Multiprocesorski sistemi

MPI
Jednostrana komunikacija

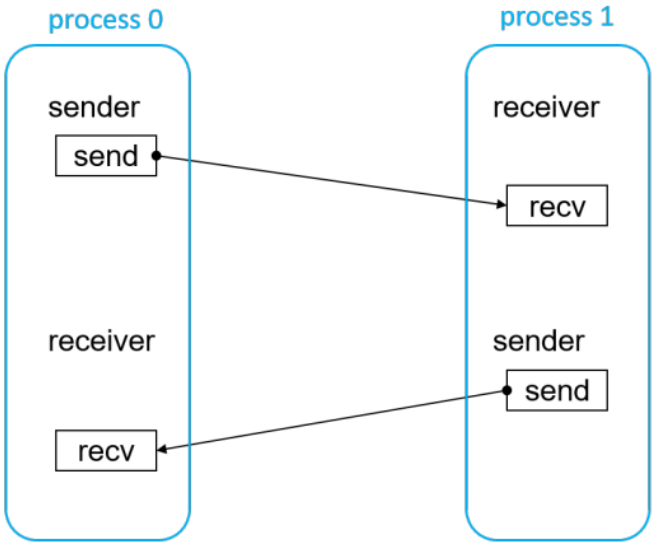
Pavle Divović, Marko Mišić

13S114MUPS, 13E114MUPS

2021/2022.

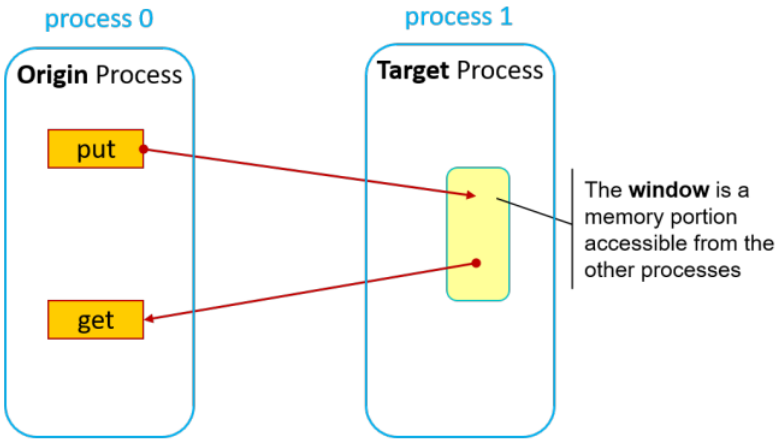
Dvostrana komunikacija

- Dvostrana komunikacija (*two-sided communication*):
  - Zahteva da oba proces učestvuju u kominkaciji
  - Procesi su ravnopravni učesnici (oba mogu da šalju i primaju)
  - Koristi se neka vrsta implicitne sinhronizacije
  - Tradicionalni model MPI komunikacije kroz *Send/Recv* rutine



Jednostrana komunikacija

- Jednostrana komunikacija (*one-sided communication*) :
  - Samo jedan process aktivan - *origin*
  - Drugi process pasivno učestvuje u komunikaciji – *target*
  - Eksplicitno se zahteva sinhronizacija
  - Model realizovan PUT/GET rutinama za pristup udaljenoj memoriji (*Remote Memory Access* - RMA)



Prednosti jednostrane komunikacije

- Umanjena sinhronizacija
  - Dvostrana komunikacija uvek ima implicitnu sinhronizaciju za svaki poslat podatak
  - Jednostrana komunikacija može da serijalizuje nekoliko PUT/GET zahteva pre sinhronizovanja
- Neblokirajuća komunikacija
  - Samo proces pošiljalac je aktivan prilikom slanja
  - Primajući proces može da obavlja drugi zadatak za vreme slanja podataka
- Skalabilnost
  - Smanjena količina nepotrebno razmenjenih podataka sa povećanjem broja procesa
- Poboljšane performanse za mnoge tipove aplikacija

