MapReduce API

Ben Withbroe and Charlie Imhoff

What is MapReduce?

- Paradigm for big data processing across distributed systems

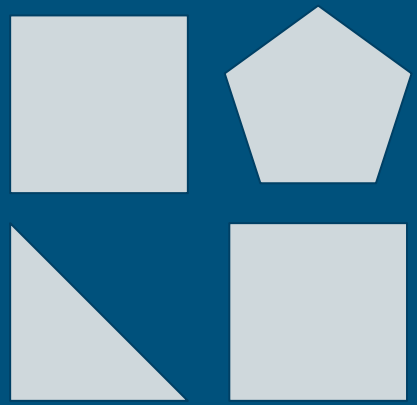
- `map(data)` `--> transformed_data`
- `reduce(data)` `--> result`

What is MapReduce?

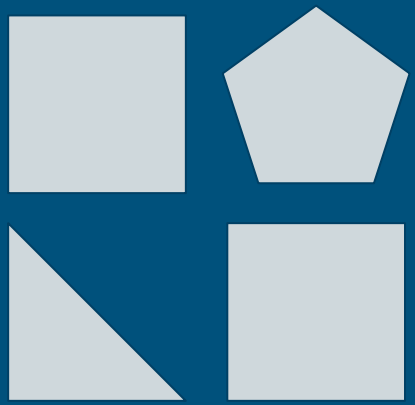
- Paradigm for big data processing across distributed systems

- `map(data, function) --> transformed_data`
- `reduce(data, function) --> result`

