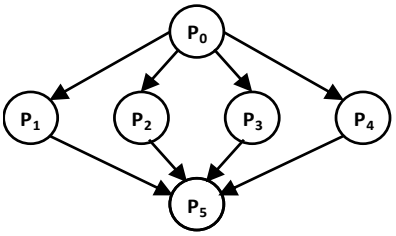
Master Degree Program in Computer Science and Networking, 2014-15

High Performance Computing

2nd appello – February 11, 2015

Write your name, surname, student identification number (numero di matricola), e-mail. 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.

1) A LC parallel program Σ is described by the following OR-graph. P_0 generates an infinite stream. Each stream element is sent to one of $\{P_1, P_2, P_3, P_4\}$ with the same probability.



P_1, P_2, P_3, P_4 are defined as follows, with M = 128K:

P_i (i = 1 .. 4) :: *int* A[M], B[M]; *channel in* input[i] (5); *channel out* output [i];
while (true) *do*
{ *receive* (input[i], A); ∀ *j = 0 .. M – 1*: B[j] = *F_i*(A[j]); *send* (output[i], B) }

Any function *F_i (i= 1 .. 4)* executes 20 instruction on the average.
The ideal service time of both P_0 and of P_5 is $10M\tau$.

Σ is executed on the multiprocessor architecture specified below. Each module is mapped onto a distinct CMP. The choice of the CMPs, onto which the six modules are mapped, is left to the program designer.

a) Study the program behavior in terms of communication support and cache management. According to the base latency, and assuming that the service time per instruction is equal to 2τ, evaluate the effective service time and the relative efficiency of Σ and of P_0, generic P_i, P_5.
b) Determine the parameters p, T_p, R_{Q0}, T_s for the evaluation of the under-load latency R_Q, assuming that P_1, P_2, P_3, P_4 are the most stressed modules.
c) *Optional*: determine the value of R_Q and re-evaluate the metrics of point *a)*.

2) Explain whether the following statements " … " are true or false or true under certain conditions:

a) the instruction STORE Ra, Rb, Rc is used for transmitting a word to the communication unit (UC) of a PE. "The instruction fails if UC doesn't contain a local memory component";

b) consider the evaluation of the cache-to-cache base latency to be used in the cost model of a given parallel program on a given multiprocessor architecture. "Such latency is independent of the parallelism degree of the program".

3) Consider a stencil-based data-parallel implementation of the following module operating on streams:

Q :: *int* A[M], B[M]; *channel in* input_stream (1); *channel out* output_stream; < init B >;
while (true) *do*
{ *receive* (input_stream, A); ∀ *i = 0 .. M – 1*: B[i] = F(A[i], B[(i + M – 1)%M]); *send* (output_stream, B) }

a) Describe the Virtual Processors version in LC.
b) For the actual implementation, discuss the impact of the stencil on the effective parallelism degree. Assume that: *i)* communications are overlapped, *ii)* data distribution/collection is not bottleneck, and *iii)* the calculation time of function *F* is much greater that the instruction service time.

Architectural specifications for question 1

Multiple-CMP NUMA-SMP machine with $N = 64$ processing elements:

- 16 4-PE CMPs with internal crossbar interconnect and one MINF;
- D-RISC CPU with 32K + 32K primary cache, 8-word blocks, and 1M inclusive secondary cache;
- *no communication processor;*
- directory-based cache coherence with home-flushing semantics;
- external interconnect: wormhole *single-buffering* binary Fat Tree, 1-word flits and links, $T_{tr} = \tau$;
- 64-Gigaword local memory for each CMP.

LC run-time support: Rdy-Ack interprocess communication with I/O-based synchronization, exclusive-mapping.

Solution

1)

a) The best homing strategy for minimizing both contention ($p = 3$) and communication latency is: each P_i processing node is the home of the VTG shared structures VTG_1 and VTG_2, corresponding to channels *input[i]* and *output[i]* respectively.

The execution of P_0 *send* provides *to flush* the A blocks (VTG_1) into C2 of P_i (C2i) through *C2C* transfers, and to invalidate such blocks locally.

Since M = 128K and γ_{C2} = 1M, there is ample space in C2i to store VTG_1, the computed B, and VTG_2.

C1i generates $4M/\sigma_I$ faults, all served through C1-C2 transfers: M/σ_I for VTG_1 reading, M/σ_I for B writing (*write-back*), M/σ_I for B reading, M/σ_I for VTG_2 writing (*write-back*). Thus, P_i performs only local accesses; main memory updating is asynchronous and has negligible impact on performance.

The execution of P_5 *receive-compute* provides to read VTG_2 through *C2C* transfers when needed.

Rdy-Ack synchronizations are implemented via I/O-based wait-notify operations, which imply no shared memory operation.

