Avaliação de Desempenho de SGBD Matriciais com Séries Temporais de Imagens de Sensoriamento Remoto

Vitor C. F. Gomes[1], Gilberto Ribeiro de Queiroz[2], and Karine Reis Ferreira[2]

[1]Instituto de Estudos Avançados – Departamento de Ciência e Tecnologia Aeroespacial, CEP 12.228-001 – São José dos Campos – SP – Brasil
[2]Coordenação de Observação da Terra – Instituto Nacional de Pesquisas Espaciais, Caixa Postal 515 – CEP 12.227-010 – São José dos Campos – SP – Brasil

16 de junho de 2017

Resumo

Este trabalho apresenta uma avaliação de desempenho do Sistema Gerenciador de Bando de Dados Matriciais SciDB comparando duas estruturas de *array* para um mesmo conjunto de dados quando executado um grupo de consultas. Foram coletados tempos de processamento e métricas de hardware como uso de CPU, memória, rede e disco a fim de identificar os pontos críticos das consultas realizadas. Os casos de teste incluíram a variação do tamanho dos *chunks* dos *arrays*. Com os nossos resultados, foi possível observar que as consultas de seleção de dados obtiveram melhor desempenho com o *array* com *chunk* menor. Para o *array* com *chunk* maior, foi observado melhor desempenho para as consultas que exigiram maior capacidade de processamento.

1 Introdução

Dados geoespaciais são recursos valiosos para análises e aplicações que subsidiam a tomada de decisão em atividades políticas, sociais, econômicas, ambientais e etc. Nos últimos anos, a quantidade disponível desses dados tem crescido, motivados pelos avanços tecnológicos em equipamentos de aquisição de dados e impulsionados pelo aumento da capacidade de armazenamento dos sistemas computacionais. Se por um lado a disponibilidade dessas informações tem permitido que novos avanços científicos e tecnológicos possam acontecer, o armazenamento, o gerenciamento e a utilização desses dados de forma eficiente representa um desafio devido a sua multidimensionalidade, distribuição irregular, densidade e seu grande volume. Em especial, o gerenciamento e análise de dados científicos tem sido um dos grandes desafios das tecnologias de bancos de dados, uma vez que estas tecnologias avançaram, principalmente, no segmento empresarial, e deixaram de atender, por muito tempo, requisitos da comunidade científica.

Na área de Observação da Terra, a maioria dos métodos de análise de dados atualmente ainda é baseada em arquivos. Em aplicações que demandam grandes volumes de dados, os usuários precisam obter centenas (ou milhares) de arquivos, para que um programa possa extrair e armazenar as informações relevantes na memória do computador ou em arquivos intermediários. Com o aumento do volume de dados, o uso desse tipo de abordagem para a análise de grandes volumes de dados será cada vez mais lenta. De maneira geral, esta prática tem colocado limites severos sobre os usos científicos de dados da Observação da Terra [Camara et al., 2014, Cudre-Mauroux et al., 2009].

Na década de 90, já eram reconhecidos os desafios da utilização de tecnologias de bancos de dados para aplicações científicas. Maier and Vance [1993] observaram que eram raras as aplicações científicas que utilizavam sistemas gerenciadores de banco de dados. Segundo os autores, uma das principais razões para isso era a falta de uma representação natural para os dados científicos nesses sistemas. Para grande parte dessas aplicações, o modelo de dados mais apropriado aos dados científicos são as matrizes multidimensionais, uma vez que dados científicos são intrinsicamente ordenados de forma espacial e/ou temporal [Maier and Vance, 1993, Rusu and Cheng, 2013, Papadopoulos et al., 2016]. Matrizes multidimensionais são produzidas na coleta de dados de sensores, imagens ou dados estatísticos. Na área de Observação da Terra, essa estrutura de dados é encontrada em séries temporais 1D, imagens 2D, séries temporais 3D ou dados atmosféricos 4D [Baumann

and Holsten, 2011, Papadopoulos et al., 2016].

Nos últimos anos, a busca por atender esse nicho específico tem estimulado o desenvolvimento de novas tecnologias, conhecidas como Sistemas Gerenciadores de Banco de Dados Matriciais[1] (SGBD-M). Esses SGBDs são projetados para manipular dados densos e/ou esparsos e utilizam matrizes multidimensionais como modelo central de dados. A concepção desses sistemas tem levado em consideração, desde o início, uma forte preocupação com questões como escalabilidade, particionamento dos dados (horizontal e vertical), replicação e a operação em clusteres computacionais [Cudre-Mauroux et al., 2009, Papadopoulos et al., 2016].

Neste contexto, este trabalho tem como objetivo explorar SGBD-M a fim de (i) identificar os recursos disponíveis para a manipulação e o armazenamento de dados matriciais; e (ii) avaliar o desempenho quando utilizados para o gerenciamento e análise de Séries Temporais de Imagens de Sensoriamento Remoto. Para a análise de desempenho, este trabalho vai além da coleta do tempo necessário para a realização de tarefas, para isso, são medidos: (i) tempo e uso de recursos do hardware (CPU, memória, rede e disco) para a realização de operações como carga e processamento dos dados; (ii) espaço em disco utilizado para armazenar o conjunto de dados; e (iii) impacto de escolhas de parâmetros de particionamento dos dados.

No restante deste trabalho é feita uma revisão sobre Sistemas Gerenciadores de Banco de Dados Matriciais, na seção 2, e são apresentados detalhes do SciDB, na subseção 2.1. Na sequência, é apresentada a metodologia empregada para a realização na avaliação do SciDB (seção 3), onde são descritos o ambiente computacional, as ferramentas, os procedimentos utilizados e os detalhes dos testes definidos. Para finalizar, são apresentados os resultados, na seção 4, e as considerações finais, na seção 5.

2 Sistemas Gerenciadores de Banco de Dados Matriciais

Diferentemente de Sistemas Gerenciadores de Banco de Dados Relacionais, que se baseiam no conceito de Relações (ou Tabelas) e na álgebra relacional, SGBD-M são centrados em Matrizes Multidimensionais (*Arrays*). A

[1]Tradução adotada para o nome original em Inglês desta tecnologia: *Multi-Dimensional Array Databases*.

diferença entre Relações e Matrizes Multidimensionais pode ser melhor entendida pela distinção entre dimensões e atributos. Uma relação pode ser considerada uma matriz adimensional de atributos, onde não existe uma função de ordenação que permite a identificação de uma tupla considerando 2 ou mais índices dimensionais. Por outro lado, em Matrizes Multidimensionais é necessário que índices dimensionais formem um chave para endereçar a relação correspondente. Nesta última abordagem, existe dependência funcional dos atributos de dimensão com os demais atributos da relação [Rusu and Cheng, 2013].

Tanto Matrizes Multidimensionais quanto Relações podem ser ser endereçadas por $[d_1, d_2, \ldots, d_N]$, onde d_i representam cada dimensão no domínio das Matrizes ou uma chave no escopo das Relações. O que distingue essas duas abordagem é a forma como os dados são organizados. A partir de $[d_1, d_2, \ldots, d_N]$ é possível acessar diretamente os valores da Matriz. O mesmo não é valido para Relações, pois, neste caso, não existe uma correspondência direta entre as índices e a localização da representação física dos dados [Rusu and Cheng, 2013].

Baumann [2017] lançou recentemente (Junho/2017) um manifesto cujo objetivo é esclarecer definições e termos usados na área de gestão de *arrays* multidimensionais. O autor inicia definindo o termo *datacube*, da seguinte forma: Um **datacube** é um *array* multidimensional massivo, também chamado de *raster data* ou *gridded data*. O termo massivo está relacionado a tamanhos significativamente superiores que a memória princial do servidor. Os valores do *array*, todos do mesmo tipo de dados, estão posicionados nos pontos de uma grade definida por $n-eixos$ de um $datacube\ n-dimensional$.

Atualmente, os principais sistemas para a gestão de dados matriciais são o RasDaMan [Baumann et al., 1998], o TileDB [Papadopoulos et al., 2016] e o SciDB [Brown, 2010].

O RasDaMan, de **Ras**ter **Da**ta **Man**ager, é uma das primeiras implementações de SGBD-M. Esse sistema é fruto de pesquisas iniciadas em 1989 pelo Prof. Peter Baumann. Atualmente é desenvolvido por uma *spin-off* Alemã. Esse sistema funciona como um *midware* entre uma aplicação cliente e um Sistema Gerenciador de Banco de Dados Relacional. Esse *midware* é responsável por fazer esse mapeamento entre a abstração dos dados como um **datacube** e os dados armazenados no SGBD-R. A Figura 1 ilustra a arquitetura básica desse sistema.

Na Figura 1, observa-se que o serviço do RasDaMan é responsável por acessar o SGBD-R, o qual gerencia o respositório de dados. Como SGBD-

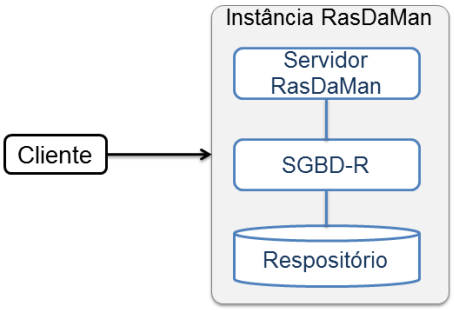

Figura 1: Arquitetura do RasDaMan.

R, o RasDaMan fornece suporte ao PostgreSQL e ao SQLite. Quando o PostgreSQL é usado como gerenciador, os metadados dos arrays são armazenados em tabelas e os arrays em campos do tipo *blob*. No caso de uso do SQLite, o funcionamento é um pouco diferente. Nessa situação, o RasDaMan somente armazena os metadados no SQLite, enquanto que os arrays são armazenados diretamente no sistema de arquivos. O RasDaMan também tem suporte a execução em cluster de computadores. Esse suporte é fornecido para a versão empresarial (*Enterprise*), a qual é paga. Nessa versão, várias instâncias do RasDaMan podem ser utilizadas para gerenciar um *array* distribuídos em computadores em rede. A versão fornecida gratuitamente (*Community*) não possui esta funcionalidade [Baumann et al., 1998].

O TileDB é uns dos mais recentes SGBD-M lançados. Sua primeira versão foi disponibilizada em Abril de 2016. Ele é desenvolvido através de uma colaboração entre o Centro de Ciência e Tecnologia para Big Data da Intel [INTEL, 2017] e o Laboratório de Ciência da Computação e Inteligência Artificial do MIT [Massachusetts Institute of Technology, 2017]. Esse sistema é disponibilizado como uma biblioteca em C, para ser integrada às aplicações dos usuários. O TileDB utiliza pastas para organizar os *arrays*, onde cada atributo é separado em um arquivo diferente. O usuário é responsável por indicar, durante a criação do *array*, como os dados serão quebrados em blocos menores (*chunks*) e em que ordem os dados devem ser gravados, varrendo a matriz horizontalmente ou verticalmente, para cada uma das dimensões [Papadopoulos et al., 2016]. Este sistema fornece um grau de abstração menor em relação ao RasDaMan e ao SciDB, pois a forma como os dados são ar-

mazenados em disco é menos transparente para o usuário.

O SciDB teve sua primeira versão lançada em 2008, sendo considerado um dos mais promissores SBGD-M da atualidade e um sistema no estado da arte [Rusu and Cheng, 2013]. É desenvolvido pela empresa Paradigm4 e distribuído em duas versões: *Community* e *Enterprise*. Em ambas as versões é possível executar o SciDB em um cluster de computadores. Os recursos extras fornecidos na versão *Enterprise*, a qual é paga, incluem suporte técnico, funções matemáticas e de aprendizado de máquina [Brown, 2010].

Considerando as funcionalidades e características do SciDB, este trabalho está focado no seu estudo e avaliação. Por conta disso, a subseção a seguir foi destinada a apresentação mais detalhada desse sistema.

2.1 SciDB

O SciDB é um sistema de código fonte livre para gerenciamento e análise de dados científicos armazenados como matrizes multidimensionais. Foi concebido para dar suporte a execução em cluster de computadores e fornece ferramentas que facilitam a carga e o processamento dos dados através de scripts em linguagem R, python e C/C++ [Brown, 2010].

Este sistema divide uma grande matriz em pedaços denominados *chunks* que são distribuídos entre diferentes instâncias do *cluster* de servidores. A Figura 2 mostra a arquitetura básica do SciDB.

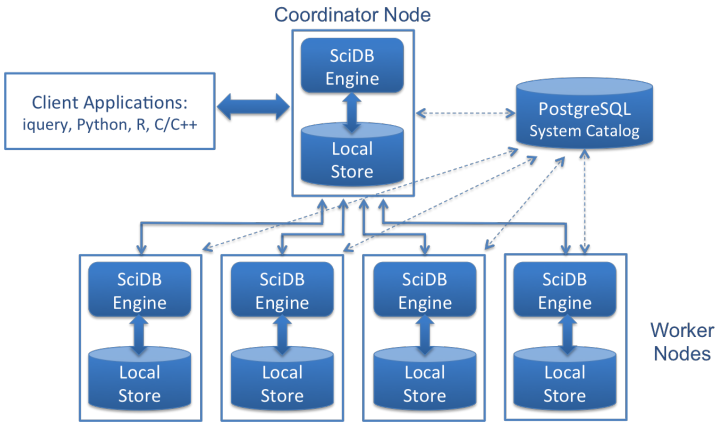

Figura 2: Arquitetura do SciDB.
Fonte: Adaptada de Paradigm4 [2016].

Cada servidor controla um conjuto de dados locais. Uma das instâncias é designada coordenadora (*coordinator*), sendo responsável por mediar toda a comunicação entre o *cluster* e as aplicações clientes, e também por orquestrar a execução das consultas. As demais instâncias do *cluster*, denominadas trabalhadoras (*workers*), apenas participam do processamento da consulta.

O SciDB explora o modelo de dados matricial para fornecer um mecanismo eficiente de armazenamento baseado em *chunks* e no particionamento vertical por atributos da matriz [Brown, 2010]. Além disso, possui duas linguagens de consulta de alto nível:

- **Array Query Language (AQL)**: linguagem parecida com a SQL, em que as cláusulas se apoiam nos conceitos de matrizes, dimensões e atributos;
- **Array Functional Language (AFL)**: linguagem que possibilita expressar consultas através da composição de funções.

A AQL possui comandos para a criação e carga de *arrays* (Data Definition Language - DDL) e comandos para para acessar e operar os dados dos *arrays* (Data Manipulation Language - DML). A linguagem AQL é traduzida, pelo processador de consultados do SciDB, em AFL antes de ser executada [Paradigm4, 2016].

2.2 Modelo de Dados

O modelo de dados do SciDB se baseia no conceito de *Arrays* e não em *Relações* (ou Tabelas). Um *array* possui um nome, a definição dos atributos das células e as dimensões do array:

```
name <atributos> [dimensões]
```

A linguagem de consulta do SciDB é baseada em uma *álgebra de arrays*, com operações de filtragem das células podendo ser expressas por valores dos atributos ou pela dimensão.