Map



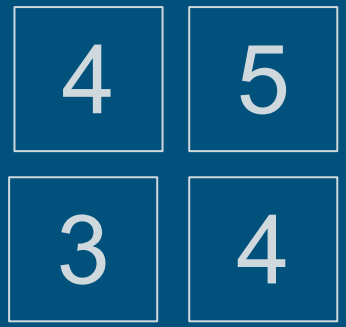
Map



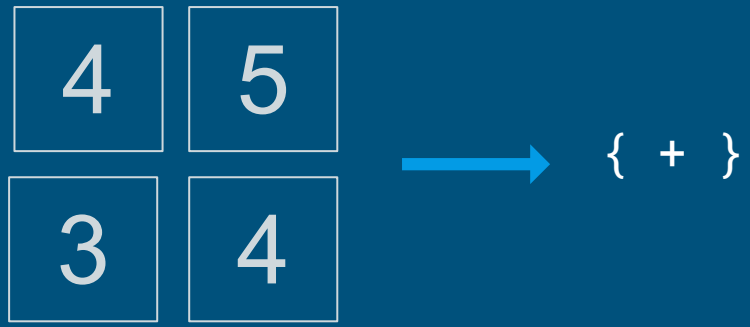
```
{ shape.sides.count }
```

Map

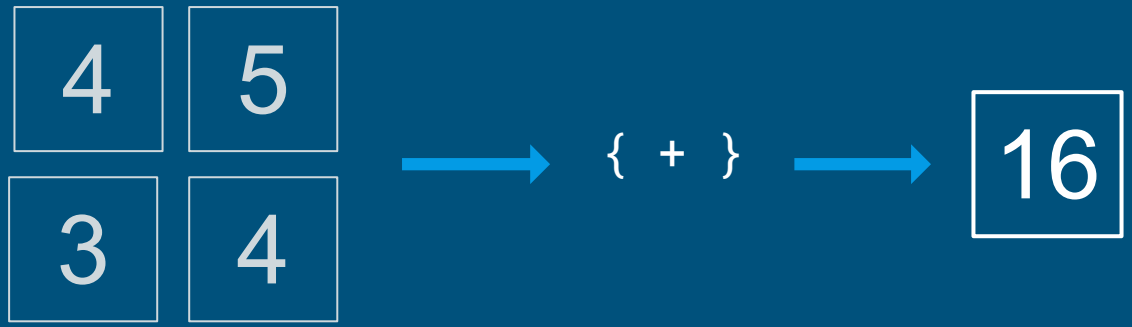
Reduce



Reduce



Reduce

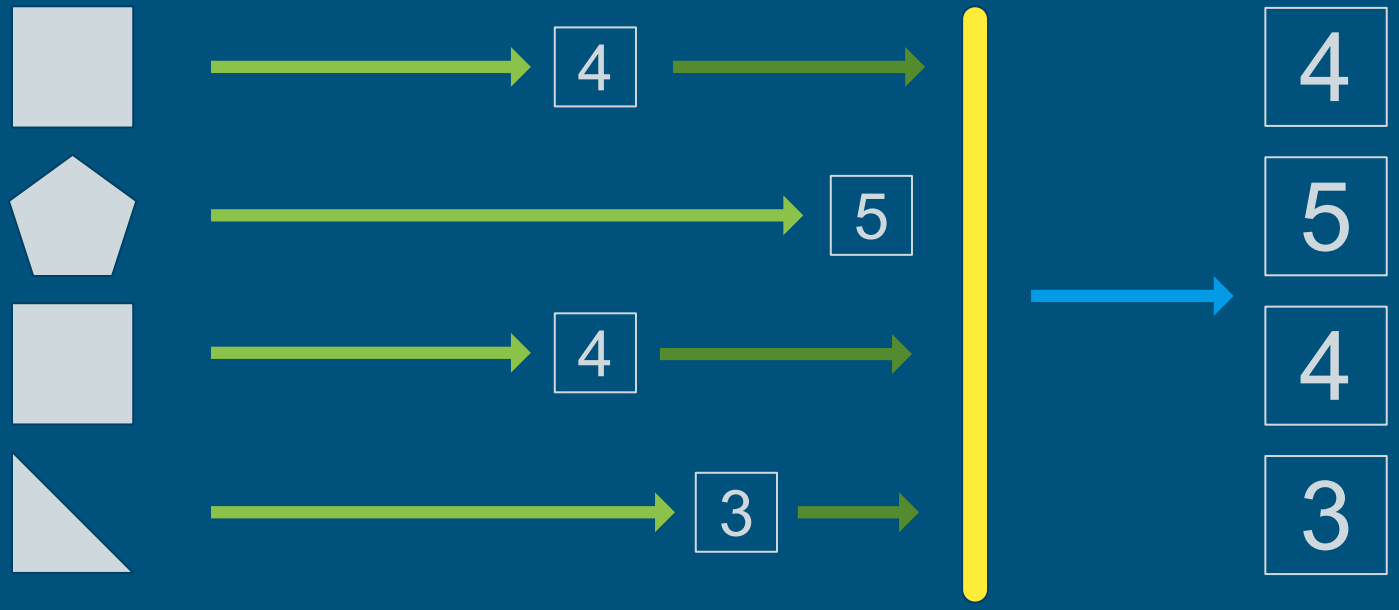


Asynchronous MapReduce

