OpenVMS Intracluster Communications

Investigation Report

by

John Covert

with assistance from

Greg Jordan
Niccole Luce
and others

VMScluster Engineering

2 April 1996

Rev 1.2

1. Introduction

The most significant distinguishing technology available in OpenVMS is the VMScluster capability. Developing this capability and enhancing the ability of VMSclusters to support the high end of the three-tier client server model is key to the success of OpenVMS and Digital Equipment Corporation.

The ability to provide an environment in which middleware providers can implement highly reliable and highly available servers within a VMScluster will further enhance the distinguishing characteristics of OpenVMS. To do this, we must provide reliable and efficient Intracluster Communications to allow applications within a cluster but distributed over several nodes to communicate with each other.

There are currently no supported Intracluster Communications methods other than those provided by networking products such as DECnet and TCP/IP. Independence from these products, which vary in availability from one system to another, is seen as a critical enhancement to OpenVMS. There are numerous Intracluster Communications needs today which cannot be met because of the lack of knowledge about which network product might be installed. We know of internal requirements and we believe there are also external requirements.

We currently believe that the needs in this space are primarily by OpenVMS components, middleware providers, and database vendors. There are also some customers who would write to a VMScluster specific API to gain access to high speed interconnects.

2. Overview of Intracluster Communications

2.1 Our Definition of Intracluster Communications

We define Intracluster Communications as Process to Process communications within a VMScluster. To be useful, Intracluster Communications must be callable from processes while running in either user mode or kernel mode. But most importantly, Intracluster Communications must work independently of any installed or running network software, without any considerations about which choice the customer might make between DECnet, TCP/IP, or third-party networking.

2.2 System Communication Services (SCS)

VMScluster software currently provides a communications technology which guarantees communications within a VMScluster: System Communication Services (SCS). By definition, every node in a VMScluster must have a direct SCS connection to every other node.

SCS can be used by applications by writing code to call the SCS interface routines. An application which uses SCS routines for Intracluster Communications has the following characteristics:

- It can perform node to node communications within a VMScluster.
- It is highly available.
- It works regardless of the type of networking software.
- It works regardless of a running network.
- It can function the very early stages of system startup.
- It gains access to high speed cluster interconnect hardware.

2.3 Options for Intracluster Communications Today

Today there are two methods for Intracluster Communications using SCS:

2.3.1 Call the SCS routines directly.

Programming an SCS SYSAP is not a trivial task and requires experienced kernel mode coders. The designation "SYSAP" is used in this document and elsewhere somewhat interchangeably for a program which provides a VMScluster service through SCS routines, for the service itself, and for the transport connection between a client and a provider of a cluster service.

2.3.2 Use the IPC interface.

The IPC interface (not to be confused with the DECnet/OSI $IPC system service) is the only existing higher-level API for process to process communications over SCS. The IPC API supports several transports in addition to SCS (including DECnet and TCP/IP). The SCS interface is provided via the SCA$Transport SYSAP. IPC has both a system service interface implemented as a user-written (or protected shareable image) system service (which can only be used by user mode programs) and a kernel mode interface.

Neither the existing IPC system service nor the existing IPC kernel mode interface is currently a supported API and internal use is limited by the following restrictions:

- Association limitations (256 active associations per system). Since each client and each server consumes an association this limitation must be removed before IPC can be a general interface.
- Logical link limitations (2048 per node). For third-tier servers (the intended market for OpenVMS), this number is too small.
- Physical link limitations (256). For performance reasons (see next bullet), this limitation also needs to be removed.
- Performance bottlenecks. As the primary example, IPC currently multiplexes all IPC connections to the same remote node over a single

SCS connection through SCA$TP. This means that any IPC client doing very large transfers blocks all other clients until the operation is complete. We are currently testing modifications to the code to remove this limitation.

- Known failover issues. For example, the behaviour of SCA$TP when disturbances occur in the underlying SCS connection is often to simply bugcheck the node. Recovery mechanisms must be designed and implemented.

The following are the current known IPC Users:

- Spiralog (Kernel Mode Interface) (Spiralog plans to migrate from IPC to the QIO server).
- DECdtm (Kernel Mode Interface)
- Batch/Print (User Mode Interface)

These are known or possible future IPC Users:

- DCE RPC (User Mode) (prototype code running today)
- Argus Management Station (User Mode)
- Affinity Work (Kernel Mode)
    - Registry
    - Single Sign-on
- Streams (possibly)

3. Investigation

During this investigation we considered a number of options:

- Overhaul IPC and SCA$Transport.
- Provide a supported API to IPC.
- Provide a supported API directly to SCS, callable from any mode.
- Provide a completely new high level API, invented here.
- Provide a VCI interface to SCS.

3.1 Overhaul IPC and SCA$Transport

We must fix the limitations, and performance problems within the IPC and SCA$Transport implementation mentioned in section 2.3.2.

As a result of recent discussions with Kevin Playford and work in progress by Niccole Luce, we believe most of these restrictions can be removed. Some can be removed on a node-by-node basis without special effort; others may require “warranted pairs” of OpenVMS versions certified for interoperability.

We have considered completely decoupling SCA$Transport and IPC in order to fix the SCA$Transport problem without bringing permanent responsibility for the future maintenance of IPC into the Cluster group.

