https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

第18回FreeBSDワークショップ

佐藤　広生 <hrs@FreeBSD.org>

東京工業大学/ FreeBSD Project

2016/6/24

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

開催背景

- **日本国内の＊BSDユーザ活動を活発化させましょう**
  - 月1回、東京近辺で定期的な会合を。
  - 講演を聞くだけでなく、話を持ち寄って双方向に議論しましょう

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

本ワークショップの進行

- 19:00〜19:30　自己紹介＋話題にしたいトピック
- 19:30〜20:00　ライトニングトーク
- 20:00〜20:10　休憩
- 20:10〜21:15　ZFSチューニングの話

意見は自由に発言ください！

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

オーガナイザの自己紹介

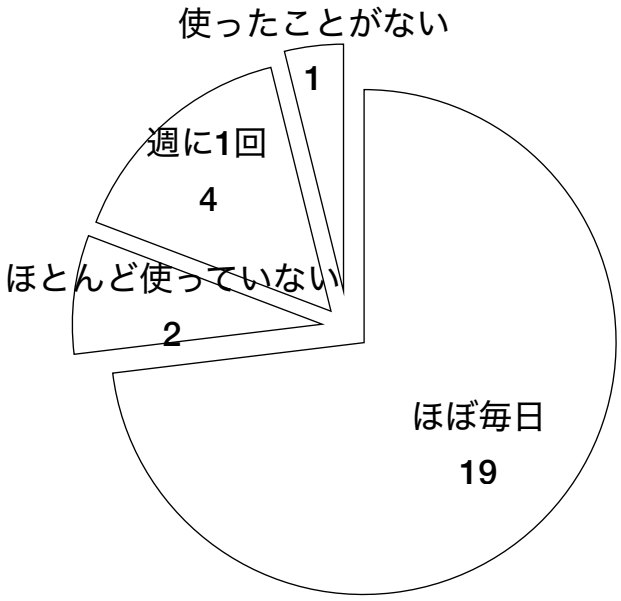
- 名前：佐藤 広生
  - FreeBSD コアチームメンバ、リリースエンジニア(2006-)
  - FreeBSD Foundation 理事(2008-)
  - その他の＊BSD／オープンソース関連の活動いろいろ
  - 東京工業大学助教(2009-)

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

自己紹介タイム

- 名前（所属）
- 開発者 or 利用者
- 興味がある／話題にしたい内容

  をどうぞ



今回の出席者内訳：新規5名、再参加者21名

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

メモ

メモ

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの性能ボトルネックの調べ方とチューニング法

佐藤　広生 <hrs@FreeBSD.org>

東京工業大学/ FreeBSD Project

2016/6/24

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

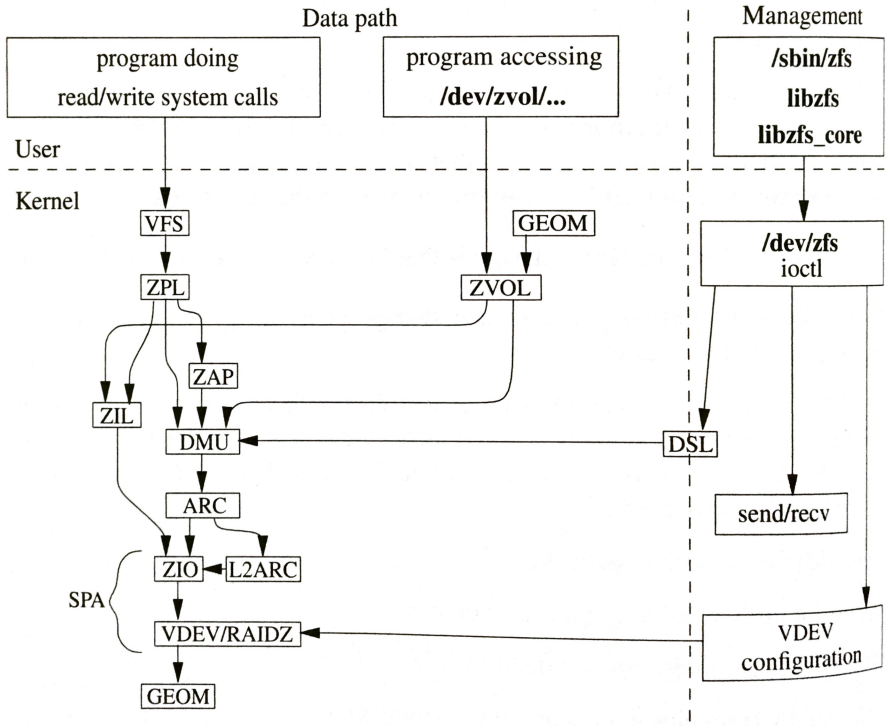
お話しすること

- システム管理者が知っておくべきZFSの動作
- 性能の測定
- チューニングすべき項目

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.

