

Praktické paralelní programování (PPP 2025)

Počítačové cvičení č. 3: Komunikátory v MPI

Jiří Jaroš (jarosjir@fit.vutbr.cz)

1 ÚVOD

Cílem tohoto cvičení je vyzkoušet si tvorbu nových komunikátorů pomocí duplikace, štěpení a práce se skupinami. Dále si vyzkoušíme nastavení jména komunikátoru a implementaci vlastní rutiny pro obsluhu chyb.

Cvičení je možné vypracovat na Merlinu (bez trasování) a superpočítačích Barbora nebo Karolina v plném rozsahu.

2 PŘIHLÁŠENÍ NA BARBORU / KAROLINU A ALOKACE VÝPOČETNÍHO UZLU

Pokud používáte cluster Karolina:

1. Zažádejte o jeden uzel v interaktivním módu:

```
$ salloc -A DD-24-108 -p qcpu_exp -N 1 --ntasks-per-node 128 -t 01:00:00
```

2. Natáhněte modul s OpenMPI:

```
$ ml GCC/12.2.0 OpenMPI/4.1.4-GCC-12.2.0 CMake/3.24.3-GCCcore-12.2.0
```

Pokud používáte cluster Barbora:

1. Zažádejte o jeden uzel v interaktivním módu:

```
$ salloc -A DD-24-108 -p qcpu_exp -N 1 --ntasks-per-node 36 -t 01:00:00
```

2. Natáhněte modul s OpenMPI

```
$ ml GCC/12.2.0 OpenMPI/4.1.4-GCC-12.2.0 CMake/3.24.3-GCCcore-12.2.0
```

2.1 PŘEKLAD

Vygenerujte překladový skript pomocí cmake a spusťte překlad:

```
$ cmake -Bbuild -S.  
$ cmake --build build
```

3 PŘÍKLAD 1. - DUPLIKACE KOMUNIKÁTORU A NASTAVENÍ JEHO JMÉNA

Zadání se nachází pod sekcí `case 1`: ve funkci `main`.

1. Nejprve si deklarujte nový komunikátor, např. `pppComm`.
2. Nyní do tohoto komunikátoru zduplikujte komunikátor `MPI_COMM_WORLD`.
3. Nastavte pro komunikátor nové textové jméno.
4. Nyní zkuste v tomto komunikátoru vytvořit nějakou chybnou komunikaci, např. `rank 3` posílá zprávu neexistujícímu ranku `1000`.
5. Po ukončení práce uvolněte komunikátor.
6. Prozkoumejte chybovou hlášku, kterou od MPI dostanete.
7. Přeložte soubor.
8. Spusťte výslednou binárku:

```
$ mpiexec ./comm 1
```

4 PŘÍKLAD 2. - IMPLEMENTACE VLASTNÍ CHYBOVÉ RUTINY

Zadání se nachází pod sekcí `case 2`: ve funkci `main`.

1. Vycházejte z předchozího příkladu.
2. Definujte objekt pro nový error handler a nastavte ho na hodnotu `pppErrorHandler`.

3. Připojte takto definovaný error handler k vašemu PPP komunikátoru.
4. Proved'te chybou komunikaci.
5. Prostudujte příloženou funkci `pppErrorHandler`, modifikujte jí tak, aby opět volala `MPI_Abort`.
6. Přeložte soubor.
7. Spust'te výslednou binárku:

```
$ mpiexec ./comm 2
```

Chybové kódy standardu MPI jsou specifikovány zde <https://linux.die.net/man/3/openmpi>

5 PŘÍKLAD 3. - ŠTĚPENÍ KOMUNIKÁTORU NA ŘÁDKY A SLOUPCE

Zadání se nachází pod sekcí `case 3`: ve funkci `main`.

