

# メモリ管理の動向

富士通 小崎資広

<kosaki.motohiro@jp.fujitsu.com>

# Who am I?

- 2008年にLKMLデビュー
- 2008年以降について集計するとvmscan.cを一番いじった人
- Linuxのメモリ管理界隈でSplit-LRU VMの共同開発者として有名(ようするに無名)
- @ITでKernel watchというLinuxの解説記事を連載している。LKMLのメールを全部読んでいる数少ない日本人の中の一人
- 2008年のLKMLでのコミットレート
  - Signed-off-by数がTOP10%にぎりぎり入るぐらい
  - mm/ ディレクトリ以下に限定すると10位
  - Reviewed-by 数はLKML全体で7位

# 言葉の壁？

... once again demonstrating that Aussies  
a sense of humor beyond the grasp of  
the rest of us mere **mortals**.

-- Paul Jackson

# 2008年のメモリ管理トピックス

- -mm tree gone
- Split-LRU VM
- Lockless pagecache
- MMU notifier
- Mem-cgroup
- Reclaim bail out
- SLQB

# -mm tree gone

|  |                                |                      |  |
|--|--------------------------------|----------------------|--|
| The latest stable version of the Linux kernel is:                        | <a href="#">2.6.28.5</a>       | 2008-02-12 17:54 UTC | <a href="#">F</a> <a href="#">V</a> <a href="#">VI</a> <a href="#">C</a> <a href="#">Changelog</a> |
| The latest <a href="#">prepatch</a> for the stable Linux kernel tree is: | <a href="#">2.6.29-rc5</a>     | 2008-02-13 22:26 UTC | <a href="#">B</a> <a href="#">V</a> <a href="#">VI</a> <a href="#">C</a> <a href="#">Changelog</a> |
| The latest 2.4 version of the Linux kernel is:                           | <a href="#">2.4.37</a>         | 2008-12-02 08:13 UTC | <a href="#">F</a> <a href="#">V</a> <a href="#">VI</a> <a href="#">C</a> <a href="#">Changelog</a> |
| The latest 2.2 version of the Linux kernel is:                           | <a href="#">2.2.26</a>         | 2004-02-25 00:28 UTC | <a href="#">F</a> <a href="#">V</a> <a href="#">Changelog</a>                                      |
| The latest <a href="#">prepatch</a> for the 2.2 Linux kernel tree is:    | <a href="#">2.2.27-rc2</a>     | 2005-01-12 23:55 UTC | <a href="#">B</a> <a href="#">V</a> <a href="#">VI</a> <a href="#">Changelog</a>                   |
| The latest <a href="#">-mm patch</a> to the stable Linux kernels is:     | <a href="#">2.6.28-rc2-mm1</a> | 2008-10-29 06:29 UTC | <a href="#">V</a>  |

**F** = full source, **B** = patch baseline, **V** = view patch, **VI** = view incremental, **C** = current [changesets](#)  
Changelogs are provided by the kernel authors directly. Please don't write the webmaster about them.  
[Customize the patch viewer](#)

- -mm は従来α版カーネルの役割を果たしてきた
- 最終更新が4ヶ月前
- mmotm (mm on the moment) に移行
- 大きな混乱は起こっていない

# きっかけ

- ここ数年はメモリ管理の変更は少なかった
- メインライン向けのパッチが-mmにも適用できる
- Mmotm自体は前からあったが誰も使わず
- 去年、6月末にlockless pagecacheとSplitLRU VMがほぼ同時にマージ
- パッチコンフリクトで-mm treeが壊れる。panic頻発
- 次々作られるパッチ。-mmとmmotmの差が広がる
- メモリ管理開発者がmmotmベースのパッチを投稿するようになる
- ますます-mmツリーのリリース頻度が落ちていく

