**Discussions:**

1. **Legal Periodic Temporal Schedule:**

From the **Allocation Table** with Jobs (**J1 - J6)** to derive a legal temporal scheduling, we invert the allocation table to get a **Scheduling Matrix**. The matrix thus obtained would look as below.

In the matrix, each grid corresponds to a particular Job’s **VP** as assigned to a Processor (**πx**) given by the Allocation Table. 

To prove that it is legal, we consider each time slab one at a time. For example, in **Time Slice 1**, all the Processors (**π1**, **π3** and **π4**) advances the **VP-1**, **VP-2** and **VP-3** of Job 1 by one time slice. In **Time Slice 2**, the Processor **π1**, advances the other **VP-4** of the same Job 1 by one time slice. Since it has been allotted two VPs of Job 1, incrementation in the second time slice, does not break the condition of **Legal Scheduling** as it does not increment the previously incrementing VP. The difference in advancement of any two VPs at any given time does not differ by more than 1 and thus adheres to the conditions of legal scheduling.

The **Schedule Vector** of Job J1 at the end of Time Slice 1 looks like, **ε(1, J1)** = (1, 0, 1, 1, 0) which corresponds to its **Progress Vector** **P(1, J1)** = (1, 0, 1, 1, 0). For Time Slice 2, the schedule vector can be represented as, **ε(2,J1)** = (1, 0, 0, 0, 0) and the Progress Vector then becomes **P(2, J1)** = (2, 0, 1, 1, 0).

The logic can be extended to other jobs across all time slices. Processor **π3** which has 2 VPs of Job 5 allotted to it, increment both by 1 in Time Slices 2 and 3 respectively and do not increment them further till Time Slice 6 by when Processor **π5** has incremented the VP5 and VP6 of Job-5 by 1. In none of the further time slices in the given period, any of the Processors advance any of the VPs in such a way that it is 2 steps ahead of the others. Thus, the scheduling can be proved to be **legal** although its efficiency is extremely on the lower side.

At the end of the 6th Time Slice, all the VPs of all the Jobs have been incremented exactly by 1. The Progress Vector for all the Jobs tally exactly with the Allotment Vectors and can be reset to 0. Here, we can say that one period is complete. Thus, **one** period would consist of **6 time slices**.

To calculate the **idling ratio**, we see that out of 30 slots in the given period, 9 slots were empty. Hence, for the above-mentioned scheduling matrix, the .

1. **Periodic Schedule with Improved Idling Ratio:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time/PE** | **π1** | **π2** | **π3** | **π4** | **π5** |
| **1** | 1 | 2 | 1 | 1 | 2 |
| **2** | 1 | 3 | 5 | 3 | 2 |
| **3** | 3 | 4 | 5 | 5 | 3 |
| **4** | 5 | 5 |  |  | 5 |
| **5** | 6 | 4 |  |  | 5 |
| **6** |  | 4 |  |  | 6 |

We can improve upon the Scheduling by adding two VPs of job 4 at the 5th and 6th time slice for the **π2 as** shown above which results in improved .

And as the Job 4 has only 1 Vp , hence doing so does not result in any Lock Step violation and maintains it’s legal scheduling status.

We can improve the Idling Ratio further by using a different allocation table as explained below.

1. **Temporal Schedule with Minimum Idling Ratio:**

But if we used the below allocation table we can achieve a scheduling with best 0% Idling Ratio.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 5 |
| 5 | 5 | 6 | 6 | 5 |
| 3 | 5 | 4 | 5 | 3 |
| 2 | 3 | 5 | 3 | 2 |
| 1 | 1 | 1 | 1 | 2 |
| **π1** | **π2** | **π3** | **π4** | **π5** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time/PE** | **π1** | **π2** | **π3** | **π4** | **π5** |
| **1** | 1 | 1 | 1 | 1 | 2 |
| **2** | 2 | 3 | 5 | 3 | 2 |
| **3** | 3 | 5 | 4 | 5 | 3 |
| **4** | 5 | 5 | 6 | 6 | 5 |
| **5** | **1** | **1** | **1** | **1** | 5 |

Here if we flip the new allocation table along the horizontal and add the another run for the VPs for Job 1 at the time slice 5 to fill the idle cells, we get the above Scheduling Matrix.

