ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

Heap Memory Usage Analysis of Thread Safe Containers

Sampath Kini K[1]

Assistant Professor, Department of Computer Engineering, NMAMIT, Nitte, Karnataka, India[1]

ABSTRACT: Major concern of multi-threaded applications that is designed to have good expansion ability is about handling performance issues such as turnaround time and memory space consumed. This project is about the study of heap memory consumption of thread-safe containers for its write operations. This project compares the heap memory usage of both synchronized and concurrent containers of Java. A systematic method of performance analysis is opted. The system takes the workload from synthesized programs. Suitable workload parameters and system parameters chosen. Eclipse IDE is used for system implementation using framework designed. System uses utility package from java for determining heap memory usage. The system front end is created with the help of Swing from Java package.

KEYWORDS: Concurrency, container, heap memory.

I. INTRODUCTION

A collection also termed as container — being an object that puts together one or more elements into a one unit. Collections are needed to store, retrieve, manipulate data. It is also used to perform aggregation of data. They bring data items together which form a group which is natural, e.g a collection of items. Familiar collections such as ArrayList, HashMap and HashSet etc. But, they were not designed to be thread-safe. It exchibits inconsistencies when multiple threads[1] work on these containers at the same time**.**Through static methods present in the Collections class, Java has provision for enabling a collection which is thread safe. For example, Collections.synchronizedList(list) Collections.synchronizedMap(map).The parameters specified inside parentheses indicates non thread-safe container.However, it as good as operation being wrapped in a synchronised block. New package java.util.concurrent contains(ConcurrentHashMap,ConcurrentSkipListMap,CopyOnWriteArrayList etc.) that are made to be safe for use by more than one thread. Newly available concurrent classes are made to work faster than using a synchronized version of the normal collections. All of these collections help avoid Memory Consistency issues by defining a happens-before relationship between an action that adds an element to the collection with subsequent actions that access or delete that element.

The work in this paper is implemented in two stages. 1) Heap memory usage monitoring 2) Presentation of heap usage. Heap memory usage monitoring is done after applying workload for the selected containers using its write operation. Heap usage of respective container is recorded separately. Thereafter, recorded values of heap memory usage is used to present the graphical analysis of concurrent containers for comparison.
Paper is organized as follows. In section II, Related work is discussed. In section III, Details of framework used for carrying out analysis of thread-safe collection is provided. In section IV, Flowchart used for system implementation is drawn. In section V, results of experiments are presented. Section VI draws the conclusion of the analysis.

II. RELATED WORK

System considers following steps while doing the performing the analysis systematically: State the Goals of the proposed system, Specify the Services and its Outcome, Choose proper performance Metrics, Idenify Parameters for containers write operations, Factors to monitor, Identify Evaluation approach, Specify workload for container write operation, Sketch down experiments for analysis and Result presentation. System goal is to come-up with a design for determining heap usage by different collections. Methods for performing mutable actions are provided by Java utility

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

itself. These functions are invoked upon end user action.A suitable graphical interface is presented by the system for specifying user inputs. System also shows heap usage in the form of graphs or charts for the user. The type of the operation being performed and size of the input data will have a direct impact on amount of heap memory used. System parameters such as CPU speed, RAM size and JVM heap size are chosen. Workload parameters such as Number of simultaneous threads, size of input are chosen for the analysis. Factors such as oncurrentHashMap, CopyOnWriteArrayList, ConcurrentSkipListMap and corresponding synchronized version of it are selected. Measurements technique is used as evaluation technique since operations are available as part of container. A synthetic program which invokes container's write operation is developed. This piece of program sends across user selected elements as function arguments.The result showing the heap memory usage is be plotted as collection v/s memory consumption in the form of bar graphs.

III. FRAMEWORK DEVELOPED FOR PERFORNING ANALYSIS

A suitable framework has been designed for the determination of the heap memory usage of thread safe containers. (Figure 1) represents the framework designed for it. To represent individual thread-safe collection, distinguished class has been implemented. Write operations of individual collection has been implemented as part of this class.

