Engineer

Висока доступність

здатність сервісу працювати далі, коли відмовляє частина

Що це таке

Висока доступність — це властивість сервісу й далі відповідати, коли одна з машин, дисків, мережевих ліній чи процесів, на яких він тримається, відмовляє. Зазвичай її будують як кластер: кілька хостів, кожен із яких може запустити сервіс, програмне забезпечення, яке за ними стежить, і процедура, що переносить роботу з хоста, який відмовив, на справний. Проєкт ClusterLabs, що супроводжує відкритий кластерний стек Pacemaker і Corosync, описує такий кластер як систему, що узгоджує запуск, моніторинг і відновлення сервісів на наборі хостів, «щоб мінімізувати простій і водночас зберегти цілісність даних».

Друга половина цього речення важить не менше, ніж перша. Сервіс, який швидко повертається з хибними даними, доступнішим не став. Він став небезпечним.

Документація PostgreSQL розрізняє дві причини, з яких кілька серверів баз даних працюють разом. Перша — щоб другий сервер міг швидко взяти роботу на себе, якщо основний відмовить; це документація називає високою доступністю. Друга — щоб кілька комп'ютерів могли віддавати ті самі дані; це вона називає балансуванням навантаження. Обидві часто спираються на ті самі механізми, але розв'язують різні задачі: висока доступність — про те, як пережити збій, а балансування навантаження й масштабованість — про те, як упоратися з більшим обсягом роботи.

Сама доступність — це число. Традиційна міра базується на часі: частка періоду, протягом якої сервісом можна було користуватися. Книга Google про SRE пояснює, що для сервісів, розподілених по всьому світу, де якась

частина майже завжди десь працює, Google натомість вимірює частку запитів, на які вдалося успішно відповісти. Обидві міри записують у відсотках, а говорять про них як про кількість дев’яток.

Будь-яке рішення з високою доступністю складається з тих самих кількох частин:

- **Резервування.** Запасний елемент для кожної частини, відмова якої зупинила б сервіс: другий хост, standby-сервер бази даних, другий мережевий шлях. Pacemaker підтримує, серед іншого, пари active/passive і схеми N+1, у яких кілька активних вузлів мають один спільний запасний.
- **Виявлення.** Хтось має помітити збій. Розділ документації PostgreSQL про failover описує heartbeat між основним сервером і standby, який постійно перевіряє з’єднання і те, чи основний сервер ще працює.
- **Failover.** Перенесення роботи: підвищення standby до основного сервера, перенесення IP-адреси сервісу, запуск процесу на іншому хості.
- **Fencing.** Гарантія того, що вузол, який відмовив, справді зупинився, перш ніж його робота почнеться деінде.
- **Відновлення запасу.** Після failover система працює без резервування, доки не з’явиться новий запасний елемент.

Висока доступність — це не резервна копія і не аварійне відновлення. Вона захищає від відмови однієї частини, поки решта дата-центру працює. Резервна копія захищає від втрати чи пошкодження даних, зокрема й такого пошкодження, яке кластер старанно скопіював на кожен вузол, а аварійне відновлення — це план на випадок втрати всього дата-центру.

Застосуйте поняття до рішення

Застосуйте «Що таке висока доступність?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Моніторинг і сповіщення, які попереджають, поки збій ще малий, і цифри, що показують, як часто так і залишається.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Запустила перший дашборд спостережуваності — перший огляд власних систем компанії

Запустила перший дашборд спостережуваності Зв'язатися з нами →

Чому це важливо

У кожного сервісу є частини, які відмовлять. Ламається обладнання, трапляється паніка ядра, віртуальну машину помилково видаляють, мережа розділяється й відрізає справний вузол від решти кластера. Документація Kubernetes відносить такі події до мимовільних збоїв (involuntary disruptions) і протиставляє їм добровільні, які хтось планує, наприклад, звільнення вузла для ремонту чи оновлення. Без резервування кожна з цих подій стає простоєм, зокрема й запланована, бо єдиний спосіб оновити один-єдиний сервер — вимкнути його.

Саме останнє часто стає першою перевагою, яку помічає команда. Розділ документації PostgreSQL про failover називає корисним регулярне перемикання між основним сервером і standby: так кожну машину можна по черзі вивести на обслуговування, а кожне перемикання перевіряє, що механізм failover спрацює, коли буде потрібен.

