Dynamic Memory Management Techniques for Optimization over Heterogeneous DRAM/NVM Systems

Manolis Katsaragakis*[+], Francky Catthoor[+], Dimitrios Soudris*

*Microprocessors and Digital Systems Laboratory, ECE, National Technical University of Athens, Greece

[+]Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

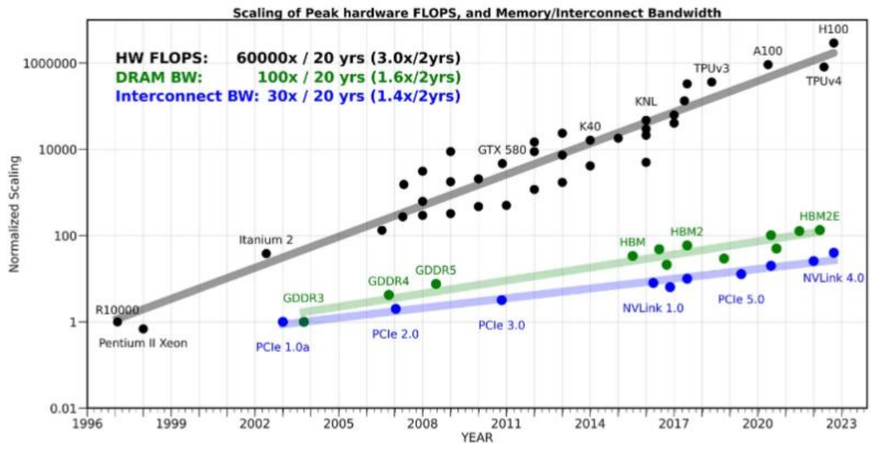
[+]emmanouil.katastaragakis@student.kuleuven.be, francky.catthoor@esat.kuleuven.be,

*{mkatsaragakis,dsoudris}@microlab.ntua.gr

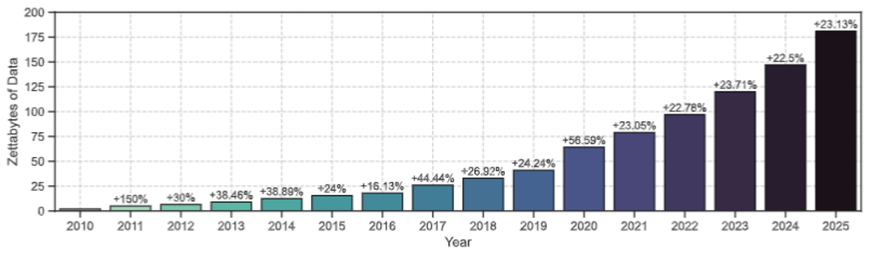
National Technical University of Athens

European Commission | Horizon 2020 European Union funding for Research & Innovation

