

# LTMS - Lups Transactional Memory Scheduler: Um escalonador NUMA-Aware para STM.

**Michael Alexandre Costa**

Prof. Dr. André Rauber Du Bois (Orientador)

Mestrado em Computação  
Centro de Desenvolvimento Tecnológico  
Universidade Federal de Pelotas  
`macosta@inf.ufpel.edu.br`

13 de junho de 2021



- 1 **Introdução**
- 2 **Memórias Transacionais**
- 3 **Arquiteturas**
- 4 **Escalonadores**
- 5 **LTMS**
- 6 **Experimentos**
- 7 **Resultados**
- 8 **Conclusão**



# Introdução

## Motivação

- Programação Paralela;
- Memórias Transacionais;
- Escalonadores de Transações; e
- Arquiteturas NUMA.



# Introdução

## Objetivos

- Projetar um escalonador de STM modular que considera a arquitetura utilizada, intitulado LTMS;
- Prototipar o escalonador LTMS, utilizando a biblioteca de STM TinySTM; e
- Análisar de desempenho do LTMS comparado a TinySTM utilizando o conjunto de benchmarks STAMP.



# Memórias Transacionais

## Características

- Fornece abstração de código;
- Reuso de código; e
- Ausência de deadlocks.

## Transações

- Atomicidade;
- Consistência; e
- Isolamento.



# Memórias Transacionais

## Problemas

- Somente reinicia a transação conflitante;
- Não evita que conflitos futuros aconteçam; e
- Em ambientes de alta contenção, tende a perder desempenho.



# Arquiteturas

## UMA

- Uniform Memory access;
- Possui um único barramento de acesso à memória; e
- Único custo de acesso à memória.

## NUMA

- Non-uniform Memory access;
- Possui mais de um barramento de acesso à memória; e
- O custo de acesso à memória é diferente conforme o núcleo utilizado.



# Escalonadores

## Escalonadores de Transações

- Buscam reduzir os números de conflitos;
- Utilizam diferentes Heurísticas de escalonamento; e
- Serializa as transações conflitantes.





# Escalonadores

## Classificação das técnicas

- Baseado em Heurística:
  - Feedback;
  - Predição;
  - Reativo; e
  - Heurística Mista.
- Baseado em Modelo:
  - Aprendizado de Máquina;
  - Modelo Analítico; e
  - Modelo Misto.



# Escalonadores

**Tabela:** Comparativo entre os escalonadores apresentados

Escalonadores	LTMS	ATS	Shrink	LUTS	ProVIT	STMap	CAR-STM
Distribuição inicial de threads	Sim	Não	Não	Sim	Não	Não	Não
Coleta de dados por threads	Sim	Não	Sim	Não	Não	Sim	Não
Migração entre filas	Sim	Não	Não	Não	Não	Não	Sim
Avalia a arquitetura	Sim	Não	Não	Não	Não	Sim	Não
Técnica de escalonamento	Reativo	Feedback	Predição	Mista	Mista	Predição	Reativo



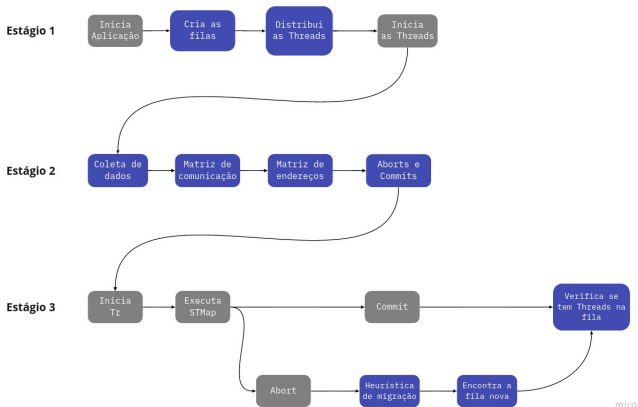
# LTMS

## Estágios

- Inicialização do sistema;
- Coleta de dados em tempo de execução; e
- Migração de Threads.



# LTMS



**Figura:** Fluxograma do LTMS



# LTMS - Estágio 1

## Inicialização do sistema

- Criação de filas; e
- Distribuição das threads.