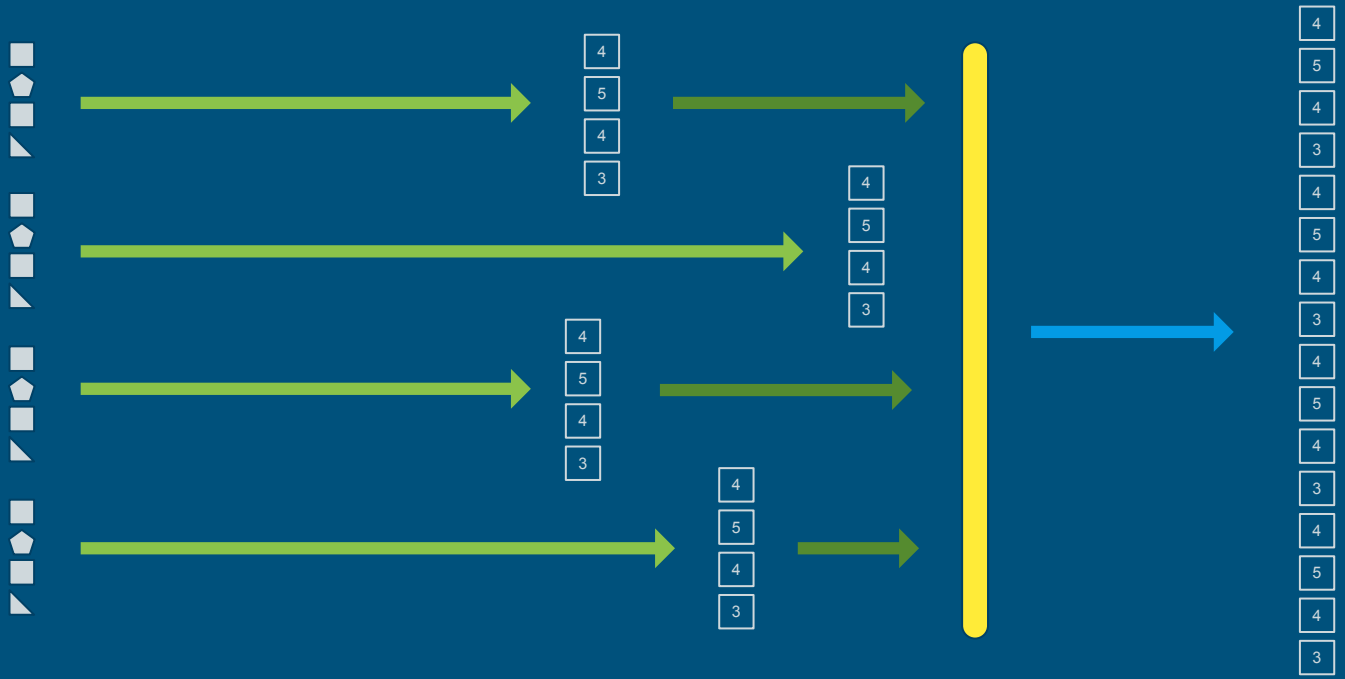
- Lack servers
- Lack networking talents
- Multi-Core machines

- ***Could parallelizing MapReduce introduce speed improvements?***

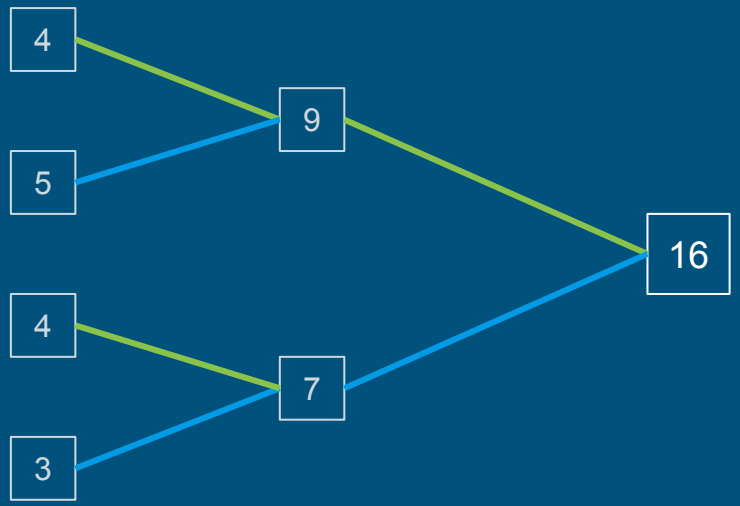
Map (parallel)



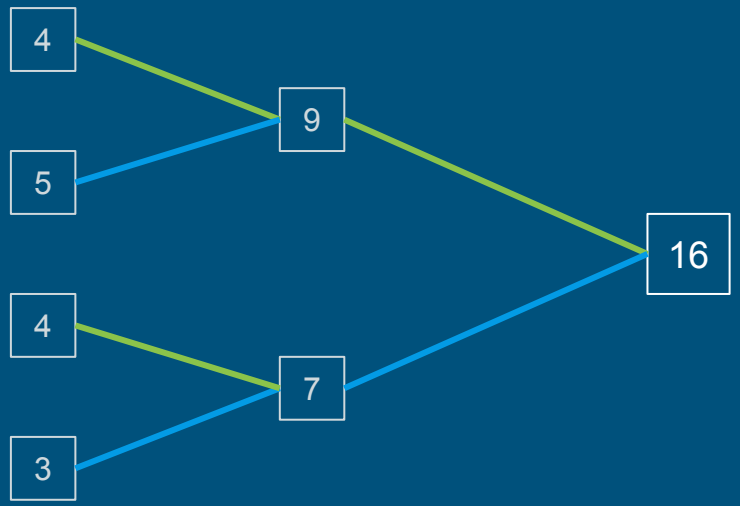
Map (parallel *chunked*)



