Subject: 6.3 reactions?
Posted by Richard French on Tue, 02 May 2006 03:31:37 GMT
View Forum Message <> Reply to Message

To those of you who have gotten IDL 6.3 up and running, what are the new
features that those of us still using 6.2 are missing? I watched the RSI 5
minute video but I couldn't understand most of the acronyms, so I think they
have a different target audience in mind.
Thanks,
Dick French

Subject: Re: 6.3 reactions?
Posted by R.Bauer on Thu, 04 May 2006 08:26:38 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
> Michael Galloy wrote:
>
>> I'm pretty excited to see whether or not Motion JPEG 2000 is as great
>> for animations as it seems at first glance. MPEG has always seemed to
>> be a bad format for scientific visualization; Motion JPEG 2000
>> corrects a lot of its weaknesses.
>
>
> I did some quick experiments: Files take a long time to write (with
> default parameters). I haven't found anything (other than IDL and a few
> high-end video editing programs) that will read them.
>
> In my experience the best format for scientific animations by far is AVI
> (Microsoft Video 1 codec, set quality to a highish value like 85%).
>
>

I do prefer flash swf because you could add text with it's font

e.g. http://swftools.org

cheers
Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

--- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
== =======

Subject: Re: 6.3 reactions?
Posted by Haje Korth on Thu, 04 May 2006 16:28:53 GMT
View Forum Message <> Reply to Message

Mark,
I played with the MJPEG2000 quite a bit during the beta and was actually impressed with it. The trick is to modify the default quality settings. I coul reduce the quality significantly without seeing any difference using the option BIT_RATE=0.5. The resulting files have a slightly smaller file size as those produced by the VP3 codec at comparable quality.

The files can be read with Windows Media Player using the Morgan MJPEG2000 codec. However, it is not available for Quicktime, which is my preferred player. IDL has a lpayer built in, but of course this is impractical for presentations at conferences.

Just a thought, if we had native AVI support in IDL, we could use the Morgan codec directly! Given Oleg's example, that would have probably been quicker than coding the idlffmjpeg2000 object, but it would be platform specific of course, which is exactly what IDL is not supposed to be. Nevertheless, I want this AVI support. There are some things that are better on Windows, which even the biggest UN*X fan can't deny! :-)

Cheers,
Haje

Subject: Re: 6.3 reactions?
Posted by codepod on Fri, 05 May 2006 14:25:57 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Tue, 02 May 2006 05:52:53 +0000, Michael Galloy wrote:

>> Everyone will have their own favorites, but I think the IDL-IDL bridge
>> will be a feature that really adds something to IDL. It will allow you
>> to start new "threads". So, for example, to do some processing while the
>> controls of your GUI are still responsible to user interaction. Or
>> simply to farm out several different tasks to run independently.

>
> Does this really start an entirely new IDL process, or is there some
> kind of lightweight user-accessible threading support now available?
> I'm imagining the kind of havoc which could result if multiple user
> threads vie for window ids and other common resources.
>
For each IDL-IDL Bridge object you create, a child process of your IDL
session is created and it is executing its own interpreter. IDL
commands are dispatched in either a synchronous or asynchronous mode
(which will trigger an event when completed). While a simple interface
is presented to exchange data between the processes, internally shared
memory is used so the transfer is fairly efficeint.

As Mike pointed out in his original post, this is a great solution to
perform background processing while maintaining an interactive user
interface. Also it provides a quick method to create a "clean" idl
session to generate save files and the like. And with the proliferation
of multi-core processors, the bridge provides the IDL user a simple way
to exploit some of the capabilities of modern hardware.

