ConfAdvisor: A Performance-centric Configuration Tuning Framework for Containers on Kubernetes

Tatsuhiro Chiba, Rina Nakazawa, Hiroshi Horii, Sahil Suneja and Seethalami Seelam

IBM Research
IBM T.J. Watson Research Center
Agenda

- Motivation, Problems and Challenges
- Design & Implementation
- Evaluation
- Summary
Performance Related Configs for Software

- **What kind of configs?**
  - the configs which can change software behavior
    - heap size, algorithm, # threads, enabling/disabling futures...

- **How many configs can we tune?**
  - Increasing tunable configs year by year [Xu+, FSE ’15]
  - Apache HTTP Server: 550+
  - MySQL: 460+
  - OpenJDK (JVM): 700+

```java
$> java -XX:+PrintFlagsFinal
[Global flags]
   uintx AdaptiveSizeDecrementScaleFactor = 4   {product}
   uintx AdaptiveSizeMajorGCDecayTimeScale   = 10 {product}
   uintx AdaptiveSizePausePolicy             = 0   {product}
   uintx AdaptiveSizePolicyCollectionCostMargin = 50 {product}
   uintx AdaptiveSizePolicyInitializingSteps  = 20 {product}
...
$> java -XX:+PrintFlagsFinal | wc
 723  3758  72844
```
Misconfig makes big performance impact

- What is the result after config tuning?
  - success: performance improvement
  - failure: performance drawback
  - config influences each other in complex manner

- system anomalies due to misconfig
  - up to 31% of cloud outage are caused by misconfiguration [Yin+, SOSP'11]
  - misconfig is a top reason of hadoop system outage [Ariel+, IEEE Software '13]
Example of Configs – Liberty and Nginx

Liberty (server.xml)

```xml
<server description="new server">
  <featureManager>
    <feature>microProfile-2.0</feature>
  </featureManager>
  <!-- optimal thread size depends on workload -->
  <executor coreThreads="1" maxThreads="10"/>
</server>
```

Nginx (nginx.conf)

```plaintext
worker_processes  100;  # depends on env
events {
  worker_connections 1024;  # depends on env
}
http {
  sendfile  off;  # better to set on
  tcp_nopush  off;  # better to set on
  keepalive_timeout 60;  # depends on role
}
```

- **Can change those configs, but ...**
  - Not easy task to set proper config manually
  - Need to know app role detail

- **What about automation?**
  - Ansible or Kubernetes Operator can fix wrong operational processes automatically
  - PerfOps cycle still remains out of the automation loop..
Why config tuning is difficult?

- Everyone cannot always make best config tuning, because
  - exists too many knobs
  - require deep knowledge about the system and apps
  - application programmer ! config tuning specialist
- Need to know what happens in the apps from various logs/metrics
  - Java Apps → GC, JIT, Method profiling, lock contention..
  - MongoDB → query stats, perf counter, i/o monitoring, index...

Config Tuning Steps
- What is the next value?
  - set Parameter
  - set XXX=10
- Run App /run
- Stop App /stop

Feedback
- Which parameter should we change?

How we can read/understand it?
- Evaluation
  - eval response time
- metrics / logs
Config Tuning Difficulties in Container/Kubernetes (1/3)

- Anyone can publish container images
  - pre-tuned config image
  - default config image
  - misconfig image

- Official images have few preset cfg
  - still remains enough tuning space

### Container & Deployment Life Cycle

- **build/pull image**
  - docker pull mongo

- **deploy service**
  - kubectl run

- **running Pods**
  - /mongod

- **Pods Performance**
  - Pods Health

---

**Table 1**

<table>
<thead>
<tr>
<th>Software &amp; Config</th>
<th>No. of tuning knobs</th>
<th>image name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nginx (nginx.conf)</td>
<td>20</td>
<td>nginx:1.15</td>
</tr>
<tr>
<td>Apache2 (httpd.conf)</td>
<td>72</td>
<td>httpd:2.4.37</td>
</tr>
<tr>
<td>Redis (redis.conf)</td>
<td>0</td>
<td>redis:5.0</td>
</tr>
<tr>
<td>MongoDB (mongodb.conf)</td>
<td>8</td>
<td>mongo:4.1.4</td>
</tr>
<tr>
<td>Open Liberty (jvm.options)</td>
<td>0</td>
<td>open-liberty:javaee8</td>
</tr>
<tr>
<td>Cassandra (jvm.options)</td>
<td>32</td>
<td>cassandra:3.3.1</td>
</tr>
</tbody>
</table>
Config Tuning Difficulties in Container/Kubernetes (2/3)

