

Estruturas de Dados

Biblioteca memlog
Aplicativo analisamem

Professores: Marcio Santos
Wagner Meira Jr

Roteiro: Caracterização de Padrão de Acesso à Memória e Localidade de Referência

1. Avalie qualitativamente o programa a ser caracterizado em termos dos acessos de memória esperados e localidade de referência. Identifique as estruturas de dados e segmentos de código críticos (p.ex., mais custosos)
2. Elabore o plano de caracterização de localidade de referência
3. Selecione os parâmetros do programa a ser caracterizado
 - a. Programa não deve executar por muito ou pouco tempo, mas o suficiente para entender o comportamento do algoritmo.
4. Execute o código com Cachegrind:
 - a. `valgrind --tool=cachegrind ./matop -m -x 5 -y 5`
 - b. `cg_annotate cachegrind.out.XXXX`
5. Execute o código com Callgrind
 - a. `valgrind --tool=callgrind ./matop -m -x 5 -y 5`
 - b. `callgrind_annotate callgrind.out.XXXX`
6. Avalie a saída do CacheGrind identificando:
 - a. Quão bem o programa se comporta em termos de memória
 - b. Estruturas de dados a serem caracterizadas
 - c. Segmentos de código a serem instrumentados

Roteiro: Caracterização de Padrão de Acesso à Memória e Localidade de Referência

7. Instrumente o código
 - a. Use a biblioteca memlog
8. Execute os experimentos do código instrumentado
9. Analise, utilizando os padrões de acesso e a localidade de referência
 - a. Use o aplicativo analisamem
 - b. Visualizações
 - i. Mapa de Acesso
 - ii. Histograma de distância de pilha
 - iii. Evolução da distância de pilha
10. Sugira, se for o caso, modificações no código ou na estrutura de dados

Monitoramento usando memlog

1. Selecionar as estruturas de dados a serem monitorizadas
 - a. Nem todos os dados e estruturas de dados tem que ser monitorizados
 - b. Para cada estrutura de dados, definir o grão do monitoramento
 - i. Vetores: atenção com o tamanho do elemento
 - ii. Registros: pode ser interessante analisar partes do registro
2. Selecionar as funções a serem instrumentadas
3. Definir as fases de monitoramento
4. Instrumentar o código
5. Definir o plano de experimentos
6. Executar os experimentos
7. Gerar as visualizações
8. Analisar os resultados e visualizações

Biblioteca memlog: memlog.h

```
typedef struct memlog{
    FILE * log;
    clockid_t clk_id;
    struct timespec inittime;
    long count;
    int fase;
    int ativo;
} memlog_tipo;
extern memlog_tipo ml;

// constantes definindo os estados de registro
#define MLATIVO 1
#define MLINATIVO 0

#define LEMEMLOG(pos,tam,id) \
    ((void) ((ml.ativo==MLATIVO)?leMemLog(pos,tam,id):0))

#define ESCREVEMEMLOG(pos,tam,id) \
    ((void) ((ml.ativo==MLATIVO)?escreveMemLog(pos,tam,id):0))
```

Biblioteca memlog: memlog.h

```
int iniciaMemLog(char * nome);  
int ativaMemLog();  
int desativaMemLog();  
int defineFaseMemLog(int f);  
int leMemLog(long int pos, long int tam, int id);  
int escreveMemLog(long int pos, long int tam, int id);  
int finalizaMemLog();
```

Biblioteca memlog: memlog.c

```
void clkDifMemLog(struct timespec t1, struct timespec t2,  
                  struct timespec * res)  
  
// Descricao: calcula a diferenca entre t2 e t1, que e  
armazenada em res  
  
// Entrada: t1, t2  
  
// Saida: res  
{  
    if (t2.tv_nsec < t1.tv_nsec){  
        // ajuste necessario, utilizando um segundo de tv_sec  
        res-> tv_nsec = 1000000000+t2.tv_nsec-t1.tv_nsec;  
        res-> tv_sec = t2.tv_sec-t1.tv_sec-1;  
    } else {  
        // nao e necessario ajuste  
        res-> tv_nsec = t2.tv_nsec-t1.tv_nsec;  
        res-> tv_sec = t2.tv_sec-t1.tv_sec;  
    }  
}
```

Biblioteca memlog: memlog.c

```
int iniciaMemLog(char * nome)
// Descricao: inicializa o registro de acessos, abrindo o arquivo nome
// Entrada: nome
// Saida: nao tem
{ // escolhe modo do relógio
  ml.clk_id = CLOCK_MONOTONIC;
  // abre arquivo de registro e verifica se foi aberto corretamente
  ml.log = fopen(nome, "wt");
  erroAssert(ml.log != NULL, "Cannot open memlog output");
  // captura o tempo inicial do registro
  struct timespec tp;
  int result = clock_gettime(ml.clk_id, &tp);
  ml.inittime.tv_sec = tp.tv_sec;  ml.inittime.tv_nsec = tp.tv_nsec;
  // inicializa variaveis do TAD
  ml.count = 1;  ml.ativo = MLATIVO;  ml.fase = 0;
  // imprime registro inicial
  int retprint = fprintf(ml.log, "I %ld %ld.%.9ld\n",
                          ml.count, tp.tv_sec, tp.tv_nsec);
  erroAssert(retprint >= 0, "Nao foi possivel escrever registro");
  return result;
}
```


