Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/09BE12Stanivuk

ДИСТРИБУИРАНИ СОФТВЕРСКИ СИСТЕМ ЗА НАДГЛЕДАЊЕ ПЕРФОРМАНСИ ЈАВА АПЛИКАЦИЈА

DISTRIBUTED SOFTWARE SYSTEM FOR PERFORMANCE MONITORING OF JAVA APPLICATIONS

Страхиња Станивук, Милан Видаковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај: *Овај рад обухвата спецификацију и развој вишеклијентског дистрибуираног система за надгледање перформанси јава апликација. Преузимање метрика перформанси је имплементирано уз ослонац на јавино проширење за управљање (JMX). У систем су интегрисана два софтверска решења отвореног кода, Prometheus, за складиштење метрика, и Grafana, за њихову визуализацију. Додатно, систем обезбеђује алармирање корисника у случају прекорачења граничних вредности метрика. Спецификација је представљена UML дијаграмима.*

Abstract: *This paper covers the specification and development of a multi-tenant distributed system for performance monitoring of java applications. Fetching of performance metrics is implemented with reliance on Java Management Extension (JMX). Two open source software solutions are integrated into the system, Prometheus, as metric storage, and Grafana, for metric visualization. In addition, the system provides user alerts in case of exceeding the metric threshold values. The specification is represented by UML diagrams.*

Кључне речи: *JMX, Performance, Java, Microservice, Prometheus, Grafana, Multi-tenant, Alerting.*

1. УВОД

Изузетно важан сегмент у животном циклусу софтверског решења, током развоја, а нарочито током употребе и одржавања, чини надгледање перформанси система. Исправна употреба и квалитет система за надгледање перформанси за компанију може значити разлику између успеха и пропасти. Овај рад ће се фокусирати на надгледање перформанси апликација писаних у Јава програмском језику, уз ослонац на Јава виртуелну машину, JVM (енгл. *Java Virtual Machine*).

Јавина виртуелна машина нам посредством проширења за управљање, JMX (енгл. *Java Management Extension*), омогућава приступ метрикама надгледане апликације, а и измену извршног кода [1].
Као инспирација за развој система описаног у овом раду послужила је десктоп апликација *JConsole* која имплементира JMX апликациони програмски интерфејс, API (енгл. *Application programming interface*) [2].

НАПОМЕНА:
Овај рад је проистекао из мастер рада чији ментор је био проф. др Милан Видаковић.

2. ОПИС СИСТЕМА

Дистрибуирани систем за надгледање перформанси јава апликација чини више слабо повезаних компоненти које, узајамним радом, теже ка остваривању заједничког циља, а то је приказ тренутног и историјског стања у којем се апликација, или више њих, налазе.
Систем је дизајниран да подржи вишеклијентски (енгл. *multi-tenant*) режим рада. У оваквим софтверским решењима, сви клијенти користе исту инстанцу апликације, док се изолација постиже на нивоу података. Описани систем би клијентима био пружен по моделу "Софтвер као услуга", SaaS (енгл. *Software as a Service*), који је један од три основна типа рачунарства у облаку (енгл. *Cloud computing*). Овај модел омогућава брзу интеграцију у постојећи софтверски екосистем клијента, ниске иницијалне трошкове и приступ подацима са било ког уређаја који има приступ интернету.

Компоненте система су:

- Јава апликација за преузимање метрика
- База података за смештање метрика
- Веб апликација за визуализацију метрика

3. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

Java Management Extension

Јавино проширење за управљање (JMX) пружа могућност управљања и надгледања апликација, уређаја па и саме јава виртуелне машине, локално или на даљину. Овај алат се повезује на ресурсе за време њиховог извршавања и у стању је да измени извршни кôд. Без обзира на коришћени протокол, ове компоненте пружају исти интерфејс за управљање, захваљујући чему је омогућено транспарентно управљање ресурсима. Омогућава лако конфигурабилну, скалабилну и поуздану инфраструктуру, при чему је инструментација потпуно независна од имплементације надгледаних ресурса.

