JVM Footprint reduction for Edge Devices

Alexandra Tsvetkova, Bojian Zheng, Marius Pirvu, Gennady Pekhimenko
Course: CSC2228 Advanced Topics in Mobile and Pervasive Computing: Edge Computing
Fall 2019

***Abstract*— This project is aimed at studying opportunities memory compression can give at the Edge environment. Using Raspberry Pi 3 we are going to get real life data from Java applications to evaluate compression ratio and performance overhead of Java applications on edge devices.**

I. Project description

A. Original project

Java Virtual Machine (JVM) is an engine that provides runtime environment to execute java applications in a safe and controlled environment, managing not only instruction execution but also memory allocation and usage. To store and keep track of its objects JVM uses heap of limited size. This could become a severe bottleneck for execution of large application loads, especially in cloud or datacenter ecosystems where low memory consumption could allow for higher execution node occupancy.

At the same time, Java objects could be compressed with a high degree of efficiency to allow for the reduced memory usage. In our experiments we compressed the heap using gzip algorithm and achieved a compression ratio of 7.19. Of course, any potential memory optimizations are significantly less attractive if the execution overhead is too high.

We propose an implementation where compression will be done within the garbage collector (GC). We add the compression stage at the end of garbage collection. Most of the overhead here can be avoided by executing compression in a parallel thread, together with the garbage collector. We are planning to perform compression of the tenured area (i.e. only for objects which are present in memory for reasonably long amount of time). Another avenue for optimization is to perform compression only on cold objects - the challenge of that approach lies in discovering the cold objects in the first place.

Another crucial factor is memory decompression latency, since it can have a severe impact on memory access performance. To address this issue we aim to implement a low latency decompression algorithm. The decompression code will be injected within the bytecode of the user application.

B. Edge computing

Typically, edge devices are resource constrained[1]. Compressing memory on the device can help us trade off compute resources for memory. Unlike desktop machines, both these resources in the Raspberry Pi are very limited. It will be interesting to study the trade off for edge devices.

C. Previous research

The previous research [2] showed high compressibility ob java heap. It showed that application can be effectively reduced by up to 86%, on average 58%.

The major benchmark I am planning to use in this research is DaCapo[3] because it consists of a set of open source, real world applications with non-trivial memory loads.

II. Timelines

This project consists of multiple parts:

Part	Timeline
Set up Raspberry Pi + IBM J9	Mid October
Fix up decompression code - support array elements	Mid November
Get heap dumps from Raspberry Pi Apps	November
Get final Performance results on Raspberry Pi	December
Find a place in the GC to store metadata	Mid January
Implement compression in GC	End of February
Get final performance data	End of March
Add Hot/Cold objects	TBD

REFERENCES

[1] Dependability in Edge Computing. Paul Wood, Heng Zhang, Muhammad-Bilal Siddiqui, Saurabh Bagchi

[2] No Bit Left Behind:The Limits of Heap Data Compression. Jennifer B. Sartor, Martin Hirzel, Kathryn S. McKinley

[3] The DaCapo Benchmarks:Java Benchmarking Development and Analysis. Stephen M Blackburn et al.