Ytelse

IMT3003 kyrre.Begnum@ntnu.no

Ytelse

- Forskjellige kategorier av ytelse
  - Maskinvare
  - Operativsystem
  - Applikasjonsnivå

Maskinvare

- Disker og kontrollere: Skrive, Lese, Søke
- Nettverkskort og infrastruktur: Gb/s
- Prosessor: GHz, kjerner, cache
- Minne kapasitet og klokke: MB, MHz
- (Skjermkort)

Maskinvare

- Ytelse er oftest i teoretiske nivåer
- Beregninger blir urealistiske, men viser likevel konkrete flaskehalser
  - Kan kontrolleres ved å oppgradere eller forandre oppsett (f.eks RAID)
  - Høy grad av forutsigbarhet per enhet (f.eks et nettverkskort blir ikke langsommere over tid)

Operativsystem

- Svært sammensatt programpakke, bestående av:
  - Kjernen, med diverse versjoner
  - Drivere til maskinvare
  - Bibliotek og systemkall som programmer kan bruke
  - Protokoller som er installert (f.eks TCP, ICMP)
  - Filsystem

Operativsystemer

- Ytelse er noenlunde forutsigbart under svært kontrollerte forhold
- Versjoner av drivere, generell alderdom og bruk påvirker ytelse
- Innstillinger i kjernen kan optimalisere i forhold til konkrete behov:
  - Schedulering
  - Minnehåndtering
  - Filsystemcaching

Filsystemer

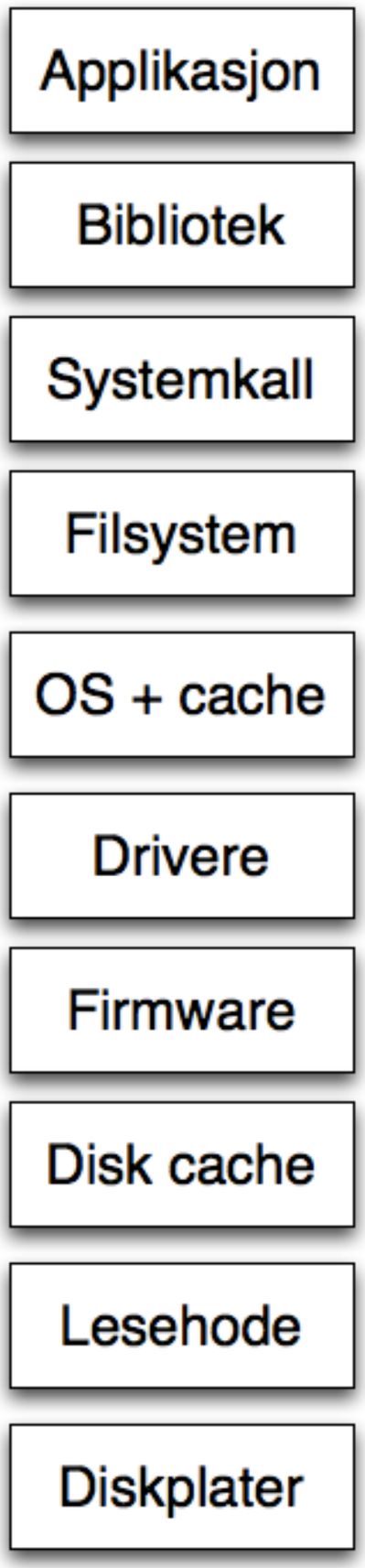
- Kan optimaliseres for store eller små filer, lesing eller skriving
- Håndterer store mengder filer forskjellig
- Det er mulig å forandre egenskaper med filsystemet, men vanskelig å bytte det når systemet er i produksjon
- Fragmenterte filsystem
  - Ikke et problem hos alle filsystem

Filsystemer

- Moderne filsystemer, slik som btrfs og ZFS støtter snapshots
- Hvis du har datafiler: legg dem på egen partisjon med skreddersydd filsystem
- Vanskelig å teste uten å gå via applikasjonslaget

Applikasjonsnivå

- Vanskelig å måle ytelse fordi bruksmønster og data er så individuelt
  - Ved variasjon i bruk vil også ytelse variere
  - Algoritmer yter forskjellig avhengig av hvordan data er lokalt
  - Likevel er det applikasjonens ytelse som er det som teller...