Архитектура јавиног проширења за управљање се састоји из три нивоа. Ниво инструментације дефинише поставку ресурса како би им апликације за управљање могле приступити и надгледати их. За инструментацију користимо јава објекте под називом *Managed Beans*, или краће *MBeans*. Они се региструју у *MBean* серверу, који се понаша као управљачки агент на надгледаној апликацији, и који припада другом, агент нивоу. Компонента која је посебно

занимљива је JMX *Connector* који омогућава приступ поменутом агенту са удаљене апликације. Она се налази на нивоу дистрибуираних сервиса који одређује комуникацију између апликација за управљање и JMX агената [3].

Prometheus

Потреба за надгледањем многобројних микросервиса у оквиру једне апликације била је мотив за стварање овог софтверског решења.

Захваљујући имплементираној подршци за велики број извозника (енгл. *exporters*) Prometheus је један од водећих софтверских система за мониторинг. Извозници омогућавају једноставно ишчитавање и прослеђивање метрика надгледаних ресурса, биле то базе података, системи за управљање порукама или пак апликације развијене у различитим програмским језицима [4].

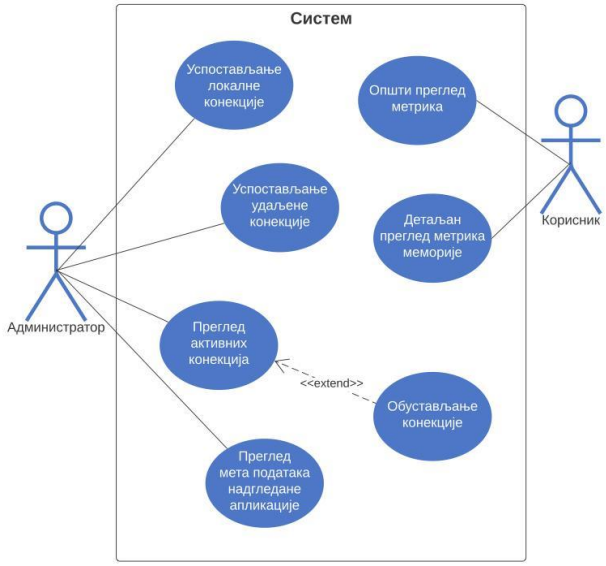
Grafana

Grafana је алат који олакшава проучавање, анализу и надгледање података кроз одређени временски период, а све то помоћу подесивих контролних табли. Она омогућава повезивање са великим бројем извора података а за сваки од њих обезбеђен је прилагођен упит за претрагу као и специфична синтакса, међу њима су *Graphite*, *Prometheus*, *Influx DB*, *MySql* и многи други. Помоћу овог алата омогућава се праћење првенствено стања апликације а потом и понашања корисника [5].

4. СПЕЦИФИКАЦИЈА АПЛИКАЦИЈЕ

Израда система за надгледање перформанси обухвата имплементацију апликације за мониторинг метрика, назване *Tracer*, као и конфигурацију два софтверска решења отвореног кода *Prometheus* и *Grafana*. У сврхе демонстрације израђена је и тест апликација (у даљем тексту надгледана апликација или *Monitored App*) која ће симулирати заузеће меморије и процесора, а чије метрике ће *Tracer* надгледати.

Дијаграмом случајева коришћења, датом на слици 1, представљени су корисници система и активности којима они имају приступ.



Слика 1 - Дијаграм случајева коришћења

Администратор система има могућност успостављања локалне и удаљене конекције ка надгледаној апликацији. Такође, постоји опција излиставања и обустављања активних конекција.

