31st Space Symposium, Technical Track, Colorado Springs, Colorado, United States of America
Presented on April 13-14, 2015

Virtualization Approaches and Lessons Learned

An Analysis of Mission Operations Centers

Primary Author

Justin Gronert, General Dynamics Mission Systems, Justin.Gronert@gd-ms.com

Co-Authors

Travis Chezick, General Dynamics Mission Systems, Travis.Chezick@gd-ms.com

Laura Burns, General Dynamics Mission Systems, Laura.Burns@gd-ms.com

ABSTRACT

This paper will provide a detailed description of two Mission Operations Centers utilizing virtualization technology at NASA Goddard: Magnetospheric Multiscale (MMS) and Landsat-8 (formerly Landsat Data Continuity of Mission, LDCM), plus the use of virtualization in the Joint Polar Satellite System (JPSS) Flight Vehicle Test Suite (FVTS). A compare & contrast will be made between these systems to illustrate design choices and the consequences to implementation. Lessons learned from the Integration and Test of these systems will be shared, followed by a brief survey of virtualization in associated operational environments.

EXECUTIVE SUMMARY

The use of virtualization in space mission operations has reached a tipping point, moving from technology demonstration projects into high-profile missions. Much like the past transitions from mainframes to microcomputers, today's designers must approach these decisions equipped with the necessary background information and an open mind to the technology's merits and constraints. The authors of this paper seek to inform the space/satellite operations community about both the former and the latter, concentrating on three themes to do so.

1. Virtualization is a powerful tool in an engineer's or a Program Manager's toolkit because it can offer substantial gains in expandability, flexibility and an decreased physical presence
2. Virtualization comes with some unique challenges as well, such as, new technical and implementation details, atypical software licensing, and a lack of familiarity within the operations team at large. Experienced personnel and proper engineering can mitigate these challenges.
3. Prototype a virtualized environment early and prior to any hardware procurement. This avoids costs from procuring additional hardware not needed once a virtualized environment is adopted.

INTRODUCTION

Mission Operations Centers within NASA have evolved with each new mission, both in their use of technology and the overall architectures to successfully support operations. A trend in recent years has been the growing use of virtualization within mission operations centers. As with many disruptive technologies, this approach was subject to skepticism in early proposals due to the lack of proven use in existing control centers. The majority of missions do not wish to take the risk of proving new technologies, instead they choose to rely on heritage designs and known processes. Fortunately, NASA's Goddard Space Flight Center (GSFC) supports a variety of missions including small, agile projects and technology demonstrators such as the Science and Planetary Operations Control Center (SPOCC). The SPOCC was an early adopter of virtualization, and that success opened the door for implementation on other missions with less risk tolerance.

This paper highlights three projects at NASA's GSFC and their implementations of virtualization. These projects are one Risk Class C and two Risk Class B missions, respectively: Magnetospheric Multiscale Mission (MMS), Landsat-8, and the Joint Polar Satellite System (JPSS) Flight Vehicle Test Suite (FVTS).[1]

General Dynamics Mission Systems

This technical paper is authored by General Dynamics Mission Systems (GDMS) personnel tasked to GSFC with background in mission operations and ground system architecture. GDMS supports GSFC with a multi-disciplinary staff providing engineering expertise on unmanned spaceflight projects from Phase B to Phase E of the NASA Systems Engineering Life Cycle. The areas of expertise include: Systems Engineering, Ground System Integration, Mission & Ground Operations Support and Information Technology (IT) Security. GDMS supports space and communications operations in a wide variety of ways including:

- Prime contractor and chief integrator of the Space Network Ground Segment Sustainment (SGSS) project, modernizing the operations of, and customer access to, the TDRSS constellation.
- Developed and produces the Command Post of the Future (CPoF) for the U.S. Marine Corps, merging customized application software with hardware and secure networking.
- Delivered the ground segment antennas for the Geostationary Operational Environmental Satellite R series (GOES-R), soon to be one of the world's premier weather satellite systems.
- Developed, built, integrated, deployed, and maintains the Rescue-21 system for the U.S. Coast Guard. Rescue-21 is a nationwide command, control and communications system which connects Coast Guard personnel with distressed mariners up to 20 nautical miles or more out to sea.
- Prime integrator of the Mobile User Objective System (MUOS) ground segment, providing communications and control interfaces to next-generation narrowband military communications.

GSFC MISSION OPERATION DESIGNS

MMS MOC Pre-Virtualization Design

The MMS Project is a NASA Solar Terrestrial Mission designed to observe the earth's Magnetosphere, which was built and operates at GSFC. The project incorporates four identical spacecraft that will fly in close formation over the course of its three-year primary mission within cislunar space. The MMS Mission Operations Center (MOC) handles the operation of the four spacecraft post-launch, including commissioning and normal operations.

Legacy Design and Implementation

The MMS MOC completed its initial Level 3 & 4 requirements for the Ground System at the end of Phase C ground system development.[2] This initial design segregated the operational (OPS) and test strings into isolated networks with servers accessible by users exclusively by NoMachine Enterprise Workstation or Windows Remote Desktop. Thus, if the host lived in an infrastructure rack, the system would be exclusively accessed via client/server software to establish a virtual session with no direct console used in a nominal situation. All workstations, defined as systems a user can sit in front of, would be accessible via console and could remotely access the servers if located in an appropriate security zone.

At the completion of this initial MOC design (reference Diagram 1), the implementation proceeded in three distinct phases:

- Phase 1: Network infrastructure & Test String

[1] Reference NASA NPR 8705.4 Risk Classification for NASA Payloads for further information on classifications.
[2] Reference NASA Systems Engineering Handbook for further information on mission phases.

