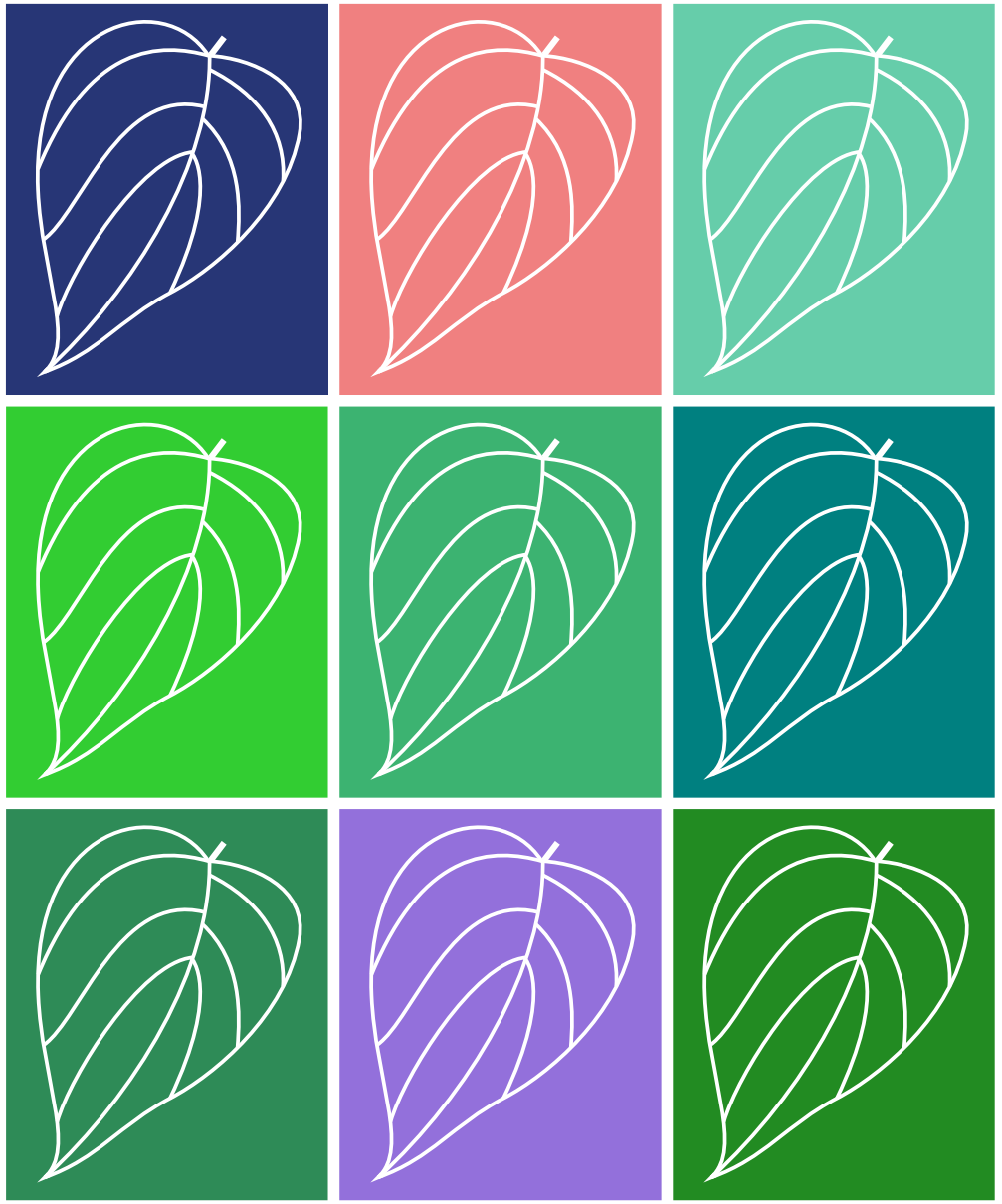
Google Developer Groups

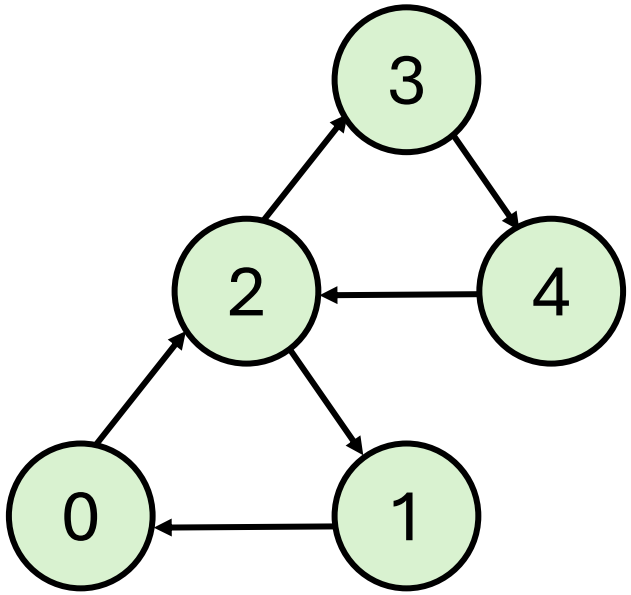
Francesco Tosoni, Philip Bille,
Valerio Brunacci,
Alessio De Angelis, Paolo Ferragina,
and Giovanni Manzini

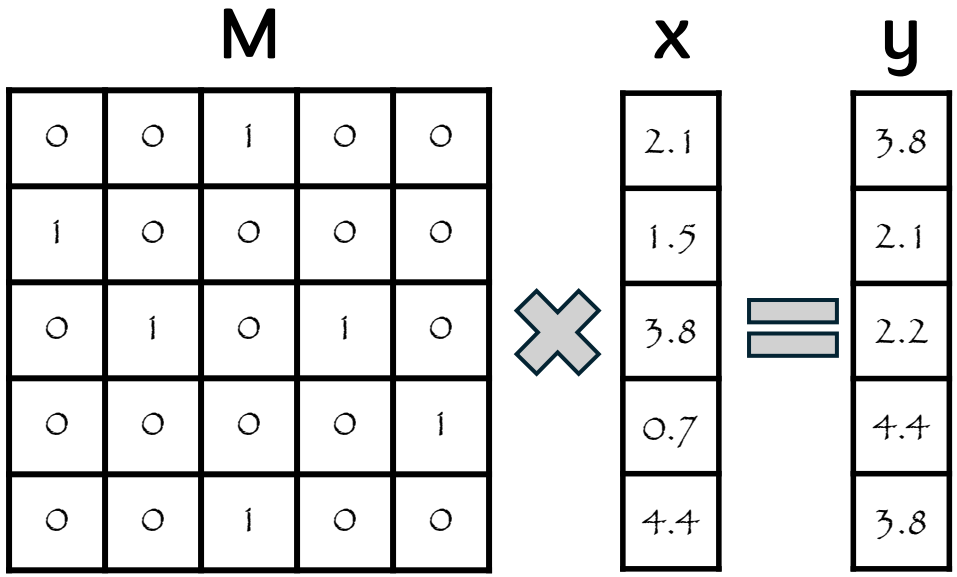
Verso operazioni più *green* su matrici tramite formati compressi senza perdita



Introduzione

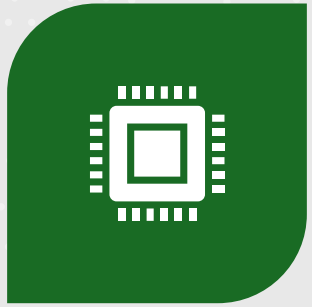
La **moltiplicazione sparsa matrice-vettore** (**SpMV**) sono rilevanti nel ML, nel calcolo scientifico e negli algoritmi su grafi.
Focus della ricerca: Investigare l'efficienza in **spazio**, **tempo** ed **energia** dell'SpMV.
Confermiamo per le strutture dati compresse risultati recenti sulla non linearità tra tempo ed energia nel calcolo multithread





Motivazione

POPOLARITÀ DI ALGORITMI DI ML (P.ES. CHATGPT, DEEPSEEK) CON UNA GENERAZIONE DI DATI CHE SUPERA LA LEGGE DI MOORE.



L’IA NON SOLO SULLE MACCHINE SERVER MA ANCHE SU DISPOSITIVI ALL’EDGE E IOT → LA DURATA DELLE BATTERIE È UN REQUISITO CRITICO.

ENERGIA COME VINCOLO PRIMARIO DI DESIGN NEI DISPOSITIVI INFORMATICI; TUTTAVIA, L'INGEGNERIA DEL SOFTWARE GREEN È ANCORA AI SUOI ESORDI.

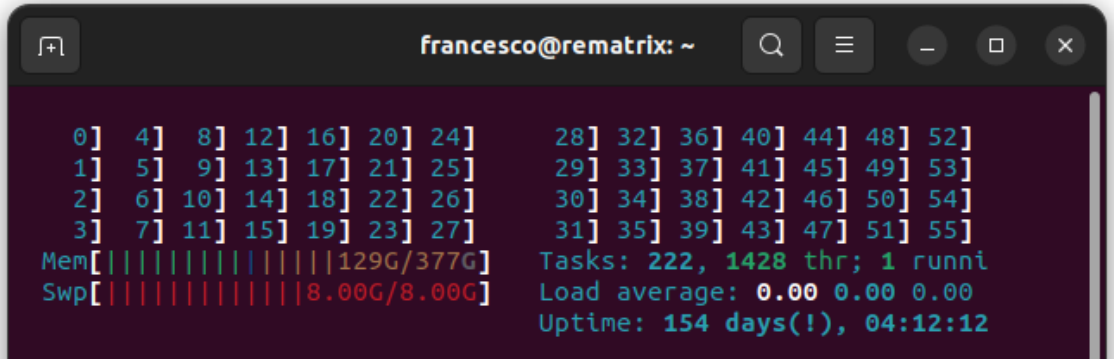
MODELLI DI COMPLESSITÀ ENERGETICA IPER-SEMPLIFICATI NON CATTURANO LE DINAMICHE DELLE ARCHITETTURE REALI → NECESSITÀ DI MODELLI DI RIFERIMENTO COMPLETI

Piattaforme

Raspberry Pi 4 model B @1.5GHz (4 core) e un **multimetro** da banco Fluke 8845A

VS.

Intel® Core™ i9-7960X (16 core, 2-way hyperthreading), un multimetro **PZEM-016** e **RAPL** (profiler di Intel)

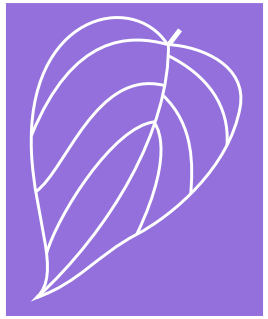


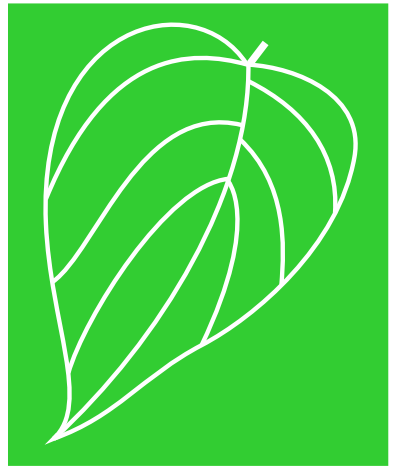
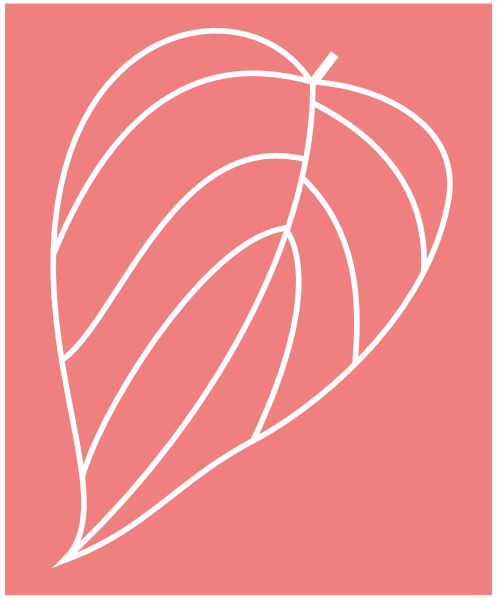
Domande di ricerca

Come influisce la compressione sull'efficienza in spazio, tempo ed energia? Alcuni formati di matrice richiedono meno spazio, ma consumano ordini di grandezza più energia.

Quale relazione sussiste tra ottimizzazione del tempo e dell’energia? Spesso il grado di parallelismo ottimale in termini energetici è inferiore a quello ottimale in termini di tempo.

Quali metriche a runtime influenzano l'efficienza energetica? Il numero di accessi alla cache L1 e L3 influisce sulle prestazioni.




Formati compressi per matrici

VLDB 2022

UNIVERSIDADE DA CORUÑA

UNIVERSIDAD DE CHILE

Formati compressi per matrici

Confrontiamo tre schemi di compressione *computation-friendly* per matrici binarie **grandi** ma **sparse**:

- Google's Zuckerli (Versari *et al.*, 2020)
- k^2-tree (Brisaboa, Ladra, and Navarro 2014)
- Matrici compresse via RePair [mm-repair] (Ferragina *et al.*, 2022)

Perché *lossless*?

