16.6.2017.

1.e1)fifo e2)sekvencijalno e3)uslovna

E2 je samo sekvencijalno jer na isti nacin vide oba procesa read x, a ja bih stavila da je uslovno e3 jer je takav primer dat na prezentacijama hahha nasla sam i na netu sigurbno je kao sto sam rekla da znate ako hocete slike poslacu vam

To je tacno , ali kazem ti sta generalno vazi za konzistentnost skladista

Ako je skladiste striktno konzistentno, onda je i sekvencijalno i uslovno i fifo (to vazi za sve)

A meni je meina rekla da gledam sekvencihjalnu kao najjaci oblik konzistencije jer ona vazi u distribuiranim i onda ovo sto je ko god napisao od vas dvoje vazi za nju da znate

2. Vrste replika: permanentne, incirane od strane servera, inicirane od strane klijenta.

Moze se slati obavestenje o obavljenom azuriraju (da bi se invalidirale zastarele kopije na klijentskim masinama), podaci koji se azuriraju, operacije koju su izazvale azuriranje.

Da li se misli samo na to, ili moze da se stavi i dobijanje permisija za azuriranje sto postoji kod protokola zasnovanih na kvorumu?

Mislim da je samo ovo sto ste napisali

3. Nr - broj read replika, Nw - broj write replika, N- ukupan br replika

Mora vaziti:

Nr+Nw>N Nw>N/2

Prvi uslov garantuje da nece biti read/write konflikta(tj na srpskom da ce proces koji trazi read kvorum sigurno dobiti najskoriju verziju podatka koji cita), a drugi da nece biti w/w konflikta

a) Nw=6 Nr=5
b) Nw=7 Nr=5
c) Nw=10 Nr=1
d) Moze i vise od ova 3…

4. Nije moguce, mrzi me da crtam

5. Node0 = k4
Node1 = k1
Node2 = /
Node4 = k2
Node6 = k3

11.9.2017

1.

a) Broj procesa koji ucestvuje u komunikaciji
b) Va=Vb oznacavaju isti dogadjaj? Va>Vb, a se desio posle b, uslovljeni su

Ne moraju da budu uslovljeni, mofu da budu i konkurentni ali da je a svakako pre b, mm?

Mislim da ne mogu da budu konkurentni ako je sve vece, eventualno su uslovljeni mozda

c) Va se desilo pre Vb
d) Va i Vb su konkurentni dogadjaji

2.a) jeste zato sto upisi nisu uslovljeni i mogu se vidi u razlicitom redosledu i drugim procesima

b) dodati R(x)2 iza W(x)2

3. ABFT - detektuje i koriguje jednostruke greske

Poslednja vrsta i kolona su kontrolne sume za ostatak matrice(sto ne ukljucuje zadnju vrstu i kolonu)

a) nastupila je greska

b) 2. Kolona, 3. Red.
c) 3->2
4. U ovom protokolu write operacije izvrsavaju se iskljucivo na primarnoj kopiji - NA PRIMARU dok se citanje moze vrsiti sa kopija koje se nalaze u blizini klijenta i na udaljenom serveru koji sadrzi taj podatak. Svaki podatak u skladistu ima pridruzen primar(ljudi sta je primar?) koji vodi racuna o koordinisanom upisu u taj podatak.
Imamo dve varijante ovog protokola: remote write i local write. Kod remote-write protokola se sve read i write operacije izvrsavaju na udaljenom(primarnom) serveru.Postoje i modifikacije koje dozvoljavaju citanje iz lokalne kopije. **Kod local-write protkola replike cesto menjaju koja je od njih primarna u azivsnosti od toga koja treba da se azurira(vrsenje write operacije)**. ---**OVO TI NIJE TACNO. Local write protoli dozvoljavaju da primar privermeno promeni poziciju i da se priblizi klijentu koji modifikuje promenljivu.**
Najjaci vid konzistencije koji se moze postici je sekvencijalna konzistencija.
Primar ti je skladiste koje jedino sadrzi taj podatak, mada ona ga koristi i u kontekstu jednog jedinstevog podatka
5. Koriste se backward recovery i forward recovery.
BR vraca sistem u zadnje sacuvano korektno stanje tj. Checkpoint.
FR stavlja sistem u novo korektno stanje (npr Erasure coding (n,k))
Nedostaci:
BR - pravljenje checkpointova je dosta skupa operacija jer treba sacuvati sve podatke u sistemu, ne isplati se ako se greske retko desavaju. Dodatna otezavajuca cinjenica je sto treba da se nadje konzistentni presek izmedju procesa. Ukoliko je jedan proces u zadnjem checkopointu zabelezio slanje i primanje poruke a drugi samo primanje bez slanja(odgovora) moze docii do domino efekta gde se sistem vraca na pocetno stanje i gubi sve podatke.
FR - treba misliti na sve moguce greske ali mehanizam zna sta treba uraditi da se vrati u ispravno stanje.
6. Prednost centralnih servera je to sto server vodi racuna o aktivnim korisnicima. Da bi se klijent prikljucio sistemu treba samo da kontaktira server i odmah ima pristup svim podacima koji su mu potrebni za lociranje resursa.
Losa strana centralnih servera je sto mogu postati usko grlo i to sto pri njihovom otkazu ceo sistem otkazuje, tj ima slabu otpornost na greske.

