Capítulo 4

Desafios para a Computação Energeticamente Eficiente

Luigi Carro e Gabriel Luca Nazar

Programa de Pós-Graduação em Computação (PPGC) - UFRGS

Abstract

New and relevant applications impose great challenges to computational systems, since they demand complex processing applied to large and growing amounts of data. With the weakening of Moore's law seen in recent years, new approaches are needed to deliver the required performance. An additional challenge is the need to contain the growing energy consumption of computing devices and infrastructures to reduce operating costs, for environmental reasons and, for mobile devices, also due to battery limitations. In this chapter, we will address the challenges of developing such systems and possible solutions, traversing the multiple levels of abstraction in computing.

Resumo

Novas e relevantes aplicações impõem grandes desafios a sistemas computacionais, já que demandam processamento complexo aplicado a grandes, e crescentes, quantidades de dados. Com a perda de força da lei de Moore observada nos últimos anos, novas abordagens são necessárias para oferecer o desempenho necessário. Soma-se a esse desafio a necessidade de contermos o crescente consumo energético de dispositivos e infraestruturas computacionais para redução de custos operacionais, por questões ambientais e, em dispositivos móveis, também por limitações de bateria. Nesse capítulo, iremos abordar os desafios de desenvolvimento de tais sistemas e possíveis soluções, atravessando os múltiplos níveis de abstração da computação.

Vídeo com a apresentação do capítulo: `https://youtu.be/OsXT15-deO8`

4.1. Introdução

Enquanto a lei de Moore esteve presente, as aplicações da informática floresceram, de tal modo que a vida cotidiana hoje em dia é fortemente baseada em algum tipo de iteração com computadores, ocultos ou explícitos. A escalada tecnológica (ou a lei de Moore) permitiu a integração de mais dispositivos em um único circuito integrado, com benefícios em velocidade de operação e energia. À medida que a lei de Moore perde a força, para que a evolução permanente da Computação se mantenha, deve-se encontrar um substituto para esta melhoria tecnológica, que forneça as mesmas vantagens. Isto significa que, à medida que o problema aumenta, é preciso permitir a escalabilidade do hardware para resolver um problema maior em um tempo constante, ou para resolver o mesmo problema em um período de tempo menor, sem alterar a base do software ou mesmo exigir modificações significativas no software.

Apenas fornecer mais hardware para execução paralela, por exemplo, claramente não é a solução escalável que se precisa. O paralelismo não favorece energia, apenas EDP, ou o produto de energia vezes tempo de execução (ou seja, executando mais rápido com a mesma energia gasta, ou gastando menos energia no mesmo tempo de execução). Além disso, são necessárias modificações severas ou mesmo radicais no software para que o paralelismo possa ser explorado em diferentes aplicações.

As aplicações atuais do domínio de Computação são caracterizadas pelo seu enorme tamanho, medido em milhões de linhas de código, como apresentado na Tabela 4.1. A transposição dessas aplicações para uma versão paralela é uma tarefa mais do que hercúlea. Sem o impulso da tecnologia, como a comunidade poderá sustentar a crescente demanda por mais desempenho?

Tabela 4.1. Tamanho, em linhas de código fonte, de diferentes aplicação. Fonte: [Desjardins 2017]

Aplicação	Milhões de linhas de código
Linux kernel 2.2.0	2,0
Windows 3.1 (1992)	2,3
Drone militar dos EUA	3,5
Photoshop C.C.6	4,5
HealthCare.gov	5,0
Google Chrome	6,0
Boeing 787, somente aviônica e sistemas de suporte on-line	6,5
Chevy Volt	10,0
Android	11,5
Boeing 787, total	14,0
Caça F-35	24,0
Carro de alto padrão	100,0

Ao mesmo tempo em que se vivencia uma crise tecnológica, pela ausência da lei de Moore como todo o setor de Ciência da Computação estava acostumado pelos últimos 40 anos, tem-se outro problema real e premente: o consumo energético excessivo, tanto

de aplicações em nuvem quanto de aplicações portáteis. Muitas vezes estes domínios aparecem na mesma aplicação. Por exemplo, um estudo recente mostra que se todos os usuários do Google usassem seu serviços de reconhecimento de voz por apenas 3 minutos por dia, toda a capacidade computacional de seus servidores teria de dobrar [Jones 2018]. Mundialmente, ao redor de 12% da energia total é gasta em datacenters e computadores pessoais, celulares e TVs, e este número tende a subir até 20,9% até 2030 [Jones 2018].

Do ponto de vista da Ciência da Computação, para reduzir os custos de datacenters e para aumentar a duração da bateria de celulares, tem-se de focar, neste momento tecnológico, no desenvolvimento de soluções de baixa energia e alto desempenho. Isto porque, embora o problema de consumo seja muito importante, o suporte que a evolução do hardware forneceu nos últimos 40 anos não estará mais presente, pelo fim da lei de Moore, como mencionado acima.

Como mostra a Figura 4.1, as camadas de abstração que usamos nos últimos 50 anos ainda estão presentes. Em alguns domínios, ao atravessar essas camadas, resultados favoráveis foram obtidos. Por exemplo, a síntese de alto nível (HLS) [Nane et al. 2016, Coussy et al. 2009] é uma transformação da camada do programa para a camada lógica, cada vez mais popular para desenvolvimento para dispositivos como *Field-Programmable Gate Arrays* (FPGAs). Infelizmente, embora a HLS tenha existido durante anos, ainda existem entraves, como a necessidade de inclusão manual de diretivas de otimização para obter implementações de melhor qualidade [AMD 2023a, Cong et al. 2022], que limitam sua aplicabilidade e produtividade em projetos de larga escala. Fazer caches visíveis para o programador também já foi usado como uma estratégia de cruzar camadas no domínio da computação de alto desempenho, para conseguir mais *hits* de cache e aumentar o desempenho, ao mesmo tempo em que se reduz a dissipação de energia. Isto, no entanto, coloca mais pressão sobre a habilidade do programador de entender as camadas mais baixas da pilha de abstração, com severas penalidades em tempo de projeto e custos de software.

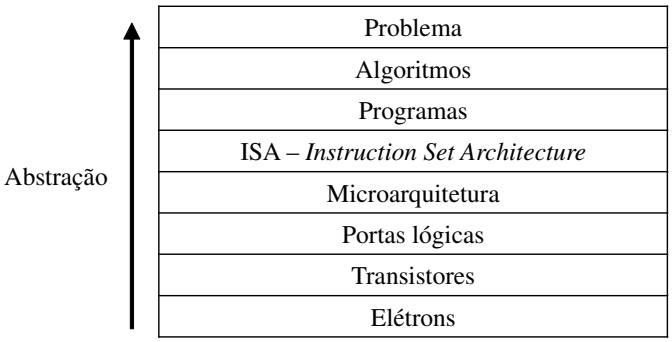

Figura 4.1. Níveis de abstração.

