РАЗВОЈ АПЛИКАЦИЈЕ ЗА ЕЛЕКТРОЕНЕРГЕТСКЕ ПРОРАЧУНЕ У ОКВИРУ AWS ОКРУЖЕЊА**DEVELOPMENT OF AN APPLICATION FOR ELECTRIC POWER CALCULATIONS WITHIN THE AWS ENVIRONMENT**Марија Томић, *Факултет техничких наука, Нови Сад***Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО**

Кратак садржај – Овај рад се бави испитивањем перформанси микросервисне апликације која врши тополошке прорачуне у AWS cloud окружењу и њиховим поређењем са перформансама при раду у Azure окружењу.

Кључне ријечи: AWS, Cloud, Azure, микросервиси, електроенергетски систем, тополошки прорачуни

Abstract – This paper examines the performance of a microservice application that performs topological calculations in an AWS cloud environment and compares them to performance when running in an Azure environment.

Keywords: AWS, Cloud, Azure, microservices, electroenergy system, topological calculations

1. УВОД

Велике компаније раде са великим количинама података које је потребно складиштити. Раде се притом и многи прорачуни и анализе система и његових података. Зато је потребно увијек тежити механизмима и технологијама са којима се постиже брза обрада и сигурно и трајно чување велике количине података.

Cloud computing или рачунарство у облаку представља технологију која податке, софтвер или хардвер испоручује кориснику као услугу, а не као производ [1].

Подаци и апликације се налазе на удаљеној локацији и најчешће се испоручују путем Интернета, што значи да ће подаци увијек и са сваке локације бити доступни. Лиценца за коришћење Cloud сервиса се не купује већ се изнајмљује на мјесечном нивоу.

Како је употреба рачунарства у облаку расла, тако је растао и развој Cloud сервиса који се у ту сврху користе. Данас у свијету постоји велики број компанија које се баве развојем Cloud сервиса, од којих свака нуди своје сервисе и услуге.

Ти сервиси углавном обављају исте или сличне функције, само на различите начине, у другачијим окружењима и под другачијим условима.

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Дарко Чапко, ред. проф.

С друге стране, све те компаније унутар себе имају мноштво различитих сервиса који нуде различите услуге и обављају задатке потребне за нормално функционисање апликација у Cloud окружењу.

1.1. Прелазак са монолитне на микросервисну архитектуру

Да би се избјегли проблеми као што је слаба стабилност система, опадање перформанси или спор развојни циклус, потребно је на вријеме размишљати о промјенама и преласку са монолитне на микросервисну архитектуру.

Најчешћи проблеми који се јављају у монолитним апликацијама настају због потенцијалних уских грла која настају под великим оптерећењем. На примјер, преоптерећење при обради споредних података у некој Web апликацији може довести до обарања цијелог сајта. Споредни подаци су подаци који не представљају главни садржај Web странице или апликације. У монолитним апликацијама сав садржај се учитава у исто вријеме тако да је тешко раздвојити главне од споредних података. У том случају, када падне дио система који се бави обрадом споредних података он са собом повуче и остатак апликације. Промјене на једном дијелу система утичу на неки други дио система иако они нису логички повезани и не очекује се да се то догоди.

Један од разлога зашто се не прелази са монолитне на микросервисну архитектуру је тај што је промјена технологије претешка. Многе монолитне апликације зависе од застарјелог софтвера.

Сви проблеми у монолитним апликацијама подижу се на један виши ниво уколико се налазе на Cloud-у. Уколико се деси неки проблем на одређеном дијелу система, након рјешавања проблема и преправљања кода потребно је сваки пут изнова publish-овати цијелу апликацију, без обзира на то што се можда промјенио само један мали дијелић. Ако систем има више клијената који користе апликацију рад на свим клијентима ће бити обустављен док се проблем не отклони, а самим тим перформансе система опадају.

Већина наведених проблема се смањује разбијањем једног великог проблема на више мањих, тј. подјелом монолитне апликације на више мањих сервиса.

1.2. Коришћење Cloud-a

Cloud замјењује скупу IT инфраструктуру, омогућава већу флексибилност у раду, једноставност и сигур-

ност за пословне процесе уз значajnu уштеду. Гарантује потпуну приватност и безбједност података уз најсавременије системе заштите. Има аутоматски *backup* који омогућава поуздан опоравак система у случају неких поремећаја и кварова. Подацима се може приступати са било ког уређаја, битно је само да је тај уређај повезан на Интернет. Међу најпопуларнијим Cloud платформама данас су *Microsoft Azure Service Fabric* и *Amazon Web Services (AWS)*.

2. AMAZON WEB SERVICES

Amazon Web Services је скуп удаљених рачунарских сервиса који заједно граде платформу за рачунарско коришћење у Cloud-у, а коју Amazon нуди путем Интернета [2].

