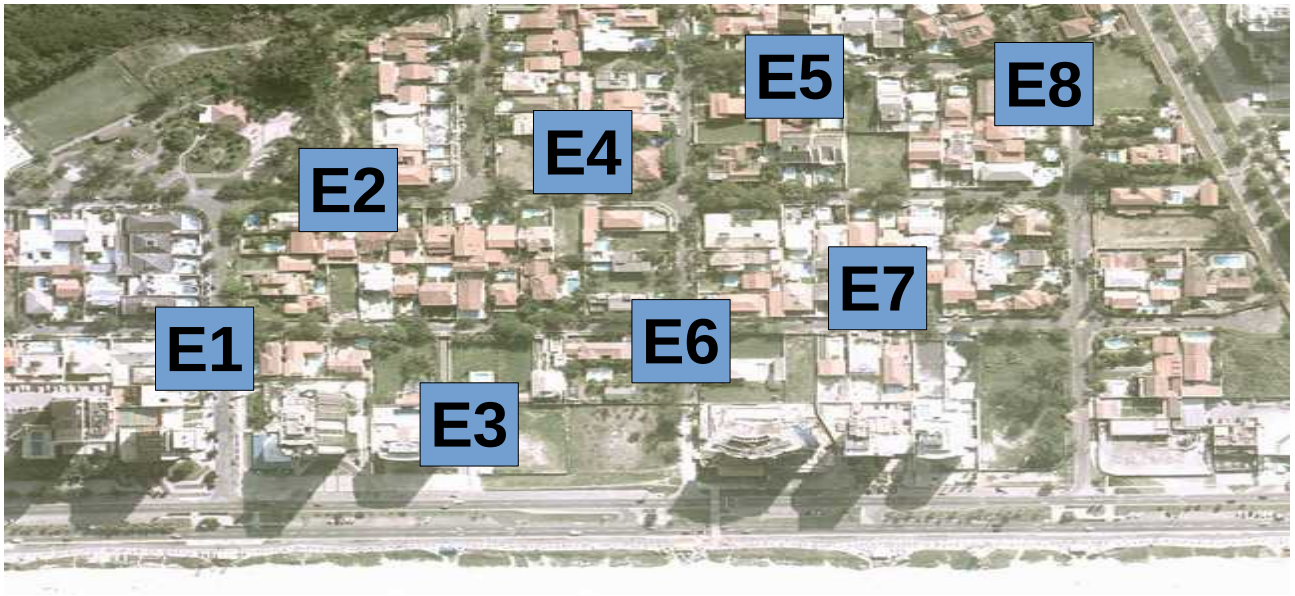
PESQUISA PARA A TESE DOUTORADO
HORUS IMAGE SERVER v1.0

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@globo.com
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ

Definição do Problema

PROBLEMA:

O crescente volume de dados produzidos pelas redes de sensores e equipamentos de gravação de imagens, torna difícil o processo de captura, transmissão, classificação e identificação dos dados coletados remotamente.



Motivações:
Capacidade de Processamento Remoto

CAPACIDADE DE PROCESSAMENTO REMOTO

O aumento na capacidade de processamento das unidades de monitoramento permite que sejam realizadas pré-classificações dos dados coletados por cada dispositivo, possibilitando a decisão pelo envio ou não da imagem e proporcionando uma redução em até 20x o volume de dados enviados.

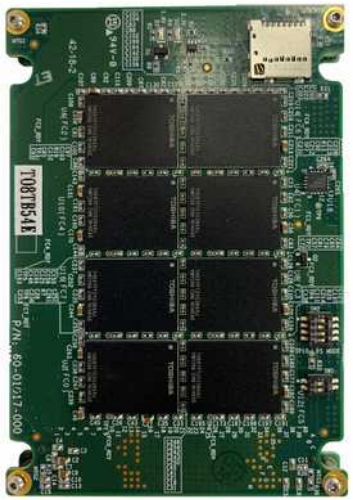
Raspberry Pi 4 - Processador Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64bit SoC – Clock 1.5 Ghz – Memória RAM: 4GB DDR4

Jetson Nano - Arquitetura NVIDIA Maxwell com 128 NVIDIA CUDA cores - Processador Quad-core ARM Cortex-A57 MPCore - 4 GB 64 -bit LPDDR4 - 16 GB eMMC 5.1 Flash

Motivações:
Computation Storage

***COMPUTATION STORAGE*:**

O conceito de computation storage vem sendo amplamente estudado com a aplicação dos recursos de processamento diretamente na unidade de armazenamento e permitindo que diversos serviços suportados anteriormente no sistema operacional ou na camada da aplicaçã, estejam disponíveis na unidade de armazenamento, reduzindo o trabalho da CPU e proporcionando maior velocidade na transferência dos dados entre as unidades de armazenamento e memória.



Objetivo da Pesquisa

OBJETIVO DA PESQUISA:

Identificação de padrões nas imagens dos equipamentos de monitoramento e supervisão, com o objetivo de reduzir o volume de dados trafegados pela rede.

Pesquisa e implementar um mecanismo de armazenamento otimizado dos modelos de aprendizado de Redes Neurais aplicado na classificação de texturas, na detecção de mudanças, e no reconhecimento de padrões, explorando a capacidade de processamento das novas unidades de armazenamento denominadas computation storage.

Classificação de Texturas

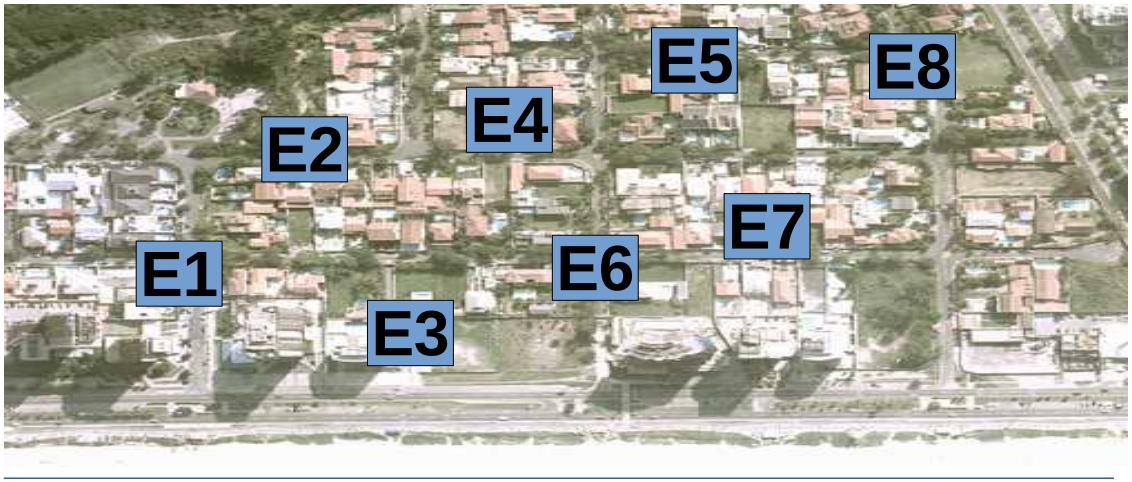
Detecção de Mudanças

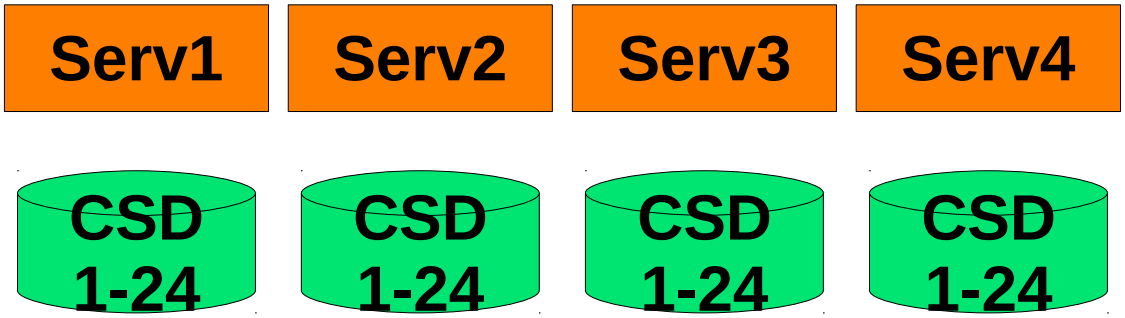
Reconhecimento de Padrões

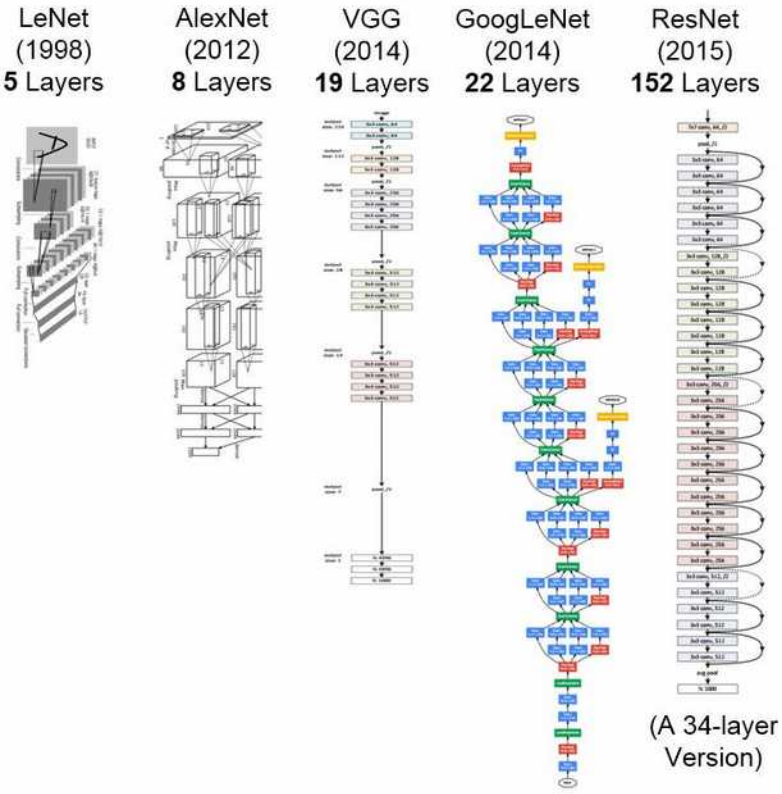
Ferramentas e Recursos: Servidor de Imagens - HORUS

Múltiplos sensores capturando imagens, e pré-classificam os dados, enviando somente as imagens que atendem determinados critérios. Os servidores recebem os dados e amazenam em unidades CSDs, que são responsáveis por completar a classificação dos dados armazenados.

Modelo de redes neurais profundas podem ser gerado nos servidores, e em seguida, carregados nos equipamentos remotos para permitir a pré-classificação dos dados. O processamento e classificação detalhado da imagem é realizado nos servidores e armazenados nas unidades CSDs.







Conceitos Iniciais: Classificação de Texturas

CLASSIFICAÇÃO DE TEXTURAS COM REDES NEURAIS

O problema de classificação de texturas em imagens é fundamental para o reconhecimento de padrões locais que se repetem em uma área maior da imagem. A identificação e classificação destes padrões locais, permite o reconhecimento de áreas da imagem que são cobertas por um mesmo padrão de textura.

Nosso objetivo e implementar um mecanismo de classificação de texturas otimizado usando Redes Neurais que permita classificar rapidamente as áreas cobertas por cada tipo de textura presente nas imagens.

Este tipo de processamento possui diversas aplicações praticas, como por exemplo, a classificação automática de diferentes tipos de solos e diferentes tipos de vegetação existentes em uma imagem de satélite.

Classificação de Texturas

Conceitos Iniciais: Detecção de Mudanças

DETECÇÃO DE MUDANÇAS COM REDES NEURAIS

O processo de detecção de mudanças em imagens possui diversas aplicações praticas, como por exemplo, o monitoramento em tempo real de áreas de segurança e a detecção automática das mudança ocorridas na vegetação nativa de uma região ao longo do tempo. Este processo vem sendo estudado de forma intensiva usando Redes Neurais, obtendo um ganho significativo de desempenho na classificação dos dados.

Detecção de Mudanças

Conceitos Iniciais: Reconhecimento de Padrões

RECONHECIMENTO DE PADRÕES EM IMAGENS COM REDES NEURAIS

O reconhecimento de padrões em imagens é um tema que vem sendo amplamente estudado e aplicado no reconhecimento de escrita, no reconhecimento facial, e etc. A figura abaixo, apresenta um exemplo de aplicação de reconhecimento de padrões em imagens.

Neste estudo, analisaremos o uso de Redes Neurais no reconhecimento de padrões em imagens, com o objetivo de implementar de forma eficiente este modelo de Redes Neurais com a expectativa de obter uma melhora significativa no treinamento do modelo e na classificação dos dados.

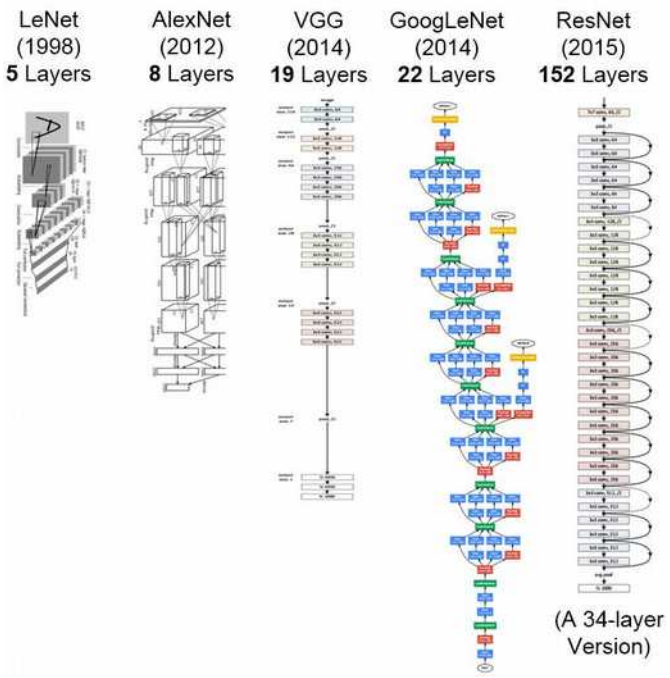
Reconhecimento de Padrões

Conceitos Iniciais:
Distributed Deep Neural Networks

DISTRIBUTED DEEP NEURAL NETWORKS (DDNN)

O aumento na capacidade de processamento das unidades de monitoramento permite que sejam realizadas classificações nos dados coletados nos dispositivo remotos, possibilitando a decisão sobre o envio ou não da imagem.

A identificação de padrões nas imagens dos equipamentos de monitoramento e supervisão, reduz em até 20x o volume dos dados trafegados pela rede.



Raspberry Pi 4 - Processador Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64bit SoC – Clock 1.5 Ghz – Memória RAM: 4GB DDR4

Jetson Nano - Arquitetura NVIDIA Maxwell com 128 NVIDIA CUDA cores - Processador Quad-core ARM Cortex-A57 MPCore - 4 GB 64 -bit LPDDR4 - 16 GB eMMC 5.1 Flash

Conceitos Iniciais: *Computation Storage*

COMPUTATION STORAGE

O conceito de computation storage vem sendo amplamente estudado, com a aplicação dos recursos de processamento diretamente na unidade de armazenamento, permitindo que diversos serviços suportados anteriormente no sistema operacional ou na camada da aplicação, estejam disponíveis na unidade de armazenamento, reduzindo o trabalho da CPU e proporcionando maior velocidade na transferência dos dados entre as unidades de armazenamento e memória.

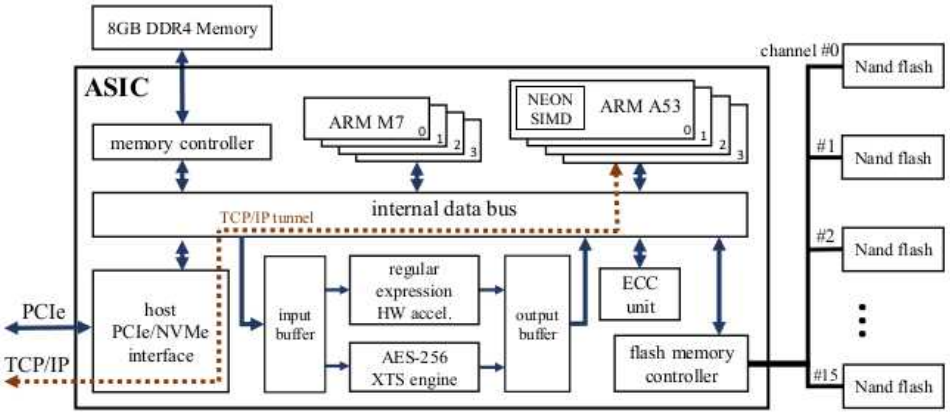
Atualmente o conceito de computation storage esta sendo largamente utilizado em unidades de armazenamento para gravar os dados com segurança e com compactação, proporcionando de forma eficiente uma maior segurança e economia de espaço de armazenamento.

Outra aplicação do conceito de computation storage está relacionada com a otimização do cache das unidades de armazenamento, na otimização de consultas a bancos de dados, e na otimização de aplicações de inteligência artificial.

O nosso estudo, envolve a aplicação do conceito de computation storage no aprendizado de maquina usando Redes Neurais para armazenamento otimizado das imagens, classificação de texturas, detecção de mudança e reconhecimento de padrões em imagens, fornecendo um modelo eficiente para ser utilizado na prática.

Arquitetura dos Equipamentos: *Computation Storage*

Neste estudo serão utilizados unidades de armazenamento SSD com capacidade de processamento de dados (*computation storage*), com o objetivo de otimizar o processamento das imagens e a busca pelas informações.



Computational Storage Device

Unidade de armazenamento SSD programável, Newport CSD, desenvolvida para sobrepor as limitações das unidades de armazenamento SSD tradicionais.

Arquitetura do *Hardware*

A arquitetura do Newport CSD foi construída em torno de uma unidade ASIC personalizada Zynq Ultra Scale + FPGA da fabricante chinesa Xlinx.

Camada de *Softwares*

Linux Ubuntu 16.04.5 LTS; com TCP/IP, mas será implementado suporte para NVMe/PCIe.

Contribuições Esperadas

DESAFIOS DE TER UM GIS EM SERVIDORES COM COMPUTATION STORAGE

Grande volume de informação composta por múltiplos arquivos de dados de grandes dimensões (>400MB) que precisam ser frequentemente processados (ex: conversão do formato, reprojeção das imagens, criação de cache com diferentes níveis de precisão, comparação e identificação de mudanças, reconhecimento de padrões e etc).

GANHOS ESPERADOS COM O USO DE *COMPUTATION STORAGE*

Distribuíndo o processamento nas unidades de armazenamento, obtemos maior agilidade nas tarefas de conversão de formato, reprojeção das imagens, criação de cache com diferentes níveis de precisão, comparação e identificação de mudanças e no reconhecimento de padrões.

CONTRIBUIÇÕES QUE PODEM SER OBTIDAS COM A PESQUISA

1. Contribuições em AI + IoT através da otimização dos processos de captura das imagens por sensores com câmeras;

2. Contribuição em *computation storage* com avaliação de processos de otimização do processamento e captura dos dados;

3. Contribuição para o GIS com um servidor de Banco de Dados de Imagens com grande capacidade de armazenamento e processamento de imagens;

RESULTADOS ESPERADOS

1. Maior desempenho;

2. Melhor eficiência energética;

Ambiente de Experimento Benchmark

PROBLEMA:

Em reunião com a equipe de cartografia do IBGE eles informaram que o maior problema que eles possuem está no armazenamento e processamento das imagens de 480.000 zonas censitárias existentes no Brasil.

ESTUDOS REALIZADOS:

Desta forma, a equipe do IBGE iniciou um estudo com imagens de satélites comerciais com precisão de 8 metros para serem aplicadas em futuros levantamentos de campo.

Estas imagens serão comparadas com imagens e elementos vetorias obtidos em levantamentos de anos anteriores, para identificação mudanças e padrões nas imagens.

I. Volume grande de imagens;

II. Imagens Multispectral e True Color com resolução de 8m

- Cada cena possui 11 imagens em tons de cinza, representando as diferentes faixas de frequencia e 1 imagem true color;

- Espaço estimado: 4.678 Giga Bytes (5 TB) somente as imagens deste ano;

Ambiente de Experimento Benchmark

CONSULTA E PROCESSAMENTO

O experimento inicial foi realizado com apenas 1 consulta que retorna até 4,7% dos dados totais, cerca de 190 imagens. Após a seleção das imagens, as imagens são convertidas de GeoTIFF para JPEG, reprojetadas de UTM para coordenadas geográficas, e simplificadas para reduzir o tamanho a 1/4 do tamanho original.

