Um Mecanismo de Auto Elasticidade com base no Tempo de Resposta para Ambientes de Computação em Nuvem baseados em Containers

Marcelo Cerqueira de Abranches (CGU/UnB)
Priscila Solis (UnB)

Introdução

- Objetivos do trabalho:
  - Propor para a CGU um ambiente elástico de computação;
  - Evitar dimensionamento de recursos inadequados;
  - Escalar recursos de acordo com a demanda.
- Algoritmo de auto elasticidade (PAS):
  - Baseado em containers
  - Alocação eficiente de recursos de acordo com a demanda;

Trabalhos relacionados

- [Gong et al. 2010]:
  - Algoritmo PRESS: Predição de carga de CPU (FFT ou cadeia de Markov de estados finitos)
- [Poddar et al. 2015]
  - Haven: Algoritmo reativo a variações de CPU/Memória
- [Google 2016b]
  - HPA: Algoritmo reativo a variações de CPU

Trabalhos relacionados

- Diferenças:
  - PRESS:
    - PRESS: Predição de carga x PAS: Reação a variações no tempo de resposta de uma aplicação;;
    - PRESS: Escalabilidade vertical (ajustes de cpu) x PAS: Horizontal (provisão de novas instâncias)
    - PRESS: VMs x PAS: Conteiners
  - Haven

Conteiners

- Diferente de virtualização;
  - Conteiners: Instâncias de processamento isoladas gerenciados por um sistema operacional;
  - Máquinas Virtuais: sistemas operacionais isolados gerenciados por um Hypervisor;
- Vantagens:
  - Imagens pequenas (binários dos processos, arquivos de configuraçaõ etc);
  - Rápido de instanciar;

Balanceadores de carga

- Distribui requisições entre servidores de forma transparente para o usuário;
- Abordagem para prover:
    - Melhor desempenho, escalabilidade e alta disponibilidade;

Controladores PID

- Algoritmo de controle;
- Permite controlar o valor de uma variável;
- É estabelecido um “setpoint” para a variável;
- Em cada iteração é calculado o erro:
    - erro = setpoint - valor atual da variável
- Um atuador é alimentado e acionado de modo a tomar uma ação que leve o erro para 0.

Solução Proposta

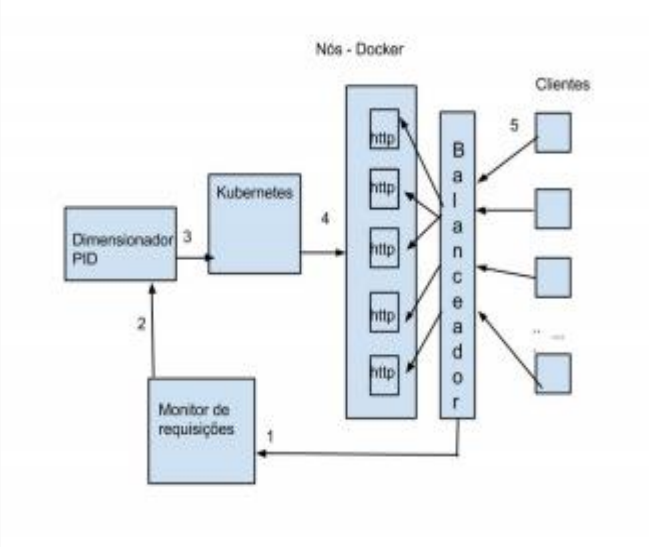
- Ambiente elástico de nuvem, auto escalável e com balanceamento de carga, para hospedagem de sistemas Web.
- Cluster baseado em conteiners;
- Algoritmo de alto elasticidade (PAS):
    - Reaje a variações no tempo de resposta médio de um sistema de modo a mantê-lo dentro de um limite;

Ferramentas utilizadas

- Haproxy
- Flume
- Spark
- Redis
- Dimensionador PAS (python)
- Kubernetes

Arquitetura da solução

- Configura-se no dimensionador PID (PAS) o tempo médio de resposta desejado para aplicação (“setpoint”);
- Monitor de requisições alimenta o dimensionador com o tempo médio atual das requisições;
- Dimensionador informa ao atuador (Kubernetes) o número de containers necessário para atingir o “setpoint”.