Em geral, são utilizados valores inteiros de 64-bit para definir as dimensões de um *array*. Os *array* podem ter dimensões com limites bem definidos (*bounded dimension*), quando define-se a priori o número total de células em uma dada dimensão (ou a cardinalidade da dimensão).

Quando não é conhecida a cardinalidade do *array* em tempo de criação, para uma de suas dimensões, limite é deixado indefinido (*unbounded dimension*).

A definição de um *array* chamado `mod13q1` com as dimenões em j, i e t $= (4, 3, *)$ pode ser feita da seguinte forma:

```
mod13q1
<atributos>
[j=1:4,2,1, i=1:3,1,1, t=1:*,3,2]
```

Cada Célula de um *array* no SciDB pode estar associada a múltiplos valores, cada um pertencente a um tipo de dados específico: int8, uint8, int16, uint16, int32, uint32, int64, uint64, float, double, string, datetime, datetimetz, etc.

Uma célula é definida pelos índices de sua dimensão: $celula = (d_1, d_2, ..., d_n)$

A definição de um *array* chamado `mod13q1` com três atributos pode ser realizada da seguinte forma: *array* `mod13q1` é:

```
mod13q1
<red:int16, nir:int16, quality:uint16>
[j=1:4,2,1, i=1:3,1,1, t=1:*,3,2]
```

O SciDB possui dois mecanismos de particionamento dos dados:

- Particionamento de um *array* em *chunks*. Cada *chunk* é armazenado em uma instância do *cluster*, o que facilita a paralelização de computações.

- Particionamento dos atributos de forma vertical (*vertical partitioning*). Essa estratégia parte do princípio que os *arrays* podem ser formados por células com muitos atributos e que as consultas nem sempre irão conter expressões com todos eles. A separação física dos valores de cada atributo pode fazer com que haja uma menor movimentação de dados entre o disco e a memória RAM, acelerando assim o processamento de consultas.

O SciDB não utiliza uma Árvore-B^+ ou Árvore-R para indexação das células. As dimensões formam a base da indexação dos dados.

O dado é dividos em *chunks* e mapeados através de uma função *hash* para cada instância do *cluster*.

Desta forma, uma questão importante trata-se de como especificar o número de células ao longo de cada dimensão que será usado para estabelecer o tamanho dos *chunks* de um *array*.

2.3 Replicação

O SciDB permite definir um fator chamado de *chunk overlap* para as células de borda dos *chunks*. Trata-se de uma boa estratégia para acelerar consultas envolvendo vizinhança (*neighborhood*).

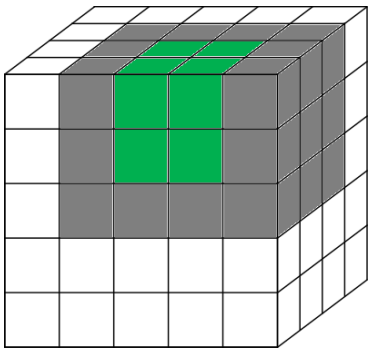

Figura 3: Esquemas de particionamento de dados no SciDB.

A Figura 3 apresenta uma ilustração da replicação das células de borda para um *array* 5 x 5 x 5, com um *chunk* de tamanho 2 x 2 x 2 e *chunk overlap* de 1 ao longo de cada dimensão.

Outro tipo de replicação existente no SciDB diz respeito à replicação dos *chunks* entre as instâncias do *cluster*. Trata-se de uma forma de aumentar a tolerância a falhas. Essa funcionalidade permite recuperar os dados de um array caso uma das instâncias perca seus dados. Entretanto, por conta da replicação dos dados, há aumento significativo no espaço ocupado pelo *array*.

3 Metodologia

Nesta seção, é apresentada a metodologia utilizada para a realização dos testes deste trabalho. Inialmente é feita a apresentação do ambiente computacional e das ferramentas utilizadas, na subseção 3.1 e, em seguida, é feita a descrição do conjunto de dados utilizados, subseção 3.2. Os métodos empregados para a carga dos dados são apresentados na subseção 3.3, enquanto que a subseção 3.4 apresenta o processo utilizado para a execução dos testes e as consultas realizadas.

3.1 Ambiente Computacional e Ferramentas

Para a realização dos testes foi utilizado o cluster e-sensing, do projeto de mesmo nome [INPE, 2017]. Este custer possui 5 servidores conectados por rede Gigabit Ethernet, cada um contendo 2 processadores Intel Xeon E5-2620 v3 de 2.4GHz, 96GB de memória RAM e 16 discos com 1 TB cada. O sistema operacional utilizado nos servidores é o Ubuntu 14.04 LTS.

Em casa servidor deste cluster são executadas 8 instâncias do SciDB versão 15.12, totalizando 40 instâncias em execução em todo o cluster. Cada instância possui acesso dedicado a seu próprio disco para evitar concorrência durante leituras e gravações.

Para a coleta das métricas monitoradas durante os testes, foi utilizado o sistema Zabbix [Zabbix, 2017], na versão 3.2.2.

O Zabbix é um software largamente difundido para fazer o monitoramente de servidores e recursos de rede para o controle de infraestrutura de Tecnologia da Informação (TI). Esse sistema possui código fonte livre, é distribuído sob licença *GNU General Public License* versão 2, tem interface baseada na web e armazena, por padrão, os dados coletados em banco de dados MySQL. A configuração do Zabbix no cluster e-sensing foi realizada da seguinte forma: (i) em uma máquina externa ao cluster (servidor TerraMA2) foi instalado e configurado o *Zabbix Server*, responsável por recolher os dados dos agentes, por armazenar, produzir estatísticas e gráficos dos dados; (ii) em cada servidor do cluster e-sensing foi instalado e configurado o *Zabbix Agent*, responsável por coletar e enviar ao *Zabbix Server* as métricas monitoradas; e (iii) foi criado um *template* com uma configuração padrão dos parâmetros a serem monitorados em cada servidor.

Por padrão, o Zabbix permite a coleta de informações como uso de CPU (total, por processo, por usuário, etc), uso de rede (entrada, saída e por interface de rede), número de processos (total, por usuário, etc), entre outros parâmetros. Além disso, é possível extender suas funcionalidades através da inclusão de *templates*. Para a coleta de parametros de uso dos discos, foi utilizado a extensão *Zabbix Disk Performance Template* [Chernyshev, 2017], a qual fornece opções para a coleta da quantidade de leituras/gravações, número de setores lidos/gravados e quantidade de Bytes lidos/gravados em cada disco monitorado.

3.2 Conjunto de Dados

Para os testes, foi utilizado um conjunto de imagens do Produto Índice de Vegetação (MOD13Q1) do sensor MODIS que cobrem a América do Sul. Esse produto fornece índices projetados para permitir a realização de comparações espaciais e temporais consistentes das condições da vegetação, sendo frequentemente utilizado em análises de cobertura florestal, cortes seletivos, agricultura e etc. Esse produto MODIS é fornecido através de imagens com 4800 por 4800 pixels, com resolução espacial de 250m e resolução temporal de 16 dias [USGS, 2017]. Na Figura 4, é destacada em vermelho a região utilizada neste trabalho, que cobre grande parte da América do Sul e é composta por 31 quandrantes.

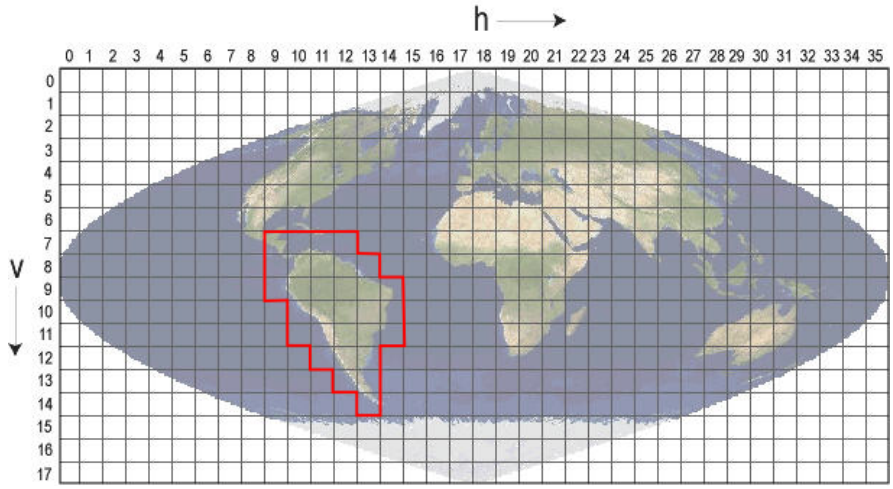

Figura 4: Quadrantes MODIS utilizados para a realização dos testes destacados em vermelho
Fonte: Adaptado de [USGS, 2017]

Para os testes, foi selecionada uma série temporal de 20 instantes do ano 2000, totalizando 620 arquivos HDF (31 quandrantes * 20 instantes).

Cada arquivo do produto MOD13Q1, no formato nativo, possui entre 70 a 200MB e, em formato bruto descompactado, ocupa 505,37MB (4800px * 4800px * 23bits).

Na Figura 5, é apresentado um mosaico com 31 quandrantes de um corte no tempo para o atributo evi do produto MOD13Q1, onde é possível ver os índices dos auadrantes utilizadas neste trabalho.

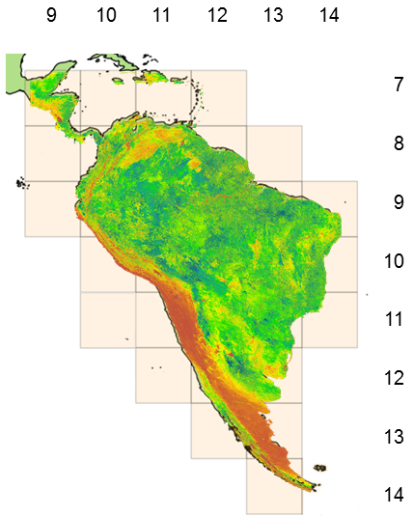

Figura 5: Quadrantes MODIS utilizados para a realização dos testes destacados em vermelho

3.3 Carga de Dados

O processo para a carga dos dados no SciDB é ilutrado na Figura 6.

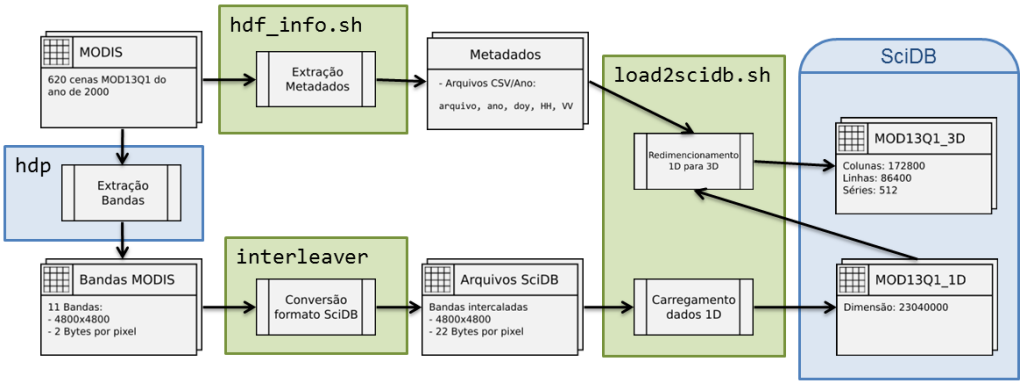

Figura 6: Processo usado para a carga de dados

Para cada arquivo do conjuto de dados MOD13Q1, é realizada a extração dos metadados através do nome do arquivo. Nesta fase, é produzido um arquivo CSV com as seguintes informações sobre o arquivo: nome, ano, dia do ano (doy), índice horizontal do quadrante e índice vertical do quadrante.

Na sequência, é realizada a extração de bandas 11 do arquivo HDF. As bandas escolhidas utilizadas são: ndvi, evi, quality, red, nir, blue, mir, view zenith, sun zenith e relative azimuth, composite doy.

Após esse processo, cada atributo da banda é intercalado formando um único arquivo bruto com todas as bandas. Esse processo é feito para atender o formato esperado pelo SciDB durante a carga dos dados.

A carga do arquivo produzido no SciDB é realizada inicialmente em um *array* unidimensional, como recomendado pela documentação do sistema. O redimensionamento desse *array* 1D para o formato desejado 3D (x, y, tempo) é realizado através do comando *redimension* e utilizando as informações dos índices dos quadrantes e dia do ano, extraídos anteriormente. Esse processo foi repetido para cada arquivo do conjunto de dados escolhido para análise.

Na Figura 6, são destacados em azul as ferramentas de terceiros utilizadas nesse processo, como o próprio SciDB e a aplicação hdp, a qual foi utilizada para a extração de bandas dos arquivos HDF. Em verde, estão destacadas os scripts e aplicações desenvolvidos para automatizar o processo de carga dos dados.

3.4 Execução dos Testes

O processo para a execução de testes no SciDB é ilutrado na Figura 7.

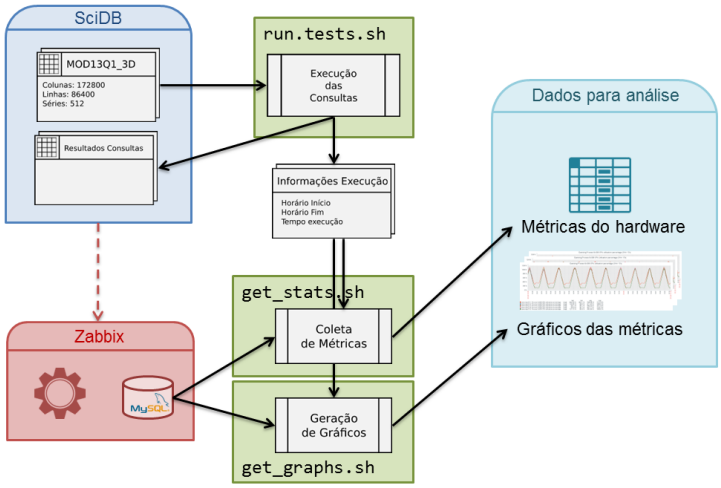

Figura 7: Processo usado para execução dos testes

Com o conjunto de dados carregado no SciDB, cada consulta é realiza e

são armazenados os tempos de início, fim e total. A partir desses valores, é feita a extração das métricas coletadas pelo Zabbix, diretamente no banco de dados MySQL utilizado por este sistema. Cada conjunto de métrica coletado para o período da execução da consulta produz uma planilha de dados, armazenada no formato CSV. Além das disso, também é realizada a produção de gráficos das métricas de hardware em análise. O Zabbix possui uma API (*Application Programming Interface*) para essa funcionalidade.