A única maneira de se sustentar a lei de Moore sem tecnologia, e incluindo a eficiência energética tão necessária hoje em dia, passa pela transformação mais eficaz de algoritmos em hardware, atravessando várias camadas de abstração de forma automática. A idéia central é manter-se as abstrações atuais de desenvolvimento de software, ao mesmo tempo em que se fornecem os mesmos efeitos que a evolução da tecnologia traria

ao cenário, explorando habilmente o desenvolvimento de hardware e transformações de software entre as camadas da pilha de abstração.

O que se propõe neste curso é mostrar que é possível o desenvolvimento, para o domínio de aplicações caracterizadas pelo uso intensivo de dados com e sem localidade, de uma mistura de soluções de software e hardware que possam substituir os avanços tecnológicos com os quais a comunidade de software se acostumou. A fim de proporcionar escalabilidade e melhor desempenho, é preciso investigar e integrar diferentes áreas.

Primeiro, deve-se revisar e discutir a arquitetura atual do sistema, de modo que os aceleradores para domínios específicos possam ser mais facilmente desenvolvidos e integrados ao sistema de computação. Geralmente, aplicações enormes e complexas possuem diferentes partes, cujo comportamento é muito diferente. Se, para cada parte de código é necessário desenvolver um acelerador, isso significa que cada acelerador deve ser facilmente integrado na plataforma de hardware, sem comprometer as outras partes de software. Além disso, um segundo passo importante é observar que, uma vez que a integração deve ser perfeita para o programador, um conjunto de ferramentas que possa ajudar essa adaptação da plataforma deve ser usado para suportar esta execução combinada, de modo que a tarefa de programação em si não seja aumentada. Por fim, destaca-se que o paradigma de computação na nuvem, ou, de forma mais generalizada, de *offloading* de carga computacional para uma infraestrutura compartilhada, é um modelo de sucesso que dificilmente se tornará obsoleto no futuro próximo. Assim, soluções para oferecer processamento energeticamente eficiente devem estar cientes da natureza distribuída dos recursos computacionais, dos custos e limitações de cada nodo computacional e da comunicação entre eles.

A crescente demanda por aplicações computacionalmente custosas em dispositivos de baixo custo e com severas restrições em capacidade de processamento e armazenamento popularizou, ao longo dos últimos anos, o uso de *cloud computing*. Essa abordagem permite que tarefas complexas sejam realizadas em grandes infraestruturas de alto desempenho, aliviando a carga computacional dos dispositivos finais dos usuários. Aplicações, por exemplo, de realidade aumentada e de aprendizado de máquina beneficiam-se de cloud computing para serem oferecidas em dispositivos de baixo custo, como aqueles que compõem a Internet das Coisas (*Internet of Things* - IoT). Quando tais aplicações demandam baixa latência, entretanto, cloud computing pode ser uma solução inviável, devido à grande distância entre os recursos computacionais e seus usuários.

A decentralização dos recursos computacionais tradicionalmente oferecidos em cloud computing os aproxima dos dispositivos dos usuários, habilitando seu uso em aplicações que demandam latência baixa e previsível. Além disso, outras vantagens são obtidas, como redução de tráfego no núcleo da rede e potenciais benefícios em segurança. Esse processo de decentralização dá origem a novos paradigmas que podem receber diferentes nomes, como *fog* ou *edge computing* [Satyanarayanan 2017, Long et al. 2018, Shi et al. 2016]. Mais recentemente, uma variação tipicamente chamada *in-network computing* também tem sido investigada [Kianpisheh and Taleb 2023].

Existem, entretanto, variações quanto às definições e à ênfase dada a cada tipo de abordagem por diferentes autores. Em [Satyanarayanan 2017] é defendida uma definição de edge restrita a servidores nas bordas da infraestrutura rede, também conhecidos como

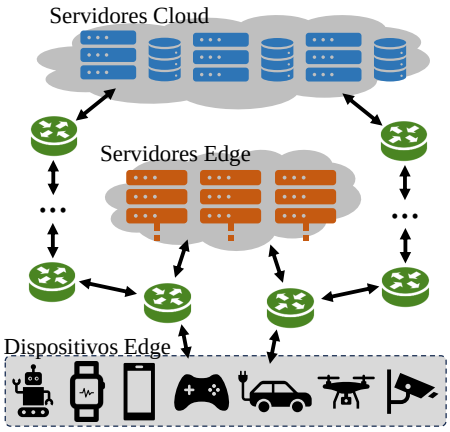

Figura 4.2. Cenário de dispositivos edge, elementos de rede, servidores edge e servidores cloud.

cloudlets [Satyanarayanan et al. 2009], sob o argumento que dispositivos com limitações mais severas de tamanho e potência não oferecem o desempenho necessário. Essa definição, entretanto, exclui diversas abordagens que habilitam aplicações relevantes, como a plataforma apresentada em [Long et al. 2018]. Em [Shi et al. 2016], qualquer dispositivo no caminho entre a fonte dos dados e os servidores centralizados da nuvem pode ser considerado edge. Isso inclui, portanto, tanto dispositivos edge de baixo custo e com severas restrições de energia e potência, como *smartphones* ou *wearables*, quanto cloudlets de maior capacidade dispostos nas bordas da rede, entre os quais as tarefas computacionais podem ser divididas. Esta definição, embora abrangente, permite pouca diferenciação de abordagens adequadas para diferentes nichos de aplicação.

Aqui, adotaremos uma definição similar à apresentada em [Kianpisheh and Taleb 2023], que permite uma distinção clara entre *in-network computing* e *edge computing*. A Figura 4.2 apresenta a relação destes paradigmas: os dispositivos edge produzem e consomem dados, que trafegam através de elementos de rede (em verde), como roteadores e *switches*. Tais elementos podem realizar processamento sobre esses dados, configurando *in-network computing*. Os dados podem, ainda, ser encaminhados até servidores edge, próximos dos dispositivos finais, ou até servidores cloud, de maior capacidade e potencialmente mais distantes. A escolha mais adequada dependerá dos requisitos da aplicação e da disponibilidade de recursos.

