ITU−VOOP−E2001
Performanceanalyse

Lars Thorup

lars.thorup@acm.org, Nov 11, 2001

Dagens overblik

- Opsamling
- Performance
  - Hvorfor
  - Er der et problem?
  - Hvor er problemet?
- Til næste gang

Opsamling

- Tråde
- Synkronisering
- Netværkslyttetråd

Skal vi måle performance?

- Er der overhovedet et problem?
- First rule of optimization:
  - Don’t do it
- Second rule of optimization:
  - Don’t do it yet
- Designet skal sikre god performance fra starten
- Koden skal ikke optimeres før til sidst

Forskellige typer performance

- Forskellige typer ressourcer
  - Memory / disk
  - Netværk
  - CPU−tid
  - Brugertid (opstartstid, svartider)
- Enkeltbrugerperformance
- Skalerbarhed

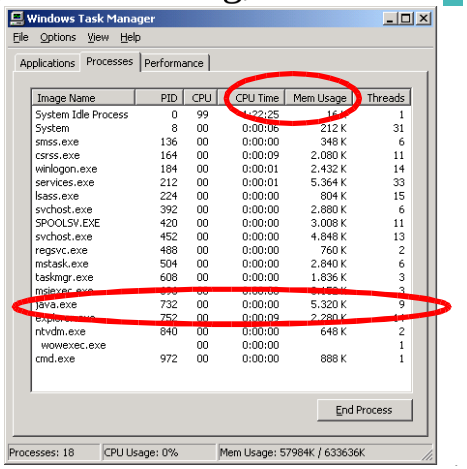
Memory− og CPU−forbrug, Linux

- `time ./confserver`
- `ps auxww | egrep "confserver|PID"`
- Server: 6Mb, Klient: 21 Mb

```
[lth@gnu lth]$ ps auxww | egrep "java|PID"
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lth       1889  0.6  2.1 88324 5648 pts/2    S    17:13   0:00 /usr/java/...
```


Memory− og CPU−forbrug, Windows

- Windows
  - Task Manager
- Server: 6Mb
- Klient: 11Mb



Brugertid

- Stopur
- java.util.Date eller System.currentTimeMillis()
  - millisecond precision
- Log time of input events
  - mouse and keypress
- Log time of last paint operation
- Fokuser på de operationer brugeren forventer går stærkt (fx tastatur−repeat).

Brugertid (2)

- Måling på lokalt kørende klienter
  - Mellem 11 og 57 millisekunder
- Bør også prøve på brugernes netværk

Skalerbarhed

- Kør klient−testsuite fra mange maskiner mod samme server
- Mål svartider som funktion af antal klienter

Performanceanalyse

- hprof: Profiling Tool i JDK
- Linux:
  - Kræver ny java 1.3.1 for at kunne køre hprof
  - virker ej alligevel (bør nok prøve 1.4)
- Windows: virker fint
- Instruktion

```
java −Xrunhprof:help
```


Performanceanalyse

- Kommandoer for måling af tid hhv memory:
  - Resultatet lagres i conf.hprof[.txt]−filen

```
java −classic −classpath .
     −Xrunhprof:cpu=times,file=conf.hprof.txt,depth=20
     dk.itu.voop.conference.server.ServerTestSuite

java −classic −classpath .
     −Xrunhprof:file=conf.hprof,format=b
     dk.itu.voop.conference.server.ServerTestSuite
```

Hvor allokeres memory?

- Kør værktøjet "hat" på conference.hprof
  - Køres via en browser på port 7000

```
start java -cp ..\lib\hat\bin\hat.zip -mx100m hat.Main
      conference.hprof
start http://localhost:7000/
```


Hvor allokeres memory?

- Show instance counts for all classes
  - click on "instance"
    - click on specific instance
      - shows where it was allocated

Hvor bruges CPU?

- Kig på mest CPU forbrugende TRACE's i rækkefølge

Kommercielt profiling-værktøj

- OptimizeIt, pris $500 = kr 4000
- CPU forbrug (online demo)
- Memory allokering

What to do

- Lazy display update
- Caching
- Brug Agent Pattern til sendetråd, hvis det bliver et problem
- JIT−compiler
- Rigtig compiler: gjc

Til næste gang

- Færdiggør implementation
- Forslag til emne sidste forelæsning
  - Gennemgang af design patterns igen
    - Af jer?
    - Af mig?