2013 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) MINI-SYMPOSIUM
AUGUST 21-22, 2013 - TROY, MICHIGAN

BEYOND VICTORY – CLOUD COMPUTING IN MILITARY VEHICLES

Mr. David Jedynak
Chief Technical Officer, COTS Solutions
Curtiss-Wright Controls Defense Solutions
Austin, TX

ABSTRACT

This paper will define a number of levels of abstracted cloud computing for vehicles. First, the transition from current architectures of dedicated hardware to basic dedicated virtualization of processes with hardware defined networks will be detailed using Commercial-of-the-Shelf (COTS) hardware and operating systems available now. Second, the transition to cloud-based processes with hardware defined networks will be detailed using the same hardware baseline. Third, the transition to software defined networks will be shown using the same hardware baseline. Once this new baseline of cloud based processing is detailed on existing hardware, the diffusion of the cloud onto an increased number of more SWaP-C optimized processing elements (specifically, ARM) will be shown. Each level of abstraction will be rated with a number of figures of merit and correlation to vehicle metrics / capabilities. A clear roadmap towards implementation will be shown, with defined phases to adapt and mature technologies to implement cloud computing in military vehicles.

INTRODUCTION

Use of dedicated computing and network assets in the enterprise is rapidly giving way to cloud-based architectures, leveraging multiple methods for virtualization of computing systems and distribution of computational tasks across physical assets. Recent advances in Software Defined Networking further enable the use of cloud architectures by abstracting the logical network from the physical network. The cloud approach gives enterprise data centers a number of advantages with regard to reliability, commonality, cost, power, scalability, and flexibility of the computing infrastructure over dedicated hardware approaches.

Like commercial vehicles, military ground vehicles utilize a number of processors and various internetworking standards to link and coordinate various control and operator systems. Unlike commercial vehicles, military vehicles of all types (ground, air, sea, manned, & unmanned) are rapidly becoming small scale mobile datacenters, overburdening the vehicle's constrained SWaP-C budget with sophisticated computing requirements. Multiple computational and internetworking systems are used to provide a myriad of mission and platform capabilities for the warfighter. Until recently, however, those systems remained completely separate and lacked any sort of interoperability. The US Army's VICTORY Architecture provides the important primary step of defining the network level interoperability between these various system, allowing the sharing and integration of data as well as initial steps towards the sharing of hardware assets. Nevertheless, most current C4ISR/EW systems still utilize dedicated hardware assets for processing with tightly defined network architectures.

Over the past several years, Curtiss-Wright performed research into network centric approaches specifically for Heavy Brigade Combat Team (HBCT) Vehicle Electronics. That research contained investigation into various cloud / process sharing architectures. Coupled with Curtiss-Wright's expertise in multiprocessor High Performance Embedded Computing (HPEC) for processing intensive ISR/EW applications, this paper will describe the next steps beyond VICTORY to create cloud-computing architecture for vehicles.

This paper will define appropriate Figures of Merit and how those correlate to vehicle metrics and capabilities. These will then be used in both the discussion of various levels of abstractions, and summarized in a set of tables.

FIGURES OF MERIT

To fully understand the benefits of abstraction to cloud computing, a number of figures of merit appropriate to the approach are presented below, with quick explanations. These will be applied to the various levels of abstraction as described later on in this paper.

Overall Cloud Processing Capability

This is simply the sum of all the processing capability of all the processors in the defined cloud. It can be measured any particular way, such as FLOPS, MIPS, or as a normalized dimensionless performance benchmark against a known processor. To stay focused on the relative merits of cloud abstraction versus the particular capabilities of a current generation of processor technology, the normalized approach will be used in this paper.

For example, a system containing three processors of performance levels 0.5, 1.0, and 1.25 would have a total cloud capacity of 0.5 + 1.0 + 1.25 = 2.75.

Useable / Unusable Capacity

Just because a processor has a particular performance capacity doesn't mean all of it is useable. A processor which is able to run only a single process will have essentially a useable capacity equal to the demands of that single process. For example, if a single-task processor has a process running on it that utilizes 25% of the processor, then the useable capacity is 25% and the unusable capacity is 75%. If, on the other hand, a processor is able to run multiple tasks, and is able to do that with only 5% required to manage the scheduling of the multiple tasks, then the processor's useable capacity is 95%.

