ICON UK

the IBM user group

Back from the Dead: When Bad Code Kills a Good Server

Serdar Basegmez - Developi - @serdar_basegmez
William Malchisky Jr. - ESS - @BillMalchisky

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

Disclaimer

"Ladies and Gentlemen. The story you are about to see is true; the names have been changed to protect the innocent." --Dragnet

Disclaimer

"Ladies and Gentlemen. The story you are about to see is true; the names have been changed to protect the innocent." --Dragnet

For example... Acme Corporation is now referred to as Acme, Inc.

Setting Expectations

- What we will cover
  - Problem analysis
  - Troubleshooting skills
  - Best practices
  - The performance impact of suboptimal applications
- What we omitted
  - Boring, rambling, dry, lectures
  - Useless drivel

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

Customer Calls

- "We're having a problem. Can you help?"
- "Absolutely. What's happening?"
- "Our mission critical DB is really $%&@#$^& our users. It's way too slow. It takes less time to reboot [Windows 3.1 on an i386 with 32MB RAM] than to open a document."
- "Any idea what changed?"
- "We don't know. We have not touched the box."

Why Domino Servers Fail?

- Lack of expertise and/or knowledge
- Unplanned and/or unexpected expansion
- No dedicated Administrator
- No change management
- No monitoring
- Workaround overloading

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

"Round Up the Usual Suspects"

- While waiting for access, request the following

notes.ini	log.nsf
sh tasks	top
vmstat	iosys
df -h	Affected user(s) to server ping results
mount	swapon -s
Server NAB DB copy, sans users	

- Helps establish the level of criticality

Data, Data Everywhere

- Ran DCT - returned a few items, but nothing applicable to the performance issue experienced
- Checked Domino stats
  - Located a key issue - needle in haystack
  - SAI fluctuated wildly, frequently, plummeting to 18% for minutes on end
- Locate any recent NSD files for analysis

Quick Example - iostat, vmstat

```
malchw@san-domino:~$ iostat
Linux 3.13.0-83-generic (san-domino)   03/23/2016   _x86_64_  (8 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6.21    0.25    3.69    0.51    0.00   89.34

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              45.34      2075.44       778.25    6028264    2260469
sdb               0.36         1.52         0.03       4422         80
dm-0             24.51       117.04       186.80     339957     542584
dm-1             16.17       415.61        79.82    1207173     231836
dm-2             17.64      1540.92       511.61    4475713    1485996

malchw@san-domino:~$ vmstat
procs  -----------memory----------  ---swap-- -----io----  -system--  ------cpu-----
 r  b  swpd  free   buff   cache    si  so    bi    bo   in   cs   us  sy  id  wa st
 1  0     0 16943764 153144 7941660    0    0   262    98  144  681    6   4  89   1  0
```

Pro Tip on Data Collection

- Watch the server when nobody else does
- Lots of strange things happen on servers overnight
- Observed the system processing over one million records in :15 twice a week, at different times

  Example: no one at Acme, Inc. knew this occurred or why

Initial Data Analysis - OS

- Swap space 50% of installed memory
- Memory was under 1GB for mission critical server

  Several key DBs contained 100k+ docs
- Combination created page faulting plague further eroding performance
- System properly patched
- Free space adequate

Initial Data Analysis - Notes.ini

- Obvious but important data points
  - Server layout
  - Where items located
  - Recognized *server.id* file
  - Server tasks

    Contrast to *sh tasks* requested earlier
- No obvious problems

Initial Data Analysis - Amgr

- Agents running all hours of the night and day
- Agents running from DBs actively being compacted
- Agents running from DBs when updall and fixup running
- Not all scheduled agents needed to run all weekend

Initial Data Analysis - Log.sf

- Compact still running when updall Program fires-off
- Compact never finished before execution time ceiling hit

  Left largest DBs in a completely suboptimal state
- Connected to servers that did not exist
- Scheduled replication documents
  - Significant delays with replica synchronization
  - Ensured data never properly synchronized across domain
  - Certain connection documents only covered two DBs

Initial Data Analysis - DBs

- Several big DBs last fixup completed two years ago
- Most heavily used files 30-75% Used
- Many views means clicking one forces a new index build
- No design, document, or attachment compression
- Design server task citing non-existent templates

