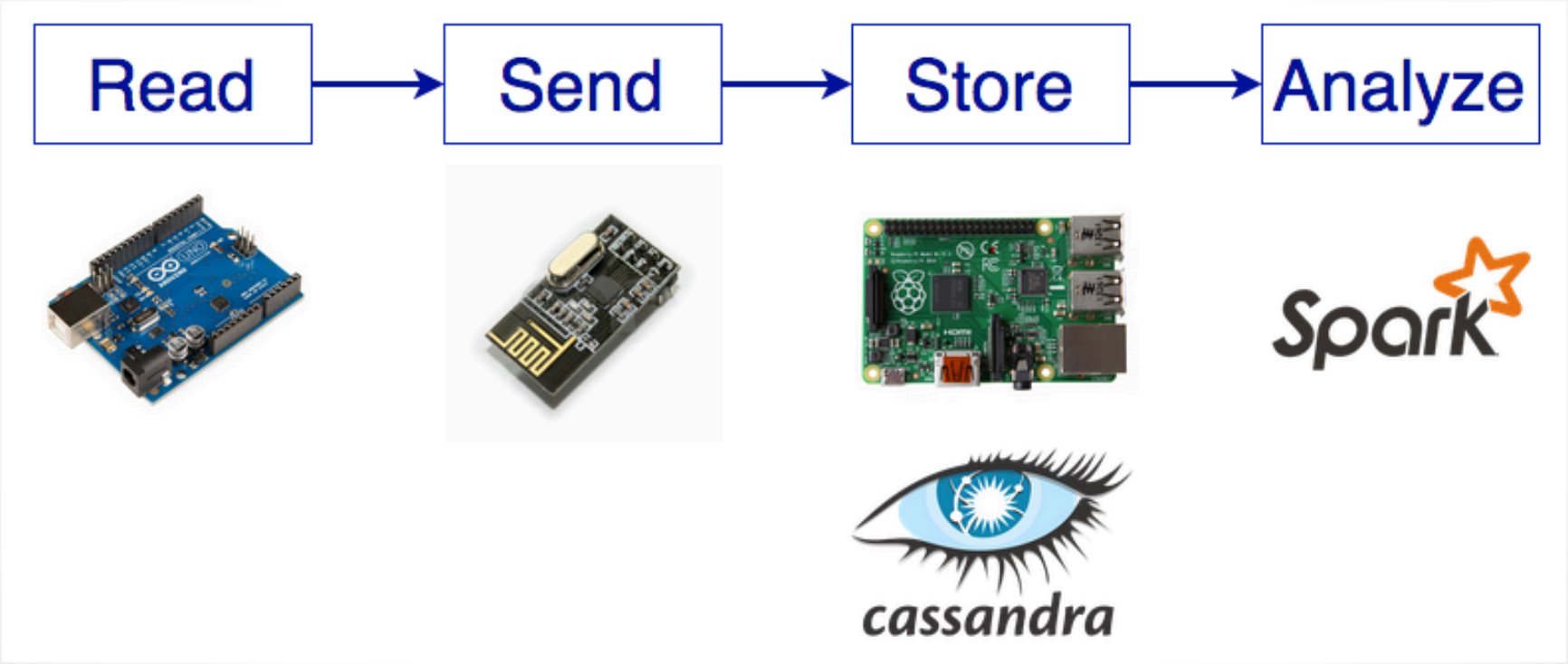
Big Data in a DIY home

Marko Švaljek
Kapsch CarrierCom d.o.o.

About me

- Marko Švaljek
- Kapsch CarrierCom d.o.o.
- Cassandra Succinctly
- Arduino Succinctly *