- Phase 2: Half of OPS MOC hosts
- Phase 3: Second half of OPS systems and Non-critical hosts

The focus of the Phase 1 deployment was to deploy one instance of all required software products and appliances per the design. During this phase, system integration and test proceeded forward, checking out interfaces and software capabilities. The majority of software applications in the test string were originally hosted on basic workstations with some rack mount servers.

The Phase 2 effort began approximately two years before launch and provided the primary equipment for on-orbit operations. This included support for the command and telemetry system, trending and analysis, mission planning, data archive & distribution, as well as additional MOC infrastructure services. The Phase 2 hardware specifications were identical to the Phase 1 equipment (rack mount servers and workstations) but in larger quantities.

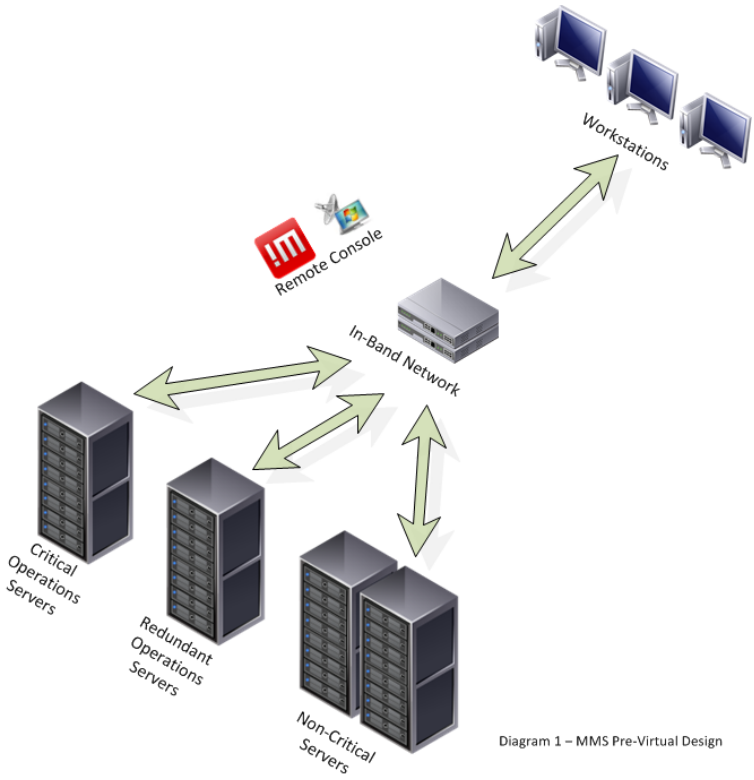

Diagram 1 – MMS Pre-Virtual Design

The Phase 3 development of the MOC was intended to provide all remaining redundant operations equipment and non-critical support services. It was during the planning for Phase 3 that the team identified the strengths of fielding a virtualized solution for a large subset of the remaining MOC hardware.

During this same time frame, the MMS Integration and Test (I&T) group, responsible for spacecraft assembly, had taken a very different approach than the MOC design above given their knowledge of the SPOCC virtualization design. The I&T team started off with the ground system built virtually and accessed through Linux Terminal Server Project (LTSP) thin clients, with tiered storage and high speed networks. This led to and allowed for rapid expandability and mobility of systems as the I&T scheduled moved forward when the spacecraft and GSE systems shipped to the various testing locations and launch site. Physical systems were used only as required to interface with the spacecraft to support specialized hardware interconnections. The I&T group also used a subset of MOC software deliverables, which helped during the MOC Phase 3 build out to determine the virtual design that could be integrated into the MOC.

Legacy Benefits and Constraints

Within the MMS MOC, all of the Phase 1 and Phase 2 hardware was procured and built without virtualization in mind. Every software platform used within the MOC was designed to have a primary and redundant physical system to meet the redundancy requirements; each software subsystem could sustain at least a single failure and still perform nominally during on-orbit operations. For the first years after the initial design was implemented, no additional hosts needed to be brought online. However, as hosts needed to be added for a new purpose and or as

additional requirements were levied onto an existing host, it was likely that additional hardware would be needed. This legacy design did not have the flexibility needed to support any new requirements without a significant impact. That impact being higher vendor hardware costs, additional staff time to coordinate the purchase, installation, and integration of the hosts, and the additional overhead of managing a diverse hardware inventory. This one to one hardware expansion was not ideal as the mission pressed forward to launch and the size of the ground system and its personnel expanded.

MMS MOC Post-Virtualization Design

About two years prior to launch, when the MMS MOC was mostly built from the Phase 2 hardware procurement efforts, the ground system engineers started to receive additional lessons learned from MMS I&T. Looking at the high expandability and the ability of many of the MMS software deliverables already proven to work within MMS I&T, the ground system engineers saw a great opportunity for the MOC to take advantage of virtualization. They analyzed the implemented MOC infrastructure as well as the new requirements developed from the I&T lessons learned to develop the MOC virtualized environment.

Design and Implementation

The virtualization architecture was initially scoped to incorporate all hosts which were incomplete at that time. These were primarily redundant subsystem and infrastructure hosts, such as the OpenLDAP, backup services, network monitoring servers, etc. The justification for undertaking virtualization came in two parts. The first was that the move would cut the overall hardware costs by purchasing two high performance systems in place of numerous mid-range systems. The second justification was the expandability offered by the virtual cluster to meet any additional pre-launch requirements without requiring more physical hardware. These conclusions were reached after conversations with the MMS I&T group regarding their successes with virtualized systems (reference Diagram 2).

