Adobe® LiveCycle® Data Services 3

LiveCycle Data Services 3 is a scalable, high performance, J2EE™ based server designed to help Java™ enterprise developers rapidly develop Flash based applications.

Introduction

User experience, especially application performance, is critical to Rich Internet Applications (RIAs). To deliver a high level of scalability and performance, Adobe LiveCycle Data Services 3 uses an asynchronous messaging infrastructure at its core to deliver these key capabilities: Server Push, Remoting, and Data Management. This is further enhanced by the use of RTMP (real time messaging protocol) and NIO (New I/O) server. LiveCycle Data Services 3 has new features such as Edge Server, Adaptive Throttling, Reliable Communications, and Conflation that enable users to develop better performing applications.

This paper reviews the performance characteristics of the LiveCycle Data Services Messaging and Remoting infrastructure, but also provides an overview of the high-performance aspects of the Adobe Flash Platform itself, including the Flash Player and the binary AMF messaging protocol. The result is the fastest performing platform for real-time RIA applications available today.

This PDF contains (as attachments) the actual load testing example, test assets, and instructions needed to run all test scenarios discussed. Customers are encouraged to use these examples to reproduce results in their own environments or perform "what-if" scenarios testing with test parameters more representative of their own applications. Please refer to Appendix B for additional information.

LiveCycle Data Services 3 is extremely efficient in handling large number of messages with very low message latency and can push up to 400,000 messages to 500 concurrent clients with an average latency of less than 15 milliseconds on a single dual-CPU machine.

![Throughput vs. Latency](image)

Figure 1: LiveCycle Data Services can push up to 400,000 messages in under 15 ms
Performance in LiveCycle Data Services 3

The messaging infrastructure is core to LiveCycle Data Services and is used by Remoting and Data Management. Hence, the performance and scalability of the messaging infrastructure is reflective of the performance of the overall product.

The ability to push messages to the client improves the user experience of RIAs by providing access to data in real-time. LiveCycle Data Services provides such an infrastructure. Depending on the nature of application, businesses may be interested in different metrics. For example, with a currency trading application, the business will likely want the latency to be very low. On the other hand, latency may not matter to an inventory management application, they may be more interested in data size and bandwidth usage.

We have run specific scenarios to help you understand the impact of different factors on server performance. While we recommend that you run tests specific to your application to determine the actual performance, information in this paper should help you understand the server performance better.

It is recommended to review the Glossary to have a better understanding of the terminology used.

Impact of server throughput on average latency

Same message scenario

In this scenario, all the clients are subscribed to a destination and new messages from the message generator were published to this destination. This setup may be desirable if the client application does not need to selectively subscribe to messages published to a destination. If your application needs to selectively subscribe to messages, please refer to the unique message scenario.

This test was conducted with the following parameters:

- Message Size: 128 bytes
- Number of Clients: 500 (single destination, no subtopics were used)
- Two machine configuration – client and server on separate machines

LiveCycle Data Services was able to achieve a throughput of 400,000 messages/second with an average latency of 15 milliseconds.

![Throughput vs. Latency for Same Messages](image)

Figure 2: LiveCycle Data Services can push up to 400,000 messages in under 15 ms
Unique message scenario

In this scenario, the clients selectively subscribed to specific subtopics on a destination, and new messages from the message generator were published to these subtopics. No two clients were subscribed to the same subtopic ensuring that the messages were unique across clients.

This setup may be desirable if the client application needs to selectively subscribe to messages published to a destination.

This test was conducted with the following parameters:
- Message Size: 128 bytes
- Number of Clients: 500 (subtopics per client: 20)
- Two machine configuration – client and server on separate machines

LiveCycle Data Services was able to achieve a throughput of 150,000 messages/second with an average latency of 5ms. The mechanism to create a unique message per subtopic and route the message through a subtopic to the client adds overhead that reduces the maximum throughput achieved from 400,000 messages/second (same message scenario) to 150,000 messages/second.

![Throughput vs. Latency for Unique Messages](image)

Figure 3: LiveCycle Data Services can push 150,000 unique msg/sec with an average latency of 5ms
Impact of concurrent clients on average latency

This test demonstrates the impact on message latency as we vary the number of clients. This test is similar to the same message scenario, except that we increase number of clients and keep the send rate constant.

