Q-Cop: Avoiding Bad Query Mixes to Minimize Client Timeouts Under Heavy Loads

Sean Tozer, Tim Brecht, and Ashraf Aboulnaga

http://cs.uwaterloo.ca/~brecht

ICDE 2010
Query Mixes

Client 1 ➔ Web Server ➔ Appl’n Server ➔ DBMS ➔ Client 1
Client 2 ➔ Web Server ➔ Appl’n Server ➔ DBMS ➔ Client 2
Client N ➔ Web Server ➔ Appl’n Server ➔ DBMS ➔ Client N

Browse ➔ Buy ➔ Browse

Queries
- Fixed Number
- Known

Current mix: 2 Browse, 1 Buy
Query Mix Example

• Assume three query types A, B, and C
• Query mixes with 3 query instances:
  10 possible mixes
  AAA, AAB, AAC, ABB, ABC, ACC,
  BBB, BBC, BCC, and CCC

• KEY: execution times for query type change with mix.

  Consider only type A: avg execution time in different mixes

  AAA, AAB, AAC, ABB, ABC, ACC
  3.1  4.1  2.0  6.9  3.5  1.0
Query Mix Example

• Assume three query types A, B, and C
• Query mixes with 3 query instances:
  10 possible mixes
  AAA, AAB, AAC, ABB, ABC, ACC, BBB, BBC, BCC, and CCC

• KEY: execution times for query type change with mix
  Consider only type A: avg execution time in different mixes

<table>
<thead>
<tr>
<th></th>
<th>AAA</th>
<th>AAB</th>
<th>AAC</th>
<th>ABB</th>
<th>ABC</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>6.9</td>
<td></td>
<td></td>
<td>3.5</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Min</td>
<td></td>
<td></td>
<td>2.0</td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Bad Mix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good Mix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why Heavy Loads Matter

• Users/Clients do not wait forever
  – They and their browsers **time out**
• Cannot over provision for peak load
  – often 1 to 2 orders of magnitude higher than average
• Thanksgiving, Xmas, Slashdot, Oprah Effect
  – Unserviced requests means lost revenue ($$$$
• IDLE server uses ~ 50% of peak power
  – **GREEN** servers must run at higher utilization
  – Less headroom for load increases
Query Mix

client software: httpperf TPC-W like browsing workload

Client 1

2 x Dual Core
2.0 GHz AMD,
10 GB RAM

Linux 2.6.22.18-0.2

Client 2

Web Server

Apache 2.2.9

Tomcat 4.1.37

Appl’n Server

Client N

DBMS

Derby 10.4.2.0
5.2 GB TPC-W
Why Heavy Loads Matter

TPC-W Benchmark with client timeout = 30 secs.
Why Heavy Loads Matter

TPC-W Benchmark with client timeout = 30 secs.
Why Heavy Loads Matter

TPC-W Benchmark with client timeout = 30 secs.
Why Heavy Loads Matter

GOAL -- Reduce Unsuccessful Requests: service more clients, make more $
Admission Control

• Reject some queries to reduce timeouts and help other queries
  – Timeouts waste resources and hurt other queries

• Q-Cop : use query mix model, predict query execution times, reject queries likely to timeout before finishing
Admission Control: Approaches

- **(NoAC)**: No Admission Control (do nothing)
- **(MPL)**: Multi-Programming Limit
  - Limit the number of simultaneous queries
  - Already exists in many systems
- **(TYPE)**: Use info about query type
  - [Elnikety et al 04]
- **(Q-Cop)**: Use Query Mix Model (QMM) [Our Approach]
  - Use info about query type and other queries running
  - Reject queries likely to time out (timeout = 30 secs)
Predicting Execution Time

• Resource utilization
  – No significant network or disk I/O (DB fits in memory)
  – Examine CPU utilization
    • Higher CPU utilization → Longer execution times?
CPU Utilization vs Response Time

Instead use info about query mix to build model
Experimentally Building a QMM

- TPC-W Browsing Mix Queries:
  6 Non Trivial Query Types (> 1 ms execution time)
  Author Search, Title Search, Best Sellers (Setup),
  Best Sellers (Main), New Products, Related Products

Run all mixes, collect stats for each query type and mix,
build statistical model to predict execution time.

