Master Degree Program in Computer Science and Networking

High Performance Computing

2nd midterm - December 21, 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 SMP all-cache multiprocessor has the following characteristics:

- N = 16 processing elements with clock cycle τ;
- N = 16 shared interleaved memory macromodules, each one with 8 interleaved modules, and clock cycle equal to 30τ;
- binary generalized fat tree interconnection network, with wormhole flow control, 32-bit links and flits, single buffering rdy-ack communication, and link transmission latency equal to 4τ;
- D-RISC scalar pipeline CPU, with time slot equal to 2τ, on-demand write-through primary data cache with 8-word blocks. The assembler machine includes floating point instructions, and in particular SQRT. The Execution Functional Unit for SQRT is 6-stage pipelined.

1) Evaluate the base latency of a memory read *request* communication, proving that the latency of *every unit* belonging to the PE-M path is equal to one clock cycle (as required for applying the pipelined communication formula).

2) Evaluate the base memory latency of a cache block transfer.

3) Evaluate T_{calc} and T_p metrics for a sequential program that transforms a real array $A[M]$ into a real array $B[M]$, where $B_i = sqrt(A_i)\ \forall i$.

4) For this computation and the given architecture, evaluate whether the base memory latency is a good approximation of the under-load latency.

Question 2

For a NUMA architecture, evaluate the synchronization latency of a locking section, executed by a processing node PE_i, assuming that

i) cache coherence is automatic, directory based with invalidation,

ii) the probability that any other PE_j ($j \neq i$) tries to enter the same locking section, while PE_i is executing the locking section, is equal to 1/10,

iii) the probability that any other PE_j ($j \neq i$) has executed the same locking section, before PE_i tries to enter the locking section, is equal to 3/10,

iv) the lock implementation uses the *notify* approach.

Note: students can request to look up the Course Notes copy on the teacher's desk for formulas or numeric values related to performance metrics.

Solution

Question 1

The generalized fat tree is implemented by a 2-ary 4-fly network, whose switching units are able to execute both the butterfly routing protocol and the tree routing protocol according to the type of message: respectively, for shared memory accesses and for interprocessor communications.

1) A memory read request communication uses a firmware message, of length $m = 2$ words (header, block physical base address), from a PE to a memory macromodule. The distance d is constant and equal to 8: $d_{net} = n = 4$ hops for the network, plus 4 hops including: source data cache, source MINF (external memory interface), source W (PE interface unit), and destination I_M (memory interface unit).

Each hop has latency $t_{hop} = \tau + T_{tr} = 5\tau$. In particular, the base latency of each switching unit is equal to τ because:

- when a header flit is received, one clock cycle is sufficient to apply the butterfly routing function (comparison of two bits in the binary representation of source and destination) and to write the flit into the proper interface (straight or oblique link),
- *and* during any clock cycle the presence of a new header flit is detected, even if a communication is on going on the other input interface, thus no delay is incurred in forwarding the new header.

The latency of W is equal to τ, because all the input interfaces are tested in parallel during any clock cycle, thus a request firmware message is forwarded to the network switching unit even if other actions are on going, e.g. serving an incoming interprocessor firmware message.

Obviously, the latency of data cache, MINF, and I_M is equal to τ.

The base latency for a read request communication is given by:

$$\Omega_{0-req} = (2m + d - 3)\, t_{hop} = 45\,\tau$$

2) The base latency of a cache block transfer is given by:

$$\Omega_0 = \Omega_{0-req} + \tau_M + \Omega_{0-reply} = 200\,\tau$$

since the reply firmware message has length $m = \sigma + 2 =$ words.

3) The basic compilation of the sequential computation is:

```
LOOP:   LOAD   RA, Ri, Ra
        SQRT   Ra, Ra
        STORE  RB, Ri, Ra
        INCR   Ri
        IF <   Ri, RM, LOOP
        END
```

which can be optimized as follows:

```
        LOAD   RA, Ri, Ra
LOOP:   SQRT   Ra, Ra
        INCR   Ri
        STORE  RB, Ri, Ra
        IF <   Ri, RM, LOOP, delayed_branch
        LOAD   RA, Ri, Ra
        END
```

The only degradation is the logical dependency induced by SQRT on STORE, with distance k = 2, probability d_k = 1/5, N_{Qk} = 2, $L_{pipe\text{-}k}$ = 6. Thus:

$$T = t + t\, d_k \left(N_{Qk} + L_{pipe-k} + 1 - k\right) = \frac{12}{5}\, t = \frac{24}{5}\, \tau$$

The calculation time with no cache fault is:

$$T_{calc-0} = 5\, M\, T = 24\, M\, \tau$$

The cache fault penalty is given by:

$$T_{fault} = N_{fault}\ \Omega = \frac{M}{\sigma}\, \Omega = 25\, M\, \tau$$

Thus:

$$T_{calc} = T_{calc-0} + T_{fault} = 49\, M\, \tau$$

and the mean time between two consecutive shared memory accesses:

$$T_p = \frac{T_{calc}}{N_{fault}} = 392\, \tau$$

4) For the SMP architecture, according to the interleaved memory properties, the average number of processing nodes in conflict for the same memory macromodule is given by:

$$p = \frac{N}{B_M}$$

where B_M is the effective bandwidth of the interleaved memory. With 16 PE and 16 memory macromodules, we have $B_M \sim 10$ (Part 2, Section 2.4.2, page 33), thus:

$$p \sim 2$$

With the T_p value evaluated in point 3), we can assert that (Part 2, Section 4.3, page 73) the base latency is a very good approximation of the under-load latency.

Question 2

We have to evaluate the latency of the pair *lock-unlock* for a locking section like

lock (X); CS; *unlock* (X)

When PE_i executes *lock* the X block has to be read into C_i with probability α = 3/10. This transfer does not cause invalidation: it is implemented by a request communication to the home node and a transfer from the owner node PE_j. The latency is

$$\alpha\, (\Omega_{req} + \Omega)$$

The modification of *X* in *lock* implies the invalidation of the block in *Cj* (and possibly in all other PEs having *X* in cache). This has a cost of about

$$\alpha\, \Omega$$

When PE_i executes *unlock*, the *X* block has to be read again into C_i with probability β = 1/10. The owner is PE_j, because the *notify lock* implementation implies a modification of *X*, thus the block transfer latency is

$$\beta\, (\Omega_{req} + \Omega)$$

In conclusion, the locking latency is evaluated as

$$(\alpha + \beta)\Omega_{req} + (2\, \alpha + \beta)\, \Omega = \frac{4}{10}\, \Omega_{req} + \frac{7}{10}\, \Omega$$