IBM MQ V9.4.3 for Linux (x86-64 platform)

Performance Report(For Monitoring Queue and Channel , Activity trace, MQ Trace)

Version 1.0 - August 2025

Keshava M Chandraiah

IBM MQ Performance

IBM India Software Lab

Bangalore

Introduction

The report has been prepared using IBM MQ V9.4.3. This report provides insights into the overhead on message throughput and CPU utilization when using the techniques mentioned below to analyse issues.

Monitoring in IBM MQ

IBM MQ provides monitoring attributes and status commands to keep track of channels (communication links) and queues (message storage). This report measures the performance impact of MQ Queue and Channel Monitoring https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=network-real-time-monitoring

MQ Activity Trace

The MQ Activity Trace uses the mqat.ini file as a configuration file. It controls how much API call information MQ records for applications. This information can be used to debug message flows, check application behaviour (Get/Put/Commit), and messaging performance

https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=multiplatforms-activity-trace-configuration-file-mqatini

MQ Trace

The MQ Trace is a diagnostic mechanism that records either internal MQ operations (internal trace) or application-level API calls (API trace). It is usually gathered on request from IBM Support, though some users have used it to confirm application behaviour. The trace can include application level MQ API calls (MQPUT, MQGET, MQOPEN, MQCLOSE, etc.).

https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=reference-strmqtrc-start-trace

Workloads

Workloads used in the generation of performance data for this report. All workloads are requester/responder (RR) scenarios which are synchronous in style because the application putting a message on a queue will wait for a response on the reply queue before putting the next message. They typically run 'unrated' (no think time between getting a reply and putting the next message on the request queue).

Workload	Description
RR-DQ-BB	Distributed queueing between two queue managers on separate hosts, with binding mode requesters and responders.
RR-BB	Binding mode requesters and responders
RR-CC	Client mode requesters, and responders on separate, unique hosts

Refer below link for more information

https://github.com/ibm-messaging/mqperf/blob/gh-pages/MQ_V9.4_Performance_Report_xLinux_v1.0.pdf

Monitoring Queue and Channel

The various workload used **-** RR-BB RR-DQ-BB RR-CC

Below commands used to alter Queue and Channel attribute

ALTER QMGR MONCHL(HIGH)

ALTER QMGR MONQ(HIGH)

1) Workload (RR-BB)

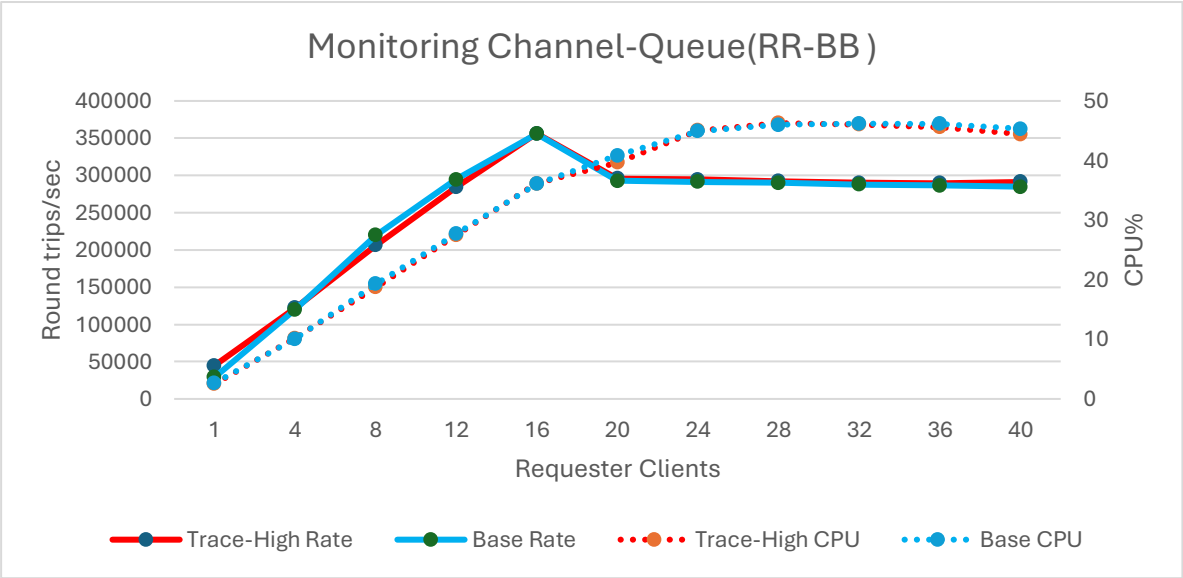


Figure 1 - Monitoring Channel & Queue using RR-BB for Non-Persistent messaging(2K)

Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*Monitoring of Channel and Queue = OFF*	*Base CPU*
Trace-High Rate	*Monitoring of Channel and Queue = High*	*Trace-High CPU*

- The throughput for various Workloads for different Requester Clients are same
- There is no impact on message throughput

