INE5645 — Programação Paralela e Distribuída

Trabalho II: Programação Distribuída

Pedro Santi Binotto (20200634)
Gabriel Lemos da Silva (18200628)
Bruno Alexandre Schmitt Costa (17105532)

Introdução do tema

Modelo de memória compartilhada entre processos

Escopo

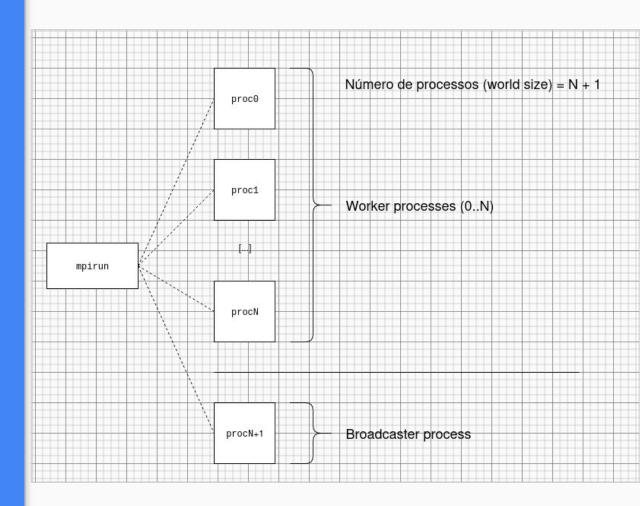
O sistema desenvolvido implementa uma rede de processos que mantém um espaço de memória compartilhada. A comunicação entre estes processos acontece através de uma API que oferece operações de leitura e escrita, através da abstração de "blocos" de memória:

- Cada processo é responsável por "gerenciar" uma parcela dos blocos totais de memória;
- Cada processo acessa os blocos remotos através da API mantida pelo processo "responsável" pelo bloco;
- > Os processos mantém *caches* locais representando os blocos remotos;

Seja N informado pelo usuário e $(N \ge 0)$, o número total de processos instanciados é dado por (N + 1).

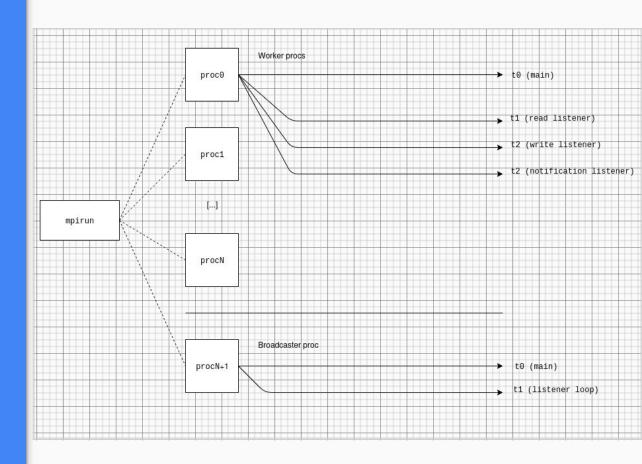
Os processos (0...N) são workers e (N+1) é o broadcaster dedicado.

- Processos "worker" mantém e gerenciam blocos da memória;
- Processos "worker" realizam computações sobre estes blocos;
- O "broadcaster" repassa notificações para os demais proessos.

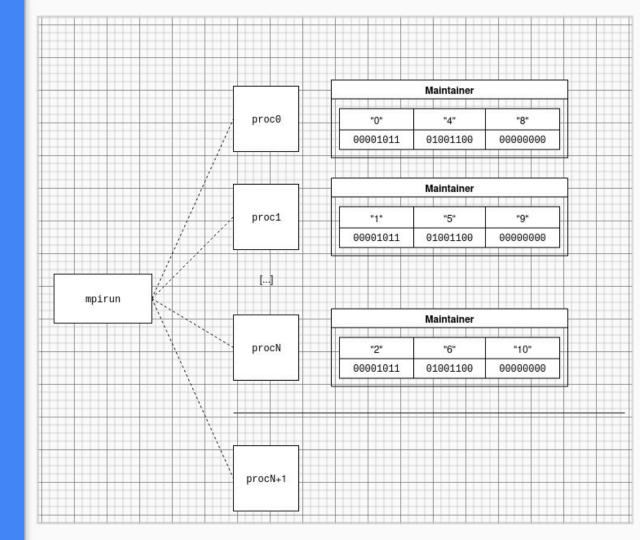


Cada processo *worker* apresenta quatro threads paralelas:

