

C2115

Praktický úvod do superpočítání

XII. lekce

Petr Kulhánek, Tomáš Bouchal

kulhanek@chemi.muni.cz

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

Architektura

Multicore-CPU procesory:

- hyperthreading

Optimalizační techniky CPU:

- spekulativní běh (speculative execution)
- běh mimo pořadí (out-of-order execution, dynamic execution)
- SIMD (single instruction multiple data)

NUMA (Non-uniform Memory Access/Architecture):

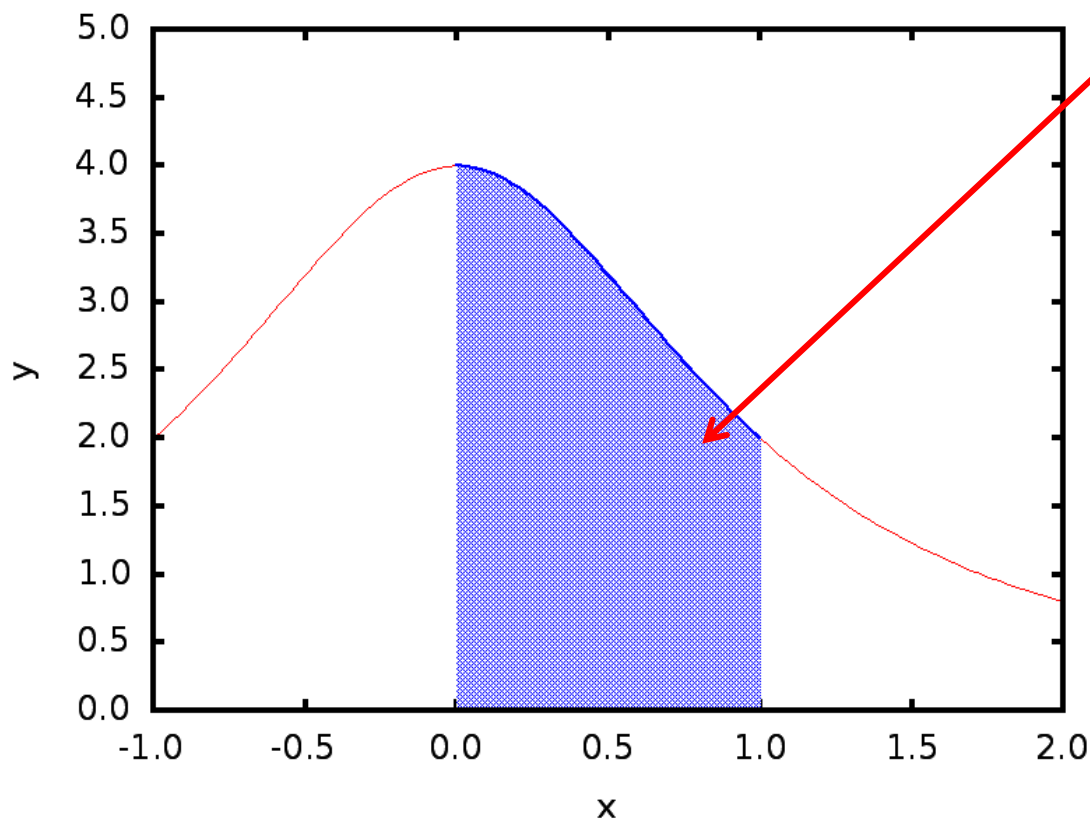
Užitečné příkazy:

```
$ lscpu
$ lstopo          # module add hwloc
$ cat /proc/cpuinfo
$ ams-host       # Infinity
```

