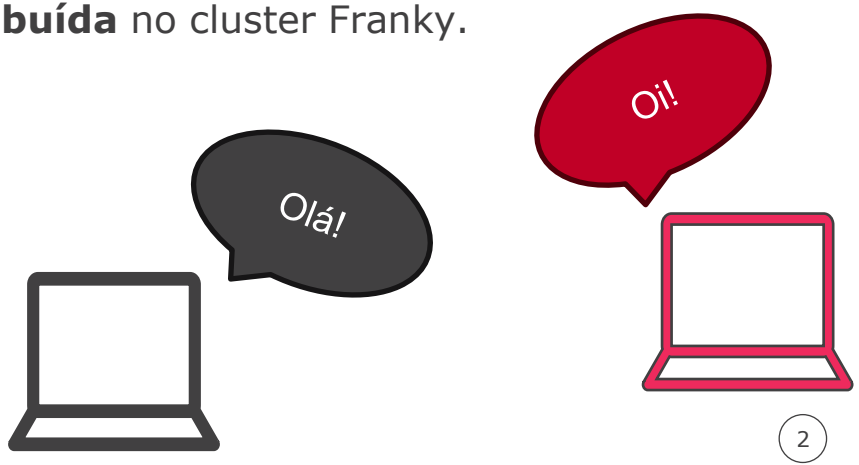
Insper

Programação Distribuída com MPI

Lícia Sales Costa Lima

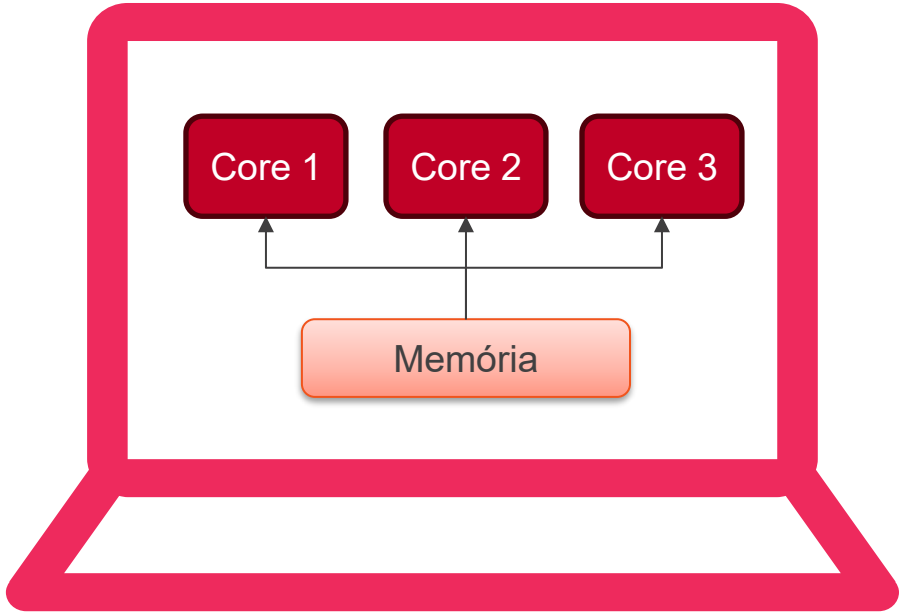
Objetivos da aula

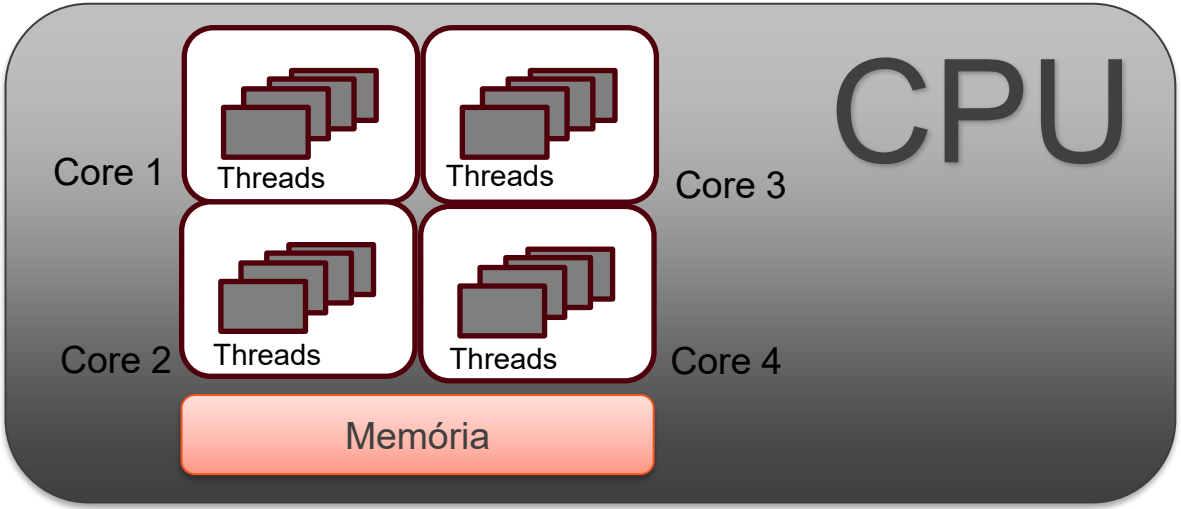
- Identificar as principais diferenças entre paralelismo com **memória compartilhada** e paralelismo com **memória distribuída**.
- Explicar **como ocorre a comunicação** entre processos em ambientes distribuídos usando MPI.
- Executar um programa com **memória distribuída** no cluster Franky.



Recapitulando...

- O problema precisa ser **paralelizável,** com tarefas **independentes entre si.**
- **Memória compartilhada** (todos os threads acessam a mesma RAM)
- Executado em **múltiplos núcleos da mesma máquina.**



