IBM MQ for z/OS V9.1.2

zHyperWrite
MQ Active Log Acceleration

Version 1.0 – March 2019

Tony Sharkey

IBM MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire

Notices

DISCLAIMERS
The performance data contained in this report was measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends upon the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS
The information set forth in this report could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

INTENDED AUDIENCE
This report is intended for architects, systems programmers, analysts and programmers wanting to understand the performance characteristics of *IBM MQ V9.1*. The information is not intended as the specification of any programming interface that is provided by IBM MQ. It is assumed that the reader is familiar with the concepts and operation of IBM MQ V9.1.

LOCAL AVAILABILITY
References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES
Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS AND SERVICE MARKS
The following terms used in this publication are trademarks of their respective companies in the United States, other countries or both:

- **IBM Corporation:** IBM

Other company, product, and service names may be trademarks or service marks of others.

EXPORT REGULATIONS
You agree to comply with all applicable export and import laws and regulations.

Introduction

IBM MQ for z/OS version 9.1.2 introduces support for zHyperWrite with MQ's active log data sets.

This paper will talk about improved active log performance, in particular with IBM® Metro Mirror and zHyperWrite.

IBM® Metro Mirror, previously known as Synchronous Peer to Peer Remote Copy (PPRC), is a synchronous replication solution between two storage subsystems, where write operations are completed on both the primary and secondary volumes before the write operation is considered to be complete.

The IBM MQ Knowledge Centre section on "Using MetroMirror with IBM MQ" discusses which types of IBM MQ data sets can be replicated using Metro Mirror.

Why does my log performance matter?

Improved MQ active log performance is important for a number of reasons:

- Allows more work to get done with the same hardware footprint
- Better able to handle workload spikes
- Reduces costs

Generally, there are a number of ways to make transactions run faster on System Z and z/OS:

1. Faster CPU
2. Software scaling, reducing contention, faster I/O
3. Faster I/O technologies such as zHPF, 16 Gb channels, zHyperWrite, etc
4. Run at lower utilisations, address Dispatcher Queuing Delays
5. Faster network such as SMC-R or SMC-D

When using high levels of persistent messaging, the rate at which MQ is able to process the workload is largely dependent on the rate at which the MQ logger task is able to complete its I/O. This rate is almost entirely dependent upon the rate at which the I/O subsystem is able to respond to the requests.

The following extract from MQ Statistics data, formatted using program MQSMF (supportpac MP1B), demonstrates the single logger task, which runs as an SRB, is 99.95% busy, of which 93.79% is waiting for I/O to complete:

```
Log write rate     331MB/s per copy
Logger I/O busy : 93.79%
Logger task busy: 99.95%
```

In this environment, one way to alleviate this constraint is to reduce the I/O times.

When running synchronous replication technologies such as Metro Mirror (PPRC), the I/O times can be significantly higher than when running without the datasets replicated. The zHyperWrite technology can significantly reduce the I/O times when replicating datasets.

As of MQ v912, we support zHyperWrite with MQ active logs. This offers benefits in a number of areas:

- Reduced I/O times by **up to 60%.**
- Reduced elapsed time for commit by **up to 60%,** which can reduce contention.
- Improved sustained log rate, allowing for each MQ queue manager to process **up to 2.4 times** the volume of workload.

Caveats:

- The benefits of zHyperWrite reduces as the distance of replication increases.
- Although improved log capacity can be achieved with zHyperWrite, increased SRB time in the MQ MSTR address space might be observed because the extra (Media Manager) I/O requests are processed under MSTR SRB when zHyperWrite is enabled.

