International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 4; Issue: 2; February -2015

Behavioral and Performance Analysis of Thread Safe Object Containers

Sampath Kini K[1]
Computer Science Department,
NMAMIT Nitte,
Karkala,Karnataka, India

Vijaya Murari[2]
Computer Science Department,
NMAMIT Nitte,
Karkala,Karnataka, India

***Abstract*: One significant problem of multithreaded applications that is designed for better scalability is about dealing with the performance issues such as response time and memory. This paper focuses on a study of resource usage of thread-safe object containers, providing insights on strategies to decide upon type of the container for a given requirement. The paper compares performance of synchronized containers and concurrent containers of java and presents the impact of these thread safe object containers in terms of resource consumption.**

Keywords— Container, concurrency, multithreading

I. INTRODUCTION

A *collection* — also called as container — is an object that groups multiple elements into a single unit. Collections **[1]** are used to store, retrieve, manipulate, and communicate aggregate data. Typically, they represent data items that form a natural group, such as a (a collection of items), a mail folder (a collection of letters), or a telephone directory. Java provides familiar collections such as arraylist, hashmap and hashset. However, they are not thread-safe. They show inconsistencies when multiple threads use these object containers simultaneously.

In section 2, we provide a brief introduction about synchronized containers and their limitations. We also discuss features of concurrent containers in Java. In section 3, we provide the systematic approach**[4]** used for implementation of performance analysis. In section 4, we describe framework developed for conducting analysis of thread-safe object containers. In section 5, we discuss about flowchart used system implementation. In section 6, experimental results are captured.

II. SYNCHRONIZED AND CONCURRENT CONTAINERS

Java provides thread-safe collection *wrappers* via static methods in the Collections class. Following are the synchronized collections in java.

Collections.synchronizedCollection(coll)
Collections.synchronizedList(list)
Collections.synchronizedMap(map)
Collections.synchronizedSet(set)

These are essentially the same as wrapping each operation on the collection in a synchronized block. New package java.util.concurrent contains collections that are optimized to be safe for use by multiple threads. Some of these containers are ConcurrentHashMap, ConcurrentSkipListMap, CopyOnWriteArrayList. These classes are generally faster than using a synchronized version of the normal collections because multiple threads are actually able to use them at the same time, to a degree. These container classes get rid of memory consistency errors by defining a happens-before relationship between an mutable and immutable operations being performed on containers. Concurrent container such as Concurrent HashMap uses [2]fine-grained critical sections. It treats individual element as the shared resource, not the *entire* hash table array. The idea is to have the operation just lock that element instead of entire array. If a thread is updating element *e*, other threads that wanted to read/write other elements besides *e* could still do so.

III. PERFORMANCE ANALYSIS APPROACH

In this section, we are capturing the systematic approach followed for the performance analysis of the concurrent and synchronized containers.

A. State goals and define system

Container classes are used for holding the data items in the memory for processing. These classes provide both mutable and immutable operations such as read/write. Goal of the system is develop framework for determination of resource consumption of the individual container classes. System will have functions for performing both mutable and immutable operations from end user. System will have relevant graphical user interface for specifying user inputs. System also provides pattern of resource usage in the form of graphs or charts.

B. List Services and outcomes

System considers operations read and write on the object container classes. The resources used by the operations depend upon type of the operation, number of data parameters used and size of the data.

C. Select Metrics

System considers following performance metrics while measuring the resource consumption on each type of operation.

- Elapsed time per operation.
- time required to complete a block of n successive operations

D. List Parameters

System considers following system parameters[3] and workload parameters.

- **System Parameters:** Speed of the CPU, Size of the main memory/RAM, Heap size of JVM and Operating system overhead
- **Workload parameters:** Number of threads and Number and sizes of the call parameters.

E. Select Factors to Study

Following factors are considered

Container Type: ConcurrentHashMap, CopyOnWriteArrayList, ConcurrentSkipListMap,Number of threads, Size of the Heap, Sizes of the data parameters

F. Select Evaluation Technique

Measurements will be used for evaluation since all the concerned operations have been implemented as part of container implementation.

G. Select workload

The workload will consist of a synthetic program generating the specified types of container operation. This program will also monitor the resources consumed and log the measured result

