9. Green SE – Part II Sustainable Software Engineering **CS4295**



Luís Cruz L.Cruz@tudelft.nl

SustainableSE 2024



Energy patterns for mobile apps Energy efficiency across programming languages Carbon-aware datacenters

pitfalls.

While learning about these works, try to be critical about them and find their

- We have seen that measuring energy consumption is not trivial
- energy efficiency
- At the same time, every now and then there are some efforts to improve expertise.

• How can we reuse these efforts?

• It is not practical considering that developers have other priorities above

energy efficiency in some cases. This is time consuming and requires

Energy Patterns for Mobile Apps

https://tqrg.github.io/energy-patterns/

Empirical Software Engineering (2019) 24:2209–2235 https://doi.org/10.1007/s10664-019-09682-0

Catalog of energy patterns for mobile applications



Luis Cruz¹ 💿 • Rui Abreu²

Published online: 5 March 2019 © Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Software engineers make use of design patterns for reasons that range from performance to code comprehensibility. Several design patterns capturing the body of knowledge of best practices have been proposed in the past, namely creational, structural and behavioral patterns. However, with the advent of mobile devices, it becomes a necessity a catalog of design patterns for energy efficiency. In this work, we inspect commits, issues and pull requests of 1027 Android and 756 iOS apps to identify common practices when improving energy efficiency. This analysis yielded a catalog, available online, with 22 design patterns related to improving the energy efficiency of mobile apps. We argue that this catalog might be of relevance to other domains such as Cyber-Physical Systems and Internet of Things. As a side contribution, an analysis of the differences between Android and iOS devices shows that the Android community is more energy-aware.

Keywords Mobile applications · Energy efficiency · Energy patterns · Catalog · Open source software

1 Introduction

The importance of providing developers with more knowledge on how they can modify mobile apps to improve energy efficiency has been reported in previous works (Li and Halfond 2014; Robillard and Medvidovic 2016). In particular, mobile apps often have energy requirements but developers are unaware that energy-specific design patterns do exist (Manotas et al. 2016). Moreover, developers have to support multiple platforms while providing a similar user experience (An et al. 2018).

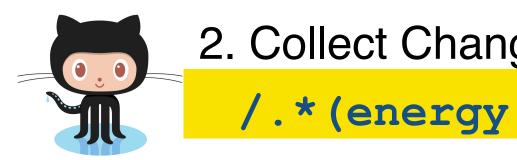
Communicated by: David Lo, Meiyappan Nagappan, Sebastiano Panichella, and Fabio Palomba

🖂 Luis Cruz luiseruz@fc.up.pt



Methodology







3. Manual Ref

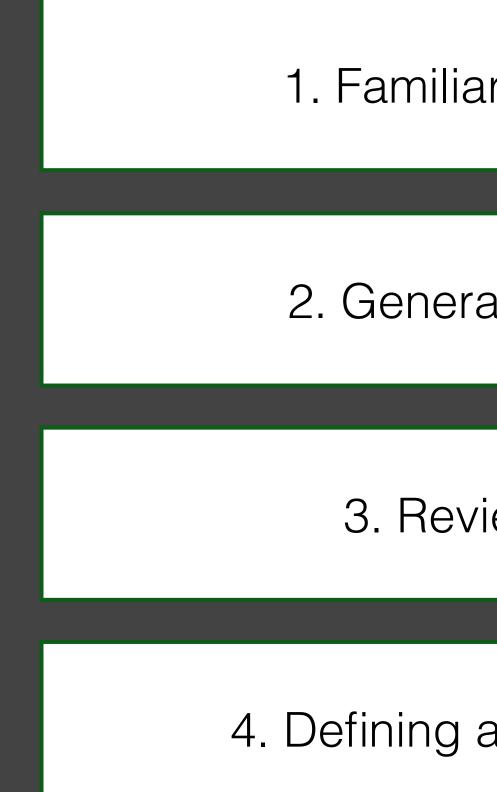


4. The



App Collection	1783 apps
nges With Potential Interest	6028
<pre>y power battery).*/</pre>	changes
finement of Subjects of	1563
Interest	changes
	431
ematic Analysis	reusable
	changes
of Energy Patterns	22 patterns

Thematic Analysis



1. Familiarization with data

2. Generating initial labels

3. Reviewing themes

4. Defining and naming themes

• Energy Pattern: design pattern to improve energy efficiency.

- 22 energy patterns.

 Each pattern is described by Context, Solution, Example, References from literature, and Occurences (links to code changes from git repositories).

Energy Patterns for Mobile Apps

A visualization with prevalence and co-occurence of patterns can be found here. News This catalog has been **accepted** to the Journal of Empirical Software Engineering. Check out the preprint.

🗎 tqrg.github.io

← show all patterns **Dark UI Colors**

Provide a dark UI color theme to save battery on devices with AMOLED screens.

TODAY 0 Context MEADING Screen is one of the major source of power LOREM consumption on mobile devices. Apps that require heavy usage of screen (e.g., reading apps) can have a IPSUM big impact on battery life. BUICE

Solution

Provide a UI with dark background colors. This is particularly beneficial for mobile devices with AMOLED screens, which are more energy efficient when displaying dark colors. In some cases, it might be reasonable to allow users to choose between a light and a dark theme. The dark theme can also be activated using a special trigger (e.g., when battery is running low).

Display a menu

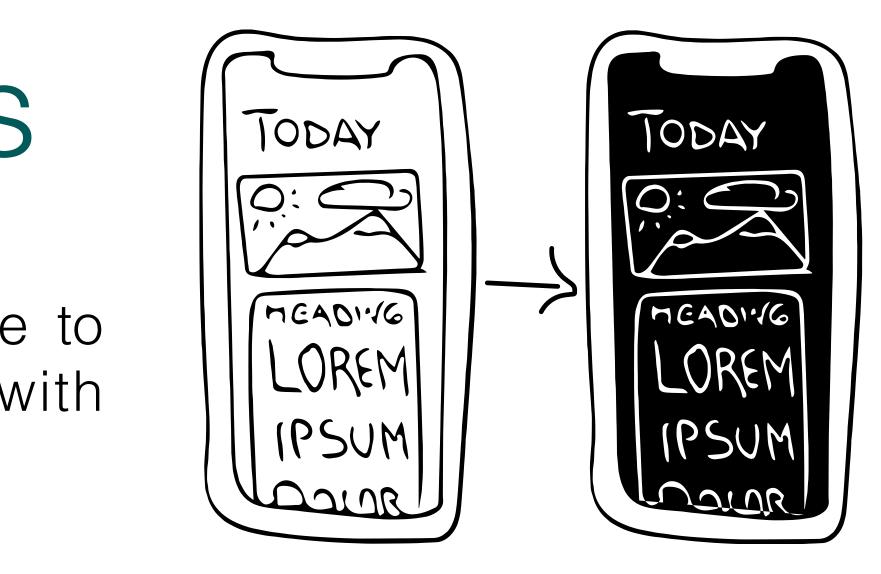
https://tqrg.github.io/energy-patterns



Dark UI Colors

Provide a dark UI color theme to save battery on devices with AMOLED screens.

