フラッシュSSDをメモリセマンティクスAPIで用いるための予備調査

丹英之†§ ， 緑川博子‡§
†（株）アルファシステムズ ‡成蹊大学 §JST, CREST

背景と目的

- 巨大な問題サイズのジョブを動かしたい
- NANDフラッシュをはじめとした不揮発性メモリの台頭
  - 省電力・大容量・遅いメモリとして利用する
- PC接続のフラッシュメモリはフラッシュストレージ

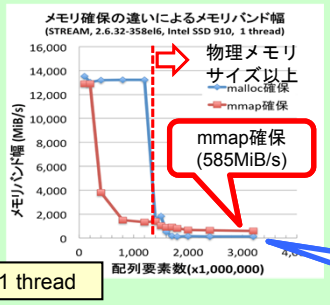
フラッシュメモリを拡張メモリとするには

- malloc確保: malloc(3)で確保，仮想メモリのページイン・アウト先のスワップ用ブロックデバイスにする．
- mmap確保: mmap(3)で確保，マップファイルを置くファイルシステムを載せるブロックデバイスとする．

フラッシュ向けLinuxスワップシステム(nvm-fast-swap)

- Fusion-io社がOpenNVMプロジェクトで提案
  (Flash Memory Summit 2013, http://opennvm.github.io/)
- DRAMのサイズを超えたワークロードの実行が目的
- スワップシステムにあるロックを細粒度化
  - グローバルロックの廃止
  - スワップパーティションやCPUごと割り当て

malloc確保/mmap確保でのメモリバンド幅(Previous Work)



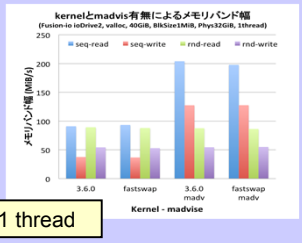
- これまでの知見では，“物理メモリサイズを越える問題サイズであれば，mmap確保がよい”であった．
- 差はアプリ次第
  - メモリアクセス特性
  - 演算/メモリアクセス比
  - コア間データ共有度
- malloc確保では物理メモリサイズを超えた時点で急激に性能が低下．

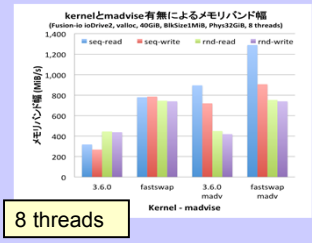
計測環境

- Fusion-io ioDrive2(1.2TB, MLC) / Intel SSD 910(400GB)
- Xeon E5-2650 2.0GHz x1 (8Core)，DDR3-1600 ECC 8GB x4 (32GB)
- CentOS 6.4, gcc 4.4.7, -O3
- fastswap(nvm-fast-swap) / 3.6.0(vanilla)

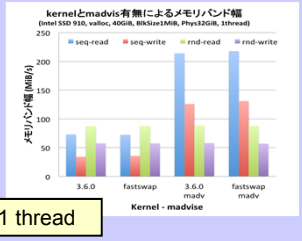
madvise有無-スレッド数の違いによるメモリバンド幅

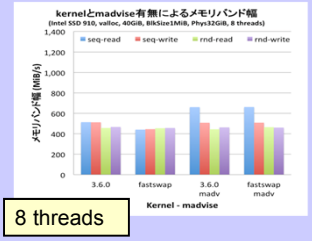
Fusion-io ioDrive2





Intel SSD 910





一部がページアウト状態でのメモリバンド幅の計測

- 各デバイスをスワップデバイスに指定
- 40GiBのデータをvalloc(3)，物理メモリ率75%
- 一旦ランダムデータを書き込む
- 確保した領域をmadvise(2)
  - なし/MADV_SEQUENTIAL/MADV_RANDOM
- memcpy(3)でIOブロック単位(1MiB)で読み書き
- シーケンシャル/ランダム
- 1 thread / 8 threadsでの操作
- 40GiBの領域全ての操作に要する時間を計測

ioDrive2, SSD910の 1 thread

- fastswapの効果は見られない．
- シーケンシャルアクセスにはMADV_SEQUENTIAL指定の効果がある．一方，ランダムアクセスにMADV_RANDOM指定は効果が見られず．

ioDrive2の8 threads

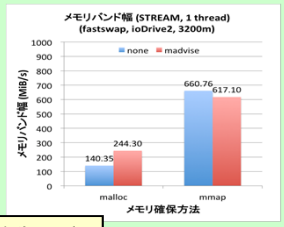
- fastswapは3.6.0の約2倍のメモリバンド幅．
- 3.6.0のmadvise(2)は効果あるがシーケンシャルアクセスだけ．
- シーケンシャルアクセスにはMADV_SEQUENTIAL指定の効果がある．一方，ランダムアクセスにMADV_RANDOM指定は効果が見られず．

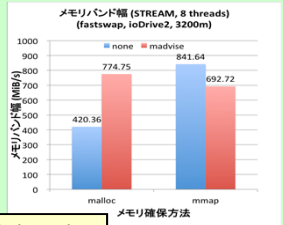
SSD910の8 threads

- 3.6.0の方が若干よい．
- madvise(2)も同程度．
- メモリアクセスパターンがシーケンシャルであればmadvise(2)指定は効果的．
- fastswapはマルチスレッド実行でioDrive2の性能を引き出せる．

STREAM

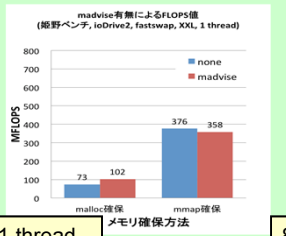
配列要素数3200m: 71.5GiB, 物理メモリ率42%

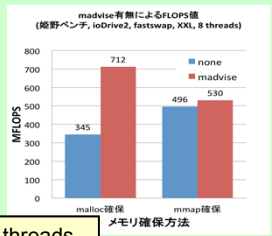




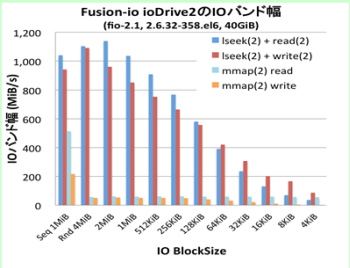
姫野ベンチマーク

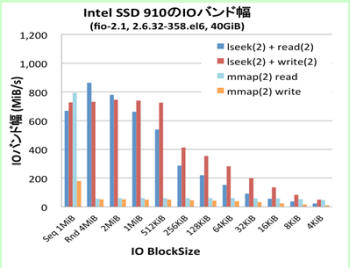
問題サイズXXL(1024x1024x2048): 112GiB, 物理メモリ率28%





基本I/O性能





基本 I/O 性能の計測

- fio-2.1，ファイルサイズ40GiB
- IO操作方法
  - lseek(2)+read(2)/write(2)
  - mmap(2)+memcpy(3)

- mmapアクセスが極端に遅い
  - マップした領域全てにmadvise(2)していた．
    - MADV_SEQUENTIAL/MADV_RANDOM
  - IO操作ごとにMS_SYNCでmsync(2)していた．
  - これらを外すとlseek + RWの半分程度のIOバンド幅を得られた．
- madvise(MADV_RANDOM)は危険?