# Monitoramento de infraestrutura de rede com Raspberry Pi e Zabbix

**Marcos Vinicios Nogueira de Souza, Marco Antonio Alves Pereira**

Faculdade de Tecnologia de Ribeirão Preto (FATEC)

Ribeirão Preto, SP - Brasil

mnogueiraa@live.com, marco.pereira@fatecrp.edu.br

**Resumo.** O sistema de gerenciamento da infraestrutura de rede deve ser utilizado como estratégia para rastrear vulnerabilidades e auxiliar na identificação rápida dos problemas, apontando as medidas que devem ser tomadas. Desta forma, é vital realizar a gestão da infraestrutura de rede para que a empresa tenha uma alta capacidade de ação, sendo possível realizar avaliações em tempo real de todos os equipamentos e descobrir a origem do problema para solucioná-lo. Este trabalho propõe uma solução de baixo custo para o monitoramento de redes de pequeno porte.

***Abstract.*** *The network infrastructure management system should be used as a strategy to track vulnerabilities and assist in the quick identification of problems, pointing out the measures that must be taken. Thus, it is vital to manage the network infrastructure so that the company has a high capacity for action, making it possible to carry out real-time assessments of all equipment and discover the source of the problem to solve it. This work proposes a low-cost solution for monitoring small networks.*

## 1. Introdução

Redes de computadores ou *network* caracterizam-se por dois ou mais computadores interligados por qualquer meio, capazes de trocar informação entre si e/ou compartilhar recursos de *hardware*. As redes modernas são compostas de diversos equipamentos, sistemas operacionais, recursos de segurança e recursos de monitoramento, manter todos esses elementos funcionando corretamente é uma tarefa complexa. (MOTA, 2013).

A complexidade e a utilização em larga escala das redes de computadores dificultam o monitoramento e o controle, tornando comum a ocorrência de anomalias. Anomalias de rede são definidas como situações em que os níveis de tráfego apresentam um desvio do seu comportamento normal. Elas podem surgir de diferentes situações como falhas de *hardware*, defeitos em *softwares*, configurações erradas e ataques de negação de serviço. (ZARPELÃO, 2010)

O monitoramento de TI (Tecnologia da Informação) permite a identificação de pontos críticos e gargalos, solucionando os pontos críticos antes que gerem indisponibilidade. Com isso, é possível agir proativamente, assegurando manutenções e atualização de prevenção ao invés de soluções emergenciais. (MICROCITY, 2020).

O artigo demostra através de um caso de uso, uma solução de baixo custo para implementar o monitoramento em redes de pequeno porte utilizando a placa computacional Raspberry Pi juntamente com o software Zabbix.

A estrutura do trabalho se encontra da seguinte forma: seção 2 demostra os objetivos, seção 3 será apresentado a revisão da literatura, seção 4 aborda os materiais e métodos utilizados, seção 5 e 6 tratarão dos resultados e conclusões, seção 7 são listadas as referências.

## 2. Objetivo

Implementar o monitoramento em um ambiente de rede em produção utilizando o software Zabbix sobre a placa computacional Raspberry Pi 3.

### 2.1. Objetivos específicos

i. Instalar o sistema operacional e os pacotes necessários no Raspberry Pi;
ii. Desenvolver um script para envio dos gráficos gerados pelo Zabbix através do aplicativo Telegram.
iii. Preparar o ambiente a ser monitorado;
iv. Realizar testes de funcionamento.

## 3. Revisão da literatura

### 3.1. Raspberry Pi

Raspberry Pi é um microcomputador de baixo custo, criado pela ONG de mesmo nome na Universidade de Cambridge (UK) em 2006 com objetivo principal de promover o ensino da computação no ensino de base. O projeto surgiu originalmente como uma ferramenta para o aprendizado de linguagem de programação, especialmente em países de terceiro mundo, oferecendo um computador barato o suficiente para que cada estudante recebesse o seu junto com outros materiais e pudesse praticar em casa.