This test was test was conducted with the following parameters:

- Message Size: 128 bytes
- Send rate: 2 msg/sec
- Two machine configuration – client and server on separate machines

A single LiveCycle Data Services server can handle up to 40,000 concurrent users. Compared to the same message scenario, supporting large number of concurrent users results in a much higher average latency and lower send rate. This may be desirable and acceptable in applications where scalability is more important than average latency.

![Concurrent Clients vs. Average Latency](image)

Figure 4: LiveCycle Data Services can support up to 40,000 concurrent users with an average latency of 400ms
Impact of Message Size on average Latency

This test demonstrates the impact on message latency as we vary the size of message received by each client. This test is similar to same message scenario, except that the message size is increased.

This test was conducted with the following parameters:

- Number of Clients: 500
- Send rate: 2 msg/sec
- Two machine configuration – client and server on separate machines

Message size does impact the performance of the server. LiveCycle Data Services server was able to send messages of 100K and maintain a send rate of 2msg/sec (Server throughput: 1000 msg/sec). Clearly, larger messages increase the average latency and reduce the server throughput.

![Message Size vs. Average Latency](image)

Figure 5: Average latency increases as message size is increased
Impact of using NIO on scalability

LiveCycle Data Services supports NIO channels that can be used with HTTP and RTMP. NIO is a non blocking I/O channel and can handle more concurrent connections than a blocking I/O Servlet channel.

This test was conducted with the following parameters:

- Message Size: 128 bytes
- Send Rate: 1 msg/sec
- Java Heap Size: -Xmx1024m

NIO channels support 10x concurrent connections than the blocking Servlet channel. Applications that tend to have large numbers of concurrent, but intermittently active, clients can significantly scale better with NIO than the Servlet channel.

![Max. Concurrent Clients in NIO vs. Servlet Endpoint](image)

Figure 6: Using NIO, LiveCycle Data Services can support 10x concurrent connections than the blocking Servlet channel
Impact on latency using LiveCycle Data Services Edge Server

The LiveCycle Data Services Edge Server, installed in the DMZ zone, proxies requests from the Flex client to the LiveCycle Data Services server, installed in the application tier. The Edge Server supports NIO channels, and can proxy RTMP(s) and HTTP(s) requests.

Flex clients can access the LiveCycle Data Services server either directly or through the Edge Server. Accessing through the Edge Server does add latency, and this test measures the latency overhead.

The test was conducted with the following parameters:

- Message Size: 128 bytes
- Number of Clients: 500
- Send rate: 120 msg/sec per client (Same/Unique Message)
- Throughput: 60,000 msg/sec (500 clients * 120 msg/sec per client send rate)
- Channel: HTTP AMF Streaming
- One machine configuration – client, server and Edge Server on the same machine

The test shows that using an edge server adds about 0.4 milliseconds to the overall message latency.
Impact of Concurrent Clients on Remoting throughput

This test demonstrates the impact on Remoting throughput as we vary the number of concurrent clients with one and two LiveCycle Data Services server instances.

This test was conducted with the following parameters:

- Response Size: 512 bytes
- Test Duration: 120 seconds
- Channel: RTMP
- Number of LCDS instances: 2
- Two machine configuration – client and server on separate machines

The results show the LiveCycle Data Services server handling 30,000 concurrent Remoting clients.

![Concurrent Clients vs. Remoting Throughput](image)

Figure 8: Server can handle 30,000 concurrent Remoting clients
Impact of Response size on Remoting throughput

This test demonstrates the impact on Remoting throughput as we vary the response size.

This test was conducted with the following parameters:

- Number of clients: 500
- Channel: RTMP
- Two machine configuration – client and server on separate machines

A single LiveCycle Data Services server can handle response sizes of 200 KB per request for 500 concurrent clients.

Figure 9: Server can handle response sizes of 200 KB per request for 500 concurrent clients
Common questions

How many messages per second can Adobe Flash Player consume and with what latency?

To answer that question, Christophe Coenraets, Adobe Evangelist, has built a Performance Console and feed generator. The Performance Console allows you to configure the throughput of the server-side feed generator as well as the client subscription details, and then measure the overall performance and health of the system.

