Санкт-Петербургский государственный университет

Кафедра системного программирования

Фадеева Анастасия Владимировна

Разработка модуля загрузки и обработки результатов автоматических тестов производительности систем хранения данных

Курсовая работа

Научный руководитель:
ст. преп. Немешев М. Х.

Санкт-Петербург
2018

Оглавление

Введение

В настоящее время количество хранимой и передаваемой информации растет. Во многих организациях объемы обрабатываемых данных имеют значительный размер, поэтому очень важно обеспечивать надежность хранения информации и быстрый доступ к ней. Для решения этих задач и применяются системы хранения данных. В случаях, когда требуется хранение больших объемов информации, важно не просто создать систему хранения, но и сделать ее оптимальной для решения конкретных задач компании, обеспечив должную производительность и надежность.

Так как производительность сильно зависит как от конкретной конфигурации, так и от характера нагрузки, существует потребность в прогоне многочисленных тестов. В компании Dell EMC есть специальный отдел, состоящий из “performance” инженеров, которые занимаются тестированием систем хранения данных, а затем проводят анализ результатов автоматизированных тестирований для дальнейшего улучшения систем [1].

Результаты автоматизированных тестирований - файлы больших размеров с несколькими тысячами строк, в таком объеме информации очень сложно ориентироваться, особенно когда нужно сравнить между собой результаты многих тестов. Представление данных в виде графиков, диаграмм и таблиц помогает значительно облегчить анализ. На данный момент инженеры пишут свои собственные скрипты для визуализации или пытаются анализировать данные в Excel. Собственные скрипты каждого инженера не обладают расширяемостью, при добавлении нового формата данных или потребности в новом виде графика приходится писать скрипты заново, а средствами Excel нельзя создать шаблон для обработки одинаковым способом разных данных - приходится делать одни и те же действия каждый раз для составления диаграмм одинакового типа.

Для решения задачи визуализации результатов автоматизированного тестирования было решено разработать фреймворк Perfstat. Это

легко расширяемый инструмент для обработки разнородных данных и генерации отчетов, состоящих из текстовых комменатриев, графиков, диаграмм и таблиц. Средствами фреймворка можно задавать шаблоны для генерации отчетов, тем самым автоматизировав визуализацию данных нескольких тестов.

Так как существует множество форматов входных данных, необходим загрузчик, который занимается преобразованием входной информации во внутренний формат фреймворка, причем добавление нового типа данных для анализа не должно создать проблем. К тому же результаты тестирования часто нуждаются в постобработке, например, фильтрации или группировке, а также подсчете статистики, поэтому необходима разработка расширяемого модуля, отвечающего за работу с разнородными данными.

1. Постановка задачи

Цель проекта Perfstat - создание расширяемого фреймворка, предназначенного для визуализации результатов автоматических тестов производительности систем хранения данных. Целью данной работы является реализация модуля, который будет отвечать за загрузку и обработку данных. Для её достижения были выделены следующие задачи:

- проведение исследования предметной области и обзора существующих решений;
- разработка расширяемого загрузчика данных различных форматов;
- реализация обработчика, позволяющего:
  - применять различные фильтры;
  - выполнять математические операции;
  - группировать и трансформировать входные данные;
- встраивание модуля во фреймворк;

2. Обзор существующих решений

В настоящее время инженеры самостоятельно пишут скрипты для обработки результатов автоматизированных тестирований и их визуализации, однако отсутствие общего расширяемого решения влечет за собой постоянное написание новых скриптов для анализа и обработки нового формата данных или для генерации другого типа диаграмм. Таким образом, очевидна необходимость фреймворка, позволяющего визуализировать данные разнообразных форматов и применять к ним различные операции.

На данный момент существует большое количество продуктов, направленных на создание комбинированных отчетов по различным данным. Мною были рассмотрены следующие решения:

- Tableau — решение компании Tableau Software, позволяющее легко визуализировать данные и представлять их в виде разнообразных графиков и диаграмм, однако недостатком является то, что Tableau требует уже трансформированных входных данных, то есть непосредственно работать с данными этот продукт не позволяет [8];

- Excel — решение Microsoft, с помощью которого можно работать с данными, создавать графики и диаграммы для добавления их в аналитические отчеты. Excel неудобен тем, что для составления отчета одного и того же типа по разным данным каждый раз нужно проделывать всю процедуру генерации отчета заново - это делает невозможным сделать получение графиков и диаграмм из данных автоматическим. К тому же очень неудобно применять операции к таблицами, состоящим из большого количества строк [2];

