R in Parallel: From Laptop to Supercomputer

Ott Toomet

Seattle, Nov 14th, 2015

Outline

R

What is R

- Tiobe top-20 programming language
- One of the most popular language for data analysis and statistics
- Superb graphics

* No built-in thread/parallel programming support
* `parallel`-package for explicit (coarse) parallelism
  - You explicitly call parallel code
* revolution R for implicit (fine) parallelism
  - The software parallelizes standard constructions automatically
  - Uses parallel libraries like *ScaLAPACK*

R

What is R

- Tiobe top-20 programming language
- One of the most popular language for data analysis and statistics
- Superb graphics

* No built-in thread/parallel programming support
* `parallel`-package for explicit (coarse) parallelism
  - You explicitly call parallel code
* revolution R for implicit (fine) parallelism
  - The software parallelizes standard constructions automatically
  - Uses parallel libraries like *ScaLAPACK*

Frontends

- R-Studio
- ESS (Emacs)
- `R CMD Batch`
- `Rscript`
  - Less bloated version of `R CMD BATCH`
  - Does not load/save workspace (see below)

Multicore Parallelism

- Almost all computers nowadays use multicore processors.
  - Shared memory
  - Fast
  - Cheap
- Let's use it!
- Example: 33 vs 75 seconds on my laptop (4 workers)
- But it does not work on windows ☹

Multicore Parallelism

- Almost all computers nowadays use multicore processors.
  - Shared memory
  - Fast
  - Cheap
- Let's use it!
- Example: 33 vs 75 seconds on my laptop (4 workers)
- But it does not work on windows ☹

Socket Clusters

- open new workers on different computers
  - including on "localhost"
- Access these over internet
- Allows to use different computers
- Example: 50 vs 75 seconds on my laptop (2 workers)
- Example with 2 computers: 23 vs 75 seconds
- Have to export data
- Communication slow
  - *top* shows the workers only partly (30%) busy with small vectors

Socket Clusters

- open new workers on different computers
  - including on "localhost"
- Access these over internet
- Allows to use different computers
- Example: 50 vs 75 seconds on my laptop (2 workers)
- Example with 2 computers: 23 vs 75 seconds
- Have to export data
- Communication slow
  - *top* shows the workers only partly (30%) busy with small vectors

Conclusion So Far

Time in seconds

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100

size	single thread	multicore	socket (localhost)	2 hosts
10^7	75	33	35	23
10^5	74	33	121	30

MPI Cluster

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100

size	single thread	multicore	MPI	2 hosts
10^7	75	33	49	23
10^5	74	33	353	30

Data Parallelism

- Run the same code on different (chunks of) data
- `pbdMPI` library
- Works well with a cluster and *mpirun*
- Can be used with distributed data (big data)

Data Parallel

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100
- hyak: $10^7 \Rightarrow$ grid 100×100, 32 CPUs

size	single thread	multicore	pbdMPI
10^7	75	33	48
10^5	74	33	46
hyak			244

Indekseerimine

Kuidas vektori/maatriksi vajalikku elementi näperdada

- `length()` – vektori pikkus
- `dim()` – maatriksi mõõtmed
- Erinevad indeksi tüübid:
  - Täisarvud: `v[c(1,2,5)]`
  - Negatiivsed täisarvud: `v[-1]`
  - Loogiline indeks: `v[c(T, F, T)]`
  - Komponentide nimed: `v[c("beta", "gamma")]`
  - Kõik komponendid: `v[]`
- Vajalikele elementidele omistamine:
  `v[v < 0] <- 0`
- Andmebaasist selekteerimine:
  `data[data$income > 0,]`

Indekseerimine

Kuidas vektori/maatriksi vajalikku elementi näperdada

- `length()` – vektori pikkus
- `dim()` – maatriksi mõõtmed
- Erinevad indeksi tüübid:
  - Täisarvud: `v[c(1,2,5)]`
  - Negatiivsed täisarvud: `v[-1]`
  - Loogiline indeks: `v[c(T, F, T)]`
  - Komponentide nimed: `v[c("beta", "gamma")]`
  - Kõik komponendid: `v[]`
- Vajalikele elementidele omistamine:
  `v[v < 0] <- 0`
- Andmebaasist selekteerimine:
  `data[data$income > 0,]`

