Masterproef industriële ingenieurswetenschappen 2016-2017

Exploratie van functionele en declaratieve ontwikkelmethodes voor cloud programming

Stijn Schildermans

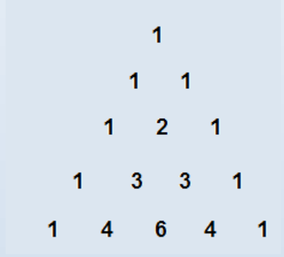
Master IW elektronica-ICT

Probleemstelling

Hoe scoren functionele talen in een cloud-context in vergelijking met hun object-georiënteerde tegenhangers?

Werkwijze

- Aan de hand van een uitgebreide literatuurstudie vergelijkt dit onderzoek de concepten en eigenschappen van enkele functionele talen. Figuur 1 geeft Alle gebruikte talen en technologieën weer;
- In de gekozen talen is een demo-applicatie ontwikkeld, gebaseerd op de Pascal-driehoek, die weergegeven is in figuur 2;
- Voor elke taal is de demo-applicatie naar minstens één cloud-platform gedeployed. Deze masterproef beschrijft dit proces in detail;
- Dit onderzoek vergelijkt de performantie van elke taal zowel lokaal als in de cloud aan de hand van de uitgewerkte demo-applicatie.



Figuur 2: Pascal driehoek [1].

Wolfram

- Hoofdzakelijk functionele taal
- Zeer symbolisch

\+ Goed gedocumenteerd
\+ Zeer compact
\+ Cloud deployment zeer eenvoudig
\- Soms inconsistent
\- Hoge connectietijd

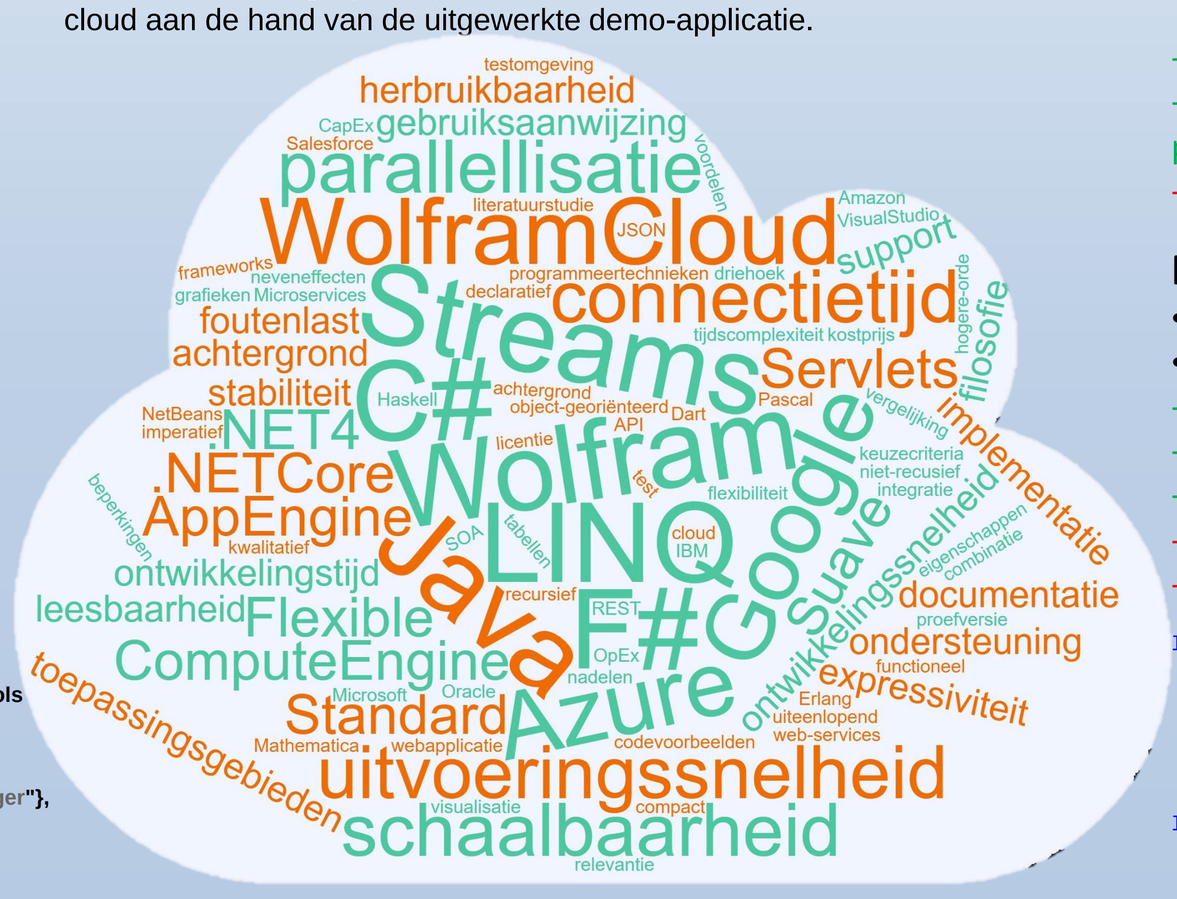
```
pascalTriangleRec[size_Integer] /; size == 1 := {{1}}
pascalTriangleRec[size_Integer] :=
Table[pascalTriangleValue[i, j], {i, 1, size}, {j, 1, i}]
pascalTriangleValue[row_Integer, col_Integer]
/; col == 1 || col == row := 1
pascalTriangleValue[row_Integer, col_Integer] :=
pascalTriangleValue[row - 1, col - 1]
+ pascalTriangleValue[row - 1, col]
```

