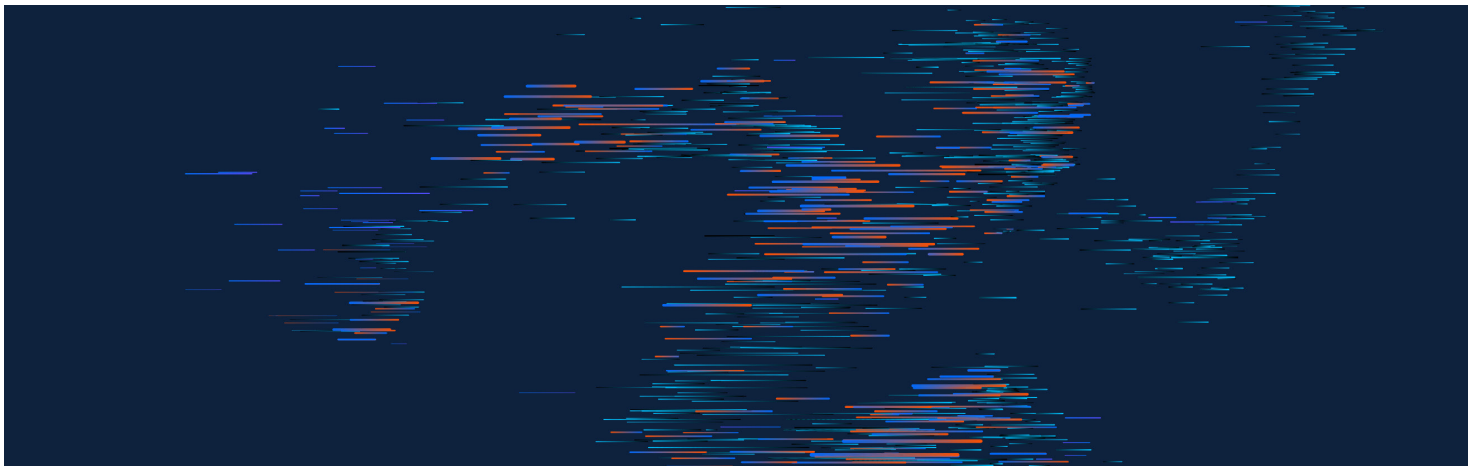
The Software Defined Mainframe – Leveraging "the Power of Modern"



Want to learn more and view live demo presentations by LzLabs?
Attend our upcoming event at the Gartner Symposium ITXPO – November 5-9, 2017 in Barcelona, Spain. The world's most important gathering of CIO's and Senior IT Executives.

The Pressure on Mainframe Users

Users of legacy mainframes are suffering under the weight of three very different, but ultimately related pressures. First, the cost of running applications on these legacy mainframes is not competitive with applications deployed on other hardware architectures such as x86 and cloud-based platforms. In addition to the high cost of legacy hardware, the cost of subsystems and third-party tools can dramatically increase the cost of running applications on a mainframe system.

Second, there are virtually no new IT professionals being trained to maintain and enhance legacy mainframe applications. As skilled legacy mainframe technicians retire, organizations increasingly face critical skill gaps in their struggle to support mission-critical applications that reside on legacy mainframes. Third, the mainframe application environment has been stubbornly resistant to modern development trends, making agile and responsive development difficult for mainframe resident applications.

Solutions

Over the years several methods have emerged to liberate users of legacy mainframes from these issues. The most popular, which will not be covered in detail here, is recompiling source-code and converting data to more modern formats. This method is only viable if the source-code is reliably available for all programs within an applications suite, and if the organization can shoulder the considerable testing burden relating to data format changes.

Another method involves migrating the applications, without rewriting or recompilation, to the target environment. Obviously, such a model requires a more complex form of rehosting infrastructure to enable the movement to occur so seamlessly. Two options exist for this model:

1) Operating System Rehosting and;

2) Application Workload Rehosting

Solutions (Cont'd)

The first option entails migrating not only the applications, but also the operating system and other required subsystems, to the target environment. In the second, the migration is focused on only the applications and data.

Operating System Rehosting

Operating System Rehosting solutions typically employ a low-level software program that emulates the behaviour of one computer hardware architecture on a different, target hardware architecture. Crucially, these rehosting solutions enable an operating system, written to run on one type of hardware, to run on a different type of hardware. To work properly, such solutions must address low-level hardware concepts as machine-code execution, interrupt handling, paging etc. The rehosting software program itself is installed on an underlying operating system such as Linux, that runs on the target hardware platform.

