



STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2016

Early Developer Experiences Extending Windows Nano Server with Enterprise Fibre Channel Software Target

Terry Spear
IOLogix, Inc.

IOLogix contact: info@IOLogix.com

Presenter contact: Terry.Spear@IOLogix.com

Introduction - Presenter

- ❑ Presenter Background
 - ❑ Co-Founder of Language Resources - micro processor dev tools
 - ❑ Founder of Breakthrough Systems, HW&OS independent storage protocol and device/virtual device software stacks for Storage OEMs
 - ❑ SCSI controller firmware for first helical scan tape drive
 - ❑ SCSI controller firmware for first dual port tape drive
 - ❑ Designed first software OEM virtual tape library
 - ❑ First Software Target Port design, and n-port Target abstraction

Introduction - Company

- ❑ IOLogix, Inc.
 - ❑ Develops and licenses Windows Server based Storage Software Appliances to OEMs and System Integrators
 - ❑ Spin out of Breakthrough Systems which has provided licensed semi-custom RTOS-based device firmware and Linux/Solaris based virtual device software to storage OEMs for 20+ years

Overview

- ❑ Why Windows Nano Server 2016 is relevant
- ❑ Nano Server Storage Application– Enterprise Fibre Channel Software Target
- ❑ Nano Server Application Development Experience
 - ❑ General Nano App Development Process and Tools
 - ❑ Case Study: Port of commercial Linux FC Software Target storage application to Nano OS
- ❑ Storage App integration with Nano OS storage features
- ❑ First-Look at Statistics and Performance
- ❑ Conclusion and Reference Materials

Why Nano Server is relevant as Enterprise Storage Server platform OS

- ❑ Designed as Cloud infrastructure, light weight, universal server platform OS
- ❑ Cloud Infrastructure Storage Server for VMs is a primary use case
- ❑ Nano Server (1/8th size) core of Windows Server 2016 (**WS-2016**) with same **enterprise storage features** available as installable add-on packages
- ❑ Nano can be installed with just the platform and storage features required for use case specific configurations
- ❑ Highest Security Windows platform available to date
- ❑ Deploy anywhere: Physical Hardware or Virtual Machines (Hyper-V, VMware, Virtual Box)

Nano Server Enterprise Storage Features

- ❑ Windows Server 2016 general improvements to overall reliability, availability, performance of OS required for Enterprise server platforms
- ❑ Enterprise Storage feature build-out
 - ❑ Storage Spaces (Software Defined Storage)
 - ❑ Resilient storage using commodity drives
 - ❑ Continuous Availability, Thin Provisioning, Tiered storage devices
 - ❑ SAS, SATA HDD and SSD, NVMe Flash device support
 - ❑ Deduplication
 - ❑ Replication (Async and Sync)
 - ❑ SMBv3.1 and NFSv4.1 File Servers
 - ❑ iSCSI Block Server

Nano Storage Application Development

Case - Fibre Channel Software Target

- ❑ Invited by Microsoft to construct first 3rd-party storage application for Nano as a development test case
- ❑ ***Fibre Channel Software Target (FCST)*** is THE missing Enterprise Storage Server feature
- ❑ FCST is a C/C++, User Mode Application
- ❑ FCST, based on a proven OS and HW independent proprietary code base, has run on x86, x64, ARM, PPC, MIPS and SPARC hardware with Linux/Solaris OS
- ❑ Various OEM custom configurations have shipped in Symantec, HP, Quantum, Veritas, etc. OEM retail Disk Array and Virtual/Physical Tape Library products
- ❑ Sizable application - 150K+ lines of source code

7

Nano Server Application Development Overview

- ❑ Phase 1:
 - ❑ Install Windows Server 2016 Standard, Desktop deployment option and Dev Tooling
 - ❑ Port FCST App to WS-2016: Build, Debug, Basic Tests
 - ❑ Integrate WS-2016 Enterprise Storage Features
 - ❑ System test
- ❑ Phase 2:
 - ❑ Install Windows Server 2016 Standard, Nano Server deployment option and Nano Specific Tools
 - ❑ Port to Nano Server 2016: Build, Debug, Basic Tests
 - ❑ Integrate Nano Server Enterprise Storage Features
 - ❑ System test
- ❑ Initially looked straightforward.....