1. Definujte si dva komunikátory, řádkový a sloupcový.
2. Proved'te `MPI_Comm_split` nad daným komunikátorem. Uvažujte, že původní proces má {4, 9, 16} ranků.
3. Vypište na standardní výstup tyto informace.

```
Row comm: %2d /%2d, Col comm: %2d / %2d, COMM_WORLD: %2d/%2d \n
```

4. Nyní každý root řádkového a sloupcového komunikátoru rozhlásí ostatním svůj původní rank v `MPI_COMM_WORLD`.
5. Každý rank vypíše informaci o tom, kdo je a jaký je rank jeho roota v `MPI_COMM_WORLD`.

```
Row and Col Id [%d, %d], World rank %d \n
```

6. Přeložte soubor.
7. Spust'te výslednou binárku:

```
$ mpiexec -np 4 ./comm 3
$ mpiexec -np 9 ./comm 3
$ mpiexec -np 16 ./comm 3
```

6 PŘÍKLAD 4. - TVORBA DLAŽDICOVÝCH KOMUNIKÁTORŮ 2X2 RANKY

Zadání se nachází pod sekci case 4: ve funkci main.

1. Vycházejme ze situace, že máme 16 ranků, uspořádaných do matice 4×4 v dlaždicích o velikosti 2×2 . Pro tyto dlaždice chceme vytvořit komunikátory.

```
0  1      2  3
4  5      6  7

8  9      10 11
12 13     14 15
```

2. Deklarujte nový komunikátor `tileComm`.
3. Deklarujte dvě skupiny procesů, jednu pro `tileComm` a jednu pro `MPI_COMM_WORLD`.
4. Vytáhněte si do jedné skupiny seznam ranků z komunikátoru `MPI_COMM_WORLD`.
5. Vytvořte `std::vector` do kterého budete ukládat seznam ranků, které stejné dlaždice.
6. Jakmile máte tento obsah vytvořen, vložte ranky do nové skupiny `MPI_Group_incl`.
7. Na základě skupiny vytvořte nový dlaždicový komunikátor.
8. Rozešlete v rámci dlaždic původní ranky leaderů.
9. Vypište na standardní výstup následující informace:

```
Who am I? [original rank, new rank, leader] = [%2d, %2d, %2d]\n
```

10. Uvolněte skupiny a komunikátor.
11. Přeložte soubor.
12. Spusťte výslednou binárku:

```
$ mpiexec ./comm 4
```

13. Pro spuštění na Merlinovi bude potřeba použít parametr `oversubscribe`:

```
$ mpiexec -np 16 --oversubscribe ./comm 4
```

7 PŘÍKLAD 5. - SKALÁRNÍ SOUČIN VEKTORŮ VE DVOUÚROVŇOVÉ DEKOMPOZICI

Zadání se nachází pod sekcí case 5: ve funkci main.

1. Vzpomeňte si na minulé cvičení, kde jsme dělali skalární součin dvou vektorů. V tomto příkladě budeme dělat totéž, ale s dvouúrovňovou dekompozicí. Uvažujme, že máme 16 ranků. Root nejprve rozptýlí data na 4 ranky, které jsou rooty druhé úrovně. Ti následně rozptýlí svoji část dále. Po ukončení výpočtu nás čekají dvě redukce na úrovni 2 a 1.
2. Deklarujte dva komunikátory, jeden pro každou úroveň dekompozice.
3. Pomocí funkce `MPI_Comm_split` vytvořte komunikátor 1. a 2. úrovně. Pro ranky, které se nepatří do první úrovně nastavte barvu `MPI_UNDEFINED`.
4. Nyní si alokujte nutné vektory na 1. a 2. úrovni.
5. Proveďte rozptýlení vektorů přes obě úrovně. Pozor, na první úrovni musíte otestovat, jestli komunikátor existuje (`MPI_COMM_Level1 != MPI_COMM_NULL`).
6. Proveďte skalární součin na dané úrovni.
7. Proveďte dvě redukce do rootů.
8. Uvolněte komunikátory.
9. Přeložte soubor.
10. Spusťte výslednou binárku:

```
$ mpiexec ./comm 5
```

11. Pro spuštění na Merlinovi bude potřeba použít parametr `oversubscribe`:

```
$ mpiexec -np 16 --oversubscribe ./comm 5
```

Na Barboře/Karolině můžete vyzkoušet i variantu, kdy komunikátor 1 úrovně spojuje procesy mezi uzly/sockety, zatímco komunikátor 2. úrovně pracuje v rámci socketu-/uzlu.