Enterprise

Google heeft enkele maanden geleden het AppEngine Cloud platform beschikbaar gemaakt. AppEngine geeft de mogelijkheid om Java-webapplicaties in de 'cloud' van Google te draaien. In dit artikel lees je hoe je dit platform moet plaatsen.

De wolk van Google

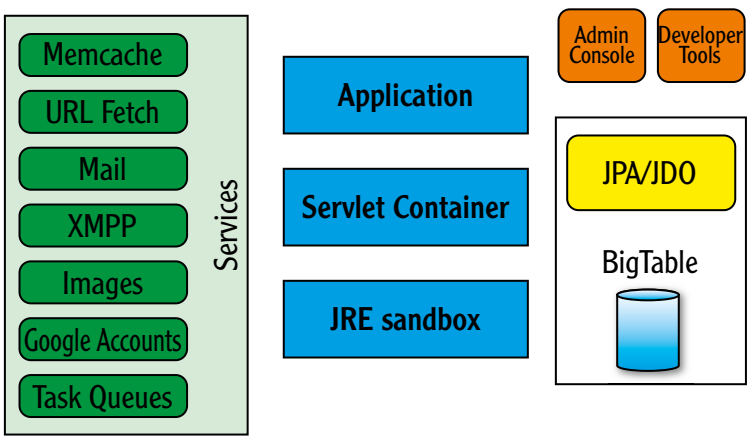
Java cloud computing eenvoudig gemaakt

Cloud computing is hot. Misschien op dit moment nog enigszins overhyped, maar met absoluut een prominente rol in de nabije toekomst. Tot dusver hebben twee typen cloud computing de nadruk gehad; Infrastructure as a Service (IaaS) en Software as a Service (SaaS).
Een mooi voorbeeld van de eerste variant is Amazons Elastic Cloud. Hiermee kun je een complete server op virtuele hardware draaien, waarbij je binnen enkele seconden machines kunt toevoegen. Een mooie vervanging voor het huren van een slecht schaalbare dedicated server, maar de configuratie en het onderhoud van het volledige systeem moet je nog wel zelf doen. Nog steeds een hoop werk dus om een (Java) applicatie online te krijgen en te houden. Software as a Service, om even alle termen een plek te geven, is het betalen voor het gebruik van een applicatie die online in de 'cloud' draait. Zeker een interessant concept, maar niet direct bruikbaar voor een Java-ontwikkelaar, tenzij je natuurlijk zelf een SaaS gaat ontwikkelen en verkopen.

De meest interessante vorm van Cloud Computing voor Java-ontwikkelaars is Platform as a Service. Hierbij is het idee dat een applicatie ontwikkeld wordt met als doel direct gedeployed te worden in de cloud. Direct betekent in dit geval dat je niet meer te maken hebt met een besturingssysteem, maar dat de applicatie naar de aangeboden service wordt geupload.
Het grote voordeel hiervan is dat alle infrastructuur voor je geregeld wordt en je je als ontwikkelaar niet meer druk hoeft te maken over allerlei randzaken zoals serverconfiguratie. Ook is een applicatie hiermee extreem schaalbaar geworden, omdat er geen enkele koppeling met een fysieke machine is. Het grote nadeel is dat je wel beperkt bent in het programmeren; afhankelijk van de mogelijkheden die de service aanbiedt. Bekende voorbeelden van een PaaS zijn Zembly, Force, en Microsofts paradepaartje Azure.

Het aanbod van Google

Sinds enkele maanden biedt Google het AppEngine (GAE) platform voor Java aan. De AppEngine bestaat overigens al meer dan een jaar voor Phyton. De AppEngine valt duidelijk in de laatst besproken categorie; een Platform as a Service. Zoals eerder gezegd is Google niet de eerste partij die een PaaS aanbiedt. Alle hype rondom de AppEngine is echter niet helemaal voor niets; Google heeft toch weer een baanbrekend product neer weten te zetten.



