Frivillig prosjektoppgave 2019: Datasenter

IN1000

Kick-off på FUI-seminar 15.11. Oppsummering Fredags-Python 29.11

Innledning

I denne oppgaven skal du lage et program for å holde oversikt over en rekke *regneklynger* (eng: computer cluster) og alle komponentene i en *regneklynge*. 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, hvorav Abel er den største.

Datasenter og regneklyngens bestanddeler

I denne oppgaven består et `Datasenter` av en eller flere regneklynger. 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 én eller flere prosessorer og minne, 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 en eller to prosessorer og et heltallig antall GB med minne.

Programdesign

Programmet skal designes med fire klasser som representerer noder, racks, regneklynge og datasenter. Et objekt av klassen `Datasenter` skal kunne referere til ett eller flere objekter av klassen 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 mer hjelp på veien kan du ta utgangspunkt i vedlagte fil som viser det *offentlige grensesnittet* for hver klasse. 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 hjelpemetoder (non-public metoder) som kun brukes innenfor den enkelte klasse (angis i Python med tegnet '_' foran metodenavnet).

Dersom du ønsker å arbeide mer selvstendig med oppgaven og lage dine egne grensesnitt for klassene er det flott – så lenge programmet omfatter disse fire klassene og har funksjonalitet som beskrevet under "Oppgaver og levering" nedenfor. Hvis du velger dette alternativet er det ekstra viktig at du kommenterer valgene dine.

Klassen Node

Klassen `Node` 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 en regneklynge har plass til like mange noder (men dette maks-tallet kan variere fra regneklynge til regneklynge). 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`.

Klassen Datasenter

Klassen `Datasenter` skal ta vare på en ordbok med regneklynger. Klassen skal også kunne lese inn regneklynger fra fil. For å få til dette trenger `Datasenter` en metode som tar imot et filnavn og oppretter en regneklynge basert på informasjonen i filen. Filnavnet vil bestå av regneklyngens navn og filendelsen .txt. Du kan bruke regneklyngens navn som nøkkel i ordboken og du kan anta at ingen regneklynger heter det samme.
Filen det skal leses inn fra har følgende format:

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

Klassen `Datasenter` trenger også en metode for å skrive ut informasjon om én spesifikk regneklynge og en metode for å skrive ut informasjon om alle regneklyngene i datasenteret.

Oppgaver

Vi anbefaler at programmet skrives med en fil per klasse og hovedprogram, og at dere starter med å lage en tegning som beskrevet i oppgave a).

a) Datastrukturtegning

Bruk gjerne notasjon fra oblig 8 og forelesninger: Variable (inkludert strenger) tegnes som navngitte bokser med verdien inni. Objekter (inkludert lister og ordbøker) tegnes som firkanter (runde hjørner) med instansvariable inni. Klassenavn kan skrives over objektet med «:» foran. Referanser tegnes som piler fra variabelen de ligger i til objektet de refererer til.

Tegn datastrukturen slik den ville sett ut etter vi hadde satt inn et objekt av `Regneklynge` i et objekt av `Datasenter`. Max antall noder per rack for denne regneklyngen skal være 2. I regneklyngen er det satt inn følgende noder:

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

b) Antall prosessorer og minnekrav

Lag en metode `antProsessorer(self)` 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(self, 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.

c) Testing underveis

Når du har laget klassen `Regneklynge` kan det være lurt å teste at alt fungerer slik det skal.
Forslag til testprogram:
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
```

d) Hovedprogram

Lag et `Datasenter` som tar inn regneklyngene `saga` og `abel` fra fil. For hver av regneklyngene skal du skrive ut navnet på regneklyngen, antall racks, antall prosessorer og hvor mange noder som har minst 32 GB, 64 GB og 128 GB minne. Utskriften kan f.eks. se slik ut:

```
Informasjon om regneklyngen Saga
Antall rack: 56
Antall prosessorer: 682
Noder med minst 32 GB: 666
Noder med minst 64 GB: 666
Noder med minst 128 GB: 16
```

Gjennomgang av løsninger

Dette er ikke en obligatorisk oppgave, og løsninger skal ikke leveres i Devilry - de vil heller ikke bli rettet av gruppelærere. Men fredag 29.11 12:15 til 14:00 blir det IN1000-avslutning med peer review av de ferdige løsningene.

To og to grupper (evt enkeltstudenter) går da gjennom hverandres løsninger i henhold til en sjekkliste og gir tilbakemeldinger. Deretter går man sammen gjennom de to løsningene og sammenligner og diskuterer forskjeller.

Det vil være lærere tilstede for å svare på spørsmål - og litt enkel juleservering. Dette blir siste IN1000-tilbud før eksamen 6. desember.