# Split LRU VM

- 従来2本だったページ管理用のLRUリストを5本に分けるパッチ(ゆえにSplit LRU)
- ページ回収処理のスケールラビリティを改善

# 多すぎるページ数問題

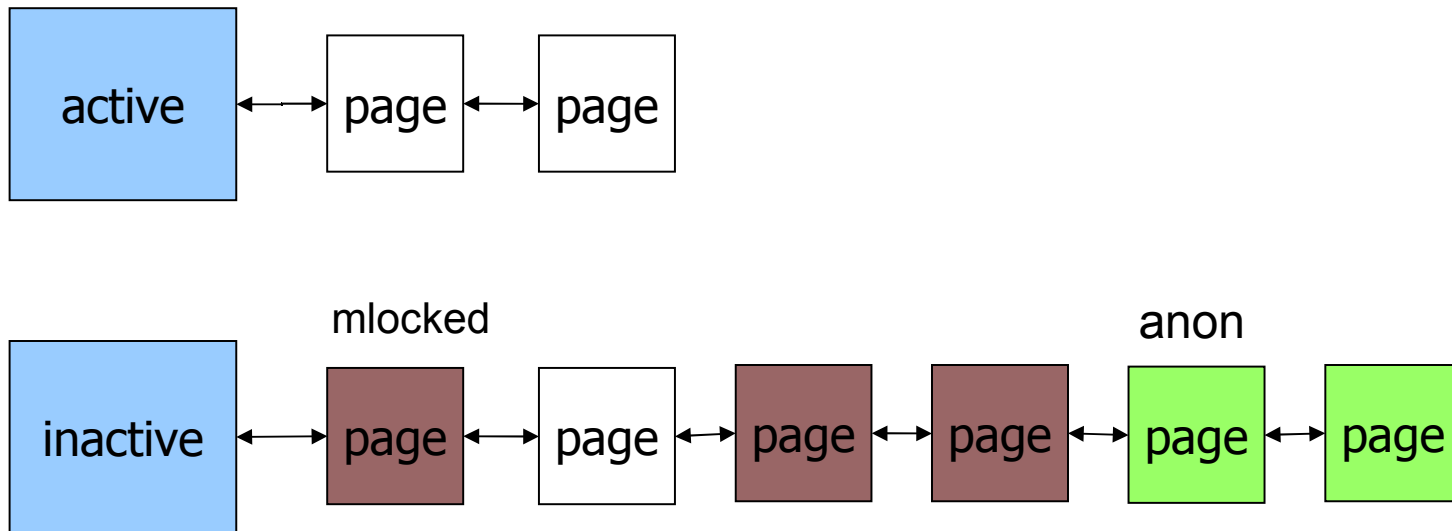
- システムのメモリ量はどんどん増える
- x86\_64は4kページあたり64byteのstruct page。つまり1GBあたり16MBのstruct page。1TBなら、なんと16GB。
- この巨大なLRUリストで捜査していかないとページ回収できない
- リスト走査はキャッシュヒット率低い
- 回収速度はどんどん悪化

# 回収できないページが多すぎる

- DBによっては、大量の共有メモリを使う。  
搭載メモリの8割が共有メモリに使用されたりする
- Linuxはメモリプレッシャーが低いうちは要Swap I/Oページは回収対象にしない
- キャッシュミスしながらページをチェックすると、8割の確率で捨てられないページ。それを何百万回もループ回す。速度が出るわけがない
- MlockされたページをつかうオンメモリDBも流行ってるけど、結果は同じ

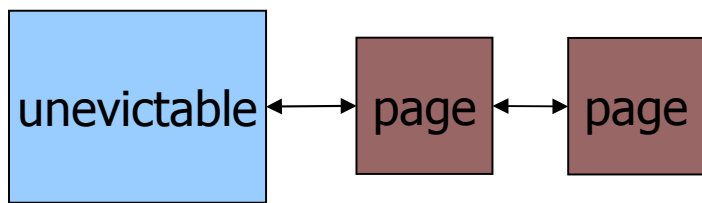
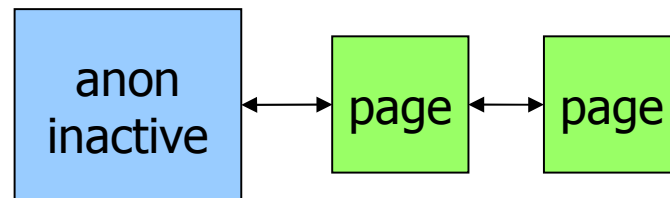
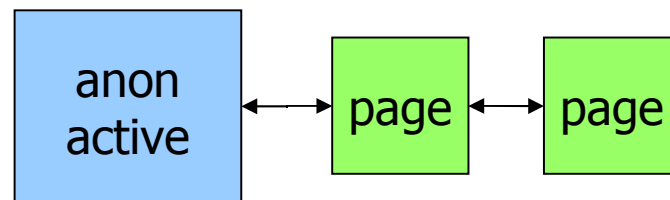
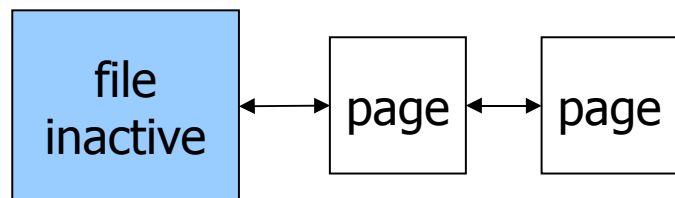
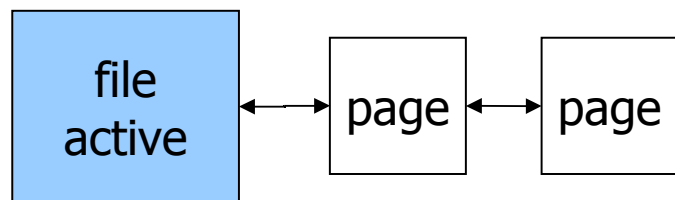
# 従来のLRU

いろいろな種類の回収ページが1つのリストに混じっている



Mlocked pageは常に回収できない。  
anon pageは低プレッシャーの時は  
回収すべきではない。

# Split LRU



リストが2個から5個へ

# スキャン量のバランシング

- LRUを分けたので、anonかfileかのどちらを優先してスキャンするのか方針が必要
- 平均して、スワップアクセスは通常ファイルの3倍遅い。スワップアウトよりファイルキャッシュ破棄を優先したい
- しかし、tmpfsのキャッシュが不要になることはよくある。anonをまったくスキャンしないのはNG
- Workloadに応じて動的に比率が変わるのが賢い
- 最近のスキャン数:最近の実際に捨てたページ数の比を使って調整

# 結果

- > Hi Rick,
- >
- > I just want to let you know your improvements in the
- > Memory manager work awesome on s390 when using
- > kvm.
- > Before, we were running into a limitation in vmscan
- > with **80 Websphere** servers running in a 45gig host.
- > Now we're limited by the bandwidth of the swap target
- > **200 Websphere** servers on the same host work well.

