Techniques for Cache Performance Improvement using Cgroups on Wi-Fi Router

By:

Ram Chandra Jangir (CS21M517)

Sneha Maganahalli Rajendranath (CS21M522)

Abstract:

There are new challenges for shared resources on CPU. Even though CPU headroom is extensively discussed, CPU caches could pose much higher challenge as they are common for the applications & Wi-Fi Workload. To overcome some of the challenges, an easy-to-use flexible resource manager framework is proposed to tune the application aggressiveness to ensure Wi-Fi throughput is not compromised.
Test results show that the "resource manager framework" can restore Wi-Fi performance by 80% of the peak Wi-Fi performance. Proposed "Resource Manager Framework" will address most of the challenges and reduces our "competitive exposure"

Introduction:

Wi-Fi workloads are routed through CPU thus sharing the CPU sub-system with other Applications running on router.

Over CPU utilization is a prime factor of concern, however a shared L2 Cache and it's trashing could be detrimental to the overall Wi-Fi performance/throughput or other Application Performance.

Problem Statement:

Step #A: Demonstrate that a shared L2 cache could be a concern for performance.

Explore a test application & build a test bed.

Challenge: Routers have their own applications and we don't have access to them. Is there a way to generalize the Applications running on router?

Step #B: Find solution to address the performance drop & build a prototype with the experiments.

Step #C: Finalize the recommendations.

Step #D: Analysis & Results.

Step #E: Conclusion.

Techniques:

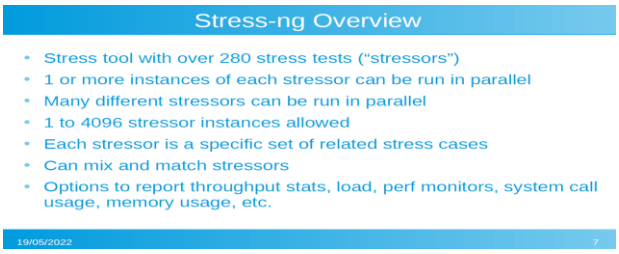
We do not have access to Applications which run on the router/Access Point. Different routers have different applications running on them (anywhere between Compute Intensive to I/O intensive)

Stress-ng to the rescue!!

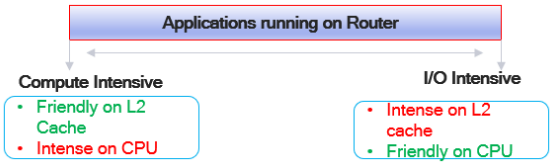
Stress-ng is an opensource application capable of executing many stressors. Experimented with Stress-ng extensively to make sure the "stressors" are working as expected.

Selected two stressors:

a. Compute Intensive.
   Eg: Vector Matrix multiplication
b. I/O intensive.
   Eg: Streaming test case



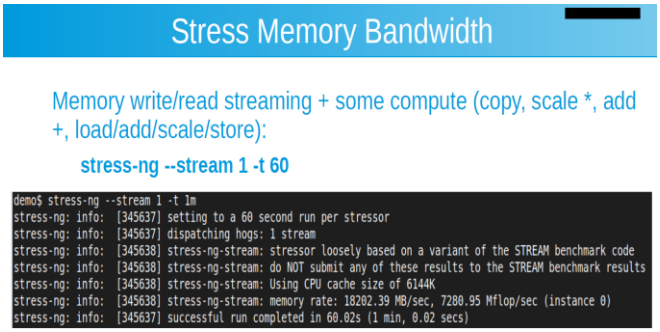
Stressors classification:



We are focusing on I/O intensive stressors in this paper since cache involves more I/O intensive work load as compared to compute intensive work load.

Step #A: Stress-ng test cases

I/O intensive stressor:



Commands used:

For running on all 'n' CPU cores:

stress-ng --stream <'n'> –perf

Step #B: Find solution to address the performance drop & build a prototype with the experiments:

Case1:

Only I/O intensive stressor is running on the system. (i.e. No Wi-Fi running on system):

4-Core I/O Intensive stressor table:

4-core app without Wi-Fi							
Average:	CPU	%user	%nice	%system	%iowait	%steal	%idle
Average:	all	97.53	0	2.47	0	0	0
Average:	0	95.3	0	4.7	0	0	0
Average:	1	97.41	0	2.59	0	0	0
Average:	2	97.8	0	2.2	0	0	0
Average:	3	99.6	0	0.4	0	0	0

