

Universidade Federal de São Carlos
Departamento de Computação

TÓPICOS AVANÇADOS

Relatório Final

Alunos

Marcelo Azevedo Diniz Oliveira - 316881

Matheus Sant'ana Lima - 316938

Motivação

Com a grande disseminação da computação em nuvem, torna-se necessário o desenvolvimento de ferramentas que auxiliem no planejamento de alocação e utilização dos recursos tanto de software como de hardware, permitindo desta forma que o provisionamento dos mesmos possam ser feitos de forma eficiente.

Introdução

A ferramenta desenvolvida tem como objetivo monitorar um conjunto de servidores, preferencialmente virtualizados em um provedor de nuvem pública, mediante a captura de dados de consumo do sistema operacional. Dentre as informações relevantes temos memória, CPU, métricas de processos, swap, IO, rede, dentre outros.

Uma vez que estas informações foram capturadas elas são enviadas para um servidor central, que as recebe e insere em um banco de dados NoSQL, para posterior visualização na forma de gráfico de *consumo x tempo*, como exemplificado na Figura 1.

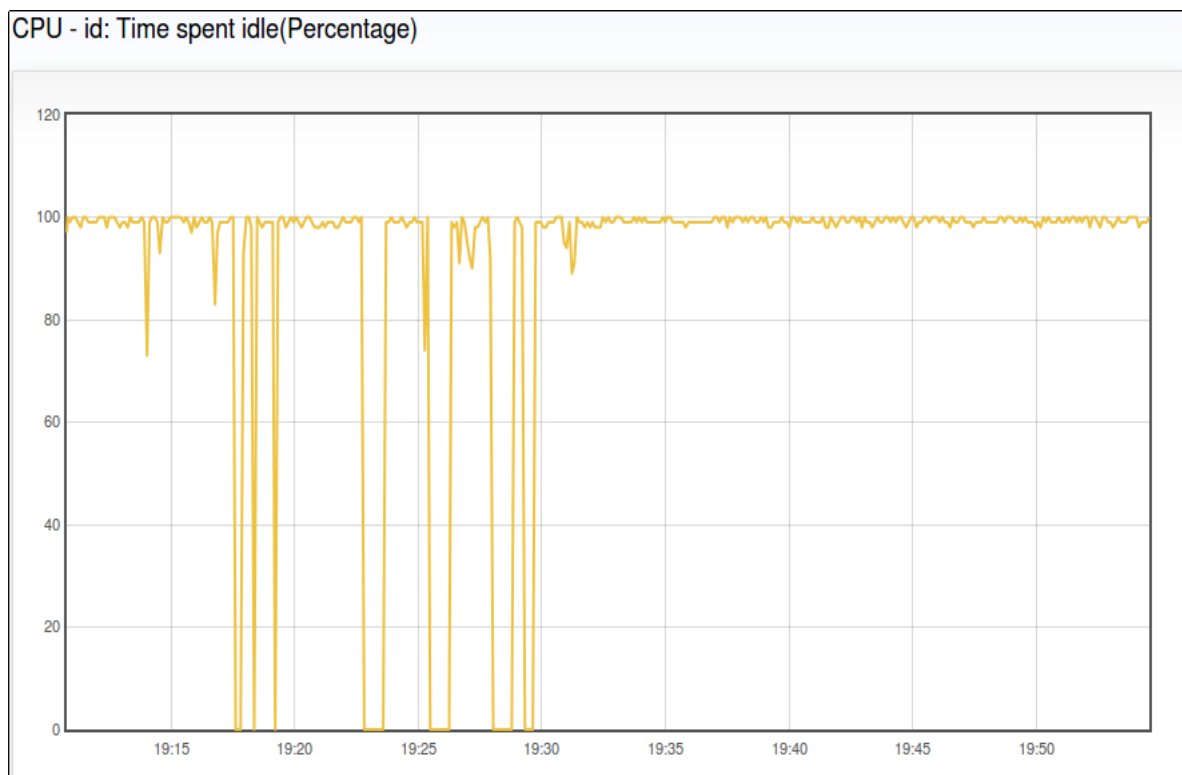


Figura 1: Exemplo de gráfico gerado pelo programa

Este tipo de ferramenta permite que administradores de rede e gerentes de infraestrutura possam melhor planejar a alocação de recursos computacionais, uma vez que disponibiliza graficamente a utilização dos mesmos no decorrer do tempo, permitindo a identificação de padrões, como por exemplo horários em que o consumo de disco e rede tendem a aumentar ou diminuir.

Alguns cenários em que esta ferramenta tem especial importância são:

- Alocação de mais recursos como memória, disco e banda em horários ou situações em que se é conhecido o aumento da demanda. Ex: Última hora do último dia para declaração do Imposto de Renda.
- Planejamento de atualização e manutenção de sistemas em momentos em que se tem um baixo consumo dos recursos, de forma a minimizar o impacto aos usuários;
- Detecção de tentativas de ataques, como por exemplo DDOS(*Distributed Denial of Service*), mediante a análise do padrão das taxas de entrada de pacotes nos servidores;

Conceitos Relacionados

Dentre os conceitos relacionados podemos destacar:

Computação em Nuvem

Termo genérico que geralmente identifica que os recursos computacionais são providos abstratamente na forma de serviços, permitindo desta forma elasticidade e fácil provisionamento. Se divide em inúmeras subáreas, dentre elas, as mais importantes são:

SaaS(*Software as a Service*)

Também conhecido como software “*sob-demanda*”. Neste modelo os usuários acessam os sistemas pela rede, mediante o uso de tecnologias apropriadas como por exemplo navegadores web, utilizando o padrão HTML. Geralmente o foco principal deste conceito são usuários finais, que acessam sistemas de e-mail e CRM pelo navegador, por exemplo. Todos os detalhes de Infraestrutura e Plataforma são transparentes ao usuário.

Exemplos de SaaS:



Figura 2: Exemplo de SaaS

PaaS(Plataform as a Service)

Provedores de serviço disponibilizam pela rede sistema operacional, linguagem de programação, banco de dados e *web services*. Isso permite um grande nível de abstração aos desenvolvedores, uma vez que facilita o *deploy* rápido de novos ambientes.

Exemplos de PaaS:



Figura 3: Google App Engine, Windows Azzure, Sales force - Exmples de PaaS.

IaaS(Infraestructure as a Service)

Neste modelo são disponibilizadas geralmente máquinas virtuais com sistemas operacionais pré-instalados, ficando a cargo da equipe de infraestrutura contrante implementar toda a pilha de serviços necessários. Apesar de mais complexo, permite maior flexibilidade e controle sobre os sistemas em execução. Além de máquinas virtuais, também são disponibilizados armazenamento, firewalls, load balancers, endereços IP, Virtual Local Area Networks (VLANs), VPNs, dentre outros.



Figura 4: Amazon EC2 and Rackspace - Examples of IaaS

Na figura 5 é possível ver um sumário dos conceitos principais de computação em nuvem:

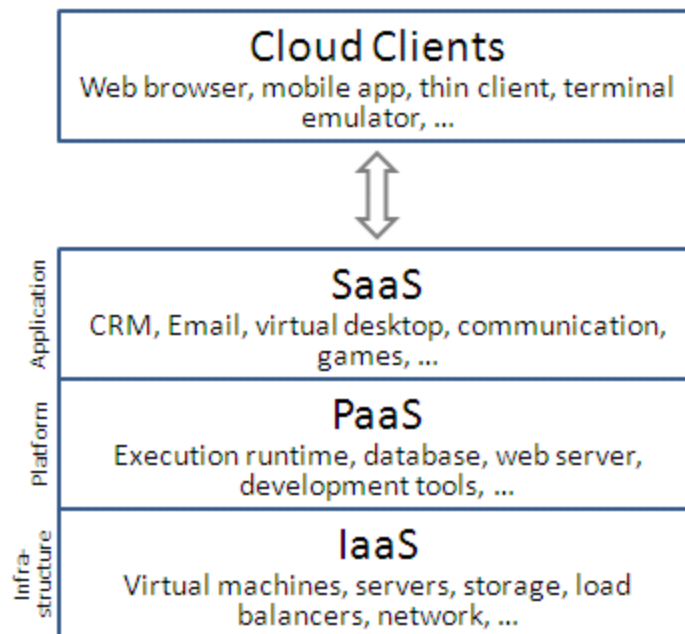


Figura 5: Diversos tipos de computação em nuvem, de acordo com nível de abstração crescente de software e hardware(IaaS até SaaS);

Nuvens Públicas, Privadas e Híbridas:

Pública:

Neste modelo, uma entidade, como por exemplo uma empresa, uma universidade, um órgão governamental, etc, contrata recursos computacionais de um provedor de serviços de nuvem. Esta por sua vez os disponibiliza pela Internet. A principal vantagem da entidade contrante é a abstração da implementação e

manutenção dos recursos à uma empresa terceira, permitindo desta forma economia de gastos. Além disso, como citado anteriormente, temos também o provisionamento rápido de novos recursos.

Privada:

Neste modelo, toda a infraestrutura de nuvem é implementado localmente no datacenter da empresa. Apesar de mais trabalhoso e geralmente mais caro que a contratação de uma nuvem pública, a nuvem privada permite total controle sobre os recursos computacionais e, em casos em que segurança é primordial como em bancos, é a única solução. Questões de privacidade dos dados são geralmente mais confiáveis neste caso.

Híbridas:

Nas nuvens híbridas temos uma composição dos modelos de nuvens públicas e privadas. Elas permitem que uma nuvem privada possa ter seus recursos ampliados a partir de uma reserva de recursos em uma nuvem pública. Essa característica possui a vantagem de manter os níveis de serviço mesmo que haja flutuações rápidas na necessidade dos recursos. A conexão entre as nuvens pública e privada pode ser usada até mesmo em tarefas periódicas que são mais facilmente implementadas nas nuvens públicas, por exemplo.

Banco de dados não-relacional (NoSQL)

Um banco de dados NoSQL fornece um mecanismo de armazenamento para modelagens que não necessitam de grande consistência entre os dados, como verificado no modelo relacional. Este tipo de banco é otimizado para recuperação e inserção de informações e serve primariamente como armazenamento. Isto permite maior flexibilidade, rapidez e escalabilidade em alguns tipos de aplicações, em que o modelo relacional não é o mais adequado.

Por estes motivos, os banco de dados NoSQL são interessantes quando se trabalha com uma grande quantidade de dados e quando a natureza dos dados não necessita de todas as restrições do modelo relacional. Desta forma este tipo de banco

somente é utilizando quando o objetivo principal é a habilidade de armazenar e recuperar grande quantidades de dados e não o relacionamento entre os elementos.

A figura 6 mostra alguns tipos de banco de dados NoSQL:



Figura 6: Exemplos de banco de dados NoSQL.

