Subject: Re: float function unexpectedly slow
Posted by Heinz Stege on Mon, 31 Mar 2014 23:48:45 GMT
View Forum Message <> Reply to Message

Hi all,

I know, I'm a little late with this message. When I read Tim's posting, I had the idea to write my own system routine for converting a string to float. And this idea I didn't get out of my mind. So here is the story.

I'm not an expert in C. To tell the truth, I'm just a beginner. There may be things in my C-code, which are not good. (Comments, which help me learning, are welcome.) I'll place the code at the bottom of this posting.

First let me give the results. str_to_double(), this is the new system routine, makes about 50% more conversions per time than IDL's double().

The following commands
```
  str='363.491'
  t0=systime(1) &for i=0,9999999 do x=double(str) &print,systime(1)-t0
```
take 5.90 s. The same with x=str_to_double(str) instead of x=double(str) takes 3.81 s. I ran the commands several times in a changing order.

I think, Chris is absolutely right with his recommendation to eliminate any loops in the IDL program. So I did another test with an array of strings:
```
  str=strtrim(randomu(seed,1000000)*1000.,2)
  t0=systime(1) &for i=0,9 do x=double(str) &print,systime(1)-t0
```
This takes 5.35 s for double() and 3.60 s for str_to_double. This is very similar to the result with the scalar string.

Interesting, that in this test the loop does not cost much CPU time. The conversion of 1 million scalar strings do not need much more time than one array with 1 milionen elements. However str_to_double() is significantly faster than double().

The tests are done with version { x86 Win32 Windows Microsoft Windows 8.0.1 Oct 5 2010   32   64}.

Cheers, Heinz

Here ist the C code:

```
static IDL_VPTR str_to_double(int argc,IDL_VPTR *argv)
{
   IDL_VPTR result;

   if (argv[0]->type != IDL_TYP_STRING) {

 IDL_MessageFromBlock(msg_block,M_MSR_NO_STRING_TYPE,IDL_MSG_
LONGJMP,argv[0]- >type);
   }

   if (argv[0]->flags & IDL_V_ARR) {
      IDL_STRING *str;
      double *vector;
      IDL_MEMINT i;

      str=(IDL_STRING *) argv[0]->value.arr->data;
      vector=(double *) IDL_MakeTempArray(IDL_TYP_DOUBLE,
                                          argv[0]->value.arr->n_dim,

argv[0]->value.arr->dim,IDL_ARR_INI_NOP,&result);
      for (i=0; i<argv[0]->value.arr->n_elts; i++) {
         vector[i]=str[i].slen? atof(str[i].s) : 0.;
      }
   }
   else {
      double value;

      value=argv[0]->value.str.slen? atof(argv[0]->value.str.s) : 0.;
      result=IDL_GettmpDouble(value);
   }

   return result;
}
```