Subject: Re: Bizarre slowness from sort()
Posted by Ivan Valtchanov on Tue, 23 Apr 2002 15:58:44 GMT
View Forum Message <> Reply to Message

Hi,

On my machine (Linux, Pentium-III, 512MB RAM, IDLv.5.4) it took 1 sec.

Here it is:

```
x = randomu(s,400000,/long)
t0=systime(1) & is=sort(x) & print,systime(1)-t0,' sec'
Output: 1.1320790 sec
```

With your save-ed file:

```
IDL> help,sortme
SORTME          LONG      = Array[376467]
IIDL> print,minmax(sortme)
% Compiled module: MINMAX.
           0       65530
IDL> t0=systime(1) & is=sort(sortme) & print,systime(1)-t0,'sec'
      0.74189603sec
IDL>
```

So there must be something strange in your version...

Cheers,
Ivan

Subject: Re: Bizarre slowness from sort()
Posted by Jonathan Joseph on Tue, 23 Apr 2002 16:14:48 GMT
View Forum Message <> Reply to Message

A follow up to this issue.

The problem seems to be in the sorting algorithm. It really doesn't like to sort a large array with a small number of distinct values. Has the SORT algorithm changed since 5.3? Or is there a different algorithm used under HP-UX?

No need to download that save file.
Here's a simple extreme example (array of 200,000 ones followed by 200,000 zeros) - takes a VERY long time to sort.

```
IDL> a = [intarr(200000) + 1, intarr(200000)]
IDL> b = sort(a)
```

-Jonathan

Jonathan Joseph wrote:
>
> Hello,
>
> My colleague complained of an incredible slowness when trying to sort
> an array of long integers (on the order of 400,000 of them). I said
> "you're nuts. Must be a bug in your code" and proceeded to generate
> a random array of 400,000 long integers and sort them very quickly.
> "See, it works fine."
>
> So, he showed me his code, and it all looked perfectly normal, and
> the sort took minutes! The data looked fine (no bizarre values)
> so we created a save file, opened up a new IDL session tried to sort
> the data and saw the same slowness!
>
> I've found that the problem occurs on SUN and Windows 2K running IDL
> 5.5, but not on HP-UX running IDL 5.3. Also, we have found
> a workaround for the integer case. Adding a small (less than 1) random
> offset to each element of the array before sorting will make it work
> quickly and yield the correct result. But this will not work properly
> unless the array to be sorted is an integer type array, otherwise you
> could be changing the sort order by adding the random offset. Just
> converting the array to float or adding a constant offset to each
> element does NOT fix the problem.
>
> This behavior seems very strange - possibly a bug in IDL. Anyone
> have any thoughts on this? Can you reproduce this bug on your
> system?
>
> The save file is located at
> http://baritone.astro.cornell.edu/~jj/idl2/
> and is called 'sort.bin' (about 1.4 megabytes)
>
> IDL> restore, 'sort.bin'
> IDL> help, sortme
>
> SORTME LONG = Array[376467]
>
> IDL> a = sort(sortme)
>
> Works, but takes minutes to return.
> If I add a random number between 0 and 0.1 to each element
> and then sort, it works very rapidly (and produces the correct
> result since it is not changing the sorting order)
>

> IDL> b = sort(sortme + randomu(seed, n_elements(sortme)) * 0.1)
>
> Works very fast as expected
>
> Anyone know what's going on?
>
> Thanks.
>
> -Jonathan

Subject: Re: Bizarre slowness from sort()
Posted by Jonathan Joseph on Tue, 23 Apr 2002 18:56:59 GMT
View Forum Message <> Reply to Message

I had the following enlightening discussion about qsort with a programmer from RSI who responded to my post, but I'm still not entirely convinced. Can anyone comment on this?

-Jonathan

(posted with permission)

RSI

You've stumbled over a well known flaw in the standard qsort() function, which forms the heart of IDL's SORT. Jon Bentley (the most excellent Programming Pearls guy) wrote a very good column about this for the now defunct Unix Review about a decade ago:

149.J. L. Bentley, Software Exploratorium: The Trouble With Qsort, UNIX Review, Vol. 10, 2, pp. 85--93, February 1992.

