SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2015

Tuning an SMB server implementation

Mark Rabinovich
Visuality Systems Ltd.

Who are we?

- Visuality Systems Ltd. provides SMB solutions from 1998.
- NQE (E stands for Embedded) is an implementation of SMB client/server for the embedded world:
  - Consumer devices: printers, MFP, routers, smart devices, etc.
  - Industrial Automation, Medical, Aerospace and Defense
  - Anything else that is neither PC, MAC or Samba.
- NQ Storage is an SMB server implementation for Storage platforms.

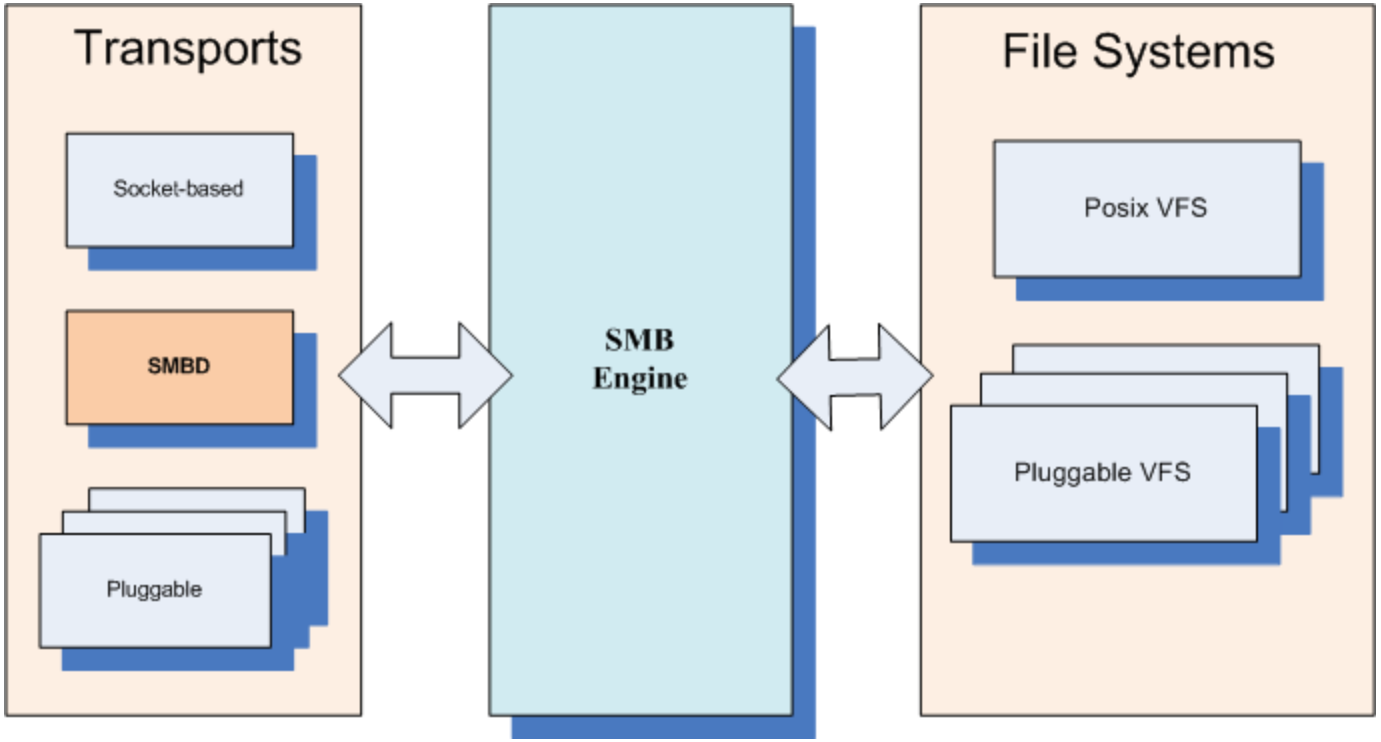
This presentation is about NQ Storage

Presentation Plan

- SMB Storage architecture highlights
- Performance factors
- Performance figures
- Tuning a server

