# VMS Project Plan for Multipath Failover

# DIGITAL CONFIDENTIAL

**Revision/Update Information:** X1.0.2

Glenn C. Everhart

24–February–1997

**Abstract**

This project plan describes the plan for implementing failover between multiple paths to devices where more than one path is presented by hardware (or, later, multiple servers).

**Review Team**

Pat St. Laurent Tom Coughlan Lenny Szubowicz John Croll

**Digital Equipment Corporation**
**Maynard, Massachusetts**

## Change History

**1.** X001–10-Jan-1997–Initial Draft

# 1 Overview

The multipath failover project exists to permit failover from one path to a device to another, regardless of the underlying device drivers, for disk I/O, when the disks enter mount verification and the path appears to have been lost (or when the equivalent happens to a shadowset member). Rather than insert code into many drivers, this is to be handled by adding some I/O switching processing logically into the beginning of existing drivers.

# 2 Business Justification

There are three reasons the work must be done.

1. OpenVMS will be connecting to a number of controller types in the near future which provide more than one path to storage. These include the HSZ4x/5x/7x systems, and also fibre channel connectors, which will generally have two loops and thus will in general have two paths to every device. It is essential that these be usable, so that customers will gain the expected fault tolerance, yet at present this cannot be obtained.
2. Users of shared SCSI busses expect that they can use a direct or a shared path to get to a disk, and that their systems can therefore tolerate faults in access. This is currently not true and is causing customer complaints.
3. When QIOserver becomes functional, failover between it, direct paths, and MSCP served paths will be required, as will means to let the rest of the system use only one invariant name for all storage and use only one path at a time.

The only path failover in clusters currently is on disks using MSCP protocols or for the MSCP server. The failover code in these areas depends critically on the ability built into the MSCP protocol to do switching, an ability not shared by present or contemplated SCSI systems. Project information for multipath failover may be found at:

- DS-MULTIPATH-SWITCH.WRITE (High level design document)
- DS-MULTIPATH-FAILOVER.SDML
- IR-MULTIPATH-FAILOVER.WRITE
- PP-MULTIPATH-FAILOVER.SDML
- Pointers to any other documents (schedules?)

# 3 Staffing

The project will need one engineer (Glenn Everhart). In addition, testing support from QTV and within the SCSI group will be needed for 4 person weeks.

# 4 Schedule Bullets

**Table 1: Schedule, All Work Sequential**

| | |
|---|---|
| 12/31/1996 | 1st draft of design specification for review |
| 1/25/1997 | 1st review complete (Holding this with Tom and Lenny and John Croll tomorrow 1/21/1997; your comments need to be factored in also.) |
| 2/1/1997 | 2nd draft of design specification for review (This will be a second 4-day limited review.) |

**Table 1 (Cont.): Schedule, All Work Sequential**

| | |
|---|---|
| 2/14/1997 | 2nd review complete |
| 2/14/1997 | Distribute design specification for group review. (I'd suggest sending it to OPENVMS_ BASE_IO and BASE_OS_CLUSTER, at least.) |
| 3/4/1997 | Test plan drafted and reviewed. Coding starts on moving existing function level to fully general form (n paths) |
| 4/2/1997 | Coding (but not debug or test) of the driver should be done. |
| 4/3/1997 | Coding of system support routines begin. |
| 4/10/1997 | Coding of most essential support routines (search, UCB hiding, postprocessing hooks, DK "reqcom bug" fixes, dk/du "look for other path" code done. (Not all of support but the most vital part for early testing.)) |
| 4/11/1997 | Start coding on control program for manual setup. |
| 4/18/1997 | Control program coded enough to do manual setup of sets of paths or issue failover requests. |
| 4/21/1997 | Unit Testing together of parts built so far begins. |
| 5/20/1997 | Unit test of driver done. (Partial unit test of other code necessarily done also.) At this point, switching functionality exists in barest form. |
| 5/21/1997 | Coding of remaining exec modifications begins |
| 5/28/1997 | Coding of exec mods done; coding of remainder of control & display program starts. |
| 6/4/1997 | Control program coded, built. |
| 6/5/1997 | Server coding begun (keepalives, switch on HSC signal, initial scan of devices, recognizing pairs) |
| 6/19/1997 | Server coded & unit tested. New functionality completed (as outlined in project plan and in that order.) Some unit testing runs concurrently (asking Bill Clogher for a trial or three) |
| 6/19/1996 | Unit testing, whatever else is needed |
| 7/21/1997 | Code inspection and unit testing complete |
| 7/28/1997 | Finish update of design document to reflect actual code |
| 7/29/1997 | Send out final design document for comment and possibly schedule review if changes have been unexpectedly extensive. |
| 7/29/1997 | Distribute code to other groups for testing also. |
| 8/1/97 | Testing complete |