- Thread t0 (main) é a thread principal do processo; realiza leitura, computação e escrita sobre os blocos de memória;
- Thread t1 (read) é a thread que realiza o listener loop para atender requisições de leitura sobre os blocos gerenciados;
- Thread t2 (write) é a thread que realiza o listener loop para atender requisições de escrita;
- Thread t3 (notification) é a thread que receberá notificações so processo broadcaster;

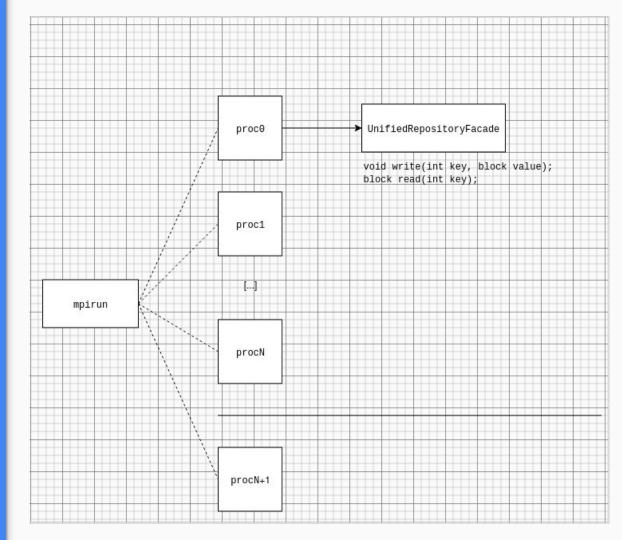


Ao iniciar a execução dos processos, os K blocos de memória serão distribuídos entre os N processos worker (round-robin, 0 . . N), resultando em um "mapa de memória" conhecido por todos os processos, que associará os endereços de cada bloco com seu processo gerenciador.

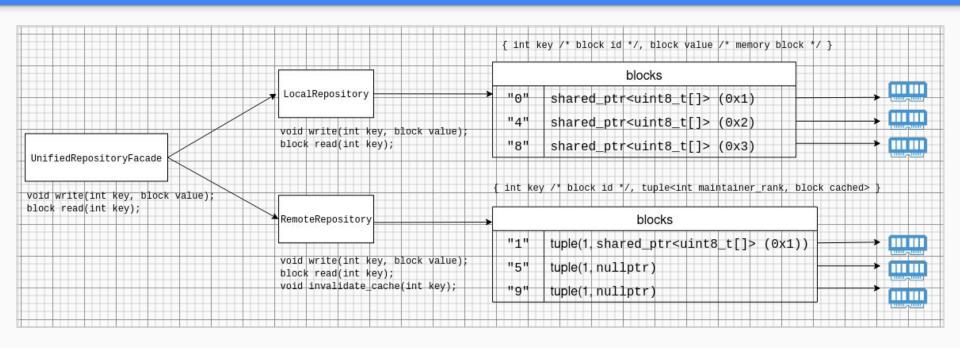


Ao receber o mapa de memória, cada processo *worker* inicializará uma cada de acesso de dados através da class UnifiedRepositoryFacade;

A abstração de acesso de dados permitirá à thread principal do processo realizar operações de leitura e escrita sobre os blocos de memória de forma transparente, sem que seja necessário implementar rotinas separadas para acessos remotos e locais.



