Carlos Michel Betemps

Exploração Precoce do Espaço de Projeto: Aplicando Heurísticas e Simulação em Nível de Sistema na Avaliação de Cargas de Trabalho sobre Arquiteturas Heterogêneas

Proposta de Tese apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação

Orientador: Prof. Dr. Bruno Zatt
Coorientadores: Prof. Dr. Daniel Munari Vilchez Palomino
Prof. Dr. Marcelo Schiavon Porto

Pelotas, 2018

RESUMO

Palavras-Chave: exploração do espaço de projeto; heurísticas; modelos em alto nível de abstração; paralelismo em nível de tarefa; heterogeneidade; sistemas embarcados

1 INTRODUÇÃO E MOTIVAÇÃO

Sistemas Embarcados (SE) implementam funções dedicadas e específicas dentro de um sistema hospedeiro (CAMPOSANO; WILBERG, 1996). São componentes de sistemas maiores usados para controlar e/ou diretamente monitorar tais sistemas (GUESSI et al., 2012). Podem ser considerados como sistemas de processamento de informações que estão embarcadas em um produto maior e que normalmente não são diretamente visíveis para o usuário (MARWEDEL, 2006).

SEs são altamente diversos, variando desde pequenos sistemas como refrigeradores, máquinas de auto-atendimento ou sensores inteligentes em automação de construções até sistemas distribuídos e complexos como sistemas de controle automotivo e aeroespacial (VOELTER et al., 2013). Software embarcado também está em constante crescimento quanto as funcionalidades implementadas, tamanho e complexidade (HENDRIKS et al., 2016). Assim, o projeto de SEs está em um contexto altamente competitivo e que evolui rapidamente a medida que novas tecnologias são introduzidas (BERTELS, 2012). A transformação de um projeto (*design*) eficiente em um produto de sucesso depende do mesmo se tornar o primeiro no mercado (*time-to-market*), proporcionando novas funcionalidades (primeiro produto a prover tais funcionalidades), obedecendo às restrições de desempenho e a um preço acessível (HERRERA et al., 2014; MISCHKALLA; HE; MUELLER, 2010).

O projeto de SEs (sistemas de hardware e software) normalmente envolve modelagem, validação e implementação (De Micheli; GUPTA, 1997). A **modelagem** trata da conceitualização e refinamento das especificações, produzindo modelos de hardware e software. A **validação** busca o alcance de um nível de confiança razoável de que o sistema funcionará como projetado. A **implementação** é a realização física do hardware e do software.

Devido à complexidade do projeto de novos produtos, a competitividade no mercado de SEs, e a lacuna existente no projeto conjunto de hardware e software, cada vez mais tarefas de exploração de alternativas de soluções são realizadas em nível de sistema (GRIES, 2004; GAJSKI et al., 2009), de forma a possibilitar a tomada de decisões em fases iniciais de desenvolvimento. Neste contexto, são utilizados modelos

em alto nível de abstração para descrever as diferentes configurações de arquitetura e as cargas de trabalho a serem submetidas ao sistema. Como as avaliações precisam ocorrer em fases iniciais de desenvolvimento, quando normalmente ainda não há disponibilidade de arquitetura e de aplicações que possam ser executadas, a utilização de simulação em nível de sistema surge como alternativa para avaliação de soluções durante a Exploração do Espaço de Projeto (DSE - *Design Space Exploration*).

Exploração do Espaço de Projeto (DSE - *Design Space Exploration*) é o processo de analisar o conjunto de possíveis soluções e definir qual será selecionada (MARWEDEL, 2006). O desempenho de um sistema depende de um conjunto diverso de fatores, tais como arquitetura da aplicação de software, arquitetura da plataforma de hardware, de como as funcionalidades da aplicação são executadas pelos elementos de processamento (como CPUs, GPUs, FPGAs, ASICs, DSPs, etc.), além de vários parâmetros tais como tamanhos de caches, tamanho de memória, dentre outros. Isto faz da DSE uma atividade chave para permitir avaliações do sistema logo no início das iniciativas de desenvolvimento (HERRERA et al., 2014). De forma geral, uma abordagem de DSE consiste de quatro componentes básicos (ASCIA et al., 2011): (*i*) um ponto de entrada representado pela(s) configuração(ões) inicial(ais), (*ii*) um modelo de avaliação das configurações, (*iii*) estratégia de exploração que consiste de um conjunto de transformações a serem aplicadas nas configurações e (*iv*) um critério de parada.

Normalmente, o processo de DSE segue a abordagem *Y-chart* (GRIES, 2004). Nesta abordagem, um modelo de aplicação (ou aplicações – a carga de trabalho a ser submetida ao sistema) – derivado a partir do domínio da aplicação alvo – descreve o seu comportamento funcional de uma maneira independente de arquitetura. Simultaneamente, um modelo de arquitetura – definido considerando a aplicação – define os recursos da arquitetura (elementos de processamento - PE (*processing elements*)) e captura suas restrições de desempenho (JIA et al., 2013). Ademais, ambos modelos são considerados, de forma separada, e um terceiro modelo – de **mapeamento** – define uma ligação entre ambos (cada tarefa da aplicação é alocada para execução em um determinado PE da arquitetura) (GRIES, 2004). O modelo de mapeamento mapeia o modelo de aplicação no modelo de arquitetura para execução (por meio de co-simulação, por exemplo), depois do qual distintas métricas de sistema podem ser quantitativamente avaliadas (JIA et al., 2013). A abordagem *Y-chart* consiste (GRIES, 2004), basicamente, do mapeamento entre a carga de trabalho (*workload*) e os elementos de processamento da arquitetura. Após, considerando métricas como, por exemplo, desempenho e dissipação de potência, é realizada a **análise** das configurações em investigação por meio de testes que podem incluir execução em hardware real, simulação ou geração de estimativas. As **métricas** de desempenho são utilizadas como base para proceder com ajustes na arquitetura, nas aplicações ou mesmos

nos mapeamentos. A Fig. 1 ilustra os passos da referida abordagem.

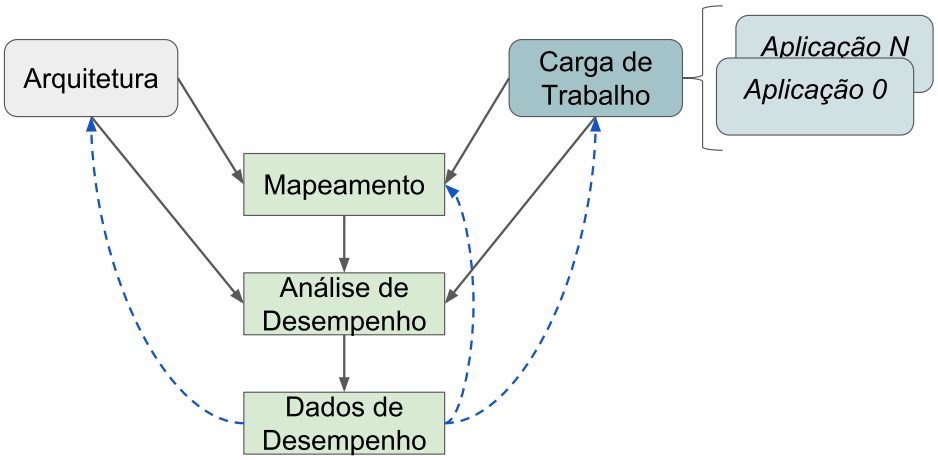

Figura 1 – Abordagem *Y-chart* para DSE - adaptado de (GRIES, 2004)

Em resumo, o processo de DSE consiste, a grosso modo, de dois componentes interdependentes (GRIES, 2004): (*i*) a avaliação de cada ponto de projeto do espaço de soluções usando, por exemplo, modelos analíticos ou simulação e (*ii*) o mecanismo de busca que sistematicamente percorre o espaço de projeto. Considerando o componente (*i*), o uso de simulação em nível de sistema apresenta características interessantes no que tange aspectos como tempo de simulação e uso de modelos de alto nível (em fases iniciais de desenvolvimento). Quanto ao componente (*ii*), o emprego de heurísticas para a seleção e o direcionamento das buscas por soluções candidatas pode reduzir o tamanho do espaço de projeto (usualmente vasto no caso de projeto de SEs).

1.1 Sistemas Heterogêneos de Computação e Aplicações

Os ambientes de computação atuais estão explorando as capacidades de uma ampla gama de diferentes processadores, tais como CPUs (*central processing units*), processadores de sinais digitais (DSPs - *digital signal processors*), processadores de mídia, processadores vetoriais, hardware reconfigurável (como FPGAs - *Field Programmable Gate Array*) e unidades de processamento gráfico (GPUs - *graphic processing units*) (GASTER et al., 2012; WAIDYASOORIYA; HARIYAMA; UCHIYAMA, 2018), integrando um relevante número de recursos de processamento heterogêneos e independentes dentro do mesmo sistema (MIELE et al., 2015).

