Xen Summit
Nanjing 2018

Qubes in Action

李枫
hkli2013@126.com
Jun 20，2018

Agenda

I. Overall Design

- Overview
- 4.0
- Security

II. Acceleration for Python-based ToolStack

- Overview
- SaltStack
- Python Runtimes
- GraalVM
- GraalPython

III. Future Evolution and Re-design

- Official
- Re-designing, Re-engineering, Re-inventing

I. Overall Design

1) Overview

- https://www.qubes-os.org/

"If you're serious about security, Qubes OS is the best OS available today. It's what I use, and free."

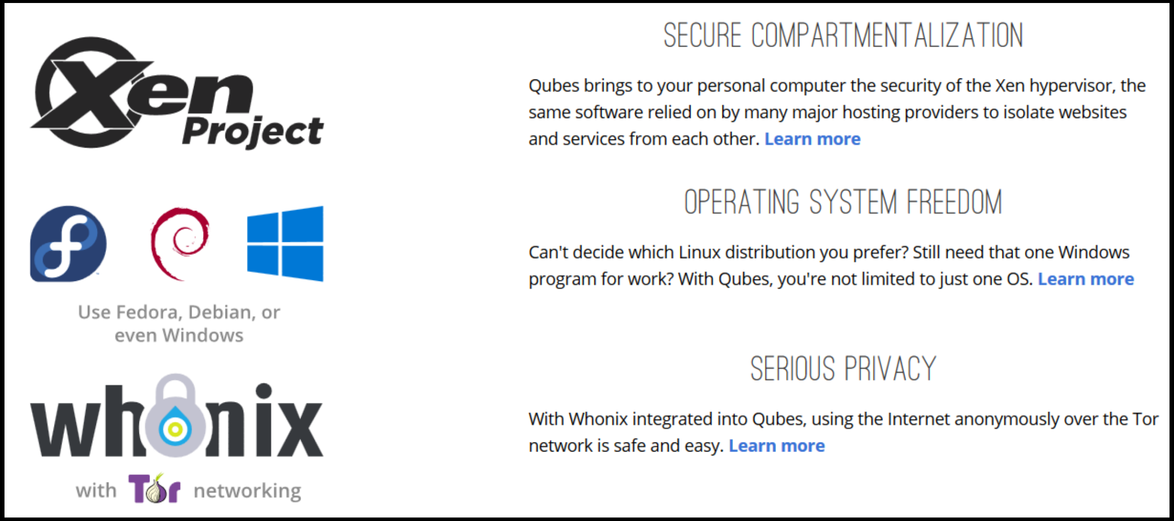
— Edward Snowden, *whistleblower and privacy advocate*

Joanna Rutkowska

rootkovska

Qubes OS and Invisible Things Lab

What's Inside of Qubes



Use Fedora, Debian, or even Windows

SECURE COMPARTMENTALIZATION

Qubes brings to your personal computer the security of the Xen hypervisor, the same software relied on by many major hosting providers to isolate websites and services from each other. Learn more

OPERATING SYSTEM FREEDOM

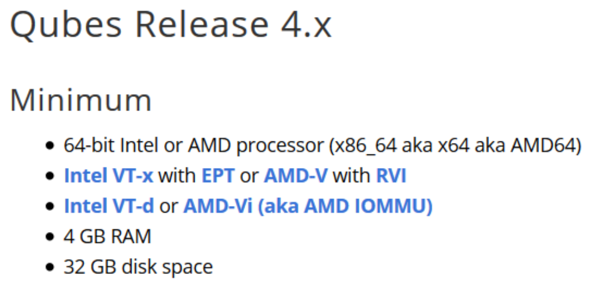
Can't decide which Linux distribution you prefer? Still need that one Windows program for work? With Qubes, you're not limited to just one OS. Learn more

SERIOUS PRIVACY

With Whonix integrated into Qubes, using the Internet anonymously over the Tor network is safe and easy. Learn more

HCL

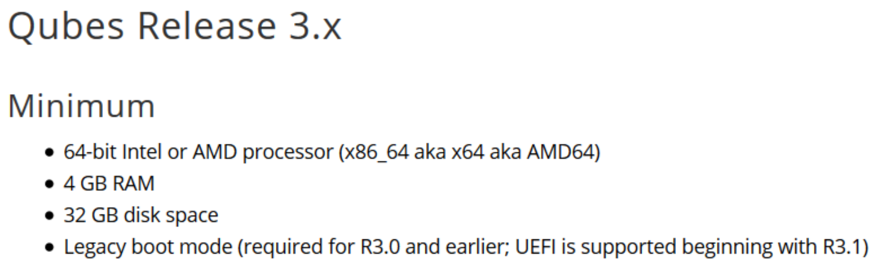
- https://www.qubes-os.org/hcl/ https://www.qubes-os.org/doc/system-requirements/



Qubes Release 4.x

Minimum

- 64-bit Intel or AMD processor (x86_64 aka x64 aka AMD64)
- Intel VT-x with EPT or AMD-V with RVI
- Intel VT-d or AMD-Vi (aka AMD IOMMU)
- 4 GB RAM
- 32 GB disk space



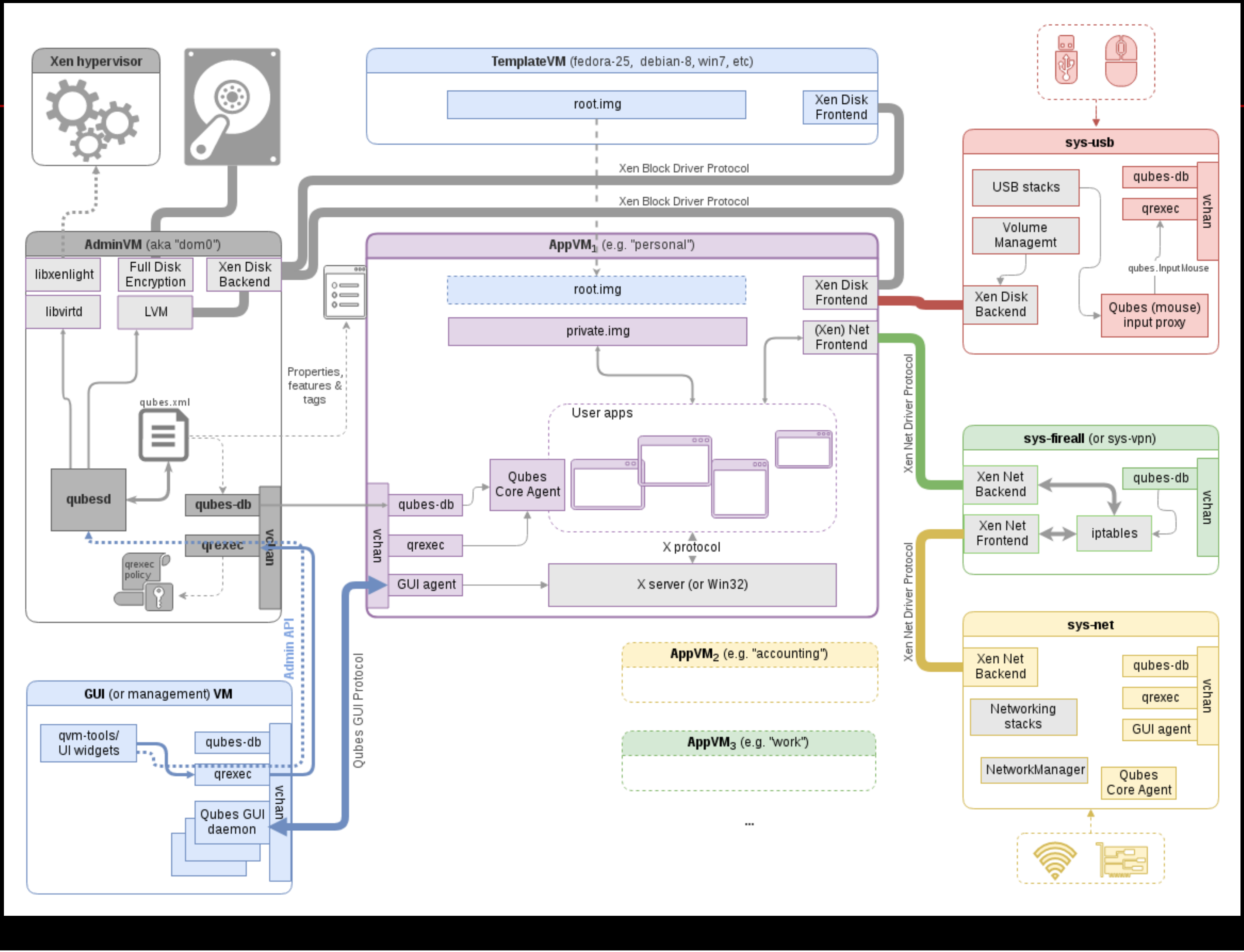
Qubes Release 3.x

Minimum

- 64-bit Intel or AMD processor (x86_64 aka x64 aka AMD64)
- 4 GB RAM
- 32 GB disk space
- Legacy boot mode (required for R3.0 and earlier; UEFI is supported beginning with R3.1)