2) Workload RR-DQ-BB

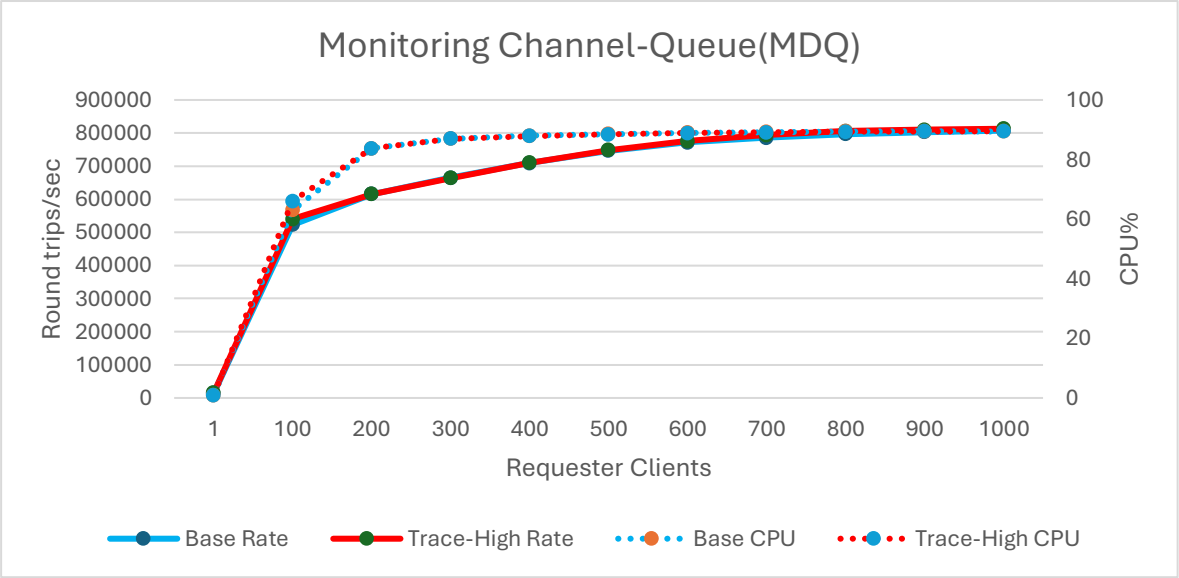


Figure 2 - Monitoring Channel & Queue using RR-DQ-BB *for Non-Persistent messaging(2K)*

Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*Monitoring of Channel and Queue = OFF*	*Base CPU*
Trace-High Rate	*Monitoring of Channel and Queue = High*	*Trace-High CPU*

- The throughput for various Workloads for different Requester Clients are same
- There is no impact on message throughput

3) Workload RR-CC

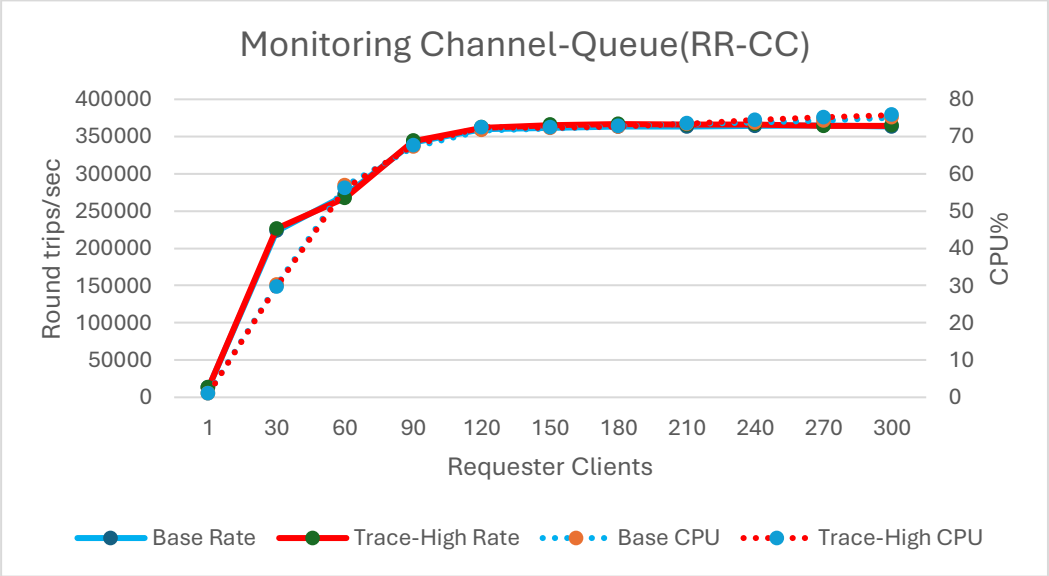


Figure 3 - Monitoring Channel & Queue using RR-CC *for Non-Persistent messaging(2K)*

Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*Monitoring of Channel and Queue = OFF*	*Base CPU*
Trace-High Rate	*Monitoring of Channel and Queue = High*	*Trace-High CPU*

- The throughput for various Workload's for different Requester Clients are same
- There is no impact on message throughput

Activity Trace

Performance Measurement – Trace Across All Application Pairs

The Activity Trace is applied across all Requester/Responder pairs for a given workload

(Refer below section to know how to enable

Appendix C : Enabling Activity Trace: *Across All Application Pairs for Trace Level=High)*

