Behind the Scenes at MySpace.com

Dan Farino
Chief Systems Architect
dan@myspace.com

Topics

- Architecture overview and history
- The stuff I get to work on (in the Windows world)
  - Monitoring
  - Administration

Topics

- Windows?!
  - It’s a good server (now leave me alone.)
  - However, the selection of tools for large-scale management is a *bit* sparse...

Where we started

Where we started

- The ideal growth scenario
  - Plan
  - Implement
  - Test
  - Go live
  - Monitor and collect ops data
  - Repeat

Where we started

- Our growth scenario:
  - Implement
  - Go live

- And while those are happening over and over:
  - Reboot servers
  - Throw hardware at performance issues
  - “Shotgun debugging”

Where we started

myspace® a place for friends

"Shotgun debugging":

Shotgun debugging is a process of making relatively undirected changes to software in the hope that a bug will be perturbed out of existence.

Where we started

- Why would anyone “shotgun debug”?
  - Don’t really know how to analyze and debug a problem
  - Need to resolve the problem *now* and collecting data for analysis would take too long

Where we started

- Web servers
  - Windows 2000 Server
  - IIS 5.0
  - ColdFusion 5
- Database servers
  - Windows 2000 Server
  - SQL Server 2000

Where we were

- Operationally
  - Batch files and robocopy for code deployment
  - “psexec” for remote admin script execution
  - Windows Performance Monitor for monitoring

Where we were

- Any sort of formal, automated QA process?
  - No.

Current architecture

Current architecture

- 4,500+ web servers
  - Windows 2003/IIS 6.0/ASP.NET
- 1,200+ “cache” servers
  - 64-bit Windows 2003
- 500+ database servers
  - 64-bit Windows 2003
  - SQL Server 2005

QA today

- Unit tests/automated testing
- We still don't "fuzz" the site nearly as thoroughly as our users do though
- There are still problems that happen only in production

QA today

- We need better operational data collection so that we know what cases we're not testing