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

Tier 1 - OS

- Swap space - No set rule these days
  - 1.5x - 2.0x RAM is good rule of thumb
- Memory - 4GB per processor on busy servers
- VMware settings if available
  - Avoid temptation of too many processors
- Review partitions and free space

Additional OS Considerations

- Check that previous made system changes stick

  Unfamiliar servers can exhibit odd behavior

- Check IBM Technotes for any recent performance issues
- Once OS is working, check to ensure that virtualization is optimal

Tier 2 - Domino

- Space properly Program Documents

  Avoid overlap with agents and other Programs
- Pause agent schedule during maintenance
- Schedule a weekend to complete first full maintenance set

  First full compact will take much longer than you realize
- Create maintenance schedule of tasks agreed to by business line managers

  Ensures all needed jobs are available when needed

Additional Items to Fix

- Review all enabled Domino features to ensure that they function properly
  - Simple configuration miscues can impact negatively
  - Cluster replication unable to locate a cluster member
- DNS errors create lookup delays
- Remove unneeded, deprecated network ports

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

Where are We?

- Domino Admin handled the first level treatment
- Server performs well, but not good enough
- Triangulated the issue to a mission-critical application
- Now what?

Somebody Else's Code

```
def deepdream(net, base_img, iter_n=10, octave_n=4, octave_scale=1.4, end='inception_4c/output'
    # prepare base images for all octaves
    octaves = [preprocess(net, base_img)]
    for i in xrange(octave_n-1):
        octaves.append(nd.zoom(octaves[-1], (1, 1.0/octave_scale,1.0/octave_scale), order=1))

    src = net.blobs['data']
    detail = np.zeros_like(octaves[-1]) # allocate image for network-produced details
    for octave, octave_base in enumerate(octaves[::-1]):
        h, w = octave_base.shape[-2:]
        if octave > 0:
            # upscale details from the previous octave
            h1, w1 = detail.shape[-2:]
            detail = nd.zoom(detail, (1, 1.0*h/h1,1.0*w/w1), order=1)

        src.reshape(1,3,h,w) # resize the network's input image size
        src.data[0] = octave_base+detail
        for i in xrange(iter_n):
            make_step(net, end=end, clip=clip, **step_params)
```

Source: http://ryankennedy.io/running-the-deep-dream

Why Domino Apps Fail?

- Lack of expertise and/or knowledge
- Developers evolved from power users
- Architecture overloading
- Unplanned and/or unexpected expansion
- Undocumented code and/or business process
- No change management
- Quick & dirty development