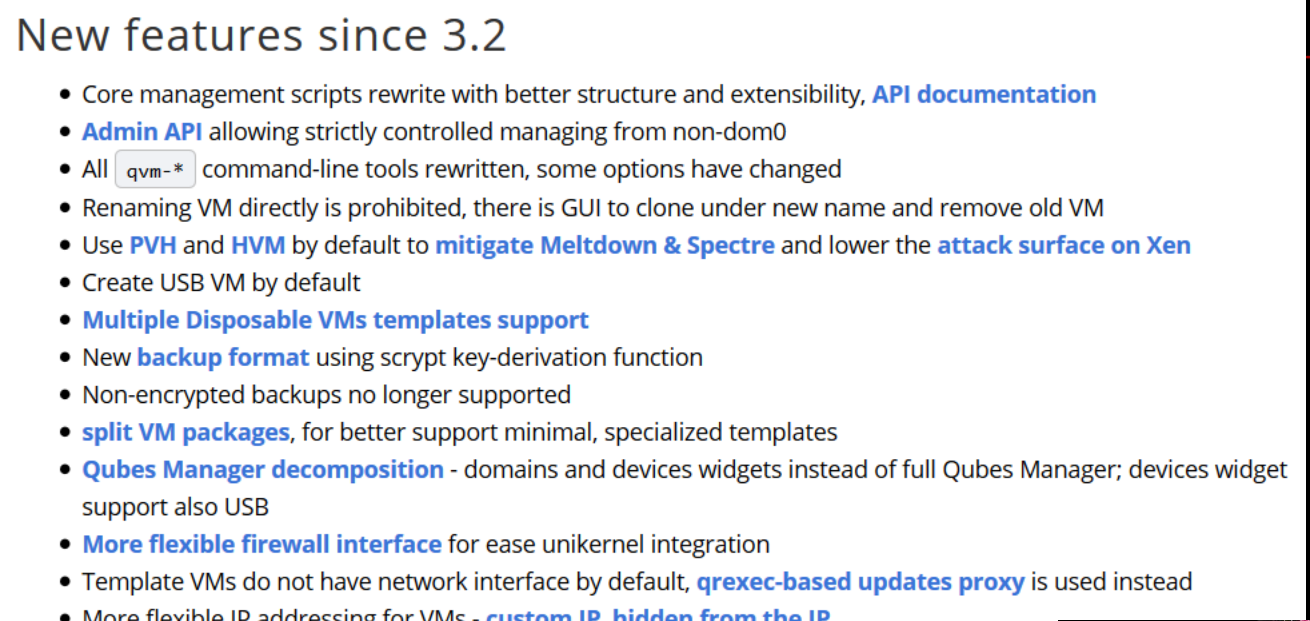
Core Stack

- https://www.qubes-os.org/news/2017/10/03/core3/



2) 4.0

- https://www.qubes-os.org/news/2018/03/28/qubes-40/
- https://www.qubes-os.org/doc/releases/4.0/release-notes/



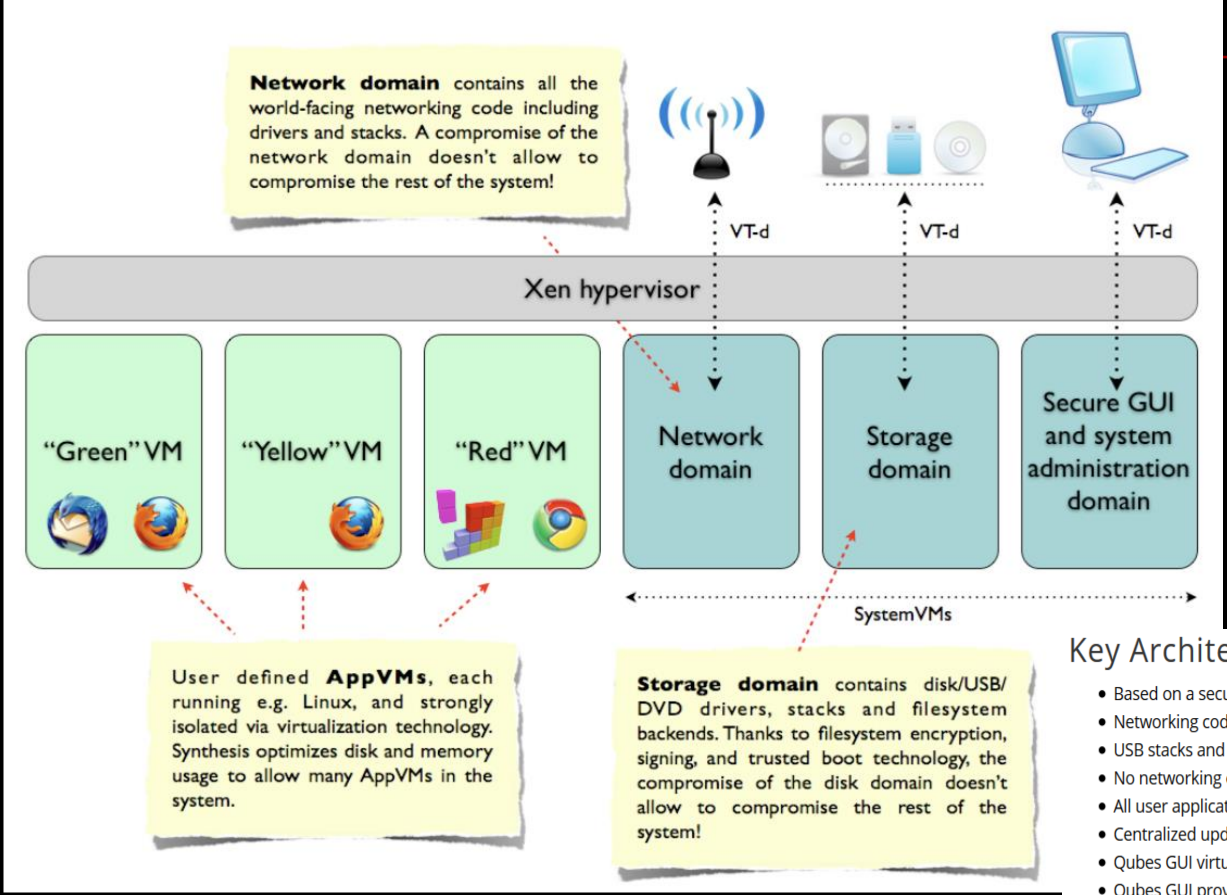
New features since 3.2

- Core management scripts rewrite with better structure and extensibility, **API documentation**
- **Admin API** allowing strictly controlled managing from non-dom0
- All `qvm-*` command-line tools rewritten, some options have changed
- Renaming VM directly is prohibited, there is GUI to clone under new name and remove old VM
- Use **PVH** and **HVM** by default to **mitigate Meltdown & Spectre** and lower the **attack surface on Xen**
- Create USB VM by default
- **Multiple Disposable VMs templates support**
- New **backup format** using scrypt key-derivation function
- Non-encrypted backups no longer supported
- **split VM packages**, for better support minimal, specialized templates
- **Qubes Manager decomposition** - domains and devices widgets instead of full Qubes Manager; devices widget support also USB
- **More flexible firewall interface** for ease unikernel integration
- Template VMs do not have network interface by default, **qrexec-based updates proxy** is used instead
- More flexible IP addressing for VMs - **custom IP**, **hidden from the IP**
- More flexible Qubes RPC policy - **related ticket**, **documentation**
- **New Qubes RPC confirmation window**, including option to specify destination VM
- **New storage subsystem design**
- Dom0 update to Fedora 25 for better hardware support
- Kernel 4.9.x

3) Security

- https://www.qubes-os.org/security

Virtualization



Network domain contains all the world-facing networking code including drivers and stacks. A compromise of the network domain doesn't allow to compromise the rest of the system!

User defined **AppVMs**, each running e.g. Linux, and strongly isolated via virtualization technology. Synthesis optimizes disk and memory usage to allow many AppVMs in the system.

Storage domain contains disk/USB/DVD drivers, stacks and filesystem backends. Thanks to filesystem encryption, signing, and trusted boot technology, the compromise of the disk domain doesn't allow to compromise the rest of the system!

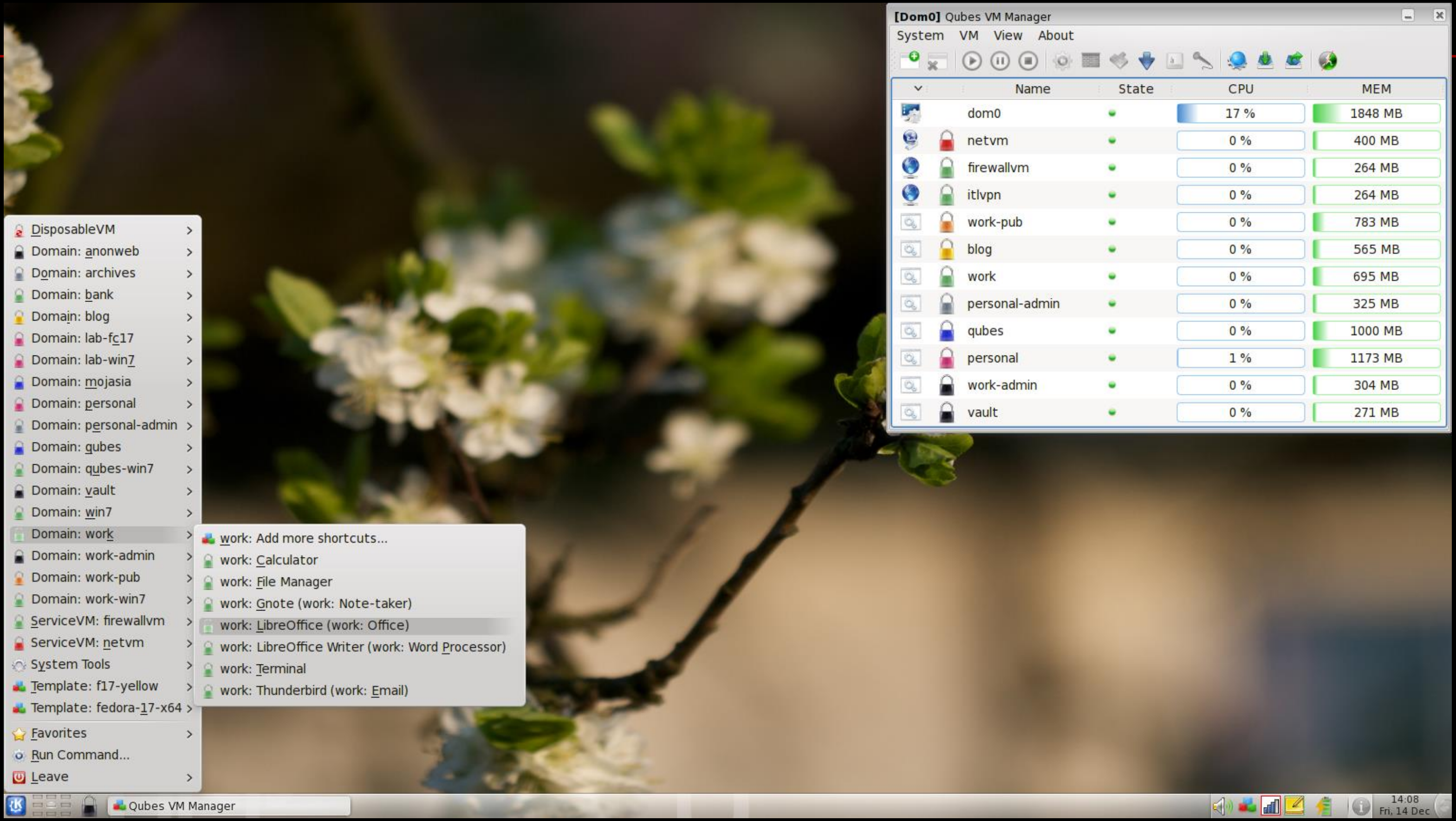
Key Architecture features

- Based on a secure bare-metal hypervisor (Xen)
- Networking code sand-boxed in an unprivileged VM (using IOMMU/VT-d)
- USB stacks and drivers sand-boxed in an unprivileged VM (currently experimental feature)
- No networking code in the privileged domain (dom0)
- All user applications run in "AppVMs," lightweight VMs based on Linux
- Centralized updates of all AppVMs based on the same template
- Qubes GUI virtualization presents applications as if they were running locally
- Qubes GUI provides isolation between apps sharing the same desktop
- Secure system boot based (optional)

