Причини за проблеми со административна скалабилност (administrative scalability) кај дистрибуирани системи и нивно надминување:

- **Причини:**
  - **Комплексност на управување**: Со зголемување на бројот на јазли, одржувањето и координацијата стануваат посложени.
  - **Недостаток на автоматизација**: Рачното администрирање на нови јазли или ресурси води до грешки и бавно управување.
  - **Несоодветни политики за ажурирање**: Тешкотии во синхронизацијата при ажурирање на конфигурации.
  - **Недоволна автоматска еластичност**: Неможност системот автоматски да се прилагоди на промените.
- **Надминување:**
  - **Автоматизација на административни задачи**: Користење алатки како Kubernetes, Terraform или Ansible.
  - **Централизирана администрација**: Воведување централизирани контролни панели за координација.
  - **Сеопфатна документација**: Стандардизирање на конфигурациските процеси.
  - **Динамичка оркестрација**: Воведување на системи што автоматски распоредуваат ресурси.

Разлика помеѓу Активна и Пасивна репликација:

- **Активна репликација**:
  - **Принцип**: Сите реплики извршуваат исти операции истовремено.
  - **Позитивни страни**: Висока достапност и издржливост на грешки.
  - **Примери за употреба**: Системи каде консистентноста е критична (банкарство).
  - **Протоколи**: Често се користи **вотинг** механизам.
- **Пасивна репликација**:
  - **Принцип**: Само примарната реплика извршува операции, а потоа ажурира другите (секундарни).
  - **Позитивни страни**: Пониска латенција во повеќе читања.
  - **Примери за употреба**: Системи каде читањето е доминантно (каталози).
  - **Протоколи**: Примарно-секундарен модел.

Safety и Liveness својства:

- **Safety**: "Ништо лошо нема да се случи."
- Може да се нарушат во конечно време
  - Пример: Меѓу две транзакции нема конфликт.
  - FIFO испорака, Каузална испорака и Целосно подредена испорака
- **Liveness**: "Нешто добро ќе се случи на крајот."
- Не можат да се нарушат во конечно време

- Пример: Трансакција секогаш ќе заврши.

Вредност на Лампортовиот часовник во настанот H:

- Формула: $LC(H) = \max(LC(A), LC(B)) + 1.$
- Со $LC(A) = 1$ и $LC(B) = 1$, вредноста ќе биде: $LC(H) = \max(1, 1) + 1 = 2.$

Релација „се случило пред“ со настанот Q:

За дадени два настани A и B:
• ако A и B се случуваат во ист процес каде што A се случува пред B тогаш пишуваме A->B
• ако A е настан на испраќање на порака и B е настан на примање на истата порака, тогаш A->B
• ако C е настан таков што A->C и C->B тогаш A->B

Добри страни на „партиционирај по клуч во рангови“:

- **Лесно пребарување**: Сите клучеви во одреден ранг се групирани.
- **Примена**: Корисно кај бази каде се потребни опсежни пребарувања.

За што се користи anti-entropy кај Dynamo?

- **Anti-entropy**: За синхронизација помеѓу јазли со размена на верзии на податоци и надминување на грешки.

Причини за географска скалабилност:

- **Причини**:
  - Висока латенција помеѓу далечни јазли.
  - Ограничувања во пропусноста.
  - Различни времиња на одговор.
- **Надминување**:
  - Локални реплики.
  - Content Delivery Networks (CDNs).
  - Оптимизација на мрежната топологија.

Проблемот на два генерали (omission failure):

- **Надминување**:
  - **Timeout механизам**: Генералите поставуваат рок за одговор.
  - **Ретрансмисија**: Повторно праќање на пораки

Кога се користи активна/пасивна репликација:

- **Активна репликација**: Кога е потребна висока достапност.
- **Пасивна репликација**: Кога се приоритизира оптимизација на ресурси.

(белешке од последната лекција):

- Каузална испорака: е својство на извршувањето кое сакаме да важи
- Каузална емисија: е алгоритам кој ни овозможува имлеменетација на каузална испорака во околина каде сите пораки се broadcast пораки

- Досега кога зборувавме за пораки, секогаш имаше еден испраќач, еден примач
  • Unicast
  • Point-to-point

- Имавме и случаи на праќање на иста порака до повеќе примачи
  • Multicast

- Емитување (broadcast) е праќање на една порака до сите
- Во околика каде сите испратени пораки се broadcast релативно лесно е да се имплементира каузална испорака

- FIFO испорака: Ако процес праќа порака m2 по праќање на порака m1, тогаш било кој процес кој врши испорака на пораките мора прво да ја испорача m1.

- FIFO испорака може да се имплементира со секвенцни броеви
- Со користење на секвенцен број SN, FIFO испораката ќе работи само ако имаме надежна испорака (reliable delivery)

- Праќање (sending) на порака е нешто што го извршува процесот
- Примање (receiving) на порака е нешто што му се случува на процесот
- Испорака (delivering) на порака е нешто што можеш да направиш со пораката при Примање (receiving)

- Каузална испорака
  • Ако испраќањето на m1 „се случило пред“ испраќањето на m2, тогаш испораката на m1 мора да „се случи пред“ испораката на m2.

- Целосно подредена испорака:
  • Ако процес врши испорака на m1 и потоа на m2, тогаш сите процеси кои вршат испорака на m1 и m2, го испорачуваат прво m1.

- Секој векторски часовник претставува информација колку настани (пораки) знае секој од процесите за другите процеси.
- Логички часовници – се користат само за подредување на настани
- Лампортови часовници – еден вид на логички часовници

- Неможноста на Лампортовите часови да ја карактеризираат каузалноста го надминуваме со воведување на нивно проширување во Векторски часовници
- Иако викаме дека Лампортовите часовници се конзистентни со каузалноста, велиме дека Лампортовите часовници не ја карактеризираат каузалноста.

- Карактеризира каузалност значи:
  LC(A) < LC(B) => A->B
  • Лампортовите часовници ја немаат оваа карактеристика

  Alice има вектор [5, 0, 0].
  • Што [5, 0, 0] значи?
  Тоа значи дека до тој настан, Alice е свесна за 5 сопствени настани, и не е свесна за ниту еден настан за Bob и Carol.

  Синхрони мрежа
  • е мрежа во која постои некое n така што не постои порака на која и се потребни повеќе од n временски единици да биде доставена.

  Асинхрона мрежа
  • е мрежа во која не постои n така што не постои порака на која и се потребни повеќе од n временски единици да биде доставена.
  • Реалните мрежи се асинхрони

  • Во рамки на дистрибуираните системи потребно е да се креираат повремени снимања на состојбата на целиот систем
  • Тоа го нарекуваме Дистрибуирана глобална слика на состојбата

  • Децентрализиран алгоритам е алгоритам во кој може да има повеќе иницијатори
  • Chandy-Lamport & Paxos се децентрализирани.

  • Централизиран алгоритам мора да биде иницира од само еден процес.