18. mars 2021

Oblig 4 i IN3030/IN4330 – v2021
Parallell Radix med variabelt antall sifre

Innlevingsfrist: Onsdag 14. april 2019 23:59:00 Oslo lokal tid

Radix-sortering (eller mer presist: HøyreRadix-sortering) er en vel kjent algoritme som sorterer en array med heltall ved å flytte elementene mellom to arrayer a[] og b[] av samme lengde n. Ved hver flytting blir tallene sortert på ett siffer, og algoritmen starter med å sortere tallene på det siste, minst signifikante sifferet (det lengst til høyre i tallene). Ved neste sortering sorteres det på det nest minst signifikante sifferet,.. osv. helt til alle sifrene i tallene er sortert.

Et siffer er et bestemt antall bit, alt fra 1 til 30, og vi velger selv hvor stort siffer vi sorterer med. Når vi sorterer med flere sifre, behøver ikke alle sifrene være like lange. Er et siffer numBit langt, er de mulige sifferverdiene **: 0..2^{numbit}** -1 (eks. er sifferet 8 bit langt, er sifferverdiene: 0,1,..,255). Effektivitetsmessig lønner det seg at det er fra 8-11 bit i et siffer. Den sekvensielle koden for Radix-sortering er finnes i precoden på github.

Denne versjonen av Radix-sortering som dere skal parallellisere i Oblig4, MultiRadix, velger selv hvor mange sifre den vil sortere på avhengig av hvor mange bit det er i Max-verdien i arrayen a []. Algoritmen inneholder 4 steg:

Radix består av 4 steg – flere valg for datastruktur og oppdeling ved parallellisering

a) Finn max verdi i a[]
b) count= opptelling av ulike sifferverdier i a[]
c) Summer opp i count[] akkumulerte verdier (pekere)
d) Flytt elementer fra a[] til b[] etter innholdet i count[]

Innlevering

Som du ser inneholder de to algoritmene til sammen 4 steg: a), b), c) og d) som du skal parallellisere. Steg a) har vi parallellisert før (se forelesningsnoter) og løsningen for steg b) blir skissert og lagt på hjemmesida. Du må parallellisere alle de resterende stegene du ikke har kode for og lage en sammenhengende parallell algoritme som sorterer i parallell og som har speedup > 1 for 'tilstrekkelig stor n'.

Det du skal levere er programkoden og en rapport som først viser kjøretider, og speedup for n= 1000, 10 000, 100 000, 1 mill, 10 mill og 100 mill.

NB: Max kjøretid for sekvensielt program for 100 million: 10 s!

Veiledning til rapport blir lagt ut på hjemmesida.

Løsningen skal også inneholde en enkel test på om arrayene er sortert (a[i-1]≤ a[i], i=1,2,..,a.length-1) og den skal gi en feilmelding med for hvilken indeks det først evt. var feil i den sorterte arrayen. Denne testen kjøres (utenfor tidtakingen) både etter den sekvensielle og parallelle sorteringa. Etter den parallelle sorteringa må du sjekke at resultatet er det samme for begge algoritmer.

Du skal utskrive dele av resultatet ved at bruke en klasse i precoden, som blir lagt ut på github. Beskriv deretter med egne ord i rapporten hvordan steg c) ble parallellisert. Tips: Det kan hende at trådene bør ta en ekstra kopi av count[] som bestemmer hvor den skal flytte sine: 0-er, 1-ere,….,

Tallene som skal sorteres skal trekkes uniformt mellom 0..n*4-1 (og denne trekkingen holdes utenfor tidtakingen). Dere skal bruke en tilfeldigtalgenerator i precoden, som blir lagt ut på github.

I alle fall skal tidene du levere beregnes som medianen av minst 7 kjøringer for hver verdi av n.

IN4330 studenter må også bruke Java Measurement Harness og – kort – kommenterer hva de fik ut av målingene.

Obliger i IN3030 innleveres i **Devilry**. Husk at det sammen selve koden på begge punktene skal ligge en rapport – i PDF format – med tabeller og forklaringer over kjøretidene som beskrevet ovenfor. Rapporten må følge den publiserte veiledning.

Oblig4 leveres individuelt.

Kildekode for MultiRadix-sortering bliver publisert på github.