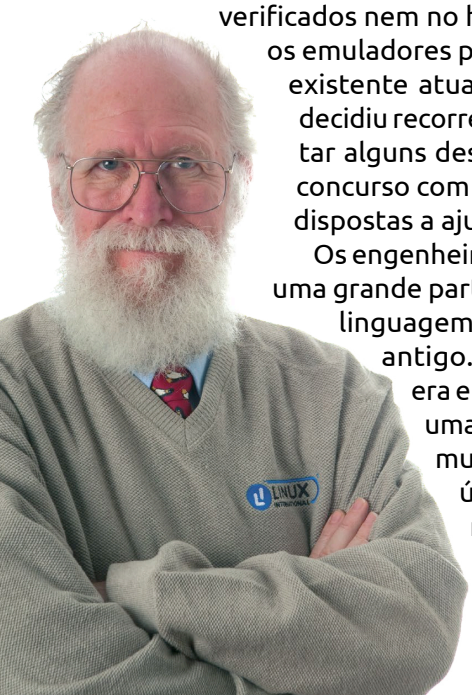
Coluna do Maddog
Linha Assembly

"maddog" analisa alguns esforços da Linaro para aprimorar o desempenho do GNU/Linux em arquiteturas ARM.

por Jon "maddog" Hall

Nos últimos meses, tenho trabalhado com o Linaro [1], uma associação de empresas que desejam ter o GNU/Linux trabalhando bem em arquiteturas ARM. Embora a ARM Holdings desenhe os chips de arquitetura ARM, várias outras empresas fabricam as CPUs, GPUs e SoCs (*Systems on a Chip*) de projetos licenciados da ARM. Algumas dessas empresas utilizam essas unidades fabricadas em seus próprios produtos, e algumas comercializam as unidades fabricadas para outras empresas e para o público em geral. Nos últimos anos, a ARM vem trabalhando em um chip de 64 bits, e suas licenças estão próximas de ter o hardware ARM de 64 bits pronto.

Um dos engenheiros da ARM determinou que 1400 módulos de código fonte diferentes em qualquer Ubuntu ou Fedora (ou ambos) possuem linguagem Assembly no código. Isso não quer dizer que a linguagem Assembly (ou a falta dela) irá interromper o módulo de trabalhar no sistema ARM64, pois pode haver código fallback em nível mais elevado (por exemplo, código escrito em C) que tomará a frente e será compilado para a linguagem Assembly ausente no ARM64. No entanto, os módulos não foram testados e verificados nem no hardware atual, ou sobre os emuladores para a arquitetura ARM64 existente atualmente. Assim, a Linaro decidiu recorrer à comunidade para portar alguns desses módulos e criou um concurso com prêmios para as pessoas dispostas a ajudar na causa [2].



Os engenheiros também notaram que uma grande parte do código que contém linguagem Assembly era bastante antigo. Foi projetado em uma era em que os sistemas tinham uma CPU única; as CPUs eram muito mais lentas, com um único core; a memória foi medida em megabytes, não em gigabytes; A Ethernet era 10Mbps, não 1,000Mbps; e os compiladores GNU não foram tão bons em otimização como compiladores comerciais. Portanto, as pessoas escreviam assembler para as partes mais rápidas e "enxutas" do sistema. Se esses programas fossem escritos hoje, no entanto, eles poderiam ter menos linguagem Assembly, e o código seria mais portável. Assim, o concurso foi ampliado para incluir a melhoria do desempenho destes módulos e (talvez) eliminar algumas palavras da linguagem Assembly antiga onde fazia sentido.

Sistemas embarcados exemplificam como a nossa perspectiva de "desempenho" mudou ao longo do tempo, na medida em que o tamanho do espaço de memória é muitas vezes uma medida de desempenho, com uma pequena economia de recursos proporcionando uma economia no processo de fabricação. A bateria de longa duração, desenvolvida ao permitir que partes do sistema possam ser desligados após a conclusão do aplicativo, também representa uma melhoria no desempenho. Em grandes fazendas de servidores, o desempenho é frequentemente medido em economia de energia eléctrica, de cooling ou na redução de compras de equipamentos e espaço físico.

Nos primeiros anos da minha carreira como programador, meu trabalho não era escrever um novo recurso, mas fazer com que programas desenvolvidos por outras pessoas funcionassem "melhor". Meu gerente me disse que se eu não conseguisse fazer com que o aplicativo trabalhasse na metade do tempo, era para não me preocupar com isso. Em quase todos os casos, eu poderia fazer um aplicativo executar não só pela metade do tempo, mas muitas vezes de cinco a 10 vezes mais rápido. Foi um trabalho muito gratificante, por isso tem sido interessante começar a investigar novas técnicas para perfis de código, encontrar problemas de desempenho, e buscar novas me-

lhorias e eficiências que podem ser feitas desde que concluí esse trabalho, há 30 anos.

Ao mesmo tempo, tenho trabalhado com alguns sistemas muito pequenos que possuem algumas características muito interessantes. O uso de GPUs para computação, chips de processamento de sinal digital, e field-programmable gate arrays (FPGA) eram todos conceituais anos atrás, mas eram proibitivos em custo e espaço. Estes conceitos tornaram-se agora não só viáveis, mas até competitivos em preço/desempenho com outros tipos mais "mainstream" de circuitos. Uma placa de uma empresa chamada Adapteva não só inclui um SoC com um processador ARM de dois core, FPGA, e chips de processamento de sinais digitais, mas também uma CPU de 16 ou 64 core. Tudo isso, além de alguma memória de sistema e portas USB, vem em uma placa com faixa de preço entre 100-150 dólares [3]. A oportunidade de aprender sobre essas arquiteturas se tornou realidade na prática.

Recentemente algumas pessoas atraíram muita atenção com a construção de um "supercomputador" desenvolvido a partir de um Pi Raspberry, um sistema single-core não convidativo para o tipo de programação que pode ocorrer em um sistema HPC real. Em um sistema HPC, cada placa pode ter várias CPUs ou vários cores em um único processador e usar o OpenMP em conjunto com o MPI e outros ambientes de computação heterogêneos. Substituir computadores, como o Banana Pi [4] ou o ODROID-U3 [5], criaria um desempenho de "supercomputador" com um aumento razoável no preço, e garantiria um mix mais realista de estilos de programação. Eu incentivo os leitores, portanto, a se inscreverem para o concurso da Linaro e ajudar o GNU/Linux a ser o melhor que pode ser. ■

Mais informações

[1] Linaro: http:// linaro.org

[2] Linaro Performance Challenge: http:// performance.linaro.org

[3] Parallela: http:// www.adapteva.com/parallella/

[4] Banana Pi: www. banana-pi.org

[5] ODROID-U3: http:// hardkernel.com/main/products/prdt_info. php