Soluzioni di **compressione *lossy***:

- Bassa precisione (p.es. FP32)
- Sparsificazione
- Quantizzazione (binaria, ternaria)

→ Richiedono attenzione & applicazione manuale

La **compressione *lossless*** è una migliore alternativa «automatica»:

- indipendente dai dati
- non richiede conoscenze a priori sui dati d'input.

Compressione *computation-friendly*

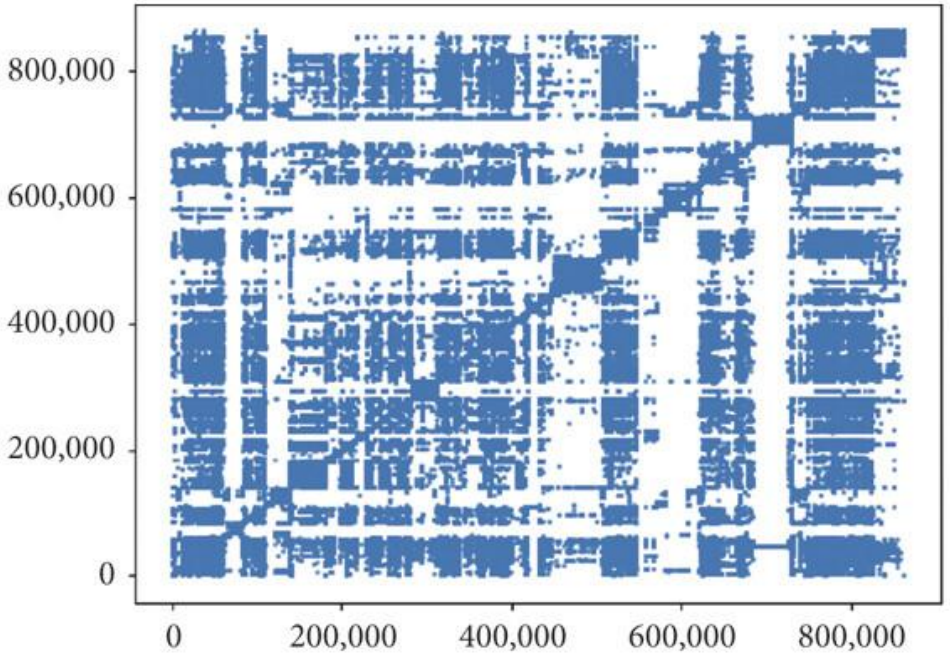
Tutti i formati testati sono *computation-friendly* per le moltiplicazioni matrice-vettore:

- Sfruttano piú della mera sparsità
- Consentono di operare nel dominio compresso
- Permettono di operare in tempo proporzionale alla dimensione della rappresentazione compressa (doppia vittoria!)

WebGraph & Zuckerli (1/2)

Sfruttare le ridondanze dei link in uscita all'interno dello stesso dominio.

- **Webgraph** (2004) copia porzioni di liste di adiacenza consecutive per comprimere ciascuna lista sulla base di una lista di riferimento (Java, C++, Rust).
- **Zuckerli** di Google (2020) applica nuove euristiche di compressione sopra Zuckerli.



Mattrice di adiacenza per for eu-2005. Figura tratta da DOI: 10.1155/2020/2354875

WebGraph & Zuckerli (2/2)

Questi formati di grafo consentono moltiplicazioni matrice-vettore *computation-friendly* (Francisco et al. 2022).

Idea: sfruttare i delta tra liste di adiacenza simili.

Implementiamo le moltiplicazioni su **Zuckerli**.

Nodo	Grado uscente	Successori
…	…	…
15	11	13, 15, 16, 17 , 18, 19, 23, 24, 203, 315, 316, 317, 3041
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
…	…	…

k^2-tree (1/3)

Sfruttare la sparsità e il clustering degli zeri.
Le sottomatrici vengono ricorsivamente suddivise in k^2 sottomatrici più piccole.

- **0**: sottomatrice vuota;
- **1**: sottomatrici non vuote.

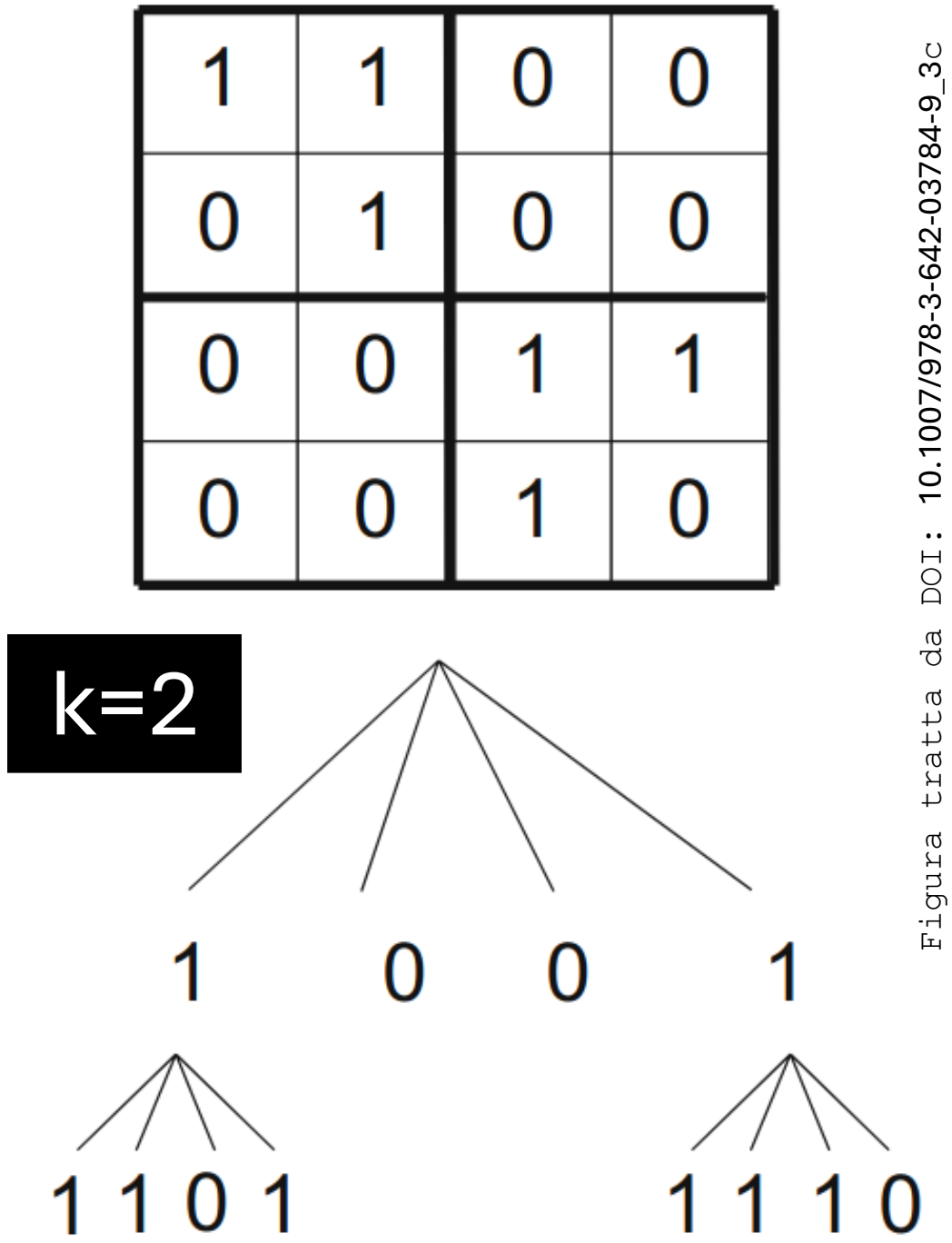


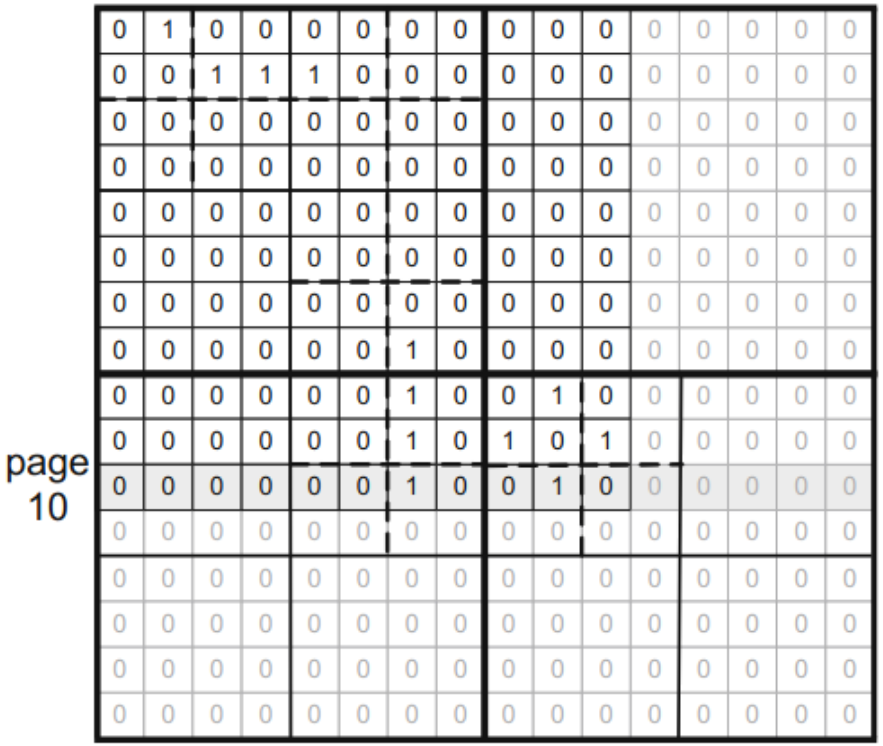
Figura tratta da DOI: 10.1007/978-3-642-03784-9_3c

k^2-tree (2/3)

L'altezza è sempre $\log_k n \Rightarrow$ l'accesso a ogni singola cella costa $O(\log_k n)$.

Moltiplicazione matrice-vettore = visita all'albero

Recupero dell'adiacenza della pagina 10



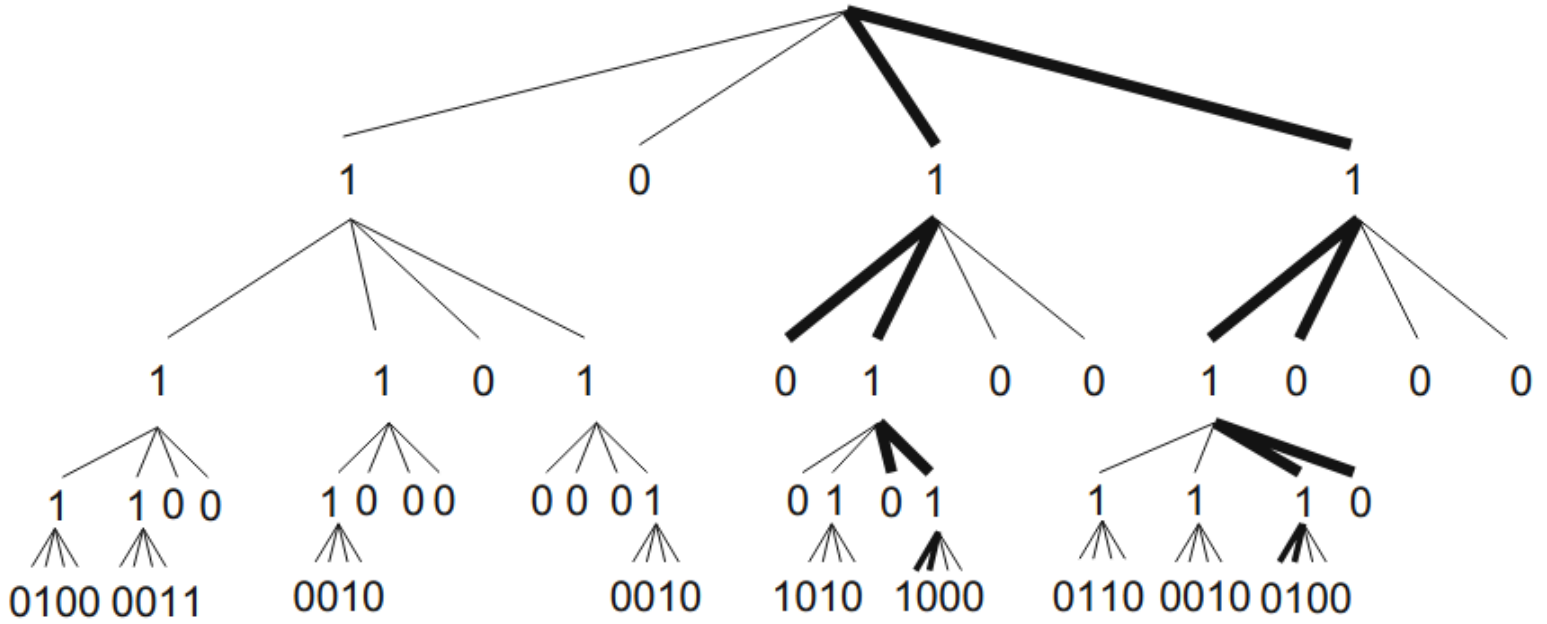


Figure taken from DOI: 10.1007/978-3-642-03784-9_3c

UNIVERSIDAD DE CHILE

UNIVERSIDADE DA CORUÑA

Millennium Institute
Foundational
Research on Data

k^2-tree (3/3)

Implementazioni:
Università della Coruña, libreria SDSL e una della Università del Cile/Istituto Millennium.
Le moltiplicazioni matrice-vettore possono essere implementate tramite una visita all'albero (in qualsiasi ordine). **Non c'è bisogno** di strutture di `rank` e `select`.

mm-repair ~ Passo #1: sfruttare la sparsità

$$\begin{matrix} 1 & 2 & 3 & 4 \end{matrix}$$
$$\begin{bmatrix} 4.1 & 5.3 & 0 & 5.3 \\ 4.1 & 0 & 2.0 & 4.1 \\ 0 & 5.3 & 2.0 & 5.3 \\ 4.1 & 5.3 & 2.0 & 4.1 \\ 4.1 & 5.3 & 2.0 & 5.3 \end{bmatrix}$$

$$\begin{matrix} 1 & 2 & 3 \end{matrix}$$
$$V = (2.0 \quad 4.1 \quad 5.3)$$

nonzeri distinti

sequenza di coppie

$$S = \langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$$$
$$\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$$$
$$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$$

Calcola la **rappresentazione CSRV di una matrice**, una modifica della rappresentazione CSR.

