1. A system $\Sigma$ at the processes level is composed of two modules $P_1$ and $P_2$. $P_1$ internally encapsulates a matrix $Z[R][M]$ and generates a stream of $R = 10^3$ messages where the $i$-th message is the $i$-th row, of size $M = 4K$ integers, of the matrix. Process $P_2$ receives the messages generated by $P_1$ and applies the computation described by the following LC-like pseudo-code:

$$P_2:: \{ \text{int } A[M], \text{int } B[M]; \ ...
\text{for } k=0 \text{ to } R-1 \text{ do } \{ \text{receive}(\text{ch}_{\text{in}}, A); \}
B = \{0\}; // \text{all } B\text{'s elements are initialized to zero}
\text{while } (\text{sum}(B) < \delta) \text{ do } \{ \text{for } i=0 \text{ to } M-1 \text{ do } B[i] = F(A[M-i-1], B[i]); \}
\text{send}(\text{ch}_{\text{out}}, \text{sum}(B)); \}
\}$$

The algorithm utilizes an array $B[M]$ encapsulated in the process $P_2$. The symbol $\delta$ denotes a positive fixed constant while $\text{sum}(B)$ means the sum of all the elements of the array $B$. Assume that the average number of iterations of the while loop is $I = 10$ and the average calculation time of the function $F$ is $T_{fr} = 180\tau$. With reference to the architectural specifications provided at the end of the page, evaluate the system by providing detailed answers to the following points:

a) According to the base latency, study the system “as it is” by finding the effective service time and the relative efficiency of the processes and of the system. In the analysis assume that the number of D-RISC instructions within the body of the innermost for loop is 12;

b) Verify that process $P_2$ is a bottleneck and study a data parallel parallelization by applying the Virtual Processors approach. Evaluate the same parameters at point a) for the parallelized system $\Sigma^{(n)}$ by using the base latency only.

2. Consider a directory-based cache-coherent NUMA multi-CMP system working with basic invalidation semantics. Provide detailed answers and explanations to the following question:

a) Provide a detailed cost model of the $T_{\text{setup}}$ parameter of the send primitive executed by the home node and the list of cache-to-cache firmware messages exchanged during this phase. Assume the RDY-ACK run-time support with shared synchronization flags.

**Architectural specifications for Exercise 1**

- Multi-CMP SMP multiprocessor architecture with 8 CMPs and an external SMP network based on a binary butterfly with wormhole flow control (1 flit = 32 bits);
- Each CMP has 16 PEs, an internal network that can be approximated as a crossbar, and 4 on-chip MINFs connected to 4 parallel off-chip WW units;
- Each PE has a C1d of 32K and a C2 of 256K both working on-demand, $T_{\text{instr}} = 2\tau$, block size $\sigma_1 = 8$ and $\sigma_2 = 128$ words;
- External memory composed of mutually interleaved macro-modules with $\tau_M = 40\tau$;
- Assume that each memory location corresponds to a word of 32 bits;
- The transfer latency of a cache block from the C2 to the C1 of a PE is $L_{C2-C1}(\sigma_1) = (\sigma_1 + 2)\tau$;
- All the firmware interfaces use double buffering with $T_{tr} \approx \tau$ inter-chip;
- Directory-based cache coherence with home flushing semantics;
- Exclusive mapping, communication overlapping and RDY-ACK run-time support based on shared synchronization flags.
Solution

1. We analyze the process $P_1$ first. The ideal service time is dominated by the send latency. We recognize locality in the cache blocks of the matrix $Z$. We assume the home node of the channel $ch$ in the PE of the receiver process. So, we have:

$$T_{id-P_1} \approx T_{send}(M) = \frac{M}{\sigma_1}(l_{read-c1}(\sigma_1) + L_{write-c2c}(\sigma_1)) + M \cdot \beta_{code}$$

The evaluation of the two base latencies are:

- $L_{read-c1}$: the path is C1-C2-W-CROSSBAR-MINF-WW-EXT_NETWORK-Im-M that is $d = 8 + d_{net}$. The external network is a 2-ary 5-fly network used as a butterfly for the PE-memory interactions, and as a fat tree for messages among PEs. Therefore, $d_{net} = 5$ and $d = 13$. The latency of a cache block transfer is:

$$L_{read-c1}(\sigma_1) = (3 + 13 - 2) \cdot T_{hop} + \tau_M + (9 + 13 - 2) \cdot T_{hop} = 108\tau$$

