Geautomatiseerd testen met Windows Azure

Wereldwijd loadtests uitvoeren met de Management API

Valéry Jacobs

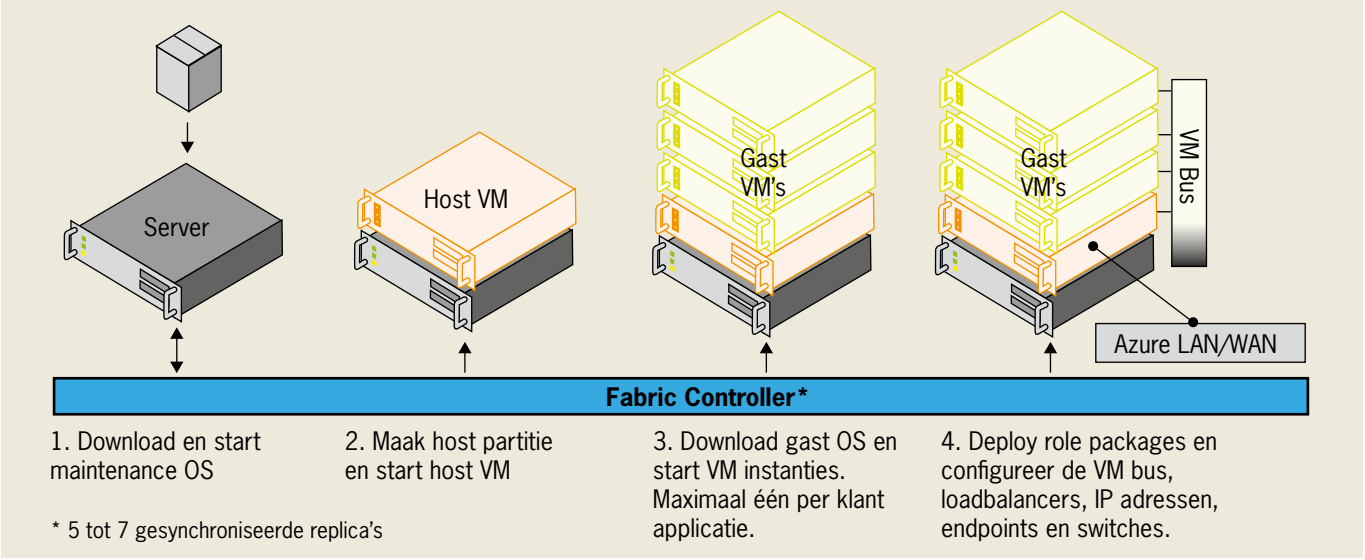
Een belangrijk kenmerk van cloud computing is het gemak waarmee je rekenkracht, opslagcapaciteit en bandbreedte beschikbaar kunt maken. In Windows Azure komen deze resources uit de Global Foundation Services. Dit is een wereldwijd netwerk van meer dan honderd grote en kleinere datacenters waar Microsoft aan heeft gebouwd, ter ondersteuning van haar Software Plus Services initiatief.

Windows Azure is een belangrijk onderdeel van deze strategie. In de datacenters draaien virtuele machines (VM) op de Windows Azure Hypervisor, een geoptimaliseerde versie van Hyper-V. De capaciteit van de hardware wordt verdeeld over de virtuele machines conform het servicemodel dat de ontwikkelaar of applicatiebeheerder heeft geconfigureerd.

In dit artikel wordt bekeken hoe services geautomatiseerd in de cloud uitgerold kunnen worden en welke impact de verschillende configuratie- en deploymentopties hebben op de performance van een loadtest-applicatie. Aangezien services wereldwijd gehost kunnen worden wordt ook bekeken of er verschillen zijn tussen de beschikbare locaties.
De basis voor het Windows Azure platform is de Windows Azure Fabric Controller die verantwoordelijk is voor het activeren en configureren van fysieke hardware en het opstarten van de virtuele machines die de webroles, workerroles en VMroles hosten (zie Afbeelding 1). De fabric controller regelt dat een host, vanwege de kans op defecte hardware, alleen virtuele machines opstart voor verschillende applicaties. Dit zijn zeer waarschijnlijk ook applicaties van verschillende klanten. Role-instanties communiceren altijd via hun host met het netwerk en zij worden daarbij voor de beveiliging van elkaar geïsoleerd.

