IN1010 vår 22 obligatorisk oppgave 1: Dataklynge

Innleveringsfrist: Mandag 7.2. kl 23:59
Versjon:1.0 29/12-21

Innledning

Denne obligatoriske oppgaven skal gjøres som en individuell oppgave og skal løses på egen hånd, dvs at du skal programmere *alle* deler av løsningen selv. Dette betyr ikke at vi fraråder samarbeid – tvert imot. Vi oppfordrer deg til å utveksle faglige erfaringer med andre. Følgende er imidlertid viktig:

> Du kan diskutere en løsning med andre, men dere skal ikke dele noen deler av løsningen (f eks ved å levere inn lik kode hvor kun variabelnavn er byttet ut).
> Dersom du tar med tekst, programkode, illustrasjoner, og annet som andre har laget, må du tydelig merke det og angi hvor det kommer fra – i en selvstendig oppgave som denne er dette noe som sjelden forekommer. Hvis du er i tvil om hva som er lovlig samarbeid, må du kontakte gruppelærer eller faglærer.

Dataklynge

I denne oppgaven skal du lage et program for å holde oversikt over alle komponentene i en *dataklynge* («computer cluster» på engelsk). En slik dataklynge kan brukes til å fordele tunge beregninger på mange maskiner slik at de kan jobbe i parallell. På den måten kan en simulering som ville tatt en måned å kjøre på en vanlig maskin, kjøres på dataklyngen på noen timer i stedet. Det finnes flere dataklynger på UiO, der Saga er den største.

Dataklyngens bestanddeler

En dataklynge består av ett eller flere *rack* (et kabinett med skinner) hvor flere noder eller kretskort kan monteres over hverandre. En *node* er en selvstendig datamaskin med et hovedkort med minne og én eller flere prosessorer, i tillegg til en del andre ting. I denne oppgaven skal vi bare se på antall prosessorer (maks 16) og størrelsen på minnet. Du kan anta at en node har et heltallig antall GByte med minne (maks 4 TByte der 1 TByte = 1024 GByte).

Programdesign

Programmet ditt skal designes med bl a tre klasser som representerer henholdsvis node, rack og dataklynge. Datastrukturen inne i et objekt av klassen Dataklynge skal referere til ett eller flere rack-objekter, der hvert rack-objekt igjen refererer til ett eller flere node-objekter. Noen flere krav og tips til design av den enkelte klassen finner du videre i oppgaven.

Under overskriften **Oppgaver og levering** nedenfor finner du beskrivelsen av hva programmet ditt skal kunne utføre. Dersom du ønsker inspirasjon, kan du ta utgangspunkt i oblig 8 i IN1000 som viser det offentlige grensesnittet for hver klasse slik det ble krevd der. I denne oppgaven skal du skrive et lignende program i Java.

Klassen Node

Når et Node-objekt blir opprettet, opprettes dette med ønsket antall prosessorer og minnestørrelse. Disse er klassens private instansvariabler og verdiene settes i klassens konstruktør. For øvrig skal klassen tilby tjenester (public-metoder) som trengs i andre deler av programmet. (Hvilke metoder som trengs vil bli klarere i del C og D.)

Klassen Rack

Et objekt av klassen Rack skal inneholde en array som refererer til de Node-objektene som hører til dette racket. Når et Rack-objekt opprettes, er det ingen noder i det. Et Rack-objekt skal bl.a. inneholde en metode som legger til en ny node i racket hvis det er færre enn maks antall noder der fra før. For enkelhets skyld skal vi anta at hvert rack i

dataklyngen har plass til like mange noder. Dette antallet skal være en konstant i programmet ditt og i hele denne obligatoriske oppgaven antar vi at verdien er 12. Du kan deklarere andre instansvariabler og metoder etter behov.

Klassen Dataklynge

Et objekt av klassen Dataklynge skal holde rede på en liste med racks, og må tilby en metode som tar imot et Node-objekt og plasserer det i et rack med ledig plass. Hvis alle rackene er fulle, skal det lages et nytt Rack-objekt som legges inn i listen, og noden plasseres i det nye racket. Når en dataklynge opprettes, er den tom (inneholder ingen rack). For å implementere en liste av racks kan du bruke en passende datastruktur fra Java-biblioteket, f eks ArrayList.

