Understanding GIL

Senthil Kumaran

orsenthil@gmail.com

Scipy.in 2009

How I use python?

- Python coredev - stdlib.
- At work – Akamai.
- Fun.

Thanks!

- David Beazly – Inside the Python GIL talk.
- Pictures are from his slides.

Outline of the talk

- Threads.
- Python Interpreter.
- Python Threading Support.
- What is GIL.
- GIL Behavior.
- Newgil in py3.2
- Parallelism – Other ways.
- etc

Threads

- Parallel units of execution within a process.
- All threads share the same memory.
- Eat and think around the table, sometimes sharing the fork and knife.
- Problems with Threading involve the Shared Resources and Thread Synchronization

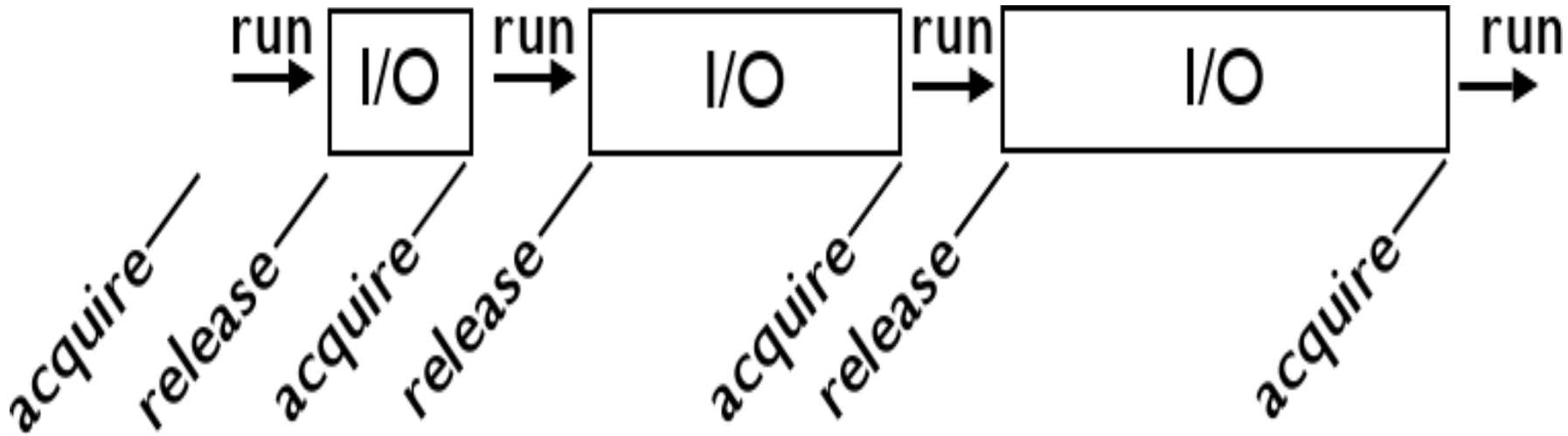
Python Programming language

- Different Implementations.
  - Cpython
  - Jython
  - IronPython.
- We are going to discuss Cpython implementation.
- Bytecode based.

Python Interpretor

- Interpreter is a single process.
- Within that process, only one thread can be interpreting python byte codes at one time.
- If that thread is blocked (waiting or I/O), another thread can be run. Some C extensions release GIL so that threads can run concurrently.

This is how


run
I/O
run
I/O
run
I/O
run
acquire
release
acquire
release
acquire
release
acquire

Python and Threading Support

- A standard library module called threading.
- But the execution of threaded programs is very basic.
- You have to define the synchronization mechanism.

What is Python Thread?

- Python threads are real system threads.
  - POSIX threads (pthreads)
  - Windows threads
- Fully managed by the host operating system.
  - All the scheduling/threading switching.
- Represent threaded execution of the Python inpterpretor process (written in C).

Thread Creation

- Python threads simply execute a “callable”
- The run() method of Thread
- Inside, the Python creates a small data structure containing some interpreter state.
- A new thread (pthread) is lauched,
- The thread calls PyEval_CallObject. It is just a C function call that runs the Python callable.

Thread Specific State

- Each thread has its own interpreter specific data structure (PyThreadState)
  - Details of the Thread currently under execution.
- The interpreter has a global variable that simply points to the ThreadState strcture of the currently running thread.
- /* Python/pystate.c */
- Operations in the Interpreter implicitly depend on this variable to know the state.

