

# Mesos on ARM

---

鲜卑拓拔枫

[XianBei2011@gmail.com](mailto:XianBei2011@gmail.com)

Apr 3, 2017

<b>Revision</b>	<b>Authors</b>	<b>Remarks</b>
<b>v1.0</b> <b>Nov 17, 2016</b>	<b>Koo Li</b>	<b>Initial Version for MesosCon Asis 2016</b>
<b>v1.0.3</b> <b>Apr 3, 2017</b>	<b>Koo Li</b>	<b>Init commit to github</b>

# Agenda

---

## **I. Background Information**

- ARM Ecosystem Today
- Raspberry Pi

## **II. Build Mesos for ARM**

- Cross Compiling
- Native Compilation
- Build Mesos with Ninja
- Summary

## **III. Clang/LLVM-based Optimization**

- Introduction
- LLD & MCLinker
- Build LLVM/Clang on RPi3 with LLD enabled
- Upcoming LLVM/Clang 4.x for Mesos
- Summary

## **IV. Virtualization-assisted Development**

- Ubuntu/Fedora ARM64
- CoreOS ARM64

## **V. Wrap-up**

---

# I. Background Information

## 1) ARM Ecosystem Today

- [https://en.wikipedia.org/wiki/ARM\\_architecture](https://en.wikipedia.org/wiki/ARM_architecture)
- <https://www.arm.com/>

### ARM at a Glance



**1,379**

Cumulative licenses signed

ARM develops and licenses technology that is at the heart of many digital devices, from sensors to smartphones and servers. Every licence signed represents the opportunity for future royalty streams, which can extend to over 25 years.



**14.9 billion**

ARM-based chips

ARM's partners shipped 14.9 billion ARM-based chips in 2015. These smart chips help make consumer electronics easier to use and more immersive, and enterprise equipment more capable and energy-efficient.



**50%**

ARMv8 in smartphones

Today, ARM-based application processors can be found in about 85% of the world's mobile devices, including smartphones, tablets and laptops. Over 50% of smartphones shipping today contain processors that implement the latest ARMv8-A architecture.



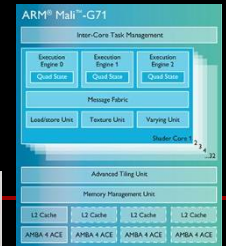
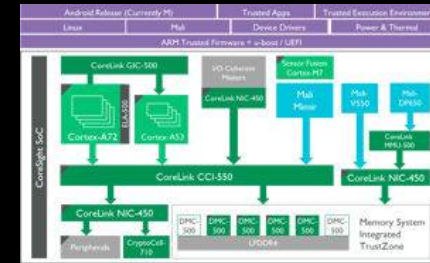
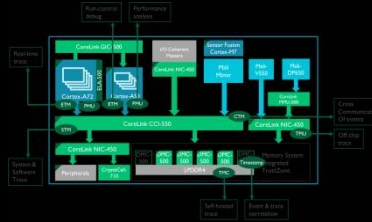
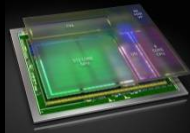
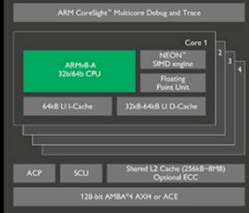
**4,500**

Patents owned or pending

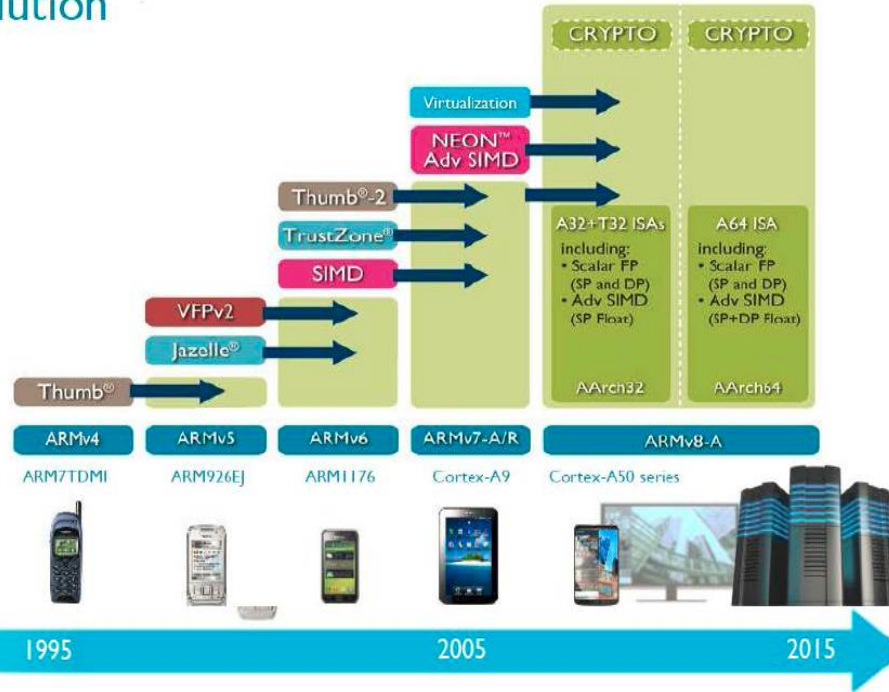
ARM filed 242 patents in 2015, taking the total number of owned or pending patents to more than 4,500.

# Leading HW/SW Technologies

## ARM Cortex®-A73



## Architecture Evolution



Increasing SoC complexity  
Increasing OS complexity  
Increasing choice of HW and SW

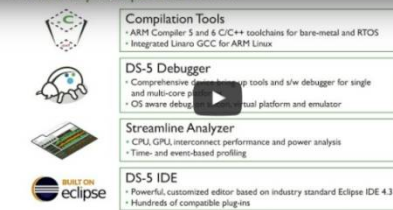
1995

2005

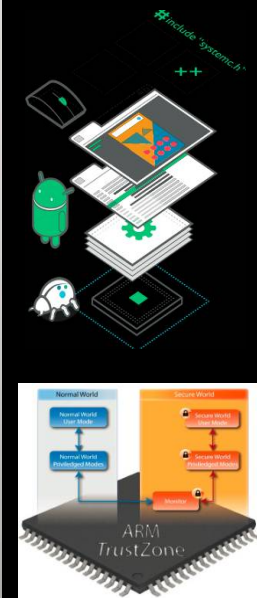
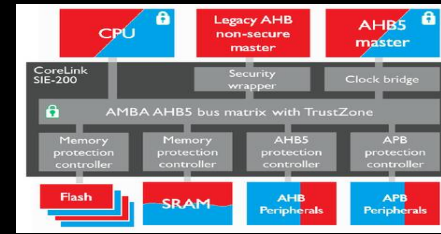
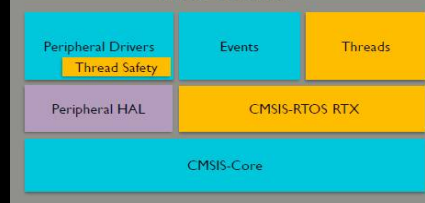
2015



## ARM DS-5 Development Studio Overview



## mbed OS Core



# Linaro

■ <http://www.linaro.org/>



**LITE**  
IoT/EMBEDDED



**LMG**  
Mobile



**LHG**  
Digital Home



**LNG**  
Networking



**LEG**  
Server

**ACPICA**  
ACPI COMPONENT ARCHITECTURE



**ceph**



**CentOS**

**debian**



**docker**

**hadoop**



**HHVM**



**kubernetes**



**KVM**



**mongoDB**

**NGINX**

**OpenJDK**



**redhat**

**Spark**



## Linaro

"The ARM situation has just improved tremendously over the last several years. It used to be a major pain to me, it has gone to almost being entirely painless."



- Linus Torvalds  
May 2015

Note:  
Linus Torvalds image from Linux Foundation, icons made by Freepik. Linus & trademarks remain the property of their respective owners and represent a range of products and services supported by Linaro.