Windows Server 2016 Install & Configure

- ❑ Select Dev Workstation x64 PC HW (server, desktop)
- ❑ Download Windows Server 2016 install media
- ❑ Install WS-2016 selecting “Standard with Desktop” option
- ❑ Install all storage related roles and features
- ❑ Install Visual Studio 2015 Community Dev Tools

❑ Note: web links to sources of above underlined references are provided at end of deck

Port Storage App to Windows Server 2016

- ❑ Visual Studio 2015 C/C++ tool chain
 - ❑ Built the FCST storage application executables from source
 - ❑ Applied porting corrections to source
 - ❑ Performed basic tests of FCST application on Dev Machine under local debugger then standalone executables
- ❑ Build and install custom driver
- ❑ Perform OS and Application HW/SW Integration, and Basic Tests
- ❑ ~6 weeks / ~60 developer days

Port to Nano Server – Not so Simple!!

- ❑ Harder than Linux to Windows Server Port!
- ❑ The well known server OS has been “refactored” by eliminating redundant APIs and components
- ❑ Rough first impressions – current Windows developer knowledge and practices provided a good framework but suddenly incomplete and different
- ❑ The key insight was expressed by one of our engineers: *“Consider Nano Server an **embedded OS** and use those Dev methodologies, then it all starts making sense!”*
- ❑ Instead of the first impression that it was more complicated, Nano OS was actually radically less complex. Where there had been 2-4 system APIs that performed similar functionality there was now exactly ONE.

Nano Server Install – Configure then Install!!

- ❑ Nano configuration steps must be performed on disk image prior to first boot
- ❑ Nano Server primary use case is as a cloud Windows VM OS – no traditional DVD/ISO based install method
- ❑ Nano Server install files are included in the Windows Server 2016 download image (ISO) in new \Nano folder
- ❑ We used the procedures in Getting Started with Nano Server article to Install Nano Server to a bootable virtual disk (VHD) image file and setup as a secondary boot on a Windows 10 Development PC

Nano Server Drivers – Good News

- ❑ Nano Server includes a large OEM “InBox” driver package which should always be installed for early development
- ❑ Install any missing and application specific drivers using Getting Started with Nano instructions
 - ❑ Install & test new drivers on the WS-2016 Desktop Dev system, helps if it is exactly same HW as the Nano Server system HW
 - ❑ Copy any Nano required additional HW drivers from the Dev system Driver Store and install on Nano VHD image using procedures in Getting Started with Nano
 - ❑ Boot Nano, check for missing drivers and repeat if necessary

Nano Drivers – Bad News

- ❑ Easy was only for “InBox” OEM drivers
- ❑ What went wrong: custom driver Nano API set issues for existing drivers, driver custom installers are not supported, only INF file driver installs
- ❑ Limited driver troubleshooting tools
- ❑ Driver must be added to Nano Server Driver Store prior to 1st boot, PnP driver install ran only once at first boot
- ❑ Some of our drivers had to be EV signed to load into kernel to overcome a ranking issue with an InBox driver

Nano Install - Learning Initial References

- ❑ Review and Practice Nano OS Configure and Install to VHD following directions in this article [Getting Started with Nano Server](https://technet.microsoft.com/windows-server-docs/get-started/windows-server-2016-technical-preview-5)
<https://technet.microsoft.com/windows-server-docs/get-started/windows-server-2016-technical-preview-5>
- ❑ Hint - if any step fails, revert to exact replication of the examples shown in document before trying other variations
- ❑ Initially use “**-Development**” option in Nano VHD build step to lower driver signing requirements for custom driver development

Nano Sever boot and management

As described in [Getting Started with Nano Server](#)

- ❑ Perform a physical HW Install using Nano VHD boot disk to an HW system identical to Dev system
- ❑ Connect local Keyboard and Video devices
- ❑ Boot Nano system, selecting the new Nano boot option
- ❑ Login into Nano and explore the displayed Emergency Management Services (EMS) console – navigate to network settings and record the Nano system IP address
- ❑ On Dev machine open a remote PowerShell session on the Nano system and explore what is available in Nano PowerShell – many familiar utilities and features are not present

Nano Server Application Development Platform Setup

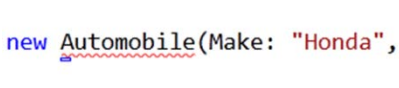
- ❑ Use previously setup WS-2016 Dev machine as Nano Dev machine
- ❑ Download and Install Visual Studio Nano Server Application Template extension and additional Nano development tools (e.g. Nano API scanner) using references in Developing Native Apps on Nano Server article:

<https://blogs.technet.microsoft.com/nanoserver/2016/04/27/developing-native-apps-on-nano-server/>

Porting App to Nano – Guidelines I

- ❑ Complete App initial development/port & test on Windows Server 2016 Desktop
- ❑ Follow guidelines and perform Nano Development setup steps in Nano Server Native Application Development article
- ❑ Convert application's Visual Studio project to use the newly installed Visual Studio Nano App Template extension
- ❑ Remove all User Interface components from Nano App build – NO GUI or console input available for Nano Apps!