Na Figura 7, estão destacados em verde as procedimento automatizados através da criação e scripts.

3.4.1 Consultas

Para a avaliação de desempenho do SciDB foram preparadas 7 consultas. O objetivo principal delas é avaliar questões como a seleção de grandes conjuntos de dados variando os planos de corte e a realização de processamento com os dados. As consultas realizadas são:

1. **Horizontal**: é realizado um corte no conjunto de dados pegando toda a dimensão horizontal, criando um plano no eixo horizontal-temporal;

2. **Vertical**: é realizado um corte no conjunto de dados pegando toda a dimensão vertical, criando um plano no eixo vertical-temporal;

3. **Timewise**: é realizada a seleção de um pixel em toda a dimensão temporal;

4. **Subset**: é realizada a seleção de um subconjunto do *array*, fazendo corte nas três dimensões;

5. **EVI**: a partir das bandas nir, red e blue, é realizado o cálcudo do EVI e gerado um novo *array* de mesma dimensão, com um único atributo resultante do cálculo;

6. **Quantile**: é realizado o calculo de 10 quantis (decis) para o array resultante da consulta anterior; e

7. **Window AVG 3x3**: é realizado o cálculo de média utilizando uma janela 3x3 sob o conjunto de dados da consulta 5.

3.5 Testes

Para realizar a avalição de desempenho do SciDB na execução das consultas escolhidas, foram definidos 2 testes variando a forma como dos dados são armazenados. Para isso, foi feita a carga de dois *arrays* com diferentes tamanhos de *chunk*. Para o primeiro teste (**Teste 1**) foram definidos *chunks* com 40 células nos eixos horizontal e vertical. Para o **Teste 2**, foram definidos *chunks* de tamanho 2400 nas dimensões horizontal e vertical. Para ambos os testes foi utilizado 20 células para o eixo temporal, o que representa todo o domínio. Desta forma, no Teste 1, os *chunks* possuem 62,5KB, enquanto que para no Teste 2, possuem 219MB.

Para a coleta de espaço em disco ocupado foi utilizada as informações fornecidas pelo SciDB através da seleção apresentadam na Listagem 1. Nesta consulta, são retornados os espaços em disco ocupado pelo *array* em cada instância do SciDB. As saídas são usize, csize e asize.

O valor UAID representa o identificador únido dado pelo SciDB para o *array*, usize é o espaço ocupado pelo *array* quando descomprimido, csize é o espaço ocupado quando comprimido e asize é o espaço total alocado em disco pelo SciDB para alocar o *array*. O SciDB utilizada uma pré-alocação de espaço em disco para otimizar a gravação dos dados em disco. Ele faz isso alocando espaços utilizando um tamanho que seja sempre potência de 2. Quando não é utilizada compreessão dos dados, usize é igual a csize, como no caso dos testes realizados neste trabalho. Para a análise, foi considerado o valor de asize, uma vez que representa o real espaço em disco necessário para armazenar os dados.

Listagem 1: Seleção de espaço ocupado em disco por um array

```
aggregate( filter(list('chunk map'), uaid = [UAID]),
           sum(usize), sum(csize), sum(asize), inst);
```

Os tempos foram coletados através do comando *date +%s* do linux.

Para a análise de desempenho, foram selecionadas os seguintes parâmetros coletados pelo Zabbix:

1. **CPU Global**: uso total de CPU em cada máquina do cluster;
2. **CPU por Processo**: uso de CPU para os processos SciDB, PostgreSQL e R. O processo R foi incluído para verificar se em algum momento o processamento é realizando utilizando a linguagem R, suportada pelo SciDB;

3. **Memória por Processo**: uso de memória para os processos SciDB, PostgreSQL e R;
4. **Leitura de Disco**: quantidade de setores lidos de cada disco de cada servidor;
5. **Gravação em Disco**: quantidade de setores gravados em cada disco de cada servidor;
6. **Dados recebidos pela rede**: quantidade de bits recebidos pela interface de rede; e
7. **Dados enviados pela rede**: quantidade de bits enviados pela interface de rede.

4 Resultados e Discussões

A partir da carga do conjunto de dados para os dois testes, considerando os diferentes tamanhos de chunks, foi possível obter o tempo total de carga e o espaço ocupado por cada teste. O tempo necessário para carregar os 620 arquivos HDF foi de 36,13 horas para o Teste 1 e de 54,59 horas para o Teste 2. Observa-se uma aumento de pouco mais de 51% para o teste com *chunks* maiores. Este fato pode estar associado principalmente a fase de redimensionamento do *array* unidimensional para 3 dimensões. Como os chunks são maiores, o potencial de paralelizar esse processo é menor, uma vez que existem menos *chunks* e que podem estar distribuídos de maneira desigual entre as instâncias dos cluster.

Quanto ao espaço ocupado em disco, no Teste 1 ocupou 988,96 GB, enquanto que no Teste 2 foram necessários 1024,06 para armazenarem o conjunto de dados. Observa-se que a variação foi pouco superior a 3,5% maior no segundo caso. Isso pode estar relacionado a pré-alocação em potência de 2 realizada pelo SciDB para armazenar os chunks.

Quanto ao tempo de execução das consultas, as 5 primeiras, Horizontal, Vertical, Timewise, Subset e EVI, tiveram tempos menores no Teste 1 em relação ao Teste 2. A Figura 8 apresenta um gráfico de barras com os dados. O eixo vertical é o tempo utilizado para a execução das consultas e está em segundos.

Destaca-se, inicialmente, que não foi possível realizar a consulta Subset para o *array* do Teste 2. Durante a execução era retornado um erro de falta

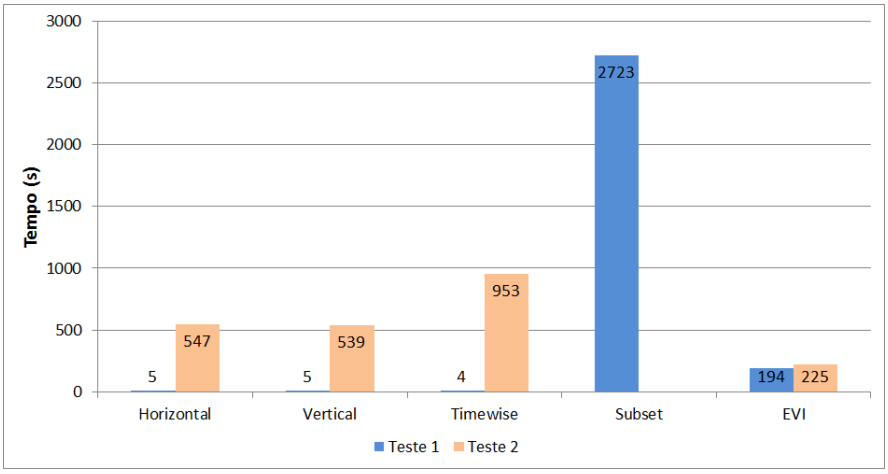

Figura 8: Tempos para as consultas Horizontal, Vertical, Timewise, Subset e EVI.

de memória, apesar de ainda haver memória disponível no servidor. Esse tipo de erro também foi encontrado por outros usuários e reportados no fórum do SciDB [Poliakov, 2017]. Como reportado por um funcionário da Paradgm4, algumas vezes o SciDB age como se houvesse algum vazamento de memória.

Para os outros resultados vistos na Figura 8, observa-se que o Teste 1 foi muito mais rápido para os testes Horizontal, Vertical e Timewise, e um pouco melhor para o teste EVI. Os três primeiros testes são praticamente a seleção de chunks e a criação de um novo *array*.

Na Figura 9, são apresentados os resultados para as consultas Quantile e Window AVG 3x3.

Para essas consultas, observa-se que o Teste 2 obteve melhores tempos em relação ao Teste 1. Essas consultas demandam mais processamento quando comparadas com as 5 primeiras, em especial a consulta Quantile, que exige que os dados sejam ordenados.

Considerando o total de métricas observadas e o número de servidores do cluster, 119 planihas de métricas foram coletadas e 28 gráficos foram gerados para cada consulta, os quais totalizam 833 planilhas e 196 gráficos, para as 7 consultas realizadas.

Considerando o escopo deste trabalho, a análise das métricas foram feitas, principalmente, através da avaliação dos gráficos e das médias das métricas, devido ao grande volume de informações a ser analisado.

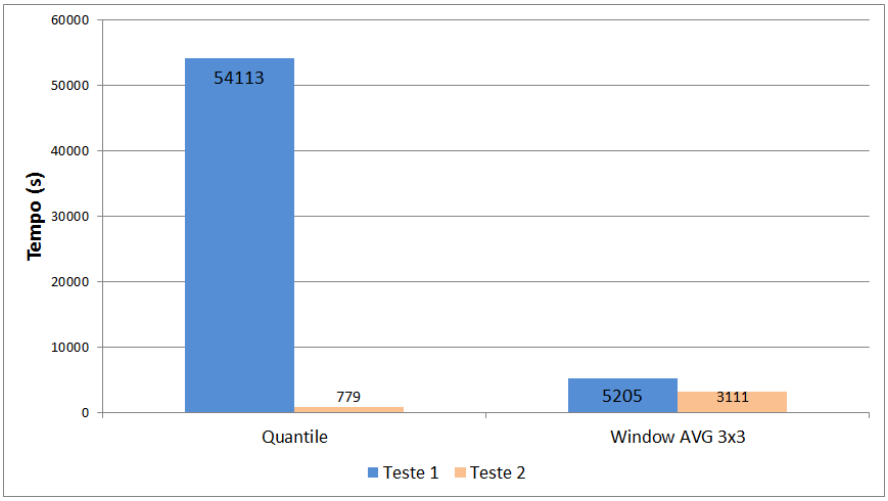

Figura 9: Tempos para as consultas Quantile e Window AVG 3x3.

Na primeira fase da análise dos resultados, verificou-se que houve pouca variação no uso de memória durante a execução das consultas. Em casa servidor, os processos SciDB usaram, em média, 20GB de memória durante toda a execução de cada consulta. Essa fato pode estar associado a uma política de retenção de memória usada pelo SciDB, para evitar que os processos fiquem alocando e desalocando memória, melhorando, assim, o desempenho global da aplicação. Quando o SciDB mantém memória alocada, mas sem utilizá-la diretamente para o processamento de uma consulta, o Zabbix não consegue identificar este fato, pois somente tem acesso ao volume destinado pelo sistema operacional para aquele processo. Por conta desses fatos, não serão apresentados os gráficos referentes ao consumo de memória durante a execução das consultas.

No momento seguinte, observou-se que gráficos usos de CPU, Memória, Rede e Disco não variaram significativamente entre os 5 servidores para uma mesma consulta. Apesar de haver variações nos valores, o compartamento geral são equivalentes e as médias mantidas semelhantes. Desta forma, somente serão apresentados os gráficos referentes a um servidor do cluster para cada métrica como forma de não poluir essa seção.

Outro fator observado quando avalia-se as métricas das consultas é um comportamento semelhante para as cinco primeiras consultas e outro comportamento para as últimas duas consultas. Basicamente a diferença entre esses dois grupos está no volume de processamente demandado por elas,

sendo que o segundo grupo demanda mais intensamente o uso de CPU.

Desta forma, optou-se por apresentar neste trabalho um caso de cada um dos dois grupos de consultas, como forma de ilustrar tais comportamentos. As consultas escolhidas foram **Horizontal** e **Windows AVG 3x3**. As subseções a seguir apresentam com mais detalhes os resultados de cada uma dessas consultas.

4.1 Consulta Horizontal

Os gráficos gerados a partir das métricas coletas durante a execução da consulta Horizontal para o Teste 1 são apresentados no Apêndice A, enquanto que os do Teste 2 estão no Apêndice B.

Na Figura 22, observa-se que durante a consulta Horizontal no Teste 1, pouco de CPU é utilizado. Além disso, verifica-se que a carga está igualmente distribuída entre os servidores do cluster.

Na Figura 23, a qual apresenta o uso de CPU por processo, verifica-se que o consumo é dominado pelo processo SciDB. O processo PostgreSQL tem leve alto no início da execução da consulta, provavelmente durante a fase onde ocorre a consulta pelos metadados dos *chunks* no catálogo global do SciDB.

Os gráficos das quantidades de setores lidos e gravados para cada disco de um servidor do cluster e-sensing são apresentados, respectivamente, nas Figuras 24 e 25. Há comportamento semelhante tanto para leitura quanto para gravação de setores, onde existe um consumo elevado no início da consulta, seguido por um uso constante dos discos. Destaca-se que o uso dos discos é homogêneo para ambos os modos, indicando que os *chunks* estão bem distribuídos entre as instâncias do servidor.

O uso da interface de rede também segue o mesmo comportamento para os dados de entrada quanto de saída. As Figuras 32 e 27 apresentam, respectivamente, os dados recebidos e os dados enviados pelos servidores do cluster e-sensing durante a execução da consulta Horizontal para o Teste 1. Observa-se um aumento do uso de rede na segunda metada de consulta, indicando que os chunks podem estar sendo distribuídos para a construção do novo *array*, resultante da consulta. O uso da rede também é homogêneo entre os servidores, mantendo uma média em torno de 2,78Mbps.

A Figura 28 apresenta o uso de CPU global dos servidores durante a execução da consulta Horizontal para o Teste 2. Neste experimento, é observado maior consumo de CPU em relação ao Teste 1 e uma variação maior

de uso entre os servidores do cluster. Além disso, observa-se pequenos vales onde o uso de CPU é reduzido. Esses vales estão associados a picos de leitura de disco, que serão apresentados na Figura 30. O comportamento do uso de CPU pelo processo SciDB continua dominante neste teste, seguido pelo processo PostgreSQL, que apresenta uso baixíssimo, abaixo de 1%. O processo R, assim como no caso do Teste 1, não apresenta qualquer uso de CPU.

Quanto a leitura de disco, verifica-se, na Figura 30, que a distribuição de leituras difere da observada no Teste 1. No Teste 2, já picos mais intensos de leitura e nem todos os discos são usados para a realização da consulta. Este fato indica que há uma distribuição não homogênea dos *chunks* ou que, dado o maior tamanho dos blocos armazenados em disco, um menor número de blocos foi acessado para o recorte, e que nem todas as instâncias possuiam dados para retornar na consulta.

Quanto a escrita de setores nos discos (Figura 31), ela é melhor distribuída no tempo e entre os discos das instâncias do SciDB em execução.

De maneira geral, podemos considerar observando os dados coletados durante a consulta que faz a seleção de uma porção horizontal no *array* que *chunks* menores permitiram um melhor uso do hardware. Além disso, que esses blocos foram melhor distibuídos entre as instâncias em execução e que *chunks* maiores causaram um maior gargalo durante a leitura, uma vez que os blocos maiores exigiram a leitura de maiores volumes de dados.