Source: https://www.qubes-os.org/doc/architecture/

Compartmentalization

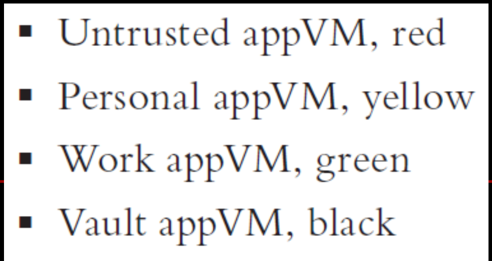
- defects of monolithic systems
- separate your digital life into security domains



Name	State	CPU	MEM
dom0	●	17 %	1848 MB
netvm	●	0 %	400 MB
firewallvm	●	0 %	264 MB
itlvpn	●	0 %	264 MB
work-pub	●	0 %	783 MB
blog	●	0 %	565 MB
work	●	0 %	695 MB
personal-admin	●	0 %	325 MB
qubes	●	0 %	1000 MB
personal	●	1 %	1173 MB
work-admin	●	0 %	304 MB
vault	●	0 %	271 MB

Source: https://www.qubes-os.org/doc/architecture/

- **Trust Level**



- Untrusted appVM, red
- Personal appVM, yellow
- Work appVM, green
- Vault appVM, black

…

- **VM Types**

AppVM

ServiceVM netVM proxyVM

TemplateVM

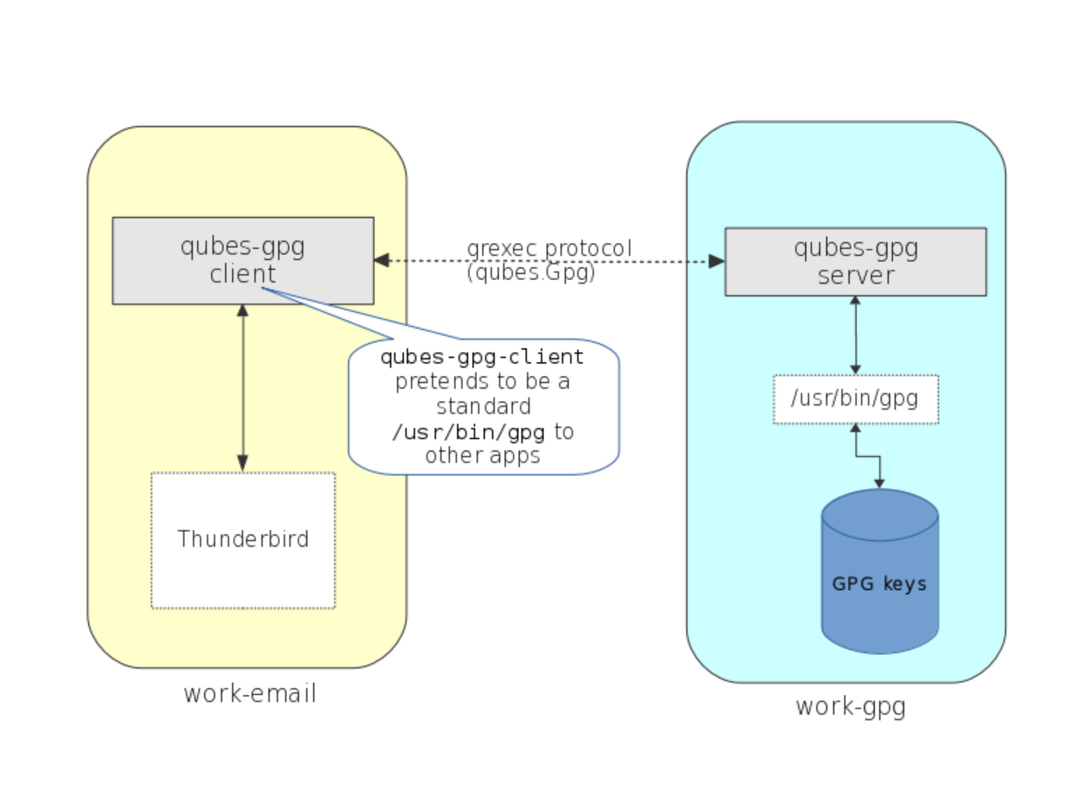
DispVM

…

- **VM Manager**

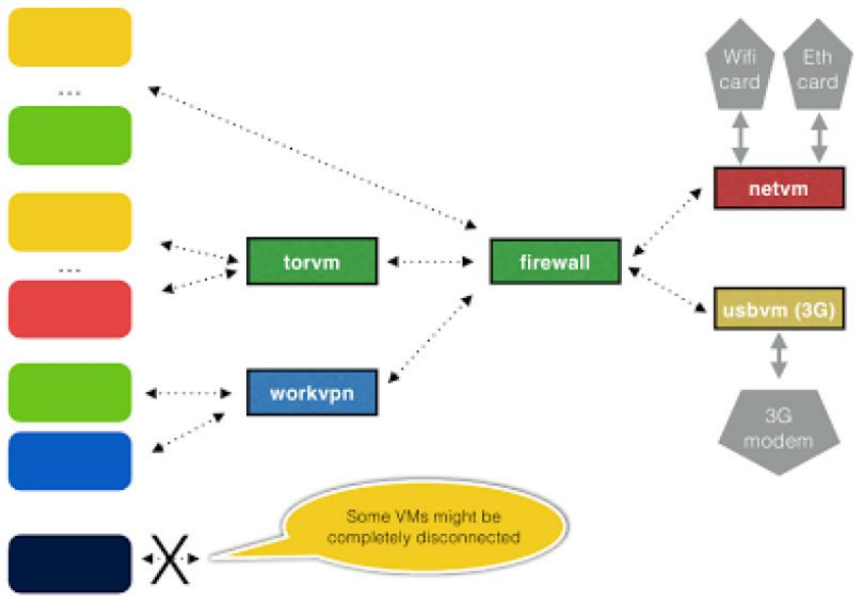
Split GPG

- https://www.qubes-os.org/doc/split-gpg/
- Poor Man's Hardware Security Module (pmHSM)


qubes-gpg client
qrexec protocol (qubes.Gpg)
qubes-gpg server
qubes-gpg-client pretends to be a standard /usr/bin/gpg to other apps
/usr/bin/gpg
Thunderbird
GPG keys
work-email
work-gpg

netVM

- https://www.qubes-os.org/doc/networking/
- https://www.qubes-os.org/doc/firewall/



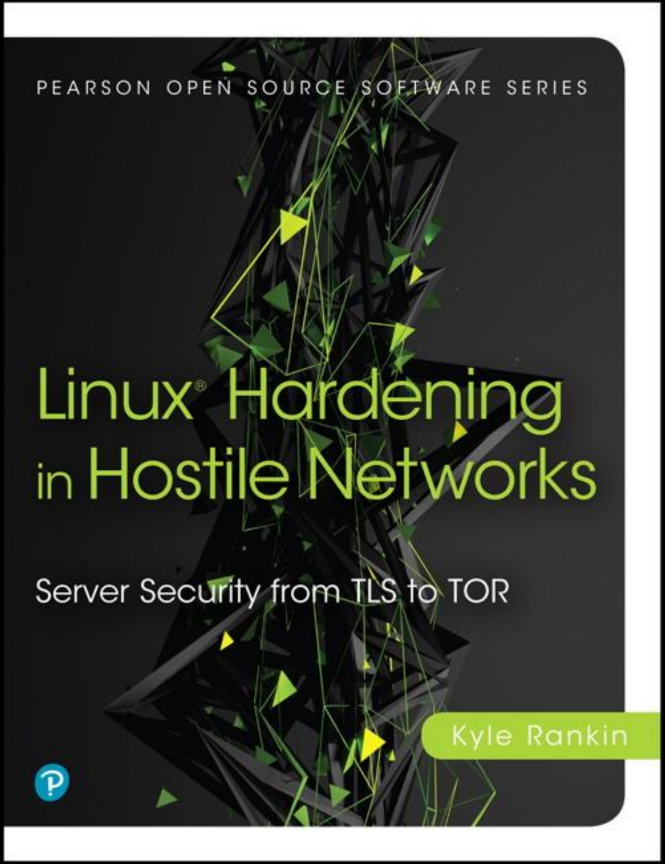
Source: https://www.qubes-os.org/attachment/wiki/slides/RMLL_2016_Improving-client-systems-security.pdf

