Cold Object Identification, Sequestration and Revitalization

A. Taware, K. B. Kent & G. W. Dueck
University of New Brunswick, Faculty of Computer Science
Charlie Gracie
IBM Canada
{ataware, gdueck, ken}@unb.ca charlie_gracie@ca.ibm.com

Cold Objects

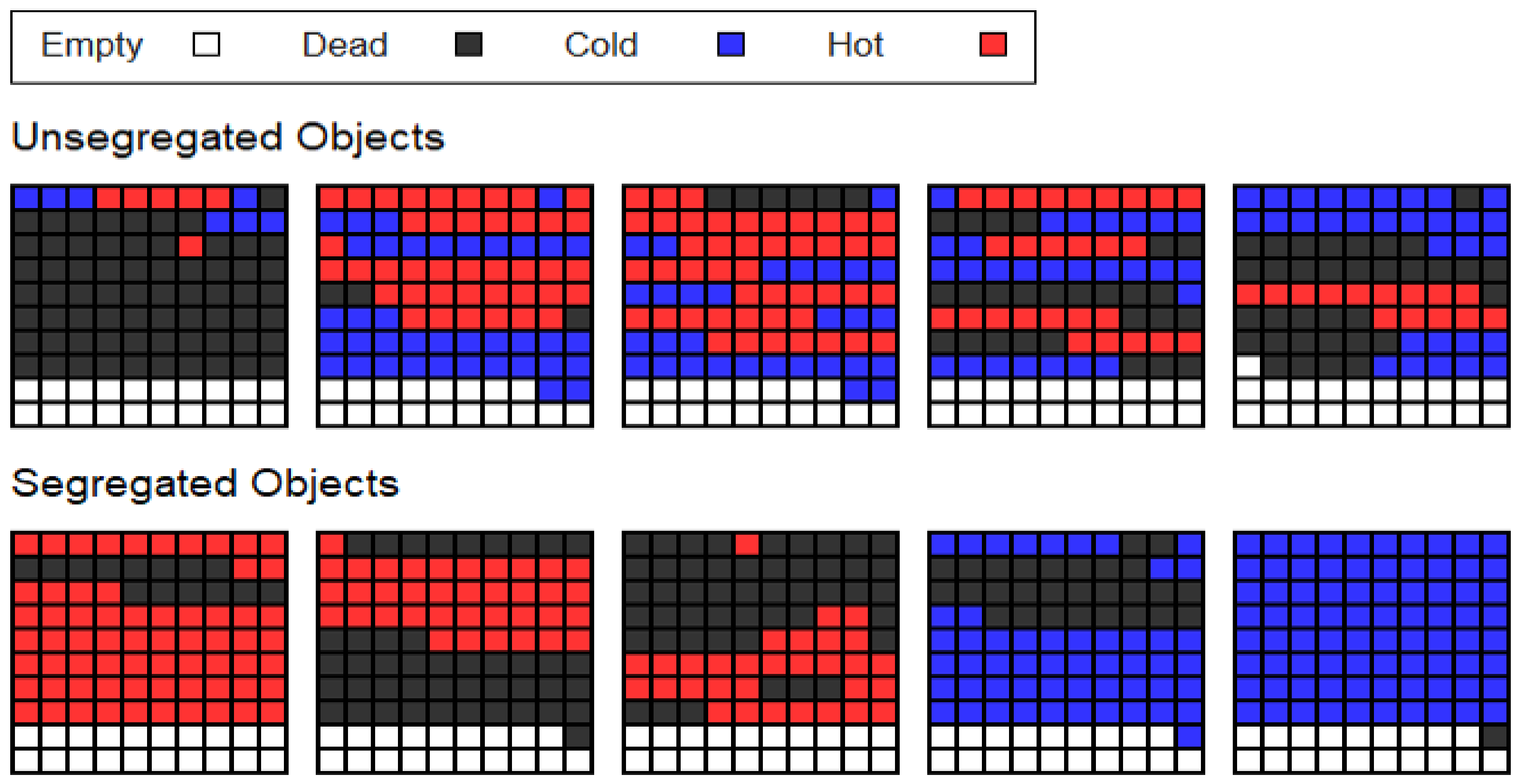
- Cold objects are alive and infrequently referenced
- They are largely unnecessary GC overhead and pollute the cache
- These issues can be tackled by segregating cold objects to a separate memory area e.g. Intel 3D Xpoint NVRAM.

