УДК 004.051

# ЭНЕРГОЭФФЕКТИВНОСТЬ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

*Ульянова М.К., Митина А.В.*

*Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники» филиал «Минский радиотехнический колледж», г. Минск, Республика Беларусь*

*Научный руководитель: Сальникова Е.А. - преподаватель первой категории, магистр*

**Аннотация.** В данной статье представлены результаты исследования времени выполнения, использования памяти и энергопотребления 27 известных языков программирования. Производительность языков отслеживалась с помощью 10 различных задач программирования, выраженных на каждом из языков. В статье также рассказывается, как использовать результаты в принятии решения о том, какой язык использовать, когда речь идет об энергоэффективности.

**Ключевые слова:** энергоэффективность, языки программирования, энергопотребление, языковой Бенчмаркинг.

***Введение.*** Основными критериями выбора языка программирования являются специфика задач, возможности сотрудников и уже имеющаяся технологическая база, однако экологичность также стоит принять в расчет. Так, скрипты, написанные на JavaScript и Python, потребляют больше ресурсов по сравнению с компилированными программами, написанными на Fortran, C++ и Rust.

В свою очередь, объектно-ориентированные языки являются более ресурсоемкими, чем императивные. Большая ресурсоемкость увеличивает энергопотребление, и, как следствие, количество углеродных выбросов.

Сейчас реальность быстро меняется, и энергопотребление программного обеспечения становится ключевой проблемой, которая изначально возникла у производителей оборудования и быстро затронула разработчиков, обычных пользователей [5].

В наше время актуально направление исследований, в котором разрабатываются различные техники анализа и оптимизации энергопотребления программных систем.

В 2017 году португальские исследователи из HASLab (High-Assurance Software Laboratory) проанализировали энергопотребление 27 самых популярных языков программирования. Результаты действительны на данный момент, так как были обновлены в 2021 году.

***Основная часть.*** Исследование направлено на анализ энергоэффективности 28 языков программирования, из которых был исключен Smalltalk, так как компилятор для этого языка является проприетарным. Также была собрана информация о потреблении энергии, времени выполнения и пиковом использовании памяти для каждого из компилируемых и выполняемых решений на каждом языке [1].

Для измерения потребления энергии использовался инструмент Intel Running Average Power Limit (RAPL), который способен предоставлять точные оценки энергопотребления.

Для измерения использования памяти использовалась утилита времени (time tool), доступная в Unix-системах. Для получения сравнимого, репрезентативного и обширного набора программ, написанных на многих из самых популярных и широко используемых языках программирования, использовался The Computer Language Benchmarks Game [4] (CLBG).

Полный список бенчмарк задач в CLBG охватывает различные вычислительные проблемы, показанные на рисунке 1.

Кроме того, полный список языков программирования в CLBG показан на рисунке 2, отсортированном по парадигмам.

| Benchmark | Description | Input |
|---|---|---|
| n-body | Double precision N-body simulation | 50M |
| fannkuch-redux | Indexed access to tiny integer sequence | 12 |
| spectral-norm | Eigenvalue using the power method | 5,500 |
| mandelbrot | Generate Mandelbrot set portable bitmap file | 16,000 |
| pidigits | Streaming arbitrary precision arithmetic | 10,000 |
| regex-redux | Match DNA 8mers and substitute magic patterns | fasta output |
| fasta | Generate and write random DNA sequences | 25M |
| k-nucleotide | Hashtable update and k-nucleotide strings | fasta output |
| reverse-complement | Read DNA sequences, write their reverse-complement | fasta output |
| binary-trees | Allocate, traverse and deallocate many binary trees | 21 |
| chameneos-redux | Symmetrical thread rendezvous requests | 6M |
| meteor-contest | Search for solutions to shape packing puzzle | 2,098 |
| thread-ring | Switch from thread to thread passing one token | 50M |

Рисунок 1 – Список бенчмарк задач CLBG

| Paradigm | Languages |
|---|---|
| Functional | Erlang, F#, Haskell, Lisp, Ocaml, Perl, Racket, Ruby, Rust; |
| Imperative | Ada, C, C++, F#, Fortran, Go, Ocaml, Pascal, Rust; |
| Object-Oriented | Ada, C++, C#, Chapel, Dart , F#, Java, JavaScript, Ocaml, Perl, PHP, Python, Racket, Rust, Smalltalk, Swift, TypeScript; |
| Scripting | Dart, Hack, JavaScript, JRuby, Lua, Perl, PHP, Python, Ruby, TypeScript; |

Рисунок 2 – Языки программирования по парадигмам

Результаты исследования показали, что скомпилированные языки являются одними из самых быстрых и энергоэффективных. Скомпилированные языки потребляли около 120 Дж для выполнения тестовых решений. В то время как для виртуальной машины и интерпретируемых языков (требующих пошаговых исполнителей исходного кода, где не происходит предрезультативного перевода) это значение составляло 576Дж и 2365Дж соответственно. Эта тенденция также прослеживается в отношении времени выполнения, поскольку скомпилированные языки занимали 5103 мс, языки виртуальных машин – 20623 мс, а интегрированные языки – 87614 мс (в среднем).