This can be applied to Overall Cloud Processing Capacity (OCPC). Taking the example above, if the third processor (1.25) is single-task only with a task consuming 20% and the other two processors are multi-task with a scheduling overhead of 5%, then the Useable Overall Cloud Processing Capability is:

$$(0.5 * 95\%) + (1.0 * 95\%) + (1.25 * 20\%) = \\ 0.475 + 0.95 + \underline{0.25} = 1.675$$

Attention is drawn to the Usable Capacity of the third processor specifically to demonstrate that significant raw processing capability can be unusable, and in fact contribute less than a lower capability processor to the Overall Cloud Processing Capability.

Network Overprovisioning Overhead

Rather than in the commonly used context of bandwidth allocation for Quality-of-Service needs, Network Overprovisioning here is used in the context of additional links and infrastructure equipment in order to meet a particular performance goal. The figure here is counted in an overhead of additional network ports, with the normal assumption that a single processing element would require a single port attached to it. Anything more is considered overprovisioning overhead, and adds additional ports, often in pairs (one at the processing element, and another at an infrastructure device, such as a switch).

For example, a normally provisioned processor has a network requirement of 2 ports (processor and switch). A processor overprovisioned for redundancy has a network requirement of 4 ports (2 processor and 2 switch). The overprovisioning overhead is therefore the excess, a total of 4 minus the base of requirement of 2, equaling 2.

Failure Criticality / Cloud Resilience

Failure analysis is a deep and well-studied subject. Rather than apply a deep and formal Failure Modes, Effects, and Criticality Analysis (FMECA) to this discussion, two simple high level metrics are introduced, Failure Criticality and Cloud Resilience.

Failure Criticality is measured in three levels:

- <u>Function Lost</u> – the essential function of the thing is no longer available
- <u>Function Maintained</u> – the essential function of the thing is available at normal performance levels
- <u>Function Degraded</u> – the essential function of the thing is available but at a degraded performance level

Cloud Resilience is a similar, related concept, and provides the measure of the cloud to survive a number of physical failures. It is described as follows:

- <u>No Resilience</u> – any single failure means the cloud suffers a Function Lost
- <u>X Redundancies Resilience</u> – a set of X failures in redundancies means the cloud suffers a Function Lost
- <u>Limited Process Resilience</u> – the cloud can experience failures but suffers at most a Function Degraded
- <u>Full Resilience</u> – the cloud can experience failures but those result in Function Maintained

It is implied that a maximum number of tolerable failures is provided in the Limited Process Resilience and Full Resilience cases.

Total Cost of Ownership (TCO)

This is a standard metric, usually expressed in monetary values. For this discussion, similar to processing capability, it is expressed in a normalized qualitative value against the TCO for a single component (processor or infrastructure). At a high level, it is broken into three major portions:

- Non-recurring Acquisition Cost
- Recurring Unit Cost
- Sustainment Cost (included Training)

Total Cost of Ownership will be used in the context of the total system comprising the cloud.

Supportability / Maintainability / Reliability

These are standard terms for acquisition; however in this context, the key focus is on qualitative metrics (High / Medium / Low).

APPROACH TO VEHICLE METRICS

The various figures of merit are correlated to the key vehicle metrics of:

- Size, Weight, and Power and Cooling (SWaP-C)
- Survivability (as it pertains to C4ISR/EW)
- Degradation (as it pertains to C4ISR/EW)

SWaP-C is largely dependent on physical number and type of components. The efficiency of use of those components (e.g. Useable Capacity, Network Overprovisioning Overhead, etc.) is an area to examine to reduce SWaP-C waste.

Survivability and Degradation are closely coupled to Cloud Resilience and Failure Criticality. The overall Survivability of a vehicle's C4ISR/EW systems depends on the Cloud Resilience. Similarly, the Degradation of the vehicle's C4ISR/EW systems depends on individual function Failure Criticality.