# Lockless pagecache

- 従来多CPUはハイエンドマシンにのみ存在、ハイエンドマシンは用途が限定的
- 時代はメニーコア。CPU数が増えればロック競合は増える
- 多CPUがローエンドに降りてきた
- lockless化への要求が増している
- 保守が難しくなるので明白な性能メリットがないとマージされない

# Lockless pagecache

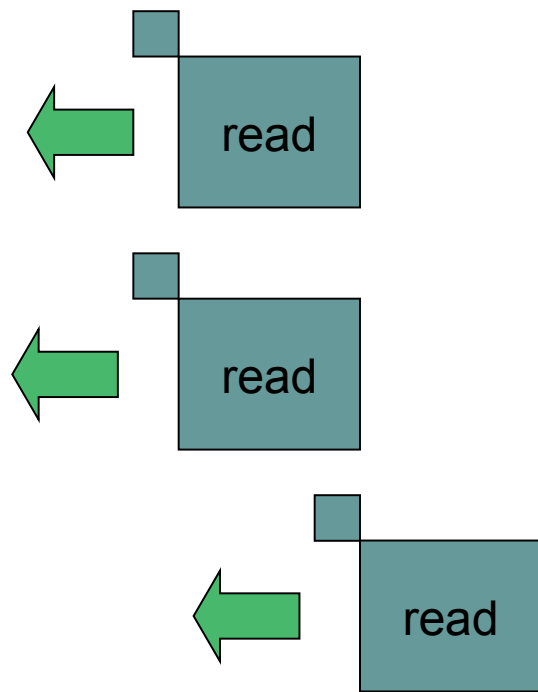
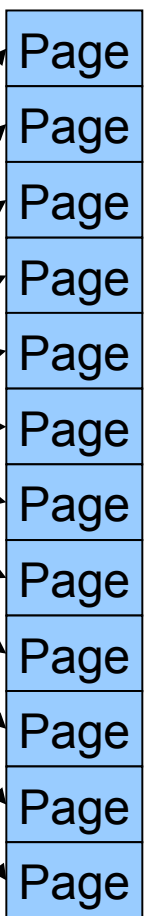
- 仮想OSのOSイメージや、通常ファイルとしてデータをストアするDBなど巨大ファイルが一般的に
- ファイル単位にロックがあり、競合が発生しやすかった

# rwlock starvation problem

従来のロック



ファイル



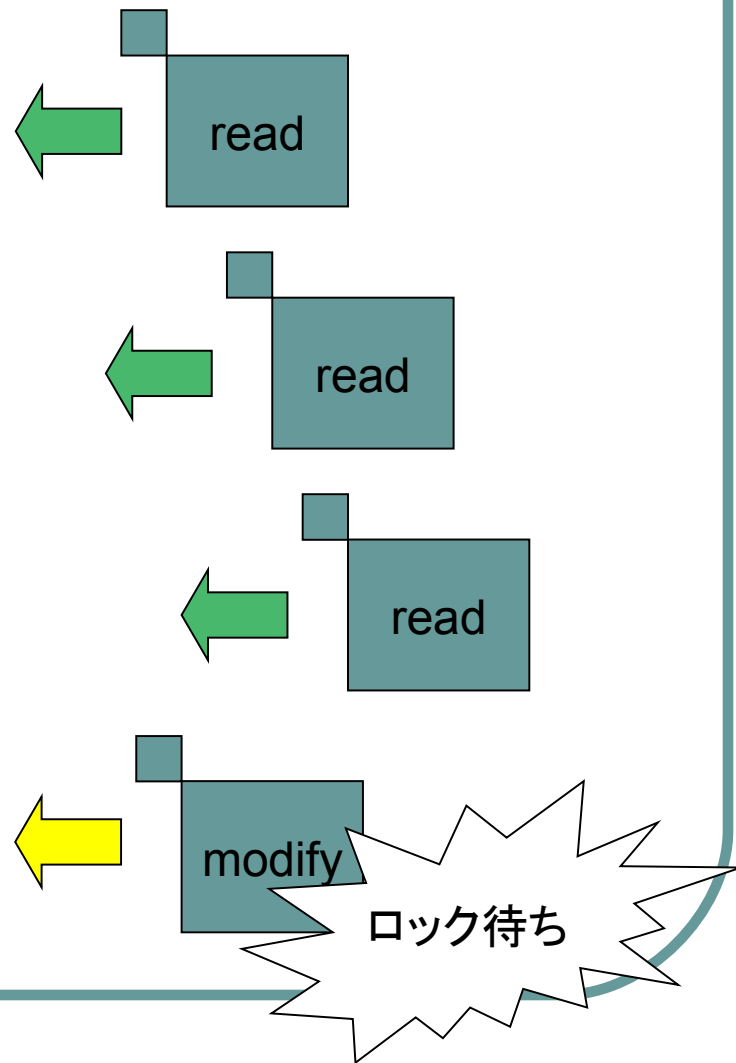
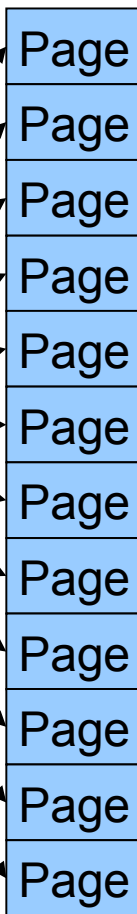
rwlockだからlookupは複数同時に可能

