Instituto de Matemática e Estatística
da Universidade de São Paulo

MAC 0499

Trabalho de Formatura Supervisionado

Proposta de TCC
Uma proposta de evolução dos testes e integração contínua do Kernel Linux

Marcelo Schmitt

Supervisor: Paulo RM Meirelles
Co-Supervisor: Fabio Kon

27 de abril de 2019

1 Introdução

Com o surgimento novas famílias de processadores que executam Linux, especialmente as de arquitetura ARM, tornou-se cada vez mais difícil garantir o funcionamento do Kernel Linux em diferentes dispositivos. Considerando também o ambiente de frequentes mudanças propiciado pela comunidade de software livre, com novas versões do Linux são lançadas a cada dois meses contendo milhares de mudanças, a manutenção do código fonte pode se tornar um processo estressante como relatado por Linus Torvalds [11]. Apesar disso, processadores ARM chegaram a estar presente em cerca de 98% dos telefones móveis[12] e permaneceram como os mais amplamente utilizados em dispositivos móveis segundo[4]. Portanto, dar suporte a tais arquiteturas é crucial para projetos derivados do Linux que são voltados para dispositivos móveis, por exemplo, o sistema Android.

Contudo, apenas compilar o código fonte do Kernel não é o suficiente para garantir que ele seja portável para um determinado processador pois podem ocorrer erros durante o seu processo de inicialização (*boot*). Além disso, alguns dispositivos podem aceitar diferentes configurações de compilação e inicialização especificadas, em arquivos do tipo defconfig ou de devicetree personalizados, como explicam Mark Brown e Kevin Hilman em palestra na OpenIoTSummit[13]. Logo, para que seja viável testar a portabilidade do Kernel na diversa variedade de arquiteturas de hardware, torna-se necessário o emprego de ferramentas de testes automatizados para que seja possível lidar com o crescente número de novos dispositivos e configurações associadas.

Uma iniciativa que tem se destacado no sentido de melhorar a portabilidade do Kernel para essa diversidade de novos dispositivos é o KernelCI. Este projeto consiste no desenvolvimento de um sistema automatizado de testes distribuídos com foco no teste de diferentes ramificações do Kernel Linux. O principal objetivo disso é elevar a qualidade dos sistemas operacionais derivados provendo testes e validação sobre uma ampla variedade de plataformas de hardware heterogêneas que executam Linux[8].

Juntamente ao KernelCI é muito comum o uso da LAVA (Linaro Automation and Validation Architecture)[6], um sistema de integração contínua (CI) para implantar imagens de Kernel Linux em hardware virtual e físico para a execução de testes automatizados. Contudo, a palavra teste pode ter diferentes significados em diferentes contextos. Testar se um código produzido "funciona" pode significar, por exemplo: testar se alterações no Kernel Linux compilam e são carregadas corretamente; testar se o código produzido pelo GCC (GNU Compiler Collection) é menor ou mais rápido; testar se uma mudança no escalonador de processos do Kernel reduz o consumo de energia sob certas condições de trabalho; entre outros. Ao longo deste trabalho, o

significado de teste será explicitado sempre que o contexto exigir isso.

As áreas de testes e CI destacam-se como um dos maiores avanços recentes da engenharia de software. Em especial, o uso de CI possibilitou entregas confiáveis e rápidas de novas funcionalidades em sistemas executando em produção. Apesar dos avanços em tal área, nota-se dois problemas: pouca sistematização da área e pouca integração com sistemas de mais baixo nível. Como exemplo de como o CI ainda não atingiu certas áreas, nota-se diversos esforços acontecendo em paralelo para consolidar tal área no Kernel Linux [[10], [9]]. Há também carência por testes específicos em vários subsistemas do Kernel, por exemplo, o teste de drivers pertencentes ao subsistema de entrada e saída industrial (IIO). Apesar do empenho de diferentes vertentes do Kernel em consolidar um mecanismo de testes/CI, essa área ainda é um desafio do ponto de vista da pesquisa e da indústria.

Com este trabalho propõem-se desenvolver uma extensão do conjunto de testes da LAVA para abranger drivers de sensores presentes no subsistema de IIO do Kernel Linux, desde o teste a chamadas de atributos até a validação da operação do dispositivo associado. Para isso deve-se montar uma infraestrutura contendo um computador servidor que realizará a compilação do Kernel e geração de imagens de inicialização destinadas aos dispositivos alvo dos testes e junto aos referidos dispositivos devem estar conectados alguns sensores nos quais drivers específicos serão automaticamente testados e validados. Inicialmente usaremos sensores e dispositivos obtidos por doações de empresas, emprestados por grupos de extensão como o Hardware Livre USP e o FLUSP (FLOSS at USP), ou pelos próprios estudantes envolvidos no projeto.

