Runtime Infrastructure - Bug #8135

replace ScopedDictionary for better performance

12/19/2023 11:11 AM - Eric Faulhaber

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		
Description			

History

#1 - 12/19/2023 11:21 AM - Eric Faulhaber

ScopedDictionary and to a lesser degree its subclass ScopedSymbolDictionary are used throughout the runtime to manage resources which have to be available at every block level and may change at any block level transition. The current implementation (and use patterns) are very heavy users of hash maps and sets, and perform a lot of expensive set and map operations.

I have been working on a replacement which still relies on hash map at its core, but the idea is to require one map lookup for the entire stack of scopes, rather than a map lookup at every scope until a value is found. This requires more memory and some "walking" of arrays to find a value, but I am hoping this will be a faster alternative.

I don't think this internal implementation change alone will be enough, however, to alleviate the full performance penalty of these data structures. Some changes to the use patterns of how we manage the changes to the resources within a particular scope during scope transitions will be needed as well.

I will create a branch for my changes so far.

#2 - 12/21/2023 02:42 PM - Eric Faulhaber

Some notes on the current prototype replacement...

I've gotten rid of the dictionaries at each scope level. There is one hash map whose keys are the keys passed in by callers, and whose values are an index of an array. Each scope has such an array. For normal lookups, we walk from the top of the stack, looking at that index in each scope's array, until we find a non-null value. I think it must be faster having a single map lookup and returning immediately if there's no hit, or walking down the scopes and doing an array dereference at each scope. But I have to test this under real conditions.

Unfortunately, there's a lot of work going on in these scoped dictionaries besides lookups. We have a lot of logic in code that uses the scoped dictionaries which makes assumptions about the internal implementation; i.e., that there are maps at each scope. During scope transitions, there is a lot of work which accesses those maps directly and does bulk copying using Map.putAll(...) or Map.keySet().addAll(...). This will have to be refactored somehow. I've already re-implemented the propogate feature of the scoped dictionary to be much faster than before, but that feature is not used as widely as I'd like.