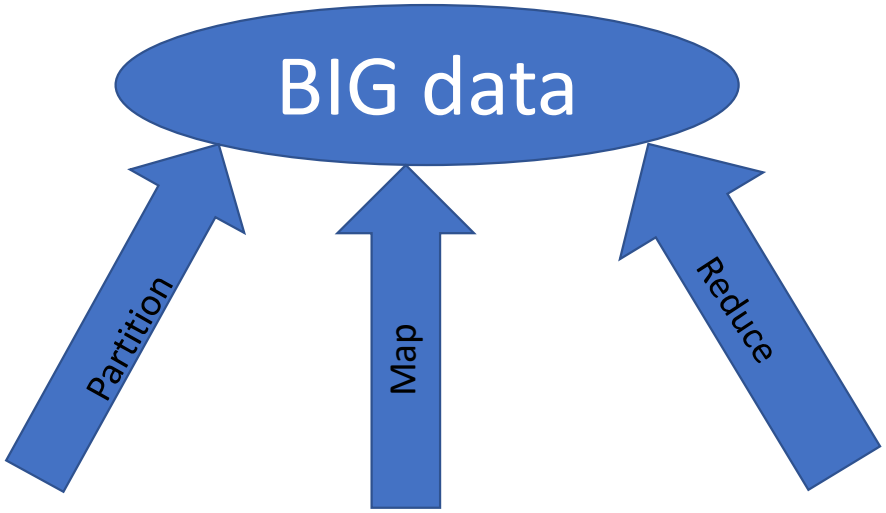
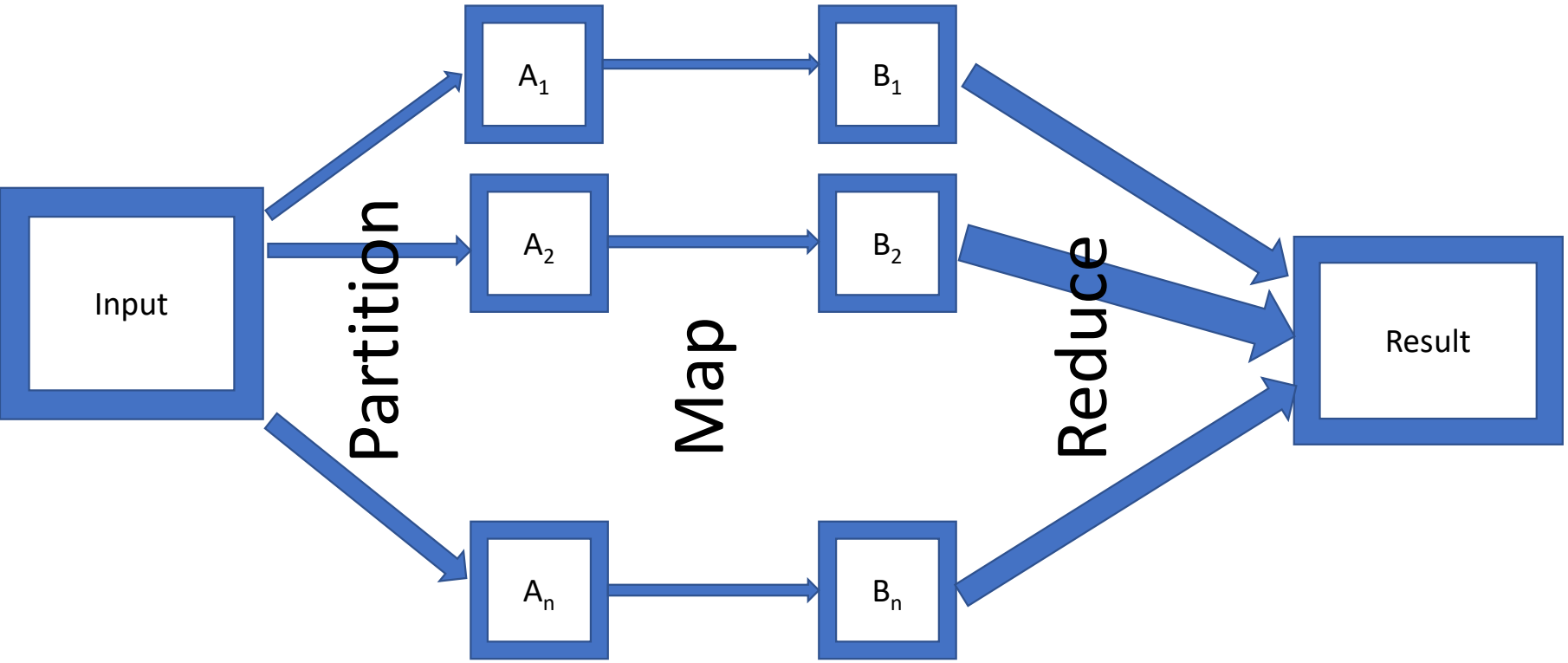
What is BIG data?

- Not solvable by “normal” DB (Oracle etc)
- Not solvable by “distributed” DB (mongoDB, Cassandra)
- “Lax” notions of transactions and availability.
- Fits a certain model (map-reduce) of problems



Three legs of the BIG data stool

Partition, Map (Parallel), Reduce (Concurrent)



Partition

Partitions the “work” into pieces

Map (Math function, especially for Spark)

f(x) => y

Reduce (Math function, especially for Spark)

Reduces a collection to a single value

Associate operation (like addition, multiplication, min, max)

f(y1, y2) => z

f(y2, y1) => z

Simplest MapReduce (Do Nothing)

Map: f(x) => x

Reduce: f(y1, y2) => [y1, y2]

Count # of lines

Map: f(x) => 1

Reduce: f(y1, y2) => y1 + y2

Count lines with Colorado

Map: f(x) => 1 if(line contains Colorado)

=> 0 otherwise

Reduce: f(y1, y2) => y1 + y2

Count lines for each State

Map: f(x) => (Colorado, 1) if(line contains Colorado)

=> (Alaska, 1) if(line contains Alaska)

.

.

. And so on

Reduce**ByKey**: f(y1, y2) => Sum by Key (State), (Repartition is implied)

Produces Grouping behavior. Can implement org based grouping

Hadoop (first generation)

Maximum **ONE** of

Partition

Map

Reduce

Spark (2nd generation)

Arbitrary number of

Partition
Map
Reduce

Chained in a pipeline.

Allows for partition optimization (reduces network traffic between workers)
Very hard problem. At the level of writing an OS.

Spark Streaming

Spark behavior applied to a “units of work” in time.