Subject: Re: Randomize array order
Posted by Conor on Thu, 26 Jul 2007 16:22:27 GMT
View Forum Message <> Reply to Message

On Jul 26, 11:49 am, Allan Whiteford
<allan.rem...@phys.remove.strath.ac.remove.uk> wrote:
> hradilv wrote:
>> On Jul 26, 9:58 am, hradilv <hrad...@yahoo.com> wrote:
>
>>> On Jul 26, 8:40 am, Conor <cmanc...@gmail.com> wrote:
>
>>>> On Jul 26, 9:30 am, Allan Whiteford
>
>>>> <allan.rem...@phys.remove.strath.ac.remove.uk> wrote:
>
>>>> >Conor wrote:
>
>>>> >>Hi everyone!
>
>>>> >> Anyone know an efficient way to randomize an array (I have a
>>>> >>sorted array that I want unsorted). Initially, I tried something like
>>>> >>this:
>
>>>> >>array = findgen(1000000)
>>>> >>unsort = array[sort(randomu(seed,1000000))]
>
>>>> >>It works, but sorting on a million elements is rather slow. Anyone
>>>> >>know a faster way?
>
>>>> >Conor,
>
>>>> >Is it a million elements you want to do?
>
>>>> >The following scales better:
>
>>>> >pro shuffle,in
>>>> > b=long(n_elements(in)*randomu(seed,n_elements(in)))
>>>> > for i=0l,n_elements(in)-1 do begin
>>>> > tmp=in[i]
>>>> > in[i]=in[b[i]]
>>>> > in[b[i]]=tmp
>>>> > end
>>>> >end
>
>>>> >but on my machine, a million elements is around about where it starts to
>>>> >become as efficient as yours. For 10 million elements the above is a bit
>>>> >(17.05 seconds vs 12.92 seconds) but for 1 million elements they both
>>>> >come in at around 1.2 seconds (1.15 seconds vs 1.26 seconds). The above

>>>> >will scale as pretty much O(n) since it doesn't do any sorting but it
>>>> >takes a hit in the practical implementation because of the loop in
>>>> >IDL-space. Your suggestion will scale worse than O(n) but it seems the
>>>> >overlap in the two methods is exactly where you want to work.
>
>>>> >Maybe my loop can be made more efficient in practical terms but I don't
>>>> >think this is any better algorithm in terms of scaling (hard to imagine
>>>> >anything that could go faster than O(n) to randomise n things).
>
>>>> >Probably not helpful but I thought it was interesting that the
>>>> >cross-over is exactly where you want to work. But, maybe I should get
>>>> >out more if I think that's especially interesting.
>
>>>> >Thanks,
>
>>>> >Allan
>
>>>> Thanks for the suggestions guys! I'll have to play around and see
>>>> what works best.
>
>>> Here's a table of results from my machine. All times are in seconds.
>>> PC single processor, WinXP, IDL6.4
>
>>>	i	Niter	Rand-meth	Loop-meth
>>>	0	100000	0.0929999	0.110000
>>>	1	166810	0.0779998	0.0940001
>>>	2	278256	0.140000	0.157000
>>>	3	464158	0.297000	0.297000
>>>	4	774263	0.578000	0.562000
>>>	5	1291549	1.09400	0.890000
>>>	6	2154435	2.06300	1.48400
>>>	7	3593812	3.84400	2.56300
>>>	8	5994841	7.09400	4.31300
>>>	9	10000000	13.0470	7.29800

>
>> More details: Single Intel 1.86GHz, 2Gb RAM
>
>> Other machine: Sun Blade 2500 - Solaris 9, IDL 6.3 - Dual processor,
>> 2Gb RAM
>

>>	i	Niter	Rand-meth	Loop-meth
>>	0	100000	0.112775	0.218330
>>	1	166810	0.194601	0.370555
>>	2	278256	0.369679	0.621675
>>	3	464158	0.700207	1.05355
>>	4	774263	1.32646	1.74441
>>	5	1291549	2.42519	2.95356
>>	6	2154435	4.38822	4.91093

>>   7   3593812   8.63800   8.35843
>>   8   5994841   15.6409   13.9243
>>   9   10000000   28.9150   23.6173
>
>> Interesting, there's a crossover at ~ 3,000,000 where the loop method
>> starts to win.
>
> Here's what I get on a dual core 3GHz Pentium 4 with 2GB of RAM running
> Linux (FC4) using IDL6.2:
>
>   i   Niter   Rand-meth   Loop-meth
>   0   100000   0.0818000   0.120713
>   1   166810   0.140054   0.205111
>   2   278256   0.255531   0.340111
>   3   464158   0.462941   0.572567
>   4   774263   0.835279   0.973762
>   5   1291549   1.53649   1.71803
>   6   2154435   3.08281   2.83829
>   7   3593812   5.27431   4.71084
>   8   5994841   10.6316   7.85549
>   9   10000000   17.4706   13.6622
>
> kind of annoying that your 1.8GHz machine running windows goes faster
> than my 3GHz running Linux. Not as bad as how slow the Sun goes though.
>
> Incidentally, previously I was quoting raw CPU times rather than the
> wall clock times which your routine prints out.
>
> Thanks,
>
> Allan

Here's what I get running it on my super old computer:

        0   100000   0.231639   0.266472
        1   166810   0.429814   0.450388
        2   278256   0.768671   0.777250
        3   464158   1.40014   1.29011
        4   774263   2.55367   2.15114
        5   1291549   4.66570   3.60980
        6   2154435   8.48878   6.04430
        7   3593812   15.3753   10.1437
        8   5994841   29.2131   20.1072
        9   10000000   52.2718   29.7969