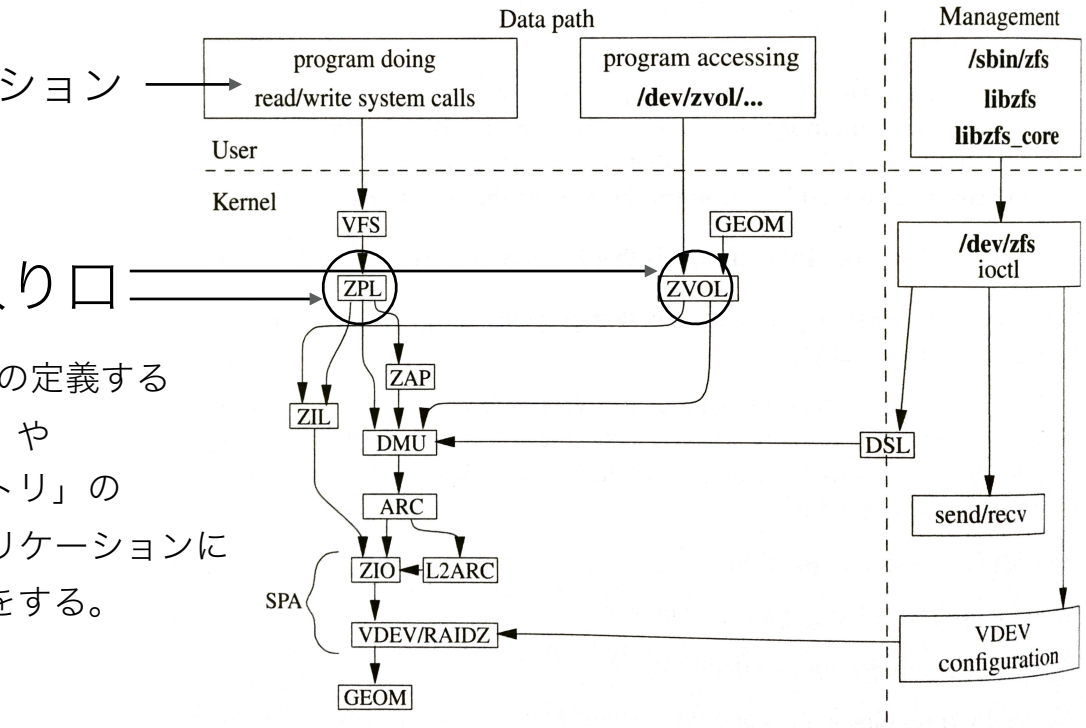


D&I 本, p.526

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.



アプリケーション

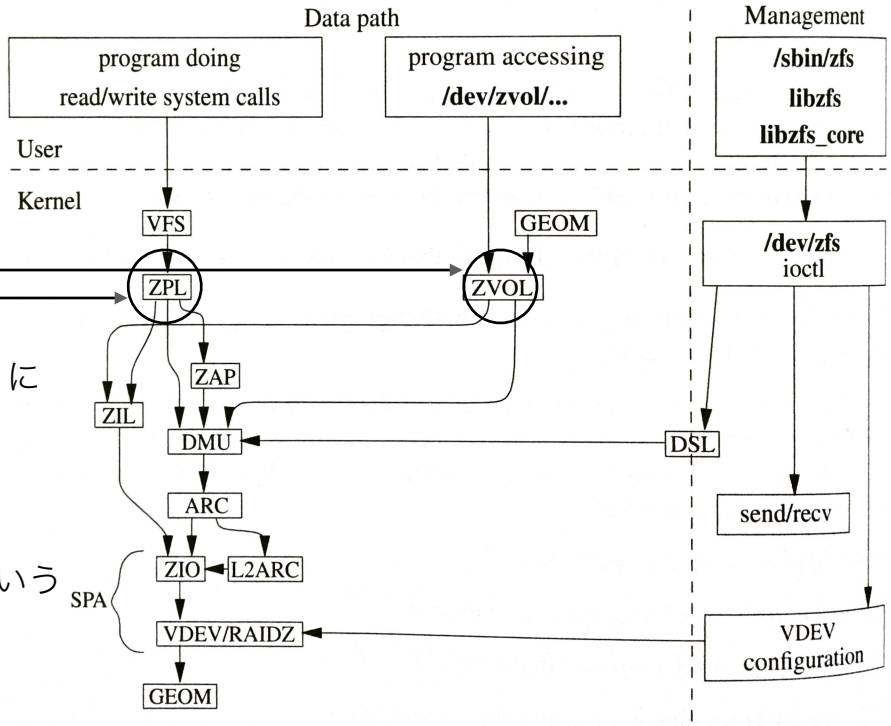
ZFSの入り口

ZPLはPOSIXの定義する
「ファイル」や
「ディレクトリ」の
構造をアプリケーションに
見せる処理をする。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.



