TODO: Title

Matheus B. Nascimento[1], **Wisllay Vitrio**[1]

[1] Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – CEP 74.001-970 – Goiânia – GO – Brasil

Abstract. *Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract.*

Resumo. *Neste trabalho é apresentada uma implementação do espaço de tuplas em Go, permitindo que vários processos o acessem simultaneamente, realizando leituras e escritas de tuplas com valores de qualquer tipo. A implemnetação foi testada utilizando um exemplo que explora paralelismo. Ganhos de speedup foram obtidos com o aumento no número de servidores.*

1. Introdução

Espaço de tuplas é um conceito de memória associativa para computação distribuída/-paralela. Desenolvido por David Gelernter na Universidade de Yale, teve sua primeira implementação na linguagem de coordenação "Linda" (homenagem a uma atriz pornô de nome Linda Lovelace, assim como o nome da linguagem "Ada" é uma homenagem à Ada Lovelace [**?**]).

O espaço de tuplas provê um repositório de tuplas que pode ser acessado concorrentemente por zero ou mais processos. Por ser baseado em memória associativa, as tuplas são acessadas por seu conteúdo e não por endereços. As tuplas não estão ligadas à nenhum processo, e qualquer um deles pode inserir, ler ou remover tuplas. Este desacoplamento total entre as partes integrantes do sistema provido pelo espaço de tuplas é sua principal vantagem.

Várias linguagens têm implementações de espaço de tuplas, sendo a especificação para Java, JavaSpaces [**?**], a mais famosa. Como parte da tecnologia Jini, JavaSpaces é utilizado em serviços de finanças e telecomunicações para alcançar escalabilidade utilizando processamento paralelo. Por outro lado, a tecnologia Jini como um todo não é sucesso comercial, e o JavaSpaces não é amplamente utilizado.

2. O Espaço de Tuplas

Como foi descrito na seção 1, o espaço de tuplas permite que processos se comuniquem sem o conhecimento um do outro, local ou remotamente, inserindo e removendo objetos representados por tuplas de valores. Para realizar o acesso aos dados do espaço de tuplas, o paradigma oferece três primitivas: *read()*, *take()* e *write()*. Esta seção explica o funcionamento de cada uma dessas primitivas e como foram implementadas levando em consideração as decisões arquiteturais tomadas.

2.1. Primitivas

Um dos exemplos de implementação do espaço de tuplas é o JavaSpaces [**?**], que foi uma solução em Java para o espaço de tuplas. Nele as tuplas são armazenadas como uma

sequência de valores primitivos *(*int, double, String, etc.) através da primitiva *write()*. E os valores são lidos passando-se um *template*, que representa a tupla procurada. A estrutura do template é a mesma da tupla, e serve como filtro para as primitivas *read()* e *take()*. Essas primitivas retornam uma tupla que combine com o template passado como parâmetro. No caso do JavaSpaces, o template é uma tupla, onde cada valor pode ser nulo, significando que qualquer valor é valido para aquela posição da tupla ou um valor específico, o qual deve combinar com a tupla procurada.

Na solução de espaço de tuplas desenvolvida nesse trabalho, foi utilizada a linguagem Go [**?**], e seguimos o mesmo modelo do JavaSpaces para a representação das tuplas, para escrita e leitura, com a que aceitamos qualquer tipo de dado que possa ser serializado como atributo de uma tupla.

