GSI OPERATION SOFTWARE: MIGRATION FROM OpenVMS TO Linux

Ralf Huhmann, Günther Fröhlich, Susanne Jülicher, V. R.W Schaa, GSI, Darmstadt, Germany

Abstract

The current operation software at GSI, controlling the linac, beam transfer lines, synchrotron and storage ring, has been developed over a period of more than two decades using OpenVMS on Alpha-Workstations. The GSI accelerator facilities will serve as an injector chain for the new FAIR accelerator complex for which a control system is currently developed. To enable reuse and integration of parts of the distributed GSI software system, in particular the linac operation software, within the FAIR control system, the corresponding software components must be migrated to Linux. Interoperability with FAIR controls applications is achieved by adding a generic middleware interface accessible from Java applications. For porting applications to Linux a set of libraries and tools has been developed covering the necessary OpenVMS system functionality. Currently, core applications and services are already ported or rewritten and functionally tested but not in operational usage. This paper presents the current status of the project and concepts for putting the migrated software into operation.

MISSION

Presently, the operation software at GSI runs on a cluster of DEC-Alpha[1] machines. The computers' OS is OpenVMS. The user interface is realized by X-Window and Motif based clients. There are some hardware display and control units, i.e. proprietary interfaced LED and LCD displays, knobs, analog gauges, etc. The software modules for operation of the linear accelerator (LINAC), the transfer lines, and common services are mainly written in Fortran77. The central software components for operation of the synchrotron (SIS) and the storage ring (ESR) are written in Pascal. All programming languages use specific DEC extensions and a lot of OpenVMS specific system calls. The X11 and Motif implementations embed the X11 events into the OpenVMS system's event scheme. We aim to get rid of OpenVMS but to reuse the main parts of the GSI operational software's source code base by porting the software to Linux. Additionally, the migration shall enable interoperability with current or future developments, e.g. in Java on Linux. It showed up that most of the operation applications written in Fortran77 could be ported to Linux with minimal modification effort utilizing the techniques, libraries, and tools described in [1]. Some other Fortran components, especially system oriented services were rewritten in C/C++/Java on Linux. The components for SIS and ESR written in Pascal could not be ported, mostly because no suitable compiler is available which handles the DEC language extensions.

[1] Alpha is today a brand of Hewlett-Packard.

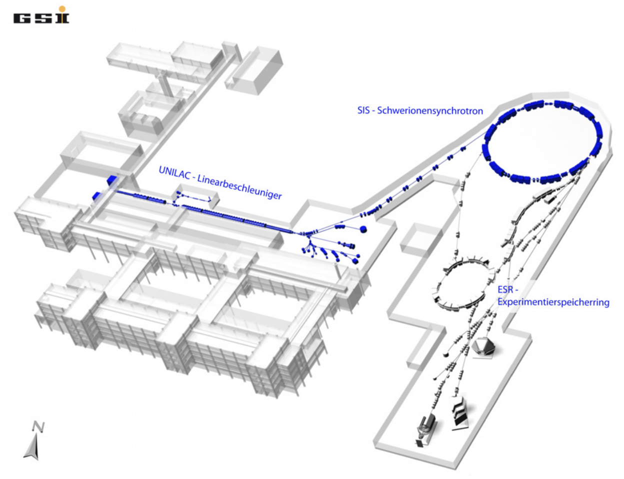

SIS - Schwerionensynchrotron
UNILAC - Linearbeschleuniger
ESR - Experimentierspeicherring
N

Figure 1: GSI accelerator overview.

ISLANDS

Fortunately, software components for operation of the SIS and ESR on one hand, and the LINAC and transfer lines on the other hand, are only minimally coupled (Fig. 2). Therefore, in step 1 a gateway mechanism was developed to bridge these islands. The SIS and ESR operation software will remain unchanged on OpenVMS and will be excluded from migration in step 1. In a second step those components will completely be replaced by the LSA framework [2] with new Java applications.

New operation applications for the LINAC and transfer lines are developed in Java on Linux and can be bridged with the island of ported Fortran software.

BRIDGES

Fortran@Linux – Fortran@OpenVMS

On OpenVMS, the coupling between SIS/ESR operation software and the LINAC software is realized by a service which sends binary messages from one application to another. It utilizes VMS mailboxes and raw ethernet communication. Actually, for the ported applications on Linux this API was implemented utilizing peer to peer TCP/IP communication and a nameservice (Fig. 4 shows a GUI monitoring the nameservice) to resolve the IP-address and IP-port by the application name. In order to connect from VMS to this framework a proxyservice was developed on

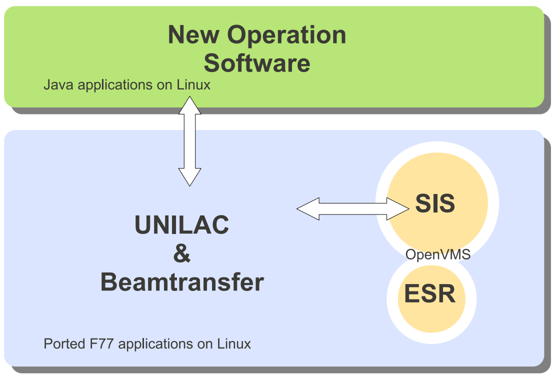


Figure 2: Groups of operation software components in step 1 of migration.

