Praktické paralelní programování (PPP 2025)
Počítačové cvičení č. 1: Párové komunikace v MPI

Jiří Jaroš (jarosjir@fit.vutbr.cz)

1 ÚVOD

Cílem tohoto cvičení je vyzkoušet si základy práce s MPI. Nejprve se zaměříme na inicializaci knihovny MPI a zjištění ranku a velikosti skupiny v komunikátoru `MPI_COMM_WORLD`. Následně si vyzkoušíme blokující a neblokující párovou výměnu zpráv a ověříme, že blokující výměna je náchylná k uváznutí. Na závěr si změříme propustnost a latenci neblokujících komunikací mezi dvěma procesy.

Většinu dnešního cvičení je možné vykonat na počítači v CVT případně na Merlinu, měření latence je však vhodnější vykonat na Barboře nebo Karolíně.

2 USNADNĚNÍ PŘIHLÁŠENÍ NA BARBORU/KAROLÍNU

Abychom si zjednodušili připojování na systémy IT4Innovations nastavíme si vlastní ssh klíče a vytvoříme konfigurační soubor pro ssh. Tento postup můžete provést na svém PC/NB či na Merlinu, a tak si zjednodušit život. Následující postup je pro učen pro Linux a MacOS a Windows 10/11 s OpenSSH.

2.1 TVORBA SSH KLÍČE

Tuto část většina z vás již provedla při vytváření účtu v IT4Innovations. Pokud chcete přidat další klíče (počítače), dostupujte dle následujícího návodu.

1. (Vaše PC): Tvorba nového ssh klíče:

```
ssh-keygen
```

stiskněte několikrát Enter - nic nevyplňujte

2. (Vaše PC): Zobrazte si ssh klíč a zkopírujte ho do schránky (CTRL + SHIFT + C):

```
cat .ssh/id_rsa.pub
```

3. (Vaše PC): Připojte se na Barboru/Karolinu pomocí původních klíčů:

```
ssh vaslogin@barbora.it4i.cz -i .ssh/id_rsa -X
ssh vaslogin@karolina.it4i.cz -i .ssh/id_rsa -X
```

4. (Barbora/Karolina): Otevřete soubor `authorized_keys` na Barboře a Karolíně a přidejte klíč vložený ze schránky (CTRL + SHIFT + V):

```
nano .ssh/authorized_keys
```

2.2 Tvorba souboru SSH config

SSH config umožňuje výrazně zjednodušit přihlašování na pomocí SSH (včetně tunelů, atd.).

1. (Vaše PC): Vytvořte soubor `config` ve složce `.ssh`:

```
touch .ssh/config
```

2. (Vaše PC): Otevřete soubor:

```
nano .ssh/config
```

3. (Vaše PC): Zkopírujte do souboru následující obsah. Nezapomeňte upravit svoje jméno. Pokud objevíte problém, odsaďte řádky 2-4 pomocí 2 tabulátorů.

```
Host karolina
        HostName karolina.it4i.cz
        User login
        ForwardX11 yes

Host barbora
        HostName barbora.it4i.cz
        User login
        ForwardX11 yes
```

4. (Vaše PC): Nyní se můžete připojovat takto snadno:

```
ssh barbora
sshfs barbora: local_folder
```

3 PÁROVÉ KOMUNIKACE

Soubor `p2p.cpp` obsahuje zadání cvičení. Abychom mohli problém řešit, přihlaste se buď na Merlin nebo na vlastní počítač, na kterém je potřeba mít nainstalovánu implementaci knihovny MPI a C++ překladač.

LINUX

Pokud pracujete na Linuxu, můžete si nainstalovat OpenMPI [1]. Knihovnu můžete nainstalovat také pomocí `apt`:

```
$ sudo apt-get install libopenmpi-dev
```

WINDOWS

Pokud pracujete na Windows, můžete si nainstalovat MS-MPI [2]. Je třeba stáhnout a nainstalovat `msmpisetup.exe` a `msmpisdk.msi`. Během instalace by měly být aktualizovány proměnné prostředí. Pokud ne, je třeba je nastavit ručně. Pokud pracujete na WSL, postupujte stejně jako na Linuxu.

