Results. The proposed scheme was implemented using compute shaders and tested on a benchmark with up to 75,000 fluid particles. The GPU-based implementation completes one full simulation step in approximately 10 ms. For comparison, a multi-threaded CPU implementation requires about 250 ms, and a single-threaded CPU implementation requires about 690 ms for an equivalent particle count. The reported speedup factor exceeds 25× compared to the multi-threaded CPU baseline and 65× compared to the single-threaded baseline, confirming that the dominant share of the gain comes from the parallel neighborhood search [5].

Pre-allocation of the hash table with size equal to the number of particles eliminates allocation overhead and stabilizes per-step timings. The locality enforced by sorting reduces irregular memory access in the subsequent density and pressure passes, which is particularly important for IISPH-based incompressible solvers where multiple Jacobi iterations are executed per step.

Conclusions. This paper presented a GPU-adapted neighborhood search procedure for SPH simulations based on compact hashing and Bitonic Sort. The combination of pre-allocated hash tables, structured parallel sorting and cell-based neighbor queries enables linear scaling with the number of particles and real-time performance for ensembles of tens of thousands of particles. Future work includes adaptive hash table sizing and extension to multi-GPU configurations.

1. Monaghan J. J. Smoothed particle hydrodynamics. Reports on Progress in Physics. – 2005. – Vol. 68. – P. 1703–1759.
2. Ihmsen M., Akinci N., Becker M., Teschner M. A parallel SPH implementation on multi-core CPUs. Computer Graphics Forum. – 2011. – Vol. 30, No. 1. – P. 99–112.
3. Ihmsen M., Cornelis J., Solenthaler B., Horvath C., Teschner M. Implicit incompressible SPH. IEEE Transactions on Visualization and Computer Graphics. – 2014. – Vol. 20, No. 3. – P. 426–435.
4. Batcher K. E. Sorting networks and their applications. Proceedings of the AFIPS Spring Joint Computer Conference. – 1968. – Vol. 32. – P. 307–314.
5. Hrytsyshyn O., Trushevskyy V. Application of IISPH for incompressible fluid dynamics simulation. Вісник Львівського університету. Серія прикладна математика та інформатика. – 2024. – Вип. 33. – С. 55–68.

Автоматизація процесів інтеграції та розгортання вебзастосунків

УДК 004.42

Петришин Ярослав[1], Мудрик Іван[2]

Тернопільський національний технічний університет імені Івана Пулюя,
[1]aroslav_petryshyn1608@tntu.edu.ua, [2]i1mudryk@tntu.edu.ua

Сучасні вебзастосунки відрізняються складною багаторівневою архітектурою, яка включає серверну частину, клієнтський застосунок, бази даних та сервіси реального часу. За таких умов ручне тестування й розгортання програмного продукту стає неефективним і ризикованим. Методологія CI/CD (Continuous Integration / Continuous Deployment) забезпечує автоматизацію процесів перевірки, складання та доставки змін до виробничого середовища,

суттєво підвищуючи надійність і швидкість випуску нових версій [1]. Сучасні вебзастосунки еволюціонували до рівня розподілених cloud-native систем, що вимагає переходу від класичних моделей CI/CD до парадигми DevSecOps. У 2026 році автоматизація розгортання вже не обмежується лише доставкою коду, а включає обов'язкові етапи динамічного аналізу безпеки (DAST), сканування образів на вразливості та автоматизоване керування інфраструктурою. Це дозволяє нівелювати ризики, пов'язані з людським фактором, та забезпечити високу доступність сервісів у високонавантажених середовищах.

Типовий процес автоматизації складається з кількох послідовних етапів: статичний аналіз коду (лінтинг), автоматичне тестування (юніт-, інтеграційні та end-to-end тести), складання артефактів та їх доставка на цільовий сервер. Інструменти на кшталт GitHub Actions, GitLab CI або Jenkins дозволяють описати ці етапи декларативно у вигляді YAML-конфігурацій, що запускаються автоматично при кожному коміті або pull request. Це забезпечує швидкий зворотний зв'язок для розробника і унеможливлює потрапляння некоректного коду в основну гілку [1].

Контейнеризація засобами Docker є невід'ємною складовою сучасних процесів автоматизації розгортання, оскільки забезпечує відтворюваність середовища на всіх етапах — від локальної розробки до виробництва. Docker Compose дозволяє декларативно описати мультисервісну інфраструктуру, а збірка образів безпосередньо в процесі доставки усуває залежність від ручного налаштування сервера. У поєднанні з процес-менеджерами це забезпечує автоматичний перезапуск сервісів і мінімізує час простою [2].

