CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 20: Paralelismo com Threads

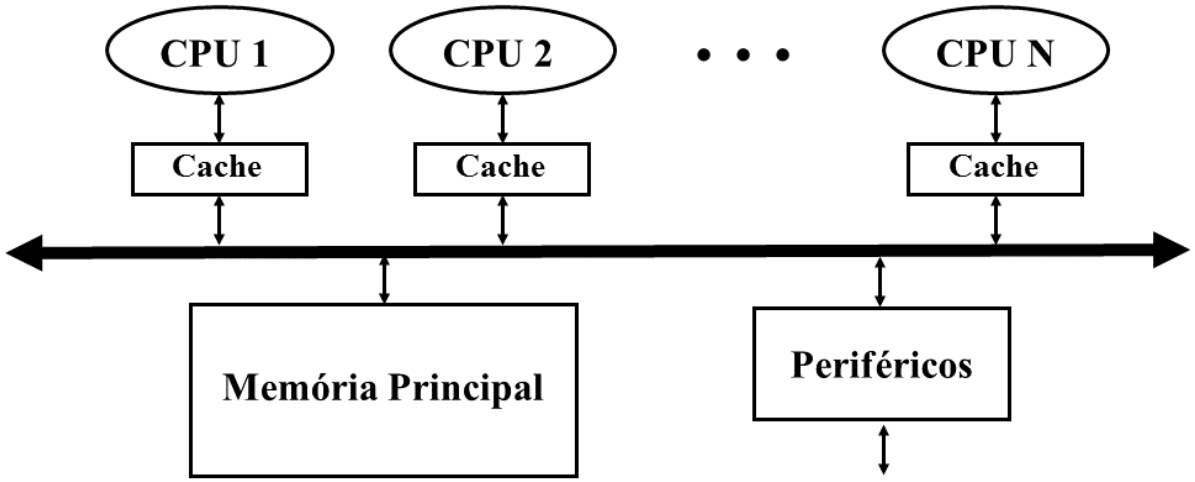
Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Programação em Mem.Compartilhada

- Objetivo: explorar todas as CPUs, minimizando o overhead de comunicação entre elas
- Paradigma preferencial: um *thread* em cada CPU
  - *thread* = fluxo de instruções, com registros e pilha próprios
  - Prog.: um único processo
  - Alocação thread/CPU é feita pelo Sist.Oper.
  - Comunicação via memória comum



Paralelismo com Threads

- Programação possível em vários níveis
  - Linguagem convencional + biblioteca (ex.: Pthreads)
  - Linguagem convencional + extensões (ex. OpenMP)
  - Inserção na linguagem (ex.: Java, C11, etc.)
- Vários tipos de threads:
  - Kernel-threads: gerenciados pelo S.O.
    - Chaveamento entre threads é mais custoso
  - User-threads: gerenciados pelo próprio usuário
    - Mais leves, porém devem ser escalonados pelo S.O.
  - Hardware-threads: suporte de hardware
    - Chaveamento entre threads pode ser feito a cada ciclo

Aspectos Essenciais com Threads

- Sincronização
  - Como criar/destruir threads, disciplinar os acessos às mesmas variáveis pelos threads
- Nomes e escopos de variáveis:
  - O que é conhecido por um thread (e somente por ele!)
- Escalonamento
  - Em qual núcleo o thread vai executar

Todos estes aspectos devem ser considerados para:

- Execução legal do programa (resultados corretos)
- Execução eficiente do programa (bom desempenho)

Programação com Pthreads

- Pthreads: POSIX threads
  - Padrão POSIX, mais de 100 funções definidas
  - Criação de threads, encerramento, sincronização, etc.
  - Interface para programas em C apenas
  - Todas as funções: *pthread_<nome>*
  - Arquivo a ser incluído: *pthread.h*
  - Compilação/link: *gcc –o prog prog.c –lpthread*
    - Invoca biblioteca *libpthread.a*
    - Disponível em quase todos os sistemas do tipo Unix

Exemplo com Pthreads

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
int global_var = 4 ;
void *func (void *value);
int main(argc,argv)
int argc; char *argv[];
{
    int i, count ;
    pthread_t *pts;
    count = global_var ;
    for (i=0; i<count; i++) pts=malloc(count*sizeof(pthread_t));
    /* Cria outros threads */
    for (i=0; i<count; i++) {
        pthread_create(&pts[i], NULL, func, (void*) i);
    }
    /* Thread principal faz algum trabalho */
    printf("Este e' o thread principal\n");
    /* Termina outros threads */
    for (i=0; i<count; i++) {
        pthread_join(pts[i], NULL);
    }
    free(pts);
}
void *func (void *value) {
    int local_var = global_var;
    /* thread realiza trabalho individual */
    printf("==> Este e' o thread %d de um total de %d\n", value, local_var);
    return NULL;
}
```

Exemplo com Pthreads (cont.)

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
int global_var = 4 ;
void *func (void *value);
int main(argc,argv)
int argc; char *argv[];
{
    int i, count ;
    pthread_t *pts;
    count = global_var ;
    for (i=0; i<count; i++) pts=malloc(count*sizeof(pthread_t));
    /* Cria outros threads */
    for (i=0; i<count; i++) {
      pthread_create(&pts[i], NULL, func, (void*) i);
    }
    /* Thread principal faz algum trabalho */
    printf("Este e' o thread principal\n");
```

variável global

Exemplo com Pthreads (cont.)

```
        pthread_create(&pts[i], NULL, func, (void*) i);
    }
    /* Thread principal faz algum trabalho */
    printf("Este e' o thread principal\n");
    /* Termina outros threads */
    for (i=0; i<count; i++) {
        pthread_join(pts[i], NULL);
    }
    free(pts);
}
void *func (void *value) {
    int local_var = global_var;
    /* thread realiza trabalho individual */
    printf("==> Este e' o thread %d de um total de %d\n", value, local_var);
    return NULL;
}
```

variável privativa do thread

Exemplo com Pthreads (cont.)

- Execução:

==> Este e' o thread 0 de um total de 4
==> Este e' o thread 1 de um total de 4
==> Este e' o thread 2 de um total de 4
==> Este e' o thread 3 de um total de 4
Este e' o thread principal