Um Ambiente Baseado em Blocos para Programação Paralela com OpenCL

Josué da Silva Gomes Júnior[1], **Alisson Vasconcelos de Brito**[2]

[1]Centro de Ciências Aplicadas e Educação – Universidade Federal da Paraíba (UFPB)
Av. Santa Elizabete, 160 – CEP 58297–000 – Rio Tinto – PB – Brazil

[2]Centro de Informática – Universidade Federal da Paraíba (UFPB)
CEP 58055–000 – João Pessoa – PB – Brazil

josue.gomes@dce.ufpb.br, alisson@ci.ufpb.br

Abstract. *This paper presents the educational software Blockly OpenCL. Its use is intended to help the teaching-learning parallel programming paradigm with OpenCL. It was used visual programming paradigm, through a block-based development environment made with Google Blockly API, which allows the user to create applications manipulating blocks and export them to the OpenCL C. The Blockly OpenCL language is also composed for a web page containing contextualizing Blockly OpenCL, material support for OpenCL API and the development environment with blocks.*

Resumo. *Este trabalho apresenta o software educativo Blockly OpenCL. Sua utilização tem por objetivo auxiliar o ensino-aprendizagem do paradigma de programação paralela com OpenCL. Nele foi utilizado o paradigma de programação visual, através de um ambiente de desenvolvimento baseado em blocos feito com a API do Google Blockly, que permite ao usuário criar aplicações manipulando blocos e exportá-los para a linguagem OpenCL C. O Blockly OpenCL também é composto por uma página web contendo uma contextualização do Blockly OpenCL, materiais apoio sobre a API OpenCL e sobre o ambiente de desenvolvimento com Blocos.*

1. Introdução

Utilizados principalmente na indústria de entretenimento, os processadores *multicore* e *manycore* trazem em si recursos de processamento distintos como CPU *(central Processing Unit)* e GPU *(Graphical Processing Unit)* respectivamente, promovendo o aumento de desempenho dos sistemas atuais. Esses processadores estão presentes em: *smartphones*, *smart tvs*, *single board computer* e computadores pessoais. Nas últimas décadas estão sendo cada vez mais utilizados em pesquisas científicas, conforme afirmam [Silveira et al. 2010].

Os mais diferentes tipos de dispositivos e plataformas fazem com que a programação paralela se torne uma atividade complexa. Atualmente, há algumas APIs *(Application Programming Interface)* e frameworks que provêm interfaces que facilitam

[0]Trabalho de Conclusão de Curso apresentado pelo aluno Josué da Silva Gomes Júnior sob a orientação do professor Alisson Vasconcelos de Brito como parte dos requisitos para obtenção do grau de Licenciado em Ciência da Computação na UFPB Campus IV

a programação para tais dispositivos, entre elas estão o CUDA *(Compute Unified Device Architecture)*, o OpenMP *(Open Multi-Processing)* e o OpenCL *(Open Computing Language)*. Mesmo trazendo algumas facilidades na programação, ainda são necessárias algumas habilidades no desenvolvimento de aplicações paralelas, como programar com baixo nível de abstração com instruções próximas ao comando de máquina. Assim, é percebido uma rigorosa curva de aprendizagem necessária para lidar com essas APIs [Scarpino 2012]. O OpenCL é um assunto que pode ser considerado complexo, isso ocorre devido ao alto nível de conhecimento exigido do desenvolvedor, como conhecer a programação do hospedeiro, a programação do dispositivo e o mecanismo de transferência de dados entre o hospedeiro e o dispositivo [Scarpino 2012].

A linguagem visual vem se mostrando bem promissora no auxílio ao ensino de programação de computadores. Com ela é possível trabalhar com os mais diversos públicos, levando em conta a forma intuitiva de programar, manipulando blocos com funções pré-programadas e convertidas ao modo textual de programação [da Silveira Júnior et al. 2015]. Essa prática facilita e agiliza o desenvolvimento de aplicações, pois, com a junção de alguns blocos, de forma logica, é possível obter várias linhas de código de um programa procedural. Diminuindo a barreira ao aprender ou desenvolver novos programas.

Tendo em vista as vantagens no uso de linguagens visuais, temos como objetivo implementar um ambiente para desenvolvimento de uma linguagem visual, baseada em blocos. Esse ambiente tornará transparente o desenvolvimento em programação paralela com OpenCL C e permitirá desenvolver aplicações paralelas. O ambiente foi desenvolvido a partir da API Blockly[Google Developers 2016], que permite a criação de novos conjuntos de blocos para a exportação em uma dada linguagem de programação.

Para este trabalho foi escolhida como base para nosso estudo a API OpenCL, pelo fato de ser aberta, livre de *royalties*, com finalidades gerais, funcionar em plataformas heterogêneas e possuir uma comunidade ativa, além de possuir suporte em algumas linguagens como C, C++, Python, Java e Javascript. Para o desenvolvimento da página web com os tutoriais foi criada seguindo o conceito de *Web-based learning* (WBL), que inclui algumas vantagens técnicas, como acessibilidade universal, fácil atualização do conteúdo e funções de hiperlink, que permite referenciar para outros recursos. A escolha do formato de informação se deu pelo impacto que essa pode ter na aprendizagem dos conteudos. De acordo com [Mitchell et al. 2005], o formato hipermídia suporta uma aproximação mais flexível para a instrução que ajuda estudantes a trabalhar com o conteúdo de diferentes perspectivas.

O trabalho está estruturado em 8 seções que serão apresentadas da seguinte maneira: a seção 2 apresenta o Fundamentação Teórica; a seção 3 apresenta os Trabalhos Relacionados; a seção 4 mostra as Contribuições para a ferramenta; a seção 5 apresenta a Metodologia; na seção 6 mostra os Resultados; na seção 7 as Considerações Finais e na seção 8 os Trabalhos Futuros.

2. Fundamentação Teórica

A Computação paralela é definida por [Gaster et al. 2012] como “uma forma de computação em que muitos cálculos são feitos simultaneamente, operando no princípio em que grandes problemas podem ser divididos em menores, e que então são resolvidos

simultaneamente".

Em razão ao alto poder computacional dos GPUs, presentes até mesmo nos computadores para fins domésticos, a comunidade cientifica passou a utilizá-los para computar grande parte dos algoritmos numéricos, auxiliando assim na resolução problemas científicos e da engenharia. [Kowalik and Puźniakowski 2012]. Antes esses dispositivos eram utilizados para fins de entretenimento com renderização de vídeos e jogos.

Segundo a [Nvidia 2016], CUDA™ é uma plataforma de computação paralela e um modelo de programação inventados pela NVIDIA. Ela permite aumentos significativos de performance computacional ao aproveitar a potência da unidade de processamento gráfico (GPU). OpenMP é uma API portátil de modelo escalável que provê para a programação multi-processo de memória compartilhada em múltiplas plataformas. Permite acrescentar simultaneidade aos programas escritos em C, C++ e Fortran [OpenMP.org 2016].

[Khronos Group et al. 2011] conceitua o OpenCL como uma API e uma linguagem de programação de padrão aberto e livre de *royalties* compatível com múltiplas plataformas para programação paralela. De diversos processadores encontrados em computadores pessoais, servidores, dispositivos móveis e plataformas embarcadas. Mantida atualmente pelo Khronos Group.

