https://orcid.org/0000-0002-9068-5837

УДК 519.718

Н.В. СЕСПЕДЕС ГАРСІЯ*

ДО ПИТАННЯ ОЦІНКИ НАДІЙНОСТІ КЛАСТЕРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

*Інститут проблем математичних машин і систем НАН України, м. Київ, Україна

Анотація. *Стаття присвячена дослідженню міжнародних підходів до оцінювання надійності кластерних обчислювальних систем та поєднання з сучасними вітчизняними методологіями. Кластерні системи широко застосовуються у фундаментальних та прикладних дослідженнях, чисельному моделюванні складних фізичних процесів, кліматичному прогнозуванні, аналізі великих обсягів даних, у аерокосмічній та автомобільній галузях, енергетиці, хмарних обчисленнях та інших сферах. Тому оцінка надійності кластерних обчислювальних систем є критично важливою для забезпечення їх безперервного функціонування та стійкості до збоїв в умовах інтенсивного навантаження. Комплексна оцінка надійності кластерних систем складається з таких етапів: аналіз компонентів структури системи; обчислення метрик надійності; моделювання сценаріїв відмов; аналіз імовірностей комбінованих збоїв; оцінка резервування. Встановлено, що в цілому кластерні системи мають так звану « k » із « n » структуру, тому метрики надійності розраховують згідно з відповідними вітчизняними методологіями. У загальному випадку відмови кластерних обчислювальних систем мають немонотонний характер. Тому для розрахунку, наприклад, імовірності безвідмовної роботи можна застосувати сучасні вітчизняні методології, засновані на DN-розподілі відмов. Оцінка надійності кластерних обчислювальних систем являє собою багаторівневий підхід, що поєднує аналітичні, експериментальні та модельні методи дослідження. Такий комплексний підхід дозволяє виявити потенційні вузькі місця в архітектурі системи, визначити критичні компоненти та оцінити ризики збоїв. Це дає змогу своєчасно впроваджувати ефективні механізми відмовостійкості, резервування та автоматичного відновлення.*

Ключові слова: *кластер, надійність, обчислювальні інформаційні системи, хаос тестування, балансування навантаження, DN-розподіл відмов.*

Abstract. *The article is devoted to the study of international approaches to assessing the reliability of cluster computing systems and their combination with modern domestic methodologies. Cluster systems are widely used in fundamental and applied research, numerical modeling of complex physical processes, climate forecasting, analysis of large volumes of data, the aerospace and automotive industries, energy, cloud computing, and other areas. Therefore, assessing the reliability of cluster computing systems is critically important for ensuring their continuous operation and resistance to failures under conditions of intensive load. A comprehensive assessment of the reliability of cluster systems consists of the following stages: analysis of system structure components, calculation of reliability metrics, modeling of failure scenarios, analysis of combined failure probabilities, and assessment of redundancy. It has been determined that, in general, cluster systems have a so-called «k» with «n» structure, therefore, reliability metrics are calculated according to the corresponding domestic methodologies. In the general case, failures of cluster computing systems are non-monotonic in nature. Consequently, to calculate the probability of a failure-free operation, modern domestic methodologies based on the DN-distribution of failures can be used. Assessment of the reliability of cluster computing systems is a multi-level approach that combines analytical, experimental, and model research methods. Such a comprehensive approach allows the identification of potential bottlenecks in the system architecture, identify critical components, and assess the risks of failures. This enables timely implementation of effective mechanisms for fault tolerance, redundancy, and automatic recovery.*

Keywords: *cluster, reliability, computing information systems, chaos testing, load balancing, DN-failure distribution.*

ISSN 1028-9763. Математичні машини і системи. 2026. № 1

DOI: 10.34121/1028-9763-2026-1-100-108

1. Вступ

Обчислювальний кластер являє собою інтегровану багатовузлову обчислювальну інфраструктуру, утворену сукупністю автономних апаратних модулів (вузлів), які об'єднані за допомогою високошвидкісного мережевого середовища (наприклад, InfiniBand або Ethernet з низькою затримкою). Таке об'єднання забезпечує спільне функціонування вузлів у межах єдиного логічного ресурсу, що сприймається кінцевим користувачем як уніфікована система. Основним призначенням кластерної архітектури є забезпечення високої продуктивності шляхом реалізації паралельної або розподіленої обробки задач великого обчислювального навантаження, що робить такі системи ефективним рішенням у сфері наукових, інженерних та аналітичних досліджень.

