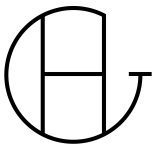

HoGent

Faculteit Bedrijf en Organisatie

Unikernels

Michiel De Wilde

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Jan Janssen

Instelling: Hogeschool Gent

Academiejaar: 2015-2016

2e examenperiode

Faculteit Bedrijf en Organisatie

Unikernels

Michiel De Wilde

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Jan Janssen

Instelling: Hogeschool Gent

Academiejaar: 2015-2016

2e examenperiode

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Voorwoord

Toen ik begon met programmeren had ik geen idee wat er gebeurde in de achtergrond van de computer. Ik starte met de simpele todolist-applicaties om alles te leren over programmeren. Na een tijd kwam ik een black box tegen: het besturingssysteem. Vooral om servers sneller te laten werken en de techniek erachter te leren kennen begon ik aan een zoektocht. Linux was de startplek bij uitstek. Package managers en file systems waren de eerste concepten die mij met verstomming lieten staan. Toen ik meer en meer naar infrastructuur keek begon ik termen te leren en alle handige tips om ervoor te zorgen dat je server altijd beschikbaar is. Toen een paar jaar geleden Docker voor het eerst echt vaart maakte met containers was ik verbaasd. Ik dacht eerst dat dit nooit zou werken. Na een tijd heb ik wel het licht gezien en gebruikte ik containers meer en meer. Toen er gevraagd werd om een onderwerp voor mijn thesis dacht ik meteen en wat volgens mij de volgende stap is: unikernels.

Inhoudsopgave

Hoofdstuk 1

Inleiding

De wereld is steeds in verandering. Dit is een zekere waarheid in de wereld van informatica. Kijk maar naar de veranderingen dat er gebeuren elke paar maanden op vlak van javascript frameworks. Best practices van vijf jaar geleden, kunnen bad practices zijn vandaag. Je moet mee met de deze stroom van veranderingen wil je relevant blijven. Dit is voor iedereen in uitdaging. In mijn mening is dit iets goed, blijven leren is een van de beste dingen die je kan doen.

Er zijn grote veranderingen aan het gebeuren op het vlak van infrastructuur. Een paar geleden was de een virtuele machine met een klassieke stack de normale gang van zaken. De verandering heeft hier ook voor veranderingen gezorgd. Een container was al een tijd een droom van vele mensen maar er was nog geen echte goede uitwerking. Tot Docker. Met Docker werden containers eenvoudiger om te gebruiken. Unikernels zitten nu in de situatie van containers vooraleer Docker opzetten kwam. Zal unikernels dezelfde revolutie ontketenen of zal het hand in hand kunnen leven met Docker? Docker heeft unikernel systems overgenomen begin vorige jaar en je kan de invloed al zien in hun releases.

Deze thesis focust niet enkel op unikernels. 1 Van de onderzoeksvragen gaat over de rol van systeembeheerder in de toekomst. Unikernels en Docker hebben een groot gevolg voor hun maar zijn nog andere factoren die een rol spelen zoals orchestration frameworks, registries, CI/CD en veel meer.

Ik zal mijn uiterste best doen om deze omgeving te beschrijven en aan te tonen wat er kan gebeuren in de toekomst maar er kunnen natuurlijk nieuwe technologieen te voorschijn komen. Dit is een antwoord op de onderzoeksvragen vanuit mijn standpunt en de huidige informatie beschikbaar.

1.1 Probleemstelling en Onderzoeksvragen

Unikernels zijn een een nieuwe stroom binnen het landschap van besturingssystemen. We hebben al aangehaald dat containers de nieuwe werkwijze is wanneer men applicaties wilt ontwikkelen en schaalbaar wilt maken. Unikernels gaat nog een stap verder. De systeembeheerders moeten dus mee met containers en de veranderingen ondergaan. De vraag is welke veranderingen er zich zullen voordoen wanneer unikernels op de markt komen. Zullen de competenties van de systeembeheerder veranderen? Wordt het opzetten van applicaties meer en meer eenvoudiger of juist niet? We kunnen wel spreken over de opvolger van containers maar is deze al werkbaar in de toekomst? Wat is de impact op beveiliging, meer bepaald aspecten als beschikbaarheid, autorisatie, integriteit en vertrouwelijkheid van gegevens?

