Rapport Leertaak 2 - BSS

United Nations Weather Data Management Institute   
Weergegevens Applicatie

**INV2B – Groep 5**  
Mark Nijboer, Kevin Haitsma   
en Rick van der Poel

Rapport Leertaak 2

**Datum:**   
09-10-2015

**Auteurs:**   
Mark Nijboer, Kevin Haitsma en Rick van der Poel

**Opdrachtgever:**   
UNWDMI

**Instituut:**  
Hanzehogeschool Groningen

#### Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Datum** | **Auteur** | **Beschrijving** |
| 0.1 | 23-09-2015 | Rick van der Poel | Opmaak verslag |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Samenvatting

In dit rapport is informatie te vinden over een aangepaste versie van een demoapplicatie die in de voorgaande leertaak is ontwikkeld. In deze aangepaste variant is de applicatie niet maar afhankelijk van een relationele database. Verder is het de bedoeling om rekening te houden met concurrency. Dit betekent dat de vele bestanden die binnen komen interactie met elkaar kunnen hebben. Uiteraard is het de bedoeling dit te voorkomen.

De demoapplicatie zal een groot aantal gegevens die binnenkomen in clusters moeten kunnen verwerken en daarna opslaan. De gegevensstroom hiervoor is enorm en het doel is dan ook te kijken waar de bottleneck zit bij het verwerken van de gegevens. Daarnaast zal er ook een correctie gedaan moeten worden op gegevens die niet volledig of juist zijn binnengekomen.

De gegevens worden aangevoerd in XML een opmaaktaal die het als “platte tekst” aanlevert. Deze data zal vervolgens bruikbaar gemaakt worden, gecorrigeerd worden en vervolgens klaargemaakt worden voor de database. Om dit allemaal te kunnen doen wordt de applicatie in Java geschreven en wordt het programma in meerdere klasse verdeeld volgens de OO-methode.

Elke klasse heeft een eigen taak en samen functioneren deze klassen als applicatie. Hiervoor zijn de volgende klassen gemaakt:

- een klasse voor het initialiseren van de applicatie  
- een klasse voor het opzetten van de connectie met de cliënt  
- een klasse voor het verwerken van de XML-data  
- een klasse voor het corrigeren van de data  
- een klasse voor het verzamelen van de gegevens voor de database en het bufferen van de database inserts  
- en een klasse voor het invoeren van de gegevens in de database

Vervolgens is de applicatie getest om te kijken hoe het reageert op de grote stroom van gegevens.

Inhoudsopgave

[Samenvatting 3](#_Toc430791301)

[1. Inleiding 6](#_Toc430791302)

[2. Probleemstelling 7](#_Toc430791303)

[3. Applicatie onderdelen 8](#_Toc430791304)

[4. Stresstest 9](#_Toc430791305)

[4.1 Specificaties testsystemen 9](#_Toc430791306)

[4.2 Verwerkingssnelheden 9](#_Toc430791307)

[4.3 Bottleneck 9](#_Toc430791308)

[5. Conclusie 10](#_Toc430791309)

Verklarende woordenlijst

**Cliënt** – De cliënt is het programma van de UNWDMI dat de weergegevens doorstuurt naar de applicatie.

**Server** – De server is de applicatie waarmee de gegevens worden ontvangen en worden verwerkt.

**XML** – Extensible Markup Language is de opmaaktaal waarin de cliënt de weergegevens aanlevert in de vorm van platte tekst.

**UNWDMI** – Staat voor United Nations Weather Data Management Institute.

**Bottleneck** – Is de beperking in de applicatie waardoor het niet optimaal kan functioneren.

1. Inleiding

In dit rapport leest u over de aanpassingen die zijn gedaan op de eerste proef- demonstratie applicatie dat gemaakt is tijdens leertaak 1. Dit rapport moet inzicht geven over de gemaakte aanpassingen en prestaties van de verbeterde demonstratie applicatie.

In dit rapport bespreken we aanpassingen als volgt:

Als eerst wordt er in hoofdstuk 2 gekeken wat precies de probleemstelling is en welke veranderingen er gemaakt moeten worden.

In hoofdstuk 3 komen de verschillende programma onderdelen aan bod die aangepast zijn.

Ook deze versie van de applicatie wordt natuurlijk onderworpen aan een stress test. Hier vind u meer informatie over in hoofdstuk 4.

Als laatste wordt er conclusie getrokken over de aanpassingen en wat deze betekenen voor het systeem in hoofdstuk 5.

