

A black and white photograph of Perseus from the 1981 movie "Clash of the Titans". He is shown from the waist up, wearing a dark, draped tunic. He holds the severed head of Medusa in his left hand, which is raised towards the top right of the frame. In his right hand, he holds a long, dark sword. The background consists of ancient stone ruins and a large, carved stone face of a deity. A semi-transparent grey rectangle is overlaid on the center of the image, containing the title text.

# Clash of the Titans: Erlang **Clusters** and Google AppEngine

# The need

**To build a tool that helps us  
manage the people in our lives,  
like the super star personal  
assistant we always dreamt of!  
SocialCaddy was born.**



# The quest

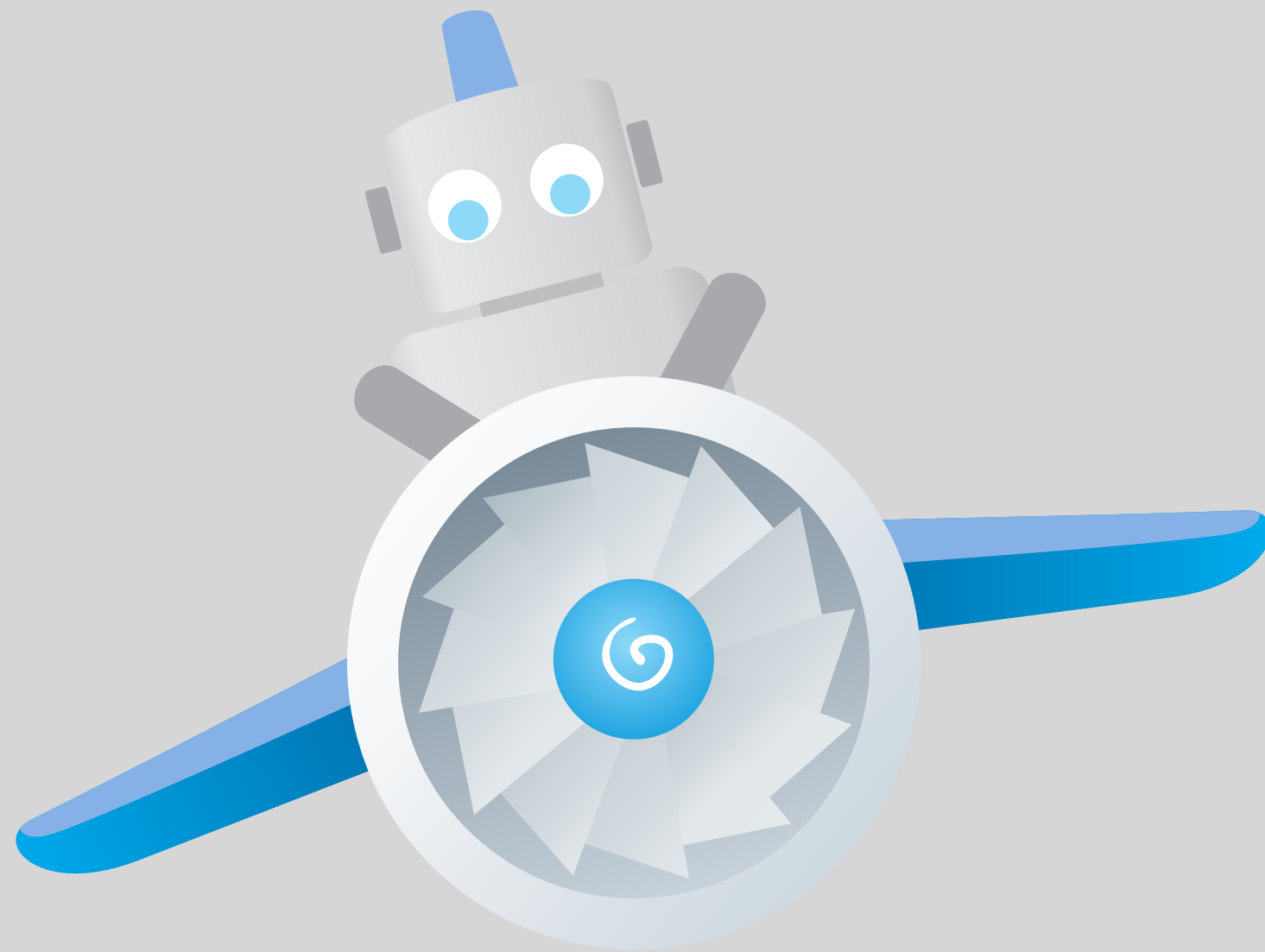
To build an application that  
would **scale** smoothly,  
without being hammered  
down by costs.