ZFSの入り口

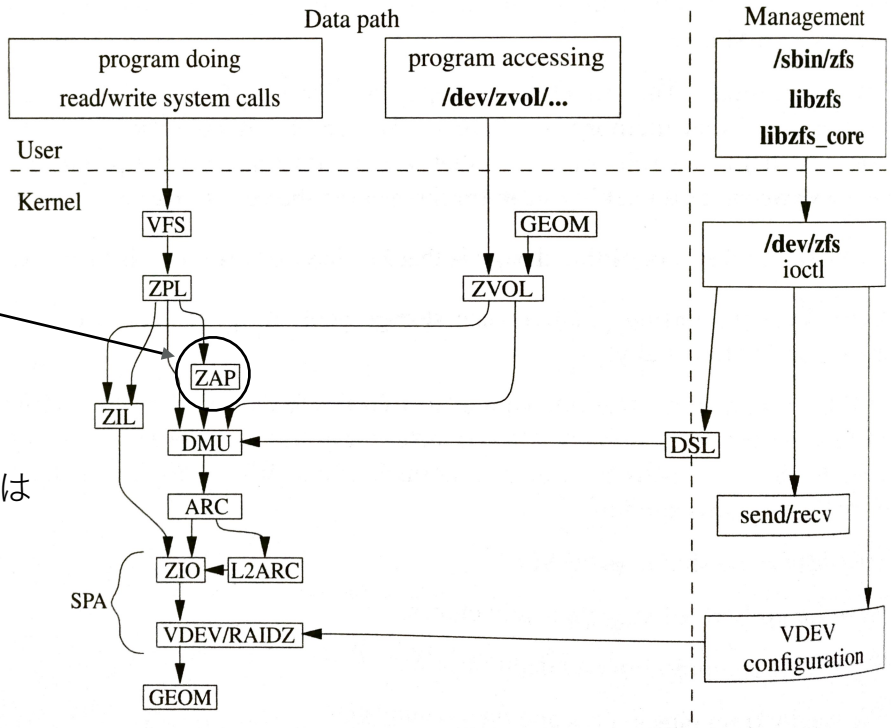
ZVOLは、/dev/da0 のように
ブロックデバイスとして
見える姿を処理する。

実際には"/dev/zvol/xxx"という
名前で見える。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.



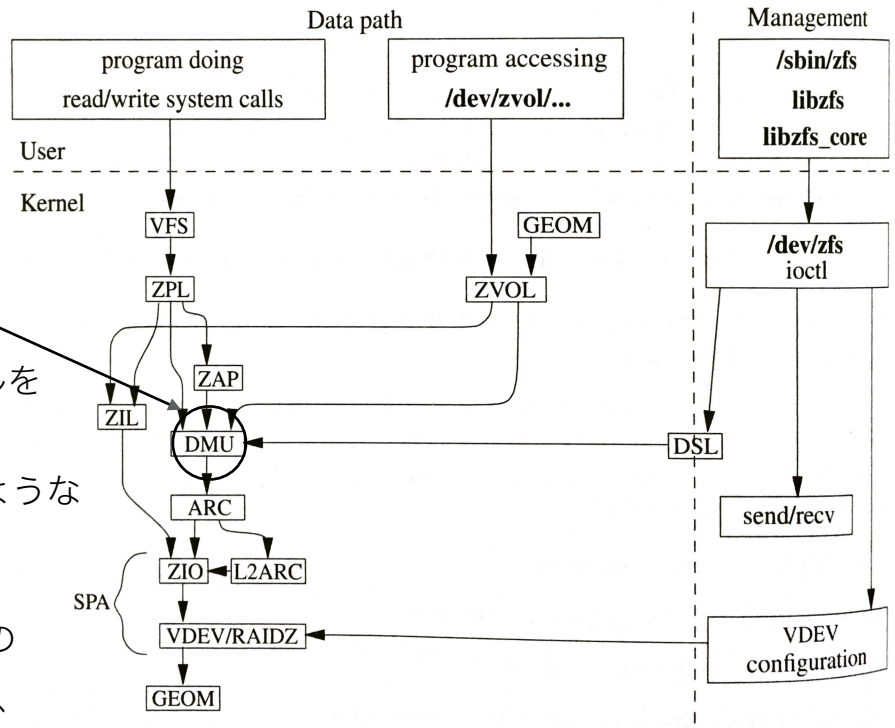
メタデータ処理

ZAPは、key-valueペアの
処理を担うところ。
ディレクトリや属性などは
ここで処理される。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.



