Revize 3

C2115 Praktický úvod do superpočítání

13. lekce / Modul 3

Petr Kulhánek

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta,
Masarykova univerzita, Kotlářská 2, CZ-61137 Brno

Numerická integrace

paralelizace pomocí

MPI

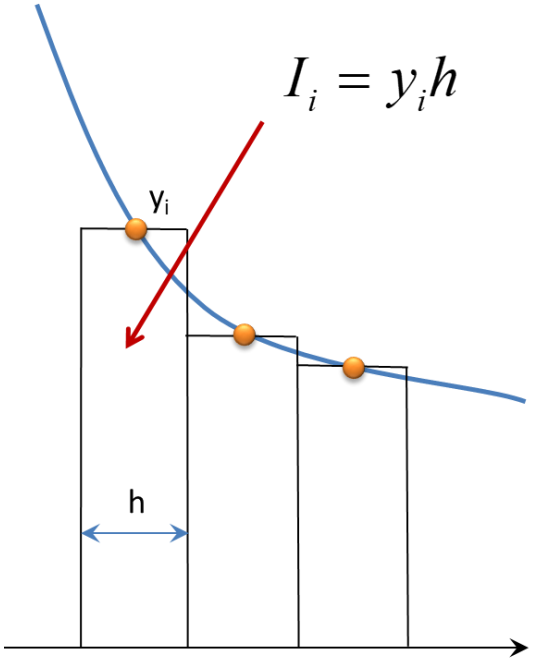
Sekvenční implementace

```
program integral

  implicit none
  integer(8)                  :: i
  integer(8)                  :: n
  double precision            :: rl,rr,h,v,y,x
  !-----------------------------------------------
  rl= 0.0d0
  rr= 1.0d0
  n = 2000000000
  h = (rr-rl)/n
  v = 0.0d0

  do i=1,n
    x = (i-0.5d0)*h + rl
    y = 4.0d0 / (1.0d0 + x**2)
    v = v + y*h
  end do

  write(*,*) 'integral = ',v

end program integral
```

$$I = \int_0^1 \frac{4}{1+x^2} dx$$

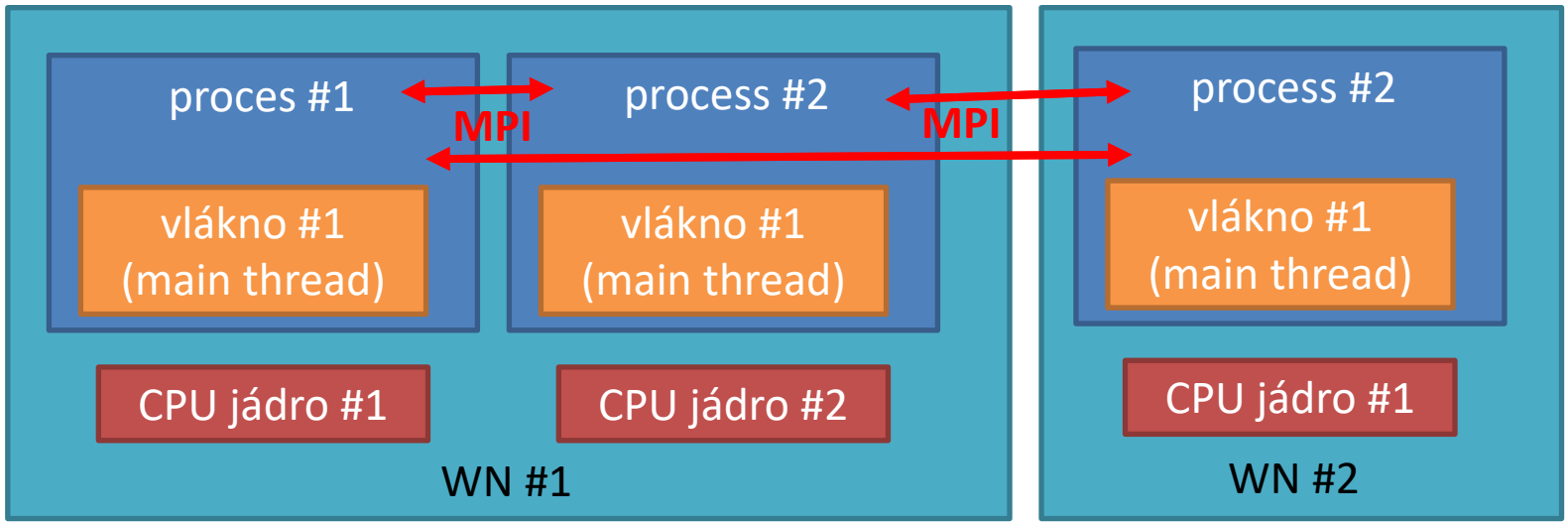
obdélníková metoda

$$I_i = y_i h$$



Paralelizace - MPI

Message Passing Interface (dále jen MPI) je knihovna implementující stejnojmennou specifikaci (protokol) pro podporu paralelního řešení výpočetních problémů v počítačových clusterech. Konkrétně se jedná o rozhraní pro vývoj aplikací (API) založené na zasílání zpráv mezi jednotlivými uzly. Jedná se jak o zprávy typu point-to-point, tak o globální operace. Knihovna podporuje jak architektury se sdílenou pamětí, tak s pamětí distribuovanou.



