Master Degree Program in Computer Science and Networking

High Performance Computing

1st midterm - November 7, 2012

The answers can be written in English or in Italian. Please, present the work in a legible and readable form. All the answers must be properly and clearly explained.

Question 1

a) For the following sequential program:

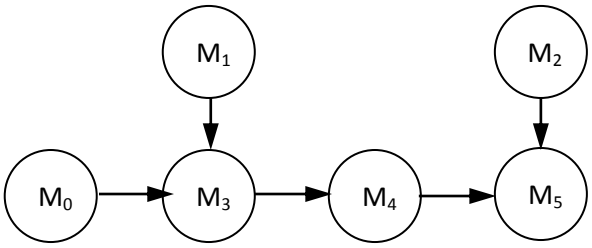
```
int A[M]; int B[M][M]; int x; bool y;
{ ∀ i = 0 .. M − 1:
        ∀ j = 0 .. M − 1:
                A[j] = F(A[j], B[i][j]);
  y = true;
  ∀ i = 0 .. M − 1:
        y = y ∧ if (A[i] > x)
}
```

where *y* is the returned result, determine a Virtual Processors parallelization.

b) Consider a module that processes the same computation of *a)* on streams: *A* is received from the input stream, *B* and *x* are constants, and *y* is sent onto the output stream. Explain if a farm parallelization is feasible and, if it is feasible, under which conditions the ideal bandwidth is achieved.

Question 2

In the following OR-graph computation Σ, each of the modules M_0, M_1, M_2 generates a stream of length *m*.



Let T_i denote the calculation time of M_i ($i = 0, \ldots, 5$).

For each *i*, the communication latency for zero-copy communications is less than T_i.

All the processing nodes, with clock cycle τ, have a communication processor.

a) Determine the relation that must exist among the values of $\{T_i\}$ in order that the efficiency of Σ is equal to one. If this relation holds, determine bandwidth and completion time of Σ, and bandwidth and efficiency of the single modules.

b) M_5 encapsulates an initialized integer variable *s*, receives integer arrays A[M] from M_2 or M_4, and computes integer values *y*. *For each A*, M_5 performs the following computation:

```
{ ∀ i = 0 .. M − 1:
        s = s + A[i];
  y = F(s)
}
```

Describe process M_5 in LC. Explain the structure of M_5 executable file and its initialization.

c) Let $10^5\tau$ the maximum value of T_5 for which the condition determined in *a)* is satisfied. Let $M = 10^3$. The calculation time of *F* is equal to $10^6\tau$.

Parallelize M_5 in order to reduce or eliminate the bottleneck. With this version of M_5, evaluate bandwidth and efficiency of Σ, and bandwidth and efficiency of M_5 and of its component modules.

Solution

to be integrated with proper explanations

Question 1

a) In the first part of the computation:

$\forall\ i = 0\ ..\ M-1:$

$\quad \forall\ j = 0\ ..\ M-1:$

$\quad\quad A[j] = F(A[j], B[i][j]);$

through loop unrolling we discover that

$A[h] = F(A[h], B[i][h])$

$A[k] = F(A[k], B[i][k])$

are independent for each *h, k.* Thus, there is parallelism on *j* and, for each *j*, sequential behavior on *i*. Also according to the owner computes rule, the Virtual Processors computation is a linear array VP[M], where the generic VP[j] encapsulates *A[j]* and the column *B[*][j]*. The first part of the computation is a data-parallel *map*.

The second part:

$y = true;$

$\forall\ i = 0\ ..\ M-1:$

$\quad y = y \wedge if\ (A[i] > x)$

is equivalent to:

$$y = reduce\ (C, \wedge)$$

where *C[M]* is a boolean array such that $C[i] = if\ (A[i] > x)$ for each *i.* We know how to parallelize a *reduce* computation by a linear array of VPs.

Each VP[j] encapsulates *C[j]* and a copy of *x* too, and computes

$\forall\ i = 0\ ..\ M-1:\ A[j] = F(A[j], B[i][j]);$

$C[j] = if\ (A[j] > x);$

take_part_to_reduce (C[j], $\wedge$);

if (j = M–1) then return y

The parallel *reduce* is tree-structured and the final result is produced by *VP[M–1]*. The tree structure is mapped onto the linear array VP[M] *(add explanations).*

b) The farm parallelization is feasible because the computation is a pure function mapping *(A, B, x)* into *y* (though re-assigned, variable *A* is not an internal state, because it is re-initialized for each new element of the stream). Each worker encapsulates a copy of *B* and *x*, and executes all the given computation sequentially.

