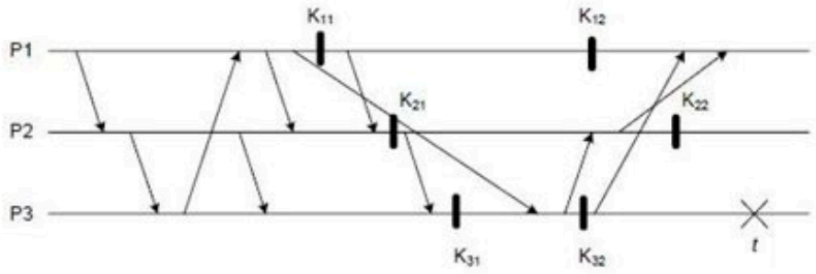
rassus rok

- Sto je iot stvar
- Uloga proxyja u sustavu s 2 posluzitelja
- Otvorenost, transparentnost, skalabilnost
- Tcp naredbe
- iaas, paas, saas
- odgođeni asinkroni rpc
- sto je u kafki topic, partition i jos nesto
- sto su mikroservisi, prednosti, navedi primjer i usporedi s monolitnom arh
- sto je ispad procesa, a sto ispad kanala, od 9 su 3 u biz ispadu, mogu li postic suglasnost
- sto je p2p mreza, kako bi testirao skalabilnost kod p2p mreze
- koliko je procesa potrebno kod
  ispada k procesa (k+1)
  ispada k bizantskih procesa (2k+1)
  suglasnost k bizantskih procesa (3k+1)
- kako replikacija i partitioning utjecu na skalabilnost
- zadatak s vektorskim oznakama
- zadatak s međusobnim iskljucivanjem pomocu prstena, racunanje vremena da se obiđe prsten
- sto su sigurnost, idempotentnost i cachable i koje od njih ima GET
- slican ovome:

Slika prikazuje tri procesa i razmjenu poruka među njima. Svaki proces neovisno o drugim procesima bilježi svoja stanja u označenim kontrolnim točkama. U trenutku *t* dolazi do ispada procesa P3. Možemo li sustav od tri procesa na slici nakon ispada procesa P3 vratiti u konzistentno stanje koristeći kontrolne točke K_{11}, K_{21} i K_{31} i objasnite zašto je to moguće ili nije moguće?



-

$$e_i^x \rightarrow e_j^y \Leftrightarrow \begin{cases} e_i^x \rightarrow e_j^y, (i=j) \wedge (x<y) \\ e_i^x \rightarrow_{msg} e_j^y \\ e_i^x \rightarrow e_k^z \wedge e_k^z \rightarrow e_j^y \end{cases}$$

a) događaji su uzročno povezani ako su jedan iza drugog na istom procesu; b) ako postoji slanje i primanje poruka između njih;
c) definira tranzitivnu uzročnost

–

5. Odaberi među ponuđenim ona koja upisuju dijeljeni podatkovni prostor.
vremenska ovisnost / vremenska neovisnost
perzistentna / tranzijentna
sinkrona / asinkrona
push / pull

–

11.

Pretpostavimo da skupina od ***n*** procesa implementira **algoritam za odabir vođe u sinkronom prstenu**. Svaki proces u prstenu odabire vlastiti jedinstveni identifikator UID koji je cijeli broj, procesi su povezani u prstenu na način da imaju jednog prethodnika i sljedbenika, UID nisu sljedbenici u prstenu, a procesi ne znaju ***n***.
a) Opišite ukratko ideju algoritma (traži se skica algoritma, a ne formalni algoritam) i
b) navedite vremensku i komunikacijsku složenost algoritma.
c) Možemo li definirati bolji algoritam u slučaju da procesi znaju ***n***, tj. postoji li kakva prednost pri definiranju algoritma za slučaj da svim procesi znaju ***n***.

a. Svaki proces inicijalno šalje svoj UID susjedu. Kada proces primi UID, ako je taj veći od njegovog UID-a prosljeđuje ga dalje, ako je primljeni UID manji od njegovog UID-a primljeni UID se odbacuje, a ako je primljeni UID jednak njegovom UID-u proces objavljuje sebe kao vođu

b. vremenska: O(n), komunikacijska: O(n^2)

bilo ih je jos par, ali ih se vise ne mogu sjetit