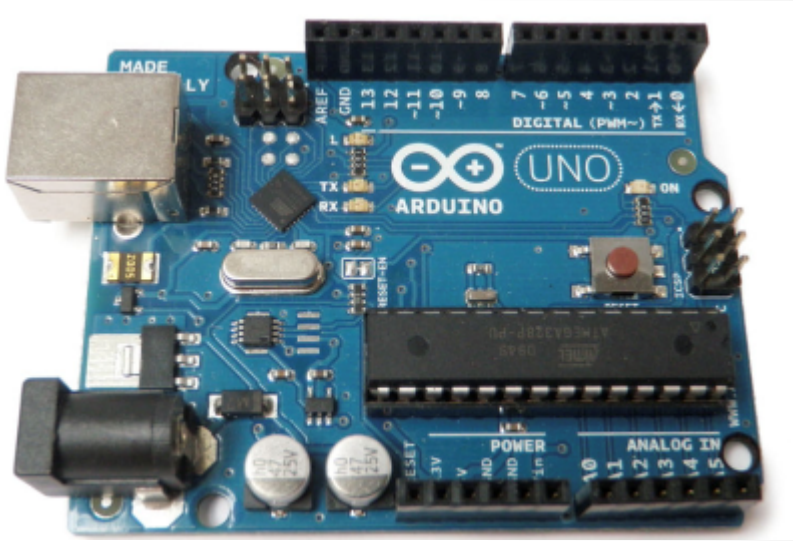
Next 30 minutes …



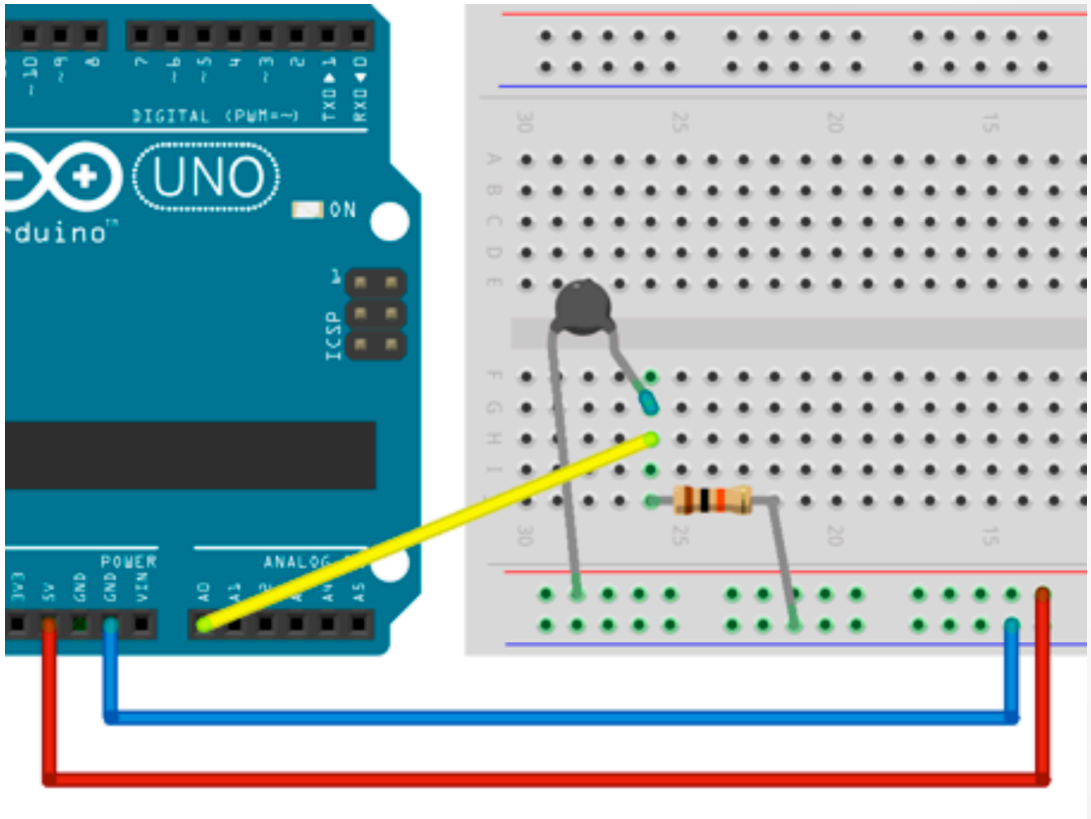
Greenhouse

Arduino

```
void setup() {
  // setup code
}

void loop() {
  // main code
}
```

