Q&A Session for Programming Languages Lecture 24

Session Number: 1206327222
Date: 2020-12-4
Starting time: 14:27

ANON - 14:30
Q: Will we be able to use late days on the final homework (HW7) since it's the end of the semester?
Priority: N/A
	Ana L. Milanova - 14:33
	A: Yes. The maximal number of late days for the homework is 2.

ANON - 14:39
Q: what is an "execution context"?
Priority: N/A
	Steven Haussmann - 14:40
	A: Context is made up of things like data and control flow -- where we are in the program and what information we have.

ANON - 14:40
Q: How is it possible to have multiple tasks doing stuff at the same time?
Priority: N/A
	Steven Haussmann - 14:41
	A: Even on just a single processor, you can have many separate, but related, threads of execution, all of which can be operating together.

ANON - 14:41
Q: So concurrency is the idea that the tasks may be active at the same time and parallelism is when two tasks are actually happening at the same time?
Priority: N/A
	Steven Haussmann - 14:41
	A: Yes, that's the distinction. So every parallel system is concurrent.
	Steven Haussmann - 14:42
	A: A single-core system running Linux is concurrent, because many things can be underway at once, even if only one thing ever executes at a time.
	Konstantin Kuzmin - 14:43
	A: Recall what we discussed in Computer Organization, when we talked about Chapter 6 material.
	Konstantin Kuzmin - 14:43

A: ? Hardware
? Serial: e.g., Pentium 4
? Parallel: e.g., quad-core Xeon e5345
? Software
? Sequential: e.g., matrix multiplication
? Concurrent: e.g., operating system

ANON - 14:42
Q: What is a thread? (I haven't taken Operating Systems)
Priority: N/A
Steven Haussmann - 14:44
A: "Thread" and "execution context" are pretty similar in meaning. A thread is a distinct execution path, and has its own state (but there could be data shared between threads!)

ANON - 14:42
Q: what is granularity?
Priority: N/A
Steven Haussmann - 14:45
A: It refers to what level the parallelism takes place at. So, a processor performing a vectorized addition operation is very fine-grained (it's just a few + operations); running a job on a distributed computing project is very coarse-grained.
Steven Haussmann - 14:47
A: Generally, fine-grained parallelism is simpler to think about. Using a vectorized addition operation doesn't require you to worry about synchronization and consistency; it's basically invisible to the programmer
Konstantin Kuzmin - 14:47
A: To add to what Steven said, recall what we discussed in Computer Organization, when we talked about Chapter 6 material. Specifically, slide 20 of part A in chapter 6 that gives an example of coarse MT, Fine MT, and SMT - different levels of granularity.

ANON - 14:44
Q: how are vector parallelism and thread parallelism different?
Priority: N/A
Steven Haussmann - 14:46
A: A vector operation performs the same operation on many pieces of data at once, which is more efficient that doing the operations individually -- look up AVX, SIMD, etc. for some more info. Thread parallelism has multiple execution contexts going at once.

ANON - 14:48
Q: what are the cons of processor abstration?
Priority: N/A

Steven Haussmann - 14:49
A: The slide was discussing abstraction in general. Most abstractions trade efficiency for convenience.
Ana L. Milanova - 14:50
A: The most common drawback of adding more layers of abstraction is poor performance.

ANON - 14:53
Q: do communication/syncrhonization refer to how threads "talk" to each other?
Priority: N/A
Steven Haussmann - 14:55
A: Yes, these concepts only come up when dealing with multiple threads of execution at once.

ANON - 14:55
Q: are there any classes that focus on code optimization and/or concurrent programming other than parallel computing?
Priority: N/A
Steven Haussmann - 14:56
A: There is a Distributed Systems class that covers distributed computing (i.e. many networked systems).
Konstantin Kuzmin - 14:57
A: CSCI 4510 - Distributed Systems and Algorithms, CSCI 4500 - Distributed Computing Over The Internet
Ana L. Milanova - 15:08
A: I don't think there is a class on code optimization for parallelization. But I have been working on some related research projects recently. If anyone is interested in a project or independent study in the area, let me know.