CONFIGURAÇÃO DO SERVIDOR

Os experimentos foram realizados em um computador basico com 2 núcleos e simulando as unidades CSDs com as seguintes configurações:

(a) Processamento em 1 único servidor hospedeiro;

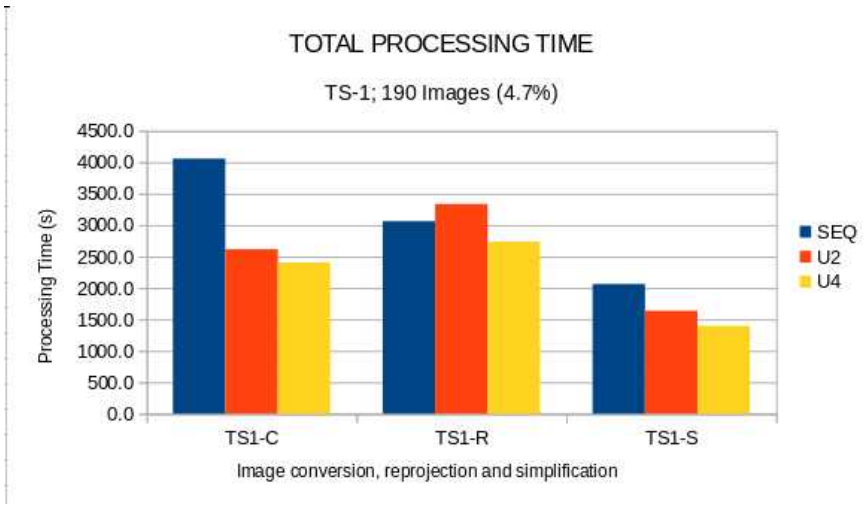
(b) Pocessamento com 2 tarefas simulando 2 unidades CSDs;

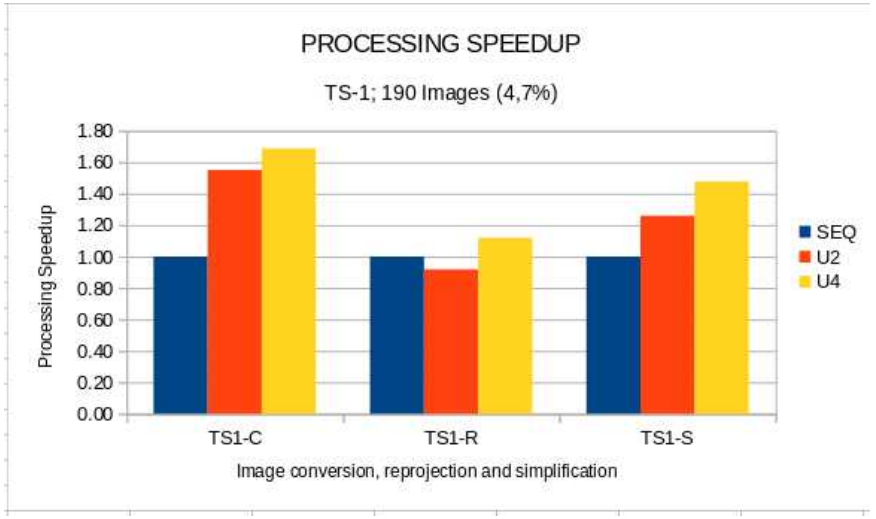
(d) Processamento com 4 tarefas simulando 4 unidades CSDs;

OBSERVAÇÕES

Os experimentos NÃO levaram em consideração o tempo de transmissão dos dados;

Análise dos Resultados





	TOTAL PROCESSING TIME (s)		
	SEQ	**U2**	**U4**
TS1-C	4057.2	2616.2	2403.8
TS1-R	3062.1	3335.7	2736.1
TS1-S	2062.8	1637.7	1396.4

Conclusões

Nossos primeiros experimentos foram realizados em um computador basico com 2 núcleos de processamento e sem unidades CSDs instaladas.

Observamos um ganho em *speedup* de 1,69 na conversão das imagens, 1,20 na reprojeção de imagens, e 1,48 na simplificação das imagens simulando o uso de 4 unidades CSDs.

Nos experimentos observamos um ganho potencial com a separação do processamento entre diferentes unidades CSDs, demonstrando o ganho que pode ser obtido com o uso de unidades de armazenamento com *computation storage*.

Para prosseguir com os estudos, será implementado as consultas similares as apresentadas no artigo “POLARDB Meets Computational Storage: Efficiently Support Analytic Workloads in Cloud-Native Relational Database”.

PESQUISA PARA A TESE DOUTORADO
HORUS IMAGE SERVER v2.0 – NOVA FASE

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@globo.com
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ

PROJETO HORUS v2.0 – NOVA FASE

1. Implementação dos Classificadores

libchange_mod.so – classificador para detecção de mudanças, usando o posicionamento geográfico da região da imagem como referência.

libfeat_mod.so – classificador dos elementos presentes nas imagens usando rede neural profunda (Deep Learning).

libpat_mod.so – classificador dos padrões de preenchimento identificados nas imagens, usando rede neural profunda (Deep Learning).

2. Aumentar Paralelismo nas Unidades CSDs

Modelo Atual: Mestre – Escravo.

Apresentar Novos Modelos:

- Avaliar distribuição dos dados à nível de bloco;

- Avaliar dois níveis de processamento: (1) seleção dos blocos distribuídos pelas unidades de armazenamento; (2) processamento dos dados; (3) transmissão do resultado para a máquina hospedeira;

IMPLEMENTAÇÃO DOS CLASSIFICADORES: CHANGE DETECTION

CHANGE DETECTION – DATASETS 2014 (http://changedetection.net/)

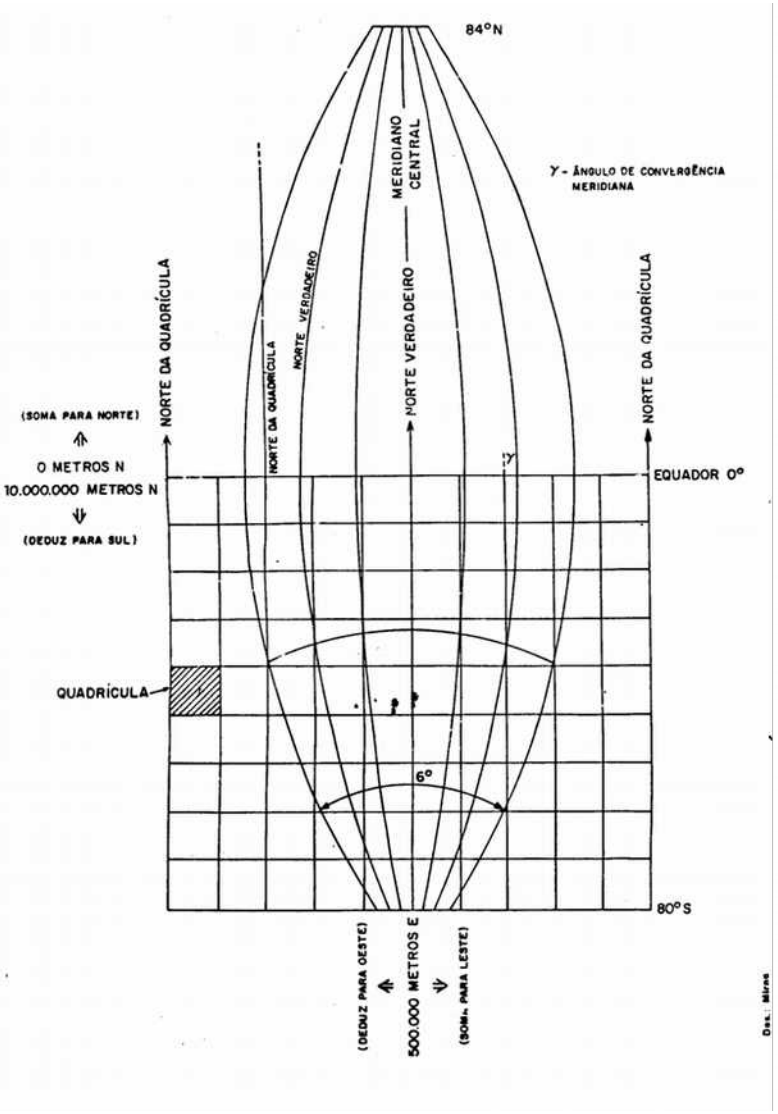
This dataset contains 11 video categories with 4 to 6 videos sequences in each category

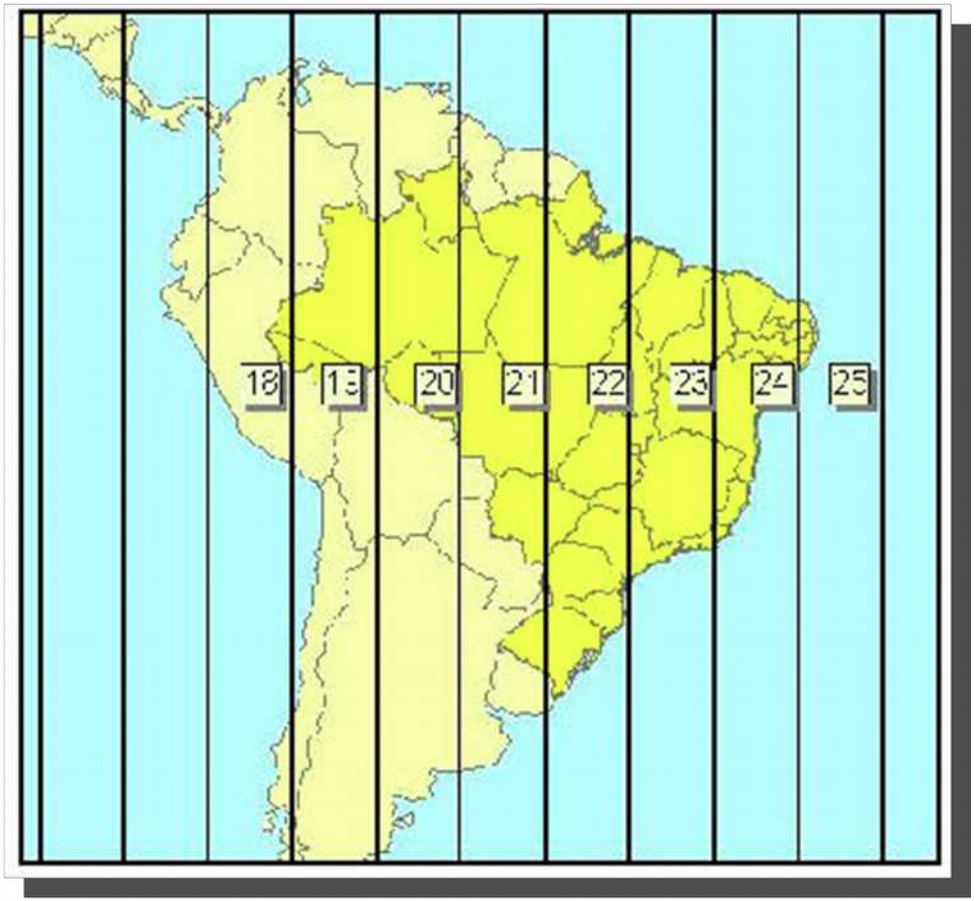
- Each individual video file (.zip or .7z) can be downloaded separately.
- Alternatively, all videos files within one category can be downloaded as a single .zip or .7z file
- Each video file when uncompressed becomes a directory which contains the following:
  - a sub-directory named "input" containing a separate JPEG file for each frame of the input video
  - a sub-directory named "groundtruth" containing a separate BMP file for each frame of the groundtruth
- The groundtruth images contain 5 labels namely
  - 0 : Static
  - 50 : Hard shadow
  - 85 : Outside region of interest
  - 170 : Unknown motion
  - 255 : Motion

Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar;
CDnet 2014: An Expanded Change Detection Benchmark Dataset,
in Proc. IEEE Workshop on Change Detection (CDW-2014) at CVPR-2014, pp. 387-394. 2014

IMPLEMENTAÇÃO DOS CLASSIFICADORES: CHANGE DETECTION

IMPLEMENTAÇÃO DOS CLASSIFICADORES: CHANGE DETECTION





IMPLEMENTAÇÃO DOS CLASSIFICADORES: CHANGE DETECTION

CLASSIFICADOR – CHANGE DETECTION (até 12/06/21)

- Imagens recortadas em regiões: 5.000m x 5.000m.

- Recorte considera uma origem comum para o mesmo sistema de coordenadas de cada imagem.

  . UTM-23S / SIRGAS 2000

- Cada recorte da imagem será comparado com versões de anos anteriores da mesma região, usando um modelo de aprendizado que irá classificar o que representa background dos novos objetos que surgiram.

- As regiões que limitam os novos objetos identificados serão armazenadas no banco de dados e poderão ser consultadas.

Aumentar Paralelismo nas Unidades CSDs

Modelo Atual:

- Mestre – Escravo, com distribuição de arquivo.

Nova Proposta (até 26/06/21):

- Mestre – Escravo, com distribuição de blocos.

- Distribuição dos dados à nível de bloco usando mecanismo similar ao sistema de arquivos distribuído implementado pelo Hadoop (HDFS);

  . As unidades CSDs serão agrupadas em unidades lógicas

  com 1 ou mais unidades CSDs.

  . Cada grupo terá 1 nó de entrada das solicitações, 1 ou mais nós de diretório de blocos, e 1 ou mais nós de armazenamento e processamento dos dados.

Apoio à Pesquisa Acadêmica

1. RECURSOS PARA A PESQUISA ACADÊMICA:

1.1. ACESSO A UM “SUBCONJUNTO” DAS IMAGENS DE SATÉLITES QUE SERÃO USADAS NO CENSO 2021/2022?

PESQUISA PARA A TESE DOUTORADO HORUS IMAGE SERVER v2.2

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ
Registro: 2000103581 CREA-RJ

Ambiente de Experimento Consulta e Processamento

#1. PREPARAÇÃO DO BANCO DE DADOS:

Para a execução do Benchmark, recriamos a estrutura do banco de dados e efetuamos uma carga inicial com os dados das imagens pré-classificados, mas sem enviar os arquivos de imagens para o servidor.

Neste processo são construidas as tabelas de usuários, as sequencias, as tabelas de dados alfanuméricos, e as tabelas de imagens sem o envio do arquivo paro o servidor.

#2. EXECUÇÃO DO BENCHMARK:

O experimento foi realizado usando uma sequencia de processamento com 3 estágios de execução.

(i) MkDir (MKDIR) – Criação da estrutura de diretórios para armazenamento das imagens no servidor.

(ii) Upload (UPLOAD) – Leitura dos arquivos de imagens e envio dos arquivos para o servidor.

(iii) Download e Processamento (PROC) – Seleção dos dados do banco de dados, retornando cerca de 4,7% dos dados totais, com cerca de 190 imagens.

Após a seleção das imagens, as imagens são convertidas de GeoTIFF para JPEG usando o módulo “libconvert_mod.so”, reprojetadas de UTM para coordenadas geográficas com o módulo “libreproj_mod.so”, e simplificadas para reduzir o tamanho a 1/4 do tamanho original usando o módulo “libsimpl_mod.so”.

#3. IMAGENS USADAS NESTE EXPERIMENTO:

Neste experimento, foram usadas as imagens da base de levantamento do IBGE da área do município do Rio de Janeiro, em escala de 1:25.000, onde cada píxel representa uma área de 25 cm x 25 cm. Deste modo, um carro convencional com tamanho de 5 m x 2 m é representado por 20 pixels x 8 pixels em imagens com este nível de definição.

As imagens usadas no experimento possuem tamanho compactado de até 100 MB, tamanho descompactado superior a 400 MB, e tamanho em memória, quando em processamento, de até 2 GB.

Ambiente de Experimento
Exemplo de Imagem

Ambiente de Experimento
Configuração do Servidor

#1. CONFIGURAÇÃO DO SERVIDOR:

Os experimentos foram realizados em um notebook básico com processador Intel i5, com 2 núcleos de processamento, 8 GB de Memória RAM e 1 TB de espaço em disco.

Neste equipamento, usamos múltiplos processos em execução na mesma máquina, competindo por recursos, e simulando o processamento de 1 servidor hospedeiro com até 4 unidades CSDs.

#2. NOTA:

(a) O equipamento usado possui POUCOS RECURSOS para processamento das imagens, e ocorreram muitas falhas por falta de recursos do computador durante a realização dos experimentos.

(b) Os experimentos NÃO levaram em consideração o tempo de transmissão dos dados.

Análise dos Resultados
Execution Time Distribution for Each Stage

RESULTADOS

		MKDIR	UPLOAD	PROC
1CSD	DS-ST-1C	0.0009	3323.4447	11066.9752
	SM-ST-1C	0.0008	3453.3635	12179.4899
	DS-MT-1C	0.0009	3474.8353	11421.6605
	SM-MT-1C	0.0009	3123.2359	10797.4683
2CSDs	DS-ST-2C	0.0008	2616.5962	7655.1958
	SM-ST-2C	0.0011	2137.3474	6862.8875
	DS-MT-2C	0.0012	2202.4704	7402.6248
	SM-MT-2C	0.0005	1920.3308	6879.6718
4CSDs	DS-ST-4C	0.0041	1201.7843	5003.0896
	SM-ST-4C	0.0026	1305.6174	5180.0557
	DS-MT-4C	0.0023	1264.2624	5276.3989
	SM-MT-4C	0.0024	1166.2446	5018.9659

#1. O Benchmark usado contém 3 estágios de execução:

(i) MKDIR: Estágio de criação dos disretórios;

(ii) UPLOAD: Estágio de envio dos arquivos de imagens para o servidor; e

(iii) PROC: Estágio de processamento das imagens.

#2. Observando o gráfico de distribuição do tempo de execução dos estágios, podemos observar que em média 78% do tempo de execução é consumido no estágio de processamento das imagens, e que 22% do tempo de execução é consumido no estágio de envio das imagens para o servidor. O tempo de execução do estágio MKDIR representa menos de 0,00001% do tempo de execução, e será desprezado nas futuras análises.

Análise dos Resultados
Comparing All Configurations

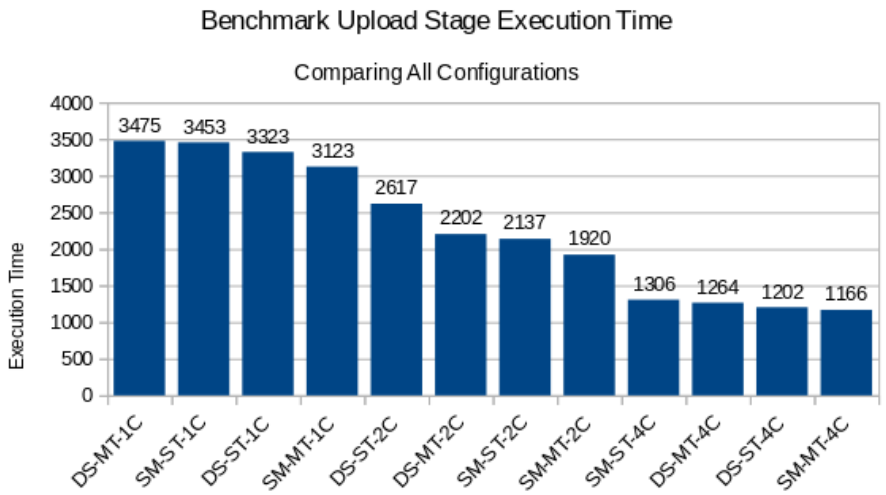
#1. COMPARE ALL CONFIGURATIONS

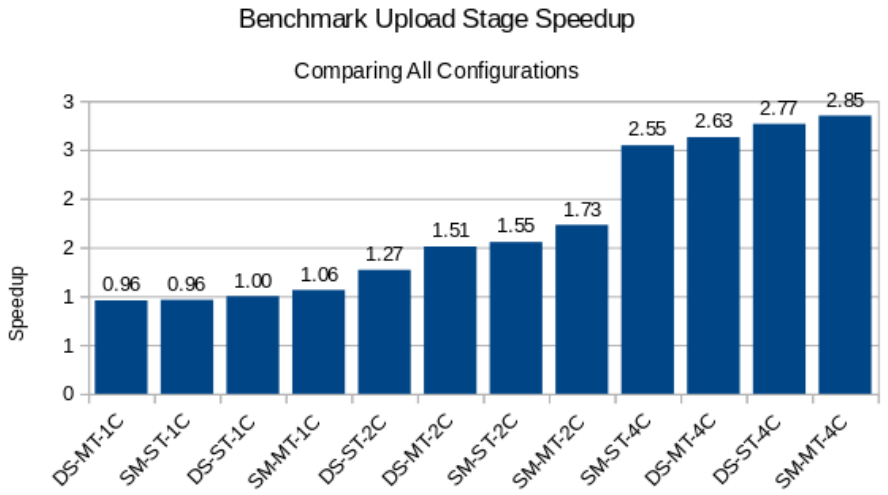
	UPLOAD	SPEEDUP
DS-MT-1C	3474.8353	0.9564
SM-ST-1C	3453.3635	0.9624
DS-ST-1C	3323.4447	1.0000
SM-MT-1C	3123.2359	1.0641
DS-ST-2C	2616.5962	1.2701
DS-MT-2C	2202.4704	1.5090
SM-ST-2C	2137.3474	1.5549
SM-MT-2C	1920.3308	1.7307
SM-ST-4C	1305.6174	2.5455
DS-MT-4C	1264.2624	2.6288
DS-ST-4C	1201.7843	2.7654
SM-MT-4C	1166.2446	2.8497

	PROC	SPEEDUP
SM-ST-1C	12179.48989	0.9087
DS-MT-1C	11421.66055	0.9689
DS-ST-1C	11066.97516	1.0000
SM-MT-1C	10797.46826	1.0250
DS-ST-2C	7655.195835	1.4457
DS-MT-2C	7402.624804	1.4950
SM-MT-2C	6879.671804	1.6086
SM-ST-2C	6862.887451	1.6126
DS-MT-4C	5276.398878	2.0974
SM-ST-4C	5180.05565	2.1365
SM-MT-4C	5018.965917	2.2050
DS-ST-4C	5003.089617	2.2120