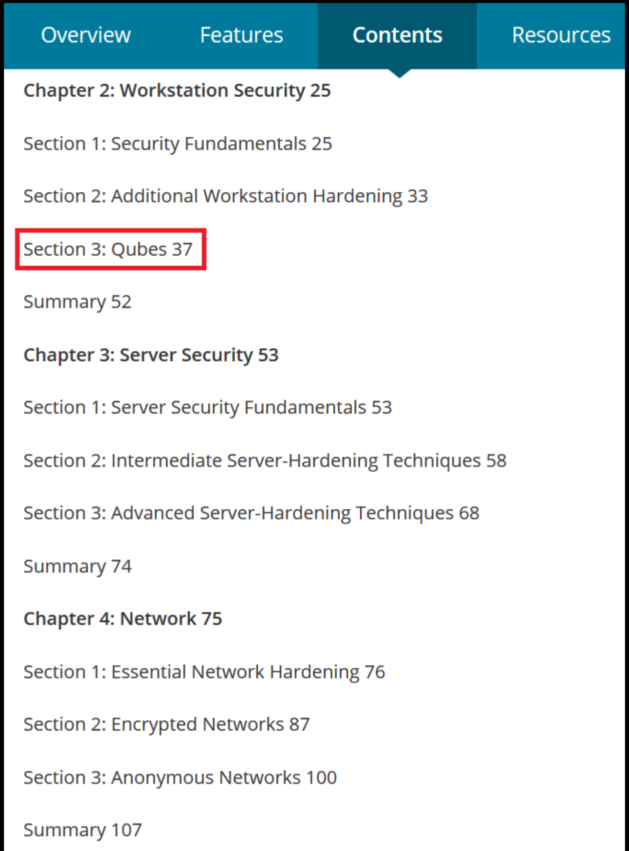
usbVM

- https://www.qubes-os.org/doc/usb/#security-warning-about-usb-input-devices
- sys-usb
- qubes-usb-proxy

Good Resource

- https://www.qubes-os.org/doc/
- https://www.pearson.com/us/higher-education/program/Rankin-Linux-Hardening-in-Hostile-Networks-Server-Security-from-TLS-to-Tor/PGM137619.html


PEARSON OPEN SOURCE SOFTWARE SERIES
Linux® Hardening in Hostile Networks
Server Security from TLS to TOR
Kyle Rankin



Chapter 2: Workstation Security 25

Section 1: Security Fundamentals 25

Section 2: Additional Workstation Hardening 33

Section 3: Qubes 37

Summary 52

Chapter 3: Server Security 53

Section 1: Server Security Fundamentals 53

Section 2: Intermediate Server-Hardening Techniques 58

Section 3: Advanced Server-Hardening Techniques 68

Summary 74

Chapter 4: Network 75

Section 1: Essential Network Hardening 76

Section 2: Encrypted Networks 87

Section 3: Anonymous Networks 100

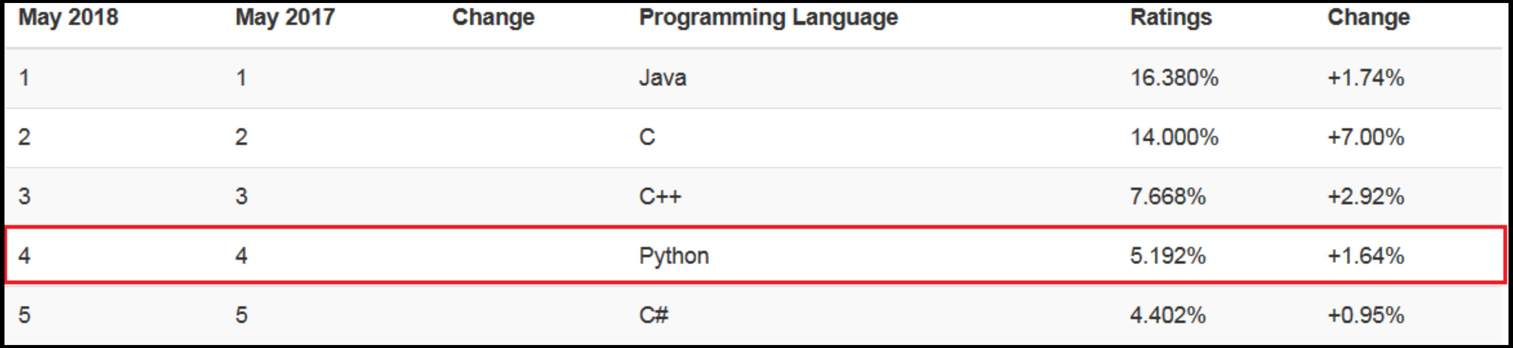
Summary 107

II. Acceleration for Python-based ToolStack

1) Overview

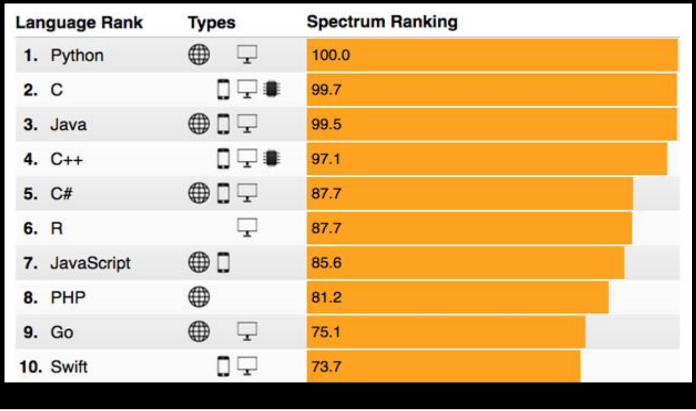
Why Python

- https://www.tiobe.com/tiobe-index/



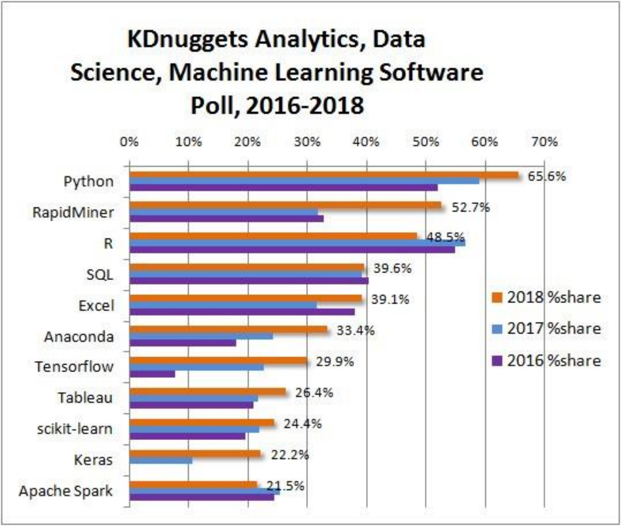
May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%

- http://pypl.github.io/PYPL.html
- https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ▦	99.7
3. Java	🌐 📱 🖥	99.5
4. C++	📱 🖥 ▦	97.1
5. C#	🌐 📱 🖥	87.7
6. R	🖥	87.7
7. JavaScript	🌐 📱	85.6
8. PHP	🌐	81.2
9. Go	🌐 🖥	75.1
10. Swift	📱 🖥	73.7

- https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html

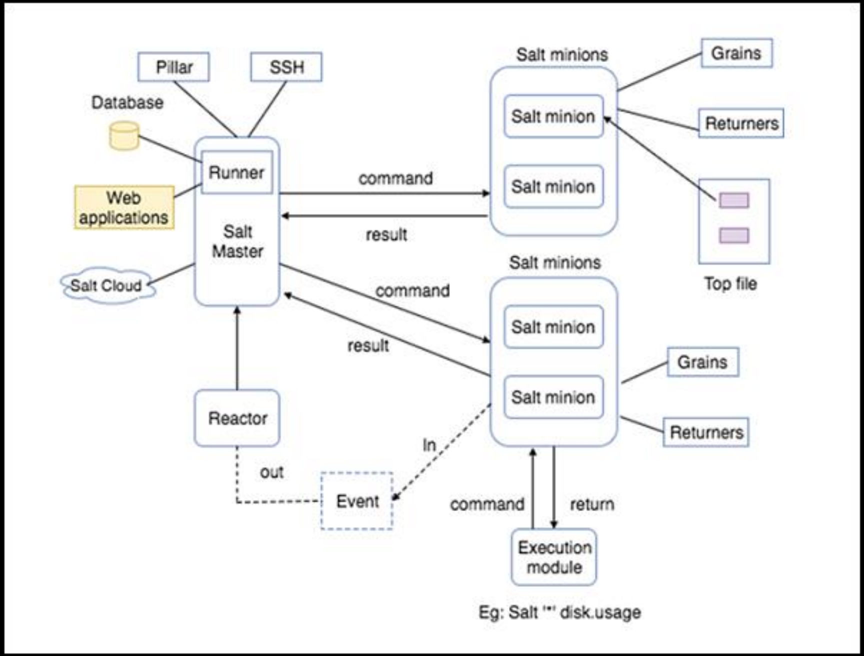


- Other Python projects

Build: Meson, SCons… DevOps: Ansible, SaltStack…
Web: Django, web2py, Flask, Tornado, TurboGears…
AI: PyTorch, Theano… Big Data: PyData, PySpark…
Science: Scipy, Sage…
Cloud/DataCenter: OpenStack
Security: a swiss knife for hackers…
…

2) SaltStack

- https://saltstack.com/

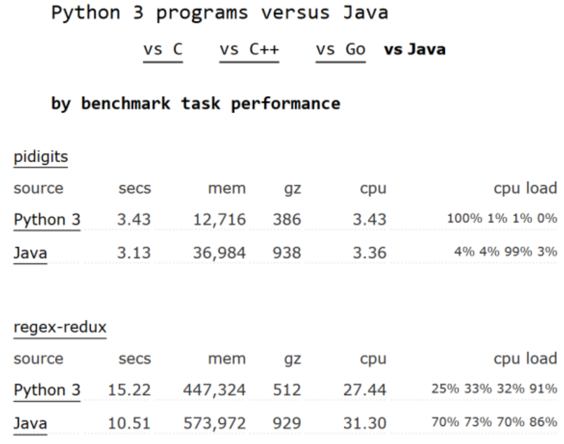
  Intelligent automation for a software-defined world
- https://www.qubes-os.org/doc/salt/
- default management engine in dom0 since Qubes 3.1
- qubesctl is inter-changeable and an alias for salt-call
- https://docs.saltstack.com/en/latest/topics/virt/index.html
- 

