OS特論 輪講

2025/06/26

鵜川研究室 M2 森本龍

About the Paper

- Authors: Faruk Guvenilir, Yale N. Patt
- Link: https://ieeexplore.ieee.org/document/9138990
- Conference: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)

Virtual Memory System

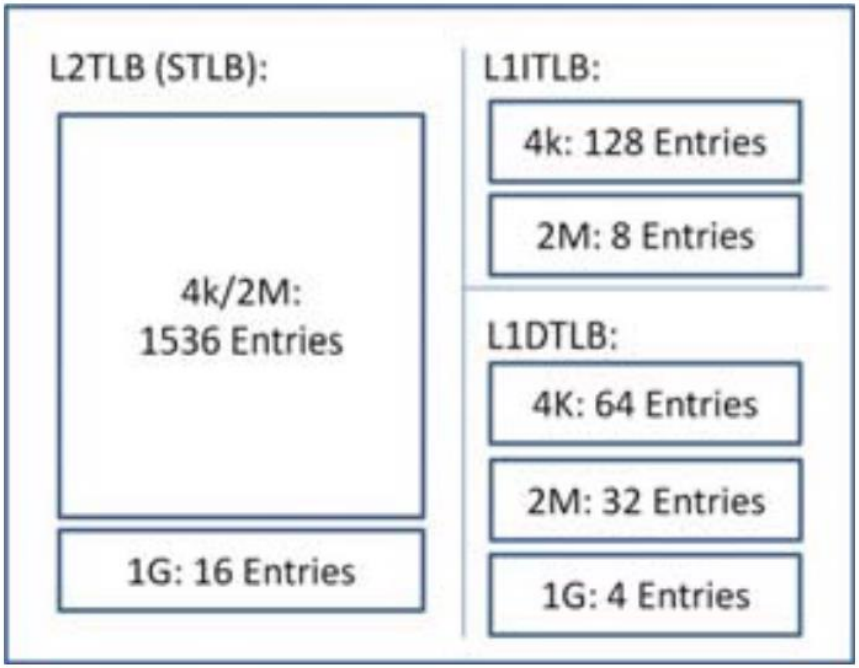
- 現在のcomputerで一般的なメモリマネジメント
- processごとに以下を提供することができる
  - 広大なメモリ量
    - 物理メモリよりも大きな量を提供可能
  - privateなアドレス空間
    - セキュリティの保護
- hardwareが仮想アドレスから物理アドレスに変換する
- OSが仮想アドレスを、仮想ページにわけ、それに物理ページを割り当てる
  - Page sizeごとに仮想アドレスを分ける(一般的には4KBで、2MB/1GBも用意されている)

ハードウェアによるアドレス変換

- Page Table
  - 仮想アドレスで引くと、物理アドレスを得ることができるtable
- TLB: Page Tableのキャッシュ
  - hitすると、ほぼ1cycleでアドレス変換可能
  - Page tableを引くのは遅い(10 ~ 100cycle)

Limitation of Current Virtual Memory Mechanism

- 物理メモリの容量の増大
  - 数10GBの物理メモリ、大きなサーバーだと数TBに及ぶことも
- アドレス変換によるoverhead
  - TLBでカバーできるアドレス空間が小さい
  - memory-intensiveなapplicationでは、TLB missが増大