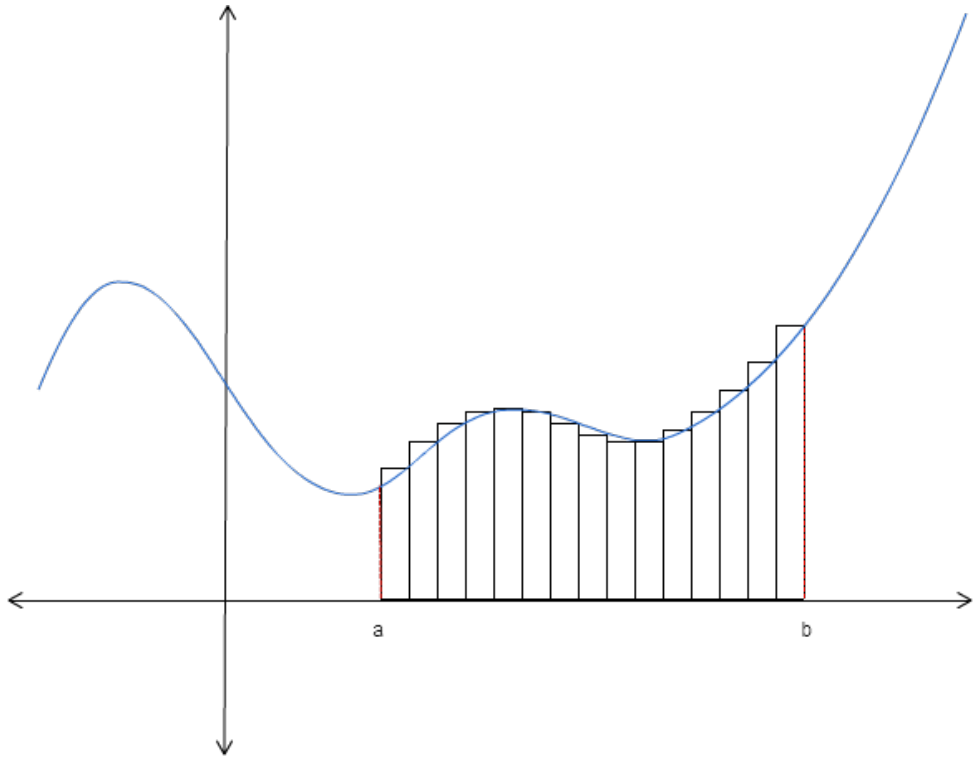

Figura 1. Variação de recebimento durante a execução do algoritmo QL

3. Caso de teste

4. Resultados

Resultados

5. Conclusão

Conclusão

Referências

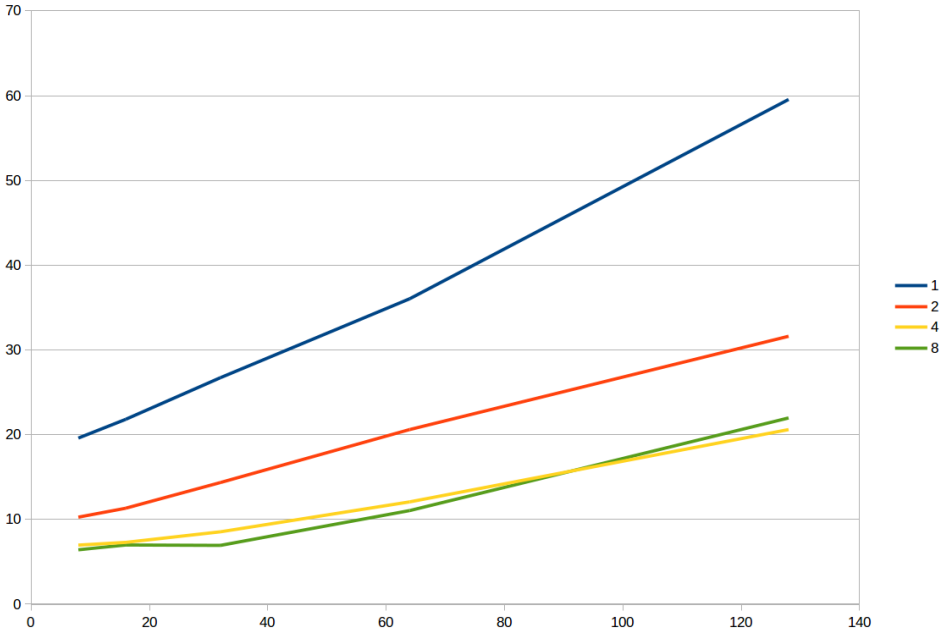

Figura 2. Variação de recebimento durante a execução do algoritmo QL

