An efficiency unit test and fuzz tools for kernel/libc porting

Self introduction

- Kernel developer from Huawei
- Linaro kernel working group assignee
- Focus on migration 32bit application from arm 32bit hardware to 64bit hardware
- Interested in in memory management

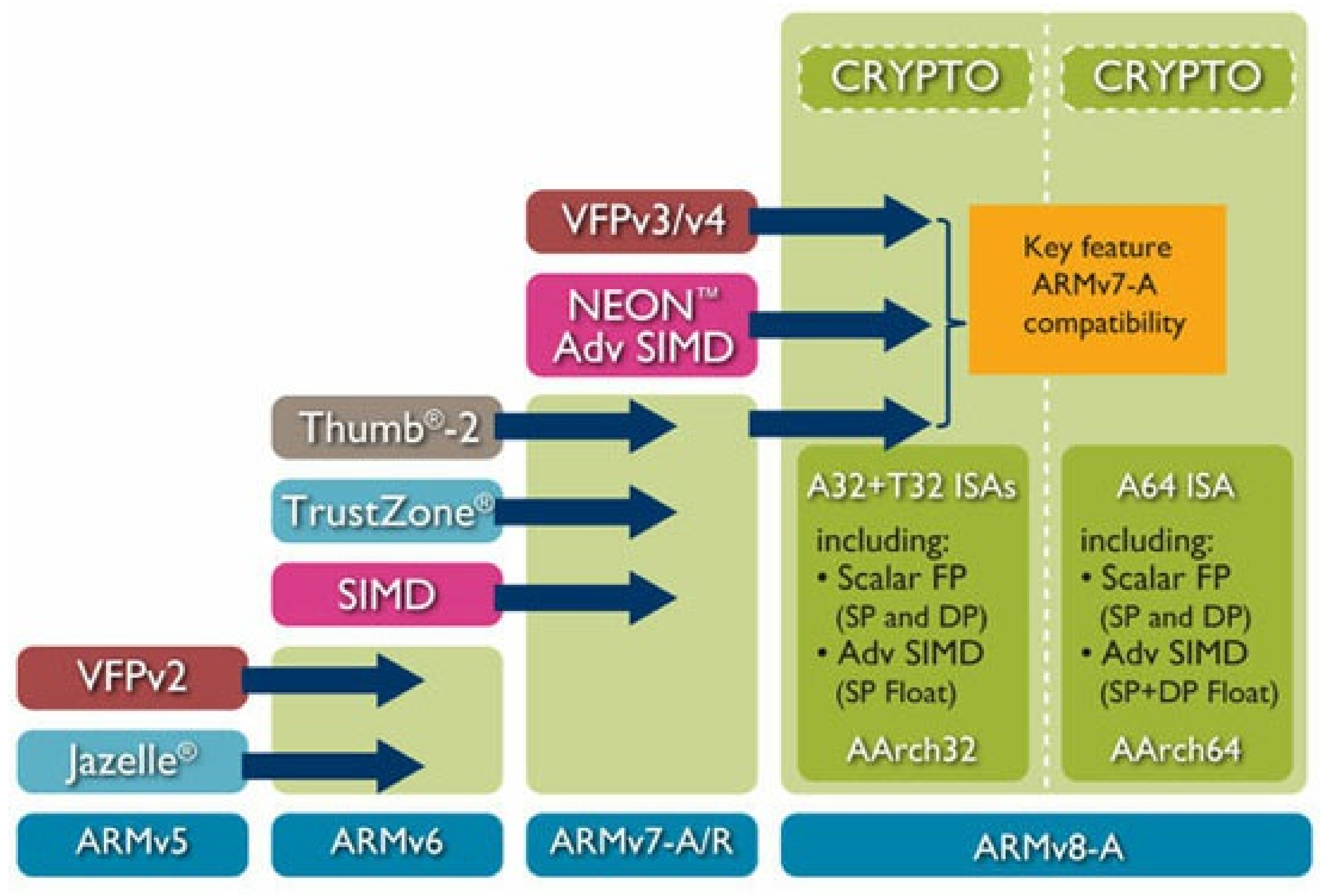
aarch64 ILP32 overview.

What is ILP32?