- have a substantial negative impact on battery life.
- background using light colors to display text. [...]



• Context: [...] Apps that require heavy usage of screen can

• **Solution:** Provide a UI theme with dark background colors. [...]

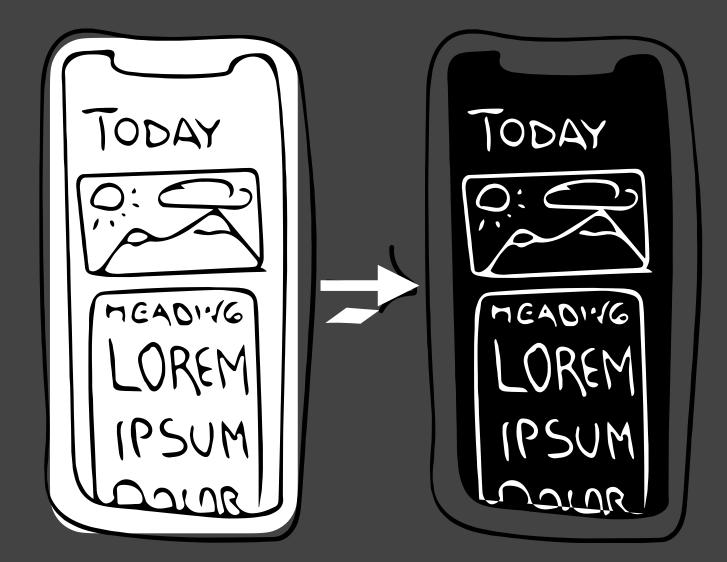
• **Example:** In a reading app, provide a theme with a dark

Dark UI Colors

Provide a dark UI color theme to save battery on devices with AMOLED screens.

- have a substantial negative impact on battery life.
- background using light colors to display text. [...]





• Context: [...] Apps that require heavy usage of screen can

• **Solution:** Provide a UI theme with dark background colors. [...]

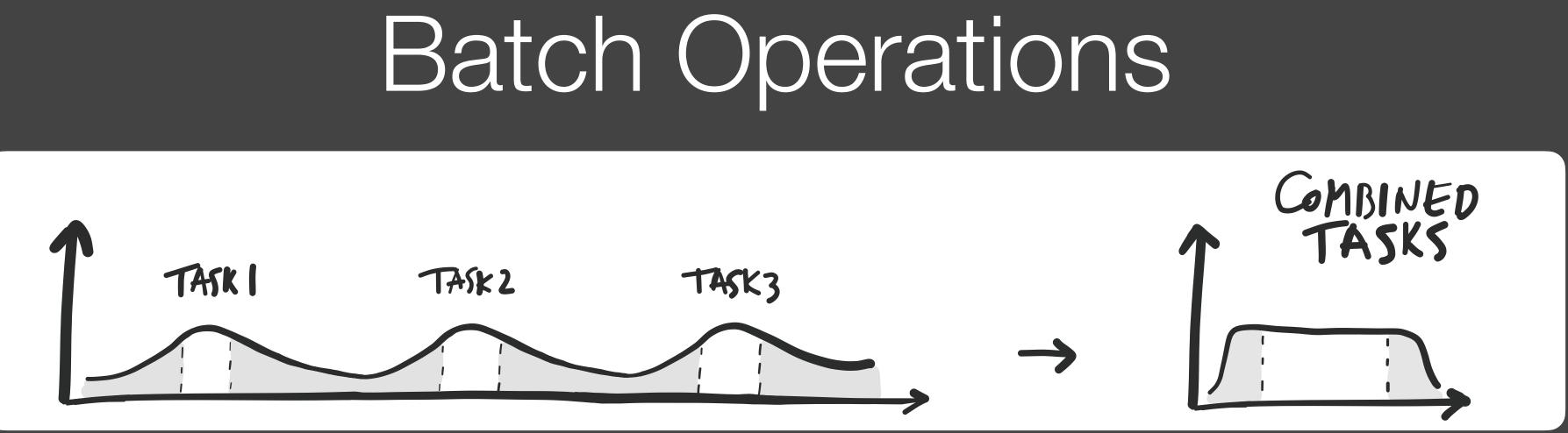
• Example: In a reading app, provide a theme with a dark

Dynamic Retry Delay

Whenever an attempt to access a resource fails, increase the time interval before retrying.

- **Context:** [...] In a mobile app, when a given resource is unavailable, the app will unnecessarily try to connect the resource for a number of times, leading to unnecessary power consumption.
- Solution: Increase retry interval after each failed connection. [...]
- **Example:** Consider a mobile app that provides a news feed and the app is not able to reach the server to collect updates. [...] use the Fibonacci series to increase the time between attempts.





- **Context:** Executing operations separately leads to extraneous tail energy consumptions
- **Solution:** Bundle multiple operations in a single one. [...]
- **Example:** Use system provided APIs to schedule background tasks. These APIs, guarantee that device will exit sleep mode only when there is a reasonable amount of work to do or when a given task is urgent. [...]