Prednost potpuno distribuiranih modela je sto nema uskog grla u sistemu i sto je sistem otporan na greske.

Losa strana je sto cvorovi moraju da vode racuna o tome ko je aktivan u mrezi, da vode racuna ko cuva koje podatke i da salju upite o lociranju drugih cvorova kako bi dosli do resursa sto bi moglo dovesti do zagusenja mreze. Takodje ukoliko se neko skine sa mreze ili udje u mrezu cvorovi moraju update-ovati svoje tabele o cvorovima.

5.5.2018. kolokvijum

a) Da l se misli na klasicne il gloablne? Hmm hmm
b) a) i b) mislim da je prelako(?)
c) Samo i

d) Ne moze da se odredi da li su uslovljene, samo se zna da se a desilo pre b, tj T(a)<T(b), obrnuto ne vazi… Zbog toga smo uveli vektorske casovnike jer kod njih moze to da se zakljuci.

1.2.2018.

1. Pod distribuiranim sistemom podrazumeva se skup medjusobno povezanih nezavisnih racunara koji klijentu deluje kao jedna masina.
Prednosti: brzina, pouzdanost,komunikacija
Mane: Veca kompleksnost, umrezavanje, problem konzistentnosti i sinhronizacije(ili to spada u vecu kompleksnost), bezbednosni problemi.

2. Kod sinhronih komunikacija proces koji je poslao poruku ne nastavlja s radom sve dok ne primi poruku od drugog procesa. Kod asinhronih komunikacija proces nastavlja s radom nakon slanja poruke i ne ceka na odgovor. Asinhroni pristup se koristi u distribuiranim sistemima jer pruza transparentnost.

To je sve? TO je to, sta drugo vise...

3.

a) Kristijanov algoritam T = Tutc + (T1-T0)/2
NTP (RPC?) protokol : t' = t3 + teta, teta= ((t1-t0)-(t3-t2))/2
Berkeley: prosek svega…
NTP- serveri vremena organizovani u stratume ili ti nivoe, na nivou nula se nalze serveri sa atomskim casovnicima- glume izvor tacnog vremena
b) Pomocu vremenskih markica kao sto su Lamportove markcie i vektorski casovnici gde se nakon svakog dogadjaja inkrementira markica u sistemu i pomocu kojih se odredjuje sekvenca dogadja… ?

4.