Source: https://www.tutorialspoint.com/saltstack/saltstack_architecture.htm

3) Python Runtimes

Why Python is Slow

- dynamically typed
- ~~no JIT support in the official CPython~~
- GIL (Global Interpreter Lock)
-



Python 3 programs versus Java

vs C vs C++ vs Go **vs Java**

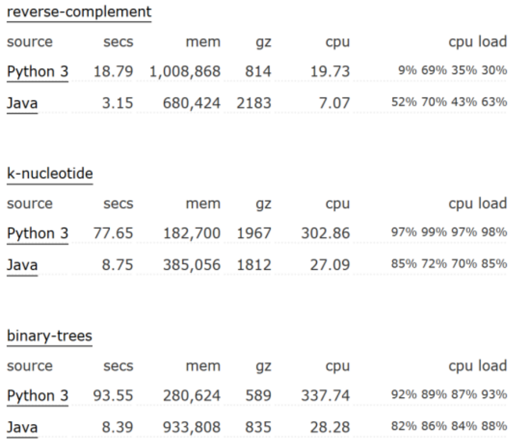
by benchmark task performance

pidigits

source	secs	mem	gz	cpu	cpu load
Python 3	3.43	12,716	386	3.43	100% 1% 1% 0%
Java	3.13	36,984	938	3.36	4% 4% 99% 3%

regex-redux

source	secs	mem	gz	cpu	cpu load
Python 3	15.22	447,324	512	27.44	25% 33% 32% 91%
Java	10.51	573,972	929	31.30	70% 73% 70% 86%



reverse-complement

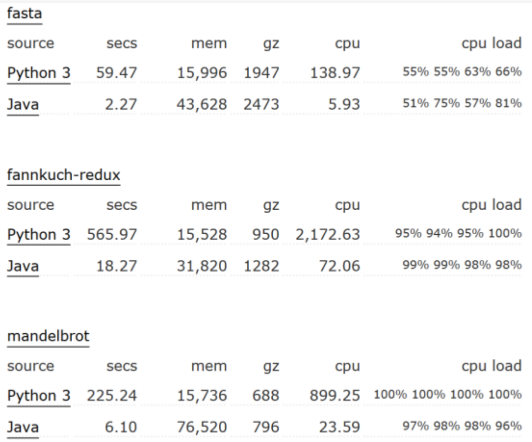
source	secs	mem	gz	cpu	cpu load
Python 3	18.79	1,008,868	814	19.73	9% 69% 35% 30%
Java	3.15	680,424	2183	7.07	52% 70% 43% 63%

k-nucleotide

source	secs	mem	gz	cpu	cpu load
Python 3	77.65	182,700	1967	302.86	97% 99% 97% 98%
Java	8.75	385,056	1812	27.09	85% 72% 70% 85%

binary-trees

source	secs	mem	gz	cpu	cpu load
Python 3	93.55	280,624	589	337.74	92% 89% 87% 93%
Java	8.39	933,808	835	28.28	82% 86% 84% 88%



fasta

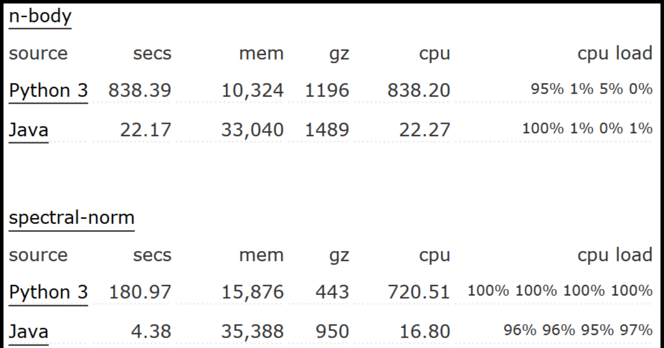
source	secs	mem	gz	cpu	cpu load
Python 3	59.47	15,996	1947	138.97	55% 55% 63% 66%
Java	2.27	43,628	2473	5.93	51% 75% 57% 81%

fannkuch-redux

source	secs	mem	gz	cpu	cpu load
Python 3	565.97	15,528	950	2,172.63	95% 94% 95% 100%
Java	18.27	31,820	1282	72.06	99% 99% 98% 98%

mandelbrot

source	secs	mem	gz	cpu	cpu load
Python 3	225.24	15,736	688	899.25	100% 100% 100% 100%
Java	6.10	76,520	796	23.59	97% 98% 98% 96%

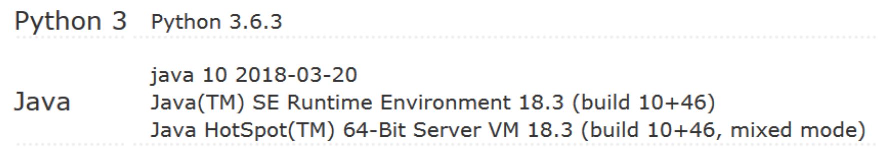


n-body

source	secs	mem	gz	cpu	cpu load
Python 3	838.39	10,324	1196	838.20	95% 1% 5% 0%
Java	22.17	33,040	1489	22.27	100% 1% 0% 1%

spectral-norm

source	secs	mem	gz	cpu	cpu load
Python 3	180.97	15,876	443	720.51	100% 100% 100% 100%
Java	4.38	35,388	950	16.80	96% 96% 95% 97%

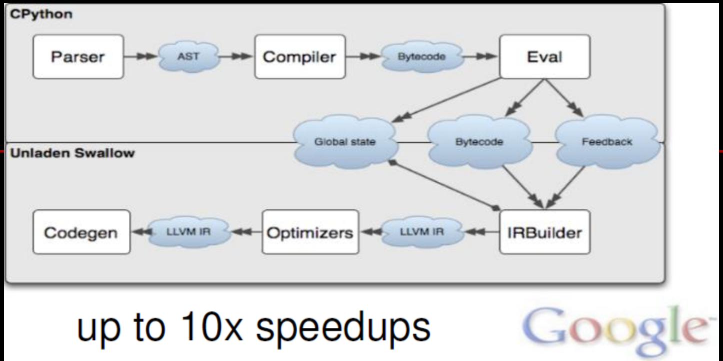


Python 3	Python 3.6.3
Java	java 10 2018-03-20 Java(TM) SE Runtime Environment 18.3 (build 10+46) Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10+46, mixed mode)

https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/python.html

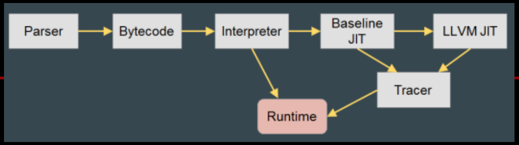
Runtimes

- LLVM-based (VMKit, MCJIT…)



up to 10x speedups

PySton



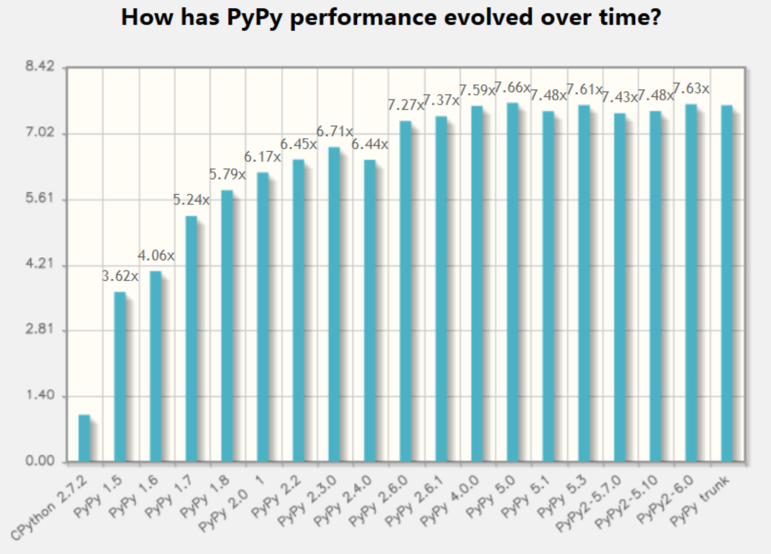
Dropbox

- pypy

RPython

Meta-tracing

…

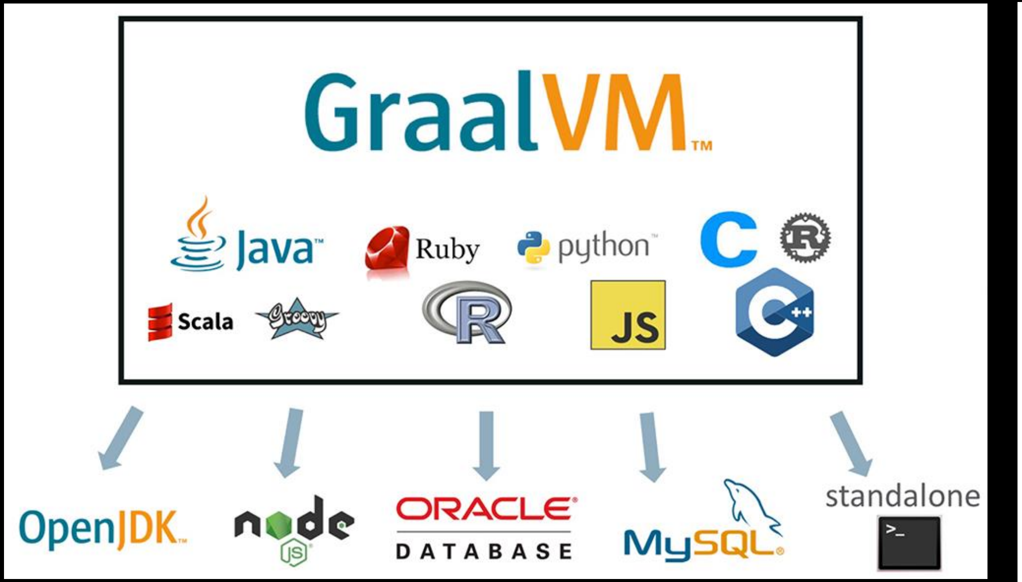


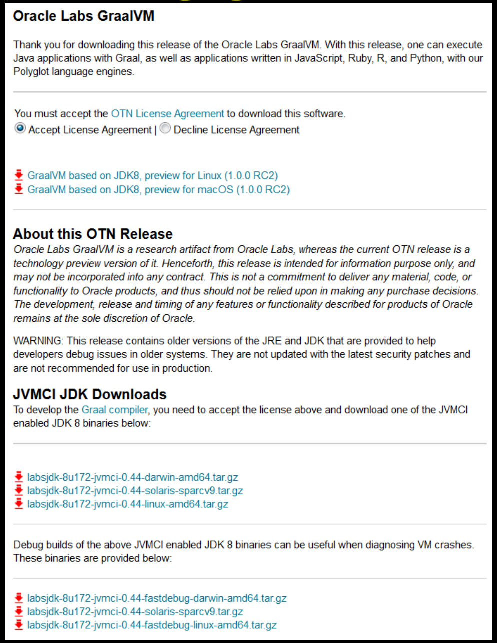
How has PyPy performance evolved over time?