Persistence

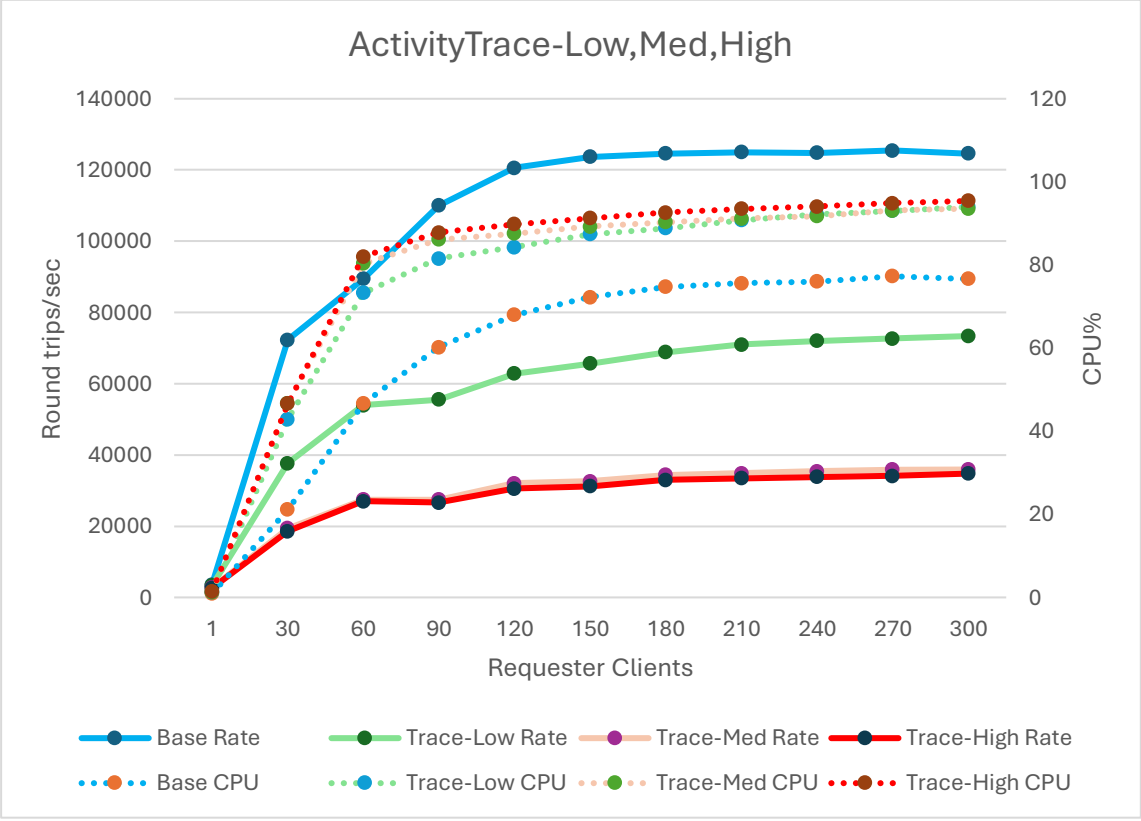


*Figure 4 - Activity Trace for all application(*RR-CC*) for Persistent messaging(2K) with Level = Low, Medium, High*

Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*No Activity Trace*	*Base CPU*
Trace-Low Rate	*Activity Trace enabled for Low*	*Trace-Low CPU*
Trace-Med Rate	*Activity Trace enabled for Medium*	*Trace-Med CPU*
Trace-High Rate	*Activity Trace enabled for High*	*Trace-High CPU*

Workload	**Message Rate**	**CPU%**	**Throughput Impact%**
Without Activity Trace	124652	76	
Activity Trace=Low	73375	94	−41
Activity Trace=Medium	35912	93	-71
Activity Trace=High	34754	95	-72

- The table gives the throughput for various Workloads when Requester Clients reaches to 300
- This shows how message throughput is affected for a given Activity Trace Level, with Medium and High trace levels having the most impact.

Non Persistence

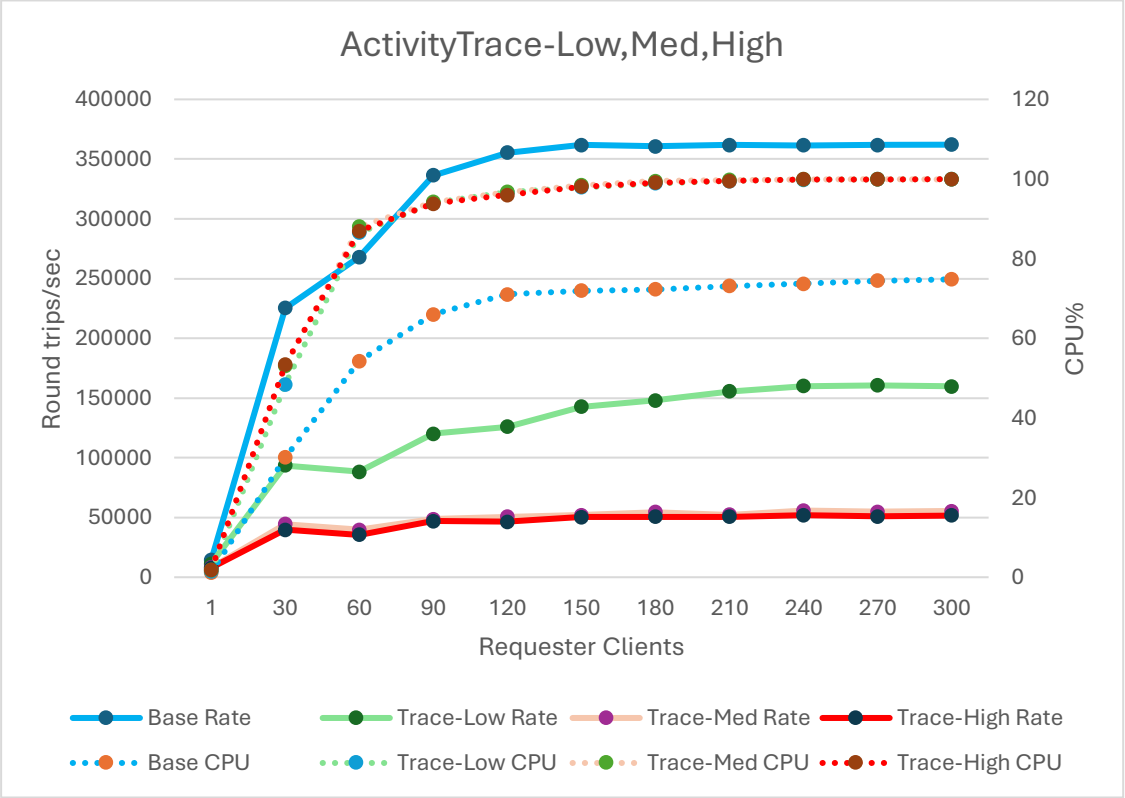


*Figure 5 - Activity Trace for all application(*RR-CC*) for Non-Persistent messaging(2K) with Level = Low, Medium, High*

Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*No Activity Trace*	*Base CPU*
Trace-Low Rate	*Activity Trace enabled for Low*	*Trace-Low CPU*
Trace-Med Rate	*Activity Trace enabled for Medium*	*Trace-Med CPU*
Trace-High Rate	*Activity Trace enabled for High*	*Trace-High CPU*

Workload	Message Rate	CPU%	Throughput Impact%
Without Activity Trace	362086	75	
Activity Trace=Low	159619	100	−56
Activity Trace=Medium	55269	100	-85
Activity Trace=High	51713	100	-86

- The table gives the throughput for various Workloads when Requester Clients reaches to 300
- This shows message throughput significantly affects for a given Activity Trace Level. It affects significantly when TraceLevel=Medium or High

Performance Measurement - Activity Trace output redirected to external file

There is no impact on throughput when Activity Trace logging to the external file system

Example /opt/mqm/samp/bin/amqsact -m -w 60 > /tmp/acttrace.log

Performance Measurement - Single Data Point Tests

The Activity Trace was enabled to capture end-to-end interactions between **Requester and Responder applications** under the **same workload of 100 Clients**. The workload itself remains constant across all runs; what changes is the scope of tracing applied.

