圧縮されたログファイルの活用ツール

お前は今までに解凍したログファイルの数をおぼえているのか？！

Klab勉強会#6

KLab株式会社　開発部8グループ
Engineering Manager

於保　俊

自己紹介

- 於保　俊（おほ　すぐる）です
- Twitter：ohomagic $\langle\psi|$
- 転職して1年半ちょっと経ちました
- 大学院で環境学→測量会社→KLab
- 専攻は自然科学（生態学・地学）と地理学・地理情報システム
- PHP、Java （その他も少し）
- 何かを作ることが大好きです
- KLabでは、ソーシャルゲームの開発・運用をしてきました

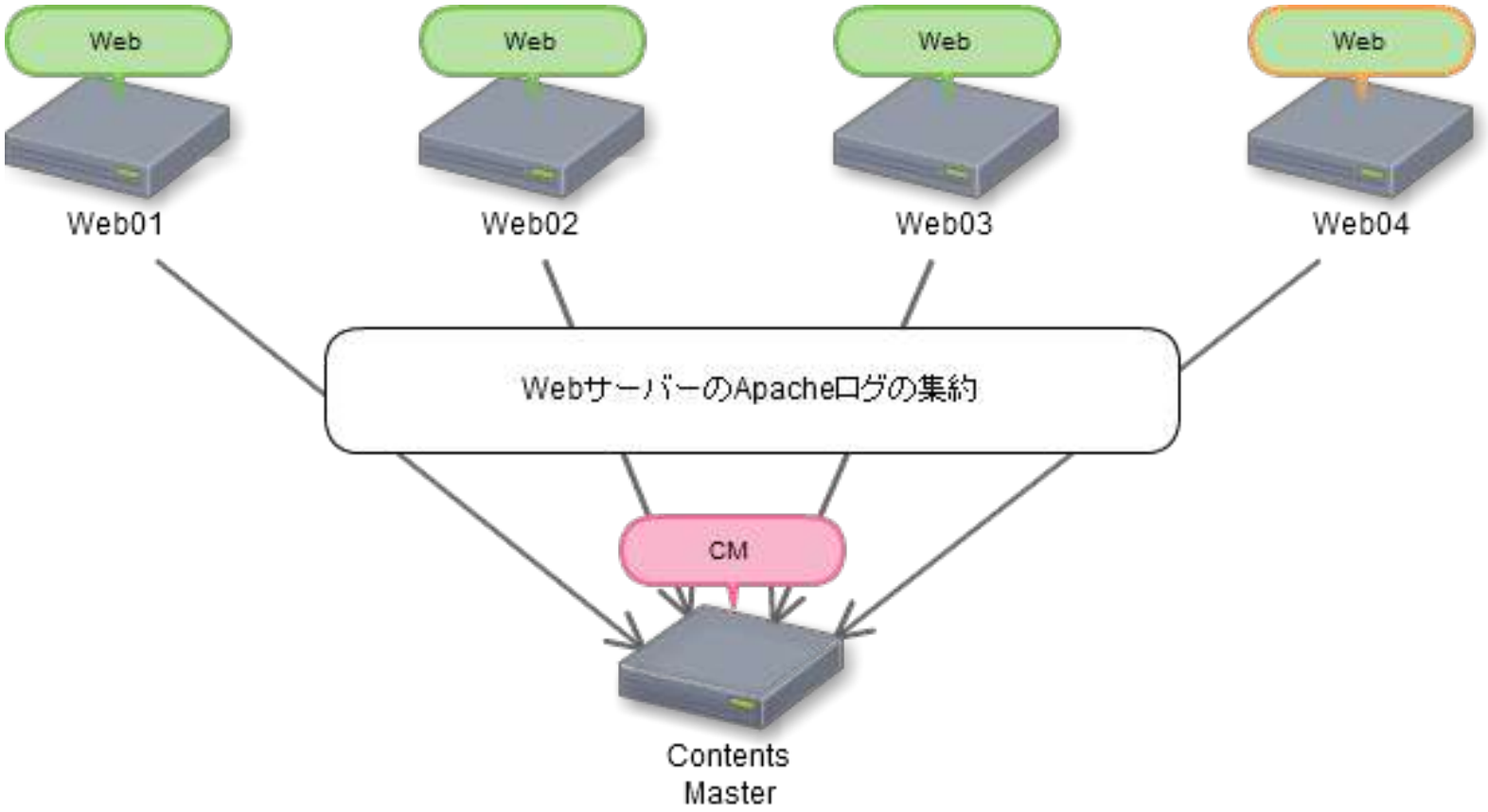
今日のお話

- KLabでは、Apacheのログをbzip2で圧縮して保存しています
- 解凍コストが高いです
- 例えば、特定の時間帯のログだとか
- 例えば、特定のユーザーの検索だとか
- 問い合わせとかで、すぐに見たい場合があるけど、一旦全解凍が必要（bzcatでも全解凍してる）
- ということで、なるべく解凍する量を少なく、色々できるツールを作ってみました