Despite the experience of the I&T group, the MOC team had to overcome two points of resistance prior to proceeding with virtualization. The first of these points was that virtualization had been considered during preliminary ground system design but rejected in favor of traditional physical server hosting. The primary rationale at that time was that the technology was unproven, but the MMS I&T and SPOCC work had obviated that argument. Recognizing what was being done around GSFC in the different MOCs, as well as within MMS I&T, the MOC managers approved moving forward with virtualization with two exceptions. The first exception was that all MOC mission critical systems would not be virtual as they were already built and being verified through testing in accordance to the existing ground system requirements. The second exception was that all secondary infrastructure hosts that were deemed mission critical would remain on physical hardware to allow for redundancy in case of a possible overall virtual cluster failure.

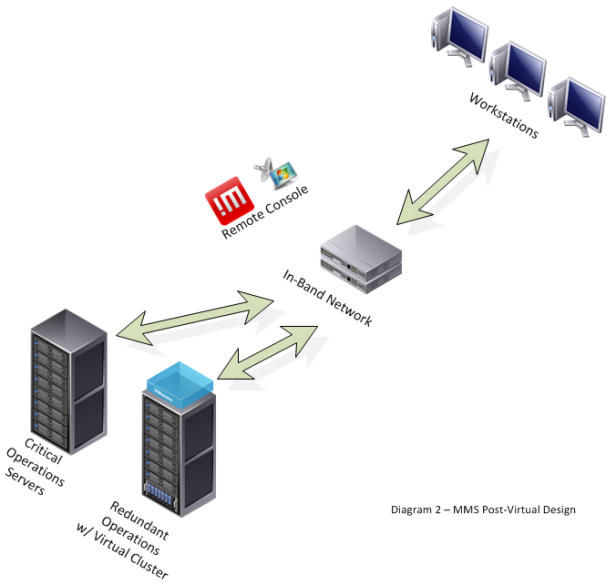

Diagram 2 – MMS Post-Virtual Design

The second point of resistance was the network provider's concern that the virtual design could cause additional IT Security vulnerabilities within the MOC because a virtual cluster that operated within the restricted mission network would have virtual machines (VMs) operating on a single physical "host" with access to multiple subnets. It also became apparent that the MMS MOC was

one of the first customers for the mission network provider to attempt to have a cluster that spanned multiple operational networks. The way all IT Security concerns were handled was to have dedicated connections from each physical host to each subnet, as well as be audited each time a system came online to ensure in no way the VM was bridging networks of different security levels. This design led to an excessive amount of connections from each hypervisor to each subnet network switch and took away from any additional non-virtual connections needed later, but was a reasonable compromise.

In the end, after an extensive review process, the team developed a multi-server VMware ESXi cluster operating on Dell servers with a Dell PowerVault Six Gigabit Direct Attached SAS storage. Ideally, most of the network bandwidth required to support the infrastructure could be handled with a single pair of trunked one gigabit network connections, however, mission network security requirements required dedicated network connections for every subnet to allow for isolation of VM traffic. So, each server had four quad port one-gigabit network adapters allocating the additional switch ports on the mission provided switches to allow for redundant paths. Even with these constraints, the final virtual implementation cost had to be equal to or less than the total planned budget for the ground system Phase 3 hardware and software procurement.

The MMS MOC also hosted two sets of simulators. The first, the Mission Training Simulators (MTS), utilized specific big-endian processors in accordance with the spacecraft design to emulate the hardware for four concurrent spacecraft. The FlatSat, located in the MOC, was built with flight engineering models, custom hardware, and commodity computers. These two simulators are expected to retain physical due to their custom hardware not allowing for virtualization.

Benefits and Constraints

The virtual implementation used for MMS was an example of integrating a virtual solution into a existing infrastructure upon recognizing evolving requirements due to mission complexity. The mission was originally designed such that all critical and non-critical systems with multiple nodes would have redundant backups should a hardware failure occur. However, the four spacecraft constellation mission required many physical systems with ever increasing complexity in the ground system with each additional host. The way to solve this was to significantly virtualize those additional backup, non-critical and infrastructure systems. The decrease in hardware maintenance support, new hardware procurement, and mission network support costs led to an overall reduced cost to the NASA customer than originally planned.

The design also enabled significant expandability, supporting numerous changes in requirements happening up until launch. It allowed the project to seamlessly create new hosts and develop procedures to meet the expanding requirements for nominal operations. The team noticed that as the Phase 1 hardware aged on the project and failed it was very easy to virtualize the failed hosts instead of fixing the aging hardware. By intentionally virtualizing physical systems (Physical to Virtual (P2V)), the team obtained contingency hardware for the remaining aging system as well as offering better overall performance for most P2V hosts. Reusing this existing hardware further reduced the costs of maintaining and procuring more spare systems.

Once the cluster was in place, all the new VMs were easily handled within the original concept of connectivity using an virtual console, NoMachine or RDP, depending on the operating system. This greatly helped the end users by keeping a consistent use case and was a selling point to project management.

As additional VMs were added, the MOC had to increase the cluster storage capacity using Dell PowerVaults, and increase usable memory within the cluster's hypervisors (physical host systems). The cost growth incurred by expanding these hardware solutions was still substantially lower than would have been required with adding traditional hardware.

Alternatively another network implementation that could have been implemented through VMware ESXi was operating with VLAN trunking. Using the native ability within ESXi to trunk connections from multiple vSwitches through a single network interface to the network switch would offer decreased physical network connectivity. This could be further taken advantage of by using Inter-VLAN routing on the network switches to allow for communication

between local subnets without crossing the firewalls, which are often a bottleneck in network throughput. By using Inter-VLAN routing together with VLAN trunking built into VMware it could offer even greater optimization of network connectivity for VM's, but would not need to be implemented should the mission have specific requirements. However given the implemented design above and the other feature choices available to the mission, it was ideal to engineer the designed solution which fit the given requirements.

