CS330 Homework 2
20180789 이해준

Introduction

해누리 A, B호의 탐사 도중 해누리 B호의 고장으로 해누리 B호의 Mission을 해누리 A호가 수행하게 되었다. 이로 인해 해누리 A호에게 주어진 task가 많아 thrashing 가능성이 매우 높다. 현재 시스템을 개선하여 이 문제를 해결해 보자. 참고로 해누리 A호에는 우주 쓰레기 충돌로 인해 제어 컴퓨터 영역에 구멍이 생겨 태양풍의 영향을 최소화하기 위한 OS가 패치되어 있다.

해당 패치 OS는 매 태양풍의 cycle마다 메모리 영역의 28개 frame을 데이터 이전을 위한 storehouse frame으로 지정하여 다른 프레임들처럼 페이지와 직접 할당되지 않고, 태양풍의 영향을 받는 중이거나 받을 예정일 frame의 데이터를 storehouse frame중 안전 구역으로 이전하는 Remapping 이라는 OS다. 데이터를 새 frame에 이전한 이후 페이지 테이블을 수정하여 frame과 page의 연결관계를 최신화해주고, 새로 연결된 frame을 storehouse frame에서 제외시키고, 기존 프레임을 storehouse frame으로 포함시킨다. 따라서, storehouse frame은 유동적으로 변화하며, 이를 desig_storehouse라는 리스트를 이용해 적한다. 이 디자인에 대한 자세한 내용은 Team44의 Design Project 2 문서에 나타나 있다.

현재 시스템을 개선하기 위해 다음과 같은 두 가지 design을 구상하였다.

- Working set 기반 : Coward working set
- Page-fault frequency 기반 : Conservative frequency feedback

Coward working set

해당 디자인은 Working Set Model에 기반하였다. 프로세스는 일정 시간동안 특정 주소를 반복적으로 참조하는 경향이 있으므로, Thrasing을 최소화하기 위해 자주 참조하는 page들을 메모리에서 swap-out되지 않고 유지하기 위한 working-set으로 지정하여 page fault를 최소화하고 현재 locality 내의 page들을 메모리에 유지시키고자 한다.

이전에 패치 OS에서 해누리A호의 전체 프레임 개수가 2000개이고, 이 중 28개를 storehouse frame으로 지정하였다. 따라서, working set window가 1972보다 너무 커질 경우 working set이 전체 frame 개수보다 커져 page fault의 빈번한 발생으로 해당 디자인이 올바르게 작동하지 않아 해누리호가 폭발할 수 있으므로, working set window는 1972보다 크지 않게 설정한다.

각 page가 working set에 포함되는지 여부를 확인하기 위해 page내에 reference bit와 history bit를 두어 확인하고자 한다. Reference bit를 한 개, hitory bit를 h개로, overhead를 최소화하면서도 working set의 성능을 유지할 수 있도록 bit수를 설정할 것이다. 각 timer interrupt마다 reference bit를 history bit로, hitory bit 내의 bits들은 shift하여 이동시키며, 가장 마지막 history bit는 제거된다. h가 클 경우 accuracy는 증가하지만, 이 경우 메모리 낭비 뿐 아니라 working set을 관리하는 데에 시간이 너무 많이 소요되어 overhead가 증가하게 된다.

하지만 Working set window(Δ) = interrupt interval * number of bits인데, 탐사 작전에서는 해누리호의 안전 문제가 최우선적으로 보장되어야 하므로, overhead보다 최소 대응 시간을 우선적으로 고려해야 한다. 해누리호에서는 malfunction과 관측이나 태양풍으로 인한

영향등을 최소화하기 위해 관측하여 axis의 방향성을 조금씩 수정하는 자세 조정이 최우선적으로 요구된다. 흑점 폭발이나 우주 쓰레기 접근 등 상황이 급격하게 변할 수 있으므로, 해당 task를 처리하는 데 긴 시간이 소요되지 말아야 한다. 따라서, working set을 계산하는 주기가 너무 길게 되면 현재 상황과 동떨어진 메모리 구성으로 대응에 늦을 수 있으므로, h를 일정 크기 이상으로 유지해야 한다. 흑점 폭발로 인한 최소 대응 시간을 고려하여 최소 h를 10으로 설정하였다.