CONVOLVE

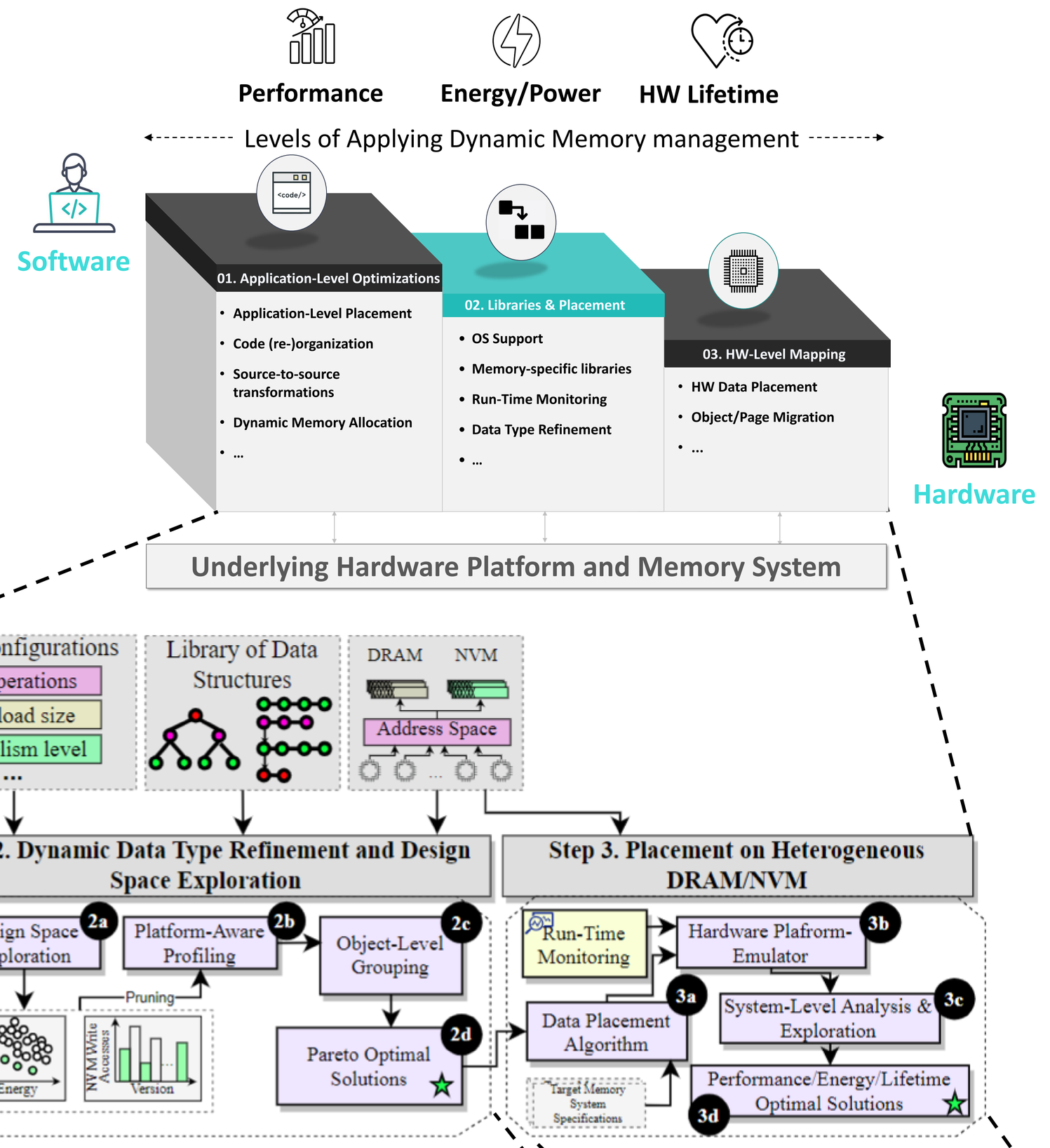


The Memory Wall Bottleneck[1]

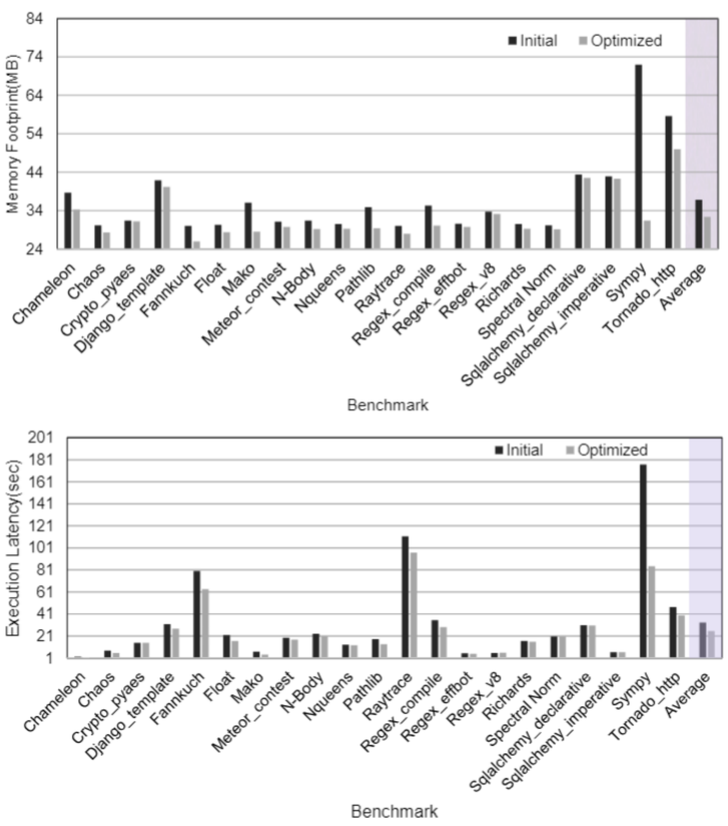


Zeta-bytes of data produced[2]

<u>PhD Thesis Objective(s):</u> Provide and end-to-end Dynamic Memory Management Flow for Performance, Energy and NVM-Lifetime Optimization



- Static to Dynamic Memory Management
- Set of Offline and Online Optimizations
- Source-to-Source Transformations
- Experiments:
  - *Pyperformance* Benchmark Suite
  - Raspberry pi4



<u>Key Outcome:</u> 64% Average Memory Footprint Reduction

<u>Key Outcome:</u> Side-effects on execution time (-51% avg) and energy consumption (-47.3% avg)

