

Instrumented Code is Better Code

Mark Williams

www.cheshamdb.com

mawilliams@cheshamdb.com

oradim.blogspot.com

It's more of a hope than a promise

- I will do my best to not simply stand here and read each slide word for word with all the excitement of a Steven Wright¹ comedy presentation. I have been on the receiving end of that before and I realize it isn't all that exciting. OK, so maybe for this slide I am breaking my own promise, but, hey it's more of a hope than a promise and that has consequences...

¹ <http://humor.mcf.com/misc/stevenwright.html>

Agenda

- The customary README.TXT information
- What I like to call themes
- What is better code?
- What is code instrumentation?
- Instrumentation implementation
- Timer frequency, resolution, and accounting errors
- The cost of self-measurement
- CPU consumption and division of labor
- Why did I use punctuation only for questions?

README.TXT - 1

- The three W's: Who, What, Why
- Who?
- Mark Williams (in case you missed the first slide or got into the wrong room)
 - Pro .NET Oracle Programming (Apress, 2004)
 - ODP.NET Column in Oracle Magazine
 - Oracle® Database Express Edition 2 Day Plus .NET Developer Guide 10g Release 2 (10.2) – contributor
 - Various OTN articles
 - Oracle ACE Director (sort of)
 - Becoming an anachronism?

README.TXT – 1.1

- Interests (lots really, but here's a short list):
 - Private pilot (not currently active though)
 - Geometry/Trigonometry
 - Languages (C/C++/C#/Assembly) – plan on Perl... and dare I say Java?
 - Languages (French/Welsh)
 - British history (historical novels – Nigel Tranter)
 - MotoGP (Rossi/Yamaha)
 - Barclay's English Premier League (Liverpool)
 - Learning piano

README.TXT - 2

- What? (As in what do I do? Not as in "What is the average airspeed velocity of an unladen swallow?")
- Even though I work for Oracle, this is not an Oracle presentation – say what?
- Formerly on the Database Internals Team
- Let's call this middle bit a sabbatical, shall we?
- Now on Platforms BDE (Non-Windows)

README.TXT - 3

- Why?
- It's a topic in which I believe
- Hopefully the rest of the presentation will fill out more details
- There is some "Gee whiz" information but not much internals
- In a shocking turn of events, I don't know everything
- That is, $\sum \text{your experience} > \sum \text{my experience}$

README.TXT - 4

- My development machine:
- Intel Core 2 Duo processor E6750 2.66Ghz
- 8 GB RAM (PC2-6400)
- Seagate Barracuda ST31000340AS (1 TB SATA)
- Currently Microsoft Vista x64 (Should I go Linux? Swallow the Mac pill? O/S 2 Warp?)
- Visual Studio 2008 with VC++ feature pack and SP1 (demos built as "Any CPU")
- CPU-Z used... I'm not a hardware person!

Fun with Themes

- These are some common themes for just about any presentation I do...
- So, you may have seen them before
- Mileage may vary from presentation to presentation

Theme #1

- Performance does not **have** to be advanced, dark, hard, mysterious, undocumented, or _underscored.

Theme #2

- Silver bullets usually are not and should not be needed.
- For example, incorrect setting of `db_file_multiblock_read_count`
- Subsequently sorted out...
- Shows up on a forum as "I set it to X and my system performance increased 500%!!!"
- And before you know it...
- Which is why I am not a Silver Bullet fan or **generally** a "Best Practices" fan

Theme #3

- Things you do outside of the database can have an impact on things inside the database.

Theme #4

- System performance can be like an orchestra - All sections and pieces need to work in concert (pun intended?) with one another for the piece as a whole to work.

Theme #5

- Begin at the beginning. Correct code should not need workarounds at the database or after deployment.
- Though I am a realist here and sometimes you get that box of chocolates...