Biblioteca memlog: memlog.c

```
int ativaMemLog()  
// Descricao: ativa o registro de acessos  
// Entrada: nao tem  
// Saida: MLATIVO  
{ ml.ativo = MLATIVO;  
  return MLATIVO;  
}  
  
int desativaMemLog()  
// Descricao: desativa o registro de acessos  
// Entrada: nao tem  
// Saida: MLINATIVO  
{ ml.ativo = MLINATIVO;  
  return MLINATIVO;  
}  
  
int defineFaseMemLog(int f)  
// Descricao: define a fase de registro de acessos  
// Entrada: f  
// Saida: valor de f  
{ ml.fase = f;  
  return f;  
}
```

Biblioteca memlog: memlog.c

```
int leMemLog(long int pos, long int tam, int id)
// Descricao: registra acesso de leitura de tam bytes na posicao pos
// Entrada: pos,tam
// Saida: resultado da obtencao do relógio
{ // verifica se registro esta ativo
  if (ml.ativo == MLINATIVO) return 0;
  // captura tempo atual
  struct timespec tp, tdif;
  int result = clock_gettime(ml.clk_id,&tp);
  // calcula a diferenca com tempo inicial, para economia de armazenamento
  clkDifMemLog(ml.inittime,tp,&tdif);
  // atualiza contador
  ml.count++;
  // imprime registro
  int retprint = fprintf(ml.log,"L %d %ld %d %ld.%.9ld %ld %ld\n",
                        ml.fase, ml.count, id, tdif.tv_sec, tdif.tv_nsec, pos, tam);
  erroAssert(retprint>=0,"Nao foi possivel escrever registro");
  return result;
}
```

Biblioteca memlog: memlog.c

```
int escreveMemLog(long int pos, long int tam, int id)
// Descricao: registra acesso de escrita de tam bytes na posicao pos
// Entrada: pos, tam
// Saida: resultado da obtencao do relógio
{ // verifica se registro esta ativo
    if (ml.ativo == MLINATIVO) return 0;
    // captura tempo atual
    struct timespec tp,tdif;
    int result = clock_gettime(ml.clk_id,&tp);
    // calcula a diferenca com tempo inicial, para economia de armazenamento
    clkDifMemLog(ml.inittime,tp,&tdif);
    // atualiza contador
    ml.count++;
    // imprime registro
    int retprint = fprintf(ml.log,"E %d %ld %d %ld.%.9ld %ld %ld\n",
        ml.fase, ml.count, id, tdif.tv_sec, tdif.tv_nsec, pos, tam);
    erroAssert(retprint>=0,"Nao foi possivel escrever registro");
    return result;
}
```

Biblioteca memlog: memlog.c

```
int finalizaMemLog()  
// Descricao: finaliza o registro de acessos a memoria  
// Entrada: nao tem  
// Saida: resultado da obtencao do relógio  
{ // captura o tempo atual  
    struct timespec tp, tdif;  
    int result = clock_gettime(ml.clk_id,&tp);  
    // calcula a diferenca com tempo inicial, para economia de armazenamento  
    clkDifMemLog(ml.inittime,tp,&tdif);  
    // atualiza contador  
    ml.count++;  
    // imprime registros finais  
    int retprint = fprintf(ml.log,"F %ld %ld.%.9ld %ld.%.9ld\n", ml.count,  
                           tp.tv_sec,tp.tv_nsec, tdif.tv_sec,tdif.tv_nsec);  
    erroAssert(retprint>=0,"Nao foi possivel escrever registro");  
    // fecha arquivo de registro  
    int retclose = fclose(ml.log);  
    erroAssert(retclose == 0,"Nao foi possivel fechar o arquivo de registro");  
    // atualiza variavel de estado  
    ml.ativo = MLINATIVO;  
    return result;  
}
```

Usando memlog: matop.c

```
void uso()  
// Descricao: imprime as opcoes de uso  
// Entrada: nao tem  
// Saida: impressao das opcoes de linha de comando  
{  
    fprintf(stderr,"matop\n");  
    fprintf(stderr,"\t-s \t(somar matrizes) \n");  
    fprintf(stderr,"\t-m \t(multiplicar matrizes) \n");  
    fprintf(stderr,"\t-t \t(transpor matriz)\n");  
    fprintf(stderr,"\t-p <arq>\t(arquivo de registro de acesso)\n");  
    fprintf(stderr,"\t-l \t(registrar acessos a memoria)\n");  
    fprintf(stderr,"\t-x <int>\t(primeira dimensao)\n");  
    fprintf(stderr,"\t-y <int>\t(segunda dimensao)\n");  
}
```

