

64-Bit CLR

64-bit Questions

The world is moving to 64-bit – should I?

64-bit is the future, but is it the present?

As a developer, what steps should I take now to prepare for the 64-bit move?

What hardware should I get?

How can I write .NET 4 code to work well on 64-bit?

Are there any issues with Visual Studio 2010 and the 64-bit CLR?

- These are the questions we need to consider as we move to 64-bit

64-bit (x64) is becoming popular

32-bit (x86) on the server is going away

- There is no 32-bit version of Windows Server 2008 R2

For Windows 7 deployments, 64-bit is becoming more popular than 32-bit

However ...

- 32-bit on the desktop and especially mobile is still popular
- Don't underestimate the sheer quantity of deployed devices/apps that are 32-bit only
- Some devices may never need a 64-bit OS (well, not for a VERY long time)
Smartphones, netbooks, iPad-like devices, embedded devices
(Anything with less than 4GB RAM)

Windows And Memory

32-bit Windows limited to less than 4GB (2^{32})

- Some of the addressing range use for device mapping, so a machine with 4GB RAM only makes slightly over 3GB available to the OS and applications

Limit for full versions of 64-Bit Windows 7

- Ultimate/Enterprise/Professional : 192GB
- Home Premium : 16GB

Limit for Windows Server 2008 R2

- Datacenter/Enterprise : 2 TB
- Standard/Web : 32 GB
- Foundation : 8 GB

What Is Holding Us Back?

Why don't we all just move to 64-bit today?

- Inertia of developers, IT departments and end-users
- It involves change, and people hate change
- Cost
- Plan to do it in future
- Need to recompile 32-bit code
(A 64-bit EXE can only load 64-bit DLLs, and a 32-bit EXE can only load 32-bit DLLs)
- May also need to continue supporting 32-bit, and having the one build may be easier
- Our app may use a third-party or system DLL that is only available as 32-bit

New PC For A 64-bit Developer

One or more Quad-core processors

- All recent processors support x64
- For Hyper-V virtualization, ensure processor support (AMD-V or Intel VT)
- If encrypting disk with BitLocker (recommended), should get Trusted Platform Module (TPM) chip

Running Windows 2008 R2 SP1 is recommended

8 GB or more of RAM

- Get as much RAM as you can afford, and ensure slots are left free for additions in future, without having to throw away existing DIMMs

At least a large 24" monitor (Full HD, 1920x1080)

- Features: TN vs. IPS (better), ability to rotate, 120Hz for 3D output, multi-touch
- Some very nice (but expensive) 27" monitors now available
- Many developers would prefer greater vertical pixel count (e.g. 1920x1200) but there can be a huge cost difference

Visual Studio 2010 and 64-bit

The Visual Studio 2010 product is compiled in 32-bit only

- There is no 64-bit Visual Studio 2010 product

The IDE can be used to create 32-bit and 64-bit applications

- The IDE runs in a separate process space to the application being developed, so they need not have the same bit-ness
- 64-bit Windows includes WOW64, a 32-bit x86 emulator, than runs 32-bit apps on x64

CLR comes in 32-bit and 64-bit versions

- Almost same functionality, but ...
- Edit-and-Continue (EnC) only supported in 32-bit - argh!!!
- Intellitrace (a VS2010 Ultimate feature) is not supported
- VS2010 ASP.NET Development Server (Cassini) is 32-bit only (use IIS7.5 instead)

AnyCPU, x86 or x64

.NET assemblies can be built as:

- x86
(Will run natively on x86 OS and run in WOW64 on x64 OS)
- x64
(won't run on x86, will run natively on x64 OS)
- or AnyCPU
(EXEs run natively - e.g. runs as x86 app on x86 OS, and x64 app on x64 OS)
(DLLs take the bitness from the EXE of the process in which then run)

When creating new projects, VS2010 by default sets:

- DLLs to be AnyCPU, and
- EXEs to be x86 (even on 64-bit Windows)

CORFLAGS

How do we determine bitness of EXE or DLL?

Windows Explorer is of no help

Instead, use corflags.exe

- A .NET SDK tool
- Call from command line, with name of EXE or DLL
- With no additional parameters, it dumps status

Interesting Fields

- Version - CLR version
- PE - PE32 means exe can run in 32 bits, PE32+ means it can't (PE32 for x86 and anyCPU, PE32+ for x64)
- 32BIT - 1 means mandatory to run as 32 bit, 0 means not mandatory (1 for x86, 0 for anyCPU and x64)

x64 And Task Manager

How do we determine bitness of running process?

