
Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available
Posted by [JD Smith](#) on Thu, 13 Jul 2006 18:15:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Thu, 13 Jul 2006 09:24:56 -0700, bokubo wrote:

- > I am pleased to announce the release of IDL 6.3 for Mac OS X on Intel.
- > This new IDL release runs as a native application on all Mac Intel
- > supported machines and offers significant performance benefits. We have
- > seen a growing popularity of Mac OS X for scientific and analysis
- > applications, and this release represents our ongoing commitment to
- > this growing base of IDL users.

And....

it's *FAST*. Here's the breakdown for some portable OSX systems,
following along the tests at:

http://idl.tamu.edu/mac_bench.php

System	TT3 (AVG)	TT3 (GEOM)	JD_TEST
=====			
=====			
PB (G4 1.67GHz, 2GB, PPC IDL native)	3.20	.13	1.86
MBP(CoreDuo 2GHz, 1GB, PPC IDL via Rosetta)	3.45	.13	3.02
MBP(CoreDuo 2GHz, 1GB, i386 IDL native)	1.69	.06	0.32

All times in seconds.

PB == PowerBook

MBP == MacBook Pro (thanks to Jason Harris for a temporary loan)

TT3 == Time Test 3, run under IDL 6.3, demo mode.

AVG == average

GEOM == geometric mean

JD_TEST ==

```
IDL> a=randomu(sd,100L*!CPU.TPOOL_MIN_ELTS)
```

```
IDL> t=systime(1) & a=sqrt(a)/(a>0.5) & print,systime(1)-t
```

So, for things which are limited by a single processor (TIME_TEST3),
we're roughly ~2x faster than a G4 PB (which was a slow IDL system, to
be fair), with only 20% more clock speed.

But, the real fun comes when running big array manipulations, like
JD_TEST, where the dual Core Duo processors can flex their muscles.
Here the speedup is closer to 6x, which is almost too good to believe.
This will of course depend on which operations you use, but testing a
variety of arithmetic ones, I found speedups of anywhere from 2-7 on
arrays large enough to make multi-threading effective.

The MBP (a portable system) is now comparable in speed to a quad-processor G5, and (I suspect) similar dual-processor Linux/Windows desktops. Very respectable. Woeful OSX/IDL performance, R.I.P.

JD

Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available
Posted by [Foldy Lajos](#) on Thu, 13 Jul 2006 19:00:44 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

a little FL promotion :-)

On Thu, 13 Jul 2006, JD Smith wrote:

> it's *FAST*. Here's the breakdown for some portable OSX systems,
> following along the tests at:

>

> http://idl.tamu.edu/mac_bench.php

>

> System TT3 (AVG) TT3 (GEOM) JD_TEST

> =====
=====

> PB (G4 1.67GHz, 2GB, PPC IDL native) 3.20 .13 1.86

> MBP(CoreDuo 2GHz, 1GB, PPC IDL via Rosetta) 3.45 .13 3.02

> MBP(CoreDuo 2GHz, 1GB, i386 IDL native) 1.69 .06 0.32

FL (64 bit linux) on a dual 1.8GHz Opteron 0.57 .02 0.13

not bad :-)

for FL, TT3 includes assoc test (0.02 s), and running JD_TEST as

```
a=randomu(sd,100L*100000L)
```

```
t=systemtime(1) & a=sqrt(a)/(a>0.5) & print,systemtime(1)-t
```

(!CPU.TPOOL_MIN_ELTS is 0 by default in FL, because FL uses different limits for different operations (eg 100000 for addition, 25000 for sin))

and another good news: the new Intel Core 2 processors will have more SSE/SSE2 execution units, so FL will be even faster.

regards,
lajos

> All times in seconds.
>
> PB == PowerBook
> MBP == MacBook Pro (thanks to Jason Harris for a temporary loan)
> TT3 == Time Test 3, run under IDL 6.3, demo mode.
> AVG == average
> GEOM == geometric mean
> JD_TEST ==
> IDL> a=randomu(sd,100L*!CPU.TPOOL_MIN_ELTS)
> IDL> t=systime(1) & a=sqrt(a)/(a>0.5) & print,systime(1)-t
>
> So, for things which are limited by a single processor (TIME_TEST3),
> we're roughly ~2x faster than a G4 PB (which was a slow IDL system, to
> be fair), with only 20% more clock speed.
>
> But, the real fun comes when running big array manipulations, like
> JD_TEST, where the dual Core Duo processors can flex their muscles.
> Here the speedup is closer to 6x, which is almost too good to believe.
> This will of course depend on which operations you use, but testing a
> variety of arithmetic ones, I found speedups of anywhere from 2-7 on
> arrays large enough to make multi-threading effective.
>
> The MBP (a portable system) is now comparable in speed to a quad-processor
> G5, and (I suspect) similar dual-processor Linux/Windows desktops. Very
> respectable. Woeful OSX/IDL performance, R.I.P.
>
> JD
>
>

Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available

Posted by [JD Smith](#) on Thu, 13 Jul 2006 19:09:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Thu, 13 Jul 2006 21:00:44 +0200, Ficz1/2LDY Lajos wrote:

> Hi,
>
> a little FL promotion :-)
>
>
> On Thu, 13 Jul 2006, JD Smith wrote:
>
>> it's *FAST*. Here's the breakdown for some portable OSX systems,
>> following along the tests at:

```

>>
>> http://idl.tamu.edu/mac_bench.php
>>
>> System                TT3 (AVG)  TT3 (GEOM)  JD_TEST
>> =====
>> =====
>> PB (G4 1.67GHz,  2GB, PPC IDL native)    3.20    .13    1.86
>> MBP(CoreDuo 2GHz, 1GB, PPC IDL via Rosetta) 3.45    .13    3.02
>> MBP(CoreDuo 2GHz, 1GB, i386 IDL native)  1.69    .06    0.32
>
> FL (64 bit linux) on a dual 1.8GHz Opteron 0.57    .02    0.13
>
> not bad :-)
>
> for FL, TT3 includes assoc test (0.02 s), and running JD_TEST as
>
> a=randomu(sd,100L*100000I)
> t=systime(1) & a=sqrt(a)/(a>0.5) & print,systime(1)-t
>
> (!CPU.TPOOL_MIN_ELTS is 0 by default in FL, because FL uses different
> limits for different operations (eg 100000 for addition, 25000 for sin))
>
> and another good news: the new Intel Core 2 processors will have more
> SSE/SSE2 execution units, so FL will be even faster.

```

Impressive. How much of FL's performance gain comes from compiler optimizations? I am suspect of speedups in operations limited by looping, since FL and GDL don't yet implement widgets or other even processing which slows IDL's loops down. However, just taking the square root of a bunch of numbers.... that's a different question.

What are the plans for FL? Any hope of combining the two efforts (FL/GDL) into one open source project?

JD

Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available
 Posted by [Karl Schultz](#) on Thu, 13 Jul 2006 20:03:33 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Thu, 13 Jul 2006 11:15:15 -0700, JD Smith wrote:

```

> On Thu, 13 Jul 2006 09:24:56 -0700, bokubo wrote:
>
>> [quoted text muted]
>
> And....

```

```

>
> it's *FAST*. Here's the breakdown for some portable OSX systems,
> following along the tests at:
>
> http://idl.tamu.edu/mac_bench.php
>
> System                      TT3 (AVG)  TT3 (GEOM)  JD_TEST
> =====
> =====
> PB (G4 1.67GHz, 2GB, PPC IDL native)    3.20    .13    1.86
> MBP(CoreDuo 2GHz, 1GB, PPC IDL via Rosetta) 3.45    .13    3.02
> MBP(CoreDuo 2GHz, 1GB, i386 IDL native)  1.69    .06    0.32
>
> All times in seconds.
>
> PB == PowerBook
> MBP == MacBook Pro (thanks to Jason Harris for a temporary loan)
> TT3 == Time Test 3, run under IDL 6.3, demo mode.
> AVG == average
> GEOM == geometric mean
> JD_TEST ==
> IDL> a=randomu(sd,100L*!CPU.TPOOL_MIN_ELTS)
> IDL> t=systemtime(1) & a=sqrt(a)/(a>0.5) & print,systemtime(1)-t
>

```

A few more datapoints (all for TT3 (AVG)):

```

iMac - i386 CoreDuo 2.0 GHz : 1.2 (ppc binaries with Rosetta: 2.3)
Pentium 4 (Windows) 2.8 GHz : 1.5
G4 ppc 0.7 GHz :          5.6
G5 dual ppc 2.0 GHz :          1.9
Xeon dual 3.4 GHz Linux :    1.1

```

Karl

Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available
 Posted by [FL](#) on Thu, 13 Jul 2006 20:10:28 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

FL can be started as 'fl --gui', which will present a very primitive interface, one window for standard input, and another for standard output (this is the default on Windows). However, the event loop is running in this case, and the results for TT3 and JD_TEST are the same.