What is GIL

- GIL Serializes the access to most part of the interpreter.
- Only one thread is running at a time.
- The GIL ensures that each thread gets exclusive access to the Interpreter internals when its running.

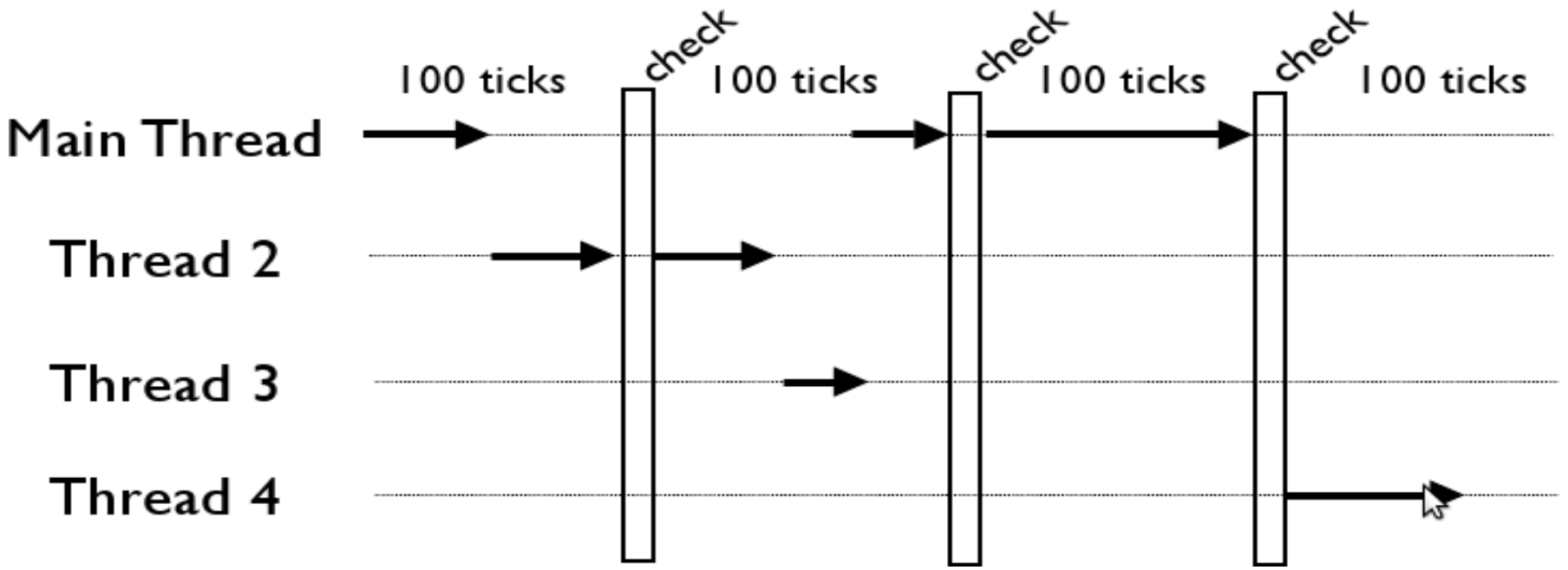
GIL Behaviour

- Its simple. Threads hold the GIL when running.
- However, they release it when blocking for I/O.
- It is coperative multi-tasking.

CPU Bound process

- To deal with CPU bound threads that never perform any I/O, the interpreter periodically performs a “check”.
- By default, every 100 interpreter “ticks”
- sys.setcheckinterval
- The check interval is independent of thread scheduling.

sys.setcheckinterval



The Periodic Check

- Release and Reacquire GIL.
- Signaling the OS to run other thread.
- Other threads get to run.. (somewhat)

