

IADS IN THE CLOUD

Michael Jones
Curtiss-Wright Defense Solutions, IADS
 190 Sierra Court, Suite A3, Palmdale, CA, USA 93550
 mijones@curtisswright.com

Abstract

This paper will evaluate the use of IADS telemetry processing software in a cloud environment. The evaluation process will utilize the products available via Microsoft Azure services at several data centers across the continental United States. The information presented will include areas of success and areas of concern. The review will reveal the potential benefits of cloud computing for distributed test operations (DTO).

Key words: IADS, cloud computing, distributed flight test

Introduction

For many years the flight test industry has seen an ever-increasing need for interfacility cooperation and use of standards-based resources. The IADS software suite for telemetry processing, display and analysis is often treated as one such common tool. Its use as a common tool is often seen across government flight test facilities and commercial flight test operations. This can become particularly important when a commercial manufacturer participates in test activities with their respective government customers.

A primary resource for modern flight test activities are flight test engineers and subject matter experts who are regularly required to travel around the world to participate in real time analysis. The topics covered here could provide some insight and opportunity to extend the mission control room environment beyond the local facility. This extension has the potential to reduce the need for some of these resources to be physically present at the local facility where the test is being performed.

This paper is intended for review by flight test engineers, flight test data management personnel and control room personnel to gain additional understanding about what capabilities and limitations might be possible combining IADS with cloud resources. It is not intended as an exhaustive review of all the possibilities, but instead as a primer to encourage further exploration by the flight test community.

Real Time Architecture Overview

The use of IADS in a real time flight test environment will always require a minimum of three functional components. The data source, the data server and some number of display and analysis clients.

The data source is one of the eighteen compatible sources-types available for use with IADS. Two of the defined sources are intended for use by third parties to feed data to the data server using published application interfaces. The data source is likely to be one or more systems or devices upstream of the data server. This functional subsystem will be called 'data source' or 'DS'

The data server receives data from the upstream data source. The data server is typically composed of a single system designed to accommodate local processing needs. The system becomes the host for the IADS Server software. This physical system together with IADS software is commonly known as 'the server' or 'the CDS'; these terms are used through the remainder of this document. The server performs a variety of operations related to real time operations, among them are the ingest of serialized data from the data source, time alignment of the serial data stream, storage of the data and delivery of the data to downstream display and analysis clients.

Display and analysis clients are some number of independent systems downstream of the server. These systems combined with software retrieve data from the upstream server. This software could be the IADS Real Time Display and Analysis Client, but it could also be third-party

software utilizing published application programming interfaces (APIs) to retrieve data from the data server. Collectively, these system and software processes together are commonly known as 'client' or 'clients'; these terms will be used through the remainder of the document.

A functional diagram of the components is shown in Figure 1. Network connectivity between the components is using TCP/IP which is shown with arrows.

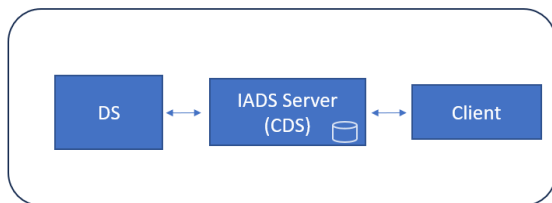


Figure 1 Functional Diagram Text

Test Environment

A standardized test environment was created using the appropriate physical hardware for each of the functional components in the real-time architecture. Two identical physical systems were allocated for use as a data source and as a server. An additional physical system was allocated for use as a client.

DS and CDS specifications:

Intel i9-9820X @ 3.3GHz, 10 Core
CPU, 64 GB RAM, RAID1 SSD Storage Array

Client specification:

Intel i7-12700H @ 2.7GHz, 14 Core
CPU, 32GB RAM, NVIDIA RTX 3070 TI
GPU

A virtualized test environment was created using Microsoft Azure Virtual Machines with the following specifications.

DS: Azure Standard E2s v3

Intel E5-2673v4 @ 2.3GHz, 2 Core
CPU, 16 GB RAM.

CDS: Azure Standard D8ads v5

AMD EPYC 7763 @ 2.44GHz, 64
Core (8 vCPU), 32 GB RAM. SSD Storage.

Client: Azure Standard D4s v5

Intel Xeon Platinum @ 2.1GHz, 4 Core
vCPU, 16GB RAM, Hyper-V Video.

