INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Green Computing - CM 5

Techniques de conception pour des logiciels plus économes

Slides réalisés avec l'aide Romain Lefeuvre, doctorant @DiverSE

Quentin Perez
quentin.perez@insa-rennes.fr
4INFO
2023-2024

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Le contexte d'exécution / mesure

Les travaux présentés dans ces slides et les leviers techniques permettant d'économiser de l'énergie au niveau logiciel sont toujours à considérer dans leurs contextes !

Une étude sur un objet (programme, un service, etc.) dans un contexte A pourra avoir des résultats différents dans un contexte B.

La **consommation d'énergie** est **fortement impactée par le contexte et l'environnement matériel et logiciel d'exécution** !

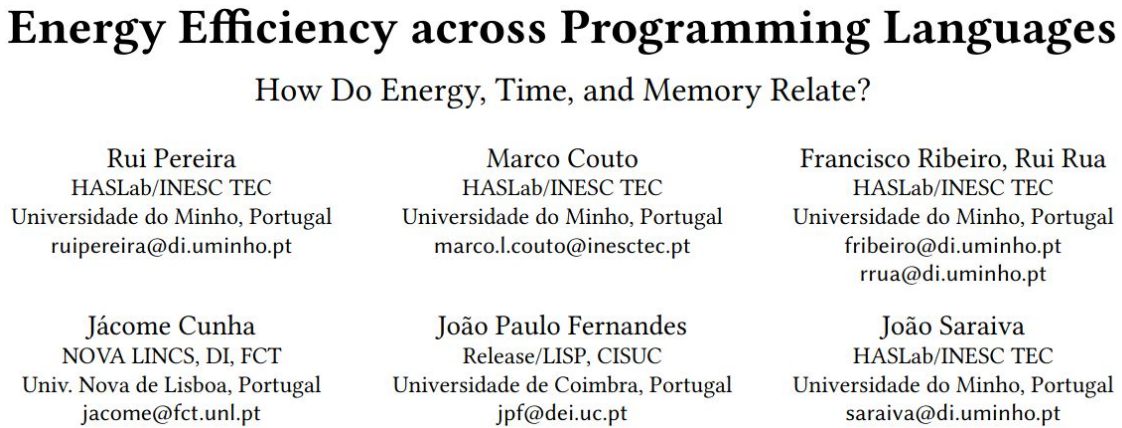
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Contexte d’exécution / de mesure et impact sur les résultats de consommation

Illustration de l'influence du contexte : le langage de programmation et la consommation

Pour illustrer le caractère contextuel, nous allons étudier une publication "connue" qui fait l'objet de nombreux débats...

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



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

[Pereira17]

Illustration de l'influence du contexte : le langage de programmation et la consommation - Questions de recherche

RQ1 : "Can we compare the energy efficiency of software languages?"

RQ2 : "Is the faster language always the most energy efficient?"

"~~How does memory usage relate to energy consumption?~~"

RQ3 : "Can we automatically decide what is the best programming language considering energy, time, and memory usage?"

Question subsidiaire : "Le comparatif des langages présenté ici est-il représentatif d'une utilisation "classique industrielle" ?

[Pereira17]

Illustration de l'influence du contexte : le langage de programmation et la consommation - Questions de recherche

Comparer des langages bien que semblant facile (#yakafocon) n'est pas quelque chose de simple :

- **Variabilité ! :**
  - **Langages compilés VS interprétés**
  - **Utilisation de machine virtuelle**
  - **Optimisations du compilateur pour la création de l'exécutable**
  - **Présence ou non d'un garbage collector**
  - **Utilisation de librairies tierces**
  - **…**

[Pereira17]

Illustration de l'influence du contexte : le langage de programmation et la consommation - Méthodologie

Pereira *et al.* ont utilisé un benchmark "connu" permettant de comparer les performances en termes de rapidité des langages :

The Computer Langage Benchmark Game

https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

⇒ **Positif pour faire des comparaisons temps VS énergie**

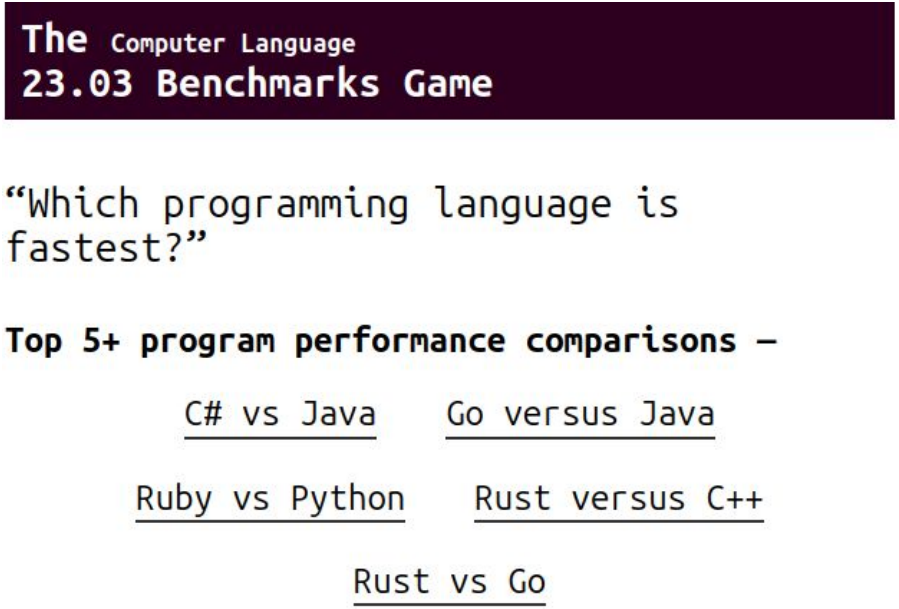


Illustration de l'influence du contexte : le langage de programmation et la consommation - Méthodologie

Pereira *et al.* ont utilisé un benchmark "connu" permettant de comparer les performances en termes de rapidité des langages :

The Computer Langage Benchmark Game

https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

⇒ Positif pour faire des comparaisons temps VS énergie

RQ1 ⇒ oui, on peut comparer l'efficacité énergétique des langages, MAIS... car il y a un mais

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

Illustration de l'influence du contexte : le langage de programmation et la consommation - Méthodologie

The Computer Langage Benchmark Game

⇒ Positif pour faire des comparaisons temps VS **énergie**, **MAIS... car il y a un mais...**

Cale pose la question de la représentativité du contexte "réel" d'utilisation de ces langages ⇒ Génère-t-on souvent des ensembles de Mandelbrot...?

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

Illustration de l'influence du contexte : le langage de programmation et la consommation - Méthodologie

27 langages comparés avec différents paradigmes :

- fonctionnels
- impératifs
- orientés objets
- de scripts

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;

Illustration de l'influence du contexte : le langage de programmation et la consommation - Méthodologie

Mesure de consommation :

- RAPL en utilisant une bibliothèque C
- Chaque programme est exécuté et mesuré 10x

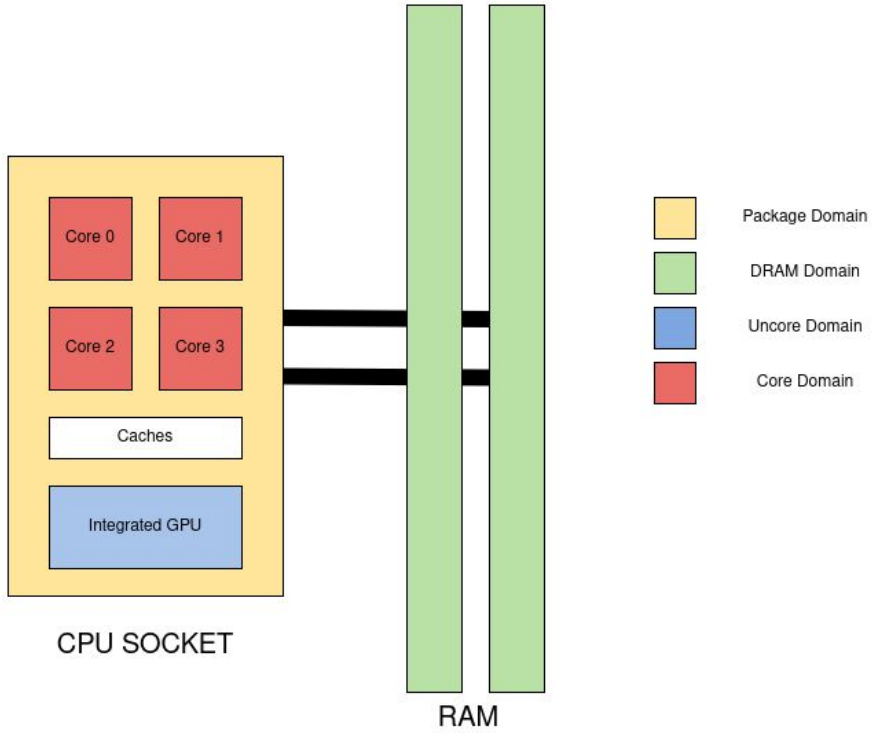


Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

C est selon le benchmark le langage le plus économe et rapide parmi les langages compilés dans le cas de **binary-trees**

	binary-trees			
	Energy	Time	Ratio	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 $\Uparrow_1$	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 $\uparrow_1 \Downarrow_{16}$	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 $\uparrow_2 \Downarrow_9$	156.71	4497	0.035	502
(v) C# $\downarrow_1 \Downarrow_1$	189.74	10797	0.018	427
(v) F# $\downarrow_3 \Downarrow_1$	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 $\uparrow_5 \Uparrow_1$	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 $\downarrow_2 \Downarrow_4$	312.14	21349	0.015	916
(i) TypeScript $\downarrow_2 \Downarrow_2$	315.10	21686	0.015	915
(c) Go $\uparrow_3 \Uparrow_{13}$	636.71	16292	0.039	228
(i) Jruby $\uparrow_2 \Downarrow_3$	720.53	19276	0.037	1671
(i) Ruby $\Uparrow_5$	855.12	26634	0.032	482
(i) PHP $\Uparrow_3$	1,397.51	42316	0.033	786
(i) Python $\Uparrow_{15}$	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.			

