Akka.NET

Aufgabe 7: Datenmengen beherrschen mit Map Reduce

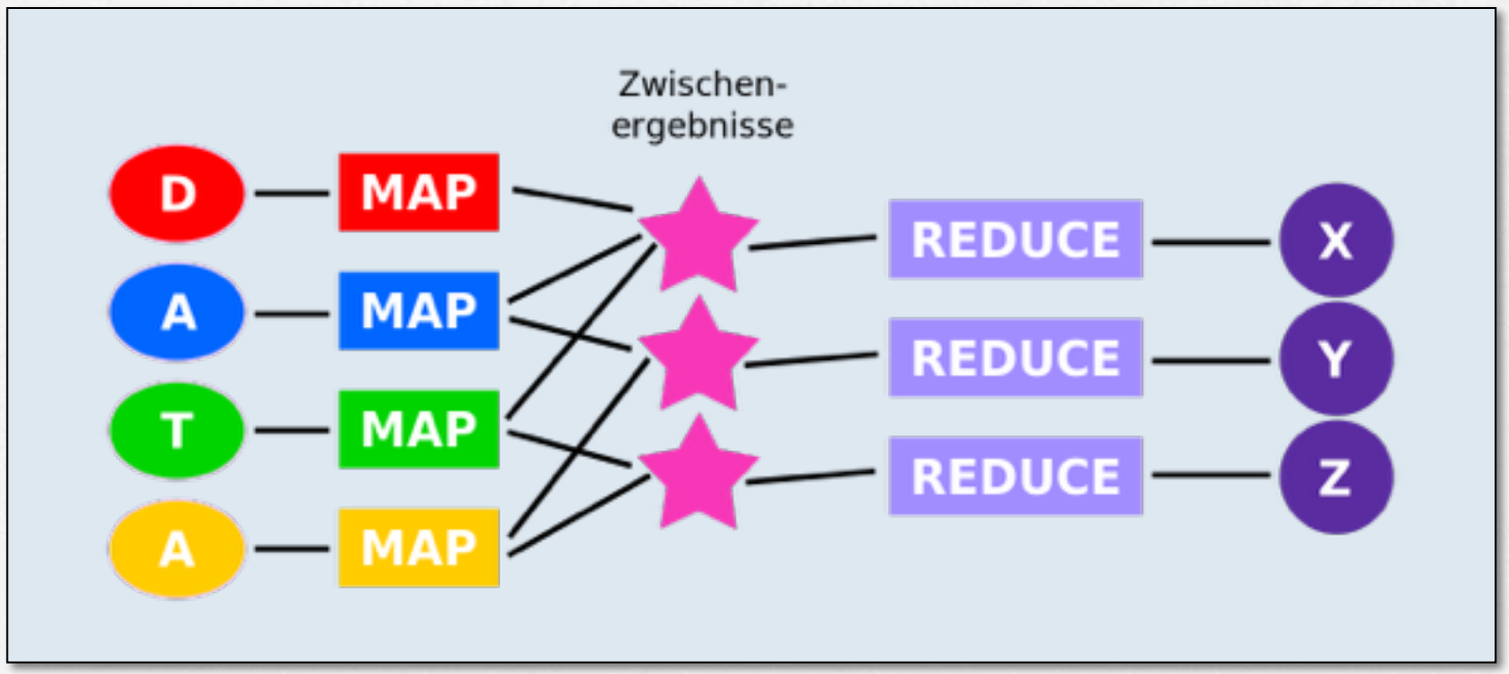
Die Problemstellung

- große umfangreiche Datenliste
  Firma, Entwickler, Programmiersprache
- zu ermitteln:
  Nennungen pro Programmiersprache
- evtl. Millionen Datensätze

```
ACME, Inc   | Mr. Green | C#
ACME, Inc   | Mr. Blue  | JavaScript

Facilities | Coolman   | Ruby

FastCoders | Speedy    | C#

SlowCoders | Snail     | Java
```

Google's Idee: map reduce

- Map: Daten in Tupel konvertieren
  - („C#“, 1)
- Reduce: nach Schlüssel aufsummieren
  - („C#“, 42)