Benchmarks

Er zijn op het internet uitstekende benchmarks beschikbaar die inzicht geven in de performance van de verschillende Windows Azure features. Een van de meest uitgebreide is te vinden op http://azurescope.cloudapp.net. Op deze site zijn een groot aantal testresultaten gepubliceerd die ontwikkelaars en architecten inzicht geven in de gevolgen van bepaalde keuzes. Zo blijkt onder



AFBEELDING 1: DE STAPPEN DIE DE FABRIC CONTROLLER DOORLOOPT BIJ HET OPSTARTEN VAN FYSIEKE EN VIRTUELE MACHINES.

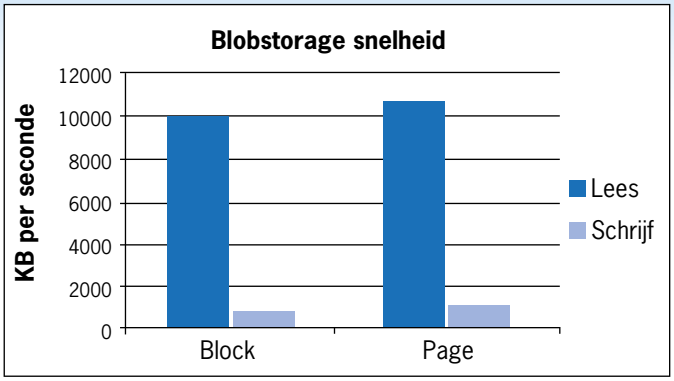


DIAGRAM 1: LEZEN EN SCHRIJVEN MET BLOCK BLOB EN PAGE BLOBS.

HTTP method	POST
Request URI	https://management.core.windows.net/<subscription-id>/services/hostedservices
Request body	<CreateHostedService xmlns="http://schemas.microsoft.com/windowsazure">
	<ServiceName>service-name</ServiceName>
	<Label>base64-encoded-service-label</Label>
	<Description>description</Description>
	<Location>location</Location>
	<AffinityGroup>affinity-group</AffinityGroup>
	</CreateHostedService>

TABEL 1: DE SPECIFICATIES VAN DE MANAGEMENT API EN DE BESCHRIJVING VAN DE REQUEST BODY.

andere uit deze tests dat pageblobs sneller zijn dan blockblobs. (zie diagram 1). Windows Azure Drive kan snelheden halen die vergelijkbaar zijn met die van een lokale schijf. Ook blijkt dat SQL Azure sneller gegevens wegschrijft dan tablestorage (zie diagram 2), maar dat dit wat betreft het lezen van data juist andersom is. De tests die genoemd worden op deze website zijn uitgevoerd vanuit één locatie. Het doel van de testapplicatie die in dit artikel wordt beschreven is echter om de mogelijke verschillen tussen datacenters en hun onderlinge bandbreedte te achterhalen.

De Management API

De performancetests worden in de testapplicatie geautomatiseerd door gebruik te maken van de Windows Azure Management API, die bereikbaar is onder https://management.core.windows.net. Deze interface wordt in de testapplicatie aangeroepen met de ServiceManagement assembly die Microsoft in de SDK samples heeft gepubliceerd. De API wordt ook gebruikt door de Management Portal, de management cmdLets en de vele open-source en commerciële management tools die voor Windows Azure beschikbaar zijn. Het geautomatiseerd aanmaken van een nieuwe Hosted Service registratie is nog niet ingebouwd in de bekende open source tools, hoewel de API deze commando's wel beschikbaar stelt. Aangezien de beoogde testscenario's over meerdere locaties worden uitgevoerd en de locatie, ofwel de region instelling, op het niveau van de Hosted Service moet worden ingesteld, moet de ServiceManagement assembly worden uitgebreid. De online documentatie beschrijft hoe de HTTP-requests naar de API er uit moeten zien (zie tabel 1).