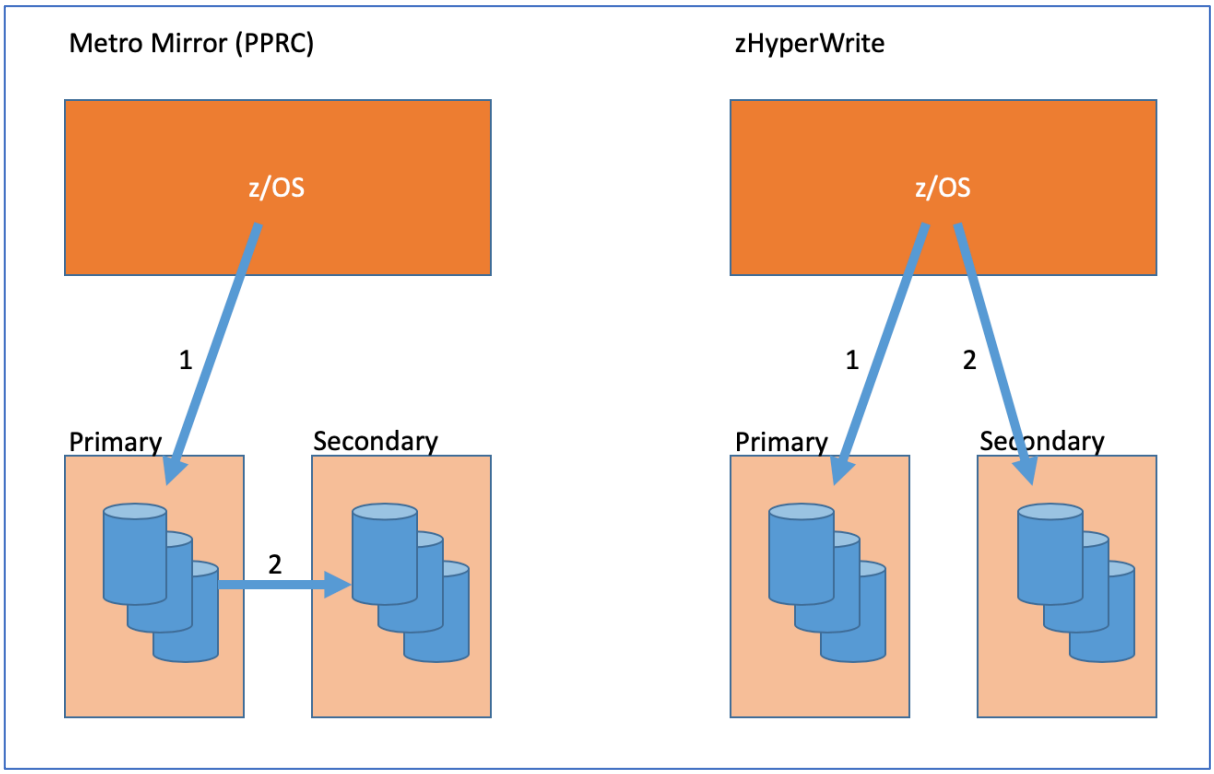
What is zHyperwrite?

The z/OS 2.3.0 Knowledge Centre describes zHyperWrite as:

"IBM zHyperWrite processing can be used by I/O drivers, such as Media Manager, for certain write I/O operations to perform software mirroring to peer-to-peer remote copy (PPRC) devices that are monitored for HyperSwap® processing (with GDPS® or TPC-R). IBM zHyperWrite data replication can be used to reduce latency in these HyperSwap environments. For maximum benefit, IBM zHyperWrite data replication should only be used when all synchronously mirrored relationships are managed by HyperSwap. Devices support IBM zHyperWrite data replication when the following conditions are true:

- The devices support IBM zHyperWrite data replication. Both the primary and secondary devices in a synchronous PPRC relationship must support this function.
- The devices in the synchronous PPRC relationship are managed by HyperSwap (either GDPS HyperSwap or TPC-R HyperSwap)."

A simplified view of the difference between PPRC and zHyperWrite is offered in the following diagram:



In the PPRC configuration the mirroring is driven from the primary control unit.

In the zHyperWrite configuration, the mirroring is requested once the I/O to the primary control unit is requested. This can reduce the time to initiate the mirrored request.

Measurement environment

The following measurements were run on a single LPAR of a 3-LPAR sysplex on a IBM z14 (3906-7A1). The LPAR had 3 dedicated general purpose processors, and for purposes of comparison is similar to a 3906-703.

A single MQ queue manager was created in 3 configurations, in each case with single active log and single archive log.

- Configuration 1: baseline – no mirroring.
- Configuration 2: PPRC – active logs mirrored using Metro Mirror.
- Configuration 3: zHyperWrite – active logs mirrored using Metro Mirror and zHyperWrite enabled at the queue manager.

The queue manager was configured with single active and archive logs as in a dual active and archive log environment we saw measurements impacted by site hardware limitations.

Tests run with dual active and archive logs are detailed in Appendix A along with information on what the impact on the performance due to our hardware limitations.

