2.2.4. LMbench によるメモリレイテンシ測定

富士通株式会社　石附 茂

1. 概要

LMbench はマシンの基本性能を測定するツールである。測定項目は以下の 2 項目に大別される。

1) バンド幅
　メモリ，ファイル入出力関連
2) レイテンシ
　キャッシュ，メモリ，コンテキストスイッチ，ファイル操作，プロセス，シグナルなど

2. メモリレイテンシの実測

LMbench を使用し、メモリレイテンシを実測した結果を報告する。対象マシンは、富士通 RX200S3 である。レイテンシ測定機構および測定時の注意点などについて述べた後、結果を考察する。

2.1. レイテンシ測定機構

レイテンシの算出方法は、以下に示すアドレス更新処理を多数回実行することで行っている。

```
p = (char **)*p
```

レイテンシ時間＝測定時間 / 実行回数 により算出される。
なお、測定結果は、ナノ秒単位で標準出力に出力される。

2.2. レイテンシ測定時の注意事項

(1)ストライド幅の設定
指定されたストライド幅で配列にアクセスしており、指定値により測定結果は変化する。
・指定値が小さい → キャッシュラインの再利用効果で過小評価となる.
・指定値が大きい → TLB ミスの影響を計測するため過大評価となる.
よって、測定対象機のキャッシュラインサイズやキャッシュサイズを考慮し、適切な値を指定しなければならない。小サイズから大サイズまで複数パターン測定し、分析する必要がある。
「2.3.測定結果について」の節で、具体例を示し解説する。

(2)アクセス範囲の設定
指定されたサイズの範囲内でストライドアクセスする。指定するサイズが小さいと、キャッシュ部だけを評価することになる。つまり、キャッシュサイズ合計より大きい値を指定する必要がある。

(3)測定結果の分析
測定結果は、実際に計測した値が列挙されており、全ての値を比較しなければならない。
一般的には、L1$ 部，L2$ 部，メモリ部などの傾向が見えるはずである。しかし、全ての値を見るのは煩わしいので、グラフ化することを推奨する。L1$ 部，L2$ 部，メモリ部は平坦な領域となり視覚的に把握することができる。

2.3. 測定結果について

富士通 RX200S3（clovertown(2.66GHz)）における測定結果を 図-1 に示す。
グラフの縦軸はレイテンシ時間(n 秒)であり、横軸はアクセス範囲(MB)である。複数の曲線は、各 stride 幅(32, 64, 128, 256, 512, 1024, 2048, 4096)に対応している。ただし、128byte 以上は、値が同じであり曲線は重なっている。この結果から、以下の 2 点が読み取れる。

(1) stride 幅の影響
Stride 幅 32byte と 128byte では計測値が異なっている。32byte では stride 幅が小さいため、キャッシュライン再利用効果の影響により計測値が小さくなったと推測される。Stride 幅が 128byte

以上 4096byte までは、ほぼ同じ値となっている。ただし、これは clovertown の結果であり、他機種でも同様とは限らない。

(2) アクセスレイテンシ

グラフの形状から、3 ヶ所の平坦域が確認できる。これらは、一次キャッシュ，二次キャッシュおよびメインメモリに対応している。Stride 幅 512byte での計測結果は、

L1$：1.14 n 秒（≒3 cycle）
L2$：5.33 n 秒（≒14 cycle）
Mem：125　n 秒（≒332 cycle）

となっている。

3. 測定環境の構築方法

公開ホームページからベンチマークセットをダウンロードし、測定対象マシンにインストールする。

・http://lmbench.sourceforge.net から、「lmbench-3.0-a8.tgz」をダウンロードする。
・「lmbench-3.0-a8.tgz」を測定対象マシンに展開する。（C 言語コンパイラが必要）

4. 実行方法

下記のように make するだけで、コンパイルから測定まで自動的に実行される。

```
% cd lmbench-3.0-a8
% make results
```

ただし、途中でパラメータの入力が必要になる。（12 項目）

パラメータ入力例

```
MULTIPLE COPIES : [リターン]
Job placement selection : [リターン]
MB : [256] ← 確保するメモリ量を指定する（file I/O とメモリの両方で使用される）
SUBSET (ALL|HARWARE|OS|DEVELOPMENT) : [リターン]
FASTMEM : [リターン]
SLOWFS : [yes]
DISKS : [リターン]
REMOTE : [リターン]
Processor mhz : [リターン]
FSDIR : [/work] ← 測定対象ファイルシステム上のディレクトリを指定する
Status output file : [リターン]
Mail results : [no]
```