De acordo com [Shen et al. 2012] OpenMP é simples e tradicional com um solido *background* nas comunidades de computação de alto desempenho (HPC) enquanto o OpenCL é um novato no mundo dos GPGPU *(General Purpose Graphics Processing Unit)*, tipicamente adotado por sua portabilidade entre GPUs e CPUs, detalha também que há uma diferença de performance entre os dois, e que em testes feitos, o OpenCL tem um melhor desempenho, uma vez que seja necessário um grande número de processos. Já em comparação ao CUDA, o OpenCL tem a preferência por ser livre e haver compatibilidade entre várias marcas e plataformas, enquanto o CUDA funciona apenas com os Produtos da Nvidia.

O desenvolvimento do OpenCL foi motivado pela necessidade de um padrão de uma plataforma de desenvolvimento de aplicações de alta performance pelo rápido crescimento de variadas plataformas de computação paralela [Kirk and Hwu 2010]. O objetivo do padrão OpenCL é unificar em um único paradigma e conjunto de ferramentas o desenvolvimento de soluções de computação paralela para dispositivos de naturezas distintas [Silveira et al. 2010]. O OpenCL padroniza o desenvolvimento de programação paralela, pois oferece um conjunto comum de ferramentas para tomar vantagem de qualquer dispositivo com suporte à OpenCL para o processamento de código paralelo, por criar uma interface de programação bem próxima ao hardware [Banger and Bhattacharyya 2013].

Em contrapartida, A API nativa do OpenCL requer do desenvolvedor uma grande quantidade de "*boilerplate code*" [1]. Outro problema ao começar a programar em OpenCL devido a necessidade de familiarizar-se com a linguagem de programação e com um novo paradigma de programação [Demidov et al. 2013].

[1]Em Programação de computadores "boilerplate code" refere-se a uma seção de código que deve ser inclusa em vários locais com pouca ou nenhuma alteração, o programador deve escrever muito código para o mínimo de resultado

[Foldoc 2016] define linguagem de programação visual como sendo "qualquer linguagem de programação que permita ao usuário criar programas manipulando elementos gráficos". A exemplo temos o Google Blockly, o MIT App inventor [MIT 2016], Scratch [MIT Scratch 2016], entre muitos outros. [Brock 2014] Fala em seu trabalho "As linguagens de programação visual pode ser a mais forte alternativa existente a prática de programação convencional, tendo como público alvo crianças do ensino básico até analistas de laboratório especializado". Segundo [Mélo et al. 2011], O uso de linguagens visuais na aprendizagem de programação garante que os alunos se concentrem muito mais na lógica de programação do que na sintaxe da linguagem, e que a linguagem visual permite que os estudantes usem a codificação em linguagem de alto nível como modelo para a construção de códigos em linguagens de programação profissionais.

O Blockly é uma interface de desenvolvimento de linguagem visual totalmente "*web-based*" onde os usuários podem arrastar blocos, e gerar novos códigos. O Blockly não é uma linguagem de programação, ele apenas converte os blocos em linguagem procedural, tais como Python, Javascript, PHP e outras [Google Developers 2016]. O Google Blockly Developer fornece um framework onde é possível criar blocos e exportá-los para a linguagem desejada. Sendo necessário apenas definir o formato, os campos e os pontos de conexão, essa funcionalidade é bastante utilizada para ajudar aprendizes em programação a criar uma conexão entre a linguagem visual e a linguagem de programação tradicional [Crawford et al. 2016]. A seguir na Figura 1, é apresentado a interface do ambiente do Google Blockly.

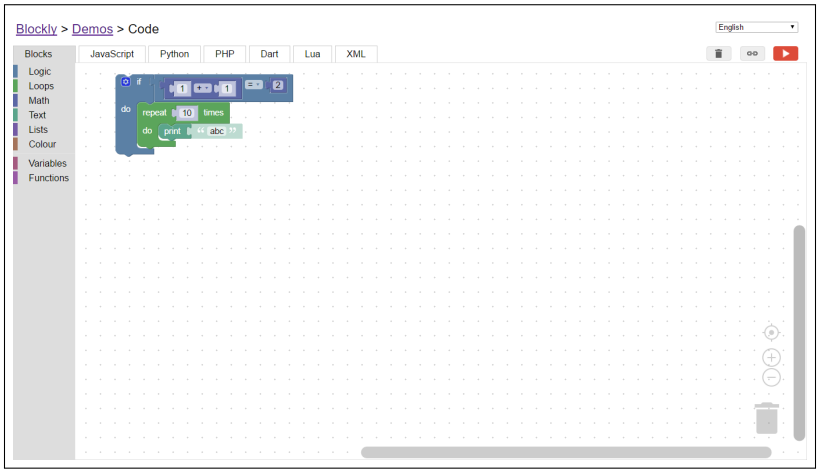

Figura 1. Interface Google Blockly

A natureza visual do Blockly adiciona usabilidade, que faz dele um grande framework para o desenvolvimento de aplicações de aprendizagem para programadores novatos. E que mais de 137 milhões de usuários usam o Blockly framework como parte do Code.org[2] e hora do código[3]. Atualmente mais de 5 milhões de estudantes estão em cursos de programação utilizando linguagem de Programação Visual (LPV). [Crawford et al. 2016].

[2]É um projeto que incentiva o ensino de ciência da computação para crianças e adolescentes https://code.org/

[3]É um movimento que visa o ensino de programação através de uma plataforma baseada em programação visual https://hourofcode.com/br/pt

2.1. Paralelismo e OpenCL

Como explica [Tupinambá 2013] em seu trabalho, uma aplicação típica em OpenCL tem quatro passos que se repetem na definição do *host* na maioria dos programas em OpenCL. São eles:

1. Levantamento dos dispositivos disponíveis, criação do contexto de execução;
2. Envio de dados para os dispositivos, copiando do *host*, que normalmente é a CPU para os dispositivos.
3. Execução do *Kernel* nos dispositivos e são passados os parâmetros através da chamada da API OpenCL, que é executado de forma assíncrona em relação ao *host*;
4. Leitura dos resultados gerados pela execução nos dispositivos

Conforme [Kowalik and Puźniakowski 2012] fala em seu livro. O OpenCL trabalha com duas estratégias de paralelismo o *Data Parallelism* (paralelismo de dados) e o *Task Parallelism* (paralelismo de tarefas).

- O Paralelismo de dados, também chamado de *Single Program Multiple Data* (SPMD). Em um código de dados paralelos, estrutura de dados como matrizes são divididos em blocos, conjuntos de linhas e colunas, e um único programa executa operações idênticas nessas partições que contém diferentes dados.
- A abordagem do paralelismo de tarefas é mais geral, assumindo que há múltiplas e diferentes tarefas independentes que podem ser computadas em paralelo. As tarefas operam em seus próprios conjuntos de dados. Paralelismo de tarefas pode ser chamado de *Multiple Programs Multiple Data* (MPMD).