Comparison of the virtual environment to the physical environment will reveal an obvious disparity in favor of the physical systems that would imply that they will always outperform the

comparative virtual systems. That would ordinarily be a correct observation, but it should be noted that the virtualized systems were tested and deemed capable of quality performance for all test scenarios as compared to the physical systems. Stated another way, the virtual systems have been appropriately sized for the purposes of these tests.

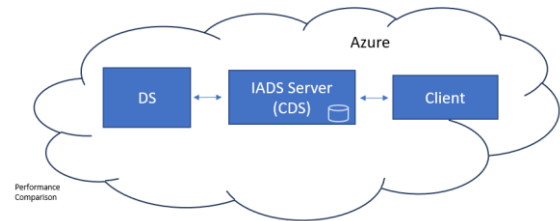


Figure 2 Cloud Functional Diagram

The physical test environment was located at the Curtiss-Wright, IADS facility in Palmdale, California. Virtual test environments were created in three Azure regions: West US, Central US and East US. West US is located in California, Central US in Iowa, East US in Virginia.

Network connectivity between the physical environment and the Azure Cloud was established using a persistent Azure VPN Gateway between the Palmdale office and Azure West. Network connectivity between Azure regions was established using Azure Peering between each region-to-region permutation.

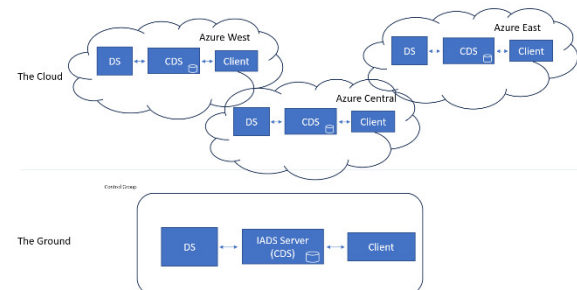


Figure 1 Representation of the entire functional environment

All IADS software used during testing was from the commercial software release of version 9.2.8 (September 2024). No optimization settings were utilized in any manner. All CDS configurations were default values as generated by an IADS Operators Console for command and control. All Client configurations were also using default values as generated by an IADS Operators Console for command and control.

All display configurations were constrained to a resolution of 1920 by 1080 ("full HD") with a 100% scaling factor.

Test Data

These values are known to be reasonably representative of commonly observed workloads. Increased workloads are known to perform comparably provided that the available network bandwidth can transmit the data from the source to the server without constraint. The performance specifications of the server system would also need to be appropriately upscaled.

A data source simulation was configured with the following characteristics: 4,300 four byte parameters with sample rate distributions between 1 and 2604 samples per second; resulting in an aggregate rate of over 400,000 samples per second.

The total parameter set available to clients was multiplied roughly 10 to 1 resulting in over 43,000 parameters. These additional parameters are composed of a wide range of derived equations known to be representative of common client workloads. The client display configurations tested included visualizations of data as observed in real world workloads as well as artificially configured displays to determine limitations. See the appendix for examples of displays.

Server Analysis

Server software was observed to run with equal performance characteristics in all cases where the data source and the server were contained within the same geographic region (i.e. ground to ground, Azure West to West, Azure Central to Central, Azure East to East).

For the above cloud-based scenarios this behavior was expected as each individual region is reasonably likely to meet the key performance metrics for the server. The server software has a design expectation for packet deliveries which at 20 mSec intervals in a streaming manner between the data source and the server.

Permutations were run to separate the geographic locations of the data source and the server to observe any potential behavioral limitations. For example, DS on the ground in Palmdale to a server in Azure West. DS in Azure West to a server in Azure Central, etc.

Analysis of geographically separated data sources and servers resulted in the following results. Shorter distances are generally capable of meeting the 20 mSec design goal for packet intervals. Longer distances will begin to see intervals which are incapable of meeting the design goal or intervals which begin to swing widely in their processing intervals.

Consistently acceptable processing was observed from Palmdale to West, West to Central and Central to East.

Analysis of an Azure West data source to an Azure East server revealed a consistent inability to meet the interface design goals. Average intervals tended to be in the 30 mSec range with regularly occurring intervals of 120 mSec or more. These intervals cause a pivotal degradation in performance between the server and its downstream clients. Observation at a client when excessive or erratic intervals are present will generally be described as a lack of smoothness or bursty updates.

The inability to achieve the 20 mSec intervals directly correlates to the network latency of the network path from data source to server. When the server is operating is highly latent from the data source, data will continue to be processed and archived without any negative effect. Unfortunately, the downstream clients will likely be negatively affected by observably erratic data flow.

