SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Computer Science

F21DP2

Distributed and Parallel Technologies

Semester 2 2013/14

Date: May 2nd, 2014

Duration: Two Hours

ANSWER THREE QUESTIONS

Answer each question in a separate script book.

1. Matrix-vector multiplication is a special instance of matrix-matrix multiplication, where the first argument is of dimension $m \times n$ and the second argument is of dimension $n \times 1$. Thus, the result is a vector of length m, where the i-th element of the result vector is obtained by performing a dot-product on the i-th row of the first matrix and the vector.

The following Haskell code implements this matrix-vector multiplication:

```
-- inner vector product
dotProd :: [Integer] -> [Integer] -> Integer
u `dotProd` v = sum (zipWith (*) u v)

-- multiplying a matrix with a vector (sequential)
mulMatVec :: [[Integer]] -> [Integer] -> [Integer]
m `mulMatVec` v = [ v' `dotProd` v | v' <- m ]
```

(a) 1. Produce a parallel version of matrix-vector multiplication.
2. Identify the parallelism paradigm you have used.
3. Discuss the parallel performance you might expect from your program. In particular do you expect good speedup, and if so why?

(8)

(b) Suggest a way to improve the performance of the program, without changing the structure of the program. Give the code to implement this improvement and explain why you expect an improvement. (8)

(c) Compare the process of tuning the parallel program in GpH, as covered in the previous question, with the same kind of optimisation performed on a C+MPI implementation of matrix-vector multiplication. In particular, discuss

- How much does the structure of the code change by introducing an optimisation?
- How easy is it to predict the order of evaluation in the program?
- How sensitive is the performance of the program on the architecture the program is running on?
- How high is the risk of introducing a deadlock in the tuning process?

(4)

2. In the context of Warehouse Computing the map/reduce computation model is commonly used.

(a) Summarise (for example in a picture) and discuss one of the implementations of the map/reduce computation model presented in the course. Discuss which parameters of the framework can be tuned in order to improve performance. (6)

(b) The MapReduce model of computation relies on the possibility to decompose a given computation $process$ on a data structure $input$ into two components. What are these components and describe formally how they are related to the overall computation $process\ input$.

Hint: Give only a high level, formal description without grouping the components. (4)

(c) Briefly explain the role of HDFS in Hadoop and why HDFS is a central component of Hadoop. (4)

(d) To compare the deviation of a sequence of observed values os from a sequence of expected values es the *chi-square* statistical function can be used, written χ. Informally, this function computes the sum over the distances over all pairs of values. The smaller this sum, the smaller the deviation. This function can for example be used in cryptography to break a permutation cipher such as the Cesar Cipher.

The formal definition of χ is as follows:

$$\chi\ os\ es\ = \sum_{i=0}^{n-1} \frac{(os_i - es_i)^2}{es_i}$$

Describe a parallel implementation of this function, using a MapReduce structure: first discuss informally how the computation is decomposed, then give a pseudo-code description of the computations that are needed for these components.

(6)

3. Consider the following SaC function:

```
1: double[.] relax( double[.] a, double eps)
2: {
3:    for( i=0; i<50; i++) {
4:       a = with {
5:                ( [1] <= iv < shape(a) -1))
6:                : (a[iv-1] + a[iv] + a[iv+1]) / 3.0;
7:                } : modarray(a);
8:    }
9:    return( a);
10:}
```

(a) Explain informally the functionality of `relax`. Use an example value for the argument `a` to underpin your explanation of the with-construct in lines 4–7. (4)

(b) What is the result of applying `relax` to

1. `[ 1.0, 2.0]`?
2. `[ 42.0]`?
3. `[ :int ]`?
4. `[ [ :int ] ]`?

(4)

(c) Explain why the with-construct can be executed in any order without affecting the overall result. What consequences does this have on the memory requirements of the `relax` function? (2)

(d) Re-write the function `relax` so that it iterates until the absolute difference of all elements between two iterations of the array `a` becomes smaller than the scalar value `eps` that is provided as a second parameter to `relax`. You may use all functions from the standard library. (4)

(e) The SaC compiler can generate code for various different architectures including shared-memory multi-processors (SMPs) and graphics cards (GPUs).

1. Explain for which arguments of `relax` you would expect a good scaling on SMP machines and why.
2. Identify two possible challenges for the compiler when creating code for GPUs from the `relax` example. Try to predict how critical these are for achieving good scaling on GPUs. Use your experiences from hand-writing openCL programs as inspiration.

(6)

4. OpenCL has evolved as one of the major standards for programming heterogeneous systems. Implementations support various forms of accelerators, most notably a wide range of graphic processing units (GPUs).

(a) Explain the memory model of openCL.

1. What kinds of memory exist? (3)
2. How can they be accessed within a kernel? (3)
3. For which of these can the need for synchronisation arise? (2)
4. How does the need for synchronisation arise? (2)
5. Give an example of a kernel fragment that requires such synchronisations. Provide the type declarations for all variables involved. You may assume adequate memory allocations and you may also assume possibly required restrictions within the index-range of the kernel invocation. (2)

(b) Consider the following openCL kernel:

```
__kernel void square(
 __global float* input_a,
 __global float* input_b,
 __global float* output,
 const unsigned int count)
{
 int i = get_global_id(0);
 output[i] = input_a[i] * input_b[i];
}
```

Assume the length of the vectors `input_a`, `input_b`, and `output` to be `count = 1024`.

1. What does this kernel do? (2)
2. Provide a suitable choice for the thread-mapping by proposing an index range and a work-group size that would lead to the desired result. (2)
3. What needs to be done in order to be able to execute the kernel successfully with vectors of a-priori-unknown length? You may assume a fixed work-group size. Describe how the overall index-range needs to be chosen and provide a suitable kernel definition. (4)

END OF PAPER