

UNIVERSIDADE ESTADUAL DE SANTA CRUZ

PATRICK SILVA FERRAZ

**FERRAMENTAS PARA GERENCIAR RECURSOS DO COMPUTADOR**

ILHÉUS - BA

2018

1. **Ferramentas para mensurar o tempo de execução de uma função específica dentro de um código.**

**Biblioteca <time.h> em C/C++**

**(Possui no CACAU, biblioteca padrão de C)**

**Descrição:**

Nativo de C/C++

**tipo da variável:** clock\_t

**Uso:**

main()

{

clock\_t start, end;

start = clock();

//CÓDIGO

end = clock();

double tempo = (end - start)/(double)CLOCKS\_PER\_SEC;

}

**Obs.:** Mede o tempo de uso da CPU e não da aplicação, ou seja, não serve para calcular tempo de cada thread em processamento paralelo.

**Mais em:** http://www.cplusplus.com/reference/ctime/

**Biblioteca <sys/time.h> em C/C++**

**(Possui no CACAU)**

**Descrição:**

Somente para sistemas UNIX e seus derivados.

**tipo da variável:**

struct timeval {  
 time\_t tv\_sec; /\* seconds \*/

suseconds\_t tv\_usec; /\* microseconds \*/  
};

struct timezone {

int tz\_minuteswest; /\* minutes west of Greenwich \*/  
 int tz\_dsttime; /\* type of DST correction \*/

};

**protótipo da função:**

int gettimeofday(struct timeval \*tv, struct timezone \*tz);

**Uso:**

main()

{

struct timeval t\_i, t\_f;   
 gettimeofday(&t\_i,NULL);

//CÓDIGO  
 gettimeofday(&t\_f,NULL);   
 double tf = (double)t\_f.tv\_usec + ((double)t\_f.tv\_sec \* (1000000.0));  
 ti = (double)t\_i.tv\_usec + ((double)t\_i.tv\_sec \* (1000000.0));

}

**Biblioteca <omp.h> em C/C++**

**(Possui no CACAU, suportado por padrão pelo gcc [depende da versão])**

**Descrição:**

Necessário compilar com a diretiva *-fopenmp.*

**Uso:**

main()

{

double start = omp\_get\_wtime( );

//CÓDIGO  
 double end = omp\_get\_wtime( );  
}

1. **Ferramentas para mensurar o tempo de execução de uma aplicação**

**Terminal Linux:**

**(Possui no CACAU)**

***time*** informa o tempo de execução da aplicação ou comando via terminal

**Uso:**

*$* ***time*** *./aplicacao*

**(Possui no CACAU)**

***times*** informa o tempo de execução da aplicação ou comando via terminal, porém não exibe o resultado da execução, mostrando somente o tempo.

**Uso:**

*$* ***times*** *./aplicacao*

**Gprof**

**(Possui no CACAU)**

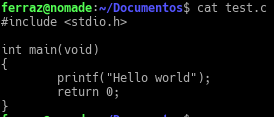
**Descrição:**

Gprof, em engenharia de software, é uma ferramenta para análise dinâmica (diferente da análise estática) da execução de programas escritos em linguagem C, Fortran e Pascal. O propósito usual desse tipo de análise é determinar o quanto de recurso computacional é consumido por cada parte do código, com o objetivo de otimizar o tempo de execução e diminuir quando possível o consumo de memória. Essa ferramenta pode ser usada em conjunto com o GCC. O Gprof é um projeto GNU, sob a licença GNU GPL.