- **Config Violation at Deployment Timing**
  - Resource Quota (requests, limits)
  - Service mesh policies (load balancing, network capacity)
  - Storages (Persistent Volume or temporal space)

```
spec:
  containers:
  - image: cassandra:3.11.2
    resources:
      limits:
        memory: 1Gi
        cpu: 2
```

Container & Deployment Life Cycle

- **build/pull image**
  - docker pull mongo

- **deploy service**
  - kubectl run

- **running Pods**
  - /mongod

- **Pods Performance**
  - Pods Health

Misconfig resides in:
- image
- deployment yaml
Dynamic config tuning after running for a while
- Based on app usage characteristics or workloads
  - Read heavy vs. Write heavy
  - Response Time vs. Throughput
- Detecting failed request, etc.
- Horizontal pod scaling vs. vertical pod scaling
## Related Works

- No generalized framework to make a config advice statically or dynamically
- No hybrid approach to use rule based and ML based tuning
- No continuous PerfOps config tuning system to support image, container, and Kubernetes

<table>
<thead>
<tr>
<th>Name</th>
<th>Approach</th>
<th>Target System</th>
<th>Limitation</th>
<th>dedicated</th>
<th>k8s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ansible</td>
<td>Rule base</td>
<td>OS, init setup</td>
<td>static</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>Dr.Erephant [1] Starfish [2]</td>
<td>Rule base</td>
<td>Hadoop Spark</td>
<td>Heuristics</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>OtterTune [3]</td>
<td>Model base</td>
<td>MySQL PostgreSQL</td>
<td>model update</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>CherryPick [4] BOAT [5]</td>
<td>Search base (Bayesian Opt.)</td>
<td>Hive, Spark, JVM VM Instance type</td>
<td>Sampling convergence</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Our Framework</td>
<td>Hybrid</td>
<td>Various Apps</td>
<td>static dynamic</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

- [1] Dr. Erephant, Spark Summit '16
- [2] Starfish, CIDR '11
- [3] OtterTune, SIGMOD '17
- [4] CherryPick, NSDI '17
- [5] BOAT, WWW '17
Challenges and Contributions

- **Challenges**
  - Achieve static/dynamic config tuning for various types of resources on k8s
  - Apply an optimized config continuously, aligned with cloud native app lifecycle
  - Support not only one specific app but many apps
  - Give a reasonable advice why my container is slow, relying on my container's config and metrics

- **Contributions**
  - Building a config tuning framework, ConfAdvisor, on Kubernetes
  - Extendable/customizable plugin system which can include tuning logic as a code
  - Case study: config tuning for Cassandra/ Liberty / MongoDB
Agenda

- Motivation, Problems and Challenges
- Design & Implementation
- Evaluation
- Summary
IC2E 2019 / ConfAdvisor: A Performance-centric Configuration Tuning Framework for Containers on Kubernetes

Config tuning opportunities in cloud-native app lifecycle

- **Image**
- **deployment**
- **Runtime**
ConfAdvisor – Definition of Advice Levels

All advices are personalized result, which is based on its config or metrics that system can observe

- **Level 1. Advice based on app internal default config**
  - Liberty: jvm.options, server.xml, persistent.xml
  - mongodb: mongod.conf

- **Level 2. Advice based on app env config such as k8s quota or architecture**
  - Liberty: heap size, app threads size
  - Spark: executor heap size, executor threads

- **Level 3. Advice based on observed app runtime metrics**
  - Liberty: lock contention, optimal heap size, optimal GC algorithm
  - mongodb: throughput aware settings / latency aware settings
ConfAdvisor – System Overview

- provides a service which gives perf-sensitive conf advice
  - continuous automated PerfOps pipeline by user or system

- pluggable/programmable/declarative rule framework
  - customize/extend advice rule by perf engineers
  - install plugin if we need

ConfAdvisor CLI

Advisor Service
- provides queryable interface by user/system
- provides detail of rules
- provides a proper config

ConfAdvisor Service
- provides a proper config

Add Plugins and Rules
- simple customizable what-if rule
- python and jinja2 template
- config/metrics are available