De acordo com [CodePlex 2016] OpenCL é dividido em duas partes, a primeira parte é chamada de host programado em C++, onde é definido os quatros passos descritos anteriormente por Tupinambá e é executada no CPU, a outra parte é o *Kernel* programado em OpenCL, que são os programas executados nos *Devices* e compilado em tempo de execução. A seguir são apresentados alguns detalhes sobre a execução do OpenCL nos *Devices*

- **Kernel**: são como funções que são executadas nos dispositivos, os kernels são as únicas funções que podem ser chamadas pelo *host*.
- **SIMT**: é um padrão chamado*Single Instrunction Multiple Thread*, e reflete como as instruções são executadas no *host*, o código e executado em paralelo por uma diferente *thread* e cada *thread* executa o código com dados diferentes, sendo levado em consideração o contexto de threads.
- **Work-Item**: o *Work-item* é a menor entidade de execução, toda vez que o *kernel* é executado, muitos Work-itens são executados, cada *Work-item* tem um ID, que é acessível pelo kernel e que é usado para distinguir os dados para ser processado por cada *Work-item*.
- **Work-group**: *Work-groups* existem para permitir a comunicação e cooperação entre *Work-itens*. Eles refletem como os *Work-itens* são organizados, como os *Work-itens* os *Work-groups* tem um único ID que pode ser referenciado pelo *Kernel*.

Na imagem abaixo vemos na Figura 2, a representação dos work-itens dentro dos work-groups, onde o kernel é enviado para cada work-item para executa-los paralelamente. Outro fator é o número de work-itens que serão utilizados e que deve ser passado

pelo usuário e assim esse número será divido entre os work-groups. Os work-itens tem em si um id que vai de zero ao número máximo de work-itens presentes no dispositivo. Toda instrução que é enviada aos dispositivos é passado através de um enfileiramento, cada dispositivo possui sua própria fila [Silveira et al. 2010].

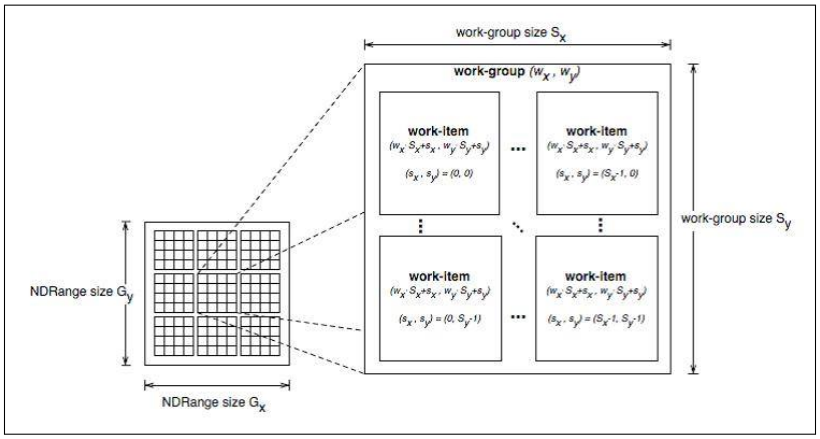

Figura 2. Representação dos work-itens e dos work-group

2.2. O potencial pedagógico da ferramenta Blockly OpenCL

Devido ao objetivo da ferramenta de auxiliar no ensino-aprendizagem, o ambiente Blockly OpenCL se caracteriza como um software educativo, conforme define [Soffa and Alcântara 2008] ”O software educativo é desenvolvido com o desígnio de levar o aluno a construir um determinado conhecimento referente a um conteúdo didático. O objetivo de um software educativo é a de favorecer os processos de ensino-aprendizagem e sua característica principal é seu caráter didático.”e ressalta também que ”O emprego destes programas não garantirá por si só a aprendizagem dos alunos, pois os mesmos são instrumentos didáticos de ensino que podem e devem estar a serviço do processo de construção e assimilação do conhecimento dos aprendizes”.

Segundo [Vieira 1999], o software educativo pode ser classificado em tipos, dentre eles: Tutorial, Exercícios e Práticas, Programação, Aplicativos, Multimédia e Internet, Simulação e Jogos. O blockly OpenCL se enquadra em dois deles: o Tutorial no qual a informação é organizada de acordo com uma sequência pedagógica particular. Seguindo essa sequência o aprendiz pode escolher a informação que desejar, e de Programação que permitem ao aprendiz representar sua ideia fazendo uma correspondência direta entre cada comando e comportamento do computador.

O Blockly OpenCL se enquadra em mais dois tipos de classificação: Sequêncial, em que o aluno segue as informações que lhe são transmitidas de forma sequêncial, memorizando e repetindo conteúdos quando solicitado, resultando em um aprendizado passivo e sem reflexão, e Relacional em que o aluno pode fazer relações com outros fatos ou outras fontes de informação. Como a ênfase é dada a interação do aprendiz com a tecnologia, o resultado é o desenvolvimento de um aprendiz isolado. Diante das características citadas acima se enquadra no Sequencial com a página de tutoriais e Relacional no ambiente de desenvolvimento com blocos.

Outro fator importante em um software educativo é se enquadrar em uma teoria de aprendizagem, nesse caso O Blockly OpenCL se adequa a teoria de aprendizagem

Comportamentalismo, devido a possuir as seguintes características: treinar os estudantes a exibir determinado comportamento, usar o reforço positivo para reforçar o comportamento desejado e usar o reforço negativo para reduzir a frequência do comportamento não desejado, além de apresentar o resultado em seções breves, o aluno pode ser testado ao fim de cada seção e apresenta feedback imediato [Martins 2002].

[Martins 2002] Define que na educação, o comportamentalismo está associado ao trabalho de Skinner, que está focado no comportamento voluntário, deliberado e observável, que ele acreditava ser a maior parte do repertório comportamental de indivíduo. No behaviorismo, aprendizagem é igual a exibir comportamento apropriado. Neste enfoque, a atividade de aprendizagem é planejada de modo a serem ensejadas situações em que o estudante evidencie comportamentos desejados.

Assim como as outras ferramentas, com o paradigma de programação visual citadas neste trabalho, o Blockly OpenCL tem o mesmo potencial pedagógico para apoiar o ensino-aprendizagem de uma linguagem de programação, observando que nesse aspecto, todas elas têm as mesmas características, que possibilitam jovens e adultos a programarem de forma mais simples e ágil, sem a necessidade de conhecer a sintaxe especifica de cada linguagem e obter o mesmo resultado. Vale ressaltar que, mesmo possibilitando o desenvolvimento de programas complexos em OpenCL na ferramenta Blockly OpenCL, o usuário necessitará buscar aprender mais sobre a linguagem e o paradigma de programação paralela.

Foge ao objetivo desse trabalho que o usuário aprenda a programar em OpenCL apenas com o uso da ferramenta, ou seja, O Blockly OpenCL, deve ajudar iniciantes com os estudos facilitando testes no desenvolvimento de uma aplicação em OpenCL de maneira ágil, possibilitando verificar o código gerado e implementar pequenas mudanças afim de verificar o resultado, ou para usuários mais experientes desenvolverem suas aplicações em um menor tempo, tendo em vista que ao arrastar um bloco é possível produzir de uma só vez, várias linhas de código, outra proposta de aplicação da ferramenta é possibilitar pesquisadores de outras áreas de conhecimento que não a ciência da computação, mas que possuam um conhecimento básico em programação, a produzir suas aplicações de maneira paralela, verificando apenas os exemplos e ou tutoriais, além das recomendações de leitura presentes na página web da ferramenta.

3. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos que abordam a facilidade de ensino de programação para crianças e adultos, novatos ou não, utilizando o paradigma de programação visual. Foi encontrado também uma Api para facilitar o desenvolvimento de aplicações em OpenCL com uma linguagem de alto nível.

