Single core(단일 processor)에서 Spinlock이 의미가 없는 이유

자동차IT융합학과 20183376 박선재

Single core(단일 processor)에서는 Task(process, thread)들이 하나의 공간(single core)에서 각자의 공간을 할당받아 처리되게 된다. 하나의 공간에서 Task들이 실행이 되기 때문에 동시에 단 하나의 Task만 실행이 가능하여 여러 Task들을 동시에 실행을 시키지 못하기 때문에 OS의 scheduling 따로 지원받아 사용자에게 동시에 실행되고 있는 듯한 illusion을 제공해야 한다. 이러한 single core의 특성으로 인해 여러 개의 Task를 context switching을 통해 single core에서 현재 실행되고 있는 Task의 상태를 memory에 save하고 다음 실행할 Task의 정보와 상태값을 memory에서 restore하는 과정이 필수적이다. 여기서 context란 특정 시점에서 CPU(core)의 모든 내부의 상태를 의미한다.

Spinlock은 다른 thread가 critical section에 대해 lock을 소유하고 있다면 sleep을 하지 않고 checking을 loop를 돌며 unlock이 될 때까지 진행하게 된다. lock을 계속 checking해야 하기 때문에 thread를 CPU에 계속 올리고 있어야 하기 때문에 CPU를 낭비하게 되는 단점이 있지만 다른 thread의 lock이 풀린 후 해당 critical section에 자신의 lock을 거는 사이의 time에 delay가 존재하지 않는다. 그러므로 비교적 짧은 critical section에 성능이 좋다.

Spinlock의 특성을 보았을 때 위에서 설명했던 context switching의 비용을 줄일 수 있다. context switching은 실행되고 있는 현재 실행되고 있는 Task를 제외하면 다른 Task들은 memory에 save되어 sleep을 하게 된다. 하지만 spinlock에서는 unlock을 loop를 돌며 계속 확인을 위해 CPU에 thread를 계속 올리고 있으므로 잘 사용한다면 OS의 scheduling을 따로 지원받지 않아 context switching이 일어나지 않게 되어 효율성을 높일 수 있다.

위의 내용들을 정리해보면, single core(단일 processor)에서는 여러 Task를 실행시키기 위해 OS의 scheduling 따로 지원받아 이루어지는 context switching이 필수적이어야 하지만, spinlock은 unlock을 loop를 돌며 계속 확인을 위해 CPU에 thread를 계속 올리고 있으므로 잘 사용한다면 OS의 scheduling을 따로 지원받지 않아 context switching이 일어나지 않는다는 것이 큰 특성이므로 single core(단일 processor)에서는 spinlock이 자신의 특성을 살리지 못해 제대로 된 효율을 발휘하지 못하게 되어 무의미하게 된다.