Trabalho prático de Sistemas Operativos

Cloud shell

Grupo de Sistemas Distribuídos
Universidade do Minho

Maio de 2015

Informações gerais

- Cada grupo deve ser constituído por até três elementos. Grupos de dimensão inferior a três terão de ser justificados e autorizados pelos docentes.
- O trabalho deve ser entregue até 2015-05-29 às 23:59. Por indicação da direcção de curso a avaliação dos trabalhos deverá ocorrer na primeira semana de Junho, servindo assim também como preparação para o teste do dia 12.
- A apresentação decorrerá na semana seguinte, após inscrição nos horários a indicar (que se procurará coincidirem com as aulas práticas e/ou teóricas.) Se, com o apoio da direcção de curso, se conseguir identificar um horário alternativo, a data de entrega poderá ser revista.

Resumo

A ideia subjacente a este trabalho prático é simular um sistema partilhado de *pay-per-use* como os que tipicamente são usados em ambientes "cloud": o fornecedor do serviço contabiliza recursos consumidos (cpu, espaço em disco, tráfego de rede, entre outros) e o utilizador tem de pagar pela utilização desses recursos.

Em termos práticos, este trabalho consiste na implementação em C de uma CloudShell com a qual o utilizador interage, em vez de interagir com a bash do sistema. Esta nova shell vai lançando os processos indicados pelo utilizador e simultaneamente faz a monitorização dos recursos gastos e vai descontando no saldo do respectivo utilizador. Quando o saldo se esgota, a shell força a terminação dos processos por si lançados e ainda em execução.

Note que interação com os utilizadores é feita exclusivamente através do teclado e das linhas que os comandos vão imprimindo no écran[1].

Como deve calcular, este serviço de "cloud" deve poder ser usado para executar aplicações arbitrárias, já existentes ou a desenvolver. Não pode por conseguinte instrumentar o código dessas aplicações para periodicamente ir determinando os recursos consumidos. Terá de ser a CloudShell a descobrir que recursos os seus "filhos" estão a gastar (editor de texto, compilador ou mesmo um serviço de longa duração). Em Linux esta monitorização poderá ser feita inspecionando o conteúdo da pasta virtual /proc. De igual modo, a "cloud" poderá ser partilhada por muitos utilizadores, pelo que será necessário fazer a contabilização de recursos por utilizador registado na "cloud". Deverá ser evitada uma solução baseada no comando `iostat`, que só fornece estatísticas globais. No entanto, uma boa alternativa à inspecção da `/proc` poderá ser a utilização do comando pidstat, obrigando nesse caso à implementação de um `popen` (que também foi um dos exercícios propostos nos guiões).

Quer a robustez quer o desempenho são aspectos fundamentais deste trabalho. Para evitar o tempo de espera dos acessos a disco poderá ser querer manter em memória central a informação de contabilização mas corre o risco de um eventual crash da CloudShell fazer "esquecer" os recursos consumidos[2], algo que não pode acontecer. Sugere-se a utilização de uma arquitectura um pouco mais complexa em que cada CloudShell vai enviando periodicamente a informação a um servidor de contabilização (com o qual comunica por um ou mais pipelines) e esse sim, responde de imediato com um OK ou KO caso o utilizador tenha ou não saldo positivo e de seguida actualiza a informação num ficheiro em disco, para não se perder caso ele mesmo sofra um crash. Não se esqueça que um serviço destes não deve terminar e tem de recuperar rapidamente em caso de falha.

Descrição

Para além da elaboração de um relatório sucinto que justifique as decisões tomadas, as dificuldades encontradas e os resultados obtidos, o trabalho consiste no desenvolvimento em C de:

- Uma CloudShell que periodicamente monitoriza os recursos consumidos pelos processos por si lançados em nome de um determinado utilizador e comunica com um serviço de contabilização para saber se tem de abortar esses processos por se ter esgotado o saldo do utilizador.

[1] Algo semelhante a fazer `telnet` ou `ssh` para uma máquina e fechar as restantes aplicações e janelas.

[2] Como acontecia há muitos anos com algumas operadoras de telemóveis, que só cobravam as chamadas quando se desligava a chamada...

- Um serviço de contabilização que responde rapidamente à pergunta anterior e simultaneamente trata de registar em disco os recursos consumidos.

É necessário incluir um script de configuração e arranque do serviços para rápida instalação durante a discussão do trabalho, por exemplo numa máquina disponibilizada pelos docentes, assim como um script que teste todas as funcionalidades implementadas[3].

Fica ao critério de cada grupo apresentar uma solução mais sofisticada, por exemplo que seja capaz de lidar com múltiplos utilizadores (com autenticação), de reabastecer o crédito, de contabilizar mais recursos, de consultar o saldo num determinado momento, etc.

Todos os ficheiros (código, makefile, script, relatório em pdf) devem ser comprimidos num único ficheiro ZIP e submetidos via blackboard até à hora indicada, sendo penalizadas as submissões tardias. Para esse efeito, em breve serão abertos no blackboard muitos (100?) grupos com capacidade de 3 elementos, devendo cada grupo de alunos escolher um grupo blackboard vazio e preencher o nome dos três elementos[4]. Uma vez conhecido o número de trabalhos submetidos, serão criados "slots" para permitir a escolha do horário de apresentação.

Notas finais

Este trabalho vale apenas 1/3 da classificação final. É relativamente simples para quem foi resolvendo os exercícios propostos nos guiões e serve essencialmente para averiguar o grau de preparação para o teste e/ou exame de recurso, onde não vai existir o auxílio externo nem dos restantes elementos do grupo.

Todos os elementos do grupo terão de estar à vontade para responder às questões durante a discussão pública, justificando as opções tomadas e mostrando conhecimento do código desenvolvido, dos testes efectuados, da configuração do sistema, etc. Pretende-se que cada um mostre o que aprendeu e que está bem preparado para a componente prática do teste/exame. Assegure-se que percebeu bem que problema resolve cada solução que implementou: que problema resolve este fork aqui, este pipeline, este wait, estes sinais....

Não tente copiar ideias de outros grupos e evite a todo o custo usar funções ou comandos que lhe tiram "muito trabalho", incluindo `system` e `popen`!

[3] O output deste teste deverá ser incluído no relatório.

[4] Caso vejam um grupo parcialmente preenchido, farão o favor de escolher um grupo vazio!