	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-MT-1C	14896.4967	0.9660
DS-ST-1C	14390.4208	1.0000
SM-MT-1C	13920.7050	1.0337
DS-ST-2C	10271.7929	1.4010
DS-MT-2C	9605.0964	1.4982
SM-ST-2C	9000.2360	1.5989
SM-MT-2C	8800.0032	1.6353
DS-MT-4C	6540.6636	2.2001
SM-ST-4C	6485.6756	2.2188
DS-ST-4C	6204.8780	2.3192
SM-MT-4C	6185.2129	2.3266

Análise dos Resultados
Comparing All Configurations – UPLOAD

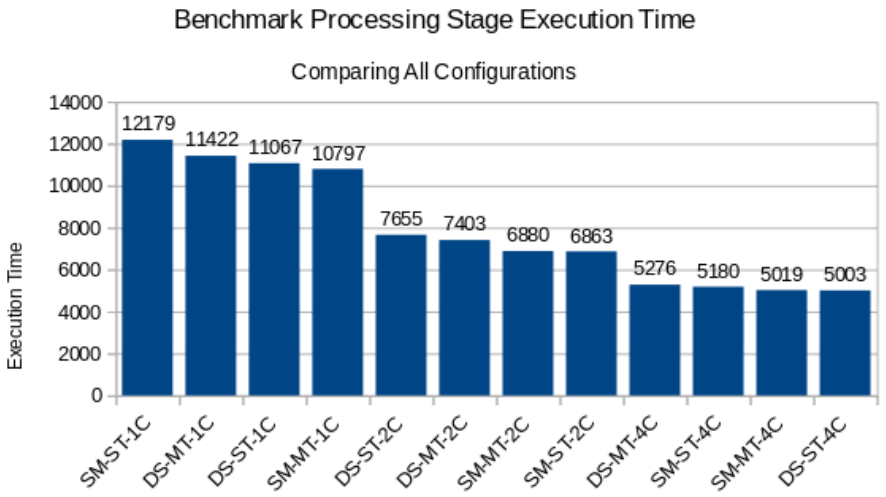


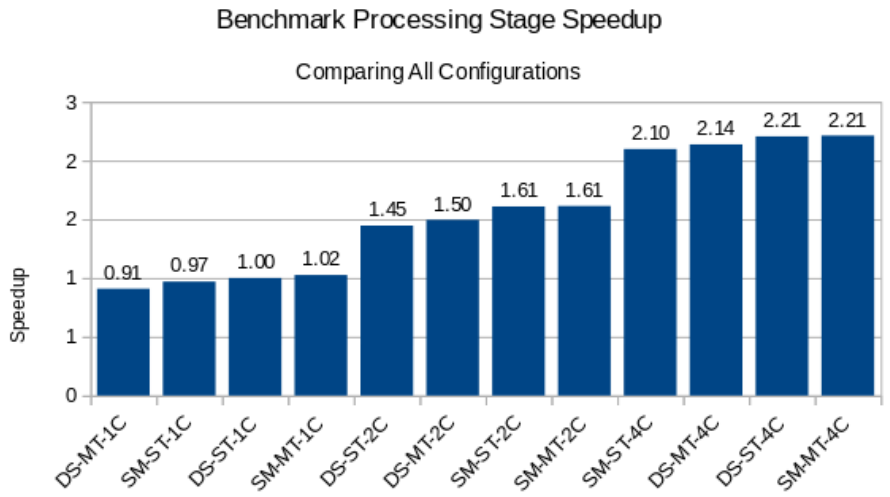


#1. Analisando o estágio de envio das imagens ao servidor, UPLOAD, podemos verificar que este estágio obtém um ganho em Speedup de até 2,85 com o aumento do paralelismo proporcionado pelo acréscimo de unidades CSDs e com o uso de processamento MULTI THREAD (MT) nas operações de leitura e escrita dos arquivo.

#2. Este estágio de execução, NÃO obtém ganhos com o compartilhamento de memória usando SHARED MEMORY (SM) entre o servidor e os módulos de processamento.

Análise dos Resultados
Comparing All Configurations – PROC

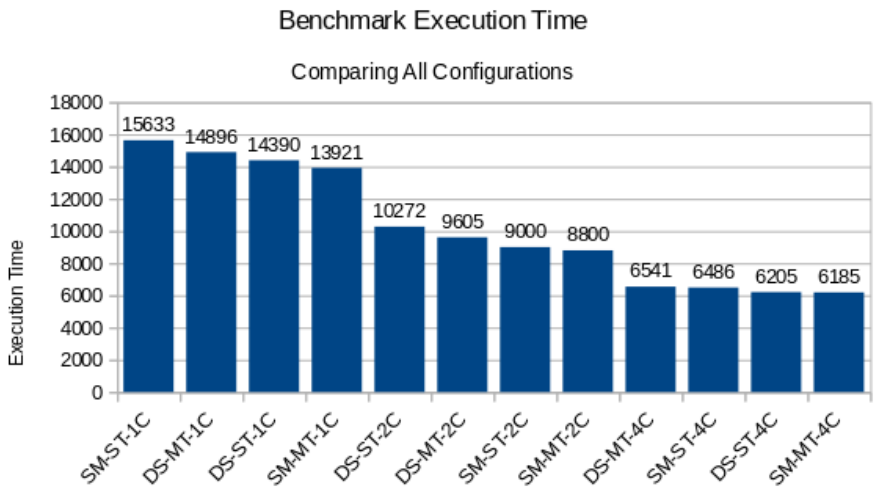


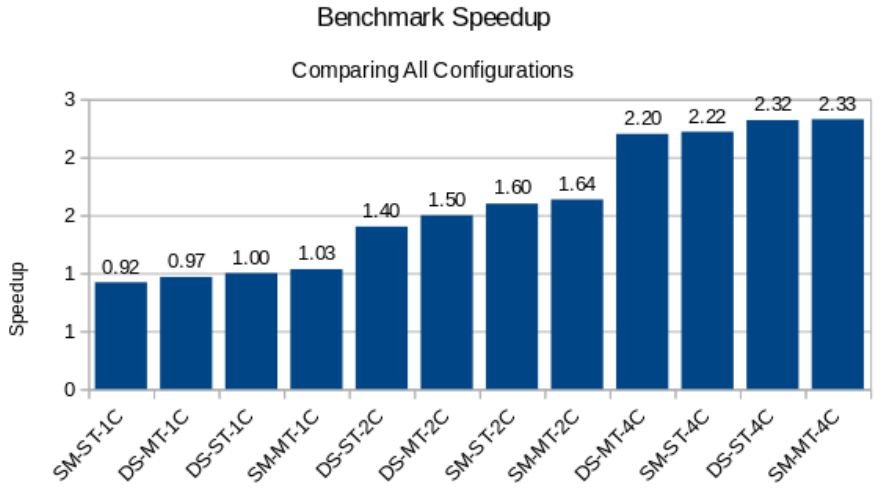


#1. O estágio de seleção, download e processamento das imagens, PROC, obtém um ganho em Speedup de até 2,21 com o aumento do paralelismo através do acréscimo de unidades CSDs, usando processamento MULTI THREAD (MT) nas operações de acesso ao disco, e local de armazenamento do arquivo de imagem temporário em memória compartilhada usando SHARED MEMORY (SM).

#2. O aumento no número de unidades CSDs gerou um aumento na concorrencia entre os processos por recursos de processamento e memória, e o melhor desempenho obtido com 4 unidades CSDs é a configuração que efetua armazenamento em disco local, DATASTORE (DS), e uma linha de execução para acesso ao disco, SINGLE THREAD (ST).

Análise dos Resultados
Comparing All Configurations





#1. Quando avaliamos o Tempo de Execução Total, considerando a execução de todos os estágios, MKDIR, UPLOAD e PROC, observamos que com o acréscimo de unidades CSDs e o uso de SHARED MEMORY (SM) para comunicação entre o servidor e os módulos de processamento, obtemos um ganho de Speedup de até 2,33 usando 4 unidades CSDs.

#2. A simulação de 4 unidades CSDs apresentou um baixo desempenho no estágio de processamento, PROC, que foi compensado pelo excelente resultado obtido no estágio de envio das imagens ao servidor, UPLOAD.

Análise dos Resultados
Comparing Datastore x Shared Memory

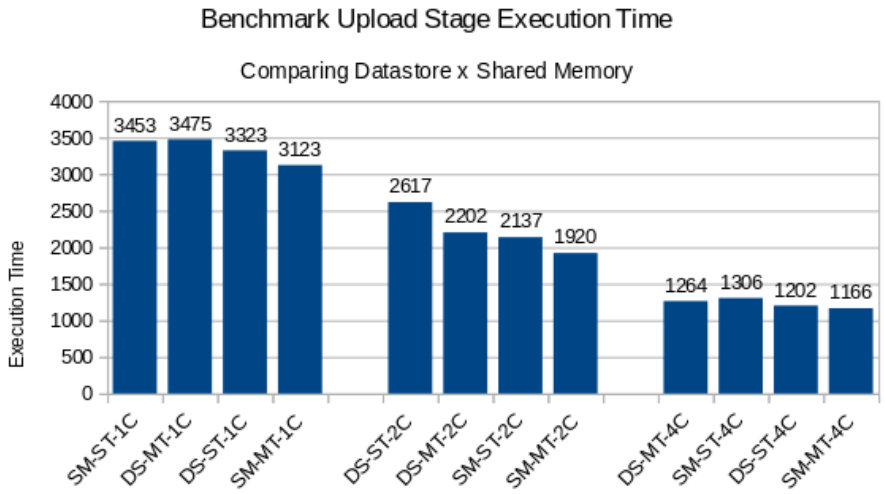
#2. COMPARE DATASTORE x SHMEM

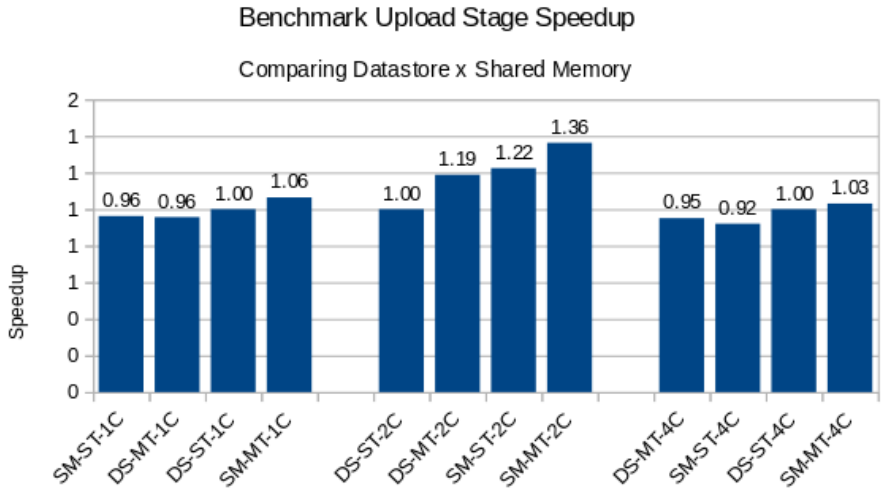
	UPLOAD	SPEEDUP
SM-ST-1C	3453.3635	0.9624
DS-MT-1C	3474.8353	0.9564
DS-ST-1C	3323.4447	1.0000
SM-MT-1C	3123.2359	1.0641
DS-ST-2C	2616.5962	1.0000
DS-MT-2C	2202.4704	1.1880
SM-ST-2C	2137.3474	1.2242
SM-MT-2C	1920.3308	1.3626
DS-MT-4C	1264.2624	0.9506
SM-ST-4C	1305.6174	0.9205
DS-ST-4C	1201.7843	1.0000
SM-MT-4C	1166.2446	1.0305

	PROC	SPEEDUP
SM-ST-1C	12179.48989	0.9087
DS-MT-1C	11421.66055	0.9689
DS-ST-1C	11066.97516	1.0000
SM-MT-1C	10797.46826	1.0250
DS-ST-2C	7655.195835	1.0000
DS-MT-2C	7402.624804	1.0341
SM-ST-2C	6862.887451	1.1154
SM-MT-2C	6879.671804	1.1127
DS-MT-4C	5276.398878	0.9482
SM-ST-4C	5180.05565	0.9658
DS-ST-4C	5003.089617	1.0000
SM-MT-4C	5018.965917	0.9968

	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-MT-1C	14896.4967	0.9660
DS-ST-1C	14390.4208	1.0000
SM-MT-1C	13920.7050	1.0337
DS-ST-2C	10271.7929	1.0000
DS-MT-2C	9605.0964	1.0694
SM-ST-2C	9000.2360	1.1413
SM-MT-2C	8800.0032	1.1672
DS-MT-4C	6540.6636	0.9487
SM-ST-4C	6485.6756	0.9567
DS-ST-4C	6204.8780	1.0000
SM-MT-4C	6185.2129	1.0032

Análise dos Resultados
Comparing Datastore x SHMEM – UPLOAD

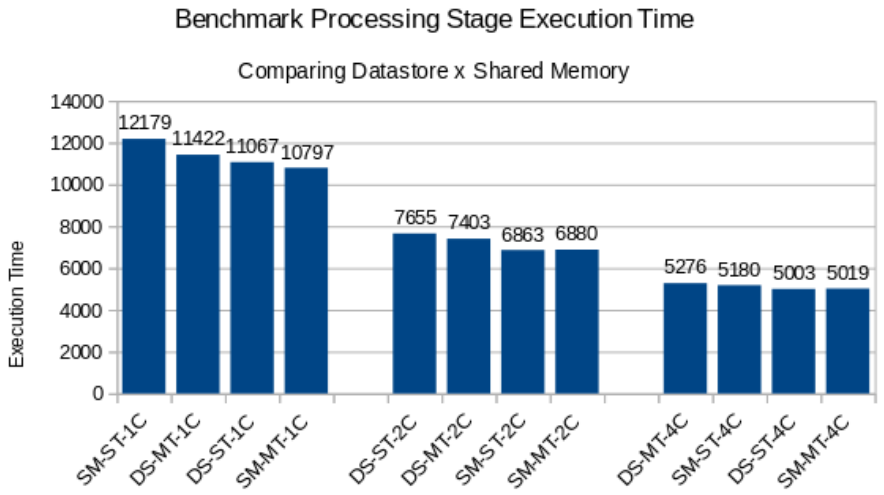


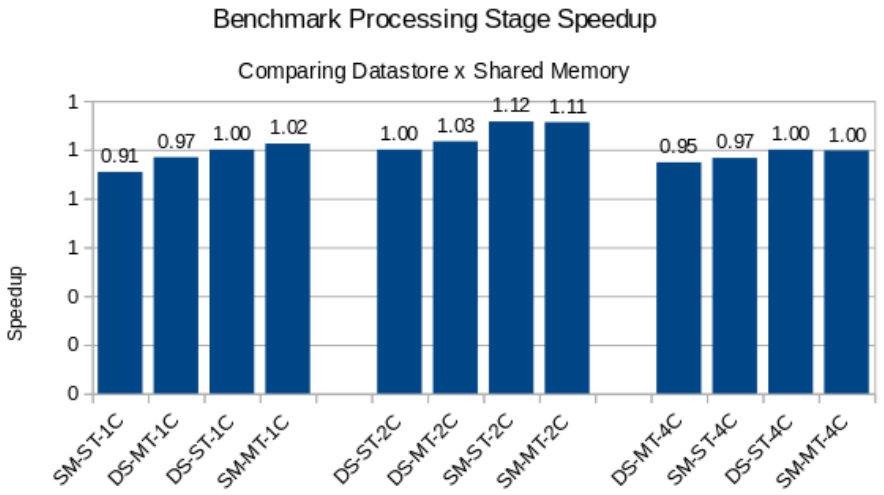


#1. Podemos observar nos gráficos que quando comparamos o armazenamento dos arquivos temporários para processamento em disco local, DATASTORE (DS), com o compartilhamento de memória, SHARED MEMORY (SM), a variação no ganho de Speedup é pequena.

#2. Isto ocorre, porque o estágio de UPLOAD não faz uso deste recurso, e a variação em Speedup verificada no estágio de UPLOAD usando 2 unidades CSDs, pode ser justificada pelo ganho proporcionado pelo uso de múltiplas linhas de execução para leitura e escrita no disco, MULTI THREAD (MT).

Análise dos Resultados
Comparing Datastore x SHMEM – PROC





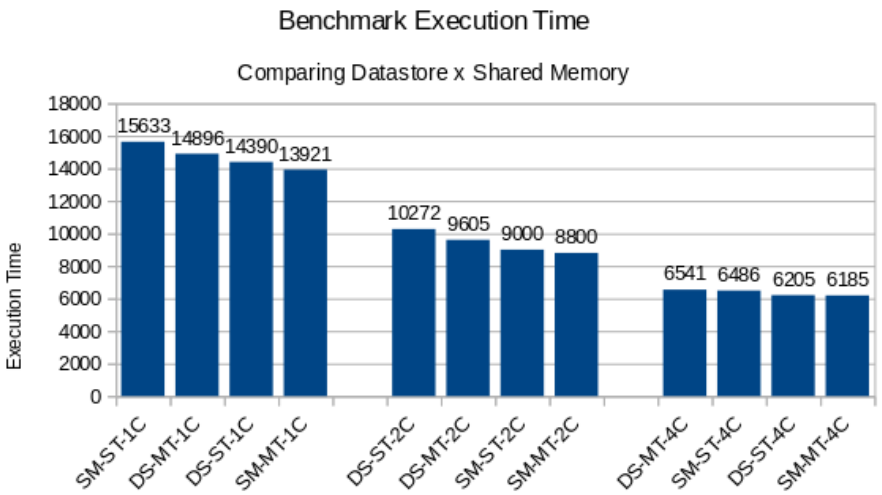
#1. No estágio de processamento das imagens, PROC, o armazenamento dos arquivos temporários pode se beneficiar bastante com o uso de memória compartilhada, SHARED MEMORY (SM), entre o servidor e os módulos de processamento.

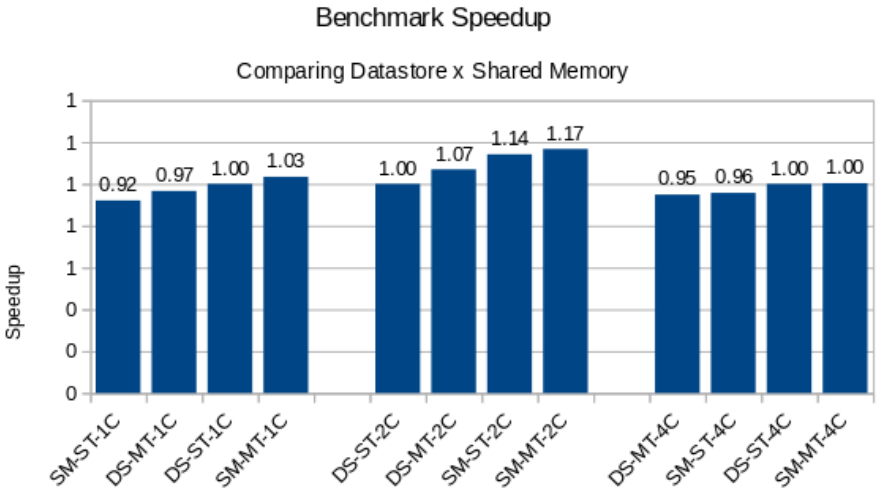
#2. Analisando os gráficos, observamos que usando 2 unidades CSDs obtemos ganhos de até 1,12 em Speedup em relação a configuração de referência para o mesmo número de unidades CSDs.

#3. Usando 4 unidades CSDs o desempenho é inferior a configuração de referência, porque a concorrência pelos recursos de processamento e memória do computador, com 8 GB RAM, usado neste experimento aumenta significativamente.

#4. Usando somente 1 unidade CSD com o compartilhamento de arquivos em memória, SHARED MEMORY (SM), e múltiplas linhas de leitura e escrita de dados, MULTI THREADS (MT), o ganho em Speedup é de somente 1,03.

Análise dos Resultados
Comparing Datastore x SHMEM





#1. Analisando os gráficos, observamos que o armazenamento dos arquivos temporários pode se beneficiar bastante com o uso de memória compartilhada, SHARED MEMORY (SM), entre o servidor e os módulos de processamento. Entretanto, devido a capacidade limitada de memória do computador usado na realização destes experimentos, o desempenho total do sistema em um ambiente simulado com 4 unidades CSDs apresentou pouco ganho quando comparado com o armazenamento em disco local.

#2. O baixo resultado obtido com usando 4 unidades CSDs ocorreu por falta de recursos no equipamento usado neste experimento (8 GB RAM). Desta forma, com o acréscimo de processos, a concorrência pelo uso de memória se torna grande e o sistema operacional faz uso intenso da área de swap de disco, prejudicando o desempenho total do sistema.