MODULY A ALOKACE UZLU

Pokud používáte cluster Karolina:

1. Nejprve si 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 --x11
   ```

2. Natáhněte modul s OpenMPI:

   ```
   $ ml OpenMPI/5.0.3-GCC-13.3.0
   ```

Pokud používáte cluster Barbora:

1. Nejprve si 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 --x11
   ```

2. Natáhněte modul s OpenMPI:

   ```
   $ ml OpenMPI/5.0.3-GCC-13.3.0
   ```

[1] Odkaz: `https://www.open-mpi.org/`

[2] Odkaz: `https://learn.microsoft.com/cs-cz/message-passing-interface/microsoft-mpi#ms-mpi-downloads`

PŘEKLAD

1. možnost: Vygenerujte překladový skript pomocí `cmake` a spusťte překlad:

```
$ ml CMake/3.30.5
$ cmake -Bbuild -S.
$ cmake --build build --config Release
```

2. možnost: Přeložte pomocí `mpic++`:

```
$ mpic++ p2p.cpp -o p2p
```

Výsledkem je spustitelný soubor `p2p` (případně `p2p_solution`), se kterým dnes budeme pracovat.

3.1 MPI HELLO WORLD

Cílem tohoto úkolu je nejprve správně inicializovat a uklidit knihovnu MPI. Následně implementovat funkce pro zjištění ranku procesu a velikosti komunikátoru.

1. Doplňte do souboru `p2p.cpp` funkci `MPI_Init` a `MPI_Finalize`.
2. Doplňte do souboru `p2p.cpp` implementace funkcí `mpiGetCommRank` a `mpiGetCommSize`.
3. Přeložte soubor projekt a spusťte výslednou binárku:

```
mpiexec -np 2 ./p2p 1
```

3.2 MPI BLOKUJÍCÍ KOMUNIKACE

Cílem tohoto úkolu je implementovat funkci `mpiBlockingExchange`, a tím zajistit vzájemnou výměnu zpráv mezi dvěma ranky.

1. Doplňte kód do funkce `mpiBlockingExchange`, aby bylo možné poslat zprávu mezi procesy. Použijte funkce pro blokující komunikaci.
2. Přeložte soubor a spusťte vaše řešení:

```
mpiexec -np 2 ./p2p 2
```

3. Ověřte, že zprávy byly doručeny.

3.3 MPI BLOKUJÍCÍ DEADLOCK TEST

Cílem tohoto cvičení je ukázat si náchylnost standardního blokujícího sendu na deadlock.

1. Přeložte soubor a spusťte vaše řešení z úlohy 2. Kód úlohy 3 není nutné upravovat:

   ```
   mpiexec -np 2 ./p2p 3
   ```

2. Při jaké velikosti zprávy dojde k deadlocku?

3.4 MPI NEBLOKUJÍCÍ KOMUNIKACE

Cílem tohoto úkolu je implementovat funkci `mpiNonBlockingExchange`, a tím zajistit vzájemnou výměnu zpráv mezi dvěma ranky neblokující komunikací.

1. Doplňte kód do funkce `mpiNonBlockingExchange`, aby bylo možné poslat zprávu mezi procesy. Použijte funkce pro neblokující komunikaci.
2. Přeložte soubor a spusťte vaše řešení:

   ```
   mpiexec -np 2 ./p2p 4
   ```

3. Ověřte, že zprávy byly doručeny.

3.5 MPI NEBLOKUJÍCÍ DEADLOCK TEST

Cílem tohoto cvičení je ukázat odolnost neblokujícího sendu vůči deadlocku.

1. Přeložte soubor a spusťte vaše řešení z úlohy 4. Kód úlohy 5 není nutné upravovat:

   ```
   mpiexec -np 2 ./p2p 5
   ```

2. Ověřte, že zprávy byly doručeny. Pokud je vaše řešení správné, nemělo by dojít k deadlocku.

3.6 MPI BANDWIDTH TEST

Cílem tohoto cvičení je porovnat propustnost při komunikaci v rámci jednoho socketu, jednoho uzlu a mezi uzly. Při práci na Merlinu můžete vyzkoušet propustnost mezi dvěma sockety.

1. Přeložte soubor a spusťte vaše řešení.

```
mpiexec -np 2 -map-by core ./p2p 6
mpiexec -np 2 -map-by socket ./p2p 6
```

2. Jaké propustnosti jste dosáhli?
3. Jaká je latence?
4. Jaká je maximální propustnost?
5. Proč křivka nejprve roste, a pak začne klesat?