Balanced Garbage Collection Policy

- Splits the heap into thousands of independently collectable regions
- Allocation context provides an abstraction to manage hot and cold regions as separate entities.
- Cold regions are backed by NVRAM with help of mmap operation.

Cold Object Segregation

- Diagram below depicts object segregation based on temperature.
- Backing the cold regions leaves more hot heap for applications.
- GC for hot regions can be prioritized over cold ones.
- Identifying cold objects is a costly operation to do at runtime.

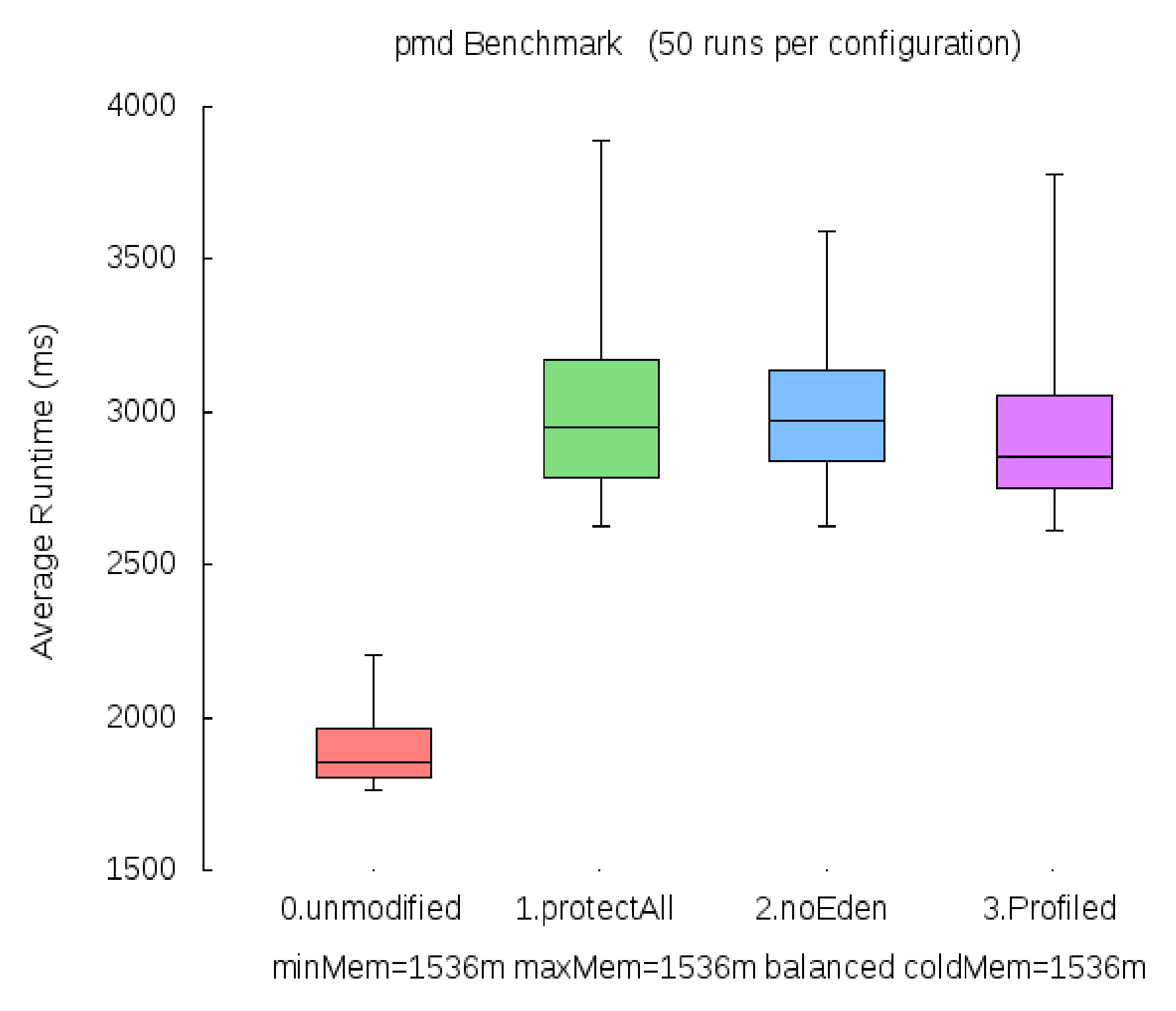


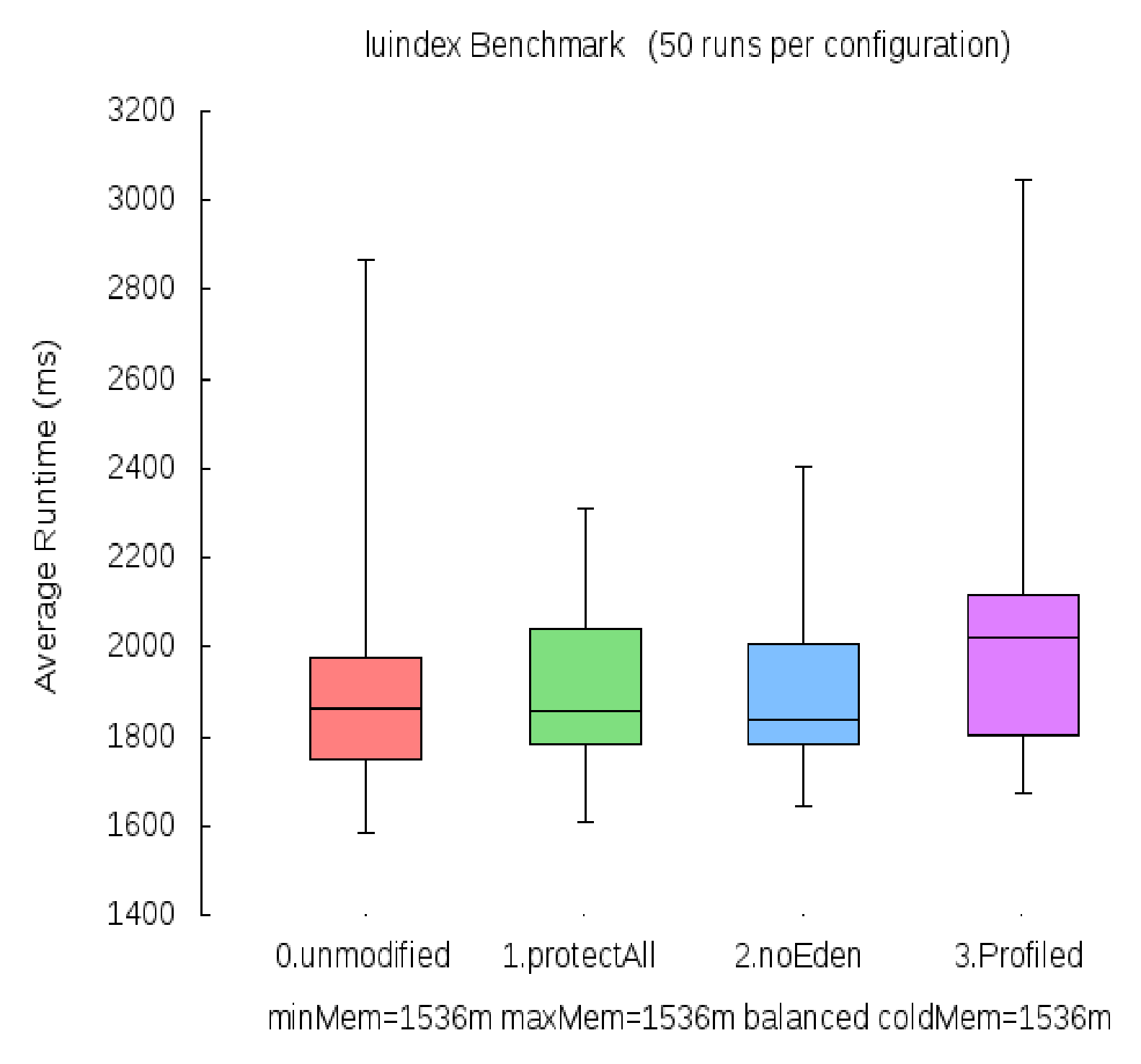
Application Profiling

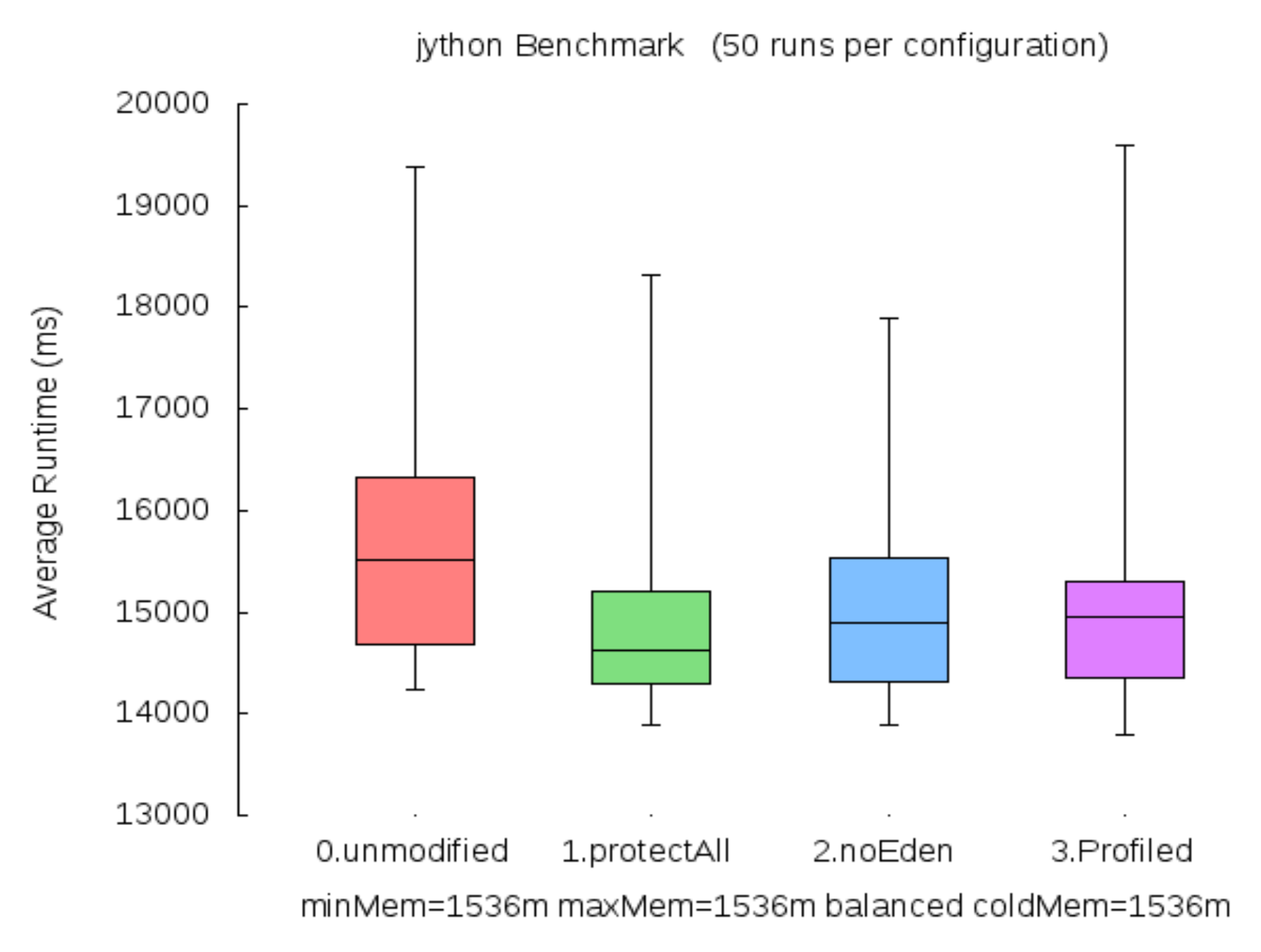
- During Partial Garbage Collection (PGC) cycle, a stop-the-world operation, the profiler collects the following information:
  - Popularity: How often an object references its children
  - Temperature: How often this object is referred to by other objects
- Data across all PGC runs is summarized to get list of cold classes.