Análise dos Resultados
Comparing Single Thread x Multi Thread

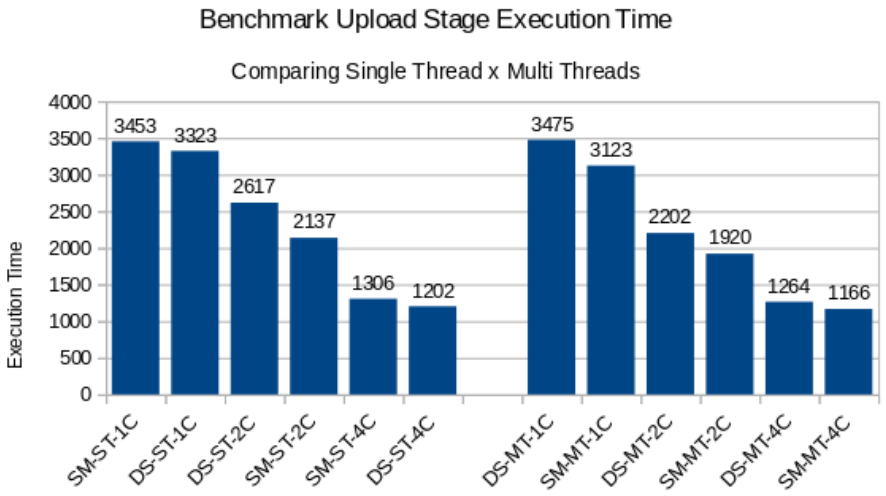
#3. COMPARE SINGLE THREAD x MULTI THREADS

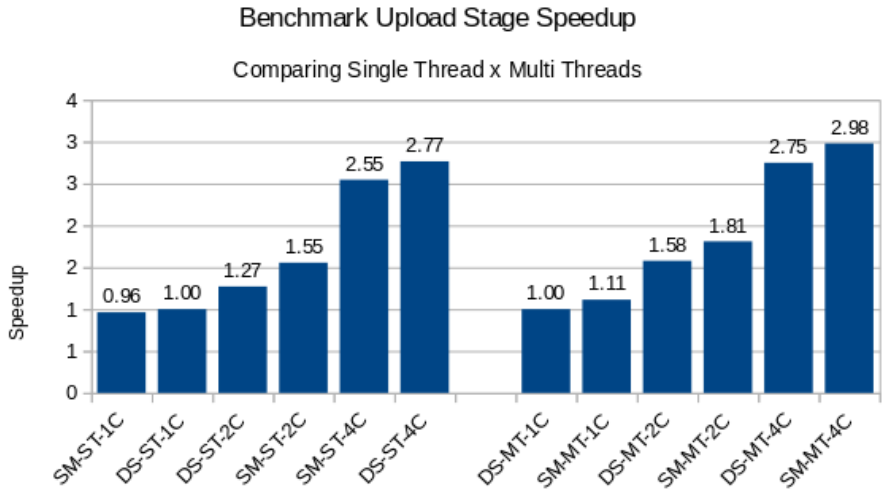
	UPLOAD	SPEEDUP
SM-ST-1C	3453.3635	0.9624
DS-ST-1C	3323.4447	1.0000
DS-ST-2C	2616.5962	1.2701
SM-ST-2C	2137.3474	1.5549
SM-ST-4C	1305.6174	2.5455
DS-ST-4C	1201.7843	2.7654
DS-MT-1C	3474.8353	1.0000
SM-MT-1C	3123.2359	1.1126
DS-MT-2C	2202.4704	1.5777
SM-MT-2C	1920.3308	1.8095
DS-MT-4C	1264.2624	2.7485
SM-MT-4C	1166.2446	2.9795

	PROC	SPEEDUP
SM-ST-1C	12179.4899	0.9087
DS-ST-1C	11066.9752	1.0000
DS-ST-2C	7655.1958	1.4457
SM-ST-2C	6862.8875	1.6126
SM-ST-4C	5180.0557	2.1365
DS-ST-4C	5003.0896	2.2120
DS-MT-1C	11421.6605	1.0000
SM-MT-1C	10797.4683	1.0578
DS-MT-2C	7402.6248	1.5429
SM-MT-2C	6879.6718	1.6602
DS-MT-4C	5276.3989	2.1647
SM-MT-4C	5018.9659	2.2757

	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-ST-1C	14390.4208	1.0000
DS-ST-2C	10271.7929	1.4010
SM-ST-2C	9000.2360	1.5989
SM-ST-4C	6485.6756	2.2188
DS-ST-4C	6204.8780	2.3192
DS-MT-1C	14896.4967	1.0000
SM-MT-1C	13920.7050	1.0701
DS-MT-2C	9605.0964	1.5509
SM-MT-2C	8800.0032	1.6928
DS-MT-4C	6540.6636	2.2775
SM-MT-4C	6185.2129	2.4084

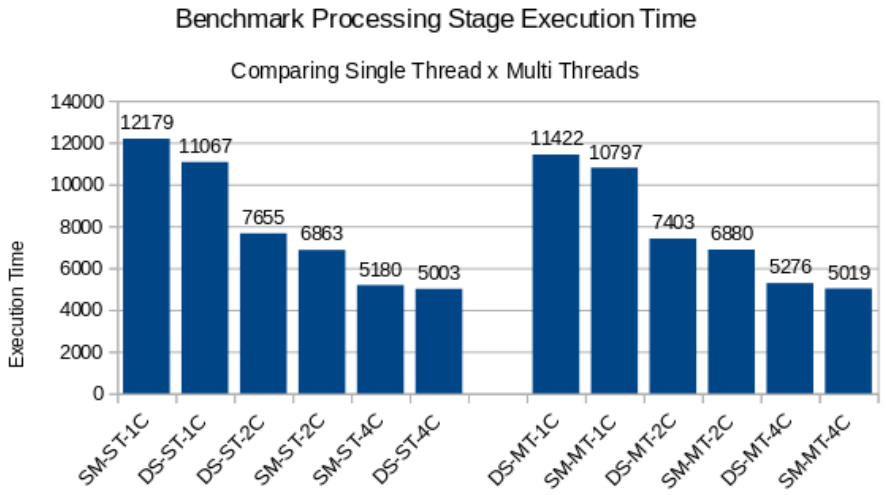
Análise dos Resultados
Comparing Single Thread x Multi Thread – UPLOAD

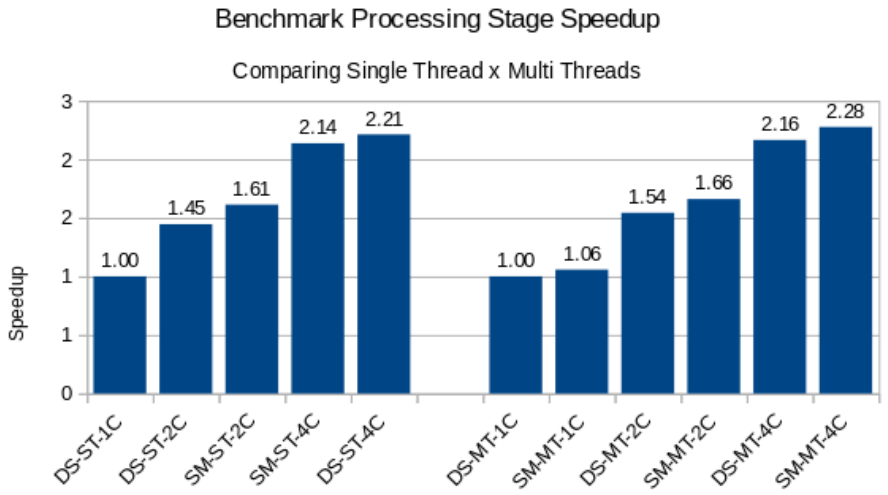




#1. Comparando a execução do estágio de envio de imagens para o servidor, UPLOAD, usando uma única linha de execução para leitura e escrita em disco, SINGLE THREAD (ST), com a utilização de múltiplas linhas de execução, MULTI THREADS (MT), verificamos a execução do Benchmark com múltiplas linhas de execução, MULTI THREADS (MT) apresentou sempre um bom resultado em relação ao uso de uma única linha de execução, SINGLE THREAD (ST), neste estágio.

Análise dos Resultados
Comparing Single Thread x Multi Thread – PROC

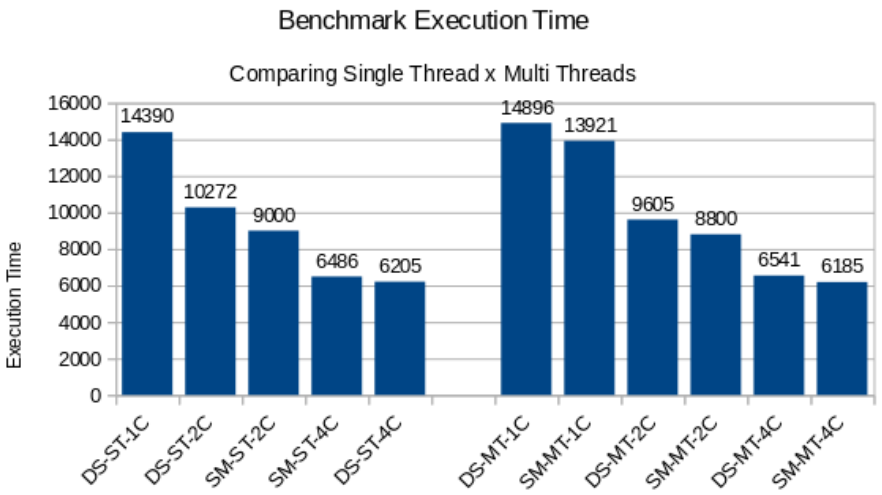


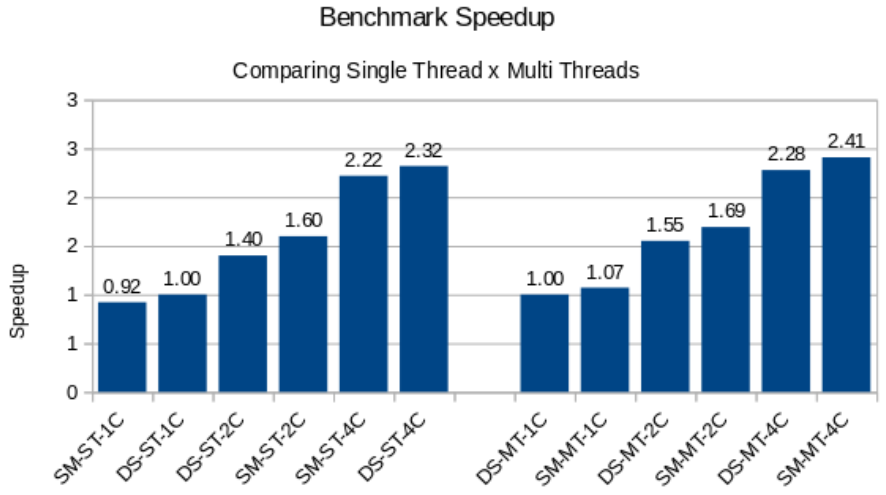


#1. O uso mútiplas linhas de execução para leitura e escrita no disco, MULTI THREADS (MT), apresentou um bom resultado em relação ao uso de uma única linha de execução, SINGLE THREAD (ST), para o estágio de processamento das imagens do Benchmark, PROC, usando até 2 unidades CSDs.

#2. Para uma configuração do Benchmark, simulando 4 unidades CSDs, o desempenho obtido foi melhor usando uma única linha de execução para leitura e escrita no disco, e os 4 processos simulando as unidades CSDs.

Análise dos Resultados
Comparing Single Thread x Multi Thread





#1. Analisando os resultados obtidos considerando os 3 estágios de execução do Benchmark, MKDIR, UPLOAD e PROC, com uma configuração com uma única linha de execução, SINGLE THREAD (ST), com o uso de múltiplas linhas de execução, MULTI THREADS (MT), verificamos que o uso de múltiplas linhas de execução para leitura e escrita em disco apresentou sempre um bom resultado quando comparamos o tempo total de execução do Benchmark.

#2. O ganho de desempenho obtido usando 4 unidades CSDs, foi proporcionado pelo bom resultado do estágio de UPLOAD, que efetua o envio de imagens para o servidor.

Análise dos Resultados
Changing Number of CSDs Units

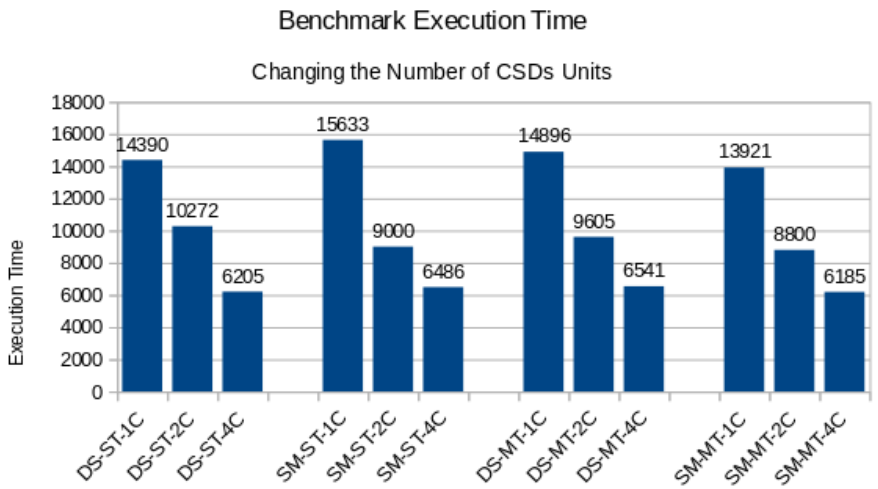
#4. CHANGE THE NUMBER OF CSDs UNITS

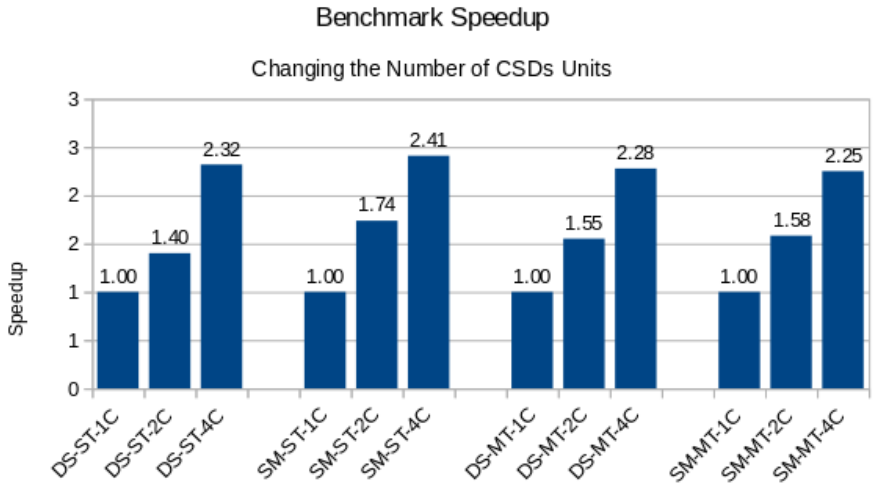
	UPLOAD	SPEEDUP
DS-ST-1C	3323.4447	1.0000
DS-ST-2C	2616.5962	1.2701
DS-ST-4C	1201.7843	2.7654
SM-ST-1C	3453.3635	1.0000
SM-ST-2C	2137.3474	1.6157
SM-ST-4C	1305.6174	2.6450
DS-MT-1C	3474.8353	1.0000
DS-MT-2C	2202.4704	1.5777
DS-MT-4C	1264.2624	2.7485
SM-MT-1C	3123.2359	1.0000
SM-MT-2C	1920.3308	1.6264
SM-MT-4C	1166.2446	2.6780

	PROC	SPEEDUP
DS-ST-1C	11066.9752	1.0000
DS-ST-2C	7655.1958	1.4457
DS-ST-4C	5003.0896	2.2120
SM-ST-1C	12179.4899	1.0000
SM-ST-2C	6862.8875	1.7747
SM-ST-4C	5180.0557	2.3512
DS-MT-1C	11421.6605	1.0000
DS-MT-2C	7402.6248	1.5429
DS-MT-4C	5276.3989	2.1647
SM-MT-1C	10797.4683	1.0000
SM-MT-2C	6879.6718	1.5695
SM-MT-4C	5018.9659	2.1513

	TEMPO TOTAL	SPEEDUP
DS-ST-1C	14390.4208	1.0000
DS-ST-2C	10271.7929	1.4010
DS-ST-4C	6204.8780	2.3192
SM-ST-1C	15632.8541	1.0000
SM-ST-2C	9000.2360	1.7369
SM-ST-4C	6485.6756	2.4104
DS-MT-1C	14896.4967	1.0000
DS-MT-2C	9605.0964	1.5509
DS-MT-4C	6540.6636	2.2775
SM-MT-1C	13920.7050	1.0000
SM-MT-2C	8800.0032	1.5819
SM-MT-4C	6185.2129	2.2506

Análise dos Resultados
Changing Number of CSDs Units





#1. O acréscimo de unidades CSDs e o aumento do paralelismo com a implementação de MULTI THREADS (MT) no acesso de leitura e escrita do disco, apresentou ganhos significativos de Speedup que variaram de 1,57 à 1,63 usando 2 unidades CSDs, e entre 2,68 à 2,75 usando 4 unidades CSDs.

#2. O ganho em Speedup usando 4 unidades CSDs e com a implementação de leitura e escrita em disco usando SINGLE THREAD (ST), obteve desempenho superior a implementação MULTI THREAD (MT) devido a quantidade limitada de memória do computador usado neste experimento, que possui somente 8 GB.

Análise dos Resultados
Changing Number of CSDs Units

#1. O estágio de envio das imagens ao servidor, UPLOAD, obtém um ganho em Speedup de até 2,85 com o aumento do paralelismo proporcionado pelo acréscimo de unidades CSDs e com o uso de processamento MULTI THREAD (MT) nas operações de leitura e escrita dos arquivo.

#2. O estágio de seleção, download e processamento das imagens, PROC, obtém um ganho em Speedup de até 2,21 com o aumento do paralelismo através do acréscimo de unidades CSDs, usando processamento MULTI THREAD (MT) nas operações de acesso ao disco, e armazenamento do arquivo de imagem temporário em memória compartilhada usando SHARED MEMORY (SM).

#3. O aumento no número de unidades CSDs gerou um aumento na concorrencia entre os processos por recursos de processamento e memória, e o melhor desempenho obtido com 4 unidades CSDs é a configuração que efetua armazenamento em disco local, DATASTORE (DS), e uma linha de execução para acesso ao disco, SINGLE THREAD (ST).

#4. Quando avaliamos o Tempo de Execução Total, considerando a execução de todos os estágios, MKDIR, UPLOAD e PROC, observamos que com o acréscimo de unidades CSDs e o uso de SHARED MEMORY (SM) para comunicação entre o servidor e os módulos de processamento, obtemos um ganho de Speedup de até 2,33 usando 4 unidades CSDs.

#5. A simulação de 4 unidades CSDs apresentou um baixo desempenho no estágio de processamento, PROC, que foi compensado pelo excelente resultado obtido no estágio de envio das imagens ao servidor, UPLOAD.

#6. No estágio de processamento das imagens, PROC, o armazenamento dos arquivos temporários pode se beneficiar bastante com o uso de memória compartilhada, SHARED MEMORY (SM), entre o servidor e os módulos de processamento.

#7. Observamos que usando 2 unidades CSDs obtemos ganhos de até 1,12 em Speedup em relação a configuração de referência para o mesmo número de unidades CSDs.

#8. O baixo resultado obtido usando 4 unidades CSDs ocorreu por falta de recursos no equipamento usado neste experimento (8 GB RAM).

#9. O acréscimo de unidades CSDs e o aumento do paralelismo com a implementação de MULTI THREADS (MT) no acesso de leitura e escrita do disco, apresentou ganhos significativos de Speedup que variaram de 1,57 à 1,63 usando 2 unidades CSDs, e entre 2,68 à 2,75 usando 4 unidades CSDs.

