Desenvolvimento de perfis paralelos para avaliação de algoritmos de balanceamento de carga

TIAGO C. BOZZETTI [1], LAÉRCIO L. PILLA [2], PHILIPPE O. A. NAVAUX [3]

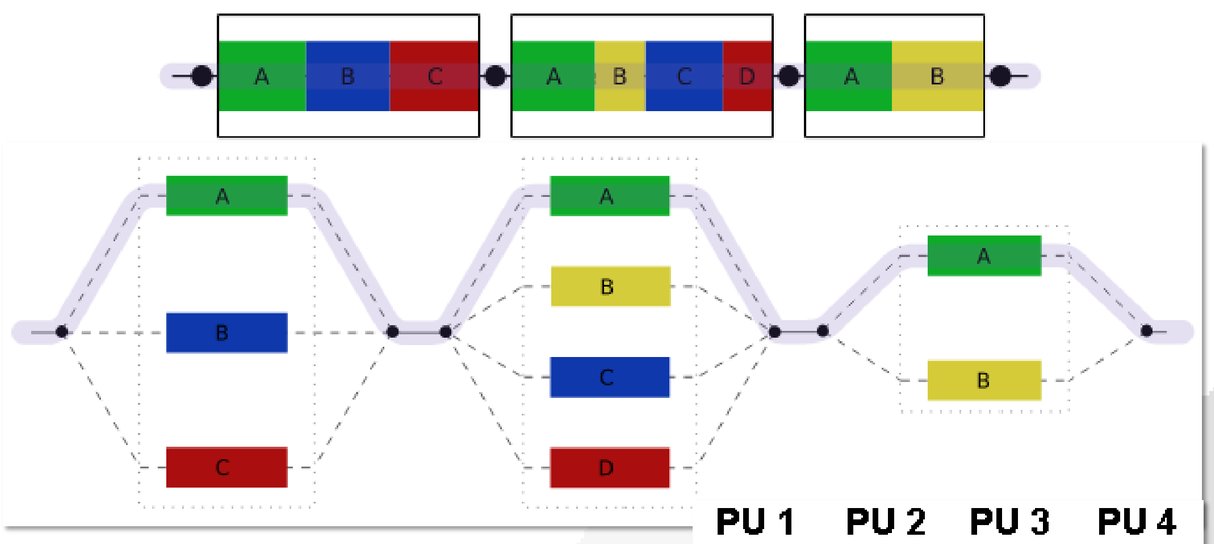
1 Tiago Covolan Bozzetti, Ciência da Computação, UFRGS
2 Laércio Lima Pilla
3 Philippe Olivier Alexandre Navaux

UFRGS PROPESQ XXV SIC Salão Iniciação Científica
CET - Ciências Exatas e da Terra

INTRODUÇÃO

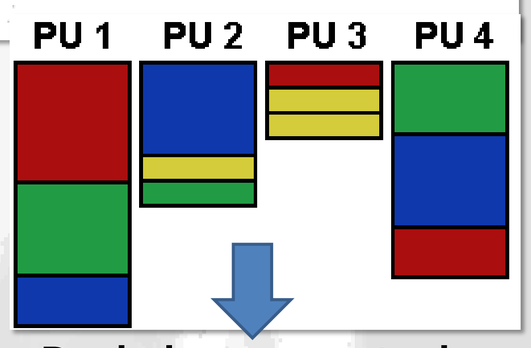
Como modificar uma aplicação para utilizar os recursos de plataformas paralelas?

- Fragmentando-a em tarefas que podem ser executadas em paralelo.



Como distribuir as tarefas entre as unidades de processamento?

- Distribuição igualitária seria o ideal.
- Unidades de processamento (PUs) podem ficar sobrecarregadas.
- Difícil de prever o comportamento da aplicação.
- Carga computacional de uma tarefa pode ser **dinâmica**.
- Problema **NP-Completo**.



Desbalanceamento de carga
↓
Unidade de processamento ociosa
↓
Desperdício de tempo
↓
Menor desempenho!

Problemática:

Como mapear tarefas para unidades de processamento de forma a explorá-las eficientemente?

Balanceadores de carga:

- Não podem prever o futuro, mas podem o estimar.
- Utilizam **heurísticas**.
- Centralizados ou distribuídos.
- Consideram informações da aplicação e da máquina.

Como avaliá-los?

- **Simular aplicações paralelas** com características distintas.
- Observar e compreender o desempenho dos algoritmos dada uma configuração de máquina e de aplicação.

