DOI: https://doi.org/10.36910/6775-2524-0560-2025-60-17
УДК 004.94
Кащенко Дмитро Олександрович, аспірант,
https://orcid.org/0000-0002-3365-7186
Черкаський Державний Технологічний Університет м. Черкаси, Україна

ВИКОРИСТАННЯ БІОНІЧНИХ ПРИНЦИПІВ У РОЗГОРТАННІ СИСТЕМ ОРКЕСТРАЦІЇ KUBERNETES

Кащенко Д.О. Використання біонічних принципів у розгортанні систем оркестрації KUBERNETES. У цій статті досліджується можливість застосування біонічних принципів у проектуванні систем розгортання оркестрації контейнерів на основі Kubernetes.. Розглядаються ключові компоненти архітектури Kubernetes як приклад складної динамічної системи, що характеризується саморегуляцією, масштабованістю та адаптивністю. Зроблено модуляцію розгортання компонентів Kubernetes. Проведено аналогії між функціональними модулями Kubernetes та структурами живих організмів, що дозволяє виявити подібності у принципах організації та функціонуванні. Запропоновано використання біонічних концепцій, таких як гомеостаз, емерджентна поведінка, суб'єктивні системи, та окреслено шляхи їхньої реалізації через сучасні технології — зокрема KEDA, Prometheus, Grafana, Kubecost. У роботі наведено приклади конфігурацій, що демонструють реактивність системи на зовнішні події та зміни навантаження, а також здатність до оптимізації використання ресурсів. Особливу увагу приділено перспективам розвитку Kubernetes як адаптивної та самозахисної платформи шляхом інтеграції з методами машинного навчання та біоінспірованими алгоритмами. Проведено порівняльний аналіз органів біологічних організмів із відповідними структурами Kubernetes, а також розглянуто можливість побудови суб'єктивних систем на основі його компонентів. Оскільки суб'єктивні системи можуть забезпечити більшу автономність компонентів Kubernetes у подальшому розвитку. Зроблено висновок про перспективність біонічного підходу як основи для підвищення стійкості та гнучкості Kubernetes у нестабільних умовах середовища.
Ключові слова: Застосування біонічних принципів в оркестрації Kubernetes, Kubernetes як динамічна та самоорганізована система, Аналогії між архітектурою Kubernetes і біологічними системами, Адаптація до нестабільного середовища через біонічний підхід, Суб'єктивні системи в Kubernetes

Kashchenko D. Utilization of Bionic Principles in the Deployment of Kubernetes Orchestration Systems. This article explores the possibility of applying bionic principles in the design of container orchestration deployment systems based on Kubernetes. The key components of the Kubernetes architecture are considered as an example of a complex dynamic system characterized by self-regulation, scalability, and adaptability. A modulation of the deployment of Kubernetes components is presented. Analogies are drawn between the functional modules of Kubernetes and the structures of living organisms, which makes it possible to identify similarities in organizational principles and functioning.
The study proposes the application of bionic concepts such as homeostasis, emergent behavior, and subjective systems, and outlines their implementation through modern technologies — in particular KEDA, Prometheus, Grafana, and Kubecost. Examples of configurations are provided to demonstrate the reactivity of the system to external events and workload changes, as well as its ability to optimize resource usage. Special attention is given to the prospects of Kubernetes evolving into an adaptive and self-protective platform through integration with machine learning methods and bio-inspired algorithms.
A comparative analysis of the organs of biological organisms and the corresponding structures of Kubernetes is conducted, along with an examination of the potential for building subjective systems based on its components. Since subjective systems can provide greater autonomy to Kubernetes components in the future, the study concludes that the bionic approach is a promising foundation for enhancing the resilience and flexibility of Kubernetes under unstable environmental conditions.
Keywords: Application of bionic principles in Kubernetes orchestration, Kubernetes as a dynamic and self-organizing system, Analogies between Kubernetes architecture and biological systems, Adaptation to unstable environments through a bionic approach, Subjective systems in Kubernetes.

У сучасних умовах розвитку інформаційних технологій дедалі більшої актуальності набувають системи, що здатні до самовідновлення, адаптації та ефективного розподілу ресурсів. Однією з таких систем є Kubernetes – платформа для оркестрації контейнерів, яка вже має низку властивостей, подібних до біологічних організмів. З іншого боку, біоніка – це наука, що вивчає принципи функціонування живих організмів та адаптує їх у технічні системи [3, 7, 8].