O Raspberry Pi é baseado em um *SoC* (*System on Chip)* Broadcom BCM2835, que combina um processador ARM11 operando a 700 MHz com uma GPU *Videocore* I*V* operando a 250 MHz. Seus componentes podem ser observados na figura 1.

O armazenamento é feito através de um *slot* para cartões de memória micro SD na parte inferior. Embora possua 4 portas USB, o raspberry pi é limitado em relação ao quanto de energia pode fornecer para dispositivos plugados na porta USB, já que a alimentação da própria placa é feita através de uma porta micro USB. Os conectores são destinados a dispositivos como teclados, *mouses*, *pen drives* e outros dispositivos de baixo consumo. A saída de vídeo primária é uma interface HDMI, que suporta resoluções de até 1080p, a placa possui também um conector de áudio de 3.5mm. O Raspberry Pi é compatível com várias distribuições Linux, incluindo Debian, Arch Linux e Fedora. (MORIMOTO, 2012)

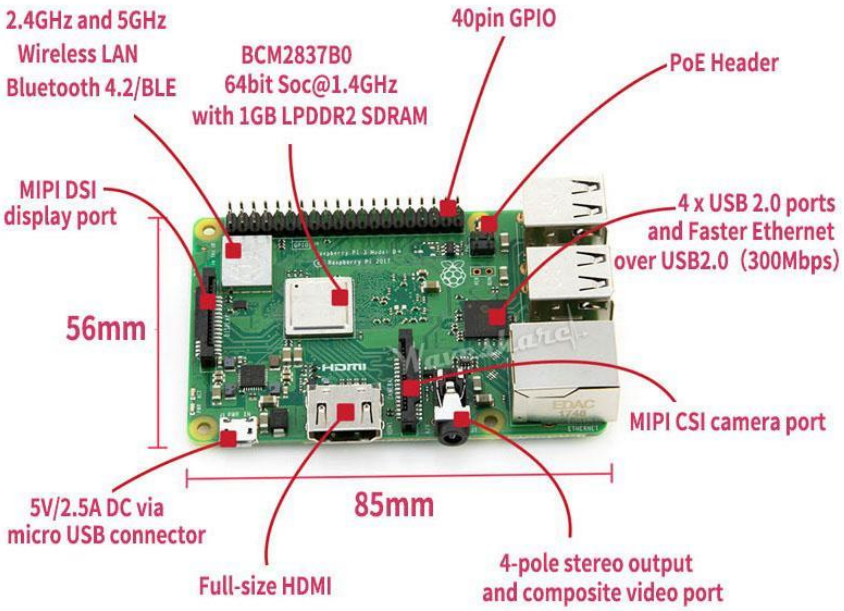


**Figura 1: Componentes do Raspberry Pi versão 3 modelo B+**
**Fonte: (RASPBERRYPI.ORG)**

## 3.2. Zabbix

Criado em 2001 pelo administrador de redes lituano Alexei Vladishev, Zabbix é uma ferramenta de monitoramento *Open Source* e multiplataforma, livre de custos de licenciamento, pois sua licença é a GPLv2 (*GNU General Public License*).

(LIMA, 2014)

Zabbix possui a capacidade de monitorar milhares de itens em apenas um servidor, além de ser possível ter um monitoramento distribuído. Dessa forma, podemos ter um servidor central de monitoramento e vários outros servidores subordinados a ele enviando métricas para o servidor central. Também é possível separar os servidores *web*, servidor de banco de dados e servidor de monitoramento para aumentar a flexibilidade e ganhar desempenho. A ferramenta possui um sistema de relatórios e gráficos bastante intuitivos, toda a configuração de monitoramento é feita através de uma interface *web* rica em detalhes, na qual podem ser criados ações ou alertas com base nas métricas recebidas.

O sistema é tão flexível, que permite obter dados através de *scripts* customizados para alerta, ação, itens e comandos remotos, tornando possível o monitoramento de itens não nativos dos agentes.

