Desenvolvimento, Validação e Manutenção de Software (mais honestamente: Sw Eng)

João Saraiva*

* *Depart. of Informática* & HASLab/INESC TEC, Univ. of Minho

greenlab.di.uminho.pt

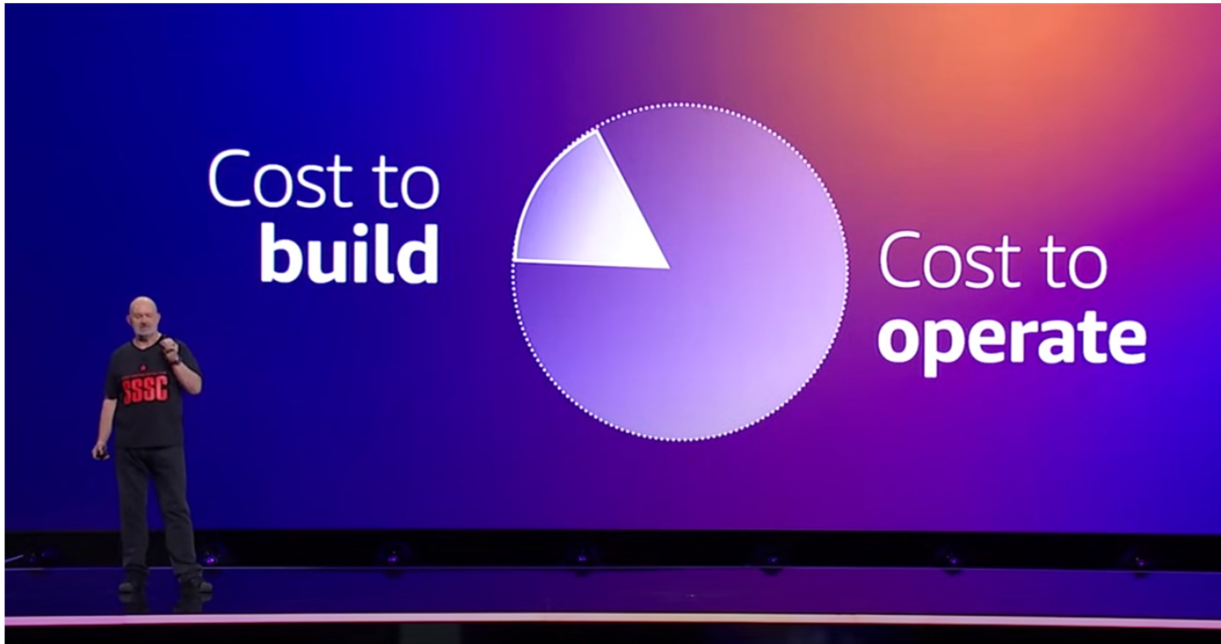
Desenvolvimento, Validação e Manutenção de Software

- Specialization in the field of **Software Engineering**

- Goals:
  - to study foundations, techniques and tools to **analyse**, to **reason**, to **measure**, to **transform**, and to **improve** (comprehensibility, runtime, energy consumption) large scale software systems.

Werner Vogels (**VP and CTO at Amazon.com**) keynote on Nov 30, 2023:

Cost to build versus Cost to operate



Desenvolvimento, Validação e Manutenção de Software

- Specialization in the field of Software Engineering

- Goals:
  - to study foundations, techniques and tools to **analyse**, to **reason**, to **measure**, to **transform**, and to **improve** (comprehensability, runtime, energy consumption,etc) large scale software systems.

