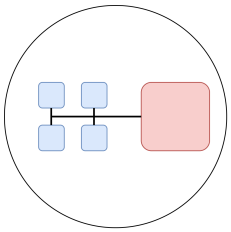
Processadores Heterogêneos



27 de novembro de 2023

Sumário

---

1 Os comos e os porquês

De cara: processadores com arquiteturas heterogêneas são aqueles que executam seus programas em duas ou mais microarquiteturas diferentes. Eles chegaram ao mainstream muitos anos atrás com o esquema big.LITTLE da Arm e finalmente estão nos nossos PC's com processadores Intel.

As razões para isso sempre estão relacionadas com balanço de carga nos processadores para o workload [1] esperado e economia de energia. Desempacotando isso: aumentar o desempenho de determinado sistema sempre

[1] A "carga de trabalho," ou seja, o tipo e a quantidade de aplicações que estão sendo executadas no nosso sistema em dado momento.

exige mais energia, mas os Watts estão escassos nos mundos de celulares e laptops (e nos grandes data centers também); o software para os usuários comuns habitualmente não consegue aproveitar mais de 4 núcleos, fazendo com que meramente adicionar mais núcleos gaste energia sem refletir no melhor desempenho para tarefas do dia-a-dia; só a melhora do desempenho dos processadores é fundamental para o modelo de negócios da indústria de microprocessadores.

Tudo isto não é tanta novidade, contudo, exceto o encapsulamento de processadores distintos no mesmo CI e a execução de um *mesmo código* em mais de uma microarquitetura, de forma transparente, no mesmo sistema. Os *sistemas* heterogêneos em si sempre existiram (veja seção 4), com processadores com diferentes ISA's da principal rodando programas com tarefas secundárias ao do microprocessador central, com códigos próprios.

1.1 Alguns exemplos de ISAs misturadas

Lá nos anos 1960 era comum haver um processador secundário para realizar as tarefas de I/O, pois ninguém queria parar o mainframe para fazer uma impressão de centenas de páginas ou a busca de um arquivo em fita magnética, que facilmente poderia levar vários segundos. Portanto, enquanto o processador principal executava todos os jobs (processos) de centenas de usuários, um outro processador ficava em conversa com ele, gerenciado pelo sistema operacional; este processador rodava um código fixo (era como se fosse um device driver), portanto era inacessível ao programado de aplicações. Este esquema é o que se conhece por *Asymmetric MultiProcessing,* AMP (em contraste com o SMP dos nossos multicores não heterogêneos).

As placas gráficas nos PC's contemporâneos são um exemplo bem mais próximo para nós. Os processadores destas GPU's (dezenas, centenas ou milhares deles em um sistema) são processadores muito simples, visando executar tarefas curtas e repetitivas aplicadas sobre uma grande massa de dados bem estruturados, como pixels ou vértices de modelos 3D, normalmente trabalhando com floats de 16 bits[2] Esta carga de trabalho seria bastante ineficiente em uma CPU normal e iria monopolizar os processadores, por isso ela é terceirizada para outra arquitetura completamente diferente, com outra ISA, outra organização de memória, outro tipo de despacho de tarefas que é independente do sistema operacional. Essa extensão de capacidade do sistema ficou mais popular apenas no século XXI.

As placas-mãe dos PC's, em compensação, têm mais de um processador há muitos anos. Embora no princípio os CI's auxiliares fossem controladoras complexas (ou seja, um circuito digital com um propósito específico, como

[2]As aplicações gráficas e de machine learning não precisam de muita precisão, a primeira por ser referente a uma tela de vídeo e a segunda por ser essencialmente probabilística / estatística. É possível trabalhar com floats de 32 bits, mas este não é o caso típico.

interfacear I/O ou cuidar da temporização), alguns processadores embarcados com ROM's começaram a aparecer. Hoje em dia temos dezenas de processadores em um laptop qualquer, a maioria deles invisível para nós, conhecidos apenas pelos projetistas das placas, como GPU's, NPU's (*Neural Processing Units*), TPM's (*Trusted Platform Modules*), processadores responsáveis pelo boot, pelo controle da bateria e assim por diante.

