Mentora MQ Accelerator Fact Sheet

Features and Benefits

Zero development time = Fast time-to-first test
A user-extensible LoadRunner MQ framework, with an out-of-the-box set of base scripts for putting and getting messages, enables you to run your first test in minutes by simply editing the script parameter file with your MQ configuration details. (See Parameter List)

Zero script maintenance time = Fast time-to-test across environments
Configure your scripts to test multiple environments (e.g., Dev, Test, Prod) by simply adding multiple configuration rows in the script parameter file. Quickly select the environment you want to test by easily marking the row you want to use in a given test.

Message round-trip times = True measure of application scalability
The actual round-trip times are obtained for each unique message by subtracting the retrieval time from its Reply Queue from the arrival time on its Arrival Queue. We do this by matching sent and received messages via uniquely-assigned ids and obtaining their Arrival Queue time by querying the MQ message header.

Intelligent load balancing = Flood your MQ infrastructure without overloading it
Our scripts have built-in intelligence to self-throttle as your queues cross a specified capacity threshold. We do this by querying the MQ Browser to determine the queue's current message depth, comparing this number to the maximum messages allowed on the queue, and idling the script the moment the threshold is exceeded.

Dynamic Transaction names = High results granularity
Our dynamic transaction names capture the scenario Group Name, the virtual user's Vuser ID, and the round-trip or one-way setting, enabling you to capture highly granular results, to, for instance, compare Group 1's message round-trip times on *local queue A* to Group 2's times on *remote queue B*.

Round-trip Time over Load Template = All the data you need to analyze scalability
Our round-trip graph template shows the min/max/avg/90th percentile times in milliseconds, and the message count for each Vuser-Group. This enables you to visualize scalability over load, compare round-trip times for different queue managers on local vs. remote servers, and obtain total messages processed. (See graph *Message Round-Trip Times over Load*)

Model your full application messaging requirements = Realistic load scenarios

Our framework enables you to model complex, multi-server, multi-queue, client and server MQ configurations, and transmit large volumes of actual message traffic to consuming applications. We do this by providing configurable parameters for every available MQ property and by enabling you to point your scripts to message directories containing any number and size of application message files. Messages and script parameters are loaded into virtual memory for fast and robust script execution.

What's In the Box

MQ Accelerator Base Scripts:

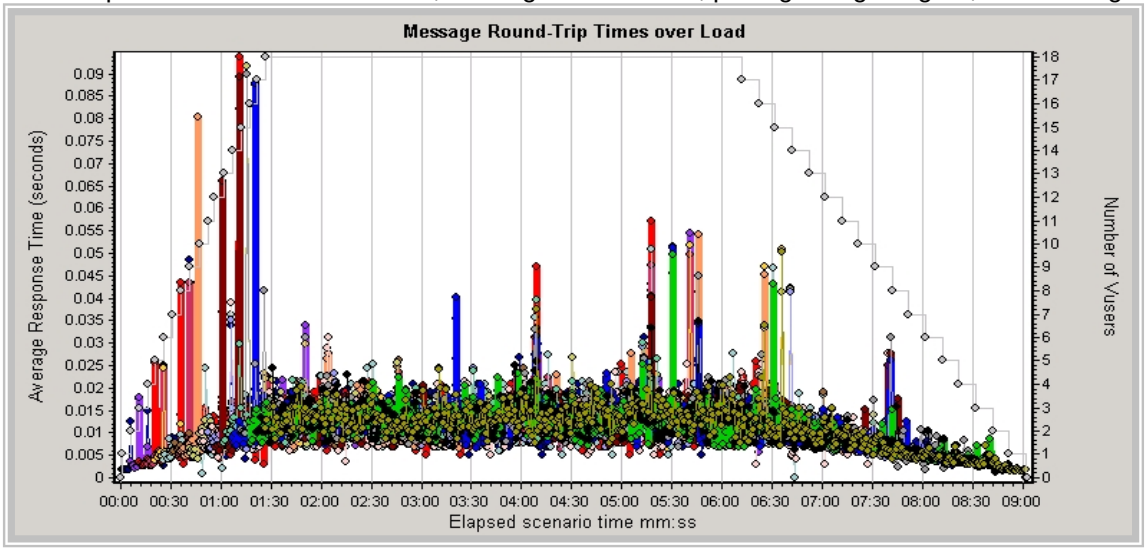
Script Name	Description
ClientPut	Sends messages to and Arrival Queue, via an MQ client to MQ Server connection
ClientGet	Gets messages from a Reply Queue, via an MQ client to MQ Server connection
ServerPut	Sends messages to and Arrival Queue, via an MQ server-to-server connection
ServerGet	Gets messages from a Reply Queue, via an MQ server-to-server connection