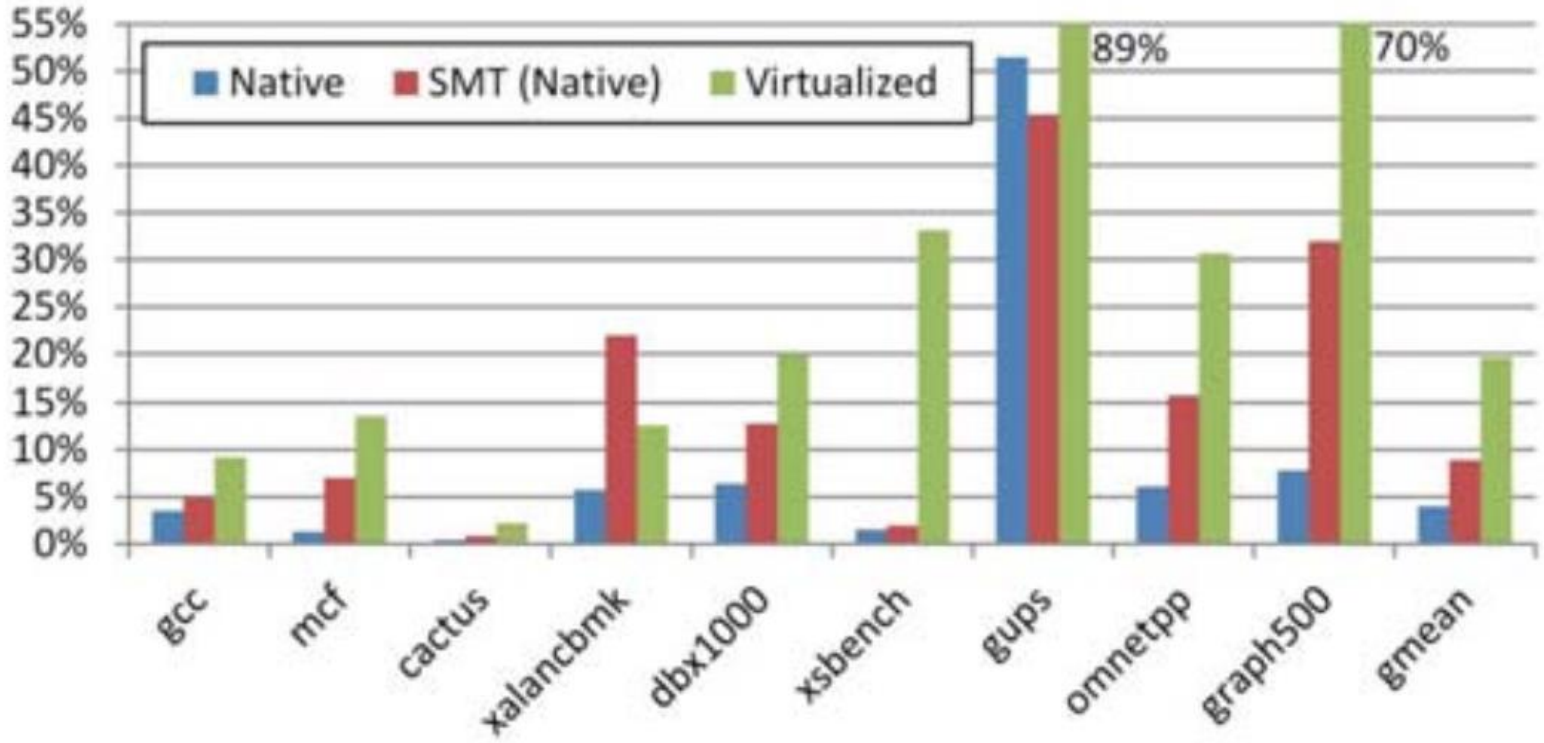
4KB -> 256KB

1GB -> 4GB

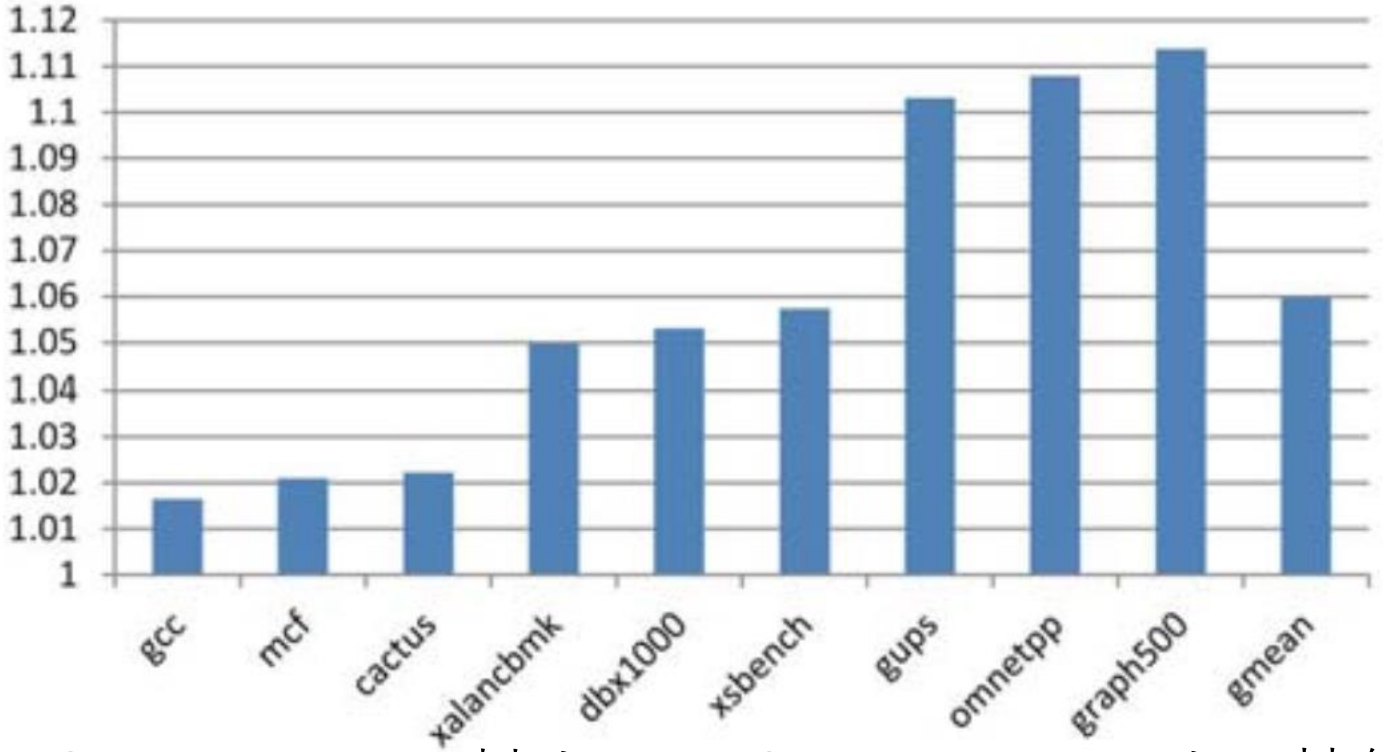
Very small reach

Page Tableへのアクセスによるオーバーヘッド

- application時間の50%をPage table walk(page tableから物理アドレスを得る)に費やしていることもある



TLBミスによるオーバーヘッド



perfect L2 TLBに対するperfect L1 TLBによる性能向上比

- out-of-order実行によりL1ミスのlatencyが隠されることがある
- 依然として、メモリアクセスがcritical passであればoverheadは避けられない

