# FlexiblePAD: Um framework reutilizável e flexível para suporte ao desenvolvimento de aplicações paralelas

**Guilherme Esmeraldo[1], Régis Brilhante[2]**

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) – *campus* Crato
Rod. CE 292, Km 15 – 63.115-500 – Crato – CE – Brasil.

[2]Faculdade Paraíso do Ceará (FAP)
Rua da Conceição, 1229 – 63.010-465 – Juazeiro do Norte – CE – Brasil.

guilhermealvaro@ifce.edu.br, regis.brilhante@fapce.com.br

***Resumo.*** *Este trabalho apresenta um framework para desenvolvimento de aplicações paralelas, cuja implementação independe de arquitetura paralela alvo, baseia-se no padrão arquitetural MVC e pode ser reutilizado e adaptado para o desenvolvimento de diversas aplicações. Como estudo de caso, o framework foi estendido para compor uma aplicação de cálculos estatísticos, onde o usuário concentra-se no desenvolvimento do cálculo em si, desconsiderando a decomposição de domínio, bem como comunicação e mapeamento de tarefas paralelas.*

## 1. Introdução

O termo algoritmo pode ser definido como "um conjunto de processos ou regras para a solução de um problema em um número finito de passos" [IEEE 1996]. De acordo com Gebaly (2011), um algoritmo paralelo é aquele onde as tarefas podem ser realizadas simultaneamente, devido a sua independência de dados. O projeto de um uma aplicação paralela deve considerar a arquitetura paralela, onde as tarefas serão executadas.

O projeto de uma arquitetura paralela, segundo Gebaly (2011), consiste na escolha 1) do modelo de processamento, que pode ser um processador simples até um superprocessador escalar com um sistema operacional *multithread*; 2) da arquitetura de comunicação, variando desde barramentos até uma infraestrutura de rede de computadores; e 3) do subsistema de memória, podendo ser dedicado ou compartilhado. Na medida que a tecnologia evoluiu e o custo do hardware reduziu, os projetistas puderam explorar todos esses parâmetros, na tentativa de otimizar o desempenho, projetando assim uma diversidade de arquiteturas paralelas [Stallings 2012].

Já a implementação de uma aplicação paralela, de acordo com Pacheco (2011), deve consistir na divisão da aplicação entre processos/*threads*, de forma que cada tarefa tenha aproximadamente a mesma quantidade de trabalho e a comunicação seja minimizada. Em Foster (1995), é proposta uma abordagem na qual o processo de desenvolvimento de uma aplicação paralela é divido em quatro etapas: 1) identificar as tarefas que podem ser executadas em paralelo; 2) determinar os requisitos de comunicação entre essas tarefas; 3) combinar as tarefas e comunicações em tarefas maiores; e 4) atribuir as tarefas a processos ou *threads*.

Visando reduzir essa complexidade, algumas abordagens, como as apresentadas em [Hassan et al. 2011] e [Daan et al. 2009], introduzem modelos de implementação, como uma linguagem de específica de domínio (DSL) e uma biblioteca de linguagem de

alto nível, respectivamente, para aumentar a produtividade na codificação das aplicações paralelas. Além destes, existem ainda as plataformas de concorrência, como, por exemplo, [MPI 2008], que permitem a coordenação, o escalonamento e o gerenciamento de recursos em sistemas paralelos. Contudo, essas abordagens requerem, como pre-requisitos a decomposição de domínio e arquitetura alvo, além de oferecerem pouca flexibilidade.

Nesse cerne, também percebe-se que arquiteturas e algoritmos paralelos são fortemente acoplados. Um algoritmo paralelo não pode ser concebido sem a arquitetura que o suportará [Gebaly 2011], tão pouco uma arquitetura poderá ser definida sem que haja, pelo menos uma classe de aplicação pré-estabelecida [Stallings 2012].

Este artigo apresenta um framework, chamado de FlexiblePAD, para suporte ao desenvolvimento de aplicações paralelas. Este framework, que é baseado no padrão arquitetural *Model-View-Controller* (MVC) [Krausner 1988], visa ser flexível e reutilizável para implementar novas aplicações paralelas, aumentando assim a produtividade, além de ser independente de arquitetura paralela.

