Quick & Dirty IPC in Python

Chuck Fox

PyCon DC 2005

Python: Make IPC Easy

- Parallel Processing in Python
  - Threads are great but GIL is an issue
  - Forking subprocesses can use multiple CPU's
  - Memory Issues too
- But how to easily do IPC…

This is Python!

- Simple Things are Simple
  - Can we pass objects between processes?
  - Use pre-existing socket infrastructure
  - Use Cpickle
  - Use the power of Interface design

Two Sides

- Object Sender
  - The client
  - Pickles an object
  - Connects to UNIX named socket
  - Sends the object off
  - Operates inline with the client

Two Sides (2)

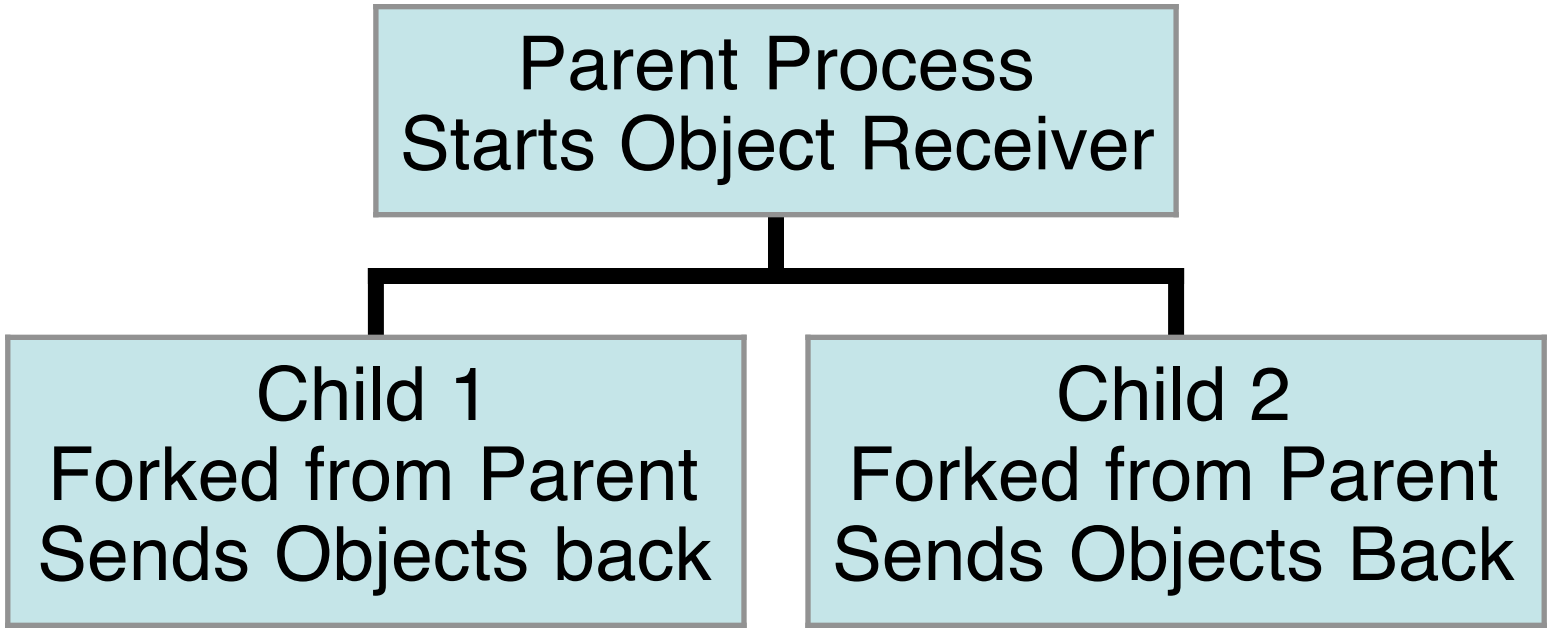
- Object Receiver
  - The server
  - Listens to the UNIX socket in a thread
  - Reads in pickled byte streams
  - Stores streams in a queue (1 object / connection)
  - Thread-safe poll () to get objects back into main thread

More Details

- Sanity:
  - Implement timeouts on sender & receiver
  - Client has retry in busy situations
  - Limits on queue size for memory
- Limitations:
  - Object Receiver will block while reading in each object
  - Usually designed for a small number of concurrent processes

Sample IPC flow

Sample Flow for IPC



The Upshot

- Does it work:
  - Yeah, in practice It's OK
- What else could be done?
  - Using the async core?
  - Making a non-blocking server?
  - Any off the wall recommendations?
- Now… onto the code.