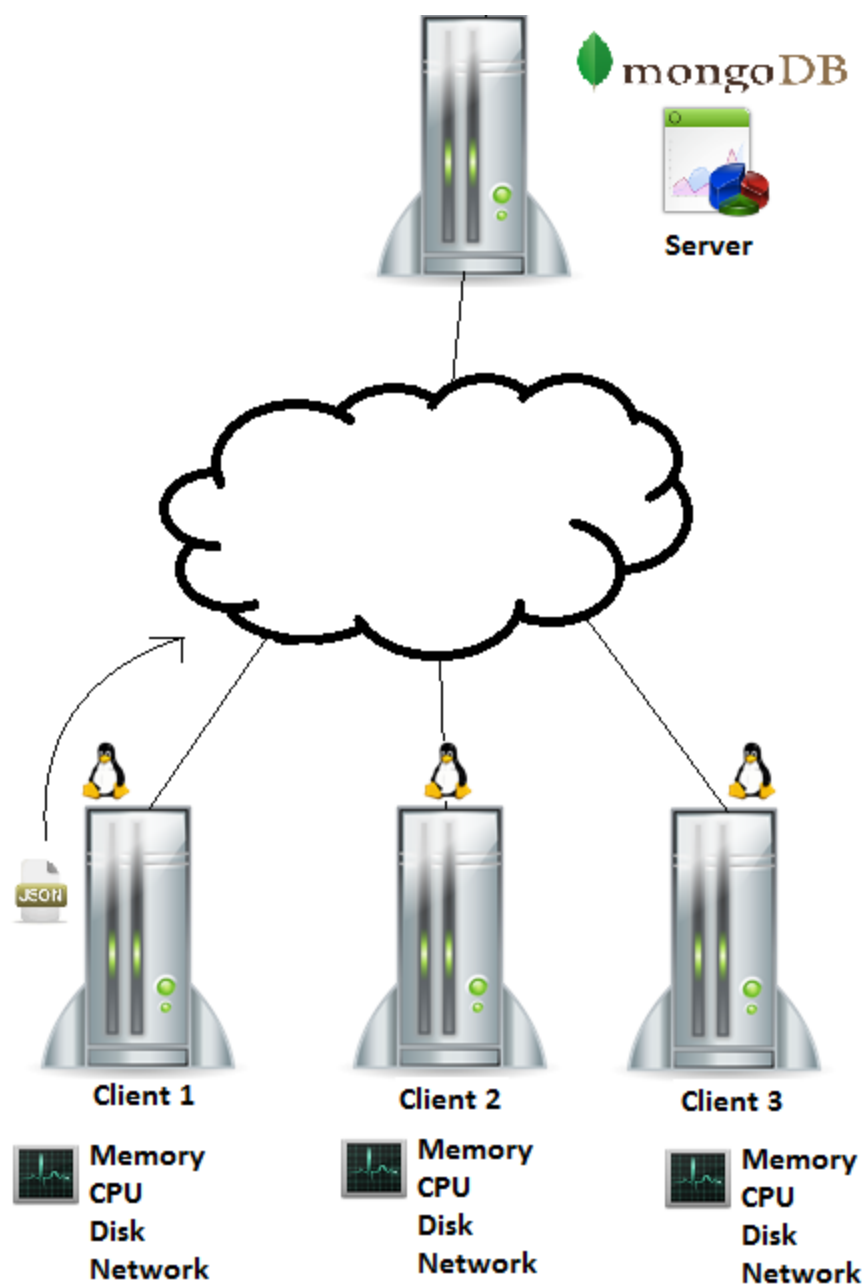
Para este projeto foi utilizado do banco de dados mongoDB, que é um banco aberto com uma vasta comunidade de usuários e desenvolvedores, suportando inúmeras linguagens de programação, dentre elas a linguagem PHP.

Metodologia e tecnologias utilizadas

Utilizando dos conceitos aprendidos em aula sobre computação em nuvem e banco de dados noSQL, o grupo implementou um programa que é responsável por monitorar um servidor virtualizado em uma nuvem pública, recebendo dados de consumo, como memória, CPU, entre outros, e, quando requisitado pelo administrador do sistema, irá gerar um grafico relacionando as taxas dos itens monitorados com um certo período de tempo determinado, para que, com base nessas informações, seja possível visualizar ou até mesmo alterar componentes desse servidor em questão. O programa tem uma interface gráfica de fácil compreensão, em que é possível gerar gráficos e comparar recursos de acordo com o tempo.

Arquitetura

A arquitetura desenvolvida é como segue:



Cliente

O Cliente foi implementado na linguagem C++. Ele é o responsável por coletar os dados de consumo do sistema operacional, no caso Linux, formatar no padrão de comunicação JSON e enviar periodicamente pela rede. A linguagem C++ foi utilizada principalmente devido ao fato que, pelo menos teoricamente, o *overhead* de outras

linguagens como Java e Python é maior, graças ao tempo gasto com a interpretação dos bytecodes. Além disso uma vez que o objetivo principal da ferramenta é monitorar servidores, a carga gerada por este *overhead* mesmo que pequena em comparação aos outros processos executados no SO, ainda é fonte de ruído considerável.

O envio dos pacotes pela internet é realizado mediante o uso da biblioteca cURL, que facilita o desenvolvimento uma vez que permite uma grande abstração da camada de rede.

cURL

libcurl é uma biblioteca de transferência de arquivos portátil, que suporta diversos sistemas operacionais como Solaris, NetBSD, FreeBSD, OpenBSD, Darwin, HP-UX, IRIX, AIX, Tru64, Linux, UnixWare, HURD, Windows, Symbian, etc mediante o uso de uma vasta gama de padrões como FTP, FTPS, Gopher, HTTP, HTTPS, SCP, SFTP, TFTP, Telnet, DICT, LDAP, LDAPS, IMAP, POP3, SMTP and RTSP. Além disso é aberta, com suporte a inúmeras linguagens de programação, como C/C++, Java, PHP e Python.

Arquitetura

Diagrama de Classe

O software cliente possui cinco classes principais, como segue:

- *Main* - Classe principal, é a responsável por instanciar as classes *vmstat* e *Instance*, responsáveis por realizar a captura e envio dos dados, assim como a leitura dos parâmetros de configuração. A classe *Main* fica em loop infinito, capturando as métricas de utilização do sistema operacional e enviando para o servidor.
- *vmstat* - Responsável por invocar a classe *InputDataInterface* para que esta realize a execução do script *vmstat_tracker.sh* para extração das métricas capturadas e posterior formatação no padrão JSON.
- *InputDataInterface* - Responsável por executar comandos no

sistema operacional e retornar os valores de saída.

- *DataOutputJson* - Armazena as strings das mensagens em formato json. É posteriormente utilizada para o envio pela rede.
- *Instance* - Classe responsável por ler o arquivo de configuração do usuário, display das informações na CLI e envio do pacote json pela rede. Esta classe se comunica com a biblioteca libcurl para o envio do mesmo.

Abaixo segue o diagrama de classe do cliente:

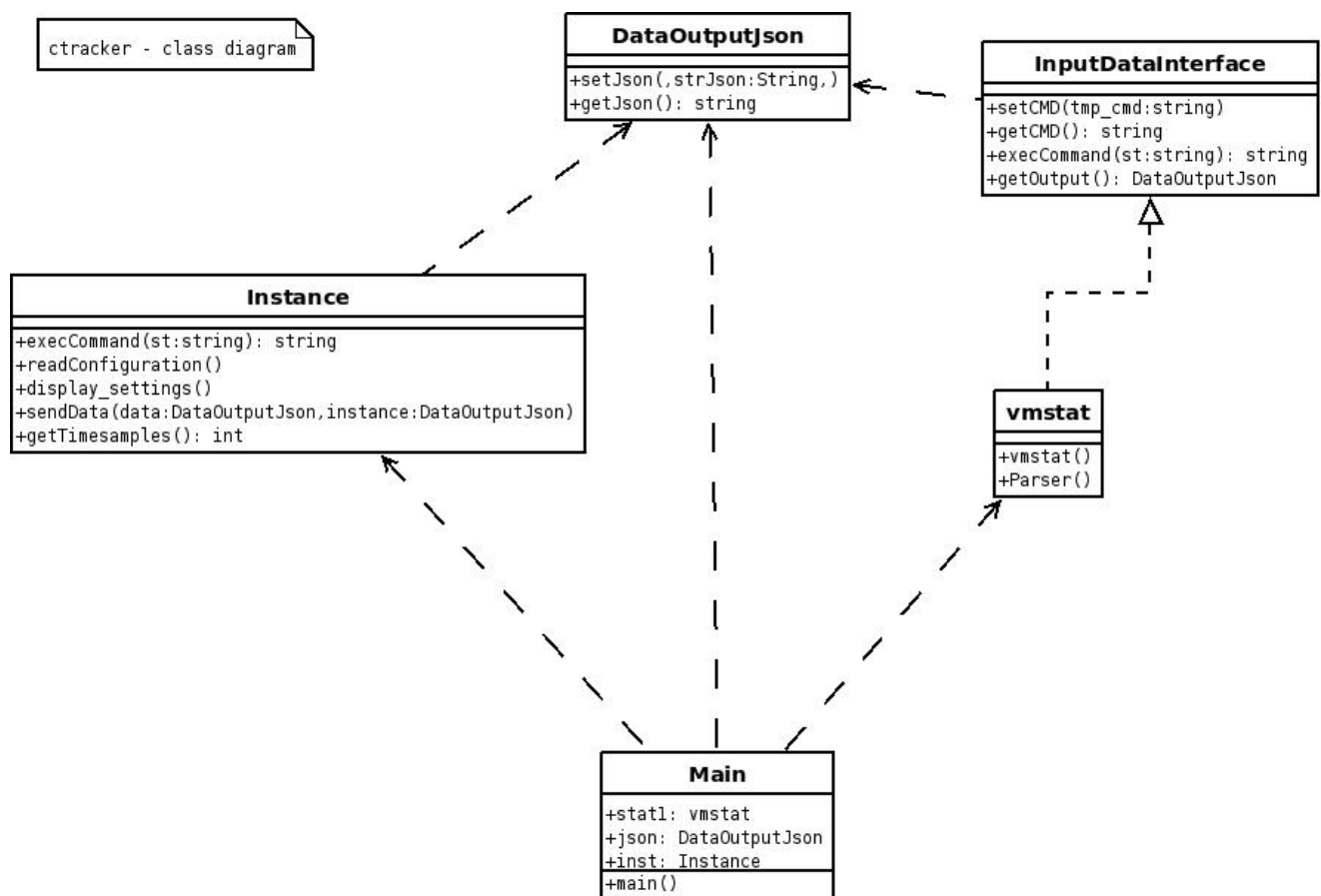
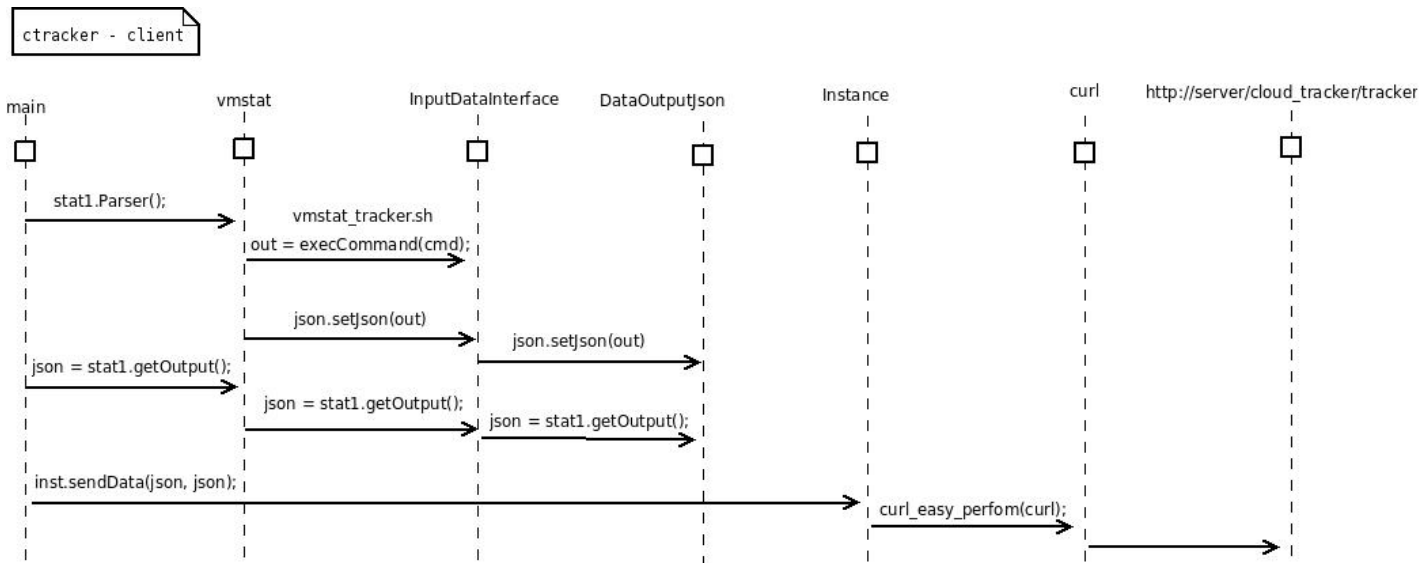