적합한 h와 Δ를 구하기 위해 성능을 평가하여 지속적으로 적합한 h를 반영하였다. 이는 page fault cost와 횟수, 메모리에 적재된 평균 page frame 수와 overhead에 소모되는 시간을 고려한 것이다. 계산식은 total cost = 평균 page fault cost * 평균 page fault횟수 + 평균 page frame 수 * overhead 소모 시간이다.

다만, 해누리호의 경우 frame의 개수가 크지 않다. 최대로 사용할 수 있는 working set의 크기는 1972밖에 되지 않으므로, locality가 이보다 클 가능성이 농후하다. 따라서 working set 모델을 사용하는 경우에도, 여러 개의 프로세스가 동시에 실행되어야 하는 경우에 locality 전체가 메모리에 적재될 수 없어 성능 저하가 발생할 수 있다. 특히 해누리호의 경우 Malfunction을 탐지함과 동시에, 자세 조정과 이를 위한 탐지가 필수적으로 요구된다. 이에 더해 관측을 진행하고 mission을 수행하기 위해 데이터를 처리하고 통신을 유지해야 하므로, working set model을 사용할 시 해누리호가 효과적으로 작동하지 않을 수 있다.

많은 프로세스를 동시에 실행하는 것은 working set의 효과를 감소시키기 때문에 실행 가능한 프로세스의 수를 제한하는 방법을 고려할 수 있다. 이 때, 해누리호의 task priority에 기반하여 프로세스의 실행을 제한하여야 한다. 이를 통해 실행 가능한 프로세스의 개수 내에서 locality의 합이 working set보다 작은 경우에 성능을 보장할 수 있다.

해당 디자인을 Coward working set으로 명명하여 안전을 최우선시하는 working set model임을 강조하고자 했다.

Conservative frequency feedback

해당 디자인은 Page-Fault Frequency Scheme Model에 기반하였다. 이는 page fault rate을 지속적으로 측정하여 적합한 프로세스 수를 산정하는 것이다. 동시에 진행중인 프로세스의 개수가 너무 많을 경우, page fault rate이 증가하여 시스템의 성능이 저하될 수 있다. 반대로, page fault rate이 너무 낮도록 프로세스의 개수를 제한할 경우, 메모리의 사용은 비효율적일 수 있다.

이 디자인에서는 이 page fault rate을 모니터링하여 적절한 프로세스 실행 개수를 설정한다. 또한, 이 디자인에서도 위의 Safe working set과 마찬가지로 28개 storehouse frame에 프로세스의 데이터를 저장하지 않는 것을 보장하고, 해당 유동적 변화 frame들은 태양풍 영향 프레임 대체 용도로만 사용하여 태양풍으로 인한 해누리호의 폭발은 사전에 방지한다.

다만, Page-Fault Frequency Scheme Model에도 위와 마찬가지로 필수 task를 실행하는 데에 소요되는 시간이 일정 수준 이하로 보장되어야 하는 문제가 있다. 그렇지 않으면 page fault가 발생할 때 시간이 너무 많이 소모되어 해누리호의 안전에 문제가 생길 수 있다.

따라서, 위에서 다룬 필수적인 task인 malfunction 탐지와 자세 조정과 이를 위한 관측 프로세스는 일반적으로 실행 가능한 프로세스에 포함시킨다. 이 위에서 통신, mission 수행 위한 관측과 데이터 처리 순의 우선순위를 고려하여 프로세스를 실행시킨다. 이렇게 프로세스를 실행시키면서 page fault rate을 지속적으로 측정하여 적합한 프로세스 실행 개수를 조정한다.

또한, 적합한 page fault rate을 낮게 보수적으로 설정하여, 이보다 page fault rate이 높아졌을 때에도 해누리호의 안전을 보장할 수 있도록 한다.

해당 디자인의 이름을 Conservative frequency feedback으로 명명하여 보수적으로 설정한 fault rate을 지속적으로 조정한다는 의미를 담고자 했다.