Landsat-8 MOC Pre-Virtualization Design

The Landsat-8 project was developed as a collaboration between NASA and the United States Geologic Survey (USGS) to continue the 40+ year legacy of Landsat imaging. The mission was developed as a Risk Class B project with a five-year primary mission life. Like MMS above, the subset of the project that handles the spacecraft's launch, commissioning and normal operations is the Mission Operations Center.

Legacy Design and Implementation

The Landsat-8 MOC was developed in three primary elements: the Mission Operations Element (MOE), the Collection Activity Planning Element (CAPE) and the MOC Ground System Infrastructure (GSI). The first of these components, the MOE, provided most of the MOC's core application software packages including Command and Telemetry, Mission Planning, Trending and Analysis, Flight Dynamics, Data Management, and Notifications. The science sequence planning was accomplished by the CAPE. The final component was the MOC GSI, which included network, authentication, timing, configuration management, database translation, and other support capabilities.

The legacy MOE computing resources were rack-mount workstations deployed in a dedicated server room. The CAPE utilized mid-range 4U machines for server-side data processing. The MOE and CAPE workstations shared a single hardware specification for all components in an effort to provide redundancy through numbers (reference Diagram 3).

For both the MOE and CAPE, the users connect to the workstations via the Teradici implementation of PCoIP, a protocol which combines hardware video compression with USB pass-through to provide a remote desktop experience through a zero client. Connections are established and monitored by a broker, providing authentication and authorization controls via Active Directory and RSA. The implementation allows the use of service accounts on the workstations where applications may continue running as shift changes occur. This also enables lights-out automation without the need to convert all legacy applications to daemons or "headless" applications if they were executed by non-service accounts.

The infrastructure services were hosted through a combination of 1U rack-mounted servers and standalone workstations, plus network switches, routers, and firewalls.

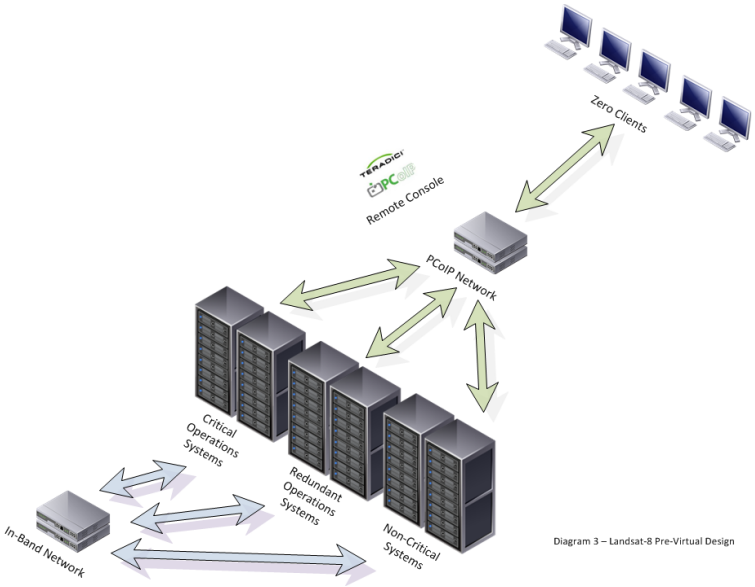

Diagram 3 – Landsat-8 Pre-Virtual Design

Legacy Benefits and Constraints

The Landsat-8 design philosophy was predicated on a concept that hardware failure in a critical function would be followed by promotion of the redundant hardware to operational status, and that, if required, the failed machine could be replaced by an unused machine from elsewhere in the system. By launch, the MOC's complement of servers, workstations, networking, and storage used six 42U racks. The total system count included nearly 100 power supplies and 200 disk drives, all of which represent moving parts which are more prone to failure. These final totals are in part due to scope creep during integration and test, when the team identified additional resources required to meet the aggressive launch development schedule such as extra workstations to support additional surge personnel.

The Landsat-8 redundancy approach provided the necessary function and availability, but resulted in higher-than-anticipated maintenance costs and infrastructure support services. This situation was exacerbated by an aggressive pre-launch development cycle, troublesome server build processes, and intermittent failures of the backup systems.

Landsat-8 MOC Post-Virtualization Design

Following launch and transition to operations, the Landsat-8 team began planning a hardware refresh cycle. The majority of MOC hardware had been purchased at the onset of ground system development and was approaching or exceeding end of life support with the Original Equipment Manufacturers (OEMs). This deployment was different than previous efforts in that it occurred entirely during Phase E (Operations), when the system design is less prone to scope creep and other design uncertainties.

The initial refresh plans followed the pre-launch paradigm, with configured hardware delivered by the element leads for integration to a refresh MOC system. The MOE vendor initially proposed a new generation of hardware built on an VMware ESXi platform and building on the lessons learned while virtualizing the Lunar Reconnaissance Orbiter (LRO) MOC and the Science and Planetary Operations Control Center. Upon analyzing the current hardware aging and reflecting on pre-launch integration challenges, the MOE vendor approached the government with a plan to collaborate with the existing Infrastructure staff in rebuilding the entire MOC in one concerted effort.

This Landsat-8 MOC Refresh effort began by analyzing the system hardware needs and performing a trade study to compare options including a direct one-for-one hardware replacement, a fully virtualized system, and a hybrid approach. The primary goals for the refresh were for the MOC to have equal function and performance, the Mission Operations software and configuration remain unchanged, and that the system resolve a handful of important system issues. The secondary goals of the refresh were to reduce maintenance manpower requirements and to improve the server build and update processes.

Design and Implementation

While the focus of the refresh was implementation of new computers, the project was envisioned as a full hardware replacement with the exceptions of a unique hardware simulator and the Caribou MYK-15a encryption devices. As such, the study team selected new network switches, firewalls, and storage devices to support a fully virtual environment.