2 Objetivos

Neste trabalho de conclusão de curso, temos o objetivo de desenvolver de uma extensão da LAVA para que faça testes automatizados em periféricos de hardware como sensores e atuadores. Com isso, teremos um protótipo de uma plataforma de CI para validação do software do Kernel Linux. Como consequência, também discutiresmo sobre as praticas de CI e sua importância para o desenvolvimento de sistemas operacionais de código aberto.

3 Metodologia de desenvolvimento

Para o desenvolvimento da extensão do sistema de CI proposta neste trabalho, serão utilizadas boas práticas de desenvolvimento de software livre e de

métodos ágeis. O desenvolvimento será feito em colaboração com membros das comunidades de software livre da Linaro e de CI do Kernel Linux, que já possuem implementações de CI para algumas das funcionalidades do Kernel. Além do conjunto de testes, as ferramentas disponibilizadas por essas comunidades também dispõem de pipelines configuráveis para tomar o código produzido pelos desenvolvedores do Kernel desde o repositório de código fonte, passando pela etapa de compilação, até a instalação e inicialização em dispositivos selecionados para validação.

O código desenvolvido será gerenciado por meio de repositório Git, hospedado no ambiente colaborativo Gitlab, e a gestão do projeto de desenvolvimento será baseada no Gitlab Issue Tracker. O software produzido será disponibilizado como software livre para que possa ser integrado a outros projetos e comunidades de software livre, em especial, ao projeto da LAVA o qual vamos tomar como ponto de partida. Para a montagem da infraestrutura de CI devem ser estudadas implementações de laboratórios já existentes como os da Bootlin [[1], [2], [3]] e da BayLibre[8].

4 Estado Atual

Este trabalho é baseado na já existente infraestrutura de CI do Kernel Linux[6] em operação juntamente com a LAVA [13]. Existe uma interface web para e exibição dos resultados dos teste realizados, como ilustrado nas Figuras 1 e 2.

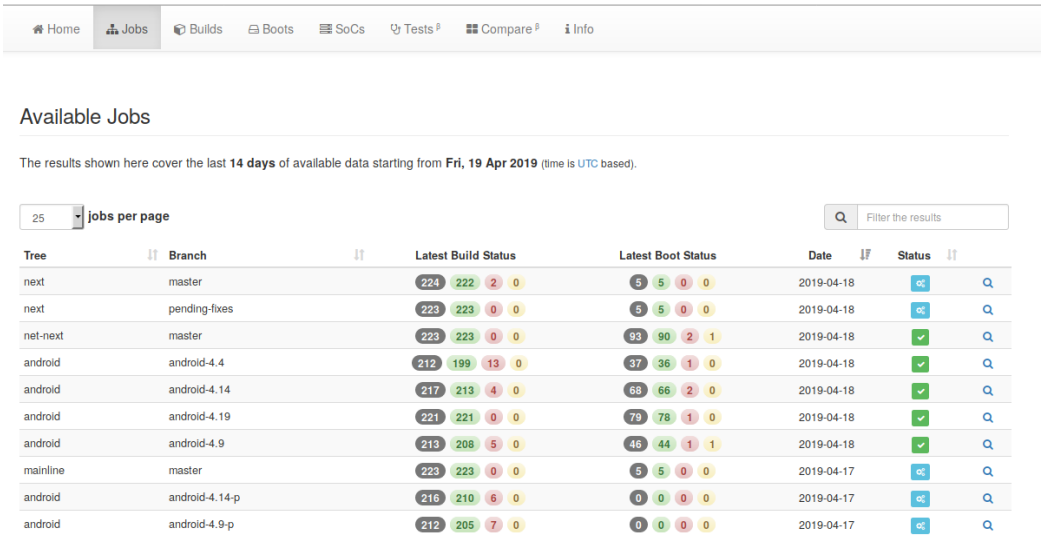

Figura 1: Visão de tarefas de teste executadas pelo kernelci.org[7]

