NTNU
Det skapende universitet

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i
TDT4225 Lagring og behandling av store datamengder

Faglig kontakt under eksamen: Kjell Bratbergsengen

Tlf.: 90617 185 / 7359 3439

Eksamensdato: Fredag 23. mai 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 16. juni 2013.

Målform/språk: Bokmål

Antall sider: 3

Antall sider vedlegg: 0

Kontrollert av:

Svein-Olaf Hvasshovd

_____

Dato Sign

TDT4225 Lagring og behandling av store datamengder
Fredag 23. mai 2013, kl. 0900-1300

Oppgave 1, Lagringsmedier magnetisk disk og flash minne (10 %)

a) Sammenlign magnetisk disk og flashminne med hensyn til deres sterke og svake sider.

Løsning
Les i boka.

Oppgave 2, RAID 5 (25 %)

	Disk nummer				
Stripe	0	1	2	3	4
0		0	1	2	3
1	4		5	6	7
2	8	9		10	11
3	12	13	14		15
4	16	17	18	19	
5		20	21	22	23
6	24		25	26	27

Figuren viser et RAID 5-system etter at disk 3 er ute av drift. Hver raid-gruppe består av 4 datablokker og en paritetsblokk. Alle diskene er av samme type. Hver disk har en kapasitet på 500 GB, antall rotasjoner per sekund (ω) er 250 og hvert spor lagrer 800 KB.

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

Anta at alle 5 diskene i systemet er fullt operative. Systemets blokkstørrelse er 64 KB. Anta at det kreves to rotasjoner for hver I/O-operasjon. Med *blokk* i spørsmålene b) til e) menes en adressert logisk blokk sett fra en brukers perspektiv. Operasjonene skjer på vilkårlig adresse.

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) Anta at disk 3 blir utilgjengelig.
Hvor mange blokker kan nå leses per sekund?

f) En ny disk settes inn for disk 3. Hvor lang tid tar det å reetablere disk 3 når disken skrives med en blokkstørrelse på 64 KB.

g) Kan reetableringstiden reduseres? I så fall på hvilken måte? Hvilke begrensende faktorer kan spille inn ?

Løsning
a) Maksimal lese- og skrivehastighet er $\omega Vs = 250 \times 0{,}800$ MB/s = 200 MB/s

Anta at det trenges gjennomsnittlig to rotasjoner for å lese eller skrive en blokk på en vilkårlig adresse.

b) Totalt rotasjonsbudsjett er 5*250 rotasjoner per sekund = 1250 rotasjoner, antall leseoperasjoner per sekund blir 625.

c) Antall datablokker som skrives: Paritetsblokk må både leses og skrives, det samme må datablokka. 4 rotasjoner på hver oppdatering, totalt 8 rotasjoner, det gir 1250/8=156,25 oppdateringer per sekund. Paritetsblokken oppdateres ved at først "fjernes" pariteten i datablokk som skyldes gammelt innhold. Så oppdateres datablokken med nytt innhold. Så må paritetsblokken "tilføre" pariteten som skyldes det nye innholdet. XOR-operasjonen brukes i begge tilfeller. Denne omstendelige prosedyren gjør at vi konservativt regner 4 rotasjoner. Tilbakeskriving av en enkeltblokk etter lesing bør kunne gå uten ny posisjonering og da bør lesing- umiddelbar oppdatering – tilbakeskriving kunne gå på 3 rotasjoner.

d) Det samme som i c.

e) Disk tre blir utilgjengelig. Normalen er 2 rotasjoner per lesing av en blokk. For hver 5. blokk som leses kreves nå lesing av 4 andre blokker, det gir 8 rotasjoner. Normalt uten feil: 5*2=10 rotasjoner for å lese 5 blokker. Nå med feil: 4*2+8=16 rotasjoner for å lese 5 blokker. Antall lesinger per sekund av en gruppe på 5 blokker: 1250/16=78,125 grupper per sekund. Hver gruppe består av 5 lesinger: Antall lesinger per sekund: 78,125 * 5 = 390,6 – en reduksjon fra 625.

f) Alle friske disker leses i parallell. Vi oppdaterer blokkene etter stigende adresse slik at all lesing og skriving går sekvensielt. Når 4 blokker innen samme paritetsgruppe er i arbeidslageret kan vi beregne blokken som er "borte" ved å "XORe" blokkene i paritetsgruppen. Lesing av de 4 diskene, xoring og skriving av den nye disken kan gå parallelt. Det er god plass i arbeidslageret slik at leseprosessene ligger foran XOR-prosessen som igjen ligger litt foran skriveprosessen. Sekvensiell skriving er begrensningen, en rotasjon per blokk som skrives. Det gir t= 500 000 000/64/250 = 31 250 sekunder, 8:40:50 timer.

g) Kan reetableringstiden reduseres? Ja, ved å lese og skrive diskene med større blokker, dvs. multipler av den brukte blokkstørrelsen på 64 KB. Det bør ikke være noe problem, vi trenger sammenfallende fysiske adresser på 5 disker. Dette er fysiske adresser etter "bad spots" er "mappet" ut.
Kan begrenses av tid for å generere pariteter.