この後の話との関連

- ここでの話は、通常業務で発生するログの処理を、いかに高速かつ軽量に行えるかという話です
- 他アプリとの連携も含めて、大規模に蓄積・分析する話は、後ほどの高田の発表でどうぞ。

KLabのログ収集の仕組み



深夜のバッチ実行により、日付ごとのディレクトリに分割され、さらに1時間毎のファイルに分割されて保存される

ログファイルの実態

- 某案件
  - １時間分づつ分割して圧縮している
  - 平均ファイルサイズ　60MByte
  - 最大ファイルサイズ　100MByte超
  - 解凍すると　およそ1.5GByte

きっかけ！？

- bzip2の解凍は、時間かかるし、場合によってはワークスペース食うし大変
  →カジュアルにやらなきゃいけないけど、
  規模によってはカジュアルにしにくい
- 欲しいのってログの一部

→「途中から解凍できないかな？」

BZip2のファイル形式を見てみよう

- ヘッダ　BZhX
- ブロックのチャンク
- ブロックの開始さえわかれば、ブロックごとに独立で解凍できる

```
.magic:16                = 'BZ' signature/magic number
.version:8               = 'h' for Bzip2 ('H'uffman coding), '0' for Bzip1 (deprecated)
.hundred_k_blocksize:8   = '1'..'9' block-size 100 kB-900 kB
```

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
42 5A 68 39 31 41 59 26 53 59 CB 1D 25 32 06 C4   BZh91AY&SYﾋ.%2.ﾄ
87 DF 80 60 10 53 6F FF FB BF FF FF FA BF FF FF   ・.｀.So.鈔..慨..
F0 61 89 AE 00 00 07 DC 00 00 00 00 00 0F BE C0   ・屋...ﾜ......ｾﾀ
CE EE 50 03 59 01 A2 80 00 00 00 00 01 A1 54 00   ﾎﾈ連.Y.｢.....｡T.
06 80 00 00 7D 00 74 00 68 03 B7 3A 80 00 00 00   ....}.t.h.ｷ:....
00 00 00 7B E2 AA F7 00 00 00 00 00 03 1C 5E 6C   ...{筱........^l
C1 C0 A0 00 00 00 00 3E F1 E9 7D E7 D3 EB 2B 43   ﾁﾀ.....>・}醺.+C
63 B6 F5 9B 3E E4 33 6D EC 1F 54 3E 44 C0 00 00   cｶ・>.3m..T>Dﾀ..
D0 00 02 E6 EF 3E F3 40 00 00 02 68 28 00 14 02   ﾐ..跣>・...h(...
A5 46 7A 74 F7 DE 80 1B 3D 52 A6 FB 80 01 80 0F   ･Fzt・..=Rｦ祥...
00 23 A0 60 00 00 02 00 3A 32 DE 3C 69 E0 4C A8   .#.｀....:2ﾞ<i澱ｨ
07 2D 9B 58 59 67 B7 75 80 87 BD D1 1D 28 01 BE   .-婢Ygｷu.・ﾑ.(.ｾ
6F 00 21 EB AD 6D BE EF 79 BB C0 A3 DD DE AF 34   o.!・mｾ・ｻﾀ」ﾝﾞｯ4
```

```
*.selector_list:1..6     = zero-terminated bit runs (0..62) of MTF'ed Huffman table
(*selectors_used)
.start_huffman_length:5  = 0..20 starting bit length for Huffman deltas
*.delta_bit_length:1..40 = 0=>next symbol; 1=>alter length
                           { 1=>decrement length;  0=>increment length }
(*(symbols+2)*groups)
.contents:2..∞           = Huffman encoded data stream until end of block
.eos_magic:48            = 0x177245385090 (BCD sqrt(pi))
.crc:32                  = checksum for whole stream
.padding:0..7            = align to whole byte
```

http://en.wikipedia.org/wiki/Bzip2

ヘッダの探索

- 3141592653を検索すればいいんじゃない？
- ハフマン符号化されてるから無理(バイトの開始位置が不定)
- それなら・・・
- ビットシフトされた8パターンのビットパターンを検索する

