

# Towards multithreaded TCG Alex Bennée alex.bennee@linaro.org KVM Forum 2015

# Introduction

### Hello!

- Alex Bennée
- Works for Linaro
- IRC: stsquad/ajb-linaro
- Mostly ARM emulation, a little KVM on the side
- Uses Emacs

### What is multi-threaded TCG?

### TCG?

- Tiny Code Generator
- Running non-native code on your desktop



#### Current process model

#### Processes



### How it looks

5 [  6 [       7 [   8 [   Mem[	1.3%] 0.7%] /31861MB] /77503MB]	6 : 9.3% sý: 1.3% n. 7 : 0.7% sy: 0.7% n. 8 : 0.7% sy: 0.0% n. Mem: <b>31861M</b> used:20360M Load average: 0.85 0.8	.: 0.0% hi: 0.0% si: 0.0% w .: 0.0% hi: 0.0% si: 0.0% w .: 0.0% hi: 0.0% si: 0.0% w .: 0.0% hi: 0.0% si: 0.0% w buffers:5480M cache:5058M 5 <b>0.71</b>	a: 0.0% s a: 0.7% s a: 0.0% s a: 0.7% s
PID USER PRI NI VIRT RES SHR S CPU% ME	M% TIME+	Command		
22953 alex 20 0 742M 29112 10988 S 100. 0	.1 0:09.20	/home/alex/lsrc/qemu/qemu	.git/arm-softmmu/qemu-system-	arm -machi
8949 alex 20 0 2715M 1723M 86292 S 14.5 5	.4 <b>5h</b> 57:08	/usr/lib/chromium-browse	/chromium-browsertype=rend	ererena
1860/alex 20 0 1066M 97M 7588 S 6.6 0	.3 <b>3h</b> 10:31	/usr/lib/chromium-browse	/chromium-browsertype=ppap	ichanne
29/24 root 20 0 2/764 3928 1368 S 2.0 0	.0 2h20:33	htop		
430 alex 20 0 2/644 38/2 13/2 5 2.0 0	.0 <b>In</b> 38:21	ntop		
15859 alex 20 0 27052 3388 1452 R 1.3 0 21405 alex 20 0 1254M 259M 70760 S 0 7 1	1 20.17 42	ntop (apt/google/shreme/shreme	type-renderer enable def	orrod imag
21405 dlex 20 0 1554M 556M 70760 5 0.7 1	5 9.29 24	/opt/google/chrome/chrome	type=rendererenable-der	erred-imag
23681 alex 20 0 10100 1010 23104 3 0.7 0	5 8.20.34	/opt/google/chrome/chrome	type-rendererenable-def	erred-imag
8975 alex 20 0 2386M 1160M 76724 S 0 7 3	6 1h03.53	/usr/lib/cbromium-browse	c/cbromium-browsertype=rend	erer - ena
19612 alex 20 0 50824 4316 1444 S 0 7 0	0 0.44 13	mosh-server new -s -c 250	S -1 LANG=en GB LITE-8 -1 LANGU	AGE=en GB
22381 alex 20 0 877M 138M 39428 S 0.0 0	4 7:28.00	/opt/google/chrome/chrome	etype=rendererenable-def	erred-imag
23188 alex 20 0 805M 291M 235M S 0.0 0	.9 17:09.42	/opt/google/chrome/chrome	etype=apu-processchannel	=23125.0.3
5595 alex 20 0 808M 127M 10632 S 0.0 0	.4 21:13.18	emacsdaemon		
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7N	ice - <mark>F8</mark> Nice	+ <mark>F9</mark> Kill <mark>F10</mark> Quit		
13:03 alex@zen/x86_64 [kvm-unit-tests.git/mttcg/cu Running with TCG /home/alex/lsrc/qemu/qemu.git/arm-softmmu/qemu-syste -device virtconsole,chardev=ctd -chardev testdev,id pend excl CPU1 online CPU2 online CPU2 online CPU3 online CPU3 online CPU3 online CPU3 online CPU2: Done, 10000000 incs	rrent-tests- em-arm -mach =ctd -displa	v2] >./arm/run ./arm/bar nine virt,accel=tcg -cpu a ay none -serial stdio -ke	rier-test.flat -smp 4append cortex-a15 -device virtio-seri rnel ./arm/barrier-test.flat -	al-device smp 4ap

