

Artigo pro Cuda Fuzzy

Edevaldo Braga dos Santos¹, Giovane de Oliveira Torres¹, Guilherme Pereira Paim¹,
Renan Zafalon da Silva¹, Vitor Alano de Ataides¹, Maurício Lima Pilla¹

¹Universidade Federal de Pelotas
Pelotas, RS - Brasil

{edevaldo.santos,gdotorres,gppaim,renan.zafalon,vaataides,pilla}@inf.ufpel.edu.br

Abstract. *Abstract aqui.*

Resumo. *Resumo aqui.*

1. Introdução

Existem diversos casos onde classes de objetos não pertencem totalmente a um conjunto. Baseado nisto, Zadeh definiu a teoria dos conjuntos *fuzzy* [Zadeh 1965], o que visa tratar problemas de imprecisão ao classificar dados no mundo real. Os conjuntos *fuzzy* possuem aplicações em sistemas de controle e de suporte à decisão, onde a descrição do problema não é feita de forma precisa [Weber and Klein 2003].

Utilizando-se dos conjuntos *fuzzy*, tem-se a base para a lógica *fuzzy*, sendo construído a partir da lógica proposicional. Com isto, os operadores foram definidos a partir dos já estabelecidos na lógica clássica, com a adição de outros para fins práticos [Tanscheit 2004]. Uma característica interessante que diferencia a lógica tradicional da *fuzzy* é que na primeira os valores que são utilizados atendem a condição de serem verdadeiros ou falsos (0 ou 1). Já na segunda, trabalha-se com conjuntos *fuzzy* – estes podem assumir um valor que pertence ao intervalo $[0, 1]$, o que permite que um conjunto *fuzzy* possa ser representado por uma infinidade de valores [Klir and Yuan 1995].

A fim de obter-se computação com bom desempenho, é importante fazer uso dos vários núcleos de processamento os quais são disponibilizados nos sistemas de computação atuais – para poder por em prática o uso do paralelismo. Neste contexto, encaixam-se as GPUs (*Graphical Processor Units*), as quais são componentes com alto poder de paralelismo [Sengupta et al. 2007]. Porém, é importante ressaltar que as GPUs são reservadas a obter bom desempenho com aplicações que possuem determinadas características [Owens et al. 2008] que incluem: (i) Requisitos computacionais grandes, (ii) Paralelismo nas aplicações e (iii) maior importância ao *throughput* do que a latência. Destacam-se alguns exemplos práticos bem-sucedidos que utilizam CUDA: Análise do fluxo de tráfego aéreo, através do uso do poder computacional de CUDA, foi possível reduzir o tempo de análise do tráfego aéreo nacional de dez minutos para três segundos. Outro exemplo relevante é o ganho de desempenho em simulações moleculares NAMD (dinâmica molecular em nanoescala), o ganho de desempenho foi possível graças as arquiteturas paralelas das GPUs [NVIDIA 2015].

Tendo estes conceitos discutidos, o objetivo deste trabalho é descrever uma biblioteca de lógica *fuzzy* voltada para GPUs, a fim de verificar como pode ser efetuado uma implementação que consiga extrair paralelismo deste tipo de arquiteturas.

O restante deste artigo está dividido da seguinte maneira: A seção 2 fala sobre a lógica fuzzy, que é a base para a construção deste trabalho. Na seção 3, é descrito a implementação efetuada da biblioteca CudaFuzzy. A seção 4 destina-se a explicar a metodologia empregada para a execução de testes na biblioteca. A seção 5 exhibe e discute os principais resultados obtidos por este trabalho, de onde se tiram as principais conclusões, observadas na seção 6, a qual ainda mostra possíveis trabalhos futuros. Por fim, a seção 7 faz uma breve discussão sobre os trabalhos relacionados ao escopo deste artigo.