上記パラメータの入力が終わると、自動的に計測が開始され、標準出力に進行状況が表示される。

5. 実行結果の確認方法

1) テキスト形式でサマリーを出力する方法

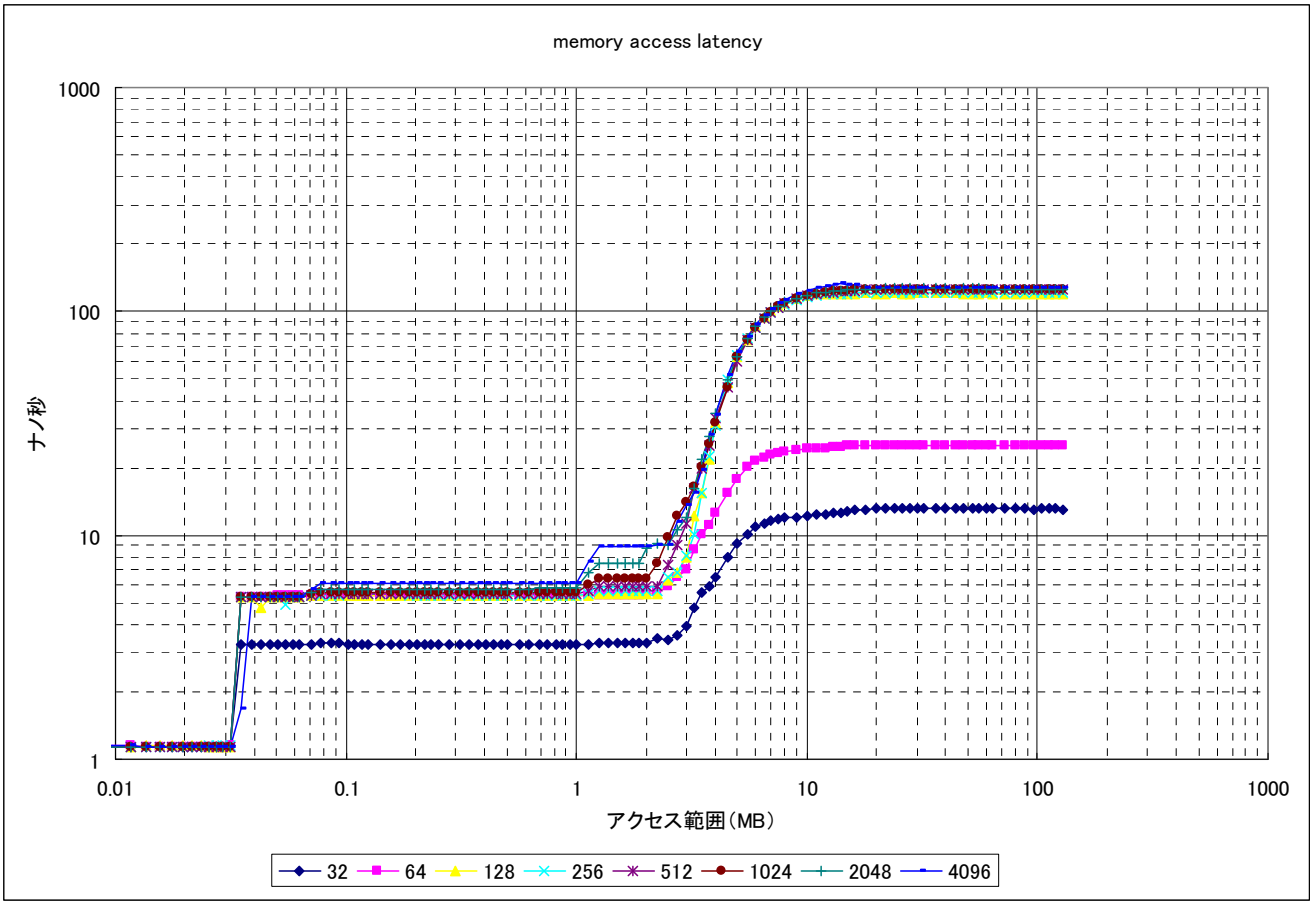
ディレクトリ results に移動後、
% make summary
と入力すると、標準出力に測定結果のサマリーが表示される。
サマリーの出力例を文末の［参考資料］に示す。