Greske u odnosu na trajanje mogu biti:
a) Prolazne(timeout za cekanje na odgovor neke poruke)
b) Periodicne (Kada zeza kontakt kod preznog kabla)
c) stalne (bilo sta sto ti padne na pamet)

Kod mirnih greski masina prestane sa radom i ne izbacuje nikakvo obavestenje o uzroku greske dok kod vizantijskih greski cvor generise pogresan rezultat i nstavlja sa radom.. Kod mirnih greski potrebno je da jedan cvor daje tacan rezultat da bi sistem funkcionisao ispravno dok kod vizantijskih gresaka vise od ⅔ cvorova mora davati korektan rezultat kako bi sistem korektno funkcionisao.
Teze je detektovati vizantijske greske, jer sistem moze nastaviti sa radom i ako greska nastupi.(ne zna se kada je komponenta otkazala jer izlaz moze da bude i korektan i ne)

5. Rezultat *jeste* korektan sa stanovista sekvencijalne konzistencije zato sto je moguce da su se dogadjaji u procesu P1 i P3 desili pre bilo kog dogadjaja u procesu P2.
Npr x=1; z=1; print(y,z) print(x,y) y=1; print(x,z);
Generise ti P1 01, P3 10 P2 11 hahahahaha dobro

6. Azuriranje replika moze da se izvede na 3 nacina:

1) Slanjem azuriranog podatka svima - za mali broj modifikacija u sistemu
2) Slanjem operacije kojom se doslo do nove verzije podatka - mala kolicina saobracaja kroz mrezu ali zahteva vise procesorskog vremena
3) I slanjem obavestenja da ostale replike nisu vise validne(ovaj deo sam morala da pogledam jer je retardirano) - kada je broj azuriranja veliki

28.4.2017 .

1. Pod skalabilnoscu se podrazumeva da sistem ne gubi na kvalitetu performansa pri sirenju sistema kroz neku od 3 dimenzija skalabilnosti. Dimenzije skabilnosti su: broj korisnika i resursa, geografska udaljenost klijenta od replike i administrativna skalabilnost (lako upravljanje)

   Tehnike skaliranja su: skrivanje komunikacionog kasnjenja (koriscenje asinhronih funkcija), distribucija podataka i zadataka (podela posla), replikacija podataka (poboljsanje vremena pristupa u slucaju velike geografske udaljenosti)
2. Server se moze locirati staticki i dinamicki. Kod statickog povezivanja klijent zna adresu servera koji treba da bude kontaktiran. Adresa hosta se nalazi u stub-u i kada zove proceduru odmah se obraca serveru. Problem se javlja ukoliko server promeni lokaciju.

   Kod dinamickog povezivanja klijent se obraca centralizovanoj bazi podataka (ili direktorijumskom serveru?) koja potom daje adresu servera koji pruza datu uslugu. Potom se klijent obraca serveru, tacnije njegovom endpointu i pita za port date usluge. Nakon toga klijent se povezuje na dati port kada biva slobodan.

3. Typedef char string[128];
   program IZBORI_PROGRAM{
   version IZBORI_VERSION{
   Void GLASANJE(string, int)=1;
   Int REZULTAT(string)=2;
   }=1;
   }=0x12345678;

4. A.
   P1 = P4,P2,P3
   P2 = P3
   P3 = nista
   P4 = P2, P3
   Ako je primaoc u KS - smesta sve zahteve u red cekanja
   AKo nije ali i ona salje zahtev za KS:
   Poredi vremena prijema i slanja (pobedjuje ranije izdat zahtev)- i ako je izgubio salje potvrdu za KS, ako je pobedio smesta zahtev u red cekanja

B.

P1 = / --svima u redu cekanja salje OK kad izadje iz KS
P2 = P3

P3= /
P4 = P2,P3

5. Isto ko na prezentacijama

<u>26.06.2017</u>

1. Uloga klijentskog staba ja da vrsi pakovanje(tj “marshaling”) vrednosti sa steka (argumenata funkcije)i sprema je za slanje udaljenoj proceduri i potomprosledjuje poruku operativnom sistemu za slanje ka serveru.
   Nakon primanja rezultata od operativnog sistema, vrsi raspakivanje (ili “unmarshaling”) i vraca rezultat klijentu.
   E sad.. Kad si vec poeco...
2. I ovo me mrzi da raidm ovo znamo
3. a) nije moguce zato sto p3 vidi da se prvo izvrilo x=1 pa x=3, a p4 obrnuto. Kod sekvencijalne konzistencije svi procesi moraju videti isti redosled dogadjaja.
   b) moguce je zato sto dogadjaji nisu medjusobno uslovljeni. U p1 i u p2 nema citanja promenljive x tako da ovde vazi kauzalna konzistencija.
   TAcno jer se ne ne vrsi citanje uslovljenih upisa
4. C2 pravi konzistentni presek zato sto nemamo nijedan slucaj gde se poruka poslala a da se u drugoj procesu nije primila poruka(ili da se primio odgovor a da u drugom procesu nije poslata poruka). Dok u C1 ne pravi konzistentni presek jer e1(4) salje poruku a poruka se prima u e2(3).

