Subject: Re: Some questions of efficiency when matching items in lists
Posted by Craig Markwardt on Fri, 01 Feb 2013 00:59:16 GMT
View Forum Message <> Reply to Message

On Thursday, January 31, 2013 7:05:40 PM UTC-5, Bogdanovist wrote:
> I have a couple of questions about how to efficiently match items in lists. There are two operations that are done many thousands of times in my processing and are causing a bottleneck. Even small improvements would be welcome.

For your first question, the obvious thing is that you are reading and writing a file for every operation. File I/O is slow. If you can, keep your values in memory and only write them out when necessary. At the very least, only rewrite your file when you have to; otherwise just /APPEND to the end.

For your second question: if DATA_ADD has a small number of elements then you are probably doing the best that you can do. You might check out MATCH or MATCH2 in the IDL astronomy library, which have been optimized for cross-matching a lot of elements. Another possibility is to create a hash table indexed by time; this has the benefit of rapid access, but you lose the ability to perform vector operations upon the whole table.

Craig

Subject: Re: Some questions of efficiency when matching items in lists
Posted by Matt Francis on Fri, 01 Feb 2013 01:15:19 GMT
View Forum Message <> Reply to Message

On Friday, 1 February 2013 11:59:16 UTC+11, Craig Markwardt wrote:
> On Thursday, January 31, 2013 7:05:40 PM UTC-5, Bogdanovist wrote:
>
>> I have a couple of questions about how to efficiently match items in lists. There are two operations that are done many thousands of times in my processing and are causing a bottleneck. Even small improvements would be welcome.
>
>
>
>
>
> For your first question, the obvious thing is that you are reading and writing a file for every operation. File I/O is slow. If you can, keep your values in memory and only write them out when necessary. At the very least, only rewrite your file when you have to; otherwise just /APPEND to the end.
>
>
>
> For your second question: if DATA_ADD has a small number of elements then you are probably doing the best that you can do. You might check out MATCH or MATCH2 in the IDL

astronomy library, which have been optimized for cross-matching a lot of elements. Another possibility is to create a hash table indexed by time; this has the benefit of rapid access, but you lose the ability to perform vector operations upon the whole table.
>
>
>
> Craig

Thanks for the info, unfortunately I can't store the values in memory as each write to file occurs during a separate 'run' of the processing software, which is fired off regularly from the cron (linux machine).

For the second part I'll checkout MATCH and MATCH2, sounds like I'll have to profile carefully to make sure it's quicker. Indeed data_add has typically less than 100 elements, so maybe it will be no quicker. Thanks again.

Subject: Re: Some questions of efficiency when matching items in lists
Posted by on Fri, 01 Feb 2013 07:55:18 GMT
View Forum Message <> Reply to Message

Den fredagen den 1:e februari 2013 kl. 02:15:19 UTC+1 skrev Bogdanovist:
> On Friday, 1 February 2013 11:59:16 UTC+11, Craig Markwardt wrote:
>> On Thursday, January 31, 2013 7:05:40 PM UTC-5, Bogdanovist wrote:
>>> I have a couple of questions about how to efficiently match items in lists. There are two operations that are done many thousands of times in my processing and are causing a bottleneck. Even small improvements would be welcome.
>
>> For your first question, the obvious thing is that you are reading and writing a file for every operation. File I/O is slow. If you can, keep your values in memory and only write them out when necessary. At the very least, only rewrite your file when you have to; otherwise just /APPEND to the end.
>
>> For your second question: if DATA_ADD has a small number of elements then you are probably doing the best that you can do. You might check out MATCH or MATCH2 in the IDL astronomy library, which have been optimized for cross-matching a lot of elements. Another possibility is to create a hash table indexed by time; this has the benefit of rapid access, but you lose the ability to perform vector operations upon the whole table.
>
>> Craig
>
>
>
> Thanks for the info, unfortunately I can't store the values in memory as each write to file occurs during a separate 'run' of the processing software, which is fired off regularly from the cron (linux machine).

Maybe you could append both new data and corrections and then deal with the corrections

properly later, in the collating phase. That should help with the bottle neck in the near real time part.

Subject: Re: Some questions of efficiency when matching items in lists
Posted by on Fri, 01 Feb 2013 12:27:25 GMT
View Forum Message <> Reply to Message

Den fredagen den 1:e februari 2013 kl. 08:55:18 UTC+1 skrev Mats Löfdahl:
> Den fredagen den 1:e februari 2013 kl. 02:15:19 UTC+1 skrev Bogdanovist:
>
>> On Friday, 1 February 2013 11:59:16 UTC+11, Craig Markwardt wrote:
>
>>> On Thursday, January 31, 2013 7:05:40 PM UTC-5, Bogdanovist wrote:
>
>>>> I have a couple of questions about how to efficiently match items in lists. There are two operations that are done many thousands of times in my processing and are causing a bottleneck. Even small improvements would be welcome.
>
>>
>
>>> For your first question, the obvious thing is that you are reading and writing a file for every operation. File I/O is slow. If you can, keep your values in memory and only write them out when necessary. At the very least, only rewrite your file when you have to; otherwise just /APPEND to the end.
>
>>
>
>>> For your second question: if DATA_ADD has a small number of elements then you are probably doing the best that you can do. You might check out MATCH or MATCH2 in the IDL astronomy library, which have been optimized for cross-matching a lot of elements. Another possibility is to create a hash table indexed by time; this has the benefit of rapid access, but you lose the ability to perform vector operations upon the whole table.
>
>>
>
>>> Craig
>
>>
>
>>
>
>>
>
>> Thanks for the info, unfortunately I can't store the values in memory as each write to file occurs during a separate 'run' of the processing software, which is fired off regularly from the cron (linux machine).
>

>
>
> Maybe you could append both new data and corrections and then deal with the corrections properly later, in the collating phase. That should help with the bottle neck in the near real time part.

Or instead of writing a single file, write all the items in files named as their time stamps. Then corrections will happen automatically as older files are overwritten. And then the collating could proceed much like you do it currently, except from these individual files.
