ASL Exercise 3

Measuring a baseline

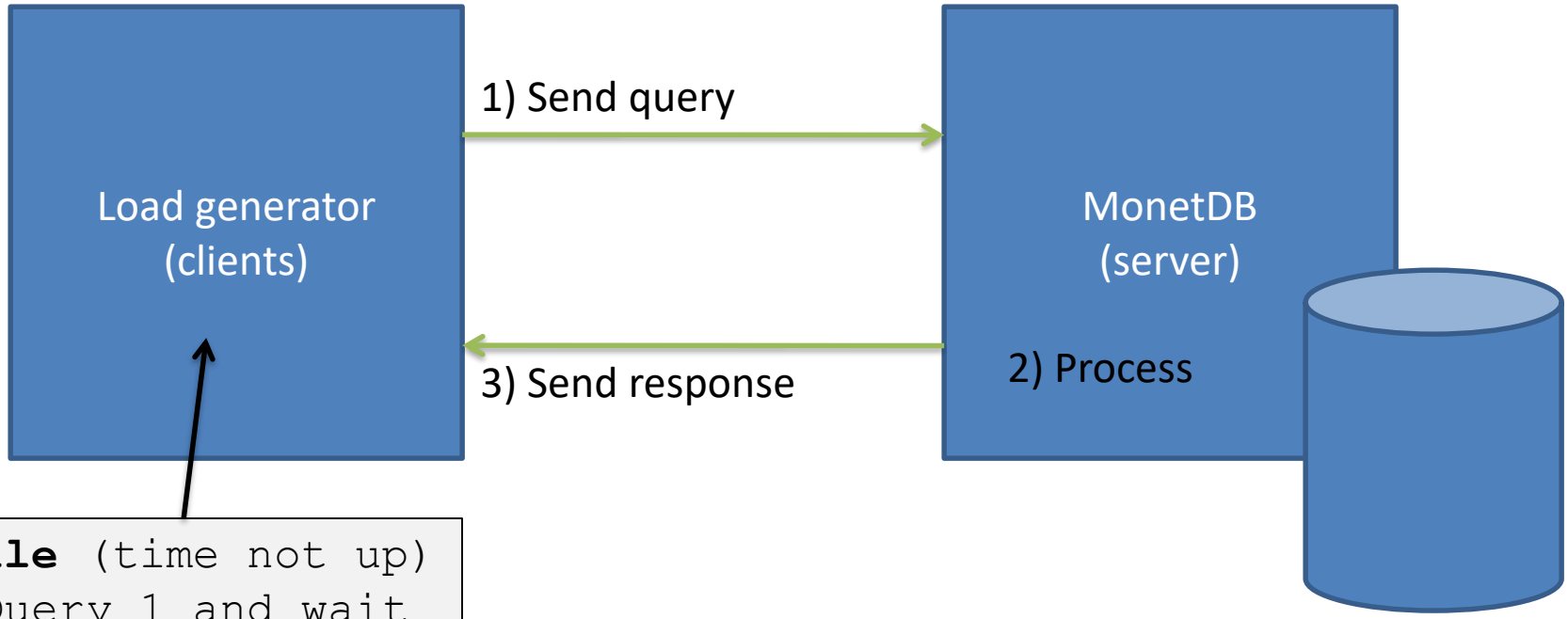
System under test

- MonetDB
  - Main-memory database (reads data off disk, keeps it in memory)
  - Best for analytical queries (no updates, etc.)
- Our clients
  - Simple scripts that run three types of queries from TPC-H
  - Log the output and runtime of each query

Testing methodology

- Experiment length
  - 4 minutes (without warmup and cooldown)
  - 3 repetitions
- Number of clients
  - Between 1 and 20
- MonetDB
  - Different multi-threading setups
- Machines
  - Two physical machines
  - 16 cores each (32 hyperthreads)

Testing setup




```
while (time not up)
  Query 1 and wait
  Query 2 and wait
  Query 3 and wait
end
```

$T_{network} << T_{processing}$

Collecting results

- mtX/ – X threads in MonetDB
  - pY/ – Y parallel clients
    - cZ/ – Data output by client number Z
      - rW/ – Repetition number W
        - dump.dat

```
…
+---------------------------+---------------------------+--------+-----------------------------------------+
| supp_nation               | cust_nation               | l_year | revenue                                 |
+===========================+===========================+========+=========================================+
| FRANCE                    | GERMANY                   |   1995 |                              481994.2376 |
| FRANCE                    | GERMANY                   |   1996 |                              629797.4221 |
| GERMANY                   | FRANCE                    |   1995 |                              581829.6063 |
| GERMANY                   | FRANCE                    |   1996 |                              642056.6361 |
+---------------------------+---------------------------+--------+-----------------------------------------+
4 tuples (639.316ms)
…
```