Diagrama de Fluxo

A execução da extração, formatação e envio das métricas é ilustrado pelo diagrama de fluxo abaixo. Nele é possível ver que a classe Main realiza chamada para a classe vmstat para realização da extração das informações de consumo. Por sua vez as classes InputDataInterface e DataOutputJson realizam a execução do script

vmstat_tracker.sh e armazenam a json resultante, respectivamente. A classe Instance é a ultima a ser chamada, ela contém as informações referentes a frequencia em que as amostras serão coletadas, URL do servidor e tokenID. É ela também a responsável por realizar o envio do pacote json pela rede.



Script vmstat_tracker

Este script foi implementado em *shell bash*, sendo o responsável por realizar a extração das metrcas de consumo da ferramenta vmstat. Posteriormente à extração, os dados são formatados no padrão json. A saída formata é então capturada pelo software cliente e enviada ao servidor. O parser implementado neste projeto extrai as seguintes informações:

- Procs
 - r: Numero de processos esperando o tempo de execução.
 - b: Numero de processos em sono ininterrupto.
- Memory
 - swpd: Quantidade de memória utilizada
 - free: Quantidade de memória ociosa.
 - buff: Quantidade de memória usada como *buffer*.
 - cache: Quantidade de memória usada como *cache*.
 - inact: Quantidade de memória inativa.
 - active: Quantidade de memória ativa.
- Swap

- si: Quantidade de memória *swapped* do disco.
- so: Quantidade de memória *swapped* para o disco.
- IO
 - bi: Blocos recebidos de um dispositivo de blocos (blocos/s).
 - bo: Blocos enviados para um dispositivos de blocos (blocks/s).
- System
 - in: Numero de interrupções por segundo, incluindo o *clock*.
 - cs: Numero de trocas de contexto por segundo.
- CPU
 - Porcentagem do total de tempo de CPU.
 - us: Tempo gasto rodando codigos fora do kernel. (tempo de usuário, incluindo o tempo de *nice time*.)
 - sy: Tempo gasto rodando o codigo do kernel. (Tempo de sistema)
 - id: Tempo ocioso.
 - wa: Tempo gasto esperando por I/O.
 - st: Tempo “roubado” da maquina virtual.

Abaixo segue exemplo de saida gerada pelo script:

```
string(1073) "{
    "InputDataName": "vmstat",
    "argSize": "6",
    "argName":["procs","memory","swap", "io", "system", "cpu"],
    "date":
    {
        "day_name":"Qua",
        "day":"10",
        "month":"Jul",
        "time":"00:44:42",
        "year":"2013",
        "type":"UTC"
    },
    "procs" :
    {
        "argOpSize":"2",
        "argOp":["r", "b"],
        "r": "0",
        "b": "0"
    },
    "memory" :
    {
        "argOpSize":"4",
```

```

"argOp":["swp","free", "buff", "cache"],
  "swpd": "9460",
  "free": "427076",
  "buff": "17536",
  "cache": "34160"
},
"swap" :
{
"argOpSize":"2",
"argOp":["si","so"],
  "si": "0",
  "so": "0"
},

"io" :
{
"argOpSize":"2",
"argOp":["bi","bo"],
  "bi": "0",
  "bo": "0"
},
"system" :
{
"argOpSize":"2",
"argOp":["in","cs"],
  "in": "18",
  "cs": "24"
},
"cpu" :
{
"argOpSize":"4",
"argOp":["us","sy", "id", "wa"],
  "us": "1",
  "sy": "0",
  "id": "99",
  "wa": "0"
},

"InstanceToken":"51dbaed0186c43.58804575"
}

```

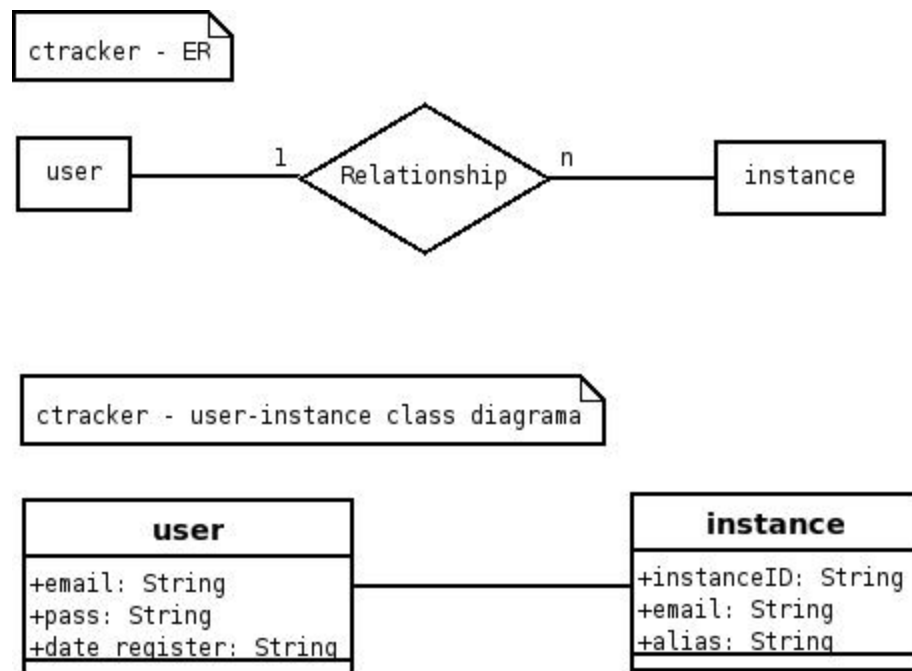
Servidor

O servidor implementado utiliza a linguagem para web PHP., sendo composto por dois módulos: captura e visualização. Para controle de acesso e de servidores monitorados foi utilizado o banco de dados MySQL. A utilização de um banco

de dados relacional para este caso é mais interessante, uma vez que as restrições relacionais mudam pouco e é necessário manter a atomicidade e sincronia entre elas.

Controle de Acesso e de Servidores

A autenticação e validação dos usuários é realizada mediante um banco de dados relacional. Dada a natureza dos dados, como dito anteriormente, foi utilizado o banco MySQL. A modelagem consiste no tipo entidade *user* (que contém email e senha do usuário) e do tipo entidade *instance* (que contém o instanceID ou tokenID, que é um valor único que identifica o servidor para um usuário específico; e o alias, que é um nome amigável definido pelo usuário para identificar o servidor). Abaixo segue o Diagrama Entidade-Relacionamento assim como o Diagrama de Classe:



Captura

No módulo de captura temos um *web-service* que recebe as mensagens enviadas pelos clientes em formato JSON e extrai os dados, inserindo -os em seguida no banco de dados MongoDB. A adoção de um banco de dados não-relacional se deu devido a própria natureza da ferramenta, uma vez que a grande maioria das interações com o banco é de Inserção e não existe restrição forte quanto a latência em que novos registros devam ser adicionados no

banco(Isso depende das necessidades do usuário ou ainda da estabilidade da rede). As operações de visualização ocorrem em poucas ocasiões, em que o administrador de sistema ou gerente de infraestrutura deseja visualizar os dados de utilização dos servidores. Além disso, cada cliente pode implementar o *parser* para capturar os dados que achar relevante, desta forma as informações enviadas ao servidor não obedecem perfeitamente as regras de normalização do modelo relacional.

MongoDB

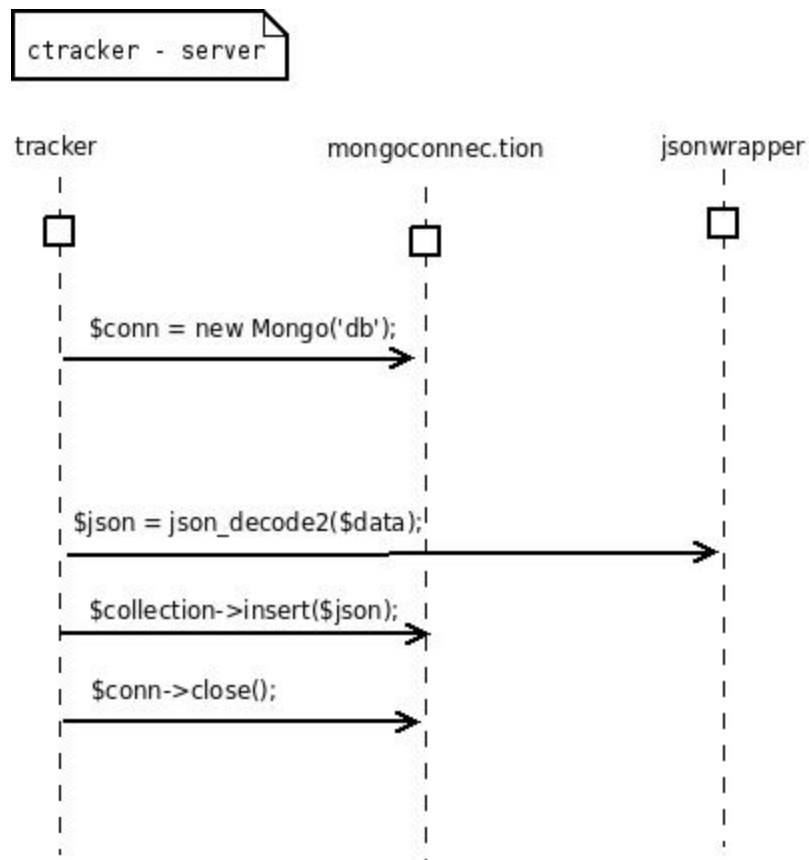
MongoDB é um banco NoSQL *open source* orientado a documentos que armazena dados estruturados na forma de documentos similar ao formato JSON conhecido como BSON, invés de tabelas, como no modelo relacional. Este tipo de documento é especialmente interessante em aplicações que se comunicam pela internet.

tracker.php

Este *web-service* recebe as mensagens em formato JSON enviadas pelos clientes e armazena no banco MongoDB. Abaixo segue o diagrama de fluxo deste processo:

Diagrama de Fluxo

Quando uma nova mensagem é recebida, uma nova conexão é criada com o banco, mediante o uso instanciando da classe Mongo. Em seguida é necessário a validação e formatação da string recebida, isto é feita pelo método `json_decode` da classe `jsonwrapper`. A inserção dos dados é feita pelo método `insert` da classe Mongo.