2. Probleemstelling

De UNWDMI heeft een groot aantal weerstations over de wereld die elk een grote hoeveelheid aan gegevens binnen halen. Op deze gegevens worden tegelijkertijd zware selecties uitgevoerd om relevante data te onderscheiden. Aan ons is de taak om een systeem te bouwen dat deze gegevens kan verwerken en waar uiteindelijk een service aan gekoppeld kan worden. Ter voorbereiding op de bouw van de uiteindelijke applicatie is er echter eerst een proef- en demonstratie systeem gebouwd met een relationele database.

De uiteindelijke applicatie mag echter niet afhankelijk zijn van een relationele database. Daarom wordt de eerste versie aangepast zodat deze niet meer afhankelijk is van een relationele database. Hiervoor moet de applicatie aan de volgende eisen voldoen:

- Correctie van de weergegevens

- Opslag van de weergegevens op een schaalbare manier zonder relationele database

- Een voorbeeld van een selectiequery op de weergegevens

Naast het aanpassen van deze functies zal ook het aangepaste demosysteem onderworpen worden aan een stresstest. Deze stresstest moet inzicht geven in de schaalbaarheid van het systeem en hoe snel het systeem de data verwerkt.

3. Applicatie onderdelen

De applicatie is opgebouwd uit meerdere onderdelen die elk een eigen functie hebben om te voldoen aan de eisen van de opdrachtgever.

**Main**  
Dit is de hoofdklasse van de applicatie waarin de applicatie wordt gestart. Deze klasse zorgt ervoor dat alle benodigde onderdelen gestart worden en er een connectie tot stand wordt gebracht.

**ClientConnection**Deze klasse zorgt dat er een verbinding tot stand komt met de cliënt en dat de weerdata van de cliënt wordt ontvangen. Daarna roept deze klasse de XML-parser aan om de weerdata te verwerken.

**XMLparser**  
De XMLparser klasse zorgt er voor dat de XML-data die wordt geleverd door de cliënt ingelezen wordt en aangepast wordt zodat het in een database gezet kan worden.

**CorrectData**Deze klasse is voor het controleren van de gegevens en zo nodig deze gegevens te corrigeren wanneer het niet goed is doorgekomen. De correctie wordt doormiddel van een gewogen gemiddelde gedaan die wordt bepaald door voorgaande gegevens van het zelfde station.

**DatabaseQueue**De DatabaseQueue klasse verzamelt alle verwerkte XML-data zodat dit gezamenlijk in één database insert verwerkt kan worden in de database.

**DatabaseInterval**De databaseInterval klasse zorgt er voor dat verzamelde gegevens in de inseert pas na elke seconde in de database worden gestopt. Hierdoor kunnen we de datastroom naar de NoSQL(MongoDB) database beperken zodat het niet overbelast wordt door het aantal connecties.

4. Stresstest

Om te kijken of de applicatie gereed is voor dagelijks gebruik, is de applicatie een stresstest ondergaan. In dit hoofdstuk leest u waaruit de testsystemen bestonden, waarop getest is, hoe de applicatie door de test is gekomen en wat de bottleneck van de applicatie is.

## 4.1 Specificaties testsystemen

|  |  |  |
| --- | --- | --- |
| **Cliënt & Server Systeem 1 – Desktop (Rick)** | | |
| Besturingssysteem | CPU | Geheugen |
| Windows 10 | Intel Core I7-4770K 3.50 GHz | 8GB RAM 2133Mhz |
| **Cliënt & Server Systeem 2 – Notebook (Mark)** | | |
| **Besturingssysteem** | **CPU** | **Geheugen** |
| Ubuntu | Intel Core I7-4712HQ 2.3 GHz | 16GB RAM 1600Mhz |

De applicatie is op elk systeem lokaal getest om de data transfer zo hoog mogelijk te houden. Dit is gedaan om te kijken welke snelheden de applicatie aan kan met de maximale doorvoer snelheid zonder de limiterende snelheidsfactor van het internet of bijvoorbeeld satelliet.