I am not able to locate a copy for you online, so I am recalling this from memory:

- qsort(), as implemented in Unix V7, would go quadratic in performance when presented with input that is already in approximate sorted order. Ironically, input that is not close to sorted order sorts very quickly,

- qsort(), as found in almost all operating systems today, is based on that now very ancient V7 code.

- Bentley recommended that vendors should modify their qsort() implementations to perturb the input order so as to duck this aspect of the straightforward quicksort algorithm. Naturally,

this is a significant complication to an otherwise simple implementation.

It would appear that Sun and Microsoft have not taken this advice, but that HP has. Your example of a large array with only a few distinct values would be a pathological case in this context.

It should be clear why your experiment with adding the random offset helped: Your input is no longer essentially pre-sorted, and it no longer is made up of just a few distinct values.

ME
--

Thank you,

You say that you are recalling this from memory, but the description of the problem as you phrase it (slow when list is approximately sorted) does not necessarily match the cases in which I am seeing the problem. The slowness seems mostly due to the list having only a few distinct values, whether or not it is nearly sorted. If I make a list with just a few values that are completely randomly distributed, it still takes a very long time (at least on SUN and WIN2K running IDL 5.5).
So, perturbing the input order does not speed up the sort.

Example:

```
IDL> a = long(randomu(seed,400000) * 10)
IDL> b = sort(a)
```

RSI

You're welcome. I just wish I had a pointer to the actual article for you...

> You say that you are recalling this from memory...

Yes, from memory, and 10 year old memory at that. Don't put too much faith in it. You did mention a case in which you perturbed the input with a small random number and that the sort did speed up. This does match the Bentley paper...

Now that you raise the point, I have to agree that your results with a large array containing only a few distinct values is different than

the case I was describing, but I think it is plausible to think that there
might be a connection, or that it is a different qsort() pathology of the
same type.

The best thing you could do, if you had a service contract with Sun, would be to file a bug. Microsoft too, though I am less confident that they will do anything about it. Getting the vendor to fix this would be a great thing for anyone who uses that system.

Here is one fact that I am 100% sure of: IDL uses qsort(), and does not try to solve the sorting problem itself. It sounds tempting, but I'm sure
you can see the potential pitfalls. That's why we (presumably) have standard
libraries for such things...

...if you do find some evidence that it is IDL, and not qsort(), then I would definitely be interested in hunting that down. I can't see it though, as IDL's sort is a pretty think wrapper on top of qsort().

Subject: Re: Bizarre slowness from sort()
Posted by Paul Van Delst[1] on Tue, 23 Apr 2002 19:58:19 GMT
View Forum Message <> Reply to Message

Jonathan Joseph wrote:
>
> I had the following enlightening discussion about qsort with a
> programmer from RSI who responded to my post, but I'm still not
> entirely convinced. Can anyone comment on this?

From Jon Bentley's "Programming Pearls", 2nd Ed, pg 119 (describing qsorts and their efficiency):

"The qsort1 [code snippet in his book] function quickly sorts an array of random integers, but how does it perform on nonrandom inputs? ..... consider an extreme case: an array of n identical elements. .... the total run time is O(n^2). The run time for n=1000000 jumps from a second to two hours."

There is also discussion of a bunch of nifty things one can do to speed up sorting. The code snippets are only about 10-15 lines long so it would be not to difficult to translate from the C code into IDL (I think).

Probably worth it if you or your colleague will be sorting the offending type of input data.

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Bizarre slowness from sort()
Posted by Robert Stockwell on Tue, 23 Apr 2002 21:10:18 GMT
View Forum Message <> Reply to Message

Jonathan Joseph wrote:

> I had the following enlightening discussion about qsort with a
> programmer from RSI who responded to my post, but I'm still not
> entirely convinced. Can anyone comment on this?
>
> -Jonathan

... interesting stuff snipped....

