RAPPORT LEERTAAK 1

Datum : 08-05-2014

Auteurs : Tom Broenink

Ward Holthof

Yuri Hoogeweg

School : Hanzehogeschool Groningen

Versiebeheer

Versienummer	Datum	Door	Commentaar
0.1	08-05-2014	Tom Broenink	Eerste opzet
0.2	14-05-2014	Yuri Hoogeweg	Toevoeging hoofdstuk 2
0.3	15-05-2014	Ward Holthof	Toevoegen hoofdstuk 3
0.4	15-05-2014	Ward Holthof	Toevoegen test resultaten
0.5	16-05-2014	Yuri Hoogeweg	Toevoeging Hoofdstuk 4
0.6	16-05-2014	Ward Holthof	Afwerken rapport

1. Inleiding

In dit rapport wordt Leertaak 1 beschreven. Dit rapport is geschreven door Tom Broenink, Ward Holthof en Yuri Hoogeweg.

In dit rapport beschrijven wij:

- Een verklaring van de programmaonderdelen die de gevraagde functies vormgeven.
- De resultaten van de stresstest inclusief verklaring voor de maximale snelheid van de gegevensverwerking.
    - Een overzicht van de gebruikte systemen en infrastructuur.
    - De behaalde verwerkingssnelheid.
    - Welke resource de *bottleneck* vormt en door welk proces dit wordt veroorzaakt.

De informatie zoals beschreven in dit rapport zal gebruikt gaan worden bij het ontwerpen en realiseren van Leertaak 5.

Inhoudsopgave

2. Verklaring programmaonderdelen

De applicatie bevat vijf classes die functionaliteit bevatten. Verder word de applicatie vanuit de 'main' class gestart. De main class zullen we in dit onderdeel verder niet bespreken omdat dat een simpele class is met één regel code. (Het enige wat deze class doet is een Receiver object maken en starten).

Classes:

Message

Alle messages die van clusters worden ontvangen worden gedefinieerd als een Message object. Dit object bevat alle informatie van een message in een hashmap.

Receiver

Luistert naar een poort voor nieuwe verbindingen van clusters. Wijst clusters toe aan een thread (ClusterConnection) in een threadpool. Aantal threads per pool, maximale aantal threadpools en de poort worden in de main method in de main class gespecifieerd.

ClusterConnection

Wordt per cluster aangemaakt, kijkt continu of er nieuwe messages worden ontvangen. Zodra deze ontvangen zijn worden ze doorgestuurd naar het DatabaseConnection object. De ClusterConnection class implementeert 'Runnable' en wordt dus ook als Thread gerund.

DatabaseConnection

Bevat alle informatie voor de verbinding naar de database, connection string, username/password etc. Zodra dit object een message binnen krijgt zal hij bij de MessageCorrector checken of deze message compleet en correct is.

MessageCorrector

Per station is er één MessageCorrector, deze houdt de afgelopen 30 messages bij, en kan hier waarden uit extrapoleren om te kijken of een nieuwe message wel compleet en accuraat is. Vervolgens kan de MessageCorrector aanpassingen maken voordat de message gereturned wordt naar de DatabaseConnection om verzonden te worden.

3. Stresstest

3.1 Overzicht gebruikte systemen en infrastructuur

Voor het uitvoeren van de stresstest hebben we een laptop gebruikt met de volgende specificaties:

- *Besturingssysteem*: Windows 8.1 x64
- *CPU*: Intel Core i5-4200 1.6GHz
- *Geheugen*: 4GB DDR3 RAM

De database draait op een externe server. Er wordt een connectie met de database gemaakt vanaf de laptop naar de server via het internet. Hierbij is er een latency van 7 ms.

De UNWDMI Generator draait op dezelfde machine als onze applicatie.

