CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 31: Comunicação Unilateral - Otimizações

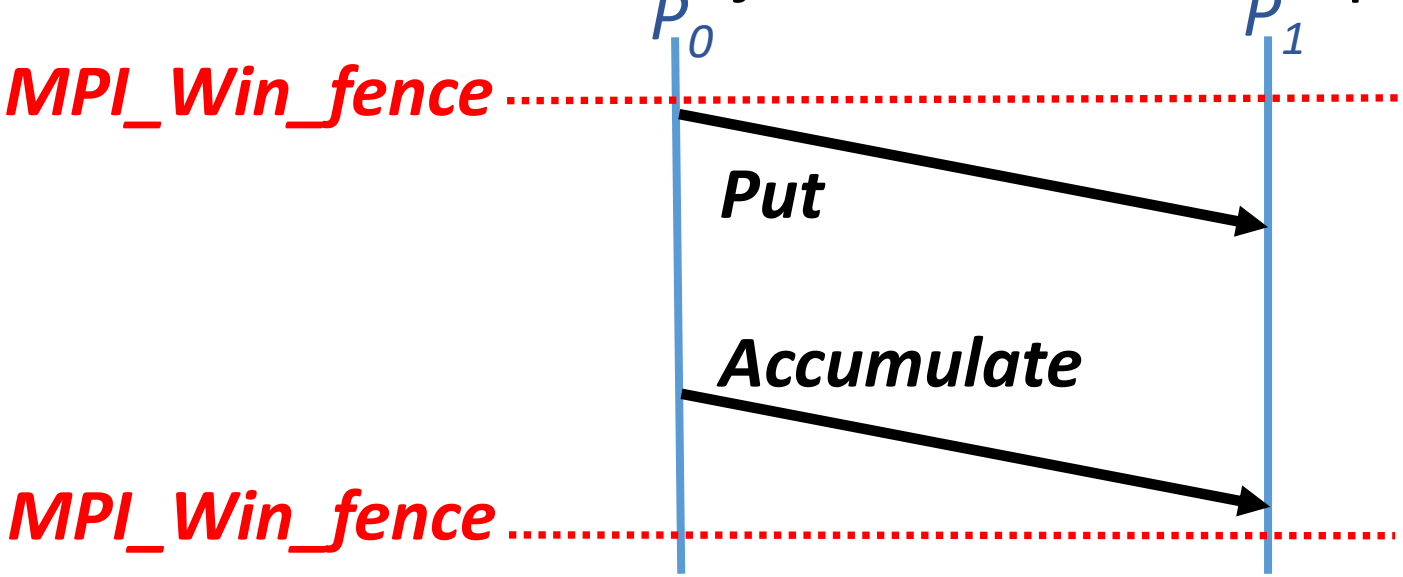
Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Sincronização com Alvo Ativo