Podemos até argumentar que desde os primórdios a arquitetura dos PC's era heterogênea, já que tanto o mouse quanto o teclado são habitualmente implementados com algum microcontrolador simples, de 8 bits, mas os periféricos não são o foco dos nossos interesses.

1.2 Mas então tudo depende do workload?

Nosso foco aqui é como executar as tarefas normais de um sistema típico de um PC ou celular de forma a aumentar o desempenho sem comprometer muito o gasto de energia, ou seja, de certa forma ainda estamos brigando com as Leis de Moore e Amdahl.

Se houver outro tipo de tarefas, mais específico, outras organizações de processadores podem ser melhores. Para um carga enorme de deep learning, por exemplo, é melhor ter um cluster com CPU's, GPU's e grandes memórias e HD's, ou seja, o que chamamos de Sistema Distribuído, com vários nós se comunicando através de uma rede. Tentar fazer deep learning num computador caseiro costuma ser lento e ineficiente.

A própria Intel sugere um caso de uso para os processadores recentes mais parrudos (o que indica o "target" de mercado que eles pretendem atingir), que são usuários com demanda muito alta de processamento simultâneo, como o sr. Thiago Gamer Streamer dos Santos, que faz lives com jogos pesadíssimos rodando em 4K com 120 fps e ainda por cima grava a transmissão localmente. Se você não está familiarizado, basta dizer que todas estas tarefas exigem muito processamento, então é necessário ter processadores rápidos, mas há muitas tarefas secundárias (o chat online, por exemplo) que podem rodar em processadores mais lentos e econômicos. Voltaremos ao sr. dos Santos mais à frente.

1.3 Os modelos da Arm e da Intel

Examinando as variações da Arm e a proposta da Intel, podemos desenvolver um pouco melhor uma intuição sobre os objetivos de microprocessadores com núcleos heterogêneos. A lógica em si é simples: quando 4 núcleos não têm desempenho suficiente e 8 núcleos comuns gastariam muita energia, colocamos 4 núcleos rápidos e que consomem muita energia e 4 núcleos lentos que economizam energia, alcançando um ponto intermediário entre um quadcore e um octacore comuns. Mas a verdade é que temos bastante variação dentre estes modelos.

2 Arm big.LITTLE

Os processadores Arm sempre foram líderes do mercado nas CPU's de telefones celulares. Na época do primeiro iPhone, as CPU's eram apenas um núcleo rodando em torno de 500 MHz e, portanto, não consumiam tanta energia assim, embora a duração da bateria tenha sido uma preocupação desde sempre. No momento em que temos quadcores com 1 GHz, não podemos mais negligenciar o consumo e precisamos fazer otimizações; para um ponto de comparação, meu velho Motorola G5 de 2017, que já não era topo de linha, tem 8 cores rodando com clock entre 800 MHz e 1,4 GHz, ajustado de acordo com a carga de trabalho.

Neste Motorola, temos 4 núcleos rápidos ("big's") que consomem bastante energia e estão ali para os casos de pico de demanda de desempenho, e 4 núcleos econômicos ("LITTLE's") que executam tarefas mais simples e menos urgentes, mas que podem ser numerosas. Este tipo de classificação de tarefas é bastante compreensível para os celulares, que normalmente passam a maior parte do tempo em *idle,* basicamente esperando algum evento de interesse, como a recepção de uma chamada ou mensagem ou o início de uma interação com o usuário. Nestes momentos, o processador terá bastante trabalho e provavelmente vai forçar a frequência de clock a aumentar por um curto período de tempo, para dar uma resposta mais rápida.

2.1 Modelos de chaveamento

Temos três formas de utilização de um processador big.LITTLE, mas o princípio é sempre o mesmo: em situações de alta demanda, seja prevista ou mensurada instantaneamente, os processadores "big" são escolhidos; quando a demanda passa, os "LITTLE" são preferidos. Quem faz esta escolha é o sistema operacional (ou seja, o Android ou o iOS, seja em celulares ou tablets), que monitora constantemente a situação.

