VOZES DIVERSAS
DIFERENTES SABERES

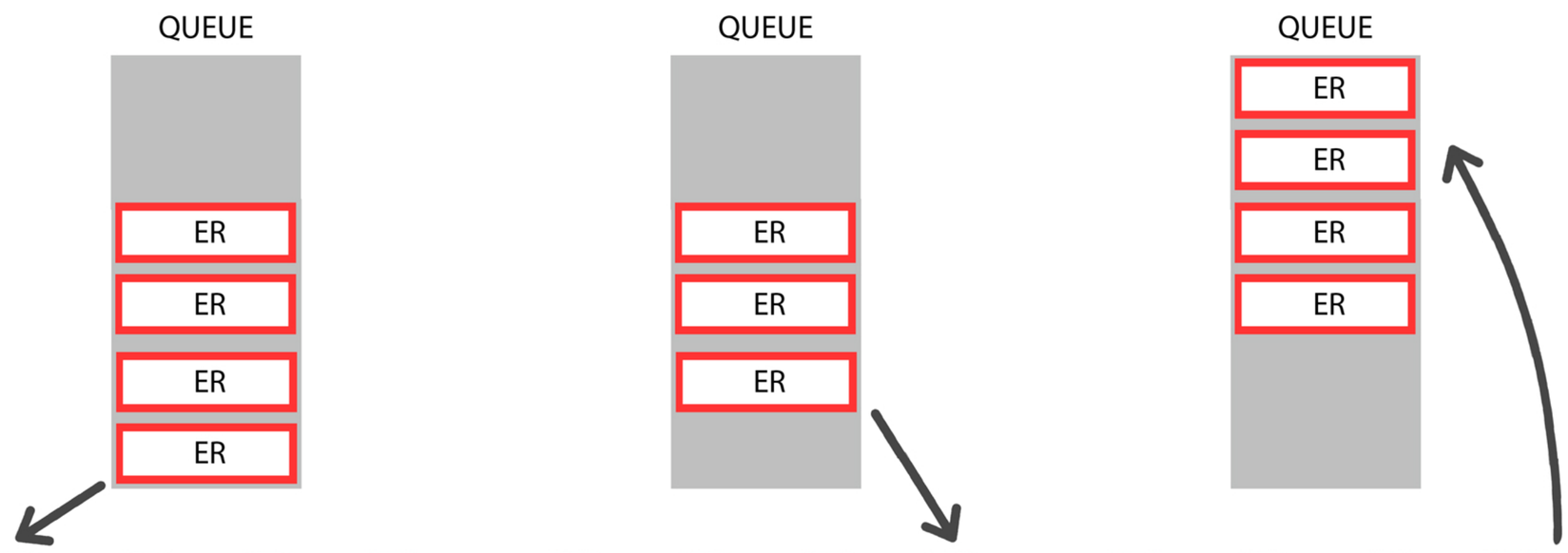
CONHECIMENTO FORMAÇÃO INOVAÇÃO
Salão UFRGS 2018

SALÃO DE INICIACAO CIENTIFICA
XXX SIC

15 A 19 OUTUBRO
CAMPUS DO VALE

UFRGS
PROPESQ

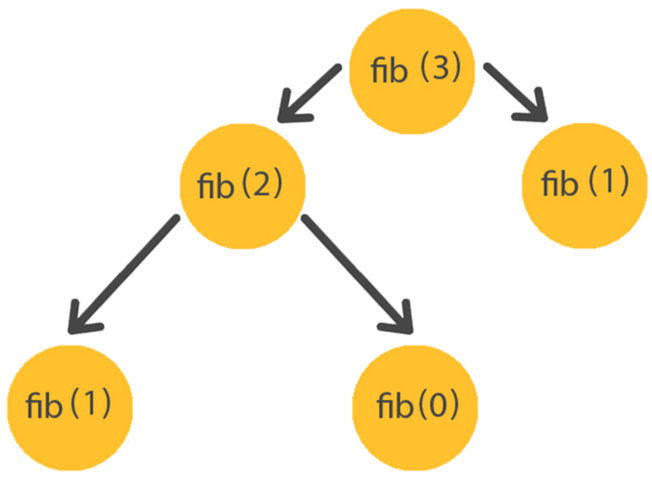
Paralelização Automática de Linguagens Funcionais baseada no Acelerador ACQuA



Traço de execução de uma função qualquer sob o modelo ACQUA: na primeira imagem, o primeiro core inicia a execução da primeira chamada; na sequência, o último core começa a executar a chamada seguinte; por fim, durante a execução da função, o último core cria mais duas chamadas

Estamos em um período em que a execução de tarefas em múltiplas unidades de processamento é indispensável para se ter um programa com bom desempenho. Entretanto, desenvolver soluções que realizam isso nem sempre é trivial: é necessário encontrar regiões que podem ser executadas concorrentemente e também levar em consideração questões de baixo nível, como sincronização e comunicação entre threads. Além disso, em linguagens procedurais, por exemplo, a ordem de execução das instruções é importante para o correto funcionamento de um programa, o que dificulta a automatização deste processo. Por outro lado, linguagens funcionais possuem ausência de efeitos colaterais: basicamente, a ordem de execução de chamadas de função independentes não é relevante.