Contribution

- Tailored Page Sizes(TPS)を提案
  - 任意の2^n(>= 4KB)のpage sizeを扱える
  - simulationにより性能を検証
    - 多くのベンチマークにおいて、L1 TLB hit率を99%以上向上

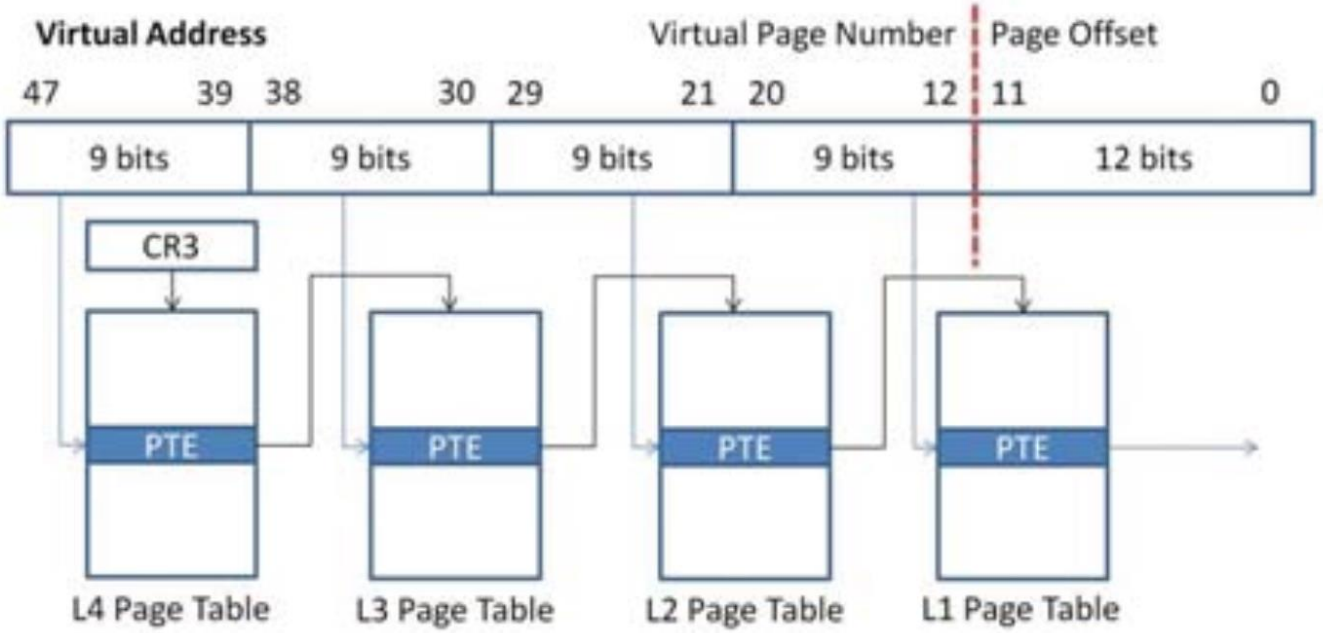
TPSの利点

- 従来手法では256MBの領域を確保する時に
  - 2MBのページを128 frame -> TLBのentryに収まりきらない(TLB miss)
  - 1GBのページを1 frame -> 768MBが無駄に
- 大きいpage sizeを用いると、page table walkでのメモリアクセスが減る

  -> 4KB/2MB/1GBという粗いpage sizeしかないのが原因

TPSのHardware的なChallenge

- x86におけるpage table walkは4段階(4KBのpage size)
  - 9bit * 4でtableを引き、残りの12bit(2^12=4KB)がoffset
  - 2MBの時には、L2から物理アドレス番号を得て、下位21bit(2^21=2MB)がoffset(**3回のアクセス**
  - 1GBの時には、L2から物理アドレス番号を得て、下位30bit(2^30=1GB)がoffset(**2回のアクセス**