У цій статті розглянуто можливість застосування біонічних принципів для вдосконалення систем розгортання та оркестрації на основі Kubernetes. Але спочатку розберемо, що таке Kubernetes.

Kubernetes – це проект з відкритим вихідним кодом, призначений для управління кластером контейнерів Linux як єдиною системою [5]. Kubernetes керує та запускає контейнери Docker на великій кількості хостів, а також забезпечує спільне розміщення та реплікацію великої кількості контейнерів. Проект був започаткований Google і тепер підтримується багатьма компаніями, серед яких Microsoft, RedHat, IBM та Docker [1, 2].

Стаття переслідує три цілі. Якщо ви користуєтеся контейнерами Docker, виникає питання, як масштабувати та запускати контейнери одночасно на великій кількості хостів Docker, а також як виконувати їх балансування.[5] У статті пропонується використовувати високорівневий API, що визначає логічне групування контейнерів, дозволяючи визначати пули контейнерів, балансувати навантаження, а також задавати їх розміщення. Також у статті пропонується використати концепцію суб'єктивних систем та методи Kubernetes для побудови кращих відмовостійких систем.

Для початку розберем будову Kubernetes:

- Nodes: Нода – це машина в кластері Kubernetes.
- Pods: Под – це група контейнерів з загальними розділами, які запускаються як єдине ціле.
- Replication Controllers: Контролер реплікації - гарантує, що певна кількість «реплік» pod-ів буде запущена в будь-який момент часу.
- Services: Сервіс у Kubernetes – це абстракція, яка визначає логічно об'єднаний набір pod-ів і політику доступу до них.
- Volumes: Розділ – це директорія, можливо, з даними в ній, яка доступна в контейнері.
- Labels: Мітка – це пари ключ/значення, які прикріплюються до об'єктів, наприклад, pod-ів. Labels можуть бути використані для створення та вибору наборів об'єктів.
- Kubectl Command Line Interface: kubectl – інтерфейс командного рядка для управління Kubernetes.

Працюючий кластер Kubernetes включає агента, запущеного на нодах (kubelet), і компоненти майстра (API, планувальник тощо), поверх рішення з розподіленим сховищем.

Nodes Kubernetes. При погляді на архітектуру системи ми можемо розбити її на сервіси, які працюють на кожній ноді, і сервіси рівня управління кластером. На кожній ноді Kubernetes запускаються сервіси, необхідні для управління нодою з боку майстра та для запуску додатків. Звісно, на кожній ноді запускається Docker.

Kubelet управляє pods, їх контейнерами, образами, розділами тощо.

Kube-Proxy. На кожній ноді запускається простий proxy-балансувальник. Цей сервіс запускається на кожній ноді та налаштовується в Kubernetes API. Kube-Proxy може виконувати просте перенаправлення потоків TCP і UDP (round robin) між набором бекендів.

Kubecost – це біллінговий інструмент для моніторингу та оптимізації витрат у Kubernetes. Він дозволяє відстежувати використання ресурсів (CPU, RAM, сховище, тощо) на рівні компонентів Kubernetes (подів, сервісів, простору імен) та розраховувати пов'язані з цим витрати. Kubecost надає інформацію про витрати в реальному часі, допомагає в оптимізації витрат та прийнятті обґрунтованих рішень.

Система управління Kubernetes розділена на кілька компонентів. Ці компоненти працюють разом, щоб забезпечити єдине представлення кластера.

- etcd: Стан майстра зберігається в екземплярі etcd. Це забезпечує надійне зберігання конфігураційних даних та своєчасне сповіщення інших компонентів про зміну стану.
- Kubernetes API Server: Kubernetes API забезпечує роботу API-сервера. Він призначений для того, щоб бути CRUD сервером з вбудованою бізнес-логікою, реалізованою в окремих компонентах або в плагінах.
- Scheduler: Scheduler прив'язує не запущені Pods до Nodes через виклик /binding API. Scheduler має підтримку множинних scheduler-ів і користувацьких scheduler-ів.
- Kubernetes Controller Manager Server: Усі інші функції рівня кластера представлені в Controller Manager.

Для прикладу налаштування була обрана Ubuntu-server 14.10 як найбільш проста для прикладу та, в той же час, дозволяючи продемонструвати основні параметри налаштування кластера.

Вимоги для запуску:

1. На всіх нодах встановлений Docker + і bridge-utils.
2. Всі машини пов'язані одна з одною, доступ до інтернету не потрібен.
3. На всі ноди можна увійти без введення логіна/пароля, з використанням SSH-ключів.