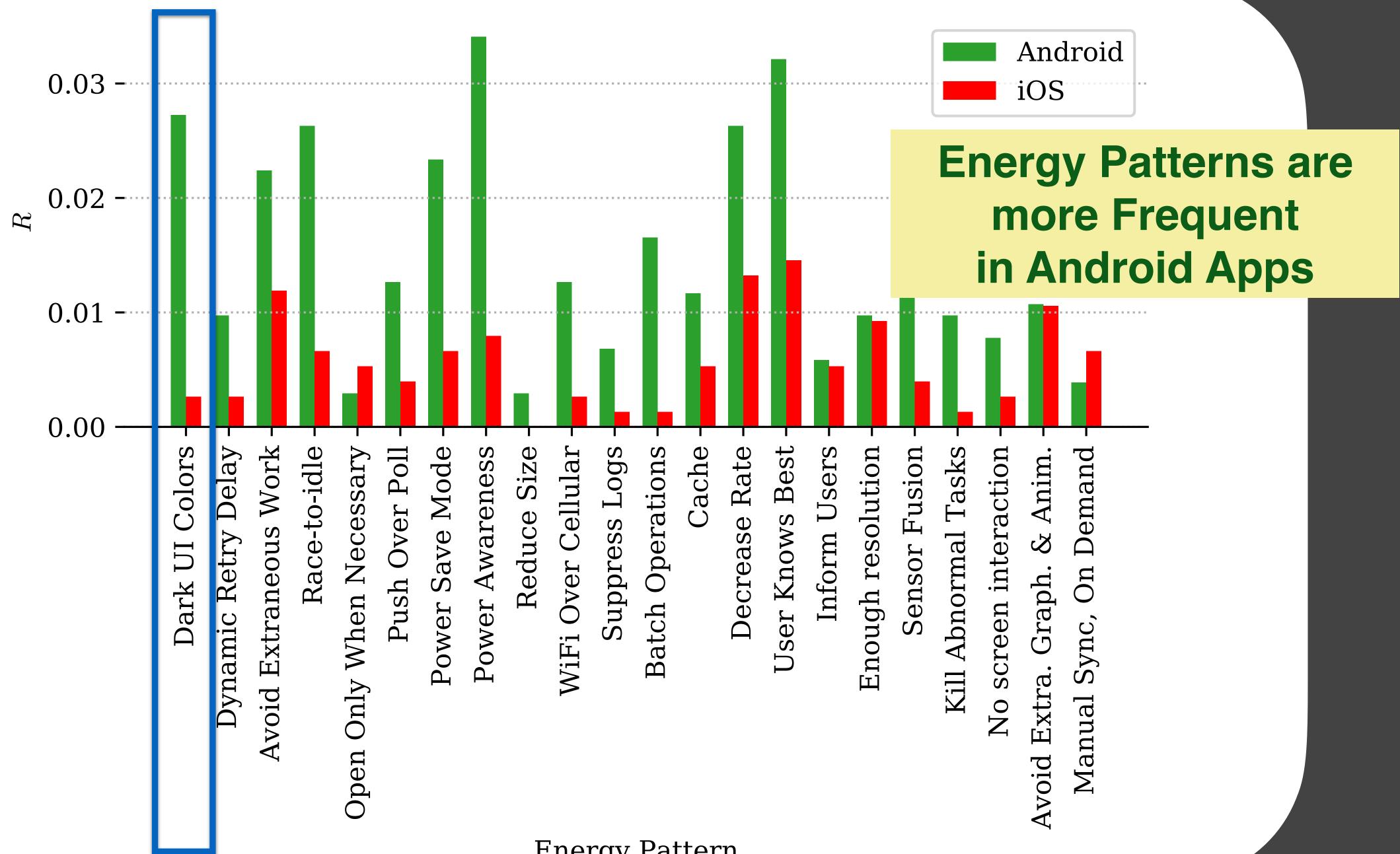
Avoid Extraneous Graphics and Animations Despite being important to improve user experience, graphics and animations are battery intensive and should be used with moderation.

- and animations. [...]
- applicable. [...]
- possible.

Context: Mobile apps that feature impressive graphics

• Solution: Study the importance of graphics and animations to the user experience and reduce them when

• **Example:** Resort to low frame rates for animations when



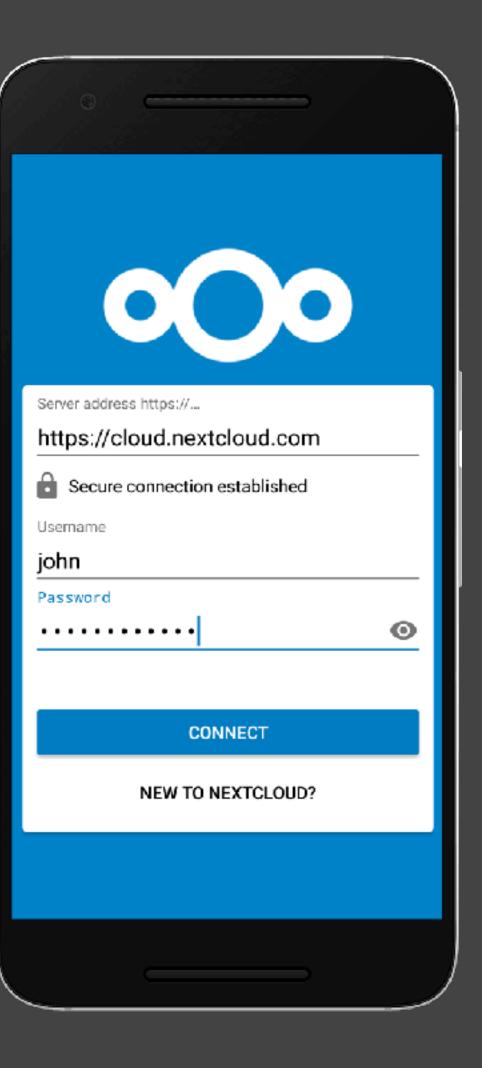
Energy Pattern

Example case: Nextcloud



Nextcloud





10" 😻 🛙	Nextcloud	`∎ «%	15:43
	Documents		
	2.5 MB, 51 minutes ago	~	
	Photos 3.0 MB, 51 minutes ago	\leq	
5	hirschmilch_drog_hous.pls 128 B, 5 Mar	4	
	Nextcloud.mp4 452 KB, 51 minutes ago	<	* * * *
revocut	Nextcloud.png 36 KB, 51 minutes ago	<	• • •
٨	Nextcloud Manual.pdf 4.4 MB, 51 minutes ago	r,	•
	4 files, 2 folders	Ŧ	
		$\overline{}$	

Example case: Nextcloud

- It is mostly used for backup. No real-time collaboration is needed.
- need all the battery you can get.
- <u>https://github.com/nextcloud/android/commit/</u> 8bc432027e0d33e8043cf40192203203a40ca29c

• Users complain that sometimes they go on a trip and Nextcloud drains their battery. Users consider uninstalling the app when battery life is essential.

• File sync can be energy-greedy. Send large files to the server, long 3G/4G data connections.