Dado que normalmente as aplicações submetidas aos sistemas apresentam uma série de diferentes comportamentos - desde aplicações intensivas em controle (*control intensive*) - como aplicações de busca, classificação e análise - até aquelas intensas em dados (*data intensive*) - como processamento de imagens, simulação e modelagem, e mineração de dados), não há uma arquitetura única que seja adequada para

cada tipo de aplicação (GASTER et al., 2012). Além disso, várias partes (trechos) das aplicações podem ser executadas em paralelo (os chamados *kernels* das aplicações), oportunizando a exploração do paralelismo em nível de tarefa (*task-level parallelism* - TLP) (SCARPAZZA et al., 2006). O TLP pode ser explorado diretamente via sistemas multiprocessados, incluindo aqueles com uma série de PEs heterogêneos.

Computação com recursos heterogêneos, como FPGAs e GPUs, podem melhorar o desempenho e eficiência energética dos sistemas em várias ordens de magnitude. Assim, sistemas heterogêneos surgem com perspectivas de avanço habilitadas, principalmente, pelo abundante paralelismo de dados – que proporciona a alta capacidade de computação e eficiência energética destes tipos de PEs (ROGERS, 2013; DURELLI et al., 2014; WAIDYASOORIYA; HARIYAMA; UCHIYAMA, 2018).

1.2 Simulação em Nível de Sistema e Modelos de Aplicações e Arquitetura

Simulação é uma forma de avaliar as diferentes soluções para um projeto. Modelagem e simulação usando alto nível de abstração tem um papel importante em fases iniciais de desenvolvimento. Ambas permitem capturar o comportamento do sistema e suas interações, normalmente demandando menores esforços de modelagem (menos detalhes a serem modelados) e de simulação (tempo de simulação) (ERBAS et al., 2007). A utilização de níveis altos de abstração no projeto de SEs é defendida por diversos autores para tratar da complexidade dos projetos (LEITE; WEHRMEISTER, 2016), favorecer abordagens mais rápidas e custo-efetivas que permitam análises relevantes dos complexos espaços de projeto (AN; GAMATIÉ; RUTTEN, 2015), e que habilitem estimativas no início do projeto baseadas em informações parciais e incertas – que podem ser utilizadas para guiar o processo de projeto na direção correta (HENDRIKS et al., 2016).

Simuladores com precisão de ciclo ou de instrução não podem atender a demanda de processamento necessária em atividades de DSE, dado o alto volume de detalhes sobre o sistema que precisam ser simulados e que podem necessitar de tempos de simulação proibitivos (HERRERA et al., 2014; GRÜTTNER et al., 2013). Para contornar os problemas relacionados à simulação com precisão de ciclo tem-se os simuladores em nível de sistema. Segundo Gries (GRIES, 2004), durante uma simulação em nível de sistema, a avaliação ocorre em um alto nível de abstração, sendo o sistema representado por uma interconexão de blocos arquiteturais que representam os processadores, as memórias e os barramentos. Assim, modelos de plataforma ou de recursos de processamento (elementos de processamento) representam a arquitetura do sistema (INDRUSIAK; DZIURZANSKI; SINGH, 2016). No caso da representação da carga de trabalho, as aplicações são descritas por modelos de granulação mais

grossa, tais como processos de interação ou mesmo procedimentos inteiros (GRIES, 2004). Por exemplo, grafos de tarefas são uma forma comum de representar as cargas de trabalhos (aplicações) no contexto de sistemas embarcados (INDRUSIAK; DZIURZANSKI; SINGH, 2016), sendo esta a forma de representação aplicada em (MIELE et al., 2015). Outras formas de representação incluem redes de processo Kahn (KPN - *Kahn Process Networks*) (ERBAS et al., 2007), modelos UML/MARTE (GRÜTTNER et al., 2013; HERRERA et al., 2014; LEITE; WEHRMEISTER, 2016), modelos de *data-flow* (como *Homogeneous Synchronous DataFlow* - HSDF) (NOGUEIRA et al., 2016), redes de Petri (como *Coloured Petri Net* - CPN) (CALLOU et al., 2011), autômatos (como autômatos temporizados - *Timed Automata*) (JIANG et al., 2013; WANG et al., 2011), dentre outros. Estes modelos representam as cargas de trabalho (aplicações) que devem ser submetidas aos sistemas em análise/construção.

O nível de abstração dos projetos têm sido aumentado para o nível de sistema também com intuito de aumentar a produtividade, gerando um grande interesse em TLM (*Transaction-Level Modeling*) (CAI; GAJSKI, 2003). Com uso de TLM os detalhes sobre comunicação são separados dos componentes de computação por meio do conceito de canais (*channels*). TLM aumenta a velocidade de simulação, permitindo explorar e validar alternativas de projeto em um nível de abstração mais alto (CAI; GAJSKI, 2003). TLM é normalmente utilizada em conjunto com SystemC (PATEL; SHUKLA, 2008). SystemC é uma linguagem de projeto (*design*) de nível de sistema com forte apoio industrial tendo sido inserida em muitos fluxos de projeto do setor. O uso da infraestrutura do C++ e da sua natureza orientada a objetos estende a usabilidade da SystemC, sendo um meio adequado para a co-simulação de hardware e software (PATEL; SHUKLA, 2008).

1.3 Abordagens de Exploração do Espaço de Projeto

Dada uma determinada carga de trabalho (conjunto de aplicações), os parâmetros de um sistema (referentes a sua arquitetura como, por exemplo, número e tipos de PEs empregados) podem ser devidamente ajustados para encontrar o melhor *trade-off* entre as métricas de sistema avaliadas (por exemplo: energia, área e *delay*). Exploração do Espaço de Projeto (DSE - *Design Space Exploration*) é a ferramenta utilizada para encontrar a *configuração* ótima (PANERATI; SCIUTO; BELTRAME, 2017). A tarefa de ajuste dos parâmetros de arquitetura basicamente caracteriza-se como um problema de otimização que envolve a maximização (ou minimização) de múltiplos objetivos — potencialmente com várias soluções ótimas. A qualidade de uma configuração é expressa pelo conjunto de métricas (ou funções objetivo) utilizadas. O conjunto de soluções ótimas em problemas de otimização multi-objetivos ficam posicionados na chamada *curva de Pareto* — são os pontos ótimos que não são dominados por ne-

nhum outro ponto (PANERATI; SCIUTO; BELTRAME, 2017).

A obtenção da *curva de Pareto* exige a avaliação de todas as possíveis configurações. No entanto, o vasto tamanho do espaço de projeto é um problema na maior parte dos projetos de sistemas embarcados, e que pode tornar-se ainda mais severo de acordo com os requisitos não-funcionais (restrições) de projeto (GLAß et al., 2017). Assim, o principal objetivo da DSE pode ser relaxado no sentido de buscar minimizar o tempo de exploração enquanto garante soluções de boa qualidade (ASCIA et al., 2011) (não necessariamente soluções ótimas). Normalmente as abordagens de DSE utilizam estratégias para podar o espaço de projeto e minimizar o número de configurações a serem avaliadas (ASCIA et al., 2011). Heurísticas (ou Meta-Heurísticas) normalmente são aplicadas neste contexto. Meta-heurísticas são metodologias gerais de mais alto nível que podem ser utilizados como estratégias guias no desenvolvimento de heurísticas para solucionar problemas de otimização específicos (TALBI, 2009, p. 1.).

Esta seção apresenta algumas abordagens para DSE, incluindo trabalhos elencados em estudo prévio realizado pelo autor desta proposta (BETEMPS, 2017) (disponível para visualização por meio de link existente no respectivo item na seção de referências), especificamente para o Exame de Qualificação no PPGC da UFPel, no qual foram levantados artigos que tratam sobre o desenvolvimento de Sistemas Embarcados utilizando Alto Nível de Abstração.

Algumas abordagens de DSE utilizam métricas como os artefatos principais no direcionamento da avaliação das soluções candidatas para percorrer o espaço de projeto. Trabalhos como (GHEORGHITA et al., 2009; RAYMOND et al., 2015; RODRÍGUEZ, 2017; FALKNER et al., 2016; ALTENBERND et al., 2016; HENDRIKS et al., 2016; HUANG; XIU, 2015) utilizam estimativas relacionadas ao desempenho para o referido propósito. Por outro lado, métricas referentes à Energia consumida ou Potência dissipada são as escolhidas em outras abordagens (OUNI et al., 2017; JIANG; ELES; PENG, 2016; SORBER et al., 2007; KIM; HONG, 2015). Métricas relacionadas ao desempenho e energia recorrentemente são utilizadas em abordagens de DSE.