Script to transform results

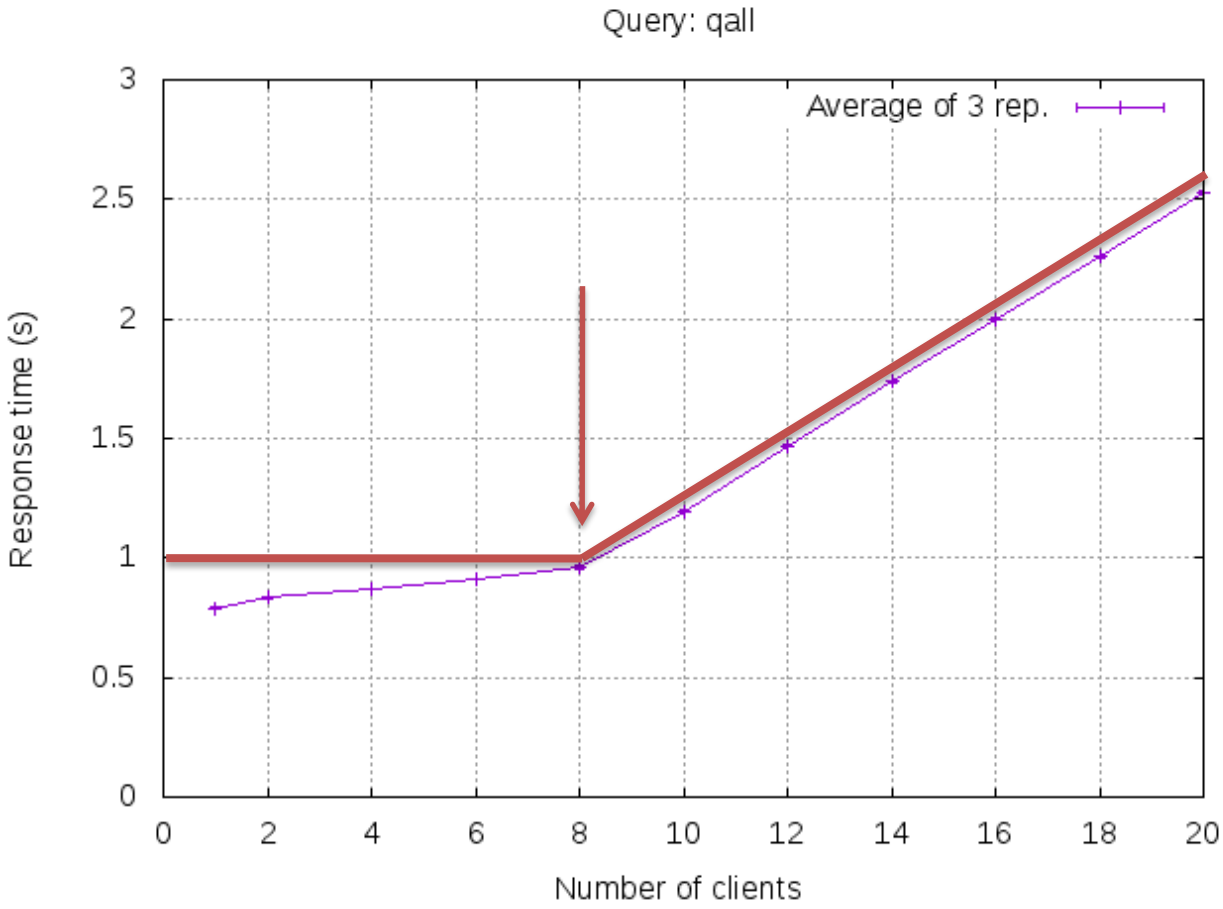
- Response times
  - Collect all response times per repetition
  - Collect response times per query type per repetition
  - Compute average and standard deviation
- Throughput
  - Count returned queries from all clients / 4 minutes
  - Compute average and standard deviation

Results are stored in one large file

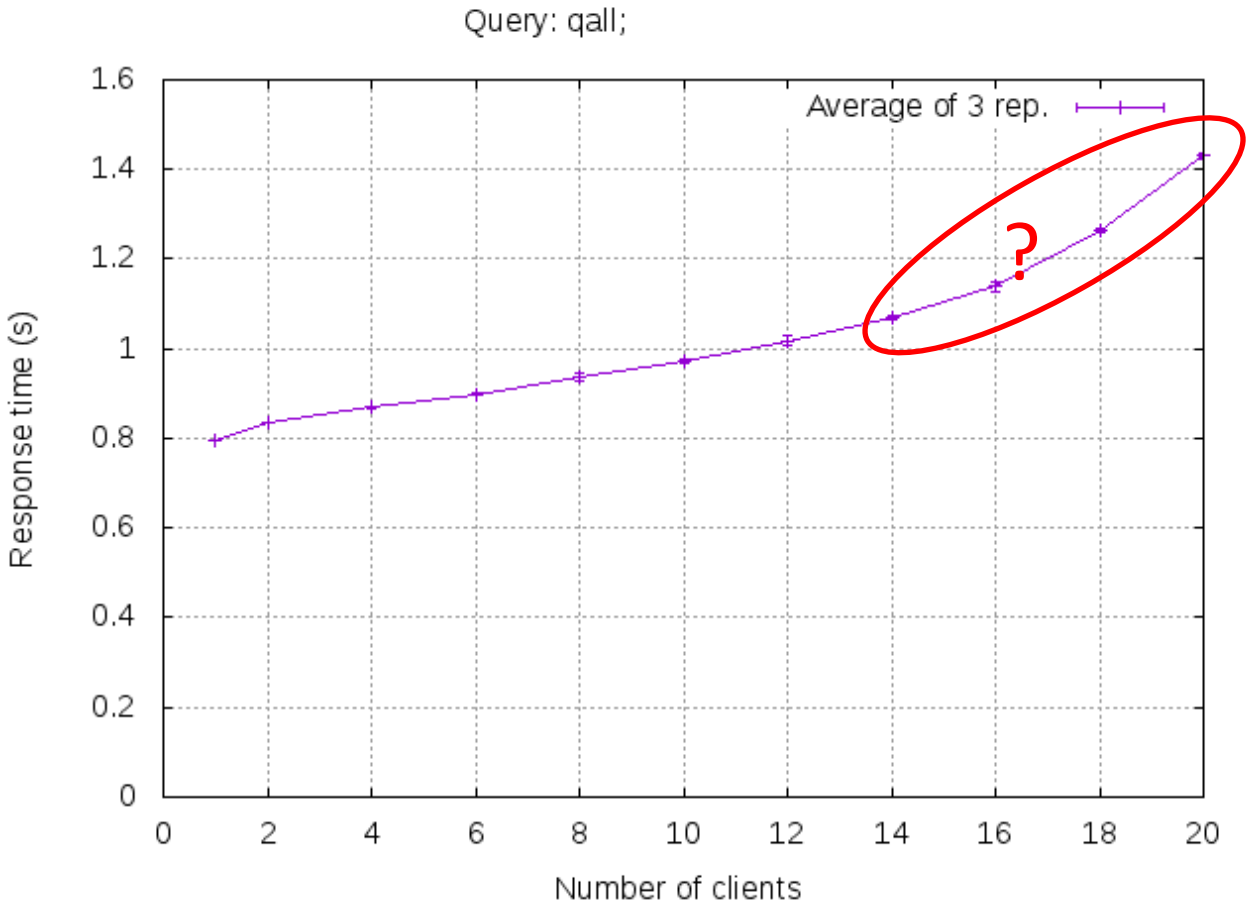
- Print one line per each experiment
- Many columns (average, stdev, etc.)
- Filter this file as needed to plot