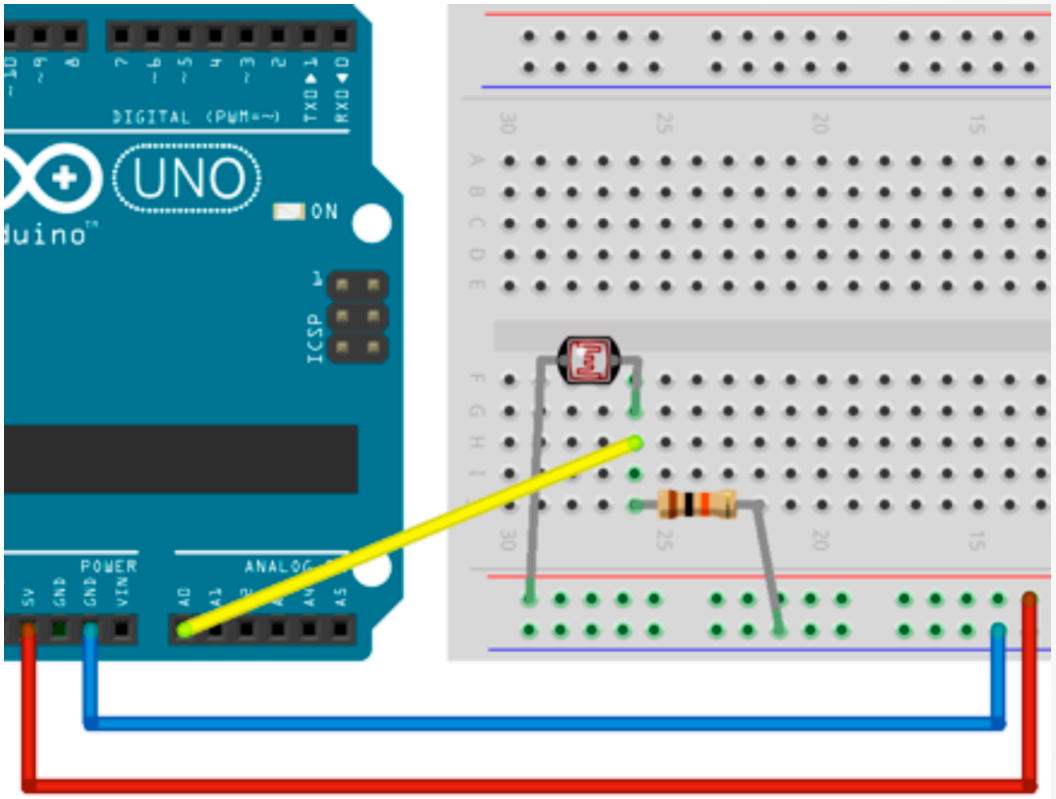


Temperature



10K Ohm NTC 5mm Thermistor

Light

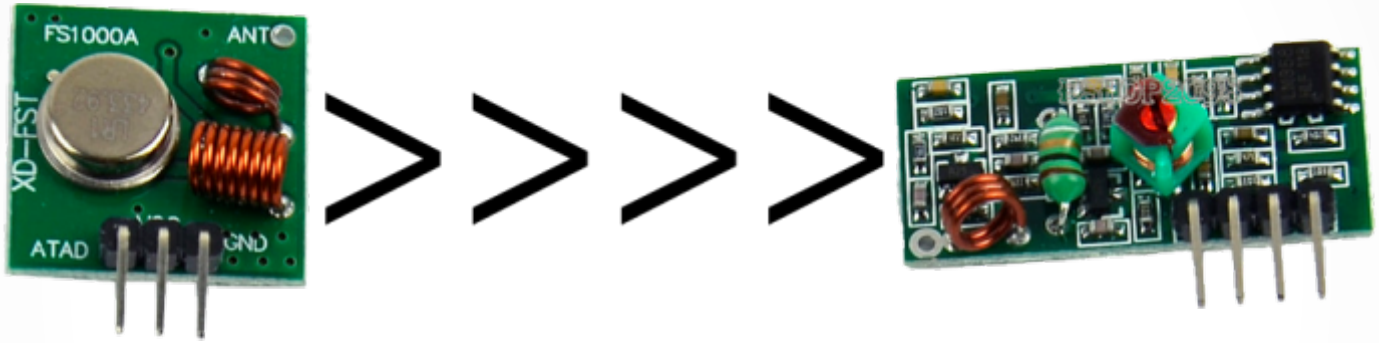


10K Ohm GL5528 Photo Resistor

Reading values

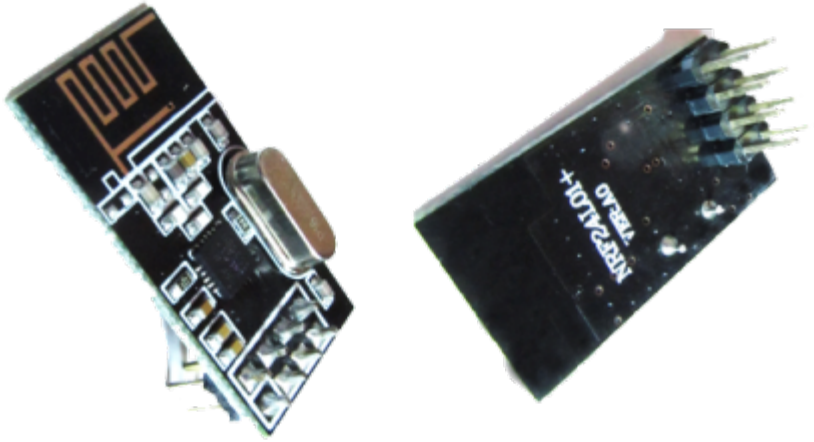
```
int reading;

void setup() {
  Serial.begin(9600);
}

void loop() {
  reading = analogRead(A0);

  int lightLevel = map(reading, 0, 1023, 0, 100);

  Serial.print("light = ");
  Serial.println(lightLevel);

  delay(3000);
}
```

Sending Values – MK



Transmitter >>>> Receiver

Sending Values - nRF24



Sending Values - nRF24

```
#include <RF24.h>

RF24 radio(9, 10);

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

void setup(void) {
  radio.begin();
  radio.setRetries(15, 15);
  radio.openWritingPipe(pipes[0]);
  radio.openReadingPipe(1, pipes[1]);
  radio.startListening();
}
```

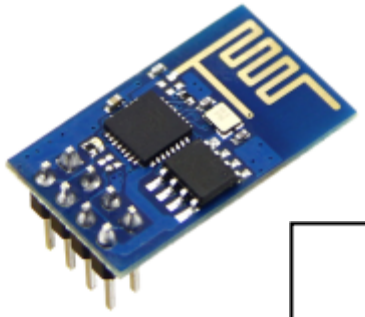
Sending Values - nRF24

```
typedef struct {
  char source;
  char destination;
  float temperatureIn;
  float temperatureOut;
  float checkTemperature;
  float humidity;
  int light;
  char action;
} SensorReadingData;

SensorReadingData data_in, data_out;
```

Sending Values - nRF24

```
void loop(void) {

…..

  radio.stopListening();
  bool ok = radio.write(&data_out, sizeof(data_out));
  radio.startListening();

  delay(10000);
}
```