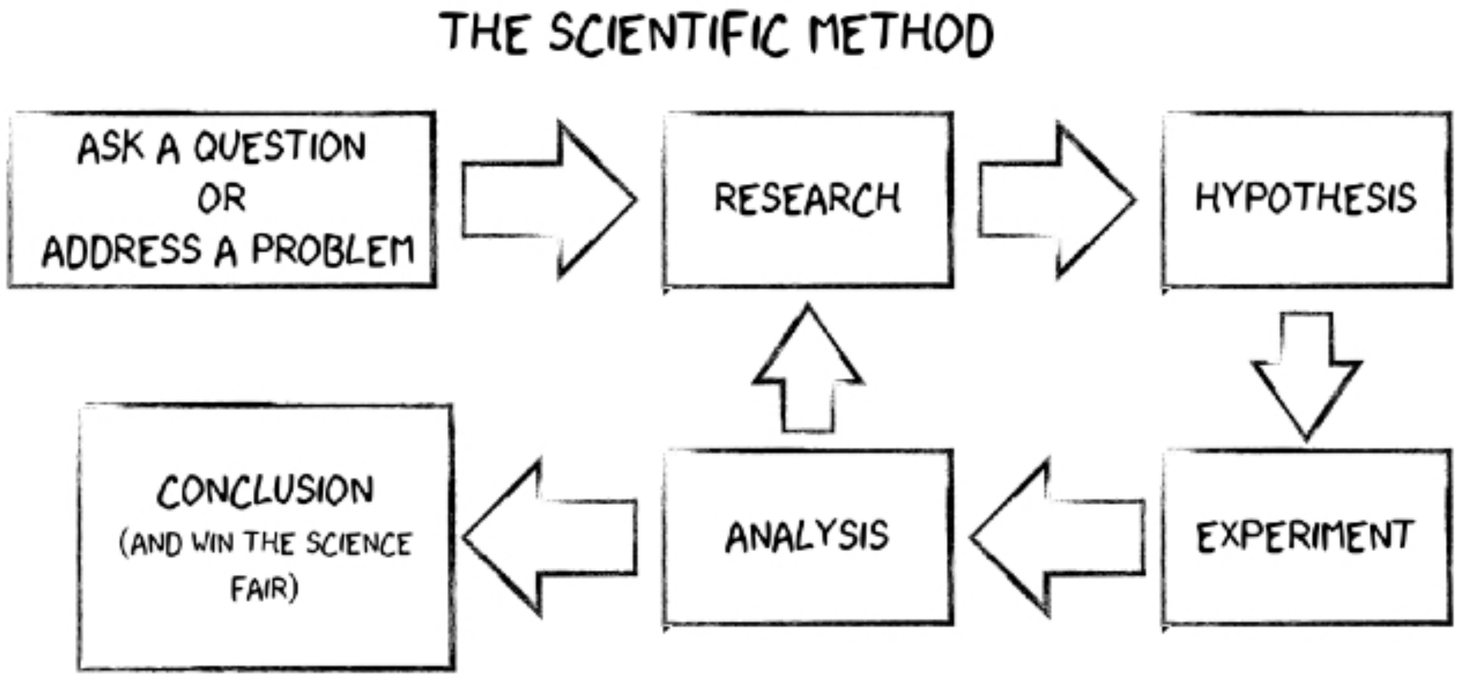
Developers vs Performance Issues

- There is no magic pill for finding a performance issue
- Many problems are circumstantial

  Depends on who/when/how…
- Repeating the problem on a controlled environment
  - Need for Proof!
  - The most difficult part of the task
- Need to be **systematical**

Science Just Works!

- Research and Assessment,
- Speculation for fixes,
- Experiment,
- Prove!



http://www.wired.com/2013/04/whats-wrong-with-the-scientific-method/

Methodology

Research	Symptoms (e.g. logs, performance data, etc.) Story (e.g. user input) Application code
Hypothesis	Speculation on possible reasons Search for ‘Usual Suspects’
Experiment	Testing for possible reasons
Analyze	Check symptoms if fixed
Conclusion	Issue validated and proved to be fixed.

Research & Assessment

- What to collect, based on the symptom;
  - CPU/memory load, hangs, spikes, crashes, etc.
  - All the time, the same time everyday or random?
  - Experienced by specific users?
- We are looking for a pattern between incidents.

Data Collection Checklist

- Log/NSD/Semaphore files
- Server configuration (inc. notes.ini)
- Server monitoring and statistics data
- Web logs (for web application issues)
- XPages and OSGi logs (for XPages specific issues)
- Application and dependencies

Isolate the Application

- Sometimes, even opening in DDE may cause issues!

  e.g. XPages components are automatically built

- Application code might have side effects

  e.g. Updating on another data source, adding audit logs, performance degradation on the server, etc.

- There will be dependencies
- Once isolated, we can start inspection…

Usual Suspects

- Database corruptions
- @Today/@Now in views
- Code snippets acting like an admin

  Updating views, replicating databases, running server commands
- Code snippets using the worst practices

  Search in a large database, wrong looping, etc.
- Anything that fits into the pattern if there is one

  e.g. An agent matching the incident timing

Nothing yet?
Digging deeper!

Team Up!

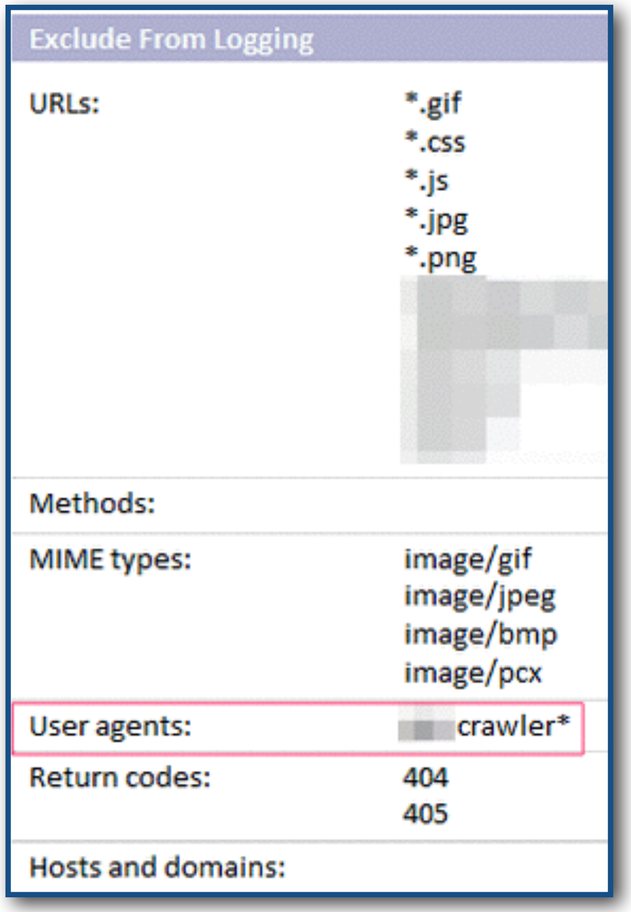
- Deeper investigation needs a teaming effort
- Admins and Developers should collaborate
  - A test setup to simulate the production environment
  - Intensive / Controlled debugging sessions in limited time windows
  - Sharing expertise
- Experimenting on production should be the last resort
- Once a repeatable error found, cooperate for a solution

