УДК 004.41

І. Боднарчук, к.т.н., доц. О. Карнаухов, здобувач третього (освітньо-наукового) рівня вищої освіти, Т. Лобур, здобувач третього (освітньо-наукового) рівня вищої освіти, С. Марценко, к.т.н., доц.
Тернопільський національний технічний університет ім. І. Пулюя, Україна

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ JULIA ТА PYTHON ДЛЯ ВИРІШЕННЯ НАУКОВИХ І МАТЕМАТИЧНИХ ЗАДАЧ

I. Bodnarchuk, Ph.D., Assoc. Prof, O. Karnaukhov, PhD student, T. Lobur, PhD student
S. Martsenko, Ph.D., Assoc. Prof.
Ternopil Ivan Puluj National Technical University, Ukraine

COMPARISON OF JULIA AND PYTHON PROGRAMMING LANGUAGES FOR SOLVING SCIENTIFIC AND MATHEMATICAL PROBLEMS

У наукових дослідженнях вибір мови програмування суттєво впливає на продуктивність, масштабованість і ефективність обчислень. Мова Python [1] вже тривалий час є стандартом у цій галузі завдяки простому синтаксису, багатій екосистемі бібліотек і широкому колу користувачів. Водночас мова Julia [2] стрімко набирає популярності як сучасний інструмент для високопродуктивних чисельних обчислень, який поєднує зручність із потужністю системного програмування. У цій роботі проведено порівняльний аналіз обох мов програмування шляхом реалізації однакового алгоритму виявлення аномалій у часовому ряді та подальшого вимірювання часу виконання для оцінки їхньої ефективності в прикладній задачі.

Мова Julia використовує Just-In-Time компіляцію через LLVM (Low Level Virtual Machine), що суттєво підвищує швидкодію, тоді як Python є інтерпретованою мовою, тому для обчислення складних завдань покладається на оптимізовані зовнішні бібліотеки, такі як NumPy. Типова перевага мови Julia – можливість писати високопродуктивний код без необхідності додаткових оптимізацій або векторизації. Мова Python, зі свого боку, має незрівнянно більшу кількість бібліотек і готових рішень, що спрощує реалізацію комплексних систем.

У межах дослідження реалізовано алгоритм виявлення аномалій у часовому ряді, основою якого є експоненційне ковзне середнє — підхід, що виконує згладження даних шляхом зваженого врахування поточних та попередніх спостережень. Завдяки цьому зменшується вплив шумових коливань і виокремлюється загальна поведінка ряду без необхідності застосування складних обчислювальних методів.

Після формування згладженого ряду визначаються залишки – різниця між фактичними значеннями та згладженими оцінками. Для виявлення аномалій використовується метод трьох сигм: статистичне правило, за яким спостереження вважається аномальним, якщо його залишок виходить за межі трьох стандартних відхилень від середнього. Це дає змогу ефективно відокремлювати значущі відхилення від природних флуктуацій, притаманних часовим рядам.

Алгоритм реалізовано у формі, яка не передбачає вкладених ітерацій або складних структур керування, що забезпечує його лінійну часову складність – $O(n)$, де n – кількість елементів у ряді. Така архітектура дозволяє легко масштабувати розв’язання задачі на великі обсяги даних без втрати продуктивності. Завдяки своїй універсальності, підхід є зручним інструментом як для технічного моніторингу чи діагностики, так і для навчальних та експериментальних досліджень.

У роботі алгоритм використано як базову експериментальну задачу для порівняння ефективності реалізацій на мовах програмування Julia та Python. Обрана структура алгоритму репрезентує типове навантаження, характерне для задач попереднього аналізу часових рядів:

вона охоплює вхід даних, обробку, умовну логіку та вивід результату. Це дає змогу об'єктивно оцінити особливості виконання одного й того ж обчислювального сценарію в контексті різних мов програмування – зокрема їхньої швидкодії, масштабованості та зручності реалізації.

Під час експерименту для кожної мови проведено 10 окремих запусків програми з однаковими вхідними даними та параметрами. Кожен запуск передбачав повне виконання всіх етапів – від початкового читання файлу до збереження списку виявлених аномалій. Час виконання вимірювався за допомогою стандартних засобів відповідних мов програмування. Отримані результати представлені у таблиці 1 для подальшого аналізу та обчислення середнього часу виконання для кожної мови в секундах.

	PYTHON (С)	**JULIA (С)**
1	2,8709	0,4680
2	2,9138	0,4370
3	2,8456	0,4060
4	2,8044	0,4060
5	2,5402	0,4060
6	2,7270	0,3900
7	2,6392	0,3910
8	2,6109	0,4060
9	2,6122	0,3910
10	2,6115	0,3910
СЕРЕДНЄ	2,71757	0,4092

Таблиця 1 – Порівняння часу виконання алгоритму мовами Julia та Python

Аналіз результатів показав, що середній час виконання для програм на мові Python склав 2,71757 секунди, тоді як для мови Julia – 0,4092 секунди. Таким чином, мова Julia працювала приблизно у 6,64 рази швидше, ніж мова Python, при однаковому алгоритмі та однакових умовах тестування. Ці результати підтверджують очікувану перевагу мови Julia в задачах, що вимагають високої продуктивності.

Експеримент підтвердив, що мова Julia є ефективним інструментом для задач, де критично важлива швидкодія, зокрема для обчислень, які складно або неможливо розпаралелювати. Завдяки JIT-компіляції мова Julia суттєво переважає над мовою Python за швидкістю при однаковій алгоритмічній складності. Водночас мова Python залишається оптимальним вибором для проєктів із багатьма залежностями, інтеграцією сторонніх бібліотек і потребою в швидкому прототипуванні. Вибір мови залежить від природи задачі: для високопродуктивних обчислень варто обрати мову Julia, для комплексних систем – мову Python.

Посилання

1. https://www.python.org/
2. https://julialang.org/