- Lecturing Team:

Prof. João Saraiva

Prof. Paulo Azevedo

PhD José Nuno Macedo

Desenvolvimento, Validação e Manutenção de Software

- **Confirmed Lecturing Guests:**

Prof. Rui Abreu (META & FEUP, Univ. Porto)

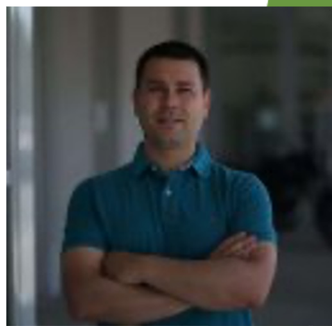
Dr. Carlos Silva (OutSystems)

Prof. Luís Cruz (Univ. Delft)

Prof. Maja Kirkeby (Roskilde Univ.)

Prof. João Fernandes (NYU at Abu Dhabi)

Prof. Jácome Cunha (FEUP, Univ. Porto)



Desenvolvimento, Validação e Manutenção de Software

- Courses:
  - Software Maintenance and Evolution

    24/25: Focus on **Software Analysis, Transformation and Testing**
  - Topics in Software Development

    24/25: Focus on **Green Software Engineering**
  - Experimentation in Software Engineering

    24/25: Focus on **Data Science for Sw Engineers/ Student Project**

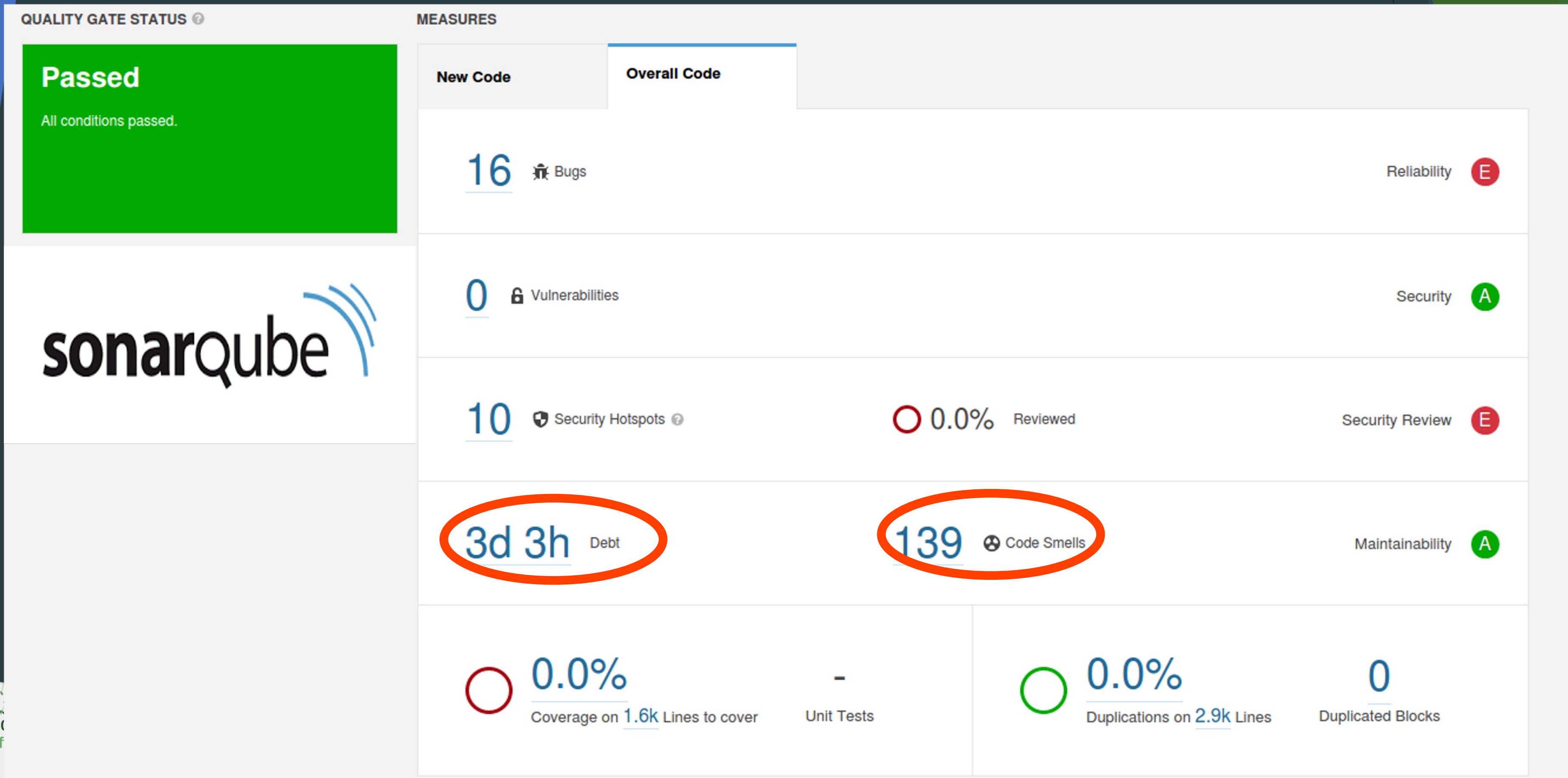
Desenvolvimento, Validação e Manutenção de Software

