Master Degree Program in Computer Science and Networking

High Performance Computing

2nd appello - February 13, 2013

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 stream parallel computation Σ has the following characteristics:

1. input stream element = *(opcode, x)*, output stream element = *(y)*, where *x* and *y* are of integer type, and *opcode* = {0, 1, 2, 3} identifies one of four distinct operations to be applied to *x*;
2. Σ is implemented by a *functional partitioning* scheme with 4 workers W_0, W_1, W_2, W_3, each one specialized for a distinct operation identified by *opcode*. The four operations are equi-probable;
3. processing elements have a scalar pipelined CPU, with clock cycle τ, time slot equal to 2τ, parallel Execution Unit with 4-stage pipelined integer functional units, 32K on-demand primary data cache, on-chip secondary cache;
4. let $u = 10^4 \tau$;
5. the interarrival time of Σ is equal to *u*;
6. $T_{setup} = u/10$, $T_{transm} = u/100$, zero-copy communication and communication processors;
7. worker W_i, for $i = 0, 1, 2$, has its own integer internal state S_i, and is defined as follows:

   $\forall$ input *a* : output $b = F_i\,(a, S_i)$; $S_i = G_i\,(a, S_i)$;
8. the mean processing times of F_0, G_0, F_1, G_1, F_2, G_2 are respectively equal to: *u, u, 4u, 8u, u, 2u*;
9. worker W_3 encapsulates an integer array *C[M]*, with $M = 10^3$, and is defined as follows:

   $\forall$ input *a* : output *b* = number of elements of *C* which are integer multiples of *a*.

   For the evaluation of W_3 processing time, assume that the event "an element of *C* is an integer multiple of *a*" has negligible probability.

a) Evaluate the service time and the relative efficiency of Σ and of each module belonging to Σ implementation.

b) Transform possible bottlenecks in order to improve the service time of Σ.

Question 2

a) Explain the following sentence: "In the performance evaluation of a parallel program on a shared memory architecture, the values of interprocess communication parameters T_{setup} and T_{transm} depend, in general, *also* on the parallel program itself (structure and implementation)".

b) Explain under which conditions we can assume that, with acceptable approximation, T_{setup} and T_{transm} are independent of the parallel program.

Question 3

With reference to a pipelined scalar CPU, explain in quantitative terms what is the impact of the Execution Unit parallelization on performance.

Solution

to be integrated with proper explanations

Question 1

a) The functional partitioning implementation of Σ consists of a distributor module IN, the four workers, and a collector module OUT.

The ideal service times of IN and OUT are respectively equal to $T_{send}(2)$ and $T_{send}(1)$, in practice both are equal to $T_{setup} = u/10 << T_A$, thus they are not bottlenecks.

According to the multiple server theorem, the interarrival time to any worker is equal to $4u$.

The ideal service times of the first three workers are:

$$T_{0-id} = T_{F0} + T_{G0} = 2u$$
$$T_{1-id} = T_{F1} + T_{G1} = 12u$$
$$T_{2-id} = T_{F2} + T_{G2} = 3u$$

All the communications are fully overlapped to computation.

W_0 and W_2 are not bottlenecks, while W_1 is a bottleneck.

Let us evaluate the ideal service time of W_3, without considering the communication primitives (see above), under the assumptions about the CPU architecture. The pseudo-code is:

```
int C[M]; int a, b; int s = 0;
for (i = 0; i < M, i++)
        if ( C[i] % a ) = 0
                s++;
```

which is compiled and optimized as follows:

```
                  LOAD   Rvtg, 0, Ra
                  LOAD   RC, Ri, Rc, don't_deallocate
1. LOOP:          MOD   Rc, Ra, Rmod
2.                INCR   Ri
3.                IF ≠ 0   Rmod, CONT
4.                INCR   Rc
5. CONT:          IF <   Ri, RM, LOOP, delayed_branch
6.                LOAD   RC, Ri, Rc, don't_deallocate
```

(Delayed branch could be applied to IF $\neq$ 0 using INCR Ri, but consequently the distance of the logical dependency induced by instruction 1 becomes 1).

We have to pay the effect of a branch (instruction 3), with probability $\lambda = 1/6$, and of a logical dependency induced by instruction 1 on 3, of distance $k = 2$, probability $d_k = 1/6$, $N_Q = 2$, $L_{pipe-k} = 4$ and no long-latency instruction in the critical sequence ($\Delta_2 = 0$). The dependency of 2 on 5 has no effect. Thus, the service time per instruction, without cache faults, is given by:

$$T = (1 + \lambda)t + \Delta_1 = (1 + \lambda)t + t\, d_k\, (N_Q + L_{pipe-k} + 1 - k) = 2t$$

The completion time without cache faults is:

$$T_{c0} = 6\, M\, T = 12\, M\, t = 24\, M\, \tau$$

Array *C* has the property of *reuse*: after the first stream element application, *C* is permanently maintained in cache. Thus, the cache fault penalty is negligible, and the ideal service time of W_3 is :

$$T_{3-id} = 24\,M\,\tau = 2.4\,u$$

W_3 is not a bottleneck.