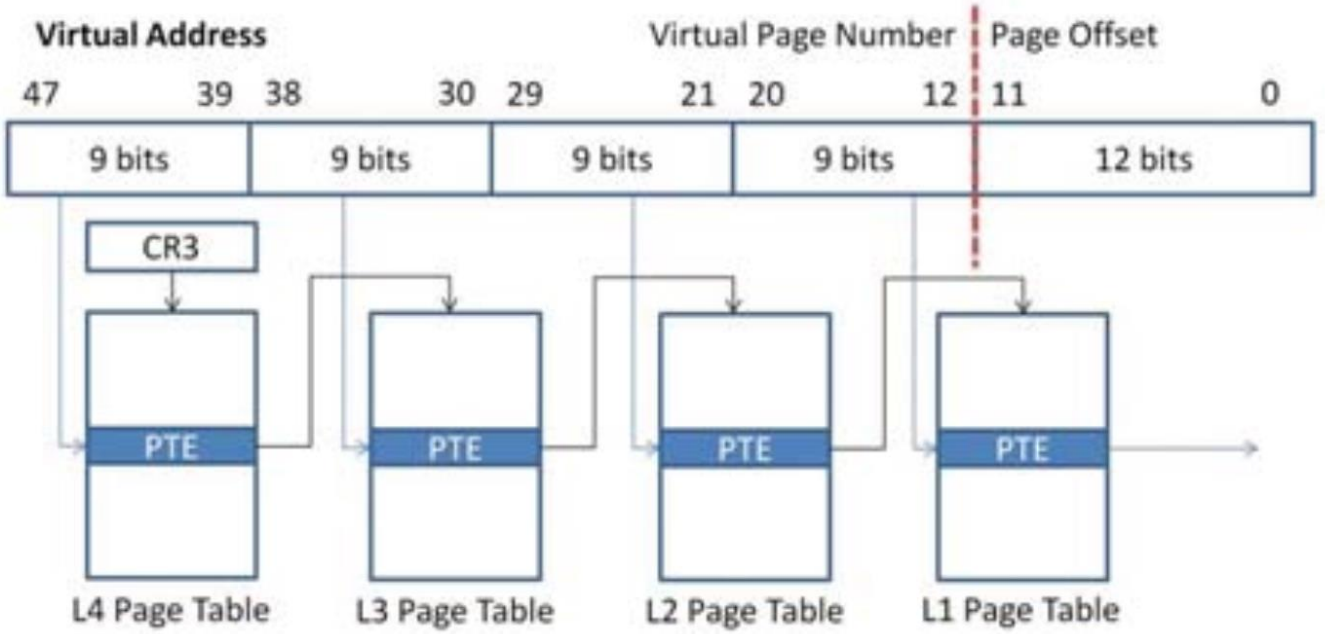


TPSのHardware的なChallenge

- 大きいpage sizeのPTEは、flagが立っている
  - page table walkの時に判断可能

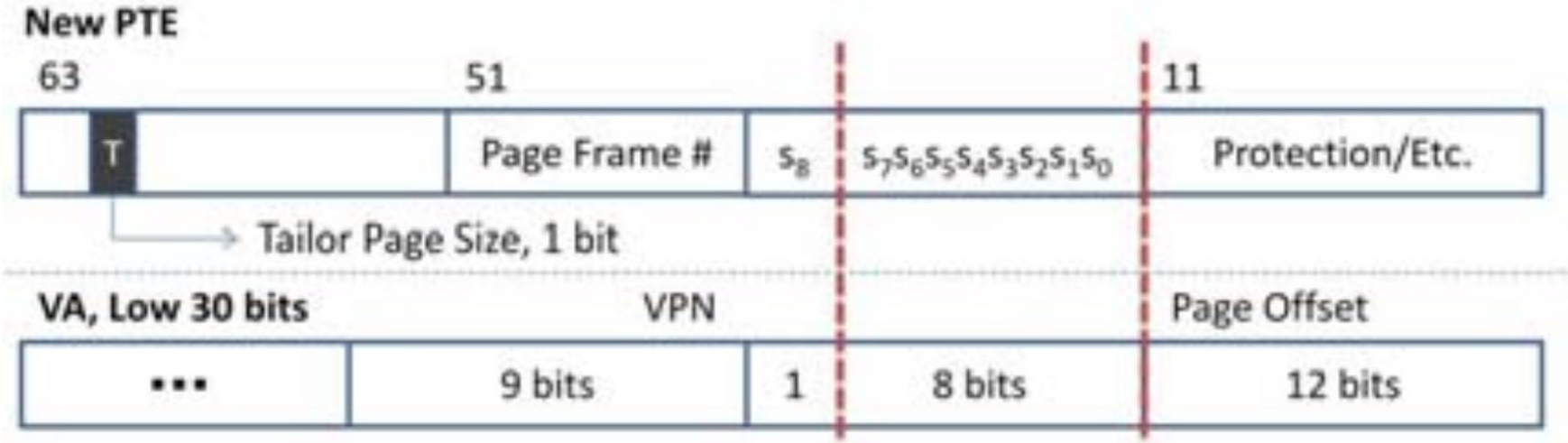
Challenge:

任意のpage sizeだと、下位何bitがoffsetかわからない



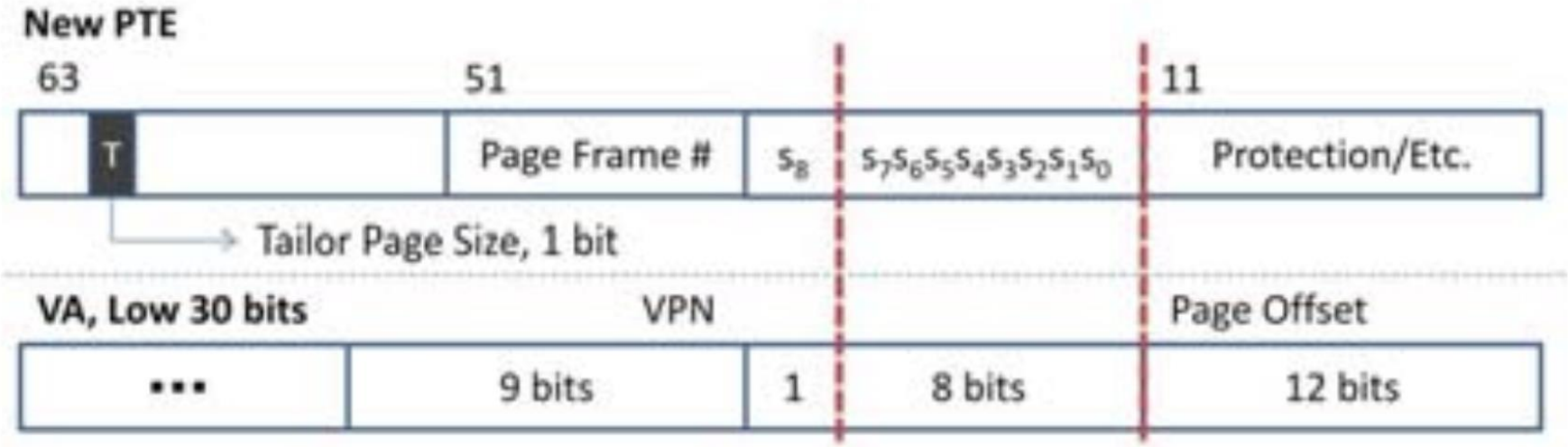
提案手法

- 追加で1bitのflagをPTEに追加する
  - pageがtailored(!=4KB)かどうかを示す
- 物理ページ番号(図中のPage Frame #)の使用されないbitを使用する
  - tailoredだと下位bitが使用されない