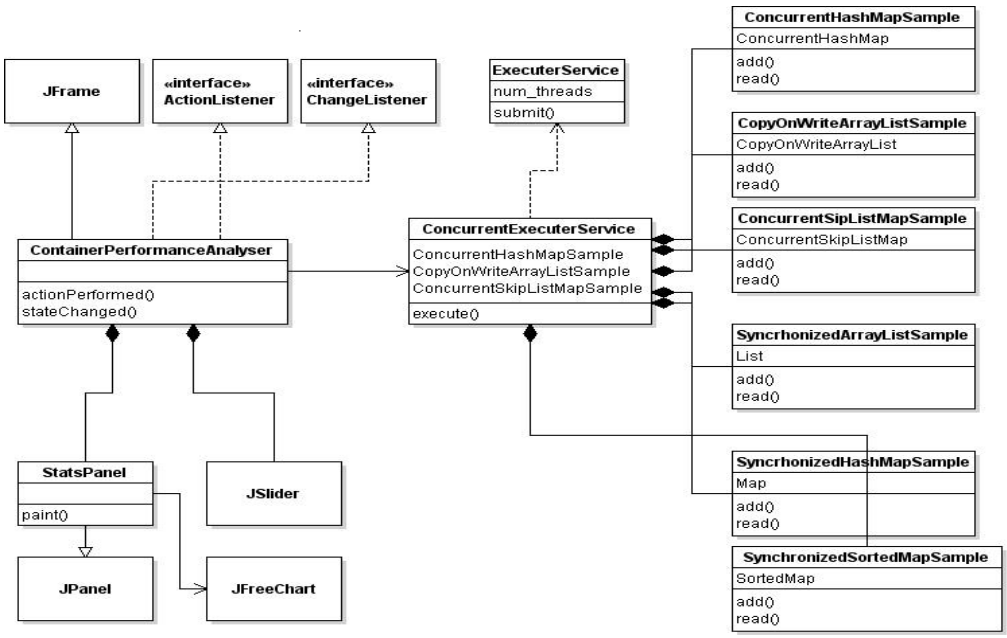

Fig. 1. Class diagram created for designing the framework for the analysis

In order to invoke write operations in multiple threads, ExecuterService class from the package java.util.concurrent was associted. As a parameter of workload, element of ten bytes size is selected. Various other packages were used in framework design realization. Swing package from Java is used for realizing user interface. Various packages such as Jpanel,JFrame, Slider is associated for specifying the user input as workload parameters. ExecuterService class is instantiated for performing write actions in more than one different threads. JFreechart component which comes as an open source library is referred for presenting the comparison graphically.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

IV. FLOWCHART DESIGNED FOR SYSTEM IMPLEMENTATION

The flowchart designed for system development is as shown below. It starts with specifying user inputs as workload parameters such as data size, number of threads, type of collection. As a next step, workload data for the container write operations is generated with the help of piece of program.

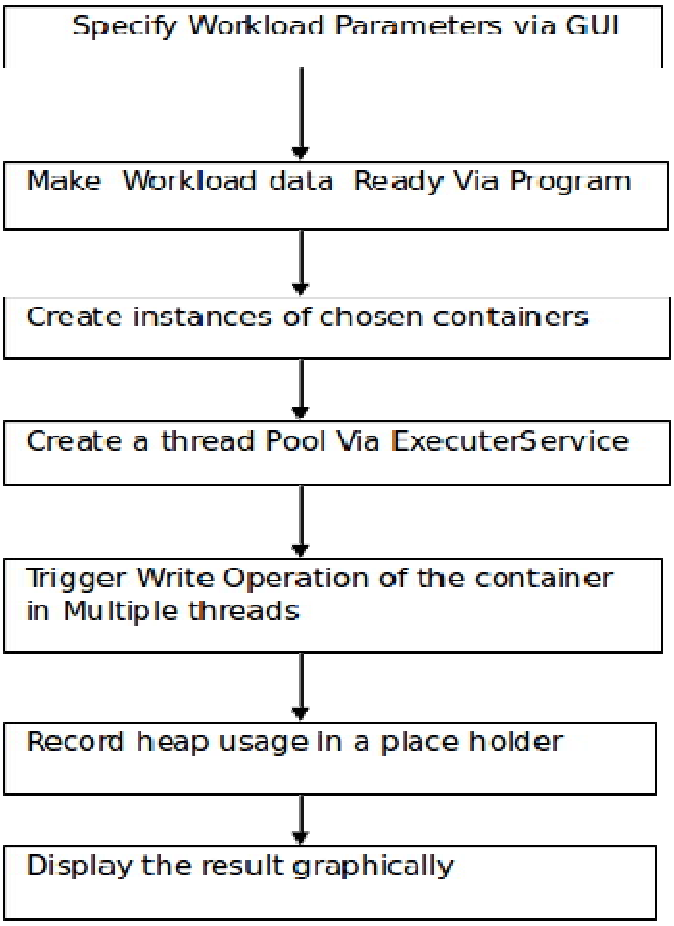

Fig. 2. Flowchart used for system implementation

After that, instances of listed containers are created. Next, thread pool is obtained via ExecuterService class in order to trigger the write functions in multiple threads. Finally the outcome is drawn graphically in the form of coloured bar chart.