### Celebrating 5 years of Open Source Engineering on ARM

**24TB**

data from June 2014 - May 2015  
615,000 downloads from >100 countries



**16 Connects**  
14 Cities on 3 continents

**32 member companies**  
Six members at launch



**4,410**  
gallons consumed at  
Linaro Connects

**1,141,014**

minutes of videos showing demos, talks and  
training sessions watched

More than **220 Engineers**  
from seed of twenty

**#3**

company contributor for  
Linux Kernels 3.11 - 3.18

[www.linaro.org](http://www.linaro.org)  
[www.96boards.org](http://www.96boards.org)

**11,589**  
patches upstream  
since 2011

**50,217**  
Wiki pages

**>1 Million**  
website users

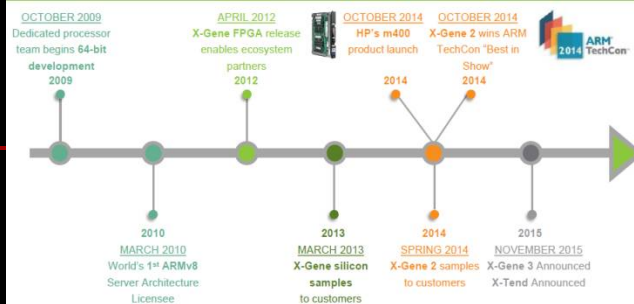
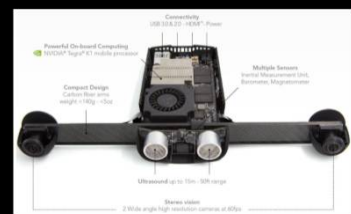


**~20**  
hardware  
platforms

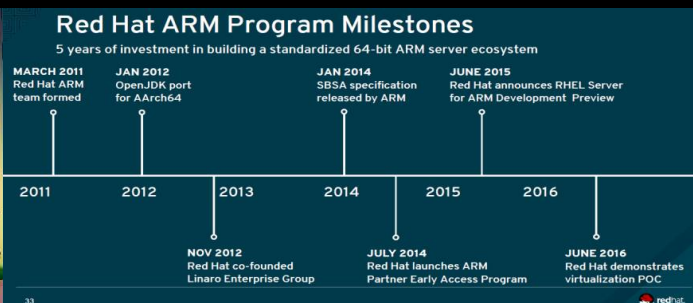
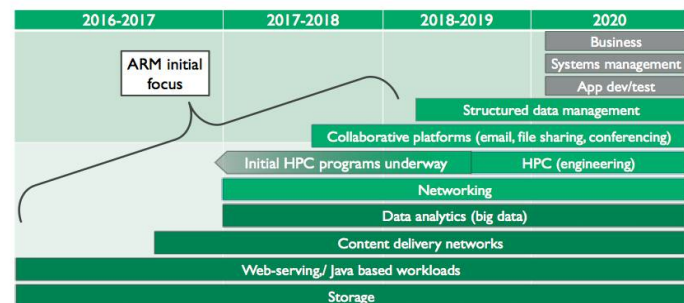




# ARM-based Solutions



## Applicability of ARM for server workloads





## 2) Raspberry Pi

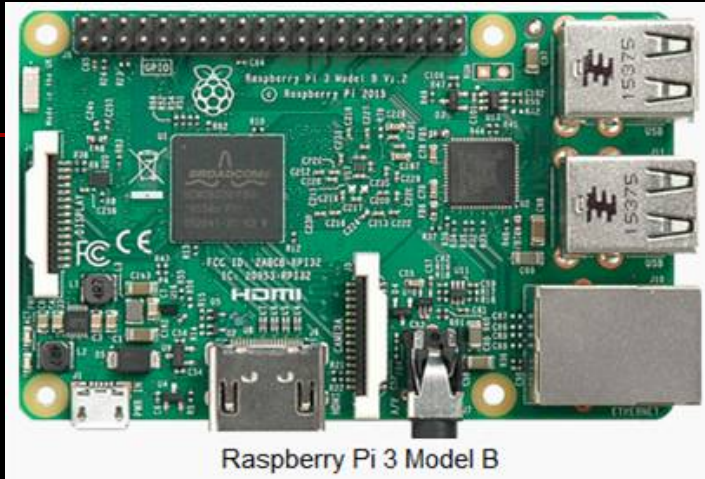
■ [https://en.wikipedia.org/wiki/Raspberry\\_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi)

■ <https://www.raspberrypi.org/>

### HW Spec

Variant	Raspberry Pi 1				Raspberry Pi 2	Raspberry Pi 3	Compute Module*	Raspberry Pi Zero	
Model	Model A	Model A+	Model B	Model B+	Model B	Model B	N/A	PCB v1.2	PCB v1.3
Release date	February 2012	November 2014 <sup>[39]</sup>	April–June 2012	July 2014 <sup>[40]</sup>	February 2015 <sup>[13]</sup>	February 2016 <sup>[14]</sup>	April 2014 <sup>[41]</sup>	November 2015 <sup>[42]</sup>	May 2016
Target price	US\$25	US\$20 <sup>[43]</sup>	US\$35 <sup>[44]</sup>	US\$25 <sup>[45]</sup>	US\$35	US\$35	US\$30 (in batches of 100) <sup>[41]</sup>	US\$5 <sup>[42]</sup>	US\$5
Architecture	ARMv6 (32-bit)				ARMv7 (32-bit)	ARMv8 (64/32-bit)	ARMv6 (32-bit)		
SoC	Broadcom BCM2835 <sup>[11]</sup>				Broadcom BCM2836	Broadcom BCM2837	Broadcom BCM2835 <sup>[41]</sup>		
CPU	700 MHz single-core ARM1176JZF-S <sup>[11]</sup>				900 MHz 32-bit quad-core ARM Cortex-A7	1.2 GHz 64-bit quad-core ARM Cortex-A53	700 MHz single-core ARM1176JZF-S	1 GHz ARM1176JZF-S single-core <sup>[42]</sup>	
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz) <sup>[46][47]</sup> OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 24.8 GFLOPS) MPEG-2 and VC-1 (with license), <sup>[48]</sup> 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder <sup>[11]</sup>							BCM2837: 1080p60)	
Memory (SDRAM)	256 MB (shared with GPU)	512 MB (shared with GPU) as of 4 May 2016. Older boards had 256 MB (shared with GPU) <sup>[49]</sup>			1 GB (shared with GPU)		512 MB (shared with GPU)		
USB 2.0 ports <sup>[30]</sup>	1 (direct from BCM2835 chip)		2 (via the on-board 3-port USB hub <sup>[50]</sup> )	4 (via the on-board 5-port USB hub) <sup>[29][40]</sup>			1 (direct from BCM2835 chip)	1 Micro-USB (direct from BCM2835 chip)	

## RPi3 Model B



## Limitations

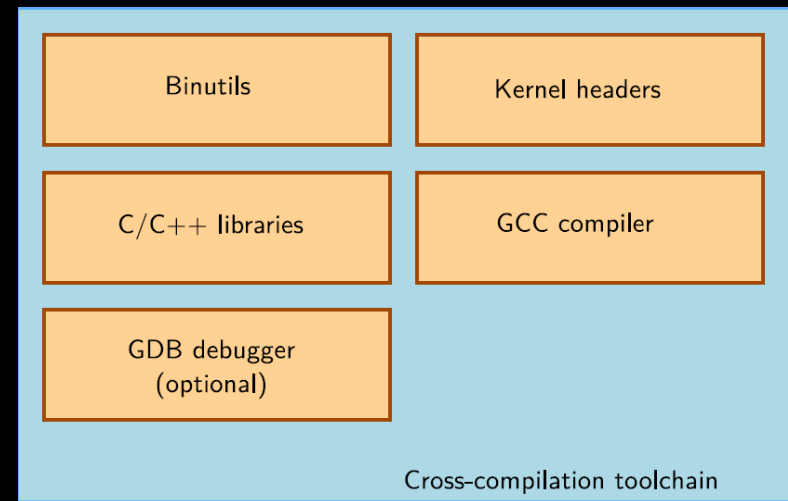
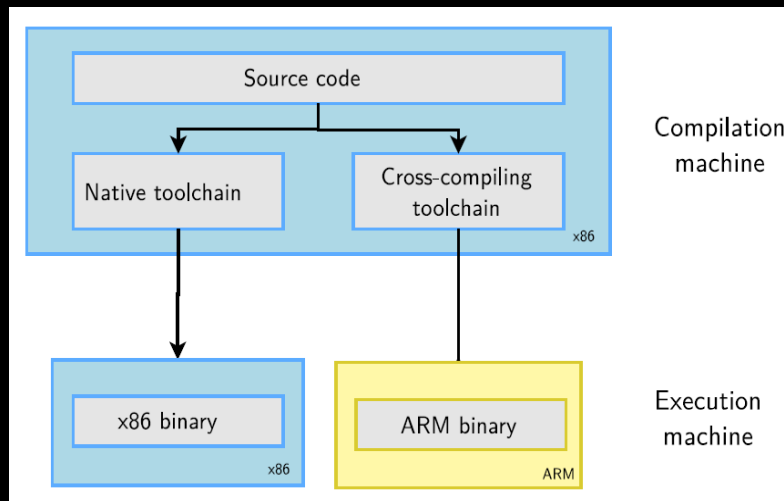
- 1) 1GB LPDDR2 RAM @900MHz
- 2) Official release (Raspbian with Linux Kernel 4.4 currently) does not support AArch64

```
pi@raspberrypi:/opt/MyWorkSpace/Mesos/mesos $ cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 4 (v7l)
BogoMIPS      : 76.80
Features       : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xd03
CPU revision   : 4
```

# II. Build Mesos for ARM

## 1) Cross Compiling

- when memory or storage on the target system is too restricted or target system is much slower than host system
- While set up the right cross-compiling environment for target system is not an easy thing