Base latency analysis

The D-RISC code of P_i is shown in the following. Since calculation and communication latencies are *O(M)*, all the actions in *receive* and *set ack*, and the setup phase in *send*, have negligible latency and are not expanded:

```
< receive >        // target VTG_S pointed to by Rvtg1 //
        & σ1-unfoding, write-back &
        {               CLEAR   Rj
          COMPUTE:      LOAD    Rvtg1, Rj, Ra
                        CALL    RF, Rret  // input and output parameter of F: in Ra, Rb respectively//
                        STORE   RB, Rj, Rb, don't_deallocate
                        INCR    Rj
                        IF <    Rj, RM, COMPUTE
        }
< set ack >
< send >  // only the message copy is shown; target variable pointed to by Rvtg2 //

        & σ1-unfoding, write-back &
        {               CLEAR   Rj
          COPY:         LOAD    RB, Rj, Rb
                        STORE   Rvtg2, Rj, Rb
                        INCR    Rj
                        IF <    Rj, RM, COPY
        }
```

The ideal service time of P_i is given by:

$$T_i = T_{i-0} + T_{fault} \sim M\,(9\,T_{istr} + T_F) + 4\,\frac{M}{\sigma_1}\,L_{c1-c2}\,(\sigma_1)$$

Notice that, since there is no communication processor, the *send* communication overhead (M/σ_1 blocks readings, M/σ_1 blocks writings for message copy) is paid entirely.

The C1-C2 block transfer (read or write-back) is evaluated as:

$$L_{C1-C2}(\sigma_1) = (\sigma_1 + 2)\,\tau = 10\,\tau$$

Thus:

$$T_i = 63\,M\,\tau$$

P_is are bottlenecks. According to the multiple-server theorem in presence of bottlenecks, the effective service times of P_0 becomes:

$$T_0 = \frac{1}{4}\,T_i = 15.75\,M\,\tau$$

According to the multiple-client theorem, the interarrival time to, and the effective service time of, P_5 is:

$$T_5 = \frac{1}{4}\,T_i = 15.75\,M\,\tau$$

In the base latency case, the performance measures of the single modules and of the whole computation are:

	ideal service time	effective service time	relative efficiency
Σ	*10 M τ*	*15.75 M τ*	*0.63*
P_0	*10 M τ*	*15.75 M τ*	*0.63*
P_i	*63 M τ*	*63 M τ*	*1*
P_5	*10 M τ*	*15.75 M τ*	*0.63*

b) *Under-load analysis*

The contention on the most stressed servers C2i is measured by:

$$p = 3$$

All the requested actions (read, write-flush, write-back) operate on a single block, thus the service time of C2i is

$$T_s = \sigma_1\,\tau = 8\,\tau$$

C2i receives the following requests from C20, C25 and C1i:

1. M/σ_1 C2C write-flush operations with latency $L_1 = L_{C2C\text{-}flush}(\sigma_1)$: from C20,
2. M/σ_1 C2C read operations with latency $L_2 = L_{C2C\text{-}read}(\sigma_1)$: from C25,
3. $2M/\sigma_1$ read and $2M/\sigma_1$ operations with latency $L_3 = L_{C1\text{-}C2}(\sigma_1) = 10\tau$: from C1i.

The respective frequencies are:

$$\pi_1 = \pi_2 = \frac{1}{6} \qquad \pi_3 = \frac{4}{6}$$

The base latency is given by:

$$R_{Q0} = \sum_{i=1}^{3} \pi_i\,L_i$$

The C2C *synchronous* flush operation consists in a request phase with a firmware message of length 11 (header, physical address, block words) and a reply phase with a pure-synchronization firmware message of length 1 (header). The C2 accesses are pipelined word by word.

The C2C read operation consists in a request phase with a firmware message of length 3 (header, physical address) and a reply phase with a firmware message of length 9 (header, block words). The C2 accesses are pipelined word by word.

Latencies L_1 and L_2 are computed as follows for single-buffering communications:

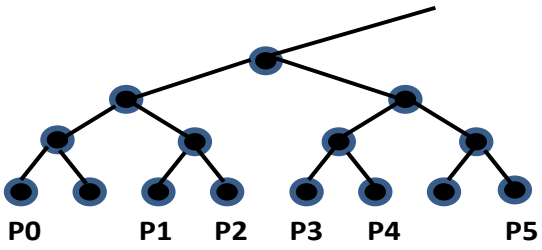
$$(2s_{req} + 2s_{reply} + 2d - 6)T_{hop}$$

with $T_{hop} = 2\tau$ as the maximum latency in the pipelined path.