mt	p	q	avg-r1	stdev-r1	avg-r2	stdev-r2	avg-r3	stdev-r3	avg-all	stdev-all	…
mt4	1	qall	0.77802	0.731165	0.788315	0.742658	0.787219	0.743625	0.784485	0.739148	
mt4	1	q1	0.350329	0.00207444	0.350596	0.00361263	0.349961	0.0014505	0.350295	0.00255444	
mt4	1	q2	1.80712	0.0118952	1.83383	0.0181015	1.8341	0.0111402	1.82493	0.0189255	
mt4	1	q3	0.176617	0.00424576	0.180513	0.0123542	0.177596	0.00372298	0.178229	0.00798717	
mt4	2	qall	0.832905	0.793062	0.829988	0.791179	0.827713	0.788787	0.830193	0.791006	
mt4	2	q1	0.362152	0.00968525	0.361454	0.0111415	0.359659	0.00918448	0.361084	0.0100902	
mt4	2	q2	1.94935	0.0397311	1.94339	0.0500328	1.93789	0.0490977	1.94353	0.0467647	
mt4	2	q3	0.18721	0.0135998	0.185117	0.0144703	0.18559	0.0132282	0.185971	0.0138031	
mt4	4	qall	0.861197	0.825381	0.868766	0.833306	0.867146	0.833964	0.865687	0.830879	
mt4	4	q1	0.372064	0.0123012	0.374679	0.0141387	0.371123	0.0119848	0.37262	0.012929	
…											

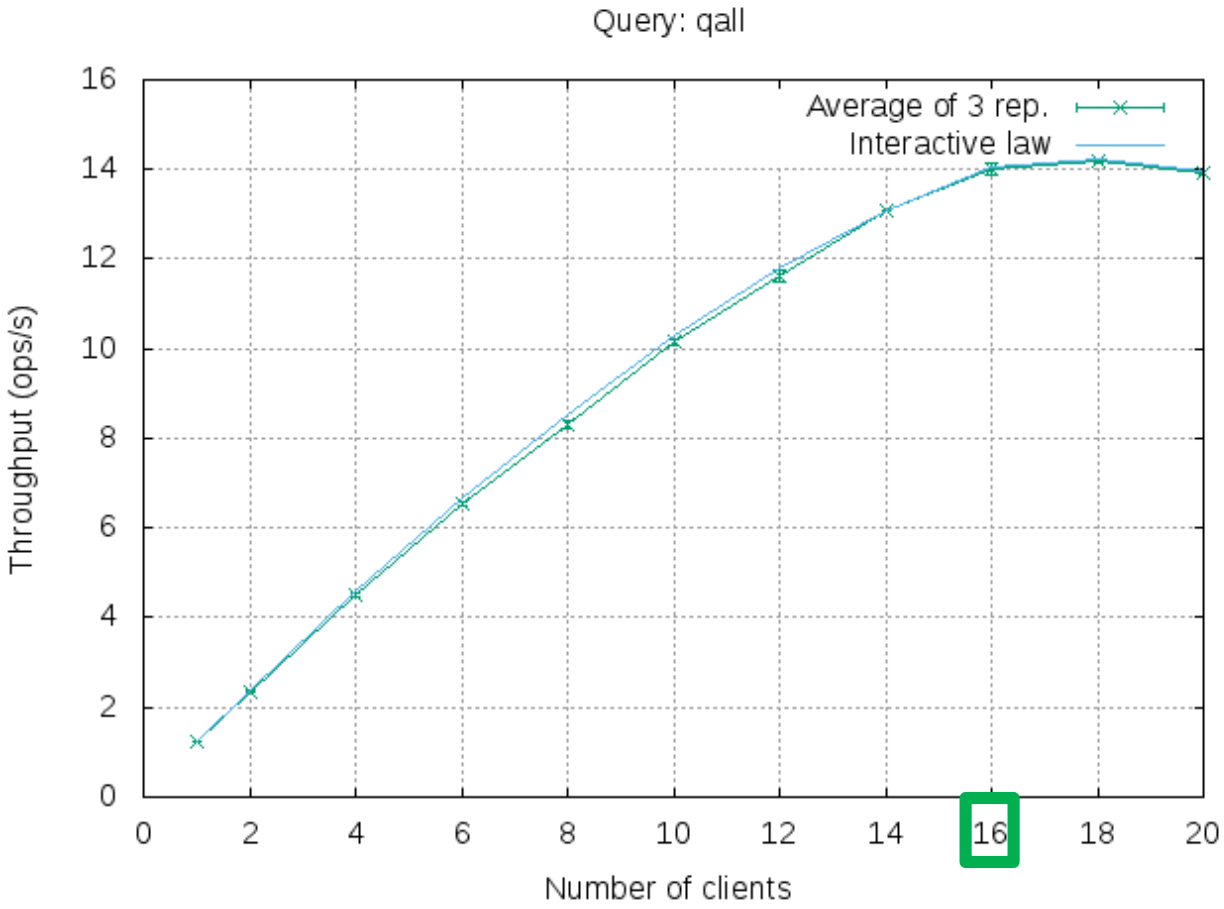
How many threads in MonetDB?