$ è il carattere di EOL

mm-repair ~ Passo #1: sfruttare sparsità

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ & 4.1 & 5.3 & 0 & 5.3 \\ & 4.1 & 0 & 2.0 & 4.1 \\ & 0 & 5.3 & 2.0 & 5.3 \\ & 4.1 & 5.3 & 2.0 & 4.1 \\ & 4.1 & 5.3 & 2.0 & 5.3 \end{matrix}$$

$$\begin{matrix} V = (& 1 & 2 & 3 \\ & 2.0 & 4.1 & 5.3) \end{matrix}$$

$S =$
$\langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$$
$\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$$
$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$

Generare coppie d'indici colonna-valore

$$\langle l, j \rangle$$

Indice del nonzero

Indice di colonna

mm-repair ~ Passo #1: sfruttare sparsità

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ & 4.1 & 5.3 & 0 & 5.3 \\ & 4.1 & 0 & 2.0 & 4.1 \\ & 0 & 5.3 & 2.0 & 5.3 \\ & 4.1 & 5.3 & 2.0 & 4.1 \\ & 4.1 & 5.3 & 2.0 & 5.3 \end{matrix}$$

$$\begin{matrix} & 1 & 2 & 3 \\ V = (& 2.0 & 4.1 & 5.3) \end{matrix}$$

$S = \langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$

Generare coppie d'indici colonna-valore

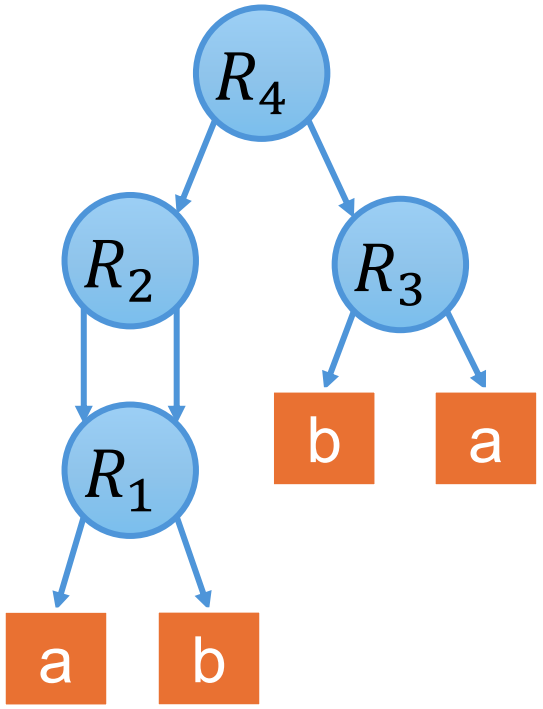
$$\langle l, j \rangle$$

Indice del nonzero

Indice di colonna

mm-repair ~ Passo #2: compressione tramite grammatiche

Utilizziamo **RePair**, un compressore senza perdita basato su grammatiche e di tipo *straight-line program* (SLP).



$$S = ababba$$

$$R_1 \rightarrow ab$$
$$R_2 \rightarrow R_1 R_1$$
$$R_3 \rightarrow ba$$
$$R_4 \rightarrow R_2 R_3$$

mm-repair ~ Passo #2: compressione tramite grammatiche

$$V = (\overset{1}{2.0} \quad \overset{2}{4.1} \quad \overset{3}{5.3})$$

$$S = \langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$$

stringa finale

$$C = N_4\$N_5\$N_6\$N_7\$N_8\$$$

$\langle 2,1\rangle$
$\langle 3,2\rangle$
$\langle 1,3\rangle$
$\langle 2,4\rangle$
$\langle 3,4\rangle$

$$N_1 \rightarrow \langle 2,1\rangle\langle 3,2\rangle$$
$$N_2 \rightarrow \langle 1,3\rangle\langle 2,4\rangle$$
$$N_3 \rightarrow \langle 1,3\rangle\langle 3,4\rangle$$
$$N_4 \rightarrow N_1\langle 3,4\rangle$$
$$N_5 \rightarrow \langle 2,1\rangle N_2$$
$$N_6 \rightarrow \langle 3,2\rangle N_3$$
$$N_7 \rightarrow N_1 N_2$$
$$N_8 \rightarrow N_1 N_3$$

R

regole

mm-repair ~ Passo #2: compressione tramite grammatiche

$$V = (\overset{1}{2.0} \quad \overset{2}{4.1} \quad \overset{3}{5.3})$$

$$S = \langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$$

stringa finale

$$C = N_4\$N_5\$N_6\$N_7\$N_8\$$$

$\langle 2,1\rangle$
$\langle 3,2\rangle$
$\langle 1,3\rangle$
$\langle 2,4\rangle$
$\langle 3,4\rangle$

$N_1 \rightarrow \langle 2,1\rangle\langle 3,2\rangle$
$N_2 \rightarrow \langle 1,3\rangle\langle 2,4\rangle$
$N_3 \rightarrow \langle 1,3\rangle\langle 3,4\rangle$
$N_4 \rightarrow N_1\langle 3,4\rangle$
$N_5 \rightarrow \langle 2,1\rangle N_2$
$N_6 \rightarrow \langle 3,2\rangle N_3$
$N_7 \rightarrow N_1 N_2$
$N_8 \rightarrow N_1 N_3$

R

regole

mm-repair ~ Passo #2: compressione tramite grammatiche

nonzeri

$$V = (\overset{1}{2.0} \quad \overset{2}{4.1} \quad \overset{3}{5.3})$$

$$S = \langle 2,1\rangle\langle 3,2\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$\langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 2,4\rangle\$ \langle 2,1\rangle\langle 3,2\rangle\langle 1,3\rangle\langle 3,4\rangle\$$

stringa finale

$$C = N_4\$N_5\$N_6\$N_7\$N_8\$$$

$\langle 2,1\rangle$
$\langle 3,2\rangle$
$\langle 1,3\rangle$
$\langle 2,4\rangle$
$\langle 3,4\rangle$

$$N_1 \to \langle 2,1\rangle\langle 3,2\rangle$$
$$N_2 \to \langle 1,3\rangle\langle 2,4\rangle$$
$$N_3 \to \langle 1,3\rangle\langle 3,4\rangle$$
$$N_4 \to N_1\langle 3,4\rangle$$
$$N_5 \to \langle 2,1\rangle N_2$$
$$N_6 \to \langle 3,2\rangle N_3$$
$$N_7 \to N_1 N_2$$
$$N_8 \to N_1 N_3$$

R

regole

mm-repair ~ Passo #3: moltiplicazione

nonzeri

$$V = \begin{matrix}1 & 2 & 3\end{matrix}\begin{pmatrix}2.0 & 4.1 & 5.3\end{pmatrix}$$

$$y = M \begin{matrix}x\\ \end{matrix}\begin{pmatrix}2\\0\\2\\4\end{pmatrix}$$

$\langle 2,1\rangle$
$\langle 3,2\rangle$
$\langle 1,3\rangle$
$\langle 2,4\rangle$
$\langle 3,4\rangle$

$$\langle l,j\rangle \quad \Rightarrow \quad eval_x(\langle l,j\rangle) = V[l]\cdot x[j]$$
$$N_i \to AB \quad \Rightarrow \quad eval_x(N_i) = eval(A) + eval(B)$$

$N_1 \to \langle 2,1\rangle\langle 3,2\rangle$
$N_2 \to \langle 1,3\rangle\langle 2,4\rangle$
$N_3 \to \langle 1,3\rangle\langle 3,4\rangle$
$N_4 \to N_1\langle 3,4\rangle$
$N_5 \to \langle 2,1\rangle N_2$
$N_6 \to \langle 3,2\rangle N_3$
$N_7 \to N_1 N_2$
$N_8 \to N_1 N_3$

R

regole

mm-repair ~ Passo #3: moltiplicazione

nonzeri

$$V = \underset{1}{(2.0} \quad \underset{2}{4.1} \quad \underset{3}{5.3)}$$

$$y = M \begin{pmatrix} x \\ 2 \\ 0 \\ 2 \\ 4 \end{pmatrix}$$

$$\langle l, j \rangle \quad \Rightarrow \quad eval_x(\langle l, j \rangle) = V[l] \cdot x[j]$$

$$N_i \rightarrow AB \quad \Rightarrow \quad eval_x(N_i) = eval(A) + eval(B)$$

8.2	$\langle 2,1 \rangle$
0	$\langle 3,2 \rangle$
4.0	$\langle 1,3 \rangle$
16.4	$\langle 2,4 \rangle$
21.2	$\langle 3,4 \rangle$

8.2	$N_1 \rightarrow \langle 2,1 \rangle \langle 3,2 \rangle$
20.4	$N_2 \rightarrow \langle 1,3 \rangle \langle 2,4 \rangle$
25.2	$N_3 \rightarrow \langle 1,3 \rangle \langle 3,4 \rangle$
29.4	$N_4 \rightarrow N_1 \langle 3,4 \rangle$
28.6	$N_5 \rightarrow \langle 2,1 \rangle N_2$
25.2	$N_6 \rightarrow \langle 3,2 \rangle N_3$
28.6	$N_7 \rightarrow N_1 N_2$
33.4	$N_8 \rightarrow N_1 N_3$

R

regole

mm-repair ~ Passo #3: moltiplicazione

nonzeri

$$V = \underset{1}{(2.0} \quad \underset{2}{4.1} \quad \underset{3}{5.3)}$$

$$y = M \begin{pmatrix} x \\ 2 \\ 0 \\ 2 \\ 4 \end{pmatrix}$$

$$\langle l, j \rangle \quad \Rightarrow \quad eval_x(\langle l, j \rangle) = V[l] \cdot x[j]$$

$$N_i \to AB \quad \Rightarrow \quad eval_x(N_i) = eval(A) + eval(B)$$

$O(|R|)$ spazio extra
$O(|R| + |C|)$ tempo

8.2	$\langle 2,1 \rangle$
0	$\langle 3,2 \rangle$
4.0	$\langle 1,3 \rangle$
16.4	$\langle 2,4 \rangle$
21.2	$\langle 3,4 \rangle$
8.2	$N_1 \to \langle 2,1 \rangle \langle 3,2 \rangle$
20.4	$N_2 \to \langle 1,3 \rangle \langle 2,4 \rangle$
25.2	$N_3 \to \langle 1,3 \rangle \langle 3,4 \rangle$
29.4	$N_4 \to N_1 \langle 3,4 \rangle$
28.6	$N_5 \to \langle 2,1 \rangle N_2$
25.2	$N_6 \to \langle 3,2 \rangle N_3$
28.6	$N_7 \to N_1 N_2$
33.4	$N_8 \to N_1 N_3$

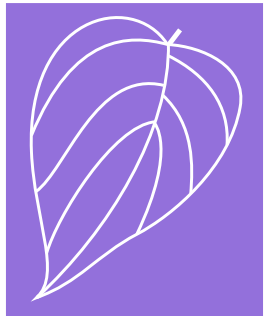
R

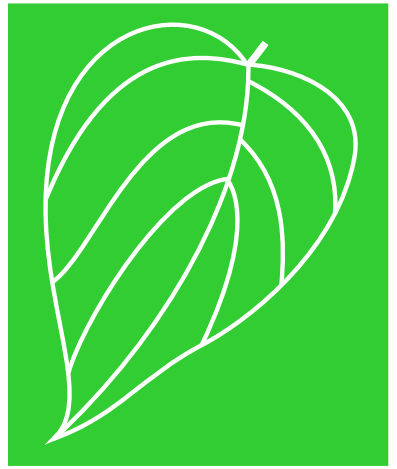
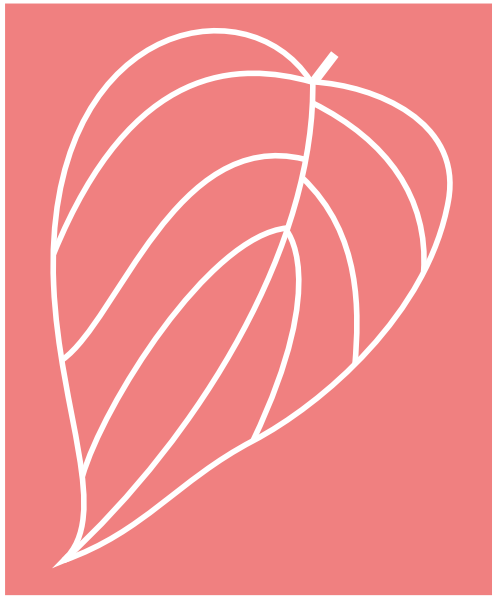
regole

