Paper reviews
Chenghao Lyu
Ottertune -- DBMS Tuning is hard!

• Tuning a DBMS’s **configuration knobs** knobs for the application’s **workload** and **hardware** is critical for performance

• Tuning even on DBMS is hard...
Ottertune -- DBMS Tuning is hard!
Existing approaches

• Auto-tuner
  • DMBS-specific provided by certain vendors
  • Heuristics-based

• “Smarter” Auto-tuner
  • Support Multiple-DMBS but including more manual steps
  • Research-only
  • No data reuse
Ottertune

- Reuse historical performance data from tuning past DBMS deployments to tune new DBMS deployments
Overview

① User specifies the target objective
② Controller starts the first **observation period**
   • *Measures external metrics, then collects DBMS-specific internal metrics*
   • *Returns: metrics, current knob configuration*
③ Tuning manager saves the data and computes the next configuration
   • *Returns: next configuration, expected improvement*
④ Controller installs next configuration on DBMS
Assumption and Limitation

- All DBMS instances run on the **same hardware**
- Blacklist for knobs that should NOT be tuned
Data Recap

- Knob configurations
- Internal metrics
- External metrics

```json
// Internal Metrics
{
"global": {
"pg_stat_archiver": {
"archived_count": "0",
"failed_count": "0",
"stats_reset": "2017-11-10 10:59:47.397075-05"
},
"pg_stat_bgwriter": {
"buffers_alloc": "87670",
"buffers_backend": "81032",
"buffers_backend_fsync": "0",
"buffers_checkpoint": "33250",
"buffers_clean": "49590",
"checkpoint_sync_time": "19",
"checkpoint_write_time": "597851",
"checkpoints_req": "2",
"checkpoints_timed": "1277",
"maxwritten_clean": "325",
"stats_reset": "2017-11-10 10:59:47.397075-05"
}
},
"local": {
"table": {
"pg_stat_user_tables": {
"history": {
"analyze_count": "0",
"autoanalyze_count": "1",
"autovacuum_count": "0",
"last_autoanalyze": "2017-11-20 15:59:02.567618-05"
}
}
}
```
```
Workload Characterization

• Goal: find metrics that best characterize an application’s workload
• Method: Use the metrics stored in Ottertune’s data repository

Internal DBMS Metrics:

- Directly affected by the knobs’ settings
- Problem: redundancy
  - Same but different units
  - Highly correlated
- Solution: prune them
  - Factor analysis to capture correlation patterns
  - K-means to group correlated metrics

Buffer pool size is too small:

\[
\frac{\text{# buffer pool misses}}{\text{total # buffer pool requests}}
\]
Workload Characterization -- result

- Reduce the metrics for Mysql v5.6 by 93%
- Reduce the metrics for Postgres v9.3 by 82%
• Goal: identify which knobs affect the DBMS’s performance
• Method: feature selection technique to order knobs by importance

Knob Identification

Configuration Knobs:

- Knobs have varying degrees of impact on the DBMS’s performance
  - Some have high impact
  - Some have no impact
  - For many, it depends on the workload

- Problem: which knobs matter?
- Solution: Lasso to determine the order of importance
- Problem: how many to tune?
- Solution: Incremental knob selection to gradually increase the number of knobs tuned during a session
Data lineage

• Metrics selection
• Knob selection
• Workload Mapping
• Prediction Model
### 1.csv

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>$k_2$</th>
<th>...</th>
<th>$k_{100}$</th>
<th>$o_1$</th>
<th>$m_1$</th>
<th>...</th>
<th>$m_k$</th>
<th>...</th>
<th>$m_{300}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4</td>
<td>...</td>
<td>100</td>
<td>519</td>
<td>120</td>
<td>...</td>
<td>0.1</td>
<td>...</td>
<td>3000</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>...</td>
<td>80</td>
<td>700</td>
<td>80</td>
<td>...</td>
<td>3.5</td>
<td>...</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>12</td>
<td>...</td>
<td>77</td>
<td>200</td>
<td>50</td>
<td>...</td>
<td>0.5</td>
<td>...</td>
<td>700</td>
</tr>
</tbody>
</table>

### 2.csv

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>$k_2$</th>
<th>...</th>
<th>$k_{100}$</th>
<th>$o_1$</th>
<th>$m_1$</th>
<th>...</th>
<th>$m_k$</th>
<th>...</th>
<th>$m_{300}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>5</td>
<td>...</td>
<td>100</td>
<td>106</td>
<td>120</td>
<td>...</td>
<td>1.1</td>
<td>...</td>
<td>6000</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>...</td>
<td>90</td>
<td>740</td>
<td>80</td>
<td>...</td>
<td>4.3</td>
<td>...</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>10</td>
<td>...</td>
<td>77</td>
<td>200</td>
<td>50</td>
<td>...</td>
<td>0.2</td>
<td>...</td>
<td>600</td>
</tr>
</tbody>
</table>

### target.csv

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>$k_2$</th>
<th>...</th>
<th>$k_{100}$</th>
<th>$o_1$</th>
<th>$m_1$</th>
<th>...</th>
<th>$m_k$</th>
<th>...</th>
<th>$m_{300}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4</td>
<td>...</td>
<td>100</td>
<td>400</td>
<td>100</td>
<td>...</td>
<td>0.7</td>
<td>...</td>
<td>1200</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>...</td>