In all cases, the path is: C2, W, internal net, MINF, WW, external net, WW, MINF, internal net, W, C2, whose distance is:

$$d = 10 + d_{external-net}$$

The best mapping on the external network consists in using only one half of the Fat Tree, in order to have a distance close to lg_2N. To evaluate $d_{external\text{-}net}$ accurately, we must study the actual mapping. Since P_0 and P_5 should be symmetric with respect to the P_is, a reasonable mapping is:



The *average* distance between P_0 or P_5 and a generic P_i is:

$$d_{external-net} = 5$$

(formally $1 + lg_2N$). Thus:

$$d = 15$$

$$L_1 = L_2 = 96\,\tau$$

$$R_{Q0} \sim 39\,\tau$$

The mean time T_p between two requests is given by the inverse of the bandwidth requested to C2i:

$$T_p = \frac{1}{B_{req}}$$

We have:

$$B_{req} = \frac{\frac{M}{\sigma_1}}{4T_0} + \frac{4\frac{M}{\sigma_1}}{T_i} + \frac{\frac{M}{\sigma_1}}{4T_5} = 6\frac{\frac{M}{\sigma_1}}{T_i}$$

Notice that, for each P_i,we must consider the interdeparture time of P_0 towards such P_i, which is equal to the service time of P_i. Analogously for P_5. Thus:

$$T_p = 84\,\tau$$

c - Optional) Solving the client-server model with $p = 3$, $T_p = 84\tau$, $R_{Q0} = 39\tau$, and $T_s = 8\tau$, we find

$$\rho = 0.19 \qquad \frac{R_Q}{R_{Q0}} = 1.02$$

Thus the evaluation of Σ and of the single modules, which has been done for the base latency case in point *a)*, remains valid with very good approximation.

2)

a) False. The STORE instruction is used in Memory Mapped I/O mode: this means that the CPU process "sees" the I/O unit as a set of memory locations referred to by logical addresses (in our case: R[Ra] + R[Rb]). However it is not necessary that such logical addresses correspond to physical locations of a memory component. The I/O unit can interpret the received information (write, physical address, data word) in the most efficient way for implementing the provided functionality. In the UC case, each received data word is sent "on the fly" (i.e. without storing it) to the PE interface unit (W) in a single clock cycle.

b) The base latency is linearly dependent on the inter-node *distance*. Thus, the statement is true only for architectures having a constant distance interconnect: in the case of C2C accesses, the only interconnect with constant distance is the crossbar. For any other interconnect (*k*-ary *n*-cubes and their particular cases, e.g. rings and 2-D meshes, Fat Trees in any implementation including Generalized Fat Tree) the statement is false: the inter-node distance depends on the parallel program structure, the number of processes and their mapping. An example has been seen in Question 1: mapping of the 6-module graph onto a 16-node Fat tree).

3)

a) The Virtual Processors version is an array VP[M] with *ring* logical interconnect:

```
parallel VP[M]; int A[M], B[M]; channel ring[M], input_stream[M], output_stream[M];

VP[i | i = 0 .. M – 1] ::    int A[i], B[i], int x; channel in ring[(i + M – 1)%M] (1), input_stream[i] (1);
                             channel out ring[(i + 1)%M], output_stream[i];
    while (true) do
        { < receive (input_stream[i], A[i]); >
          send (ring[(i + 1)%M], B[i]);
          receive (ring[(i + M – 1)%M], x);
          B[i] = F(A[i], x);
          < send (output_stream[i], B[i]) >
        }
```

It is necessary that the ring channels (stencil channels) are *asynchronous*, in order to avoid deadlock. Asynchrony degree $k = 1$ is exactly what is needed.

Despite asynchronous overlapped communications, the stencil is *synchronous*, since each VP must wait the *x* value before calculating function *F*.

Notice that the Virtual Processor description doesn't include data distribution and collection. Even the communications from/onto the input/output stream (< … >) are not fundamental in the Virtual Processors description.

b) In the actual implementation with *n* workers, each worker encapsulates a partition of A and of B of size $g = M/n$. The input stream value A is *scattered* into the workers. The worker results are *gathered* into the output stream.

In the actual stencil implementation, each worker sends the last element *B[g-1]* of B partition to the successive worker in the ring, and the value received from the preceding worker represents the new first element *B[0]* of the B partition. Thus, *the stencil channels are of type integer* (1 word), as in the Virtual Processors version.

Because of the assumptions *i), ii), iii)*, the optimal parallelism degree is evaluated on the sequential version of Q as follows:

$$n_0 = \frac{M\,T_F}{T_A} \quad \rightarrow \quad T^{(n_0)} = T_A$$

where T_A denotes the interarrival time.

As said, the synchronous stencil feature forces to wait a communication latency $L_{com}(1)$. With parallelism degree n_0, the service time becomes:

$$T^{(n_0)} = g\,T_F + L_{com}(1) = \frac{M\,T_F}{n_0} + L_{com}(1) > T_A$$

The degradation is negligible if

$$L_{com}(1) \ll T_A$$

In general, the effective optimal parallelism degree n_1, i.e., able to guarantee the ideal bandwidth $1/T_A$, is such that:

$$\frac{M\,T_F}{n_1} + L_{com}(1) = T_A \qquad \rightarrow \qquad n_1 = \frac{M\,T_F}{T_A - L_{com}(1)}$$

It is: $T_A - L_{com}(1) > 0$, otherwise no parallelization is possible.