# rwlock starvation problem

従来のロック



ファイル



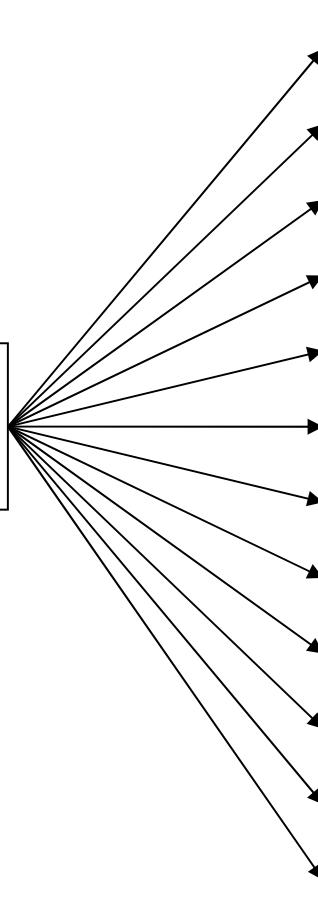
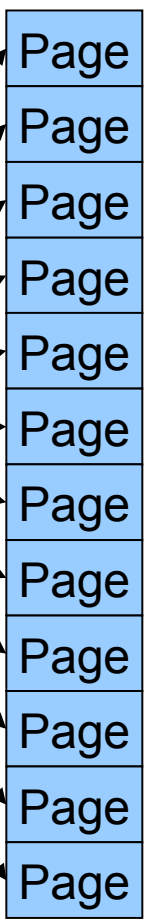
Cacheをremoveしたい。  
Readerがいなくなるまで  
待つ

# rwlock starvation problem

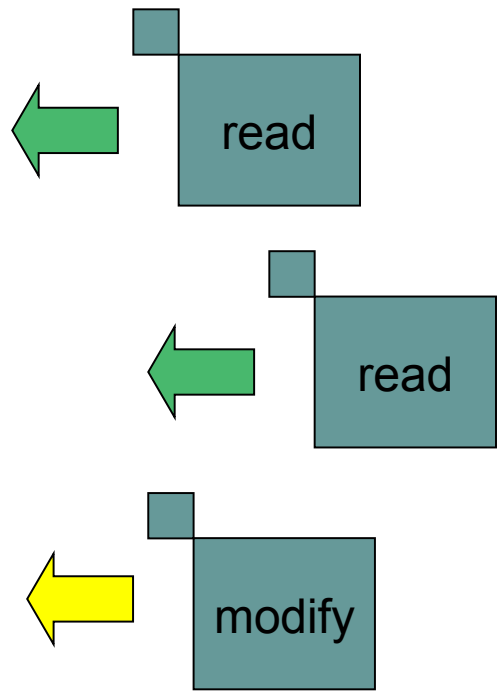
従来のロック



ファイル



さらに待つ

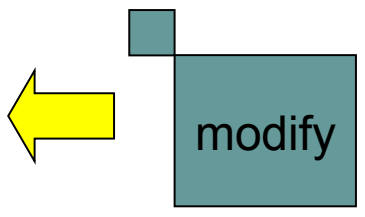
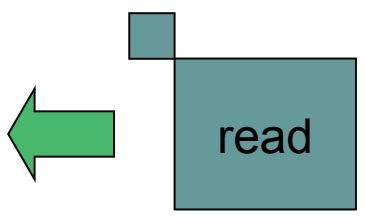
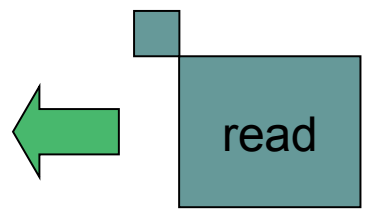
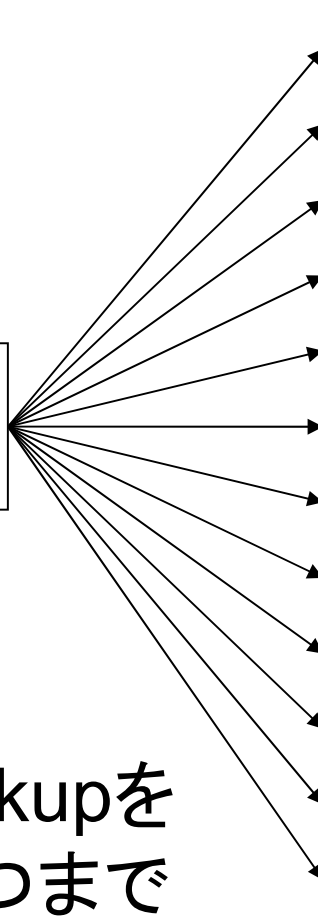
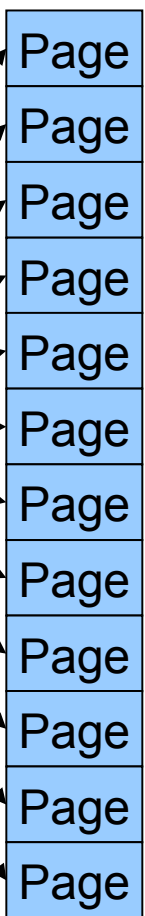


