

SuperComputação

Aula 21 – Comunicação coletiva

2018 – Engenharia

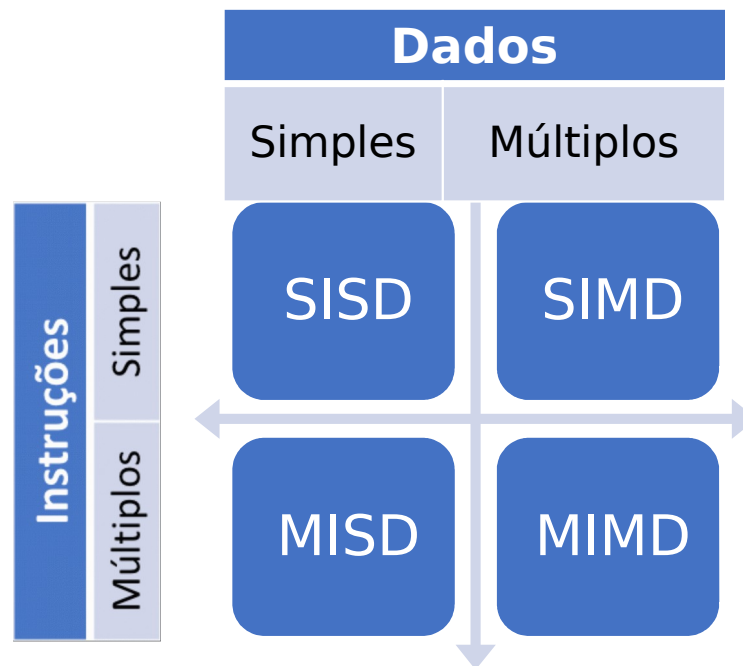
Igor Montagner, Luciano Soares [<igorsm1@insper.edu.br>](mailto:igorsm1@insper.edu.br)

Objetivos

- Comunicação coletiva entre processos
- Usar a API do `boost::mpi` para comunicação coletiva

Taxonomia de Flynn

- Descreve como um sistema paralelo funciona em dois eixos: dados e instruções



Arquiteturas de comunicação

- Mestre/Escravo
 - Mestre distribui tarefas
 - Escravos executam as tarefas e reportam resultados
- Cliente/Servidor
 - Cliente requisita tarefas/dados
 - Servidor distribui tarefas/dados

Mensagens coletivas

- Comunicação entre processos
 - Todos para todos
 - Todos para um
 - Um para todos

Barrier

Todos os processos param até chegar naquele ponto.
Igual às barreiras do OpenMP.

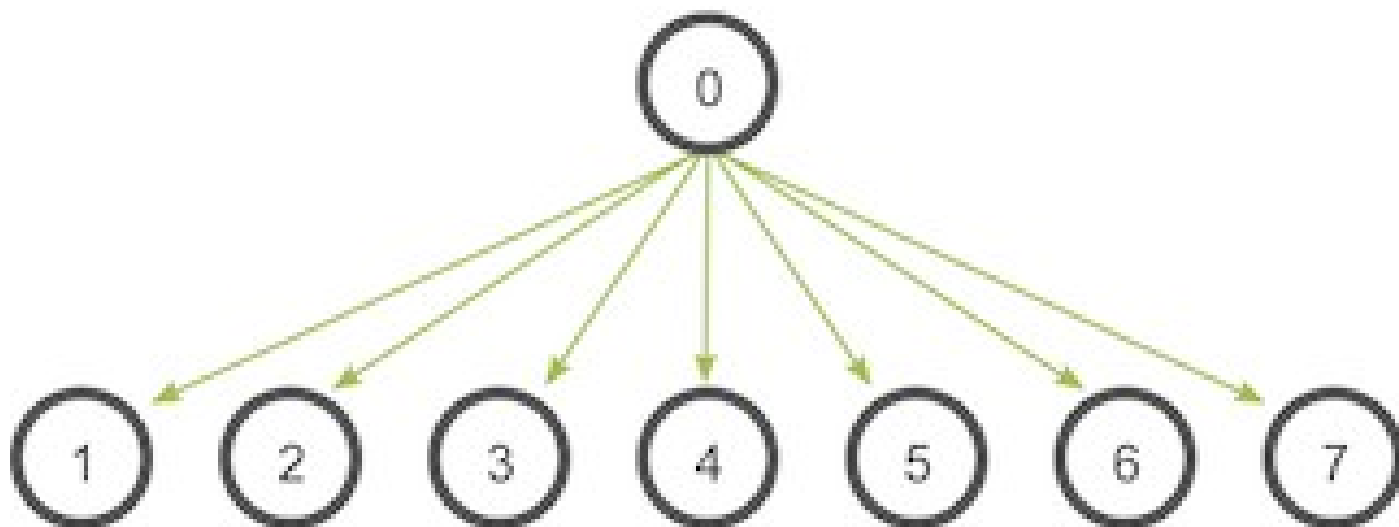
```
communicator::barrier();
```

Cuidado: todos os processos precisam alcançar o código, senão o programa inteiro trava!