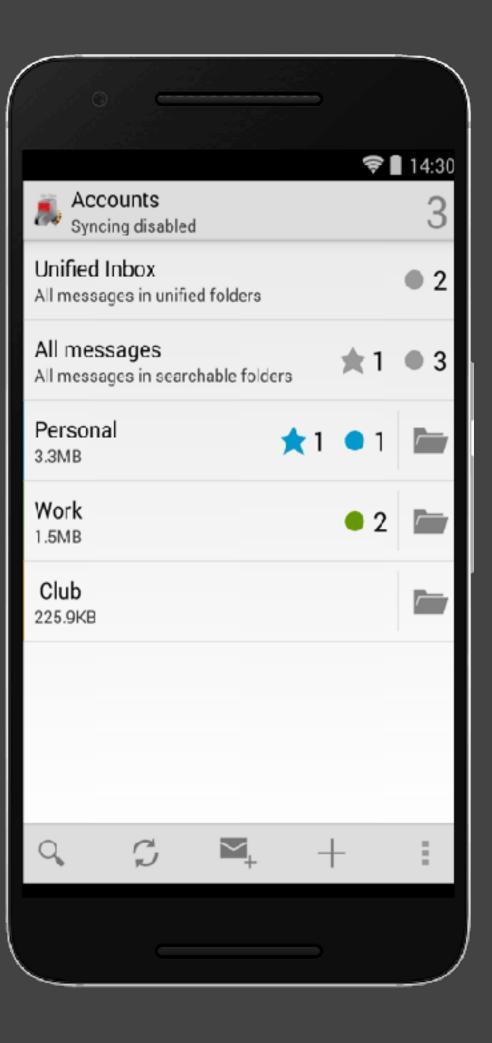
• Energy requirements vary depending on context and user. Some days you really

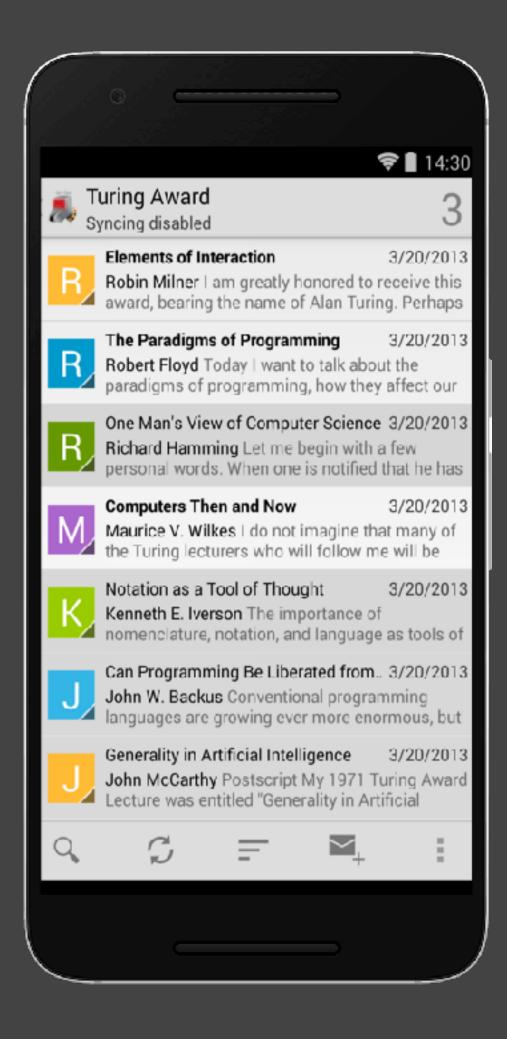


Example case: K-9 mail









Example case: K-9 mail

- Some users noticed that K-9 mail was spending more energy than usual.
- A user that was having issues with a personal mail server noticed that K-9 mail was the one of the most energy-greedy apps. IMAP IDLE protocol for real-time notifications.
- When a connection is not possible, the app automatically retries later.
- <u>https://github.com/k9mail/k-9/commit/</u> 86f3b28f79509d1a4d613eb39f60603e08579ea3



EcoAndroid

- Plugin for IntelliJ (Android Studio)
 - Dynamic Retry Delay
 - Push Over Poll
 - Reduce Size
 - Cache
 - Avoid Graphics and Animations

JET BRAINS	Marketplace	Edu Courses	Themes	Plugin Ideas	Build Plugins	📥 Sign In	
EcoOndroi	Refactoring Android ECOAnd Add Software Reliability Lab Versions Reviews	roid		C	ompatible with IntelliJ	Ge I IDEA (Ultimate, Co Educational), Andro	m
	en en () () () () () () () () () ()			and as 1 The second	en process terri deci deci docardo nel terri a deci decivit conservativano especial estr deci arterna e os e deci d la del decivit d		•



Energy Efficiency Across Programming Languages

Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva

https://sites.google.com/view/energy-efficiency-languages

Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Marco Couto

Rui Percira HASLab/INESC TEC Universidade do Minho, Portugal ruipercira@di.uminho.pt

Jácome Cunha NOVA LINCS, DI, FCT Univ. Nova de Lisboa, Portugal jacome@fct.unl.pt HASLab/INESC TEC Universidade do Minho, Portugal marco.l.couto@inesctec.pt

João Paulo Fernandes Release/LISP, CISUC Universidade de Coimbra, Portugal jpf@dei.uc.pt Francisco Ribeiro, Rui Rua HASLab/INESC TEC Universidade do Minho, Portugal fribeiro@di.uminho.pt rrua@di.uminho.pt

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

Abstract

This paper presents a study of the runtime, memory usage and energy consumption of twenty seven well-known software languages. We monitor the performance of such languages using ten different programming problems, expressed in each of the languages. Our results show interesting findings, such as, slower/faster languages consuming less/more energy, and how memory usage influences energy consumption. We show how to use our results to provide software engineers support to decide which language to use when energy efficiency is a concern.

CCS Concepts → Software and its engineering → Software performance; General programming languages;

Keywords Energy Efficiency, Programming Languages, Language Benchmarking, Green Software

ACM Reference Format:

Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy Efficiency scross Programming Languages: How Do Energy, Time, and Memory Relate?. In Proceedings of 2017 ACM SIGPLAN International Conference on Software Language Engineering (SLE'17). ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3136014.3136031

1 Introduction

Software language engineering provides powerful techniques and tools to design, implement and evolve software languages. Such techniques aim at improving programmers

Dermission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies hear this notice and the full clustion on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is pennitted. To copy otherwise, or republish, to past on servers or to redistribute to lists, requires prior specific permission and/or a fee. Bequest permissions from permissions@scm.org.

SLE 17, October 27–36, 2017, Vancouver, Canada # 2017 Association for Computing Machinery. ACM ISBN 978-1-4501-5528-4517/19... \$15.00 https://doi.org/10.1145/3136014.3136031 productivity - by incorporating advanced features in the language design, like for instance powerful modular and type systems - and at efficiently execute such software - by developing, for example, aggressive compiler optimizations. Indeed, most techniques were developed with the main goal of helping software developers in producing faster programs. In fact, in the last century *performance* in software languages was in almost all cases synonymous of *fast execution time* (embedded systems were probably the single exception).