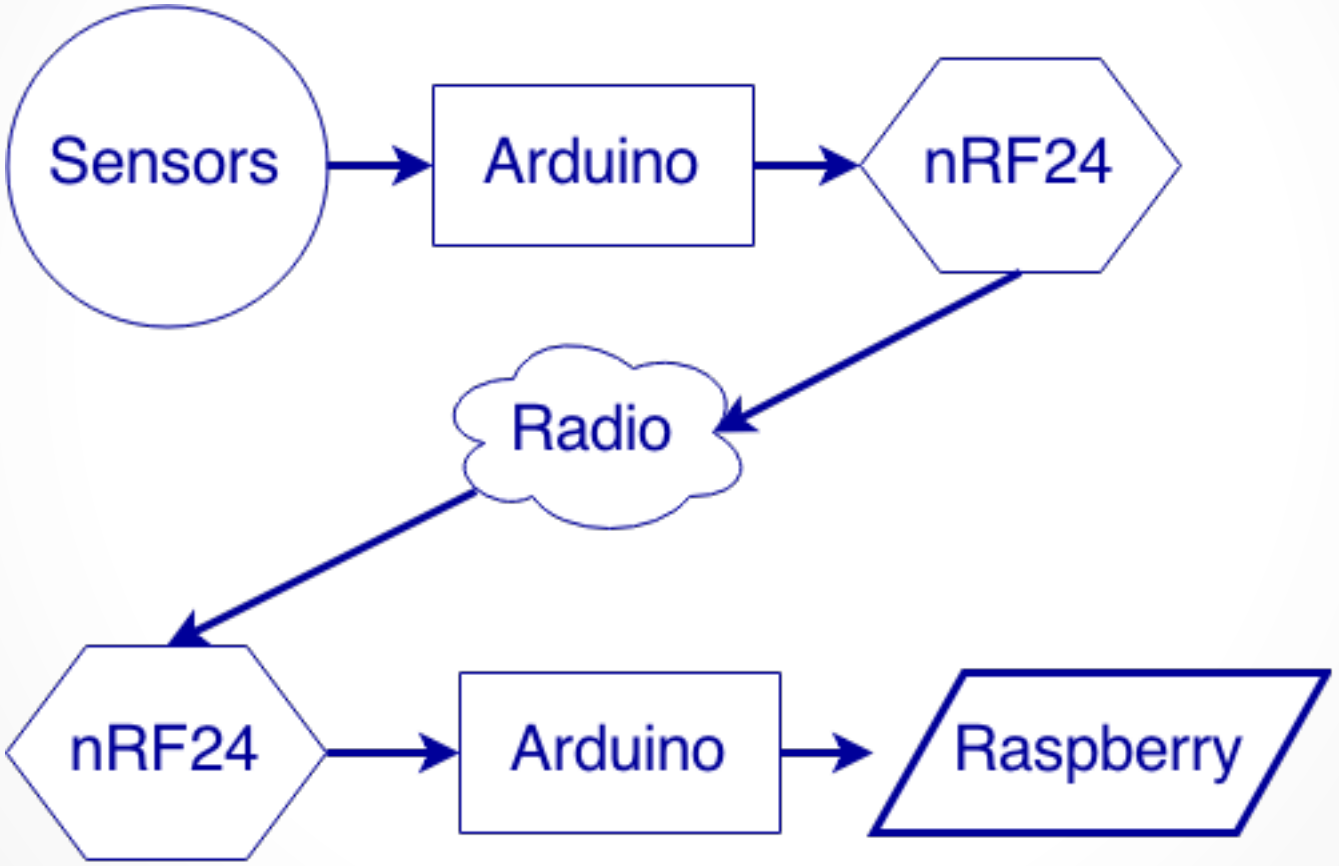
Sending Values - ESP8266



Backside

GND - GND	TX - (3)
GPIO 2	CH_PD - 3.3V
GPIO 0	RESET
RX - (2)	VCC - 3.3 V

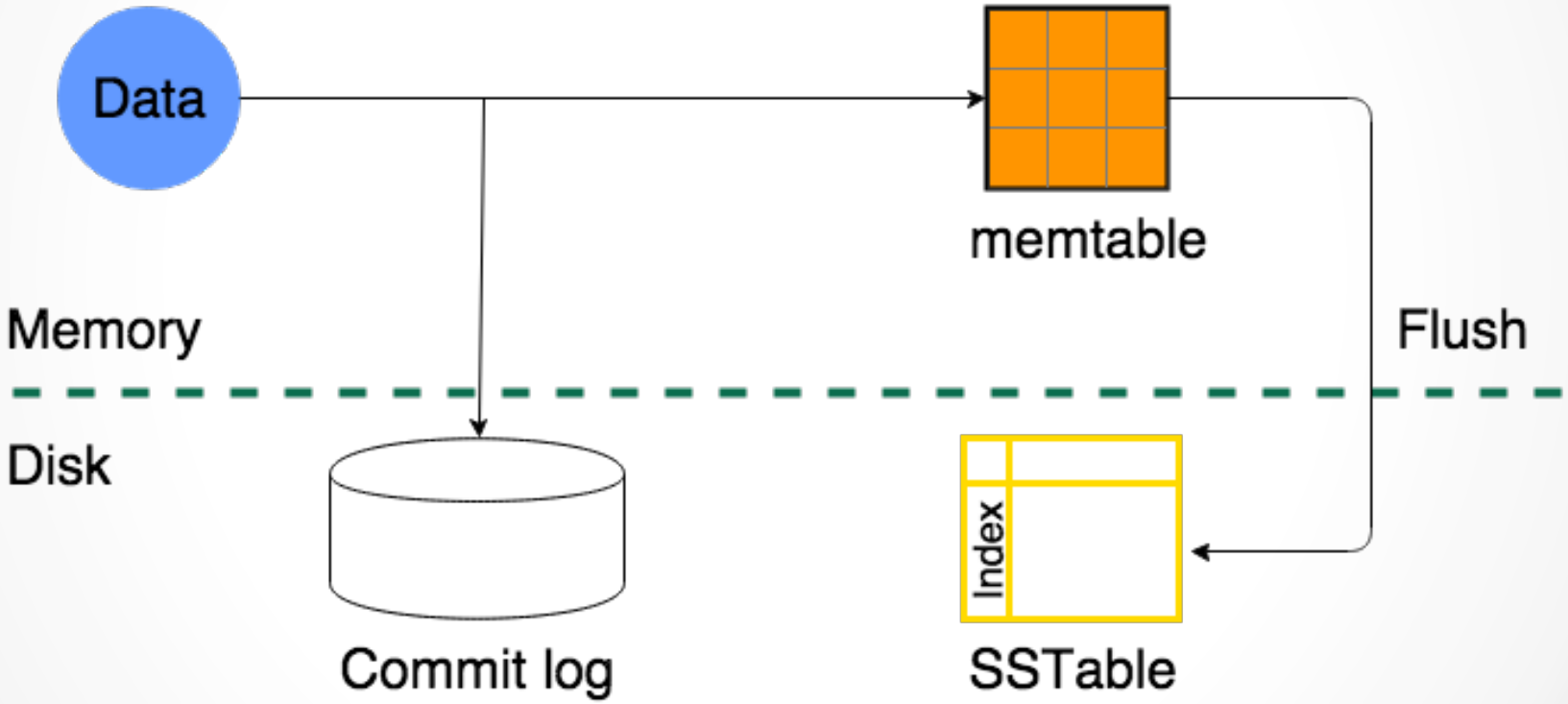