# rwlock starvation problem

従来のロック



ファイル

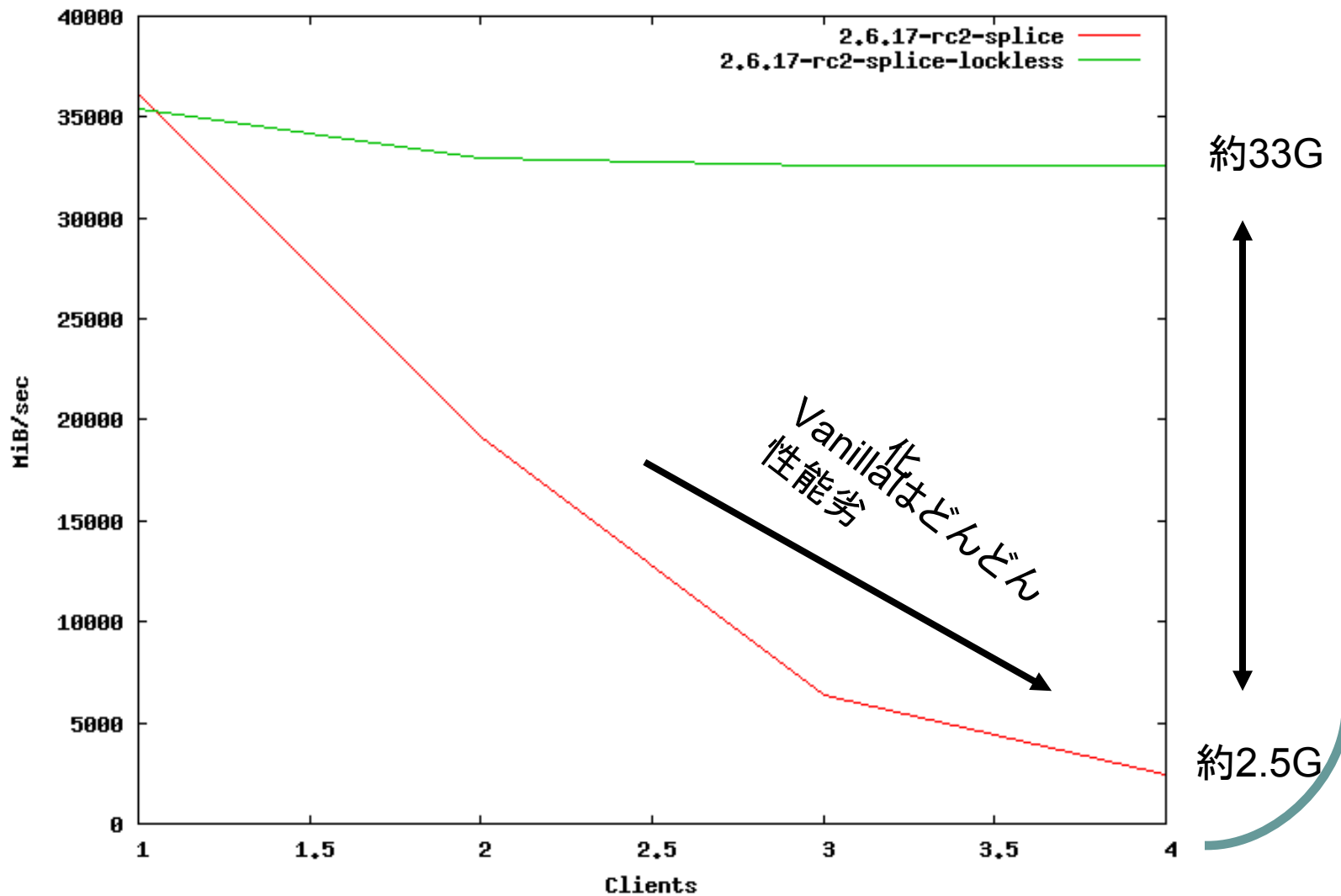


他のCPUがlookupを繰り返すと、いつまでたってもreaderが0にならない

# 何が起きるか？

- メモリ枯渇時にファイルキャッシュを回収する
- Cacheが外せないとメモリの空きが増えない
- メモリ回収がwrite lock starvationで待たされる
- 結局処理が進まなくなる
- メモリ不足が深刻化して、reader側の数が減ると解決、ライブロックにはならない。しかし、レイテンシは大きく悪化
- この部分をlockless化して競合を排除