PESQUISA PARA A TESE DOUTORADO
HORUS IMAGE SERVER v2.2
PROFILE ANALISYS

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ
Registro: 2000103581 CREA-RJ

Análise dos Resultados
Profile Flat - Tabelas 1 e 2

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
1	memNCpyUtil(unsigned char*^unsigned char*^int)	49.55	16.87	16.87	66988	0	0
2	memSetNullUtil(unsigned char*^int)	37.45	29.62	12.75	44410	0	0
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^lon	8.02	32.35	2.73	22144	0	0
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk_l	2.76	33.29	0.94	22226	0	0
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_bloc	2.06	33.99	0.7	22185	0	0
6	getHash(char*)	0.09	34.02	0.03	44238	0	0
7	CSequence::debugEntry(int^char const*^char const*^s	0.03	34.03	0.01	243725	0	0
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(lon	0.03	34.04	0.01	15	0	0.37
9	CDskSuperBlockMan::countItemsByPathOidAndDiskDe	0.03	34.05	0.01	15	0	0
10	warnMsg(int^char const*^char const*^char const*)	0	34.05	0	620905	0	0
11	getCurrentTimestamp()	0	34.05	0	66487	0	0
12	strNCmpCaseUtil(char*^char*^int^int)	0	34.05	0	53390	0	0
13	strNCpyUtil(char*^char*^int)	0	34.05	0	44997	0	0
14	openFileUtil(_IO_FILE**^char const*^char const*^long)	0	34.05	0	44445	0	0
15	strNCmpUtil(char*^char*^int)	0	34.05	0	44424	0	0
16	CAppMain::getSequencePtr()	0	34.05	0	44341	0	0
17	CSequence::nextVal(char*)	0	34.05	0	44303	0	0
18	CSequence::findItem(char*^sequence_struct**)	0	34.05	0	44281	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned char	0	34.05	0	22200	0	0
20	CDskDiskDev::getOid()	0	34.05	0	22166	0	0
21	CDskDiskGroup::getOid()	0	34.05	0	22159	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned char	0	34.05	0	22138	0	0
23	CDataTable::checkScoreList(double*^long^double)	0	34.05	0	8160	0	0
24	CCfgConfigTag::debugEntry(int^char const*^char const	0	34.05	0	1038	0	0
25	strSetEmptyUtil(char*)	0	34.05	0	993	0	0
26	strNCpyCaseUtil(char*^char*^int^int)	0	34.05	0	926	0	0
27	CCfgConfigTag::addNewConfigTag(int^int^char*)	0	34.05	0	926	0	0
28	CDskDiskManExec::getThreadAtPtr(long)	0	34.05	0	780	0	0
29	CDataTable::debugEntry(int^char const*^char const*^d	0	34.05	0	390	0	0
30	CAppExec::addDataHist(double)	0	34.05	0	390	0	0
31	CDskDiskManExec::resetThread(dsk_diskman_thread_	0	34.05	0	174	0	0
32	strRemoveEolUtil(char*^int)	0	34.05	0	139	0	0
33	errMsgIfNull(char const*^char const*^char const*^void	0	34.05	0	132	0	0
34	CCfgConfigTag::setConfigTagValue(char*^char*)	0	34.05	0	117	0	0
35	CCfgConfigTag::findConfigTagByName(char*)	0	34.05	0	117	0	0
36	strNCatUtil(char*^char*^int)	0	34.05	0	116	0	0

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
37	allocDataArray(int^int)	0	34.05	0	115	0	0
38	freeDataArray(void*)	0	34.05	0	89	0	0
39	CDskDiskManExec::initThread(dsk_diskman_thread_st	0	34.05	0	72	0	0
40	CDskDiskManExec::destroyThread(dsk_diskman_threa	0	34.05	0	72	0	0
41	CCmdListener::resetConnection(connection_struct*)	0	34.05	0	43	0	0
42	CDskPathMan::debugEntry(int^char const*^char const	0	34.05	0	40	0	0
43	CDskDiskGroup::getDiskDevAtPtr(int)	0	34.05	0	30	0	0
44	CDskDiskManExec::debugEntry(int^char const*^char c	0	34.05	0	30	0	0
45	CDskDiskManExec::getNextThreadPtr(long^long)	0	34.05	0	30	0	0
46	CImageTable::getImageTableName()	0	34.05	0	22	0	0
47	CAppMain::getConfigPtr()	0	34.05	0	22	0	0
48	CAppMain::getContextPtr()	0	34.05	0	22	0	0
49	CDskPathMan::findItem(long^char*^dsk_path_struct**)	0	34.05	0	21	0	0
50	CCfgConfig::getCurrRemoteUnitPtr()	0	34.05	0	18	0	0
51	CDskDiskGroup::getSuperBlockMan()	0	34.05	0	18	0	0
52	CAppExec::doRemoveImage(char*)	0	34.05	0	17	0	0
53	CExtModule::initModule(int^char***^char**)	0	34.05	0	15	0	0
54	CExtModule::loadModule()	0	34.05	0	15	0	0
55	CExtModule::terminateModule()	0	34.05	0	15	0	0
56	CExtModule::start()	0	34.05	0	15	0	0
57	CExtModule::execute()	0	34.05	0	15	0	0
58	CExtModule::CExtModule(long^char*^char*)	0	34.05	0	15	0	0
59	CCfgContext::getImgDstDir()	0	34.05	0	15	0	0
60	CDskPathMan::debug(int^char const*^char const*)	0	34.05	0	15	0	0
61	CDskDiskManExec::addReadThread(void*^void*^void*^	0	34.05	0	15	0	0
62	CDskDiskManExec::addWriteThread(void*^void*^void*^	0	34.05	0	15	0	0
63	CTableMetadata::getCurrNumEntries()	0	34.05	0	14	0	0
64	freeData(void*)	0	34.05	0	12	0	0
65	allocData(int)	0	34.05	0	12	0	0
66	CCfgContext::getDataDir()	0	34.05	0	12	0	0
67	CDskPathMan::findItem(long*dsk_path_struct**)	0	34.05	0	12	0	0
68	CCfgRemoteUnit::getOid()	0	34.05	0	12	0	0
69	CTableMetadata::getItemAt(int)	0	34.05	0	12	0	0
70	CImageTable::debugEntry(int^char const*^char const*^	0	34.05	0	11	0	0
71	CImageTable::findItem(long*image_table_struct*)	0	34.05	0	11	0	0
72	CAppExec::execIMG_TBLIMGRIO(long*image_table_st	0	34.05	0	11	0	0

Análise dos Resultados
Profile Flat - Tabelas 3 e 4

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
73	CAppMain::getImageTablePtr(char*)	0	34.05	0	11	0	0
74	CDskPathMan::setItemData(dsk_path_struct**/long*/lor	0	34.05	0	10	0	0
75	CDskPathMan::addItem(char*/char*/long*/long*/long*/lor	0	34.05	0	10	0	0
76	CDskDiskManExec::waitThreadGroup(long*/long)	0	34.05	0	10	0	0
77	CDskDiskManExec::resetAllThreadsByGroup(long)	0	34.05	0	10	0	0
78	showMessage(char const*/long)	0	34.05	0	9	0	0
79	CDskDiskMan::showListDir(long*/long*/long)	0	34.05	0	8	0	0
80	CDskDiskMan::doListDir(dsk_path_struct**/long*/long*	0	34.05	0	8	0	0
81	CDskPathMan::findAllChildByPathParent(dsk_path_str	0	34.05	0	8	0	0
82	CDskPathMan::getNumEntriesByPathParent(long*/long	0	34.05	0	8	0	0
83	strIsEmptyUtil(char*)	0	34.05	0	7	0	0
84	CCfgRemoteUnit::getName()	0	34.05	0	7	0	0
85	CCfgParam::loadConfig(int*/int*/CCfgConfigTag*)	0	34.05	0	7	0	0
86	CDskShMem::deleteShMemObject()	0	34.05	0	7	0	0
87	CCfgContext::getTempDir()	0	34.05	0	6	0	0
88	CCmdRequest::getAction()	0	34.05	0	6	0	0
89	CDskDiskMan::doReadFileMT(unsigned char**/long*/lo	0	34.05	0	6	0	0.94
90	CCmdListener::isRunning()	0	34.05	0	6	0	0
91	CDskDiskGroup::getNumOfDisks()	0	34.05	0	6	0	0
92	getFileExt(char*/int*/char*/int)	0	34.05	0	5	0	0
93	readFileUtil(char const**/unsigned char**/long*)	0	34.05	0	5	0	0
94	strCharCountUtil(char*/int*/char)	0	34.05	0	5	0	0
95	strPiece(char*/int*/char*/int*/char*/int)	0	34.05	0	5	0	0
96	CDataTable::getDataTableFullPath()	0	34.05	0	5	0	0
97	CDataTable::debug(int*/char const**/char const*)	0	34.05	0	5	0	0
98	CDataTable::loadFile(char*)	0	34.05	0	5	0	0
99	CDataTable::saveFile(char*)	0	34.05	0	5	0	0
100	CDataTable::CDataTable(char*/char*/char*/char*/int)	0	34.05	0	5	0	0
101	CCfgContext::getImgSrcDir()	0	34.05	0	5	0	0
102	CDskDiskMan::doWriteFileMT(unsigned char*/long*/long	0	34.05	0	5	0	0
103	CDskDiskMan::getDiskGroupPtr(long)	0	34.05	0	5	0	0
104	CDskDiskMan::getDiskGroupAtPtr(long)	0	34.05	0	5	0	0
105	CDskDiskMan::doMakeDir(long*/long*/char*/long*)	0	34.05	0	5	0	0
106	CDskDiskGroup::getCurrNumDiskDev()	0	34.05	0	5	0	0
107	CCfgRemoteUnit::loadConfig(int*/CCfgConfigTag*)	0	34.05	0	5	0	0
108	CAppExec::doProcessImage_Simpl(long*/image_table_	0	34.05	0	5	0	0

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
109	CAppExec::doProcessImage_Reproj(long*/image_table_	0	34.05	0	5	0	0
110	CAppExec::doProcessImage_Convert(long*/image_table	0	34.05	0	5	0	0
111	CDskShMem::newShMemObjectForWrite(char*/unsigne	0	34.05	0	5	0	0
112	CDataTable::getDataTableName()	0	34.05	0	4	0	0
113	CCmdListener::sendDataMessageToAny(char*/int*/unsi	0	34.05	0	4	0	0
114	CCmdListener::sendDataMessageToINET(sockaddr_in*	0	34.05	0	4	0	0
115	CCmdListener::getAvailableConnection()	0	34.05	0	4	0	0
116	CCfgDiskGroup::getOid()	0	34.05	0	4	0	0
117	CCfgConfig::getHostServerPtr()	0	34.05	0	3	0	0
118	CCfgModule::loadConfig(int*/CCfgConfigTag*)	0	34.05	0	3	0	0
119	CCmdParser::getCurrPart()	0	34.05	0	3	0	0
120	CCmdParser::getNumParts()	0	34.05	0	3	0	0
121	CCmdParser::getActionName()	0	34.05	0	3	0	0
122	CCmdParser::getRemoteUnitOid()	0	34.05	0	3	0	0
123	CCmdParser::getActionFromString(char*)	0	34.05	0	3	0	0
124	CCmdParser::parser(char*)	0	34.05	0	3	0	0
125	CCmdParser::getAction()	0	34.05	0	3	0	0
126	CCmdParser::CCmdParser()	0	34.05	0	3	0	0
127	CCmdParser::~CCmdParser()	0	34.05	0	3	0	0
128	CCfgContext::getBaseDataFile(char*/int*/long*/long)	0	34.05	0	3	0	0
129	CCfgContext::getBaseDataFilePath(char*/int*/long*/long)	0	34.05	0	3	0	0
130	CCfgContext::getBaseBlockTableFile(char*/int*/long*/lor	0	34.05	0	3	0	0
131	CCfgContext::getBaseBlockTableFilePath(char*/int*/lon	0	34.05	0	3	0	0
132	CCfgDiskDev::loadConfig(void*/void*/int*/int*/CCfgConfi	0	34.05	0	3	0	0
133	CCfgDiskDev::getDataFile()	0	34.05	0	3	0	0
134	CCfgDiskDev::getDataFilePath()	0	34.05	0	3	0	0
135	CCfgDiskDev::getBlockTableFile()	0	34.05	0	3	0	0
136	CCfgDiskDev::getBlockTableFilePath()	0	34.05	0	3	0	0
137	CCfgDiskDev::debug(int*/char const**/char const*)	0	34.05	0	3	0	0
138	CCfgDiskDev::getOid()	0	34.05	0	3	0	0
139	CCfgDiskDev::getName()	0	34.05	0	3	0	0
140	CCmdRequest::getCurrPart()	0	34.05	0	3	0	0
141	CCmdRequest::getNumParts()	0	34.05	0	3	0	0
142	CCmdRequest::getRemoteUnitOid()	0	34.05	0	3	0	0
143	CCmdRequest::initOldCmdRequest(long*/char*)	0	34.05	0	3	0	0
144	CCmdRequest::debug()	0	34.05	0	3	0	0

Análise dos Resultados
Profile Flat - Tabelas 5 e 6

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
145	CCmdRequest::reset()	0	34.05	0	3	0	0
146	CCmdRequest::CCmdRequest(long*char*)	0	34.05	0	3	0	0
147	CCmdRequest::~CCmdRequest()	0	34.05	0	3	0	0
148	CDskDiskDev::init(void*^void*^long^long^long^long^long	0	34.05	0	3	0	0
149	CDskDiskDev::terminate()	0	34.05	0	3	0	0
150	CDskDiskDev::CDskDiskDev(void*^void*^long^long^lon	0	34.05	0	3	0	0
151	CCmdListener::sendDataMessageToHost(unsigned cha	0	34.05	0	3	0	0
152	CCmdListener::sendStringMessageToHost(char*)	0	34.05	0	3	0	0
153	CCmdResponse::toResultXml(char*)	0	34.05	0	3	0	0
154	CCmdResponse::doResponseSuccess(char*)	0	34.05	0	3	0	0
155	CCmdResponse::debug()	0	34.05	0	3	0	0
156	CCmdResponse::CCmdResponse(long)	0	34.05	0	3	0	0
157	CCmdResponse::~CCmdResponse()	0	34.05	0	3	0	0
158	CDskBlockMan::init(int^char*^char*)	0	34.05	0	3	0	0
159	CDskBlockMan::loadFile(char*)	0	34.05	0	3	0	0
160	CDskBlockMan::saveFile(char*)	0	34.05	0	3	0	0
161	CDskBlockMan::terminate()	0	34.05	0	3	0	0
162	CDskBlockMan::CDskBlockMan(int^char*^char*)	0	34.05	0	3	0	0
163	CDskBlockMan::~CDskBlockMan()	0	34.05	0	3	0	0
164	CCfgDiskGroup::getDiskDevAtPtr(int)	0	34.05	0	3	0	0
165	CDskDiskGroup::addDiskDev(long*char*^char*^char*^c	0	34.05	0	3	0	0
166	CCfgHostServer::getInPort()	0	34.05	0	3	0	0
167	CCfgHostServer::getIpAddr()	0	34.05	0	3	0	0
168	CDskRoundRobin::addRoundRobinTableItem(CCfgRemo	0	34.05	0	3	0	0
169	CAppMain::getDiskManPtr()	0	34.05	0	3	0	0
170	CCmdExec::doExec()	0	34.05	0	3	0	1.91
171	CCmdExec::CCmdExec(CCmdRequest*^CCmdRespons	0	34.05	0	3	0	0
172	CCmdExec::~CCmdExec()	0	34.05	0	3	0	0
173	CCfgConfig::getRemoteUnitPtr(int)	0	34.05	0	2	0	0
174	CDataTable::calculateDataPart(long*long*long*)	0	34.05	0	2	0	0
175	CDataTable::findAllByClassifScoreList(data_table_struc	0	34.05	0	2	0	0.05
176	CDskDiskMan::getCurrDir()	0	34.05	0	2	0	0
177	CDskDiskMan::getPathMan()	0	34.05	0	2	0	0
178	CTableSpace::findItem(long*tablespace_struct**)	0	34.05	0	2	0	0
179	CCmdListener::setRunning(int)	0	34.05	0	2	0	0
180	CCmdListener::stop()	0	34.05	0	2	0	0

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
181	CCmdListener::getInPort()	0	34.05	0	2	0	0
182	CCfgRemoteUnit::getInPort()	0	34.05	0	2	0	0
183	CCfgRemoteUnit::getIpAddr()	0	34.05	0	2	0	0
184	CAppExec::initDataHist()	0	34.05	0	2	0	0
185	CAppExec::debugDataHist(char*)	0	34.05	0	2	0	0
186	CAppExec::execTSn_TBLPATSAND(data_table_struct*	0	34.05	0	2	0	0.05
187	CAppExec::CAppExec()	0	34.05	0	2	0	0
188	CAppExec::~CAppExec()	0	34.05	0	2	0	0
189	CAppMain::getDataTablePtr(char*)	0	34.05	0	2	0	0
190	CDskShMem::createShMemObject()	0	34.05	0	2	0	0
191	CDskShMem::init()	0	34.05	0	2	0	0
192	CDskShMem::terminate()	0	34.05	0	2	0	0
193	CDskShMem::CDskShMem()	0	34.05	0	2	0	0
194	CDskShMem::~CDskShMem()	0	34.05	0	2	0	0
195	CSequence::debug(int^char const*^char const*)	0	34.05	0	2	0	0
196	_GLOBAL__sub_I_gAppMain	0	34.05	0	1	0	0
197	_GLOBAL__sub_I_gCmdListener	0	34.05	0	1	0	0.02
198	_GLOBAL__sub_I_gExtSched	0	34.05	0	1	0	0
199	showParams(cmdline_struct*)	0	34.05	0	1	0	0
200	parserParams(int^char**^cmdline_struct*)	0	34.05	0	1	0	0
201	getLocalTimeStr(char*)	0	34.05	0	1	0	0
202	showHeaderMessage()	0	34.05	0	1	0	0
203	showMessageAndWaitForKey(char const*)	0	34.05	0	1	0	0
204	__static_initialization_and_destruction_0(int^int)	0	34.05	0	1	0	0
205	__static_initialization_and_destruction_0(int^int)	0	34.05	0	1	0	0
206	__static_initialization_and_destruction_0(int^int)	0	34.05	0	1	0	0.02
207	CCfgConfig::loadConfig(char*)	0	34.05	0	1	0	0
208	CCfgConfig::getDiskGroupAtPtr(int)	0	34.05	0	1	0	0
209	CCfgConfig::getCurrNumDiskGroups()	0	34.05	0	1	0	0
210	CCfgConfig::init(int^int^int)	0	34.05	0	1	0	0
211	CCfgConfig::debug(int^char const*^char const*)	0	34.05	0	1	0	0
212	CCfgConfig::terminate()	0	34.05	0	1	0	0
213	CCfgConfig::CCfgConfig(int^int^int)	0	34.05	0	1	0	0
214	CCfgConfig::~CCfgConfig()	0	34.05	0	1	0	0
215	CCfgContext::initDefaults()	0	34.05	0	1	0	0
216	CCfgContext::resetDataLocation(char*)	0	34.05	0	1	0	0

Análise dos Resultados
Profile Flat - Tabelas 7 e 8

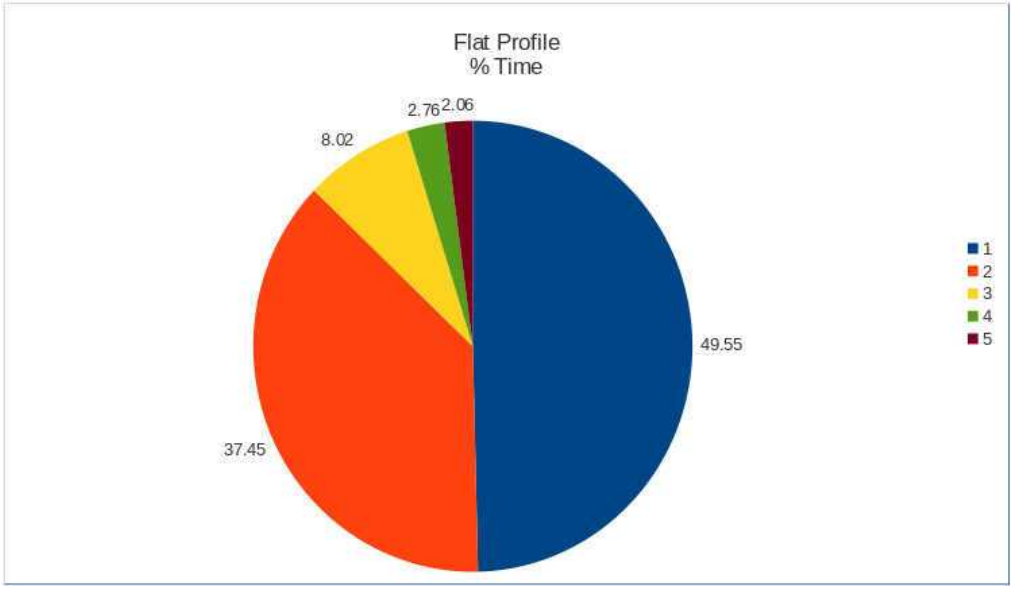
Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
217	CCfgContext::getBaseDirTableFile(char*^int)	0	34.05	0	1	0	0
218	CCfgContext::getBaseDirTableFilePath(char*^int)	0	34.05	0	1	0	0
219	CCfgContext::getBaseSuperBlockTableFile(char*^int^lo	0	34.05	0	1	0	0
220	CCfgContext::getBaseSuperBlockTableFilePath(char*^	0	34.05	0	1	0	0
221	CCfgContext::CCfgContext()	0	34.05	0	1	0	0
222	CCfgContext::~CCfgContext()	0	34.05	0	1	0	0
223	CDskDiskMan::addDiskGroup(long*long*long*char*^cha	0	34.05	0	1	0	0
224	CDskDiskMan::initDiskGroup()	0	34.05	0	1	0	0
225	CDskDiskMan::testDiskMan_MakeDir(long*^long)	0	34.05	0	1	0	0
226	CDskDiskMan::init(char*^char*^int*int*int*int)	0	34.05	0	1	0	0
227	CDskDiskMan::doChDir(long)	0	34.05	0	1	0	0
228	CDskDiskMan::terminate()	0	34.05	0	1	0	0
229	CDskDiskMan::CDskDiskMan(void*^char*^char*^int*int	0	34.05	0	1	0	0
230	CDskDiskMan::~CDskDiskMan()	0	34.05	0	1	0	0
231	CDskPathMan::getMaxNumDskPath()	0	34.05	0	1	0	0
232	CDskPathMan::init(int^char*^char*)	0	34.05	0	1	0	0
233	CDskPathMan::loadFile(char*)	0	34.05	0	1	0	0
234	CDskPathMan::saveFile(char*)	0	34.05	0	1	0	0
235	CDskPathMan::terminate()	0	34.05	0	1	0	0
236	CDskPathMan::CDskPathMan(int^char*^char*)	0	34.05	0	1	0	0
237	CDskPathMan::~CDskPathMan()	0	34.05	0	1	0	0
238	CImageTable::getImageTableFullPath()	0	34.05	0	1	0	0
239	CImageTable::debug(int^char const*^char const*)	0	34.05	0	1	0	0
240	CImageTable::loadFile(char*)	0	34.05	0	1	0	0
241	CImageTable::saveFile(char*)	0	34.05	0	1	0	0
242	CImageTable::CImageTable(char*^char*^char*^char*^int	0	34.05	0	1	0	0
243	CTableSpace::debug(int^char const*^char const*)	0	34.05	0	1	0	0
244	CTableSpace::loadFile(char*)	0	34.05	0	1	0	0
245	CTableSpace::saveFile(char*)	0	34.05	0	1	0	0
246	CTableSpace::CTableSpace(int)	0	34.05	0	1	0	0
247	CCmdDispatch::CCmdDispatch(char*)	0	34.05	0	1	0	0
248	CCmdDispatch::~CCmdDispatch()	0	34.05	0	1	0	0
249	CCmdListener::setOutPort(long)	0	34.05	0	1	0	0
250	CCmdListener::resetAllConnections()	0	34.05	0	1	0	0.02
251	CCmdListener::sendDataMessageToCurrRemoteUnit(unl	0	34.05	0	1	0	0
252	CCmdListener::sendStringMessageToCurrRemoteUnit(c	0	34.05	0	1	0	0