Design comparison

Coward working set은 가용 가능한 프레임 개수보다 working set size를 작게 설정하여 page fault의 발생을 원천적으로 차단한다. 이 경우에 page fault가 발생하지 않아, 필수적으로 지속적으로 수행해야 하는 malfunction 탐지와 자세 조정, 안전을 위한 관측의 시행은 보장할 수 있지만, 실행 가능한 프로세스의 수가 상당히 보수적으로 제한적이어서 이보다 priority가 낮은 task들의 실행은 종종 유보될 수 있다.

예를 들어, Malfunction 탐지와 자세 조정을 위한 관측에 소모되는 frame 수는 많지 않을 것이다. 다만, 자세 조정이 발생할 경우에는 사용되는 resource와 computation이 클 것이다. 따라서, 특정 상황에서 자세를 조정하는 경우에는 이보다 우선순위가 낮은 다른 프로세스의 실행이 상당히 지연될 수 있다. 통신의 경우 세분화할 수 있는데, 긴급 통신에 사용되는 resource는 크지 않고 관측 데이터 전송에 사용되는 resource는 크다. 따라서, 긴급 통신의 priority를 더 크게 설정하여 우선적으로 실행될 수 있도록 한다. 이보다 우선순위가 낮은 mission수행을 위한 관측과 관측 데이터 처리에는 상당히 많은 resource와 computation이 소요된다. 이들은 남는 메모리를 사용하여 처리한다. 따라서 이 디자인의 경우 안전은 잘 보장되지만, 관측 성능은 좋지 않을 수 있다.

Conservative frequency feedback의 경우에도 Coward working set과 마찬가지로 우선순위를 고려하여 프로세스를 할당한다. 다만, 이 디자인은 page fault를 원천적으로 차단하지 않고, 낮은 수준을 유지하도록만 하고, 지속적으로 page fault rate을 참고하여 실행 가능한 프로세스의 수를 조정하기 때문에, 평균적인 성능은 Coward working set보다 높을 수 있다.

다만, 우선순위가 낮은 프로세스를 더 많이 메모리에 포함시킴으로써 page fault가 발생했을 때, disk 접근 시간으로 인해 우선순위가 높은 프로세스의 지속적 실행이 보장되지 않을 수 있다. 또한, 어느 시점에서 허용된 프로세스의 수가 큰 편일 때, priority가 높은 자세 조정이 발생하여 평소보다 더욱 많은 resource가 사용되어 page falut rate이 안전 범위를 벗어날 가능성이 있고, 이는 해누리호의 안전을 보장할 수 없는 상황에 이르게 만들 수 있다.

Discussion & Conclusion

국민의 혈세와 순수한 우리 기술로 만든 해누리 A호와 B호는 과학적인 의의 뿐 아니라 상징적인 의미도 있다. 나로호, 누리호, 해누리호에 오기까지 우리는 로켓 발사를 할 때면 온 국민이 한 마음이 되어 뉴스를 지켜보며 응원하고는 했다. 한국 과학의 집약체인 누리호와 해누리호의 발사 성공은 대한민국 과학 진보의 상징이며 세계 정상에 올라섰다는 국민들의 자부심이었다.

하지만, 해누리 B호의 폭발은 이런 자부심과 자긍심에 금이 가게 만들었다. 이는 어쩌면 우리 기술이 아직 선진국의 반열에 오르기에는 역부족인가 하는 의문을 내부에서부터 불러일으켰다. 이렇게 해누리호는 그저 “태양 탐사선”의 의미 외에도 “대한민국의 성공”이라는 더 큰 가치를 지닌다.

따라서, 우리는 “빠른 탐사와 과학적 지식의 습득” 보다도 “느리지만 안전하고 성공적인 탐사”에 조금 더 포커스를 맞춰야 할 것이다. 해누리 B호의 폭발은 어쩌면 우리가 너무 들뜬 나머지 안전 불감증에 젖어 있었음을 이야기하는 것일지도 모른다. 이미 B호가 폭발한 상황에서 A호까지 더 위험에 빠뜨릴 수는 없다. 따라서 컴퓨터 공학자로서 나는 안전을 더욱 중시하는 Coward working set 모델을 사용해야 한다고 주장한다. 이 Coward는 자신에게 닥칠 위험이 무서워서 뒤에 숨기만 하는 겁쟁이가 아니다. 우리 사회가 위험을 피해갈 수 있도록 발 벗고 나서서 오명을 쓰는 대인배다.