Note: This is rather ambitious and does presume no serious glitches will occur. The "server" bullets noted include relevant code in other modules which will need to handle "the other end" of the communications which will go on. The functioning of the testbed code suggests this is not as far fetched as it may often be, but unknown issues are after all unknown.

The scheme to have partial function implemented early however is intended to provide assurance that if it becomes necessary to trim functions, that will still leave at least basic functionality usable.

Should the driver be written in parallel with the other code in the project the job is split roughly into two halves. The schedule bullets in that case look like this:

**Table 2: Schedule, Switching Driver Only**

| | |
|---|---|
| 3/4/1997 | Test plan drafted and reviewed. Coding starts on driver (and should begin on the rest also) |
| 4/2/1997 | Coding (but not debug or test) of the driver should be done. |

**Table 2 (Cont.): Schedule, Switching Driver Only**

| | |
|---|---|
| 4/3/1997 | Unit testing of driver begins; REQUIRES that the control program be able to set up path pairs manually and request switching manually by this time. |
| 5/6/1997 | Unit test of driver done. (Partial unit test of other code necessarily done also.) At this point, switching functionality exists in barest form. |
| 5/6/1997 | Schedule final code inspection & review |
| 6/2/1997 | Code inspection and unit testing complete |
| 6/15/1997 | Finish update of design document to reflect actual code |
| 6/15/1997 | Send out final design document for comment and possibly schedule review if changes have been unexpectedly extensive. |
| 6/2/1997 | Distribute code to other groups for testing also. |
| 7/3/97 | Testing complete |

## 5 Schedule Discussion

A two path implementation of switchover exists already, written largely in Macro-32, which provided basic switchover and support for manual switchover. The driver has around 4500 lines of code. The final code has however been directed to be written in C. This is partially done (1300+ lines of code) and another perhaps 2000 lines remain to be written to duplicate existing functionality, and perhaps another 100 for some other functions needed (e.g., forcing idle). The 2 path code provides a detailed design framework for the bulk of this, and most of the rest can use example code in some other drivers I have available for guidance.

There must also be utility code to switch paths, report statistics, and do manual connections or disconnections written. Much of this function exists within the Macro32 code also, but needs to be redone in C. The Macro version is about 1600 lines long. The new C version will however not need all of the driver setup that is provided in the Macro32 version, since this functionality is being moved into the driver. Path switching to the next available path is also in the driver so that some shrinkage might occur. Thus estimate 1000 lines of code for this utility code.

Also there are needed a number of modifications to the boot path which will ensure the same names for devices, and to DUdriver (DUTUSUBS actually), DKdriver, MKdriver, and possibly the MSCP server, to ensure that disk paths all have the same name and all load and connect switching code if needed, and hide any non-first-seen UCBs. IOSUBNPAG will also need some additions to its common search code. The current recognition code for DKdriver that looks for MSCP served paths takes 256 lines of source. For the overall magnitude of these changes, therefore, we will presume about 1000 lines of code. These modifications are to Macro32 code and will therefore be in Macro.