Бенчмарки (англ. benchmarks) – это стандартные тесты, которые используются для измерения производительности и/или функциональности компьютерных систем, программного обеспечения или аппаратного обеспечения.

Чтобы разобраться более подробно, результаты двоичных деревьев были взяты для иллюстрации сути. На рисунке 3 изображен график данных об энергии и времени для двоичных деревьев.

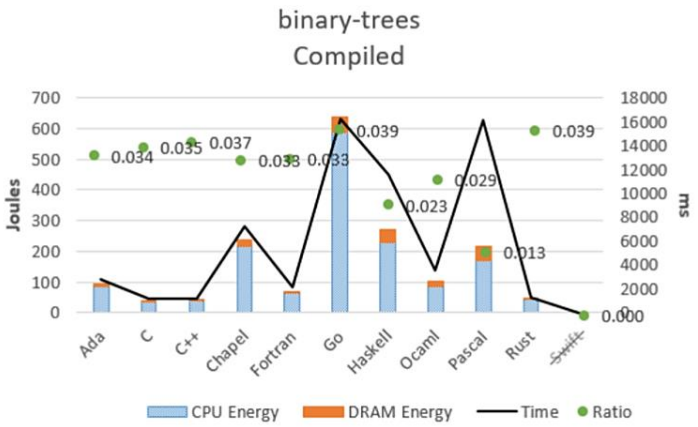


Рисунок 3 – Данные об энергии и времени для двоичных деревьев

Можно сделать вывод, что “скомпилированные языки” являются самыми быстрыми и энергоэффективными». C и C++ оказались наиболее эффективными и быстрыми языками. Go стал худшим из категории скомпилированных языков.

Глядя на двоичные деревья для категории виртуальных машин (рисунок 4), Java или Erlang оказались наиболее эффективными, но далеко позади скомпилированной категории.

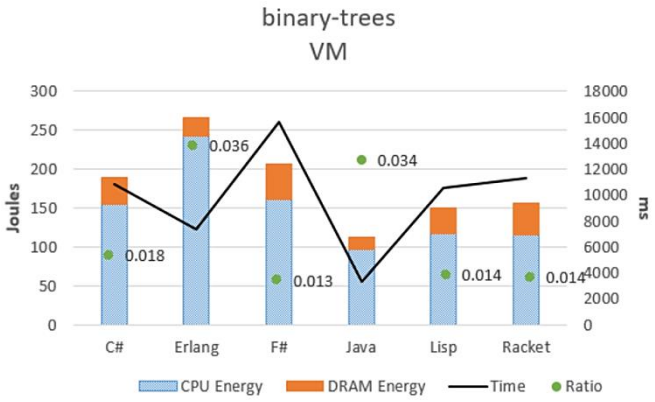


Рисунок 4 – Данные об энергии и времени для двоичных деревьев

Неэффективные языки переходят к интерпретируемым языкам, таким как Perl, Lua или Python (рисунок 5).

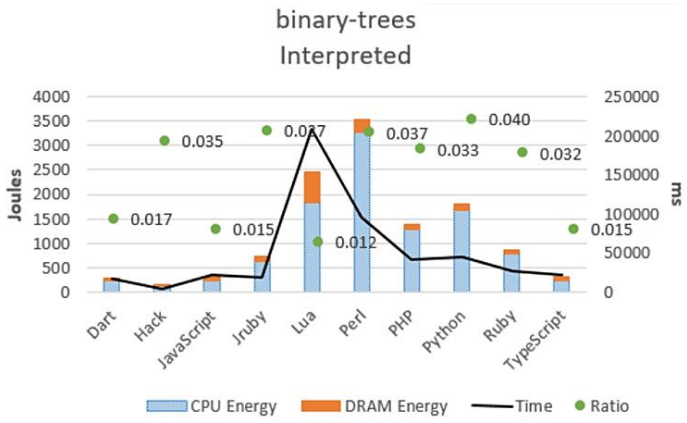


Рисунок 5 - Данные об энергии и времени для двоичных деревьев

Рассматривая исключительно общие результаты, самый энергоэффективный язык в тестах почти всегда является самым быстрым, важно понимать, что нет языка, который был бы неизменно лучше других. Это позволяет сделать вывод, что ситуация, в которой будет ис-

пользоваться язык, является ключевым аспектом для определения языка как наиболее энергоэффективного варианта.

Хоть была возможность незначительной корреляции между энергопотреблением и пиковым использованием памяти, на удивление, эта взаимосвязь практически отсутствует. Это указывает на то, что потребление энергии имеет очень мало общего с тем, сколько памяти сохраняется в данный момент, но, больше с тем, как она используется, поскольку такие факторы, как сбор мусора, использование кэша, будут оказывать сильное влияние на потребление энергии.