- CP

Subject: Re: 6.3 reactions?
Posted by JD Smith on Fri, 05 May 2006 16:24:17 GMT
View Forum Message <> Reply to Message

On Fri, 05 May 2006 07:25:57 -0700, codepod wrote:

> JD Smith wrote:
>> On Tue, 02 May 2006 05:52:53 +0000, Michael Galloy wrote:
>
>>> Everyone will have their own favorites, but I think the IDL-IDL bridge
>>> will be a feature that really adds something to IDL. It will allow you
>>> to start new "threads". So, for example, to do some processing while the
>>> controls of your GUI are still responsible to user interaction. Or
>>> simply to farm out several different tasks to run independently.
>>
>> Does this really start an entirely new IDL process, or is there some
>> kind of lightweight user-accessible threading support now available?
>> I'm imagining the kind of havoc which could result if multiple user
>> threads vie for window ids and other common resources.
>>
> For each IDL-IDL Bridge object you create, a child process of your IDL
> session is created and it is executing its own interpreter. IDL
> commands are dispatched in either a synchronous or asynchronous mode
> (which will trigger an event when completed). While a simple interface
> is presented to exchange data between the processes, internally shared

> memory is used so the transfer is fairly efficeint.
>
> As Mike pointed out in his original post, this is a great solution to
> perform background processing while maintaining an interactive user
> interface. Also it provides a quick method to create a "clean" idl
> session to generate save files and the like. And with the proliferation
> of multi-core processors, the bridge provides the IDL user a simple way
> to exploit some of the capabilities of modern hardware.

Very interesting. Since it uses shared memory, can you inherit from
the IDL_IDLBridge class, and have the entire object structure
modifiable from both processes at the same time (which of course would
have its dangers if not orchestrated carefully)? I also take it that
all of the color/window/file handle/etc. resources would be completely
separate, not shared.

JD

Subject: Re: 6.3 reactions?
Posted by David Fanning on Fri, 05 May 2006 16:55:10 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> Very interesting. Since it uses shared memory, can you inherit from
> the IDL_IDLBridge class, and have the entire object structure
> modifiable from both processes at the same time (which of course would
> have its dangers if not orchestrated carefully)? I also take it that
> all of the color/window/file handle/etc. resources would be completely
> separate, not shared.

Uh, JD, this is IDL we are talking about. I think you
must be thinking about version 5.0 of the IDL Bridge. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: 6.3 reactions?
Posted by codepod on Mon, 08 May 2006 14:29:54 GMT
View Forum Message <> Reply to Message

JD, the implementation is simpler than what you outline. The object is
only present on the "parent" process, so the actual object instance
data is not shared between the processes. The child process has no
indication it's being run as a worker: it just receives commands and
executes them. IDL variables are transferred between the processes by
the bridge using shared memory. The parent process can Get or Set
variables in the child process. Overall it's a very simple interface
that brings a powerful new dimension to IDL applications.

Also as you mention, this is an independent process so resources are
not shared between them.

The underlying process management technology used in the bridge is
actually used in several places currently in IDL (Export Bridges,
LabView script node) and will enable some amazing new technology that
should be shown later this year.

-CP

Subject: Re: 6.3 reactions?
Posted by JD Smith on Mon, 08 May 2006 20:25:11 GMT
View Forum Message <> Reply to Message

On Mon, 08 May 2006 07:29:54 -0700, codepod wrote:

> JD, the implementation is simpler than what you outline. The object is
> only present on the "parent" process, so the actual object instance
> data is not shared between the processes. The child process has no
> indication it's being run as a worker: it just receives commands and
> executes them. IDL variables are transferred between the processes by
> the bridge using shared memory. The parent process can Get or Set
> variables in the child process. Overall it's a very simple interface
> that brings a powerful new dimension to IDL applications.

Thanks for the info. Does Get/Set actually copy variables, or
just provide references in shared memory (which of course would require
IDL setting up the existing heap for sharing)? That would allow you to
Get/Set self in an object, and have it operated on independently in both
sides.

JD

Subject: Re: 6.3 reactions?
Posted by Richard French on Tue, 09 May 2006 04:23:57 GMT
View Forum Message <> Reply to Message

I hope that someone who really understands all of this IDL Bridge stuff and
who doesn't use objects for every programming task will provide a simple
description for civilians of just what this capability can do for the
everyday user, and what it can't do. I'm having trouble figuring out whether
this is a cool feature that I'd be crazy not to employ, an easy way to try
to do parallel processing, or something that would require me to reprogram
everything I do to obtain the leverage from multiple processors. So, all of
you Ernest Hemingways out there, how about some simple, declarative
sentences with an example or two of how this feature works and why it is a
Good Thing. Many of us will be grateful!