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

binary-trees	Energy	Time	Ratio	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust ⇓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 ↓1 ⇑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 ↓4 ⇓6	155.81	11261	0.014	467
(i) Hack ↑2 ⇓9	156.71	4497	0.035	502
(v) C# ↓1 ⇓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 ↑5 ⇑4	237.29	7265	0.033	335
(v) Erlang ↑5 ⇑1	266.14	7327	0.036	433
(c) Haskell ↑2 ⇓2	270.15	11582	0.023	494
(i) Dart ↓1 ⇑1	290.27	17197	0.017	475
(i) JavaScript ↓2 ⇓4	312.14	21349	0.015	916
(i) TypeScript ↓2 ⇓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 ↓1	2,452.04	209217	0.012	1961
(i) Perl ↑1	3,542.20	96097	0.037	2148
(c) Swift	n.e.			

regex-redux	Energy	Time	Ratio	Mb
(i) TypeScript ↓2 ⇓12	24.65	2009	0.012	587
(c) C ↑1	24.83	805	0.031	151
(i) JavaScript ↓2 ⇓9	25.68	2096	0.012	525
(i) PHP ↑2 ⇓1	34.57	1667	0.021	182
(c) Pascal ↓1 ⇑4	35.20	2282	0.015	106
(i) Hack ↑2 ⇓2	38.96	2052	0.019	268
(c) Rust	40.26	2287	0.018	218
(c) Chapel ⇓11	97.19	4534	0.021	1055
(c) Ada ⇑5	148.66	5157	0.029	157
(i) Python ↓1	161.62	7116	0.023	429
(c) Ocaml ↓3 ⇓6	172.43	12978	0.013	948
(c) C++ ↓1 ⇑6	176.24	10656	0.017	216
(i) Ruby ↓4 ⇑4	192.88	14282	0.014	305
(v) Java ↑4 ⇓6	194.65	5694	0.034	1225
(i) Dart ↓1 ⇑4	197.92	13485	0.015	459
(i) Perl ↑4 ⇑13	236.24	7164	0.033	154
(i) Jruby ↑2 ⇓4	348.44	13477	0.026	1369
(v) Racket ↓1	358.20	26152	0.014	983
(v) C# ↑1 ⇑3	522.59	14723	0.035	851
(c) Swift ⇑5	538.11	41703	0.013	677
(v) F# ⇑7	650.51	46905	0.014	667
(c) Haskell	n.a.			
(c) Fortran	n.a.			
(c) Go	n.e.			
(i) Lua	n.e.			
(v) Erlang	n.a.			
(v) Lisp	n.e.			

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

binary-trees	Energy	Time	Ratio	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust ⇓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 ↓1 ⇑2	100.74	3525	0.029	148
(v) Java ↑1 ⇓16	111.84	3306	0.034	1120
(v) Lisp ↓3 ⇓3	149.5			
(v) Racket ↓4 ⇓6	155.8			
(i) Hack ↑2 ⇓9	156.7			
(v) C# ↓1 ⇓1	189.7			
(v) F# ↓3 ⇓1	207.1			
(c) Pascal ↓3 ⇑5	214.6			
(c) Chapel ↑5 ⇑4	237.2			
(v) Erlang ↑5 ⇑1	266.1			
(c) Haskell ↑2 ⇓2	270.1			
(i) Dart ↓1 ⇑1	290.2			
(i) JavaScript ↓2 ⇓4	312.1			
(i) TypeScript ↓2 ⇓2	315.16	21886	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 ↓1	2,452.04	209217	0.012	1961
(i) Perl ↑1	3,542.20	96097	0.037	2148
(c) Swift	n.e.			

regex-redux	Energy	Time	Ratio	Mb
(i) TypeScript ↓2 ⇓12	24.65	2009	0.012	587
(c) C ↑1	24.83	805	0.031	151
(i) JavaScript ↓2 ⇓9	25.68	2096	0.012	525
(i) PHP ↑2 ⇓1	34.57	1667	0.021	182
(c) Pascal ↓1 ⇑4	35.20	2282	0.015	106
(i) Hack ↑2 ⇓2	38.96	2052	0.019	268
(c) Rust	40.26	2287	0.018	218
	97.19	4534	0.021	1055
	48.66	5157	0.029	157
	61.62	7116	0.023	429
	72.43	12978	0.013	948
	76.24	10656	0.017	216
	92.88	14282	0.014	305
	94.65	5694	0.034	1225
	97.92	13485	0.015	459
	36.24	7164	0.033	154
	48.44	13477	0.026	1369
	58.20	26152	0.014	983
(v) C# ↑1 ⇑3	522.59	14723	0.035	851
(c) Swift ⇑5	538.11	41703	0.013	677
(v) F# ⇑7	650.51	46905	0.014	667
(c) Haskell	n.a.			
(c) Fortran	n.a.			
(c) Go	n.e.			
(i) Lua	n.e.			
(v) Erlang	n.a.			
(v) Lisp	n.e.			

Deux benchmarks (contextes) deux résultats très différents ! ⇒ influence des tâches effectuées et du contexte

Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

Le tableau ici est la normalisation sur la moyenne des benchmarks évoqués plus tôt. Il n'est ainsi pas représentatif d'un corpus de projets "génériques" ou "industriels" ⇒ **ne pas le sortir de son contexte !** (et ainsi éviter le café du commerce...)

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

Le tableau ici est la normalisation sur la moyenne des benchmarks évoqués plus tôt. Il n'est ainsi pas représentatif d'un corpus de projets "génériques" ou "industriels"

⇒ **RQ2 Ici, les langages les plus rapides ne sont pas les plus économes !**

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

Illustration de l'influence du contexte : le langage de programmation et la consommation - Résultats

Le tableau donne la vision en fonction du front de Pareto du choix des langages ⇒ **si un tableau devait être retenu, ce devrait être celui-ci !**

⇒ RQ3 Oui, on peut décider de manière automatique du langage à utiliser pour 1 critère mais pas pour plusieurs (temps, énergie, etc.)

Table 5. Pareto optimal sets for different combination of objectives.

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		

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Influence du type de boucle en Python

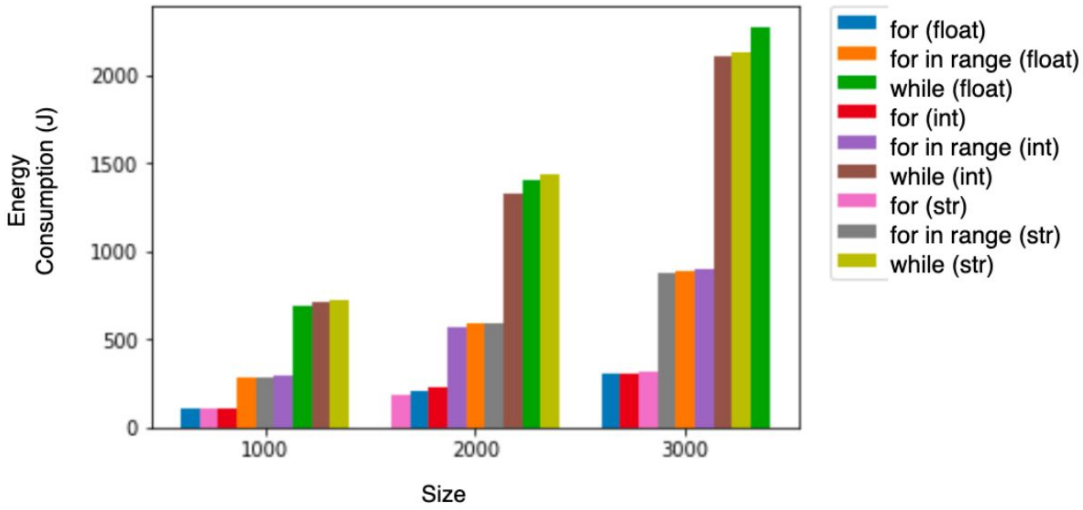
Comparaison de trois types de boucles :

```
for (i in range(len(n)))
```

```
for (element in collection)
```

```
while
```

⇒ Il faut privilégier les façons “Pythonic” de programmation à savoir éviter les while !



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Influence du type de boucle en Python

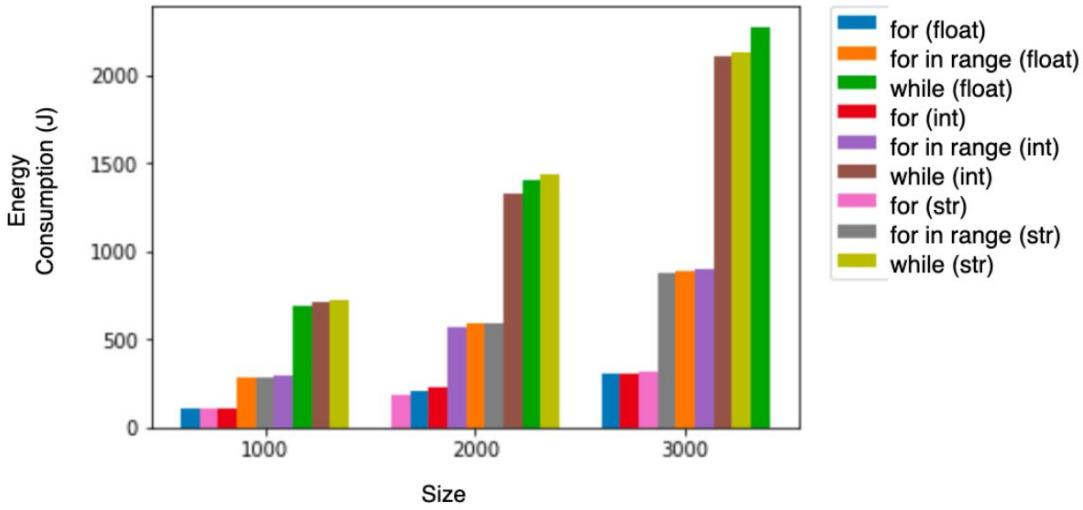
Comparaison de trois types de boucles :

```
for (i in range(len(n)))
```

```
for (element in collection)
```

```
while
```

Le while est toujours le plus consommateur ⇒ coût dû à l'incrément dans la boucle



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Influence du type de boucle en Python

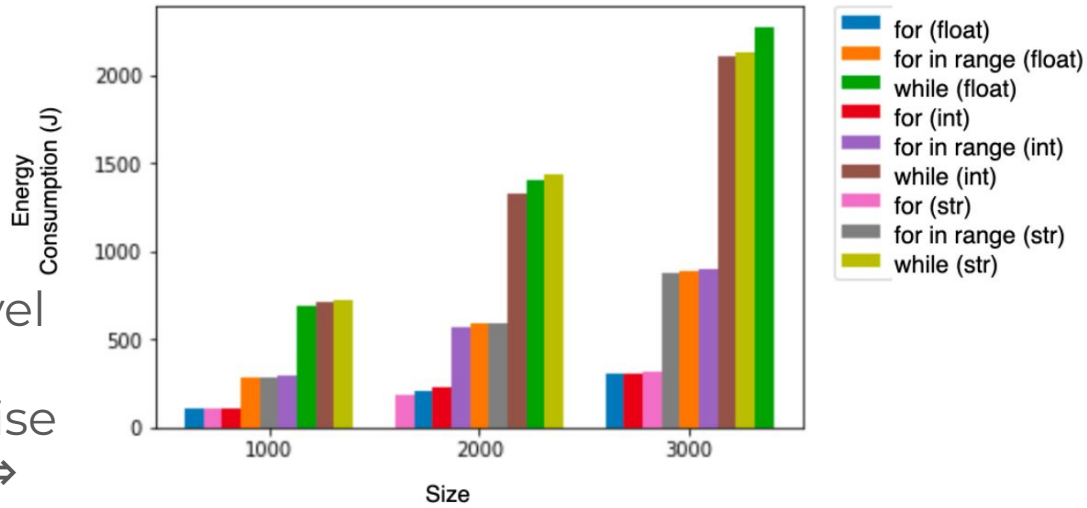
Comparaison de trois types de boucles :

```
for (i in range(len(n)))
```

```
for (element in collection)
```

```
while
```

Le “for in range” créé un nouvel itérateur à l’exécution pour chaque tour de boucle et utilise la fonction C range ($O(n)$=1)) ⇒ cout !