The issues that remain in this area are:

- Should this work be for VAX and Alpha or Alpha only?
- Can this work correctly in mixed version VMSclusters?

3.2 Provide a supported API to IPC

The existing unsupported API is provided in two completely separate flavors: a privileged shareable image (or user system service) which provides the interface from user mode, and a separate interface for kernel mode users. A program using either of these interfaces may communicate with another program using the same or the other interface without any special considerations.

A supported API should be a "real" system service, written to be callable from any mode. A preliminary specification for such a system service already exists. Since DECnet has already taken the name $IPC, a series of system services with names such as $TCI_CONNECT, $TCI_TRANSMIT, etc., has been proposed.

A characteristic of the new interface would have to be that it is usable without any special security implications (i.e. no privileges other than NETMBX should be required) and that it be possible to communicate between applications written with this new interface and applications written with either of the old interfaces without any special programming considerations.

3.3 Provide a supported API directly to SCS

This option would provides a new supported API which would allow users to directly access SCS services without using IPC. However, it would not solve any problems with IPC, and would not eliminate IPC until all applications currently using IPC could be rewritten and all cluster members upgraded. In the meantime, all applications would have to simultaneously support both methods.

3.4 Provide a completely new high level API

There was a specific request for a new interface to solve some specific needs of the Affinity project. The request was to provide a high level Intracluster Communications API with special capabilities needed for the Affinity project. It would be a single service for user through kernel modes, and would include a name service to find servers. It would provide automatic failover between clients and servers within the VMScluster and load balancing.

During the investigations we discussed whether this would be sufficiently general purpose to be used by other users. We also discussed who would be responsible for this interface, whether it would be documented, and whether it would communicate directly with SCS or be implemented on top of IPC.

3.5 Provide VCI on SCS

VCI is a standardized API to communications hardware (LAN and WAN device drivers) for use by networking products. Implementing a VCI interface to SCS (making SCS look like a private LAN connecting all nodes within a cluster) would provide access to SCS interconnects for all networking products and as a result to all DECnet and TCP/IP applications communicating within the VMScluster.

However, since a running network is still required, this doesn't solve the needs of OpenVMS for a network-independent Intracluster Communications solution.

We also are told that there are routing issues with TCP/IP which require further investigation.

4. Conclusions

After considering these options, we have reached the following conclusions:

4.1 Overhaul IPC and SCA$Transport

IPC and SCA$Transport need work, and we consider this a high priority item. Work on this is already in progress. The full scope of this work needs to be determined, but we need to become more familiar with the code before we can make final decisions on all the work to be done. As mentioned in section 2.3.2, we need to lift certain restrictions in IPC to make usable by more than the current limited number of users. Current users will benefit from these enhancements. Thus the high priority items for improvement in this overhaul are:

- Association limitations
- Logical link limitations
- Physical link limitations
- Performance bottlenecks
- Known failover issues

Decisions about future support have not been finalized. Because IPC calls not only SCA$Transport but also DECnet and TCP/IP interfaces, we remain unsure about the appropriateness of long-term ownership of IPC by the cluster group. However, the work is now underway and being done by Niccole Luce.

4.2 Provide a supported API to IPC

This appears to give us the most benefits of all the remaining options. It will interoperate with the existing undocumented interfaces without making them obsolete. It will work in mixed-version environments. Users of the supported API will benefit from all the enhancements in section 4.24.1.

John Covert is sufficiently familiar with implementation of multimode system services to begin writing a preliminary functional specification. We need to determine ownership and staffing responsibility. This is not an extremely large project, but it does require careful planning in order to reserve necessary data structures in future releases of OpenVMS.

4.3 Provide a supported API directly to SCS

This option doesn't really seem to provide as much of a benefit as enhancing IPC, specifically because it introduces a new interface to design, implement, and support without eliminating IPC in any reasonable timeframe, if ever. IPC could continue to be required forever to handle connections outside the cluster through other networking products which it currently supports.

4.4 Provide a completely new high level API

This option, too, does not seem to provide a benefit to a broad enough segment of the OpenVMS community to justify being done by the Clusters group. Instead, we propose providing consulting to the Affinity project to help them implement what they need on top of IPC, using the current interface if necessary for the short term, with easy migration to the new supported interface when it is ready. For the "name service" capability the Affinity project seeks, we propose a solution using the Clusterwide Logicals being implemented as a separate project.

4.5 Provide VCI on SCS

This, too, looks like a great thing to do, but not something for which we think we have resources to allocate at the present time.

5. Cost

5.1 Overhaul IPC and SCA$Transport

This project is owned by Dan O'Shaughnessy and Niccole Luce; they must provide cost estimates.

5.2 Provide a supported API to IPC

1. 6 weeks to write a Project Plan, Functional Spec, and Design Spec, including spec reviews and updates. (John Covert)
2. 3-4 weeks for an unspecified resource to learn background necessary to do the implementation. This number is somewhat of a guess, since we don't yet know who the resource will be.

3. 3-4 weeks for this same unspecified resource to implement the design provided and do initial testing.

5.3 Consulting to Affinity Project

1. 2 weeks over a 9 week period to assist with conceptualizing and designing a “name service” using clusterwide logicals.
2. 1 week during the same 9 week period to consult on failover and load balancing concepts and design using existing cluster event notification services and new services.
3. 3 weeks consulting during the implementation by the Affinity Project team on the use of various cluster services.

End of Report.