Usando memlog: matop.c

```
while ((c = getopt(argc, argv, "smtp:x:y:lh")) != EOF)
switch(c) {
    case 'm':avisoAssert(opescolhida===-1,"Mais de uma operacao
escolhida");
        opescolhida = OPMULTIPLICAR;
        break;
    case 's':avisoAssert(opescolhida===-1,"Mais de uma operacao
escolhida");
        opescolhida = OPSOMAR;
        break;
    case 't':avisoAssert(opescolhida===-1,"Mais de uma operacao
escolhida");
        opescolhida = OPTRANSPOR;
        break;
    case 'p':strcpy(lognome,optarg);
        break;
    case 'x':optx = atoi(optarg);
        break;
    case 'y':pty = atoi(optarg);
        break;
    case 'l':regmem = 1;
        break;
    case 'h':
default: uso(); exit(1);
```

Usando memlog: matop.c

```
// iniciar registro de acesso
iniciaMemLog(lognome);

// ativar ou nao o registro de acesso
if (regmem){
    ativaMemLog();
}
else{
    desativaMemLog();
}

.....

return finalizaMemLog();
```

Usando memlog: mat.c

```
void multiplicaMatrizes(mat_tipo *a, mat_tipo *b, mat_tipo *c)
// Descricao: multiplica as matrizes a e b e armazena o resultado em c
// Entrada: a,b
// Saida: c
{
    int i,j,k;
    // verifica a compatibilidade das dimensoes
    erroAssert(a->tamy==b->tamx,"Dimensoes incompativeis");
    // cria e inicializa a matriz c
    criaMatriz(c,a->tamx, b->tamy,c->id);
    inicializaMatrizNula(c);
    // realiza a multiplicacao de matrizes
    for (i=0; i<c->tamx;i++){
        for (j=0; j<c->tamy;j++){
            for (k=0; k<a->tamy;k++){
                c->m[i][j] += a->m[i][k]*b->m[k][j];
                LEMEMLOG((long int) (&(a->m[i][k])),sizeof(double),a->id);
                LEMEMLOG((long int) (&(b->m[k][j])),sizeof(double),b->id);
                ESCREVEMEMLOG((long int) (&(c->m[i][j])),sizeof(double),c->id);
            }
        }
    }
}
```


Usando memlog: matop

Opções matop:

matop

-s	(somar matrizes)
-m	(multiplicar matrizes)
-t	(transpor matriz)
-c <arq>	(cria matriz e salva em arq)
-p <arq>	(arquivo de registro de acesso)
-l	(registrar acessos a memoria)
-x <int>	(primeira dimensao)
-y <int>	(segunda dimensao)

Linha de comando:

```
bin/matop -m -p /tmp/multlog.out -l -x 5 -y 5
```

Usando memlog: matop.c

```
case OPMULTIPLICAR:
```

```
    // cria matrizes a e b aleatorias, que sao multiplicadas para matriz c  
    // matriz c é impressa e todas as matrizes sao destruidas
```

```
    defineFaseMemLog(0);
```

```
    criaMatriz(&a, optx, opty, 0);
```

```
    inicializaMatrizAleatoria(&a);
```

```
    criaMatriz(&b, opty, optx, 1);
```

```
    inicializaMatrizAleatoria(&b);
```

```
    criaMatriz(&c, optx, optx, 2);
```

```
    inicializaMatrizNula(&c);
```

```
    defineFaseMemLog(1);
```

```
    acessaMatriz(&a);
```

```
    acessaMatriz(&b);
```

```
    acessaMatriz(&c);
```

```
    multiplicaMatrizes(&a, &b, &c);
```

```
    defineFaseMemLog(2);
```

```
    acessaMatriz(&c);
```

```
    if (regmem) imprimeMatriz(&c);
```

```
    destroiMatriz(&a);
```

```
    destroiMatriz(&b);
```

```
    destroiMatriz(&c);
```

```
break;
```

Usando memlog: mult.log

```
I 1 616722.226236424
E 0 2 0 0.000012139 140725505023504 8
E 0 3 0 0.000016063 140725505023512 8
E 0 4 0 0.000016811 140725505023520 8
E 0 5 0 0.000017406 140725505023528 8
E 0 6 0 0.000018033 140725505023536 8
...
L 1 420 1 0.000280593 140725505023904 8
E 1 421 2 0.000281198 140725505024048 8
L 1 422 0 0.000281743 140725505023584 8
L 1 423 1 0.000282253 140725505023752 8
E 1 424 2 0.000282786 140725505024056 8
L 1 425 0 0.000283299 140725505023592 8
L 1 426 1 0.000283810 140725505023792 8
E 1 427 2 0.000284342 140725505024056 8
...
F 652 616722.226709428 0.000473004
```

Usando memlog: mult.log

Evento de Inicio:

I 1 616722.226236424

- **I**: Rótulo de início
- **1**: Identificador de evento
- **616722.226236424**: Tempo absoluto de início em segundos

Eventos de Leitura e Escrita:

L 1 420 1 0.000280593 140725505023904 8

E 1 421 2 0.000281198 140725505024048 8

- **L/E**: Rótulo de Leitura ou Escrita
- **1**: Fase
- **420/421**: Identificador de evento
- **1**: Identificador de estrutura de dados
- **0.000280593**: Tempo desde o início
- **140725505023904**: Endereço do acesso
- **8**: Tamanho do dado acessado