H. Design experiments

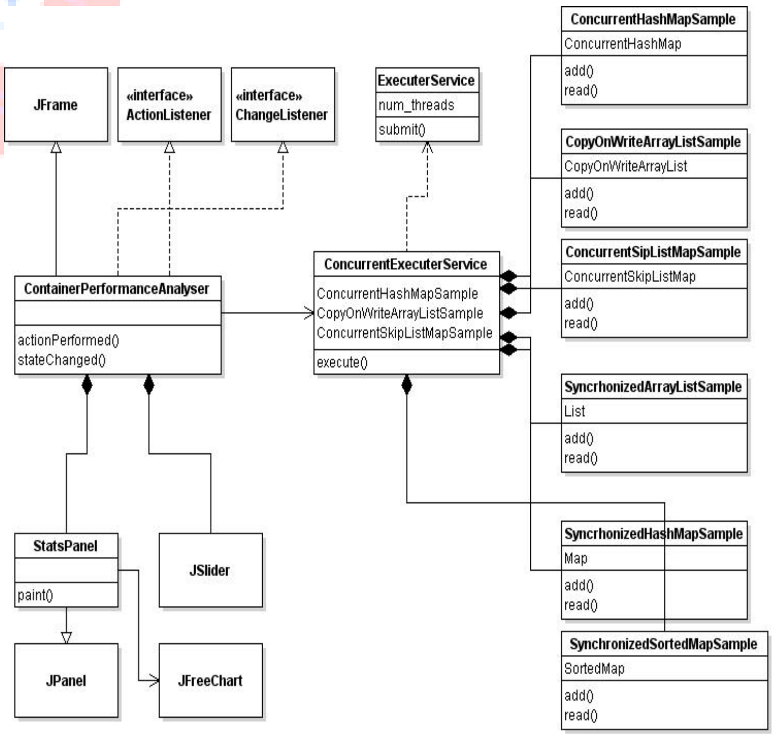
A full factorial experimental design by considering all factors mentioned above with sufficient number of experiments will be used

I. Present results

The final result will be plotted as container versus resource consumption in the form of graphs or charts.

IV. FRAMEWORK FOR THE SYSTEM DESIGN

We have designed a framework for the implementation of the performance analysis of thread safe object containers. Class diagram (Figure 4.a) is developed for it. A separate class has been identified for each of the container type. Both mutable and immutable operations are implemented in these container classes. ExecuterService class of the java package named java.util.concurrent is used for performing container operations in multiple threads. The container element with the size of ten bytes is chosen for workload parameter

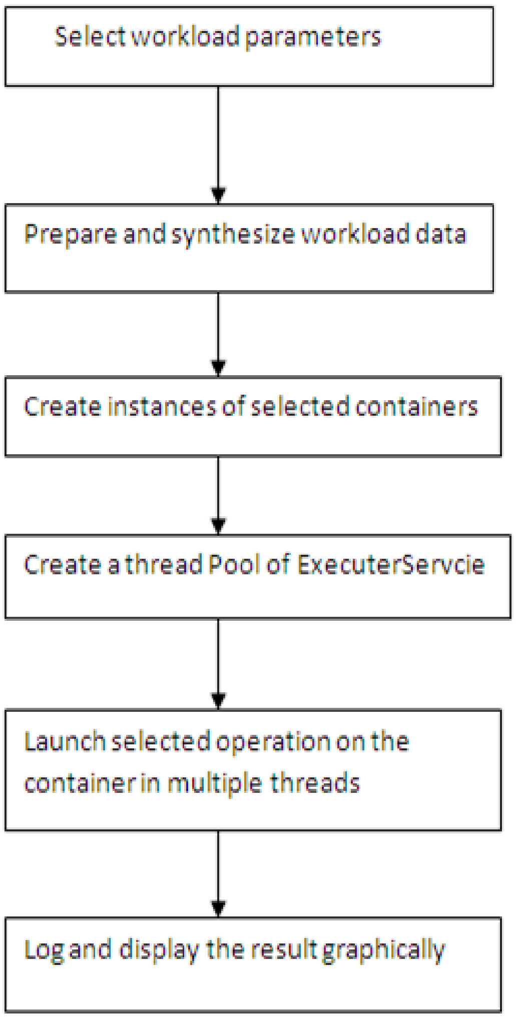


.

FIGURE 4.A CLASS DIAGRAM OF THE SYSTEM

V. IMPLEMENTATION OF THE FRAMEWORK

In this section we are specifying key steps followed in realizing the framework that has been designed.Java's swing package is used for developing user interface. The components such as JFrame, JPanel, JSlider is used for specifying the user input for the workload parameters. ExecuterService class is used for performing both write and read operations in multiple threads. JFreechart a open source library is used for showing the comparison graphically in the system.


Select workload parameters
Prepare and synthesize workload data
Create instances of selected containers
Create a thread Pool of ExecuterServcie
Launch selected operation on the container in multiple threads
Log and display the result graphically

VI. EXPERIMENTAL RESULTS

In this section measured data for each of the container type is compared and shown as graph.(Figure 6.A,6.B,6.C). These experiments were conducted on a machine with dual core processor and 2 GB RAM. Several runs of the workload are conducted for each of container and its operation, and finally average of all runs is considered for the comparison. An utility function provided in Java's system package is used for determining elapsed time for the container's operation.