Broadcast

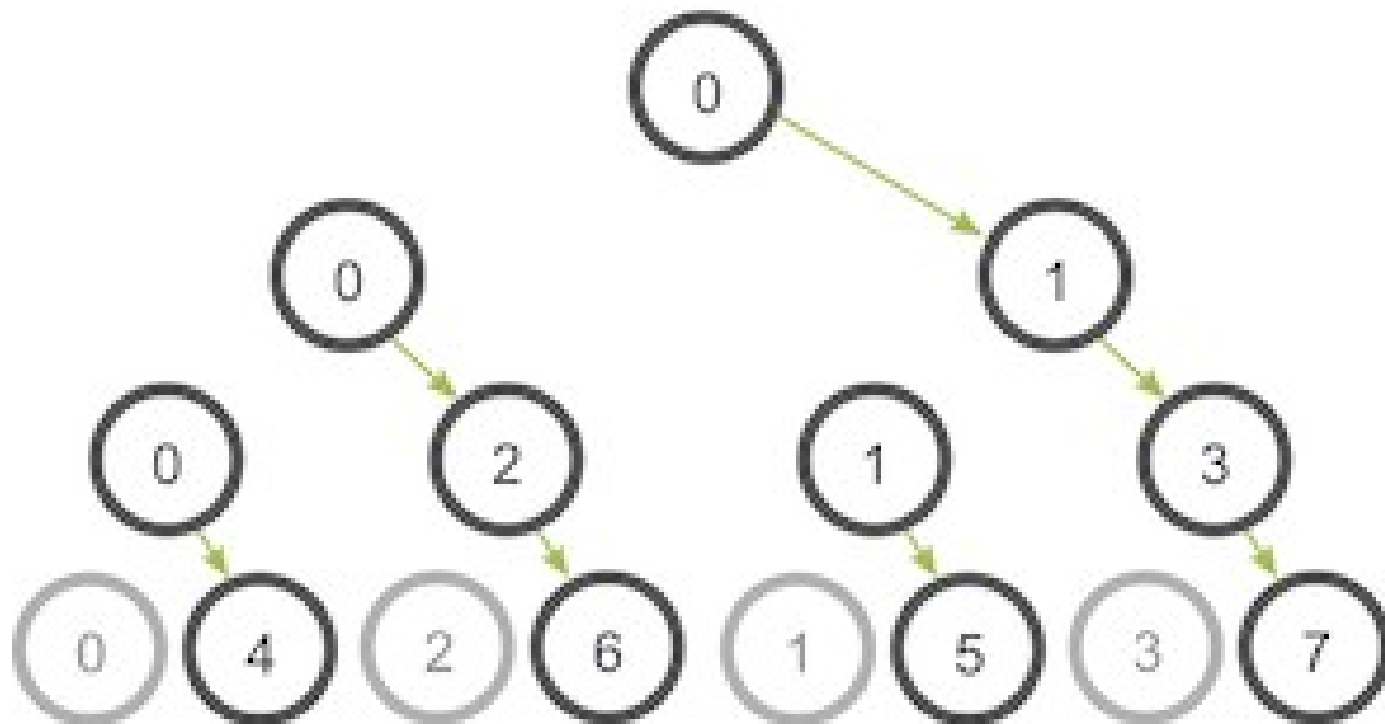
Replica um dado para todos os processos.

```
boost::mpi::broadcast(world, data, root_rank);
```



