IN1010 V17, Obligatorisk oppgave 1: Regneklynge

Innleveringsfrist: Tirsdag 06.02. kl 10:00

Versjon 2018.1.0 Kristian Hustad, modifisert av Siri Moe Jensen og Silje Merethe Dahl

Innledning

I denne oppgaven skal du lage et program for å holde oversikt over alle komponentene i en regneklynge ("computer cluster" på engelsk). En slik regneklynge 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 regneklyngen på noen timer i stedet. Det finnes flere regneklynger på UiO, der Abel er den største.

Oppgaven er i store trekk den samme som oblig 8 i IN1000 høsten 2017, men skal besvares i Java i stedet for Python – og uten oppgitt grensesnitt i Java.

Regneklyngens bestanddeler

En regneklynge består av ett eller flere rack (et kabinett med skinner) hvor mange noder kan monteres over hverandre. En node er en selvstendig maskin 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 2) og størrelsen på minnet. Du kan derfor anta at en node har enten en eller to prosessorer og et heltallig antall GB med minne.

Programdesign

Programmet skal designes med tre klasser som representerer noder, racks og regneklynge. Et objekt av klassen regneklynge skal kunne referere til ett eller flere rack-objekter, der hvert rack-objekt igjen refererer til en 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 vedlagte fil som viser det offentlige grensesnittet for hver klasse fra fjorårets oppgave (NB: Python). Her finner du en dokumentert signatur (metode-hode) for de metodene klassene skal tilby utad, og som du skal implementere (skrive ferdig). Det kan i tillegg være nyttig å implementere andre hjelpemetoder - gjør egne vurderinger og gi en begrunnelse for disse gjennom kommentarer i koden.

Klassen Node

Klassen skal kunne initiere nye objekter med ønsket minnestørrelse og prosessorantall, og for øvrig tilby tjenester (metoder) som trengs i andre deler av programmet.

Klassen Rack

Klassen Rack skal lagre Node-objektene som hører til et rack i en liste. Vi skal kunne legge til noder i racket hvis det er færre enn maks antall noder der fra før. For enkelhets skyld skal vi anta at hvert rack i regneklyngen har plass til like mange noder. Opprett andre instansvariable og metoder etter behov.

Klassen Regneklynge

Klassen Regneklynge skal holde rede på en liste med racks, og må tilby en metode som tar imot et nodeobjekt 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.

Tips: Det kan være lurt å ta inn antall noder per rack i konstruktøren til Regneklynge.

Oppgaver og levering

Oppgavene du skal løse er beskrevet under. Lever den ferdige løsningen i Devilry med en fil per klasse og en egen klasse for hovedprogrammet, samt tegning fra deloppgave A). Legg også ved eventuelle tekstfiler som er nødvendige for at hovedprogrammet skal fungere. **Ikke** lever .zip eller liknende.

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 datastrukturtegning. 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 Regneklynge. La max antall noder per rack være 2.

- Node 1: 16 GB minne, 1 prosessor
- Node 2: 16 GB minne, 1 prosessor
- Node 3: 128 GB minne, 2 prosessorer

Relevante Trix-oppgaver: 1.04, 1.25b, 1.26, & 2.19.

Del B: Klasser

Skriv klassene beskrevet under “Programdesign”. Der det ikke er oppgitt hva slags datatype som skal brukes (for eksempel ved valg av listetype) forventes det at du selv gjør fornuftige valg og begrunner disse gjennom kommentarer i koden.

Relevante Trix.oppgaver: 2.09, 2.10, 2.15 & 3.01.

Del C: Antall prosessorer og minnekrav

Lag en metode *antProsessorer* i Regneklynge som returnerer det totale antall prosessorer i regneklyngen.

Noen programmer trenger mye minne, typisk et gitt antall GB 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 Regneklynge som returnerer antall noder med minst *paakrevdMinne* GB minne.

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

Del D: Hovedprogram

Skriv en klasse *Hovedprogram* med en *main*-metode for å teste at klassene virker som de skal. Lag en regneklynge, *abel*, og la det være plass til 12 noder i hvert rack. Legg inn 650 noder med 64 GB minne og en prosessor hver. Legg også inn 16 noder med 1024 GB minne og to prosessorer.

Sjekk hvor mange noder som har minst 32 GB, 64 GB og 128 GB 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 32 GB: 666
Noder med minst 64 GB: 666
Noder med minst 128 GB: 16

Antall prosessorer: 682
Antall rack: 56
```

Del E: Lese fra fil

Skriv en ny konstruktør til klassen Regneklynge. Denne skal ha et filnavn som parameter (i stedet for max antall noder per rack), og lese alle data om regneklyngen fra fil. Filen er bygget opp slik at første linje beskriver antall noder per rack, mens de neste linjene beskriver hvor mange noder, med hvor mye minne og antall prosessorer som skal settes inn, slik:

```
Max antall noder per rack
AntallNoder MinnePerNode AntallProsessorerPerNode
AntallNoder MinnePerNode AntallProsessorerPerNode
...
```

For regneklyngen med bestanddeler som i deloppgave D blir innholdet i filen slik (*regneklynge.txt*):

```
12
650 64 1
16 1024 2
```

Endre hovedprogrammet fra deloppgave D slik at du oppretter en regneklynge med data fra filen "*regneklynge.txt*" (som vist over) før det gjør beregningene beskrevet i deloppgave D). Test gjerne med flere variasjoner av data i filen og sjekk at du får riktig resultat (eksempelfiler finner du på informasjonssiden for oppgaven).

Merk: Du trenger bare å levere den *siste* versjonen av klassene for Regneklynge og Hovedprogrammet, det vil si klassene slik de ser ut etter at du har løst denne deloppgaven.

Relevante Trix-oppgaver: 2.12, 2.18 og spesielt 2.20.