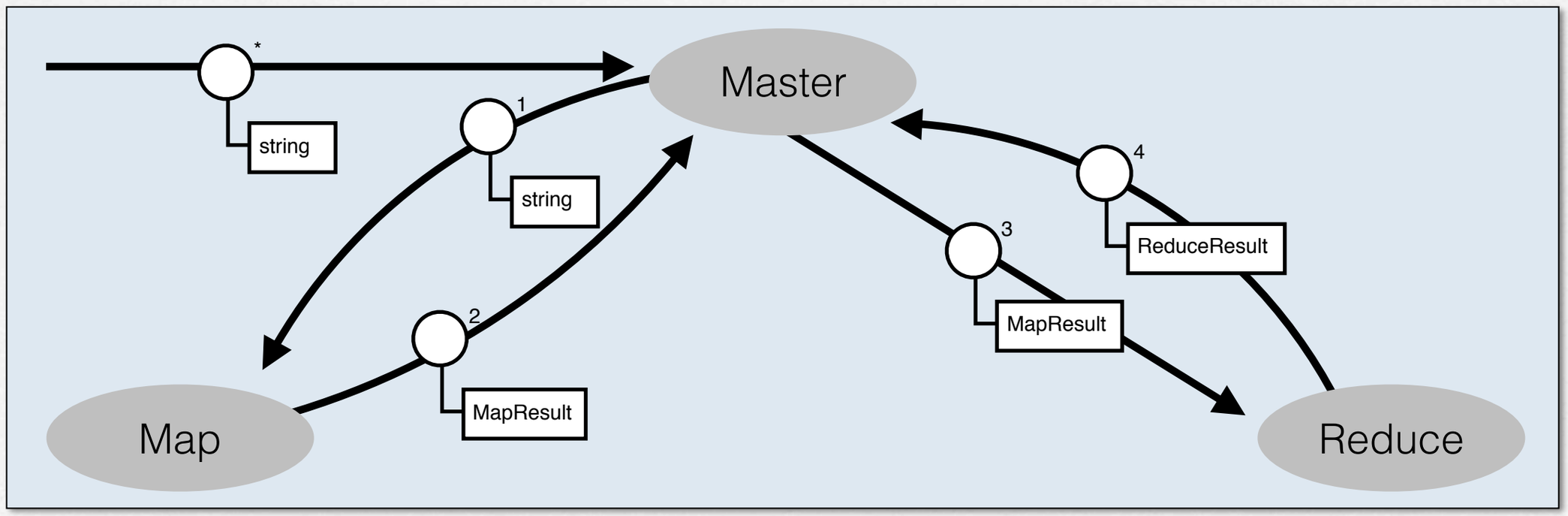


Einsatzgebiete

- Word Count, Top-N
- Summen, Minima, Maxima pro Periode
- Join über mehrere Datenquellen
- gemeinsame Freunde
- Kunden kauften auch
- ...

Datenfluss

- Master koordiniert alles
- Map: Tupel erzeugen
- Reduce: Kondensieren



Map

- Text spaltenweise bearbeiten
- Letzte Spalte (Sprache) wird zu Tupel

```
public class KeyCount
{
    public string Key { get; set; }
    public int Count { get; set; }

    public KeyCount(string key, int count = 1)...
}
```

```
public class Mapper : ReceiveActor
{
    public Mapper()
    {
        Receive<string>(s => Map(s));
    }

    private void Map(string input)
    {
        var mapResult = new MapResult();

        using (var reader = new StringReader(input))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                if (!String.IsNullOrWhiteSpace(line))
                {
                    var key = line.Split(new [] { '|' }).Last().Trim();

                    mapResult.Counts.Add(new KeyCount(key));
                }
            }
        }

        Sender.Tell(mapResult);
    }
}
```