## Heurísticas de Distribuição

- Sequential; e
- Chunks.



# LTMS - Heurísticas

4 Threads  
2 Cores



1ª etapa  
Distribui 1 thread para cada fila



2ª etapa  
Se ainda tem thread a distribuir  
Distribui 1 thread para cada fila



**Figura:** Heurística Sequential

# LTMS - Heurísticas

4 Threads  
2 Cores



1ª etapa  
Distribui chunks por fila



2ª etapa  
Distribui chunks por fila



**Figura:** Heurística Chunks

# LTMS - Estágio 2

## Coleta de dados em tempo de execução

- Aborts e Commits;
- Matriz de Comunicação; e
- Matriz de Endereços.





# LTMS - Matrizes

## Matriz de Comunicação

- Quantidade de comunicação entre pares de threads;
- Eventos de Comunicação; e
- 1 evento a cada 100 acessos.



# LTMS - Matrizes

## Matriz de Endereços

- Endereço mais acessado entre pares de threads;
- Tabela Hash;
- Endereços de memória; e
- Quantidade de acessos recebidos.



# LTMS - Estágio 3

## Migração de Threads

- Abort;
- Identificação; e
- Heurísticas de migração.



# LTMS - Filas e Threads

## Identificação das filas e threads

- Identificação das threads conflitantes; e
- Matriz de comunicação.



# LTMS - Heurísticas

## Threshold

- Nível de contenção (Abort/Commit);
- Maior contenção;
- Menor contenção; e
- Limiar de 0.8 (80% de contenção).



# LTMS - Heurísticas

## Latency

- Matriz de endereços;
- Nodos NUMA;
- Bancos de memória; e
- Latencia.



# Experimentos

## Aplicação

- TinySTM 1.0.5; e
- STMAP 0.9.10.

## Arquitetura

- Intel Xeon E5-4650;
- 96 núcleos e 192 threads;
- 468Gb de memória RAM.



# Experimentos

## Testes

- Cenários de threads:
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, e 512;
- Heurísticas de Distribuição-Migração:
  - Sequential-Threshold;
  - Chunks-Threshold;
  - Sequential-Latency;
  - Chunks-Latency;
- TinySTM; e
- Baterias de 30 execuções.





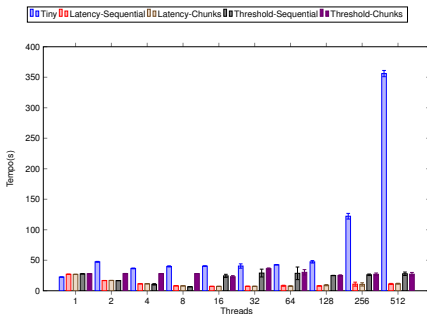
# Resultados

## Benchmarks

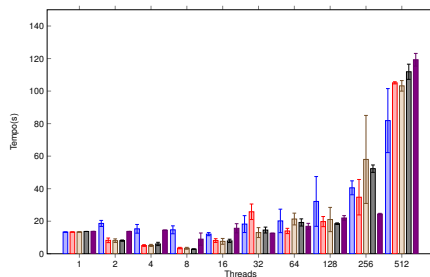
- Bayes;
- Intruder;
- Kmeans; e
- Labyrinth, Vacation, Yada.