AVG		(%)
	Branch Misses	0.06
	IPC	20.32
	Overall Cache misses	7.09
L1		
	L1 iCache load misses	4.86
	L1 dCache load misses	7.07
	L1 dcache store misses	NA
L2		
	L2 load misses	8.15
	LLC store misses	0.00

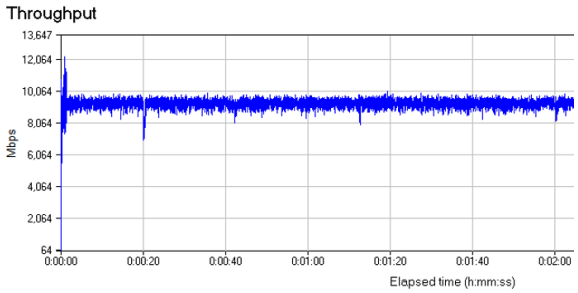
Un-intentional DDR write-backs as a % of total write-backs	99.9993888
Intentional DDR Write-backs as a % of total write-backs	0.00061124

Case 2:

Wi-Fi characterization for 4-core:

Only Wi-Fi is running on the system (i.e. no other I/O intensive applications are running on the system including the stress-ng application).

Wi-Fi Throughput Achieved: 9 Gbps



4-Core Wi-Fi performance table:

4-core Wi-Fi							
Average:	CPU	%user	%nice	%system	%iowait	%steal	%idle
Average:	all	0.08	0	54.75	0	0	45.17
Average:	0	0	0	67.44	0	0	32.56
Average:	1	0.21	0	64.89	0	0	34.9
Average:	2	0	0	46.95	0	0	53.05
Average:	3	0.1	0	40.82	0	0	59.08

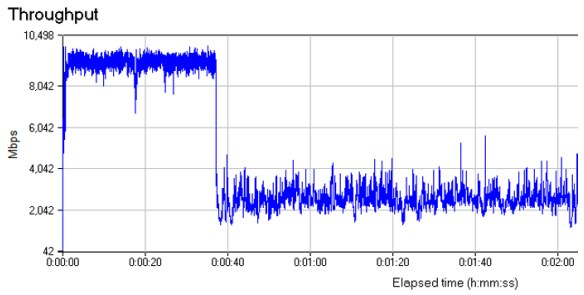
AVG	Counter	(%)
	Branch Misses	3.39
	IPC	46.04
	Overall Cache misses	2.36
L1		
	L1 iCache load misses	1.60
	L1 dCache load misses	2.25
	L1 dcache store misses	NA
L2		
	L2 load misses	9.19
	LLC store misses	7.93

write to read ratio	33.5059474
Un-intentional DDR write-backs as a % of total write-backs	77.0402877
Intentional Write-backs as a % of total write-backs	22.9597123
Invalidates as % of total reads	4.08607017

Case 3:

Both I/O Intensive Stressor and Wi-Fi are running simultaneously on the system *without* Cgroups:

Wi-Fi Throughput Achieved: ~2 Gbps (stable)



4-Core with Wi-Fi + I/O Intensive stressor table:

4-core Wi-Fi							
Average:	CPU	%user	%nice	%system	%iowait	%steal	%idle
Average:	All	40.85	0	59.13	0	0	0.02
Average:	0	34.47	0	65.43	0	0	0.1
Average:	1	38.86	0	61.14	0	0	0
Average:	2	41.82	0	58.18	0	0	0
Average:	3	48.25	0	51.75	0	0	0

AVG	Counter	(%)
	Branch Misses	0.84
	IPC	25.83
	Overall Cache misses	5.13
L1		
	L1 iCache load misses	0.31
	L1 dCache load misses	5.24
	L1 dcache store misses	NA
L2		
	L2 load misses	16.04
	LLC store misses	1.11

AVG	Counter	(%)
	Branch Misses	0.84
	IPC	25.83
	Overall Cache misses	5.13
L1		
	L1 iCache load misses	0.31
	L1 dCache load misses	5.24
	L1 dcache store misses	NA
L2		
	L2 load misses	16.04
	LLC store misses	1.11

Step #C: Finalize the recommendations:

Case 4:

Both I/O Intensive Stressor and Wi-Fi are running simultaneously on the system *with* Cgroups:

