venkateshcm.com

Reactor Pattern Part 1 - Applications with Blocking I/O

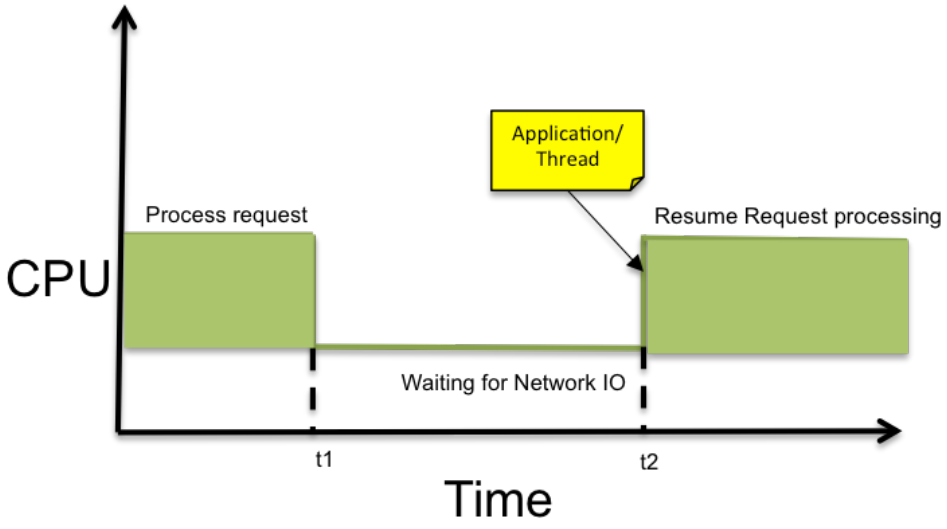
Venkatesh CM

3 minutes

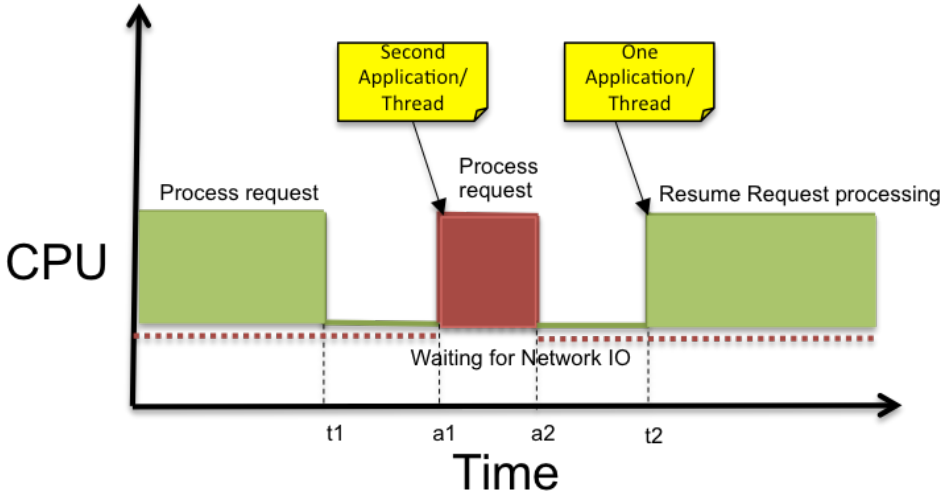
Applications with Blocking I/O

I am assuming a simple scenario of single threaded application like Ruby on Rails Application running on a computer with single CPU. In real world, OS splits CPU time to multiple applications and does a regular context switching.

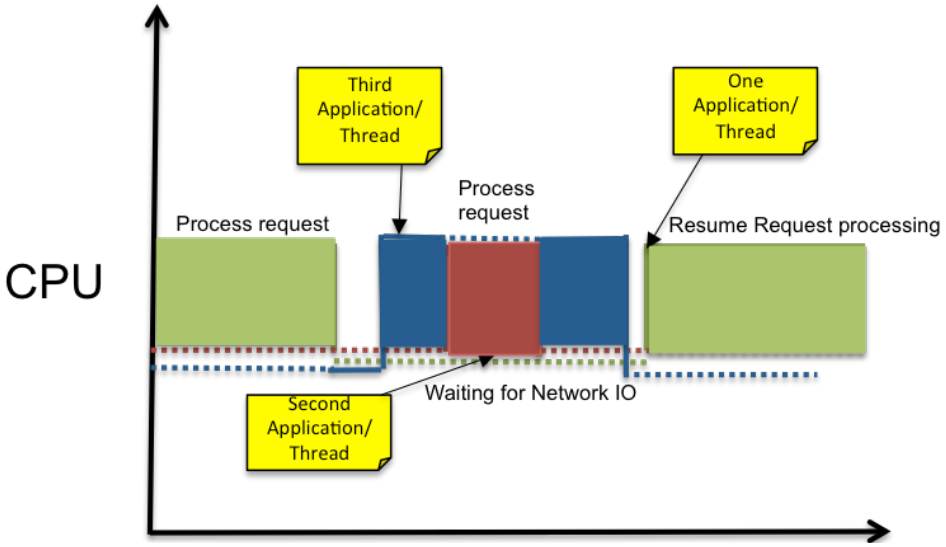
In a single threaded application like Ruby on Rails Applications, requests are processed by a single thread. When the thread makes a I/O bound call like database query or network call, application/thread is blocked even though it could be used to work on other requests.



A solution to get around the above problem is to have multiple applications running on the same box. So when one application's thread is blocked another application's thread can proceed with another request processing. In below diagram there are two applications competing for CPU time and second application consumes CPU between a2 and a1 time. But we still see that CPU is idle between a1 and t1 and between t2 and a2.



To avoid any idle time on CPU we can add more applications, but this will lead to unnecessary context switch which will degrade performance further. For example in the diagram below, third application is switched with second application while third application is still processing a request and needs CPU time.



As shown in the above diagram, we can notice two issues

- OS switches CPU from an application which needs CPU for processing.
- OS switches CPU to an application which is still waiting for IO.

These two issues are due to pre-emptive thread switching. To achieve optimal CPU allocation, application should be able to request for CPU time or give-up CPU time in a cooperative manner instead of pre-emptive switching.

C10K Problem

Early in 2000, a single server could not handle more than 10000 connections at a time. It was a limitation under which applications worked and developers were not able to exceed 10000 connections limit on a single box. The solution that was found is to use nonblocking I/O on each thread i.e. Non-blocking IO stared as a scalability soultion to C10K Problem.

Reactor Pattern provides an work around for the above problem using epoll.

- http://en.wikipedia.org/wiki/Nonpreemptive_multitasking
- http://en.wikipedia.org/wiki/Preemption_(computing)
- http://en.wikipedia.org/wiki/Reactor_pattern

In part 2, we will look at Non-Blocking IO as an alternative solution to maximise CPU usage.