Avaliação experimental - Ambiente

- Cluster Kubernetes v1.1.2, instalado no sistema operacional CoreOS 899.6.0, virtualizado em VMWare ESXi 5.5.0
  - 1 nó master (4 vCPUs, 6 GB de RAM);,
  - 1 nó etcd (4 vCPUs, 6 GB de RAM) ;
  - 4 nós workers (4 vCPUs, 6 GB de RAM)
- VMs Ubuntu 14.04.3 LTS, virtualizado em VMWare ESXi 5.5.0
  - 1 nó Haproxy 1.5.4 (4 vCPUs, 4G GB de RAM)

Avaliação experimental - Containers

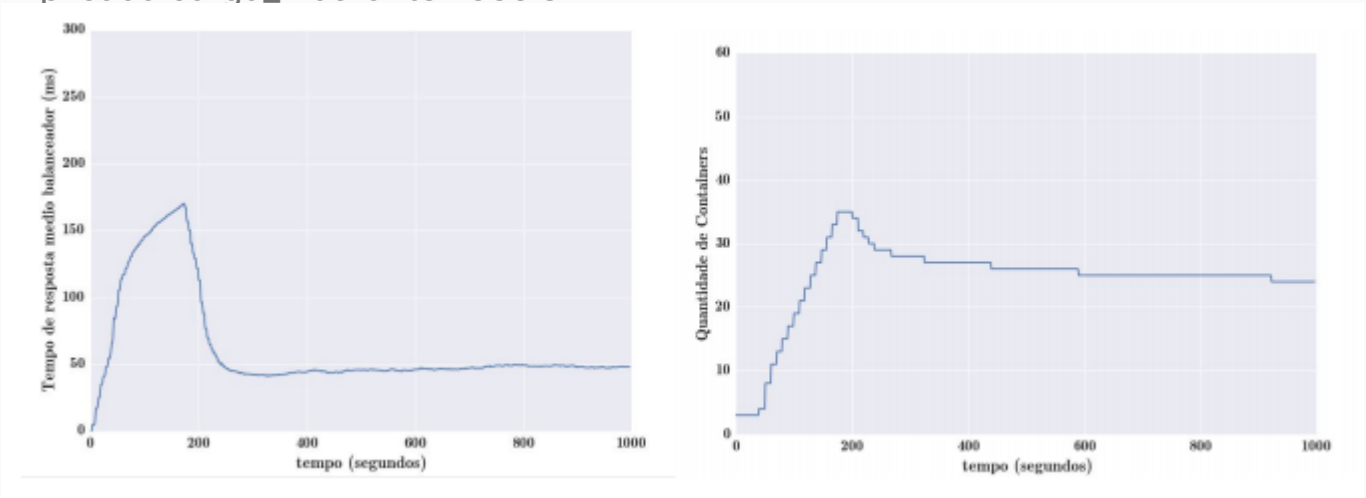
1. Conteiners rodando cherrypy 5.1.0, configurado com um link que gera um array aleatório de 1000 a 10000 elementos em cada requisição;
2. Conteiners limitados a 24 milicores de CPU e 18 MB de memória RAM;

Avaliação experimental - Carga de trabalho

- Gerada com a ferramenta *ab (apache bench)*:
- Perfil de requisições gerado a partir de perfil de requisições do portal da transparência:
  - Série autossimilar (H=0.87, caracterizada com o método Kettani-Gubner),
- Teste executado em diferentes escalas de intensidade
  - 1x (carga_1),
  - 1.5x (carga_1.5)

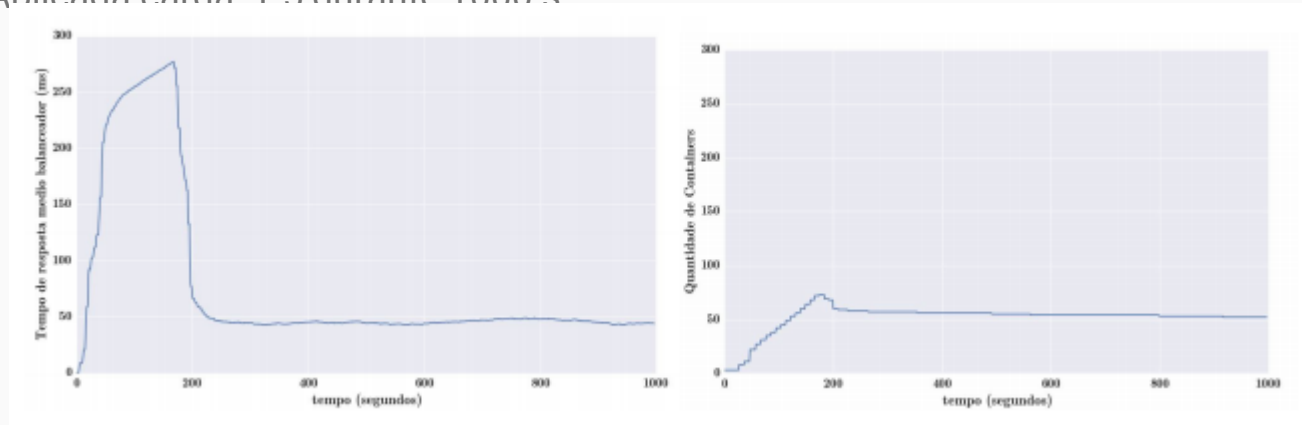
Avaliação experimental - Cenário 1

- O limiar de tempo de resposta no balanceador (setpoint) foi estabelecido em 50 ms;
- Aplicada carga_1 durante 1000 s



