NTNU
Det skapende universitet

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i
TDT4225 Lagring og behandling av store datamengder
Kontinuasjonseksamen

Faglig kontakt under eksamen: Kjell Bratbergsengen

Tlf.: 906 17 185 / 7359 3439

Eksamensdato: Fredag 15. august 2014

Eksamenstid (fra-til): 0900 - 1300

Hjelpemiddelkode/Tillatte hjelpemidler: *D*
Ingen trykte eller håndskrevne hjelpemiddel er tillatt.
Bestemt, enkel kalkulator tillatt

Annen informasjon:

Sensur innen: mandag 1. september 2014.

Målform/språk: Bokmål

Antall sider: 3

Antall sider vedlegg: 0

Kontrollert av:

Svein Erik Bratsberg

_____

Dato Sign

TDT4225 Lagring og behandling av store datamengder
Fredag 15. august 2014, kl. 0900-1300

Oppgave 1, diskkontrollere, magnetisk disk og flashdisk (SSD) (10 %)

a) Beskriv oppgavene til en kontroller for magnetiske disker.

b) Beskriv oppgavene til en kontroller for flashminne.

Løsning
Les i boka.

Oppgave 2, RAID 1, speilte disker (20 %)

Blokk nr	Disk 0
0	
1	
2	
3	
4	
5	

Blokk nr	Disk 1
0	
1	
2	
3	
4	
5	

Figuren viser et RAID 1-system hvor de to diskene skal inneholde identiske data. Diskene er av samme type. Hver disk har en kapasitet på 2 TB, antall rotasjoner per sekund (ω) er 250 og hvert spor lagrer 1000 KB.

a) Hvor stor er hver disks maksimale lese- og skrivehastighet?

For spørsmålene b, c og d gjelder: Systemet er satt opp til å bruke en blokkstørrelse på 64 KB, og adressene er vilkårlige, dvs. ”randomisert” lesing og skriving.

b) Hvor mange blokker kan systemet *lese* per sekund?

c) Hvor mange blokker kan systemet *skrive* per sekund?

d) Hvor mange blokker kan systemet *oppdatere* per sekund?

e) Hvis en av diskene må skiftes ut, hvor lang tid tar det å overføre dataene til den nye disken? Anta at hele disken må skrives og at annen trafikk er blokkert i perioden.

Løsning

a) Maksimal lese- og skrivehastighet er $\omega Vs = 250 \times 1{,}0$ MB/s = 250 MB/s

b) Konservativt regner vi at hver lesing og skriving på vilkårlig adresse krever 2 rotasjoner. Dermed klarer vi 125 blokker per disk per sekund. Vi kan lese fra to disker og totalkapasitet blir da 250 lesinger per sekund.

c) Samme datablokk må skrives til begge diskene. Det innebærer posisjonering, rotasjonsventetid og selve skrivingen. Vi regner som for lesing at en skriving tar 2 rotasjoner, i.e. 125 skrivinger per sekund.

d) Oppdatering krever både lesing og skriving. Begge disker må oppdateres, men vi trenger bare å lese en disk. Lesing krever 2 rotasjoner, tilbakeskriving vil gå på en rotasjon for den disken som er lest. For den disken som bare skal skrives bruker vi to rotasjoner. Totalt krever oppdatering 5 rotasjoner ut av et totalt rotasjonsbudsjett for to disker på 500 rotasjoner per sekund. Gjennomsnittlig klarer systemet 500/5=100 oppdateringer per sekund.

e) Både lesing og skriving er sekvensiell. Hvis vi bruker blokkstørrelse på 64 KB trenger vi en rotasjon per blokk:
$T = \frac{2 \times 2^{40}}{2^6 \times 2^{10} \times 250} = 134218$ sekunder, litt over 37 timer.
Har her regnet at en T er 2^{40}.

Vi bør øke blokkstørrelsen til minimum ett spor, dvs. 1 MB. Da trenger vi to rotasjoner per spor og tiden reduseres til $T = \frac{2 \times 2 \times 2^{40}}{2^{20} \times 2^{10} \times 250} = 16777$ sekunder, som er vel 4 timer.

Den minimale tiden får vi ved å dividere med maksimal overføringshastighet, $T_{min} = \frac{2 \times 2 \times 2^{40}}{2^{20} \times 250}$ = 8389 sekunder eller ca. 2 timer 20 minutter.

Oppgave 3, lagringsstrukturer (15 %)

a) Beskriv hvordan en R-trefil for to-dimensjonale objekter er organisert.

b) Beskriv i korte trekk hvordan en søker etter objekter som ligger helt eller delvis innenfor et gitt areal.

c) Et nytt objekt skal settes inn. Hvordan bestemmes hvilken datablokk som skal få objektet?

d) Hvis ingen relevant datablokk har plass for det nye objektet, hvordan skaffes ny plass?

Løsning
Les i boka.