	Stringa finale C	Regole R	Nonzeri V
re_32	32 bit	32 bit	64 bit
re_iv	packed arrays	packed arrays	64 bit
re_ans	ANS (codifica entropica)	packed arrays	64 bit

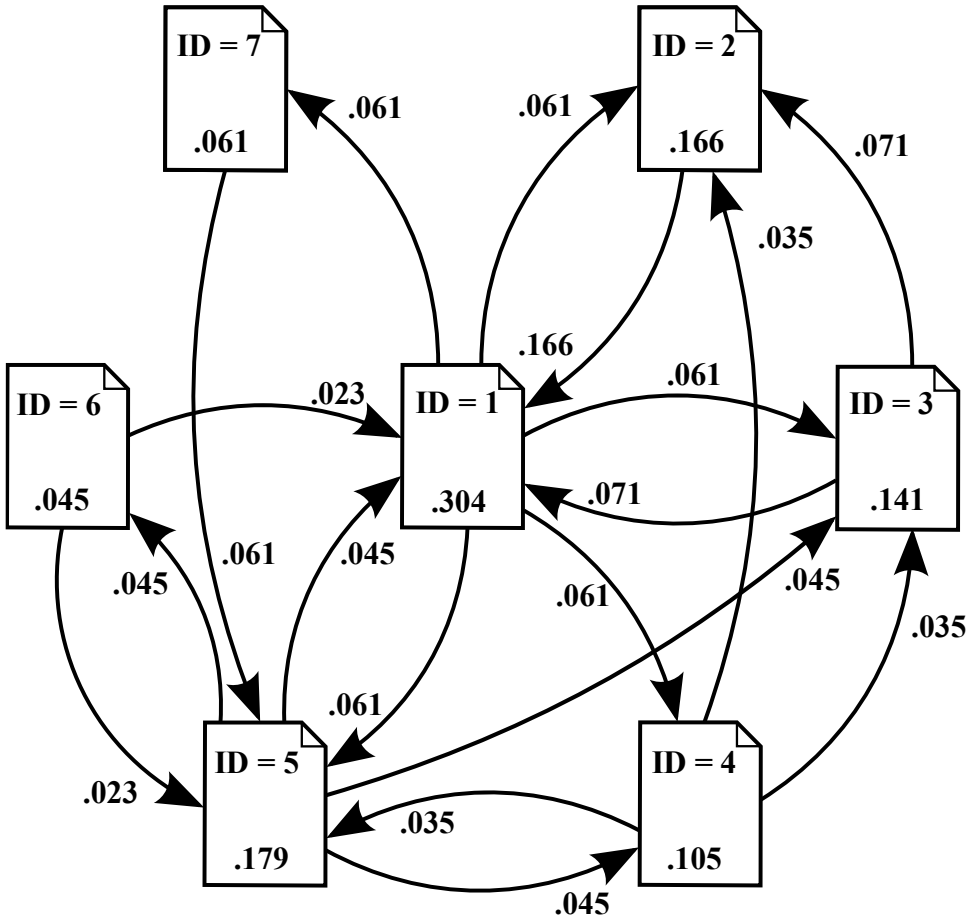
mm-repair ~ rappresentazioni fisiche

Diversi compromessi tra spazio e tempo




Setup sperimentale



PageRank

Un algoritmo classico di analisi su grafi.

Parametro di teletrasporto (α=0,15)

$$\vec{\pi}_t = \alpha \cdot \vec{\pi}_0 + (1 - \alpha) \cdot \vec{\pi}_{t-1} \cdot M$$

Distribuzione iniziale di probabilità

Moltiplicazione sinistra matrice-vettore

File:Rhacophorus nigropalmatus.jpg originariamente caricato su Commons da User:Rushenb
File:Linkstruct2.svg caricato su Commons in formato SVG da User:RobertG

Dataset	#vertici	#archi	Grafo Web?
eu-2005	862 664	19 235 140	✓
hollywood-2009	1 139 905	57 515 616	✗
in-2004	1 382 908	16 917 053	✓
ljournal-2008	5 363 260	79 023 142	✗
indochina-2004	7 414 866	194 109 311	✓
uk-2002	18 520 486	298 113 762	✓
arabic-2005	22 744 080	639 999 458	✓
uk-2005	39 459 925	936 364 282	✓
it-2004	41 291 594	1 150 725 436	✓

Datasets

Dal framework WebGraph, tramite la SuiteSparse Matrix Collection. Molti dei grafi derivano da **Web crawling**, con i vertici ordinati lessicograficamente in base all'URL invertito.

Abbiamo incluso anche due grafici di social network:

- `hollywood-2009`: grafo non orientato che rappresenta attori cinematografici; gli archi collegano attori che hanno recitato nello stesso film.
- `ljournal-2008`: grafo orientato che illustra le amicizie (asimmetriche) nel social network LiveJournal.

Architetture 📜

Intel® Core™ i9-7960X

•**CPU**: single-socket Intel® Core™ i9-7960X
•**Core**: 16 core fisici (Intel® Hyper-Threading a 2 vie)

Memoria:
•RAM totale: 384 GB (12 x 32 GB DDR4)
•Velocità: 2666 MT/s

Sistema operativo:
•Ubuntu 22.04.3 LTS (64 bit)

Architettura cache:
•L1d: 32 KiB (8-way set associative)
•L1i: 32 KiB (8-way set associative)
•L2: 1024 MiB (16-way set associative)
•L3: 22 MiB (11-way set associative)

Raspberry Pi 4 Model B

•**CPU**: 4 x ARM Cortex-A72 @ 1.5GHz
•**Core**: 4 core fisici

Memory:
•Total RAM: 4 GB LPDDR4 SDRAM

Operating system:
•Ubuntu Server 24.04 LTS (64-bit)

Cache architecture:
•L1d: 128 KiB per core; L1i: 192 KiB per core
•L2: 1 MB condiviso (16-way set associative)
•No L3!

Datasets testati: $\leq 10^7$ vertici

Setup del codice

Pre-elaborazione

Trasporre la matrice

Comprimere tramite Zuckerli, k^2-tree, mm-repair

Mantenere array con grado uscente O[1, n] per ogni vertice

Esecuzione

Eseguire100 iterazioni di PageRank

Ripetere per ogni formato compresso e ogni dataset

Compilazione

Tutti i codici sono scritti in C/C++

Compilati con -O3 flag (massimo livello di ottimizzazione) → risparmi in energia (<43% per raffronto con –O0)

Parallelismo dei dati

1.2	3.4	5.6	0	2.3
2.3	0	2.3	4.5	1.7
1.2	3.4	2.3	4.5	0
3.4	0	5.6	0	2.3
2.3	0	2.3	4.5	0
1.2	3.4	2.3	4.5	3.4

b=3

Sfruttiamo architetture multicore con versioni di **parallelismo dei dati**.

Dividiamo ogni matrice in blocchi di b righe. $y = Mx$ consiste in b moltiplicazioni indipendenti su un singolo blocco.

Utilizziamo C/C++ con thread POSIX (Pthread) e meccanismi di fork-join

Misura della potenza sulla Intel® Core™ i9-7960X

Misura della **potenza** assorbita attraverso un multimetro **PZEM-016** (~20 EUR). Nel range di potenza della nostra macchina, esegue stime con un errore dello 0,5%.

Interfaccia **Intel RAPL** (Running Average Power Limit), accessibile tramite il profiler Linux `perf` (versione 5.15.149).

- Stima di energia a livello *package*: include core, controller di memoria, cache di ultimo livello e altri componenti
- **Accuratezza disputata.** Integrato con strumenti come Scaphandre, CodeCarbon e Green Metrics Tool, ma contributi recenti in letteratura hanno dimostrato che RAPL può commnettere errori fino al 150%.
- Integrato in `perf`, che permette anche di contare cicli CPU, hit/miss della cache e conteggi delle istruzioni.

Misura della corrente
sul Raspberry Pi 4

Il voltaggio (5V) è costante ⇒ la potenza è proporzionale alla corrente.

Usiamo un multimetro da banco **Fluke 8845A** (~1000 EUR).

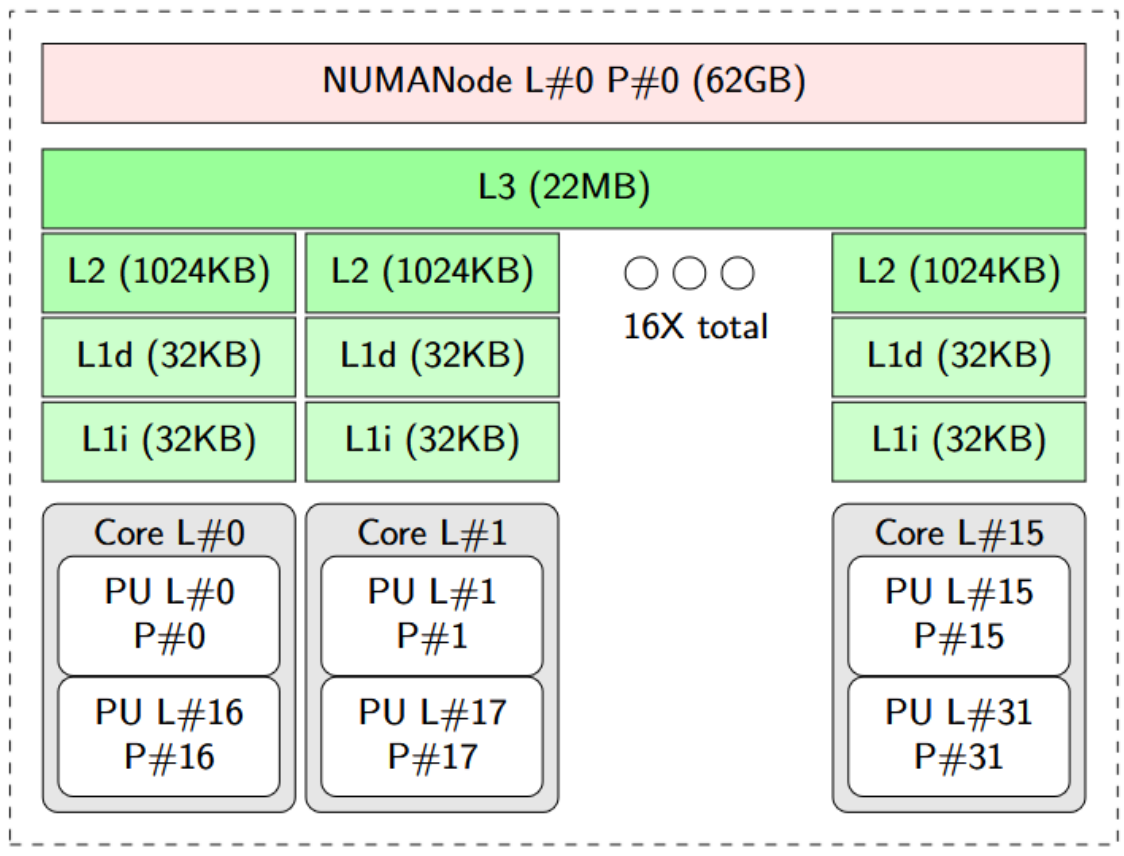
- Multimetro collegato in serie col cavo di alimentazione USB-C.
- Range di $R = 10$ A
- Risoluzione di 4½ cifre
- Incertezza di misura $\epsilon = k_1 i + k_2 R$ con $k_1 = 0{,}15\%$ e $k_2 = 0{,}0080\%$
- Frequenza di campionamento di 3 s^{-1}.