Linux. In Fig. 3 you see as an example two OpenVMS applications `V1` and `V3` which connect via TCP/IP to a service on Linux acting as a proxy for `V1` and `V3` and registering those in the nameservice with the proxy's address and port. Messages from `V1` or `V3` to `L1` or `L2` on Linux are sent to

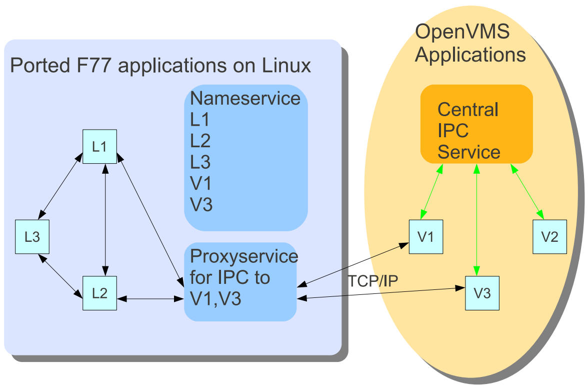
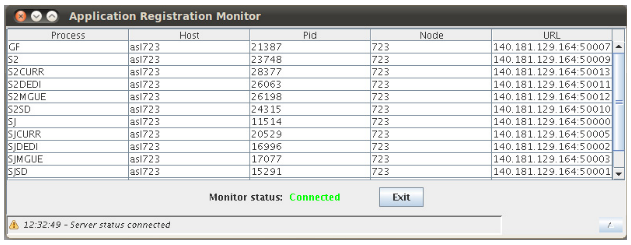


Figure 3: Binary data exchange between Linux and OpenVMS.

the proxy which transmits the packet to its destination application using the name service resolution on Linux. Vice versa, a message from `L1` or `L2` to `V1` or `V3` is received by the proxy since it registered `V1` and `V3` with the proxy's address and port in the nameservice. The proxy forwards the message to OpenVMS via the corresponding TCP/IP socket connection.

Fortran@Linux – Java@Linux

By the *uv*-architecture (Fig. 5) which is described in more detail in [1] an observer pattern is realized to connect Java applications to Fortran applications for data exchange. Fortran applications serve as publisher, Java applications subscribe to data objects. Requests, replies, and notifications are serialized over XML-streams.



Process	Host	Pid	Node	URL
GF	asl723	21387	723	140.181.129.164:50007
S2	asl723	23748	723	140.181.129.164:50009
S2CURR	asl723	28377	723	140.181.129.164:50013
S2DEDI	asl723	26063	723	140.181.129.164:50011
S2MGUE	asl723	26198	723	140.181.129.164:50012
S2SD	asl723	24315	723	140.181.129.164:50010
SJ	asl723	11514	723	140.181.129.164:50000
SJCURR	asl723	20529	723	140.181.129.164:50005
SJDEDI	asl723	16996	723	140.181.129.164:50002
SJMGUE	asl723	17077	723	140.181.129.164:50003
SJSD	asl723	15291	723	140.181.129.164:50001

Monitor status: Connected

12:32:49 - Server status connected

Figure 4: GUI monitor of nameservice.

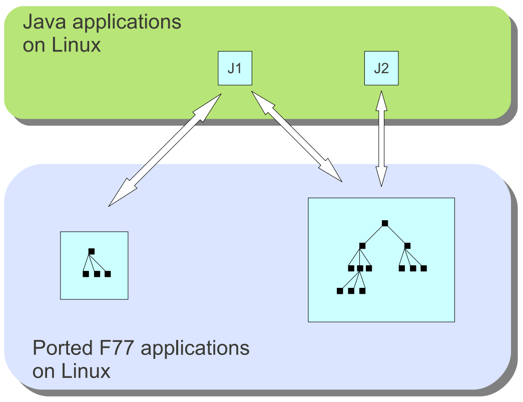


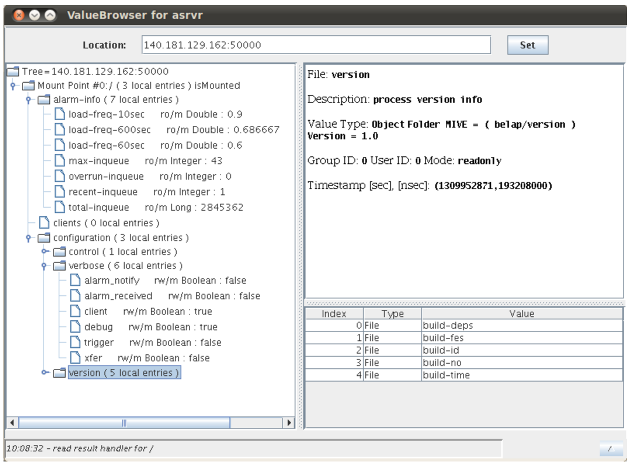
Figure 5: *uv*-architecture for data object exchange between Fortran and Java applications.