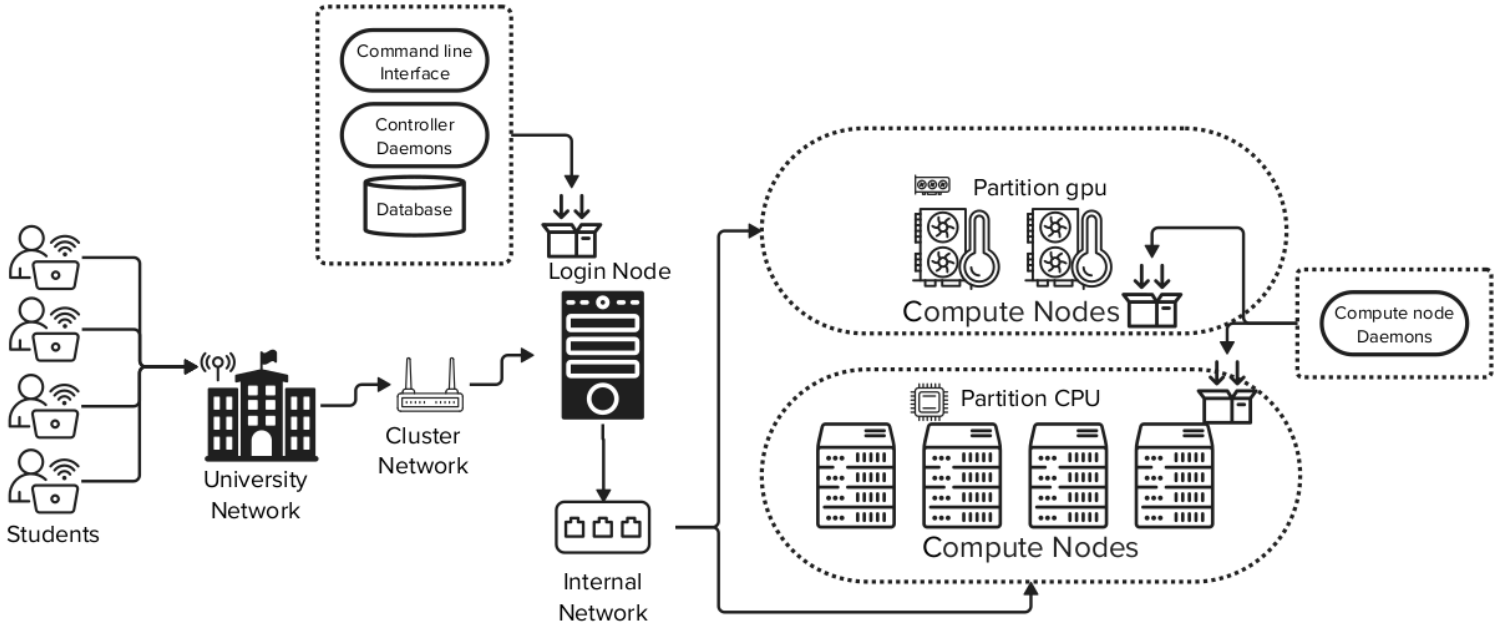


Cores = hardware → unidades físicas de processamento.
Threads = software → fluxos de execução que podem rodar dentro de um core.

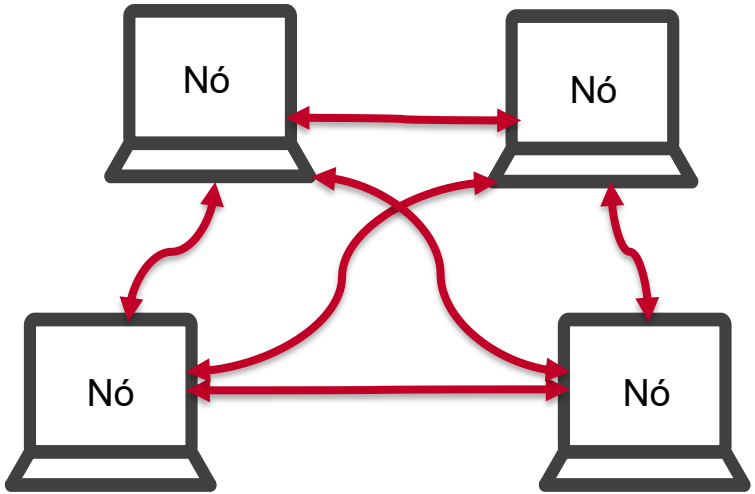
