CS240H Final Project: MapReduce in Haskell

By Arpad Kovacs

Overview

- What is MapReduce?
- Single-node implementation
- Distributed implementation
- Challenges, future work

What is MapReduce?

- Parallel, distributed programming model introduced by Dean and Ghemawat from Google [OSDI 2004]

- Inspired by functional programming (Map and Reduce aka Fold functions).

Mapper

- Map: processes input key-value tuple, outputs intermediate key-value tuples.

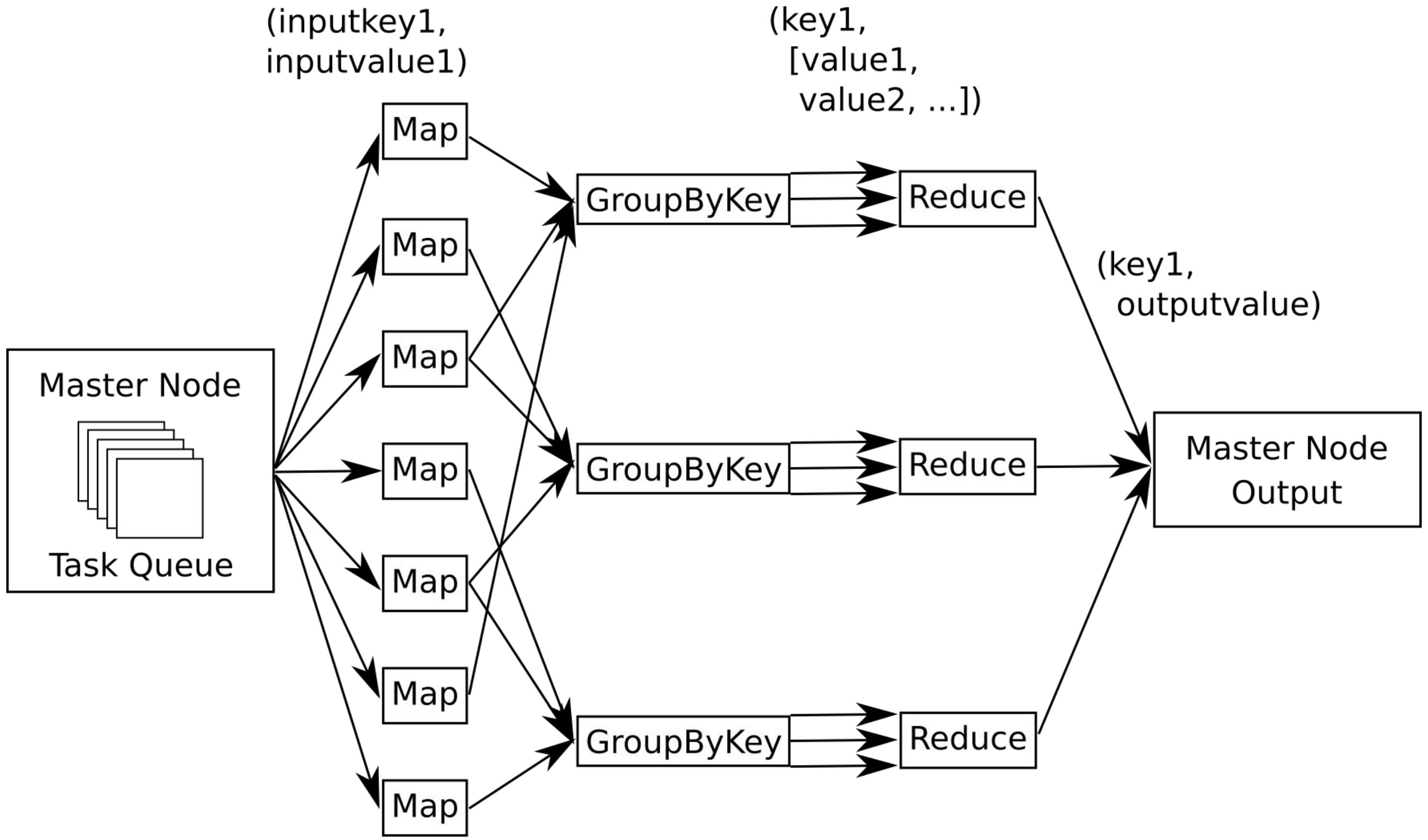
```
type Mapper inputKey inputValue
intermediateKey intermediateValue =
(inputKey, inputValue) ->
[(intermediateKey, intermediateValue)]
```