Além dessas diversas possibilidades de local, existem diferentes opções para o dispositivo responsável pela realização da tarefa em si, criando efetivamente um cenário de grande heterogeneidade arquitetural. Uma vez que cada dispositivo é mais adequado para aplicações com determinadas características, uma infraestrutura heterogênea potencialmente poderá oferecer o dispositivo mais adequado para cada tarefa, com benefícios em desempenho, custo e consumo de energia [Cooke and Fahmy 2020]. Dentre os dispositivos mais comumente utilizados, destacamos processadores de propósito geral (*General-Purpose Processors* - GPPs), unidades de processamento gráfico (*Graphics Processing Units* - GPUs) e dispositivos reconfiguráveis como *Field-Programmable Gate Arrays* (FPGAs).

4.2. Domínio de Aplicação

A necessidade de maior desempenho ainda está fortemente presente no mercado: veículos auto-guiados, indústria do entretenimento, compras on-line, automação de serviços usando inteligência artifical. Os algoritmos subjacentes que suportam essas aplicações importantes exigem muitos recursos de processamento e dados, e sua futura implantação ou escalabilidade é limitada pela quantidade de hardware que se pode usar para calcular as funções necessárias.

Essas aplicações requerem o processamento de uma enorme quantidade de dados, de forma estruturada (armazenados em banco de dados) mas também desestruturada (armazenadas na web). Os dados estruturados aqui são aqueles com uma alta localidade espacial, que pode ser observada na execução de aplicativos multimídia, por exemplo. Desestruturado significa que os dados estão espalhados na memória, no disco, ou mesmo na rede, sem localidade espacial, e onde uma solução simples usando arquiteturas *Single Instruction Multiple Data* (SIMD) como GPUs não pode ser facilmente implantada.

A arquitetura de processadores, até recentemente, sempre seguiu as necessidades do mercado. O co-processador numérico de ponto flutuante, as instruções MMX, SSE, AVX, todos tornaram-se realidade porque havia uma clara necessidade de mercado. À medida que a lei de Moore continuava, os dispositivos extras que poderiam ser integrados foram usados para suportar melhor os requisitos de software. O primeiro desvio desse padrão ocorreu apenas recentemente, com máquinas *multicore*. À medida que a lei de Moore perdia momentum, e a frequência não podia ser aumentada devido às limitações de fabricação e potência, a indústria de hardware de computação forneceu ao mercado mais núcleos, e não um desempenho maior do processador individualmente. Olhando para o presente e o futuro próximo, o tamanho de dados de aplicativos futuros só pode aumentar. À medida que a automação substitui a fabricação e os serviços, a quantidade de dados que podem ser coletados e analisados pode alcançar facilmente a escala dos terabytes [Jun et al. 2017, Blat et al. 2016, Algur and Sakri 2015].

A análise e implantação de tal dilúvio de dados são atualmente observadas em sistemas de recomendação, propagação de crenças, detecção de fraude, reconhecimento de imagem, processamento de sinais, compressão multimídia e outros. Essas aplicações, por sua vez, popularizaram os algoritmos subjacentes, como redes neurais, aprendizagem baseada em árvores de decisão, decomposição de valor único, decomposição de tensor e autovalor, gradiente descendente estocástico e muitos outros. O que o hardware deve fornecer para essas aplicações, que abrangem domínios tão amplos? Nos exemplos acima, os dados não são apenas grandes, mas também têm muitas lacunas. Ainda, observa-se que para muitas dessas aplicações, há uma tolerância intrínseca a imprecisões nos resultados, ou seja, um resultado aproximado ainda pode ser considerado aceitável. Por exemplo, em aplicações de áudio e vídeo, lidamos com as limitações sensoriais humanas. Já em aplicações de reconhecimento de imagens, divergências numéricas podem ocorrer e ainda assim termos uma classificação correta. A partir de todas essas características, identificamos que qualquer acelerador de hardware para esses domínios deve:

- possibilitar com que os dados sejam processados onde eles são gerados, sem movimentos desnecessários entre processador e memória que drenam energia [Santos

et al. 2021a, Santos et al. 2021b];

- fornecer uma maneira de lidar com dados que, por vezes, apresentam alta localidade (aplicativos de vídeo, por exemplo), onde um SIMD de baixa potência pode suportar o desenvolvimento, mas também acelerar a aplicação onde os dados apresentam uma localização muito baixa (uma característica fundamental da maioria dos algoritmos acessando árvores ou grafos) [de Lima et al. 2022];
- explorar eficientemente possibilidades de redução de precisão para atender diferentes requisitos não funcionais, como uso de recursos [Leipnitz and Nazar 2019b], processamento em tempo real [Leipnitz and Nazar 2019a] ou vazão total de processamento [Leipnitz and Nazar 2020];
- aproveitar oportunidades de computação com tecnologias alternativas, que embora garantam baixa energia, trazem outro conjunto de problemas a serem resolvidos [de Lima et al. 2022, de Lima and Carro 2022];
- prover tolerância a *soft errors*, cada vez mais presentes em tecnologias próximas ao limite da lei de Moore [Nazar et al. 2021a, dos Santos et al. 2022, Nazar et al. 2021b];
- permitir a alteração automática de algoritmos, para aproveitar as características dos dados do problema de modo a realizar oportunidades de computação com baixa energia [Gonçalves et al. 2019].

O mercado atual encontra-se muito concentrado em aplicativos no domínio da análise de dados, aprendizado de máquinas, automação de serviços, onde a quantidade de dados é enorme e a localidade variável dos mesmos dados exclui o uso de uma arquitetura única. Este conjunto de aplicações possui as características comuns que podem ser exploradas, pois:

- são complexas o suficiente para serem significativas como estudo de caso, mas ao mesmo tempo factíveis para uma equipe universitária;
- possuem características próprias de sistemas complexos que precisam ser escaláveis, pelo uso de hardware e software, e por envolver algoritmos de diferentes domínios e uma massa de dados sempre crescente;
- devem ser realizadas com alto grau de automação, para serem controladas por uma equipe pequena de projeto;
- podem ser prototipadas em placas facilmente disponíveis, como GPUs e/ou com FPGAs de alto desempenho, a um custo relativamente baixo.

4.3. Detalhamento do problema

Nesta seção, detalharemos problemas atuais que trataremos nesse curso. Na seção 4.3.1, apresentaremos limitações nas arquiteturas atuais que impõem custos de desenvolvimento e limitam a eficiência dos sistemas computacionais. Na seção 4.3.2, discutiremos dificuldades na adoção de arquiteturas promissoras, como FPGAs, em infraestruturas compartilhadas.

