

## MEMCACHED COMO UMA ALTERNATIVA DE CACHE PARA ARQUITETURAS DE ALTA ESCALABILIDADE

BERGAMASCHI, Felipe

GOMEDE, Everton (Orientador Prof. de Gerenciamento de Projetos, Mestre em Ciência da Computação - evertongomede@gmail.com)

### INTRODUÇÃO

Este artigo visa avaliar a ferramenta de cache, Memcached, aplicada em um ambiente de missão crítica, construído em uma arquitetura REST com a linguagem PHP, comparando as métricas estabelecidas de performance como também a estabilidade do ambiente.

O ambiente modelo utilizado é a plataforma AVA da instituição EAD da Unicesumar. A plataforma é considerada legado devido a sua idade e versões dos componentes, utiliza o Moodle como base sendo o mesmo implementado na linguagem PHP. Esse sistema passou por um processo batizado de "facelift" no qual foi refatorada a interface para aprimorar a interação com o usuário e também separar a comunicação com o banco de dados, a fim de disponibilizar a toda infraestrutura da instituição um serviço com APIs no padrão REST. Todo o decurso pretende dar mais um ano de vida, impulsionando a plataforma e engine do Moodle considerado legado.

### REVISÃO DE LITERATURA

De acordo com o Patrick Galbraith (2009), o Memcached pode ser definido como um simples servidor de memória de alta performance que disponibiliza uma camada de cache para aplicações armazenarem dados e informações de maneira que não sobrecarregue o banco de dados.

Cache é uma camada de armazenamento de informações de acesso rápido (preferencialmente de alta performance) para substituir o acesso a meios mais lentos, como um banco de dados. Performance é definido como desempenho da aplicação, sua velocidade, ou seja, a capacidade de executar algo rapidamente.

Da mesma forma, de acordo com Patrick Galbraith (2009), a parte de Front-end de uma arquitetura pode ser definida como servidor que entrega os arquivos e acessos que o usuário irá interagir.

O Back-end é representado pelos servidores que possuem todo a lógica e regra de negócio, neles estão concentrados todo o processamento e interação com banco de dados.

Segundo Patrick Galbraith (2009), com a programação Orientada a Objetos, o foco do usuário é escrever uma aplicação que use o objeto sem estar preocupado com a forma que esse objeto funciona internamente. A orientação do objeto torna a implementação mais fácil e rápida, pois você possui um código reutilizável e também o torna esteticamente agradável para ler e mais fácil de evoluir.

O Encapsulamento é um termo usado na orientação a objetos que define como o acesso a um determinado membro de uma classe (um atributo ou método) é escondido dentro da mesma, ou seja, impedindo o acesso direto a eles. A interface da classe é quem fornece acesso a membros encapsulados. Handler é entendido como um manipulador que processa determinados tipos de arquivos e gera algum tipo de conteúdo.

### MATERIAIS & MÉTODOS

Será utilizado uma infraestrutura de datacenter HP, que conta com 12 lâminas com processadores Xeon fornecendo 500GHz, 5TB de memória e storage em SSD de alta performance 3PAR, com ambientes virtuais gerenciados pelo VMWare vSphere 5 Enterprise com intuito de levantar a plataforma completa e realizar os benchmarks necessários.

As versões dos pacotes utilizados foram durante os procedimentos são as últimas disponíveis nos repositórios oficiais e homologados pela Redhat na versão 7.x de seu sistema operacional, o que visa garantir a máxima segurança, performance e estabilidade.

Na tabela 01 está especifica as instalações e configurações de cada ambiente que foi realizado o estudo.

Tabela 01 – Configuração das máquinas virtuais.