- Power BI— еще один коммерческий продукт компании Microsoft для представления и анализа различной информации, с его помощью можно создавать кастомизированные отчеты, в отличие от

Tableau, Desktop версия имеет набор инструментов для очистки и обработки данных, однако их очень мало [3];

- QlikView — коммерческий продукт компании QlikTech, дает возможность загружать и визуализировать любые данные благдаря встроенному обработчику, однако на ознакомление с синтаксисом этого инструмента может уйти достаточно много времени, к тому же для изменения типа визуализации иногда нужно полностью переписать скрипт [6];

- CrystalReports — решение компании SAP, позволяющее совмещать информацию из разных источников, обрабатывать данные и создавать наглядные отчеты [7];

Все вышеперечисленные продукты нацелены на решение разнообразных задач, связанных с формированием отчетов и визуализацией данных, распространением информации среди пользователей, их интегрирование займет много времени. Для данной проблемы функциональность многих решений избыточна, а возможностей некоторых (например, Tableau, Excel), наоборот, недостаточно для решения поставленной задачи из-за их ориентации либо только на работу с данными, либо лишь на красивую и легко изменяемую визуализацию.

К тому же не все из рассмотренных продуктов позволяют создавать шаблоны, позволяющие генерировать множество отчетов одного типа, но по разным данным, а необходимость каждый раз проделывать одни и те же действия, чтобы визуализировать данные, значительно замедляет процесс анализа. Все решения очень трудно адаптируются под конкретную семантику данных, что очень неудобно, так как результаты автоматизированных тестирований обычно обладают определенной семантикой, которую нужно учитывать при визуализации.

В результате можно сделать вывод об актуальности задачи создания расширяемого фреймворка с простым интерфейсом, адаптирующегося к данным различных форматов, позволяющего трансформировать их, применяя к ним разнообразные операции, а затем получать комбинированный отчет, состоящий из разнообразных графических элементов.

3. Реализация

3.1. Инструменты

В качестве языка реализации был выбран язык Python [5], так как программы, написанные на нем, достаточно просто встроить в процесс автоматизированного тестирования, к тому же для него написано множество библиотек для работы с данными. Для реализации модуля загрузки и обработки использовалась библиотека Pandas [4], содержащая функции, которые позволяют эффективно работать со структурированными данными.

3.2. Структура фреймворка

На рис. 1 представлена схема работы всего фреймворка. На вход Perfstat подаются входные данные (Data), загрузчик (Parser) преобразует их во внутренний формат фреймворка (Dataframe), затем данные фильтруются (Filter) и обрабатываются (Processor), возможно применение нескольких фильтров и операций к данным, далее средствами фреймворка по заданным пользователям параметрам создаются различные диаграммы, графики, текстовые поля и таблицы (Drawer). На выходе получается отчет, скомпонованный из графических элементов.

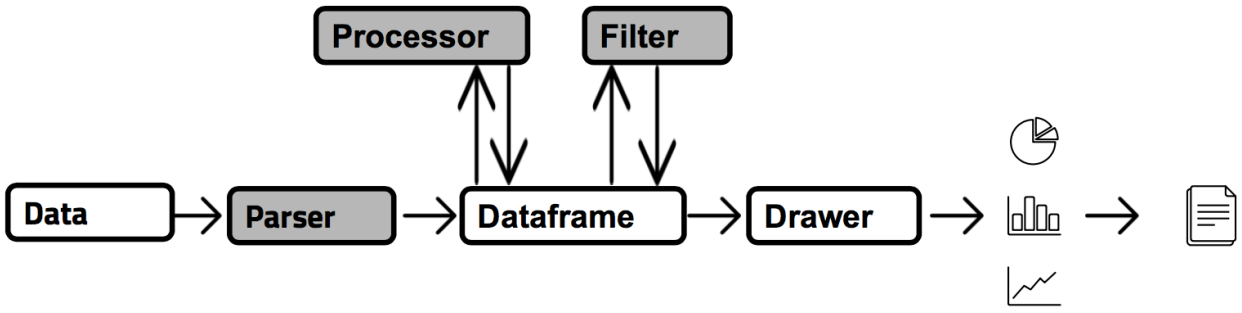

Рис. 1: Схема работы фреймворка

Моей задачей в рамках проекта была реализация модуля, ответственного за загрузку и обработку данных. Модуль состоит из двух

частей: загрузчика (Parser) и обработчика (Processor и Filter), далее будет подробный обзор этих элементов.

