

ZeroMQ

MAKE YOUR WEB APP DO HEAVY SHIT

#SYPY

@CHEWXY

MESSAGE QUEUES – A PRIMER

- Facilitates inter-process communications
- Often used in decoupling heavy processing from live requests
- Poor man's distributed processing

MESSAGE QUEUES – A PRIMER

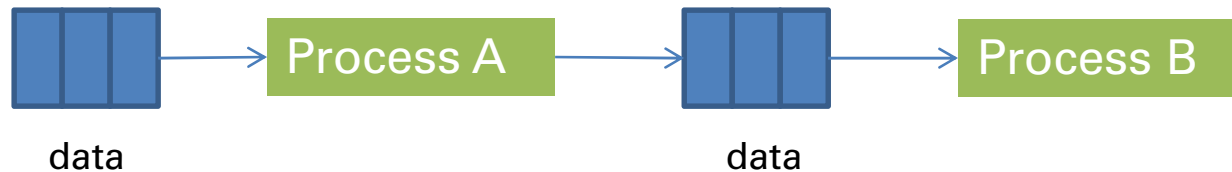
- Take this: $[[1, 2, 3], [4, 5, 6], [7, 8, 9]]$
- Transform into: $[1, 4, 9, 16, 25, 36, 49, 64, 81]$

