Real-Time Streaming with Python ML Inference

Marko Topolnik

About Us

Hazelcast started in 2008 as a distributed cache

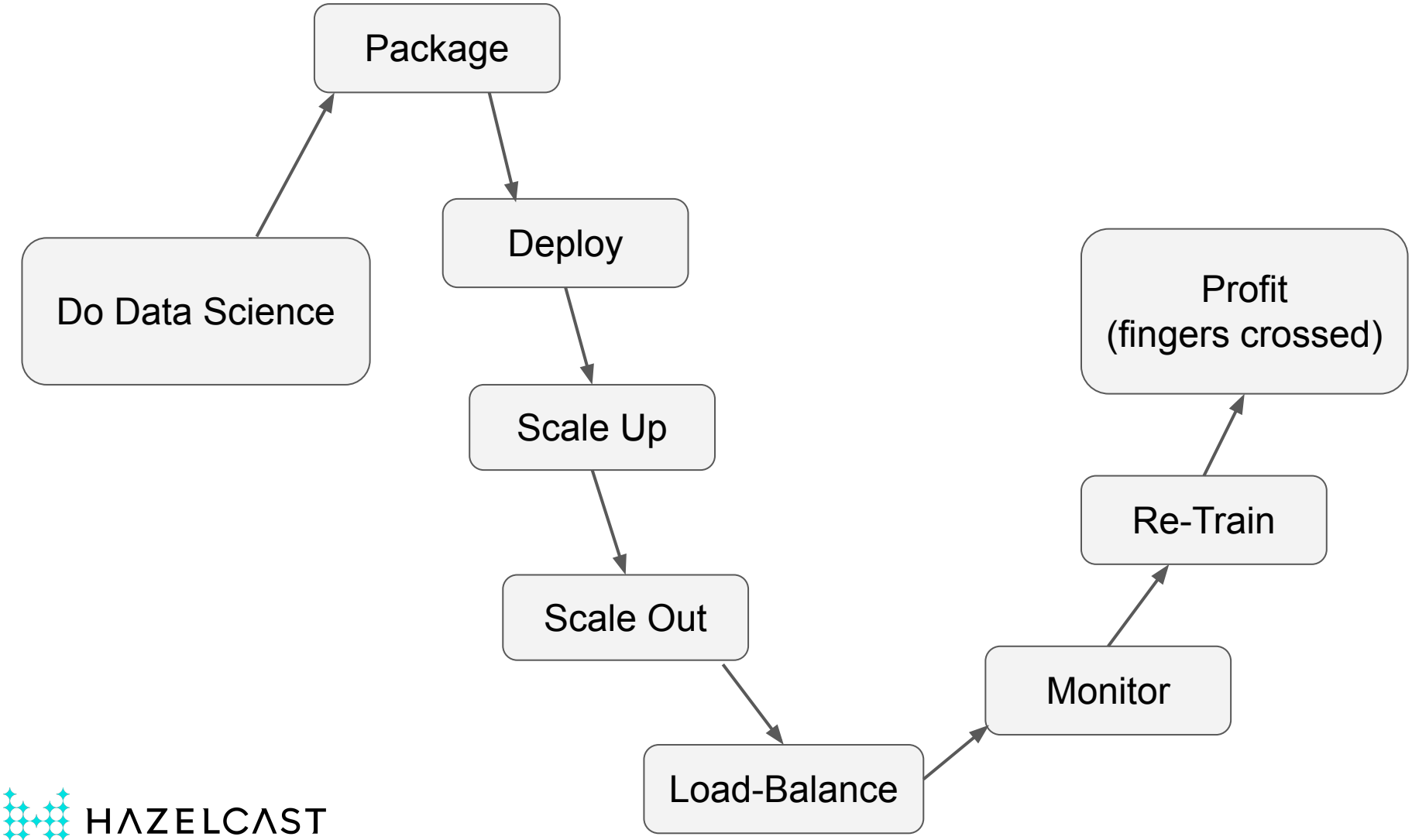
Today: main focus on real-time distributed stream processing

Our claim to fame is best-in-class latency

I co-authored the execution engine

Data Science: The Hype

Data Science: The Truth



HAZELCAST

Example: Salary Prediction

Python, SciKit Learn, Random Forest

Training Data

```
{
 "age": 25,
 "workclass": "Self-emp",
 "fnlwgt": 176756,
 "education": "HS-grad",
 "education-num": 9,
 "marital-status": "Never-married",
 "occupation": "Farming-fishing",
 "relationship": "Own-child",
 "capital-gain": 0,
 "capital-loss": 0,
 "hours-per-week": 35,
 "native-country": "United-States"
 "income": "<=50K" -- train ML to predict this!
}
```