In our configuration, the mirrored DASD was at zero distance. Synchronously replicated DASD over a distance may see different results.

In each configuration, a put-commit/get-commit workload is run using a range of message sizes from 2KB to 4MB.

With regards to the disk subsystem we have a single frame DS8870 dedicated for performance testing, with dedicated links between the z14 and DS8870.

The PPRC configuration has 4 dedicated 16Gb Fibre channels for the PPRC-specific traffic. All non-PPRC specific traffic is spread across 4 dedicated 16Gb Fibre channels.

This means that all I/O is written to the non-PPRC specific channels except for the mirrored active logs in the PPRC configuration.

In the zHyperWrite configuration, both primary and secondary copies of each active log are written using the non-PPRC fibre channels.

In all configurations, zHPF (z High Performance FICON) was used.

Note that all I/O is over 16Gb Fibre channels.

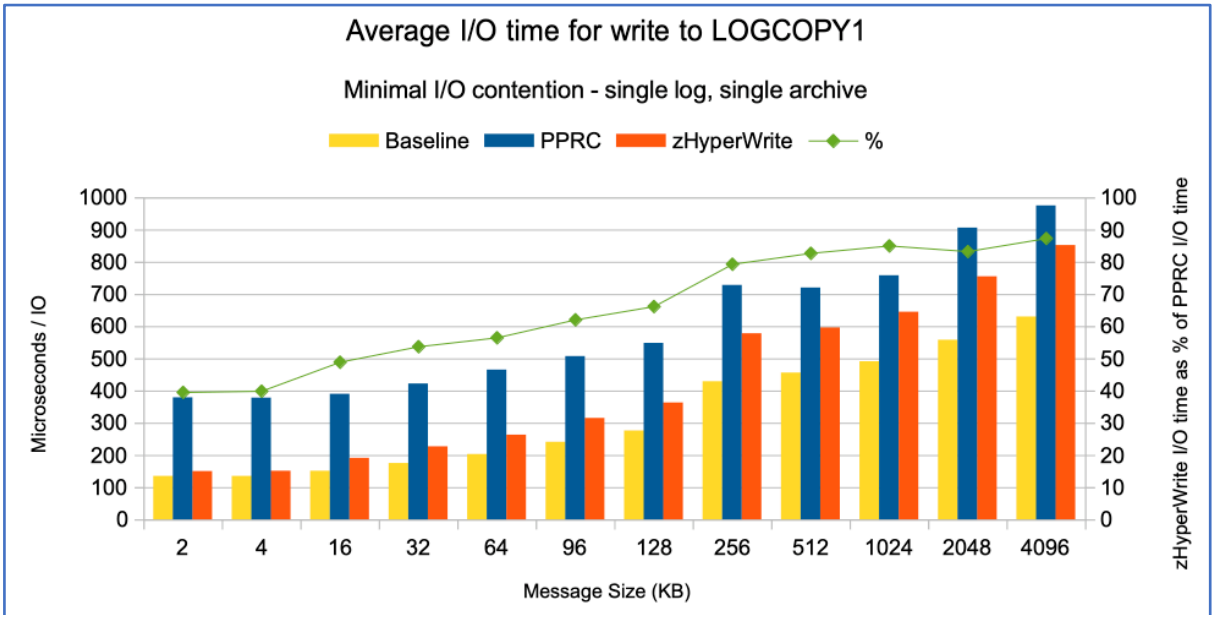
The measurements ran for a number of SMF intervals and aimed to drive the MQ logger task at the maximum sustainable throughput rate.

Reduced I/O time

When a persistent workload is throughput constrained, it can be due to the MQ logger task running at capacity. The overall throughput achieved may vary depending on message size(s) – and to achieve a higher throughput there are several options:

- Split the workload over multiple queue managers
- Reduce the I/O response time.

The following chart uses the MQ Statistics data to plot the average I/O times for the logger task:



For messages up to 96KB, zHyperWrite is taking less than 60% of the time of PPRC I/O requests.