4.2 Consulta Window AVG 3x3

Os gráficos gerados a partir das métricas coletas durante a execução da consulta Window AVG 3x3 para o Teste 1 são apresentados no Apêndice C e os gráficos para o Teste 2 dessa consulta no Apêndice D.

O uso de CPU global pelos servidores durante a execução do teste Window AVG 3x3 no Teste 1 são apresentados na Figura 22. Verifica-se que o há duas fases distintas que fazem uso intenso de CPU, separadas por um vale de uso reduzido. Em média, foi utilizado pouco mais de 30% da soma da capacidade de todas as CPUs no período da consulta. A Figura 23 apresenta o uso por processo. Verifica-se que o processo SciDB teve picos de uso de 1377,6% e CPU, o que indica o uso de quase 14 CPUs de um servidor completamente. Essas caracteríscas nos expõem a demanda por CPU necessária para a realização desta consulta.

Para os dados de leitura e escrita em disco, ilustrados respectivamente pelas Figuras 24 e 25, também detecta-se duas fases. Na fase inicial há bas-

tante leitura, seguido por um segundo momento com baixíssimo uso. Para a escrita, também há abundante leitura na primeira fase, seguido por um segundo momento com escritas menos intensas. Em ambos os casos, é homogênea a distribuição de uso entre os discos do servidor.

Quando observamos os dados de uso de rede, Figuras 32 e 27, verificamos que entre as duas fases identificadas anteriormente, ocorre um pico com intenso uso de rede, tanto de entrada quanto de saída nos servidores. Os picos chegam a 1,61Gbps, matendo médias próximas a 7Mbps.

As características encontradas nas métricas apresentadas para a consulta Window AVG 3x3 no Teste 1 nos leva a crer que durante a primeira fase, ocorre o cálculo e leitura das células que deverão ser compartilhadas entre as intâncias para o processamento da média usando uma janela 3x3. As gravações que ocorrem nesta fase indicariam a gravação em arquivos intermediários dos valores a serem distribuídos entre os servidores. O pico de uso de rede indicaria o momento dessa distribuição dos valores das bordas dos *chunks*, necessários para o cálculo da média local.

Quando observamos as métricas coletadas durante a execução da consulta Window AVG 3x3 para o Teste 2, encontramos um comportamento diferente do Teste 1. A Figura 28 apresenta um uso global de CPU sem a definição clara de duas fases, como acontece no Teste 1. Observa-se que há menor uso médio de CPU, que o verificado no Teste 1, como reforça o conteúdo da Figura 29. O processo SciDB, neste caso, tem pico de uso de CPU de 961,6%.

O uso de disco também é melhor distribuído no tempo, como podemos observar nas Figuras 30 e 31. Apesar disso, há variação entre o quanto cada disco é usado em média durante a consulta. Novamente nos indicando um uso não homogêneo dos recursos quando usados *chunks* maiores.

Para a métrica de rede, as Figuras 32 e 27 apresentam resultados semelhantes entre si. Há uso contínuo da rede, sem haver picos em momentos específicos. Observa-se que em média, usou-se cerca de 500Kbps em ambos os sentidos durante a execução dessa consulta, valor bem inferior aos 7Mbps observados durante a execução com o Teste 1.

Considerando as características da consulta, que requer uso de informações de células vizinhas, há um pior desempenho para a estrutura do *array* que utiliza *chunks* menores. Isso é explicado pelo fato de ser necessário o compartilhamento de mais células vizinhas quando os *chunks* são menores. Nesta situação, o uso intenso de Rede e CPU durante o Teste 1 fizeram com que o tempo da consulta fosse maior que o Teste 2, mesmo quando este último não aproveitou plenamente os recursos de disco, como observado.

Lembra-se, entretanto, que no caso de uso de *chunk overlap* os resultados desta consulta poderiam ser completamente diferentes, uma vez que para janelas pequenas, como utilizada, não haveria a necessidade de compartilhar as céluas vizinhas entre os servidores. Destaca-se, porém, que a penalização pelo uso do *chunk overlap* é o aumento uso de disco, que será em maior escala, quanto menor for o tamanho do *chunk*.

5 Considerações Finais

Este trabalho apresentou o estudo o estudo de SGBD-M e a análise das características de desempenho de um conjunto de consultas no SciDB para dois tamanhos distintos de *chunks*. Para isso, foi preparado um ambiente computacional através da instrumentação de um cluster para a coleta de métricas de hardware. Além disso, foram produzidas ferramentas para conversão de dados HDF para o formato suportado pelo SciDB. As ferramentas desenvolvidas estão disponíveis em `https://github.com/vconrado/avaliacao-sgbd-m`.

Com os resultados obtidos nos testes, verifica-se que o tamanho do *chunk* impacta no espaço ocupado e no desempenho das consultas. No geral, consultas que realizam seleção de dados foram executadas em menor tempo para o array com *chunks* menores, enquanto que consultas que exigiram mais processamento ou acesso a células vizinhas foram executadas mais rapidamente no teste com *chunks* maiores.

Observa-se, desta forma, que não existe um valor ótimo para o particionamento, sendo necessário avaliar os tipos de operações que se desejam realizar para que o array possa ser configurado de maneira a otimizar seu acesso.

Os resultados obtidos neste trabalho nos motivam a estabelecer uma continuidade. Planejamos fazer avaliações de desempenho com diferentes tamanhos de *chunks* e avaliar o impacto do uso de *chunk overlaps*. Consideramos que também seja interessante, avaliar alterar o número de instâncias do SciDB para avaliar se reduzindo o número de instâncias em um mesmo servidor, há ganhos em desempenho por reduzir a quantidade de comunicação.

Referências

P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann. The multidimensional database system rasdaman. *SIGMOD Rec.*, 27(2):575–577, June 1998. ISSN 0163-5808. doi: 10.1145/276305.276386. URL `http://doi.acm.org/10.1145/276305.276386`.

Peter Baumann. The datacube manifesto, 2017. URL `http://earthserver.eu/sites/default/files/upload_by_users/The-Datacube-Manifesto.pdf`. [Online; accessado 09-Junho-2017].

Peter Baumann and Sönke Holsten. A comparative analysis of array models for databases. In *Database Theory and Application, Bio-Science and Bio-Technology - International Conferences, DTA and BSBT 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2001 in Conjunction with GDC 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings*, pages 80–89, 2011.

Paul G. Brown. Overview of scidb: Large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 963–968, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0032-2. doi: 10.1145/1807167.1807271.

Gilberto Camara, Max Egenhofer, Karine Ferreira, Pedro Ribeiro Andrade, Gilberto Queiroz, Alber Sanchez, Jim Jones, and Lubia Vinhas. Fields as a generic data type for big spatial data. *GIScience*, 2014.

Grigory Chernyshev. Zabbix disk performance template, 2017. URL `https://github.com/grundic/zabbix-disk-performance`. [Online; accessado 30-Maio-2017].

P. Cudre-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik. A demonstration of scidb: A science-oriented dbms. *Proc. VLDB Endow.*, 2(2): 1534–1537, August 2009. ISSN 2150-8097. doi: 10.14778/1687553.1687584. URL `https://doi.org/10.14778/1687553.1687584`.

INPE. e-sensing: big earth observation data analytics for land use and land cover change information, 2017. URL http://www.esensing.org. [Online; accessado 30-Maio-2017].

INTEL. Intel science and tecnology center for big data, 2017. URL http://istc-bigdata.org/.

David Maier and Bennet Vance. A call to order. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '93, pages 1–16, New York, NY, USA, 1993. ACM. ISBN 0-89791-593-3. doi: 10.1145/153850.153851. URL http://doi.acm.org/10.1145/153850.153851.

Massachusetts Institute of Technology. Mit computer science and artificial intelligence laboratory, 2017. URL http://www.csail.mit.edu.

Stavros Papadopoulos, Kushal Datta, Samuel Madden, and Timothy Mattson. The tiledb array data storage manager. *Proc. VLDB Endow.*, 10 (4):349–360, November 2016. ISSN 2150-8097. doi: 10.14778/3025111.3025117. URL https://doi.org/10.14778/3025111.3025117.

Paradigm4. Scidb documentation, 2016. URL https://paradigm4.atlassian.net/wiki/display/ESD/SciDB+Documentation.

Alex Poliakov. Error: Not enouth memory, 2017. URL http://forum.paradigm4.com/t/error-not-enough-memory/495.

Florin Rusu and Yu Cheng. A survey on array storage, query languages, and systems. *CoRR*, abs/1302.0103, 2013. URL http://arxiv.org/abs/1302.0103.

USGS. Vegetation indices 16-day l3 global 250m: Mod13q1, 2017. URL https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod13q1.

Zabbix. Zabbix: The enterprise-class monitoring solution for everyone, 2017. URL http://www.zabbix.com/. [Online; accessado 30-Maio-2017].

A Gráficos de métricas coletadas durante a execução da consulta Horizontal do Teste 1

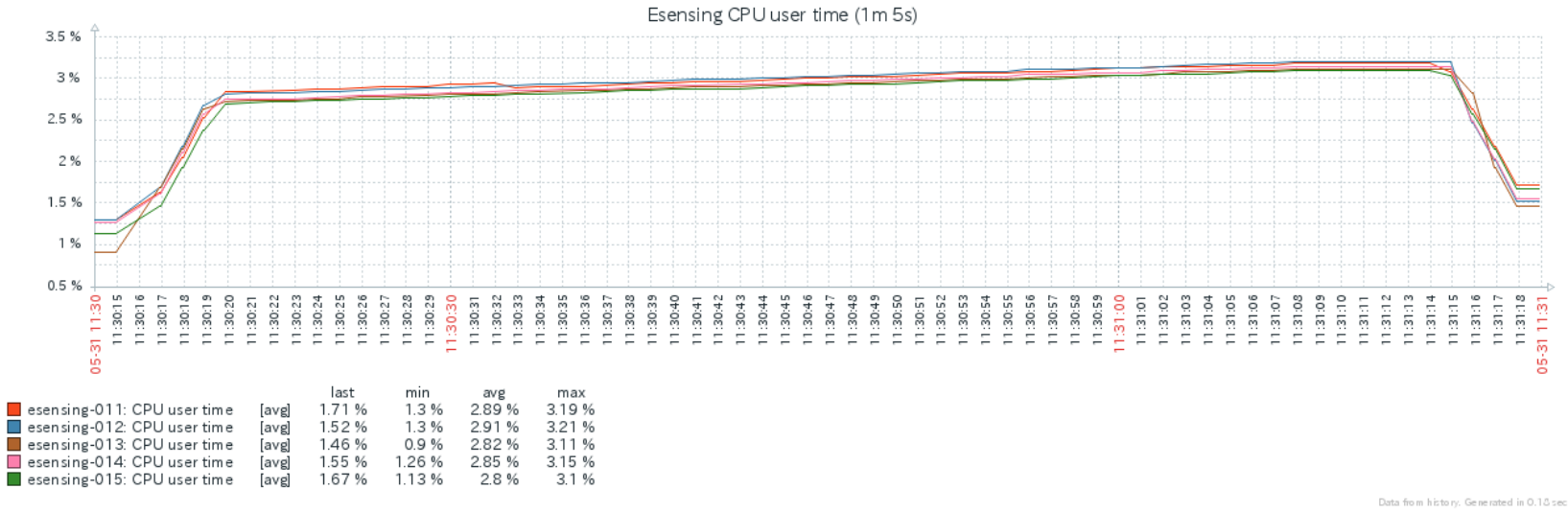

Figura 10: Uso de CPU Global durante a consulta Horizontal do Teste 1.

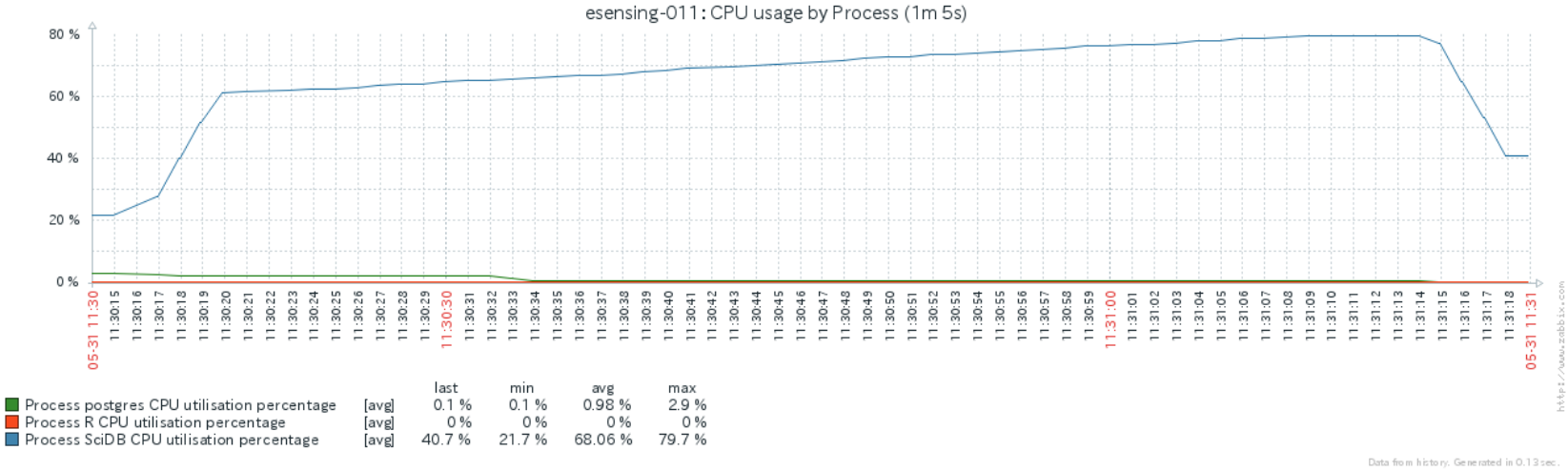

Figura 11: Uso de CPU por processo durante a consulta Horizontal do Teste 1.