No modo mais simples (fig. 1), temos dois clusters, um de alto desempenho e outro de baixo desempenho, e o sistema operacional simplesmente escolhe um deles para usar, deixando o outro cluster efetivamente inoperante. Portanto, temos um modo de alto desempenho e um modo de economia de energia, que vão se alternando conforme a carga de utilização dos processadores varia, sendo esta medida pelo sistema.

É interessante notar que o processo de chaveamento de um cluster para o outro é transparente para os programas (ou seja, o aplicativo migra de um processador para outro e nem sabe que isto ocorreu) e é extremamente rápido, considerando-se a complexidade de uma operação destas. [3]

Um segundo modo de operação, *task migration,* trabalha com pares de processadores, um de alto desempenho e outro econômico, e o sistema es-

[3] O power-up de um processador comum costuma levar uma eternidade, então não podemos simplesmente cortar a alimentação e religar quando conveniente

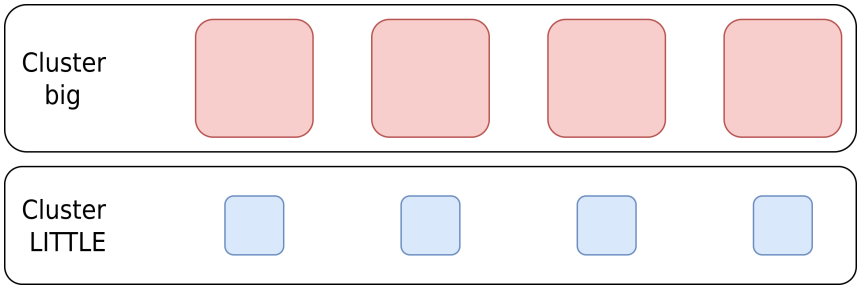


Figura 1: Arquitetura ARM big.LITTLE no modo de chaveamento de clusters.

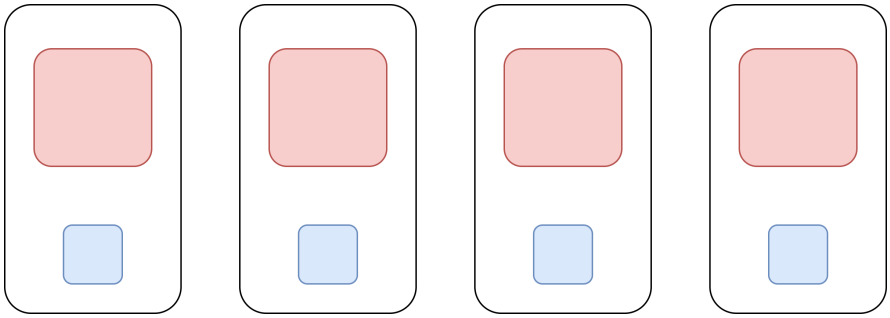


Figura 2: Arquitetura ARM big.LITTLE no modo de migração de tarefas ou seleção dentro de um par. Para cada par, apenas um processador, o vermelho ou o azul, estará funcionando, com o outro em standby.

colhe apenas um destes para estar ativo em dado instante, de acordo com as necessidades da carga de trabalho sendo executada. Este modo é ilustrado na figura 2. Esta organização é muito mais flexível do que a anterior, embora nos casos limites seja igual, ou seja, no máximo desempenho temos ativos apenas processadores poderosos, enquanto que no modo mais econômico temos apenas os que gastam menos. Aqui podemos ter metade dos processadores em alto desempenho, ou todos, ou apenas um, ou nenhum, conforme a necessidade, dando mais granularidade à escolha.

Ambos os esquemas apresentados possuem uma falha grave de marketing, no entanto: o sistema operacional vai detectar e utilizar metade dos microprocessadores físicos disponíveis em um chip! Se tivermos oito núcleos presentes na pastilha de silício, sendo quatro big's e quatro LITTLE's, o sistema irá utilizar apenas quatro em qualquer momento, e vai reportar a existência de apenas quatro ao usuário.