DMU

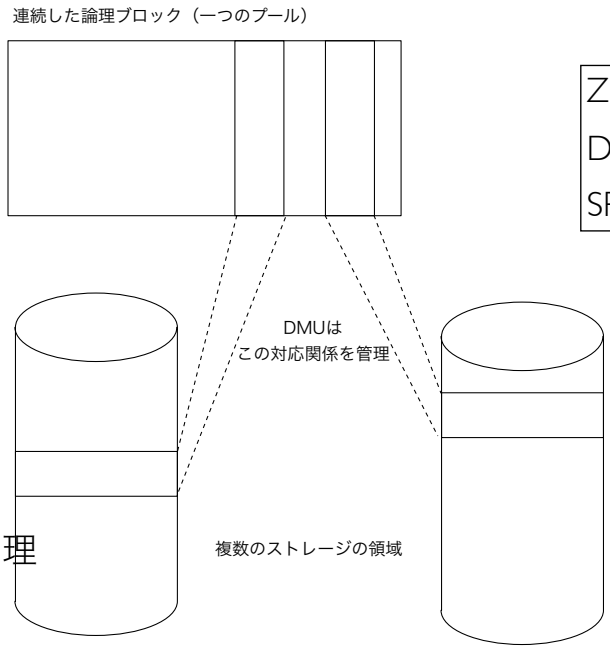
DMUは、ストレージプールを
管理する部分。
仮想メモリのMMUと同じような
発想だと思っていて良い。

DMUの上側は、HDDなどの
デバイスの存在を意識せず、
データはプールの論理ブロック番号で管理される。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

- DMUが管理している領域のイメージ



DMUはdnodeで管理
（inodeみたいなもの）

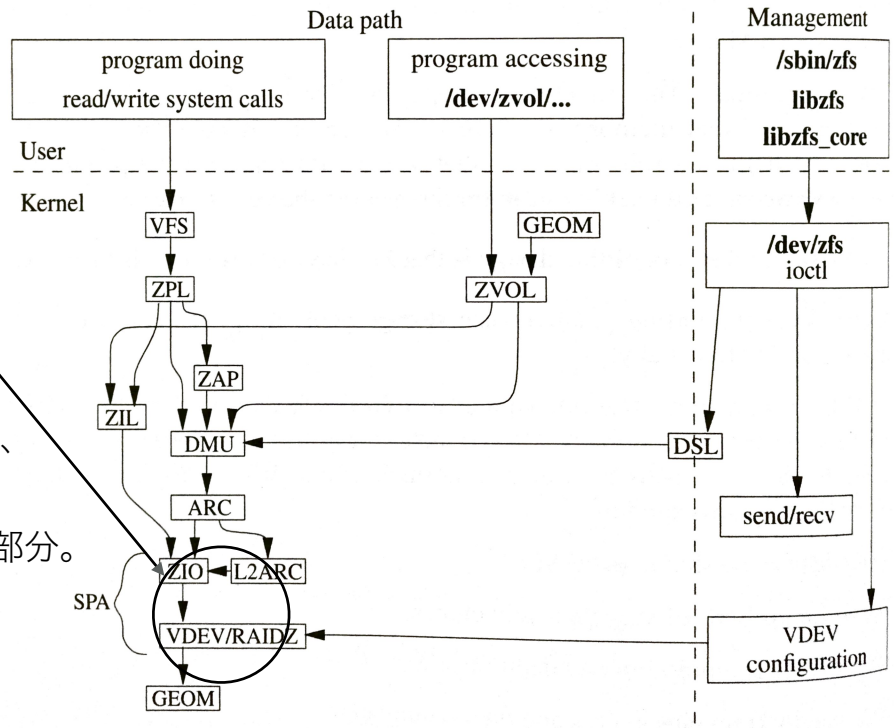
ZPLはznode,
DMUはdnode,
SPAはブロックポインタ

ストレージの領域は
ブロックポインタで管理

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの構造

Figure 10.2 ZFS module layering. See Table 10.1 for acronyms.



SPA

SPAは、DMUで読み書きが
必要になった領域に対して、
それを disk I/O に翻訳して
I/O リクエストを発行する部分。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSの動作の特徴

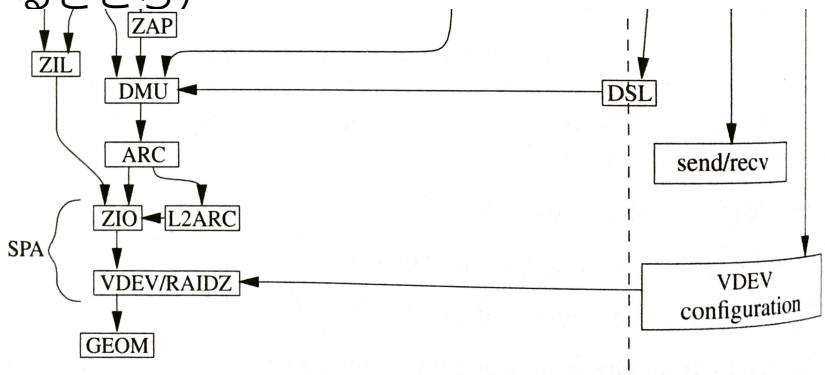
- ファイルアクセスは、DMUを通過する
- DMUの処理は、「連続する論理ブロック（＝プール）」に
  対応するメモリ空間への操作になる
- メモリ空間への操作がSPAによって
  disk I/O に変換され、実際のストレージとの
  データのやり取りが行われる
- つまり、でかいmmap()のようなもの