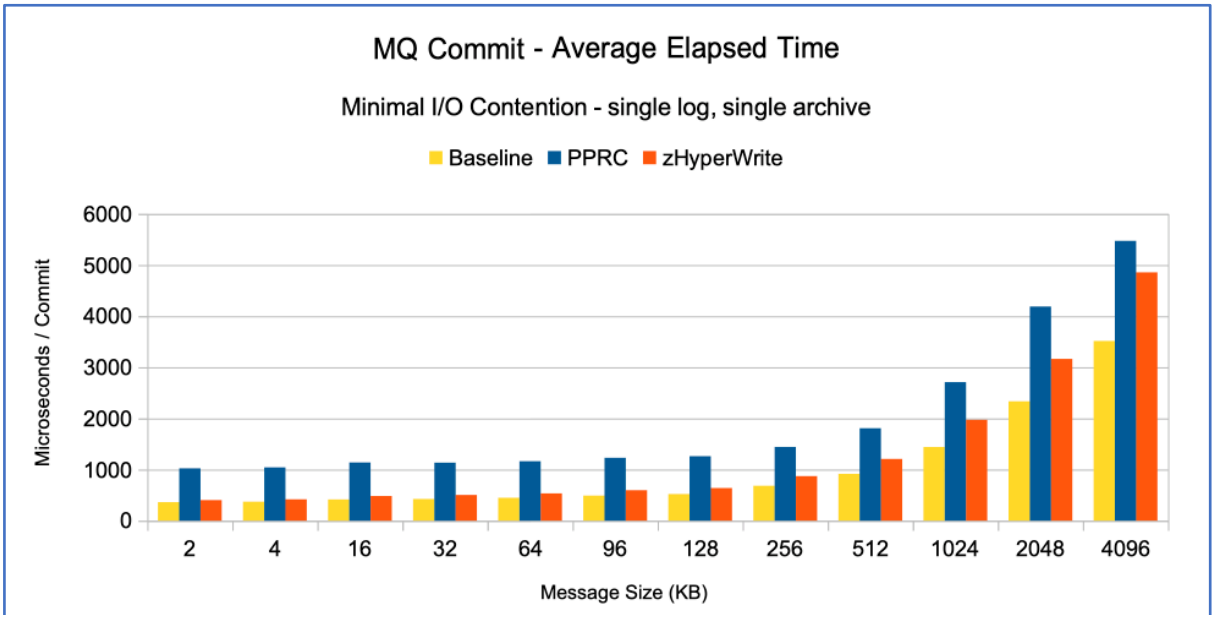
Once the messages exceed 96KB, the I/O times get closer, but in this configuration where the I/O subsystem is not constrained, zHyperWrite shows I/O times reduced by in excess of 12% for all message sizes.

Reduced Elapsed time for MQ Commit

In transactions, the wait time for the MQ log write I/O (typically at commit time) often makes up a significant proportion of the transaction latency, particularly when disk replication is enabled.

zHyperWrite is designed to speed up the latency of the log writes in a synchronous peer-to-peer remote copy (PPRC) environment. With zHyperWrite enabled, MQ triggers a log write I/O to the secondary disk subsystem in parallel to the log write I/O to the primary disk subsystem. By overlapping the 2 I/Os, which are run serially without zHyperWrite enabled, a significant reduction in MQ log write I/O write time is possible.

The following chart shows the average elapsed time for a range of message sizes.



For messages up to 128KB, the average time of the commit in the zHyperWrite configuration is less than 50% of the PPRC Metro Mirror configuration – and largely comparable to the baseline non-mirrored configuration.

The reduction in average commit time diminishes once the message size exceeds 128KB until, in the worst case, at 4MB messages, the average commit time is still 11% less than the PPRC configuration.

In some cases, another indirect benefit of zHyperWrite can be observed. The reduced log write wait time (from zHyperWrite) can lead to reductions in other types of contentions, which in turn can result in more CPU time savings because MQ has fewer contentions (suspend/resume) to manage.

Note that for most messages, the amount of data logged at put time far exceeds the amount of data logged at get time. See MP16 “How much log space does my message use?” for guidance on how much data is logged.