Најпознатији сервиси су *Amazon EC2* (обезбјеђује виртуелне машине) и *S3* (обезбјеђује капацитете за складиштење). Још један веома значајан сервис који се обично користи у комбинацији са наведена два, јесте *Elastic Beanstalk*, помоћу кога се апликације могу брзо и лако *deploy*-овати и управљати без претјеране потребе за познавањем инфраструктуре саме апликације.

AWS је најопсежнија и широко прихваћена Cloud платформа која нуди око 165 услуга из центара података широм свијета. Милиони купаца, укључујући највећа предузећа и водеће агенције вјерују AWS-у и повјеравају му да унапређује њихову инфраструктуру, да смањује трошкове и повећава агилност. У последњих десет година откако је објављен, AWS је израстао у водећу компанију за Cloud инфраструктуру која, према истраживањима организације *Synergy Research Group*, снабдијева 34% Cloud тржишта, док његови најближи ривали - Microsoft, IBM и Google заједно снабдијевају око 24% тржишта [3].

AWS сервисима се може управљати путем корисничког интерфејса заснованог на Web-у (*AWS Management Console*) и преко интерфејса командне линије (CLI) или програмски путем SDK-а. Најбоље мјесто за почетак рада или за експериментисање са AWS-ом јесте *AWS Management Console*. Ова конзола је веома прегледна и погодна за кориснике који су почетници у коришћењу AWS-а јер нуди брз и јасан преглед његових сервиса. Такође је погодна за подешавање жељене инфраструктуре и покретање нових виртуелних машина, инстанци итд. На слици 1 приказана је почетна страница *AWS Management Console*.

Слика 1. Изглед почетне странице *AWS Management Console*

AWS има интегрисане блокове који подржавају развој микросервиса. Два популарна приступа која се користе су *AWS Lambda* и *AWS Fargate*.

AWS Lambda омогућава покретање кода без обезбјеђивања или управљања серверима. Са овим сервисом, корисник је дужан само да једноставно пренесе свој код, а *AWS Lambda* се брине за све што је потребно за покретање и скалирање извршења. *AWS Lambda* заправо представља *serverless* архитектуру. *Serverless* архитектура је начин развоја и покретања апликација и сервиса без потребе за одржавањем инфраструктуре.

AWS Fargate је *serverless* рачунарска машина за контејнере који раде са *Amazon Elastic Container Service (ECS)* и *Amazon Elastic Kubernetes Service (EKS)*.

С обзиром на потребе истраживања овог рада, неће се користити уобичајени приступ развоја микросервиса у AWS. Потребно је имплементирати систем тако да корисник сам брине о броју инстанци, *load balancing*-у итд. Такав систем ће бити реализован помоћу *Amazon Elastic Beanstalk* и *Amazon EC2* сервиса који ће бити описани у наставку.

AWS нуди „*pay-as-you-go*“ приступ плаћања за преко 160 сервиса. Наплаћују се само појединачне услуге које се користе, за период који су коришћене, без икаквих дугорочних уговора или лиценцирања. Рачун од AWS-а је сличан рачуну за струју јер се наплаћује само вријеме када су сервиси или виртуелне машине биле покренуте.

Неки од најпопуларнијих AWS сервиса који се данас користе су: *Amazon S3*, *Amazon Elastic Compute Cloud (EC2)*, *AWS Lambda*, *Amazon Glacier*, *Amazon SNS*, *Amazon CloudFront*, *Amazon Elastic Block Store (EBS)*, *Amazon Kinesis*, *Amazon Virtual Private Cloud (VPC)*, *Amazon Simple Queue Storage (SQS)*, *Amazon Elastic Beanstalk*, *DynamoDB*, *Amazon Relational Database Service (RDS)*, *Amazon ElastiCache*, *Amazon Redshift*.

2.1. Elastic Beanstalk

Amazon Elastic Beanstalk је најатрактивнији сервис за програмере. Програмери више не морају да брину о инфраструктури и могу да се фокусирају на развој свог софтвера или апликације. Потребно је само да програмер *upload*-ује апликацију на Cloud, а *Elastic Beanstalk* аутоматски обрађује детаље о обезбјеђивању капацитета, скалирању, *load balancing*-у и надгледању здравља апликација. Корисник има пуну контролу над AWS ресурсима који напајају његову апликацију и може у било ком тренутку да им приступи. Може се управљати *Elastic Beanstalk*-ом помоћу *AWS Management Console*, AWS интерфејса командне линије (AWS CLI) или CLI високог нивоа дизајнираног посебно за *Elastic Beanstalk*.

