CS311 Operating System and Labs Homework 2

20200130 Yujun Kim

Thrashing은 degree of multiprogramming이 일정 수준을 넘어갈 때, 각 process당 사용 가능한 메모리가 너무 적어 CPU resource 대부분을 swap-in/out 등의 page allocation에 사용하게 되는 현상을 말한다. 해누리호 A,B가 분담하던 mission을 모두 해누리호 A가 분담하게 되면 해누리호 A는 더욱 많은 process를 만들 것이고, trashing이 일어날 수 있다.

1. Design based on the Working Set Model

- 구현

Working set model은 가장 최근에 reference W(고정된 상수)개의 frame을 track한다. 이 W개의 frame은 서로 중복될 수 있는데 중복을 제외하면 working set size(WSS)를 얻는다. 각 process의 WSS를 더하면 total demand frame number D를 얻는다. D는 시간에 따라 변한다. D가 total number of available frame인 m보다 많아지면 thrashing이 일어나게 된다. 그렇기 때문에 trashing 조건 D>m이 일어날 때 process하나를 suspend하는 방식이 working set model이다. 만약 충분한 extra frame이 남아있다면 새로운 process를 initialize할 수 있다.

해누리호의 경우도 각 process마다 window W를 설정하고 WSS를 추적하여 working set model을 구현할 수 있다. 해누리호의 task는 두개의 priority를 가지고 있기 때문에 suspend할 process를 정할 때 lower priority인 task가 우선적으로 suspend되도록 알고리즘을 구현한다. 추가적인 simulation을 통해 lower priority를 가진 task만 suspend하여도 모든 task가 잘 처리되는 것을 확인 한다면 문제가 없다. 하지만, low priority를 가진 task가 계속 suspend되어 starving하는 현상이 발견된다면 suspend하는 횟수가 일정 수 이상이 되면 priority를 잠깐 늘려주는 알고리즘이 필요하다. 현재 진행중인 process가 모두 같은 priority를 가지고 있는데 한 process를 suspend해야 한다면 가장 오래전에 시작된 process를 suspend한다.

- 장단점(단점 -> Hidden Problem)

Working set model의 장점은 최근에 사용된 page들은 working set이기에 memory에 저장되어 있어 locality 측면에서 이점이 있다는 것이다. Working set은 thrashing이 일어날 조건이 충족될 때 process를 하나 suspend하기 때문에 thrashing으로부터 안전하며, thrashing이 일어나지 않는 조건에서 degree of multiprogramming을 최대화하며 CPU utilization을 늘릴 수 있다. 다만 working set를 정확하게 추적하고, working set size를 계산해야 한다는 것이 단점이 된다. Page reference가 한번 일어날 때 마다 working set이 변하고 이를 매번 계산하는 것은 inefficient하다. 또한 적절한 window size W를 정해주어야 하는 것 또한 단점이다.

이 단점을 보완하기 위해서 working set model의 approximation을 생각할 수 있다. 이는 주기적으로 update되는 reference bit를 추적하여 구현 가능하다. 매 F개의 page reference마다 timer interrupt를 통해 reference bit가 clear된다고 하자. 각 page당 마지막 R개의 reference bit를 저장한다고 하면, 현재 reference bit을 포함해 총 F*(R+1)번의 reference동안 page가 reference되었는지를 확인할 수 있다. 원하는 window의 size W가 있을 때, F*(R+1)를 W와 비슷한 값으로 만들어 W를 사용한 working set model을 근사할 수 있다. F가 커질 때와 작아질 때의 장단점을 비교해보자. F가 커질 때는 더욱 가끔씩 reference bit history를 기록하게 되고, 이를 위해 사용되는 CPU resource가 적어진다. 하지만, F가 커질수록 multiprogramming을 maximize하는 exact working set model과는 점점 달라지게 된다. 반면에 F가 작아질수록 reference bit history를 더 많이 가지고 있어야 하고, history의 update도 더 자주 일어나 이를 위한 CPU resource가 더 많이 사용되어야 하지만 exact working set model에 더욱 근접한 구현이 가능하다.

같은 priority를 가진 process중에 suspend할 process를 정할 때 가장 오래전에 시작된 process를 suspend하는 scheme 또한 장단점이 있다. 단점은 이미 거의 수행이 끝나가는 process가 suspend되어버릴 수 있다는 것이고 장점은 같은 priority의 모든 process가 공평하게 utilize된다는 것이다. 해누리호와 같은 satellite의 OS의 경우 completion time보다 respond time이 작은 것이 중요하다고 생각하여 이와 같은 scheme을 택하였다.

2. Design Based on Page-fault Frequency(PFF) Scheme

어떤 Process의 page fault rate는 그 process에 allocate된 frame의 수와 관련이 있다.