Container

Image

ConfAdvisor Framework (API / Functions / ML)

Config

Metrics
Implementation: Capturing config and metrics

- **Metrics collector**
  - deploy app exporter as a side car
  - periodically captured by prometheus

- **Container config collector**
  - periodically captured by crawler(\*) daemonset
  - crawls config immediately by detecting container creation event

- **Image config collector**
  - crawls config inside of image immediately when finding a new image in registry

(*) https://github.com/cloudviz/agentless-system-crawler
Implementation: Storing configs and metrics

- **Metrics storage**
  - stores all metrics into Prometheus

- **Config storage**
  - Data normalization by using Augeas (*)
  - Store them into ElasticSearch

(*2) [http://augeas.net/](http://augeas.net/)
Implementation: Advisor Framework

- **Advisor Service & Framework**
  - handle advice request for an app
  - load a specified plugin
  - calc. optimized config for the app
  - give back the result

- **Plugin**
  - Written in Python
  - Implemented logic and rules per each config
    - Liberty, MongoDB, K8S, Cassandra, etc..
  - can use predefined functions by framework
  - can access various data from API
Implementation: Declarative Rules (1/2)

- **Declarative Config Template**
  - apply it when *what-if* rule is matched
  - *current* is a placeholder to access the current value
  - advice will be generated from a formula in *advice*

- **Customizable**
  - append any rules for a specified config if you want

```
"jvm.options": [
{
   "name": "mx",
   "what-if": "current.mx > current.limits_memory",
   "advice": "current.limits_memory * 0.75",
   "order": 0,
   "message": "should keep mx less than memory limits"
},
{
   "name": "ms",
   "what-if": "current.ms != advice.mx",
   "advice": "current.limits_memory * 0.75",
   "order": 1,
   "message": "should keep ms same as mx"
}
]```
Implementation: Declarative Rules (2/2)

- **Extendable plugins**
  - can write any tuning logic as a code in a plugin
  - prepare it as an UDF function (e.g. `rate_cpu_usage`)
  - the function is automatically available inside of declarative rule

- **Framework API supports**
  - exposing time-series data as a pandas dataframe
  - utilizing prebuilt ML model