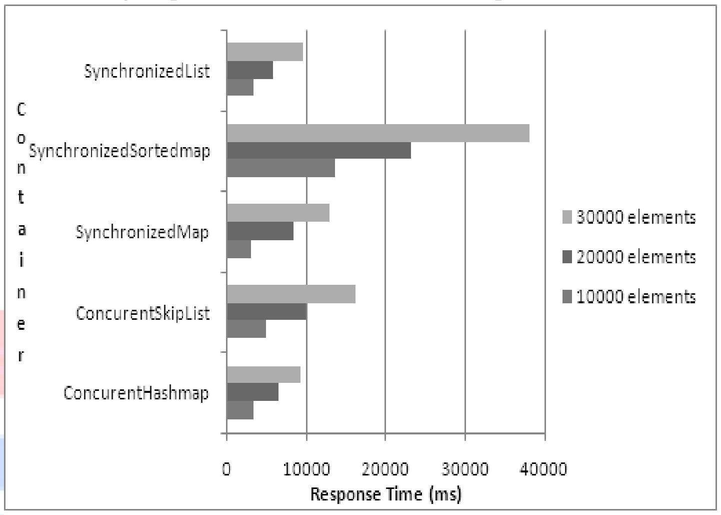


FIGURE 6.A COMPARISON ON WRITE OPERATION

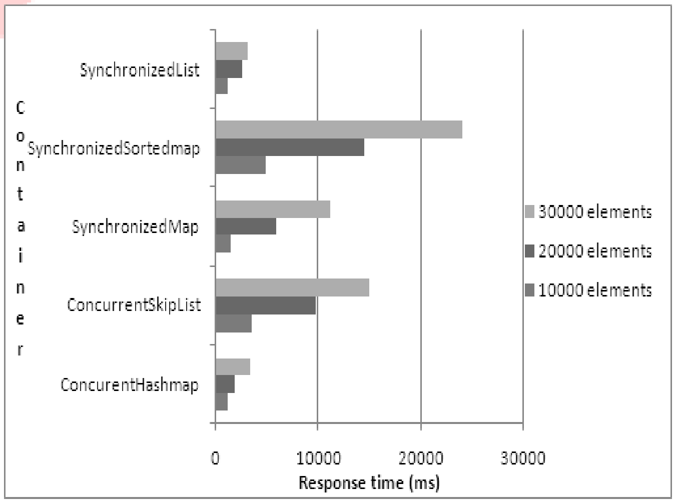


FIGURE 6.B COMPARISON ON READ OPERATION

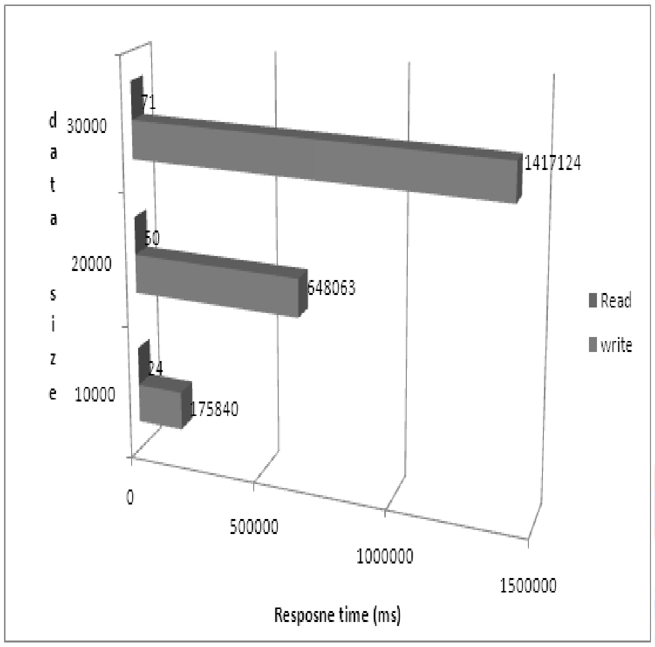


FIGURE 6.C COMPARISON WRITE AND READ OF COPYONWRITEARRAYLIST

VII. CONCLUSION

This paper identified the system resource usage of Java thread safe object containers. The system resource consumption differs across container types. Both synchronized and concurrent versions of container differ in their resource consumption. As we see from the graph concurrent version of container performs better in most of the situations with respect to response time for both read and write operations. This paper showed that the container CopyOnwriteArrayList has minimal response time among all containers for read operation. However, its response time is more for write operation among all containers. Though each of the container type has its own specific usage based on the need, this performance analysis will help in deciding on the container type for a given requirement. The presented study focused mainly on the response time, but the proposed techniques can also be extended for determining heap usage of it.

VIII. REFERENCES

[1] "Impact of data structure layout on performance", 2013 21st Euromicro International Conference.

[2] "Data layouts for object-oriented programs," in

Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer

[3] "A benchmark suite for high performance java," Available: http://dx.doi.org/10.1002/1096-9128(200005)12:6¡375::AID-CPE480¿3.0.CO;2-M

[4] The art of computer systems performance analysis