The server code which enforces policy globally may turn out not even to be needed initially (the review has not yet taken place) if one can live without keepalives, special controller-specific path policy issues, or performance enhancing preeemptive switching. However, to get the server architecturally included, these things seem worth having. For an initial build I would leave controller specific policy issues out (and just use round robin path choice) until switching is better understood in practice. Doing this means a small server which responds to HSZ controller failure would be all that needs to be built. This may turn out to be 500 lines or so. Even this may turn out after review to be deferred until later however.

Thus lines of code still to be done are 2100 + 1000 + 1000 + 500, or 4600 lines of C and 1000 lines of Macro in all. At a speed of 50 lines/day for workable C code and 100 lines/day for Macro this is ~82 days' effort. As a rough guesstimate I am estimating 50% of this time is for coding itself and the other 50% for bug squashing.

The driver will contain the logic to do all the actual path switching work and path setup, so that other modules need only issue specialized $QIO calls to it to control it. Therefore it is entirely feasible that the process code for manually switching, hooking up paths, and reporting statistics to be done by someone else. If this path should be followed, some throw-away test driver code would need to be built to test the new driver early in the game if the "real" test code were not available before then. This might however just be a rework of some of the existing macro code and take a couple days to get together. The server code could also be split off. This would move perhaps 1500 lines of code (crudely 30 days of effort) to where it might be done in parallel. Moving path searching code and other support code would be more involved (since the switching driver needs to know those interfaces fairly intimately to perform its lookups and switching) but this can also be done. This is at an estimate 10 days (2 work weeks) of effort.

If the schedule should be run in parallel, development time for driver code thus works out at 42 days, and time for the rest at 40 days.

In mixed clusters, when QIOserver client access exists, the switching problem will exist even on VAXen, so the switching driver and client end of the cluster path detection will need to be there also. When this is to be supported, the switching driver in Macro and the Macro utility code may need to be brought back and updates completed on them. The Alpha driver has been ruled to be done in C. However, the actual Vax work (code modification and tests) will be deferred until after the Alpha code is ready, since the dual path situation can be avoided on the Vax platform until after QIOserver paths exist. The server component will be written in C for portability and easier debug.

An additional month for "unit" testing is needed. Such testing will need to verify that dual path configurations work in all combinations of the following:

1. Standalone uniprocessor
2. Standalone multiprocessor
3. Clustered uniprocessor
4. Clustered multiprocessor
5. Dual path HSZ connects on single systems
6. Dual path HSZ connects on shared SCSI busses
7. Dual path HSZ connects on unshared SCSI busses on different cluster machines.
8. Clients seeing dual served paths (to be done after QIOserver exists only; not feasible until then.)

The tests with dual path HSZ will cover served access as well as local for some of the configurations described (see pictures in the investigation report). It is expected that another month or so of QTV testing will be needed also after unit test.

Finalizing documentation will be done concurrently with the QTV driven testing and will take at least 2 weeks. This is new functionality and the code must be documented well enough to facilitate addition of more special-case code for different types of controllers, and the system for I/O interception will need to be well described for the benefit of any third parties who may use it. (It is to our mutual benefit if they use a compatible intercept. We gain assurance that the I/O posting path can be made shorter. They gain assurance our intercepts don't interfere with theirs.)

The approximate order for implementation will be

1. The switching driver
2. I/O exec infrastructure (including naming)
3. The control/display program code, productizing the existing functions
4. The rest of the control/display program
5. The server, starting with probing of HSZ functions
6. The DKdriver mods to report HSZ specific messages
7. Tests of server run late in system operation
8. Final bits of DKdriver and DUdriver support and server handling of messages

The move to a C driver means no function testing can begin until at least the driver and the more vital systems routine support code is done. This will be done first, however. The control program will be coded interleaved here, the idea being to produce something that can manually set up paths or request path switch first so the driver functions can be debugged. Once this basic functionality can be achieved, the rest of the package can be added in, testing at each stage to ensure against breakage. This should minimize unpleasant surprises later on and discover any problems early. Occasional calls on SCSI group test support are envisioned to run a go/no-go test or two to ensure things are still sane. This is a risk minimizing strategy.