O framework proposto será descrito, com maiores detalhes, na próxima seção. A Seção 3 apresenta um estudo de caso, que consiste de uma aplicação de cálculos paralelos estatísticos desenvolvida a partir do uso do framework proposto, bem como alguns resultados parciais. Por fim, na Seção 4 são apresentadas as conclusões.

## 2. Framework FlexiblePAD

O framework FlexiblePAD busca flexibilizar o projeto de uma aplicação paralela através de uma arquitetura de software baseada no padrão MVC (ver Figura 1).

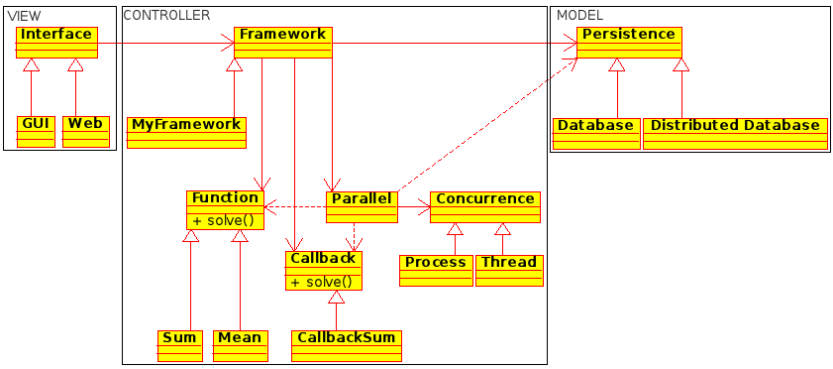


**Figura 1. Diagrama de classes do framework FlexiblePAD.**

Na Figura 1 observa-se que as camadas *View* e *Model* utilizam as classes abstratas *Interface* e *Persistence* para definir modelos de interface com o sistema e de persistência de dados, respectivamente. O uso desses modelos permite implementar tipos diferentes de interfaces gráficas e de acesso à persistência de dados. Na Figura 1, pode-se ver ainda a camada *Controller*, a qual inclui o framework (classe *Framework*), podendo ser estendido para adição de novas funcionalidades. Essa classe inclui outras três, que são: 1) *Function*, define um modelo para função paralela; 2) *Callback*, define o modelo para função de retorno; e, por fim, 3) *Parallel*, responsável por coordenar a execução das funções paralelas e da função de retorno. A classe *Concurrence* define um

modelo para o método de paralelização, através de alguma plataforma de concorrência.

Para codificação do FlexiblePad utilizou-se a linguagem de programação Python, devido ao suporte à programação orientada a objetos; *wrappers* para uso de bibliotecas compiladas em C/C++, permitido otimizar do desempenho da aplicação; e ser portável. Além disso, Python oferece uma sintaxe de alta redigibilidade, permitindo a redução do tempo de prototipação e, consequentemente, aumento de produtividade.

A próxima seção descreve como o framework FlexiblePAD foi estendido para o desenvolvimento de uma aplicação de cálculos estatísticos paralelos, bem como alguns resultados parciais.

## 3. Estudo de caso e resultados parciais

Para avaliar a abordagem proposta, o estudo de caso consistiu no desenvolvimento de uma aplicação de suporte ao desenvolvimento de cálculos estatísticos paralelos. A aplicação consiste de uma interface para adicionar, remover e selecionar conjuntos de dados, selecionar funções estatísticas paralelas e de retorno, bem como o número de processos para as funções paralelas. A interface permite ainda o desenvolvimento de novas funções estatísticas, como, por exemplo, o cálculo de média e regressão linear.

Para a implementação da camada *View*, optou-se por interface web, codificada com o framework CherryPy. A camada *Model* foi implementada localmente com o SGBD PostgreSQL e a biblioteca Psycopyg2 (interface Python para PostgreSQL). Desta forma, cada conjunto de dados poderá ser carregado no banco de dados, tornando-se acessível por todos os elementos de processamento da arquitetura paralela. Na camada *Controller*, o módulo *Concurrence* foi desenvolvido utilizando MPI4Py, que é um *wrapper* para OpenMPI (implementação código aberto em C da especificação MPI-2).

