1. Hugepage簡介

Page Table的限制

OS透過Virtual Address的技術，使得Logic Memory可以比Physical Memory大很多。每個Proccess擁有從0開始的Virtual Address，不僅可以讓program脫離實際記憶體限制的考量，也可以保護不同process間的資料。而Virtual Address與Physical Address間的轉換則透過存放在記憶體中的page table完成，透過查表找出對應的Physical Address，然而，每次存取記憶體都要透過page table查表會產生過多的overhead，因此系統透過Translation Look-aside Buffers(TLBs)，將常用的page mapping儲存在CPU內，透過hardware support提高效率。問題是，TLBs大小有限，是相當珍貴的資源，不可能將整個page table放在裏面，因此若要更加提升效率必須有其他機制。

Regular page

標準的page size為4K，假如現有的TLB Cache只能存放64bit的mapping，那麼TLB只能涵蓋4K*64=256K的hot address mapping。

Huge page

為了增加hot address mapping所能涵蓋的address範圍，有兩個方向進行突破：一是擴大TLB的大小，二是增加page size。前者由於hardware工藝技術以及設計成本，較為困難。而後者就是本文所介紹的Huge page。

Linux Kernel 2.6之後，Huge page便成為Linux系統的一部份。在使用huge page的系統中，huge page size有不同大小（2MB到256MB），page table增加了”Hugepage”屬性，因此page可以被紀錄為regular page或是huge page。以下羅列huge page的幾個主要特色。

a. 使用Huge page減少了page的總數量，使得page table的查找更有效率。
b. 使用Huge page可以讓TLB覆蓋日漸龐大的Physical Address，讓更多address可以被hot mapping。
c. Huge page會在系統啟動時，直接分配並保留相應大小的記憶體，若沒有系統管理員介入，系統不會釋放或改變huge page。
d. Huge page不會被swap，也就是不會page in/out，會一直被pin在memory中，不會被page或swap到secondary storage。

Transparent Huge Page(THP)

THP是RHEL 6後引入的功能，標準的Huge Page是預先分配的，開機後便不再更動，而THP是動態分配。THP和傳統Huge Page若同時使用會造成性能問題和系統重啟，因此在部份版本後刪除THP（Oracle Linux6.5）。傳統Huge Page很難手動管理，於是Red Hat在Linux 6後加入THP，（但是THP不建議在資料庫系統中使用），THP是抽象層（Abstraction Layer）可以自動產生、管理。

Application

Huge page可以改善TLBs Cache Miss所造成的Page table lookup，也就是hot mapping分散，以及process超過TLBs size的系統。例如在資料庫系統中，經常會使用huge page技術加速連續資料的存取。在Main Memory為數十甚至數百G的大型server或cluster中，也常會用Huge Page來增加TLBs的覆蓋範圍。

Some Bad Effect

在使用NUMA的系統，若是寫入操作密集的程式運作，那會使得Cache寫入衝突的機率大幅增加，比喻來說，就是原本用來保護10行數據的鎖現在用來保護1000行數據，導致鎖在各個process間的搶奪機率就大幅增加。此外，也會導致某些連續數據原本應該是透過hot mapping存取，結果因為數據被迫分佈在兩個page間，並且因為CPU親和力權重被迫分配在兩個不同CPU，導致CPU不得不通過CPU inter-connect去remote CPU存取數據。因此在NUMA的系統Huge page若使用不當可能帶來負面影響。

2. 參考資料

a. Huge Page 是否是拯救性能的萬能良藥？
   http://cenalulu.github.io/linux/huge-page-on-numa/
b. Linux傳統Huge Pages與Transparent Huge Pages再次學習總結
   http://www.zendei.com/article/37419.html