Alt henger sammen

- Ytelsen kan forbedres på en rekke nivåer helt fra applikasjonen og ned til disken
- Man kan velge å teste på forskjellige nivå for å finne eventuelt svinn

Alt henger sammen

- Ytelsen kan forbedres på en rekke nivåer helt fra applikasjonen og ned til disken
- Man kan velge å teste på forskjellige nivå for å finne eventuelt svinn
- Jo større avstand, desto vanskeligere å konkludere noe

Forutsigbarhet

- Nøkkelen til god ytelsesanalyse er å få riktig oversikt over systemets oppførsel
- Spør deg selv ikke om systemet er raskt, men om det er raskt nok til enhver tid gjennom regelmessig overvåking
- Informasjon finner man vanligvis i log-filer, men analysen må man gjøre selv
- Grunnleggende deskriptiv statistikk er ofte det som skal til for å klare å *beskrive* det man ser
- Uten forutsigbarhet har man ikke mulighet til å se om evt. forbedringer har noe effekt

Aritmetisk gjennomsnitt

- Summen av alle verdiene delt på antall verdier

$$\frac{\sum_{0<i<N} n_i}{N}$$

Eks:

9, 9,12, 27, 22, 22,19, 6, 26, 7, 9

Gjennomsnitt: 15.27

Median

- Verdien som er i midten av en sortert rekke av verdier

  Eks:

9, 9, 12, 27, 22, 22, 19, 6, 26, 7, 9

Sortering

6, 7, 9, 9, 9, 12, 19, 22, 22, 26, 27

Median: 12

Modus

- Verdien som opptrer oftest (Mest sannsynlig verdi)

9, 9, 12, 27, 22, 22, 19, 6, 26, 7, 9

Modus: 9

Fordelinger (Distribution plots)

- Fordelingen av verdiene i en tallserie forteller om sannsynligheten til verdiene i fremtiden
- Den mest kjente fordelingen er normalfordelingen
  - Men den er ikke den vanligste i IT verden
- Ofte har data en kompleks og sammensatt fordeling

Fordelinger (Distribution plots)

- Dersom verdiene har mange forskjellige faktorer som påvirker dem, blir fordelingene ofte sammensatt av flere maksimum (multimodal)
- Det finnes ikke mange verktøy som viser dette for oss (men vi kan jo programmere, så...)
- Utfordringen er å hente ut tallene og å 'vaske' dem

Spredning

- Gir inntrykk av hvor bredt dataene er fordelt
- Ved ordinalnivå og ikke normalfordeling, så blir percentiler brukt til å beskrive bredde rundt median

Fordelinger: Huskeregler

- Avstand mellom gjennomsnitt og median betyr at fordelingen er usymetrisk
- Ved multimodale fordelinger gir gjennomsnitt lite mening
- Liten spredning betyr homogenitet (likeartede verdier)
- Stor spredning betyr heterogenitet (stor variasjon i verdiene)

LibreOffice hjelp

- Vi bruker et LibreOffice regneark til hjelp for å analysere data
- Det eneste man trenger å gjøre er å få tak i tallene og å få dem på en kolonne

Måle filsystem ytelse

IMT3003 kyrre.Begnum@ntnu.no

To måter å undersøke ytelsen

- Aktive tester
  - Går inn og presser ytelsen til det maksimale
  - Finner ekstremverdier som vi ofte ikke ser i vanlig bruk
- Passive tester
  - Samler data fra vanlig bruk

Aktive tester

- Vi har hovedsaklig to design på aktive tester:
  - Syntetisk. En serie eksperimenter der vi kontrollerer så mye av prosessen som mulig og sanker inn data.
  - Realistisk. Vi bruker systemet slik som brukeren ville gjort det og måler ytelsen på operasjonene.
- Mange syntetiske tester prøver å gjøre så realistiske operasjoner som mulig.
- Å gjøre en aktiv test utgjør naturligvis en stor belastning. Man kan altså ikke gjøre det når man vil.