Architecture

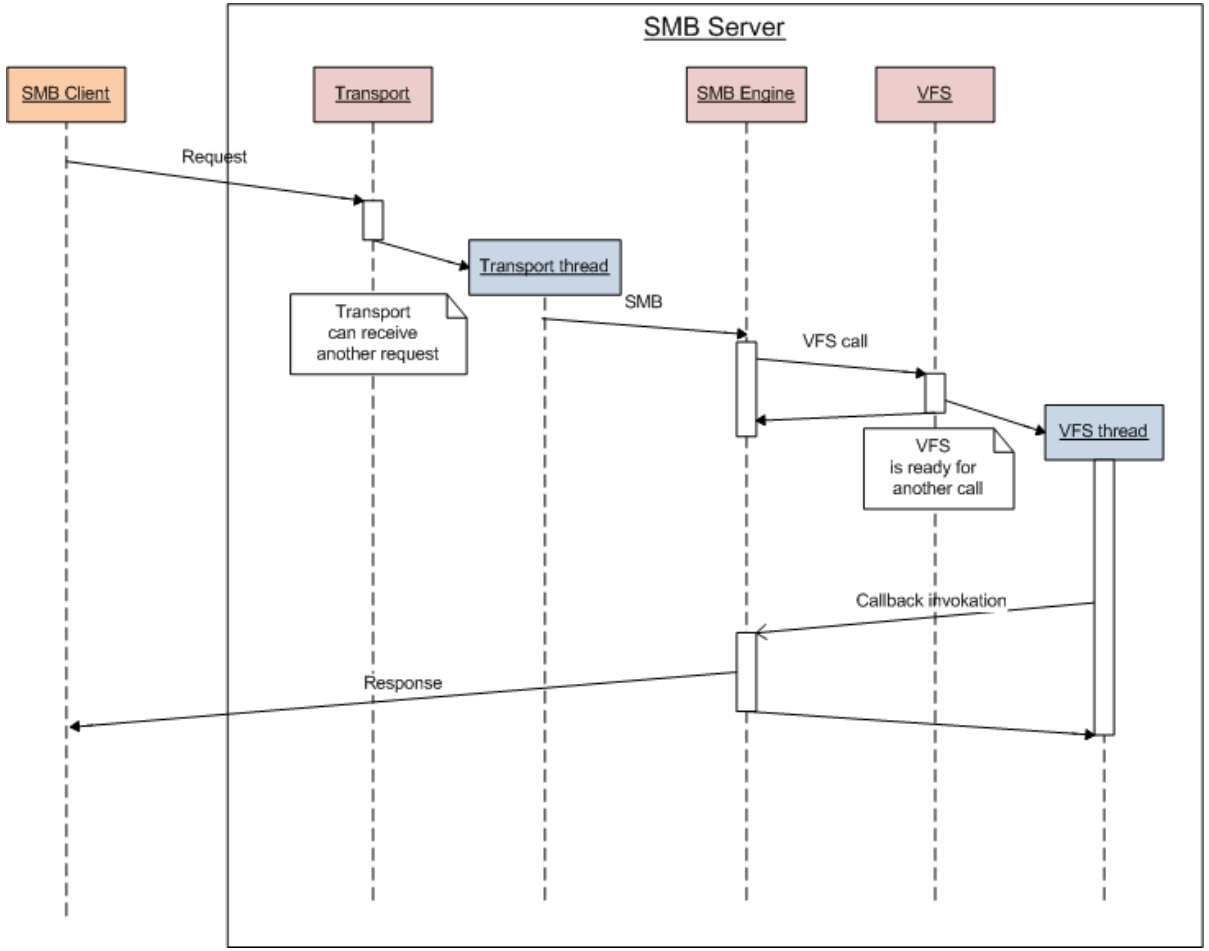
Architecture in general



Architecture explained

- Transport:
  - Responsible for receiving SMB requests and responding
  - Delegates requests to SMB Engine
  - TCP (socket) transport
  - SMBDirect (SMBD) transport over RDMA
  - More platform-dependent transports can be plugged in
- SMB Engine
  - Is responsible for parsing SMB requests and composing responses
  - Is responsible for internal SMB semantics (e.g. - IPC$)
- VFS
  - Responsible for file operations
  - Posix VFS implements basic VFS on top of the local OS
  - An external VFS can be plugged-in

SMB request flow



- Transport module handles concurrent requests
- VFS module handles concurrent calls

SMB flow explained

- This flow exposes the “async” case which is of a particular interest for this presentation.
- Transport receives a request and delegates it to a transport thread.
- SMB Engine parses the request and calls VFS.
- VFS may decide to delegate the call to a VFS thread.
- When finished, VFS invokes an SMB Engine’s callback which send the response. This call may happen in the context of a VFS thread.

Performance Factors

Platform factors

- CPU.
  - Frequency
  - Number of cores
  - Hyper-threading – effectively doubles the number of cores
- Network
  - Throughput (1Gb/s, 10Gbs/s, Infiniband, RoCE, etc.)
  - NIC offloading (different techniques)
  - RDMA offload
- Drive
  - HDD
  - SSD

Server parameters

- We assume that each transport has a thread pool.
  - Serve concurrent requests
- VFS components may use separate thread pools for:
  - Create
  - Read
  - Write
  - Time-consuming IOCTLs (set file info, trim, etc.)
  - Query Directory
  - Other meta-operations
- Credit window
- Other parameters

Credits

- The credit window should not be a factor. We can easily have enough buffers of 1MB. How many buffers will be enough?
- Satisfactory credit window is:

Max credits =
<num of effective cores> +
<NIC offload factor> +
<drive speed factor>
<overhead>

- "NIC offload factor" – how many SMBs can an adapter receive and store in its buffers. For simplicity we count receiving and do not consider transmitting.
- "Drive speed factor" – how many pending threads do we need to load the CPU while drive performs an I/O. .

<drive speed factor> = <memory access speed> / <drive speed>

Credits (cont.)

- Memory access speed (typical for DDR3) – 5000MB/s
- Drive speed (typical):
  - HDD 115MB/s
  - SSD 400 MB/s
- Example: 6 + 2 + .5000 / 115 + 5 = 56

- Is the above formula accurate?
  - NIC offload factor depends on hardware and it is not always easy to comprehend.
  - Drive speed factor varies
- If we could know the number of threads, credit window could be easily and accurately calculated

Credits (cont.)

- An alternative method - uses mostly software parameters

Max credits =
<transport threads> +
<max VFS threads> +
<NIC offload factor> +
<overhead>

- Example: 20 + 20 + 2 + 3 = 45.
- We still depend on the NIC offload factor

Thread pool size

The credit windows question may be transited to thread pool size(s).

How big?

- Big enough to utilize all cores of the CPU
- Not too big - bigger numbers lead to saturation.

Which numbers are optimal?

We will try to find tendencies

- Trying different scenarios
- Trying various parameters
- The server platform remains the same

Other parameters

- Buffer pre-allocation
  - SMB Request buffers
  - SMB Response buffers
  - RPC buffers

The optimal buffer pre-allocation may be calculated, while the optimal number of threads is not that easy to calculate.