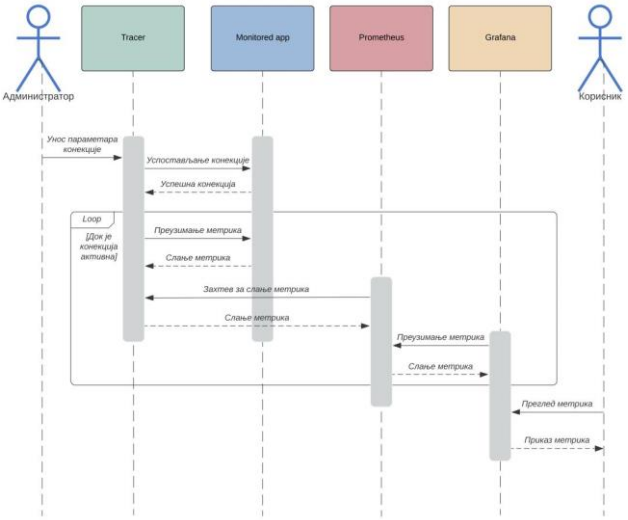
Администратор исто тако има опцију да добави мета податке оперативног система и јава виртуелне машине надгледане апликације. Корисник система има могућност прегледа метрика надгледаних апликација у реалном времену или у интервалу из прошлости.

Дијаграм секвенце приказан на слици 2, даје нам увид у хронолошки редослед операција обухваћених процесом успостављања удаљене конекције и прегледом метрика надгледане апликације.

У првом кораку администратор успоставља конекцију са надгледаном апликацијом. Затим започиње кружни процес у којем *Tracer* преузима метрике од надгледане апликације, потом их шаље ка *Prometheus*-u, да би коначно оне стигле на *Grafanu*.

Овај процес се понавља у одређеном временском интервалу од 1000ms све док је конекција између *Tracer*-а и надгледане апликације активна.

У последњем кораку корисник путем веб апликације прегледа метрике које се ажурирају у реалном времену.

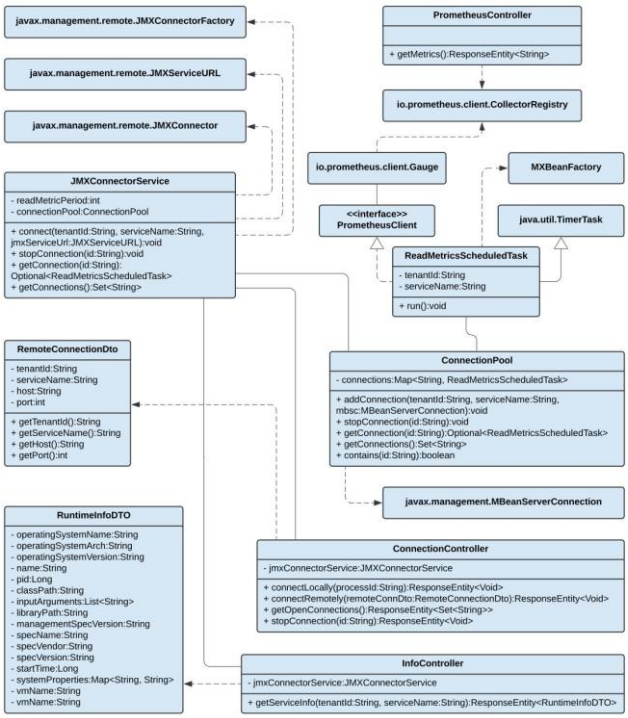


Слика 2 - Дијаграм секвенце

Дијаграмом класа, који је приказан на слици 3, дат је детаљан увид у организацију и интеракције између класа компоненте система *Tracer*.

Централно место заузимају класа *JMXConnector-Service* која је задужена за успостављање конекције са надгледаном апликацијом и *ConnectionPool* класа која води рачуна о активним конекцијама.

Конекција је представљена *ReadMetricScheduledTask* класом која периодично преузима метрике надгледане апликације.