Evento de Fim:

F 652 616722.226709428 0.000473004

- **F**: Rótulo de fim
- **652**: Identificador de evento
- **616722.226709428**: Tempo absoluto de fim em segundos
- **0.000473004**: Tempo desde o início

Usando memlog: fixaddr

```
I 1 616722.226236424
E 0 2 0 0.000012139 140725505023504 8
E 0 3 0 0.000016063 140725505023512 8
E 0 4 0 0.000016811 140725505023520 8
E 0 5 0 0.000017406 140725505023528 8
E 0 6 0 0.000018033 140725505023536 8
...
L 1 420 1 0.000280593 140725505023904 8
E 1 421 2 0.000281198 140725505024048 8
L 1 422 0 0.000281743 140725505023584 8
L 1 423 1 0.000282253 140725505023752 8
E 1 424 2 0.000282786 140725505024056 8
L 1 425 0 0.000283299 140725505023592 8
L 1 426 1 0.000283810 140725505023792 8
E 1 427 2 0.000284342 140725505024056 8
...
F 652 616722.226709428 0.000473004
```

```
I 1 616722.226236424
E 0 2 0 0.000012139 0 8
E 0 3 0 0.000016063 8 8
E 0 4 0 0.000016811 16 8
E 0 5 0 0.000017406 24 8