All of these is for py < 3.2

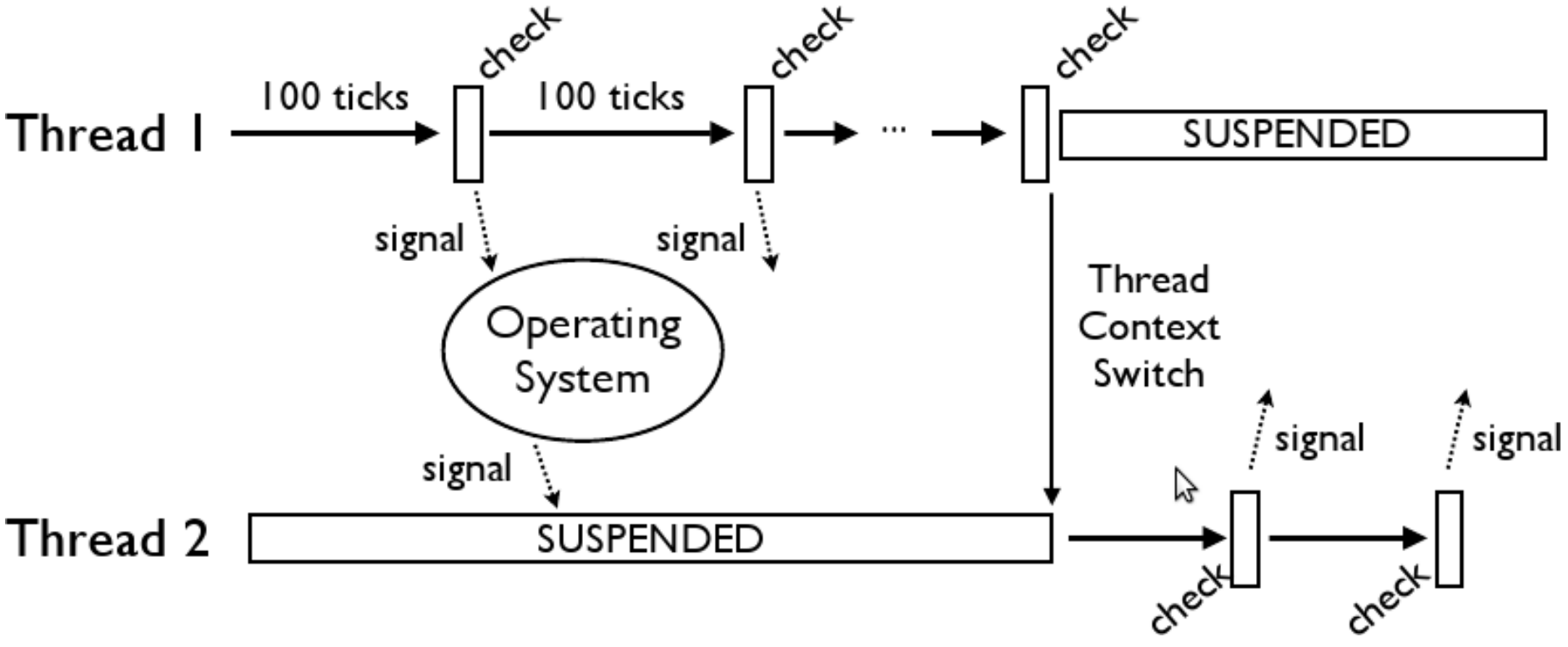
Interpreter ticks

- They are not time based (for < py3.2)
- Ticks are uninterruptible
- Long Operations can block everything.

Thread Scheduling

- Interpreter does not do any thread scheduling.
- There is no notion of thread priorities, preemption, round-robin scheduling.
- All thread scheduling is left to the host operating system.
- There comes the problem with signals. Only main thread handles signals and interpretor cannot schedule it execution, but has to wait for OS to schedule it when the any signal arrives.

