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

24 de junho de 2021

Introdução

Motivação

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

Introdução

Objetivos

- Investigar escalonadores de Memórias Transacionais em NUMA.

Contribuições

- Projeto de um escalonador de STM intitulado LTMS;
- Prototipação do escalonador LTMS; e
- Análise de desempenho do LTMS comparado a TinySTM.

Introdução

Características

- Mecanismo para leitura da arquitetura e criação de filas;
- Duas diferentes heurísticas de distribuição inicial de threads;
- Mecanismo que em tempo de execução coleta informações sobre as threads;
- Mecanismo de migração de threads entre as filas de execução; e
- Duas heurísticas de migração.

Conceitos abordados no trabalho

Principais Conceitos

- Memórias Transacionais;
- Escalonadores de transações; e
- Arquiteturas.

Memórias Transacionais

Características

- Fornece abstração 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.

Escalonadores

Escalonadores de Transações

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

Escalonadores

Trabalhos Estudados

- ATS;
- CAR-STM;
- Shrink;
- LUTS;
- ProVIT; e
- STMap.

Escalonadores

Tabela: Comparativo entre os escalonadores apresentados

Escalonadores	LTMS	STMap	ATS	Shrink	LUTS	ProVIT	CAR-STM
Distribuição inicial de threads	Sim	Não	Não	Não	Sim	Não	Não
Coleta de dados por threads	Sim	Sim	Não	Sim	Não	Não	Não
Migração entre filas	Sim	Não	Não	Não	Não	Não	Sim
Avalia a arquitetura	Sim	Sim	Não	Não	Não	Não	Não
NUMA	Sim	Sim	Não	Não	Não	Não	Não

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.