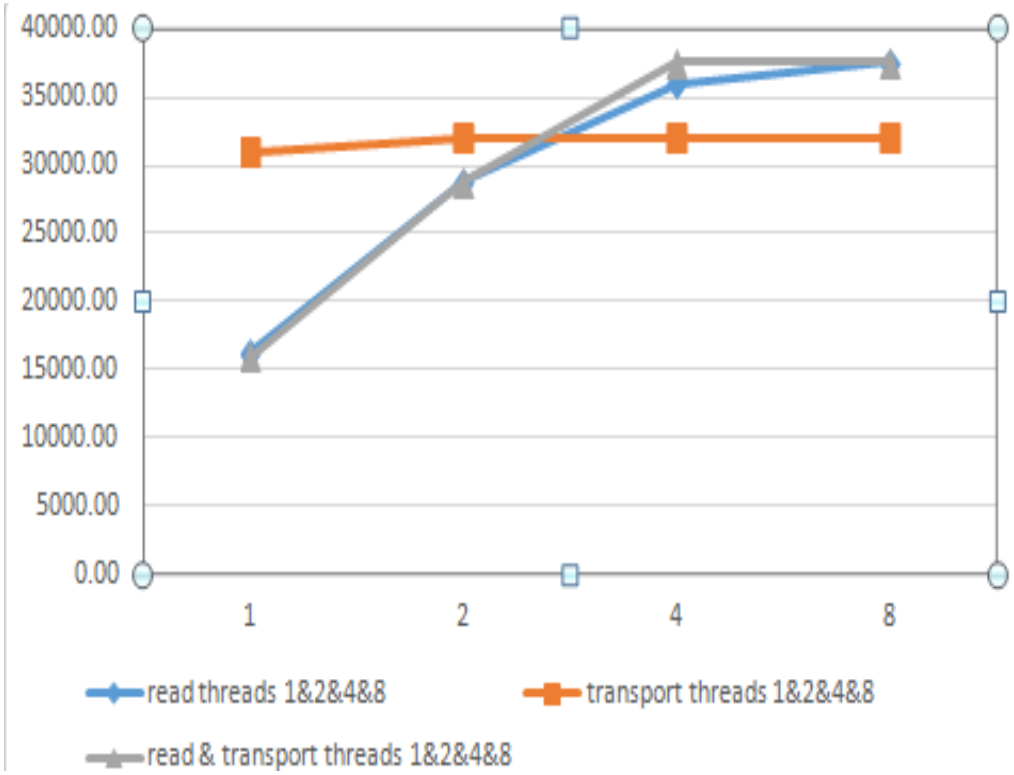
Performance Figures

Test platform

- HP ProLiant ML350P Generation 9
- Intel® Xeon® 1.90GHz/6-cores
- 1000GB HP HDD over SATA
- HP Ethernet 1Gb/s
- HP Ethernet 10Gb/s

Performance … by server threads (cont.)

Case: File download



Testware:

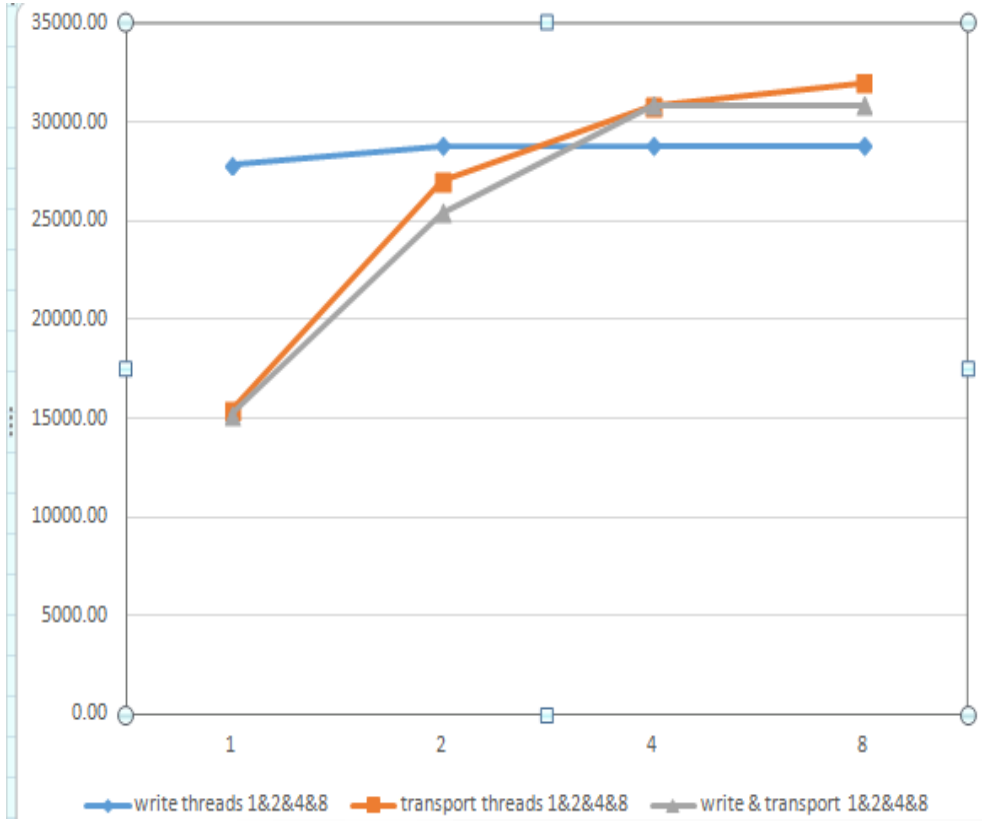
- SwiftTest, 20 users.
- 100MB file
- 64K packets

Legend:

- Increasing Read threads leaving Transport threads unchanged.
- Increasing Transport threads leaving Read threads unchanged.
- Increasing Transport and Read threads.

… by server threads (cont.)