E 0 6 0 0.000018033 32 8
...
L 1 420 1 0.000280593 376 8
E 1 421 2 0.000281198 488 8
L 1 422 0 0.000281743 80 8
L 1 423 1 0.000282253 224 8
E 1 424 2 0.000282786 496 8
L 1 425 0 0.000283299 88 8
L 1 426 1 0.000283810 264 8
E 1 427 2 0.000284342 496 8
...
F 652 616722.226709428 0.000473004
```

analisamem: make use

```
if test -d /tmp/out; then rm -rf /tmp/out; fi
mkdir /tmp/out ; mkdir /tmp/out/teste
fixaddr/fixaddr.csh teste/multlog.out /tmp/out
$(EXE) -i /tmp/out/teste/multlog.out.fixed -p /tmp/out/mult
fixaddr/fixaddr.csh teste/somalog.out /tmp/out
$(EXE) -i /tmp/out/teste/somalog.out.fixed -p /tmp/out/soma
fixaddr/fixaddr.csh teste/transplog.out /tmp/out
$(EXE) -i /tmp/out/teste/transplog.out.fixed -p /tmp/out/transp
gnuplot /tmp/out/*.gp
ls /tmp/out/
```

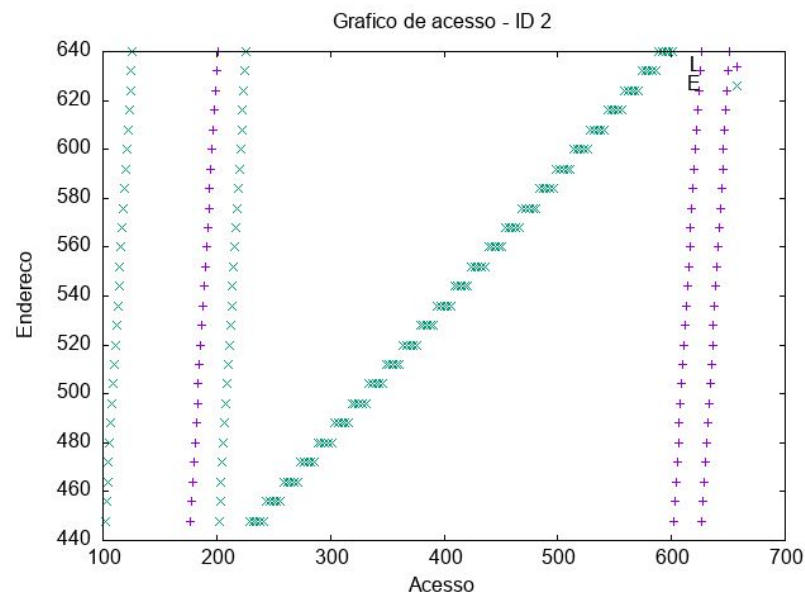
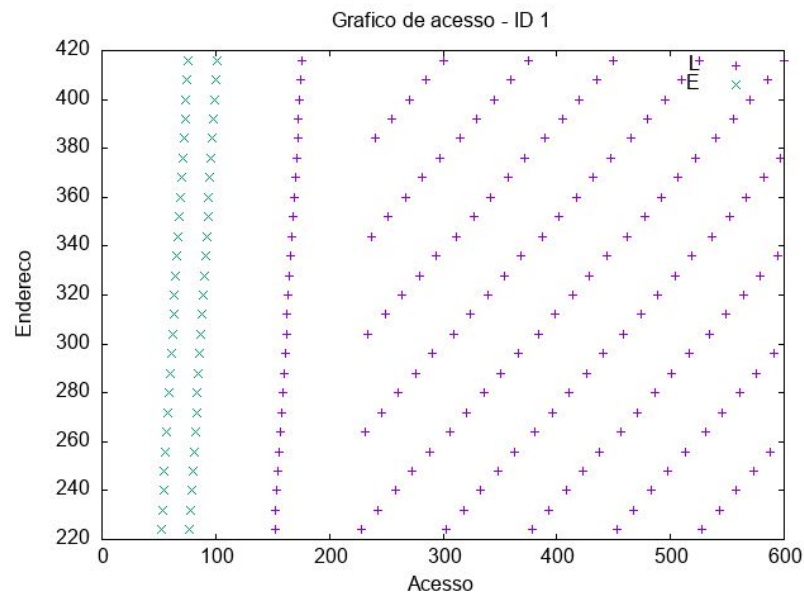
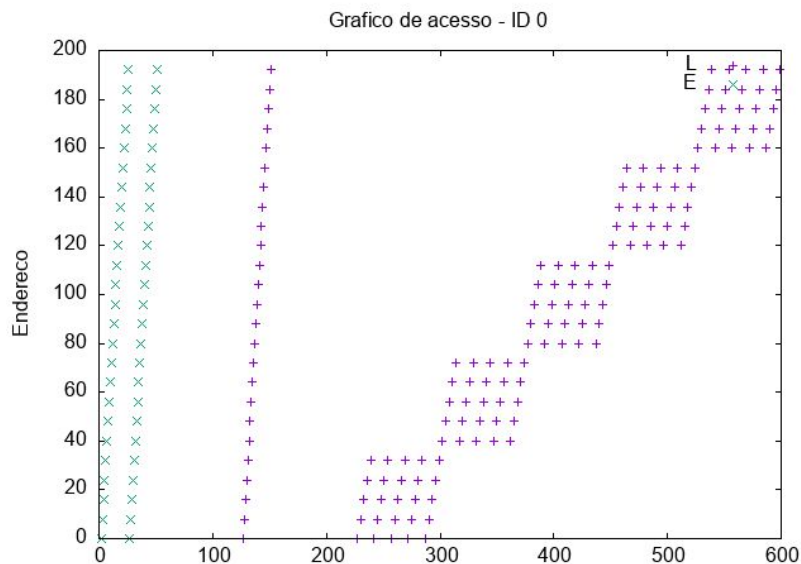
Resultado para transplog:

transp-acesso-0-0.gpdat	transp-distp-0.png	transp-hist-1-0.png
transp-acesso-0.gp	transp-hist-0-0.gp	transp-hist-2-0.gp
transp-acesso-0.png	transp-hist-0-0.gpdat	transp-hist-2-0.gpdat
transp-acesso-1-0.gpdat	transp-hist-0-0.png	transp-hist-2-0.png
transp-acesso-2-0.gpdat	transp-hist-1-0.gp	transp-distp-0.gp
transp-hist-1-0.gpdat		

analisamem: Mapa de Acesso

```
set term png
set output "/tmp/out/mult-acesso-0.png"
set title "Grafico de acesso - ID 0"
set xlabel "Acesso"
set ylabel "Endereco"
plot "/tmp/out/mult-acesso-0-0.gpdat" u 2:4
w points t "L",
"/tmp/out/mult-acesso-1-0.gpdat" u 2:4 w
points t "E"
```

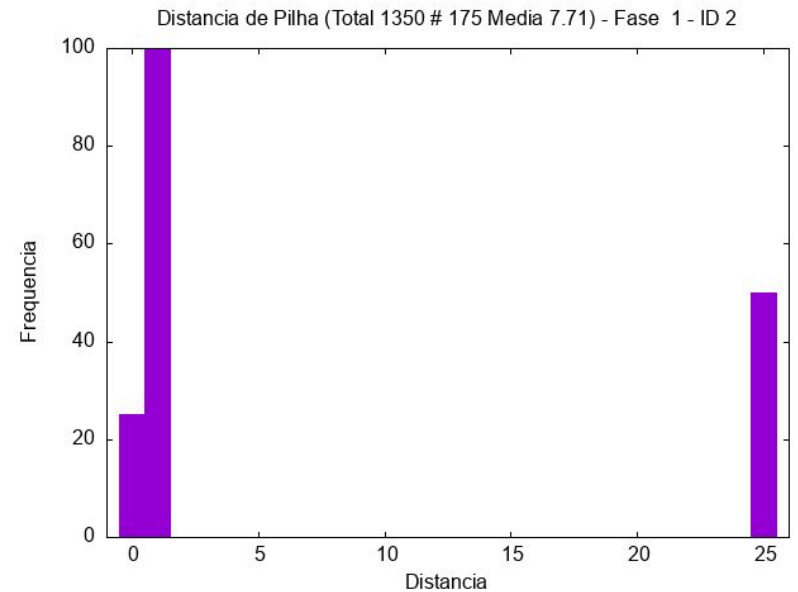
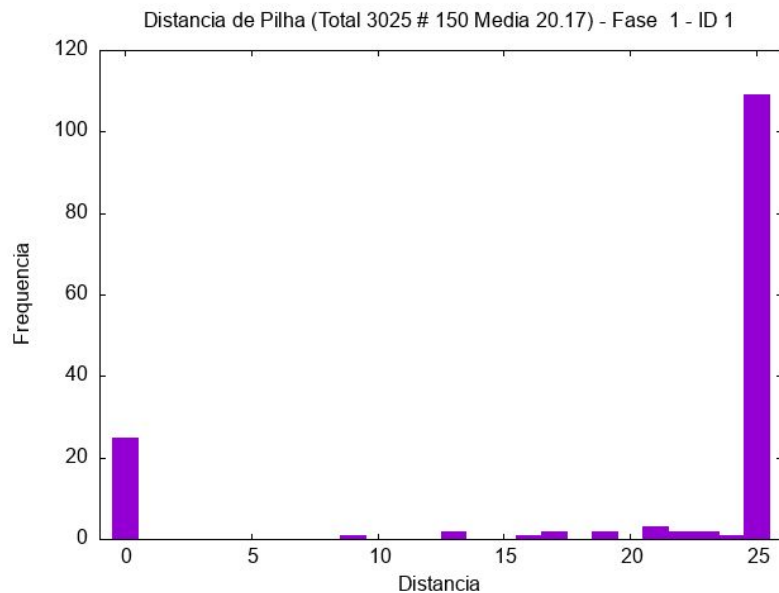
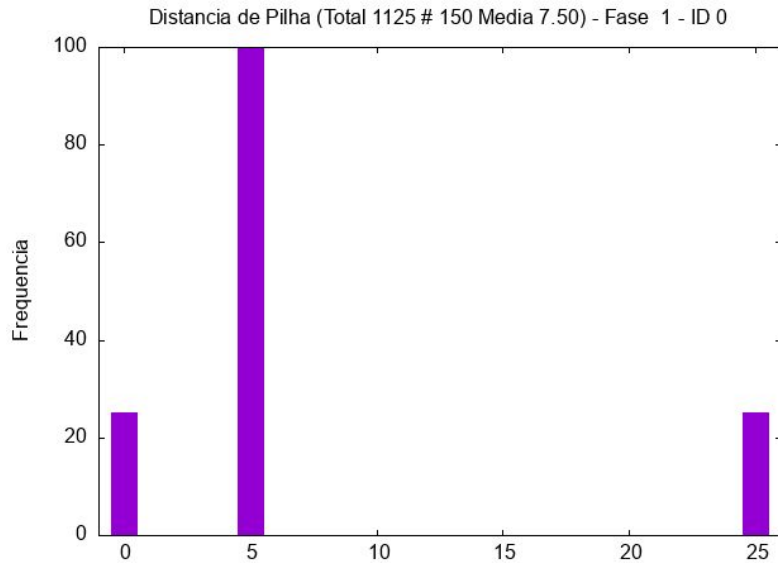
analisamem: Mapa de Acesso



analisamem: Histograma de DP

```
set term png
set output "/tmp/out/mult-hist-0-0.png"
set style fill solid 1.0
set title "Distancia de Pilha (Total 625 #
50 Media 12.50) - Fase 0 - ID 0"
set xlabel "Distancia"
set ylabel "Frequencia"
plot [-1:26] "/tmp/out/mult-hist-0-0.gpdat"
u 3:4 w boxes t ""
```