In that context, the core concept of "move / shoot / communicate" for a vehicle is discussed, driving to a logical and reasoned connection between vehicle level cloud computing and essential vehicle capabilities.

LEVEL 0 – FUNCTIONAL ENCAPSULATION

The decoupling of systems from proprietary and legacy interfaces is the fundamental layer of abstraction, upon which everything else grows. Proprietary interfaces may still exist in deeper levels of design, but are encapsulated via standard interfaces. VICTORY provides that Level 0 abstraction by ensuring that functions (either as a physical component or a virtual service), are formally encapsulated with open standard interfaces on a networked databus.

With full implementation at this level, the function itself is no longer intertwined with the specific implementation of hardware and software tasks (sub-functions) used to achieve the function. This is not to say that the *performance* of the function is not dependent on the specific hardware and software to implement the function, as the quality, optimization, and capability of the implementation has a direct correlation with the performance of the function itself.

For example, a Position Function is fundamentally a location and accuracy, coupled with a refresh rate (timeliness and availability). A surveyor and his tools of the trade can provide highly accurate information, but not with a very high rate of refresh. A cellular device with understanding of local tower locations can give a high rate of refresh, but with a poor level of accuracy compared to the surveyor, or a more advanced system, such as GPS. Nevertheless, if all these systems provide their fundamental data in a standardized format, other functions can use this information at will, regardless of the underlying technologies used to gather the information. With full disclosure of the quality of the information, the overall performance of the system in meeting its required capabilities can be assessed, optimized, and accommodated.

In this fashion, a box is drawn around the function, and the box is painted black. This is VICTORY today, as shown in Figure 1, depicting Functions A-D attached to the VICTORY Databus (VDB).

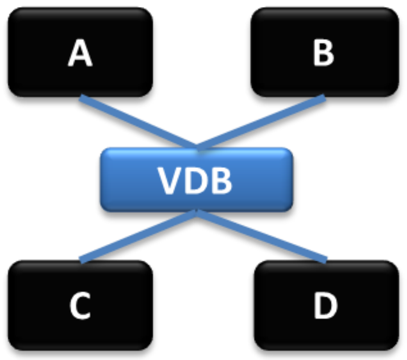

Figure 1: Level 0 – Functional Encapsulation

LEVEL 1 – APPLICATION ABSTRACTION

With Level 0 abstraction, basic components and services (functions) are now black boxes. The internal implementation is no longer relevant for its own sake, and is only important as it pertains to the quality of the function. Consider again the example of position information. A black box function provides position with a given quality using a standardized interface. For an application (such as moving map) to utilize the function, it must have access to the standardized interfaces. Without Level 0 abstraction, an application would be highly intertwined with the particular proprietary interfaces to the function. With functional encapsulation, the application only has a strong dependency on the standardized interfaces, and relies heavily on the presence of those interfaces in its operating environment.

Application abstraction comes about by ensuring that all applications are created to solely rely on the standard interfaces, disallowing the use of a proprietary interface to both functions and its local operating environment. At this point, significant concern can arise in that application developers may urgently cite multiple operating environment dependencies which are thought to be essential to the application's performance. This is a valid concern when viewed in the context of "external functions" (e.g. position function) and "local capabilities" (e.g. hardware accelerated graphics for mapping graphics); however, reframing the graphics capability instead to an encapsulated function, with a standard set of interfaces (e.g. OpenGL), preserves the model of application abstraction from the local operating environment.

The System level capability of "show my current location on a moving map" becomes decomposed into an application which joins a position with a map, receiving position from a position function and providing output to a graphics function that simply renders graphical elements per the desired graphical design (e.g. a blue icon with accuracy circle layered on a satellite photograph).

The critical step here is that the application no longer needs to run on a particular processor associated with the particular non-standard I/O of a function's inner-workings. The application solely needs to run on an open standard networked host which ensures network connectivity to the application's required functions, as shown in Figure 2.