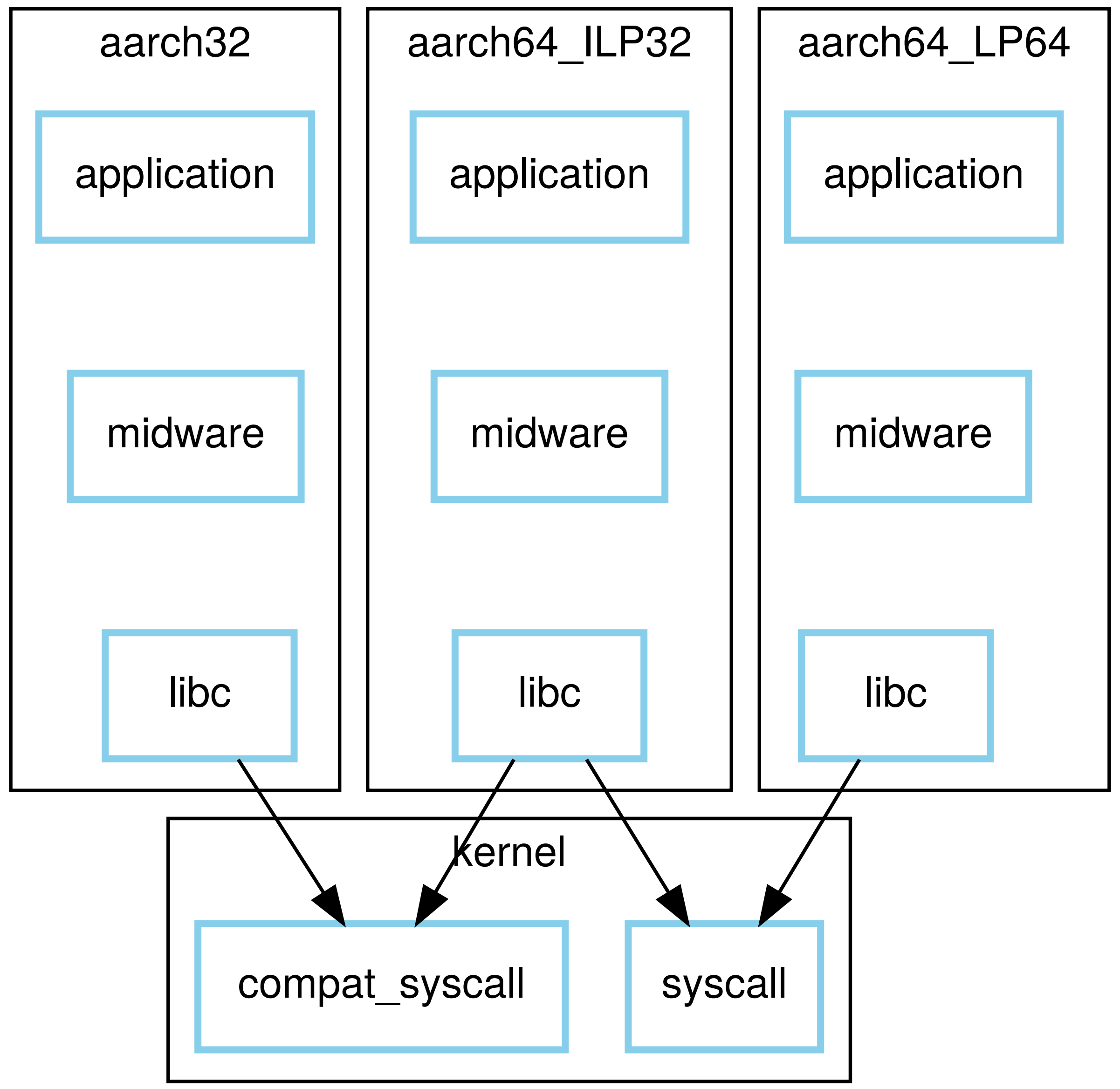
Data model

	ILP32	LP64	LLP64	ILP64
char	8	8	8	8
short	16	16	16	16
int	32	32	32	64
long	32	64	32	64
long long	64	64	64	64
size_t	32	64	64	64
pointer	32	64	64	64
	Arm/aarch32/aarch64 ILP32	aarch64		

