FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Praktické paralelní programování (PPP 2023)
Počítačové cvičení č. 4: Datové typy MPI

Jiří Jaroš (jarosjir@fit.vutbr.cz)

1 ÚVOD

Cílem dnešního cvičení bude vyzkoušet si tvorbu odvozených datových typů v MPI. Nejprve se zaměříme na kontinuální datový typ pro rozeslání řádků matice, následně se podíváme na tvorbu vektorového datového typu, který využijeme pro rozeslání bloků sloupců a nakonec si vytvoříme obecný řádkový datový typ.

2 PŘIHLÁŠENÍ NA BARBORU/KAROLINU A ALOKACE VÝPOČETNÍHO UZLU

Pokud používáte cluster Karolina:

1. Zažádejte o jeden uzel v interaktivním módu.

```
$ salloc -A DD-23-135 -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-23-135 -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. - Vytvoření datového typu pro rozesílání řádků

Cílem tohoto cvičení je vytvořit kontinuální datový typ, který bude sloužit pro rozptýlení řádků matice mezi jednotlivé ranky. Každý tedy obdrží blok dat o velikosti `blockSize * nCols`. Zadání se nachází pod sekcí `case 1:` ve funkci `main`:

1. Nejprve v procesu `MPI_ROOT_RANK` alokujte matici o velikost `nRows * nCols`. Tuto matici dále inicializujte voláním metody `initMatrix` a následně ji vypište na standardní výstup pomocí volání funkce `printMatrix`.
2. Ve všech procesech dále alokujte matici `block` o velikosti `blockSize * nCols`.
3. Nyní si deklarujte odvozený datový typ `rowType`, zkonstruujte ho voláním funkce `MPI_Type_contiguous` a komitněte ho do MPI.
4. S využitím tohoto datového typy rozptylte matici.
5. Vytiskněte obsah jednotlivých bloků na každém ranku pomocí volání funkce `printMatrix`.
6. Uvolněte odvozený datový typ.
7. Přeložte soubor.
8. Spusťte výslednou binárku (případně s parametrem `oversubscribe` pro 16 procesů):

```
$ mpiexec -np 2 ./type 1
$ mpiexec -np 4 ./type 1
$ mpiexec -np 8 ./type 1
$ mpiexec -np 16 ./type 1
```

4 Příklad 2. - Vytvoření sloupcového bloku dat a jeho rozeslání pomocí ISend+Recv

Cílem tohoto cvičení je vytvořit dva odvozené datové typy tak, aby bylo možné rozeslat sloupcové bloky matice pomocí funkcí `MPI_Isend` a `MPI_Recv` (náhrada za `MPI_Scatter`, který v tomto přístupu nebude fungovat). Na straně rootu je nutné vytvořit vektorový datový typ, na straně příjemců pak stačí kontinuální datový typ. Zadání se nachází pod sekcí `case 2:` ve funkci `main`:

1. Nejprve v procesu `MPI_ROOT_RANK` alokujte matici o velikost `nRows * nCols`. Tuto matici dále inicializujte voláním metody `initMatrix` a následně ji vypište na standardní výstup pomocí volání funkce `printMatrix`.
2. Ve všech procesech dále alokujte matici `block` o velikosti `nRows * blockSize`.
3. Deklarujte dva odvozené datové typy MPI.
4. V rootu vytvořte vektorový datový typ, který rozdělí matici na stejně velké bloky o tloušťce `blockSize`. Použijte funkci `MPI_Type_vector`.
5. Ve všech procesech vytvořte kontinuální datový typ pro příjem bloku dat.
6. Pomocí smyčky rozptylte matici. Využijte volání `MPI_Isend` a `MPI_Recv`. Nezapomeňte, že i root musí přijmout svoji část. Pozor na to, že datový typ v sendu se bude lišit od datového typu pro recv.
7. Vytiskněte obsah jednotlivých bloků na každém ranku pomocí volání funkce `printMatrix`.
8. Uvolněte odvozený datový typ.
9. Přeložte soubor.
10. Spusťte výslednou binárku (případně s parametrem `oversubscribe` pro 16 procesů):