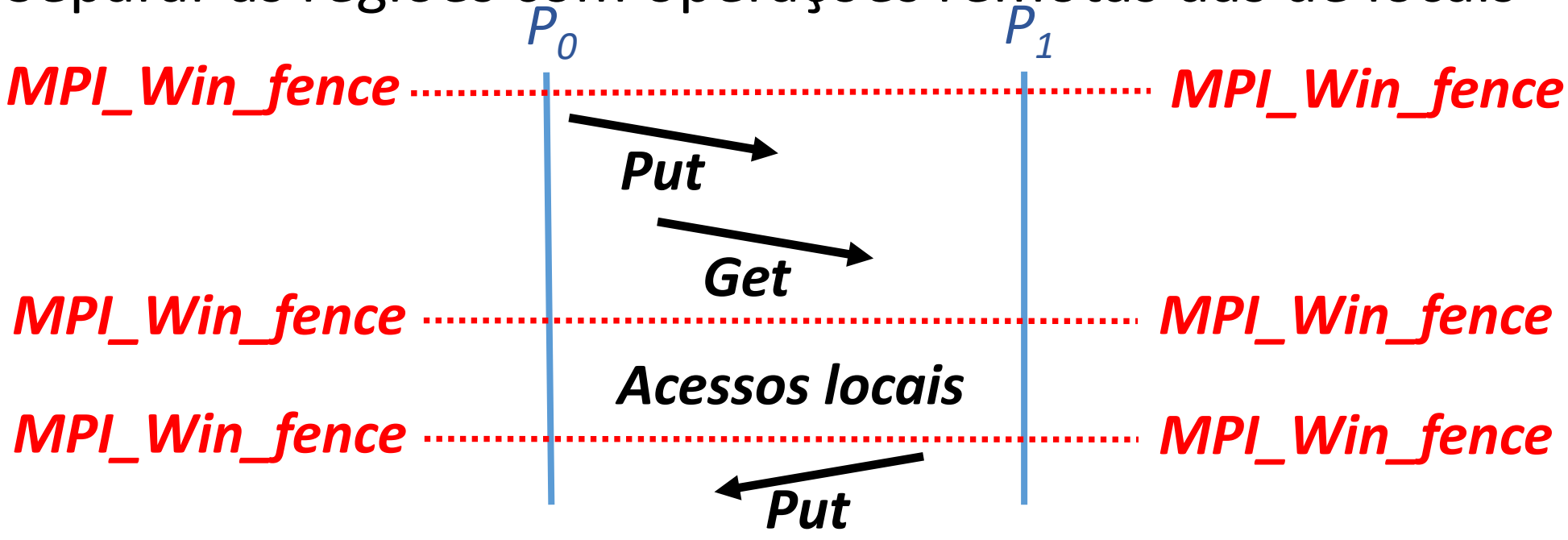
- MPI_Win_fence:
  - Primitiva de sincronização: *Active Target Synchronization*
  - Operação coletiva, chamada por todos os membros com acesso à janela, inclusive o proc. *Target*



→ Apenas P_0 participa das movimentações de dados. Porém, tanto P_0 como P_1 estão envolvidos na sincronização com *fence*

Sincronização com Alvo Ativo

- Principais Utilizações de *fence*:
  - Garantir que as operações RMA já terminaram
  - Separar as regiões com operações remotas das de locais