With regards to the computing capabilities, the study team examined the software used within the MOC and determined that the majority would easily be virtualized. Four software platforms were potential outliers in this regard:

- Flight Dynamics System (FDS) – Factory tests revealed performance issues when using customized attitude and orbit visualization tools in a purely virtual environment. The study team identified a graphics card which was designed to provide physical Graphical Processor Unit (GPU) resources to virtual machines.

- Trending and Analysis – The data trending system utilized CPU-intensive data processing to generate statistical data reports. The study team used SPEC ratings from the Standard Performance Evaluation Corporation for comparable hardware and made the decision to accommodate the trending system in the ESXi cluster and size accordingly.
- Software-only spacecraft simulators – No specific information was available regarding the performance of the simulators in a virtualized environment. Prior to launch, the team had partial success in fielding a software simulator on updated hardware. It was suspected that the upgraded hardware was causing the system timing to be intermittently out of sync. The virtualization of these software simulators was agreed to be a stretch goal, with a fall-back plan to maintain the legacy systems.
- Active Directory – Vendors indicated that virtualizing the Active Directory could have functional and/or stability issues. As the only remaining component, the study team elected to proceed with virtualizing this component and managing the associated risk.

For all four outliers, the vendor worked with the government and agreed to proceed despite the potential risks in implementation. The team also identified the hardware, contractual, cost, and schedule changes which would be required to revert these outliers to physical systems, if required (reference Diagram 4).

Based on historical processor and memory usage, the team elected to build a cluster of four servers, allowing full operational capabilities with any one host offline. The backup MOC (bMOC), a single-string variant of the primary MOC, had two hosts. While the bMOC was not specifically designed to be fault tolerant, the system could operate on a single server with coordination of activities.

In part to complete the design within a hardware budget ceiling, the study team selected a two-tier approach to data storage. The virtual machine storage resides on a RAID enclosure populated with 900GB 10krpm SAS drive, while bulk data storage and virtual machine backups reside on separate enclosures utilizing 4TB 7.2krpm disks. This hybrid solution proved to be an effective balance of cost, storage space, and performance.

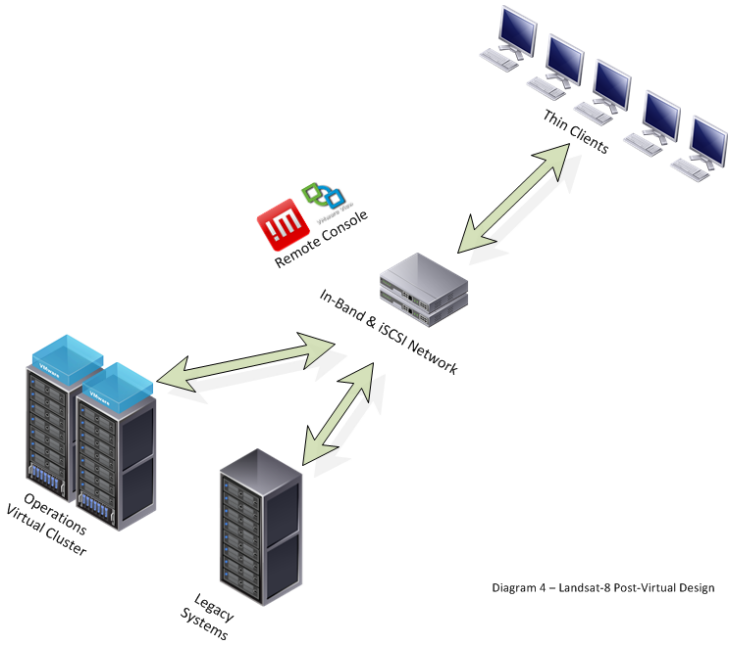

Diagram 4 – Landsat-8 Post-Virtual Design

The user interface to the system makes use of thin clients and a connection broker for user authentication and authorization. This approach mimics the behavior of the legacy system and eases transition training, while migrating to more commonly supported standards. The thin clients are configured to boot from a central LTSP server.

As the refresh design called for fully virtualizing the MOC, the team selected a prepackaged software appliance to perform file system backups. This application takes advantage of the underlying virtual machine infrastructure to capture snapshots and seamlessly manage system backups. The implementation successfully backs up all virtual machines with no impact to the running instances.

The team used Cobbler, Puppet, and LTSP to streamline system deployments and to provide configuration management control of the virtual machines and thin client devices. Other specific solutions developed by the team throughout integration included:

- Migrated from RAID-6 configuration to Dynamic Disk Pooling. Resulted in RAID throughput increase of 30%.
- Migrated from RDP 8.1 to VMware Horizon View. Resulted in video performance improvement of nearly an order of magnitude (as measure in frames per second)

- Altered host Power Related BIOS settings. Reduced process time from 42 minutes to 17 minutes for a CPU-intensive task.

The integration and test team was split between those who had prior experience with hands-on virtualization and those who had received formal training from the vendors. The individuals with formal training had more insights into design options and virtualization capabilities, while those with prior experience were more able to apply their knowledge to advance the system integration. A significant lesson learned was that the formal vendor training would be most valuable if it occurred prior to completion of the system design, rather than just prior to system integration.

Benefits and Constraints

The use of the dedicated video cards and serial interface cards to specific virtual machines impeded the ability to migrate those virtual machines to another ESXi host in the event of a failure or planned maintenance. In addition, the backup process is based on a cloning mechanism which was also impeded by the presence of the Direct Device Mapping. The team considered migrating these to physical computing resources, but determined that this would create drawbacks that were equivalent or worse. As such, the systems remained virtual, but with additional steps required for VM migration and routine backups.

Virtualization of the computer hardware eventually decreased the cycle time for implementing new functions and creating new machines, but was not a magic bullet for the work of configuring and integrating the functions which comprise an ops center.

