UART

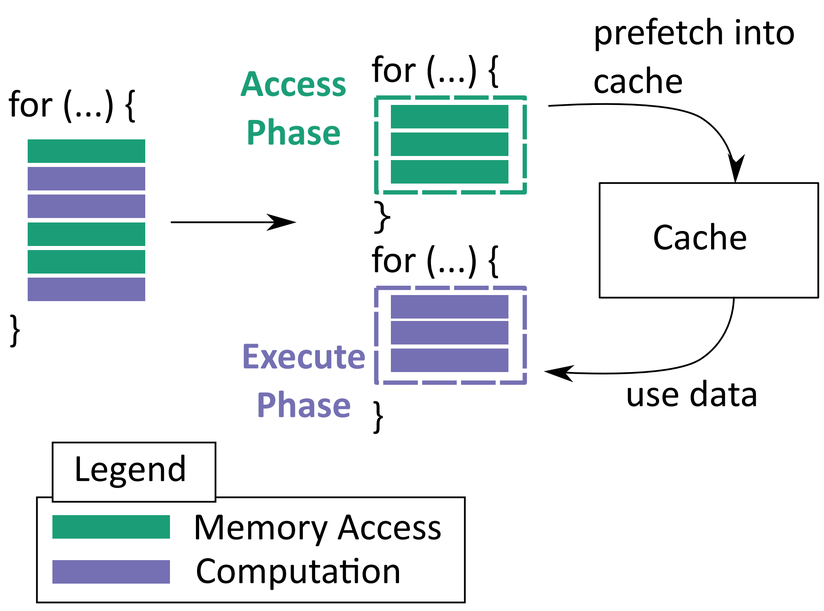
Uppsala Architecture Research Team

Software Decoupled Access-Execute

Kim-Anh Tran, Konstantinos Koukos,
Stefanos Kaxiras, Alexandra Jimborean

UPPSALA UNIVERSITET

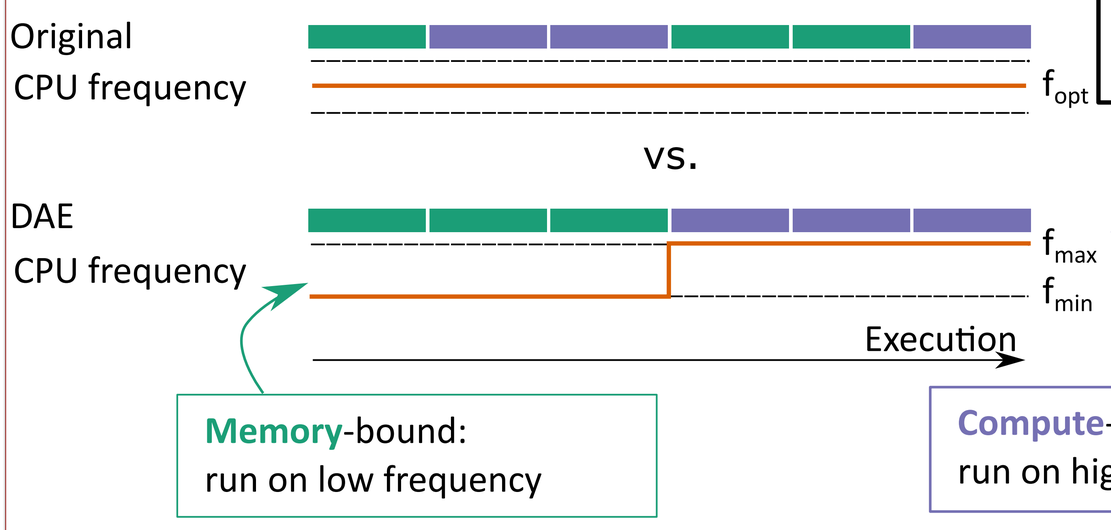
DAE = Decoupling Uses from Loads



Why DAE? DAE + Frequency Scaling Saves Energy

DAE helps to **save energy** by adjusting the frequency.

Original code: **compromise** frequency to balance between energy and performance.



How Can We Improve DAE?

Problem:
Hoisting *all loads*
Into an Access Phase
Is *infeasible:* address computation overhead diminishes benefits!