<td>80</td>
<td>680</td>
<td>60</td>
<td>...</td>
<td>2.2</td>
<td>...</td>
<td>1800</td>
</tr>
<tr>
<td>50</td>
<td>18</td>
<td>...</td>
<td>66</td>
<td>250</td>
<td>40</td>
<td>...</td>
<td>0.8</td>
<td>...</td>
<td>1200</td>
</tr>
</tbody>
</table>

### 50.csv

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>$k_2$</th>
<th>...</th>
<th>$k_{100}$</th>
<th>$o_1$</th>
<th>$m_1$</th>
<th>...</th>
<th>$m_k$</th>
<th>...</th>
<th>$m_{300}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>...</td>
<td>120</td>
<td>200</td>
<td>140</td>
<td>...</td>
<td>0.3</td>
<td>...</td>
<td>3200</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>...</td>
<td>60</td>
<td>600</td>
<td>60</td>
<td>...</td>
<td>0.6</td>
<td>...</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>...</td>
<td>55</td>
<td>120</td>
<td>55</td>
<td>...</td>
<td>0.4</td>
<td>...</td>
<td>4300</td>
</tr>
</tbody>
</table>

Training Set
Step 1: Metric Pruning
→ Pick $o_1$ and $m_k$ via FA and K-means (assume only two clusters)

Step 2: Knob Selection
→ Lasso
→ Incremental selection
→ Get 12 knobs
### Step 1: Metric Pruning

--> Pick \( o_1 \) and \( m_k \) via FA and K-means

### Step 2: Knob Selection

→ Lasso
→ Incremental selection

### Step 3: building GPR

→ Each workload
→ Each pruned metric
→ Building a GPR

Get 50 * 2 = 100 GPRs

---

**Pruned_1.csv**

<table>
<thead>
<tr>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>...</th>
<th>( k_{12} )</th>
<th>( o_1 )</th>
<th>( m_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4</td>
<td>...</td>
<td>100</td>
<td>519</td>
<td>0.1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>...</td>
<td>80</td>
<td>700</td>
<td>3.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>72</td>
<td>12</td>
<td>...</td>
<td>77</td>
<td>200</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Pruned_2.csv**

<table>
<thead>
<tr>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>...</th>
<th>( k_{12} )</th>
<th>( o_1 )</th>
<th>( m_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>5</td>
<td>...</td>
<td>100</td>
<td>106</td>
<td>1.1</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>...</td>
<td>90</td>
<td>740</td>
<td>4.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>102</td>
<td>10</td>
<td>...</td>
<td>77</td>
<td>200</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Pruned_50.csv**

<table>
<thead>
<tr>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>...</th>
<th>( k_{12} )</th>
<th>( o_1 )</th>
<th>( m_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>...</td>
<td>120</td>
<td>200</td>
<td>0.3</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>...</td>
<td>60</td>
<td>600</td>
<td>0.6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>...</td>
<td>55</td>
<td>120</td>
<td>0.4</td>
</tr>
</tbody>
</table>

---

(build)

**Training Set**

(build)
Step 1: Metric Pruning
-->
Pick \( o_1 \) and \( m_k \) via FA and K-means

Step 2: Knob Selection
→ Lasso
→ Incremental selection

Step 3: GPR building
→ Each workload
→ Each pruned metric
→ Building a GPR
Get 50 * 2 = 100 GPRs

Step 4: workload mapping
-->
"target.csv" to 1.csv

Compare the average distances among all the pruned metrics AFTER normalizing them by using “bin metric values".
Step 1: Metric Pruning

-> Pick $o_1$ and $m_k$ via FA and K-means

Step 2: Knob Selection

→ Lasso
→ Incremental selection

Step 3: building GPR

→ Each workload
→ Each pruned metric
→ Building a GPR
Get $50 \times 2 = 100$ GPRs

Step 4: workload mapping

→ “target.csv” to 1.csv

Step 5: building a GPR model via the pairwise data

Step 6: prediction / recommendation
Gaussian Process Regression

Trade-off between
- Mean
- Variance (uncertainty)

\[
\begin{bmatrix}
    f^* \\
    f
\end{bmatrix} \sim \mathcal{N}
\left(
    \begin{bmatrix}
        0 \\
        K(X^*, X^*)
    \end{bmatrix},
    \begin{bmatrix}
        K(X^*, X) & K(X^*, X) \\
        K(X, X^*) & K(X, X)
    \end{bmatrix}
\right),
\tag{1}
\]

\[f \mid X, X^*, f^* \sim \mathcal{N}(m, \Sigma),\] where

\[
m = K(X, X^*)K(X^*, X^*)^{-1}f^*
\]

\[
\Sigma = K(X, X) - K(X, X^*)K(X^*, X^*)^{-1}K(X^*, X)
\]

(b) Posterior
Experiment Setup

• DBMSs: MySQL (v5.6), Postgres (v9.3)

• Training data collection
  • 15 YCSB workload mixtures
  • ~30k trials per DBMS

• Experiments conducted on Amazon EC2
Ottertune VS Ituned

• Ottertune: green
  • Reuse data to train a GPR

• Ituned: blue
  • Only use init 10 confs to train GPR

**Figure 6: Tuning Evaluation (TPC-C)** – A comparison of the OLTP DBMSs for the TPC-C workload when using configurations generated by OtterTune and iTuned.
Efficiency Comparison

Figure 10: Efficacy Comparison (MySQL) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) Lithuanian DBA configuration, and (5) Amazon RDS configuration.

Figure 11: Efficacy Comparison (Postgres) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) expert DBA configuration, and (5) Amazon RDS configuration.