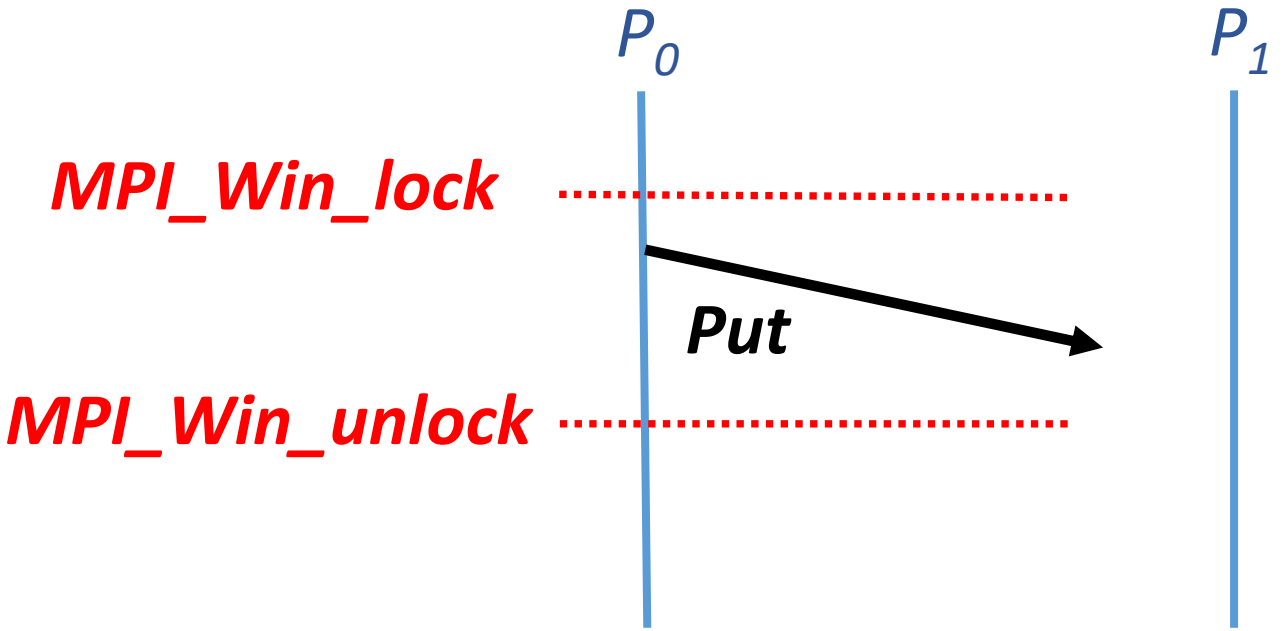


Sincronização com Alvo Passivo

- Idéia:
  - Impor que apenas um lado participe da sincronização
  - Sincronização "passiva": *Passive Target Synchronization*
    - Processador *target* não precisa chamar funções MPI
- Utilização:
  - *Locks* em MPI
  - Chamadas ocorrem apenas no processo de origem
  - Funções: *MPI_Win_lock( ), MPI_Win_unlock( )*
  - Comunicação permanence sendo com *MPI_Put/Get/etc*

Sincronização com Alvo Passivo

- Utilização (cont.):
  - Chamadas de MPI apenas em P_0
  - Argumento: janela em *target* e *rank* (alvo:P_1)



Sincronização com Alvo Passivo

- Chamada para *lock* em MPI:

  MPI_Win_lock(int lock_type, int rank, int assert, MPI_Win win)

  - *lock_type*: dois tipos possíveis
    - *MPI_LOCK_SHARED*: acessos de várias origens são permitidos
    - *MPI_LOCK_EXCLUSIVE*: apenas um acesso simultâneo
  - *rank*: rank do processador da janela alvo
  - *assert*: parametro para otimização da chamada
  - *win*: objeto janela

Sincronização com Alvo Passivo

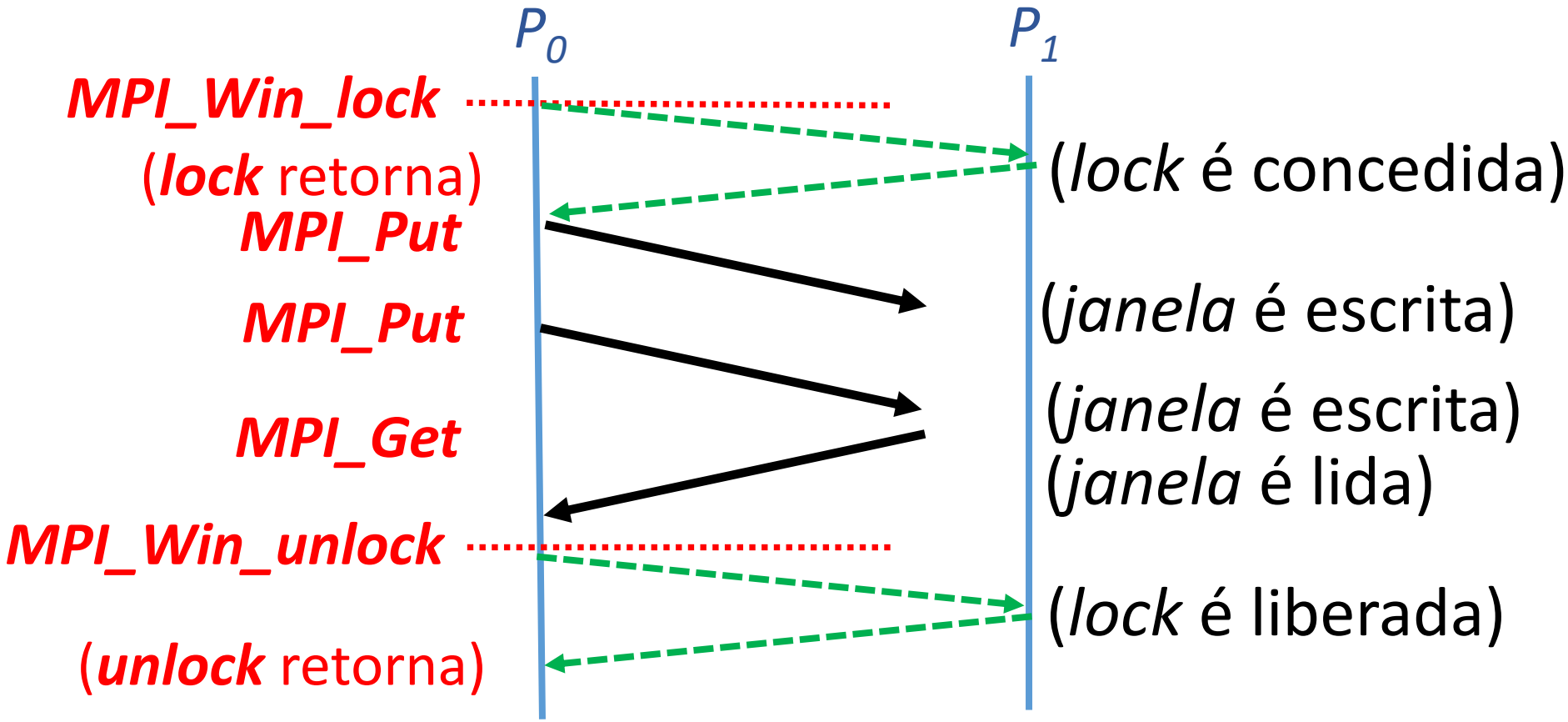
- Chamada para *unlock* em MPI:

  MPI_Win_unlock(int rank, MPI_Win win)
  - *rank*: rank do processador da janela alvo
  - *win*: objeto janela

  Após o retorno de *MPI_Win_unlock( )*, operações RMA executadas entre *lock()* e *unlock()* estarão completas. Notar que o processador alvo (onde a janela reside) não efetua chamadas MPI!

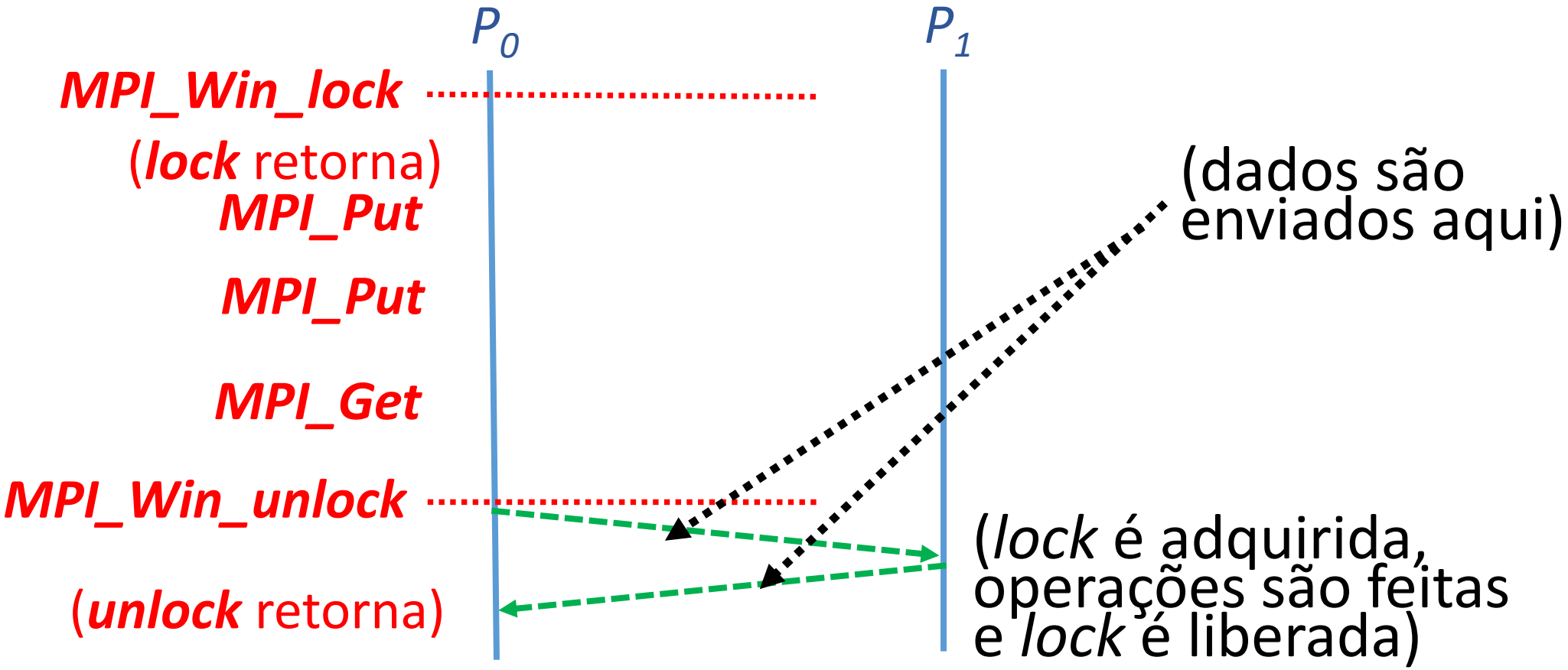
Sincronização com Alvo Passivo

- Possíveis implementações: primeira forma



Sincronização com Alvo Passivo