Numerická integrace

Cvičení LIII.3

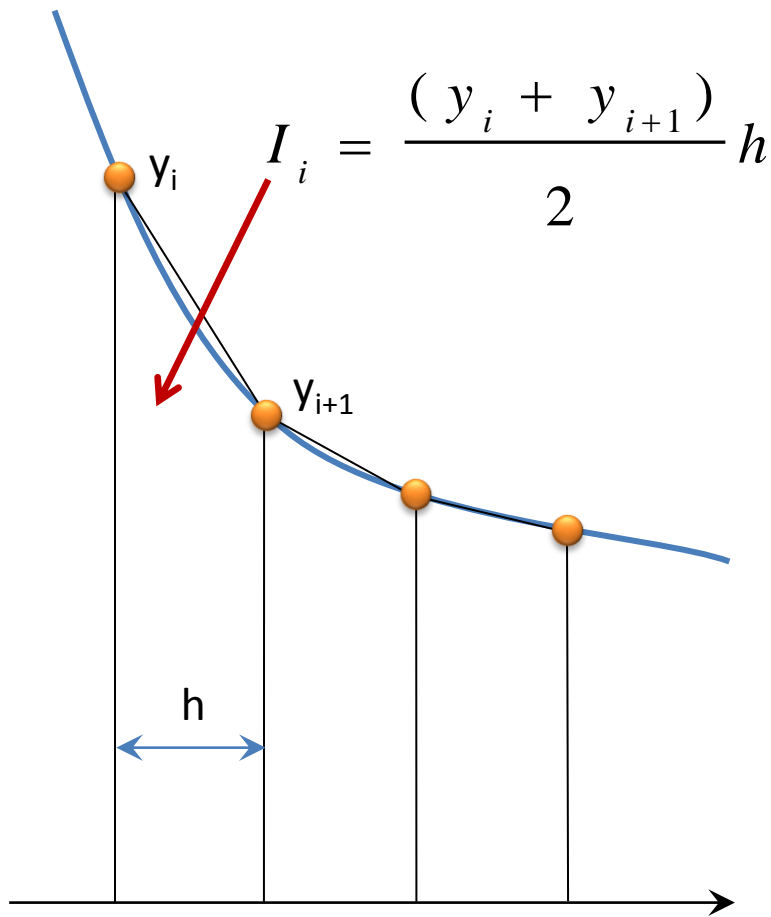
1. Napište program, který vypočte určitý integrál uvedený níže. K integraci použijte lichoběžníkovou metodu.



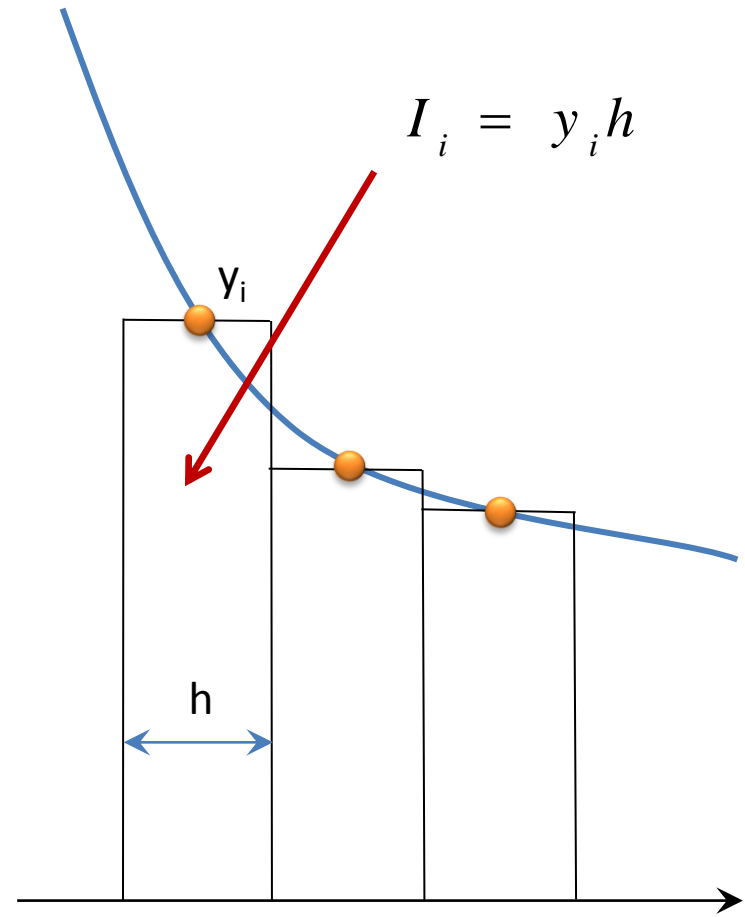
$$I = \int_0^1 \frac{4}{1+x^2} dx$$

určitý integrál je plocha pod
křivkou v rozsahu integračních
mezí

Lichoběžníková vs obdélníková metoda



lichoběžníková metoda



obdélníková metoda

Sekvenční implementace

obdélníková metoda

```
program integral
```

```
implicit none
```

```
integer(8) :: i
```

```
integer(8) :: n
```

```
double precision :: h,v,y,x
```

```
!-----
```

```
n = 2000000000
```

```
h = 1.0d0/n
```

```
v = 0.0d0
```

```
do i=1,n
```

```
  x = (i-0.5d0)*h
```

```
  y = 4.0d0/(1.0d0+x**2)
```

```
  v = v + y*h
```

```
end do
```

