ØMQ for OpenVMS I64 and Alpha

Brett Cameron (brett.cameron@hp.com), John Apps (john.apps@hp.com) May 2010

1. Introduction

Thank your for your interest in this port of ØMQ (ZeroMQ) to OpenVMS. The current release of ØMQ for OpenVMS is based on the ØMQ 2.0.6 distribution.

ØMQ (http://www.zeromq.org) is a messaging system that aims to address many of the problems of more traditional enterprise messaging solutions such as complexity and bloat. ØMQ tackles these issues by taking a different approach. Instead of inventing new APIs and complex wire protocols, ØMQ extends the socket API, eliminating the learning curve and allowing a network programmer to master it in just a few of hours.

The wire protocols employed by ØMQ are deliberately simplistic, even trivial, and performance of ØMQ matches and often exceeds that of raw sockets. Speeds of over 8 million messages per second with a latency of some 12µs have been measured using standard Intel hardware and Linux together with Infiniband.

This OpenVMS port of ØMQ includes almost all ØMQ. The port presently does not provide support for reliable multicast (via OpenPGM) and does not support direct (non-TCP/IP) inter-process communication. It is anticipated that these deficiencies will be addressed in future releases.

2. Acknowledgements

The authors would like to acknowledge the support and assistance of the ØMQ team in the creation of this release.

3. Requirements

The kit you are receiving has been compiled and built using the operating system and compiler versions listed below. While it is highly likely that you will have no problems installing and using the kit on systems running higher product versions of the products listed, we cannot say for sure that you will be so lucky if your system is running older versions. If you require a kit for a different configuration, we will do what we can to oblige. Note that the UnZip utility is required in order to unpack the ZIP kit.

- OpenVMS 8.3 (I64 or Alpha) or higher
- HP TCP/IP Services V5.6 or higher

  It has not been verified whether the kit works with the MultiNet TCP/IP stack, but there is a good chance that it will.
- C compiler - HP C V7.1-015 or higher
- C++ compiler - HP C++ V7.3-009 or higher

  Note that the C++ is required in order to build ØMQ applications. Specifically, the `CXXLINK` utility must be used on OpenVMS Alpha in order to link applications; on OpenVMS Integrity, the standard `LINK` command may be used.
- UnZip 5.42 (or similar) for OpenVMS (required to unpack the ZIP kit and can be found on the OpenVMS Freeware CD)

In addition to the above requirements, it is assumed that the reader has a good knowledge of OpenVMS and of software development in the OpenVMS environment.

4. Recommended reading

Before getting too carried away, be sure to read (or at least browse) the very comprehensive documentation on the ØMQ web site (http://www.zeromq.org). In addition to programming guides, there are whitepapers and assorted other documents that provide plenty of useful information on how ØMQ can be used.

5. Contents of the kit

The kit is currently provided as a ZIP file (created using Zip 2.3 for OpenVMS), which allows individual users to install the software under their own accounts, if so desired.