**Uso:**

Compilar com gcc com a informação para gerar saída ao gprof

**Código.:**

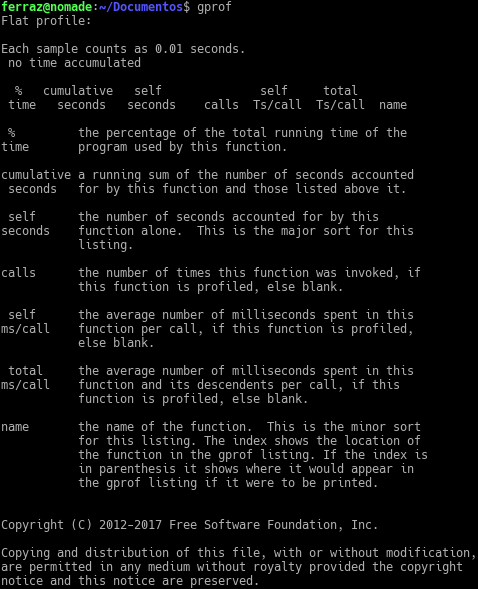
****

*$* ***gcc*** *-pg test.c // o gprof utiliza a.out (executável padrão gerado pelo gcc)*

*// após o comando acima o a.out é gerado*

*$* ***./a.out***  *// execute para gerar a saída gmon.out para o gprof*

*$* ***gprof*** *// basta executar o gprof na na pasta corrente*



Alguns testes podem ser realizados sem otimizações, dependendo da finalidade, com as diretivas abaixo:

*$* ***gcc*** *-std=c89 -pedantic -Wall -Werror -O0 -pg test.c*

**Outros**

Pode-se também utilizar as bibliotecas abaixo:

**Biblioteca <time.h> em C/C++**

**Biblioteca <sys/time.h> em C/C++**

**Biblioteca <omp.h> em C/C++**

Apenas colocando as chamadas das funções, dos exemplos em 1, no início e final do código.

1. **Ferramenta para monitorar o consumo de memória de uma aplicação**

**Valgrind**

**(Possui no CACAU)**

**Descrição:**

Valgrind é um software livre que auxilia o trabalho de depuração de programas criado por Julian Seward. Ele possui ferramentas que detectam erros decorrentes do uso incorreto da memória dinâmica, como por exemplo os vazamentos de memória, alocação e desalocação incorretas e acessos a áreas inválidas.  
O diferencial deste programa está no fato de que usa uma máquina virtual para simular o acesso à memória do programa em teste, eliminando a necessidade de uso de outras bibliotecas auxiliares ou mudanças drásticas no código.  
Apesar de estar direcionado para programas codificados em C ou C++, a máquina virtual torna possível o uso do Valgrind com programas que foram codificados em outras linguagens, como o Java.

**Instalação linux:**

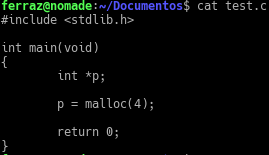
*$* ***sudo apt-get install*** *valgrind*

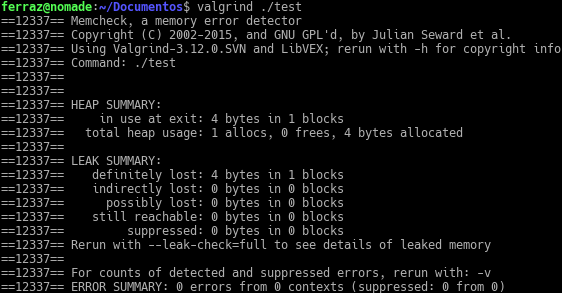
**Uso:**

*$* ***valgrind*** *./aplicacao*

**Obs:** Utilizado para verificar memory leak e uso de memória.

**Imagem de uma saída do valgrind:**

****

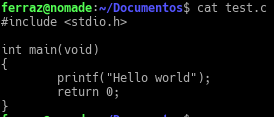
****

**Obs.:** Para informações detalhadas do memory leak basta executar com *--leak-check=full*

**Splint**

**(Possui no CACAU)**

Verifica o memory leak estaticamente (percorrendo o código fonte)

