## CPUfreq/sched & VM guest workloads problems Saravana Kannan <saravanak@google.com> David Dai <davidai@google.com>



### Continuing my LPC 2022 talk

running on the host even if you assume virtualization overhead is zero.

- Load tracking/task placement inside VM is broken because: • VM has no awareness of architecture differences between pCPUs (Eg: big/little, P/E) • VM has no awareness of pCPU min/max/current frequency • Host side CPU frequency scaling is broken because: Host has no idea of load on each vCPU.
  - Host has no idea of workload types inside vCPU (eg: io\_boost)

Workloads running inside a VM have terrible power/performance when compared to



## Plumbers Conference | Richmond, VA | Nov. 13-15, 2023 **Architecture awareness and CPU min/max frequencies**

Added crosvm support to insert the relevant DT properties:

- Add dmips-per-MHz to CPU nodes to make VM aware of CPU differences Add operating-points-v2 table to CPU nodes to make VM aware of CPU min/max
- frequencies





## Plumbers Conference | Richmond, VA | Nov. 13-15, 2023 **Current CPU frequency and load awareness**

CPU frequency MMIO device:

• Register to get vCPU's frequency – Traps to host and gets the pCPU's current frequency

used in a similar fashion and is registered with the scheduler.

• Register to set vCPU's "hardware" frequency – Traps to host and set vCPU thread's uclamp min

this register.

When the VM doesn't support virtualized arch CPU performance counters, this register is

The host is indirectly made aware of the vCPU loads by the VM cpufreq governor setting



## Does it work at a fundamental level? Synthetic workloads



#### Plumbers Conference | Richmond, VA | Nov. 13-15, 2023 Finitian Conference | Richmond, VA | Nov. 13-15, 2023

00:00:00 900 000 000	(			00:00:00 950 000 0 -180.2ms -	000			00:00:0	1 000			0 0
	 thre	three				+brog	three	three	three	three		
	unre	(III ea	(iiiea	(iiiea	unrea	(III ea		unrea	unea	unea	unrea	

ns	+78.7 ms	+98.7 ms	+118.7 ms	+138.7 ms	+1.58.7 ms	+178.7 ms	+198.7 ms	+218.7 ms	+238
162]	CLO		crosvm_v	cpu0 [7162]				crosvm_vc	pu0 (7
				Į					

00:00:01 400 000 000		00:00:01 450 000 000	183ms —		00:00:01 500 000 000			00:0 550	0:01 000 000	
m_vcpu0 [4041]										
	C Crosv C	cros crosv	CTOSV CTOSV	cro	S Crosv	CTOSV CTOS.	Cros	V	CIOS	C
		1 1 1 11 1	1 17 8 81 1 1 11	1						





#### VM after: **46ms** to hit Fmax on Little and then upmigrate to big

	347 125 000			380 000 000
1	\$ ≡			
	Cpu 0 (little)	Ŧ		
	Cpu 6 (big)	Ŧ		
	Cpu 0 Frequency	Ŧ	5 GHz	
	Cpu 6 Frequency	Ŧ	5 GHz	



Linux

## Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

#### Host: **133ms** to get to Fmax on big

02:30:12 + 611 425 000			00:00:00 900 000 000	00:0 950
≎ =				
Cpu 0 (little)	Ŧ 🗆			
Cpu 6 (big)	<b>∓</b> □		thre threa threa.	. thr
Cpu 0 Frequency		5 GHz		
Cpu 6 Frequency		5 GHz		

#### VM before: **188ms** to get to Fmax on big

2700.9 s +	4.6 s	+7.9 ms	+27.9 ms	+47.9 ms	+67.9 ms	+87.9
\$						
Сри О	* 🗆	crosvm	1_vcpu0 [1			
Cpu 7	* 🗆			CI	rosvm_vcpu1	[1072
Cpu 0 Frequency	★□ 5	GHz				
Cpu 7 Frequency	* 🗆 5	GHz				

#### VM after: **135ms** to get to Fmax on big 01:48:34 347 125 000 00:00:01 450 000 000 ≎ = crosvm\_vcpu0... **₽**□ Cpu 0 (little) **₽** Cpu 6 (big) 5 GHz Cpu 0 Frequency **₽**□ 5 GHz Cpu 6 Frequency





## New thread boost

### Host: **33ms** to Fmax on Little



83721.9 s +	3.6 s	+699.6 us	+10.7 ms	+20.7 ms	+30.7 ms	+40.7 ms	+50.7 ms	+60.7 ms	+70.7 ms	+80.7 ms	+90.7 ms	+100.7 ms	+110.7 ms	+120.7 ms	+130.7 ms	+140.7 ms	+150.7 ms	+160.7 ms	+170.7 ms
×												1		77.7113	1			1	
Cpu 0	* 🗆	C	C	U cr		crosvm_v	vcpu0 [8696]		crosvm_v	2 Cros	iv c an	crosvm_vc	1111 	Wr.	crosv	m_vcpu0 [8696]		W.C.	
Cpu 7	* 🗆		C CLOR	/m															
Cpu 0 Frequency	* 🗆	5 GHz																	
Cpu 7 Frequency	* 🗆	5 GHz																	