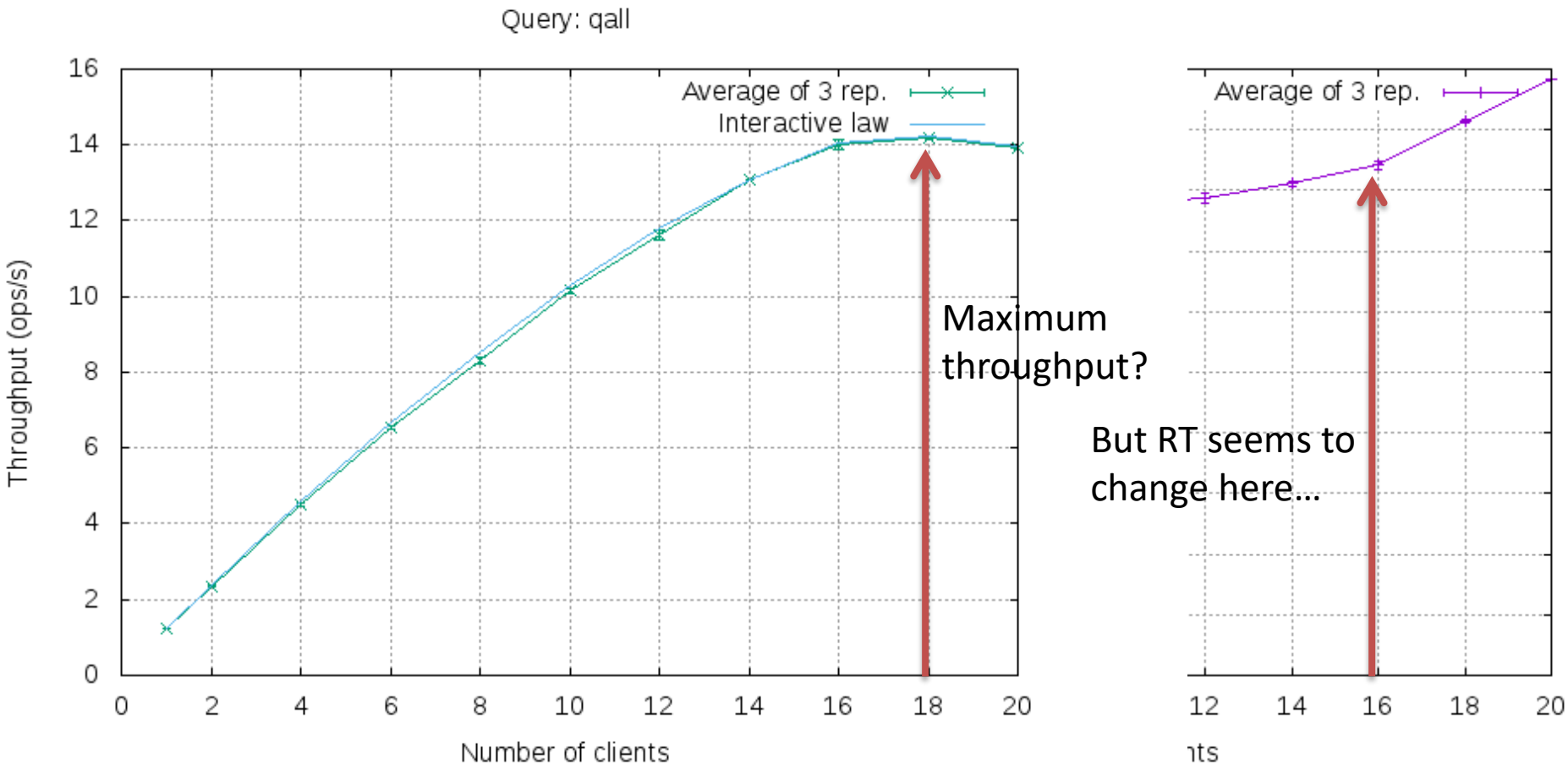
And now?


Query: qall;
Average of 3 rep.
Response time (s)
Number of clients
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
2
4
6
8
10
12
14
16
18
20
?

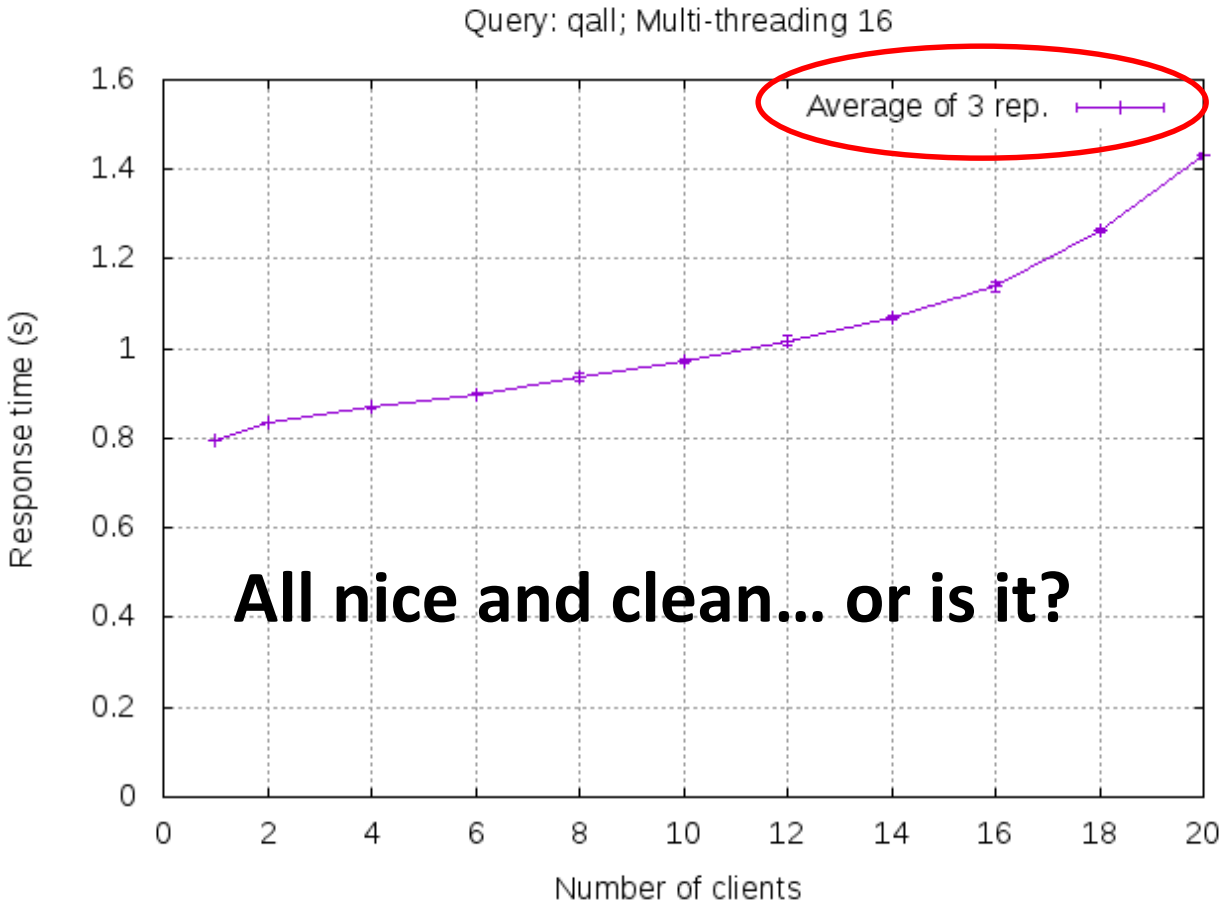
Maybe if we look at the TPUT graph?