De acordo com [Mezős 2016], o NoooCL é uma API baseada em node que usa o paralelismo do OpenCL em uma linguagem de alto nível, que é o Javascript e vem facilitar a programação em OpenCL para usuários que prefiram ou necessitem usar uma aplicação em node ao invés de C ou C++. Segundo o site [Nodebr 2016], Node.js é uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos. O NoooCL

tem semelhanças com o Blockly OpenCL, pois ele pretende facilitar o desenvolvimento de aplicações paralelas, com a utilização de uma linguagem de alto nível, a diferença é que o Blockly usa blocos para gerar os códigos em C++ e CL e o NoooCL usa bibliotecas para a criação do programa em Javascript.

Foi encontrado apenas a ferramenta OpenBlocks que gera códigos de forma semelhante ao Google Blockly. O OpenBlocks é um ambiente de desenvolvimento baseado em blocos que disponibiliza uma API para desenvolver novos blocos para determinada linguagem [Roque 2007]. O OpenBlocks É uma ferramenta semelhante ao Blockly, mudando apenas a linguagem a ser desenvolvida os blocos e a aparência, enquanto o Blockly usa Javascript para desenvolver os blocos, o OpenBlocks utiliza Java. O Blockly foi escolhido devido a facilidade de criar os blocos com a ferramenta BlockFactory, a comunidade ativa e por já existir um ambiente que serve como exemplo e se adequa as necessidades do ambiente planejado para o Blockly OpenCL.

Algumas das ferramentas encontradas se compara ao Blockly OpenCL, ao ter como objetivo auxiliar no ensino-aprendizagem de determinada linguagem de programação com o uso de Linguagens Visuais de Programação.O BlueJ é uma ferramenta para o auxílio ao ensino aprendizagem de programação orientada a objetos e utiliza o paradigma de programação visual sem ser em blocos. Apesar de possuir um formato mais simples que o Blockly, o BlueJ possui mais opções de conexão devido as características da orientação a objetos. A ligação dos elementos visuais é feita através de setas semelhantes ao padrão de UML*(Unified Modeling Language)* e não diretamente como no padrão de blocos. [Van Haaster and Hagan 2004].

Outra ferramenta para ensino de orientação a objetos é o Grace, uma ferramenta de desenvolvimento com programação visual que fornece blocos para programação orientada a objetos. Oferece funcionalidades a mais, mostrando a relação dos objetos utilizados, comumente empregados em orientação a objetos, e também uma animação que deixa mais evidente a que bloco cada linha ou grupo de códigos pertencem [Homer and Noble 2014].

Seguindo o mesmo padrão, foi a ferramenta Block-C é um ambiente de programação visual com blocos que exporta a os blocos montados de maneira lógica para a linguagem C. O Block-C foi desenvolvido no intuito de facilitar a aprendizagem da linguagem C para novatos. O Block-C tem muitas semelhanças com o Blockly OpenCL, pois tem o intuito de apoiar aprendizagem de uma linguagem de programação através de uma ferramenta de desenvolvimento através de blocos, mas diferente do Blockly OpenCL o Block-C foi feito com o OpenBlocks.

Já o Snap!, é uma ferramenta web de desenvolvimento baseada em blocos, onde é possível criar os blocos e usa-los no ambiente de acordo com o seu contexto e exportar como um programa executável. O Snap! não se enquadra para ser utilizado no Blockly OpenCL, por não poder exportar o código para outra linguagem.

Com um formato diferente dos citados a cima o Flowgorithm é um programa de linguagem visual baseada em fluxograma, que gera códigos para algumas linguagens. Além de não possibilitar a criação de novos elementos para uma nova linguagem, o Flowgorithm utiliza um padrão baseado em fluxograma que torna mais difícil a compreensão da conexão entre os elementos.

4. Contribuições

Este trabalho tem como objetivo contribuir na produção de um ambiente de desenvolvimento onde é possível criar aplicações utilizando o paradigma de programação paralela em uma linguagem visual e exportá-las para a linguagem OpenCL C/C++ que é apenas uma parte do trabalho em desenvolvimento pelo mestrando e com orientação do Prof. Doutor na Universidade Federal da Paraíba.

Dentre as contribuições estão o desenvolvimento de blocos utilizando o framework Google Blockly, que servirão para implementar os programas em linguagem visual e exportá-los para a linguagem OpenCL C++. A construção dos blocos se dará objetivando o desenvolvimento de aplicações com paradigma de programação paralela, como soma e multiplicação de matrizes.

Foi desenvolvida uma página web, que serve como documentação, contendo exemplos de como implementar aplicações de soma e multiplicação de matrizes com a linguagem visual, tutoriais para utilizar o ambiente e uma breve contextualização sobre OpenCL e o paradigma de programação paralela, para que pessoas com níveis básicos de conhecimento em OpenCL, possa compreender a relação entre a linguagem visual com blocos e a linguagem em formato textual em OpenCL.

5. Metodologia

Este trabalho será desenvolvido nas etapas descritas a seguir: Preparação do ambiente para desenvolvimento OpenCL, Desenvolvimento dos Blocos para o Framework Blockly, Descrição do Block Factory, Descrição da Ferramenta Blockly OpenCL, Desenvolvimento da página web, Descrição do exemplo principal e a validação do código gerado com exemplos disponíveis.

Segundo [Falkembach 2005] um software educativo deve seguir alguns passos durante seu desenvolvimento, tais como: Análise e Planejamento, Modelagem, Implementação, Avaliação e Manutenção e Distribuição.

- **Análise e Planejamento**: Nesta etapa é definido alguns passos iniciais durante o processo de desenvolvimento de um software educativo. são eles:
    1. **solução do problema**: foi realizada uma pesquisa para encontrar ferramentas que já abordassem o tema para auxiliar o ensino aprendizagem de programação paralela com OpenCL. visto que não havia nenhuma ferramenta que se encaixasse com o nosso objetivo, buscamos identificar alguma forma de solucionar o problema. a solução encontrada foi a de deixar o OpenCL com alto nível de abstração. Assim, foi realizada uma pesquisa de ferramentas que tornasse isso possível, encontramos então a possibilidade de utilizar uma linguagem visual que convertesse elementos visuais em código OpenCL. Foram encontradas duas ferramentas que possibilitam utilizar elementos visuais e exportá-los para uma linguagem procedural, O Google blockly e o OpenBlocks.
    2. **Definições**: Escolhemos o Google Blockly por apresentar uma documentação bem definida, uma comunidade ativa e também pela aparência dos elementos, além de já disponibilizar uma ferramenta para a construção dos blocos e também um ambiente modelo que se enquadram

melhor na utilização do OpenL C e por ser uma ferramenta *web-based* que não precisará ser feito o *download* ou instalação. percebemos ainda que apenas utilizar o ambiente de desenvolvimento não seria o suficiente para que o aluno ou usuário pudesse compreender o paralelismo e as funções do OpenCL, por isso decidimos que seria necessário uma página com os tutoriais sobre o OpenCL e o ambiente de programação a ser desenvolvido.

3. **Público Alvo**: O público alvo são, desde desenvolvedores iniciantes a avançados que possuam a necessidade de utilizar o paradigma de programação paralela a pesquisadores de outras áreas de conhecimento que precisem utilizar programação paralela em suas pesquisas.