Setup per riproducibilità

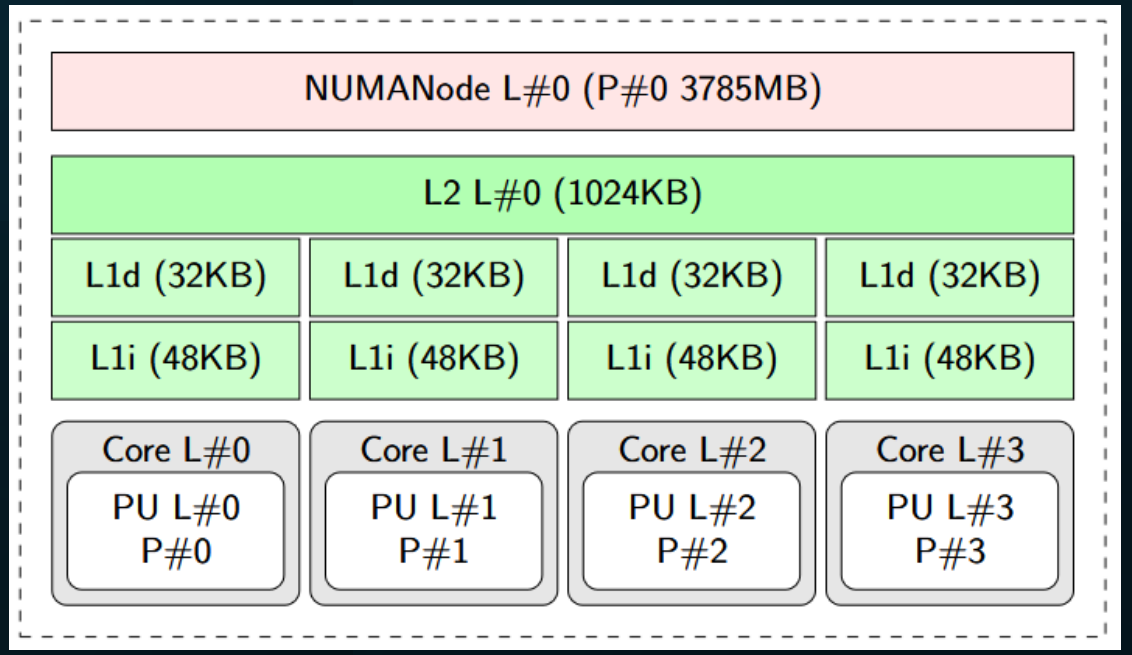
- Macchine dedicate
- Dimensione del problema limitata alla memoria principale (RAM) per evitare swapping ⇒ dataset più piccoli su Raspberry Pi
- Affinità della CPU: `pthread_setaffinity_np` per assegnare thread ai core logici ⇒ massimizzare il parallelismo anche tramite l'Hyper-Threading.
- Ventole a velocità fissa: massima velocità per un consumo energetico costante.
- Frequenza della CPU fissa tramite DVFS `powersave`.
- Verifica della coerenza dei risultati di PageRank tra diversi formati di matrice.

Obiettivo: Misurazioni affidabili e riproducibili del consumo energetico del nostro algoritmo di PageRank.

Architetture: `lstopo`

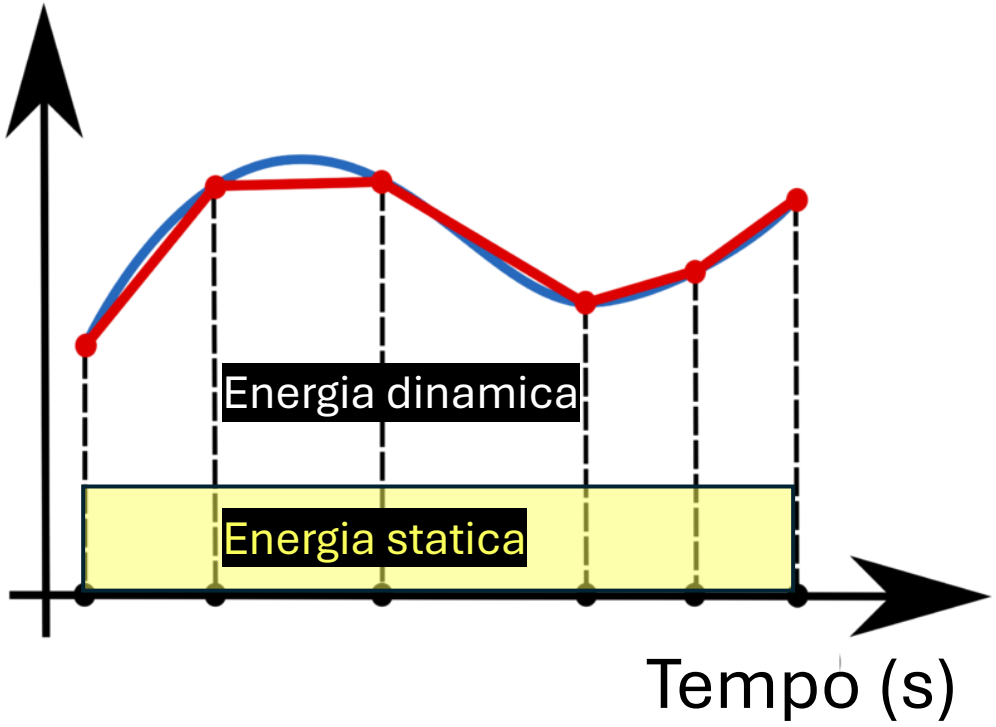


Intel® Core™ i9-7960X



Raspberry Pi 4

Energia dinamica



Consideriamo l'**energia dinamica** E_D dissipata durante l'esecuzione degli algoritmi.

$$E_D = E_T - P_{idle}T = \int (p(t) - P_{idle})dt$$

L'**accuratezza** della misura viene quindi scalata sulla componente dinamica stimata

$$A = \frac{P_D - \epsilon}{P_D}$$

dove $P_D = E_D/T$ è la potenza dinamica media.

Risultati e discussione

Baseline per i rapporti di compressione

dataset	re_32	re_iv	re_ans	k^2-tree	Zuckerli	gzip	xz
eu-2005	9.56	7.58	6.76	4.07	2.26	2.05	0.46
hollywood-2009	14.17	11.26	10.52	7.22	4.22	2.18	0.87
in-2004	11.38	8.98	7.26	2.92	1.31	1.93	0.39
ljournal-2008	26.99	21.34	19.42	14.27	9.69	2.71	1.52
indochina-2004	5.86	4.93	4.09	2.41	0.79	1.87	0.27
uk-2002	9.34	8.23	6.67	3.12	1.29	1.94	0.40
arabic-2005	6.20	5.49	4.64	2.75	0.96	1.89	0.32
uk-2005	6.79	6.12	5.03	2.72	0.96	1.90	0.34
it-2004	6.16	5.56	4.64	2.76	0.97	1.89	0.32

piú compresso

Spazio su disco

`gzip`: Rapporti di compressione variabili rispetto a Zuckerli

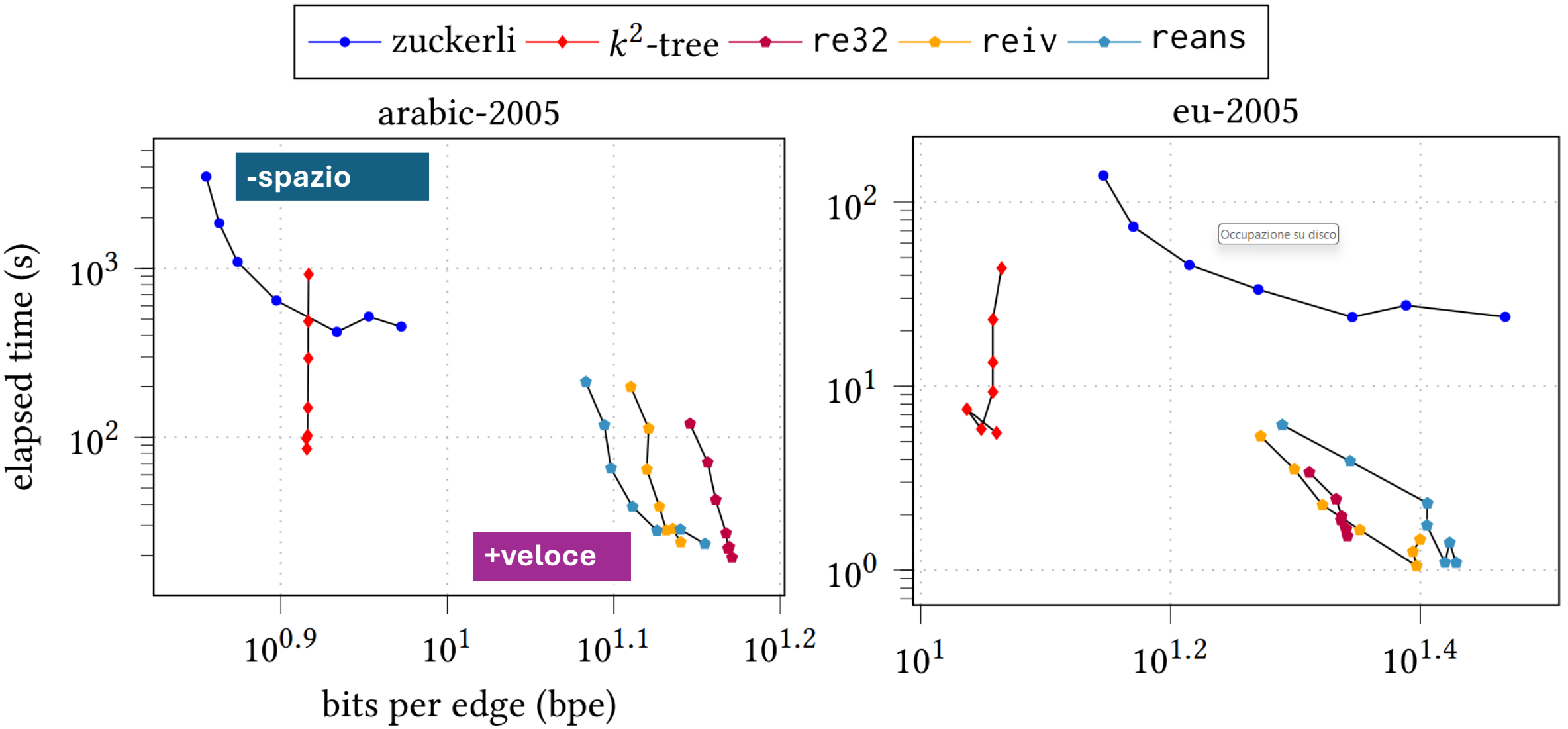
`xz`: Il più compresso ma anche il più lento in (de)compressione

L'utilizzo della memoria supera lo spazio su disco a causa di

- Strutture dati temporanee aggiuntive
- Suddivisione in blocchi per la moltiplicazione multithread

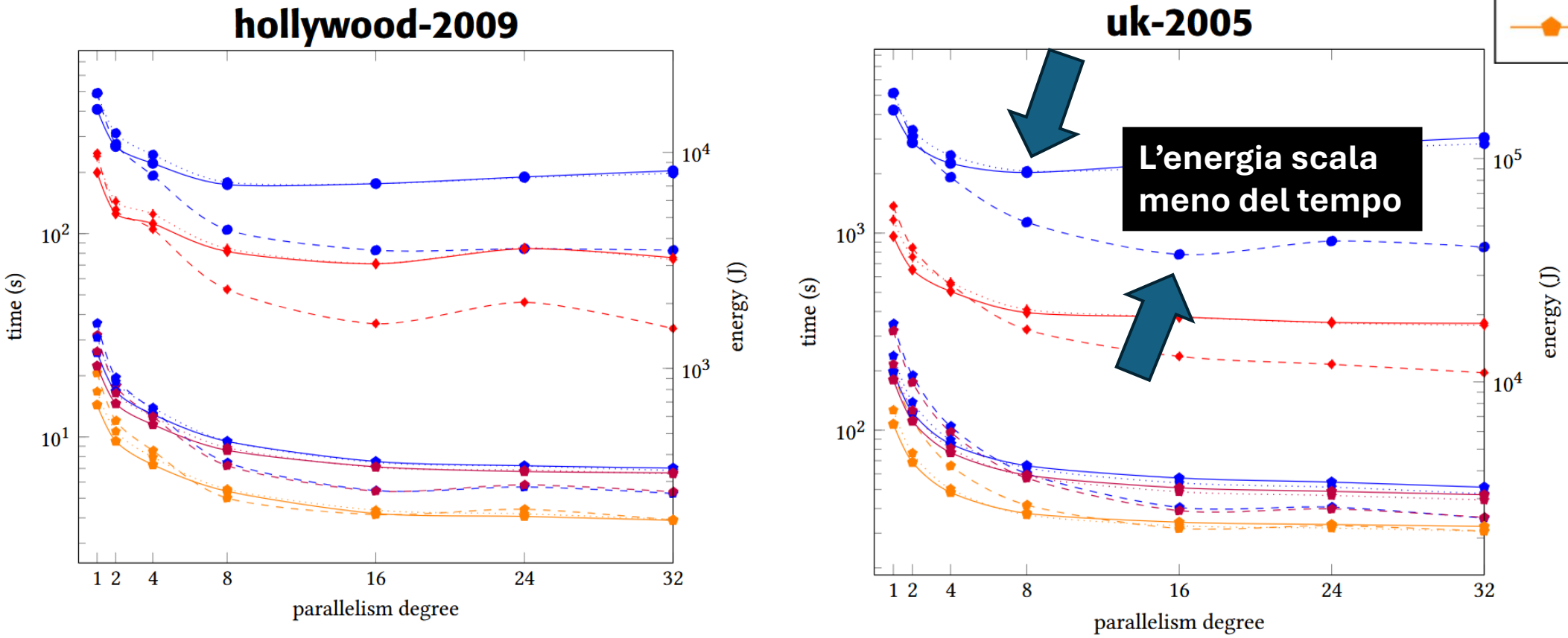
Prestazioni in spazio e tempo

sull'Intel® Core™ i9-7960X



Prestazioni in tempo (___) ed energia (___)

