadlocks.

Share Improve this answer Follow

answered Sep 21, 2008 at 15:39

Stefan Rådström
933 ● 6 ● 7

2

I figured out that I can create a FIFO, with ONLY ONE WRITER AND ONLY ONE READER, without using any synchronization instruments.

(so a master-slave with 2*n FIFO .. without any mutex / semaphone !!)

If you have a long linked list, you don't need to synchronize for inserting at one end and removing at other end.

The trick is to keep always one element in the list (-; the code is really small

My pride was 'dented' when a hardware guy told me ..that's obvious (-:

Share Improve this answer Follow

answered Sep 21, 2008 at 15:50

user414441

1

One of the most interesting things I have done with threads was write a multi-threaded application to solve a maze.

While it's nothing ground breaking, it was definitely interesting.

Share Improve this answer Follow

answered Sep 21, 2008 at 14:37

Ian P

13k ● 6 ● 50 ● 70

1

I create a windows service to consult a bunch of RSS feeds and store the information retrieved in a database. Since the application can contain a lot of RSS feeds, a pool of threads queries every n times packets of RSS.

Like Thorsten79 comented, the most exciting part is watching your threads cooperate and work together as a team.

Share Improve this answer Follow

edited May 23, 2017 at 12:28

Community Bot
1 ● 1

answered Sep 21, 2008 at 15:45

jdecuyper
3,963 ● 9 ● 40 ● 51

1

Distributed link checking system for a web crawler. Since web crawling is a very easily threaded solution I don't know if that counts...

I did write an algorithm to crack DES when I was in college that ran on a custom 256 CPU machine at the University. That was pretty neat, but was really just a divide and conquer type of problem.

Share Improve this answer Follow

answered Sep 21, 2008 at 16:38

Jason Short
5,314 ● 1 ● 31 ● 44

Totally agree - crawlers are probably the simplest (and IMHO at least) most fun to write as threaded applications - it balances perfectly the I/O bound nature of each request with the ability to dramatically increase throughput. – stephbu Sep 23, 2008 at 4:47

0

Ian P, Mind to elaborate? AFAIK you don't need to use threading to solve a maze, unless, of course the maze is so complicated that the wait time becomes so unbearable that you have to add in status bar so that the users would not get bored and thought that your program hangs.

Share Improve this answer Follow

answered Sep 21, 2008 at 14:41

Graviton
83.2k ● 147 ● 439 ● 611

0

Oh -- they were not complicated mazes.

The mazes were defined in an array, similar to this:

```
String[] MazeArray = new String[5];

MazeArray[0] = "---X---X-------XF";
MazeArray[0] = "-X-X-X---XXXXXXX-";
MazeArray[0] = "-X-X-X-X-X---X---";
MazeArray[0] = "-X-X-X-XXX-X-X-X-";
MazeArray[0] = "SX---X-----X---X-";
```

I'd spawn a new worker thread when there is a fork in the path, and have that thread investigate that path. Then, through some basic logic, I could determine the shortest path, longest path, etc.

The example listed is obviously over simplified, but it should illustrate the point. It's a fun exercise, you should try it if you have a few minutes to spare.

Ian

Share Improve this answer Follow

answered Sep 21, 2008 at 14:48
Ian P
13k ● 6 ● 50 ● 70

So how did the other four rows of MazeArray look like? :) (ie. check your indices) – Antti Rasinen Sep 21, 2008 at 14:54

haha -- sorry, copy/paste won that one. – Ian P Sep 21, 2008 at 17:37

Hummm, quite interesting. Solving a problem that would be a typical recursion problem with threads. Quite amazing !! – Gustavo Carreno Oct 2, 2008 at 13:00

0

Well this is really vanilla, but it's a nice stepping stone toward more creative threading:

In upload-new-data-and-process-it-for-me type requests, the request thread accepts the data and throws it in a queue, and the user goes on their merry way. One or more background threads continually dequeues items and processes them in some way.

Share Improve this answer Follow

answered Sep 21, 2008 at 15:04
easeout
8,736 ● 5 ● 45 ● 51

0

A simple client server with asynchronous input/output.

Share Improve this answer

answered Sep 21, 2008 at 15:28

Follow

INS
10.8k ● 8 ● 61 ● 93

0

I wrote a small fcgi servlet model in C++ which allocated a new thread instantly for each new request.

If you don't think thats cool, you should see what happened when i pumped 3K req/s through it. I accidentally forgot to clean them up, and even thou they all self terminated and stopped actually using memory, they still consumed *addressing* and I had the app quicky reseve more memory than I had and cease to create threads.

(I was on 32bit at the time and it literally stopped after creating 2^32 threads. Goodness knows what it would do with 64bits)

Also, I created a multi-threaded (well, forking) breadth-first fork-on-directory replacement for the famous command `rm -rf` . Mainly I was frustrated with rm -rf seeming to wait for IO to respond with a yay/nay response, which made it slow on some directory structures (such as squid caches). The only real caveat of this code is it only had entertainment value, and if ever used on a whole filesystem, it would be a race between 2 scenarios:

1. Disk being wiped making it unusable, and possibly erasing the commands that permitted you to tell it to die.
2. The system "Fork Bombing" due to the massive fork rate and making it so highly operation intensive it no longer even *responds* to commands.