3.3. Загрузчик

Для того чтобы визуализировать данные, необходимо их сначала загрузить и преобразовать во внутренний формат фреймворка. На рис. 2 представлена архитектура загрузчика. Есть главный базовый класс, каждый из наследников которого отвечает за чтение конкретного формата.

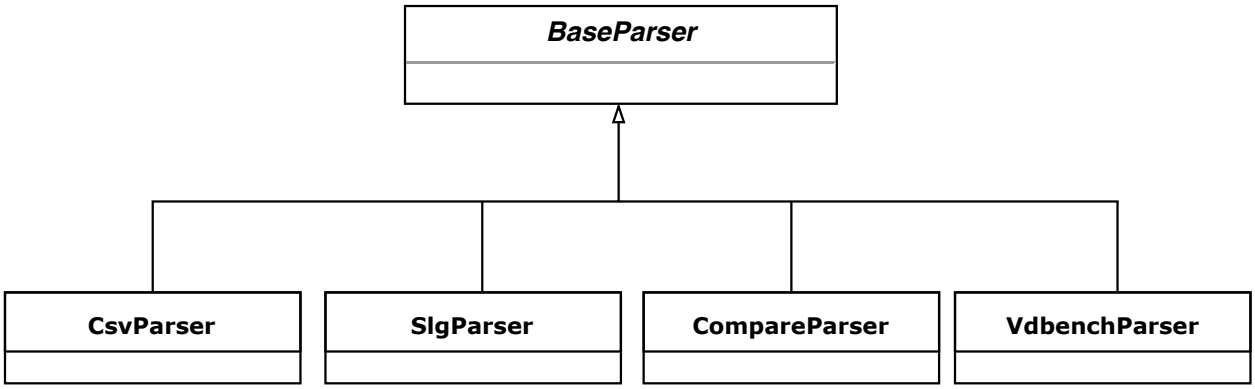

Рис. 2: Архитектура загрузчика

Существует множество форматов результатов автоматизированных тестирований, для написания загрузчика было решено выбрать основные форматы:

- csv - самый распространенный формат представления результатов автоматизированного тестирования;
- slg - файлы, в которых содержится статистика по каждому элементу тестирующейся системы. При разработке загрузчика этого формата была учтена синхронизация результатов по времени: пользователь может сам задавать начало и шаг тестирования;
- compare - файлы, в которых содержатся результаты анализа поведения системы при компрессии или дедупликации. Данные по

двум видам тестирования находятся в разных файлах, соответственно, был реализован считыватель данных, который позволит в дальнейшем сравнивать результаты двух тестов между собой;

- vdbench - файлы, порожденные утилитой vdbench - генератором рабочей нагрузки для измерения производительности хранилища. Работа класса vdbenchParser состоит из вызова утилиты vdbench и трансформирования ее выходных результатов в формат фреймворка;

Помимо методов, отвечающих непосредственно за чтение данных и перевод их во внутреннее представление фреймворка, в загрузчиках было реализовано чтение сразу нескольких файлов, названия которых соответствуют регулярному выражению, заданному пользователем. Это удобно для комбинирования анализа результатов нескольких тестирований в одном отчете.

Архитектура расширяемая: если нужно считывать новый формат данных, инженеру необходимо только отнаследоваться от базового класса и написать собственный загрузчик требуемого формата, преобразующий исходные данные во внутренний формат фреймворка, оставив остальные компоненты системы без изменений.

3.4. Обработчик

Часто необработанные данные являются нерепрезентативными для проведения анализа тестов, соответственно, стояла задача разработки обработчика данных. На рис. 3 показана архитектура компонента. Аналогично загрузчику, имеется главный базовый класс, каждый из наследников которого отвечает за определенный вид операций над данными. Согласно основным типам действий, которые можно производить над результатами тестирований (группировка, фильтрация и применение математических операций), мною были реализованы следующие классы-наследники:

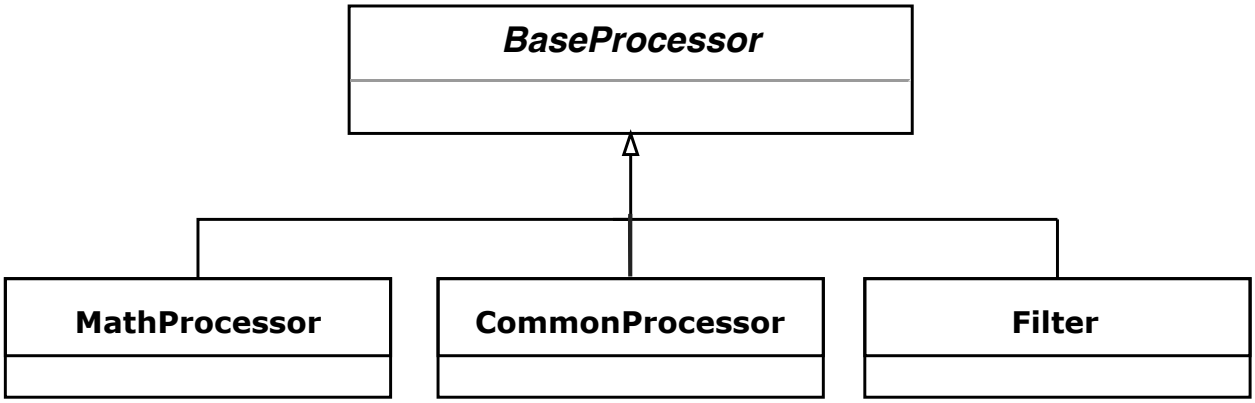

Рис. 3: Архитектура обработчика

- CommonProcessor - отвечает за трансформацию и группировку данных, обладает следующими функциональностями:
  - получение датафреймов для уникальных сущностей по какой-либо колонке в виде словаря вида: {уникальное значение: датафрейм}. Данная функциональность полезна для визуализации данных отдельно по каждому элементу тестирования и дальнейшего анализа отдельных элементов на общем фоне результатов системы;
  - группировка данных по значению в какой-либо колонке для последующего вычисления статистики - анализ всей системы в целом и сравнение результатов по каждому элементу тестирования с показателями всей системы;
  - объединение нескольких датафреймов в один - полезно для сравнительного анализа данных, полученных из разных файлов;
- MathProcessor - класс для подсчета статистики, позволяет производить ряд математических операций над данными:
  - нахождение минимума, максимума, среднего значения или суммы элементов;

  - округление;
  - отнять из каждой строки предыдущую;
  - применить свою функцию к данным: если инженеру нужно произвести какую-то определенную операцию над данными, он может написать свою функцию, указать для нее входные параметры и получить результат, который затем можно будет добавить в отчет;

- Filter - класс, отвечающий за фильтрацию данных, позволяет:
  - фильтровать данные по численному значению, используя операции сравнения;
  - фильтровать данные по строковому значению, используя регулярные выражения - полезно, если необходимо вывести результат только по системам с определенными именами;
  - убрать первые и/или последние нулевые строки - обычно необходимо удалить несколько нулевых начальных значений, так как в начале тестирования системе нужно время, чтобы разогнаться.

Заключение

В ходе работы были получены следующие результаты:

- проведено исследование предметной области и сделан обзор существующих решений;
- разработан расширяемый загрузчик данных в различных форматах:
  - csv;
  - vdbench;
  - slg;
  - compare;
- разработан обработчик, позволяющий:
  - применять различные фильтры;
  - выполнять математические операции;
  - группировать и трансформировать входные данные;
- модуль встроен во фреймворк;

В дальнейшем планируется расширить функциональность обработчика, поддержать больше входных форматов, а также разработать элемент, отвечающий за непосредственно анализ данных (детекция пиков, трендов, прогнозирование).

Список литературы

[1] Education EMC. Information Storage and Management: Storing, Managing, and Protecting Digital Information / Ed. by EMC Education Services. — 2nd edition.

[2] Microsoft. Excel // Microsoft Office official page. — Access mode: `https://products.office.com/ru-ru/excel` (online; accessed: 14.05.2018).

[3] Microsoft. Power BI // Power BI official page. — Access mode: `https://docs.microsoft.com/ru-ru/power-bi/` (online; accessed: 14.05.2018).

[4] Pandas // Pandas official page. — Access mode: `https://pandas.pydata.org` (online; accessed: 14.05.2018).

[5] Python 3 // Python official page. — Access mode: `https://docs.python.org/3/` (online; accessed: 14.05.2018).

[6] Qlik. Qlik View // Qlik official page. — Access mode: `https://www.qlik.com/ru-ru` (online; accessed: 14.05.2018).

[7] SAP. Crystal Reports // SAP official page. — Access mode: `https://www.sap.com/products/crystal-reports.html` (online; accessed: 14.05.2018).

[8] Software Tableau. Tableau // Tableau official page. — Access mode: `https://www.tableau.com` (online; accessed: 14.05.2018).