Projeto 1 - Servidor de Consultas Linux

Cesar Marcondes - DC/UFSCar

June 15, 2015

1 Objetivos Didáticos do Projeto

- Entender como funciona a interação entre cliente e servidor via CGI
- Entender como funciona a interface de programação de sockets em python
- Entender como criar aplicativo servidor multi-threaded

2 Descrição da Atividade

A aplicação que será desenvolvida pelo grupo, irá permitir a um usuário realizar uma busca de resultados de comandos de linha, a partir de um conjunto de máquinas Linux, através de uma interface web. Especificamente, a aplicação começa apresentando ao usuário uma página web. Nessa página, o usuário poderá selecionar K máquinas de uma lista, e para cada máquina, selecionar um ou mais dos seguintes comandos: ps, df, finger e uptime. Uma vez que essa interface web (em python) receber estas instruções do browser do usuário, um aplicativo backend, também em python, irá se conectar (sequencialmente, ou em paralelo) a um conjunto de "daemons" rodando em cada uma das máquinas da lista. O programa backend então passará os comandos que precisam ser executados às respectivas máquinas remotas. Os "daemons" receberão o comando do programa backend e executarão localmente o comando correspondente. Eles então redirecionarão a saída desses comandos, e o backend juntará todas as respostas para criar uma página web de resultados.

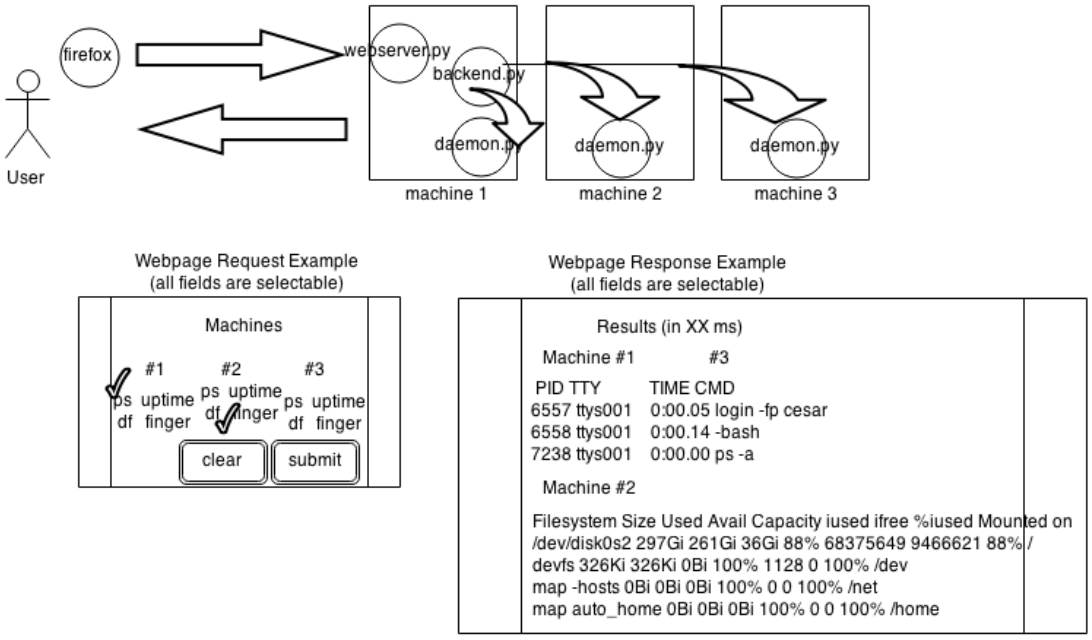

Figure 1: Overall architecture

Para finalizar esse projeto é preciso que o seu grupo escreva os seguintes programas:

1. webserver.py (um programa CGI acoplado a um servidor web apache) ou standalone,
2. página HTML em formato de formulário com os dados da requisição (servida pelo programa webserver ou pelo servidor web apache + CGI),
3. backend.py (o programa que gerencia o backend, ele pode ser acoplado ao webserver.py, ou separado se preferir),
4. o aplicativo daemon.py, executa comandos localmente, processa a saída e é replicado em várias máquinas.

Um requisito desse projeto é que seu "daemon" deve funcionar com o backend de qualquer outro grupo da disciplina. Para alcançar esse objetivo, será preciso padronizar a sintaxe de cabeçalhos do protocolo da camada de aplicação. Desse modo, o formato de cabeçalho da camada de aplicação comunicando com o "daemon" será formada pelos seguintes campos de cabeçalho, em formato ASCII, em uma única STRING onde os componentes são delimitados por espaço:

1. campo descrevendo o tipo da mensagem: "REQUEST" or "RESPONSE"
2. campo descrevendo o programa sendo executado: somente os comandos "ps", "df", "finger" and "uptime" devem ser suportados. Desse modo, o nosso padrão será o seguinte: número 1 para "ps", número 2 para "df", número 3 para "finger", e número 4 para "uptime".
3. campo com argumentos opcionais do comando correspondente (somente na mensagem de REQUEST), por exemplo, ps "-ef". Adicionalmente, o seu "daemon" deve fazer uma checagem prévia destas opções antes de executá-las, garantindo que parametros maliciosos como "|", ";", e ">" não sejam executados.

3 Ambiente de Testes e Submissão Contínua

Serão criadas contas Linux para cada grupo em três máquinas do laboratório ASGARD. Cada grupo poderá colocar executando os seus programas no mesmo servidor apache, criando uma pasta /var/www/html/grupoX (onde X é o número do grupo) e /usr/local/apache2/cgi-bin/grupoX. Cada grupo usar uma PORTA de comunicao em socket usando o nmero 9000 + X, por exemplo porta TCP 9001 está reservada para o grupo 1. Cada grupo criará um repositório no github (https://github.com/) e enviará o link ao professor, e colocará o mesmo como colaborador (github user: cmarcond). Esse passo é importante, pois o professor dará a nota individual para cada aluno, baseado em sua contribuição no respectivo projeto do grupo. Portanto, é importante que o grupo divida igualmente a carga de trabalho, e cada aluno tenha participação equivalente em alguma parte do projeto.

4 Totalização da Nota

- (25%) O programa "daemon.py" aceita pelo menos uma conexão socket por vez, faz o parsing do comando remoto enviado, executa o comando requisitado e retorna a saída.
- (25%) O "daemon.py" é multithreaded e pode aceitar um número arbitrário (porém dentro dos limites do normal) de conexões simultâneas.
- (10%) O programa webserver.py ou apache + CGI pode receber uma requisição web no servidor local e processar os resultados do daemon em formato de página web.
- (30%) O programa webserver.py ou apache + CGI pode conectar sequencialmente com cada máquina remota selecionada, e comunicar com o daemon correspondente enviando a requisição, depois coletar os resultados e mostrar os mesmos com uma página web formatada em formato HTML com os resultados das saídas dos comandos.
- (10%) A organização da estrutura dos programas, comentários, etc.

5 Referencias Bibliográficas

https://docs.python.org/2/howto/sockets.html