Data Driven Performance Repository to Classify and Retrieve Storage Tuning Profiles

Ismael Solis Moreno
IBM
Agenda

- The challenge of tuning performance for SDS.
- A data driven performance platform.
- A ML-based configuration profiles repository.
- Classifying & recovering configuration profiles.
- Comments & questions.
The challenge of tuning performance for SDS

Software-defined storage (SDS) enables users and organizations to uncouple or abstract storage resources from the underlying hardware platform for greater flexibility, efficiency and faster scalability by making storage resources programmable.

This approach enables storage resources to be an integral part of a larger software-designed data center (SDDC) architecture, in which resources can be easily automated and orchestrated rather than residing in siloes.
The challenge of tuning performance for SDS

When *administered correctly*, SDS increases performance, availability, and efficiency.

This is not so easy since SDS are normally mounted over complex environments:

- Large number of nodes
- Hardware heterogeneity
- Diverse and complex workload
- Intricate network characteristics

So, *How should I configure my SDS* solution to get the best performance, availability and efficiency?
The challenge of tuning performance for SDS

Intrinsically, SDS solutions are very flexible. It means that are highly customizable, and it means that there are several parameters that can be adjusted in order to provide the mentioned benefits.

So far the determination of tuning configurations are based on:

- General application best practices
- Highly skilled human experts
The challenge of tuning performance for SDS

Best practices are always a good start point but:

- Best practices normally do not cover all the possible scenarios.
- Best practices can get deprecated with the time.

On the other hand, knowledge acquired by human experts:

- Can be diminished by rotation of personnel and lack of expertise and skills sharing.

These create the need of a knowledge repository and a computing artifact to smartly classify and recover tuning profiles.
A Data Driven Performance Platform

Objectives

To research and develop a scalable, platform to efficiently process, store, query and analyze the data from the performance evaluation of different IBM solutions. The proposed platform must have the following characteristics:

• Provide a mechanism to ETL performance data into well defined structure to facilitate its exploitation.

• Results from performance benchmarks will be stored in a central repository for an efficient access, query, and reporting.

• Facilitate the latest as well as historic view of the performance results to the development, customer adoption, and other customer service teams.

• Provide an interactive user interface to perform data mining for descriptive, predictive and cognitive analytics (using data from different sources).

• Provide mechanisms to create, classify and retrieve performance tuning profiles according to the collected data on benchmark results, resource usage logs, workload and hardware characteristics.
A Data Driven Performance Approach

**Project Objectives**
- Automated Data processing
- Data Storage instead of Document storage
- Historic view of performance results
- Performance Insights (descriptive, predictive & cognitive)

**Conceptual Overview**

1. Parsing & Structuring Engine
2. Central Data Repository
3. Processing Engine
4. Results Presentation Layer

**Performance Analysis dashboards**
- Tuning profiles classification and selection
A Data Driven Performance Platform

Architecture Overview

- Performance Data
  - Performance Results
  - System Profiles
  - Customer and Workload Data

- Data Driven Platform
  - Extract, Transform & Load (ETL)
  - Scalable Data Storage
  - Scalable Data Processing
  - Machine Learning & Statistics
  - Interface for Cognitive

- Interactive User Interface
  - Reporting
  - Analytics
  - Trends
  - Alerts
  - Custom Data view & Analysis
  - API

Other IBM tools or Platforms

User 1

User 2
A Data Driven Performance Platform

Deployment Overview

Application Layer

- Django
- Python
- Cassandra - Python Driver
- MongoDB - Python Driver
- Spark Python API

Big Data Storage

- Cassandra Cluster
- Python
- MongoDB Cluster

Big Data Processing

- Spark - Cassandra Connector
- Spark - MongoDB Connector
- Spark Cluster
A Data Driven Performance Platform

Benefits

- Flexible, extensible, efficient, and scalable framework to store, query, and process performance data.
- Support historical query, analysis, and view of the results.
- Extend to not only store regression results but also other performance story results as well as new storage system evaluation results.
- Faster and centralized data availability.
- Custom data analysis to produce deeper insights about performance results and correlate them with customer, workload & systems configuration data.
- Use of Machine Learning to enhance the data utilization in configuration and tuning processes.
A ML-based configuration profiles repository

Example Use Case

“Selecting the closest tuning configuration profile according to system characteristics”.

Scenario:

Performance evaluations (for e.g., performance evaluation of new file system features or enhancements or workload, during regression tests on performance clusters), results in generation of the following data:

--Description of the overall solution (node CPU/memory, network, storage, disk-type, OS software etc).
--Description of the tested workload (DB, SWB, VDI, Metadata intensive, I/O access pattern).
--Full description of the tuned file system configuration.
--Observed Performance Results and conclusions.
A ML-based configuration profiles repository

Case Based Reasoning

- An approach to model what humans think.
- An approach to build intelligent systems.
- Sub-discipline of Artificial Intelligence.
- Belongs to Machine Learning Methods.