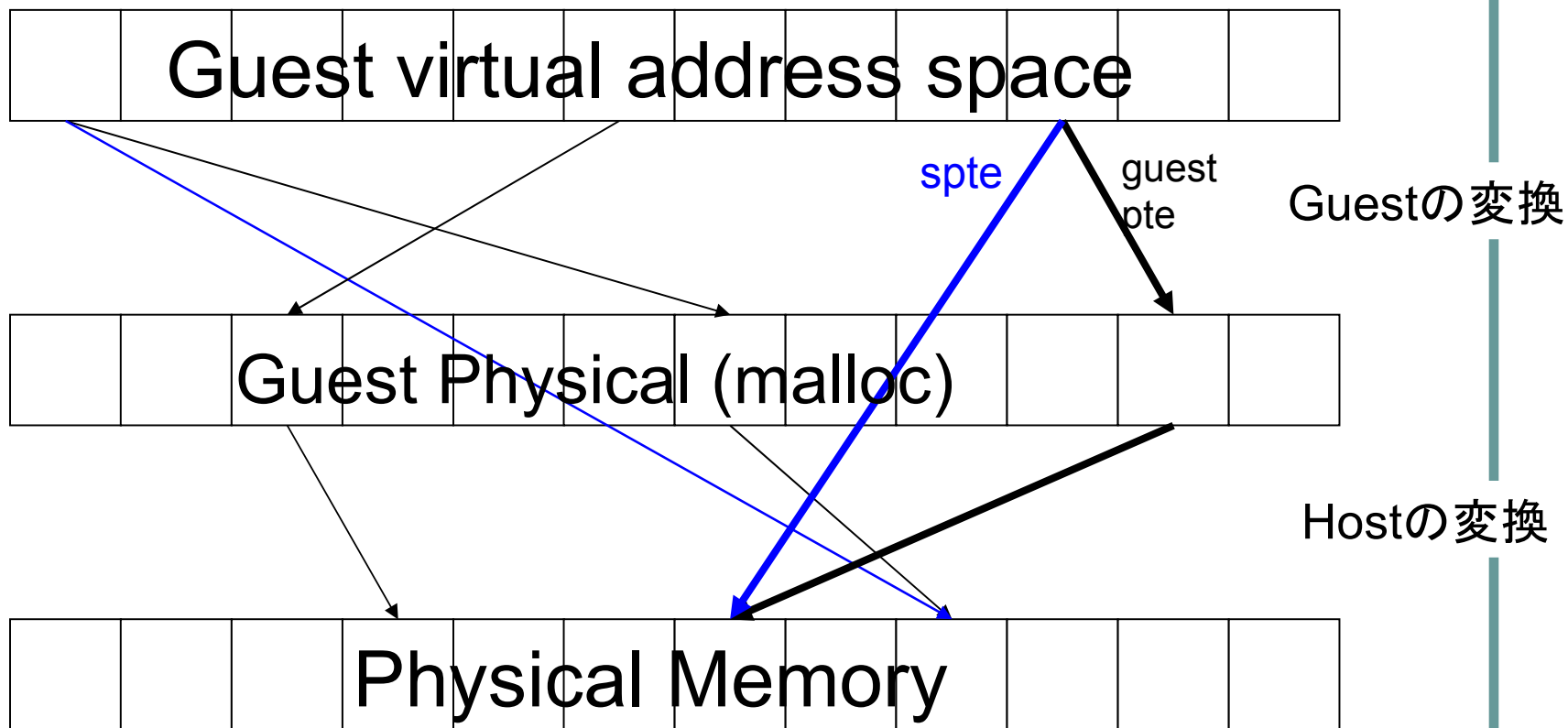
# RESULT



# MMU notifier

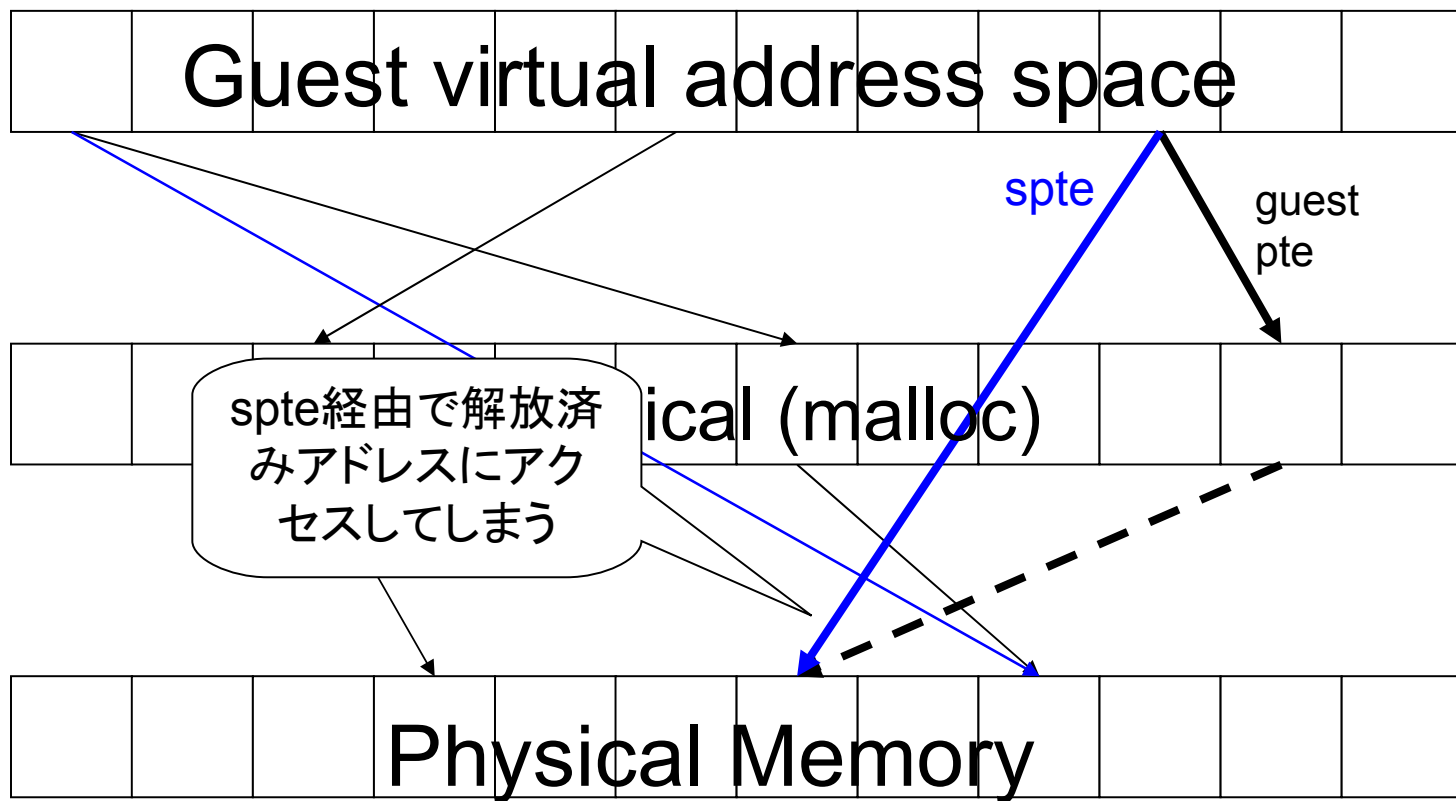
- 時代は仮想化。仮想化支援も流行の1つ
- 従来、KVMではGuestに割り当てたページをswap-outできなかった
- このパッチはKVM Guestのswap-outを可能にする
