27

메모리 가상화를 정리하는 대화

학생 : *(꿀꺽) 와~, 내용이 엄청나게 많았네요.*

교수 : *그렇구나, 그리고?*

학생 : *음, 이걸 어떻게 다 기억하라는 말이죠? 저, 시험 때를 대비해서 말이죠.*

교수 : *세상에나, 그것 때문에 이 내용들을 기억하려는 것이 아니길 바라네.*

학생 : *그러면 왜 기억해야 하죠?*

교수 : *이런, 그보다 더 잘 알고 있을 텐데. 세상에 나갔을 때 시스템들이 실제로 어떻게 동작하는지 알기 위해서, 여기서 무언가를 배우는 노력을 하는 중이지 않니.*

학생 : *흠... 예를 하나 들어주실 수 있나요?*

교수 : *물론! 대학원에 있을 때 한 번은 내 친구가 메모리 접근이 얼마나 오래걸리는지를 측정하고 있었지. 가끔씩 그 수치가 우리가 예상했던 것보다 너무 높게 나왔었어. 모든 데이터가 레벨 2 하드웨어 캐시에 모두 다 들어간다고 생각을 하고 있었거든. 그래서 접근 속도가 엄청나게 빨랐어야 한다는 거지.*

학생 : *(끄덕임)*

교수 : *무슨 일인지 알지 못했어. 그러한 경우에는 무엇을 해야겠니? 쉬워, 교수님께 여쭤보는거야! 우리 지도 교수님들 중에 한 분께 찾아가서 여쭤봤지. 그 교수님은 우리가 만든 그래프를 보시더니, 간단하게 "TLB"라고 말씀하셨지. 아하! 그렇지, TLB 미스들! 왜 그것을 생각하지 못했을까? 가상 메모리가 어떻게 동작하는지에 대한 모델을 잘 이해하고 있으면 여러 종류의 흥미로운 성능 관련 문제들을 진단하는 데 도움이 된다네.*

학생 : *이해할 수 있을 것 같군요. 이런 것들이 어떻게 동작하는지 개념적 모델을 만들려고 노력하는 중이지요. 현장에 나가 혼자 작업 하고 있을 때, 시스템이 예상한 것과 다르게 동작하더라도 놀라지 않도록 말이죠. 시스템이 어떻게 동작할지를 생각만하는 것으로도 예상을 할 수 있게 되겠네요.*

교수 : *정확하네. 그래서 무엇을 배웠니? 자네의 머릿속에는 가상 메모리가 어떻게 동작하고 있나?*

학생 : *음, 프로세스에 의해서 메모리가 참조되면 어떤 일들이 일어나는지에 대한 개념은 이제 잘 잡은 것 같아요. 여러 번 언급하셨는데요, 프로세스가 각 명령어들을 반입하는 것과 또한 명시적으로 로드와 스토어를 수행할 때 어떻게 동작하는지를 이해했어요.*

교수: *좋아 보이는구나—더 이야기 해줄 수 있을까?*

학생: *음, 이거 하나는 항상 기억할 건데요, 예를 들어 C 언어로 작성된 프로그램에서 볼 수 있는 주소들은 말이죠...*

교수: *다른 언어들은 어떤 것이 있니?*

학생: *(계속하면서) ... 네, 교수님이 C를 좋아하시는 것 알아요. 저도 그렇죠! 어쨌든, 제가 말하려고 했던 것은요, 이제는 프로그램 내에서 볼 수 있는 주소들은 가상 주소들이라는 것을 정확히 알겠어요. 데이터와 코드가 메모리에 있는 것같은 환상을 프로그래머인 저에게 심어 준다는 것도 알겠어요. 예전에는 포인터의 주소를 출력해 볼 수 있다는 것이 멋지다고 생각을 했었죠. 그렇지만, 이제는 불만스러워요—그저 가상 주소일 뿐이잖아요! 데이터가 존재하는 실제 물리 주소를 볼 수가 없단 말이죠.*