Influence du type de boucle en Python

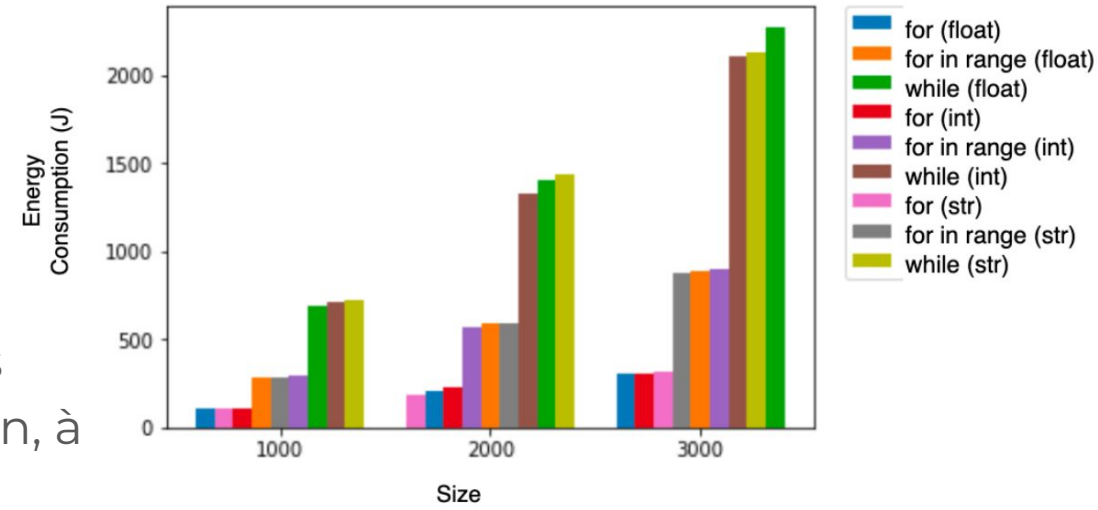
Comparaison de trois types de boucles :

```
for (i in range(len(n)))
```

```
for (element in collection)
```

```
while
```

⇒ Il faut privilégier les façons “Pythonic” de programmation, à savoir éviter les while !



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

Green Computing | INFO4 | Quentin Perez | 2023-2024

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

Comparaison de la consommation CPU de différentes bibliothèques pour les entrées/sorties sur les fichiers :

- lectures/écritures
- fichiers de différentes tailles
- différents types de lecture/écriture (ex : lecture bufferisé ou non...)

Objectif :

Identifier la méthode la plus efficace pour l'ouverture de fichiers

Cas appliqué à un "contexte réel" !

Evaluating The Energy Consumption of Java I/O APIs

Zakaria OURNANI
Orange Labs/ Inria / Univ. Lille
France
zakaria.ournani@inria.fr

Romain ROUVOY
Univ. Lille / Inria / IUF
France
romain.rouvoy@univ-lille.fr

Pierre RUST
Orange Labs
France
pierre.rust@orange.com

Joel PENHOAT
Orange Labs
France
joel.penhoat@orange.com

[Ournani21]

TABLE I: The list of file sizes.

Method	File Size	Lines count	Line length
Tiny	100 KB	1,000	100
Small	15 MB	100,000	150
Medium	200 MB	1,000,000	200
Medium-large	3.2 GB	8,000,000	400
Large	16 GB	40,000,000	400

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

Comparaison de la consommation CPU de différentes bibliothèques pour les entrées/sorties sur les fichiers :

- lectures/écritures
- fichiers de différentes tailles
- différents types de lecture/écriture (ex : lecture bufferisé ou non...)

Objectif :

Identifier la méthode la plus efficace pour l'ouverture de fichiers

Cas appliqué à un "contexte réel" !

Class	Acronym	Method	Purpose	Availability
java.io.InputStreamReader	IOSTREAM	read(Byte[])	Read	JDK 1.0
java.io.OutputStreamWriter	IOSTREAM	write(String)	Write	JDK 1.0
java.io.FileInputStream	IOSTREAM	Skip(long) readAllBytes()	Seek ReadAll	JDK 1.0
java.io.BufferedInputStream	BIOSTREAM	readLine()	Read	JDK 1.0
java.io.BufferedOutpuStream	BIOSTREAM	write(String)	Write	JDK 1.0
java.io.FileReader	FILEREADER	read(char[])	Read	JDK 1.1
java.io.FileWriter	FILEWRITER	write(String,int,int)	Write	JDK 1.1
java.io.BufferedReader	BFILEREADER	readLine()	Read	JDK 1.1
java.io.BufferedWriter	BFILEWRITER	write(String)	Write	JDK 1.1
java.nio.channels.FileChannel	CHANNEL	read(ByteBuffer) write(ByteBuffer) position(long)	Read Write Seek	JDK 1.4
java.nio.FileChannel	OMCHANNEL	map(MapMode,long,long)	Read	JDK 1.4
java.nio.Files	NIOF	readLine() write(String) readAllLines(Path)	Read Write ReadAll	JDK 1.7
java.util.Scanner	SCANNER	nextLine()	Read	JDK 1.5
java.io.RandomAccesFile	RAF	readLine() writeBytes(String) seek(long)	Read Write Seek	JDK 1.0
apache.commons.io.FileUtils	APACHE	readFileToString(File,Charset) write(File,List¡String¿,Boolean) readFileToByteArray(File, Charset)	Read Write ReadAll	NA
google.common.io.CharSource google.common.io.CharSink google.common.io.Files	GUAVA	readLine() write(String) readLines(file, Charset)	Read Write ReadAll	NA

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

Comparaison de 9 méthodes de lecture.

Utiliser un Channel comparativement à d'autres méthodes permet d' économiser plus de 30% d'énergie comparativement à NIO.

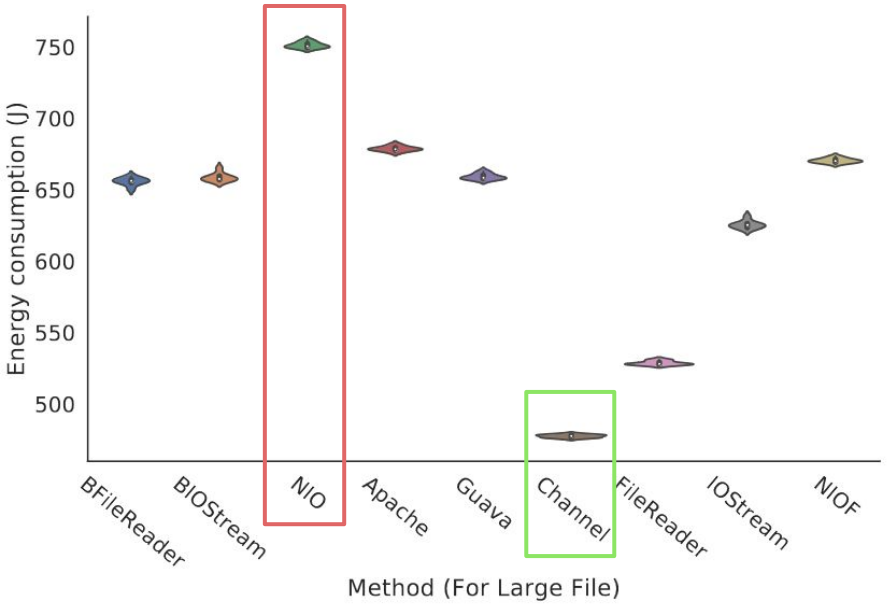


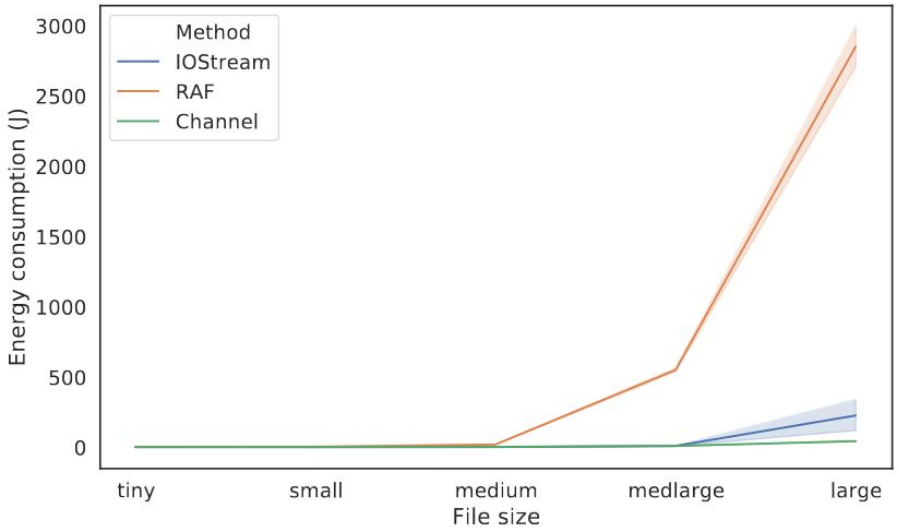
Fig. 3: Energy consumption of read methods for a large file.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

Comparaison de 9 méthodes de lecture.

Constance de la consommation des channels sur l'ensemble des tailles.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Bibliothèques I/O Java et consommation

TABLE V: Energy consumption (joules) and execution time (ms) for writing files of different sizes by chunks.

Method	Tiny		Small		Medium		Medium-Large		Large	
	Energy	Time	Energy	Time	Energy	Time	Energy	Time	Energy	Time
Apache	0.20	13.3	8.4	760	8.7	$7.8 * 10^3$	679	$6.2 * 10^4$	2526	$2.3 * 10^5$
BFileWriter	0.24	15.1	9.5	794	8.6	$7.8 * 10^3$	$6.1 * 10^4$	$11.3 * 10^3$	2501	$2.3 * 10^5$
BIOStream	0.22	13.7	8.9	770	82	$7.5 * 10^3$	662	$6.0 * 10^4$	2502	$2.3 * 10^5$
Channel	0.23	14.5	10	919	99	$9.1 * 10^3$	798	$7.3 * 10^4$	3293	$2.8 * 10^5$
FileWriter	0.15	10.5	8.4	764	83	$7.6 * 10^3$	669	$6.1 * 10^4$	2518	$2.3 * 10^5$
Guava	1.18	80.7	95	6882	962	$7.1 * 10^4$	7507	$5.5 * 10^5$	15592	$1.7 * 10^6$
IOStream	0.12	9	8.5	765	83	$7.5 * 10^3$	671	$6.1 * 10^4$	2522	$2.3 * 10^5$
NIOF	0.32	22.3	27	1714	238	$1.5 * 10^4$	1897	$1.2 * 10^5$	3684	$3.4 * 10^5$
RAF	0.14	10.6	10	920	98	$9 * 10^3$	795	$7.3 * 10^4$	2932	$2.6 * 10^5$

Écrire un fichier consomme plus que le lire :

- 500 joules pour la lecture de fichier de grande taille dans le cas le plus efficace
- **x5 pour écrire** (dans le cas le plus favorable)

Guava pour tout type de fichier consomme 6x plus d'énergie que la meilleure solution

Channel ⇒ **le plus efficace sur tout type de lecture**, est **30% plus consommateur que la meilleure solution sur des fichiers larges.**

Bibliothèques I/O Java et consommation

Contrairement à la lecture, dans le cas de l'écriture, il n'y a pas de bibliothèque qui permet d'avoir une consommation constante sur la taille.