Установлюємо програмне забезпечення на ноди. Додаткових налаштувань Docker після установки не потрібно, оскільки це буде виконане скриптом установки Kubernetes. Виконуємо додавання SSH-ключів. Виконуємо установку Kubernetes. Налаштовуємо Kubernetes повністю виконується перед установкою через конфігураційні файли. Для запуску сервісу необхідно підготувати Docker контейнер, на основі якого буде створено сервіс. Обов'язковими складовими

сервісу є Replication Controller та Service. Запуск сервісу можливий вручну, або за допомогою конфігураційних файлів. Для цього способу запуску необхідно створити конфігураційний файл для Replication Controller'a та Services. Kubernetes приймає конфігураційні файли у форматах yaml і json.

Є кілька важливих моментів, які виникають під час налаштування системи. Вони пов'язані з роботою kube-proxy, того самого модуля, який дозволяє перетворити розрізнений набір елементів у сервіс.

Для прикладу використаємо iptables-save:

```
-A PREROUTING -j KUBE-PORTALS-CONTAINER
-A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER
-A OUTPUT -j KUBE-PORTALS-HOST
-A OUTPUT ! -d 127.0.0.0/8 -m addrtype --dst-type LOCAL -j DOCKER
-A POSTROUTING -s 10.1.42.0/24 ! -o docker0 -j MASQUERADE
-A KUBE-PORTALS-CONTAINER -d 11.1.0.2/32 -p tcp -m comment --comment "default/kubernetes:" -m tcp --dport 443 -j REDIRECT --to-ports 47041
-A KUBE-PORTALS-CONTAINER -d 11.1.0.1/32 -p tcp -m comment --comment "default/kubernetes-ro:" -m tcp --dport 80 -j REDIRECT --to-ports 58240
-A KUBE-PORTALS-HOST -d 10.1.0.2/32 -p tcp -m comment --comment "default/kubernetes:" -m tcp --dport 443 -j DNAT --to-destination 172.16.67.68:47041
-A KUBE-PORTALS-HOST -d 10.1.0.1/32 -p tcp -m comment --comment "default/kubernetes-ro:" -m tcp --dport 80 -j DNAT --to-destination 172.16.77.58:58240
```

Усі запити до IP-адреси сервісу, які потрапляють в iptables, перенаправляються на порт який слухає kube-proxy. У зв'язку з цим виникає одна проблема - Kubernetes сам по собі не вирішує проблему зв'язку з користувачем. Тому зараз це вирішується зовнішніми засобами, наприклад: gcloud – платна розробка від Google, bgp – за допомогою анонсування підмереж.

Біоніка пропонує ряд концепцій, які можна застосовувати до розподілених обчислювальних систем. Зокрема, принципи самоорганізації, адаптації, емерджентної поведінки та гомеостазу. У контексті Kubernetes це проявляється у вигляді контролерів, які відстежують стан системи, автоматично масштабують служби та забезпечують їхню стабільність (таблиця 1).

Таблиця 1. Аналогії між Kubernetes і біологічними системами

Біологічна система	Компонент Kubernetes	Аналогія
Клітини	Pod	Основна функціональна одиниця
Органи	Service	Функціональні зв'язки між компонентами
Нервова система	Scheduler / Controller Manager	Координація дій
Імунітет	Liveness/Readiness Probes	Виявлення і лікування помилок
Мозок	Control Plane	Централізоване управління

Одним із прикладів є використання системи масштабування на події (KEDA), яка працює подібно до реакції організму на зовнішні подразники.[4] Системи моніторингу, як-от Prometheus і Grafana, виконують роль сенсорних систем (Alertmanager).[6]

Використання біонічних принципів у побудові інфраструктур на основі Kubernetes відкриває нові підходи до адаптивного, стійкого та самовідновлюваного управління. Принципи, запозичені з біології, не лише дозволяють краще зрозуміти наявну архітектуру Kubernetes, але й сприяють розробці інноваційних моделей, здатних ефективно функціонувати в умовах мінливого навантаження та зовнішніх впливів.