In each iteration, the trace is applied to different sets of Requester/Responder pairs as below:

- **One Pair of Applications:** Req0 ↔ Res0
- **Five Pairs of Applications:** Req0/Res0 … Req5/Res5
- **Ten Pairs of Applications:** Req0/Res0 … Req10/Res10
- **Twenty Pairs of Applications:** Req0/Res0 … Req20/Res20
- **Forty Pairs of Applications:** Req0/Res0 … Req40/Res40
- **Sixty Pairs of Applications:** Req0/Res0 … Req60/Res60

Each configuration is executed in a separate iteration, with traces collected only for the selected pairs while the workload continues for all 100 clients.

This method ensures consistent workload conditions while analysing the impact of tracing across varying levels of application concurrency, making it possible to compare performance trends

(*Refer below section to know how to enable*

Appendix C : Enabling Activity Trace: Single Data Point Tests)

Persistence

Workload(RR-CC)-100 Req/Res-Low Level Activity Trace

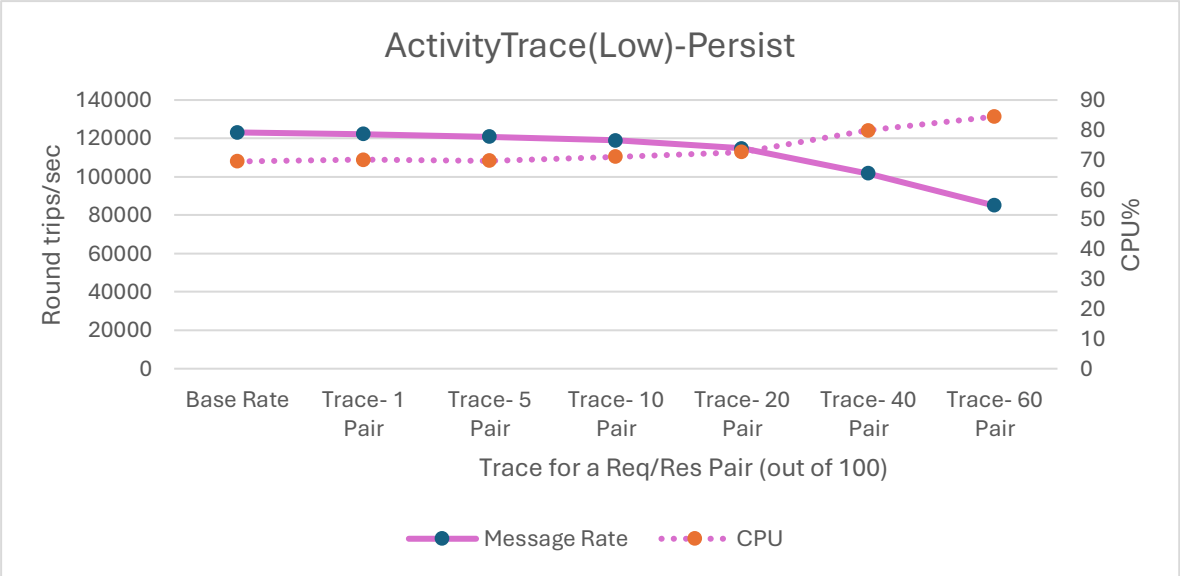


Figure 6 - Activity Trace for Different Pair of applications(RR-CC) *for Persistent messaging(2K) with Level = Low*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to Applying Activity Trace(Low) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications*	*CPU*

Workload	**Message Rate**	**CPU%**	**Throughput Impact%**
No Activity Trace	123112	70	

Activity Trace-Pair of 1	122297	70	-1
Activity Trace-Pair of 5	120939	69	-2
Activity Trace=Pair of 10	118929	70	−3
Activity Trace=Pair of 20	114727	72	−7
Activity Trace=Pair of 40	101697	80	−17
Activity Trace=Pair of 60	85006	84	−31

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 85006/sec for a trace of 60 Req/Res pairs).

Workload(RR-CC)-100 Req/Res-Medium Level Activity Trace

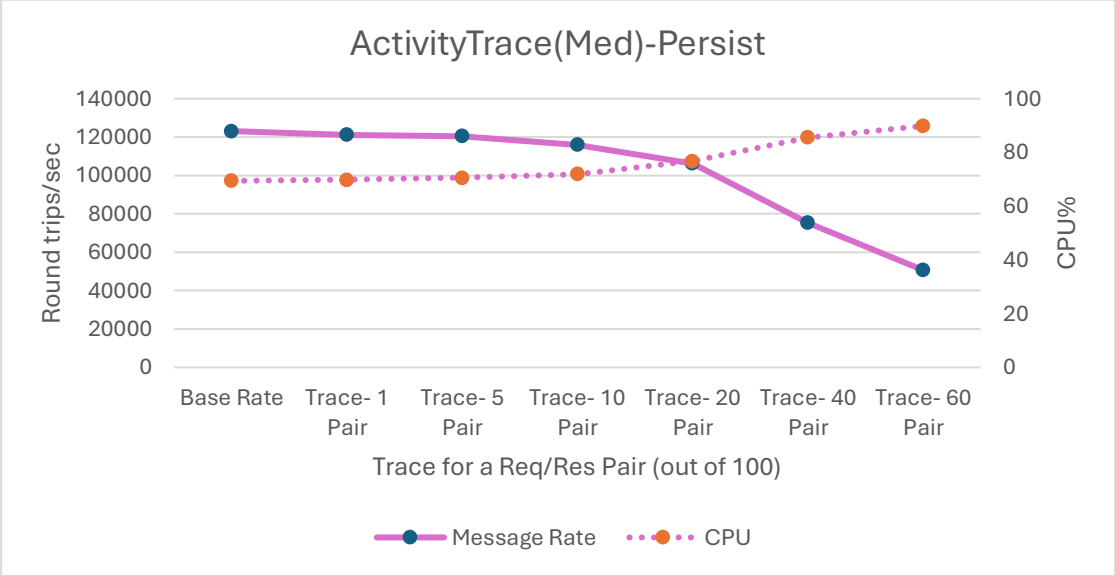


Figure 7 - Activity Trace for Different Pair of applications(RR-CC) *for Persistent messaging(2K) with Level = Medium*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to Applying Activity Trace(Medium) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications*	*CPU*