The importance of careful software licensing in a virtualized environment cannot be overstated. In several cases, the vendors who supplied software licenses were unable to correctly ascertain the licensing requirements for the virtualized system. In other cases, the application of existing knowledge was found to be grossly incorrect in the virtualized environment. Additionally, the virtualization power to replicate VMs can put the organization at risk of inadvertently violating the existing licensing.

The management of security scanning and patching was also improved through the use of virtualization. In particular, the security team was able to replicate existing virtual machines for scanning without impact to ongoing operations. This feature improved the system availability and decreased the manpower required to maintain the refresh system.

JPSS FVTS Pre-Virtualization Design

The JPSS mission is a weather and climate satellite being developed as a partnership of NASA and the National Oceanic and Atmospheric Administration (NOAA). It contains five science instruments used to measure the Earth's weather and climate through its seven-year primary mission lifetime. The Flight Vehicle Test Suite (FVTS) is the set of simulation capabilities currently in development to support the JPSS spacecraft constellation.

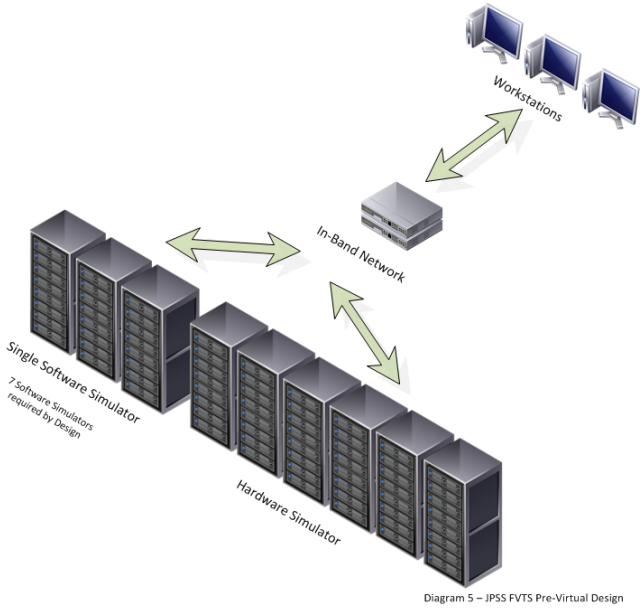

Diagram 5 – JPSS FVTS Pre-Virtual Design

Initial Design

In 2012, the NASA team developed the initial requirements for JPSS FVTS based on experience building ground support equipment for the James Webb Space Telescope and the lessons learned from

JPSS's predecessor, the Suomi National Polar-orbiting Partnership (S-NPP). A major lesson learned from S-NPP was that its one simulator, the Flight Vehicle Simulator (FVS), was constantly oversubscribed. The S-NPP FVS was a hardware-based simulator and was fragile to maintain. The resulting design of the JPSS FVTS led to one hardware based simulator and a more easily replicated software-based simulator. The current plan is for seven instances of the software-based simulator (reference Diagram 5).

The simulation functionality includes simulations of the spacecraft and each of the five science instruments. Vendors of the flight components built their respective simulations. There is also a Ground Link Simulator built by the vendor of the JPSS Ground System and a Simulator Control component developed in-house at NASA's Goddard Space Flight Center.

The initial design of FVTS planned for separate machines for each component with the hardware and software specifications being defined by the vendors. Minimal constraints were defined in the requirements with regards to operating systems or software development. This resulted in eight very different systems. Most vendors used Red Hat Enterprise Linux, but two of them used VxWorks and one used Windows. The hardware specifications defined a broad spectrum of machines with many hardware brands and varying maintenance contracts. This initial design of the FVTS system led to a very complex hardware environment being integrated with the existing complexity of the simulation software.

JPSS FVTS Post-Virtualization Design

As the infrastructure design team got a better picture of the various vendor provided components, it became clear that having individual workstations for each component was inefficient and difficult to maintain. Managing systems upgrades and tracking relevant drivers made maintenance even more challenging. Beyond being difficult to maintain, the resulting system lacked redundancy. If one piece of hardware went down, then the whole simulator system was down. After understanding these issues, the decision was made to standardize the hardware into a single virtual platform to help ease overall integration.

Design and Implementation

Moving to the virtual environment, the FVTS team developed a virtualization system that applied to both the hardware and software based simulators (reference Diagram 6). Both systems are significantly easier to maintain and are fully redundant within the virtual systems. Drivers became a non-issue since they are now standardized across all of the VMs. In the hardware based simulator, significant physical equipment remains. A hardware interface layer, e.g. 1553 or SpaceWire, exists between all of the flight simulators, but the application layer for several systems can run in the virtualized environment. For the software based FVTS simulators, four of the eight components could be completely virtualized. Three additional components could be partially virtualized.

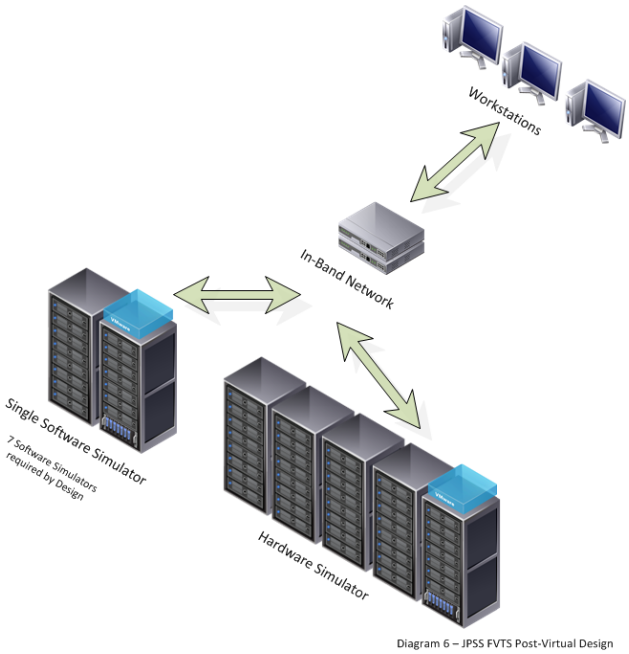

Diagram 6 – JPSS FVTS Post-Virtual Design

The FVTS was built on three HP Blade Servers. Each blade server has two CPU sockets, each of them populated with an 8 CPU chip resulting in 16 CPU cores per blade server. Each blade also has 128 GB of RAM. For data storage, each FVTS instance has 3 StoreVirtual Arrays containing 12 three Terabyte drives resulting in a 36

