JVM Performance

Engineering and Troubleshooting

Compilation of Ram Lakshmanan's Blog and Articles

Author

Ram Lakshmanan
(yCrash)

Index

Garbage Collection

1. "I don't have to worry about Garbage collection" – Is it true? ---------- 1
2. How to do GC Log analysis? ---------- 4
3. Garbage collection patterns to predict outages ---------- 7
4. Memory tuning: Key Performance Indicators ---------- 14
5. Tips to reduce Long GC Pauses ---------- 17
6. How many millions of dollars enterprises waste due to Garbage collection? ---------- 23
7. Simple &amp; effective G1 GC tuning tips ---------- 30
8. CMS deprecated – next steps? ---------- 37
9. Rotating GC Log Files ---------- 40
10. Sys time greater than User time Pattern ---------- 44
11. Real time greater than User and Sys time Pattern ---------- 46

Threads

12. How to take Thread dumps? – 8 options ---------- 49

Index

13. Overhead added by collecting thread dumps 57

14. Java Virtual Threads – quick introduction 60

15. Look for exceptions, errors in thread dumps 65

16. Thread dump analysis pattern – TREADMILL 68

17. Thread dump analysis pattern – ATHLETE 70

18. Thread dump analysis pattern –ATHEROSCLEROSIS 72

19. Thread dump analysis pattern – REPETITIVE STRAIN INJURY (RSI) 75

20. Thread dump analysis pattern – TRAFFIC JAM 79

21. Thread dump analysis pattern – ALL ROADS LEAD TO ROME 81

22. Thread dump analysis pattern – ADDITIVES 83

23. Thread dump analysis pattern – LEPRECHAUN TRAP 85

24. Thread dump analysis pattern – SEVERAL SCAVENGERS 88

Memory

25. How to Capture Java Heap dumps? – 7 options 90

26. Benefits of setting initial and maximum memory size 96

Index

to the same value

27. Flavors of OutOfMemoryErrors 101

28. Detect proactively whether application's memory is under-allocated 103

29. Best practices: Java memory arguments for Containers 108

30. Large or Small memory size for my application? 114

31. OutOfMemoryError related JVM arguments 118

32. How much memory is my application wasting? 121

33. Eclipse MAT – Tidbits 126

34. Shallow Heap, Retained Heap 130

JVM Performance

35. 7 JVM arguments of highly effective applications 135

36. Java Hashtable, HashMap, ConcurrentHashMap – performance impact 146

37. Java class loading – performance impact 152

38. Java UUID generation – Performance impact 161

Index

39. java.lang.System.getProperty()- performance impact 165

40. Java String intern(): Interesting Q &amp; A 169

41. StackOverflowError: Causes &amp; Solutions 174

42. Virtual Machine Error 179

43. JVM c1, c2 compiler thread – high CPU consumption? 183

44. finalize() method: behind the scenes 188

45. All you need to know about System.gc() 192

Chaos Engineering

46. Chaos Engineering – Metaspace OutOfMemoryError 196

47. Chaos Engineering – Blocked Threads 199

48. Chaos Engineering – Thread Leak 203

49. Chaos Engineering – Stackoverflow Error 207

50. Chaos Engineering – Simulating OutOfMemoryError 210

51. Chaos Engineering – Deadlock 214

52. Chaos Engineering – CPU spike 219

Index

Real-world case studies

53. Troubleshooting Microservice’s OutOfMemoryError: Metaspace 222

54. Memory leak due to improper exception handling 226

55. Garbage Collection tuning success story – reducing young gen size 231

56. Memory Wasted by Spring Boot application 236

57. Disappointing story on memory optimization 243

58. Troubleshooting deadlock in an apache opensource library 248

59. Troubleshooting App unresponsiveness due to Oracle DB 253

60. Troubleshooting HTTP 502 bad gateway in AWS EBS 258

61. AWS Cloud watch + yCrash = Monitoring + RCA 263

62. Troubleshooting CPU spike in a major trading application 267

1. “I don’t have to worry about Garbage collection” – Is it true?

I have heard a few of my developer friends say: **“Garbage Collection is automatic. So, I do not have to worry about it.“** The first part is true, i.e., “Garbage Collection is automatic” on all modern platforms – Java, .NET, Golang, Python… But the second part i.e., “I don’t have to worry about it.” – may not be true. It is arguable, questionable. Here is my case to showcase the importance of Garbage Collection:

1. Unpleasant customer experience

When a garbage collector runs, it pauses the entire application to mark the objects that are in use and sweep away the objects that don’t have active references. During this pause period, all customer transactions which are in motion will be stalled (i.e., frozen). Depending on the type of GC algorithm and memory settings that you configure, pause times can run anywhere from a few milliseconds to a few minutes. Frequent pauses in the application can cause stuttering, juddering, or halting effects to your customers. It will leave an unpleasant experience for your customers.

2. Millions of dollars wasted

Here is a white paper we published, explaining factually how enterprises are wasting millions of dollars due to garbage collection. Basically, in a nutshell, modern applications are creating thousands/millions of objects. These objects must be continuously investigated to determine whether they have active references or are they ready for garbage collection. Once objects are garbage collected, the memory becomes fragmented. Fragmented memory must be compacted. All these activities consume ***enormous compute cycles***. These compute cycles translate to millions of dollars in spending. If Garbage collection performance can be optimized, it can result in several millions of dollars in cost savings.