Maximum throughput?

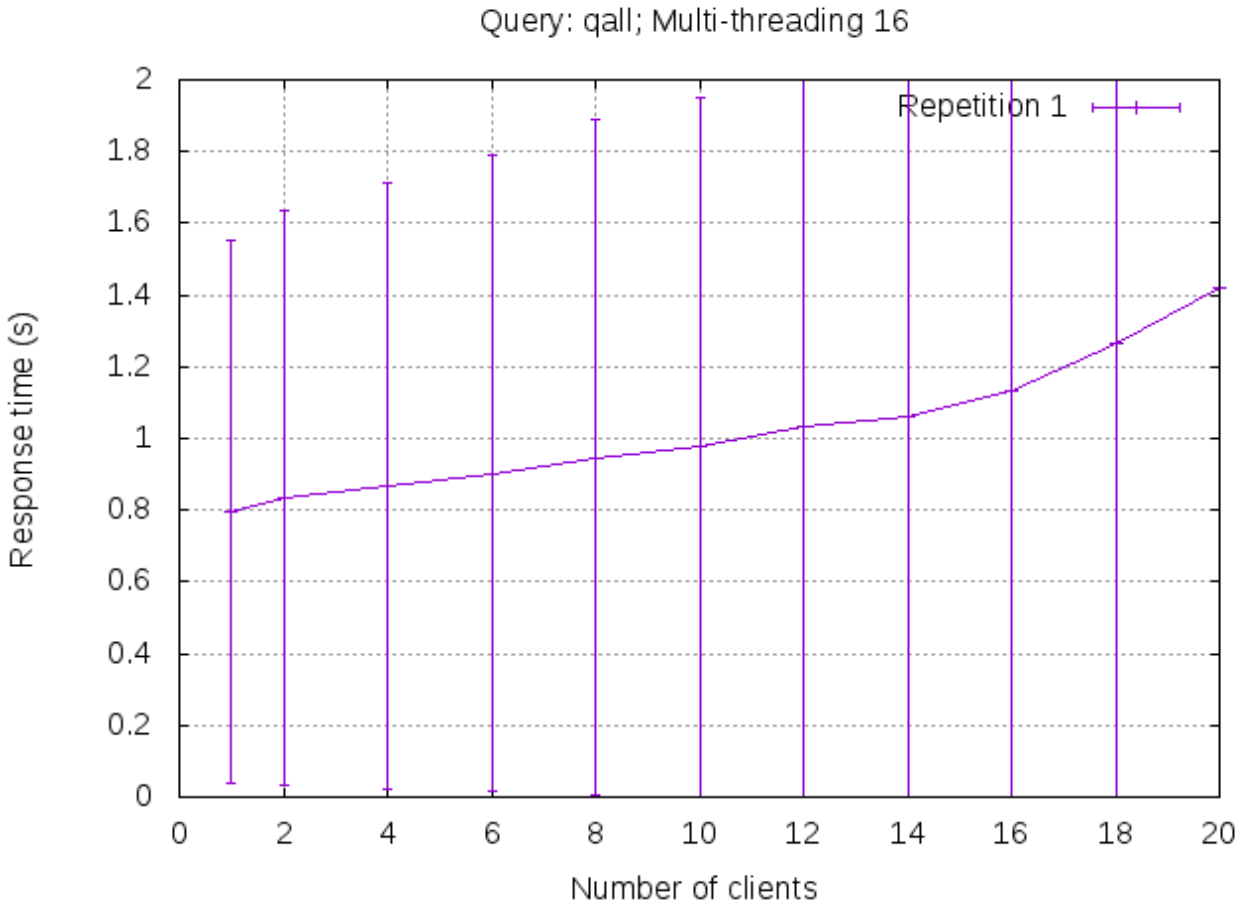


A word on standard deviation



A word on standard deviation (II)