Theme #6

- My naïve definition of performance:
The overall time taken to perform a task
As measured at or by the client/user

Theme #7

- I'm pretty fond of quotes – such as:
- "If you can not measure it, you can not improve it.", Lord Kelvin (1824-1907)
- "X-rays will prove to be a hoax.", Lord Kelvin
- "Don't confuse activity with accomplishment.", Zig Ziglar
- "Oracle system performance and code instrumentation? That's hot!", Anonymous
- Google to find citations...

Assumption

begin

```
dbms_presenter.set_assumptions (  
    primary_attendee => 'DBA',  
    secondary_attendee => 'Developer',  
    dev_language => 'C#',-- curly braces and ;  
    app_type => null  
);
```

end;

/

Some Time Definitions

- "I know you're in there, you're just out of sight... Time passages...", Al Stewart
- Centisecond: $1/100$ 'th of a second (0.01)
- Millisecond: $1/1,000$ 'th of a second (0.001)
- Microsecond: $1/1,000,000$ of a second (0.000001)
- Nanosecond: $1/1,000,000,000$ of a second (0.000000001)
- Friday at 4:30 PM: Time of notice of new X?

Instrumented Code is Better Code

- Didn't things used to be simpler (but not necessarily better)?
- If things are more complex, wouldn't something that helps narrow down issues be welcomed and used?
- Yet, code instrumentation in a corporate application is rarely seen
- Why is that?

Instrumented Code is Better Code

- What is better code?
- Hint: Subjective and Objective
- You get decide
- But then you need to measure against your decision(s)

Which Code is Better?

Behind Door #1

```
using System;

namespace HelloWorld {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("Hello, world.");
        }
    }
}
```

Behind Door #2

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, world.");
        }
    }
}
```

Which Code is Better?

- OK, obviously code needs to compile and execute (correctly)...
- For many environments whitespace (and the kind of – space vs. tab) is not important, but it can be... i.e. "make"
- For many people better code is code that executes correctly, uses less resources, and is quicker. 4 out of 5 managers said this is true.
- Not many year-end bonuses are given out for blazing fast code that is not correct.

What is Code Instrumentation?

- It sounds complicated and involved – is it? (Maybe just because it is new)
- Does it require advanced techniques?
- Here's my definition:

Code instrumentation is the **strategic** placement of host-language constructs (function or procedure calls) such that an **application may emit performance information at run-time.**

What is Code Instrumentation?

- Isn't that just logging or debug code?
- Why is code instrumentation needed?
- It's laughably simple...
- Because now you can tell how long something takes!

What is Code Instrumentation?

Have you ever seen a lawyer/legal show? If so, you've likely at one point or another heard:

"It goes to intent, your honor!"

Survey Says...

Q: Is poor performance always the fault of the database?

A: **"It [performance] is many times an application issue and when the application is spread over 14 tiers of complexity, tracking down the bottleneck is grievously hard. If you just whip together an application and throw it out there without any thought to monitoring it over time, be prepared to have poor performance and no clue as to why or where."**

- Instrumentation (<http://tkyte.blogspot.com/2005/06/instrumentation.html>)

Ever Seen Code Like This?

```
using System;
```

```
namespace HelloWorld {  
    class Program {  
        static void Main(string[] args) {  
            Console.WriteLine("Hello, world - ");
```