Dick French

> On Mon, 08 May 2006 07:29:54 -0700, codepod wrote:
>
>> JD, the implementation is simpler than what you outline. The object is
>> only present on the "parent" process, so the actual object instance
>> data is not shared between the processes. The child process has no
>> indication it's being run as a worker: it just receives commands and
>> executes them. IDL variables are transferred between the processes by
>> the bridge using shared memory. The parent process can Get or Set
>> variables in the child process. Overall it's a very simple interface
>> that brings a powerful new dimension to IDL applications.
>
> Thanks for the info. Does Get/Set actually copy variables, or
> just provide references in shared memory (which of course would require
> IDL setting up the existing heap for sharing)? That would allow you to
> Get/Set self in an object, and have it operated on independently in both
> sides.
>
> JD
>

Subject: Re: 6.3 reactions?
Posted by codepod on Tue, 09 May 2006 17:30:49 GMT
View Forum Message <> Reply to Message

The following code gives an idea of how this can work. Note: This isn't
exact, but should outline how the IDL-IDL bridge is used.

Preconditions:
Assume you have a variable MyData that you want to use as input to a
time consuming routine called MyRoutine. If you ran this in an IDL
application, the application would block until processing is done.
However with the bridge you can have it process in the background. The
general structure of the code is the following:

```
  oBridge = obj_new("IDL_IDLBridge")    ; creates the child process

  ;; Put the data in the child process session. This creates
  ;; a variable called Data in the child process and copies the
  ;; value of MyData into it. This exchange uses shared memory
  ;; to transfer values, but the actual data memory is not shared
  ;; between each process. (2 copies exist).
  oBridge->SetVar, "Data", MyData

  ;; Lauch background processing
   oBridge->Execute, "result = MyRoutine( Data )", /nowait

   ;; This call returns immediately

   ;; At this point this work will execute in the background
   ;; process. At a later time you can check and see if the
   ;; task is completed
   iStatus = oBridge->Status()
   if(iStatus eq 2)then begin ;; execution is completed
       Result = oBridge->GetVar("Result") ;; get the result
   endif

   ;; When you are completed with the process, just destroy the object
   obj_destroy, oBridge
```

This simple example shows a "polling" method to check when the task is
completed. The object also has a callback methodology which operates
similar to an event callback. It will call a routine you specify when
an action happens (command done, error signaled).

Hopefully this helps.

-CP

Subject: Re: 6.3 reactions?
Posted by David Fanning on Tue, 09 May 2006 17:50:59 GMT
View Forum Message <> Reply to Message

codepod@gmail.com writes:

> The following code gives an idea of how this can work. Note: This isn't
> exact, but should outline how the IDL-IDL bridge is used.

Off-loading a compute intense process only really seems
to make sense if you can off-load it to a computer or
processor that is underutilized. Is there any way to

direct the bridge to run on a particular processor
(assuming a multi-processor system) or a particular
machine on the network?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: 6.3 reactions?
Posted by JD Smith on Tue, 09 May 2006 18:54:35 GMT
View Forum Message <> Reply to Message

On Tue, 09 May 2006 00:23:57 -0400, Richard G. French wrote:

> I hope that someone who really understands all of this IDL Bridge stuff
> and who doesn't use objects for every programming task will provide a
> simple description for civilians of just what this capability can do for
> the everyday user, and what it can't do. I'm having trouble figuring out
> whether this is a cool feature that I'd be crazy not to employ, an easy
> way to try to do parallel processing, or something that would require me
> to reprogram everything I do to obtain the leverage from multiple
> processors. So, all of you Ernest Hemingways out there, how about some
> simple, declarative sentences with an example or two of how this feature
> works and why it is a Good Thing. Many of us will be grateful!

Having not used it, as far as I can tell it's a convenient wrapper
around pre-existing functionality. You could already, with IDL
versions prior to 6.3, spawn another IDL process, setup a shared
memory channel of communication between them, copy variables back and
forth, and use an inter-process communication method, with special
communication code running on each side, to notify the parent process
when the child process is done with processing. Then you could read
back some results, and make use of them. It was just a big pain.

