University of BRISTOL

Garbage Collection for Edge Computing

Andrés Amaya García
David May
Ed Nutting

bristol.ac.uk

Introduction

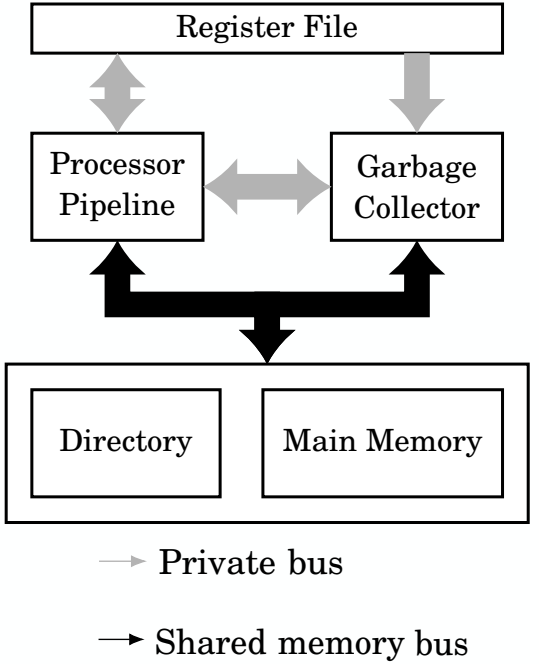
- Modern programming languages provide a high-level view of data and control
  - Python, C#, JavaScript, etc...
  - Productivity
  - Trust
- Modern languages are not used in embedded systems because software garbage collectors:
  - Impose high performance and memory overheads
  - Cannot be used in real-time systems

Objectives

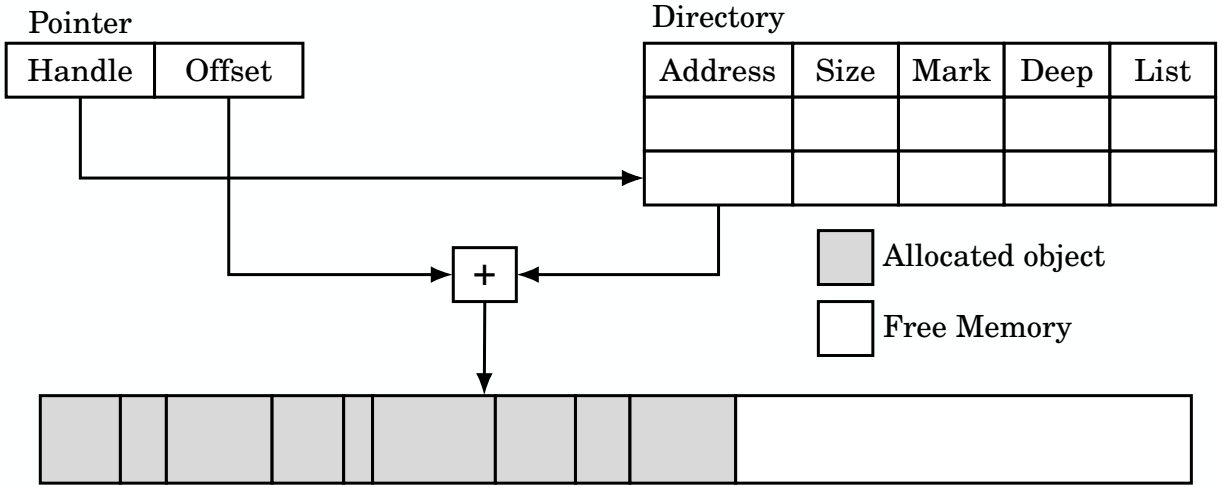
- Efficiently support modern languages in embedded systems – think small IoT edge devices e.g. ARM Cortex-M processors.
- Investigate hardware –instead of software– collectors
  - High run-time performance
  - Low memory requirements
  - Capable of hard real-time

Integrated Hardware Garbage Collection (IHGC)

