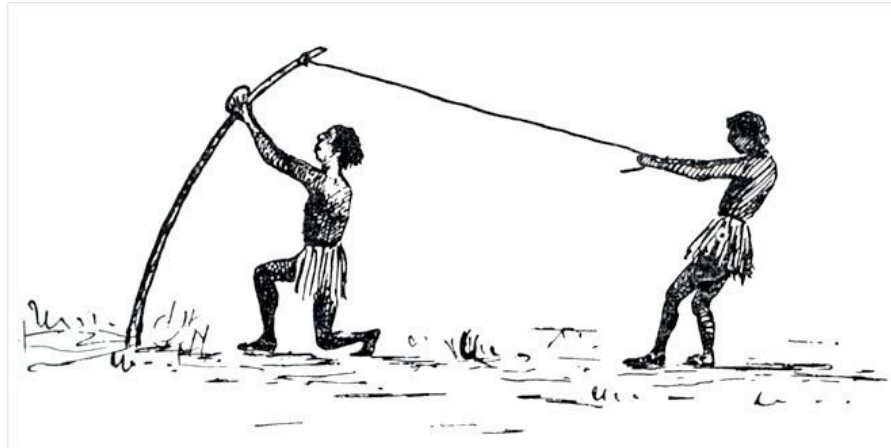


# Real-time Notification System

working for **INVENIO** @ CERN



State-of-the-art notification system



**Hi, I'm Jorge!**

Openlab Summer Student 2015  
Computer Science

**Supervisor:** Jiri Kunkar

# Real-time? What's that?

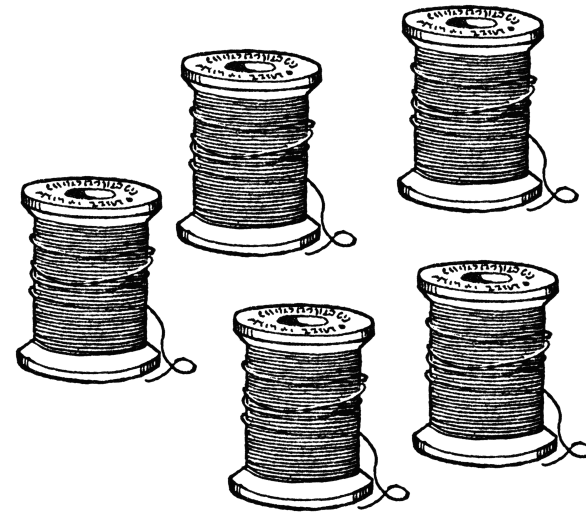
Systems subject to a time constraint



# System requirements

First week of research

**Asynchrony**



Tons of threads processing  
independent jobs

# System requirements

First week of research

Asynchrony

**High-availability**

**24 / 7**

Now, now and now!

# System requirements

First week of research

Asynchrony

High-availability

**Fault tolerance**



# System requirements

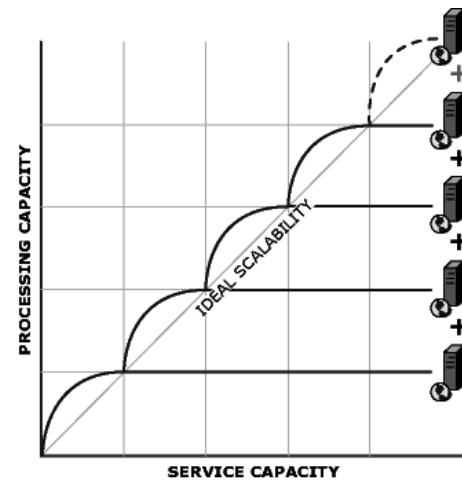
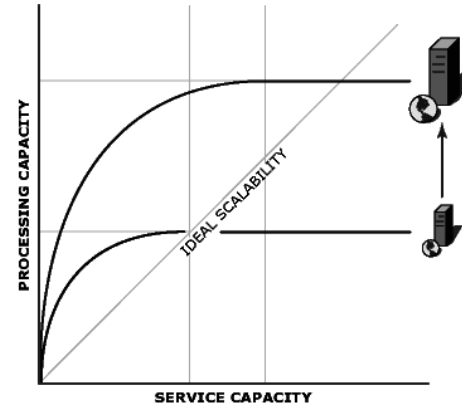
First week of research

