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

Load generator (clients) -----> MonetDB (server)

1) Send query
2) Process
3) Send response

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

$T_{\text{network}} \ll T_{\text{processing}}$
Collecting results

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

<table>
<thead>
<tr>
<th>supp_nation</th>
<th>cust_nation</th>
<th>l_year</th>
<th>revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRANCE</td>
<td>GERMANY</td>
<td>1995</td>
<td>481994.2376</td>
</tr>
<tr>
<td>FRANCE</td>
<td>GERMANY</td>
<td>1996</td>
<td>629797.4221</td>
</tr>
<tr>
<td>GERMANY</td>
<td>FRANCE</td>
<td>1995</td>
<td>581829.6063</td>
</tr>
<tr>
<td>GERMANY</td>
<td>FRANCE</td>
<td>1996</td>
<td>642056.6361</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>mt</th>
<th>p</th>
<th>q</th>
<th>avg-r1</th>
<th>stdev-r1</th>
<th>avg-r2</th>
<th>stdev-r2</th>
<th>avg-r3</th>
<th>stdev-r3</th>
<th>avg-all</th>
<th>stdev-all</th>
</tr>
</thead>
<tbody>
<tr>
<td>mt4</td>
<td>1</td>
<td>qall</td>
<td>0.77802</td>
<td>0.731165</td>
<td>0.788315</td>
<td>0.742658</td>
<td>0.787219</td>
<td>0.743625</td>
<td>0.784485</td>
<td>0.739148</td>
</tr>
<tr>
<td>mt4</td>
<td>1</td>
<td>q</td>
<td>0.350329</td>
<td>0.00207444</td>
<td>0.350596</td>
<td>0.00361263</td>
<td>0.349961</td>
<td>0.0014505</td>
<td>0.350295</td>
<td>0.00255444</td>
</tr>
<tr>
<td>mt4</td>
<td>1</td>
<td>q2</td>
<td>1.80712</td>
<td>0.0118952</td>
<td>1.83383</td>
<td>0.0181015</td>
<td>1.8341</td>
<td>0.0111402</td>
<td>1.82493</td>
<td>0.0189255</td>
</tr>
<tr>
<td>mt4</td>
<td>1</td>
<td>q3</td>
<td>0.176617</td>
<td>0.00424576</td>
<td>0.180513</td>
<td>0.0123542</td>
<td>0.177596</td>
<td>0.00372298</td>
<td>0.178229</td>
<td>0.00798717</td>
</tr>
<tr>
<td>mt4</td>
<td>2</td>
<td>qall</td>
<td>0.832905</td>
<td>0.793062</td>
<td>0.829988</td>
<td>0.791179</td>
<td>0.827713</td>
<td>0.788787</td>
<td>0.830193</td>
<td>0.791006</td>
</tr>
<tr>
<td>mt4</td>
<td>2</td>
<td>q</td>
<td>0.362152</td>
<td>0.00968525</td>
<td>0.361454</td>
<td>0.0111415</td>
<td>0.359659</td>
<td>0.00918448</td>
<td>0.361084</td>
<td>0.0100902</td>
</tr>
<tr>
<td>mt4</td>
<td>2</td>
<td>q2</td>
<td>1.94935</td>
<td>0.0397311</td>
<td>1.94339</td>
<td>0.0500328</td>
<td>1.93789</td>
<td>0.0490977</td>
<td>1.94353</td>
<td>0.0467647</td>
</tr>
<tr>
<td>mt4</td>
<td>2</td>
<td>q3</td>
<td>0.18721</td>
<td>0.0135998</td>
<td>0.185117</td>
<td>0.0144703</td>
<td>0.18559</td>
<td>0.0132282</td>
<td>0.185971</td>
<td>0.0138031</td>
</tr>
<tr>
<td>mt4</td>
<td>4</td>
<td>qall</td>
<td>0.861197</td>
<td>0.825381</td>
<td>0.868766</td>
<td>0.833306</td>
<td>0.867146</td>
<td>0.833964</td>
<td>0.865687</td>
<td>0.830879</td>
</tr>
<tr>
<td>mt4</td>
<td>4</td>
<td>q1</td>
<td>0.372064</td>
<td>0.0123012</td>
<td>0.374679</td>
<td>0.0141387</td>
<td>0.371123</td>
<td>0.0119848</td>
<td>0.37262</td>
<td>0.012929</td>
</tr>
</tbody>
</table>
How many threads in MonetDB?
And now?
Maybe if we look at the TPUT graph?
Maximum throughput?

Query: qall

Average of 3 rep. Interactive law

Maximum throughput?

But RT seems to change here...
A word on standard deviation

All nice and clean... or is it?

Query: qall; Multi-threading 16

Response time (s)

Number of clients

Average of 3 rep.
A word on standard deviation (II)

Let’s see AVG and STDEV inside a repetition...

![Graph showing response time vs number of clients for Query: qall; Multi-threading 16. The graph illustrates the trend of response time as the number of clients increases, with error bars indicating variability.](image)
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?
If we average over three classes of queries, the standard deviation is going to be high.
Interactive law

TPUT = 1/RT * #Clients

Query: q2; Multi-threading 16

Average of 3 rep
Interactive law

Throughput (ops/s)

Number of clients
Interactive law (II)

$$TPUT = \frac{1}{RT} \times \#\text{Clients}$$

• 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
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.