Em (FALKNER et al., 2016) é proposta uma abordagem integrada, baseada em modelos abstratos, para a predição de desempenho de sistemas de defesa e sistemas de sistemas (SoS - *Systemsof-Systems*) dirigida por modelo (uma abordagem MDE – *Model Driven Engineering*). O sistema de prototipação arquitetural utiliza o conceito de cenários e uma série de DSMLs (*domain specific modeling languages*) para a descrição das avaliações a serem realizadas na DSE. Estimativas de tempo em nível de código-fonte são tratadas em (ALTENBERND et al., 2016) por meio da geração de um Modelo de Tempo a partir do código-fonte (não do executável). Recozimento simulado (*Simulated Annealing*) é utilizado para a obtenção do Modelo Linear de Tempo e, por consequência, o WCET (Tempo Estimado no Pior Caso - *Worst-case*

Estimation Time).

O uso de modelos UML (GOMAA, 2011) e de seus perfis, como MARTE, como elementos centrais de modelagem são aplicados em algumas abordagens de DSE (HERRERA et al., 2014; GRÜTTNER et al., 2013; PEDRE et al., 2016). Uma abordagem de **Projeto Baseado em Plataforma** para a condução do DSE é utilizada em (GRÜTTNER et al., 2013) como parte de um *framework* de projeto denominado COMPLEX. Modelos UML/MARTE são utilizados para a geração automática de modelos executáveis *SystemC* para fins de validação. A definição do espaço de projeto utiliza o *profile* MARTE onde mapeamentos arquiteturais (espaço de alocação), atributos da plataforma (atributos do espaço) e arquiteturas da plataforma (espaço arquitetural) são definidos. Também associado ao projeto COMPLEX, Herrera et al. (HERRERA et al., 2014) apresentam uma metodologia para DSE que integra tecnologias de Engenharia Dirigida por Modelos (MDE - *Model Driven Engineering*) e Nível do Sistema Eletrônico (ESL - *Electronic System Level*) juntamente com a exploração de projeto. O espaço de projeto é capturado com o uso de modelos UML/MARTE que permitem a geração de modelos executáveis. A ferramenta de simulação SCoPE+ é utilizada para a avaliação do desempenho das configurações. SCoPE+ inclui uma biblioteca de estimativas de desempenho para diferentes mapeamentos arquiteturais. Uma metodologia de co-projeto (Co-design) para SEs centrados em processador com aceleradores de hardware usando FPGA é apresentada em (PEDRE et al., 2016). Orientação a Objetos e UML são utilizados para modelagem das aplicações, C++ para implementação e ferramentas de tradução permitem a tradução semi-automática de código C para HDL.

Alguns trabalhos utilizam as capacidades de reconfiguração de dispositivos para lidar com o projeto de SEs. Diguet e seus colegas (DIGUET; EUSTACHE; GOGNIAT, 2011) apresentam um gerenciador de configuração com capacidades de controle e de decisão em tempo de execução para buscar a otimização da arquitetura – adaptação em tempo de execução. FoRTReSS (*Flow for Reconfigurable archiTectures in Real time SystemS*) (DUHEM et al., 2015) usa o conceito de Computação Reconfigurável (por intermédio da aplicação de FPGA) para a atividade de alocação de tarefas em elementos de processamento e, possivelmente, utilizando a parte ou região reconfigurável do sistema com intuito de otimização. Aplicações adaptativas em MPSoCs são tratadas em (AN; GAMATIÉ; RUTTEN, 2015). A DSE é implementada em uma ferramenta protótipo denominada CLASSY (*CLock AnalysiS SYstem*) por meio de busca exaustiva ou por exploração baseada em heurística (algoritmo evolucionário NSGA-II). Em (HUANG; HSIUNG; SHEN, 2010) é apresentada a plataforma de co-projeto de hardware/software baseada em UML (UCoP - *UML-based hardware/software Codesign Platform*). O foco deste trabalho é em sistemas de segurança em rede parcialmente reconfiguráveis dinamicamente (DPRNSS - *Dynamically partially reconfigurable network security systems*) e utiliza a capacidade de reconfiguração de dispositivos

FPGA. SEs de segurança crítica são abordados por (ADLER et al., 2011), onde solucionadores de restrições de programação que usam Teoria de Módulo de Satisfação (SMT - *Satisfiability Modulo Theory*) são as ferramentas utilizadas para encontrar configurações preferíveis no espaço de projeto.

Algoritmos genéticos ou, de forma mais geral, **algoritmos evolucionários**, recorrentemente podem ser observados nas abordagens de DSE apresentadas na literatura. Uma abordagem DSE para Sistemas Embarcados Multimídia é descrita em (NOGUEIRA et al., 2016), no qual as aplicações são representadas como grafos HSDF (*homogeneous synchronous dataflow* - HSDF) e a plataforma de hardware é representada como um grafo dirigido ATG (*Architecture Template Graph*). Um algoritmo genético capaz de lidar com múltiplos objetivos (desempenho, preço e consumo de potência) é utilizado visando encontrar soluções ótimas de Pareto. O formalismo DEVS (técnica formal para modelagem, análise e simulação de sistemas de evento-discreto - *Discrete EVent system Specification*) é utilizado para avaliar as soluções candidatas por meio de simulação. Intervalos de confiança são utilizados como critério de parada das simulações.

Uma abordagem de automação de projeto eletrônico (EDA - *Eletronic Design Automation*) e que aplica algoritmos evolucionários multi-objetivo (MOEA - *Multi-Objective Evolutionary Algorithms*) e técnicas *fuzzy* é apresentada em Ascia et al. (ASCIA et al., 2011). A abordagem de DSE utiliza um aproximador de soluções (sistema *fuzzy*), verificando se a solução obtida é confiável. Quando a solução não é confiável, principalmente durante a fase de treinamento do algoritmo, é utilizada simulação para a geração de métricas. A configuração do sistema é mapeada para uma estrutura de cromossomo e utilizada pelo algoritmo evolucionário em suas interações.

Em (TRINDADE; CORDEIRO, 2016), foram aplicadas diferentes técnicas como programação linear inteira (ILP - *Integer Linear Programming*), algoritmos genéticos (GA) e ferramenta de verificação baseada em SMT (*satisfiability modulo theories*) para resolver um problema de particionamento entre hardware (HW) e software (SW).

Stralen e Pimentel (STRALEN; PIMENTEL, 2010) utilizam cenários de carga de trabalho (*workloads scenarios*) para capturar o comportamento dinâmico das aplicações (nos níveis intra e interaplicações) e apresenta uma abordagem de DSE baseada em cenário para arquiteturas MPSoCs. A referida abordagem utiliza o conceito de coevolução para avaliar dois problemas simultaneamente: o espaço de projeto do MPSoC e o espaço de cenários para encontrar um subconjunto representativo dos mesmos. Assim, a abordagem usa um algoritmo genético co-evolucionário (*coevolutionary genetic algorithm*).

Erbas et al. (ERBAS et al., 2007) apresentam o DSE de aplicações multimídia em que as aplicações são representadas por modelos de computação KPN (*Kahn Process Network*) gerados a partir de especificações em C/C++. O ambiente de modelagem

e simulação *Sesame* permite a análise de desempenho seguindo os princípios da abordagem **Y-chart** – os modelos de arquitetura e aplicações são desacopladas. Para lidar com a otimização multi-objetivos é aplicado o algoritmo evolucionário de Pareto para selecionar soluções ótimas para o mapeamento.

De forma geral, os trabalhos se baseiam, direta ou indiretamente, em especificações definidas em nível de código, fazendo com que o nível de abstração utilizado seja menor. Ademais, algumas abordagens exigem conhecimento especializado em linguagens formais específicas, o que ocasiona o aumento de detalhes que necessitam ser modelados e/ou especificados durante as atividades de modelagem.

1.4 Heurísticas aplicadas na Exploração do Espaço de Projeto

Heurísticas, ou Meta-heurísticas, são métodos de solução que controlam a interação entre procedimentos de melhora local e estratégias de alto nível para criar processos capazes de escapar de ótimos locais e que realizem uma busca robusta de um espaço de solução (GENDREAU; POTVIN, 2010a). Por outro lado, (TALBI, 2009, p. 1.) define meta-heurísticas como metodologias gerais de mais alto nível (*templates*) que podem ser utilizados como estratégias guias no desenvolvimento de heurísticas relacionadas para solucionar problemas de otimização específicos .

1.5 Tema e Contribuições da Tese

Nesta Seção são apresentados o **tema da tese** – estruturada em três partes: `Contexto`, `Questão de Pesquisa` e `Hipótese` – e as **contribuições esperadas** com o desenvolvimento do trabalho em tela.