Visualização

O módulo de visualização é o responsável por disponibilizar os dados de uma forma intuitiva e simples para o usuário, de forma que seja criado conhecimento baseado nestes dados. Este módulo foi desenvolvido em JQuery/Javascript, dada a maturidade e a grande quantidade de bibliotecas disponiveis. Os plugins e bibliotecas utilizadas neste projeto foram:

JQuery

jQuery é uma biblioteca Javascript open-source com suporte para inumeros browsers. Foi desenhado para simplificar o desenvolvimento de scripts que rodam no navegador.

Flot

Flot é uma plugin open-source para JQuery desenvolvido para realizada de plotagem de gráficos. Possui foco na simplicidade de uso,

visual atrativo e funcoes interativas. Permite a criação de gráficos em tempo real, gráficos de barras, interatividade, zoom, dentre outros.

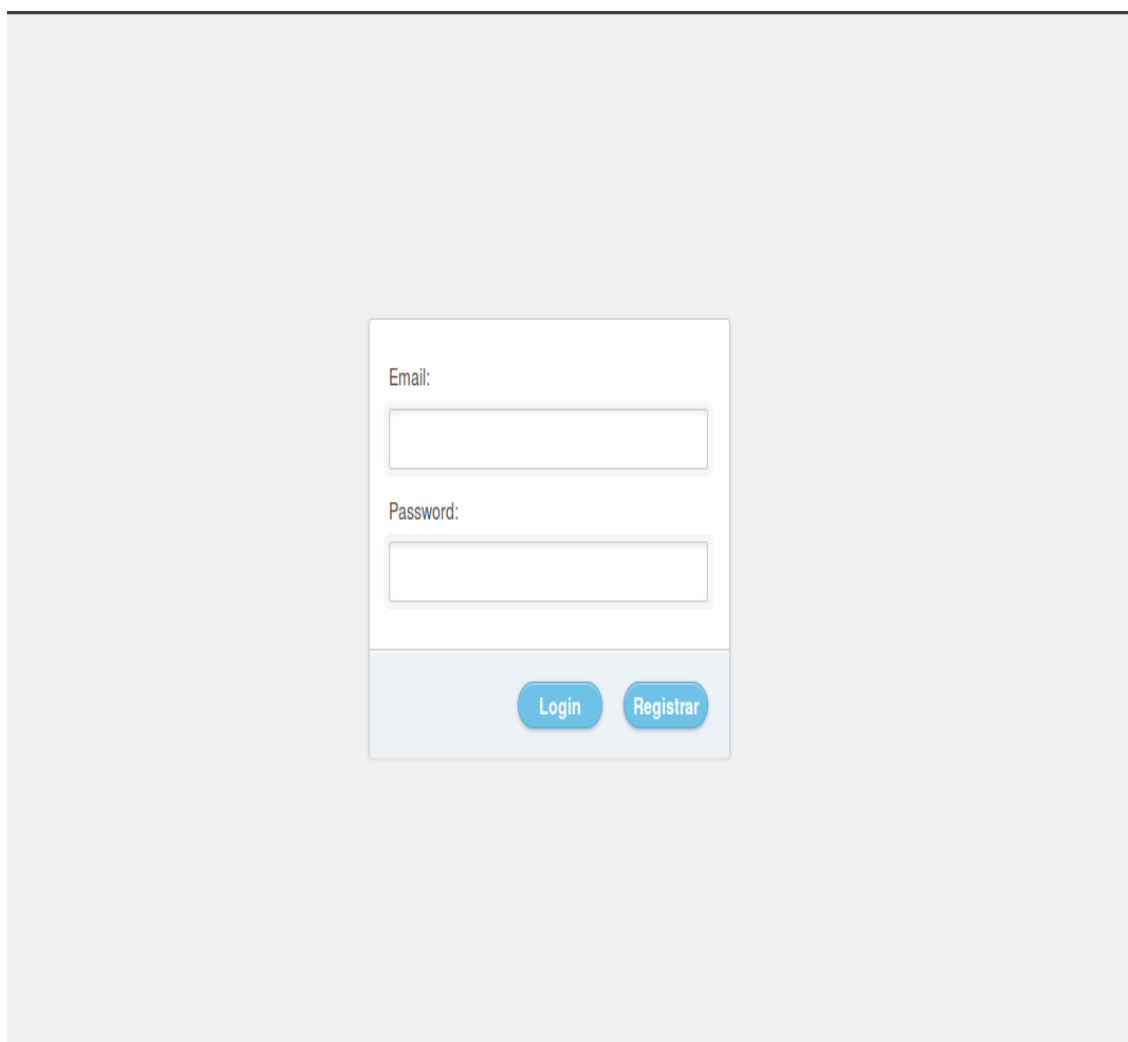
Table-sorter

Table-sorter é um plugin open-source para JQuery que permite a realização de tabelas de forma simples. Dentre as funcionalidades temos: auto-ordenação, highlighting, paginação, suporte a links e processamento de datas.

Screenshots

Segue abaixo screenshots das telas do sistema:

Login:

A login and registration form centered on a light gray background. The form is a white rectangle with a light blue footer. It contains two text input fields: one for 'Email' and one for 'Password'. Below the 'Password' field is a light blue bar containing two rounded buttons: 'Login' and 'Registrar'.

Email:

Password:

Login Registrar

Painel de controle:

Monitoramento para Cloud Computing

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero.

instanceID	Alias	Operation
51dbaed0186c43.58804575	test	Todas as métricas Gráficos Remover

10 ▾ Entries Per Page



Displaying Page 1 of 1

Tabela com todas as métricas capturadas:

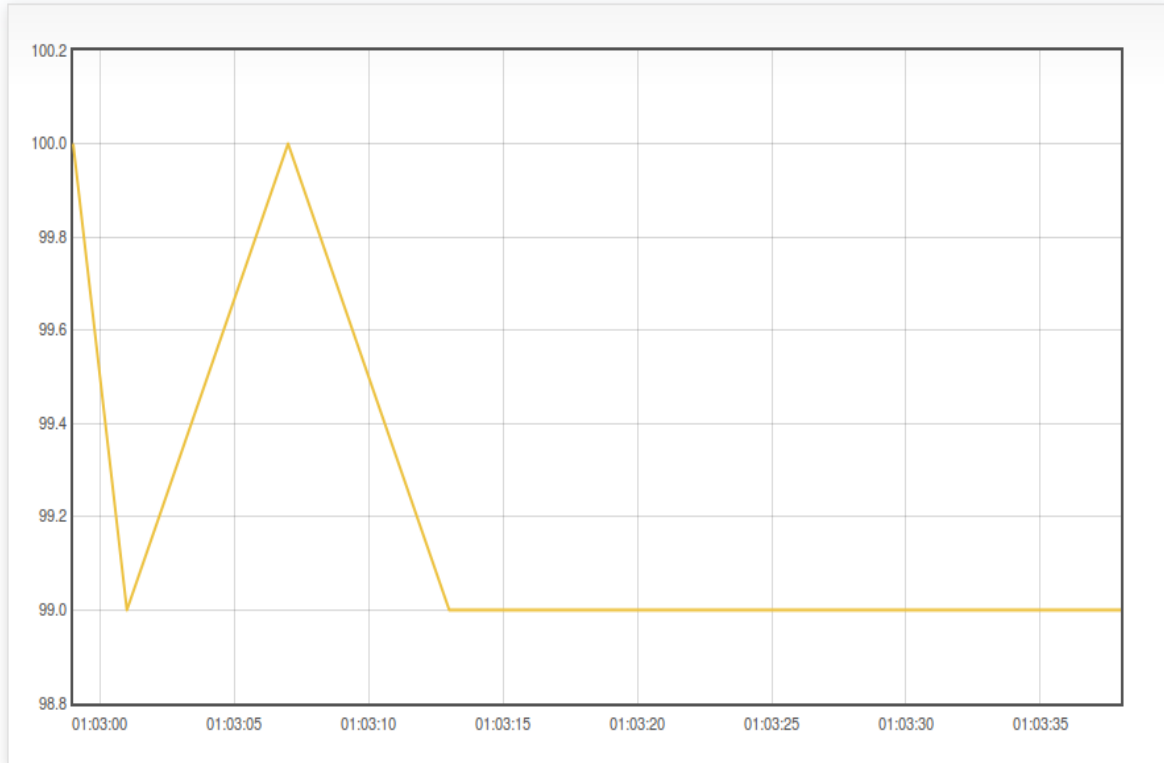
Instance: 51dbaed0186c43.58804575

InputDataName	Procs	Probs	Memory:swp	Memory:buff	Memory:cache	Memory:free	Swap:s	Swap:cs	IO:rd	IO:wr	System:in	System:cs	CPU:u	CPU:s	CPU:io	CPU:wr	Dia	Me	An	Hora
vmstat	0	2	16132	788	17656	4604	0	6140	0	6140	412	175	7	68	0	25	9	Jul	2013	19:17:49
vmstat	1	1	15412	2432	134300	347116	0	0	40	18864	201	287	0	25	0	75	9	Jul	2013	19:29:19
vmstat	0	1	15412	4416	387208	88328	0	0	0	23068	200	309	1	24	0	75	9	Jul	2013	19:29:32
vmstat	0	1	15412	3860	468036	6360	0	0	0	18864	219	348	0	42	0	58	9	Jul	2013	19:29:38
vmstat	0	0	0	17952	41336	420892	0	0	20	0	14	35	0	0	97	3	9	Jul	2013	19:10:47
vmstat	0	0	0	17956	41340	420884	0	0	0	0	35	29	0	0	100	0	9	Jul	2013	19:10:49
vmstat	0	0	0	17964	41340	420884	0	0	0	0	49	27	0	1	99	0	9	Jul	2013	19:10:55
vmstat	0	0	0	17972	41340	421148	0	0	0	0	48	28	0	0	100	0	9	Jul	2013	19:11:01
vmstat	0	0	0	17980	41340	421148	0	0	0	4	47	29	0	0	100	0	9	Jul	2013	19:11:08
vmstat	0	0	0	17988	41340	421148	0	0	0	0	43	32	0	1	99	0	9	Jul	2013	19:11:14
vmstat	0	0	0	17996	41340	421148	0	0	0	0	13	25	0	2	98	0	9	Jul	2013	19:11:21
vmstat	0	0	0	18004	41340	421024	0	0	0	0	13	25	0	0	100	0	9	Jul	2013	19:11:27
vmstat	0	0	0	18012	41340	421148	0	0	0	0	12	20	0	0	100	0	9	Jul	2013	19:11:33
vmstat	0	0	0	18020	41340	421148	0	0	0	0	13	24	0	1	99	0	9	Jul	2013	19:11:40
vmstat	0	0	0	18028	41340	421024	0	0	0	0	14	26	0	1	99	0	9	Jul	2013	19:11:46
vmstat	0	0	0	18036	41340	421024	0	0	0	0	18	30	0	1	99	0	9	Jul	2013	19:11:52
vmstat	0	0	0	18044	41340	421024	0	0	0	0	34	28	0	1	99	0	9	Jul	2013	19:11:59
vmstat	0	0	0	18052	41340	421148	0	0	0	0	43	27	0	0	100	0	9	Jul	2013	19:12:05
vmstat	0	0	0	18060	41340	421024	0	0	0	0	37	27	0	0	100	0	9	Jul	2013	19:12:11
vmstat	0	0	0	18068	41340	421024	0	0	0	0	36	26	0	0	100	0	9	Jul	2013	19:12:18
vmstat	0	0	0	18076	41340	421148	0	0	0	0	39	27	0	2	98	0	9	Jul	2013	19:12:24
vmstat	0	0	0	18084	41340	421024	0	0	0	0	38	27	0	0	100	0	9	Jul	2013	19:12:30
vmstat	0	0	0	18092	41340	420900	0	0	0	0	34	24	0	0	100	0	9	Jul	2013	19:12:37
vmstat	0	0	0	18100	41340	421024	0	0	0	0	34	26	0	0	100	0	9	Jul	2013	19:12:43
vmstat	0	0	0	18108	41340	421024	0	0	0	0	34	24	1	0	99	0	9	Jul	2013	19:12:49
vmstat	0	0	0	18116	41340	421024	0	0	0	0	30	22	0	2	98	0	9	Jul	2013	19:12:56
vmstat	0	0	0	18124	41340	421024	0	0	0	0	26	30	0	1	99	0	9	Jul	2013	19:13:02
vmstat	0	0	0	18132	41340	421024	0	0	0	0	23	28	0	1	99	0	9	Jul	2013	19:13:08

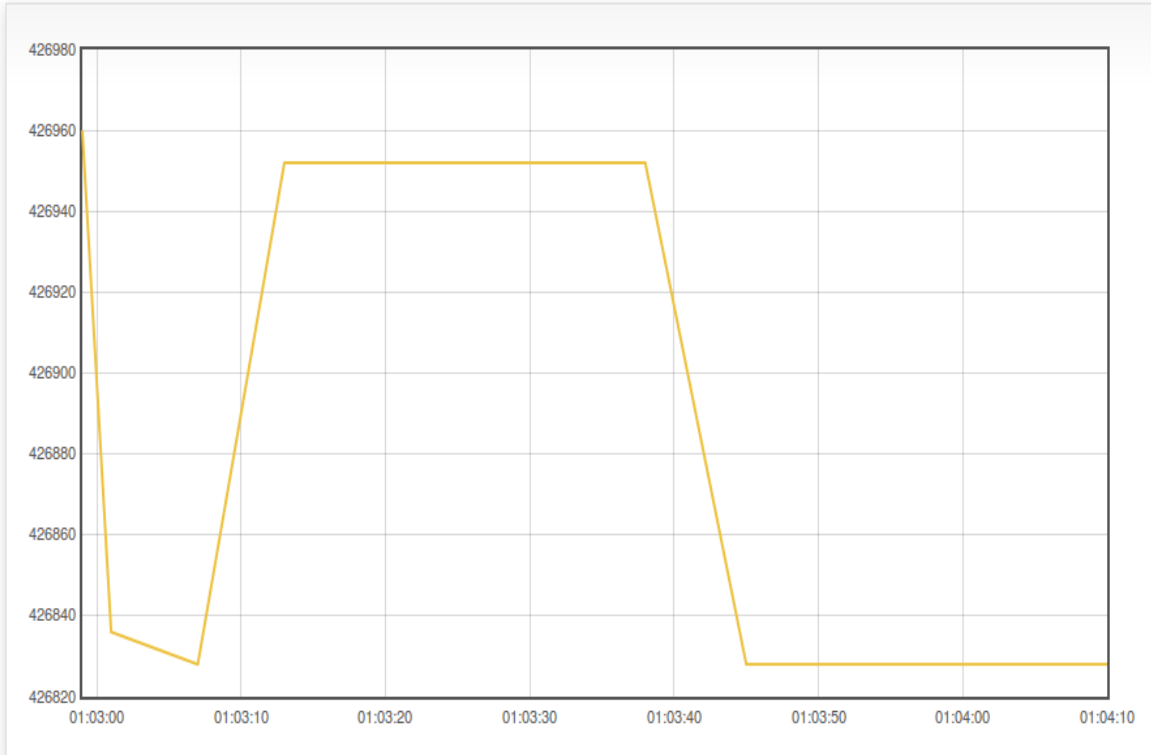
Gráficos em tempo real:

Real-time updates

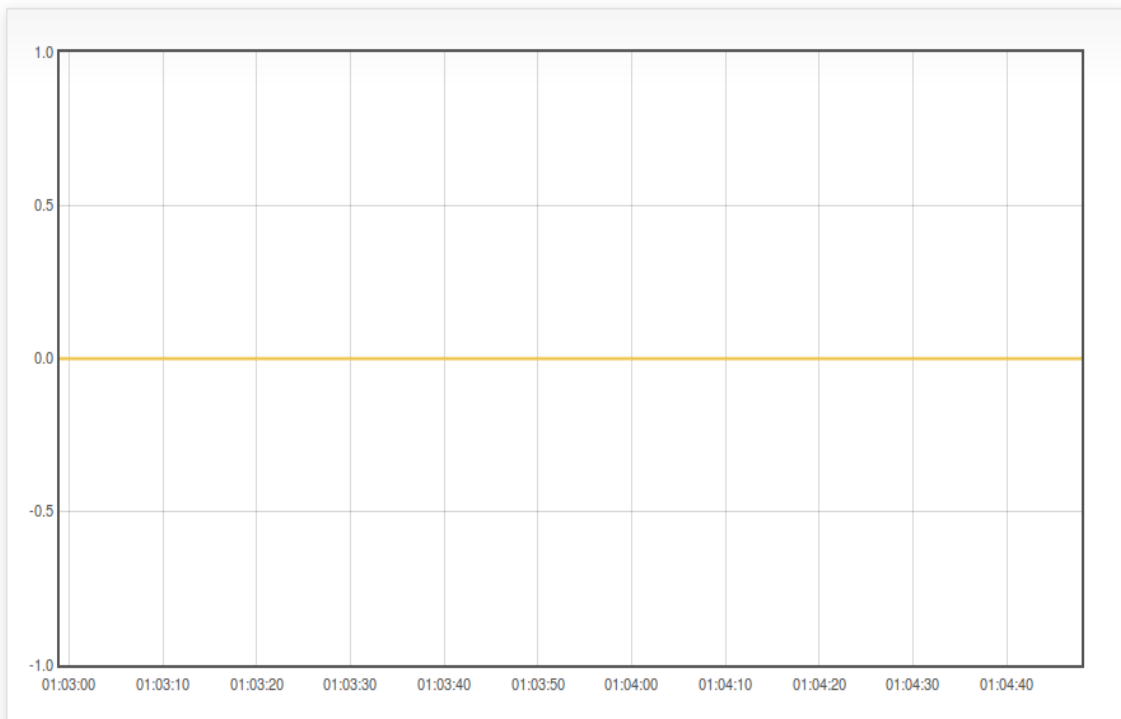
CPU - id: Time spent idle(Percentage)



Memory - free

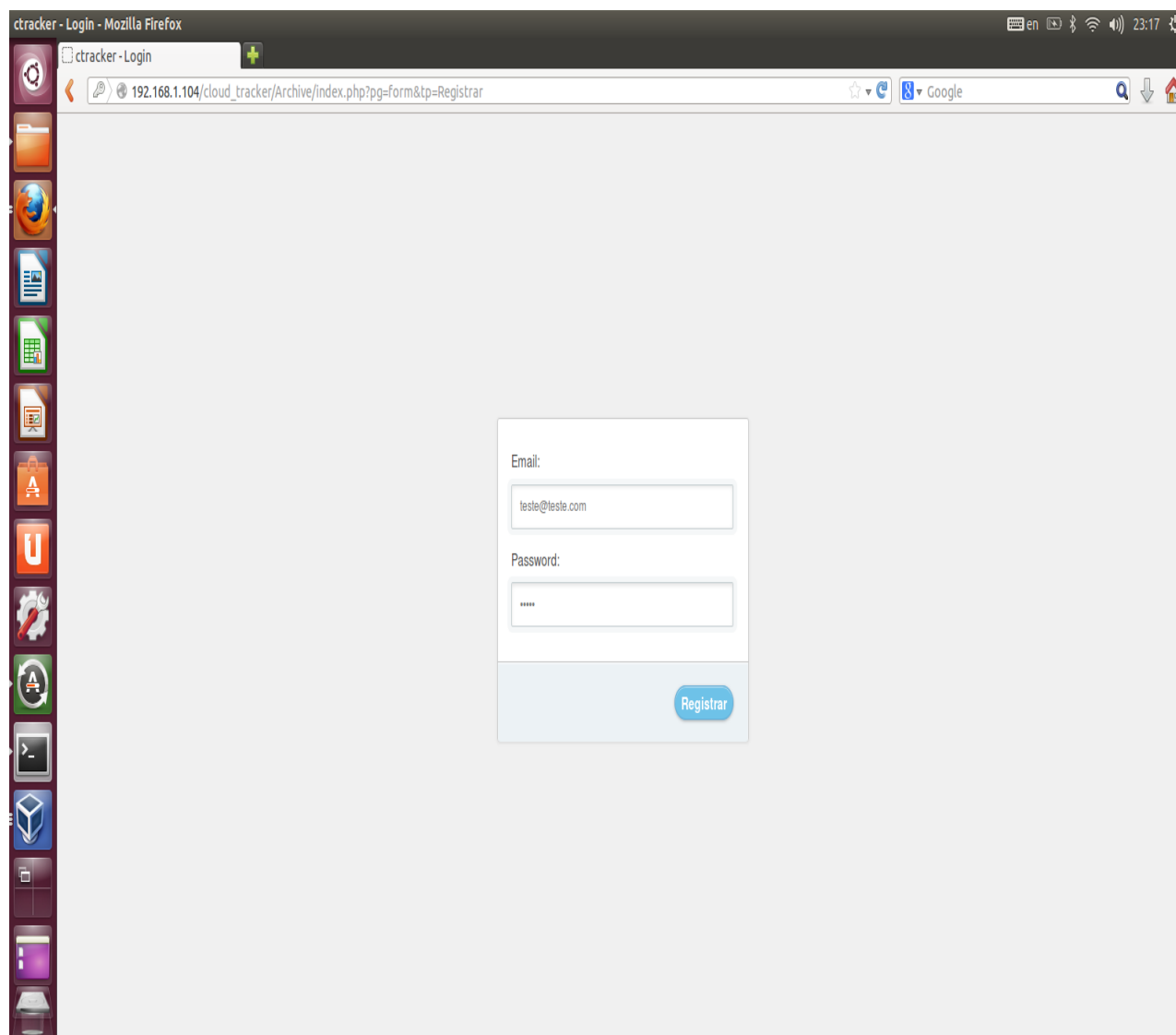


Disk - bo: Blocks sent to a block device (blocks/s)