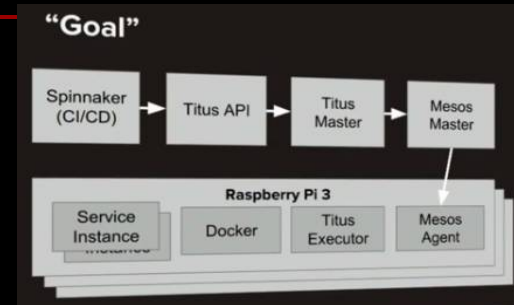


Source: <http://free-electrons.com/doc/training/embedded-linux>

## 2) Native Compilation

### Successful Cases

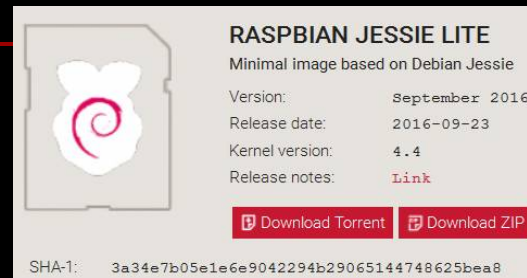
- <https://github.com/Netflix-Skunkworks/mesos-on-pi>  
Built Mesos 0.24.1 natively based on older guides



- <http://likemagicappears.com/projects/raspberry-pi-cluster/mesos-on-raspbian/>
  - <http://blog.haosdent.me/2016/04/24/mesos-on-arm/>
  - <http://a.frtzlr.com/running-apache-mesos-on-a-raspberry-pi-2/>
- Limitations: use low GCC version (< 6) and Mesos version (< 1.0), disable Java/Python for some cases**
- <https://www.tapataalk.com/topic/63351-odroid-forum-hardkernel/22544-guide-how-to-build-a-mesos-cluster-on-odroid-c2> (Mesos 1.0.0-rc2)  
**meveric's debian jessie (ARM64) image, ODROID-C2 has Amlogic S905 2Ghz 4 core CPU and 2GB 32bit DDR3 @912MHz**

## My Attempt

- Step 1. install Raspbian Lite distribution(no GUI) and upgrade it to “Stretch” with GCC(arm-linux-gnueabi) 6.2.0



```
pi@raspberrypi:~ $ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	925M	39M	746M	12M	139M	830M
Swap:	12G	0B	12G			

```
deb http://archive.raspbian.org/raspbian/ stretch main contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://archive.raspbian.org/raspbian/ stretch main contrib non-free rpi
~
configure: error: GCC 4.8 or higher required (found 4.7.3)

if test "x$is_ge_gxx48" != "xyes"; then
# GCC < 4.8 is not supported.
as_fn_error $? "GCC 4.8 or higher required (found $ax_cv_cxx_compiler_version)" "$LINENO" 5
fi
```

<https://github.com/apache/mesos/archive/mesos-1.1.0.tar.gz> (with cmake)

<http://www.apache.org/dist/mesos/1.1.0/mesos-1.1.0.tar.gz> (no cmake)



- **Step 2. install prerequisite libraries and utilities**  
**zlib1g-dev libcurl4-nss-dev libsasl2-dev libsasl2-modules**  
**libapr1-dev libaprutil1-dev libsvn-dev...**  
**cmake ninja-build autoconf automake libtool...**
- 

- **Step 3. Set up the build environment**  
**switching linker**

If you are building LLVM/Clang on an ARM board with 1G of memory or less, please use gold rather than GNU ld. In any case it is probably a good idea to set up a swap partition, too.

**sudo ln -sf /usr/bin/ld.gold /usr/bin/ld**

```
pi@raspberrypi:/usr/bin $ ll |grep ld
lrwxrwxrwx 1 root root      6 Oct 17 11:12 arm-linux-gnueabi-hf-ld -> ld.bfd*
-rwxr-xr-x 1 root root 506916 Oct 17 11:12 arm-linux-gnueabi-hf-ld.bfd*
-rwxr-xr-x 1 root root 4269876 Oct 17 11:12 arm-linux-gnueabi-hf-ld.gold*
-rwxr-xr-x 1 root root   6275 Jul 31 10:57 dpkg-buildflags*
-rwxr-xr-x 1 root root  24177 Jul 31 10:57 dpkg-buildpackage*
-rwxr-xr-x 1 root root   7503 Jul 31 10:57 dpkg-checkbuilddeps*
-rwxr-xr-x 1 root root  26384 Feb 27 2016 fold*
lrwxrwxrwx 1 root root      7 Oct 17 11:12 gold -> ld.gold*
-rwxr-xr-x 1 root root  22352 Oct  1 11:41 gtk-builder-tool*
lrwxrwxrwx 1 root root      7 Nov 11 20:35 ld -> ld.gold*
lrwxrwxrwx 1 root root      26 Oct 17 11:12 ld.bfd -> arm-linux-gnueabi-hf-ld.bfd*
-rwxr-xr-x 1 root root   5312 Oct 26 00:35 ldd*
lrwxrwxrwx 1 root root      27 Nov 11 20:34 ld.gold -> arm-linux-gnueabi-hf-ld.gold*
-rwxr-xr-x 1 root root   125 Sep 25 20:22 perldoc*
-rwxr-xr-x 1 root root 14008 Oct 26 00:35 pldd*
lrwxrwxrwx 1 root root      26 Aug 17 22:27 pybuild -> ../share/dh-python/pybuild*
```



## Make a Big SWAP File (6~16G)

<https://www.cyberciti.biz/faq/linux-add-a-swap-file-howto/>

```
pi@raspberrypi:~ $ free -h
```


	total	used	free	shared	buff/cache	available
Mem:	925M	39M	746M	12M	139M	830M
Swap:	12G	0B	12G			

### ■ Step 4. build

`cd $MESOS-1.1-0_HOME/build`

`cmake ..`

`make`



```
diff --git a/cmake/CompilationConfigure.cmake b/cmake/CompilationConfigure.cmake
index 11a8507..7c82bca 100644
--- a/cmake/CompilationConfigure.cmake
+++ b/cmake/CompilationConfigure.cmake
@@ -21,7 +21,7 @@ option(BUILD_SHARED_LIBS "Build shared libraries (DLLs)." FALSE)
string(COMPARE EQUAL ${CMAKE_SYSTEM_NAME} "Linux" LINUX)

# Check that we are targeting a 64-bit architecture.
-if (NOT (CMAKE_SIZEOF_VOID_P EQUAL 8))
+if (NOT (CMAKE_SIZEOF_VOID_P EQUAL 4))
  message(
    FATAL_ERROR
    "Mesos requires that we compile to a 64-bit target. Following are some "
@@ -32,7 +32,7 @@ if (NOT (CMAKE_SIZEOF_VOID_P EQUAL 8))
    "  cmake -G \"Visual Studio 10 Win64\".\n"
    "  * OS X: add 'x86_64' to the 'CMAKE_OSX_ARCHITECTURES':\n"
    "  cmake -DCMAKE_OSX_ARCHITECTURES=x86_64.\n"
  -endif (NOT (CMAKE_SIZEOF_VOID_P EQUAL 8))
  +endif (NOT (CMAKE_SIZEOF_VOID_P EQUAL 4))

  if (DEBUG)
    set(CMAKE_BUILD_TYPE Debug)
```

**Note:** you can speed up the compiling process by using “make -jN” with  $1 < N \leq 4$  but you might run into an out of memory situation that way.

- Unfortunately, it always got failed after building 7~9 hours for lack of memory -- either use cmake or Autotools, no matter use ld or gold linker, and though Java/Python is disabled...

```
from /opt/MyWorkSpace/Mesos/mesos-1.1.0/include/mesos/type_utils.hpp:24,  
from /opt/MyWorkSpace/Mesos/mesos-1.1.0/src/cli/execute.cpp:21:  
/usr/include/c++/6/bits/unique_ptr.h:49:28: note: declared here  
    template<typename> class auto_ptr;  
                                ^~~~~~  
/usr/bin/ld: fatal error: ../mesos-execute: mmap: failed to allocate 393159524 bytes for output file: Cannot  
allocate memory  
collect2: error: ld returned 1 exit status  
make[2]: *** [src/mesos-execute] Error 1  
make[1]: *** [src/cli/CMakeFiles/mesos-execute.dir/all] Error 2  
make: *** [all] Error 2  
icojson-1.3.0 -I/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/protobuf-2.6.1/src/protobuf-2.6.1-lib/lib/  
include -I/usr/include/subversion-1 -I/opt/MyWorkSpace/Mesos/mesos-1.1.0/src/src -I/opt/MyWorkSpace/Mesos/me  
sos-1.1.0/3rdparty/libprocess/include -I/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/http_parser-2.6.2/  
src/http_parser-2.6.2 -I/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/libev-4.22/src/libev-4.22 -I/opt/M  
yWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/zookeeper-3.4.8/src/zookeeper-3.4.8/src/c/include -I/opt/MyWorkS  
pace/Mesos/mesos-1.1.0/build/3rdparty/zookeeper-3.4.8/src/zookeeper-3.4.8/src/c/generated -I/opt/MyWorkSpace  
/Mesos/mesos-1.1.0/build/3rdparty/leveldb-1.4/src/leveldb-1.4/include -std=c++11 -g -o CMakeFiles/mesos-e  
xecute.dir/execute.cpp.o -c /opt/MyWorkSpace/Mesos/mesos-1.1.0/src/cli/execute.cpp  
2016-11-12 12:29:22 [ 92%] Linking CXX executable ../mesos-execute  
2016-11-12 12:29:22 cd /opt/MyWorkSpace/Mesos/mesos-1.1.0/build/src/cli && /usr/bin/cmake -E cmake_link_scri  
pt CMakeFiles/mesos-execute.dir/link.txt --verbose=1  
2016-11-12 12:29:22 /usr/bin/c++ -std=c++11 -g CMakeFiles/mesos-execute.dir/execute.cpp.o -o ../mesos-  
execute -L/usr/lib/arm-linux-gnueabi/libapr-1.so -L/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/glo  
g-0.3.3/src/glog-0.3.3-lib/lib/lib -L/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/3rdparty/protobuf-2.6.1/src/p  
rotobuf-2.6.1-lib/lib/lib -L/usr/lib/arm-linux-gnueabi/libsvn_delta-1.so -L/usr/lib/arm-linux-gnueabi/
```

or