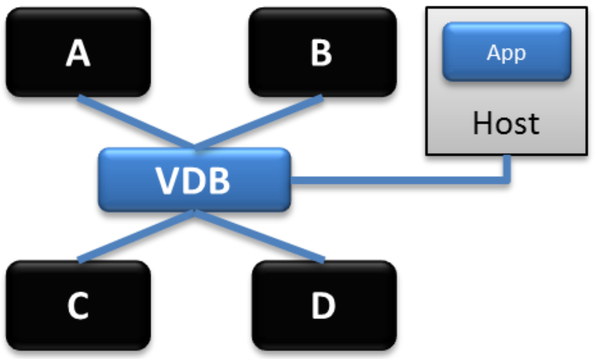

Figure 2: Level 1 – Application Abstraction

A host processor with a standard operating environment which provides access to the networked databus is the fundamental requirement for the application to run. A standard Commercial-of-the-Shelf (COTS) hardware host running a standard operating system (e.g. Linux) can provide this basic environment.

When multiple applications are involved, the simplistic approach is to deploy additional hosts, one for each application, in a single-task manner, as shown in Figure 3.

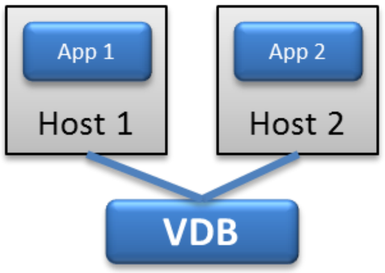

Figure 3: Single-task application hosting

Unless the host processor capabilities are exactly sized to the application processing requirements, this approach results in significant unusable capacity. For example, if the applications each require 50% of the host processor, and the host processors have processor capabilities of 1.0, then the Overall Cloud Processing Capacity is 2.0, but the Useable Overall Cloud Processing Capacity is (1.0 * 50%) + (1.0 * 50%) = 1.0. From a SWaP-C perspective, half is wasted on unusable capacity. Other similar figures of merit have a level of waste as well, such as TCO and the various logistic metrics.

Since applications are abstracted at the operating environment level, various methods of virtualizing the operating environment can be employed, ranging from basic task scheduling (multi-tasking) in a shared environment, to virtual machines (e.g. Java), to virtualized guest hosts. The particular method is not of concern in this discussion, but the end result is critical – multiple applications reside upon a single host processor (see Figure 4), making better use of the SWaP-C, reducing TCO, supportability, and maintainability. Negatively, however, reliability can be seen as dropping, directly affecting the survivability of a vehicle since a single host failure results in multiple Functions Lost.

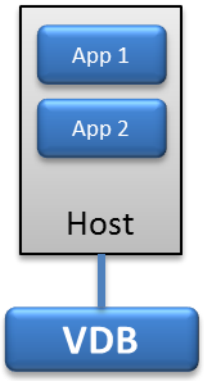

Figure 4: Multi-task application hosting

An important aspect of this is that the particular network connection from the databus to the host is well defined, and static (hardware defined). Reliability can be increased by overprovisioning, through the use of additional network ports from the host to the databus, but the fundamental topology of the databus (application on host connected at defined ports) does not change.

Simply progressing to this level of abstraction is an important step, but fails to realize the true benefits of cloud based computing concepts, as overall survivability is decreased since multiple Functions Lost can occur with a number of single failures (host failure, switch failure, cable failure). For this reason, pressing forward to the next level of abstraction is critical.

LEVEL 2 – HOST ABSTRACTION

An important aspect of cloud computing is abstracting the hosts of applications in such a way that the particular physical host is no longer relevant. All that matters is that there are hosts which create a cloud in which applications can run, as shown in Figure 5.

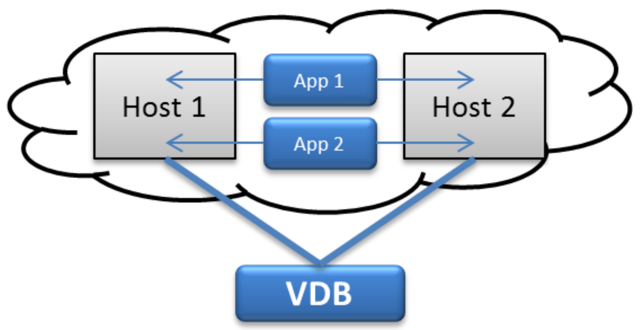

Figure 5: Level 2 – Host Abstraction (Host Cloud)

This approach immediately provides both benefits and drawbacks versus Level 1. The very nature of a cloud with multiple hosts capable of running multiple applications means that the redundancy starts to build into the system, as it's possible that a single host failure could result in Function Maintained, or at least Function Degraded. Now that a cloud exists, a level or Resilience starts to grow, at minimum X Redundancies Resilience based on the number of hosts in the cloud and the Overall Cloud Processing Capability versus the applications' processing requirements.