Broadcast - vantagem

Usa algoritmo inteligente de distribuição

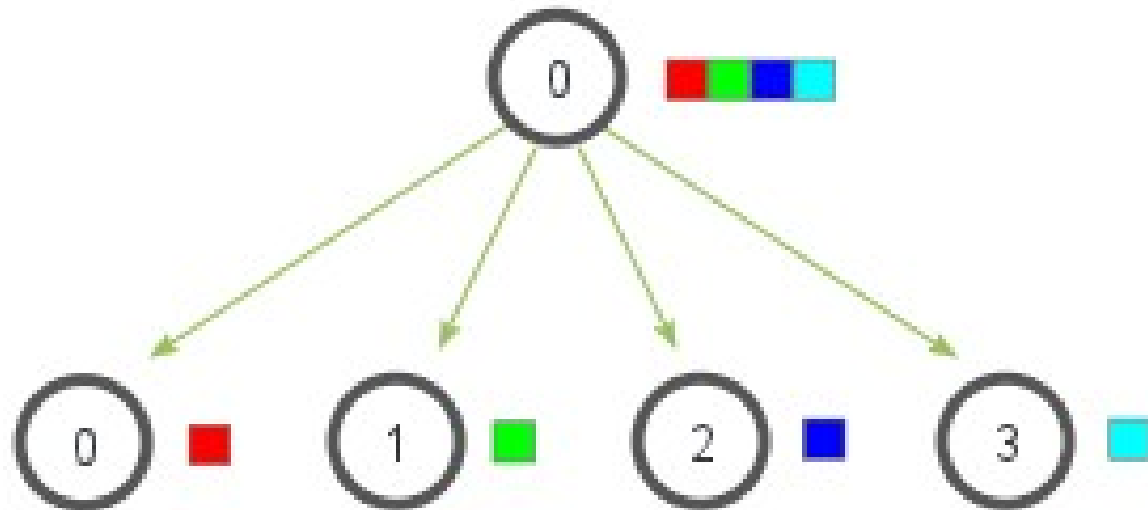


Scatter

Divide dados entre os processos, de modo que cada um pegue um pedaço de igual tamanho.

```
void scatter(const communicator & comm, const  
std::vector< T > & in_values, T &out_value, int root);
```