4.3.1. Limitações das arquiteturas atuais

Quando se chega ao fim da lei de Moore, extrair mais desempenho dos atuais processadores torna-se cada vez mais difícil. A exploração do paralelismo utilizando GPUs e múltiplos núcleos fora de ordem, que são as soluções atuais propostas pela indústria, são claramente viáveis se e somente se a tecnologia continuar a permitir mais núcleos e/ou maior frequência, e, para ambos os casos, as perspectivas tecnológicas são desafiadoras. A tecnologia pode ajudar, mas em diferentes setores: a integração com biologia, sensores, processamento quântico, processamento com tubos de nanocarbono e outras técnicas estão no horizonte. No entanto, em cada exemplo, perde-se o que tem sido o estilo fundamental de desenvolvimento de hardware e software, que é a integração total de dispositivos de computação em um único dispositivo.

Outra solução atual é o uso extensivo do paralelismo. No entanto, existem vários impedimentos para esta abordagem. O primeiro e mais óbvio é o desafio de desenvolvimento de programas paralelos. A maioria dos programadores não sabe como programar em paralelo, e muitas aplicações são simplesmente muito complexas para serem convertidas em uma versão paralela. Outro aspecto importante é que, ao usar máquinas paralelas, o ganho é limitado pelo número de processadores ou unidades paralelas, e ainda é mais limitado pela lei de Amdahl (a parte serial dominará o tempo de execução). Além disso, em termos energéticos, o paralelismo é inócuo: como $Energia = Potência \times Tempo$, se alguém pudesse usar N processadores, o melhor fator de redução de tempo seria N, mas a energia gasta é a mesma, já que os N processadores vão dissipar N vezes mais potência. Dadas as limitações reais impostas pela lei de Amdahl, é provável que a energia aumente. Em um cenário onde as demandas de aplicativos aumentam e o desenvolvimento de hardware encontra-se estagnado, há apenas uma maneira de aumentar o desempenho com menor energia: mudar o tempo de execução, reduzindo os estrangulamentos realmente presentes nas transformações de algoritmos a serem executados em um determinado dispositivo de hardware.

Neste curso esses estrangulamentos serão identificados, e propõem-se soluções que podem ser automatizadas, permitindo que a pilha de software seja tão abstrata quanto os programadores assim o desejarem. Desta forma, pode-se sustentar a lei de Moore para o futuro próximo, enquanto outros avanços tecnológicos podem vir a ajudar em domínios diferentes.

Dedicar mais tempo ao desenvolvimento de algoritmos não pode ser a solução: os programadores de primeira linha não são apenas caros, são difíceis de encontrar. Além disto, atualmente os aplicativos podem chegar facilmente à região de milhões de linhas de código (ver Tabela 1). Este aumento nas linhas de código é explicado por vários fatores, mas dois são os mais óbvios: a complexidade natural da aplicação final e o uso de linguagens de programação mais abstratas por programadores para aumento de produtividade. Como apontado em [Cass 2022], as cinco linguagens mais demandadas pelo mercado são SQL, Java, Python, JavaScript e C#. A complexidade do tamanho, no domínio do software, sempre foi resolvida por uma maior abstração, o que explica a popularidade das linguagens acima mencionadas.

Infelizmente, é claro também que uma maior abstração não ajuda o desempenho, pelo contrário: a memória é quase completamente abstraída, quase nenhuma construção

de paralelismo está disponível, já que o objetivo de uma maior abstração é exatamente ocultar detalhes de hardware, que são necessários quando se pensa em desempenho e melhores algoritmos. Quando se acrescenta estagnação de hardware com uma maior complexidade de aplicação exigindo linguagens de abstração mais altas, o cenário para a execução do software em um futuro próximo é menor desempenho ou um tempo de desenvolvimento mais longo. Ambos os casos estão contra a tradição dos últimos 40 anos na computação, pelo menos, e têm um efeito severo no ambiente econômico moderno, cada vez mais dependente do desenvolvimento de software e hardware.

Os circuitos de processadores atuais utilizam a maior parte de sua área na memória incorporada, suportando vários níveis de armazenamento em cache [Nalamalpu et al. 2015, Kurd et al. 2010, Lotfi-Kamran et al. 2012, Santos et al. 2016]. É evidente que uma questão interessante é como se pode usar melhor esta área, para favorecer a computação de baixa energia nas aplicações com uso intensivo de dados.

O centro da estratégia é a identificação de que, para algoritmos com diferentes características, diferentes estruturas de hardware devem estar disponíveis. Este é o princípio do que é ter um conjunto de aceleradores dedicados, prontamente disponíveis e suportados por uma linguagem de programação de alto nível. A generalização deste princípio "acelerador quando necessário" não é claramente uma tarefa fácil. Os tipos de dados mais abstratos não são tão facilmente mapeados para uma estrutura de hardware. No passado recente, esta estratégia tem sido utilizada para grandes mercados. Por exemplo, as GPUs usam vários processadores e se destacam no suporte a aplicações SIMD [Abraham et al. 2015, Liu et al. 2013]. Os programadores podem alcançar alto desempenho usando uma linguagem de suporte de alto nível (neste caso, CUDA [NVIDIA 2023]). Outro bom exemplo pode ser encontrado analisando-se FPGAs nos domínios de telecomunicações e finanças, onde suas operações de bit permitem que operadores customizados e canais massivamente paralelos sejam processados de uma maneira eficiente em termos de energia [Lindtjorn et al. 2011, Park et al. 2015, Giefers et al. 2014].

Dado o tamanho e a complexidade das aplicações atuais e futuras, é improvável que um único modelo de computação, como SIMD para GPUs ou operações maciças MIMD para FPGAs possa abranger todos os aspectos de um programa complexo. Existem alguns trabalhos que tentam mesclar, por exemplo, CPUs e GPUs de forma perfeita, como em [Jablin et al. 2011, Kim et al. 2012]. No entanto, a dissipação de potência desta estratégia é significativa, comprometendo a escalabilidade [Deshpande and Draper 2016, Gamell et al. 2013]. Outra estratégia foi utilizada no processamento em memória (*Processing In Memory* - PIM), delegando algumas operações com pouca localidade à memória, como em [Siegl et al. 2016, Scrbak et al. 2017, Santos et al. 2021a, Santos et al. 2021b, de Lima et al. 2022]. Como não há localidade, o sistema de caches torna-se altamente ineficiente e um enorme dreno de energia.