Improved sustainable log rate

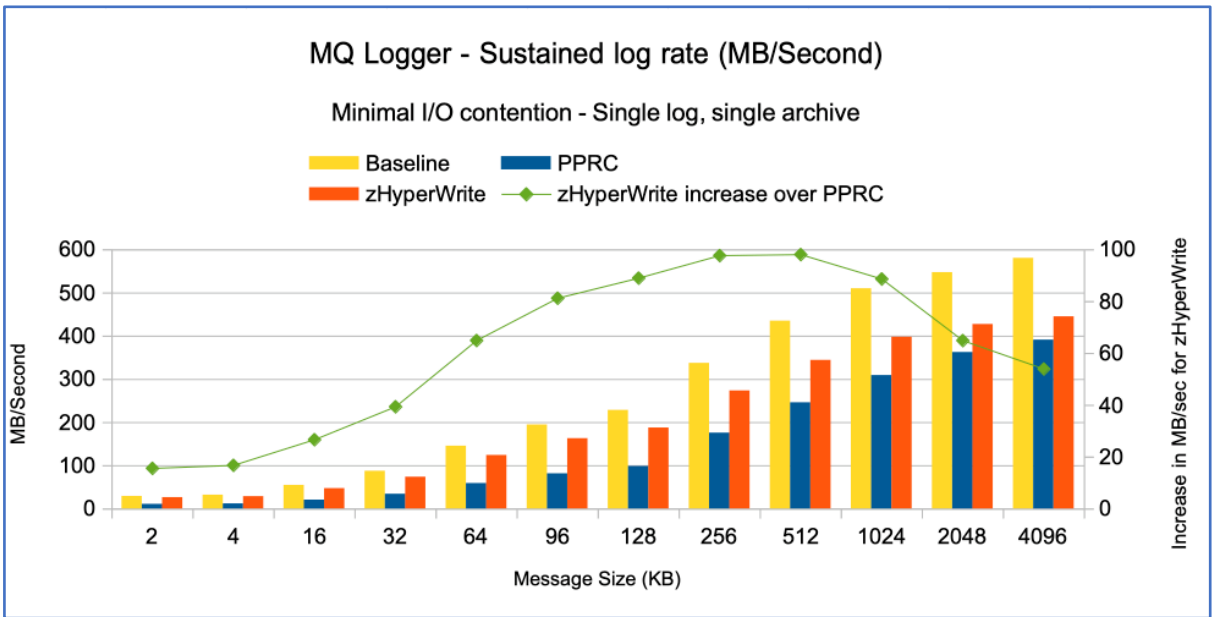
The chart in this section shows the rate in MB per second that the single active log was able to sustain on our systems.

In all cases, the queue manager is configured with single logs (and single archives), so for every MB put by the application, there are writes of between 2 MB (for baseline) to 3 MB (for PPRC or zHyperwrite) in total volume to the disks, i.e.:

- 1 MB to logs (1MB per log)
- 1 MB to archives (once the test is running, the queue manager driving the archive process constantly)
- 1 MB to the mirrored logs (1MB/log).

What this means is that when the charts show logging of 250 MB/sec, there is approximately 500MB/second written to the I/O subsystem in the baseline configuration and 750MB/second in the PPRC and zHyperwrite configurations.

In all of these measurements, the I/O subsystem is not being driven hard enough to fully utilise the cache. Appendix A demonstrates the implications of increased load on the I/O subsystem through the use of dual logs and dual archives, and how that impacts the sustained log rate.



Notes on chart:
The green line highlights the difference in MB/Second that zHyperwrite offers over PPRC.

For messages up to 128KB, zHyperwrite is achieving double the rate of the PPRC measurement.

As noted previously, the benefits of zHyperWrite grow with message size - in this case actual difference (MB/sec) is peaking between 128KB to 1MB.