Source: http://speed.pypy.org/

4) GraalVM

- https://www.graalvm.org/
- http://www.oracle.com/technetwork/oracle-labs/program-languages/overview/index.html
- https://blogs.oracle.com/developers/announcing-graalvm





Oracle Labs GraalVM

Thank you for downloading this release of the Oracle Labs GraalVM. With this release, one can execute Java applications with Graal, as well as applications written in JavaScript, Ruby, R, and Python, with our Polyglot language engines.

You must accept the OTN License Agreement to download this software.
● Accept License Agreement | ○ Decline License Agreement

- GraalVM based on JDK8, preview for Linux (1.0.0 RC2)
- GraalVM based on JDK8, preview for macOS (1.0.0 RC2)

About this OTN Release

Oracle Labs GraalVM is a research artifact from Oracle Labs, whereas the current OTN release is a technology preview version of it. Henceforth, this release is intended for information purpose only, and may not be incorporated into any contract. This is not a commitment to deliver any material, code, or functionality to Oracle products, and thus should not be relied upon in making any purchase decisions. The development, release and timing of any features or functionality described for products of Oracle remains at the sole discretion of Oracle.

WARNING: This release contains older versions of the JRE and JDK that are provided to help developers debug issues in older systems. They are not updated with the latest security patches and are not recommended for use in production.

JVMCI JDK Downloads

To develop the Graal compiler, you need to accept the license above and download one of the JVMCI enabled JDK 8 binaries below:

- labsjdk-8u172-jvmci-0.44-darwin-amd64.tar.gz
- labsjdk-8u172-jvmci-0.44-solaris-sparcv9.tar.gz
- labsjdk-8u172-jvmci-0.44-linux-amd64.tar.gz

Debug builds of the above JVMCI enabled JDK 8 binaries can be useful when diagnosing VM crashes. These binaries are provided below:

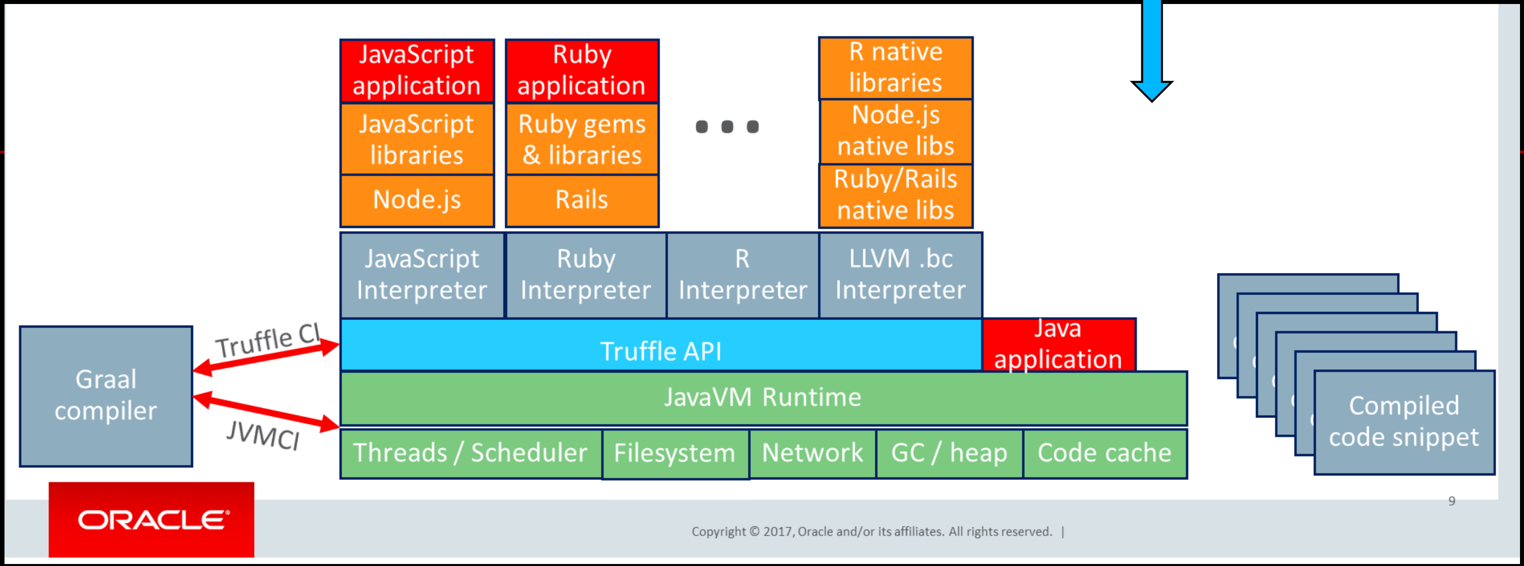
- labsjdk-8u172-jvmci-0.44-fastdebug-darwin-amd64.tar.gz
- labsjdk-8u172-jvmci-0.44-solaris-sparcv9.tar.gz
- labsjdk-8u172-jvmci-0.44-fastdebug-linux-amd64.tar.gz

- High-Performance Polyglot VM
- A meta-runtime for Language-Level Virtualization
- Currently base an Oracle Labs JDK 8 with JVMCI support
- http://openjdk.java.net/jeps/243(JVMCI): experimental in JDK 9