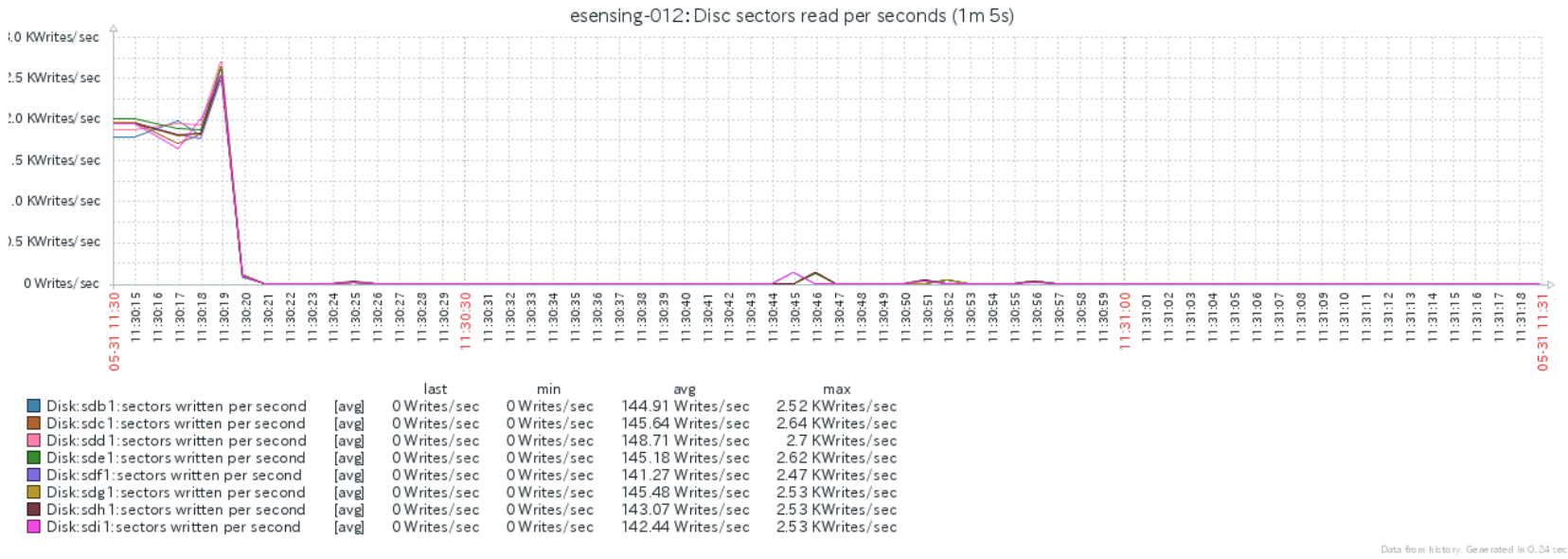

Figura 12: Quantidade de setores lidos em cada disco durante a consulta Horizontal do Teste 1.

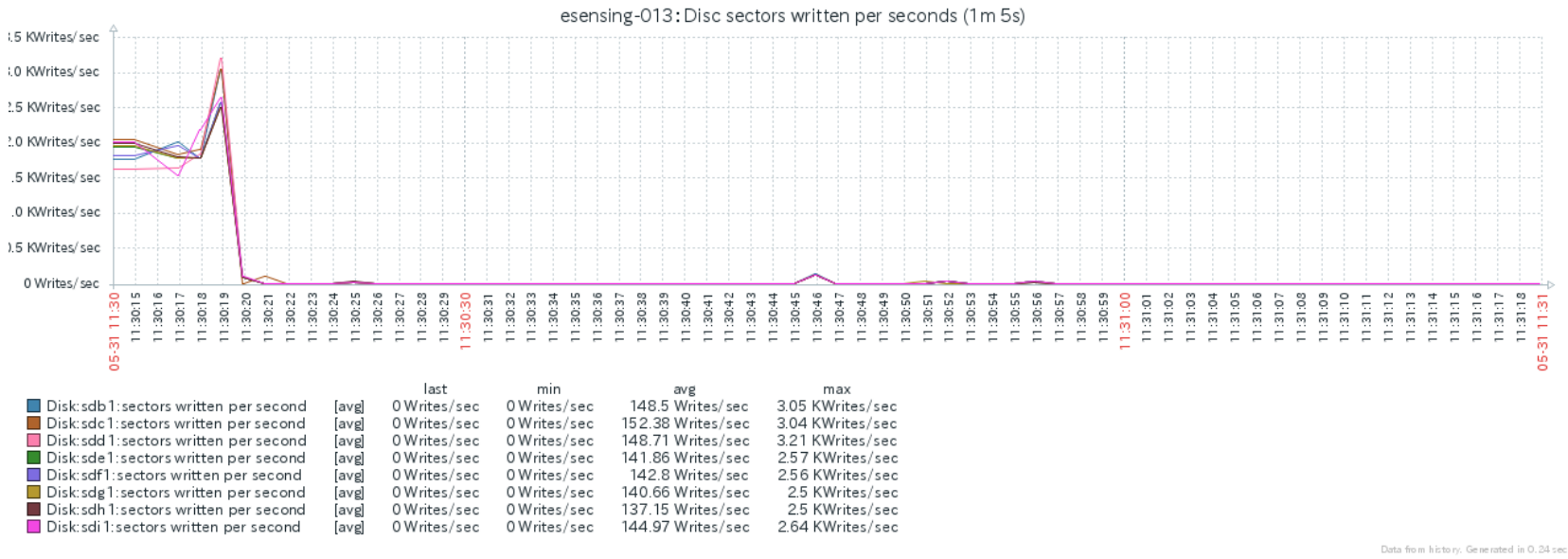

Figura 13: Quantidade de setores escritos em cada disco durante a consulta Horizontal do Teste 1.

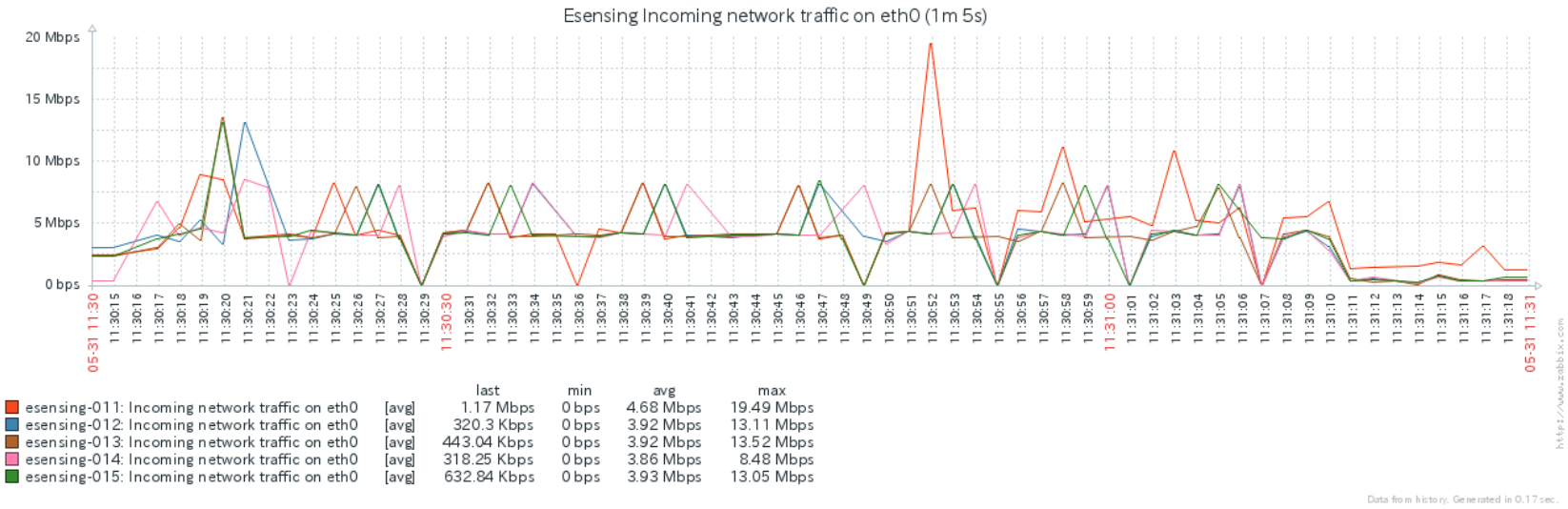

Figura 14: Volume de dados recebidos por cada servidor durante a consulta Horizontal do Teste 1.

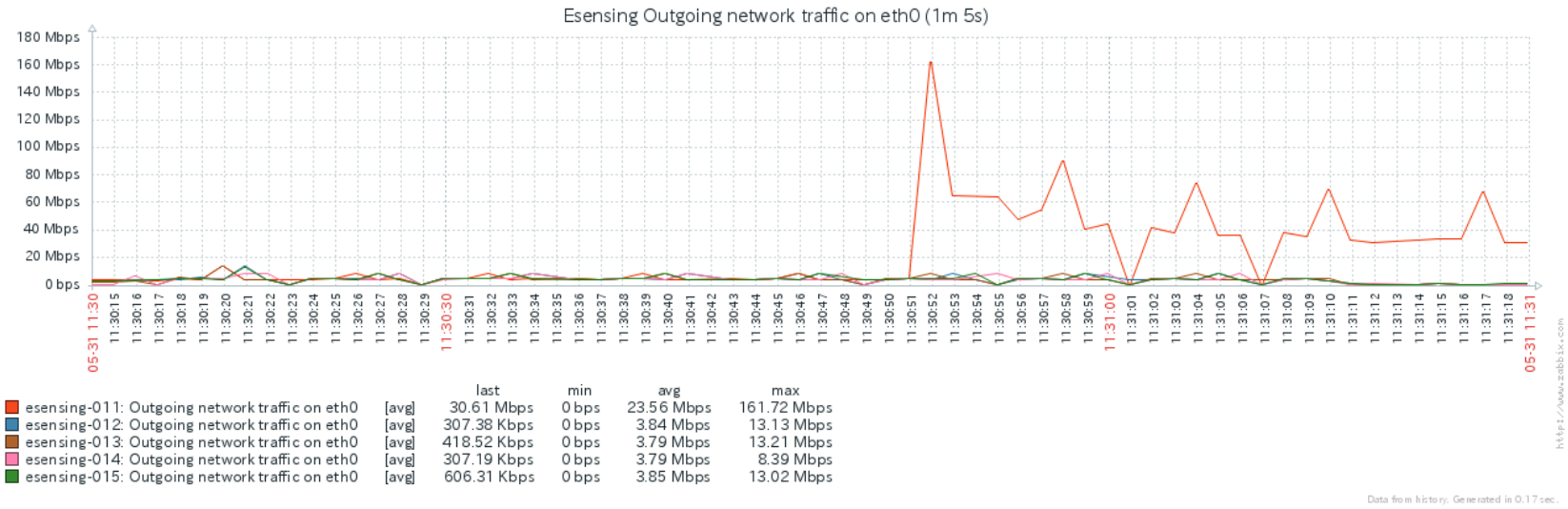

Figura 15: Volume de dados enviados por cada servidor durante a consulta Horizontal do Teste 1.

B Gráficos de métricas coletadas durante a execução da consulta Horizontal do Teste 2

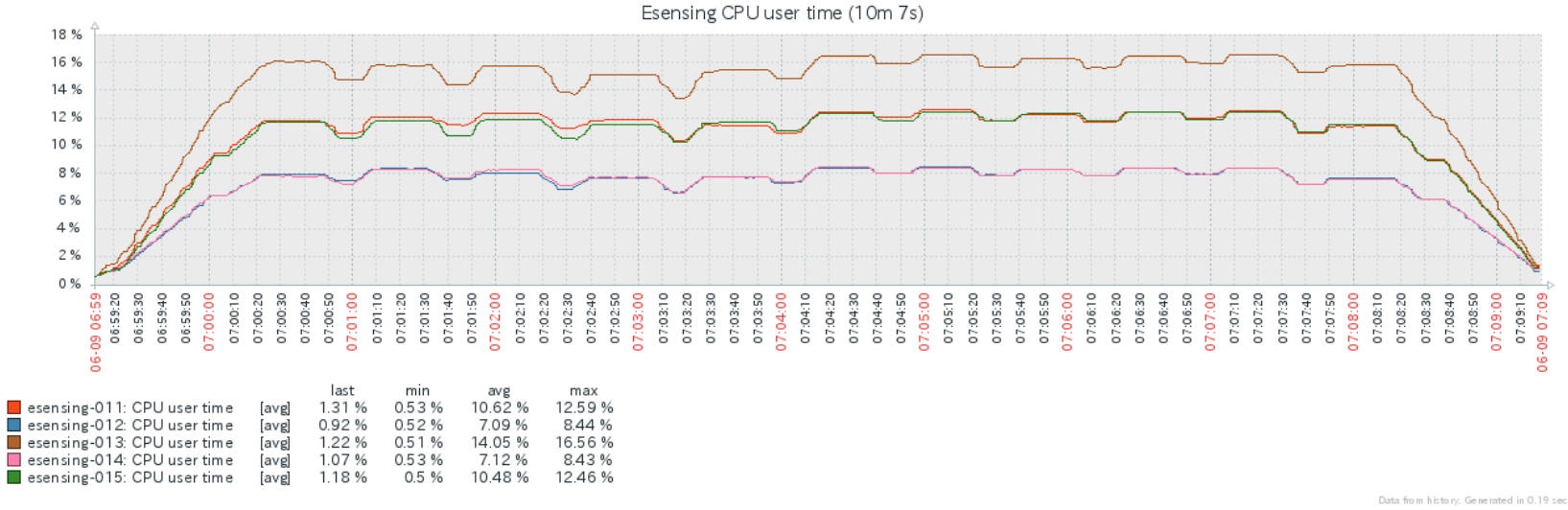

Figura 16: Uso de CPU Global durante a consulta Horizontal do Teste 2.

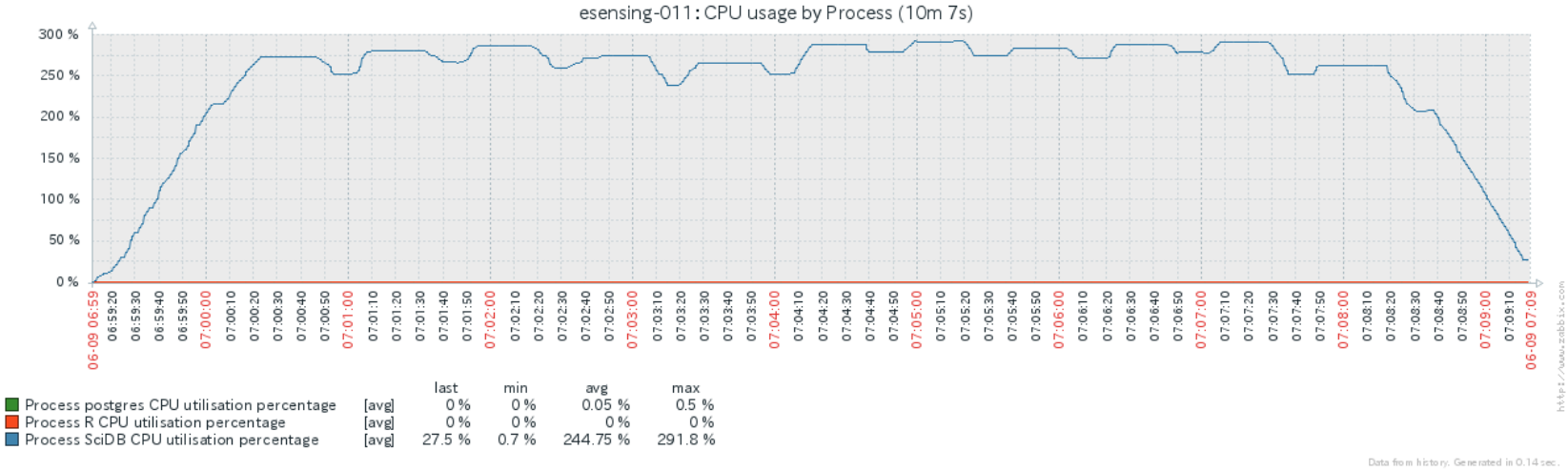

Figura 17: Uso de CPU por processo durante a consulta Horizontal do Teste 2.