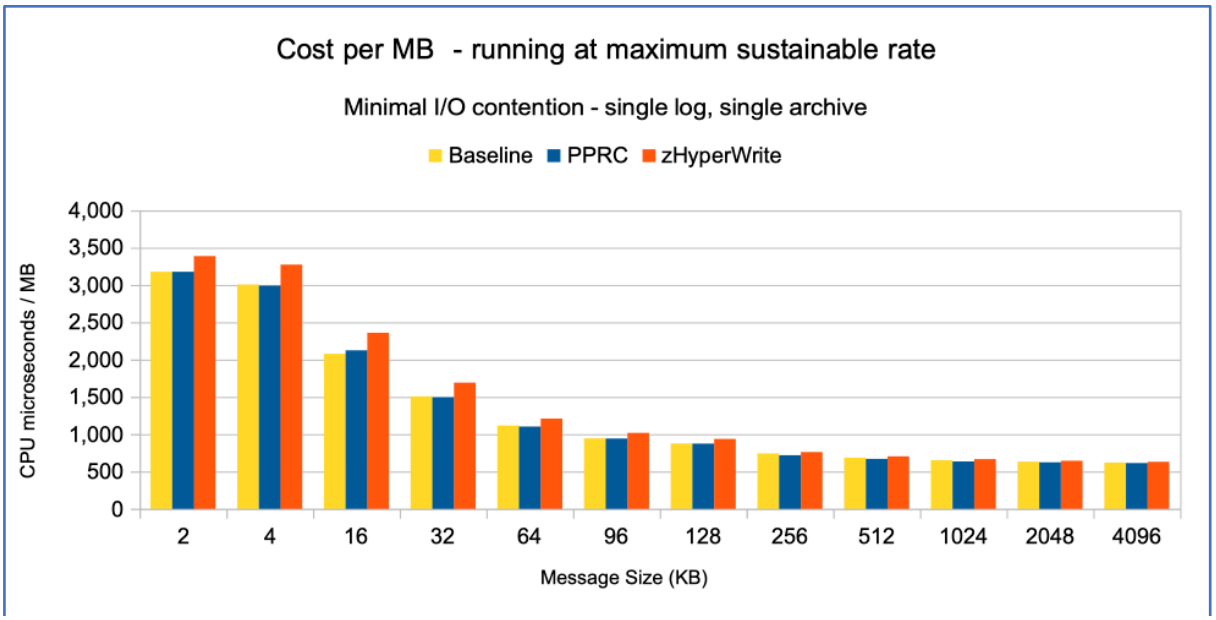
Impact to MQ queue manager costs

As previously mentioned the zHyperWrite configuration may see increased SRB costs in the MQ MSTR address space.

However, there are a number of factors which can affect the actual scale of the impact, and it is difficult to predict exactly how much any particular workload might be affected. Some of the factors include:

- Message size.
- Utilisation of MQ logger task.
- Amount of data written per I/O.

If we compare the cost per MB in the MQ MSTR address space when running at the maximum sustainable log rate, we can plot the following chart:



In this example, the maximum increase is using a 16KB workload, zHyperWrite is 11% more expensive than the PPRC configuration.

In these configurations with single archive/archive logs, the cost increase is typically of the order of 7.5%. In a dual active/archive log configuration, the increase is more typically 6%.

In the majority of these data points associated with smaller messages, the zHyperWrite configuration is logging at a significantly higher rate – which can increase the contention within the queue manager, which in turn can increase costs.

Using a message size where the log rate is similar should minimize any additional costs incurred from higher throughput. In these measurements, the 4MB workload achieves the most similar throughput.

Configuration	Rate MB/Second	Average queue manager CPU (including SRB) CPU uSeconds/MB	Average queue manager SRB CPU uSeconds/MB
Baseline	580	620	297
PPRC	391	613	295
zHyperWrite	445	630	333

This confirms that the zHyperWrite adds approximately 2.8% CPU but 12% in SRB over the PPRC configuration for our 4MB workload.

These calculations were made from data gathered when the MQ logger task was fully utilised.

In measurements where the transaction rate was artificially constrained, to achieve the same logging rate across configurations, effectively driving the logger task at a lower utilisation in the configurations where the I/O subsystem was more responsive, the I/O times made a difference to the amount of data logged per I/O, which in turn affected the average cost per MB.

For example, with 128KB messages running at a log rate of 40MB/second:

- PPRC's typical SRB cost per MB to be 0.84 CPU milliseconds per MB.
- zHyperWrite costs varied, ranging between 0.69 CPU milliseconds to 1.05 CPU milliseconds per MB. The reason for the variation was that despite the messaging rate being identical, the amount of data written per I/O dropped from 39 pages to 16 pages, resulting in extra I/Os and additional cost.

What this variability suggests is that predicting any increase in SRB is dependent on a number of factors, including the timing of the applications committing their messages. A good run, where more data per I/O is written may result in lower costs or minimal increase, but in cases where the faster response time resulted in less data per I/O saw MQ MSTR SRB costs increase up to 35%.

Summary

Whilst these measurements were run on a queue manager configured with single active and single archive logs, this is not the recommended configuration.

Single logs were used to demonstrate the impact of zHyperWrite, particularly when the I/O subsystem can be configured such that any additional I/O load does not cause NVS saturation.

At zero distance replication, zHyperWrite can:

- Reduce the elapsed time from an MQ Commit by up to 60%.
- Reduce the average I/O time when writing to the MQ active log(s) by up to 60%.
- Improve the maximum sustainable log rate by up to 2.4 times over PPRC.