- $L_{write-c2c}(\sigma_1)$: we assume that the two processes are mapped onto different CMP, we choose CMP0 and CMP1. The path is C1-C2-W-CROSSBAR-MINF-WW-EXT_NETWORK-WW-MINF-CROSSBAR-W-C2-C1 (the message can be flushed in the C1 of the home node). That is, $d = 12 + d_{net}$. The network is used as a fat tree. According to the mapping chosen, we have $d_{net} = 5$:

$$L_{write-c2c}(\sigma_1) = (11 + 17 - 2) \cdot T_{hop} + (1 + 17 - 2) \cdot T_{hop} = 84\tau$$

The ideal service time is:

$$T_{id-P_1} \approx 28M\tau$$

Which is also the inter-arrival time to the second process that should be analyzed in detail. We recognize locality and reuse in the cache blocks of A and B which can be both contained in the C1. We have:

$$T_{calc-P_2} = 10M(12 \cdot T_{instr} + T_F) = 10M(24 + 180\tau) = 2040M\tau$$

Since the output message generated by $P_2$ is a single word, it has no practical impact on the ideal service time of the process:

$$T_{id-P_2} \approx T_{calc-P_2} = 2040M\tau$$

Therefore, the process is a bottleneck and its optimal parallelism degree is given by:

$$n_{opt} = \left\lfloor \frac{T_{id-P_2}}{T_{id-P_1}} \right\rfloor = \left\lfloor \frac{2040M\tau}{28M\tau} \right\rfloor = 73$$

The following table summarizes the results of the first phase:

<table>
<thead>
<tr>
<th>Module</th>
<th>Ideal service time</th>
<th>Effective service time</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$28M\tau$</td>
<td>$2040M\tau$</td>
<td>0.014</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$2040M\tau$</td>
<td>$2040M\tau$</td>
<td>1</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>$28M\tau$</td>
<td>$2040M\tau$</td>
<td>0.014</td>
</tr>
</tbody>
</table>

In the second phase, we present a data parallel parallelization of the second process. To respect the owner compute rule, we recognize $M$ Virtual Processor each one defined as follows:

$$VP_{i=0, \ldots, M-1} = \{A[M - i - 1], B[i]\}$$

Accordingly, the scheme of the parallelization is a map-reduce pattern, where the reduce is synchronous with respect to the map phase. In fact, we observe that the global reduce result on B is needed to evaluate the condition of the while loop executed by all the worker processes and to decide whether a new iteration
must be computed or not. The following figure depicts one possible solution and implementation for the given problem, where the reduce phase is implemented in a centralized fashion by a process R responsible of the global reduce result and the management of the control flow of the overall computation.

The homing of the channel is chosen in order to obtain a low-p mapping:

- ch_in is homed in the PE of process S;
- ch_Win[i] is homed in the PE of the i-th worker;
- ch_Wout[i] is homed in the PE of the i-th worker;
- ch_R[i] is homed in the PE of the i-th worker;
- ch_out is homed in the PE of the R process.

Furthermore, we choose this mapping of processes onto CMPs:

- CMP0: process P1
- CMP1: processes S and R
- CMP2: 16 workers
- CMP3: 16 workers
- CMP4: 16 workers
- CMP5: 16 workers
- CMP6: 9 workers

We can observe that according to this mapping the evaluation of the ideal service time of the first process is exactly identical to the one that we did for the initial evaluation.

While the array B is statically partitioned among workers, the array A must by dynamically partitioned per stream element by a dedicated scatter process which assign to worker \(W_i\) partition \(A_{n-i-1}\) of the array A where \(n = 73\). The scatter ideal service time is:

\[
T_{scatter} = n \cdot T_{send} \left( \frac{M}{n} \right) \approx M \cdot T_{trasm} + \frac{M}{\sigma_1} (l_{write-c2c}(\sigma_1)) + M \cdot \beta_{code}
\]

In fact, we observe that the cache blocks of A have been flushed into the C1 of the process S and the cache lines of the target variables of each ch_Win[i] are flushed into the C1 of the corresponding worker.

To analyze the base latency of a synchronous STORE with flush annotation we evaluate the average distance between the scatter process and the workers using the decided mapping in the fat tree:

\[
d_{net} = \frac{32}{73} \cdot 7 + \frac{41}{73} \cdot 9 = 8.12 \approx 8
\]

The base latency of the STORE with flush (synchronous) is:

\[
l_{write-c2c}(\sigma_1) = (11 + 20 - 2) \cdot T_{hop} + (1 + 20 - 2) \cdot T_{hop} = 96\tau
\]
The scatter ideal service time is:

\[ T_{\text{scatter}} = 16Mτ < T_{\text{id-P1}} = 28Mτ \]

and it is not a bottleneck.

