Parallelism and Distribution

Opdracht parallellisme: parallelle analyse van online contributies

Professor: Elisa Gonzalez Boix
Assistent: Sam Van den Vonder
E-mail: svdvonde@vub.be

Deadline: 8 juni 2018, 23:59 CEST

1 Project overzicht

In dit project ga je berichten op het sociale media-platform Reddit analyseren om er achter te komen hoe mensen zich voelen tegenover bepaalde automerken (positief, neutraal of negatief). Het project bestaat uit 3 delen: een parallelle implementatie van een berichten-analyser d.m.v. Java Fork/Join, een evaluatie van de performantie, en een verslag dat de implementatie en de evaluatie beschrijft.

Deadline 8 juni 2018, 23:59 CEST.

Deliverables Maak een zip file van je code en verslag als PDF. Indien je project dependencies heeft die niet standaard in Java zitten, voeg deze dan toe. Geef de zip de naam `voornaam-achternaam-parallellisme.zip`. Op de PointCarré pagina van de cursus, ga naar *Opdrachten* > *Project Parallellisme* en dien daar je zip file in.

Evaluatie Het project bepaalt 25% van je eindscore, en zal beoordeeld worden op basis van de *correctheid van de implementatie* volgens de specificatie, de *kwaliteit van het verslag*, en de *verdediging* op het einde van het jaar. Kwaliteit van de code is ook een belangrijke factor, gebruik dus voldoende commentaar, duidelijke variabele namen, duidelijke opdeling van je code, enz.

Dit project is individueel. Je mag het natuurlijk bespreken met je mede-studenten, maar uiteindelijk moet de code volledig geschreven zijn door jezelf, en je moet ze zelf beschrijven. Als we merken dat er code gekopieerd wordt, of dat je de code niet begrijpt, bestaan hiervoor gepaste sancties.

2 Merkenanalyse

Reddit[1] is een sociaal netwerk waarop gebruikers content kunnen toevoegen en waarop andere gebruikers kunnen stemmen. De hoeveelheid stemmen bepaalt de zichtbaarheid van de content op de website. Alle content is georganiseerd in verschillende categorieën die men *subreddits* noemt. Zo zijn er subreddits voor mensen uit hetzelfde land, regio, mensen die houden van dieren, auto's, enz. Reddit is ook een plaats waar mensen hun gedachten delen met anderen. Reddit is momenteel de 7e populairste website op de wereld[2].

In dit project ga je een analyse maken van *comments* of berichten op Reddit om er achter te komen of mensen een bepaald merk positief of negatief ervaren. In eerste instantie ga je daarvoor berichten analyseren die geplaatst zijn in de subreddit die hoort bij een automerk (zoals "BMW" of "Audi"). In een tweede fase ga je een analyse maken van berichten die geplaatst zijn zowel binnen als buiten de subreddit van een automerk, zo lang het bericht de naam van het respectievelijke automerk bevat (m.a.w. wanneer "BMW" een substring is van het bericht). De analyse die je gaat uitvoeren kent een gevoelsscore toe aan elk bericht op basis van een set van positieve en negatieve woorden (en combinaties daarvan) die specifiek geselecteerd zijn voor sociale media. Dit geeft je een algemene indicatie van de mening van een auteur. Een voorbeeld van een negatief bericht is het volgende.

> *Have you seen the VW scandal how they were cheating on the diesel emission tests and the government banned them from selling diesels until they fix the problem. (source)*

Deze krijgt van de analyser een positieve score van 0, een neutrale score van 0.633, een negatieve score van 0.337, en een samengestelde score van -0.9042 ($[-1, 1]$, m.a.w. overweldigend negatief). Het volgende bericht is positief.

> *Not OP but I like my 33x12.50 Duratracs pretty well, they're definitely on par with most other ATs, plus they're great in snow and ice (source)*

Deze krijgt een positieve score van 0.505, een neutrale score van 0.495, een negatieve score van 0, en uiteindelijk een samengestelde score van 0.9657.

Wat je in dit project gaat doen is hetzelfde maar op grotere schaal: Je gaat een totaalscore berekenen van alle berichten in een bepaalde dataset en hiervan het gemiddelde nemen. Dit geeft je een indicatie over hoe positief/negatief de berichten in het algemeen zijn. Hoe meer data je verwerkt, hoe accurater je analyse zal zijn voor de gehele populatie. Je wil dus zeker genoeg data verwerken, maar het sequentieel verwerken van deze data is traag. Daarom is het jouw opdracht om dit proces te parallelliseren a.d.h.v. Java Fork/Join.