- MPI úlohu je možné spustit na jednom či více WN
- MPI je možné kombinovat s OpenMP.

MPI zprávy

MPI implementace

```
do i=1,n
   x = (i-0.5d0)*h + rl
   y = 4.0d0 / (1.0d0 + x**2)
   v = v + y*h
 end do
```

- Problém je nutné rozdělit tak, aby každý proces počítal část cyklu.
- Procesy nesdílí paměť, proto je nutné informace o rozdělení úlohy a předání dílčích výsledků řešit pomocí **předávání zpráv**.
- Jeden z procesů má úlohu organizátora, který řídí ostatní procesy a komunikuje s okolím.

Zdrojové kódy:
/home/kulhanek/Documents/C2115/code/integral/mpi

Komentovaný výklad k souboru **integral.f90**

MPI kompilace

aktivace MPI vývojového prostředí (OpenMPI)

```
$ module add openmpi:3.1.5-gcc
$ mpif90 -O3  integral.f90 -o integral
```

kompilátor Fortranu s podporou MPI (interně se kompiluje pomocí gfortan)

```
[kulhanek@wolf mpi]$ ldd integral
        linux-vdso.so.1 (0x00007ffe24944000)
        libmpi_mpifh.so.40 => /software/ncbr/softrepo/devel/openmpi/3.1.5-gcc/x86_64/para/lib/libmpi_mpifh.so.40 (0x0000149914f34000)
        libgfortran.so.4 => /usr/lib/x86_64-linux-gnu/libgfortran.so.4 (0x0000149914b55000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x0000149914764000)
        libmpi.so.40 => /software/ncbr/softrepo/devel/openmpi/3.1.5-gcc/x86_64/para/lib/libmpi.so.40 (0x0000149914453000)
        libopen-rte.so.40 => /software/ncbr/softrepo/devel/openmpi/3.1.5-gcc/x86_64/para/lib/libopen-rte.so.40 (0x000014991419e000)
        libopen-pal.so.40 => /software/ncbr/softrepo/devel/openmpi/3.1.5-gcc/x86_64/para/lib/libopen-pal.so.40 (0x0000149913ed9000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x0000149913cd5000)
        librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x0000149913acd000)
        libutil.so.1 => /lib/x86_64-linux-gnu/libutil.so.1 (0x00001499138ca000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00001499136ad000)
        libhwloc.so.5 => /software/ncbr/softrepo/devel/hwloc/1.11.13/x86_64/single/lib/libhwloc.so.5 (0x0000149913470000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00001499130d2000)
        libnuma.so.1 => /usr/lib/x86_64-linux-gnu/libnuma.so.1 (0x0000149912ec7000)
        libxml2.so.2 => /usr/lib/x86_64-linux-gnu/libxml2.so.2 (0x0000149912b06000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00001499128e7000)
        libquadmath.so.0 => /usr/lib/x86_64-linux-gnu/libquadmath.so.0 (0x00001499126a7000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x000014991248f000)
        /lib64/ld-linux-x86-64.so.2 (0x000014991538e000)
        libicuuc.so.60 => /usr/lib/x86_64-linux-gnu/libicuuc.so.60 (0x00001499120d7000)
        liblzma.so.5 => /lib/x86_64-linux-gnu/liblzma.so.5 (0x0000149911eb1000)
        libicudata.so.60 => /usr/lib/x86_64-linux-gnu/libicudata.so.60 (0x0000149910308000)
        libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x000014990ff7f000)
```

MPI spuštění

počet procesů, které aplikace využívá k výpočtu

```
$ mpirun -np 2 ./integral
```

soubor, který obsahuje seznam uzlů, na kterých se spouští procesy

```
$ mpirun -np 2 -machinefile nodes ./integral
```

wolf01 slots=2
wolf02 slots=3

počet CPU

název výpočetního uzlu

Předpoklady:

- ssh bez hesla
- aplikace musí být ve stejné cestě na všech uzlech, na kterých se spouští procesy

MPI a dávkový systém PBSPro:

- správně konfigurované MPI je schopné načíst přidělené zdroje automaticky z dávkového systému
- v manuálním režimu je možné použít soubor s výpočetními uzly (proměnná PBS_NODEFILE) a počet přidělených CPU v proměnné PBS_NCPU

Cvičení M3.1

Zdrojové kódy:
/home/kulhanek/Documents/C2115/code/integral/mpi

1. Zkompilujte program **integral.f90** s optimalizací **-O3** a podpory MPI.
2. Spouštějte program **integral** postupně pro 1, 2, 3, až N procesů, kde N je maximální dostupný počet CPU jader. Pro každé spuštění určete dobu běhu. Získaná data zapisujte do tabulky a vyhodnoťte jako ve cvičení M2.1.
3. Spouštějte program **integral** postupně pro 1, 2, 4, 8 až N procesů (násobky 2), kde N je maximální dostupný počet CPU na dvou uzlech klastru WOLF (volte neobsazené uzly). Pro každé spuštění určete dobu běhu. Získaná data zapisujte do tabulky a vyhodnoťte jako ve cvičení M2.1. V jiném terminálu monitorujte běžící procesy na obou výpočetních uzlech příkazem top.
4. Ovlivňuje počet CPU procesů výslednou hodnotu integrálu? Proč tomu tak je?