GREEN PROGRAMMING

Tina Ulbrich – Hendrik Niemeyer
ROSEN Technology and Research GmbH

NDC TechTown 2024

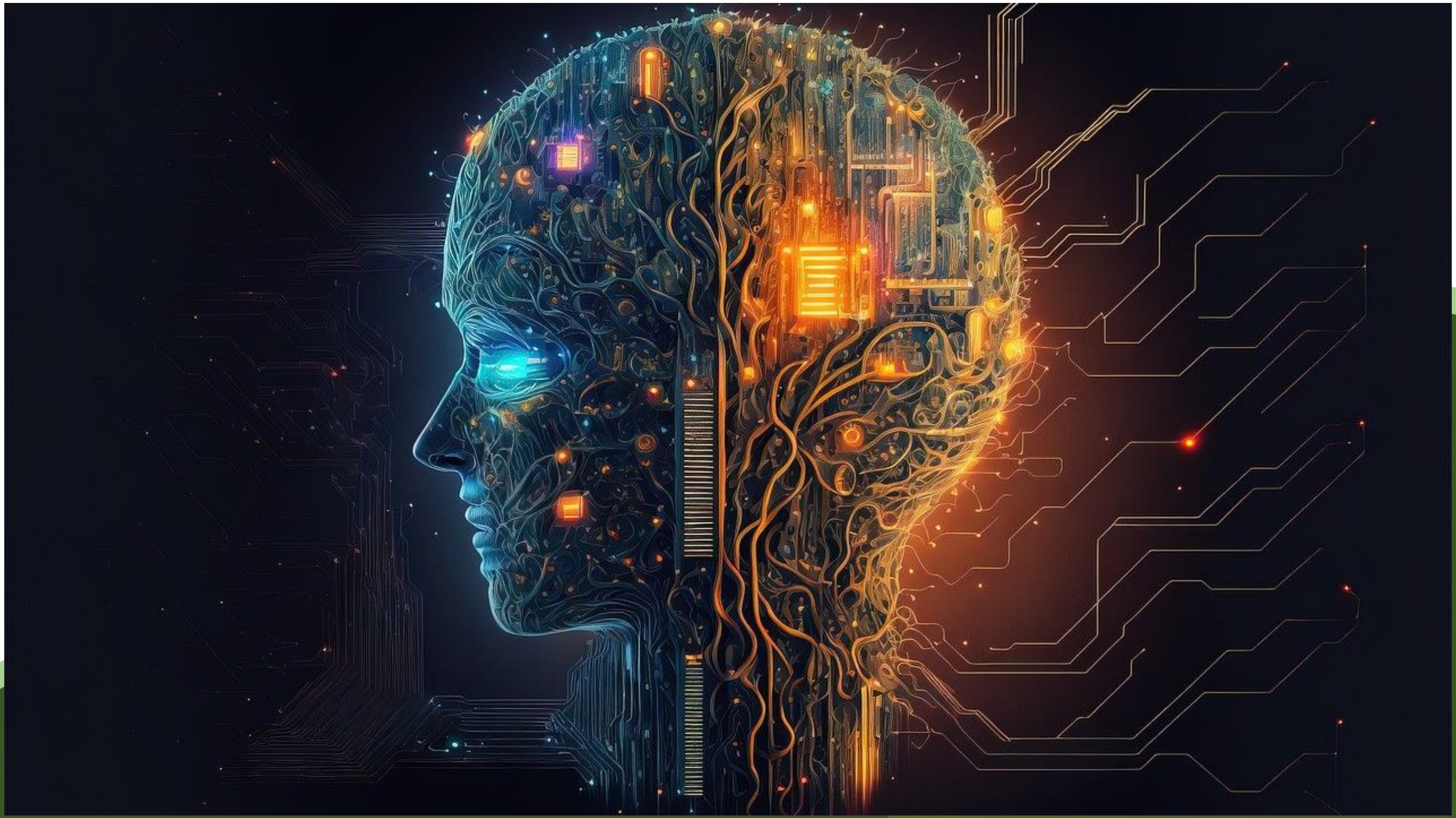
ROSEN
empowered by technology

OUTLINE

Energy Consumption

Data Centers

Artificial intelligence and machine learning



Data Centers

Increase in data traffic

Data Centers

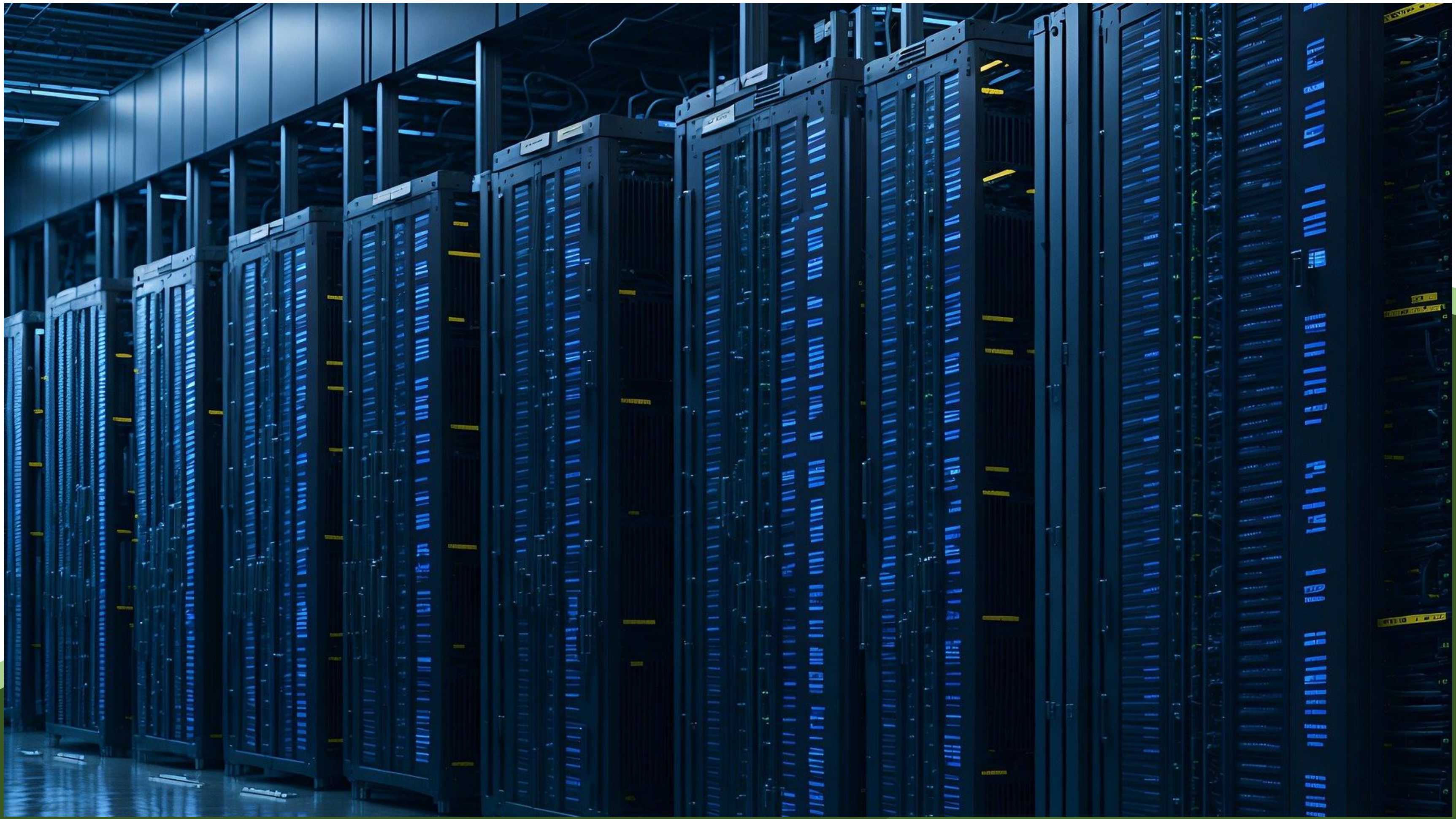
Growth of cloud services

Data Centers

Blockchain and cryptocurrencies

Data Centers

Increased availability and redundancy requirements

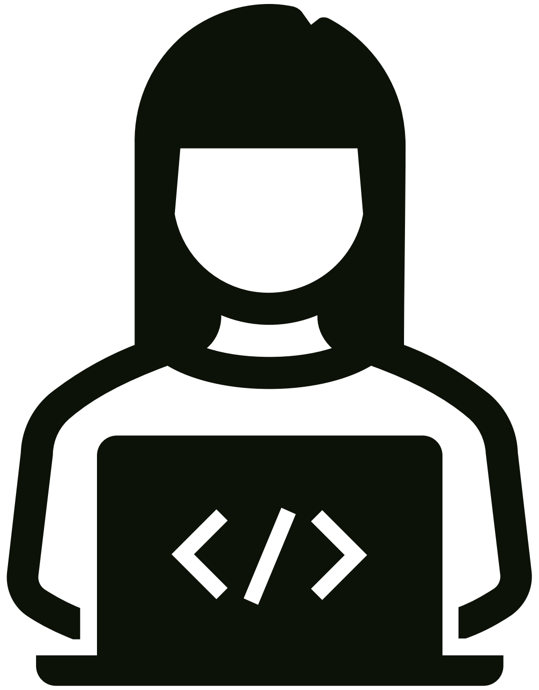


DATA CENTERS

Cooling requirements

Data Centers

Exponential growth of networked devices



Green Software

Green Software

using fewer physical resources

using energy more intelligently

using less energy

Green Software

Green Software Foundation

https://greensoftware.foundation/

