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, focusing on the requested issues.

1. A system $\Sigma$ at the processes level is composed of three modules interconnected as depicted in the figure below. The first process is in charge of producing a stream of integers (each represented by one word of 32 bits) with an average calculation time $T_{\text{calc-P1}} = 10^3 \tau$. Each stream element is transmitted either to the second or to the third process with probability 1/3 and 2/3 respectively. Process $P_2$ executes the following pseudo-code where $s$ is an integer variable statically initialized to zero and the two functions $F_1$ and $F_2$ have the same calculation time equal to $1.7 \times 10^3 \tau$:

   $P_2:: \text{ while true do } \{ \text{ receive(ch\_in, x); } \ s = F_1(x, s); y = F_2(s); \text{ send(ch\_out, y); } \} \ldots$

   The last process non-deterministically receives from the two input channels the results produced from $P_1$ and $P_2$, and for each input $x$ executes the following pseudo-code to produce a result $y$ (integer):

   $P_3:: \text{ while true do } \{ \ x' = G_1(x); y = G_2(x', z); z = G_3(x', z); \} \ldots$

   The pseudo-code uses three functions whose average calculation times are $T_{G_1} = T_{G_2} = 0.9 \times 10^3 \tau$ and $T_{G_3} = 2.0 \times 10^3 \tau$. The whole program is executed on a CMP-based SMP multiprocessors with exclusive mapping strategy. The inter-process communication latency is considered negligible.

   a) For the whole system, and for each of its processes, find the relative efficiency and the effective service time at the steady state.

   b) Propose a parallelization of the system where bottlenecks are eliminated or reduced as much as possible. Discuss the parallelization(s) and compute the effective service time of the whole system. (Optional) Propose optimizations to increase the relative efficiency of the processes/sub-systems by still achieving the highest bandwidth as possible.

2. A process $P$ has two input channels each conveying arrays of $M = 24K$ integers (each represented by one word of 32 bits). Its behavior is described by the following LC pseudo-code where the call $F(x, V)$ has pure calculation time $12M\tau$ and accesses exactly one time all the cache lines of the array $V$ passed as second parameter to the function:

   $P:: \ldots <\text{initialization of arrays } A \text{ and } B> \ldots$

   \{
   \text{ while true do } \text{ alternative } \{
   \text{ receive(ch\_in1, A) do } z = F(A[0], B); \\
   \text{ receive(ch\_in2, B) do } z = F(B[0], A); \\
   \} \text{ send(ch\_out, z); }
   \}

   Analyze the performance of the process by providing detailed answers to the following two points (add all the necessary justifications to support your analysis).

   a) Provide a worst-case and a best-case approximation of the ideal service time of the process $P$ by assuming that each PE of the underlying architecture has C1d/C2 private caches of capacity 32K and 512K respectively. Assume that the transfer latency of a cache line from the C2 has cost $L_{c2-c1} = (\sigma + 2)\tau$ (with $\sigma = 8$ words) while the latency from the main memory is of $L_{\text{read-c1}} = 80\tau$.

   b) Assume that the arrival rate from the first channel is of 250 arrays per second while from the second one is of 100 arrays per second. Provide an approximation of the ideal service time of the process in the average case.
Solution Overview

1. The ideal service time of the first process is given by:
   \[ T_{id-P1} = T_{calc-P1} = 1000\tau \]

   The inter-arrival time to the second process is given by the multiple-server theorem of our graph analysis:
   \[ T_{A-2} = \frac{T_{id-P1}}{1/3} = \frac{1000\tau}{1/3} = 3000\tau \]

   The ideal service time of the second process is:
   \[ T_{id-P2} = T_{calc-P2} = 3400\tau \]

   The utilization factor of the second process is:
   \[ \rho_2 = \frac{T_{id-P2}}{T_{A-2}} = \frac{3400\tau}{3000\tau} = 1.13 > 1 \]

