What is a deadlock?

Asked 16 years, 3 months ago Modified 1 year, 11 months ago

Viewed 174k times

207

When writing multi-threaded applications, one of the most common problems experienced are deadlocks.

My questions to the community are:

1. What is a deadlock?
2. How do you detect them?
3. Do you handle them?
4. And finally, how do you prevent them from occurring?

multithreading concurrency locking deadlock

Share

Improve this question

Follow

edited Feb 26, 2016 at 0:09

Levent Divilioglu

11.6k ● 5 ● 60 ● 110

asked Aug 29, 2008 at 15:56

bmurphy1976

31.1k ● 12 ● 34 ● 24

56 *You first, my dear.* – Voicu Oct 18, 2014 at 21:40

18 Answers Sorted by: Highest score (default)

266

A **lock** occurs when multiple processes try to access the same resource at the same time.

One process loses out and must wait for the other to finish.

A **deadlock** occurs when the waiting process is still holding on to another resource that the first needs before it can finish.

So, an example:

Resource A and resource B are used by process X and process Y

- X starts to use A.
- X and Y try to start using B
- Y 'wins' and gets B first
- now Y needs to use A
- A is locked by X, which is waiting for Y

The best way to avoid deadlocks is to avoid having processes cross over in this way. Reduce the need to lock anything as much as you can.

In databases avoid making lots of changes to different tables in a single transaction, avoid triggers and switch to optimistic/dirty/nolock reads as much as possible.

Share Improve this answer Follow

edited Aug 29, 2008 at 16:06

answered Aug 29, 2008 at 15:58

Keith

155k ● 82 ● 306 ● 446

14 I'm using process here as a generalisation, not specifically an OS Process. These could be threads, but could also be completely different applications, or database connections. The pattern is the same. – Keith Sep 23, 2008 at 14:42

1 Hi, given this scenario: Thread A locks resource A and having a long process. Thread B waiting to lock resource A. CPU time usage : 20%, can you consider that a deadlock situation? – rickyProgrammer Sep 28, 2017 at 16:37

3 @rickyProgrammer no, that's just a regular lock wait, though the difference is a little academic. B waiting on slow A is a lock, B waiting for A waiting for B is a deadlock. – Keith Sep 28, 2017 at 17:05

So deadlock is more of two processes with locked resources waiting for those resources to be released..
– rickyProgrammer Sep 28, 2017 at 17:46