Reduce (parallel)



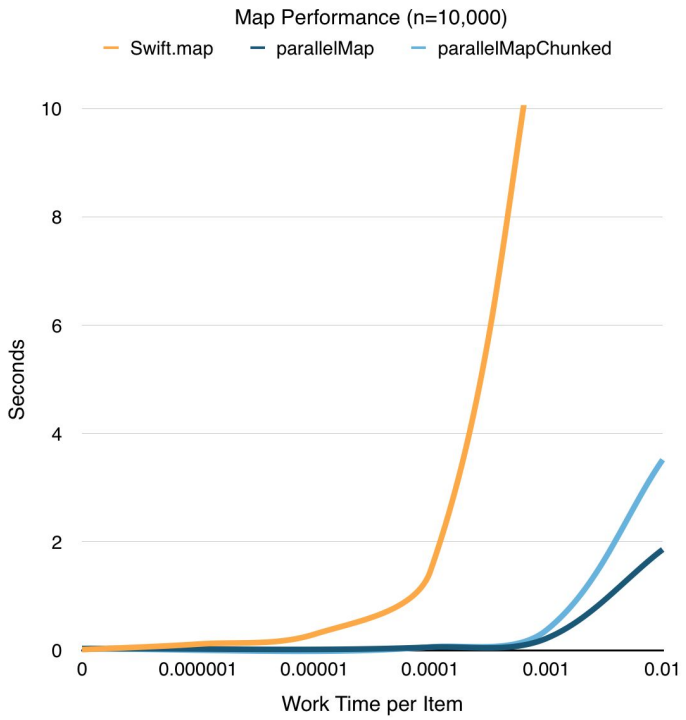
Reduce (parallel)