5. P1 = (1 = P3, 2 = P3, 4= P6, 8=P18, 16=P18)
   P3 = (1= P6, 2 = P6, 4= P18, 8=P18, 16 = P24)
   P6 = (1=P18, 2=P18, 4 = P18, 8 = P18, 16 = P24)
   P18 = (1= P24, 2= P24, 4 = P24, 8 = P1, 16 = P3)
   P24 = (1 = P1, 2 = P1, 4= P1, 8= P1, 16 =P18)
   Moze li neki hack za ovo gore:D
   Evo ovako dakle za P1 prvo
   1+1 = 2, proces 2 je mrtav, tako da je najblizi br veci od 2 3, sto znaci proces 3.
   1+2 = 3, P3,
   1+4 = 5 , P6 i tako sve redom
   Al onda dodjes npr do P18 i za 8 npr uradis ovako
   18 + 8 = 26, najbilizi sledeci je P1.
   I za 16 uradis 16+18 = 34 mod 32 = 2, i odatle je proces P3
   Isti postupak za P24 u zadnjoj liniji -nemoj da brises ovo

<u>29.9.2017.</u>

2. b) 16. Znamo formulu…

3.moguce je samo y=0,x=1,z=0 ili 1? :/ mislim da je 1 posto se prvo izvrsava P1?

Za ovakve male primere, ima mnogo dobar hack, napises sve moguce kombinacije printova, 000,001...111 i onda samo gledas sta moze sta ne. Tako, nikad neces da se zajebes i da ti trazi ispravne i neispravne.

P1---A =1 ----x=A-------y=A--------z=y-----
P2-------------------------print y-------print x---- print z----

4. Uloga NameNoda je da pamti sve promene koje se desavaju u sistemu (ili u recku?). Da cuva metapodatke(podaci po podacima) tj. Kako su podaci podeljeni u blokove, koji DataNodovi cuvaju koje blokove, faktor replikacije blokova. Takodje pristupa podacima (preko DN), u direktnoj je kominikaciji sa DN. Koristi heartbeat signale da provere da li je DN ziv. Otpornost na greske u HDFS-u se postize uz pomoc SNN (sekundarnih name nodova). SNN preuzima checkpoint fajlove od NN. Primenjuje promene iz CP i koristi ga NN u slucaju da je potreban oporavak od gresaka.

11.10.2017.

1. Ovo je lako
2. Samo pod a
3. Samo pod d
4. Sta nemoguce: 010 nije moguce da je z=0 ako y ima vrednost 1, a x=1, 011 isto, 101 x i z imaju vrednost 1, ne moze y=0, 110
   Za x=0, y=1, z=0,

I ovo ne moze da se desi x=0, y=1, z=1

5. Problem dve armije - ispravni procesi, nesavrseni kanali
   Stvara problem visetrukih potvrda -> usaglasavanje procesa nije moguce zbog neispravnih komunikacionih kanala i ne postoji protokol koji moze da resi ovaj problem

10.9.2018.

1. a) maybe semantika - To znaci da ce da se desi jednom ili nijednom, ali nemas nikakvu garanciju
   b) at-least-once - ova semantika trazi da se udaljena procedura izvrsi makar jednom. Ova semantika vazi za idempotentne procedure (procedure koje ne menjaju resurs ako se izvrse 2x, recimo read ili abs(x))
   c) at-most-once - -||- najvise jednom (je l ovo isto sto i maybe onda?) ne jer ce to sigurno jednom da se desi samo jednom a least bar jednom napisi ga samo jednom i procedure nisu idempotentne
   Al at most-once se mozda izvrsi, ne garantuje, zato kaze najvise jednom… ne garantuje
   Okej