- Hostのswap-out時にKVMにspte(secondary pte)を無効にするための通知を送信する

# KVM spte (secondary pte) 基礎



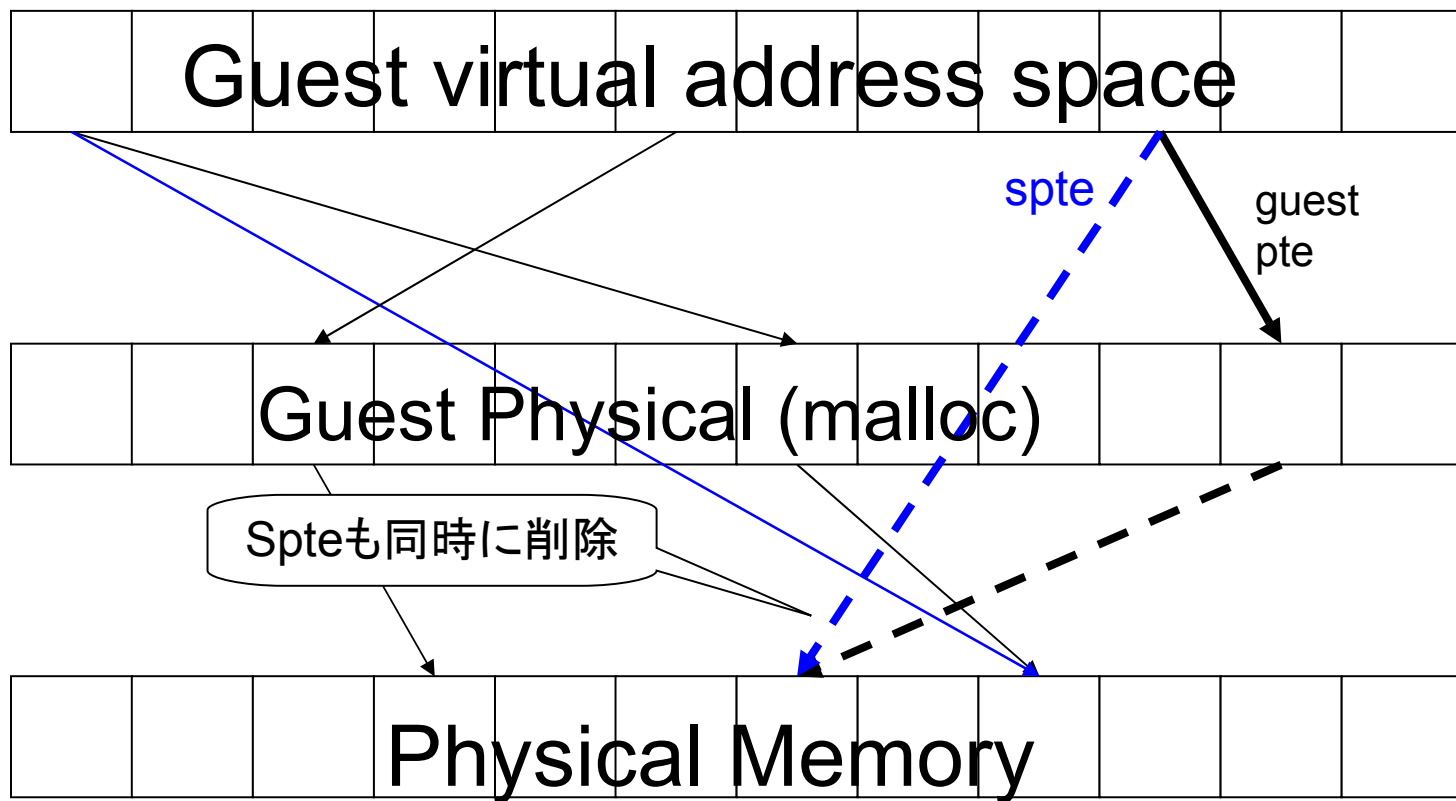
毎回2段階の変換は遅い。Guest virtual -> Physical  
の変換テーブルを隠し持つ。それがspte(青)