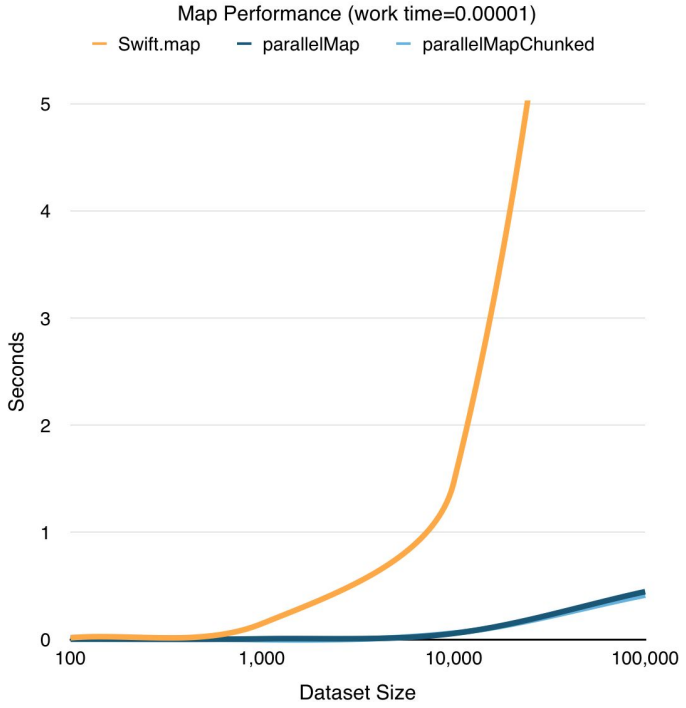


- *Dependencies!*
- Limited Threads
- Deadlocks

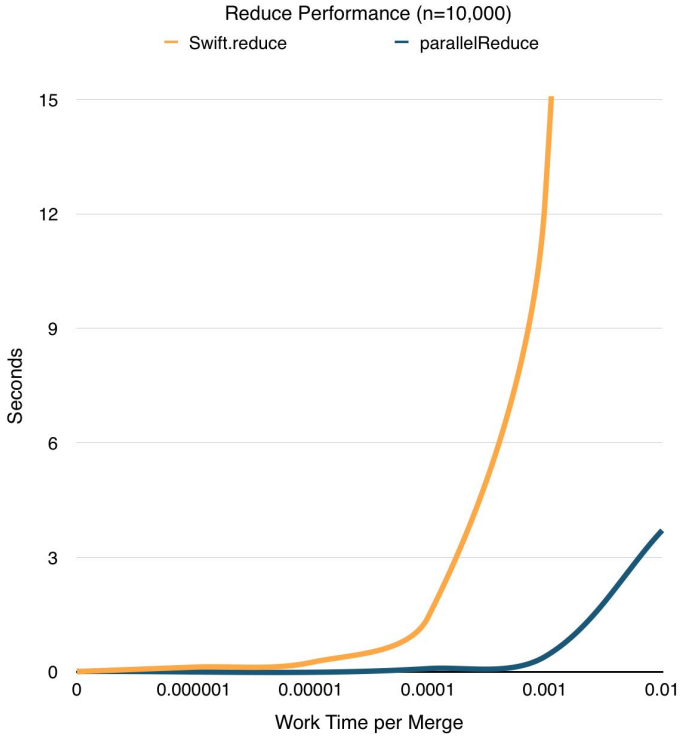
- Much more careful about allocating work on new threads
- Chunked by default

