Однако использование среднего значения всех соседей в качестве нового значения синхронизации является проблематичным, если смещения являются большими. Во время запуска узла аппаратный тактовый регистр инициализируется нулем, что, возможно, приводит к огромному смещению узлов, которые уже синхронизированы с сетью. Такое огромное смещение заставит все остальные узлы повернуть свои часы, что нарушает принцип причинности. Вместо этого, если узел узнает, что часы соседа находятся дальше, чем определенное пороговое значение, он переходит на значение часов соседей [4]. Используя этот механизм начальной загрузки, узел, присоединяющийся к сети, быстро синхронизируется с остальной частью сети. В худшем случае может потребоваться до O(D) времени, чтобы все узлы свободно синхронизировались, где D – диаметр сети.

Список использованных источников:

1. N. Burri, P. von Rickenbach, and R. Wattenhofer. Ultra-low Power Data Gathering in Sensor Networks, 2007.
2. M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol, 2004.
3. J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks, 2004.
4. R. Fan and N. Lynch. Gradient Clock Synchronization, 2004.
5. T. K. Srikanth and S. Toueg. Optimal Clock Synchronization, 1987.

Гонтаренко І.В.
студент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МОДЕЛЬ БАЛАСУВАННЯ НАВАНТАЖЕННЯ У ВІРТУАЛЬНОМУ ОБЧИСЛЮВАЛЬНОМУ КЛАСТЕРІ НА ОСНОВІ КОНТЕЙНЕРІВ ДОДАТКІВ

В даній статті розглянуто практичне використання розподіленого обчислювального середовища, яке дозволить конфігурувати надані обчислювальні ресурси у відповідності до вимог програми, а також систематизація та порівняння поширених методів масштабування додатків та алгоритмів балансування навантаження.

У сучасному світі людина з усіх сторін оточена обчислювальними пристроями. Потужності обчислювальної техніки протягом всього періоду її розвитку постійно зростають, але навіть незважаючи на закон Мура [1], завжди існували та існують завдання, яким існуючих потужностей поодиноких комп'ютерів виявляється недостатньо. Саме тому останнім часом велику популярність та важливу роль відіграють високонавантажені системи

обчислення. Ефективним вирішення таких завдань є масштабування додатків, балансування навантаження та об'єднання комп'ютерів у різні обчислювальні мережі або кластери. Для забезпечення узгодженої роботи вузлів обчислювальної мережі на стороні провайдера використовується спеціалізоване проміжне програмне забезпечення, що забезпечує моніторинг стану обладнання і програм, балансування навантаження, забезпечення ресурсів для вирішення завдання. Також актуальною проблемою є вибір засобу балансування навантаження, який би найкращим чином відповідав задачам створення конкретного сервісу в конкретних умовах.

В наш час популярність хмарних технологій зростає з кожним днем а разом і з цим потреба в масштабуванні додатків та балансуванні навантаження, яке впливає на підвищення ефективності високонавантаженої системи. Саме програмне забезпечення на стороні провайдера забезпечує узгоджену роботу вузлів, моніторинг стану додатків та обладнання, балансування навантаження та інше. Актуальність цієї роботи полягає у виборі методів масштабування додатків та алгоритмів балансування навантаження у віртуальному кластері на основі контейнерів конкретного додатку з певним задачами в конкретних умовах.

Поняття віртуалізація має велику кількість визначень. В нашому випадку розглядаються інструменти та підходи, які дозволяють абстрагуватися від апаратної основи обчислювальної системи. Робота з системою на більш високому рівні абстракції дозволяє більш гнучко працювати з ресурсами. Віртуалізація дозволяє, як запускати додатки в ізольованому середовищі, так і запускати сервера на машині. Це зручний інструмент для запуску безлічі додатків на одному вузлі, розділяючи його ресурси між додатками, як рівномірно, так і нерівномірно між собою.

Віртуалізація допомагає вирішити такі проблеми як:

- запуск додатків різних версій;
- запуск додатків, які конфліктують між собою;
- конфлікт залежностей; безпека.

На даний момент існує декілька засобів для віртуалізації, розглянемо апаратну віртуалізацію та віртуалізацію на рівні ОС [2].

Дані будуть агрегуватися методом запитів на засіб для управління контейнерами з певним інтервалом часу. Даний засіб буде збирати статистику з усіх активних додатків, і повертати такі дані.

Docker надає API для отримання статистики використання обчислювальних ресурсів контейнера. Дані про запити на контейнер, часу відповіді, будуть агрегуватися з логів проксі-сервера.