Case: File upload



Testware:

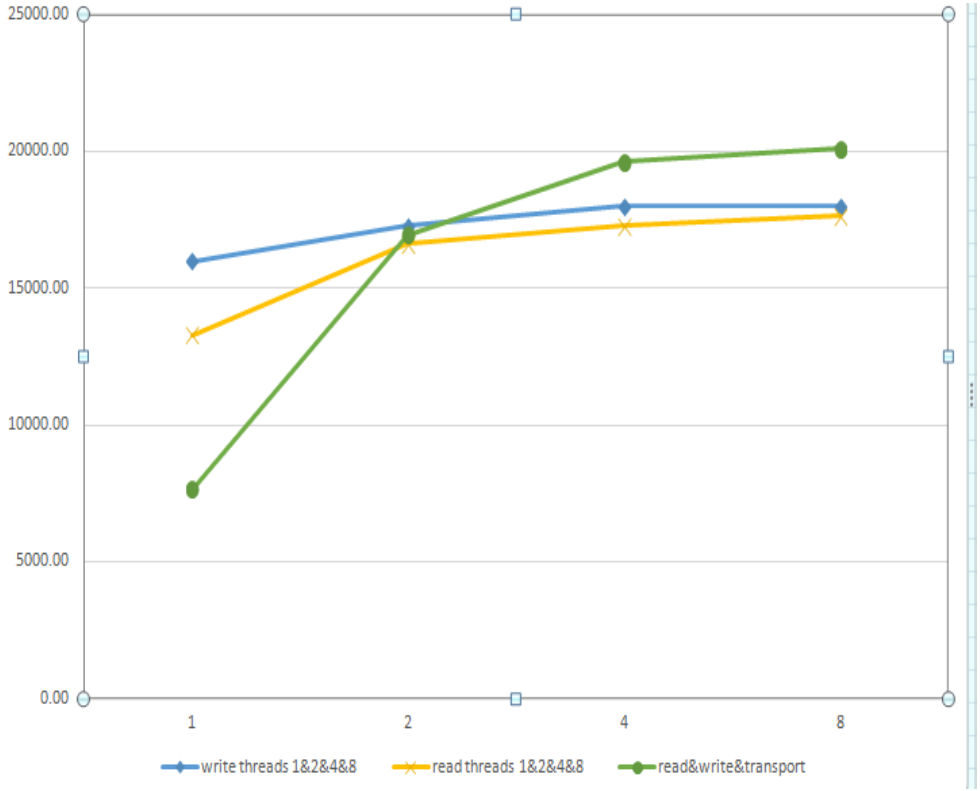
- SwiftTest, 20 users.
- 100MB file
- 64K packets

Legend:

- Increasing Write threads leaving Transport threads unchanged.
- Increasing Transport threads leaving Write threads unchanged.
- Increasing Transport and Write threads.

… by server threads (cont.)

Case: File upload/download mix



Testware:

- SwiftTest, 20 users.
- 100MB file
- 64K packets

Legend:

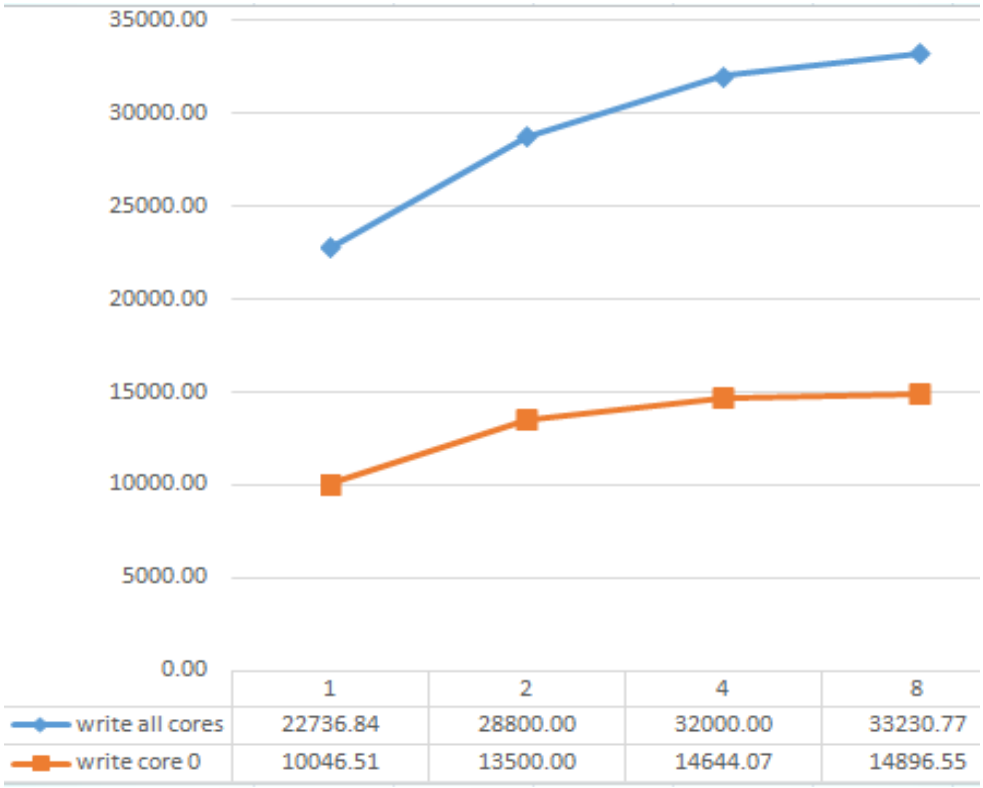
- Increasing Write threads leaving Transport and Read threads unchanged.
- Increasing Read threads leaving Write and Transport threads unchanged.
- Increasing Transport, Read and Write threads.

… by server threads (cont.)

- Adding too many threads does not help – “saturation”.
- Increasing transport threads alone does not help. Apparently, backend becomes the server’s bottleneck.
- Increasing VFS threads helps for read and write scenarios. We still need transport threads for the mixed case.
- Reading is more sensible to multiplexing than writing.