Influence de la taille sur la consommation à partir des fichiers de tailles médiums ⇒ Contexte !

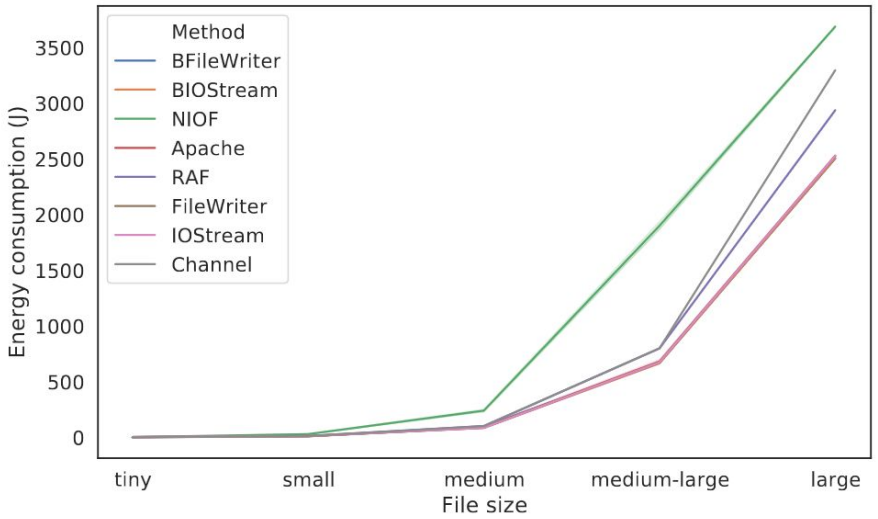


Fig. 7: Energy consumption to write files for several file sizes.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

Green Computing | INFO4 | Quentin Perez | 2023-2024

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

Comparaison de 12 distributions de JVM et évaluation sur un panel de 12 projets de natures variées pour minimiser les biais de spécificité

Evaluating the Impact of Java Virtual Machines on Energy Consumption

Zakaria Ournani
Orange Labs / Inria / Univ. Lille
zakaria.ournani@inria.fr

Mohammed Chakib Belgaid
Inria / Univ. Lille
chakib.belgaid@inria.fr

Romain Rouvoy
Univ. Lille / Inria / IUF
romain.rouvoy@univ-lille.fr

Pierre Rust
Orange Labs
pierre.rust@orange.com

Joël Penhoat
Orange Labs
joel.penhoat@orange.com

Distribution	Provider	Support	Selected versions
HotSpot	**Adopt OpenJDK**	All	8.0.275, 11.0.9, 12.0.2, 13.0.2, 14.0.2, 15.0.1
HotSpot	**Oracle**	All	8.0.265, 9.0.4, 10.0.2, 11.0.2, 12.0.2, 13.0.2, 14.0.2, 15.0.1, 16.ea.24
Zulu	**Azul Systems**	All	8.0.272, 9.0.7, 10.0.2, 11.0.9, 12.0.2, 13.0.5, 14.0.2, 15.0.1
SapMachine	**SAP**	All	11.0.9, 12.0.2, 13.0.2, 14.0.2, 15.0.1
Librca	**BellSoft**	All	8.0.275, 11.0.9, 12.0.2, 13.0.2, 14.0.2, 15.0.1
Corretto	**Amazon**	Mjr	8.0.275, 11.0.9, 15.0.1
HotSpot	**Trava OpenJDK**	LTS	8.0.232, 11.0.9
Dragonwell	**Alibaba**	LTS	8.0.272, 11.0.8
OpenJ9	**Eclipse**	All	8.0.275, 11.0.9, 12.0.2, 13.0.2, 14.0.2, 15.0.1
GraalVM	**Oracle**	LTS	19.3.4.r8, 19.3.4.r11, 20.2.0.r8, 20.2.0.r11
Mandrel	**Redhat**	LTS	20.2.0.0

[Ournani21]

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

Comparaison de 12 distributions de JVM et évaluation sur un panel de 12 projets de natures variées pour minimiser les biais de spécificité

Benchmark	Description	Focus
ALS	Factorize a matrix using the alternating least square algorithm on spark	Data-parallel, compute-bound
Avrora	Simulates and analyses for AVR microcontrollers	Fine-grained multi-threading, events queue
Dotty	Uses the dotty Scala compiler to compile a Scala codebase	Data structure, synchronization
Fj-Kmeans	Runs K-means algorithm using a fork-join framework	Concurrent data structure, task parallel
H2	Simulates an SQL database by executing a TPC-C like benchmark written by Apache	Query processing, transactions
Lusearch	Searches keywords over a corpus of data comprising the works of Shakespeare and the King James bible	Externally multi-threaded
Neo4j	Runs analytical queries and transactions on the Neo4j database	Query Processing, Transactions
Philosophers	Solves dining philosophers problem	Atomic, guarded blocks
PMD	Analyzes a list of Java classes for a range of source code problems	Internally multi-threaded
Reactors	Runs a set of message-passing workloads based on the reactors framework	Message-passing, critical-sections
Scrabble	Solves a scrabble puzzle using Java streams	Data-parallel, memory-bound
Sunflow	Renders a classic Cornell box; a simple scene comprising two teapots and two glass spheres within an illuminated box	Compute-bound

Impact du type de JVM sur la consommation d'énergie

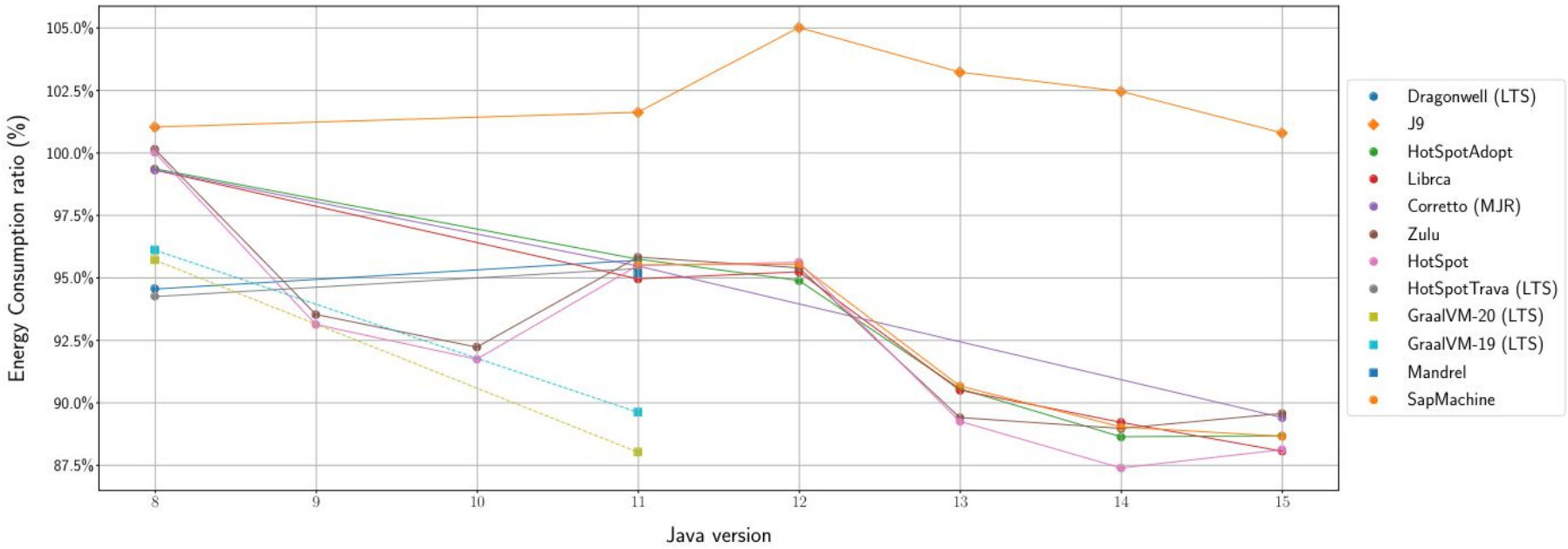


Figure 1: Energy consumption evolution of selected JVM distributions along versions.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

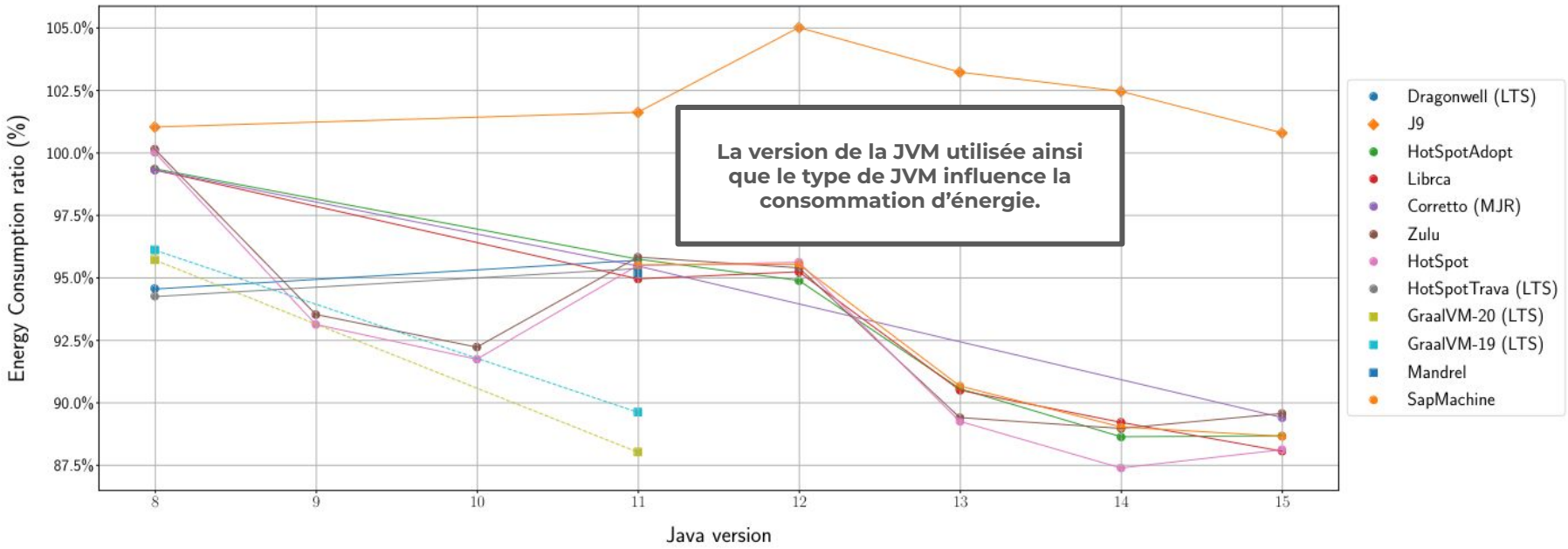


Figure 1: Energy consumption evolution of selected JVM distributions along versions.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