Для контейнера буде обчислюватися середнє використання CPU в годину, середнє використання RAM в годину, середня пропускна здатність контейнера, середній час відповіді, а також максимальну кількість відповідей за відведений час при поточній навантаженні [3].

На підставі вибірки з логів буде проводитися розрахунок максимального числа відповідей за певний час при середньому часу відповіді на користувача.

Розрахунок буде здійснюватися за формулою:

$$X = \{ x1, x2, \dots, x_N \} - \text{вибірка}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{E} x_i$$

$$R_{max} = \frac{\mathrm{T}}{\bar{x}}$$

де T – це розрахунковий час на відповідь. Таким чином, чим більше середній час відповіді користувачу, тим менше сервер зможе обробити запитів за відведений час [4]. В якості розрахункового часу на відповідь буде взята 1 секунда.

На підставі загальної кількості запитів за годину, пропускна здатність системи буде, розраховується за формулою:

$$R = \{ r1, r2, \dots, r_N \} - \text{вибірка}$$

$$R = \sum_{i=1}^{N} r_i$$

$$T = \frac{\mathrm{R}}{60}$$

Це дозволить користувачеві оцінювати пропускну здатність системи і масштабувати її в залежності від навантаження, що збільшується, для забезпечення комфортного користування системою. Також дані метрики можуть показати користувачеві що кількість обчислювальних ресурсів які він вибрав для свого застосування є зайвими і він може їх скоротити [5].

Після розробки прототипу і налаштування ресурсів, необхідно було провести перевірку системи. Для цієї мети було проведено ряд обчислювальних експериментів з різними системними параметрами. Далі проводився запуск контрольних завдань по завершенню роботи яких було отримано час виконання кожної програми. Результати серії експериментів наведені нижче.

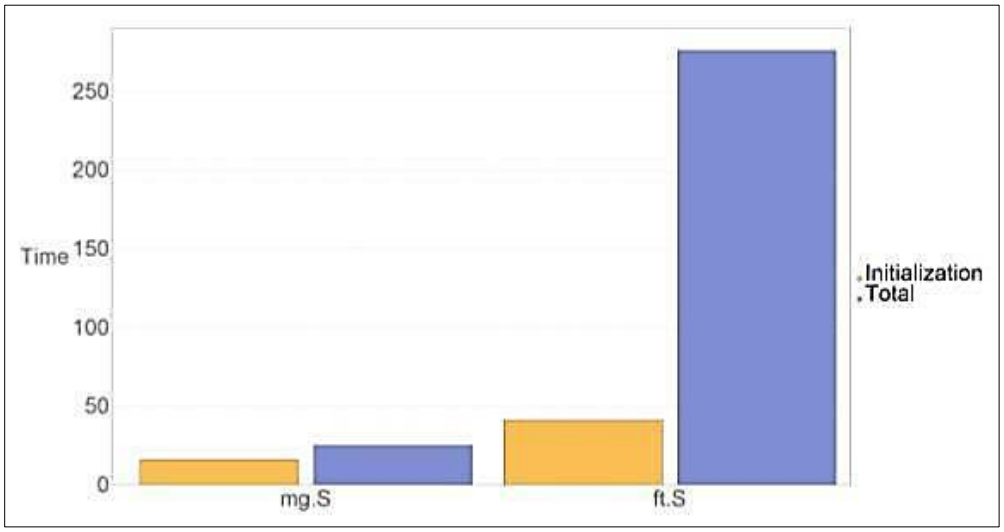

Рис. 1. Пам’ять 256 MB, мережа 100 kbit/s

Використовуючи дані обмеження відпрацювали тільки два завдання. Інші ж не змогли запуститися при даних параметрах.