Principais características do Zabbix são (LIMA, 2014):

i. Servidores que rodam em sistemas *Unix-like*;
ii. Agentes nativos para *Unix-like* e versões Microsoft Windows;
iii. Administração e monitoramento via interface *web*;
iv. Autodescobrimento de servidores e dispositivos de redes;
v. Escalabilidade;
vi. Flexibilidade;
vii. Monitoramento em tempo real;
viii. Sistema de notificação;

ix. Autenticação segura de usuários;
x. Visualização de relatórios, gráficos, telas e mapas;
xi. Monitoramento de acordo com nível de serviço.

Zabbix integra todas as aplicações de um sistema de gerenciamento de redes sem a necessidade de *plug-ins*, e é totalmente personalizável a qualquer tipo de ambiente. Oferece um pacote completo, com mapas de redes, gráficos e telas, além de enviar alertas por *e-mail*, SMS, ou qualquer outra ferramenta de conversas através de *API 's (Application Programming Interface)*, além de poder executar ações, como um comando remoto para recuperar um serviço sem a intervenção do administrador.

### 3.2.1. Arquitetura

A figura 2 exibe a arquitetura no modelo *three-tier*, que faz uma abordagem em três camadas, que são: *back-end*, base de dados e *front-end*. O *back-end* é responsável por fazer a coleta dos dados nos ativos de rede. A base de dados fica responsável por armazenar as informações, já a camada de *front-end* permite o acesso às informações de monitoramento e fornece informações para aplicações que utilizam *API (Application Programming Interface)*.

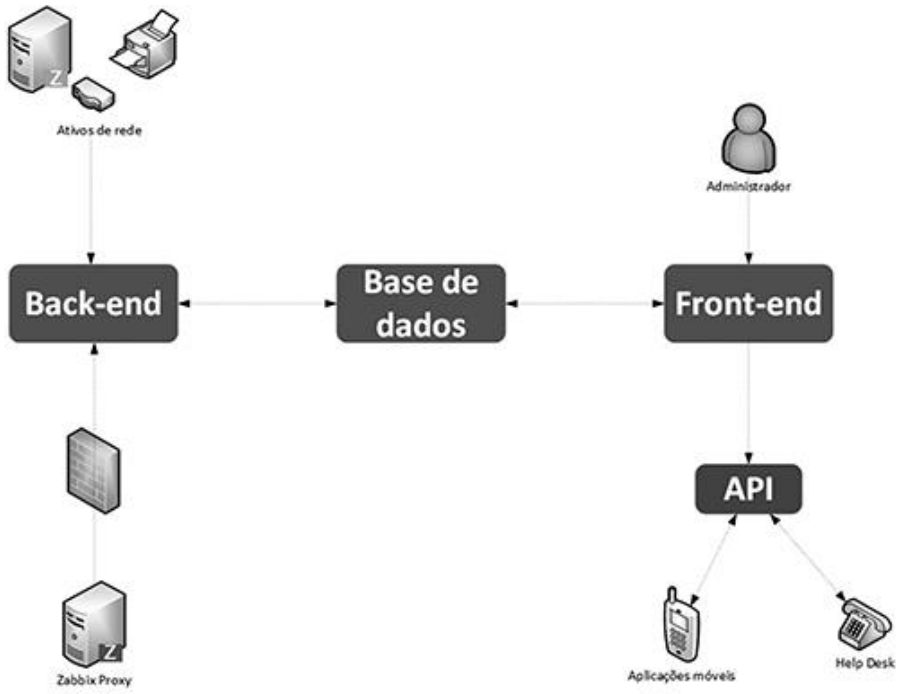


**Figura 2: Arquitetura utilizada pelo Zabbix.**
**Fonte: (LIMA, 2014)**

O *back-end* do Zabbix foi desenvolvido com a linguagem C e o *front-end* foi desenvolvido em PHP, também foi desenvolvido para suportar os principais SGBDs (Sistemas de Gestão de Base de Dados).