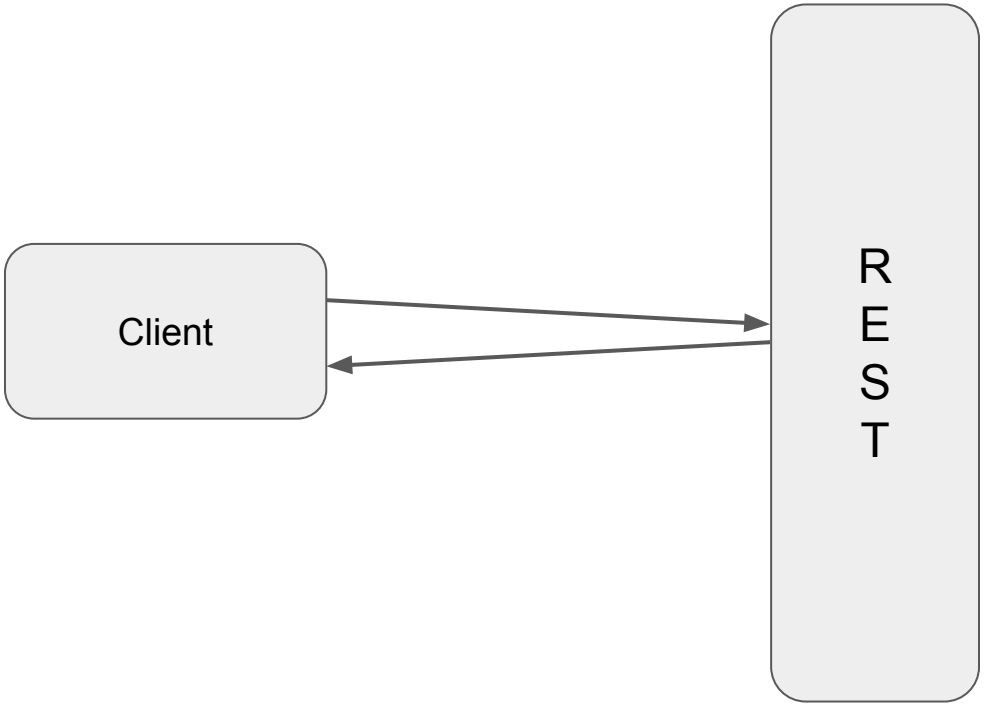
Sample Input and Output

```
Input: {
 "age": 25,
 "workclass": "Self-emp",
 "fnlwgt": 176756,
 "education": "HS-grad",
 "education-num": 9,
 "marital-status": "Never-married",
 "occupation": "Farming-fishing",
 "relationship": "Own-child",
 "capital-gain": 0,
 "capital-loss": 0,
 "hours-per-week": 35,
 "native-country": "United-States"
}
```

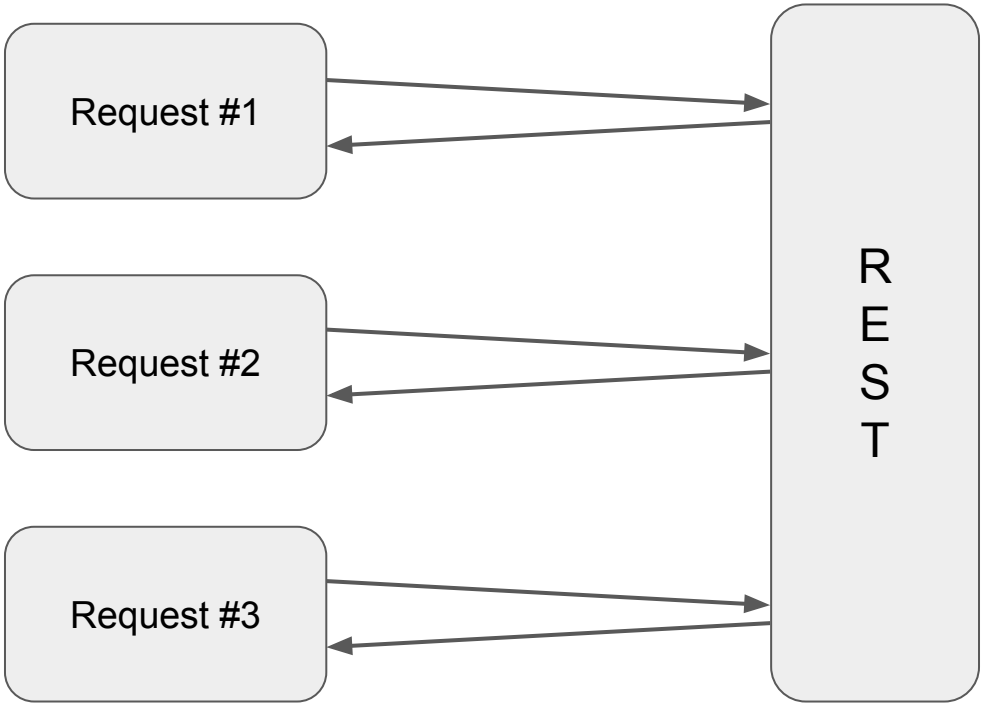
```
Output: {
 "probability": 0.85
 "income": "<=50K"
}
```

(Showing Project Directory)

We have a Web Service Doing ML!

