Subject: IDL program runs faster on slower CPU
Posted by Deckard++; on Sun, 16 Mar 2014 22:15:04 GMT
View Forum Message <> Reply to Message

Hi,

I have this very strange performance problem that I can't quite figure out. I have a rather complex minimization problem for which I use the MPFIT library. I run my code on two machines:

- MacBook Pro with 4 CPU Intel Core i7 @ 2.3 GHz, 8 GB of RAM, IDL 7.1.1, Mac OS 10.9.2
- Dell workstation with 12 CPU Intel Xeon @ 2.9 GHz, 48 GB of RAM, ILD 8.2, Scientific Linux 6.5

Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.

The only difference that I see is the IDL version, but I wouldn't expect this to be the problem. I assume it is a more complicated problem, but I have no precise idea of what it is. Any advice on this subject? Thanks a lot for you help.

Cheers,

-- Arthur;

--
Arthur Vigan
Laboratoire d'Astrophysique de Marseille

Subject: Re: IDL program runs faster on slower CPU
Posted by Craig Markwardt on Mon, 17 Mar 2014 17:33:27 GMT
View Forum Message <> Reply to Message

[I'm having lots of trouble posting from Google Groups the past few days.]

On Sunday, March 16, 2014 6:15:04 PM UTC-4, Arthur Vigan wrote:
> Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.

I would suggest using PROFILER to find out where the bottleneck is. My first guess is you have some numerical faults like NaNs which are being handled by the two platforms differently. Sometimes numerical exceptions are handled very slowly because they take a round trip to the kernel.

Craig

Subject: Re: IDL program runs faster on slower CPU
Posted by Deckard++; on Tue, 18 Mar 2014 12:22:44 GMT
View Forum Message <> Reply to Message

Le lundi 17 mars 2014 18:33:27 UTC+1, Craig Markwardt a écrit :
> [I'm having lots of trouble posting from Google Groups the past few days.]
>
Me too!

> On Sunday, March 16, 2014 6:15:04 PM UTC-4, Arthur Vigan wrote:
>
>> Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.
>
>
>
> I would suggest using PROFILER to find out where the bottleneck is. My first guess is you have some numerical faults like NaNs which are being handled by the two platforms differently. Sometimes numerical exceptions are handled very slowly because they take a round trip to the kernel.

It could be a possibility, but after running the profiler I am not so sure. The profiler output seems to show very mixed results, with some things running faster on the Mac, and some things running faster on the server. But more generally, any complex program seems to run faster on the Mac

I just ran a simple test on the two machines:

```
Profiler,/SYSTEM & Profiler
t0 = systime(/sec)
a = replicate(!dpi,10000,10000)
e = exp(a)
l = alog(a)
s = sin(a)
c = cos(a)
print,systime(/sec)-t0
Profiler,/REPORT
```

On the Mac:

Module	Type	Count	Only(s)	Avg.(s)	Time(s)	Avg.(s)
ALOG	(S)	1	1.661171	1.661171	1.661171	1.661171
COS	(S)	1	1.357197	1.357197	1.357197	1.357197
EXP	(S)	1	0.541147	0.541147	0.541147	0.541147
PRINT	(S)	1	0.111763	0.111763	0.111763	0.111763
PROFILER	(S)	1	0.000024	0.000024	0.000024	0.000024
REPLICATE	(S)	1	0.119898	0.119898	0.119898	0.119898
SIN	(S)	1	1.213518	1.213518	1.213518	1.213518

SYSTIME (S) 2 0.000008 0.000004 0.000008 0.000004

On the Linux server:

Module	Type	Count	Only(s)	Avg.(s)	Time(s)	Avg.(s)
ALOG	(S)	1	0.729371	0.729371	0.729371	0.729371
COS	(S)	1	0.721096	0.721096	0.721096	0.721096
EXP	(S)	1	0.635157	0.635157	0.635157	0.635157
PRINT	(S)	1	0.000040	0.000040	0.000040	0.000040
PROFILER	(S)	1	0.000014	0.000014	0.000014	0.000014
REPLICATE	(S)	1	0.137368	0.137368	0.137368	0.137368
SIN	(S)	1	4.108430	4.108430	4.108430	4.108430
SYSTIME	(S)	2	0.000006	0.000003	0.000006	0.000003

Some basic functions seem to run much faster on linux (alog, cos), while others run faster on the Mac (sin, replicate). I really don't understand...