… by CPU cores

Case: File upload by CPU cores



	1	2	4	8
write all cores	22736.84	28800.00	32000.00	33230.77
write core 0	10046.51	13500.00	14644.07	14896.55

Testware:

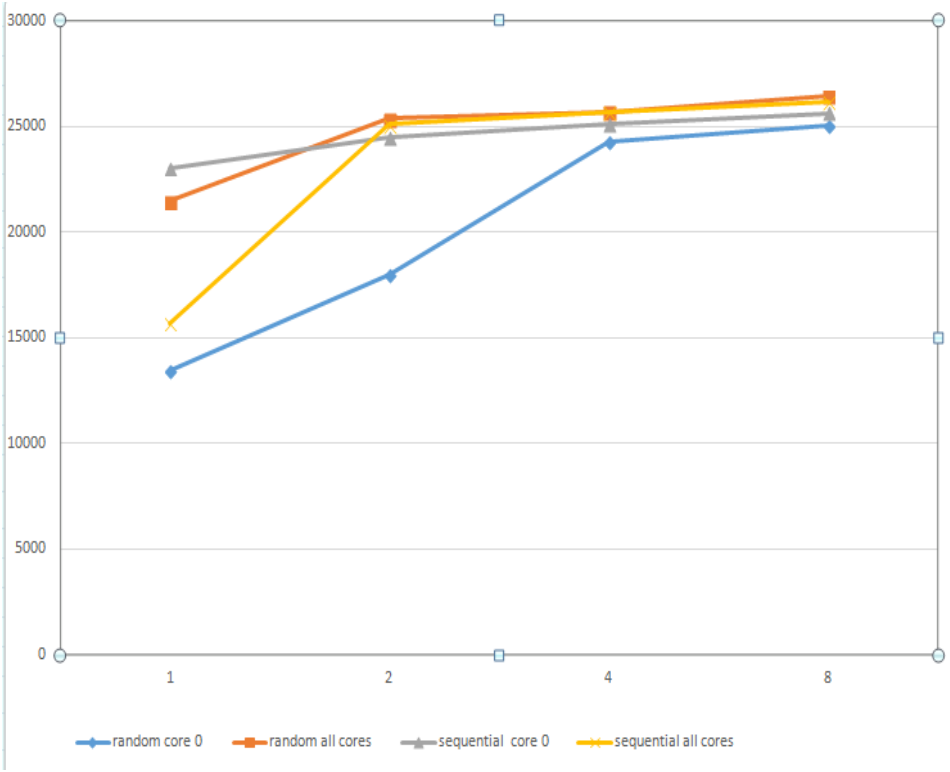
- SwiftTest, 20 users.
- 100MB file
- 64K packets

Legend:

- All cores.
- One core.

… by CPU cores (cont.)

Case: SQL Server traffic simulation



Testware:

- SQLIO.
- 60 sec run
- 4K packets
- 8 outstanding requests

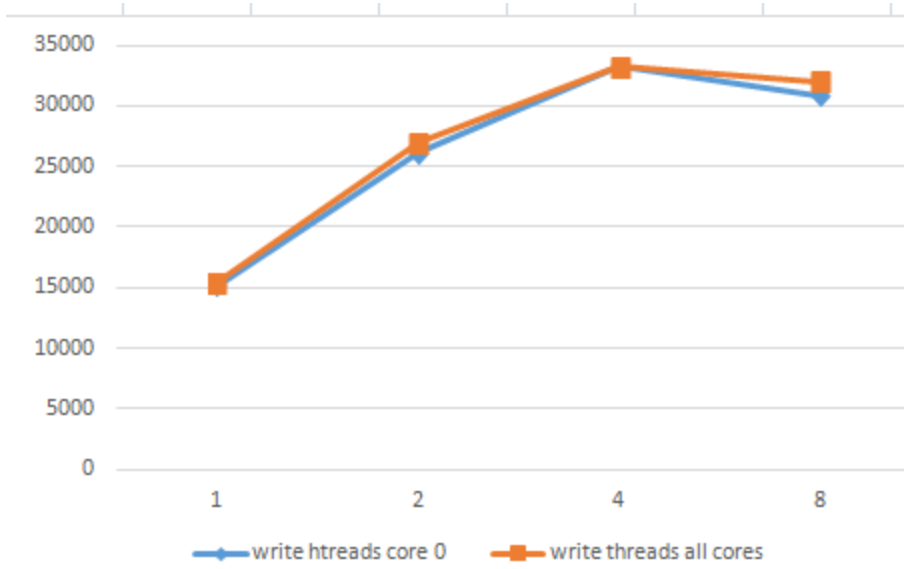
Legend:

1. Random file access with single core.
2. Random file access with six cores.
3. Sequential file access with single core.
4. Sequential file access with six cores

Both Transport, Read and Write threads are increasing.

… by CPU cores (cont.)

Case: Low load file uploading



Testware:

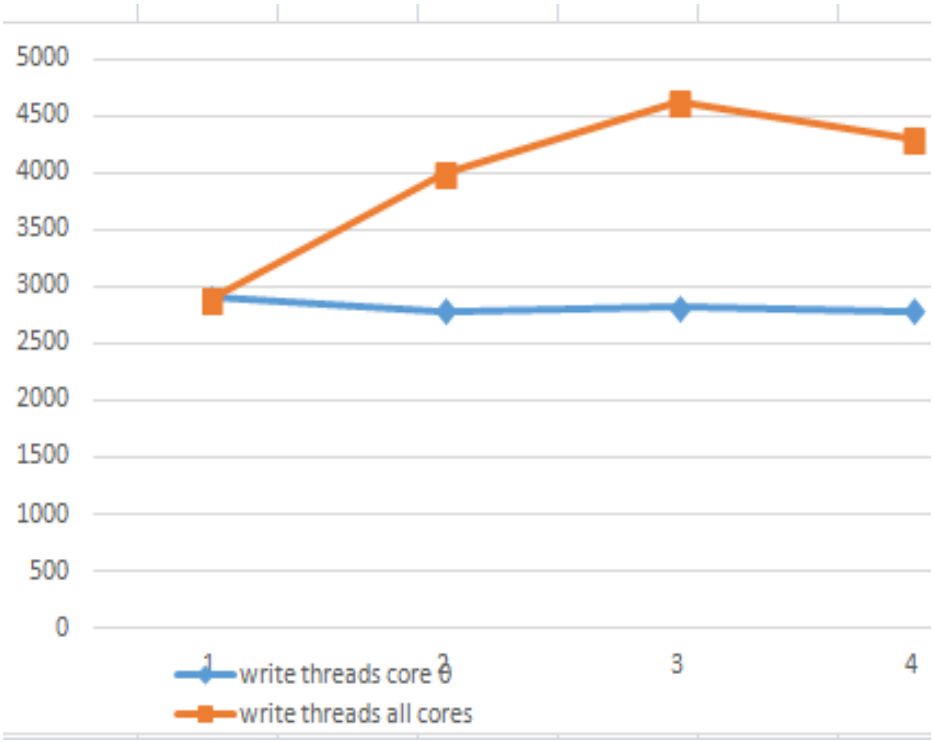
- SwiftTest, 20 users.
- 100MB file
- 64K packets

1. File uploading over multiple connections with a single core.
2. File uploading over multiple connections with six cores.