Operacije jednostrane komunikacije

- Kreiranje i alokacija prozora
  - Svaki proces u grupi procesa obuhvaćenoj komunikatorom definiše deo sopstvene memorije – prozor
  - Prozoru mogu pristupiti udaljeni procesi
- Rutine za udaljeni pristup (RMA)
  - Neblokirajući pristup udaljenim prozorima
    - PUT/GET/ACCUMULATE i slične
- Rutine za sinhornizaciju
  - Okružuju rutine za neblokirajući pristup
  - Garantuju da su RMA operacije lokalno i udaljeno završene
  - Garantuju da su sve neophodne keš operacije implicitno završene
  - Dva tipa rutina: aktivne i pasivne

Prozori (1)

- Jednostrana komunikacije se odvija kroz koncept prozora
- Prozor je memorijski prostor jednog procesa koji je vidljiv i dostupan za pristup drugim procesima
- Proces primalac definiše prozor memorije koji je dostupan drugim procesima
  - **`MPI_Win_create`** – stvara prozor nad postojećom memorijom
  - **`MPI_Win_allocate`** – alocira novu memoriju
  - **`MPI_Win_dynamic`** – alocira dinamički bafer (nepoznate veličine)
  - Kolektivne operacije
- Svaka alokacija prozora mora biti praćena dealokacijom
  - **`MPI_Win_free`**

Prozori (2)

- Sve operacije vraćaju ručku (objekat) prozora **`win`**
  - Ona se kasnije koristi kod svih RMA poziva
- **`MPI_Win_create (baseptr, size, disp_unit, info, comm, win)`**
- **`MPI_Win_allocate (size, disp_unit, info, comm, baseptr, win)`**
  - **`baseptr`** – početna adresa kreiranog ili alociranog prozora
  - **`size`** – broj bajtova
  - **`disp_unit`** – veličina elementa bafera
  - **`info`** – tipično **`MPI_INFO_NULL`**
- **`MPI_Win_free(win)`**

```
MPI_Win_allocate((MPI_Aint)(10*sizeof(int)), sizeof(int),
        MPI_INFO_NULL, MPI_COMM_WORLD, &buf, &win);
MPI_Win_free(&win);
```

Remote memory access - RMA (1)

- Proces pošiljalac koristi rutine za pristup memorijskom prozoru
  - **`MPI_Put`** – čuva podatak iz pozivajućeg procesa u prozoru primaoca
  - **`MPI_Get`** – čita podatak iz prozora udaljenog procesa
- Akumulacione rutine:
  - **`MPI_Accumulate`** – izvršava operaciju akumuliranja nad prethodnom vrednosti
  - **`MPI_Get_accumulate`** – izvršava operaciju akumuliranja i vraća prethodnu vrednost
  - **`MPI_Fetch_and_op`**, **`MPI_Compare_and_swap`**
- Rutine su atomične samo na nivou elementa!

Remote memory access - RMA (2)