- **Modelagem**: a modelagem e dividida em tres etapas:

    1. **Modelo conceitual**: Inicialmente foi definido como modelo, utilizar uma página wiki e o ambiente desenvolvido no Google Blockly, mas após definir a estrutura dos conteúdos, observamos que utilizar uma página web que concentrasse tudo, tanto o ambiente quanto a página com os tutoriais seria o mais adequado e possibilitaria uma melhor contextualização, com um layout mais agradável e fácil de navegar.
    2. **Modelo de Navegação**: Para facilitar o carregamento do conteúdo da página web foi decidido utilizar o *layout* de páginas web baseado em uma única página, que contenha um menu com *hyperlinks* para facilitar a navegação e encontrar o conteúdo desejado. Então foi decidido que haveriam uma página inicial com a contextualização da ferramenta, outra com os tutoriais e a página com o ambeite de programação baseada em blocos.
    3. **Modelo de Interface**: A interface de todo o ambiente, utilizou o framework Bootstrap que fornece bibliotecas CSS e Javascipt que deixam as páginas html com uma interface mais fluida e responsiva, com isso pesquisamos um template que mais se adequasse ao conceito que queriamos representar, então utilizamos dois templates, um para a página inicial e outro para os tutoriais.

- **Implementação:** A implementação ocorreu em três etapas descritas nas subseções seguintes utilizando a metodologia de desenvolvimento ágil Scrum por se adequar melhor a um projeto com equipes pequenas e que precisa de resultados a cada iteração e com modificações nas estratégias no desenvolvimento dos blocos, conforme evidencia [Rising and Janoff 2000] em que os ciclos de desenvolvimento foram divididos em semanas a cada semana eram realizados os *sprints* separando os *sprints backlogs* a partir do *Guidecard reference* do OpenCL que serviu como *product backlog*.
- **Avaliação e Manutenção**: A avaliação do Ambiente de programação Blockly OpenCL ocorreu, pela verificação do código gerado pelos blocos, ao serem colocados de maneira lógica para seguir determinada função, o código gerado deveria ser similar ao código base que foi uilizado para gerá-lo. houve testes para verificar se as alterações feitas nos blocos estavam sendo realizadas no código gerado.
- **Distribuição**: O ambiente Blockly OpenCL está disponível através do link `http://www.ccae.ufpb.br/pibid/blocklyopencl/`

5.1. Desenvolvimento dos blocos para o framework Blockly em OpenCL C/C++

Para o desenvolvimento dos blocos na API Blockly foram necessários uma breve observação da documentação e de exemplos, também foi utilizado o Block Factory para a construção dos blocos, onde é possível selecionar o formato do bloco, as entradas do usuário, os possíveis encaixes e digitar o código em OpenCL C++ que será exportado em código Javascript para ser incorporado ao ambiente.

5.2. Block Factory

O Block Factory é uma ferramenta disponibilizada pela API Google Blockly. Que oferece uma interface amigável disponibilizando a criação de blocos com o mesmo paradigma, arrastando blocos. Com o Block Factory é possível criar todos os tipos de blocos que se possa necessitar, com ela é possível definir cores, formas tipos de encaixes, restrições de conexão por tipos, entrada do usuário e textos, possibilitando exportar o bloco para a ferramenta através de dois códigos, o primeiro é o *Language Code* que define da aparência do bloco e o outro *Generator Stub* que define o conteúdo com os códigos que serão gerados na linguagem indicada, que no nosso caso é a OpenCL, como pode ser visto na Figura 3.

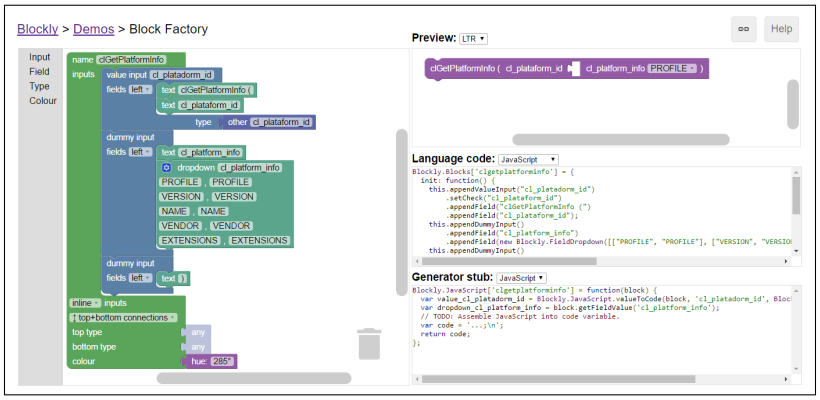

Figura 3. Interface Google Block Factory

5.3. A produção dos Blocos

Os blocos foram produzidos seguindo a documentação OpenCL API 1.1 Quick Reference, ou seja, os blocos foram produzidos para gerar códigos em OpenCL seguindo tais parâmetros de entrada e de restrições. Devido a uma questão de compatibilidade com os dispositivos, foi escolhido gerar os códigos em OpenCL na versão 1.1 que segue o padrão ISO C99 e com a linguagem C++. fazendo com que o código gerado a partir da ferramenta seja compatível com um maior número de dispositivos, já que as versões mais recentes do OpenCL funciona apenas com as plataformas mais recentes.

A interface de desenvolvimento Blockly OpenCL tem em sua estrutura um menu com os blocos divididos em categorias, e abas; na primeira aba é possível encontrar os blocos que foram arrastados e montados, é possível encontrar botões com opções de controle para o zoom e também o botão de apagar o projeto. na segunda aba é possível encontrar o código gerado em OpenCL no formato CPP que define o ”host”e na terceira aba é possível encontrar os códigos do *Kernel* que são as funções que serão executadas nos dispositivos

de processamento paralelo, CPUs, e GPUs, na quarta e última aba é possível observar os códigos em Xml, que possibilita salvar o projeto dos blocos, assim é necessário apenas copiar e salvar o código Xml, e recolocá-lo no ambiente para recomeçar de onde parou.

Foi adicionado a ferramenta uma opção de segurança, para que ao tentar fechar a aba do navegador em que está aberta o Blockly OpenCL aparece uma janela alertando o usuário do perigo de fechar a aba sem salvar o projeto, seja exportando os códigos nas abas da linguagem ou o xml que definem os blocos utilizados na área de desenvolvimento conforme apresentado na Figura 4.