Both Transport, and Write threads are increasing.

… by CPU cores (cont.)

Case: High load file uploading



Testware:

- SwiftTest, 1000 users.
- 100MB file
- 64K packets

1. File uploading over multiple connections with a single core.
2. File uploading over multiple connections with six cores.

Both Transport, and Write threads are increasing.

… by CPU cores (cont.)

- More cores utilize more threads.
- One core can also (but less) benefit from threading. This apparently happens because some of them are locked on I/O.
- Server is more sensible to the number of threads when it comes to random access scenarios.
- Server is more sensible to the number of threads when it comes to smaller chunks.
- On a higher load a the number of cores becomes a more essential factor.

Tuning a Server

Platforms

Typical server platforms:

- SOHO NAS: ARM 1.2GHz Dual Core, HDD
- Mid-level storage: Atom® 2.13GHz Quad Core, HDD
- Top-end storage: Intel® Xeon® 3.4GHz Quad Core, SSD

Apparently, the ideal parameter numbers will be different for each of these categories. Even in the same category (e.g., - Top-end storage) the numbers may differ between two different platforms.

We need a methodology of choosing ideal parameters

The challenge

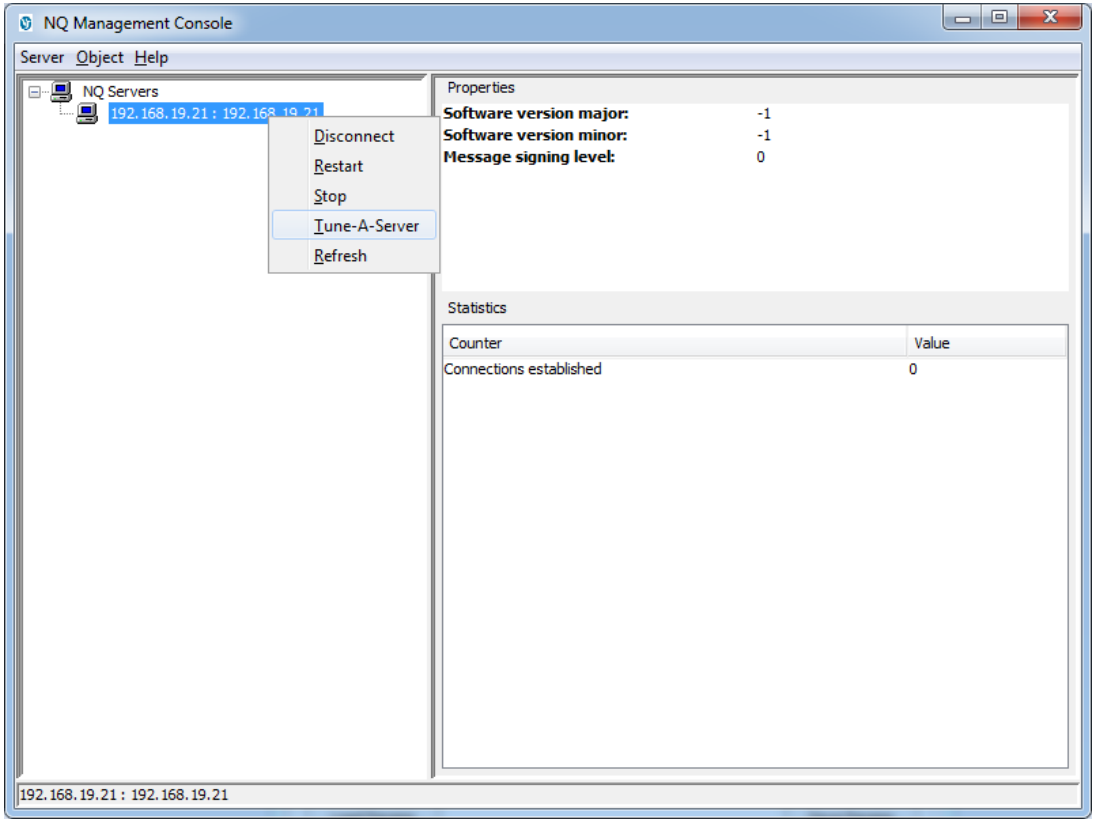
- Find out the optimal parameters.
- Do it fast or, at least, do it automatically.
- Do it reliably