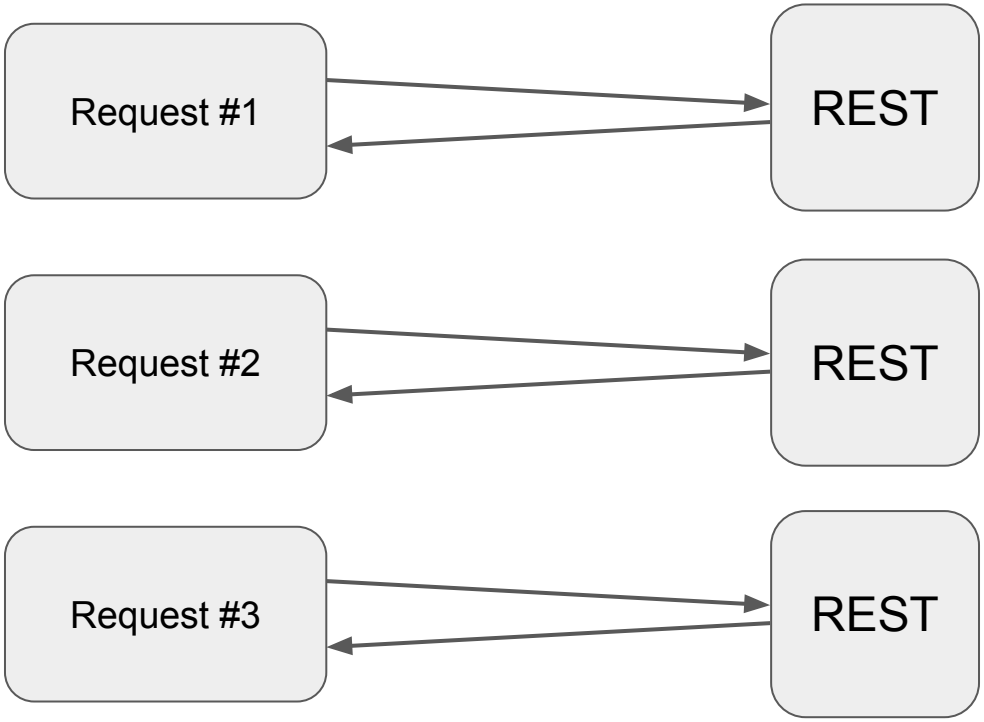


Productionizing the REST service



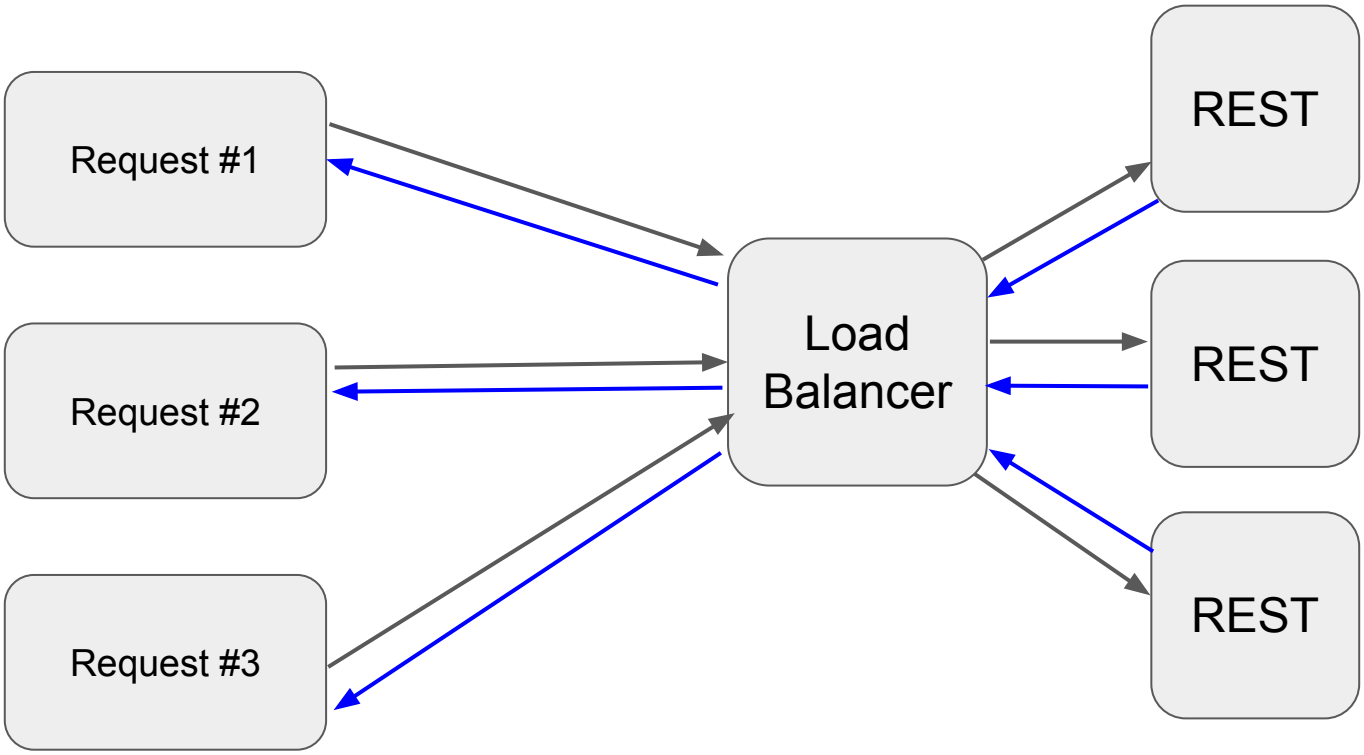
Parallelism?