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

性能を決めるのはどこか？

- SPAの効率が性能に大きく影響するため、
  その動作を知ることが重要！

- DMUより下位の要素：
  ARC（キャッシュ）
  ZIO（I/Oを発行するところ）



https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：読み出し

- ZFSはUFSと比較して、データブロックを読み出すまでに
  ブロックポインタをたどる回数が多い
  (uberblock -> objset -> objset -> ... -> data)

- 同一の処理に対して、より多くのIOPSが必要ということ！

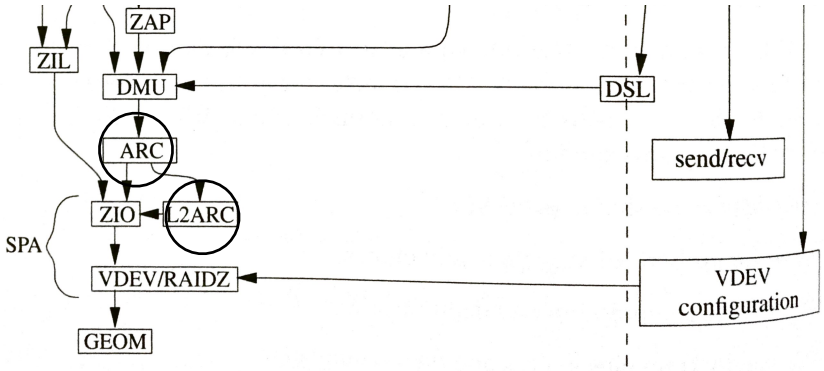
- ほとんどの場合、キャッシュによってその差は見えなくなる

- メタデータのキャッシュヒット率が低下すると、
  UFSと比較してかなり遅くなってしまうことがある

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：読み出し

- キャッシュはどこにある？　＝　DMU, ARC, L2ARC
- DMUのキャッシュはdnodeレベル、
  ARC, L2ARCはブロックポインタレベルで保持する
- ARCはメモリ、
  L2ARCは
  HDD, SSDなど
  の記憶装置



https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：読み出し

- ヒット率を調べるには？
- sysutils/zfs-stats をインストールして zfs-stats(8)を
  使いましょう

```
hrs@pool % zfs-stats -E

------------------------------------------------------------------------
ZFS Subsystem Report                            Fri Jun 24 15:31:52 2016
------------------------------------------------------------------------

ARC Efficiency:                                 9.16b
        Cache Hit Ratio:                87.49%  8.01b
        Cache Miss Ratio:               12.51%  1.15b
        Actual Hit Ratio:               86.94%  7.97b

        Data Demand Efficiency:         79.96%  98.35m
        Data Prefetch Efficiency:       35.42%  35.41m

        CACHE HITS BY CACHE LIST:
          Most Recently Used:           0.93%   74.29m
          Most Frequently Used:         98.45%  7.89b
          Most Recently Used Ghost:     0.61%   48.86m
          Most Frequently Used Ghost:   0.23%   18.37m

        CACHE HITS BY DATA TYPE:
          Demand Data:                  0.98%   78.64m
          Prefetch Data:                0.16%   12.54m
          Demand Metadata:              95.33%  7.64b
          Prefetch Metadata:            3.53%   283.22m

        CACHE MISSES BY DATA TYPE:
          Demand Data:                  1.72%   19.71m
          Prefetch Data:                2.00%   22.87m
          Demand Metadata:              84.43%  967.76m
          Prefetch Metadata:            11.86%  135.91m

------------------------------------------------------------------------
```

ZFSのI/O：読み出し

- Demand Metadata が 90% を切っていると
  ちょっと厳しい
- キャッシュに使うメモリが足りないか、
  メタデータに割り当てている量が足りない

- 調整１：
  ZFSプロパティのprimarycache, secondarycache
- デフォルトは all になっている。metadataに設定すると、
  メタデータだけをキャッシュするようになる。
- secondarycache は L2ARCの設定。
- 注：primarycacheをmetadataにすると、L2ARCは
  allにしてもmetadataだけになる

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：読み出し

- Demand Metadata が 90% を切っていると
  ちょっと厳しい
- キャッシュに使うメモリが足りないか、
  メタデータに割り当てている量が足りない

- 調整２：
  sysctl の vfs.zfs.arc_meta_limit
- ARCのうち、メタデータに使う量（バイト単位）
- デフォルトは全物理メモリの 1/4 に設定される
- vfs.zfs.arc_meta_limit="9G" のように指定できる