Resultados mostram que, com o uso da linguagem Python e de suas bibliotecas auxiliares, foi possível codificar toda a aplicação utilizando apenas 924 linhas. Isso mostra ser possível aumentar a produtividade no projeto de uma nova aplicação paralela, através da redução do tempo de prototipação.

Para avaliar o desempenho da aplicação, utilizou-se uma arquitetura paralela SMP, contendo um processador AMD A8 Quad Core de 1,6GHz e 6 GB de memória RAM e sistema operacional GNU/Linux. Na Tabela 1, são comparados os tempos obtidos para aplicar a função somatório, de forma sequencial e paralela, a um conjunto de dados com um milhão de números de ponto flutuante, armazenados em arquivo e em banco de dados.

**Tabela 1. Comparativo de desempenho da aplicação paralela considerando modelo de persistência e paralelismo.**

| **Métrica** | **Desempenho (em segundos)** | | | | |
|---|---|---|---|---|---|
| Tempo da função somatório sequencial (dados em arquivo). | 27,14 | | | | |
| Tempo da função somatório paralela (dados em banco de dados). | **Número de Processos** | | | | |
| | **1** | **2** | **4** | **8** | **16** |
| | 3.16 | 1,86 | 2,20 | 2,79 | 3,42 |

De acordo com a Tabela 1, percebe-se que o uso de banco de dados e paralelismo permitiu aumentar consideravelmente o desempenho da aplicação, atingindo o desempenho máximo com 2 processos. Observa-se ainda que, na medida que se aumenta o número de processos, o desempenho da aplicação sofre degradação.

## 4. Conclusões

O processo de desenvolvimento de uma aplicação paralela deve considerar muitos requisitos de projeto, como os parâmetros de configuração da arquitetura paralela alvo, além das técnicas e abordagens utilizadas para paralelização da aplicação.

Para reduzir essa complexidade, o trabalho aqui apresentado propõe um framework baseado no padrão MVC, chamado de FlexiblePAD, o qual pode ser estendido para suportar o desenvolvimento de novas aplicações paralelas. O framework mostrou-se flexível por permitir o uso de modelos de interface, persistência e concorrência diferentes, podendo ser reutilizados para compor outras aplicações em diferentes arquiteturas paralelas.

Os resultados mostraram que é possível, com o uso do FlexiblePAD, aumentar a produtividade através da redução do tempo de prototipação, por utilizar uma linguagem de programação de alto nível e com alta redigibilidade. Além disso, foi possível constatar a eficiência da aplicação desenvolvida no estudo de caso, comparando o desempenho sob diferentes condições de persistência e de paralelismo.

## Referências Bibliográficas

IEEE, Standards Coordinating Committee 10, Terms and Definitions. (1996) The IEEE Standard Dictionary of Electrical and Electronics Terms, J. Radatz, Ed. IEEE.

Pacheco, P. S. (2011), An Introduction to Parallel Programming, Morgan Kaufmann.

Gebaly, F. (2011), Argorithms and Parallel Computing. Willey and Sons.

Foster, I. (1995) Designing and Building Parallel Programs, Addison-Wesley.

Stallings, W. (2012) Computer Organization and Architecture. Prentice Hall. 9th edition.

Hassan, C., Arvind, K., Sujeeth, K. J. B., HyoukJoong, L., Anand, R. A., Kunle O. (2011) "A domain-specific approach to heterogeneous parallelism", In: Proceedings of the 16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, p35-46.

Daan, L., Wolfram, S., Sebastian, B., (2009) The Design of a Task Parallel Library, In: Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications.

Krausner, G. E., Stephen T. P. (1988). "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System", (Report). ParcPlace Systems, Inc.. Retrieved 2012-06-05.

MPI Forum (2008) "Message passing interface forum", http://www.mpi-forum.org/.