Despite the RMF CPU report indicating that the LPAR was less than 25% busy for the entire measurement period, disabling archiving improved the log rate for the 4MB workload by up to 16%. This was due to periods where the number of CPU's available was insufficient for the number of tasks on the LPAR.

Appendix A details the performance observed on our systems when the queue manager was configured with dual active and archive logs. The additional load on the I/O subsystem from dual copies of active and archive logs coupled with the mirroring of the active logs was enough to result in NVS saturation, which often resulted in the performance benefits of zHyperWrite being less distinct.

Appendix A – dual active and archive logs – hitting I/O limitations

When running with dual active and archive logs on our single DS8870, the high throughput measurements saw increased I/O response times due to the saturation of non-volatile storage (NVS) in the I/O subsystem, which manifests itself in the form of increased Disk Fast Write Bypass (DFWBP).

What is Disk Fast Write Bypass?

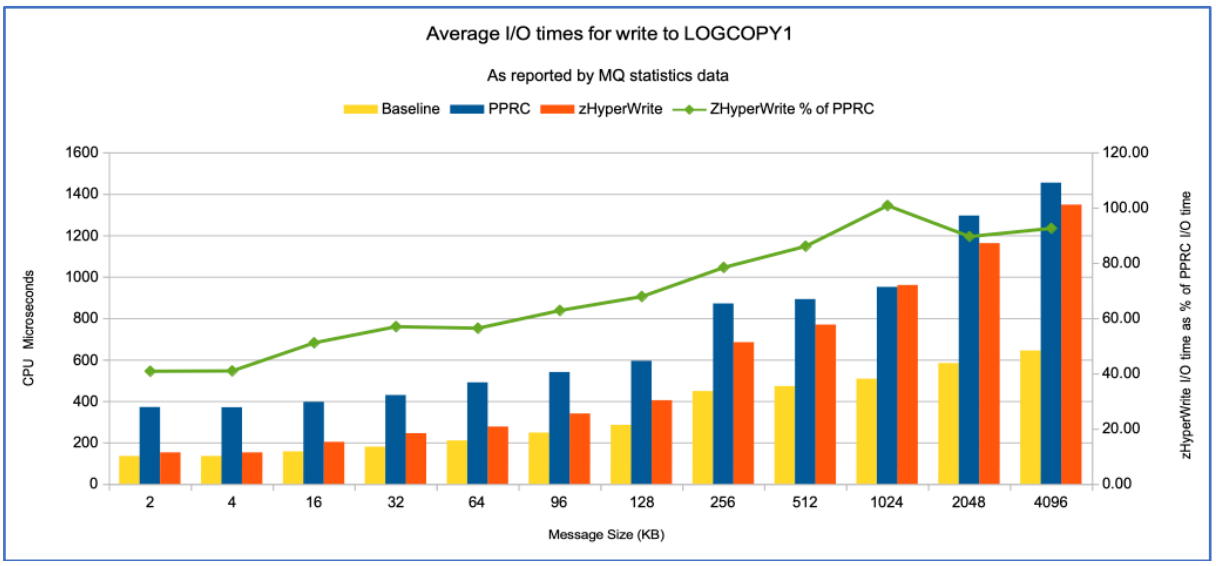
In the 3990/3390 era, when NVS is full, the write I/O bypasses the NVS and the data is written directly to the disk module (DDM).

In DS8000, when the NVS is full, the write I/O is retried from the host until the NVS space becomes available. So DFW Bypass must be interpreted as DFW Retry for DS8000.
If RMF shows that the DFW Bypass divided by the total I/O rate is greater than 1%, that is an indication of NVS saturation.

Impact on reduced I/O time

The following chart demonstrates the impact on the I/O time from increased DFWBP.

In our configuration, messages of 512KB and larger resulted in the I/O time for the zHyperWrite measurements being much closer to the PPRC measurements due to NVS saturation in excess of 15% - with the 4MB workload seeing DFWBP for 60% of the I/O.



For workloads that are not impacted by DFWBP i.e. messages up to 96KB, zHyperWrite is taking less than 60% of the time of PPRC I/O requests.

Sustainable log rate

The chart in this section shows the rate in MB per second that each log copy was able to sustain on our systems.