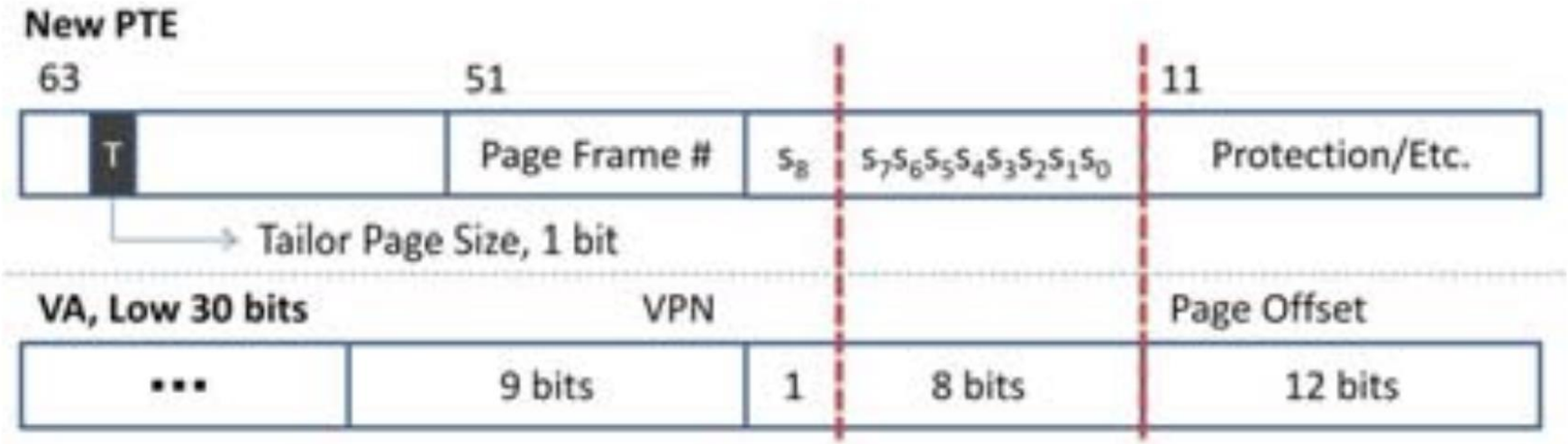
提案手法

- (Page Frame #) * (4KB)が物理ページアドレスとなる
- 8KB(tailored)だと、必ず(Page Frame #)は2の倍数となる(下位1bitは必ず0)
- 16KB(tailored)だと、必ず(Page Frame #)は4の倍数となる(下位2bitは必ず0)