#### Multi-threaded TCG



### Reality?

### Multithreaded programming



### Why do we want it?

### Living in a Multi-core world

#### Raspberry Pi 2

### Quad-core Cortex A7 @900Mhz

\$25

#### Dragonboard 410c



#### Quad-core Cortex A53@1.4Ghz

\$75

### Nexus 5



### Quad Core Krait 400 @ 2.26Ghz

\$339

### My Desktop



### Intel i7 (4 core + 4 hyperthreads) @ 3.4 Ghz \$600

#### **Build Server**



### 2 x Intel Xeon (6+6 hyperthreads) @ 3.46 Ghz \$2-3k

### Android Emulation



- Android emulator uses QEMU as base
- Most modern Android devices are multi-core

#### Per-core performance



via @HenkPoly

Other reasons to care

Using QEMU for System bring up

- Increasingly used for prototyping
  - new multi-core systems
  - new heterogeneous systems
- Want concurrent behaviour
  - Bad software should fail in QEMU!

As a development tool

- Instrumentation and inspection
- Record and playback
- Reverse debugging

### **Cross Tooling**

### Building often complex



http://lukeluo.blogspot.co.uk/2014/01/linux-from-scratch-forcubietruck-c4.html Just use qemu-linux-user?

- Make sure binfmt\_misc setup
- Mess around with multilib/chroots
- Hope threads/signals not used

#### Or boot a multi-core system

Quit anyway? (y or n) y 09:28 alex@zen/x86\_64 [qemu.git/mttcg/multi\_tcg\_v7@greensocs] >./arm-softmmu/qemu-system-arm -machine virt -cpu cortex-al5 -machine type=virt -display none -serial telnet:127.0.0.1:4444 -mon itor stdio -smp 4 -m 4096 -kernel ../images/aarch32-current-linux-kernel-only.img --append "console=ttyAMA0 root=/dev/vdal" -drive file=../images/jessie-arm32.qcow2.id=myblock.index=0.if=none \_\_\_\_\_\_\_device virtio-blk-device.drive=myblock -netdev user.id=unet.hostfwd=tcp::2222:22 -device virtio-net-device.netdev=unet -D /tmp/qemu.log -d int.unimp -name debug-threads=on QEMU 2.3.90 monitor - type 'help' for more information

(qemu)

CPU revision : 1		
Hardware : Generic DT based system Revision : 0000		2 3
Serial : 00000000000000		
root@debian:~# uname -a		
Linux debian 4.1.0-rc6-ajb #7 SMP Fri Jun 12 17:58:11 BST	2015 armv7l GNU/Linux	6
root@debian:~# cat /proc/cpuinfo		
processor : 0		8
model name : ARMv7 Processor rev 1 (v7l)		Mem
BogoMIPS : 125.00		Swp
Features : half thumb fastmult vfp edsp neon vfpv3	tls vfpv4 idiva idivt vfpd32 lpae evtstrm	
CPU implementer : 0x41		PID
CPU architecture: 7	25	658
CPU variant : 0x2	13	695
CPU part : 0xc0f	27	216
CPU revision : I	30	592
	12	427
processor : I	6	086
Model name : ARMV/ Processor rev I (V/L)		667
BOGOMIPS : 125.00		888
CDU implementer : 0x41	cits vipv4 idiva idivt vipd32 tpae evisirm 31	730
CPU implementer : 0x41	2	2/0
CPU variant . Av2	25	265
CPU part : 0xc0f	C	202
CPU revision 1	20	538
	12	186
processor · 2	25	532
model name : ARMv7 Processor rev 1 (v71)	12	220
BogoMIPS : 125.00	21	747
Features : half thumb fastmult vfp edsp neon vfpv3	tls vfpv4 idiva idivt vfpd32 lpae evtstrm 12	273
CPU implementer : 0x41	5	794
CPU architecture: 7	3	115
CPU variant : 0x2		240
CPU part : 0xc0f	20	268
CPU revision : 1	18	988
	24	827
processor : 3	12	618
model name : ARMv7 Processor rev 1 (v7l)	7	589
BogoMIPS : 125.00	12	551
Features : half thumb fastmult vfp edsp neon vfpv3	tls vfpv4 idiva idivt vfpd32 lpae evtstrm 31	986
CPU implementer : 0x41		962
CPU variant (Av2	د م	223
CPU part , 0x2		064
CPU rovision · 1	0	904
		082
Hardware Generic DT based system	0 F1	Hol
Revision : 0000		net
Serial : 00000000000000	- 5	
root@debian:~#		
<pre>@zen 0:Emacs 1:qemu.git 2:qemu-jessie* 3:bash- 4:kvm-un</pre>	it-tests 5:netcat:4444 6:presentations	