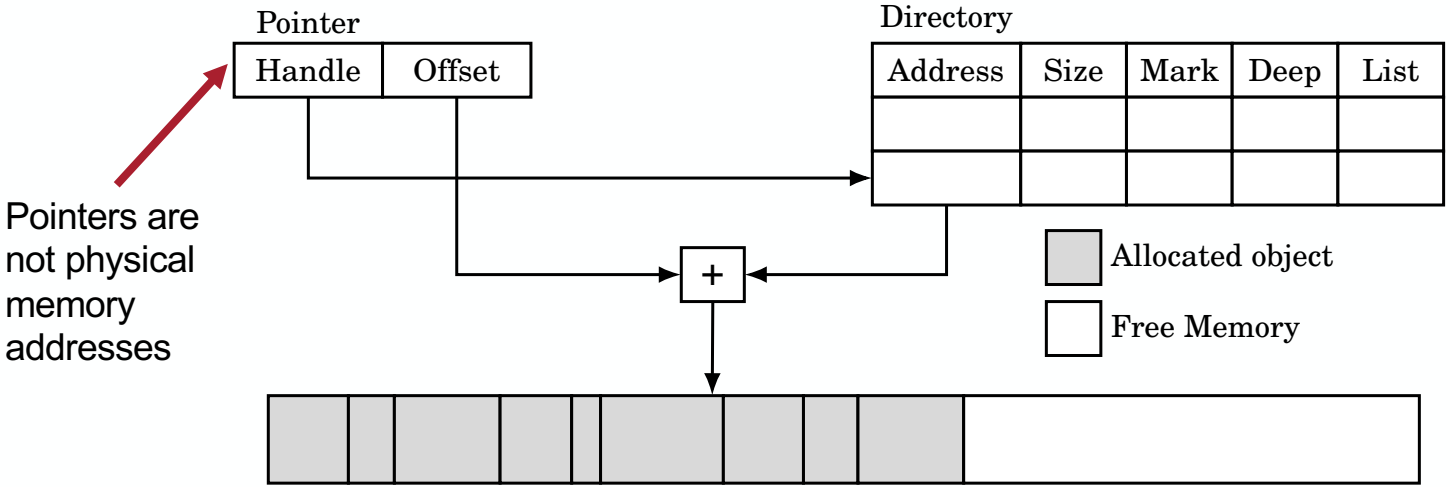
- Exact collector
  - Every word has a 1-bit tag: pointer vs data
- Indirection through handles
  - Directory memory contains object's metadata
  - Tightly integrated with processor
- Hardware state machine for collector
  - Each state transition performed in one memory cycle
  - Collector operates when the processor is not accessing memory
- Allocations via an instruction