```
#if DEBUG
```

```
    Console.WriteLine("I'm a debug build!");
```

```
#else
```

```
    Console.WriteLine("I'm a release build!");
```

```
#endif
```

```
    }
```

```
}
```

```
}
```

Ever had a Performance Ticket with Oracle?

- Did they:
- Fly a specialist debug team out to your site
- Replace the Oracle binaries with debug builds
- Step through the code using a debugger
- Get the needed information
- Replace the debug builds with release builds
- Thank you and say, "Have a good day!"
- I suspect it was a bit different from this

Survey Says Part II...

"Also, make this instrumentation part of the production code, don't leave it out! Why? Because, funny thing about production - you are not allowed to drop in "debug" code at the drop of a hat, but you are allowed to update a row in a configuration table, or in a configuration file! **Your trace code, like Oracle's **should always be there**, just waiting to be enabled."**

- Instrumentation (<http://tkyte.blogspot.com/2005/06/instrumentation.html>)

(Very) Quick Review of 10046

2

- Several ways to enable:
- alter session, spfile, dbms_system
- Multiple levels (dictate what you get):
- Level 1 = Standard SQL_TRACE (i.e. SQL)
- Level 4 = Level 1 + binds
- Level 8 = Level 1 + waits (what are these?)
- Level 12 = Level 4 + Level 8

(Very) Quick Review of 10046

PARSE #5:c=15600,e=100024,p=4,cr=146,cu=0,mis=1,r=0,dep=0,og=1,
plh=3447538987,tim=592842132530

EXEC #5:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3447538987,
tim=592842132530

WAIT #5: nam='SQL*Net message to client' ela= 3 driver id=1111838976 #bytes=1
p3=0 obj#=-1 tim=592842132884

WAIT #5: nam='db file sequential read' ela= 9756 file#=5 block#=83 blocks=1
obj#=70296 tim=592842142861

WAIT #5: nam='db file scattered read' ela= 849 file#=5 block#=84 blocks=5
obj#=70296 tim=592842143803

FETCH #5:c=0,e=0,p=6,cr=7,cu=0,mis=0,r=1,dep=0,og=1,plh=3447538987,
tim=592842132530

(Very) Quick Review of 10046

- MetaLink Note:601528.1 (Times Reported For Waits in Trace File Are Too High)
 - Windows: boot.ini "/usepmtimer"
 - Linux: clock=pmtmr
 - Sun Solaris 10 3/05 apply the 1/06 (Update 1) Patch (or later)
- Bug 7522002 (Timing flaw in 10046 trace data)

Something a little bit extra

- Tanel Poder's blog has a recent and nice discussion of Oracle diagnostic events
- <http://blog.tanelpoder.com/2009/03/03/the-full-power-of-oracles-diagnostic-events-part-1-syntax-for-ksd-debug-event-handling/>

Instrumentation Implementation

- Pseudo-code of basic instrumentation:

```
t0 = get_current_timing_value();
```

```
perform_action();
```

```
t1 = get_current_timing_value();
```

```
duration = t1 - t0;
```

Instrumentation Implementation

- Warning! Remember my definition said "strategic"?

```
int i = 0;
```

```
t0 = get_current_timing_value();
```

```
i++; // no way I would instrument this! Over-instrumentation warning!
```

```
t1 = get_current_timing_value();
```

```
duration = t1 - t0;
```

Instrumentation Implementation

- Using DateTime (in .NET land)

```
DateTime timeStart;
```

```
DateTime timeEnd;
```

```
double duration;
```

```
timeStart = DateTime.Now;
```

```
perform_action();
```

```
timeEnd = DateTime.Now;
```

```
duration = timeEnd.Subtract(timeStart).TotalSeconds;
```

Instrumentation Implementation

- Under the covers `DateTime.Now` is a call to `UtcNow.ToLocalTime` (property.method)
- Which in turn ends up calling the `GetSystemTimeAsFileTime` Win API function
- The end result is the current date and time for the computer with the time zone taken into account.
- `gettimeofday` if on *nix system (man)
- `gethrtime` & `gethrvtime` may be used (timing)

Instrumentation Implementation

- Using Stopwatch (again in .NET land)

long duration;