[1] http://reddit.com
[2] https://www.alexa.com/topsites

3 Implementatie

Om je te helpen voorzien we voorbeeldcode die je kan gebruiken als leidraad om de gevraagde functionaliteit te implementeren. Allereerst vind je in de `files/` directory 2 gecomprimeerde datasets in JSON formaat die je kan gebruiken bij het testen van je implementatie op je eigen computer. In `be.vub.parallellism.data` vind je code die je helpt om deze datasets uit te lezen naar een `ArrayList` met objecten van type `Comment`. In `be.vub.parallellism.solutions` vind je de klasse `SequentialAnalyser` die sequentieel een analyse gaat uitvoeren van de gegeven dataset. De lege klasse `ParallelAnalyser` is het startpunt van jouw oplossing met Fork/Join.

3.1 Fase 1

In de eerste fase van je analyse ga je alle berichten van een bepaalde subreddit analyseren. Zo kan je bijvoorbeeld alle berichten van BMW analyseren en vergelijken met de andere automerken[3]. Je maakt dus een procedure die, gegeven een lijst van berichten, al deze berichten m.b.v. Fork/Join gaat analyseren en één getal teruggeeft welke de gemiddelde gevoelsscore is. Ligt deze score hoog dan is het algemene gevoel van de berichten positief, ligt deze score laag dan is het gevoel negatief. Gegeven in de `files` folder zijn 2 datasets die je kan gebruiken om je code te testen, en de resultaten te vergelijken met de sequentiële implementatie.

3.2 Fase 2

Het kan zijn dat mensen die geïnteresseerd zijn in een bepaald automerk (en dus aangesloten zijn bij de bijhorende subreddit) van nature positiever praten over het merk. Of vanuit een ander perspectief, dat de algemene sfeer in de subreddit van een bepaald automerk positiever is dan mensen over het algemeen praten het automerk. Om deze hypothese te testen ga je in deze fase berichten analyseren *ongeacht* de subreddit waarin ze zijn geplaatst, maar *alleen* wanneer het bericht expliciet praat over het gekozen merk. Dus wanneer, bijvoorbeeld voor automerk "BMW", het woord *"BMW"* een substring is van het bericht. Ook dit zoeken van een keyword in een substring ga je oplossen met Fork/Join.

3.3 Eigenschappen

Je implementaties van zowel fase 1 als 2 hebben de volgende eigenschappen:

[3] Voorbeelden van subreddits met voldoende gebruikers zijn: BMW, Audi, Ford, Honda, Jeep, mazda, subaru, teslamotors, Toyota, Volkswagen, Volvo

1. De implementatie is *correct*. De belangrijkste eigenschap is dat je oplossing correct de gemiddelde gevoelsscore bepaalt van een gegeven set berichten.
2. De implementatie streeft naar *efficiëntie*. Gebruik Fork/Join zoals gezien in de les, en vermijd dynamische geheugenallocaties, frequente toegang tot geheugen dat waarschijnlijk niet in je cache zit, enz. Langs de andere kant, vermijd ook micro-optimisaties ten koste van de leesbaarheid van de code.
3. De implementatie exploiteert zo veel mogelijk parallellisme (m.a.w. de *span* van het probleem minimaliseren). Je zal dus op z'n minst het bepalen van de score parallelliseren (fase 1), en in tweede instantie het onderzoeken of een bericht spreekt over een bepaald automerk (fase 2).
4. Implementeer waar nodig een sequentiële cut-off (threshold). Experimenteer hiermee!

4 Performantie-evaluatie

Het hoofddoel van je experimenten is om de performantie te evalueren van je parallelle implementatie (en dus *niet* de resultaten van de merkenanalyse, ookal zijn deze interessant als streefdoel). Allereerst ga je de performantie van je oplossing benchmarken op je eigen laptop d.m.v. de bijgeleverde (kleine) datasets. Dan ga je je implementatie benchmarken op Serenity (een krachtige SOFT-server) met een grotere dataset. In je verslag (sectie 5) zal je een aantal vragen moeten beantwoorden i.v.m. de performantie van je implementatie.

4.1 Benchmarks op je eigen laptop

In eerste instantie ga je voor fase 1 en 2 benchmarks uitvoeren op je eigen laptop.

4.1.1 Dataset 1

Benchmark je code voor fase 1 met de gegeven dataset `dataset_1.json`:

1. Bereken de overhead van je implementatie ten opzichte van de sequentiële implementatie.
2. Bereken de speedup van je implementatie voor verschillende waarden van `P` (bv. 1, 2, 4, en 8 cores).