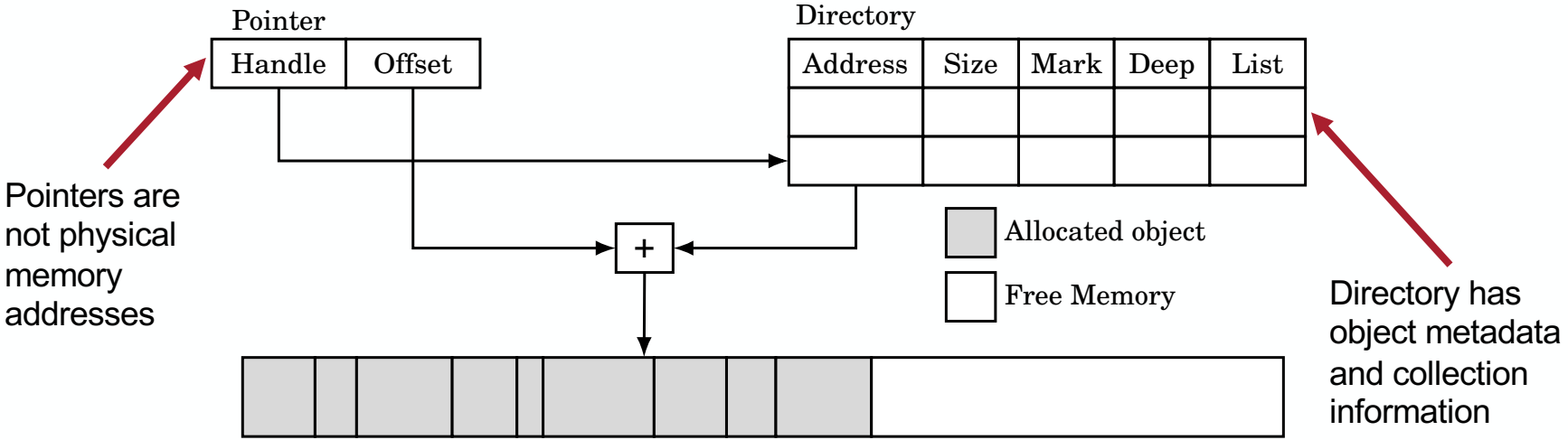
IHGC – Indirection Through Handles



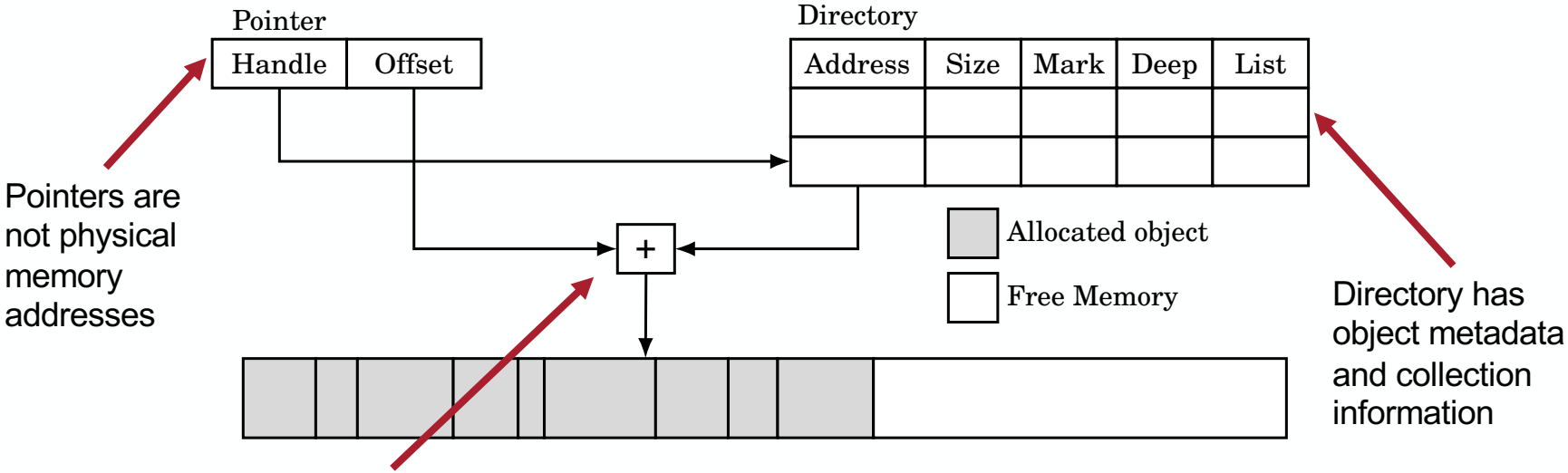
IHGC – Indirection Through Handles



IHGC – Indirection Through Handles



IHGC – Indirection Through Handles

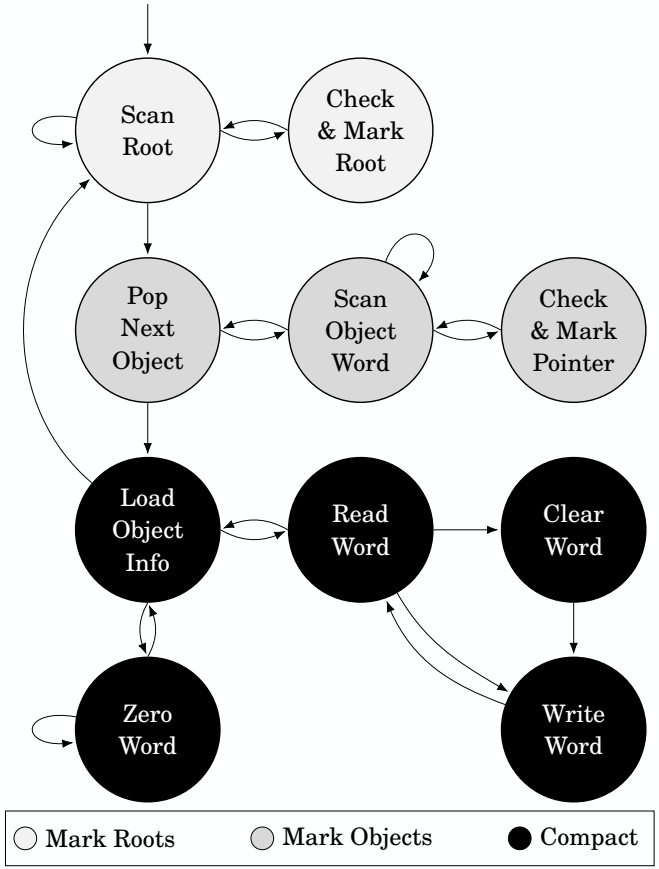


Memory access in two steps:

1. Calculate main memory address
2. Access main memory

IHGC – Garbage Collection State Machine

- Mark-Compact collector
- Roots are pointers in the registers
- Each state transition is performed in one memory cycle



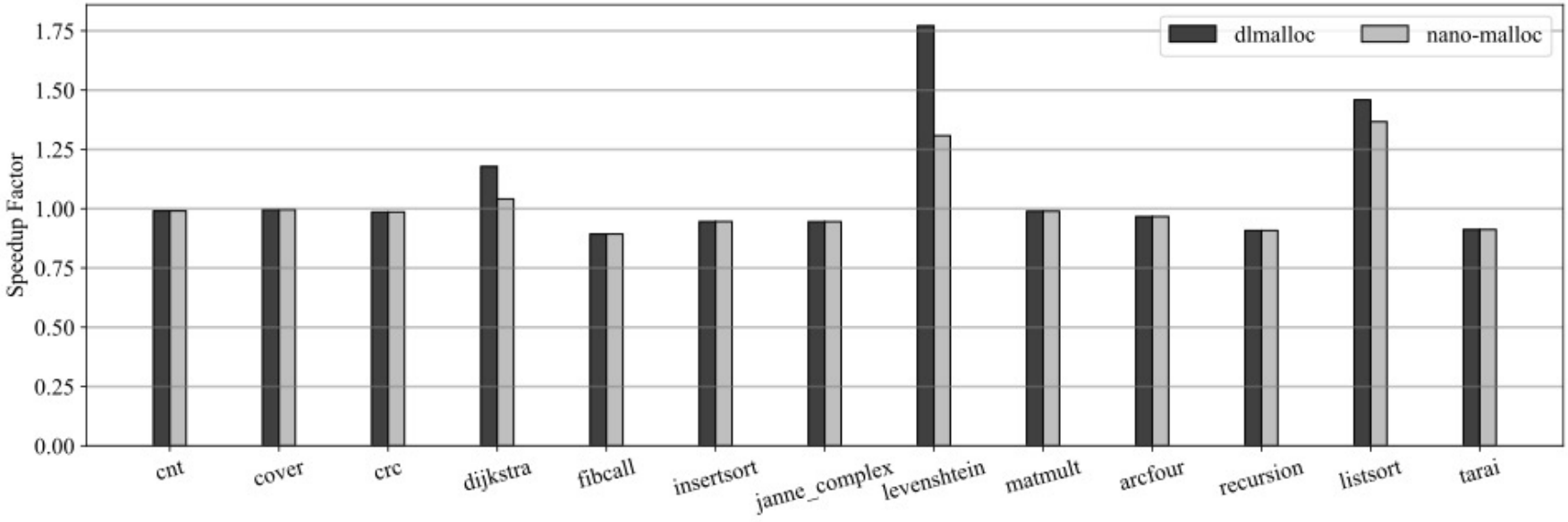
IHGC – Coordination

- Mark stage
  - Pointers loaded from main memory are processed for marking
  - Conceptually similar to a traditional *read barrier*
- Compact stage
  - Redirect memory access to the correct location when the object accessed is being compacted
- Transparent from the program's point of view
- Does not incur pauses or performance penalties