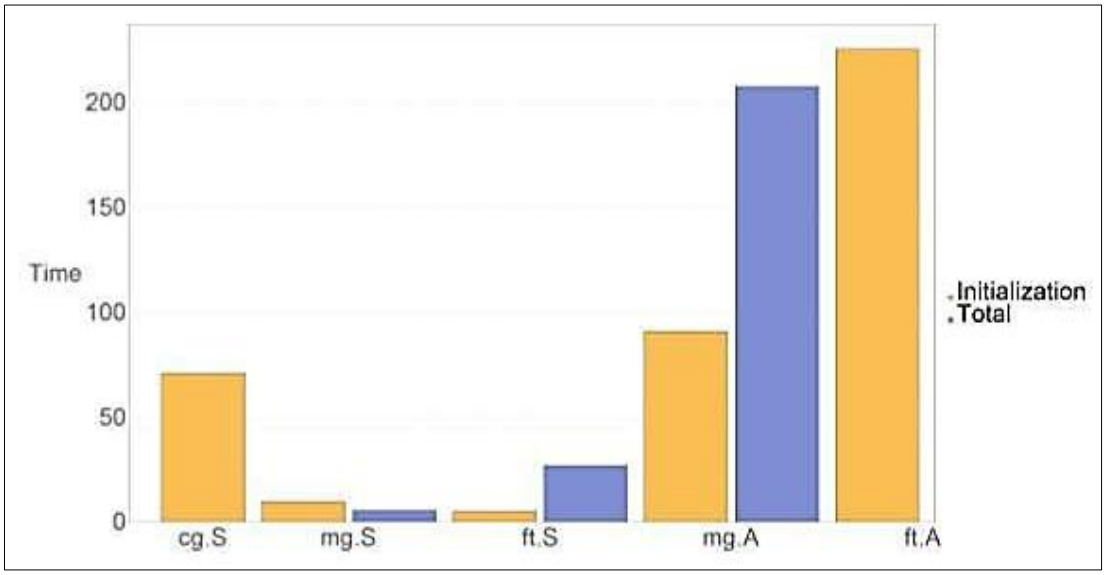

Рис. 2. Пам'ять 512 MB, мережа 1024 kbit/s

При обмеженнях використаних на рис. 2 запустилися і відпрацювали всі завдання класу S. Однак при запуску завдань класу A успішно закінчила виконання тільки завдання mg. При цьому завдання ft даного класу виконала ініціалізацію. Вирахувала всі ітерації і на етапі фінального підрахунку даних перестала відповідати.

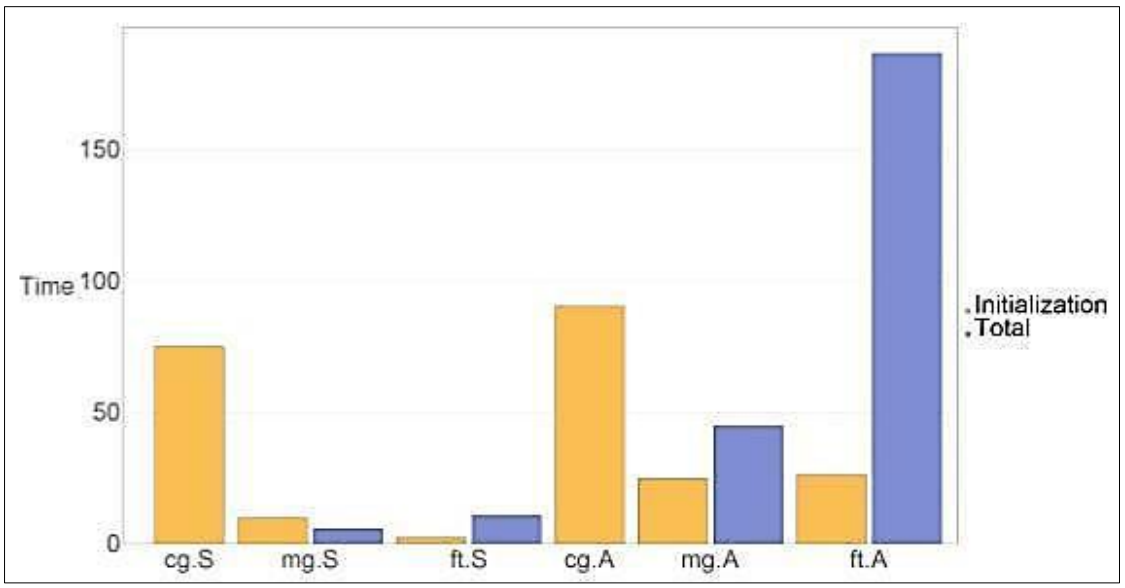

Рис. 3. Пам'ять 1024 MB, мережа 10240 kbit/s

При цьому наборі параметрів всі завдання успішно відпрацювали і тепер є можливість підбити попередні підсумки за всіма трьома формами. Як видно з гістограм, що вводяться обмеження впливають на час виконання програм, зокрема це можна помітити на прикладі завдання ft розміру S. При першому обмеження швидкість її виконання склала 276.06 секунд, при другій

конфігурації 26.98, при третьої 10.91. Варто зазначити, що не всі завдання запустилися при заданих змінах наприклад завдання ft розміру A на першій конфігурації не відкривається, а на другий конфігурації запустилася, виконала етап ініціалізації і перестала відовідати.Теперь є можливість провести більш детальний аналіз і підібрати оптимальні параметри для кожної з задач. Однак перед цим був проведений експеримент паралельного запуску двох завдань.