Benchmark je code voor fase 2 met de gegeven dataset `dataset_1.json`, ervan uitgaande dat *alle* berichten de gezochte substring bevatten (anders kan je deze benchmark niet vergelijken met die van fase 1). Je zal dus nog steeds het bericht opsplitsen

in substrings, maar wanneer de substring één woord bevat geef je als resultaat voor elk woord `true` (m.a.w. elk woord van het bericht is het gezochte woord). Evalueer hiermee de invloed van een sequentiële cut-off (threshold) `T` op de performantie van je implementatie.

1. Bereken voor `P=4` de overhead van je implementatie voor verschillende waarden van de threshold.
2. Bereken voor `P=4` de applicatie speed-up van je implementatie voor verschillende waarden van de threshold.

4.1.2 Dataset 2

Benchmark je code voor fase 1 met de gegeven dataset `dataset_2.json`. Hier zal je voor elk bericht sequentiëel bepalen of het woord `"BMW"` (hoofdlettergevoelig) deel uitmaakt van het bericht.

1. Bereken de overhead van je implementatie ten opzichte van de sequentiële implementatie.
2. Bereken de speedup van je implementatie voor verschillende waarden van `P` (bv. 1, 2, 4, en 8 cores).

Benchmark je code voor fase 2 met de gegeven dataset `dataset_2.json`. Hier zal je voor elk bericht in parallel bepalen of het woord `"BMW"` (hoofdlettergevoelig) deel uitmaakt van het bericht.

1. Bereken voor `P=4` de overhead van je implementatie voor verschillende waarden van de threshold.
2. Bereken voor `P=4` de applicatie speed-up van je implementatie voor verschillende waarden van de threshold.

4.2 Benchmarks op Serenity

Serenity is een krachtige server met 64 cores (4 processors met 16 cores, 2,3Ghz per core) en 128GB geheugen, en is daarom uitermate geschikt om experimenten uit te voeren waar veel parallellisme aan te pas komt. Alvorens je benchmarks gaat uitvoeren op Serenity ga je je implementatie aanpassen om een zo hoog mogelijke speedup te behalen (om dus zo veel mogelijk data te verwerken in een zo kort mogelijke tijd). Hier kan je bijvoorbeeld een sequentiële cut-off aanpassen, introduceren, of verwijderen. Leg de focus op het gebruik van technieken die je hebt gezien, en niet op micro-optimisaties of het herdefiniëren van het probleem.

Benchmark je aangepaste code als volgt:

1. Bereken de speedup van je implementatie voor verschillende waarden van `P` (bv. 1, 2, 4, 8, ...) voor de kleine datasets `dataset_1.json` en `dataset_2.json`. Bij dataset 1 bereken je de gevoelsscore van alle berichten (zoals in 4.1.1), en bij dataset 2 zoek je op berichten met `"BMW"` als substring (zoals in 4.1.2).

2. Kies een grotere dataset[4] op basis van de runtime. Studenten met een efficiënte implementatie kunnen een grotere dataset nemen (of een combinatie van datasets), anderen een kleinere. Hou er rekening mee dat je je benchmarks meerdere keren moet herhalen om tot een gemiddelde runtime te komen (zie de richtlijnen in sectie 4.3), moet herhalen met en zonder je toegevoegde optimisaties uit deze sectie, en moet herhalen voor verschillende hoeveelheid cores. Zorg dus dat je geen té grote dataset neemt waardoor het benchmarken te lang duurt voor je gegeven slot op Serenity.

4.3 Richtlijnen

Hoewel je veel vrijheid hebt in de manier waarop je je experimenten opstelt, vragen wij wel dat je dit doet op een wetenschappelijke manier gebruikmakende van de volgende richtlijnen.

- Om het parallellisme van je implementatie te meten moet je de experimenten uitvoeren op een machine met minstens 4 *fysieke* cores. Indien je niet beschikt over zo'n computer stuur je ons best een e-mail.
- Gebruik dezelfde machine en JVM voor alle experimenten die je met elkaar vergelijkt, en beschrijf de (relevante) details van beide in het verslag.
- Wees bewust van de dingen die een invloed kunnen hebben op je resultaten. Ondersteunt je processor een techniek zoals *hyperthreading*[5] of *turbo boost*[6]? Vermijd ook om applicaties te draaien die CPU- of geheugenintensief zijn tijdens je benchmarks.
- Houd er rekening mee dat benchmarken op de JVM bijna een kunst op zich is door geavanceerde technieken om de performantie te verhogen, en je hebt altijd een warm-up periode nodig waar de JVM de kans krijgt om alvast de code at-runtime te optimaliseren.

[4] Datasets van verschillende grootte zijn beschikbaar in de gedeelde read-only folder `/data/PD/` op Serenity.