I use gcc 3.4.x and gcc 4.1.x for developing FL, I think ITTVIS/RSI uses

the same compiler (on linux, Intel compiler on Windows). I have written MMX/SSE/SSE2 array arithmetics routines in inline assembly, this can be accounted for some speedup. I have written my own memory allocators (more than one, different allocators for different usage patterns.) But the real difference comes from overall design, from the very first day performance was my main concern. Implementation is very important. Take eg. matrix multiplication in IDL and Matlab: for 1000x1000 matrices, Matlab is 6-7 times faster, although both are using the same BLAS API. The difference is in the implementation: IDL uses a simple C BLAS, while Matlab uses ATLAS (<http://math-atlas.sf.net>).

My plans:

- very short term: some more vacation :-)
- short term: widgets have absolute priority. Some basic functionality already work, usable widgets will be available, let's say, by end of September.
- long term (1-2 years):

DFL (Distributed FL): similar to the IDL-IDL bridge, but slave processes can be invoked on different machines, communicating through TCP/IP (FL's of the world, unite! :-)

Matlab front-end: I guess 90% of the present code can be reused in a Matlab-like interpreter. So, why not create a 2-in-1 monster?

The bad news is that FL will remain a one man show. I used to work in small groups earlier, but I have very bad memories. Countless hours were spent for meetings and communication, instead of productive work.

regards,
lajos

On Thu, 13 Jul 2006, JD Smith wrote:

- > Impressive. How much of FL's performance gain comes from compiler optimizations? I am suspect of speedups in operations limited by
- > looping, since FL and GDL don't yet implement widgets or other even
- > processing which slows IDL's loops down. However, just taking the
- > square root of a bunch of numbers.... that's a different question.
- >
- > What are the plans for FL? Any hope of combining the two efforts
- > (FL/GDL) into one open source project?
- >

> JD
>

Subject: Re: IDL 6.3 for Mac OS X on Intel Now Available
Posted by [Paolo Grigis](#) on Fri, 14 Jul 2006 08:00:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

I think it would be very interesting to compare performance with a macbook with the same specification running windows via bootcamp... will this give a further increase in performance due to windows optimizations? Or at least somebody with a native windows core duo laptop could report.

Otherwise I cannot understand why a windows laptop with a pentium M clocked at 1.6 GHz performs 1.27 in TT3... or are core duo processors *expected* to be less efficient than similarly clocked pentium Ms for single processor task?

(For comparison, on that system JD_TEST returns 0.73, a bit more than a factor 2 worse than dual core, as expected).

Ciao,
Paolo

JD Smith wrote:

> On Thu, 13 Jul 2006 09:24:56 -0700, bokubo wrote:

>
>

>> I am pleased to announce the release of IDL 6.3 for Mac OS X on Intel.
>> This new IDL release runs as a native application on all Mac Intel supported machines and offers significant performance benefits. We have
>> seen a growing popularity of Mac OS X for scientific and analysis applications, and this release represents our ongoing commitment to
>> this growing base of IDL users.

>
>

> And....

>

> it's *FAST*. Here's the breakdown for some portable OSX systems,
> following along the tests at:

>

> http://idl.tamu.edu/mac_bench.php

>

> System TT3 (AVG) TT3 (GEOM) JD_TEST

```

> =====
> =====
> PB (G4 1.67GHz, 2GB, PPC IDL native)    3.20    .13    1.86
> MBP(CoreDuo 2GHz, 1GB, PPC IDL via Rosetta) 3.45    .13    3.02
> MBP(CoreDuo 2GHz, 1GB, i386 IDL native)    1.69    .06    0.32
>
> All times in seconds.
>
> PB == PowerBook
> MBP == MacBook Pro (thanks to Jason Harris for a temporary loan)
> TT3 == Time Test 3, run under IDL 6.3, demo mode.
> AVG == average
> GEOM == geometric mean
> JD_TEST ==
> IDL> a=randomu(sd,100L*!CPU.TPOOL_MIN_ELTS)
> IDL> t=systime(1) & a=sqrt(a)/(a>0.5) & print,systime(1)-t
>
> So, for things which are limited by a single processor (TIME_TEST3),
> we're roughly ~2x faster than a G4 PB (which was a slow IDL system, to
> be fair), with only 20% more clock speed.
>
> But, the real fun comes when running big array manipulations, like
> JD_TEST, where the dual Core Duo processors can flex their muscles.
> Here the speedup is closer to 6x, which is almost too good to believe.
> This will of course depend on which operations you use, but testing a
> variety of arithmetic ones, I found speedups of anywhere from 2-7 on
> arrays large enough to make multi-threading effective.
>
> The MBP (a portable system) is now comparable in speed to a quad-processor
> G5, and (I suspect) similar dual-processor Linux/Windows desktops. Very
> respectable. Woeful OSX/IDL performance, R.I.P.
>
> JD
>

```