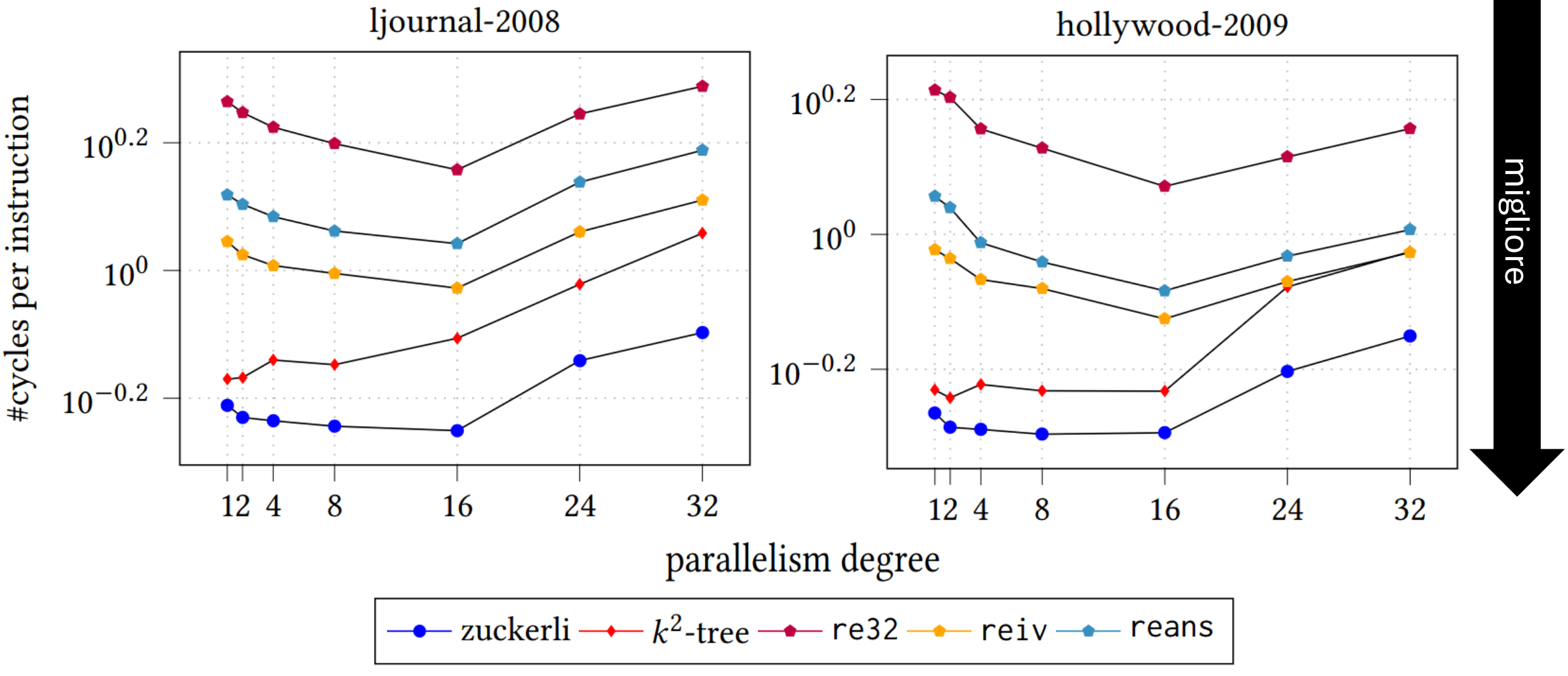
sull'Intel® Core™ i9-7960X



Approccio multi-criterio nelle applicazioni multi-thread
(p.es., selezionare la soluzione più veloce entro i vincoli energetici)

Throughput d'istruzioni
sull'Intel® Core™ i9-7960X

Zuckerli e il k^2-tree hanno throughput maggiori

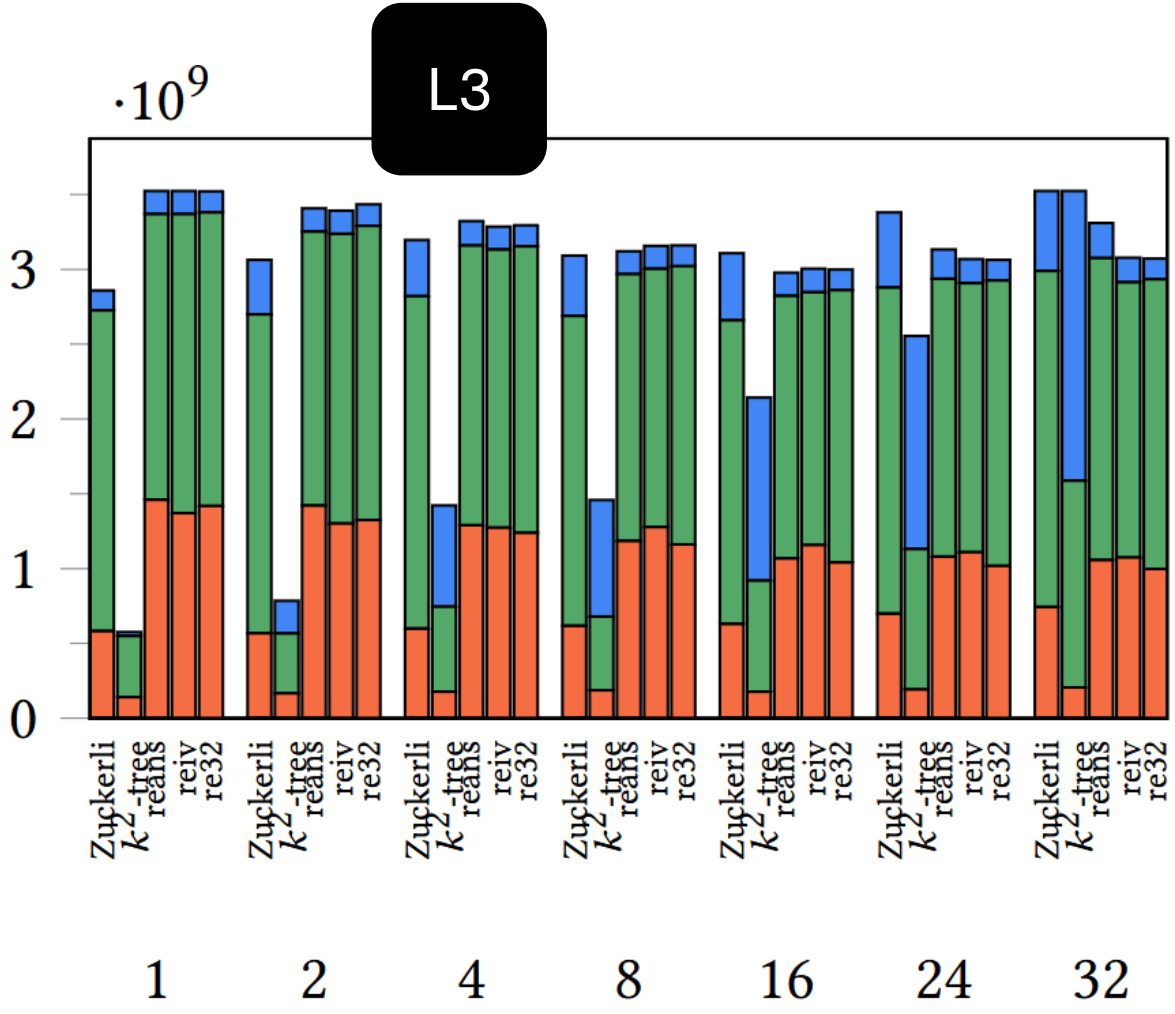


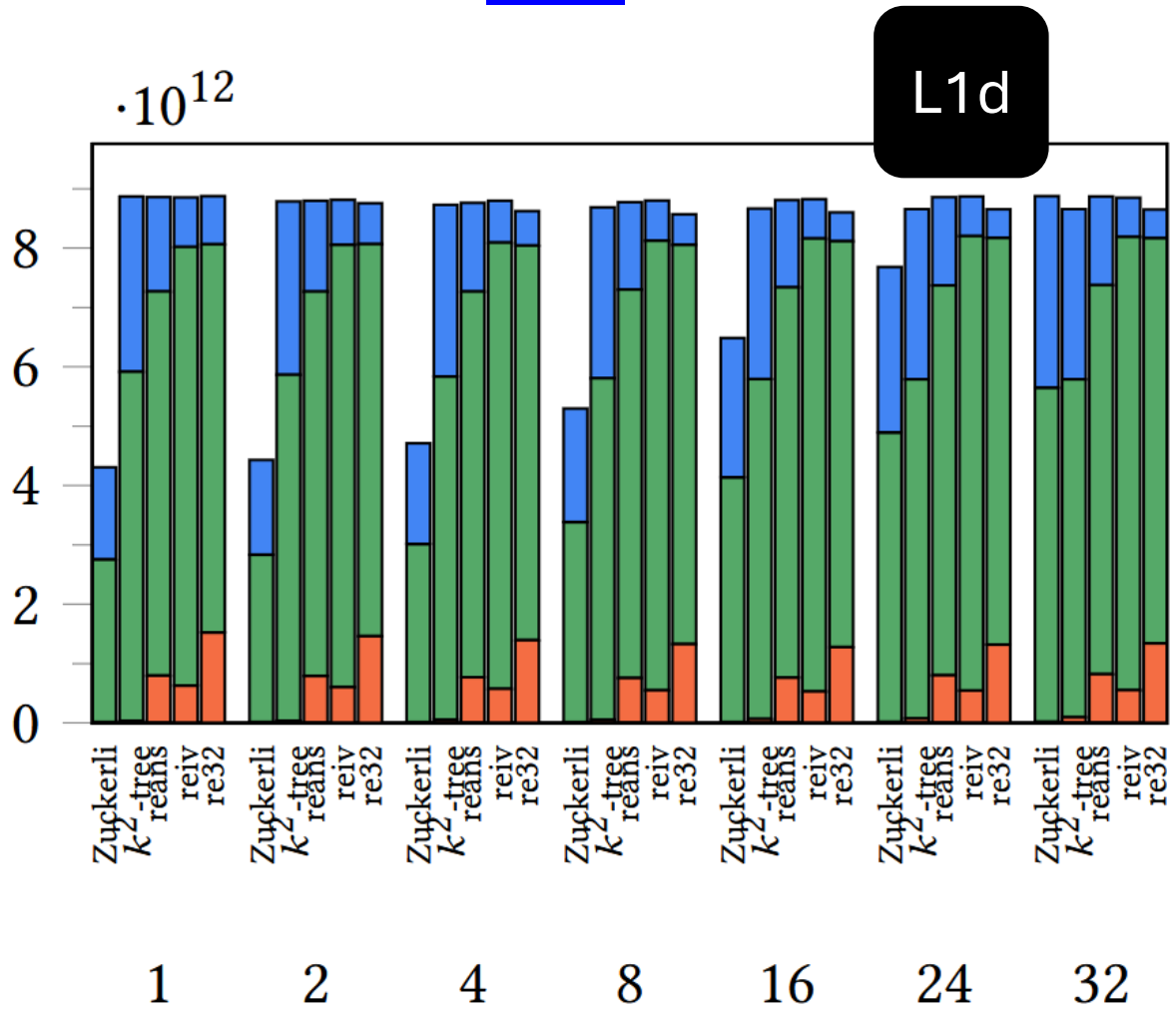
Pattern d’accesso alle cache L1d/L3

sull'Intel® Core™ i9-7960X

Dataset: ljournal-2008

load cache misses,
load cache hits,
operazioni di store.