Interpreter does NO thread scheduling



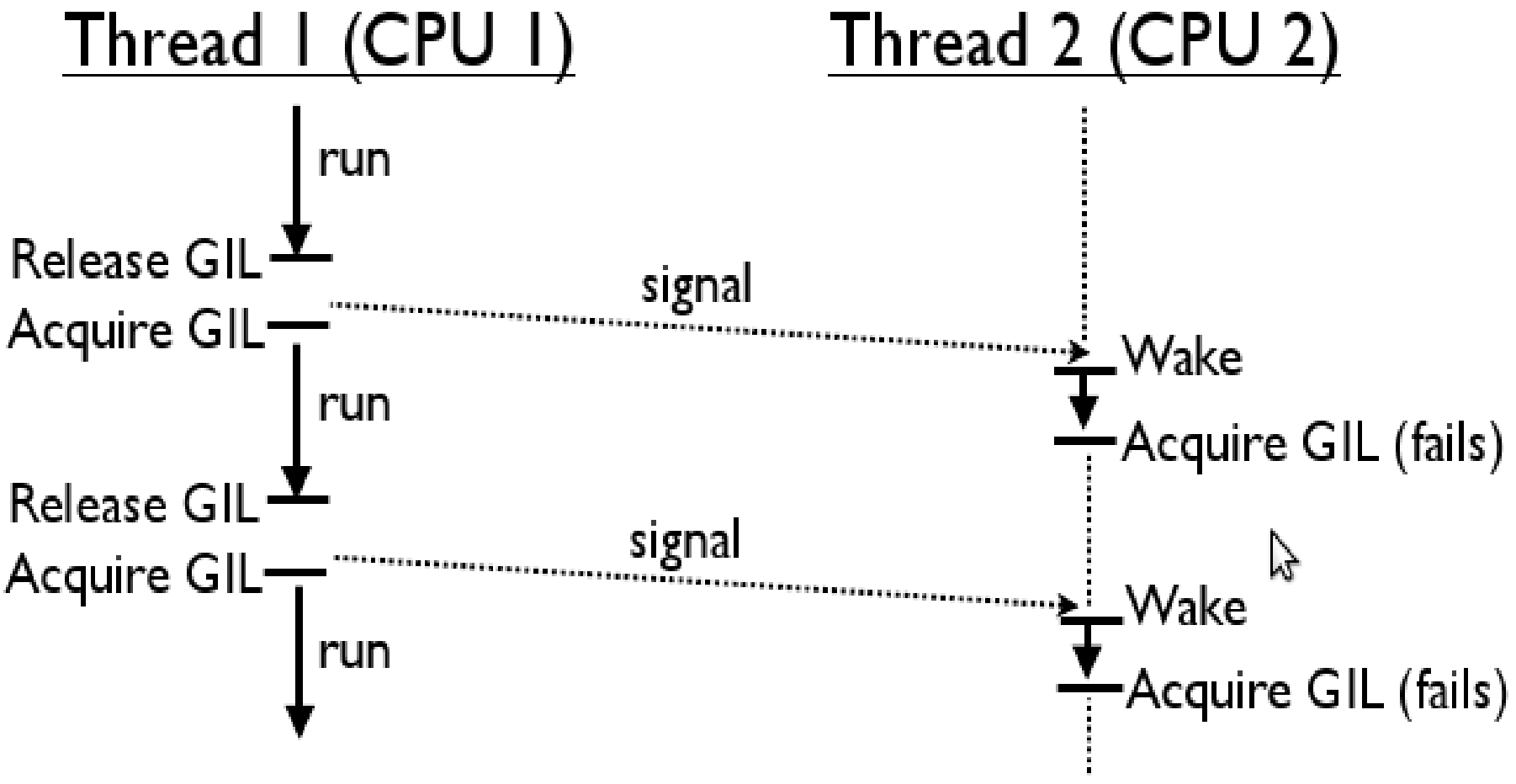
More details on GIL

- Its not a simple mutex.
- It is some kind of semaphore.
- The acquisition and release is based on the signals from the OS.
- OS does the scheduling based on priorities like CPU bound task has low priority and IO bound tasks have higher priority.

Multicore GIL contention

- CPU bound operations get scheduled simultaneous.
- And they have a GIL battle in the Interpeter.

GIL Contention



Newgil implementation

- Antoine Pitrou – Oct 2009
- Ditch the switching based on tick in the newgil implementation. (No Py_Ticker based count)
- Use fixed intervals (5 milliseconds) for the main thread to release the GIL.
- This fixed interval switching has significant benefits on thread contentions.
- Forced thread switching. Improves the thread switch latency.

ccbench

- Pure Python workload (PI Calculation).
- C workload which does not release GIL (re.search)
- C workload which does release GIL.

Other Concurrency Options

- Practical programming scenarios (IO and CPU Bound)
- Multi processing.
- C extension that can release GIL.
- Asynchronous event based – Great for Networking applications.
- Coroutines – python 2.5 where you can pass values to generators.
- Other Python implementations. Jython, IronPython, Unladen Swallow.

Thanks

- orsenthil@gmail.com
- http://uthcode.sarovar.org
- Search or David Beazly talk on Inside the Python GIL.
- Newgil implementation is currently in py3k branch (which will become python 3.2, est. by Jun 2010)

Notes

- sys.{set,get}checkinterval
- gil_locked and gil_mutex
- gil_drop_request
- gil_cond variable (waiting for interval)
- switch_cond check for gil_last_hold.