Terabyte array. The array is expandable, but only with drives that are exactly the same as the existing drives. It can't be extended with drives with different capacity or brands. The Left Hand OS software by HP configures and manages the storage array, including the redundancy settings. As a result of the redundancy settings, 24 Terabytes are available for subscription.

All three blade servers are up all the time. Resources are subscribed to by each VM, and resources can be oversubscribed. A given virtual machine doesn't own a given resource such as RAM, it subscribes to the RAM. All of the virtual machines running on a given blade server may subscribe to more than 100 percent of a resource. Since the three blade servers share resources, a VM can be seamlessly transferred without impact to the user from one blade to another when resources become limited. The blades provide automatic failure recovery. If an entire blade were to fail, the VMs running on that machine are seamlessly re-loaded on one of the other blade servers without human involvement. To the user, the hardware failure would only appear as if the system had restarted.

The FVTS blade servers connect outside components over a 10 Gigabit Ethernet switch. This connects the blade servers to the storage array and enables the lower rate switch. The system uses two teamed ports which allows for bandwidth sharing as well as redundancy. The ten Gigabit switch is connected to a one Gigabit switch which connects to administration network and the non-virtual components. Communications between virtual machines on the blade servers communicate over a 34 Gigabit virtual switch.

The virtual machines are accessed by networked workstations with a dual headed displays. FVTS uses the vSphere virtual console manager as an access point for all of the virtual machines. The system administrators granted permissions to users based on user permissions and roles. When a user logs in they see a selectable list of virtual machines based on their approved user roles. Authentication is granted via active directory. The multiple FVTS system racks share trusts so that a given workstation can access virtual machines on multiple racks. Each FVTS instance must be able to run independently, so it can act as as an enterprise in a box managing all of the necessary security functions.

Benefits and Constraints

The FVTS concept of operations necessitated a system that is easily replicated across the multiple instances and sites. For security reasons, the FVTS systems are not connected to an external network, thus the initial plan for deployment of upgrades was not straightforward. By virtualizing much of the system, the deployment process was significantly simplified, saving time during upgrades. Using VMs, the FVTS team creates golden VM image at the factory and deploys them as a whole unit to the systems in the field. A challenge using this deployment process is that user data can be lost during upgrades. The FVTS team is investigating several potential solutions for this problem and hasn't settled on a solution at the time of this writing. The majority of the FVTS user data is concentrated in known specific locations making the potential solutions easier. The first option is to copy the data to the new VM during upgrade. The other option involves mapping the user areas to shared folders that will not be overwritten during upgrades.

This ease of replication also allowed the test teams to create testing environments that were not possible with the conventional hardware approaches. The Simulator Control Component was designed to control all of the simulation aspects within FVTS, therefore it has eight external interfaces. The initial design had the detailed design test team with one test environment that only enabled the operation of one or two test interfaces at a time using physical laptops or workstations. Using the virtual environment, the test team created three test environments with nine virtual machines in each. This provided a significant schedule benefit by allowing multiple test activities to occur at the same time. It also allowed for more robust testing of the software and interfaces by enabling tests where data is flowing across all interfaces simultaneously.

The FVTS team deployed virtual machines for certain applications on individual user laptops to allow users to draft scripts and familiarize themselves with the user interfaces without tying up the primary resources. This also allowed for simplified training where sessions can be run with each user able to have hands-on access to the software.

Alongside the ease of replication comes the ever present concern over software licensing. At a given point FVTS may have hundreds of VMs spread across the eight FVTS systems, the test environments, and individual user VMs. This can be a logistical nightmare to manage the software licenses and to ensure licensing agreements are not violated. Some of the software vendors made this pretty straightforward and easy to manage. Others made it very difficult and expensive. When designing a virtual systems it is important to plan in advance for the licensing agreements and the potential for future expansion. In one instance, FVTS had an issue where the vendor did not plan for all the required licences nor growth in the deployment of the systems. The acquisition of the additional licenses took months and threatened the overall system schedule. It is best to have a plan for how to acquire additional licenses rather than try to go back and add additional licenses during integration.

The FVTS was deployed into very full computing facilities with limited space for the simulators. By going to the virtualized approach, the rack footprint each simulator was reduced by one rack saving the cost and computing center footprint of eight racks across all of the simulators. The FVTS virtual design allowed for increased flexibility, especially during a complex integration and deployment process. It allowed the FVTS team to expand the memory, CPU resources, and available storage space of a given VM as needed to address unexpected performance issues with a given component simulator.

FVTS was not designed from the outset for virtualization, therefore the design didn't take into account all of the positive aspects a fully virtualized environment could provide. If the system had been designed with virtualization in mind from the outset, additional features could have been leveraged to make the system even more robust. The FVTS instances at a given physical location could be consolidated further saving the cost of equipment and computing center floor space. The linked systems could have provided backup and resource sharing functionality across all of the systems. If a secure network was created across all of the FVTS deployment sites, then all of the FVTS instances could be linked together and provide redundancy and back-ups for each other.