A alternativa que permite ao sistema usar todos os processadores é *não deixar nenhum processador inoperante.* Neste modo, denominado de multiprocessamento heterogêneo, o sistema operacional faz duas coisas principais: escalonar os processos (determinar em qual processador cada programa de-

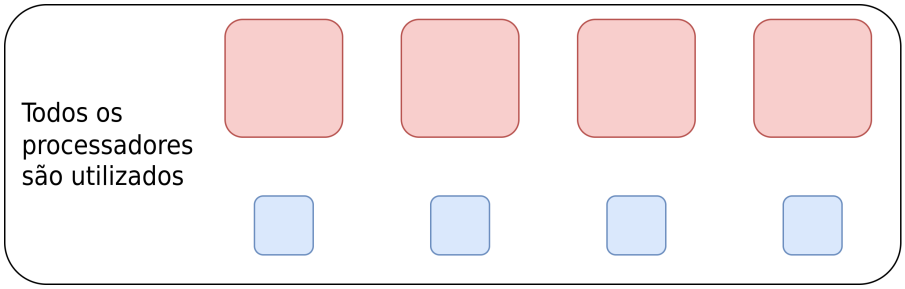


Figura 3: Arquitetura ARM big.LITTLE no modo de utilização heterogêneo de todos os processadores físicos.

verá rodar) de acordo com a necessidade de processamento deles e ajustar individualmente a frequência de funcionamento de cada processador. Meu bom e velho Moto G5 octacore tem 4 cores big e 4 LITTLE e, examinando a ocupação dos processadores, é fácil ver a variação brutal de frequência de acordo com as tarefas sendo executadas. [4]. A figura, meio idiota, que ilustra isso é a fig. 3.

É claro que oito núcleos consomem, de forma geral, mais energia do que quatro núcleos, mas com boas medições e ajustes bem feitos consegue-se um bom balanceamento dos trade-offs envolvidos.

Os Arms mais recentes podem ter até 3 microarquiteturas no mesmo sistema, que no caso limite são um LITTLE, um big e um "extreme"; este extreme (os Cortex-X) são projetados para máximo desempenho sem se preocupar com economia ou eficiência de energia nem de área no silício.

2.2 Desenvolvimento de processadores Arm

Como um aparte a esta seção, vamos examinar muito brevemente uma questão óbvia para consumidores de eletrônicos: por que os celulares da Apple e Samsung parecem tão mais rápidos do que a concorrência, se todos os telefones usam a mesma ISA Arm? Por que os chips Qualcomm são tão melhores do que os outros? A resposta é multifacetada, mas se resume a desenvolvimento *in-house* e a patentes e propriedade intelectual, gerando grandes diferenças na implementação e variações na microarquitetura. [5]

A Arm trabalha "meramente" com licenciamento de propriedade intelectual (I.P.), ou seja, a companhia em si não produz nenhum circuito integrado físico, restringindo-se a ceder a tecnologia que possui para terceiros, cobrando taxas de licenciamento fixas (e milionárias) e, em cima destas,

[4] Alguns aplicativos, como o CPU-Z, permitem acessar estes dados

[5] A Qualcomm em particular possui centenas de patentes em tecnologias fundamentais de telecom e técnicas auxiliares, que às vezes garantem exclusividade de uso e desempenho otimizado.

royalties de alguns centavos por microprocessador fabricado. [6]

Ela tem vários modelos funcionais, incluindo repassar o circuito lógico pronto de um microprocessador (que deverá ser instanciado em silício, o que tem seus desafios particulares), ou então permitir alterações e adaptações a este circuito feitas pelo cliente e fornecendo uma consultoria para isso, ou então licenciar a ISA de alguma versão de seus processadores, deixando toda a implementação a cargo do licenciado.