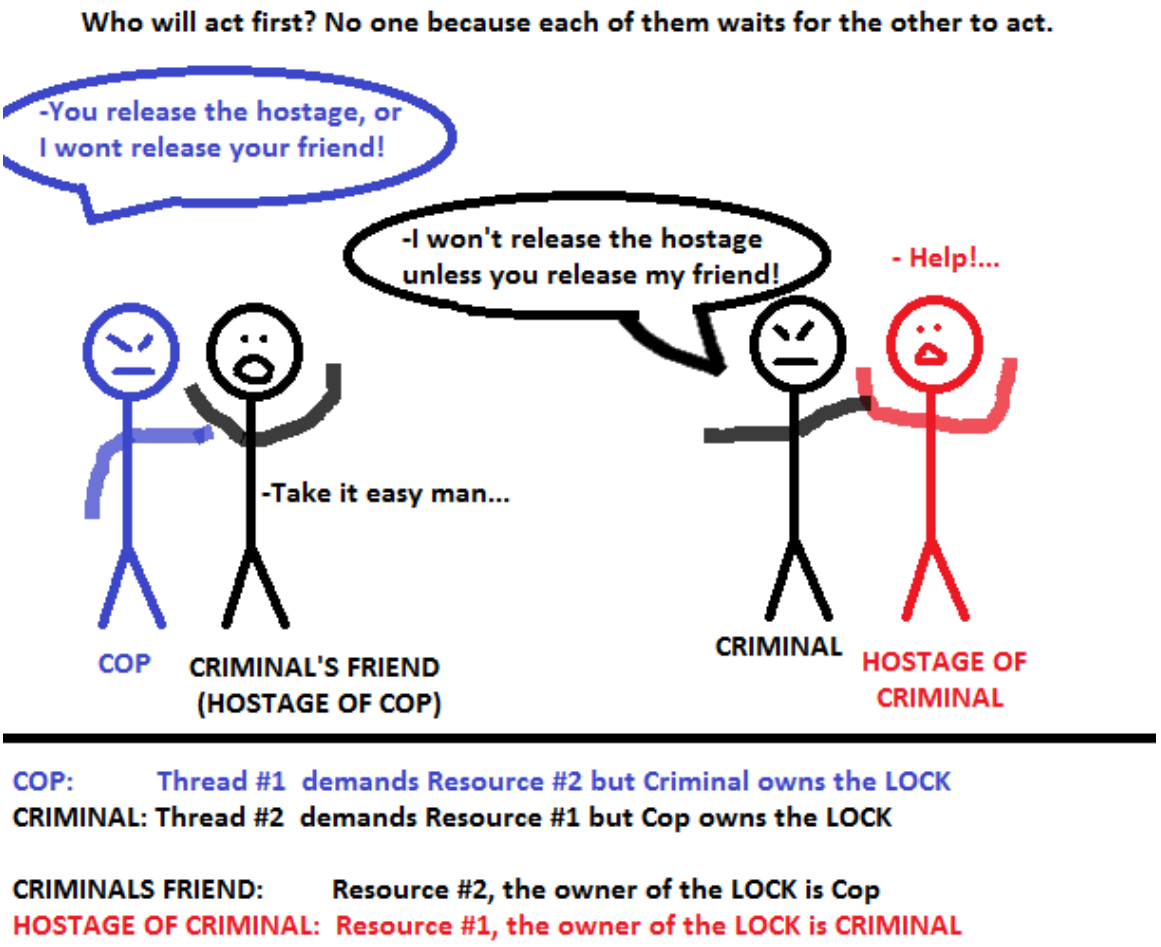
2 @rickyProgrammer it's a lock that won't become free, no matter how long you wait, because of the circular queue.
– Keith Sep 28, 2017 at 19:37

162

Let me explain a real world (not actually real) example for a deadlock situation from the crime movies. Imagine a criminal holds an hostage and against that, a cop also holds an hostage who is a friend of the criminal. In this

case, criminal is not going to let the hostage go if cop won't let his friend to let go. Also the cop is not going to let the friend of criminal let go, unless the criminal releases the hostage. This is an endless untrustworthy situation, because both sides are insisting the first step from each other.

Criminal & Cop Scene



So simply, when two threads needs two different resources and each of them has the lock of the resource that the other need, it is a deadlock.

Another High Level Explanation of Deadlock : Broken Hearts

You are dating with a girl and one day after an argument, both sides are heart-broken to each other and waiting for an **I-am-sorry-and-I-missed-you** call. In this situation, both sides want to communicate each other if and only if one of them receives an **I-am-sorry** call from the other. Because that neither of each is going to start communication and waiting in a passive state, both will wait for the other to start communication which ends up in a deadlock situation.

Share Improve this answer Follow

edited May 9, 2016 at 5:36

answered Feb 25, 2016 at 23:35

Levent Divilioglu
11.6k ● 5 ● 60 ● 110

Shoudnt the threads belong to different processes?, can threads belonging to the same process also cause a deadlock? – lordvcs Dec 10, 2017 at 11:41

1 @diabolicfreak It doesn't matter if the threads belong to the same process or not. – Sam Malayek May 15, 2018 at 5:15

4 Another example from real life could be four cars coming to the crossing of two equal roads in four directions simultaneously. Every one needs to give a way to a car from the right-hand side, so no one can proceed. – LoBo Jul 28, 2019 at 15:07

2 Those real life examples are very desriptive and also fun. – Soup Endless Feb 19, 2021 at 13:53

1 `Thread #1 demands Resource #2 but Criminal owns the LOCK` and `Resource #2, the owner of the LOCK`

`is Cop` sound like the captions are mixed up. – bravmi Mar 12, 2023 at 8:48

39

Deadlocks will only occur when you have two or more locks that can be aquired at the same time and they are grabbed in different order.

Ways to avoid having deadlocks are:

- avoid having locks (if possible),
- avoid having more than one lock
- always take the locks in the same order.

