УДК 004.382.76

ПОРІВНЯННЯ ШВИДКОДІЇ JAVA НА МІКРОКОМП'ЮТЕРІ RASPBERRY PI[1]

Дідух О. І., магістр; Тищенко В. В., магістр
Національний технічний університет України
«Київський політехнічний інститут», м. Київ, Україна,
sasha.didukh@gmail.com

COMPARISON OF FAST-ACTING OF JAVA ON MICROCOMPUTER RASPBERRY PI

Didukh O. I., Tyshchenko V. V.
National Technical University of Ukraine, Kyiv Politechnic Institute, Kiev, Ukraine,
sasha.didukh@gmail.com

Вступ

Мініатюризація та збільшення швидкодії є загально признаною тенденцією розвитку сучасних комп'ютерів. Сьогодні популярності набули мікрокомп'ютери, зокрема *Raspberry Pi* [1]. На їх основі будуються різноманітні системи: керування, контролю, спостереження, розпізнавання і т. д. Розроблений британським Фондом *Raspberry Pi Foundation* мікрокомп'ютер *Raspberry Pi Model* B має 700 *MHz* процесор *ARM1176-JZFS* та 512 МБ оперативної пам'яті *LPDDR*2-800. Таке оснащення дозволяє використовувати високорівневі мови програмування [2]. Однією з них є об'єктно-орієнтована мова програмування *Java*.

Метою даної статті є аналіз швидкодії різних версій *Java* на *Raspberry Pi*, визначення оптимальної версії *Java* для застосування.

Теоретичні викладки

Практичним методом перевірки швидкодії Java є визначення часу виконання певної програми на віртуальній машині *Java* (*JVM*) [3]. Для порівняння використаємо дві функціонально відмінні програми. В першому випадку, реалізуємо сучасний швидкий алгоритм пошуку простих чисел до заданого цілого числа *N* на *Java* (решето Аткіна) [4].

В основі даного алгоритму лежать 3 стандартні теореми теорії елементарних чисел:

1. n — просте, якщо:

$$4 \cdot x^2 + y^2 = n \; (x > 0, y > 0)$$
$$n \bmod 4 = 1$$

n — непарне число.

2. n — просте, якщо:

[1] http://radap.kpi.ua/radiotechnique/article/view/994

$$3 \cdot x^2 + y^2 = n \ (x > 0, y > 0)$$
$$n \bmod 6 = 1$$

n — непарне число.

3. n — просте, якщо:

$$3 \cdot x^2 - y^2 = n \ (x > 0, y > 0)$$
$$n \bmod 12 = 11$$

n – непарне число.

Для реалізації алгоритму необхідно виконати наступні пункти:

1. Створити решето (масив відповідності простим числам для всіх додатних цілих чисел починаючи з 2). Початково всі елементи решета позначаються як складові.

2. Для кожного числа n в решеті поміняти значення в решеті на протилежне, якщо залишок від ділення по модулю 60:

а) дорівнює 1, 13, 17, 29, 37, 41, або 53, та $n = 4 \cdot x^2 + y^2$;

б) дорівнює 7, 19, 31, або 43, та $n = 3 \cdot x^2 + y^2$;

в) дорівнює 11, 23, 47, або 59, та $n = 3 \cdot x^2 - y^2$.

(де x та y цілі додатні числа)

3. Взяти найменше число з решета, позначене як просте, і позначити всі елементи решета, кратні квадрату цього простого числа як складові.

4. Повторити пункт 3.

Алгоритм решета Аткіна має асимптотичну складність:

$$O(\frac{N}{loglogN})$$

та потребує наступну кількість біт пам’яті:

$$O(N^{\frac{1}{2}+o(1)})$$

В другому випадку, розглянемо програму, яка буде циклічно виконувати операції множення, ділення, та додавання над числами з плаваючою комою.

При обчисленні часу роботи проведемо $N = 10$ запусків алгоритму та отримаємо відповідні часи роботи: $(t_1, \ldots, t_N)$. Визначимо мінімально можливий час роботи алгоритму $t_{min} = \min\limits_{i=1,\ldots,N} t_i$, максимальний $t_{max} = \max\limits_{i=1,\ldots,N} t_i$ та середній $\bar{t} = \frac{1}{N} \sum_{i=1}^{N} t_i$. Для порівняльного аналізу розглянемо середній час роботи, оскільки при його визначенні враховані впливи сторонніх факторів, так як мінімальний час отримати нелегко.