LTMS

Estágios

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

LTMS

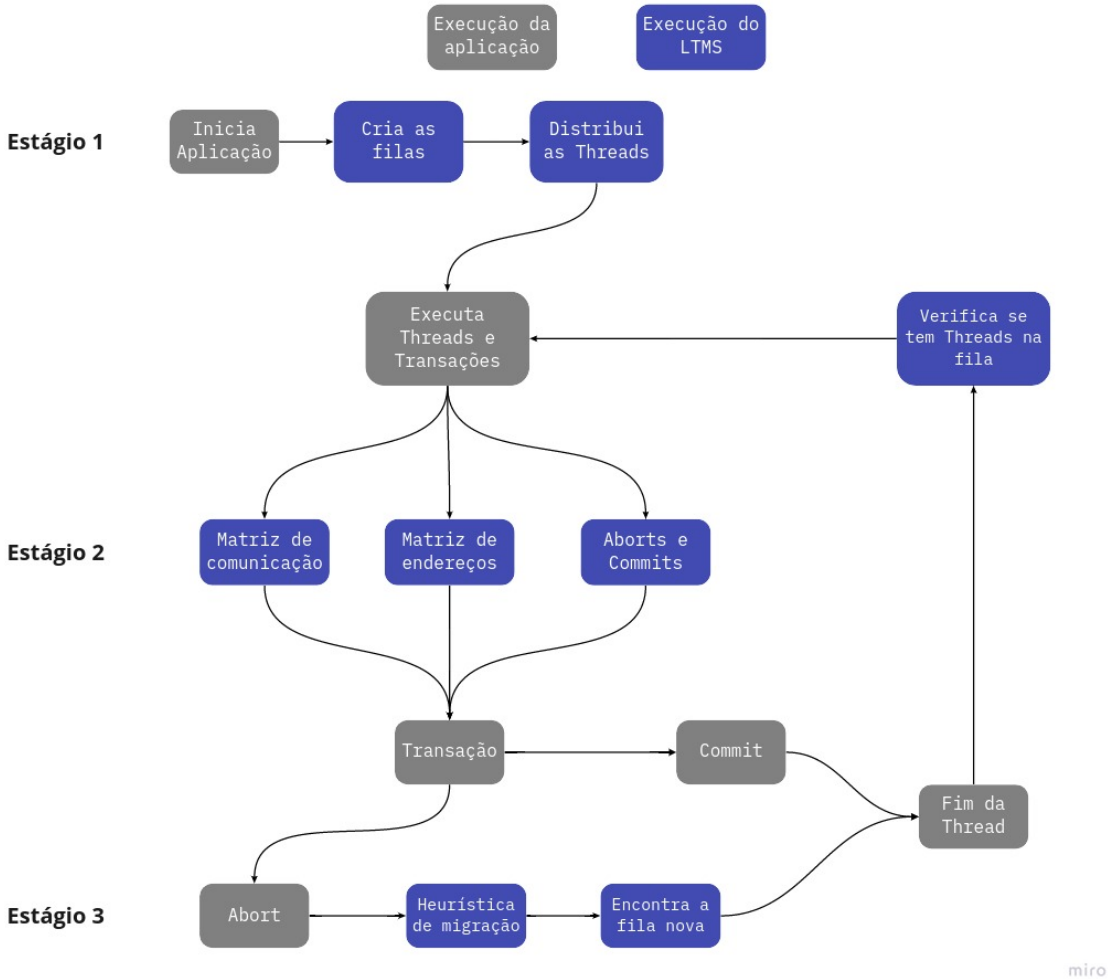


Figura: Fluxograma do LTMS

LTMS - Estágio 1

Inicialização do sistema

- Criação de filas; e
- Distribuição das threads.

Heurísticas de Distribuição

- Sequential; e
- Chunks.

