



WebSphere®
Application Server z/OS

Modernizing at High Speed

At a Glance

When ALCS and WAS z/OS are co-located on the same LPAR, it opens up the possibility of using the WAS z/OS Optimized Local Adapters, or "OLA." OLA is a very fast cross-memory exchange mechanism.

In this brochure we will spell out the value of that combination to you and your business.

Airline Control System (ALCS)

Airline Control System (ALCS) is a proven workhorse of the travel and financial world. Like its close cousin TPF, it's able to achieve very high rates of throughput by focusing on streamlined design and a minimum of overhead. Unlike TPF, which is its own operating system, ALCS uses z/OS as its operating system.

WebSphere Application Server for z/OS

WebSphere Application Server z/OS is IBM's flagship open standard Java EE runtime platform. It provides the ability to modernize existing data subsystems with capabilities such as:

- Encryption (with hardware assist)
- Access methods such as HTTP and web services SOAP
- Robust development tooling support

Optimized Local Adapters: "OLA"

A feature made available in WAS z/OS V7.0.0.4 is the Optimized Local Adapter (OLA) support. OLA is a high speed *bi-directional* exchange mechanism between Java programs in WAS z/OS and non-Java programs outside WAS z/OS.

Programs outside WAS z/OS include CICS, IMS, Batch and ALCS.

WAS z/OS + OLA + ALCS

ALCS by its design has limited user access capability. There is no GUI interface, no easy way to host web applications, no easy way to host a web services interface.

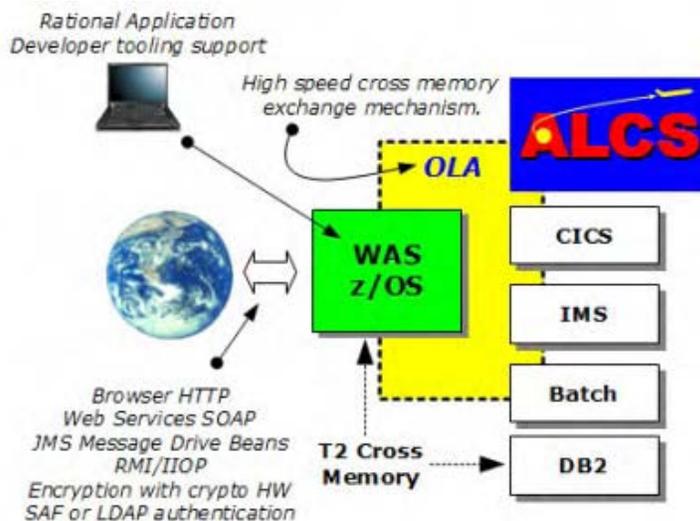
ALCS is what it is: a very fast transaction processing facility. To accomplish that goal ALCS keeps a tight focus on doing only what it's designed to do, and doing that with as little overhead as possible. ALCS does not try to be all things to all people.

WebSphere Application Server z/OS, on the other hand, is by its design a functionally rich open standard application serving environment. Browser and web service interfaces are built in features. Connectivity with a wide variety of systems such as CICS, IMS and DB2 is part of its design. It has a powerful transaction management system embedded inside.

In short, WebSphere Application Server z/OS makes a wonderful "front-end" to all those systems as well as ALCS. Start to think of WAS z/OS as a functionally robust and modern "glue" system for ALCS and the other subsystems typically used on z/OS.

And with WAS z/OS and OLA the connectivity is cross memory and very fast. WAS z/OS and OLA will feed ALCS quickly and allow ALCS to do what it does best: process requests rapidly.

The following picture emerges:



WAS z/OS and OLA -- Front-End Modernizer

WAS z/OS Java and zAAP Processors

System z and z/OS have what are known as "speciality engines," which are offload processors for work such as Java and certain DB2 work. The zAAP processor is designed for Java work.

These speciality engines have a much lower acquisition cost compared to general processors. But perhaps the most compelling cost attribute is this: *they are not used to count towards software licensing charges.*

In quick summary format, zAAPs ...

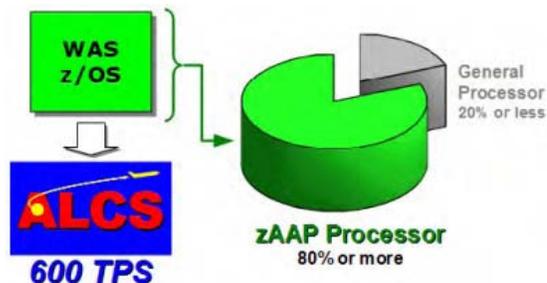
- Have a lower cost of acquisition profile
- Do not contribute to software MLC
- Makes running Java workloads on the mainframe competitive with distributed platforms.

The Java offload is entirely transparent to applications. It is a function of the Java SDK and the z/OS dispatcher. WebSphere Application Server for z/OS offloads to zAAPs.

zAAP specialty engines significantly reduce the cost of running Java on the mainframe.