Hoofdstuk 2

Methodologie

Hoofdstuk 3

Virtualisatie

Vroeger was de tijd dat je een computer kon gebruiken beperkt. Vooral bij de eerste computers had men problemen om programma's en concepten uit te werken omdat de computer door meerdere mensen gebruikt werd. Een voorbeeld vanuit een situatie uit die tijd is het ontwikkelen van een programma. De source code van een programma werd manueel ingegeven en als een job in een queue gezet. Pas een paar dagen later kon men de resultaten bekijken. Als je een kleine schrijf- en/of logische denkfout maakte dan kon je opnieuw beginnen. De grootste bijdrage tot de ontwikkelsnelheid van programma's is de lengte van de feedbackcyclus: hoe snel kan je je programma testen laten werken. Dit leidt tot een verlies van tijd en dus geld.

Timesharing werd uitgevonden om het verlies van tijd te beperken. Bij timesharing konden de gebruikers inloggen op een console en zo computer gebruiken. Dit was een technische uitdaging. De computer zou van de ene context naar de andere moeten kunnen veranderen. Timesharing en verschillende gebruikers op 1 computer zou de basis vormen voor de moderne besturingsystemen. 1 Van de nadelen van timesharing is de isolatie van twee verschiilende processen. Als er een kans bestaat dat ze elkaar kunnen beinvloeden is dit een heel groot probleem.

Een besturingsysteem moet kunnen werken op verschillende soorten hardware. Om al deze hardware te ondersteunen moet er een standaard zijn waarop een besturingssysteem kan gebouwd worden. Dit zou betekenen dat veel van de complexiteit van het besturingssysteem naar de hardware zou verhuizen. Spijtig genoeg gebeurt dit nog altijd niet. Dit is 1 van de voornaamste redenen waarom de grootte van een OS tussen de 200MB en 1GB kan liggen. Een aantal besturingssystemen doen niet de moeite om de vele soorten hardware apparatuur te ondersteunen. Zij tonen de specificaties die een hardware apparaat moet hebben wanneer je het besturingssysteem wil gebruiken.

Doorheen de tijd werden computers meer krachtig en applicaties konden niet alle middelen van de computer ten volle benutten. Dit zorgde voor de creatie van virtual resources. Deze gingen de fysieke hardware simuleren om zo verschillende applicaties tegelijkertijd te laten werken op dezelfde virtuele machine. Zo worden de middelen van een fysieke machine optimaal gebruikt. De algemene term voor dit concept is virtualisatie. Bij het virtualiseren van een server gaat men een fysieke server opdelen in verschillende kleine en geisoleerde delen. Deze delen kunnen dan gebruikt worden door verschillende gebruikers. De voordelen van het virtualiseren van een server zijn de volgende: financieel voordeel (men kan van 1 taak naar meerdere taken gaan op 1 server), het besparen van energie (minder servers gebruiken want ze worden beter benut), betere beschikbaarheid.

3.1 Hypervisor

Een hypervisor is een stuk software, firmware of hardware waarop een virtuele machine zich bevindt. De host machine zorgt voor de middelen zoals CPU, RAM, ... Elke virtuele machine die zich bevindt op de host machine zal dan gebruik maken van deze middelen. Doordat virtualisatie alomtegenwoordig werd in datacenters heeft het ervoor gezorgd dat de meeste logica ook meer kwam te liggen bij de hypervisor. Cloud computing komt zeer sterk opzetten en dit zorgt ervoor dat de meeste mensen niet in contact komen met hypervisors.

3.1.1 Hosted Hypervisors

Een hosted hypervisor zal zich bevinden op een het besturingssysteem van de host machine en heeft geen directe toegang tot de hardware. Dit heeft als voordeel dat de hardware niet zo belangrijk is maar zorgt voor een extra laag tussen de hardware en de hypervisor. Een goede regel is: "hoe minder lagen we hebben, hoe beter de performantie."Wanneer je een Ubuntu instantie hebt in een virtuele machine dan zal de computer de hypervisor zijn. De computer kan dan de virtuele machine beheren en veranderingen uitvoeren.

Voorbeelden van een hosted hypervisor zijn: Virtualbox en VMware Workstation.

3.1.2 Bare-metal Hypervisors

Een alternatief is een bare-metal hypervisor. Hierbij is er geen extra laag tussen de hardware en de virtuele machine. We hebben geen host besturingssysteem nodig omdat de hypervisor zich rechtstreeks bevindt op de hardware. Dit zorgt voor betere performantie, schaalbaarheid en stabiliteit.

