Technische Documentatie Project NetCentric

Gilles Lesire

Het cloud computing project bestaat uit 4 Java projecten, de loadbalancer, de scaler, de workers en de client.

# LoadBalancer project

## Org.webapp.controllers

### BalanceController.java

Deze klasse dient voor het versturen van een taken naar workers

#### sendJarFileToWorker(org.webapp.model.JarFile jarFile, String workerDNS)

Versturen van een taak JarFile naar een worker met DNS adres workerDNS

### QueueManager.java

Deze klasse dient voor een singleton te maken van de klasse ThreadQueue.java

#### public static ThreadQueue getInstance()

Geeft de singelton instance van ThreadQueue terug

### ResultManager.java

Deze klasse houdt een singelton bij van de hashmap van resultaten. Met een key en een resultaat. Ook genereert deze klasse de key voor het volgende resultaat.

#### public static Integer getResultCount()

Genereert een key voor een resultaat.

#### public static ResultList getInstance()

Geeft de instance van ResultList terug.

### WorkerManager.java

Deze klasse houdt de singleton instance bij van WorkerQueue.

#### public static WorkerQueue getInstance()

Geeft de instance van WorkerQueue terug.

## Org.webapp.model

### JarFile.java

Deze klasse is een dataklasse en houdt de data bij van een taak. In deze klasse zit een link naar de jarfile, een geserializeerde string van het callable object en een resultkey die gebruikt wordt om het resultaat op te vragen.

### ResultList.java

Deze klasse bevat een hashmap met alle afgeleverde resultaten. De resultaten worden opgezocht via hun resultkey.

### ThreadQueue.java

Deze klasse bevat een queue met alle uit te voeren taken op de load balancer. De taken zijn JarFile objecten.

### WorkerQueue.java

Deze klasse bevat een queue met alle DNS addressen van de workers die zich hebben ingeschreven voor het ontvangen van taken.

## Org.webapp.services

### LoadBalancer.java

Deze klasse is de webservice van de load balancer. Via deze webservice kunnen de clients en de workers communiceren met de load balancer voor het doorgeven van taken of aanvragen van taken, het opvragen van resultaten of het plaatsten er van. En ook opvragen hoeveel taken en workers er in de queues staan.

#### public String getResult(Integer key)

Voor het opvragen van een resultaat, deze geeft false terug als het resultaat nog niet ter beschikking is.

#### public void putResult(Integer key, String result)

Voor het plaatsen van een resultaat. Het resultaat wordt opgeslagen onder de meegegeven key waaronder deze later kan opgevraagd worden door een client.

#### public Integer putRunnable(String urlString, String objectString)

Als er reeds een worker in de WorkerQueue staat dan zal hij deze taak daar onmiddellijk naartoe sturen in het ander geval zal hij deze taak toevoegen aan de queue. De functie geeft een integer terug. Deze integer is de key waarmee de client zijn resultaat kan opvragen.

#### public void rePutJarFile(String urlString, String objectString, Integer resultKey)

Deze functie plaatst taken terug vanvoor in de queue. Dit zijn taken die al eens opgevraagd zijn geweest maar door een worker die wordt afgesloten teruggeplaatst worden. Deze komen dus vooraan in de queue omdat ze normaal al aan het draaien moeten zijn.

#### public void subscribeWorker(String worker)

Als er reeds taken in de ThreadQueue staan stuurt hij meteen een taak terug naar de worker. Anders voegt hij deze worker toe aan de WorkerQueue.

#### public void unsubscribeWorker(String worker)

Deze functie gaat een worker uit de WorkerQueue halen als hij daar in staat. Dit wordt gebruikt als een worker geterminate wordt om te vermijden dat de load balancer later een taak gaat proberen te sturen naar een instance die niet meer runt.

#### public int queueSize()

Geeft de lengte van de ThreadQueue terug. (Het aantal taken in de queue).

### WorkerCallbackHandler.java en WorkerStub.java

Dit zijn gegenereerde webservice klassen voor het communiceren met de workers.

# Scaler

## Org.webapp.controllers

### AwsConsoleApp.java

Gedownloade klasse van AWS om te kunnen communiceren met AWS instances.

### InstanceManager.java

Deze klasse dient voor het opstarten en terminaten van EC2 instances. Ook kunnen er keychains en securitygroups mee aangemaakt worden.

#### public void createSecurityGroup()

Aanmaken van een securitygroep met al de rechten die nodig zijn om met de instances te kunnen werken.

#### public String createKeyPair()

Aanmaken van een keypair voor authenticatie.

#### public Instance createInstance(String imageId, String instanceName)

Opstarten van een EC2 instance met de naam instanceName en gebaseerd op het image met id imageId.

#### public void terminateInstance(String instanceId)

Beëindigen van de instance met id instanceId.

## RunScaler.java

Deze klasse is de main klasse. Deze bevat een eeuwige lus en zal dynamisch machines bij opstarten of afsluiten indien nodig. Ook zal het de load balancer opstarten in het begin. Verder zal het de GUI updaten met up to date informatie.