The new Bridge just makes that all easy for you... no need to setup
shared memory and/or some other access (like a semaphore) to
communicate back and forth, no need to poll to find out if a process
is done with some calculation. Just use Get/Set to send variables (by
copying) back and forth through shared memory, use callbacks to get
notified when processing is complete, etc. This would be useful if you
have a GUI front-end, which you'd like to keep responsive while
another IDL process chugs away on some big calculation. Basically, if

you like background processing of widget events, you'll love the
bridge, because you can have *anything* be background processed.

I don't know what the overhead for starting this sub-process and setting
up shared memory is, so it may only pay for really intensive calculations.
It would also be a potential speedup if you have a problem which
doesn't make use of large enough arrays (a few hundred thousand
elements) to benefit from IDL's native threading on multi-processor
machines. Again, the key factor determining how useful this will be
is the overhead involved in setting up the child process. Does it
really go through the entire rigmarole of contacting the license
server, expanding and searching the IDL_PATH, etc., establishing the
graphics devices, etc.? If so, it will be too heavy-weight for
speeding up smaller scale operations, but of course will be useful for
those 1 min or longer operations which otherwise would have blocked.

One other factor to consider: I presume that if you had a large pile
of data in process A, and you wanted to spawn sub-process B to work on
that data in the background, you'd end up with *two* copies of that
data in memory, one each in the space of processes A and B, unless you
specifically remove that data on one side, then the other (e.g. with
TEMPORARY). You thought preventing memory leaks was hard with one IDL
process...

JD

Subject: Re: 6.3 reactions?
Posted by JD Smith on Tue, 09 May 2006 18:59:55 GMT

View Forum Message <> Reply to Message

On Tue, 09 May 2006 11:50:59 -0600, David Fanning wrote:

> codepod@gmail.com writes:
>
>> The following code gives an idea of how this can work. Note: This isn't
>> exact, but should outline how the IDL-IDL bridge is used.
>
> Off-loading a compute intense process only really seems
> to make sense if you can off-load it to a computer or
> processor that is underutilized. Is there any way to
> direct the bridge to run on a particular processor
> (assuming a multi-processor system) or a particular
> machine on the network?

In terms of processor allocation: that's the operating system
scheduler's job, and it usually does a good one (depending on OS). I
think a good analogy for what you could do with this is iMovie,

Apple's easy video editing software. When you render an effect on a
clip, it gets done in the background, allowing you to continue to
re-organize and name clips, import more data, start other rendering
processes, etc. Even on a single processor chip, the usability is
greatly enhanced by not having to sit there twiddling your thumbs
for 45s while some complicated render completes. I think it needs an
easy hook for progress in Status(), not just "done" or "not done".

JD

Subject: Re: 6.3 reactions?
Posted by Mark Hadfield on Tue, 09 May 2006 23:21:36 GMT
View Forum Message <> Reply to Message

codepod@gmail.com wrote:
```
>     ;; Lauch background processing
>      oBridge->Execute, "result = MyRoutine( Data )", /nowait
```

That reminds me of a question that's been at the back of my mind. We
know the IDL VM does not support the EXECUTE function right? I don't
know if the reason for this limitation has ever been stated, but
presumably it's related to the fact that EXECUTE requires run-time
compilation, which the VM does not support.

So can I create an IDL_IDLbridge object and call its Execute method from
the VM? If so, surely I can use it to work around the EXECUTE
limitation? If not, why doesn't it say so in the documentation?

--
Mark Hadfield   "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: 6.3 reactions?
Posted by Craig Markwardt on Wed, 10 May 2006 15:23:50 GMT
View Forum Message <> Reply to Message

Mark Hadfield <m.hadfield@niwa.co.nz> writes:
```
>  codepod@gmail.com wrote:
>>     ;; Lauch background processing
>>      oBridge->Execute, "result = MyRoutine( Data )", /nowait
>
>  That reminds me of a question that's been at the back of my mind. We
>  know the IDL VM does not support the EXECUTE function right? I don't
```

> know if the reason for this limitation has ever been stated, but
> presumably it's related to the fact that EXECUTE requires run-time
> compilation, which the VM does not support.