VM	Memcached	Front-end	Back-end
CONFIGURAÇÃO	<ul style="list-style-type: none"> <li>OS Redhat 7.4</li> <li>Processador 4 núcleos</li> <li>Memória 16GB</li> <li>Disco 20GB</li> <li>Rede 10 Gigabit</li> <li>Memcached 1.4.15</li> </ul>	<ul style="list-style-type: none"> <li>OS Redhat 7.4</li> <li>Processador 8 núcleos</li> <li>Memória 8GB</li> <li>Disco 20GB</li> <li>Disco NFS de 1TB</li> <li>Rede 10 Gigabit</li> <li>Apache 2.4</li> <li>PHP 5.4.16</li> </ul>	<ul style="list-style-type: none"> <li>OS Redhat 7.4</li> <li>Processador 4 núcleos</li> <li>Memória 8GB</li> <li>Disco 20GB</li> <li>Disco NFS de 1TB</li> <li>Rede 10 Gigabit</li> <li>Apache 2.4</li> <li>PHP 5.4.16</li> </ul>

Foi utilizado o módulo “php-pecl-memcache” para o PHP ter acesso aos recursos do Memcached bem como para disponibilizar o handler de session a partir de uma interface de código orientada a objetos. Como também é necessário uma interação da aplicação com o memcached, foi implementada algumas classes e abstrações de serviços para facilitar a configuração e manipulação.

As classes não estabelecem conexão diretamente com o Memcached pois utilizam da session do PHP assim evitando conexões extras pois a mesma é gerenciada pelo próprio servidor que por sua vez já está sendo armazenada no memcached. Essa dinâmica evita codificações e tratamentos extras que podem surgir possíveis pontos de falhas. Resumidamente seria que as classes encapsulam a interação com a session e criam uma área reservada para informações de cache.

Com todas as configurações definidas e estabelecidas à aplicação, alteramos o PHP para escrever a sessão (PHP Session) no Memcached usando o handler, disponibilizado pelo módulo, ao invés de manipular em arquivos físicos, que por sua vez é ineficiente no cenário atual.

### REFERÊNCIAS

- Soliman, Ahmed. Getting Started with Memcached. Birmingham: Packt Publishing, novembro, 2013.
- Galbraith, Patrick. Developing Web Applications with Perl, memcached, MySQL® and Apache. Indianapolis: Wiley Publishing, junho, 2009.
- Apache JMeter™. Apache. Disponível em: <<http://jmeter.apache.org/index.html>>. Acesso em: 12 de janeiro de 2018.
- memcache. Pecl. Disponível em: <<https://pecl.php.net/package/memcache>>. Acesso em: 12 de janeiro de 2018.
- Memcache. PHP. Disponível em: <<http://www.php.net/memcache>>. Acesso em: 12 de janeiro de 2018.

