How do we write parallel programs?

Wednesday, February 7, 2024 6:48 PM

- Task parallelism
  - Partition various tasks carried out solving the problem among the cores.
- Data parallelism
  - Partition the data used in solving the problem among the cores.
  - Each core carries out similar operations on it's part of the data.

Example of both:

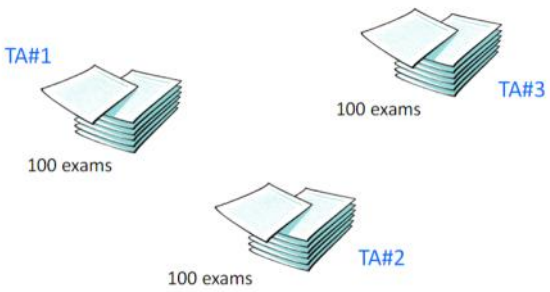
Professor P: A problem I can relate to

15 questions
300 exams
3 TAs

Professor P's grading assistants

Division of Work - data parallelism

Division of work – data parallelism

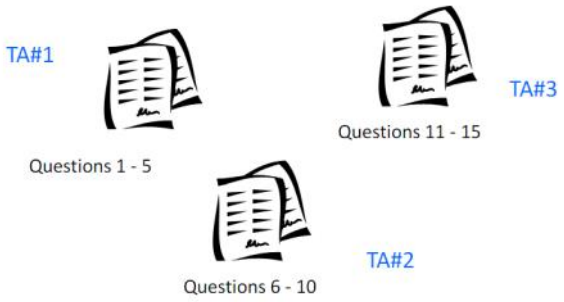


Division of work – data parallelism

```
sum = 0;
for (i = 0; i < n; i++) {
   x = Compute_next_value(. . .);
   sum += x;
}
```

Division of Work - task parallelism

Division of work – task parallelism



Division of work – task parallelism

```
If (I'm the master core) {
   sum = my_x;
   for each core other than myself {
      receive value from core;
      sum += value;
   }
} else {
   send my_x to the master;
}
```

Tasks

1) Receiving
2) Addition

Coordination

- Cores usually need to coordinate their work.
- Communication – one or more cores send their current partial sums to another core.
- Load balancing – share the work evenly among the cores so that one is not heavily loaded.
- Synchronization – because each core works at its own pace, make sure cores do not get too far ahead of the rest.

Notes:

- Fundamentals of parallelism