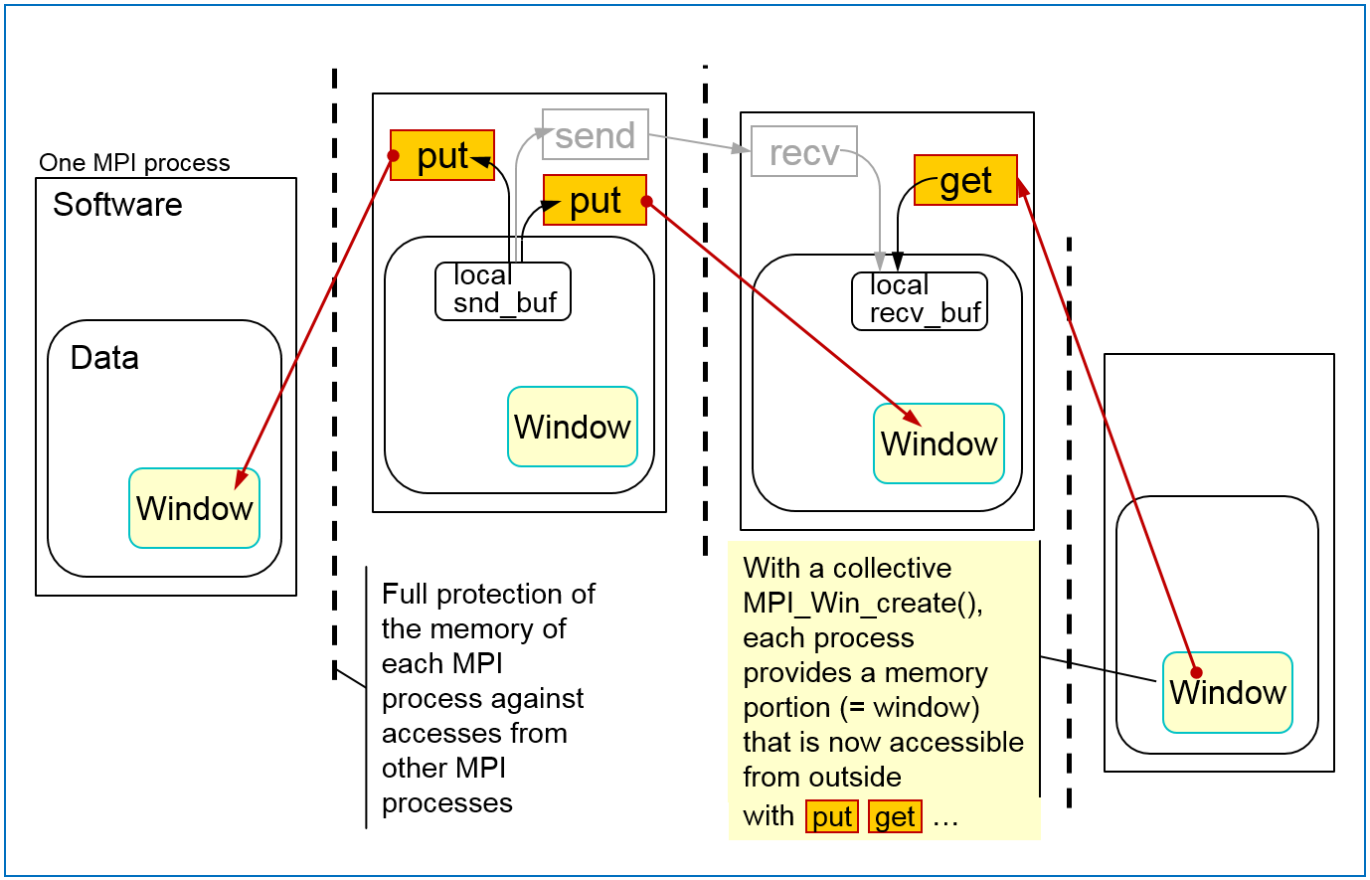
- **`MPI_Put(o_addr, o_cnt, o_dtype, t_rank, t_offs, t_cnt, t_dtype, win)`**
  - Podatak se upisuje na adresu `t_start + t_offs*t_displ_unit`, definisane u prozoru *target*-a
- **`MPI_Get(o_addr, o_cnt, o_dtype, t_rank, t_offs, t_cnt, t_dtype, win)`**
- **`MPI_Accumulate(o_addr, o_cnt,o_dtype, t_rank, t_offs, t_cnt, t_dtype, op, win)`**
  - `Op` je tipa **`MPI_Op`**
  - **`MPI_SUM, MPI_MIN, MPI_PROD, MPI_REPLACE`**
    - Kao kod **`MPI_Reduce`**

Remote memory access - RMA (3)

- Zajednički argumenti PUT/GET/ACCUMULATE
  - **`o_addr`** – adresa podatka iz pozivajućeg procesa koja ne mora biti u prozoru
  - **`o_cnt`** – broj elemenata tipa **`o_dtype`** koji će biti transferisan
  - **`t_rank`** – rang *target* procesa
  - **`t_offs`** – pomeraju odnosu na prozor *target* procesa izražen u **`disp_unit`** kod stvaranja prozora
  - **`o_dtype`**, **`t_dtype`** – MPI tip elementa
  - **`win`** – objekat prozora

Primer komunikacije (1)

- Svaki proces je tipično i *origin* i *target*



Primer komunikacije (2)