```
$ mpiexec -np 2 ./type 2
$ mpiexec -np 4 ./type 2
$ mpiexec -np 8 ./type 2
$ mpiexec -np 16 ./type 2
```

5 Příklad 3. - Tvorba vektorových datových typů pro rozptýlení matic po sloupcích s využitím funkce Scatter

Cílem tohoto cvičení je vytvořit čtyři odvozené datové typy tak, aby bylo možné rozptýlit sloupce matice pomocí funkcí `MPI_Scatter`. Jelikož budeme odesílat více než 1 sloupec současně, musíme použít i konstruktoru resized. Zadání se nachází pod sekcí `case 3:` ve funkci `main`:

1. Nejprve v procesu `MPI_ROOT_RANK` alokujte matici o velikost `nRows * nCols`. Tuto matici dále inicializujte voláním metody `initMatrix` a následně ji vypište na standardní výstup pomocí volání funkce `printMatrix`.
2. Ve všech procesech dále alokujte matici `block` o velikosti `nRows * blockSize`.
3. Deklarujte 2 odvozené datové typy - `matColType` pro odeslání a `blockColType` pro příjem.
4. V rootu vytvořte pomocný datový typ pro odeslání jednoho sloupce. Z něj vytvořte resized datový typ se správně nastavenou dolní mezí a extententem do `matColType`. Nezapomeňte výsledný typ komitnout a uvolnit pomocný typ.
5. Ve všech procesech vytvořte pomocný datový typ pro příjem jednoho sloupce. Pokud je počet přijímaných sloupců na ranku větší než 1 (`blockSize > 1`), z pomocného datového typu vytvořte resized datový typ se správně nastavenou dolní mezí a extentem do `blockColType` a uvolněte pomocný datový typ uvolněte. Pokud se bude přijímat pouze jeden sloupec, pouze přesuňte obsah pomocného datového typu do `blockColType`. Nakonec nový datový typ komitněte.
6. Použijte `MPI_Scatter` pro rozptýlení matice po sloupcích
7. Vytiskněte obsah jednotlivých bloků na každém ranku pomocí volání funkce `printMatrix`.
8. Uvolněte odvozený datový typ.
9. Přeložte soubor.
10. Spusťte výslednou binárku (případně s parametrem `oversubscribe` pro 16 procesů):

```
$ mpiexec -np 2 ./type 3
$ mpiexec -np 4 ./type 3
$ mpiexec -np 8 ./type 3
$ mpiexec -np 16 ./type 3
```

6 Příklad 4. - Rozptýlení dlaždic matice přes jednotlivé ranky

Cílem tohoto cvičení je vytvořit datový typ subarray a rozptýlit matici po dlaždicích mezi jednotlivé ranky. Jelikož budeme odesílat více dlaždic současně, navíc řazené s variabilním rozestupem, musíme použít i konstruktoru resized a funkci `MPI_Scatterv`. Zadání se nachází pod sekcí `case 4:` ve funkci `main`:

1. Nejprve v procesu `MPI_ROOT_RANK` alokujte matici o velikost `nRows * nCols`. Tuto matici dále inicializujte voláním metody `initMatrix` a následně ji vypište na standardní výstup pomocí volání funkce `printMatrix`.
2. Ve všech procesech dále alokujte dlaždici `tile` o velikosti `tileSize * tileSize`.

3. Deklarujte 2 odvozené datové typy - `tileType` pro odeslání dlaždice a `blockType` pro příjem.
4. V root ranku spočtěte správně počty dlaždic a jejich odsazení pro funkci `MPI_Scatterv`.
5. Vytiskněte obsah jednotlivých dlaždic na každém ranku pomocí volání funkce `printMatrix`.
6. Uvolněte odvozený datový typ.
7. Přeložte soubor.
8. Spusťte výslednou binárku (případně s parametrem `oversubscribe` pro 16 procesů):

```
$ mpiexec -np 4 ./type 4
$ mpiexec -np 16 ./type 4
```