Specificaties van de server:

- *Besturingssysteem*: Ubuntu 14.04 x32
- *CPU*: QEMU Virtual CPU 2GHz
- *Geheugen*: 4GB DDR3 RAM
- *Hardeschijf*: 40GB SSD Disk

3.2 Gehaalde verwerkingssnelheid

We zijn begonnen door te testen op stand 1 (1 cluster per seconde). Op stand 1 worden er 10 records per seconde ingevoerd. We hebben deze instelling getest op 60 en op 120 seconden. Beide keren kon onze applicatie deze data zonder problemen verwerken.

Vervolgens hebben we de generator op stand 10 gezet. Bij 60 seconden is er al te zien dat onze applicatie moeite begint te krijgen. Bij 120 seconden is het probleem duidelijker.

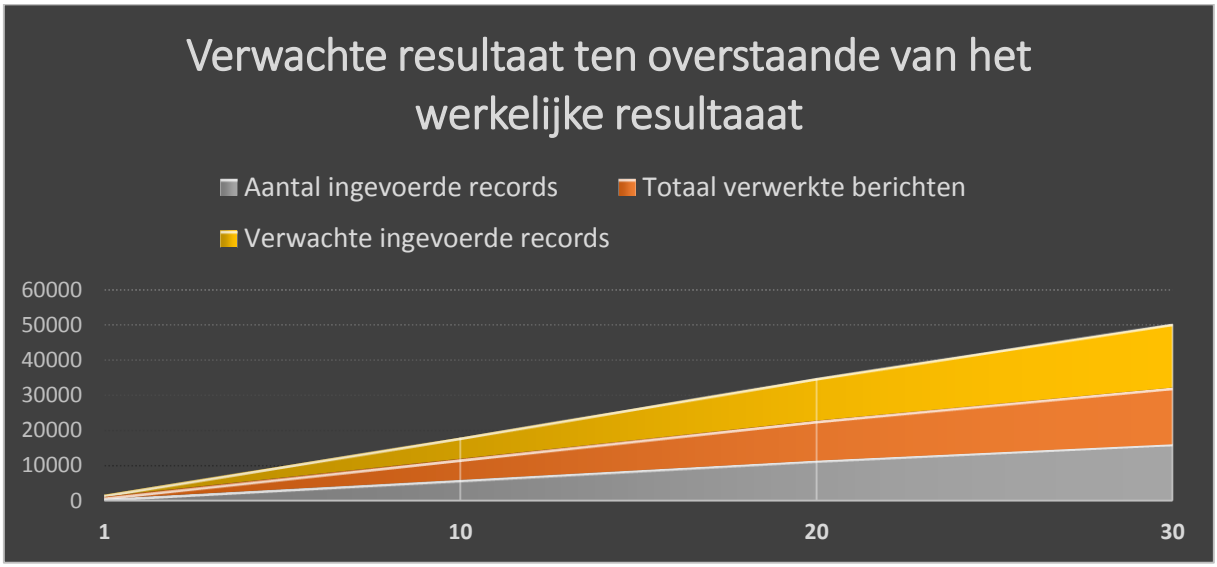
Vervolgens hebben we nog getest met stand 20, 30, 100,200 en 400. Stand 800 was niet uitvoerbaar op onze computer.

Onze gemiddelde gehaalde verwerkingssnelheid zit rond de 67 gegevens per seconde.

Onze resultaten zijn te vinden in het bijgevoegde document *testresultaten*.