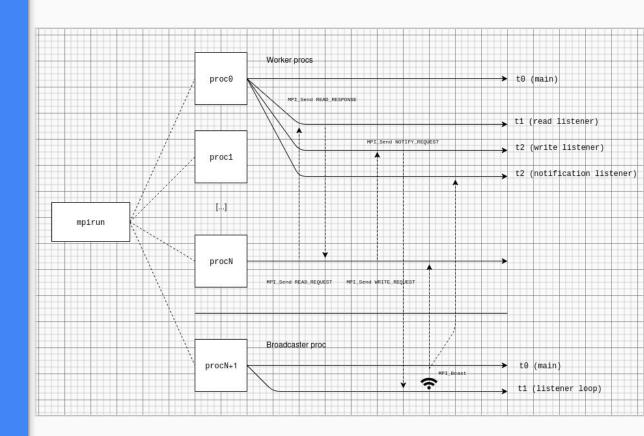
A nivel de implementação, a classe UnifiedRepositoryFacade oferece uma interface simples para acessar as estruturas de manutenção (LocalRepository) de dados local assim como de leitura/escreita remota (RemoteRepository)



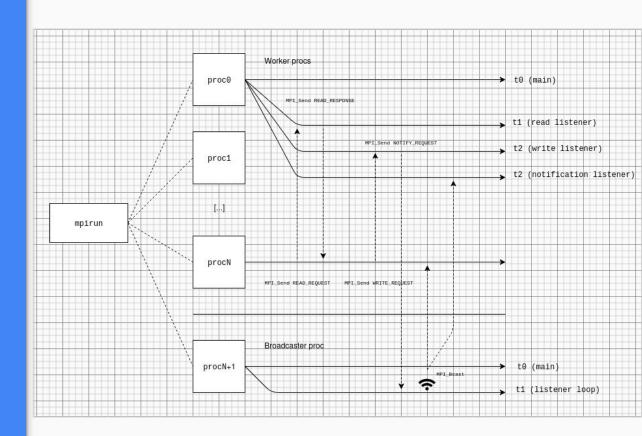
O modelo de "life-cycle" básico do processo então é o seguinte:

- A thread principal do processo worker realizará operações de leitura e escrita sobre os blocos;
- As threads auxiliares, por sua vez, executam um loop que aguarda a comunicação de eventos de leitura, escrita, e notificações de atualização vindos de outros processos;
- O processo broadcaster

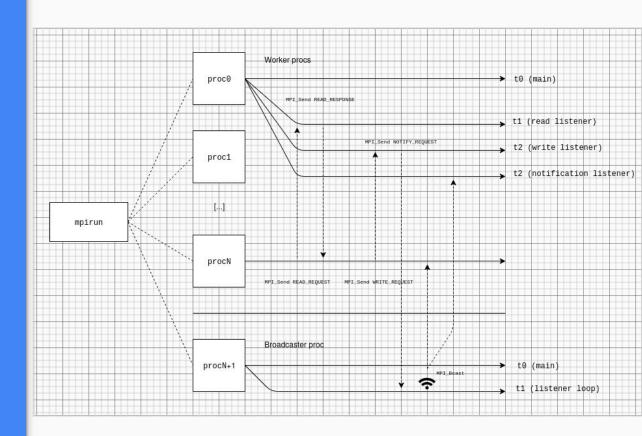
 aguardará o evento de
 atualização de um bloco para
 realizar o broadcast do evento.



- Quando uma operação de escrita é realizada sobre um bloco (localmente ou por um evento remoto), o processo gerenciador do(s) bloco(s) atualizado(s) enviará uma requisição de notificação para o processo broadcaster, contendo o ID do bloco atualizado, juntamente com o timestamp (UNIX) do momento da atualização;
- A notificação será enviada em broadcast para todos os workers, que invalidarão suas caches locais referentes àquele bloco, consultando diretamente do processo remoto na próxima operação de leitura.



- A nível de processo, as operações de leitura e escrita sobre os blocos são protegidas de condições de corrida através do uso de mutexes (read/write) que permitem leituras concorrentes mas não escritas/invalidações;
- O evento de notificação enviado em **broadcast** contém também o *timestamp* da computação do bloco, para que, em um cenário "concreto", possam ser realizadas invalidações de produções que fossem realizadas utilizando valores obsoletos, etc.



