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

Aula 34: Programação MPI+OpenMP

Celso L. Mendes, Stephan Stephany

LAC / INPE

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

MPI com OpenMP

- Adição de threads a um programa MPI
  - Várias formas possíveis, compatíveis com MPI
  - Forma de criação dos threads não é definida por MPI
  - Forma comum: OpenMP
    - Diretivas OpenMP em loops ou regiões paralelas
    - Threads são genenciados pelo compilador compatível com OpenMP
  - Cenário de execução do programa MPI+OpenMP:
    - Um ou mais processos MPI (ranks) em cada nó
    - Vários threads por processo (rank), um em cada CPU

Programa MPI + OpenMP

- Número de threads num programa MPI+OpenMP:
  - Uma das formas possíveis de especificação é com a variável ambiental *OMP_NUM_THREADS*
  - Porém, algumas implementações de MPI não propagam o ambiente aos outros processos além do rank-0
    - Solução: criar código para propagar o ambiente do rank-0, e definir explicitamente no programa o número de threads

Exemplo:

```
if (rank==0) {
        nthreads_str = getenv("OMP_NUM_THREADS");
        if (nthreads_str)  nthreads=atoi(nthreads_str);
        else nthreads=1;
}
MPI_Bcast(&nthreads, 1, MPI_INT, 0, MPI_COMM_WORLD);
omp_set_num_threads(nthreads);
```

Caso Geral (Sem Threads)

```
int main(int argc,&argv)
{
        int buf[100];
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);

        for (i=0;i<100;i++)  {
                computa(buf[i]);

                < ... comunicação com MPI ... >
        }
        MPI_Finalize();
        return 0;
}
```

Caso MPI_THREAD_FUNNELED

```
int main(int argc,&argv)
{
        int buf[100], provided;
        MPI_Init_thread(&argc,&argv, MPI_THREAD_FUNNELED, &provided);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
#pragma omp parallel for
        for (i=0;i<100;i++)  {
                computa(buf[i]);
#pragma omp master
                < … comunicação com MPI … >
        }
        MPI_Finalize();
        return 0;
}
```

Threads existem para a execução do loop

Chamadas MPI continuam sendo feitas apenas pelo thread principal (thread que chamou MPI_Init)

Caso MPI_THREAD_SERIALIZED

```
int main(int argc,&argv)
{
        int buf[100], provided;
        MPI_Init_thread(&argc,&argv, MPI_THREAD_SERIALIZED, &provided);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
#pragma omp parallel for
        for (i=0;i<100;i++)  {
                computa(buf[i]);
#pragma omp critical
                < … comunicação com MPI … >
        }
        MPI_Finalize();
        return 0;
}
```

Threads existem para a execução do loop

Chamadas MPI estão na seção crítica, e são feitas por um thread de cada vez

Caso MPI_THREAD_MULTIPLE

```
int main(int argc,&argv)
{
        int buf[100], provided;
        MPI_Init_thread(&argc,&argv, MPI_THREAD_MULTIPLE, &provided);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
#pragma omp parallel for
        for (i=0;i<100;i++)  {
            computa(buf[i]);

            < ... comunicação com MPI ... >
         }
        MPI_Finalize();
        return 0;
}
```

Threads existem para a execução do loop

Chamadas MPI podem ser feitas livremente por qualquer thread

Caso MPI_THREAD_MULTIPLE

- Restrições:
  - Usuário deve garantir que operações coletivas num comunicador sejam feitas na mesma ordem pelos threads
  - Liberação de objetos alocados deve ser feita pelo mesmo thread que fez a alocação
  - Chamadas com bloqueio irão bloquear apenas o thread the fez a chamada; outros threads podem continuar a execução
  - Lembrar que o desempenho geral da biblioteca pode ser inferior ao do caso sem threads