Asynchrony

High-availability

Fault tolerance

**Scalability**



# System requirements

First week of research

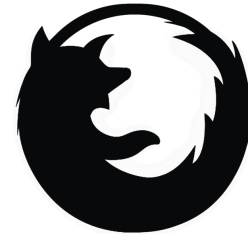
Asynchrony

High-availability

Fault tolerance

Scalability

**Device compatibility**





# System requirements

First week of research

Asynchrony

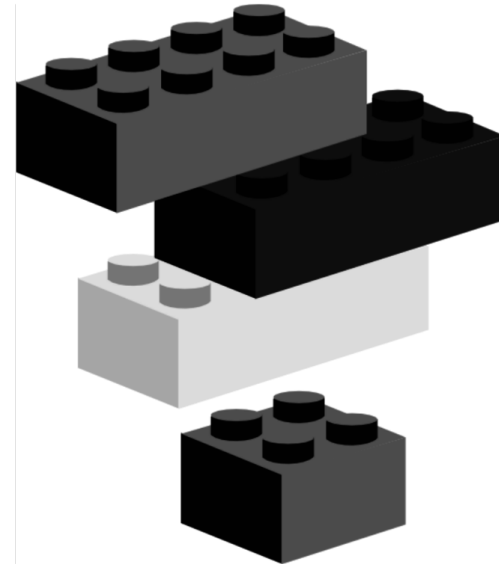
High-availability

Fault tolerance

Scalability

Device compatibility

**Consumer agnostic**



Logstash, PostgreSQL, Webhooks...

# Challenges

Learning a new technology stack



# Flask

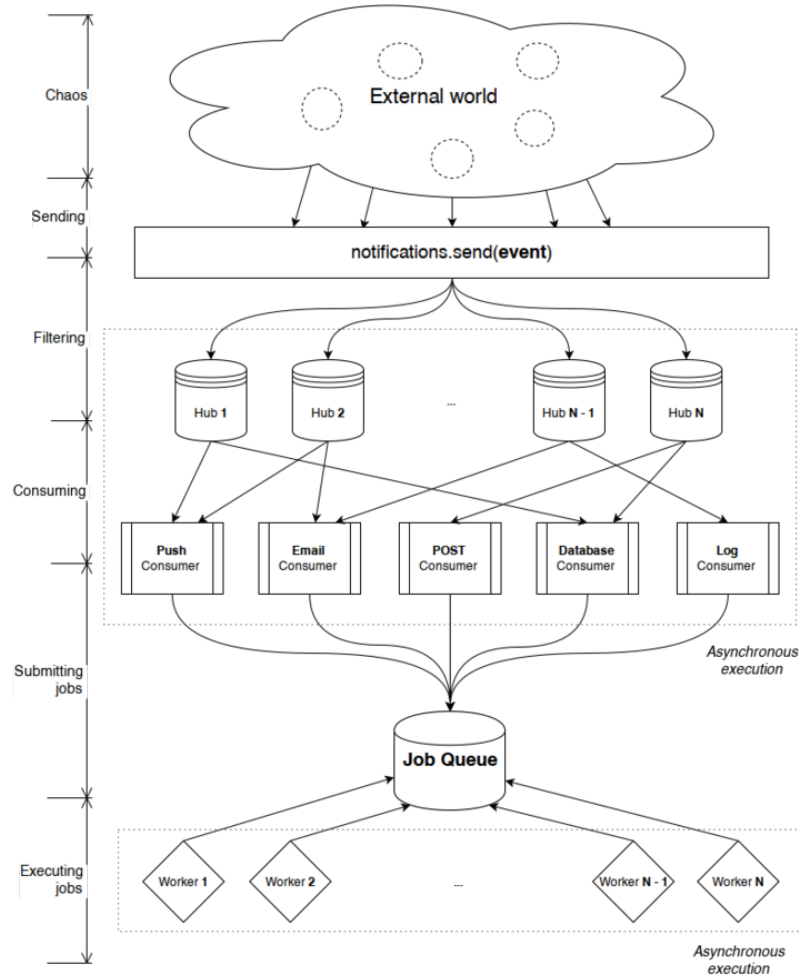
web development,  
one drop at a time

and more technologies without logo like...