Tanto a Samsung quanto a Apple trabalham com o desenvolvimento dos seus processadores com laboratórios e pessoal de tecnologia de ponta, integrando os componentes periféricos, funções e barramentos e aproveitando todas as oportunidades de otimização que um caso específico permite, ao invés de apenas usar o circuito genérico do IP provido pela Arm. Assim, todos os parâmetros de seus processadores podem ser alterados para combinar melhor com as memórias, controladoras, GPU e sistema operacional, já que eles têm controle sobre todos estes módulos, ao contrário de uma fabricante menor, que possui laboratórios com tecnologia um pouco defasada e não podem customizar tudo devido aos preços envolvidos.

É assim que a Apple, sempre querendo imprimir sua personalidade, rebatizou os núcleos big e LITTLE como Firestorm e Icestorm. Isso é marketing em cima do desenvolvimento real, mas é bem melhor do que a Intel fez.

3 Intel 12ª geração e além

Enquanto a Arm sempre declarou economia de energia como objetivo em seus núcleos heterogêneos, a Intel podia se dar ao luxo de aumentar o desempenho dos seus multicores homogêneos, já que ela sempre foi insignificante no mercado *mobile* e apenas uma participante menor nos mercados de tablets (já nos laptops, as ISAs Arm só foram realmente entrar no mercado com a Apple em 2021, e as baterias são bem maiores, portanto não sendo tão significantes quanto nos outros nichos).

No entanto, o aumento da quantidade de núcleos e das frequências encontrou um impasse severo: *o software "típico" não iria conseguir aproveitar eficientemente este aumento de desempenho previsto.* Em outras palavras, teríamos um CI que gastaria mais energia do que os antigos mas sem trazer ao usuário um aumento perceptível na velocidade de execução.

Por mais que isso pudesse ser inferido pelas decisões empresariais, a Intel

[6] Pela Anandtech: os royalties ficam entre 1% e 2% do preço do componente, sendo que o custo dos processadores varia: nos celulares, um Qualcomm custa U$40, um MediaTek U$17 e um Unisoc sai por U$6, mas lembre que se vendem mais de 1 bilhão deles por ano. Já o licenciamento em 2013 variava entre 1 a 10 milhões de dólares (e o modelo de "assinatura" um múltiplo disto, mas em 2018 existiam modelos mais simples, em torno de U$ 100k). Veja mais detalhes em `https://www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works/2`. Em 2023, a Arm anunciou que mudaria seu licenciamento.

declarou publicamente esta posição num whitepaper, [7] de certa forma culpando o software por não conseguir aproveitar melhor o hardware. O fato que gera esta consequência foi enunciado, em paráfrase, como: "a maioria esmagadora dos programas não consegue escalonar além de 4 núcleos executando simultaneamente;" este platô de desempenho significa que ir além de 8 núcleos não deve trazer nenhum ganho sensível de desempenho para quase todos os consumidores (à exceção do sr. Thiago Gamer, que ficaria feliz com 128 cores, se pudesse).

3.1 Os núcleos para desempenho e os núcleos para eficiência

A solução da Intel foi criar um CI heterogêneo, dividindo-o de forma similar à dos Arm, com núcleos focados em desempenho denominados P-cores (de *performance*) e núcleos otimizados para economia de energia chamados de E-cores (de *efficient*). Talvez o departamento de marketing estivesse despreocupado, porque apenas nerds vão ouvir falar desta distinção e não precisam ser convencidos, mas desconfio que nerds são o público-alvo de tecnologia... Enfim.

Os P-cores são processadores projetados para tarefas pesadas, trabalhando com uma frequência máxima mais alta, CPI alta e com todo tipo de otimização para alto desempenho, o que faz com que os circuitos fiquem maiores (pois as otimizações de desempenho costumam exigir circuitos auxiliares com grande quantidade de portas). Já os E-cores devem economizar energia, maximizando o *desempenho por watt* e, portanto, precisam minimizar os circuitos e trabalham em frequências menores, tudo isso voltado a tarefas mais leves, desocupando os núcleos P.

Uma das vantagens dessa assimetria é a quantidade de núcleos E que podemos colocar: se acreditarmos na legenda da foto do silício de um Intel i9-13900K [8] , a área de um único núcleo P é equivalente à de quatro núcleos E. Portanto o uso destes núcleos mais simples aumenta muito o número total de threads disponível no mesmo chip.

