Insper

SuperComputação

Aula 17 – Introdução a MPI

2019 – Engenharia

Igor Montagner, Luciano Soares <igorsm1@insper.edu.br>

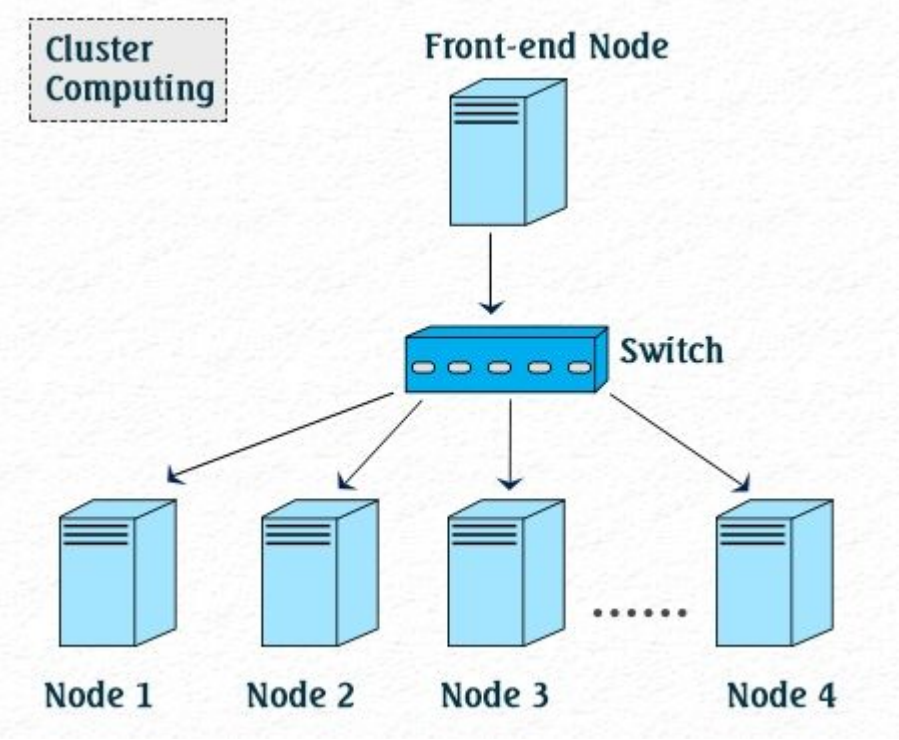
Visão geral do curso

1. Multi core – tarefas independentes
2. Concorrência – sincronização

} Memória compartilhada

3. GPU - paralelismo massivo ← Cópias host <-> device
4. Sistemas com memória distribuída

Cluster Computing — conceitual



- Resolução de um problema combinando recursos de várias máquinas independentes ligadas em rede
- Requer programação específica
- Arquitetura “fixa”: não suporta nós entrando e saíndo do cluster durante a execução de um programa

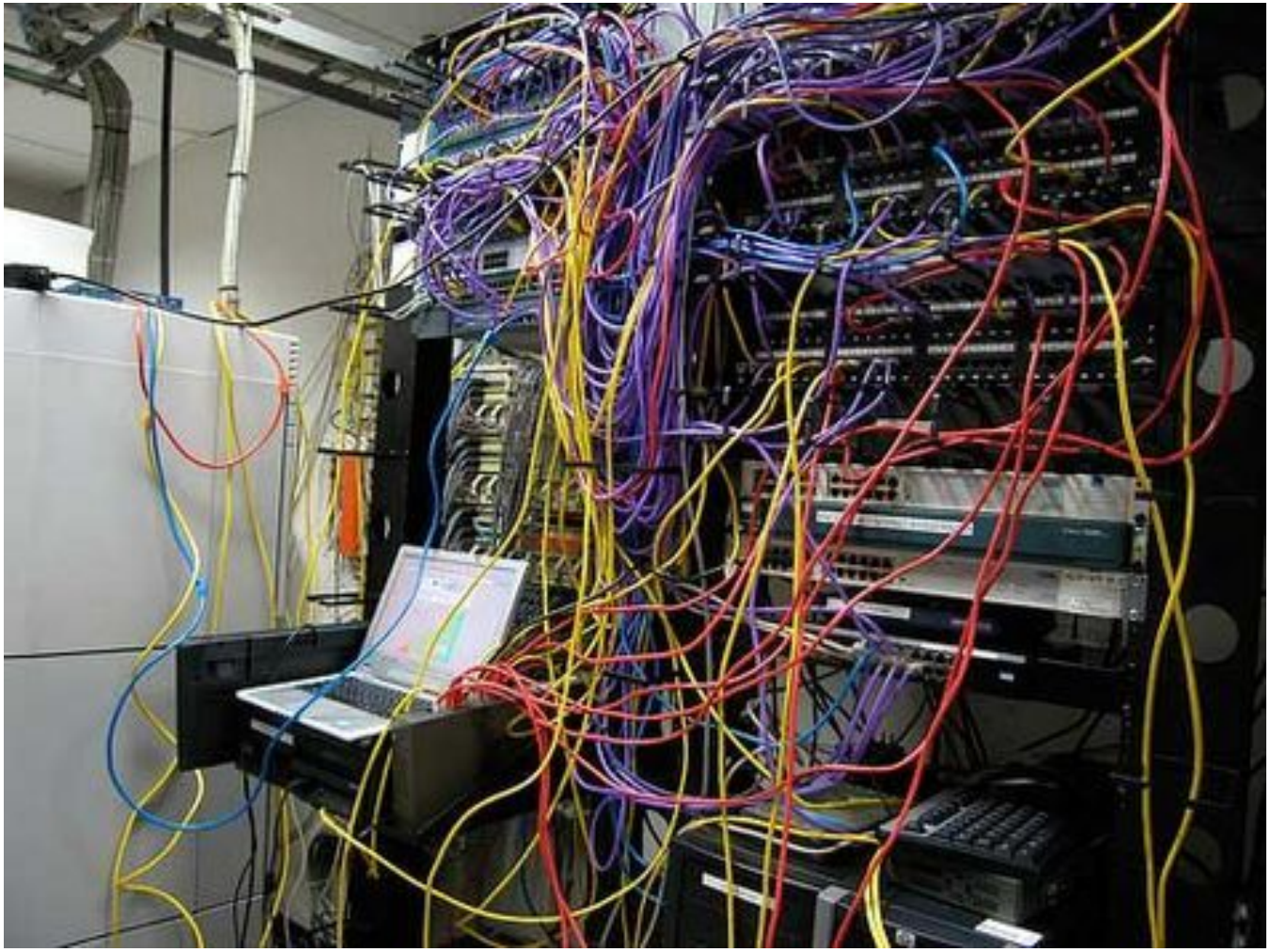
https://techspirited.com/differences-similarities-between-grid-cluster-computing

