High-load real time apps with Python. Tips and tricks

Solving an issues which might occur using it

Dmitry Karpov
Software Engineer
Wargaming

Motivation to use Python in high-load real time apps

Why Python?

Performance

Expressive power

Median Hours to Solve Problem

Development Cost

Feature implementation cost

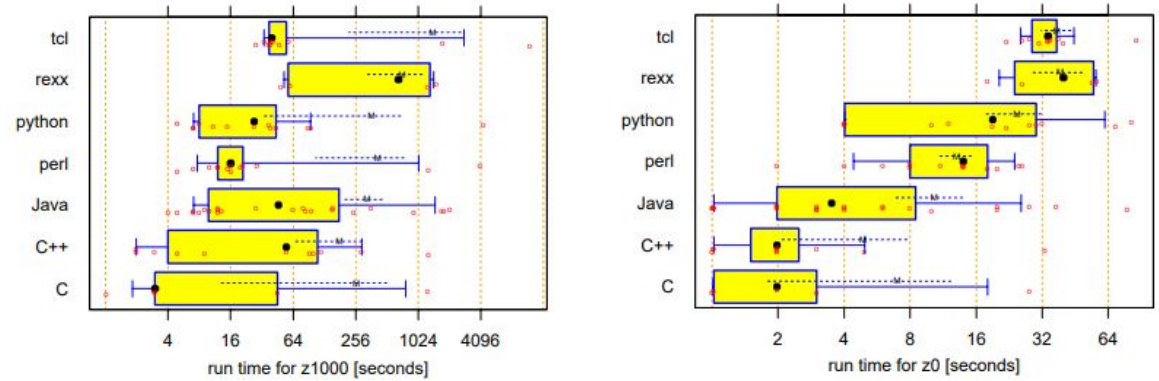
Debug complexity minimization

Motivation

Agenda

Performance

- Slower, but it doesn't matter for non-heavy calculations
  - https://www.python.org/doc/essays/comparisons/
  - https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/python.html
- Python memory management limits vectorization
- But JIT and 3rd party libraries bindings are solving heavy calculations issues partially



Source: Prechelt, An emperical comparison… , http://page.mi.fu-berlin.de/~prechelt/Biblio/jccpprtTR.pdf

Expressive power

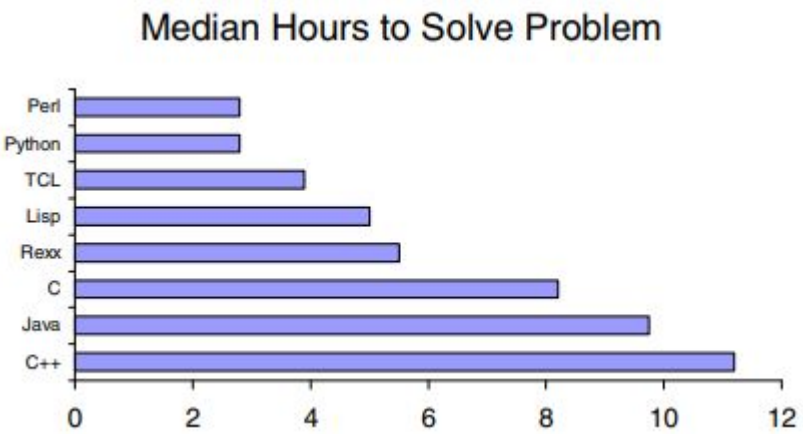
LANGUAGE	LEVEL RELATIVE TO C
C	1
C++	2.5
Python	6

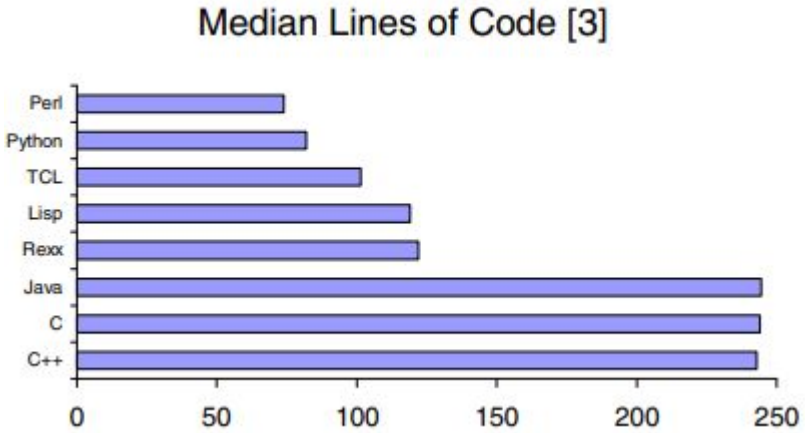
Source: Code Complete, 2nd Ed

- Code conciseness and readability
- Development terms are proportional to expressive power in inverse ratio
- Production cost ~ development terms

- Hint: It's almost impossible to obfuscate it due to it's architecture

Median Hours to Solve Problem





Source: Prechelt, An emperical comparison… , http://page.mi.fu-berlin.de/~prechelt/Biblio/jccpprtTR.pdf
Garret, Lisp as an Alternative to Java , http://www.flownet.com/gat/papers/lisp-java.pdf
Compiled by https://www.connellybarnes.com/documents/language_productivity.pdf

- 300-400 thousands online in average
- 10 arenas per second in evening peak
- ~10^4 events sent per client during a battle
- 815 000 on the Russian portal in spring 2013 in peak
- 4,5 M weekly unique visitors on European World of Tank Portal (exc. Russia)