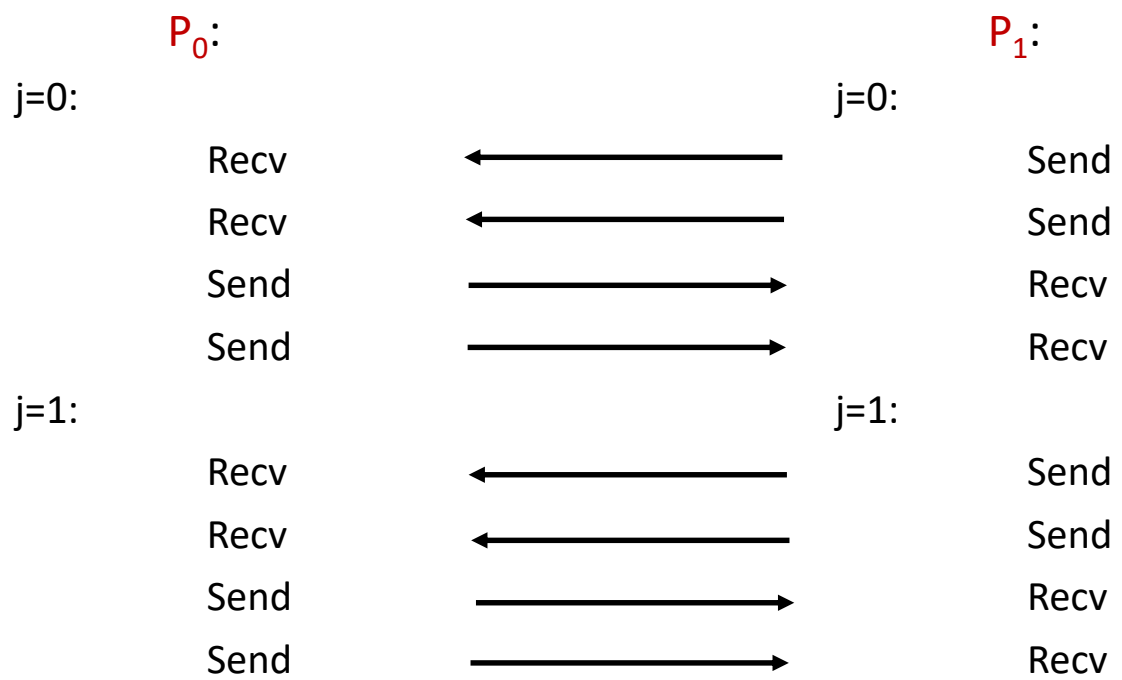
Perigos de MPI_THREAD_MULTIPLE

Exemplo:

```
for (j=0; j<2; j++) {
        if (rank==1) {
                for (i=0;i<2; i++)
                        MPI_Send(NULL,0,MPI_CHAR,0,0,MPI_COMM_WORLD);
                for (i=0; i<2; i++)
                        MPI_Recv(NULL,0,MPI_CHAR,0,0,MPI_COMM_WORLD,&stat);
        }
        else { /* rank==0 */
                for (i=0; i<2; i++)
                        MPI_Recv(NULL,0,MPI_CHAR,1,0,MPI_COMM_WORLD,&stat);
                for (i=0;i<2; i++)
                        MPI_Send(NULL,0,MPI_CHAR,1,0,MPI_COMM_WORLD);
        }
}
```