Cluster Computing — anos 2000

https://icme.hpc.msstate.edu/mediawiki/index.php/Construction_%26_Utilization_of_a_Beowulf_Cluster

Insper

Cluster Computing — atualmente



https://www.pinterest.com/pin/391742867565427821/

Insper

Cluster Computing — de verdade

https://www.aei.mpg.de/96456/20_Computer_Clusters

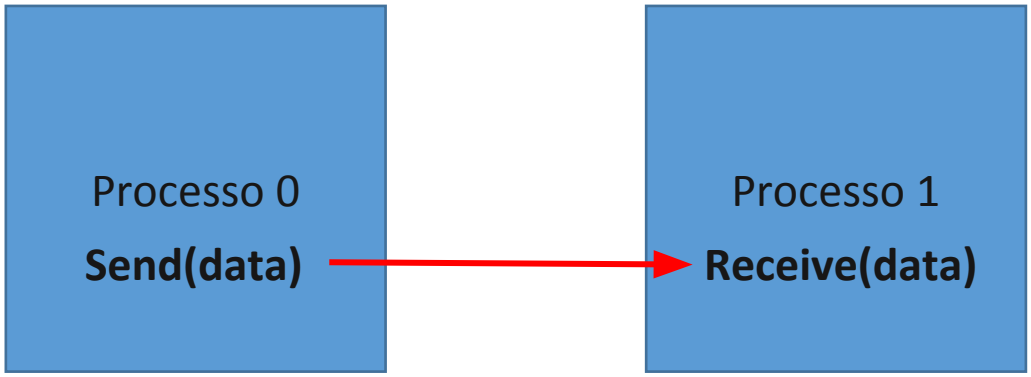
Objetivos

- Disparar vários processos em um mesmo computador
- Sincronizar processos por troca de mensagens
- Planejar troca de mensagens entre processos

Troca de mensagens

- Programa divido em processos na mesma ou em outras máquinas;
- Não existe memória compartilhada, dados são trocados enviando (e recebendo) mensagens;
- Mensagens podem ser síncronas ou não;

Troca de mensagens



1. processo 0 precisa estar na mesma máquina que processo 1?
2. processo 0 compartilha memória com processo 1?
3. é barato enviar a mensagem de 0 para 1?
    a. depende do quê?

MPI (Message Passing Interface)

- Criado pelo fórum MPI organizado em 1992 com participação de: IBM, Intel, TMC, SGI, Convex, Meiko
- Várias versões (MPI-1 –1994, MPI-2 – 1997, MPI3 – 2012)
  - http://www.mpi-forum.orghttp://www.mpi-forum.org
- MPI não é um produto (implementação), é um protocolo

MPI (Message Passing Interface)

Debian/Ubuntu:

apt install libboost-mpi-dev

MPI (Message Passing Interface)

Ambiente (environment):

- Responsável pela inicialização do ambiente MPI
- Obrigatório no início de todo programa. Quando sai de escopo finaliza o MPI

Comunicador (communicator):

- Grupo de processos, cada um com seu id (rank)
- Comunicador padrão é chamado de world e contém todos os processos
- Usado para enviar e receber mensagens entre os processos

MPI (Message Passing Interface)

Só retorna quando os dados foram enviados de fato.

send (recipient, tag, data):

- Envia dados para *recipient*
- *tag* para registrar o tipo da mensagem

Só retorna quando algo for recebido

recv (sender, tag, data_by_ref):