MESSAGE QUEUES – A PRIMER

```
def processA(data):  
    return map(lambda x: [y ** 2 for y in x], data)  
  
def processB(data):  
    return reduce(list.__add__, data, [])  
  
def main(data):  
    data = processA(data)  
    print processB(data)  
  
>>> data = [[1,2,3],[4,5,6],[7,8,9]]  
>>> main(data)  
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

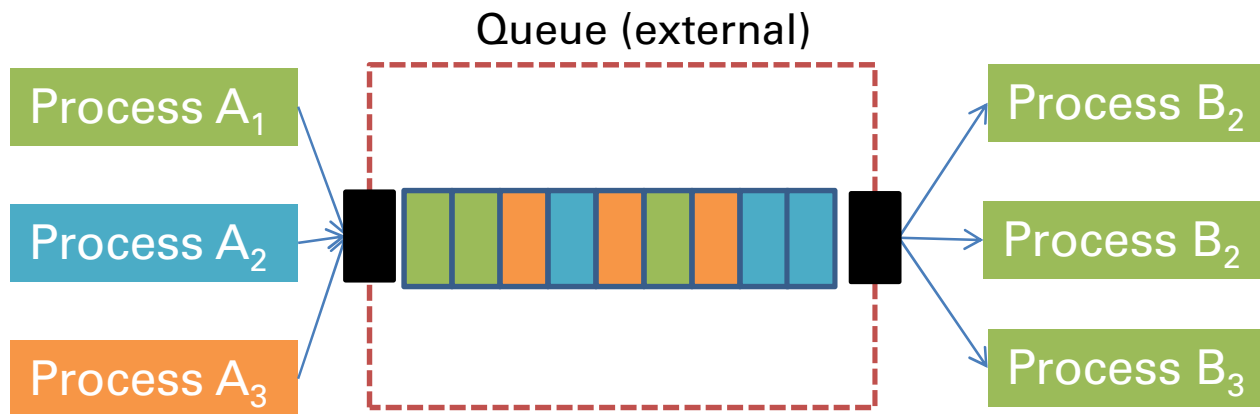
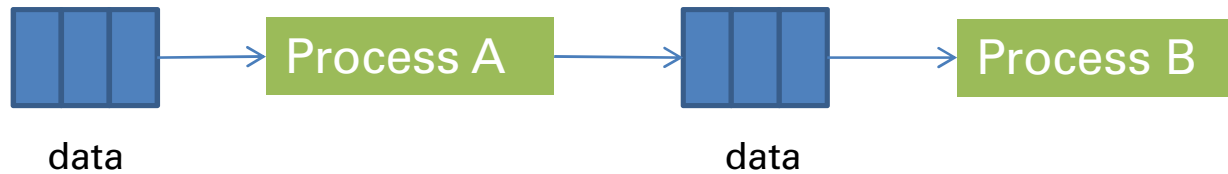
Sure it's easy with functional programming tools.