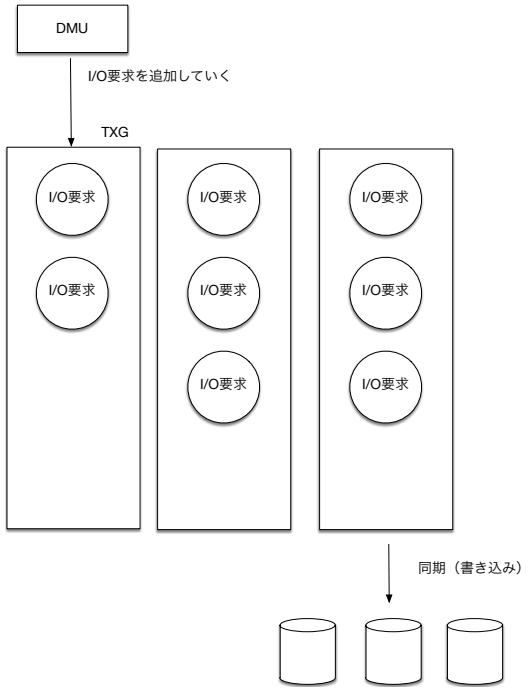
https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- ZFSは、特に書き込みの性能にチューニングが必要！

SPAの処理構造

- DMUで変更があった部分 (dirty data) は、I/O要求として蓄積されていく
- この単位をTXGと呼ぶ
- TXGは最大3つ。
  Syncing TXGというTXGにある要求が処理される



https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- SPAの同期
  - TXGは、
    「処理データが一定量に達する」か、
    「一定時間が経過する」かのいずれかで
    ストレージへの同期を行う
  - 一定量とは、全物理メモリ量の1/10である。
  - 一定時間とは、5 秒である。

- 例えば16GBのRAMならどうなる？
  - 5秒間に最大1.6GB/sの書き込みが生じる！
  - HDDは間に合うか？　自分の用途でこの速度は十分か？

ZFSのI/O：書き込み

- **同期が間に合わないとどうなる？**
  - 重要：同期が終わるまで、ZFSの読み出しI/Oは
    ブロックされてしまう
  - 典型的な症状：
    - 書き込み負荷が高くなると、数秒おきにカクンカクンとシステムの
      応答が極端に遅くなったり、止まって見える
    - 止まって見えなくても、書き込み速度はSPAで律速になってしまう
  - 大きなメモリ、遅いストレージの組み合わせが要注意

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf
ZFSのI/O：書き込み

- **チューニングするには**
  - 原則：想像だけでチューニングしないこと
    「とりあえず変更できるところを全部変更して様子をみよう」、
    「誰かの設定をコピーしよう」、というのは愚の骨頂です。
  - チューニング前に、同期が間に合っているのかを調べよう

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- **性能の測定**
  - DTraceを活用！大変便利です。
- まず、自分の使いたい負荷をかけた状態でディスクI/Oを測定

```
# dtrace -s diskio.d -c "sleep 30"
```

https://people.allbsd.org/~hrs/FreeBSD/diskio.d

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

```
# dtrace -s diskio.d -c "sleep 30"
  write
           value  ------------- Distribution ------------- count
              16 |                                         0
              32 |                                         31
              64 |@@@@@                                    505
             128 |@@@@@@@@@@@@@@@@@@@@@@@@@@               2531
             256 |@@@@@@                                   588
             512 |@@                                       223
            1024 |@                                        57
            2048 |                                         46
            4096 |@                                        71
            8192 |                                         22
           16384 |                                         34
           32768 |@                                        69
           65536 |                                         27
          131072 |                                         0

  read
           value  ------------- Distribution ------------- count
              16 |                                         0
              32 |                                         2
              64 |@@@@@@@@@@@@                             4404
             128 |@@@@@@@@@@@@@@@@@                        5937
             256 |                                         131
             512 |                                         33
            1024 |                                         116
            2048 |@                                        226
            4096 |@                                        510
            8192 |@@@@                                     1546
           16384 |@@@@                                     1418
           32768 |@                                        323
           65536 |                                         44
          131072 |                                         7
          262144 |                                         0

                               avg latency      stddev        iops  throughput
write                               1974us      9683us       140/s     1632k/s
read                                5043us     10832us       489/s     1394k/s
```

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

```
                                avg latency      stddev        iops  throughput
write                             1974us       9683us       140/s     1632k/s
read                              5043us      10832us       489/s     1394k/s
```

- この結果を確認し、チューニング前後でIOPSと
  スループットが増加するかどうかを調べること

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- **性能の測定**
  - DTraceを活用！大変便利です。

- 次に、Syncing TXGの処理が間に合ってるのかどうか確認

```
# dtrace -s delayed-io.d -c "sleep 30"
dtrace: script 'delayed-io.d' matched 1 probe
dtrace: pid 58771 has exited

  no delay                                                        12151
```