Arch

- A hybrid of static & dynamic runtimes

Substrate VM



JavaScript application

JavaScript libraries

Node.js

Ruby application

Ruby gems & libraries

Rails

• • •

R native libraries

Node.js native libs

Ruby/Rails native libs

JavaScript Interpreter

Ruby Interpreter

R Interpreter

LLVM .bc Interpreter

Java application

Truffle API

JavaVM Runtime

Threads / Scheduler

Filesystem

Network

GC / heap

Code cache

Graal compiler

Truffle CI

JVMCI

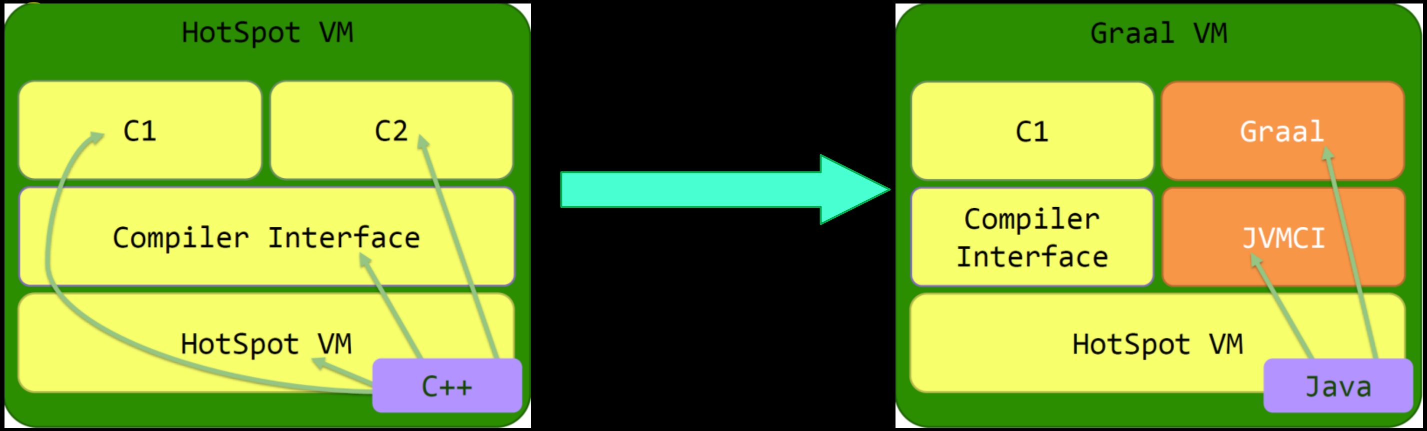
Compiled code snippet

ORACLE

9

Source: https://ics.psu.edu/wp-content/uploads/2017/02/GraalVM-PSU.pptx

-



HotSpot VM

C1

C2

Compiler Interface

HotSpot VM

C++

Graal VM

C1

Graal

Compiler Interface

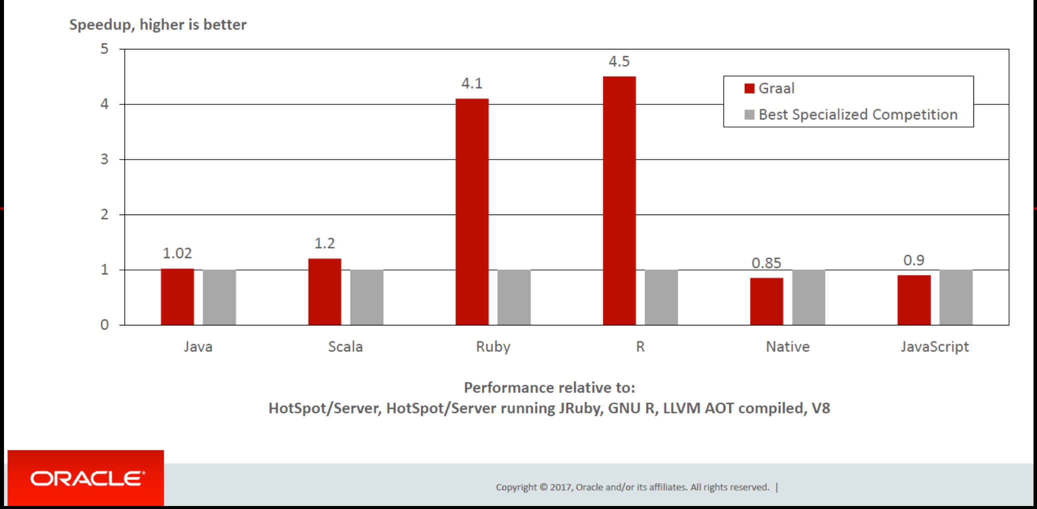
JVMCI

HotSpot VM

Java

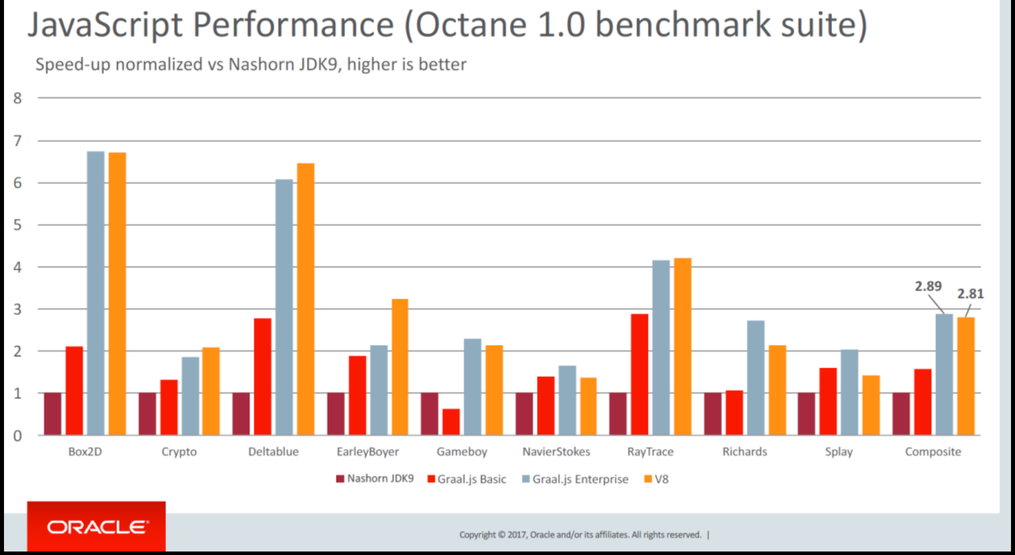
Source: https://www.slideshare.net/jyukutyo/jvmgraalopenj9

Performance



Performance relative to:
HotSpot/Server, HotSpot/Server running JRuby, GNU R, LLVM AOT compiled, V8

Source: http://lafo.ssw.uni-linz.ac.at/papers/2017_PLDI_GraalTutorial.pdf

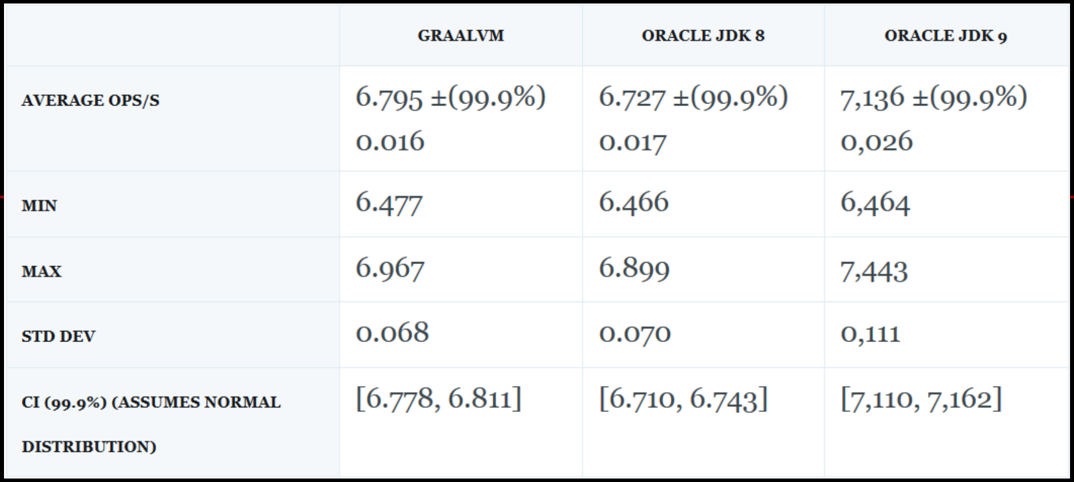


JavaScript Performance (Octane 1.0 benchmark suite)

Speed-up normalized vs Nashorn JDK9, higher is better

Source: http://dbpl2017.org/slides/DBPL-2017-s2.pdf

- but for GraalVM 1.0.0 RC1

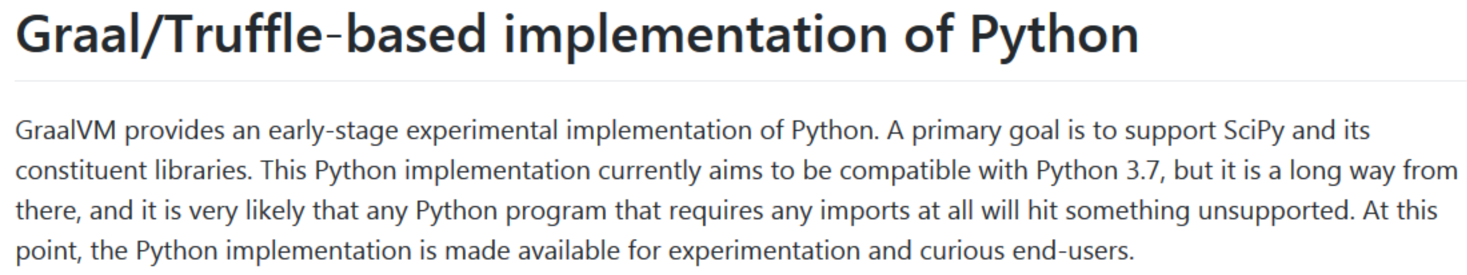


	GRAALVM	ORACLE JDK 8	ORACLE JDK 9
AVERAGE OPS/S	6.795 ±(99.9%) 0.016	6.727 ±(99.9%) 0.017	7,136 ±(99.9%) 0,026
MIN	6.477	6.466	6,464
MAX	6.967	6.899	7,443
STD DEV	0.068	0.070	0,111
CI (99.9%) (ASSUMES NORMAL DISTRIBUTION)	[6.778, 6.811]	[6.710, 6.743]	[7,110, 7,162]

Source: https://blog.frankel.ch/first-impressions-graalvm

- still have plenty of room for improvement!

5) GraalPython



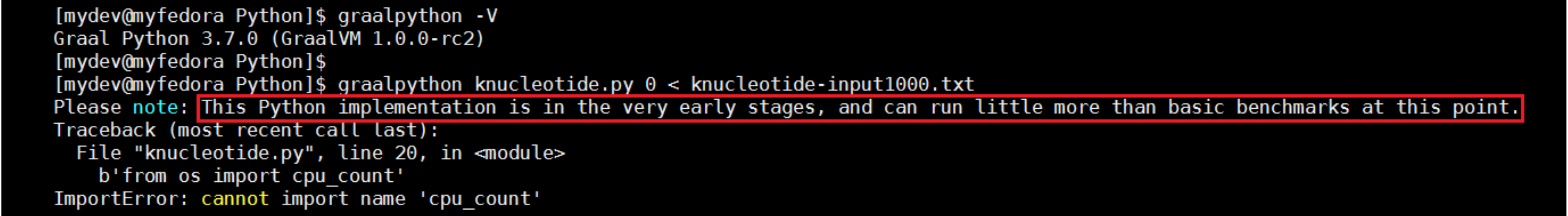
Graal/Truffle-based implementation of Python

GraalVM provides an early-stage experimental implementation of Python. A primary goal is to support SciPy and its constituent libraries. This Python implementation currently aims to be compatible with Python 3.7, but it is a long way from there, and it is very likely that any Python program that requires any imports at all will hit something unsupported. At this point, the Python implementation is made available for experimentation and curious end-users.

- https://github.com/graalvm/graalpython
- https://www.graalvm.org/docs/reference-manual/languages/python/

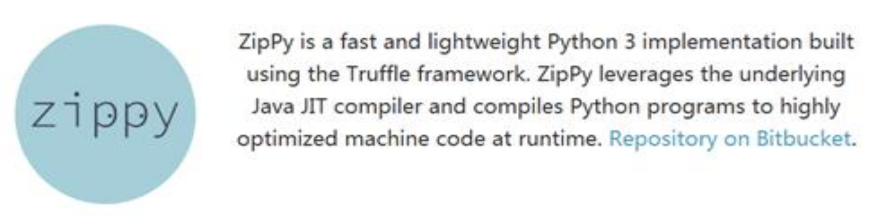
	Java 10.0.1	CPython 3.6.5	GraalPython ee-1.0.0-rc2
n-body	9.676s	11m56.642s	15m57.543s

Test on Dell XPS 15z: i5-2410M@2.3Ghz, 6G RAM, Fedora 28 for X64 with Kernel 4.16.14