1.5.1 Questão de Pesquisa e Hipótese

`Contexto`

- No contexto de desenvolvimento de sistemas embarcados, sabe-se que o *tamanho do espaço de projeto* pode ser vasto. No caso de um fluxo de projeto de hardware detalhado, a *exploração do espaço de projeto* demanda *proibitivos tempos* para ser realizada de forma completa. *Exploração Precoce do Espaço de Projeto* (*Early DSE*) prevê que, em *fases iniciais* de desenvolvimento, modelos de *alto nível de abstração* para ambos a plataforma e carga de trabalho pretendidas sejam avaliadas, permitindo que o *vasto espaço de projeto* possa ser explorado de forma rápida. Assim, o espaço inicial pode ser reduzido significativamente para ser avaliado de forma detalhada em fluxos tradicionais de projeto de hardware. Considerando o *contexto* descrito, são apresentadas a *Questão de Pesquisa* e *Hipótese* do trabalho.

`Questão de Pesquisa`

- Considerando o `contexto` apresentado, como mitigar significativamente o tamanho do espaço de projeto a ser fornecido como entrada em fluxos detalhados de projeto de hardware usando, como base, modelos de alto nível de abstração para a descrição da arquitetura e da carga de trabalho (aplicações) específicas de um sistema, assim como restrições gerais de funcionamento, como energia consumida e desempenho?

`Hipótese`

- Entende-se que *heurísticas* para seleção de soluções candidatas mais promissoras, sem necessidade da exploração exaustiva de todas as possíveis soluções, possam ser aplicadas em fases iniciais de desenvolvimento com intuito de prover um *espaço de projeto reduzido*. Ademais, *simulação de nível de sistema* pode ser utilizada para gerar dados de avaliação de forma rápida sobre as configurações experimentadas, possibilitando uma análise *comparativa* (ou qualitativa) entre as configurações, mesmo que não forneça dados quantitativamente precisos.

1.5.2 Contribuições Esperadas da Tese

»Diferencial do trabalho em tela? Contribuições?

»Uso de heurísticas de seleção de soluções candidatas e simulação em nível de sistema na exploração precoce do espaço de projeto com intuito de minimizar o espaço de projeto a ser explorado em fluxos posteriores mais detalhados quanto ao projeto de hardware

2 OBJETIVOS E RESULTADOS

Esta seção apresenta o objetivo geral e objetivos específicos referentes ao doutoramento, assim como os resultados esperados com o desenvolvimento da tese.

2.1 Objetivo Geral

0. Definir e avaliar uma metodologia de **Exploração Precoce do Espaço de Projeto** (*Early Design Space Exploration - DSE*) usando **heurísticas** de seleção de soluções candidatas e **simulação em nível de sistema** na análise de *cargas de trabalho* específicas sobre *arquiteturas multiprocessadas heterogêneas*.

A Fig. 2 ilustra o contexto referente ao objetivo geral do trabalho. Observa-se que o escopo do trabalho engloba a definição da entrada do projeto, consistindo de dados a respeito da carga de trabalho (aplicações), dos elementos da arquitetura do sistema e de restrições associadas ao sistema. Pretende-se que o tamanho do espaço de projeto gerado a partir dos dados de entrada seja reduzido para um conjunto de soluções considerando uma exploração precoce do espaço de projeto utilizando heurísticas de seleção. Heurísticas, ou Metaheurísticas, são métodos de solução que controlam a interação entre procedimentos de melhora local e estratégias de alto nível para criar processos capazes de escapar de ótimos locais e que realizem uma busca robusta de um espaço de solução (GENDREAU; POTVIN, 2010a).

2.2 Objetivos Específicos

Considerando o *objetivo geral* apresentado, nesta seção são apresentados os *Objetivos Específicos* que detalham cada aspecto necessário ao alcance do objetivo geral:

1. **Avaliar o Espaço de Projeto em Fases Iniciais de Desenvolvimento**: em fases iniciais de desenvolvimento pode não estar disponível a arquitetura de sistema ou mesmo as aplicações para a realização de testes. Assim, modelos em alto nível de abstração e simulação em nível de sistema podem ser aplicados

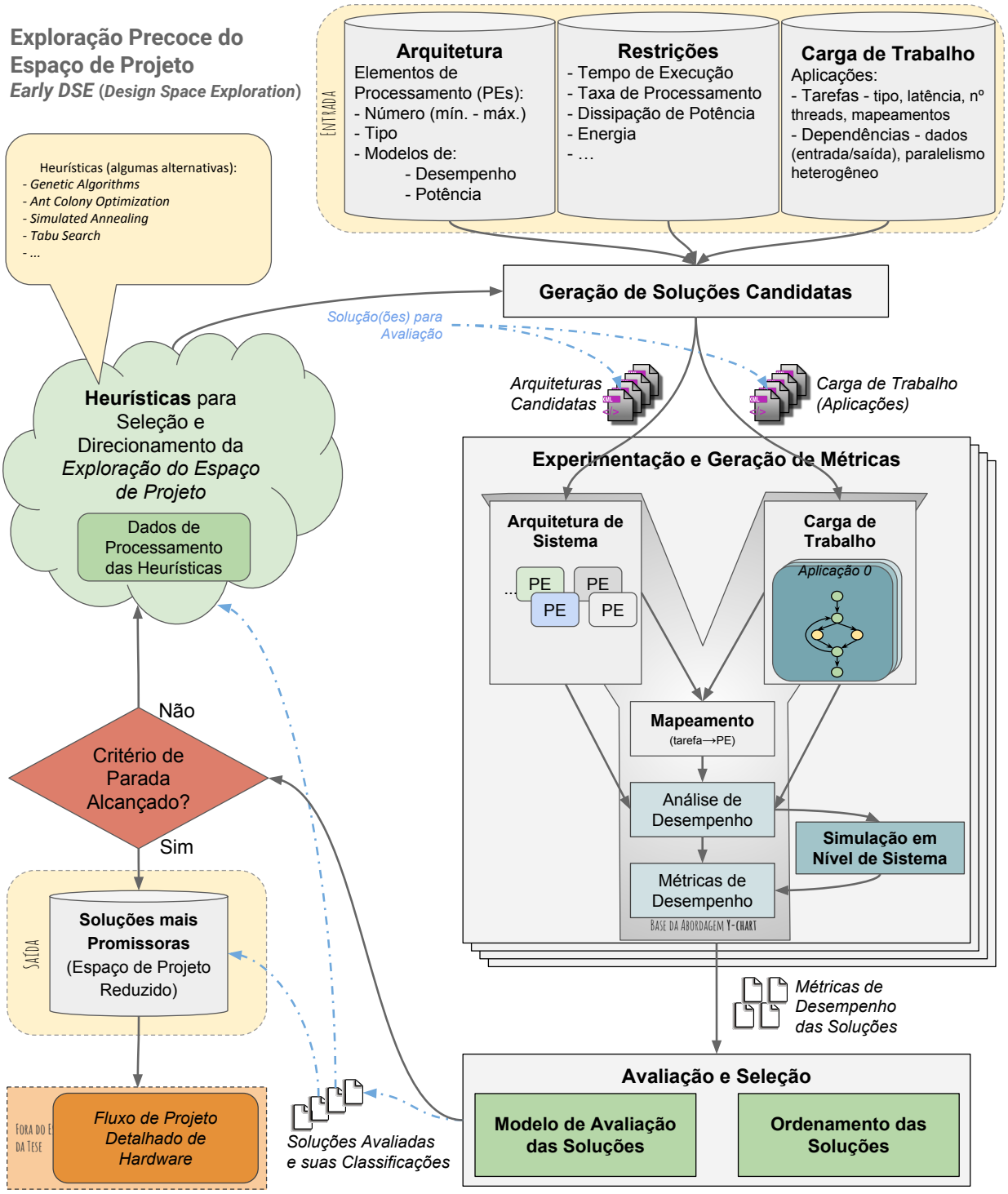

Figura 2 – Exploração Precoce do Espaço de Projeto (*Early DSE*)

para a realização da Exploração do Espaço de Projeto em fases iniciais de desenvolvimento (em inglês: *Early Design Space Exploration*), permitindo a tomada de decisões mesmo em situações onde não há disponibilidade de aplicações ou mesmo arquitetura de sistema para a realização de experimentos.

2. **Empregar Heurísticas na Seleção de Soluções Candidatas**: avaliar a utilização de diferentes heurísticas para otimizar o processo de seleção de soluções candidatas. Algumas alternativas são Algoritmos Genéticos (GA - *Genetic Algorithms*)(REEVES, 2010), recozimento simulado (*Simulated Annealing*) (NIKOLAEV; JACOBSON, 2010), Busca Tabu (*Tabu Search*) (GENDREAU; POTVIN, 2010b) e Otimização de Colonia de Formigas (ACO - *Ant Colony Optimization*) (DORIGO; STÜTZLE, 2010), dentre outras alternativas passíveis de avaliação.

3. **Utilizar Simulação em Nível de Sistema**: modelos de granularidade mais alta são utilizados em simuladores de nível de sistema. Estes simuladores normalmente permitem a rápida simulação de possíveis soluções de um dado projeto; além de utilizarem modelos de aplicações e de arquitetura normalmente gerados com menor esforço de modelagem.

