THREAD-BASED, REGION-BASED MEMORY MANAGEMENT

Tristan Basa, Gerhard Dueck, Kenneth Kent
University of New Brunswick, IBM Canada
Faculty of Computer Science
tbasa@unb.ca, gdueck@unb.ca, ken@unb.ca

INTRODUCTION

Region-based memory management divides the heap into fixed-sized regions such that partial garbage collection (GC) need only be done on certain parts of the heap. This offers the flexibility of being able to allocate only on certain parts of the heap. By allocating only in regions owned by a thread, and subsequently collecting only in those regions (Figure 1), we aim to avoid long system-pause times caused by a *stop-the-world* approach stopping all mutator threads while GC is being performed.

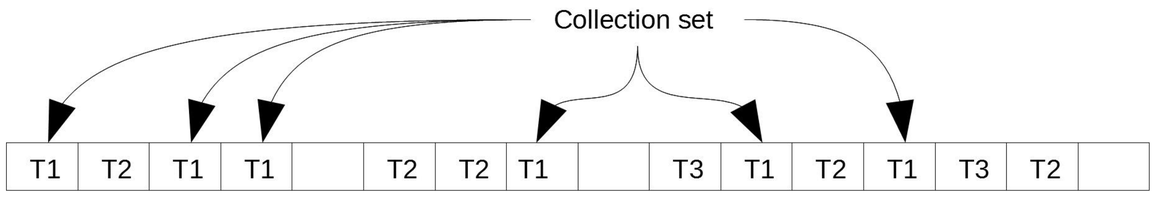

Figure 1. Thread-based, region-based memory management.

THREAD-BASED SOLUTION

- *Region ownership* — assign regions to threads.
- Allocate only in regions that were assigned to the allocating thread.
- GC is done only on regions of the chosen thread so that other threads can continue execution.
- Other threads that may have to be stopped during a GC can be determined via *escaping objects.*
- Escaping objects are objects that have pointers from objects in regions that belong to other threads (Figure 2).
- Escaping objects are the basis for thread relationships.

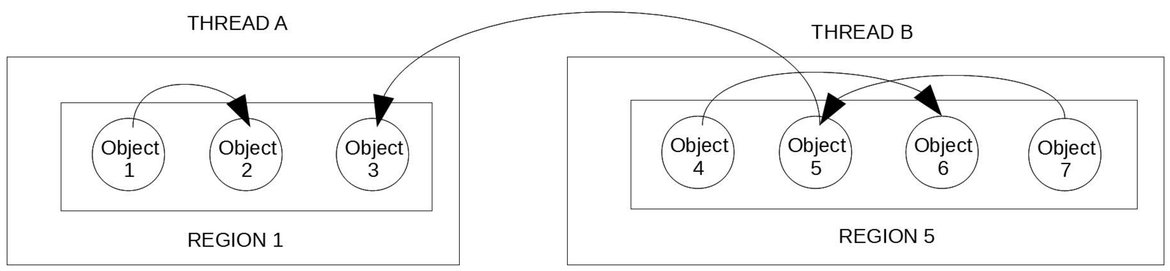

Figure 2. Object 3 escapes to thread B.

EXPERIMENTAL SETUP

- Memory management operations and relevant information were captured into trace files by instrumenting the Java Virtual Machine (JVM) (Figure 3).
- The GC simulator takes a trace file as input and outputs a log file that contains statistics on the benchmark and performance of the GC algorithm.
- Benchmarked with DaCapo 9.12 and SPECjvm2008.
- Only benchmarks that triggered at least one GC were considered (Table 1).

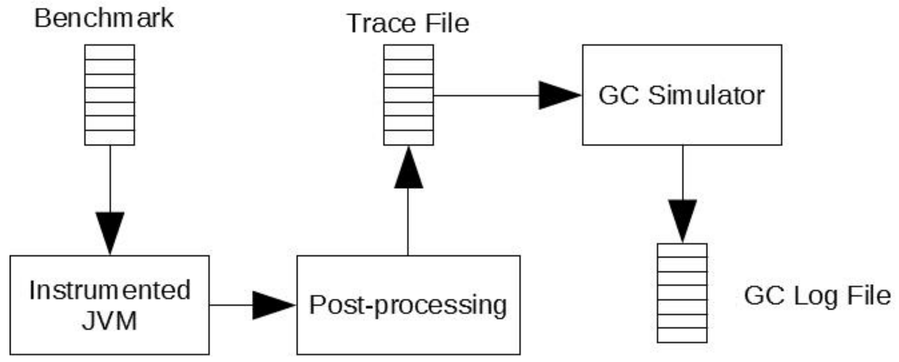

Figure 3. Experimental Setup.

EXPERIMENTAL RESULTS

When collection is triggered, the chosen thread for GC could potentially stop at most the threads that are related to it. Table 1 shows the related threads after the first GC.

Table 1. Related threads and percentage of total threads.

Benchmark	Related Threads	# Related	Total	%
compiler.compiler	T1,T5,T6,T8,T9	5	16	31.25%
compiler.sunflow	T8,T9	2	16	12.5%
lusearch	T0,T1,T6–29	27	33	81.81%
serial	T0,T1,T8,T9	4	16	25%
sunflow	T6,T10,T16,T18	4	24	16.66%
xalan	T1,T5–28	25	33	75.75%
xml.transform	T0,T1,T6,T8	4	16	25%
jython	T0,T1,T3,T5	4	10	40%

CONCLUSION

Results show that with thread-based, region-based memory management, on average, about 62% of the threads can continue executing without having to stop when collection occurs. This can potentially avoid long system pause times.