Disk-backed STL container classes? [closed]

Asked 16 years, 2 months ago Modified 14 years ago Viewed 6k times

41

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 6 years ago.

Improve this question

I enjoy developing algorithms using the STL, however, I have this recurring problem where my data sets are too large for the heap.

I have been searching for drop-in replacements for STL containers and algorithms which are disk-backed, i.e. the data structures on stored on disk rather than the heap.

A friend recently pointed me towards stxxl. Before I get too involved with it... Are any other disk-backed STL

replacements available that I should be considering?

NOTE: I'm not interested in persistence or embedded databases. Please don't mention boost::serialization, POST++, Relational Template Library, Berkeley DB, sqlite, etc. I am aware of these projects and use them when they are appropriate for my purposes.

UPDATE: Several people have mentioned memory-mapping files and using a custom allocator, good suggestions BTW, but I would point them to the discussion here where David Abraham suggests that custom iterators would be needed for disk-backed containers. Meaning the custom allocator approach isn't likely to work.

c++ algorithm data-structures stl on-disk

Share

Improve this question

Follow

edited Oct 2, 2008 at 4:23

prakash

59.6k ● 25 ● 94 ● 115

asked Sep 29, 2008 at 16:29

oz10

158k ● 27 ● 98 ● 129

If your datasets are too large for the heap, you should consider whether your system architecture is correct (e.g., whether to move to a 64-bit system; these are more common now than when the question was asked). You should also consider whether the STL is the right approach; it may be

making assumptions about dataset size which don't hold for you. – Donal Fellows Dec 1, 2010 at 6:15

@Donal He might not be reserving the correct amount of memory. – the_drow Mar 9, 2011 at 22:58

4 Answers

Sorted by: Highest score (default)

10

I have implemented some thing very similar. Implementing the iterators is the most challenging. I used **boost::iterator_facade** to implement the iterators. Using `boost::iterator_facade` you can **easy adapt** any cached on **disk data structures** to have a STL container interface.

Share Improve this answer

Follow

answered Sep 30, 2008 at 3:06

Ted

15.3k ● 6 ● 30 ● 28

I use Boost, but iterator_facade is new to me. I'll have to take a look at that, thanks for sharing. – oz10 Sep 30, 2008 at 5:53

1 Care to share the implementation? – the_drow Mar 9, 2011 at 22:56

Unfortunately I can't share that code. However if you have any questions I would be more than happy to help. – Ted Mar 15, 2011 at 22:16

I've never had to do anything quite like this, but It might be possible to do what you want to do by writing a custom

8

allocator that makes use of a memory mapped files to back your data.

See boost::interprocesses for docs on their easy to use implementation of memory mapped files, this Dr. Dobbs article for a detailed discussion on writing allocators, and this IEEE Software column for a description of the problem and example code.

Share Improve this answer

Follow

edited Dec 1, 2010 at 6:00

Diomidis Spinellis

19.3k ● 6 ● 66 ● 85

answered Sep 29, 2008 at 17:32

christopher_f

1,985 ● 1 ● 13 ● 12

> I am aware of the DDJ article, nice article. However, there was a discussion on boost's mailing list between Terpstra, Kuehl, & Abraham suggesting that custom iterators would be needed to deal with on-disk data structures... which kind of rules out the custom allocator approach. – oz10 Sep 29, 2008 at 18:04

3

If (as you write) you're not interested in persistence the simplest solution would be to increase your heap size and use your operating system's virtual memory facilities. The part of the heap that will not fit in your computer's physical memory will end up being paged on disk, giving you exactly what you want: normal STL access to data often stored on disk. The operating system will take care

of caching the most used pages in the physical memory and evicting to disk those you don't use a lot. Your code will remain the same, and you can increase its performance simply by adding more physical memory.

To increase your heap size check your operating system's parameters, like ulimit(1) on Unix systems and System properties - Advanced - Performance - Advanced - Virtual Memory on Windows XP. If you've hit the 32-bit 4GB limit consider moving to a 64 bit architecture or compiling your program for 64 bits.

Share Improve this answer

Follow

answered Sep 29, 2008 at 16:46

Diomidis Spinellis

19.3k ● 6 ● 66 ● 85

I am a proficient systems administrator, I have considered the approach that you suggested. I have a amd64 machine running unix. I cannot afford adding more physical memory. My swap space is 2GB, my data set is 42GB, my hard drive is 1TB... – oz10 Sep 29, 2008 at 16:51

1 So, how about increasing your swap space? – user3458 Sep 29, 2008 at 17:50

Sure, I've thought about rebuilding my server to accommodate this project. However this isn't dedicated hardware, rather a machine used for general research. Beyond that, eventually collaborators will need to run the code too, and I don't think requiring them to rebuild their machines will work. – oz10 Sep 29, 2008 at 18:02

2

I don't know much about the subject, but it might be possible to write an STL-like interface to a memory mapped file?

edit: This approach might be suitable if you're trying to get at a specific part of a huge file. If you're attempting to do something with the entire file, you'll likely generate a huge number of page faults as you read in uncached parts of the file.

Share Improve this answer Follow

edited Sep 29, 2008 at 16:44

answered Sep 29, 2008 at 16:38

luke

37.4k ● 8 ● 61 ● 81

I have considered doing this... but I'd rather not write it myself if something useable has already been done. – oz10 Sep 29, 2008 at 16:44

I can appreciate not wanting to reinvent the wheel. I'm not familiar with a library that does this; hopefully someone can recommend one. – luke Sep 29, 2008 at 16:49

I believe Boost.Interprocess can be used to write to a memory mapped file. Haven't actually tried it, though. – Nemanja Trifunovic Sep 29, 2008 at 17:05