And in the case of "fork bombing" the massive spawn rate could result in the recursive rm stopping itself (or hit the Ulimit if any)

Share Improve this answer Follow

edited Sep 21, 2008 at 15:30

answered Sep 21, 2008 at 15:19

Kent Fredric
57.3k ● 14 ● 111 ● 151

0

I guess implementing multi-threading for dos was a coolest thing i done with threads ..

Share Improve this answer Follow

answered Sep 21, 2008 at 15:58

Ilya
3,138 ● 4 ● 24 ● 30

I created a threading library with lock-free intertask communication to simplify multithreading programming.

0

Delphi only: OmniThreadLibrary.

(To give credit where it is due - I didn't write the lock-free structures, GJ did.)

Share Improve this answer Follow

answered Sep 21, 2008 at 16:02

gabr
26.8k ● 10 ● 79 ● 142

0

I wrote an image filtering framework in Java that uses thread pools.

I was surprised how much faster the filters run in multiple parallel threads even on a single processor single core machine. When I find some free time, I want to actually figure out why that is; all I'm doing is accessing memory and mathematical computation.

Threads rock (as long as they don't lock.)

Kudos to Java thread pools, too.

Share Improve this answer Follow

answered Sep 21, 2008 at 17:35

lajos
25.7k ● 19 ● 66 ● 75

0

I wrote a multi-threading library for HDOS. HDOS was an 8-bit OS that ran on HeathKit systems. I intercepted the system clock tick (Every 55 millseconds, if I remember correctly) and had a scheduler that would decide which thread to run next in a round-robin fashion. Of course since the OS itself was not multi-threading, only one thread was allowed to be in the OS at any given time.

I never did anything useful with it. It was just a fun project I decided to tackle to see if I could do it.

Share Improve this answer Follow

answered Sep 22, 2008 at 1:45
Ferruccio
101k ● 38 ● 229 ● 303

0

I made an implementation for quick sort where the left and right sides of the partition are sorted concurrently. It was almost twice as fast as the same code using only 1 thread.

Share Improve this answer Follow

answered Sep 22, 2008 at 22:48
nahano
3,835 ● 5 ● 39 ● 57

0

I wrote a lock-free cache that's 10x faster than the most common, lock-based, caching library. It was really interesting to figure out how to do complex CASing logic and come up with an alternative to LRU that didn't suffer suffer locking overhead.

I also wrote a distributed master-worker framework, which I later prototyped abstractions to support fork-join and map-reduce. I need to redo those at some point, as they weren't production quality, but it was quite a fun diversion.

I'd really love to write a SkipList data structure, just to learn how. There's already a core implementation, but its such a cool and simple idea I'd love to dig into it. It would be purely throw away code, but educational.

Share Improve this answer Follow

answered Sep 23, 2008 at 4:27
Ben Manes
9,561 ● 3 ● 38 ● 40

0

This isn't an *application* of multi-threading, but a small snippet that shows off C# 3.0 features including lambdas and object initializers. Not what you had in mind, I'm sure—but the "cooleset thing [I've] done with threads" nonetheless.

```
new Thread(() =>
{
    // do stuff in a new thread's context
})
{
    Name = "Thread " + GetHashCode().ToString(),
    Priority = this.threadPriority
}
.Start();
```

Share Improve this answer Follow

edited Sep 30, 2008 at 10:23

answered Sep 30, 2008 at 10:18
core
33k ● 45 ● 140 ● 195

0

I wrote an implementation of actors/message passing for multithreading in C#, ever since I wrote that multithreading has been so easy! Writing that in itself was pretty fun, lockless data structures for message queues. There isn't a thread for each actor but instead a pool of threads which loops through the actors and runs each of them. Since multithreading is so easy now that I have this system to play with I've written several cool things:
A simple socket server/client demo app
A multithreaded webcrawler (I'm going to go back to that one)
A procedural content generator for games
and some other things, but they're all small and boring

Share Improve this answer Follow

answered May 16, 2009 at 22:54

Martin
12.4k ● 14 ● 66 ● 129

0

Multi-threaded calls to a (third party application) which does not support simultaneous calls from different apps or threads? Although multiple instances could be executed (and this is how I eventually implemented it), certain applications operations could not be executed simultaneously on different app instances.

Share Improve this answer Follow

answered Jan 4, 2011 at 21:15

winwaed
7,791 ● 6 ● 42 ● 84

0

See this parallel N-puzzle solver. It solves the Npuzzle problem by iterative deepening searching forking grains to implement the search. This is done in a parallel programming language I designed which makes it *easy* to fork grains; almost the whole magic in program is hiding in the "fork parallel" operator **(|| ...)** You have to look for it in the code.

Share Improve this answer

Follow

edited May 23, 2017 at 10:25

Community Bot

1 ● 1

answered Mar 16, 2013 at 18:03

Ira Baxter

95.2k ● 23 ● 181 ● 353