Parametrização e detalhes de implementação

O programa permite a parametrização de atributos para produzir comportamentos distintos sob circunstâncias diversas. Através no Makefile, podem ser parametrizados dos valores:

- Tamanho do bloco (em bytes);
- Número de blocos (total, será distribuído entre os processos);
- Número de workers;

Observabilidade

Para que seja possível observar e analisar o comportamento dos processos, o programa oferece diferentes níveis de log, assim como saída para stdout e escrita em arquivos de log (separados por rank do processo).

Observabilidade

Também é possível executar o programa no modo debug, que manterá uma instância do gdb (GNU Debugger) para cada processo, possibilitando o monitoramento mais precisa da execução do código.

Arquivos .log

Ao executar a aplicação, serão produzidos no diretório log/arquivos que registram a atividade dos processos, permitindo que seja observado o que foi executado por cada instância, em ordem cronológica.

Arquivos . log

Em cada arquivo, encontra-se o que o processo submeteu ao stdout durante a execução.

Dependendo do nível de log configurado, isso pode incluir apenas as transaçõs (recebeu dados para tal bloco, enviou requisição de leitura, etc.) como também um *dump* da memória representando o estado interno do processo a cada iteração.

```
[BLOCK SIZE, NUM BLOCKS] não informados; usando parâmetros padrão:
[8, 4]
Process assigned world rank 0 successfully parsed program args
Process assigned world rank 0 successfully validated program args
[1752262034] (@proc 0) Started as worker process
[New Thread 0x7fffee1ff6c0 (LWP 124222)]
[1752262034] (@proc 0) Notification listener thread started
[1752262034] (@proc 0) Notification listener probing...
[New Thread 0x7fffed9fe6c0 (LWP 124224)]
[1752262034] (@proc 0) Write thread started
[1752262034] (@proc 0) Write listener probing...
[New Thread 0x7fffed1fd6c0 (LWP 124229)]
[1752262034] (@proc 0) Read thread started
[1752262034] (@proc 0) Started helper threads
[1752262034] (@proc 0) Read listener probing...
[1752262034] (@proc 0) Hello, World! from processor bridge, rank 0 out of 5
processors
[1752262034] (@proc 0) WRITE operation to block 2 called at remote repository
level
[1752262034] (@proc 0) Received method call to execute remote WRITE operation to
00000000 on process ID 2
[1752262034] (@proc 0) Encoding write message from of key: 2, value: 01100111
buffer 00000010 00000000 00000000 00000000 01100111 11000110 01101001 01110011
01010001 11111111 01001010 00000000
[1752262034] (@proc 0) Sending WRITE request of key: 2, value: 01100111 11000110
```