注：no delay だけなら問題無し

`https://people.allbsd.org/~hrs/FreeBSD/delayed-io.d`

ZFSのI/O：書き込み

- delayed が出ているようなら、間に合ってないので
  latencyを測定する。

```
# dtrace -s delay-mintime.d -c "sleep 30"
```

latencyのヒストグラムが出ます。

https://people.allbsd.org/~hrs/FreeBSD/delay-mintime.d

ZFSのI/O：書き込み

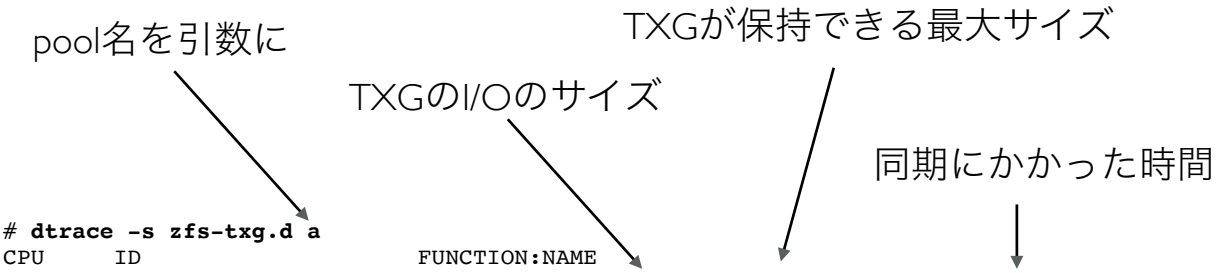
- 自分の負荷の基礎情報が揃ったので、
  次にSyncing TXGの動作を見てみましょう。

```
# dtrace -s zfs-txg.d a
CPU     ID                    FUNCTION:NAME
  8  57917                 none:txg-synced   64MB of 2454MB synced in 0.22 seconds
 11  57917                 none:txg-synced  702MB of 2454MB synced in 0.20 seconds
  7  57917                 none:txg-synced  444MB of 2454MB synced in 0.17 seconds
 11  57917                 none:txg-synced  442MB of 2454MB synced in 0.16 seconds
  4  57917                 none:txg-synced  384MB of 2454MB synced in 0.18 seconds
  1  57917                 none:txg-synced  478MB of 2454MB synced in 0.47 seconds
  2  57917                 none:txg-synced 1320MB of 2454MB synced in 0.75 seconds
  1  57917                 none:txg-synced 1672MB of 2454MB synced in 0.62 seconds
  8  57917                 none:txg-synced 1502MB of 2454MB synced in 0.26 seconds
  0  57917                 none:txg-synced  500MB of 2454MB synced in 0.19 seconds
  0  57917                 none:txg-synced  500MB of 2454MB synced in 0.22 seconds
  3  57917                 none:txg-synced  501MB of 2454MB synced in 0.24 seconds
  8  57917                 none:txg-synced  615MB of 2454MB synced in 0.34 seconds
  8  57917                 none:txg-synced  886MB of 2454MB synced in 0.28 seconds
  6  57917                 none:txg-synced    3MB of 2454MB synced in 0.21 seconds
```

https://people.allbsd.org/~hrs/FreeBSD/zfs-txg.d

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

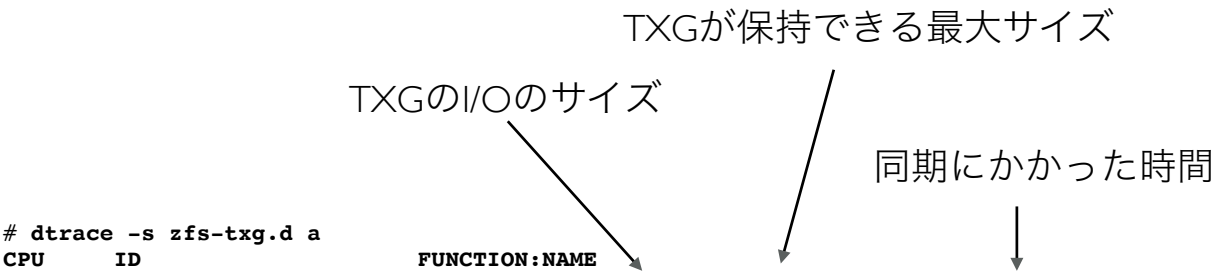



```
# dtrace -s zfs-txg.d a
CPU     ID                    FUNCTION:NAME
  8  57917                 none:txg-synced   64MB of 2454MB synced in 0.22 seconds
 11  57917                 none:txg-synced  702MB of 2454MB synced in 0.20 seconds
  7  57917                 none:txg-synced  444MB of 2454MB synced in 0.17 seconds
 11  57917                 none:txg-synced  442MB of 2454MB synced in 0.16 seconds
  4  57917                 none:txg-synced  384MB of 2454MB synced in 0.18 seconds
  1  57917                 none:txg-synced  478MB of 2454MB synced in 0.47 seconds
  2  57917                 none:txg-synced 1320MB of 2454MB synced in 0.75 seconds
  1  57917                 none:txg-synced 1672MB of 2454MB synced in 0.62 seconds
  8  57917                 none:txg-synced 1502MB of 2454MB synced in 0.26 seconds
  0  57917                 none:txg-synced  500MB of 2454MB synced in 0.19 seconds
  0  57917                 none:txg-synced  500MB of 2454MB synced in 0.22 seconds
  3  57917                 none:txg-synced  501MB of 2454MB synced in 0.24 seconds
  8  57917                 none:txg-synced  615MB of 2454MB synced in 0.34 seconds
  8  57917                 none:txg-synced  886MB of 2454MB synced in 0.28 seconds
  6  57917                 none:txg-synced    3MB of 2454MB synced in 0.21 seconds
```