Reading and Sending part



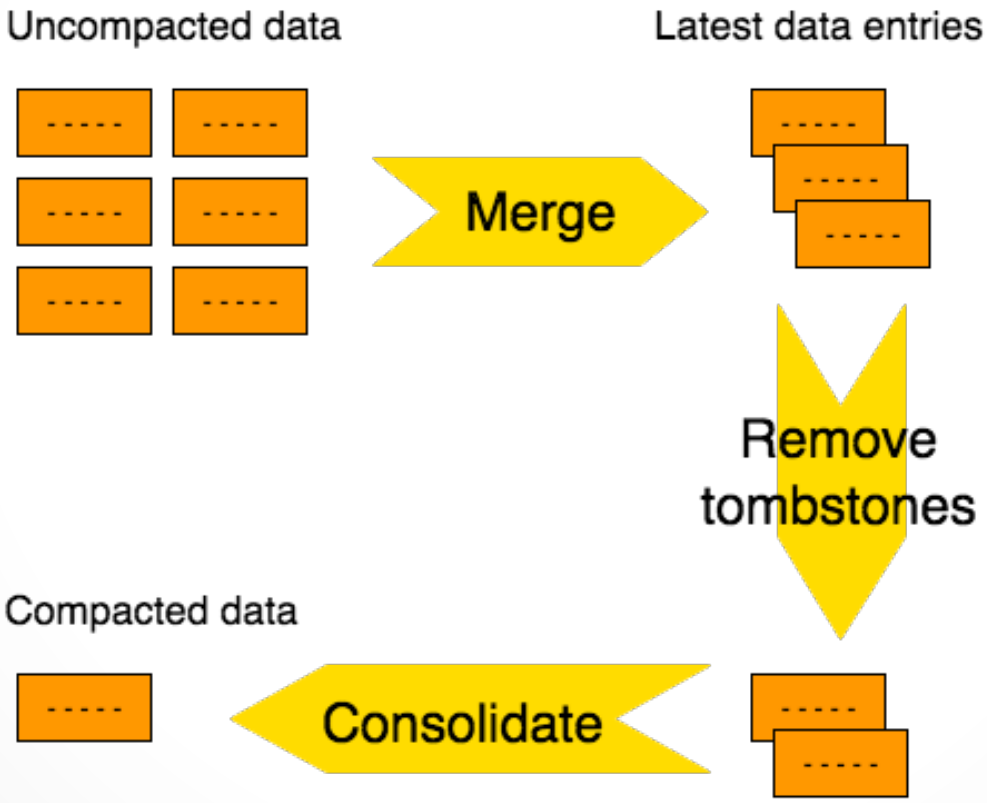
Why Cassandra?

- High volume by design
- Orders data automatically
- Great fit for time series data
- Easy to scale