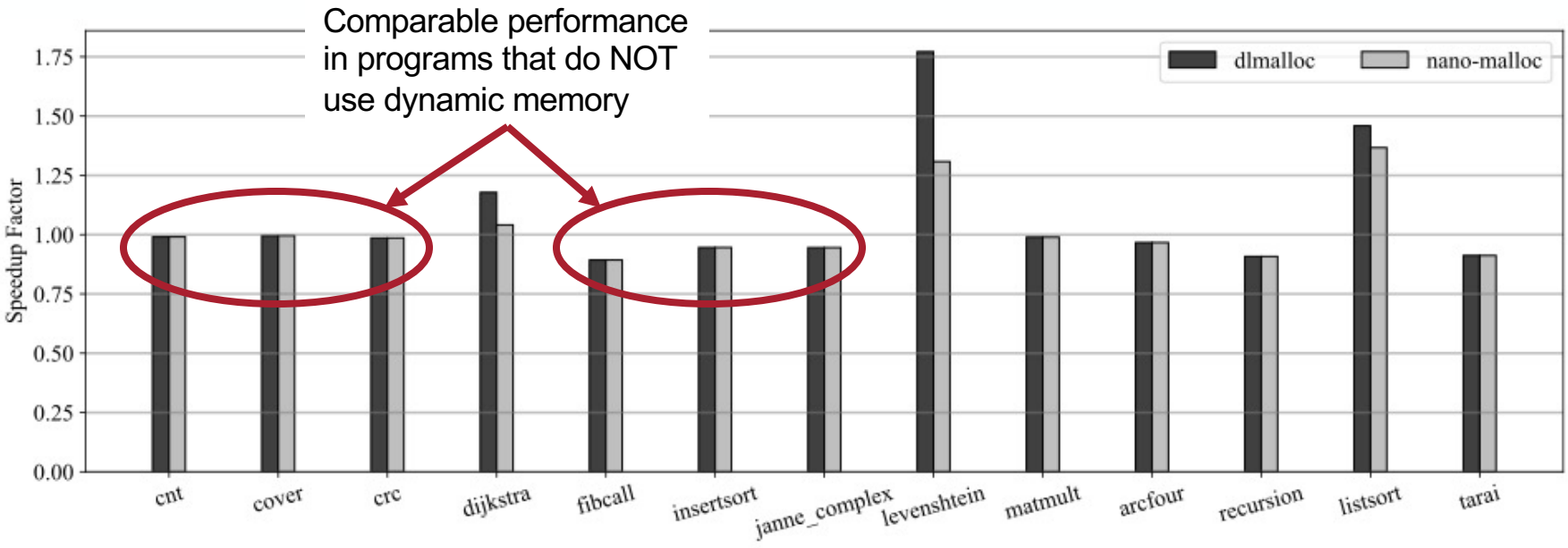
Experimental Setup

- Modelled an ARM Cortex-M0 alongside the IHGC
- Benchmarks:
  - BEEBS
  - Python 3 scripts running on MicroPython
- LLVM compiler

