Master Degree Program in Computer Science and Networking

High Performance Computing

2014-15

Homework 6

All the answers must be properly and clearly explained.

Exercise 1 (*Graph Analysis*):
Consider a computation Σ with the OR-graph structure in the figure. Module P produces a stream of elements with interdeparture time T_p. For i = 0 ... n-1, let T_i be the *ideal* service time of Q_i, and p_i the probability that an element is sent from P to Q_i, with $\sum_{i=0}^{n-1} p_i = 1$. Derive formally the relation among T_p, $\{T_i\}$, $\{p_i\}$ such that the *effective bandwidth* of Σ is equal to the *ideal bandwidth*. Provided that such relation among T_p, $\{T_i\}$, $\{p_i\}$ holds, evaluate the *ideal service time*, *effective service time*, and *relative efficiency* of Σ, P, Q_0, …, Q_{n-1}.

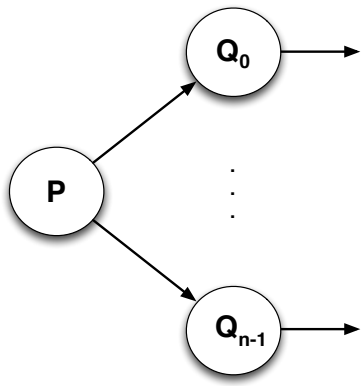


Figura 1: Computation graph of the first exercise.

Exercise 2 (*Parallelization of bottlenecks*):
Consider a parallel computation Σ having the following OR-graph structure:

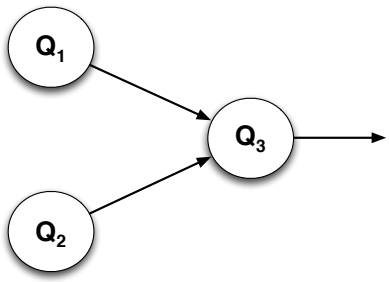
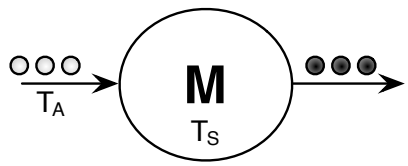


Figura 2: Computation graph of the second exercise.

Module Q_1 (Q_2) encapsulates an integer array $A_1[M]$ (an integer array $A_2[M]$) with M =10^4, and generates a stream, of length M, whose elements are the elements of A_1 (of A_2). For each stream element, Q_3 executes an integer function F with processing time equal to *50t*, where *t* is a conventional time unit whose value is much greater that the instruction service time of the processing nodes. Determine:

a. a parallel version of Q_3 such that Σ achieves the best processing bandwidth as possible on a parallel architecture having $N = 128$ nodes with communication processor, T_{setup}= 0.9t and T_{trasm}= 0.1t.
b. evaluate the *service time* and the *relative efficiency* of Σ of Q_1 , of Q_2 and of Q_3 , and the *completion time* of Σ.

Exercise 3 (*Conceptual question*):
Let us consider a sequential processing module *M* operating on an input stream of integers $x_1\ x_2, \dots ,\ x_m$ with inter-arrival time T_A. The computation of *M* is as follows:




```
int s = 0;
while (true) do
  <receive integer x from input_stream>;
  (y, s) ← F(x, s);
  <send y onto the output_stream>;
EndWhile
```

Figura 3: Pseudo-code of module M.

In the most formal way as possible discuss if this computation can be parallelized according to the *farm* paradigm.