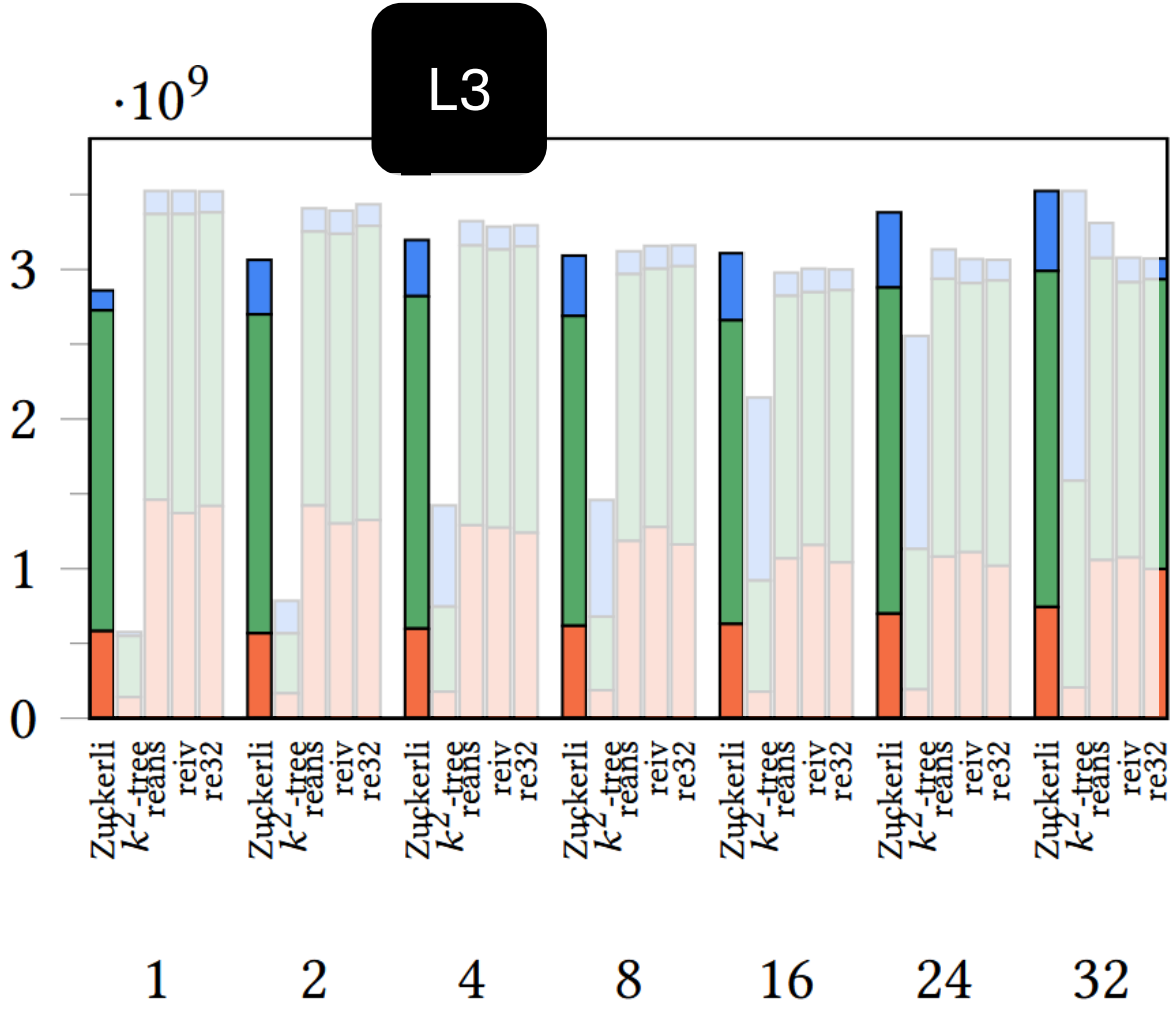


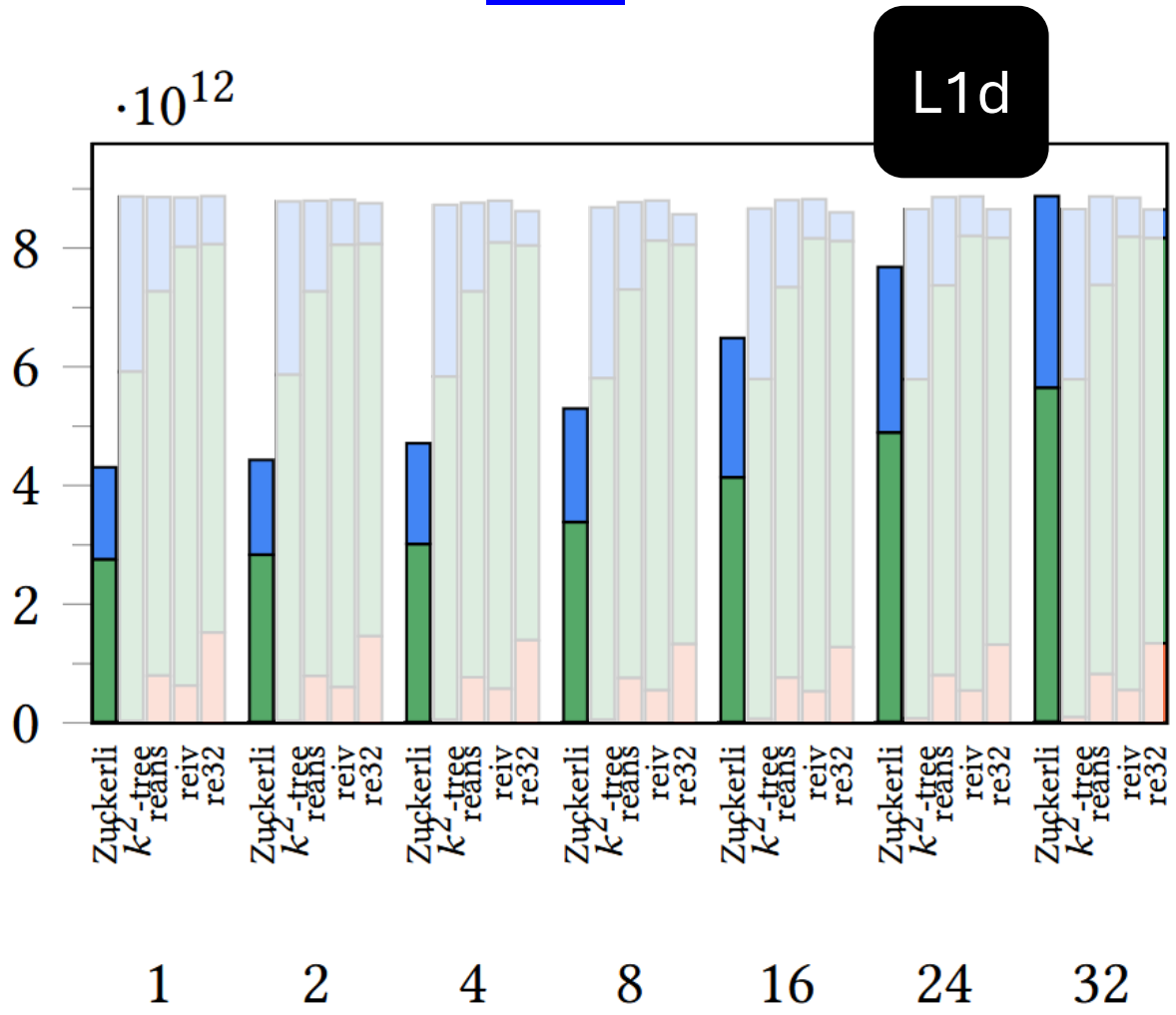
Pattern d’accesso alle cache L1d/L3

sull'Intel® Core™ i9-7960X

Dataset: ljournal-2008

load cache misses,
load cache hits,
operazioni di store.



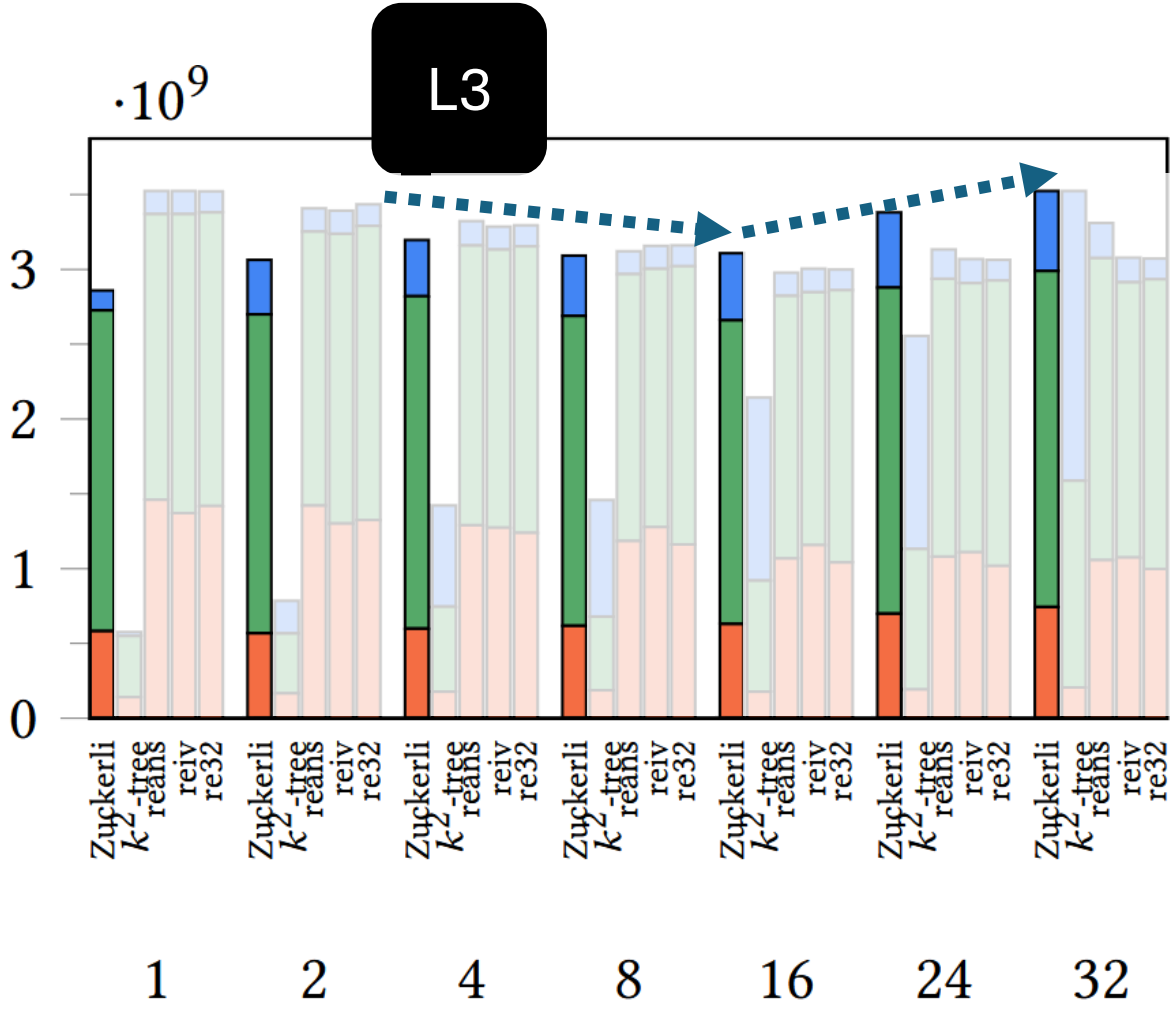


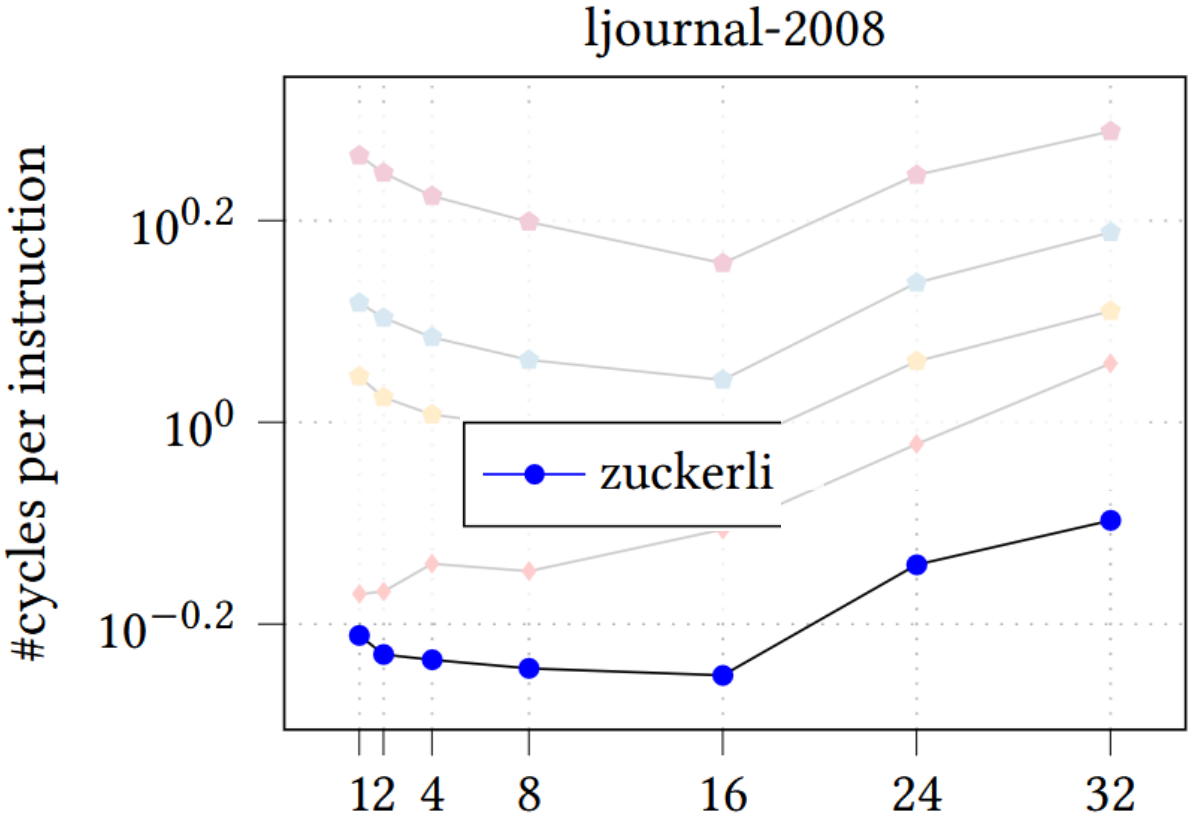
Pattern d’accesso alle cache L1d/L3

sull'Intel® Core™ i9-7960X

Dataset: ljournal-2008

load cache misses,
load cache hits,
operazioni di store.