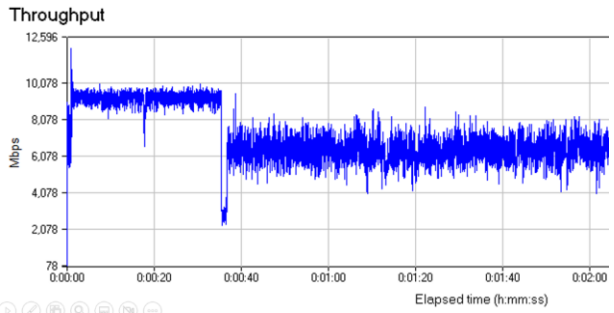
Key Observations & Analysis:

4-cores for Wi-Fi as well as other router Applications (SMP-safe applications)

I/O intensive applications would significantly dent the Wi-Fi performance (9 Gbps Vs 2 Gbps)

4-Core with Wi-Fi + I/O Intensive stressor with Cgroups cpushares = 512

Wi-Fi Throughput Achieved: 6.5 Gbps:



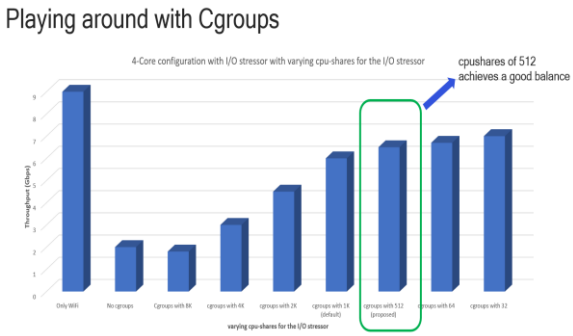
I/O Intensive Stressor with Wi-Fi with Cgroups table:

4-core WiFi with I/o stressor with cgroups cpushares = 512							
Average:	CPU	%user	%nice	%system	%iowait	%steal	%idle
Average:	all	18.67	0	81.31	0	0	0.02
Average:	0	25.32	0	74.68	0	0	0
Average:	1	16.38	0	83.62	0	0	0
Average:	2	17.17	0	82.73	0	0	0.1
Average:	3	15.78	0	84.22	0	0	0

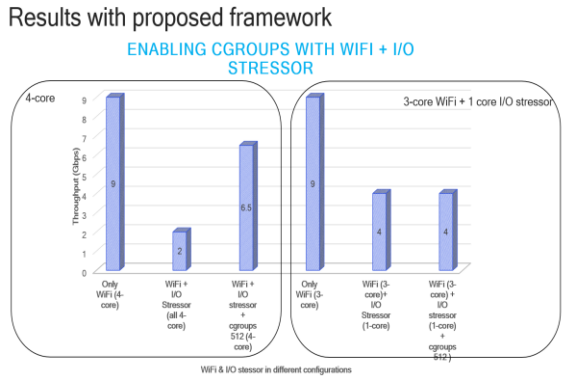
AVG	Counter	(%)
	Branch Misses	0.93
	IPC	29.40
	Overall Cache misses	4.38
L1		
	L1 iCache load misses	0.34
	L1 dCache load misses	4.37
	L1 dcache store misses	NA
L2		
	L2 load misses	22.52
	LLC store misses	2.52

Step #D: Analysis & Results:

Cgroups configuration with CPU shares option:



Results:



Test results show that Wi-Fi performance drops from 9 Gbps to as low as 2.5Gbps when an I/O intensive application is running on all the four cores along with WiFi in a SMP config.

But the Wi-Fi throughput increases from 2.5Gbps to 6.5 Gbps with the help of Cgroups.

Step #E: Conclusion:

Conclusion:

Cache Poisoning / Pollution is inevitable when there is a shared Cache. Simple L3 cache without any partitioning will not cater to the desired system behavior. Cache lines of high priority applications could get evicted by low priority.

A new resource manager framework is proposed to control the application aggressiveness and ensure that Wi-Fi performance is met. Proposed framework is based on Cgroups. It is made configurable, scalable, and highly transparent for user adoption. Users can decide application aggressiveness vs Wi-Fi throughput and tune it by themselves.

References:

Stress-ng Application References:

a) https://wiki.ubuntu.com/Kernel/Reference/stress-ng#:~:text=Stress%2Dng%20measures%20a%20stress,as%20an%20accurate%20benchmarking%20figure.

b) https://linuxfoundation.org/wp-content/uploads/Colin-Ian-King-Mentorship-Stress-ng.pdf

Cgroups References:

https://man7.org/linux/man-pages/man7/cgroups.7.html