## uclamp aggregation issues

#### Host: **12ms** to go to Fmax

Linux

01:18:54 + 112 184 000	00:00:01 100 000 000	00:00:01 120 000 000 140 000	00:00:01 0000 160 000 000	00:00:01 180 000 000	00:00:01 200 000 000
\$ <i>=</i>					
Cpu 6 (big)	Th thread	I-1 [4 thread1-1	thread1-1 [4 thread0-0 [ threa	thre thr threa t	thread1 thread0 th
Cpu 7 (big)	Th thread	0-0 [444 thread0-0 [444			
Cpu 6 Frequency	F GHz				
Cpu 7 Frequency	F GHz				

#### VM with **default** uclamp: **66ms** to Fmax. The min is applied to rq util. Aggregation between threads is **"max of min"**

00:07:58 + 127 371 000		00:00:01 700 000 000	00:00:01 720 000 000	00:00:01 740 000 000	00:00:01 760 000 000	65 7ms	00:00:01 780 000 000	00:00:01 800 000 000
≎ =						_03.7113		
Cpu 6 (big)	Ŧ□	t t thread threa	thread0-0 th	crosv crosvm_v th	thread0-0 [3	crosvm_vc	pu6 crosvm_vc t th.	thread0 crosvm_v
Cpu 7 (big)	<b>₽</b> □	c crosvm cr	crosvm_vcpu7					
Cpu 6 Frequency	<b>F</b> 🗖 5	5 GHz				-		
Cpu 7 Frequency	<b>F</b> 🗖 5	5 GHz				J		

#### VM with additive uclamp min: 12ms to Fmax. The min is applied to the thread's util. Aggregation between threads is additive.

			applied to the thic				
00:14:41 + 285 666 000		00:00:01 380 000 000	00:00:01 400 000 000	00:00:01 420 000 000	00:00:01 440 000 000	00:00:01 460 000 000	00:00:01 480 000 000
: =				12ms			
Cpu 6 (big)	∓□	thr thread0	thread0 th	cr crosvm c threa	cr cro thread t	crosvm t thread0 d	cros cr thre
Cpu 7 (big)	Ŧ 🗖 🧧	crosvm_vcpu7	crosvm cro				
Cpu 6 Frequency	F GHz						
Cpu 7 Frequency	F GHz						

## Deeper dive: upstream patches/plans



### v1: Hypercall + util\_guest prototype



### **Pros**:

- No expensive MMIO exit to userspace required, resulting in fast performance
- Adding a new scheduler signal in the host kernel allows for more accurate/better scheduler behavior

### Cons:

1.

- Hypercalls are hypervisor specific and  ${\bullet}$ require custom implementation
- Inflexible, policy is determined by the lacksquarekernel
- Difficult to maintain due to guest ABI considerations<sup>1</sup>

**Discussion on LKML** 

## Deeper dive

### v3: MMIO + uclamp



### **Pros**:

- No host side kernel changes required.
- Simple/Generic interface between host and guest
- Host OS/Hypervisor agnostic
- Policy decided by userspace

### Cons:

- Performance suffers due to expensive context switches(High latency per exit)
- Limited by the syscall interface where some UClamp may lacks expressiveness

## Deeper dive

### v3 + eBPF



### **Pros**:

- Flexible/Portable ABIs
- Handles Guest MMIO aborts without exiting to userspace
- eBPF progs can optionally defer MMIO exits to userspace
- eBPF progs can be updated without having to update kernel/userspace

### Cons:

• eBPF permission controls are less flexible(no SECCOMP for bpf helpers)

## Real world results















### Additive uclamp

- Simple proposal: Add a sched attr flag that also applies uclamp min/max to task util.
- eBPF for virtual MMIO devices

  - Need basic support Will Deacon talked about this in KVM summit • Need to optimize it for our needs - Call BPF programs without migration disable/enable

## Discussions

```
_inux
 Plumbers
 Conference | Richmond, VA | Nov. 13-15, 2023
--- b/include/uapi/linux/sched.h
+++ b/include/uapi/linux/sched.h
#define SCHED FLAG KEEP PARAMS
                                         0x10
#define SCHED_FLAG_UTIL_CLAMP MIN
                                         0x20
#define SCHED_FLAG_UTIL_CLAMP_MAX
                                         0x40
+#define SCHED_FLAG_UTIL_CLAMP_TASK
                                         0x80
--- b/kernel/sched/fair.c
+++ b/kernel/sched/fair.c
+unsigned long uclamp task value(struct task struct *p, enum uclamp_id clamp_id)
+ {
        if (!p->uclamp req[clamp id].apply_to_task)
+
                return uclamp none(clamp id);
        return uclamp eff value(p, clamp id);
static inline unsigned long task util(struct task struct *p)
        return READ ONCE (p->se.avg.util avg);
        return clamp val(READ ONCE(p->se.avg.util avg),
                         uclamp task value (p, UCLAMP MIN),
                         uclamp task value(p, UCLAMP MAX));
 static inline unsigned long task util est(struct task struct *p)
        struct util est ue = READ ONCE(p->se.avg.util est);
        return max(ue.ewma, (ue.enqueued & ~UTIL AVG UNCHANGED));
        return clamp val(max(ue.ewma, (ue.enqueued & ~UTIL AVG UNCHANGED))
                         uclamp task value (p, UCLAMP MIN),
                         uclamp task value(p, UCLAMP MAX));
```

## additive uclamp





# Plumbers Conference

Richmond, Virginia | November 13-15, 2023