[5] Processors met hyperthreading hebben meerdere hardware threads (gewoonlijk 2) per fysieke core. Het is perfect mogelijk dat je Operating System daarom zegt dat je laptop 8 logische cores heeft, terwijl de processor beschikt over 4 fysieke cores. Echter zal je zelfs in het perfecte geval hier geen 8x speed-up behalen.

[6] Turbo boost is een techniek om je processor aan op hogere frequentie te klokken in bepaalde omstandigheden, bijvoorbeeld wanneer de werklast hoog (of net laag) is, afhankelijk van stroomvoorziening (netstroom/batterij), temperatuur, enz.

- Herhaal je experimenten meerdere keren om een voldoende accuraat gemiddelde te krijgen van de run-time. Bij korte experimenten is de variatie tussen verschillende runs groter dan bij lange experimenten. Kies een voldoende hoeveelheid herhalingen en vermeld deze zeker in het verslag.
- Rapporteer over je resultaten op een wetenschappelijke manier. Maak voor elk experiment een box plot van je runtimes. Hoewel runtimes het makkelijkst zijn om met elkaar te vergelijken zijn ze niet altijd de beste manier om performantie te evalueren. Vooral bij het evalueren van parallellisme bereken je ook relatieve metrieken.
  1. Bereken de application speed-up ($\frac{T_{seq}}{T_P}$) en de computational speed-up ($\frac{T_1}{T_P}$) van je implementatie.
  2. Uit de vorige metrieken kan je de overhead van je parallelle implementatie berekenen ($\frac{T_1}{T_{seq}}$).
- Wanneer je resultaten toont in het verslag gebruik je grafische voorstellingen (bv. met box plots, histogrammen, enz) tegenover tabellen met getallen.
- Wanneer je de resultaten interpreteert maak je duidelijk wat de diagrammen laten zien, en leg je ook je resultaten uit. Waren deze resultaten verwacht? Zo ja, waarom wel? Zo niet, waarom niet? Vermijd wilde speculaties, want je kan altijd bijkomstige experimenten draaien om je hypotheses te motiveren.

5 Verslag

Het verslag is opgedeeld in 3 delen: de beschrijving van je implementatie van fase 1 en 2, de evaluatie van beiden op je eigen machine, en de evaluatie van beiden op Serenity. Je verslag heeft de volgende structuur:

1. **Overzicht**
   Geef een korte beschrijving van hoe je het probleem hebt aangepakt.
2. **Fase 1**
   - 2.1. **Implementatie**
     Beschrijf kort de strategie die je gebruikt hebt om fase 1 te parallelliseren.
   - 2.2. **Evaluatie**
     - Bespreek je resultaten met een focus op de volgende aspecten: Hoe efficiënt is je parallelle implementatie? Wat is de overhead t.o.v. de sequentiële implementatie? Is deze overhead acceptabel? Wat kan je doen om deze overhead te verkleinen? Is dit afhankelijk van de benchmark, en zo ja, waarom?

- Hoe goed maakt je parallelle implementatie gebruik van de mogelijkheden van je computer om parallelle berekeningen uit te voeren? Wat als je computer veel meer cores had, zou je implementatie schalen? Zou ze schalen voor alle benchmarks?

3. **Fase 2**

   3.1. **Implementatie**
   Beschrijf kort hoe je fase 2 hebt geïmplementeerd.

   3.2. **Evaluatie**
   Bespreek de invloed van een sequentiële cut-off (threshold) `T` op de performantie van je implementatie.

   - Op welke manier heeft de threshold invloed op de overhead van je implementatie? Leg uit waarom.
   - Wanneer je je programma moet uitvoeren op een krachtige computer, welke threshold waarde zou je nemen? Waarom geen hogere of lagere waarde? Gegeven je keuze voor threshold `T`, wat is volgens jou (in de praktijk) de toegevoegde waarde van de parallellisatie van fase 2 uit te voeren?

4. **Serenity**

   4.1. **Implementatie**
   Leg uit welke aanpassingen je hebt gemaakt aan je implementatie voor deze fase. Geef ook de redenering achter de aanpassingen, en op welke manier ze de performantie zouden beïnvloeden.

   4.2. **Evaluatie kleine datasets**
   Leg kort de behaalde resultaten uit. Voldoen deze aan je verwachtingen? Zo niet, verklaar.

   4.3. **Evaluatie grotere dataset**
   Welke dataset(s) heb je gebruikt? Leg uit a.d.h.v. je benchmarks welke invloed de veranderingen hebben in vergelijking met je oorspronkelijke implementatie.

Als je vragen hebt, aarzel niet om een e-mail te sturen naar svdvonde@vub.be. Succes!