3.2 Como o sistema operacional organiza isso?

O algoritmo utilizado exige, logicamente, uma coordenação entre o microprocessador e o sistema operacional. [9] O sistema operacional irá levantar

[7] `https://cdrdv2-public.intel.com/685861/211115_Hybrid_WP_1_Introduction_v1.2.pdf`

[8] Lá em `https://en.wikichip.org/wiki/File:intel_raptor_lake_die_(8%2B16)_(annotated).png`.

[9] Os principais sistemas atuais no mercado de consumo são desenvolvidos em paralelo com os novos processadores, até porque temos essencialmente apenas três grandes empresas – Arm, AMD e Intel – se articulando com dois sistemas – Windows e Unix-like. Deste modo, mesmo combinações como Apple com iOs ou Samsung com Google Android não irão apresentar tanta diferença assim.

informações sobre o workload atual, estimando a necessidade de cada processo (chutando de acordo com o que está ocorrendo, é uma heurística); parte destas informações será dada pelo processador em si, que fornece algumas medidas para esta estimativa. O controle da frequência de cada processador e a atribuição de um processador a um programa é feito dentro deste processo de escalonamento e *throttling* adaptativo ("aceleração") da frequência de clock individual de cada processador.

Mas os passos para escalonar um novo processo a executar são simples:

1. Se um P-core está ocioso, coloque o processo nele;
2. Se nenhum P-core está ocioso, os processos mais leves serão executados nos E-cores;
3. Se todos os P-cores e E-cores estão com no mínimo um processo rodando, então os P-cores vão começar a funcionar com Hyperthreading, executando dois processos simultaneamente em um único núcleo físico que funciona como dois núcleos lógicos.

Este esquema visa tentar atingir diversos objetivos, dependentes da carga de processamento:

- todos os processos que demandam alto desempenho devem ter um P-core associado se possível, o que é especialmente útil em casos comuns de picos transitórios de alto processamento;
- todos os processos secundários, de baixa exigência ou urgência, devem rodar em E-cores sem atrapalhar o desempenho dos P-cores para workloads médios e pesados;
- em caso de workloads muito leves, eles serão executados em P-cores o mais rapidamente possível, talvez na frequência mínima de operação, e portanto o CI vai consumir o mínimo de energia possível;
- nos workloads muito pesados, todos os núcleos estarão ocupados; os processos dos P-cores irão perceber uma queda relativa de desempenho devido à divisão de recursos por conta do Hyperthreading.

Desta forma, garantimos ganhos de desempenho perceptíveis ao usuário, em relação às gerações anteriores de processadores, sem onerar muito o consumo de energia.

Fora destes sistemas, a conversa é outra; há bastante variação em uma miríade de nichos diferentes.

4 O nosso PC de cada dia

Dentro de um computador típico (ou telefone celular), há tarefas bastante pesadas e complexas, em especial aquelas que controlam a passagem de grandes volumes de dados entre dois elementos. Algumas são implementadas com microprocessadores, outras não.

Controladores de barramentos que possuem protocolos mais simples não necessariamente precisam de um processador embutido e podem ser implementados diretamente com circuitos lógicos, o que contribui para diminuir uma possível latência na operação. Isso inclui o protocolo USB que, dependendo do caso, pode ser implementado diretamente com lógica ou então com um processador e um firmware simples.

Outros casos não têm escolha. Para os discos rígidos, podemos ter *mais de um* microprocessador, já que os protocolos de comunicação com a placa mãe não são tão simples, há grande quantidade de informações lidas especulativamente e guardadas em buffers, escritas também são ordenadas e bufferizadas, e o controle dos acionamentos dos motores e cabeças também não é nem um pouco simples (exigindo um controlador próprio para os métodos sofisticados necessários).

Pen drives e SSD's também exigem bastante controle autônomo da alocação das informações – setores estragados são automaticamente realocados de forma transparente, por exemplo. Novamente, buffers enormes são utilizados para acelerar as requisições e devem ser gerenciados pelo circuito, e um microprocessador pode ser vantajoso nesta função.

