2 level page-table

TLB

TLB의 주요한 아이디어는 PTE를 cache하는 것이다

Multilevel Page Table

사용자의 프로그램의 크기에 상관 없이 MPT는 일정하다.
100KB로 만들어지는 프로그램의 PT의 크기는 4MB로 항상 고정된다. → 그래서 비효율적이다.

→ 배보다 배곱이 더 크다

2 level Page Table 구현

- 구현방법 – 예) Two level page-table
  - Outer-page table을하나더두어, 페이지테이블들을가리키도록한다.
  - 예) 앞서의예에서, 20 비트를차지하는페이지번호를다시아래와 같이나눈다.
    - 10 비트 페이지 번호
    - 10 비트 페이지 주소
  - 결국 32 비트 주소의 모양은 아래와 같게 된다.

그런 비효율때문에 생각한 것이 Two level page-table이다.
이것은 page의 level을 두개로 만든 것이다.

virtual memeory는 하나로 되어 있다. → monoithic이라고 말해도 좋다.
L1, L2라고 하기도하는데, 이는 context를 잘 읽어야 한다.

2 level Page Table 도식

L1PT에서는 물리 메모리를 넣고 있는 것이다.

모든 entry가 물리 메모리 주소를 가리키고 있다.

L2PT의 entry는 PTE이다.

PTE로써 물리메모리를 가리키고 있는 것이다.

2 level Page Table의 Table Walk

실제 물리메모리를 찾아가는 과정을 table walk라고 한다.

MMU가 p1, p2를 가지고 PT를 찾는다.

이전에는 CPU가 메모리주소를 가지고 오면 바로 메모리 주소를 찾았었다

그러나 이제는 접근하는 단위가 늘어났다

이전에는 CPU에서 주소를 가져오면 바로 frame을 찾았는데,
이제는 그 단계가 늘어난 것이다.
여기서는 memory access를 3번한다.

(TLB시절에는 memory access를 2번 했다)
따라서 MMU성능이 중요해지는 것이다.

MMU를 하드웨어적으로 잘 만들어서 접근이 빨리 일어나
atomic하게 일어나야한다.
tablework를 할때 interrept가 일어나면 처음부터 시작해야한다.

사실 TLB도 중요

가능하면 table walk를 하지 말아야 한다.

TLB에서 hit이 나면 table walk를 안해도 된다.

이는 근처의 메모리 접근이 일어난다는 의미이다.

지금 접근하고 있는 page와 같은 페이지를 또 접근한다는 의미이다.

이래서 go-to를 쓰지 말라는 것이다.

go-to를 쓰면 locality가 떨어져 성능저하가 일어난다.

학생질문

기왕이면 같은 프로세스의 thread를 연 이어서 스케줄링을 하면 TLB hit이 높아지지 않겠는가?

그걸 고려하면서 스케줄링을 하는가?

→ 현재 스케줄러는 그것을 고려하면서 스케줄링하지 않는다. 그걸 고려하면 너무 복잡하다

프로세스 메니지먼트와 메모리 메니지먼트가 서로 독립적으로 움직여야 함

간단한 프로그램 – 페이지 매핑도식

프로세스의 크기가 12KB이다.

즉, 3 페이지가 있다.

이것들이 어떻게 접근되는지를 보자는 것이다.

→ text 페이지 하나, data 페이지 하나, stack 페이지 하나가 있다.
각각이 충분히 떨어져있다면,
L1PT에서 3개의 entry를 차지한다.

그러면 각각의 L1PT entry는 L2PT의 전체에 해당한다.

L2PT 3개가 있는데, 그중에서도 필요한 PT는

0, 1024, 2048이 3가지로, 회색으로 표시된 것 밖에는 없다.

그러면 이 3가지 PT가 물리메모리를 접근한다.

교수님 질문 :

12KB짜리 프로세스를 2 level Page Table로 맵핑을 시켰을 때

PT가 필요로 하는 메모리리 크기가 얼마인가?

→ PT의 갯수가 몇개인가? 4개이다. (L1PT 하나, L2PT 3개)

→ PT하나의 크기는 얼마인가?

→ PT에 들어가는 entry 숫자는 몇개인가? → 2^{10}개이다.

(페이지 번호를 나타내는 비트의 수가 10개이기 때문이다.)

→ PTE의 크기는 4Bytes(32bits)가 일반적이다.

(20bits는 frame주소가 되고 나머지는 flag가 된다.)

그렇다면 PT의 크기는 얼마인가?

4B * 2^{10}= 4KB이다.

PT의 갯수는 4개이다. 따라서 4KB * 4 = 16KB가 된다.

