Scheduler Fairness

Vincent Guittot LPC 2020 Scheduler MC

01101110 01101101 10110 01100101 01110 01100100 00100000 01101111 101101 00100000 01 010 01110011 001000 01101110 01100100 011011111 01101110 00100000 0111 J0011 00 01100101 01100100 00100 0 01110011 101 01110111 01101000 011

01101101 01100001 0110

01101110 01101101

10011 00100000 01110100 011011

10111 01101000 01100

11100101 01110010 011

01110101 01110010

10 01111001

01100 01101100 01111001

0101 01110101 01110010 011

0010 01101

en en 100101

11



Fairness problem description

- How to balance a use case that can't be balanced?
- When the load granularity doesn't match with the topology
 - \circ ~ Odd number of tasks on even number of cpus as an example
- Apply only when system is overloaded
 - For one sched group at least
 - Otherwise tasks have enough runtime
 - scheduling latency not taken into account



Fairness problem example

- 9 tasks on 8 CPUs
 - Can't be balanced
 - But should be fair
- Test with 9 always running rt-app tasks on 2x4 CPUs
 - Count effective work per task: number of run events
 - 16 test iterations
- Average unfairness above 20%
 - Ratio between min and max values
- One iteration with more than 40% difference
 - Can't be higher than 50%



Current rules

Migrate task with load/2 < imbalance

OR

- Any tasks when nr_balance_failed > cache_nice_tries •
- The result is an unfair decision





01 01110010

.01110 01101101

00100000 01110100 01101111 0010000

10111 01101000 01100101 011

10110 01100101 01110010 01101000 011

01101 00100000 01110100 01101111 0

01100101 01110010 0110000

100100 00100000 01101111

Fairness issues

- Simultaneous busy Load Balance (LB)
 - Smaller domain faster to pull task
 - Some CPUs fail to pull task

Decrement by 1 jiffies the busy interval

• Huge busy interval on large system

	2 x 4 cores	2 nodes x 28 cores x 4 threads
SMT		4ms: 1 tick
		128ms : 32 ticks
MC	4ms : 1 tick	112ms: 28 ticks
	128ms : 32ticks \rightarrow 64ms	3584ms : 896 ticks → 1792ms
DIE	8ms : 2 ticks	
	256ms : 64 ticks \rightarrow 128ms	
NUMA		224ms : 56 ticks
		7168ms : 1792 ticks \rightarrow 3584ms

Reduce default interval: use ilog(weight) instead of weight?

Reduce busy_factor ?



Fairness issues

- Task selection
 - o load/2
 - Any task : LB > sd->cache_nice_tries

Monitor average min_vruntime increase per sched_slice

or

01 01110010

0100000 01110100 01

Use load_avg/util_avg ratio to calculate imbalance

or

Accumulate imbalance of failed LB in shared data

or

Increase margin during task selection as nr_failed increase

- Active migration
 - LB > sd->cache_nice_tries+2

Remove this active migration use case

Test results

- Same test environnement as previous test
- Average unfairness around 8%
- One iteration with 12% difference
 - \circ $\,$ Can't be higher than 50% $\,$
- Don't solve completely the "always same task" problem
 - Only reduce the occurence



Fairness issues

- Often the same task that is pulled
 - Kind of sync between sched_slice and LB interval
- imbalance_pct threshold discards task migration
 - 25% at DIE, NUMA levels
 - Test with 11 always running rt-app tasks on 8 cpus





Thank you

An (m100101

110 01111001

11

101 01110101 01110010

10011 00100000 01110100 0110111

110111 01101000 011001