# Tempo de execução

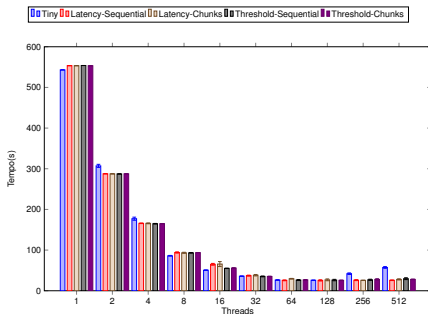


(a) Intruder

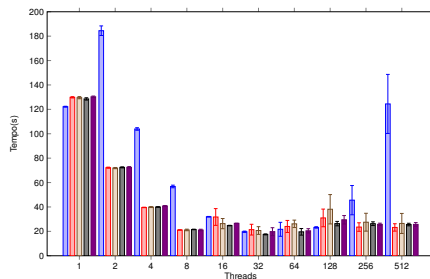


(b) Kmeans

# Tempo de execução

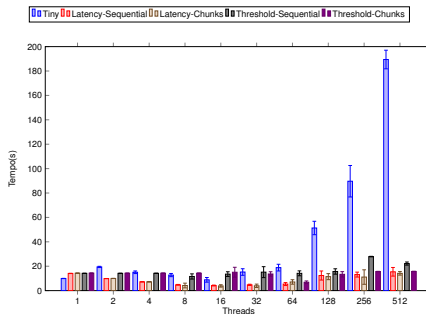


(c) Labyrinth



(d) Vacation

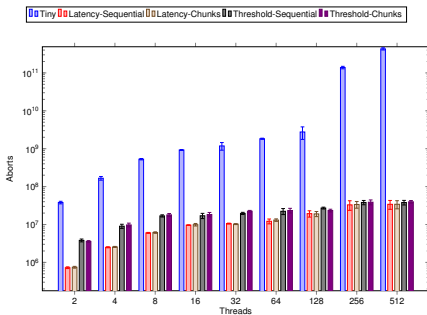
# Tempo de execução



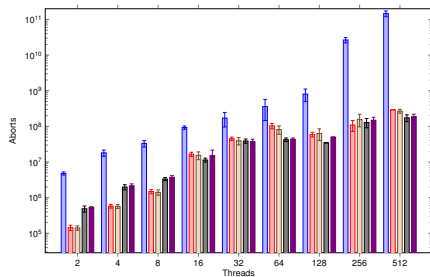
(e) Yada



# Aborts



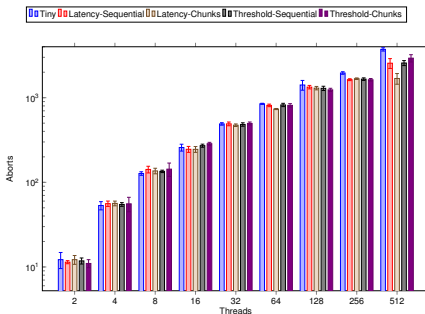
(f) Intruder



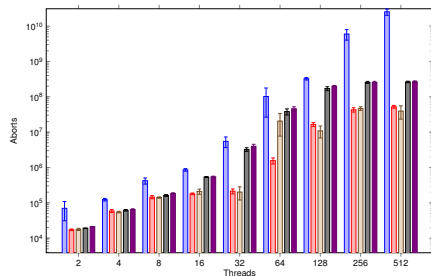
(g) Kmeans



# Aborts

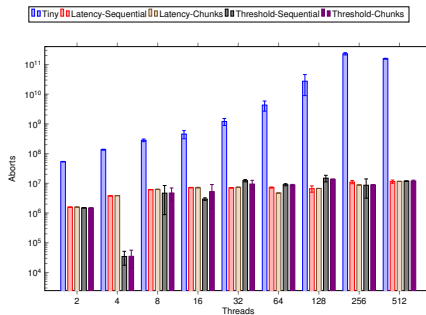


(h) Labyrinth



(i) Vacation

# Aborts



(j) Yada

# Conclusão

## Analise

- Aplicações com conjunto pequeno de leitura e escrita;
- Transação com tempo longo, médio, ou baixo;
- Contenção alta, média ou baixa;
- Redução de 96% no tempo de execução; e
- Redução de 99% na ocorrência de aborts.





# Conclusão

## Trabalhos futuros

- Novas Heurísticas de distribuição;
- Heurísticas de migração híbrida; e
- Impacto energético dos escalonadores de STM.



# **LTMS - Lups Transactional Memory Scheduler: Um escalonador NUMA-Aware para STM.**

**Michael Alexandre Costa**

Prof. Dr. André Rauber Du Bois (Orientador)

Mestrado em Computação  
Centro de Desenvolvimento Tecnológico  
Universidade Federal de Pelotas  
`macosta@inf.ufpel.edu.br`

13 de junho de 2021