MPI_Scatter

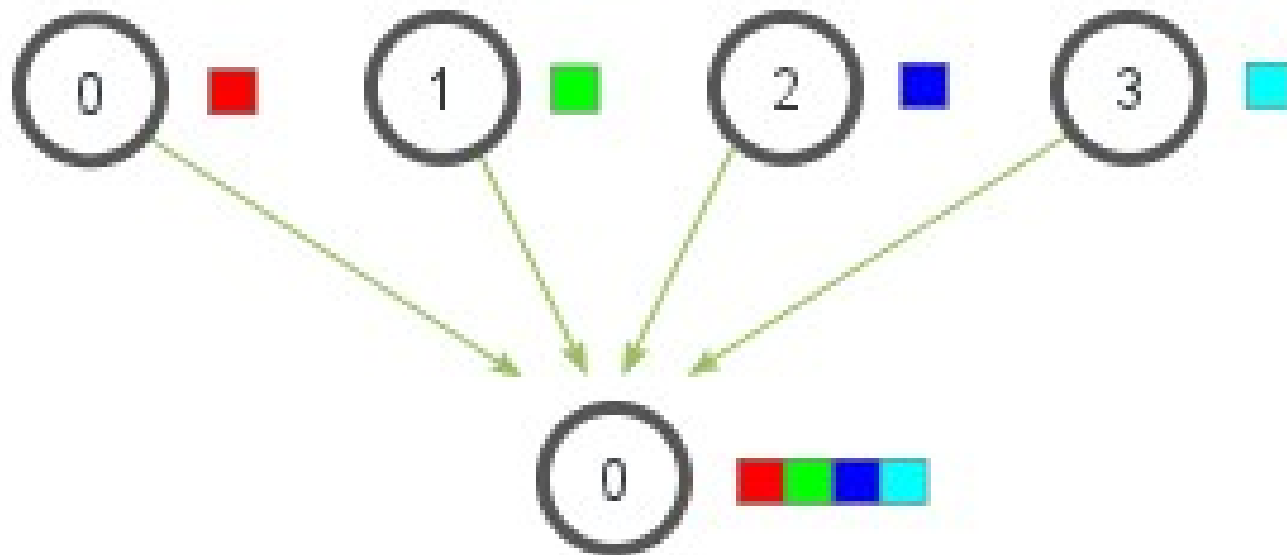


Gather

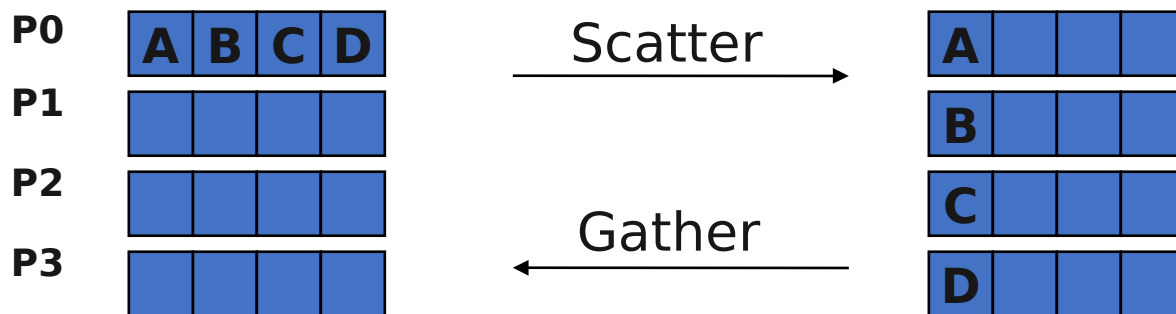
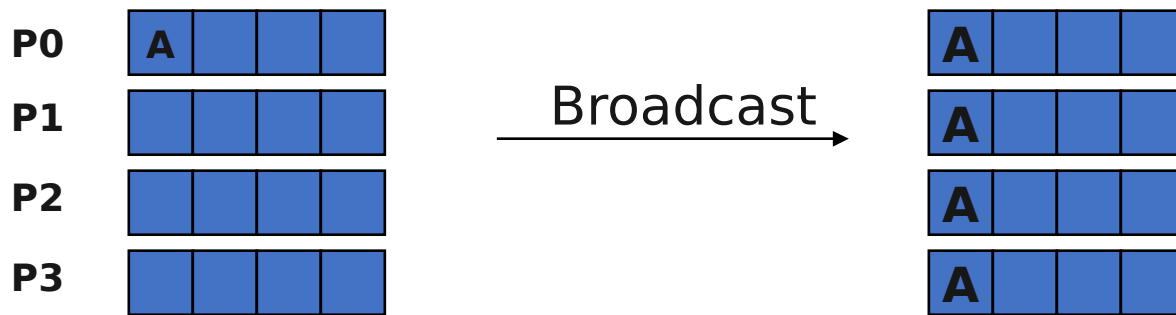
Recolhe informações de vários processos e junta no processo de rank *root*

```
void gather(const communicator & comm, const T & in_value,  
            std::vector< T > & out_values, int root);
```

MPI_Gather



Broadcast vs Scatter vs Gather

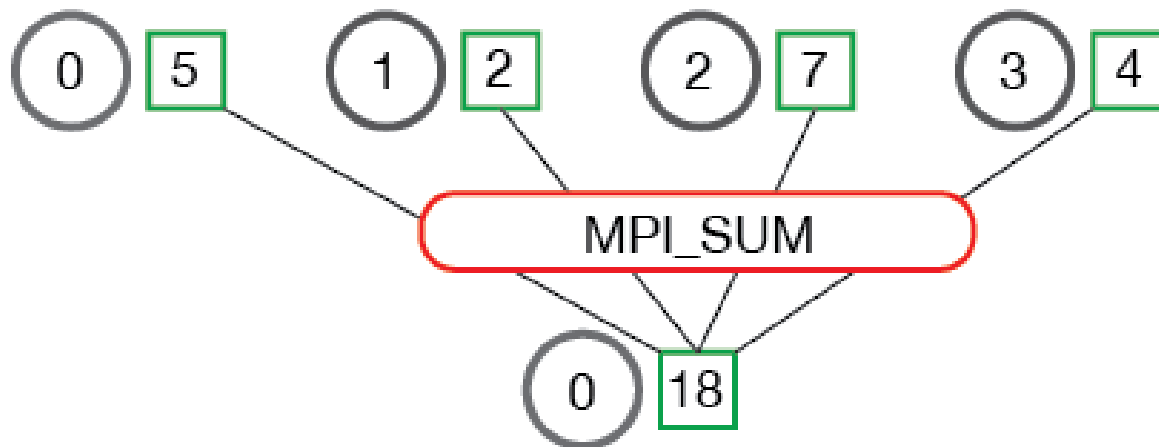


Reduce

Junta informações e aplica função, igual ao OpenMP

```
void reduce(const communicator & comm, const T & in_value,  
            T & out_value, Op op, int root);
```

MPI_Reduce



Reduce - operações

C Constant	Boost.MPI Equivalent
MPI_BAND	<code>bitwise_and</code>
MPI_BOR	<code>bitwise_or</code>
MPI_BXOR	<code>bitwise_xor</code>
MPI LAND	<code>std::logical_and</code>
MPI_LOR	<code>std::logical_or</code>
MPI_LXOR	<code>logical_xor</code>
MPI_MAX	<code>maximum</code>
MPI_MAXLOC	<code>unsupported</code>
MPI_MIN	<code>minimum</code>
MPI_MINLOC	<code>unsupported</code>
MPI_Op_create	<code>used internally by Boost.MPI</code>
MPI_Op_free	<code>used internally by Boost.MPI</code>
MPI_PROD	<code>std::multiplies</code>
MPI_SUM	<code>std::plus</code>

Atividade prática

- Aula 21 no Github
- Atividade 3

Referências

- Livros:
 - STERLING, T.; ANDERSON, M. High Performance Computing: Modern Systems and Practices. 1. ed. Morgan Kaufmann Publishers, 2017.
- Internet:
 - https://www.boost.org/doc/libs/1_67_0/doc/html/mpi/tutorial.html
 - <https://theboostcpplibraries.com/boost.mpi-collective-data-exchange>
 - https://computing.llnl.gov/tutorials/mpi_advanced/DavidCronkSlides.pdf
 - <http://mpitutorial.com/tutorials/>

Insper

www.insper.edu.br