### RESULTADOS

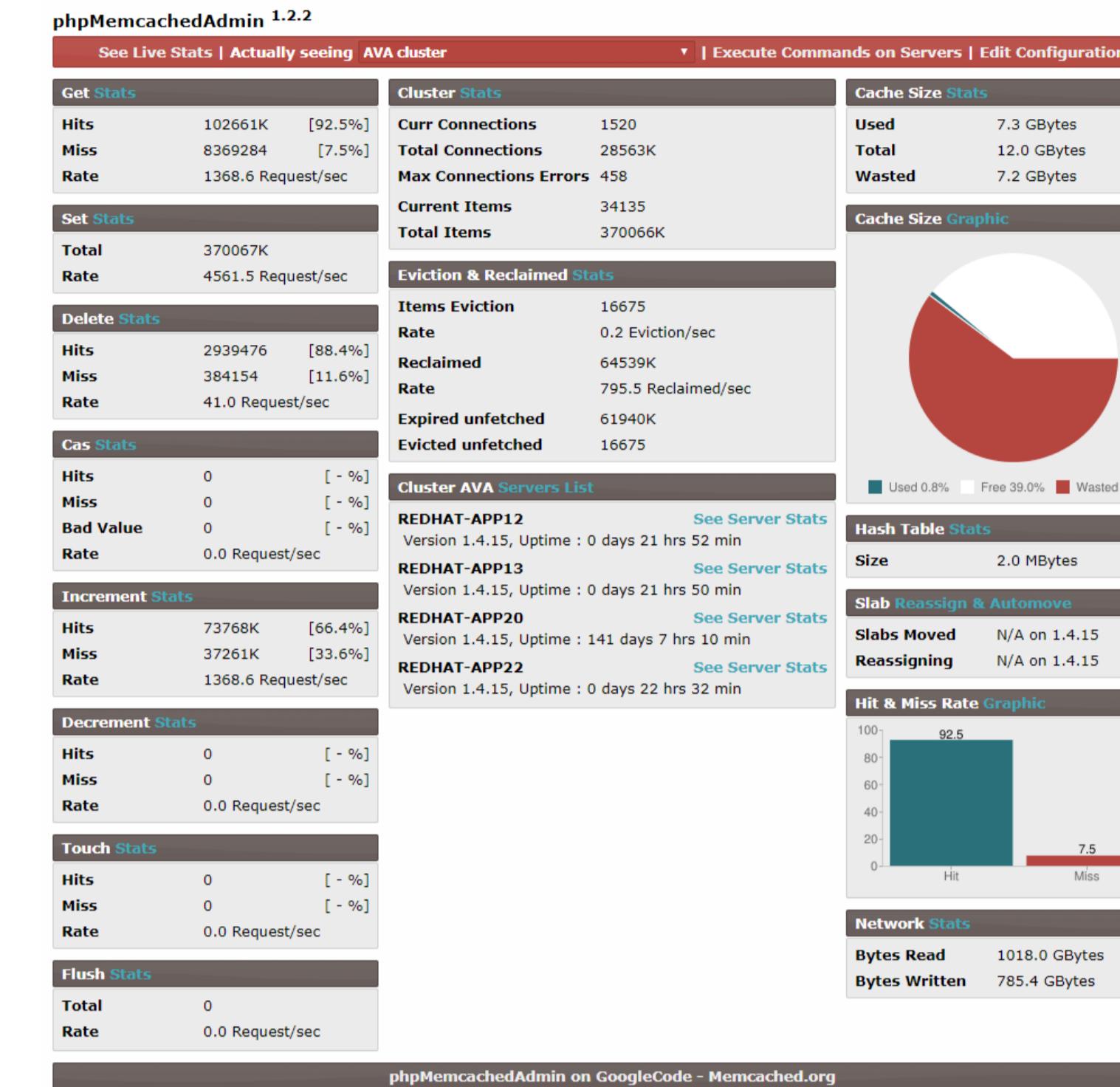


Figura 01 - Captura do dashboard em 08/12/2017 às 09:40.