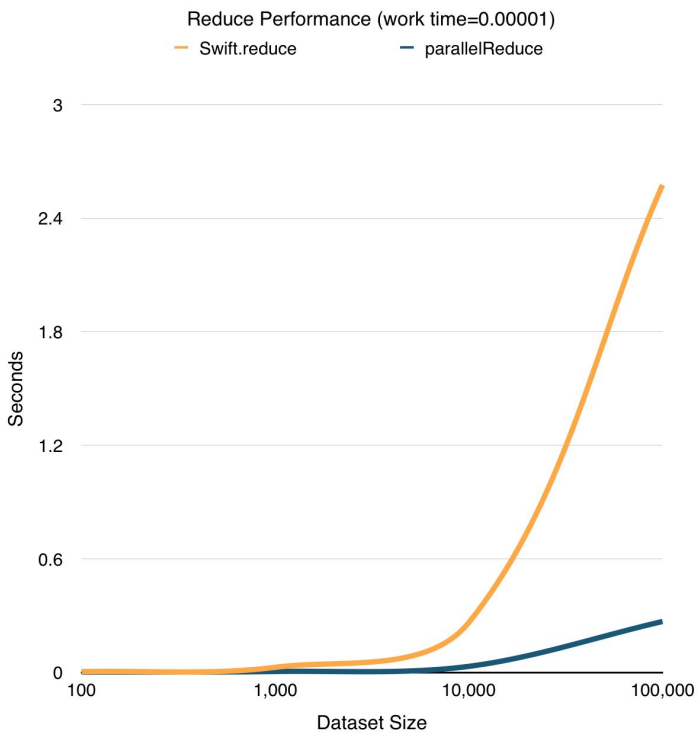
Unit Testing Performance: *Map*





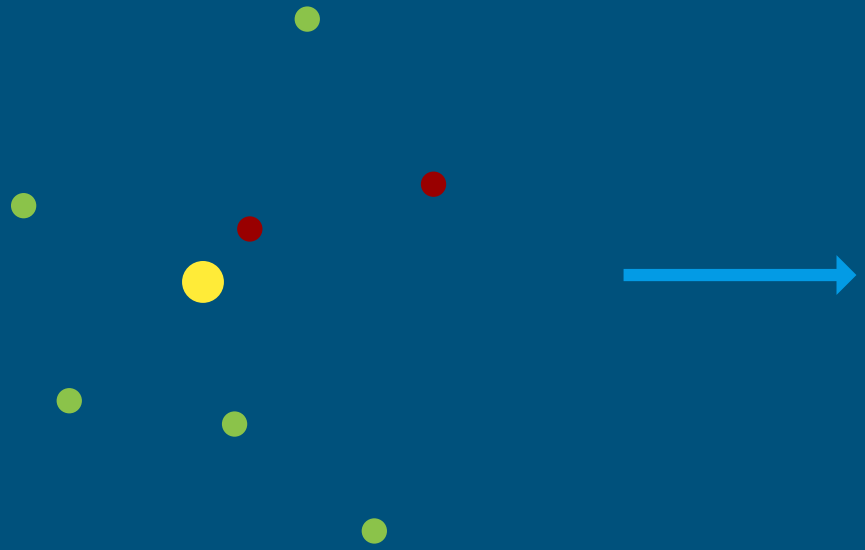
Unit Testing Performance: *Reduce*





MapReduce for kNN

Map for kNN

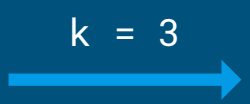


```
[ (green, 7.2),
  (green, 4.2),
  (green, 3.5),
  (red,   6.9),
  (green, 5.6),
  (red,   0.9),
  (green, 7.5) ]
```

Reduce for kNN

```
[ (green, 7.2) ],

[ (green, 4.2) ]
```

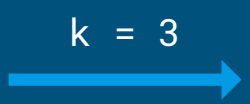



```
[ (green, 4.2),
  (green, 7.2) ]
```

Reduce for kNN

```
[ (green, 4.2),
  (green, 7.2) ]

[ (green, 3.5),
  (red,   6.9) ]
```



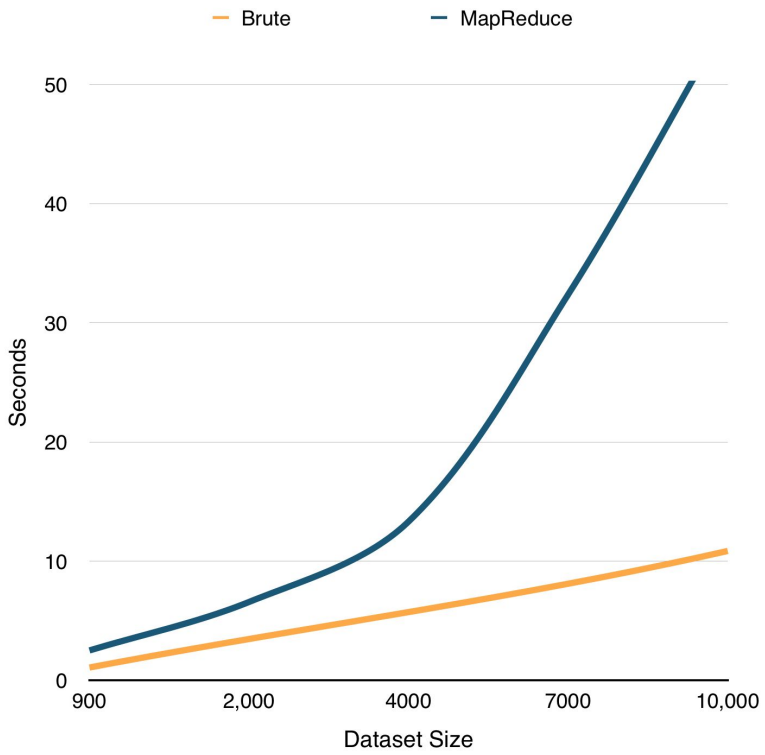

```
[ (green, 3.5),
  (green, 4.2),
  (red,   6.9) ]
```

Reduction Cleanup for kNN

```
[ (red,   0.9),
  (green, 3.5),
  (green, 4.2) ]
```

```
green
```

kNN Performance



Why is MapReduce garbage for kNN?

- MapReduce is effective when the map, reduce functions are non-trivial
- We are not splitting the data efficiently
  - Map function only works for a single test point

Future Directions

- Write a MapReduce function tailored to kNN
  - Our generic map function only operates on a single (or batched) test points at once
  - Can’t work with point to point intersections cleanly
- Find another algorithm that would be more suited to this framework
- Apply the framework to problems with longer work times
  - Our implementation works best when per iteration work is non-trivial
  - Networking contexts… wait isn’t that where MapReduce comes from?