Oppgave 4, sortering (15 %)

Du skal sortere en fil med 200 millioner poster, postlengde er100 byte, derav utgjør nøkkel 12 byte. Til rådighet har du 200 MB arbeidslager. CPU-hastigheten er slik at du kan regne med at gjennomsnittstiden for å sammenligne to poster er ett mikrosekund.

Alle filer (innfil, mellomlagerfiler og resultatfil) ligger på samme *flashdisk* (SSD) som har følgende parametre: Fast blokkstørrelse er 4 KB, lesing og skriving tar begge 0,1 ms per blokk.

a) Hvor lang tid trenger CPU for å gjøre initiell sortering?

b) Anta at IO og CPU arbeider parallelt.
Hvor lang tid tar hele sorteringen?

Løsning

a) Vi bruker reservoarsortering med tapertre. Reservoaret er stort; 200 MB. Postlengden er 100 byte, i tillegg trenger vi 8 eller 16 byte per post i administrative data. Reservoaret har altså plass for $\frac{200\times2^{20}}{116} = 1807889$ poster. Høyden i tapertreet er: $h = \lceil \log_2 1807889 \rceil = 21$.
Totalt CPU-arbeid for å gjøre initiell sortering er $T_{initiell}^{CPU} = 2 \times 10^8 \times 1 \times 21 = 4200 \times 10^6 \mu s = 4200\ s$.

Hele datamengden er: $200 \times 100\ MB = 20\ GB$. Lesing eller skriving av denne datamengden tar totalt $T_{IO} = \frac{20\times2^{30}}{4\times2^{10}} \times 0{,}0001 = 524\ sekunder$. Under initiell sortering skal hele datamengden både leses og skrives, det tar 1048 sekunder. For initiell sortering er CPU flaskehals.

Antall delfiler er: $N = \frac{20\ GB}{2\times200\ MB} = 50$. Under fletting vil tapertreets høyde være:
$h_{fletting} = \lceil \log_2 50 \rceil = 6$.

Totalt CPU-arbeid under fletting er $T_{fletting}^{CPU} = 200 \times M \times 6\mu s = 1200\ s$. Total IO under fletting (lesing og skriving) krever 1048 sekunder; igjen så er det CPU som er flaskehals.

Total sorteringstid er: 4200+1200=5400 sekunder.

Oppgave 5, Bloomfilter (20 %, c-spørsmålet teller halvparten)

a) Forklar hvordan Bloomfilter virker.

b) Du skal lagre 10 millioner poster med unike nøkler. Hvor mye plass vil du da avsette til Bloomfilteret? Forklar hvordan du kommer fram til svaret.

c) Et kredittkortselskap har utstedt ca. 300 millioner kredittkort. Ca. 5 % av kortene er blokkert pga. mislighold, tyveri, og lignende. For hver transaksjon må en kontrollere om kortet er gyldig. Kredittkortets nummer er på 16 siffer. Skisser et system (datastrukturer) som effektivt kan brukes til å utføre gyldighetskontrollen. Beregn hvor stor plass (arbeidslager og sekundærlager) løsningen din krever og hvor mange kort den kan kontrollere per sekund.

Løsning

a) Les i boka.

b) En filtertetthet på 12,5 % gir god treffsikkerhet. Det tilsvarer en byte (8 bit) per nøkkel. Bloomfilteret krever da 10 MB i arbeidslager.

c) Antall kortnummer som må lagres som sperret er 5 % av 300 millioner, det er 15 millioner nummer.
Vi kan velge en filtertetthet på 0,1. Antall falske positive er da < 1 prosent. Hvor mange som prøver å bruke sperrede kort er usikkert, antakelig er de fleste sperrede kort ubrukt. Sperrede kortnummer vil alle gå gjennom filteret, det antas å utgjøre vesentlig mindre enn 5 % av transaksjonene.

Alle sperrede kortnummer må være lagret i en struktur som gir hurtige oppslag. Vi har eksakt match og kan bruke randomisert fil med progressivt overløp. Totalvolumet for sperrede kortnumre utgjør 16*15 = 240 MB når hvert siffer lagres som en byte. Dette er egentlig ikke mer enn det er plass for i et arbeidslager. Imidlertid kan det være aktuelt å addere informasjon til hvert kortnummer, for eksempel sperredato og årsak. 100 byte per kortnummer kan være en aktuell poststørrelse.
Blokkfaktor blir da 40, 80 eller 160 for blokker på hhv. 4 KB, 8 KB og 16 KB.
Med fyllingsgrad 0,8 eller mindre gir dette nært en aksess per oppslag.
Diskplass beregnes til $V = \frac{15000000 \times 100}{0{,}8} = 1{,}875\ GB$.
Vi velger flashdisk som lagringsmedium og antar at en aksess krever 0,1 ms.
Vi ignorerer trafikken fra sperrede kort og antar at 1 % av transaksjonene slipper gjennom Bloomfilteret. Den randomiserte filen klarer da 10000 aksesser per sekund. Det gir en øvre grense for antall transaksjoner per sekund på 1000000.