-- Arthur;

Subject: Re: IDL program runs faster on slower CPU
Posted by Jim Pendleton on Tue, 18 Mar 2014 21:40:55 GMT
View Forum Message <> Reply to Message

On Tuesday, March 18, 2014 6:22:44 AM UTC-6, Arthur Vigan wrote:
> Le lundi 17 mars 2014 18:33:27 UTC+1, Craig Markwardt a écrit :
>
>> [I'm having lots of trouble posting from Google Groups the past few days.]
>
>>
>
> Me too!
>
>
>
>> On Sunday, March 16, 2014 6:15:04 PM UTC-4, Arthur Vigan wrote:
>
>>
>
>>> Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.
>
>>
>
>>
>

>>
>
>> I would suggest using PROFILER to find out where the bottleneck is. My first guess is you have some numerical faults like NaNs which are being handled by the two platforms differently. Sometimes numerical exceptions are handled very slowly because they take a round trip to the kernel.
>
>
>
> It could be a possibility, but after running the profiler I am not so sure. The profiler output seems to show very mixed results, with some things running faster on the Mac, and some things running faster on the server. But more generally, any complex program seems to run faster on the Mac
>
>
>
> I just ran a simple test on the two machines:
>
>
>
> Profiler,/SYSTEM & Profiler
>
> t0 = systime(/sec)
>
> a = replicate(!dpi,10000,10000)
>
> e = exp(a)
>
> l = alog(a)
>
> s = sin(a)
>
> c = cos(a)
>
> print,systime(/sec)-t0
>
> Profiler,/REPORT
>
>
>
> On the Mac:
>
>
>
> Module Type Count Only(s) Avg.(s) Time(s) Avg.(s)
>
> ALOG (S) 1 1.661171 1.661171 1.661171 1.661171
>
> COS (S) 1 1.357197 1.357197 1.357197 1.357197

>
> EXP (S) 1 0.541147 0.541147 0.541147 0.541147
>
> PRINT (S) 1 0.111763 0.111763 0.111763 0.111763
>
> PROFILER (S) 1 0.000024 0.000024 0.000024 0.000024
>
> REPLICATE (S) 1 0.119898 0.119898 0.119898 0.119898
>
> SIN (S) 1 1.213518 1.213518 1.213518 1.213518
>
> SYSTIME (S) 2 0.000008 0.000004 0.000008 0.000004
>
>
>
> On the Linux server:
>
>
>
> Module Type Count Only(s) Avg.(s) Time(s) Avg.(s)
>
> ALOG (S) 1 0.729371 0.729371 0.729371 0.729371
>
> COS (S) 1 0.721096 0.721096 0.721096 0.721096
>
> EXP (S) 1 0.635157 0.635157 0.635157 0.635157
>
> PRINT (S) 1 0.000040 0.000040 0.000040 0.000040
>
> PROFILER (S) 1 0.000014 0.000014 0.000014 0.000014
>
> REPLICATE (S) 1 0.137368 0.137368 0.137368 0.137368
>
> SIN (S) 1 4.108430 4.108430 4.108430 4.108430
>
> SYSTIME (S) 2 0.000006 0.000003 0.000006 0.000003
>
>
>
> Some basic functions seem to run much faster on linux (alog, cos), while others run faster on the Mac (sin, replicate). I really don't understand...
>
>
>
> -- Arthur;

What are the outputs on both machines from
IDL> help, /str, !cpu

Unless you've changed defaults, you have enough elements in your array (1.e8) that the thread pool should be kicking in for the vectorized functions.

Any chance you have one or more process limits on your Linux account that could cause memory to be paged?

Jim P.

Subject: Re: IDL program runs faster on slower CPU
Posted by Craig Markwardt on Wed, 19 Mar 2014 01:47:56 GMT
View Forum Message <> Reply to Message

On Tuesday, March 18, 2014 8:22:44 AM UTC-4, Arthur Vigan wrote:
> Le lundi 17 mars 2014 18:33:27 UTC+1, Craig Markwardt a écrit :
>> On Sunday, March 16, 2014 6:15:04 PM UTC-4, Arthur Vigan wrote:
>>
>>> Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.
>>
>> I would suggest using PROFILER to find out where the bottleneck is. My first guess is you have some numerical faults like NaNs which are being handled by the two platforms differently. Sometimes numerical exceptions are handled very slowly because they take a round trip to the kernel.
>
>
>
> It could be a possibility, but after running the profiler I am not so sure. The profiler output seems to show very mixed results, with some things running faster on the Mac, and some things running faster on the server. But more generally, any complex program seems to run faster on the Mac
>
>
>
> I just ran a simple test on the two machines:
...