### 3.2.2. Requisitos de Hardware

Zabbix requer um mínimo de 128 MB de memória e 256 MB de espaço livre em disco, se tratando de um sistema que alimenta suas informações em banco de dados, precisa-se de mais memória física e armazenamento em disco. Isso é necessário quando a quantidade de *hosts* monitorados e a quantidade de parâmetros configurados são extensas, essa questão também é válida para o processador que será utilizado no servidor.

Para grandes ambientes é recomendado que se use o servidor de banco de dados separado do Zabbix Server, evitando assim a concorrência por processamento e memória entre ambos. (LIMA, 2014)

### 3.2.3. Requisitos de software

A tabela 2 exibe as versões mínimas de *softwares* suportadas para o funcionamento do Zabbix.

**Tabela 2: Versões mínimas suportadas. A escolha do banco de dados deverá ser entre os itens marcados com (*).**

**Somente um banco de dados é necessário para o funcionamento do Zabbix**

| Software | Versão Mínima |
|---|---|
| Apache | 1.3.12 |
| PHP com suporte bc, XML, session, socket, multibyte | 5.1.6 |
| PHP GD | 2.0 |
| MySQL* | 5.0 |
| Oracle* | 10G |
| IBM DB2* | 9.7 |
| PostgreSQL | 8.1 |
| SQLite* | 3.3.5 |

**Fonte: (LIMA, 2014)**

### 3.2.4. Monitoramento

O Zabbix utiliza alguns elementos para realizar o monitoramento, que são (LIMA, 2014):

i. Host: é qualquer dispositivo presente na rede com um IP ou nome DNS *(Domain Name System)*;

ii. Item: é a fonte de informação que o Zabbix utiliza para coletar dados com o objetivo de retornar uma métrica. A busca por essa informação é realizada de várias maneiras, a ser escolhida no momento do cadastro de um item;

iii. Agente passivo: a consulta é realizada pelo servidor;

iv. Agente ativo: os dados são processados pelo agente e transmitidos para o servidor;

v. Monitoramento simples: executado pelo servidor, não tem necessidade de instalação do agente;

vi. Agente SNMP (*Simple Network Management Protocol*): protocolo presente em diversos dispositivos de rede. Este foi o método escolhido para este projeto;

vii. *Trapper*: algum objeto externo pode injetar dados dentro do Zabbix *server* usando o zabbix *sender*;

viii. Arquivos de *log*: arquivos de *log* dos sistemas *Unix-like* e *Event Viewer* do Windows;

ix. JMX: monitoramento Java;

x. IPMI: monitoramento inteligente de *hardware*;

xi. Banco de dados: estatísticas de base de dados através de *query*.

### 3.2.5. Trigger

Uma vez que um *host* está sendo monitorado e o Zabbix faz a coleta de um item, temos a possibilidade de tratar esse item com um *trigger*. O *trigger* é uma regra que vai ser avaliada cada vez que a coleta de um item ocorrer, sempre que um novo valor chegar para o Zabbix, e este estiver um *trigger* associado, o Zabbix pode tomar uma decisão de acordo com a expressão lógica configurada. A partir daí podemos ter alertas com alguns níveis de severidade, no Zabbix temos seis níveis de severidade, que são (LIMA, 2014):

i. Não Classificada;

ii. Informação;

iii. Atenção;

iv. Média;

v. Alta;

vi. Desastre.

As expressões dos *triggers* possuem a seguinte sintaxe:

*{host:key.function(param)}=X*

*hots* = Dispositivo monitorado;

*key* = item que foi coletado;

*function* = função do *trigger*;

*param* = valor coletado que será comparado;

X = a regra do *trigger*.