Os exemplos acima indicam a direção correta, pois eles suportam aceleradores especializados em mercados de massa. O conjunto de aplicações atual diz respeito ao processamento massivo de dados com ou sem localidade espacial, com um mínimo gasto de energia. O principal conceito a ser explorado é que o processamento deve ser desenvolvido onde é menos dispendioso fazê-lo. Além disso, sempre que possível, cruzam-se as camadas de abstração, para que se possa superar o processador commoditizado de última

geração construído na mesma tecnologia.

4.3.2. Arquiteturas heterogêneas em infraestruturas compartilhadas

O uso de dispositivos heterogêneos traz grandes vantagens para infraestruturas compartilhadas, como as encontradas em edge e in-network computing. Como mencionado, aplicações distintas, com requisitos diversos, podem ser atendidas com melhor eficiência por um conjunto de dispositivos heterogêneos. Nesse contexto, FPGAs são uma solução promissora, já que aliam alta capacidade de processamento e eficiência com uma grande flexibilidade e a possibilidade de reprogramação após implantação, particularmente relevante em infraestruturas onde a totalidade das aplicações de interesse não é conhecida a priori. Entretanto, desafios importantes ainda devem ser endereçados para permitir o usufruto pleno dessas vantagens.

FPGAs são dispositivos reconfiguráveis que integram blocos lógicos configuráveis (*Configurable Logic Blocks* - CLBs), capazes de realizar qualquer função booleana com uma quantidade limitada de entradas, e uma malha de circuitos de interconexão, também reconfigurável [Boutros and Betz 2021]. Assim, através de um arquivo de reconfiguração, também chamado de *bitstream*, o dispositivo pode ser reconfigurado para implementar circuitos lógicos complexos. FPGAs típicos incluem, ainda, circuitos dedicados para funções aritméticas, memórias e blocos de entrada e saída para conexão com dispositivos externos.

A geração do *bitstream* é, em geral, feita por ferramentas de software fornecidas pelo próprio fabricante, a partir de descrições no nível de transferência de registradores (*Register Transfer Level* - RTL), codificadas em uma linguagem de descrição de hardware (*Hardware Description Language* - HDL) como Verilog ou VHDL. Como alternativa a esse processo, geralmente bastante oneroso, pode ser utilizada a síntese de alto nível (*High-Level Synthesis* - HLS) [Cong et al. 2022]. HLS consiste em gerar modelos RTL de um circuito, tipicamente a partir de código em alguma linguagem de programação de mais alto nível, como C/C++. Ferramentas de HLS vêm em constante evolução e aumento de popularidade, devido às melhorias na qualidade dos circuitos produzidos e aos importantes ganhos em produtividade [Liang et al. 2012].

Um aspecto relevante de HLS é a possibilidade de utilizar diretivas para guiar o processo de síntese quanto ao uso de técnicas de otimização como *loop unrolling*, *pipelining* e *array partitioning* [AMD 2023a]. Tais técnicas têm impacto direto nas métricas de desempenho e uso de recursos e de energia e, através de seu uso, desenvolvedores podem encontrar soluções mais adequadas para as restrições da aplicação em questão. Essa capacidade é particularmente relevante no contexto de edge e in-network computing, já que, conforme mencionado, frequentemente teremos restrições de desempenho ou de uso de recursos variadas para cada aplicação e a possibilidade de termos múltiplas versões de uma mesma funcionalidade pode ser vantajosa. Observa-se, porém, que a inserção manual de tais diretivas volta a onerar o processo de desenvolvimento, na contramão dos benefícios esperados de HLS.

Particularmente para o caso de FPGAs, o seu uso em infraestruturas compartilhadas impõe algumas dificuldades. A troca da configuração de um FPGA, realizada através da transferência de um novo bitstream, é um processo substancialmente mais longo que

uma troca de contexto em um GPP, o que dificulta o uso de compartilhamento temporal para aplicações que continuamente processam *streams* de dados, bastante comuns em edge e in-network computing [Gobatto et al. 2022]. Embora *overlays* possam facilitar a gerência do dispositivo em tais aplicações [Bachini Lopes et al. 2023], para contornar essa limitação, o compartilhamento espacial do dispositivo, ou seja, utilizar partes dos recursos reconfiguráveis concomitantemente alocadas a diferentes aplicações, torna-se uma funcionalidade importante [Vaishnav et al. 2018]. Para permitir a alocação independente de aplicações em regiões do FPGA, a reconfiguração parcial dinâmica (*Dynamic Partial Reconfiguration* - DPR) é uma funcionalidade suportada pelos FPGAs modernos dos principais fabricantes [AMD 2023b, Intel 2023].

DPR consiste em utilizar bitstreams parciais para reconfigurar parte dos recursos do FPGA, enquanto os demais componentes permanecem em operação ininterrupta. Essa funcionalidade permite compor múltiplas combinações de funções a partir de um conjunto enxuto de bitstreams parciais. Além disso, cada funcionalidade pode ser substituída independentemente, permitindo a instanciação de novas aplicações sem interromper aquelas já em operação. Para sua utilização, entretanto, DPR demanda um particionamento prévio dos recursos do dispositivo, criando partições dinamicamente reconfiguráveis fixas. O adequado dimensionamento dessas partições dá origem a uma variação do problema de *floorplanning*, tradicionalmente encontrado em projeto de dispositivos eletrônicos e já abordado para FPGAs em trabalhos como [Seyoum et al. 2019, Galea et al. 2018, Tang et al. 2020], porém para conjuntos fixos e previstos de módulos que compartilham o dispositivo.

Após o particionamento dos recursos no FPGA, existe ainda o desafio de definir qual tarefa será realizada em cada partição, ou seja, o posicionamento e escalonamento de tarefas dentro do dispositivo, como realizado em [Tang et al. 2020, Yao et al. 2022, Dhar et al. 2022], porém para um único dispositivo. Em uma escala mais abrangente, esse problema ainda deve considerar que temos diversos nodos na infraestrutura, potencialmente com dispositivos de diferentes capacidades, dispostos com variadas distâncias do usuário final, e com diferentes cargas de ocupação. Ou seja, o uso de FPGAs com DPR adiciona mais um aspecto a ser considerado pelo problema de posicionamento de tarefas em infraestruturas de edge e in-network computing. A modelagem dos recursos dos dispositivos é frequentemente simplificada em trabalhos da área, que consideram o FPGA como uma coleção de recursos sem uma estrutura fixa [Sharma et al. 2020] ou não consideram os diferentes recursos demandados por cada tarefa [Cooke and Fahmy 2020]. Essas simplificações podem levar a soluções teóricas ineficientes ou que não são implementáveis na prática devido às restrições geométricas do dispositivo que devem ser respeitadas [AMD 2023b]. Em [Sun et al. 2019] é apresentado um algoritmo para o posicionamento de tarefas em múltiplos FPGAs que modela os recursos heterogêneos e as restrições geométricas de particionamento, entretanto desconsiderando o possível caráter distribuído desses dispositivos, que pode introduzir restrições de latência e banda para as soluções.