## 4.2 Verwerkingssnelheden

De applicatie is op de verschillende systemen getest en hieruit zijn de volgende metingen naar voren gekomen:

|  |  |
| --- | --- |
| **Systeem 1 - Desktop** | |
| Max. verwerking per sec | 800 |
| Verlies bij maximale doorvoer van 800 clusters | 0 |

|  |  |
| --- | --- |
| **Systeem 2 - Notebook** | |
| Max. verwerking per sec | 800 |
| Verlies bij maximale doorvoer van 800 clusters | 0 |

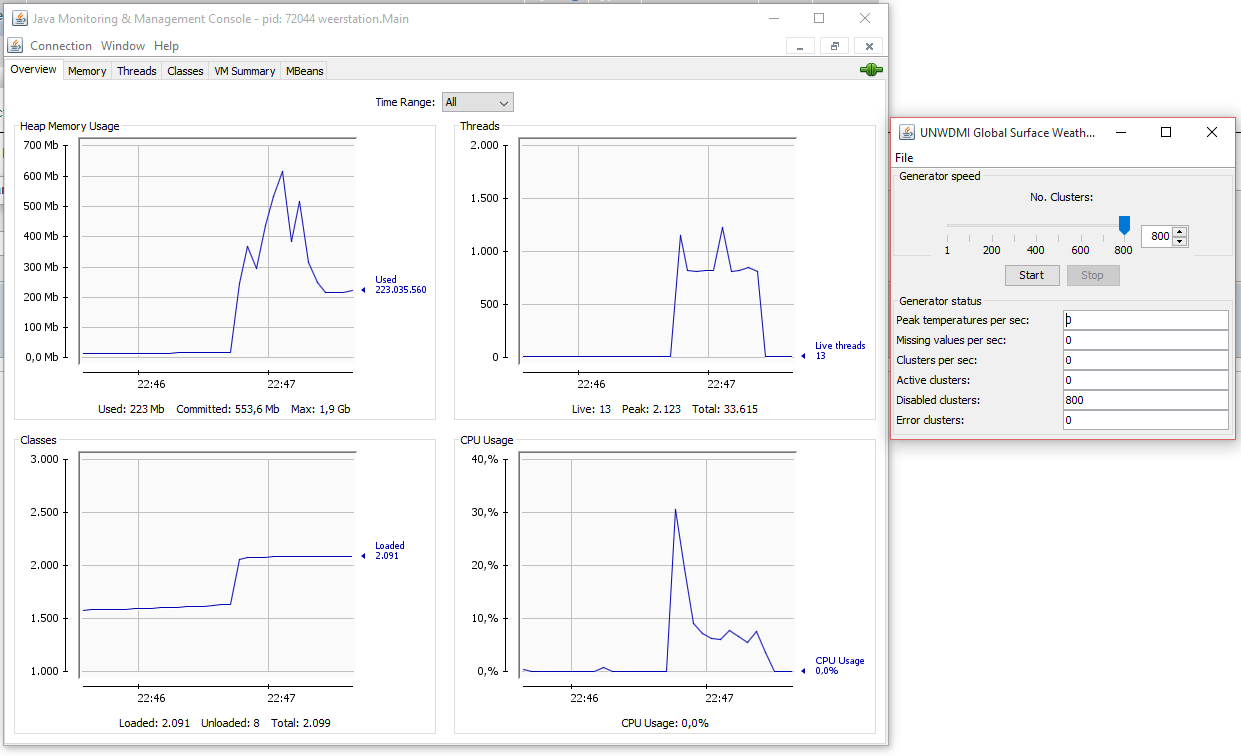
Onze testen laten blijken dat we de maximale snelheid van de stroom aan gegevens kunnen verwerken. Er is geen verlies van data en alle gegevens worden 100% in de database geplaatst.