It may be more of a marketing issue. When the RSI guys came to Goddard, Richard Cooke said something along the lines of, "if we put EXECUTE() into the IDL VM, we would be undercutting the main IDL product." I kind of see his point.

Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: 6.3 reactions?
Posted by Haje Korth on Wed, 10 May 2006 16:33:35 GMT
View Forum Message <> Reply to Message

Craig,
we had a similar discussion with RSI here at APL when the VM first came out. While I understand that you could theoretically have a free fully licensed IDL version, in pratice the person trying to rewrite the IDLDE as a widget in IDL must be a complete nut. And I still claim that, if one wanted to, and I most certainly do NOT encourage anyone to do so, there are far easier methods to get around licensing mechanisms. FLEXLM is useful for license management to ensure that an appropriate number of licenses is available in an organization, but its use as copy protection to prevent illegal use is in my opinion nothing but smoke and mirrors.

It is rather worth investigating how execute could be allowed in VM. Illegal use would be to read a text file line for line and execute each statement. But what if one would limit the number of execute statements per second or add a small delay after each execute. This would allow occasional use of execute, but using it for whole programs would make it painfully slow. Not sure whether this would be worth the effort, but there must be some way.

Haje

Subject: Re: 6.3 reactions?
Posted by Foldy Lajos on Wed, 10 May 2006 17:24:23 GMT

View Forum Message <> Reply to Message

Hi,

On Wed, 10 May 2006, Haje Korth wrote:

> Craig,
> we had a similar discussion with RSI here at APL when the VM first came
> out. While I understand that you could theoretically have a free fully
> licensed IDL version, in pratice the person trying to rewrite the IDLDE
> as a widget in IDL must be a complete nut.

you do not need a complete IDE, just a command line interface.

1. create runcmd.pro

```
pro runcmd
cmd=''
while 1 do begin
   read, cmd, prompt='IDL> '
   x=execute(cmd)
endwhile
end

runcmd
end
```

2. compile it and save, file='runcmd.sav', /routines

3. run idl -vm with runcmd.sav, and you would have a command prompt if
EXECUTE would be operational. (This would work on linux/unix. On windows,
you could create an editable text widget with EXECUTE in the event handling
routine.)

regards,
lajos

Subject: Re: 6.3 reactions?
Posted by Haje Korth on Thu, 11 May 2006 17:06:39 GMT
View Forum Message <> Reply to Message

Lajos,
yes, if you are happy with such scaled -down version then that will
work. I tend to used more of the functionality of IDLDE. BTW: Does this
mean you can buy an idl runtime version and turn it into a full
version? This is sort of the same problem, right?

Cheers,
Haje

Subject: Re: 6.3 reactions?
Posted by wem on Thu, 11 May 2006 17:26:50 GMT
View Forum Message <> Reply to Message

Haje Korth wrote:
> Lajos,
> yes, if you are happy with such scaled -down version then that will
> work. I tend to used more of the functionality of IDLDE. BTW: Does this
> mean you can buy an idl runtime version and turn it into a full
> version? This is sort of the same problem, right?
>
> Cheers,
> Haje
>

From my (short) experience with IDL and Callable IDL, I do think that
is possible.

Subject: Re: 6.3 reactions?
Posted by codepod on Thu, 11 May 2006 20:15:06 GMT
View Forum Message <> Reply to Message

Some key items we've found when using the bridge:
- Create several worker bridges at application startup and just use
them during the lifespan of your application. This minimizes any
bridge/process start up costs (most of which are path searching and a
quick license check).
- Minimize transfers between processes. The transfers are fast, but not
as fast as in process ops.
- Dispatch discrete tasks to the bridges for execution. Really, this
applies to any type of background/multi-thread type of processing work.

Concerning the VM posts in this thread:
- The execute method on the bridge is disabled in VM mode.
- The execute system routine was disabled to prevent the construction
of simple command line application creation using the VM (as the
example in this thread shows).

And on the question concerning using a runtime license to create a
command line, you could do it since execute is enabled, but code
compliation is disabled which is limiting.

-CP