Solution example – Tune-a-Server

Tune-a-Server

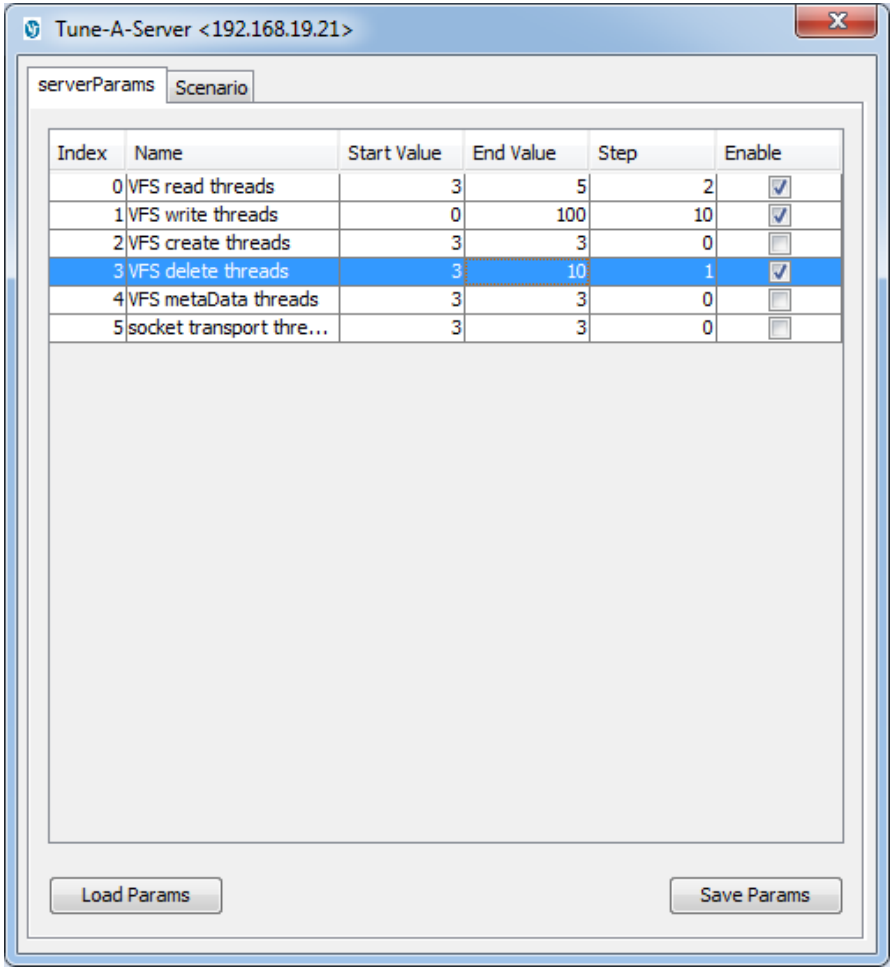
- Is a part of NQ Server Management
- Enumerates each single combination of the server parameters.
- Runs a set of tests for each combination
  - Test result is the time it takes to run the test. The less the better.
  - Each test have a weight.
- Calculates the result for each parameter combination by applying test weights to test results.

Tune-a-Server (cont.)



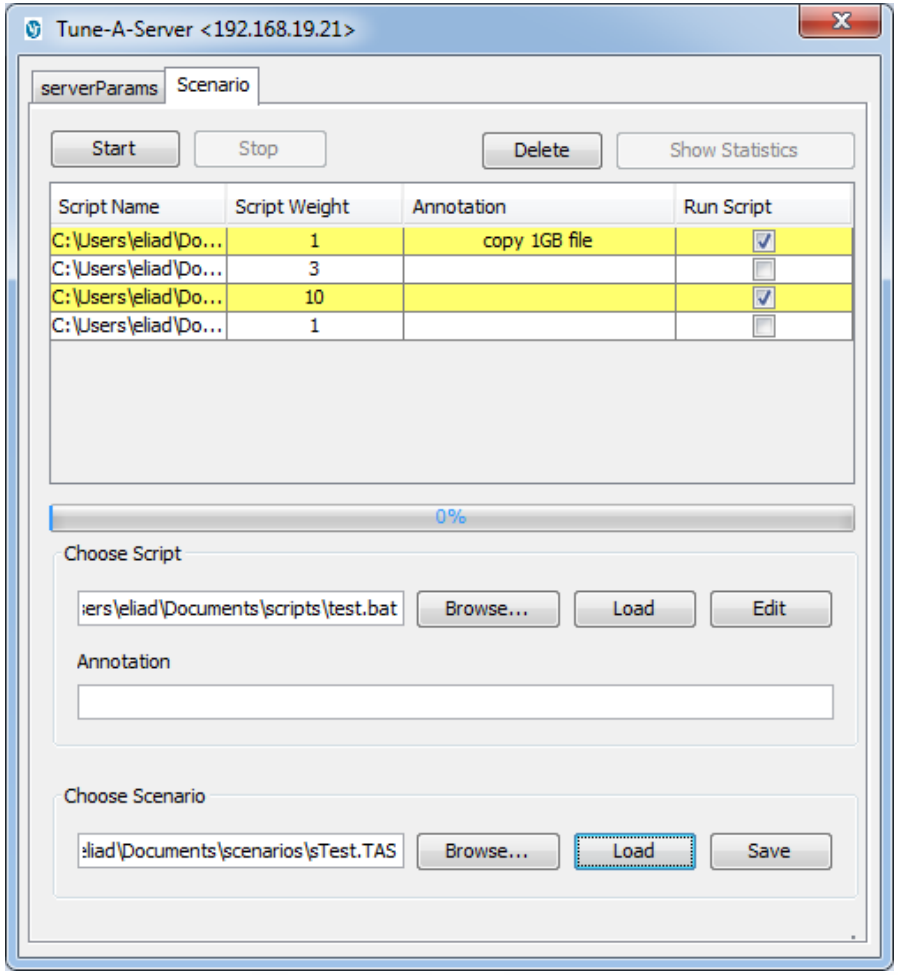
- Choose Tune-a-Server from NQ Management Console
- This will start a Wizard

Tune-a-Server (cont.)



- Select the parameters of the interest.
- For each of them choose the range
- Other parameters will keep their default value.

Tune-a-Server (cont.)