							~				o. oo
Ţ				39.3	3%	1 :	22	3.4%	5y: 15	5.9% n1:	0.0% h1
2				52.0	1%	2 :	20	9.0%	5y: 32	2.0% n1:	0.0% h1
5				40.4	1%] 10 <b>1</b>	3 :	11	/.1%	5y: 29	9.3% N1:	0.0% n1
4			!	51	/%]	4 :	10	0.0%	5y: 35	0.1% n1:	0.0% n1
5				/	/%]	5 :	20	9.4%	5y: 5∠	2.6% n1:	0.0% ni
0				51.4	1%	6 :	19	9.4%	5y: 31	L.9% n1:	0.0% ni
		!!!!!!		40.3	5%		14	4.1% S	sy: 26	0.2% n1:	0.0% ni
8   M				/21061	)%] 4D1	8 :	11	3.7%	y: 56	0.8% N1:	0.0% n1
mem			0217,	/31861	IR I	Mem:	315	361M (	ised:4		Ters:66
Swp			535,	///5031	1B <b>]</b>	Load	a١	/erage	9: 4.4	15 2.95 <b>I</b> .	85
PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
0658	alex	20	0	7413M	4691	11412	5	375.	1.5	12:37.51	./arm-s
3095	alex	20	0	10/419	53/M	82964	S	44.1	1./	3n10:32	/usr/ti
210	alex	20	0	83504	1/188	3480	S	9.0	0.1	Ln52:49	/USF/D1
1592 1427	alex	20	0	159219	252M	10106	S	7.0	0.8	Ln29:40	chromiu
2427	alex	20	0	9481	23211	10190	S	0.9	0.7	Ln10:47	emacs -
0000	root	20	0	27824	389Z	1300	S	3.4	0.0	<b>IN</b> 12:19	ntop
100/	atex	20	0		1.50M	21/92	2	3.4	0.4	38:10.20	/opt/go
1000	alex	20	0	MTCO	1160	25904	2	3.4	0.5	39:12.70	/opt/go
1/30	alex	20	0	422M	20604	17072	с С	2.1	0.4	22.51 65	/upr/go
2270	alax	20	0	42.511	2100	17972	2	2.1	0.1	22:31.03	/USI/DI
2005	alex	20	0	20912 1400M	2100 256M	74700	ĸ	2.1	0.0	3/:4/.20	ntop (ant/ga
2000	alex	20	11	1400M	4904	74700	S	1.4	1.1	21.46.42	/opt/go
2230	alex	9	-11	00911	4004	2990	S	1.4	0.0	51:40.45	/usr/pi
1238	alex	20	0	17520	1301	2/224	S	1.4	0.4	5:20.82	/opt/go
2100	alex	20	0	1/03060	10600	75512	S	0.7	1.0	<b>1100</b> :15	/opt/go
2220	alex	20	0	11704	40000	2004	S	0.7	0.1	0:03.33	mosync
2220	alex	20	0	1217M	406M	10020	с С	0.7	2.1	21:19.95	/opt/go
L/4/	alex	20	0	101/M	100M	14520	2	0.7	1.2	4.20 05	/opt/go
22/3	alex	20	0	1403M	651M	14520	2	0.7	2.5	4:29.00	/opt/go
)/94 )11c	atex	20	0	140311	10720	20204	2	0.7	2.0	7:55.50	/upt/go
2112	alax	20	0	122M	10/20	2049	2	0.0	0.1	2100:07	/USI/DI
3240		20	0	057M	157M	12221	2	0.0	0.1	2.44.07	(opt/go
0000		20	0	937M	105M	21//0	2	0.0	0.5	0.02 60	/opt/go
1077		20	0	029M	50276	15476	2	0.0	0.5	0.05.00	/opt/go
2618	alex	20	6	12/0M	30270 30270	28872	2	0.0	1 0	38.42 88	/opt/go
7590	alex	20	6	12490	24660	1216	2	0.0	0 1	5.11 06	tmux po
2551	alex	20	0	1386M	24000 /12M	1210 125M	с С	0.0	1 3	10.53 60	/ont/go
1086	alov	20	٥ ٥	MNND	1/0/	32280	s	0.0	0.5	0.35 08	/opt/go
962	avahi	20	0	32596	1580	1172	s S	0.0	0.5	0.05.48	avahi_d
2775	alov	20	0 0	60//8	2044	1628	c c	0.0	0.0	0.05.40	rodshif
1203	alex	20	õ	1649M	845M	67348	ŝ	0.0	27	1.58 99	/ont/go
3964	alex	20	0	957M	157M	32096	S	0.0	0.5	0:43.06	/ont/go
5895	alex	20	0	22680	3776	1700	S	0.0	0.0	0.40.00	/hin/ha
5083	root	20	0	24920	2040	1172	ŝ	0.0	0.0	3.36.17	tmux no
Help	F2	etup F3S	earcl	F4Fili	ter <mark>F5</mark> Ti	ree F	5 <b>S</b> c	ortBy	-7Nice	e - F8Nice	+F9Kill
5											
					Thu Au	.ug 13 (	99	32 (1	lavg 4	1.14, 2.72	2, 1.74)