Até tarefas mais simples exigem outros processadores. O processo de carga da bateria de laptops costuma ser feito com um microcontrolador ARM econômico, de baixo desempenho (antigamente se usavam processadores de 8 bits, como os AVR's dos menores Arduinos); ele provê algumas funções ao sistema operacional. Mouses e teclados utilizam processadores de 8 bits há décadas. O processamento de áudio embutido tem um processador dedicado que pode ser visto soldado na placa mãe dos laptops. Claro que nenhum destes programas é exposto ao sistema, que apenas utiliza estes componentes como se fossem subcircuitos quaisquer. Portanto *o sistema em si* foi sempre heterogêneo, com vários processadores completamente diferentes; mas o processamento principal não era, já que os programas que o usuário executa sempre rodavam exclusivamente na CPU homogênea x86.

Cabe citar as já mencionadas placas de vídeo como um assunto à parte destes. As GPU's (Graphical Processing Units) incluem dezenas ou centenas de processadores muito simples, mas com altíssima capacidade de paralelismo. Ao contrário dos casos anteriores, estes processadores, com ISA muito diferente da x86-64, rodam sim programas dos usuários. Embora a execução não seja transparente (ou seja, tem que haver um código específico escrito para a GPU, que não pode rodar um programa comum feito para o x86-64), ao utilizarmos a placa de vídeo dedicada estamos efetiva-

mente usando computação heterogênea, com um aplicativo se desdobrando em código para as CPU's e códigos distintos para a GPU.

5 Os processadores dentro de processadores

Certas subtarefas dentro de um processador são bastante complexas, requerendo circuitos de controle específicos e grandes. Como alternativa, podemos substituir alguns destes circuitos por microprocessadores dedicados com um firmware próprio.

5.1 Bootloaders

Antigamente nos PC's, o boot era feito por um trecho de código na BIOS (*Basic Input/Output System,* código em ROM que incluía as funções básicas para acessar o hardware), executado no power-on e no reset. Hoje em dia, nós dizemos que a BIOS foi substituída por sua versão mais moderna, a UEFI (sigla meio imbecil: *Universal Extensible Firmware Interface*), que é atualizável por estar localizada em uma flash RAM. Mas isso é um pouco simplista.

O procedimento de boot se tornou muito complexo e inclui atualizações seguras de firmware, recuperação do sistema após um crash, acesso a múltiplos dispositivos como possível origem do sistema operacional (pen drives, memórias flash, HD's e SSD's, DVD's, dispositivos remotos – todos exigindo drivers para o acesso e um sistema de arquivos), menus básicos de configuração com gráficos e mouse, tudo isso além do procedimento de carregar o sistema operacional em si através da execução do bootloader.

A certa altura, ficou mais fácil simplesmente ter um processador completo para realizar todas estas tarefas. A Intel usava um ARM 32 bits até uns anos atrás (o que parece meio irônico), mas hoje em dia utiliza um x86-32 com uma versão customizada do MINIX 3 [10] como sistema operacional, denominados Intel Management Engine (IME), localizado no chamado chipset. É como ter um computador para bootar outro computador. Ele é considerado uma possível vulnerabilidade de segurança (a Intel não divulga direito seus detalhes, o que configura "segurança por obscuridade"), tendo comportamentos meio assustadores – ele pode entrar em execução mesmo com o PC aparentemente desligado e acessar a rede sem pedir autorização no momento).

A propósito, mesmo o reset e o power-on não são muito simples, já que podemos hibernar e suspender. Os processadores de PC's têm vários estados de economia de energia que vão além de "ligado normalmente" e

[10] Se você não sabe, o MINIX é um sistema educacional que indiretamente deu origem ao Linux. Os bate-bocas antológicos entre o Torvalds e o Tanenbaum ainda podem ser lidos na internet.

"completamente desligado." Estes modos intermediários variam: um deles apenas limpa as caches e paralisa a CPU, outro desliga tudo na placa mãe mas copia o conteúdo da RAM para um arquivo no HD, para habilitar uma partida rápida ao religar a máquina. [11] Tudo isso deve ser gerenciado pela UEFI / IME num PC com processador Intel.