### 3.2.6. Evento

Evento é qualquer acontecimento gerado por diferentes fontes no Zabbix, na ocorrência de eventos o Zabbix pode tomar algumas decisões, essas fontes de eventos podem ser através de (DOS REIS LIMA, 2014):

i. *Triggers* = enviar email, executar comando remoto, executar um *script*.

ii. Descoberta = buscar uma característica em *hosts*.

iii. Auto registro = adicionar ou remover registro de *host* automaticamente.

### 3.2.7. Template

Template é um conjunto padrão de elementos que podem ser aplicados em vários *hosts* que serão gerenciados utilizando o mesmo esquema. Com a utilização de *templates*, tudo acontece por herança, ou seja, um *host* pode estar associado a vários *templates*, e esses mesmos *templates* podem estar associados a outros *templates*. Com isso todos os objetos, como itens, gráficos, *triggers*, serão herdados e associados ao *host* em questão.

O Zabbix possui suporte nativo ao protocolo SNMP (*Simple Network Management Protocol*) em todas as versões. Este protocolo é utilizado para gerenciar recursos de rede como: *switches* e roteadores. (LIMA, 2014)

## 4. Materiais e métodos

Para desenvolvimento deste projeto foi utilizado o raspberry pi 3 B+ com sistema operacional Debian, espaço em disco de 16GB e memória RAM de 1GB. Foi realizada a instalação padrão do Zabbix versão 4.4.10 utilizando o banco de dados PostgreSQL. Para um melhor monitoramento da infraestrutura de rede, foi desenvolvido um *script* em *python* para envios de alarmes e visão dos gráficos via aplicativo de mensagens Telegram. Foram configurados alguns *triggers* nos equipamentos para que enviasse os alarmes via *script*.

A solução foi testada em uma pequena parte de um ambiente de recepção e tratamento de sinal de TV digital, em uma empresa de TV a cabo. A análise será realizada de maneira qualitativa.

### 4.1. Equipamentos monitorados

Para a execução deste trabalho foram inseridos os seguintes equipamentos em monitoração:

i. 4 *switches* Cisco Catalyst 4500 *Layer* 3 - utilizados para agregação dos sinais de TV Digital;
ii. 2 servidores de mídia Cisco DCM D9002 - possui a função de tratamento dos canais recepcionados;
iii. 5 receptores via satélite Cisco D9824 - recepciona o sinal vindo do satélite.
iv. Toda parte de configuração do protocolo SNMP (*Simple Network Management Protocol*) e segurança implementados no projeto não serão tratados neste trabalho.

### 4.2. Topologia

A figura 3 demonstra a topologia lógica onde foi inserido o raspberry pi para realização deste projeto. A indexação das interfaces é meramente ilustrativa.