2. Lógica Fuzzy

Lógica fuzzy foi criada com base na teoria de conjuntos Fuzzy, a ideia principal de fuzzy é a seguinte: Um valor é verdadeiro (grau 1) ou falsos (grau 0), sendo que o grau pode variar de 0 à 1. A teoria do conjunto fuzzy foi inventada com o objetivo de oferecer ferramentas matemáticas para solucionar problemas imprecisos ou vagos. Dentre as inúmeras aplicações com lógica fuzzy, o projeto IMMO-RATE, é bastante interessante, pois ele permite análise de sustentabilidade em imóveis, considerando questões chaves específicas que utilizam a lógica fuzzy.

2.1. Operadores Fuzzy

Conjuntos são definidos por uma condição específica que define se um conjunto pertence ou não a um conjunto. Para exemplificar os operadores em fuzzy logo abaixo foram utilizados conjuntos A e B. Os operadores mais comuns em conjuntos fuzzy são: união ($A \cup B = x \mid x \in A \vee x \in B$) e intersecção ($A \cap B = x \mid x \in A \wedge x \in B$).

3. CudaFuzzy

CudaFuzzy é o nome dado a biblioteca de lógica fuzzy desenvolvida neste trabalho, voltada à GPU. Para o desenvolvimento, a biblioteca criada necessitou ser composta de vários operadores para trabalhar sobre os conjuntos fuzzy. Foram escolhidas operações básicas [Klir and Yuan 1995] para implementação, sendo estas 2 operadores AND (\wedge), 2 operadores OR (\vee) e 3 operadores NOT (\sim). A Figura 1 mostram os operadores implementados.

$$\begin{aligned} AND1(x, y) &= \min(x, y) \\ AND2(x, y) &= x * y \\ OR1(x, y) &= \max(x, y) \\ OR2(x, y) &= (x + y) - (x * y) \\ NOT1(x) &= 1 - x \\ NOT2(x) &= \sqrt{1 - x^2} \\ NOT3(x) &= \sqrt[3]{1 - x^3} \end{aligned}$$

Figura 1. Operações implementadas na biblioteca CudaFuzzy

Referenciar cada operador mostrado? Explicar mais a operação? Parece óbvio..

Os operadores foram implementados em C, utilizando-se da biblioteca CUDA, e foram desenvolvidos como funções que recebem como argumentos um ou dois valores

do tipo `double` (um para operações NOT, e dois para operações AND e OR), sendo estes os operandos e retornam um valor do tipo `double`, o qual é o resultado da operação. A Figura 2 exemplifica a implementação da operação NOT3.

```
inline __device__ double d_Not3(double x) {  
    return pow(1 - pow(x, 3), 1.0 / 3);  
}
```

Figura 2. Código de implementação da operação de lógica fuzzy NOT3

Porém, é importante observar que somente a implementação destes operadores por si só representa pouco do trabalho efetuado. Como a arquitetura alvo para esta biblioteca são as GPUs, precisou-se elaborar uma maneira de aproveitar o paralelismo disponível nestas arquiteturas. A estratégia abordada por este trabalho buscou agrupar conjuntos de operações de um mesmo tipo em um *array* de instruções. Com isto, é possível executar múltiplas operações de lógica *fuzzy* nos vários núcleos disponibilizados pela GPU. As operações são agrupadas em um mesmo tipo para serem executadas em paralelo devido à taxonomia destas arquiteturas, sendo elas consideradas SIMT (*Single instruction multiple thread*) [Keckler et al. 2011], o que indica que ...