In this century, this reality is quickly changing and software energy consumption is becoming a key concern for computer manufacturers, software language engineers, programmers, and even regular computer users. Nowadays, it is usual to see mobile phone users (which are powerful computers) avoiding using CPU intensive applications just to save battery/energy. While the concern on the computers' energy efficiency started by the hardware manufacturers, it quickly became a concern for software developers too [28]. In fact, this is a recent and intensive area of research where several techniques to analyze and optimize the energy consumption of software systems are being developed. Such techniques already provide knowledge on the energy efficiency of data structures [15, 27] and android language [25], the energy impact of different programming practices both in mobile [18, 22, 31] and desktop applications [26, 32], the energy efficiency of applications within the same scope [2, 17]. or even on how to predict energy consumption in several software systems [4, 14], among several other works.

An interesting question that frequently arises in the software energy efficiency area is whether a faster program is also an energy efficient program, or not. If the answer is yes, then optimizing a program for speed also means optimizing it for energy, and this is exactly what the compiler construction community has been hardly doing since the very beginning of software languages. However, energy consumption does not depends only on execution time, as shown in the equation $E_{nergy} = T_{ime} \times P_{ower}$. In fact, there are several research works showing different results regarding



- Is a faster programming language also more energy efficient?
- Comparing different programming languages is not an easy task. lacksquare
 - They differ in many mechanisms:
 - Interpreted vs Compiled
 - Optimisations at the compiler level
 - Virtual machine
 - Garbage collector ullet
 - Available libraries

Research Questions

- Can we compare the energy efficiency of software languages?
- Is the faster language always the most energy efficient?
- How does memory usage relate to energy consumption? (We don't cover this one)
- Can we automatically decide what is the best programming language considering energy, time, and memory usage?
- How do the results of our energy consumption analysis of programming languages gathered from rigorous performance benchmarking solutions compare to results of average day-to-day solutions?



Nethodology

The Computer Language Benchmarks Game

<u>https://benchmarksgame-team.pages.debian.net/benchmarksgame/</u>

••• • • < >	D 🔒 benchmarksgame-team.pages.debian.net 网络 🖒	⊕ ₫ + ©
	The Computer Language 22.02 Benchmarks Game	
	"Which programming language is fastest?" Let's go measure <u>benchmark programs</u> !	
	Fastest means lowest … <i>cpu</i> seconds?	
	<u>C++</u> <u>C</u> <u>Rust</u> <u>Fortran</u> <u>Julia</u> <u>C#</u>	
	Ada Chapel Haskell Go Pascal	
	<u>F# OCaml Java Swift JavaScript</u>	
	Lisp Dart Racket PHP Erlang	
	Lua Python Smalltalk Ruby Perl	
Display a menu	<pre>{ box plot summary charts }</pre>	



Problems in the Computer Language Benchmarks Game

Benchmark

n-body
fannkuch-redux
spectral-norm
mandelbrot
pidigits
regex-redux
fasta
k-nucleotide
reverse-complement
binary-trees
chameneos-redux
meteor-contest
thread-ring

Description

Double precision Indexed access to Eigenvalue using Generate Mandell Streaming arbitrat Match DNA 8mer Generate and wri Hashtable update Read DNA sequen Allocate, traverse Symmetrical threa Search for solutio Switch from threa

	Input
N-body simulation	50M
o tiny integer sequence	12
; the power method	5,500
lbrot set portable bitmap file	16,000
ary precision arithmetic	10,000
rs and substitute magic patterns	fasta output
rite random DNA sequences	25M
e and k-nucleotide strings	fasta output
nces, write their reverse-complement	fasta output
e and deallocate many binary trees	21
ead rendezvous requests	6M
ons to shape packing puzzle	2,098
ead to thread passing one token	50M



- 27 Programming languages across different paradigms
 - Functional (e.g., Ocaml, F#, Haskell)
 - Imperative (e.g., C, Go, Python)
 - Object-oriented (e.g., C++, C#, Java)
 - Scripting (or interpretative) (e.g., JavaScript, Python, Ruby)
 - (These are not mutual exclusive)
- Intel RAPL's C library to measure energy consumption

- Execute each benchmark solution 10 times.
 - Collect energy data and timestamps.
- Two-minute interval between executions

binary-trees	_ (7)			1
	Energy (J)	Time (ms)	Ratio (J/ms)	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust \Downarrow_2	49.07	1263	0.039	180
(c) Fortran ↑1	69.82	2112	0.033	133
(c) Ada ↓1	95.02	2822	0.034	197
(c) Ocaml $\downarrow_1 \Uparrow_2$	100.74	3525	0.029	148
(v) Java ↑1 ↓16	111.84	3306	0.034	1120
(v) Lisp ↓3 ↓3	149.55	10570	0.014	373
(v) Racket $\downarrow_4 \Downarrow_6$	155.81	11261	0.014	467
(i) Hack $\uparrow_2 \downarrow_9$	156.71	4497	0.035	502
(v) C# $\downarrow_1 \Downarrow_1$	189.74	10797	0.018	427
(v) F# ↓3 ↓1	207.13	15637	0.013	432
(c) Pascal ↓3 ↑5	214.64	16079	0.013	256
(c) Chapel \uparrow_5 \Uparrow_4	237.29	7265	0.033	335
(v) Erlang ↑ ₅ ↑ ₁	266.14	7327	0.036	433
(c) Haskell $\uparrow_2 \Downarrow_2$	270.15	11582	0.023	494
(i) Dart $\downarrow_1 \Uparrow_1$	290.27	17197	0.017	475
(i) JavaScript ↓2 ↓4	312.14	21349	0.015	916
(i) TypeScript $\downarrow_2 \Downarrow_2$	315.10	21686	0.015	915
(c) Go ↑3 ↑13	636.71	16292	0.039	228
(i) Jruby ↑ ₂ ↓ ₃	720.53	19276	0.037	1671
(i) Ruby ↑5	855.12	26634	0.032	482
(i) PHP ↑3	1,397.51	42316	0.033	786
(i) Python ↑15	1,793.46	45003	0.040	275
(i) Lua \downarrow_1	2,452.04	209217	0.012	1961
(i) Perl \uparrow_1	3,542.20	96097	0.037	2148
(c) Swift		n.e.		