5.2 Trusted Platform Module

Os TPM's são módulos auxiliares para garantir a segurança do sistema, fornecendo primitivas e operações relacionadas a criptografia tais como geração de chaves secretas, encriptação e decriptação de dados, gerador randômico verdadeiro, verificação de integridade de firmware e outros. A ideia é fornecer ao processador principal um "domínio seguro" inacessível a um programa normal; por exemplo, a chave secreta gerada se encontra num registrador que não está ligado diretamente a nenhum circuito da CPU, sendo usada por outras funções do TPM.

Este módulo pode ser implementado de várias formas: pode ser um processador fisicamente externo ligado à CPU, ou pode estar na mesma pastilha de silício, ou pode ser um hardware discreto (não um processador) implementado dentro dela ou pode mesmo ser uma parte da CPU desconectada das outras partes (p.ex., o registrador com a chave secreta fica ligado apenas aos módulos de encriptação e decriptação, sendo invisível ao restante do circuito).

Particularmente interessante para segurança de sistemas é a verificação de integridade, que funciona de maneira análoga aos testes com checksum MD5 ou SHA-2 para arquivos instaláveis baixados da rede. O circuito de TPM calcula um hash de todo o firmware, abrangendo código e configurações, e o compara com o último hash utilizado; a alteração de qualquer bit, potencialmente maliciosa e catastrófica, é praticamente impossível de ser acobertada. Atualizações de firmware também precisam fornecer hashes e chaves criptográficas válidas, caso contrário serão recusadas.

5.3 Outros

Várias tarefas secundárias podem ser implementadas com microprocessadores dentro do sistema. Normalmente a vantagem é o aumento de desempenho, para tarefas em que carregar o processador principal não é interessante.

Um exemplo já visto é o medidor de desempenho. Não possuímos muitos detalhes sobre as suas características, mas é razoável supor que os principais motivos para não implementá-lo diretamente em lógica são a complexidade

[11] É anti-intuitivo, mas *reiniciar* um PC faz com que todo o procedimento de boot seja realizado, desde o início; já *"desligar"* um PC pelo menu do sistema operacional pode não desligar tudo, salvando parte dos dados atuais e não realizando a carga do sistema a partir do zero na próxima vez.

e a flexibilidade desejados, já que as métricas envolvem uma amostragem direta do mix de instruções sendo executado a cada instante dentro do processador superescalar. Sendo efetivamente um programa, ele pode ser muito facilmente configurado por algumas instruções assembly da CPU ou por atualizações de firmware.

As NPU's (*Neural Processing Units*) estão se popularizando bastante. Trata-se de um "processador de arquitetura não-von Neumann" [12] dedicado a fazer inferências típicas de redes neurais para acelerar as aplicações que utilizam *machine learning*. Embora o processador principal usualmente tenha instruções vetoriais (AVX nos x86-64, NEON nos ARM) que aceleram estes casos, elas ainda assim limitam bastante o paralelismo máximo possível. Como o caso típico são operações comuns de ponto flutuante ou ponto fixo, normalmente com poucos bits de resolução, mas que são executadas às centenas ou milhares, podemos implementar um circuito simples com registradores que guardam estes números e suas interligações e que executa somas, multiplicações e comparações às centenas por cada clock. Obviamente, as tarefas que puderem usar este processador terão uma aceleração brutal no desempenho.

[12]Estas unidades são chamadas de processadores e tratadas como tal (afinal, parte da carga de trabalho da CPU é repassada a elas), mas notem que elas *não são processadores genéricos*, ou seja, não são máquinas Turing-completas para computação de propósito geral: servem para um tipo específico de problema, com regras rígidas de computação. Talvez fosse melhor considerá-las como **coprocessadores,** ou seja, circuitos anexados ao processador para realizar mais rapidamente tarefas secundárias. Pesquise o coprocessador aritmético de ponto flutuante 80387 e suas variantes, usados antigamente na ISA x86-32.