- **Sw Maintenance and Evolution - Sw Analysis, Transformatio and Testing**

  - Parser Combinators
    (grammars as programs)
  - Strategic Programming
    (large scale language transformations)
  - Smells, Refactorings and Technical Debt (SonarQube)
  - Software Testing
    - Automate Unit Test Case Generation (EvoSuite)
      - Test Gamification: Code Defenders (student's tournament)
      - Property Based Testing (QuickCheck, Hipothesis)
  - Fault Localization
  - Automated Program Repair (CodeBERT, CodeGPT, GPT4)

Green Software Lab

Code Smells in a Real Program



Desenvolvimento, Validação e Manutenção de Software

► **Topics in Sw Development - Mobile Software Development**

- ► Software Development in the Android Ecosystem

  Tutorial: Simão Cunha
- ► Cross Platform Software Development

  (react-native, vue-native, pwa-progressive)

  Tutorial: Dr. Carlos Silva (OutSystems)
- ► Low Code

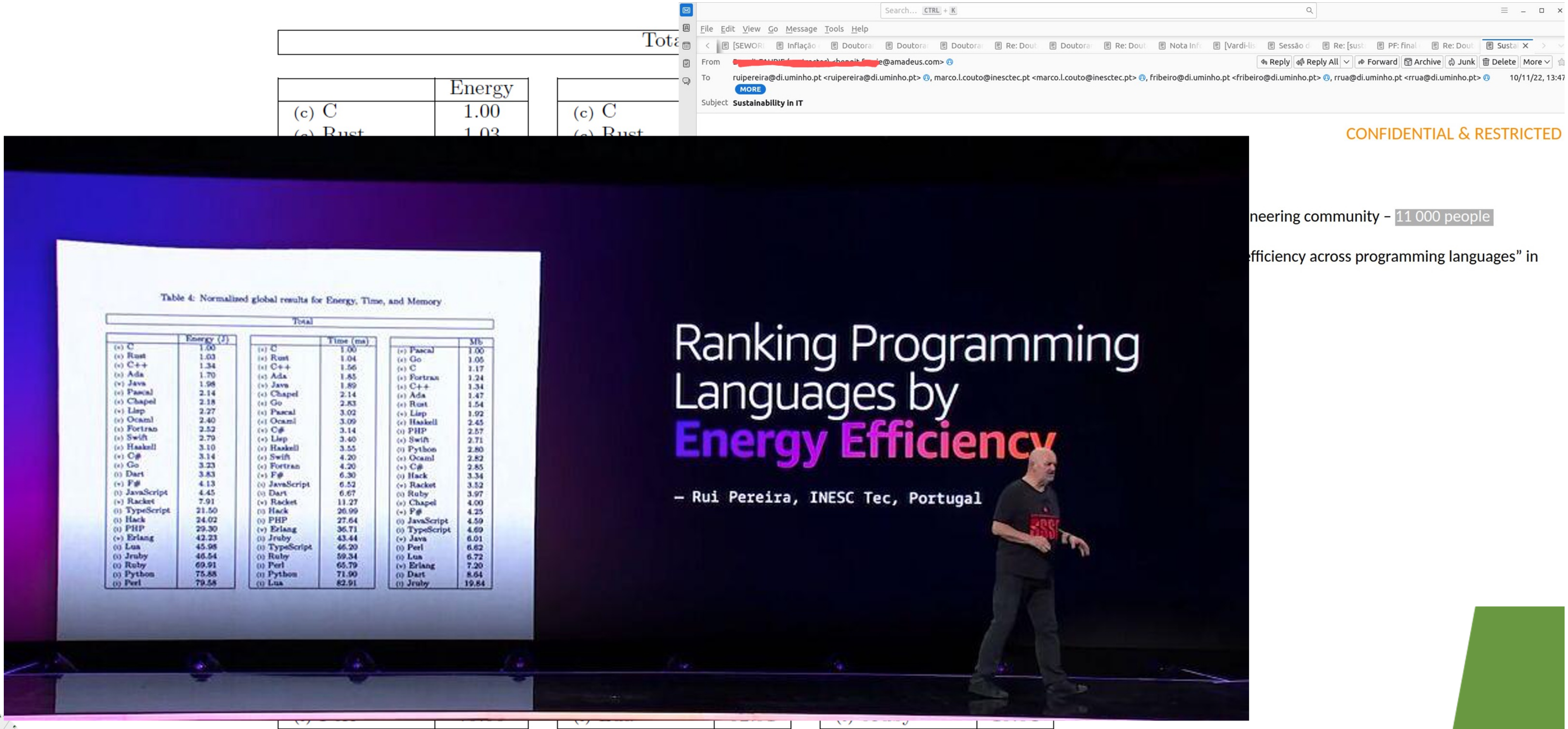
  Tutorial: Prof. Jácome Cunha, FEUP
- ► Testing of Interactive Applications

  Web Testing (selenium)

  Testing in Android (robotium, monkey, rerun)
- ► **Green Computing**

  pyANADROID, E-MANAFA

Efficiency of Programming Languages



Werner Vogels (VP and CTO at Amazon.com) keynote (Nov 30, 2023): "It shocked the development world!"
[Keynote Video]

Green Software Lab

Desenvolvimento, Validação e Manutenção de Software

- **Experimentation in Sw Engineering - Data Science for Sw Engineers**
  - Análise Exploratório de Dados
  - Análise Estatística
  - Construção de Modelos de Previsão para Sw. Eng.
  - Desenvolvimento de Projeto em grupo

Green Software Lab

Energy vs. Time vs. Memory (Pareto Optimization)

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		

Desenvolvimento, Validação e Manutenção de Software

- **Course Language: Portuguese (English if necessary)**

- **Teaching Material: English**
- **Evaluation: Portuguese (and English if necessary)**

- **Avaliação em cada disciplina: 40% Teste individual**
  40% Projeto prático
  20% Avaliação Continua

  - Most motivated student(s)
    Grant(s) to attend an international MSc/PhD Summer School

Desenvolvimento, Validação e Manutenção de Software

- ►Most motivated student(s)
  (3 grants to attend **SusTrainable 2022)**
  4-8 July 2022. Rijeka, Croacia