4. **Empregar Modelos de Alto Nível de Abstração no DSE**: modelos de alto nível são usados para descrever **aplicações** e **arquitetura** do sistema (basicamente, seus elementos de processamento) visando a rápida geração destes modelos para avaliação em fases iniciais de desenvolvimento.

5. **Aproveitar a Heterogeneidade de Processamento Existente na Arquitetura dos Sistemas**: considerar as diferentes características dos *kernels* das aplicações e aproveitá-las na alocação dos heterogêneos elementos de processamento (PE) existentes no sistema. Os *kernels* de aplicação podem ser executados em diferentes tipos de PE, até mesmo por mais de um simultaneamente, e utilizando uma ou mais *threads* de execução. Propiciar que possam ser utilizados como elementos de processamento pelos menos CPUs, GPUs e FPGAs. Outros elementos como DSPs (*Digital Signal Processor*), ASICs (*Application-Specific Integrated Circuit*) e ASIPs (*Application-Specific Instruction-set Processor*) também podem ser avaliados no contexto do trabalho.

2.3 Resultados Esperados

Como principal resultado esperado do desenvolvimento do trabalho pode-se indicar a definição de uma **Metodologia de Exploração Precoce do Espaço de Projeto** (*Early DSE - Design Space Exploration*) a ser aplicada em *fases iniciais* de projetos de desenvolvimento de *Sistemas Embarcados*, considerando *restrições* gerais de projeto

e *modelos* em alto nível de *abstração* para as aplicações e arquitetura do sistema, também buscando ampliar a *heterogeneidade* aplicada na execução da carga de trabalho (*workload* - aplicações) submetida para tirar proveito das diferentes características das *tasks* (especificamente os *kernels*) constituintes de cada aplicação.

3 METODOLOGIA

Esta Seção apresenta a metodologia de trabalho para o desenvolvimento e alcance dos objetivos geral e específicos previamente apresentados. Inicialmente são apresentadas as atividades previstas, seguidas pela apresentação do ferramental e subsídios necessários ao andamento do trabalho.

3.1 Atividades para o Desenvolvimento do Trabalho

Para o alcance dos objetivos geral e específicos apresentados na Seção 2, o seguinte conjunto de atividades precisa ser executado para contemplar cada um dos objetivos específicos do trabalho que, por sua vez, propiciam o alcance do Objetivo Geral do Trabalho.

0. (Objetivo Geral) Definir e avaliar uma metodologia de **Exploração Precoce do Espaço de Projeto** (*Early Design Space Exploration - DSE*) usando **heurísticas** de seleção de soluções candidatas e **simulação em nível de sistema** na análise de *cargas de trabalho* específicas sobre *arquiteturas multiprocessadas heterogêneas*

   (a) »Escrever, Revisar e Entregar Texto da Tese de Doutorado

   (b) »Preparar e Realizar a Defesa da Tese de Doutorado

1. (Objetivo) Avaliar o Espaço de Projeto em Fases Iniciais de Desenvolvimento

   (a) Estruturar o ambiente necessário para Exploração do Espaço de Projeto em fases iniciais de desenvolvimento;

   (b) Avaliar e definir métricas que possam ser utilizadas para avaliação de desempenho;

   (c) Avaliar e definir o formato e estrutura dos dados de entrada do DSE, considerando aspectos de(a):

   - Arquitetura do sistema: número e tipo de elementos de processamento; modelos de potência e de desempenho;

- Carga de trabalho a ser submetida ao sistema: descrição das aplicações e suas tarefas (*tasks*);
- Restrições de Projeto referentes ao sistema ou aplicações, como, por exemplo: tempo de execução, taxa de processamento (*throughput*), dissipação de potência, consumo de energia, dentre outros possíveis.

(d) Elencar, avaliar e definir *estratégias de geração de soluções candidatas* que possam ser utilizadas considerando os dados de entrada;

(e) Elencar, avaliar e definir modelo(s) de avaliação da qualidade das soluções (configurações) experimentadas na Exploração Precoce do Espaço de Projeto;

(f) Estruturar e documentar a metodologia definida para *Early DSE*;

(g) Projetar experimento de avaliação da metodologia de *Early DSE*.

2. (Objetivo) Empregar Heurísticas na Seleção de Soluções Candidatas

(a) Buscar, na literatura acadêmica, por heurísticas de seleção de soluções candidatas a serem aplicadas em uma metodologia de DSE;

(b) Avaliar e elencar as heurísticas (inicialmente pretende-se utilizar quatro heurísticas: **A**–**D**) encontradas na literatura;

(c) Projetar, Implementar e Conduzir experimento de avaliação da heurística **A**;

(d) Projetar, Implementar e Conduzir experimento de avaliação da heurística **B**;

(e) Projetar, Implementar e Conduzir experimento de avaliação da heurística **C**;

(f) Projetar, Implementar e Conduzir experimento de avaliação da heurística **D**.

3. (Objetivo) Utilizar Simulação em Nível de Sistema

(a) Avaliar e definir simulador que possa ser utilizado na rápida simulação de uma dada carga de trabalho executada sobre uma arquitetura de sistema. O simulador SAVE (MIELE et al., 2015) apresenta características promissoras para a implementação e teste das hipóteses a serem trabalhadas na tese. O referido simulador foi alterado para permitir a execução em paralelo, e para a mesma aplicação, de tarefas (*tasks*) implementadas para diferentes elementos de processamento (como CPU e GPU). O simulador com estas alterações passou a ser denominado `SAVE-htlp` (onde `htlp` significa *heterogeneous task-level parallelism*). A Seção 3.3.1 descreve superficialmente o referido simulador;

(b) Implementar e adicionar ao simulador `SAVE-htlp` uma unidade de processamento que se enquadre como Reconfigurável, do tipo FPGA (*Field Programmable Gate Array* - Matriz de Portas Programáveis em Campo) ou CGRA

(*Coarse Grained Reconfigurable Architecture* - Arquitetura Reconfigurável de Grossa Granulação) (MANSUREH; CHO; CHOI, 2017).

4. (Objetivo) Empregar Modelos de Alto Nível de Abstração no DSE

   (a) Avaliar e definir modelos em alto nível de abstração que possam ser utilizados como entrada de projeto para o detalhamento de carga de trabalho (aplicações), arquitetura do sistema (elementos de processamento), e alternativas de mapeamento dos *kernels* das aplicações. A seção 3.3.1 descreve alguns aspectos do simulador SAVE-htlp, incluindo os modelos de arquitetura e aplicações (*workload*) utilizados pelo referido simulador, os quais julga-se adequados para utilização no contexto deste trabalho;

   (b) Estudar formas automatizadas ou semi-automatizadas para a geração de modelos de arquitetura e de carga de trabalho de um sistema em avaliação, assim como as alternativas de mapeamento disponíveis.

5. (Objetivo) Aproveitar a Heterogeneidade de Processamento Existente na Arquitetura dos Sistemas

   (a) Garantir a geração de soluções candidatas que considerem a utilização de todas as diferentes unidades de processamento existentes na arquitetura do sistema;

   (b) Garantir a geração de soluções candidatas que considerem, quando possível, a utilização simultânea de unidades de processamento heterogêneas para a execução de tarefas (especificamente *kernels* de aplicações) de mesmo objetivo (porém com implementações para diferentes tipos de elementos de processamento). No contexto deste trabalho esta possibilidade é denominada de Paralelismo Heterogêneo em Nível de Tarefa - HLTP (*Heterogeneous Task-level Paralelism*).

3.2 Obrigações Regimentais junto ao PPGC/UFPel

Considerando o regimento do Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, nesta seção são abordadas as atividades obrigatórias referentes ao curso de Doutorado. As atividades obrigatórias do programa são descritas abaixo, seguidas por um breve histórico da respectiva atividade ou uma descrição sobre o planejamento para a execução da mesma.

1. »Cursar os Créditos Obrigatórios do programa: atualmente (2018/1) 38 créditos já foram aproveitados ou cursados em disciplinas ofertadas pelo PPGC/UFPel;

2. »Preparar e Apresentar o Exame de Qualificação: em Outubro de 2017 foi apresentado o Exame de Qualificação junto a banca de avaliação, sendo considerado Aprovado;

3. »Preparar e Defender a Proposta de Tese: atividade que gerou a proposta em tela e que deve ser ser defendida (até o prazo máximo de 15 de Outubro de 2018) perante arguição de banca específica para este fim;

4. »Preparar e Apresentar Seminário de Andamento: até o final do semestre 2018/2 deverá ser apresentado o Seminário de Andamento. Este tem por finalidade apresentar as atividades desenvolvidas no trabalho considerando a proposta previamente avaliada;

5. »Escrever e Submeter Artigo(s) em Evento/Periódico da Área de Interesse: o regimento do PPGC/UFPel aponta como necessária a submissão e publicação de artigo(s) que descrevem resultados do trabalho desenvolvido durante o doutoramento.