The following file listing represents the output from the `unzip -l` command run against the ZIP file kit on OpenVMS Alpha (the listing on OpenVMS I64 would be very similar).

```
Archive:  CCOX05$DKB100:[BIGGLES]ZEROMQ-206-VMS-AXP.ZIP;1
  Length     Date   Time    Name
 --------    ----   ----    ----
        0  05-10-10 19:55   zmq/bin/
     9603  05-10-10 19:47   zmq/copying.lesser
    35149  05-11-10 15:11   zmq/copying.txt
        0  05-10-10 19:55   zmq/doc/
        0  05-10-10 19:55   zmq/examples/
        0  05-10-10 19:55   zmq/include/
        0  05-10-10 19:55   zmq/lib/
      535  05-10-10 21:24   zmq/zmq$startup.com
  2060288  05-11-10 14:37   zmq/bin/zmq$shr.exe
   185856  05-11-10 14:37   zmq/bin/zmq_forwarder.exe
   195072  05-11-10 14:38   zmq/bin/zmq_queue.exe
   185344  05-11-10 14:38   zmq/bin/zmq_streamer.exe
    18010  05-10-10 19:50   zmq/doc/zmq.html
    11275  05-10-10 19:50   zmq/doc/zmq_bind.html
     9018  05-10-10 19:50   zmq/doc/zmq_close.html
    11374  05-10-10 19:50   zmq/doc/zmq_connect.html
    16343  05-10-10 19:50   zmq/doc/zmq_cpp.html
    14749  05-10-10 19:50   zmq/doc/zmq_epgm.html
     8164  05-10-10 19:50   zmq/doc/zmq_forwarder.html
     9870  05-10-10 19:50   zmq/doc/zmq_init.html
    11170  05-10-10 19:50   zmq/doc/zmq_inproc.html
    10753  05-10-10 19:50   zmq/doc/zmq_ipc.html
     9554  05-10-10 19:50   zmq/doc/zmq_msg_close.html
     9847  05-10-10 19:50   zmq/doc/zmq_msg_copy.html
     9066  05-10-10 19:50   zmq/doc/zmq_msg_data.html
     9554  05-10-10 19:50   zmq/doc/zmq_msg_init.html
    10281  05-10-10 19:50   zmq/doc/zmq_msg_init_data.html
     9623  05-10-10 19:50   zmq/doc/zmq_msg_init_size.html
     9453  05-10-10 19:50   zmq/doc/zmq_msg_move.html
     9078  05-10-10 19:50   zmq/doc/zmq_msg_size.html
    14749  05-10-10 19:50   zmq/doc/zmq_pgm.html
    14317  05-10-10 19:50   zmq/doc/zmq_poll.html
     8148  05-10-10 19:50   zmq/doc/zmq_queue.html
    10726  05-10-10 19:50   zmq/doc/zmq_recv.html
    10867  05-10-10 19:50   zmq/doc/zmq_send.html
    22673  05-10-10 19:50   zmq/doc/zmq_setsockopt.html
    14495  05-10-10 19:50   zmq/doc/zmq_socket.html
     8163  05-10-10 19:50   zmq/doc/zmq_streamer.html
     9145  05-10-10 19:50   zmq/doc/zmq_strerror.html
    14497  05-10-10 19:50   zmq/doc/zmq_tcp.html
     8813  05-10-10 19:50   zmq/doc/zmq_term.html
     9173  05-10-10 19:50   zmq/doc/zmq_version.html
     1084  05-11-10 13:39   zmq/examples/build.com
     2776  05-10-10 19:51   zmq/examples/local_lat.c
     3759  05-10-10 19:51   zmq/examples/local_thr.c
     3255  05-10-10 19:51   zmq/examples/remote_lat.c
     2657  05-10-10 19:51   zmq/examples/remote_thr.c
     8365  05-10-10 19:51   zmq/include/zmq.h
     6264  05-10-10 19:51   zmq/include/zmq.hpp
  5551616  05-11-10 14:37   zmq/lib/libzmq.olb
 --------                   -------
  8594571                   50 files
```

5.1. *Installing the kit*

Unpacking and installing the ZIP file kit is very straightforward. After copying the supplied ZIP file (`ZEROMQ-206-VMS-AXP.ZIP` for OpenVMS Alpha or `ZEROMQ-206-VMS-I64.ZIP` for OpenVMS I64) to a suitable location, unpack the contents of the relevant ZIP file using the `unzip` command.

For example, for the Alpha kit, you would enter the following command:

```
$ unzip ZEROMQ-206-VMS-AXP.ZIP
```

After unpacking the kit, you will have a `[.zmq]` directory below your current directory that contains files listed above. Note that you can install the software onto either an ODS2 or an ODS5-formatted disk.

5.1.1. Post-installation steps

To complete the installation it is necessary to define the concealed logical name `ZMQ$ROOT` to point to your top-level `[.zmq]` directory and to define a logical name for the shareable image `ZMQ$SHR.EXE`. You may optionally wish to install the shareable image.