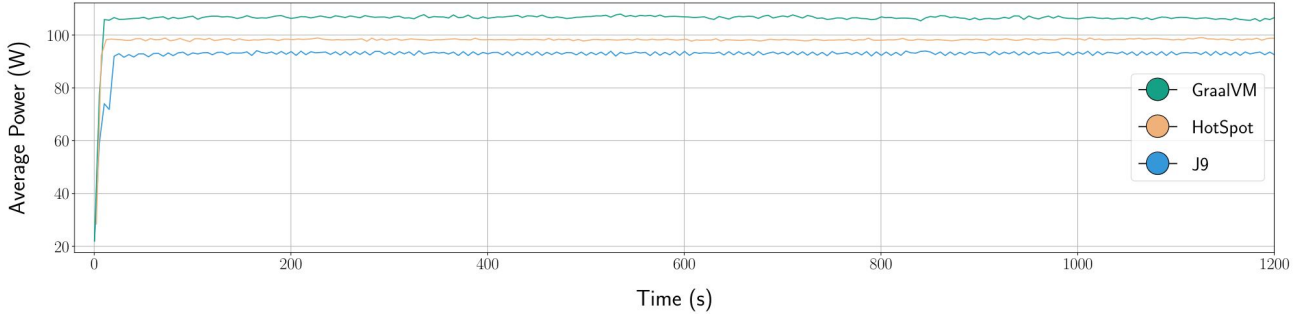
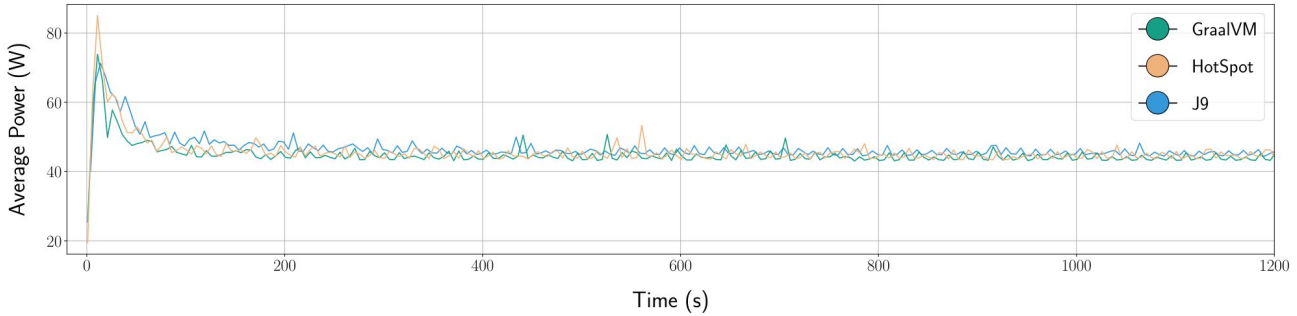


Figure 4: Power consumption of Scrabble as a service for HotSpot, GraalVM & J9.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact du type de JVM sur la consommation d'énergie

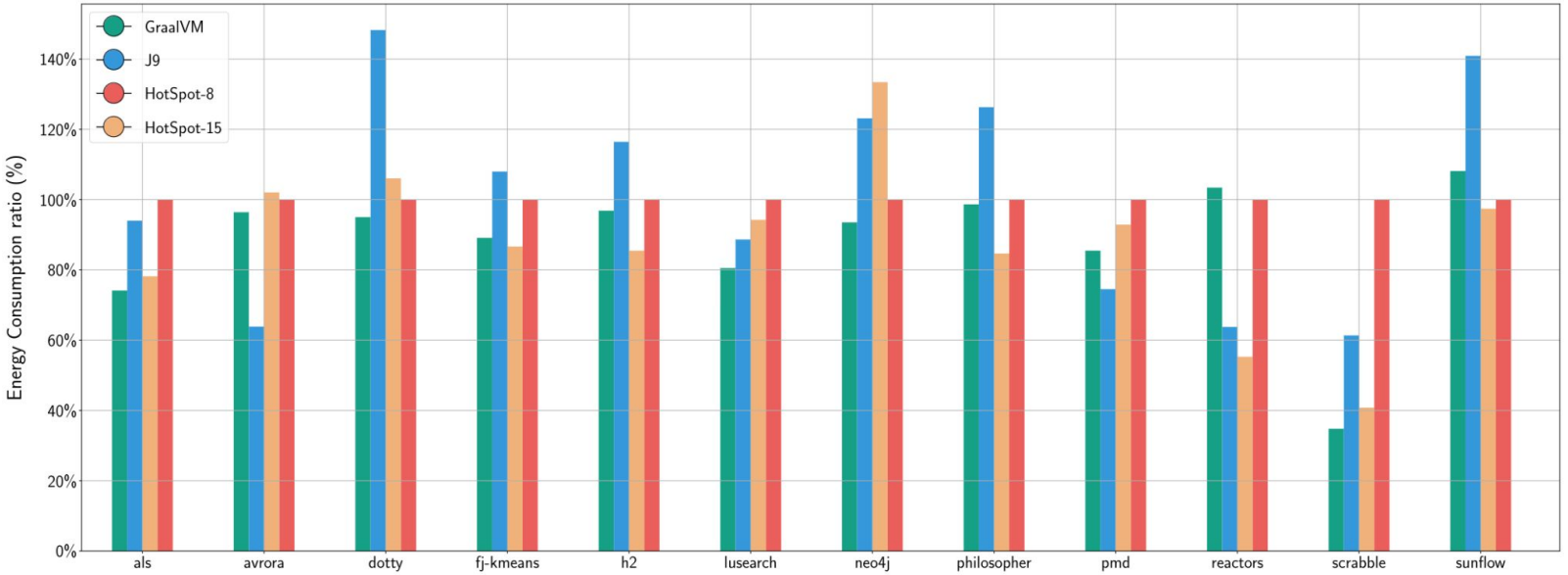


Figure 3: Energy consumption comparison across Java benchmarks for HotSpot, GraalVM & J9.

Impact du type de JVM sur la consommation d'énergie

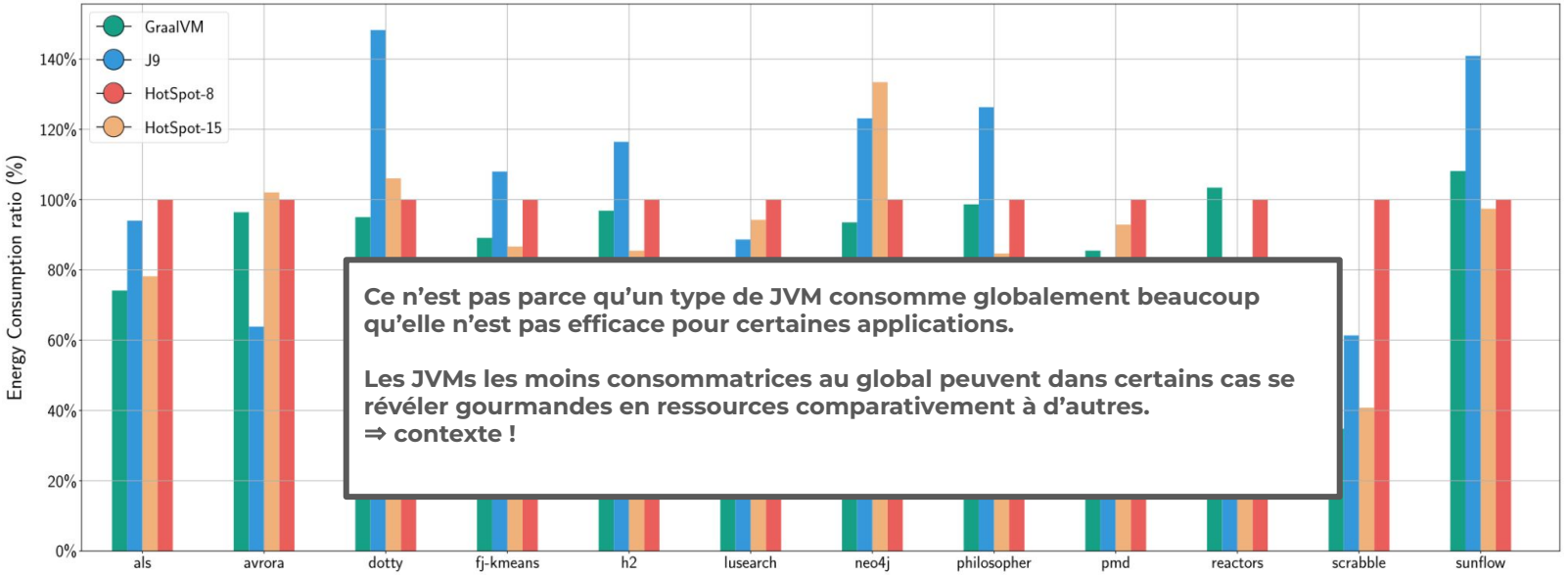


Figure 3: Energy consumption comparison across Java benchmarks for HotSpot, GraalVM & J9.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

Green Computing | INFO4 | Quentin Perez | 2023-2024

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

Intuition : utiliser Docker induit un coût non négligeable notamment lié à la conteneurisation et à la couche Docker

Objectif : étudier ce cout énergétique lié à l'utilisation de Docker

Comparaison en utilisant des applications "industrielles dockerisés"

How does docker affect energy consumption? Evaluating workloads in and out of Docker containers

Eddie Antonio Santos*, Carson McLean, Christopher Solinas, Abram Hindle

Department of Computing Science, University of Alberta, Edmonton, Canada

[Santos18]

Software	Version	Docker Image
Distribution	Ubuntu Server 16.04.1 LTS	
Kernel	Linux 4.4.0	
Docker	1.12.1	
Apache	2.4.10	php:5.6-apache
PHP	5.6.24	php:5.6-apache
MySQL	5.7.15	mysql:5.7.15
WordPress	4.6.0	wordpress:4.6-apache
Redis	3.2.3	redis:3.2.3
PostgreSQL	9.5.4	postgres:9.5.4

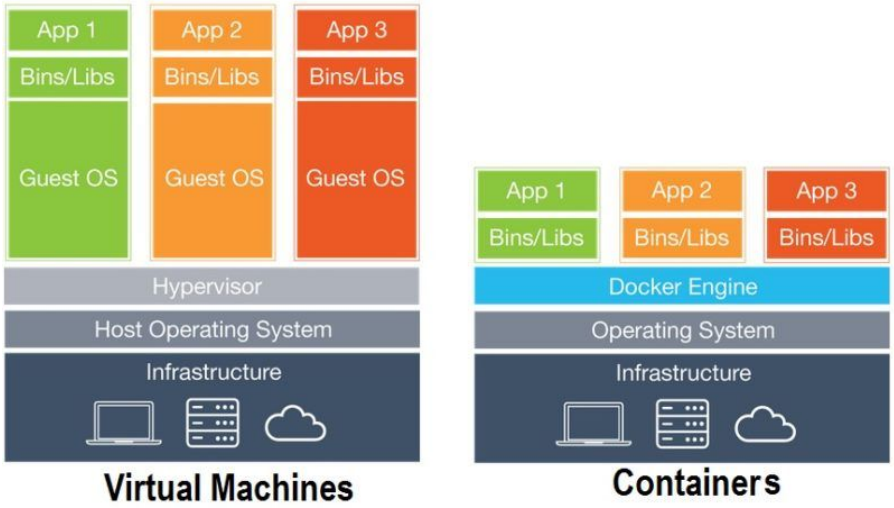
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