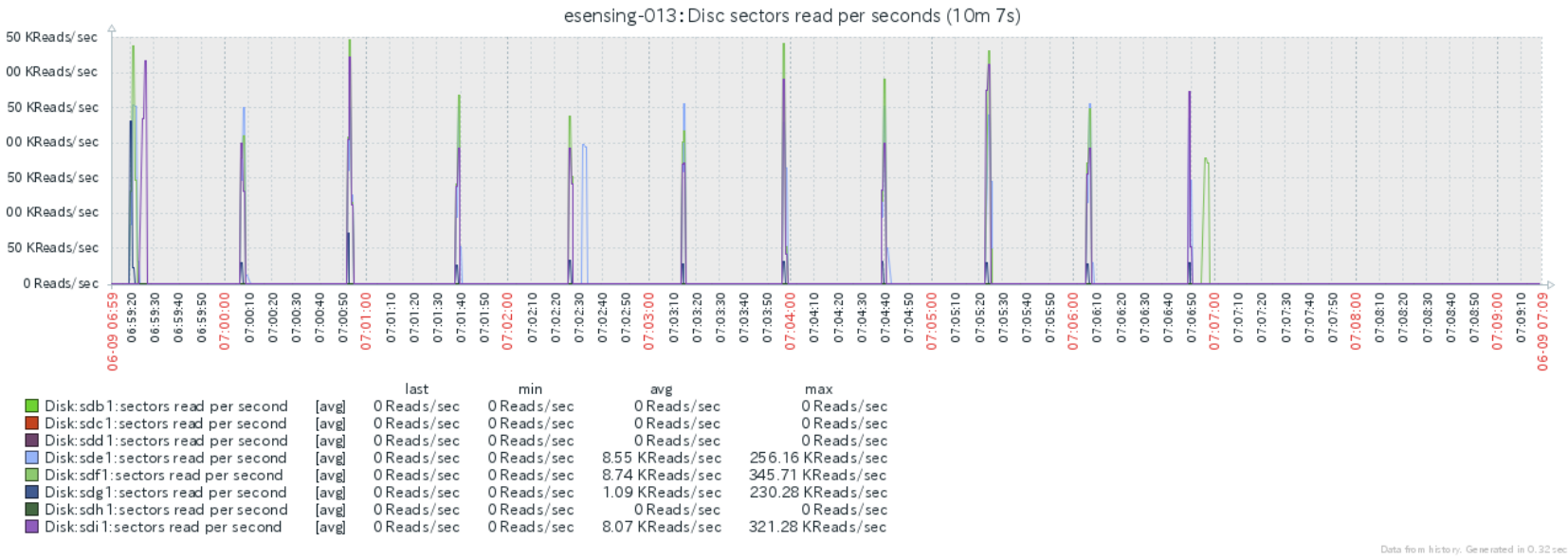

Figura 18: Quantidade de setores lidos em cada disco durante a consulta Horizontal do Teste 2.

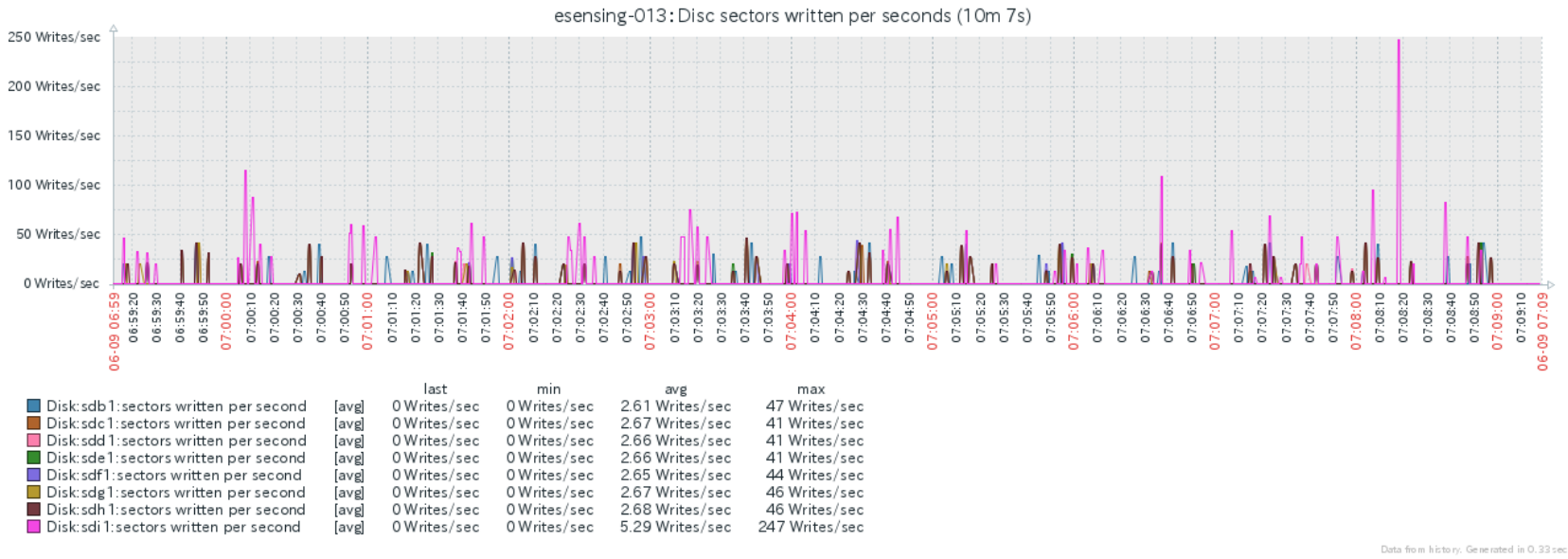

Figura 19: Quantidade de setores escritos em cada disco durante a consulta Horizontal do Teste 2.

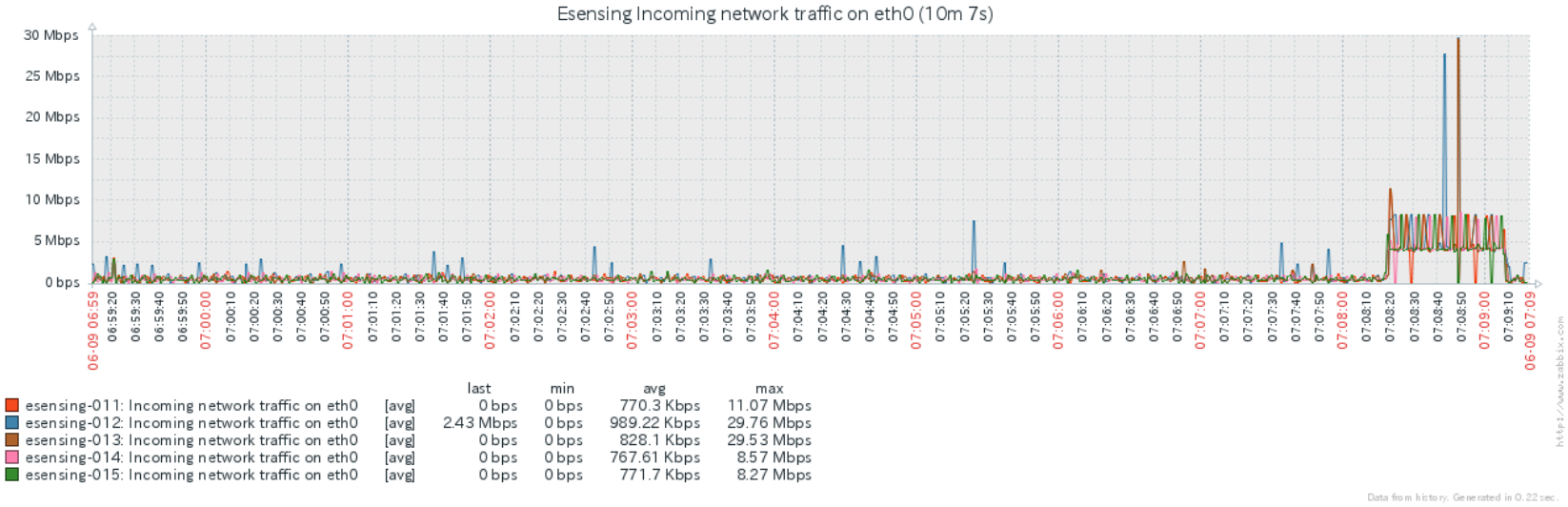

Figura 20: Volume de dados recebidos por cada servidor durante a consulta Horizontal do Teste 2.

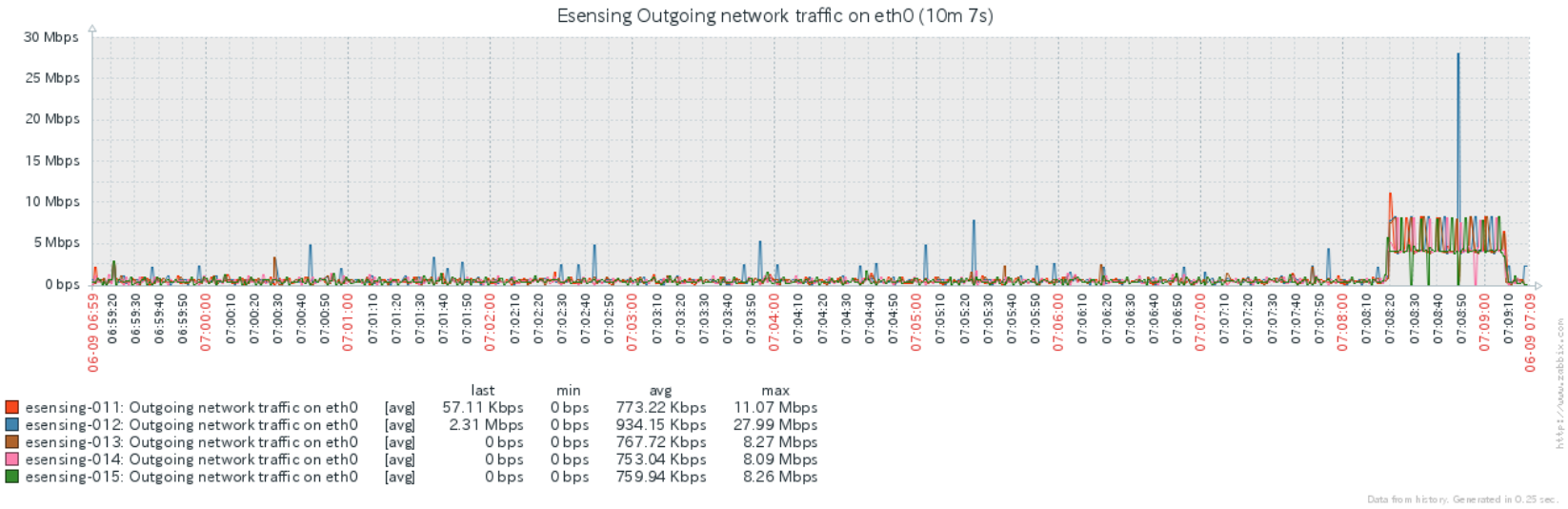

Figura 21: Volume de dados enviados por cada servidor durante a consulta Horizontal do Teste 2.

C Gráficos de métricas coletadas durante a execução da consulta Windows AVG 3x3 do Teste 1

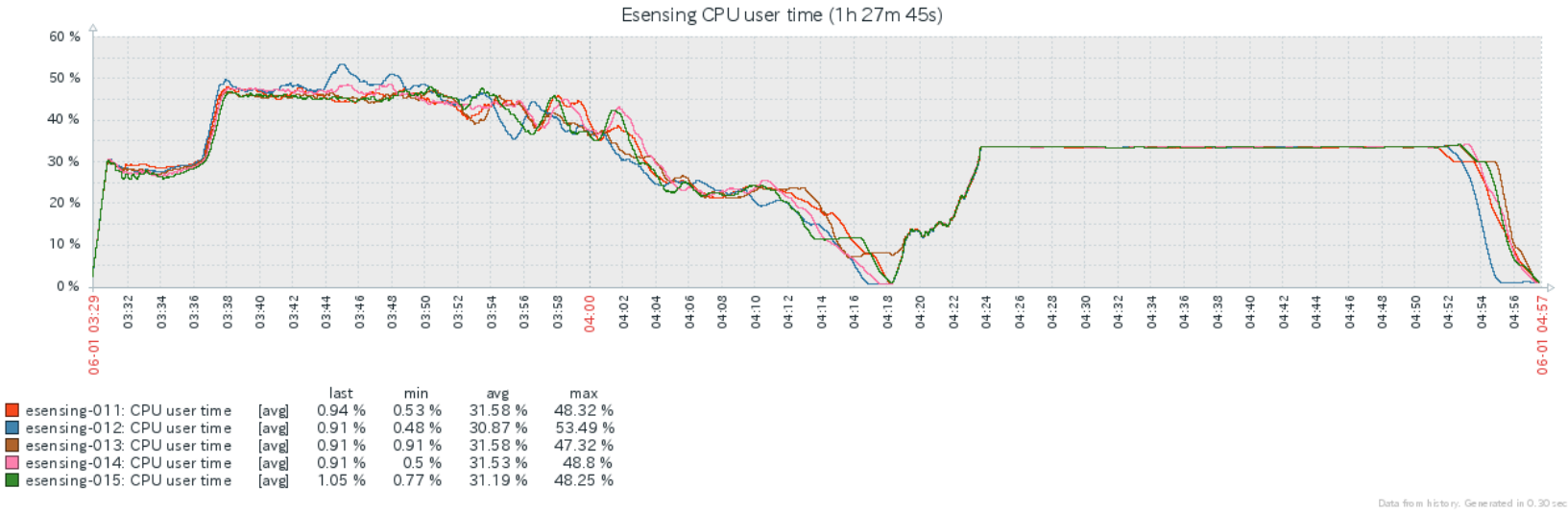

Figura 22: Uso de CPU Global durante a consulta Windows AVG 3x3 do Teste 1.