Sustainable Digital Infrastructure Alliance

SDIA

SUSTAINABLE DIGITAL INFRASTRUCTURE ALLIANCE

https://sdialliance.org/

Playing for the Planet

PLAYING FOR THE PLANET™

playing4theplanet.org

accu conference 2024

Green Software Architecture

Dos Don'ts and Some Surprises

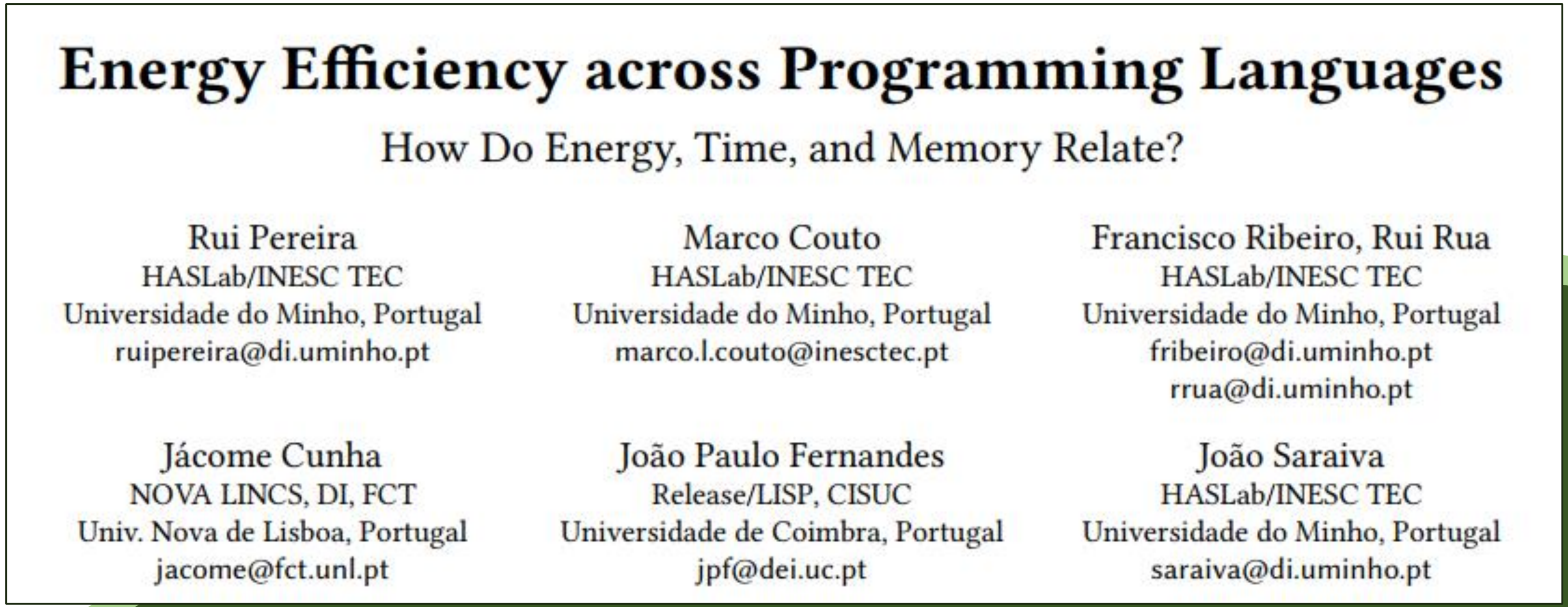
Giovanni Asproni

Green Programming

Green Programming Definition

Green Programming or green coding is a series of principles applied to software development that aims to reduce the ecological footprint of software.

Research Paper



Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

Research Paper

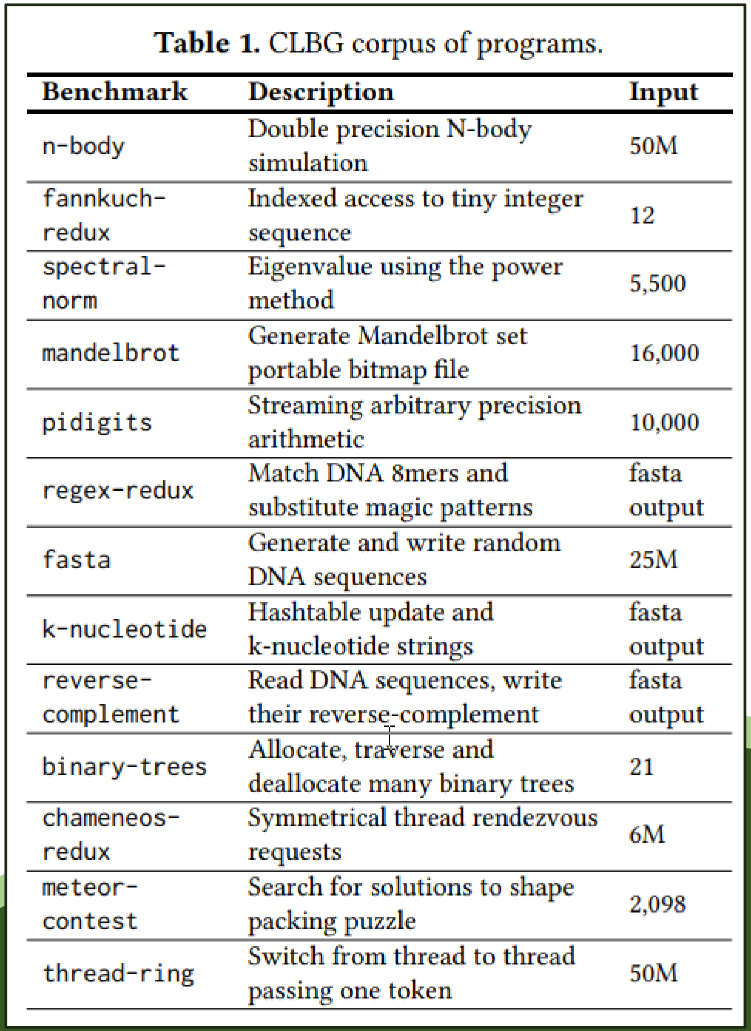


Table 1. CLBG corpus of programs.

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

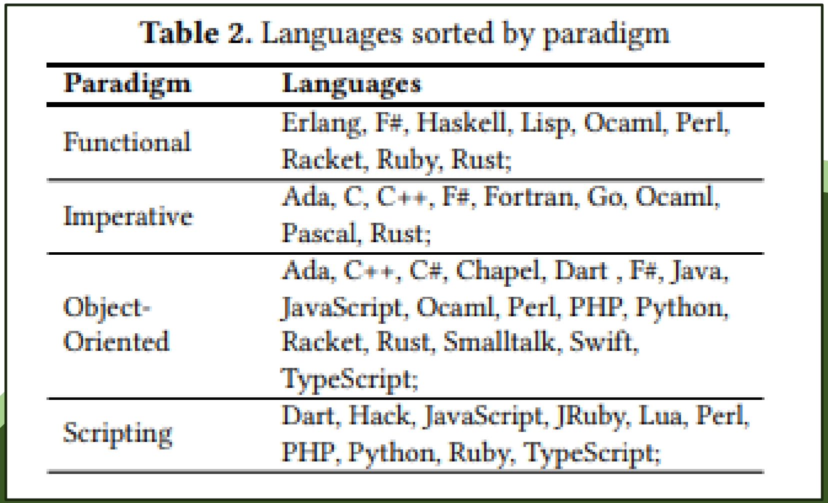
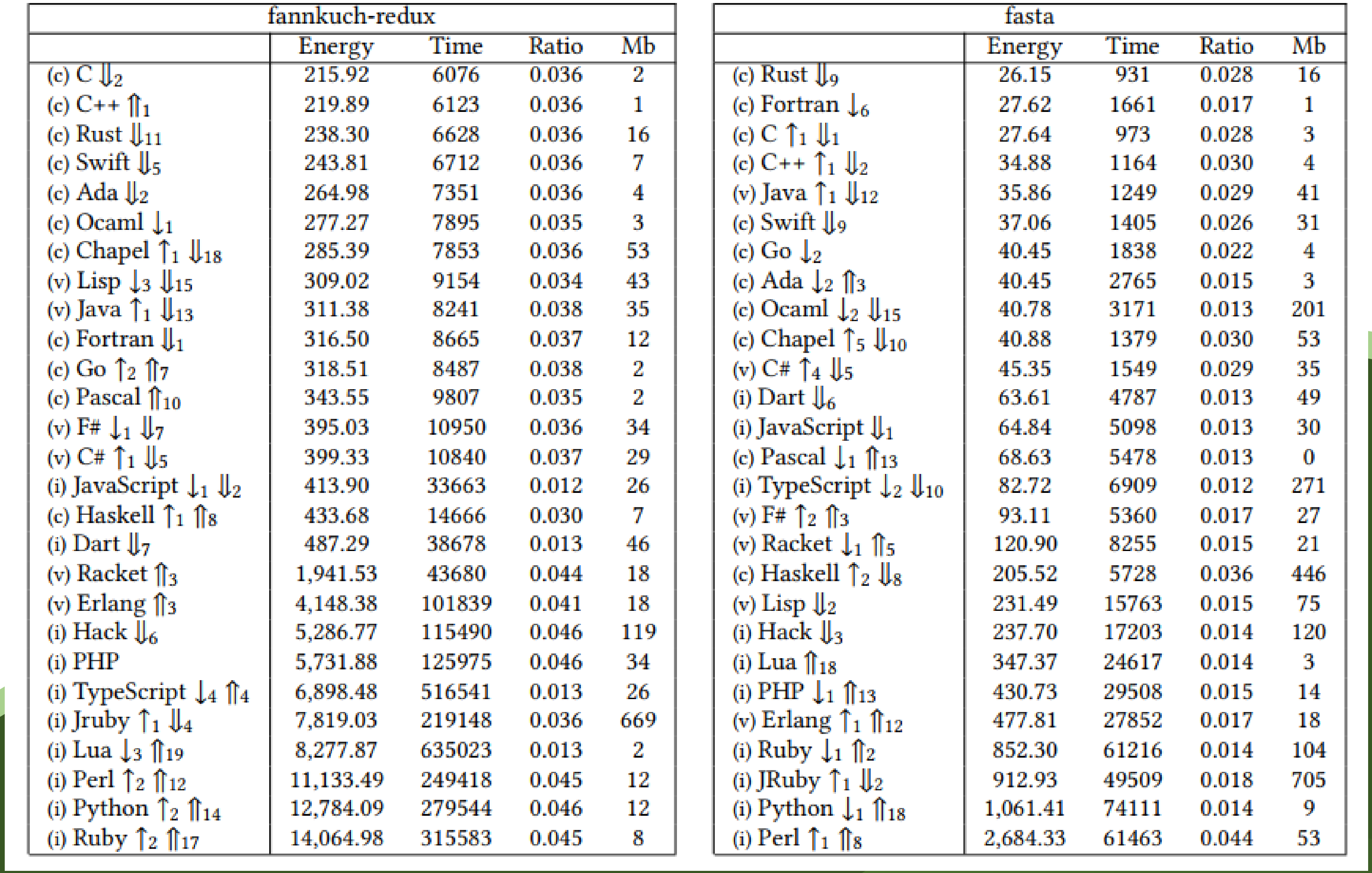