Afbeelding 1: Schematische weergave van de architectuur van de AppEngine.

De reeds bestaande PaaS oplossingen bieden namelijk een complete set van frameworks aan die specifiek zijn voor die service. Dat betekent dus afstappen van de gebruike-

Paul Bakker
is trainer/consultant
bij InfoSupport.

lijke Java-frameworks en leren werken met de veel beperktere frameworks, specifiek voor die sevice. Bovendien is dit natuurlijk een enorme lock-in op de specifieke aanbieder. Het voordeel van werken in de cloud komt dus met een hoge prijs. Google daarentegen biedt Java, een Servlet 2.5 container en een datastore met ondersteuning voor JPA en JDO. Een Servlet container betekent dat vrijwel elk Java web framework potentieel in de AppEngine kan draaien. Bijna volledige vrijheid voor de ontwikkelaar dus. Naast de Servlet container en de datastore heb je de beschikking over tal van andere Google-services om bijvoorbeeld mail te versturen en authenticatie te regelen.

Java aan de ketting

Volledige vrijheid zit er echter toch niet helemaal in; Java is flink aan banden gelegd. Google gebruikt het standaard Java Security model om bepaalde API's uit te schakelen. Het is bijvoorbeeld niet mogelijk om threads te starten, naar het filesysteem te schrijven en de reflection API is alleen op eigen classes te gebruiken. Dit is ook goed te begrijpen, want het is wel zo fijn als een kwaadwillende applicatie niet de volledige cloud om zeep kan helpen. Overigens kun je ook opmerken dat als je nette Servlet code schrijft, je meestal toch deze features al weinig of niet gebruikt. Het hoeft dus zeker geen groot probleem te zijn, maar toch zitten er bij het gebruik van frameworks wat scherpe randjes aan.

In ieder geval is het dus prima mogelijk om plain Servlet en JSP applicaties te ontwikkelen. Ondanks dat niemand meer applicaties lijkt te bouwen zonder minimaal vijf extra frameworks, is dit helemaal niet zo'n slechte optie. Inmiddels is het werken met Servlets en JSP met behulp van de laatste versie van de Expression Language best prettig in gebruik en kom je hier al een heel eind mee.

Het is in veel gevallen toch van grote waarde een framework als JSF, Spring, MVC of Grails te gebruiken.

Database zonder schema

De datastore van AppEngine is geen relationele database, al kun je hem wel als een 'ouderwetse' database gebruiken. De datastore is gericht op schaalbaarheid, en ook extreem grote datasets kunnen efficiënt worden gebruikt. Ook is de datastore gericht op webapplicaties die als eigenschap hebben dat er in verhouding veel lees- en weinig schrijfacties plaatsvinden. Anders dan bij een relationele database is de dataset schemaless. Dat betekent dat twee rijen van hetzelfde type heel andere velden kunnen bevatten. Als dit niet gewenst is moet je dat aan de applicatiekant afdwingen, de datastore doet dat niet. Het gebruik van deze datastore klinkt misschien erg onhandig, omdat het conceptueel anders is dan een relationele database, gelukkig valt dit heel erg mee. Google heeft het mogelijk gemaakt om de datastore te gebruiken op basis van JPA of JDO. Je applicatie gebruikt dus JPA of JDO en merkt niets van het feit dat er geen relationele database wordt gebruikt. Dit is een best-of-both-worlds scenario. Extreme schaalbaarheid door het niet relationele karakter en toch een bekende API. Een nadeel op dit moment is dat een aantal JPA features (nog) niet werkt op de datastore, zoals join queries en polymorphic queries. Wellicht dat deze features nog wel gaan komen in de toekomst, maar een aantal is ook onmogelijk te implementeren op basis van de datastore. Overigens is er ook een low-level API beschikbaar om direct tegen de datastore aan te programmeren.