Oppgave 3, Blokkorganisering (15 %)

Du skal lage et filsystem for lagring av poster som kan ha ulik lengde. Postlengden kan variere fra 12 byte til 30 000 byte. Blokkstørrelsen er 64 KB. Alle poster starter med en nøkkel på 8 byte. Vis hvordan du vil organisere dataene i blokken og begrunn dine valg.

Løsning

Se også læreboka. Det vil alltid være plass for minst to poster i en blokk, da kan en bruke blokksplitting hvis en trenger mer plass. Hver blokk inneholder en pekertabell som viser hvor i blokken hver post begynner.

Oppgave 4, Sortering (20 %)

Du skal sortere en fil med 100 millioner poster, postlengde er100 byte, derav utgjør nøkkel 12 byte. Til rådighet har du 100 MB arbeidslager. CPU-hastigheten er slik at du kan regne med at gjennomsnittstiden for å sammenligne to poster er ett mikrosekund.

a) Alle filer (innfil, mellomlagerfiler og resultatfil) ligger på samme *magnetiske disk* (HDD). Disken har følgende data: Antall rotasjoner per sekund er 250, sporstørrelse er 1 MB og fast blokkstørrelse er 64 KB.
Hvor lang tid tar sorteringen?

b) Alle filer (innfil, mellomlagerfiler og resultatfil) ligger på samme *flashdisk* (SSD). Disken har følgende data: Fast blokkstørrelse er 4 KB, lesing og skriving tar begge 0,1 ms per blokk.
Hvor lang tid tar sorteringen?

Løsning

Størrelser, gitte og beregnede:

ω	250	
Vs	1 000	KB
b_{HDD}	64	KB
M	100	MB
b_{SSD}	4	KB
n	100 000 000	
l	100	
V	10 000	MB
V	10 000 000	KB
N	100	delfiler
m	1 000 000	poster i tapertreet
h	20	høyde i tapertre.
Tt	2 000	tid i tapertre
T_{iohdd}	2 660	Totaltid i for I/O, disk
T_{iossd}	1 000	Totaltid I/O for SSD

Forskjellige parametre og avledete størrelser er beregnet og satt opp i tabellen over. I/O-tiden er i begge tilfeller beregnet for en passering under initiell sortering og en passering under fletting.
Passeringstid (lesing og skriving av hele volumet) for magnetisk disk:
$T_{HDD} = \frac{2V}{\omega}\left(\frac{1}{b_{HDD}} + \frac{1}{V_s}\right) = \frac{2\times10^{10}}{250}\left(\frac{1}{64\times10^3} + \frac{1}{10^6}\right)$=1330 sekunder.
Initiell sortering krever 1330 sekunder for lesing og skriving (for HDD), mens trekking gjennom tapertreet krever $\tau hn = 10^{-6} \times 20 \times 10^8 = 2000\ sekunder$. I tillegg trenger vi en sammenligning per post for å bestemme hvilken delfil den nyinnleste posten skal legges inn i. CPU-tiden for initiell sortering er derfor 2100 sekunder.

a) Antall delfiler blir: N=V/(2M)=10 000 MB/ (2*100 MB)=50. Alle disse flettes samtidig. Høyden i tapertreet for fletting er $\lceil \log_2 50 \rceil = 6$. CPU-tiden for fletting er derfor $\tau h_f n = 10^{-6} \times 6 \times 10^8 = 600\ sekunder$.
Totaltid for HDD er max(2100,1330)+max(600,1330)=3430 sekunder.

b) Passeringstid når mediet er SSD blir:
$T_{SSD} = \frac{2Vt(b)}{b} = \frac{2 \times 10^{10} \times 0{,}0001}{4096} \approx 500\ sekunder$. For SSD blir CPU-tiden begrensende i begge faser.
Totaltiden blir max(2100, 500)+max(600,500) = 2700 sekunder.

CPU-tidene for initiell sortering og fletting er de samme i begge tilfeller.

Oppgave 5, Relasjonsalgebra på parallelle maskiner (10 %)

a) Forklar hvordan operasjonene **aggregering** kan utføres på en parallell maskin.

b) Forklar hvordan operasjonen **divisjon** kan utføres på en parallell maskin.

Løsning

a) **Aggregering:**
Gjør lokal aggregering i hver node. Send deretter de aggregerte postene til en møteplassnode. Møteplass for hver gruppe bestemmes av hashformel anvendt på grupperingsattributtet. Aggregeringen fullføres på møteplassnoden.

b**) Divisjon R=A/B:**
To varianter.
Alternativ 1:
Møteplass kan bestemmes av grupperingsattributt. Da må alle poster i andre operand *kringkastes* til alle noder. Deretter sendes postene i førsteoperand til møteplassnode bestemt av hashformel anvendt på grupperingsattributt – første attributt i første operand. Alle A-poster må relokaliseres.