Workload	Message Rate	CPU%	Throughput Impact%
No Activity Trace	123112	70	
Activity Trace-Pair of 1	121270	70	−1
Activity Trace-Pair of 5	120555	70	−2
Activity Trace=Pair of 10	115998	72	−6
Activity Trace=Pair of 20	106294	77	−14
Activity Trace=Pair of 40	75395	85	−39
Activity Trace=Pair of 60	50697	90	−59

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 50697/sec for a trace of 60 Req/Res pairs).

Workload(RR-CC)-100 Req/Res-High Level Activity Trace

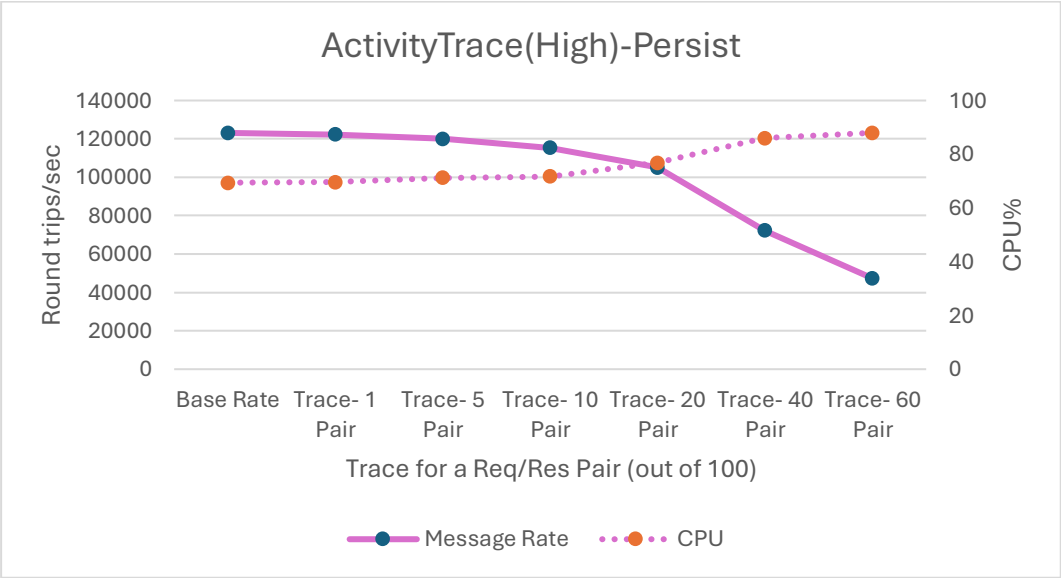


Figure 8 - Activity Trace for Different Pair of applications(RR-CC) *for Persistent messaging(2K) with Level = High*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to Applying Activity Trace(High) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications*	*CPU*

Workload	Message Rate	CPU%	Throughput Impact%
No Activity Trace	123112	69	
Activity Trace-Pair of 1	122209	69	-1
Activity Trace-Pair of 5	120055	71	−2
Activity Trace=Pair of 10	115276	71	−6
Activity Trace=Pair of 20	105176	76	−15
Activity Trace=Pair of 40	72270	86	−41
Activity Trace=Pair of 60	47346	87	−62

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 47346/sec for a trace of 60 Req/Res pairs).

NonPersistence

Workload(RR-CC)-100 Req/Res-Low Level Activity Trace

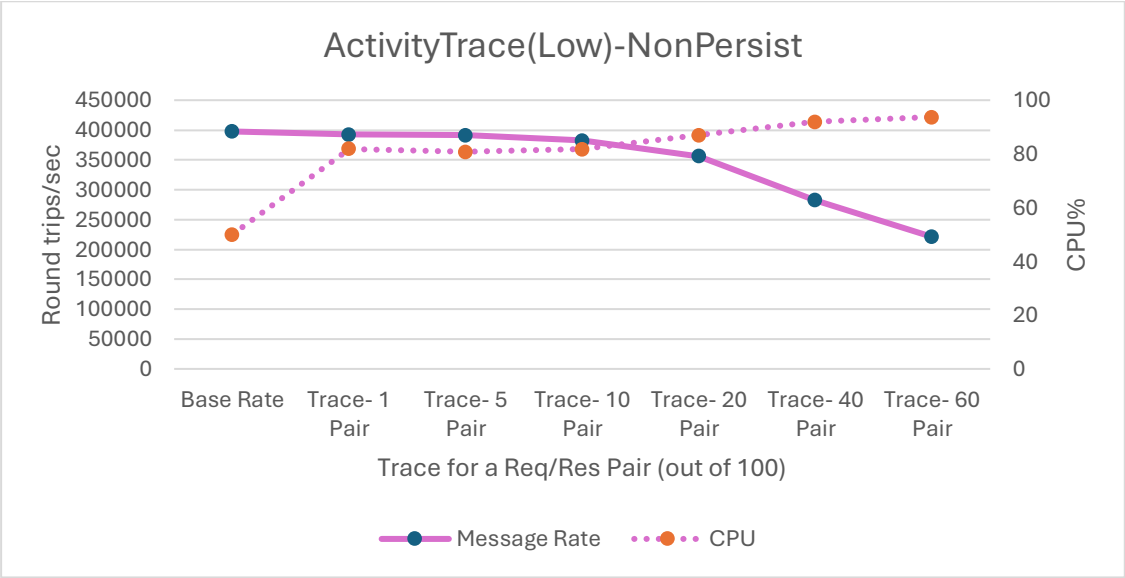


Figure 9 - Activity Trace for Different Pair of applications(RR-CC) *for NonPersistent messaging(2K) with Level = Low*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to Applying Activity*	*CPU*

	Trace(Low) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications	

Workload	**Message Rate**	**CPU%**	**Throughput Impact%**
No Activity Trace	397646	50	
Activity Trace-Pair of 1	392391	81	−1
Activity Trace-Pair of 5	391168	80	−2
Activity Trace=Pair of 10	382602	81	−4
Activity Trace=Pair of 20	356086	87	−10
Activity Trace=Pair of 40	282702	92	−29
Activity Trace=Pair of 60	221154	94	−44

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 221154/sec for a trace of 60 Req/Res pairs).