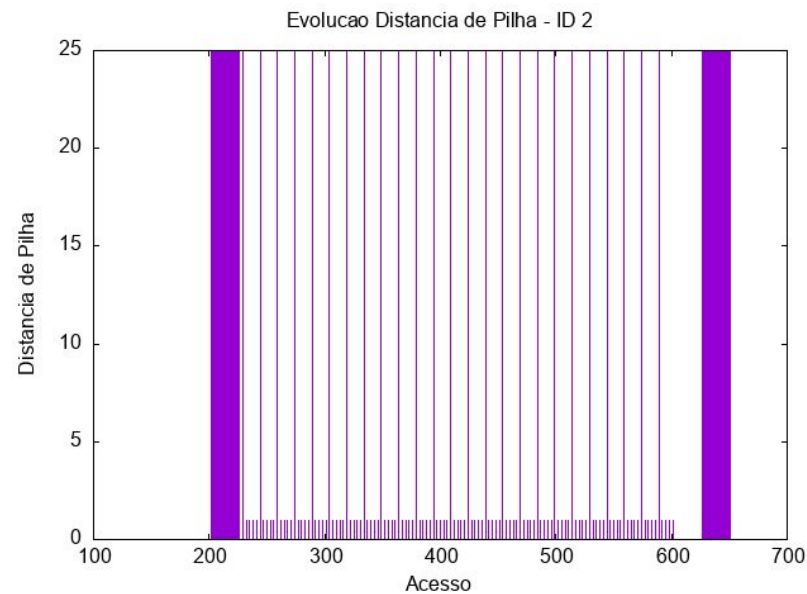
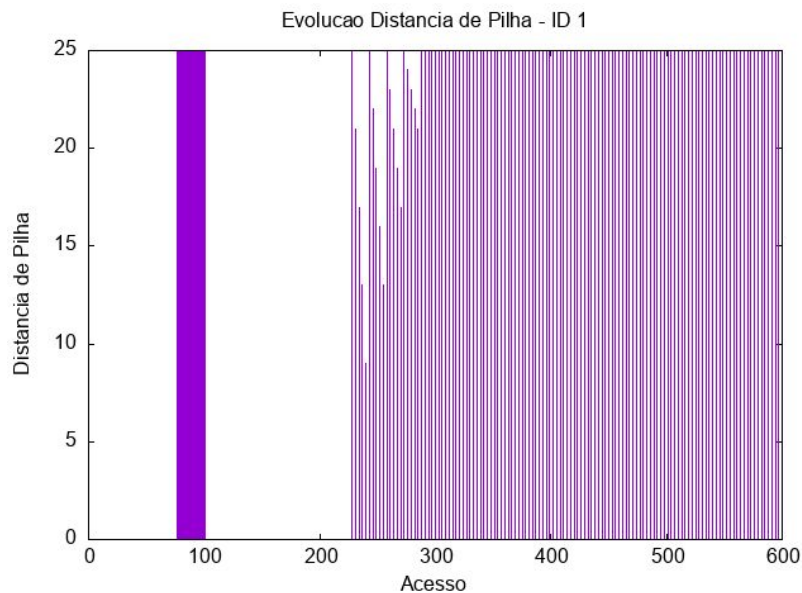
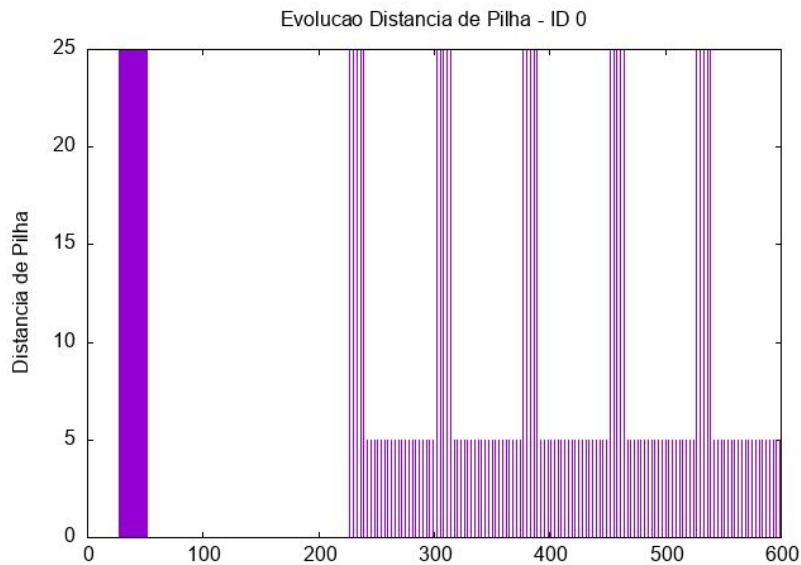
analisamem: Histograma de DP



analisamem: Evolução DP

```
set term png
set output "/tmp/out/mult-distp-0.png"
set title "Evolucao Distancia de Pilha - ID
0"
set xlabel "Acesso"
set ylabel "Distancia de Pilha"
plot "/tmp/out/mult-acesso-2-0.gpdat" u 2:5
w impulses t ""
```

analisamem: Evolução DP



Executando Experimentos

- Como o desempenho é medido em termos de tempo de execução, vários cuidados tem que ser tomados:
 - ❑ Desative o máximo de programas quando for medir
 - ❑ Não execute programas nem deixe entrar em descanso
 - ❑ Verifique a carga da máquina com aplicativos tipo `top`

```
top - 18:15:56 up 7 days,  2:15,  1 user,  load average: 1,48, 1,59, 1,81
Tasks: 295 total,   1 running, 294 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6,0 us,   3,1 sy,   0,0 ni, 90,6 id,   0,3 wa,   0,0 hi,   0,0 si,   0,0 st
MiB Mem : 15738,0 total, 917,8 free, 12785,4 used,  2034,8 buff/cache
MiB Swap:  2048,0 total,   0,0 free,  2048,0 used.  1168,4 avail Mem
```

PID	USER	PR	NI	VIRTRES	SHR	S	%CPU	%MEM	TIME+	COMMAND
76464	meira	20	0	5754044	1,4g	41588	S	12,3	9,2	184:45.12 Web Con+
2320	meira	20	0	1690660	463756	1812	S	4,6	2,9	3:35.33 snap-st+
1090	root	20	0	337284	4364	1800	S	4,3	0,0	38:43.56 Network+
2060	meira	20	0	4927440	263692	14320	S	4,0	1,6	398:06.71 gnome-st

Executando Experimentos

- Não use máquinas diferentes, resultados não serão comparáveis
- Em relação ao matop, desative registro de acessos à memória (não use a opção `-l`)
- Tempo de execução pode ser obtido no arquivo de saída do registro de acessos:
- Multiplicação de Matrizes 100x100
 - Tempo de execução **sem** registro de acesso à memória

```
I 1 613296.868793901
F 2 613296.875192111 0.006398210
```
 - Tempo de execução **com** registro de acesso à memória

```
I 1 613587.026878001
F 4070002 613588.526492373 1.499614372
```