Share Improve this answer

Follow

answered Aug 29, 2008 at 16:19

Mats Fredriksson

20.1k ● 6 ● 38 ● 57

1 The 3rd point to prevent a deadlock (always take the locks in the same order) is vital, which is rather easy to be forgotten in coding practice. – Qiang Xu Oct 13, 2012 at 17:12

13

A deadlock happens when a thread is waiting for something that never occurs.

Typically, it happens when a thread is waiting on a mutex or semaphore that was never released by the previous owner.

It also frequently happens when you have a situation involving two threads and two locks like this:

```
Thread 1                Thread 2

Lock1->Lock();          Lock2->Lock();
WaitForLock2();         WaitForLock1();   <-- Oops!
```

You generally detect them because things that you expect to happen never do, or the application hangs entirely.

Share Improve this answer Follow

edited Aug 29, 2008 at 16:16

answered Aug 29, 2008 at 16:04

17 of 26

27.4k ● 13 ● 68 ● 85

1 A deadlock happens when a thread is waiting for something that *cannot* occur. – user207421 Feb 3, 2014 at 5:36

5

You can take a look at this wonderful articles, under section **Deadlock**. It is in C# but the idea is still the same for other platform. I quote here for easy reading

> A deadlock happens when two threads each wait for a resource held by the other, so neither can proceed. The easiest way to illustrate this is with two locks:

```
object locker1 = new object();
object locker2 = new object();
```

```
new Thread (() => {
                    lock (locker1)
                    {
                      Thread.Sleep (1000);
                      lock (locker2);      //
Deadlock
                    }
                  }).Start();
lock (locker2)
{
  Thread.Sleep (1000);
  lock (locker1);                          //
Deadlock
}
```

Share Improve this answer

Follow

answered Sep 25, 2013 at 6:45

onmyway133

47.9k • 32 • 266 • 271

4

Deadlock is a common problem in multiprocessing/multiprogramming problems in OS. Say there are two processes P1, P2 and two globally shareable resource R1, R2 and in critical section both resources need to be accessed

Initially, the OS assigns R1 to process P1 and R2 to process P2. As both processes are running concurrently they may start executing their code but the PROBLEM arises when a process hits the critical section. So process R1 will wait for process P2 to release R2 and vice versa... So they will wait for forever (DEADLOCK CONDITION).

A small ANALOGY...

> Your Mother(OS),
> You(P1),
> Your brother(P2),
> Apple(R1),
> Knife(R2),
> critical section(cutting apple with knife).
>
> Your mother gives you the apple and the knife to your brother in the beginning.
> Both are happy and playing(Executing their codes).
> Anyone of you wants to cut the apple(critical section) at some point.
> You don't want to give the apple to your brother.
> Your brother doesn't want to give the knife to you.
> So both of you are going to wait for a long very long time :)

Share Improve this answer Follow

edited Oct 4, 2018 at 17:48

answered Feb 4, 2016 at 19:29

Rohit Singh
18.1k ● 8 ● 96 ● 95

3

Deadlock occurs when two threads aquire locks which prevent either of them from progressing. The best way to avoid them is with careful development. Many embedded

systems protect against them by using a watchdog timer (a timer which resets the system whenever if it hangs for a certain period of time).

Share Improve this answer

Follow

answered Aug 29, 2008 at 15:59

Joseph Sturtevant

13.4k ● 12 ● 77 ● 90

2

A deadlock occurs when there is a circular chain of threads or processes which each hold a locked resource and are trying to lock a resource held by the next element in the chain. For example, two threads that hold respectively lock A and lock B, and are both trying to acquire the other lock.