Below are the t-schedule vector and corresponding progress for each job.

Job 1: **A(J1) =** (1, 1, 1, 1, 0)

**P(0, J1)** **+ ε(1, J1)** = (0, 0, 0, 0, 0) + (1, 1, 1, 1, 0) = **P(1, J1)** = (1, 1, 1, 1, 0)

**A(J1) = P(1, J1) => P(2, J1) =** (0, 0, 0, 0, 0)

And then again at the time slice 5 we perform another run for Job 1 VPs

**P(4, J1)** **+ ε(5, J1)** = (0, 0, 0, 0, 0) + (1, 1, 1, 1, 0) = **P(5, J1)** = (1, 1, 1, 1, 0)

**A(J1) = P(5, J1) => P(6, J1) =** (0, 0, 0, 0, 0)

Job 2: **A(J2) =** (1, 0, 0, 0, 2)

**P(0, J2) + ε(1, J2)** = (0, 0, 0, 0, 0) + (0, 0, 0, 0, 1) = **P(1, J2)** = (0, 0, 0, 0, 1)

**P(1, J2)** + **ε(2, J2)** = (0, 0, 0, 0, 1) + (1, 0, 0, 0, 1) = **P(2, J2)** = (1, 0, 0, 0, 2)

**A(J2) = P(2, J2) => P(3, J2) =** (0, 0, 0, 0, 0)

Job 3: **A(J3) =** (1, 1, 0, 1, 1)

**P(1, J3)** **+ ε(2, J3)** = (0, 0, 0, 0, 0) **+** (0, 1, 0, 1, 0) = **P(2, J3)** = (0, 1, 0, 1, 0)

**P(2, J3)** + **ε(3, J3)** = (0, 1, 0, 1, 0) + (1, 0, 0, 0, 1) = **P(3, J3)** = (1, 1, 0, 1, 1)

**A(J3) = P(3, J3) => P(4, J3) =** (0, 0, 0, 0, 0)

Job 4: **A(J4) =** (0, 0, 1, 0, 0)

**P(2, J4)** **+ ε(3, J4)** = (0, 0, 0, 0, 0) **+** (0, 0, 1, 0, 0) = **P(3, J4)** = (0, 0, 1, 0, 0)

**A(J4) = P(3, J4) => P(4, J4) =** (0, 0, 0, 0, 0)

Job 5: **A(J5) =** (1, 2, 1, 1, 2)

**P(1, J5) + ε(2, J5) =** (0, 0, 0, 0, 0) + (0, 0, 1, 0, 0) = **P(2, J5) =** (0, 0, 1, 0, 0)

**P(2, J5) + ε(3, J5) =** (0, 0, 1, 0, 0) + (0, 1, 0, 1, 0) = **P(3, J5) =** (0, 1, 1, 1, 0)

**P(3, J5) + ε(4, J5) =** (0, 1, 1, 1, 0) + (1, 1, 0, 0, 1) = **P(4, J5) =** (1, 2, 1, 1, 1)

**P(4, J5) + ε(5, J5) =** (1, 2, 1, 1, 1) + (0, 0, 0, 0, 1) = **P(5, J5) =** (1, 2, 1, 1, 2)

**A(J5) = P(5, J5) => P(6, J5) =** (0, 0, 0, 0, 0)

Job 6: **A(J6) =** (0, 0, 1, 1, 0)

**P(3, J6) + ε(4, J6) =** (0, 0, 0, 0, 0) + (0, 0, 1, 1, 0) = **P(4, J6) =** (0, 0, 1, 1, 0)

**A(J6) = P(4, J6) => P(5, J6) =** (0, 0, 0, 0, 0)

Here as we have seen that at each step the Legal execution of a job is performed as all VPs of a job have executed the same number of time slices(2 for VPs of Job1 and 1 VPs of other Jobs), and No period starts with some VP of some job being ahead of its peers of the same job by one time slice.

Furthermore in this we do not have any idle cycle hence it’s the best Scheduling Matrix for this Allocation configuration.