LTMS - Heurísticas

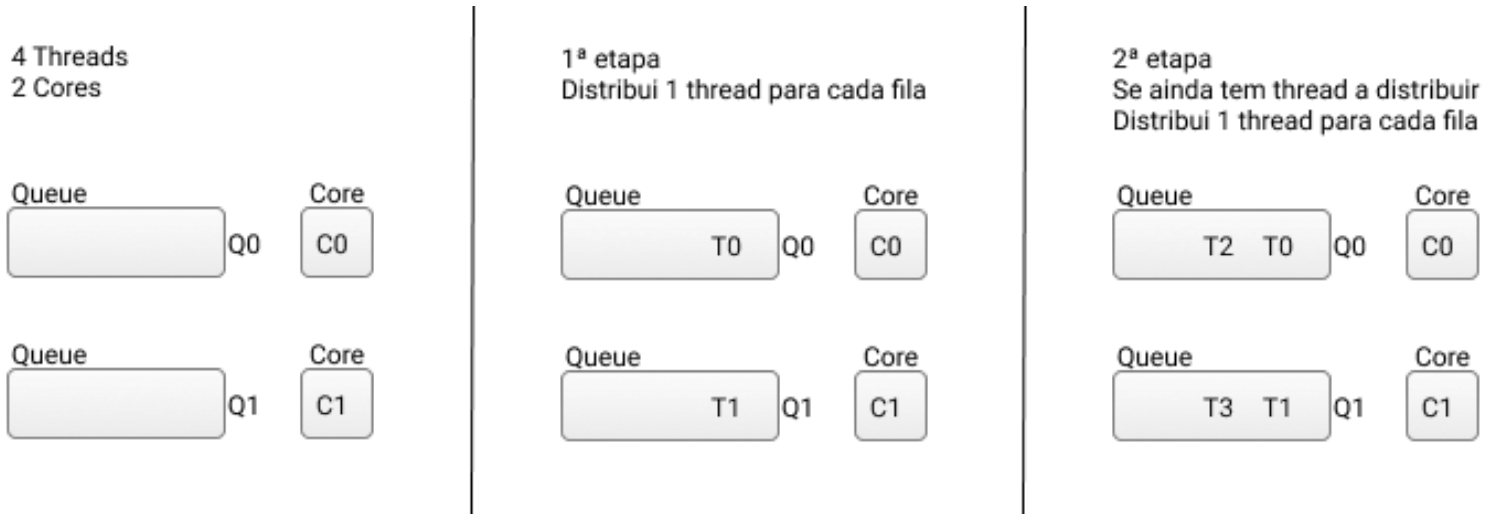


Figura: Heurística Sequential

LTMS - Heurísticas

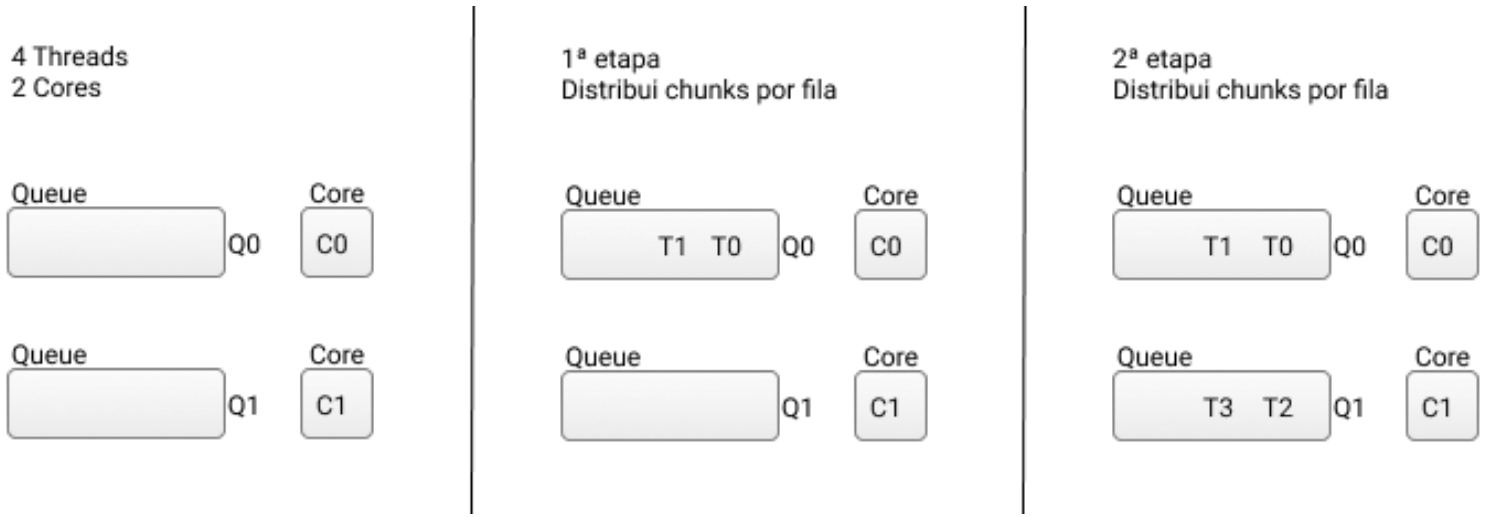


Figura: Heurística Chunks

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

- Eventos de comunicação são acessos em comum à memória;
- Quantidade de eventos de comunicação entre pares de threads; e
- É coletado 1 evento de comunicação a cada 100 acessos.

LTMS - Matrizes

Matriz de Endereços

- Endereços em comum mais acessados entre os pares de threads;
- Utiliza uma Tabela Hash;
- Chave: Endereços de memória; e
- Valor: Quantidade de acessos recebidos.

LTMS - Estágio 3

Migração de Threads

- Quando ocorre um abort;
- Identificar a melhor fila; e
- Heurísticas de migração.

LTMS - Filas e Threads

Escolha das filas

- Identifica a fila que possui a thread com mais acessos em comum;
- Utiliza a matriz de comunicação; e
- Busca uma melhor coerência de cache.

LTMS - Heurísticas

Threshold

- Avalia o nível de contenção (Abort/Commit);
- Limiar alto permite maior contenção e gera menos migrações;
- Limiar baixo permite menor contenção e gera mais migrações; e
- Limiar de 0.8 (80% de contenção).

LTMS - Heurísticas

Latency

- Avalia a latência de acesso à memória;
- Matriz de endereços;
- Nodos NUMA; e
- Bancos de memória.

Experimentos

Aplicação

- TinySTM 1.0.5; e
- STAMP 0.9.10.