Display Updates as a Measure of Display Performance

Client software was observed to run with equal performance characteristics in all cases where the server and the client were located within the same geographic region (i.e. ground to ground, Azure West to West, Azure Central to Central, Azure East to East).

In all scenarios where the server and the client are in different locations or regions it becomes possible to observe a degradation in client performance. See Figure 4 for a visual example which illustrates the relocation of a client from an all Azure-West layout to a layout where the client has been moved to the local physical system in Palmdale.

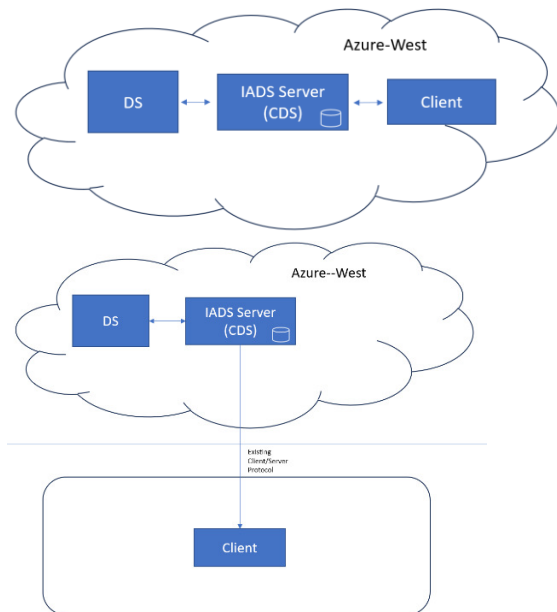


Figure 2 moving a client from local to remote

The most commonly observed change is related to the client's ability to generate display updates. This metric is available in any client by activating Performance Window. An example of this window is available in Figure 5.

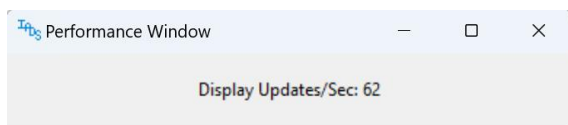


Figure 5 A crop of the IADS Client performance Window

This value is continuously computed and updated as each display frame is passed off to the video display of the system. The default maximum value which can be achieved is 64. The minimum value which can be displayed is 0.

With the concept of display updates per second in mind, an example of a performance change for a non-local client is as follows. A window of 30 stripcharts runs on a local client at 50 updates/s but the remote client is only able of rendering the same display at 18 updates/s. Another example: a window of 40 stripcharts runs locally at 39 updates/s but the remote system is only capable of 12 updates/s

I ask a question of the reader, "Is this change in behavior acceptable or unacceptable?" I believe this question may only be answered by the engineer or analyst responsible for decision-making based on the visualization being presented. Anecdotally, the human brain is quite comfortable watching a movie at 24 frames per second in a movie theater. At this rate the average viewer likely has enough information flowing from the eyes to the brain. Different analysts will have different needs.

An important note, it is possible for display updates to be so low in performance such that the appropriate metric becomes "Seconds per Update". If this condition occurs, the result will be presented in the Performance Window as "Display Updates/Sec: 0"

Extending the examples from above. Adding a single stripchart to the remote system dropped the update rate to 2.5 seconds per update. A seemingly simple increase from 40 to 41 stripcharts, and the display becomes objectively unusable. One single additional stripchart resulted in a drop from 12 updates/s to 2.5 seconds per update. I shall refer to this dramatic drop in behavior later as 'the tipping point'

Analysis of the server to client communication metrics has revealed that as network latency increases there is a correlation to the drop in display performance at the client. Artificially induced network latency on a co-located server/client relationship produces the same change in behavior.

It was additionally observed that as the latency increases the reduction in performance quickly devolves to objectively unusable in most cases. Running a server in Azure-East generally does not result in "acceptable" performance. In contrast, subjectively acceptable results were regularly achievable when the server to client relationship was roughly separated by a single significant geographic hop. For example, a server on the ground in Palmdale to a client in Azure-West; a server in Azure-Central to a client in Azure-East.

Display Updates and the Tipping Point

Section When the server to client relationship can maintain a network latency in an average round trip time of roughly 30 mSec or less, the client display will quite predictably produce lower and lower display updates/s as the workload at the client is increased (more displays, more parameters, higher sample rates, etc.). The updates per second progress downward from a maximum of 64 to 0. Thus, a particular analyst can create a predictably repeatable display update performance in the range of 0 to 64 updates/s (with a presumption of running on similar hardware). Some analysts are known to create displays where are predicted to run in the range of 1 to 15 updates/s.