Cada core pode executar **uma ou mais threads** simultaneamente.

E se eu quiser mais de um nó no cluster?

Sistemas de HPC

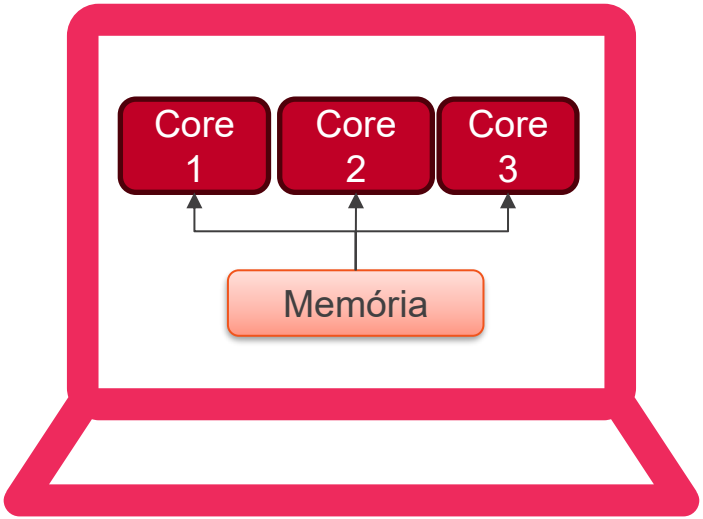


Programação distribuída:



Múltiplos processos rodando em **máquinas diferentes**

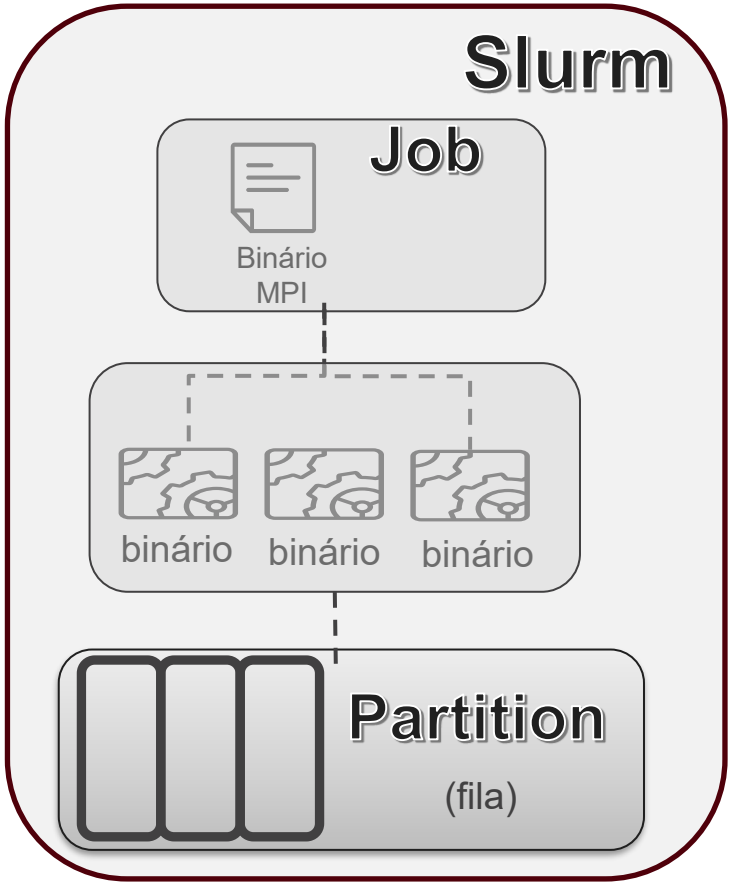
Programação paralela:



Múltiplos processos rodando em **uma mesma máquina**

MPI (Message Passing Interface) Conceitos:

- MPI é uma especificação mantida pelo MPI-Forum e se tornou um padrão para comunicação em sistemas distribuídos.
- Cada processo tem um identificador único (chamado rank).
- O mesmo binário está disponível em todos os nós de computação.

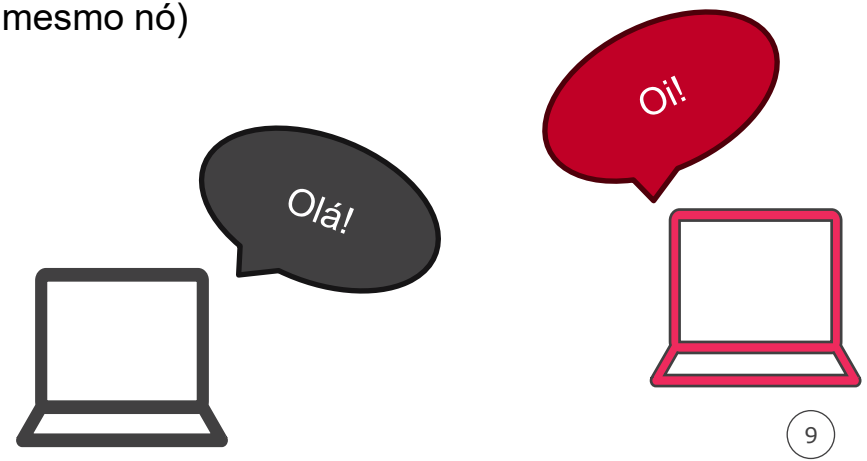


O que o MPI usa para trocar mensagens?