Workload(RR-CC)-100 Req/Res-Medium Level Activity Trace

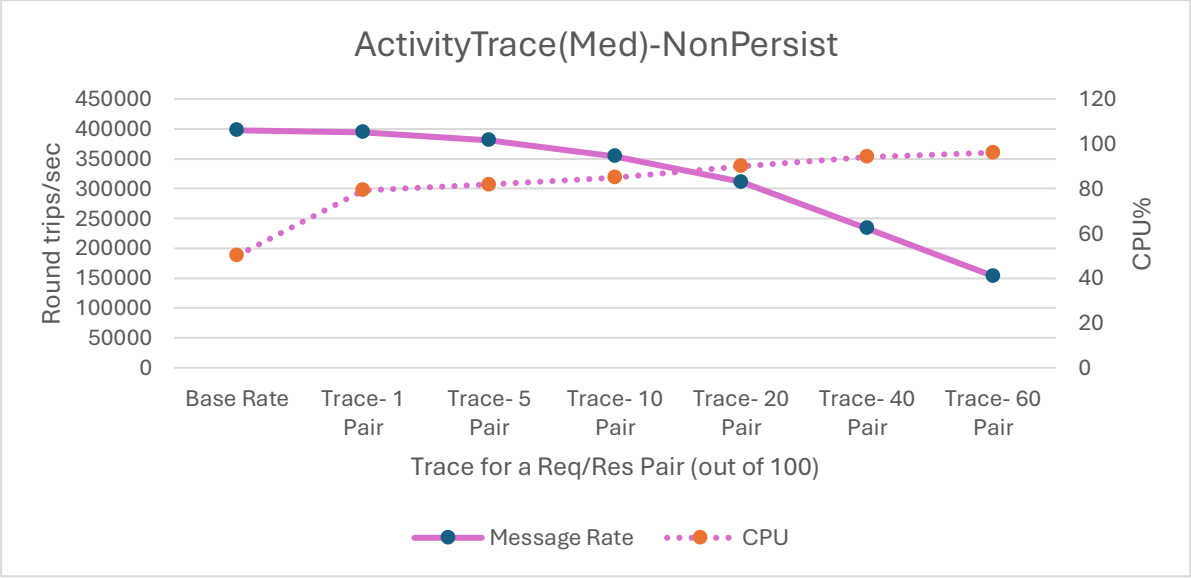


Figure 10 - Activity Trace for Different Pair of applications(RR-CC) *for NonPersistent messaging(2K) with Level = Medium*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to*	*CPU*

	Applying Activity Trace(Medium) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications	

Workload	Message Rate	CPU%	Throughput Impact%
No Activity Trace	397646	50	
Activity Trace-Pair of 1	394274	79	-1
Activity Trace-Pair of 5	381073	82	−4
Activity Trace=Pair of 10	354069	85	−11
Activity Trace=Pair of 20	311358	90	−22
Activity Trace=Pair of 40	233442	94	−41
Activity Trace=Pair of 60	153806	96	−61

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 153806/sec for a trace of 60 Req/Res pairs).

Workload(RR-CC)-100 Req/Res-High Level Activity Trace

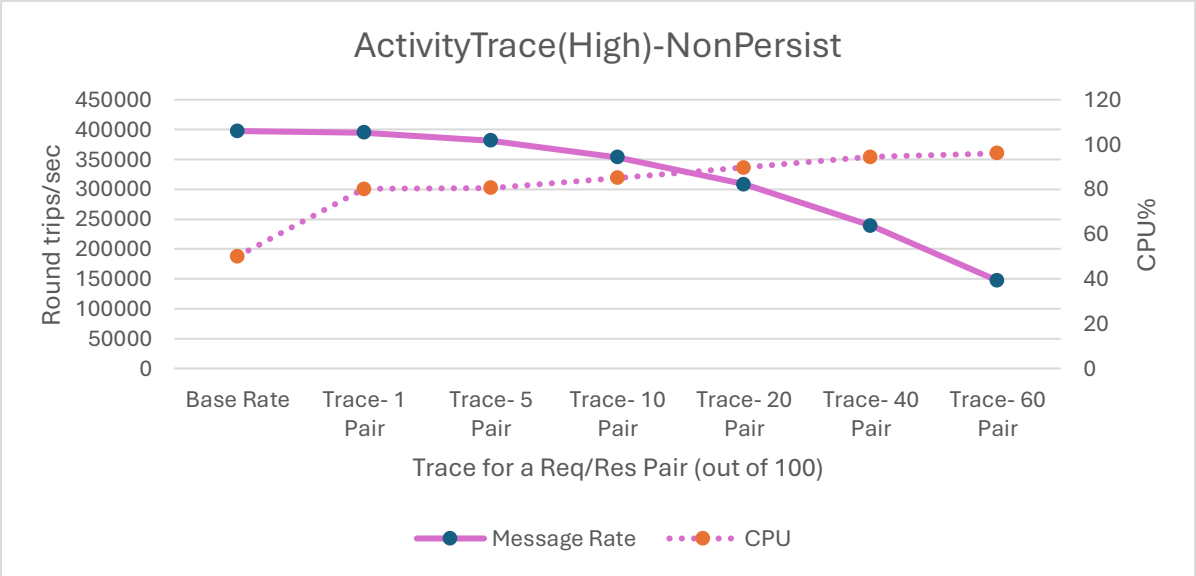


Figure 11 - Activity Trace for Different Pair of applications(RR-CC) *for NonPersistent messaging(2K) with Level = High*