```
{0x31, 0x41, 0x59, 0x26, 0x53},
{0x62, 0x82, 0xB2, 0x4C, 0xA6},
{0xC5, 0x05, 0x64, 0x99, 0x4D},
{0x8A, 0x0A, 0xC9, 0x32, 0x9A},
{0x14, 0x15, 0x92, 0x65, 0x35},
{0x28, 0x2B, 0x24, 0xCA, 0x6B},
{0x50, 0x56, 0x49, 0x94, 0xD6},
{0xA0, 0xAC, 0x93, 0x29, 0xAC}
```

```
     3   1   4   1   5   9   2   6   5   3
0000 00110001010000010101100100100110010100 11
     6   2   8   2   B   2   4   C   A   6
0000 01100010100000101011001001001100101001 10
     C   5   0   5   6   4   9   9   4   D
0000 11000101000001010110010010011001010011 01
     8   A   0   A   C   9   3   2   9   A
0001 10001010000010101100100100110010100110 10
     1   4   1   5   9   2   6   5   3   5
0011 00010100000101011001001001100101001101 01
     2   8   2   B   2   4   C   A   6   B
0110 00101000001010110010010011001010011010 11
     5   0   5   6   4   9   9   4   D   6
1100 01010000010101100100100110010100110101 10
     A   0   A   C   9   3   2   9   A   C
1000 10100000101011001001001100101001101011 00
```

解凍ライブラリ

- Apache Commons Compress を使うことに

解凍ストリームの改造

- Org.apache.commons.compress.compressors.bzip2.BZip2CompressorInputStream を使用
- 前のブロックから引き継ぐ情報をコンストラクタで与えられるようにする
- ブロックの終わりで止まるようにする

```
public BZip2CompressorInputStream(final InputStream in) throws
IOException {
```

↓

```
public BZip2PartedCompressorInputStream(final InputStream in,int
bsBuff,int bsLive,int blockSize100k, boolean serialMode) throws
IOException {

    endBlock();
    initBlock();
    SetupBlock();
```

↓

```
    endBlock();
    if(serialMode){
     initBlock();
     setupBlock();
    } else {
      complete();
    }
```

前のブロックから引継ぐ情報？

- ハフマン符号で、バッファに入っているビット列と何ビット目を読んでるかの0~7までの数字