Passive tester

- Beskriver den ytelsen man har sett så langt, istedenfor å finne yttergrensene
- Belaster ikke systemet og kan gjøres hele tiden
- Informasjonen ligger vanligvis i /proc i Linux
- Gir et her-og-nå bilde
- Eksempel vi kjenner: Munin

Bonnie++

- Aktiv filsystem benchmark for Linux
- Installeres med apt-get install bonnie++
- Syntetisk test som gjør lav-nivå fil operasjoner
  - Bruker systemkall og måler hvor lang tid de tok, samt hvor mye CPU som ble brukt til hvert kall.
  - En enkelt test:
    bonnie++ -d mappe -m maskinnavn -q

bonnie++ i system

- En enkelt test er ikke representativ nok
- Vi vil gjenta testen ofte og vise resultatene på en smart måte
- La bonnie++ kjøre 10 tester og lagre output til fil:

  bonnie++ -d mappe -m maskinnavn -q -x 10 > bout.dat
- Filen bout.txt vil nå inneholde ti rader med resultater
- Vi kan bruke to verktøy for å lage enten vakker tekst eller HTML av resultatfilen:
  cat bout.dat | bon_csv2html > bout.html
  cat bout.dat | bon_csv2txt > bout.txt

HTML utskrift

		Sequential Output						Sequential Input				Random Seeks			Sequential Create						Random Create					
	Size:Chunk Size	Per Char		Block		Rewrite		Per Char		Block				Num Files	Create		Read		Delete		Create		Read		Delete	
		K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU	/sec	% CPU		/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU
WD MyBook 1T 7200rpm eSATA	6544M	72359	98	84625	13	40130	8	77035	96	86323	10	146.8	0	16	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++
Dell optiplex 160GB Serial ATA II 3Gb/s 10000rpm	6544M	67384	95	79145	17	41923	9	75939	95	95430	11	445.8	0	16	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++
zimbu RAID5	8G	57018	96	47356	13	17363	1	15400	10	26249	0	221.0	0	10	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++
os11	1G	11916	27	23789	4	9638	0	12022	13	9749	0	208.5	0	10	8754	16	+++++	+++	+++++	+++	+++++	+++	+++++	+++	+++++	+++
zimbu VM	1G	19984	38	28471	5	11686	0	41000	49	143696	1	663.2	0	100	46287	82	+++++	+++	36657	69	51194	96	+++++	+++	5336	

Tekst utskrift

```
Version  1.03c       ------Sequential Output------ --Sequential Input- --Random-
                     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
WD MyBook 1T  6544M 72359  98 84625  13 40130   8 77035  96 86323  10 146.8   0
Dell optiplex 6544M 67384  95 79145  17 41923   9 75939  95 95430  11 445.8   0
zimbu RAID5      8G 57018  96 47356  13 17363   1 15400  10 26249   0 221.0   0
os11             1G 11916  27 23789   4  9638   0 12022  13  9749   0 208.5   0
zimbu VM         1G 19984  38 28471   5 11686   0 41000  49 143696   1 663.2   0
                     ------Sequential Create------ --------Random Create--------
                     -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
files:max:min         /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
WD MyBook 1T 720  16 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
Dell optiplex 16  16 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
zimbu RAID5       10 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
os11              10  8754  16 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
```

iostat

- Passiv ytelsesmåler for disker (ikke filsystem)
- Overvåker antall IO operasjoner over tid
- Installasjon: apt-get install sysstat
- Eksempel på kjøring:
  iostat -k -d
  watch iostat -k -d

hdparm

- Aktiv ytelsesmåler for disk og diskcache:
  hdparm -tT /dev/disk
- Kan også vise info om disker og sette lav-nivå instillinger:
  hdparm /dev/disk
  - readahead
    hdparm -a tall /dev/disk
  - spindown og power-management
    hdparm -B tall -S tall /dev/disk

Måle en webservers ytelse

Hvilke variabler er vi interessert i?

