Base Language - Bug #8154

refactor scopeable stacks

12/21/2023 12:45 PM - Constantin Asofiei

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	30%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 12/21/2023 12:46 PM - Constantin Asofiei

From #7026-296:

> Constantin Asofiei wrote:
>
> > On a side note: all scope support is managed in various stacks (or scope dictionaries) at the specific Scopeable implementation; but, I think, some of these belong to the TransactionManager$WorkArea.blocks stack (in other words, at the BlockDefinition). I think we can create a specific top-level BlockDefinition sub-class to hold top-level block state and check which scopeable state can be moved there.
> >
> > Why I mention this: all this stack push/pop, my feeling is that is expensive; on a conceptual approach, keeping this state encapsulated at the Scoppeable implementation is OK, but I think it hurts us on a performance POV.

> Greg Shah wrote:
>
> > It is a good idea. I wonder if we would also benefit by implementing the stack as a pre-allocated array using named WorkArea variables to index into it (int currentBlock, int nearestTopLevel...). There would be no push/pop operations and no search of the stack for things like the nearest top level. The only time we would take extra effort is if we have to reallocate the array to grow it. That should be rare if we size it appropriately.

#2 - 12/21/2023 01:19 PM - Eric Faulhaber

Constantin Asofiei wrote:

> [...]
> Why I mention this: all this stack push/pop, my feeling is that is expensive; on a conceptual approach, keeping this state encapsulated at the Scopeable implementation is OK, but I think it hurts us on a performance POV.

I agree. This is why I have been working on #8135. How much do you think this idea (i.e., refactoring scoped dictionaries to instead work within the TransactionManager$WorkArea.blocks stack) will obviate the need for #8135? If we will still need scoped dictionaries, I will continue this effort. I am pretty deep into it.

#3 - 12/21/2023 01:51 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Constantin Asofiei wrote:
>
> > [...]
> > Why I mention this: all this stack push/pop, my feeling is that is expensive; on a conceptual approach, keeping this state encapsulated at the Scopeable implementation is OK, but I think it hurts us on a performance POV.
>
> I agree. This is why I have been working on #8135. How much do you think this idea (i.e., refactoring scoped dictionaries to instead work within the TransactionManager$WorkArea.blocks stack) will obviate the need for #8135? If we will still need scoped dictionaries, I will continue this effort. I am pretty deep into it.

I need to digest this more; for scoped dictionaries, in my mind I have something like this for lookup: instead of going down the stack and look in each dictionary for a key, keep a reverse lookup for each key to its 'scope list'. Are your changes something like this?

#4 - 12/21/2023 02:43 PM - Eric Faulhaber

Constantin Asofiei wrote:

> I need to digest this more; for scoped dictionaries, in my mind I have something like this for lookup: instead of going down the stack and look in each dictionary for a key, keep a reverse lookup for each key to its 'scope list'. Are your changes something like this?

Similar. Please see #8135-2.

There are other issues that I'm working through besides what is mentioned there, but the main thing is knowing whether the effort is worth it, given the idea in this issue. It seems to me there is room for both, but I wanted to get your opinion.

#5 - 04/01/2024 12:51 PM - Constantin Asofiei

- Status changed from New to WIP

- % Done changed from 0 to 30

- Assignee set to Constantin Asofiei

Created task branch 8154a from trunk rev 15108.

Rev 15109 contains first pass at moving scoped stacks into (TopLevel)BlockDefinition.

I need to do more profiling with these changes, and parts of them I may extract and commit in different branches.

#6 - 04/01/2024 06:41 PM - Greg Shah

Code Review Task Branch 8154a Revision 15109

The changes look promising. There is clearly quite a bit more that can be done.

#7 - 04/29/2024 10:56 AM - Constantin Asofiei

- Related to Bug #8703: NO-UNDO option at DMO must be removed added

#8 - 04/29/2024 10:56 AM - Constantin Asofiei

- Related to deleted (Bug #8703: NO-UNDO option at DMO must be removed)