## 4.3 Bottleneck

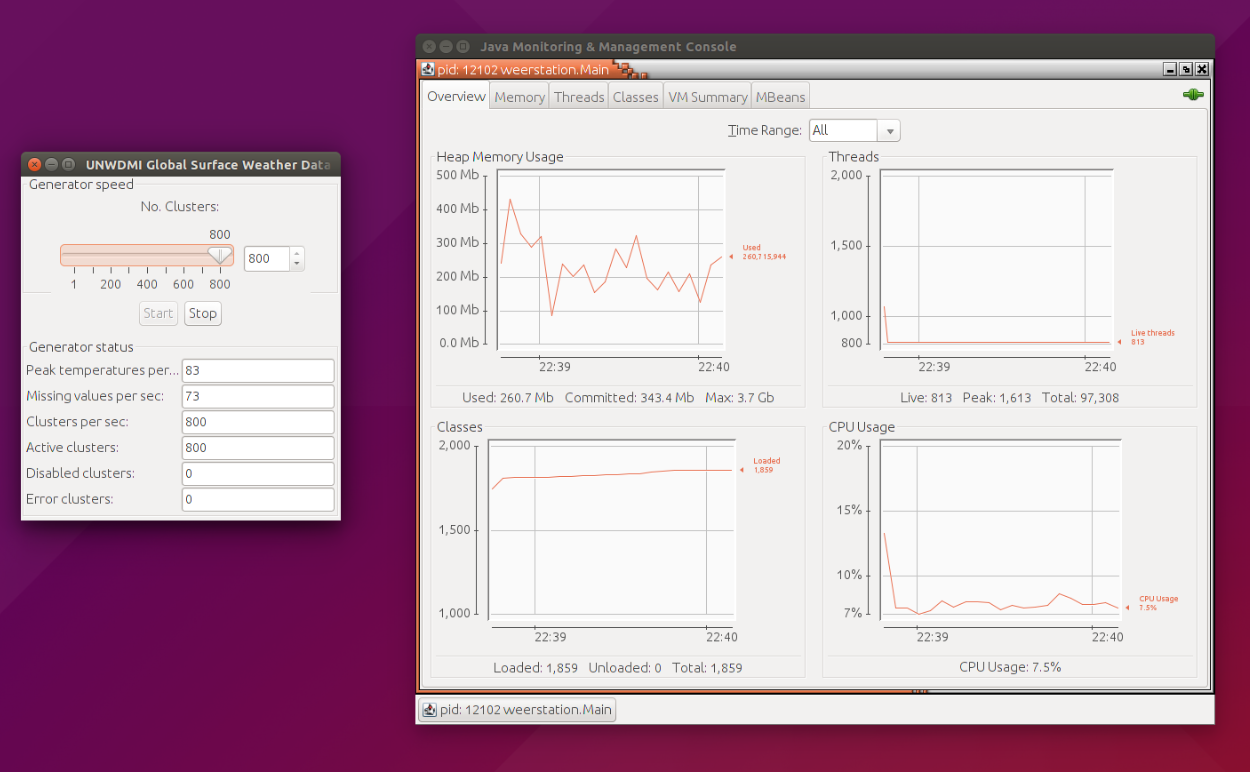
Het systeem is getest op verschillende doorvoersnelheden getest door middel van het veranderen van het aantal clusters dat de cliënt per seconden doorstuurt. Uit deze testen blijkt dat de applicatie op elke doorvoersnelheid naar behoren werkt. Echter kunnen we niet kijken of de applicatie ook met hogere snelheden ook naar behoren werkt.

Toch zijn er wel een tweak nodig aan de MySQL client om te garanderen dat de stroom aan gegevens goed worden verwerkt. Het kan namelijk voorkomen dat met de standaard MySQL instellingen de hoeveelheid packages niet verwerkt kunnen worden. Hiervoor moet in de settings de “max\_allowed\_packet” verhoogd worden. Zodra deze verdubbeld wordt zou er geen probleem voor moeten komen. Mocht de doorvoersnelheid verhoogd worden dan zou ook de “max\_allowed\_packet” waarde weer verhoogd moeten worden.

Verder lijkt het er op dat de applicatie netjes binnen de normen van het vermogen van het systeem valt. In de afbeeldingen hieronder kunt u het verbruik op de beide test systemen zien.



Figuur - Testdata systeem 1



Figuur - Testdata systeem 2

Op de afbeeldingen ziet u dat applicatie ruim binnen de normen valt qua processor en geheugen gebruik. Op het eerste systeem gebruikt de applicatie maar een maximum van ~610 MB op de 1900 MB die beschikbaar is. Daarnaast wordt ook de CPU gemiddeld maar 8 tot 10% gebruikt. Wel is er een piek te zien bij het opstarten van de applicatie maar die komt niet hoger als 31% van de CPU.

Op systeem 2 gebruikt de applicatie maar een maximum van ~420 MB op de 3700 MB die beschikbaar is. Ook wordt de CPU gemiddeld maar voor 8 tot 9% gebruikt. Ook hier is een piek te zien bij het opstarten maar deze is niet hoger dan 14% van de CPU.

De daadwerkelijke bottleneck zou waarschijnlijk niet bij de MySQL cliënt liggen. De “max\_allowed\_packet” staat standaard op 1MB en kan naar maximaal 32MB gezet worden. Toch blijft de vraag of de cliënt een groter aantal gegevens binnen een seconde kan verwerken.

Wij denken dat de daadwerkelijke bottleneck bij het binnenhalen van de XML-data en het corrigeren hiervan ligt. Een grote stroom aan gegevens kan aanzienlijk meer CPU en geheugen gebruiken en daarom de interval van het verwerken hiervan beïnvloeden.

5. Conclusie