- Response time - Hvor raskt vi får en pakke tilbake
- Connection rate - Hvor mange nye forbindelser man kan opprette per sekund
- Reply rate - Hvor raskt vi får siden vi ber om
- Timeouts - Hvor ofte den timer ut
- Båndbredde brukt
- Variasjon

Enkel tidstest

- Man kan bruke wget kommandoen:
  time wget -t 2 -T 5 -p -q <URL>
- Vil ta tiden på å laste ned hele siden med alle GET requests

httperf

- Et kraftig verktøy for å måle ytelse på en webserver
- Aktiv test, der formålet er å presse serveren til det ytterste
  - Bør ikke gjøres på et produksjonssystem
- Komplisert å lære seg, mange opsjoner

httperf eksempel

- Installasjon:
  `apt-get install httperf`
  - Hent siden /index.php 3 ganger per HTTP forbindelse.
  - 10 forbindelser i sekundet.
  - 1500 forbindelser totalt.
  - Tillat 4 sekunder før du får en timeout

```
httperf --rate 10 --num-conns 1500
--num-calls 3 --timeout 4 --server
<maskin> --hog --uri «/index.php»
```

Resultat

Maximum connect burst length: 1

Oversikt

Total: connections 1500 requests 4313 replies 4020 test-duration 162.734 s

Connection rate: 9.2 conn/s (108.5 ms/conn, <=160 concurrent connections)
Connection time [ms]: min 468.9 avg 5727.2 max 32245.5 median 517.5 stddev 7971.1
Antall forbindelser
Connection time [ms]: connect 45.3
Connection length [replies/conn]: 2.682

og hastighet

Request rate: 26.5 req/s (37.7 ms/req)
Request size [B]: 76.0

Reply rate [replies/s]: min 0.0 avg 25.1 max 46.8 stddev 10.4 (32 samples)
Respons
Reply time [ms]: response 1913.7 transfer 131.2
Reply size [B]: header 196.0 content 573615.0 footer 1.0 (total 573812.0)
Reply status: 1xx=0 2xx=3673 3xx=0 4xx=0 5xx=347

Systemverdier

CPU time [s]: user 19.09 system 143.40 (user 11.7% system 88.1% total 99.9%)
Net I/O: 13844.5 KB/s (113.4*10^6 bps)

Antall feil

Errors: total 294 client-timo 0 socket-timo 1 connrefused 0 connreset 293
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0

Utfordringer

- Hvordan måler du ytelsen på et diger tjeneste fra én enkelt maskin?
  - httperf vil gå tom for ressurser lenge før tjenesten
  - Man ser at «ytelsen faller», men feiltolker årsaken
- Hva om man har gjort en feil selv?
  - Lavt num_cons / rate forhold
  - Skrivefeil i URI (får kun 404 tilbake)

autobench

http://www.xenoclast.org/autobench/

- httperf satt i system
- Kjører httperf i flere runder med økende belastning
- Kan kjøres distribuert (veldig praktisk mot kraftige clustre)
- Utskrift kan brukes direkte mot gnuplot for å generere grafer

Autobench installasjon
på Debian / Ubuntu

- Installer httperf:
  `apt-get install httperf`
- Last ned autobench:
  wget http://www.xenoclast.org/autobench/downloads/debian/autobench_2.1.2_i386.deb
- Installer:
  `dpkg -i autobench_2.1.2_i386.deb`

eksempel

- En maskin mot en enkelt webside:

```
autobench --single_host --host1 mywebserer
--uri1 /run.php --low_rate 20 --high_rate 200
--rate_step 20 --num_call 10 --num_conn 5000
--timeout 5 --file output.tsv
```

Forklaring

- --host1 maskin, --uri1 /run.php
  tester url'en http://maskin/run.php
- --low_rate 10, --high_rate 50
  Første test har 10 forbindelser i sekundet.
  Siste test har 50.
- --rate_step 10
  Øker med 10 forbindelser i sekundet for hver test

Forklaring forts.