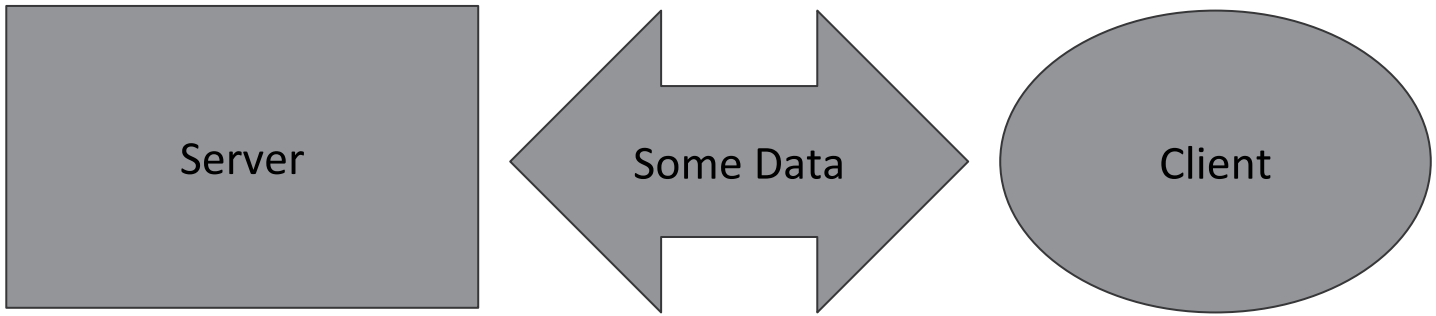
What high-load and real-time means

in terms of this presentation

Potential issues

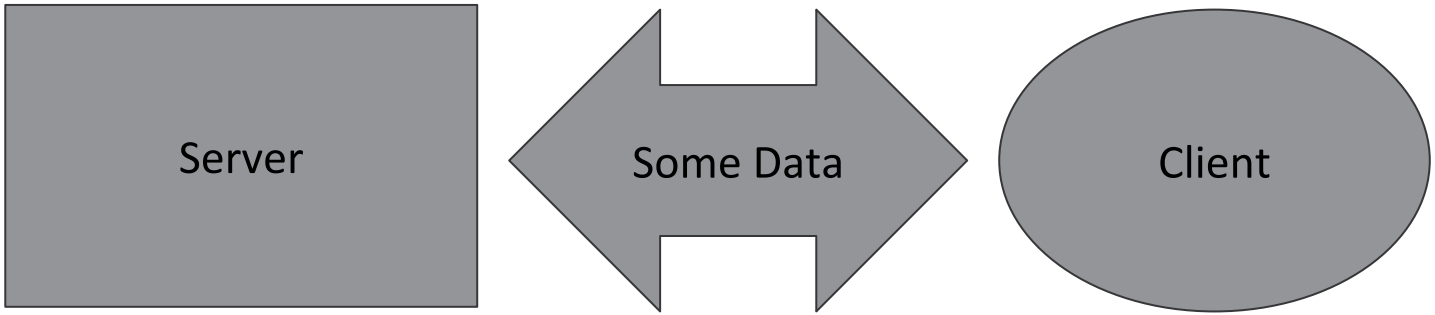
When using Python as main PL for your real time app