Alternativ 2:
Først sendes alle B-poster til møteplassnode bestemt av hashformel anvedt på sitt eneste attributt. Deretter sendes alle A-poster til møteplassnode basert på hashverdi av settattributten (2. attributt i A). Lokale komplette sett etableres. Til slutt sendes grupperingsattributt for lokale *komplette* sett til møteplassnode bestemt av grupperingsattributtets hashverdi.
Alle A-poster må relokaliseres, men en slipper kringkasting av B - bare relokering av B. I tillegg må en relokere grupperingsattributter for lokale komplette sett. Det siste antas å være et lite volum. Operasjon fullføres på møteplassnode.

Oppgave 6, Relasjonsalgebra (20 %)

Tabelldata:	Resultat	Basetabeller	
	R	A	B
Nøkkellengde i byte	8	8	8
Postlengde i byte	200	600	300
Antall poster	150 000	500 000	1 200 000

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 15 MB. Beregn samlet transportvolum for de algoritmene du prøver.

Løsning

Tabelldata	Resultat	Basetabeller		
	R	A	B	WS
Nøkkellengde i byte	8	8	8	
Postlengde i byte	200	600	300	
Antall poster	150 000	500 000	1 200 000	
Tabellvolum i MB	30	300	360	15
Resultatdel, postlengde i bytes		100	100	
Nettofil med data, MB		50	120	
Bare nøkkeldata, MB		4	9,6	

Mål

Vi kan ikke gjøre det bedre enn å lese begge operandene en gang og skrive R: 30+300+360=690 MB.

Varianter av gjentatte gjennomløp

Kommentarer: Fra A skal 100 byte fra hver post inn i resultatet og A sitt nettovolum blir 50 M. Det krever 4 oppfyllinger av WS i gjentatte gjennomløp. I basic-versjonen leses B 4 ganger.
I "netto B"-versjonen lagres en temporær B som bare inneholder data fra B som kan inngå i resultatet. Den må skrives 1 gang og leses 3 ganger, men som en ser er det meget lønnsomt.

Nested Loop, basic	Volumer	Nettovolum	WS	n
Les A	300	50	15	4
Les B	360	120		
Skriv R	30			
Reread B (original)	1080			
Total	**1770**			

Nested Loop, netto B	Volumer	Nettovolum	WS	n
Les A	300	50	15	4
Les B	360	120		

Skriv R	30
Skriv og Reread B (netto B)	480
Total	**1170**

Nested Loop, filter A	Volumer	Nettovolum	WS	n
Les A	300	50	15	4
Les B	360	120		
Skriv R	30			
Skriv netto (n-1)/(n*A), lag filter	37,5			
Skriv netto B gjennom filter	15			
Fyll WS med netto B	15			
Les netto A	37,5			
Total	**795**			

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, som det er plass til i WS. Tar utgangspunkt i antall resultatposter. Det blir ikke flere matchende B-poster enn det finnes poster i resultatet.
Marginer baseres på at noen B-poster inngår i mange resultater. Den siste gjennomgangen gjøres ved at A og B bytter roller. B er nå minst og vi fyller WS med poster fra temporær B.

Partisjonering

Partisjonering, uten filter	Volumer	Nettovolum	WS	
Les A	300	50	15	
Les B	360	120		
Skriv R	30			
Skriv netto A	50			
Skriv netto B	120			
Les netto A	50			
Les netto B	120			
Total	**1030**			

Partisjonering, med filter	Volumer	Nettovolum	WS	
Les A	300	50	15	
Les B	360	120		

Skriv R	30
Skriv netto A, lag filter	50
Skriv netto B, gjennom filter	15
Les netto A	50
Les netto B	15
Total	**820**

De to siste tabellene gjelder for partisjonering. Meget robuste algoritmer, og som er nesten like gode som nested loop med filter.

Ukonvensjonell metode

Vi ser at vi har plass for alle nøklene til A (og også B) i arbeidslager.
Les A, nøkkelen lagres i WS og strippet A lagres partisjonert i filene TA.
Les B, poster med tilsalg i WS lagres strippet i filene TB.
Les og foren partisjonene i TA og TB.

Ukonvensjonell metode	Volumer
Les A	300
Les B	360
Skriv R	30
Skriv netto A til TA, og nøkkel i WS	50
Skriv netto B som matcher	15
Les netto A	50
Les netto B	15
Total	820

Vi får samme volum som partisjonering med filter.

Vi bruker bare 4 MB av WS til A's nøkler. Når vi leser B har vi 11 MB ledig i WS, der kan vi lagre matchende B-poster. Bare de det ikke er plass til må skrives ut, det er i verste fall 4 MB.
I beste fall er alle matchende B-poster i WS. Hvis vi er heldig og en B-post inngår i flere R-poster vil antall B-poster som skal tas vare på være mindre enn B-volumet i R, nemlig 15 MB.
Da vil totalvolumet for den metoden komme på 820-30 MB som er 790 MB.
Det er det beste vi har kommet til.

Vurderer ikke sortering og delvis partisjonering.

Kjb/30.4.2014