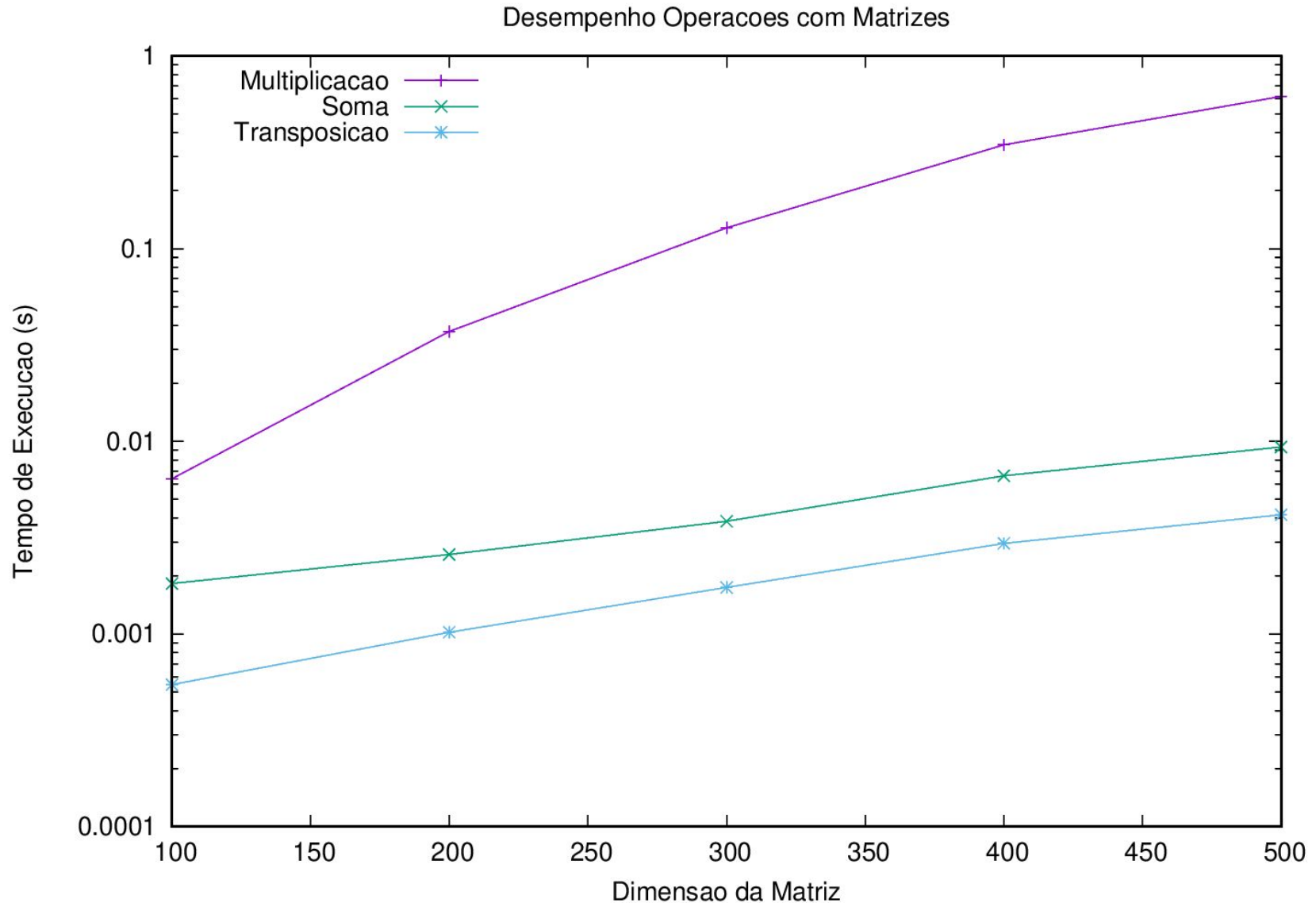
Resultados matop: perf.gpdat

100	0.006398210	0.001829699	0.000546951
200	0.037136802	0.002587916	0.001020990
300	0.128376628	0.003845188	0.001746546
400	0.345936870	0.006618190	0.002952549
500	0.616801968	0.009343938	0.004156377

Resultados matop: perf.gp

```
set term postscript eps color 14
set output "perf.eps"
set title "Desempenho Operacoes com Matrices"
set xlabel "Dimensao da Matriz"
set ylabel "Tempo de Execucao (s)"
set logscale y
set key left top
plot "perf.gpdat" u 1:2 w linesp t "Multiplicacao", \
    "perf.gpdat" u 1:3 w linesp t "Soma", \
    "perf.gpdat" u 1:4 w linesp t "Transposicao"
```


Resultados matop: gnuplot perf.gp



Estruturas de Dados

Biblioteca memlog
Aplicativo analisamem

Professores: Marcio Santos
Wagner Meira Jr