분명, 이 디자인에는 성능이라는 한계가 있다. 따라서 다음과 같은 방식으로 이를 개선할 수 있을 것이다.

먼저 task별 priority를 조금 더 세분화한다. 지금 우리에게 주어진 task는 “Malfunction 탐지”, “자세잡기”, “관측”, “데이터 처리”, “통신” 등이다. 이를 “필수” 수준의 우선순위인 “Malfunction 탐지”, “자세 조정을 위한 관측과 데이터 처리”, “자세 조정”, “높음” 수준의 우선순위인 “긴급통신”, “보통” 수준의 “탐사를 위한 관측과 데이터 처리”, “분석 데이터 통신”으로 세분화한다. 그리고, “필수” 수준의 task는 무조건 실행되도록, “높음” 수준의 task는 차우선적으로 실행되도록, “보통” 수준의 task는 이 외의 resource에 맞게 실행되도록 한다.

다음으로, page fault시에 처리 순서를 조금 조정한다. 위의 priorty대로면 “필수” 수준의 우선순위인 task들은 항상 메모리에 적재되어 있다. 따라서 page fault시에는 이보다 낮은 수준의 프로세스를 실행하게 되는데, 이 때 바로 disk에 접근하지 않고, disk 접근 이전에 현재 “필수” 수준의 task가 어떻게 실행되고 있는지를 먼저 확인한다. 이 task들을 시행하는데에 특이사항이 없고, 많은 resource를 사용하는 자세 조정이 시행중이 아닐 때에만 disk 접근을 허용한다. 이 외에는 해당 page fault의 handling을 조금 유보하고, 먼저 우선순위가 더 높은 task를 처리한 후 resource에 여유가 있고 안전이 보장될 때에 다시 진행한다.

마지막으로, working set window의 크기를 확장한다. Coward working set에서는 working set window를 최대한 보수적으로 설정하여 page fault의 발생 자체를 차단했다. 하지만, 프로세스는 빈번하게 같은 페이지를 참조하므로, 일반적으로 최대 프레임 개수보다 working set 사이즈는 매우 작다. 따라서, working set window를 확장하여 page fault를 어느 정도 허용한다면 성능을 매우 크게 향상시킬 수 있을 것이다.

이럼에도, 위에서 priority를 고려하여 page fault의 처리 과정을 조정했기 때문에 안전을 보장할 수 있다. 다만, 태양풍과 관련하여 문제가 발생할 수 있으므로 page-fault가 발생하더라도 태양풍 교란 frame에 접근하여 해누리호가 폭발하지 않도록 기존 OS에서 Cycle이 바뀌기 직전에 수행하였던 데이터 백업 시간을 조금 앞당겨, Disk 접근 시간에는 1tick이 소요되므로, 1tick 이상 앞당긴다면, page fault로 인한 해누리호 폭발 문제도 예방할 수 있다.

또한, Conservative frequency feedback에서 사용했던 것처럼 page fault rate을 지속적으로 측정하여 working set window를 조정한다. 하지만, 너무 크게 조정되는 것을 막기 위해 working set window의 최대 사이즈를 제한한다.

이 개선방안을 적용한 디자인을 “Compass working set”이라고 명명한다. 이 디자인은 해누리호에게 안전한 방향으로 나아갈 수 있도록 지침을 제공하고, B호의 폭발로 길을 잃고 주저앉은 우리 사회가, 과학계가 나아가야 할 방향이 어디인지, 다시 한 번 상기시켜 줄 것이다. 분명 이 디자인은 여전히 성능보다 안전에 포커스를 맞추고 있기에 비판받을 수 있지만, 태양 탐사 특성상 촉박한 시간동안 연구가 진행되지 않고, 긴 시간동안 탐사를 수행하기 때문에 성능은 사실 크게 중요하지 않다. 우리가 우리 손으로 쏘아올린 해누리호는 성공의 이름으로 역사책에 기록될 것이고, 우리의 아이들은 해누리호를 보며 과학자의 꿈을 키울 것이다.