Reduce

- Anzahl der Vorkommen des Schlüssels zählen
- Datenmenge: Anzahl Schlüssel

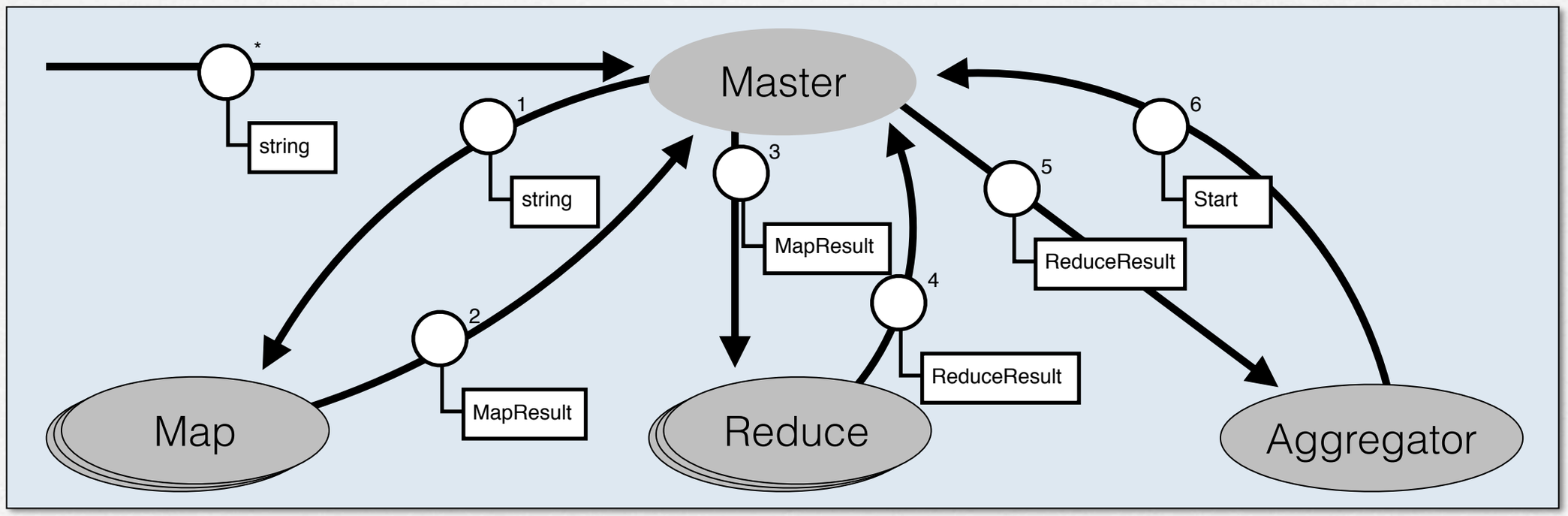
```
public class Reducer : ReceiveActor
{
    public Reducer()
    {
        Receive<MapResult>(m => Reduce(m));
    }

    private void Reduce(MapResult mapResult)
    {
        var reduceResult = new ReduceResult();

        foreach (var count in mapResult.Counts)
        {
            if (reduceResult.Result.ContainsKey(count.Key))
            {
                reduceResult.Result[count.Key] += count.Count;
            }
            else
            {
                reduceResult.Result[count.Key] = count.Count;
            }
        }

        Sender.Tell(reduceResult);
    }
}
```

Parallelisierung

- Quelldaten lassen sich aufteilen
- mehrere Map Prozesse (parallel, verteilt)
- mehrere Reduce Prozesse
- Resultate müssen nochmals aggregiert werden

Parallel Datenfluss

- Master koordiniert alles
- alle Schritte parallelisiert
- Aggregator sammelt Reduce zusammen



Projekt „MapReduce“

- leider nur Fake-Daten
- Anzahl Mapper/Reducer steuerbar
- Beobachtung bei wenig oder einem Map/Reduce Schritt
- Aufgabe: Anzahl Firmen anstelle Anzahl Sprachen zählen. Welcher Schritt muss verändert werden?

Lösungsvorschlag

- Mapper angepasst
- Rest bleibt :-)

```
public class Mapper : ReceiveActor
{
    public Mapper()
    {
        Receive<string>(s => Map(s));
    }

    private void Map(string input)
    {
        var mapResult = new MapResult();

        using (var reader = new StringReader(input))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                if (!String.IsNullOrWhiteSpace(line))
                {
                    var key = line.Split(new [] { '|' }).First().Trim();

                    mapResult.Counts.Add(new KeyCount(key));
                }
            }
        }

        // simulate some runtime...
        Thread.Sleep(50);

        Sender.Tell(mapResult);
    }
}
```