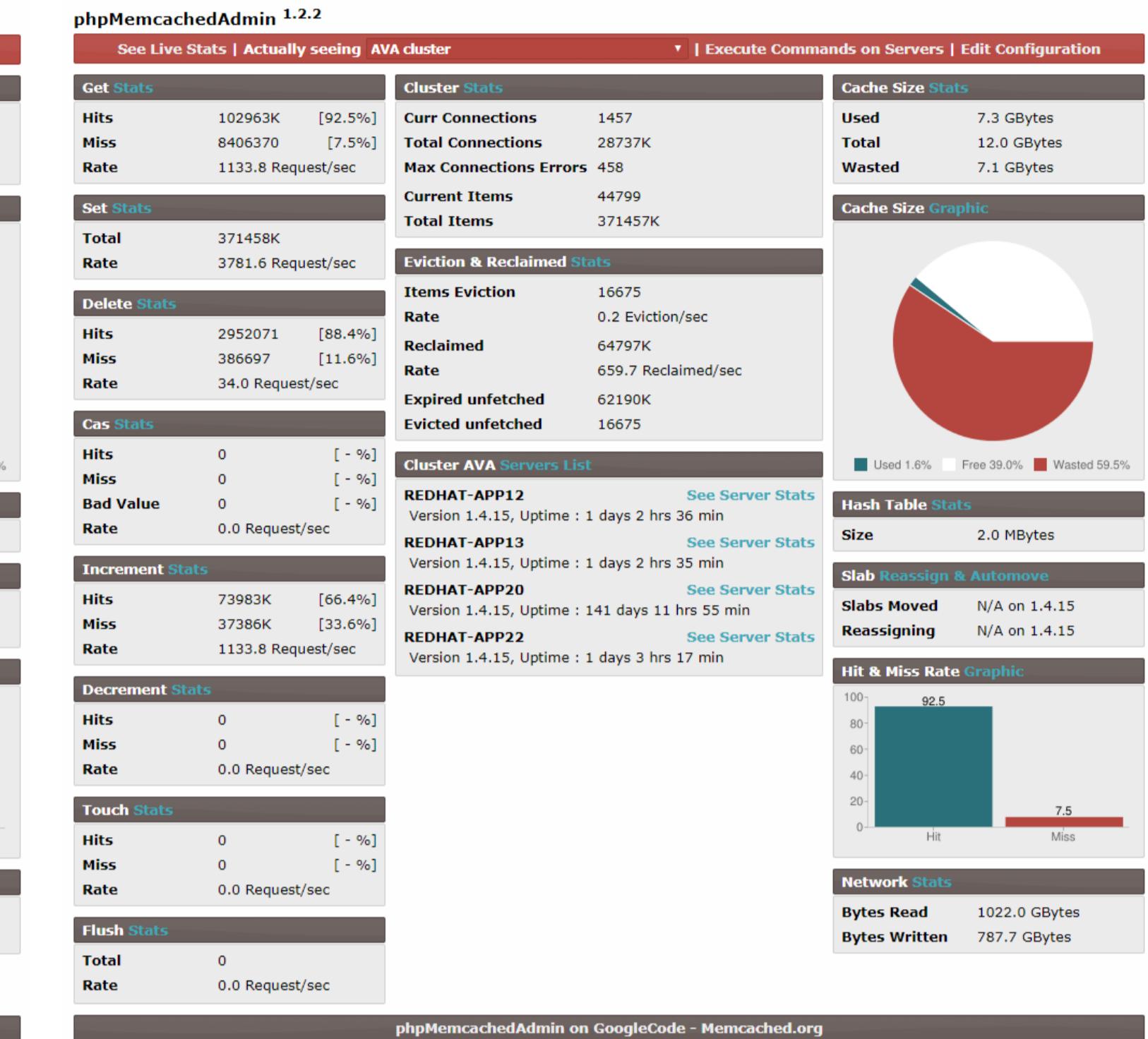


Figura 02 - Captura do dashboard em 08/12/2017 às 14:25.