De IServiceManagement interface moet worden uitgebreid met de asynchrone calls voor CreateHostedService en DeleteHostedService (zie codevoorbeeld 1). Met attributen wordt aangegeven dat WCF een asynchrone call (HTTP post) moet doen. Daarnaast wordt er een UriTemplate gespecificeerd die correspondeert met de request-URI. Elk met accolades omgeven segment in de UriTemplate komt overeen met de parameterwaarde die aan de methode wordt meegegeven. Het datacontract voor CreateHostedServiceInput moet overeenkomen met de requestbody (zie codevoorbeeld 2).

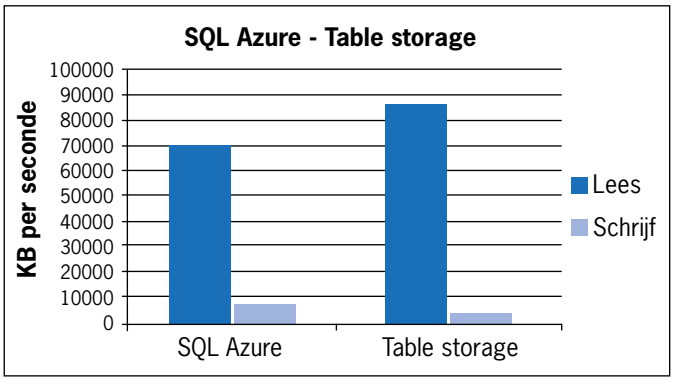


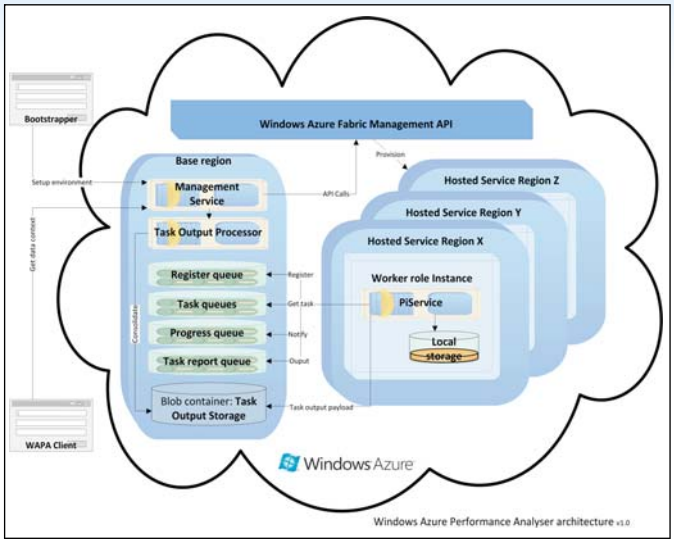
DIAGRAM 2: SQL AZURE EN TABLE STORAGE VERGELEKEN.

```
    public partial interface IServiceManagement
    {
       [OperationContract(AsyncPattern = true)]

[WebInvoke(Method = "DELETE", UriTemplate = @"{subscriptionId}/
services/hostedservices/{serviceName}")]

IAsyncResult BeginDeleteHostedService(string subscriptionId,
string serviceName, AsyncCallback callback, object state);
      void EndDeleteHostedService(IAsyncResult asyncResult);

      [OperationContract(AsyncPattern = true)]
      [WebInvoke(Method = "POST",UriTemplate = @"{subscriptionId}/
      services/hostedservices")]

IAsyncResult BeginCreateHostedService(string subscriptionId,
CreateHostedServiceInput input, AsyncCallback callback, object
state);
      void EndCreateHostedService(IAsyncResult asyncResult);

  ...
  }
```