Наведемо приклад конфігурації об'єкта `ScaledObject` для KEDA, який реагує на метрику Prometheus (`http_requests_total`). Цей об'єкт масштабує кількість реплік залежно від інтенсивності HTTP-запитів, що є аналогією до реакції нервової системи на зовнішні подразники:

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
```

```
metadata:
  name: keda-scaledobject
spec:
  scaleTargetRef:
    name: my-deployment
  triggers:
  - type: prometheus
    metadata:
      serverAddress: http://prometheus.monitoring.svc:9090
      metricName: http_requests_total
      threshold: '100'
      query: sum(rate(http_requests_total[2m]))
```

У межах розгляду біоінспірованих підходів до побудови IT-систем доцільно розглянути ще низку концепцій:

•Гомеостаз: у біології – це здатність підтримувати внутрішню стабільність. У Kubernetes – це автоматичне балансування навантаження, автоматичний перезапуск невдалих Pod-ів, використання реплік для підвищення стійкості.[2]

• Еволюційність: біологічні системи змінюються у відповідь на навколишнє середовище. Сучасні CI/CD-підходи з GitOps дозволяють автоматично адаптувати систему до нових версій програмного забезпечення, тестуючи і обираючи найкращі варіанти.[9]

• Взаємодія мікросервісів як екосистеми: подібно до симбіозу в природі, різні сервіси взаємодіють, доповнюючи одне одного, забезпечуючи узгоджену поведінку всієї системи.

У сучасних дослідженнях з оркестрації контейнерів зростає інтерес до застосування біоінспірованих алгоритмів, таких як:

• Генетичні алгоритми: використовуються для оптимізації розміщення Pod'ів у кластері.[9] Але саме, генетичний алгоритм в тому форматі якій використовується зараз, моделює процеси філогенезу. Тому пропонується використати концепцію суб'єктивних систем, яка дозволяє моделювати процеси онтогенезу для кращої оптимізації процесів Kubernetes.
• Ройовий інтелект (Swarm Intelligence): дозволяє враховувати локальні метрики для прийняття глобальних рішень про масштабування або балансування.

• Алгоритми, засновані на імунній системі: дозволяють ідентифікувати аномалії у поведінці системи та реагувати на них до того, як відбудеться збій.

Приклад autoscaling на основі CPU:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: cpu-scaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 60
```

Такий підхід (показаний у Таблиці 1) не враховує всіх залежностей біологічних моделей, які можна застосувати у відмовостійких системах. Тому що, початкова взаємодія в організмах починається з клітини, яка безпосередньо взаємодіє з середовищем. Крім того дана аналогія показує,

що Kubernetes не може бути організмом, тому що відсутні багато потрібних органел і зв'язків з функціями потрібними для автономного та відмовостійкого існування. Як мінімум бракує механізмів для повної безпосередньої взаємодії із зовнішнім середовищем та внутрішньою акумуляцією ресурсів. Тому і пропонується моделювати біологічні процеси в Kubernetes, саме з використанням концепції суб'єктивних систем для збільшення відмовостійкості. А саме, спробуємо побудувати відмовостійку структуру Kubernetes шляхом застосування концепції суб'єктивних систем та її логічної моделі. Але для цього підемо від зворотного, спробуємо провести аналіз функціоналу Kubernetes відносно побудови повного функціоналу суб'єктивної системи (Таблиця 2).

В даному методі розгортанні суб'єктивної системи повинні заздалегідь бути визначенні: що саме, в навколишньому середовищі, буде використовуватися системою, як елемент для своєї побудови та енергетичного живлення, що саме, буде внутрішньою та зовнішньою структурою системи, та як буде накопичуватися та розподілятися ресурс при необхідності. Що саме і як саме буде допомагати змінювати систему для пристосування до навколишнього середовища.

Для прикладу, системі потрібні електроенергія, зв'язок (інтернет), гроші (для закупівлі обладнання та послуг) в обмін за надійний сервіс для клієнтів (забезпечення захисту, цілісності та доступності даних). Для взаємодія з навколишнім середовищем суб'єктивній системі потрібні: структурна одиниця побудови, структурна одиниця енергії, структурна одиниця обміну (гроші/дані). Фактично, запити клієнтів на ресурси і є основою існування такої системи, але в подальшому її можна переорієнтувати на безпосередню взаємодію з іншими процесами навколишнього середовища. Але саме в даному випадку, використання таких підходів збільшить відмовостійкість та автономність систем побудованих за допомогою Kubernetes.

Таблиця 2. Аналіз можливого використання компонентів Kubernetes для побудови суб'єктивних систем.