De datastore is wel transactioneel. Meerde updates kunnen dus binnen een ACID transactie uitgevoerd worden. Als één van de acties faalt, wordt de gehele transactie teruggedraaid. Buiten een transactie heeft het isolatie niveau van de datastore het meest weg van READ_COMMITTED. Binnen een transactie is dat SERIALIZABLE, op basis van snapshot isolation.

Framework as a Service?

Ondanks dat Servlets & JSP best krachtig zijn in hun huidige vorm, is het in veel gevallen toch van grote toegevoegde waarde om een webframework te gebruiken zoals JSF, Spring MVC of Grails. Al deze frameworks zijn in de basis natuurlijk gewoon gebaseerd op de Servlet API en kunnen dus potentieel binnen de AppEngine draaien. De vraag is alleen of een framework niet tegen andere beperkingen van het platform aanloopt. De ervaringen hiermee zijn gemengd.

Om te beginnen kunnen we EJB 3 direct vergeten. Die draaien, hoe lichtgewicht ze inmiddels ook zijn, niet in een Servlet container. Het is wel in te beelden dat Google in de toekomst EJB 3.1 Light gaat ondersteunen, maar daar is nog helemaal niets over bekend. Voorlopig zijn we dus beperkt tot frameworks die draaien in een Servlet container.

Zoals je kunt verwachten kun je Google's eigen GWT gebruiken. JSF 1.2 werkt ook zonder problemen. Zelfs een snapshot release van 2.0 werkt volledig na aanpassing van een enkele initialisatie parameter. In combinatie met JPA heb je hier al een behoorlijk stevige web stack in handen. Tot zover geen problemen in ieder geval. Als we kijken naar frameworks die meer zijn dan alleen een webframework ligt het iets complexer.

Het Spring framework doet het voor het grootste deel prima. Dat betekent het MVC framework, dependency injection en zaken zoals declaratieve transacties.

Wat niet werkt

Spring doet echter ook een aantal dingen die niet werken op AppEngine. Het nieuwe Object/XML mapping framework van Spring 3.0 doet het bijvoorbeeld niet. Dit heeft niet direct te maken met Spring, maar met problemen met de frameworks die de XML mapping voor hun rekening nemen.

Onder andere JAXB2, Castor en XStream willen nog niet in de sandbox van AppEngine werken. Ook bijvoorbeeld de job engine van Spring gaat niet werken in AppEngine. Op zich niet onoverkomelijk, maar het betekent wel dat een applicatie niet altijd eenvoudig te porten is naar GAE. Het is ook lastig dat je niet van tevoren weet of een framework wel of niet gaat werken in GAE en dat dit een kwestie van uitproberen is.

Een aantal frameworks biedt echter ook al expliciete ondersteuning voor GAE aan. Een van de meest interessant frameworks is het op Groovy gebaseerde webframework Grails. GAE ondersteuning is beschikbaar in de vorm van een plugin die ook de deployment server bevat en geautomatiseerde deployment ondersteund. Grails werkt volledig, op een belangrijk uitzondering na. GORM wordt nog niet ondersteund en je moet dus JPA of JDO gebruiken. Dit werkt overigens wel prima. Het Grails team werkt ook al aan integratie van GORM met JPA. De combinatie Grails en GAE is een enorm snel ontwikkeld en deployment platform. Een applicatie bouw je niet meer alleen binnen korte tijd, hij draait ook direct online. Zeker voor de 'low-end' webmarkt die nu beheerst wordt door vooral PHP, bij gebrek aan behoorlijke shared hosting voor Java, is dit een bijzonder interessant alternatief.