그리하면 4MB(=2^{20}*4B)를 차지하던 PT의 크기가 16KB로 줄어든 것이다.

16KB/4MB = 1/256 → 무려 256배나 줄어들었다

L1 page table과 L2 page table 모두 page table size는 4KB로 고정되어 있는데, L1 page table은 process 당 반드시 생성되는 반면 L2 page table은 L1 page table에서 populate된 entry에만 생성되는 것인가요?
Yes : populate도 아주 정확한 단어이다.

PT의 값이 들어가면 populate되었다고 표현한다.

Inverted Page Table –필요성

64bit체제에서는 PT가 엄청나게 커진다.

멀티 레벨 PT로도 해결이 안되었다.

Inverted Page Table – think differently

물리주소를 기준으로 그 물리주소를 사용하는 프로세스를 찾자

Inverted page table –구현

- 시스템 전체에 하나의 페이지 테이블만 둔다
  - 페이지 테이블 인덱스 : 프레임의 번호
  - 페이지 테이블 내용 : 프로세스 번호(Process ID)와 페이지번호
  - 참조하고자 하는 메모리주소에 프로세스 번호, 페이지 번호, 페이지 주소를 함께 넣음
  - 예) 12번 프로세스의 23번 페이지, 34번째 데이터
    - 페이지 테이블을 처음부터 검색하여 12, 23의 내용을 가지는 부분의 인덱스를 보고 프레임을 알 수 있음
      - 찾고자 하는 data와 배열의 content가 일치함을 확인했을 때, 그 배열의 index가 검색 결과

        - 해당 프레임의 34번째 데이터에 접근
- 페이지테이블은 보다 적은 용량을 차지하지만, 테이블을 검색하는데 시간이 오래 걸린다 .

  "페이지테이블은 보다 적은 용량을 차지"한다는 것은 프로세스의 갯수가 늘어남에도 불구하고 TP가 고정되어 있다는 뜻이다.

  모든 프로세스의 메모리를 합친 것에 비하여 적은 메모리를 사용한다

  - 해쉬 테이블등을 사용하여 단축 가능

프로세스의 크기나 갯수가 아닌, 물리메모리의 크기에 비례하여 커진다.

메모리크기에 대해서 고정되었기에 모든 프로세스의 메모리를 다 합친 것에 비해서 적은 용량을 차지한다.

64bit에서의 PT의 크기는 얼마가 적절한가?
→ PT는 2의 지수제곱으로 증가해버림
→ 그럼에도 불구하면 4kb를 계속 쓴다.
어떤 경우에는 Huge page(64MB를 쓴다. 그렇지만 4kb를 계속 쓴다.

Inverted Page Table 도식

PT가 있고, virtual 주소가 주어졌다.

virtual 주소를 이전에는 PT의 index로 사용했지만

여기에서는 P를 가지고 entry의 내용(content) 가리키도록 하는 것이다.

그리고 p과 pid가 매칭이 되면, 그때 index i가 물리 frame이다.

Inverted Page Table은 물리메모리가 주어지면

그 물리메모리의 크기에 의해서 Inverted Page Table이 결정된다.

context switching

PT는 context switching을 하면 바꿔줘야 한다.(context가 바뀌니까)

Inverted Page Table은 출발이 어디인가?

각각의 entry가 물리 frame당 하나씩 있는 것이다.

그렇기에 context switching을 할 때 바뀌지 않는다.

pid와 p가 바뀌어야한다.

Inverted Page Table은 프로세스의 숫자와는 관련이 없다.

오직 물리메모리의 크기에 따라서만 결정이 된다.

PT는 최소로 사용되는 것이 2-level을 기준으로 16KB이지만,

프로세스 숫자가 늘어나면 늘어날수록 계속 더해줘야 한다.

Inverted Page Table에서는 프로세스마다 페이지 테이블이 있는게 아니라 글로벌하게 한개가 있는건가요?

→ yes

물리메모리의 크기 * 4B = Inverted Page Table의 크기

inverted page table을 사용하여 물리 frame을 찾을때 시간이 오래 걸리는 이유는 page table의 index가 아니라 content를 찾기 때문에 오래 걸리는 건가요?

→ yes

key를 계속 비교해야한다.

그래서 해쉬같은 것을 사용한다.

교수님 질문
Q. inverted page table에서 TLB가 도움이 될까?
A. TLB비슷한 것을 만들 수 있을 것이다.
→ 그러면 성능에 도움이 된다.
무엇이든지caching이 가능하고, caching은 도움이 된다. Inverted PT은 cashing이 더 도움된다

Q. inverted page table의 TLB는 엔트리 형식이 inverted page table과 달라야 하지 않나요?

A. 잘 응용해 봐라