Clients which experience increasing network latency from the server will lose this predictably repeatable update rate decline at the tipping point. Displays which are already running somewhere in the territory of 15 updates/s will likely hit the tipping point if complexity is increased.

It is important to note that there are client complexity factors which are beyond the end-user's ability to control. Even though an end-user has strict control over the numbers and types of displays they want, they may not have any control over the underlying data which is visualized in the client. A parameter's defined sample rate or the complexity of an equation for a derived parameter could be changed without an end-user's knowledge.

Display Performance for Display Types

The client contains a wide variety of display objects which are available for real time data visualization and analysis. User-defined display types may also be added to the environment. Investigation of varying display types as it relates to hitting a tipping point was assessed.

Observations revealed that a display composed exclusively of the following display types could result in acceptable update rates up to a threshold number of displays:

- Alphanumerics: 65
- Stripcharts: 30
- Crossplots: 30
- Sliders: 60
- Annunciators: 65
- Alpha Table: 65 cells
- Frequency Plots: 45
- Attitude Direction Indicators (3 parameters per display): 20
- Text Primitives: 60
- Horizontal Situation Indicators (8 parameters per display): 5

It was further observed that scaling the numbers and types of display to combine different display types was also generally acceptable. For example, 50% of two types – 15 stripcharts together with 15 crossplots. 25% of four types – 15 alphanumerics, 8 stripcharts, 10 frequency plots, 15 sliders.

It is important to note that display types are not the sole arbiter of client performance. The numbers above were achieved using a moderate sample rate. As other performance factors are considered there will be observable changes to these numbers for acceptable performance.

Display Performance in Relation to Parameter Sample Rates

It was observed that parameter sample rates are a factor for reaching an unacceptable tipping point in display performance on network-latent systems. As an example, consider a display

composed of 30 stripcharts. Parameters of varying sample rates were then associated with all 30 displays. Sample rates of 10, 81 and 325 samples/s could produce acceptable display rates of 20, 19 and 18 respectively. 651 samples/s data hit a tipping point to become unusable.

651 sample/s data could achieve acceptable performance with 25 stripcharts. Similarly, 2604 sample/s data could be acceptable at 20 stripcharts.

Display Performance in Relation to Derived Parameters

Tipping points were clearly observed at lower data sample rates as compared to non-derived. As an example, consider a display composed of 30 stripcharts. Derived parameters of random complexity at known sample rates were associated with all 30 displays. Sample rates of 10, 81 samples/s could produce display rates of 20 and 19 respectively. 325 sample/s data hitting a tipping point to unusable performance

325 sample/s parameters with acceptable performance could be achieved with 27 stripcharts. 651 sample/s data were fine with 23 stripcharts. 2604 sample/s data was fine with 18.

Display Performance, Computed vs Viewable Updates

Irrespective of whether a client is co-located with a server or remote to a server, consideration must be made for how the results of the display updates will be made visible to a user. In a traditional mission control room environment, an analyst is viewing a monitor which is directly connected to the system on which the client software is running. It would be extremely rare for the client software to generate display updates at a rate which is greater than the ability of the system to display the result on the screen. A typical system will easily be capable of presenting all the frames which the client has computed.

For systems which are being viewed remotely, the display performance is constrained to the viewable update rate which is imposed by whatever technology is employed to remotely view the system. No attempt has been made to explore the many options available on the market. All remote viewing was performed using Microsoft Remote Desktop (RDP).

The following observations were made with regard to RDP for remote view of clients. The maximum viewable display update rate tends to be roughly 30 updates per second. As network latency between the client system and the viewer increases the maximum viewable update rate decreases. When the computed update rate at

the client is lower than 30 updates/s the viewable frame rate tends to lower itself to match the computed display rate. For example, an RDP connection which is capable of viewing the remote system at 30 updates per second will self-throttle to 15 viewable updates/s to match a computed rate of 15 updates/s. RDP is consistently capable of providing acceptable update rates.

Use of remote desktop to view a client co-located with the server is extremely useful.