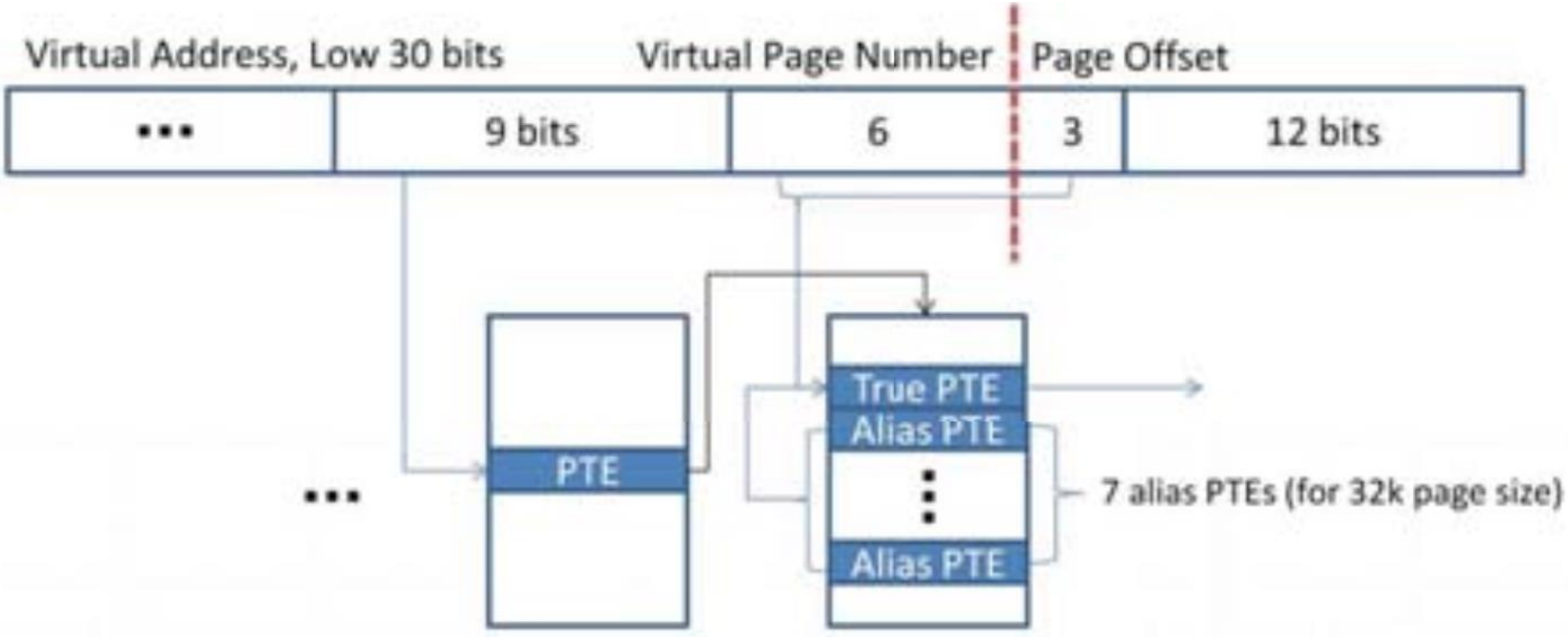
提案手法

- T=1, s0=1 -> 8KBのtailored
- T=1, (s1 s0)=(1, 0) -> 16KBのtailored



Duplicated PTE

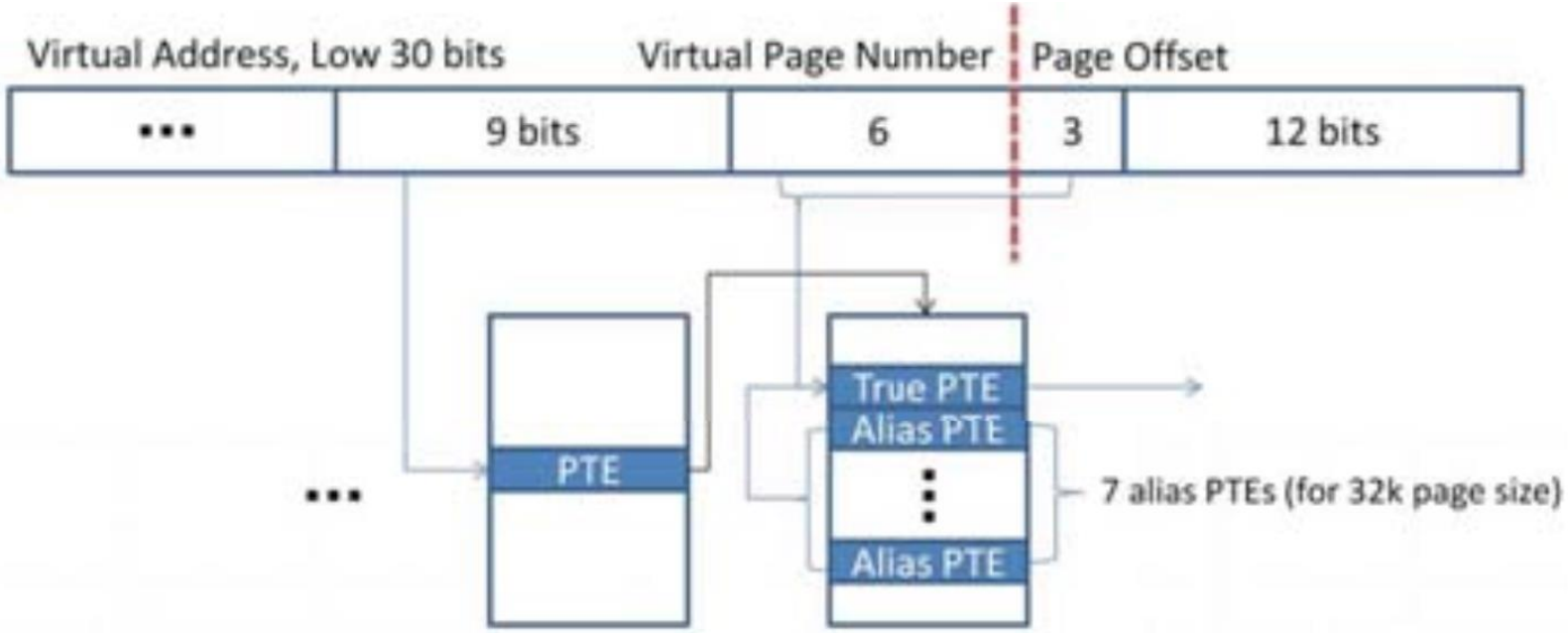
- indexにoffset bitの一部も含まれるので、対応するPTEが複数にまたがる



32KB(tailored)のpage table walk

- 3bitが混入するので、8 entryにまたがる

Duplicated PTE



- page確保時にOSがまとめてentryを確保する
- 一つだけ本物のentryとし、他を引くと本物を引き直す
  - 混入したoffset=0に対応するentryが本物
  - alias PTE(偽物)を引いても、offset数は分かる
  - 余分な1アクセスが発生してしまう

Hardware以外での提案手法

- 論文では、TPSを行うにあたってのOSの変更点
  - Tailoredで確保したページでも、段階的に物理ページを割り当てる
    - メモリ使用量
    - Lazy allocation
- Fragmentationを軽減する方法

Evaluation

- 2段階評価を使用
  - 本来ならmemory-intensiveなworkloadを最初から最後まで動かすのが理想だが、cycle base simulatorでは時間がかかりすぎる
- まずTLB hit率を評価
- Cycle数を評価
  - TLB hit率が寄与しているか