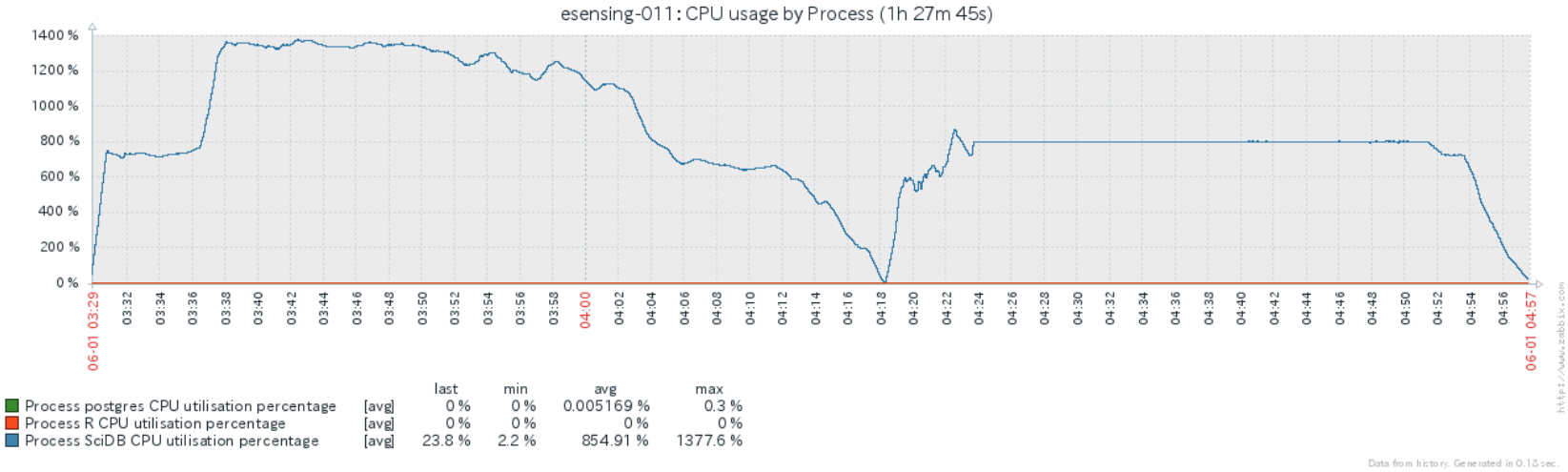

Figura 23: Uso de CPU por processo durante a consulta Windows AVG 3x3 do Teste 1.

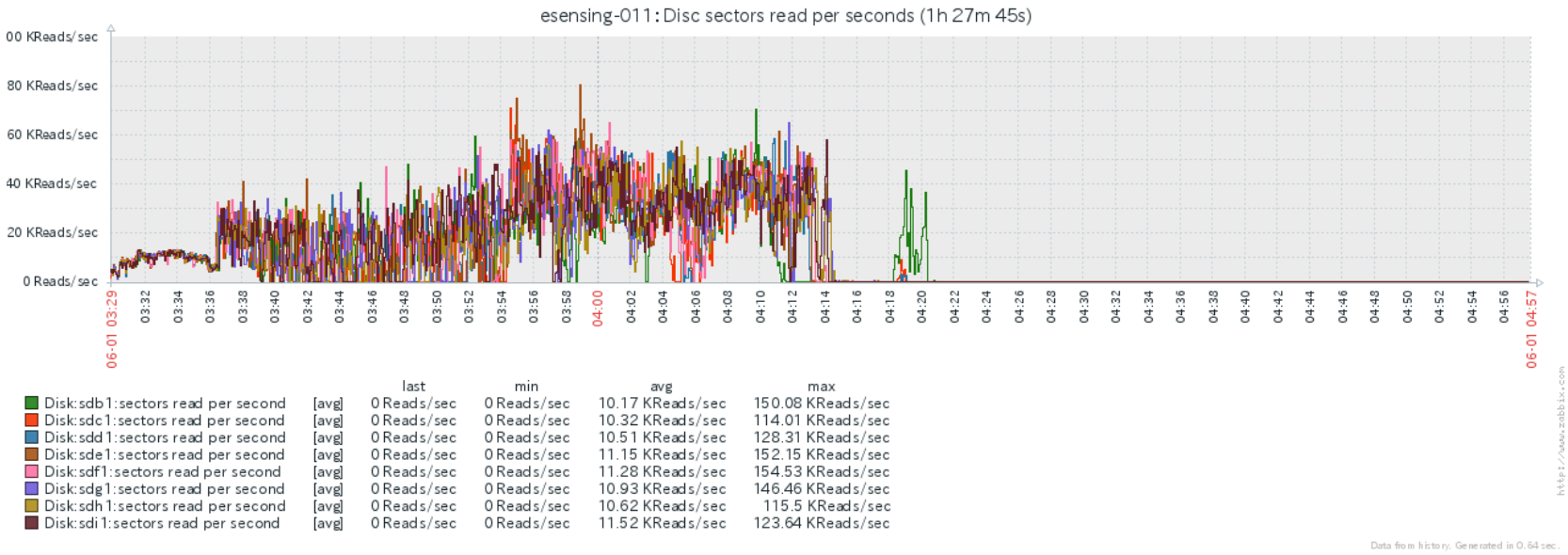

Figura 24: Quantidade de setores lidos em cada disco durante a consulta Windows AVG 3x3 do Teste 1.

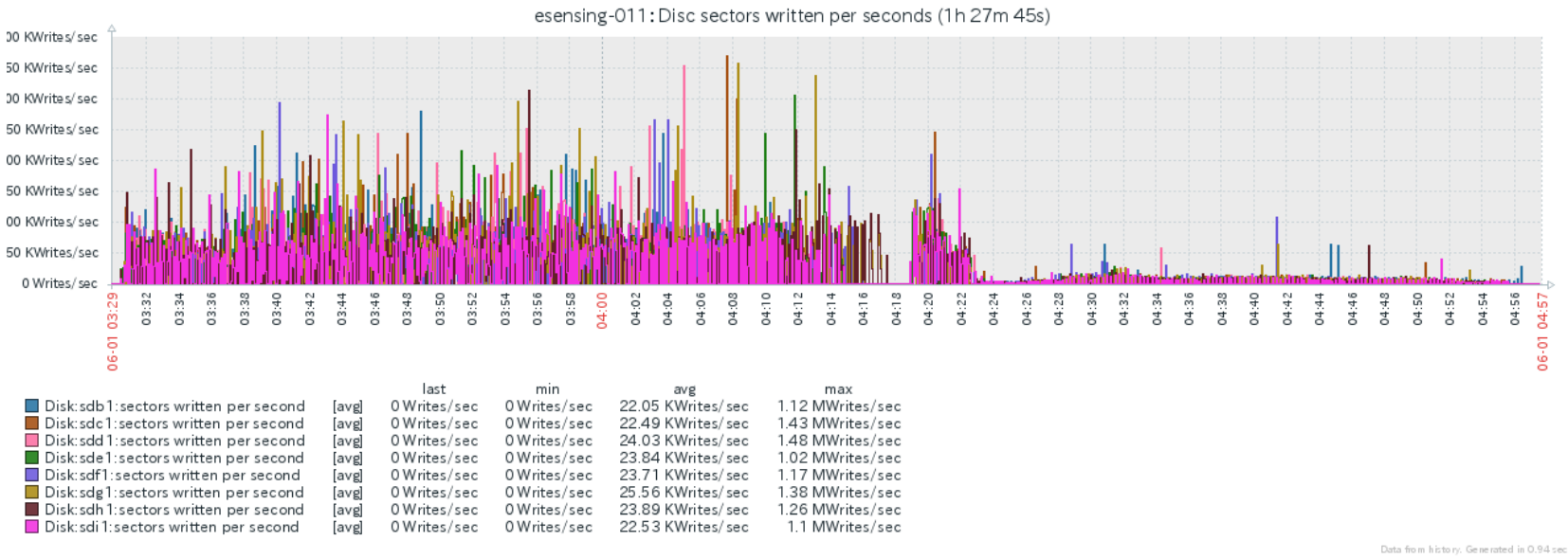

Figura 25: Quantidade de setores escritos em cada disco durante a consulta Windows AVG 3x3 do Teste 1.

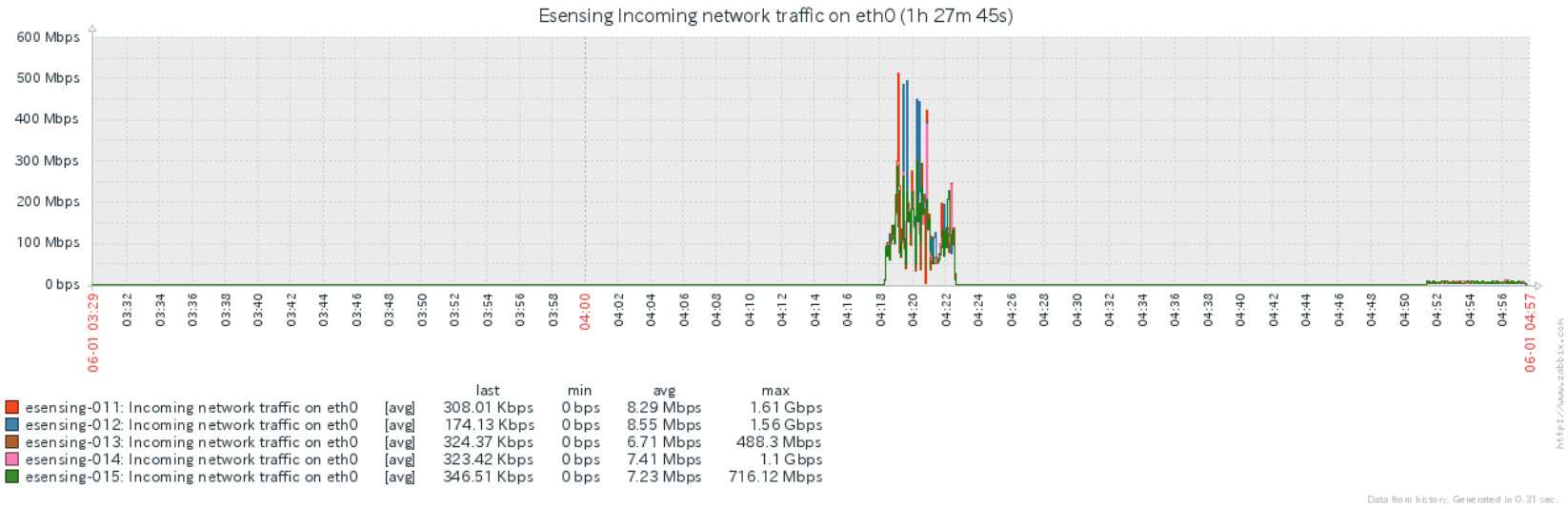

Figura 26: Volume de dados recebidos por cada servidor durante a consulta Windows AVG 3x3 do Teste 1.

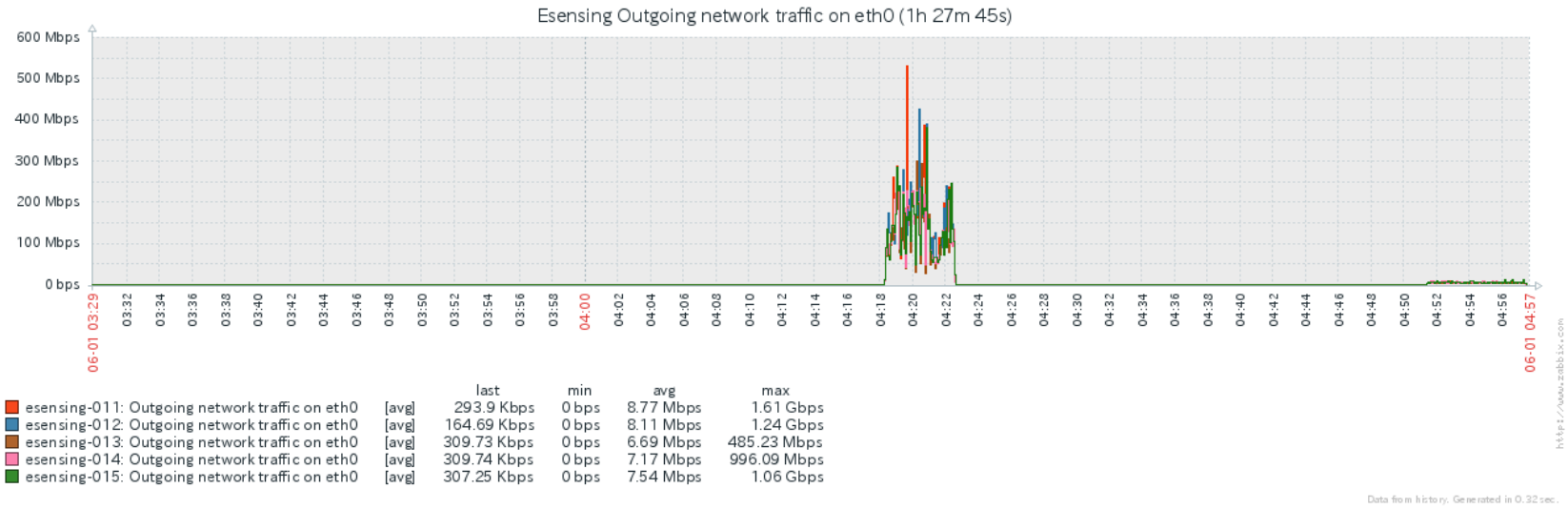

Figura 27: Volume de dados enviados por cada servidor durante a consulta Windows AVG 3x3 do Teste 1.

D Gráficos de métricas coletadas durante a execução da consulta Windows AVG 3x3 do Teste 2

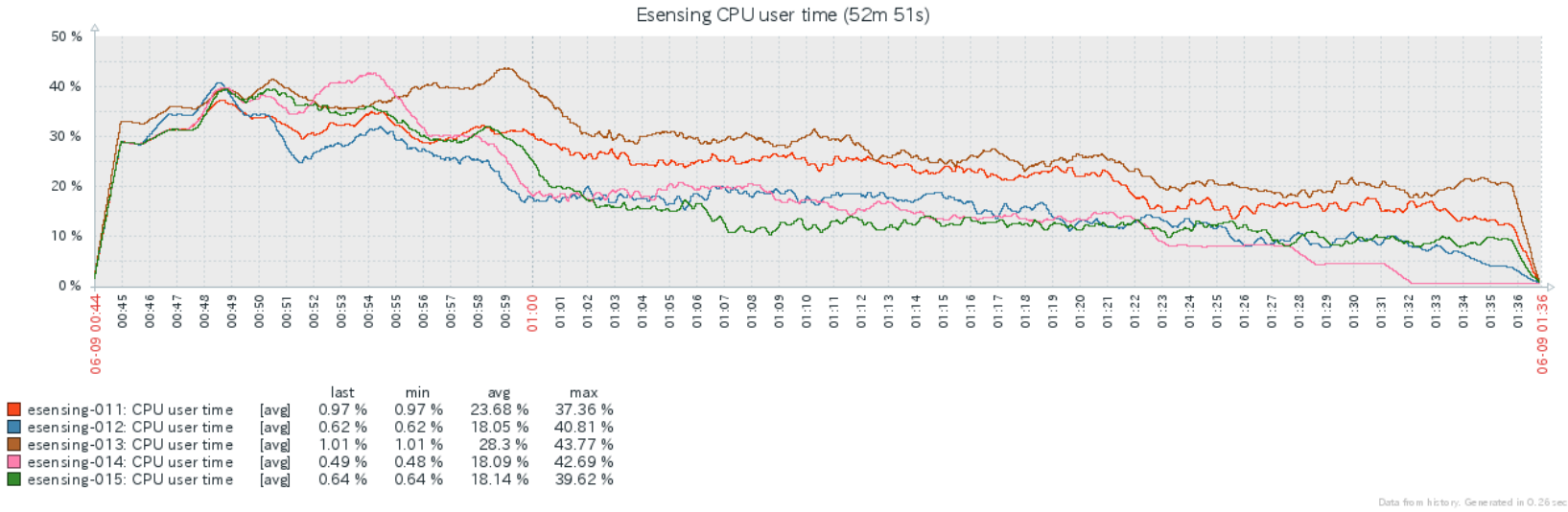

Figura 28: Uso de CPU Global durante a consulta Windows AVG 3x3 do Teste 2.