3.2 Hardware Virtualization

De virtuele machine is een toepassing van de virtualisatie van de hardware. Een virtuele machine bevindt zich op een fysieke machine door gebruik te maken van een hypervisor. De nadelen van een virtuele machine als de kleinste eenheid is de schaalbaarheid en de veiligheid. Wanneer je je eigen server opstelde dan moet je een besturingssysteem kiezen. Meestal gaan we kiezen voor een Linux distributie. Daarna moet je de software installeren die ervoor zorgt dat de applicatie kan werken. Daaropvolgend moet je je server beveiligen. Het veranderen van de SSH-poort, zorgen dat de root-user niet te bereiken is. IPtables en firewalls opstellen. Voor dit te vergemakkelijken kunnen we gebruikmaken van een configuratie management tool (Chef, Puppet, Ansible...).

Meest bekende voorbeelden van bare-metal hypervisors zijn VMware ESXi, Microsoft Hyper-V en Xen.

3.3 Operating System-level Virtualization

Naast hardware virtualisatie kunnen we ook gebruikmaken van de virtualisatie van het besturingssysteem. Bij deze toepassing van virtualisatie gaan we de mogelijkheden van de kernel van het besturingssysteem gebruiken. De kernel van bepaalde besturingssystemen laat ons toe om meerdere geisoleerde user spaces tegelijkertijd te laten werken. Dit zorgt ervoor dat de dat er maar 1 besturingssysteem en een kernel zijn. De verschillende user spaces maken gebruik van CPU, geheugen en netwerk van de host server. Elke user space heeft zijn eigen configuratie omdat de user spaces op zichzelf staan en geisoleerd zijn de andere user spaces. Tegenover hardware virtualisatie zal de besturingssysteem virtualisatie minder gebruik maken van het geheugen en de CPU omdat er maar 1 kernel is en niet meerdere besturingssystemen. Het opstarten neemt maar een fractie van de tijd in beslag tegenover virtuele machines. Deze user spaces worden ook wel containers genoemd.

Hoofdstuk 4

Docker

Containers hebben aan populariteit gewonnen door Docker. Docker heeft gezorgd voor ecosysteem, waarbij je gemakkelijk kan starten met containers te gebruiken. We starten met het maken van een DockerFile waar we de base-image definieren en wat we verder willen aanpassen. Deze DockerFiles kunnen gedeeld worden en zorgen steeds voor dezelfde uitkomst wanneer je van de DockerFile een image zou maken. Deze image kunnen we dan uitvoeren en dan hebben we een werkende container. Dockerhub versnelt dit proces in grote mate door DockerFile open te stellen in repositories. Als je bijvoorbeeld een nginx container nodig hebt, dan zoek je voor nginx en neem je versie die jij nodig hebt. De meest gebruikte repositories hebben documentatie hoe je deze container moet gebruiken en hoe je andere containers met elkaar moet verbinden.

Meest bekende voorbeelden zijn Docker en Rocket.

Er werd al aangehaald bij de virtuele machines dat veiligheid een probleem kan zijn. Dit probleem verdwijnt niet bij containers omdat ze 1 kernel delen. Wanneer de beveiliging van 1 container kan geschonden worden dan wordt het direct gemakkelijker om toegang te verschaffen tot de andere containers. Het gebruiken van 1 OS als de basis voor een aantal containers kan ervoor zorgen dat wanneer dit besturingssysteem niet meer opstaat al deze applicaties niet meer werken.

Hoofdstuk 5

Unikernels

5.1 Inleiding

Als je stilstaat bij hoe complex een computer is dan zul je beseffen dat het een wonder is dat alles eigenlijk werkt. Een applicatie doorloopt verschillende stadia om te beginnen werken. De eerste fase is het compileren of interpreteren van de code dat er geschreven is. Deze zal bepaalde systeem APIs aan roepen van je besturingssysteem. Deze APIs bevinden zich op een besturingssysteem, die zich dan weer bevindt op een virtuele machine. Er zijn nog meer lagen die zich tussen de applicatie en de hardware bevinde. Zoals we al eerder hebben aangehaald is het aantal lagen tussen een applicatie en de hardware 1 van de grootste redenen voor de snelheid van een applicatie.

