GUIA DE REFERÊNCIA — MPI COM PYTHON (mpi4py)

Este repositório reúne vários exemplos didáticos de programação paralela com MPI + Python (mpi4py). Cada exemplo foca uma técnica de comunicação/coordenação (coletivas, ponto-a-ponto, modos não bloqueantes, envio bufferizado, tipos derivados, grupos/comunicadores, etc.) com analogias simples para ensino.

ONDE ESTÃO OS CÓDIGOS?
Todos os scripts citados abaixo estão na pasta <SOURCE>.

PRÉ-REQUISITOS
- Python 3.8+
- mpi4py
- Uma implementação de MPI (OpenMPI/MPICH no Linux/macOS; MS-MPI/MPICH no Windows)

Instalação do mpi4py:
    pip install mpi4py

Dica: em Linux/macOS use mpiexec/mpirun. Em Windows, prefira mpiexec.

COMO EXECUTAR OS EXEMPLOS
    mpiexec -n <NUM_PROC> python <SOURCE>/<arquivo>.py
    # ou
    mpiexec -n <NUM_PROC> python3 <SOURCE>/<arquivo>.py

Alguns exemplos exigem número de processos potência de 2 (2, 4, 8, 16, …).

---

1) COLETIVAS DE REDUÇÃO E AGREGAÇÃO

- mpi_pi_criancas.py — Allreduce (soma) para aproximar π
  Cada processo integra um pedaço e soma global com allreduce.
  Técnicas: COMM_WORLD.allreduce, particionamento por stride.
  Rodar: mpiexec -n 4 python <SOURCE>/mpi_pi_criancas.py

- mpi_media_desvio_allreduce.py — média e desvio padrão globais
  Primeiro allreduce para a soma global (média), depois outro allreduce para a soma das diferenças² (desvio padrão).
  Técnicas: duas passagens de allreduce (SUM).

- mpi_medias_allgather.py — troca de médias locais com Allgather
  Cada processo calcula sua média local e todos trocam com allgather para computar a média da turma.
  Técnicas: COMM_WORLD.allgather.

- mpi_gather_criancas.py — coleta de vetores no líder
  O líder junta blocos iguais vindos de todos.
  Técnicas: COMM_WORLD.gather (objeto Python), concatenação no líder.

---

2) PONTO-A-PONTO (BLOQUEANTE, NÃO BLOQUEANTE, SÍNCRONO, BUFFERIZADO)

- mpi_aleatorio_criancas.py — Send/Recv + descoberta do tamanho recebido
  O rank 0 envia uma quantidade aleatória de inteiros; o rank 1 recebe com buffer maior e usa Status.Get_count para saber quantos chegaram (ANY_SOURCE/ANY_TAG).
  Técnicas: Send/Recv, Status.Get_count, MPI.ANY_SOURCE, MPI.ANY_TAG.

- mpi_isend_criancas.py — não bloqueante com Isend/Irecv
  Trocas em recursive doubling; posta Isend/Irecv, depois Waitall.
  Técnicas: Isend/Irecv, Request.Waitall, padrão recursive doubling (par em rank ± i).

- mpi_sincrona_criancas.py — envio síncrono com Ssend
  Em cada rodada, metade envia primeiro e a outra metade recebe primeiro, evitando deadlock.
  Técnicas: Ssend/Recv, alternância de ordem, recursive doubling.

- mpi_bsend_criancas.py — envio bufferizado com Bsend
  Anexa um buffer (Attach_buffer), calcula tamanho de mensagem e faz trocas recursive doubling guardando o máximo elemento a elemento.
  Técnicas: Attach_buffer/Detach_buffer, Bsend/Recv, cálculo de BSEND_OVERHEAD.

---

3) TIPOS DERIVADOS (DADOS NÃO CONTÍGUOS / STRUCTS)

- mpi_bcast_coluna_vector_fix.py — broadcast de uma coluna da matriz
  No root, a coluna é não contígua em memória. Criamos um hvector e ancoramos no endereço absoluto do 1º elemento com Create_struct + MPI.BOTTOM. Demais processos recebem em vetor contíguo.
  Técnicas: Datatype.Create_hvector, Datatype.Create_struct, MPI.BOTTOM, Get_address.

- mpi_particulas_criancas.py — struct “Partícula” + broadcast
  Definimos numpy.dtype e um tipo MPI equivalente com Create_struct; o root preenche e difunde para todos.
  Técnicas: Datatype.Create_struct, Bcast, compatibilização numpy.dtype ↔ tipo MPI.

- mpi_particulas_grafico.py — broadcast + transformação local + gather + gráfico
  Após o broadcast, cada rank aplica um deslocamento e o root faz Gather para plotar (x × y) por processo.
  Técnicas: Bcast, Gather, visualização com matplotlib.

---

4) “QUEM TEM O MÁXIMO E ONDE?” (MAXLOC EMULADO)

- mpi_maxloc_criancas.py — emulação de MPI_MAXLOC
  Para cada posição, faz Allreduce(MAX) no valor e, dentre quem empatou, Reduce(MIN) do rank (quem não empatou manda “∞”).
  Técnicas: Allreduce(MAX) + Reduce(MIN) sobre candidatos.