Script Parameter list:

Scope	Parameter	Description	Example
Script	ScriptName	Name of the MQ script	ServerPut
Arrival (PUT) Queue	Qmgr	Arrival queue manager name	AQM1
	QmanagerIP	Arrival queue manager IP address	10.0.1.242
	Qname1	Arrival queue name	LOOP.TO.AQM2
	Channel	Client-to-server connection name	LRUNNER
	Port	Arrival queue manager listener port	1415
Reply (GET) Queue	rQmgr	Reply queue manager name	AQM1
	rQmanagerIP	Reply queue manager IP address	10.0.1.242
	rQname1	Reply queue name	LOOP.END
	rport	Reply queue manager listener port	1414
Message	DataFile	Directory path where application messages are located	c:\\mq\\msg_dir
	MsgID	Prefix prepended to a message's unique id	app_A_
	Persistence	[Y/N] determines whether the message should be stored and forwarded in the event of a broken message path connection	N
	Expiry	Message expiration time in 10ths of seconds [-1 if message does not expire]	-1
	CorrelID	[Optional] MQ Correlation ID used to correlate response messages with request messages when an application invokes a request-response operation	101
Operation	roundtrip	[Y/N] determines whether a script should wait for a reply ("round-trip") message	Y
	WaitInterval	Maximum time a Get script waits for a reply message, in milliseconds	800
Script	LastItem	Keyword "end" terminates the parameter list (makes the list future-expandable)	end

Analysis Graph Template: *Message Round-Trip Times over Load*

Round-trip times of an18 v-users test, running for 9 minutes, putting and getting 83,000 messages.



Color	Group	Vuser #	Transaction Name	Min. (ms)	Ave. (ms)	Max. (ms)	90th % (ms)	Msg. Count
	ServerPut_1	10	DynamicQ.ServerPut_1.10.0.+RoundTrip	2	13	48	22	4930
	ServerPut_1	11	DynamicQ.ServerPut_1.11.0.+RoundTrip	2	12	82	22	4254
	ServerPut_1	12	DynamicQ.ServerPut_1.12.0.+RoundTrip	3	13	94	23	3997
	ServerPut_1	13	DynamicQ.ServerPut_1.13.0.+RoundTrip	3	12	50	23	3849
	ServerPut_1	14	DynamicQ.ServerPut_1.14.0.+RoundTrip	4	13	52	23	3770
	ServerPut_1	15	DynamicQ.ServerPut_1.15.0.+RoundTrip	1	13	51	23	3762
	ServerPut_1	16	DynamicQ.ServerPut_1.16.0.+RoundTrip	6	13	52	23	3757
	ServerPut_1	17	DynamicQ.ServerPut_1.17.0.+RoundTrip	5	13	83	21	3794
	ServerPut_1	18	DynamicQ.ServerPut_1.18.0.+RoundTrip	4	13	92	22	3846
	ServerPut_1	19	DynamicQ.ServerPut_1.19.0.+RoundTrip	3	13	39	22	3935
	ServerPut_1	20	DynamicQ.ServerPut_1.20.0.+RoundTrip	5	12	88	21	4044
	ServerPut_1	21	DynamicQ.ServerPut_1.21.0.+RoundTrip	5	12	89	22	4224
	ServerPut_1	22	DynamicQ.ServerPut_1.22.0.+RoundTrip	4	12	40	22	4436
	ServerPut_1	23	DynamicQ.ServerPut_1.23.0.+RoundTrip	3	12	88	21	4712
	ServerPut_1	24	DynamicQ.ServerPut_1.24.0.+RoundTrip	2	12	90	23	5075
	ServerPut_1	25	DynamicQ.ServerPut_1.25.0.+RoundTrip	2	11	50	22	5474
	ServerPut_1	26	DynamicQ.ServerPut_1.26.0.+RoundTrip	1	11	35	22	6506
	ServerPut_1	27	DynamicQ.ServerPut_1.27.0.+RoundTrip	1	11	50	22	8975
			Totals (all vusers)	**3.1**	**12.3**	**65.2**	**22.2**	**83340**
			Load (v-users)			**18**		