Мета статті — дослідження міжнародних підходів до оцінювання надійності кластерних обчислювальних систем та поєднання з сучасними вітчизняними методологіями.

2. Кластерна архітектура паралельних обчислювальних систем. Загальні принципи та галузь використання

Обчислювальний кластер — це взаємопов'язане середовище, що складається з множини вузлів (комп'ютерів), з'єднаних через високошвидкісну мережу, які функціонують як єдина обчислювальна система для паралельного або розподіленого вирішення спільного завдання. Комп'ютери, які утворюють кластер (так звані вузли або ноди кластера), як правило, є відносно незалежними, тобто зупинка або відключення окремого вузла (наприклад, для технічного обслуговування або встановлення нового обладнання) не впливає на загальну працездатність кластера.

Як обчислювальні вузли можуть використовуватись звичайні однопроцесорні персональні комп'ютери, сервери з двома або чотирма процесорами (так звані SMP-системи — симетричні багатопроцесорні) [1]. Кожен вузол має власну копію операційної системи (найчастіше це Linux, Windows NT, Solaris тощо). Слід зазначити, що вузли можуть бути різними за потужністю та конфігурацією, що дозволяє створювати неоднорідні кластери.

Вибір конкретного комунікаційного середовища (мережа та програмні протоколи) визначається багатьма факторами: особливостями класу завдань, необхідністю подальшого розширення кластера тощо. В конфігурацію кластера можливе включення спеціалізованих комп'ютерів, наприклад, файл-сервера, з можливістю віддаленого доступу на кластер через Internet [1].

Під поняття «кластер» підпадає дуже широкий спектр систем — від найпростіших до надскладних. Наприклад, кластером можна вважати як пару ПК, з'єднаних через звичайну 10-мегабітну мережу Ethernet, так і кластерну суперкомп'ютерну систему ASCI Red (Sandia National Laboratories, США), яка має більше ніж 9000 вузлів на базі процесорів Pentium II Xeon, з'єднаних високошвидкісною мережею Myrinet із пропускною здатністю до 1 Гбіт/с між вузлами та продуктивністю більш ніж трильйон (10^{12}) операцій на секунду (1 TFLOPS).

Також слід зазначити кластер Beowulf — один із перших проєктів, що дав ім'я цілому класу паралельних систем, був створений у рамках проєкту Earth and Space Sciences в науково-космічному центрі NASA у 1994 році. Кластер Beowulf складався з 16 процесорів Intel 486DX4/100 МГц, на кожному вузлі було встановлено по 16 Мбайт оперативної пам'яті та по три мережеві Ethernet-адаптери. Дана система виявилася дуже вдалою щодо ціни/продуктивності, тому подібну архітектуру почали розвивати та широко використовувати в інших наукових організаціях та інститутах. Для створення кластерів Beowulf зазвичай використовується програмне забезпечення з відкритим вихідним кодом, що дозволяє зменшити фінансові витрати та розширює можливості налаштування програмного забез-

печення відповідно до власних потреб [2]. Більшість кластерів Beowulf використовують BSD, Linux чи Solaris — операційні системи. Використання Message Passing Interface (MPI) та Parallel Virtual Machine (PVM) дозволяє розподілити виконання задач по групі комп'ютерів, які об'єднані у мережу. Прикладами реалізації MPI є Open MPI та MPICH [2]. На рівні суперкомп'ютерів це не просто складання споживчих комп'ютерів у кластер, а об'єднання спеціально розроблених вузлів (blade servers) за допомогою спеціального мережевого забезпечення та спеціальних систем охолодження.

Під час проєктування та розгортання високопродуктивних інформаційних систем критично важливими є показники надійності, доступності та обчислювальної ефективності. Для досягнення цих цілей комп'ютерні ресурси об'єднують у кластерну архітектуру, яка дозволяє створити єдине масштабоване обчислювальне середовище з можливістю динамічного розподілу навантаження та підвищеної відмовостійкості.