Ainda, a complexidade de desenvolvimento de implementações que façam uso eficiente de FPGAs é um fator que pode desestimular seu uso. Embora HLS ofereça uma alternativa de maior produtividade, a aplicação de diretivas de síntese ainda é uma tarefa dispendiosa e essencialmente manual. Além da expertise necessária para identificar boas oportunidades de uso de cada técnica de otimização, longos ciclos de síntese são

necessários para avaliar cada solução candidata. O uso de heurísticas de exploração de espaço de projeto (*Design Space Exploration* - DSE) é uma alternativa investigada para automatizar esse processo e reduzir tempo de projeto [Schafer and Wang 2020]. Uma vez que a quantidade de recursos utilizados é afetada pelas decisões de tais ferramentas, elas também têm impacto nas dimensões mais adequadas das partições definidas durante o floorplanning, uma possibilidade ainda não investigada no contexto de infraestruturas de computação de borda e em rede.

Sumarizando, embora o uso de arquiteturas heterogêneas, em particular aquelas dotadas de FPGAs, seja promissor, diversos desafios ainda são encontrados. Do ponto de vista de desenvolvimento, ferramentas de HLS integradas a heurísticas de DSE são capazes de produzir múltiplas implementações de cada funcionalidade. Essas, entretanto, ainda não estão adequadamente integradas a ferramentas de gerência cientes das particularidades desses dispositivos. Em particular, o dimensionamento de partições com o objetivo de acomodar múltiplos designs produzidos automaticamente através de DSE em HLS ainda carece de solução. Ainda, o posicionamento de tarefas em infraestruturas heterogêneas deve ser realizado de forma ciente das partições dos FPGAs, garantindo que a disponibilidade de recursos de cada partição seja respeitada.

4.4. Possíveis soluções

Os desafios identificados para computação energeticamente eficiente estão espalhados por todas as camadas de abstração vistas na Figura 4.1. Assim, soluções adequadas para esse problema deverão estar igualmente em todas as camadas e, frequentemente, explorando múltiplas camadas de forma simultânea e colaborativa.

Quando olhamos para cada dispositivo isoladamente, temos que o enfraquecimento da lei de Moore e a dificuldade de explorarmos paralelismo de forma eficiente, são desafios prementes. Nesse nível, tanto abordagens arquiteturais, como processamento em memória (PIM) [Santos et al. 2021b] e arquiteturas heterogêneas, quanto abordagens em nível de circuito, como voltagens próximas à tensão de *threshold* dos dispositivos (*Near-Thresold Voltage* - NTV) [Tonetto et al. 2022], são promissoras.

Em um nível mais abrangente, considerando a integração de múltiplos dispositivos em sistemas compartilhados de computação de nuvem e borda, também identificamos desafios relevantes: *i*) como oferecer recursos computacionais flexíveis, porém de alto desempenho e eficiência energética; *ii*) como desenvolver aplicações para tais recursos, capazes de atender diferentes requisitos não funcionais, com custos de projeto reduzidos; e *iii*) como instanciar aplicações nessa infraestrutura distribuída e heterogênea de recursos, ocupando-os de forma eficiente e igualmente ciente dos requisitos de cada usuário.

A Figura 4.3 apresenta um conjunto de soluções promissor para tais desafios. Primeiramente, o uso de HLS permite a implementação de aceleradores dedicados por desenvolvedores de software. Ao utilizarmos dispositivos como FPGAs, mantemos uma infraestrutura flexível, capaz de atender diferentes aplicações à medida em que as demandas são recebidas ao longo do tempo. Ainda, através de heurísticas de exploração de espaço de projeto (DSE), é possível produzir diferentes designs, ocupando diferentes pontos da fronteira de Pareto que envolve uso de recursos e desempenho. Assim, é possível utilizar a implementação de menor custo que ainda atende cada requisição de usuário. Podemos,

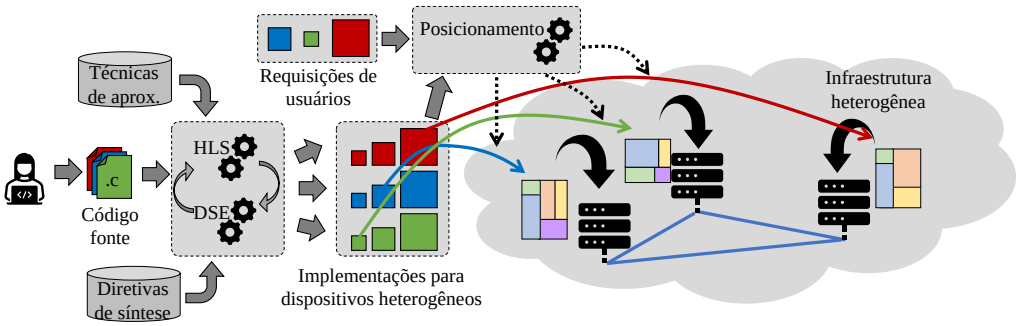

Figura 4.3. Fluxo de exploração de espaço de projeto e posicionamento de tarefas em uma infraestrutura heterogênea.

nesse ponto, considerar também a precisão demandada para cada aplicação: reduções substanciais de custo podem ser obtidas através de técnicas de computação aproximativa, que introduzem aproximações controladas na aplicação para reduzir uso de recursos ou de potência ou para aumentar o desempenho. A exploração do espaço conjunto que integra diretivas de síntese e técnicas de aproximação, embora complexa, pode encontrar soluções superiores em termos do *trade-off* entre custos e desempenho.

Após a elaboração desse conjunto diverso de implementações heterogêneas, resta ainda o problema de dispô-las adequadamente na infraestrutura. Uma vez que os enlaces de comunicação têm suas próprias latências e bandas limitadas, a garantia do atendimento dos requisitos do usuário não pode se dar somente a partir das características isoladas de cada acelerador. Deve-se, portanto, considerar o cenário completo do acelerador inserido em um ambiente de computação de borda e em rede. Através de mecanismos automatizados para esse posicionamento, cientes desses cenário completo, podemos facilitar a gerência dessas infraestruturas e identificar as configurações mais adequadas de acelerador para cada caso.

