NG2C - N-Generational GC for Big Data Memory Management

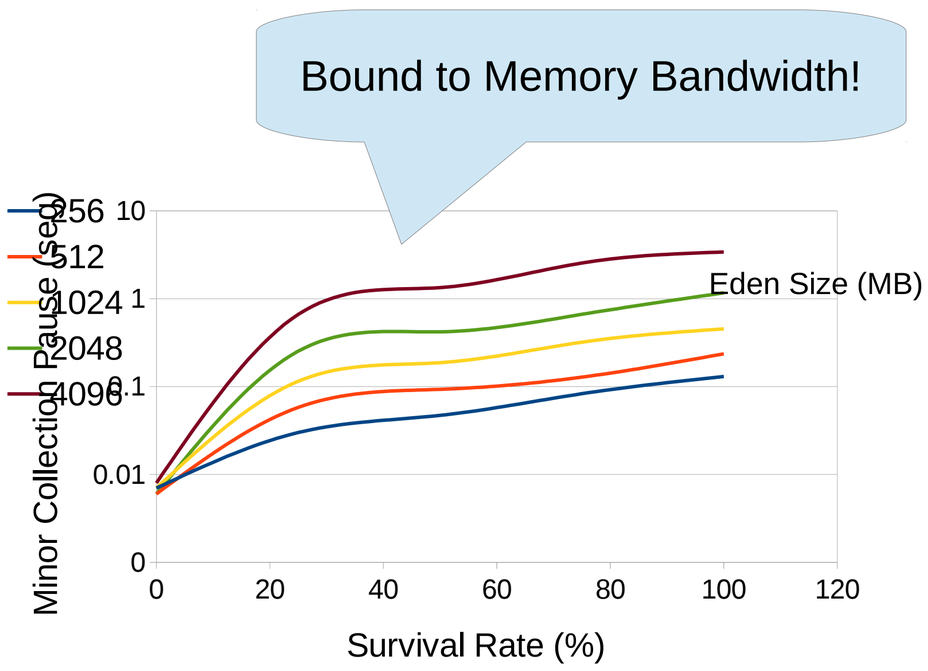
Rodrigo Bruno, Paulo Ferreira – rodrigo.bruno@tecnico.ulisboa.pt, paulo.ferreira@inesc-id.pt

1. Problem

Big Data platforms use lots of **memory to enable fast data access**.

Most of this data (eg. caches or processing queues) stays in memory for some time, where they get **copied several times** until reaching the **old generation**.

Applications experience **severe pauses** because **not all objects die young!**



2. Solution

Avoid copying objects by **allocating directly** in specific **generations** according to their **estimate life cycle**.

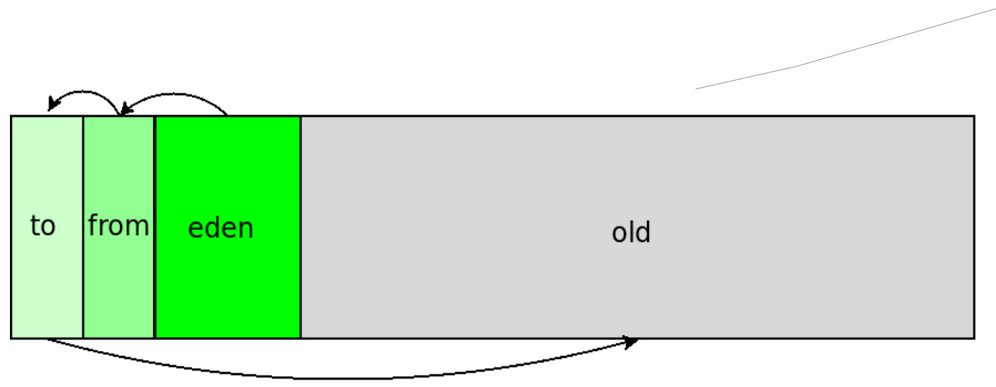
Each generation contains objects that will be mostly dead by the same time (this is done with the help of the programmer).

This prevents **unnecessary copying** of objects and **improves GC** efficiency.

3. Other Approaches

- **Complex GC tunning** to match the application allocation rate with the heap size;
- Implement complex memory management **strategies to control object allocation**;
- Resort to **Off-heap** memory.