### Things in our way

- Global State in QEMU
- Guest Memory Models

### **Global State**

- Numerous globals in TCG generation
- TCG Runtime Structures
- Device emulation structures

Guest Memory models

- Atomic behaviour
- LL/SC Semantics
- Memory barriers

# How can we do it?

### 3 broad approaches

### Use threads/locks



#### Use processes/IPC



http://ipads.se.sjtu.edu.cn/\_media/publications/coremuppopp11.pdf

### Re-write from scratch

Pros/Cons of each approach

1

Aproach	Threads/Locks	Process/IPC	<b>Re-write</b>
Pros	Performance	Correctness	Shiny and New!
Cons	Performance, Complexity	Performance, Invasive	Wasted Legacy, New problems

### What we have done

- Protected code generation
- Serialised the run loop
  - translated code multi-threaded
- New memory semantics
- Multi-threaded device emulation

### Things in our way

- Global State in QEMU
- Guest Memory Models
### Code generator globals



TCG Runtime structures

- SoftMMUTLB
- Translation Buffer Jump Cache
- Condition Variables (tcg\_halt\_cond)
- Flags (exit\_request)

- exit\_request -> cpu->exit\_request
- tcg\_halt\_cond -> cpu->halt\_cond

per-CPU variables

# Quick reminder of how TCG works

Code Generation

- target machine code
- intermediate form (TCG ops)
- generate host binary code

# Input Code

ldr	r2,	[r3]
add	r2,	r2, #1
str	r2,	[r3]
bx	lr	

#### TCG Ops

mov\_i32 tmp5,r3
qemu\_ld\_i32 tmp6,tmp5,leul,3
mov\_i32 r2,tmp6

movi\_i32 tmp5,\$0x1
mov\_i32 tmp6,r2
add\_i32 tmp6,tmp6,tmp5
mov\_i32 r2,tmp6

mov\_i32 tmp5,r3
mov\_i32 tmp6,r2
qemu\_st\_i32 tmp6,tmp5,leul,3

exit\_tb \$0x7ff368a0baab

# Output Code

mov (%rsi),%ebp
inc %ebp
mov %ebp,(%rsi)