The future of simulators is moving towards fully virtualized environments. The current FVTS design has some simulators running on VxWorks or custom hardware to enable the emulation of the flight hardware. Software options, such as WindRiver Simics, allow instrument and spacecraft vendors to run unaltered flight software on an emulation of the flight hardware environment rather than custom software. Two of the science instruments within FVTS use this technique already, future simulator specifications will require all of the systems to run as software in a virtual environment.

ADDITIONAL LESSONS LEARNED AND CONSIDERATIONS FOR FUTURE PROJECTS

In addition to the detailed experiences, designs, benefits and constraints provided thus far, the authors of this paper gathered lessons learned that translate across these projects and could apply to future virtualization efforts on other projects.

The ground system implementation for any mission should begin by taking an in-depth look at the complete environment, with designs drawn from requirements, to cover every possible scenario. There will always be some requirements or events that will not be known upfront and will have to be analyzed when they arise. Addressing these new requirements will often be easier if a virtual design is adopted early on in mission development. The more time personnel had working with a virtual environment, the more comfortable they became especially when addressing unforeseen requirements or events. It was the lack of comfort in early virtual designs developed and tested by various missions at GSFC which led to its absence in newer missions.

It is also entirely possible that depending on the size of the infrastructure being developed a virtualized environment might not be required and could be more expensive. The true power of virtualization comes with the ability to have a large infrastructure with a minimal hardware footprint. But, if the ground system is not large enough, virtualization can be cost-prohibitive and counter-productive to build out a completely redundant virtual cluster. It is still possible to implement a standalone hypervisor to take some of the benefits (such as low costs) with a virtual design, but it would not be redundant or fault tolerant.

From the authors' experiences when developing a virtual environment, it is a sound approach to design a very expandable solution that can evolve as requirements change. We found it to be a good practice to build a system

with, at a minimum, double the resources needed to support the initial design. This will create a virtual cluster that does not need change in its overall physical resources after requirements creep. Typically, once a virtual infrastructure is put into place the architecture will soon follow by immediate expandability of new systems and designs. With this in mind, the system design should allow for later expandability of cluster resources and the project should have a budget reserve to allow for this eventuality. Building a virtual cluster that does not allow for any expandability with respects to storage, hosts, power and networking can ultimately cause failure in the overall design.

Finally, it is nearly impossible to have a completely virtual environment. Yet, having a highly-virtualized environment does offer several benefits. The ability to seamlessly deploy systems on the fly and handle physical host failure allows, in an operational environment, almost uninterpretable up time. Implementing a completely virtual solution with the software available at the time of this paper has its limitations, as noted on some of the projects mentioned above. There can be a lack of graphics acceleration, issues with specialized hardware mapping, and issues with the performance of the thin or zero clients offered to access the virtual machines. It is definitely possible to have a highly virtual environment with noticeable performance differences between it and a completely physical environment of similar design.

SUMMARY OF FINDINGS

This paper highlighted three projects at NASA GSFC and their implementations of virtualization. For each of the projects, virtualization offered substantial gains in expandability and lower power and hardware support costs. While engineers on each of the projects had to overcome obstacles such as a lack of familiarity for the operations teams and additional licensing costs, the benefits outweighed these challenges. Experiences from each project also illustrated that if virtualization had been considered earlier in the design process, some hardware procurement costs due to a fully physical system could have been avoided (reference the summary table).

Mission		MMS	Landsat-8	JPSS FVTS
Legacy	Benefits	Operational Reliability Single Hardware Vendor	Operational Reliability Excellent Desktop User Experience to Systems	Operational Reliability Built to Mission Requirements
	Constraints	1 to 1 Expandability High Power, Network and Hardware Support Costs	1 to 1 Expandability High Power, Network and Hardware Support Costs PCoIP Connections required specific Hardware and Dense Network Presence	1 to 1 Expandability High Power, Network and Hardware Support Costs Varying Vendor Support Costs and Hardware compatibility Difficult to maintain
Post-Virtualization	Benefits	Operational Reliability Single Hardware Vendor Lower Network, Power and Hardware Support Costs Increased Expandability	Operational Reliability Single Hardware Vendor Lower Power and Hardware Support Costs Decreased Physical & Network Density Increased Expandability Increased System & Network Performance	Operational Reliability Consolidated Physical Foot Print Lower Power and Hardware Support Costs Increased Expandability Simplified Replication
	Constraints	Kept existing Host Connectivity Scheme Increased Licensing Costs	Increased License Support Costs Large Engineering Effort to Build a Fully Virtual Solution Different Connectivity Protocols to Systems	Increased License Support Costs
Key Take-Away		Virtual Design offered increased Expandability at a Comparable Cost with decreased Hardware, Power and Network Size to a Physical Solution.	Virtual Design offered increased Expandability at a Comparable Cost with decreased Hardware, Power and Network Size to a Physical Solution. Fully Virtual Solution required substantial Engineering to resolve issues with the Mission Requirements.	Virtual solutions offered greatly increased replication and expandability to resolve Mission Requirements. The Virtual solution also required significant cost and research towards Software Licenses.

Virtual solutions offer several key improvements over comparable physical designs, which if implemented should be developed early on in the development cycle. It is best to develop a solution based of the requirements given for the project and use sound engineering principals to decide what benefits, if any, virtualization can offer. The use of physical systems should not

be discounted if proper engineering dictates it within the design. Lastly, even within the complicated environment of the space operations, valuable lessons on implementing computer resources to support critical operations can be learned from other industries demanding highly secure, highly connected, highly available systems such as banking, telecom, medical, homeland security, and military.

HONORABLE MENTION

The authors of this paper would like to thank the generous support of the following individuals, without whom this paper would not have been possible: Noah Williams, David Gomme, Dan Whorton, and Georgie Brophy.