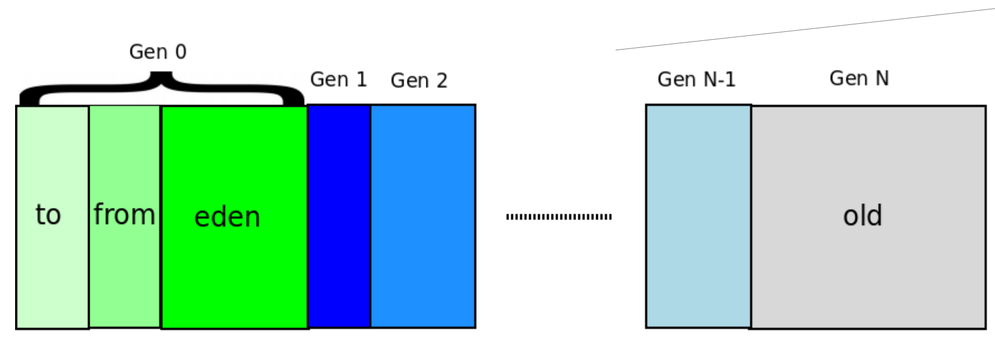
4. 2-Gen vs N-Gen Heap Layout



2-Gen(most common) Heap Layout

✗ Objects allocated in the Eden are copied several times;

✗ Old generation is hard to collect, potentially leading to full GCs;



N-Generational Heap Layout

✔ Objects allocated in specific generation, near other objects with the same life cycle.

✔ Heap is organized according to object life cycle. Easy to collect.

5. Code Sample

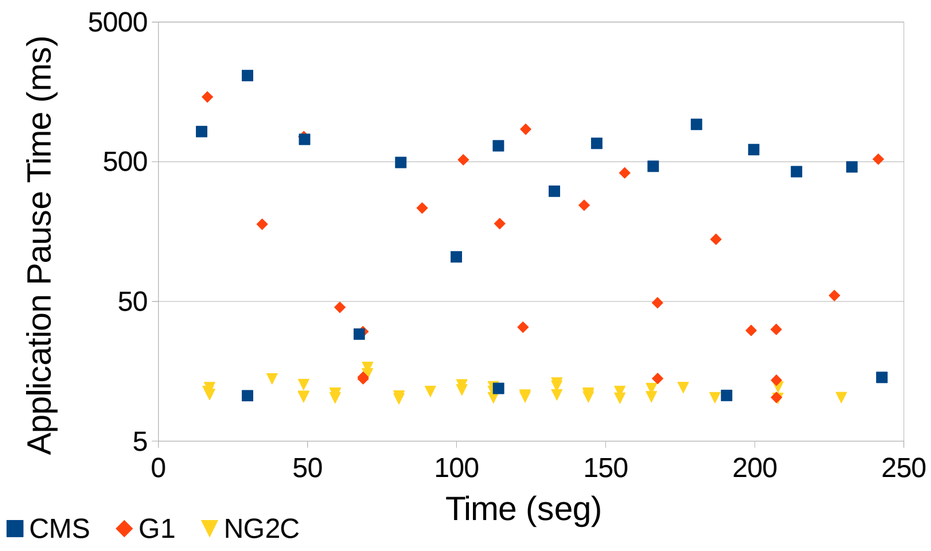
```
public void runTask() {
    System.newGen();
    while (running) {
        DataChunk data = new @Gen DataChunk();
        initializeData(data);
        doComplexProcessing(data);
    }
    System.collectGen();
}
```

Creates a new generation.

Annotation for allocating object in specific generation (other than Eden).

Creates a new epoch in the current generation. Memory previously allocated in the current generation is now ready to be collected.

6. Results



- 4 threads processing tasks with 1 GB of data;
- Working set of 4 Gbs;
- CMS and G1 with 8 GB young gen;
- NG2C works fine with 1GB. Heap size of 12 GB.

Application pauses severely reduced!

Percentiles	50	75	90	99	99.9	99.99
CMS (ms)	461	670	853	1873	2048	2065
G1 (ms)	94	241	588	838	853	854
NG2C (ms)	11	12	14	16	16	16

For more information, please contact us or visit our website at www.gsd.inesc-id.pt/~rbruno