```
/usr/bin/ld: failed to set dynamic section sizes: Memory exhausted  
collect2: error: ld returned 1 exit status  
make[2]: *** [libmesos.la] Error 1  
make[1]: *** [all] Error 2  
make: *** [all-recursive] Error 1
```

### 3) Build Mesos with Ninja

■ <https://ninja-build.org/>  
**MESOS-5656**

**cd \$MESOS\_HOME/build**

**cmake -G Ninja ../**

```
3rdparty/  
build.ninja  
CMakeCache.txt  
CMakeFiles/  
cmake_install.cmake  
CTestTestfile.cmake  
include/  
rules.ninja  
src/
```

pi@raspberrypi:/opt/MyWorkSpace/Mesos/mesos-master/build \$ ninja

ninja: error: '3rdparty/zookeeper-3.4.8/src/zookeeper-3.4.8/src/c/lib/libzookeeper\_mt.a', needed by 'src/mesos-execute', missing and no known rule to make it

**#official patch: [https://reviews.apache.org/r/52690/diff/1#index\\_header](https://reviews.apache.org/r/52690/diff/1#index_header)**

3rdparty/CMakeLists.txt	
Revision 770a3828a6e0ffaa4f185392fdc1a2152446449d	
New Change	
+ 418 lines	
419	PATCH_COMMAND \${ZOOKEEPER_PATCH_CMD}
420	CONFIGURE_COMMAND \${ZOOKEEPER_CONFIG_CMD}
421	BUILD_COMMAND \${ZOOKEEPER_BUILD_CMD}
422	INSTALL_COMMAND \${ZOOKEEPER_INSTALL_CMD}
423	URL \${ZOOKEEPER_URL}
424	)
425	
426	if (NOT WIN32)
427	set(LEVELDB_CONFIG_CMD cd \$(LEVELDB_ROOT) && ./configure --prefix=\$(LEVELDB_ROOT)-lib)
428	set(LEVELDB_BUILD_CMD cd \$(LEVELDB_ROOT) && make)
429	
430	endif (NOT WIN32)
431	endif (NOT WIN32)
432	endif (NOT WIN32)
433	PATCH_COMMAND \${LEVELDB_PATCH_CMD}
438	CONFIGURE_COMMAND \${CHAKE_NOOP}
439	BUILD_COMMAND \${LEVELDB_BUILD_CMD}
440	INSTALL_COMMAND \${CHAKE_NOOP}
441	URL \${LEVELDB_URL}
442	)
443	endif (NOT WIN32)
419	PATCH_COMMAND \${ZOOKEEPER_PATCH_CMD}
420	CONFIGURE_COMMAND \${ZOOKEEPER_CONFIG_CMD}
421	BUILD_COMMAND \${ZOOKEEPER_BUILD_CMD}
422	INSTALL_COMMAND \${ZOOKEEPER_INSTALL_CMD}
423	URL \${ZOOKEEPER_URL}
424	BUILD_BYPRODUCTS \${ZOOKEEPER_LIB}/lib/libzookeeper_mt.a
425	)
426	
427	if (NOT WIN32)
428	set(LEVELDB_CONFIG_CMD cd \$(LEVELDB_ROOT) && ./configure --prefix=\$(LEVELDB_ROOT)-lib)
429	set(LEVELDB_BUILD_CMD cd \$(LEVELDB_ROOT) && make)
430	
431	endif (NOT WIN32)
432	endif (NOT WIN32)
433	PATCH_COMMAND \${LEVELDB_PATCH_CMD}
438	CONFIGURE_COMMAND \${CHAKE_NOOP}
439	BUILD_COMMAND \${LEVELDB_BUILD_CMD}
440	INSTALL_COMMAND \${CHAKE_NOOP}
441	URL \${LEVELDB_URL}
442	BUILD_BYPRODUCTS \${LEVELDB_ROOT}/lib/leveldb.a
443	)
444	
445	endif (NOT WIN32)

```
#my workaround: do not change current Mesos code
git clone https://github.com/google/kati $KATI_HOME
cd $KATI_HOME; make
cd $MESOS_HOME/build
cmake ../
$KATI_HOME/m2n --kati_stats
./ninja.sh
```

---

- **Use Ninja to build Mesos 1.1.0 ~10 hours (without error) and stop it, cannot wait for build to end...**

```
pi@raspberrypi:/opt/MyWorkSpace/Mesos/mesos-1.1.0/build/src $ ll
total 1254564
drwxr-xr-x 13 pi pi      4096 Nov 13 21:11 ./
drwxr-xr-x  7 pi pi      4096 Nov 13 11:59 ../
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 cli/
drwxr-xr-x  4 pi pi      4096 Nov 13 11:59 CMakeFiles/
-rw-r--r--  1 pi pi     1994 Nov 13 11:59 cmake_install.cmake
-rw-r--r--  1 pi pi       466 Nov 13 11:59 CTestTestfile.cmake
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 docker/
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 launcher/
-rw-r--r--  1 pi pi 1164509022 Nov 13 21:04 libmesos-1.1.0.a
-rw-r--r--  1 pi pi 48385064 Nov 13 13:30 libmesos-protobufs.a
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 local/
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 log/
-rw-r--r--  1 pi pi    281490 Nov 13 11:59 Makefile
drwxr-xr-x  3 pi pi      4096 Nov 13 13:15 master/
-rwxr-xr-x  1 pi pi 71427560 Nov 13 21:12 mesos-docker-executor*
drwxr-xr-x  2 pi pi      4096 Nov 13 13:15 messages/
drwxr-xr-x  6 pi pi      4096 Nov 13 11:59 slave/
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 tests/
drwxr-xr-x  3 pi pi      4096 Nov 13 11:59 usage/
```

there is something wrong with current  
cmake build files in Mesos which will  
generate big binary executable files

## 4) Summary

- **Memory** is the most important thing to natively build big software project like Mesos on resource constrained ARM devices
- 
- A lot of performance hidden in linker
  - A better C++ template compiler is needed?
  - Modern build systems like **Ninja**, **Shake**, **Buck**, **FASTBuild**, **Meson**, and **llbuild** is worth trying

# III. Clang/LLVM-based Optimization

## 1) Introduction

■ <https://en.wikipedia.org/wiki/LLVM>

■ <http://clang.llvm.org/>

### GCC vs LLVM



GPL v3	UIUC, MIT
Front-end: CC1 / CPP	Front-end: Clang
ld.bfd / ld.gold	lld / mclinker
gdb	lldb
as / objdump	MC layer
libstdc++	libc++
libsupc++	libc++abi
libgcc	libcompiler-rt
libgccjit	libLLVMMCJIT

How is LLVM being used today?

XCode, Swift

FreeBSD, OpenMandriva Lx

Android

Debian experimenting with Clang as an additional compiler

...

### Clang Goals

- GCC compatibility
- Fast compilation and low memory footprints
- Can reduce the linking time
- User friendly diagnostics
- Tooling
  - static analyzers
  - sanitizers





## 2) LLD & MCLinker

■ <http://lld.llvm.org/>

- Key design choices
  - Do not abstract file formats (c.f. BFD)
  - Emphasis on performance at the high-level, do minimal amount as late as possible.
  - Have a similar interface to existing system linkers but simplify where possible

### **ELF Linker a drop in replacement for GNU ld**

- Support for AArch64, amd64, ARM (sort of), Mips, Power, X86 targets
- Focused on Linux and BSD like ELF files suitable for demand paging
- FreeBSD team aiming at using it for AARCH64 lld
- Many ld command line options supported
- Linker script support in active development
- Not ready for being a callable library yet

**Source:** <http://connect.linaro.org/resource/las16/las16-414>

■ <https://github.com/mclinker/mclinker>

- A system linker
- GNU ld options compatibility
- Support cross linking
- Can be used as a library or a stand-alone tool
- Support multiple targets
- Fast, small with low memory footprint

**Design for on-device linking**

**A candidate linker of Android and BSD standard systems**

### 3) Build Clang/LLVM on RPi3 with LLD enabled

#### Build Mesos with Clang/LLVM for x64

##### //configure.ac