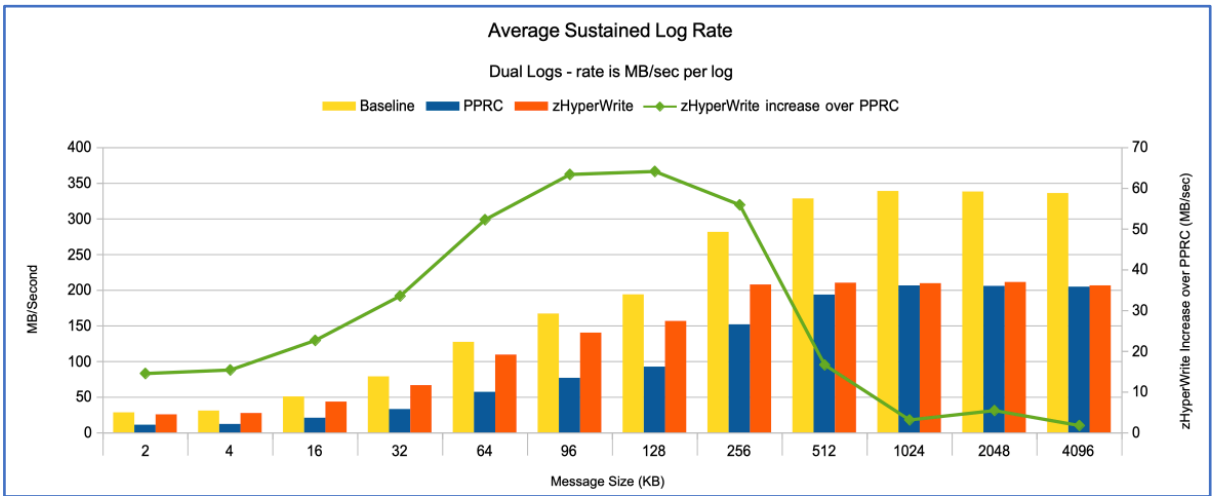
In all of these measurements, the queue manager is configured with dual active and archive logs, so for every MB put by the application, there are writes of between 4 MB (for baseline) to 6 MB (for PPRC or zHyperwrite) in total volume to the disks, i.e.:

- 2 MB to logs (1MB per log)
- 2 MB to archives (once the test is running, we are nigh on constantly archiving)
- 2 MB to the mirrored logs (1MB/log).

What this means is that when the charts show logging of 250 MB/sec, there is approximately 1GB/second written to the I/O subsystem in the baseline configuration and 1.5GB/second in the PPRC and zHyperwrite configurations.

As discussed previously, at the higher volumes, this causes waits for cache in the I/O subsystem, which is why the performance is flat-lining with the larger messages and the benefit of zHyperWrite is less obvious.

In the case of the PPRC measurement, because we are using a separate fibre channel, we actually see less waits for cache – the I/O subsystem is not being driven hard enough to fully utilise the cache.



Notes on chart:
The green line highlights the difference in MB/Second that zHyperwrite offers over PPRC.

For messages of 64KB or less, the zHyperwrite configuration is achieving double the rate of the PPRC measurement.

As noted previously, the benefits of zHyperWrite grow with message size - in this case actual difference (MB/sec) in sustained log rate is peaking with message sizes between 96KB to 512KB. Workloads with these message sizes are seeing 60MB/second being logged.

After this point the benefits diminish, largely because the zHyperWrite is being affected I/O cache waits (disk fast write bypass) and PPRC is not.

How we would (expect to) reduce Disk Fast Write Bypass

In the dual active/archive log measurements, we saw significant impact from DFWBP.

This impact affected the larger message sizes, particularly in the zHyperWrite configuration, such that the benefits over PPRC were minimal.

On our DS8870, there is 32GB of non-volatile storage (NVS) available. This is split evenly between the odd and even half.

The way our MQ datasets are organised means that data is written unevenly to each half:
ODD half
- LOGCOPY1
- Mirror of LOGCOPY1

EVEN half
- LOGCOPY2
- Mirror of LOGCOPY2
- Archive log 1
- Archive log 2

This means that we are writing significantly more to the even half, but we can configure measurements such that we use a single log and archive which will not contend for NVS.

Ideally, we would configure 2 DASD boxes:
- box 1: LOGCOPY1 and archive log 1
- box 2: LOGCOPY2 and archive log 2

This should ensure even load to each box and run at a rate where NVS is not saturated.