Let's see AVG and STDEV inside a repetition…



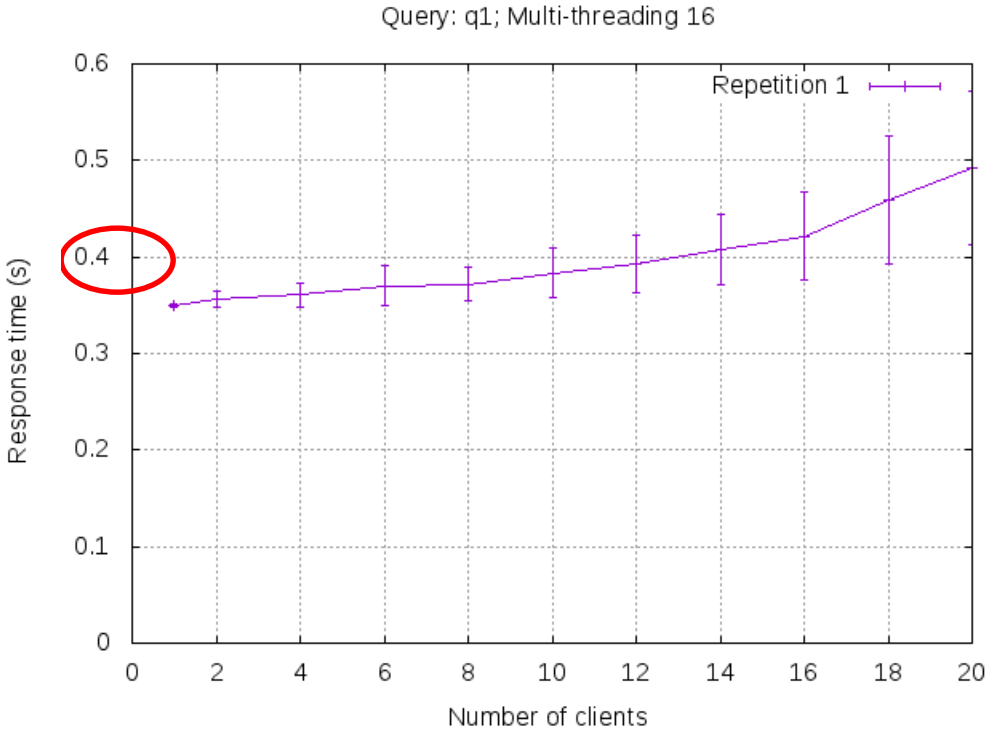
What happened to our STDEV?

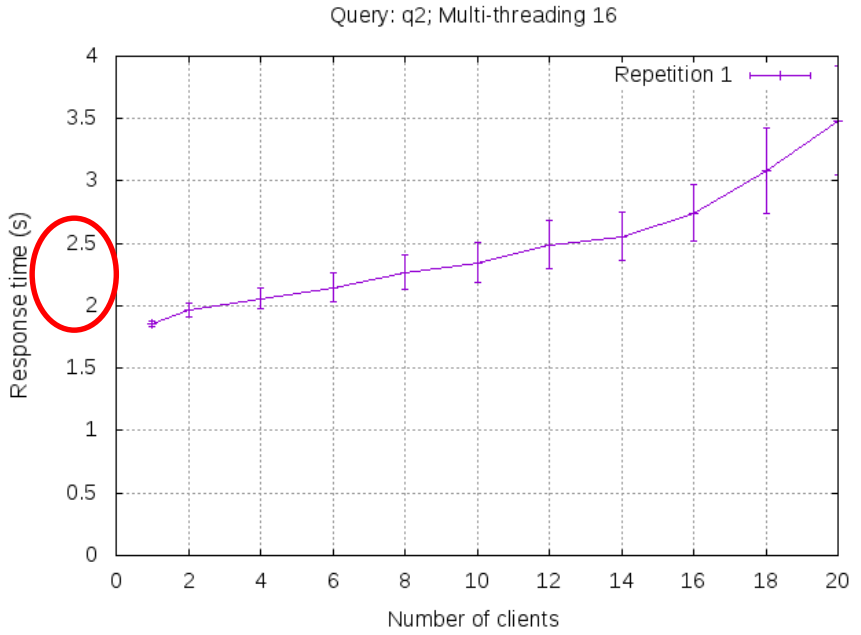
- What could be the reason?

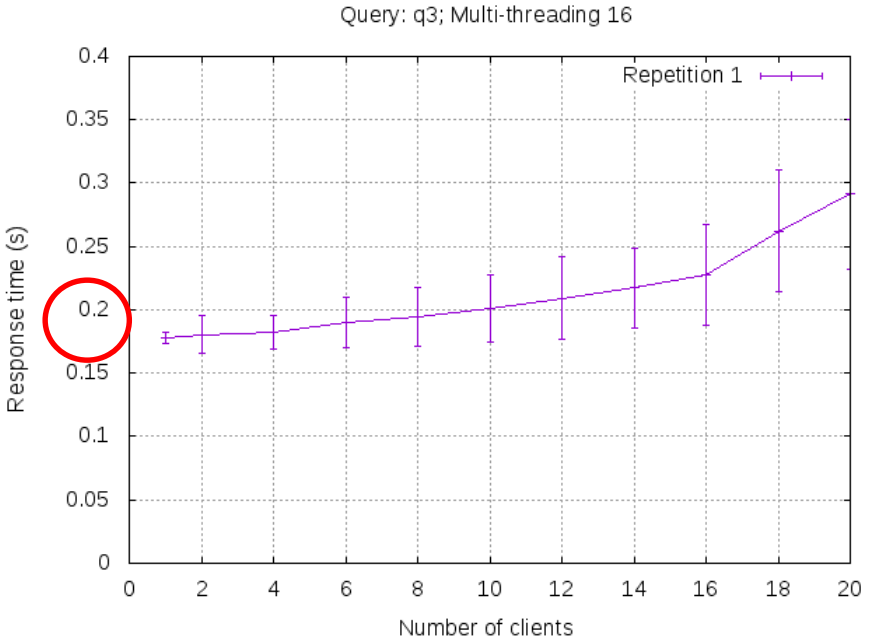
What happened to our STDEV?

- What could be the reason?
- Recall: Each client sends three types of queries…
- What if their response times are different?