```
# Default to gcc toolchain (we rely on some atomic builtins for now,  
# that are also present with clang).  
AC_PROG_CXX([clang++-3.9])  
AC_PROG_CC([clang-3.9])
```

or

```
CC='clang-3.9' CXX='clang++-3.9' cmake -G Ninja -DCMAKE_INSTALL_PREFIX=./Install ..
```

or

```
cmake -G Ninja -DCMAKE_TOOLCHAIN_FILE=./Toolchain_Clang-LLVM-X64_For_Mesos.cmake ..
```

- [Clang for x86\\_64 Ubuntu 14.04 \(.sig\)](#)
- [Clang for x86\\_64 Ubuntu 16.04 \(.sig\)](#)
- [Clang for x86\\_64 Debian 8 \(.sig\)](#)

##### //Toolchain\_Clang-LLVM-X64\_For\_Mesos.cmake

```
set(CMAKE_SYSTEM_NAME Linux)
```

```
set(CMAKE_SYSTEM_PROCESSOR amd64)
```

```
set(tools /opt/DevSW/Toolchain/LLVM/clang-llvm-3.9.0-x86_64-linux-gnu-ubuntu-16.04)
```

```
set(CMAKE_C_COMPILER ${tools}/bin/clang)
```

```
set(CMAKE_CXX_COMPILER ${tools}/bin/clang++)
```

## ■ Virtual Machine with 16 vCPU, 128G RAM, Ubuntu Desktop64 16.10

ninja -j 16 (GCC 6.2.0)  
(with MESOS-5656 patch)

```
4096 Nov 14 00:50 ./
4096 Nov 14 00:31 ../
4096 Nov 14 00:31 cli/
4096 Nov 14 00:31 CMakeFiles/
2906 Nov 14 00:31 cmake_install.cmake
618 Nov 14 00:31 CTestTestfile.cmake
4096 Nov 14 00:31 docker/
4096 Nov 14 00:31 launcher/
1202872966 Nov 14 00:49 libmesos-1.1.0.a
53459766 Nov 14 00:42 libmesos-protobufs.a
4096 Nov 14 00:31 local/
4096 Nov 14 00:31 log/
4096 Nov 14 00:42 master/
281947616 Nov 14 00:51 mesos-agent*
58756112 Nov 14 00:49 mesos-containerizer*
72164976 Nov 14 00:50 mesos-docker-executor*
400246560 Nov 14 00:51 mesos-execute*
85058408 Nov 14 00:50 mesos-executor*
55263624 Nov 14 00:50 mesos-fetcher*
391281832 Nov 14 00:51 mesos-local*
73010432 Nov 14 00:49 mesos-log*
32512080 Nov 14 00:49 mesos-logrotate-logger*
207835304 Nov 14 00:50 mesos-master*
37167040 Nov 14 00:49 mesos-usage*
4096 Nov 14 00:42 messages/
4096 Nov 14 00:31 slave/
48454064 Nov 14 00:50 test-helper*
4096 Nov 14 00:31 tests/
4096 Nov 14 00:31 usage/
```

~20 minutes

ninja -j 16 (Clang/LLVM 3.9.0)  
(with MESOS-5656 patch)

```
4096 Nov 14 20:16 ./
4096 Nov 14 20:01 ../
4096 Nov 14 20:01 cli/
4096 Nov 14 20:01 CMakeFiles/
2642 Nov 14 20:01 cmake_install.cmake
574 Nov 14 20:01 CTestTestfile.cmake
4096 Nov 14 20:01 docker/
4096 Nov 14 20:01 launcher/
995962070 Nov 14 20:15 libmesos-1.1.0.a
50271156 Nov 14 20:11 libmesos-protobufs.a
4096 Nov 14 20:01 local/
4096 Nov 14 20:01 log/
4096 Nov 14 20:10 master/
303861872 Nov 14 20:17 mesos-agent*
60223488 Nov 14 20:16 mesos-containerizer*
74539888 Nov 14 20:16 mesos-docker-executor*
430519200 Nov 14 20:17 mesos-execute*
90062528 Nov 14 20:16 mesos-executor*
56606224 Nov 14 20:16 mesos-fetcher*
420589456 Nov 14 20:17 mesos-local*
72441688 Nov 14 20:16 mesos-log*
32318288 Nov 14 20:15 mesos-logrotate-logger*
217117512 Nov 14 20:16 mesos-master*
37749648 Nov 14 20:15 mesos-usage*
4096 Nov 14 20:10 messages/
4096 Nov 14 20:01 slave/
50091536 Nov 14 20:16 test-helper*
4096 Nov 14 20:01 tests/
4096 Nov 14 20:01 usage/
```

~17.5 minutes

there is something wrong with current  
cmake build files in Mesos which will  
generate big binary executable files

## Getting started

- <http://llvm.org/docs/GettingStarted.html>
  - [http://lld.llvm.org/getting\\_started.html](http://lld.llvm.org/getting_started.html)
  - <http://lldb.llvm.org/build.html>
  - <https://github.com/mclinker/mclinker/wiki/Getting-Started>
- 

`$LLVM_HOME/llvm`

`$LLVM_HOME/llvm/tools/clang`

`$LLVM_HOME/llvm/tools/clang/tools/clang-tools-extra`

`$LLVM_HOME/llvm/tools/lld`

`$LLVM_HOME/llvm/projects/compiler-rt`

`$LLVM_HOME/llvm/projects/libcxx`

`$LLVM_HOME/llvm/projects/libcxxabi`

`$LLVM_HOME/llvm/projects/libunwind`

`$LLVM_HOME/llvm/projects/openmp`

`$LLVM_HOME/llvm/projects/test-suite`



## Natively compile upstream Clang/LLVM for RPi3

- <http://llvm.org/docs/HowToBuildOnARM.html>
- Use Ninja to build ~22 hours on RPi3 and finally got failed

```
cmake -G Ninja -DCMAKE_INSTALL_PREFIX=./Install -DCMAKE_BUILD_TYPE=Release -DLLVM_ENABLE_LLD=True  
-DLLVM_BUILD_DOCS=False -DLLVM_TARGETS_TO_BUILD=ARM ..
```