4.5. Conclusão

Neste capítulo, apresentamos os principais desafios vislumbrados para o desenvolvimento de sistemas computacionais energeticamente eficientes. Observamos que esses desafios englobam as mais diversas camadas desses sistemas: desde o desenvolvimento de software para novas aplicações, uma tarefa que demanda cada vez mais abstração para garantir produtividade, passando pela gerência de infraestruturas heterogêneas, e chegando até novos paradigmas arquiteturais. Este cenário desafiador portanto, dificilmente será resolvido por soluções limitadas a uma única camada de abstração dos sistemas, demandando soluções que cooperativamente reduzem consumo energético em todas as camadas.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

[Abraham et al. 2015] Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., and Lindahl, E. (2015). Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1-2:19–25.

[Algur and Sakri 2015] Algur, S. P. and Sakri, L. I. (2015). Parallelized genomic sequencing model: A big data approach for bioinformatics application. In *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pages 69–74.

[AMD 2023a] AMD (2023a). Vitis High-Level Synthesis User Guide.

[AMD 2023b] AMD (2023b). Vivado Design Suite User Guide - Dynamic Function eXchange.

[Bachini Lopes et al. 2023] Bachini Lopes, F., Schaeffer-Filho, A. E., and Nazar, G. L. (2023). Modular vnf components acceleration with fpga overlays. *IEEE Transactions on Network and Service Management*, 20(1):846–857.

[Blat et al. 2016] Blat, J., Evans, A., Kim, H., Imre, E., Polok, L., Ila, V., Nikolaidis, N., Zemčík, P., Tefas, A., Smrž, P., Hilton, A., and Pitas, I. (2016). Big data analysis for media production. *Proceedings of the IEEE*, 104(11):2085–2113.

[Boutros and Betz 2021] Boutros, A. and Betz, V. (2021). Fpga architecture: Principles and progression. *IEEE Circuits and Systems Magazine*, 21(2):4–29.

[Cass 2022] Cass, S. (2022). Top Programming Languages 2022.

[Cong et al. 2022] Cong, J., Lau, J., Liu, G., Neuendorffer, S., Pan, P., Vissers, K., and Zhang, Z. (2022). Fpga hls today: Successes, challenges, and opportunities. *ACM Trans. Reconfigurable Technol. Syst.*, 15(4).

[Cooke and Fahmy 2020] Cooke, R. A. and Fahmy, S. A. (2020). A model for distributed in-network and near-edge computing with heterogeneous hardware. *Future Generation Computer Systems*, 105:395–409.

[Coussy et al. 2009] Coussy, P., Gajski, D. D., Meredith, M., and Takach, A. (2009). An introduction to high-level synthesis. *IEEE Design & Test of Computers*, 26(4):8–17.

[de Lima et al. 2022] de Lima, J. a. P. C., Brandalero, M., Hübner, M., and Carro, L. (2022). Stap: An architecture and design tool for automata processing on memristor tcams. *J. Emerg. Technol. Comput. Syst.*, 18(2).

[de Lima and Carro 2022] de Lima, J. P. C. and Carro, L. (2022). Quantization-aware in-situ training for reliable and accurate edge ai. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1497–1502.

[Deshpande and Draper 2016] Deshpande, A. M. and Draper, J. T. (2016). A new metric to measure cache utilization for hpc workloads. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, page 10–17, New York, NY, USA. Association for Computing Machinery.

[Desjardins 2017] Desjardins, J. (2017). How Many Millions of Lines of Code Does It Take?

[Dhar et al. 2022] Dhar, A., Richter, E., Yu, M., Zuo, W., Wang, X., Kim, N. S., and Chen, D. (2022). Dml: Dynamic partial reconfiguration with scalable task scheduling for multi-applications on fpgas. *IEEE Transactions on Computers*, 71(10):2577–2591.

[dos Santos et al. 2022] dos Santos, F. F., Malde, S., Cazzaniga, C., Frost, C., Carro, L., and Rech, P. (2022). Experimental findings on the sources of detected unrecoverable errors in gpus. *IEEE Transactions on Nuclear Science*, 69(3):436–443.

[Galea et al. 2018] Galea, F., Carpov, S., and Zaourar, L. (2018). Multi-start simulated annealing for partially-reconfigurable fpga floorplanning. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1335–1338.

[Gamell et al. 2013] Gamell, M., Rodero, I., Parashar, M., and Poole, S. (2013). Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies. In *20th Annual International Conference on High Performance Computing*, pages 226–235.

[Giefers et al. 2014] Giefers, H., Plessl, C., and Förstner, J. (2014). Accelerating finite difference time domain simulations with reconfigurable dataflow computers. *SIGARCH Comput. Archit. News*, 41(5):65–70.

[Gobatto et al. 2022] Gobatto, L., Saquetti, M., Diniz, C., Zatt, B., Cordeiro, W., and Azambuja, J. R. (2022). Improving content-aware video streaming in congested networks with in-network computing. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1813–1817.

[Gonçalves et al. 2019] Gonçalves, L. R., Moura, R. F. D., and Carro, L. (2019). Aggressive energy reduction for video inference with software-only strategies. *ACM Trans. Embed. Comput. Syst.*, 18(5s).

[Intel 2023] Intel (2023). Intel Quartus Prime Software Features Partial Reconfiguration.

[Jablin et al. 2011] Jablin, T. B., Prabhu, P., Jablin, J. A., Johnson, N. P., Beard, S. R., and August, D. I. (2011). Automatic cpu-gpu communication management and optimization. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, page 142–151, New York, NY, USA. Association for Computing Machinery.

[Jones 2018] Jones, N. (2018). How to stop data centres from gobbling up the world's electricity. *Nature*, 561(7722):163–167.

[Jun et al. 2017] Jun, S.-W., Xu, S., and Arvind (2017). Terabyte sort on fpga-accelerated flash storage. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 17–24.

[Kianpisheh and Taleb 2023] Kianpisheh, S. and Taleb, T. (2023). A survey on in-network computing: Programmable data plane and technology specific applications. *IEEE Communications Surveys & Tutorials*, 25(1):701–761.

[Kim et al. 2012] Kim, J., Seo, S., Lee, J., Nah, J., Jo, G., and Lee, J. (2012). Snucl: An opencl framework for heterogeneous cpu/gpu clusters. In *Proceedings of the 26th ACM International Conference on Supercomputing*, ICS '12, page 341–352, New York, NY, USA. Association for Computing Machinery.

[Kurd et al. 2010] Kurd, N. A., Bhamidipati, S., Mozak, C., Miller, J. L., Wilson, T. M., Nemani, M., and Chowdhury, M. (2010). Westmere: A family of 32nm ia processors. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 96–97.