Існують дві основні моделі архітектури кластерних систем:

• Централізована модель, у якій використовується єдиний сервер зберігання (центральне сховище даних). Такий підхід забезпечує високу надійність та цілісність інформації, а також стійкість системи до збоїв окремих обчислювальних вузлів, оскільки дані не дублюються на кожному з них.

• Децентралізована модель, що передбачає функціонування автономних вузлів із власними локальними ресурсами. У разі виходу з ладу одного з вузлів система виконує динамічний перерозподіл навантаження між активними елементами кластера, що дозволяє зберігати високу продуктивність та забезпечує відмовостійкість без централізованого компонента.

Конкретна архітектура кластерної системи визначається характером задач, які вона має вирішувати, що, зі свого боку, зумовлює вибір апаратних і програмних рішень, а також принципів побудови.

У загальному випадку виділяють три основні типи кластерів:

• Кластери високої доступності (High-Availability Clusters, HA) — орієнтовані на забезпечення безперервного функціонування критично важливих сервісів за рахунок механізмів відмовостійкості та автоматичного перемикання на резервні вузли у разі збою [3].

• Кластери з балансуванням навантаження (Load Balancing Clusters) — призначені для рівномірного розподілу вхідного трафіку або обчислювальних задач між усіма доступними вузлами з метою оптимізації використання ресурсів і підвищення продуктивності системи.

• Обчислювальні кластери (High Performance Computing Clusters, HPC) — спеціалізовані на виконанні ресурсоємних паралельних обчислень, що характерно для наукових досліджень, інженерного моделювання та обробки великих масивів даних [3].

Кластерні високопродуктивні сервери (HPC-системи) широко застосовуються у фундаментальних та прикладних дослідженнях, чисельному моделюванні складних фізичних процесів, кліматичному прогнозуванні, аналізі великих обсягів даних (Big Data) та виконанні обчислювально інтенсивних симуляцій. Сучасні кластерні суперкомп'ютери, що об'єднують мільйони процесорних ядер у єдину паралельну обчислювальну інфраструктуру, використовуються в таких галузях, як-от метеорологія, квантова фізика, геноміка, генно-інженерні дослідження, аерокосмічна та автомобільна галузі, енергетика, хмарні обчислення та сервіси [4].

Кластерні обчислювальні системи активно застосовуються в задачах глибокого машинного навчання, зокрема для тренування нейронних мереж. Глибокі нейромережі (deep neural networks) є обчислювально ємними та потребують значних ресурсів, що робить кластери ідеальним середовищем для розподіленої обробки. Завдяки паралельному виконанню на багатьох вузлах, кластер з 8–16 GPU здатен прискорити навчання моделей у 10–100 разів порівняно з одним вузлом [4]. Після завершення етапу навчання кластерні системи

ефективно використовуються для масштабованого застосування: обробки мільйонів зображень, потокового відео чи запитів користувачів у реальному часі з мінімальною затримкою.

Сучасні фреймворки для машинного навчання, як-от TensorFlow, PyTorch, мають вбудовану підтримку розподіленого навчання у кластерному середовищі [5]. У хмарних обчислювальних платформах (AWS, GCP, Azure) масштабовані тренувальні задачі розгортаються з використанням систем оркестрації — Kubernetes, Horovod (розроблений Uber), Ray, Dask тощо. У сценаріях розподіленого навчання (distributed training) модель та дані поділяються між декількома вузлами, що дозволяє ефективно масштабувати навчання та зменшити загальний час обробки навіть при роботі з великими наборами даних.

Тому оцінка надійності кластерних обчислювальних систем є критично важливою для забезпечення їх безперервного функціонування та стійкості до збоїв в умовах інтенсивного навантаження. Вона дозволяє своєчасно виявляти потенційні вузькі місця, підвищувати надійність інфраструктури та мінімізувати ризики втрати даних чи зниження продуктивності. Такий аналіз є основою для прийняття обґрунтованих рішень щодо проєктування, масштабування та резервування систем.

3. Комплексна оцінка надійності кластерних обчислювальних систем

Надійність кластерної обчислювальної системи визначається як її здатність забезпечувати безперервність та коректність функціонування у присутності збоїв окремих апаратних або програмних компонентів. Високий рівень надійності досягається за рахунок вбудованих механізмів резервування, автоматичного виявлення відмов та перенесення навантаження (failover), а також здатності до відновлення критичних сервісів без втрати даних або порушення доступності. Така властивість кластерної системи є ключовою для забезпечення її надійності, стійкості до збоїв і відповідності вимогам високодоступних обчислювальних середовищ, зокрема, в контексті обробки критичних або безперервних завдань.

