Paralelní výpočty v eRku

(odkaz na tyto stránky: www.bit.ly/paralelni-vypocty-v-R)

Pokud má váš počítač několik CPU jader (ať už reálných nebo virtuálních), případně máte přístup ke skupině propojených jednojádrových počítačů, možnost paralelizace výpočtů pro vás může být zajímavá [1)]. Paralelizace spočívá v tom, že vytvoříte virtuální systém "šéfa" (*master*) a pracovníků (*workers*, což jsou vlastní jádra), kteří si určitý úkol rozdělí, zpracují ho nezávisle na sobě a na konci ho předají zpět "šéfovi". Paralelizovat jde jen určitý typ výpočtů, ne každý úkol jde totiž rozdělit na nezávislé podúkoly. Příkladem můžou být například permutační testy, které spočívají ve znáhodnění testovaných dat a výpočtu testové statistiky na nich: jednotlivé permutace jsou na sobě nezávislé, a můžou probíhat paralelně na několika jádrech najednou. Co naopak nejde paralelizovat jsou na sebe navazující úkoly, kdy následující úkol pracuje s výsledkem předchozího úkolu. Zároveň platí, že jednotlivý úkol (*chunk*) by neměl být příliš krátký. Paralelizmus funguje tak, že daný úkol šéf zadá pracovníkovi, ten ho provede a vrátí zpět šéfovi, a čeká na další úkol. Vlastní proces poslání pracovníkovi a vrácení úkolu ale zabere čas - proto pokud budete zadávat pracovníkovi mnoho krátkých úkolů za sebou, výsledný čas se zbytečně protáhne o proces komunikace se šéfem.

Od verze 2.14 jsou paralelní výpočty podporovány knihovnou `parallel`, která je součástí základní instalace programu R. Technické detaily hledejte například v tzv. *task view* k tématu High-Performance and Parallel Computing with R, případně v úvodu k této knihovně. Následující přehled je jenom instantní odvar komplikované tematiky, která se k problematice paralelních výpočtů váže.

Kolik jader můžete použít?

```
library (parallel)
detectCores()
```

Funkce, která se pokouší zjistit počet jader na vašem počítači (s různou úspěšností, viz. nápověda k funkci).

Krátká ukázka použití paralelních výpočtů

```
library (parallel)
library (snow)
library (doSNOW)
library (foreach)
library (vegan)

#kolik jader?
detectCores ()

# nacist data
vltava.spe <- read.delim
('http://www.davidzeleny.net/anadat-r/data-download/vltava-spe.txt', row.names = 1)
vltava.env <- read.delim
('http://www.davidzeleny.net/anadat-r/data-download/vltava-env.txt', row.names = 1)

# ukazka s paralelnim vypoctem zabudovanym do funkce permutest v knihovne vegan (je
treba mit nejnovejsi verzi veganu z r-forge)
# toto neni dulezite: hellingerova standardizace druhovych dat
vltava.spe.hell <- decostand (sqrt (vltava.spe), 'hell')
```

```
# ... a vlastni vypocet rda, kde pH je vysvetlujici promenna
rda.1 <- rda (vltava.spe.hell ~ pH.H, vltava.env)

# porovnani casu pro testovani prukaznosti prvni osy - 1 a 4 jadra
system.time (permutest (rda.1, perm = 40000)) # klasicky permutacni test na 1 jadru
system.time (permutest (rda.1, perm = 40000, parallel = 4))   # permutacni test na
4 jadrech


# Ukazka pouziti knihovny doSMP a foreach
# prikazy: makeCluster - inicializuje jadra, registerDoSNOW - jadra se registruji
do procesu
# foreach - obdoba for, %dopar% - pouziva se pro posilani jednotlivych cyklu na
paralelne bezici jadra


# Nasledujici skript je cyklus 500 kroku, v kazdem kroku se spocita prumer z
milionu nahodne vygenerovanych cisel (nema to smysl, ale trva to dlouho)
system.time (
{
workers <- makeCluster(4, type = "SOCK"); registerDoSNOW(workers)
temp.res <- foreach (sp = 1:500) %dopar% mean (rnorm (1000000))
stopCluster (workers)
})

# to same, ale bezici jen na jednom jadru:
system.time (temp.res <- foreach (sp = 1:500) %do% mean (rnorm (1000000)))
```

[1)]
V případě Windows OS zjistíte počet jader jednoduše tak, že kliknete na spodní lištu Windows na možnost *Spustit správce Windows*, ve kterém zvolíte záložku výkon; počet boxů v možnosti *Historie využití procesoru* odpovídá počtu dostupných jader

From:
https://anadat-r.davidzeleny.net/ - **Analysis of community ecology data in R**

Permanent link:
https://anadat-r.davidzeleny.net/doku.php/cs:mix:parallel

Last update: **2017/10/11 20:36**