# The quest

Note: **Greeks** don't enjoy  
system administration.

# The solution



wasn't it obvious?

# The problem

**Importing and merging 500  
users by breaking into  
background tasks  
worked nicely.**



# The problem

**But when we tried with  
3000 users we hit the first  
restrictions**

# And now?



**Time for the big guys**



# The experiments

**BERT-rpc (Github)>too  
complicated, battletested,  
customized for GAE**

# The experiments

**Stackless Python  
> it's not Erlang**

# The experiments

**Disco project Map/Reduce**  
**> too expensive and lot's of rewrites**



# The solution

**It was time to build our own  
distributed platform using the  
right tools**

# The requirements

**Simplicity**  
**Language agnostic**  
**Tight integration with GAE**

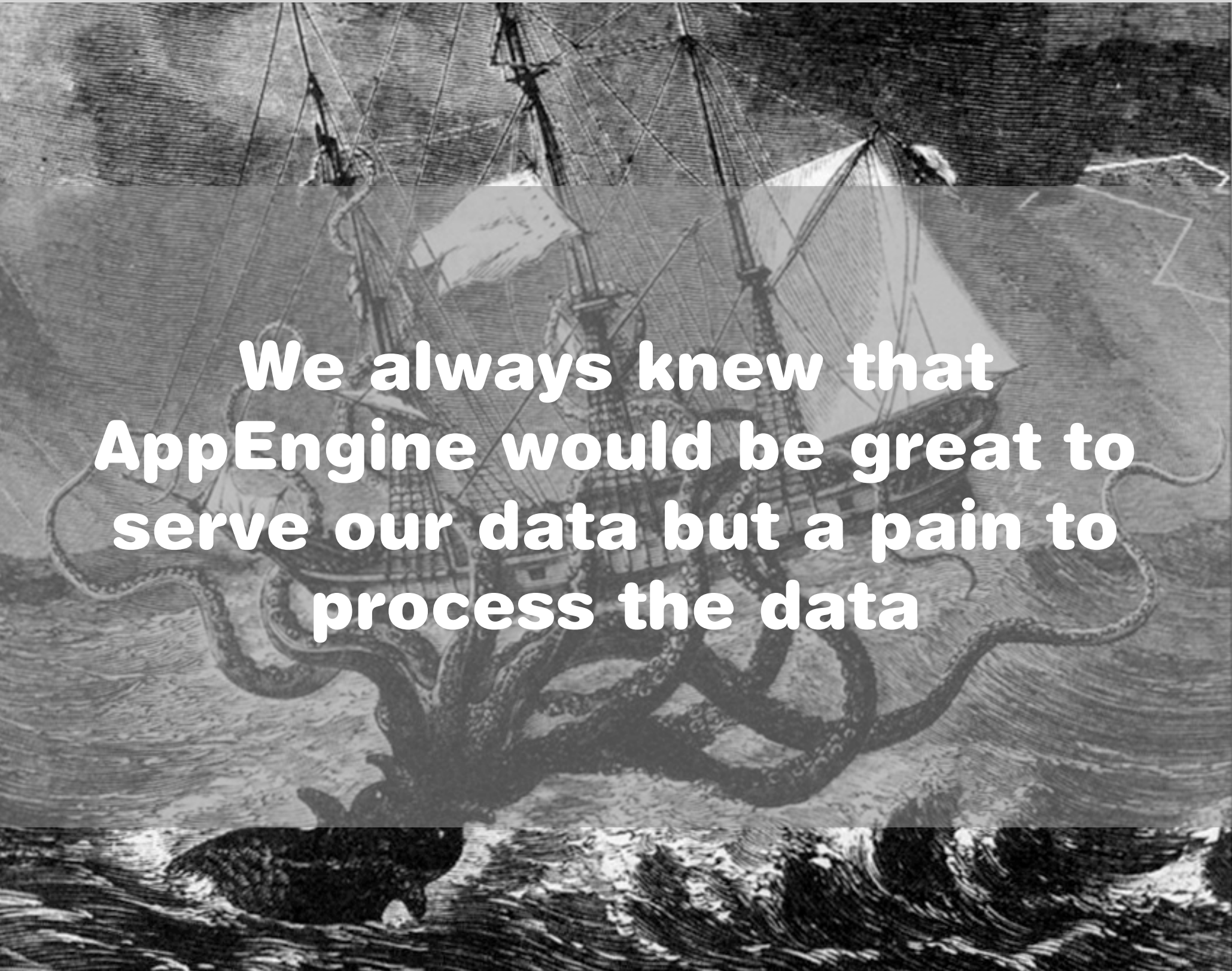
# The soap-opera



(the characters)