On a 32-bit OS, all processes are 32-bit

On a 64-bit OS, processes could be 32-bit or 64-bit

Could use CORFLAGS to determine compilation

- but what about AnyCPU EXEs?

Solution - Use Task Manager

- If process name is succeeded by *32, it is a 32-bit process
- Otherwise, it is a 64-bit process

From code, can determine this simply by checking size of IntPtr (see later)

64-Bit And IIS 7.5

When you create an ASP.NET MVC app, you are creating a DLL
VS2010 supports three types of web servers

- Visual Studio Development Server (also called “CASSINI”)
- Local IIS Web Server
- Custom Web Server

These provide the worker process which loads that DLL

- Cassini only supplies a 32-bit worker process
- IIS7.5 supplies both 32-bit and 64-bit worker processes

By default, your DLL is marked as Any CPU

- If you set it to x64, it won't run on CASSINI, but will on IIS 7.5

short/Int16, int/Int32, long/Int64

C# language primitives = short, int, long

.NET Framework System Types = System.Int16, System.Int32, System.Int64

- Imagine a software architect with a team of 20 developers, of varying experience levels, and she asks the question:
- “For 32-bit and 64-bit OSes, how many bits are in short, int, long, Int16, Int32 & Int64?”
- Will get a range of answers - right one (for .NET) is same on both OSes
- On other platforms/OSes/languages, size of ints do change with bit-ness of OS

Aside: could also ask the question - is it signed or unsigned?

- Most developers will remember UInt64, and so (correctly) assume Int64 is signed

Coding Guidelines

Due to confusion over size of integer values ...

We recommend using .NET Framework types exclusively

- Ignore language primitives
- 100% of developers know Int32 has 32-bits, certainly not true for int

Developers need to know .NET Framework types, as they will be seen in Framework APIs (System.Convert.ToInt32)

- There are multiple .NET languages, and public custom API names should include framework types, not language-specific primitives

Approach recommended by Jeffrey Richter in excellent “CLR via C#”

IntPtr

C# can work with memory pointers and OS handles

- Unmanaged memory
- IMPORTANT: apps should not be using Int32 to store a memory pointer

Two structs - `System.IntPtr` and `UIntPtr` - change with bitness

- One of the very few places in .NET that does

`IntPtr.Size` property

- 4 on 32-bit platforms, 8 on 64-bit platforms

`IntPtr.ToPointer`

- Converts to a void*

Unsafe Code And Pointer Arithmetic

- When porting from 32-bit to 64bit, need to carefully verify

Binary Serialization

So, I'm a 64-bit .NET Developer!

The best way to become a 64-bit .NET Developer is ...

- Simply to write code

To-do list

- Get a 64-bit OS
- Use VS2010 to create EXEs and DLLs
- Mark them as anyCPU or x64
- Be aware of integer sizing not changing
- Be aware of IntPtr changing
- Be aware of small number of missing VS2010 features (Edit and Continue)

There is very little difference between 32-bit CLR and 64-bit CLR

Recommendations

If developing new server applications

- Strongly recommend you base it on 64-bit

For new and existing desktop applications

- Really need to produce a 64-bit build, but for many apps will also need to produce a 32-bit build
- For enterprise desktop apps, where you know installed base is 64-bit, then build 64-bit app
- For mobile apps, 32-bit is for now, but be aware of ultimately having to port to 64-bit

So, should I wait for a 128-bit OS?

NO! How long do you wish to wait for?

- The principle reason for moving to 64-bit programming is to support additional RAM
- We won't see a general purpose OS with 128-bit RAM addresses for a long time

$2^{64} = 1,844,674,407,370,955$, or nearly 1,700,000 GB

- When will that not be enough RAM? (some day, yes!)

We will see 128-bit usage for:

- Multimedia operations
- SIMD operations (Single instruction, multiple data vector programming)
- Filesystem addressing (e.g. Oracle/Sun ZFS is an interesting 128-bit filesystem)

Needed for:

- Oil&gas, particle physics, astronomy, media, defense all involve huge data volumes