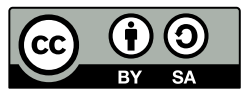
Masinių uždavinių sprendimas

Saulius Gražulis

2010 ruduo

Vilnius University, Faculty of Mathematic and Informatics
Institute of Informatics



Tai kiek gi duomenų mes turime?

- LHC generuoja 1 Terabaitą per sekundę.
  http://blogs.discovermagazine.com/cosmicvariance/2006/09/27/lhc-factoids/
  2009-12-07
- „When it starts in 2007 the LHC will ... produce 15 million Gigabytes of data a year“ (15 Petabaitų/metus $\approx 15 \times 10^{15}$ baitų/metus– S.G.)
  http://www.physorg.com/news10895.html
  2009-12-07

Tai kiek gi duomenų mes turime?

- „Release 57.11 of 24-Nov-09 of UniProtKB/Swiss-Prot contains 512994 sequence entries, comprising 180531504 amino acids abstracted from 184920 references.“ (1.81×10^8 a.r., 5.13×10^5 sekų– S.G.)
  http://www.expasy.ch/sprot/relnotes/relstat.html
  2009-12-07
- „As of Tuesday Dec 01, 2009 at 4 PM PST there are 61808 Structures“
  http://www.rcsb.org/pdb/statistics/holdings.do
  2009-12-07

Tai kiek gi duomenų mes turime?

- „Cambridge Structural Database 1 January 2009 Total No. of structures 469611“
  http://www.ccdc.cam.ac.uk/products/csd/statistics/
  2009-12-07
- „Cambridge Structural Database 1 January 2010 Total No. of structures 501857“
  http://www.ccdc.cam.ac.uk/products/csd/statistics/
  2010-12-20
- „Currently there are 86216 entries in the COD“
  http://www.crystallography.net/
  2009-12-07
- „Currently there are 126492 entries in the COD“
  http://www.crystallography.net/
  2010-12-20

Didelio duomenų kiekio apdorojimas

- Duomenų bazės: MySQL, PostgreSQL, ...
  - greitas veikimas, duomenų vientisumo garantijos
  - standartinė duomenų paieškos kalba (SQL)
  - reikalinga specialiai sukurta duomenų bazė ir serveris
  - duomenų struktūros apribotos reliacinėmis lentelėmis ir duomenų bazės schema
- bendros paskirties programavimo kalbos (b.p.p.k): Perl, Java, Python, Fortran, C, C++, Pascal, Ada, FORTH, ...
  - gali viską (Turing complete)
  - gali perskaityti duomenis iš bet kokių šaltinių (failų, tinklo prievadų, aparatūros)
  - reikalingas sudėtingas *ad-hoc* programavimas
  - duomenų mainai su failų sistema gali tapti „siaura vieta“
- *x sistemų komandos: `for`, `find`, `xargs`, ...
  - lengvai apdoroja duomenis failuose
  - universalios, beveik kaip b.p.p.k.
  - lengvai leidžia sukurti sudėtingus apdorojimo konvejerius
  - sudėtingoms užduotims gali veikti lėtai

Komanda `find`

- Ieško failų su nurodytais kriterijais visame failų medyje, surastų failų sąrašą išveda į `STDOUT`:

```
find . -name '*.pdb'
```

```
sh% time find ~/struct/cod/cif/ -name \*.cif | wc -l
126599

real0m3.570s
user0m0.936s
sys0m0.252s

sh% date +%F; time find ~/struct/cod/cif/ -name .svn -prune -o \
                                          -name '*.cif' -print \
    | wc -l
2016-11-25
369187

real0m0.669s
user0m0.424s
sys0m0.276s
```

Svarbūs komandos `find` argumentai

- Sąlygos (tests):
  - `-name '*.txt',-iname '*.txt'`
  - `-type f`
  - `-maxdepth, -mindepth`
  - `-cmin, -cnewer, -ctime`
  - `-amin, -anewer, -atime`
  - `-perm`
- Veiksmai (actions):
  - `-exec`
  - `-print, -print0`
  - `-prune`

Komanda xargs

- Komanda xargs skaito failų sąrašą iš savo STDIN ir kiekvienam failui įvykdo komandą:

```
ls -1 *.pdb | xargs grep ^COMPND

find . -name '*.pdb' | xargs grep ^COMPND
```

`xargs` komandos svarbiausi parametrai

- `xargs -n`, `xargs -n1`:
  paduoti komandai po *n* argumentų vienu metu (pvz. `-n1` – paduoti po vieną argumentą kiekvienam programos kvietimui).
- `xargs -i`, `xargs -iX`:
  Įterpti argumentą vietoj simbolių „{}“ (opcija `-i`), arba vietoj simbolio, nurodyto po `-i` (pvz. `-iX` – vietoj simbolio „X“).
- `xargs -p`
  Klausti patvirtinimo prieš vykdant kiekvieną komandą
- `xargs -t`
  Spausdinti kiekvieną komandą prieš ją vykdant
- `xargs -0`
  Laikyti, kad argumentai atskirti ne ”\n”, bet ”\0” simboliais (GNU sistemose veikia su `find -print0`, `grep -Z`, `sort -z` komandomis).

Komandinių interpretatorių `for` bei `while` ciklai

- Ciklas `for`

```
for i in *.pdb; do grep '^ATOM  ' $i; done
for i in *.pdb; do grep '^ATOM  ' $i; done | wc -l
for i in $(seq 2 2 1000); do expr $i \* $i; done
```

- Ciklas `while`

```
n=1; while [ $n -lt 1000 ]; do echo $n; awk "END{print␣sqrt($n)}"; \
n=$(($n+1)); done

n=1; while [ $n -lt 1000 ]; do echo $n $(awk "BEGIN{print␣sqrt($n)}"); \
n=$(($n + 1)); done
```