Implementações de MPI (como OpenMPI, MPICH) escolhem o meio mais eficiente de comunicação disponível

MPI não define os protocolos diretamente, mas usa **protocolos de transporte**, como:

- **TCP/IP** (em clusters convencionais)
- **InfiniBand** (em clusters de alto desempenho)
- **SHM** (shared memory, se estiverem no mesmo nó)



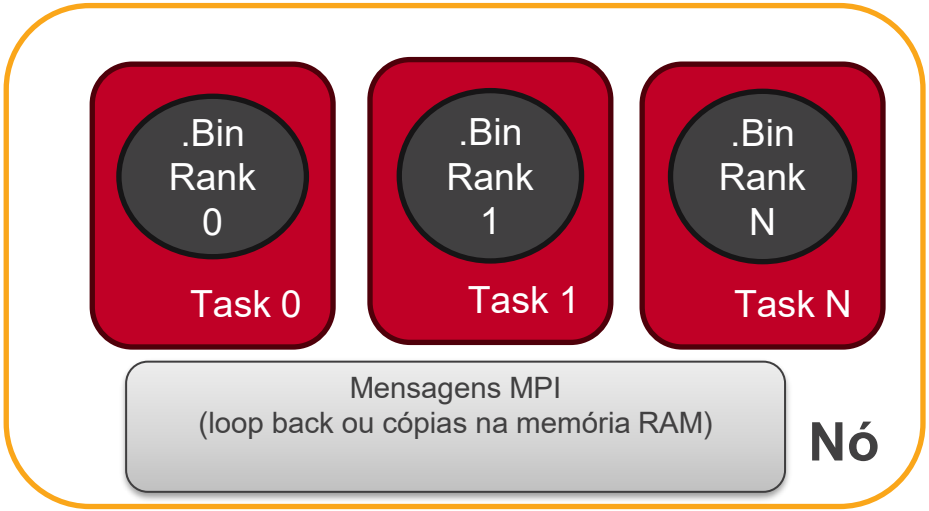
Pedindo Recursos ao SLURM

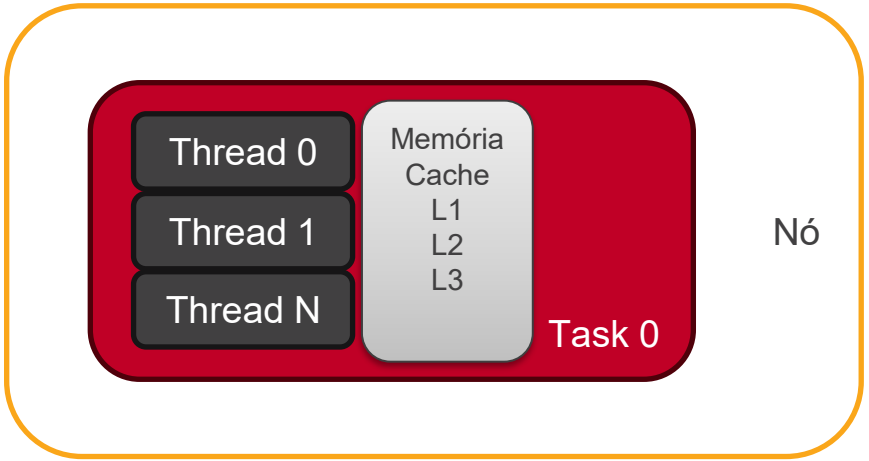
```
#!/bin/bash
#SBATCH --job-name=mpi_hello
#SBATCH --output=saida_%j.txt
#SBATCH --nodes=2   # 2 nós (2 computadores)
#SBATCH --ntasks=5  # 5 processos (5 task MPI)
#SBATCH --cpus-per-task=1 # 1 thread
#SBATCH --time=00:01:00
#SBATCH --partition=gpu
#SBATCH –mem=2G

mpirun -np $SLURM_NTASKS ./seu_binario
```

O que acontece se fizer isso?

```
#!/bin/bash
#SBATCH --job-name=mpi_hello
#SBATCH --output=saida_%j.txt
#SBATCH --nodes=1   # 1 nó (1 computador)
#SBATCH --ntasks=5  # 5 processos (5 task MPI)
#SBATCH --cpus-per-task=1 # 1 thread
#SBATCH --time=00:01:00
#SBATCH --partition=gpu
#SBATCH –mem=2G

mpirun -np $SLURM_NTASKS ./seu_binario
```

