UtilClamp usage on Android

Plans on adopting UtilClamp in Android for task utilization boosting and capping

android

Task utilization boosting / capping

Basic idea: change utilization of a task or a task group to make it look bigger (boosting) or smaller (capping) to the scheduler

Affects

- frequency selection - schedutil uses utilization signal for OPP selection
- task placement decisions - EAS uses utilization signal for task placement

Available mechanisms

Features	SchedTune	UtilClamp
Mainline support	out-of-tree custom cgroup controller	core bits upstream since v5.3 cgroups support likely queued for v5.4
APIs	**cgroup v1 only**, single level hierarchy **limited** number of **TG** supported	**cgroup v1/v2**, unlimited nested groups **unlimited** number of **TG** supported procfs based **system-wide defaults** syscall based **per-task API**
OPPs Selection Biasing	non-linear (positive) boosting difficult to tune only experimental negative boosting	simple threshold based utilization boosting/capping mechanism
Boost Holding	hardcoded 50ms	WIP: tunable timeouts
Task Placement Biasing	prefer-idle feature for latency-sensitive but not critical tasks	Initial integration with EAS WIP: “latency niceness” **[1]**
RT Tasks Support	N/A	OPP biasing WIP: capacity awareness **[2]**

JankBench comparison results

Test	SchedTune			UtilClamp		
	Mean frame time (ms)	99 percentile	Power (mW)	Mean frame time (ms)	99 percentile	Power (mW)
List View	3.30	6.89	274.3	3.58 (8.4%)	6.57(-4.6%)	292.1 (6.5%)
Image List View	3.64	7.07	305.2	3.45 (-5.1%)	6.36 (-9.9%)	310.3 (1.7%)
Shadow Grid	3.79	7.40	287.8	3.40 (-10.4%)	6.67 (-9.9%)	290.8 (1.0%)
Low Hitrate Text	3.86	7.38	293.6	3.71 (-4.0%)	6.56 (-11.1%)	302.9 (3.2%)
High Hitrate Text	3.43	6.98	285.0	3.36 (-2.0%)	6.34 (-9.2%)	293.5 (3.0%)
Edit Text	3.58	11.28	273.3	3.33 (-7.0%)	8.57 (-23.9%)	287.0 (5.0%)

Results are based on 10 runs of jankbench tests

UtilClamp used default top-app min clamp of 100, boosted top-app min clamp of 700

SchedTune sets freq floors when boosting while UtilClamp does not

UtilClamp uses shares to implement prefer-idle

Rollout plans

AOSP release and kernel requirements

Android R

Kernel v5.4

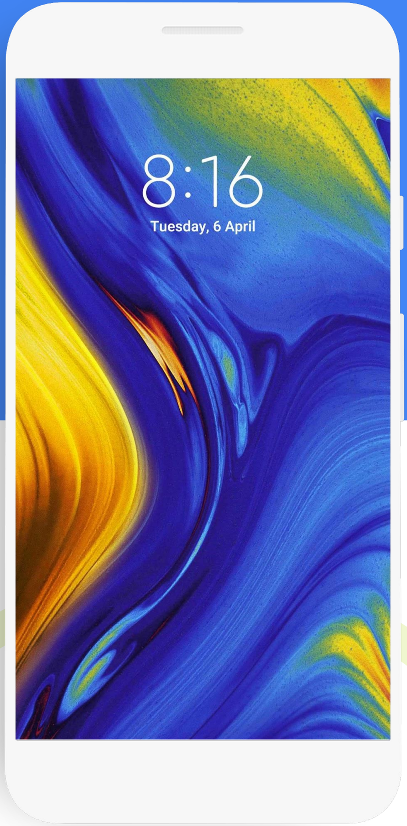
Possible backports to v4.19

Additional considerations

cgroups v2 vs v1

use of unified hierarchy

per-role vs per-app task grouping



Things to work through

Testing and Evaluation

a. cpu controller can only be enabled when all RT processes are in the root cgroup (is this really an obstacle?)

b. cgroups under cpu controller will lead to bandwidth distribution between them (need careful evaluation of possible unintended consequences)

WIP Features

a. BOOST_HOLD feature - retain max boost level from task enqueue for a minimum period
b. prefer_idle feature - bias CPU selection towards the least busy one to improve task wakeup latency
c. RT tasks placement biasing

Discussion points: prefer_idle replacement

Option 1

Use two conditions as prefer-idle hint:

a. Task is boosted (cpu.uclamp_min > 0)
b. Task is allocated high shares (cpu.shares > DEFAULT_SHARES)

If we really need also the "prefer_idle NON boosted" case maybe just add a threshold in the condition (a) above?

Option 2

Introduce new “`cpu.latency_tolerant`” property.

Migration process from SchedTune to UtilClamp

Task profile changes

- Change appropriate task profiles to use cpu.uclamp_min, cpu.uclamp_max instead of stune.boost

PowerHal hint changes

- Replace prefer_idle usage for touchboost with cpu.uclamp boosting and increased cpu.shares or a new cpu.latency_tolerant property

Init script changes

- Mount cpu instead of schedtune controller and create appropriate hierarchy
- Set default clamp values

Questions ?

android