TeaStore
A Micro-Service Application for Benchmarking, Modeling and Resource Management Research

Presentation

Jóakim von Kistowski, Simon Eismann, André Bauer, Norbert Schmitt, Johannes Grohmann, Samuel Kounev

April 9, 2018

https://github.com/DescartesResearch/TeaStore
Example Research Scenario

Auto-Scaling and Placement

- Placement at run-time

Performance Modeling

- An approach for the auto-scaling + placement problem

- Build or extract model

- Use Model for placement decision
Challenge: Evaluation of ...

- Placement algorithms
- Auto-scalers
- New modeling formalisms
- Model extractors

Reference applications help to
- Measure placement power consumption and performance
- Measure auto-scaler elasticity
- Evaluate model (extractor) accuracy

➢ Require **reference and test applications**
Requirements for a Test Application

- Scalable
- Allows for changes at run-time
- Reproducible performance results
- Diverse performance behavior
- Online monitoring
- Load Profiles
- Simple setup
- Modern technology stack
Micro-Service test application

- Five Services + Registry
- Uses Netflix “Ribbon” client-side load balancer
  - Swarm/Kubernetes supported, not required
- Pre-instrumented variant with Kieker
- Has Docker Images
  - Alternatively: documentation for manual deployment
Services I

Registry
- Simplified Eureka
- Service location repository
- Heartbeat

RegistryClient
- Dependency for every service
- Netflix “Ribbon”
- Load balances for each client

WebUI
- Servlets/Bootstrap
- Integrates other services into UI
  - CPU + Memory + Network I/O

Authentication
- Session + PW validation
- SHA512 + BCrypt
  - CPU
Services II

**PersistenceProvider**
- Encapsulates DB
- Caching + cache coherence
  - Memory

**ImageProvider**
- Loads images from HDD
- 6 cache implementations
  - Memory + Storage

**Recommender**
- Recommends products based on history
- 4 different algorithms
  - Memory or CPU

**TraceRepository**
- AMQP Server
- Collects traces from all services
## Additional Performance Properties

<table>
<thead>
<tr>
<th>Two types of caches</th>
<th>Load independent tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Black-box persistence cache</td>
<td>➢ Periodic recommender retraining (optional)</td>
</tr>
<tr>
<td>➢ White-box image provider cache</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Different load types</th>
<th>Startup behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ CPU</td>
<td>➢ Auth and WebUI start “instantly”</td>
</tr>
<tr>
<td>➢ I/O</td>
<td>➢ Recommender needs training on startup</td>
</tr>
<tr>
<td>➢ Network</td>
<td>➢ Image Provider creates images on startup</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal state</th>
<th>Configuration options</th>
</tr>
</thead>
<tbody>
<tr>
<td>➢ Database size influences resource</td>
<td>➢ Recommender algorithms</td>
</tr>
<tr>
<td>demands</td>
<td>➢ Recommender retraining interval</td>
</tr>
<tr>
<td></td>
<td>➢ Image Provider cache implementations</td>
</tr>
<tr>
<td></td>
<td>➢ Database size</td>
</tr>
</tbody>
</table>
HTTP load generator

- Supports varying load intensity profiles
  - Can be created manually
  - Or using LIMBO

- Scriptable user behavior
  - Uses LUA scripting language
  - “Browse” Profile on Github

Example load intensity profile:

“Browse” user profile:
Example: Energy Efficiency of Placements

Placement Candidate 1

<table>
<thead>
<tr>
<th></th>
<th>Web UI</th>
<th>Auth</th>
<th>Img</th>
<th>Persist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Max \(\uparrow\) 1011.9 Tr/s
- Max \(\downarrow\) 179.6 W
- Geo \(\downarrow\) 4.4 Tr/J

Placement Candidate 2

<table>
<thead>
<tr>
<th></th>
<th>Web UI</th>
<th>Auth</th>
<th>Img</th>
<th>Persist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Max \(\uparrow\) 1067.7 Tr/s
- Max \(\downarrow\) 187.0 W
- Geo \(\downarrow\) 4.3 Tr/J
Does it scale?
Thank You!

https://github.com/DescartesResearch/TeaStore