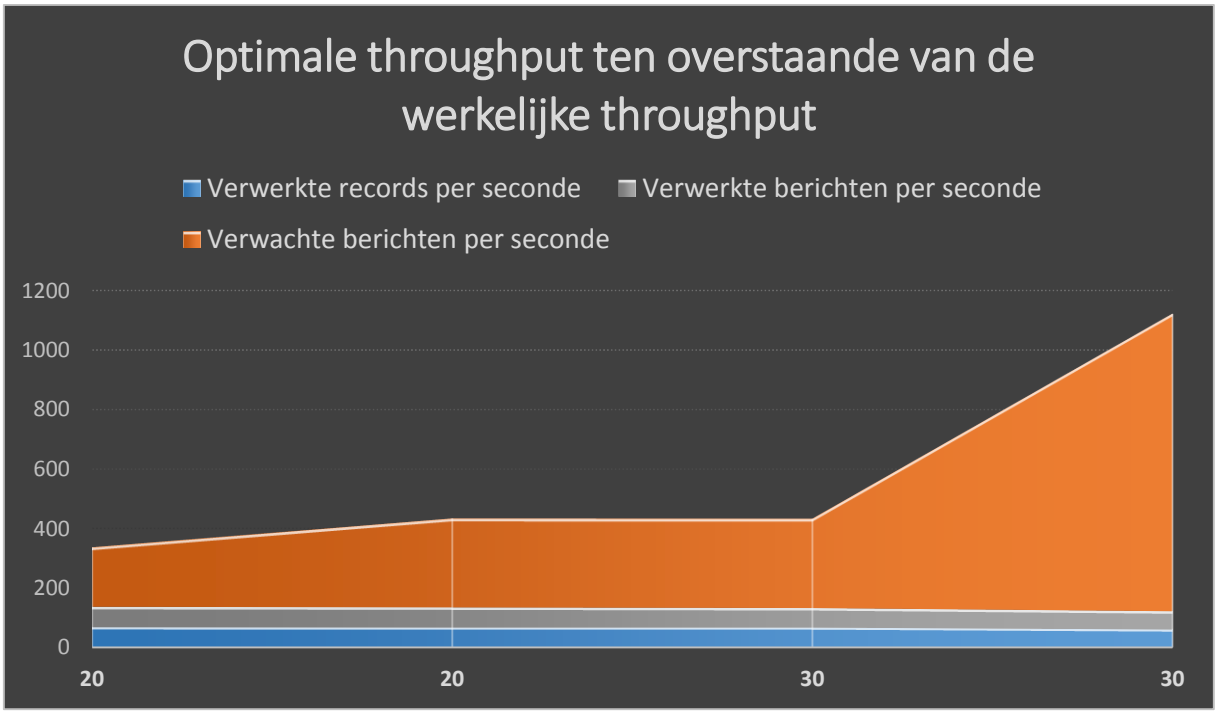
3.3 Resultaten stresstest

Uit onderstaande grafiek is af te lezen dat het aantal daadwerkelijke verwerkte data drastisch minder wordt naarmate we de generator meer data laten genereren.



Zoals te zien in bovenstaande grafiek verliezen we bij hogere standen een zeer hoog data percentage. Onze applicatie lijkt niet veel meer te kunnen verwerken dan 1 á 2 clusters per seconde.

Na verder testen blijkt dan onze maximale throughput eigenlijk op zijn max zit bij 67 metingen per seconde.