In this case, a single Flash Player 10.1 instance is able consume approximately 2,000 msg/sec on a single ThinkPad W500 laptop.
Below are some of the tests run with the AIR version of the Performance Console. The first eight columns provide the test parameters. The last three columns provide the actual results of the test.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Threads</th>
<th>Sleep (ms)</th>
<th>Target Server msps</th>
<th>Channel</th>
<th>Max Freq.</th>
<th>Frame Rate</th>
<th>Render</th>
<th>Total msgs</th>
<th>Actual Client msps</th>
<th>Avg Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>2</td>
<td>500.0</td>
<td>my-nio-ar</td>
<td>0</td>
<td>24</td>
<td>true</td>
<td>29,070</td>
<td>494.5</td>
<td>0.7</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>1,000.0</td>
<td>my-nio-ar</td>
<td>0</td>
<td>24</td>
<td>true</td>
<td>59,511</td>
<td>993.5</td>
<td>0.3</td>
</tr>
<tr>
<td>1000</td>
<td>3</td>
<td>2</td>
<td>1,500.0</td>
<td>my-rtmp</td>
<td>0</td>
<td>99</td>
<td>true</td>
<td>88,150</td>
<td>1,469.3</td>
<td>5.1</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>1</td>
<td>2,000.0</td>
<td>my-rtmp</td>
<td>0</td>
<td>150</td>
<td>true</td>
<td>116,138</td>
<td>1,935.6</td>
<td>3.4</td>
</tr>
</tbody>
</table>

While it will be rare that any application would require sending 2,000 messages per second to its clients, it is nonetheless good to know that the Adobe Flash Player is capable of handling this load level. Throttling messages using an appropriate LiveCycle Data Services messaging policy (Ignore, Buffer, Merge, or Custom) that is appropriate for your application is encouraged to avoid unnecessarily using resources and bandwidth.

Note that the default Flash Player frame rate must be increased to process extremely high message volumes when using the RTMP channel. For more details regarding this scenario and instructions to setup and run the test locally visit the related blog entry: [http://coenraets.org/blog/2010/04/performance-tuning-real-time-trader-desktop-with-flex-and-lcds/](http://coenraets.org/blog/2010/04/performance-tuning-real-time-trader-desktop-with-flex-and-lcds/)

How does the Adobe Flash/Flex and AMF combination perform versus alternative client and transport technologies?

James Ward, Adobe Evangelist, has built a Performance Console to demonstrate the performance of some mainstream client and transport technologies, including Flex and AMF.

In short, the combination of Flash, Flex and AMF is one of the fastest performing client and transport combinations available anywhere today, far outperforming most alternatives.

The application can be found and run here: [http://www.jamesward.com/census2/](http://www.jamesward.com/census2/)

Sample results from running the application are below, with Flex and AMF displayed as the first bar:
Conclusion
User experience, especially application performance and responsiveness, is critical to successful Rich Internet Applications. To deliver the highest level of scalability and performance, the use of Adobe LiveCycle Data Services 3, together with the Adobe Flash Platform provides the best performing and scalable end-to-end solution for Rich Internet Applications available anywhere today.

Performance and capacity planning depend on a number of factors that are unique to each application. This performance brief should help customers understand the performance of LiveCycle Data Services under specific test scenarios. Customers should be able to replicate these tests in their own environment.

For an accurate representation of performance and capacity planning, Adobe recommends that customers conduct performance testing that is tailored to their application.

Additional resources

Try LiveCycle Data Services
Download the software for free and see how you can streamline rich Internet application development.

Documentation
LiveCycle ES2 and LiveCycle ES2.5 Documentation

Application modeling plug-in
Download the LiveCycle Application Modeling plug-in and begin creating your own user interface.

Developer center
Adobe LiveCycle Data Services

Capabilities
Capabilities of Adobe LiveCycle Data Services ES2

Frequently asked questions
Frequently asked questions about Adobe LiveCycle Data Services ES2
Appendix A: Test environment
This section describes the test environment used during the benchmarking tests.

Hardware
Benchmarking tests used the following configuration:
  • System model: HP ProLiant DL380 G6
  • Processor: Dual Quad-Core Intel® Xeon® processor 5500 sequence
  • Memory: 32 GB

Network hardware platform
All benchmark testing used a switched Gigabit Ethernet network fabric configuration to connect the various hardware components. The test network was isolated from any outside traffic.