Для оцінки похибки вимірювання обрахуємо стандартне відхилення (σ) за формулою:

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (t_i - \bar{t})^2}$$

Кількість запусків алгоритму підібрана таким чином, щоб стандартне відхилення не перевищувало 5% від середнього часу роботи, в іншому випадку результат вимірювання не буде достовірним.

Методика та засоби експериментальних досліджень

Дослідження часу роботи алгоритму проводиться згідно з наступним планом:

1. Підготовка мікрокомп'ютера.
3. Проведення трьох холостих запусків алгоритму [5].
4. Запуск програми на виконання.
5. Обрахунок часу виконання програми.
6. Визначення мінімального, максимального та середнього значення часу виконання.
7. Визначення стандартного відхилення та похибки вимірювання.

На етапі підготовки мікрокомп'ютера були зупинені всі процеси які можуть знизити швидкість виконання програм або внести суттєву нерівномірність в результати роботи (відео та звукові програвачі, браузери, поштові клієнти, автоматичне оновлення системи і т. п.).

Перед початком вимірювань необхідно зробити декілька холостих запусків алгоритму (без вимірювання часу), щоб виключити вплив кешів процесора на результат вимірювання, який може внести додаткову похибку. Експериментально виявлено, що при роботі програм використаних в даній статті достатньо трьох холостих запусків алгоритму.

Для порівняння проведено запуск програми пошуку простих чисел та програми роботи над числами з плаваючою комою на мікрокомп'ютері *Raspberry Pi* з встановленими різними версіями *JVM*: *Oracle Java SE* 8 (з *JavaFX*) та *OpenJDK Java*.

Результати експериментів

На мікрокомп'ютері встановлено *Oracle Java SE* 8 (з *JavaFX*):

java version "1.8.0" *Java* (TM) *SE Runtime Environment* (*build* 1.8.0-*b*132)

Java HotSpot (TM) *Client VM* (*build* 25.0-*b*70, *mixed mode*)

Результат виконання програми пошуку простих чисел від 1 до $5 \cdot 10^7$ на *Raspberry Pi*:

Мінімальний час розрахунку становить 43694 мс

Середній час розрахунку становить 43718.0 мс

Максимальний час розрахунку становить 43751 мс

SKO = 15.9721985

Відношення СКО до середнього часу = 0.036534607 %

На мікрокомп'ютері *Raspberry Pi* можливо змінити частоту процесора [6]. Використаємо частину доступних значень: 700 *MHz*, 900 *MHz*, 1000 *MHz*.

Проведемо запуск програми на кожній з цих частот окремо і результати занесемо до табл. 1.

Далі встановимо *OpenJDK Java* на *Raspberry Pi* [7]:

java version "1.7.0_65" *OpenJDK Runtime Environment* (*IcedTea* 2.5.3) (7*u*71-2.5.3-2~*deb*7*u*1+*rpi*1) *OpenJDK Zero VM* (*build* 24.65-*b*04, *mixed mode*).

Результати виконання програми на *OpenJDK* занесемо до табл. 1.

Таблиця 1 Результат виконання програми пошуку простих чисел за алгоритмом решето Аткіна

Платформа для тестування	$\bar{t}$, мс	t_{min}, мс	t_{max}, мс	σ	$\frac{\sigma}{\bar{t}}$, %
Raspberry Pi, *Oracle*, 700 *MHz*	43718.0	43694	43751	15.9721985	0,0365346
Raspberry Pi, *Oracle*, 900 *MHz*	34607.0	34583	34656	20.295046	0.058644336
Raspberry Pi, *Oracle*, 1000 *MHz*	29605.0	29584	29747	49.978886	0.16881907
Raspberry Pi, *JDK*, 700 *MHz*	613509.0	591445	634453	18636.754	3.0377312
Raspberry Pi, *JDK*, 900 *MHz*	490957.0	474098	474098	17348.08	3.274904
Raspberry Pi, *JDK*, 1000 *MHz*	403535.0	397945	409229	5291.9624	1.3114011

Повторимо вищеописані кроки з програмою для обчислення операцій з плаваючою комою. Результати розрахунків зображені в табл. 2.