- Recebe dados de *sender*
- Mensagem precisa estar marcada com a mesma *tag*
- Guarda dados em *data_by_ref*

Mensagens são síncronas

Insper

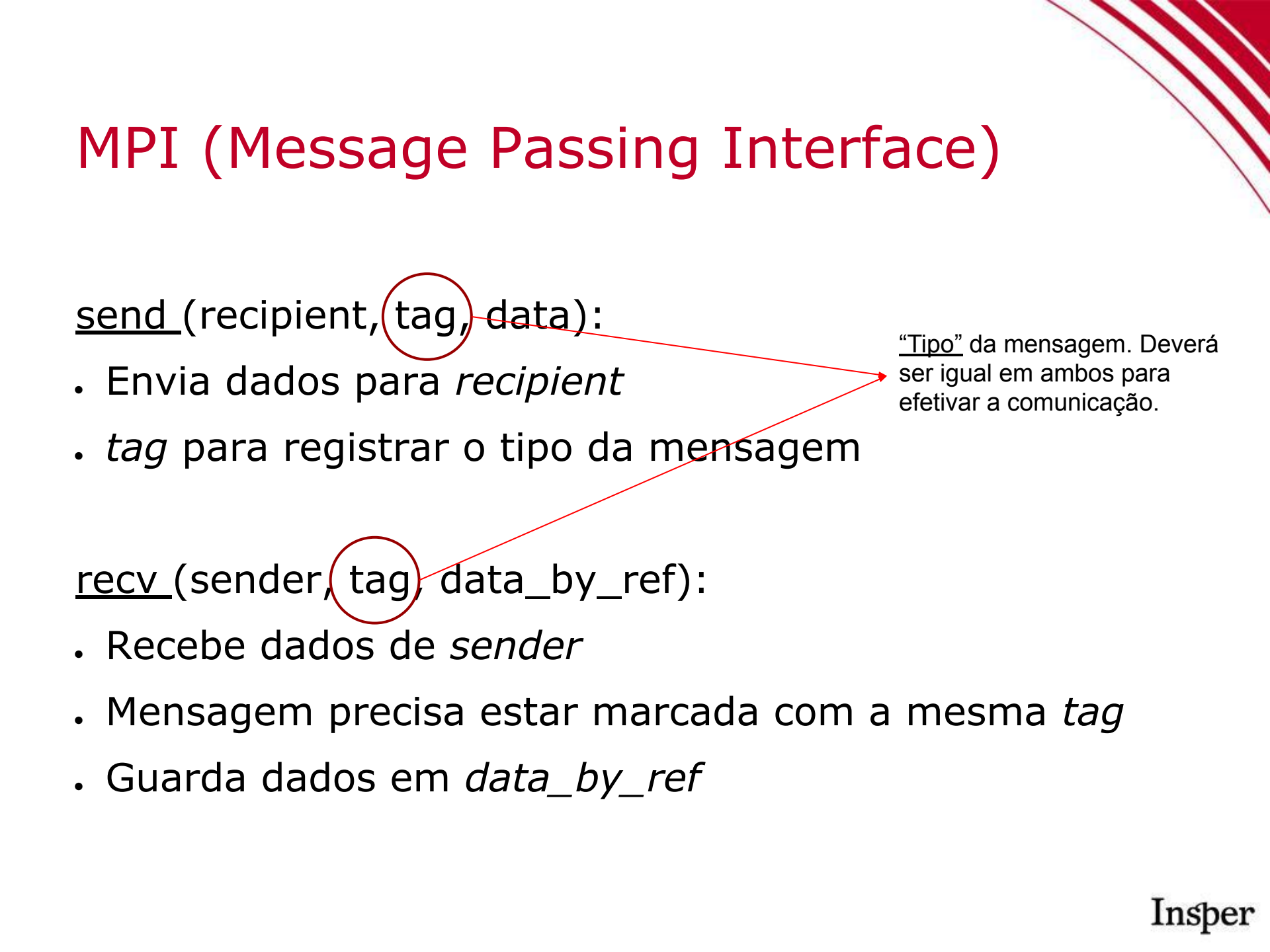
MPI (Message Passing Interface)

send (recipient, tag, data):

- Envia dados para *recipient*
- *tag* para registrar o tipo da mensagem

recv (sender, tag, data_by_ref):

- Recebe dados de *sender*
- Mensagem precisa estar marcada com a mesma *tag*
- Guarda dados em *data_by_ref*



MPI — desafios (aula de hoje)

- Divisão do trabalho: cuidado para não sobrecarregar um processo e deixar outros ociosos
- Deadlocks: toda mensagem enviada deve ser recebida. Se isto não for verdade seu programa pode travar.

Atividade prática

Roteiro de introdução a MPI com C++

Referências

- Livros:
  - Hager, G. ; Wellein, G. **Introduction to High Performance Computing for Scientists and Engineers**. 1ª Ed. CRC Press, 2010.
- Internet:
  - https://www.boost.org/doc/libs/1_67_0/doc/html/mpi/tutorial.html
  -

Insper

www.insper.edu.br