Potential issues

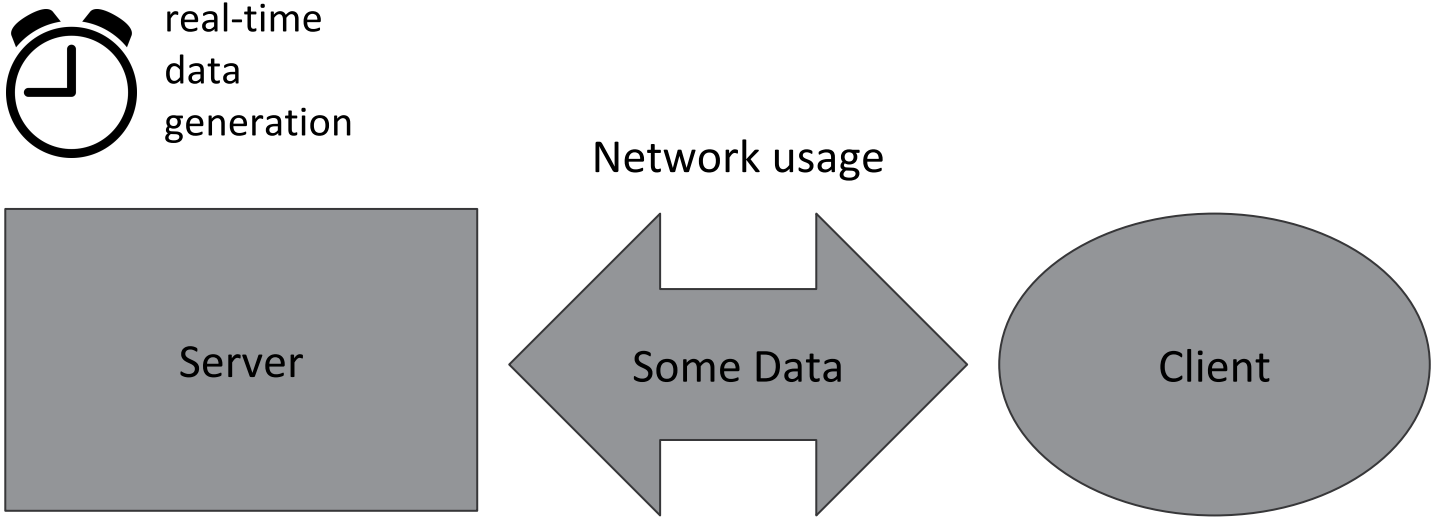


Potential issues

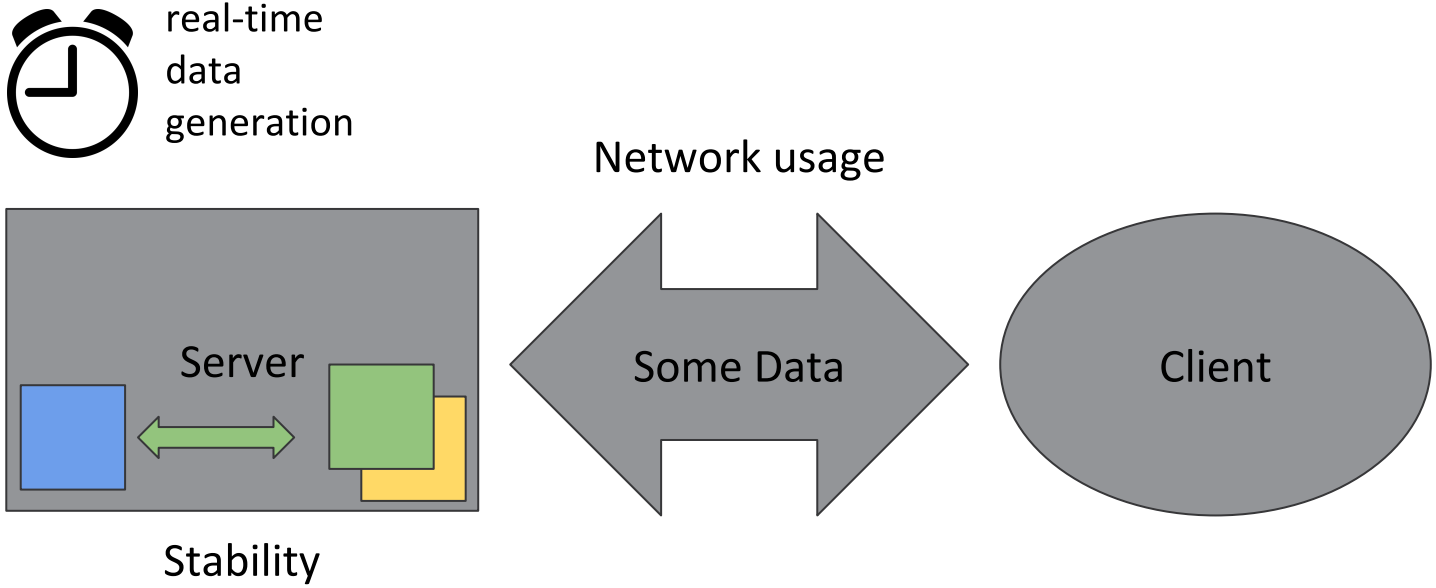
real-time
data
generation



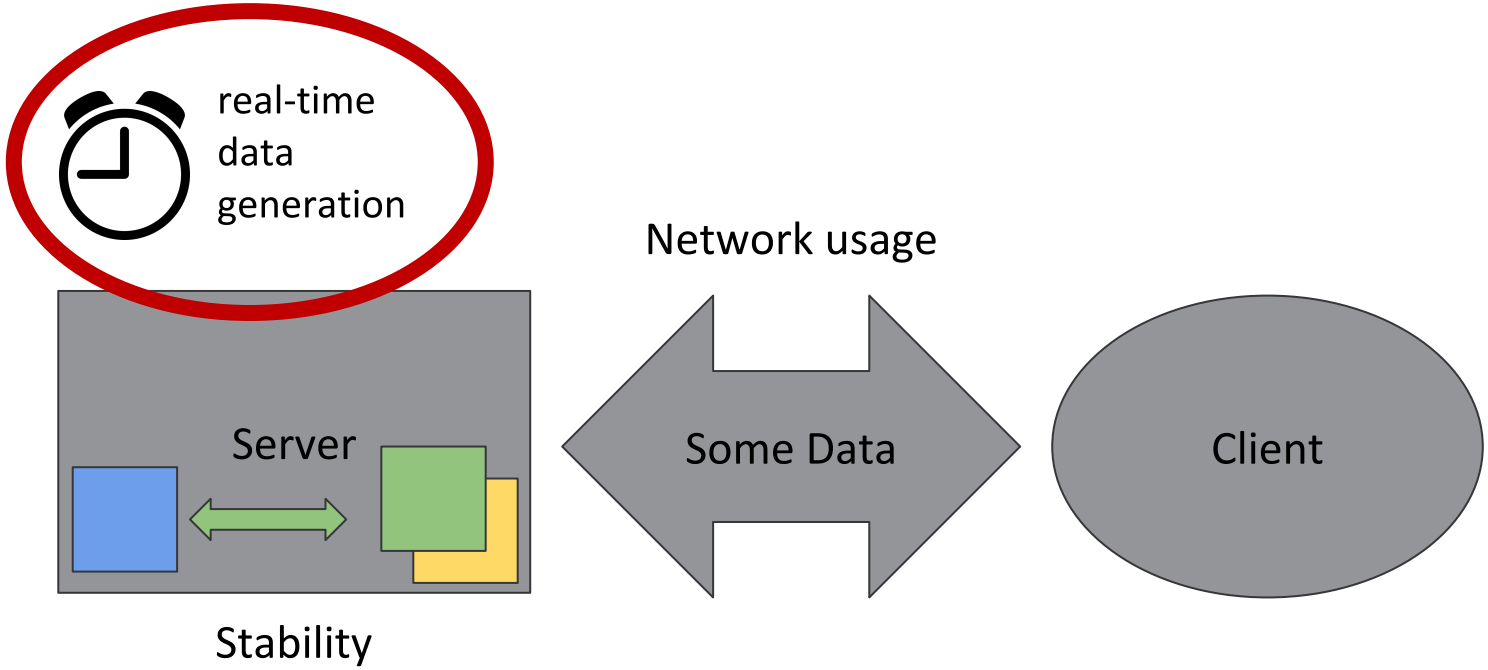
Potential issues



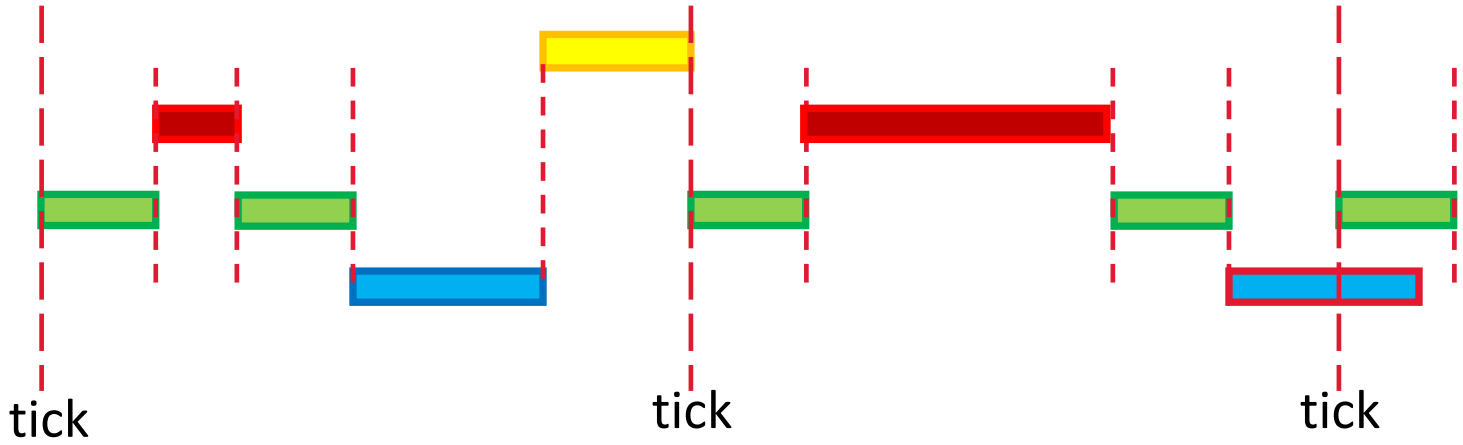
Potential issues



Potential issues



Issue #1: Real-time maintenance. GIL



Issue #1: Real-time maintenance. GIL

- Preemptive multitasking:
  - 100 bytecode instructions in Python 2
  - 15 milliseconds in Python 3