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

```
end program integral
```

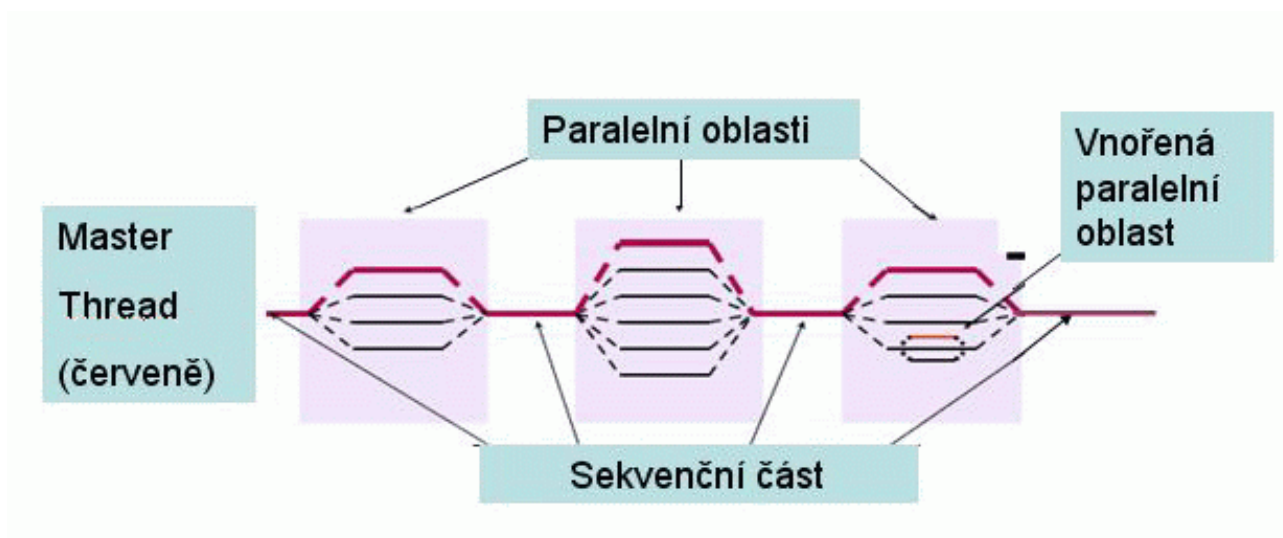
Cvičení 1

1. Zkompilujte program **integral.f90** z adresáře **/home/kulhanek/Data/C2115/Lesson12/integral/single** s optimalizací **-O3**
2. Určete dobu běhu aplikace potřebnou pro integraci funkce. K měření doby použijte program **/usr/bin/time**.

Paralelizace - OpenMP

OpenMP je soustava **direktiv** pro překladač a knihovných procedur pro paralelní programování. Jedná se o standard pro programování počítačů se sdílenou pamětí. OpenMP usnadňuje vytváření vícevláknových programů v programovacích jazycích Fortran, C a C++.

www.wikipedia.org



Specifikace: www.openmp.org

OpenMP implementace

```
ncpu = 1
!$ ncpu = omp_get_max_threads()
write(*,*) 'Number of threads = ',ncpu

!$omp parallel

!$omp do private(i,x,y),reduction(+:v)
do i=1,n
    x = (i-0.5d0)*h
    y = 4.0d0/(1.0d0+x**2)
    v = v + y*d
end do
!$omp end do

!$omp end parallel
write(*,*) 'integral = ',v
```

výhody x nevýhody

OpenMP kompilace

```
$ gfortran -O3 integral.f90 -o integral
```

```
$ ldd ./integral
```

```
linux-vdso.so.1 =>
```

```
libgfortran.so.3 => /usr/lib/x86_64-linux-gnu/libgfortran.so.3
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

```
libquadmath.so.0 => /usr/lib/x86_64-linux-gnu/libquadmath.so.0
```

```
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6
```

```
/lib64/ld-linux-x86-64.so.2
```

```
$ gfortran -O3 -fopenmp integral.f90 -o integral
```

```
$ ldd ./integral
```

```
linux-vdso.so.1 => (0x00007fff593ff000)
```

```
libgfortran.so.3 => /usr/lib/x86_64-linux-gnu/libgfortran.so.3
```

```
libgomp.so.1 => /usr/lib/x86_64-linux-gnu/libgomp.so.1
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

```
libquadmath.so.0 => /usr/lib/x86_64-linux-gnu/libquadmath.so.0
```

```
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6
```

```
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1
```


```
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
```

```
/lib64/ld-linux-x86-64.so.2
```