	Energy (J)	Time (ms)	Ratio (J/ms)	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust \Downarrow_2	49.07	1263	0.039	180
(c) Fortran ↑1	69.82	2112	0.033	133
(c) Ada \Downarrow_1	95.02	2822	0.034	197
(c) Ocaml $\downarrow_1 \Uparrow_2$	100.74	3525	0.029	148
(v) Java ↑1 ↓16	111.84	3306	0.034	1120
(v) Lisp $\downarrow_3 \Downarrow_3$	149.55	10570	0.014	373
(v) Racket $\downarrow_4 \Downarrow_6$	155.81	11261	0.014	467
(i) Hack ↑2 ↓9	156.71	4497	0.035	502
(v) C# $\downarrow_1 \Downarrow_1$	189.74	10797	0.018	427
(v) F# ↓3 ↓1	207.13	15637	0.013	432
(c) Pascal $\downarrow_3 \uparrow_5$	214.64	16079	0.013	256
(c) Chapel \uparrow_5 \Uparrow_4	237.29	7265	0.033	335
(v) Erlang 个5	266.14	7327	0.036	433
(c) Haskell $\uparrow_2 \Downarrow_2$	270.15	11582	0.023	494
(i) Dart $\downarrow_1 \Uparrow_1$	290.27	17197	0.017	475
(i) JavaScript $\downarrow_2 \Downarrow_4$	312.14	21349	0.015	916
(i) TypeScript $\downarrow_2 \Downarrow_2$	315.10	21686	0.015	915
(c) Go ↑3 ↑13	636.71	16292	0.039	228
(i) Jruby $\uparrow_2 \Downarrow_3$	720.53	19276	0.037	167
(i) Ruby ↑5	855.12	26634	0.032	482

	Total				
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Normalized global results for Energy, Time, and Memory.

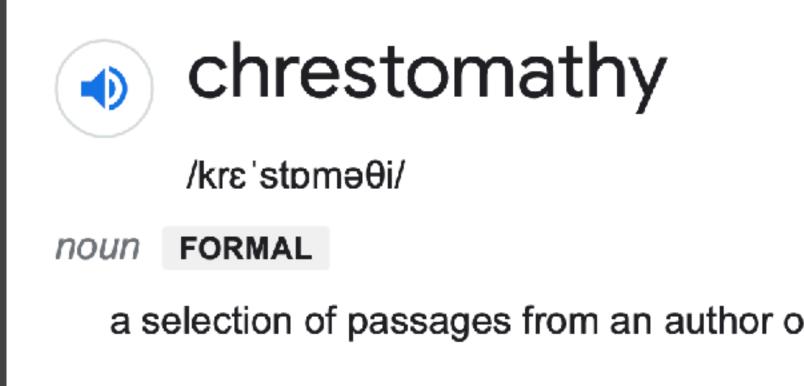
Critical thinking

- There is no doubt this is an excellent study. Yet, as any excellent study, there's a lot we can **discuss** and **criticise** constructively.
- What kind of issues you see in drawing conclusions from such a table of results?
 - Is the benchmark representative of the most common usage behaviour?
 - Are the implemented solutions representative?
 - Does it make sense to use the average to compare energy consumption across different problems?



Reproducing with Rosetta Code

Rosetta Code is a programming chrestomathy repository \bullet



- \bullet
- same task is solved in a language you are not familiar with.

(?)

a selection of passages from an author or authors, designed to help in learning a language.

900 usecases/tasks solved across 700 different programming languages

Purpose: if you know a programming language we can easily learn how the

Remove-duplicates

	Energy (J)	Time (ms)
(c) Rust	0.01	1
(c) C++	0.12	5
(c) C	0.14	10
(c) Go	0.32	13
(i) Lua	0.51	21
(i) Perl	1.31	53
(i) JavaScript	1.73	73
(v) Erlang	2.36	96
(v) Java	2.96	214
(i) PHP	2.99	121
(i) Python	4.93	206
(i) Ruby	6.13	259
(v) Racket	7.54	318

Rosetta Code Globa	l Ranking
Position	Language
1	С
2	Pascal
3	Ada
4	Rust
5	C++, Fortran
6	Chapel
7	OCaml, Go
8	Lisp
9	Haskell, JavaScript
10	Java
11	PHP
12	Lua, Ruby
13	Perl
14	Dart, Racket, Erlang
15	Python

Revisiting Research Questions

- Can we compare the energy efficiency of software languages?
- Is the faster language always the most energy efficient?
- How does memory usage relate to energy consumption?
- Can we automatically decide what is the best programming language considering \bullet energy, time, and memory usage?
- How do the results of our energy consumption analysis of programming languages gathered from rigorous performance benchmarking solutions compare to results of average day-to-day solutions?
 - What would happen if we cherry picked the tasks?



Carbon-Aware Computing for Datacenters

Ana Radovanovicí, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne

https://sites.google.com/view/energy-efficiency-languages

20 un DC SS 50v 90 2

Carbon-Aware Computing for Datacenters Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao,

Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne

an electricity grid varies by time of day and substantially varies by location due to the types of generation. Networked collections of warehouse scale computers, sometimes called Hyperscale Computing, emit more carbon than needed if operated without regard to these variations in carbon intensity. This paper introduces Google's system for Carbon-Intelligent Compute anagement, which actively minimizes electricity-based carbon ootprint and power infrastructure costs by delaying temporally flexible workloads. The core component of the system is a suite of analytical pipelines used to gather the next day's carbon intensity orecasts, train day-ahead demand prediction models, and use risk-aware optimization to generate the next day's carbon-aware of Google's internal workloads that tolerate delays as long as Virtual Capacity Curves (VCCs) for all datacenter clusters across their work gets completed within 24 hours. Typical examples Google's fleet. VCCs impose hourly limits on resources available o temporally flexible workloads while preserving overall daily capacity, enabling all such workloads to complete within a day. Data from operation shows that VCCs effectively limit hourly many of the tasks that make information found through Google capacity when the grid's energy supply mix is carbon intensive and delay the execution of temporally flexible workloads to 'greener" times.

aware compute management, power management.

