Network Automation Showdown: Go vs. Python

NANOG 90 – February 13th, 2024

Moderator: Cat Gurinsky

Panelists:

Brandon Bennett, Roblox
Ryan Hamel, i3D.net
Daniel Hertzberg, Arista Networks
Frank Seesink, UNC Chapel Hill
Claus Töpke, Telcomanager Technologies

Go vs. Python

The goals of this panel

NANOG™

What we'll cover

For each language we'll discuss:

- Pros & Cons
- What the language excels at
- What the language struggles with
- What modules / libraries exist for network purposes
- Who should consider using it and why

NANOG™

Go vs Python quick comparison

Python:

- Ecosystem: lots of special libraries
- Learning Curve: more intuitive for beginners
- Dynamically Typing: streamlines the coding process

Go:

- Compiled Nature: Simplifies deployments
- Concurrency: great performance at scale
- Statically Typed: more predictable with upfront declarations
- Error Handling: proactive approach for better resilience

NANOG™

Taken from: https://www.packetcoders.io/python-vs-go-for-network-automation/

Static vs. Typed
Interpreted vs. Compiled

- **Dynamic typing:** Used by Python, type checking happens at runtime. Types don’t have to be specified.
- **Static typing:** Used by Go, type checking happens when compiling. Types should be specified.
- **Interpreted Language:** Python, the source code of a program is converted into bytecode that is then executed by the interpreter.
- **Compiled Language:** Go, converted directly into machine code that the processor can execute, stand alone and the resulting binary doesn’t require installing dependencies.