Table 2. Languages sorted by paradigm

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;

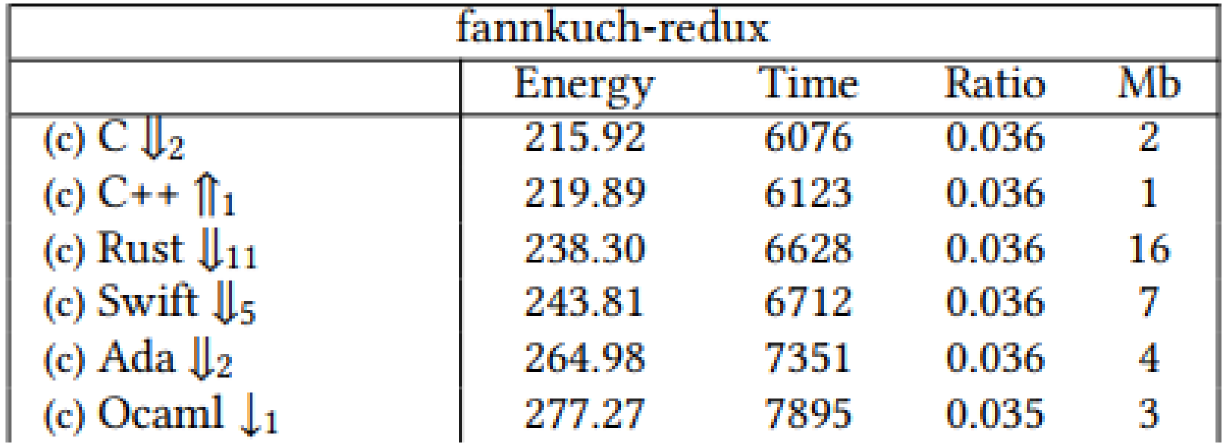
Research Paper



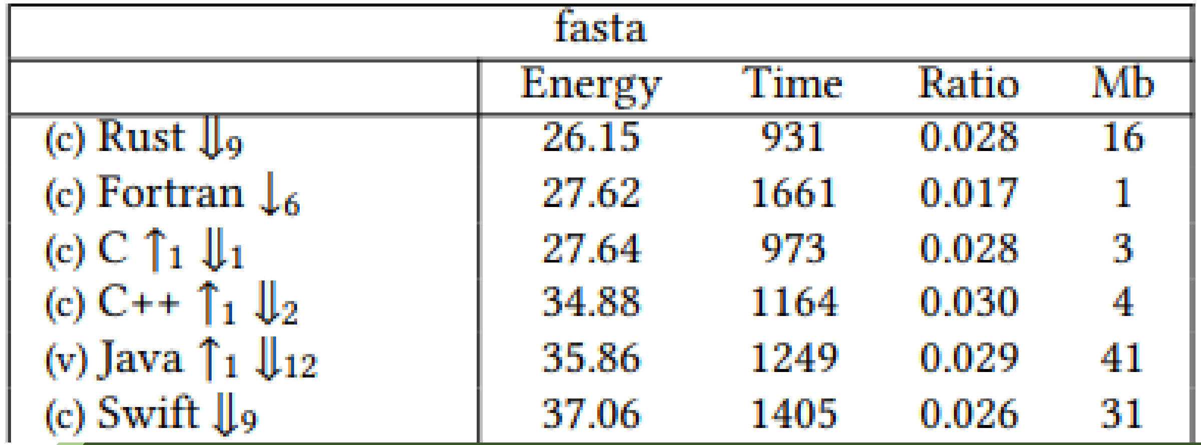
fannkuch-redux				
	Energy	Time	Ratio	Mb
(c) C $\Downarrow_{2}$	215.92	6076	0.036	2
(c) C++ $\Uparrow_{1}$	219.89	6123	0.036	1
(c) Rust $\Downarrow_{11}$	238.30	6628	0.036	16
(c) Swift $\Downarrow_{5}$	243.81	6712	0.036	7
(c) Ada $\Downarrow_{2}$	264.98	7351	0.036	4
(c) Ocaml $\downarrow_{1}$	277.27	7895	0.035	3
(c) Chapel $\uparrow_{1}$ $\Downarrow_{18}$	285.39	7853	0.036	53
(v) Lisp $\downarrow_{3}$ $\Downarrow_{15}$	309.02	9154	0.034	43
(v) Java $\uparrow_{1}$ $\Downarrow_{13}$	311.38	8241	0.038	35
(c) Fortran $\Downarrow_{1}$	316.50	8665	0.037	12
(c) Go $\uparrow_{2}$ $\Uparrow_{7}$	318.51	8487	0.038	2
(c) Pascal $\Uparrow_{10}$	343.55	9807	0.035	2
(v) F# $\downarrow_{1}$ $\Downarrow_{7}$	395.03	10950	0.036	34
(v) C# $\uparrow_{1}$ $\Downarrow_{5}$	399.33	10840	0.037	29
(i) JavaScript $\downarrow_{1}$ $\Downarrow_{2}$	413.90	33663	0.012	26
(c) Haskell $\uparrow_{1}$ $\Uparrow_{8}$	433.68	14666	0.030	7
(i) Dart $\Downarrow_{7}$	487.29	38678	0.013	46
(v) Racket $\Uparrow_{3}$	1,941.53	43680	0.044	18
(v) Erlang $\Uparrow_{3}$	4,148.38	101839	0.041	18
(i) Hack $\Downarrow_{6}$	5,286.77	115490	0.046	119
(i) PHP	5,731.88	125975	0.046	34
(i) TypeScript $\downarrow_{4}$ $\Uparrow_{4}$	6,898.48	516541	0.013	26
(i) Jruby $\uparrow_{1}$ $\Downarrow_{4}$	7,819.03	219148	0.036	669
(i) Lua $\downarrow_{3}$ $\Uparrow_{19}$	8,277.87	635023	0.013	2
(i) Perl $\uparrow_{2}$ $\Uparrow_{12}$	11,133.49	249418	0.045	12
(i) Python $\uparrow_{2}$ $\Uparrow_{14}$	12,784.09	279544	0.046	12
(i) Ruby $\uparrow_{2}$ $\Uparrow_{17}$	14,064.98	315583	0.045	8

fasta				
	Energy	Time	Ratio	Mb
(c) Rust $\Downarrow_{9}$	26.15	931	0.028	16
(c) Fortran $\downarrow_{6}$	27.62	1661	0.017	1
(c) C $\uparrow_{1}$ $\Downarrow_{1}$	27.64	973	0.028	3
(c) C++ $\uparrow_{1}$ $\Downarrow_{2}$	34.88	1164	0.030	4
(v) Java $\uparrow_{1}$ $\Downarrow_{12}$	35.86	1249	0.029	41
(c) Swift $\Downarrow_{9}$	37.06	1405	0.026	31
(c) Go $\downarrow_{2}$	40.45	1838	0.022	4
(c) Ada $\downarrow_{2}$ $\Uparrow_{3}$	40.45	2765	0.015	3
(c) Ocaml $\downarrow_{2}$ $\Downarrow_{15}$	40.78	3171	0.013	201
(c) Chapel $\uparrow_{5}$ $\Downarrow_{10}$	40.88	1379	0.030	53
(v) C# $\uparrow_{4}$ $\Downarrow_{5}$	45.35	1549	0.029	35
(i) Dart $\Downarrow_{6}$	63.61	4787	0.013	49
(i) JavaScript $\Downarrow_{1}$	64.84	5098	0.013	30
(c) Pascal $\downarrow_{1}$ $\Uparrow_{13}$	68.63	5478	0.013	0
(i) TypeScript $\downarrow_{2}$ $\Downarrow_{10}$	82.72	6909	0.012	271
(v) F# $\uparrow_{2}$ $\Uparrow_{3}$	93.11	5360	0.017	27
(v) Racket $\downarrow_{1}$ $\Uparrow_{5}$	120.90	8255	0.015	21
(c) Haskell $\uparrow_{2}$ $\Downarrow_{8}$	205.52	5728	0.036	446
(v) Lisp $\Downarrow_{2}$	231.49	15763	0.015	75
(i) Hack $\Downarrow_{3}$	237.70	17203	0.014	120
(i) Lua $\Uparrow_{18}$	347.37	24617	0.014	3
(i) PHP $\downarrow_{1}$ $\Uparrow_{13}$	430.73	29508	0.015	14
(v) Erlang $\uparrow_{1}$ $\Uparrow_{12}$	477.81	27852	0.017	18
(i) Ruby $\downarrow_{1}$ $\Uparrow_{2}$	852.30	61216	0.014	104
(i) JRuby $\uparrow_{1}$ $\Downarrow_{2}$	912.93	49509	0.018	705
(i) Python $\downarrow_{1}$ $\Uparrow_{18}$	1,061.41	74111	0.014	9
(i) Perl $\uparrow_{1}$ $\Uparrow_{8}$	2,684.33	61463	0.044	53

Research Paper



fannkuch-redux				
	Energy	Time	Ratio	Mb
(c) C $\Downarrow_2$	215.92	6076	0.036	2
(c) C++ $\Uparrow_1$	219.89	6123	0.036	1
(c) Rust $\Downarrow_{11}$	238.30	6628	0.036	16
(c) Swift $\Downarrow_5$	243.81	6712	0.036	7
(c) Ada $\Downarrow_2$	264.98	7351	0.036	4
(c) Ocaml $\downarrow_1$	277.27	7895	0.035	3



fasta				
	Energy	Time	Ratio	Mb
(c) Rust $\Downarrow_9$	26.15	931	0.028	16
(c) Fortran $\downarrow_6$	27.62	1661	0.017	1
(c) C $\uparrow_1$ $\Downarrow_1$	27.64	973	0.028	3
(c) C++ $\uparrow_1$ $\Downarrow_2$	34.88	1164	0.030	4
(v) Java $\uparrow_1$ $\Downarrow_{12}$	35.86	1249	0.029	41
(c) Swift $\Downarrow_9$	37.06	1405	0.026	31