CODEVOORBEELD 1: DE WCF ATTRIBUTEN VOOR HET AANROEPEN VAN DE REST API.

```
    [DataContract(Name = "CreateHostedService", Namespace = Con-
stants.
ServiceManagementNS)]
    public class CreateHostedServiceInput : IExtensibleDataObject
    {
        [DataMember(Order = 1, EmitDefaultValue = false)]
        public string ServiceName { get; set; }

        [DataMember(Order = 2)]
        public string Label { get; set; }

        [DataMember(Order = 3)]
        public string Description { get; set; }

        [DataMember(Order = 4, EmitDefaultValue = false)]
        public string Location { get; set; }

        [DataMember(Order = 5, EmitDefaultValue = false)]
        public string AffinityGroup { get; set; }

        public ExtensionDataObject ExtensionData { get; set; }
    }
```

CODEVOORBEELD 2. HET DATACONTRACT DAT OVEREENKOMT MET DE SPECIFICATIES VOOR DE REQUESTBODY.



AFBEELDING 2: DE ARCHITECTUUR VAN DE TESTAPPLICATIE.

Sommige requestbody-elementen, in dit voorbeeld het Label veld, moeten worden omgezet in Base64 encoded formaat. In de responseheader van de HTTP-requests zit een request-Id waarmee de voortgang van de asynchrone operaties kunnen worden gevolgd. De operatie https://management.core.windows.net/<subscription-id>/operations/<request-id> retourneert de status InProgress, Succeeded of Failed.

De testapplicatie

Er zijn wetenschappelijke methoden om systemen te testen, maar het doel van deze testapplicatie is een eenvoudig vergelijk te maken tussen verschillende serviceconfiguraties en hostinglocaties. De testactiviteiten worden uitgevoerd op verschillende tijdstippen om te kunnen meten of piek- en daluren en het zogenaamde 'internet weer' van invloed zijn op de performance. Kortom, er wordt getoetst in hoeverre het de Windows Azure Fabric Controller lukt om consistente roles met een voorspelbare performance beschikbaar te stellen en er worden indicatieve metingen gedaan van de netwerkperformance tussen datacenters, binnen en buiten hun region.

Onderdeel van de activiteiten is een loadtest die wordt uitgevoerd met een executable geschreven in C, die complexe wiskundige berekeningen uitvoert. De executable berekent de wiskundige constante π zeer nauwkeurig waarbij de CPU maximaal belast wordt en een grote hoeveelheid geheugen en schijfruimte in beslag wordt genomen. Deze applicatie is door de wiskundige Fabrice Bellard ontwikkeld. Hij heeft er in 2009 het voormalig wereldrecord van 2,6 miljard getallen achter de komma mee weten te realiseren op een desktopcomputer. Het record stond daarvoor op naam van een peperduur rekencentrum.

Het uitvoeren van executables in web of workerroles gebeurt op dezelfde manier als bij een lokale machine. Er moet echter wel een localstorage resource in de servicedefinitie worden opgenomen waar de output naar toe kan worden geschreven. De output wordt vervolgens met de Windows Azure Storage service weggeschreven in een blob en de workerrole maakt tijdens deze activiteiten zijn status kenbaar via de Windows Azure Queue-service. Er wordt tot slot een rapportagebericht naar de reportqueue gestuurd. Dit bericht geeft door hoe lang de taak heeft geduurd, in welke blob de output is weggeschreven en hoe lang het uploaden duurde. Afbeelding 2 is een weergave van de architectuur van de applicatie.

De uitvoering

Er zijn een paar stappen die handmatig moeten gebeuren voordat de testapplicatie kan draaien. Allereerst moet er een Windows Azure account beschikbaar zijn waar een managementcertificaat aan is toegevoegd en waarin een of meerdere storageaccounts zijn aangemaakt. De gegevens van deze accounts zijn nodig voor de bootstrapper die de testomgeving installeert en de managementservice operationeel maakt. Afbeelding 3 toont de interface van de testapplicatie. Een uitgebreide beschrijving van deze stappen is op de projectsite te vinden. Door uit een lijst van regions te kiezen kunnen hosted services worden gegenereerd. Vervolgens kunnen packages worden gedeployed en kan de deploymentslot (staging of production) worden opgegeven, samen met het aantal instanties en de VMSize. De deployments komen na verloop van tijd in een lijst terecht en kunnen dan calculatietaken toegewezen krijgen die in corresponderende queues worden gezet.