## 6 Testing Plans

The system will and must be tested incrementally during build, as well as stress tested after completion. The design specification is fairly detailed since it draws on lessons learned building a working test model of the switching logic. Nevertheless, a complete design document covering the final system will be written and reviewed, and the code will be subject to review, though its extent is such that a formal review is likely to be infeasible. Since it is likely that the QIOserver people will have their hands in this code as it is being built, though, it is expected that it will have considerable informal scrutiny during construction which would normally not be had.

Since modifications to the rest of OVMS are minimal and will in general be done first so that normal Raven testing will have long exposure to them, it is expected further that the testing proposed to verify correct functioning will be all that is needed and that cross interactions with other OVMS components will never be seen.

A final formal design review will be held, as a last chance to identify design problems, as soon as the final design documents can be prepared, in addition to the reviews of the design specification to be held after the current review by a few people of the first draft.

## 7 Performance Issues

The performance of the subsystem must be tested relative to unswitched paths. This is simply done by insertion or removal of the switching intercept (which is designed to support this). The time cost of the switch must be below 0.1% of the time needed in main path I/O. Also, the setup code must not insert the switching code where no multiple paths exist, so that no overhead will be added there.

## 8 Project Dependencies

The initial work of this project has only one serious dependency: it MUST be able to make some exec changes (and preferably a couple of upward compatible structure changes also, without which meeting the performance goals will be difficult).

The "REQCOM" problem (which in DKdriver manifests also as not having all IRPs call driver start_io) must be solved.

It should be noted that support of switching of /FOReign - mounted disks failover is to be limited to manual failover on command. Thus no mount verify dependencies are known.

The SHdriver interactions are only generally known at this point, and it may be necessary to get some additional information between SHDRIVER and the failover code to make path failover of shadowset members work properly.

Completion of QIOserver will mean that at least testing with QIOserver-served paths will be needed, and some QIOserver code may need to be merged into the server defined herein. Also subsequently, the extended device naming project will need to use some of the routines in the server for its cluster name arbitration. This will probably be most efficient with that name arbitration loop merged into the one needed for this project to find duplicate paths. (The routines will be set up to facilitate this.) Thus these two projects can avoid some duplication of work.

The estimated number of defects in the code is:

- 5500 lines of new code, at one defect per 25 lines: 220 defects.

The total is 220 defects. These figures are quite crude.

## 9 Project Risks

The chief risks in this project are schedule ones. Unforeseen problems, particularly in the interoperation with hardware, could cause delays, and other problems could arise. Switching paths has run in "test jig" setups, validating the basic approach, but changes to any of mount verify, fast path code, MSCP, or issues which may arise with QIOserver could add complexity not currently foreseen. Furthermore the coding speed and size estimates are rough. (It is believed that the size estimate may be high but that the coding speed estimate may also be high; these may cancel one another out.)

The intent is to measure the size of code each day or week that has been completed and compare with expected total end size as reported here to track progress. The presumption will be that moving or slightly reworking existing code can proceed very rapidly; it already presumes "n" paths in a number of places (where the assumption is that "n" is small, i.e., less than 8 or so), and many of the needed cleanups are already identified in comments. Once

this is done, the tracking will be able to be against "new" code, to permit early discovery of schedule problems should any develop.

## 10 Project Deliverables

1. Design documents for the completed work
2. Switching driver
3. Utility program to switch paths manually and to set up paths to be switched manually.
4. Complete server process code able to handle HSZ50 messages and to do duplicate detection and matching for devices on HSZ or with WW IDs that can be obtained, and able to do cluster checking as needed at boot time to ensure naming is consistent.
5. DKdriver modifications to support these functions
6. IO exec definition changes and (small) code modifications to support interceptions.