Оцінка надійності (reliability) включає такі етапи.

3.1. Аналіз компонентів структури системи

Аналіз компонентів структури системи містить:

- визначення вузлів, мереж, дисків, балансувальників тощо;
- виявлення критичних точок (single point of failure (SPOF));
- створення моделі системи та аналіз (наприклад, граф взаємозв’язків компонентів).

Інструменти для моделювання та аналізу: *Draw.io, Lucidchart* призначений для побудови системних діаграм та графів залежностей; *Graphviz, Neo4j* призначений для побудови граф-моделей (аналітика залежностей); *Prometheus + Grafana* для отримання метрик щодо визначення навантажень на компоненти; *Nmap, Netbox* для сканування мереж та документування топології; *Kubernetes manifests* — це ключовий інструмент**,** який описує архітектуру, поведінку системи при збоях, стратегії відновлення та резервування компонентів у кластері. *Kubernetes manifests* служать основою для автоматизованого тестування стійкості (через chaos/fault testing) та є джерелом даних для формального аналізу архітектури (SPOF, redundancy, recovery time).

3.2. Обчислення метрик надійності

Найпоширеніші метрики:

- MTTF (Mean Time To Failure) — середній час до відмови;
- MTTR (Mean Time To Repair) — середній час відновлення;
- Availability (Готовність) — $A = MTTF / (MTTF + MTTR)$;
- Failure Rate — інтенсивність відмов (λ);

• Reliability $R(t)$ — імовірність безвідмовної роботи в інтервалі часу.

Для розрахунку ймовірності безвідмовної роботи необхідно розробити структурну схему надійності кластерної системи. У загальному випадку відмови кластерних обчислювальних систем мають немонотонний характер, що проявляється у непередбачуваній динаміці виникнення збоїв внаслідок складної взаємозалежності між компонентами системи та їх реакції на часткові порушення працездатності. Тому для розрахункку ймовірності безвідмовної роботи можна застосувати методології, які вичерпно викладені у роботах [6, 7] та складаються з таких обчислень.

Обчислення середнього наробітку до відмови T_1 кластерної системи зі структурою елементів (компонентів) «k» із «n» за формулою відповідно до структурної схеми надійності системи [6]:

$$T_1 = \frac{T_{1j}\left(n-k+1\right)}{\sqrt{n}},$$

де n — загальне число елементів (компонентів);

k — мінімальне число працездатних елементів (компонентів), яке необхідне для нормального функціонування системи $(k=1)$;

T_{1j} — середній наробіток до відмови елементів (компонентів) j-го типу.

Обчислення *відносного напрацювання* системи «k» із «n» за формулою [6]:

$$x = \frac{t_н}{T_1},$$

де $t_н$ — сумарне напрацювання системи.

Обчислення параметра форми (коефіцієнта варіації) *DN*-розподілу за формулою [6]:

$$V = V\left(n-k+1\right)^{-1/2}.$$

Обчислення ймовірності безвідмовної роботи за таблицею *DN*-розподілу [6] для значення V за формулою

$$_c^f R_s^q = c^s\left(1-{}^f F_s^q\right),$$

де ${}^f F_s^q$ — функція ймовірності відмови з урахуванням параметрів f, q та s;

$${}^f F_s^q = DN(x; v, f, q, s);$$

s — кількість резервів, спочатку доступних для підключення;

q — кількість модулів, що забезпечують задану продуктивність системи (характеристика актуальна для систем, продуктивність яких залежить від кількості одночасно працюючих ресурсів);

c — ступінь компенсації наслідків відмови (умовна ймовірність того, що при виникненні відмови в працюючій системі остання здатна відновити інформацію і продовжити її обробку без довготривалої втрати даних);

f — здатність модуля допускати f одиничних відмов до того, як він стане непрацездатним.

Функція ймовірності відмови для DN-розподілу має такий вид:

$$DN(x; v) = \Phi\left(\frac{x-1}{v\sqrt{x}}\right) + \exp\left(2v^{-2}\right)\Phi\left(-\frac{x+1}{v\sqrt{x}}\right),$$

де Φ — функція нормованого нормального розподілу.