Дев'ятки можна перевести в час. Для року з 365 днів (525 600 хвилин) і місяця з 30 днів (43 200 хвилин) розрахунок такий:

Доступність	Допустимо за рік	Допустимо за 30-денний місяць
99%	3,65 дня	7,2 години
99,9%	8,76 години	43,2 хвилини
99,95%	4,38 години	21,6 хвилини
99,99%	52,56 хвилини	4,32 хвилини
99,999%	близько 5,26 хвилини	близько 26 секунд

Кожна додаткова дев'ятка зменшує допустимий час удесятеро, а вартість росте швидше. Книга про SRE застерігає, що ще один крок у надійності може коштувати в сто разів більше, ніж попередній, і що з певного моменту користувачі вже не бачать різниці: людина з телефоном, який сам надійний лише на 99%, не помітить, чи сервіс за ним працює на 99,99% чи на 99,999%. Тому книга розглядає ціль доступності водночас як нижню межу і як стелю. Сервіс, набагато доступніший, ніж потрібно, витратив гроші й інженерний час, які могли б принести більше користі деінде.

Ось чому ціль — це бізнес-рішення, яке ухвалюють окремо для кожного сервісу. Число фіксує ціль рівня обслуговування; висока доступність — це інженерна робота, яка робить його досяжним.

Як це виглядає на практиці

Візьмімо вигаданий сервіс бронювання з трьома рівнями: вебрівнем, рівнем застосунку та базою даних PostgreSQL, кожен на окремому хості. Усі числа нижче вигадані для прикладу. Якщо кожен рівень працює 99,9% часу, а сервісу потрібні всі три, їхня доступність перемножується. Дублювання рівня діє навпаки, бо недоступними мають стати обидві копії одночасно:

```
Три рівні послідовно, кожен доступний на 99,9%:
  0,999 × 0,999 × 0,999 ≈ 0,997      близько 26 годин простою на рік

Один рівень продубльовано, якщо дві копії відмовляють незалежно:
  1 − 0,001 × 0,001 = 0,999999       близько пів хвилини на рік
```

Друге обчислення тримається на одному слові: незалежно. Дві копії на тому самому хості, в тій самій стійці, за тим самим комутатором чи в тій самій зоні хмарного провайдера мають спільну долю, і арифметика перестає працювати. Формула також припускає, що перемикання з однієї копії на іншу миттєве й завжди вдається, а це може показати лише тестування.

У Kubernetes саме для цього існують topology spread constraints. Власний приклад документації — робоче навантаження, яке може зменшитися до двох подів: не хочеться, щоб обидва опинилися на одному вузлі, де одна відмова виведе з ладу все навантаження. Обмеження в шаблоні пода розподіляє репліки між зонами:

```
topologySpreadConstraints:
    - maxSkew: 1
      topologyKey: topology.kubernetes.io/zone
      whenUnsatisfiable: DoNotSchedule
      labelSelector:
          matchLabels:
              app: booking
```

З `DoNotSchedule` планувальник радше залишить под нерозміщеним, ніж поставить його туди, де зони відрізнятимуться більш ніж на один. Запланованому обслуговуванню потрібен ще один захист. PodDisruptionBudget обмежує, скільки реплік застосунку можуть бути

недоступні одночасно через добровільні збої. `kubectl drain` дотримується бюджету, бо просить Eviction API виселити кожен под; пряме видалення подів його обходить:

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
    name: booking-pdb
spec:
    maxUnavailable: 1
    selector:
        matchLabels:
            app: booking
```

Документація радить `maxUnavailable` замість `minAvailable`, бо цей параметр автоматично враховує зміну кількості реплік.

База даних — найскладніший рівень, бо вона зберігає стан. PostgreSQL може тримати hot standby, який приймає запити лише на читання, або warm standby, до якого не можна підключитися, доки його не підвищать. Вибір між синхронною та асинхронною реплікацією — це вибір того, що може втратити failover. За синхронної реплікації підтвердження коміту чекає на відповідь від заданої кількості синхронних резервних серверів на визначеному рівні підтвердження. Ці умови задають `synchronous_standby_names` і `synchronous_commit`. Під час перемикання треба обрати резервний сервер, який має підтверджені дані; асинхронний резервний сервер може ще відставати. Ціна першого варіанта — швидкодія: документація зазначає, що повністю синхронне рішення в повільній мережі може знизити продуктивність більш ніж удвічі.

PostgreSQL сам не вирішує, коли перемикатися. Його документація каже, що він не постачає системного програмного забезпечення, яке виявляло б відмову основного сервера й сповіщало standby; це завдання, як і перенесення IP-адреси, належить зовнішньому кластерному програмному забезпеченню. Саме підвищення — один крок: `pg_ctl promote` або функція `pg_promote()`. Те, що відбувається далі, плани зазвичай забувають. Коли старого основного сервера вже немає, кластер працює на одному сервері, і документація називає це виродженим станом, доки не збудують новий standby. Утиліта `pg_rewind` може пришвидшити це у великих кластерах.

На цьому сайті кейс про пошту з failover і журналом доставки показує той самий шаблон, застосований до зовнішньої залежності. Три поштові провайдери сховані за одним інтерфейсом, перемикання між ними відбувається автоматично, а не через зміну конфігурації під тиском, а перевірка під час запуску надсилає справжній лист, тож staging і production не стартують без провайдера, який працює. Як зазначено в кейсі, невдалий день у провайдера тепер означає роботу в деградованому режимі, а не аварію, ціною трьох інтеграцій, які треба підтримувати в робочому стані. Наскільки доступною має бути система, — окреме рішення: сторінка про рівень обслуговування пояснює, що доступність записують в угоду окремо для кожної системи, бо нічний звіт і платіжний сервіс потребують різних відповідей.

Де виникають проблеми

Резервування зі спільною долею. Дві копії на одному хості, одному блоці живлення, одному комутаторі чи в одній хмарній зоні відмовляють разом. Простежте, від чого залежить кожен запасний елемент, і розподіляйте репліки між доменами відмов свідомо, а не навмання.

Split brain. Коли вузли кластера втрачають зв’язок один з одним, але далі працюють, кожна сторона може вирішити, що інша мертва, і сама запустити сервіс. Документація Pacemaker описує наслідок: пакети до IP-адреси, піднятої на двох хостах, навмання потрапляють то на один, то на інший, а для бази даних шкодою можуть стати пошкоджені або розбіжні дані. Захистом слугує fencing, який ще називають STONITH (Shoot The Other Node In The Head): fence-пристрій, наприклад керований кластером вимикач живлення, від’єднує вузол, що відмовив, перш ніж його робота почнеться деінде. Документація прямо каже, що без fencing кластер не може безпечно відновитися після відмови вузла, який не відповідає, і що fence-пристрій не має залежати від власної мережі кластера, інакше втрата цієї мережі позбавить кластер і засобів ізоляції.

Failover, якого ніхто не пробував. Standby, якого жодного разу не підвищували, є надією, а не планом. Міняйте ролі за розкладом, як пропонує документація PostgreSQL, запишіть процедуру й відпрацьовуйте її так само, як хаос-інженерія відпрацьовує інші збої.

Забутий вироджений стан. Після failover запасного елемента немає. Якщо відновлення standby повільне чи ручне, наступна несправність, хай яка дрібна, стає простоєм. Вважайте відновлення резервування частиною інциденту, а не завданням на потім.

Віра в те, що асинхронна реплікація нічого не втрачає. Вона може втратити останні транзакції, зафіксовані перед збоєм. Для кожної бази даних вирішіть, чи це прийнятно, і зробіть цей компроміс видимим для тих, кому належать дані.

Реплікація замість резервної копії. Standby копіює помилковий `DELETE` так само старанно, як і правильний запис. Висока доступність тримає сервіс у роботі; повернутися до даних у тому вигляді, в якому вони були, команді дозволяє лише резервна копія.

Два вузли й жодного арбітра. Документація Pacemaker звертає увагу, що втрата вузла через знеструмлення виглядає так само, як утрата мережевого зв'язку з ним. Коли учасників двоє, кожна сторона бачить саме це. Kubernetes вимагає щонайменше трьох вузлів площини керування в топології stacked, а документація PostgreSQL згадує третій witness-сервер, який може запобігти деяким хибним перемиканням, але застерігає, що додаткова складність виправдана лише за ретельного налаштування й ґрунтовного тестування.

Плата за дев'ятки, яких ніхто не помітить. Кожна додаткова дев'ятка коштує більше, ніж попередня, а користувачі доходять до сервісу через мережі й пристрої, менш надійні, ніж ціль. Вибирайте число з огляду на те, що потрібно користувачам і бізнесу, а потім будуйте під нього.

Що почитати далі

- ClusterLabs — відкритий кластерний стек для високої доступності з визначенням того, що узгоджує кластер і навіщо йому fencing.
- High Availability, Load Balancing, and Replication — розділ PostgreSQL про основні сервери та standby, про синхронні й асинхронні рішення та їхню ціну.
- Failover — PostgreSQL про heartbeat, witness-сервери, STONITH, підвищення й вироджений стан.
- Fencing — Pacemaker Explained про split brain, fence-пристрої, fencing і кворум.

- Embracing Risk — книга Google про SRE: як вимірювати доступність, скільки коштує кожна дев’ятка і як вибрати ціль.
- Disruptions — добровільні й мимовільні збої в Kubernetes і те, як PodDisruptionBudget їх обмежує.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке висока доступність?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/high-availability/