METODOLOGIA

- Foi utilizado o framework Charm++ para o desenvolvimento do perfil paralelo. Ele possui balanceadores de carga que podem ser avaliados.
- O Charm++ é uma extensão de C++ que permite objetos da linguagem se comunicarem em um ambiente paralelo.
- O perfil paralelo utiliza uma série de parâmetros que são determinantes para avaliar um balanceador de carga, como:
  - Número de tarefas
  - Número de iterações
  - Volume de dados das tarefas
  - Tempo de processamento das tarefas
  - Grafo de comunicação
  - Tamanho das mensagens
- Os parâmetros são definidos como expressões e avaliados por um compilador C++. Portanto, é possível defini-los em função do número de tarefas e iterações, o que garante uma grande **expressividade para explorar diferentes características das aplicações**. Exemplos:

n = Número de tarefas m = Número de iterações

C_{ik} = Carga computacional da tarefa k na iteração i

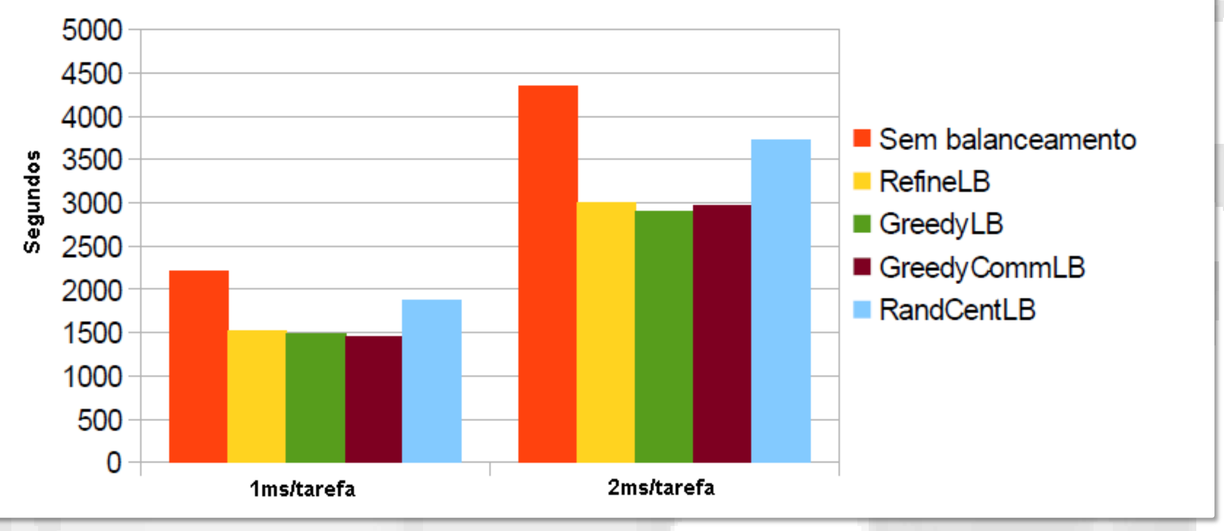
$$C_{ik} = \sin(k * i)$$

$$C_{ik} = \begin{cases} i^2, & i \leq m/2. \\ (m/2)^2, & i > m/2. \end{cases} \qquad 1 \leq k \leq n; \quad 1 \leq i \leq m$$

$$C_{ik} = 2^k + i$$

RESULTADOS

Tempo de execução do benchmark com 500 tarefas, 100 iterações, carga incremental de 1ms e 2ms por tarefa e 16 processadores.



- Na configuração utilizada, tarefas possuem cargas diferentes, as quais são mal distribuídas entre as unidades de processamento. O desbalanceamento é tão evidente que até um balanceador que toma decisões aleatoriamente (RandCentLB) consegue um melhor desempenho que a execução sem balanceamento (mas ainda longe dos outros algoritmos).

CONCLUSÃO

- O perfil paralelo desenvolvido fornece o suporte necessário para simular aplicações com diferentes características e **revelar atributos dos balanceadores** que não perceptíveis em um primeiro momento.
- Trabalhos futuros incluem testes em diferentes plataformas e com diferentes variações de parâmetros.

REFERÊNCIAS

J. Y. T. Leung, Handbook of scheduling: algorithms, models, and performance analysis, ser. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, 2004.

G. Zheng, A. Bhatele, E. Meneses, and L. V. Kale, “Periodic Hierarchical Load Balancing for Large Supercomputers,” International Journal of High Performance Computing Applications (IJHPCA), Mar. 2011.

L. L. Pilla, C. P. Ribeiro, P. Coucheney, F. Broquedis, B. Gaujal, P. O. A. Navaux, and J.-F. Méaut, “A Topology-Aware Load Balancing Algorithm for Clustered Hierarchical Multi-Core Machines,” Future Generation Computer Systems, 2013.

25 anos
MODALIDADE DE BOLSA
CNPq Universal