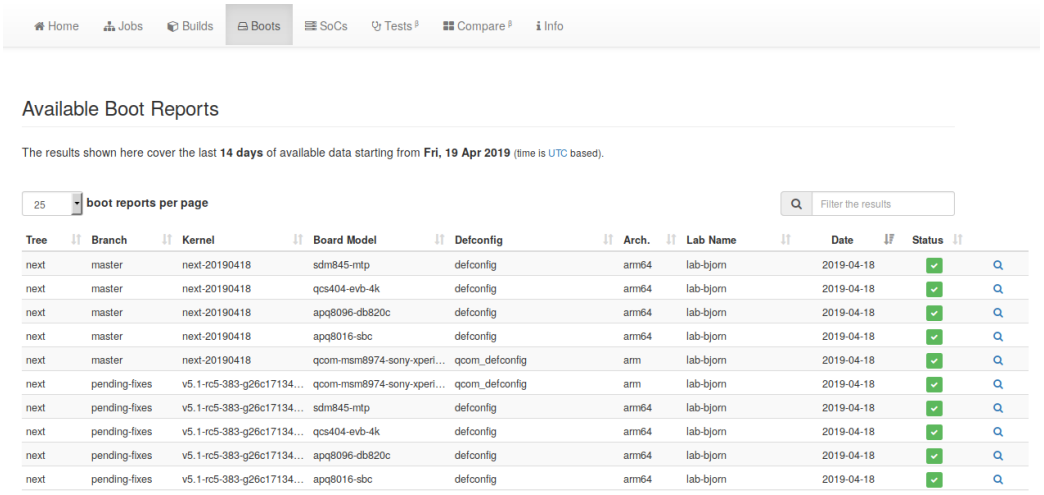

Figura 2: Visão dos resultados de testes de *boot* executados pelo kernelci.org[5]

Algumas placas doadas pela Linaro (Figura 3) já estão disponíveis para a condução de testes logo que a infraestrutura de CI estiver operando.

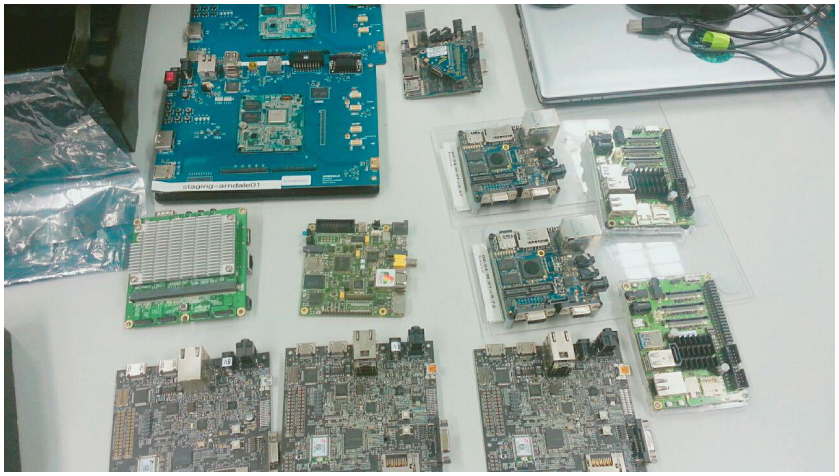

Figura 3: Placas doadas pela Linaro ao FLUSP

Para capturar os *logs* de *boot* do Kernel, as placas normalmente se comunicam com o servidor por serial UART. Como teste preliminar da capacidade de estabelecer comunicação por UART foi obtido um terminal de comando interativo em uma Raspberry Pi por meio do referido protocolo, como demonstrado na Figura 4.

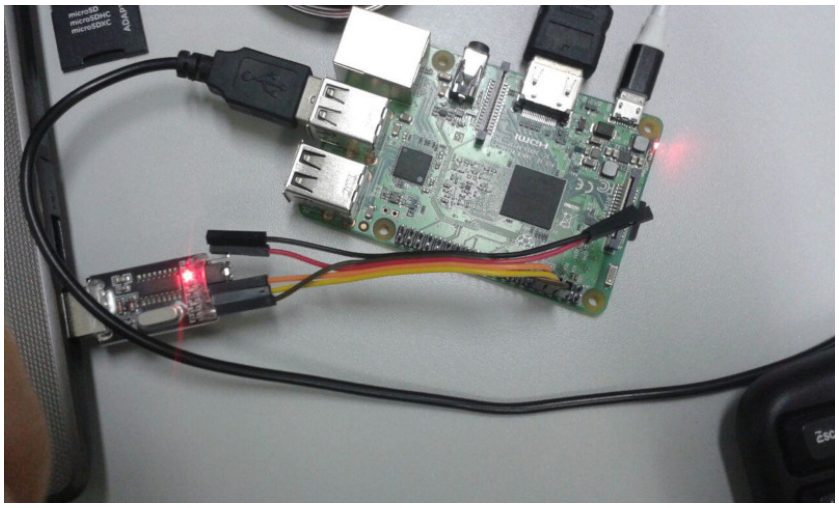

Figura 4: Experimento de conexão UART com Raspberry Pi[14]