## Scaler.java

Deze klasse dient voor het managen van de instances. Voor het aanmaken en terminaten van de workers en load balancer. Deze is gescheiden van de RunScaler klasse voor redundantie te vermijden en beter overzicht te behouden. Het beheren van de load balancer en een worker instance is namelijk redelijk lineair.

#### public void setFrame(GUI frame)

GUI Frame meegeven voor het kunnen schrijven naar de console.

#### public void createSecurityGroup()

Aanmaken van de securitygroep

#### public void createKeyPair()

Aanmaken van de key pair

#### public void createLoadBalancer()

Aanmaken van de load balancer

#### public Instance createWorker()

Opstarten van een worker

#### public double getInstanceWorkLoad(Instance instance)

De workload van een bepaalde instance opvragen.

#### public int getQueueLength()

De lengte van de queue in de load balancer opvragen.

#### public void terminateWorker(Instance worker)

Deze funtie dient voor het afsluiten van een worker.

## Org.webapp.gui

### GUI.java

Deze klasse dient voor het generen van de grafische user interface.

## Org.webapp.model

### InstanceList.java

Deze klasse houdt een lijst bij van alle draaiende instances en dewelke aan het opstarten zijn.

## Org.webapp.services

### LoadBalancerCallbackHandler.java en LoadBalancerStub.java

Dit zijn gegenereerde webservice klassen voor het communiceren met de load balancer.

### WorkerCallbackHandler.java en WorkerStub.java

Dit zijn gegenereerde webservice klassen voor het communiceren met de workers.

# Worker

## Org.webapp.controllers

### LoadManager.java

De loadmanager draait in een aparte thread en gaat elke 5 seconden de load van de worker berekenen en op deze manier een gemiddelde van de laatste minuut berekenen.

#### public double getAverageUsage()

Met deze functie kan je de gemiddelde work load van de laatste minuut opvragen

### RunThread.java

Deze klasse zal een bepaalde taak draaien in een aparte thread. De taak wordt meegegeven via een JarFile object.

### SerializeController.java

### Deze klasse wordt gebruikt voor het deserializeren van Callable onjecten via de URLClassLoader en het geserializeerde object.

## Org.webapp.listeners

### StartListener.java

Deze listener start op bij het opstarten van de server waarbij deze dan de LoadThread zal opstarten.

## Org.webapp.model

### AddressData.java

Deze klasse houdt in een singelton data bij over de adres van de worker zelf en het adres van de load balancer.

### ClassLoaderObjectInputStream.java

New ObjectInputStream-subclass that has access to my custom ClassLoader

### JarFile.java

Data object voor het bijhouden van informative in verband met een bepaalde taak. Bevat data van de geserializeerde string, de url naar de jarfiles en de resultkey.

### LoadSingleton.java

Beheert de instance van LoadManager. Je moet altijd met dezelfde instance van loadmanager werken.

### Subscribing.java

Houdt de huidge subscriber staat bij van de worker. Deze dient om te onthouden of de worker reeds is ingeschreven of niet voor een nieuwe taak om te vermijden dat hij zich meermaals tegelijk gaat inschrijven bij de load balancer.

### ThreadList.java

Houdt een lijst bij van alle momenteel draaiende taken.

### ThreadSingleton.java

Beheert de enige instance van ThreadList.

## Org.webapp.services

### Worker.java

Dit is de klasse die services bevat om te communiceren met de worker.

#### public double getWorkLoad()

Geeft de huidige werklast voor de werker terug.

#### public boolean ditchThreads()

Deze zal er voor zorgen dat de werker zijn draaiende taken stopzet, terugstuurt naar de load balancer en zijn threadlist leegmaakt. Deze functie wordt uitgevoerd vlak voordat de worker wordt afgezet.

#### public String executeJar(String urlString, String objectString, Integer resultKey)

Voer een taak uit met de volgende informative, url van de jarfile en de geserializeerde objectstring. Alsook de key die gebruikt moet worden voor het resultaat terug te geven aan de load balancer.

### LoadBalancerCallbackHandler.java en LoadBalancerStub.java

Dit zijn gegenereerde webservice klassen voor het communiceren met de load balancer.

# Client

## Org.client

### Client.java

Dit is de main klasse van het project. Deze zal de taken naar de load balancer sturen gebaseerd op het DNS adres van de load balancer.

### LoadController.java

Deze klasse dient voor het berekenen van de work load op de lokale computer. Deze wordt momenteel niet meer gebruikt omdat alle taken vanaf nu in de cloud gedraaid worden en niets meer lokaal.

### SendRunnable.java

Aparte threadklasse voor het versturen van callables naar de load balancer.

### SerializeController.java

Deze klasse wordt gebruikt voor het serializeren van SimpleRunnable objecten voor te versturen naar de load balancer.

### SimpleRunnable.java

Dit is een callable object dat bedoelt is om een taak te bevatten dat naar de cloud gestuurd kan worden. Deze kan draaien in een aparte thread maar ik heb toch voor het callable concept gekozen omdat je dan het resultaat kan opvragen. Callables draaien net zoals runnables in een eigen thread.

## Org.webapp.services

### LoadBalancerCallbackHandler.java en LoadBalancerStub.java

Dit zijn gegenereerde webservice klassen voor het communiceren met de load balancer.