Porting App to Nano – Guidelines II

- Identify Nano incompatible Windows API usage and refactor application API usage to use only the Nano supported API subset
- Use Visual Studio Nano App Template Extension “red squiggly”  marks identifying Nano API compatibility errors in source code, then refactor API call to the corresponding Nano compatible API
- Use the Nano API compatibility scan tool to identify remaining API use errors in the .exe and .dll files

Porting App to Nano – Guidelines III

- ❑ Boot Nano test system, use procedures in Nano Server Native Application Development to
 - ❑ Configure Visual Studio for remote debugging session
 - ❑ Start remote debugger and copy App executable file(s) to Nano system disk
 - ❑ Begin remote debugger execution check application module load window for load errors
 - ❑ Drivers must be refactored as necessary to only use the Nano APIs and rebuilt using WDK
 - ❑ After successful install to the Nano OS Driver Store and the 1st boot has occurred check the setupapi.log and pnpdevice command to verify the driver loaded without errors

Porting FCST App to Nano Server – I

- ❑ Performed the step by step Nano App build instructions provided in Developing Native Apps on Nano article
- ❑ Corrected Application Nano incompatible API usage in source code as identified by Visual Studio Nano extension by refactoring – substituting APIs called and parameters passed
- ❑ Corrected several link time API errors reported, again by refactoring the source code to use only Nano supported functional equivalent APIs
- ❑ Used Nano Server API Scan tool to check application .exe and .dll files for Nano incompatible API use – discovered additional network client-side code that was not required and was removed eliminating the API violations

Porting FCST App to Nano Server – II

- ❑ Started the app in the debugger, ran it locally (default) on the WS-2016 Desktop development machine
- ❑ Checked for missing and DLL load errors in debugger module load window
- ❑ During execution additional API usage errors were reported that required additional source refactoring to resolve (e.g. unsupported printf formats)
- ❑ More surprises: No stdin, stderr – caused runtime errors and “hangs”

Porting FCST App to Nano Server – III

- ❑ Achieved clean Nano App build, link, load, with all API issues resolved and execution on local machine without runtime errors
- ❑ Changed debugger settings to enable remote debugging on the Nano machine
- ❑ Copied App binary executable and debug symbols to Nano disk Application folder and started debug execution
- ❑ Corrected several additional load and run time errors (API references to DLLs not present in Nano) to get application running without errors on Nano OS

Porting FCST App to Nano Server – IV

- ❑ FC Software Target App User Interface required architectural refactoring in order to implement a command line interface
- ❑ UI component already met Nano requirements by using a remote win32 executable running on the Development Machine using a XML-RPC https network connection to the FC Software Target application running on the Nano Server machine

Integration of Nano Server Storage features

- ❑ Basic Integration tests were performed with available Nano Server storage resources to prove feasibility of various disk class media as virtual FC LUNs media.
 - ❑ Memory Channel Storage simulator1 (physical DRAM)
 - ❑ Memory Channel Storage simulator2 (3rd party, not available on Nano)
 - ❑ VHD files located on NTFS formatted physical media:
 - ❑ SATA/SAS HDD/Flash, NVMe Flash
 - ❑ Storage Spaces provisioned volumes
 - ❑ Remote SMB3.1 and NFS4.1 file server virtual volumes
 - ❑ Physical devices

Windows Server 2016 Storage Features Integration Test Results

- ❑ Basic Integration tests across all media types confirmed feasibility for use as FC LUN media subject to additional qualifying performance and reliability testing on specific storage server HW configurations
- ❑ Physical storage device access tests revealed non uniform API implementations that required minor code changes to accommodate
- ❑ Initial integration tests determined that Storage Feature availability, feasibility and issues were the same between Window Server 2016 Standard and Nano

Integration of Windows Server 2016 Advanced Storage Features - TBD

- ❑ Integration Test FC Software Target LUN volume VHD remote replication by Nano Replica feature
- ❑ Integration test of Nano Software Defined Storage advanced features: Thin Provisioning, Storage Device Tiers, etc.
- ❑ Integration of Nano Software Defined Storage features management API into Software Defined FC Software Target management API and UI

Nano Server FC Software Target Prototype1

- ❑ FC Software Target Proto_1 – use case: Portable 2U SFF Demo System
 - ❑ I7-5820, 3.3Ghz 6c,15M LLC, 16GB DDR4 RAM
 - ❑ 2 QLE2562 2 port 8gFC HBAs (Host Bus Adapters)
 - ❑ Available Storage devices:
 - ❑ SATA SSD
 - ❑ NVMe M.2 Flash
 - ❑ MemoryChannelStorage
 - ❑ Simulator1 (Physical RAMDisk) ,
 - ❑ Simulator2 (IOStack RAMDisk)
 - ❑ SMB3 and NTFS4 file server 1, 10gbE