Legend	***Description***	***Corresponding CPU Usage***
Message Rate	*Represents From Base Rate(No trace) to Applying Activity Trace(High) for 1 Pair, 5 Pair, 10 Pair, 20 Pair, 40 pair, 60 Pair of Req/Res Applications*	*CPU*

Workload	**Message Rate**	**CPU%**	**Throughput Impact%**
No Activity Trace	397646	50	
Activity Trace-Pair of 1	395040	80	-1
Activity Trace-Pair of 5	381711	80	−4
Activity Trace=Pair of 10	353848	85	−11
Activity Trace=Pair of 20	308267	90	−22
Activity Trace=Pair of 40	239130	94	−40
Activity Trace=Pair of 60	147179	96	−63

- The table gives the throughput for a trace of different pairs of Req/Res among 100 Requester Clients
- The message throughput is impacted progressively as the number of applications being traced increases (dropping to 147179/sec for a trace of 60 Req/Res pairs).

MQ Trace v/s Activity Trace (Low, Medium, High)

Comparison of IBM MQ Trace and Activity Trace at Low, Medium, and High levels

MQ Trace is a facility typically used to support IBM in diagnosing behaviour. It typically has a heavy impact on performance due to the depth of tracing that is performed. MQ Trace enabled at the API level.

The Activity Trace is applied across all Requester/Responder pairs for a given workload, with varying trace levels (Low, Medium, High)

Persistence

Workload(RR-CC)

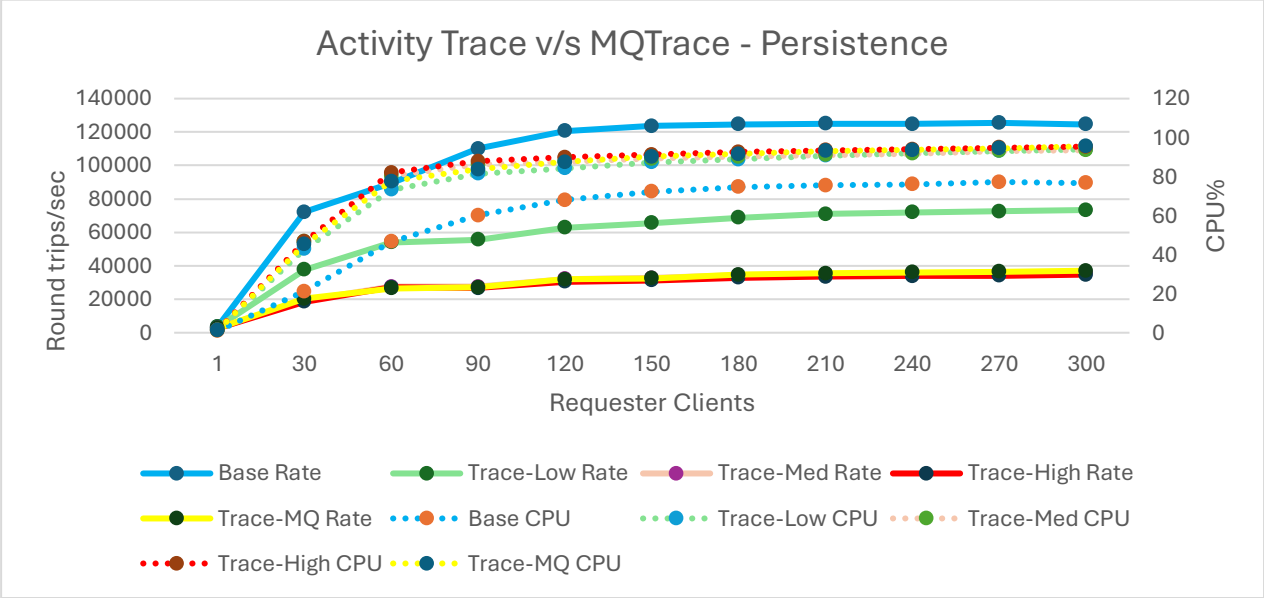


*Figure 12 - Activity Trace v/s MQ Trace(API) - applications(*RR-CC*) for Persistent messaging(2K)*

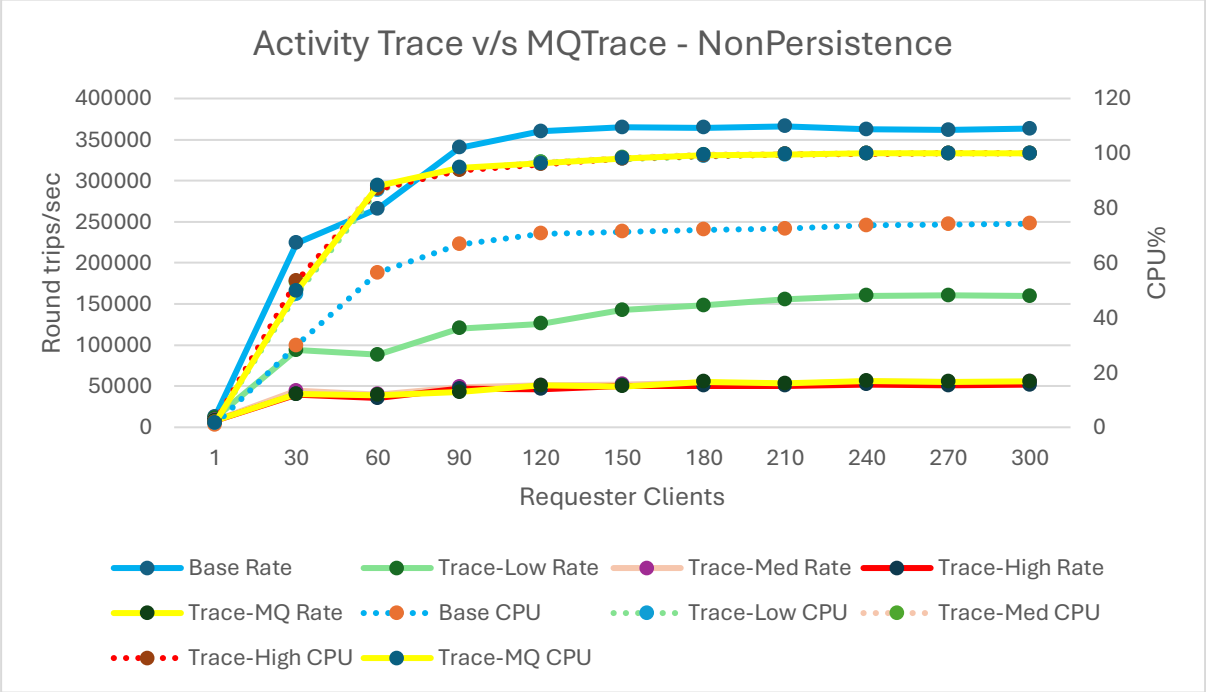
Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*No Activity Trace*	*Base CPU*
Trace-Low Rate	*Activity Trace enabled for Low*	*Trace-Low CPU*
Trace-Med Rate	*Activity Trace enabled for Medium*	*Trace-Med CPU*
Trace-High Rate	*Activity Trace enabled for High*	*Trace-High CPU*
Trace-MQ Rate	*MQ Trace (API only) enabled*	*Trace-MQ CPU*