- Recall **similar** experiences (made in the past) from memory.
- Reuse that experience in the context of the new situation (*reuse it partially, completely or modified*).
- New experience obtained this way is **stored to memory** again.
Classifying & recovering configuration profiles

Step 1: Using the ETL module for extracting and structuring the data

- Performance data from user stories & customers
- Set of scripts for extracting, cleaning and transforming data
- Domain-ontology for mining a vector of components to describe the environment

Extract, Transform & Load (ETL)

- Scalable Data Storage
  - Distributed Wide-column DB
- Structured data for querying, reporting & further analytics
- Documents containing tunings to specific environment description

<table>
<thead>
<tr>
<th>ess</th>
<th>GL6</th>
<th>genomic-workload</th>
<th>4.2.1</th>
<th>infinia nd</th>
<th>flash</th>
</tr>
</thead>
</table>

Component Vector that contains key descriptors of the environment. It is used as index for classifying and storing tuning profile documents.
Classifying & recovering configuration profiles

Step 2: Storing the configuration profile and using ML (CBR) to assign indexes

1. Once the component vector is mined, the descriptors are used to classify the new case.
2. A CBR exemplar is created and a UUID is assigned.
3. The Exemplar is stored in the case base.
4. The actual tuning document is stored in a document oriented DB using the Exemplar UUID as key.
5. Tuning documents can be stored as Json format in such way they are human readable and also easily consumable to other IBM tools.

NOTE: CBR is a ML technique which uses solutions of similar past cases to solve new problems. In this context, an exemplar is just a case in the case base.
Classifying & recovering configuration profiles

Snippet of the Ontology using OWL

```xml
<!--
http://www.ddp.com/ontologies/ss/environment.owl#InfinibandNetwork
-->
<owl:Class rdf:about="http://www.ddp.com/ontologies/ss/environment.owl#InfinibandNetwork">
  <rdfs:label xml:lang="en">IB</rdfs:label>
  <rdfs:label xml:lang="en">IB Network</rdfs:label>
  <skos:prefLabel xml:lang="en">InfiniBand</skos:prefLabel>
</owl:Class>
```

Snippet of the cases insertion in MongoDB

```javascript
examlar = {
"_id": "ess|gl6|genomicworkload|gpfs|5.0.1|infiniband|flash",
"mmfsadm_dump_config": {
  "gpfs_parameters": {
    "aclGarbageCollectorDutyCycle": 0,
    "aclHashSpaceSize": 2000,
    "afmAsyncDelay": 15,
    "afmAsyncOpWaitTimeout": 300,
    ...
  }
}
```
Classifying & recovering configuration profiles

Step 3: Requesting the closest configuration profile based on the environment description.

- User can be a human or other tool submitting an environment description and receiving, the most similar case in the case base.

- Through the UI, the ML module supported by the domain-ontology creates a component vector of the request.

- Once the UUID of the most similar exemplar is determined, the tuning document is requested to the document oriented DB and passed to the user.

- The component vector is used to retrieve all the exemplars that have matching characteristics.

- The component vector is also used to measure the similarity between the request and the retrieved exemplars.
Classifying & recovering configuration profiles

Step 4: Retrieving the closest configuration profile based on the environment description.

1. Once the requested component vector is created, it needs to be parsed through the ontology for retrieving exemplars with matching characteristics.

2. The component vectors for each retrieved exemplar is built.

3. The requested component vector (R) is compared against the component vectors of the retrieved exemplars (En) using an appropriate similarity function $S(R, E_n)$.

4. The exemplar with closest match is selected to query the Document Oriented DB.

5. The selected tuning document is retrieved and provided to the user.
Classifying & recovering configuration profiles

Step 5: Evaluation and Adaptation of New Cases.

Once the tuning configuration of the most similar case is retrieved, it needs to be evaluated in the new system environment and adapted accordingly.

The adaptation of the tuning is then submitted to the proposed platform as a new case. Thus, this completes the CBR four-step process.
Classifying & recovering configuration profiles

Benefits of the Example Use Case

• Using a domain ontology as structure for classification allows to eliminate terminology ambiguity using synonyms and other relationships between the terms. For example: IB can be described as InfiniBand or Infini-Band.

• With more evaluated cases, the system develops expertise (similar to humans).

• The tuning profile is not just a document, but aggregation of performance data, workload, and system characteristics linked through appropriate data structures in the DB.

• The proposed architecture allows the interaction not only with humans but also with other tools aimed at automating the entire configuration process.
Classifying & recovering configuration profiles

Related Successful Use Cases


Ongoing Work

- We are in development, but need to evaluate the platform.
- Explore more similarity functions.
- Explore Deep Learning for the revising phase of CBR.
- Take this approach to different environments and significantly populate the knowledge base.
- Integrate the knowledge base with other approaches that are looking for the best configuration profiles.
Comments & Questions

Ismael Solís Moreno
ismael@mx1.ibm.com
https://www.linkedin.com/in/ismaelsm/