Intuition : utiliser Docker induit un coût non négligeable notamment lié à la conteneurisation et à la couche Docker

Objectif : étudier ce cout énergétique lié à l'utilisation de Docker

Comparaison en utilisant des applications "industrielles dockerisés"

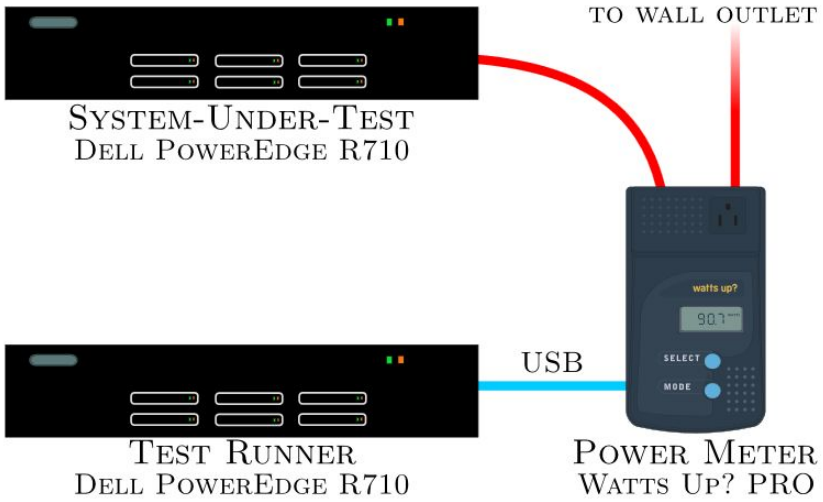


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

Comparaison en utilisant des applications "industrielles dockerisés"

Monitoring de la machine sous test avec un wattmètre physique relié en USB à une machine de mesure.



Hardware configuration of the System-Under-Test and the test runner.

CPU	2 × six-core Intel Xeon X5670 at 2.93 GHz
RAM	72 GiB ECC DDR3
Network	Gigabit Ethernet connection
Storage	146GB SAS hard drive at 15,000 RPM
Power supply	870 W (120 V ~ 12A at 60 Hz)

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

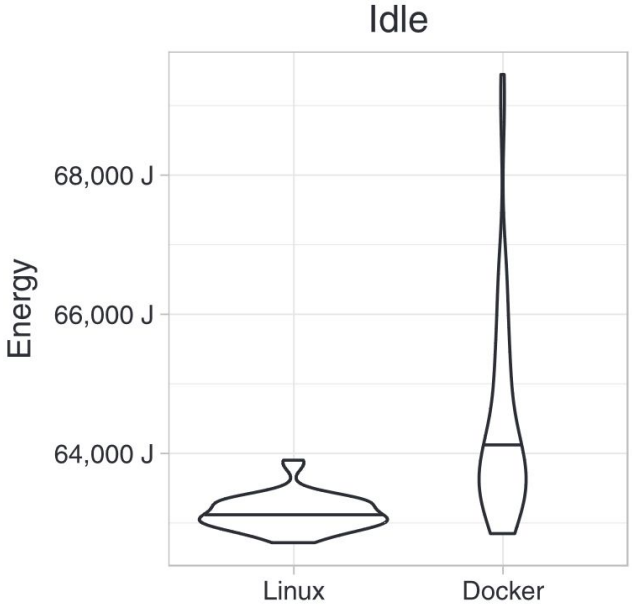
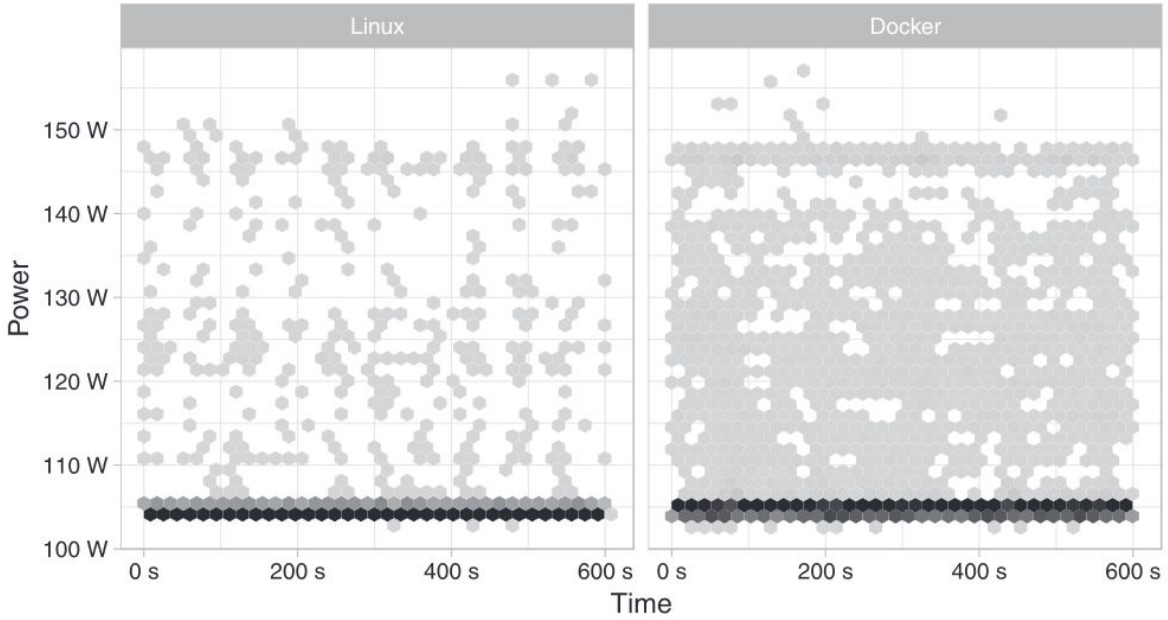


Fig. 2. Violin plot of idle energy consumption.



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie

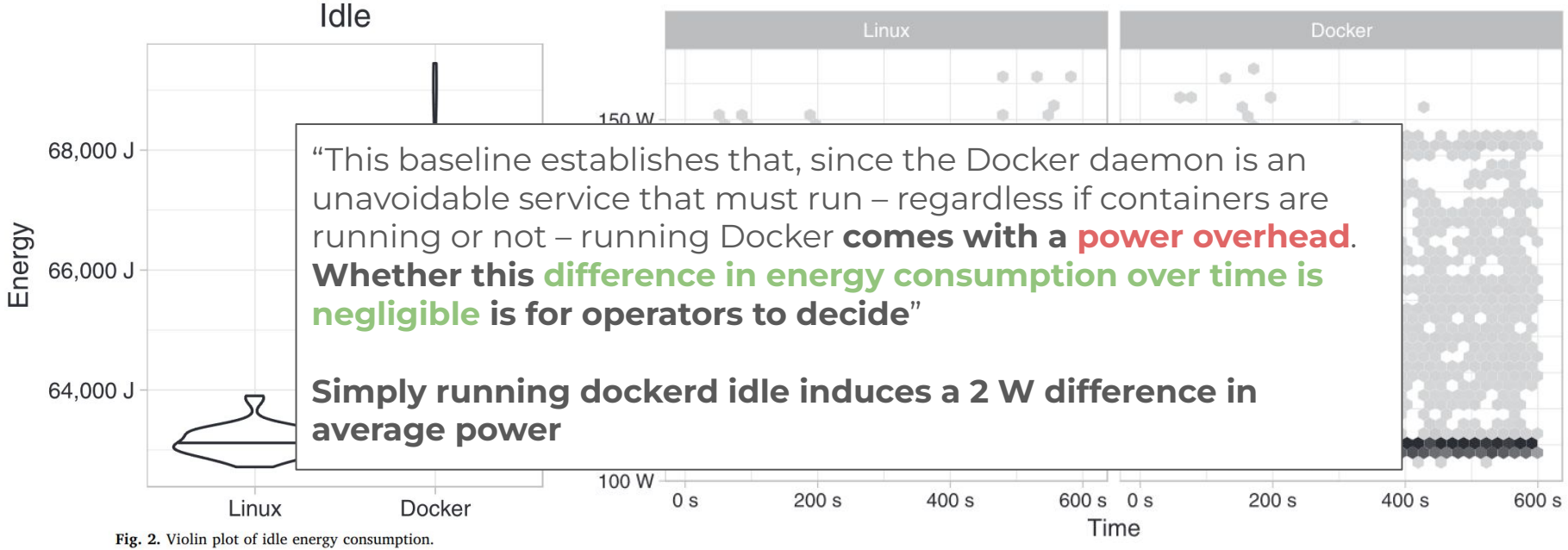


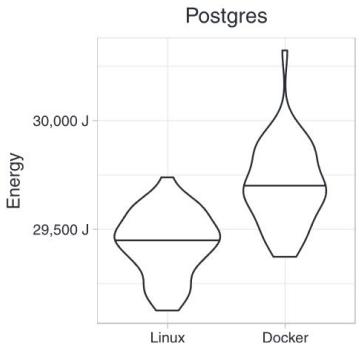
Fig. 2. Violin plot of idle energy consumption.

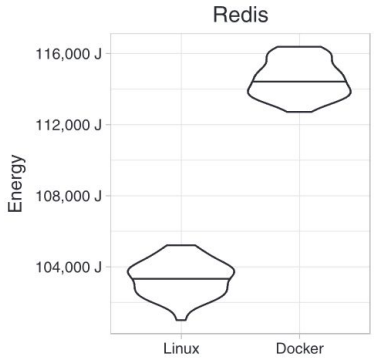
"This baseline establishes that, since the Docker daemon is an unavoidable service that must run – regardless if containers are running or not – running Docker **comes with a power overhead**. **Whether this difference in energy consumption over time is negligible is for operators to decide**"

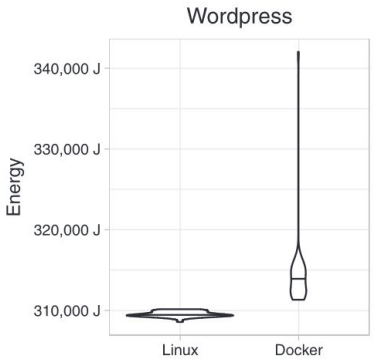
Simply running dockerd idle induces a 2 W difference in average power

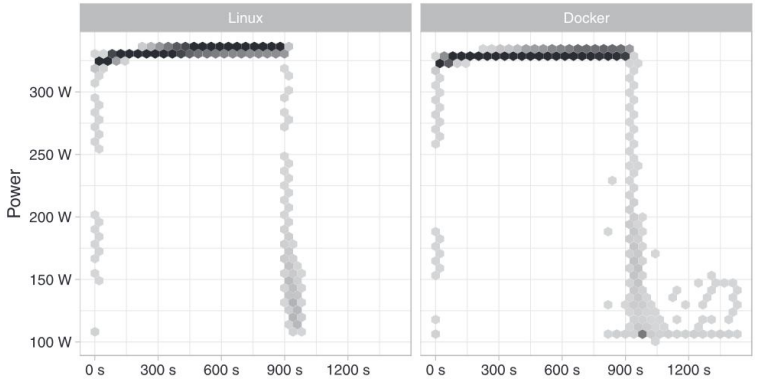
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