   The process is a bottleneck. We correct the inter-departure time from the source as follows:
   \[ T_{D-1} = T_{id-P1} \cdot 1.13 = 1130\tau \]

   The inter-arrival time to the third process is:
   \[ T_{A-3} = \frac{1}{\frac{1}{T_{D-1-3}} + \frac{1}{T_{D-2}}} = \frac{1}{\frac{1}{1695\tau} + \frac{1}{3400\tau}} = 1130\tau \]

   This is an expected result, since it must be equal to the effective service time of the source. The ideal service time of the third process is:
   \[ T_{id-P3} = 3800\tau \]

   The utilization factor of the third process is:
   \[ \rho_3 = \frac{T_{id-P3}}{T_{A-3}} = \frac{3800\tau}{1130\tau} = 3.36 > 1 \]

   The third process is a greater bottleneck than the second one. We correct again the inter-departure time from the source:
   \[ T'_{D-1} = T_{D-1} \cdot 3.36 = 3800\tau \]

   Also, this result is expected by the theory. The inter-departure time from the last process, i.e. the effective service time of the system is:
   \[ T_{\Sigma} = T_{id-P3} = 3800\tau \]

To answer point a), we provide the following table of results.

<table>
<thead>
<tr>
<th>Module/System</th>
<th>Ideal Service Time</th>
<th>Effective Service Time</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1000\tau</td>
<td>3800\tau</td>
<td>0.26</td>
</tr>
<tr>
<td>P2</td>
<td>3400\tau</td>
<td>11400\tau</td>
<td>0.30</td>
</tr>
<tr>
<td>P3</td>
<td>3800\tau</td>
<td>3800\tau</td>
<td>1</td>
</tr>
<tr>
<td>\Sigma</td>
<td>1000\tau</td>
<td>3800\tau</td>
<td>0.26</td>
</tr>
</tbody>
</table>

We repeat the analysis by removing the bottlenecks. When we visit the second process, we discovered that its utilization factor was \( \rho_2 = 1.13 \), so the module needs to be parallelized and its optimal parallelism degree is \( n_2^{opt} = 2 \). We propose a pipeline parallelization as follows:
The first stage is in charge of computing the function $F$ by reading and updating the private variable $s$ whose value is sent to the second stage in charge of computing the second function to produce the output result $y$. The two stages are balanced and the ideal service time of the parallelization is:

$$T_{id\text{-pipe}} = 1700\tau$$

The bottleneck has been eliminated and the source inter-departure time does not need to be corrected. The inter-arrival time to the third process becomes:

$$T_{A-3} = \frac{1}{\frac{1}{T_{D-1-3}} + \frac{1}{T_{D-2}}} = \frac{1}{\frac{1}{1500\tau} + \frac{1}{3000\tau}} = 1000\tau$$

As expected. The utilization factor of the third process is:

$$\rho_4 = \frac{T_{id\text{-}p3}}{T_{A-3}} = \frac{3800\tau}{1000\tau} = 3.80 > 1$$

With optimal parallelism degree equal to $n_{2}^{opt} = 4$. We can immediately state that no parallelization of the third process can remove the bottleneck (at least with the information we have about the computation performed by $P_3$, which is based on the sequential application of three distinct functions). To alleviate the problem, we propose a parallelization based on the data-flow paradigm where:

- we have true data dependencies (read-after-write) between the application of the function $G_1$ with the functions $G_2$ and $G_3$;
- we have an anti-dependency (write-after-read) between functions $G_2$ and $G_3$.