6. »Escrever, Revisar e Entregar Texto da Tese de Doutorado

7. »Preparar e Realizar a Defesa da Tese de Doutorado

3.3 Subsídios e Ferramentas para o Desenvolvimento do Trabalho

De forma geral, o levantamento de referências da literatura acadêmica será realizado com base em bibliotecas digitais (ou repositórios) e ferramentas de busca especializadas disponíveis na internet. Também poderão ser consultados títulos disponíveis nas bibliotecas da instituição. Algumas bibliotecas digitais e fontes de consulta que podem ser utilizadas são:

- IEEEXplore - `https://ieeexplore.ieee.org/Xplore/home.jsp`;
- ACM Digital Library - `https://dl.acm.org/`;
- Elsevier - Science Direct - `https://www.sciencedirect.com/`;
- Springer Link - `https://link.springer.com/`;
- Google Acadêmico - `https://scholar.google.com.br/`;

Com respeito aos equipamentos necessários para o desenvolvimento do trabalho, dado que não há necessidade, *a priori*, de equipamentos especiais para a condução do trabalho, serão utilizados computadores/servidores disponíveis nos laboratórios da instituição, assim como *notebook* pessoal do autor do trabalho. Nos referidos computadores/servidores poderão ser realizadas tarefas gerais relacionadas às atividades

de modelagem, escrita, simulação, programação, revisão, projeto e definição de experimentos, tratamento e organização de dados, edição e geração de gráficos e figuras, dentre outras que forem julgadas necessárias ao trabalho.

3.3.1 SAVE-htlp - Simulador em Nível de Sistema

Em (MIELE et al., 2015) é apresentado um simulador implementado em SystemC e TLM para validar políticas de gerência de recursos em Arquiteturas de Sistemas Heterogêneos (*Heterogeneous System Architectures* - HSA) (HSA Foundation, 2018). O simulador (MIELE et al., 2015) busca lidar com gerenciamento em tempo de execução na alocação de recursos de processamento para as aplicações, baseado na eficiência das unidades de processamento e para atender requisitos de nível de serviços (na forma de um *Service Level Agreement*). As aplicações são modeladas como grafos de tarefas que incluem informações sobre os tipos de tarefas e suas latências, contadores de desempenho, números de linhas de execução (*threads*), dentre outros. A arquitetura de hardware é modelada como um conjunto de recursos genéricos que descrevem modelos de desempenho e de potência. A Fig. 3 ilustra modelos de aplicação e de arquitetura utilizados no referido simulador. A parte de Software (SW) da plataforma virtual é descrita à esquerda da figura. Cada nó do grafo representa uma tarefa (*task*) da aplicação, sendo caracterizadas pela latência de execução em algum modelo de elemento de processamento (PE), frequência de operação, tipo de tarefa (*elaboration* ou *main*), tamanho dos dados de entrada e saída e, quando for o caso, ponto de chamada da função de atualização de *heartbeats* (HOFFMANN et al., 2010). Além disso, para cada tarefa são definidos os mapeamentos possíveis para as unidades de processamento existentes. A arestas representam as dependências entre as tarefas e são caracterizadas pelo seu tipo e número de ciclos (para este último o valor *default* é 1). Os aspectos da arquitetura (Hardware - HW) são descritos no lado direito da figura. Os elementos de processamento da arquitetura são descritos pelo seu tipo, modelo, e dados de *dvfs* (*dynamic voltage and frequency scaling*). Os dados de *dvfs* são frequência de operação, potência em operação (*Power*) e potência em estado ocioso (*IdlePower*). Pelo menos uma descrição de *dvfs* deve existir para cada unidade de processamento.

Foi obtido junto aos autores do referido simulador (MIELE et al., 2015) acesso ao respectivo código-fonte e este parece promissor quanto às possibilidades de extensão que possam ser incorporadas como parte do trabalho de doutorado para implementação, testes e avaliação das hipóteses da tese. Assim, este simulador foi adaptado visando o aumento da heterogeneidade aplicada na execução de aplicações que possuam *tasks* (*kernels*) com paralelismo em nível de tarefa (TLP) e que permitam a criação e execução simultânea de *threads* destinadas à distintas unidades de processamento (tais como CPUs e GPUs). A versão modificada no simulador passou a

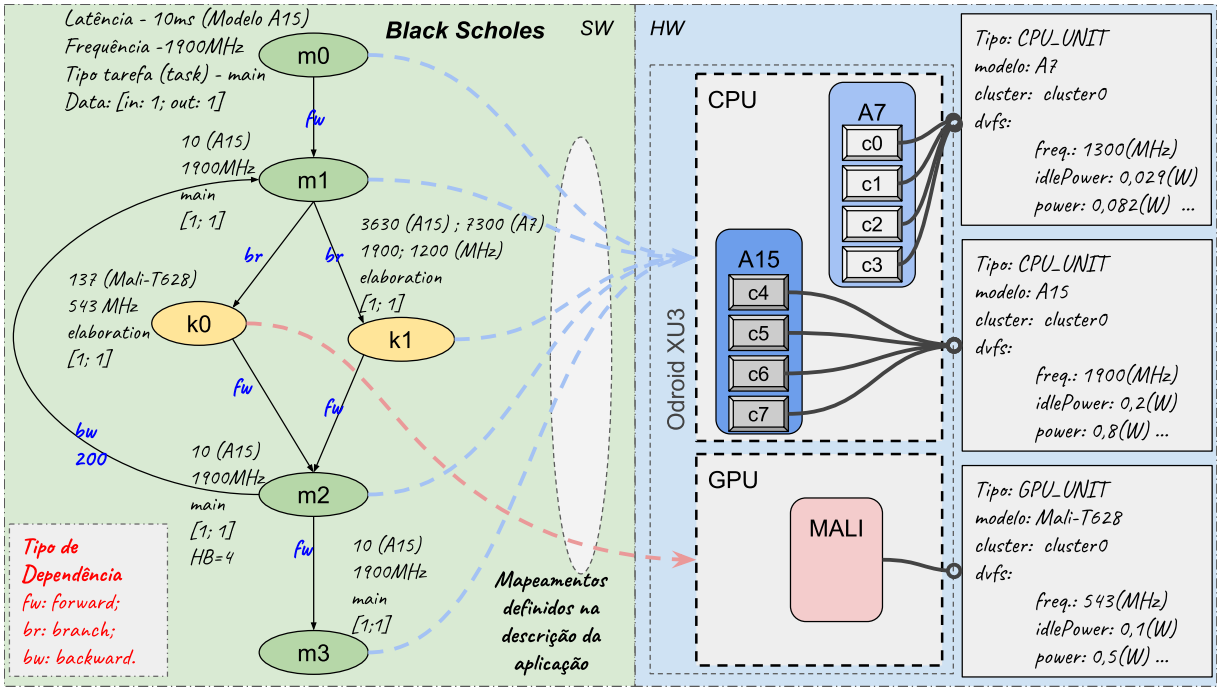

Figura 3 – Modelos de Aplicação e Arquitetura no Simulador *SAVE-htlp*

ser denominada SAVE-htlp (onde *htlp* significa *heterogeneous task-level parallelism* - paralelismo heterogêneo em nível de tarefa). Com o SAVE-htlp foram planejados e realizados alguns experimentos usando diferentes configurações para a arquitetura do sistema, combinando as *threads* referentes às *tasks* das aplicações com diferentes unidades de processamento e variando o número de *threads* executadas em cada caso. Nos experimentos, um modelo de arquitetura baseado na plataforma ODROID XU3 (HARDKERNEL, 2018) foi empregado e, como estudo de caso, foi utilizada uma aplicação de *Block Matching Algorithm* (BMA - Algoritmo de Correspondência de Blocos). BMA é aplicado em áreas como codificação de vídeo e visão computacional e para os experimentos foi utilizada como base as implementações apresentadas por Melo et al. (MELO et al., 2016). Considerando os resultados dos experimentos foi escrito um artigo científico, com o título "*Exploring Heterogeneous Task-Level Parallelism in a BMA Video Coding Application using System Level Simulation*", e submetido ao evento SBESC 2018[1] - atualmente o artigo está em processo de análise pelo comitê de programa do evento.

[1] http://sbesc.lisha.ufsc.br/sbesc2018/Home

4 CRONOGRAMA

Esta seção apresenta o cronograma de realização das atividades necessárias para o alcance do objetivo geral do trabalho. As atividades estão descritas na Seção 3, tendo sido descritas e identificadas conforme o objetivo específico ao qual cada atividade está mais diretamente relacionada.