---

5) BARREIRAS E DIFUSÃO SIMPLES

- mpi_barreira_criancas.py — barreira (“portão do parquinho”)
  Ninguém avança até todos chegarem; o rank 0 “atrasado” espera teclado/tempo.
  Técnicas: Barrier, uso de Wtime/Wtick (opcional).

- mpi_broadcast_criancas.py — broadcast de um valor escalar
  O líder lê um inteiro e difunde para todos (versão simples via bcast de objeto Python).
  Técnicas: COMM_WORLD.bcast (objeto), alternativa com Bcast + numpy.

---

6) GRUPOS E COMUNICADORES

- mpi_grupos_criancas.py — união de grupos arbitrários e novo comunicador
  Criamos dois grupos “A” e “B” e depois a união (em mpi4py: MPI.Group.Union(g1, g2)). O novo comunicador contém os membros da união; cada participante ganha um novo rank.
  Técnicas: Get_group, Group.Incl, MPI.Group.Union, Comm.Create, Group.Get_rank.

- mpi_grupos_meia_turma.py — divisão em metades fixas (8 processos)
  Metade baixa [0..3] e metade alta [4..7]; cada grupo tem seu comunicador e faz Allreduce(SUM) dos ranks antigos.
  Técnicas: Group.Incl, Comm.Create, allreduce intra-comunicador.

- mpi_grupos_meia_turma_flex.py / mpi_grupos_meia_turma_stats.py — divisão em metades para qualquer N ≥ 2
  Particionamento em “Baixo” e “Cima” (o de cima pode ter 1 a mais se N ímpar), criação do comunicador e estatísticas do time: membros, tamanho, soma e média.
  Técnicas: Get_group, Group.Incl, Comm.Create, gather de membros, allreduce da soma, cálculo de média.

---

7) “CARTÃO DE IDENTIDADE” DO JOB MPI

- mpi_funcoes_criancas.py — versão do MPI, rank/size, nome da máquina, timers
  Mostra versão/subversão, rank/size, hostname e mede tempo com Wtime (precisão Wtick).
  Técnicas: Get_version, Get_processor_name, Wtime/Wtick.

- mpi_funcoes_criancas_gather.py — resumo agregado no líder
  Cada processo envia seu “cartão” e o líder imprime tabela ordenada por rank.
  Técnicas: gather de dicionários/objetos Python e formatação no líder.

PADRÕES QUE APARECEM COM FREQUÊNCIA
- Recursive Doubling (trocas em distâncias 1, 2, 4, …): mpi_isend_criancas.py, mpi_sincrona_criancas.py,
  mpi_bsend_criancas.py.
- Tipos derivados:
  * Strided (não contíguo): Create_hvector + Create_struct + MPI.BOTTOM — ver

mpi_bcast_coluna_vector_fix.py.
  * Structs (campos mistos): Create_struct compatível com numpy.dtype — ver
mpi_particulas_criancas.py.
- Coletivas essenciais: bcast/Bcast, gather/allgather, reduce/allreduce, Barrier.
- Status e contagem recebida: Status.Get_count para saber quantos elementos chegaram — ver
  mpi_aleatorio_criancas.py.

DICAS E SOLUÇÕES DE PROBLEMAS
- Potência de 2: alguns exemplos assumem -n como 2, 4, 8, …
- Bufferizado (Bsend): anexe um buffer suficientemente grande (Attach_buffer) e lembre de
Detach_buffer();
  em mpi4py, Detach_buffer() retorna só o buffer.
- Não contíguo (vector/hvector): o lado root pode usar endereços absolutos com MPI.BOTTOM para
evitar
  erros de “ndarray não contíguo”.
- Nomes de métodos: em mpi4py use Get_rank() (e não Get_Rank()), etc.
- ANY_SOURCE/ANY_TAG: use MPI.ANY_SOURCE/MPI.ANY_TAG com Recv quando quiser flexibilidade (e cheque
Status).

ORGANIZAÇÃO
<SOURCE>/
  mpi_pi_criancas.py
  mpi_media_desvio_allreduce.py
  mpi_medias_allgather.py
  mpi_gather_criancas.py
  mpi_aleatorio_criancas.py
  mpi_isend_criancas.py
  mpi_sincrona_criancas.py
  mpi_bsend_criancas.py
  mpi_bcast_coluna_vector_fix.py
  mpi_particulas_criancas.py
  mpi_particulas_grafico.py
  mpi_maxloc_criancas.py
  mpi_barreira_criancas.py
  mpi_broadcast_criancas.py
  mpi_funcoes_criancas.py
  mpi_funcoes_criancas_gather.py
  mpi_grupos_criancas.py
  mpi_grupos_meia_turma.py
  mpi_grupos_meia_turma_flex.py
  mpi_grupos_meia_turma_stats.py

Este guia foi montado a partir de exemplos e explicações didáticas produzidas durante nossa
conversa, consolidando técnicas clássicas de MPI no contexto de mpi4py.