A framework for Experiment-driven System Management

Vamsidhar Thummala
Duke University

Joint work with Shivnath Babu and Jeff Chase
Day-to-day System Management Tasks

• Performance tuning
• Benchmarking
• Capacity planning
• Troubleshooting/diagnosis
Performance tuning a Database System (1/2)

• Parameters can control
  – Memory distribution
  – I/O optimization
  – Parallelism
  – Optimizer’s cost model

• Number of parameters ~ few dozens
  – 10-15 critical parameters depending on OLAP vs. OLTP

• Administrator’s Goal: Find the best configuration setting that satisfies SLA
Performance tuning a Database System (2/2)

TPC-H Workload 3Q7+3Q13

Average running time (sec)

effective_cache_size(MB)  shared_buffers(MB)
Benchmarking a Web Server

WebServer

Normalized Peak Rate

Number of nfds

Number of disks

Normalized Peak Rate

Number of disks
Why is System Management challenging?

Process is:
- Manual
- Time consuming
  - Hours (if not days)
- Demands expert knowledge

Result

Mgmt. task

Are more experiments needed?

Yes

**Process**
output to extract information

**Plan**
next set of experiments

**Conduct**
experiments on workbench
### Our vision for System Management

<table>
<thead>
<tr>
<th>Current</th>
<th>vs.</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td></td>
<td>Automate</td>
</tr>
<tr>
<td>Time consuming</td>
<td></td>
<td>Avail cloud to reduce time</td>
</tr>
<tr>
<td>Demands expert knowledge</td>
<td></td>
<td>Simplify and provide necessary tools for an average administrator</td>
</tr>
</tbody>
</table>
Opportunity in Cloud

• Cloud provides
  – Illusion of infinite resources
  – On demand provisioning

• Private clouds
  – Large enterprises

• Public clouds
  – SMB

• “On cloud, from the cost perspective -- running 100 experiments sequentially on one machine is same as running 100 experiments in parallel on 100 machines”
Our Contributions

- A framework for experiment-driven management (.eX)
  - Declarative language (eXL)
  - Intelligent analysis
    - Systematic exploration of design space
    - Adaptive/Sequential Sampling
  - Automatic deployment (Chef/Puppet)
  - Automatic execution
  - Interactive visualization
.eX Overview

- Users & tools express needs declaratively eXL
- .eX core plans and conducts experiments on designated resources
- Intuitive visualization of results
Declarative Language (eXL)

• eXL: experiment specification and query language

• Build on the top of SQL
  – EXPLAIN [SAMPLES] [ANALYZE] [PLOT] <Query>
Example (1/5)

EXPLAIN SAMPLES $\min(\text{Performance.responsetime}),\ \max(\text{Performance.throughput})$

FROM Workload.clients, Application.shared_buffers, Application.work_mem, System.mem, Budget.cost

WHERE Workload.clients IN [10..100] AND
    Application.shared_buffers IN [1GB..2GB]
    Application.work_mem IN [1MB..10MB]
    Application.shared_buffers + Workload.clients*Application.work_mem < 0.7*System.mem
    Performance.responsetime <= 100ms

USING "LHS"

LIMIT 10
Example (2/5)

• Output of **EXPLAIN SAMPLES**

<table>
<thead>
<tr>
<th>Workload. clients</th>
<th>Application. shared_buffers</th>
<th>Application. work_mem</th>
<th>System. mem</th>
<th>Budget. cost</th>
<th>Performance. responsetime</th>
<th>Performance. throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.5GB</td>
<td>5MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>1.7GB</td>
<td>2MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>30</td>
<td>1.1GB</td>
<td>8MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>45</td>
<td>1.8GB</td>
<td>4MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>60</td>
<td>1.4GB</td>
<td>10MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>75</td>
<td>1GB</td>
<td>3MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>100</td>
<td>1.3GB</td>
<td>7MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Adaptive Sampling

1. **Bootstrapping:**
   - Conduct initial set of experiments

2. **Sequential Sampling:**
   - Select NEXT experiment based on previous samples

**Main idea:**
1. Compute the utility of the experiment
2. Conduct experiment where utility is maximized
3. We used Gaussian Process for computing the utility

**Stopping Criteria:**
Based on budget

- **Bootstrapping**
  - Latin Hypercube Sampling
  - k-Furthest First
Example (3/5)

- Output of **EXPLAIN SAMPLES ANALYZE** with 3-way parallelism

<table>
<thead>
<tr>
<th>Workload clients</th>
<th>Application shared buffers</th>
<th>Application work_mem</th>
<th>System mem</th>
<th>Budget cost</th>
<th>Performance responsetime</th>
<th>Performance throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.5GB</td>
<td>5MB</td>
<td>4GB</td>
<td>Default</td>
<td>70ms</td>
<td>40</td>
</tr>
<tr>
<td>65</td>
<td>1.7GB</td>
<td>2MB</td>
<td>4GB</td>
<td>Default</td>
<td>120ms</td>
<td>70</td>
</tr>
<tr>
<td>30</td>
<td>1.1GB</td>
<td>8MB</td>
<td>4GB</td>
<td>Default</td>
<td>90ms</td>
<td>60</td>
</tr>
<tr>
<td>45</td>
<td>1.8GB</td>
<td>4MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>85</td>
<td>1GB</td>
<td>10MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>70</td>
<td>1.3GB</td>
<td>3MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>100</td>
<td>1.4GB</td>
<td>7MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Example (4/5)