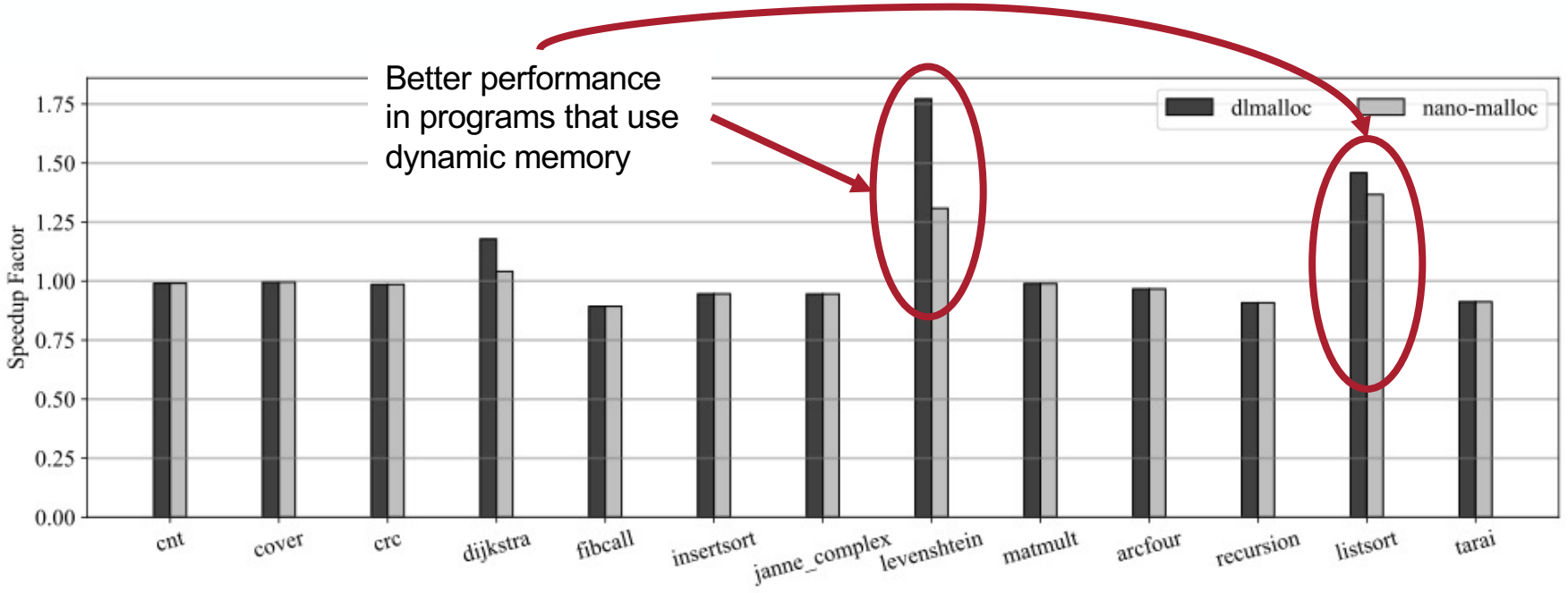
Results – BEEBS



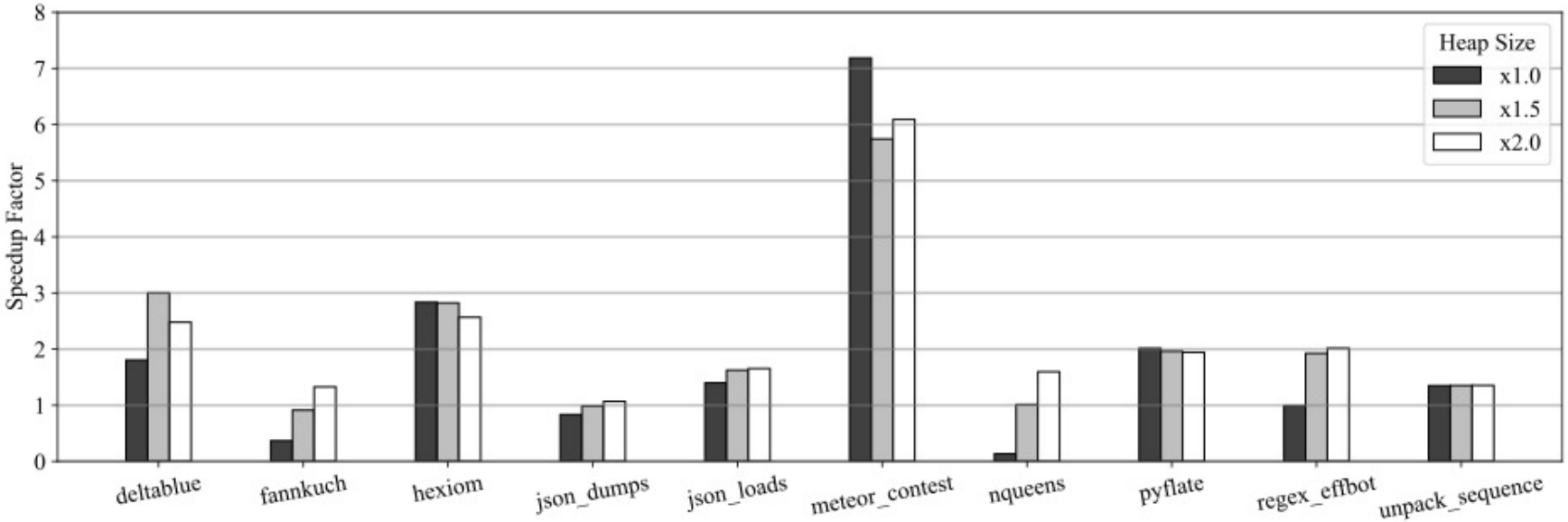
Results – BEEBS



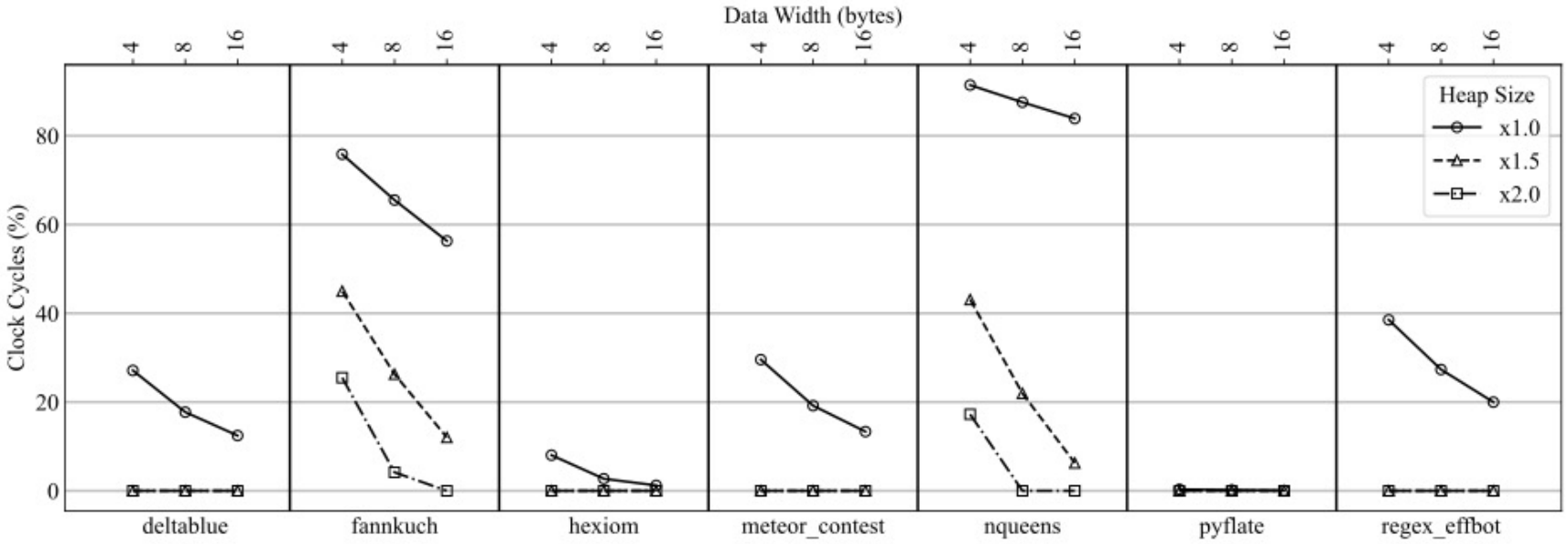
Results – BEEBS



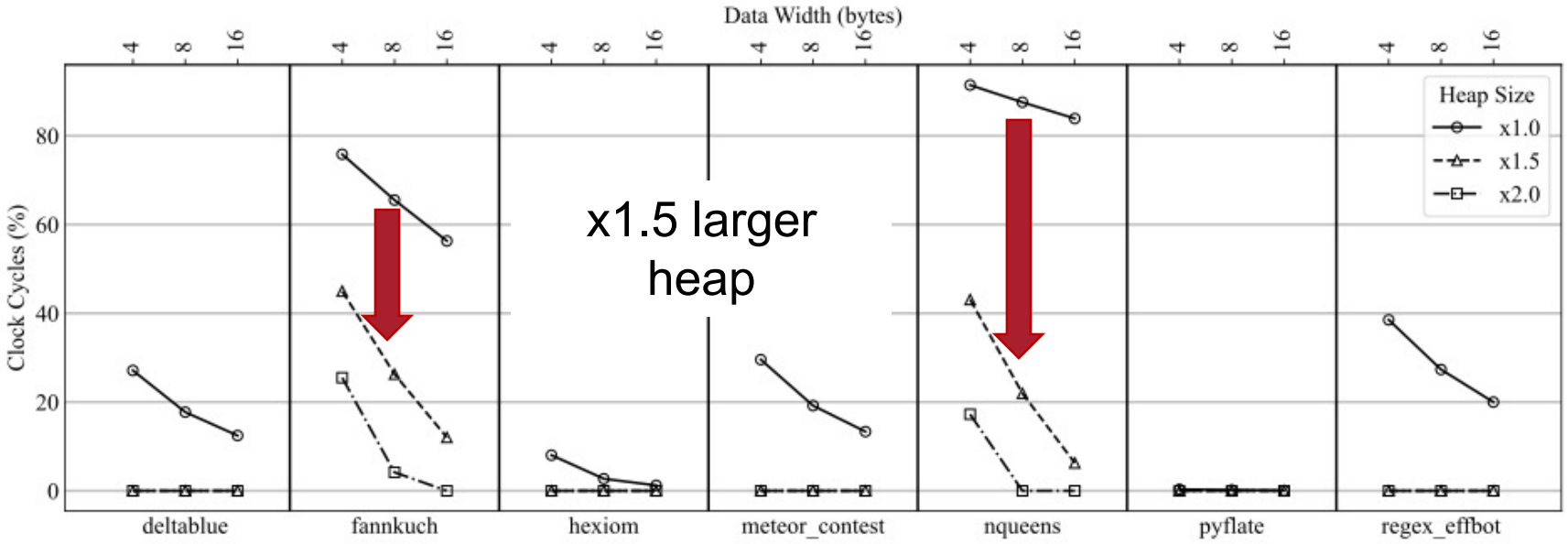
Results – MicroPython



