**PROJECT 3 REPORT**

**Experimental Setup**

In our experiment we ran the exact same function that makes use of the resource allocation multiple times for each deadlock handling method. Our data consists of our independent variables and execution times.

We had three resource types with 100 instances of each. The independent variable *N* is the number of threads accessing those resources. Each thread makes two requests, waits a small amount of time and releases all resources. These requests are randomized through the use of *rand()*. However, because no seed is set, the program works the same way every time. Finally, the independent variable *K* denotes the ratio of existing resource amount to maximum possible demand amount for each resource type for each thread. Setting *K >= N* ensures that no deadlock is possible. This way, we can test the overhead isolated from other factors.

Experiment code can be found in the submission under the name *experiment.c* .

**Data**

|  |  |  |
| --- | --- | --- |
| **Avoidance** | **Nothing** | **Detection** |
| 728 | 597 | 590 |
| 783 | 544 | 572 |
| 789 | 606 | 593 |
| 707 | 563 | 588 |
| 786 | 550 | 598 |

*Table 1. Execution time in microseconds for N = 5 and K = 10*

|  |  |  |
| --- | --- | --- |
| **Avoidance** | **Nothing** | **Detection** |
| 1645 | 786 | 893 |
| 1796 | 708 | 997 |
| 1830 | 761 | 1188 |
| 1673 | 727 | 783 |
| 1562 | 857 | 860 |

*Table 2. Execution time in microseconds for N = 10 and K = 10*

**Interpretation and Conclusions**

Between the handling methods we see that *avoidance* takes the longest; *nothing* and *detection* is generally close with *detection* taking longer most of the time. It is expected that *avoidance* would take the longest, because with each request the safety algorithm is run and the other methods do not have as much overhead. *Detection* has more overhead than *nothing*, because it calls the detection method while *nothing* has no extra calls. Therefore *detection* takes longer than *nothing* in general. However, their difference is not as defined as their differences with *avoidance*. Because, the number of times the detection method is called is very small compared to the number of requests made. Moreover, threads finish relatively quickly so error from scheduling has a considerable effect.

The difference between two experiments is the execution times. In the second experiment each thread demands a higher portion of the resources. Hence, contention increases leading to increased execution times. As predicted from the previous paragraph, increased execution times diminishes error from scheduling and also, allows more detection methods to be run. Thus, difference between *nothing* and *detection* becomes more defined.

In summary,

* In execution time: *avoidance > detection > nothing.*
* *Avoidance* has the highest overhead because each request comes with a safety check, unlike others.
* As execution time increases the difference between *detection* and *nothing* increases, because more detection calls can be made.