Слика 3 - Дијаграм класа

Како је наведено у наслову рада, решење је имплементирано као дистрибуирани софтверски систем, пратећи микросервисну архитектуру. Њега чини скуп компонената тј. сервиса које крајњи корисник доживљава као јединствену апликацију. Овакви системи се скоро без изузетака испоручују преко платформи које пружају инфраструктуру за рачунарство у облаку.

Нека од примењених готових решења (енгл. *design patterns*) или пројектних образаца микросервисне архитектуре у систему су:

- Декомпозиција на поддомене - систем је издељен на целине које имају јасне границе овлашћења и одговорности. Сваки микросервис је изграђен око једне целине.
- *Sidecar* - патерн описује ситуацију када се помоћна апликација испоручује уз главну апликацију, пружајући додатне функционалности. У овом систему, компонента *Tracer* се као додатак испоручује уз надгледану апликацију, омогућавајући прикупљање метрика.

5. ИМПЛЕМЕНТАЦИЈА

Систем чине три компоненте. Имплементација обухвата комплетну израду јава апликације *Tracer*, задужене за мониторинг метрика а која је заснована на JMX технологији. Такође, ово поглавље описује и конфигурацију софтверских решења отвореног кода *Prometheus* и *Grafana* који представљају преостале две компоненте.

Tracer

Tracer је компонента система имплементирана у јава програмском језику, уз ослонац на JMX технологију која омогућава повезивање на удаљену апликацију у циљу управљања или мониторинга.

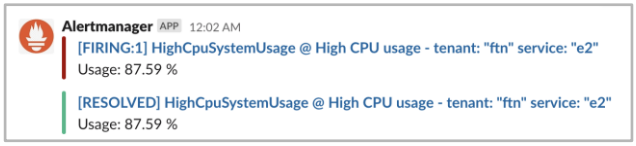
Класа *JMXConnectorService* апстрахује коришћење класа из пакета јавиног проширења за управљање. Одговорна је за успостављање конекција као и преглед и обустављање активних конекција. Конекција са удаљеном надгледаном апликацијом се успоставља помоћу хост и порт параметара прослеђених кроз интерфејс *ConnectionController* класе. При томе тражени порт није онај на којем се апликација извршава, већ порт који је резервисан за комуникацију са JMX агентом а задаје се кроз опције виртуелне машине при покретању надгледане апликације.

Активна конекција је представљена инстанцом класе *ReadMetricsScheduledTask*. Она наслеђује класу *TimerTask* из *java.util* пакета која омогућава извршавање акције у периодичним циклусима. Такође, имплементира интерфејс *PrometheusClient* путем којег се региструју колектори за сваку од надгледаних метрика понаособ. Приликом инстанцирања објекта ове класе позивају се методе класе *MBeanFactory* које отварају посредну везу са *MXBean*-овима надгледане апликације, а из којих се потом извлаче метрике.

Задужење класе *PrometheusController* је обрада захтева за преузимање метрика који стижу од *Prometheus* компоненте система. Метрике смештене у колекторима се ишчитавају и у одговарајућем текстуалном формату враћају као одговор на захтев.

Prometheus

Prometheus, готово софтверско решење, је уз потребно конфигурисање интегрисано у систем за надгледање перформанси. Поред минималне конфигурације, додатно је подешен део система за одашиљање узбуна, *Alertmanager*. Минималним подешавањима је дефинисан интервал преузимања метрика, интервал евалуације услова узбуњивања, као и путање до апликација од којих ће метрике бити преузимане. Дефинисана су два правила узбуњивања. Прва узбуна се окида у случају да метрика која представља оптерећење процесорске јединице било које надгледане апликације пређе 80%, а параметризована порука говори о којем кориснику и сервису се ради. Друга се односи на компоненту система Tracer и окинуће се у случају да је дошло до застоја у раду или пада овог дела система. Додатно, дефинисан је примаоц узбуна, алат за тимску колаборацију, *Slack*, и шаблон послате поруке. Изглед примљене поруке о узбуни приказан је сликом 4.