```
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h: In instantiation of 'struct llvm::cast_retty<clang::ObjCAtFinallyStmt, const clang::Stmt* const>':  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:248:5: required by substitution of 'template<class X, class Y> typename std::enable_if(!  
  llvm::is_simple_type<Y::value>(), typename llvm::cast_retty<X, const Y::ret_type>::type llvm::cast_or_null(const Y&) [with X = clang::ObjCAtFina  
llyStmt; Y = const clang::Stmt*]'  
2016-11-13 13:20:51 ../tools/clang/include/clang/AST/StmtObjC.h:228:73: required from here  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:182:72: warning: ignoring attributes on template argument 'llvm::simplify_type<const clang:  
:Stmt* const>::SimpleType {aka const clang::Stmt*}' [-Wignored-attributes]  
2016-11-13 13:20:51      typename simplify_type<From>::SimpleType>::ret_type ret_type;  
2016-11-13 13:20:51      ~~~~~~  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h: In instantiation of 'struct llvm::cast_retty<clang::Expr, const clang::Stmt* const>':  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:248:5: required by substitution of 'template<class X, class Y> typename std::enable_if(!  
  llvm::is_simple_type<Y::value>(), typename llvm::cast_retty<X, const Y::ret_type>::type llvm::cast_or_null(const Y&) [with X = clang::Expr; Y =  
const clang::Stmt*]'  
2016-11-13 13:20:51 ../tools/clang/include/clang/AST/StmtOpenMP.h:1977:56: required from here  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:182:72: warning: ignoring attributes on template argument 'llvm::simplify_type<const clang:  
:Stmt* const>::SimpleType {aka const clang::Stmt*}' [-Wignored-attributes]  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h: In instantiation of 'bool llvm::isa(const Y&) [with X = clang::CaseStmt; Y = const clang::  
Stmt*]':  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:298:16: required from 'typename llvm::cast_retty<X, Y>::ret_type llvm::dyn_cast(Y*) [wit  
h X = clang::CaseStmt; Y = const clang::Stmt; typename llvm::cast_retty<X, Y>::ret_type = const clang::CaseStmt*]'  
2016-11-13 13:20:51 ../tools/clang/include/clang/AST/Stmt.h:737:69: required from here  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:133:74: warning: ignoring attributes on template argument 'llvm::simplify_type<const clang:  
:Stmt* const>::SimpleType {aka const clang::Stmt*}' [-Wignored-attributes]  
2016-11-13 13:20:51      return isa_impl_wrap<X, const Y,  
2016-11-13 13:20:51      ~~~~~~  
2016-11-13 13:20:51      typename simplify_type<const Y>::SimpleType>::doit(Val);  
2016-11-13 13:20:51      ~~~~~~  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h: In instantiation of 'bool llvm::isa(const Y&) [with X = clang::Expr; Y = const clang::Stmt  
*]':  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:298:16: required from 'typename llvm::cast_retty<X, Y>::ret_type llvm::dyn_cast(Y*) [wit  
h X = clang::Expr; Y = const clang::Stmt; typename llvm::cast_retty<X, Y>::ret_type = const clang::Expr*]'  
2016-11-13 13:20:51 ../tools/clang/include/clang/StaticAnalyzer/Core/PathSensitive/ProgramState.h:736:40: required from here  
2016-11-13 13:20:51 ../include/llvm/Support/Casting.h:133:74: warning: ignoring attributes on template argument 'llvm::simplify_type<const clang:  
:Stmt* const>::SimpleType {aka const clang::Stmt*}' [-Wignored-attributes]  
2016-11-13 13:20:51 ninja: build stopped: subcommand failed.
```

■ clang-llvm-3.9.0-armv7a-linux-gnueabi.hf

- [illegible]



## 4) Upcoming LLVM/Clang 4.x for Mesos

- <https://llvmdevelopersmeetingbay2016.sched.org/>

### ORC – LLVM's Next Generation of JIT API

---

A good news for Java, Python Frameworks on Mesos

### LLVM Coroutines – Bringing resumable functions to LLVM

Actor Model based libprocess may benefit from it

### llbuild – A New Architecture for Building Software

The build system of Mesos may need some update

### Performance improvements in libcxx

Fix some regressions on LLVM/Clang 3.9.0

And many other optimizations...

## 5) Summary

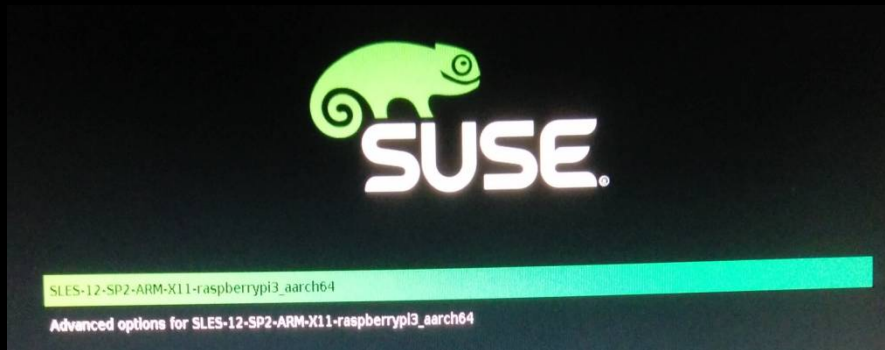
- LLVM/Clang is a great innovation for software projects and programmers
- Clang/LLVM can directly running on ARM devices like RPi now
- LLVM based next generation linkers and linking technologies is important to build Mesos on ARM device
- It'll be better tomorrow

# IV. Virtualization-assisted Development

## AARCH64 distribution for RPi3

- <https://www.suse.com/newsroom/post/2016/new-suse-linux-enterprise-12-service-pack-2-speeds-innovation-with-reliability/>
  - Support for **ARMv8-A**, including enablement for the Raspberry Pi3, making SUSE Linux Enterprise Server 12 SP2 one of the first commercially available enterprise Linux platforms for this architecture.

### Take a try



```
linux:/ # uname -a
Linux linux 4.4.21-69-default #1 SMP Tue Oct 25 10:58:20 UTC 2016 (9464f67) aarc
h64 aarch64 aarch64 GNU/Linux

linux:~ # free -h
```

	total	used	free	shared	buffers	cached
Mem:	785M	316M	468M	5.8M	2.3M	193M
-/+ buffers/cache:		120M	664M			
Swap:	494M	0B	494M			

# 1) Ubuntu/Fedora ARM64













## Ubuntu Cloud

- <http://www.cnx-software.com/2016/05/10/how-to-run-ubuntu-16-04-aarch64-64-bit-arm-cloud-images-on-your-intelamd-linux-computer/>

<https://releases.linaro.org/components/kernel/uefi-linaro/16.02/release/qemu64/>

	Name	Last modified	Size	License
	Parent Directory			
	QEMU_EFI.fd	25-Feb-2016 12:00	2.0M	open
	QEMU_EFI.img.gz	25-Feb-2016 12:00	723.4K	open

<https://cloud-images.ubuntu.com/yakkety/current/>

	yakkety-server-cloudimg-arm64-lxd.tar.xz	10-Nov-2016 17:41	840	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-arm64.img	10-Nov-2016 17:36	292M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-arm64.manifest	10-Nov-2016 17:40	13K	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-arm64.squashfs	10-Nov-2016 17:41	138M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-arm64.squashfs.manifest	10-Nov-2016 17:41	13K	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-arm64.tar.gz	10-Nov-2016 17:43	221M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf-lxd.tar.xz	10-Nov-2016 17:37	840	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf.img	10-Nov-2016 17:33	387M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf.manifest	10-Nov-2016 17:36	13K	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf.squashfs	10-Nov-2016 17:37	138M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf.squashfs.manifest	10-Nov-2016 17:37	13K	Ubuntu Server 16.10 (Yakkety Yak)	daily builds
	yakkety-server-cloudimg-armhf.tar.gz	10-Nov-2016 17:39	356M	Ubuntu Server 16.10 (Yakkety Yak)	daily builds

## apt-get install qemu qemu-utils cloud-utils

### //generate a ssh key pair to ~/.ssh

```
mydev2@mydev2-virtual-machine:~/.ssh$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/mydev2/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mydev2/.ssh/id_ecdsa.
Your public key has been saved in /home/mydev2/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:dXncuQyHuu3E7bj1k7X5VZaGxJ99isEw2n4kSS0k0Fc mydev2@mydev2-virtual-machine
The key's randomart image is:
+---[ECDSA 256]---+
|    .. ..E    |
|    ..0.    .0...|
|    ... * 0=00.|
|    * B0.* =   |
|    S +.+ B=   |
|    . 0+000+   |
|    ..0+.0*    |
|    .0 +=0     |
|    +..=       |
+-----[SHA256]-----+
```

### //edit ubuntucloud.config

```
users:
- name: ubuntucloud
  ssh-authorized-keys:
  - ssh-ecdsa AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBG3neetMc4QArUwCkpqoLpR5NLDzwxVrdw4p
    Ajp2kXM6ZMY2ttVmWnjw1jxhRDgKRjCsHMuLNwkGzZ7Cbyp8bG4= mydev2@mydev2-virtual-machine
  sudo: ['ALL=(ALL) NOPASSWD:ALL']
  groups: sudo
  shell: /bin/bash
```

## cloud-localds ubuntucloud-yakkety.img ubuntucloud.config

### //edit start-ubuntu-yakkety-cloud-arm64.sh

```
##### Ubuntu Cloud
```

```
## yakkety
```

```
qemu-system-aarch64 -smp 2 -m 16384 -M virt -bios QEMU_EFI.fd -nographic \  
-device virtio-blk-device,drive=image \  
-drive if=none,id=image,file=yakkety-server-cloudimg-arm64.img \  
-device virtio-blk-device,drive=cloud \  
-drive if=none,id=cloud,file=ubuntucloud-yakkety.img \  
-netdev user,id=user0 -device virtio-net-device,netdev=user0 -redir tcp:2222::22 \  
-cpu cortex-a53
```

### ./start-ubuntu-yakkety-cloud-arm64.sh

```
error: no suitable video mode found.  
EFI stub: Booting Linux Kernel...  
EFI stub: EFI_RNG_PROTOCOL unavailable, no randomness supplied  
EFI stub: Using DTB from configuration table  
EFI stub: Exiting boot services and installing virtual address map...  
[ 0.000000] Booting Linux on physical CPU 0x0  
[ 0.000000] Linux version 4.8.0-27-generic (buildd@bos01-arm64-047) (gcc version 6.2.0 20161005 (Ubuntu 6.2.0-5ubuntu12) ) #29-Ubuntu SMP Thu Oct 20 21:02:10 UTC 2016 (Ubuntu 4.8.0-27.29-generic 4.8.1)  
[ 0.000000] Boot CPU: AArch64 Processor [410fd034]  
[ 0.000000] efi: Getting EFI parameters from FDT:  
[ 0.000000] efi: EFI v2.60 by EDK II  
[ 0.000000] efi: SMBIOS 3.0=0x43c030000 ACPI=0xffff0000 ACPI 2.0=0xffff0014 MEMATTR=0x43ec5e01:  
[ 0.000000] No NUMA configuration found  
[ 0.000000] NUMA: Faking a node at [mem 0x0000000000000000-0x0000000043ffffff]  
[ 0.000000] NUMA: Adding memblock [0x40000000 - 0xffff4ffff] on node 0  
[ 0.000000] NUMA: Adding memblock [0xffff50000 - 0xffffaffff] on node 0
```

```
...
```



```
[ OK ] Started Login Service.  
[FAILED] Failed to start LXD - container startup/shutdown.  
See 'systemctl status lxd-containers.service' for details.  
[ OK ] Started LSB: Record successful boot for GRUB.  
[ OK ] Started iSCSI initiator daemon (iscsid).  
Starting Login to default iSCSI targets...  
Starting Authenticate and Authorize Users to Run Privileged Tasks...  
[ OK ] Started Login to default iSCSI targets.  
[ OK ] Reached target Remote File Systems (Pre).  
[ OK ] Reached target Remote File Systems.  
Starting LSB: automatic crash report generation...  
Starting Permit User Sessions...  
Starting LSB: daemon to balance interrupts for SMP systems...  
[ OK ] Started Authenticate and Authorize Users to Run Privileged Tasks.  
[ OK ] Started Accounts Service.  
[ OK ] Started Permit User Sessions.  
Starting Hold until boot process finishes up...  
Starting Terminate Plymouth Boot Screen...  
[ OK ] Started Hold until boot process finishes up.  
[ OK ] Started Terminate Plymouth Boot Screen.
```

---

...

## //Might meet SSH fail !