General Client Operations

The period of time between client execution and reaching the desktop login dialog correlates to a constraint of network bandwidth. Execution of a client begins with a handshake to the sever. The client makes a request to the sever requesting deliver of the current mission configuration. This configuration file could conceptually be many hundreds of megabytes in size. A configuration file of 90 Mbytes being transferred locally is effectively instant in a local high bandwidth environment. When the server and client are not colocated together, there might exist a significant bandwidth limitation. The Palmdale, CA Curtiss-Wright facility to the Azure-West region was constrained to a 100 Mbit link. This bandwidth requires 20 seconds to transfer the client configuration.

When the desktop login dialog is presented to the user at the client, the time required to build the desktop to the time that an analyst could functionally utilize the displays takes a substantially longer amount of time on non-local clients. The time needed to create the desktop advances in a non-linear manner. For example, a 2 second local desktop login taking 25 seconds on a remote system and a 5 second login on a local system can take 125 seconds for the remote system. Another example, 12 seconds local became 6 minutes on the remote system. Users have the option of changing to a different pre-defined desktop configuration at any time.

Client desktops are composed of a series of individual Analysis Windows which are composed of varying display types. During real time operations the client software performs a great deal of pre-building to provide the capability to the engineer to rapidly switch between windows. The transition time from one window to another requires more time for latent clients. For example, a windows transition from Window A to Window B on a local client could take 2 seconds, but the same transition on a remote client could require 15 seconds before it is useable.

Client Analysis Summary

IADS client software components operate successfully in a cloud environment without modification.

However, there are several limiting factors induced by cloud-based or remote use of the software as compared to a traditional localized mission control room environment.

- Increased geographic separation of the client from the server to which it is connected induces additional network latency. This extra latency correlates to reduced display updates per second.
- Increased network latency induces the possibility of hitting a tipping point for minimum display updates per second.
- Interacting with a client using remote viewing technologies has a similar effect resulting in a reduction of viewable updates per second.
- As network latency between the client and the server increases, display and analysis complexity at the client needs to be reduced to achieve acceptable performance.

Current Considerations For Improvement

- 1.Client/Server protocol improvements.
- 2.Data Source to server protocol improvements.
- 3.Explore adding supplemental servers that are co-located with their clients.
- 4.Add performance warnings and alarms to the client.
- 5.Improve Analysis Window change times.
- 6.Improve Desktop build times Text

Conclusion

The viability of utilizing cloud services within the context of distributed flight test is here. Is it as seamless as an all-local and all-physical environment? No, but that does not preclude it from being useful in specific use-cases. The time for real world experimentation is here.

Appendix

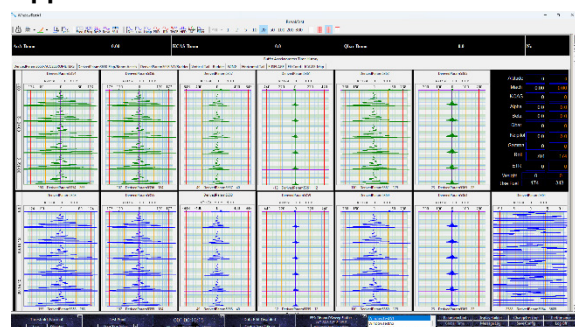


Figure 3 a simple real world example screen

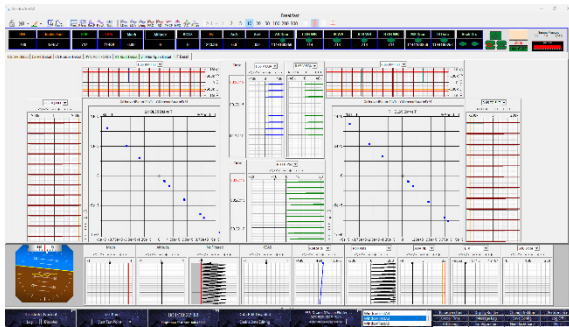


Figure 4 a moderate real world example



Figure 5 a highly complex real world example

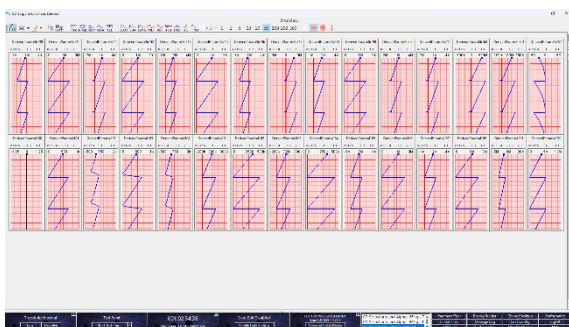


Figure 6 an example of an artificial test display