Library of Dynamic Data Types, Source Code Integration & Data Placement

Data Structure	Short Description
VEC	An array that can grow or shrink dynamically(Vector)
SLL	Single Linked List
SLLR	SLL with a pointer to the last accessed element
DLL	Double Linked List
DLLR	DLL with a pointer to the last accessed element
MAP-RB	An associative container(map in STL) that stores pairs of key-value in a Red-Black tree
MAP-AVL	A map that stores key-value pairs in an AVL tree

```
#include <Vector>
#include <map>
//Include our library of data implementations
#include <ddtr>

using namespace std;
class SamplePoint {...};

//Native data structure implementation
//vector<SamplePoint> v1;
//Replacing with library data structure
ddtr_vector<SamplePoint> v1;

//Native data structure implementation
//map<int, SamplePoint> mp1;
//Replacing with library data structure
ddtr_map<int, SamplePoint> mp1;

//Defining iteratator
map<int,SamplePoint>::iterator itr;

//Inserting element
mp1.insert(pair<int, SamplePoint>(1,new
    SamplePoint));
...
```

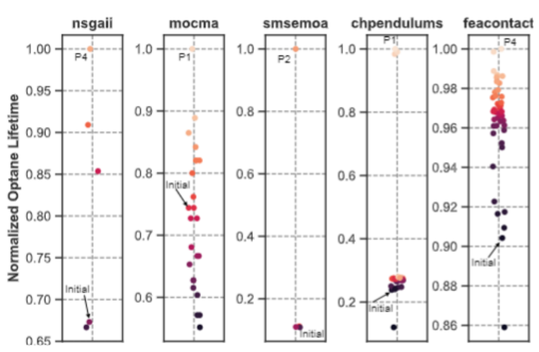
Applications:

- Databases (TPC-C, YCSB)
- ML-based (Shark-ML)
- Physics Engine (Chrono)

Target Hardware:

- DRAM/Intel Optane DCPM

Pruned Design Space and Pareto Optimal Solutions



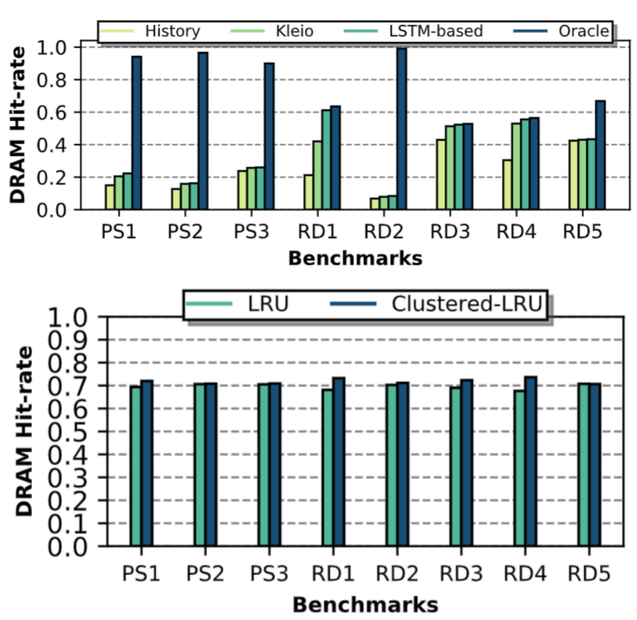
NVM Lifetime Projection

<u>Key Outcome:</u> up to 58.7% less execution time, up to 48.3% less energy consumption, up to 72.6% less NVM accesses

<u>Key Outcome:</u> 21.6% average extended lifetime of Optane DCPM

- Integration of LSTMs for page placement, enhanced with neighbor pages behavior
- Clustered-LRU replacement policy

Benchmark Suite	Benchmark	ID	Application Domain	Page Write Accesses	Page Read Accesses
PARSEC [1]	Blackscholes	PS1	Finance	4881	66216
	Bodytrack	PS2	Computer Vision	32411	122371
	Streamcluster	PS3	Data Mining	165495	31915
Rodinia 3.1 [3]	Backprop	RD1	Machine Learning	286869	302419
	Bplustree	RD2	Graph Theory	108274	346723
	Hotspot	RD3	Physics Simulation	5186	197473
	K-means	RD4	Data Mining	187364	356603
	Lud	RD5	Linear Algebra	370481	329342



<u>Key Outcome:</u> 20.2% less energy consumption on average

<u>Key Outcome:</u> *clustered-LRU* provides up to 8.1% further optimization compared to LRU

1. https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8
2. https://www.statista.com/statistics/871513/worldwide-data-created/