It's still worth it to check your actual problem, not a simple test.

CM

Subject: Re: IDL program runs faster on slower CPU
Posted by Deckard++; on Wed, 19 Mar 2014 08:45:23 GMT
View Forum Message <> Reply to Message

Le mercredi 19 mars 2014 02:47:56 UTC+1, Craig Markwardt a écrit :
> On Tuesday, March 18, 2014 8:22:44 AM UTC-4, Arthur Vigan wrote:
>
>> Le lundi 17 mars 2014 18:33:27 UTC+1, Craig Markwardt a écrit :
>
>>> On Sunday, March 16, 2014 6:15:04 PM UTC-4, Arthur Vigan wrote:
>
>>>
>
>>>> Strangely, the code runs about 4 to 5 times faster on the MacBook Pro. The code is strictly identical, with the same starting point, and it finds exactly the same result in the same number of iterations. I also mention that the program does not rely on disk access that could slow things down: all the data is in memory.
>
>>>
>
>>> I would suggest using PROFILER to find out where the bottleneck is. My first guess is you have some numerical faults like NaNs which are being handled by the two platforms differently. Sometimes numerical exceptions are handled very slowly because they take a round trip to the kernel.
>
>>
>
>>
>
>>
>
>> It could be a possibility, but after running the profiler I am not so sure. The profiler output seems to show very mixed results, with some things running faster on the Mac, and some things running faster on the server. But more generally, any complex program seems to run faster on the Mac
>
>>
>
>>
>
>>
>
>> I just ran a simple test on the two machines:
>
> ...
>
>
>
> It's still worth it to check your actual problem, not a simple test.
>
>
>

> CM

Yes, true. I have investigated a bit more by placing systime(/sec) commands around different portions of my code, and the result is what I was saying above: every chunk of code seems to run slower on the linux workstation.

Concerning the numerical faults that you were mentioning, do you know if there is any way to check if there are any? Are they reported somewhere by the system, triggering some kind of message?

-- Arthur;

Subject: Re: IDL program runs faster on slower CPU
Posted by Craig Markwardt on Wed, 19 Mar 2014 17:45:41 GMT
View Forum Message <> Reply to Message

On Wednesday, March 19, 2014 4:45:23 AM UTC-4, Arthur Vigan wrote:
> Le mercredi 19 mars 2014 02:47:56 UTC+1, Craig Markwardt a écrit :
> Concerning the numerical faults that you were mentioning, do you know if there is any way to check if there are any? Are they reported somewhere by the system, triggering some kind of message?

!EXCEPT=2 is the usual method.

Craig

Subject: Re: IDL program runs faster on slower CPU
Posted by Deckard++; on Wed, 19 Mar 2014 19:47:25 GMT
View Forum Message <> Reply to Message

I am getting a floating underflow error:

% Program caused arithmetic error: Floating underflow

It is generated by a call to la_eigenql() that performs some singular value decomposition to perform principal component analysis. I had already tried with the eigenql() function but that did not influence the performance.

I had come to believe that such underflow errors don't really matter, but maybe I am wrong?

-- Arthur;

Subject: Re: IDL program runs faster on slower CPU

Posted by Craig Markwardt on Wed, 19 Mar 2014 23:06:59 GMT
View Forum Message <> Reply to Message

On Wednesday, March 19, 2014 3:47:25 PM UTC-4, Arthur Vigan wrote:
> % Program caused arithmetic error: Floating underflow
>
> It is generated by a call to la_eigenql() that performs some singular value decomposition to perform principal component analysis. I had already tried with the eigenql() function but that did not influence the performance.
>
> I had come to believe that such underflow errors don't really matter, but maybe I am wrong?

Underflows are usually not a problem. But you can test whether EIGENQL is the bottleneck or not.