Publisher The simple server-API for the Fortran application allows to create and change at runtime a value tree-structure (like adding files and folders to a file-system), to change locally the numerical values of the leaf-nodes, and to put data objects (i.e. arbitrary *uv*-trees) in event-queues. Services like tree transfer of a read result, value changes by clients, sending value change notifications to clients, processing structure changes and subscriptions are hidden in the implementation of the API and are transparent for the application. Value changes initiated by a *uv*-client are notified to the application via callback.

Subscriber The client-API for the Java application is to read the value tree structure or subtree structures. It subscribes to folders for receiving structure changes or to leaf-nodes for receiving numerical value changes or to event-queues for receiving complex data objects. It allows setting values of leaf-nodes of an *uv*-server. Figure 6 shows as an example the value tree of a application seen by a generic value browser utilizing the Java client-API.

CURRENT STATUS

Currently (middle of 2011), the main operation applications for the LINAC and transfer lines are ported to Linux using the VMS system emulation libraries [1]. Some services have been reimplemented, e.g. a central alarm service, a service to launch and monitor applications on selected beam lines, communication modules for dedicated



```
140.181.129.162:50000

Tree=140.181.129.162:50000
  Mount Point #0:/ ( 3 local entries ) isMounted
    alarm-info ( 7 local entries )
      load-freq-10sec    ro/m Double : 0.9
      load-freq-600sec    ro/m Double : 0.686667
      load-freq-60sec    ro/m Double : 0.6
      max-inqueue    ro/m Integer : 43
      overrun-inqueue    ro/m Integer : 0
      recent-inqueue    ro/m Integer : 1
      total-inqueue    ro/m Long : 2845362
    clients ( 0 local entries )
    configuration ( 3 local entries )
      control ( 1 local entries )
      verbose ( 6 local entries )
        alarm_notify    rw/m Boolean : false
        alarm_received    rw/m Boolean : false
        client    rw/m Boolean : true
        debug    rw/m Boolean : true
        trigger    rw/m Boolean : false
        xfer    rw/m Boolean : false
      version ( 5 local entries )

File: version

Description: process version info

Value Type: Object Folder MIVE = ( belap/version )
Version = 1.0

Group ID: 0 User ID: 0 Mode: readonly

Timestamp [sec], [nsec]: (1309952871,193208000)

10:08:32 - read result handler for /
```

Index	Type	Value
0	File	build-deps
1	File	build-fes
2	File	build-id
3	File	build-no
4	File	build-time

Figure 6: Value tree browser.

hardware control units in the main control room, and a library for accelerator device access. A framework to build the complete software stack from a tagged subversion repository was developed. The above mentioned services and APIs to bridge to remaining VMS software and to new Java applications are implemented and tested.

MIGRATION PATH

Still missing are full integration tests of all operation software components, a production run time environment on Linux, deployment to the production system, performance tests under production conditions, and control room tests without and with beam.

Currently, a test console is set up which will serve as a test site outside the main control room for integration tests and run time environment. In order to switch the operation software to Linux it will be necessary to perform several temporary integration tests in the main control room (Fig. 7), first without beam, later with beam. Hence, to sustain normal operation in the meantime, a procedure has to be defined to enable switching between operation on Linux and back to OpenVMS till the final quality standards are reached. This switching procedure must cover the access to the hardware control units of the consoles, and the interoperation with the left OpenVMS software components. Figure 8 shows the scheduled time scale of the remaining migration path.

Figure 7: Main Control Room at GSI.

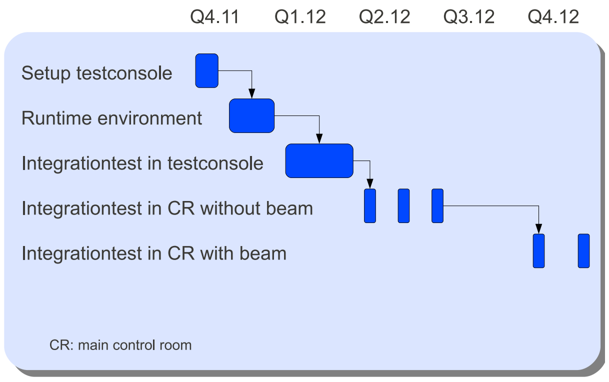


Figure 8: Schedule for migration of operation software.

CONCLUSION

Now, the feasibility studies and most of the implementation, porting, and unit testing has been completed. However, integration and launching in production environment demand further efforts.

REFERENCES

[1] R. Huhmann, G.Fröhlich, S. Jülicher, V.RW Schaa, "GSI Operating Software Migration Openvms to Linux", Proceedings of PCaPAC08, Ljubljana, Slovenia, MOX02, p. 4.

[2] J. Fitzek, R. Mueller, D. Ondreka, GSI, Darmstadt, Germany, "Settings Management within the FAIR Control System Based on the CERN LSA Framework", Proceedings of PCaPAC 2010, Saskatoon, Saskatchewan, Canada, WEPL008, p. 63.