De broncode van deze applicatie is gepubliceerd op cloudcompanion.codeplex.com.

Testresultaten

Hier volgt een kort overzicht van een paar testresultaten. Op de projectsite zullen meer resultaten worden gepubliceerd en met de testapplicatie kun je zelf eigen scenario's uitvoeren met verschillende parameters.

Calculatie test

De snelheid waarmee de calculaties (zie diagram 3) werden uitgevoerd is bij de meeste regions redelijk constant en vergelijkbaar. De enige afwijking zien we bij de South Central US instantie die meer tijd nodig had voor de calculatie van π met tien miljoen getallen achter de komma.

Blob upload

Deze test heeft gedurende een etmaal gedraaid, voor het diagram is een fragment hieruit genomen. In Windows Azure wordt de bandbreedte, die via de host VM loopt, gedeeld. Dit betekent dat instanties op een fysieke node een groter deel van de bandbreedte krijgen naarmate de andere instanties minder capaciteit vragen. Dit kan een verklaring zijn voor de grote verschillen per upload. Duidelijk is te zien dat de instantie in de West Europe region meer bandbreedte tot zijn beschikking had. Reden hiervoor is dat storage account ook in deze region is ondergebracht. Als de test

AFBEELDING 3: DE INTERFACE VAN TESTAPPLICATIE.

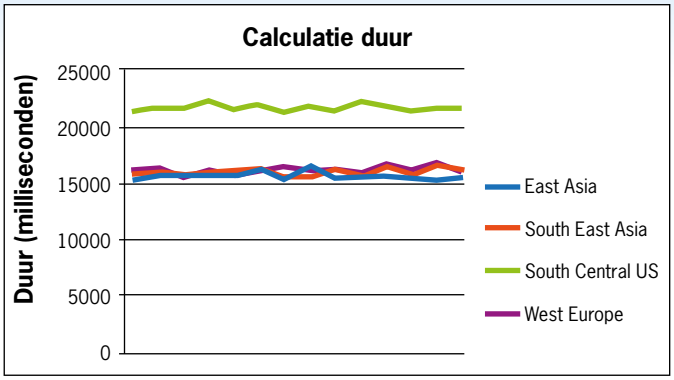


DIAGRAM 3: DE DUUR VAN DE CALCULATIES BIJ EEN LOADTEST UITGEVOERD IN VIER REGIONS, VERSPREID OVER DE CONTINENTEN. DE VMSIZE VOOR DE INSTANTIES WAS SMALL.

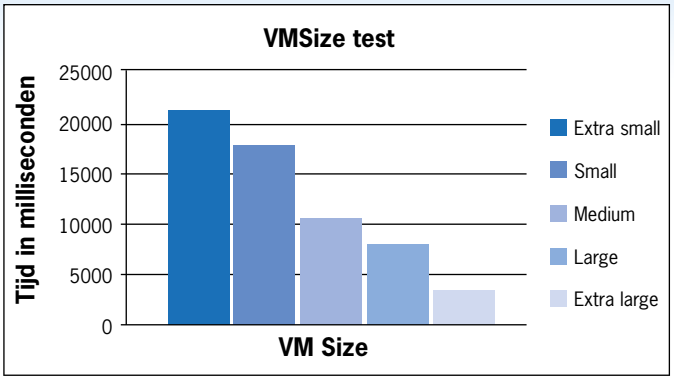


DIAGRAM 5: VMSIZE TEST. EEN LOADTEST IN MEERDERE HOSTED SERVICES MET VERSCHILLENDE VMSIZE OPTIES.

op grotere VMSizes wordt gedraaid zal de bandbreedte ook toenemen.
Het aanpassen van de VMSize heeft het te verwachten effect (zie diagram 5) hoewel de verschillen niet evenredig zijn met het aantal cores. Het resultaat is afhankelijk van de sourcecode en hoe deze gebruik maakt van de extra cores. De ExtraSmall VMSize deelt zijn CPU met andere instanties en heeft minder bandbreedte tot zijn beschikking.