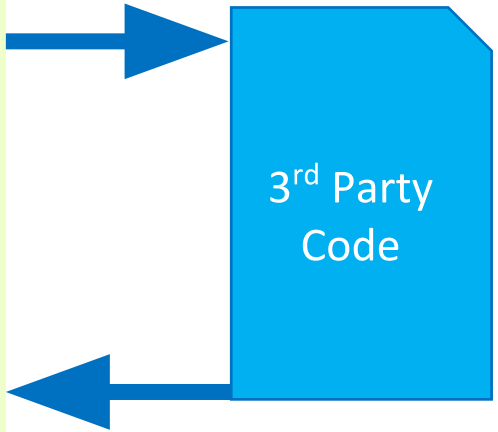
The `interval` value is available for the user to read and modify using the Python API `**sys.{get,set}switchinterval()**`.

Source: **ceval_gil.h**

Issue #2: Real-time maintenance. Non-Python calls


```
def myfunc(alist):
    return len(alist)
```

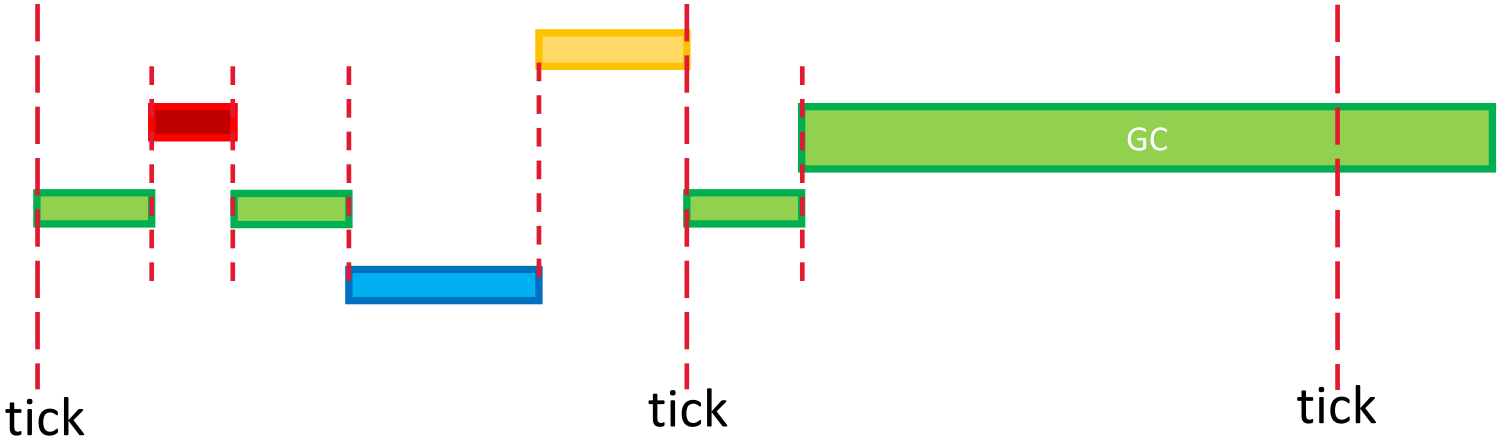
```
LOAD_GLOBAL 0 (len)
LOAD_FAST 0 (alist)
CALL_FUNCTION 1


# GIL is free here


RETURN_VALUE
```



Solution:

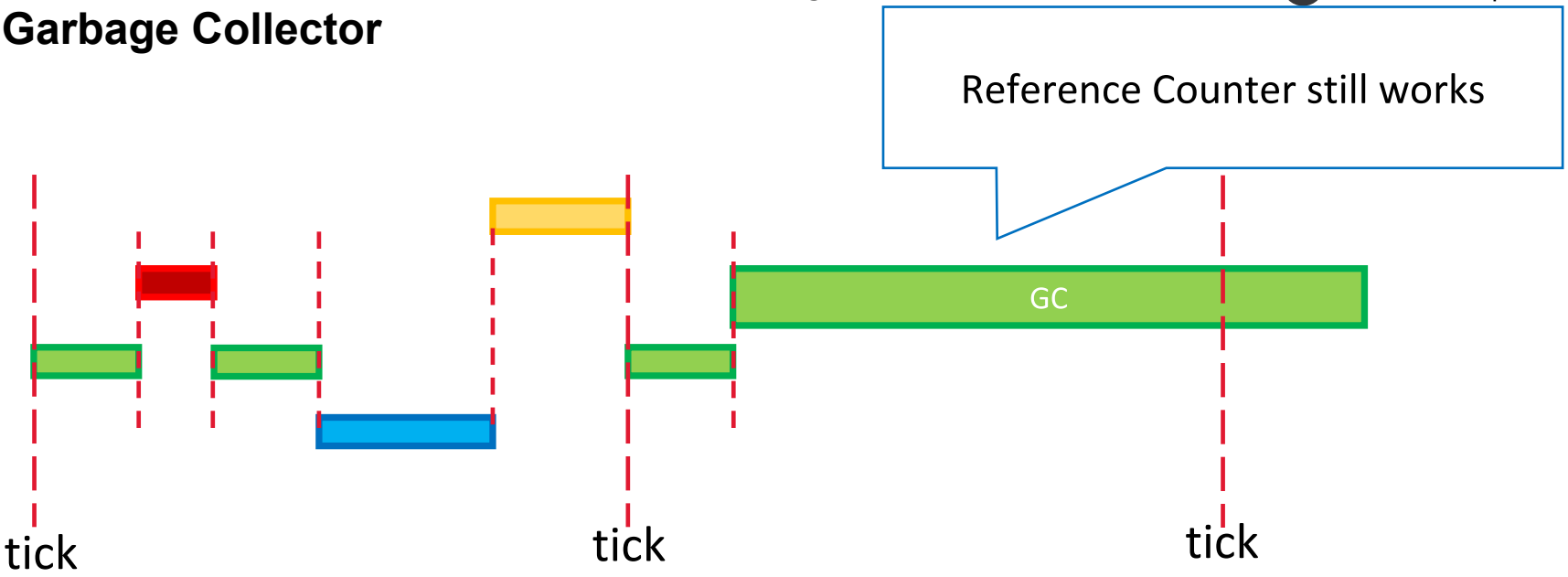
Reviews
Gatekeeping
Performance Tests
Quality Assurance

Issue #3: Real-time maintenance. Heavy calls. Garbage Collector



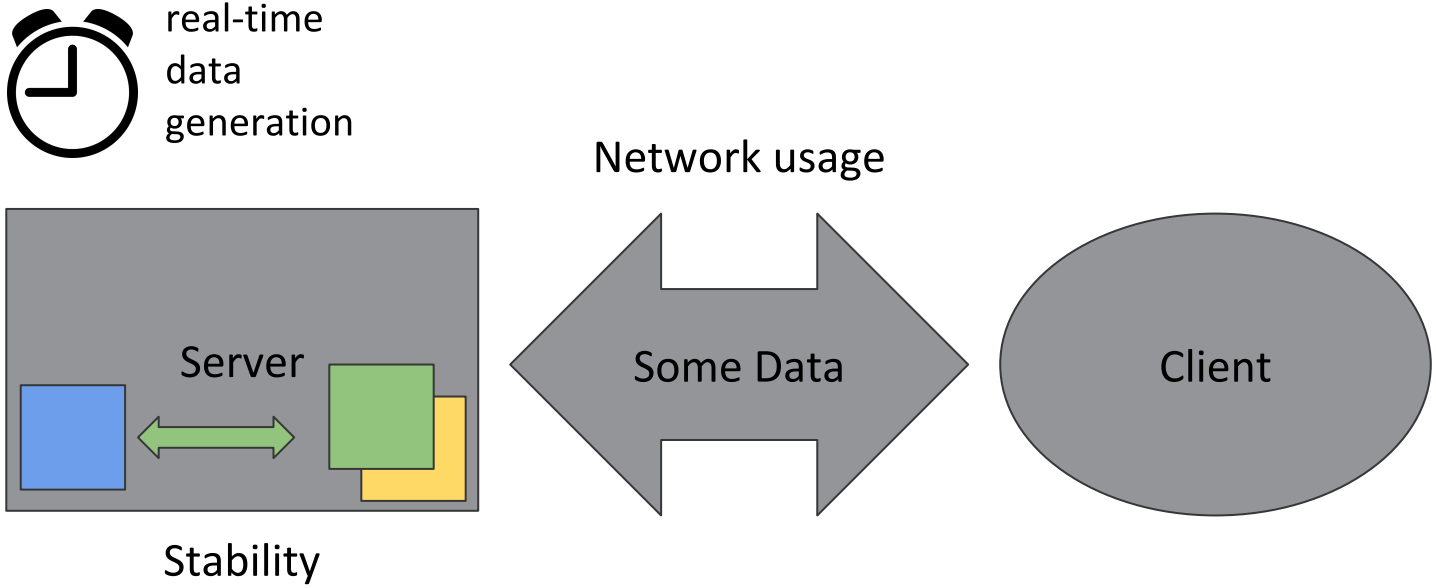
An expensive operation that can occur at any time, blocking the main thread in the server applications