# Basic Block



## **Block Chaining**



TCG Global State

- Code generation globals
- Global runtime

Translated code is safe

- Only accesses vCPU structures
- We need to careful leaving the translated code

Exit Destinations

- Back to Run Loop
- Helper Function



### Simplified Run Loop



#### Helper Functions



Types of Helper

- Complex Operations
  - should only touch private vCPU state
  - no locking required\*
- System Operations
  - Iocking for cross-cpu things
  - some operations affect all vCPUs

# Stop the World!

- Using locks
  - expensive for frequently read vCPU structures
  - complex when modifying multiple vCPUs data
- Ensure relevant vCPUs halted, modify at "leisure"

Deferred Work

- Existing queued\_work mechanism
  - add work to queue
  - signal vCPU to exit
- New queued\_safe\_work
  - waits for all vCPUs to halt
  - no lock held when run

TCG Summary

- Move global vars to per-CPU/Thread
  - exit and condition variables
- Make use of tb\_lock
  - uses existing TCG context tb\_lock
  - protects all code generation/patching
  - protects all manipulation of tb\_jump\_cache
- Add async safe work mechanism
  - Defer tasks until all vCPUs halted

# Things in our way

- Global State in QEMU
- Guest Memory Models

# No Atomic TCG Ops



# Atomic Behaviour is easy when Single Threaded



### Considerably harder when Multi-threaded



Load-link/Store-conditional (LL/SC)

- RISC alternative to atomic CAS
- Multi-instruction sequence
- Store only succeeds if memory not touch since link
- LL/SC can emulate other atomic operations

LL/SC in QEMU

- Introduce new TCG ops
  - qemu\_ldlink\_i32/64
  - qemu\_stcond\_i32/64
- Can be used to emulate
  - Ioad/store exclusive
  - atomic instructions

# SoftMMU

# What it does

- Maps guest loads/stores to host memory
  - uses an addend offset
- Fast path in generated code
- Slow path in C code
  - Victim cache lookup
  - Target page table walk

### How it works: Stage one



### How it works: Stage two



### How it works: Stage three



How does this help with LL/SC?

- Introduced new TCG ops
  - qemu\_ldlink\_i32/64
  - qemu\_stcond\_i32/64

Using the SoftMMU slow path we can implement the backend in a generic way

## LL/SC in Pictures



LL/SC Summary

- New TLB\_EXCL flag marks page
- All access now follows slow-path
  - trip exclusive flag
- Store conditional always slow-path
  - Will fail if flag tripped

Memory Model Summary

- Multi-threading brings a number of challenges
- New TCG ops to support atomic-like operations
- SoftMMU allows fairly efficient implementation
- Memory barriers still an issue.

# **Device Emulation**
KVM already done it ;-)

- added thread safety to a number of systems
- introduced memory API
- introduced I/O thread

TCG access to device memory

- All MMIO pages are flagged in the SoftMMU TLB
- The slowpath helper passes the access to the memory API
- The memory API defines regions of memory as:
  - lockless (the eventual driver worries about concurrency)
  - Iocked with the BQL

## Thanks KVM!

# Current state

# Performance & Demo

• Hand over to Frederic

# What's left

- LL/SC Patches
- MTTCG Patches
- Memory Barriers
- Enabling all front/back ends
- Testing & Documentation

## LL/SC Patches

- Majority of patch set independent from MTTCG
- Been through a number of review cycles
- Hope to get merged soonish now tree is open

## Who/where?

- Alvise Rigo of Virtual Open Systems
- https://git.virtualopensystems.com/dev/qemu-mt.git
- Latest branch: slowpath-for-atomic-v4-no-mttcg

Virtual Open Systems

MTTCG Patches

- Clean-up and rationlisation patches
  - starting to go into maintainer trees
- Delta to full MTTCG reducing

Who/where?