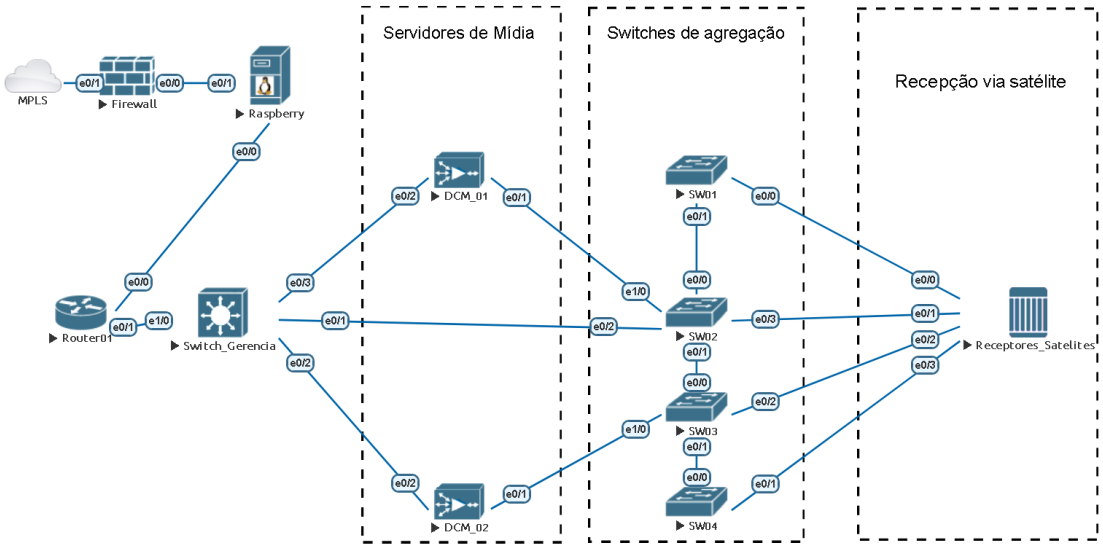


**Figura 3 - Topologia lógica do ambiente monitorado**
**Fonte: (próprio autor)**

Quando algum parâmetro atinge o valor de *trigger*(gatilho) configurado, é acionado o envio de mensagem via Telegram, o mesmo acontece quando o problema é normalizado, conforme demonstrado na figura 4.

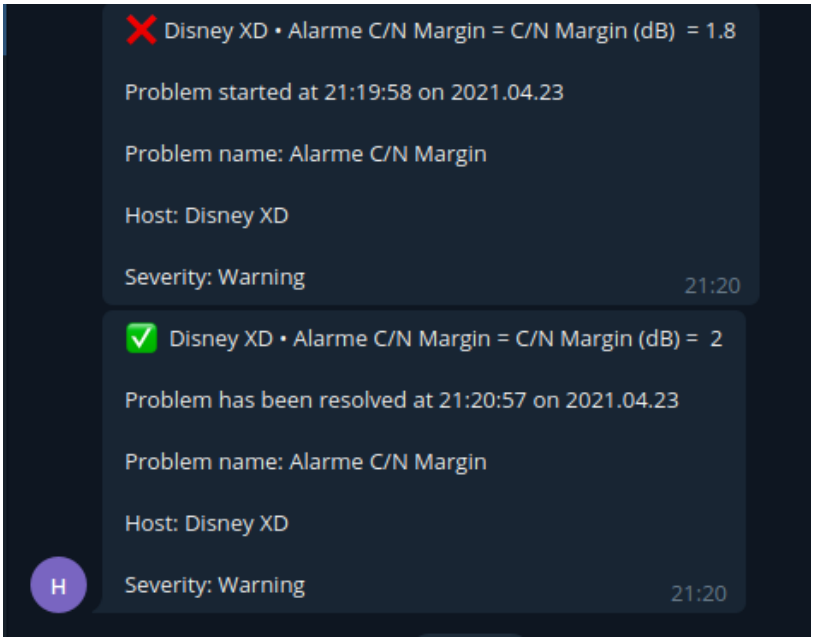


**Figura 4 - Exemplo de alarme enviado via Telegram quando acionado um trigger.**
**Fonte: (próprio autor)**

## 5. Resultados

Os testes foram realizados monitorando um total de onze equipamentos, onde o raspberry pi trabalhou no seu limite, quando inserido o décimo segundo equipamento o serviço degradou afetando todo o monitoramento. O único gargalo foi da memória RAM, que no dispositivo utilizado é de apenas 1GB. Porém, contando que uma rede de pequeno porte certamente terá menos dispositivos, a proposta do trabalho foi alcançada.

As figuras 5 a 7 demonstram o comportamento do raspberry pi com onze equipamentos monitorados. Podemos notar que o consumo de CPU e o tráfego para internet ficaram com valores adequados, já o consumo de memória cache ficou com valores alarmantes, com consumo médio de 80,7417%. O item responsável pela degradação da memória foi o *cache* de configuração. O Zabbix utiliza o *cache* de configuração para armazenar informações sobre os *hosts* e itens monitorados, por padrão o *cache* é atualizado a cada 60 segundos contendo um espaço total de 8 MB. (ZABBIX BLOG, 2011)