Also it looks fancy and complicated, which is why it's in
this slide

MESSAGE QUEUES – A PRIMER



From the example above, the function `processA` has to perform its map on every element in the data list first before passing on data to `processB`

MESSAGE QUEUES – A PRIMER



MESSAGE QUEUES – A PRIMER

- RabbitMQ
- IBM Websphere *
- Beanstalk'd
- Celery (Python based) *

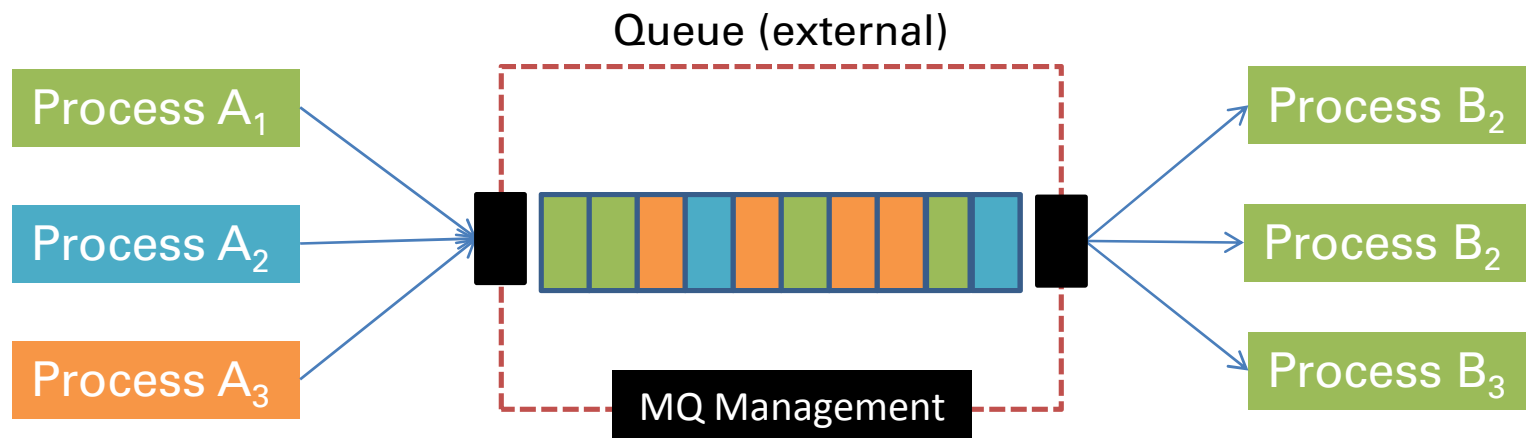
Some are technically not message queues but for the purposes of SOA and the like, but for the purposes of this talk, let's consider them MQs

