Software Decoupled Access-Execute
Kim-Anh Tran, Konstantinos Koukos, Stefanos Kaxiras, Alexandra Jimborean

DAE = Decoupling Uses from Loads

for (...) {
} for (...) {
} for (...) {
}

Access Phase
Cache

for (...) {
} for (...) {
}

Execute Phase

Legend
- Memory Access
- Computation

Why DAE? DAE + Frequency Scaling Saves Energy

DAE helps to save energy by adjusting the frequency.

Original CPU frequency

DAE CPU frequency

f_{opt}

f_{max}

f_{min}

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

Selecting Loads: Creating Alternative Access Phases

Create one Access Phase for each level of indirection:

1. Which loads to select for prefetching?
2. How to avoid address recomputation?

Evaluating Versions at Runtime and Selecting the Best Performing One

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

Run-time

0 10 20 30 40 50 60 Iteration

Loop Slice

Pick best combination for remaining iterations

Runtime Evaluation Phase:

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

Department of Information Technology, Uppsala University
http://it.uu.se/