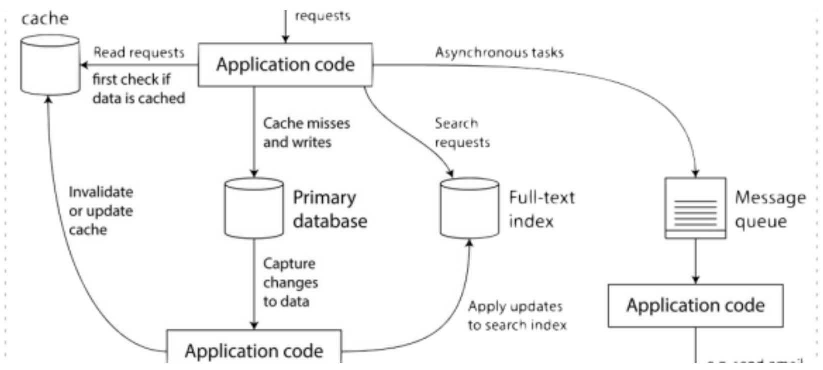
Concurrency - Race Condition (Concurrency Problem)

Table of Contents



1 - About

A Race condition is the only concurrent problem that can happen when two threads manipulate the same state (value) in the same time-lapse, the last thread to write the state will overwrite the state modification of the first thread.

same as Concurrency - Thread Interference (Interleave on shared data) ??

Race conditions have a reputation of being difficult to reproduce and debug, since the end result is non-deterministic and depends on the relative timing between interfering threads.

Problems occurring in production systems can therefore disappear when running in debug mode, when additional logging is added, or when attaching a debugger, often referred to as a W wiki/Heisenbug. It is therefore better to avoid race conditions by careful software design rather than attempting to fix them afterwards.

If two threads run simultaneously without locking or synchronization, the outcome of the operation could be wrong.

1 - Articles Related

1 - Example

- An object with a counter property with the state `1`
- The `thread 1` enters an object and see the state `1`
- The `thread 2` enters also the method and see the state `1`
- The `thread 1` adds 1 to the counter, the state is `2`
- The `thread 2` adds 1 to the counter, the state is `2` (whereas it should be 3)

1 - Resolution

- locking
- or synchronization

1 - Data Problem

Raise conditions lead to data problem called phenomena.

1 - Documentation / Reference

- W wiki/race condition