FlexHeap: Dynamic DRAM Partitioning Between Managed Heap and Page Cache

Iacovos G. Kolokasis
kolokasis@ics.forth.gr

Shoaib Akram
shoaib.akram@anu.edu.au

Foivos Zakkak
fzakkak@redhat.com

Polyvios Pratikakis
polyvios@ics.forth.gr

Angelos Bilas
bilas@ics.forth.gr
Analytics frameworks use managed runtimes

To process **large amounts of data** they need **large heaps**

DRAM in a single server **scales slower** than data growth!
- Increase power consumption and heat dissipation
- DRAM capacity is declining

Analytics frameworks extend the managed heap (beyond DRAM) using
- Fast block-addressable storage devices (e.g., NVMe SSD)
- Byte-addressable non-volatile memory (NVM)
- Remote memory
Trade-offs of organization of hybrid managed heaps

- Caching hides heterogeneity of the tiers
- GC scans over the slow tier → High page swappings
Trade-offs of organization of hybrid managed heaps

Managed heaps with caching

Fast tier
Page Cache
A B C D
Slow tier
Managed Heap
A B C D E F G H I

- Caching hides heterogeneity of the tiers
- GC scans over the slow tier → High page swappings

Managed heaps with tiering

Fast tier
Young gen.
A B C
Old gen. (hot)
D E F
Slow tier
Old gen. (cold)
G H I

- Reduced page swappings
- High object reference adjustment cost
Trade-offs of organization of hybrid managed heaps

Managed heaps with caching

- Fast tier
  - Page Cache
  - A B C D

- Slow tier
  - Managed Heap
  - A B C D E F G H I

- Caching hides heterogeneity of the tiers
- GC scans and compactions over the slow tier

Managed heaps with tiering and caching

- Fast tier
  - Fast heap
  - A B C
  - Page cache
  - D E

- Slow tier
  - Slow heap
  - D E F G H I

- No object reference adjustment cost
- No GC scans and compactions to the slow tier

Managed heaps with tiering

- Fast tier
  - Young gen. (hot)
  - A B C
  - Old gen. (hot)
  - D E F

- Slow tier
  - Old gen. (cold)
  - G H I

- Reduces page swappings
- High object reference adjustment cost

Merge the benefits from both worlds!
Static division of DRAM between fast heap and page cache

- **Problem 1:** Requiring hand tuning configuration
  - Impractical in real-life deployments
  - Application and dataset change frequently

- **Problem 2:** Changing application behavior
  - Different memory requirements at different periods
Shortcomings of static division of DRAM in TeraHeap

- Applications have different phases
- Demand space for H1
  - Generate large amount of objects
  - High memory pressure $\rightarrow$ High GC
- Demand space for page cache
  - Heavily access objects in H2
  - High I/O traffic
- Dynamic division of DRAM is essential!
Outline

- Motivation

- FlexHeap design
  - Considering GC and I/O overheads
  - Repartitioning DRAM dynamically
  - Enhance responsiveness in application behavior changing

- Evaluation

- Conclusions
FlexHeap

- Dynamically division of DRAM between H1 and I/O cache for slow heap
  - Reduce memory pressure
  - Reduce I/O traffic

- Transparent mechanism
  - No application or OS modifications

- Adapt to application with dynamic changing behavior

- Makes practical the fast and slow heap approach
Considering GC and I/O overheads

- FlexHeap divides its execution
  - Sampling intervals between minor GC cycles

- I/O cost in terms of CPU iowait time

- For the GC cost FlexHeap estimate the next major GC cycle pause time

\[
F_{i-1} = \frac{FreeSpace}{SizeH1} 
\]  

\[
NetGCPauseTime = P \cdot (1 - F_{i-1}) 
\]  

\[
TimeToGC = \frac{F_i \cdot T_{i-1}}{F_{i-1}} 
\]  

\[
GCTime = \frac{NetGCPauseTime}{TimeToGC} \cdot T_{interval} 
\]
Repartitioning DRAM dynamically

- FlexHeap compares GC and I/O every minor GC

- Possible actions:
  - Increase the size of H1 (GrowH1)
  - Move objects to H2 (MoveH2)
  - Shrinking H1 to grow page cache (ShrinkH1)

- OS moves memory between H1 and page cache
  - Delay in observing the resizing action impact

- FlexHeap stops making decisions until their effect occurs
Enhance responsiveness in application behavior changes

- FlexHeap follows multihop decision paths
  - Reduce responsiveness

- Add new FSM transitions
  - Allows FlexHeap to jump to certain states
Testbed

- We implement FlexHeap on top of TeraHeap
  - TeraHeap uses Parallel Scavenge garbage collector
  - OpenJDK 17 and OpenJDK8

- We use one server with 2 TB NVMe SSD and 256 GB DRAM

- Real world application
  - Spark with Spark benchmark suite
  - Giraph with Graphalytics benchmark suite
  - Neo4j with Graphalytics benchmark suite

- Limit DRAM capacity with cgroups
### Static vs dynamic memory adjustment

- The performance gains range from 5% (Spark-LgR) to 70% (Giraph-CDLP).
- FlexHeap improves performance between 3% and 73% (13 out of 18 workloads).
- Reduction of GC and I/O cost up to 80%.
Performance with limited DRAM

- FlexHeap reduces DRAM capacity demands between $1.3\times$ (BFS) and $1.6\times$ (SSSP)
- Acceptable performance degradation ranging from $1.2\times$ (BFS) to $1.8\times$ (PR)
Key Takeaway

- Hybrid heaps setups exhibit dynamic variation in memory requirements
- Size of fast heap dominates GC cost
- Size of page cache dominates I/O cost for accessing objects in the slow heap
- FlexHeap dynamically divides a fixed DRAM budget between
  - Fast heap
  - I/O page cache
- FlexHeap adapts to the behavior of real-world big data analytics frameworks
  - Improves performance up to 73% compared to static approaches
FlexHeap: Dynamic DRAM Partitioning Between Managed Heap and Page Cache

We thankfully acknowledge the support of the European Commission projects AERO (GA No 10048318) Iacovos G. Kolokasis is supported by the Meta Research PhD Fellowship (2022 – 2024)

kolokasis@ics.forth.gr