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
253	CCmdListener::start()	0	34.05	0	1	0	0
254	CCmdListener::setInPort(long)	0	34.05	0	1	0	0
255	CCmdListener::setIpAddr(char*)	0	34.05	0	1	0	0
256	CCmdListener::CCmdListener()	0	34.05	0	1	0	0.02
257	CCfgConfigTag::buildConfigTags()	0	34.05	0	1	0	0
258	CCfgConfigTag::loadConfigTagsValue(char*)	0	34.05	0	1	0	0
259	CCfgConfigTag::init(int^char*)	0	34.05	0	1	0	0
260	CCfgConfigTag::terminate()	0	34.05	0	1	0	0
261	CCfgConfigTag::CCfgConfigTag()	0	34.05	0	1	0	0
262	CCfgConfigTag::~CCfgConfigTag()	0	34.05	0	1	0	0
263	CCfgDiskGroup::loadConfig(int^CCfgConfigTag*)	0	34.05	0	1	0	0
264	CCfgDiskGroup::getNumOfDisks()	0	34.05	0	1	0	0
265	CCfgDiskGroup::getNumOfCopies()	0	34.05	0	1	0	0
266	CCfgDiskGroup::getSuperBlockTableFile()	0	34.05	0	1	0	0
267	CCfgDiskGroup::getSuperBlockTableFilePath()	0	34.05	0	1	0	0
268	CCfgDiskGroup::getName()	0	34.05	0	1	0	0
269	CDskDiskGroup::init(long*long*long*long*char*^char*^c	0	34.05	0	1	0	0
270	CDskDiskGroup::terminate()	0	34.05	0	1	0	0
271	CDskDiskGroup::CDskDiskGroup(void*^long*long*long	0	34.05	0	1	0	0
272	CCfgHostServer::loadConfig(CCfgConfigTag*)	0	34.05	0	1	0	0
273	CCfgHostServer::initDefaults()	0	34.05	0	1	0	0
274	CCfgHostServer::init()	0	34.05	0	1	0	0
275	CCfgHostServer::terminate()	0	34.05	0	1	0	0
276	CCfgHostServer::CCfgHostServer()	0	34.05	0	1	0	0
277	CCfgHostServer::~CCfgHostServer()	0	34.05	0	1	0	0
278	CCfgRemoteUnit::getOutPort()	0	34.05	0	1	0	0
279	CDskRoundRobin::getMaxNumRoundRobinTable()	0	34.05	0	1	0	0
280	CDskRoundRobin::init(int)	0	34.05	0	1	0	0
281	CDskRoundRobin::terminate()	0	34.05	0	1	0	0
282	CDskRoundRobin::CDskRoundRobin(int)	0	34.05	0	1	0	0
283	CDskRoundRobin::~CDskRoundRobin()	0	34.05	0	1	0	0
284	CTableMetadata::loadFile(char*)	0	34.05	0	1	0	0
285	CTableMetadata::saveFile(char*)	0	34.05	0	1	0	0
286	CTableMetadata::CTableMetadata(int)	0	34.05	0	1	0	0
287	CDskDiskManExec::freeAllThreads()	0	34.05	0	1	0	0
288	CDskDiskManExec::init(void*^int)	0	34.05	0	1	0	0

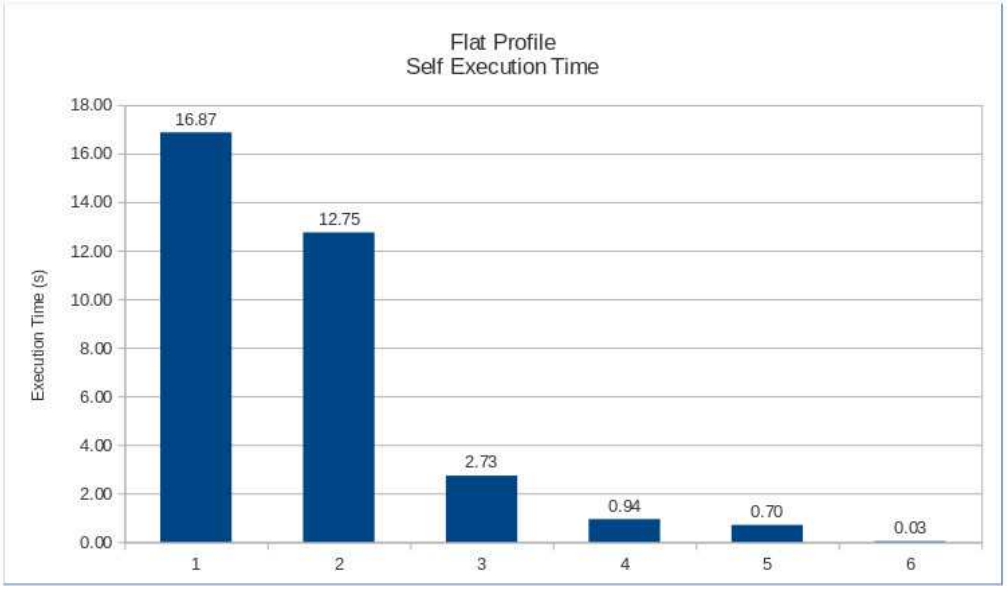
Análise dos Resultados
Profile Flat - Tabela 9

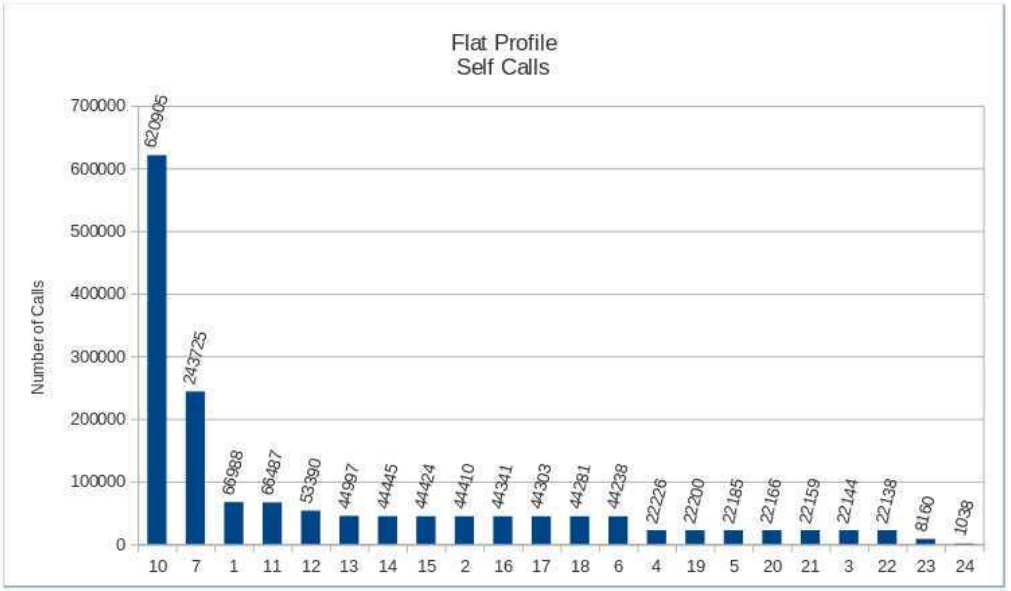
Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
289	CDskDiskManExec::terminate()	0	34.05	0	1	0	0
290	CDskDiskManExec::CDskDiskManExec(void*^int)	0	34.05	0	1	0	0
291	CDskDiskManExec::~CDskDiskManExec()	0	34.05	0	1	0	0
292	CDskSuperBlockMan::init(int^char*^char*)	0	34.05	0	1	0	0
293	CDskSuperBlockMan::loadFile(char*)	0	34.05	0	1	0	0
294	CDskSuperBlockMan::saveFile(char*)	0	34.05	0	1	0	0
295	CDskSuperBlockMan::terminate()	0	34.05	0	1	0	0
296	CDskSuperBlockMan::CDskSuperBlockMan(int^char*^c	0	34.05	0	1	0	0
297	CDskSuperBlockMan::~CDskSuperBlockMan()	0	34.05	0	1	0	0
298	CAppExec::doProcessImage_Upload(data_table_struct	0	34.05	0	1	0	0
299	CAppExec::doProcessImage_DownloadAndProcess(da	0	34.05	0	1	0	5.62
300	CAppMain::initConfig(char*)	0	34.05	0	1	0	0
301	CAppMain::initDiskMan()	0	34.05	0	1	0	0
302	CAppMain::initSequences(int)	0	34.05	0	1	0	0
303	CAppMain::initTablespace(int)	0	34.05	0	1	0	0
304	CAppMain::terminateConfig()	0	34.05	0	1	0	0
305	CAppMain::terminateDiskMan()	0	34.05	0	1	0	0
306	CAppMain::initDataTableList()	0	34.05	0	1	0	0
307	CAppMain::initTableMetadata()	0	34.05	0	1	0	0
308	CAppMain::initImageTableList()	0	34.05	0	1	0	0
309	CAppMain::terminateSequences()	0	34.05	0	1	0	0
310	CAppMain::terminateTablespace()	0	34.05	0	1	0	0
311	CAppMain::terminateDataTableList()	0	34.05	0	1	0	0
312	CAppMain::terminateTableMetadata()	0	34.05	0	1	0	0
313	CAppMain::terminateImageTableList()	0	34.05	0	1	0	0
314	CAppMain::init(int^char*)	0	34.05	0	1	0	0
315	CAppMain::terminate()	0	34.05	0	1	0	0
316	CAppMain::CAppMain()	0	34.05	0	1	0	0
317	CCmdExec::doCmdActionTSnMkDir(char*^double*^long	0	34.05	0	1	0	0
318	CCmdExec::doCmdActionTSnUpload(char*^double*^lon	0	34.05	0	1	0	0.05
319	CCmdExec::doCmdActionTSnDownloadAndProcess(cha	0	34.05	0	1	0	5.67
320	CExtSched::CExtSched(int)	0	34.05	0	1	0	0
321	CSequence::loadFile(char*)	0	34.05	0	1	0	0
322	CSequence::saveFile(char*)	0	34.05	0	1	0	0
323	CSequence::CSequence(int)	0	34.05	0	1	0	0

Análise dos Resultados
Profile Flat – Graficos 1 e 2



Análise dos Resultados
Profile Flat – Graficos 3 e 4





Análise dos Resultados
Profile Flat – PROBLEMAS IDENTIFICADOS

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
1	memNCpyUtil(unsigned char*^unsigned char*^int)	49.55	16.87	16.87			
2	memSetNullUtil(unsigned char*^int)	37.45	29.62	12.75			
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^	8.02	32.35	2.73			
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk	2.76	33.29	0.94			
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_bl	2.06	33.99	0.7			
6	getHash(char*)	0.09	34.02	0.03			
7	CSequence::debugEntry(int^char const*^char const*^s	0.03	34.03	0.01			
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(l	0.03	34.04	0.01			
9	CDskSuperBlockMan::countItemsByPathOidAndDiskD	0.03	34.05	0.01			
10	warnMsg(int^char const*^char const*^char const*)	0	34.05	0			
11	getCurrentTimestamp()	0	34.05	0			
12	strNCmpCaseUtil(char*^char*^int^int)	0	34.05	0			
13	strNCpyUtil(char*^char*^int)	0	34.05	0			
14	openFileUtil(_IO_FILE**^char const*^char const*^long)	0	34.05	0			
15	strNCmpUtil(char*^char*^int)	0	34.05	0	44424	0	0
16	CAppMain::getSequencePtr()	0	34.05	0	44341	0	0
17	CSequence::nextVal(char*)	0	34.05	0	44303	0	0
18	CSequence::findItem(char*^sequence_struct**)	0	34.05	0	44281	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned cl	0	34.05	0	22200	0	0
20	CDskDiskDev::getOid()	0	34.05	0	22166	0	0
21	CDskDiskGroup::getOid()	0	34.05	0	22159	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned c	0	34.05	0	22138	0	0
23	CDataTable::checkScoreList(double*^long^double)	0	34.05	0	8160	0	0
24	CCfgConfigTag::debugEntry(int^char const*^char const	0	34.05	0	1038	0	0
25	strSetEmptyUtil(char*)	0	34.05	0	993	0	0
26	strNCpyCaseUtil(char*^char*^int^int)	0	34.05	0	926	0	0
27	CCfgConfigTag::addNewConfigTag(int^int^char*)	0	34.05	0	926	0	0
28	CDskDiskManExec::getThreadAtPtr(long)	0	34.05	0	780	0	0
29	CDataTable::debugEntry(int^char const*^char const*^d	0	34.05	0	390	0	0
30	CAppExec::addDataHist(double)	0	34.05	0	390	0	0
31	CDskDiskManExec::resetThread(dsk_diskman_thread_	0	34.05	0	174	0	0
32	strRemoveEolUtil(char*^int)	0	34.05	0	139	0	0
33	errMsgIfNull(char const*^char const*^char const*^void	0	34.05	0	132	0	0
34	CCfgConfigTag::setConfigTagValue(char*^char*)	0	34.05	0	117	0	0
35	CCfgConfigTag::findConfigTagByName(char*)	0	34.05	0	117	0	0
36	strNCatUtil(char*^char*^int)	0	34.05	0	116	0	0

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^	8.02	32.35	2.73	22144	0	0
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk	2.76	33.29	0.94	22226	0	0
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_bl	2.06	33.99	0.7	22185	0	0
6	getHash(char*)	0.09	34.02	0.03	44238	0	0
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(l	0.03	34.04	0.01	15	0	0.37
9	CDskSuperBlockMan::countItemsByPathOidAndDiskD	0.03	34.05	0.01	15	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned cl	0	34.05	0	22200	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned c	0	34.05	0	22138	0	0

Análise dos Resultados
Profile Flat – SUJESTÕES

#1. Adição de índices nas tabelas de dados.

#2. Implementação da estrutura de Diretório, da lista de Superblocos e da lista de Blocos armazenados em memória usando “HashTables”.

#3. Aumentar o número de threads simultaneamente disparadas, com o objetivo de melhorar o desempenho no processamento das solicitações de leitura ou gravação de dados no disco.

FOCO INICIAL:

#2. Implementação da lista de Superblocos e da lista de Blocos armazenados em memória usando “HashTables”.

*** CdskSuperBlockMan::findFirstFreeSuperBlock();**
*** CdskSuperBlockMan::findAllByPathOidAndDiskDev();**
*** CdskSuperBlockMan::countItemsByPathOidAndDiskDev();**

*** CdskBlockMan::findFirstFreeBlock();**
*** CdskBlockMan::findBlockByBlockNum();**

*** CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)**
*** CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)**

PESQUISA PARA A TESE DOUTORADO
HORUS IMAGE SERVER v2.3
PROFILE ANALISYS

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ
Registro: 2000103581 CREA-RJ

Análise dos Resultados após Reajustes
Profile Flat – COMPARAÇÃO

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
1	memNCpyUtil(unsigned char*^unsigned char*^int)	49.55	16.87	16.87	66988	0	0
2	memSetNullUtil(unsigned char*^int)	37.45	29.62	12.75	44410	0	0
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^long^long^long^long^long^dsk_superblock_struct**^long*)	8.02	32.35	2.73	22144	0	0
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk_block_struct**^long*^long*)	2.76	33.29	0.94	22226	0	0
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_block_struct**^long*^long*)	2.06	33.99	0.7	22185	0	0
6	getHash(char*)	0.09	34.02	0.03	44238	0	0
7	CSequence::debugEntry(int*^char const*^char const*^sequence_struct*)	0.03	34.03	0.01	243725	0	0
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(long^long^dsk_superblock_struct**^long*)	0.03	34.04	0.01	15	0	0.37
9	CDskSuperBlockMan::countItemsByPathOidAndDiskDev(long^long^long*)	0.03	34.05	0.01	15	0	0
10	warnMsg(int*^char const*^char const*^char const*)	0	34.05	0	620905	0	0
11	getCurrentTimestamp()	0	34.05	0	66487	0	0
12	strNCmpCaseUtil(char*^char*^int*^int)	0	34.05	0	53390	0	0
13	strNCpyUtil(char*^char*^int)	0	34.05	0	44997	0	0
14	openFileUtil(_IO_FILE**^char const*^char const*^long)	0	34.05	0	44445	0	0
15	strNCmpUtil(char*^char*^int)	0	34.05	0	44424	0	0
16	CAppMain::getSequencePtr()	0	34.05	0	44341	0	0
17	CSequence::nextVal(char*)	0	34.05	0	44303	0	0
18	CSequence::findItem(char*^sequence_struct**)	0	34.05	0	44281	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	34.05	0	22200	0	0
20	CDskDiskDev::getOid()	0	34.05	0	22166	0	0
21	CDskDiskGroup::getOid()	0	34.05	0	22159	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	34.05	0	22138	0	0
23	CDataTable::checkScoreList(double*^long*^double)	0	34.05	0	8160	0	0
24	CCfgConfigTag::debugEntry(int*^char const*^char const*^config_tag_struct*)	0	34.05	0	1038	0	0
25	strSetEmptyUtil(char*)	0	34.05	0	993	0	0
26	strNCpyCaseUtil(char*^char*^int*^int)	0	34.05	0	926	0	0

Tabela 1a: Resultado inicial do PROFILE da aplicação

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
1	memNCpyUtil(unsigned char*^unsigned char*^int)	54.11	20.5	20.5	76550	0	0
2	memSetNullUtil(unsigned char*^int)	42.57	36.63	16.13	50698	0	0
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk_block_struct**^long*^long*)	3.06	37.79	1.16	25440	0	0
6	getHash(char*)	0.08	37.82	0.03	50679	0	0
9	CDskSuperBlockMan::countItemsByPathOidAndDiskDev(long^long^long*)	0.08	37.85	0.03	15	0	0
14	openFileUtil(_IO_FILE**^char const*^char const*^long)	0.03	37.86	0.01	50923	0	0
15	strNCmpUtil(char*^char*^int)	0.03	37.87	0.01	50535	0	0
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(long^long^dsk_superblock_struct**^long*)	0.03	37.88	0.01	15	0	0.46
NOVO	_GLOBAL__sub_I_65535_0__ZN11CDskDiskManC2EPvPcS1_ii	0.03	37.89	0.01	0	0	0
10	warnMsg(int*^char const*^char const*^char const*)	0	37.89	0	707216	0	0
24	CSequence::debugEntry(int*^char const*^char const*^sequence_struct*)	0	37.89	0	279137	0	0
11	getCurrentTimestamp()	0	37.89	0	76153	0	0
12	strNCmpCaseUtil(char*^char*^int*^int)	0	37.89	0	53390	0	0
13	strNCpyUtil(char*^char*^int)	0	37.89	0	51297	0	0
18	CSequence::findItem(char*^sequence_struct**)	0	37.89	0	50671	0	0
17	CSequence::nextVal(char*)	0	37.89	0	50610	0	0
16	CAppMain::getSequencePtr()	0	37.89	0	50572	0	0
20	CDskDiskDev::getOid()	0	37.89	0	25454	0	0
21	CDskDiskGroup::getOid()	0	37.89	0	25448	0	0
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^long^long^long^long^long^dsk_superblock_struct**^long*)	0	37.89	0	25436	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	37.89	0	25395	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	37.89	0	25341	0	0
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_block_struct**^long*^long*)	0	37.89	0	25147	0	0
23	CDataTable::checkScoreList(double*^long*^double)	0	37.89	0	8160	0	0
24	CCfgConfigTag::debugEntry(int*^char const*^char const*^config_tag_struct*)	0	37.89	0	1038	0	0
25	strSetEmptyUtil(char*)	0	37.89	0	993	0	0

Tabela 1b: Resultado do PROFILE após ajuste da aplicação

Análise dos Resultados após Reajustes
Profile Flat – COMENTÁRIOS

OTIMIZAÇÃO EFETUADA – HORUSWRK_v2.3.20220222

#1. Criação de Classes de estrutura de dados para auxiliar a implementação dos índexes das tabelas do sistema.
ESTRUTURAS DE DADOS: (a) FIFO; (b) STACK; (c) DOUBLE-LINKED LIST; (d) HASHTABLE; (e) BALANCED TREE (RUBRO-NEGRA);

#2. Otimização das funções para eveitar a busca por blocos livres no caso de haver espaço disponível na tabela.
*CDskSuperBlockMan::findFirstFreeSuperBlock();
*CDskBlockMan::findFirstFreeBlock();

OTIMIZAÇÕES A SEREM EFETUADAS – HORUSWRK_v2.4.202203xx

#1. Incluir a quantidade de blocos de um arquivo armazenados em cada unidade CSD.
* CdskSuperBlockMan::countItemsByPathOidAndDiskDev();