RT by Query type



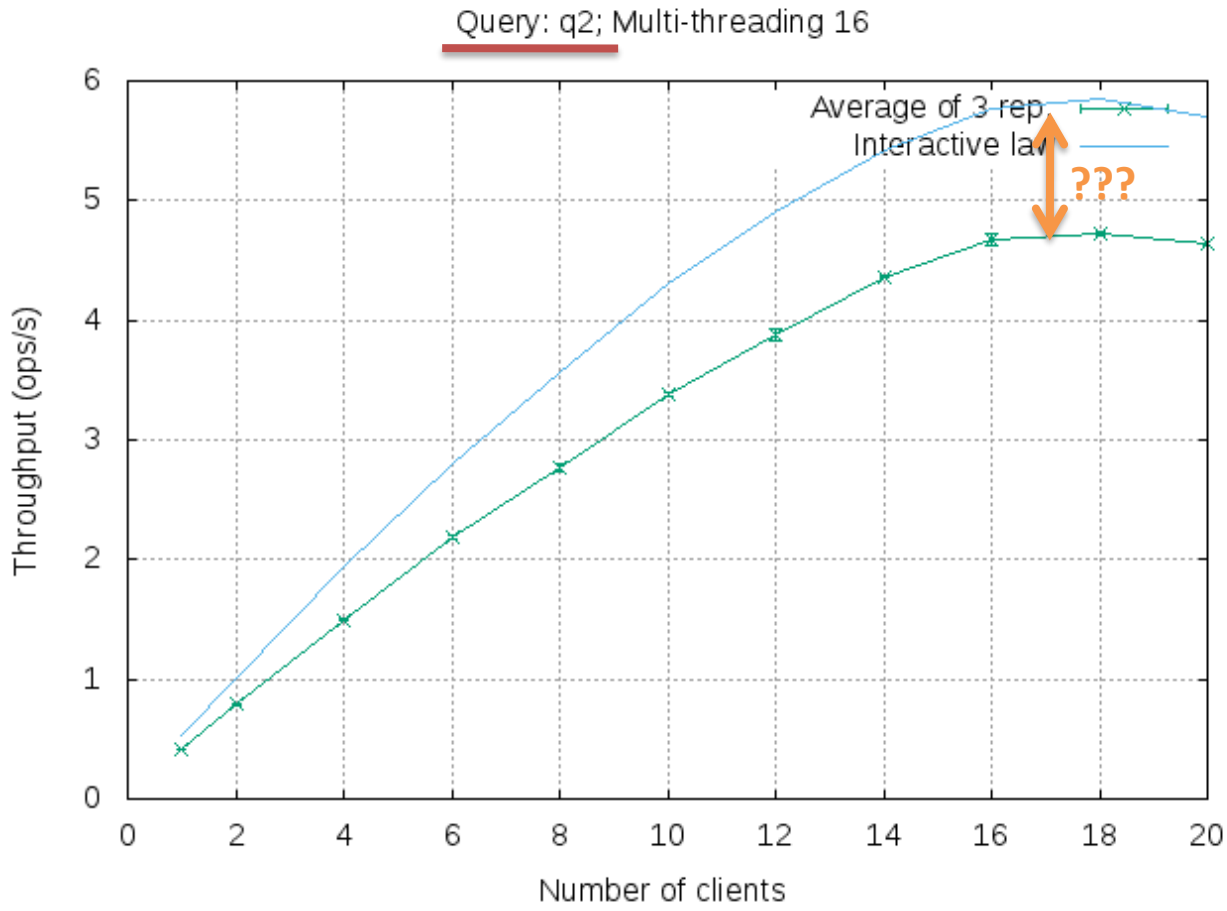




If we average over three classes of queries, the standard deviation is going to be high