```
        3   1   4   1   5   9   2   6   5   3
0000 00110001010000010101100100100110010100 11
        6   2   8   2   B   2   4   C   A   6
0000 01100010100000101011001001001100101001 10
        C   5   0   5   6   4   9   9   4   D
0000 11000101000001010110010010011001010011 01
        8   A   0   A   C   9   3   2   9   A
0001 10001010000010101100100100110010100110 10
        1   4   1   5   9   2   6   5   3   5
0011 00010100000101011001001001100101001101 01
        2   8   2   B   2   4   C   A   6   B
0110 00101000001010110010010011001010011010 11
        5   0   5   6   4   9   9   4   D   6
1100 01010000010101100100100110010100110101 10
        A   0   A   C   9   3   2   9   A   C
1000 10100000101011001001001100101001101011 00
```

豆知識

- Bzip2recoverコマンドの紹介
- Bzip2ファイルが壊れていた時に、正常な部分だけむりやり解凍できるコマンドです
- ブロック単位で解凍されて、シリアルナンバーのついたファイル群が生成されます
- デバッグで使いました

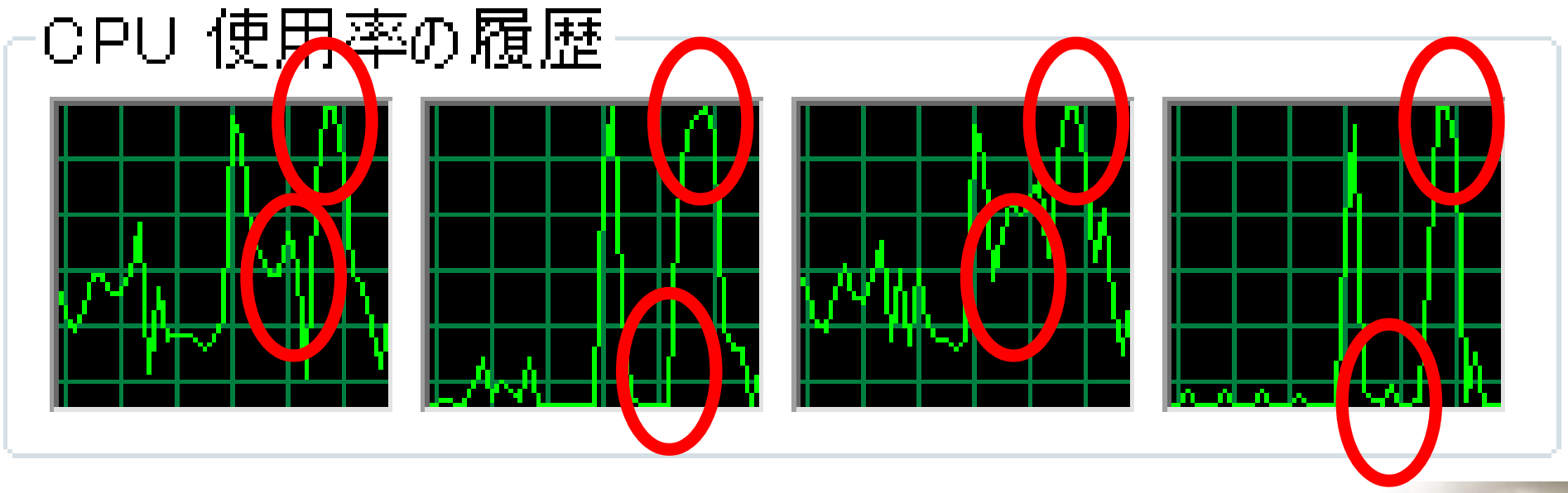
とりあえず並列化してみた

- ファイルから単一スレッドで読み込む
- ブロック探索して、ブロックを見つけたら切り出し、解凍キューに登録
- 解凍スレッドが並列に解凍処理
- 解凍したデータを書き出しキューにため
- 結合して書き出し

- ここまで6時間くらい　クイックハック！！
- ついでに社内で表彰されました。ベストテクノロジー賞獲得！！

結果

- 全コア使いきって頑張ってる
- 速くなった　8192ms→3576ms



pbzip2の紹介

- 並列bzip2
- http://compression.ca/pbzip2/

- 他にも並列実装はあるようです

さてここから本題

ログの調査で多いこと

- ある時間のログの抽出
- あるユーザーのログの抽出
- あるキーワードでのログの抽出

ある時間帯のログの抽出

- 「二分検索でブロック探索」

- ブロックを見つけたら先頭の２行だけ解凍
- 解凍したログの１行分の時間を見て二分探索
- 分割サイズがしきい値以下なら、そこから終了時間まで解凍＆出力
- 命名「BzBinSearch」

デモ

結果

$ time bzcat tsubasa.app.2011-12-01_22.bz2 | grep 22:30:00 > testlog2.log

real **1m2.744s**

user 0m56.996s

sys 0m2.124s