# The soap-opera



**We always knew that  
AppEngine would be great to  
serve our data but a pain to  
process the data**



# The soap-opera

**We tried to break data in small  
chunks> background tasks.**

**“OMG it works! TechCrunch  
here we come”**



# The soap-opera

**We celebrated our victory of  
importing and merging  
500-600 contacts on GAE**

**“OMG! Google I/O  
here we come!”**



# The soap-opera

**We poured a scotch and  
waited for Nikos to test.**

**Before the first sip,  
GAE had died.**



# Why so serious?

**Merging the data has  
polyonimal complexity and as  
data grows time is needed.**

**And it gets worse: We only  
have 30 seconds!**



# Tough decisions!

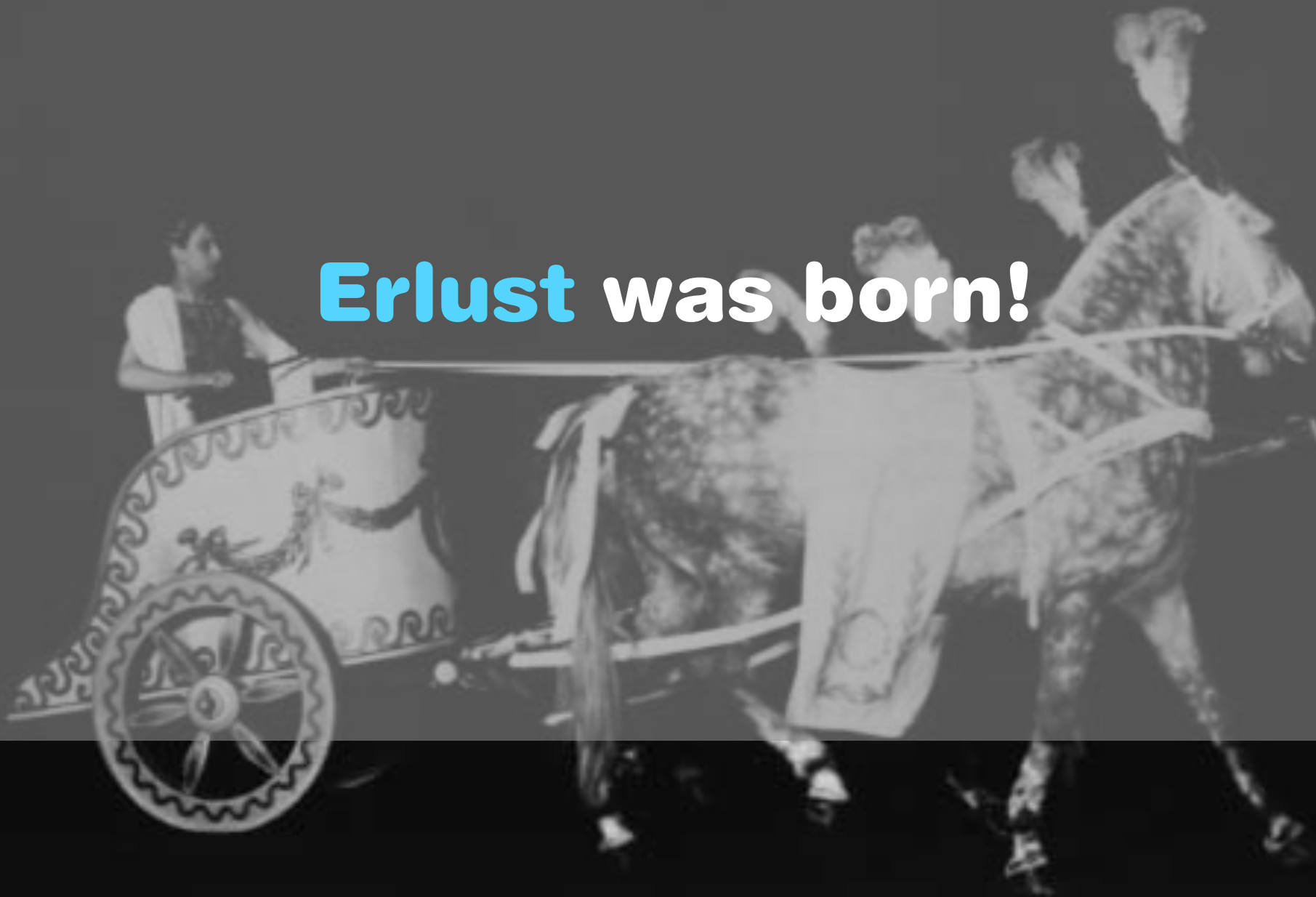


**Bye bye to GAE and  
to zero-administration or  
deep into Erlang**



# The solution

**Erlust** was born!



# The solution

**We use battle tested software  
such as MochiWeb and  
RabbitMQ combined with our  
beloved Python libs and the  
GAE remote API.**



# The solution

**That's right! We run the same merge algorithm on Erlust. We left back the smashed GAE quotas and the deadline exceeded errors and entered the world of set and forget**

# Reducing Map/Reduce

**Map/Reduce needs lots of resources and we come from semi-bankrupt Greece :)  
We chose the Consumer/Producer architecture that allowed us to control resources**