Wanneer we kijken om iets gemakkelijker te maken of de performantie te verbeteren moeten we kijken naar alle delen. De hardware wordt al meer efficient door gebruik te maken van een hypervisor. Het verschil in de interfaces die de hardware openstellen wordt hierdoor al verminderd. Hypervisors zijn battle-tested. Alle grote cloud providers maken gebruik van hypervisor met succes. Een ander deel dat kan intrigeren is het besturingssysteem. Een besturingssysteem waarop een applicatie bevindt is in de meeste gevallen een Linux besturingssysteem. 1 Van de meest bekende besturingssystemen voor servers is Ubuntu. Het lijkt raar dat een besturingssysteem dat zich wil openstellen aan het grote publiek ook gebruikt wordt voor servers. Sommige delen van een besturingssysteem zijn ook niet meer nodig. We hebben het eerder al gehad over een deel dat misschien niet thuishoort op servers: time-sharing. Time-sharing zorgt ervoor dat de systeem en hardware componenten van de gebruiker worden afgeschermd. Een gebruiker die een programma liet werken zou een andere programma gestart door een andere gebruiker kunnen laten stoppen door een bepaalde instructie uit te voeren. Dit probleem is bijna niet meer voorkomend doordat hardware zou goedkoop is geworden en iedereen zijn eigen computer heeft.

Niet alleen hebben sommige delen quasi geen nut meer maar sommige delen zullen

ervoor zorgen dat de performantie van een server gehinderd wordt voor bepaalde taken. Het is dezelfde analogie als het gebruiken van een hamer voor alle klusjes die je moet doen. Het is niet omdat een hamer goed is voor bepaalde taken dat het goed is voor alles. Dit probleem kunnen we zeker bezien door te bekijken hoeveel besturingssystemen zijn veranderd doorheen de laatste 20 jaar. Er zijn zoveel veranderingen op het vlak van computing dat het raar lijkt dat sommige delen perfect blijven.

Soms kan starten met blanke pagina ervoor zorgen dat er een betere oplossing gevonden wordt. Wanneer we blijven denken met besturingssysteem dan zal de oplossing een variant van een besturingssysteem zijn. Het weggooien van het concept van een besturingssysteem voor servers lijkt radicaal maar kan het veel slechter zijn dan de huidige situatie?

Het programmeren van al de system calls die we nodig hebben om een applicatie te laten werken is veel werken. Maar we moeten niet al de functionaliteit van een besturingssysteem uitwerken. Het laten van werken van een 1 applicatie systeem is veel gemakkelijker te implementeren dan een systeem voor een 1 gebruiker.

Ik heb al aangehaald dat we besturingssystemen gebruikten, als de provisoning unit op onze servers. Nadat ik al de nadelen van de huidige besturingssystemen heb laten zien, kun je wel opmaken dat dit misschien geen goed idee is. 1 Van de voordelen van deze aanpak is dat we besturingssystemen al kennen omdat we er elke dag met werken. Als we een ander concept introduceren dan zou het veel meer moeite zijn om uw infrasstructuur te laten werken. We verloren wel een deel van onze performantie.

De reactie op het verspillen van deze resources was de container. Docker zorgde voor een gemakkelijke gebruikerservaring om containers te gebruiken en dit zorgde voor een revolutie binnen devops. Het nadeel van containers is de veiligheid. Er zijn veel voorbeelden van containers die niet optimaal van elkaar gescheiden zijn.

Daar is waar unikernels te voorschijn komen. Waarom zou je je applicaties niet rechtstreeks op de hypervisor kunnen laten werken? De hypervisor zorgt voor een interface op de hardware en de unikernels zal de nodige functionaliteit implementeren. Dit zorgt ervoor dat elke applicatie afzonderlijk kan worden geoptimaliseerd. De hypervisor kan zelf bepalen hoeveel geheugen of CPU een bepaalde applicatie nodig heeft.