Share Improve this answer

Follow

answered Feb 3, 2014 at 5:38

user207421

311k ● 44 ● 320 ● 488

I vote up to you. Your answer is more concise then above because they make confuse deadlock happen by process or thread. Some one say process, some one say thread :)
– Hai Nguyen Sep 3, 2017 at 5:45

2

Lock-based concurrency control

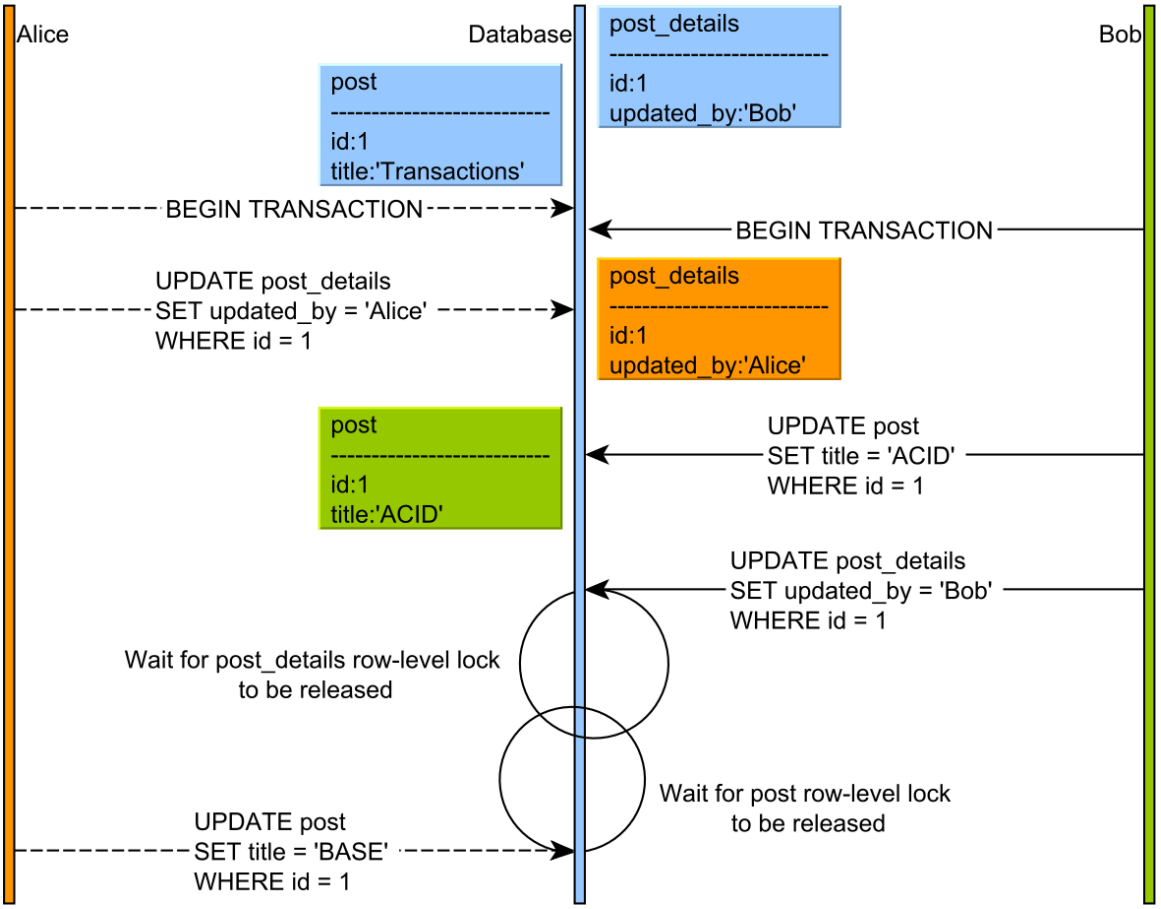
Using locking for controlling access to shared resources is prone to deadlocks, and the transaction scheduler alone cannot prevent their occurrences.

For instance, relational database systems use various locks to guarantee transaction `ACID` properties.

No matter what relational database system you are using, locks will always be acquired when modifying (e.g., `UPDATE` or `DELETE`) a certain table record. Without locking a row that was modified by a currently running transaction, `Atomicity` would be compromised).

What is a deadlock

A deadlock happens when two concurrent transactions cannot make progress because each one waits for the other to release a lock, as illustrated in the following diagram.



Because both transactions are in the lock acquisition phase, neither one releases a lock prior to acquiring the next one.

Recovering from a deadlock situation

If you're using a Concurrency Control algorithm that relies on locks, then there is always the risk of running into a deadlock situation. Deadlocks can occur in any concurrency environment, not just in a database system.