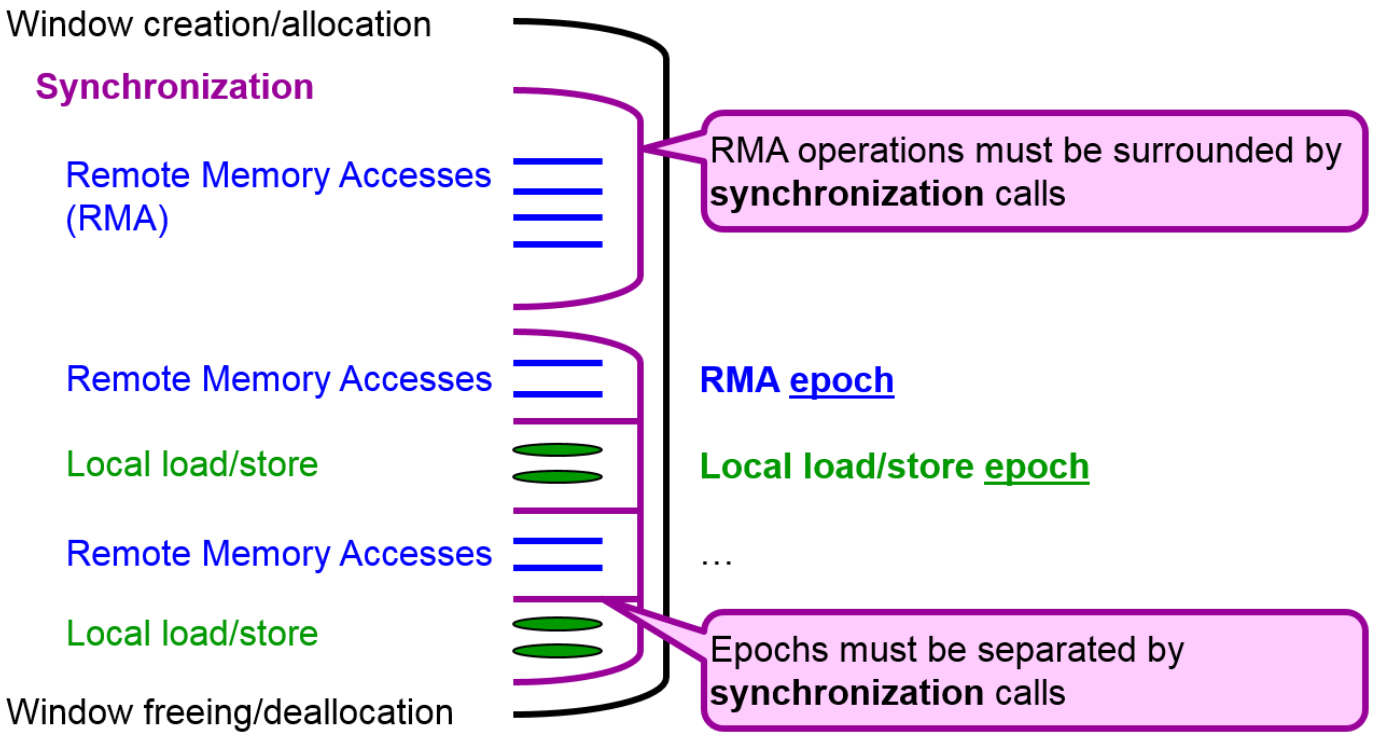
- Svaki MPI proces ima svoj (zaštićeni) adresni prostor
- Prilikom dvostrane komunikacije:
  - Jedan proces zove **`MPI_Send`** operaciju koja čita podatak iz lokalnog bafera za slanje
  - Drugi proces zove **`MPI_Recv`** operaciju koja smešta podatak u lokalni bafer za prijem
  - Zadatak MPI biblioteke je da organizuje prenos podataka
- Prilikom jednostrane komunikacije:
  - Svaki proces definiše prozor za komunikaciju sa drugim procesima
  - Proces koji šalje podatak poziva **`MPI_Put`** da podatak iz lokalnog bafera za slanje postavi u prozor udaljenog procesa
  - Alternativno, proces poziva **`MPI_Get`** da dohvati podatak iz udaljenog prozora i smesti ga u lokalni bafer
  - U okviru *target* procesa se ne izvršava nikakav poziv!