The immediate drawback is also clear: SWaP-C has gone back up from a multi-task host in a Level 1 approach. However, a subtle alteration to TCO occurs – it is possible that the multiple hosts are common part numbers, resulting in a better TCO than dissimilar single-task hosts. This is largely due to reduced acquisition and sustainment costs (single part), and secondarily due to increased volumes leading to reduced recurring costs.

From a vehicle standpoint, survivability increases, since the cloud allows for applications to run in any available host capacity within the cloud, and for the first time, a managed and graceful degradation is possible via prioritization of applications within the cloud (e.g. prioritize communications function over vetronics if that is the commander's intent)

Nevertheless, a major weakness is still apparent in the host abstraction – the network databus. The physical connection to hosts is a single point failure. The Network Overprovisioning Overhead involved in adding a secondary, overlapping databus, with multiple connections to each host and the various functions is not insignificant, resulting increased port counts, switch counts, and network management efforts, as shown in Figure 6. In this case, the Network Overprovisioning Overhead is 6 additional ports (assuming a single function for the example), 3 additional cables, and 1 additional VDB infrastructure switch. Taken as a whole, this is a doubling of network items.

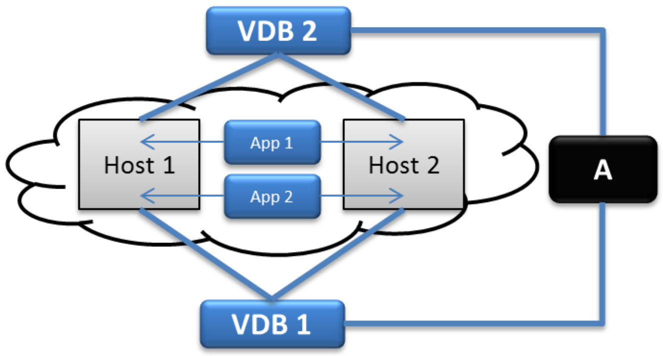

Figure 6: Redundant Network for Host Cloud

Nevertheless, two faults, one on each redundant databus (in broken yellow), could degrade the cloud such that a Function Lost occurs, as shown in Figure 7. The cloud loses half of its capacity from the view of either of the two redundant networks despite the fact that both hosts in the cloud are still functioning.

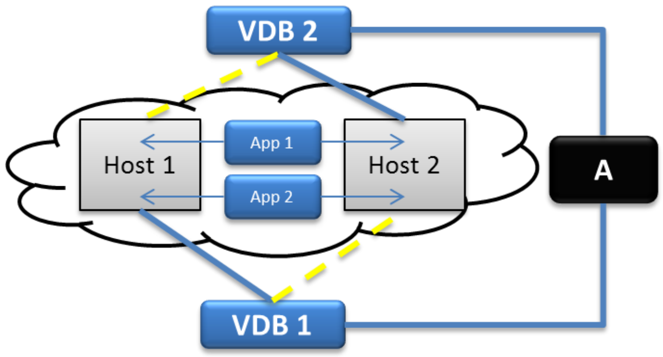

Figure 7: Broken Redundant Network

Ultimately, the hardware-based network becomes a limitation to the potential of abstracted hosts, and demands a further level of abstraction, involving the network itself.