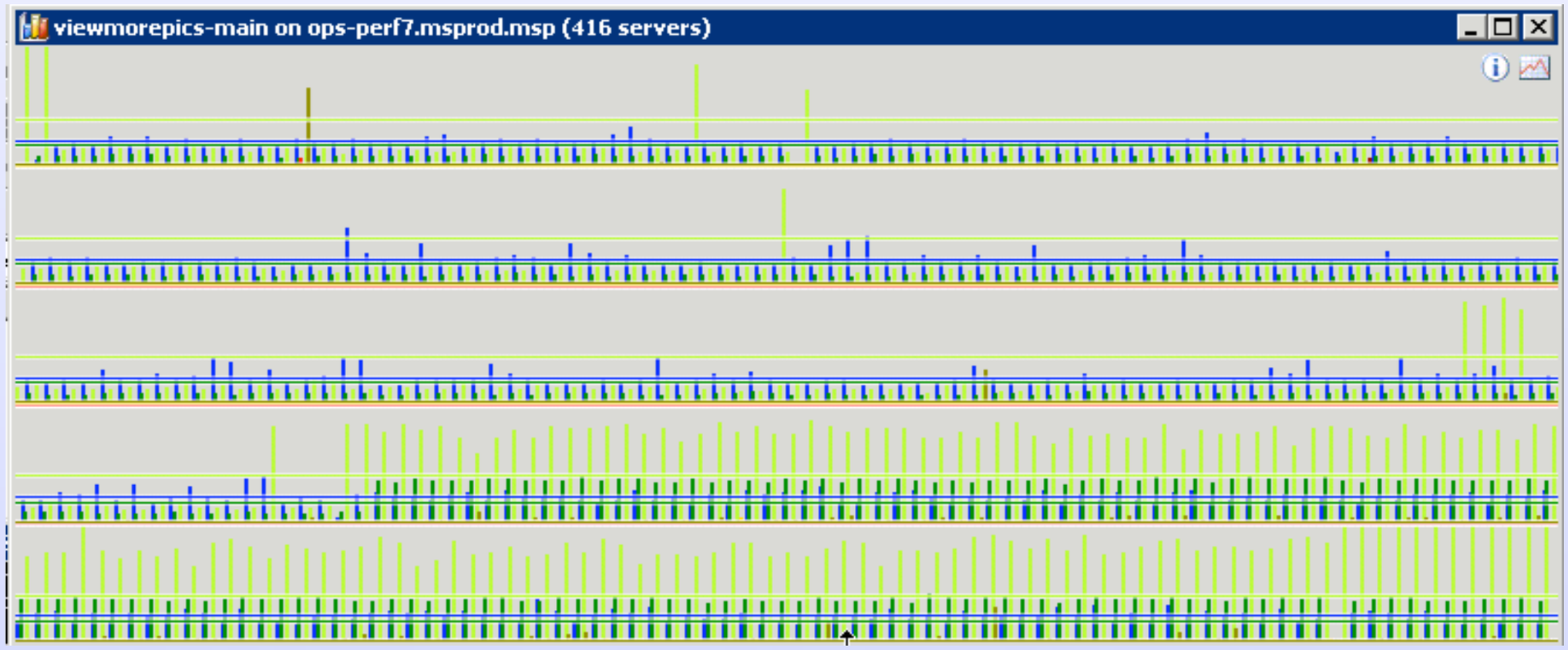
Operational Data Collection

Ops Data Collection

- Two general types of systems:
  - Static
    - Collect, store and alert based on pre-configured rules
  - Dynamic
    - Write an ad-hoc script or application to collect data for an immediate or one-off need

Ops Data Collection

- Our current “static” Windows Performance counter monitor:



Ops Data Collection

- Cons of static system:
  - Relatively central configuration managed by a small number of administrators
  - Bad for one-off requests: change the config, apply, wait for data
  - Developer's questions usually go unanswered

Ops Data Collection

- Developers looking at production?!
  - Developers like to see their creations come to life (I know I do)
  - The more a developer can see once their code goes live, the more they're going to know for V2

Ops Data Collection

- Cons of the dynamic system:
  - It's not really a "system" at all...it's an administrator running a script
  - Is a privileged operation: scripts are powerful and can potentially make changes to the system
  - Even run as a limited user, bad scripts can still DoS the system

Ops Data Collection

- Cons of the dynamic system:
  - One-shot data collection is possible but learning about *deltas* takes a lot more code (and polling, yuck)
  - Different custom-data collection tools that request the same data point cause duplicated network traffic

Ops Data Collection

- A recent example of an ad-hoc task using our current "dynamic" system:
  - `get-adservers | run-agent ps /e '"Version: $(gcm F:\file.dll | %{$_.FileVersionInfo.FileVersion})"' | select Host, Message`

Ops Data Collection

- Ideally, all operational data available in the entire server farm should be able to queried:
  - Safely
  - Instantly
  - With change-notification

Ops Data Collection

- I'd like to be able to do something like this:
  - SELECT CpuTime.*, ExceptionsPerSecond
    WHERE WebService.Status = 'UP'
    AND serving = 'profile.myspace.com'
    OR serving = 'home.myspace.com'

Ops Data Collection

I'd also to be able to leave that query "hanging" and be notified of changes like:

- A selected field has changed for a known data point
- A new server has come online and meets the criteria (or vice-versa)

Our new operational data collection platform

Ops Data Collection

- Our new operational data-subscription platform:
  - On-demand
  - Supports both “one-shot” and “persistent” modes
  - Can be used by non-privileged users

Ops Data Collection

- Our new operational data-subscription platform:
  - Eliminates the need for the consumer to poll for changes
  - If a data source requires polling, that operation is pushed as close to the source as possible