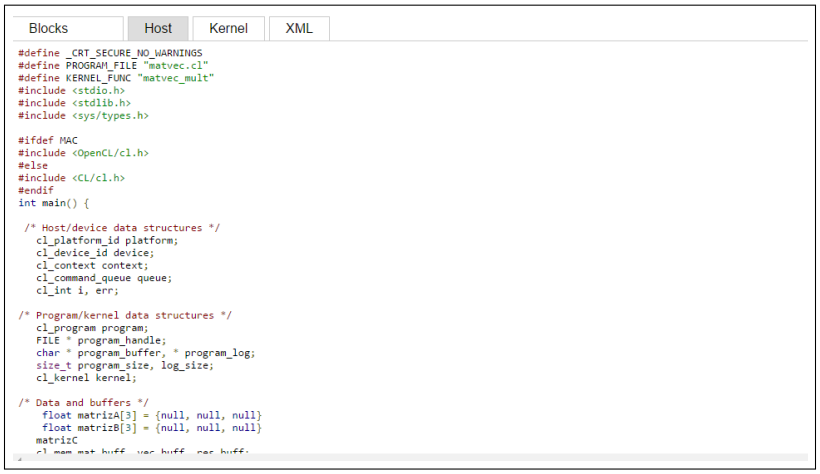

Figura 4. Interface Blockly OpenCL com um exemplo de código para exportar

5.4. Desenvolvimento da página Web

Foi desenvolvido uma página web, com as informações sobre a ferramenta, exemplos disponíveis comparando a linguagem visual com a linguagem procedural, tutorial da preparação do ambiente, contextualização da programação paralela e do OpenCL e auxiliar as pessoas que queiram aprender a programar com o paradigma de programação paralela". O desenvolvimento ocorreu utilizando a Visual Studio, que serve para facilitar o desenvolvimento de páginas web, foi utilizado um template responsivo em Bootstrap, o template é disponibilizado para download gratuitamente através do link https://github.com/BlackrockDigital/startbootstrap-freelancer

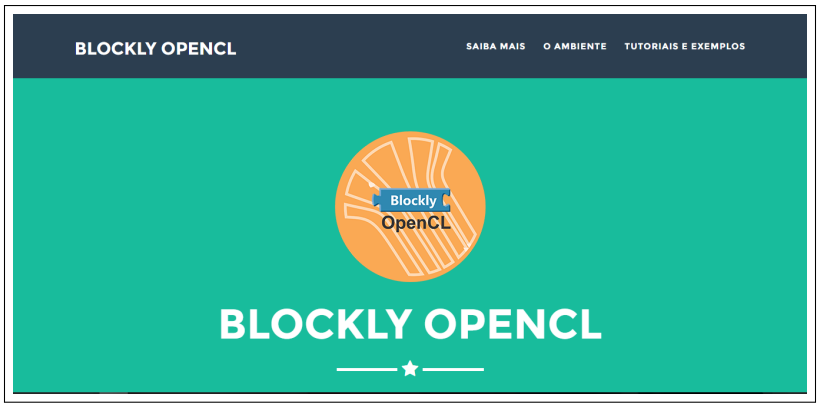

Figura 5. Interface da página Web Blockly OpenCL

O site serve como base para iniciar estudos sobre o OpenCL, pois além dos tutoriais, a página apresenta um conjunto de informações e sugestões de leitura sobre o tema, ou seja, o site se torna um ponto de partida para as pessoas que queiram aprender a programar em OpenCL. O site tem o propósito de auxiliar com o aprendizado na utilização do ambiente de programação Blockly OpenCL, detalhando passo a passo o uso dos blocos relacionando o código gerado tanto para o *host*, quanto para o kernel, possibilitando ao usuário perceber como funciona o paralelismo no código gerado através dos blocos, na Figura 5 é possível observar a página inicial da página desenvolvida.

5.5. Conjunto de blocos que formam um programa com paradigma de programação paralela

Nesta seção será descrito em detalhes o bloco principal feito para exemplo de um programa com o paradigma de programação paralela em OpenCL, é um bloco que soma vetores e matrizes de forma paralela, que se adequa muito bem ao uso de paralelismo para exemplificar de forma didática a conexão entre os blocos, o *Host* e o *Kernel* que nesse contexto, com o arrastar de apenas 10 blocos é possível criar um programa completo, como podemos ver na Figura 6 e descrito detalhadamente nos itens seguintes.

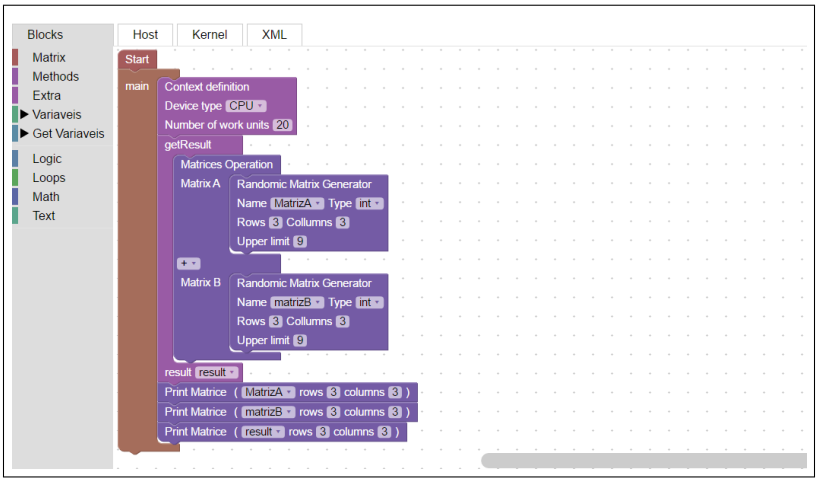

Figura 6. Exemplo dos blocos de soma de matrizes ou vetores

Ao arrastar o bloco na categoria *"matrix"* é possível encontrar os blocos, listados a seguir:

- O *"start"* que gera os códigos no host de definições de arquivos com os cabeçalhos comuns em programas C e c++;
- O bloco *"main"*, possui a definição do método *"main"* que gera o código de execução principal do host e na aba kernel é definido a função do kernel de soma de metrizes passando como atributos as matrizes a serem somadas e a matriz que guardará os resultados;
- *"Context definition"*, gera os códigos referentes a criação do contexto passando as informações dos buffers e acesso ao dispositivos.
- *"get result"*, atribui o resultado gerado pela soma das matrizes a uma variável chamada result;
- O bloco *"Matrices Operation"* gera o contexto da aplicação e seleção dos devices e recebe dois parametros que são as matrizes ou vetores que serão somados e no kernel e definido como é feita a soma através do endereçamentos dos work-itens.

- O bloco de definição de ”Random matrix generator”, denominados “MatrizA” e “MatrizB”. Ao encaixar os blocos do tipo matrizes no espaço indicado as informações contidas nos blocos serão passadas ao bloco Operação com matrizes, assim será possível informar o tipo dos matrizes, o tamanho e dimensões, o nome das variáveis e os valores atribuídos as variáveis mencionadas. No kernel é feita a chamada da função gerador de matrizes passando os atributos informados anteriormente;
- O bloco *”print matrices”* captura a variável a qual foi alocada as matrizes para ser impressas.

Ao definir essas informações o usuário poderá clicar nas abas *“Host”*, que serve como a definição do host na linguagem C++ e a aba *“Kernel”* na linguagem OpenCL e que tem a extensão CL, para obter os respectivos códigos, que já possuirá declarações e métodos padrões para execução de um código em OpenCL, como capturar informações dos dispositivos e plataformas, criação do buffer, do contexto e da execução do kernel, para exportá-los para a IDE e executar o programa de soma ou multiplicação de matrizes ou vetores.