LEVEL 3 – NETWORK ABSTRACTION

With the advent of more capable processors, able to process network traffic in software at rates previously needing purpose-built network hardware, software defined networking (SDN) becomes possible. Similar to the abstraction of hosts into a cloud, at its core, SDN abstracts a group of physical network switches into a single network switch fabric, which appears essentially as one giant switch, with software driven logical topology. Without SDN, the simple modification to the Broken Redundant Network in Figure 7 of connecting the two databuses together would require significant careful network management to ensure that the network remained converged and coherent. Each switch would need to be individually managed to carefully update the topology to return the cloud to a fully connected state. With SDN, the network is seen as one manageable entity, by a central management application, which itself can reside in the host cloud, as shown in Figure 8 (still showing the two broken network paths).

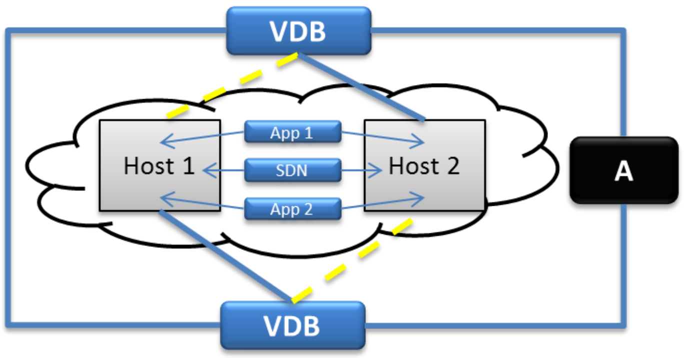

Figure 8: Level 3 – Network Abstraction

The key to this level of abstraction is that the system itself becomes extremely flexible and resilient since any network topology and any application / host allocation can be performed, and can be managed dynamically. It is this very concept that results in a highly survivable system which can be managed through graceful degradation while driving the Useable Cloud Processing Capacity as high as possible.

Information Assurance Considerations

The very nature of cloud computing and software defined networking is complete control and encapsulation of data and processes. Despite the mix of processes and data on a common set of network and processing hosts, separation of enclaves is, by its very nature, absolutely critical to the proper function of a cloud. The underlying cloud technology will need assessment and concerns addressed, after which it should be a common platform for multi-enclave systems.

Real-time Ethernet concerns

As discussed in the previous paper "Open Standard Approach for Real-time Control over Ethernet" (D. Jedynak), the real-time performance of applications on the cloud can be addressed using common shared clocks and well-defined multi-task scheduling / virtualization.

END STATE – DIFFUSION

Given both host and network abstraction, an interesting and useful phenomenon can occur – diffusion of the entire system into all available computing assets.

Revisiting highly federated pre-VICTORY systems, each has its own dedicated single-task processing systems and no real interconnection between functions. The Overall Cloud Processing Capacity was high, but with very low useable capacity. Failure Criticality was often high (Lost), and there was no Cloud Resilience at all, as shown in Figure 9.

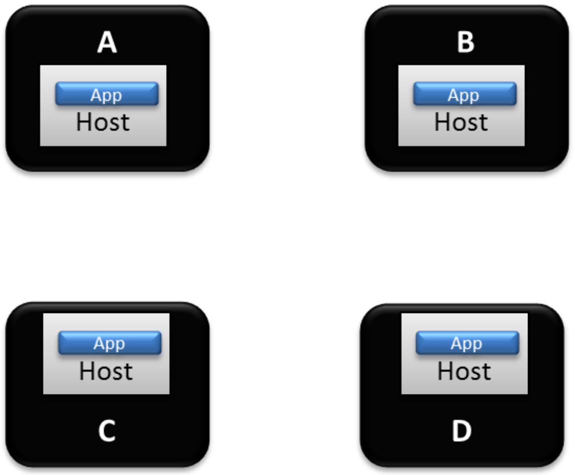

Figure 9: Pre-VICTORY Federated system

When considering a fully implemented VICTORY system implementation, including VICTORY interfaces on all systems, regardless of how insignificant (e.g. a 28VDC power distribution unit), it becomes immediately apparent that the very hosts required to provide VICTORY interfaces can immediately join a cloud through the use of open standard cloud management software. Through multiple network interface ports on the host processors, standalone SDN-capable switches are rapidly augmented with SDN on the individual multi-port hosts in each of the various functions. The end state is that the very concept of centralized host processors for applications evaporates into a diffuse SDN-connected processing cloud across all of the various functions installed in the system, as shown in Figure 10.