Проведемо розрахунок імовірності безвідмовної роботи кластерної системи «k» із «n» за формулами, наведеними вище. Як приклад маємо кластерну систему «k» із «n» (k=2, $n=3$) з середнім наробітком T_1=*1000 год.*, середньоквадратичне відхилення σ =*200 год.* Розрахувати ймовірність безвідмовної роботи кластерної системи «k» із «n» до часу роботи $t_н$ =*900 год.*

Рішення: В даному випадку коефіцієнт варіації середнього наробітку V =0,2. Усі наступні параметри розраховуємо за формулами, наведеними вище: T_1=1154,7 *год.*, x =0,7794, V =0,1414, ${}^{f}F_s^q$ =0,044445, імовірність безвідмовної роботи ${}_c^f R_s^q$ =*0,955 (c=1).*

Щодо реальної статистики у кластерних системах через відхилення у виробництві умови експлуатації (температура, пил, напруга), навантаження (CPU, I/O, storage) навіть однакові сервери (один виробник, одна модель, одна дата випуску) не мають однакової надійності. Тому для вузлів однакової конфігурації зазвичай аналізують варіацію (коефіцієнт варіації) та характер розподілу часу до відмови (MTBF), щоб оцінити відхилення від середнього ресурсу. У середньому за статистикою для однотипних вузлів у сучасних дата-центрах коефіцієнт варіації MTBF зазвичай становить близько 10–30 %, що відображає значний розкид часу безвідмовної роботи навіть за однакових технічних характеристик. Для великих кластерів з понад 1000 вузлів типовою є ситуація, коли близько 5% вузлів формують більшу частину загальної кількості відмов, що обов'язково враховується під час планування надмірності та резервування.

Згідно зі статистикою Google, 2007: «Failure Trends in a Large Disk Drive Population» однакові HDD від одного виробника мають різницю у відмовах від 2 % до 13 % на рік у різних партіях. Регулярні звіти Backblaze Drive Stats показують, що одна модель HDD (12 ТБ) у різних дата-центрах має середній річний відсоток відмов (AFR) 0,5 %–6 % залежно від умов. У випадку з SSD розбіжність значень від 0,5 до 5 % на рік через сильну залежність від режиму експлуатації (інтенсивності записів і читань).

У кластерних системах суперкомп'ютерів LANL (Los Alamos National Laboratory) або BlueGene для однотипних обчислювальних вузлів фіксували розкид середнього часу безвідмовної роботи (MTBF) від 6 до 12 місяців навіть за однакових технічних параметрів. Згідно з дослідженнями NASA та Міністерства енергетики США (DOE), приблизно 10–20 % вузлів демонструють суттєво нижчий ресурс безвідмовної роботи через фактори, які важко передбачити заздалегідь, зокрема локальний перегрів, деградацію з'єднань або відмови модулів пам'яті.

3.3. Моделювання сценаріїв відмов

Моделювання сценаріїв відмов необхідно для перевірки стійкості системи та включає в себе**:**

- вимкнення вузлів або груп сервісів;
- імітацію мережевих проблем (затримка (latency), втрату пакета (packet loss));
- симуляцію втрати живлення/ресурсів;
- Fault Injection Testing (наприклад Chaos Monkey, Chaos Mesh).

Fault Injection та Chaos Testing — це два дуже близькі, але не тотожні підходи, які допомагають оцінити надійність кластерних обчислювальних систем. Обидва служать для того, щоб проактивно виявити слабкі місця до того, як відбудеться реальна відмова [8].

Fault Injection Testing (Тестування шляхом введення збоїв)

Fault Injection Testing — це контрольоване введення помилок або збоїв компонентів системи з метою перевірки реакції системи на них. Це можуть бути як апаратні, так і програмні помилки чи збої.

Типи «ін'єкційних» збоїв:

Категорія	Приклади
CPU/Memory	Перевантаження CPU, вичерпання RAM.
Диск/Файлова система	Затримка I/O, помилки читання/запису.
Мережа	Втрата пакетів, затримки, зміна DNS.
Сервіси	Вимкнення мікросервісу, зависання API.
Безпека/доступ	Втрата конфіденційних даних, сертифікатів, логинів, паролів.
Data Faults	Неконсистентні або пошкоджені дані.