Avaliação experimental - Cenário 1

- O limiar de tempo de resposta no balanceador (setpoint) foi estabelecido em 50 ms;
- Aplicada carga 1.5 durante 1000 s;



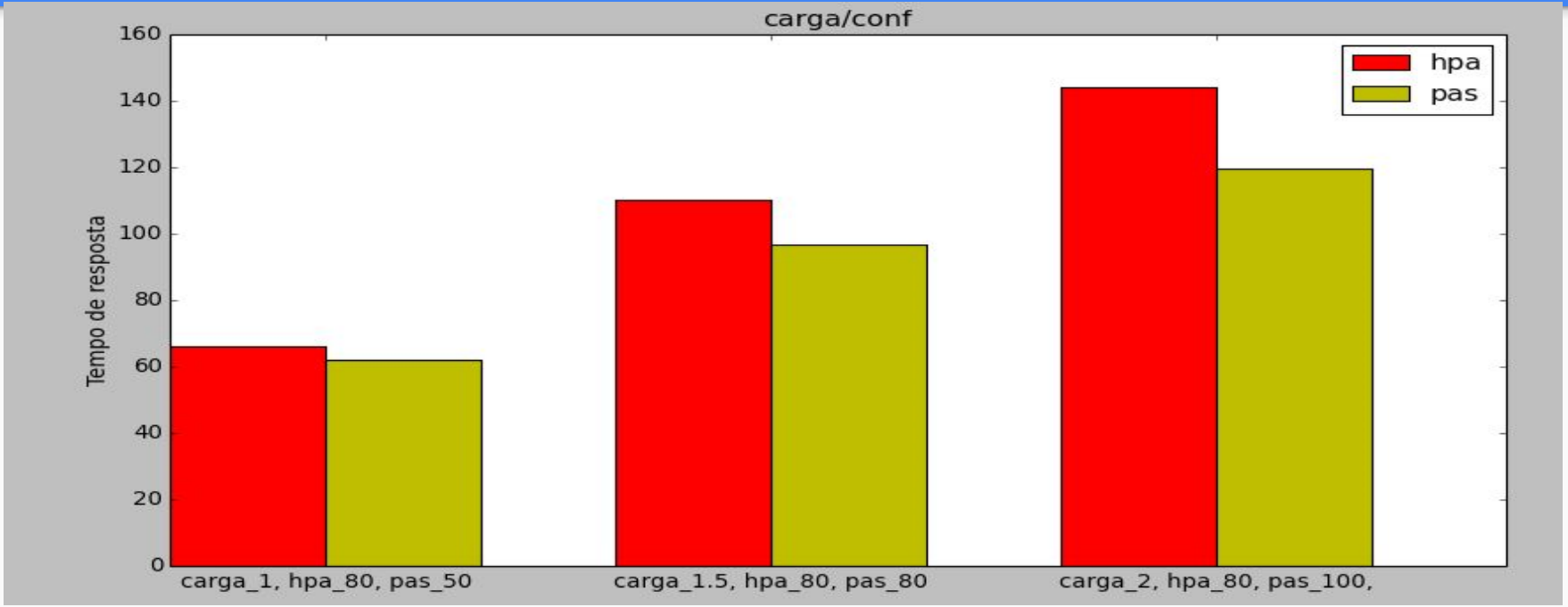
Avaliação experimental - Cenário 2

- Compara a proposta deste trabalho (PAS) com o HPA
- Configurado HPA para escalar com 80 % de carga de CPU (HPA_80):
  - Exposto às cargas carga_1, carga_1.5 e carga_2 durante 1000 segundos.
  - Avaliado tempo médio de resposta na camada de aplicação (usuário);
  - PAS configurado para entregar tempo médios de resposta similares ao HPA para cada carga;
  - Resultados comparados verificando a quantidade média de containers durante os testes e os tempos de resposta obtidos na camada de aplicação dos clientes;