#2. Implementação da estrutura de Diretório, da lista de Superblocos e da lista de Blocos armazenados em memória usando "HASHTABLE" ou "BALANCED TREE".
* CdskSuperBlockMan::findAllByPathOidAndDiskDev();
* CdskBlockMan::findBlockByBlockNum();

#3. Aumentar o número de threads simultaneamente disparadas, com o objetivo de melhorar o desempenho no processamento das solicitações de leitura ou gravação de dados no disco.
* CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)
* CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)

PESQUISA PARA A TESE DOUTORADO
HORUS IMAGE SERVER v2.6
PROFILE ANALISYS

Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: lmarcio@cos.ufrj.br
lmarcio@tlmv.com.br
luiz.marcio.viana@gmail.com
Phone: +55-21-99983-7207
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ
Registro: 2000103581 CREA-RJ

Análise dos Resultados após Reajustes
Profile Flat – COMPARAÇÃO

Code	Function Name	% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
1	memNCpyUtil(unsigned char*^unsigned char*^int)	49.55	16.87	16.87	66988	0	0
2	memSetNullUtil(unsigned char*^int)	37.45	29.62	12.75	44410	0	0
3	CDskSuperBlockMan::findFirstFreeSuperBlock(long^long^long^long^long^long^dsk_superblock_struct**^long*)	8.02	32.35	2.73	22144	0	0
4	CDskBlockMan::findBlockByBlockNum(long^long^dsk_block_struct**^long*^long*)	2.76	33.29	0.94	22226	0	0
5	CDskBlockMan::findFirstFreeBlock(long^long^dsk_block_struct**^long*^long*)	2.06	33.99	0.7	22185	0	0
6	getHash(char*)	0.09	34.02	0.03	44238	0	0
7	CSequence::debugEntry(int^char const*^char const*^sequence_struct*)	0.03	34.03	0.01	243725	0	0
8	CDskSuperBlockMan::findAllByPathOidAndDiskDev(long^long^dsk_superblock_struct**^long*)	0.03	34.04	0.01	15	0	0.37
9	CDskSuperBlockMan::countItemsByPathOidAndDiskDev(long^long^long*)	0.03	34.05	0.01	15	0	0
10	warnMsg(int^char const*^char const*^char const*)	0	34.05	0	620905	0	0
11	getCurrentTimestamp()	0	34.05	0	66487	0	0
12	strNCmpCaseUtil(char*^char*^int*^int)	0	34.05	0	53390	0	0
13	strNCpyUtil(char*^char*^int)	0	34.05	0	44997	0	0
14	openFileUtil(_IO_FILE**^char const*^char const*^long)	0	34.05	0	44445	0	0
15	strNCmpUtil(char*^char*^int)	0	34.05	0	44424	0	0
16	CAppMain::getSequencePtr()	0	34.05	0	44341	0	0
17	CSequence::nextVal(char*)	0	34.05	0	44303	0	0
18	CSequence::findItem(char*^sequence_struct**)	0	34.05	0	44281	0	0
19	CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	34.05	0	22200	0	0
20	CDskDiskDev::getOid()	0	34.05	0	22166	0	0
21	CDskDiskGroup::getOid()	0	34.05	0	22159	0	0
22	CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)	0	34.05	0	22138	0	0
23	CDataTable::checkScoreList(double*^long*double)	0	34.05	0	8160	0	0
24	CCfgConfigTag::debugEntry(int*^char const*^char const*^config_tag_struct*)	0	34.05	0	1038	0	0
25	strSetEmptyUtil(char*)	0	34.05	0	993	0	0
26	strNCpyCaseUtil(char*^char*^int*^int)	0	34.05	0	926	0	0

Tabela 1a: Resultado inicial do PROFILE da aplicação

			% Execution Time	Cumulative Time (s)	Self Time (s)	# Self Calls	Total Time per Call (s)	Time per Call
		*^int)	51.57	10.03	10.03	71368	0	0
			28.33	15.54	5.51	46526	0	0
			7.61	17.02	1.48	51262286	0	0
		*/)	2.83	17.57	0.55	43968	0	0
			2.52	18.06	0.49	74873207	0	0
			2.06	18.46	0.4	54907857	0	0
NOVO	7	CIdxDescriptor::getKeyType()	1.75	18.8	0.34	57370879	0	0
NOVO	8	warnMsg(int^char^const*^char^const*^char^const*)	1.26	19.05	0.25	60437031	0	0
4	9	CDskBlockMan::findBlockByBlockNum(long^long^dsk_block_struct*	0.77	19.2	0.15	25619	0	0
NOVO	10	CIdxEntry::setEntryPrevPtr(CIdxEntry*)	0.31	19.26	0.06	163299	0	0
NOVO	11	_GLOBAL__sub_I_65535_0__ZN11CDskDiskManC2EPvPcS1_iii	0.18	19.29	0.04	0	0	0
6	12	getHash(char*)	0.15	19.35	0.03	40496	0	0
NOVO	13	CidxEntry::getEntryObjPtr()	0.15	19.32	0.03	22138644	0	0
18	14	CSequence::findItem(char*^sequence_struct**)	0.1	19.37	0.02	42554	0	0
NOVO	15	CSequence::findItem(long^sequence_struct**)	0.05	19.45	0.01	0	0	0
NOVO	16	CIdxEntry::terminate()	0.05	19.44	0.01	0	0	0
NOVO	17	dsk_diskman_read(void*)	0.05	19.43	0.01	0	0	0
NOVO	18	CIdxDescriptor::getNumberOfEntries()	0.05	19.42	0.01	15	0	0
22	19	CDskDiskDev::writeBlock(long^long^long^unsigned^char*^dsk_bloc	0.05	19.41	0.01	18906	0	0
3	20	CDskSuperBlockMan::findFirstFreeSuperBlock(long^long^long^long	0.05	19.4	0.01	19978	0	0
15	21	strNCmpUtil(char*^char*^int)	0.05	19.39	0.01	40580	0	0
13	22	strNCpyUtil(char*^char*^int)	0.05	19.38	0.01	87223	0	0
7	23	CSequence::debugEntry(int^char^const*^char^const*^sequence_struct*)	0	19.45	0	229600	0	0
NOVO	24	CIdxEntry::setEntryNextPtr(CIdxEntry*)	0	19.45	0	138138	0	0
NOVO	25	CIdxDoubleLinkedList::getDescriptorPtr()	0	19.45	0	67796	0	0
NOVO	26	CIdxEntry::getEntryPrevPtr()	0	19.45	0	65992	0	0
/* *** */	/* *** */	/* *** */	/* *** */	/* *** */	/* *** */	/* *** */	/* *** */	/* *** */
19	46	CDskDiskDev::readBlock(long^long^long^unsigned^char*^dsk_bloc	0	19.45	0	22187	0	0
5	50	CDskBlockMan::findFirstFreeBlock(long^long^dsk_block_struct**^lo	0	19.45	0	19551	0	0
8	106	CDskSuperBlockMan::findAllByPathOidAndDiskDev(long^long^dsk_	0	19.45	0	15	0	0.24
9	375	CDskSuperBlockMan::countItemsByPathOidAndDiskDev(long^long^	0	0	0	0	0	0

Tabela 1b: Resultado do PROFILE após ajuste da aplicação

Análise dos Resultados após Reajustes
Profile Flat – COMENTÁRIOS

OTIMIZAÇÃO EFETUADA – HORUSWRK_v2.6.20220417 (UNIDADES CSDs)

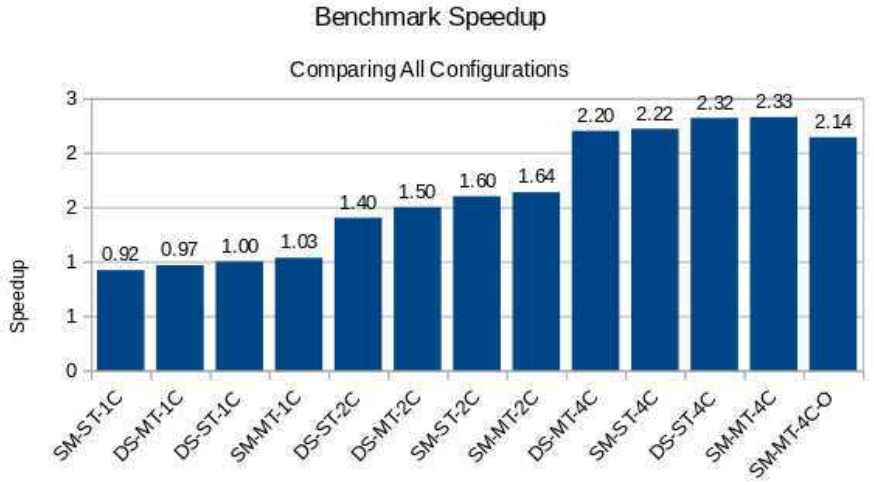
#1. Criação de Classes de estrutura de dados para auxiliar a implementação dos índexes das tabelas do sistema.
ESTRUTURAS DE DADOS: (a) FIFO; (b) STACK; (c) DOUBLE-LINKED LIST; (d) HASHTABLE; (e) BALANCED TREE (RUBRO-NEGRA);

#2. Otimização das funções para eveitar a busca por blocos livres no caso de haver espaço disponível na tabela.
*CDskSuperBlockMan::findFirstFreeSuperBlock();
*CDskBlockMan::findFirstFreeBlock();

#3. Incluir a quantidade de superblocos e blocos de um arquivo armazenados em cada unidade CSD.
#4. Implementação da estrutura de Diretório, da lista de Superblocos e da lista de Blocos armazenados em memória usando "HASHTABLE" ou "BALANCED TREE".
- Implementação de índice de superblocos e blocos usando HASHTABLE + DOUBLE-LINKED-LIST (chaves: path_oid + block_num)
- Quantidade de superblocos e blocos obtida do índice correspondente
* CdskSuperBlockMan::countItemsByPathOidAndDiskDev();
* CdskSuperBlockMan::findAllByPathOidAndDiskDev();
* CdskBlockMan::findBlockByBlockNum();

#5. Aumentar o número de threads simultaneamente disparadas, com o objetivo de melhorar o desempenho no processamento das solicitações de leitura ou gravação de dados no disco.
- Após otimização das estruturas de superblocos e blocos NÃO foi necessário efetuar otimização na leitura e escrita de blocos
* CDskDiskDev::readBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)
* CDskDiskDev::writeBlock(long^long^long^unsigned char*^dsk_block_struct**^long*^long*)

Análise dos Resultados
Comparing All Configurations – EXEC



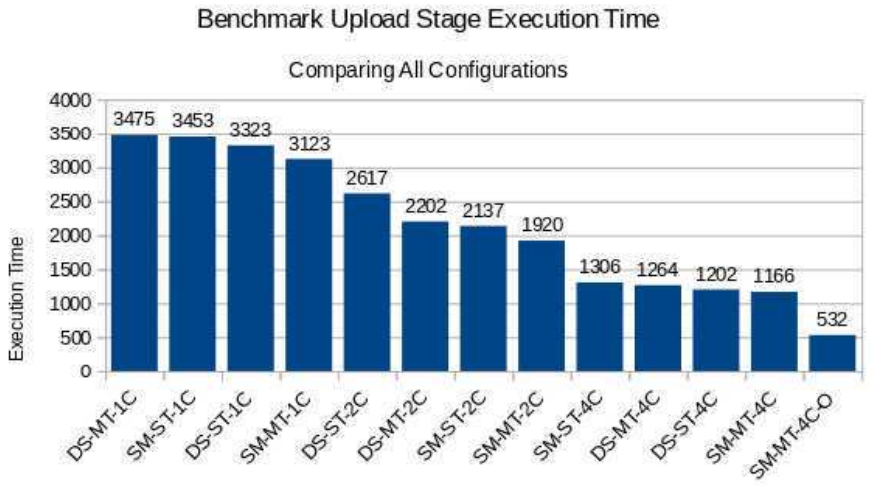
#1. COMPARE ALL CONFIGURATIONS

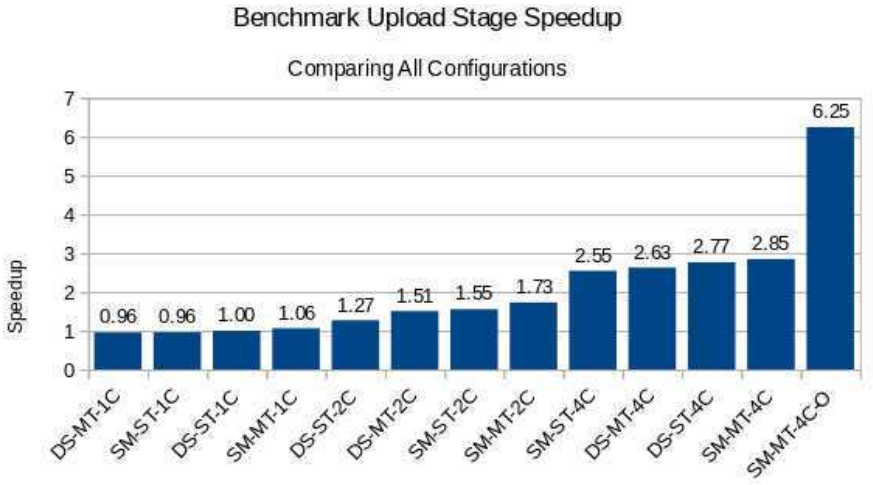
	UPLOAD	SPEEDUP
DS-MT-1C	3474.8353	0.9564
SM-ST-1C	3453.3635	0.9624
DS-ST-1C	3323.4447	1.0000
SM-MT-1C	3123.2359	1.0641
DS-ST-2C	2616.5962	1.2701
DS-MT-2C	2202.4704	1.5090
SM-ST-2C	2137.3474	1.5549
SM-MT-2C	1920.3308	1.7307
SM-ST-4C	1305.6174	2.5455
DS-MT-4C	1264.2624	2.6288
DS-ST-4C	1201.7843	2.7654
SM-MT-4C	1166.2446	2.8497
SM-MT-4C-O	531.7809	6.2497

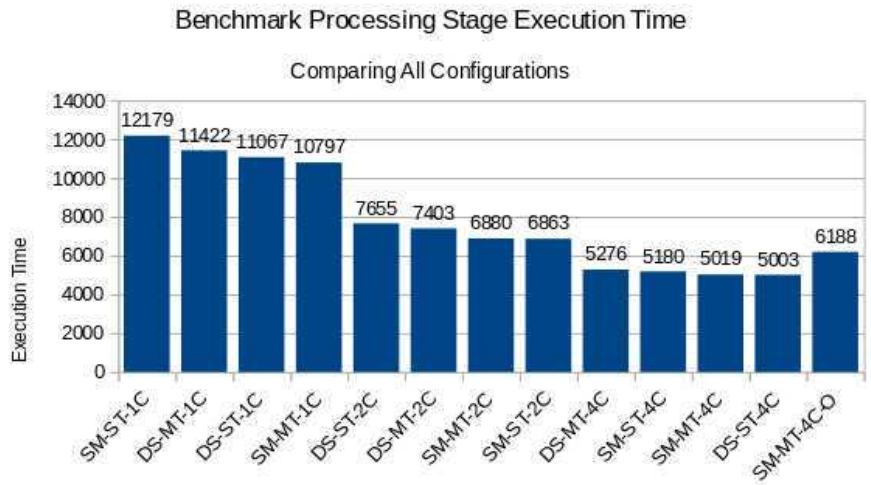
	PROC	SPEEDUP
SM-ST-1C	12179.48989	0.9087
DS-MT-1C	11421.66055	0.9689
DS-ST-1C	11066.97516	1.0000
SM-MT-1C	10797.46826	1.0250
DS-ST-2C	7655.495835	1.4457
DS-MT-2C	7402.624804	1.4950
SM-MT-2C	6879.671804	1.6086
SM-ST-2C	6862.887451	1.6126
DS-MT-4C	5276.398878	2.0974
SM-ST-4C	5180.05565	2.1365
SM-MT-4C	5018.965917	2.2050
DS-ST-4C	5003.089617	2.2120
SM-MT-4C-O	6188.274868	1.7884

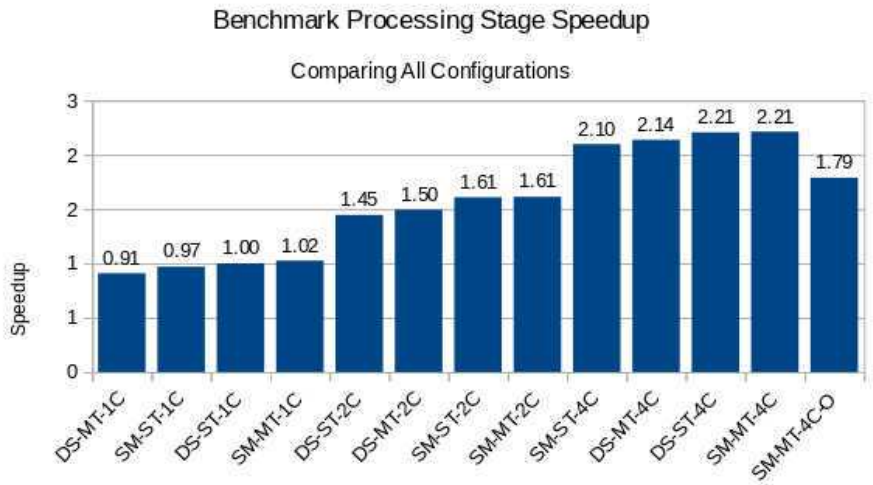
	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-MT-1C	14896.4967	0.9660
DS-ST-1C	14390.4208	1.0000
SM-MT-1C	13920.7050	1.0337
DS-ST-2C	10271.7929	1.4010
DS-MT-2C	9605.0964	1.4982
SM-ST-2C	9000.2360	1.5989
SM-MT-2C	8800.0032	1.6353
DS-MT-4C	6540.6636	2.2001
SM-ST-4C	6485.6756	2.2188
DS-ST-4C	6204.8780	2.3192
SM-MT-4C	6185.2129	2.3266
SM-MT-4C-O	6720.0582	2.1414

Análise dos Resultados
Comparing All Configurations – UPLOAD + PROC









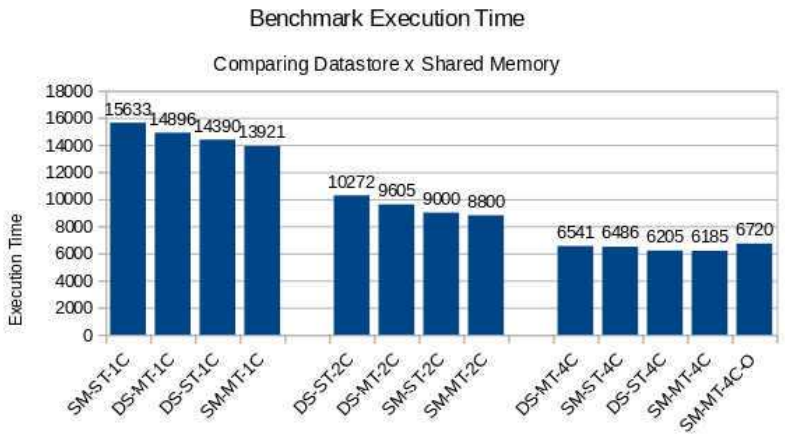
Análise dos Resultados
Comparing Datastore x Shared Memory – EXEC

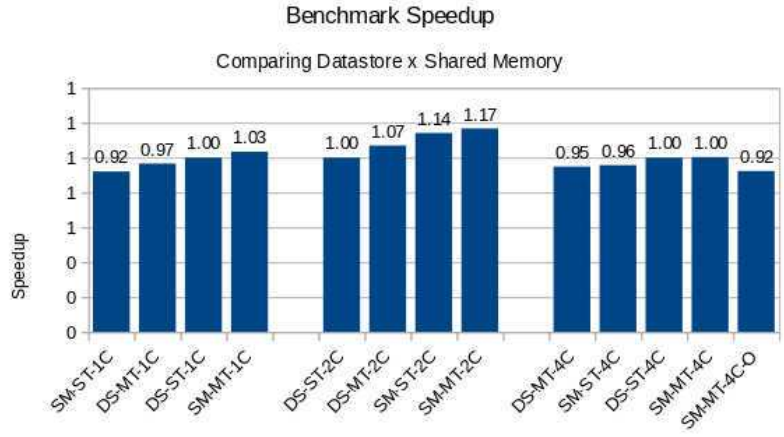
#2. COMPARE DATASTORE x SHMEM

	UPLOAD	SPEEDUP
SM-ST-1C	3453.3635	0.9624
DS-MT-1C	3474.8353	0.9564
DS-ST-1C	3323.4447	1.0000
SM-MT-1C	3123.2359	1.0641
DS-ST-2C	2616.5962	1.0000
DS-MT-2C	2202.4704	1.1880
SM-ST-2C	2137.3474	1.2242
SM-MT-2C	1920.3308	1.3626
DS-MT-4C	1264.2624	0.9506
SM-ST-4C	1305.6174	0.9205
DS-ST-4C	1201.7843	1.0000
SM-MT-4C	1166.2446	1.0305
SM-MT-4C-O	531.7809	2.2599