Process assigned world rank 0 successfully initialized MPI

Modo debug

O modo debug permite também analisar a execução individual de cada processo em tempo real, através do GNU Debuger. Exemplo ao lado, com N = 4 (4 workers + 1 broadcaster).

```
[1752262294] (@proc 4) Successfully submitted broadcast message with total conte
        52262295] (@proc 1) WRITE operation to block 1 called at local repository le
                                                                                                              0000 00000000 00000000 00000000
   [1752262295] (@proc 1) Sending out update notification request for block 1...
[1752262295] (@proc 1) Encoding notification message from of key: 1, timestamp:
                                                                                                               1752262295] (@proc 4) Notification broadcaster probing...
1752262295] (@proc 4) Detected NOTIFIATON operation request at `listener` level
    1752262295] (@proc 4) Processing NOTIFICATION operation request at `handler` 1
                                                                                                             vel...
[1752262295] (@proc_4) Successfully interpreted at `handler` level NOTIFICATION
    [1752262295] (@proc 1) Completed WRITE request from process of ID {0} successful
                                                                                                            000000000 00000000
                                                                                                             [1752262295] (@proc 1) Decoding notification message from bytearray buffer: 0000
    00000 00000000 00000000
    000000 00000000 00000000
                                                                                                               1752262295] (@proc 4) Allocated memory for buffer
    1752262295] (@proc 1) Allocated memory for buffer
                                                                                                               1752262295] (@proc 4) Copied buffer to memory
    [1752252295] (@proc 1) Copied buffer to memory
[1752252295] (@proc 1) Constructed object; key 2, timestamp 1752252248 from raw-
message buffer 00000010 00000000 00000000 001000000 01101000 01100110 0111001 01
                                                                                                            [1752262295] (@proc 4) Constructed object; key 2, timestamp 1752262248 from raw
message buffer 00000010 00000000 00000000 01101000 01100110 01110001 01
                                                                                                              [1752262295] (@proc 4) Successfully submitted broadcast message with total conte
    [1752262295] (@proc 1) Requesting cache invalidation for block 2
[1752262295] (@proc 1) Erasing local cache for block 2
[1752262295] (@proc 1) Notification listener probing...
                                                                                                             nts 00000010 00000000 00000000 00000000 01101000 01100110 01110001 01101000 0000
                                                                                                             0000 00000000 00000000 00000000
                                                                                                            × - □
   1752262295] (@proc 2) WRITE operation to block 2 called at local repository lev
                                                                                                          [175256295] (@proc 3) Detected READ operation request at `listener` level
[175256295] (@proc 3) Processing READ operation request at `handler` level...
[1752562295] (@proc 3) Successfully interpreted at 'handler' level that targeted
  block for READ operation is 3
                                                                                                            1752262295] (@proc 3) READ operation to block 3 called at local repository leve
   1752262295] (@proc 2) Completed WRITE request from process of ID {0} successful
                                                                                                          [1752262295] (@proc 3) Completed READ request from process of ID 2 successfully.
                                                                                                            Sending out response for black 3...
  0 00000000 00000000
 00000 00000000 00000000
                                                                                                          000000 000000000 00000000
[1752282295] (@proc 2) Allocated memory for buffer
[1752282295] (@proc 2) Copied buffer to memory
[1752282295] (@proc 2) Constructed object; key 2, timestamp 1752252248 from raw
message buffer 00000010 00000000 00000000 01101000 01100110 01110001 01
                                                                                                          [1752262295] (@proc 3) Allocated memory for buffer
                                                                                                          [173225259] (@proc 3) Constructed object; key 2, timestamp 1752252248 from raw
[1752252295] (@proc 3) Constructed object; key 2, timestamp 1752252248 from raw
   message buffer 00000010 00000000 00000000 00000000 01101000 01100110 01110001 01
  1752262295] (@proc 2) Skipping update notification for locally-maintained block
                                                                                                           [1752262295] (@proc 3) Requesting cache invalidation for block 2
[1752262295] (@proc 3) Erasing local cache for block 2
[1752262295] (@proc 3) Notification listener probing...
  1752262295] (@proc 2) Notification listener probing...

    umax-sessionizer ~/projects/priage/pec/frontena

  pedro@bridge → project (IP main)
 00 01010101 00000000 00000000
                                                                                                          pedro@bridge → project (!P main) make run
00000000 00000000 over MPI
edro@bridge - project (!V main)
[1752262295] (@proc 0) Decoding notification message from bytearray buffer: 0000
                                                                                                          pedro@bridge → project (10 main)
pedro@bridge → project (!P main)
 00000 00000000 00000000
[175225225] (Genoc 0) Allocated memory for buffer [175225225] (Genoc 0) Copied buffer to memory [17522525] (Genoc 0) Copied buffer to memory [175225225] (Genoc 0) Copied buffer to memory [1752252525] 
  [1752262295] (@proc 0) Requesting cache invalidation for block 2
  [1752262295] (@proc 0) Erasing local cache for block 2
   1752262295] (@proc 0) Notification listener probing...
```

Obrigado!

Por: hmitt Cos

Bruno Alexandre Schmitt Costa [17105532] Gabriel Lemos da Silva [18200628] Pedro Santi Binotto [20200634]