Productionizing the REST service



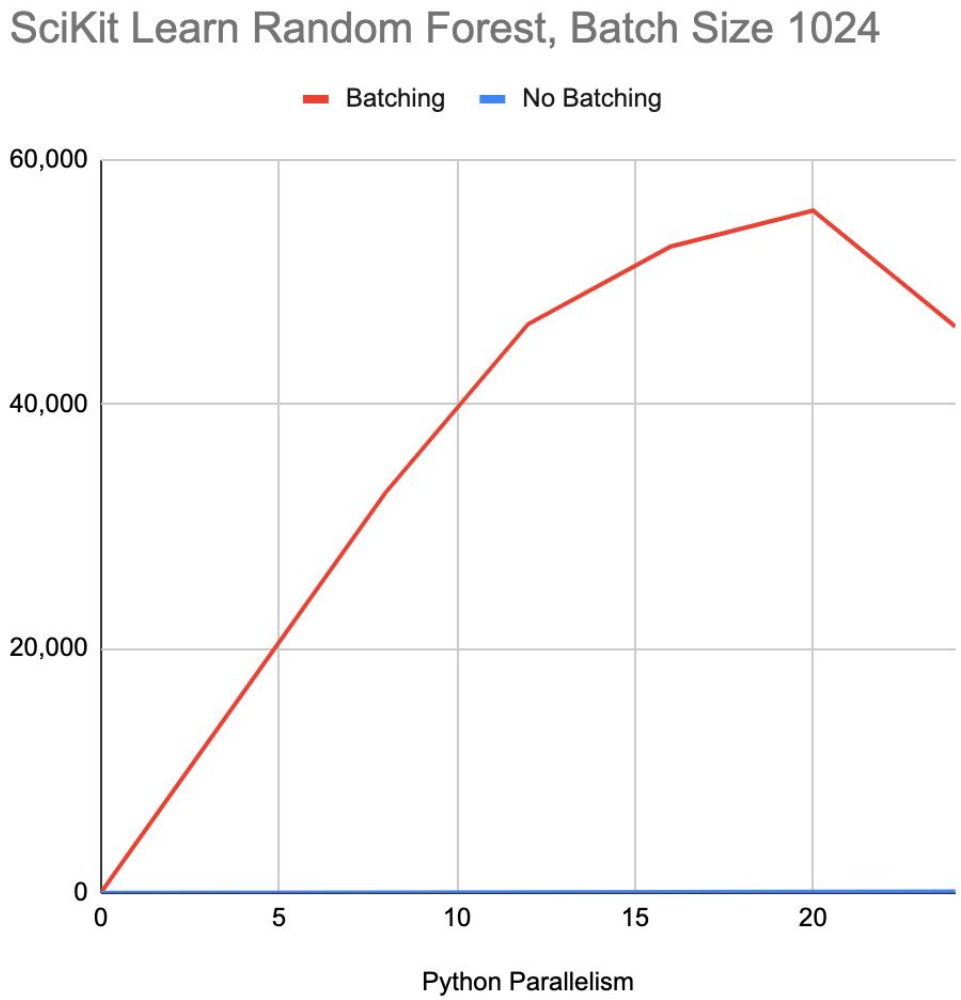
Load-Balancing?