Ao Arrastar o bloco ”main”é gerado na aba *”Kernel”* a função do *”Kernel”* atribuida ao bloco, como é possivel ver no campo de código a baixo:

```
__kernel void main(__global int* matrizA,
                   __global int* matrizB,
                   __global int* matrizC) {

   int valor;
   int gid = get_global_id(0);
   unsigned long int seed = (unsigned long int)( (gid + 1111) *
      (gid + 1011) * (gid + 1010)) ;
}
```

Onde é definido com a palavra reservada `"__Kernel"`, o nome do kernel, e os atributos entre parênteses indicando que são do tipo de memoria `"__global"`, dentro da função é possível capturar o *id* através da variável ”gid”. Depois é possível realizar a soma de acordo com o endereço da matriz, que será o mesmo do ID, assim a instrução irá ser replicada para cada work-item e irá somar os dados definidos em cada endereço a partir do endereço zero. Assim será gerada uma nova matriz `matrizC` com o resultado da soma.

No código de soma de matrizes de forma procedural, vemos que a complexidade e o número de passos é muito maior, pois diferente do paralelismo em que todas as somas são feitas em uma única iteração, na linguagem procedural o código irá executar o número de iterações de acordo com o tamanho da matriz.

No código a baixo podemos ver o código que converte os tipos de estruturas utilizadas para o formato de buffer em que o kernel trabalha, apontando a variável mat_buff, vec_buff e res_buff sendo passado através do método clCreateBuffer.

```
   mat_buff = clCreateBuffer(context, CL_MEM_READ_ONLY |
       CL_MEM_COPY_HOST_PTR, sizeof(float) *3, matrizA, &err);
   if (err < 0) {
```

```
        perror("Couldnt create a buffer object");
        exit(1);
    }

    vec_buff = clCreateBuffer(context, CL_MEM_READ_ONLY |
        CL_MEM_COPY_HOST_PTR, sizeof(float) *3, matrizB, &err);
    if (err < 0) {
        perror("Couldnt create a buffer object");
        exit(1);
    }

    res_buff = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
        sizeof(float) * 3, matrizC, &err);
    if (err < 0) {
        perror("Couldnt create a buffer object");
        exit(1);
    }
```

O código a baixo é passado os buffers para o kernel com as variáveis já convertidas.

```
/* Create kernel arguments from the CL buffers */
    err = clSetKernelArg(kernel, 0, sizeof(cl_mem), & mat_buff);
    if (err < 0) {
        perror("Couldnt set the kernel argument");
        exit(1);
    }

    err = clSetKernelArg(kernel, 1, sizeof(cl_mem), & vec_buff);
    if (err < 0) {
        perror("Couldnt set the kernel argument");
        exit(1);
    }

    err = clSetKernelArg(kernel, 2, sizeof(cl_mem), & res_buff);
    if (err < 0) {
        perror("Couldnt set the kernel argument");
        exit(1);
    }
```

No código a baixo é criada a fila de execução para o *device*.

```
/* Create a CL command queue for the device*/
    queue = clCreateCommandQueue(context, device, 0, & err);
    if (err < 0) {
        perror("Couldnt create the command queue");
        exit(1);    }

    work_units_per_kernel = 3; /* work-units per kernel */
    err = clEnqueueNDRangeKernel(queue, kernel, 1, NULL, &
       work_units_per_kernel,
```

```
        NULL, 0, NULL, NULL);
    if (err < 0) {
        perror("Couldnt enqueue the kernel execution command");
        exit(1);
    }
```

Através do código abaixo é possível obter o resultado da soma feita no *device*.

```
 /* Read the result */
    err = clEnqueueReadBuffer(queue, res_buff, CL_TRUE, 0, sizeof
       (float) *3,
     matrizC, 0, NULL, NULL);
    if (err < 0) {
        perror("Couldnt enqueue the read buffer command");
        exit(1);
    }
```

Ao fim da disposição dos blocos como mostrado na imagem acima é possível observar na aba kernel o código mostrado a seguir:

```
/*Save this file with name "matvec.cl" in the same folder wich
   the host file*/
__kernel void main(__global int* matrizA,
                   __global int* matrizB,
                   __global int* matrizC) {

    int valor;
    int gid = get_global_id(0);
    unsigned long int seed = (unsigned long int)( (gid + 1111) *
       (gid + 1011) * (gid + 1010)) ;

       valor = genRand(&seed, 9);
       MatrizA[gid] = valor;

       valor = genRand(&seed, 9);
       matrizB[gid] = valor;

     matrizC[gid] = matrizA[gid] + matrizB[gid];
}
    int genRand(unsigned long int* i, int limite){
        (*i) ^= (*i) >> 12; // a
        (*i) ^= (*i) << 25; // b
        (*i) ^= (*i) >> 27; // c
        int ret = ( (*i) * 2685821657736338717) % limite;
        if (ret < 0) {
             ret *= -1;
        }
        return ret;
        }
```

O paralelismo é utilizado em dois momentos nesse *Kernel* mostrado acima, ini-

cialmente na soma de matrizes e depois na geração das matrizes. Na soma de matrizes podemos na linha 16 onde é definido a matrizC[Gid] = matrizA[Gid] + matrizB[Gid], em que GID é a definição dos endereços dos workitens em que cada posição dos work-itens receberá essa mesma função e fará o cálculo de acordo com seu endereço.

e como podemos ver no código a seguir o mesmo código é realizado utilizando a linguagem pseudocódigo de forma estrutural e tem um desempenho muito inferior para matrizes com um tamanho grande devido ao número de iterações necessárias para a realização desses cálculos.

```
1 int matrizA[100][100], matrizB[100][100], matrizC[100][100],i,j
2
3 for (i=0;i<100;i++) {
4         for(j=0;j<100;j++){
5                 matrizC[i][j] = MatrizA[i][j] + MatrizB[i][j]);
6         }
7 }
```

6. Resultados

Como resultado desse estudo, foi desenvolvido um ambiente de produção de código em OpenCL com a linguagem visual Blockly. Nela foram criados blocos que, ao serem manipulados de forma lógica, geram códigos em OpenCL, de maneira procedural, para posteriormente serem exportados para uma IDE e executados no computador do usuário.

O Blockly OpenCL foi desenvolvido com intuito de apoiar o ensino-aprendizagem para desenvolvimento de programas com o paradigma de programação paralela. Foram utilizadas as Flags do OpenCL, para usuários mais avançados, e blocos de programas simples prontos para executar, sendo necessários apenas alguns ajustes para que o usuário com menos conhecimento em OpenCL possa utilizar e verificar o código gerado.

Para apoiar a utilização da ferramenta foi produzido um site. Nele são apresentadas informações sobre o projeto e tutoriais, como o de instalação do SDK, nos sistemas operacionais Windows e Linux, e de produção de novos blocos para a ferramenta Blockly OpenCL. Também é apresentada no site uma seção com exemplos detalhados, onde é comentado o exemplo em OpenCL em paralelo com o bloco feito no ambiente de desenvolvimento Blockly OpenCL.

7. Considerações Finais

A programação paralela é um paradigma que consiste em computar dados de forma simultânea. Sua utilização pode trazer benefícios relacionados a custos e frequência de processamento. O OpenCL é uma linguagem de programação que faz uso desse paradigma. Ele apresenta um grande potencial no processamento paralelo com volumes elevados de dados com problemas algébricos. Com isso, sua utilização pela academia, apesar de relativamente recente, vêm sendo bem aceita por pesquisadores. Entretanto, devido à complexidade, utiliza-lo ainda apresenta desafios.

Com intuito de apoiar o ensino-aprendizagem de programação paralela com OpenCL, este trabalho apresentou o software educacional Blockly OpenCL, um ambiente de programação visual que permite ao usuário criar aplicações com o paradigma de

programação paralela em OpenCL e exportar o código gerado para a linguagem OpenCL C de maneira ágil e intuitiva.