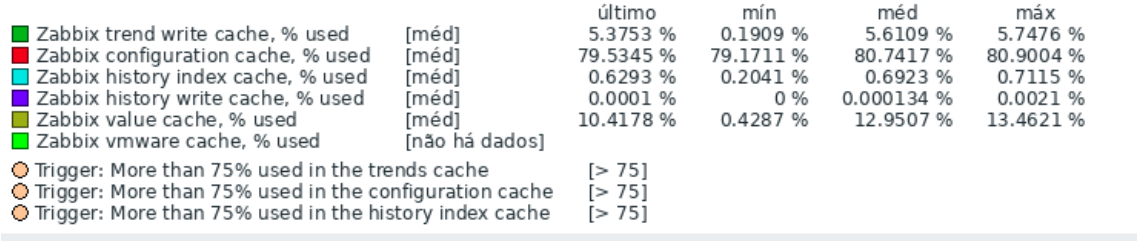


**Figura 5 - Consumo de memória cache.**
**Fonte: (próprio autor)**

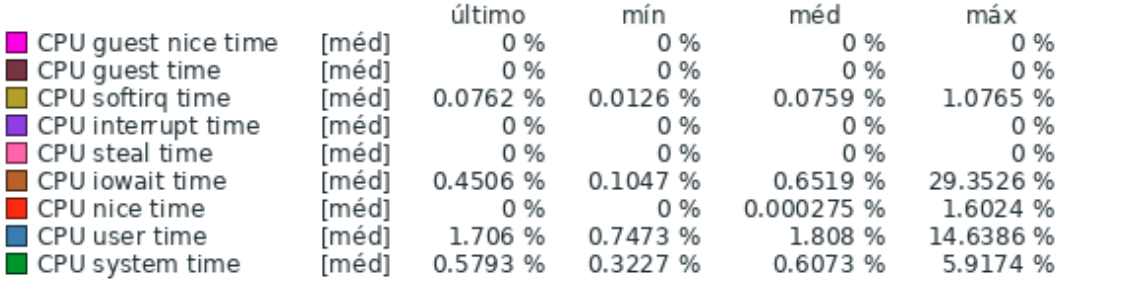


**Figura 6 - Porcentagem de uso de CPU**
**Fonte: (próprio autor)**


**Figura 7 - Tráfego de dados para internet**
**Fonte: (próprio autor)**

As figuras 8 a 10 nos mostram a degradação do serviço ao inserir o décimo segundo equipamento em monitoração. Neste cenário o raspberry pi já não conseguia coletar os dados dos dispositivos.