Tendo isso em vista, foi desenvolvido o acelerador ACQuA [1] [2]. Essencialmente, ele coloca todas as chamadas de função em uma fila, e qualquer núcleo pode executar qualquer chamada de função que esteja nela. Durante a execução, quando um núcleo encontra uma nova chamada de função, a mesma é colocada ao final da fila. O objetivo deste trabalho é aplicar a ideia por trás do acelerador ACQuA em software. Isso será feito da seguinte forma: dado um programa escrito em uma linguagem funcional, como Haskell, será gerado um código intermediário, neste caso em C++ com OpenMP. Esse código intermediário implementa o mesmo algoritmo recebido como entrada, mas seguindo o modelo ACQuA e já tratando a sincronização e comunicação entre as threads.



Árvore de chamadas recursivas de fib de 3

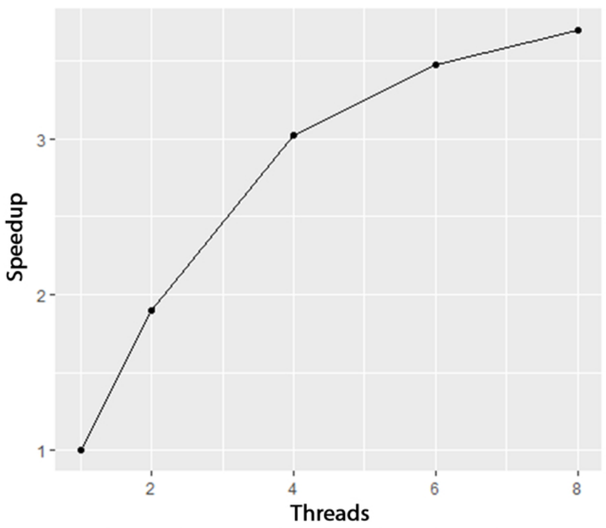


Gráfico da escalabilidade do algoritmo recursivo de Fibonacci sob o modelo ACQuA

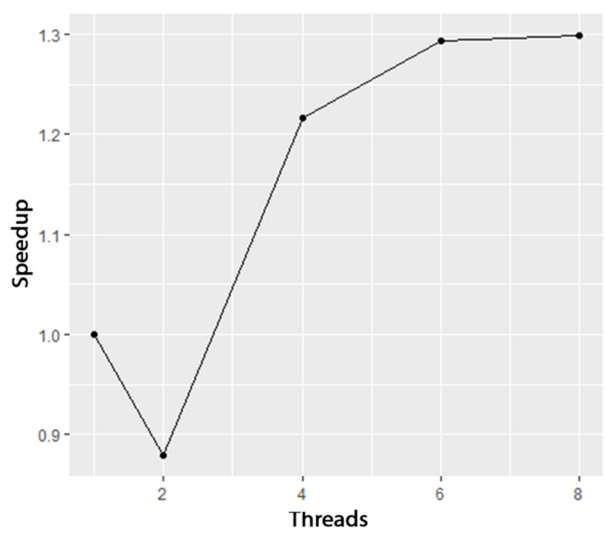


Gráfico da escalabilidade do Merge Sort sob o modelo ACQuA

Até o momento foram realizados testes com a versão recursiva de Fibonaci e com o algoritmo de ordenação Merge Sort. Em ambos os casos foi realizada a tradução manual de um código escrito em Haskell para um código em C++ com OpenMP seguindo o modelo de computação da arquitetura ACQuA. No primeiro caso, com 8 threads, foi obtido um speedup igual a aproximadamente 4 em comparação com a execução do mesmo código com apenas uma thread. Com o Merge Sort o speedup foi de apenas aproximadamente 1.3, o que ainda é um bom resultado levando em consideração que limite teórico do speedup para esse algoritmo seria 2 segundo a Lei de Amdahl.

Ainda é necessário avaliar a viabilidade e o desempenho da utilização deste modelo de computação com algoritmos mais complexos, nos quais a exploração de paralelismo não é trivial. Além disso, também é preciso realizar a tradução automática de um código escrito em uma linguagem funcional, como Haskell, para um código em C++ com OpenMP seguindo este modelo.

REFERÊNCIAS

[1] TANUS, F.; NAZAR, G.; MOREIRA, A. **Parallelization of Function Applications with the ACQuA Architecture.** Relatório técnico, 2017.

[2] TABORDA, T. **Implementação de Acelerador com Arquitetura Multi-Núcleos ACQuA. Monografia, 2017.** Disponível em: https://www.lume.ufrgs.br/handle/10183/169060