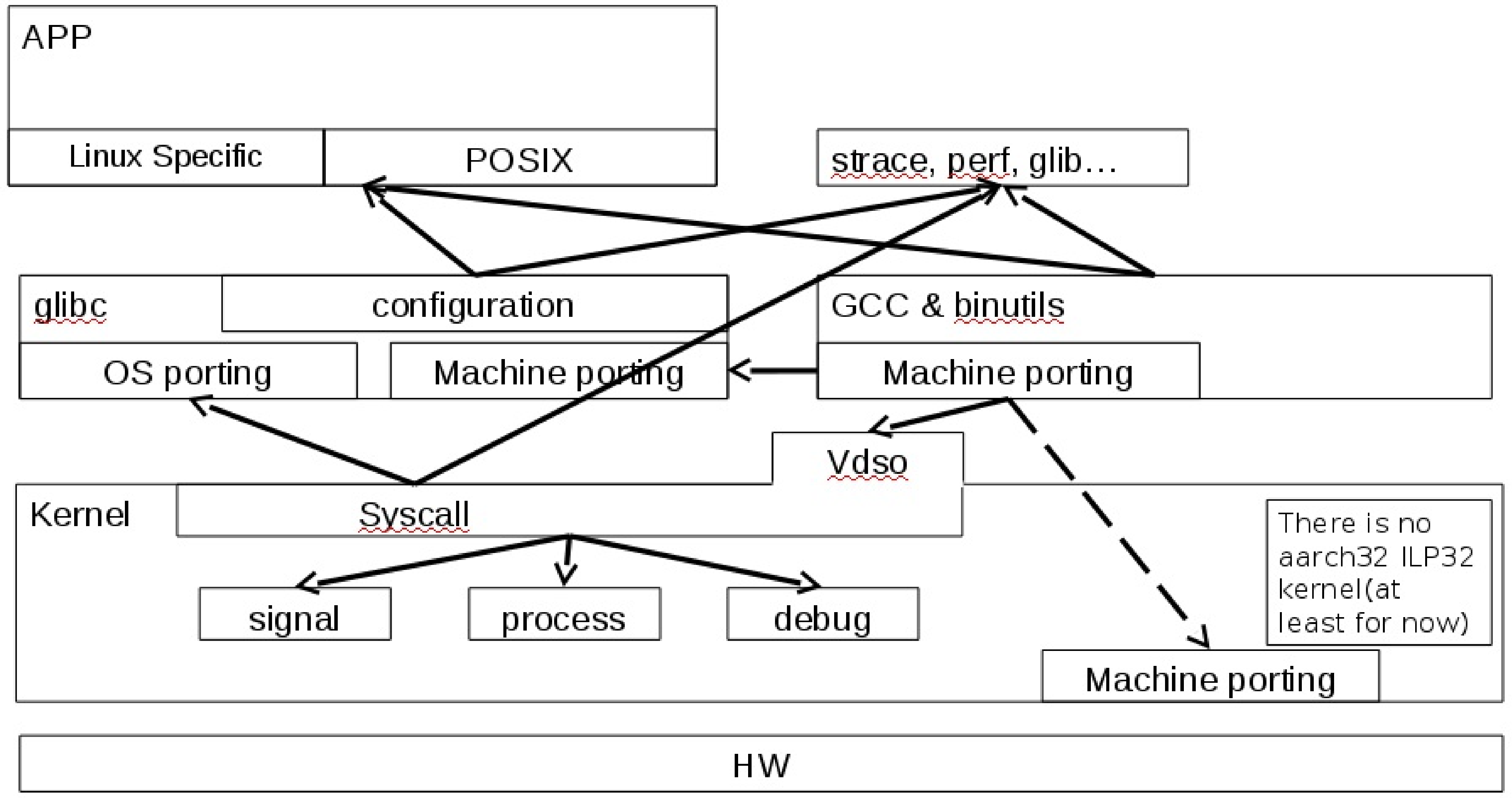
arm architecture



Migrate 32bit application to 64bit hardware



ILP32 enablement



Why we need unit test for ILP32?

There are actually lots of choices for a new api.

- The definition of basic type, such as time_t, off_t and so on.
- Argument passing.
- delouse.

Lots of abi changes.