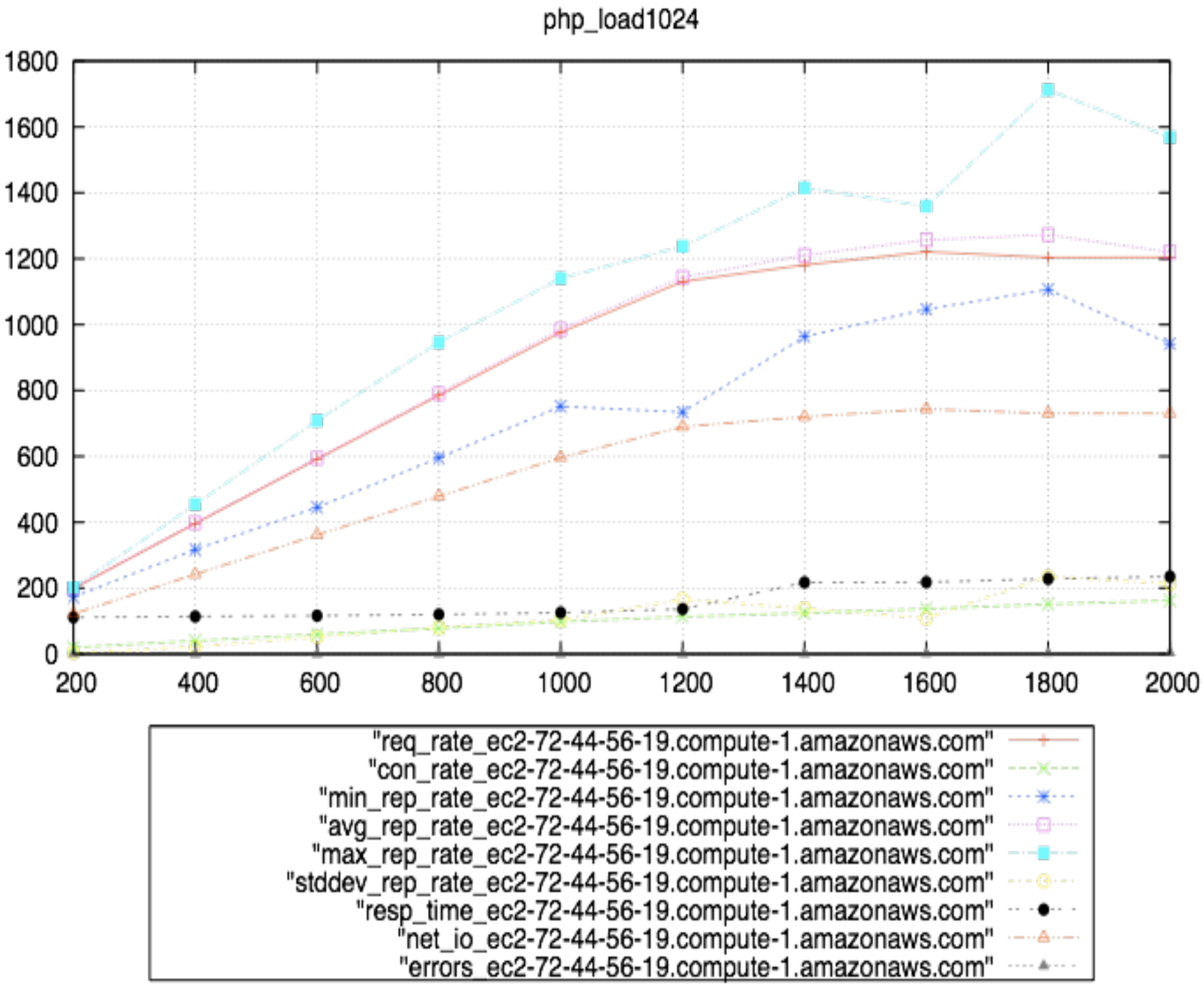
- --num_conn 5000
  5000 totale forbindelser ved hver test
- --num_call 5
  5 HTTP GET requests per forbindelse.
  Totalt per test: num_call * num_conn = 25000
- --timeout 5
  Mer en 5 sekunder på å svare blir logget som timeout