Fault Injection Testing використовують для визначення точки відмови (Single Point of Failure), перевірки поведінки під навантаженням, перевірки роботи резервних компонентів, блоків, оцінки часу реакції та відновлення [8].

Chaos Testing (Chaos Engineering) (Хаотичне тестування)

Chaos Testing — це хаотичне введення помилок або збоїв у рамках усієї інфраструктури, системи або мікросервісної архітектури з метою перевірки реакції на них**.**

Інструмент Chaos Testing	Особливості
Chaos Mesh	Kubernetes-native, підтримує мережу, поди*, IO, CPU.
Gremlin	SaaS-платформа з GUI та API, гнучкі сценарії.
LitmusChaos	Kubernetes Chaos Engineering, інтеграція з CI/CD
Toxiproxy	Для мережевих збоїв.
Netflix Simian Army	Оригінальний Chaos Monkey + Chaos Gorilla, Kong.

*Поди (*pod*) у кластері (наприклад, у Kubernetes) — це найменша одиниця розгортання. Він містить один або кілька контейнерів, які запускаються разом і мають спільне середовище. Всі контейнери в pod'і мають спільну IP-адресу та порти, спільне сховище, контейнери в pod'і запускаються і завершують роботу разом.

Fault Injection Testing та Chaos Testing використовують для оцінки надійності такі компоненти кластера, як-от вузли (сервери Kubernetes, звичайні сервери), сервіси (API, бази даних, кеші), мережі, балансувальники навантаження, конфіденційні дані та сховища [8].

Оцінка надійності через хаос-тестування — це перевірка системи на здатність впевнено працювати в умовах несподіваних збоїв, швидко та коректно відновлюватися. Тобто, це перевірка підключення резервних вузлів, перевірка роботи автоматичного відновлення, перевірка перевищення параметрів затримки, кількості відмов, перевірка системи сповіщення [8].

Для моделювання сценаріїв відмов використовують TLA+ для формального моделювання, Chaos Engineering-фреймворки, Custom, Shell, Python-скрипти.

TLA+ — це формальна мова моделювання, яка використовується для опису, проєктування, верифікації та аналізу складних систем, особливо розподілених і багатопотокових. TLA+ мовна система (разом з інструментами) дозволяє описати поведінку складної системи (наприклад, кластера серверів, бази даних, протоколів тощо); виявити помилки в логіці, наприклад, втрату узгодженості, неправильне відновлення після відмови; моделювати сценарії з відмовами, перезапусками, мережевими затримками, неконсистентними (розсинхронізованими) станами і перевірити, чи зберігається гарантія роботи [9]. При оцінці надійності кластерних систем TLA+ дозволяє формально описати алгоритми відновлення після відмов вузлів, збоїв комунікації чи зіпсованих даних; створити абстрактну модель поведінки кластера у випадку відмови частини системи; автоматично перевірити, чи завжди система приходить у правильний стан (наприклад, чи завжди відновлюється кластер, чи узгоджені дані); знайти «рідкісні, нетипові» випадки, які важко передбачити або відтворити при звичайному тестуванні.

Chaos Engineering-фреймворки — це інструменти для навмисного створення збоїв у системі**,** щоб перевірити її стійкість до відмов**,** реакцію на несподівані події та здатність до самовідновлення.

Custom, Shell, Python-скрипти — це самостійно написані невеликі програми (скрипти), які автоматизують типові або спеціалізовані завдання в системному адмініструванні, DevOps, тестуванні тощо.

3.4. Аналіз імовірностей комбінованих збоїв

Для аналізу ймовірностей комбінованих збоїв у складних IT-системах використовуються методи теорії надійності, які дозволяють змоделювати, прорахувати та зменшити ризики одночасних або послідовних відмов компонентів. Нижче наведено основні інструменти та їх застосування [10]:

• Діаграми відмов (Fault Trees) або Reliability Block Diagrams — графічне представлення логічних зв’язків між відмовами компонентів, блоків і загальним збоєм системи. Використовується для розрахунку ймовірностей збоїв, для аналізу критичних шляхів відмов та виявлення слабких місць у резервуванні.