```
"server.xml": [
  {
    "key": "maxThreads",
    "what-if": "current.maxThreads < (current.cpu * 5)",
    "advice": "current.cpu * 5",
    "order": 0,
    "message": "should start with 5 * vcpu"
  },
  "k8s": [
    {
      "key": "scale_pod",
      "what-if": "rate_cpu_usage(vars.interval) > 0.8"
      "advice": "current.replica += 1",
      "order": 0,
      "message": "check average cpu usage in last interval"
    }
  ]
```
Agenda

- Motivation, Problems and Challenges
- Design & Implementation
- Evaluation
- Summary
### Example of Implemented Declarative Rules

<table>
<thead>
<tr>
<th>App</th>
<th>Config</th>
<th>what-if rule</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberty</td>
<td>server.xml</td>
<td>maxPoolSize != coreThreads</td>
<td>maxPoolSize = coreThreads</td>
</tr>
<tr>
<td>Redis</td>
<td>redis.conf</td>
<td>maxmemory == None</td>
<td>maxmemory = mem_limit * 0.8</td>
</tr>
<tr>
<td>Node.js</td>
<td>k8s yaml spec</td>
<td>rate_cpu_usage(30m) &gt; 0.9</td>
<td>replica++</td>
</tr>
<tr>
<td>MongoDB</td>
<td>mongod.conf</td>
<td>cacheSizeGB &gt; mem_limit</td>
<td>cacheSizeGB = mem_limit / 2</td>
</tr>
<tr>
<td>Cassandra</td>
<td>jvm.options</td>
<td>-XX:MaxHeapSize &gt; 16GB</td>
<td>-XX:+UseG1GC</td>
</tr>
<tr>
<td>Cassandra</td>
<td>jvm.options</td>
<td>jdk_version &lt; 1.8.0_192-b01</td>
<td>-XX:+UseCGroupMemoryLimitForHeap</td>
</tr>
</tbody>
</table>

- **Implemented advisor plugin for apps**
  - Liberty, Httpd, Nginx, Node.js, Redis, MongoDB, Cassandra

- **Advisor source coming from,**
  - Official Tuning Guide (RedBook, official site)
  - JIRA, Github Issue, Quora, etc.
  - manually ported from those docs
Runtime Overhead – querying performance

- Evaluated end-to-end ConfAdvisor performance
  - L1/L2: simple rule and advice processing
  - L3: including data loading from Prometheus
  - Make it scale by putting ConfAdvisor replicas more

<table>
<thead>
<tr>
<th>category</th>
<th>avg</th>
<th>stdev</th>
<th>min</th>
<th>max</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1/L2</td>
<td>25.6</td>
<td>1.93</td>
<td>20.1</td>
<td>29.7</td>
<td>25.7</td>
</tr>
<tr>
<td>L3</td>
<td>3.32</td>
<td>1.92</td>
<td>0.56</td>
<td>5.55</td>
<td>4.39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>Xeon E5 2683 v3 (Xen), SMT2</td>
</tr>
<tr>
<td>VM</td>
<td>8 vCPU, 16GB RAM, Ubuntu 16.04</td>
</tr>
<tr>
<td>Instance</td>
<td>4</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>1.11.1</td>
</tr>
<tr>
<td>Docker</td>
<td>18.06.1-ce</td>
</tr>
<tr>
<td>Prometheus</td>
<td>2.3.1</td>
</tr>
</tbody>
</table>
Runtime Overhead – crawling performance

- Evaluated config crawling overhead
  - depends on number of containers on each node
  - not so big overhead

- Evaluated metrics crawling overhead
  - depends on the number of metrics and frequency
  - seems to be low
Case Study: Cassandra + YCSB benchmark (1/2)

- particular OpenJDK do not take cgroup limit into consideration
  - `-XX:+UseCGroupMemoryLimitForHeap`
  - `what-if: version > 1.8.0_192-b01`: no need to set
  - `what-if: version < 1.8.0_192-b01`: must set the option explicitly

- recommends to append the option
  - w/o advice: can not finish workload (KILLED by the system)
  - w/ advice: keep smaller heap than 1GB limitation

---

spec:
  containers:
    - image: cassandra:3.11.2
  resources:
    limits:
      memory: 1Gi
      cpu: 2
Case Study: Cassandra + YCSB benchmark (2/2)

- recommends to append one more JVM option
  - LIMIT: -XX:+UseCGroupMemoryLimitForHeap (-Xmx256m)
  - LIMIT+RAM: -XX:+UseCGroupMemoryLimitForHeap -XX:MaxRAMFraction=1 (-Xmx1g)

- By fixing the JVM option misconfig
  - successfully run workload
  - achieved 1.7x or 2.5x improvement
Case Study: memory sizing

- **Liberty + DayTrader**
  - Nursery heap/tenured heap size optimization

- **MongoDB + YCSB**
  - WiredTiger cache size optimization

### Graph
![Graph showing relative throughput gain](image)

- **DayTrader**
  - Duration: 1800 sec
  - Ramp up: 300 sec
  - # Client thread: 60

- **YCSB**
  - Data size: 1KB
  - Record: 1M
  - Dist.: Uniform
  - # Client thread: 8
  - Workload A: read/update = 50/50
  - Workload B: read/update = 95/5
Memory Sizing: Liberty + DayTrader

- Heap size ratio affects performance
  - Large nursery heap improves throughput
  - Too small tenured heap causes long gc pause time
- What kind of advice can we make?
  - what-if: \( 3 \times \text{current GC interval} < \text{latest GC interval} \)
  - advice: nursery heap ++
Memory Sizing: MongoDB + YCSB

- **Mongo does not consider memory limit**
  - tries to reserve half of system memory as a cache...

- **In-memory cache size affects performance**
  - workload A: 1.45x improvement
  - workload B: 1.20x improvement

- **What kind of advice can we make?**
  - what-if: read/write ratio ≒ 1.0
  - advice: cache = total memory / 2

spec:
- containers:
  - image: mongo:4.1.4
- resources:
  - limits:
    - memory: 1Gi
    - cpu: 1
Summary

- Built a sustainable config tuning framework on Kubernetes
- Provided extendable/programmable plugin system
- Confirmed perf improvement for the apps on Kubernetes

Future
- Imports various expert knowledge (build a ecosystem)
- Prepares Bayesian Optimization feature
- Imports various ML model