- Possíveis implementações: outra forma



Sincronização com Alvo Passivo

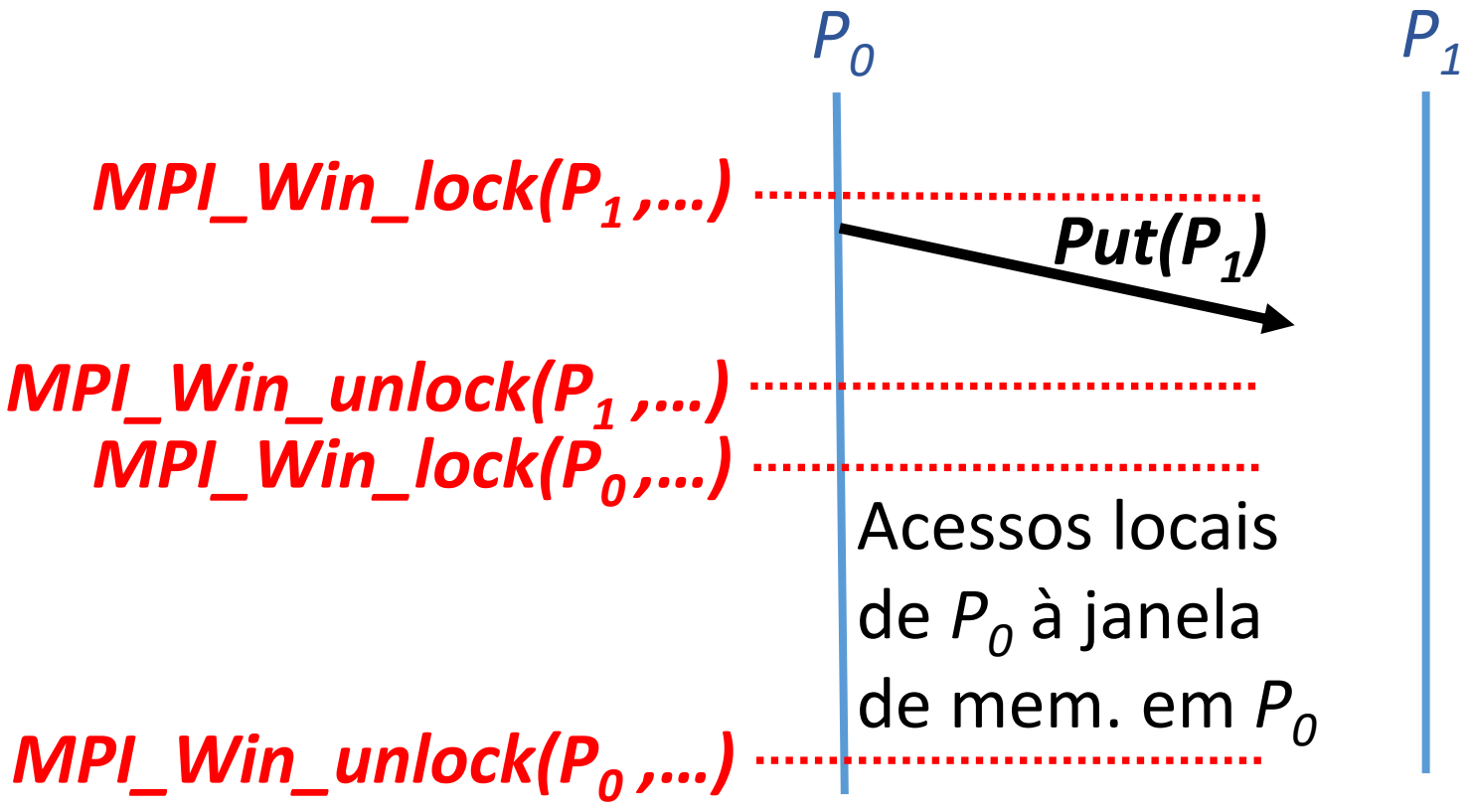
- Qual a forma de implementação mais eficiente?
  - Vários fatores a considerar:
    - Volume dos dados a serem transferidos
    - Número de operações de *put/get*
    - Existência de computação concorrente
- Solução comum
  - Decidir a forma baseado no volume de dados esperado
- Uma única certeza:
  - Transferências estarão completas somente após o retorno da função *unlock*