The problem as mentioned sounds to me like they have
a 'gt', when they want a 'ge' in the piece of code where
they evaluate whether to move an element. (and maybe move
an element that is equal to the elements it is moving past)
:)

-bob

Subject: Re: Bizarre slowness from sort()
Posted by William Clodius on Tue, 23 Apr 2002 21:11:32 GMT
View Forum Message <> Reply to Message

Jonathan Joseph wrote:

> I had the following enlightening discussion about qsort with a
> programmer from RSI who responded to my post, but I'm still not
> entirely convinced. Can anyone comment on this?
>
> -Jonathan
>
> (posted with permission)<snip>

They should not be using quicksort in any form for integer or floating point data. The quoted average 0(n ln(n)) performance is not as good as the 0(n) that is achievable for these special data types. (Further this "average" has assumptions about the randomness of data that is often not obeyed in practice.) For other datatypes, e.g., strings, they should also not be using quicksort, as other sorting algorithms have the same 0(n ln(n)) asymptotic behavior and their relative scale factors depend on the relative costs of memory fetching, storing, and comparisons. Typical arguments favoring quicksort are based on values for these relative costs that were typical for integer and floating point data on machines of the late 70's/early 80's. See Richard O'Keefe's discussion of his applicative merge sort in a variety of newsgroups in the 90's.

Subject: Re: Bizarre slowness from sort()
Posted by thompson on Tue, 23 Apr 2002 22:42:33 GMT
View Forum Message <> Reply to Message

Jonathan Joseph <jj21@cornell.edu> writes:

> Hello,

> My colleague complained of an incredible slowness when trying to sort
> an array of long integers (on the order of 400,000 of them). I said
> "you're nuts. Must be a bug in your code" and proceeded to generate
> a random array of 400,000 long integers and sort them very quickly.
> "See, it works fine."

> So, he showed me his code, and it all looked perfectly normal, and
> the sort took minutes! The data looked fine (no bizarre values)
> so we created a save file, opened up a new IDL session tried to sort
> the data and saw the same slowness!

> I've found that the problem occurs on SUN and Windows 2K running IDL
> 5.5, but not on HP-UX running IDL 5.3. ...

Just to contribute to the platform comparisons:

```
IDL> print,!version
{ alpha OSF unix 5.4.1 Jan 16 2001      64      64}
IDL> i1=systime(1) & s=sort(sortme) & i2=systime(1) & print,i2-i1
      0.57970500
```

> ... Also, we have found
> a workaround for the integer case. Adding a small (less than 1) random
> offset to each element of the array before sorting will make it work
> quickly and yield the correct result. ...

Interestingly enough, on my computer this increased the time it took to do the sort

```
IDL> i1=systime(1) & s=sort(sortme2) & i2=systime(1) & print,i2-i1
       2.1228211
```

One interesting thing to do is to actually plot the sort indices

```
 IDL> plot,s,psym=3
```

On my computer one definitely gets a very organized pattern. Adding in the random number generator fuzzes this pattern out.

> ... But this will not work properly
> unless the array to be sorted is an integer type array, otherwise you
> could be changing the sort order by adding the random offset. Just
> converting the array to float or adding a constant offset to each
> element does NOT fix the problem.
>
> This behavior seems very strange - possibly a bug in IDL. Anyone
> have any thoughts on this? Can you reproduce this bug on your
> system?

> The save file is located at
> http://baritone.astro.cornell.edu/~jj/idl2/
> and is called 'sort.bin' (about 1.4 megabytes)

> IDL> restore, 'sort.bin'
> IDL> help, sortme

> SORTME   LONG   = Array[376467]

> IDL> a = sort(sortme)

> Works, but takes minutes to return.
> If I add a random number between 0 and 0.1 to each element
> and then sort, it works very rapidly (and produces the correct
> result since it is not changing the sorting order)

> IDL> b = sort(sortme + randomu(seed, n_elements(sortme)) * 0.1)

> Works very fast as expected

> Anyone know what's going on?

> Thanks.

> -Jonathan