Arquitetura

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

Experimentos

Testes

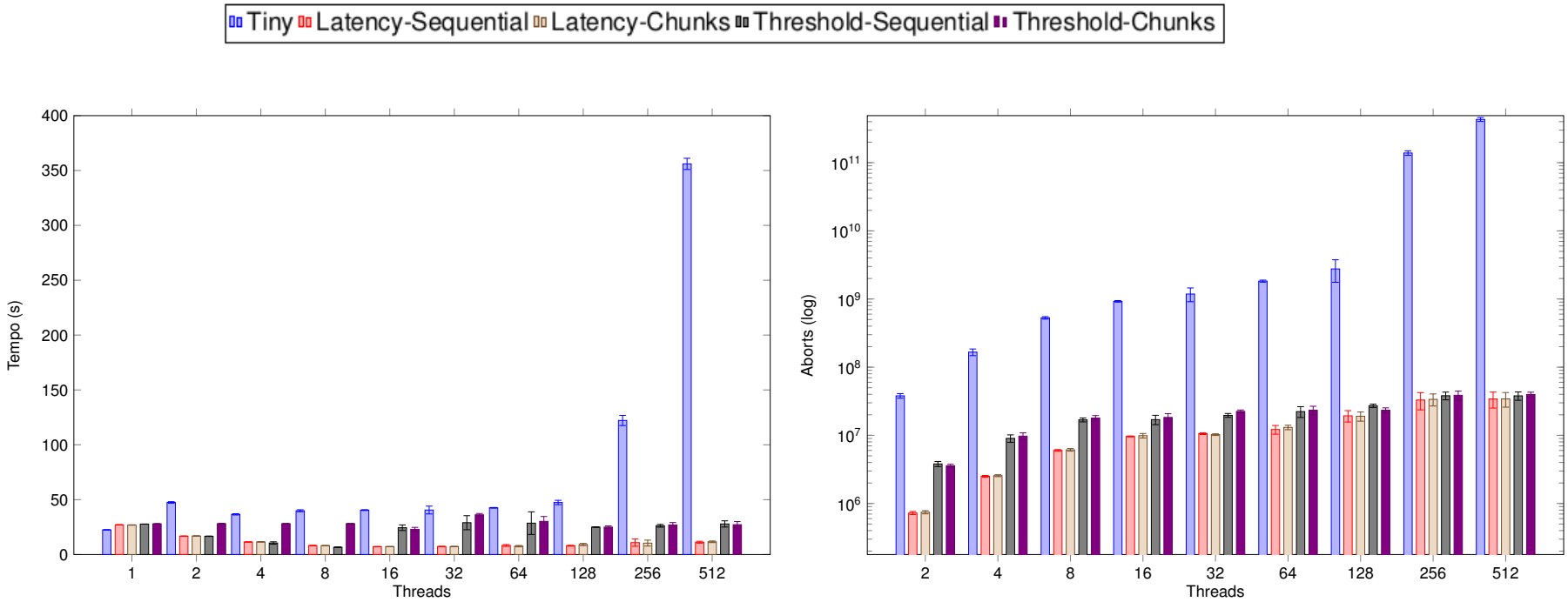
- Cenários de threads:
  - 1, 2, 4, 8, 16, 32, 64, 128, 256, e 512;
- Heurísticas de Distribuição-Migração:
  - Sequential-Threshold;
  - Chunks-Threshold;
  - Sequential-Latency;
  - Chunks-Latency;
- TinySTM; e
- Baterias de 30 execuções.