Productionizing the REST service



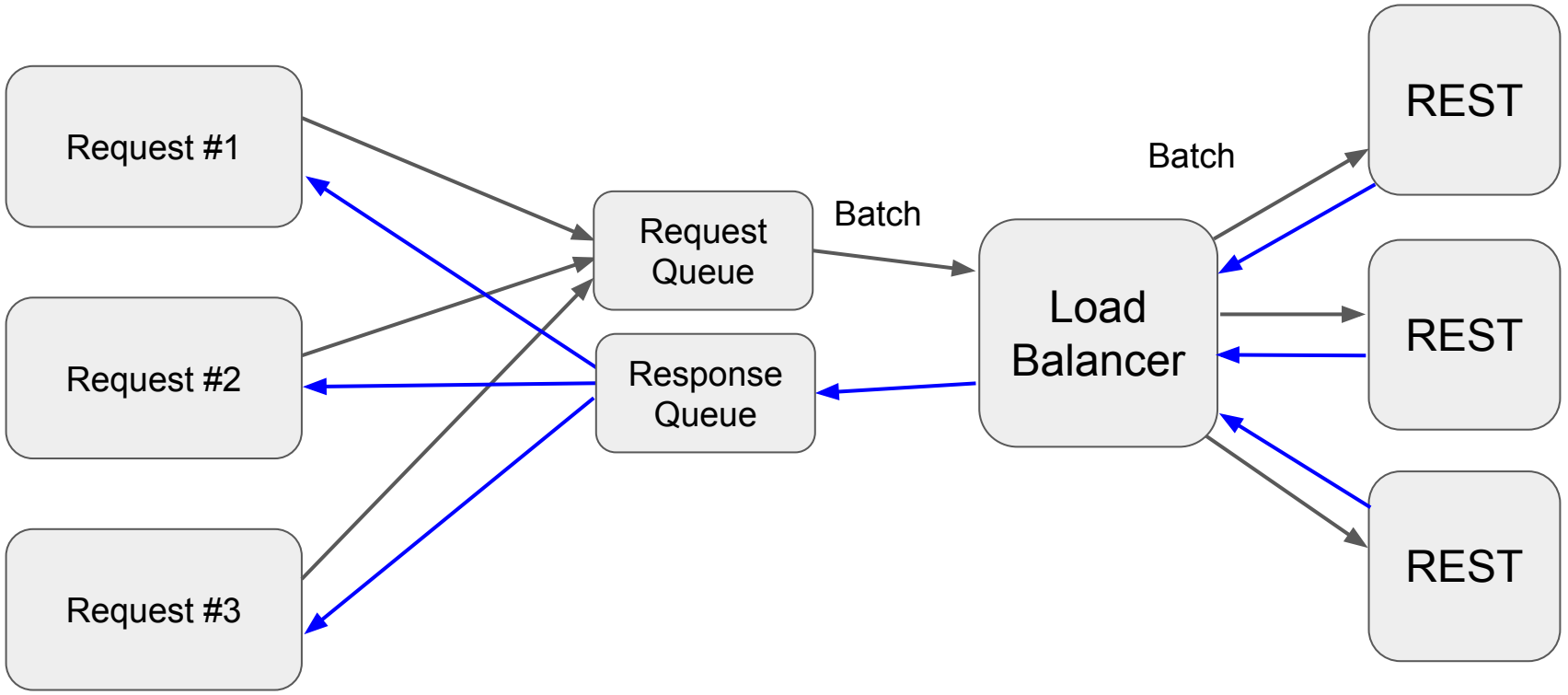
Batching?