V. RESULTS OF EXPERIMENTS

Measured and recorded heap usage by individual collection is compared and drawn via bar charts.(Figure 3).The experiments have been conducted on a computer system having processor with four cores and RAM of 4GB space. Several runs of the experiments have been conducted on each of collection using its write operation.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

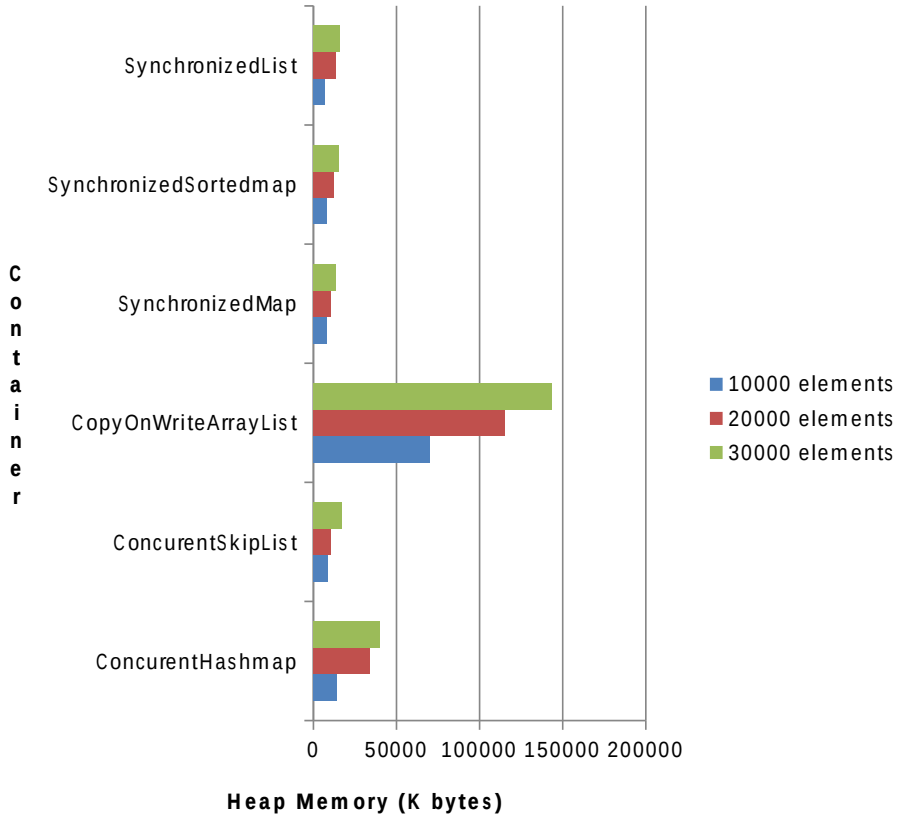

Fig. 3. Chart showing the experimental results of container's write operations.

Experiments are conducted by applying input data having size as 10000,20000 and 30000 elements as shown in the figure. All runs average was applied for the comparison in the end. To determine heap usage for the container's write operation, utility methods present in Java's system package is invoked.

VI. CONCLUSION

This paper determined heap memory usage of Java concurrent and synchronized containers. It is found that heap memory usage does not differ across synchronized verisons as seen from the chart.However, heap memory usage differs across concurrent container types. As we see from the graph, CopyOnwriteArrayList container has more heap usage in all of the situations with respect write operations. It is seen that ConcurrentHashMap has next highest heap memory usage in most of the experimenal runs. This paper showed the comparison of heap memory usage among the container types. While every container type has its own place, this comparison would assist in choosing on the type of collection for a given circumstance. These techniques can also be applied for determining heap consumption across C++ and C# technologies.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 6, June 2016

REFERENCES

[1] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "Aportable programming interface for performance evaluation on modern processors," Int. J. High Perform. Comput. Appl.,vol. 14, no. 3, pp. 189–204, Aug. 2000.

[2] M. Hirzel, "Data layouts for object-oriented programs," in Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, ser. SIGMETRICS '07.New York, NY,*USA: ACM, 2007, pp. 265–276,2007*

[3] J. M. Bull, L. A. Smith, M. D. Westhead, D. S. Henty, and R. A. Davey, "A benchmark suite for high performance java," Concurrency: Practice and Experience, vol. 12, no. 6, pp. 375–388, 2000.