A sample command procedure called `ZMQ$STARTUP.COM` that illustrates these logical name definitions and installation of the shareable image is provided in the top level `[.zmq]` directory (see listing below). You may wish to modify this file as required and include it in your system start-up procedure.

```
$ ! ZMQ$STARTUP.COM
$ !+
$ ! 10-May-2010
$ ! Startup file for ZeroMQ 2.0.6 on OpenVMS
$ !-
$
$ set noon
$ !
$ if f$trnlnm("zmq$root","lnm$system_table",,,,) .eqs. ""
$ then
$    ! Modify definition for zmq$root as appropriate
$    !
$    define/sys/exec/tran=(conc,term) zmq$root dorone$dka0:[zmq.]
$ endif
$
$
$ write sys$output "ZMQ-I-INSTALL, Installing ZMQ$SHR"
$ define/sys/exec zmq$shr zmq$root:[bin]zmq$shr.exe
$ install replace zmq$shr/header_resident/open/shared
$
$ write sys$output "ZMQ-I-DONE, Startup complete"
$ !
$ exit
```

Note that in the above example the `ZMQ$ROOT` logical name is defined in the system table. It is possible to define the logical name at a lower level (such as `GROUP` level) in order that multiple users may run 0MQ in their own process space.

5.2. *Privileges and quotas*

Generally speaking there are no special quota or privilege requirements for applications developed using ØMQ, although a high `BYTLM` is recommended, and `SYSPRV`, `BYPASS`, or `OPER` privilege will be required if ØMQ processes need to utilise privileged ports (ports below 1024).

The authors typically operate (on OpenVMS IA64) with quota settings similar to the following, which should be more than adequate for most purposes:

```
Maxjobs:          0  Fillm:        256  Bytlm:        128000
Maxacctjobs:      0  Shrfillm:       0  Pbytlm:            0
Maxdetach:        0  BIOlm:        150  JTquota:        4096
Prclm:           50  DIOlm:        150  WSdef:          4096
```

```
Prio:              4  ASTlm:        300  WSquo:         8192
Queprio:           4  TQElm:        100  WSextent:     16384
CPU:          (none)  Enqlm:       4000  Pgflquo:     256000
```

6. Sample applications

The `zmq$root:[samples]` directory contains two simple examples that can be used to measure latency[1] and throughput. These examples can be compiled and linked using the provided build procedure (`build.com`). Once built, these programs are simple to run. For example, for the latency example, to measure the latency for a 16-bytes message using a sample size of 10000 messages, on one machine we could run the following command:

```
$ mcr []local_lat.exe "tcp://16.156.32.107:5555" 16 10000
```

And on another machine we would enter the following command:

```
$ mcr []remote_lat.exe "tcp://16.156.32.107:5555" 16 10000
```

When the run completes, `remote_lat.exe` will display the results as follows (the latency will vary, depending on your specific hardware, operating system, and network configuration):

```
message size: 16 [B]
roundtrip count: 10000
average latency: 296.000 [us]
```

In addition to these examples, additional example code may be found on the ØMQ web site. For example, see http://www.zeromq.org/docs:cookbook.

7. What's missing?

As noted previously, the bulk of the ØMQ functionality is present, and it should be possible to do much of what is described on the ØMQ web site. Support for reliable multicast (via OpenPGM) is not currently supported, nor is non-TCP/IP IPC.

Under UNIX/Linux, ØMQ leverages UNIX sockets to provide an efficient non-TCP/IP IPC mechanism; however such facilities are not available on OpenVMS, and the authors are considering various other options.

ØMQ also supports a range of language bindings, including scripting languages such as Ruby and Lua, and 3GL languages such as Ada. The authors are working to provide similar options on OpenVMS, including a version of the ØMQ API that can be readily called from any OpenVMS 3GL, such as COBOL, FORTRAN, and so on.

Appendix

Tools

Over time the authors will update this document with tools they have found useful in the course of their work. The authors welcome any suggestions and will include them in future versions of the document.

- Zip and UnZip for OpenVMS may be obtained from the OpenVMS freeware CD at http://h71000.www7.hp.com/openvms/freeware/.

[1] Latencies on the order of 200µs on an rx2620 have been achieved on OpenVMS Integrity. Latencies on OpenVMS Alpha may be 4 or 5 times greater, depending on the age and configuration of the hardware and software in question.