

CoCoALib - Feature #437

MemPool: order free blocks?

08 Feb 2014 15:18 - John Abbott

Status:	In Progress	Start date:	08 Feb 2014
Priority:	Normal	Due date:	
Assignee:		% Done:	10%
Category:	Tidying	Estimated time:	15.00 hours
Target version:	CoCoALib-1.0	Spent time:	1.30 hour
Description Computation speed in C5 often degrades over time; I think this may be due to loss of data locality in the RAM. In any it would be nice to offer the use the chance to sort the free blocks into order. Perhaps there could also be a "global" option to make all @MemPool@s sort their free blocks?			

History

#1 - 02 Apr 2014 10:05 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99534 Seoul14

#2 - 17 Jul 2014 14:18 - John Abbott

- Target version changed from CoCoALib-0.99534 Seoul14 to CoCoALib-1.0

Moving this to after Seoul -- it's too risky to try implementing it in a rush (and then pass sleepless nights trying to find out why it doesn't work).

Anyway, even if I do implement it, when will it be called?

One possibility would be to have a global register of mempools, and every so often a "message" is sent to them saying to reorder their free lists (e.g. this could be done just before printing out the user prompt in CoCoA-5).

Or maybe a MemPool could count how many new/deletes, and reorder every 65536 calls...

#3 - 17 Jul 2014 14:24 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

- Estimated time set to 15.00 h

Christof made some interesting suggestions:

1. copy the free list (or first part of it) into a vector, sort the vector, and then overwrite the free list with the sorted equivalent -- this looks like it would have nice memory locality (and is probably faster than sorting the free list in place)
2. another Christof idea is to maintain free stack (perhaps of limited size); if the stack fills ups, sort it and write its (half?) contents out into a free list; if the stack becomes empty, refill it (half-way?) from the start of the free list (perhaps also sorting?)

Idea (1) looks to be easy (& quick?) to implement. Idea (2) implies a more radical rewrite of MemPool. Perhaps I'll see what Schoenemann says...

#4 - 18 Jul 2014 14:57 - Anna Maria Bigatti

John Abbott wrote:

Or maybe a MemPool could count how many new/deletes, and reorder every 65536 calls...

I have a mild preference for this choice

#5 - 09 Dec 2014 16:22 - John Abbott

It seems the correct (portable) way to sort a vector<void*> is to use `sort(v.begin(), v.end(), std::less)`.

Aparently `std::less` is guaranteed to respect a total order on the pointers; *e.g.* see the post <http://stackoverflow.com/questions/1098966/universal-less-for-pointers-in-c-standard>

#6 - 13 Apr 2015 23:31 - John Abbott

I have just looked at some stats about MemPool in one very particular example, and now wonder whether sorting the entire free list is such a clever move.

The computation was simply multiplying two largish multivariate polynomials (entries in the classical adjoint of an 8x8 matrix each of whose entries is a distinct indet). The two polys contained 5040 terms each, and their product contained almost 10000000 terms.

I observed that during the computation it often happened that many loaves were about 99% full, with just a few loaves having more space (often these were the first, smaller loaves). Sorting the free list may actually lead to poor data locality in these conditions... :-/

What to do?