Effect of Batching on Throughput

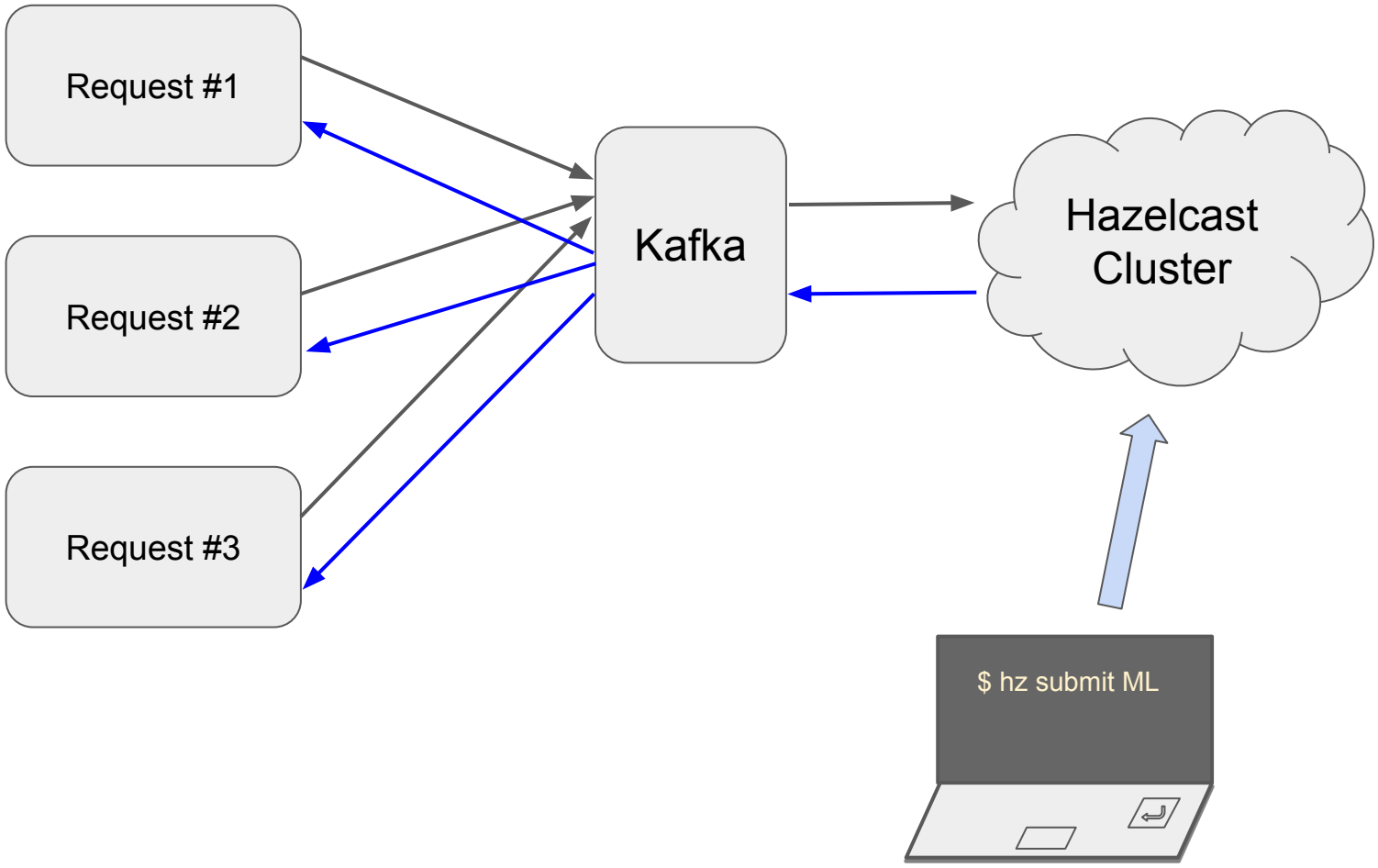


HAZELCAST

Productionizing the REST service



Replace REST with Distributed Streaming

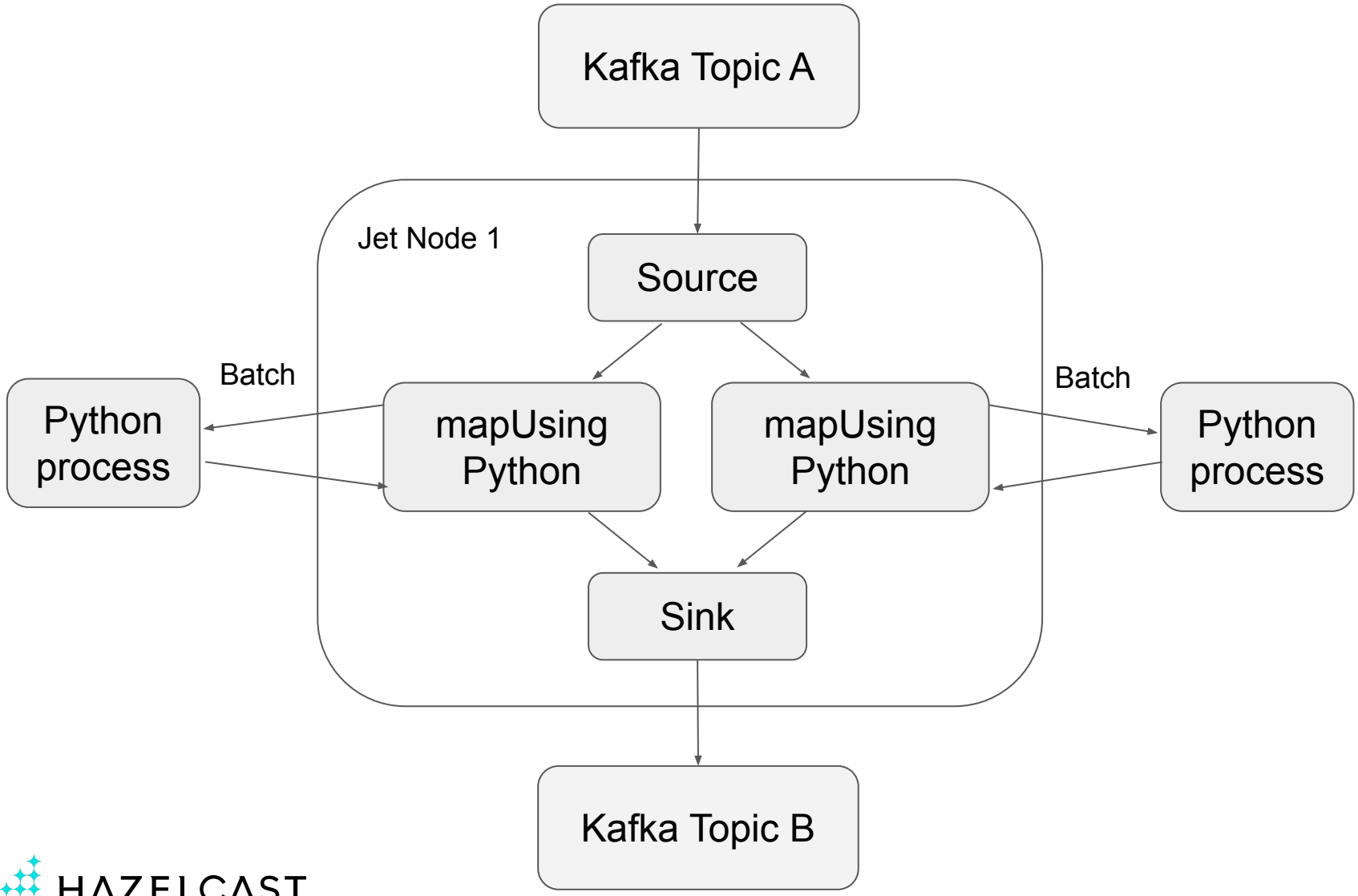


Hazelcast Pipeline Code

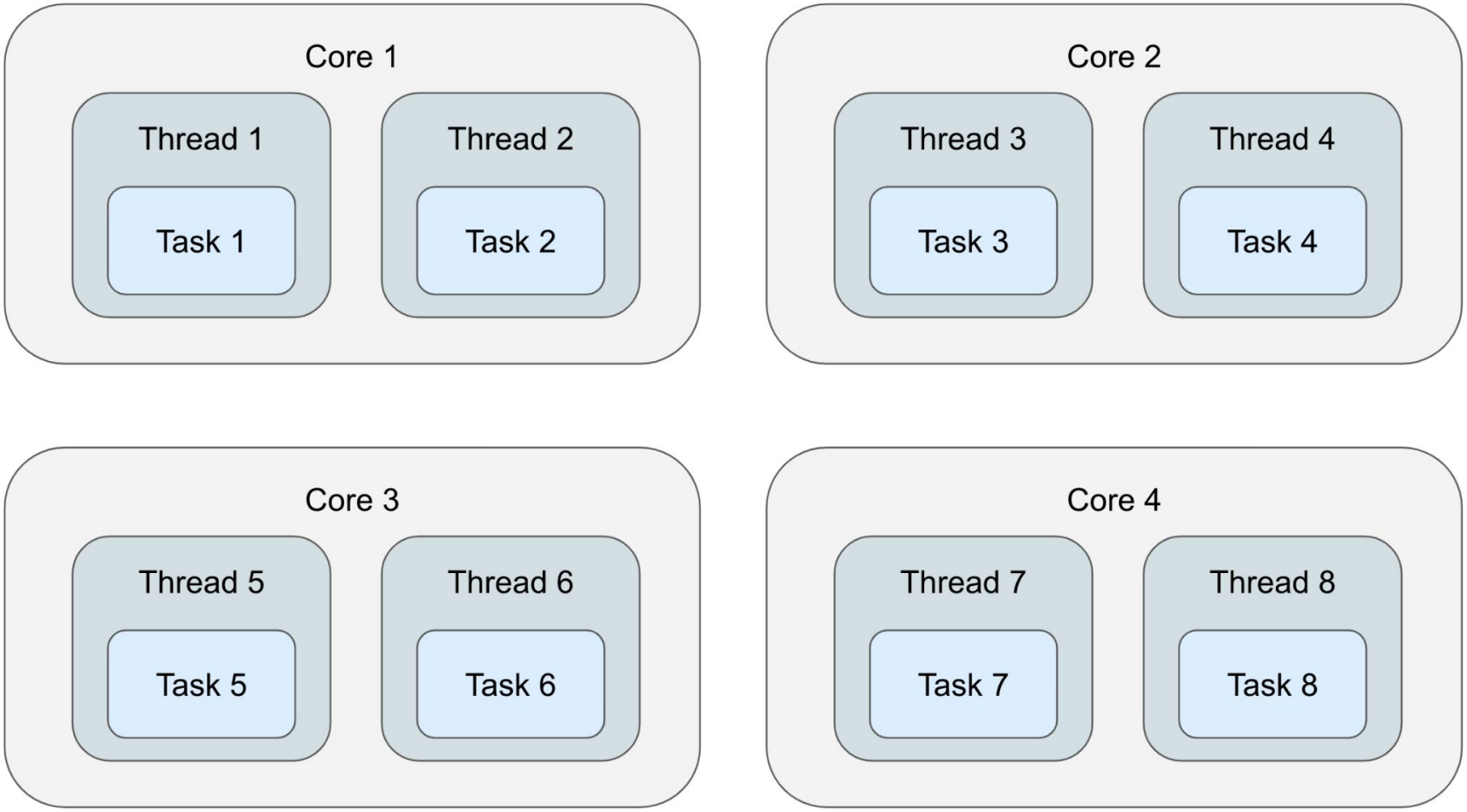
```
Pipeline p = Pipeline.create();
p.readFrom(Kafka.source())
 .apply(mapUsingPython(new PythonServiceConfig()
         .setBaseDir("/Users/mtopol/dev/python/sklearn")
         .setHandlerModule("example_1_inference_jet")))
 .writeTo(Kafka.sink());

hz.newJob(p);
```

```
$ mvn package
$ hz submit target/my-job.jar
```