Oppgaver og levering

Oppgavene du skal løse er beskrevet under. Du skal bare levere svar på oppgave E. Lever den ferdige løsningen i Devilry med en .java-fil for hver klasse:

- Dataklynge.java
- Node.java
- Rack.java
- Hovedprogram.java

I tillegg skal du levere tegning fra deloppgave A. **Ikke** lever .zip-filer eller liknende.

Besvarelsen skal kunne kjøres på ifi sine Linux-maskiner. Bruk bare de delene av Java som er forelest og finnes i læreboka. Avanserte utviklingsverktøy (IDE-er) skal ikke være nødvendig for å lage programmet.

Del A: Datastrukturtegning

VIKTIG: Les følgende instruksjoner før du løser denne deloppgaven:

1) I IN1010 skal tegningen være i tråd med notasjonen beskrevet i **notatet om datastrukturer**. Tegningen leveres som en egen fil i .png, .pdf eller lignende format.

2) Denne deloppgaven skal løses basert på programmet slik det ser ut etter at deloppgave B, C og D er løst, men før vi legger til funksjonalitet for lesing fra fil i deloppgave E.

Tegn datastrukturen slik den ville sett ut etter at vi har satt inn følgende noder i et nytt objekt av Dataklynge. La maksimalt antall noder per rack være 2.

- Node 1: 2 prosessorer og 128 GByte minne,
- Node 2: 8 prosessorer og 1 TByte minne.
- Node 3: 2 prosessorer og 512 GB minne

Relevante Trix-oppgaver: **1.09, 1.10 og 1.18.**

Del B: Klasser

Skriv klassene Dataklynge, Rack og Node.

Relevante Trix-oppgaver: **2.08, 2.16, 2.18 og 3.01.**

Del C: Antall prosessorer og minnekrav

Lag en metode antProsessorer i Dataklynge som returnerer det totale antall prosessorer i dataklyngen.

Noen programmer trenger mye minne, typisk et gitt antall GByte med minne på hver node vi bruker. Vi er derfor interessert i å vite hvor mange noder som har nok minne til at vi kan bruke dem. Lag en metode

noderMedNokMinne(int paakrevdMinne) i Dataklynge som returnerer antall noder med minst paakrevdMinne antall GByte minne.

Utvid klassene Node og Rack slik at de støtter implementeringen av disse metodene.

Relevant Trix-oppgave: **2.24a**

Del D: Hovedprogram

Skriv en klasse Hovedprogram med en main-metode for å teste at klassene virker som de skal. Lag en dataklynge med navn saga. Legg inn 450 noder med fire prosessor og 512 GByte minne hver. Legg også inn 16 noder med 8 prosessorer og 1024 GByte minne hver.

Sjekk hvor mange noder som har minst 128 GByte, 512 GByte og 1024 GByte minne. Finn totalt antall prosessorer, og sjekk hvor mange rack som brukes. Skriv ut svarene i terminalen. Utskriften kan f.eks. se slik ut:

```
Noder med minst 128 GB: 466
Noder med minst 512 GB: 466
Noder med minst 1024 GB: 16

Antall prosessorer: 1928
Antall rack: 39
```

Del E: Lese fra fil

Skriv et nytt hovedprogram som leser data om regneklyngen fra fil. Filen er bygget opp slik at hver linje beskriver et antall like noder som skal settes inn. Først på hver linje står hvor mange slike noder det skal være og deretter hvor mange prosessorer og hvor mye minne disse nodene har, slik:

```
AntallNoder  AntallProsessorerPerNode  MinnePerNode
AntallNoder  AntallProsessorerPerNode  MinnePerNode
...
```

For dataklyngen med bestanddeler som i deloppgave D blir innholdet i filen slik (dataklynge.txt):

```
450 4 512
16 8 1024
```

La enten filnavnet være en parameter til programmet eller bruk filen "dataklynge2.txt" som du finner i oblig 1-mappen på semestersiden under obligatoriske innleveringer. Gjør deretter sjekkene, beregningene og utskriftene som er beskrevet i deloppgave D. Test gjerne med flere variasjoner av data i filen og sjekk at du får riktig resultat (eksempel på flere testfiler finner du også i oblig 1-mappen). Skriv programmet slik at det terminerer på en hyggelig måte selv om det er feil i datafilen (som i dataklynge4.txt), i alle fall om det er for mange prosessorer eller for mye minne i en node.

Merk: Du trenger bare å levere den siste versjonen av klassene for Dataklynge og Hovedprogram, det vil si klassene slik de ser ut etter at du har løst deloppgave E.

Relevante Trix-oppgaver: **2.12, 2.17 og 2.22.**