Linux メモリ管理の課題と進化
100 万倍のスケーラビリティーを求めて

小崎資広

概要

現在 Linux コミュニティにおいて VM の改善の議論が活発である。本チュートリアルでは、なぜ長期にわたって良好に動いてきた VM が時代に合わなくなってきたのか。また現在コミュニティ上でどのような解決方法が提案されているかを説明する。

はじめに

近年 CPU クロックの成長は鈍化を示し、技術トレンドはマルチコアへと移りつつある。一方 DRAM 容量はいまだムーアの法則が破れておらず倍々の伸びが続いている。

また Linux の適用分野は過去 10 年において大きく拡大し、現在ではローエンドはメモリ 16M 程の組み込み機器からハイエンドは 16TB 程のスーパーコンピュータまで広く使われている。これは Linux のメモリ管理は、100 万倍のメモリ容量差に対してスケールする事が求められている、と言い換えることが出来る。

ページ回収処理は（誤解を恐れず端的に言えば）「全メモリページの中からもっとも不要なページを選択する操作」であるため、Linux のメモリ管理の中でもメモリ量の拡大にダイレクトに影響を受ける部分である。

本チュートリアルでは現在の Linux 実装が直面している課題とそれに対してコミュニティ上で提案されている改善案を紹介する。

Linux のページ回収概要

よく知られているように Linux のページ回収論理は Mach の FIFO with second chance に若干の修正を加えられたものが採用されている。

これは Active, Inactive の 2 つの FIFO リストにページを LRU 順に連結していき、もっとも参照されなかったページを回収するという論理である。

メモリが不足し、ページ回収処理が始まるとリストの先頭から回収可能かをチェックしていき Referenced ビットが落ちていれば最近アクセスされていないとい

う印であるので回収し、逆に Referenced ビットが立っていたら最近アクセスしたという印なので Referenced ビットを落とした後リスト最後尾につなぎなおす。というセカンドチャンス論理が実装されている。

問題点

「ページをチェックして Referenced ビットの落ちているページを見つける」と言葉で書くと一瞬で終わるイメージがあるが、ページのメタデータのトラバースはそれほど軽い処理ではない。たとえば x86_64 においては、ページサイズは 4K であり、１ページあたり 64byte の page 構造体（ページのメタデータを保存する管理領域）を必要とするが、これはシステムが 1TB のメモリをもつ場合 16GB の領域を使用するという事を意味する。これは運が悪いとメモリアクセス時間だけで数十秒の時間を要するメモリ量である。別の言い方をすると、最初にセカンドチャンスを与えたページをもう一度チェックするまでに最短でも数十秒あるということである。

これは活発なプロセスがもう一度メモリにアクセスして Referenced ビットを立てるに十分な時間である為、２回目のチェックもやはり回収できないということが容易に起こり得る。当該プロセスが同じように活発に動作しつづける限り３回目、４回目も同様に回収に失敗する。

つまりページ回収はシステムのメモリ量に比例して処理時間が延びるという特性と、処理時間がある閾値を越えるとページがまったく回収できない状況が発生するようになるという特性を持つ。

なお悪い事に近年 CPU の性能向上はコア数の増加に向かっている。上述の Referenced ビットのレースが 8Core のシステムで発生した場合、8Core 中ビットを立てる処理(プロセスのメモリアクセス)をする Core が 7 つ、ビットを落とす処理(ページ回収)をする Core が 1 つのような極端にページ回収に不利なレースになる事が起こりうる。

このため上記のような終わらないページ回収問題が発生しないよう、ページ速度の高速化することは近年ホットな話題の一つとなっている。

提案されている改善内容

１．Split LRU

現在の実装は同じ LRU 上に回収不可能なページ（e.g. mlocked page)、通常ファイルのキャッシュ、スワップを使用するページ（e.g. 匿名ページ、tmpfs、共有メモリ）など様々なページが混在しており、現在の VM の「メモリプレッシャーが軽度の場合は通常ページのキャッシュのみを回収する」というポリシーを実装するのに不必要に多くのページチェック（とそれに伴うリスト・トラバーサル）

が必要である。データ構造を分けることにより、このネックを根本的に改善する事が出来る。

２．vcompound ページ

そもそもページ回収が重い処理となっているのは 4k というページサイズが今となっては小さすぎ、ページ構造体の数が増えすぎている為である。これは全てのページのページサイズ 2M とする事で激減させる事ができるが、無駄になってしまうメモリが大幅に増加し現実的ではない。

そこで連続したメモリを使用するときのみ自動的に 2M ページに格上げされるパッチが提案されている。

これによりチェックすべき page 構造体の数が最良で 1/512(= 4k/2M)となるためページ回収処理を大幅に高速化できる。

なお当日は、実装についてより詳細に解説を加える予定である。

参考文献

[1] Linux Kernel ソースコード
http://www.kernel.org
[2] LKML アーカイブ
http://marc.info/?l=linux-kernel
[3] [PATCH 00/24] VM pageout scalability improvements (V12)
http://marc.info/?l=linux-kernel&m=121321037203588&w=2
[4] linux-mm　Problem Workloads
http://linux-mm.org/ProblemWorkloads
[5] linux-mm　Page Replacement Design
http://linux-mm.org/PageReplacementDesign
[5] Cristoph Lameter “Bazillions of Pages” In Proceedings of the Linux Symposium 2008
http://www.linuxsymposium.org/2008/ols-2008-Proceedings-V2.pdf