Analysere utskrift

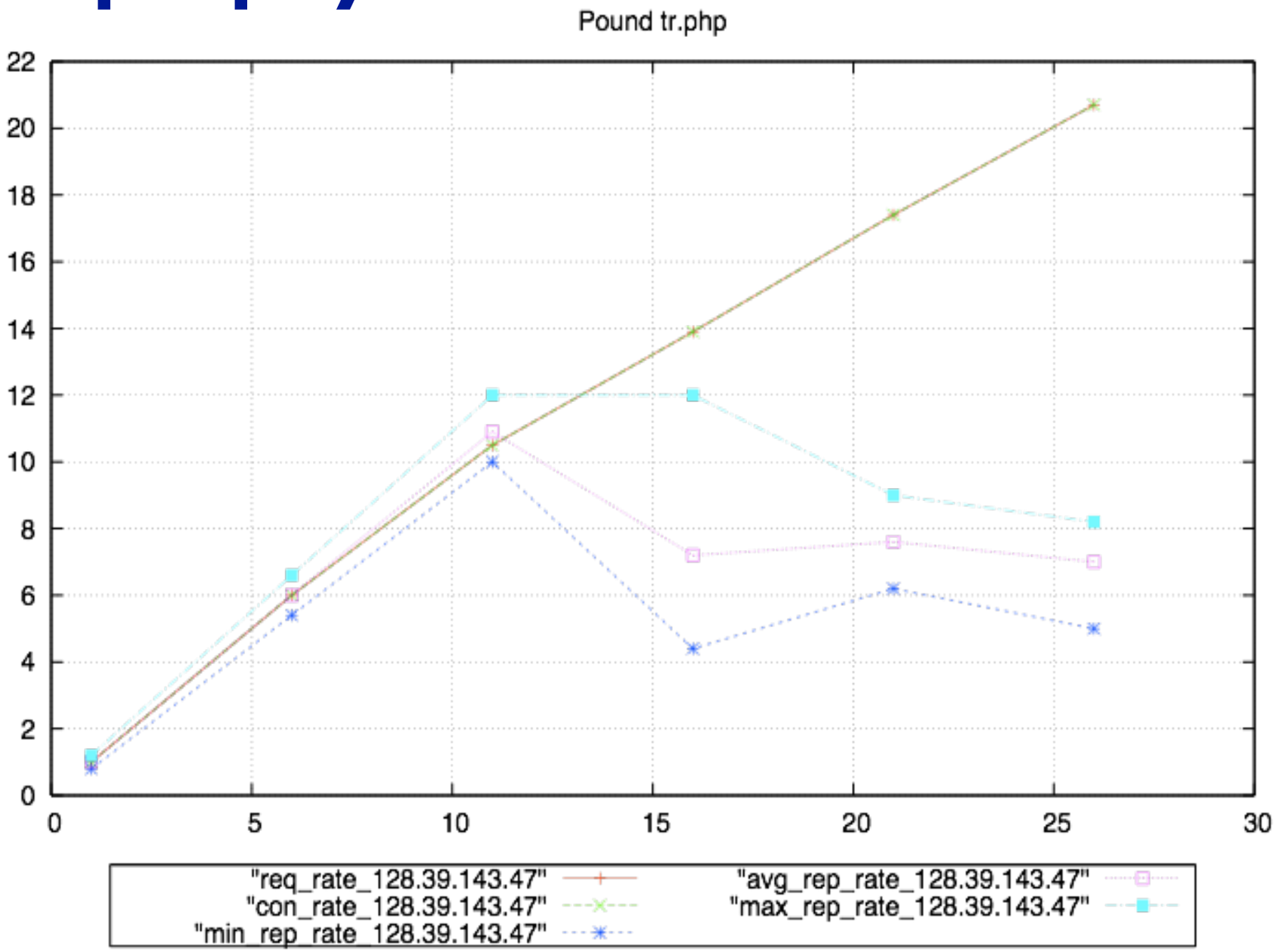
- Man kan lese outskrift-filen ved hjelp av cat eller less
- Det enkleste er å bruke gnuplot til å lage en graf:

```
apt-get install gnuplot gawk
bench2graph outfile.tsv outfile.eps
# bench2graph outfile.tsv outfile.eps 1,2,3,4,5,6
ps2pdf outfile.eps outfile.pdf
```

autobench plot eksempel



tr.php ytelse - ved lastbalansering




```
autobench --single_host --host1 128.39.143.47 --uri1 /tr.php      \
          --low_rate 1 --high_rate 30 --rate_step 5 --num_call 1 \
          --num_conn 500 --timeout 5 --file pound.tsv
```