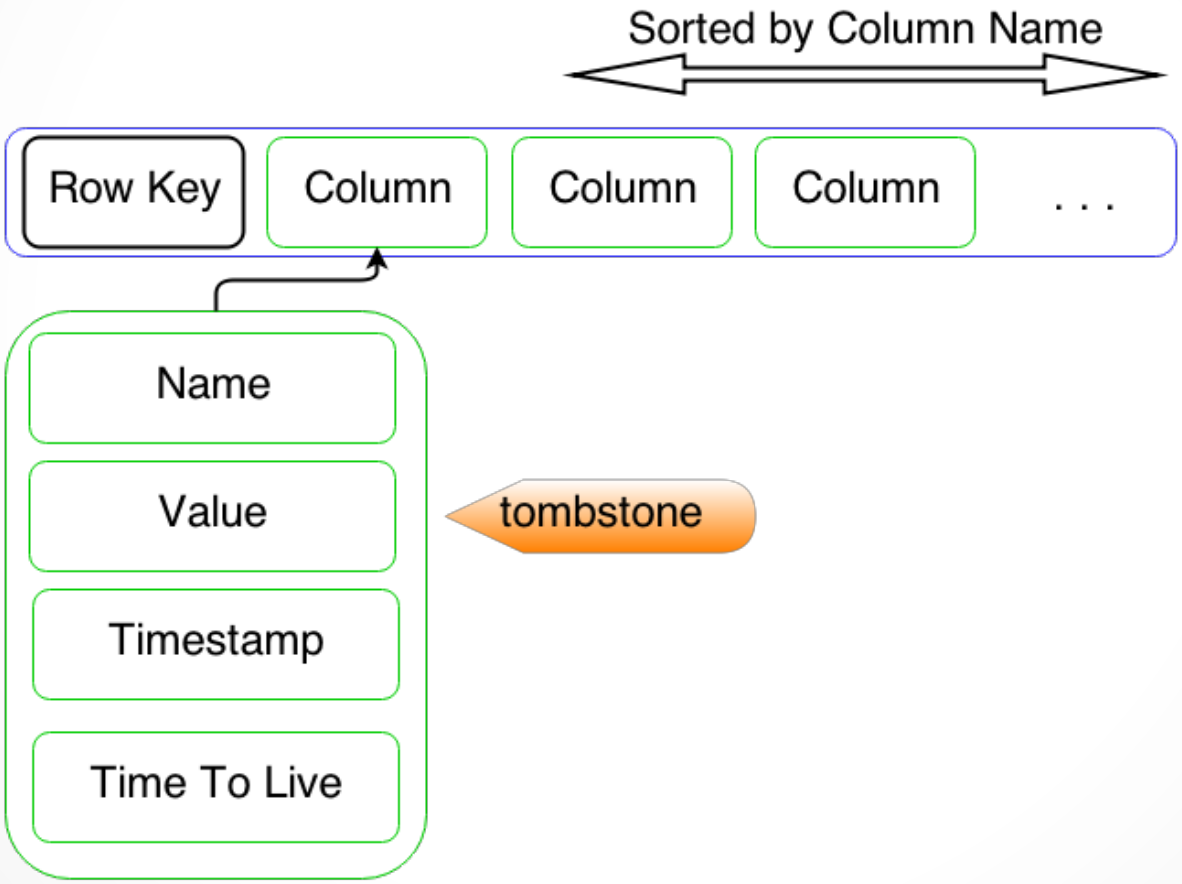
Write path



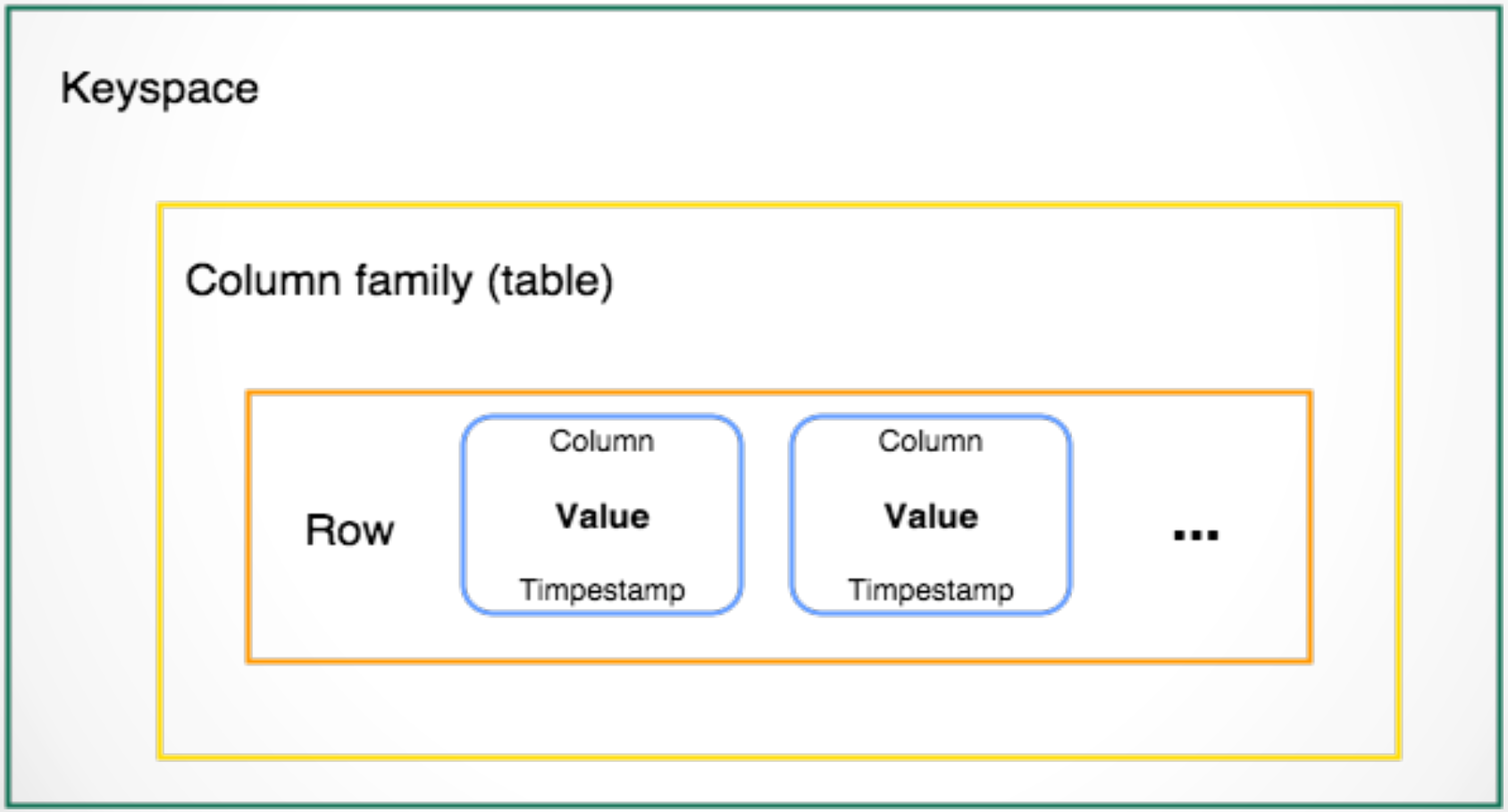
Compaction


Uncompacted data
Merge
Latest data entries
Remove tombstones
Compacted data
Consolidate

Wide Row


Sorted by Column Name
Row Key
Column
Column
Column
. . .
Name
Value
tombstone
Timestamp
Time To Live

Data hierarchy



Greenhouse

```
CREATE TABLE greenhouse (
    source text,
    day text,
    time timestamp,
    temperaturein decimal,
    temperatureout decimal,
    temperaturecheck decimal,