I. INTRODUCTION

worldwide has been continuously growing, now accounting dependencies and placement consequences, which generally for approximately 1% of total electricity usage [1]. Between have high uncertainty and are hard to predict (i.e., we do 2010 and 2018, global datacenter workloads and compute not know in advance what jobs will run over the course instances increased more than sixfold [1]. In response, new of the next day). Fortunately, in spite of high uncertainties efficiency are required to limit their growing environmental, consumption at a cluster-level and beyond have demonstrated economic and performance impacts [2], [3].

emissions reductions in electricity grids. A considerable fraction global costs, carbon footprint, and future resource utilization. of compute workloads have flexibility in both when and where The workload scheduler implementation must be simple (i.e., they run. Given that emissions from electricity production with as little as possible computational complexity in making vary substantially by time and location [4]-[7], we can exploit placement decisions) to cope with the high volume of job load flexibility to consume power where and when the grid requests. is less carbon intensive. By effectively managing its load, The core of the carbon-aware load shaping mechanism is a the datacenter industry can contribute to a more robust, set of cluster-level [9] Virtual Capacity Curves (VCCs), which resilient, and cost-efficient energy system, facilitating grid are hourly resource usage limits that serve to shape each cluster decarbonization. Electric grid operators, in turn, can possibly resource and power usage profile over the following day. These benefit by as much as EUR 1B/year [8].

time and space can decrease peak demand for resources and and their uncertainty, hourly carbon intensity forecasts [10],

The authors are with Google, Inc. Mountain View, CA, 94043 (Email: anaradovanovic@google.com, ross@google.com, ischneid@google.com, bokanchen@google.com, alexandredu@google.com, binzroy@google.com, usage limits set by energy providers for different datacenters diyuexiao@google.com, haridasan@google.com, phfhung@google.com, across Google's fleet. ncare@google.com, stalukdar@google.com, ericmullen@google.com, kendalsmith@google.com, meacottman@google.com, walfredo@google.com)

Abstract-The amount of CO2 emitted per kilowatt-hour on power. Since datacenters are planned based on peak power and resource usage, smaller peaks reduce the need for more capacity. Not only does this save money, it also reduces environmental impacts

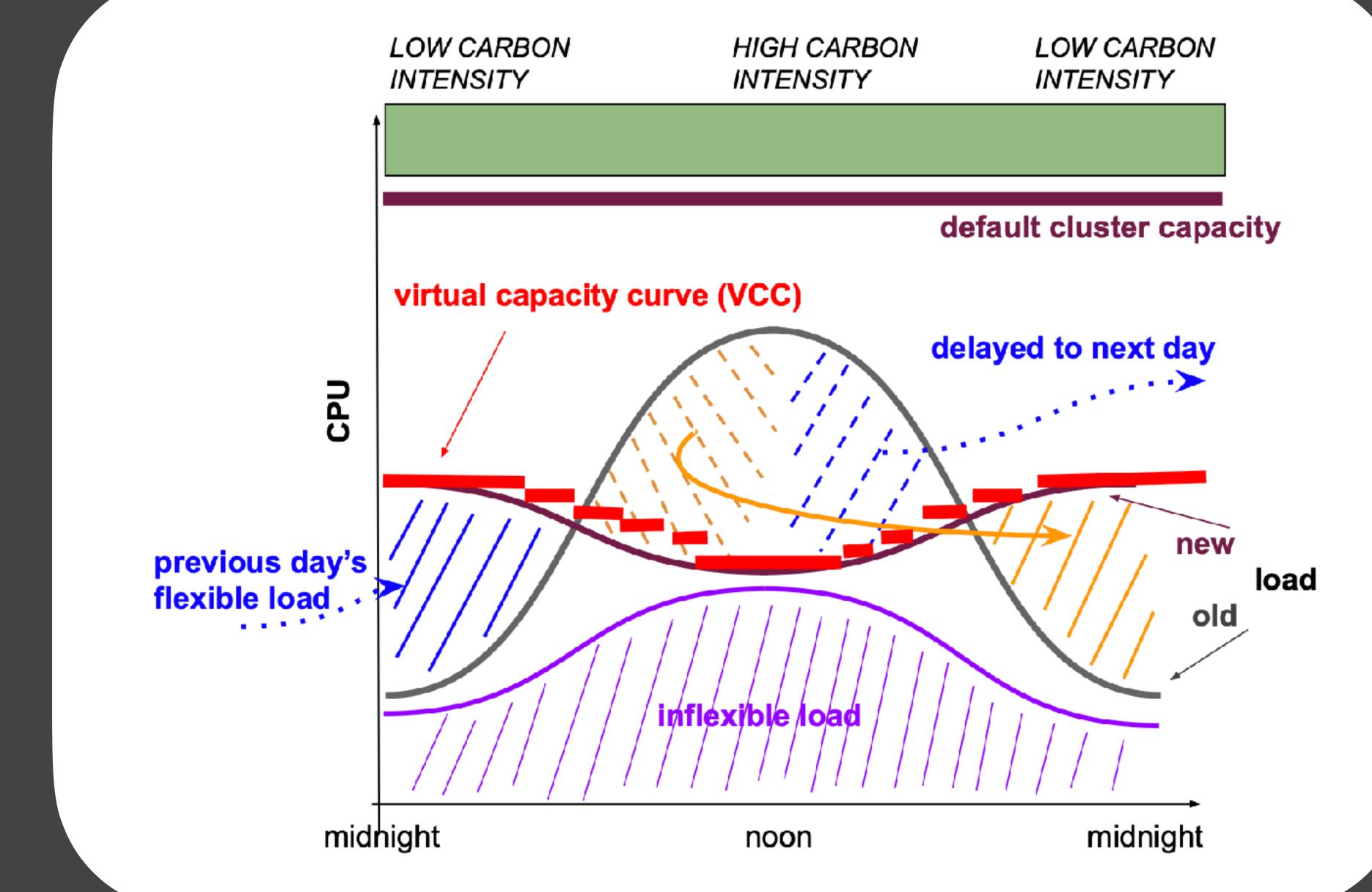
This paper describes the methodology and principles behind Google's system for Carbon-Intelligent Compute management, which reduces grid carbon emissions from Google's datacenter electricity use and reduces operating costs by increasing resource and power efficiency. To accomplish this goal, the system harnesses the temporal flexibility of a significant fraction of such workloads are data compaction, machine learning, simulation, and data processing (e.g., video processing) pipelines products more accessible and useful. Note that other loads include user-facing services (Search, Maps and YouTube) that people rely on around the clock, and our cloud customers' Index Terms-Datacenter computing, carbon- and efficiency- workloads running in allocated Virtual Machines (VMs), which are not temporally flexible and therefore not affected by the new system.