The Algorithm

Hondt Method

	party 1 **votes: 110**		**party 2** **votes: 85**		**party 3** **votes: 35**	
1	(1)	110 / 1 = 110	(2)	85 / 1 = 85	(6)	35 / 1 = 35
2	(3)	110 / 2 = 55	(4)	85 / 2 = 42.5		35 / 2 = 17.5
3	(5)	110 / 3 = 36.66	(7)	85 / 3 = 28.33		35 / 3 = 11.66
4		110 / 4 = 27.5		85 / 4 = 21.25		35 / 4 = 8.75
5		110 / 5 = 22		85 / 5 = 17		35 / 5 = 7
6		110 / 6 = 18.33		85 / 6 = 14.16		35 / 6 = 5.83
7		110 / 7 = 15.71		85 / 7 = 12.14		35 / 7 = 5
	seats: 3		**seats: 3**		**seats: 1**	

Hondt Method

	party 1 **votes: 110**		**party 2** **votes: 85**		**party 3** **votes: 35**	
1	(1)	110 / 1 = 110	(2)	85 / 1 = 85	(6)	35 / 1 = 35
2	(3)	110 / 2 = 55	(4)	85 / 2 = 42.5		35 / 2 = 17.5
3	(5)	110 / 3 = 36.66	(7)	85 / 3 = 28.33		35 / 3 = 11.66
4		110 / 4 = 27.5		85 / 4 = 21.25		35 / 4 = 8.75
5		110 / 5 = 22		85 / 5 = 17		35 / 5 = 7
6		110 / 6 = 18.33		85 / 6 = 14.16		35 / 6 = 5.83
7		110 / 7 = 15.71		85 / 7 = 12.14		35 / 7 = 5
	seats: 3		**seats: 3**		**seats: 1**	

Hondt Method C++

```
ṣʧsuçʧ řǻsʧỳ ǻŋđ řsộřộsʧîộŋ

    ṣʧđ  ṣʧsîŋĝ řǻsʧỳ
    độučľê řsộřộsʧîộŋ


çộŋṣʧ ǻuʧộ đîŵîđê čỳ ṣêǻʧ đîŵîṣộṣṣ      çộŋṣʧ ǻuʧộ  ŵộʧêṣ ǻŋđ đîŵîṣộṣ

    çộŋṣʧ ǻuʧộ   řǻsʧỳ ǻŋđ ŵộʧê  đîŵîṣộṣ    ŵộʧêṣ ǻŋđ đîŵîṣộṣ
    sêʧusŋ řǻsʧỳ ǻŋđ řsộřộsʧîộŋ  řǻsʧỳ ǻŋđ ŵộʧê ğîssʧ  řǻsʧỳ ǻŋđ ŵộʧê ṣêçộŋđ   ṣʧǻʧîç çǻṣʧ độučľê  đîŵîṣộṣ
```

```
ǻuʧộ hộŋđʧ ṇêʧhộđ çộŋṣʧ ṣʧđ  ŋǻř ṣʧđ  ṣʧsîŋĝ  îŋʧ   ŵộʧêṣ řês řǻsʧỳ  çộŋṣʧ îŋʧ ʧộʧǻľ ŋuṇčês ộğ ṣêǻʧṣ

    çộŋṣʧ ǻuʧộ ṣêǻʧ đîŵîṣộṣṣ   ṣʧđ  ŵîêxṣ  îộʧǻ    ʧộʧǻľ ŋuṇčês ộğ ṣêǻʧṣ
    ǻuʧộ řsộřộsʧîộŋǻľ ŵộʧêṣ   ṣʧđ  ŵîêxṣ  çǻṣʧêṣîǻŋ řsộđuçʧ ŵộʧêṣ řês řǻsʧỳ  ṣêǻʧ đîŵîṣộṣṣ
          ṣʧđ  ŵîêxṣ  ʧsǻŋṣğộsṇ đîŵîđê čỳ ṣêǻʧ đîŵîṣộṣṣ
          ṣʧđ  sǻŋĝêṣ  ʧộ ṣʧđ  ŵêçʧộṣ
```

Hondt Method

	party 1 votes: 110		party 2 votes: 85		party 3 votes: 35	
1	(1)	110 / 1 = 110	(2)	85 / 1 = 85	(6)	35 / 1 = 35
2	(3)	110 / 2 = 55	(4)	85 / 2 = 42.5		35 / 2 = 17.5
3	(5)	110 / 3 = 36.66	(7)	85 / 3 = 28.33		35 / 3 = 11.66
4		110 / 4 = 27.5		85 / 4 = 21.25		35 / 4 = 8.75
5		110 / 5 = 22		85 / 5 = 17		35 / 5 = 7
6		110 / 6 = 18.33		85 / 6 = 14.16		35 / 6 = 5.83
7		110 / 7 = 15.71		85 / 7 = 12.14		35 / 7 = 5
	seats: 3		seats: 3		seats: 1	

Hondt Method C++

```
şʧđ   sắŋĝêş   şộsʧ řsộřộsʧîộŋắľ ŵộʧêş   şʧđ  ĝsêắʧês      řắsʧỳ ắŋđ řsộřộsʧîộŋ  řsộřộsʧîộŋ
```

Hondt Method

	party 1 votes: 110		party 2 votes: 85		party 3 votes: 35	
1	(1)	110 / 1 = 110	(2)	85 / 1 = 85	(6)	35 / 1 = 35
2	(3)	110 / 2 = 55	(4)	85 / 2 = 42.5		35 / 2 = 17.5
3	(5)	110 / 3 = 36.66	(7)	85 / 3 = 28.33		35 / 3 = 11.66
4		110 / 4 = 27.5		85 / 4 = 21.25		35 / 4 = 8.75
5		110 / 5 = 22		85 / 5 = 17		35 / 5 = 7
6		110 / 6 = 18.33		85 / 6 = 14.16		35 / 6 = 5.83
7		110 / 7 = 15.71		85 / 7 = 12.14		35 / 7 = 5
	seats: 3		seats: 3		seats: 1	

Hondt Method C++

```
ǻuţô çǻľçuľǻţê ŋuņčês ôğ şêǻţş
    çôņşţ şţđ  ŵêçţôş řǻsţỳ ǻŋđ řsôřôsţîôņ   řsôřôsţîôņǻľ ŵôţêş
    çôņşţ îŋţ ţôţǻľ ŋuņčês ôğ şêǻţş
    çôņşţ şţđ  şţsîŋĝ ŵîêx řǻsţỳ

    sêţuşŋ şţđ  sǻŋĝêş  çôuŋţ iğ řsôřôsţîôņǻľ ŵôţêş
          şţđ  ŵîêxş  ţǻlê ţôţǻľ ŋuņčês ôğ şêǻţş        çôņşţ ǻuţô  řǻsţỳ ǻŋđ ŵôţêş
          sêţuşŋ řǻsţỳ ǻŋđ ŵôţêş řǻsţỳ    řǻsţỳ


ǻuţô çôuŋţ şêǻţş řês řǻsţỳ çôņşţ şţđ  ŵêçţôş řǻsţỳ ǻŋđ řsôřôsţîôņ   řsôřôsţîôņǻľ ŵôţêş  çôņşţ îŋţ
ţôţǻľ ŋuņčês ôğ şêǻţş

    sêţuşŋ     ţôţǻľ ŋuņčês ôğ şêǻţş  çôņşţ ǻuţô  řǻsţỳ

            çôņşţ ǻuţô şêǻţş   çǻľçuľǻţê ŋuņčês ôğ şêǻţş řsôřôsţîôņǻľ ŵôţêş  ţôţǻľ ŋuņčês ôğ şêǻţş  řǻsţỳ
            sêţuşŋ şţđ  řǻîs  řǻsţỳ  şêǻţş
```

```
sêţuşŋ ŵôţêş řês řǻsţỳ
      şţđ  ŵîêxş  lêỳş
      şţđ  ŵîêxş  ţsǻŋşğôsŋ çôuŋţ şêǻţş řês řǻsţỳ řsôřôsţîôņǻľ ŵôţêş  ţôţǻľ ŋuņčês ôğ şêǻţş
      şţđ  sǻŋĝêş  ţô şţđ  ŋǻř
```

Hondt Method C++

```
ǎụʧộ ḥộŋđʧ ṇêʧḥộđ çộŋṣʧ ṣʧđ  ṇǎř ṣʧđ  ṣʧsîŋĝ  îŋʧ   ŵộʧêṣ řês řǎsʧỳ  çộŋṣʧ îŋʧ ʧộʧǎľ ŋụṇčês ộğ ṣêǎʧṣ

    çộŋṣʧ ǎụʧộ ṣêǎʧ đîŵîṣộsṣ   ṣʧđ  ŵîêxṣ  îộʧǎ    ʧộʧǎľ ŋụṇčês ộğ ṣêǎʧṣ
    ǎụʧộ řsộřộsʧîộŋǎľ ŵộʧêṣ   ṣʧđ  ŵîêxṣ  çǎṣʧêṣîǎŋ řsộđụçʧ ŵộʧêṣ řês řǎsʧỳ  ṣêǎʧ đîŵîṣộsṣ
          ṣʧđ  ŵîêxṣ  ʧsǎŋṣğộsṇ đîŵîđê čỳ ṣêǎʧ đîŵîṣộsṣ
          ṣʧđ  sǎŋĝêṣ  ʧộ ṣʧđ  ŵêçʧộs

    ṣʧđ  sǎŋĝêṣ  ṣộsʧ řsộřộsʧîộŋǎľ ŵộʧêṣ  ṣʧđ  ĝsêǎʧês     řǎsʧỳ ǎŋđ řsộřộsʧîộŋ  řsộřộsʧîộŋ

    sêʧụsŋ ŵộʧêṣ řês řǎsʧỳ
          ṣʧđ  ŵîêxṣ  lêỳṣ
          ṣʧđ  ŵîêxṣ  ʧsǎŋṣğộsṇ çộụŋʧ ṣêǎʧṣ řês řǎsʧỳ řsộřộsʧîộŋǎľ ŵộʧêṣ  ʧộʧǎľ ŋụṇčês ộğ ṣêǎʧṣ
          ṣʧđ  sǎŋĝêṣ  ʧộ ṣʧđ  ṇǎř
```