Utrkivanje operacija (1)

- Operacije pristupa prozorima ne garantuju sekvencijalan pristup od više procesa
  - Dve PUT operacije prave *write-write* konflikt
- Potrebno je sinhronizovati pristupe različitih procesa prozoru
  - Kroz pasivnu ili aktivnu sinhronizacija
- RMA operacija se odvijaju u okviru koncepta *epohe*
  - Epoha je definisana sinhronizacionim pozivima
  - Lokalnu epohu čini sekvenca operacija *target* procesa nad sopstvenim prozorom
  - RMA epohu čini sekvenca operacija *origin* procesa nad prozorom nekog *target* procesa

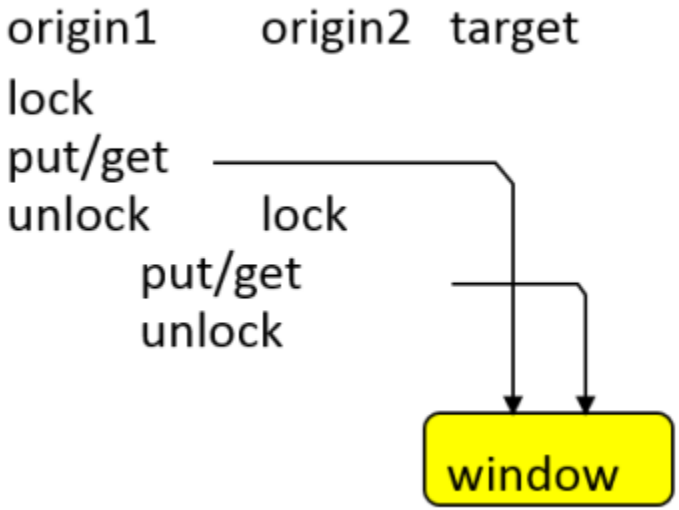
Utrkivanje operacija (2)

- Primer sekvence operacija jednostrane komunikacije u nekoliko epoha



Pasivna sinhronizacija

- Koristi model deljene memorije
- *Origin* proces vrši sinhronizaciju rutinama LOCK/UNLOCK

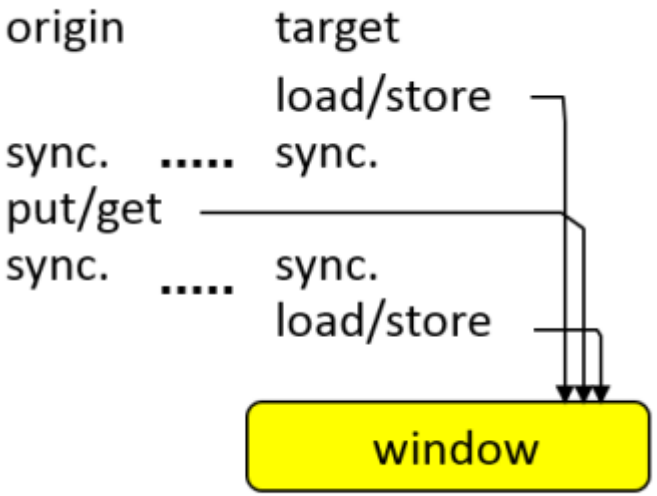


Lock/unlock

- Jednom prozoru mogu pristupati samo procesi koji poseduju *lock*
  - *Target* ne zna koji procesi imaju bravu
- **`MPI_Win_lock(lock_type, rank, assert, win)`**
  - **`lock_type`** – **`MPI_LOCK_EXCLUSIVE`** ili **`MPI_LOCK_SHARED`**
  - **`assert`** – parametar za optimizaciju (0 ne radi optimizaciju)
  - **`win`** – objekat prozora
- **`MPI_Win_unlock(rank, win)`**

Aktivna sinhronizacija

- *Target* proces učestvuje u sinhronizaciji
- Postoje dva pristupa:
  - **`MPI_Win_fence`** - slično barijeri
  - **`MPI_Win_start, MPI_Win_complete, MPI_Win_post, MPI_Win_wait, MPI_Win_test`**



Fence operacija

- **`MPI_Win_fence(assert, win)`**
- Funkcioniše kao barijera
  - Sinhronizuje sekvencu RMA poziva
- Kolektivna operacija
- Mora biti pozvana pre i posle pristupa