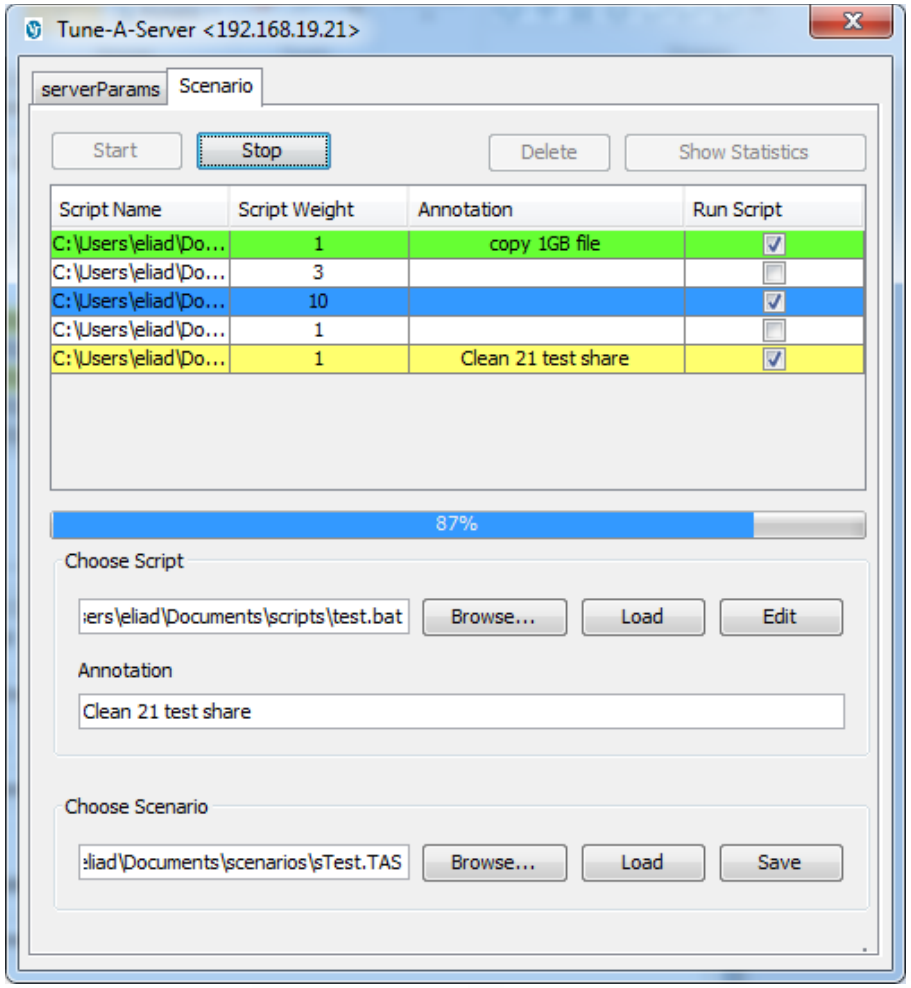


- Select scripts to run.
  - Script == test
- Choose script weights.
  - Weight means script importance.

Tune-a-Server (cont.)

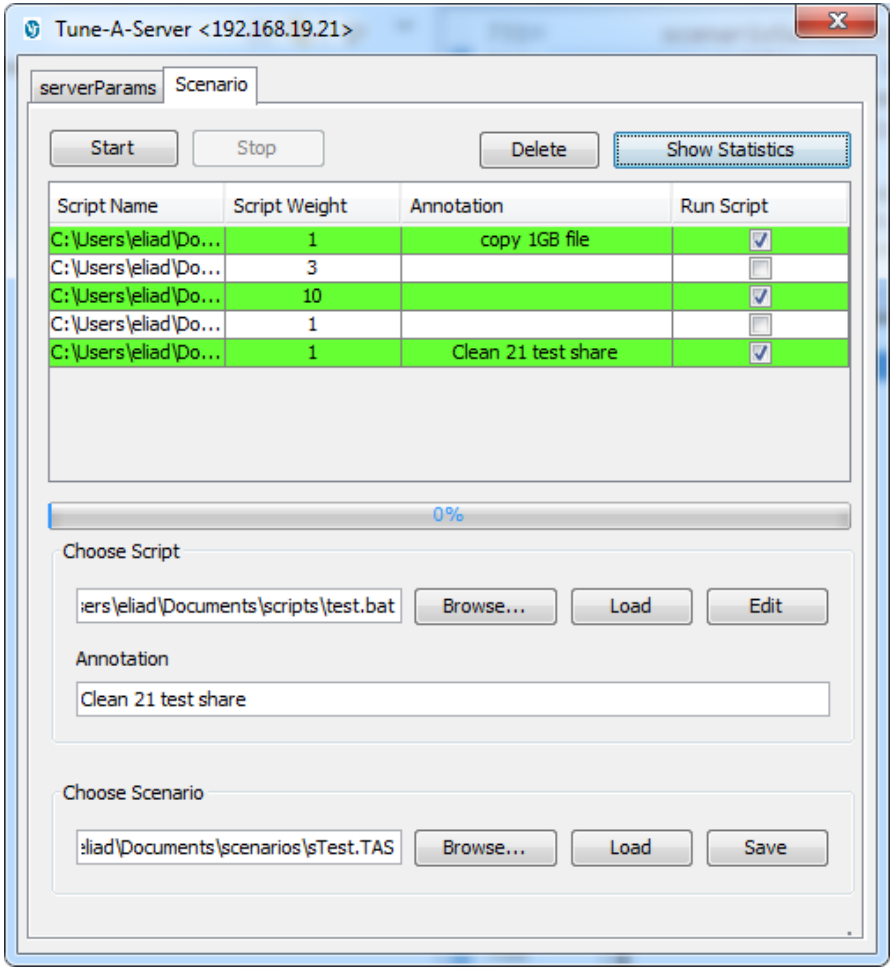
- Script explained:
  - A script runs a test.
  - A script is expected to emulate a use case scenario
  - A script can be any program whose results evaluate in the time of run. The less the time, the better the result.
  - Each of the experiments from this presentation may be a script.
  - We need more script ideas – suggestions welcome.
- Weight explain:
  - For instance: a tool like SQLIO has bigger weight than file upload/download since it emulates more practical case(s).
  - Writing is more sensible to threading than reading (see performance results above). We can consider giving more weight for the upload script.

Tune-a-Server (cont.)



- Run scripts.
- This may take long – we usually run overnight.

Tune-a-Server (cont.)



- When done, the results may be exported to Excel and analyzed.

Your feedback is very important for us.

Thank you

markr@visualitynq.com