Sincronização com Alvo Passivo

- Semântica para *lock* e *unlock* em MPI:
  - *MPI_Win_lock*: inicia acesso a alvo passivo
  - *MPI_Win_unlock*: termina acesso a alvo passivo
- Caso especial:
  - Rank do processador da janela alvo pode ser o mesmo que faz a chamada a *MPI_Win_lock( )*
    - Pode ser usado para proteger acessos à janela local, da mesma forma que *locks* convencionais

Sincronização com Alvo Passivo

- Caso especial: aplicação



Operações Unilaterais com Bloqueio

- *MPI_Put/Get/Accumulate* originais
  - Operam sem bloqueio: retorno imediato
  - Transferências podem ocorrer em paralelo com o código que vem em seguida
- Criação de versão com bloqueio
  - Basta chamar *lock()* antes e *unlock()* após a chamada da função sem bloqueio correspondente
  - Neste caso, só há um retorno após a transferência ser concluída

Operações Unilaterais com Bloqueio

Exemplo:

```
int My_Blocking_Put(...<args>...)
{
        MPI_Win_lock(...)
        int err = MPI_Put(...<args>...)
        MPI_Win_unlock(...)
        return err;
}
```

Retorno só ocorrerá após a conclusão da transferência

Alocação de Memória para Janelas

- Com alvos ativos (*active target sync*)
  - Qualquer buffer local pode ser usado como janela
- Com alvos passivos (*passive target sync*)
  - Pode ser mais difícil usar "qualquer" buffer
  - O padrão MPI permite restrições ao tipo de buffer a ser utilizado em janelas
  - Duas formas de alocação possíveis em MPI-3
    - *MPI_Win_allocate* (já vista): preferível, pois permite alocação mais uniforme em todos os processos da janela
    - *MPI_Alloc_mem* + *MPI_Win_create*

Alocação de Memória para Janelas

*int MPI_Win_allocate(MPI_Aint size, int disp_unit, MPI_Info info, MPI_Comm comm, void *baseptr, MPI_Win win)*

*int MPI_Alloc_mem(MPI_Aint size, MPI_Info info, void *baseptr)*

*Int MPI_Free_mem(void *base)*

- *Free_mem* pode ser usada para liberarar memória alocada com qualquer tipo de alocação