NOT POSSIBLE – TOO MANY MIXES

Number of Mixes = \((MPL + T - 1)\) choose MPL

For MPL = 20, T = 6  177,100 possible mixes
For MPL = 30, T = 6  1,623,160 possible mixes

Instead Use Sample of Query Mixes
Latin Hypercube Sampling (LHS)

- Latin Hypercube Sampling, generalizes Latin Square
- good coverage over possible mixes at different MPLs
- specify number of samples (1,000 in our case)
- create clients that repeatedly request each query in a mix
  - 1,000 query mixes, 70,000 query executions
- sample a small subset of the millions of possible mixes

Sample:
- each row
- each column

Random Sample  Latin Square Sample
Query Mixes: Best Sellers

![Graph showing execution time (sec) vs MPL (Max)](image-url)
Query Mixes: Best Sellers

![Graph showing execution time vs MPL with max and min lines.](image-url)
• Query mix makes a significant difference
Query Mixes: Best Sellers

- As MPL increases, mean execution times increase
Query Mixes: Best Sellers

- Good MPL depends on the mix of queries
Query Mixes: Best Sellers

Similar results for other query types [Tozer’s thesis]
Query Mix Model (QMM)

- Expected execution time of query type $i$, given a mix
- We use a simple linear model (others possible)
- $\text{Est}_i = (c_{i1} \times n_1) + (c_{i2} \times n_2) + \ldots + (c_{iN} \times n_N) + C_i$
  - $c_{ij}$ : time query of type $j$ adds to query of type $i$
  - $n_j$ : number of queries of type $j$
  - $C_i$ : estimated time of query type $i$ with no load
  - $N$ : number query types
- Any statistical tool can be used to derive coefficients
  (WEKA/Excel)
Using QMM for Admission Control

• When a query arrives use QMM to predict its execution time if it is added to the mix currently in the system
• Reject if predicted execution time > 29 sec
Different Approaches Compared

- **(NoAC)** No Admission Control (do nothing)
- **(MPL)** Multi-Programming Limit
  - Off-line: use live workload, run with MPL = 5, 10, 15, ...
  - On-line: use best MPL = 40 (no info about timeout)
- **(TYPE)** Info about query type
  - Off-line: use live workload, collect stats, build model
  - On-line: use model for query type, reject > 29 sec
- **(Q-Cop)** Query Mix Model
  - Off-line: use LHS, run subset of query mixes, collect stats, build model
  - On-line: use model for query mix, reject > 29 secs
Comparing Approaches

Queries Not Serviced (%) vs. Requests/sec

NoAC
Comparing Approaches

NoAC
MPL = 40

Queries Not Serviced (%)

Requests/sec

0 10 12 14 16 18 20 22 24 26 28 30
Comparing Approaches

Queries Not Serviced (%) vs Requests/sec

- NoAC
- MPL = 40
- TYPE
Comparing Approaches

- NoAC
- MPL = 40
- TYPE
- Q-Cop

Queries Not Serviced (%) vs Requests/sec
## Comparing Approaches

### Graph

- **NoAC**
- **MPL = 40**
- **TYPE**
- **Q-Cop**

### Table

<table>
<thead>
<tr>
<th></th>
<th>NoAC</th>
<th>MPL = 40</th>
<th>TYPE</th>
<th>Q-Cop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg % Not Serviced</td>
<td>31.9</td>
<td>19.6</td>
<td>8.6</td>
<td>4.6</td>
</tr>
<tr>
<td>Q-Cop Reduction %</td>
<td>85.8</td>
<td>76.8</td>
<td>46.9</td>
<td>--</td>
</tr>
</tbody>
</table>
Rejected and Unsuccessful Requests

Queries Not Serviced

Requests/sec

TYPE (Rejected)
Rejected and Unsuccessful Requests

Queries Not Serviced

Requests/sec

TYPE (Rejected)

TYPE (Unsuccessful)
Rejected and Unsuccessful Requests

Queries Not Serviced

Requests/sec

TYPE (Rejected)
TYPE (Unsuccessful)

Timeouts
Rejected and Unsuccessful Requests

- TYPE (Rejected)
- TYPE (Unsuccessful)
- Q-Cop (Rejected)
Rejected and Unsuccessful Requests

- TYPE (Rejected)
- TYPE (Unsuccessful)
- Q-Cop (Rejected)
- Q-Cop (Unsuccessful)

Queries Not Serviced vs. Requests/sec
Rejected and Unsuccessful Requests

Q-Cop rejects same # of queries, has fewer timeouts
Summary of Q-Cop

• Uses query mix model to predict query execution times
• Reject queries that are likely to time out before completion
• Significantly reduces the number of unserviced requests
  – 47% fewer unserviced requests than next best approach
• Does not require a priori knowledge of the workload mix
  – only need to know different query types