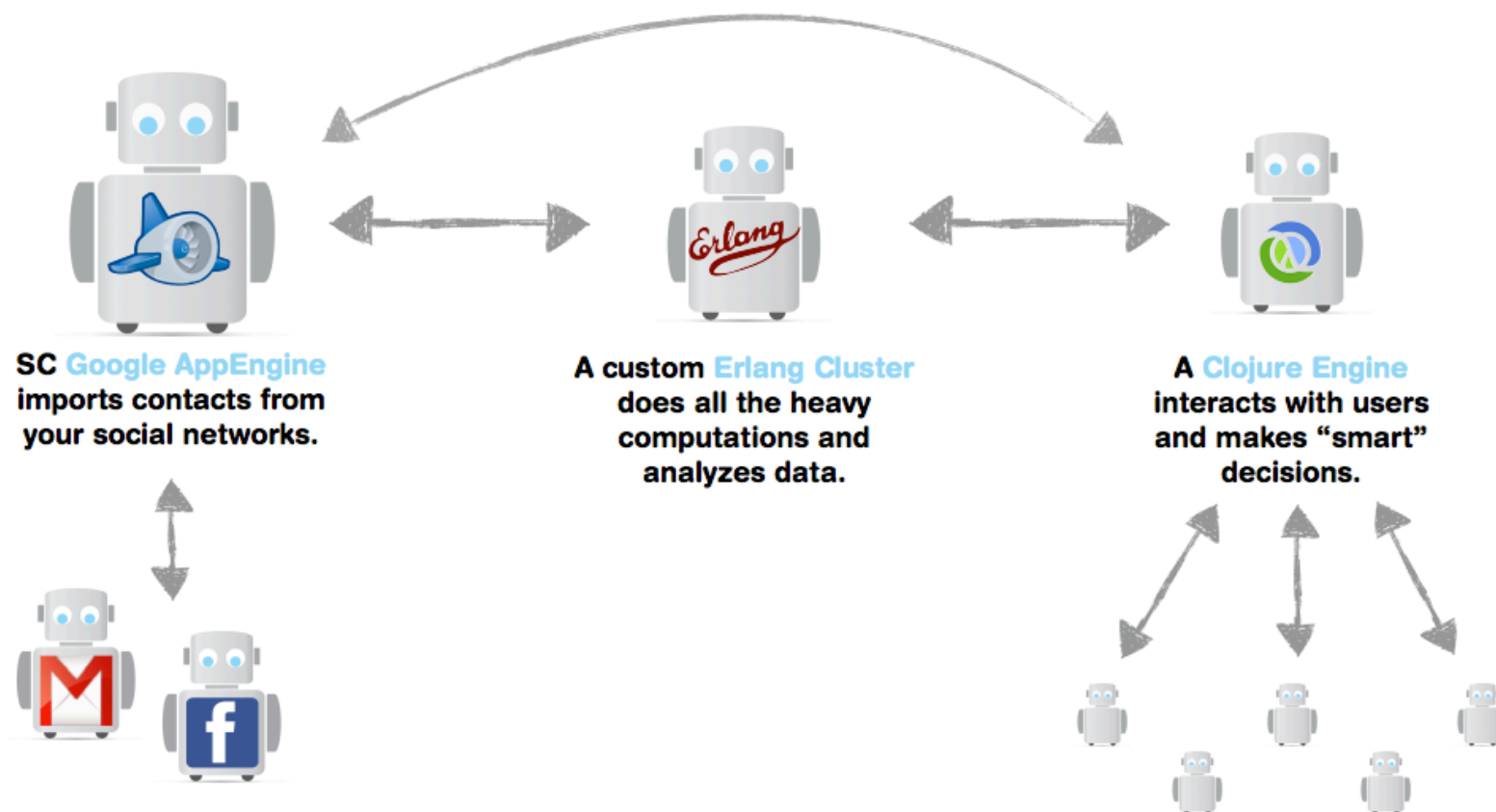


# Some code

```
"job": {  
    "id": "1",  
    "when": "now",  
    "callback_url": "autogenerated",  
    "security_hash": "secret",  
    "language": "python",  
    "num_of_nodes": "4",  
    "source": open("gmail_consumer.py", "r").read()  
}
```

# Overview

## What makes it work?





# How does it work?

**GAE tells the consumer to get the data**  
**Once the producers have finished, GAE tells**  
**Erlust to Rock n Roll (fire off the consumers**  
**that crunch the data in parallel)**  
**Once a consumer is finished it sends the**  
**crunched to GAE via JSON requests or**  
**Remote API**

# The best part?

**Erlust does not know about the code and  
we never tell it to checkout biz logic code  
GAE sends to Erlust the code to execute.**