Суб'єктивна система	Компонент Kubernetes	Аналогія та недоліки
ЗПМ (захисно-перетворювальний механізм)	Pod, Service, Control Plane Білінгова система Kubecost (гроші системи від клієнтів), Gcloud, bgp	Є - основна функціональна одиниця, централізоване управління. Відсутні - механізми перетворення, або механізм придбання та постачання серверних компонентів, компонентів зв'язку та інших компонентів системи для потреб побудови системи. Відсутні - механізми отримання електроенергії (закупівлі чистої, або вироблення через генератори чи сонячні панелі) для енергетичного безвідмовного живлення системи. Відсутні - механізми побудови та повного контролю ліній зв'язку (інтернет чи інших) для роботи з навколишнім середовищем. Відсутня нейромережа для розпізнавання, категоризації, спілкування з клієнтами та прогнозування розвитку самої системи
РНМ (розподільно-накопичувальний механізм**)**	Pod, Service, Control Plane	Є - основна функціональна одиниця, централізоване управління. Відсутні банківські рахунки для системи акумуляції запасу грошей, які в подальшому можуть бути використані для придбання та монтажу нових компонентів серверів чи ліній зв'язку. Kubecost орієнтований тільки на клієнтів. І не має можливості накопичувати та розподіляти гроші для потреб самої системи Kubernetes. Відсутні - контролери, акумулятори, запаси палива, резервні лінії зв'язку та інші ресурси накопичення енергії

BIM (видозмінний механізм)	Scheduler / Controller Manager, Liveness/Readiness Probes, Control Plane .	Є - Координація дій, Виявлення і лікування помилок, Централізоване управління та розгортання. Відсутня – нейромережа (LLM + bionic dataset) для зміни коду самої системи за запитами ЗПМ та РНМ для потреб самої системи.

Застосування біоінспірованих методів у Kubernetes може бути розширене через використання машинного навчання та штучного інтелекту.[10] Концепція суб'єктивних систем, прогнозування навантаження, динамічне виявлення відхилень, самонавчальні системи управління – усе це дозволить створювати по-справжньому розумні кластери, здатні до самопідтримки, самозахисту та самонастроювання. У майбутньому можливе створення та масштабування повністю автономних відмовостійких кластерів, які будуть керуватись лише за допомогою політик та цільових параметрів, аналогічно до біологічних організмів, які функціонують на основі внутрішніх програм та зовнішніх умов середовища.

Список бібліографічного опису

1. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). *Borg, Omega, and Kubernetes*. Communications of the ACM, 59(5), 50–57.
2. Hightower, K., Burns, B., & Beda, J. (2017). *Kubernetes: Up and Running*. O'Reilly Media.
3. Turing, A. M. (1952). *The chemical basis of morphogenesis*. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 237(641), 37–72.
4. KEDA – Kubernetes-based Event Driven Autoscaling. https://keda.sh
5. The Kubernetes Documentation. https://kubernetes.io/docs/
6. Prometheus Documentation. https://prometheus.io/docs/
7. Базюк, В. П., Мельник, Т. А. (2020). Біонічний підхід до моделювання складних систем. *Вісник НТУУ "КПІ"*, Серія: Автоматика і управління, 66(1), 45–52.
8. Шкуро, В. А. (2019). Біоніка в сучасному технологічному дискурсі. *Сучасні інформаційні технології у сфері безпеки та оборони*, №4, 61–67.
9. Zolotukhin, V., & Bondarenko, I. (2021). Application of Bionic Principles in Intelligent Cloud Systems. *Computer Science and Information Technologies*, 2(34), 15–22.
10. Taleb, N. N. (2012). *Antifragile: Things That Gain from Disorder*. Random House.

References

1. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. Communications of the ACM, 59(5), 50–57.
2. Hightower, K., Burns, B., & Beda, J. (2017). Kubernetes: Up and Running. O'Reilly Media.
3. Turing, A. M. (1952). The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 237(641), 37–72.
4. KEDA – Kubernetes-based Event Driven Autoscaling. Retrieved from: https://keda.sh
5. The Kubernetes Documentation. Retrieved from: https://kubernetes.io/docs/
6. Prometheus Documentation. Retrieved from: https://prometheus.io/docs/
7. Bazyuk, V. P., & Melnyk, T. A. (2020). A Bionic Approach to Modeling Complex Systems. Visnyk NTUU "KPI". Series: Automation and Control, 66(1), 45–52.
8. Shkuro, V. A. (2019). Bionics in Modern Technological Discourse. Modern Information Technologies in the Sphere of Security and Defense, (4), 61–67.
9. Zolotukhin, V., & Bondarenko, I. (2021). Application of Bionic Principles in Intelligent Cloud Systems. Computer Science and Information Technologies, 2(34), 15–22.
10. Taleb, N. N. (2012). Antifragile: Things That Gain from Disorder. Random House.