Використання Docker та оркестрації залишається фундаментом відтворюваності. Проте акцент змістився на оптимізацію розміру образів та впровадження Infrastructure as Code (IaC) за допомогою Terraform або Pulumi. Це дозволяє декларативно описувати не лише сервіси (через Docker Compose), а й усю хмарну мережеву інфраструктуру, бази даних та системи кешування (наприклад, Redis), мінімізуючи розбіжності між середовищами розробки та продуктиву.

Дослідження ефективності автоматизованих процесів інтеграції та розгортання показують, що команди скорочують час від коміту до розгортання в середньому на 60–80% порівняно з ручними підходами. Крім того, частота виробничих інцидентів, пов'язаних із помилками розгортання, зменшується на 40–50% завдяки обов'язковому проходженню автоматичних перевірок на кожному етапі.

Аналіз впровадження сучасних автоматизованих платформ (базуючись на метриках DORA) показує, що перехід до повної автоматизації дозволяє досягти: збільшення частоти розгортання (Deployment Frequency) у 5–10 разів; скорочення середнього часу відновлення сервісу (MTTR) на 60%; зниження частки невдалих змін (Change Failure Rate) до рівня менше 5% завдяки жорстким автоматизованим фільтрам якості.

Особливої ваги набуває захист ланцюгів постачання програмного забезпечення (Software Supply Chain Security), оскільки за даними останніх досліджень 2025 року [2, 4], вразливості в інфраструктурі збірки є критичним фактором ризику для Cloud-Native систем.

Таким чином, впровадження автоматизованих процесів інтеграції та розгортання є критично важливою практикою для підтримки стабільності й масштабованості сучасних вебплатформ. Описані підходи є актуальними для широкого класу застосунків і можуть бути адаптовані відповідно до потреб конкретного проєкту [2]. Трансформація процесів інтеграції та розгортання у цілісну екосистему DevSecOps є критичною умовою для масштабованості вебплатформ. Актуальність дослідження полягає у переході від простого скриптування до створення інтелектуальних систем доставки, що здатні самостійно адаптуватися до навантажень та безпекових викликів.

1. Мачужак А. В. Дослідження методології DevOps для розробки та підтримки веб-застосунків : кваліфікаційна робота магістра. – Тернопіль : ТНТУ, 2023. – 114 с.
2. Спасітєлєва С. О. Безперервна інтеграція та безперервна доставка (CI/CD) як практика безпечної розробки ПЗ // Кібербезпека: освіта, наука, техніка. – 2023. – № 21. – С. 193–210.
3. Evolution of DevSecOps and Its Influence on Application Security: A Systematic Literature Review // MDPI: Applied Sciences. – 2025. – Vol. 13, No. 12. – P. 548–565.
4. Research Directions in Software Supply Chain Security // ACM Transactions on Software Engineering and Methodology. – 2025. – Vol. 34, No. 5. – P. 112–134.

Інструментальні засоби аналізу впливу характеристик комерційних SPAD-детекторів на стійкість протоколу BB84+decoy-state

УДК 004.056.55:535.14

Олексій Пирогов[1], Василь Різак[2]

Ужгородський національний університет,
[1]oleksii.pyrohov@uzhnu.edu.ua, [2]vrizak@uzhnu.edu.ua

QKD вийшов на промисловий рівень: китайська мережа CN-QCN (China Quantum Communication Network) охоплює понад 10 000 км волокна, 145 магістральних вузлів та 20 метромереж у 80 містах [1]. Транскордонний сегмент EuroQCI розгортається з 2026 р. З 20 квітня 2025 р. набрав чинності Закон № 4336-IX про кіберзахист державних інформаційних ресурсів, але не містить вимог до QKD-систем; галузевого стандарту на QKD в Україні немає. Вибір SPAD-детектора критично впливає на максимальну дальність каналу та параметри секретного ключа, але відкритий інструмент аналізу стійкості відсутній.

BB84 [2] із decoy-state розширенням [3] — найпоширеніший протокол комерційних QKD-систем на волокні. Утім, чи здатна така система генерувати секретний ключ і на якій відстані ще зберігається стійкий режим — визначає не протокол, а характеристики SPAD: η_d (квантова ефективність детектування), вакуумний yield Y_0 (зумовлений темновим рахунком), мертвий час, післяімпульсація, похибка оптичного вирівнювання e_{det} — саме вони задають