Dwarfguard 0.8.0 performance tests.

.

Dwarfguard 0.8.0 follows performance tests of MAMAS (preceding product branding) versions 0.7.0 and 0.6.0. The results should be comparable when the new features are not enabled (especially DCL which may impact performance noticeably when a big number of devices is sending data).

There are three different test types in Dwarfguard testing:

- stability tests ... runs for a number of hours. Tests stability under standard conditions and raises flags in case of memory leaks.
- benchmark tests ... to measure different HW / VMs for comparability and clue for sizing deployments. Proves Dwarfg deployment with particular sizing is able to handle the tested number of devices.
- peak tests ... to discover Dwarfg SW and deployments limits and breaking points.

Next to the test results, HW specs for environments, test description and methodology is given in the document.

The Dwarfg 0.8.0 early adopter 3 is intended to be used for up to 40000 devices.
The subject of early adopter version testing are stability and benchmark testings.
Find out more in test overview.

General Dwarfguard 0.8.0 performance testing verdict:

Stability basic tests	PASSED	2023-10-03	Test sets: Stab-2-2:C1; Stab-4-2:C2; Stab-8-2:C3; Stab-16-2:C4; Stab-16-2-H1
Stability medium test	PASSED	2023-10-06	Test sets: Stab-4-48
Stability max (40000) test	PASSED	2023-11-25	Test sets: Stab-4-48-top
Stability long test			Test sets: Stab-4-672

Testing overview

As mentioned earlier (Testing summary), there are three types of performance tests. All are here because we need to measure:
- number of accepted and dropped requests (and resulting percentage) FOR ALL TESTS
- time for peak and benchmark tests
- memory usage for stability tests
- CPU utilization (%) for stability tests
- dwarfgd log review to make detail search for warnings and errors (for stability tests)
- number of pushing threads causing requests drops+resends (peak test)

Next to measurements there are a few important calculated metrics:
- ideal maximal throughput (benchmark test)
- typical maximal throughput (20% of ideal) (benchmark test)
- number of pushes per second (benchmark and peak test)
- maximal recommended # of devices per deployment type (HW / VM specs)

Environment specs		CPU cores	CPU threads	RAM MiB	stability test	benchmark	peak test
Proxmox Linux container Intel Xeon E5 2.2GHz 4C HT	C1	1	1	512			
	C2	2	2	1024			
	C3	4	4	2048			
	C4	8	8	4096			
AWS instance	A1 (small)						
	A2 (medium)						
	A3 (big)						
Baremetal AMD E350@1.6 GHz 2C	H0	2	2	16384			
Baremetal Core i5 1.7GHz 4C HT	H1	4	8	16384			
Baremetal Core i7 2.7GHz 4C HT	H2	4	8	32768			
Baremetal Core i7 3.2GHz2 GHz 6C HT	H3	6	12	32768			

Tests specs	ID	SSL?	# of devices	push/work T	# of loops	# of minutes	Human-time	Notes
Stability	Stab-2-2	Yes	1000	2/2	N/A	120	2 hours	
	Stab-4-2	Yes	3000	4/2	N/A	120	2 hours	
	Stab-8-2	Yes	10000	8/2	N/A	120	2 hours	
	Stab-16-2	Yes	30000	16/2	N/A	120	2 hours	
	Stab-4-48	Yes	3000	4/2	N/A	2880	48 hours	
	Stab-8-48-top	Yes	40000	8/2	N/A	2880	48 hours	New in 0.8.0
	Stab-4-672	Yes	3000	4/2	N/A	40320	4 weeks	
Benchmark	Bench-6-SSL	Yes	1200	6/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-6-noSSL	No	1200	6/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-12-SSL	Yes	2400	12/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-12-noSSL	No	2400	12/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-24-SSL	Yes	4800	24/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-24-noSSL	No	4800	24/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-48-SSL	Yes	9600	48/1,2,4,8	10	N/A		Re-def in 0.8.0
	Bench-48-noSSL	No	9600	48/1,2,4,8	10	N/A		Re-def in 0.8.0
Peak	PeakR-64-max	No	61440	64	N/A	10	10 minutes	
	PeakR-256-max	No	61440	256	N/A	10	10 minutes	
	PeakR-1024-max	No	61440	1024	N/A	10	10 minutes	
	PeakD-64-max	No	61440	64	N/A	10	10 minutes	
	PeakD-256-max	No	61440	256	N/A	10	10 minutes	
	PeakD-1024-max	No	61440	1024	N/A	10	10 minutes	
	PeakF-64-max	No	61440	64	N/A	10	10 minutes	
	PeakF-256-max	No	61440	256	N/A	10	10 minutes	
	PeakF-1024-max	No	61440	1024	N/A	10	10 minutes	