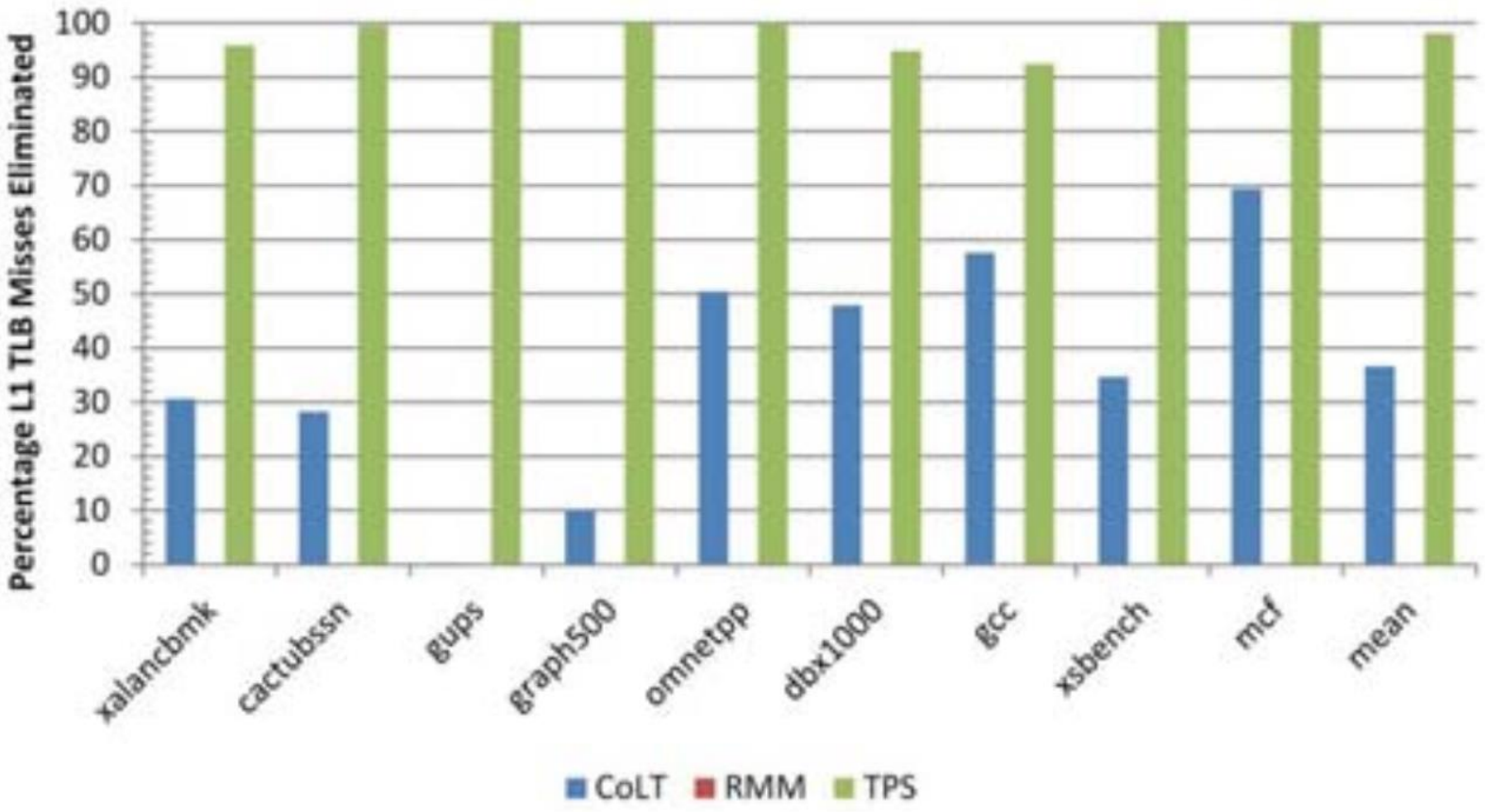
Benchmark

- SPEC17 suite
  - TLB missesのMPKI(1000命令あたりのmiss数)が5を超えるものを使用
- Graph500
- GUPS
- XSBench
- XSBench

評価手法

- TLB miss率を見るために
  - PIN-based OS-allocator
  - virtual memory simulator
- Cycle数を見るために
  - ZSim, a cycle-based simulator of an x86 superscalar out-of-order