    humidity decimal,
    light int,
    PRIMARY KEY ((source, day), time)
)
WITH CLUSTERING ORDER BY (time DESC);
```

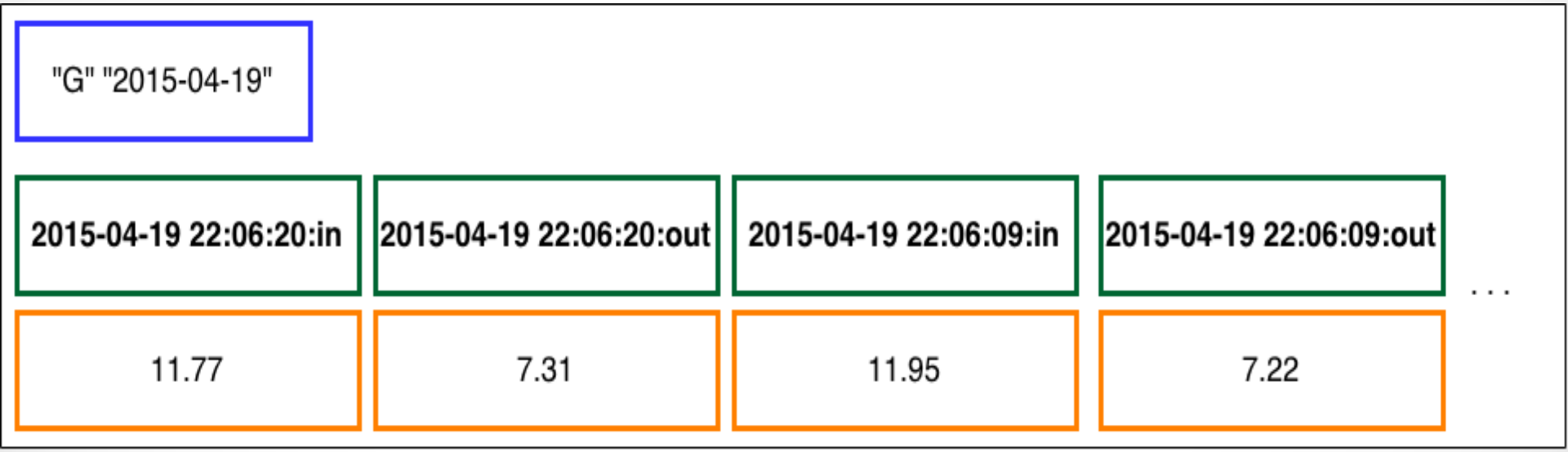
Fetching Data

```
SELECT source, day, time, temperaturein, temperatureout
    FROM greenhouse
        WHERE source = 'G'
            AND day = '2015-04-19' LIMIT 3;
```

source	day	time	in	out
G	2015-04-19	2015-04-19 22:06:20	11.77	7.31
G	2015-04-19	2015-04-19 22:06:09	11.77	7.31
G	2015-04-19	2015-04-19 22:05:59	11.77	7.31

PRIMARY KEY ((source, day) , time)