2) 測定結果をグラフにする方法

ディレクトリ results に移動後、
% make ps
と入力すると、測定結果をグラフにしたポストスクリプトファイルが作成される。
ポストスクリプトファイルの出力例を文末の［参考資料］に示す。

☆注意事項

ノード内に複数 CPU 存在する場合は、プロセスと CPU をバインドする必要がある。
バインドが困難な場合は、入力の「MULTIPLE COPIES」にノード内 CPU 数を指定することで、プロセスの CPU 間移動を抑える効果が期待できる。ただし、実行時間が極端に長くなる。



機種：	RX200S3, Xeon X5355(clovertown 2.66GHz), FSB 1333MHz
L1$ ：	32 KB
L2$ ：	8 MB

図-1. メモリレンテンシ測定結果

以上

[参考資料]

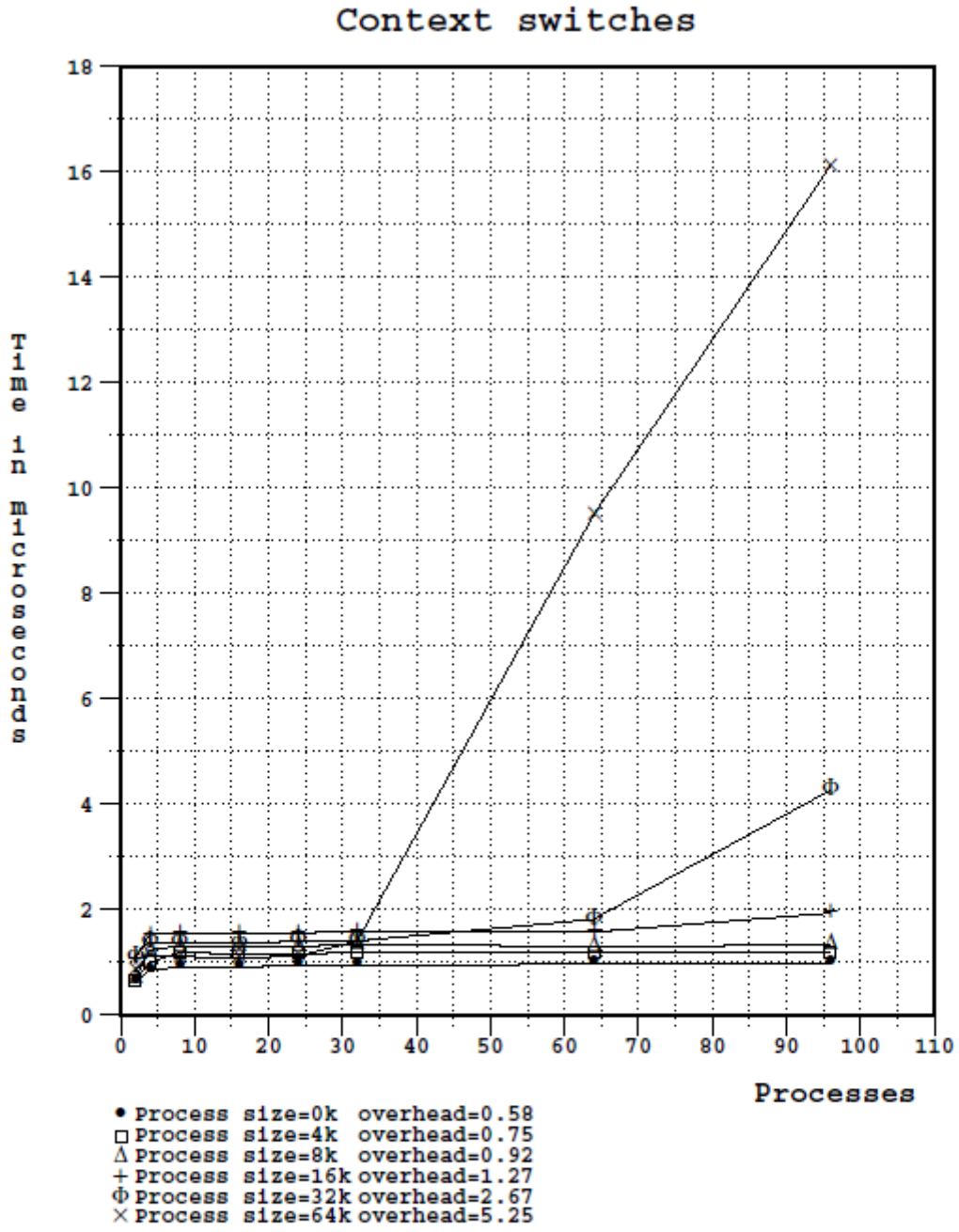
■サマリー出力例

```
                 L M B E N C H  3 . 0   S U M M A R Y
                 ------------------------------------
                 (Alpha software, do not distribute)

Basic system parameters
----------------------------------------------------------------------------
Host                 OS Description              Mhz  tlb  cache  mem   scal
                                                     pages line   par   load
                                                           bytes
--------- ------------- ----------------------- ---- ----- ----- ------ ----
Machine           Linux                linux-gnu 2984     8   128 6.1200    1

Processor, Processes - times in microseconds - smaller is better
----------------------------------------------------------------------------
Host                 OS  Mhz null null      open slct sig  sig  fork exec sh
                         call  I/O stat clos TCP  inst hndl proc proc proc
--------- ------------- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
Machine           Linux 2984 0.09 0.14 2.58 2.88 8.16 0.18 1.28 82.0 261. 1051

Basic integer operations - times in nanoseconds - smaller is better
-------------------------------------------------------------------
Host                 OS  intgr intgr  intgr  intgr  intgr
                          bit   add    mul    div    mod
--------- ------------- ------ ------ ------ ------ ------
Machine           Linux 0.3400 0.1700 1.0200   11.3 6.8700

Basic uint64 operations - times in nanoseconds - smaller is better
-------------------------------------------------------------------
Host                 OS int64  int64  int64  int64  int64
                         bit    add    mul    div    mod
--------- ------------- ------ ------ ------ ------ ------
Machine           Linux        0.1700

Basic float operations - times in nanoseconds - smaller is better
-----------------------------------------------------------------
Host                 OS  float  float  float  float
                         add    mul    div    bogo
--------- ------------- ------ ------ ------ ------