Figuur 3: Pascal-driehoek in Wolfram.

Cloud-deployed REST-API

```
pascalTriangleDecoder[type_String, size_Integer] := Which[
type == "sequential", pascalTriangleRec[size],
type == "parallelrows", pascalTriangleRecParallelRows[size],
type == "parallelcols", pascalTriangleRecParallelCols[size],
type == "parallelrowscols", pascalTriangleRecParallelRowsCols
[size],
True, Throw["Invalid input!", invalidInputException]]

CloudDeploy[APIFunction[{"type" → "String", "size" → "Integer"},
ExportForm[Catch[pascalTriangleDecoder[#type, #size],
invalidInputException],
"JSON"] &], Permissions -> "Public"]
```

Figuur 4: RESTful web-API in Wolfram.



Figuur 1: Overzicht van de behandelde onderwerpen.

Java/C#

- Object-gerichte talen met functionele eigenschappen;

\+ Functionele implementatie compacter
\+ Functionele implementatie beter parallelliseerbaar
\- Deployment in de cloud omslachtiger

F#

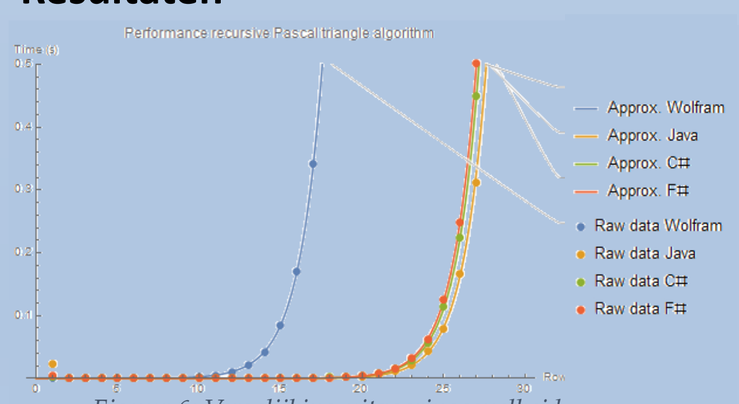
- Hoofdzakelijk functionele taal
- Sterke gelijkenissen met Haskell

\+ Gebruikt .NET runtime
\+ Eenvoudige integratie met C#
\+ Makkelijk in Azure te deployen
\- Complex voor nieuwe gebruikers
\- Weinig beschikbare cloud-platformen