ANON - 14:57
Q: So cuncurrency = processes starting at the same time. Parallel = processes running at the same time?
Priority: N/A
Steven Haussmann - 14:59
A: Parallelism explicitly requires that two or more tasks can be actively executing at the same time. A concurrent system just has to have several tasks in existence.
Konstantin Kuzmin - 15:03
A: "Concurrent software can run on serial hardware, such as operating systems for the Intel Pentium 4 uniprocessor, or on parallel hardware, such as an OS on the more recent Intel Core i7. The same is true for sequential software.
Konstantin Kuzmin - 15:03
A: For example, the MATLAB programmer writes a matrix multiply thinking about it sequentially, but it could run serially on the

Pentium 4 or in parallel on the Intel Core i7",
        Konstantin Kuzmin - 15:04
        A: from our Computer Organization textbook, "Computer Organization And Design, 5th Edition 2014"

ANON - 15:03
Q: How does the runtime schedule these for us? (slide 16)
Priority: N/A
        Steven Haussmann - 15:06
        A: That's a bit of a dive into the JVM! Calling start() will cause VMThread to allocate a new thread; the virtual machine will run threads forever until they all exit or the program is forced to exit.

ANON - 15:06
Q: What are examples of atomic actions other than read or write?
Priority: N/A
        Steven Haussmann - 15:07
        A: Atomic operations generally combine a read and write (plus, possibly, some work on the data). For example, the AtomicInteger class provides an atomic increment operation, which reads and writes in one go.

ANON - 15:09
Q: So read and write are the basic operations and then everything else is just a more complicated version of that?
Priority: N/A
        Steven Haussmann - 15:10
        A: Reads and writes are, usually, inherently atomic -- only one thing happens. More complex atomic operations require careful design.

ANON - 15:14
Q: what exactly is the check-and-act data race describing?
Priority: N/A
        Steven Haussmann - 15:15
        A: Any situation where we check a piece of data before performing an action. This can cause problems in concurrent systems if the condition changes after we check it, possibly invalidating the action.

ANON - 15:18
Q: what are the advantages of syncrhonized Blocks?
Priority: N/A
        Ana L. Milanova - 15:20
        A: Synchronized blocks are Java's primary mechanism of

synchronization. Usually, we do need to synchronize relative order of operations of different threads for correctness.

Steven Haussmann - 15:23

A: They make it easy to control access to a critical section of code without having to deal with setting and checking mutexes yourself.

ANON - 15:22
Q: So the Instrict/Implicit Lock is always the this Object?
Priority: N/A

Steven Haussmann - 15:23

A: That's specifically in the case of a synchronized method, since it's just shorthand for a method whose entire body is synchronized on this

Ana L. Milanova - 15:24

A: Every object has an intrinsic lock that's associated with it. When we have a "synchronized method", then yes, the lock this method refers to is the lock of the "this" object.

ANON - 15:29
Q: why was the implementation of data not safe?
Priority: N/A

Konstantin Kuzmin - 15:39

A: Think about atomicity.

ANON - 15:29
Q: if locking "forces" thread B to wait for thread A to finish, how's that different from just performing the tasks sequencially?
Priority: N/A

Steven Haussmann - 15:31

A: If the two threads aren't fighting over access to a specific synchronized block, they can both do work at the same time (in a parallel system)

Steven Haussmann - 15:31

A: Even in a concurrent system, this can be useful: a thread that can't proceed can just pause itself and let something else run. Your OS does this when waiting for data to be read from disk, for example.

Konstantin Kuzmin - 15:35

A: Inroducing synchronization causes the code to execute more serially. In the extreme case where no work can be completed concurrently due to synchronization constraints, the execution will happen effectively sequentially.

ANON - 15:30
Q: Wait so the synchronizedList could still fail if this was modified?

I'm not sure of this answer and was wondering about this
Priority: N/A
        Steven Haussmann - 15:34
        A: Consider what synchronizedList guarantees: that all of its methods are marked as synchronized. Does that give you any guarantees if you need to call several methods?

ANON - 15:32
Q: Asyncio vs Executor framework?
Priority: N/A
        Steven Haussmann - 15:33
        A: if I'm not mistaken, these are from Python and Java, respectively; I'm not sure it'd be trivial to compare them!

ANON - 15:33
Q: Alright so the synchornized list would prevent two threads from writing at the same time only?
Priority: N/A
        Ana L. Milanova - 15:39
        A: The "synchronized" will make sure that data.contains and data.add are executed sequentially. Think about what happens if the check-and-act (if !data.contais(p) data.add(p)) is interrupted by another thread.

ANON - 15:36
Q: On the final exam, will we be expected to debug Java code for concurrency bugs?
Priority: N/A
        Ana L. Milanova - 15:40
        A: Yes. But we will see a good number of examples in lecture and in practice sets.

ANON - 15:39
Q: What exactly does the Exectuor Framework do?
Priority: N/A
        Ana L. Milanova - 15:41
        A: It is an abstraction over threads. It abstracts away the explicit creation of threads. The Executor creates and manages the threads.

ANON - 15:44
Q: What is "race freedom"?
Priority: N/A
        Ana L. Milanova - 15:44
        A: It refers to a program that is free of race conditions.