Issue #3: Real-time maintenance. Heavy calls. Garbage Collector

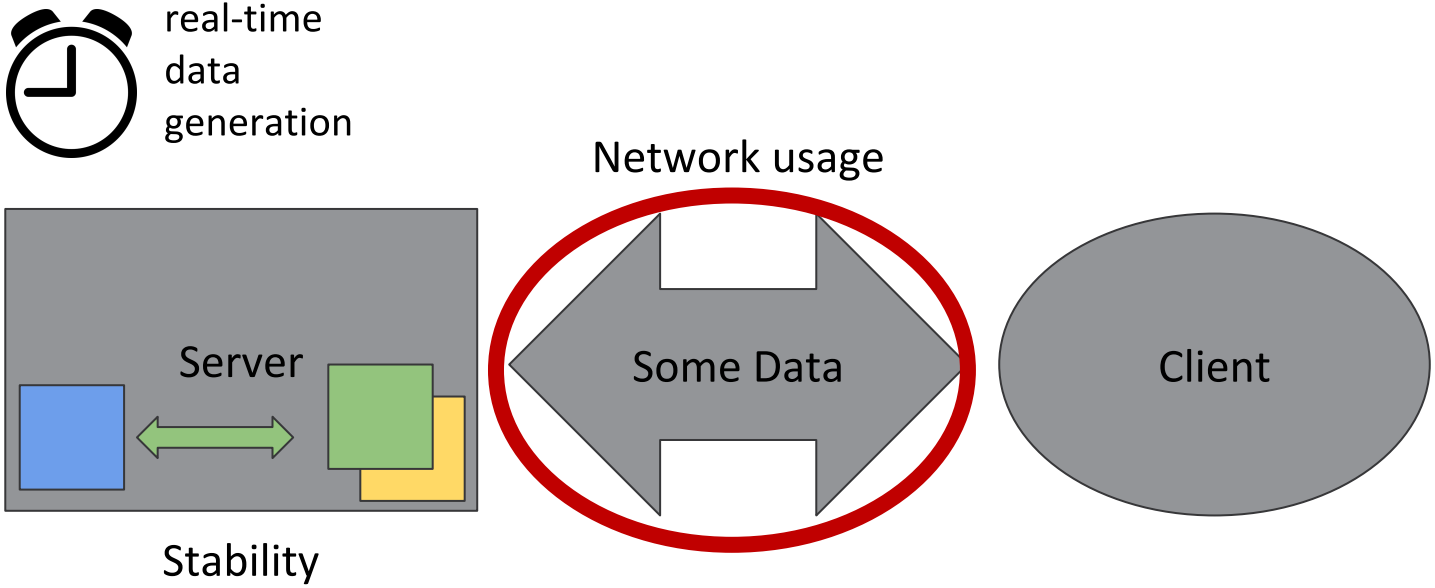


An expensive operation that can occur at any time, blocking the main thread in the server applications

Potential issues



Potential issues



Issue #4: Variable types

Python variable types are not declared

we have to use variable's type information

- to optimize network usage
- for proper and fast unpacking
- to use DB efficiently

```
class Packer:
    def __init__(self, *metaData):
        self._metaData = tuple(metaData)

    def pack(self, dataDict):
        l = []
        for index, entry in enumerate(self._metadata):
            name, tType, default, packer, aggFunc = entry
            v = dataDict.get(name, default)
            if v is None:
                pass
            # other checks for data to be fit with meta
            elif type is not None and not isinstance(v, type):
                v = tType(v)
                if v == default:
                    v = None
            l[index + 1] = v

_RESULTS = Packer(
    # Some results.
    ("example", int, 0, None, 'skip'),
)

results = {"example": 1}
resultsPacked = (zlib.compress(cPickle.dumps(_RESULTS.pack(results), -1)))
resultsReceived = _RESULTS.unpack(cPickle.loads(zlib.decompress(resultsPacked)))
```

Issue #4: Variable types. Examples

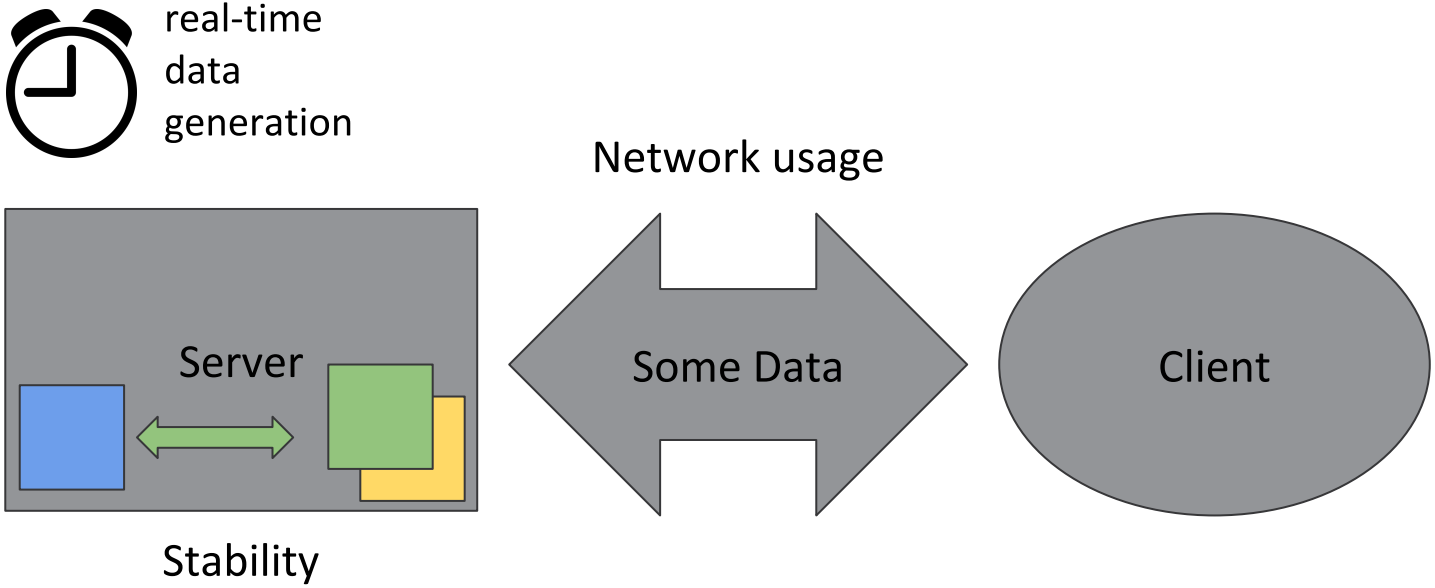
- DB (in avsc schemas for example):

```
{
    "type": "record",
    "name": "account_name",
    "namespace": "some_namespace",
    "doc": "Record for an account",
    "fields": [
        { "name": "account_property", "type": "string" },
    ]
}
```