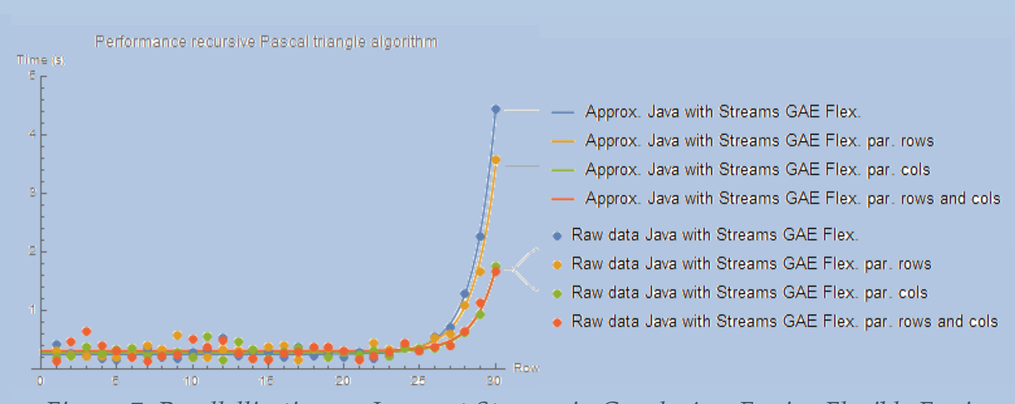
```
let rec pascalTriangleValue row col =
    match col with
        | 1 -> 1
        | col when col = row -> 1
        | _ -> pascalTriangleValue (row - 1) (col-1)
             + pascalTriangleValue (row - 1) col

let pascalTriangle size = [1..size] |>
    List.map (fun r->[1..r] |>
    List.map (fun c->pascalTriangleValue r c))
```

Figuur 5: Pascal-driehoek in F#.

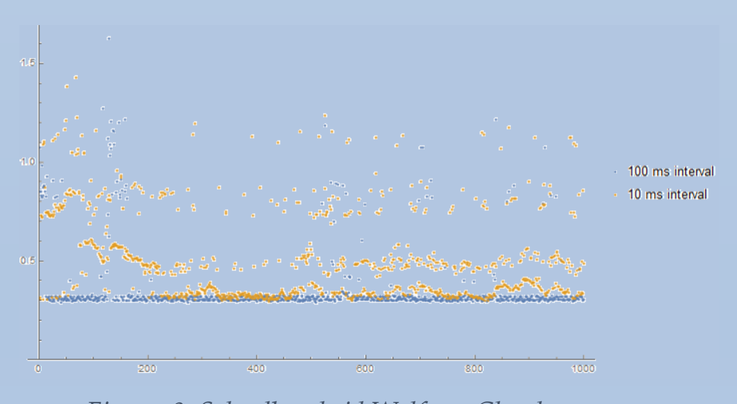
Resultaten



Figuur 6: Vergelijking uitvoeringssnelheid.



Figuur 7: Parallellisatie voor Java met Streams in Google App Engine Flexible Environment.



Figuur 8: Schaalbaarheid Wolfram Cloud.

Tabel 1: Overzicht van de belangrijkste conclusies.

Cloud platform	Taal en/of Framework	Connectie-tijd	Uitvoerings-snelheid	Snelheidswinst Parallellisatie	Schaalbaarheid
Google App Engine Standard	Java 7	0.4	Slecht	< 1x	Goed
Google App Engine Flexible	Java 8	0.2	Goed	2.5x	OK
Google Compute Engine	.NET 4.5	0.2	OK	2x	Geen
Azure	.NET Core	0.2	Slecht	3.5x	Slecht
Azure	F#	0.2	Slecht	3.5x	Goed
Wolfram Cloud	Wolfram	1.7	Goed	1x	Uitstekend

Vaststellingen

- Figuur 6, 7 , en 8 geven een overzicht van de aard van de uitgevoerde tests en de resultaten ervan. Deze lopen sterk uiteen; Tabel 1 geeft hier een algemeen overzicht van;
- Cloud deployment voor functionele talen blijkt eenvoudiger dan voor OO-talen. Alle functionele implementaties zijn compacter en leesbaarder;
- Parallellisatie is veel eenvoudiger voor functionele varianten;
- Performantie bij parallellisatie in de cloud is onder de verwachtingen;
- Schaalbaarheid van functionele talen is veel beter dan OO-talen.

Besluit

Functioneel programmeren biedt belangrijke voordelen in de cloud wat betreft programmeergemak en schaalbaarheid.

Promotoren / Copromotoren: Dr. Kris Aerts
Ing. Thomas Machiels
Ing. Leo Rutten

Bibliografie

[1] Anoniem, „Pascal Triangle Program in Java,” Sitesbay, [Online]. Available: http://www.sitesbay.com/java-program/java-pascal-triangle. [Geopend 3 april 2017].

UHASSELT

KU LEUVEN