For instance, a multithreading program can deadlock if two or more threads are waiting on locks that were previously acquired so that no thread can make any progress. If this happens in a Java application, the JVM cannot just force a Thread to stop its execution and release its locks.

Even if the `Thread` class exposes a `stop` method, that method has been deprecated since Java 1.1 because it can cause objects to be left in an inconsistent state after a thread is stopped. Instead, Java defines an `interrupt` method, which acts as a hint as a thread that gets interrupted can simply ignore the interruption and continue its execution.

For this reason, a Java application cannot recover from a deadlock situation, and it is the responsibility of the application developer to order the lock acquisition requests in such a way that deadlocks can never occur.

However, a database system cannot enforce a given lock acquisition order since it's impossible to foresee what other locks a certain transaction will want to acquire further. Preserving the lock order becomes the responsibility of the data access layer, and the database can only assist in recovering from a deadlock situation.

The database engine runs a separate process that scans the current conflict graph for lock-wait cycles (which are caused by deadlocks). When a cycle is detected, the database engine picks one transaction and aborts it, causing its locks to be released, so that the other transaction can make progress.

Unlike the JVM, a database transaction is designed as an atomic unit of work. Hence, a rollback leaves the database in a consistent state.

Share Improve this answer Follow

edited Jan 10, 2021 at 10:38

answered Oct 29, 2019 at 10:47

Vlad Mihalcea

153k ● 84 ● 588 ● 974

1

A classic and very simple program for understanding **Deadlock** situation :-

```
public class Lazy {

    private static boolean initialized = false;
```

```
    static {
        Thread t = new Thread(new Runnable() {
            public void run() {
                initialized = true;
            }
        });

        t.start();

        try {
            t.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        System.out.println(initialized);
    }
}
```

When the main thread invokes Lazy.main, it checks whether the class Lazy has been initialized and begins to initialize the class. The main thread now sets initialized to false , creates and starts a background thread whose run method sets initialized to true , and waits for the background thread to complete.

This time, the class is currently being initialized by another thread. Under these circumstances, the current thread, which is the background thread, waits on the Class object until initialization is complete. Unfortunately, the thread that is doing the initialization, the main thread, is waiting for the background thread to complete. Because the two threads are now waiting for each other, the program is **DEADLOCKED.**

answered Aug 22, 2018 at 6:54

Vivek Pratap Singh
9,904 ● 5 ● 22 ● 34

Let's say one thread wants to transfer money from "A Account => B Account" and another thread wants to transfer money from "B Account => A Account", simultaneously. This can cause a `deadlock`.

The first thread will take a lock on "Account A" and then it has to take a lock on "Account B" but it cannot because second thread will already took lock on "Account B". Similarly second thread cannot take a lock on A Account because it is locked by the first thread. so this transaction will remain incomplete and our system will lose 2 threads. To prevent this, we can add a rule that locks the database records in sorted order. So thread should look at accounts name or IDs and decide to lock in a sorted order: A => B. There will be a race condition here and whoever wins, process its code and the second thread will take over. This is the solution for this specific case but deadlocks may occur in many reasons so each case will have a different solution.

Os has a deadlock detection mechanism with a certain interval of time, and when it detects the deadlock, it starts a recovery approach. More on deadlock detection

In the example we lost 2 threads but if we get more deadlocks, those deadlocks can bring down the system.

answered Sep 10, 2022 at 5:50

Yilmaz
48.7k • 18 • 210 • 264

A deadlock is a state of a system in which no single process/thread is capable of executing an action. As mentioned by others, a deadlock is typically the result of a situation where each process/thread wishes to acquire a lock to a resource that is already locked by another (or even the same) process/thread.

There are various methods to find them and avoid them. One is thinking very hard and/or trying lots of things. However, dealing with parallelism is notoriously difficult and most (if not all) people will not be able to completely avoid problems.

Some more formal methods can be useful if you are serious about dealing with these kinds of issues. The most practical method that I'm aware of is to use the process theoretic approach. Here you model your system in some process language (e.g. CCS, CSP, ACP, mCRL2, LOTOS) and use the available tools to (model-)check for deadlocks (and perhaps some other properties as well). Examples of toolset to use are FDR, mCRL2, CADP and Uppaal. Some brave souls might even prove their systems deadlock free by using purely symbolic methods (theorem proving; look for Owicki-Gries).