The data-flow graph has the following structure:

The internal state $z$ is allocated in a primary copy in the process in charge of computing $G_3$. However, the behavior of process $P_{3,2}$ must guarantee that the updated value of $z$ is always utilized in the computation. Its LC pseudo-code should be like that:

```plaintext
P_{3,2}: ...
{
    while true do
        receive(ch_in1, x');
    
```
z = G_3(x', z);
send(ch_out, z);
receive(ch_in2, z); // we need the new value of the state for the next stream element
}

For the graph analysis, we can check that processes P_{3,1} and P_{3,2} are not bottlenecks while the process in charge of updating the state is a bottleneck:

\[ \rho_{3,3} = \frac{2000\tau}{1000} = 2 > 1 \]

Process P_{3,2} is forced to wait the slowest stream (the one from P_{3,3}) and its inter-departure time (effective service time of the data-flow parallelization and of the whole system) becomes:

\[ T_\Sigma = T_{D-DataFlow} = 2000\tau \]

The whole system is the following:

The presence of a bottleneck (although reduced) in process P_{3,3} would require to re-evaluate the inter-arrival time to the second module (parallelized as a pipeline) by correcting again the inter-departure time from the source:

\[ T''_{D-1} = 1000\tau \cdot \rho_{3,3} = 2000\tau \]

The new inter-arrival time to the pipeline becomes 6000\tau and the pipeline parallelization is no longer necessary to improve bandwidth. So, it is convenient to replace the second module with the original sequential system (one process) in order to increase the relative efficiency of the second module (it was of 0.29 in the pipeline version, it becomes 0.57 with the sequential process, a bit better).

2. It is quite evident that the communication latency of the result (1 integer) can be neglected with the given size of the arrays M = 24K words. Therefore, the ideal service time of the process is equal to its calculation time:

\[ T_{calc} = T_{calc-o} + T_{fault} = 12M\tau + T_{fault} \]

In both the input guards of the alternative command, the code of the process uses two arrays, the one received in the guard (only the first cache line to access the first element) and the whole set of cache lines of the other array (M/\sigma cache lines according to the assumptions on the function F). In the worst-case situation, the process receives an element from one input channel and processes it, and then receives a new element from the other input channel (in other words we have a round-robin behavior). For example, suppose that P receives an array A from the first channel ch_in1. By computing F, it accesses to all the cache lines of B and only to the first line of A. Such M/\sigma + 1 cache lines will be at the end of the
computation in the C2/C1d of the processor executing $P$. Then, in the worst-case, the process receives the next stream element from the second channel ch_in2 (a new array $B$). To compute $F$, it has to load all the remaining cache lines of a $A$ from the memory (only the first cache line has been already in C2/C1d) and the first cache line of $B$ (also from the memory). Therefore:

$$T_{fault} \sim \frac{M}{\sigma}L_{readc1} = 10M\tau$$

The symmetric behavior happens when the process receives from ch_in2 and then from ch_in1. In those cases, the ideal service time of the process is:

$$T_{id-P} \sim 22M\tau$$

The best-case analysis assumes that the process $P$ receives two consecutive stream elements from the same input channel. In that case, it can fully exploit reuse of the cache lines of the array passed as second argument to the function $F$, since all its cache lines are likely in C2/C1d. In this case, the process pays one miss for the first cache line of the first array passed to the function ($A$ if received from ch_in1, $B$ from ch_in2). This overhead is totally negligible. Therefore:

$$T_{id-P} \sim 12M\tau$$

To answers point b) we have to take care of the input rates of the two input streams to computes some probabilities that allow us to properly weight the worst-case and best-case situations. In one second the process receives 350 new arrays from the two channels. The probability to receive two consecutive messages from ch_in1 is:

$$p_{1-1} = \frac{250}{350} \cdot \frac{249}{349} = 0.51$$

The probability to receive two consecutive messages from the second input channel is:

$$p_{2-2} = \frac{100}{350} \cdot \frac{99}{349} = 0.09$$

The probability to receive from different channels is:

$$p_{1-2} = p_{2-1} = \frac{250}{350} \cdot \frac{100}{349} = 0.20$$

Therefore, we have:

$$T_{fault} \sim 0.40 \frac{M}{\sigma}L_{readc1} = 4M\tau$$

And the ideal service time of the process is:

$$T_{id-P} \sim 16M\tau$$