5.2 Microservices

Er is ook een andere evolutie aan de gang maar dan eerder binnen de architectuur van applicaties. De laatste 20 jaar werd er meestal 1 grote applicatie gemaakt die alle functionaliteit op zich nam. Deze applicatie is gemakkelijk te maken omdat men steeds verder bouwt op vorige functionaliteit. Het probleem bij deze manier van werken is de schaalbaarheid van een applicatie. Wanneer een bepaald deel van een applicatie een bottleneck wordt dan bestaat de mogelijkheid om deze te herschrijven en te optima-

liseren. Maar soms moet een deel van een applicatie gewoon veel kunnen verwerken dan een ander deel. Microservices is een concept dat al langer bestaat maar dit is de uitgesproken oplossing voor dit fenomeen. De applicatie opsplitsen in componenten met 1 verantwoordelijke zorgt ervoor dat het veel beter te schalen is. Als je een applicatie hebt met al de functionaliteit in dan zou je een nieuwe virtuele machine moeten opzetten. Dit is niet schaalbaar. Bij microservices moet je gewoon een paar nieuwe instanties van een component starten. Dit vraagt natuurlijk wel om een totaal nieuwe manier van werken. Microservices zijn ook meer op de voorgrond gekomen door het gebruik van containers want zij maken het simpeler om een architectuur op te zetten die gebaseerd is op microservices.

Dit zal ook wel een moeilijkheid meebrengen dat het een puzzel is die in elkaar moet passen. Je moet goed doordacht te werk gaan en je visie naar de toekomst toe ook in het achterhoofd houden. Het geeft het ontwikkelingsteam wel de vrijheid om nieuwe technologieen die uitkomen veel vlugger te gebruiken. Wanneer je iets nieuw wilt gebruiken in een monolithische applicatie dan moet je al een groot stuk herschrijven wat grote kosten met zich meebrengt. Dit zorgt voor wendbaarheid in een omgeving die zeer competitief is.

5.3 Voordelen

Unikernels kan je zien als besturingssysteem dat 1 ding doet maar dit goed doet. Veel van de overbodige functionaliteit wordt weggenomen en alleen de nodige componenten worden samengehouden. Dit resulteerd in een kleinere image.

Er ligt ook een voordeel bij de veiligheid. Wanneer een hacker binnendringt op een virtuele machine die een traditionele stack heeft dan kan hij tools installeren door de package manager te gebruiken en andere commando's gebruiken om meer informantie te krijgen en de server misschien infecteren met een virus. Dit scenario is veel moeilijker bij unikernels omdat veel van commando's weggenomen zijn en er geen package manager aanwezig is. De hacker zou al heel veel werk moeten leveren om de unikernel te infecteren omdat hij zelf de vulnerability zou moeten schrijven. Dan blijft er enkel nog de hypervisor over om te misbruiken. De onveiligheden van hypervisors zijn veel moeilijker uit te buiten omdat ze al een paar decenia in gebruik zijn van veel grote bedrijven. Ze zijn dus al battle-tested.

De image dat gemaakt wordt wanneer de unikernel gecompiled word zal zeer klein zijn. Omdat verschillende drivers zelf geimplementeerd worden. Er moet ook niet in dezelfde mate als het besturingssysteem rekening gehouden worden met andere drivers. De hypervisor zorgt ook voor een stabiele interface en deze zorgt ervoor dat we geen uitgebreide drivers moeten schrijven om te zorgen voor compatibiliteit voor de verschillende hardware apparaten.

1 Van de voordelen van unikernels is uiterst interessant. Het opstarten van een

virtuele machine kan 1 minuut of langer duren. De meeste unikernels starten op in een tiende van een seconde. Dit zorgt ervoor dat reageren op veranderingen qua gebruik veel gemakkelijker wordt. Containers bevinden zich tussen unikernels en virtuele machines op het vlak van opstarttijd. Door deze minieme opstarttijd kunnen we een unikernel opstarten wanneer er een request binnenkomt. En dit kan zorgen voor veel lagere kosten.

Als je al kan opmaken vanuit de tekst is unikernels de opvolging voor containers in bepaalde gevallen. Zoals bij elke nieuwe technologie zijn er voor- en tegenstanders. Alles wat een container doet, doet een unikernel bijna beter.

Veel van de commentaar op unikernels komt van de onmogelijkheid om in productie te bekijken wat er fout aan het gaan is bij een unikernel. Voor de developers is het soms gemakkelijker om een applicatie aan te passen in productie om te zien of een bepaalde bug wordt verholpen. Sowieso is dit geen best practice. Er moet uitgebreid getest worden in development cycle om fouten/bugs tegen te gaan. Wanneer er dan toch iets fout gaat in productie dan zal men de situatie proberen na te bootsen en dan applicaties aan te passen om zo de fout/bug op te lossen.

5.4 Hedendaags gebruik

Hoofdstuk 6

Conclusie

Bibliografie

Lijst van figuren

Lijst van tabellen