TLS Miss Elimination



- TLB missにおいて、98%の向上

Cycle数の高速化

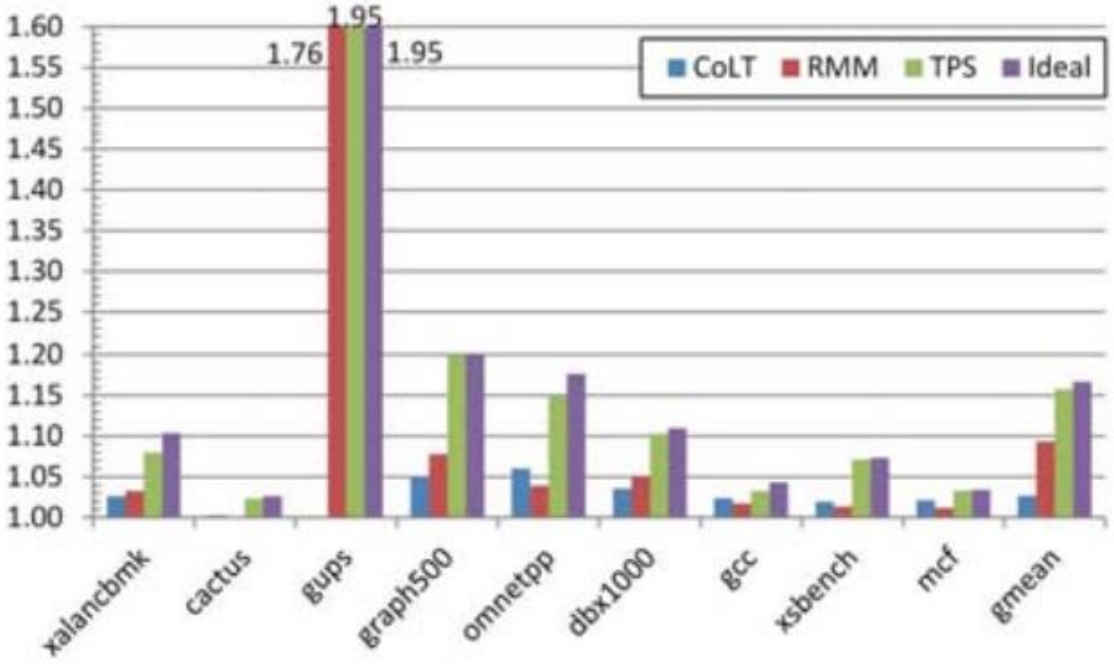
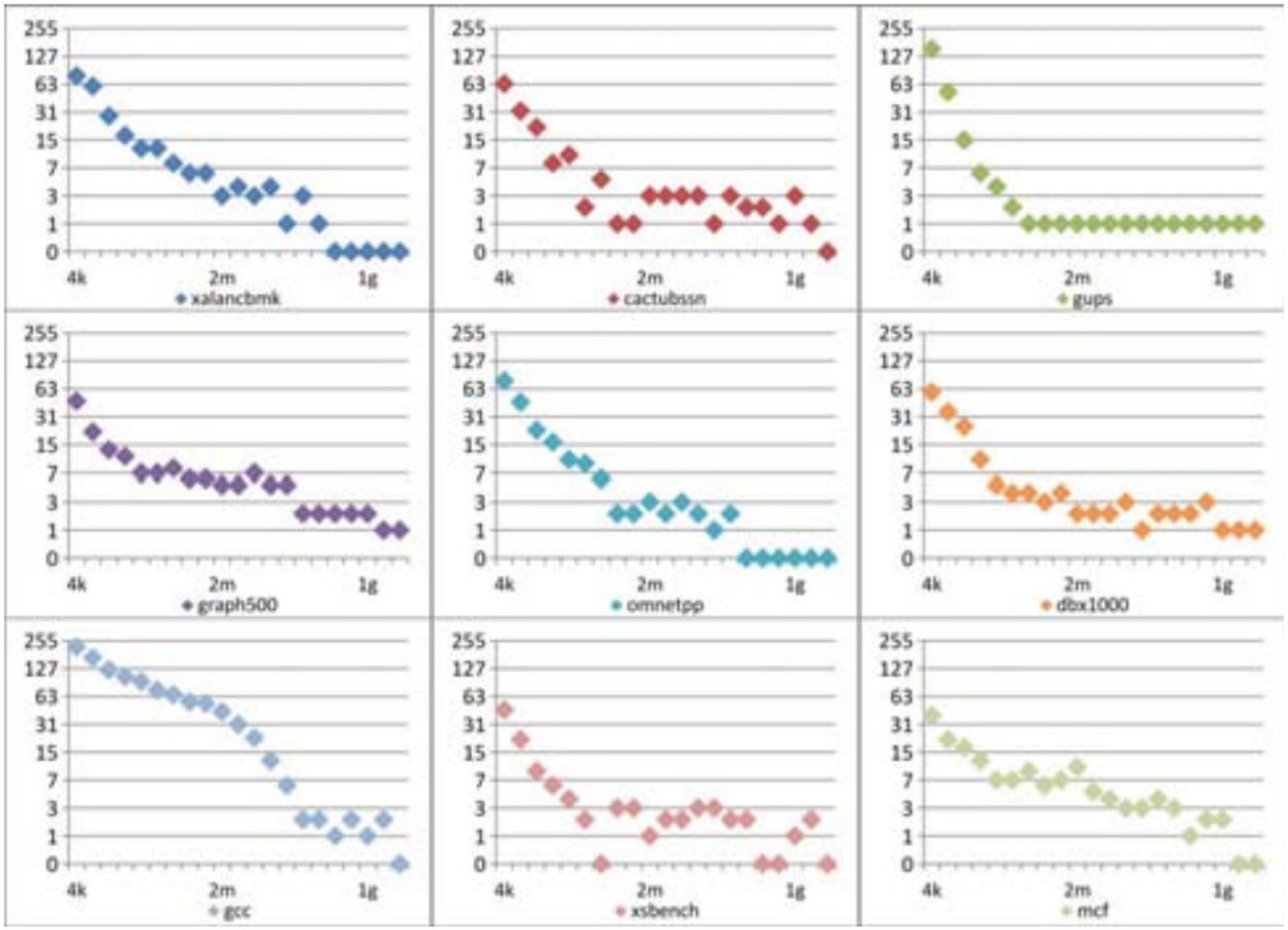


Fig. 13. Speedup - Native (no SMT)

- IdealはTLB missやpage table walkがない時
- TPSで平均15.7%の向上
- Idealの99.2%に相当

各Page sizeの使用数



- 比較的小さな(8K~2M)が多く使用されている
- これらにより性能向上