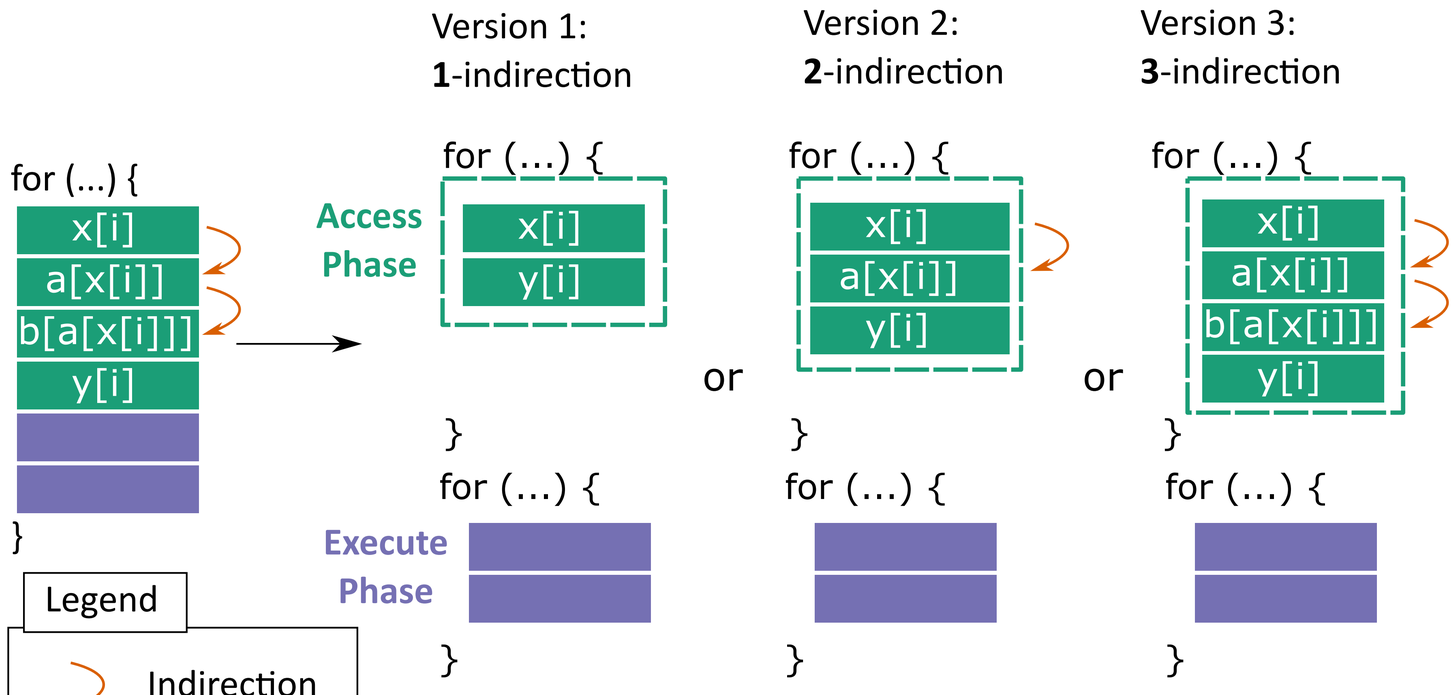
1 **Which** loads to select for prefetching?

Problem:
Address *recomputation* in the Execute Phase is *expensive*.

2 How to **avoid** address recomputation?

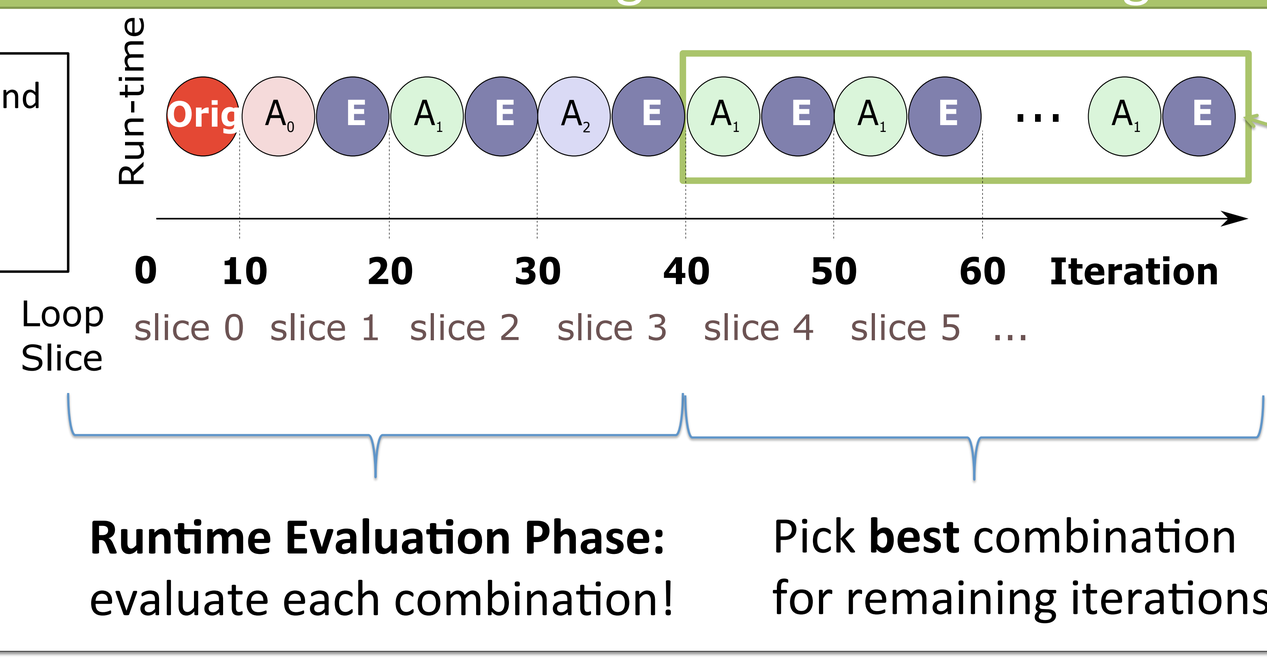
1 Selecting Loads: Creating Alternative Access Phases