```
System.Diagnostics.Stopwatch sw = new System.Diagnostics.Stopwatch();
```

```
sw.Start();
```

```
perform_action();
```

```
sw.Stop();
```

```
duration = sw.ElapsedMilliseconds;
```

Instrumentation Implementation

- `Stopwatch.Start` calls the `GetTimestamp` method which can return either the result of a call to the `QueryPerformanceCounter` Win API function (if high-resolution) or the number of system ticks equivalent to the current `DateTime` object (if not high-resolution) – more on ticks later...
- `Stopwatch.Stop` does the same thing but it also increments the "elapsed" property. (`Timespan`)
- `Stopwatch.ElapsedMilliseconds` returns the total measured milliseconds.

Instrumentation Implementation

- If you examine properties for the Stopwatch class and the DateTime/TimeSpan structures you'll see that they expose values in terms of milliseconds...
- Pop Quiz: Incidentally what's the main difference between a class and a structure in .NET?
- In .NET classes are reference types (managed heap, GC) whereas structures are value types (stack).

Timer Frequency, Resolution, and Accounting Errors

- As shown, when software measures its own performance it:
- Obtains a discrete timing value
- Performs an action (or actions)
- Obtains another discrete timing value
- Calculates the difference
- Are there any problems with this?

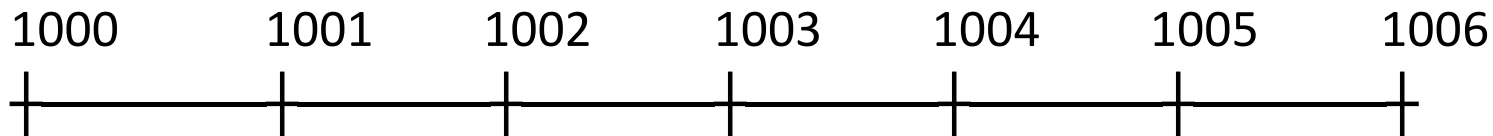
Timer Frequency, Resolution, and Accounting Errors

- What if an alarm was set for 06:00 AM and the clock read as follows:

05:59 AM

Timer Frequency, Resolution, and Accounting Errors

- Tick, tick, tick, tick (like a fixed metronome!)
- The disco flashback
- Frequency and resolution



XX

The Beat Goes On

- Time keeps on slipping, slipping, slipping into the future (Steve Miller Band, Fly Like an Eagle)
- Run Queue Run (Starring Tom Hanks as the operating system scheduler)
- Things can happen to your process/thread whilst timing is in progress

Gathering Timer Information

3

- Timer (Stopwatch) is likely based on a high-resolution timer. The static "IsHighResolution" field will report this. (I don't have a system where this is "false".)
- Another helpful static field is "Frequency". This is the number of ticks/second.
- We can calculate accuracy (resolution) in terms of nanoseconds by dividing 1,000,000,000 by frequency.

Gathering Timer Information

- `if (Stopwatch.IsHighResolution) {`
 `...`
 `}`
- `Int64 frequency = Stopwatch.Frequency;`
- `Int64 nanosecPerTick = (1000000000) /`
 `frequency;`
- Timer frequency in ticks per second = 14318180
 (on my dev system)
- Timer is accurate within 69 nanoseconds (dev)
- That is, a pulse/tick every 69 nanoseconds

Gathering Timer Information

- Earlier I mentioned that Stopwatch and TimeSpan expose properties in terms of milliseconds...
- But, at least on my dev system, there is a higher resolution available
- Stopwatch.ElapsedTicks property exposes number of ticks (elapsed naturally) which can be used with resolution....
- $\text{ElapsedTicks} * 69 == \text{Elapsed nanoseconds}$

Gathering Timer Information

- You should reset timer if using elapsed property between retrieval of information (elapsed just keeps growing and growing and growing – i.e. start/stop/elapsed/start/stop...)
- The Stopwatch ticks are not the same as DateTime/TimeSpan ticks... but a tick is a tick
- DateTime/TimeSpan ticks are 100 nanosecond intervals (since 01/01/0001) and, as we now know, Stopwatch ticks may be a different interval (ticks are ticks, interval is different)

The Cost of Self-Measurement

- The Lilly Endowment Fund – what does this have to do with instrumentation?
- The word "endowment" can have special meaning
- There's this thing called "Endowment Effect"
- http://en.wikipedia.org/wiki/Endowment_effect
- Perceived loss of something currently owned...
- Maybe the grass is not always greener?

The Cost of Self-Measurement

- This is one of the most often used excuses!