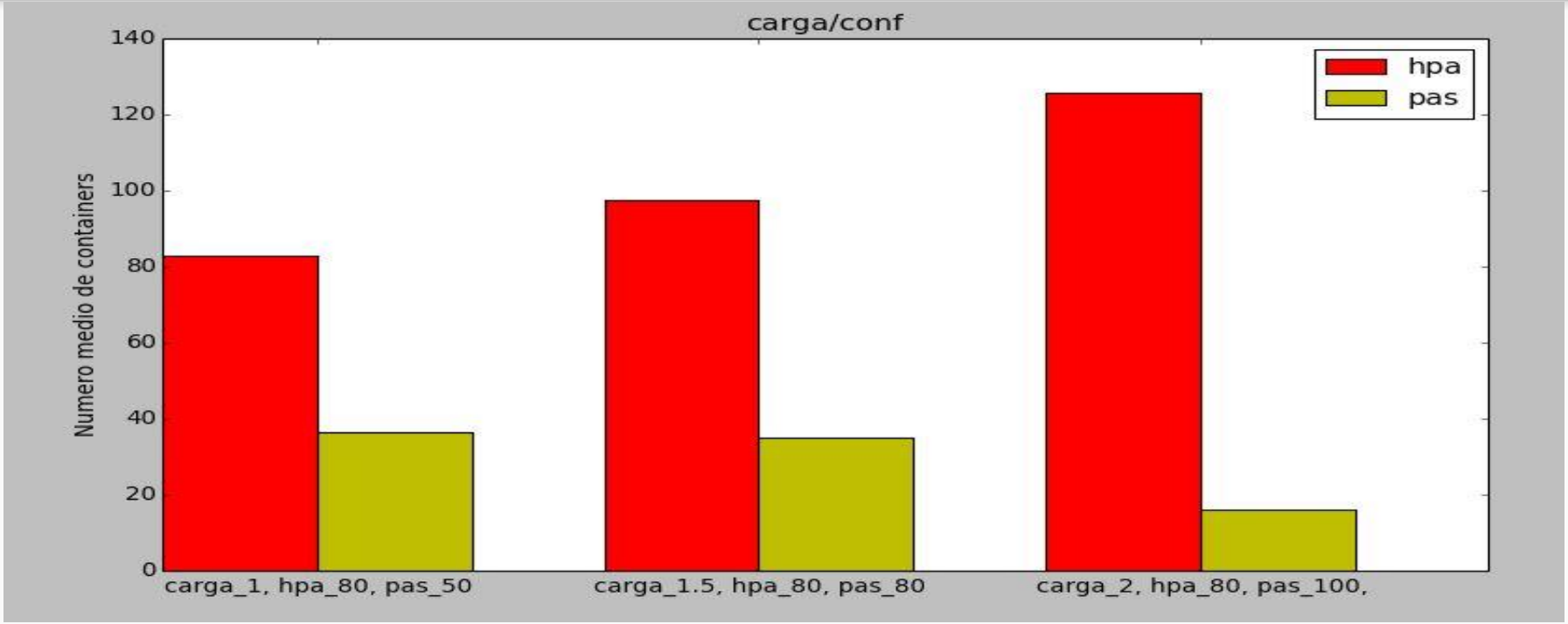
Avaliação experimenta - Cenário 2

- Para carga_1:
  - HPA_80: tempo médio de resposta 64 ms
    - PAS_50: setpoint configurado em 50 ms
- Para carga_1.5:
  - HPA_80: tempo médio de resposta 105 ms
    - PAS_80: setpoint configurado em 80 ms
- Para carga_2:

Avaliação experimental - Cenário 2



Avaliação experimental - Cenário 2



Avaliação experimental - Cenário 2

- Para tempos médios de resposta similares:
  - Carga_1: PAS_50 alocou 44.02 % do que foi alocado pelo HPA_80
  - Carga_1.5: PAS_80 alocou 36.07 % do que foi alocado pelo HPA_80
  - Carga_2.0: PAS_100 alocou 12.72 % do que foi alocado pelo HPA_80

Trabalhos Futuros

- Avaliar resultados experimentais em um maior número de cenários;
  - Cargas de trabalho reais;
  - Ambientes de produção;

Conclusão

- Resultados demonstram potencial para prover auto elasticidade a um ambiente de computação em nuvem baseado em conteiners;
- Comparação com a ferramenta de auto escalabilidade nativa do Kubernetes, mostra uma maior eficiência da solução proposta para alocação da quantidade de conteiners para tempos de resposta similares na execução de requisições para aplicações Web;
- Uso de ferramentas como Spark e Flume possibilita que a arquitetura seja escalável, podendo vir a ser utilizada por sites com servidores com grande número de acessos;

Obrigado !!!

Dúvidas???