****

****

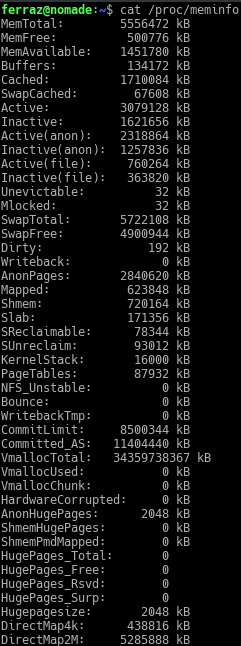
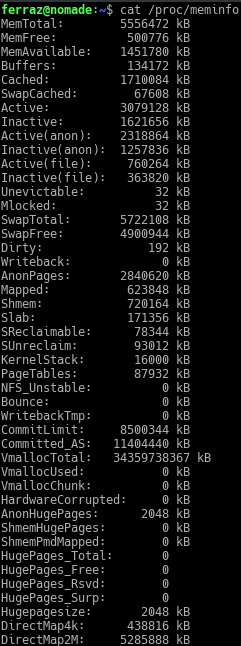
**Comandos nativos do Linux para medir memória**

**(O SO do CACAU é linux Red Hat, por conta disso, todos os comandos abaixo é possível realizar no CACAU)**

*$* ***free*** *// Informa alguns dados sobre a memória disponível e em uso no sistema*



*$* ***cat******/proc/meminfo*** *// detalhes adicionais não mostrados por free*

****

**Informações de memória**

*$* ***ps***

*$* ***top***

**Uso do /proc**

*$* ***cat /proc/****<pid>****/statm***

**Saída:**

tamanho total do programa |  
tamanho do conjunto de residentes |  
páginas compartilhadas |  
texto (código) |  
dados / pilha |  
biblioteca |  
páginas sujas |

*$* ***cat /proc/****<pid>****/status***

**Saída:**

Tamanho de Vm: 2772 kB  
Vm Lck: 0 kB - ???  
Vm RSS: 1624 kB  
Dados Vm: 404 kB  
Vm Stk: 24 kB  
Exe Vm: 608 kB  
Vm Lib: 1440 kB

*$* ***cat /proc/****<pid>****/maps*** *// mostra as áreas de memória que foram mapeadas*

**Obs.:** O *<pid>* do processo pode ser obtido através do comando ***top*** informando anteriormente.

1. **Ferramenta de profiling de aplicações**

**SLURM**

**(Possui no CACAU)**

O Slurm Workload Manager (anteriormente conhecido como Simple Linux Utility para Resource Management ou SLURM), ou Slurm , é um agendador de tarefas gratuito e de código aberto para Linux e kernels semelhantes ao Unix, usado por muitos dos supercomputadores e clusters de computadores do mundo. Ele fornece três funções principais. Primeiro, ele atribui acesso exclusivo e/ou não exclusivo a recursos (nós de computador) aos usuários por algum tempo para que eles possam executar o trabalho. Em segundo lugar, ele fornece uma estrutura para iniciar, executar e monitorar o trabalho (normalmente, um trabalho paralelo, como o MPI) em um conjunto de nós alocados. Finalmente, ele arbitra a contenção de recursos gerenciando uma fila de tarefas pendentes.

**Comandos:**

*$* ***sinfo*** *[-p parition\_name ou -M cluster\_name] // Informações sobre nós e partições*

*$* ***squeue*** *--user=nome\_de\_usuario // verifica o estado dos trabalho de um usuário*

*$* ***srun*** *// executar um comando nos nós de computação alocados*

*$* ***snodes*** *[nó do cluster/ estado da partição] // exibir informações do nó*

*$* ***fisbatch*** *// lançar um trabalho interativo*

*$* ***sranks*** *// listar prioridade de tarefas enfileiradas*

*$* ***sueff*** *user-name // obter a eficiência de um trabalho em execução*

*$* ***suacct*** *data-inicio user-name // obter informações de contabilidade do SLURM para tarefas de um usuário desde a data de início até agora*

*$* ***slist*** *<jobid> // obter informações sobre contabilidade e nó do SLURM para um trabalho*

*$* ***slogs*** *data-inicio lista-usuarios // obter informações sobre contabilidade de uso de recursos para as tarefas de um usuário desde a data de início até a data atual*

*$* ***stimes*** *// obter as horas de início estimadas para tarefas enfileiradas*

*$* ***scontrol*** *[-options] // utilizado para visualizar ou modificar configurações do SLURM*