Workload	**Message Rate**	**CPU%**	**Throughput Impact%**
Without Trace	124652	76	
Activity Trace=Low	73375	94	−41
Activity Trace=Medium	35912	93	−71
Activity Trace=High	34754	95	−72
MQ Trace	37083	95	−70

- The table gives the throughput for various Workloads when Requester Clients reaches to 300
- Turning on Activity Trace at medium or high levels has a performance impact similar to MQ Trace. However, this impact can be reduced by applying the trace selectively, as demonstrated in the previous sections

Non Persistence

Workload(RR-CC)



Legend	***Description***	***Corresponding CPU Usage***
Base Rate	*No Activity Trace*	*Base CPU*
Trace-Low Rate	*Activity Trace enabled for Low*	*Trace-Low CPU*
Trace-Med Rate	*Activity Trace enabled for Medium*	*Trace-Med CPU*
Trace-High Rate	*Activity Trace enabled for High*	*Trace-High CPU*
Trace-MQ Rate	*MQ Trace (API only) enabled*	*Trace-MQ CPU*

Workload	Message Rate	CPU%	Throughput Impact%
Without Trace	363210	74	
Activity Trace=Low	159619	100	−56
Activity Trace=Medium	55269	100	−84
Activity Trace=High	51713	100	−85
MQ Trace	55607	100	−84

*Figure 13 - Activity Trace v/s MQ Trace(API) - applications(*RR-CC) *for NonPersistent messaging(2K)*

- The table gives the throughput for various Workloads when Requester Clients reaches to 300
- Turning on Activity Trace at medium or high levels has a performance impact similar to MQ Trace. However, this impact can be reduced by applying the trace selectively, as demonstrated in the previous sections

Conclusion

- Monitoring of channels and queues provides valuable operational insights while adding no measurable performance overhead, making it safe to use
- The overhead of Activity Trace depends on the trace level selected. By applying trace only to specific applications, the overhead can be reduced, though the impact should always be carefully evaluated before use.
- At Higher and Medium levels, Activity Trace introduces an overhead comparable to MQ Trace(API only). Therefore, both tracing methods should be enabled selectively and only for diagnostic purposes, with careful consideration of performance impact
- It was also observed that enabling Activity Trace alone (even without applying changes to the mqat.ini) impacts message rate significantly, with up to an 80% reduction as by default all applications would be traced.

Appendix A: Test Configurations

A.1 Hardware/Software – Set1 All of the testing in this document (apart from when testing results are shown from a different platform and are clearly identified as such) was performed on the following hardware and software configuration:

A.1.1 Hardware Server1, client1 & client2 are three identical machines:

Lenovo ThinkSystem SR630 V3

Processor: 2 × 16-core Intel® Xeon® Gold 6544Y CPUs @ 3.60 GHz

Memory: 256 GB RAM

Storage:

2 × 800 GB Lenovo ThinkSystem 2.5in PM1655 Mixed Use SAS 24Gb HS SSDs, RAID0 (Boot/OS)

2 × 1.6 TB NVMe SSDs, RAID0 (Data)

Networking:

2 × 10 GbE interfaces

2 × 100 GbE interfaces

Lenovo System x3550 M5 – [5463-L2G] 2 x 12 core CPUs. Core: Intel® Xeon® E5-2690 v3 @ 2.60GHz 128GB RAM

A.1.2 Software

Red Hat Enterprise Linux Server release 9.6 (Plow)

MQ-CPH MQI test driver (see Appendix B :)

Appendix B : Resources

MQ-CPH (The IBM MQ C Performance Harness)

https://github.com/ibm-messaging/mq-cph

https://ibm-messaging.github.io/mqperf

https://github.com/ibm-messaging/mqperf/blob/gh-pages/MQ_V9.4_Performance_Report_xLinux_v1.0.pdf

Appendix C: Enabling Activity Trace (Modification to mqat.ini)

Across All Application Pairs for Trace Level = High

Below stanza enables trace for all applications which having prefix Requester or Responder

ApplicationTrace:

ApplName=Requester*

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder*

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=*

Trace=OFF

Single Data Point Tests

Below stanza enables trace for 5 Pair of applications such as Requester0/Responder0 Requester1/Responder1 Requester2/Responder2 Requester3/Responder3 Requester4/Responder4 for a Trace Level = HIGH

ApplicationTrace:

ApplName=Requester0

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder0

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Requester0

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder1

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Requester1

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder2

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Requester2

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder3

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Requester4

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=Responder4

Trace=ON

TraceLevel=HIGH

ApplicationTrace:

ApplName=*

Trace=OFF