The ideal bandwidth is equal to the inverse of the interarrival time T_A. The effective bandwidth is equal to the ideal one on condition that

1. the emitter is not a bottleneck:

$$T_E = T_{send}(M) < T_A$$

2. the number of processing nodes is greater than, or equal to, the optimal parallelism degree:

$$n = \left\lceil \frac{T_{calc}}{T_A} \right\rceil = \left\lceil \frac{M^2\ T_F + L_{seq-reduce}}{T_A} \right\rceil \ (\sim \left\lceil \frac{M^2\ T_F}{T_A} \right\rceil)$$

Question 2

a) According to the assumption on communications, $T_{com} = 0$ for each module, thus the *ideal service time* of M_i is equal to T_i, for $i = 0 \dots 5$.

The efficiency of Σ is equal to one if no module is a bottleneck, that is if (the graph is acyclic):

$$\rho_i = \frac{T_i}{T_{Ai}} \leq 1 \quad \forall i = 0 \dots 5$$

Let us express this condition.

The interdeparture times from M_0 and M_1 are respectively T_0 and T_1. According to the multiple-client theorem, the interarrival time to M_3 is:

$$T_{A3} = \frac{1}{\frac{1}{T_0} + \frac{1}{T_1}}$$

Therefore, it must be:

$$T_3 \leq \frac{1}{\frac{1}{T_0} + \frac{1}{T_1}}$$

so the interdeparture time from M3 is:

$$T_{p3} = T_{A3} = \frac{1}{\frac{1}{T_0} + \frac{1}{T_1}}$$

which is equal to T_{A4}, thus it must be:

$$T_4 \leq \frac{1}{\frac{1}{T_0} + \frac{1}{T_1}}$$

so the interdeparture time from M_4 is:

$$T_{p4} = T_{A4} = \frac{1}{\frac{1}{T_0} + \frac{1}{T_1}}$$

The interarrival time to M_5 is:

$$T_{A5} = \frac{1}{\frac{1}{T_{p2}} + \frac{1}{T_{p4}}} = \frac{1}{\frac{1}{T_0} + \frac{1}{T_1} + \frac{1}{T_2}}$$

Therefore, it must be:

$$T_5 \leq \frac{1}{\frac{1}{T_0} + \frac{1}{T_1} + \frac{1}{T_2}}$$

The ideal bandwidth of Σ (for a stream length $3m$) is the bandwidth for the stream generation:

$$B_{\Sigma - id} = \frac{1}{T_0} + \frac{1}{T_1} + \frac{1}{T_2}$$

The effective bandwidth is equal to the inverse of the interdeparture time from M_5. If the condition on T_5 holds, we have:

$$B_\Sigma = \frac{1}{T_{p5}} = \frac{1}{T_{A5}} = \frac{1}{T_0} + \frac{1}{T_1} + \frac{1}{T_2}$$

thus $\varepsilon_\Sigma = 1$, as required.

The completion time of S is:

$$T_c = 3m\,T_{p5}$$

The effective bandwidth and efficiency of the various modules are:

module	*ideal bandwidth*	*effective bandwidth*	*efficiency*
M_0	$1/T_0$	$1/T_0$	1
M_1	$1/T_1$	$1/T_1$	1
M_2	$1/T_2$	$1/T_2$	1
M_3	$1/T_3$	$1/T_0 + 1/T_1$	ρ_3
M_4	$1/T_4$	$1/T_0 + 1/T_1$	ρ_4
M_5	$1/T_5$	$1/T_0 + 1/T_1 + 1/T_2$	ρ_5

b) Σ is a parallel program:

parallel M0, M1, M2, M3, M4, M5;

where:

```
M5::  int s = initial_value; int y: int A[M]; channel in ch2 (k), ch4 (k); channel out ch5; < definition of F >;
        // asynchrony degree k to be determined with an additional investigation based on interarrival times
        and their distributions //
      while (true) do
          alternative
              { receive (ch2, A) do
                      { for (i = 0; i < M; i++)
                              s = s + A[i];
                        y = F(s);
                      send (ch5, y)
                      }
            or  receive (ch4, A) do
                      { for (i = 0; i < M; i++)
                              s = s + A[i];
                        y = F(s) );
                      send (ch5, y)
                      }
              }
```