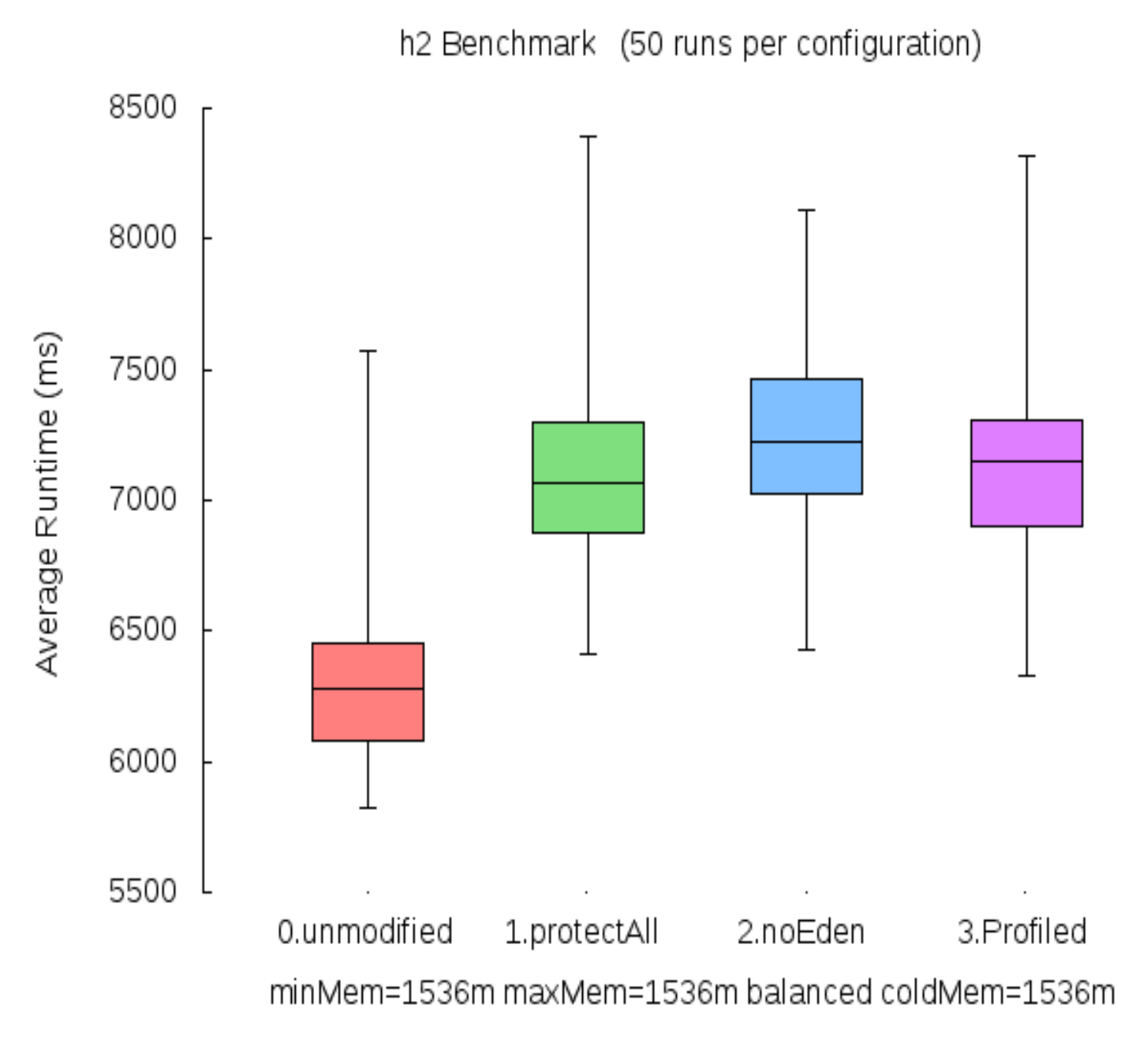
Page Protection and Profiling

- OS paging is used as access barrier to identify hot pages.
- Profiled data is further used to segregate hot/cold objects.
- Page protection is enabled only for non-Eden (older) regions, which more likely host cold objects.
- Results for DaCapo benchmarks depict both benefits and overhead.
- Runs are taken for unmodified JVM, protected pages solution, protection for non-Eden regions and profiling applied to protected non-Eden regions.









Conclusion

- Application profiling helped in case of pmd, but added overhead for luindex benchmark.
- Application behavior may not be same across runs, which requires updating this data at runtime without affecting performance.

Future Work

- Profiled data is kept in a hash-map. Encoding this information in object headers will reduce search time.
- Stack walking can be used to update profiled data at runtime. This should be done for oldest regions only for short intervals so as not to affect the performance.