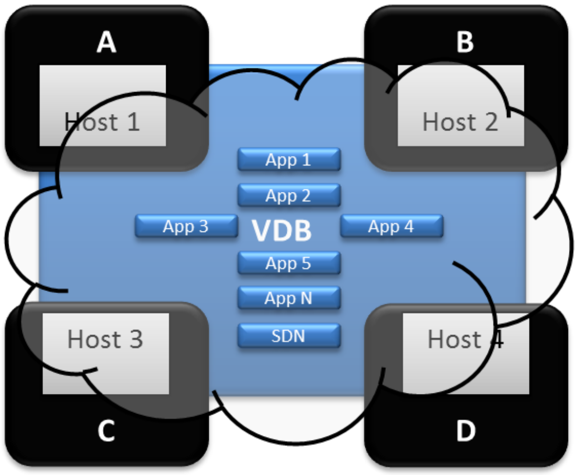

Figure 10: End State – Diffusion

What becomes even more interesting is the realization that the diffusion of the cloud means that previously unusable capacity is now available to what was previously overburdened capacity in other systems. For instance, a modest ARM processor in a power controller could now provide critical processing capability to a sensor or software defined radio application which is currently driving up significant SWaP-C, TCO, and other negative metrics (e.g. poor reliability due to excessive thermal).

In order for this to happen, the next step is to understand the system as one large task, which is then distributed across its cloud automatically. This sort of task management is common in the "Big Data" applications.

HADOOP AND AN ARMY OF ARMS

"Big Data" involves the massively complex processing of data which is generally unmanageable (too big) by common discrete tools. The fundamental approach to Big Data is to Map the task out to a very large number of sub-tasks, then Reduce the results of the sub-tasks to easily useable / actionable / presentable information. This complex approach is fully implemented in the Open Source High-availability distributed object-oriented platform (HADOOP). It reduces complex tasks into small chunks which can be run on numerous modest commodity hardware hosts, including a distributed file system, and even handles lost processing nodes.

Embedding low cost and SWaP-C optimized ARM processors supporting HADOOP on a vehicle would provide a Fully Resilient Cloud for operating the vehicle and its C4ISR/EW needs.

The effect of this is dramatic. SWaP-C would be significantly more optimized because the untapped processing capability of extremely low cost processors would be available to even the most sophisticated algorithms needing to run on the vehicle. The resilient nature of the cloud would mean that failures due to damage would be fully mitigated or gracefully degraded per the current priorities of "move / shoot / communicate", resulting in a higher level of overall vehicle survivability. The processor technology aspect of TCO could be largely sidestepped since the intent is to use modest processors with standard interfaces, embracing a COTS model significantly more than in current federated systems.

METRIC SUMMARIES

The following tables include summaries of the various levels and approaches.

Table 1: Pre-VICTORY Federated

Metric	Value
Overall Cloud Processing Capability	Sum of subsystem processors
Useable Capacity	Sum of subsystem processes
Network Overprovisioning Overhead	None (no network)
Failure Criticality	Per subsystem, Function Lost
Cloud Resilience	None
TCO	Sum of subsystems
Supportability / Maintainability / Reliability	Per subsystem
SWaP-C	Sum of subsystems
Survivability	Independent systems
Degradation	Per subsystem

Table 2: Level 0 – Functional Encapsulation

Metric	Value
Overall Cloud Processing Capability	Sum of subsystem processors
Useable Capacity	Sum of subsystem processes
Network Overprovisioning Overhead	None (single network)
Failure Criticality	Per subsystem, Function Lost
Cloud Resilience	None
TCO	Sum of subsystems
Supportability / Maintainability / Reliability	Per subsystem, with some increase in supportability due to open standards for interfacing
SWaP-C	Sum of subsystems
Survivability	Independent systems
Degradation	Per subsystem

Table 3: Level 1 – Application Abstraction (reduced hosts)

