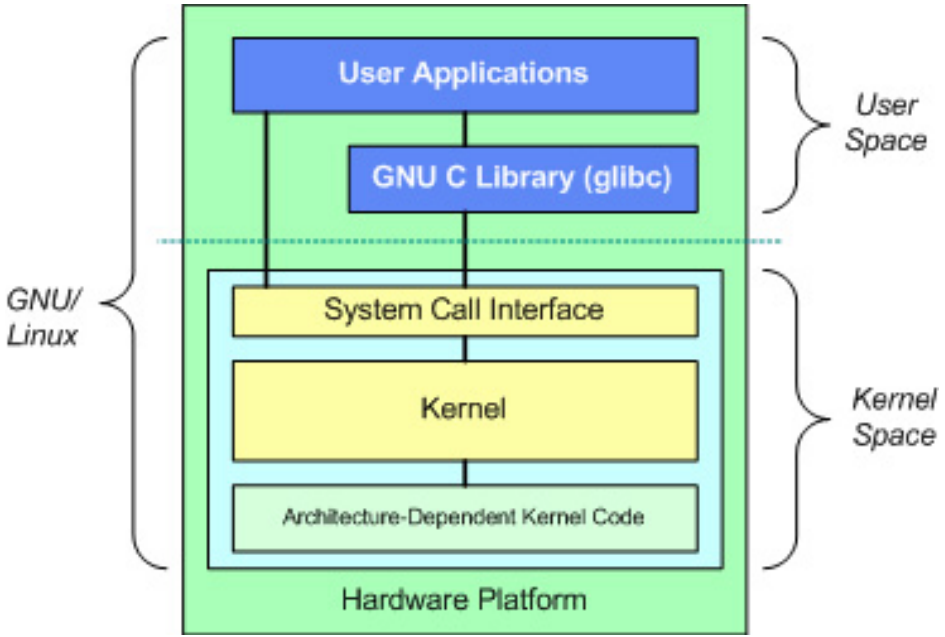
Non-blocking I/O with Gevent

by Praveen Kumar

Problem - Concurrency

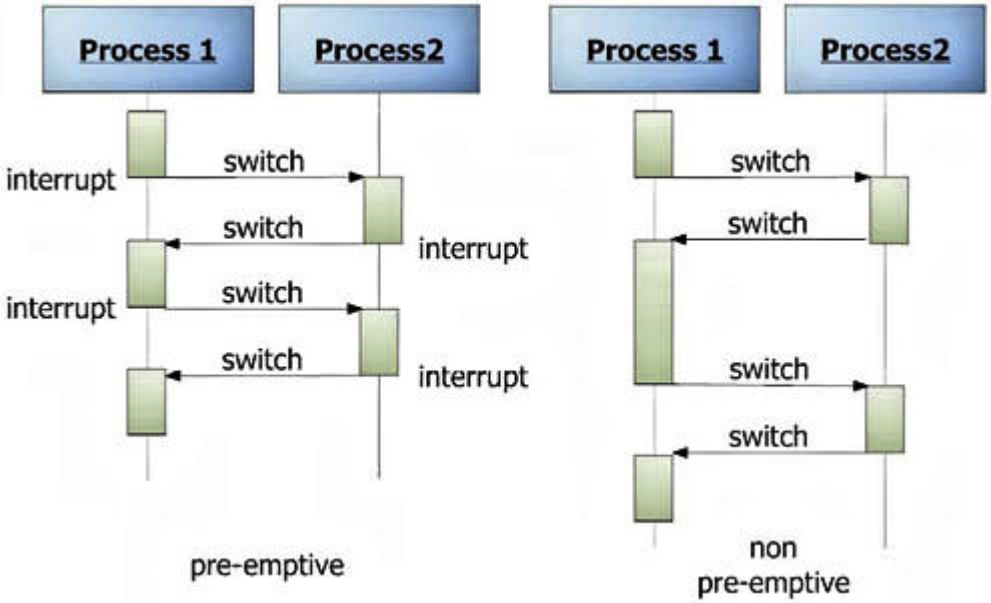
- Need to run multiple things at same time!
- Concurrency is not parallelism.
- Threading (preemptive scheduling)
- Pseudo-Threads (cooperative scheduling)
- Existing code needs to be concurrent!

User/Kernel space in linux



Source: http://knowstuffs.files.wordpress.com/2012/06/figure2.jpg

Scheduling types



Source: http://www.embedded.com/design/embedded/4204740/Getting-real--time--about-embedded-GNU-Linux

Preemptive scheduling

- Threads exhibit preemptive scheduling in kernel space.
- Threads/Processes added to run queue.
- Run queue length affects context switching.
- Higher context switches = higher CPU load.
- Alternatives?

Cooperative Scheduling

- Each task runs inside a green thread.
- Switches can happen in user space.
- There is only one thread visible to kernel.
- Kernel context switches are costly compared to user space context switches.
- Drawbacks: No number crunching!

Gevent NIO - intro

- Gevent is a coroutine based networking library
- Coroutines for suspend-resume mechanism.
- In gevent, non-blocking I/O is achieved by cooperative scheduling blocking calls.
- Uses event loop(libev, libevent) to register callback.

Gevent NIO - coroutines

- Gevent coroutines use *yield* idea internally.

- It is a mechanism to suspend your code and resume later.

- There are various types depending on how you deal with stack, generally your stack is saved in heap.