При выборе языка разработчики рассматривают более, чем одну характеристику эффективности. На рисунке 6 представлено четыре многоцелевых рейтинга: время и память, энергия и время, энергия и память, энергия с временем и памятью.

| Time & Memory | Energy & Time | Energy & Memory | Energy & Time & Memory |
|---|---|---|---|
| C • Pascal • Go | C | C • Pascal | C • Pascal • Go |
| Rust • C++ • Fortran | Rust | Rust • C++ • Fortran • Go | Rust • C++ • Fortran |
| Ada | C++ | Ada | Ada |
| Java • Chapel • Lisp • Ocaml | Ada | Java • Chapel • Lisp | Java • Chapel • Lisp • Ocaml |
| Haskell • C# | Java | OCaml • Swift • Haskell | Swift • Haskell • C# |
| Swift • PHP | Pascal • Chapel | C# • PHP | Dart • F# • Racket • Hack • PHP |
| F# • Racket • Hack • Python | Lisp • Ocaml • Go | Dart • F# • Racket • Hack • Python | JavaScript • Ruby • Python |
| JavaScript • Ruby | Fortran • Haskell • C# | JavaScript • Ruby | TypeScript • Erlang |
| Dart • TypeScript • Erlang | Swift | TypeScript | Lua • JRuby • Perl |
| JRuby • Perl | Dart • F# | Erlang • Lua • Perl | |
| Lua | JavaScript | JRuby | |
| | Racket | | |
| | TypeScript • Hack | | |
| | PHP | | |
| | Erlang | | |
| | Lua • JRuby | | |
| | Ruby | | |

Рисунок 6 – Оптимальные наборы по Парето для 4-х многоцелевых рейтингов.

Чтобы сравнить языки, использующие более одной характеристики, используется алгоритм многоцелевой оптимизации для сортировки этих языков, известный как оптимизация по Парето [2, 3].

Для каждого ранжирования каждая строка представляет оптимальный набор по Парето, содержащий языки эквивалентные друг другу для базовых целей.

Наиболее распространенными характеристиками производительности, являются время и памяти. Здесь C, Pascal и Go будут эквивалентны.

Однако, если посмотреть энергию и время, C является лучшим решением, поскольку он доминирует в обеих отдельных целях.

Если рассмотреть энергию и память, C и Pascal составляют оптимальный набор по Парето.

Если разработчик озабочен только временем выполнения и энергопотреблением, то почти всегда можно автоматически решить, какой язык программирования является наилучшим.

Во всех других рейтингах большинство позиций составлено набором оптимальных по Парето языков, которые эквивалентны с учетом базовых характеристик. В этих случаях разработчику необходимо решить, какие характеристики наиболее важные в каждом конкретном сценарии.

***Заключение.*** Невозможно полностью определить какой язык программирования является наилучшим, учитывая энергопотребление, время выполнения и максимальное использование памяти, необходимое программам. Для различных ситуаций можно найти наиболее подходящий.

Например, веб-приложения с большей вероятностью будут запрограммированы с использованием Python и JavaScript. Встроенные устройства в автомобильной промышленности и здравоохранении запускают программное обеспечение, написанное на C, C ++ или Rust. Приложения, работающие в облаке, пишутся на Go или Scala. Мобильные приложения все чаще пишутся на Swift или Kotlin.

***Список литературы***

*1. Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, João Saraiva HASLab/INESC TEC Universidade do Minho, Portugal Energy Efficiency across Programming Languages How Do Energy, Time, and Memory Relate?*

*2. C. Sahin, L. Pollock, J. Clause, How do code refactorings affect energy usage?, in: Proceedings of 8th ACM/IEEE International Symposium on Empirical SoftwareEngineering and Measurement, ESEM, 2014, p. 36.*

*3. R. Pereira, T. Car,c˜ao, M. Couto, J. Cunha, J. P. Fernandes, J. Saraiva, Helping programmers improve the energy efficiency of source code, in: Proceedings of the 39th International Conference on Software Eng. Companion, ICSE-C, ACM, 2017.*

*4. S. Hao, D. Li, W. G. J. Halfond, R. Govindan, Estimating mobile application energy consumption using program analysis, in: Proceedings of the 2013 Inter national Conference on Software Engineering, ICSE '13, IEEE Press, 2013, pp. 92–101.*

*5. M. Dimitrov, C. Strickland, S.-W. Kim, K. Kumar, K. Doshi, Intel R power gov ernor: 2015-10-12 (2015).*

UDC 004.051

# ENERGY EFFICIENCY OF PROGRAMMING LANGUAGES

*Ulyanova M.K, Mitina A.V*

*Belarusian State University of Informatics and Radioelectronics*
*affiliate Minsk Radioengineering College, Minsk, Republic of Belarus*

*Salnikova E.A. - teacher of the 1st category, master*

**Annotation**. This article presents the results of a study on the runtime, memory usage, and energy consumption of 27 popular programming languages. The performance of the languages was tracked using 10 different programming tasks expressed in each of the languages. The article also discusses how to use the results in making decisions about which language to use when energy efficiency is a concern.

**Keywords**. energy efficiency, programming languages, energy consumption, language benchmarking.