Operating system
Suse Enterprise Linux 10 SP2

Server socket optimizations
  • ulimit -n 204800 command was executed for Server socket optimization.
  • The following lines were added to the /etc/security/limits.conf file on both the client and server machines to increase the per-process file descriptor limits:

    soft      nofile 10240
    hard      nofile 30480

TCP Settings
  • TCP settings in /etc/sysctl.conf file:

    # Disable response to broadcasts.
    # You don’t want yourself becoming a Smurf amplifier.
    net.ipv4.icmp_echo_ignore_broadcasts = 1
    # enable route verification on all interfaces
    net.ipv4.conf.all.rp_filter = 1
    # enable ipV6 forwarding
    #net.ipv6.conf.all.forwarding = 1
    net.core.rmem_max = 16777216
    net.core.wmem_max = 16777216
    net.ipv4.tcp_rmem = 4096 87380 16777216
    net.ipv4.tcp_wmem = 4096 87380 16777216
    net.ipv4.tcp_no_metrics_save = 1
    net.ipv4.tcp Moderate_rbuf =1
    net.core.netdev_max_backlog = 2500

  • After the above changes, run the following commands:

    sysctl -p
    ifconfig eth0 txqueuelen 2000
Clock Synchronization

If multiple machines are used for latency testing, the clocks on the test machines should be in synch.

The time synchronization method used in our testing is NTP.

Java

- Apache Tomcat server 6.0.20
- Sun JRE version 1.6.0_18 (build 1.6.0_18-ea-b05)
- JRE settings for both Server and Load test tool:
  -Xms8192m -Xmx8192m -XX:ParallelGCThreads=8 -XX:GCTimeRatio=10

-Xms8192m -Xmx8192m

Allocating larger heap size gives more time to young objects to die out before minor collection starts. Otherwise, the objects get promoted to the old generation and get collected in full GC. By preventing objects from getting tenured to old generation, full GC time is minimized. Setting initial and max heap sizes to the same value prevents heap resizing, which eliminates the pause time caused by heap resizing.

-XX:ParallelGCThreads=8

Reduces the number of garbage collection threads. The default would be equal to the processor count, which would be unnecessarily high.

-XX:GCTimeRatio =10

A hint to the virtual machine that it’s desirable that not more than 1 / (1 + nnn) of the application execution time be spent in the collector.

For example -XX:GCTimeRatio=10 sets a goal of 10% of the total time for GC and throughput goal of 90%. That is, the application should get 10 times as much time as the collector.

By default the value is 99, meaning the application should get at least 99 times as much time as the collector. That is, the collector should run for not more than 1% of the total time. This was selected as a good choice for server applications. A value that is too high will cause the size of the heap to grow to its maximum.

Also, the larger the heap, larger GCTimeRatio means shorter pause time.

Appendix B: Reproduce steps

Contents of the Performance Brief

- Performance Brief
- Performance Brief Tests – Contains configuration and instructions for individual test cases.

How to reproduce performance brief scenarios

1. Setup Test environment
   - Ensure that the test environment is as specified in Appendix A
   - Unzip load-test-tool.zip (attached to this PDF portfolio)
   - Follow instructions in Load test tool readme document.

2. Run Tests
   - Go to Performance Brief Tests folder.
   - Follow instructions to run the desired test scenario in the {test scenario}.pdf
Appendix C: Glossary of terms

Send Rate
This is the rate at which the message generator is generating messages to send to the LiveCycle Data Services server. In other words, the send rate is also the incoming message rate to the LiveCycle Data Services server.

Server Throughput
Server Throughput is the outgoing message rate of the LiveCycle Data Services server. The server throughput may not be the same as the send rate. In some cases, the server can send an incoming message to multiple clients thus sending more messages out of the server than it receives.

Latency
Latency is measured from the time a message is generated to the time that a client receives the message.

\[
\text{Latency} = \text{Message Received Time} - \text{Message Created Time}
\]

Client receive rate
Client receive rate is the rate at which messages are received by a client

Same message scenario
This is the scenario where all clients subscribe to a destination (without subtopics) and messages are sent to the destination (without subtopics). In this case, every client receives every message.

Unique message scenario
This is the scenario where each client subscribes to its unique subtopic on the destination and each message is sent to a single subtopic. In this case, each message goes to a specific client only and not to all clients.

Remoting throughput
Remoting Throughput is the rate at which LiveCycle Data Services server returns a response to Remoting requests.