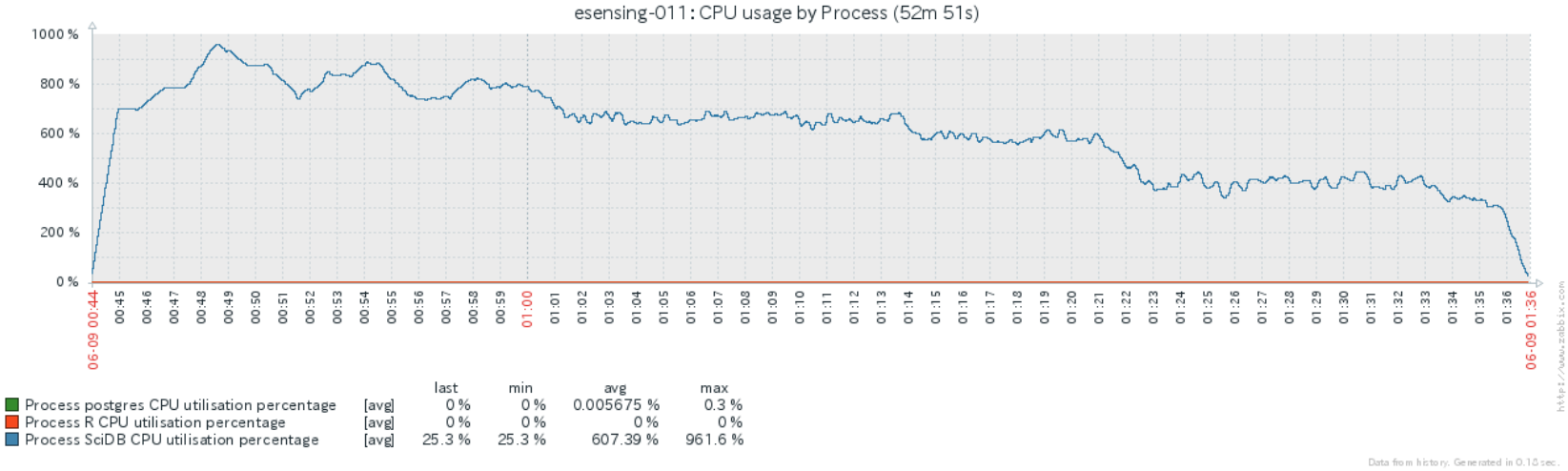

Figura 29: Uso de CPU por processo durante a consulta Windows AVG 3x3 do Teste 2.

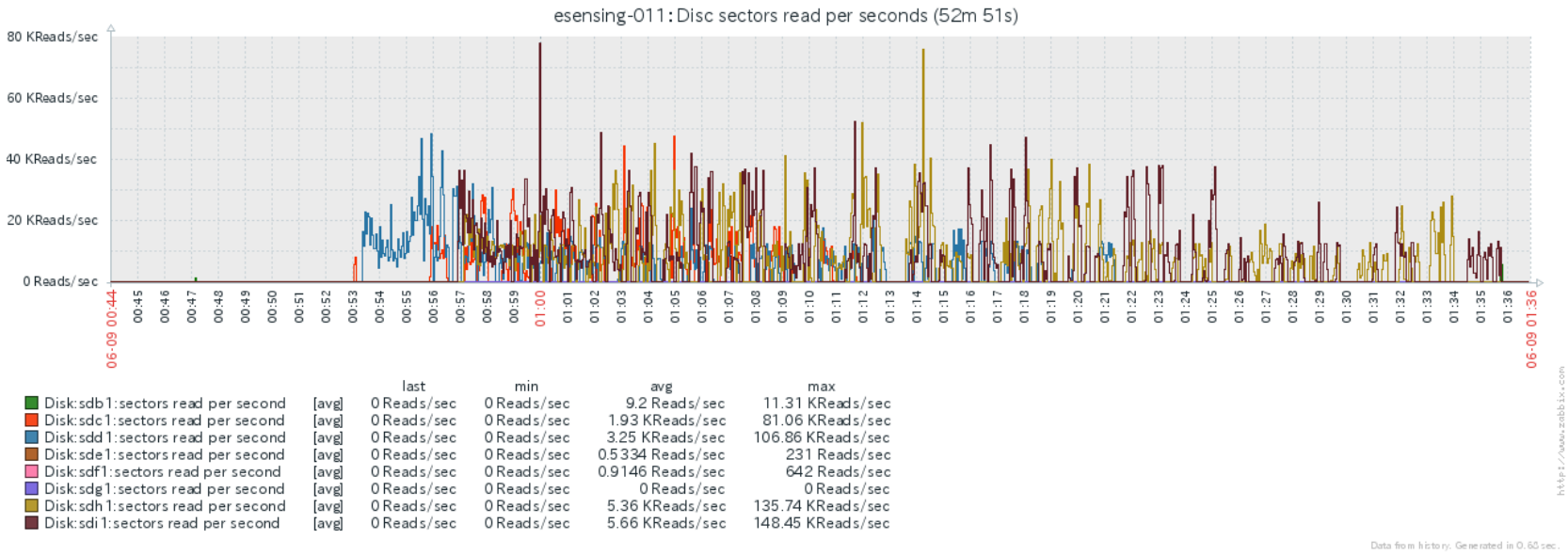

Figura 30: Quantidade de setores lidos em cada disco durante a consulta Windows AVG 3x3 do Teste 2.

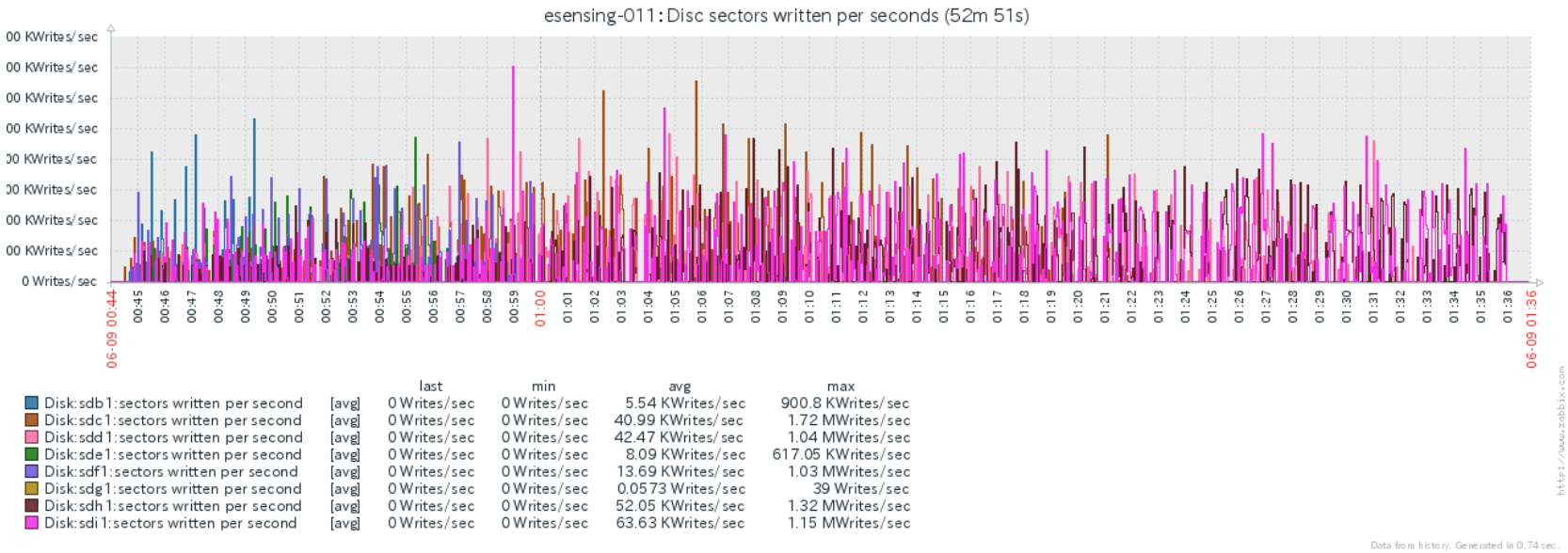

Figura 31: Quantidade de setores escritos em cada disco durante a consulta Windows AVG 3x3 do Teste 2.

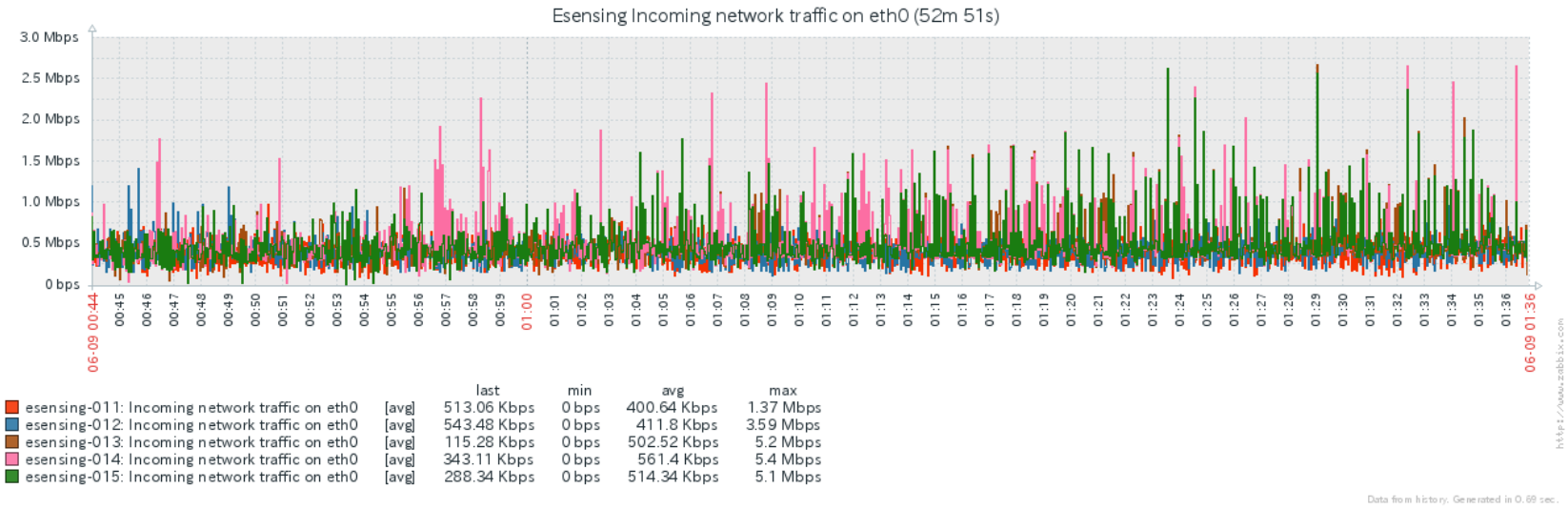

Figura 32: Volume de dados recebidos por cada servidor durante a consulta Windows AVG 3x3 do Teste 2.

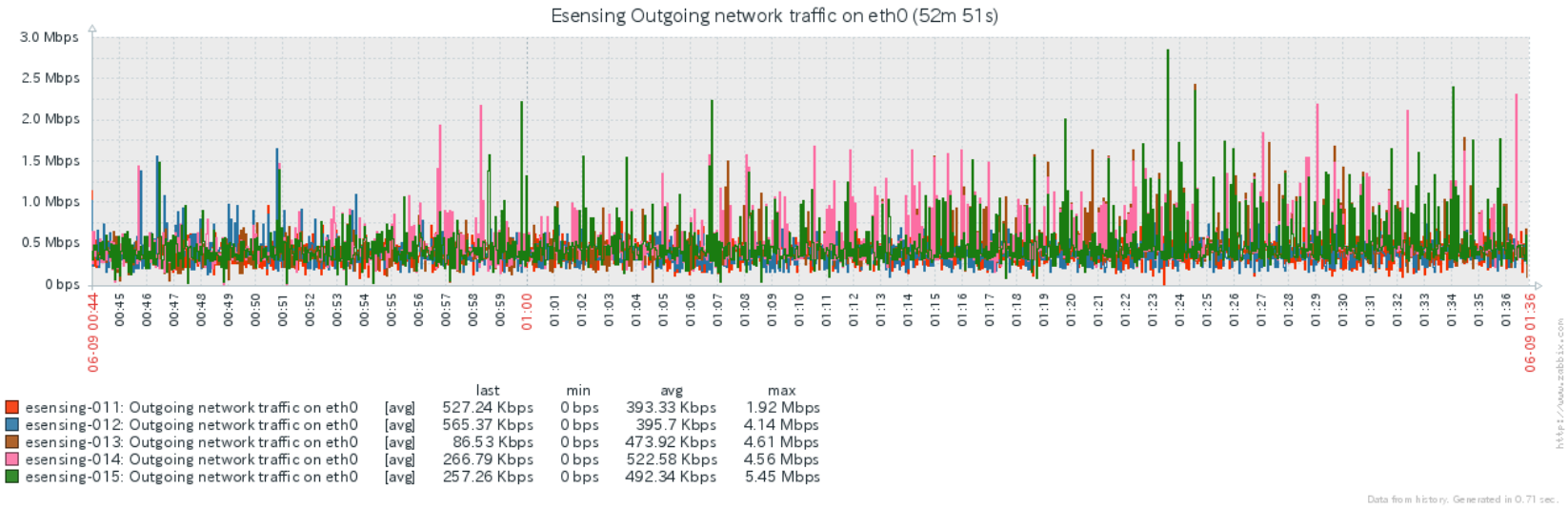

Figura 33: Volume de dados enviados por cada servidor durante a consulta Windows AVG 3x3 do Teste 2.