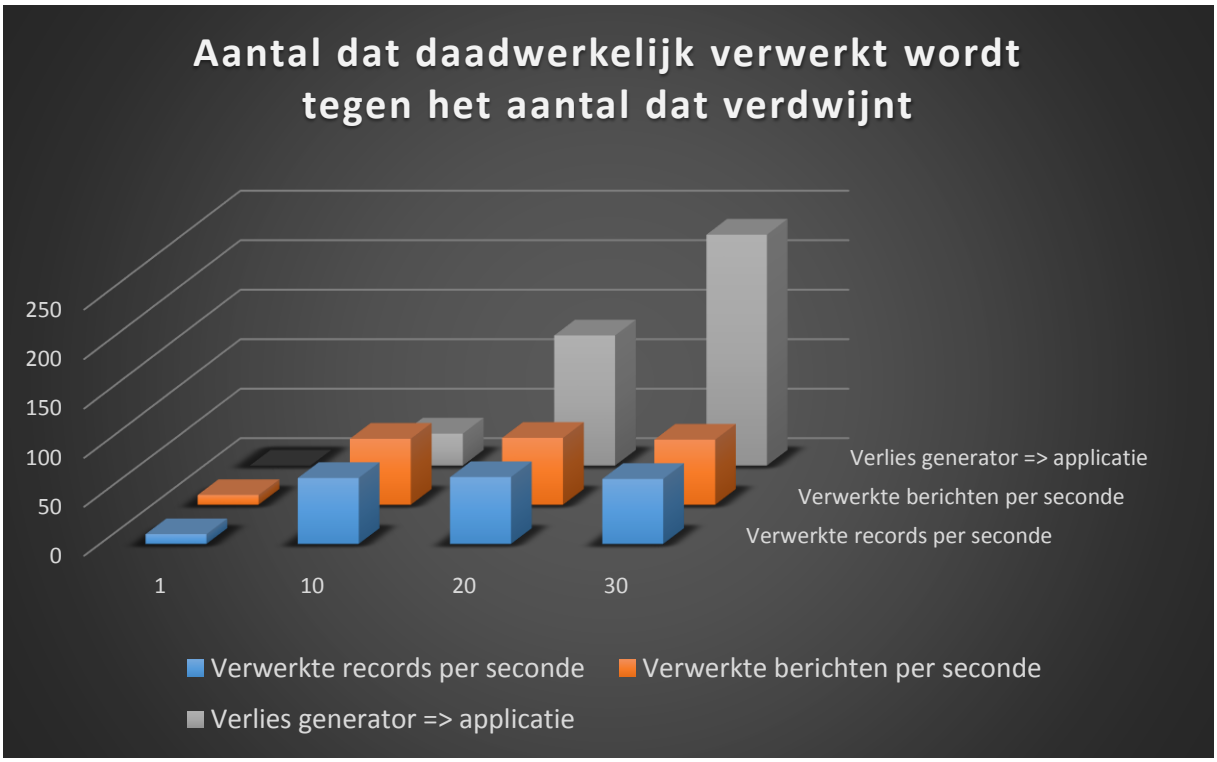


Onze applicatie kan momenteel 67 gegevens (gemiddelde) per seconde verwerken iets wat opgehoogd zal moeten worden naar 8000.

3.4 Verdwenen data

Hoe hoger we de generator zetten hoe moeilijker het wordt om alle data goed bij de database te krijgen. Een deel van de data verdwijnt op weg van de generator naar de applicatie en op weg van de applicatie naar de database.

Door uitbundig te testen hebben we aan kunnen tonen waar we data verliezen.



Zoals af te lezen aan de grafiek verdwijnt bij een hoger cluster aantal een schrikbarend aantal gegevens.

Het grootste deel van deze data wordt door onze applicatie verloren. We kunnen aan de hand van de tests zien dat we veel meer data binnenkrijgen dan dat we daadwerkelijk naar de database sturen. Onze applicatie heeft dus niet alleen moeite met het verwerken van de data, een groot deel van de data wordt helemaal verloren.

4. Conclusie

De conclusie van dit korte onderzoek is dat er nog een enorme bottleneck ligt. Deze bottleneck ligt hoogstwaarschijnlijk in de applicatie. Vermoedelijk ligt de bottleneck in de ‘messagecorrector’ van de applicatie, echter is het niet uit te sluiten dat andere onderdelen van de applicatie ook niet optimaal werken en voor vertraging kunnen zorgen. Bijvoorbeeld in het message object worden hashmaps gebruikt om elke waarde die uit de XML wordt geparsed bij te houden. Dit zorgt voor een extreem lange lookup time wanneer deze waarden opgehaald dienen te worden.

Hardwarematig is er nog geen bottleneck verschenen, zowel het CPU als RAM gebruik fluctueren niet veel. Deze blijven zelfs merkwaardig laag (op zo’n 20%).