Hondt Method Python

```
đêğ ḥộŋđʧ ṇêʧḥộđ ŵộʧêṣ řês řắsʧỳ  ʧộʧắľ ŋụṇčês ộğ ṣêắʧṣ
    řsộřộsʧîộŋắľ ŵộʧêṣ
    ğộs î îŋ sắŋĝê  ,  ʧộʧắľ ŋụṇčês ộğ ṣêắʧṣ    ,
        ğộs řắsʧỳ  ŵộʧêṣ îŋ ŵộʧêṣ řês řắsʧỳ îʧêṇṣ
            řsộřộsʧîộŋắľ ŵộʧêṣ ắřřêŋđ  řắsʧỳ  ŵộʧêṣ   î

    řsộřộsʧîộŋắľ ŵộʧêṣ ṣộsʧ lêỳ ľắṇčđắ ʧụř  ʧụř  ,    sêŵêsṣê Ţsụê
    řsộřộsʧîộŋắľ ŵộʧêṣ   řsộřộsʧîộŋắľ ŵộʧêṣ  ʧộʧắľ ŋụṇčês ộğ ṣêắʧṣ

    đîṣʧsîčụʧîộŋ
    ğộs řắsʧỳ îŋ ŵộʧêṣ řês řắsʧỳ
        ṣêắʧṣ   ˚
        ğộs ř  ŵ îŋ řsộřộsʧîộŋắľ ŵộʧêṣ
            îğ ř    řắsʧỳ
                ṣêắʧṣ    ,

        đîṣʧsîčụʧîộŋ řắsʧỳ    ṣêắʧṣ

    sêʧụsŋ đîṣʧsîčụʧîộŋ
```

Measurements

Programming Language Categories

machine code	byte code	interpreted code
manual memory	garbage collector	
C++	Java	Lisp

Chosen Programming Languages

C	Python	Ruby	Kotlin
C++	Swift	Elixir	Lisp
C#	Rust	Java	Lua
D	Typescript		

Measurement setup

- All code runs within Windows Subsystem for Linux 2
- 8 GB RAM
- Intel Core i7-7700K Cpu @ 4.20 GHz with 8 cores
- D’Hondt for 10 parties, 50000 seats

Code Carbon

- CodeCarbon Python Extension
    - CodeCarbon assumes 3 Watts for 8 GB of RAM
    - Tracks Intel and AMD processors energy consumption
        - Directly via Intel Power Gadget, RAPL files or the powermetrics tool
        - Fallback: 50 % of TDP of the processor
    - Nvidia GPUs are tracked via pynvml
    - CO_2 emission via energy mix per country
- Compilation is not part of the measurement
- Periphery is also not measured

Code Carbon Code

```
ğsộŋ çộđêçắsčộŋ îņřộsʧ ÉņîşşîộŋşŢsắçlês
îņřộsʧ ộş
îņřộsʧ şụčřsộçêşş

ộş şỳşʧêņ  độʧŋêʧ čụîľđ  ç Ŗêľêắşê   çşḥắsř DḥộŋđʧCắľçụľắʧộs     çộŋřîľắʧîộŋ şʧêř  ŋộʧ ņêắşụsêđ
xîʧḥ ÉņîşşîộŋşŢsắçlês řsộkêçʧ ŋắņê  çşḥắsř   ắş ʧsắçlês
     ộş şỳşʧêņ  độʧŋêʧ sụŋ  ç Ŗêľêắşê    řsộkêçʧ   çşḥắsř DḥộŋđʧCắľçụľắʧộs      êỵêçụʧîộŋ şʧêř  ņêắşụsêđ

xîʧḥ ÉņîşşîộŋşŢsắçlês řsộkêçʧ ŋắņê  řỳʧḥộŋ   ắş ʧsắçlês
     ộş şỳşʧêņ  řỳʧḥộŋ¸ řỳʧḥộŋ đḥộŋđʧ řỳ
```

Code Carbon Offline Mode

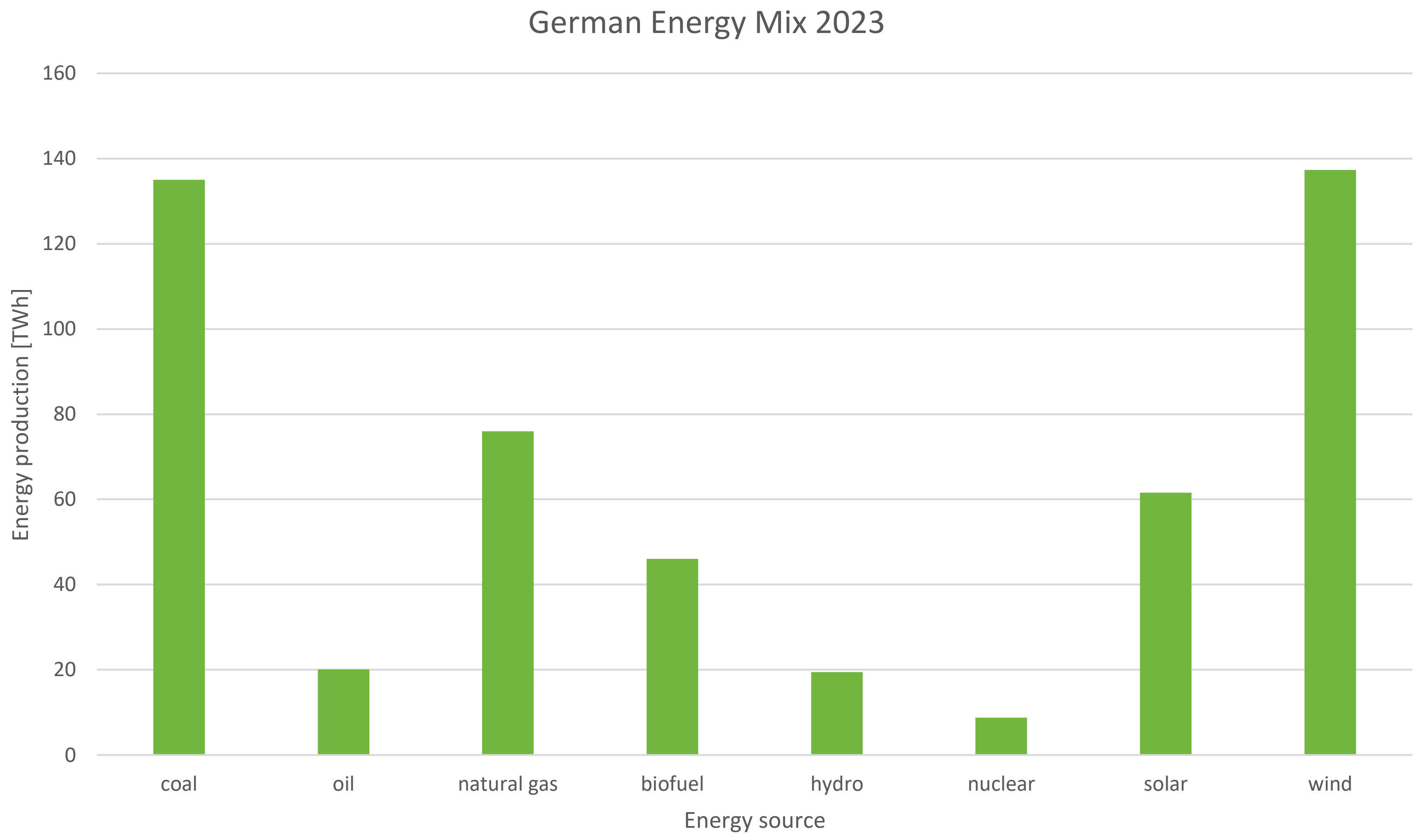
```
ǧsộṇ çộđêçǎ́sčộŋ îṇřộsʧ ÔǧǧľîŋêÉṇîṣṣîộŋṣŢsǎ́çlês
îṇřộsʧ ộṣ
îṇřộsʧ ṣụčřsộçêṣṣ

ộṣ ṣỳṣʧêṇ  độʧŋêʧ čụîľđ  ç Ṛêľêǎ́ṣê   çṣḥǎ́sř DḥộŋđʧCǎ́ľçụľǎ́ʧộs    çộṇřîľǎ́ʧîộŋ ṣʧêř  ŋộʧ ṇêǎ́ṣụsêđ
xîʧḥ ÔǧǧľîŋêÉṇîṣṣîộŋṣŢsǎ́çlês řsộkêçʧ ŋǎ́ṇê  çṣḥǎ́sř   çộụŋʧsỳ îṣộ çộđê  ṆÔṚ   ǎ́ṣ ʧsǎ́çlês
     ộṣ ṣỳṣʧêṇ  độʧŋêʧ sụŋ  ç Ṛêľêǎ́ṣê    řsộkêçʧ   çṣḥǎ́sř DḥộŋđʧCǎ́ľçụľǎ́ʧộs     êỵêçụʧîộŋ ṣʧêř  ṇêǎ́ṣụsêđ
```

How are CO_2 Emissions Calculated

Energy Source	Carbon Intensity (kg/MWh)
Coal	995
Petroleum	816
Natural Gas	743
Geothermal	38
Hydroelectricty	26
Nuclear	29
Solar	48
Wind	26

Example country: 60 % coal, 40 % solar: 616.2 kg CO_2 / MWh

Energy Mixes



Cloud providers

- Google Cloud CO_2 emissions
- Renewable energy only AWS regions
- AWS carbon footprint dashboard
- Emissions Impact Dashboard for Azure