Gevent - intro

- Gevent works without any major changes to existing code.
- Core libraries are rewritten! co-op. sockets!

```
>>> import gevent
>>> from gevent import socket
>>> urls = ['www.google.com', 'www.example.com', 'www.python.org']
>>> jobs = [gevent.spawn(socket.gethostbyname, url) for url in urls]
>>> gevent.joinall(jobs, timeout=2)
>>> [job.value for job in jobs]
['74.125.79.106', '208.77.188.166', '82.94.164.162']
```

Gevent - monkey patching

- Even better, core libraries can be monkey patched.

```
>>> from gevent import monkey; monkey.patch_socket()
>>> import urllib2 # it's usable from multiple greenlets now
```

- Handy when there is existing code needing concurrency.

Gevent - monkey patching

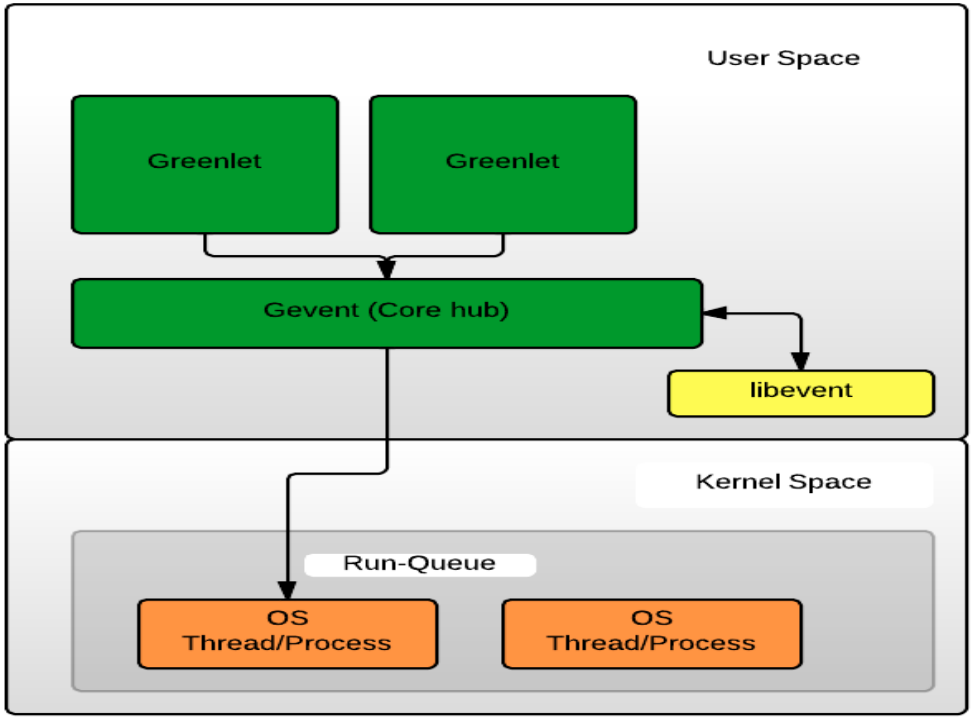
- patch_all() could prove evil, use selective patching as much as possible!

```
from gevent import monkey, sleep       # This sleep yields i.e non-blocking
monkey.patch_all()                     # This could bite you later!!!!!!!!!!!
import threading

class FooBar(threading.Thread):
    def __init__(self, parent):
        threading.Thread.__init__(self)

    def run(self):
        while True:
            self.doSomethingImportant()
            sleep(5)
```

Gevent - Full overview



Gevent - Actors Pattern

- Actors based concurrency? Subclass greenlet!

```
import gevent
from gevent.queue import Queue

class Actor(gevent.Greenlet):
    def __init__(self):
        self.inbox = Queue()
        Greenlet.__init__(self)

    def receive(self, message):
        raise NotImplemented()

    def _run(self):
        self.running = True
        while self.running:
            message = self.inbox.get()
```

Gevent - Actors Pattern

```
class Pinger(Actor):
    def receive(self, message):
        print(message)
        pong.inbox.put('ping')
        gevent.sleep(0)

class Ponger(Actor):
    def receive(self, message):
        print(message)
        ping.inbox.put('pong')
        gevent.sleep(0)

ping = Pinger()
pong = Ponger()
ping.start()
pong.start()
ping.inbox.put('start') # this starts the producer/consumer cycle
gevent.joinall([ping, pong])
```

Gevent - 3rd party libs

- Most of the extensions are easier to write with primitive constructs like queue, pool available in gevent (blocking/non-blocking).
- Web servers like uwsgi, gunicorn.
- Other libs pysnmp, pykka, async db drivers like psycopg!

Gevent vs Other NIO libs

- Alternatives (Twisted usually, Tornado rarely)
- ….and there is node.js (fibers/libuv)
- Gevent’s synchronous API is great (subj.)
- Code is readable.
- Greenlets abstract all the callbacks - you can get a handle on them when you need it.

Gevent - drawbacks

- Monkey patching *can* be evil! (import order)
- Only one OS thread, anything that blocks CPU is not allowed. (there are work arounds)
- Need pycharm to debug gevent.
- It is deterministic, non-determinism can creep in!

Future of NIO in python

- Like any prediction it is subject to change!
- Python 2.x (twisted, tornado, gevent, tulip > 2.6)
- Python 3.x (tulip, asyncio > 3.4 in stdlib)
- I think gevent will be supported in python 3.x

Q & A

- Disclaimers:
  - no kernel hacker
  - not tried uwsgi
  - played with node.js, twisted but not tornado

Take away thoughts...

- Cannot use cooperative scheduling for number crunching. (Use multiprocessing, gipc)
- Gevent gives code a synchronous feel which abstracts callbacks making code more readable.
- Mashing up data from disparate sources(like HTTP) is an ideal use case for gevent.