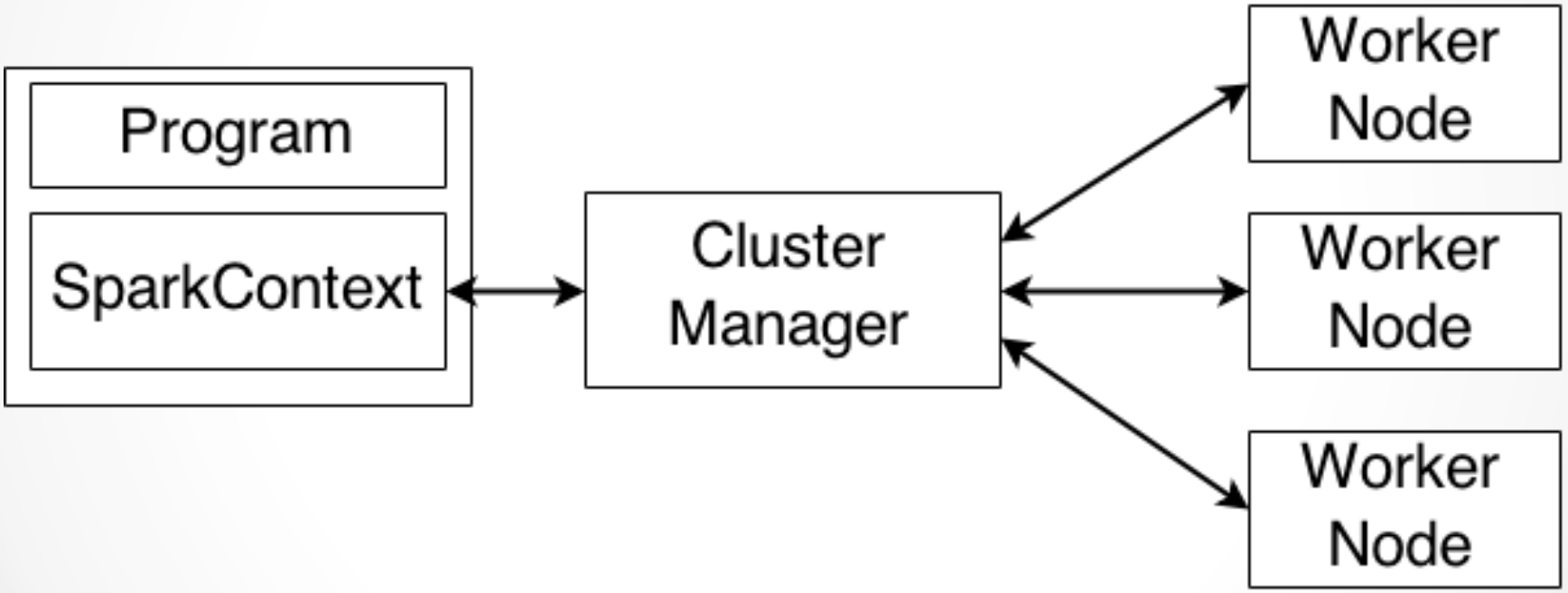
How it's stored



Spark

- Uses memory and disk to cache data
- Most active Big Data project
- 2014-11 sort 100 TB, 4.27 TB / min (1.42)

Spark


Program
SparkContext
Cluster Manager
Worker Node
Worker Node
Worker Node

Spark

JavaCro'15

Spark Master at spark://mm-marko.lan:7077

URL: spark://mm-marko.lan:7077
Workers: 1
Cores: 4 Total, 0 Used
Memory: 7.0 GB Total, 0.0 B Used
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Id	Address	State	Cores	Memory
worker-20150412091923-mm-marko.lan-59048	mm-marko.lan:59048	ALIVE	4 (0 Used)	7.0 GB (0.0 B Used)

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20150412093137-0000	InitialTest	4	512.0 MB	2015/04/12 09:31:37	msvaljek	FINISHED	2.7 min

Spark Analysis

```
SparkConf conf = new SparkConf();
conf.setAppName("Analysis");
conf.setMaster("local"); // spark://mm-marko.lan:7077
conf.set("spark.cassandra.connection.host",
                    "192.168.1.10");

Analysis app = new Analysis(conf);
app.run();
```

Spark Analysis

```
JavaRDD<Measurement> measurements =
        javaFunctions(sc)
            .cassandraTable(
                "home", "greenhouse",
                        mapRowTo(Measurement.class));

    measurements.persist(
                StorageLevel.MEMORY_AND_DISK());
```

Spark Analysis

```
final BigDecimal coldNightTemp = new BigDecimal(2);

JavaRDD<Measurement> coldMeasurements =
        measurements.filter(
                new Function<Measurement, Boolean>() {
                public Boolean call(Measurement x) {
                        return x.getTemperatureout()
                                .compareTo(
                                coldNightMinTemp) <= 0; }
                });
```

Spark Analysis

JavaCro'15

```
final SimpleDateFormat dateFormat =
        new SimpleDateFormat("yyyy-MM-dd");

JavaRDD<String> coldNights = coldMeasurements.map(
        new Function<Measurement, String>() {
                @Override
                public String call(Measurement m)
                        throws Exception {
                        return dateFormat
                                .format(m.getTime());
                }
        });

JavaRDD<String> coldNightsDistinct = coldNights.distinct();
```

Analysis Results

Average In
16.85

Average Out
12.45

Lowest Out
{source='G', day='2015-04-19', time=Sun Apr 19 05:32:03 CEST 2015, temperaturein=**3.59**, temperatureout=**-2.0**, temperaturecheck=5.0, humidity=41.0, light=0}

JavaCro'15

Thank you!

Q & A

@msvaljek

msvaljek@gmail.com