- Frederic Konrad of Greensocs
- http://git.greensocs.com/fkonrad/mttcg.git
- Latest branch: multi\_tcg\_v7



## Memory Barriers

- No code yet
- Current proposal is one (or two) barrier TCG ops
- Hard to trigger barrier issues on x86 backend

Enabling all front/back ends

- Current testing is ARM32 on x86
- Aim to enable MTTCG on all front/backends
- Front-ends need to use new TCG ops
- Back-ends need to support new TCG ops
   may require incremental updates

Testing & Documentation

- Both important for confidence in design
- Torture tests
  - hand-rolled
  - using kvm-unit-tests
- Want to have reference in docs/ on how it should work

# Questions?

# The End

# Extra Material

# Full TLB Walk Diagram



### Annotated TLB Walk Code (In)

0x40000000:	e3a00000	mov	r0, #0	; 0×0		
0x40000004:	e59f1004	ldr	r1, [pc	;, #4]	;	0x40000010

### Annotated TLB Walk Code (Ops)

---- prologue
ld\_i32 tmp5,env,\$0xfffffffffffffff
movi\_i32 tmp6,\$0x0
brcond\_i32 tmp5,tmp6,ne,\$L0

---- 0x40000000 movi\_i32 tmp5,\$0x0 mov\_i32 r0,tmp5

---- 0x40000004 movi\_i32 tmp5,\$0x4000000c movi\_i32 tmp6,\$0x4 add\_i32 tmp5,tmp5,tmp6 qemu\_ld\_i32 tmp6,tmp5,leul,1 mov\_i32 r1,tmp6

### Annotated TLB Walk Code (Opt Op)

```
OP after optimization and liveness analysis:
    ---- prologue
    ld_i32 tmp5, env, $0xfffffffffffff
    movi_i32 tmp6, $0x0
    brcond_i32 tmp5, tmp6, ne, $L0
    ---- 0x40000000
    movi_i32 r0, $0x0
    ---- 0x40000004
    movi_i32 tmp5, $0x40000010
    qemu_ld_i32 tmp6, tmp5, leul, 1 (val, addr, index, opc)
    mov_i32 r1, tmp6
```

#### Annotated TLB Walk Code (Out Asm)

prologue		
0x7fffe1ba1000:	MOV	-0xc(%r14),%ebp
0x7fffe1ba1004:	test	%ebp,%ebp
0x7fffe1ba1006:	jne	0x7fffe1ba10c9
0x400000	Θ	
0x7fffe1ba100c:	xor	%ebp,%ebp
0x7fffe1ba100e:	mov	%ebp,(%r14)
0x400000	4	
- movi_i32		
0x7fffe1ba1011:	mov	\$0x40000010,%ebp
- qemu_ld_i3	2	
0x7fffe1ba1016	mov	%rbp,%rdi - r0
0x7fffe1ba1019:	mov	%ebp,%esi - r1
<b>0x7fffe1ba101f</b> :	and	\$0xfffffc03,%esi
- index into	tlb_ta	<pre>ble[mem_index][0]+target_page</pre>
0x7fffe1ba101b:	shr	\$0x5,%rdi
0x7fffe1ba1025:	and	\$0x1fe0,%edi
0x7fffe1ba102b:	lea	0x2c18(%r14,%rdi,1),%rdi
0x7fffe1ba1033:	cmp	(%rdi),%esi
0x7fffe1ba1035:	mov	%ebp,%esi
0x7fffe1ba1037:	jne	0x7fffe1ba111b
offset to	"host a	ddress"
0x7fffe1ba103d:	add	0x10(%rdi),%rsi
actual loa	d	

0x7fffe1ba1041: mov (%rsi),%ebp --- mov\_i32 r1, tmp6 0x7fffe1ba1043: mov %ebp,0x4(%r14)

slow pat	h funct:	ion call	
0x7fffe1ba111b:	mov	%r14,%rdi	
0x7fffe1ba111e:	mov	\$0x21,%edx	
0x7fffe1ba1123:	lea	-0xe7(%rip),%rcx	<pre># 0x7fffe1ba1043</pre>
0x7fffe1ba112a:	mov	\$0x555555653980,%r10	<pre># helper_le_ldul_mmu</pre>
0x7fffe1ba1134:	callq	*%r10	
0x7fffe1ba1137:	mov	%eax,%ebp	
0x7fffe1ba1139:	jmpq	0x7fffe1ba1043	

## Locking in run loop



	flush queued safe work
	flush queued work
	husn_queueu_work
'	



SoftMMU Slowpath Reasons

- Missing mapping
  - first access (fill)
  - crossed target page (refill)
- Mapping invalidated
- Page not dirty
- Page is MMIO