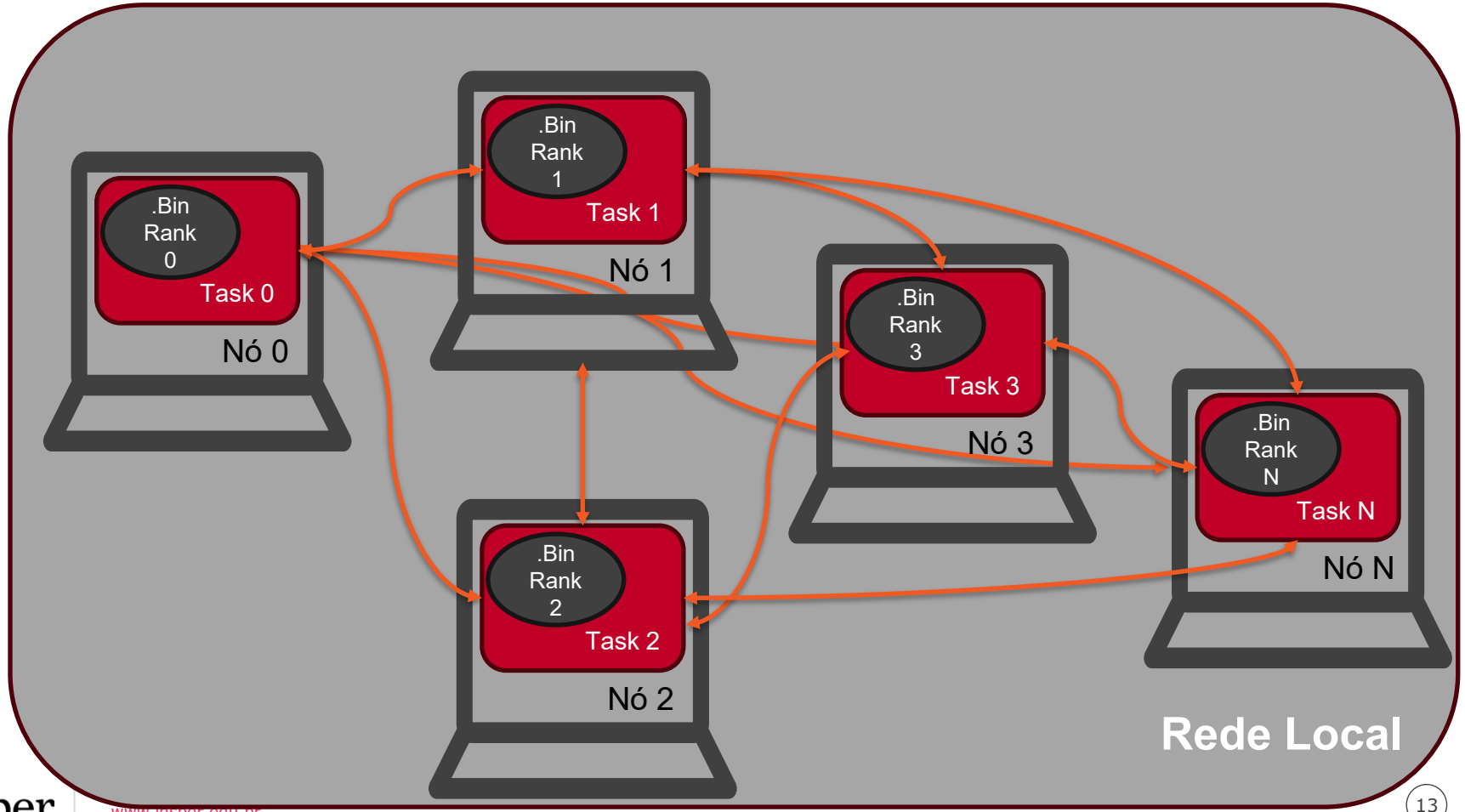
MPI x OpenMP



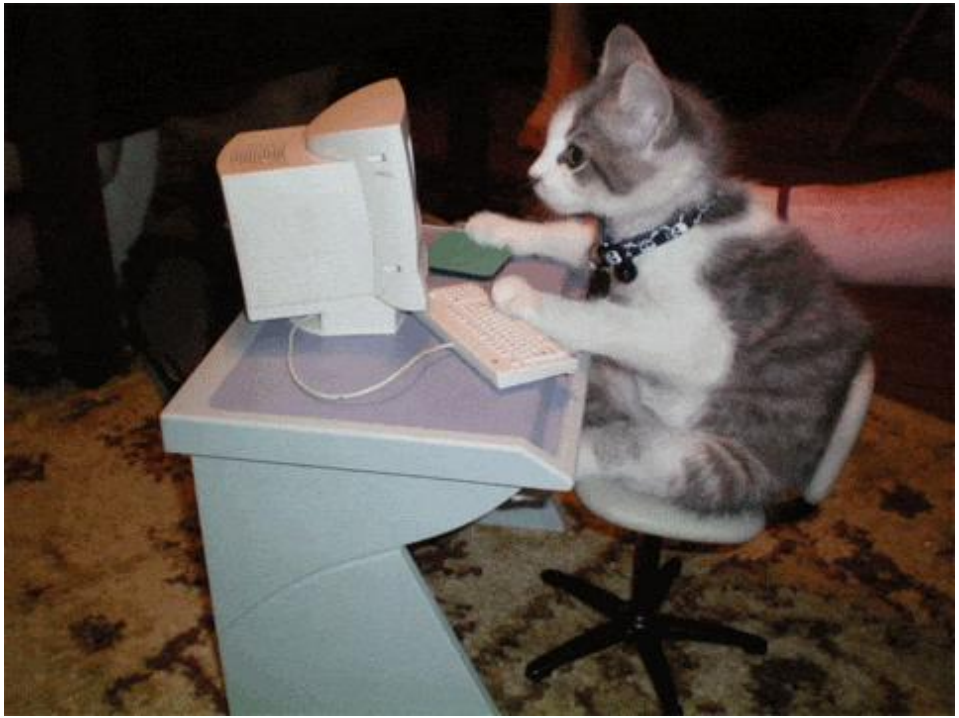


MPI intra-nó: processos precisam mandar cópia da mensagem pelo mecanismo de comunicação (pode ser via loopback ou via região compartilhada criada pelo MPI).

OpenMP: threads acessam diretamente a mesma variável em memória.

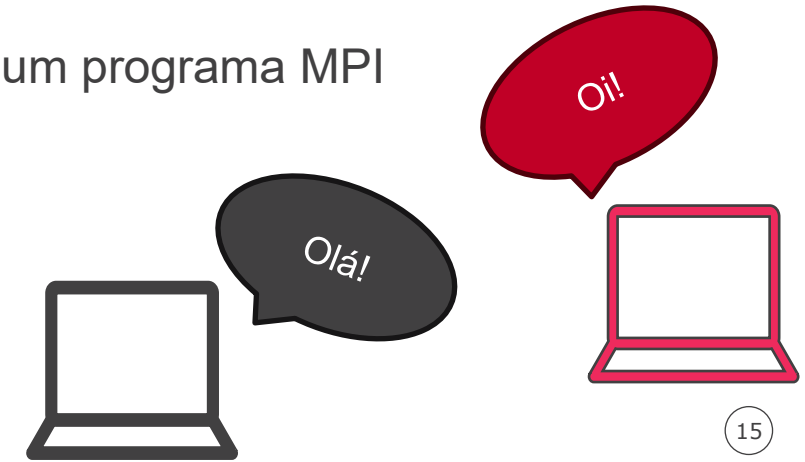

.Bin
Rank
0
Task 0
Nó 0
.Bin
Rank
1
Task 1
Nó 1
.Bin
Rank
2
Task 2
Nó 2
.Bin
Rank
3
Task 3
Nó 3
.Bin
Rank
N
Task N
Nó N
Rede Local

Vamos testar juntos!



Retomando...

- Quais são as principais diferenças entre paralelismo com memória compartilhada e paralelismo com troca de mensagens?
- Como ocorre a comunicação entre diferentes máquinas em ambientes distribuídos usando MPI?
- Dúvidas sobre como fazer a submissão de um programa MPI em um Cluster de HPC?



Obrigada!

Lícia Sales Costa Lima
liciascl@insper.edu.br