Ainda com esse intuito foi desenvolvido um site, que pode ser utilizado com material independente ou complementar a ferramenta. Nele são apresentados tutoriais e o próprio ambiente de desenvolvimento. Esse material tem a finalidade de possibilitar ao iniciante em desenvolvimento OpenCL aprender a preparar o ambiente de desenvolvimento em OpenCL, que compreende: a instalação do SDK do OpenCL, do compilador C e da IDE Eclipse. Além disso, são recomentados materiais da literatura sobre o tema em questão.

8. Trabalhos Futuros

Como trabalhos futuros, vemos a possibilidade de executar a aplicação diretamente no ambiente de desenvolvimento do Blockly OpenCL, através de um *console*, para que possa ser executado diretamente no servidor e não seja necessário fazer o download dos códigos gerados para executá-lo, realizar testes da aplicação com alunos e pesquisadores que necessitem utilizar programação paralela, com o intuito de analisar o nível de intuitividade e agilidade ao desenvolver uma aplicação utilizando o Blockly OpenCL, para assim saber se a ferramenta realmente contribui no ensino-aprendizagem de programação paralela com OpenCL.

Referências

[Banger and Bhattacharyya 2013] Banger, R. and Bhattacharyya, K. (2013). *OpenCL Programming by Example*. Packt Publishing.

[Brock 2014] Brock, K. (2014). Composing accessible code. *Computers and Writing*.

[CodePlex 2016] CodePlex (2016). Opencl tutorials. http://opencl.codeplex.com/wikipage?title=OpenCL%20Tutorials%20-%201. Acessado junho 11, 2016.

[Crawford et al. 2016] Crawford, C. S., Andujar, M., Jackson, F., Applyrs, I., and Gilbert, J. E. (2016). Using a visual programing language to interact with visualizations of electroencephalogram signals. *ASEE-SE Annual Meeting*.

[da Silveira Júnior et al. 2015] da Silveira Júnior, G., Diniz Rossi, F., Lincoln Ramires Izolan, P., and Renan da Silva Almeida, J. (2015). Análise da ferramenta de programação visual blockly como recurso educacional no ensino de programação. *III Seminário Argentina-Brasil de Tecnologias da Informação e da comunicação*.

[Demidov et al. 2013] Demidov, D., Ahnert, K., Rupp, K., and Gottschling, P. (2013). Programming cuda and opencl: A case study using modern c++ libraries. *SIAM Journal on Scientific Computing*, 35(5):C453–C472.

[Falkembach 2005] Falkembach, G. A. M. (2005). Concepção e desenvolvimento de material educativo digital. *RENOTE*, 3(1).

[Foldoc 2016] Foldoc (2016). Visual programming language. https://en.wikipedia.org/wiki/visual_programming_language. Acesso em 05 de abril, 2016.

[Gaster et al. 2012] Gaster, B., Howes, L., Kaeli, D. R., Mistry, P., and Schaa, D. (2012). *Heterogeneous Computing with OpenCL: Revised OpenCL 1*. Newnes.

[Google Developers 2016] Google Developers (2016). Blockly instalation. https://developers.google.com/blockly/installation/overview. Acesso em 21 de março, 2016.

[Homer and Noble 2014] Homer, M. and Noble, J. (2014). Combining tiled and textual views of code. In *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*, pages 1–10. IEEE.

[Khronos Group et al. 2011] Khronos Group, O. W. G. et al. (2011). The opencl specification. *version 1.1*, 1(44):385.

[Kirk and Hwu 2010] Kirk, D. and Hwu, W. (2010). *Programming Massively Parallel Processors: A Hands-on Approach*. Applications of GPU Computing Series. Elsevier Science.

[Kowalik and Puźniakowski 2012] Kowalik, J. and Puźniakowski, T. (2012). *Using OpenCL: Programming Massively Parallel Computers*. Advances in parallel computing. IOS Press.

[Martins 2002] Martins, K. L. (2002). Teorias de aprendizagem e avaliação de software educativo. *Monografia) Especialização em Informática Educativa–Universidade Federal do Ceará*.

[Mélo et al. 2011] Mélo, F., CUNHA, R., SCOLARO, D., and CAMPOS, J. (2011). Do scratch ao arduino: Uma proposta para o ensinointrodutório de programação para cursos superiores de tecnologia. In *XXXIX Congresso Brasileiro de Educação em Engenharia, Blumenau, Brasil*.

[Mezős 2016] Mezős, G. (2016). Nooocl npm. https://github.com/unbornchikken/NOOOCL. Acesso em 30 de abril, 2016.

[MIT 2016] MIT (2016). Mit app inventor. http://appinventor.mit.edu/explore/. Acesso em 13 de junho de 2016.

[MIT Scratch 2016] MIT Scratch (2016). Mit app inventor. https://scratch.mit.edu/. Acesso em 13 de junho de 2016.

[Mitchell et al. 2005] Mitchell, T. J., Chen, S. Y., and Macredie, R. D. (2005). Hypermedia learning and prior knowledge: domain expertise vs. system expertise. *Journal of Computer Assisted Learning*, 21(1):53–64.

[Nodebr 2016] Nodebr (2016). O que é node.js. http://nodebr.com/. Acesso em 11 de junho, 2016.

[Nvidia 2016] Nvidia (2016). Cuda parallel computing platform. http://www.nvidia.com.br/object/cuda_home_new_br.html. Acessado junho 11, 2016.

[OpenMP.org 2016] OpenMP.org (2016). Openmp. https://pt.wikipedia.org/wiki/OpenMP/. Acesso em 11 de junho, 2016.

[Rising and Janoff 2000] Rising, L. and Janoff, N. S. (2000). The scrum software development process for small teams. *IEEE software*, 17(4):26.

[Roque 2007] Roque, R. V. (2007). *OpenBlocks: an extendable framework for graphical block programming systems*. PhD thesis, Massachusetts Institute of Technology.

[Scarpino 2012] Scarpino, M. (2012). *OpenCL in Action: How to Accelerate Graphics and Computation*.

[Shen et al. 2012] Shen, J., Fang, J., Sips, H., and Varbanescu, A. L. (2012). Performance gaps between openmp and opencl for multi-core cpus. In *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, pages 116–125. IEEE.

[Silveira et al. 2010] Silveira, C. L., da Silveira Jr, L. G., and Cavalheiro, G. G. H. (2010). Programação em opencl: Uma introdução prática.

[Soffa and Alcântara 2008] Soffa, M. M. and Alcântara, P. R. d. C. (2008). O uso do software educativo: reflexões da prática docente na sala informatizada. *Acesso em*, 22.

[Tupinambá 2013] Tupinambá, A. (2013). Programação em gpu utilizando opencl. pages 1–11.

[Van Haaster and Hagan 2004] Van Haaster, K. and Hagan, D. (2004). Teaching and learning with bluej: An evaluation of a pedagogical tool. In *Information Science+ Information Technology Education Joint Conference, Rockhampton, QLD, Australia*, pages 455–470.

[Vieira 1999] Vieira, F. M. S. (1999). Avaliação de software educativo: reflexões para uma análise criteriosa. *http://www. edutecnet. com. br/edmagali2. htm.¿ Acessado em*, 5(11):06.