»Inserir diagrama de Gannt relacionando as atividades, suas dependências e respectivos prazos

5 ANOTAÇÕES

5.1 Referência: (PANERATI; SCIUTO; BELTRAME, 2017)

Abordagens de DSE normalmente necessitam utilizar simulação (ou estimativa) para avaliar as métricas no nível do sistema (ou seja, as funções objetivo) do grande número de configurações que são examinadas.

A automação de DSE envolve dois sub-problemas:

- **(a)** identificação de soluções candidatas plausíveis (i.e., configurações válidas de sistema) e
- **(b)** a avaliação de métricas de interesse destas soluções objetivando selecionar configurações ótimas.

Algoritmos Evolucionários (EAs - *Evolutionary Algorithms*) são amplamente usados em DSE. EAs discriminam e selecionam soluções usando uma combinação de métricas representadas por uma função de *fitness*. Além disso, EAs são normalmente fáceis de aplicar e não necessitam conhecimento detalhado do espaço a ser explorado. Abordagens Evolucionárias para otimização multi-objetivos (classificação):

1. Algoritmos que utilizam funções agregadoras (*aggregating functions*)
2. Algoritmos que utilizam funções não-agregadoras (*non-aggregating functions*) mas abordagens não baseadas em Pareto
3. Abordagens baseadas em Pareto

Considerando um escopo mais amplo, para além das abordagens evolucionárias, classificam-se as abordagens para o sub-problema (a) em:

- **Classe 1**: *Abordagens de Otimização Pseudo Aleatória e Heurísticas*. Estas abordagens normalmente trabalham com a ideia de *clusters* a serem explorados exaustivamente. Exemplos: Otimização por multi-agentes (como *Particle Swarm Optimization* - PSO), Recozimento Simulado (*Simulated Annealing* - SA), Busca Tabu (*Tabu Search*) e Algoritmos de Pesquisa de Operações (*operations research algorithms*);

- **Classe 2**: *Algoritmos Evolucionários*. Caracterizam-se por modificações aleatórias sobre o conjunto inicial de configurações buscando de forma iterativa melhorar o conjunto de soluções de Pareto. Exemplos: Algoritmos Genéticos (*Genetic Algorithms* - GA). Técnicas podem ser combinadas com métodos exatos - e.g.: DSE é transformado em um problema de Programação Linear Inteira 0-1 Multi-Objetivos e um solucionador pseudo-Booleano (PB) é usado para restringir o GA dentro de um espaço de busca factível.

- **Classe 3**: *Abordagens Estatísticas sem Conhecimento do Domínio*. Usam um *metamodelo* extraído do domínio para prever as próximas configurações a serem avaliadas - e.g.: Projeto de Experimentos (*Design of Experiments* - DoE).

- **Classe 4**: *Abordagens Estatísticas com Conhecimento do Domínio*. Utilizam regras pré-definidas e conhecimento específico do espaço de projeto para encontrar as soluções mais promissoras. É criado um framework probabilístico para guiar a identificação de novas soluçõe candidatas. E.g.: Processo de Decisão de Markov (*Markov Decision Process* - MDP).

Para o sub-problema **(b)** a DSE pode ser conduzida por meio de **simulação detalhada** ou **modelos preditivos mais simples**, ou mesmo uma combinação de ambos.

Exemplos de algumas Métricas para Avaliação de Conjuntos Aproximados de Pareto:

- (a) ADRS - *Average Distance from Reference Set*
- (b) *Non-uniformity*
- (c) *Concentration*

Esforços Iniciais de Configuração e Sensitividade dos Parâmetros (*Initial Setup Effort and Parameter Sensitivity*)

Exemplos de *frameworks* e bibliotecas para Problemas de Otimização Multi-Objetivos: jMetal (`http://jmetal.sourceforge.net/`), PaGMO/PyGMO (`https://github.com/esa/pagmo/` –> `https://github.com/esa/pagmo2`), MOMHLib++ (`https://github.com/derino/maponoc/tree/master/libs/libmomh-1.91.3`) e NASA (*Non Ad-hoc Search Algorithm*).

REFERÊNCIAS

ABDALLAH, F.; TRABELSI, C.; ATITALLAH, R.; ABED, M. Model-Driven Approach for Early Power-Aware Design Space Exploration of Embedded Systems. **Journal of Signal Processing Systems**, [S.l.], p.1–16, 2016.

ADLER, R.; SCHAEFER, I.; TRAPP, M.; POETZSCH-HEFFTER, A. Component-based Modeling and Verification of Dynamic Adaptation in Safety-critical Embedded Systems. **ACM Trans. Embed. Comput. Syst.**, New York, NY, USA, v.10, n.2, p.20:1—-20:39, 2011.

ALTENBERND, P.; GUSTAFSSON, J.; LISPER, B.; STAPPERT, F. Early execution time-estimation through automatically generated timing models. **Real-Time Systems**, [S.l.], v.52, n.6, p.731–760, 2016.

AN, X.; GAMATIÉ, A.; RUTTEN, E. High-level design space exploration for adaptive applications on multiprocessor systems-on-chip. **Journal of Systems Architecture**, [S.l.], v.61, n.3–4, p.172–184, 2015.

ASCIA, G. et al. Performance evaluation of efficient multi-objective evolutionary algorithms for design space exploration of embedded computer systems. **Applied Soft Computing**, [S.l.], v.11, n.1, p.382–398, 2011.

BERTELS, K. (Ed.). **Hardware/Software Co-design for Heterogeneous Multi-core Platforms**: The hArtes Toolchain. [S.l.]: Springer Netherlands, 2012. p.1–8.

BETEMPS, C. M. **Desenvolvimento de Sistemas Embarcados usando Alto Nível de Abstração**. Exame de Qualificação de Doutorado. Programa de Pós-Graduação em Computação (PPGC) - Universidade Federal de Pelotas (UFPel). Orientador: Bruno Zatt. Co-Orientadores: Marcelo Porto e Daniel Palomino. Banca de Avaliação: Júlio C. B. de Mattos e Maurício L. Pilla. Data do Exame: 25-Out-2017, URL: `https://drive.google.com/file/d/0BzYkPIeXn1jaLUJsR25RNFlhd0U/view?usp=sharing`.

CAI, L.; GAJSKI, D. Transaction Level Modeling: An Overview. In: IEEE/ACM/IFIP INTERNATIONAL CONFERENCE ON HARDWARE/SOFTWARE CODESIGN AND

SYSTEM SYNTHESIS, 1., 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p.19–24. (CODES+ISSS '03).

CALLOU, G. et al. Energy consumption and execution time estimation of embedded system applications. **Microprocessors and Microsystems**, [S.l.], v.35, n.4, p.426–440, 2011.

CAMPOSANO, R.; WILBERG, J. Embedded system design. **Design Automation for Embedded Systems**, [S.l.], v.1, n.1, p.5–50, 1996.

De Micheli, G.; GUPTA, R. K. Hardware/software co-design. **Proceedings of the IEEE**, [S.l.], v.85, n.3, p.349–365, 1997.

DIGUET, J.-P.; EUSTACHE, Y.; GOGNIAT, G. Closed-loop–based Self-adaptive Hardware/Software-Embedded Systems: Design Methodology and Smart Cam Case Study. **ACM Trans. Embed. Comput. Syst.**, New York, NY, USA, v.10, n.3, p.38:1—-38:28, 2011.

DORIGO, M.; STÜTZLE, T. Ant Colony Optimization: Overview and Recent Advances. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2010. p.227–263.

DUHEM, F.; MULLER, F.; BONAMY, R.; BILAVARN, S. FoRTReSS: a flow for design space exploration of partially reconfigurable systems. **Design Automation for Embedded Systems**, [S.l.], v.19, n.3, p.301–326, Sep 2015.

DURELLI, G. C. et al. Runtime resource management in heterogeneous system architectures: The save approach. In: PARALLEL AND DISTRIBUTED PROCESSING WITH APPLICATIONS (ISPA), 2014 IEEE INTERNATIONAL SYMPOSIUM ON, 2014. **Anais...** [S.l.: s.n.], 2014. p.142–149.

ERBAS, C.; PIMENTEL, A. D.; THOMPSON, M.; POLSTRA, S. A framework for system-level modeling and simulation of embedded systems architectures. **EURASIP Journal on Embedded Syst.**, [S.l.], n.1, p.82123, 2007.

FALKNER, K. et al. Model-driven performance prediction of systems of systems. **Software & Systems Modeling**, [S.l.], p.1–27, 2016.

GAJSKI, D. D.; ABDI, S.; GERSTLAUER, A.; SCHIRNER, G. **Embedded system design**: modeling, synthesis and verification. [S.l.]: Springer Science & Business Media, 2009.

GASTER, B. et al. **Heterogeneous Computing with OpenCL**. [S.l.]: Morgan Kaufmann, 2012.

GENDREAU, M.; POTVIN, J.-Y. **Handbook of Metaheuristics**. 2.ed. [S.l.]: Springer Science+Business Media, 2010. v.146.