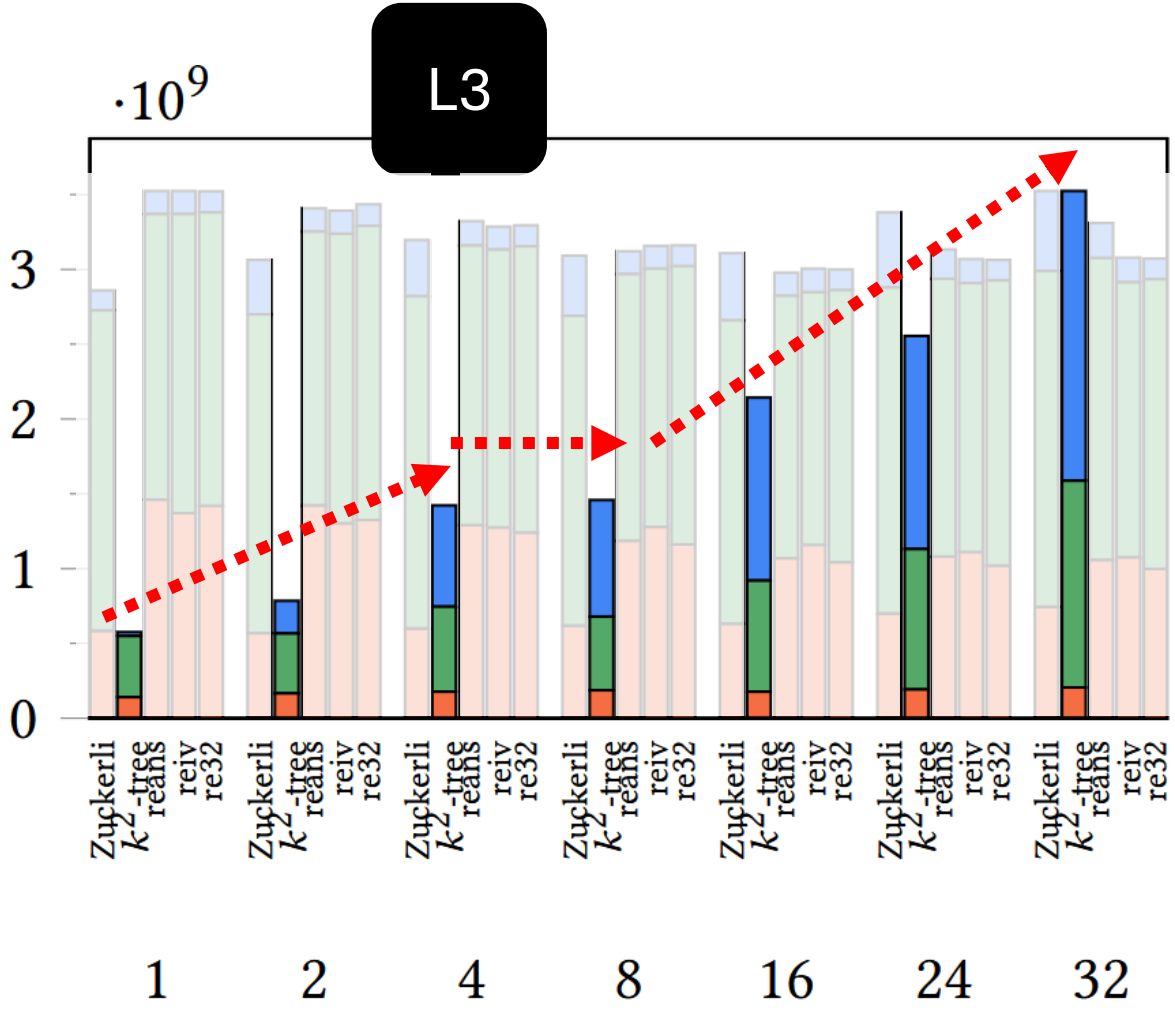


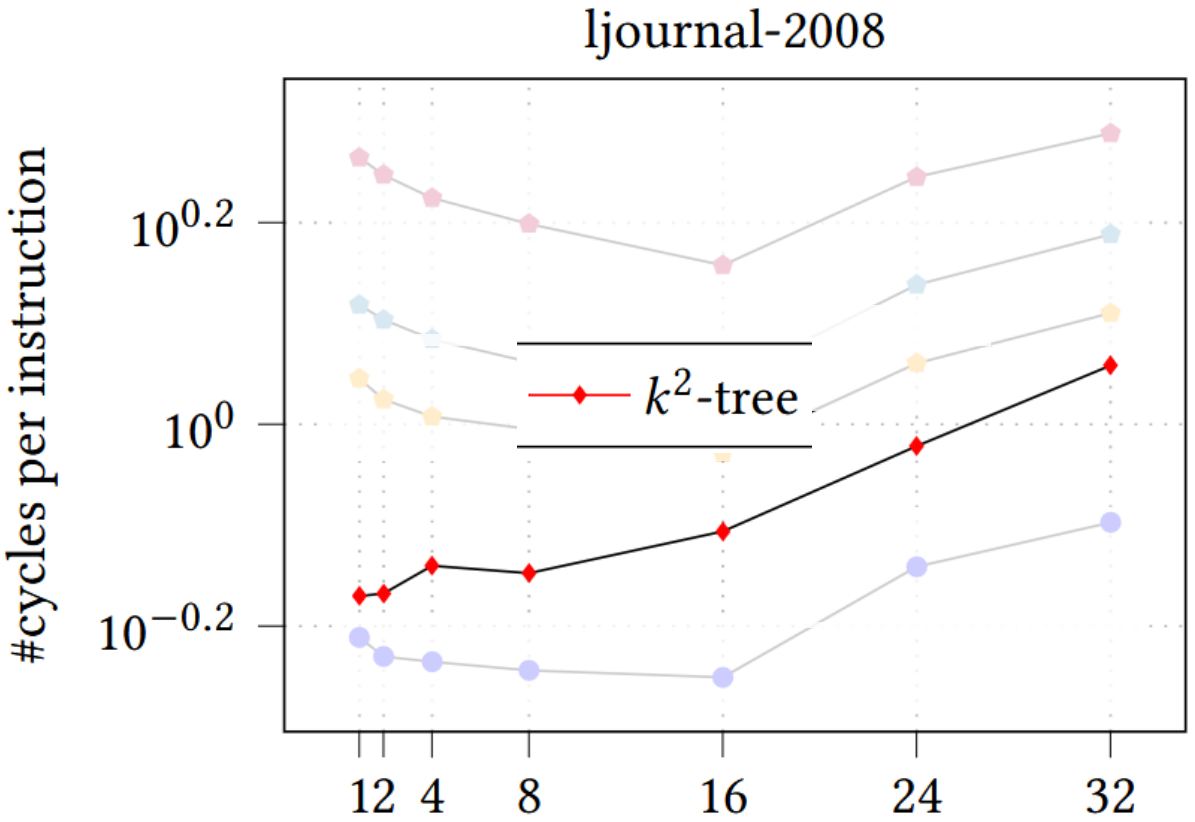
Pattern d’accesso alle cache L1d/L3

sull'Intel® Core™ i9-7960X

Dataset: ljournal-2008

load cache misses,
load cache hits,
operazioni di store.

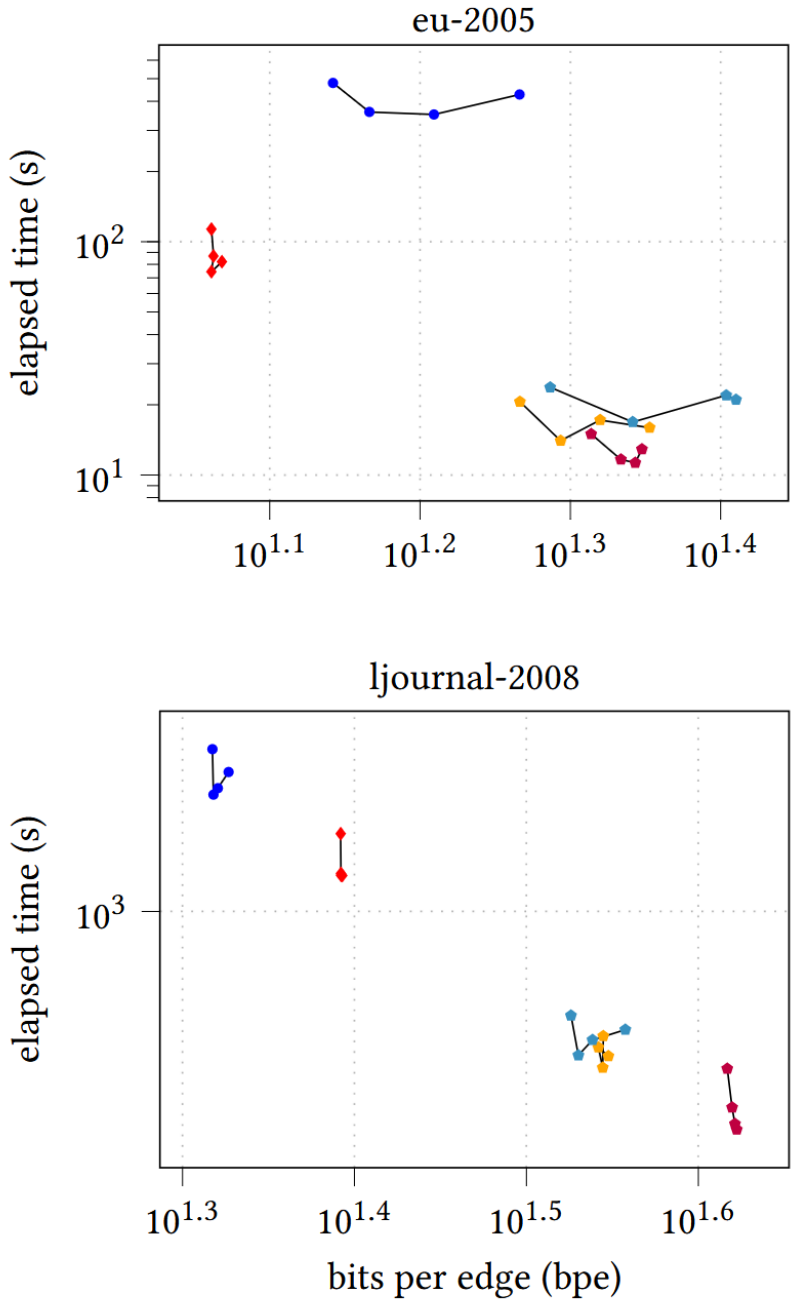




Spazio-tempo sul Raspberry Pi

5 grafici più piccoli. Fino a ≤8 thread.

- Andamento non monotono osservato quando si passa da 4 a 8 thread (in particolare per Zuckerli).
- Il k^2-tree a thread singolo supera tutte le configurazioni Zuckerli su Raspberry Pi
- La tendenza convessa meno pronunciata su Intel® suggerisce una diversa gestione delle risorse.



Conclusioni

Una selezione accurata delle rappresentazioni compresse consente di gestire grandi dataset su dispositivi con risorse limitate.

Una selezione attenta della rappresentazione compressa può abbattere il consumo energetico di uno o due ordini di grandezza su diverse piattaforme.

Il k^2-tree ha velocità simile ai compressori grammaticali ed efficiente in spazio quanto Zuckerli, con una degrado minimo del rapporto di compressione all'aumentare dei thread.

Grado ottimo di parallelismo può variare a seconda che si voglia ridurre al minimo l’energia o il tempo d’esecuzione.

L'aumento delle operazioni di cache L1 e L3 influisce sui cicli per istruzione

Futuri sviluppi

- Esaminare ulteriori formati di compressione lossless per matrici e vettori.
- Includere altre applicazioni oltre PageRank.
- Indagare ulteriormente i compromessi tra energia e tempo. → fornire linee guida per ingegneri del software per ridurre le emissioni di carbonio ed estendere la durata delle batterie.
- Studiare implementazioni a basso consumo energetico delle principali strutture di dati compresse (p.es., l'FM-index, le strutture rank e select, gli array di suffissi, le topologie succinte per alberi, ...)
- Ottimizzazione del ML: esplorare combinazioni di strategie di compressione *lossless* e *lossy* allo stato dell'arte

Trasparenza e riproducibilità

Il codice per riprodurre gli esperimenti è pubblicato su github:
https://github.com/acubeLab/green-lossless-spmv

Datasets da:
https://sparse.tamu.edu/LAW

GitHub

Dr. ric. Francesco Tosoni

Prof. Giovanni Manzini

Sig. Valerio Brunacci

Prof. Alessio De Angelis

Prof. Paolo Ferragina

Prof. Philip Bille

Coautori

- 4 università
- 3 gruppi di ricerca
- 2 aree di ricerca

Gruppi di ricerca

Visitate le nostre pagine per rimanere aggiornati sui nostri risultati!

A3 Lab

A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA
DIPARTIMENTO DI ECCELLENZA
MUR 2023/2027

DTU

Acube

EMERLab

AIgoLoG

Proposte di tesi

Prof. Paolo Ferragina

Software Heritage

- Archivio universale del codice sorgente
- Conserva, organizza e rende accessibile il software mondiale
- Utile per ricerca, industria e preservazione del patrimonio digitale
- Collaborazione con istituzioni, aziende e comunità open-source

Domande & risposte

Francesco Tosoni, dott. ric.
Assegnista di ricerca

Informatica
L.go B. Pontecorvo 3
56127 Pisa PI
Italia

francesco.tosoni@di.unipi.it
pages.di.unipi.it/tosoni