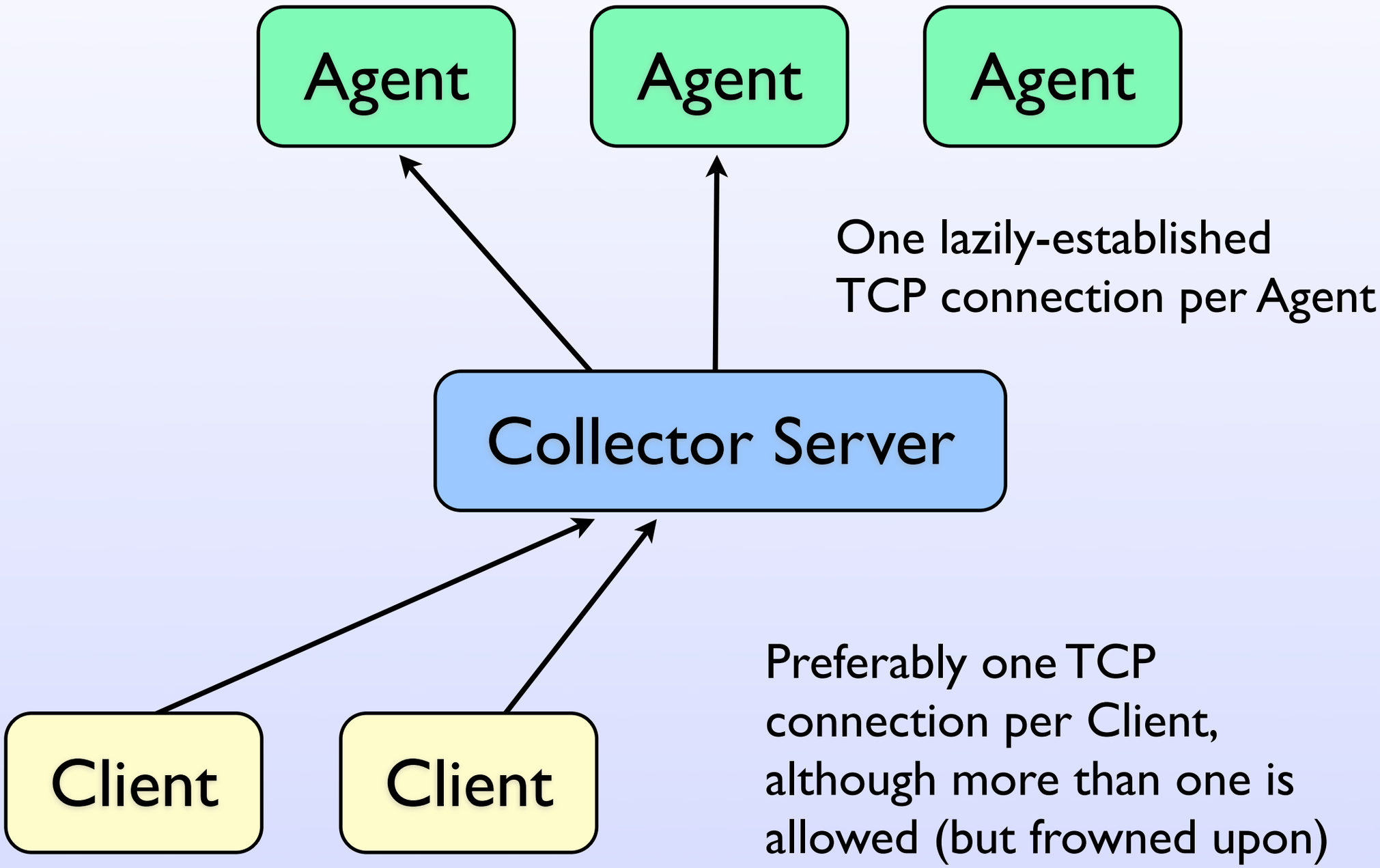
Ops Data Collection

- A Client makes *one* TCP connection to a “Collector” server
  - Can receive data related to thousands of servers via this one connection
- As long as the connection is up, the client is kept up-to-date

Ops Data Collection

- A little bit like:
  - Having all of the servers in a chat room and being able to talk to a selected subset of them at any time (over *one* connection)
  - Initial idea came from looking at using XMPP+ejabberd for command and control