However, these formal methods typically do require some effort (e.g. learning the basics of process theory). But I

guess that's simply a consequence of the fact that these problems are hard.

Share Improve this answer

Follow

answered Sep 2, 2008 at 20:42

mweerden
14k ● 5 ● 34 ● 32

0

Deadlock is a situation occurs when there is less number of available resources as it is requested by the different process. It means that when the number of available resources become less than it is requested by the user then at that time the process goes in waiting condition .Some times waiting increases more and there is not any chance to check out the problem of lackness of resources then this situation is known as deadlock . Actually, deadlock is a major problem for us and it occurs only in multitasking operating system .deadlock can not occur in single tasking operating system because all the resources are present only for that task which is currently running.....

Share Improve this answer

Follow

answered Apr 19, 2015 at 13:57

puja bharti
1

0

Above some explanations are nice. Hope this may also useful: https://ora-data.blogspot.in/2017/04/deadlock-in-oracle.html

In a database, when a session (e.g. ora) wants a resource held by another session (e.g. data), but that session (data) also wants a resource which is held by the first session (ora). There can be more than 2 sessions involved also but idea will be the same. Actually, Deadlocks prevent some transactions from continuing to work. For example: Suppose, ORA-DATA holds lock A and requests lock B And SKU holds lock B and requests lock A.

Thanks,

Share Improve this answer

Follow

answered Apr 15, 2017 at 13:52

Sapna

71 ● 2

0

Deadlock occurs when a thread is waiting for other thread to finish and vice versa.

How to avoid?

- Avoid Nested Locks
- Avoid Unnecessary Locks
- Use thread join()

How do you detect it?

run this command in cmd:

```
jcmd $PID Thread.print
```

reference : geeksforgeeks

Share Improve this answer

Follow

edited Apr 24, 2018 at 23:20

answered Apr 24, 2018 at 23:11

Arun Raaj

1,800 ● 1 ● 22 ● 20

0

Deadlocks does not just occur with locks, although that's the most frequent cause. In C++, you can create deadlock with two threads and no locks by just having each thread call join() on the std::thread object for the other.

Share Improve this answer

Follow

answered Aug 28, 2019 at 22:44

Raghav Navada

317 ● 2 ● 8

0

The best solution to the deadlock is not to fall into the deadlock

1. system resources or critical space should be used in a balanced way.
2. give priority to key variables.
3. If more than one lock object will be used, the sequence number should be given.
4. Deadlock occurs as a result of incorrect use of synchronization objects.

For Example Deadlock with C#:

```
public class Deadlock{

static object o1 = new Object();
static object o2 = new Object();

private static void y1()
{
    lock (o1)
    {
        Console.WriteLine("1");
        lock (o2)
        {
            Console.WriteLine("2");
        }
    }
}
private static void y2()
{
    lock (o2)
    {
        Console.WriteLine("3");
        lock (o1)
        {
            Console.WriteLine("4");
        }
    }
}
public static void Main(string[] args)
{
    Thread thread1 = new Thread(y1);
    Thread thread2 = new Thread(y2);
    thread1.Start();
    thread2.Start();
}
```

}

Share Improve this answer

Follow

answered Jan 21, 2023 at 9:51

Enes Ceylan

1

-3

Mutex in essence is a lock, providing protected access to shared resources. Under Linux, the thread mutex data type is pthread_mutex_t. Before use, initialize it.

To access to shared resources, you have to lock on the mutex. If the mutex already on the lock, the call will block the thread until the mutex is unlocked. Upon completion of the visit to shared resources, you have to unlock them.

Overall, there are a few unwritten basic principles:

- Obtain the lock before using the shared resources.
- Holding the lock as short time as possible.
- Release the lock if the thread returns an error.

Share Improve this answer

Follow

answered Nov 8, 2013 at 2:54

Marcus Thornton

6,193 ● 8 ● 51 ● 53

3 This describes a lock, not a deadlock. – user207421 Oct 17, 2015 at 9:23