OpenMP spuštění

```
$ export OMP_NUM_THREADS=4
$ ./integral
Number of threads =          4
integral =      3.1415925965295672
```

počet vláken, které může aplikace využít



Poznámka: pokud není proměnná `OMP_NUM_THREADS` nastavena, použije se maximální počet dostupných CPU jader (na klastru WOLF je však výchozí hodnota proměnné `OMP_NUM_THREADS` explicitně nastavena na 1)

Cvičení 2

1. Zkompilujte program **integral.f90** z adresáře **/home/kulhanek/Data/C2115/Lesson12/integral/openmp** s optimalizací **-O3** bez podpory OpenMP.
2. Určete dobu běhu aplikace potřebnou pro integraci funkce. K měření doby použijte program **/usr/bin/time**.
3. Zkompilujte program **integral.f90** s optimalizací **-O3** a zapnutou podporou OpenMP.
4. Určete počet CPU jader na vašem počítači (lscpu).
5. Spouštějte program **integral** postupně pro 1, 2, 3, až N vláken, kde N je maximální dostupný počet CPU jader. Pro každé spuštění určete dobu běhu. Získaná data zapisujte do následující tabulky a vyhodnoťte.

N	T_{real} [s]	Speedup	CPU effectivity [%]
1	27.8	1.0	100.0
2	14.7	1.9	94.8
3	11.0	2.5	84.1
4	8.2	3.4	84.7

Diagram illustrating the calculation of Speedup and CPU effectivity:

Speedup = $\frac{T_{real}(N=1)}{T_{real}}$

CPU effectivity = $\frac{Speedup}{N} \cdot 100$

Arrows indicate that the measured time (27.8s) is used for the numerator of the Speedup formula, and the Speedup value (3.4) is used for the numerator of the CPU effectivity formula.

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 implementace

výhody x nevýhody

MPI kompilace

```
$ mpif90 -O3 integral.f90 -o integral
```

MPI spuštění

počet procesů, které aplikace využívá k výpočtu

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

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

seznam uzlů, na kterých se spouští procesy,
v dávkovém prostředí se používá hodnota
proměnné PBS_NODEFILE

Předpoklady:

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

Cvičení 3

1. Zkompilujte program **integral.f90** z adresáře **/home/kulhanek/Data/C2115/Lesson07/integral/mpi** s optimalizací **-O3**.
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 2
3. Spouštějte program **integral** postupně pro 1, 2, 4, 8 až N procesů (násobky 2), kde N je maximální dostupný počet uzlů klastru WOLF. Pro každé spuštění určete dobu běhu. Získaná data zapisujte do tabulky a vyhodnoťte jako ve cvičení 2. Spouštění je nutné koordinovat s ostatními uživateli klastru. V jiném terminálu monitorujte běžící procesy příkazem **top**.

Cvičení 4

1. Zkompilujte program **mult_mat_blas_dp.f90** z adresáře **/home/kulhanek/Data/C2115/Lesson12/matmult** s optimalizací **-O3** proti systémové knihovně blas (program obsahuje jiný způsob měření času oproti LIV.2).
2. Určete dobu běhu aplikace **mult_mat_blas_dp** programem **/usr/bin/time**.
3. Zkompilujte program **mult_mat_blas_dp.f90** s optimalizací **-O3** proti knihovně MKL.

```
gfortran -O3 mult_mat_blas_dp.f90 -o mult_mat_blas_dp \  
-lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_core \  
-lgomp -lpthread \  
-L/software/ncbr/softrepo/common/intelcore/2015.0.090/x86_64/para/lib
```

4. Určete dobu běhu aplikace **mult_mat_blas_dp** programem **/usr/bin/time**. Pro spuštění aplikace musíte aktivovat modul **intelcore:2015.0.090** Porovnejte dobu běhu mezi MKL a systémovou knihovnou blas. Která knihovna poskytuje větší výpočetní výkon?
5. Spouštějte program **mult_mat_blas_dp** postupně pro 1, 2, 3, až N vláken, kde N je maximální dostupný počet CPU jader. Počet vláken pro MKL knihovnu se nastavuje pomocí systémové proměnné **MKL_NUM_THREADS**. Pro každé spuštění určete dobu běhu. Získaná data zapisujte do tabulky a vyhodnoťte jako ve cvičení 2.