Resultados

Benchmarks

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

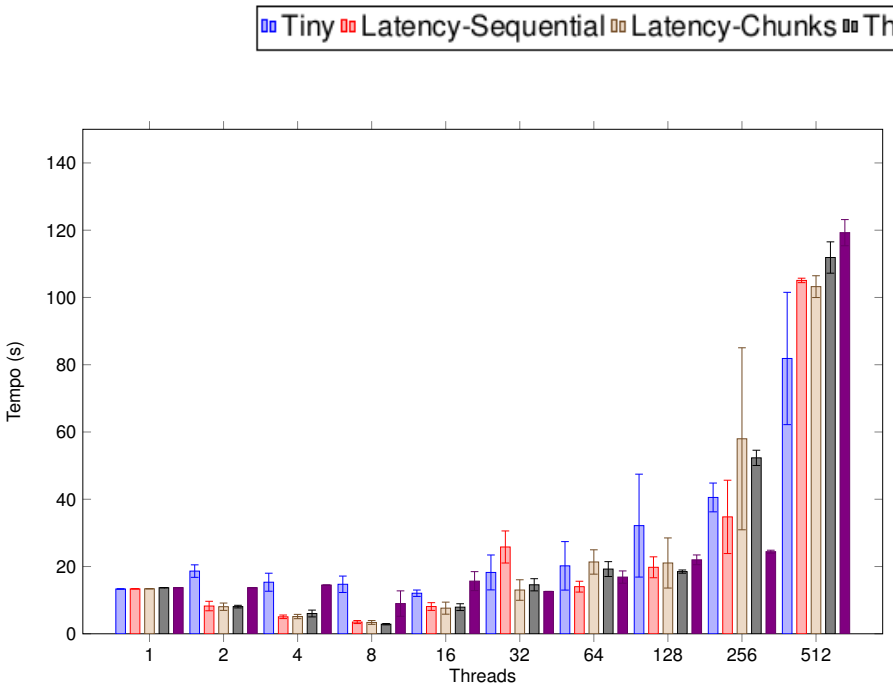
Intruder



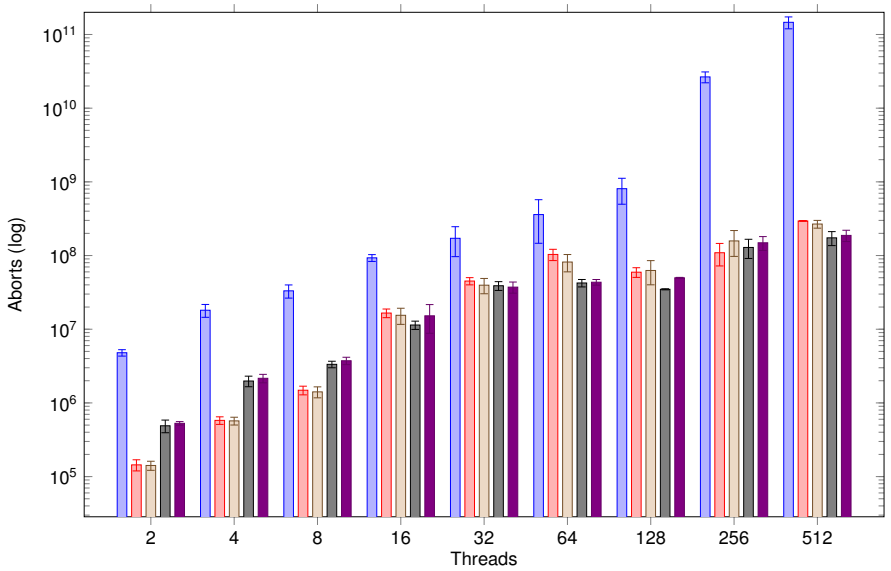
(a) Tempo de execução

(b) Aborts

Kmeans

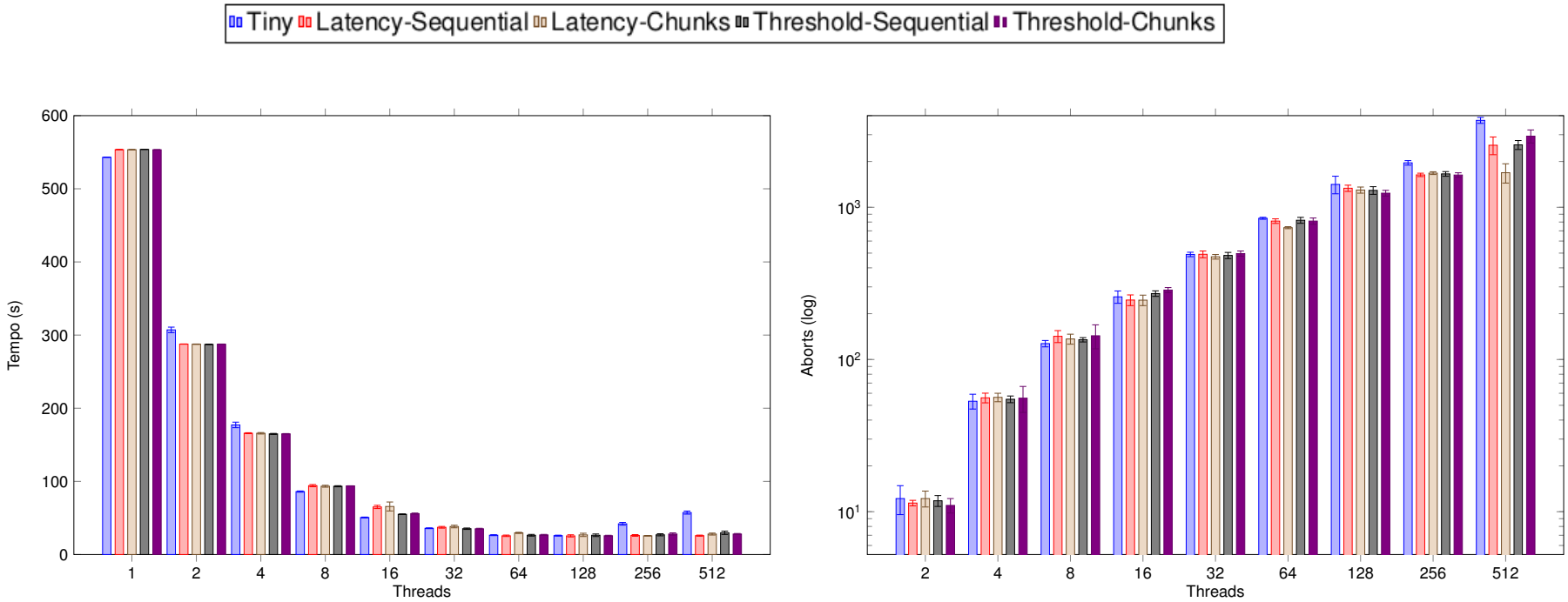


(c) Tempo de execução



(d) Aborts

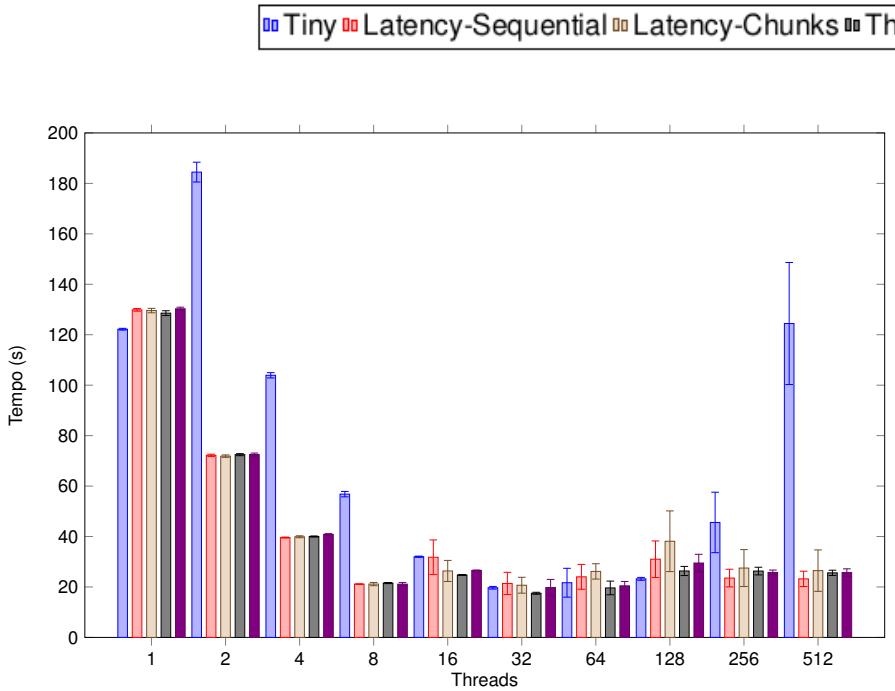
Labyrinth



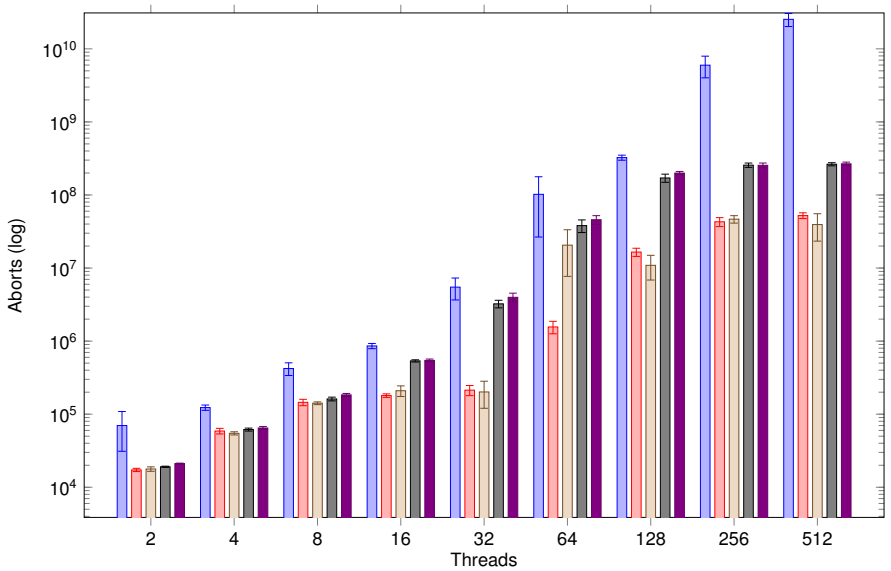
(e) Tempo de execução