In conclusion, only W_1 is a bottleneck, and its steady-state interarrival time becomes *12u*. The steady-state interarrival time of W_0, W_2, W_3 must be re-evaluated. The interdeparture time from IN becomes:

$$\frac{1}{4}\,T_{1-id} = 3u$$

Thus, the steady-state interarrival time to W_0, W_2, W_3 are:

$$\frac{3u}{\frac{1}{4}} = 12u$$

According to the multiple clients theorem, the interarrival time to OUT, equal to the effective service time of Σ, is given by:

$$\frac{1}{T_W} = \frac{1}{T_{p0}} + \frac{1}{T_{p1}} + \frac{1}{T_{p2}} + \frac{1}{T_{p3}} = \frac{1}{T_{A-0}} + \frac{1}{T_{1-id}} + \frac{1}{T_{A-2}} + \frac{1}{T_{A-3}} = \frac{4}{12\,u}$$

Thus, we confirm that the effective service time is equal to the steady-state interarrival time of Σ:

$$T_W = 3\,u$$

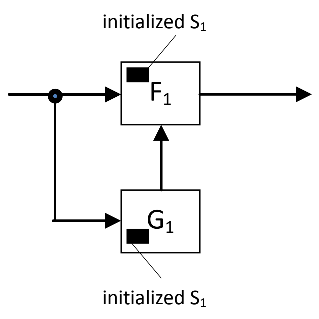
$$\varepsilon_W = \frac{T_{W-id}}{T_W} = \frac{T_A}{T_W} = \frac{u}{3\,u} = 0.33$$

For the component modules:

Module	ideal service time	effective service time	efficiency
IN	~ T_{setup} = 0.1 *u*	$T_W = 3\,u$	ρ_{IN} = 0,033
OUT	~ T_{setup} = 0.1 *u*	$T_W = 3\,u$	ρ_{OUT} = 0,033
W_0	*2 u*	*12 u*	ρ_{W0} = 0,17
W_1	*12 u*	*12 u*	*1*
W_2	*3 u*	*12 u*	ρ_{W2} = 0,25
W_3	*2.4 u*	*12 u*	ρ_{W0} = 0,2

b) Only the data-flow paradigm can be applied to parallelize the bottleneck module W_1, because it is a module with state and operates on elementary types.

AND-logic data-flow graph of W_3



F_1 and G_1 can be executed in parallel on *a*, and the new state produced of G_1 is sent to F_1 (this communication has negligible effect on the service time).

Thus, because F_1 has to wait for the updated state, the effective service time of W_1 is reduced to $T_{G1} = 8u$, and W_1 remains a bottleneck. All the steady-state interarrival times of workers are now equal to *8u*.

For Σ we have:

$$T_W = 2\,u$$

$$\varepsilon_W = 0.5$$

Question 2

a) Parameters T_{setup} and T_{transm} are proportional to the under-load latency Ω for memory access. Notably, $T_{setup} \sim 5\Omega$ and $T_{transm} \sim \Omega/\sigma$ for typical architectural characteristics.

The under-load latency is a function of the base latency Ω_0 and of parameters p and T_p *(insert their definition)* which depend on the parallel program structure (uswed parallel paradigms), on *parallelism degree n*, and on program *mapping*.

In general, in order to evaluate a parallel program, we need to apply an iterative procedure (determine n for a by-experience initial value of T_{setup} and T_{transm}; then determine Ω_0 , p and T_p, which depend also on n; then determine new values of T_{setup} and T_{transm}; then re-evaluate n, …) until convergence. (*Explain clearly the relative dependencies among these parameters*)

b) T_{setup} and T_{transm} are approximately independent of the parallel program if two conditions hold:

i) the impact of p and T_p on Ω is limited in such a way that we can assume that $\Omega \sim \Omega_0$;

ii) because Ω_0 is a function of the average *distance* between nodes, this introduces another element of dependency on the parallel program mapping. Thus, the approximation is acceptable if the mapping and/or the network is not too sensible to the distance.

Question 3

The EU parallelization is able to minimize the EU ideal service time (one time slot): this has an obvious positive effect on the ideal service time of the architecture and, most important, on the mean waiting time in queue, $W(\rho)$, of instructions delivered to the EU server. In fact, the delay Δ incurred by IU for logical dependency is proportional to the EU response time:

$$R_Q = W(\rho) + L_s$$

On the other hand, the EU functional units are parallelized according to the pipeline paradigm, which has a negative effect on EU latency L_s compared to an ideal realization with latency equal to the service time (*explain why, in practice, the pipeline paradigm is the only parallelization forms for streams of arithmetic operations*). This is the reason for which the goal of the logical dependency optimizations is to mask the effects of the EU latency.