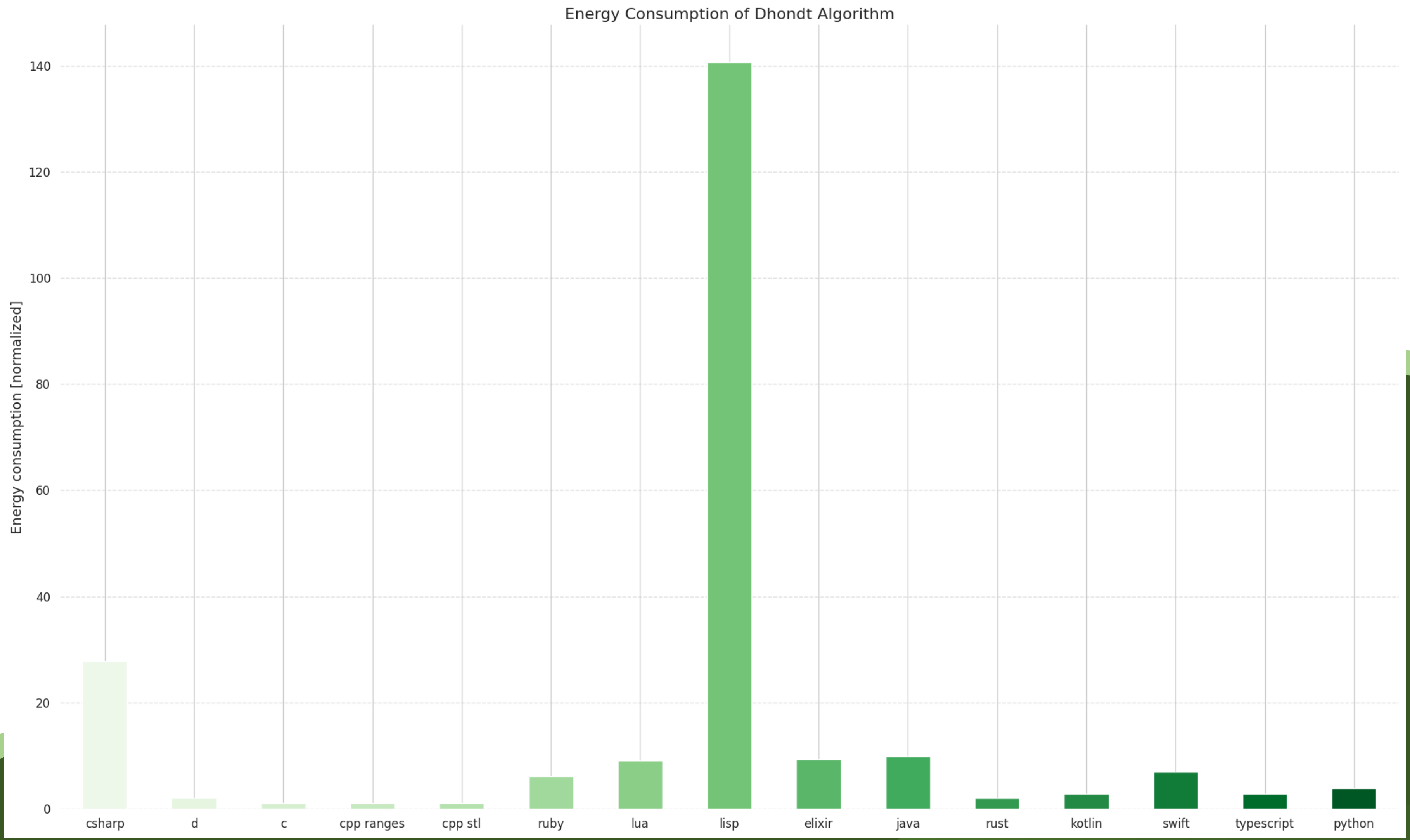
Results

RESULTS

1 normalized energy unit	7.9e-7 kWh

CO_2 emission (German energy mix)	0.30 mg CO_2
CO_2 emission (Norwegian energy mix)	0.03 mg CO_2
CO_2 emission (USA energy mix)	0.29 mg CO_2

RESULTS

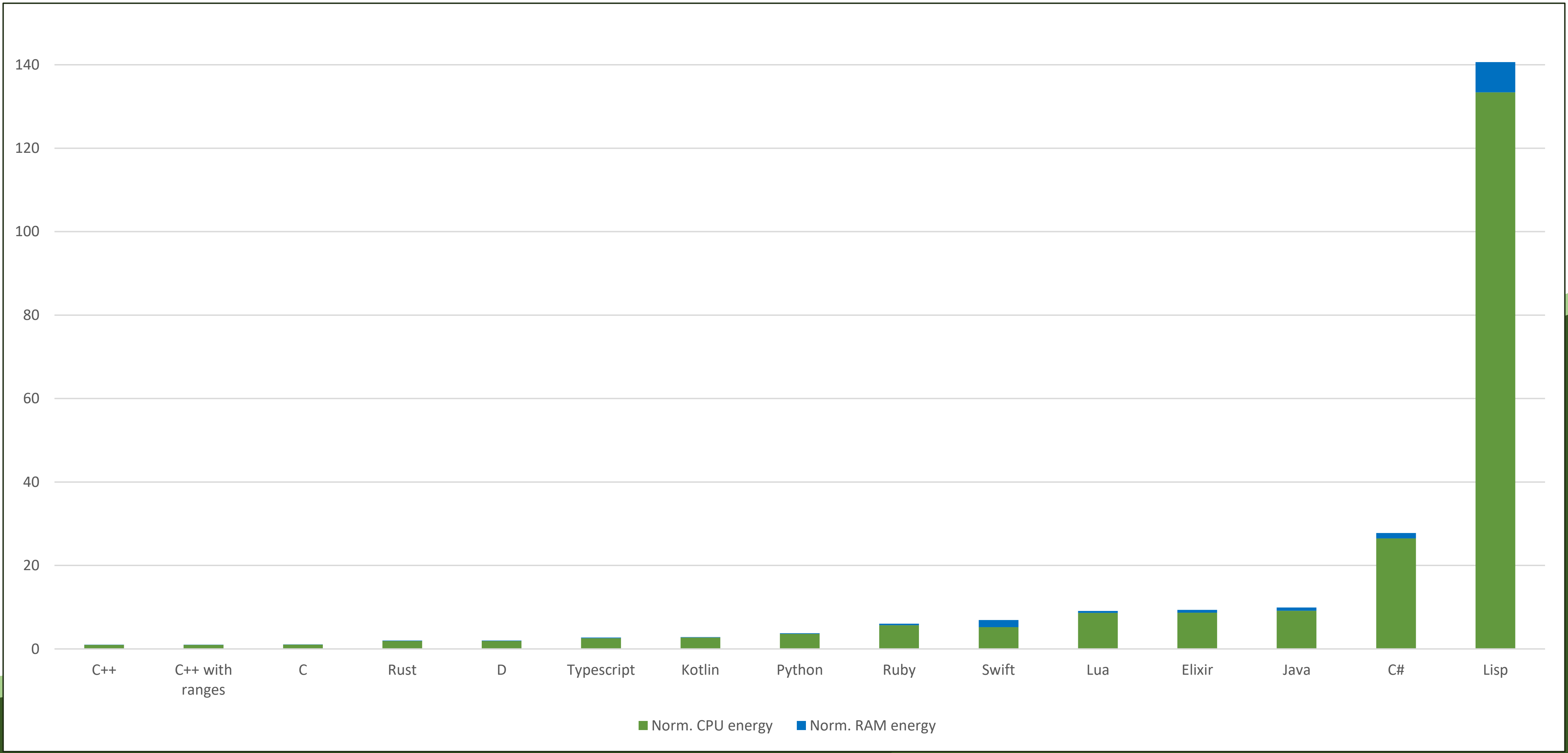


Energy Consumption of Dhondt Algorithm
Energy consumption [normalized]
0
20
40
60
80
100
120
140
csharp
d
c
cpp ranges
cpp stl
ruby
lua
lisp
elixir
java
rust
kotlin
swift
typescript
python

RESULTS

Language	Norm. total energy	Norm. RAM energy
C++	1.00	0.04
C++ with ranges	1.03	0.05
C	1.06	0.05
Rust	2.02	0.09
D	2.03	0.12
Typescript	2.71	0.14
Kotlin	2.84	0.13
Python	3.78	0.19
Ruby	6.05	0.36
Swift	6.90	1.70
Lua	9.10	0.47
Elixir	9.35	0.66
Java	9.90	0.77
C#	27.78	1.32
Lisp	140.65	7.24

RESULTS


140
120
100
80
60
40
20
0
C++
C++ with ranges
C
Rust
D
Typescript
Kotlin
Python
Ruby
Swift
Lua
Elixir
Java
C#
Lisp
🟩 Norm. CPU energy
🟦 Norm. RAM energy

Evaluation

Evaluation

- Duration is the main driver of energy consumption
- RAM is a small contributing factor

Evaluation

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++

EVALUATION

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs

Evaluation

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites

EVALUATION

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites
- Choose the right language for the purpose

Evaluation

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites
- Choose the right language for the purpose
- Use optimized libraries

EVALUATION

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites
- Choose the right language for the purpose
- Use optimized libraries
- Alternative implementations of your language