During developemnt of ILP32, there are three big change in ILP32 which lead to lots of duplicate verfication work.

Version A

- Most of syscall is as same as 64bit syscall.
- time_t and off_t is 64bit. But in POSIX, time_t should be 32bit for 32bit application.

Version B

- Most of syscall is compat syscall.
- time_t and off_t is 32bit
- Pass 64bit variable through one 64bit register.
- Do the sign extend when enter into kernel.

Version C

- Most of syscall is compat syscall.
- time_t is 32bit and off_t is 64bit
- Pass 64bit variable through two 32bit register.
- Clear the top-havies of of all the registers of syscall when enter kernel.

How many issues found by trinity when LTP syscall fails is < 20?

0

Compare the exist kernel/glibc test tools

- Whether easy to reproduce the failure.
- Whether support coverage
- Whether support libc test
- Whether generate the full random data to basic data type

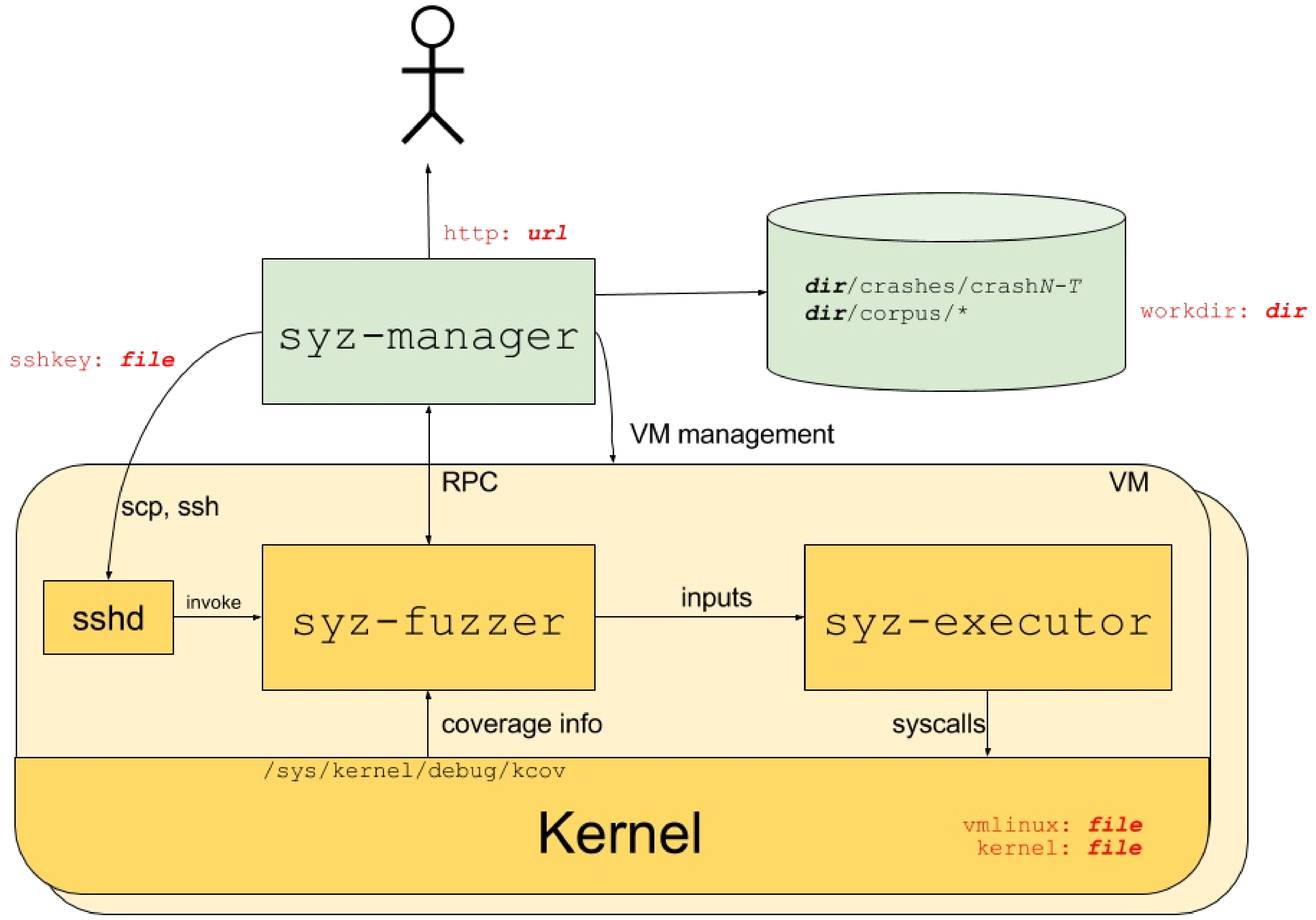
LTP and glibc testsuite

- The tridition testsuite for kernel and glibc.
- No fuzz test. Test pass may hide some issue.