Компаније које тренутно користе овај сервис су: *Zillow*, *Prezi*, *BMW*, *GeoNet*, *League of Women Voters*, *Samsung*, *Crowd Chat* итд. *Elastic Beanstalk* се не наплаћује – наплаћују се само ресурси (нпр. EC2 инстанце или *S3 Buckets*) који су потребни за складиштење и покретање апликација. *Elastic Beanstalk* је најбржи и најлакши начин за подизање апликација на AWS. За неколико минута, апликација

ће бити спремна за употребу. Корисник се може фокусирати на писање кода и не трошити вријеме на управљање и конфигурирање сервера, база података, *load balancer*-а и мрежа. На слици 2. приказан је ток рада *Elastic Beanstalk*-а.

Слика 2. Ток рада *Elastic Beanstalk*-а

Прво је потребно креирати *Elastic Beanstalk* апликацију. Након креирања апликације, потребно је *upload*-овати жељену верзију са локалне машине, након чега се прави и покреће окружење у којем ће *upload*-ована верзија да ради. Након тога може да се управља датим окружењем, након чега се *upload*-ују нове верзије апликације и процес покретања апликације се понавља.

2.2. Elastic Compute Cloud (EC2)

EC2 може се користити за покретање виртуелних сервера у складу са корисниковим захтјевима. *Amazon EC2* је централни дио цијелог AWS екосистема. Омогућава скалабилне рачунарске ресурсе на захтјев, омогућавајући да се AWS окружење лако прошири и смањи у неколико минута. Због своје велике флексибилности, EC2 још може лако да се интегрише са другим AWS сервисима као што је S3, VPC или RDS.

3. AZURE SERVICE FABRIC

Azure Service Fabric је дистрибуирана системска платформа која олакшава паковање, распоређивање и управљање скалабилним и поузданим микросервиси-ма и контејнерима. *Service Fabric* представља платформу нове генерације за изградњу и управљање пословним класама и Cloud базираним апликацијама које раде у контејнерима [4]. *Azure Service Fabric* омогућава стварање и управљање апликацијама заснованим на микросервиси-ма. Ти микросервиси раде на заједничком скупу машина, који се назива кластер. *Service Fabric* подржава лагано вријеме извршавања које подржава *statefull* и *stateless* микросервиси. *Stateless* микросервиси, као што су улазне тачке и проксији, не чувају промјенљива стања изван захтјева и одговора сервиса. *Statefull* микросервиси, попут корисничких налога, база података, шопинг картица и редова, чувају промјенљива стања изван захтјева и одговора сервиса. Актор (*actor*) је изолована, независна јединица која је намјењена за ситне прорачуне или промјене стања и која се извршава на једној нити [4].

Service Fabric Reliable Actor је имплементација актор дизајн патерна. Актор дизајн патерн представља модел за конкурентне и дистрибуиране системе у којима велики број актора може да се извршава истовремено и независно један од другог. Актори се најчешће користе када проблем укључује велики број (чак и по више хиљада) малих, независних и изолованих јединица стања или логики, или када се

жели радити са *single-threaded* објектима који не захтјевају значајну интеракцију са спољашњим елементима. *Service Fabric* је прилагођен да креира изворне Cloud сервисе који могу да покрену мале али и велике апликације са масовним подацима уколико је потребно. Данашње Интернет апликације се састоје од комбинације *statefull* и *stateless* микросервиса.

Service Fabric данас посједује многе *Microsoft*-ове услуге, као што су: *Azure SQL* база података, *Azure Cosmos DB*, *Cortana*, *Microsoft Power BI*, *Microsoft Intune*, *Azure Event Hubs*, *Azure IoT Hub*, *Skype for Business* и многе друге основне *Azure* услуге.

4. ИМПЛЕМЕНТАЦИЈА РЈЕШЕЊА

Апликација на којој се тестирају сервиси јесте .NET конзолна апликација заснована на микросервисној архитектури. Апликација се бави тополошким прорачунима у датом електроенергетском систему. Електроенергетски систем (ЕЕС) је технички систем чији је основни задатак да осигура квалитетну испоруку електричне енергије уз минималне трошкове у ЕЕС. Електроенергетске мреже се најчешће представљају графовима. Чворовима графа представљају се сабирнице високонапонских постојења или спојна места (рачве) у дистрибутивним мрежама, а гранама графа су моделовани водови и трансформатори. На мјестима гдје се спајају грана и чвор могу се налазити елементи графа којима се моделује расклопна опрема (прекидачи, растављачи итд.), релејна заштита и мјерни инструменти. Електродистрибутивна радијална мрежа се састоји од скупа коријена и острва. *Коријен* чине сви елементи који су напајани из једне напојне гране и они представљају енергизован дио мреже. Острва представљају скуп елемената који су међусобно повезани и са свих страна су изоловани од енергизованих дијелова мреже отвореним прекидачима. Топологија мреже представља логичку дефиницију међусобне повезаности опреме преко затворених прекидача. Тополошка анализа има задатак да одреди галвански повезане цјелине (коријене и острва) и прилагоди податке о повезаности мреже као и да снабдијева сваки елемент мреже потребним подацима, да би корисник имао увид у тренутно стање мреже. Након обављене тополошке анализе, модел који је освјежен представља основу за брзо и ефикасно извршење функција електроенергетских прорачуна [5]. Апликација врши тополошке прорачуне на захтјев корисника, по унапријед дефинисаном алгоритму обиласка графа дистрибутивне мреже, над задатом електроенергетском мрежном структуром. На слици 3 приказана је архитектура имплементираних рјешења.