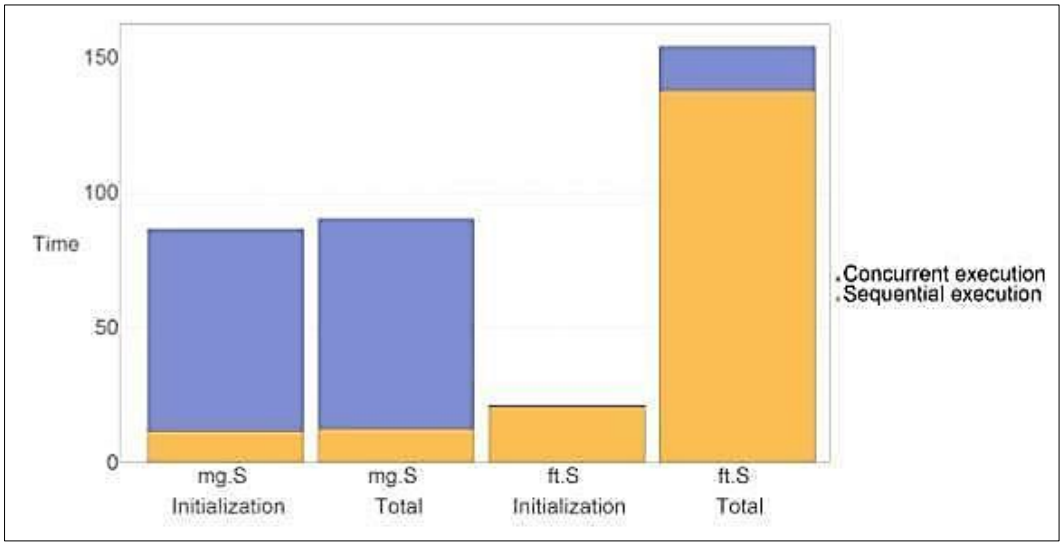

Рис. 4. Пам’ять 512 MB, мережа 200 kbit/s, одночасний запуск на однакових контейнерах

Обмеження по пам'яті 512 MB, обмеження по мережі 200 kbit/s. На гістограмі представлені два випадки запуску, послідовного і паралельного, на одних і тих же контейнерах. Як можна бачити з гістограми, час виконання програм помітно відрізняється, в порівнянні з послідовним запуском. Нижче представлений модуль різниці послідовного виконання і паралельного в разі, коли завдання виконувалися на різних контейнерах.

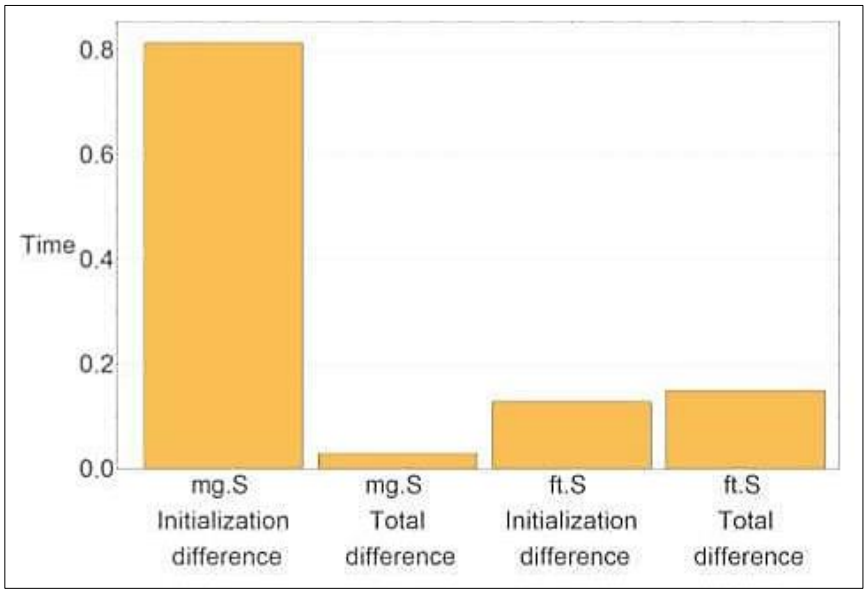

Рис. 5. Пам’ять 512 MB, мережа 200 kbit/s, одночасний запуск на однакових контейнерах

Як видно з рис. 5, крім вирішення основного завдання було досягнуто властивість ізольованості. На жаль, через те, що експерименти проводилися на базі віртуальних машин, тому мають місце накладні витрати в обчисленнях. Далі була взята завдання ft розміру S і проведено ряд експериментів, щоб обчислити її оптимальні параметри запуску. Спершу була зафіксована пам'ять в 100 MB і в якості змінного параметра позначена пропускна здатність мережі.