Таблиця 2 Результат виконання програми для обчислення операцій з плаваючою комою

Платформа для тестування	$\bar{t}$ мс	t_{min}, мс	t_{max}, мс	σ	$\frac{\sigma}{\bar{t}}$, %
Raspberry Pi, Oracle, 700 MHz	24345	24338	24351	4.642796	0.019070841
Raspberry Pi, Oracle, 900 MHz	18845	18836	18861	7.3181663	0.038833465
Raspberry Pi, Oracle, 1000 MHz	16798	16795	16807	3.9015667	0.023226377
Raspberry Pi, JDK, 700 MHz	332697	331817	334380	1147.6833	0.34496352
Raspberry Pi, JDK, 900 MHz	257903	257584	258571	450.38467	0.17463335
Raspberry Pi, JDK, 1000 MHz	229275	229205	229352	66.79633	0.029133717

Висновки

Аналіз результатів показує, що на мікрокомп'ютері *Raspberry Pi* час виконання програм залежить від версії встановленої *Java* та частоти роботи процесора. Отримано різні часи виконання програм, наведені в табл. 1, 2. На прикладі програми пошуку простих чисел, бачимо, що час виконання в 13,6 -14,16 разів більший при встановленій *Java OpenJDK* в порівнянні з часом роботи на *Java* від *Oracle*. Програма для роботи над числами з плаваючою комою при встановленій *Java OpenJDK* виконується в 13,65 -13,69 разів довше в порівнянні з часом роботи на *Java* від *Oracle*. (табл. 3).

Таблиця 3

Платформа для тестування	$\frac{t_{OpenJDK}}{t_{Oracle}}$ Решето Аткіна	$\frac{t_{OpenJDK}}{t_{Oracle}}$ Операції з плаваючою комою
Raspberry Pi, 700 *MHz*	14,03	13,67
Raspberry Pi, 900 *MHz*	14,19	13,69
Raspberry Pi, 1000 *MHz*	13,63	13,65

На мікрокомп'ютері *Raspberry Pi* елементарні операції над числами з плаваючою комою виконуються швидше в порівнянні зі складним алгоритмом пошуку простих чисел. Також очевидно, що *Java* від *Oracle* працює в 13-14 разів швидше від *OpenJDK* (див. табл. 3).

Отже, в порівнянні з *OpenJDK Java* на *Raspberry Pi Java SE* 8 дає хороші показники роботи і рекомендується для встановлення та використання на мікрокомп'ютерах.

Перелік посилань

1. З початку продажів по світу розійшлося вже 5 млн мікрокомп'ютерів Raspberry Pi [Електронний ресурс]. – Режим доступу: http://mikrotik.kpi.ua/index.php/component/k2/item/24-5-billion-raspberry-pi. — Назва з екрану.

2. Rainer E. Raspberry Pi and Java – Love At First Sight [Електронний ресурс] / Eschrich Rainer // M2M Summit 2013 ; September, 2013. – Режим доступу : http://m2m-summit.com/files/m2m_oracle_java___raspberry_pi _v2.pdf. – Назва з екрану.

3. Java theory and practice: Dynamic compilation and performance measurement [Електронний ресурс] / DeveloperWorks. – Режим доступу : http://www.ibm.com/developerworks/library/j-jtp12214/?S_TACT=105AGX99 &S_CMP=CP. – Назва з екрану.

4. Atkin A. O. L. Prime Sieves Using Binary Quadratic Forms / A. O. L. Atkin, D. J. Bernstein // Mathematics of Computation. – *2004.* – Vol. 73, No. 246. – pp. 1023-1030.

5. Акиньшин А. Учимся писать правильные C#-бенчмарки / Андрей Акиньшин ; Хабрахабр, 28 августа 2013. – Режим доступу : http://habrahabr.ru/ company/enterra/blog/191636. – Назва з екрану.

6. Monk S. Raspberry Pi Cookbook / Simon Monk. – O'Reilly Media Inc., 2014. –

408 p.