(f) Aborts

Vacation

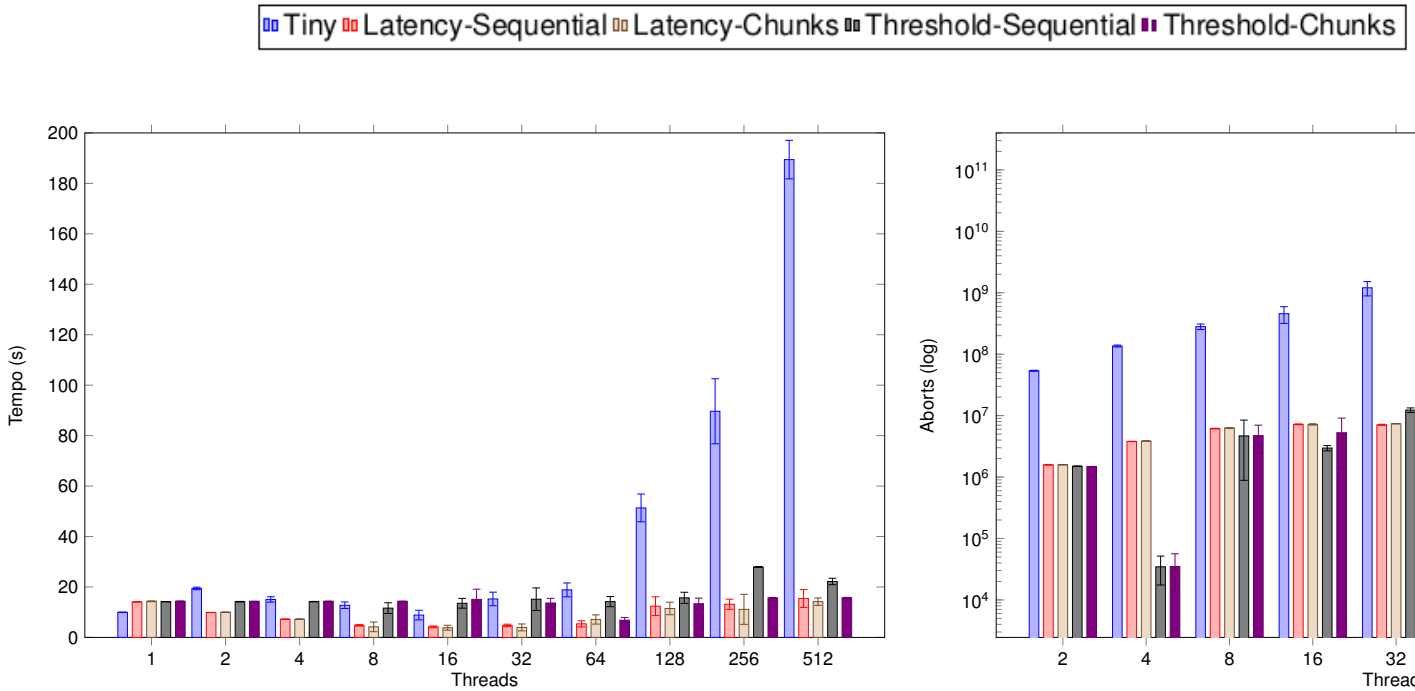


(g) Tempo de execução



(h) Aborts

Yada



(i) Tempo de execução

(j) Aborts

Conclusão

Conclução

- Apresentamos o Escalonador LTMS;
- Foi prototipado utilizando a TinySTM; e
- Possui 3 etapas de execução.

Conclusão

Analise

- Aplicações com conjunto pequeno de leitura e escrita;
- Tamanho médio de transação apresentou melhor execução;
- Alta contenção apresentou melhor tempo execução;
- Melhor caso com redução de 96% no tempo de execução;
- Melhor caso com redução de 99% na ocorrência de aborts; e
- Latency apresentou resultados melhores para maioria dos testes.

Conclusão

Trabalhos futuros

- Novas Heurísticas de distribuição;
- Juntar as heurísticas de migração Threshold e Latency; 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

24 de junho de 2021