ZEROMQ

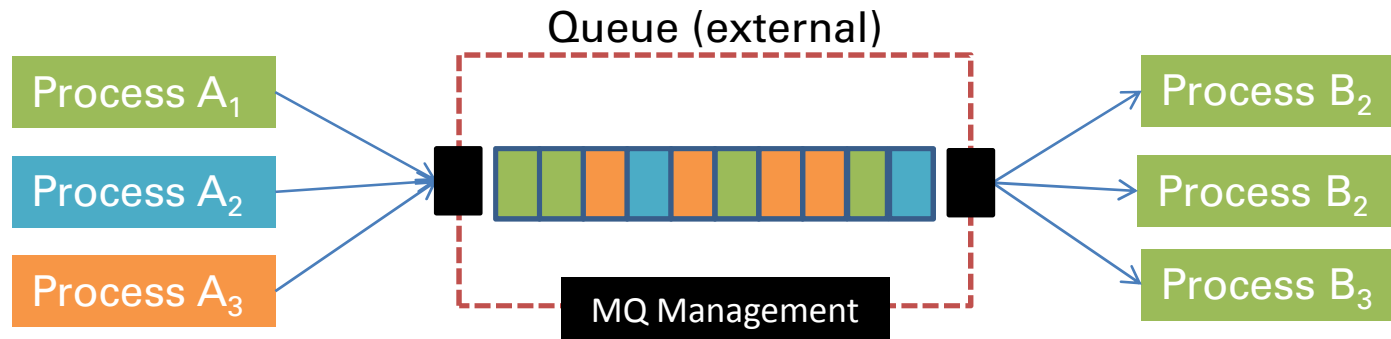
- Literally, 0 MQ (or rather, no broker)
- ZeroMQ uses sockets
- I think of ZeroMQ as a network stack with built in MQ

ZEROMQ

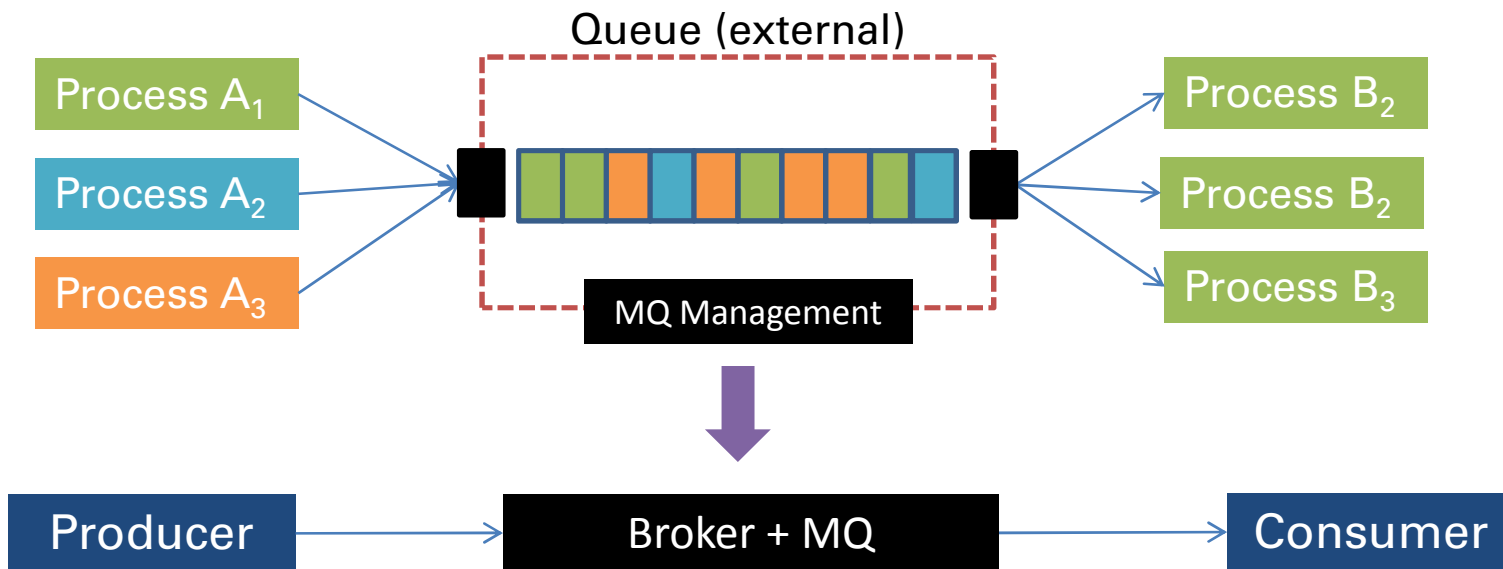
- Literally, 0 MQ (or rather, no broker)
- ZeroMQ uses sockets
- I think of ZeroMQ as a network stack with built in MQ



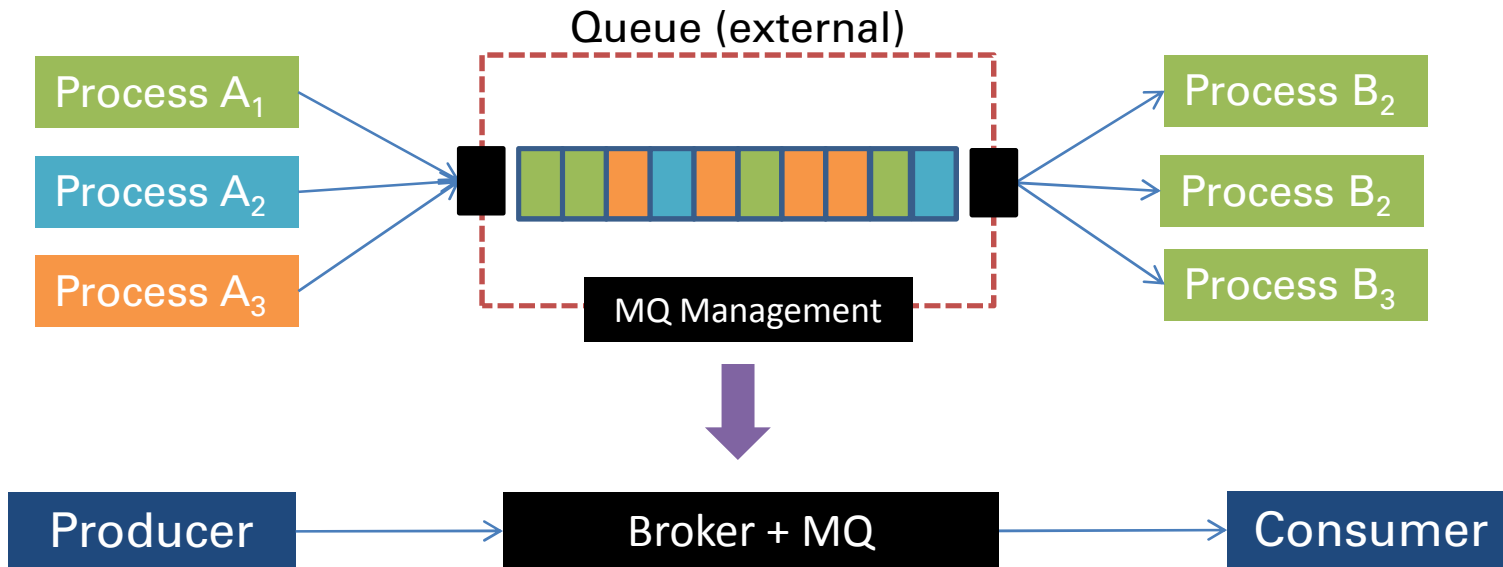
ZEROMQ



ZEROMQ



ZEROMQ



The Genius of ZeroMQ



ZEROMQ