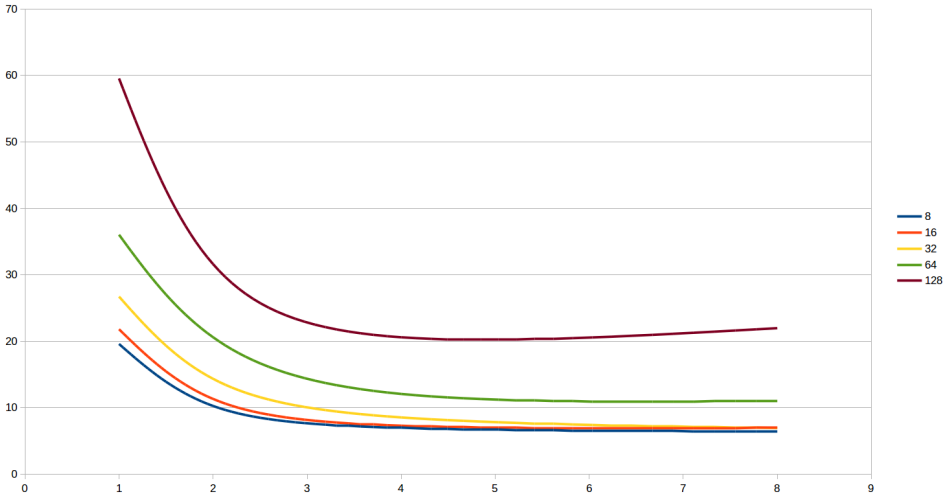

Figura 3. Variação de recebimento durante a execução do algoritmo QL

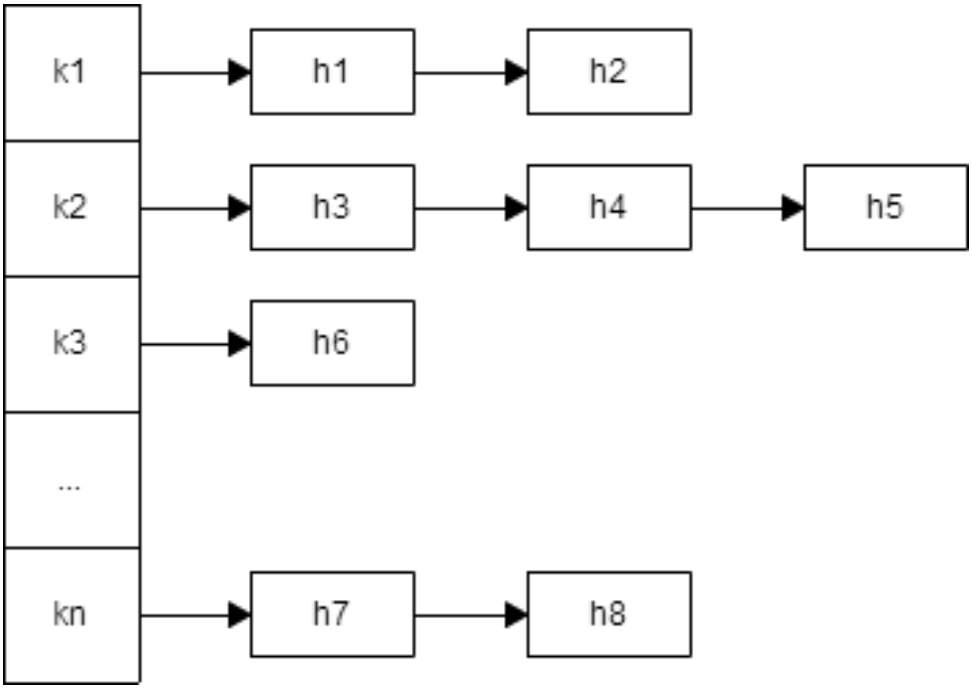

Figura 4. Variação de recebimento durante a execução do algoritmo QL

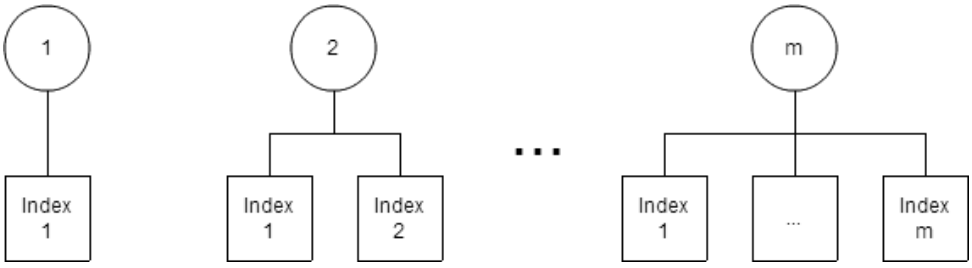

Figura 5. Variação de recebimento durante a execução do algoritmo QL