Untitled presentation
A story of data streaming and async I/O

From Scala to Go
Task: Data Enrichment

```json
{
    "proto": "tcp",
    "service": "ssl",
    "duration": 4.325414,
    "timestamp": 1533888289171,
    "srcip": "2a02:aa12:1500:3580:e575:1b16:a8c4:c800",
    "dstip": "2001:1458:201:66::100:14",
    "srcip_country": "Switzerland",
    "srcip_org": "UPC Schweiz GmbH"
}
```
Spark Streaming - not “real” streaming

discretized stream processing

records

Receiver

batches (RDDs)

batches processed with tasks

Spark

Long-running DNS queries

records processed in batches with short tasks
each batch is a RDD (partitioned dataset)

IDLE TIME
Complexity for mitigation

- 1st layer cache
- 2nd layer cache (Redis)
- Backup job to pick up slack
Advantages of being asynchronous
Enter Go

```go
func (m *Msg) finalProcess(val []byte) {
    fmt.Printf("%s\n", m.to)
    f, v := m.process.Process(val)
    if f {
        p := m.producer
        v := p.Process(v)
    }
}(msg.Value)
```
Moving parts

Go impl.  

Spark impl.

Cache
Easy scale-out
Conclusions / stats

Our DNS servers are warning that this host has been sending an UNACCEPTABLE rate of queries for one hour (876 requests/sec).
Extra juicy bit