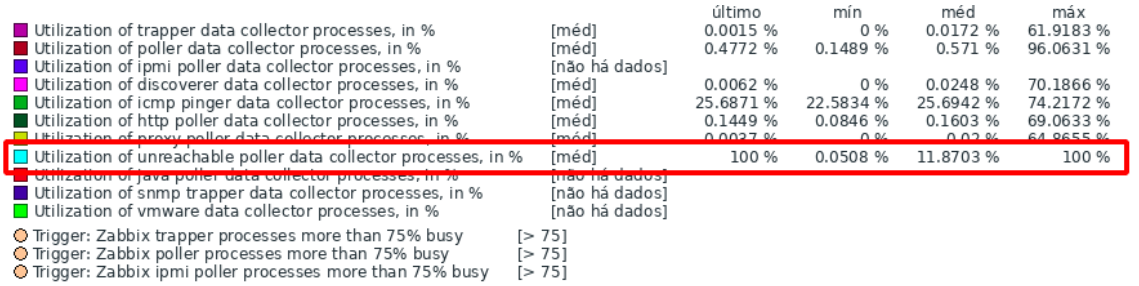


**Figura 8 - Utilização de processos do servidor. *Poller* atingiu 100% e degradou a coleta dos dados.**

**Fonte: (próprio autor)**

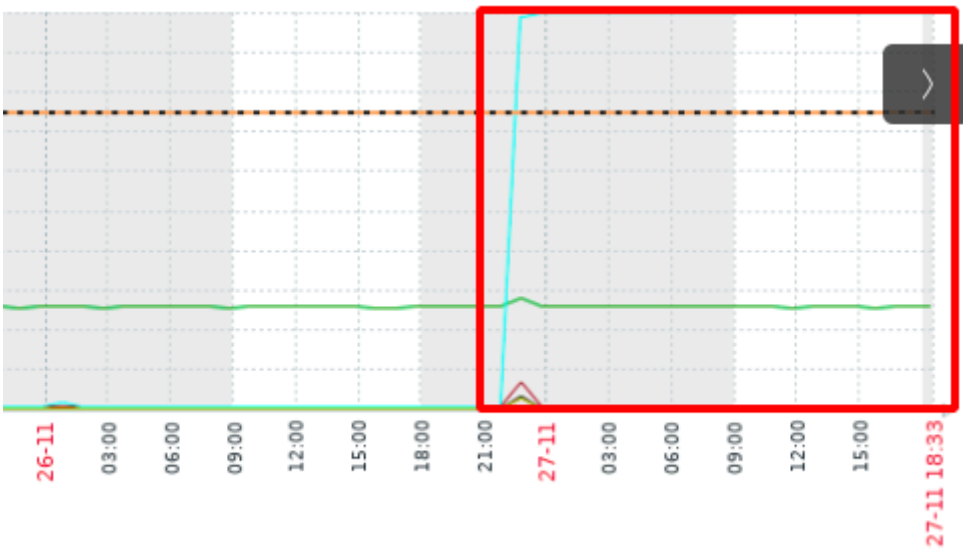


**Figura 9 - Momento exato da degradação do *poller collector*.**
**Fonte: (próprio autor)**

| | | último | mín | méd | máx |
|---|---|---|---|---|---|
| Free swap space | [méd] | 36.2 KB | 0 B | 15.98 MB | 100 MB |
| Total swap space | [méd] | 100 MB | 100 MB | 100 MB | 100 MB |

**Figura 10 - Após o consumo de toda memória RAM, o servidor utilizou todo espaço reservado para *swap*.**

**Fonte: (próprio autor)**

## 6. Conclusões

Baseado nos testes podemos afirmar que apesar do *hardware* limitado, o raspberry pi apresentou uma boa performance ao monitorar um total de onze equipamentos. Para a proposta do trabalho de monitoramento de redes de pequeno porte, o objetivo foi alcançado.

Caso a quantidade de itens a ser monitorados seja maior, é indicado realizar a instalação do banco de dados em um dispositivo separado.

## 7. Referências

BOGAZ ZARPELÃO, BRUNO. Detecção de Anomalias em Redes de Computadores**.** Tese de Doutorado FEEC-Unicamp, Campinas, 2010.

E. MORIMOTO, CARLOS. A revolução de Raspberry Pi: 2012. Disponível em <https://www.hardware.com.br/artigos/raspberrypi>. Acesso em: 20/03/2021.

ERIBERTO MOTA, JOÃO. Análise de Tráfego em Redes TCP/IP. São Paulo, Editora NOVATEC São Paulo, 2013.

RASPBERRY PI. About Raspberry Pi Foundation: 2020. Disponível em <https://www.raspberrypi.org/about>. Acesso em: 01/03/2021.

DOS REIS LIMA, JANSSEN. Monitoramento de Redes com Zabbix: 2014. Rio de Janeiro, Editora BRASPORT, 2014.

MICROCITY. Downtime: Como evitar a indisponibilidade da infraestrutura de TI: 2020. Disponível em: <https://www.microcity.com.br/downtime>. Acesso em: 30/04/2021.

ZABBIX BLOG. Reloading configuration cache: 2011. Disponível em <https://blog.zabbix.com/reloading-configuration-cache/528>. Acesso em: 04/05/2021.