Workloads are comprised of compute jobs. The system needs Demand for computing resources and datacenter power to consider compute jobs' arrival patterns, resource usage, methodologies for increasing datacenter power and energy at the job level, Google's flexible resource usage and daily to be quite predictable within a day-ahead forecasting horizon. The datacenter industry has the potential to facilitate carbon The aggregate outcome of job scheduling ultimately affects

limits are computed using an optimization process that takes Furthermore, shifting execution of flexible workloads in account of aggregate flexible and inflexible demand predictions explicit characterization of business and environmental targets infrastructure and workload performance expectations, and

> The cluster-level VCCs are pushed to all of Google's datacenter clusters prior to the start of the next day, where they



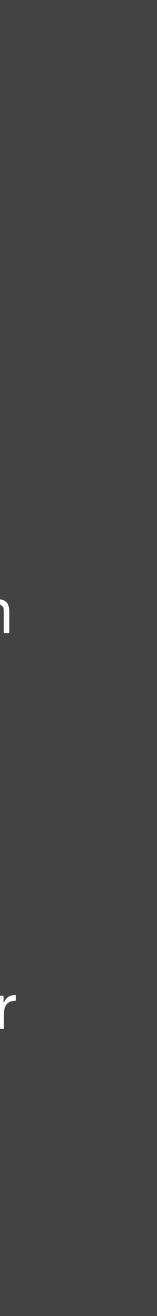




- Google's Carbon-Intelligent Computing System (CICS)
- Main idea: use carbon-intensity data to shift datacenter jobs in time ightarrow
- Typically, job schedulers use a metric of cluster capacity to schedule a job in a particular cluster.
 - factors in Carbon intensity
 - usage and assigns it to a cluster if the VCC is not exceeded.

CICS overrides this metric with the virtual capacity curve (VCC) that

• When a new job comes in, the scheduler estimates its CPU load and power

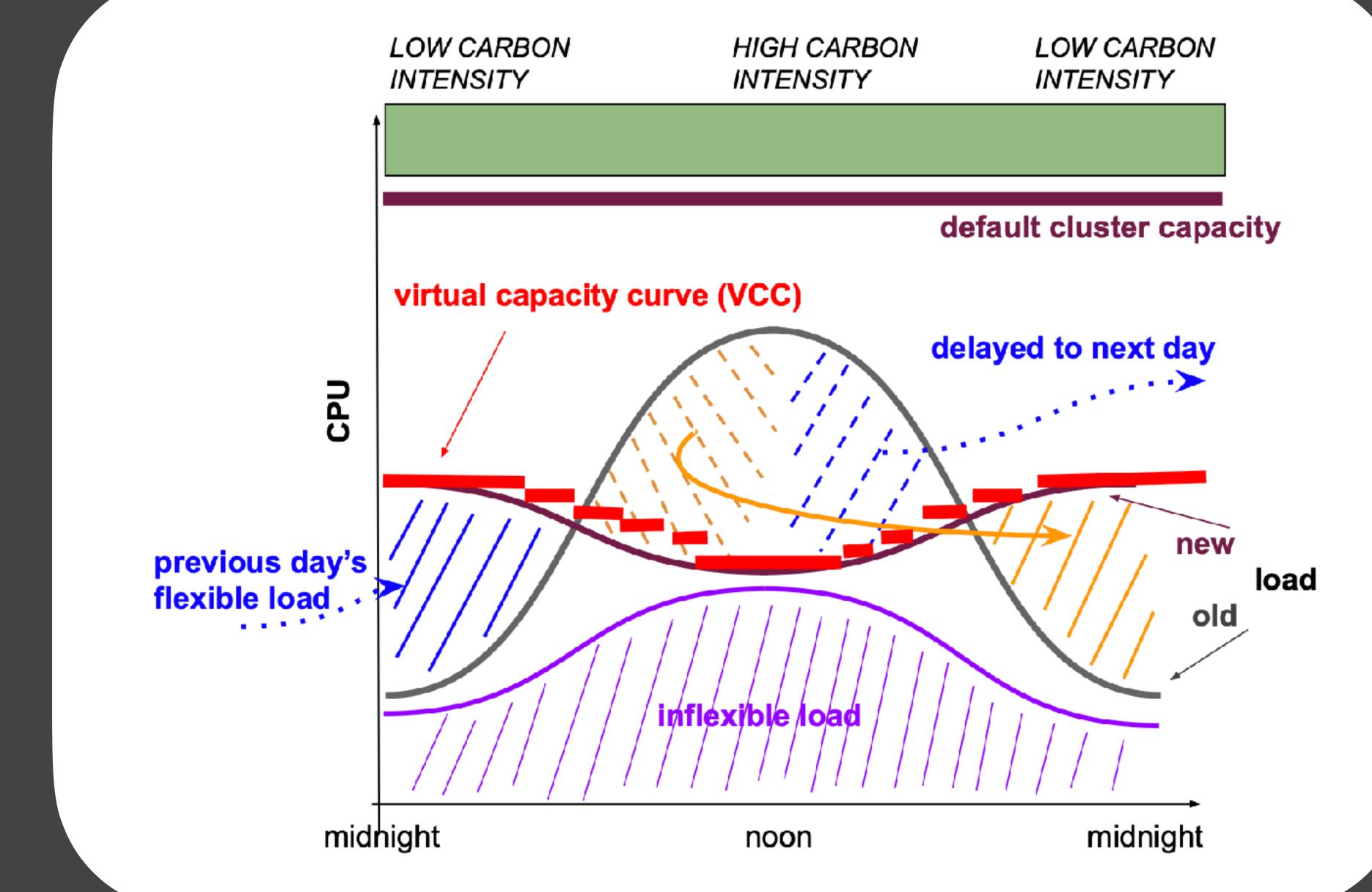


- Jobs are divided between **flexible** and **inflexible**. lacksquare
 - compute (CPU) demand is preserved

- - They forecast VCC for the next day

• Flexible load is considered shapeable/shiftable as long as its total daily

 The system needs to consider that, while running a job, the virtual capacity curve (VCC) might drop. Hence, this job should not start in the first place.





Virtual Cluster Capacity (VCC)

- some amount of daily computation!
- Considers Carbon intensity ullet
 - Using data from <u>electricityMap.org</u>
- It does not use carbon intensity directly. Carbon intensity is converted to a cost (kgCO2 -> \$\$)
 - cost saved by preventing peak load
 - temperature).
 - By using money cost instead of carbon cost, they have more data available.

Aims at reducing the peak load at carbon intensive hours but in total it should allow for the

• This way, they can factor in other metrics that can also be converted to money. E.g., the

• Peak workload entails extra cost at the infrastructure level (e.g., control facilities'

Virtual Cluster Capacity (VCC)

- If the forecast of VCC fails it might lead to shifting flexible workload more than expected.
- This happens because the amount of workload forecasted was below the workload needed and VCC was "aggressively" low.
 - The systems falls back to the real cluster capacity for **1** week until results start being realistic again.

ן

.

Next steps

- They will consider spatial-flexibility
 - I.e., tasks that can be shifted over time and over space.
 - It needs to factor in relocation overheads, though

er time and over space.

Critical questions

- Who defines what is **flexible** and **inflexible**?
- How do you estimate the CPU load of a given task?