- Server-Sent Events (SSE)
- Gevent

# Results

The whole architecture



# Client API

*Initialise &  
create hub*

```
notifications = Notifications(app=flask_app)
user_hub = notifications.create_hub()
```

*Register  
consumer with  
decorator*

```
@user_hub.consumer( celery_task_name="app.write_to_file" )
def write_to_file(event, *args, **kwargs):
    with open("events.log", "a+w") as f:
        f.write(str(event))
```

*Register  
predefined  
consumer*

```
push_consumer = PushConsumer(redis, user_hub_id)
user_hub.register_consumer(push_consumer)
```

*Decide which  
events will be  
processed*

```
expected_receivers = ["jiri", "tibor"]
user_hub.filter_by(
    WithSender("jorge") | WithReceivers(expected_receivers)
)
```

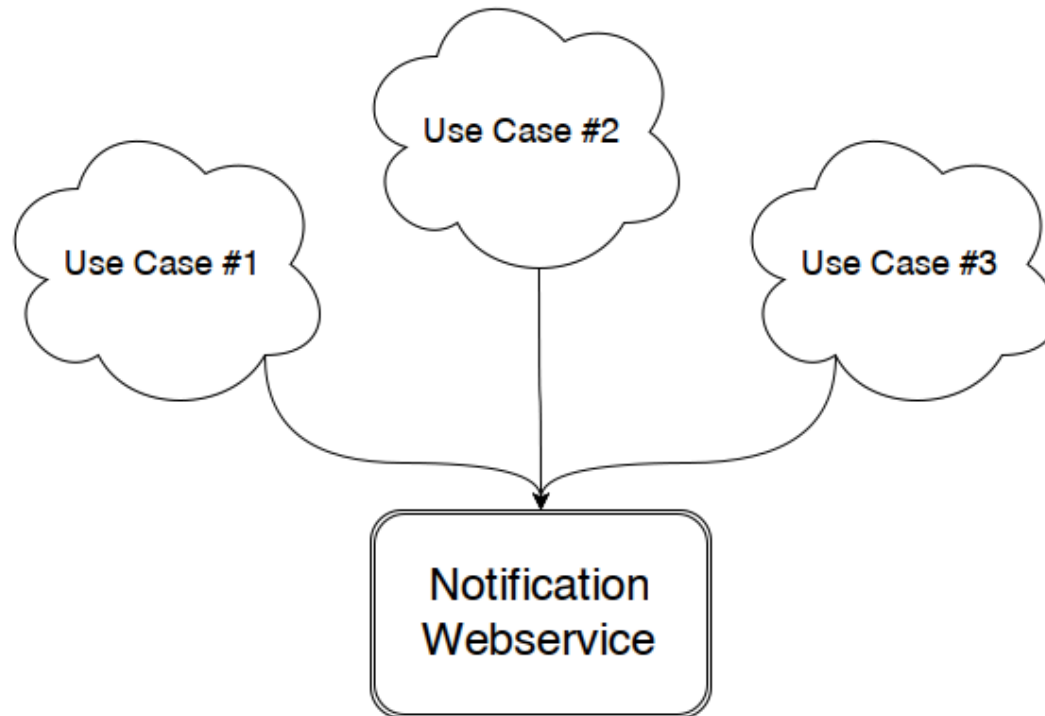
*Create and  
process event  
in JSON*

```
event = Event(event_id=None,
               event_type="user",
               title="This is a user test",
               body="This is the body of the test",
               sender="jorge",
               receivers=["jiri", "tibor"])
```

```
notifications.send(event.to_json())
```

# Close future

What is supposed to be?



# Thanks for your attention

And thanks to my supervisor and colleagues