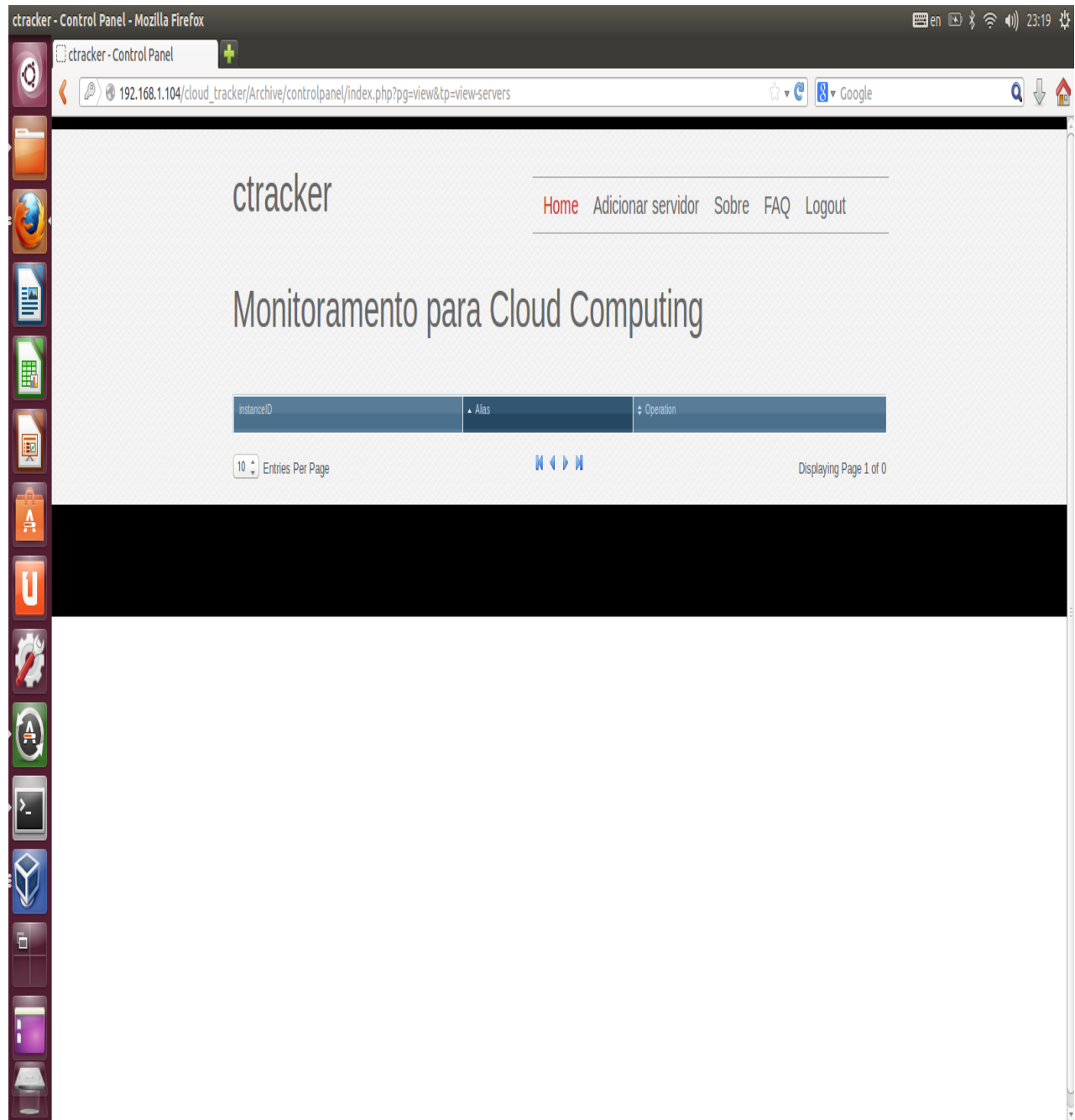


Como utilizar

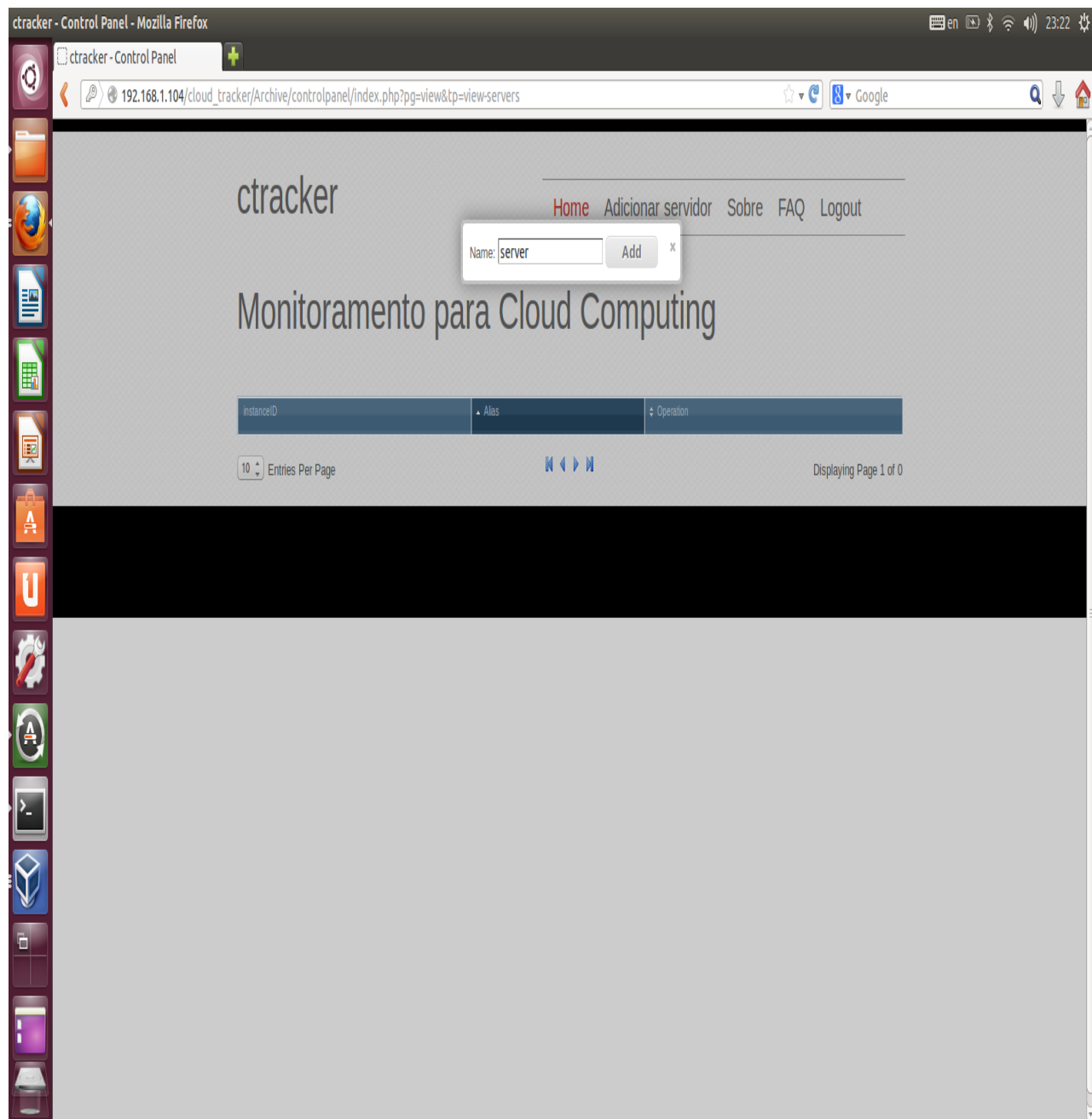
Primeiro é necessário criar uma conta de usuário no sistema. Para isso acesse o site pelo navegador e preencha com um novo e-mail para o usuário e uma senha:



Em seguida, clique no botão Registrar. Ao final do processo um novo usuário será criado e o sistema irá abrir o Painel de Controle. Nele é possível adicionar um novo servidor, remover e acompanhar os gráficos gerados:



Para adicionar um novo servidor clique em "Adicionar servidor" e digite um nome para o mesmo:



Em seguida, o novo servidor será mostrado na tabela do Painel de Controle:


```
client@UbuntuServer:~/client$ ./dist/Debug/GNU-Linux-x86/cloud
```

O cliente começara o envio das mensagens JSON com as métricas coletadas, como segue:

```
},
"memory" :
{
  "argOpSize": "4",
  "argOp": ["swp", "free", "buff", "cache"],
  "swpd": "0",
  "free": "409892",
  "buff": "18104",
  "cache": "45736"
},
"swap" :
{
  "argOpSize": "2",
  "argOp": ["si", "so"],
  "si": "0",
  "so": "0"
},
"io" :
{
  "argOpSize": "2",
  "argOp": ["bi", "bo"],
  "bi": "0",
  "bo": "0"
},
"system" :
{
  "argOpSize": "2",
  "argOp": ["in", "cs"],
  "in": "10",
  "cs": "19"
},
"cpu" :
{
  "argOpSize": "4",
  "argOp": ["us", "sy", "id", "wa"],
  "us": "0",
  "sy": "0",
  "id": "100",
  "wa": "0"
},
"InstanceToken": "51e4aeb34613d1.25062641"
}"}
```

Para visualizar o uso do servidor existem duas opções no Painel de Controle: Gráficos em tempo real e tabela com todos os dados.