Chen Shapira: I've worked for an instrumentation vendor, and I also talked to many of them as part of my production DBA role. Whenever an instrumentation vendor talks to a prospective customer, **the first question is always: "What is the overhead?"**. Not "How it can help me?", "How much I can expect to improve my performance?" or "Is it easy to use?".

<http://prodlife.wordpress.com/2009/02/04/psychology-of-instrumentation/>

The Cost of Self-Measurement

- Here's a HotSos'ism (maybe just a Cary'ism)...
- Q: What's the fastest way to do something?
- A: Just **don't** do it (the Nike antithesis?)
- Umm, so, what's the slowest way to do something?
- I don't know, maybe do it a bunch of times...

The Cost of Self-Measurement

- Here's a proposed method of calculating this cost:
 - Create a stopwatch (sw1)
 - Create a stopwatch (sw2)
 - Start sw1
 - Loop 1,000,000 times and start/stop sw2
 - Stop sw1
 - Get elapsed milliseconds and divide by 1,000,000

The Cost of Self-Measurement

- This test results in an average time on my main development system of a whopping 0.00124 ms ($\approx 1 \mu\text{s}$)
- That doesn't seem like a long time to me
- Of course this is just (an approximation of?) the time for the Stopwatch code path... you may do other things besides simply capture the time information (like, say, write data to a file or something crazy like that).

Hey! Another Quote!

- By judiciously instrumenting your applications you have the ability to find performance issues at run-time. That's important! A best practice?

"I'm suspicious about most so-called best practices... There is one "best practice," though, that I believe in deeply: "You should always measure your application's performance and target your optimization efforts at places where your code will benefit from it most."

For Developers: Making Friends with the Oracle Database (http://method-r.com/downloads/cat_view/38-papers-and-articles)

CPU Consumption and Division of Labor

- It's been said that it's a binary world (or it isn't...) There's 10_b kinds of people...
- So, your application is either consuming CPU or it isn't.
- If it is consuming CPU that CPU can be in either:
 - Kernel mode
 - User mode

CPU Consumption and Division of Labor

- The Process class allows us to determine how much CPU time has been used.
- There are three properties of interest for this purpose:
- TotalProcessorTime: The total processor time used by this process.
- UserProcessorTime: The total processor time in user mode.
- PrivilegedProcessorTime: The total processor time in kernel mode.
- But that is for a process... aggregated/v\$view...
- The ProcessThread class can be used to get this information at the thread level.

CPU Consumption and Division of Labor

- The Process class (eventually) calls the GetProcessTimes Win API function.
- The ProcessThread class (eventually) calls the GetThreadTimes Win API function.
- These functions take pointers to FILETIME structures which are populated with kernel and user values.
- getrusage if on *nix system (man)

CPU Consumption and Division of Labor

- CPUTime Demo (mostly contrived)
- Perform operations using kernel & user CPU
- Create a file, write to it, and delete it – kernel
- Compute a square of a value - user
- Display results
- Total Processor Time: 216 ms
- Privileged Processor Time: 76 ms
- User Processor Time: 140 ms

There's really a single reason...

- If you are not convinced that instrumented code is better code, there's really a single best reason why this is so:

"Why guess when you can know?"

(Cary Millsap, For Developers: Making Friends with the Oracle Database
(http://method-r.com/downloads/cat_view/38-papers-and-articles) for one
example)

Closing Recommendations

- Do it! (In all layers)
- Modularize your implementation
- Create files (not on-screen)
- Develop in-house if you want - always do what is best in your environment. (NIH concerns)
- Run-time enable, not compile-time
- Be aware of any security concerns – especially if you mix instrumentation with debug/trace data

Thanks!