5 Cronograma de Atividades

Este trabalho está baseado na investigação do estado-da-prática e do estado-da-arte das área de CI. O desenvolvimento será feito em colaboração com membros das comunidades de software livre da Linaro e de CI do Kernel Linux, que já possuem implementações de CI para algumas das funcionalidades do Kernel. Além do conjunto de testes, as ferramentas disponibilizadas por essas comunidades também dispõem de pipelines configuráveis para tomar o código produzido pelos desenvolvedores do Kernel desde o repositório de código fonte, passando pela etapa de compilação, até a instalação e inicialização em dispositivos selecionados para validação.

Trabalharemos com o software de drivers desenvolvidos pela renomada comunidade do IIO (Industrial Input Output); esses drivers pertencem a diversas categorias de sensores usados em aplicações de IoT, como acelerômetros, giroscópios, cor e luminosidade, magnetômetros, pressão, proximidade, temperatura e várias outras. Contudo, as plataformas de testes automatizados que existem atualmente executam testes somente sobre alguns aspectos do software do Kernel Linux. Resta ainda implementar uma ampla variedade de testes relativos aos sensores para que haja um pipeline de DevOps que suporte adequadamente a operação de dispositivos de IoT.

A Tabela 1 resume as atividades previstas com uma previsão da execução as tarefas. As atividades de pesquisa proposta são: (1) estudo e colaboração com o processo de desenvolvimento do kernel; (2) realizar um levantamento bibliográfico do estado da arte da área de CI; (3) realizar um levantamento bibliográfico do estado da arte da área de DevOps; (4) apontar referências teóricas que indiquem a correlação entre CI e DevOps; (5) realizar entrevistas com especialista de mercado nas áreas de CI e DevOps de modo a obter uma visão geral das práticas utilizadas; (6) sugerir a adoção de novas práticas e melhorias naquelas já adotadas. Além disso, planejamos um conjunto de tarefas de técnicas, como: (1) montar uma infraestrutura local constituída por computadores executando uma plataforma de CI e, ligados a eles, dispositivos de IoT que receberão imagens do kernel Linux vindas dessa plataforma; (2) instanciar uma plataforma de CI; (3) criação de pipelines de testes específicos para dispositivos de IoT; (4) inserção de funcionalidades que permitam testar os drivers responsáveis pela operação de sensores presentes no IIO; (5) geração de registros de mensagens de depuração (logs) para avaliação do resultado dos testes; (6) avaliar o enquadramento da plataforma de CI construída como um pipeline de DevOps.

Tabela 1: Cronograma de atividades por mês de desenvolvimento

Atividade / Mês	**Mai/Jun**	**Jul/Ago**	**Set/Out**	**Nov/Dez**
Colaboração com o processo de desenvolvimento do kernel	X	X		
Levantamento bibliográfico do estado da arte da área de CI e DevOps	X	X		
Entrevistas com especialista de mercado nas áreas de CI e DevOps	X	X		
Escrita da monografia		X	X	X
Montar uma infraestrutura local		X	X	
Instanciar uma plataforma de CI			X	
Criação de *pipelines* de testes específicos para dispositivos de IoT			X	
Inserção de funcionalidades que permitam testar os drivers			X	X
Integração da infraestrutura local com a infraestrutura distribuída do kernelci.org				X
Elaboração de pôster				X

6 Referências

Referências

[1] Building a bards farm: Continuous integration and remote control.

[2] Building a bards farm: Continuous integration and remote control.

[3] Farm posts at bootlin.

[4] An interview with steve furber.

[5] Kernel ci dashboard - boot reports.

[6] Kernel ci dashboard - home.

[7] Kernel ci dashboard - job reports.

[8] kernelci: Linux kernel testing at kernelci.org.

[9] kernelci/kernelci-doc.

[10] Updates on the kernelci project.

[11] *Re: [GIT PULL] omap changes for v2.6.39 merge window*. LKML.ORG - the Linux Kernel Mailing List Archive, 2011.

[12] *ARM architecture*. Wikipedia, 2019.

[13] The Linux Foundation. Kernelci.org needs you!, Abr. 2017.

[14] Marcelo Schmitt. Como obter um terminal interativo de uma raspberry pi ligada à porta usb de um computador.