Opção Gráficos:



Opção "Todas as métricas"

Totas as metricas

192.168.1.104/cloud_tracker/vmstat?itoken=51e4aeb34613d1.25062641

Google

Instance: 51e4aeb34613d1.25062641

InputDataName	Procs:r	Probs:b	Memory:swpd	Memory:buff	Memory:cache	Memory:free	Swap:si	Swap:so	IO:bi	IO:bo	System:in	System:cs	CPU:us	CPU:sy	CPU:id	CPU:wa	Dia	Mes	Ano	Hora
vmstat	0	1	0	18112	45736	409892	0	0	0	0	12	24	0	0	0	100	16	Jul	2013	02:32:43
vmstat	0	1	0	18120	45736	410016	0	0	0	0	17	35	0	0	0	100	16	Jul	2013	02:32:46
vmstat	0	1	0	18144	45736	409876	0	0	0	0	15	30	0	0	0	100	16	Jul	2013	02:33:04
vmstat	0	1	0	18168	45736	409752	0	0	0	0	14	23	0	0	0	100	16	Jul	2013	02:33:25
vmstat	0	1	0	18208	45736	409752	0	0	0	0	64	27	0	0	0	100	16	Jul	2013	02:33:55
vmstat	0	1	0	18240	45736	409752	0	0	0	0	10	21	0	0	0	100	16	Jul	2013	02:34:20
vmstat	0	1	0	18272	45736	409876	0	0	0	0	14	19	0	0	0	100	16	Jul	2013	02:34:45
vmstat	0	1	0	18288	45736	409752	0	0	0	0	91	28	0	0	0	100	16	Jul	2013	02:34:58
vmstat	0	1	0	18296	45736	409752	0	0	0	0	14	27	0	0	0	100	16	Jul	2013	02:35:04
vmstat	0	0	0	18068	45736	410016	0	0	0	0	13	27	0	0	99	1	16	Jul	2013	02:31:39
vmstat	0	0	0	18072	45736	409892	0	0	0	0	13	23	0	0	100	0	16	Jul	2013	02:31:41
vmstat	0	0	0	18080	45736	409768	0	0	0	52	19	36	0	0	99	1	16	Jul	2013	02:31:44
vmstat	0	0	0	18080	45736	410016	0	0	0	0	12	23	0	0	100	0	16	Jul	2013	02:31:47
vmstat	0	0	0	18104	45736	410016	0	0	0	0	17	23	0	0	100	0	16	Jul	2013	02:32:36
vmstat	0	0	0	18104	45736	409892	0	0	0	0	10	19	0	0	100	0	16	Jul	2013	02:32:37
vmstat	0	0	0	18104	45736	409892	0	0	0	0	17	30	0	0	100	0	16	Jul	2013	02:32:40
vmstat	0	0	0	18120	45736	409892	0	0	0	0	9	19	0	0	100	0	16	Jul	2013	02:32:49
vmstat	0	0	0	18128	45736	409892	0	0	0	0	14	28	0	0	100	0	16	Jul	2013	02:32:52
vmstat	0	0	0	18128	45736	409892	0	0	0	0	11	20	0	0	100	0	16	Jul	2013	02:32:55
vmstat	0	0	0	18136	45736	409876	0	0	0	0	20	39	0	0	85	15	16	Jul	2013	02:32:58
vmstat	0	0	0	18136	45736	409876	0	0	0	0	10	20	0	0	100	0	16	Jul	2013	02:33:01
vmstat	0	0	0	18144	45736	409876	0	0	0	0	13	26	0	0	3	97	16	Jul	2013	02:33:07
vmstat	0	0	0	18152	45736	409876	0	0	0	0	14	27	0	0	100	0	16	Jul	2013	02:33:13
vmstat	0	0	0	18152	45736	409876	0	0	0	0	12	24	0	0	100	0	16	Jul	2013	02:33:16
vmstat	0	0	0	18160	45736	409752	0	0	0	0	13	25	0	0	100	0	16	Jul	2013	02:33:19
vmstat	0	0	0	18160	45736	409752	0	0	0	0	13	27	0	0	100	0	16	Jul	2013	02:33:22
vmstat	0	0	0	18168	45736	409752	0	0	0	0	15	29	0	0	4	96	16	Jul	2013	02:33:28
vmstat	0	0	0	18176	45736	409752	0	0	0	0	15	29	0	0	49	51	16	Jul	2013	02:33:31
vmstat	0	0	0	18176	45736	409876	0	0	0	0	13	25	0	0	100	0	16	Jul	2013	02:33:34
vmstat	0	0	0	18184	45736	409876	0	0	0	0	12	24	0	0	100	0	16	Jul	2013	02:33:37
vmstat	0	0	0	18184	45736	409752	0	0	0	0	12	23	0	0	100	0	16	Jul	2013	02:33:40
vmstat	0	0	0	18192	45736	409752	0	0	0	0	15	27	0	0	100	0	16	Jul	2013	02:33:43
vmstat	0	0	0	18192	45736	409752	0	0	0	0	32	28	0	0	100	0	16	Jul	2013	02:33:46
vmstat	0	0	0	18200	45736	409752	0	0	0	0	59	23	0	0	100	0	16	Jul	2013	02:33:49

Simulação de uso:

Afim de simular o uso dos servidores, é possível a utilização de um conjunto de scripts e códigos.

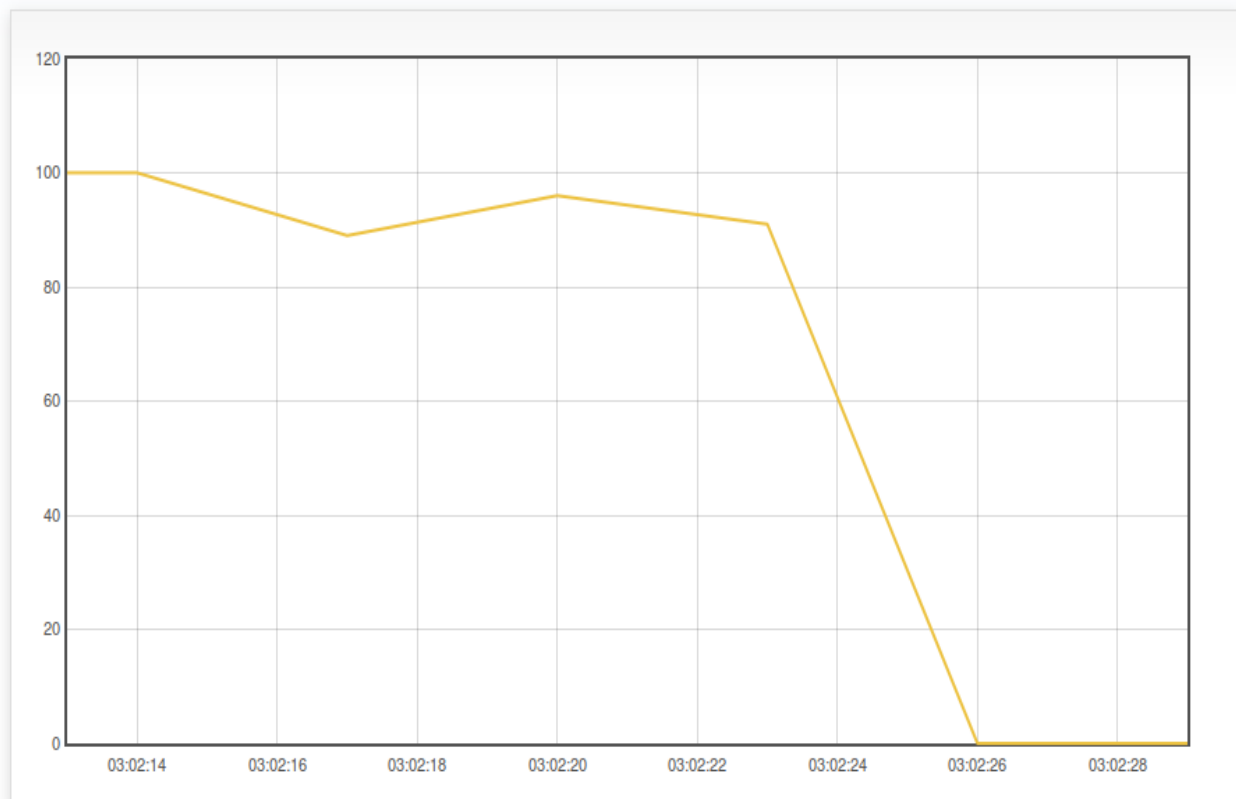
Carga de CPU:

Para simular o uso de CPU, utilizamos script que "varre" constantemente uma matriz bi-dimensional, criando inúmeros forks dos processos. Isso força o uso da CPU uma vez que operações lógico-aritméticas são realizadas:

```
client@UbuntuServer:~/bechmark/scripts$ ./cpu_loadserver.sh  
PIDS: 2523 2522 2521 2520 2519 2518 2517 2516 2515 2514  
client@UbuntuServer:~/bechmark/scripts$
```

Resultado obtido com a opção Gráficos:

CPU - id: Time spent idle(Percentage)



Neste gráfico o tempo ocioso(idle - id) da CPU cai para zero por cento.

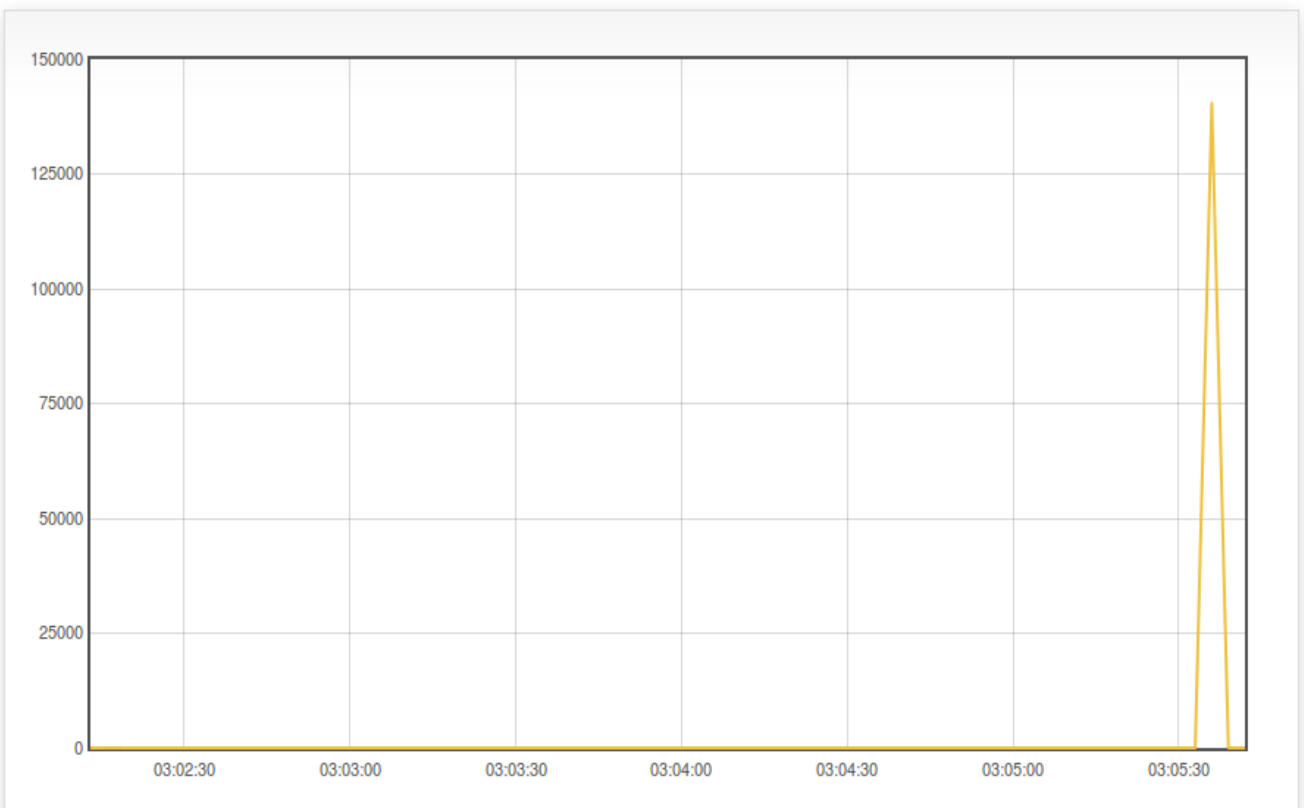
Uso de Disco:

Este script realiza a criação de um arquivo de 500MB cheio de 0(zero) no disco, criando consideravel carga:

```
client@UbuntuServer:~/bechmark/scripts$ ./disk.sh
```

Resultado obtido com a opção Gráficos:

Disk - bo: Blocks sent to a block device (blocks/s)



Neste gráfico é possível observar que blocos de 150kb estão sendo enviados ao disco, por isso o gráfico aumenta abruptamente. Ao final da criação do arquivo a quantidade de blocos enviados volta para zero.

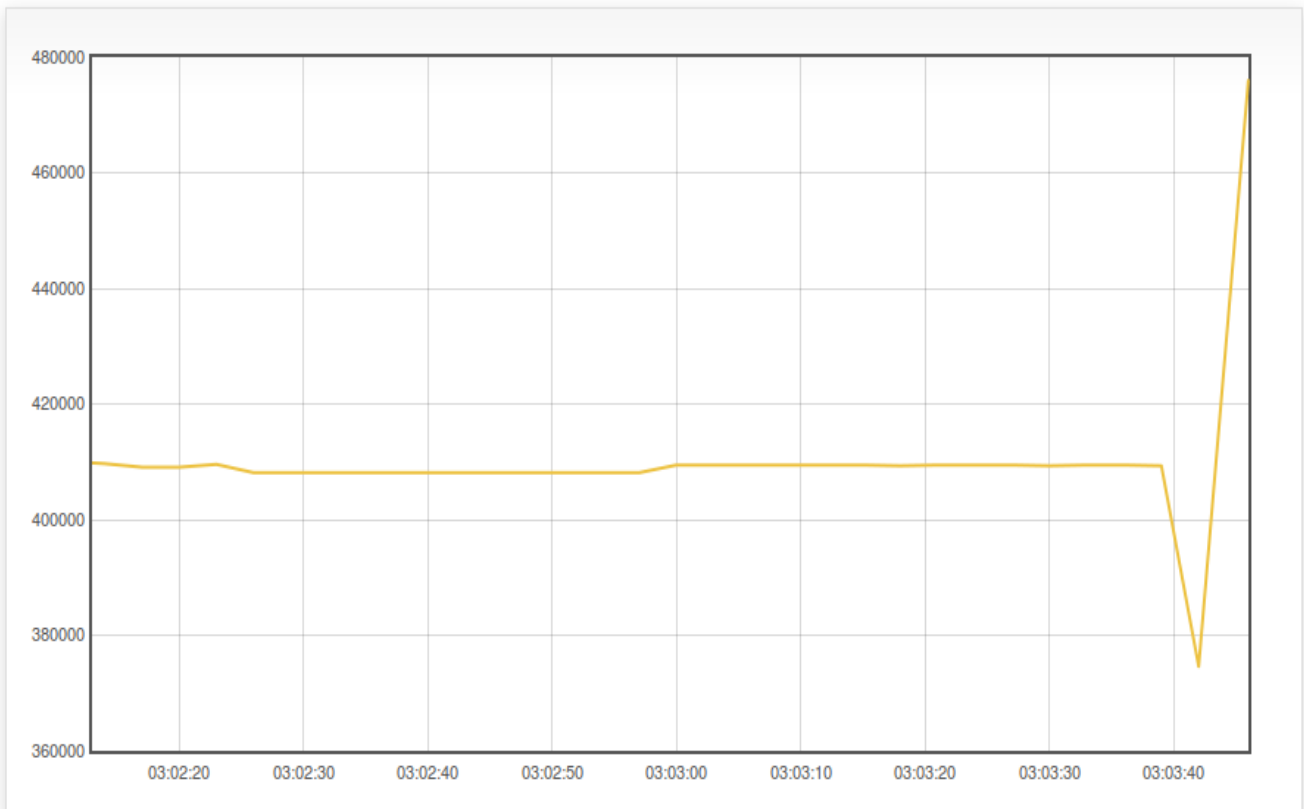
Uso de memória:

Este código feito em C realiza a alocação de memória de forma dinâmica por um tempo especificado. Isso acarreta num grande consumo da mesma e, quando o limite está sendo alcançado, força o sistema operacional a realizar swap com o disco, uma vez que utiliza a função realloc:

```
client@UbuntuServer:~/bechmark/scripts$ ./high_memory_usage
```

Resultado obtido com a opção Gráficos:

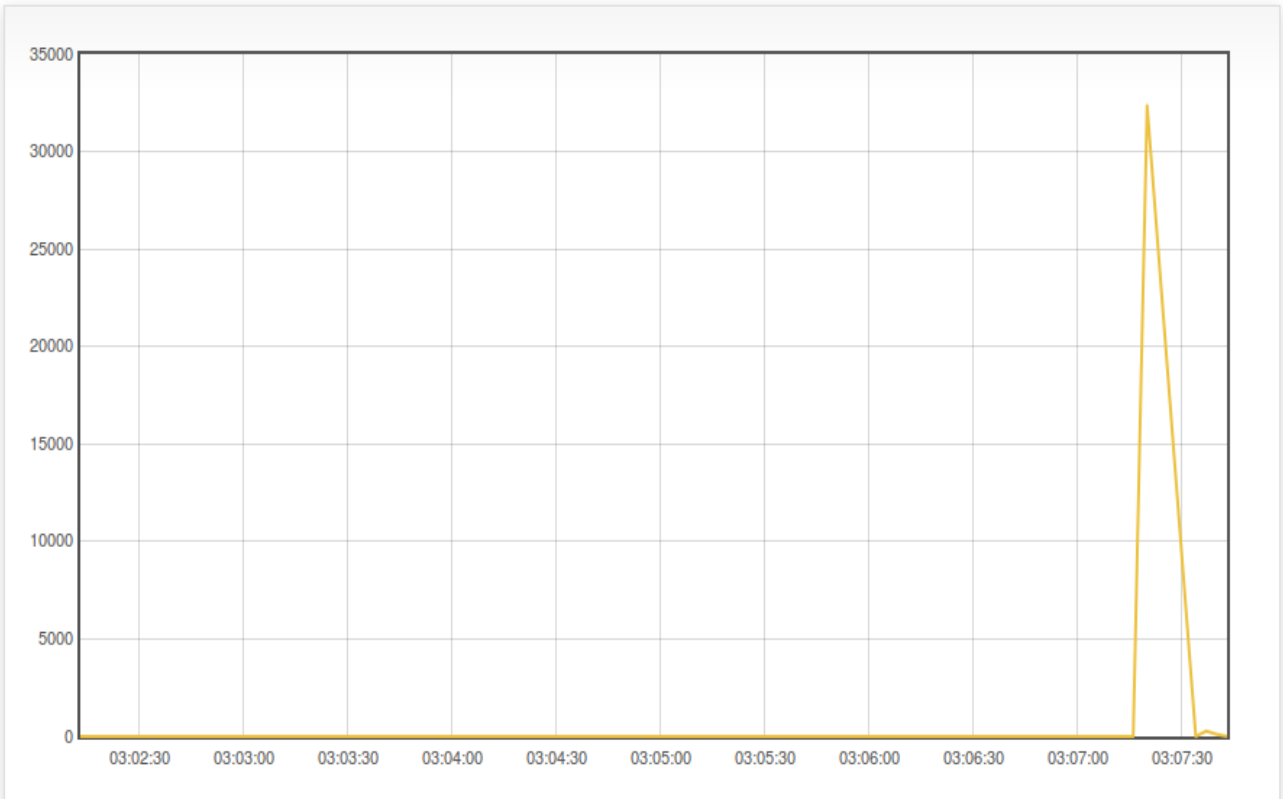
Memory - free



Neste gráfico é possível ver que a quantidade de memória livre (free) cai consideravelmente devido à execução do código. Quando o processo termina o tempo de alocação de memória a mesma é liberada. Como

ilustrada no gráfico.

Swap - so: Amount of memory swapped to disk (/s).



Neste gráfico é possível verificar que quando o uso da memória chega a um determinado limite, o sistema operacional realiza o swap para o disco, afim de poder continuar fornecendo memória para o processo.

Resultados obtidos

Após implementado, o programa foi capaz de ler informações sobre os itens analisados e gerar gráficos *consumo x tempo* que ilustrarão a variação do consumo durante um certo período de tempo que será determinado pelo usuário. Com base nesses dados, o administrador tem uma ilustração de como os recursos de seu servidor estão sendo utilizados, podendo investir mais no aprimoramento de um item específico. O programa busca, com isso, aumentar o desempenho de servidores em nuvem, já que, com base nos gráficos gerados, é possível investir dinheiro e tempo em uma única parte

de hardware, por exemplo, que está sendo muito requisitada. Assim, haverá uma economia grande, seja financeira ou de tempo, para resolver possíveis problemas que o servidor venha a sofrer.

Bibliografia

- Amazon EC2 - <http://aws.amazon.com/ec2/>
- Cloud Computing - https://en.wikipedia.org/wiki/Cloud_computing
- libcurl - <http://curl.haxx.se/>
- JQuery - <http://jquery.com/>
- Flot - <http://flotcharts.org/>
- Tablesorter - <http://www.scriptiny.com/2009/03/table-sorter/>
- PHP - <http://php.net/>
- MySQL - <http://www.mysql.com/>
- MongoDB - <http://www.mongodb.org/>
- NoSQL - <https://en.wikipedia.org/wiki/Nosql>
- JSON - <http://en.wikipedia.org/wiki/JSON>