7. Могильний С. Б. Мікрокомп’ютер Raspberry Pi – інструмент дослідника / С. Б. Могильний. – К. : Талком, 2014. – 340 с. – ISBN 978-617-7133-48-2

References

1. Z pochatku prodazhiv po svitu roziishlosia vzhe 5 mln mikrokomp'iuteriv Raspberry Pi [From sales beginning in the world have already dispersed 5 million microcomputers Raspberry Pi]. – Available at: http://mikrotik.kpi.ua/index.php/component/k2/item/24-5-billion-raspberry-pi.

2. Rainer Eschrich (2013) Raspberry Pi and Java - Love At First Sight. *M2M Summit 2013*. Avaible at: http://m2m-summit.com/files/m2m_oracle_java___raspberry_pi_v2.pdf.

3. *Java theory and practice: Dynamic compilation and performance measurement*. – Available at: http://www.ibm.com/developerworks/library/j-jtp12214/?S_TACT=105AGX99&S_CMP=CP.

4. Atkin A. O. L. and Bernstein D. J. (2004) Prime Sieves Using Binary Quadratic Forms. *Mathematics of Computation*, Vol. 73, No. 246, pp. 1023-1030.

5. Akin'shin A. (2013) *Uchimsya pisat' pravil'nye C#-benchmarki* [Learn to write correctly C# - benchmarks]. – Available at : http://habrahabr.ru/company/enterra/blog/191636.

6. Monk S. (2014) *Raspberry Pi Cookbook*. O'Reilly Media, Inc., 408 p.

7. Mohylnyi S. B. (2014) *Mikrokomp’iuter Raspberry Pi – instrument doslidnyka* [Microcomputer Raspberry Pi - a tool for researcher]. Kyiv, Talkom Publ., 340 p. – ISBN 978-617-7133-48-2

Дідух О. І., Тищенко В. В. ***Порівняння швидкодії java на мікрокомп’ютері Raspberry Pi.*** *При роботі з високорівневою мовою програмування на мікрокомп’ютері важливим є швидкість виконання певного набору інструкцій. Для визначення швидкодії Java на мікрокомп’ютері Raspberry Pi застосовано метод порівняння часів виконання двох функціонально відмінних програм на різних версіях віртуальної машини Java. Реалізовано програму сучасного швидкого алгоритму пошуку простих чисел до заданого цілого числа N (решето Аткіна) та програму для виконання елементарних операцій над числами з плаваючою комою.*

Ключові слова: *Raspberry Pi; Java; мікрокомп’ютер; порівняння швидкодії; Oracle Java; OpenJDK.*

Дидух А. И., Тищенко В. В. ***Сравнение быстродействия java на микрокомпьютере Raspberry Pi.*** *При работе с высокоуровневым языком программирования на микрокомпьютере важным является скорость выполнения определенного набора инструкций. Для определения быстродействия Java на микрокомпьютере Raspberry Pi применен метод сравнения времен выполнения двух функционально отличных программ на разных версиях виртуальной машины Java. Реализована программа современного быстрого алгоритма поиска простых чисел до заданного целого числа N (решето Аткина) и программа для выполнения элементарных операций над числами с плавающей точкой.*

Ключевые слова: *Raspberry Pi; Java; микрокомпьютер; сравнения быстродействия; Oracle Java; OpenJDK.*

Didukh O. I., Tyshchenko V. V. ***Comparison of fast-acting of java on microcomputer raspberry pi.***

Introduction. The speed of a specific set of instructions is important working with high-level programming language for microcomputers. The method of comparing execution times of two functionally different programs on different versions of Virtual Machine Java is used to determine the performance of Java on Raspberry Pi microcomputer. Application of modern fast search algorithm primes up to a given integer N (Atkin sieve) and a program to perform basic operations on floating point numbers are implemented.

Results. Two algorithms microcomputer operating frequencies: 700 MHz, 900 MHz, 1000 MHz are tested. Two versions of Java for Raspberry Pi: Oracle Java and OpenJDK are used.

Conclusions. In microcomputer Raspberry Pi elementary operations on floating-point run are faster than with complex search algorithm primes. It is established that Java from Oracle is 13-14 times is faster than OpenJDK. Oracle Java on Raspberry Pi gives good performance and it is recommended for installation and use on microcomputer.

Keywords*: Raspberry Pi; Java; microcomputer; comparing performance; Oracle Java; OpenJDK.*