Machine           Linux 1.0200 1.3500 7.4800 5.8100

Basic double operations - times in nanoseconds - smaller is better
-------------------------------------------------------------------
Host                 OS  double double double double
                         add    mul    div    bogo
--------- ------------- ------ ------ ------ ------
Machine           Linux 1.0200 1.6900   12.2   14.6

Context switching - times in microseconds - smaller is better
-------------------------------------------------------------------------
Host                 OS  2p/0K 2p/16K 2p/64K 8p/16K 8p/64K 16p/16K 16p/64K
                         ctxsw  ctxsw  ctxsw ctxsw  ctxsw   ctxsw   ctxsw
--------- ------------- ------ ------ ------ ------ ------ ------- -------
Machine           Linux 0.6100 1.0600 0.8300 1.5600 1.1000 1.54000 1.08000

*Local* Communication latencies in microseconds - smaller is better
---------------------------------------------------------------------
Host                 OS 2p/0K  Pipe AF     UDP  RPC/   TCP  RPC/ TCP
                        ctxsw       UNIX         UDP         TCP conn
--------- ------------- ----- ----- ---- ----- ----- ----- ----- ----
Machine           Linux 0.610 2.291 4.63 5.211 9.051 6.330  12.3 11.6

File & VM system latencies in microseconds - smaller is better
-------------------------------------------------------------------------------
Host                 OS   0K File      10K File     Mmap    Prot   Page   100fd
                        Create Delete Create Delete Latency Fault  Fault  selct
--------- ------------- ------ ------ ------ ------ ------- ----- ------- -----
Machine           Linux                                1509.0 0.446 0.99480 7.004

*Local* Communication bandwidths in MB/s - bigger is better
-----------------------------------------------------------------------------
Host                OS  Pipe AF    TCP  File   Mmap  Bcopy  Bcopy  Mem   Mem
                        UNIX      reread reread (libc) (hand) read write
--------- ------------- ---- ---- ---- ------ ------ ------ ------ ---- -----
Machine           Linux 1676 3383 1714 2378.4 3490.8 1434.5 1404.4 3215 1181.

Memory latencies in nanoseconds - smaller is better
    (WARNING - may not be correct, check graphs)
----------------------------------------------------------------------------
Host                 OS   Mhz   L1 $   L2 $    Main mem    Rand mem    Guesses
--------- -------------   ---   ----   ----    --------    --------    -------
Machine           Linux  2984 1.0150 4.7580  111.2       125.3
```