Create **one** Access Phase **for each** level of indirection*:*



Evaluating Versions at Runtime and Selecting the Best Performing One

Run **each version** (Original and alternative Access-Execute versions) for a **couple of iterations**



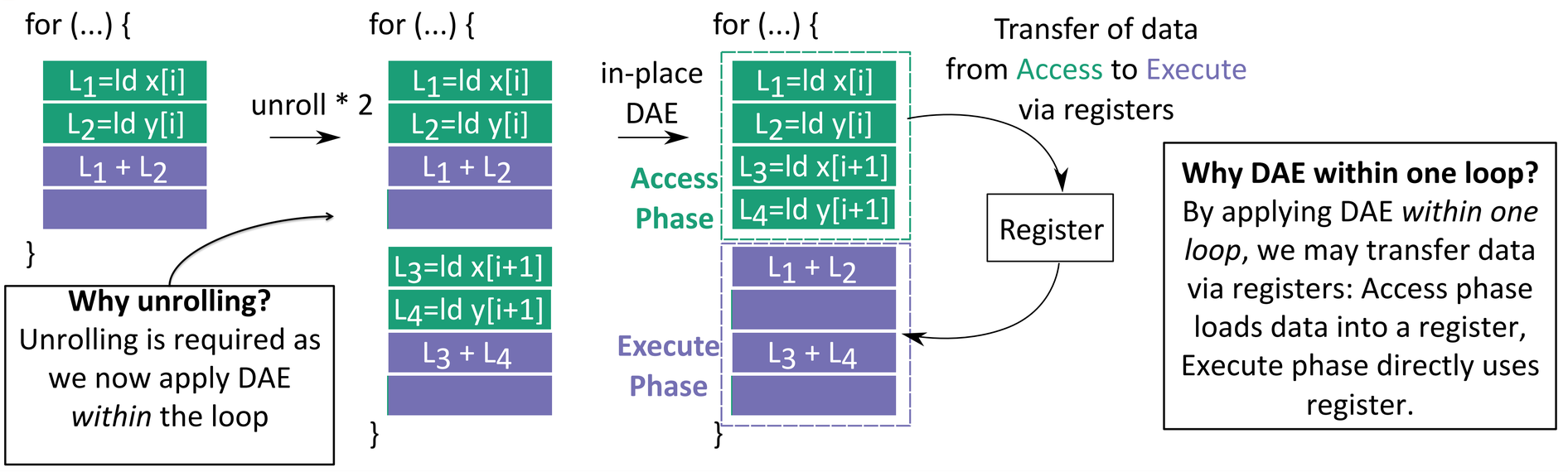
A_1 E was identified as the best performing version: settle on this combination!

Runtime Evaluation Phase: evaluate each combination!

Pick **best** combination for remaining iterations

Legend
- Orig — Original Code
- A_n — n-th Access Version
- E — Execute Phase (same for all A_n)

2 Overcoming Address Recomputation: Hoisting Loads to Access Phases



Why unrolling? Unrolling is required as we now apply DAE *within* the loop

Why DAE within one loop? By applying DAE *within one loop*, we may transfer data via registers: Access phase loads data into a register, Execute phase directly uses register.

Info

Reference:
A. Jimborean, K. Koukos, V. Spiliopoulos, D. Black-Schaffer, S. Kaxiras. Fix the code. Don't tweak the hardware: A new compiler approach to Voltage-Frequency scaling. In Proc. of CGO'14

Contact:
kim-anh.tran@it.uu.se