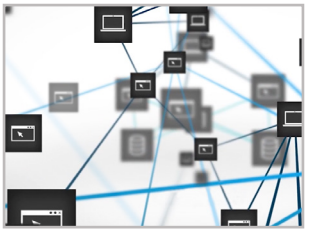
By working at such a fundamental level these Operating Systems Rehosting solutions eliminate many of the system compatibility issues between different architectures. User applications interact with the legacy operating system in the same way as before, since the legacy operating system is also running in the rehosted environment. Indeed, in these rehosting solutions the entire software stack of the legacy system is required to execute applications on the new hardware architecture.

Simulating the hardware of the source platform requires the use of additional instructions which add overhead to the application. In addition, certain legacy system behaviours such as paging are performed not only by the rehosted operating system, but also by the operating system of the target platform. This duplication of system overhead can be significant in large systems, often negating any price-performance benefit offered by the new platform. Operating System Rehosting solutions can also introduce timing problems since processes running in the rehosted environment may operate at different speeds than they did when running on the legacy platform. In highly parallel systems such low-level timing behaviours can result in unpredictable and difficult to diagnose problems that show-up intermittently.

Perhaps most importantly Operating Systems Rehosting fails to solve any of the three problems identified above. Software costs typically account for a substantial portion of the total expense of a legacy mainframe environment. But software costs are not reduced in a solution that requires migrating the entire legacy software stack. And the organization faces the added challenge of licensing software for the new platform. Staff and skill issues remain since the same skills with legacy operating systems, subsystems, and procedures are needed, yet remain in short supply. And agile development remains out of reach for legacy applications that find themselves embedded in an increasingly complex software stack.

It's all About the Applications – Application Workload Rehosting

When looking at legacy mainframe workload rehosting options, it's important to maintain focus on what needs to be preserved. ***The only important consideration is to ensure the applications themselves continue to operate reliably and with same outcomes.*** The LzLabs Software Defined Mainframe® design objective is to enable the applications to be rehosted on x86 without source-code changes or recompilation; yet with behavioural equivalence.



The Software Defined Mainframe – A Unique Solution to the Problem

A Software Defined Mainframe solves the legacy migration problem by presenting a platform that runs legacy workloads without the need for a legacy operating system or subsystems. Because no legacy system software runs on the operating platform, the Software Defined Mainframe puts the focus where it belongs, on the **application programs**. A customer application, written in COBOL, PL/I or Assembler source code will, at some point in its history, have been compiled into an object module and linked into what is known as a Load Module. This Load Module contains the machine code representation of the customer application, plus references to various application, transactional and DBMS subsystems as well as to the language runtime of the legacy mainframe.

The LzLabs Software Defined Mainframe begins by exporting a load module. During the export process references to legacy mainframe subsystems or to the language runtime are replaced with references to APIs of the Software Defined Mainframe. This export process produces a slightly modified mainframe object module, and the new APIs provide equivalent behavior to the legacy mainframe variants they replace. This functional compatibility ensures that the rehosted program will produce the same results it did when running in the legacy application environment.

Next, as the object module is imported into the Software Defined Mainframe, LzLabs performs its patented Dynamic Instruction-Set Architecture Translation (DIT) process. This DIT step transforms the mainframe object module into an x86 object module. Such translation involves much more than simply converting one instruction set into another, memory management must be preserved plus a range of other considerations taken to ensure functional compatibility. But, the end result is a x86 program that behaves exactly like the original mainframe application program. Since the remainder of its stack runs natively in the x86 environment, the Software Defined Mainframe avoids the technical shortcomings and increased overhead of Operating System Rehosting.

Finally, one of the benefits of the LzLabs approach is the preservation of data integrity. Since the application programs are not modified in any way, the SDM must preserve the original legacy mainframe encoding mechanism (EBCDIC), data types and data navigation functions. The re-hosted load module expects the same fields, from the same storage architecture in the same encoding structure, and the SDM ensures the behaviour of the legacy data is equivalent. This is a key feature of the SDM and provides reduced cost and risk when migrating customer applications. ■

For a direct glimpse at LzLabs software solutions in action, view this video where Christian Wehrli, Head of Post Sales at LzLabs, demonstrates how to move a transactional mainframe application to Microsoft Azure in matter of minutes.

About LzLabs

LzLabs GmbH is a software company that develops innovative solutions for enterprise computing customers including the **LzLabs Software Defined Mainframe®**. The LzLabs managed software container provides enterprises with a viable way to migrate applications from mainframes onto Red Hat Linux computers or into cloud environments such as Microsoft Azure. The company was founded in 2011 and is headquartered in Zürich, Switzerland. Learn more at **LzLabs.com**

Contact Us

LinkedIn: LzLabs GmbH
Twitter: @LzLabsGmbH

info@lzlabs.com

LzLabs GmbH
Richtiarkade 16
CH-8304 Wallisellen,
Switzerland