교수: *그래, 운영체제는 그러한 정보들을 볼 수 없도록 하였지. 또 무엇이 있지?*

학생: *음, TLB가 가장 핵심인 것 같아요. 주소 변환을 위한 적은 하드웨어 캐시를 갖는 시스템을 지원하는 TLB가 실제로 가장 핵심인 것 같아요. 페이지 테이블은 대체적으로 상당히 크기 때문에 크고 느린 메모리에 존재하죠. 그 TLB가 없다면, 프로그램들은 분명히 아주 심각하게 느리게 동작하게 될 거에요. 가상 메모리를 실제적으로 가능하게 만드는 것이 TLB인 것 같아요. 그것 없이 시스템을 만드는 것은 상상할 수가 없군요! 그리고 TLB가 다룰 수 있는 범위를 넘어서는 작업 세트를 갖는 프로그램을 생각하면 몸서리가 처지는 군요. 그 많은 TLB 미스들을 보기 어려울 것 같아요.*

교수: *그래, 아이들 눈을 가려야할 것 같구나! TLB 외에는 무엇을 배웠니?*

학생: *그리고 페이지 테이블이 꼭 알아야 하는 그런 자료 구조 중에 하나라는 것을 이제는 이해했어요. 그래도 그저 자료 구조일 뿐이죠. 그리고 그 뜻은 어떤 자료 구조도 사용될 수 있다는 것이죠. 먼저 배열과 같은(선형 페이지 테이블이라고도 불림) 간단한 자료 구조로 시작하였어요, 그리고 멀티레벨 테이블(트리처럼 보이는 것)로 발전하였지요. 그리고는 커널의 가상 메모리의 페이지 테이블에 페이지를 사용할 수 있는 좀 더 말도 안 되는 것도 보았어요.*

교수: *그랬었지.*

학생: *그리고, 한 가지 또 중요한 것이 있어요. 주소 변환 자료 구조는 프로그래머가 자신의 주소 공간으로 무엇이든 할 수 있도록 충분히 유연해야 할 필요가 있다는 것이에요. 멀티레벨 테이블과 같은 자료 구조는 그런 면에서 완벽하지요. 사용자가 주소 공간의 일부가 필요할 때만 테이블에 공간을 생성하거든요. 그래서 적은 양만 낭비가 되지요. 간단한 베이스와 바운드 레지스터를 사용하던 초기의 시도들은 충분히 유연하지 못했어요. 자료 구조는 사용자가 예상하는 수준에 부합하여야 하고 가상 메모리 시스템에서 원하는 것을 제공해야 해요.*

교수: *멋진 측면을 보았구나. 디스크로 스왑하는 것에 대하여 배웠던 것들은 어떠니?*

학생: *음, 분명히 공부하기 재미있었고요, 페이지 교체 정책이 어떻게 동작하는지 알게 되어 좋았어요. 기본적인 정책들은 약간은 당연했고요(예를 들어 LRU와 같은). 그렇지만 실제 가상 메모리 시스템을 만드는 것은 좀 더 흥미로운 것 같아요. VMS*

사례 연구에서 보았던 것처럼 말이죠. 하지만 왜 그런지, 기법들이 정책들보다 더 흥미로운 것 같아요.

교수: *오 왜 그랬을까?*

학생: *음, 말씀하셨듯이, 결국에는 정책에 대한 문제에 대한 최선의 해법은 간단했어요. 메모리를 더 많이 사라고 하셨죠. 하지만, 기법은 실제로 어떻게 동작하는지를 이해하고 있어야 한단 말이죠. 말이 나온 김에 ...*

교수: *그래?*

학생: *음, 제 기계가 요즘에 상당히 느려진 것 같아요... 메모리가 그렇게 비싸지도 않으니까...*

교수: *오 좋아, 괜찮아! 여기 얼마 줄게. 가서 DRAM 좀 사거라. 구두쇠 씨.*

학생: *교수님, 감사합니다! 이제 디스크로 절대로 스왑하지 않으리—또는 만약 하더라도, 최소한 어떤 일이 실제로 일어나는지는 알겠습니다!*