GENDREAU, M.; POTVIN, J.-Y. Tabu Search. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2010. p.41–59.

GHEORGHITA, S. V. et al. System-scenario-based Design of Dynamic Embedded Systems. **ACM Trans. Des. Autom. Electron. Syst.**, New York, NY, USA, v.14, n.1, p.3:1—-3:45, 2009.

GLAß, M.; TEICH, J.; LUKASIEWYCZ, M.; REIMANN, F. Hybrid Optimization Techniques for System-Level Design Space Exploration. In: HA, S.; TEICH, J. (Ed.). **Handbook of Hardware/Software Codesign**. Dordrecht: Springer Netherlands, 2017. p.217–246.

GOMAA, H. **Software modeling and design**: UML, use cases, patterns, and software architectures. [S.l.]: Cambridge University Press, 2011.

GRIES, M. Methods for evaluating and covering the design space during early design development. **Integration, the VLSI journal**, [S.l.], v.38, n.2, p.131–183, 2004.

GRÜTTNER, K. et al. The COMPLEX reference framework for HW/SW co-design and power management supporting platform-based design-space exploration. **Microprocessors and Microsystems**, [S.l.], v.37, n.8, p.966–980, 2013.

GUESSI, M.; NAKAGAWA, E. Y.; OQUENDO, F.; MALDONADO, J. C. Architectural Description of Embedded Systems: A Systematic Review. In: INTERNATIONAL ACM SIGSOFT SYMPOSIUM ON ARCHITECTING CRITICAL SYSTEMS, 3., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p.31–40. (ISARCS '12).

HARDKERNEL. **ODROID-XU3**. [S.l.]: Hardkernel co., 2018. `http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127`.

HENDRIKS, M. et al. A blueprint for system-level performance modeling of software-intensive embedded systems. **International Journal on Software Tools for Technology Transfer**, [S.l.], v.18, n.1, p.21–40, 2016.

HERRERA, F. et al. The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems. **Journal of Systems Architecture**, [S.l.], v.60, n.1, p.55–78, 2014.

HOFFMANN, H. et al. Application Heartbeats: A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments. In: INT. CONF. ON AUTONOMIC COMPUTING, 7., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.79–88.

HSA Foundation. [S.l.]: HSA Foundation, 2018. http://www.hsafoundation.com/.

HUANG, C.-H.; HSIUNG, P.-A.; SHEN, J.-S. UML-based hardware/software co-design platform for dynamically partially reconfigurable network security systems. **Journal of Systems Architecture**, [S.l.], v.56, n.2–3, p.88–102, 2010.

HUANG, K.; XIU, S.-w. Profiling and annotation combined method for multimedia application specific MPSoC performance estimation. **Frontiers of Information Technology & Electronic Engineering**, [S.l.], v.16, n.2, p.135–151, 2015.

INDRUSIAK, L. S.; DZIURZANSKI, P.; SINGH, A. K. **Dynamic Resource Allocation in Embedded, High-Performance and Cloud Computing**. [S.l.]: River Publishers, 2016.

JIA, Z. J. et al. A System-level Infrastructure for Multidimensional MP-SoC Design Space Co-exploration. **ACM Trans. Embed. Comput. Syst.**, New York, NY, USA, v.13, n.1s, p.27:1–27:26, Dec. 2013.

JIANG, K.; ELES, P.; PENG, Z. Power-Aware Design Techniques of Secure Multimode Embedded Systems. **ACM Transactions on Embedded Computing Systems**, New York, NY, USA, v.15, n.1, p.1–29, 2016.

JIANG, Y. et al. Design and Optimization of Multi-clocked Embedded Systems Using Formal Technique. In: JOINT MEETING ON FOUNDATIONS OF SOFTWARE ENGINEERING, 2013., 2013, New York, NY, USA. **Proceedings...** ACM, 2013. p.703–706. (ESEC/FSE 2013).

KIM, D. H.; HONG, J. E. ESUML-EAF: a framework to develop an energy-efficient design model for embedded software. **Software and Systems Modeling**, [S.l.], v.14, n.2, p.795–812, 2015.

LEITE, M.; WEHRMEISTER, M. A. System-level design based on UML/MARTE for FPGA-based embedded real-time systems. **Design Automation for Embedded Systems**, [S.l.], v.20, n.2, p.127–153, 2016.

MANSUREH, M. S.; CHO, J.-M.; CHOI, K. Reconfigurable Architectures. In: HA, S.; TEICH, J. (Ed.). **Handbook of Hardware/Software Codesign**. Dordrecht: Springer Netherlands, 2017. p.335–376.

MARWEDEL, P. **Embedded system design**. [S.l.]: Springer, 2006. v.1.

MELO, M. et al. A parallel Motion Estimation solution for heterogeneous System on Chip. In: INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2016 29TH SYMPOSIUM ON, 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

MIELE, A.; DURELLI, G. C.; SANTAMBROGIO, M. D.; BOLCHINI, C. A System-Level Simulation Framework for Evaluating Resource Management Policies for Heterogeneous System Architectures. In: DIGITAL SYSTEM DESIGN, 2015 EUROMICRO CONF. ON, 2015. **Anais...** [S.l.: s.n.], 2015. p.637–644.

MISCHKALLA, F.; HE, D.; MUELLER, W. Closing the gap between UML-based modeling, simulation and synthesis of combined HW/SW systems. In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), 2010, 2010. **Anais...** [S.l.: s.n.], 2010. p.1201–1206.

NIKOLAEV, A. G.; JACOBSON, S. H. Simulated Annealing. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2010. p.1–39.

NOGUEIRA, B. et al. Multi-objective optimization of multimedia embedded systems using genetic algorithms and stochastic simulation. **Soft Computing**, [S.l.], p.1–18, 2016.

OUNI, B. et al. Multi-level energy/power-aware design methodology for MPSoC. **Journal of Parallel and Distributed Computing**, [S.l.], v.100, p.203–215, 2017.

PANERATI, J.; SCIUTO, D.; BELTRAME, G. Optimization Strategies in Design Space Exploration. In: HA, S.; TEICH, J. (Ed.). **Handbook of Hardware/Software Codesign**. Dordrecht: Springer Netherlands, 2017. p.189–216.

PATEL, H. D.; SHUKLA, S. K. **Ingredients for Successful System Level Design Methodology**. [S.l.]: Springer, 2008.

PEDRE, S.; KRAJNÍK, T.; TODOROVICH, E.; BORENSZTEJN, P. Accelerating embedded image processing for real time: a case study. **Journal of Real-Time Image Processing**, [S.l.], v.11, n.2, p.349–374, 2016.

RAYMOND, P. et al. Timing analysis enhancement for synchronous program. **Real-Time Systems**, [S.l.], v.51, n.2, p.192–220, 2015.

REEVES, C. R. Genetic Algorithms. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2010. p.109–139.

RODRÍGUEZ, R. J. A Petri net tool for software performance estimation based on upper throughput bounds. **Automated Software Engineering**, [S.l.], v.24, n.1, p.73–99, 2017.

ROGERS, P. Heterogeneous system architecture overview. In: IEEE HOT CHIPS 25 SYMP. (HCS), 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.1–41.

SCARPAZZA, D. P. et al. Software Simultaneous Multi-Threading, a Technique to Exploit Task-Level Parallelism to Improve Instruction- and Data-Level Parallelism. In: INTEGRATED CIRCUIT AND SYSTEM DESIGN. POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION, 2006, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2006. p.12–23.

SORBER, J. et al. Eon: A Language and Runtime System for Perpetual Systems. In: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 5., 2007, New York, NY, USA. **Proceedings...** ACM, 2007. p.161–174. (SenSys '07).

STRALEN, P. van; PIMENTEL, A. Scenario-based design space exploration of MPSoCs. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.305–312.

TALBI, E.-G. **Metaheuristics**: from design to implementation. New Jersey, USA: John Wiley & Sons, 2009.

TRINDADE, A. B.; CORDEIRO, L. C. Applying SMT-based verification to hardware/software partitioning in embedded systems. **Design Automation for Embedded Systems**, [S.l.], v.20, n.1, p.1–19, 2016.

VOELTER, M.; RATIU, D.; KOLB, B.; SCHAETZ, B. Mbeddr: Instantiating a language workbench in the embedded software domain. **Automated Software Engineering**, [S.l.], v.20, n.3, p.339–390, 2013.

WAIDYASOORIYA, H. M.; HARIYAMA, M.; UCHIYAMA, K. **Design of FPGA-Based Computing Systems with OpenCL**. [S.l.]: Springer, 2018.

WANG, R.; SONG, X.; ZHU, J.; GU, M. Formal modeling and synthesis of programmable logic controllers. **Computers in Industry**, [S.l.], v.62, n.1, p.23–31, 2011.

6 ASSINATURAS

Carlos Michel Betemps
Proponente

Bruno Zatt
Prof. Orientador