Слика 4 - Порука о узбуни на платформи Slack

Grafana

Приликом интеграције софтверског решења отвореног кода, *Grafana*, дефинисан је *Prometheus* као извор метрика и интервал преузимања. Конфигурисане су контролне табле и графикони на њима тако да визуализују метрике од значаја.

Визуализација метрика обавља се преко две контролне табле: општи преглед метрика сервиса и детаљан преглед метрика меморије. Графикони на првој табли (слика 5) су подешени тако да приказују заузеће процесорске јединице, укупно заузеће меморије, број учитаних класа и активних нити, време активности надгледане апликације и број процесорских језгара.

Слика 5 - Контролна табла са општим метрикама

Графикони на табли са детаљима меморије (слика 6) дају нам увид у заузеће Heap и Non-Heap меморијских простора. Део меморије под називом Heap је одговоран за смештање свих објеката креираних од стране покренутог процеса. Меморијски простор Heap је физички подељен на два дела, такозване генерације, младу (енгл. *young generation*) и стару (енгл. *old generation*). Млада генерација се састоји из два сегмента, *Eden* и *Survivor*, овај простор је резервисан за нове објекте.

Слика 6 - Контролна табла са метрикама меморије

6. ЗАКЉУЧАК

Овим радом представљен је вишеклијентски, дистрибуиран и робустан систем који, ослањајући се на могућности JMX технологије и путем богатог графичког интерфејса, пружа увид у перформансе надгледаних апликација. Додатно, систем обезбеђује алармирање у случају прекорачења граничних вредности.

Описан софтверски систем је развијен у актуелним технологијама и прати индустријски стандард за излагање, транспорт и визуализацију метрика. Због своје модуларне архитектуре систем је погодан за даља проширења и усавршавања.

У систему су коришћени искључиво предефинисани *MBean* објекти који пружају информације о стању и заузећу физичких ресурса, којима надгледана апликација располаже. Наредни корак ка унапређењу би могао бити креирање прилагођених *MBean* објеката који би омогућили инспекцију специфичних метода или делова кода од интереса.

Овим би се отворио простор за надгледање четири златна сигнала, препознатих од стране инжењера Гугла. У питању су латенција или време потребно за обраду захтева, оптрећење тј. број захтева у јединици времена, стопа грешака са њиховим порукама и засићење, које је управо и представљено овим радом кроз приказ заузећа централне процесорске јединице, радне меморије и складишног простора.

7. ЛИТЕРАТУРА

[1] Oracle Monitoring and Management Guide, *https://docs.oracle.com/en/java/javase/13/management/*
[2] Using JConsole, *https://docs.oracle.com/en/java/javase/13/management/using-jconsole.html*
[3] Architecture of the JMX Technology https://docs.oracle.com/javase/tutorial/jmx/overview/architecture.html
[4] Prometheus exporters and integrations https://prometheus.io/docs/instrumenting/exporters/
[5] Grafana https://grafana.com/

Кратка биографија:

Страхиња Станивук, рођен 14. XII 1992. у Бачкој Тополи, где завршава основну школу “Никола Тесла”. Средње образовање стиче у електротехничкој школи “Михајло Пупин” у Новом Саду 2011. године када уписује основне академске студије на Факултету техничких наука у Новом Саду, смер Рачунарство и аутоматика. Мастер студије на истом факултету наставља на модулу Електронско пословање, 2016. године. Упоредо са мастер студијама бави се радом у привреди, са посебним интересовањем за развој високоперформантних пословних софтверских система.

Милан Видаковић је рођен у Новом Саду 1971. године. На Факултету техничких наука у Новом Саду завршио је докторске студије 2003. године. На истом факултету је 2014. године изабран за редовног професора из области *Примењене рачунарске науке и информатика*.