Indekseerimine

Kuidas vektori/maatriksi vajalikku elementi näperdada

- `length()` – vektori pikkus
- `dim()` – maatriksi mõõtmed
- Erinevad indeksi tüübid:
  - Täisarvud: `v[c(1,2,5)]`
  - Negatiivsed täisarvud: `v[-1]`
  - Loogiline indeks: `v[c(T, F, T)]`
  - Komponentide nimed: `v[c("beta", "gamma")]`
  - Kõik komponendid: `v[]`
- Vajalikele elementidele omistamine:
  `v[v < 0] <- 0`
- Andmebaasist selekteerimine:
  `data[data$income > 0,]`

Arvutamine

- Põhilised matemaatilised operatsioonid: `+`, `-`, `*`, `/`
- Loogikatehted `!`, `&`, `|`, `==`, `<`, `<=`, `%in%`, ...
- Täisarvuline jagamine `\`, jääk `%%`
- Maatrikskorrutis `%*%`
- Transponeerimine `t()`
- Igasugu (vektor)funktsioonid: `log()`, `sqrt()`, `exp()`, ...
- Vektorite operatsioonid: *recycling*
- Numbriline optimeerimine: `nlm()`, `optim()`
- Numbriline võrrandite lahendamine: `uniroot()`
- Numbriline integraal: `area()`
- `options(digits=)`

Arvutamine

- Põhilised matemaatilised operatsioonid: `+`, `-`, `*`, `/`
- Loogikatehted `!`, `&`, `|`, `==`, `<`, `<=`, `%in%`, ...
- Täisarvuline jagamine \, jääk %%
- Maatrikskorrutis %*%
- Transponeerimine `t()`
- Igasugu (vektor)funktsioonid: `log()`, `sqrt()`, `exp()`, ...
- Vektorite operatsioonid: *recycling*
- Numbriline optimeerimine: `nlm()`, `optim()`
- Numbriline võrrandite lahendamine: `uniroot()`
- Numbriline integraal: `area()`
- `options(digits=)`

Arvutamine

- Põhilised matemaatilised operatsioonid: `+`, `-`, `*`, `/`
- Loogikatehted `!`, `&`, `|`, `==`, `<`, `<=`, `%in%`, ...
- Täisarvuline jagamine `\`, jääk `%%`
- Maatrikskorrutis `%*%`
- Transponeerimine `t()`
- Igasugu (vektor)funktsioonid: `log()`, `sqrt()`, `exp()`, ...
- Vektorite operatsioonid: *recycling*
- Numbriline optimeerimine: `nlm()`, `optim()`
- Numbriline võrrandite lahendamine: `uniroot()`
- Numbriline integraal: `area()`
- `options(digits=)`

Näide

- Laadi `http://www.obs.ee/~siim/ETU95.csv`
- Selekteeri vajalikud muutujad
- Arvuta vajalikud lähtesuurused
- Salvesta vahetulemused

Väljavõte ETU1995 andmebaasist

C18E0000	bruttopalk 1994 sügisel, EEK
G21	haridus: 1,2 – alg, 3,4 – kesk; 5-7 – kõrg
H01	perekonnaseis: 2,3 – (vaba)abielu
I01EKOOD	elukoha kood: 1 – Tallinn
J01	kas töötab uuringunädalal: 1 – jah, 2 – ei
L02A00	sünniaasta (kahekohaline)
L02D00	sugu: 1 – mees, 2 – naine

Näide

- Laadi `http://www.obs.ee/~siim/ETU95.csv`
- Selekteeri vajalikud muutujad
- Arvuta vajalikud lähtesuurused
- Salvesta vahetulemused