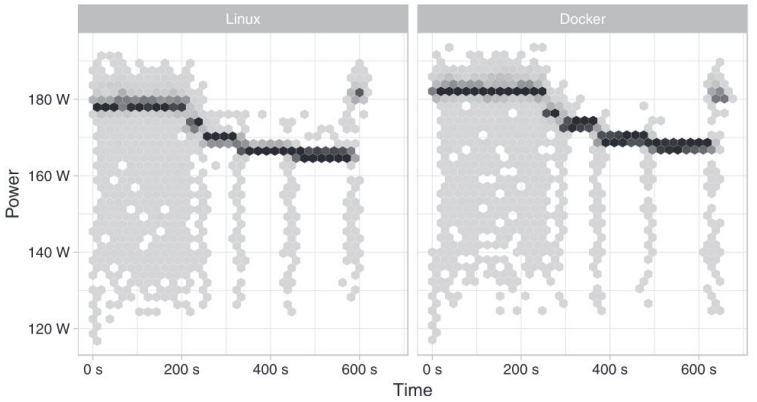
Utilisation de Docker et consommation d'énergie







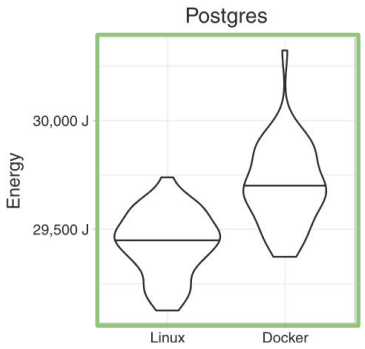


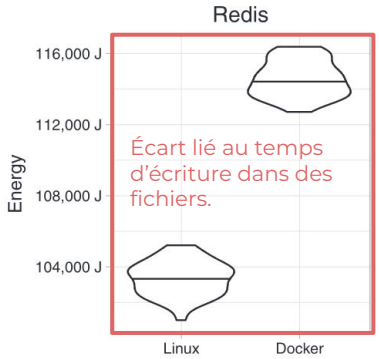


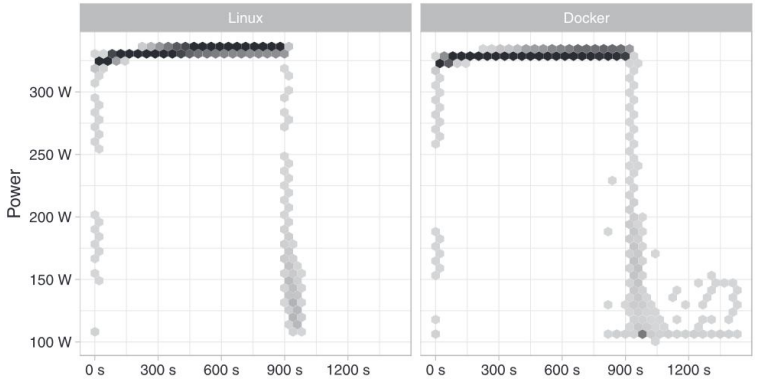
Case Study	Example Task	Linux	Docker	% Difference	$ Difference
Idle	One week idle	$1.20	$1.22	2.08%	+$0.02
WordPress				1.56%	
Redis	10^9 queries	$1.29	$1.43	10.85%	+$0.14
PostgreSQL	10^6 transactions	$0.55	$0.55	0.97%	+$0.00

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

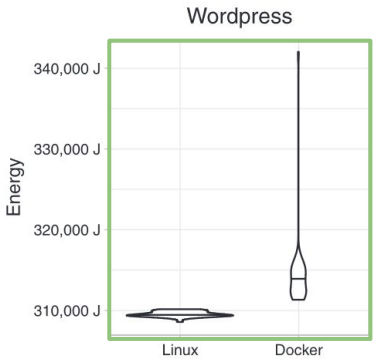
Utilisation de Docker et consommation d'énergie

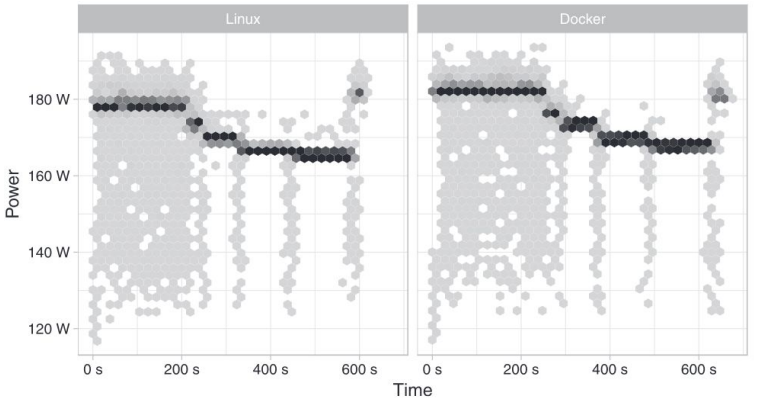






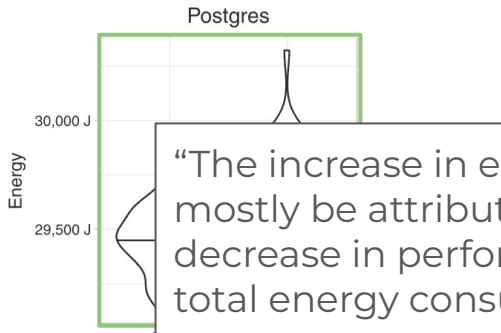
Case Study	Example Task	Linux	Docker	% Difference	$ Difference
Idle	One week idle	$1.20	$1.22	2.08%	+$0.02
WordPress				1.56%	
Redis	10^9 queries	$1.29	$1.43	10.85%	+$0.14
PostgreSQL	10^6 transactions	$0.55	$0.55	0.97%	+$0.00

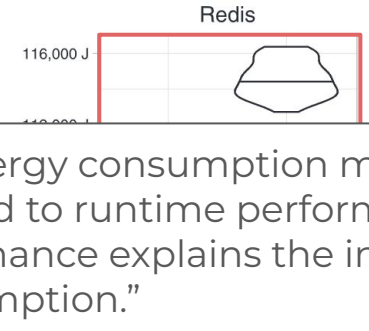


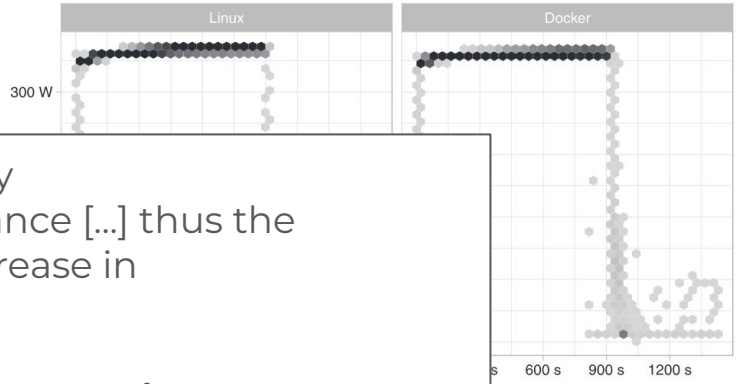


INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Utilisation de Docker et consommation d'énergie



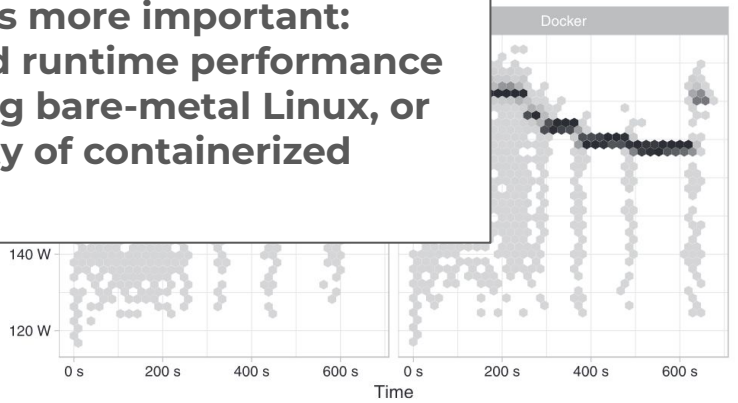




"The increase in energy consumption may mostly be attributed to runtime performance [...] thus the decrease in performance explains the increase in total energy consumption."

"Operations teams must decide which is more important: sustainability, energy consumption, and runtime performance of reduced resource usage by employing bare-metal Linux, or the process isolation and maintainability of containerized applications of Docker"

Case Study	Example Task	Lin			
Idle WordPress	One week idle	$1			
Redis	10^9 queries	$1			
PostgreSQL	10^6 transactions	$0.55	$0.55	0.97%	+$0.00



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Impact de la gestion des threads sur la consommation d'énergie

Green Computing | INFO4 | Quentin Perez | 2023-2024

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Énergie et stratégies de multi-threading

Question : quelle stratégie de multi-threading adopter pour trouver le meilleur compromis performance / énergie ?

Objectif : comparer 3 stratégies de multi-threading

Benchmark fait sur un panel de 9 projets “populaires” ayant diverses utilités (calcul mathématique, base de données, etc.)

Monitoring matériel du CPU

Understanding Energy Behaviors of Thread Management Constructs

Gustavo Pinto
Federal University of Pernambuco
ghlp@cin.ufpe.br

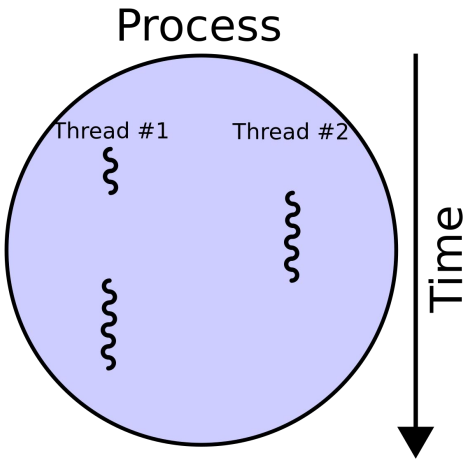
Fernando Castor
Federal University of Pernambuco
castor@cin.ufpe.br

Yu David Liu
SUNY Binghamton
davidL@binghamton.edu

Énergie et stratégies de multi-threading

3 stratégies de multi-threading comparées :

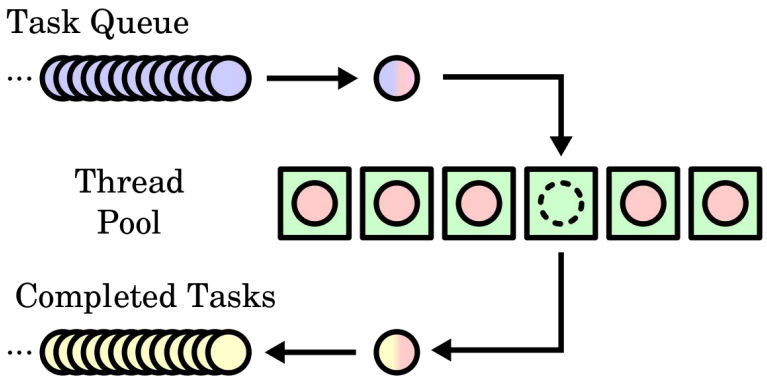
1. **Explicit Threading ⇒**
   Assignation manuelle de tâches à des threads