# KVM spte cont.



Guest Physical->Physical 変換はHost側でunmapできるが、spteはHostからは見えない。このままでは不正参照発生

# KVM spte cont.



Guest Physical(host virtual) unmap時にKVMに通知。  
KVMはspteを無効化。Guestアプリからは透過的

# Mem-cgroup

- 3ヶ月前に聞いたばかりよね？

# Reclaim bail out

- ページ回収時に回収しすぎるケースがあった。
- 活発なworkload(ゲームとか)を実行中はほとんどのpageに参照bitがたっている場合がある
- その場合、なかなか回収できないのでVMはスキャン量を上げていく
- トライするスキャン量が上がった後、回収可能ページが大量に見つかりと非常に多くのページを回収してしまう
- スキャン量だけではなく回収量によってもreclaim中止すべき。
- 一見、何も問題ない提案に見えるが...

NAK！理由は忘れた！

There was a reason for not doing this, but **I forget** what it was. It might require some changelog archeology.

-- Andrew Morton

# 開発者は納得しない

- 侃々諤々の議論のあと、理由なしのNAKはダメだよ。という話になりマージすることに
- Akpm: finger crossed (神に祈るわ)
- ところで、-mmにマージされるとパッチ作成者と関係者に通知メールが飛んでくる
- それを見ると……

# patch descriptionに加筆

akpm: a historical interlude...

We tried this in 2004:

```
:[PATCH] vmscan.c: dont reclaim too many pages  
(snip)
```

And we reverted it in 2006:

(snip)

**And we haven't demonstrated that whatever problem caused that reversion is not being reintroduced by this change in 2008.**

# いや、実は・・・

- > I have ML archive at that time and I think
- > I can explain why not reintroduced.
- > please wait soon.

thanks - I was hoping you would ;)

# 結果

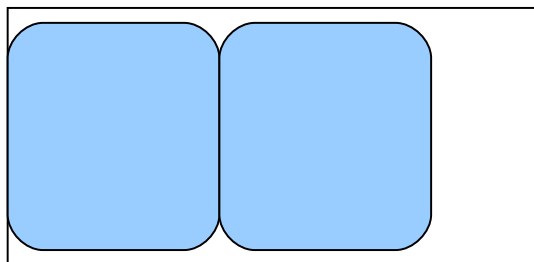
Hackbenchでメモリを使い切るぐらい大量にプロセスを生成  
(単位:秒)

|     | vanilla | bail out |           |
|-----|---------|----------|-----------|
| avg | 185.4   | 142.0    | (30% up!) |
| std | 74.2    | 55.9     |           |
| min | 46.1    | 62.5     |           |
| max | 305.0   | 227.8    |           |

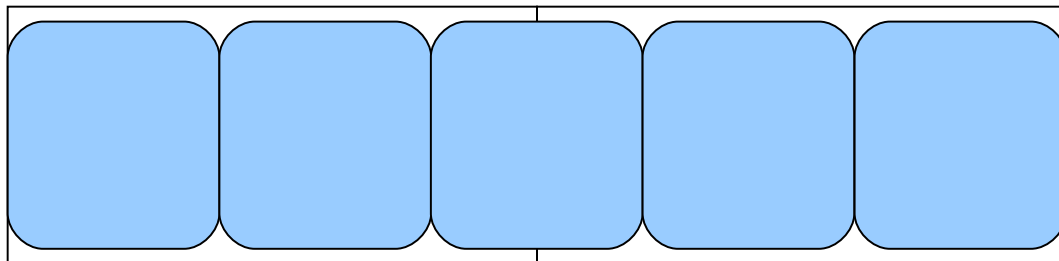
# SLQB

- いい加減SLABを捨てたい
- SLUBが未だにSLABに負けるケースがある。
- SLQB: SLUBより高速なアロケータだと表明
- パラレルに進む様々な議論。とても追えない
- ベンチマーク競争大流行。SLUB, SLQBともにどんどんコードが変わる。
- 勝負の行方は不透明。しかし、どちらが勝ってもユーザにはメリットあり

# 例：High order allocation 是非論争



Page\_size ÷ object\_size が端数が出る  
ことがある



複数のページをくっつけると、  
端数を減らすことができる

何ページ用意するのが最適か？

# Benchmarks

Benchmarks are **essential**, please. Good ones.

-- Andrew Morton

Questions?

# なんでマージしないの？

Yes, if this is really in use by everybody,  
then **not merging it is kind of pointless.**

-- Linus