**Even better we do this in the famous  
AppEngine one-click deploy**

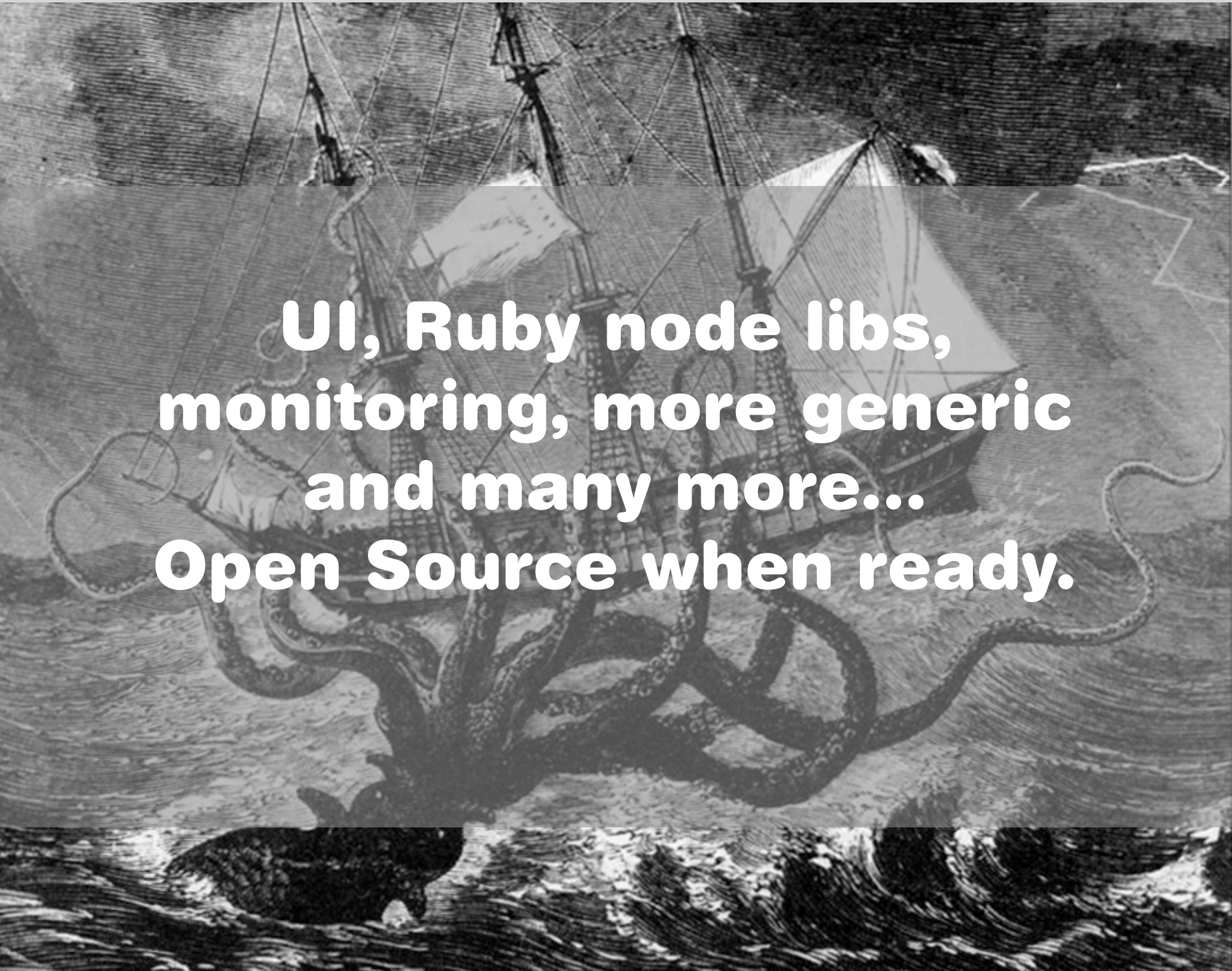
**VERY easy to use!**



# Some more code

```
def main():  
    node.ready()  
    queue = node.q_connect("fb_updates")  
  
    nd = node.q_get(queue)  
    while nd:  
        calc(nd):  
        nd = node.q_get(queue)  
  
    node.q_close(queue)  
    node.done()
```

# And now?



**UI, Ruby node libs,  
monitoring, more generic  
and many more...  
Open Source when ready.**



# Thanks for watching

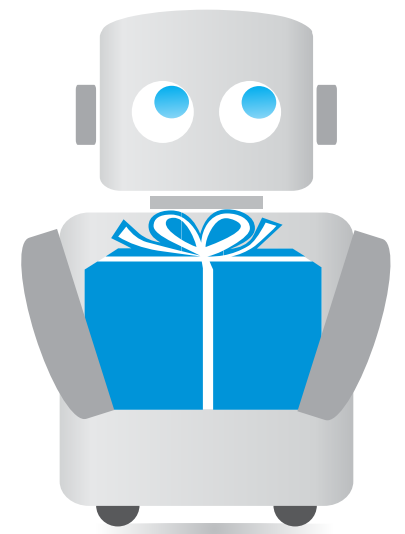
**..and a small present!**

**Be the first to check  
SocialCaddy alpha  
but please be gentle..!**

**alpha.socialcaddy.com**

**code: erlangrulez**

**stay tuned!**  
**blog.socialcaddy.com**



**Questions?**  
**jon@socialcaddy.com,**  
**panos@socialcaddy.com,**  
**nikos@socialcaddy.com**