2. a.) 1. Pravilo : Svi elementi casovnika se inicijalizuju na nulu. Vi[i]=0 i=0..n-1
   2. Pravilo : Pre svakog dogadjaju u procesu Pi, proces inkrementira svoje polje za 1 Vi[i]= Vi[i] +1

3.Pravilo: kada proces Pj primi poruku od procesa Pi ooredi lokalni casovnik sa promljenim element po element i postavlja element u svom lokalnom vektoru na vecu od dve vrednosti gde je Vj[k] lokalna vrednost, Vi[k] primljena vrednost - **Vj[k]=max(Vj[k],Vi[k]) k=0..n-1**

b)Pod c, da l je moguce da se emina smilovala u nekom roku xd

3. PONOVLJENO
4. PONOVLJENO nmg da verujem
5. PONOVLJENO

2.7.2018.

1.Middleware je sloj izmedju distribuirane aplikacije i lokalnog OS. Middleware se brine o komunikaciji sa udaljenim procesima, sinhronizaciji, bezbednosti, o konzistenciji i tolerantnosti na greske. Zahvaljujuci ovom sloju programer je oslobodjen brige o komunikaciji procesa… U middleware spadaju protokoli opste namene (kao sto je autentikacija i autorizacija npr) koji ne pripadaju transportnim protokolima. Takodje ima ulogu da sakrije heterogenost platforme. (mozda moze da se kaze i da je komunikacija skrivena iza RPC-a, RMI-a i MPI-a)

2. Kod mirnih gresaka je potrebno da bar jedna komponenta radi ispravno, tako da je ovde maksimalni broj koji moze da otkaze 4?
Sto se tice vizantijskog tipa tu je potrebno da ostanu u radu vise od ⅔, sto znaci da je maksimalni broj koji moze da otkaze za 5 procesa 2. Kaze u prez "*ako postoji m izdajnika i n<=3*m generala nemoguce je postici konsenzus*".
Ne razumem sta je htela sa ovim pitanjem na dalje

Je l treci uslovna konzistencija? Mislim da je sekvencijalna, namerno je ruzno raspodelila ove labele da te zbuni. A jeste moze i sekvencijalna jer svi vide upise na isti nacin Ne znam je l' postoji nacin da se to nauci da te ne predje

4. PREZENTACIJA

5. U unix semantici postoji apsoulutno vremensko uredjenje operacija. Svaka read operacija vidi efekte prethodnih operacija. Ova semtantika se lako postize ako fajlom upravlja server i ako klijenti ne kesiraju podatke. Kod semantike sesije klijent otvara fajl i zapocinje sesiju. Sesija se zavrsava kada klijent zatvori fajl i u medjuvremenu ostali procesi (na drugim masinama) ne mogu videti trenutno stanje fajla. Mogu azurirati fajl tek nakon okoncanja sesije.

E sad ovo drugo pitanje nema u prezentaciji al ja bih napisao ovako.
Semantika poboljsava performanse DFS-a zato sto klijent koji je zapoceo sesiju kesira fajl na svojoj masini. Medjutim ovde je slaba konzistentnost jer se fajl na serveru azurira tek kada klijent zatvori fajl (okonca sesiju) tako da za to vreme ostali procesi nece videti najnovija desavanja u fajlu.

Sto se tice te prezentacije nmg da dam svoje misljenje jer ga nemam. I necu ga imati do 3 ujutru otp
E sad mi pade na pamet kad pisemo ono program glupost { version bla { void operacija(string) nece da tupi ako ne stavimo argument ? pa to i ja mislim nije ovo tako savrseno da zna mislim da moramo da stavimo argument

Je l imate 12.10.2018. ?