Metric	Value
Overall Cloud Processing Capability	Sum of host processors
Useable Capacity	OCPC – multi-task overhead
Network Overprovisioning Overhead	Negative (reduced connections)
Failure Criticality	Multiple Function Lost possible
Cloud Resilience	None (single fault)
TCO	Reduced Acquisition and Sustainment
Supportability / Maintainability / Reliability	Higher Supportability and Maintainability due to merged hosts, lower reliability (single point)
SWaP-C	Lower (merged)
Survivability	Lower (single point)
Degradation	Allows some management

Table 4: Level 2 – Host Abstraction Cloud (no redundant network)

Metric	Value
Overall Cloud Processing Capability	Sum of cloud processors
Useable Capacity	OCPC – multi-task overhead
Network Overprovisioning Overhead	None
Failure Criticality	Potential for Function Maintained and Function Degraded. Single points still possible
Cloud Resilience	Limited Process Resilience
TCO	Potential for reductions (commonality)
Supportability / Maintainability / Reliability	Potentials for improvement in Sustainability and Maintainability with commonality, higher Reliability
SWaP-C	Sum of hosts and network
Survivability	Higher due to resilience
Degradation	Allows significant management

Table 5: Level 2 – Host Abstraction Cloud (redundant network)

Metric	Value
Overall Cloud Processing Capability	Sum of cloud processors
Useable Capacity	OCPC – multi-task overhead
Network Overprovisioning Overhead	Equal to original network
Failure Criticality	Potential for Function Maintained and Function Degraded. Single points still possible
Cloud Resilience	Limited Process Resilience
TCO	Potential for reductions (commonality)
Supportability / Maintainability / Reliability	Potentials for improvement in Sustainability and Maintainability with commonality, higher Reliability with network redundancy
SWaP-C	Sum of hosts plus double network
Survivability	Higher due to resilience
Degradation	Allows significant management

Table 6: Level 2 – Network Abstraction

Metric	Value
Overall Cloud Processing Capability	Sum of cloud processors
Useable Capacity	OCPC – multi-task overhead
Network Overprovisioning Overhead	Ranges based on network reliability required (single, dual, multiple, or mesh connected)
Failure Criticality	Potential for Function Maintained and Function Degraded. Single points at edge functions only
Cloud Resilience	Full Resilience
TCO	Potential for reductions (commonality)
Supportability / Maintainability / Reliability	Potentials for improvement in Sustainability and Maintainability with commonality, higher Reliability with full resilience
SWaP-C	Sum of hosts plus network
Survivability	Higher due to resilience
Degradation	Allows significant management

Table 7: Level 2 – Diffusion

Metric	Value
Overall Cloud Processing Capability	Sum of cloud processors
Useable Capacity	OCPC – multi-task overhead
Network Overprovisioning Overhead	Ranges based on network reliability required (single, dual, multiple, or mesh connected)
Failure Criticality	Potential for Function Maintained and Function Degraded. Single points at edge functions only
Cloud Resilience	Full Resilience
TCO	Potential for reductions (commonality)
Supportability / Maintainability / Reliability	Potentials for improvement in Sustainability and Maintainability with commonality, higher Reliability with full resilience
SWaP-C	Blended back into edge functions (big reduction)
Survivability	Higher due to resilience
Degradation	Allows significant management

ROADMAP FORWARD

Rather than attempting to move from Level 0 (VICTORY) through Levels 1, 2, and 3, the better roadmap is to force diffusion immediately by requiring that all new systems support the common cloud environment (e.g. HADOOP) on any processing elements. Mission Computing assets should be the first target, along with small microprocessor systems using ARM or other highly optimized processors in edge-functions. This greedy approach will immediately free up a significant amount of unusable processing capacity for use in over-capacity systems, leading to a first round of load balancing.

The next major step is to create the unified task model of the vehicle, showing it as one large “Bit Data” style process, which can be simulated and deployed on the actual system, fully integrated and proven.

CONCLUSION

The next steps beyond VICTORY lead to a formidable level of SWaP-C optimization and survivability for Military Vehicles. With a holistic vehicle task view, cloud computing concepts can be used to great benefit in a creating more capable, survivable, and supportable vehicle.