Alternatieve testtools

Er zijn sinds de komst van Windows Azure al veel tools beschikbaar gekomen om cloudapplicaties te monitoren en performancegegevens te verzamelen. Deze tools maken gebruik van de diagnose voorzieningen van het Windows Azure platform, zoals tracing en diagnostic infrastructure logs, en meetinstrumenten van Windows Server 2008 (R2) als eventlogs, performance counters, etcetera. Een minder bekende tool is de Windows Azure Throughput analyser die beschikbaar is gesteld door Microsoft Research. Met deze tool kunnen metingen worden gedaan van de overdrachtsnelheid tussen on-premise en de cloud voor blobstorage, tablestorage en queuestorage data. Een andere handige testtool is ApacheHammer waarmee je vanuit Windows Azure loadtests op een URL kunt uitvoeren.

Conclusie

De Windows Azure Management API maakt het erg eenvoudig services volledig automatisch te installeren en aan te sturen. Dit kan bijvoorbeeld nuttig zijn als je geautomatiseerd testen in de cloud overweegt of om Windows Azure aan te sluiten op bestaande managementtools. Automatische deployment kan, naast autoscaling, operationele kosten nog verder reduceren door alleen tijdens kantooruren of op verzoek services beschikbaar te stellen. Een service in de cloud kan geoptimaliseerd worden door bijvoorbeeld de tijdzones te volgen en de internationale piekuren vanuit een lokaal datacenter te opvangen, conform het follow-the-sun principe.

Met Windows Azure profiteer je van de voordelen van Platform-as-a-Service. Niet alleen omdat het beheer automatisch wordt uitgevoerd, maar ook omdat applicaties voorzieningen tot hun beschikking hebben die bijdragen aan een optimale performance en beschikbaarheid. De testapplicatie zal via de codeplex projectsite (zie referenties) worden uitgebreid met scenario's waarbij ook andere Windows Azure features worden getest zoals Windows Azure Connect, de Appfabric Service Bus en Appfabric Caching.

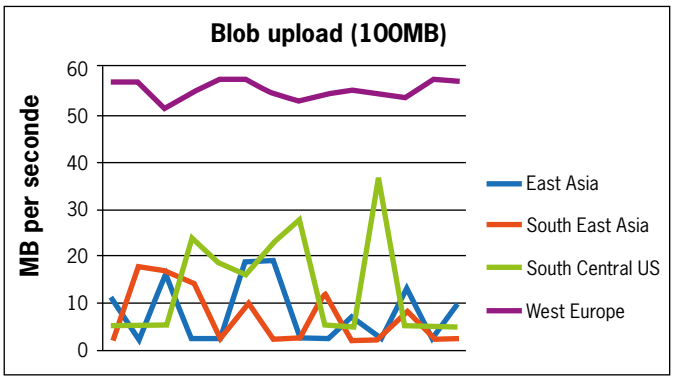


DIAGRAM 4: DE BLOB UPLOAD DUUR BIJ EEN LOADTEST UITGEVOERD IN VIER REGIONS, VERSPREID OVER DE CONTINENTEN. VMSIZE VOOR DE INSTANTIES WAS SMALL.

Links

- cloudcompanion.codeplex.com – projectsite met de sourcecode van de testapplicatie.
- azurescope.cloudapp.net – The Azurescope website van de eXtreme Computing Group.
- http://bit.ly/allKot - MSDN site voor de management API documentatie.
- channel9.msdn.com/Shows/Cloud+Cover – de Cloud Cover show op Channel 9.
- www.wadewegner.com - Het blog van Wade Wegner, co-host van de Cloud Cover show.
- bellard.org/ en en.wikipedia.org/wiki/Fabrice_Bellard- De website en wikipagina van Fabrice Bellard, de auteur van de π calculator executable.
- http://bit.ly/aTywfz - Download site voor de Windows Azure Throughput Analyser.

Valéry Jacobs, is als Technisch Consultant werkzaam bij Valid en gespecialiseerd in cloud computing en met name Windows Azure. Hij is te bereiken via vjacobs@valid.nl en te volgen op twitter via @valeryjacobs.