Explicar
SIMT

Assim, as operações são feitas sobre uma função que leva o prefixo Bulk. Um Bulk fará uma operação de lógica *fuzzy* sobre um *array* de operandos (Em caso de operações BulkNOT) ou sobre dois *arrays* de operandos (Em caso de operações BulkAND e BulkOR). No caso de operações que trabalham com dois *arrays*, as operações serão efetuadas de maneira correspondente nos *arrays*, i.e., o primeiro operando de um *array* será combinado com o primeiro operando do outro *array*, gerando assim uma computação parcial da operação, sendo este o primeiro elemento do *array* de resultado. Logo, quando a operação requer dois *arrays* como entrada, os mesmos deverão conter o mesmo número de elementos. A Figura 3 exhibe uma computação parcial da operação BulkAND1.

```
__global__ void kernel_And(double* array, double* array2, double* result, int size) {  
  
    int idx = blockIdx.x * blockDim.x + threadIdx.x;  
  
    if(idx < size) {  
        result[idx] = d_And(array[idx], array2[idx]);  
    }  
}
```

Figura 3. Código de implementação de uma única operação BulkAND1

4. Metodologia

5. Resultados e Discussão

6. Conclusões e Trabalhos Futuros

7. Trabalhos Relacionados

Existem diversas implementações relacionadas à lógica *fuzzy*. Os artigos visualizados na bibliografia normalmente fazem utilização de lógica *fuzzy* voltada para um tipo de problema, não descrevendo uma biblioteca genérica. Como trabalhos descritos desta forma,

existe [Sugeno and Yasukawa 1993], utiliza-se lógica *fuzzy* para a discussão de um método para modelagem qualitativa. Em [Li et al. 2011], é empregada a lógica *fuzzy* para fazer o gerenciamento de energia e bateria de automóveis híbridos do tipo *plug-in*.

Como bibliotecas relacionadas à lógica *fuzzy*, existe uma desenvolvida na linguagem Java, chamada de jFuzzyLogic [Cingolani and Alcala-Fdez 2012, Cingolani and Alcalá-Fdez 2013]. Esta biblioteca é uma implementação de sistemas *fuzzy* que permite projetar controladores de lógica *fuzzy*. Por fim, existe a biblioteca FuzzyGPU [Defour and Marin 2014], a qual é o trabalho relacionado mais próximo ao que este artigo apresenta. FuzzyGPU é uma implementação de biblioteca de aritmética *fuzzy* voltada à GPUs.

Referências

- Cingolani, P. and Alcala-Fdez, J. (2012). jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In *FUZZ-IEEE*, pages 1–8. Citeseer.
- Cingolani, P. and Alcalá-Fdez, J. (2013). jfuzzylogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. *International Journal of Computational Intelligence Systems*, 6(1):61–75.
- Defour, D. and Marin, M. (2014). Fuzzygpu: a fuzzy arithmetic library for gpu. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 624–631. IEEE.
- Keckler, S. W., Dally, W. J., Khailany, B., Garland, M., and Glasco, D. (2011). Gpus and the future of parallel computing. *IEEE Micro*, (5):7–17.
- Klir, G. and Yuan, B. (1995). *Fuzzy sets and fuzzy logic*, volume 4. Prentice Hall New Jersey.
- Li, S. G., Sharkh, S., Walsh, F. C., and Zhang, C.-N. (2011). Energy and battery management of a plug-in series hybrid electric vehicle using fuzzy logic. *Vehicular Technology, IEEE Transactions on*, 60(8):3571–3585.
- NVIDIA (2015). Plataforma de computação paralela | cuda | nvidia | nvidia. Disponível em: <http://www.nvidia.com.br/object/cuda_home_new_br.html>.
- Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., and Phillips, J. C. (2008). Gpu computing. *Proceedings of the IEEE*, 96(5):879–899.
- Sengupta, S., Harris, M., Zhang, Y., and Owens, J. D. (2007). Scan primitives for gpu computing. In *Graphics hardware*, volume 2007, pages 97–106.
- Sugeno, M. and Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1):7–31.
- Tanscheit, R. (2004). Sistemas fuzzy. *Inteligência computacional: aplicadaa administração, economia e engenharia em Matlab*, pages 229–264.
- Weber, L. and Klein, P. A. T. (2003). *Aplicação da lógica fuzzy em software e hardware*. Editora da ULBRA.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3):338–353.