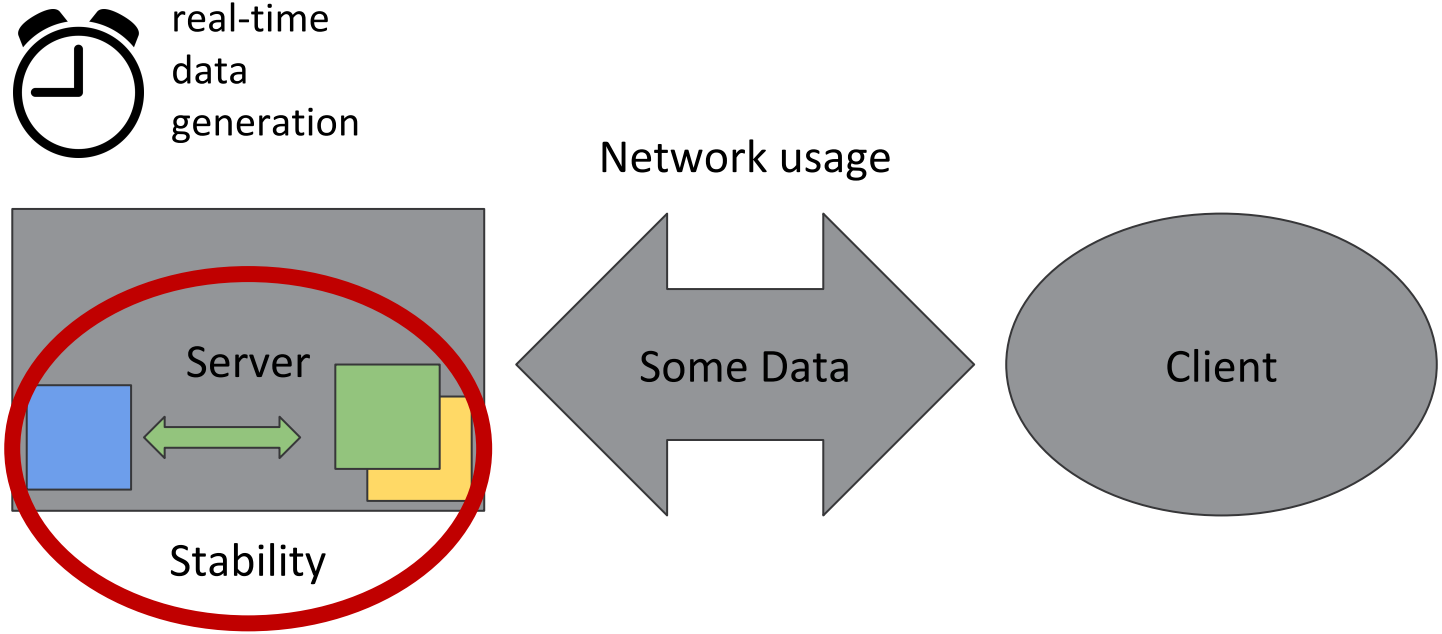
- Game Engine (in xmls)

```
<!-- Account -->
<name>
   <Type> STRING </Type>
   <Flags> SOME_FLAG </Flags>
   <Persistent> true </Persistent>
   <DatabaseLength> 96 </DatabaseLength>
</name>
```

Potential issues



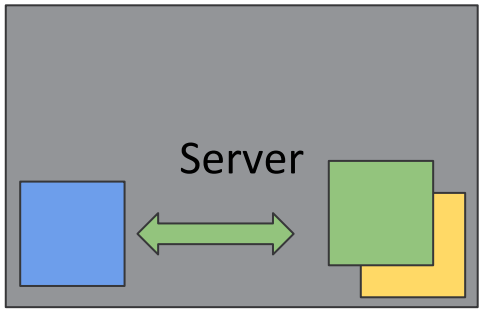
Potential issues

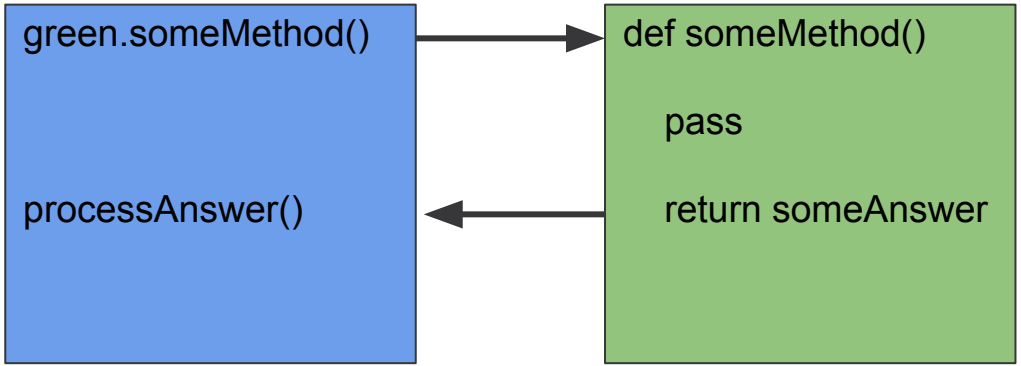


Issue #4: Stability. Fault tolerance

Additional information

Real-time systems have to be fault tolerant and are able to continue work after failure