Examples

Example Case - Analysis

- JVM Crash with the HTTP task
  - Random times
  - No pattern in the log
  - Memory dumps point a leak in the JVM Heap
- Inspected XPages applications, nothing found
- Triangulated the problem into one XPages app, following clues in intensive debugging on memory
- Isolated the application for a load test, nothing found
- Increased logging, to collect more data, no hope!

Example Case - Resolution



- Checked the server configuration and noticed
  - Logging data incomplete
  - Removed exclusions
- New logs pointed the problem:
  - Searching software crawling a specific page
  - Page generates state data and fills up the memory
- Simulated the same crash on the test environment
- One line of code fixed the issue

Another Case - Analysis

- A mission critical application at a bank
  - Web application with 2000+ users
  - CPU spikes and random hangs, mostly afternoon
  - Logs are clear, no crashes, no error messages
- Isolated the application, inspected the ‘usual suspects’
- Found a web agent updating a view!
- Triangulated the problem using web logs and SEMDEBUG
- But, cannot validate the issue on the test environment…

Another Case - Resolution

- Cooperated with the Domino Admin
- Detailed assessment on the server configuration
  - We found the issue!
  - “ServerTasksAt14” running an updall task.
  - Another Program file running Updall on a specific database, every 30 minutes
- Applied to the test platform, validated by a load test
- Problem solved!

Our Story in Forty-five Minutes

- Preface
- Chapter I - The Beginning
- Chapter 2 - Searching for Clues
- Chapter 3 - Creating a Solid Platform
- Chapter 4 - The Softside of Performance Gains
- The Final Chapter - Results

Quality Analysis Yields Quality Results

- Page faults reduced to zero
- General DB usage and administration tasks work well
- SAI now over 80%
- Weird overnight (agent) system operations resolved
- Key DBs have 93% used space now
- All DBs compressed: design, documents, all attachments
- Program documents, agents all adjusted: finish, no overlap

Note on Performance

When done properly, few users tend to notice the change, but if removed they will all complain

Teamwork vs. Performance

Neither an admin nor a developer

could solve all of these issues alone!

Bonus Slide

cooperteam	MartinScott
teamstudio	Ytria

- You can get help inspecting applications and servers
- teamstudio also sponsored ICON UK!

Serdar Başeğmez

- IBM Champion (2011 - 2016)
- Developi Information Systems, Istanbul
- Contributing…

  OpenNTF / LUGTR / LotusNotus.com

- Featured on…

  Engage UG, IBM Connect, ICON UK, NotesIn9…

- Also…

  Blogger and Podcaster on Scientific Skepticism

William Malchisky Jr.

- IBM Champion (2011 - 2016)
- Effective Software Solutions, LLC
- Co-founder of Linuxfest at Lotusphere/Connect
- Speaker at 25+ Lotus/IBM related events/LUGs
- Co-authored two IBM Redbooks
- Co-wrote the IBM Education Administration track for Domino 8.5

Follow Up - Contact Information

Serdar Basegmez

serdar.basegmez@developi.com

@serdar_basegmez
Skype: sbasegmez
Blog: lotusnotus.com

Bill Malchisky Jr.

william.malchisky@effectivesoftware.com

@billmalchisky
Skype: FairTaxBill
Blog: billmal.com

Questions and Answers