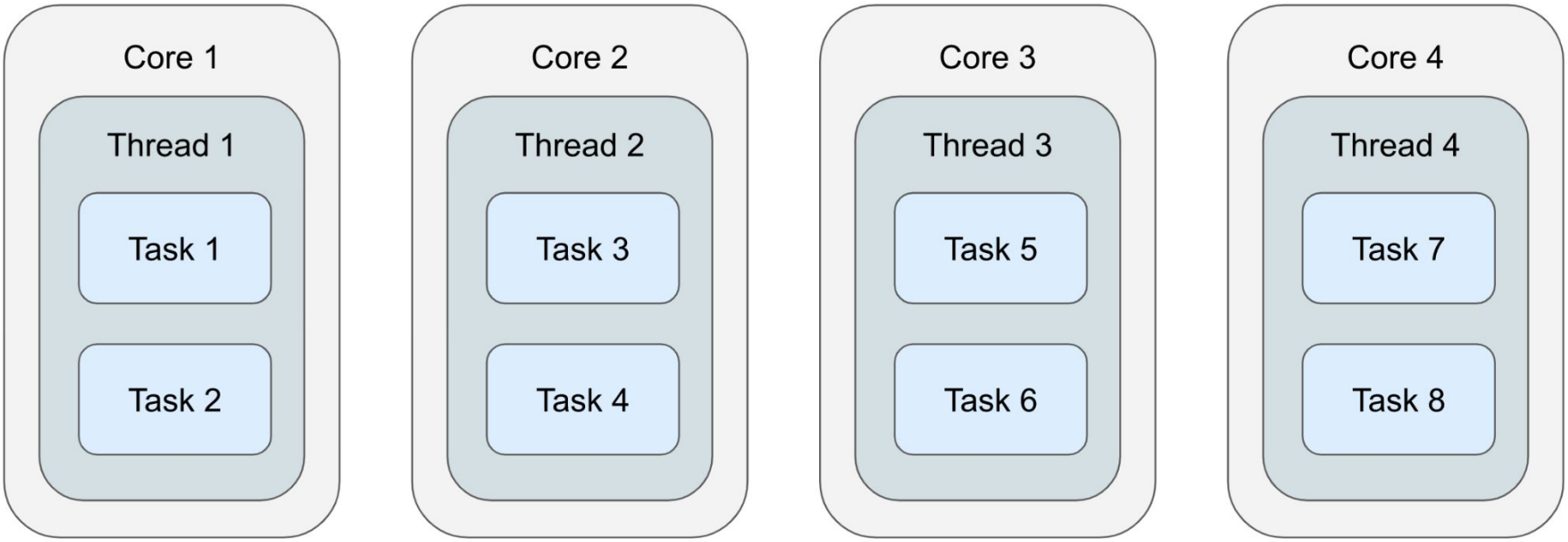
Pipeline Execution Plan



Traditional Engine: Thread per Task



Hazelcast's Engine: Thread per CPU Core



Let's Start a Jet Cluster!

Cluster Elasticity and Resilience

- processing jobs are fault-tolerant
- nodes can join and leave the cluster, jobs go on
- automatically rescale to available hardware

Cluster Self-Formation

Hazelcast natively supports:

- Amazon AWS
- Google GCP
- Kubernetes

With simple configuration, the nodes self-discover in these environments

Source and Sink Connectors

- Kafka
- Change Data Capture: MySQL, PostgreSQL, ...
- HTTP: WebSocket, Server-Sent Events
- Hadoop HDFS
- S3 bucket
- JDBC
- JMS queue and topic

Stream Operators

- windowed aggregation using Event Time
  - sliding, session window
  - count, sum, average, linear regression, ...
  - custom aggregate function
- rolling aggregation
- streaming join (co-grouping)
- hash join (enrichment)
- contact arbitrary external services
  - mapUsingPython uses this

Thanks for attending!

Q&A

marko@hazelcast.com
@mtopolnik