Subject: h5_parse() in the profiler
Posted by MarioIncandenza on Fri, 08 Jul 2016 16:50:13 GMT
View Forum Message <> Reply to Message

Hi IDL Wizards,

I'm working on an application requiring chunking through a huge quantity of HDF5 files. For (EXTREME) ease of coding, my code does
IDL> H5DATA = H5_PARSE(HDF5_file,/READ_DATA)
, and then operates on H5DATA. So so easy to code, but that call to H5_PARSE() is very time-consuming. I ran the IDL Profiler (as elegantly described here: http://www.idlcoyote.com/code_tips/whyslow.html), and found that all the time was being spent in two routines:

Routine	Calls	Only		Total	
CREATE_STRUCT (S)	1320	61.130619	0.046311	61.130619	0.046311
H5D_READ (S)	92	53.353344	0.579928	53.353344	0.579928

The 'H5D_READ' I understand, that is the low-level I/O and it is constrained by the system. But the 'CREATE_STRUCT' surprised me.

I guess CREATE_STRUCT() is where the memory allocation is occurring, but does it seem right that this takes more time than the actual disk I/O?

Any insights are welcome. I could rewrite the code to pull specific data out of the HDF5 file by hand, but that would be hundreds of lines of code, and I'd really rather not...

--Edward H.

Subject: Re: h5_parse() in the profiler
Posted by Markus Schmassmann on Fri, 08 Jul 2016 17:46:59 GMT
View Forum Message <> Reply to Message

On 07/08/2016 06:50 PM, Edward Hyer wrote:
> Hi IDL Wizards,
>
> I'm working on an application requiring chunking through a huge
> quantity of HDF5 files. For (EXTREME) ease of coding, my code does
> IDL> H5DATA = H5_PARSE(HDF5_file,/READ_DATA)
> , and then operates on H5DATA. So so easy to code, but that call to
> H5_PARSE() is very time-consuming. I ran the IDL Profiler (as
> elegantly described here:
> http://www.idlcoyote.com/code_tips/whyslow.html), and found that all
> the time was being spent in two routines:
> Routine Calls Only Total
> CREATE_STRUCT (S) 1320 61.130619 0.046311 61.130619
> 0.046311 H5D_READ (S) 92 53.353344 0.579928
> 53.353344 0.579928

>
> The 'H5D_READ' I understand, that is the low-level I/O and it is
> constrained by the system. But the 'CREATE_STRUCT' surprised me.
>
> I guess CREATE_STRUCT() is where the memory allocation is occurring,
> but does it seem right that this takes more time than the actual disk
> I/O?
>
> Any insights are welcome. I could rewrite the code to pull specific
> data out of the HDF5 file by hand, but that would be hundreds of
> lines of code, and I'd really rather not...
>
> --Edward H.
>
create_struct is called much more often, possibly - without looking into
h5d_read - the struct is being created like that:
 temp=[]
 for i=1,n-1 do temp=create_struct(temp,tagname[i],tagvalue[i])
 struct=temp
terribly inefficient, better to create a string and then use
 execute(string)

--Markus Schmassmann, IDL wizard apprentice - at best ;-)

Subject: Re: h5_parse() in the profiler
Posted by MarioIncandenza on Sat, 09 Jul 2016 04:38:20 GMT
View Forum Message <> Reply to Message

On Friday, July 8, 2016 at 10:47:02 AM UTC-7, Markus Schmassmann wrote:
> for i=1,n-1 do temp=create_struct(temp,tagname[i],tagvalue[i])
> terribly inefficient, better to create a string and then use
> execute(string)

Hmmm... Yes! EXECUTE() is a non-starter, this needs to be fully usable in compiled code. But I'm sure there is some clever way to do this with fewer calls to CREATE_STRUCT().
If I come up with something that actually is faster, I'll post to this thread.

Subject: Re: h5_parse() in the profiler
Posted by Jim Pendleton on Sat, 09 Jul 2016 14:05:11 GMT
View Forum Message <> Reply to Message

On Friday, July 8, 2016 at 10:38:22 PM UTC-6, Edward Hyer wrote:
> On Friday, July 8, 2016 at 10:47:02 AM UTC-7, Markus Schmassmann wrote:
>> for i=1,n-1 do temp=create_struct(temp,tagname[i],tagvalue[i])
>> terribly inefficient, better to create a string and then use

>> execute(string)
>
> Hmmm... Yes! EXECUTE() is a non-starter, this needs to be fully usable in compiled code. But I'm sure there is some clever way to do this with fewer calls to CREATE_STRUCT().
> If I come up with something that actually is faster, I'll post to this thread.

Have you tried returning an ordered hash instead?

Each structure array in IDL represents a chunk of contiguous memory. That is, each of the consecutive tags in the structure is consecutive in memory, with some redirection for items such as strings. The nested calls to CREATE_STRUCT will be much like an array append operation, a = [a, newstuff], which can become quite inefficient for large arrays due to the need to make a new copy of the data at each iteration.

By using the /ORDEREDHASH keyword to H5_READ (added in 2014), the storage of the individual values is not restricted to contiguous memory and the overhead of recursive copying is no longer present.

Jim P.