```
mydev2@mydev2-virtual-machine:~/ssh$ ssh -p 2222 ubuntucloud@localhost  
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.  
ECDSA key fingerprint is SHA256:jTSER1EtfWboTboSzZvHY4KK6IUvaisnHi/M9SKXhW8.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.  
Permission denied (publickey).
```


# Fedora

- <https://fedoraproject.org/wiki/Architectures/AArch64/F24/Installation>
- [https://fedoraproject.org/wiki/Architectures/AArch64/Install\\_with\\_QEMU](https://fedoraproject.org/wiki/Architectures/AArch64/Install_with_QEMU)

## Installing Fedora aarch64 with QEMU and libvirt


These steps will work on both x86 and aarch64 hardware. If running on actual aarch64 hardware, the virt-install commands should automatically request **KVM** for maximum performance.

### Get the necessary bits

- Fedora 22 host or later is required
- Grab the latest qemu-system-aarch64, libvirt, and virt-manager
- Grab UEFI builds for QEMU and AARCH64: `sudo dnf install edk2-aarch64` (fedora 23 & newer)
  - Note: These bits are not part of fedora 22 and older due to licensing issues. See [Using UEFI with QEMU#EDK2\\_Licensing\\_Issues](#) for more info.
  - Install Gerd's nightly firmware repo, as described here: <https://www.kraxel.org/repos/> 
  - Install the relevant bits: `sudo dnf install edk2.git-aarch64`

## Installing F23 aarch64 from URL

- This example uses the F23 aarch64 install tree. The virt-install command is:

```
sudo ./virt-install \
--name f23-aarch64-urllinst --ram 2048 --arch aarch64 \
--boot uefi --disk size=8 \
--location https://dl.fedoraproject.org/pub/fedora-secondary/releases/23/Server/aarch64/iso/ 
```

## Installing F23 aarch64 from CDROM

- Grab the ISO:
  - This example uses the F23 aarch64 install DVD: <https://dl.fedoraproject.org/pub/fedora-secondary/releases/23/Server/aarch64/iso/Fedora-Server-DVD-aarch64-23.iso> 
  - Move it to `/var/lib/libvirt/images`
  - From the virt-manager git checkout, run:

```
sudo ./virt-install \
--name f23-aarch64-cdrom --ram 2048 --arch aarch64 \
--boot uefi --disk size=8 --os-variant fedora22 \
--cdrom /var/lib/libvirt/images/Fedora-Server-DVD-aarch64-23.iso
```

## 2) CoreOS ARM64

- <https://coreos.com/blog/coreos-on-arm64/>
  - <https://github.com/glevand/hikey-coreos>
  - <https://github.com/coreos/docs/blob/master/os/booting-with-qemu.md>
- 

### Startup

#download a alpha/beta/stable image according to <https://coreos.com/releases/>

#e.g., <https://alpha.release.core-os.net/amd64-usr/current/>

```
[file] coreos\_production\_qemu.DIGESTS
[file] coreos\_production\_qemu.DIGESTS.asc
[file] coreos\_production\_qemu.DIGESTS.sig
[file] coreos\_production\_qemu.README
[file] coreos\_production\_qemu.README.sig
[file] coreos\_production\_qemu.sh
[file] coreos\_production\_qemu.sh.sig
[file] coreos\_production\_qemu\_image.img.bz2
[file] coreos\_production\_qemu\_image.img.bz2.DIGESTS
[file] coreos\_production\_qemu\_image.img.bz2.DIGESTS.asc
[file] coreos\_production\_qemu\_image.img.bz2.sig
[file] coreos\_production\_qemu\_ufi.DIGESTS
[file] coreos\_production\_qemu\_ufi.DIGESTS.asc
[file] coreos\_production\_qemu\_ufi.DIGESTS.sig
[file] coreos\_production\_qemu\_ufi.README
[file] coreos\_production\_qemu\_ufi.README.sig
[file] coreos\_production\_qemu\_ufi.sh
[file] coreos\_production\_qemu\_ufi.sh.sig
[file] coreos\_production\_qemu\_ufi\_efi\_code.fd
[file] coreos\_production\_qemu\_ufi\_efi\_code.fd.sig
[file] coreos\_production\_qemu\_ufi\_efi\_vars.fd
[file] coreos\_production\_qemu\_ufi\_efi\_vars.fd.sig
[file] coreos\_production\_qemu\_ufi\_image.img.bz2
[file] coreos\_production\_qemu\_ufi\_image.img.bz2.DIGESTS
[file] coreos\_production\_qemu\_ufi\_image.img.bz2.DIGESTS.asc
[file] coreos\_production\_qemu\_ufi\_image.img.bz2.sig
```

**#generate a ssh key pair to ~/.ssh  
ssh-keygen**

**#vim ~/.ssh/config**

```
Host coreos
  HostName localhost
  Port 2222
  User core
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
```

**bzip2 -d coreos\_production\_qemu\_uefi\_image.img.bz2**

**#modify settings: ./coreos\_production\_qemu\_uefi.sh**

`#!/bin/sh`

```
SCRIPT_DIR="`dirname "$0"`"
VM_BOARD='arm64-usr'
VM_NAME='coreos_production_qemu_uefi-1221-0-0'
VM_UUID=
VM_IMAGE='coreos_production_qemu_uefi_image.img'
VM_KERNEL=
VM_INITRD=
VM_MEMORY='16384'
VM_CDROM=
VM_PFLASH_R0='coreos_production_qemu_uefi_efi_code.fd'
VM_PFLASH_RW='coreos_production_qemu_uefi_efi_vars.fd'
VM_NCPUS="`grep -c ^processor /proc/cpuinfo`"
SSH_PORT=2222
SSH_KEYS=""
CONFIG_FILE=""
CONFIG_IMAGE=""
SAFE_ARGS=0
USAGE="Usage: $0 [-a authorized_keys] [--] [qemu options...]"
```

**./coreos\_production\_qemu\_uefi.sh -a ~/.ssh/id\_rsa.pub -- -nographic**

...

```
[ 94.686589] systemd[1]: Started Cleanup udevd DB.
[ OK ] Started Cleanup udevd DB.
[ OK ] Reached target Switch Root.
[ 94.748990] systemd[1]: Reached target Switch Root.
[ 94.875395] systemd[1]: Starting Switch Root...
Starting Switch Root...
[ 95.456545] systemd[1]: Switching root.
[ 96.159314] systemd-journald[157]: Received SIGTERM from PID 1 (systemd).
[ 98.243453] systemd: 14 output lines suppressed due to ratelimiting
[ 98.691957] ip_tables: (C) 2000-2006 Netfilter Core Team
```

Welcome to CoreOS 1221.0.0 (MoreOS)!

```
[ OK ] Stopped Switch Root.
[ OK ] Stopped Journal Service.
Starting Journal Service...
[ OK ] Created slice system-addon\x2drun.slice.
[ OK ] Listening on /dev/initctl Compatibility Named Pipe.
[ OK ] Started Forward Password Requests to Wall Directory Watch.
[ OK ] Reached target Remote File Systems.
[ OK ] Set up automount Arbitrary Executab...ats File System Automount Point.
[ OK ] Created slice User and Session Slice.
[ OK ] Created slice system-addon\x2dconfig.slice.
[ 109.117588] audit: type=1305 audit(1479119644.993:2): audit_enabled=1 old=1 auid=4294967295 ses=4294967
295 subj=kernel res=1
Starting Apply Kernel Variables...
Mounting Debug File System...
[ OK ] Reached target Slices.
[ OK ] Started Dispatch Password Requests to Console Directory Watch.
[ OK ] Stopped target Switch Root.
[ OK ] Stopped target Initrd File Systems.
```

...