Väljavõte ETU1995 andmebaasist

C18E0000	bruttopalk 1994 sügisel, EEK
G21	haridus: 1,2 – alg, 3,4 – kesk; 5-7 – kõrg
H01	perekonnaseis: 2,3 – (vaba)abielu
I01EKOOD	elukoha kood: 1 – Tallinn
J01	kas töötab uuringunädalal: 1 – jah, 2 – ei
L02A00	sünniaasta (kahekohaline)
L02D00	sugu: 1 – mees, 2 – naine

Funktsioonid

- Interaktiivne käivitamine
- Argumendid
- Tulemused
- Kontrollstruktuurid:
  - `for()`
  - `break`
  - `if()`
  - `else`
- Trükkimine: `cat()`
- Silumine: `browser()`, `traceback()`
- Meetodid

Funktsioonid

- Interaktiivne käivitamine
- Argumendid
- Tulemused
- Kontrollstruktuurid:
  - `for()`
  - `break`
  - `if()`
  - `else`
- Trükkimine: `cat()`
- Silumine: `browser()`, `traceback()`
- Meetodid

Funktsioonid

- Interaktiivne käivitamine
- Argumendid
- Tulemused
- Kontrollstruktuurid:
  - `for()`
  - `break`
  - `if()`
  - `else`
- Trükkimine: `cat()`
- Silumine: `browser()`, `traceback()`
- Meetodid

Statistilised mudelid

- OLS – *linear model*

```
> model <- lm(response ~ explanatory + variables)
> summary(model)
```

- Logit/probit: osa üldistatud lineaarsetest mudelitest *generalised linear models*:

```
> model <- lm(response ~ explanatory + variables,
family=binomial(link="logit"))
> summary(model)
```

Statistilised mudelid

- OLS – *linear model*

```
> model <- lm(response ~ explanatory + variables)
> summary(model)
```

- Logit/probit: osa üldistatud lineaarsetest mudelitest *generalised linear models*:

```
> model <- lm(response ~ explanatory + variables,
family=binomial(link="logit"))
> summary(model)
```

Juhuslikud arvud

- Jaotused:

`.unif`	ühtlane jaotus
`.norm`	normaaljaotus
`.exp`	eksponentjaotus
`.chisq`	χ^2-jaotus
`.t`	t-jaotus
`.binom`	binoomjaotus
`.pois`	Poissoni jaotus
	...

- Statistilised tabelid:
  - `r...` juhuslike arvude generaator (`rnorm`)
  - `d...` tõenäosustihedus (`dnorm`)
  - `p...` kumulatiivne jaotusfunktsioon (`pnorm`)
  - `q...` jaotuse kvantiilid (`qnorm`)

Juhuslikud arvud

- Jaotused:
  - `.unif` ühtlane jaotus
  - `.norm` normaaljaotus
  - `.exp` eksponentjaotus
  - `.chisq` χ^2-jaotus
  - `.t` t-jaotus
  - `.binom` binoomjaotus
  - `.pois` Poissoni jaotus
  - ...
- Statistilised tabelid:
  - `r...` juhuslike arvude generaator (`rnorm`)
  - `d...` tõenäosustihedus (`dnorm`)
  - `p...` kumulatiivne jaotusfunktsioon (`pnorm`)
  - `q...` jaotuse kvantiilid (`qnorm`)

Maximum likelihood

ML tuleb teha nagu alati:

1. Kirjuta likelihoodi funktsioon
2. Maksimeeri parameetri järgi.

Näide: genereerime normaaljaotusega juhuslikke arve ja leiame valimi keskmise:

$$\ell_i(\mu, \sigma; x_i) = \log\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right)\right) = \quad (1)$$

$$= -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2} \quad (2)$$

Graafika

- Mõned näited
- Lihtne rida: `plot(x)`
- $x - y$ plot: `plot(x, y)`
- Histogramm: `hist(x)`
- Kernel tõenäosustihedus: `plot(density(x))`
- Võrdle jaotuse kvantiile: `qqnorm()`
- Funktsiooni kõver: `curve()`