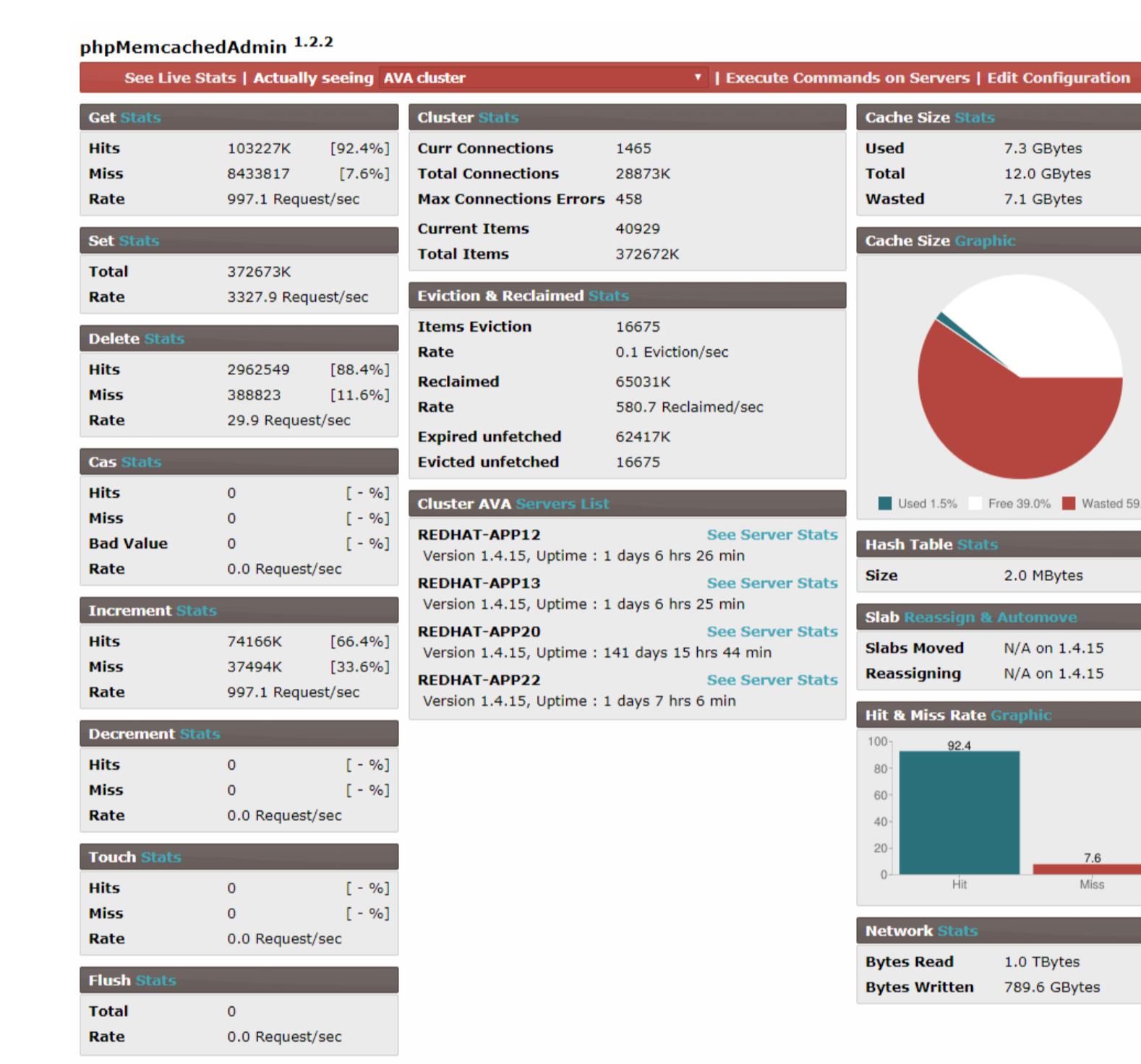


Figura 03 - Captura do dashboard em 08/12/2017 às 18:15.

Como os dados foram coletados em um período de fim do ano letivo os acessos a plataforma é menor e apresenta diariamente uma queda constantemente, ou seja, os alunos não estão acessando a plataforma logo não estressando a mesma.

O projeto já está ativo por mais de um ano e durante esse período de utilização do memcached seus respectivos servidores foram reiniciados algumas vezes que por sua vez teve seus dados perdidos já que o serviço é volátil, ou seja, não persiste os dados.

Analizando a figura 03, no dashboard do memcached é notável que na sessão network stats foi escrito mais de 790 GB e lido 1 TB de dados. Os indicadores de rate em get stats e set stats também mostram que foram feitos mais de 1300 requests por segundo para a leitura e mais de 4500 requests por segundo para a escrita das informações. Outro indicador em cluster stats mostra que foram estabelecidas mais de 1500 conexões simultâneas entre os servidores de aplicação e o memcached.

Na mesma figura 03 vemos em cache size stats que os servidores ao todo disponibilizam um cache de 12GB, sendo 3 GB cada, e de acordo com a captura está sendo utilizado 7.3GB, no entanto apesar do grande armazenamento, 7.1 GB está sendo classificada como desperdiçada, ou seja, não utilizada para armazenar uma informação. O tuning da alocação de memória, sendo eles o slab, page e chunk, não era um dos objetivos do projeto.

O indicador de miss em get stats nos mostra que apenas 7.6% de todas as requisições para obter um determinado valor cacheado não é encontrado, equivalente a 8.433.817 de requisições. Em contrapartida o indicador de hit já está em 103.227.000 que representa 92.4%, este por sua vez é o indicador de que o cache está funcionando, toda vez que é requisitado um valor ao servidor ele é encontrado.

A partir da análise dos relatórios gerados pela ferramenta, o atendimento do suporte e da equipe de gerenciamento do datacenter, notamos que a engine do PHP gerenciou a sessão com maior eficácia e eficiência, reduzindo seus tempos de resposta, o que levou a um aumento da estabilidade da plataforma em períodos de sobrecarga e que podemos considerá-la uma aplicação mais robusta.

A configuração do PHP e a implementação do código no front-end evitaram a sobrecarga do back-end quando realizado o cache dos requests, logo as melhorias no ambiente ocorriam à medida que o tempo avançava e as operações aconteciam.

### CONSIDERAÇÕES FINAIS

Em comparação as marcas iniciais do ambiente de 8000 máximo de conexões e a plataforma agora suporta mais de 15000 conexões.

Houve melhora de 58,3% nos tempos de resposta, reduzindo de uma média de 6 segundos para 3,5 segundos.

A aplicação em um todo passou a suportar mais de 4000 usuários simultâneos, o que antes recebia aproximadamente 2000 usuários, comprovando assim uma evolução na estabilidade maior que 200% em comparação ao modelo anterior.