더 많은 frame이 allocate될수록 그 process의 page fault rate는 줄어든다. 덕분에 Page-fault rate에 bound를 두어 allocate할 frame 수를 결정할 수 있으며, 이를 design based on page-fault frequency scheme이라고 한다. 얼마나 자주 이 작업을 실행할지 또한 결정을 해야 한다.

Thrashing이 일어나게 되면 page fault rate가 급격히 늘어나 그 process에 더 많은 frame을 allocate해 주어야 한다. 만약, 모든 process가 더 많은 frame을 allocate하기를 기다리고 있는데 더 이상 allocate해 줄 수 있는 frame이 없다면 process를 골라서 swap out해야 할 것이다. Swap out할 process를 정할 때는 working set model때와 마찬가지로 lower priority의 process를 우선적으로 배제하고, 같은 priority 내에서는 가장 오래전에 시작한 process를 배제하는 방식을 택한다.

- 장단점(단점 -> Hidden Problem)

PPF scheme은 working set model에 비해 관리가 쉽다는 장점이 있지만, working set model에 비해 필요한 memory 수 관리의 optimality가 떨어진다. Working set model은 고정된 window size의 working set 관리의 optimality를 보장한다. 하지만 PPF는 page fault rate를 indicator로 삼아 allocate해 줄 frame수를 정할 뿐, 우리가 정한 page fault rate의 upper/lower bound가 optimal하게 필요한 frame수와 직접적으로 어떤 연관이 있는지는 설명하기 어렵다. 또한 적절한 page fault rate의 upper/lower bound를 직접 설정해주어야 한다는 것 자체도 하나의 단점이다.

두번째 단점(Hidden problem)에 대한 해결책은 다음과 같다. Page fault rate의 upper bound는 클수록 degree of multiprogramming은 늘어나지만 swap-in/out에 사용되는 resource가 많을 것이다. Lower bound는 작을수록 한 process당 더 많은 page가 할당될 수 있고, degree of multiprogramming은 줄지만 swap-in/out은 더 적게 할 수 있다. 적절한 bound는 simulation을 통해 원하는 degree of multiprogramming이 유지될 수 있는 선에서 조정을 해주어야 한다.

3. Discussion between the Two Design

두 design의 비교와 장단점은 section1,2에서 충분히 이루어졌다. Task별로 필요한 computation의 경우, working set model은 window의 크기가 W로 주어졌을 때 그 working set size를 계산하는 작업이 매 page reference마다 O(W)만큼 걸리고, memory

역시 O(W)만큼 필요하다. Working set approximation의 경우, W=(R+1)*F일 때, process 당 O(1/F)만큼의 computation이 필요하고(F번의 page reference마다 O(1)의 작업을 해주기 때문), memory는 (process에게 allocate된 page수)*R만큼이 필요하다.

PPF design의 경우, page fault rate는 가장 마지막 K개의 page reference중 얼마나 많은 page fault가 일어났는지를 기준으로 산정한다고 했을 때, O(K)만큼의 memory resource가 필요하고, page를 주거나 빼앗는 작업을 F번의 page reference마다 한다고 했을 때, O(1/F)만큼의 computation이 필요하게 된다.

4. My Final Design

결론적으로 제시하고 싶은 design은 section1에서 제시한 working set model을 approximate하는 방식 중 reference bit history를 update하는 주기 F가 꽤 큰 design이다. 적절한 window size W는 해누리호 A,B의 task를 모두 A가 수행하도록 한 simulation을 통해 확인 할 수 있을 것이고, (R+1)*F와 W의 값이 비슷해 지도록 page당 refence bit history의 수 R을 결정할 것이다. 또한 section1에서 언급한 이유로, suspend할 process를 결정할 때는 lower priority task를 우선적으로 배제하고, 같은 priority의 process중 배제를 결정할 때는 모두 비슷하게 utilize될 수 있도록 가장 오래전에 실행이 시작된 process를 배제한다.

이렇게 제시된 디자인의 성능을 평가하기 위해서는 결국 얼마나 많은 task들을 처리했는지를 나타내는 CPU utilization을 측정해야 한다. Working set model의 approximation이 working set을 계산하는 과정을 제외할 때 optimal한 exact working set model에 비해 얼마나 많은 CPU utilization을 가져가는지를 측정해야 한다. 또한, simulation을 통해 적절한 window값 W와 reference bit history update주기 F를 얻어야 할 것이다

이 디자인의 한계로는 exact working set model에 비해서 degree of multiprogramming이 떨어진다는 점이 있지만, computational efficiency 측면에서 보았을 때, approximation을 하는 것이 더 합리적이다.