■サマリー出力項目一覧

項目	サマリ内記述
CPU クロック (MHz)	Mhz
TLB エントリ数	tlb pages
キャッシュラインサイズ	cache line bytes
実行多重度	scal load
OS 呼び出しのレイテンシ	null call
read,write 関数の平均レイテンシ	null I/O
stat 関数のレイテンシ	stat
open,close の合計レイテンシ	open clos
ソケット read,write の合計レイテンシ	slct TCP
シグナル設定(sigemptyset)のレイテンシ	sig inst
シグナル送信(kill)のレイテンシ	sig hndl
fork のレイテンシ	fork proc
execv のレイテンシ	exec proc
sh 起動のレイテンシ	sh proc
整数の XOR 演算レイテンシ	intgr bit
整数の加算レイテンシ	intgr add
整数の乗算レイテンシ	intgr mul
整数の除算レイテンシ	intgr div
整数の剰余レイテンシ	intgr mod
64 ビット整数の XOR 演算レイテンシ	int64 bit
64 ビット整数の加算レイテンシ	int64 add
64 ビット整数の乗算レイテンシ	int64 mul
64 ビット整数の除算レイテンシ	int64 div
64 ビット整数の剰余レイテンシ	int64 mod
単精度実数の加算レイテンシ	float add
単精度実数の乗算レイテンシ	float mul
単精度実数の除算レイテンシ	float div
単精度実数四則演算{ a=(b+c)*(d-e)/f }のレイテンシ	fload bogo
倍精度実数の加算レイテンシ	double add
倍精度実数の乗算レイテンシ	double mul
倍精度実数の除算レイテンシ	double div
倍精度実数四則演算{ a=(b+c)*(d-e)/f }のレイテンシ	double bogo
メモリ 0KB のプロセス 2 個のコンテキスト切り替え時間	2p/0K ctxsw
メモリ 16KB のプロセス 2 個のコンテキスト切り替え時間	2p/16K ctxsw
メモリ 64KB のプロセス 2 個のコンテキスト切り替え時間	2p/64K ctxsw
メモリ 16KB のプロセス 8 個のコンテキスト切り替え時間	8p/16K ctxsw
メモリ 64KB のプロセス 8 個のコンテキスト切り替え時間	8p/64K ctxsw
メモリ 16KB のプロセス 16 個のコンテキスト切り替え時間	16p/16K ctxsw
メモリ 64KB のプロセス 16 個のコンテキスト切り替え時間	16p/64K ctxsw
Pipe によるプロセス間待ち時間	Pipe
ソケットによるプロセス間通信待ち時間	AF UNIX
UDP/IP 経由のプロセス間通信待ち時間	UDP
UDP 経由の RPC の待ち時間	RPC/UDP
TCP/IP 経由のプロセス間通信待ち時間	TCP
TCP 経由の RPC の待ち時間	RPC/TCP
ソケットのコネクション時間	TCP conn
0KB のファイル生成時間	0K File Create
0KB のファイル削除時間	0K File Delete
10KB のファイル生成時間	10K File Create
10KB のファイル削除時間	10K File Delete
Mmap アクセスレイテンシ	Mmap Latency
Protection fault のレイテンシ	Prot Fault
Page fault のレイテンシ	Page Fault
100 個のデスクリプタに対する select システムコールの時間	100fd selct
ファイル読み込みを繰り返した時の平均バンド幅	File reread
Mmap アクセスを繰り返した時の平均バンド幅	Mmap reread
bcopy 関数のバンド幅	Bcopy (libc)
配列コピーのバンド幅	Bcopy (hand)
メモリ読み込みのバンド幅	Mem read
メモリ書き出しのバンド幅	Mem write
1 次キャッシュのアクセスレイテンシ	L1 $
2 次キャッシュのアクセスレイテンシ	L2 $
メモリのアクセスレイテンシ	Main mem
ストライド幅を変化させた場合のメモリアクセスレイテンシ	Rand mem