[Leipnitz and Nazar 2019a] Leipnitz, M. T. and Nazar, G. L. (2019a). High-level synthesis of approximate designs under real-time constraints. *ACM Trans. Embed. Comput. Syst.*, 18(5s).

[Leipnitz and Nazar 2019b] Leipnitz, M. T. and Nazar, G. L. (2019b). High-level synthesis of resource-oriented approximate designs for fpgas. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6.

[Leipnitz and Nazar 2020] Leipnitz, M. T. and Nazar, G. L. (2020). Throughput-oriented spatio-temporal optimization in approximate high-level synthesis. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pages 316–323.

[Liang et al. 2012] Liang, Y., Rupnow, K., Li, Y., Min, D., Do, M. N., and Chen, D. (2012). High-level synthesis: Productivity, performance, and software constraints. *JECE*, 2012.

[Lindtjorn et al. 2011] Lindtjorn, O., Clapp, R., Pell, O., Fu, H., Flynn, M., and Mencer, O. (2011). Beyond traditional microprocessors for geoscience high-performance computing applications. *IEEE Micro*, 31(2):41–49.

[Liu et al. 2013] Liu, Y., Wirawan, A., and Schmidt, B. (2013). Cudasw++ 3.0: accelerating smith-waterman protein database search by coupling cpu and gpu simd instructions. *BMC bioinformatics*, 14:1–10.

[Long et al. 2018] Long, C., Cao, Y., Jiang, T., and Zhang, Q. (2018). Edge computing framework for cooperative video processing in multimedia iot systems. *IEEE Transactions on Multimedia*, 20(5):1126–1139.

[Lotfi-Kamran et al. 2012] Lotfi-Kamran, P., Grot, B., Ferdman, M., Volos, S., Kocberber, O., Picorel, J., Adileh, A., Jevdjic, D., Idgunji, S., Ozer, E., and Falsafi, B. (2012). Scale-out processors. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, page 500–511, USA. IEEE Computer Society.

[Nalamalpu et al. 2015] Nalamalpu, A., Kurd, N., Deval, A., Mozak, C., Douglas, J., Khanna, A., Paillet, F., Schrom, G., and Phelps, B. (2015). Broadwell: A family of ia 14nm processors. In *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pages C314–C315.

[Nane et al. 2016] Nane, R., Sima, V.-M., Pilato, C., Choi, J., Fort, B., Canis, A., Chen, Y. T., Hsiao, H., Brown, S., Ferrandi, F., Anderson, J., and Bertels, K. (2016). A survey and evaluation of fpga high-level synthesis tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(10):1591–1604.

[Nazar et al. 2021a] Nazar, G. L., Kopper, P. H., Leipnitz, M. T., and Juurlink, B. (2021a). Lightweight dual modular redundancy through approximate computing. In *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–8.

[Nazar et al. 2021b] Nazar, G. L., Kopper, P. H., Leipnitz, M. T., and Juurlink, B. (2021b). Precep: Automatic insertion of partial redundancy based on critical error probability. *Microelectronics Reliability*, 126:114226. Proceedings of ESREF 2021, 32nd European Symposium on Reliability of Electron Devices, Failure Physics and Analysis.

[NVIDIA 2023] NVIDIA (2023). CUDA C++ Programming Guide.

[Park et al. 2015] Park, S.-W., Park, J., Bong, K., Shin, D., Lee, J., Choi, S., and Yoo, H.-J. (2015). An energy-efficient and scalable deep learning/inference processor with tetra-parallel mimd architecture for big data applications. *IEEE Transactions on Biomedical Circuits and Systems*, 9(6):838–848.

[Santos et al. 2016] Santos, P. C., Alves, M. A. Z., Diener, M., Carro, L., and Navaux, P. O. A. (2016). Exploring cache size and core count tradeoffs in systems with reduced memory access latency. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 388–392.

[Santos et al. 2021a] Santos, P. C., de Lima, J. P. C., de Moura, R. F., Alves, M. A. Z., Beck, A. C. S., and Carro, L. (2021a). Enabling near-data accelerators adoption by through investigation of datapath solutions. *International Journal of Parallel Programming*, 49:237–252.

[Santos et al. 2021b] Santos, P. C., Forlin, B. E., and Carro, L. (2021b). Providing plug n' play for processing-in-memory accelerators. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 651–656.

[Satyanarayanan 2017] Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

[Satyanarayanan et al. 2009] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

[Schafer and Wang 2020] Schafer, B. C. and Wang, Z. (2020). High-level synthesis design space exploration: Past, present, and future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2628–2639.

[Scrbak et al. 2017] Scrbak, M., Islam, M., Kavi, K. M., Ignatowski, M., and Jayasena, N. (2017). Exploring the processing-in-memory design space. *Journal of Systems Architecture*, 75:59–67.

[Seyoum et al. 2019] Seyoum, B. B., Biondi, A., and Buttazzo, G. C. (2019). Flora: Floorplan optimizer for reconfigurable areas in fpgas. *ACM Trans. Embed. Comput. Syst.*, 18(5s).

[Sharma et al. 2020] Sharma, G. P., Tavernier, W., Colle, D., and Pickavet, M. (2020). Vnf-aapc: Accelerator-aware vnf placement and chaining. *Computer Networks*, 177:107329.

[Shi et al. 2016] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

[Siegl et al. 2016] Siegl, P., Buchty, R., and Berekovic, M. (2016). Data-centric computing frontiers: A survey on processing-in-memory. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, page 295–308, New York, NY, USA. Association for Computing Machinery.

[Sun et al. 2019] Sun, Z., Zhang, H., and Zhang, Z. (2019). Resource-aware task scheduling and placement in multi-fpga system. *IEEE Access*, 7:163851–163863.

[Tang et al. 2020] Tang, Q., Wang, Z., Guo, B., Zhu, L.-H., and Wei, J.-B. (2020). Partitioning and scheduling with module merging on dynamic partial reconfigurable fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 13(3).

[Tonetto et al. 2022] Tonetto, R. B., Beck, A. C. S., and Nazar, G. L. (2022). Snap: Selective ntv heterogeneous architectures for power-efficient edge computing. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 357–364.

[Vaishnav et al. 2018] Vaishnav, A., Pham, K. D., and Koch, D. (2018). A survey on fpga virtualization. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 131–1317.

[Yao et al. 2022] Yao, R., Zhao, Y., Yu, Y., Zhao, Y., and Zhong, X. (2022). Fast search and efficient placement algorithm for reconfigurable tasks on modern heterogeneous fpgas. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(4):474–487.