*$* ***scontrol*** *show job <jobid> // relatório detalhado par ao trabalho*

*$* ***smap*** *// tela interativa com mapa de processos*

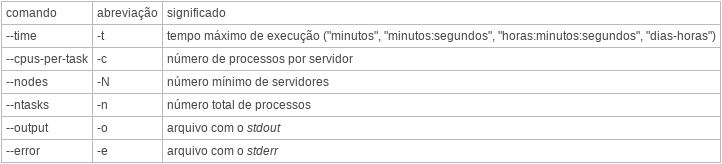
*$* ***sacct*** *// histórico dos processos já concluídos e a contabilidade*

**Script modelo:**

#!/bin/sh  
#SBATCH --time=1  
/bin/hostname

*$* ***sbatch*** *script-file // envia um script para execução posterior*

**Parâmetros:**



*$* ***salloc*** *// Alocar nós de computação para uso interativo*

*$* ***scancel*** *<jobid> // Cancelar um trabalho pendente ou em execução*

**Obs.:** SLURM é utilizado em 60% dos computadores do TOP500 (<https://en.wikipedia.org/wiki/Slurm_Workload_Manager>).

**Mais em:** <https://slurm.schedmd.com/>

**OpenHPC**

O OpenHPC é um conjunto de ferramentas FOSS (software gratuito e de código aberto) conduzidas pela comunidade para HPC (computação de alto desempenho) baseada em Linux. O OpenHPC não possui requisitos específicos de hardware.

O OpenHPC fornece uma coleção integrada e testada de componentes de software que, junto com uma distribuição Linux padrão suportada, pode ser usada para implementar um cluster de computação completo.

**Enduro / X**

Enduro/X é uma plataforma de middleware de código aberto para processamento de transações distribuídas. Ele é construído sobre APIs comprovadas, como XATMI e XA do X/Open. A plataforma é projetada para criar aplicativos baseados em microsserviços em tempo real com opção de clusterização. O Enduro/X funciona como uma alternativa de substituição estendida para o Oracle (R) Tuxedo (R). A plataforma usa filas do Kernel POSIX na memória que garantem uma alta taxa de transferência de comunicação entre processos .

**HTCondor**

O HTCondor é uma estrutura de software de computação de alto rendimento de código aberto para paralelização distribuída de granulação grosseira de tarefas computacionalmente intensivas. Ele pode ser usado para gerenciar a carga de trabalho em um cluster dedicado de computadores ou para distribuir o trabalho para computadores desktop ociosos - a chamada eliminação de ciclos. O HTCondor é executado nos sistemas operacionais Linux, Unix, Mac OS X, FreeBSD e Microsoft Windows. O HTCondor pode integrar tanto recursos dedicados (clusters montados em rack) quanto máquinas desktop não dedicadas (eliminação de ciclo) em um ambiente de computação.

**Moab Cluster Suite**

O Moab Cluster Suite é um pacote de gerenciamento de carga de trabalho de cluster, disponível na Adaptive Computing, Inc., que integra o agendamento, gerenciamento, monitoramento e relatório de cargas de trabalho de cluster. O Moab Cluster Suite simplifica e unifica o gerenciamento em um ou vários ambientes de hardware, sistema operacional, armazenamento, rede, licença e gerenciador de recursos.

**Outros:** OpenLava (GNU), Grid MP (Univa), Plataforma LSF (GNU, substituido por OpenLava), Oracle Grid Engine (Oracle), Xgrid (Apple).

**Referências**

<https://pt.wikipedia.org/wiki/Valgrind>

<https://pt.wikipedia.org/wiki/Gprof>

<https://elinux.org/Runtime_Memory_Measurement>

<https://ncc.unesp.br/its/projects/gridunesp/wiki/SLURM>

<https://en.wikipedia.org/wiki/Comparison_of_cluster_software>

<https://en.wikipedia.org/wiki/OpenHPC>

<https://en.wikipedia.org/wiki/Enduro/X>

<https://en.wikipedia.org/wiki/HTCondor>

<https://en.wikipedia.org/wiki/Moab_Cluster_Suite>