```
import zmq
context = zmq.Context()
socket = context.socket(zmq.PUSH)
socket.bind('tcp://*:1234')

>>> socket.send('test')
```

```
import zmq
context = zmq.Context()
socket = context.socket(zmq.PULL)
socket.connect('tcp://*:1234')

while True:
    msg = socket.recv()
    print msg

test
```

If you are reading this during the presentation that means the live coding didn't go well

ZEROMQ

- Many patterns supported:
 - Push/Pull
 - Request/Reply
 - Pub/Sub
 - Many More!
- Many protocols supported:
 - TCP
 - IPC
 - InProc
 - PGM
- Automatic load balancing

ZEROMQ

- ZeroMQ is easy:
 - Wrote a push to talk chat app in < 10 minutes
 - ~ 60 lines of code with just one of the above pattern
 - Shows the concept of “sockets with Message Queues built-in”
 - See Examples

ZEROMQ IN ACTION

- As a superfast version of a traditional message queue
- To create a Service Oriented Architecture
- Personally, I like the UNIX Philosophy better.

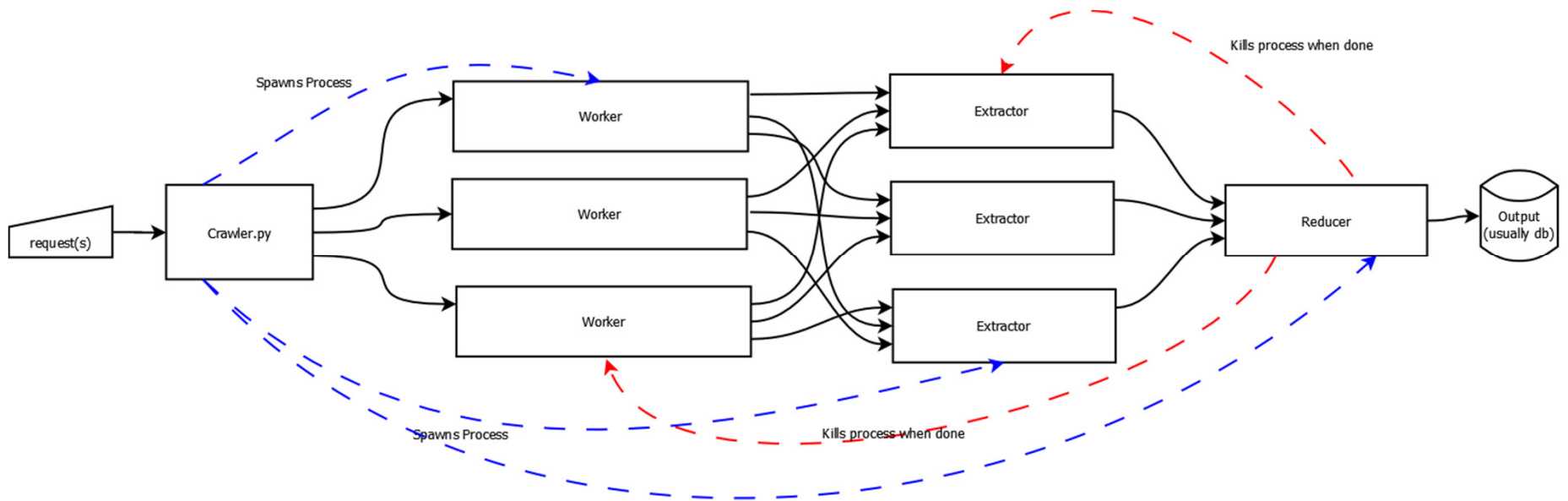
Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

ZEROMQ IN ACTION

My projects using ZeroMQ:

- edgeyo (<http://edgeyo.com>) – *project shelved*
- Strangers for Dinner (<http://strangersfordinner.com>) – *pivoting*
- A Brand Safety Product
- A Data Management Platform
- A Media Analysis Toolkit

ZEROMQ IN ACTION



DO HEAVY SHIT

- Spend time planning your architecture and network topology.
- You cannot tack on ZeroMQ like you tack on RabbitMQ
- Spread your load across many processes
- Not a poor man's distributed processing

MAKE YOUR SITE FAST

- POST Handlers
- When POSTs are made, reply the user instantly with cached data.
- Handle the POST information separately in the back end (by PUSHing your data to your POST handler app)
- If you need to communicate results with the user, *socket.io* is a good idea.

THANK YOU

- Examples:
<http://github.com/chewxy/SyPyOct2012>
- Email: chewxy@pressyo.com
- Twitter: @chewxy

p/s: follow me now and a unicorn will present bacon and whiskey to you tonight