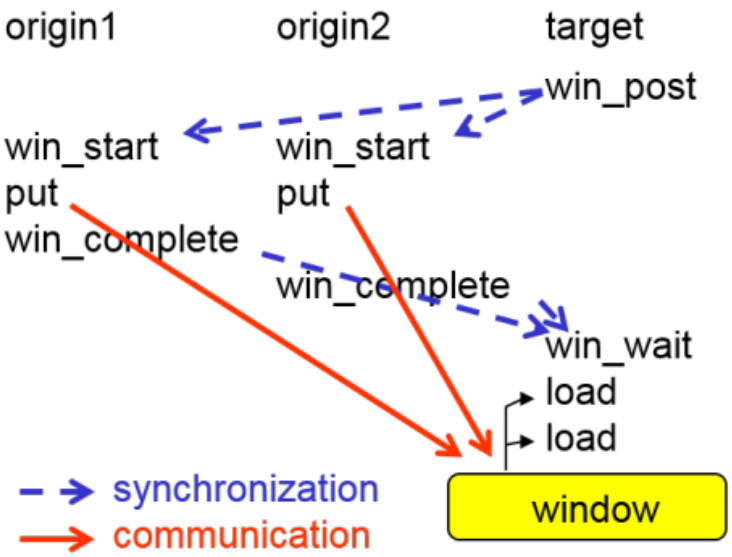
```
MPI_Win_fence(0, win);
for (idx = 0; idx < num_neighbours; idx++){
        count = 1; // dests[idx] se inkrementira za 1
        MPI_Accumulate(&count, 1, MPI_INT, dests[idx], 0, 1, MPI_INT,
                MPI_SUM, win);
}
MPI_Win_fence(0, win);
```

Start/complete – post/wait (1)

- *Target* poziva **`MPI_Win_post`**
  da dozvoli pristup prozoru
  - Započinje RMA epohu i omogući pristup svom prozoru
- *Origin* poziva **`MPI_Win_start`**
  da započne RMA sekvencu operacija
- *Target* poziva **`MPI_Win_wait`**
  gde čeka na završetak sekvence
  - Alternaivno **`MPI_Win_wait`** za neblokirajuće testiranje
- *Origin* poziva **`MPI_Win_complete`**
  gde signalizira kraj sekvence *target-u*
  - Završava se RMA epoha

Start/complete – post/wait (2)

- Moguća je komunikacija više *target*-a sa više *origin*-a
  - Međutim, svi partneri u komunikaciji moraju biti poznati
  - Definiše se putem odgovarajuće MPI grupe



Post/wait operacije

- Više procesa iz grupe može istovremeno pristupati istom prozoru
  - *Target* zna kojim procesima otvara prozor
- **`MPI_Win_post(group, assert, win)`**
  - **`group`** – grupa procesa koji mogu pristupiti prozoru
- **`MPI_Win_wait(win)`**
- **`MPI_Win_test(win, flag)`**
  - Neblokirajuća provera, rezultat se cuva u **`flag`**

Start/complete operacije

- Jedan *origin* proces u toku sekvence može pristupiti više *target* procesa
  - Deljeni prozori
- **`MPI_Win_start(group, assert, win)`**
  - **`group`** – grupa *target* procesa kojima se pristupa u RMA sekvenci, tipično jedan
- **`MPI_Win_complete(win)`**

Rekapitulacija jednostrane komunikacije

- Glavna prednost jednostrane komunikacije je u redukcija sinhronizacije
  - Mogućnost da se više RMA operacija izvrši u jednoj epohi
- Potrebno pažljivo definisanje epoha
  - Kroz metode pasivne i aktivne komunikacije
- Sa odgovarajućom podrškom hardvera, može doneti značajno poboljšanje performansi

Izvori

- PRACE, One-Sided Communication and the MPI Shared Memory Interface, FutureLearn, 2021., https://www.futurelearn.com/courses/mpi-shared-memory-interface/
- PRACE, MPI: A Short Introduction to One-sided Communication, FutureLearn, 2020. https://www.futurelearn.com/courses/mpi-one-sided/
- David Cronk, Advanced MPI,LLNL, 2007., https://hpc.llnl.gov/sites/default/files/DavidCronkSlides.pdf
- MPI Forum, MPI Standard 4.0, https://www.mpi-forum.org/