• Марковські моделі для обчислення складних станів, складних сценаріїв з відновленням, кількох режимів роботи. Створюють станову модель системи з переходами між станами, що дає змогу враховувати переходи між станами, ймовірності та час до відновлення**.**

• Monte Carlo Simulation — імітація випадкових сценаріїв відмов. Створює статистичну модель на основі випадкових подій згідно з заданими розподілами (наприклад, нормальним нормованим). Застосовується для визначення середнього часу до відмови (MTTF), середнього часу відновлення (MTTR) тощо.

3.5. Оцінка резервування

На даному етапі необхідно перевірити наявність:

• Резервування вузлів (*redundancy*) — наявність резервних вузлів, у тому числі критичних компонентів (сервери, канали зв’язку, джерела живлення) та їх коректне підключення у разі збоїв.

• Розподілу навантаження (*load balancing*) — наявність балансувальника навантаження (LB) для рівномірного розподілу навантаження після збою, наявність перевірки працездатності вузлів.

• Перемикання на резерв (*failover*) — наявність перевірки точного відтворення стану після перемикання — збереження даних, сесій користувачів, налаштувань, швидкості перемикання на резерв, проведення тестування сценаріїв аварійного перемикання.

4. Висновок

Оцінювання надійності кластерних обчислювальних систем є ключовим етапом у забезпеченні їхньої безперебійної роботи в умовах високої інтенсивності обчислювальних навантажень. Такий аналіз дозволяє виявити потенційні вузькі місця в архітектурі системи, визначити критичні компоненти та оцінити ризики збоїв. Це дає змогу своєчасно впроваджувати ефективні механізми відмовостійкості, резервування та автоматичного відновлення. Забезпечення високої надійності є необхідною умовою для стабільного функціонування обчислювальної інфраструктури в науці, промисловості та IT-сфері.

Оцінка надійності кластерних обчислювальних систем являє собою багаторівневий підхід, що поєднує аналітичні, експериментальні та модельні методи дослідження. До її складу входять аналітичні розрахунки, які дозволяють оцінити ймовірнісні характеристики функціонування системи; емпіричні випробування, зокрема, fault injection та chaos testing, що спрямовані на виявлення поведінкових особливостей системи під час відмов; а також моделювання за допомогою інструментів, як-от Reliability Block Diagrams, марковських процесів і сценарного аналізу. Такий комплексний підхід забезпечує глибоке розуміння механізмів відмов і підвищує ефективність заходів щодо підвищення стійкості до збоїв.

СПИСОК ДЖЕРЕЛ

1. Barney B., Frederick D. HPC @ LLNL: Introduction to Parallel Computing Tutorial. URL: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial?utm_source=chatgpt.com.
2. Beowulf (кластер). URL: https://ru.wikipedia.org/wiki/Beowulf_(%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80).
3. HPC @ LLNL: Introduction to Parallel Computing Tutorial. URL: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial.
4. CelerData: How Parallel Processing Shaped Modern Computing. URL: https://celerdata.com/glossary/how-parallel-processing-shaped-modern-computing?utm_source=chatgpt.com.
5. Learn: Distributed GPU training guide (SDK v2). URL: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-train-distributed-gpu?view=azureml-api-2&utm_source=chatgpt.com.
6. Стрельников В.П., Федухин А.В. Оценка и прогнозирование надежности электронных элементов и систем. К.: Логос, 2002. 486 с.
7. ДСТУ 2862-94. Надійність техніки. Методи розрахунку показників надійності. Загальні вимоги. К.: Видавництво Держстандарту України, 1995. 40 с.
8. Embracing the Chaos: Using Fault Injection Testing to Build Resilient AWS Architectures. URL: https://vitiya99.medium.com/embracing-the-chaos-using-fault-injection-testing-to-build-resilient-aws-architectures-ff25e77e6830.
9. Lamport L. My TLA+ Home Page. URL: https://lamport.azurewebsites.net/tla/tla.html.
10. HBK: Fault Tree Analysis, Reliability Block Diagrams and BlockSim. URL: https://www.hbkworld.com/en/knowledge/resource-center/articles/fault-tree-analysis-reliability-block-diagrams-and-blocksim?utm_source=chatgpt.com.

Стаття надійшла до редакції 03.12.2025 / прийнята до друку 12.02.2026