Reducer

- Reduce: takes an intermediate key and list of intermediate values, and folds those values into single output value.

```
type Reducer reduceKey reduceValue =
reduceKey -> [reduceValue] -> reduceValue
```

MapReduce Data Flow



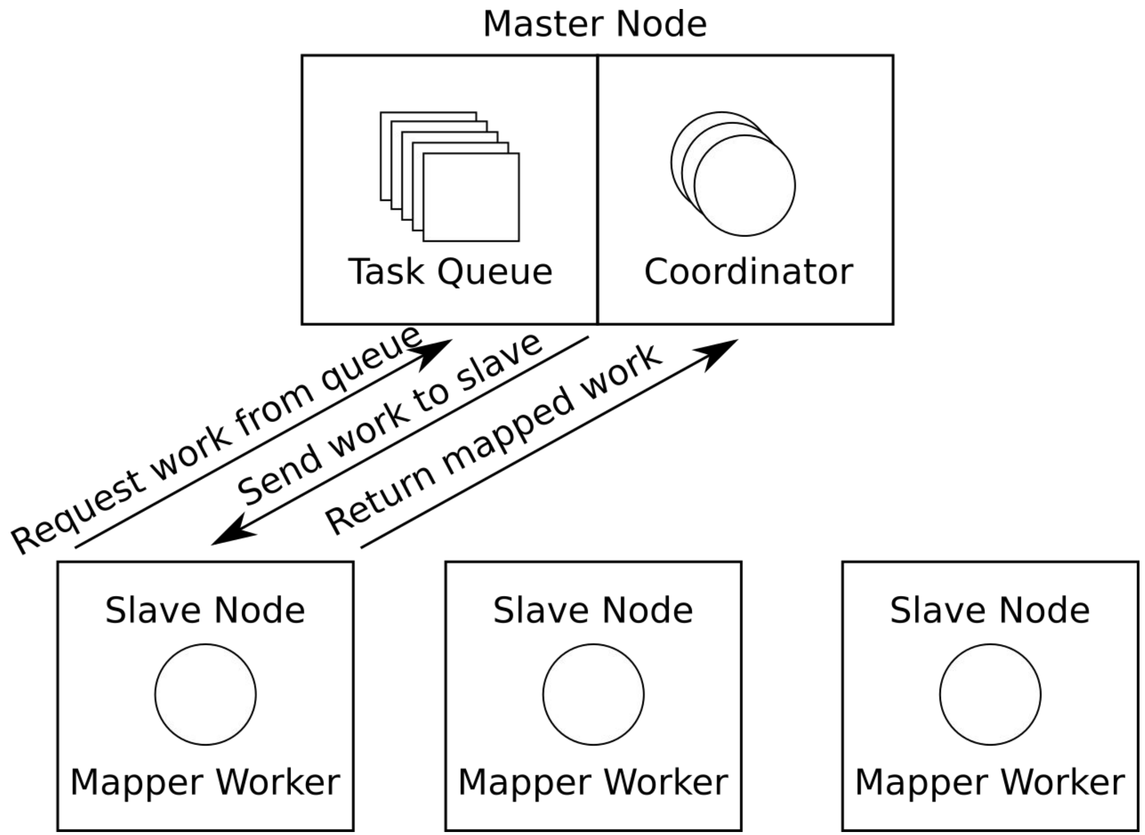
Demo - Word Count

```
mapReduce :: (Ord reduceKey)
=> Mapper inputKey inputValue reduceKey reduceValue
-> Reducer reduceKey reduceValue
-> Map.Map inputKey inputValue
-> Map.Map reduceKey reduceValue
mapReduce mapper reducer = reduce reducer . groupByKey . asList . mapInput
mapper
where mapInput mapper = concatMap mapper . Map.toList
asList keyValuePairs = [(key, [value]) | (key, value) <- keyValuePairs]
groupByKey = Map.fromListWith (++)
reduce reducer = Map.mapWithKey reducer
```

Distributed Implementation

- Execute Mappers in parallel on a shared-nothing architecture.
- Reduce operation is associative
- Used Distributed-Process (actor-based message-passing framework)

Distributed Architecture



Citations

Github repo: https://github.com/akovacs/cs240h-project

Mapreduce: Simplified data processing on large clusters [OSDI 2004]

Towards Haskell in the cloud [4th ACM Symposium on Haskell, 2011]