ANON - 15:44
Q: But if the schronized makes sure that the !data.contains() and data.add() execute sequentially, how can another method interuppt it?
Priority: N/A
Ana L. Milanova - 15:46
A: Another thread cannot interrupt either data.contains or data.add. But it can "interrupt" in between the execution of data.contains() and data.add().

ANON - 15:44
Q: How come different executions would make the bug happen sometimes but not other times?
Priority: N/A
Steven Haussmann - 15:45
A: The exact order in which concurrent operations happens is unpredictable. Synchronization is required to ensure ordering where it matters.
Steven Haussmann - 15:45
A: Importantly, that ordering might change when in a debugging context, due to breakpoints, timing differences, different memory layouts, etc.

ANON - 15:50
Q: how is there a data race on element count? I'm confused
Priority: N/A
Ana L. Milanova - 15:53
A: The definition of a data race states "there is a data race if two theads can access the same location simultaneously and at least one is a write". Here the write to elementCount in removeAllElements and the read in lastIndexOf(elem) can happen simultaneous

---

ANON - 15:57
Q: How can the code be interuppted between data.contains and data.add though? I thought the lock this would ensure that there are no interuppts between those two? Isn't that how synchronization works so that the critical section executes fully without interup
Priority: N/A
Ana L. Milanova - 15:58
A: That example is related to our discussion on atomicity in this part of lecture.
Ana L. Milanova - 15:59
A: Thread A releases the lock after running data.contains(p). Thread B manages to obtain the lock right after and runs its own data.contains(p), then releases the lock. A obtains it and runs data.add(p).

Ana L. Milanova - 16:00
A: The idea is that the check-and-act must be atomic.

ANON - 16:05
Q: I haven't been understanding the past few examples - what are good strategies to figure out where the bugs are?
Priority: N/A

ANON - 16:07
Q: Alright I think I understand, but how do you know that the lock i released after data.contains()
Priority: N/A

ANON - 16:10
Q: Is there possibly a typo in # 7 of HW 7. A is never defined. In the while loop, we add term (A[i]) to result, however we'e never told what A is, so I'm unsure if this is a typo since we cannot know for sure what compute returns without knowing A's contents
Priority: N/A
Ana L. Milanova - 16:23
A: There is no typo. A is never defined explicitly, however, implicitly, it is an array. For the answer, it is asking you to provide an expression in terms of the elements of A (possibly unevaluated), not a single value.

ANON - 16:10
Q: I don't see any error on slide 43?
Priority: N/A
Konstantin Kuzmin - 16:13
A: Check Lec24_Part3.mp4, timestamp 28:57.
Ana L. Milanova - 16:24
A: There is no mistake in the slide text, as far as I know. The mistake is in the video: "if synchronized (sb) {" should be "syncrhonized (sb) {".

ANON - 16:12
Q: Are the final exam preparation materials going to be posted in Submitty soon?
Priority: N/A
Ana L. Milanova - 16:25
A: Yes, my goal is to have them tonight or tomorrow morning. I'll spam the class with an email announcement once I have them posted.

ANON - 16:12
Q: Does that mean we do not return a specific value, since they want us to say what compute returns?
Priority: N/A

ANON - 16:13
Q: Could you explain task execution again?
Priority: N/A

ANON - 16:13
Q: Alright I finally understand it I think. So contains and add themselves are synchronized, but the Code that uses both with the check-and-act, would lead to the duplicate add like you mentioned above? So if that Code synchronized, then it would be correct?
Priority: N/A
Ana L. Milanova - 16:25
A: Yes, that is correct.

ANON - 16:17
Q: So Slide 42, if we changed the synchronized(this) to synchronized(seen) would everything be fine?
Priority: N/A
Ana L. Milanova - 16:25
A: Yes.

ANON - 16:17
Q: can the executor framework cause probelms like race conditions, deadlocks, etc.?
Priority: N/A
Ana L. Milanova - 16:28
A: The executor framework does not prevent those concurrency errors. It only manages the tasks the programmer has specified in their code. It's still the responsibility of the programmer to ensure that there are no dangerous data races, atomicity errors, etc.

ANON - 16:18
Q: are concurrency and multithreading related?
Priority: N/A
Konstantin Kuzmin - 16:19
A: Multithreading is one of the ways to achieve concurrent execution. Other examples are ILP, map-reduce,etc.

ANON - 16:21
Q: Is Map-Reduce Another Name For = Vector Paralellism?

Priority: N/A

Ana L. Milanova - 16:31

A: No. These are two different concepts. MapReduce refers to the paradigm for alrge scale distributed computing on the cloud. Vector parallelism refers to low-level parallelism via hardware support (vector instructions in hardware).