Test run map 0.7.0	C1	C2	C3	C4	A1	A2	A3	H0	H1	H2	H3
Stab-2-2	Yes										
Stab-4-2		Yes									
Stab-8-2			Yes								
Stab-16-2				Yes					Yes		
Stab-4-48								Yes			
Stab-8-48-top											Yes
Stab-4-672								TBD			
Bench-6-SSL	Yes	Yes	Yes	Yes				Yes	Yes	Yes	Attempted
Bench-6-noSSL	Yes	Yes	Yes	Yes				Yes	Yes	Yes	Attempted
Bench-12-SSL		Yes	Yes	Yes				Yes	Yes	Yes	Attempted
Bench-12-noSSL		Yes	Yes	Yes				Yes	Yes	Yes	Attempted
Bench-24-SSL			Experimental	Experimental							
Bench-24-noSSL			Experimental	Experimental							
Bench-48-SSL											
Bench-48-noSSL											
PeakR-64-max			v. 1.0.0	v. 1.0.0				v. 1.0.0			
PeakR-256-max			v. 1.0.0	v. 1.0.0				v. 1.0.0			
PeakR-1024-max			v. 1.0.0	v. 1.0.0				v. 1.0.0			
PeakD-64-max				v. 1.0.0				v. 1.0.0			
PeakD-256-max				v. 1.0.0				v. 1.0.0			
PeakD-1024-max				v. 1.0.0				v. 1.0.0			
PeakF-64-max				v. 1.0.0				v. 1.0.0			
PeakF-256-max				v. 1.0.0				v. 1.0.0			
PeakF-1024-max				v. 1.0.0				v. 1.0.0			