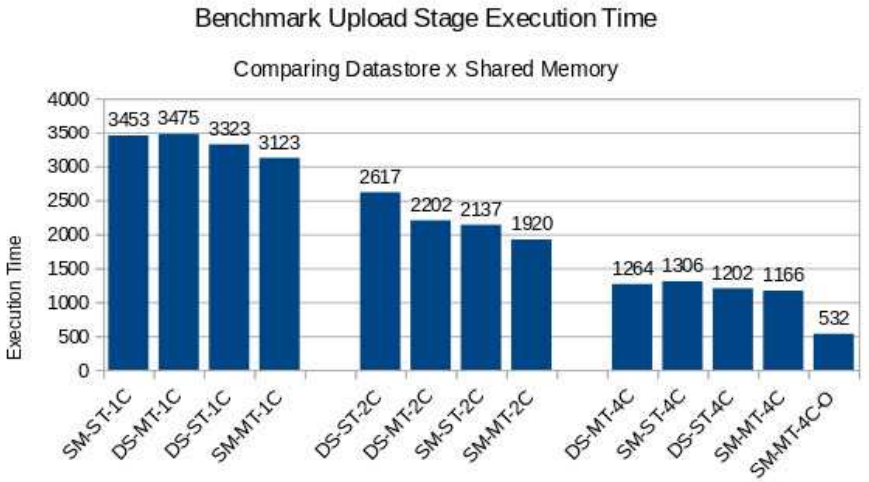
	PROC	SPEEDUP
SM-ST-1C	12179.48989	0.9087
DS-MT-1C	11421.66055	0.9689
DS-ST-1C	11066.97516	1.0000
SM-MT-1C	10797.46826	1.0250
DS-ST-2C	7655.195835	1.0000
DS-MT-2C	7402.624804	1.0341
SM-ST-2C	6862.887451	1.1154
SM-MT-2C	6879.671804	1.1127
DS-MT-4C	5276.398878	0.9482
SM-ST-4C	5180.05565	0.9658
DS-ST-4C	5003.089617	1.0000
SM-MT-4C	5018.965917	0.9968
SM-MT-4C-O	6188.2749	0.8085

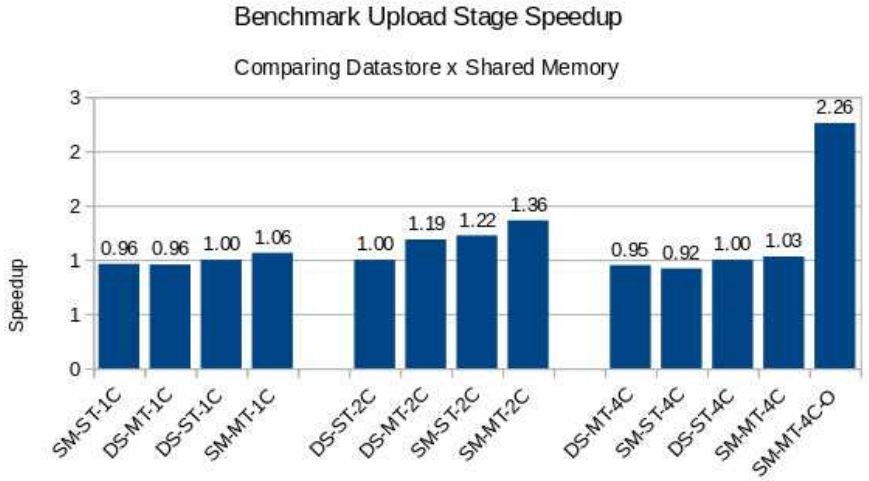
	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-MT-1C	14896.4967	0.9660
DS-ST-1C	14390.4208	1.0000
SM-MT-1C	13920.7050	1.0337
DS-ST-2C	10271.7929	1.0000
DS-MT-2C	9605.0964	1.0694
SM-ST-2C	9000.2360	1.1413
SM-MT-2C	8800.0032	1.1672
DS-MT-4C	6540.6636	0.9487
SM-ST-4C	6485.6756	0.9567
DS-ST-4C	6204.8780	1.0000
SM-MT-4C	6185.2129	1.0032
SM-MT-4C-O	6720.0582	0.9233

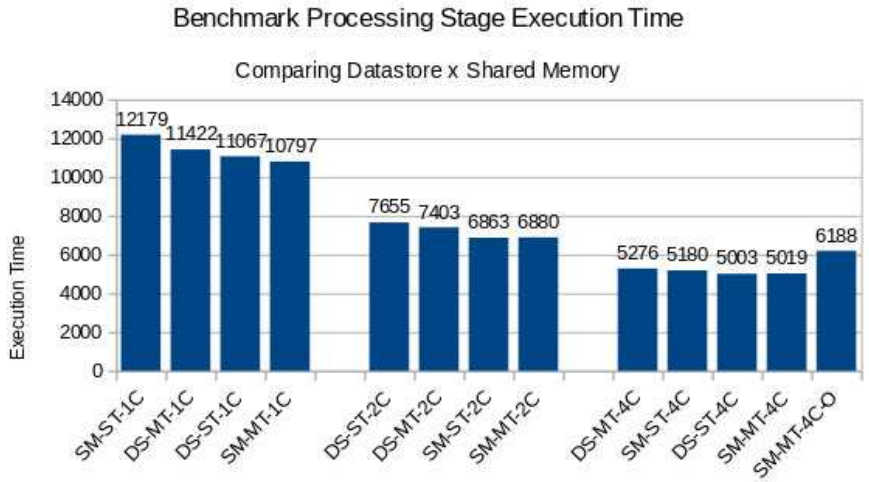


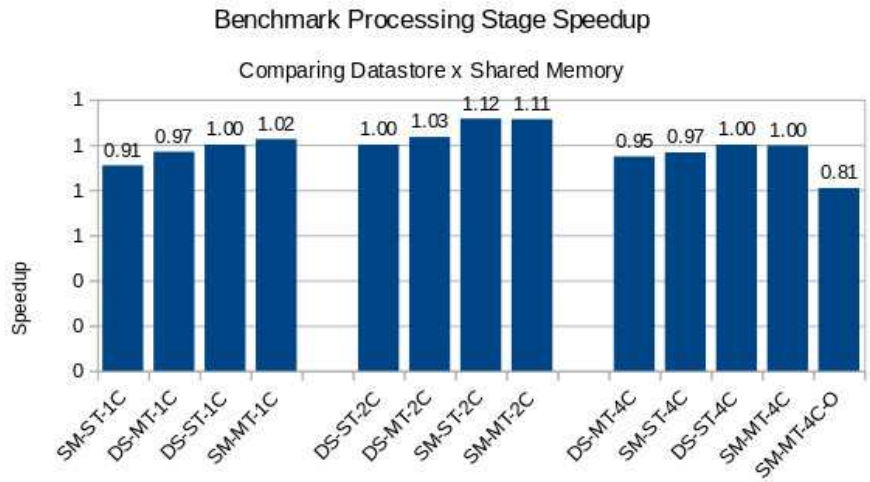


Análise dos Resultados
Comparing Datastore x Shared Memory – UPLOAD + PROC









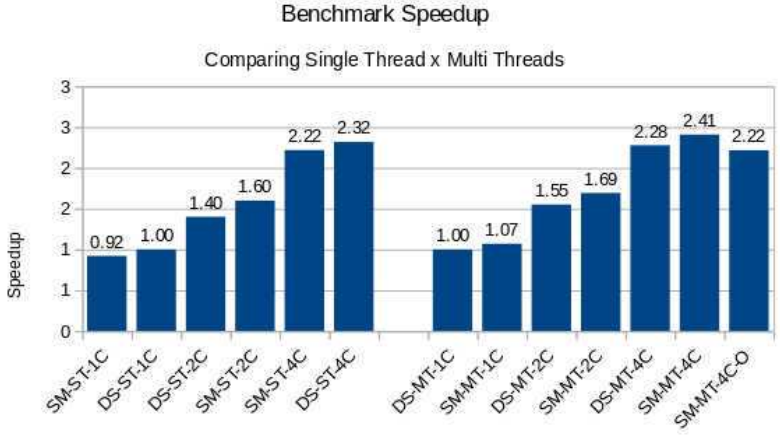
Análise dos Resultados
Comparing Single Thread x Multi Thread - EXEC

#3. COMPARE SINGLE THREAD x MULTI THREADS

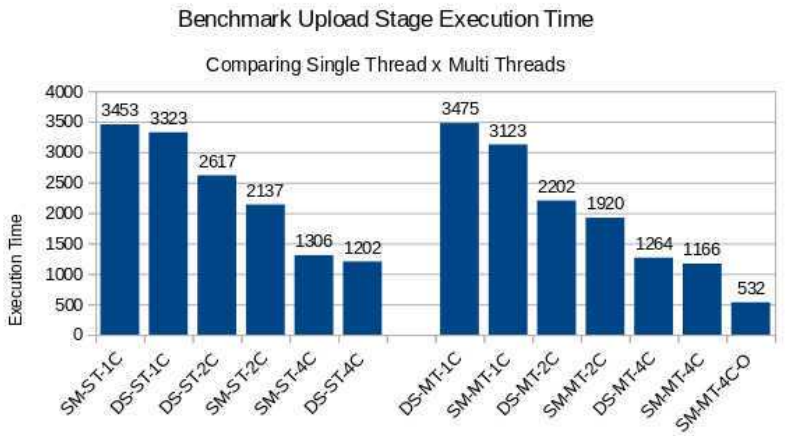
	UPLOAD	SPEEDUP
SM-ST-1C	3453.3635	0.9624
DS-ST-1C	3323.4447	1.0000
DS-ST-2C	2616.5962	1.2701
SM-ST-2C	2137.3474	1.5549
SM-ST-4C	1305.6174	2.5455
DS-ST-4C	1201.7843	2.7654
DS-MT-1C	3474.8353	1.0000
SM-MT-1C	3123.2359	1.1126
DS-MT-2C	2202.4704	1.5777
SM-MT-2C	1920.3308	1.8095
DS-MT-4C	1264.2624	2.7485
SM-MT-4C	1166.2446	2.9795
SM-MT-4C-O	531.7809	6.5343

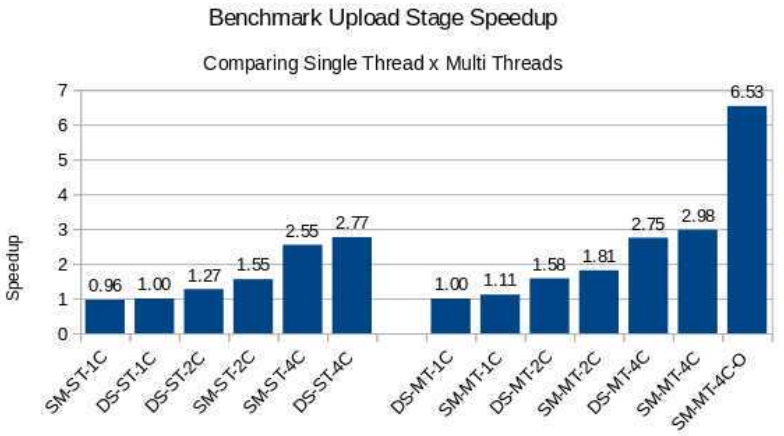
	PROC	SPEEDUP
SM-ST-1C	12179.4899	0.9087
DS-ST-1C	11066.9752	1.0000
DS-ST-2C	7655.1958	1.4457
SM-ST-2C	6862.8875	1.6126
SM-ST-4C	5180.0557	2.1365
DS-ST-4C	5003.0896	2.2120
DS-MT-1C	11421.6605	1.0000
SM-MT-1C	10797.4683	1.0578
DS-MT-2C	7402.6248	1.5429
SM-MT-2C	6879.6718	1.6602
DS-MT-4C	5276.3989	2.1647
SM-MT-4C	5018.9659	2.2757
SM-MT-4C-O	6188.2749	1.8457

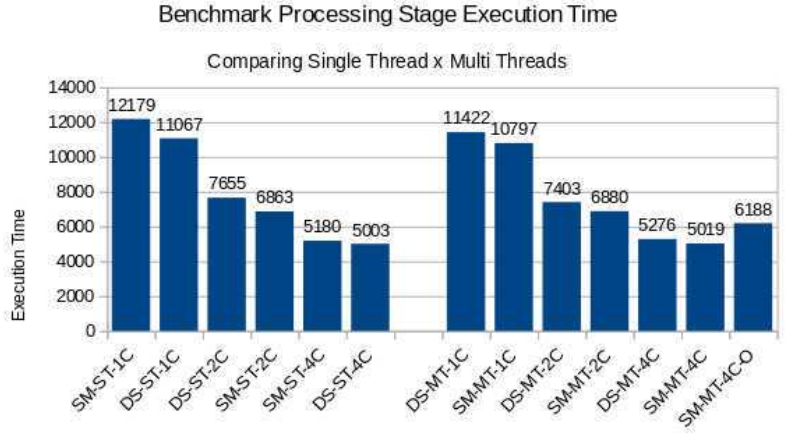
	TEMPO TOTAL	SPEEDUP
SM-ST-1C	15632.8541	0.9205
DS-ST-1C	14390.4208	1.0000
DS-ST-2C	10271.7929	1.4010
SM-ST-2C	9000.2360	1.5989
SM-ST-4C	6485.6756	2.2188
DS-ST-4C	6204.8780	2.3192
DS-MT-1C	14896.4967	1.0000
SM-MT-1C	13920.7050	1.0701
DS-MT-2C	9605.0964	1.5509
SM-MT-2C	8800.0032	1.6928
DS-MT-4C	6540.6636	2.2775
SM-MT-4C	6185.2129	2.4084
SM-MT-4C-O	6720.0582	2.2167

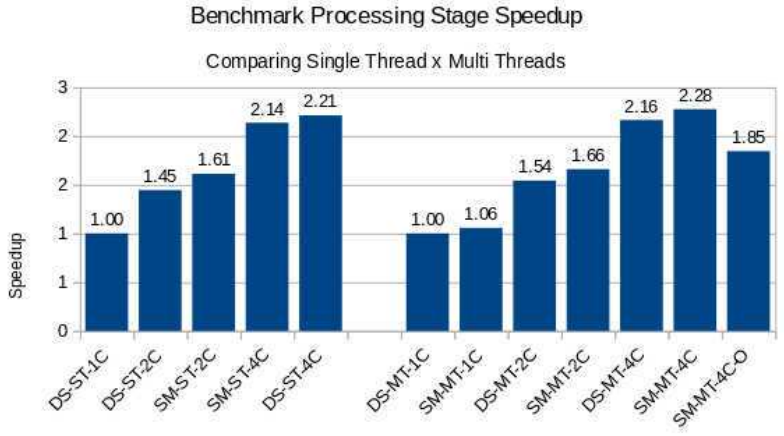


Análise dos Resultados
Comparing Single Thread x Multi Thread – UPLOAD + PROC









Análise dos Resultados
Changing Number of CSDs Units

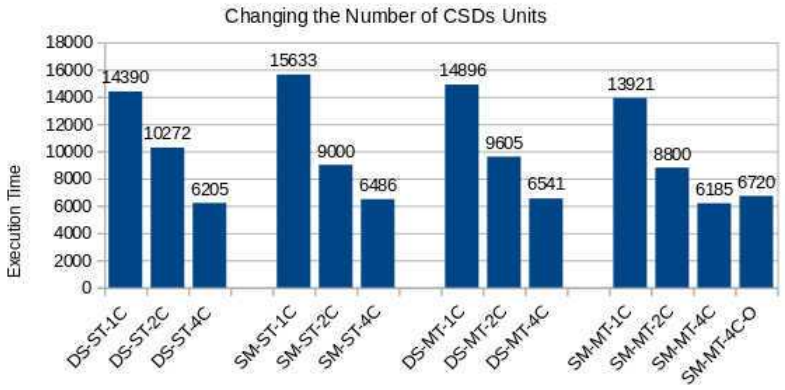
#4. CHANGE THE NUMBER OF CSDs UNITS

	UPLOAD	SPEEDUP
DS-ST-1C	3323.4447	1.0000
DS-ST-2C	2616.5962	1.2701
DS-ST-4C	1201.7843	2.7654
SM-ST-1C	3453.3635	1.0000
SM-ST-2C	2137.3474	1.6157
SM-ST-4C	1305.6174	2.6450
DS-MT-1C	3474.8353	1.0000
DS-MT-2C	2202.4704	1.5777
DS-MT-4C	1264.2624	2.7485
SM-MT-1C	3123.2359	1.0000
SM-MT-2C	1920.3308	1.6264
SM-MT-4C	1166.2446	2.6780
SM-MT-4C-O	531.7809	5.8732

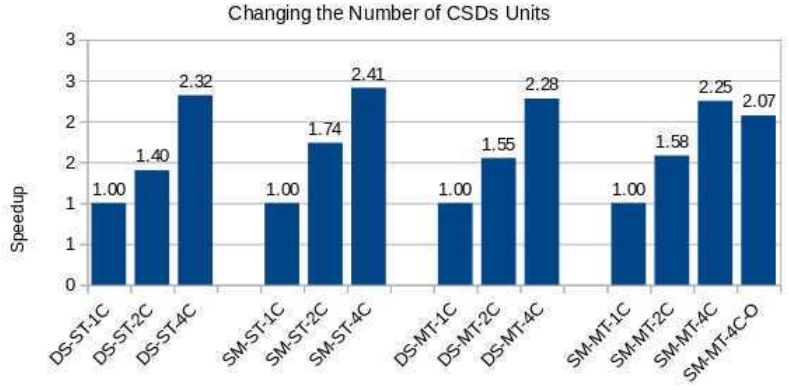
	PROC	SPEEDUP
DS-ST-1C	11066.9752	1.0000
DS-ST-2C	7655.1958	1.4457
DS-ST-4C	5003.0896	2.2120
SM-ST-1C	12179.4899	1.0000
SM-ST-2C	6862.8875	1.7747
SM-ST-4C	5180.0557	2.3512
DS-MT-1C	11421.6605	1.0000
DS-MT-2C	7402.6248	1.5429
DS-MT-4C	5276.3989	2.1647
SM-MT-1C	10797.4683	1.0000
SM-MT-2C	6879.6718	1.5695
SM-MT-4C	5018.9659	2.1513
SM-MT-4C-O	6188.2749	1.7448

	TEMPO TOTAL	SPEEDUP
DS-ST-1C	14390.4208	1.0000
DS-ST-2C	10271.7929	1.4010
DS-ST-4C	6204.8780	2.3192
SM-ST-1C	15632.8541	1.0000
SM-ST-2C	9000.2360	1.7369
SM-ST-4C	6485.6756	2.4104
DS-MT-1C	14896.4967	1.0000
DS-MT-2C	9605.0964	1.5509
DS-MT-4C	6540.6636	2.2775
SM-MT-1C	13920.7050	1.0000
SM-MT-2C	8800.0032	1.5819
SM-MT-4C	6185.2129	2.2506
SM-MT-4C-O	6720.0582	2.0715

Benchmark Execution Time



Benchmark Speedup



Analise dos Resultados e Conclusoes

ANALISE DOS RESULTADOS E CONCLUSOES – HORUSWRK_v2.6.20220417 (UNIDADES CSDs)

#1. Analisando os resultados, verificamos que as OTIMIZACOES REALIZADAS, e a adicao de indices as estruturas de SUPERBLOCOS e BLOCOS, melhoram muito o desempenho no acesso aos dados em disco para *UPLOAD* e *DOWNLOAD* das imagens, chegando a obter um speedup de 6,25 na fase de UPLOAD dos dados do Benchmark;

#2. A pior configuracao com 4 nucleos de processamento, DS-MT-4C, obteve um speedup de 2,20, enquanto a melhor configuracao com 4 nucleos de processamento, SM-MT-4C, obteve speedup de 2,33.

#3. Desta forma, a configuracao otimizada SM-MT-4C-O, com SHARED MEMORY + MULTITHREADING + 4 NUCLEOS DE PROCESSAMENTO + OTIMIZACAO, obteve um speedup de 2,14, o pior resultado entre as configuracoes com 4 nucleos, porque o tempo consumido no processamento das imagens e muito maior que o tempo consumido no *UPLOAD* e *DOWNLOAD* dos dados;

TODO:

#1. Criacao de indices para as tabelas de dados, usando HASHTABLE + ARVORE BALANCEADA (RUBRO-NEGRA);

#2. Otimizacao dos modulos de processamento das imagens: CONVERT_MOD, REPROJ_MOD, e SIMPL_MOD

#3. Implementacao dos modulos de aprendizado e classificacao das imagens: CHANGE

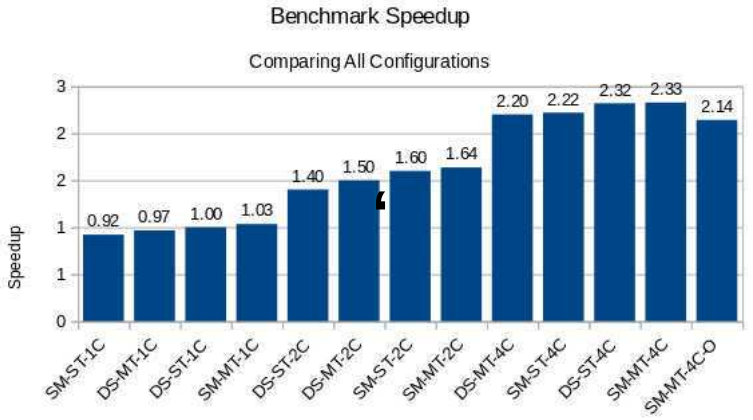
#4. Implementacao de aplicacao para demonstracao (FRONTEND);

PROBLEMAS:

#1. FALTA DE RECURSOS COMPUTACIONAIS:

- MEMORIA E ESPACO EM DISCO PARA PROCESSAMENTO;

#2. CHAVE DE ACESSO AO LABORATORIO LAM DO PESC!!!



Dúvidas