Parallelization of Scala Programs by Refactoring

Artúr Poór

Eötvös Loránd University
Budapest, Hungary

August 2016

Motivation

- Multi-cores, multi-cores everywhere
- Heterogeneous systems
- Parallel thinking
  - High-level programming constructs
  - Deadlocks are eliminated
  - Communication is abstracted
- Restructure existing code

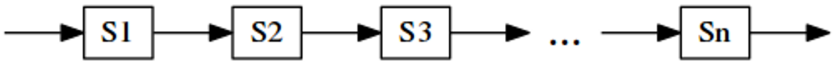
Pattern-based parallelism

High-level approach to parallel programming

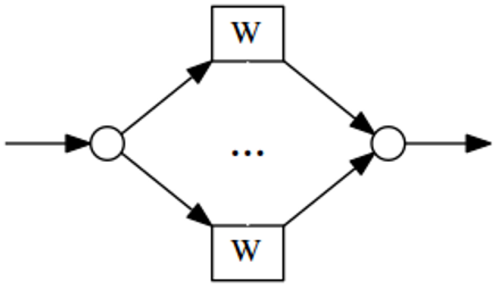
- Rely on a library of algorithmic skeletons
- Easier to develop code
- Easier to modify, maintain
- Better utilization of resources
  - Dynamic (re)mapping on heterogeneous hardware

Parallel patterns

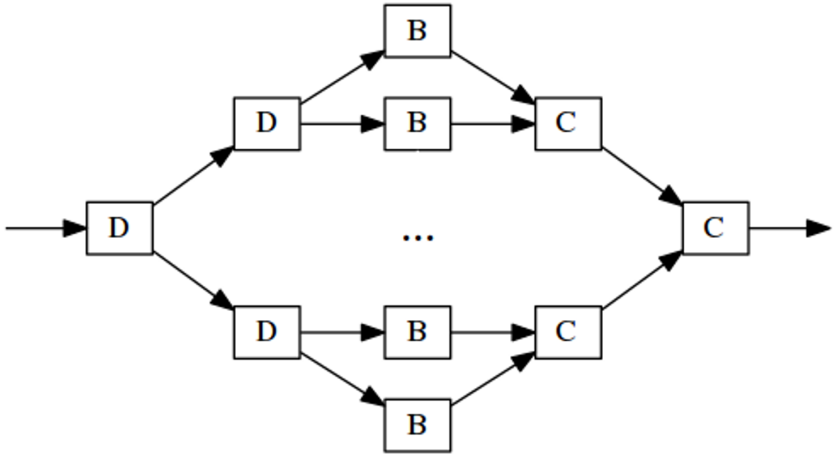
Pipeline



Farm



Divide & conquer



Refactoring programs

How to make programs parallel?

- Refactor
  - Using a tool
  - Guided, semi-automatic transformations
- Benchmark, validate
- Repeat

Where to make programs parallel?

- Find candidates automatically
  - Using a tool
  - Requires static program analysis

Example: parsing modules

Before

```
for (module <- modules)
  yield parse(scan(read(module)))
```

After

```
Farm(Pipe(parse _) compose (scan _) compose (read _)),
     5,
     modules)
```