■context switch の出力例



コンテキストの切り替え時間

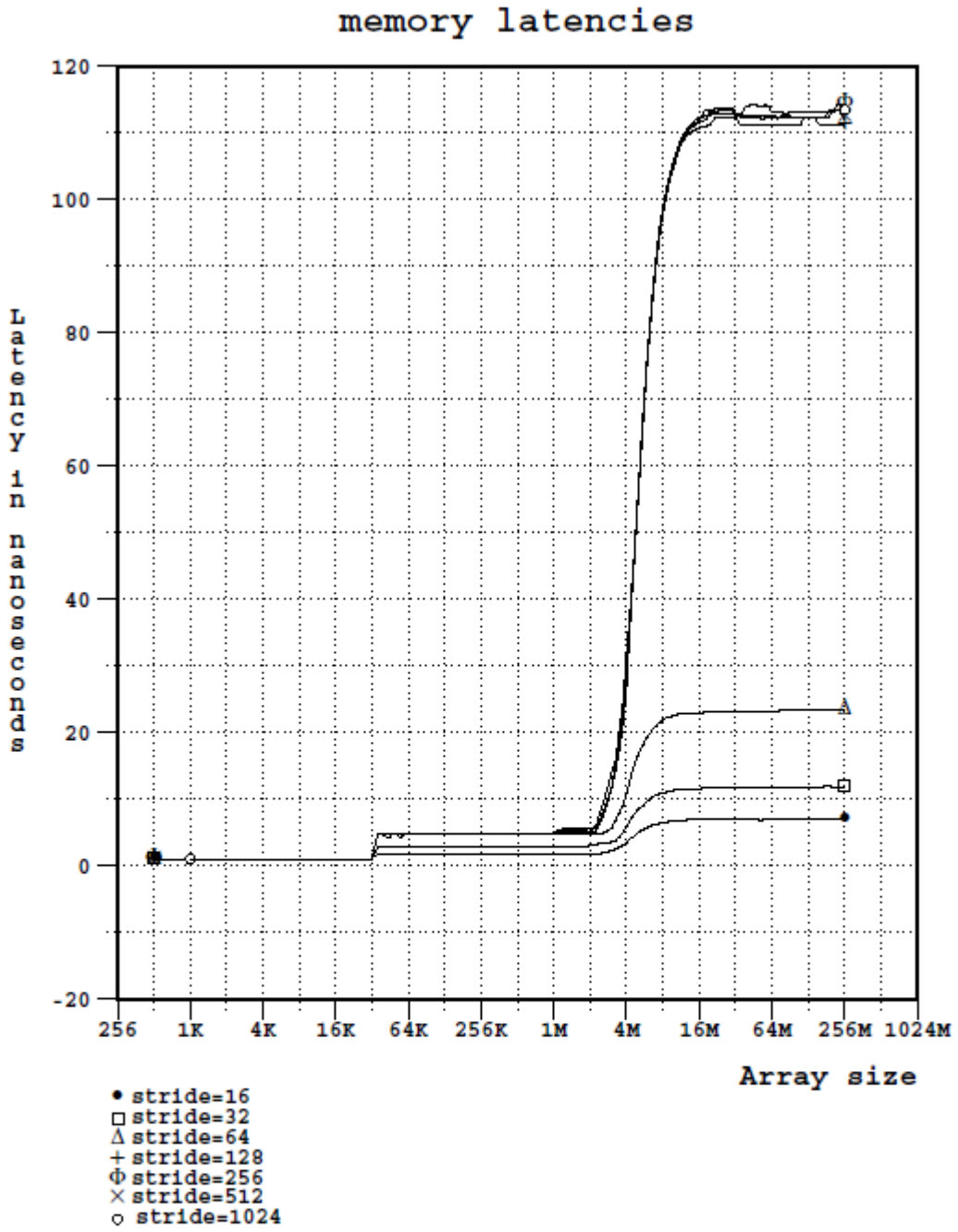
size はアロケートするメモリ量

overhead はメモリアロケート・配列定義参照などの処理時間

X 軸は生成(fork)するプロセス数

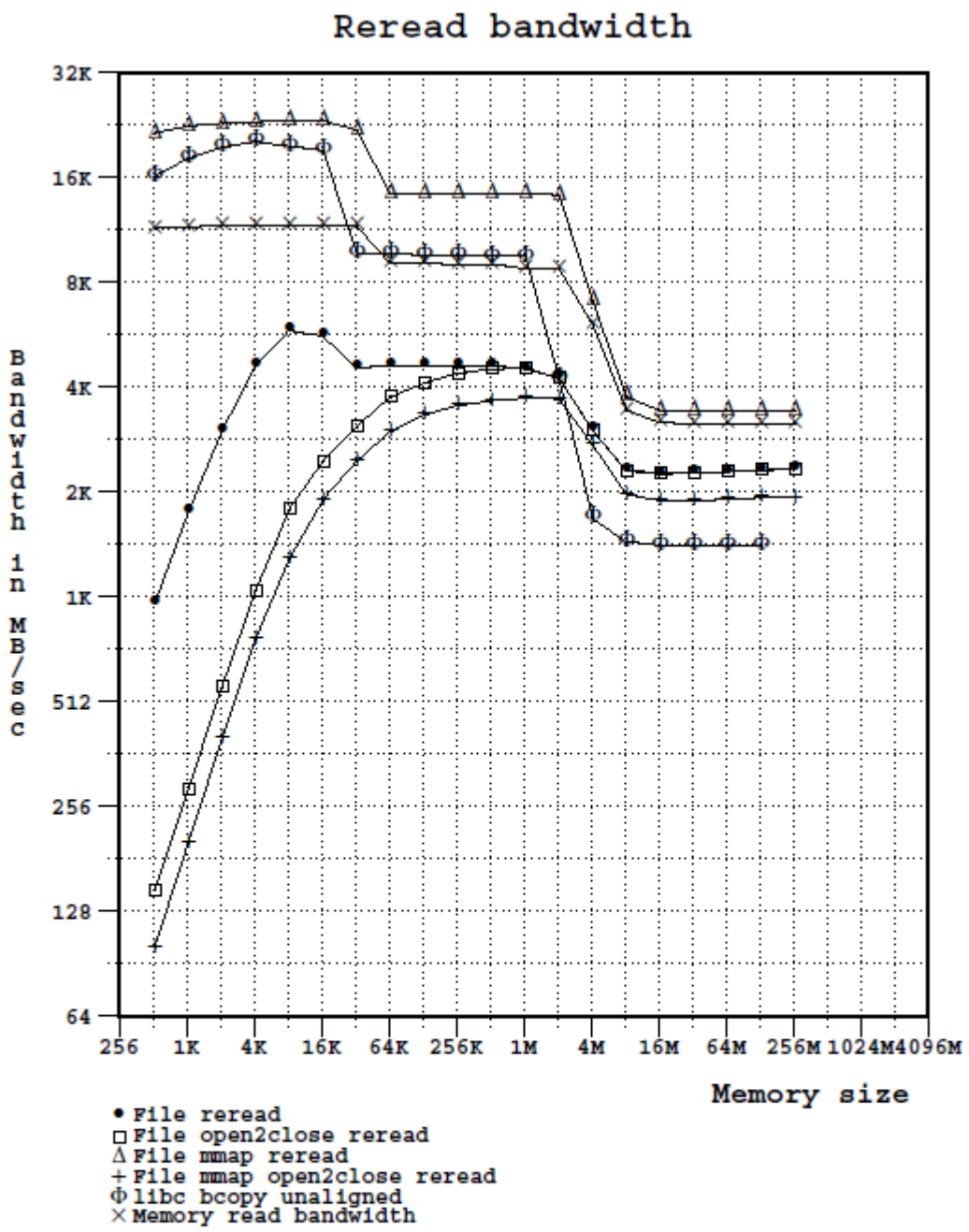
測定時間から overhead 時間を減算した値が評価値

■memory latency の出力例



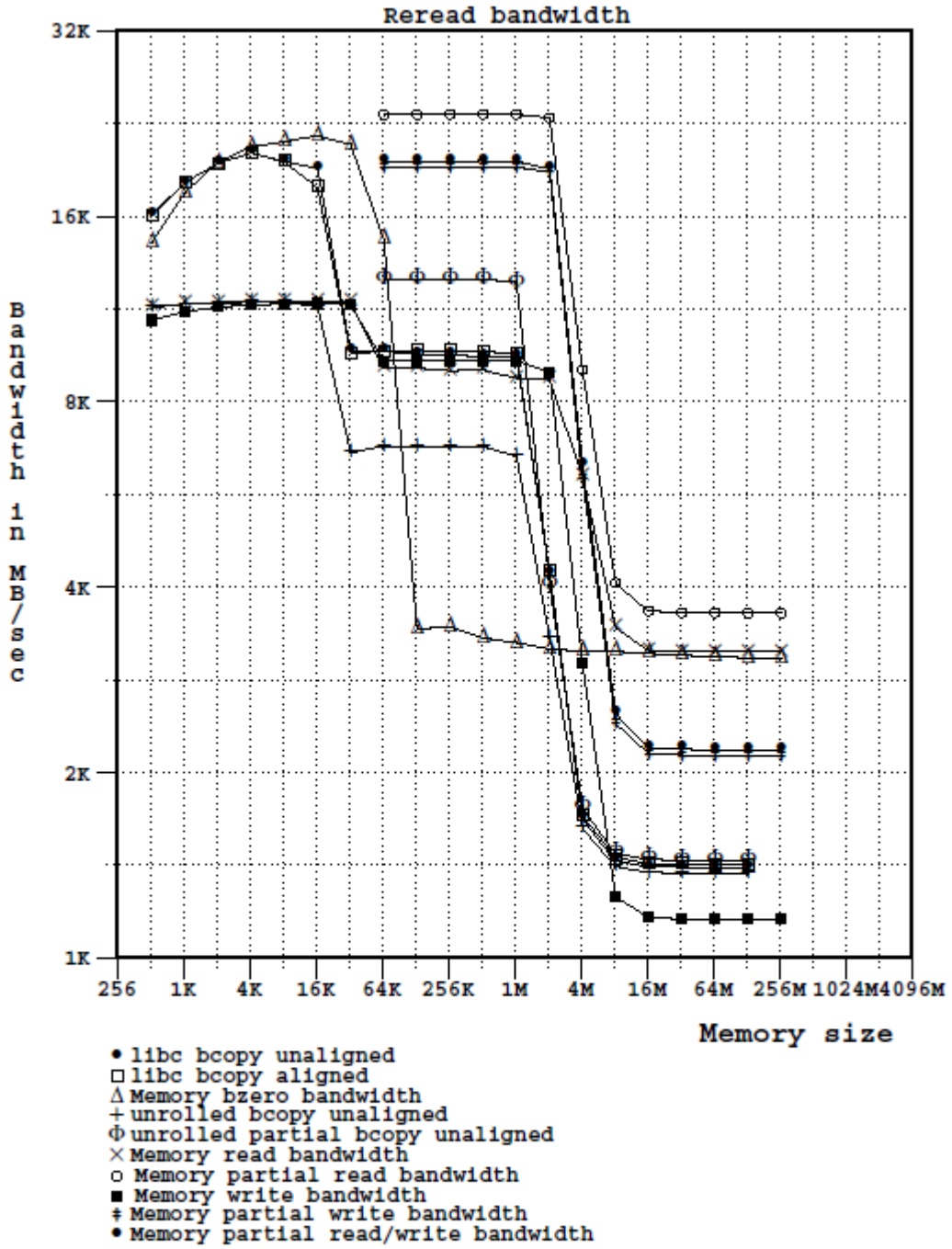
メモリアクセスのレイテンシ時間
stride はアクセス時の幅

■file read bandwidth の出力例



File reread : ファイルポインタを先頭に戻して read を繰り返す時間からバンド幅を計算