OLA, ALCS, WAS z/OS Benchmark Tests

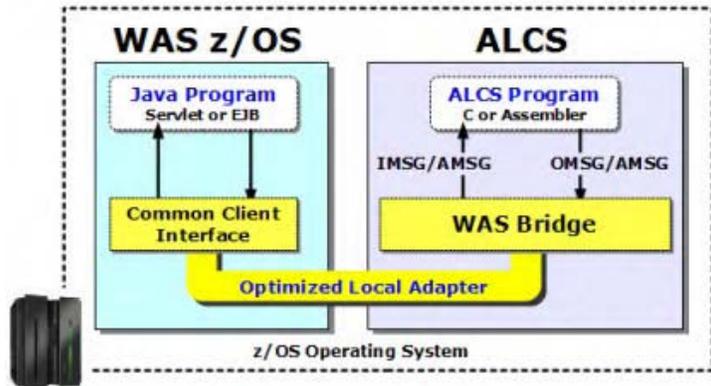
A benchmark study of WAS z/OS exercising 600 transactions per second to ALCS using OLA yielded a zAAP usage of approximately 80% of overall WAS z/OS CPU usage:



Result: high transaction rates with most CPU cycles offloaded to the lower-cost zAAP for a competitive execution model.

The ALCS WAS Bridge

A "WAS Bridge" function has been written that runs inside of ALCS. It provides the ALCS address space what's necessary to link up and use the OLA exchange mechanism:



Is OLA Transparent to ALCS Programs?

Yes. The WAS Bridge function hides the OLA implementation from the ALCS programs. To an ALCS program the Java program in WAS is a terminal or a printer. *ALCS programs remain unchanged.*

OLA does supply a set of native APIs that you may write to if you wish. That would provide you greater control over the WAS \leftrightarrow ALCS interaction. But that's not necessary when you first begin with OLA. The WAS Bridge is using the APIs at a lower level and hides that complexity from your program.

The WAS Bridge function makes OLA transparent to the ALCS program. Java programs in WAS appear as terminals or printers across a very fast exchange path.

What About Coding Java to OLA?

OLA comes with a JCA resource adapter (ola.rar) that implements the Common Client Interface (CCI) for your Java programs to use.

For calls from WAS to ALCS the Java program would use the methods of the CCI and supply parameters to indicate the OLA connection to use. For calls from ALCS to WAS the target EJB simply implements the `Execute()` and `ExecuteHome()` interfaces using the supplied OLA class files. The ALCS program appears to the Java program as a remote EJB.

Java on z/OS ... Different?

It's not "different" at the interface layer. Java is Java, and WAS is WAS ... across all supported platforms. That's the nature and benefit of agreed-to open standards.

But that does not mean that Java on z/OS and WAS z/OS are exactly the same as on other platforms. The Java JDK on z/OS is written to take full advantage of the z/OS and System z instruction set. And WAS z/OS performs deliberate exploitation of the z/OS platform as well. OLA is a good example of that exploitation.

Why WAS z/OS? Application layer commonality at the open standard specification line with *exploitation of specific z/OS platform services below the line.*

Other Benefits of WAS+ALCS Co-Location?

Our emphasis to this point has been on achieving the highest possible throughput by reducing overhead in the exchange between ALCS and Java in WAS. That's why we introduced OLA into the discussion.

Beyond OLA we have all the benefits of the z/OS platform -- Parallel Sysplex for scalability and availability; RRS for robust transactional management and rollback; WLM for system-wide workload management; SMF for activity reporting; and RMF for capacity planning. And of course the hardware design with a mean time between failures measured in *years*.

There's another benefit of co-locating WAS z/OS and ALCS: it provides a *structured and focused runtime environment*. That translates to better change control, easier problem determination, more deterministic tuning, and more structured backup, restore and disaster recovery.

Multi-tier distributed architectures have their place in a modern enterprise. But they do not serve all needs equally well. The mainframe environment is a proven business-reliable platform. Your ALCS programs are core to your business. Supporting Java programs should be treated as equally valuable to the business.

Whom to Contact for More Information

Jonathan Collins

Product Line Manager, TPF, z/TPF and ALCS
IBM Software Group, Application and Integration Middleware Software
Phone: 845-433-2612
E-mail: jlcollin@us.ibm.com

Nico J. Wigmans

IBM Netherlands
Phone: 31 - (0)205137838
E-mail: WIGMANS@nl.ibm.com

Reference Links

ALCS / OLA Links:

ibm.com/tpf/alcs/news.htm#Header_38

Includes the announcement letter and user's guide for the WAS Bridge support for ALCS.

WAS z/OS V7 Information Center

publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp

Then search on `cdat_ola.html`

OLA Techdocs

ibm.com/support/techdocs

Then search on WP101490

Includes a design planning guide and an API usage primer as well as this ALCS document.

OLA Redbook

www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4550.html?Open

Why WAS z/OS Techdoc

ibm.com/support/techdocs -- search on WP101532