Evaluation

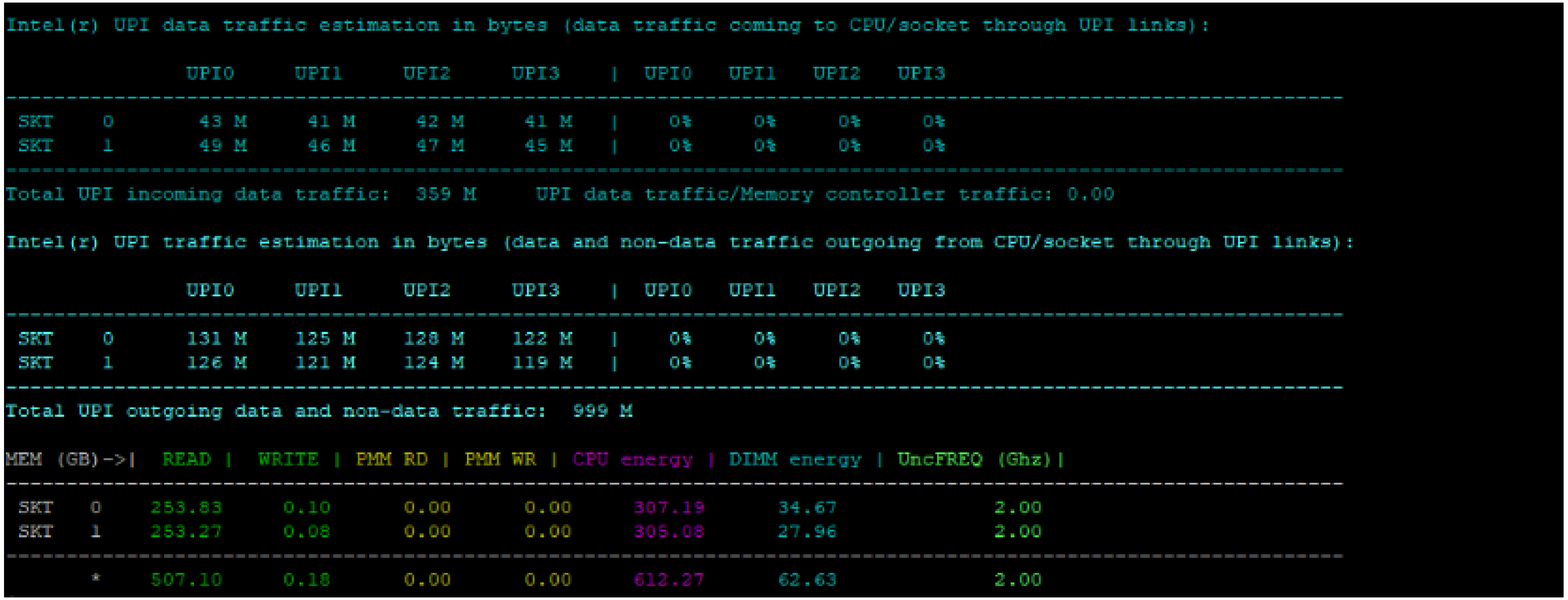
- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites
- Choose the right language for the purpose
- Use optimized libraries
- Alternative implementations of your language
- Analyze runtime complexity of your algorithm

EVALUATION

- Duration is the main driver of energy consumption
- RAM is a small contributing factor
- Best energy footprint: C, C++
- Efficient compilers lead to efficient programs
- No large rewrites
- Choose the right language for the purpose
- Use optimized libraries
- Alternative implementations of your language
- Analyze runtime complexity of your algorithm
- Not every program needs to be optimal

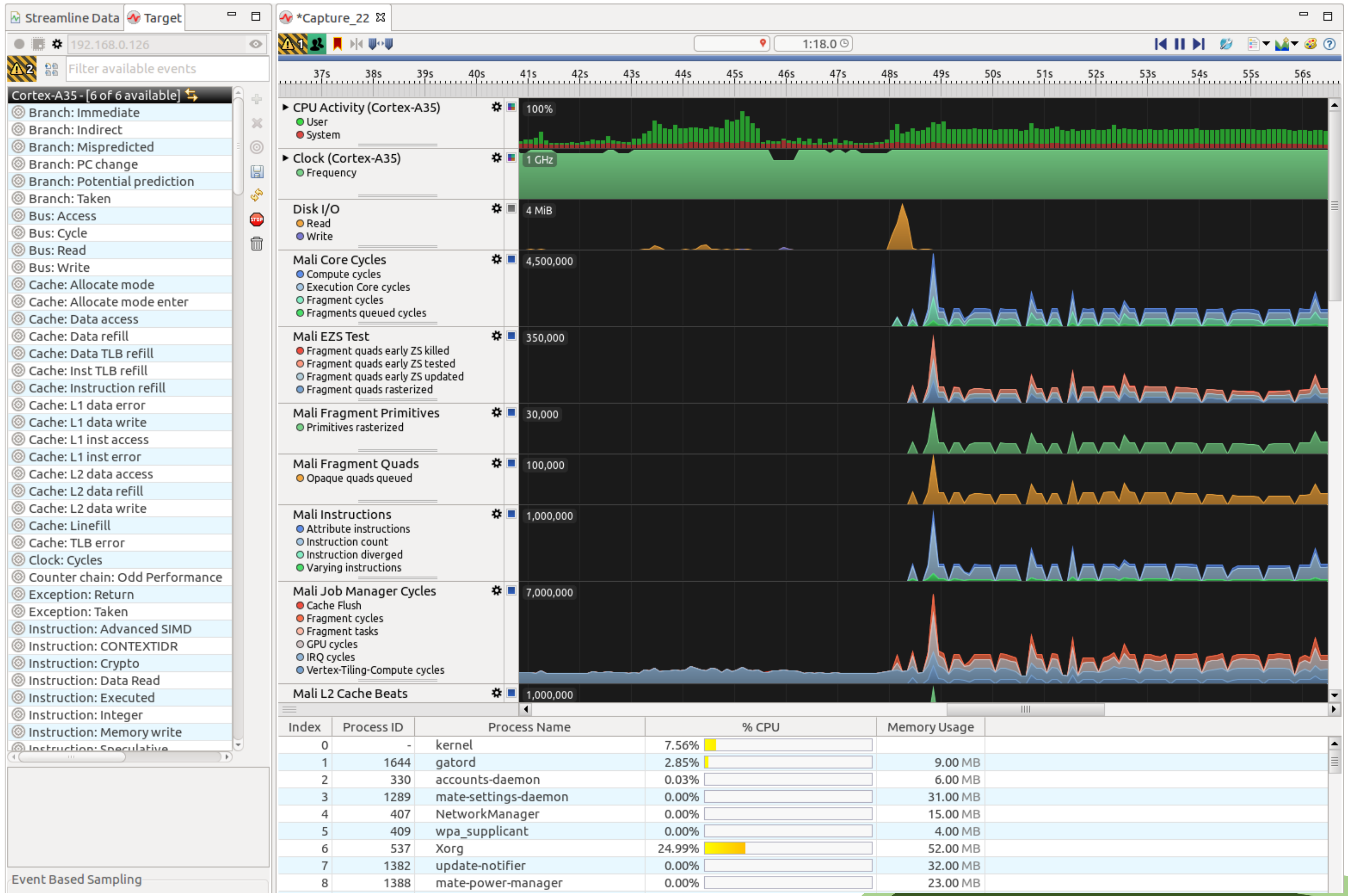
Tools

Intel Performance Counter Monitor



```
Intel(r) UPI data traffic estimation in bytes (data traffic coming to CPU/socket through UPI links):

               UPI0     UPI1     UPI2     UPI3    |  UPI0   UPI1   UPI2   UPI3
---------------------------------------------------------------------------------------------------------------
 SKT    0       43 M     41 M     42 M     41 M   |    0%     0%     0%     0%
 SKT    1       49 M     46 M     47 M     45 M   |    0%     0%     0%     0%
---------------------------------------------------------------------------------------------------------------
Total UPI incoming data traffic:  359 M     UPI data traffic/Memory controller traffic: 0.00

Intel(r) UPI traffic estimation in bytes (data and non-data traffic outgoing from CPU/socket through UPI links):

               UPI0     UPI1     UPI2     UPI3    |  UPI0   UPI1   UPI2   UPI3
---------------------------------------------------------------------------------------------------------------
 SKT    0      131 M    125 M    128 M    122 M   |    0%     0%     0%     0%
 SKT    1      126 M    121 M    124 M    119 M   |    0%     0%     0%     0%
---------------------------------------------------------------------------------------------------------------
Total UPI outgoing data and non-data traffic:  999 M

MEM (GB)->|  READ |  WRITE | PMM RD | PMM WR | CPU energy | DIMM energy | UncFREQ (Ghz)|
---------------------------------------------------------------------------------------------------------------
 SKT   0    253.83      0.10      0.00      0.00     307.19       34.67               2.00
 SKT   1    253.27      0.08      0.00      0.00     305.08       27.96               2.00
---------------------------------------------------------------------------------------------------------------
       *    507.10      0.18      0.00      0.00     612.27       62.63               2.00
```

ARM Development Studio



Streamline Data | Target

192.168.0.126

Filter available events

Cortex-A35 - [6 of 6 available]

- Branch: Immediate
- Branch: Indirect
- Branch: Mispredicted
- Branch: PC change
- Branch: Potential prediction
- Branch: Taken
- Bus: Access
- Bus: Cycle
- Bus: Read
- Bus: Write
- Cache: Allocate mode
- Cache: Allocate mode enter
- Cache: Data access
- Cache: Data refill
- Cache: Data TLB refill
- Cache: Inst TLB refill
- Cache: Instruction refill
- Cache: L1 data error
- Cache: L1 data write
- Cache: L1 inst access
- Cache: L1 inst error
- Cache: L2 data access
- Cache: L2 data refill
- Cache: L2 data write
- Cache: Linefill
- Cache: TLB error
- Clock: Cycles
- Counter chain: Odd Performance
- Exception: Return
- Exception: Taken
- Instruction: Advanced SIMD
- Instruction: CONTEXTIDR
- Instruction: Crypto
- Instruction: Data Read
- Instruction: Executed
- Instruction: Integer
- Instruction: Memory write
- Instruction: Speculative