Stability tests									
ID / Environment →			Stab-2-2 / C1	Stab-4-2 / C2	Stab-8-2 C3	Stab-16-2 C4	Stab-16-2 H1	Stab-4-48 H0	Stab-8-48-top H3	Stab-4-672 C2
Test specs		Devices	1000	3000	10000	30000	30000	3000	40000	3000
		Emulator threads	2	4	8	16	16	4	8	4
		Set: Time / minutes	120	120	120	120	120	2880	2880	40320
		Set loop time / sec	200	200	200	200	200	200	200	200
HW specs		CPU cores	1	2	4	8	8	2	12	2
		RAM / MiB	512	1024	2048	4096	32768	16384	32768	1024
Resources	Pre-test	Available MiB OS	296.00	783.00	1800.00	3800.00	30000.00	14000.00	29618.00	
		Used MiB OS	215	240	226	217	608	790	2420	
		dwarfgd RSS (MiB)	44	44	44	44	49	49	48	
	Post-test	Available MiB OS	265	687	1400	2700	29000	14000	28859	
		Used MiB OS	246	336	617	1300	1800	883	3179	
		mamasd RSS (MiB)	57	75	186	415	395	88	498	
		avg load 15	0.07	0.16	0.51	1.18	0.78	0.27	1.99	
		avg load 15 / cpu core	0.07	0.08	0.13	0.15	0.10	0.14	0.17	0.00
Results	Test numbers	Processesd loops	34	33	32	31	31	769	674	
		Total time	7302	7308	7260	7353	7427	173013	172817	
		Real Loop time/sec (1)	215	221	227	237	240	225	256	#DIV/0!
		Estimated data-pushes	36000	108000	360000	1080000	1080000	2592000	34560000	36288000
		Performed Pushes	34000	99000	320000	930000	930000	2307000	26960000	
		Estim. Avg reqs/sec	5.00	15.00	50.00	150.00	150.00	15.00	200.00	15.00
		Rough avg reqs/sec (1)	4.66	13.55	44.08	126.48	125.22	13.33	156.00	
	Errors	Data ERR/retries	0	0	0	0	31	0	2	
		push errors (%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	#DIV/0!
		push errors (1=100%)	0.000000000	0.000000000	0.000000000	0.000000000	0.000033333	0.000000000	0.000000074	#DIV/0!
		Crashes	0	0	0	0	0	0	0	
		Reboots	0	0	0	0	0	0	0	
		Log entries	2	0	1	1	0	1	0	
		Log analysis	CFG profile sync	None	Mid-air profile collision	Mid-air profile collision	1 Invalid ID	Mid-air profile collision	None	
	Calculations	RSS increase MiB	13	31	142	371	346	39	450	
		RSS increase %	29.55	70.45	322.73	843.18	706.12	79.59	937.50	
		MiB per device	0.013	0.010	0.014	0.012	0.012	0.013	0.011	
		RAM utilization %	48.05	32.81	11.04	5.30	1.86	4.82	7.39	
		Max safe # of devs	2081	9143	90619	566267	1616842	62218	541620	
	Summary	RAM utilization note	reasonable	ok	perfect (low)	perfect (low)	perfect (low)	perfect (low)	perfect (low)	
		CPU utilization note	perfect (low)	perfect (low)	perfect (low)	perfect (low)	perfect (low)	perfect (low)	perfect (low)	
		Verdict	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	TBD

Notes	
(1)	The results are not exact as the total time includes registration time. In reality, there are a little bit more requests per second and the loop time is a little shorter than that.

Log analysis explained	
CFG profile sync	(Warning) Configuration profile sync to DB was delayed during device registration period (Warning). This may happen when server is processing registration requests coming very fast. As fot testing case on C1 container on 1 CPU core this is considered normal when registration of 1000 devices happens in a straight line.
Mid-air profile collision	(Warning) Configuration profile with default values for a firmware creation attempted more than once in parallel. As all of the Advantech router device types in this test share the same profile, this is perfectly possible to happen. No impact on the system or data.
Invalid ID	(Not logged in daemon log) One of the 30000 emulated devices failed to register properly (not important to log into daemon log as there may be lots of declined requests). This caused that device with invalid ID to fail on every of the consecutive 31 data push loops. In reality, the device would re-register again on the next loop (but that functionality is not implemented in the Emulator)

Results are # of processed device data pushes per second									
	Spec:	C1	C2	C3	C4	H0	H1	H2	H3
	CPU thr	1	2	4	8	2/2	4/8	4/8	6/12
	RAM	512	1024	2048	4096	16384	16384	32768	32768
	HW	Intel Xeon E5 2CPU 4/8 each				E-350	Core i5	Core i7	Core i7
	Arch	Server				L PWR	Mobile		Desktop
	GHz	2.2				1.6	1.7	2.7	3.2
Test	Handlers								
6-SSL	1	125.53	237.45	284.02	323.93	96.64	237.43	322.88	223.66
	2	143.52	263.74	280.05	322.22	106.94	258.84	346.91	225.62
	4	155.61	259.27	279.35	324.21	122.41	245.27	342.75	236.99
	8	163.62	256.22	305.75	318.47	110.94	250.21	345.59	226.99
6-noSSL	1	532.73	593.70	542.25	571.30	399.55	536.65	478.85	394.96
	2	557.02	555.35	571.09	510.49	387.63	463.24	454.16	390.84
	4	529.33	583.16	509.44	489.84	412.97	389.06	481.05	333.86
	8	428.75	542.35	460.86	478.21	386.09	420.42	398.39	393.66
12-SSL	1		422.55	558.79	640.98	128.42	638.80	710.52	558.69
	2		428.03	631.88	720.40	136.54	638.95	708.44	556.69
	4		447.04	635.47	727.37	140.65	658.71	708.79	556.94
	8		449.95	581.68	727.29	143.33	658.76	705.20	556.31
12-noSSL	1		789.99	1032.92	1049.33	570.51	947.36	1019.84	920.30
	2		785.82	1045.74	1044.52	574.89	932.94	1031.92	861.75
	4		788.45	1050.50	1051.98	581.74	932.42	838.75	910.65
	8		787.75	830.65	1049.64	579.74	876.20	1012.96	909.75
24-SSL	1			730.70	884.27				
	2			734.38	803.13				
	4			732.87	883.83				
	8			738.21	880.18				
24-noSSL	1			1012.33	1112.64				
	2			1018.36	1112.61				
	4			1031.06	1114.66				
	8			1030.49	1120.56				
48-SSL	1								
	2								
	4								
	8								
48-noSSL	1								
	2								
	4								
	8								
Ideal max devs		25106	47490	56804	64786	19328	47486	64576	
Safe max devs		20085	37992	45443	51829	15462	37989	51661	