Interactive law

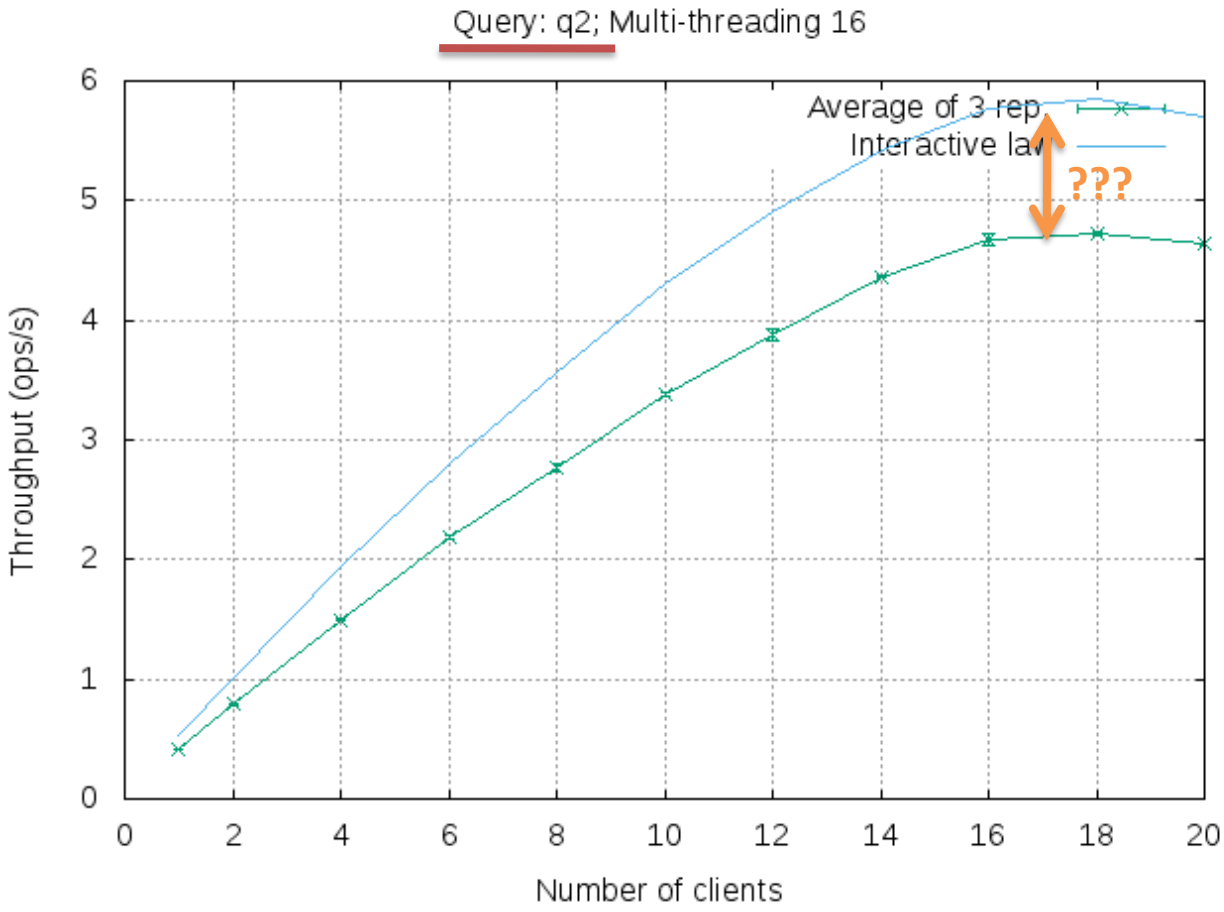
TPUT = 1/RT * #Clients



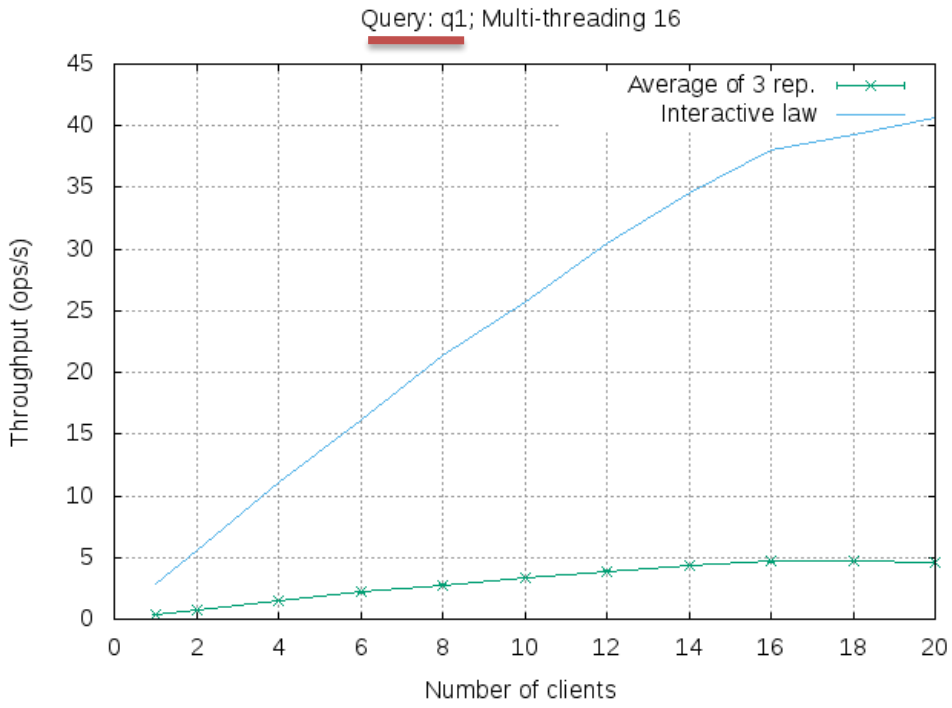
Interactive law (II)

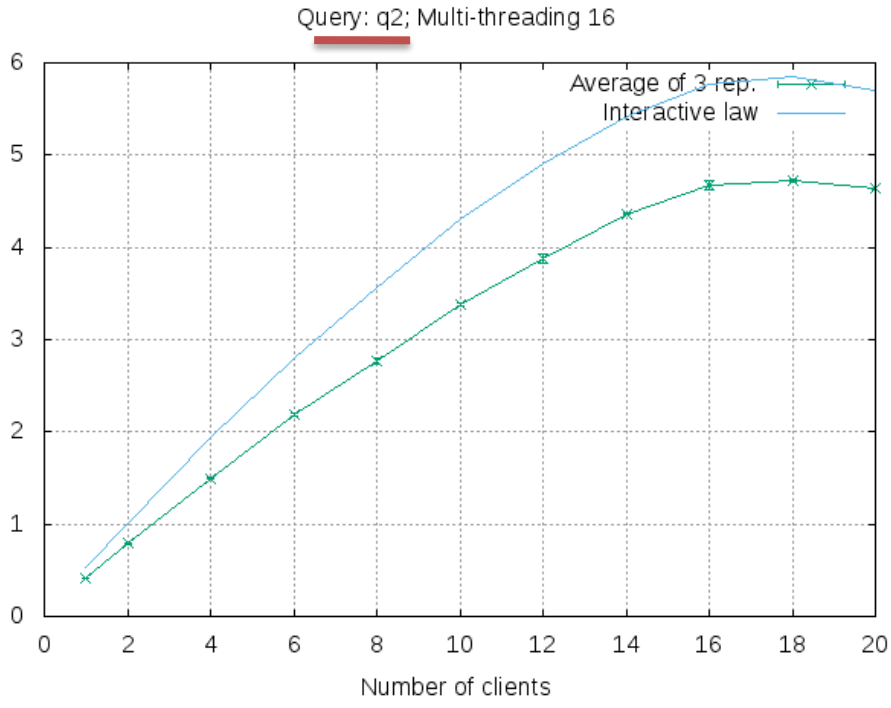
TPUT = 1/RT * #Clients

- But we don't only run Q2!
- If we would run Q2 alone we would get the blue line. The green line is 1/3 the global throughput!



Interactive law (III)





Response time ~ 0.4s at 16 clients

Response time ~ 2.75s at 16 clients

Actual throughput ~5 at 16 clients

Actual throughput ~5 at 16 clients

Because system throughput (Q1+Q2+Q3) ~14 at 16 clients

Plotting Best Practices

- Start axis at zero, try and keep same range for related graphs
- Label both axis, state units clearly
  - Use Ops/s not Ops/minute, and other “exotic” units
  - Instead of 12000000 use 1.2 million
- Caution with logarithmic scales on axis
- Include error bars!
- Make sure system configuration is easily found

- You will see more examples in the exercises…

Administrativa

- Azure:
  - Should be able to log and see money (if you sent request to us)
  - Send request ASAP if haven't done so yet!

- Next week:
  - Tips for avoiding bad design decisions in Java, related to networking and worker threads.