Énergie et stratégies de multi-threading

3 stratégies de multi-threading comparées :

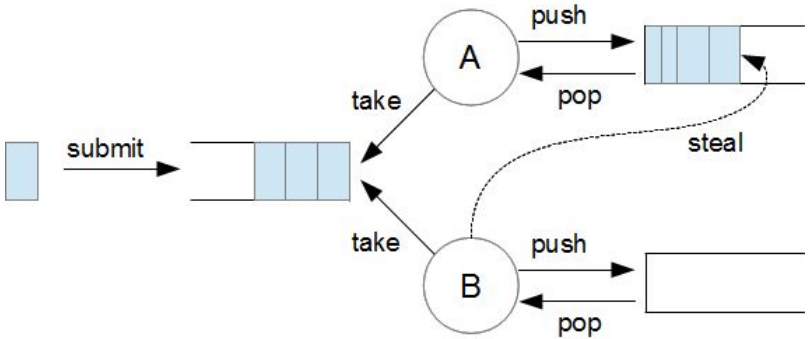
1. **Explicit Threading** ⇒ Assignation manuelle de tâches à des threads
2. **Thread Pooling (Executor Style)** ⇒ Pool de threads alimenté par un buffer de tâches (task queue)



Énergie et stratégies de multi-threading

3 stratégies de multi-threading comparées :

1. **Explicit Threading** ⇒ Assignation manuelle de tâches à des threads
2. **Thread Pooling (Executor Style)** ⇒ Pool de threads alimenté par un buffer de tâches (task queue)
3. **Work Stealing (Fork Join Style)** : Pool de threads ayant des buffers séparés et où les threads peuvent utiliser les threads peuvent “piocher” automatiquement dans les buffers voisins pour équilibrer la charge

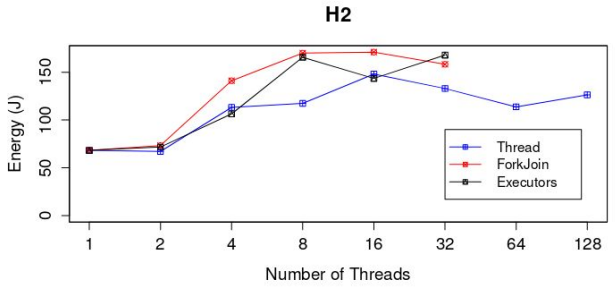


Énergie et stratégies de multi-threading

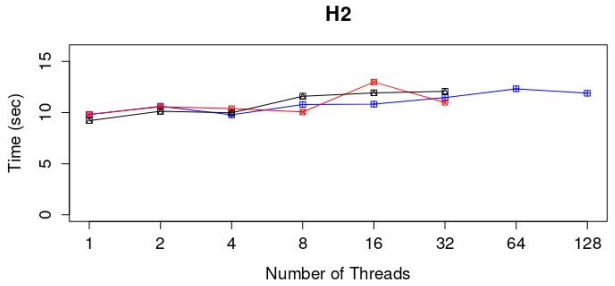
"Thread style performs well in I/O-bound" ⇒ Contrairement à ce que l'on pourrait attendre

"The Executor style and the ForkJoin style build an additional layer of thread management on top" ⇒

"**This higher layer of decision making may disagree with the OS**, missing some opportunities for context switching in the presence of long-latency I/O operations"



(c)



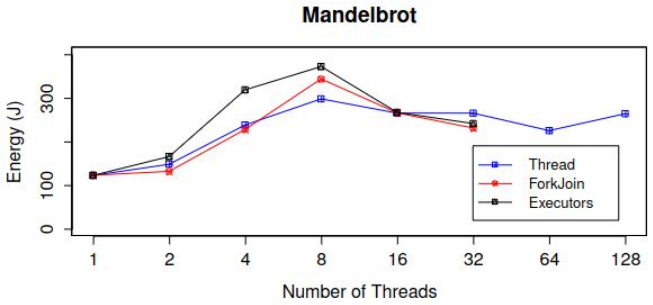
(f)

Énergie et stratégies de multi-threading

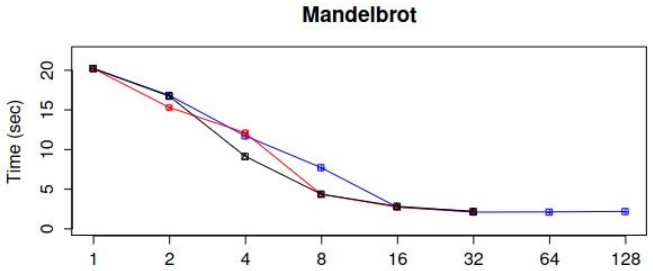
"Thread style performs well in I/O-bound" ⇒ Contrairement à ce que l'on pourrait attendre

"The Executor style and the ForkJoin style build an additional layer of thread management on top" ⇒

"**This higher layer of decision making may disagree with the OS**, missing some opportunities for context switching in the presence of long-latency I/O operations"



(g)



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

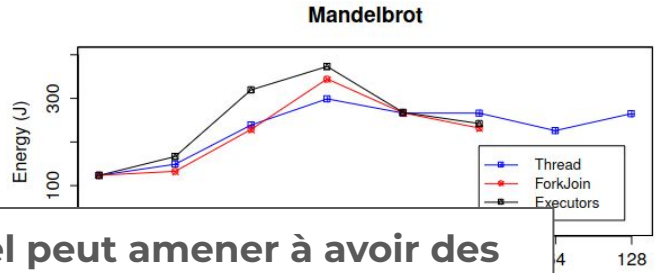
Énergie et stratégies de multi-threading

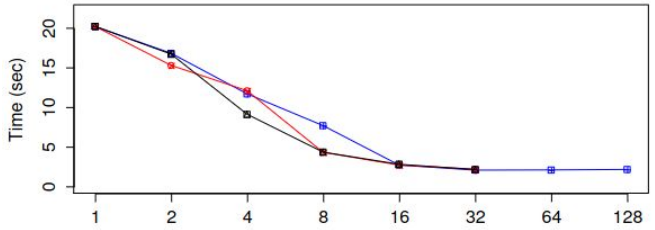
"Thread style performs well in I/O-bound" ⇒ Contrairement à ce que l'on pourrait attendre

"The Executo
style build ar
thread mana

Le fait d'utiliser un threading manuel peut amener à avoir des performances supérieures à une répartition de charge automatique à cause des coûts de synchronisation et de changement de contexte.

"**This higher layer of decision making may disagree with the OS**, missing some opportunities for context switching in the presence of long-latency I/O operations"





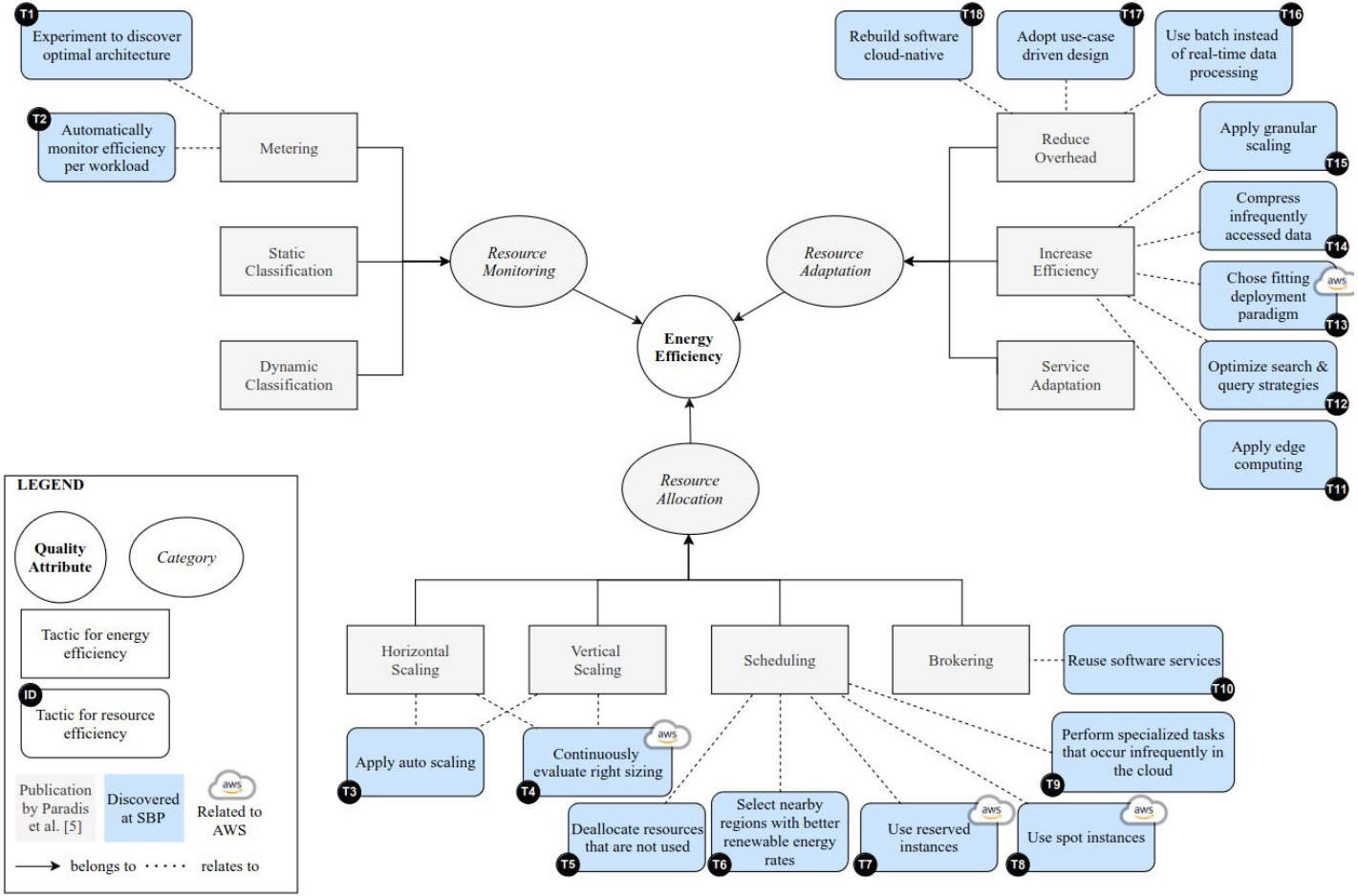
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Leviers (tactics) sur les architectures Cloud pour réduire la consommation d’énergie

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Leviers (tactics) sur les architectures Cloud pour réduire la consommation d'énergie

[Vos *et al.*, "Architectural Tactics to Optimize Software for Energy Efficiency in the Public Cloud", 2022]



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Conclusion

- La consommation d'un logiciel/service est fortement contextuelle (matériel, OS, threading, type d'usage, etc.)
- Nécessité de comparer et mesurer, car il y a des phénomènes contrintuitifs
- Plusieurs niveaux pour l'optimisation :
  - langages / pratiques d'implémentation
  - système / OS / Virtualisation (VM / Docker)
  - chaine de build / pipeline DevOps
  - Cloud et services
  - Bibliothèques utilisées
  - etc.