Notes/colors explained	
	Errors occured (network connection/resolution). The real result would be higher (another device handled instead of test tool thread waiting for request timeout) Also, refused device would retry later.
	At the time tests were performed, available traffic generator HW performance was too low to saturate the device under test. Unable to provide correct results at the moment but it would definitely be higher.
	Amounts are valid for agent regular period of 200 seconds. For e.g. the regular data-push period of 100 seconds you need to divide the number by 2! (NOTE: the default interval is 260 seconds leaving some manuever space). NOTE: the result does NOT take into account machine RAM, it is simply an approximation of the maximal number of devices that can be handled - in reality, you need to check the stability tests results for memory limits! Also note that the numbers are computed from the lowest throughput values available (from the weakest test) and for SSL-deployment. Looking at the bigger machines able to handle stronger tests the numbers are practically multiplied and note you can also offload SSL layer thus bring the number of processed devices higher again.

Findings

The number of Dwarfguard handling threads does not affect the throughput much. The likely reason is that the processing of SSL layer and processing done by Apache before handing the request over to the application itself is taking more resources than the Dwarfguard, thus the throughput does not scale by adding more handler threads.

Next round of tests needs to compare the (CPU) resource usage of Dwarfguard with Apache and MariaDB to find the reason of throughput not scaling with number of handling threads.

Peak tests are

Comparison of some data with later versions

Notes:

Each minor version adds a lot of functionality but also brings optimizations. Comparable results (100%) are considered success and results within 101-120% of resources usage and 99-80% throughput are considered ok.

Testing methodics is updated (and testing tools improved) with every version making comparison very hard.

The measurements and comparison are based on the basic stability (Stab-2-2 to Stab-16-2) and basic performance (originally Bench-4-SSL but that got replaced by Bench-6-SSL in 0.8.0) test results.

Findings for 0.8.0

Memory consumption optimizations resulted in 25% decrease of memory footprint after test. What's even better is roughly 33-45% decrease of the per-device memory footprint.

A number of computation optimizations increased the throughput by 30-70% but given the accent was on bringing down the resource usage and not increasing throughput, the resulting throughput increase is a nice bonus.

		RAM – RSS MiB	% of previous	KiB per device	% of previous	Pushes/sec	% of previous
0.6.0	C1	49	100	15	100	142	100
	C2	54	100	7	100	182	100
	C3	112	100	8	100	189	100
	C4						
	H1	61	100	8	100	193	100
0.7.0 - introduced DCL	C1	57	116	20	133	143	101
	C2	95	176	19	271	175	96
	C3	271	242	23	288	161	85
	C4	665	100	21	100	190	100
	H1	491	805	15	188	155	80
0.8.0	C1	57	100	13	65	126	88
	C2	75	79	10	53	237	135
	C3	186	69	14	61	284	176
	C4	415	62	12	57	324	171
	H1	395	80	12	80	237	153
	H0	88	100	13	100	97	100