The ideal service time of the worker must take into account the latency of the global reduce and of the multicast. Although a lower latency can be obtained with tree-based structures mapped onto the workers (for the reduce and multicast), we choose a centralized solution because of the low cost of the global reduce calculation and transmission (it is one word). Therefore:

\[ T_{\text{id-w}} \approx \frac{10M}{n} (12 \cdot T_{\text{instr}} + T_P) + L_{\text{reduce}} + L_{\text{multicast}} \]

We can observe that the latency of the reduce is:

\[ L_{\text{reduce}} = T_{\text{send}}(1) + (n-1)T_+ \]

Owing to the small size of the messages for the local reduce results on each partition (produced by the workers), which is one integer only, the small number of workers (few tens) and the very fine-grained nature of the reduce operator (it is a sum), this latency is approximated in the order of few hundreds of clock cycles with a completely negligible impact in the worker ideal service time. The same considerations can be applied for the multicast latency which is:

\[ L_{\text{multicast}} = n \cdot T_{\text{send}} \approx 10^3τ \]

The ideal service time can be approximated as:

\[ T_{\text{id-w}} = 27.82Mτ \]

The difference with the inter-arrival time from the first process is sufficient to amortize completely the latency of the reduce and multicast activities, even in the case they are implemented sequentially.

The ideal service time of the R process is negligible and this can be easily verified. In conclusion:

<table>
<thead>
<tr>
<th>Module</th>
<th>Ideal service time</th>
<th>Effective service time</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1 )</td>
<td>28Mτ</td>
<td>28Mτ</td>
<td>1</td>
</tr>
<tr>
<td>( S )</td>
<td>16Mτ</td>
<td>28Mτ</td>
<td>0.57</td>
</tr>
<tr>
<td>( Wi )</td>
<td>27.82Mτ</td>
<td>28Mτ</td>
<td>0.99</td>
</tr>
<tr>
<td>( R )</td>
<td>( \approx 0 )</td>
<td>28Mτ</td>
<td>( \approx 0 )</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>28Mτ</td>
<td>28Mτ</td>
<td>1</td>
</tr>
</tbody>
</table>

2. The home node of the channel is the PE of the sender process. With any asynchrony degree, the send primitive manipulates a VTG_S structure whose logical address is obtained by accessing to the channel’s support variables private of the sender (i.e. the CH_P array with the logical addresses of the VTG_S structures and the index variable stating which is the VTG_S structure to use for the next send). Each VTG_S structure consists in one cache block (called CH) for the RDY and ACK flags and \( L/σ_1 \) blocks for the target variable. The \( T_{\text{setup}} \) parameter consists in the fixed cost paid independently of the length of the message to be copied.

In the first step, the sender executes a LOAD instruction to access the value of the ACK flag in the block CH. The request is generated by its C1 and reaches C2 where the GSK entry of CH is contained. Accordingly, the C2 of the sender transmits a C2C read request to the C2 of the receiver PE, and the words of the blocks are received and allocated in C2 and C1 of the sender. If the ACK flag is 1, the sender directly goes to step two. Otherwise, if it is 0 the sender executes a busy-waiting loop in its C1 which does not cause any cache-to-cache firmware message. When the receiver executes the set_ack primitive on that VTG_S structure, a STORE notification message is sent from the PE of the receiver to the one of the sender and the block CH is invalidated in the C2 and C1 of the sender PE. Consequently,
the next LOAD in the busy-waiting loop causes another C2C read request transmitted by the C2 of the sender to the C2 of the receiver, and the sender process goes to the second step.

In the second step, the sender executes two STORE instructions, the first puts ACK to zero while the second puts RDY to one. The latter is marked with the write_back annotation and has a synchronous semantics. Its interpretation causes the update of the block CH in C2 of the sender and the generation of an invalidation message to the C2 (propagated to the C1) of the receiver’s PE. The acknowledgment message from the receiver’s C2 is explicitly waited by the sender’s C2 owing to the synchronous semantics of the STORE instruction. In conclusion, we have:

\[ T_{\text{setup}} = (1 + p_{\text{wait}}) \cdot L_{\text{C2C}}(s_1) + L_{\text{invalidation}} + \beta_{\text{code}} \]

Where \( p_{\text{wait}} \) is the probability that once the CH block is read by the sender’s PE, the value of the RDY flag is zero and the sender process has to wait before overwriting the target variable with a new message. The parameter \( \beta_{\text{code}} \) is the execution time of few assembler instructions (one LOAD, one conditional jump, and two STORE instructions that are part of the setup time of the inter-process communication latency).