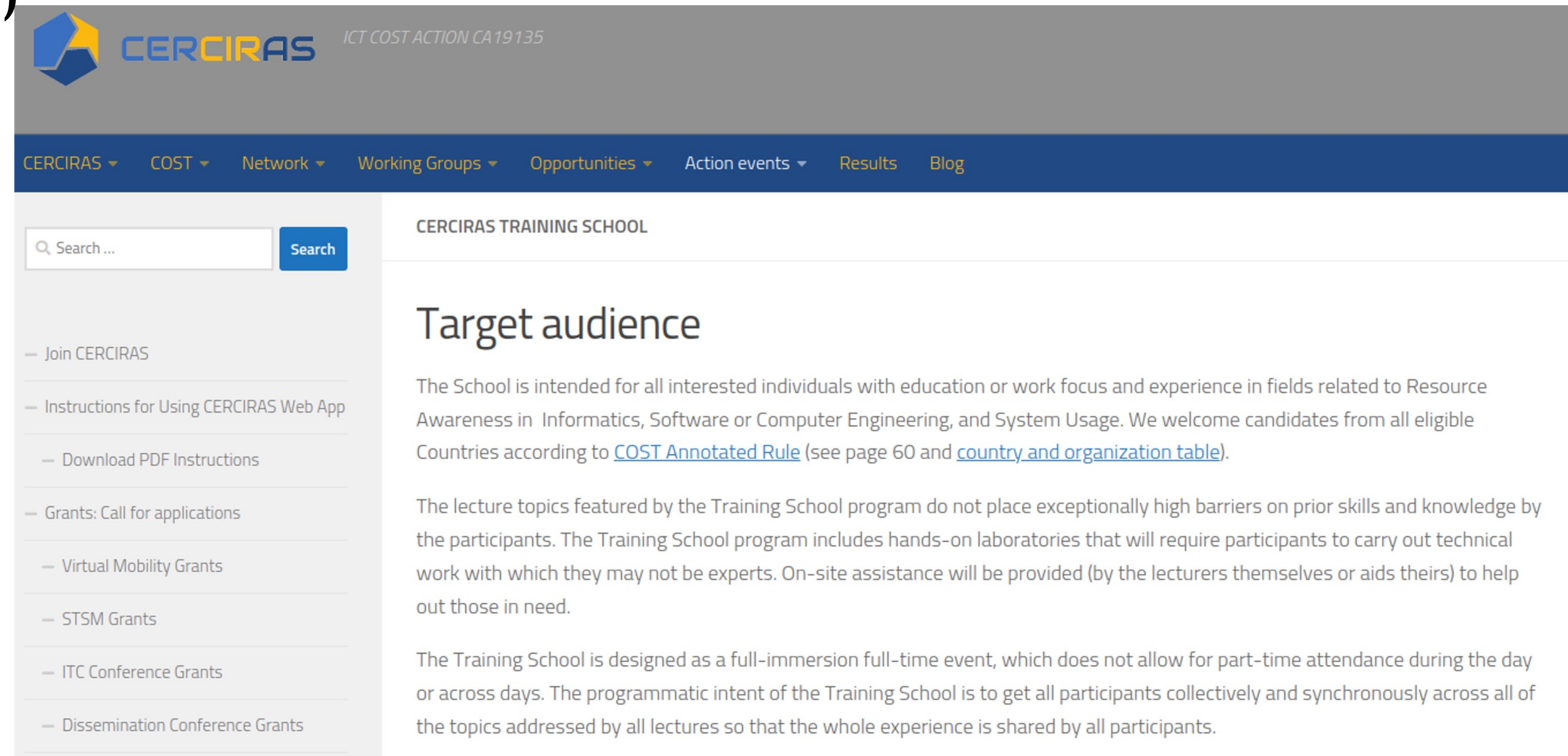
Desenvolvimento, Validação e Manutenção de Software

- Most motivated student(s)
  (**5** grants to attend **SusTrainable 2023)**
  10-14 July. Coimbra, Portugal

Desenvolvimento, Validação e Manutenção de Software

▶ Most motivated student(s)

(2 grants: **CERCIRAS Training School, Austria. 1st week, September 2024)**


CERCIRAS

ICT COST ACTION CA19135

CERCIRAS ▾ | COST ▾ | Network ▾ | Working Groups ▾ | Opportunities ▾ | Action events ▾ | Results | Blog

Search ... | Search

- Join CERCIRAS
- Instructions for Using CERCIRAS Web App
  - Download PDF Instructions
- Grants: Call for applications
  - Virtual Mobility Grants
  - STSM Grants
  - ITC Conference Grants
  - Dissemination Conference Grants

CERCIRAS TRAINING SCHOOL

Target audience

The School is intended for all interested individuals with education or work focus and experience in fields related to Resource Awareness in Informatics, Software or Computer Engineering, and System Usage. We welcome candidates from all eligible Countries according to COST Annotated Rule (see page 60 and country and organization table).