Як видно з наведених вище графіків, поведінка завдань схожа між собою. Однак при емуляції сценарію, коли додатки ділять між собою ресурси графік змінюється на рахунок осі абсцис. Тим самим для більш ефективного запуску завдання потрібно більше ресурсів. Крім цього в тестовому режимі була додана інтеграція з базою даних.

Аналізуючи отримані результати, можна визначити оптимальні параметри для наступних завдань:

	Пам’ять (MB)	Смуга проп. (MB/s)
mg.S	12	3
ft.S	16	1
mg.A	70	7
ft.A	70	13

Зміни параметрів впливають на час виконання завдань. Крім цього є можливість запускати кілька завдань використовуючи одні й ті ж ресурси. В відміну від класичної моделі кластера в даному випадку завдання виконуються ізольовано і практично не впливають один на одного. Варто зазначити, що експерименти проводилися в хмарі на віртуальних машинах, тобто результати менш точні, в порівнянні з результатами, отриманими на апаратній основі. Однак наявність розподіленої системи вузлів дозволило наблизити процес проектування і прототипування до реальності. Зокрема було виявлено ситуація, коли політика безпеки мережі, в якій знаходиться ресурс, забороняє доступ. Аналізуючи використані завдання можна прийти до висновку, що у кожної з задач присутній мінімальний поріг пам'яті. Якщо завдання надати пам'яті менше, то у неї не вистачить ресурсів для коректного завершення роботи. При цьому, якщо пам'яті достатньо, то подальше збільшення пам'яті не впливає на продуктивність. При цьому вплив, який чиниться зміною параметрів пропускної здатності мережі більш помітно. В рамках даної роботи розглядалася невеликі по витратам ресурів завдання. Це дозволило виробити більшу кількість запусків, тим самим більш детально вивчити поведінку програми. Поставлена мета була досягнута частково, незважаючи на те, що поставлені завдання були виконані і є робочий прототип не на всі аналізовані параметри були введені обмеження. В даний момент не вирішена задача обмеження CPU, зважаючи на особливості ресурсів.

Таким чином, у статті було розглянуто практичне використання розподіленого обчислювального середовища, яке дозволить конфігурувати

надані обчислювальні ресурси у відповідності до вимог програми, а також систематизація та порівняння поширених методів масштабування додатків та алгоритмів балансування навантаження.

Використання такої системи надає можливості відтворювати безліч екземплярів конкретного додатку з різними конфігураціями продуктивності, а також в систематизації та порівнянні методів масштабування та алгоритмів балансування навантаження.

Список використаних джерел:

1. Moore’s law. – Режим доступу: https://en.wikipedia.org/wiki/Moore's_law
2. Armbrust M. et al. A view of cloud computing //Communications of the ACM. – 2010. – Т. 53. – №. 4. – С. 50-58.
3. Turnbull J. The Docker Book. – Lulu. com, 2014.
4. Merkel D. Docker: lightweight linux containers for consistent development and deployment // Linux Journal. – 2014. – Т. 2014. – №. 239. – С. 2.
5. Fishman A. et al. HVX: Virtualizing the Cloud //HotCloud. – 2013.

Іванківа В.І.
студентка;
Долженкова О.В.
кандидат технічних наук, доцент,
старший науковий співробітник,
Дніпровський національний університет імені Олеся Гончара

ДИНАМІКА ТРАВМАТИЗМУ В УКРАЇНІ

Як показує світова практика основною гарантією стабільності та якості будь-якого виробництва є перш за все безпека праці. Вона виступає запорукою високої продуктивності роботи, адже лише безпечні техніка, прийоми та обладнання дають можливість робітнику зосередитись на продуктивності та якості виконуваної роботи. До того ж, відсутність нещасних випадків позначається не лише на професійній активності працюючих, а й на моральному кліматі в колективі, а отже і на ефективності та продуктивності праці. Проте, на жаль, не всі роботодавці виконують свої обов’язки в сфері забезпечення гідних умов праці та не застосовують міжнародні стандарти безпеки, що призводить до поширення практики виробничого травматизму [1].

Виробничий травматизм вже давно став актуальною питанням у всіх країнах світу, в тому числі і в Україні. Протягом усього ХХ ст. зростала кількість нещасних випадків на виробництві зі смертельним наслідком, з переходом на інвалідність, з тимчасовою втратою працездатності. Сьогодні в економічно розвинених країнах світу травми займають провідне місце серед причин смертності населення, причому серед осіб працездатного віку.