Services en beperkingen

De sandbox van de AppEngine zorgt er voor dat sommige standaardfunctionaliteit van webapplicaties toch wat lastiger wordt. Voorbeelden hiervan zijn het versturen van mail en het bewerken van plaatjes. Beide zijn niet met de standaard beschikbare API's mogelijk door de restricties van de sandbox. Binnen AppEngine zijn er echter een aantal services te gebruiken die dit soort functionaliteit via een Google API aanbieden. Op dit moment zijn er onder andere services voor: caching, versturen van mail, bewerken van plaatjes, authenticatie op basis van Google-accounts, een instant messaging API en een task queue. Een openstaand probleem is nog wel het uploaden en opslaan van grote bestanden. Op dit moment kunnen bestanden van maximaal een megabyte opgeslagen worden, wat te weinig is voor veel applicaties. Voor grote bestanden kan er samengewerkt worden met een andere service, zoals S3 van Amazon, maar de samenwerking moet zelf ontwikkeld worden. Google heeft al wel bekend gemaakt dat er gewerkt wordt aan een nieuwe service voor de AppEngine die dit probleem gaat oplossen.

Tools

Google biedt voor verschillende platforms een set van command line tools aan om AppEngine-applicaties te ontwikkelen en te uploaden. Deze tools gebruik je echter meestal niet direct. Google biedt namelijk ook een Eclipse plugin aan waarmee de tooling netjes in de IDE geïntegreerd is. Ook Intellij heeft inmiddels goede AppEngine-support. Het belangrijkste onderdeel van de tooling is de developmentserver. Om applicaties te kunnen ontwikkelen heb je een omgeving nodig die lijkt op de AppEngine omgeving, met de JRE sandbox, de datastore en andere services. De developmentserver zorgt hiervoor. Dit werkt over het algemeen prima, al blijkt in de praktijk dat de omgeving van de developmentserver toch iets anders is dan de AppEngine omgeving zelf. Erg vervelend als je er pas laat achter komt dat dingen die het tijdens development nog deden het niet doen na het uploaden van de applicatie. Stel het testen op de live omgeving dus zeker niet te lang uit en maak dit onderdeel van het ontwikkelproces. Met dit in het achterhoofd is de developmentserver wel prima te gebruiken en voorkomt dat er steeds opnieuw gedeployed moet worden.

Het uploaden van een applicatie is overigens wel bijzonder eenvoudig en erg snel. Vanuit Eclipse en Intellij is dit na een initiële upload niet meer dan een druk op een knop en enkele seconden tot minuten later staat de applicatie online, zonder dat de vorige versie offline is geweest.

Kosten

Google heeft ervoor gekozen het proberen van de AppEngine heel laagdrempelig te maken. Je kunt gratis een account maken en een applicatie uploaden. Dit blijft ook gratis tot aan bepaalde quota. Voor kleine applicaties die niet enorm veel bezoekers ontvangen zullen deze quota vaak al genoeg zijn en is het gebruik van de AppEngine dus volledig gratis. Als de quota wel overschreden worden, betaal je alleen voor wat je daadwerkelijk gebruikt. De kosten hiervan zijn, naar mijn idee, laag. Om te voorkomen dat je voor onverwachte kosten komt te staan moet je overigens zelf aangeven hoeveel je maximaal wilt gaan betalen.
Bereik je dit maximum, dan stopt de applicatie met werken. Het is daarom wel belangrijk om goed in de gaten te houden hoe zwaar een applicatie gebruikt wordt en de quota aan de hand hiervan te configureren. Overigens zijn er ook maximum quota van

Proberen van de AppEngine van Google is bewust laagdrempelig gehouden.

Het is nog niet zover dat alle servers de deur uit kunnen.

wat een applicatie aan resources kan gebruiken om te voorkomen dat een enkele applicatie de complete cloud lam legt. Dit zal voor een normale webapplicatie echter geen enkel probleem zijn, ook niet als deze heel veel requests af moet handelen. Alle quota's zijn uiteraard duidelijk terug te vinden in de AppEngine documentatie.