```
[ 245.756722] x9 : 0000000000000000 x8 : 0000ffff98ecdbe0
[ 245.824506] x7 : 0000000000000000 x6 : 0000000000000004
[ 245.923328] x5 : 0000000000000002 x4 : 0000000000000000
[ 246.033085] x3 : 0000000000000002 x2 : 0000000000000004
[ 246.108227] x1 : 0000ffff96911180 x0 : 0000ffff969111a0
[ 246.192201]
[ 246.319095] audit: type=1701 audit(1479121626.494:3): auid=4294967295 uid=235 gid=235 ses=4294967295 su
bj=kernel pid=600 comm="polkitd" exe="/usr/lib/polkit-1/polkitd" sig=11
```

```
This is localhost (Linux aarch64 4.8.6-coreos) 11:07:11
SSH host key: SHA256:H61NZsE4ljbFXC5DhnVBYN9LUULDNL45rwN1SPyBUpw (RSA)
SSH host key: SHA256:6Tm/pXyqI8hjFR9AAn60jjXw1UrK5tPb1YX5iK791/U (DSA)
SSH host key: SHA256:r5poVlFFDxG232jwpW10Mc12H2UK8u4HPC3WkdtHK0o (ED25519)
SSH host key: SHA256:8MMKSafzuoy1T/GosPg2Ly1JTVpJHhp9XKPKZ5CVays (ECDSA)
eth0: 10.0.2.15 fec0::5054:ff:fe12:3456
```

localhost login:



## ssh coreos

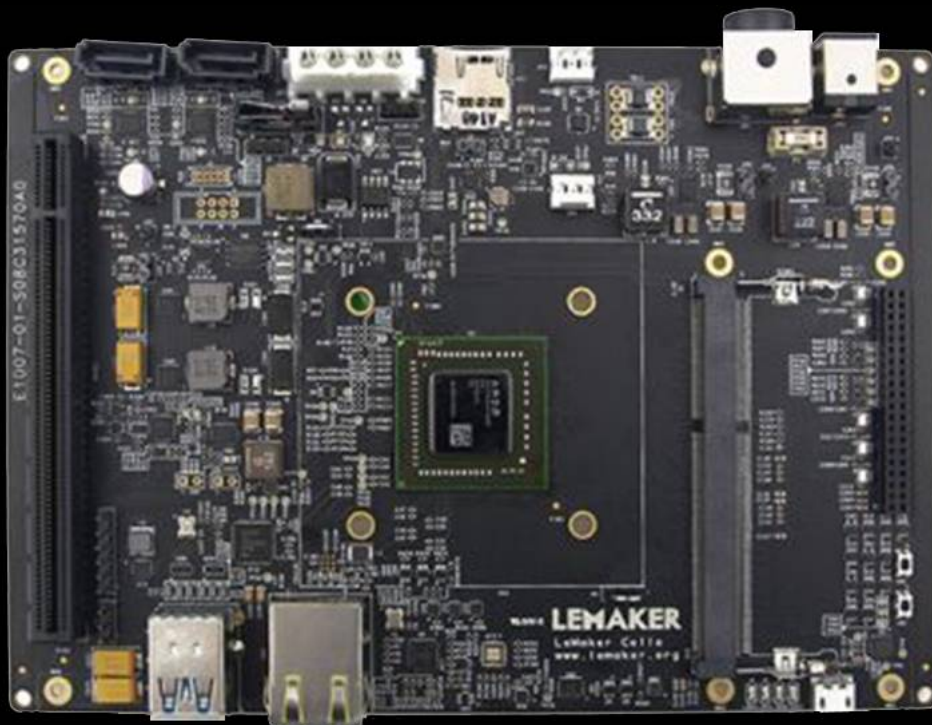
```
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.  
Enter passphrase for key '/home/mydev2/.ssh/id_rsa':  
CoreOS alpha (1221.0.0)  
Failed Units: 1  
    polkit.service  
core@localhost ~ $
```

---

```
core@localhost ~ $ uname -a  
Linux localhost 4.8_6-coreos #1 SMP PREEMPT Thu Nov 3 04:31:31 UTC 2016 aarch64 GNU/Linux
```

# V. Wrap-Up

- Moving to new experiment platform in 2017:
  - Upcoming Raspberry Pi 4
  - 96boards
  - Cortex-A73 development board



## ARM Cortex®-A73

ARM CoreSight™ Multicore Debug and Trace

ARMv8-A  
32b/64b CPU

NEON™  
SIMD engine

Floating  
Point Unit

64kB L1 I-Cache

32kB-64kB L1 D-Cache

Core 1

2

3

4

ACP

SCU

Shared L2 Cache (256kB-8MB)  
Optional ECC

128-bit AMBA®4 AXI4 or ACE

- **A fully customized Linux distribution:**
    - Upstreaming Kernel
    - Full support for AARCH64
    - LLVM/Clang 4.x as the unique toolchain
    - Development related packages preinstalled
    - Debian-based
- 

...



Rank	Distribution	H.P.D*
1	<a href="#">Mint</a>	2880▼
2	<a href="#">Debian</a>	1668→
3	<a href="#">Ubuntu</a>	1347→
4	<a href="#">openSUSE</a>	1234▲
5	<a href="#">elementary</a>	1085→
6	<a href="#">Manjaro</a>	1039▼
7	<a href="#">Fedora</a>	982▼
8	<a href="#">Zorin</a>	914→
9	<a href="#">CentOS</a>	773→
10	<a href="#">deepin</a>	736▲

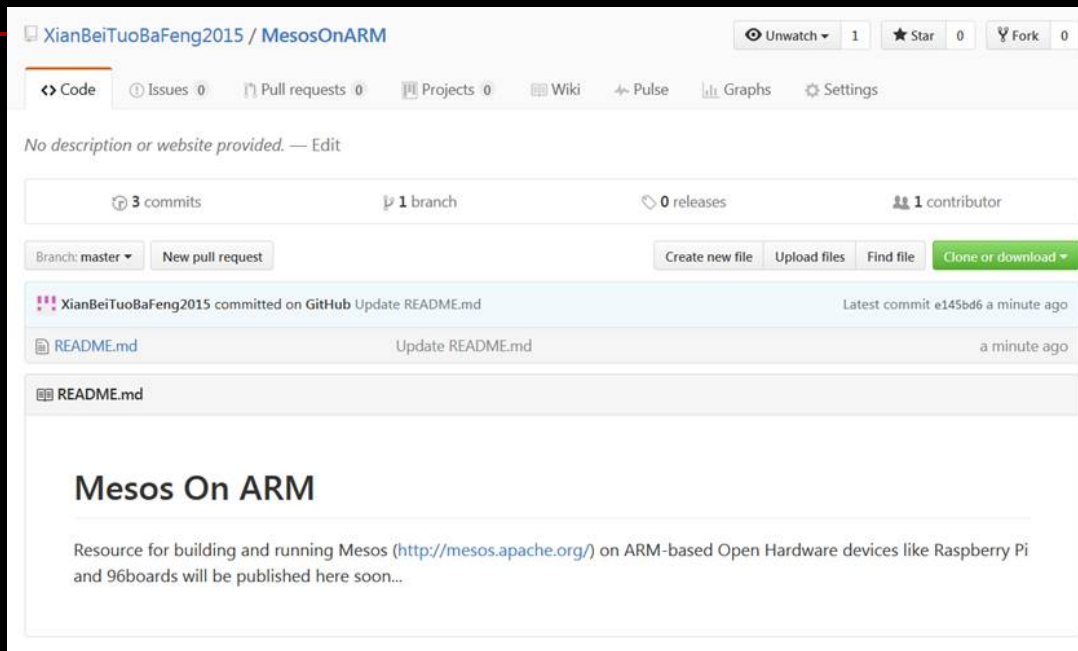
- **Clang/LLVM is the system compiler on several platforms in FreeBSD 10.0 and later, and GCC is not installed by default. FreeBSD is still pursuing the use of the LLVM Linker (LLD) on its base system, and its ARM64 support continues getting better.**

- **Build Mesos (version  $\geq 1.0$ ) on single RPi3 board seems to be “Mission Impossible”**



But why I still insist on trying to build Mesos on single RPi3 board? Because I believe that there is still much room left for further optimization, and I am aware that the success is depend on whole system optimization.

- **My github:**  
<https://github.com/XianBeiTuoBaFeng2015/MesosOnARM/>  
[XianBei2011@gmail.com](mailto:XianBei2011@gmail.com)



- **ARM-based Solution will certainly play a key role in next generation Data Center & Cloud Infrastructure, so make Mesos running stably, fastly, and smoonthly on ARM is very important!**



---

**Q & A**

# THANK YOU!

---



# Reference

Slides/materials from many and varied sources:

- <http://en.wikipedia.org/wiki/>
- <http://www.slideshare.net/>

---

- <https://www.kernel.org/>
- [http://elinux.org/Main\\_Page](http://elinux.org/Main_Page)
- <http://free-electrons.com/>
- <http://llvm.org>
- [http://llvm.linuxfoundation.org/index.php/Main\\_Page](http://llvm.linuxfoundation.org/index.php/Main_Page)
- <https://www.freebsd.org/>
- <http://mesos.apache.org/>
- <https://github.com/apache/mesos>
- <https://cmake.org/>
- [http://wiki.qemu.org/Main\\_Page](http://wiki.qemu.org/Main_Page)
- ...