Trinity

- Generate the fuzz data in a set of data type
- Generate the random address instead of basic data type for most of pointers.
- Support lots of architecture
- Takes too long to produce an issue and takes more and more to re-produce and analysis it
- Is going to add the coverage support(?)

Syzkaller


http: *url*
syz-manager
dir/crashes/crash*N-T*
dir/corpus/*
workdir: *dir*
sshkey: *file*
VM management
RPC
VM
scp, ssh
sshd
invoke
syz-fuzzer
inputs
syz-executor
coverage info
syscalls
/sys/kernel/debug/kcov
Kernel
vmlinux: *file*
kernel: *file*

Syzkaller(Cont.)

1. ● Syzkaller could recursively randomize base date type
2. ● Syzkaller could generate the readable short testcases
3. ● Syzkaller could do the coverage
4. ● Syzkaller do not test glibc

AFL and triforce

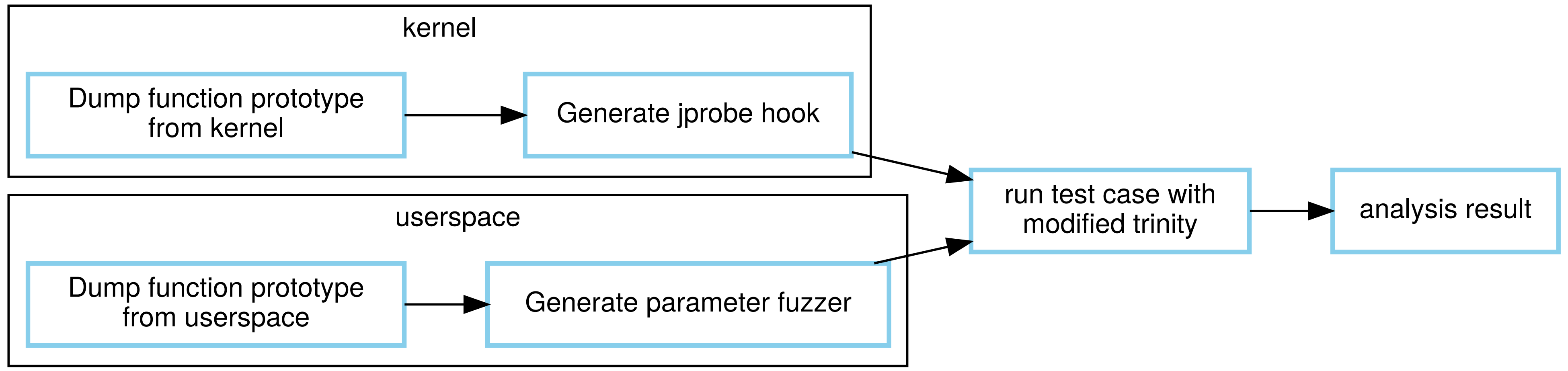
- Do not need the coverage support in kernel. Cool for the old kernel
- Need special instrucion in qemu.

What's missing?

- There is no testsuite care about the porting of libc and kernel.
- There is no full unit test for syscall.

Introduce syscall unit test

The test flow of syscall unit test


kernel
Dump function prototype from kernel
Generate jprobe hook
userspace
Dump function prototype from userspace
Generate parameter fuzzer
run test case with modified trinity
analysis result

Found twos issue with our tools in a specific version

- readahead
- sync_file_range

What is the future of syscall unit test?

Contribution to LTP and/or glibc testsuite? Or keep it as a standalone test case?

TODO list

- Support all the syscalls which are not wrapped by libc.
- Full automation in generating the fuzz code.

Q & A