The lecture topics featured by the Training School program do not place exceptionally high barriers on prior skills and knowledge by the participants. The Training School program includes hands-on laboratories that will require participants to carry out technical work with which they may not be experts. On-site assistance will be provided (by the lecturers themselves or aids theirs) to help out those in need.

The Training School is designed as a full-immersion full-time event, which does not allow for part-time attendance during the day or across days. The programmatic intent of the Training School is to get all participants collectively and synchronously across all of the topics addressed by all lectures so that the whole experience is shared by all participants.

Green Software Lab

Students' Research Results

SLE@SPLASH, Pasadena, USA, October 2024

Students presenting at th 3rd workshop on Resource AWareness of Systems and Society (RAW 2024)

Maribor, Slovenia, July 24

ACM DIGITAL LIBRARY | Association for Computing Machinery | Universidade do Minho | Browse

Journals | Magazines | Proceedings | Books | SIGs | Conferences | People | Search ACM Digital Library

Conference | Proceedings | Upcoming Events | Authors | Affiliations | Award Winners

Home > Conferences > SPLASH > Proceedings > SLE '24 > Trading Runtime for Energy Efficiency: Leveraging Power Caps to Save Energy across Programming Languages

research-article | OPEN ACCESS |

Trading Runtime for Energy Efficiency: Leveraging Power Caps to Save Energy across Programming Languages

Authors Simão Cunha, Luís Silva, João Saraiva, João Paulo Fernandes Authors Info & Claims

SLE 2024: Proceedings of the 17th ACM SIGPLAN International Conference on Software Language Engineering • Pages 130 - 142
https://doi.org/10.1145/3687997.3695638

Published 17 October 2024 Publication History · Check for updates

Related Artifact: Energy-Languages-PowerCap • October 2024 • software • https://doi.org/10.6084/m9.figshare.27087901.v1

SLE 2024: Proceedings of the 17th ACM SIGPL...
Trading Runtime for Energy Efficiency:...
Pages 130 – 142
← Previous · Next →

Abstract
References
Index Terms
Recommendations
Comments

Abstract

Energy efficiency of software is crucial in minimizing environmental impact and reducing operational costs of ICT systems. Energy efficiency is therefore a key area of contemporary software language engineering research. A recurrent discussion that excites our community is whether runtime performance is always a proxy for energy efficiency. While a generalized intuition seems to suggest this is the case, this intuition does not align with the fact that energy is the accumulation of power over time; hence, time is only one of the factors in this accumulation. We focus on the other factor, power, and the impact that capping it has on the energy efficiency of running software. We conduct an extensive investigation comparing regular and power-capped executions of 9 benchmark programs obtained from The Computer Language Benchmarks Game, across 20 distinct programming languages. Our results show that employing power caps can be used to trade running time, which is degraded, for energy efficiency, which is improved, in all the programming languages and in all benchmarks that were considered. We observe overall energy savings of almost 14% across the 20 programming languages, with notable savings of 27% in Haskell. This saving, however, comes at the cost of an overall increase of the program's execution time of 91% in average. We are also able to draw similar observations using language specific benchmarks for programming languages of different paradigms and with different execution models. This is achieved analyzing a wide range of benchmark programs from the nofib Benchmark Suite of Haskell Programs, DaCapo Benchmark Suite for Java, and the Python Performance Benchmark Suite. We observe energy savings of

Green Software Lab