

# C2115

# Praktický úvod do superpočítání

13. lekce / Modul 2

Petr Kulhánek

[kulhanek@chemi.muni.cz](mailto:kulhanek@chemi.muni.cz)

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

# Numerická integrace

paralelizace pomocí

# OpenMP

# Sekvenční implementace

```
program integral
```

```
implicit none
```

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

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

```
double precision :: r1, rr, h, v, y, x
```

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

```
r1= 0.0d0
```

```
rr= 1.0d0
```

```
n = 2000000000
```

```
h = (rr-r1)/n
```

```
v = 0.0d0
```

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

```
  x = (i-0.5d0)*h + r1
```

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

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

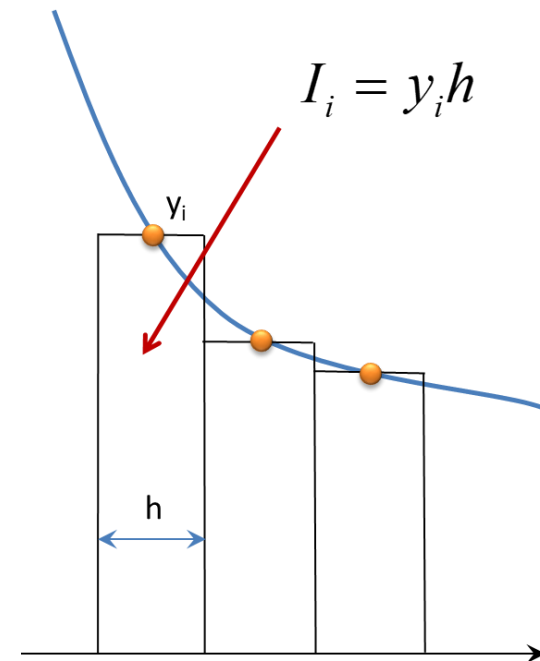
```
end do
```

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

```
end program integral
```

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

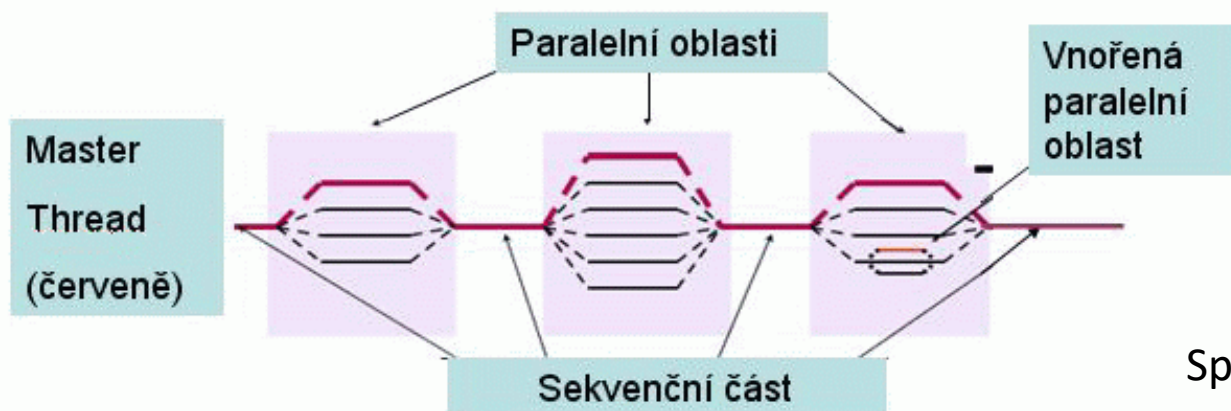
obdélníková metoda



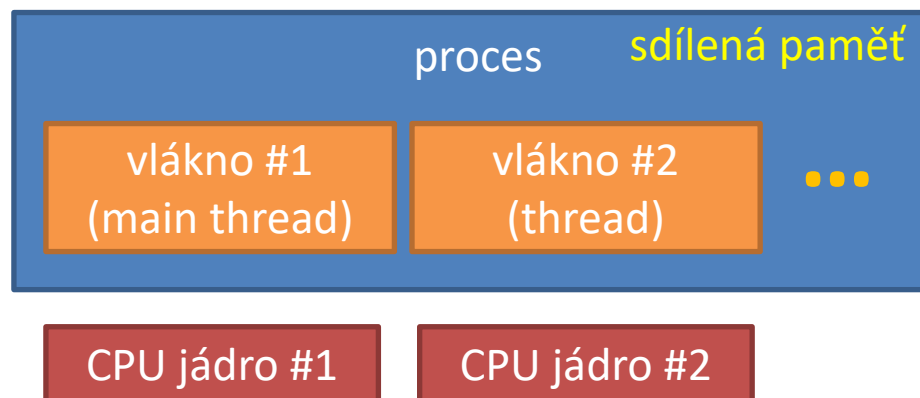
# Paralelizace - OpenMP

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

[www.wikipedia.org](http://www.wikipedia.org)



Specifikace: [www.openmp.org](http://www.openmp.org)



OpenMP je omezeno na SMP výpočetní uzel a vícejaderné CPU, popř. jejich kombinaci.

# OpenMP implementace

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

!$omp parallel

!$omp do private(i,x,y),reduction(+:v)
do i=1,n
  x = (i-0.5d0)*h + r1
  y = 4.0d0/(1.0d0+x**2)
  v = v + y*d
end do
!$omp end do

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

# 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
```

počet vláken, které může aplikace využít

```
$ ./integral
```

```
Number of threads =          4  
integral =      3.1415925965295672
```

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)

## OpenMP a dávkový systém PBSPro:

- podle konfigurace může dávkový systém hodnotu proměnné OMP\_NUM\_THREADS nastavit automaticky (dle hodnot ncpus a mpirprocs v definici bloku (chunk))
- hodnotu proměnné OMP\_NUM\_THREADS je možné nastavit explicitně:

```
export OMP_NUM_THREADS=$PBS_NCPUS
```

počet přidělených CPU, úloha musí požadovat pouze jeden výpočetní uzel


# Cvičení M2.1

## Zdrojové kódy:

/home/kulhanek/Documents/C2115/code/integral/openmp

1. Zkompilujte program **integral.f90** s optimalizací **-O3** a bez podpory OpenMP.
2. Určete dobu běhu aplikace potřebnou pro integraci. 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 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.
6. Ovlivňuje počet CPU jader výslednou hodnotu integrálu? Proč tomu tak je?

naměřený čas



| 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                |

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

$$CPUeffectivity = \frac{Speedup}{N} 100$$