УДК 004.415.2:004.432.42:004.451

Швець Д.В., к.т.н., доцент,
Котов І.А. , д.т.н., професор,
Карабут Н.О., ст. викладач
Криворізький національний університет

МЕХАНІЗМИ ПОЛІМОРФНОЇ АДАПТАЦІЇ JAVA-ПАТЕРНІВ ПІД СПЕЦИФІЧНІ АРХІТЕКТУРИ ОПЕРАЦІЙНИХ СИСТЕМ

Концепція «Write Once, Run Anywhere» (WORA) сьогодні зазнає архітектурних змін, перетворюючись із механізму простої сумісності на складну парадигму поліморфної адаптації. Дилема між кросплатформною інваріантністю програмного коду та необхідністю досягнення нативної продуктивності в гетерогенних операційних середовищах вирішується через інтеграцію об'єктно-орієнтованих патернів у внутрішню структуру віртуальної машини Java (JVM).

Метою даної роботи є аналіз того, як класичні механізми поліморфізму та динамічного зв'язування використовуються для приховування складності системних викликів, забезпечуючи при цьому оптимальну взаємодію з ядрами розповсюджених операційних систем, зокрема Windows, Linux та macOS.

Центральну роль у процесі системної адаптації відіграє поліморфізм, який у контексті JVM виступає шаром абстракції над низькорівневими ресурсами. Через механізм динамічного зв'язування віртуальна машина ідентифікує специфіку операційного середовища безпосередньо під час виконання програми, що дозволяє підставляти відповідні реалізації системних інтерфейсів без втручання в основний код застосунку. Це забезпечує стабільність уніфікованого прикладного програмного інтерфейсу при одночасному використанні специфічних механізмів кожної системи, що простежується в еволюції структурних патернів проєктування в межах стандартних бібліотек JDK.

Зокрема, реалізація мережевого стеку та механізмів вводу-виводу в сучасній Java демонструє вдале використання патерна «Міст» (Bridge) [1], де абстракція каналу відокремлена від його системно-залежної імплементації. Це дозволяє розробнику оперувати універсальними об'єктами, тоді як на рівні ядра операційної системи задіюються принципово різні механізми, такі як epoll у Linux, kqueue у системах сімейства BSD або IOCP у середовищі Windows. Аналогічно, використання патерну Abstract Factory у файлових системах дозволяє інкапсулювати відмінності в ієрархіях шляхів та правах доступу, роблячи процес адаптації абсолютно прозорим для кінцевого

користувача. Еволюція цих патернів свідчить про поступовий відхід від статичної уніфікації на користь динамічної спеціалізації.

Окремим вагомим етапом у розвитку засобів адаптації є впровадження ініціатив у межах проєкту Panama, зокрема Foreign Function & Memory API. Ця технологія дозволяє вийти за межі традиційних обмежень Java Native Interface (JNI), пропонуючи натомість безпечні об'єктно-орієнтовані абстракції для прямої взаємодії з некерованою пам'яттю та системними бібліотеками мов C/C++. Важливим аспектом тут виступає здатність Java поліморфно адаптувати структури даних під специфічний порядок байтів та вирівнювання пам'яті, характерні для конкретних архітектур процесорів, що мінімізує обчислювальні витрати на маршалінг та забезпечує детерміновану поведінку системи в мультиплатформних конфігураціях.

Сучасні версії JDK, починаючи з релізу 17 та закінчуючи 21, вносять додаткові рівні інтелектуальної адаптації через механізми закритих типів та віртуальних потоків. Використання закритих ієрархій класів дозволяє JIT-компілятору виконувати агресивнішу оптимізацію системно-залежних патернів, оскільки заздалегідь визначений набір реалізацій для конкретних ОС спрощує аналіз потоків керування. Водночас віртуальні потоки радикально трансформують модель конкурентності, переносячи відповідальність за планування завдань із рівня ядра операційної системи на рівень об'єктно-керованих абстракцій JVM. Це дозволяє забезпечити масштабованість, де патерни багатопоточності адаптуються не до лімітів ядра ОС, а до потреб прикладної архітектури.

Слід констатувати, що еволюція об'єктно-орієнтованої парадигми в екосистемі Java відображає зсув у бік інтенсифікації взаємодії між високорівневими програмними абстракціями та низькорівневими архітектурами операційних систем. Цей процес вже не обмежується традиційним прагненням до ізоляції коду в межах віртуальної машини, а навпаки, спрямований на створення прозорого та безпечного доступу до апаратних ресурсів через розвинені механізми поліморфної адаптації. Саме ці механізми дозволяють зберегти концептуальну цілісність і сувору типізацію мови, одночасно розширюючи напрями її застосування в критично важливих, високонавантажених середовищах, де раніше домінували виключно мови системного програмування.

Список використаних джерел:

1. Chin S., Vos J., Weaver J. The Definitive Guide to Modern Java Clients with JavaFX. Berkeley, CA : Apress, 2024. 626 p. DOI: 10.1007/979-8-8688-0998-9