Nano Server FC Software Target Test Configuration Proto_1



8GbFC 4 lanes
(direct cabled)
FC IO traffic only



1/10GbE LAN

Test FC Host - IOmeter work load gen
Windows Server 2016 (Desktop)
Dell i7-6700, 4c, 3.4Ghz, 8MB LLC
16GB DDR4 RAM
Qlogic QLE2564 4 port 8GbFC HBA
Test SMB3 File Server

Test FC SCSI Target Proto_1
Windows Server 2016 (Nano)
i7-5820, 6c 3.3Ghz, 15M LLC,
16GB DDR4 RAM
QLE2562 2 port 8gFC HBAs
Storage devices: SATA SSD,
NVMe M.2 Flash,
MemoryChannelStorage
SMB3, NFS4 NAS

Nano Server FC Software Target Prototype1 Initial Measurements I

- ❑ Startup: Nano OS: 3 seconds; Application: 5 seconds
- ❑ Shutdown: 3 seconds
- ❑ FC Software Target App size: 1.7MB+1GB IO buffers depending on number of LUNs
- ❑ OS size on disk: 16GB (full OEM driver package, all storage features, IIS web sever)
- ❑ Driver size: 500KB

FC Software Target Prototype1 Initial Measurements II

- ❑ FC Software Target Proto_1 virtual device configuration
 - ❑ 4 FC ports
 - ❑ 2 LUNs per FC port (8 total LUNs)
- ❑ FC LUN media: MemoryChannelStorage Simulator1
- ❑ IOmeter measurements:
 - ❑ 512B Read 750KIOPS Qd=16
 - ❑ 4KB Read 700KIOPS Qd=16
 - ❑ 4KB Read 54us Qd=1 (Latency)
- ❑ System load
 - ❑ HW 6 cores (i7 5820) 92% CPU Utilization
 - ❑ OS System calls <10K/s
 - ❑ FC HBA Interrupts <1/s
 - ❑ Virtual Memory faults: 0

Nano Server FC Software Target Prototype1 - Additional Measurements

- ❑ **Updates and Additional performance measurements are in progress and those completed will be included in live presentation at SDC16 on Thursday**
- ❑ FC LUN media: MemoryChannelStorage Simulator2
- ❑ FC LUN media: NVMe M.2 Flash Samsung 950 Pro
- ❑ FC LUN media: SATA SSD Flash Samsung 850 Pro
- ❑ FC LUN media: SMB3 File Server
- ❑ FC LUN media: NVMe AIC on Prototype2 (Dell R730 mainstream server hardware)

- ❑ Updated presentation deck will be available from SNIA SDC website approximately 2 weeks after SDC16 Conference

Conclusions

- ❑ Windows Nano Server 2016 shows great technical potential as a solid Enterprise Storage Server platform OS for classic and future novel storage applications
- ❑ Nano storage application development degree of difficulty is less than RTOS, or Linux/Windows kernel mode applications, but more difficult than normal user mode applications
- ❑ New storage applications development should be easier than porting existing Windows or Linux applications since the Nano compatible API set will be used from the outset

Conclusion (continued)

- ❑ Testing underway to evaluate performance and hardware sizing characteristics of the following popular classic configurations:
 - ❑ **FC-AFA (All Flash Array)**
 - ❑ **FC-HFA (Hybrid Flash Array)**
 - ❑ **Unified Hybrid Storage Server with FC Software Target , SMB3.1 and NFS4.1 High Performance File Servers and iSCSI Target Server**, utilizing Flash and High Capacity HDD auto tiers (Storage Spaces volumes)
- ❑ Full Enterprise Storage features are also available internally for storage platform Hyper-V VMs (**Converged platform**) given appropriate sizing of HW resources

References

- ❑ Windows Server 2016 install media download
 - ❑ <https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-technical-preview>
- ❑ Nano Server Native Application Development
 - ❑ <https://blogs.technet.microsoft.com/nanoserver/2016/04/27/developing-native-apps-on-nano-server/>
 - ❑ Article above also contains links for following references in presentation
 - ❑ Getting Started with Nano Server
 - ❑ Visual Studio 2015 Nano Application Extension
 - ❑ Additional links for Nano related dev resources
- ❑ Nano application API scanner
 - ❑ <https://blogs.technet.microsoft.com/nanoserver/2016/04/27/nanoserverapiscan-exe-updated-for-tp5/>

THANK YOU FOR ATTENDING SDC16 AND THIS PRESENTATION

The **IOLogix Fibre Channel Software Target for Windows Server 2016** is currently in prototype and available only to OEM and System Integrator Beta Partners

Contact Terry.Spear@IOLogix.com for more information