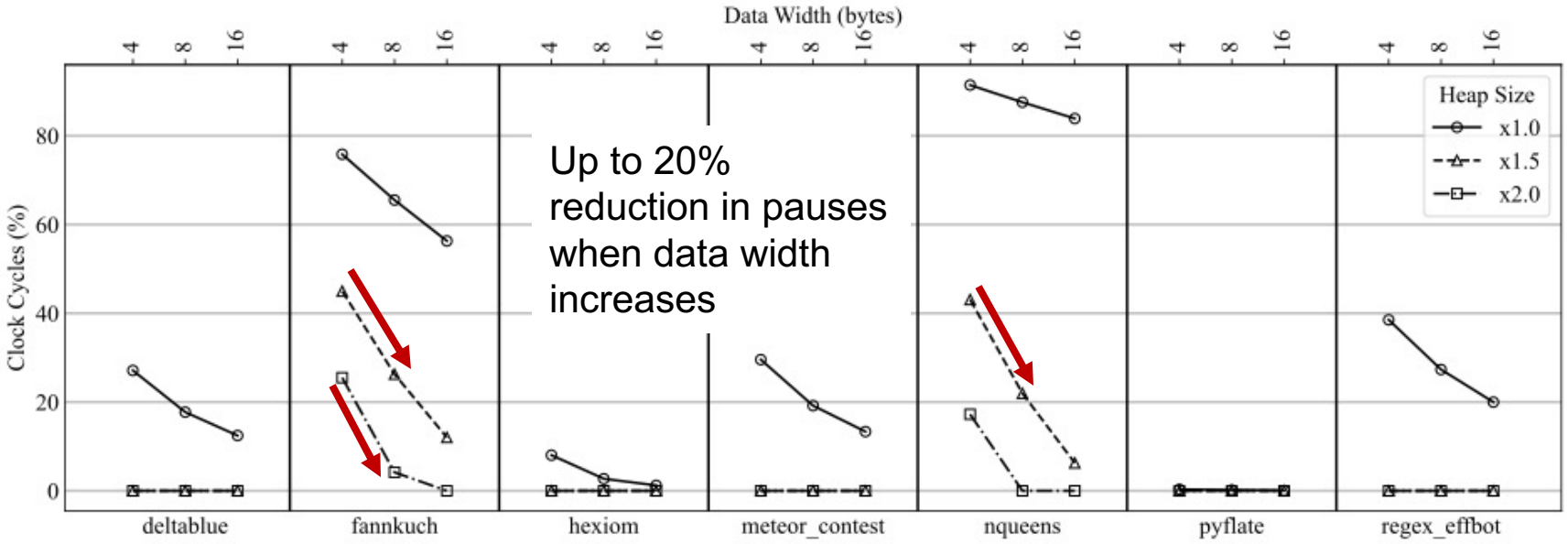
Results – Pauses



Results – Pauses



Results – Pauses



Conclusion

- IHGC splits collection work at the granularity of a memory cycle
- Collection worked performed when the processor is not using the memory
- Comparable or better performance when running C programs
- Python scripts run 1.5-7 faster in IHGC system
- Hard real-time

University of BRISTOL

Thank You!

Questions?

Andrés Amaya García
andres.amayagarcia@bristol.ac.uk
sourcecodeartisan.com

bristol.ac.uk