File open2close reread : open -> read -> close を繰り返す時間からバンド幅を計算
File mmap reread : mmap でマップ作成後、アクセスを繰り返す時間からバンド幅を計算
File mmap open2close reread : open -> mmap -> [access] -> close -> munmap を繰り返す時間
libc bcopy unaligned : bcopy を繰り返す時間からバンド幅を計算（8byte 境界ではない）
Memory read bandwidth : メモリ参照のバンド幅

■memory read bandwidth の出力例



libc bcopy unaligned : bcopy を繰り返す時間からバンド幅を計算（8byte 境界ではない）
libc bcopy aligned : bcopy を繰り返す時間からバンド幅を計算（8byte 境界）
Memory bzero bandwidth : bzero を繰り返す時間からバンド幅を計算
unrolled bcopy unaligned : 配列コピーを繰り返す時間からバンド幅を計算
unrolled partial bcopy unaligned : 32 バイトのストライドで配列コピーを繰り返す時間からバンド幅を計算
Memory read bandwidth : 連続メモリ参照のバンド幅
Memory partial read bandwidth : 32 バイトのストライドでメモリ参照のバンド幅
Memory write bandwidth : 連続メモリ定義のバンド幅
Memory partial write bandwidth : 32 バイトのストライドでメモリ定義のバンド幅
Memory partial read/write bandwidth : 32 バイトのストライドでメモリ参照&定義（同領域への総和）のバンド幅