1º. PROJETO DE PESQUISA

Explorando tecnologias orientadas a serviços

Implementando um Repositório de Tarefas

O objetivo deste projeto é construir uma aplicação que permita que processos executando em diferentes máquinas possam realizar partes de uma tarefa em paralelo. A idéia geral é que um processo mestre coloque sub-tarefas de uma computação em um repositório de tarefas (*task bag*) e processos escravos selecionem tarefas do repositório para executá-las, retornando os resultados para o mesmo repositório. O mestre, então, coleta os resultados e os combina para produzir o resultado final.

Neste projeto, você terá que implementar o repositório de tarefas como um *web service* e usá-lo como base para realizar a computação paralela em diferentes máquinas. O processo-mestre e os processos-escravos serão clientes do repositório de tarefas.

O repositório de tarefas

O repositório de tarefas é um *web service* cuja funcionalidade é armazenar *Pairs*. Cada *pair* consiste de duas partes - uma chave (*key*) e um valor (*value*). O *valor* contém valores associados à execução da tarefa, enquanto a *chave* é qualquer coisa que possa ser usada para referenciar o *pair* (um número ou nome, por exemplo). Um *pair* é usado pelo *master* para descrever tarefas e pelos escravos para descrever resultados. Clientes podem *(i)* adicionar *pairs* ao repositório, *(ii)* removê-los, e *(iii)* obtê-los. Para acessar um *pair*, o cliente precisa especificar uma *chave*.

A interface do repositório está descrita abaixo:

pairIn(key, value)	Faz com que um *pair (key, value)* seja adicionado ao repositório. O processo cliente continua imediatamente.
pairOut(key) -> value	Faz com que um pair que combine com a chave key seja removido do repositório; o value é retornado e o processo cliente continua. Se nenhum pair estiver disponível, uma exceção é retornada e o cliente deve esperar por um tempo aleatório até fazer uma próxima tentativa (método *poll*). Se vários *pairs* com a mesma chave estiverem disponíveis, qualquer um deles é removido/retornado.
readPair(key) -> value	O mesmo que *pairIn(key)*, exceto que o *pair* permanece no repositório.

A aplicação

Nesse projeto, o aluno deve escolher uma aplicação que necessite de computação intensiva e que possa ser facilmente dividida em um número de sub-tarefas idênticas e independentes (aplicação *bag-of-task*). Alguns exemplos são *(a)* o cálculo de números primos, *(b)* a multiplicação de matrizes, *(c)* a compilação paralela de módulos de um programa etc.

Programação Paralela com o Repositório de Tarefas

Um processo mestre provê um conjunto de tarefas a ser feito por uma coleção de processos escravos idênticos. Cada escravo é capaz de realizar qualquer uma das sub-tarefas de um computação particular. No caso mais simples, há apenas uma sub-tarefa. Um escravo repetidamente obtém uma tarefa, executa e, então, coloca o resultado no repositório. Os resultados são coletados pelo mestre. O programa executa da mesma maneira se há 1, 10 ou 1000 escravos.

Para exemplificar, considera-se aqui uma aplicação que multiplica duas matrizes A e B. Neste caso, um processo mestre cria as tarefas de multiplicação e coleta os resultados gerados por um ou mais escravos. Cada escravo repetidamente pega um elemento para calcular e coloca o resultado de volta no repositório.

Como os escravos sabem qual é a próxima tarefa

Em muitas computações, existe, de fato, uma coleção de tarefas. Cada escravo repetidamente pega uma delas. Antes de um escravo começar a execução de uma tarefa, entretanto, ele precisa saber qual é a próxima tarefa a realizar. Um *pair* com a chave "Next Task" pode ser usado para este propósito. O mestre coloca, então, a primeira tarefa no repositório e cada escravo, por sua vez, remove o *pair* "Next Task", incrementa seu valor e o adiciona de volta no repositório. O número de tarefas é uma constante da aplicação.

No caso da multiplicação das matrizes A e B, o número de linhas da matriz A e o número de colunas da matriz B devem ser conhecidos. O *pair* "Next Task" deve conter a posição do elemento a ser calculado (i,j), sendo continuamente atualizado por cada escravo.

Os resultados dos escravos

Escravos executam tarefas semelhantes e podem retornar valores com chaves idênticas para o repositório de tarefas. O repositório, portanto, tem que ser implementado de maneira que muitos *pairs* com a mesma chave possam ser manipulados ao mesmo tempo.

Mas nesse exemplo da multiplicação de matrizes, cada escravo precisa usar uma chave diferente no resultado gerado, indicando a linha e a coluna do elemento calculado, de modo que o mestre possa montar a matriz resultante. Quando um escravo calcula um elemento, ele precisa especificar a chave conforme descrito abaixo:

```
String key= "Element" + row + column;

pairIn(key, <valor calculado>);
```

O mestre coleta os resultados da forma abaixo:

```
valor = pairOut("Element" + row + column);
```

Fornecendo Dados para os Escravos

Em algumas computações, os escravos precisam de dados para realizar suas tarefas. No caso da multiplicação de matrizes, um escravo precisará da linha i da matriz A e da coluna j da matriz B de modo a multiplicá-las. Estes dados são colocados no repositório pelo mestre e podem ser acessados

pelos escravos que precisarem. O mestre pode colocar linhas da matriz A e colunas da matriz B como segue:

```
pairIn("A1", <primeira linha de A>);
pairIn("A2", <segunda linha de A>);
...
pairIn("B1", <primeira coluna de B>);
pairIn("B2", <segunda coluna de B>)
```

Questões de Ordem

- O trabalho deve ser feito em duplas, e postado no Moodle até 10h do dia 22/04/2018
- Cada dupla deve desenvolver seu repositório de tarefas utilizando interfaces/arquiteturas *web service*[1] e REST[2]. Sugere-se o estudo dessas tecnologias antes de iniciar o projeto.
- A escolha da aplicação a ser paralelizada é livre, podendo ser, inclusive, a multiplicação de matrizes aqui descrita.
- Os alunos devem apresentar a aplicação funcionando no laboratório.
- Os alunos devem entregar um relatório por dupla (com destaque para contribuições individuais), contendo:
    - os objetivos do trabalho
    - uma descrição dos elementos que compõem o repositório de tarefas
    - uma descrição de como sua tarefa foi realizada pelo mestre e escravos
    - uma descrição resumida da tecnologia utilizada
    - uma análise do grau de tolerância a falhas de sua aplicação (você pode descrever possíveis problemas e indicar soluções)
    - alternativas para a sincronização entre os clientes (alternativas para o método *poll* inicialmente proposto quando da chamada ao serviço *pairOut*)
    - análise da concorrência existente (indique possíveis necessidades de controlar tal concorrência, se for o caso)
    - uma análise de desempenho simples, objetivando ressaltar o ganho de *performance* mediante o paralelismo viabilizado pela usa aplicação
    - sua opinião pessoal sobre o projeto
    - referências utilizadas

A nota é individual e o valor máximo será dado ao aluno que demonstrar conhecimento e aprendizado satisfatório com o projeto (domínio das tecnologias *web service* e REST). Outros requisitos como cumprimento das datas, trabalho coordenado em grupo e inserção de novas funcionalidades no projeto também serão consideradas para emissão da nota final.

[1] https://pt.wikipedia.org/wiki/Web_service
[2] https://pt.wikipedia.org/wiki/REST