- Output of `EXPLAIN SAMPLES ANALYZE` with 3-way parallelism

<table>
<thead>
<tr>
<th>Workload. clients</th>
<th>Application. shared_buffers</th>
<th>Application. work_mem</th>
<th>System. mem</th>
<th>Budget. cost</th>
<th>Performance. responsetime</th>
<th>Performance. throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.5GB</td>
<td>5MB</td>
<td>4GB</td>
<td>Default</td>
<td>40ms</td>
<td>20</td>
</tr>
<tr>
<td>65</td>
<td>1.7GB</td>
<td>2MB</td>
<td>4GB</td>
<td>Default</td>
<td>120ms</td>
<td>70</td>
</tr>
<tr>
<td>30</td>
<td>1.1GB</td>
<td>8MB</td>
<td>4GB</td>
<td>Default</td>
<td>70ms</td>
<td>60</td>
</tr>
<tr>
<td>45</td>
<td>1.8GB</td>
<td>4MB</td>
<td>4GB</td>
<td>Default</td>
<td>55ms</td>
<td>80</td>
</tr>
<tr>
<td>85</td>
<td>1GB</td>
<td>10MB</td>
<td>4GB</td>
<td>Default</td>
<td>80ms</td>
<td>110</td>
</tr>
<tr>
<td>70</td>
<td>1.3GB</td>
<td>3MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>100</td>
<td>1.4GB</td>
<td>7MB</td>
<td>4GB</td>
<td>Default</td>
<td>95ms</td>
<td>100</td>
</tr>
</tbody>
</table>
Example (4/5)

- Output of `EXPLAIN SAMPLES ANALYZE` with 3-way parallelism

<table>
<thead>
<tr>
<th>Workload clients</th>
<th>Application shared buffers</th>
<th>Application work_mem</th>
<th>System mem</th>
<th>Budget cost</th>
<th>Performance responsetime</th>
<th>Performance throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.5GB</td>
<td>5MB</td>
<td>4GB</td>
<td>Default</td>
<td>40ms</td>
<td>20</td>
</tr>
<tr>
<td>65</td>
<td>1.7GB</td>
<td>2MB</td>
<td>4GB</td>
<td>Default</td>
<td>120ms</td>
<td>70</td>
</tr>
<tr>
<td>30</td>
<td>1.1GB</td>
<td>8MB</td>
<td>4GB</td>
<td>Default</td>
<td>70ms</td>
<td>60</td>
</tr>
<tr>
<td>45</td>
<td>1.8GB</td>
<td>4MB</td>
<td>4GB</td>
<td>Default</td>
<td>55ms</td>
<td>80</td>
</tr>
<tr>
<td>85</td>
<td>1GB</td>
<td>10MB</td>
<td>4GB</td>
<td>Default</td>
<td>80ms</td>
<td>110</td>
</tr>
<tr>
<td>70</td>
<td>1.3GB</td>
<td>3MB</td>
<td>4GB</td>
<td>Default</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>100</td>
<td>1.4GB</td>
<td>7MB</td>
<td>4GB</td>
<td>Default</td>
<td>95ms</td>
<td>100</td>
</tr>
</tbody>
</table>
Example (5/5)

- Output of `EXPLAIN SAMPLES ANALYZE PLOT Application.shared_buffers, Application.work_mem, Performance.responsetime`
Empirical Evaluation of Adaptive Sampling Algorithm

• Experiment Settings
  – PostgreSQL v8.2
  – 3 Sun Solaris machines with 3 GB RAM, 1.8 GHz processor
  – Real dataset
    • TPC-H benchmark
      – SF = 1 (1GB, total database size = 5GB)
      – SF = 10 (10GB, total database size = 20GB)
    • W1-SF1: TPC-H Q18 on SF1
    • W2-SF1: Random mix of 100 TPC-H queries on SF 1
    • W2-SF100: Random mix of 100 TPC-H queries on SF 100
Comparison of Adaptive Sampling Tuning Quality
## Comparison of Adaptive Sampling Tuning Time

Cutoff time for an experiment: 90 minutes

<table>
<thead>
<tr>
<th></th>
<th>MOWILE</th>
<th>BruteForce</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1-SF1</td>
<td>2 hours</td>
<td>8 hours</td>
<td>1.4 hours</td>
</tr>
<tr>
<td>W2-SF1</td>
<td>6.2 days</td>
<td>21.7 days</td>
<td>4.6 days</td>
</tr>
<tr>
<td>W2-SF10</td>
<td>18 days</td>
<td>68 days</td>
<td>14.8 days</td>
</tr>
</tbody>
</table>
Back of the envelope calculation

- Administrators cost $250/day; Consultants $100/hr
- 1 day of experiments gives wealth of info
  - Opportunities due to cloud
  - Cost of running experiments
    - Server: $10/day
    - Storage: $0.4/day
    - I/O: $5/day
    - Total: $15/day
Conclusion

• Automated experiment-driven management: The time has come
  – Need, infrastructure, & promise are all there

• Challenges
  – Data migration
  – Auto Deployment
  – Sampling in larger dimensions
  – Cross-layer interactions
Q & A

• Thank you!

• Check out for our first prototype release in early 2011!

  vamsi@cs.duke.edu