| | |
|---|---|
| **Started on** | Friday, 26 April 2024, 9:21 PM |
| **State** | Finished |
| **Completed on** | Friday, 26 April 2024, 9:22 PM |
| **Time taken** | 1 min 14 secs |
| **Grade** | **2.00** out of 2.00 (**100**%) |

**Question 1**

Correct

Mark 1.00 out of 1.00

In parallel computing, an embarrassingly parallel workload or problem (also called embarrassingly parallelizable, perfectly parallel, delightfully parallel or pleasingly parallel) is **one where little or no effort is needed to separate the problem into a number of parallel tasks**.

A code implementing the Jacobi method, where the data used as input is not modified throughout the computation of the code is embarrassingly parallelizable. However, a code implementing the Gauss-Seidel method in which the data gets modified throughout the computation of the code and is therefore used for both input and output, is not embarrassingly parallelizable since loop carried dependencies exist and they need to be respected in order to preserve correctness.

Select one:

- True ✔
- False

Well done!

The correct answer is 'True'.

**Question 2**

Correct

Mark 1.00 out of 1.00

Let's assume that we want to execute the Jacobi and Gauss-Seidel implementations seen in the course on a distributed memory machine.

We implement a row-wise distribution of the matrix to P processors where each processor gets n/P consecutive rows;

Which of the following statements are true?

- [ ] a. For Jacobi: Task definition = Block of n/P consecutive rows, i.e. each processor executes a single (coarse grain) task, sequentializes the execution. Therefore one needs to apply blocking in order to exploit parallelism.
- [x] b. For Gauss-Seidel, if the matrix was distributed among the processors by columns so that each processor gets a set of n/P consecutive columns, then we would need to apply blocking to the rows, so that each task works on a submatrix of size R rows by n/P columns. ✔ Exactly: we need to apply blocking to the other dimension.
- [x] c. For Gauss-Seidel: Task definition = Block of n/P consecutive rows, i.e. each processor executes a single (coarse grain) task, sequentializes the execution. ✔ True. Execution time will be worse than that of the original sequential code due to the overheads.
- [x] d. For Gauss-Seidel with blocking, instead of creating a single task per processor one creates several tasks per processor, where each task works on a submatrix of size n/P rows by C columns. ✔ Exactly!
- [x] e. For Gauss-Seidel one needs to apply blocking in the columns in order to exploit wave-front parallelism. ✔ True!

Your answer is correct.

The correct answers are:
For Gauss-Seidel: Task definition = Block of n/P consecutive rows, i.e. each processor executes a single (coarse grain) task, sequentializes the execution.,

For Gauss-Seidel one needs to apply blocking in the columns in order to exploit wave-front parallelism.,

For Gauss-Seidel with blocking, instead of creating a single task per processor one creates several tasks per processor, where each task works on a submatrix of size n/P rows by C columns.,

For Gauss-Seidel, if the matrix was distributed among the processors by columns so that each processor gets a set of n/P consecutive columns, then we would need to apply blocking to the rows, so that each task works on a submatrix of size R rows by n/P columns.