Апликација садржи три микросервиса који врше највећи дио посла. *TopologyRequest* шаље корисников захтјев за прорачун са унапријед задатим параметрима. *TopologyManager* примјењује алгоритам за расподјелу коријена на пет листи гдје ће свака листа на крају имати исти број елемената и над сваком листом позива *TopologyAnalyzer* у оквиру кога је имплементирана тополошка анализа. Тополошка анализа се врши за сваки коријен посебно и обрађују се само неажурни коријени.

Слика 3. Архитектура имплементираниог рјешења

Апликација је раније имплементирана тако да ради у Azure окружењу и тестирана је локално [6]. За потребе овог истраживања тестирана је оптимизована верзија те апликације у *Azure Service Fabric* и AWS окружењу. Како би се на најбољи могући начин реализовала имплементација апликације у AWS-у, потребна је припрема у смислу одабира сервиса који ће се користити и њиховог обезбјеђивања. Након што је одлучено да ће се користити *Elastic Beanstalk* и EC2, да би се реализовала једна таква имплементација одрађена је конфигурација и припрема окружења на које ће се подизати жељени микросервиси. Најлакши начин за то је кроз *AWS Management* конзолу.

5. ТЕСТИРАЊЕ

Тестирана је апликација у Azure окружењу и апликација у AWS окружењу. Обе апликације су тестиране по 50 пута за одређен број коријена. Тестиране су апликације за тополошке прорачуне за 10, 20, 50, 100, 200, 300 и 400 коријена. Извучене су средње вриједности времена која су том приликом добијена. Резултати тестирања апликације у AWS и у Azure окружењу су приказани у табели 1.

Табела 1. Приказ резултата тестирања

Бр. коријена	Azure [s]	AWS [s]
10	2	2
20	3	3
50	7	8
100	14	26
200	42	68
300	82	94
400	131	166

Из табеле се види да се тополошка анализа брже извршава у Azure окружењу за већи број коријена. За мањи број коријена (10, 20, 50) вријеме извршавања прорачуна је исто, док се за већи број коријена времена доста разликују.

6. ЗАКЉУЧАК

Из резултата се види да микросервисна архитектура боље функционише на *Azure* платформи, тј. прорачуни тополошке анализе се брже извршавају на њој. Доказано је да се вријеме извршавања смањује коришћењем динамички подигнутих сервиса на *Azure Service Fabric* платформу. Из прикупљених информација током истраживања о теми може се закључити да је AWS погоднији за комплексније системе и апликације гдје корисници имају више простора и опција са којима раде.

7. ЛИТЕРАТУРА

- [1] G.Kulkarni, J. Gambhir, R. Palwe, "Cloud Computing-Software as Service", *International Journal of Cloud Computing and Services Science*, Vol. 1, pp. 11-16, Март 2012.
- [2] S. Nagaprasad, A. VinayaBabu, K.Madhukar, D. Marlene G Verghese, V. Mallaiah and A. Sreelatha, "Reviewing some platforms in cloud computing", *International Journal of Engineering and Technology*, Vol. 2, pp. 348-353, 2010.
- [3] Synergy Research Group, "The Leading Cloud Providers Continue to Run Away with the Market", <http://mng.bz/qDY0>, 2017.
- [4] <https://docs.microsoft.com/bs-latn-ba/azure/service-fabric/service-fabric-overview> (приступљено у новембру 2019.)
- [5] Ковач ТЖ, Бекут ДД, Сарић АТ. „Topological analysis of unbalanced distribution networks with single-phase switching equipment and temporary elements.“ *Int Trans Electr Energ Syst.*, 2017.
- [6] Н. Петковић, Динамичка расподела микросервиса у дистрибуираном систему, *ФТН Нови Сад*, 2018.

Кратка биографија:

Марија Томић рођена је у Бијељини 1996. године. Дипломирала је на Факултету техничких наука из области Електротехнике и рачунарства – Примјењено софтверско инжењерство 2018.године.