УДК 004.4
І. Яковенко, І. Мудрик, канд. техн. наук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ОРІЄНТОВАНИЙ НА ДАНІ ДИЗАЙН. АНАЛІЗ ТА ПЕРЕВАГИ

UDC 004.4
I. Yakovenko, st., I. Mudryk, ass.

DATA-ORIENTED DESIGN. ANALYSIS AND ADVANTAGES

Під час розробки програмного забезпечення часто виникають проблеми продуктивності, через які час виконання програми для різних завдань зупиняється на неприпустимо довгий час. Причин цього може бути кілька, найочевиднішою з яких є неоптимізовані алгоритми виконання, але це не завжди так. Досить часто сповільнення викликано неефективним управлінням пам'яттю. [1, 2] Щоб мати можливість більш ефективно керувати своїми даними, можна здійснити перехід від об'єктно-орієнтованого мислення до орієнтованого на дані дизайну, підмножиною якого є патерн Система компонент суб'єкта. Ціль дизайну – повністю перемістити фокус з об'єктів на обробку фактичних даних.

У більшому масштабі оптимальне розміщення даних у пам'яті може сильно вплинути на продуктивність програмного забезпечення, тому важливо звернути увагу на типи даних та спосіб їх обробки. В ідеалі дані розміщуються якомога однорідніше та безперервно, щоб їх можна було обробляти послідовно.

Ідеальний дизайн може бути досягнутий шляхом розбиття об'єктів на різні компоненти та групування цих компонентів у пам'яті за їх типом. Це призводить до однорідних і послідовно оброблюваних даних. Цей підхід дуже добре працює з великими групами об'єктів, на відміну від ООП, яке в основному фокусується на одному об'єкті.

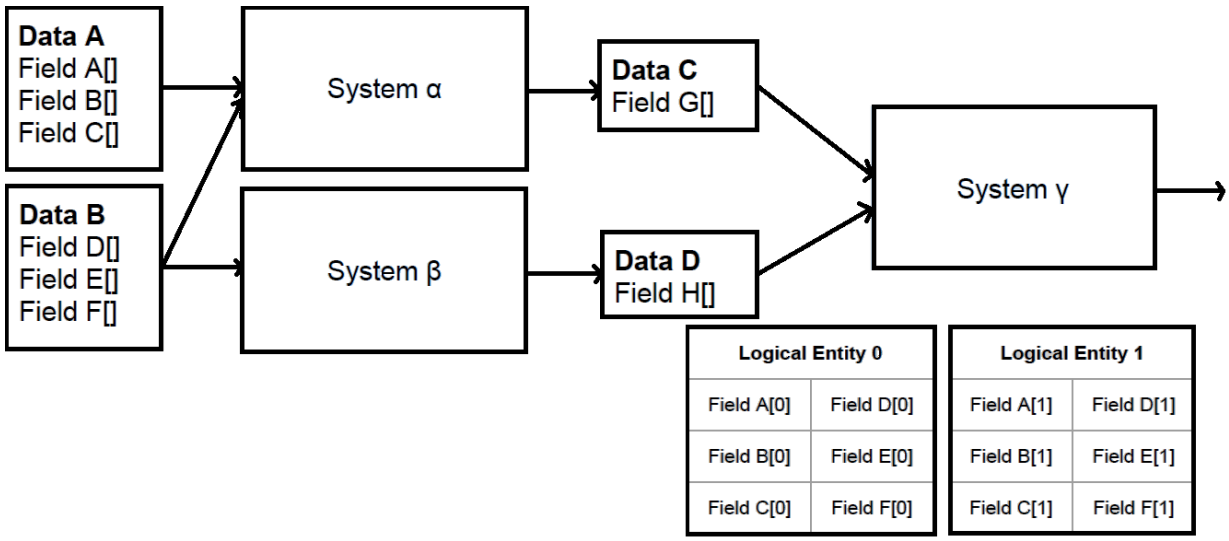

Рисунок 1. Діаграма використання даних у орієнтованому на данні дизайні

В об'єктно-орієнтованому програмуванні поділ процесу між кількома ядрами може бути складною операцією через помилки синхронізації, викликані різними потоками, які намагаються одночасно отримати доступ до одних і тих же даних. Якщо потоки чекають своєї черги для доступу до даних, це призводить до великої кількості простою, а віддача з точки зору підвищення продуктивності може бути незадовільною. Орієнтований на дані дизайн спрощує розпаралелювання. Оскільки дані обробляються групами, їх легко розділити між різними потоками, і код не викличе проблем, пов'язаних із синхронізацією [3].

Іншою сильною стороною дизайну є можливість оптимізованого використання кешу. У сучасному обладнанні ключовим моментом для досягнення високої продуктивності є впорядкування даних у пам’яті, щоб дані можна було ефективно використовувати знову і знову. Якщо дані розміщуються в пам’яті безперервно, їх можна обробляти з майже ідеальним використанням кешу, що призводить до чудової продуктивності. Хоча оптимізація алгоритмів, що використовуються для перетворення даних, безумовно важлива, якщо поглянути на те, як обробляються дані, цей підхід може вплинути на підвищення продуктивності на пізніх етапах розробки програмного забезпечення [2].

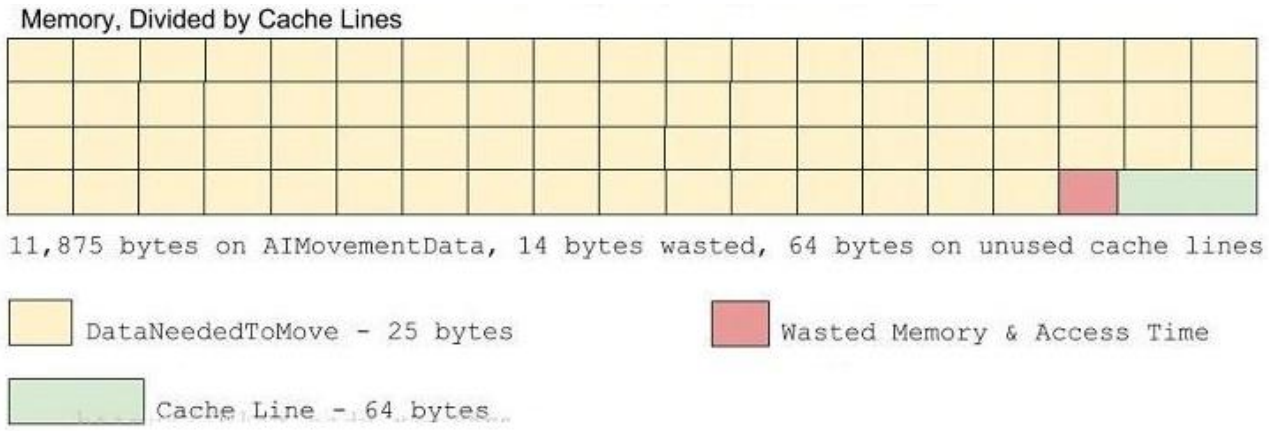

Рисунок 2. Використання пам’яті при використанні підході орієнтованому на данні

Література.

1. M. Infantino, Entity-Component Systems & Data Oriented Design In Unity, 19 August 2018.
2. N. Llopis, Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot with OOP), 4 December 2009 [Online].
3. A.Martin, Data Structures for EntitySystems: Multi-threading and networking, 2 May 2015.