https://people.allbsd.org/~hrs/FreeBSD/zfs-txg.d

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み




```
# dtrace -s zfs-txg.d a
CPU     ID                    FUNCTION:NAME
  8  57917                 none:txg-synced   64MB of 2454MB synced in 0.22 seconds
```

書き込みI/Oがあれば、少なくとも5秒で1回の同期が発生する。

一回の同期が5秒以上かかっている　＝　間に合ってない。

https://people.allbsd.org/~hrs/FreeBSD/zfs-txg.d

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- チューニングはどうやったら良いの？
  - 1) TXGのサイズ調整
  - 2) TXGの同期I/O発行量
  - 3) TXGのタイムアウト

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- 1) TXGのサイズ調整

- 同期までに溜め込むI/Oの量は、次のsysctlで決まっている

vfs.zfs.dirty_data_max: 2573481984
　溜め込む量（バイト）。
vfs.zfs.dirty_data_max_percent: 10
　data_maxをメモリ量から起動時に自動計算する時の%値。
vfs.zfs.dirty_data_max_max: 4294967296
　data_max_percentで計算された値の上限。

- これらを増減させると、一回の同期量が変わる。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- 2) TXGの同期I/O発行量
- 同期用のI/Oを、HDDやSSDにいくつ並行で送るかという数
- sysctl とデフォルトは次の通り

vfs.zfs.vdev.trim_max_active: 64
vfs.zfs.vdev.trim_min_active: 1
vfs.zfs.vdev.scrub_max_active: 2
vfs.zfs.vdev.scrub_min_active: 1
vfs.zfs.vdev.async_write_max_active: 10
vfs.zfs.vdev.async_write_min_active: 1
vfs.zfs.vdev.async_read_max_active: 3
vfs.zfs.vdev.async_read_min_active: 1
vfs.zfs.vdev.sync_write_max_active: 10
vfs.zfs.vdev.sync_write_min_active: 10
vfs.zfs.vdev.sync_read_max_active: 10
vfs.zfs.vdev.sync_read_min_active: 10
vfs.zfs.vdev.max_active: 1000

非同期書き込み=10
同期書き込み=10
非同期読み出し=3
同期読み出し=10
scrub = 2

合計：35（注意：VDEV単位）

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- 2) TXGの同期I/O発行量
- デバイスのキュー長を調べて、余裕があるなら増やす

```
# camcontrol identify /dev/da0 | grep Q

pass0: 600.000MB/s transfers, Command Queueing Enabled
Tagged Command Queuing (TCQ)   no       no
Native Command Queuing (NCQ)   yes              32 tags
NCQ Queue Management           no
NCQ Streaming                  no
Receive & Send FPDMA Queued    no
```

- 増やした時にTXG同期が
  間に合ってるかどうか入念にチェックすること。
- デフォルトの値はSSDなど、IOPSが大きくとれるものに
  対しては少ないので、増やした方が良い。

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのI/O：書き込み

- 3) TXGのタイムアウト
  - 5秒のタイムアウトを短くする
    vfs.zfs.txg.timeout

- 短くするよりも、自分の負荷のデータ量で制限する方が良い

- 5秒より短いtimeoutは、書き込み負荷が小さい時の
  処理オーバヘッドが増加したり、
  フラグメンテーション増加の要因になる

- そもそも書き込み負荷が小さい用途なら、増やすのも一つの
  方法

https://people.allbsd.org/~hrs/sato-FBSDW20160624.pdf

ZFSのチューニングまとめ

- まだまだありますが、今回はこのへんで...
  （要望があれば次回以降に同様のネタを持ってくるかも）

- **読み込み：**
  キャッシュヒットがキーポイント。
  足りなければ増やすしかないが、特にメタデータの
  キャッシュヒット率が低いと全体的な性能が落ちる
- **書き込み：**
  TXGの同期量がキーポイント。
  自分の処理したい負荷、自分のマシンのメモリ、
  ストレージの能力に対して適正かどうかを測定して判断

告知

- FreeBSDワークショップ（ほぼ月一回）
  （次回は7月）

- AsiaBSDCon 2017
  2017/3/9-12
- 東京理科大学 森戸記念館
  飯田橋駅から徒歩5分、東京理科大学の施設