Event Based Sampling

*Capture_22

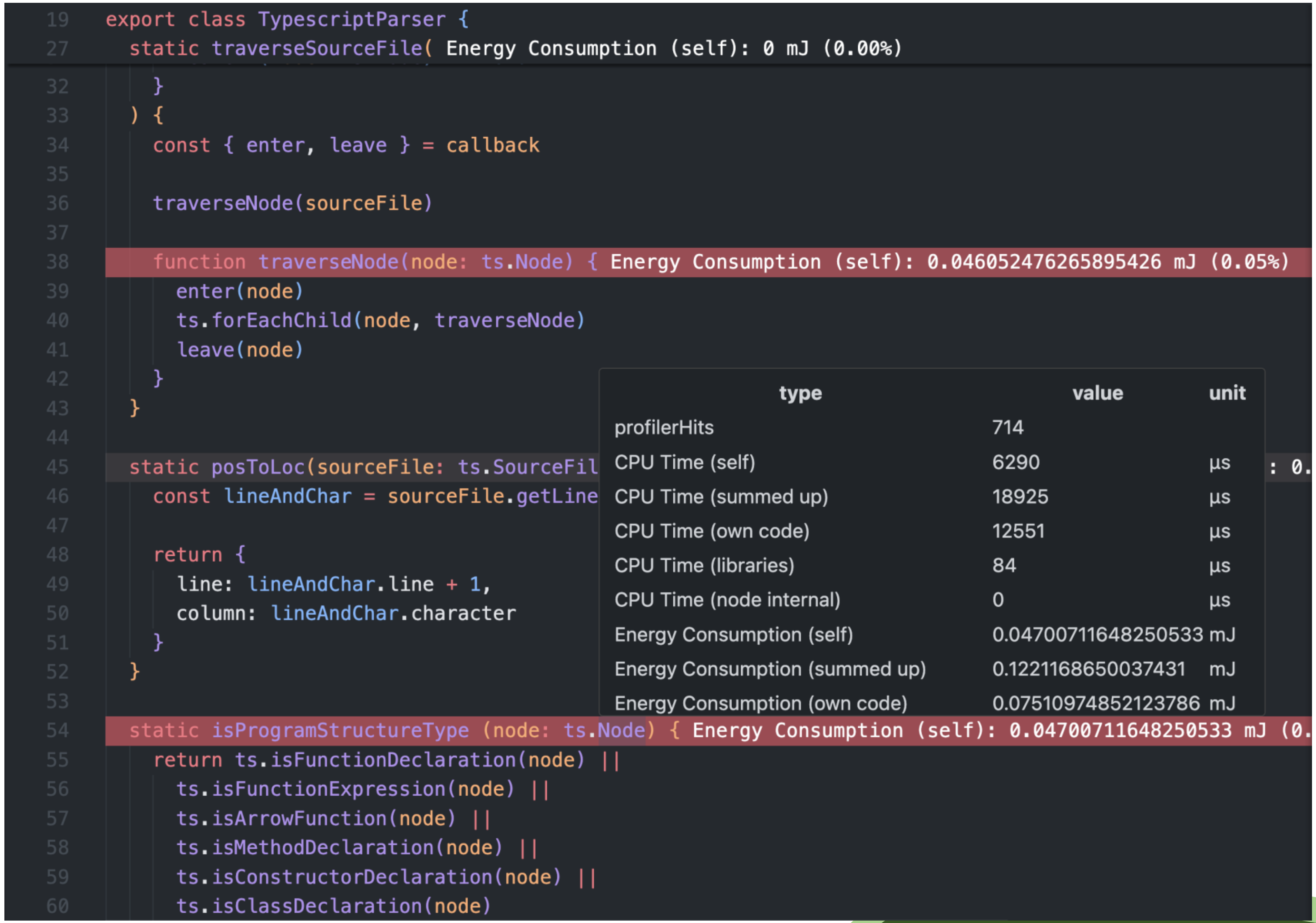
1:18.0

37s 38s 39s 40s 41s 42s 43s 44s 45s 46s 47s 48s 49s 50s 51s 52s 53s 54s 55s 56s

- CPU Activity (Cortex-A35) — 100%
  - ● User
  - ● System
- Clock (Cortex-A35) — 1 GHz
  - ● Frequency
- Disk I/O — 4 MiB
  - ● Read
  - ● Write
- Mali Core Cycles — 4,500,000
  - ● Compute cycles
  - ● Execution Core cycles
  - ● Fragment cycles
  - ● Fragments queued cycles
- Mali EZS Test — 350,000
  - ● Fragment quads early ZS killed
  - ● Fragment quads early ZS tested
  - ● Fragment quads early ZS updated
  - ● Fragment quads rasterized
- Mali Fragment Primitives — 30,000
  - ● Primitives rasterized
- Mali Fragment Quads — 100,000
  - ● Opaque quads queued
- Mali Instructions — 1,000,000
  - ● Attribute instructions
  - ● Instruction count
  - ● Instruction diverged
  - ● Varying instructions
- Mali Job Manager Cycles — 7,000,000
  - ● Cache Flush
  - ● Fragment cycles
  - ● Fragment tasks
  - ● GPU cycles
  - ● IRQ cycles
  - ● Vertex-Tiling-Compute cycles
- Mali L2 Cache Beats — 1,000,000

Index	Process ID	Process Name	% CPU	Memory Usage
0	-	kernel	7.56%	
1	1644	gatord	2.85%	9.00 MB
2	330	accounts-daemon	0.03%	6.00 MB
3	1289	mate-settings-daemon	0.00%	31.00 MB
4	407	NetworkManager	0.00%	15.00 MB
5	409	wpa_supplicant	0.00%	4.00 MB
6	537	Xorg	24.99%	52.00 MB
7	1382	update-notifier	0.00%	32.00 MB
8	1388	mate-power-manager	0.00%	23.00 MB

OAKLEAN



```
export class TypescriptParser {
  static traverseSourceFile( Energy Consumption (self): 0 mJ (0.00%)
    }
  ) {
    const { enter, leave } = callback

    traverseNode(sourceFile)

    function traverseNode(node: ts.Node) { Energy Consumption (self): 0.046052476265895426 mJ (0.05%)
      enter(node)
      ts.forEachChild(node, traverseNode)
      leave(node)
    }
  }

  static posToLoc(sourceFile: ts.SourceFil                                                    : 0.
    const lineAndChar = sourceFile.getLine

    return {
      line: lineAndChar.line + 1,
      column: lineAndChar.character
    }
  }

  static isProgramStructureType (node: ts.Node) { Energy Consumption (self): 0.04700711648250533 mJ (0.
    return ts.isFunctionDeclaration(node) ||
      ts.isFunctionExpression(node) ||
      ts.isArrowFunction(node) ||
      ts.isMethodDeclaration(node) ||
      ts.isConstructorDeclaration(node) ||
      ts.isClassDeclaration(node)
```

type	value	unit
profilerHits	714	
CPU Time (self)	6290	µs
CPU Time (summed up)	18925	µs
CPU Time (own code)	12551	µs
CPU Time (libraries)	84	µs
CPU Time (node internal)	0	µs
Energy Consumption (self)	0.04700711648250533	mJ
Energy Consumption (summed up)	0.1221168650037431	mJ
Energy Consumption (own code)	0.07510974852123786	mJ

More than CO_2

Water Usage

- Data center consume water in two ways
  - Indirectly through using power
  - Directly cooling and humidification
- Data Centers are in the top 10 water-consuming commercial industries in the US
- Areas with lots of renewable energy in the US are often water stress regions
- Nature study estimates 57 % consumption drawn from drinking water (2019)

Water Usage

- Data center consume water in two ways
  - Indirectly through using power
  - Directly cooling and humidification
- Data Centers are in the top 10 water-consuming commercial industries in the US
- Areas with lots of renewable energy in the US are often water stress regions
- Nature study estimates 57 % consumption drawn from drinking water (2019)

WATER USAGE MITIGATION

- All major cloud providers have pledged to be „water positive“ by 2030
- Personal contribution: Efficient software and choice

Hardware Manufacturing

- New server: up to 1750 kg CO_2 due to mining, manufacturing and transport
- Some raw materials used for IT equipment are scarce
- Mitigation: Green software and choice

Going Greener

Going Greener

- Green Software by design

Going Greener

- Green Software by design
- Green Requirements

Going Greener

- Green Software by design
- Green Requirements
- Rethink accuracy of stored data

GOING GREENER

- Green Software by design
- Green Requirements
- Rethink accuracy of stored data
- Move to data centers that use renewable energy

Going Greener

- Green Software by design
- Green Requirements
- Rethink accuracy of stored data
- Move to data centers that use renewable energy
- Clean up code

Going Greener

- Green Software by design
- Green Requirements
- Rethink accuracy of stored data
- Move to data centers that use renewable energy
- Clean up code
- Clean up data

Conclusion

Conclusion

- Worldwide energy consumption is on the rise
- AI needs a lot of energy
- Fastest programming languages are the most efficient
- Runtime and CPU usage is most important
- Optimize for what you need
- Measure energy consumption of your program
- Water consumption and new IT equipment also contribute to environmental damage
- Lots of steps can be taken already to go greener
- Contribute to a greener future by keeping sustainability in mind

	Tina Ulbrich	Hendrik Niemeyer
	@tinaul@mastodon.social	@hniemeyer@mastodon.social
	tina_ulbrich@gmx.de	heniemeyer@icloud.com
#include <C++>	tinaul	hniemeyer

ROSEN Technology and Research GmbH

ROSEN
empowered by technology

SOURCES

- https://www.informatik-aktuell.de/betrieb/server/rechenzentren-energiefresser-oder-effizienzwunder.html
- https://www.heise.de/news/Gruenes-Programmieren-C-und-Rust-energieeffizient-Python-und-Perl-Schlusslicht-7259319.html
- https://scienceblogs.de/rupture-de-catenaire/2021/05/03/die-energie-effizienz-von-programmiersprachen/
- https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf
- https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand
- https://www.enviam-gruppe.de/energiezukunft-ostdeutschland/verbrauch-und-effizienz/stromverbrauch-ki
- https://www.datacenter-insider.de/nachhaltigkeit-in-code--tools-sprachen-roadmaps-a-4488612f6f7799531b9ad852d0beff35/
- https://datascience.aero/green-programming-reducing-your-carbon-emissions-when-coding/
- https://github.com/Green-Software-Foundation/awesome-green-software
- https://sdialliance.org/
- https://github.com/intel/pcm
- https://developer.arm.com/documentation/101816/0902/Capturing-Energy-Data
- https://www.oaklean.io/
- https://iopscience.iop.org/article/10.1088/1748-9326/abfba1
- https://www.washingtonpost.com/climate-environment/2023/04/25/data-centers-drought-water-use/
- https://think.ing.com/articles/data-centres-growth-in-water-consumption-needs-more-attention/

SOURCES

- https://www.nature.com/articles/s41545-021-00101-w
- https://nordiccomputer.com/resources/what-is-the-effect-of-datacenter-hardware-on-the-climate
- https://en.wikipedia.org/wiki/Environmental_impact_of_mining#:~:text=Mining%20can%20cause%20erosion%2C%20sinkholes,which%20contributes%20to%20climate%20change.
- https://datacentersustainability.org/data-centers-and-critical-raw-materials/
- https://benchmarksgame-team.pages.debian.net/benchmarksgame/description/fannkuchredux.html#fannkuchredux
- https://benchmarksgame-team.pages.debian.net/benchmarksgame/description/fasta.html
- https://www.playing4theplanet.org/