Resource	Unit	Unit cost
Outgoing Bandwidth	gigabytes	$0.12
Incoming Bandwidth	gigabytes	$0.10
CPU Time	CPU hours	$0.10
Syored Data	gigaytes per month	$0.15
Recipients Emailed	recipients	$0.0001

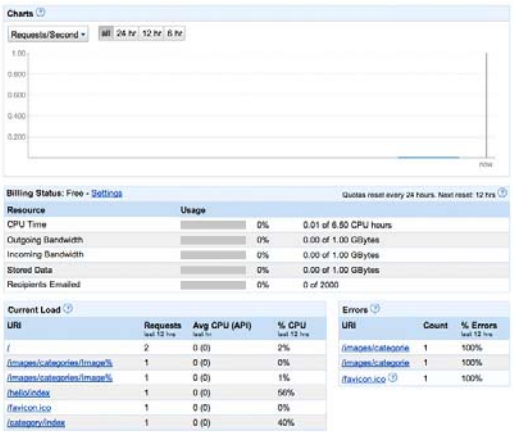
Tabel 1: De kosten op een rijtje.

Resource		
Requests	1,300,00	7,400
Outgoing Bandwidth (billable, includes HTTPS)	1 gigabyte	56 megabytes/ minute
Incoming Bandwidth (billable, includes HTTPS)	1 gigabyte	56 megabytes/ minute
CPU Time (billable)	6.5	15 CPU-

Tabel 2: De gratis quota.

Management console

Natuurlijk is het belangrijk om te kunnen monitoren hoe een applicatie gebruikt wordt, zeker wanneer je hier quota op moet afstellen en voor afgerekend wordt. AppEngine heeft hier een eenvoudig te gebruiken management console voor, die inzicht geeft in het gebruik en je diverse instellingen laat doen op de applicatie. De logfiles van de applicatie zijn hier bijvoorbeeld te vinden en je kunt een eigen domeinnaam aan een applicatie koppelen. Afbeelding 1 laat het dashboard zien, waarop het gebruik van de applicatie is af te lezen.



Afbeelding 2. Het dashboard van de management console.

Query taal

Aangezien de eerder besproken datastore geen relationele database is, kun je ook niet standaard SQL tools gebruiken. De management console van AppEngine heeft wel de mogelijkheid om door de data te browsen en hier zelfs GQL (een query taal die lijkt op SQL) op uit te voeren.

Een erg interessante feature van de GAE is dat je meerdere versies van een applicatie online kunt hebben en hier simpelweg tussen kunt switchen. Blijken er problemen te zitten in een nieuwe versie, dan kan er met een enkele klik worden teruggeschakeld naar een vorige versie. De verschillende versies zijn ook afzonderlijk bereikbaar op aparte URLs. Dit maakt het mogelijk om een nieuwe versie online te testen voordat andere gebruikers dit zien.

Het einde van de fysieke server?

Kunnen we al concluderen dat alle servers de deur uit kunnen en dat we voortaan Java-webapplicaties op AppEngine draaien?

Zover is het nog even niet, maar AppEngine geeft hier al wel een goede basis voor. Om te beginnen is AppEngine toegespitst op echte webapplicaties en is het niet bedoeld voor bijvoorbeeld batch verwerkingen. Daarnaast kan de sandbox waarin applicaties draaien een probleem zijn, ondanks de grote vrijheid die AppEngine aanbiedt. In dat geval is het waarschijnlijk interessanter om naar bijvoorbeeld de Elastic Cloud van Amazon te kijken.

Voor veel applicaties zijn de mogelijkheden van AppEngine echter ruim voldoende. In dat geval is AppEngine een goedkope, extreem schaalbare en bijzonder eenvoudig te gebruiken platform. Google werkt ook hard om de mogelijkheden van AppEngine verder uit te breiden. Het is onmogelijk om te voorspellen hoe groot de rol van AppEngine en vergelijkbare concurrenten in de toekomst gaat zijn. Google biedt in ieder geval wel een platform dat, zeker in de 'low-end' web markt, erg interessant is. «

Links

AppEngine website:
http://code.google.com/appengine/
Grails AppEngine plugin:
http://grails.org/plugin/app-engine
Grails AppEngine screencast:
http://grails.org/dist/screencasts/grails-appengine-screencast.mov