NANOG™

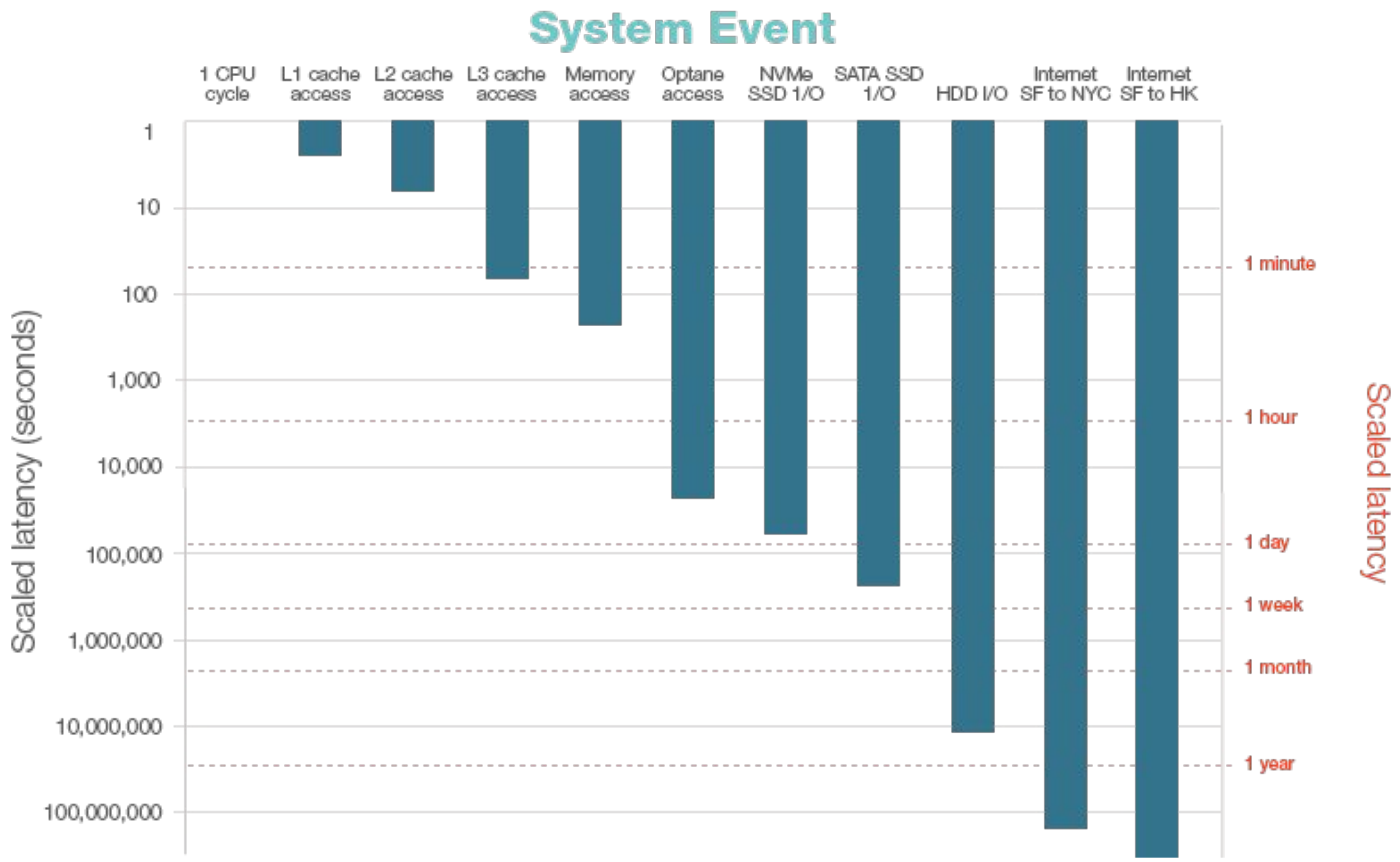
Concurrency & Parallelism

- CPython GIL (Global Interpreter Lock)
    - Limited to a single core (work being done in PEP703)
    - Threads
    - Multiprocessing
    - concurrent.futures
    - asyncio
        - Coroutines
- Goroutines are not the same as coroutines
    - Green thread based scheduler
    - Can be spread across cores

https://docs.oracle.com/cd/E36784_01/html/E36868/mtintro-6.html

Performance

- At what scale does performance matter?



NANOG™

Easy vs Simple

- Python is easy. Go is simple. Simple is not easy.
- Python → Go cheat sheets

```python
temperatures = [
    {"city": "City1", "temp": 19},
    {"city": "City2", "temp": 22},
    {"city": "City3", "temp": 21},
]

filtered_temps = {
    entry["city"]: entry["temp"] for entry in temperatures if entry["temp"] > 20
}
```

```go
type CityTemperature struct {
    City      string
    Temp float64
}

// ...

temperatures := []CityTemperature{
    {"City1", 19},
    {"City2", 22},
    {"City3", 21},
}

filteredTemps := make(map[string]float64)
for _, ct := range temperatures {
    if ct.Temp > 20 {
        filteredTemps[ct.City] = ct.Temp
    }
}
```

https://preslav.me/2023/11/27/python-is-easy-golang-is-simple-simple-is-not-easy/

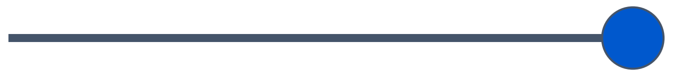
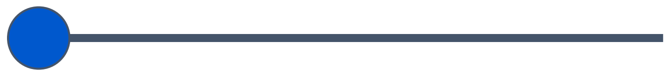
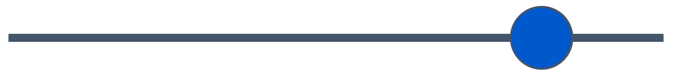
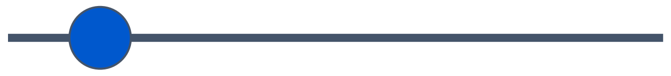
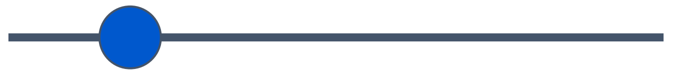
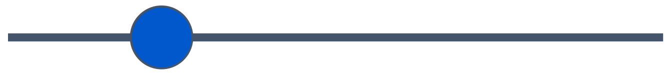
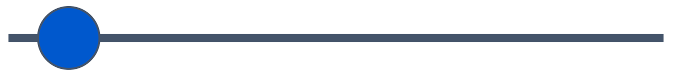
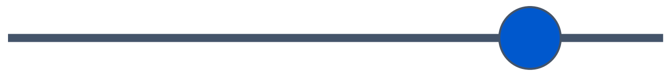
Deployments & Dependencies

- Python
  - Plenty of tool chain based helpers
  - Jupyter notebooks
  - REPL: Read-Eval-Print-Loop
  - Requirements
  - Virtual Environments
- Go
  - No external dependencies
  - After compilation it’s a single binary
  - Can cross compile for other OS
  - Built in unit testing
  - Formatting
  - Typing

Network libraries Go vs Python

- Python
  - Paramiko (SSH) / Netmiko (SSH network devices)
  - Nornir (automation framework)
  - NAPALM (Network Automation and Programmability Abstraction Layer with Multivendor support)
  - pyGNMI
  - netaddr
- Go
  - Openconfig Go Modules(yGOT,yGNMI,gRIBI,gNMI,goYANG)
  - goBGP
  - Netaddr
  - Prometheus

Dev Time vs. Execution Time

	Development Time	Execution Time
Assembler		
C		
Go		
Python		

Thank you

13-FEB-2024

NANOG™

Resources

- Python -> Go Cheat Sheet Examples:
  - https://www.353.solutions/py2go/index.html
- Getting started with Go tutorial
  - https://go.dev/doc/tutorial/getting-started
- Getting started with Python
  - https://www.python.org/about/gettingstarted/