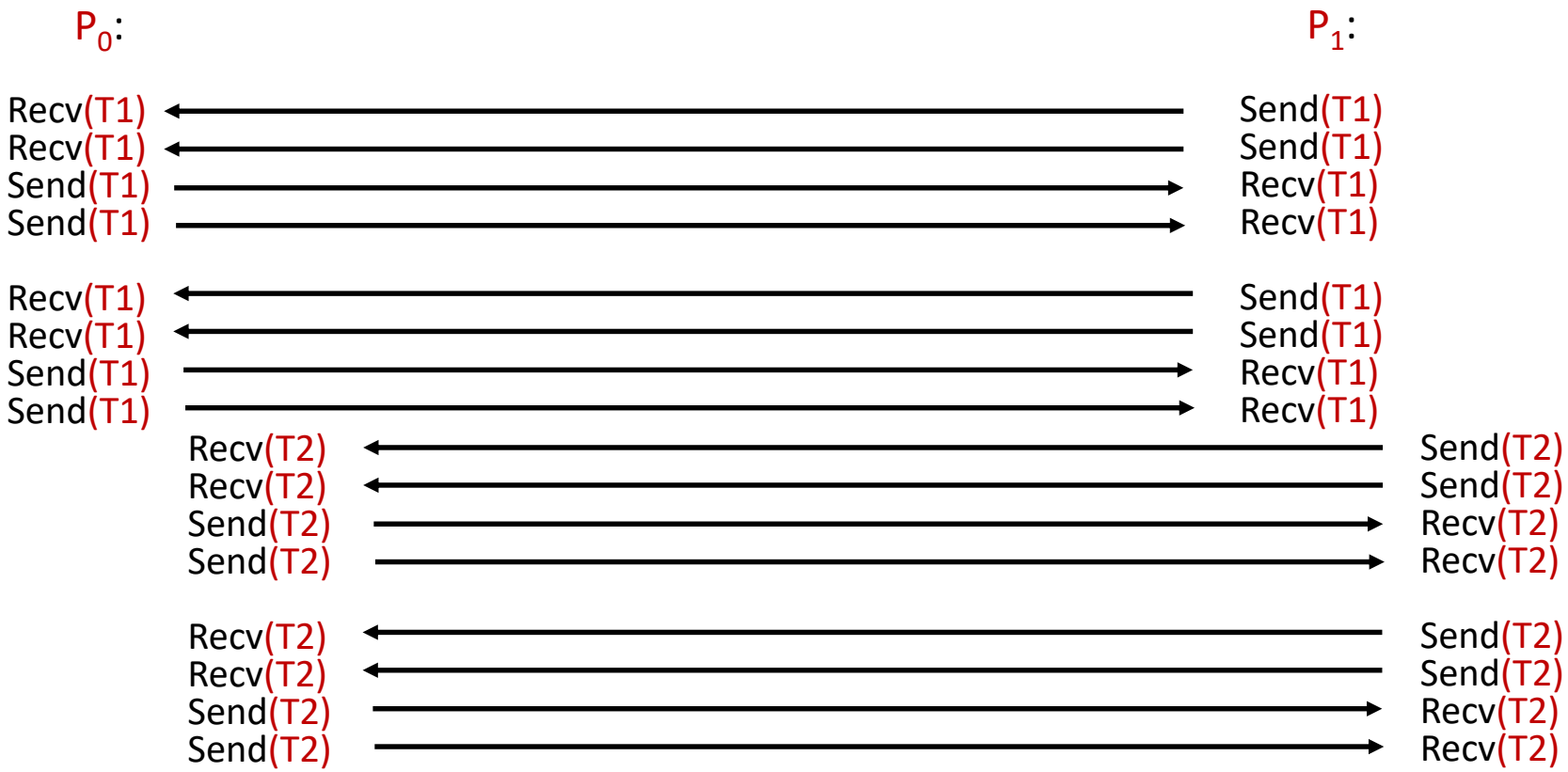
Perigos de MPI_THREAD_MULTIPLE

Ordem de execução normal, sem threads:



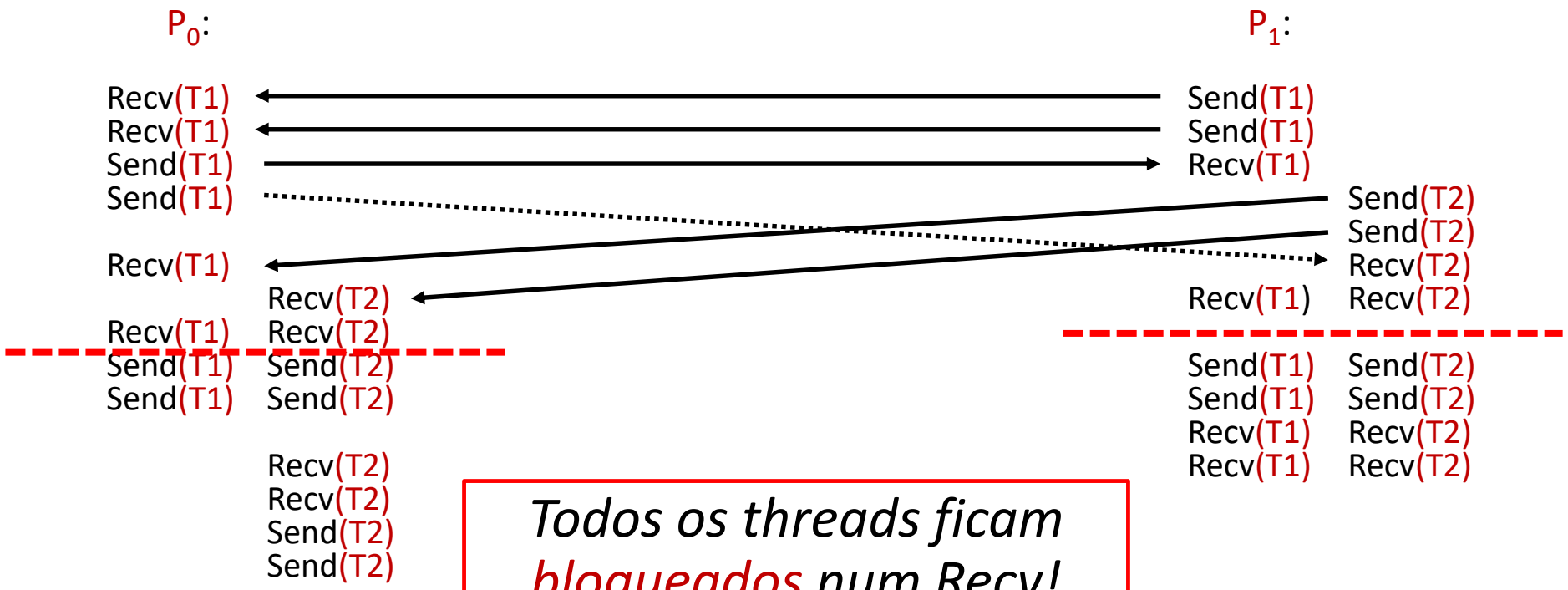
Perigos de MPI_THREAD_MULTIPLE

Ordem de execução esperada, com 2 threads T1,T2:



Perigos de MPI_THREAD_MULTIPLE

Ordem de execução possível, com 2 threads T1,T2:



Todos os threads ficam bloqueados num Recv!

Sumário

- Complemento a troca de mensagens em MPI:
  - MPI + Pthreads
  - MPI + OpenMP
  - MPI + shared-memory (ainda não visto)