1. **Isto**
2. **Ajde ovaj da uradimo**
    a. **Pod a je narusena uslovna // pa znaci nije uslovna , ovde ne treba nista da stoji upravo tako**
    b. **Sekvencijalna //slazem se**
    c. **Kauzalna //isto**
    d. **Kauzalna + sekvencijalna**
    e. **kauzalna + sekvencijalna //mislim da nije sekvencijalna jer nece istu vrednost videti procesi p3 i p4, ali ima dva puta r, ne kasni onda mozda p4 samo kasni**
    f. **kauzalna + sekvencijalna + striktna**

3. **a) jeste lokaciono transparentan, kada se ostvari povezivanje sa serverom klijent ne mora da vodi racuna da l je fajl lokalan il udaljen.**
   b) ne znam sta ovo znaci… :/

4.
```
struct operandi {
        int x;
        int[] niz;
};
        program PRG {
        version PRG_VERSION {
        int NADJI(operandi) = 1;
        }=1
}=0x29999999
Klijent:
CLIENT *clt;
char* server;
clt= clnt_create(server,PRG, PRG_VERSION, “udp”);
Operandi op;
Op.x =2;
Op.niz = {1,2,4,5,6,8,7,2,2};
Int* broj = nadji_1(&op,clt);  // mora i klijent, vraca int pointer
```

ako je fajl velicine 519B i faktor replikacije je 3, i velicina bloka je 64B , koliko ima blokovau sistemu. I pod b je bilo nesto kako se vrsi upis ili citanje, nisam siguran.

Sto se tice zadatka za hdfs, 519B (bajtova) podelis sa 64, zaokruzis na veci broj, kolko znam matematiku ispada 9. E tako je. 9x3 (faktor replikacije) imas ukupno 27 blokova.

CHORD OBJASNJENJE

Dakle kao prvo ip adrese i imena fajlova se hesiraju i po tome se oni sortiraju.Dakle ako imas npr max 32 cvora u sistemu onda u onoj tablici imas pokazivace na 1, 2, 4, 8, i 16. Cvor od tog trenutnog cvora. E sad vrednost kljuceva kolko znam treba da se modulu sa 32. E sad uzmemo u obzir da od ta 32 cvora rade samo cvorovi 0 10 i 28. Cvor p moze da cuva kljuceve koje imaju hes vrednost <= njegove hes vrednosti. Sto znaci da u ovoj situaciji cvor 10 moze da cuva kljuceve (1,2,3...10)

Sto se tice ovog znaci cvor p cuva kljuceve za koje vazi k<=p. To je cela sustina

Usaglasavanje u prisustvu gresaka

1) Problem dve armije - ispravni procesi, nesavrseni kanali
   Problem visetrukih potvrda -> usaglasavanje procesa nije moguce i ne postoji protokol koji moze da resi ovaj problem
2) Problem vizantijskih generala - nesavrseni procesi, ispravni kanali
   N generala koji treba da se dogovore. M izdajnickih generala koji sprecavaju konsenzus.(Greske vizantijskog tipa)
   - Svaki od generala daje svoj predlog, izdajice salju razlicitim generalima razlicite poruke
   - Ako postoji m generala izdajnika, u sistemu mora da postoji 3m+1 generala ge su 2m+1 lojalni (⅔ su lojalni) i m+1 runda razmene poruka

   Dekompozicija na n identicnih problema
   - general i n-1 porucnik - jedna grupa u kojoj postoji m porucnika (ili m-1 porucnik i general)koji su izdajice.
   - **Algoritam: lojalni salje svima istu poruku (ili general i porucnik), onda svaki porucnik salje drugim porucnicima poruku od generala. Kada se razmene poruke, primenjuje se vecinsko glasanje(ili se izlasa akcija ili se koristi podrazumevana)**

Mora se postici da svi lojalni porucnici donesu istu odluku i da ako je general lojalan, svaki lojalan porucnik mora da postuje naredjenje

Strategije za oporavak od greske

Povratak u stanje pre greske (backward)
-prethodno stanje definisano checkpointovima, svaki proces povremeno pamti svoje stanje na disku u tackama provere. U DS potrebno je pronaci liniju oporavka(KONZISTENTNI PRESEK- skup checkpointa od kojih moze krenuti oporavak) - ne moze se krenuti od bilo kod checkpionta. Uslovi koje treba da zadovolje checkpointi da bi se od njih mogao nastaviti oporavak: **ako se u skupu tacaka provere C nalazi dogadjaj e i vazi da dodgadja e' -> e tada vazi da i e' se nalazi u C. (e i e' mogu biti u razlicitim procesima)**

Povratak u novo stanje(forward)
-erasure coding