Med et Bloomfilter med tetthet 0,1 er det optimale antall uavhengige hashfunksjoner 7. En taper lite ved å brukte bare 5 uavhengige hashfunksjoner. En kan anta at CPU-arbeidet med å beregne hashfunksjoner blir den mest begrensende faktor. Her kan multikjerneprosessorer utnyttes. Med 1 million transaksjoner per sekund skal 5 millioner forskjellige bit testes. Det tilsvarer 200 ns per bit.

Oppgave 6, foreningsalgoritmer (20 %)

Tabelldata	Resultat	Basetabeller		
	R	A	B	WS
Nøkkellengde i byte	8	8	8	
Postlengde i byte	200	600	300	
Antall poster	250 000	300 000	1 200 000	
Tabellvolum i MB	50	180	360	10
Resultatdel, postlengde i bytes		100	100	
Nettofil med data, MB		30	120	
Bare nøkkeldata, MB		2,4	9,6	

Finn den beste algoritmen for å gjøre operasjonen **R=A* B** (forening av tabellene **A** og **B**).
Resultatpostene i **R** henter 100 byte fra hver av operandene. Tilgjengelig arbeidslager WS er 10 MB.
Beregn samlet transportvolum for de algoritmene du prøver.

Kommentar: Tabellen over inneholder også noen avledete størrelser, ikke oppgitt i oppgavesettet.

Mål

Vi kan ikke gjøre det bedre enn å lese begge operandene en gang og skrive R: 180+360+50=590 MB.

Varianter av gjentatte gjennomløp

Kommentarer: Fra A skal 100 byte fra hver post inn i resultatet og A sitt nettovolum blir 30 M. Det krever 3 oppfyllinger av WS i gjentatte gjennomløp. I grunnversjonen leses B 3 ganger.
I "netto B"-versjonen lagres en temporær B som bare inneholder data fra hver B-post som kan inngå i resultatet. Temporær B må skrives 1 gang og leses 2 ganger, og som en ser er det meget lønnsomt.

Nested Loop, basic	Volumer	Nettovolum	WS	n
Les A	180	30	10	3
Les B	360	120		
Skriv R	50			
Reread B (n-1) ganger	720			
Total	1310			

Nested Loop, netto B	Volumer	Nettovolum	WS	n
Les A	180	30	10	3
Les B	360	120		
Skriv R	50			
Skriv og reread netto B	360			
Total	950			

Nested Loop, filter A	Volumer	Nettovolum	WS	n
Les A	180	30	10	3
Les B	360	120		
Skriv R	50			
Skriv netto (n-1)/nA, lag filter	20			
Skriv netto B gjennom filter	15			
Fyll WS med netto B	15			
Les netto A	40			
Total	680			

Størst forbedring ved midlertidig lagring av B. Filter vil ytterligere forbedre resultatet.
Kommentarer til den siste varianten "filter A": Når WS går full under lesing av A fortsetter lesing av A, men i stedet for å stille opp poster i WS skrives postene til en temporærfil og en lager et filter A. Når B leses gjøres to aksjoner: B-posten sjekkes mot WS for å lage resultatposter. B-poster som får match i A-filteret skrives (netto) til en temporær B-fil. Dette er poster som kan "matche" A-poster som ikke er i WS nå.
Størrelsen på den temporære B-filen beregnes til 15 MB. Temporær A er 20 MB. Maksimalstørrelsen på *tB* beregnes ut fra antall resultatposter i R. Det blir ikke flere matchende B-poster enn det finnes poster i resultatet. Marginer baseres på at noen B-poster inngår i to eller flere resultater. I de siste gjennomløpene har A og B byttet roller. B er nå minst og vi fyller WS med poster fra temporær B, i to gjennomløp.

Partisjonering

Partisjonering, u filter	Volumer	Nettovolum	WS	n
Les A	180	30	10	3
Les B	360	120		
Skriv R	50			
Skriv netto A	30			
Skriv netto B	120			
Les netto A	30			
Les netto B	120			
Total	890			

Partisjonering, m filter	Volumer	Nettovolum	WS	n
Les A	180	30	10	3
Les B	360	120		
Skriv R	50			
Skriv netto A, lag filter	30			
Skriv netto B, gjennom filter	30			
Les netto A	30			
Les netto B	30			
Total	710			

De to siste tabellene gjelder for partisjonering. Meget robuste algoritmer, og som er nesten like gode som gjentatte gjennomløp med filter. Når A splittes settes opp et Bloomfilter basert på nøklene i A. Når så B splittes, skrives bare poster som ”matcher” i Bloomfilteret til splittfilene.

Vurderer ikke sortering og delvis partisjonering.

Kjb/4.8.2014