$ time bzbinsearch 2011-12-01_22:30:00 2011-12-01_22:30:01 tsubasa.app.2011-12-01_22.bz2 > testlog1.log

real **0m4.220s**

user 0m3.156s

sys 0m0.924s

あるユーザーのログの抽出

- あらかじめ、インデックスを作っておく
- ブロックのマップ

  ブロックのNo.　何バイト目から始まるか　解凍に必要な情報
- あるユーザーIDがどのブロックに含まれるか

  ユーザーID　ブロックNo.の羅列
- 検索時はその情報を見て、必要最小限のブロックだけ解凍
- 運用としては、ログ集約の際に同じバッチでインデックスを作成する

デモ

結果

$time bzcat tsubasa.app.2012-04-01_23.bz2 | grep opensocial_viewer_id=38527024

real **1m13.775s**

user 1m13.621s

sys 0m1.904s

$time bzusersearch tsubasa.app.2012-04-01_23.bz2 38527024

real **0m6.166s**

user 0m13.829s

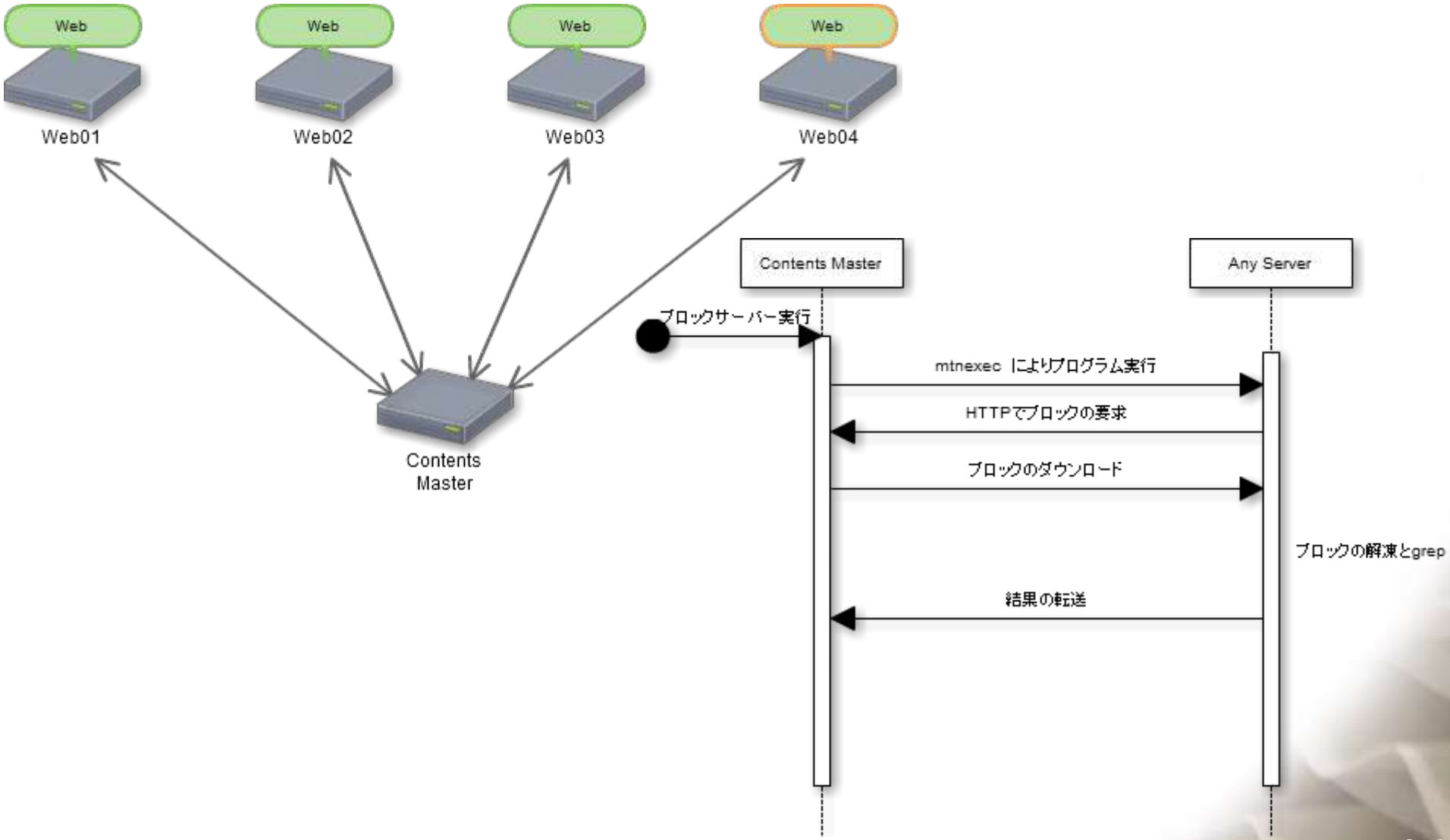
sys 0m0.796s

あるキーワードの検索

- 細かいブロック単位で解凍できることをいいことになんちゃって分散処理してみた
- 手元のBZip2ファイル（ブロックインデックス作成済み）をブロック単位で配信するWebサーバーを作成

  Nettyを使った　サーバーがブロックNo.を保持し、リクエストごとに違うブロックを送出
- DSASのリモートシェル機能（mtnexec）を利用
- 手元でブロックサーバーを起ち上げる→mtnexecでcurlでサーバーからブロック取得→パイプでbunzip→パイプでgrep→標準出力に書き出されたものをmtnexecで集約
- 命名「mtnDP（MoTtaiNai Distributed Processing）」

図に表すと


Web
Web
Web
Web
Web01
Web02
Web03
Web04
Contents
Master
Contents Master
Any Server
ブロックサーバー実行
mtnexec によりプログラム実行
HTTPでブロックの要求
ブロックのダウンロード
ブロックの解凍とgrep
結果の転送

デモ

結果

$ bzcat tsubasa.app.2011-12-01_12.bz2 | grep CardStack

real **1m13.547s**

user 1m13.121s

sys 0m1.640s

$ time mtnexec -Ri -P 10 'for a in $(seq 1 200); do curl http://w112:56818/ | bunzip2 | grep CardStack; done' ::: $(seq 1 10) 1> testmtnexec.log 2> testmtnerror.log

real **0m25.474s**

user 0m0.160s

sys 0m1.284s

プログラムはこちら（予定）

- KLabのgithubアカウントで公開予定
- https://github.com/KLab

まとめ

- こんなことできました！！
- これでbzip2がもっと使いやすくなるはず

ご清聴ありがとうございました

質疑応答