Ops Data Collection



Ops Data Collection

- Provides:
  - Windows Performance Counters
  - WMI objects
    - Event logs
    - Hardware data
    - Custom WMI objects published from out-of-process
- Log file contents

Ops Data Collection

- Provides:
  - On Linux, plans are to hook into something like D-Bus so that processes can provide operational data to the Agent in a loosely-connected manner

Ops Data Collection

- The Collector service:
  - A Windows Service in C#
  - Completely async I/O (never blocks a thread)
  - Uses Microsoft's "Concurrency and Coordination Runtime"
- An Agent running on each host

Ops Data Collection

- Wire protocol is Google's Protocol Buffers
- Clients and Agents can be easily written in any of the languages for which there is a PB implementation

Ops Data Collection

- Why not use XMPP+ejabberd?
  - Wanted to use Protocol Buffers instead of XML
  - Wanted lazily-established TCP connections to the Agents
  - Wanted to see if C#+CCR could handle the load (yes it can)

Why develop a whole new platform?

Ops Data Collection

- Why develop something new?
  - There doesn't seem to be anything out there right now that fits the need
  - And my requirements also include free and open source...

Ops Data Collection

- To do it properly, you really need to be using 100% async I/O.
- Libraries that make this easy are relatively new
  - CCR, Twisted, GTask, Erlang

Ops Data Collection

- Most established products were written before the multi-core/async craze

Ops Data Collection

- What does it enable?
  - The individual that is actually interested in the data can gather it himself
  - No central config, no need to involve an administrator
  - This includes developers

Ops Data Collection

- What does it enable?
  - There is a very low “barrier to entry”
  - It’s almost like exploring a database with some ad-hoc SQL queries
  - “I wonder…” questions are easily answered without a lot of work

Ops Data Collection

- What does it enable?
  - Charting/alerting/data-archiving systems no longer concern themselves with the data-collection intricacies.
  - We can spend time writing the valuable code instead of rewriting the same plumbing every time

Ops Data Collection

- What does it provide?
  - Abstracts physical server-farm from the user
  - If you know machine names, great. But you can also say “all servers serving ‘profile.myspace.com’” or “all cache servers in Los Angeles”

Ops Data Collection

- What does it provide?
  - Guaranteed to keep you "up-to-date"
  - Get your initial set of data and then just wait for the deltas
  - Pushes polling as close to the source as possible

Ops Data Collection

- What does it provide?
  - Eliminates duplicate requests
  - Hundreds of clients can be monitoring the “% Processor Time” for a server and it will only be sent from that server *once* when it changes

Ops Data Collection

- What does it provide?
  - Only collects data that someone is currently asking for
  - This is how we avoid having explicit configuration on the server

Ops Data Collection

- Is this really a good way to do things?
- Having too much data *pushed* at you is a *bad* thing
- Being able to *pull* from a large selection of data points is a *good* thing

Ops Data Collection

- For developers, knowing that they will have access to instrumentation data *even in production* encourages more detailed instrumentation

Ops Data Collection

- Better instrumentation = more data available = more detailed feedback to QA and developers

Ops Data Collection

- Ease-of-use is a very high priority
- Easy and fun APIs encourage adoption

Ops Data Collection

- LINQ via C#:

```
var collector = new Collector(...);

var counters =
  from server in collector
  where server.subdomain = “www.myspace.com”
  select server.WindowsPerfCounter
  into counters
  where counters.category = “Processor”
  select server.Name, counters.Instance,
    counters.Value
```

Ops Data Collection

- LINQ via C# and CLINQ ("Continuous LINQ") = instant monitoring app (in about 10 lines of code):

```
var counters = ...
MainWpfWindow.MainGrid = counters;
// Go grab a beer
```

Ops Data Collection

- Tail a file across thousands of servers
  - With the filtering expression being run on the remote machines
  - At the same time as someone else is (with no duplicate lines being sent over the network)

Ops Data Collection

- Open source?
  - Hopefully!
- Other implementations?
  - I may write a GTask or Erlang version as a “weekend” project

Thank you!