The executable file of process M_5 contains the *virtual memory* of M_5 produced at compile time, having the following structure:

- Code of sequential part (instructions for the program-visible part of M_5, including F and calls to run-time procedures *alternative* and *receive*): initialized
- Program-visible data structures:
  - A: *(k+1)*M* words, in order to implement zero-copy communication on channel ch2, ch4 with asynchrony degree *k*: not initialized;
  - s: one word: initialized;
  - y: one word: not initialized.

- Code of run-time support procedures for send, receive, alternative, low-level scheduling (context-switch, process wake-up, termination), interrupt and exception handling: initialized
- Run-time data shared data-structures:
  - Channel descriptors for ch2, ch4, ch5 and other interpreter-visible channels, each one with its own structure and size: initialized (except buffer and PCB_ref fields);
  - PCB and TAB_RIL of M_5: partially initialized, in particular the register image is initialized in PCB, as well as pointers to head and tail pointers of Ready List, and pointer to TAB_RIL;
  - PCBs of M0, M1, M2, M3, M4 and of other interpreter-visible processes: non initialized;
  - Ready List: not initialized.

Moreover, a *configuration file* is associated, containing proper information about shared data structures and process working set, in order to allow the Memory Manager and Loader to take correct and efficient decisions.

c) The maximum value of T_5 for which the condition determined in *a)* is satisfied is the interarrival time to M_5. Thus:

$$T_{A5} = 10^5\tau$$

The ideal service time of M_5 is equal to the calculation time of F ($10^6\tau$), plus the calculation time of the loop:

$$s = s + reduce\ (A, +)$$

which is of the order of $10^3\tau$ (any way < $10^4\tau$), thus negligible.

In this condition ($T_{5\text{-}id} \sim T_F > T_A$) M_5 is a bottleneck. The ideal parallelism degree of a parallel version of M_5 is:

$$n = \frac{T_{5-id}}{T_{A5}} \sim \frac{T_F}{T_{A5}} = 10$$

which can be exploited on condition that a suitable parallelism form is found.

M_5 is a module with *internal state* (s), thus a farm cannot be applied to the *whole* computation of M_5:

{ ∀ *i = 0 .. M – 1:*
  s = s + A[i];
 y = F(s)
}

Data-parallelism does not solve the problem, because the only part operating on arrays is the loop, and the loop calculation time is < T_A.

We observe that, for each A, the computation structure is the *sequential composition* of two computations:

$$s = s + reduce\ (A, +)$$

which modifies the internal state, and

$$y = F(s)$$

which is a pure function. Thus, M_5 can be parallelized as a *pipeline of two stages* corresponding to the two phases. The first stage, M_{50}, encapsulates and modifies the internal state, and passes the new state value to the second stage, M_{51}, which computes F.

The first stage computes *reduce* sequentially.

The second stage can be parallelized by a *farm* with parallelism degree n = 10. According to the assumptions on part *a)* about communications, the emitter is not a bottleneck, since $T_E = T_{send}(1)$.

In this condition the whole parallelism degree of M_5 is equal to $n + 3 = 13$, and $n_\Sigma = 18$. Assuming that the number of processing nodes is greater or equal than 18, the parallelized version of M_5 is able to achieve the ideal service time

$$T_{5\text{-}eff} = T_{5\text{-}id} = T_A$$

with

$$\varepsilon_5 = 1$$

For Σ the ideal conditions of part *a)* are satisfied again.

For the modules internal to M_5 we have:

module	*ideal service time*	*effective service time*	*efficiency*
M_{50}	$< 10^4\tau$	$T_A = 10^5\tau$	$\rho_{M50} < 0.1$
Emitter of M_{51}	$T_{send}(1) << T_A$	T_A	$\rho_E << 1$
Worker of M_{51}	$T_{calc} = 10^6\tau$	$n\ T_A = 10^6\tau$	$\rho_W = 1$
Collector of M_{51}	$T_{coll} = T_{send}(1)$	T_A	$\rho_{coll} << 1$