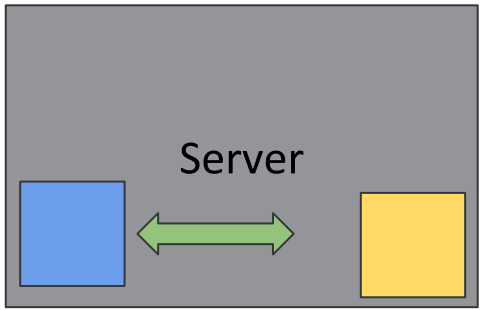


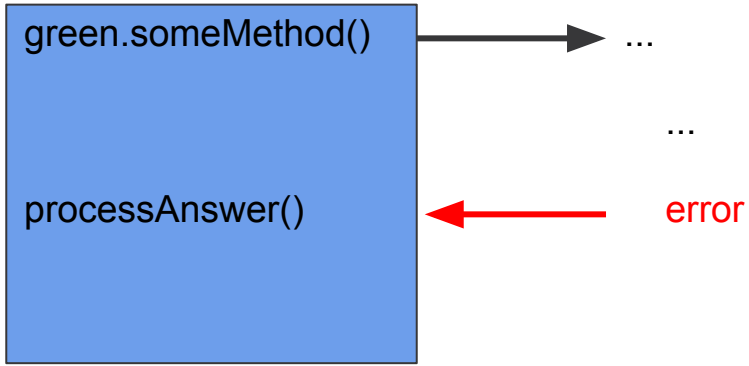


Issue #4: Stability. Fault tolerance

Additional information

Real-time systems have to be fault tolerant and are able to continue work after failure

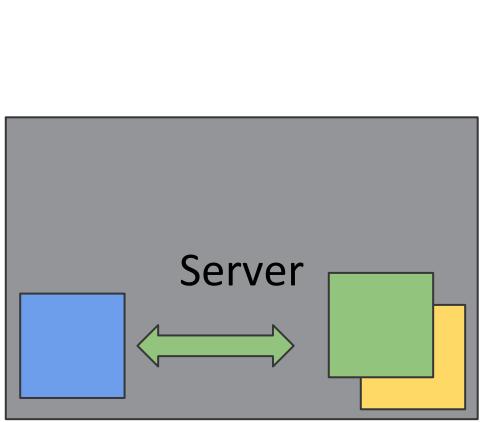


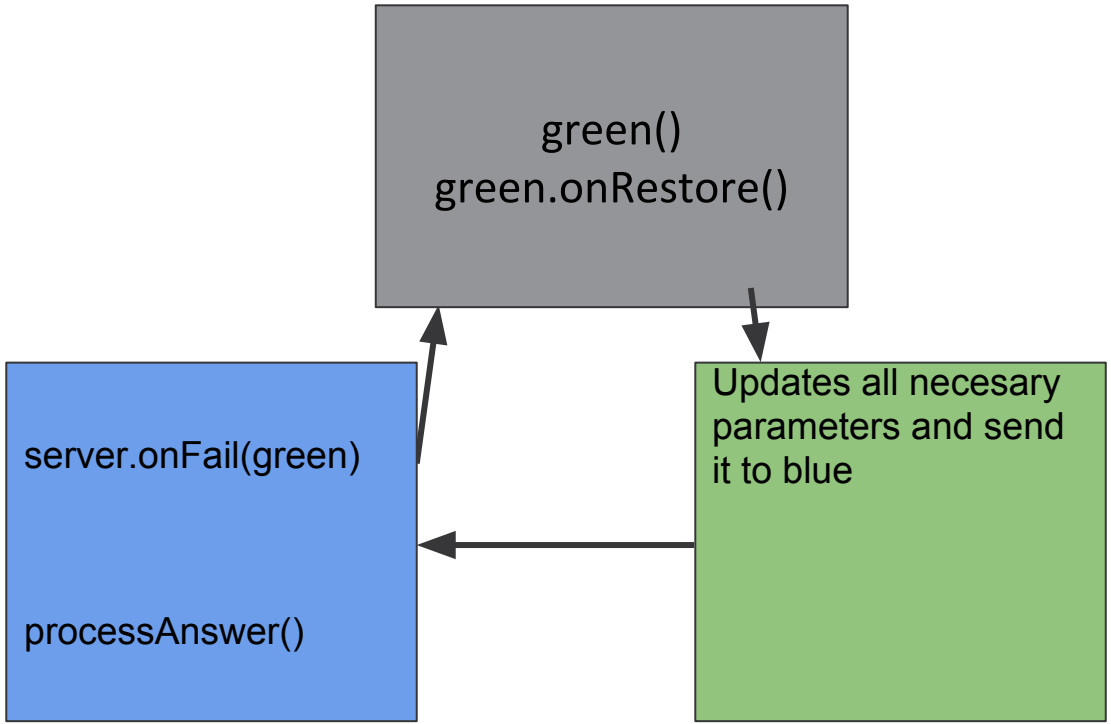


Issue #4: Stability. Fault tolerance

Additional information

Real-time systems have to be fault tolerant and are able to continue work after failure





Conclusions

- Python have some issues to be used for high load and real time apps
  - GIL capture control
  - non-Python calls performance control
  - GC
  - variable types

Conclusions

- Python have some issues to be used for high load and real time apps
  - GIL capture control
  - non-Python calls performance control
  - GC
  - variable types
- These issues can be solved with some modifications
  - tune it
  - take care of it :)
  - switch off, take care of references
  - use variables AS they are typed
-

Conclusions

- Python have some issues to be used for high load and real time apps
  - GIL capture control
  - non-Python calls performance control
  - GC
  - variable types
- These issues can be solved with some modifications
  - tune it
  - take care of it :)
  - switch off, take care of references
  - use variables AS they are typed
- It worth efforts

Thank you

Dmitry Karpov
dakarpov@gmail.com
wargaming.net