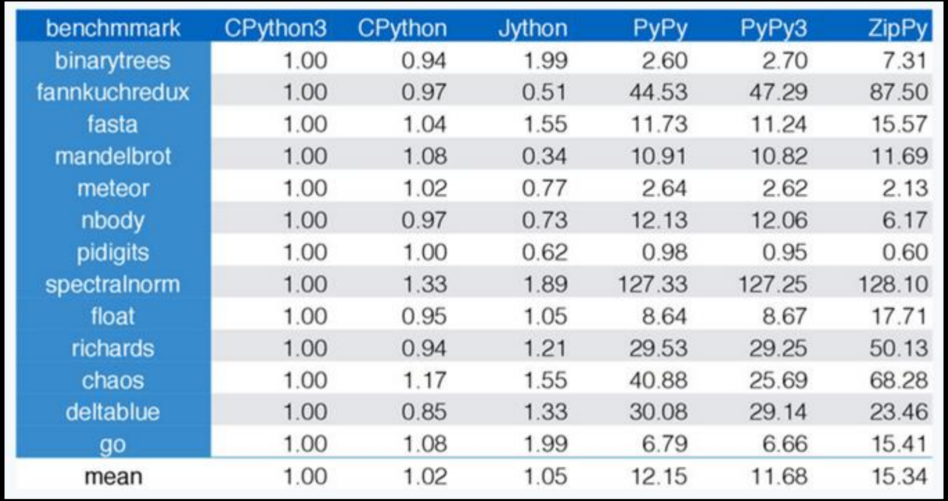
```
[mydev@myfedora Python]$ graalpython -V
Graal Python 3.7.0 (GraalVM 1.0.0-rc2)
[mydev@myfedora Python]$
[mydev@myfedora Python]$ graalpython knucleotide.py 0 < knucleotide-input1000.txt
Please note: This Python implementation is in the very early stages, and can run little more than basic benchmarks at this point.
Traceback (most recent call last):
  File "knucleotide.py", line 20, in <module>
    b'from os import cpu_count'
ImportError: cannot import name 'cpu_count'
```

ZipPy

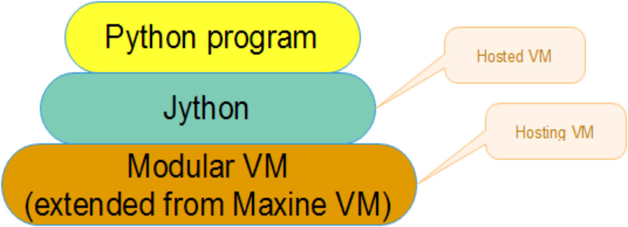


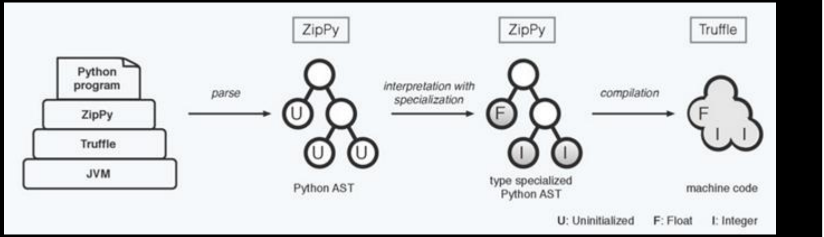
ZipPy is a fast and lightweight Python 3 implementation built using the Truffle framework. ZipPy leverages the underlying Java JIT compiler and compiles Python programs to highly optimized machine code at runtime. Repository on Bitbucket.

- http://thezhangwei.com/
- https://github.com/securesystemslab/zippy
- Optimizations
  - – Numeric Types, Type Specializations, Efficient Data Representation
  - – Control Flow Specializations, Generator Peeling, Optimizing Object Model and Calls



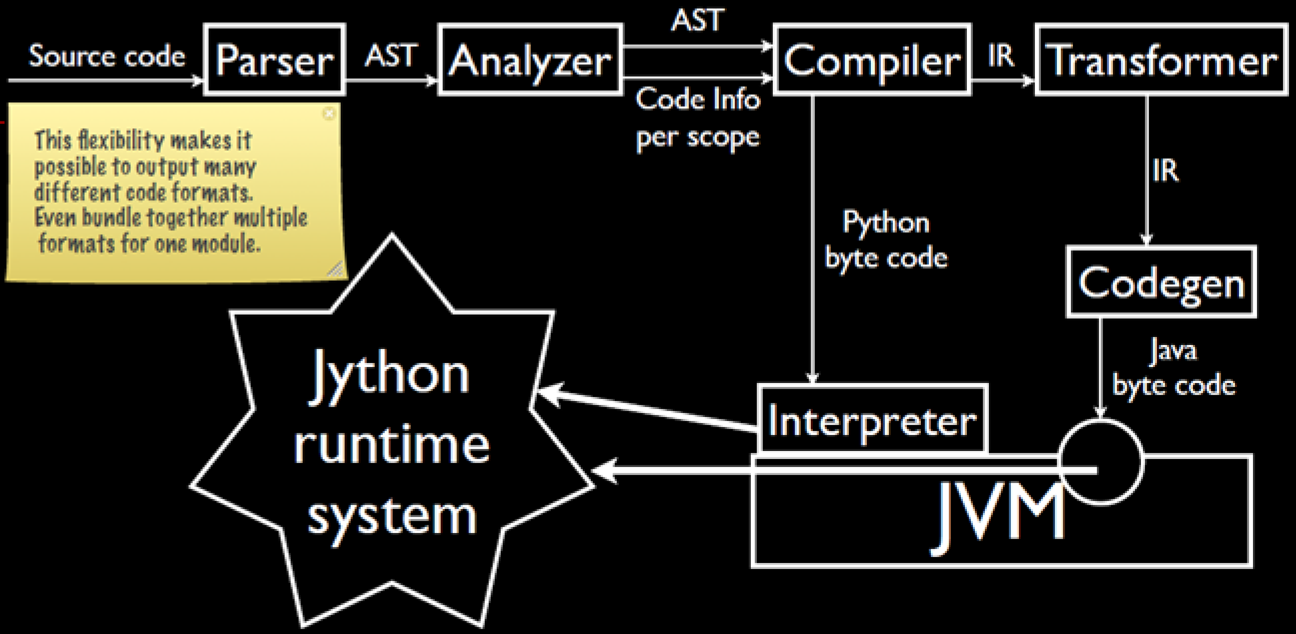
benchmmark	CPython3	CPython	Jython	PyPy	PyPy3	ZipPy
binarytrees	1.00	0.94	1.99	2.60	2.70	7.31
fannkuchredux	1.00	0.97	0.51	44.53	47.29	87.50
fasta	1.00	1.04	1.55	11.73	11.24	15.57
mandelbrot	1.00	1.08	0.34	10.91	10.82	11.69
meteor	1.00	1.02	0.77	2.64	2.62	2.13
nbody	1.00	0.97	0.73	12.13	12.06	6.17
pidigits	1.00	1.00	0.62	0.98	0.95	0.60
spectralnorm	1.00	1.33	1.89	127.33	127.25	128.10
float	1.00	0.95	1.05	8.64	8.67	17.71
richards	1.00	0.94	1.21	29.53	29.25	50.13
chaos	1.00	1.17	1.55	40.88	25.69	68.28
deltablue	1.00	0.85	1.33	30.08	29.14	23.46
go	1.00	1.08	1.99	6.79	6.66	15.41
mean	1.00	1.02	1.05	12.15	11.68	15.34





Jython

- http://www.jython.org //No new release since 2015…



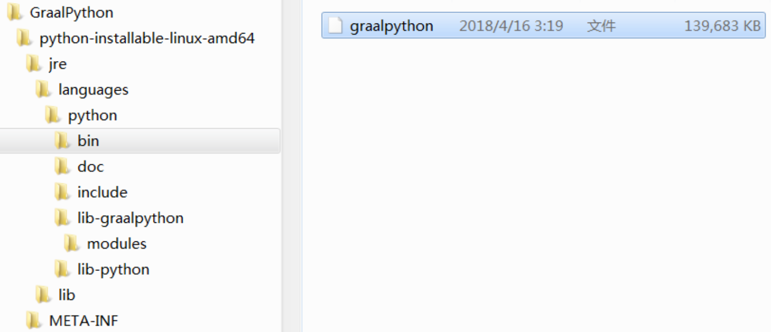
This flexibility makes it possible to output many different code formats. Even bundle together multiple formats for one module.

VOC

- https://github.com/pybee/voc/
- A transpiler that converts Python code into Java bytecode

Integration

- https://github.com/graalvm/graalpython/releases/download/vm-1.0.0-rc2/python-installable-linux-amd64.jar



```
GraalPython
  python-installable-linux-amd64
    jre
      languages
        python
          bin
          doc
          include
          lib-graalpython
            modules
          lib-python
      lib
    META-INF
```

graalpython	2018/4/16 3:19	文件	139,683 KB

- GraalVM EE 1.0.0 RC2

```
graalvm-ee-1.0.0-rc2
├── 3rd_party_licenses_graalpython.txt ->
├── 3rd_party_licenses.txt
├── bin
├── COPYRIGHT
├── db
├── GRAALVM-README.md
├── include
├── javafx-src.zip
├── jre
├── lib
├── LICENSE
├── LICENSE_GRAALPYTHON -> jre/languages/
├── man
├── README.html
├── release
├── src.zip
├── THIRDPARTYLICENSEREADME-JAVAFX.txt
└── THIRDPARTYLICENSEREADME.txt
```

```
jre
├── bin
├── COPYRIGHT
├── languages
├── lib
├── LICENSE
├── plugin
├── README
├── THIRDPARTYLIC
├── THIRDPARTYLIC
├── tools
└── Welcome.html
```

```
bin
├── ControlPanel
├── gemasrv
├── graalpython -> ../languages/python/bin/graalpython
├── gu
├── java
├── javaws
├── jcontrol
├── jjs
├── js
├── keytool
├── lli
├── native-image
├── node
├── npm
├── orbd
├── pack200
├── policytool
├── polyglot
├── rmid
├── rmiregistry
├── servertool
├── tnameserv
└── unpack200
```

```
languages
├── js
│   ├── asm-debug-all.jar
│   ├── bin
│   ├── graaljs.jar
│   ├── icu4j
│   ├── icu4j.jar
│   ├── include
│   ├── native-image.properties
│   ├── NODE_README.md
│   ├── npm
│   ├── README.md
│   └── trufflenode.jar
├── llvm
│   ├── bin
│   ├── libsulong.bc
│   ├── libsulong.so
│   ├── native-image.properties
│   ├── polyglot.h
│   ├── README.md
│   └── sulong.jar
└── python
    ├── 3rd_party_licenses_graalpython.txt
    ├── bin
    ├── doc
    ├── graalpython.jar
    ├── include
    ├── lib-graalpython
    ├── lib-python
    ├── LICENSE_GRAALPYTHON
    ├── native-image.properties
    ├── README_GRAALPYTHON.md
    └── release
```

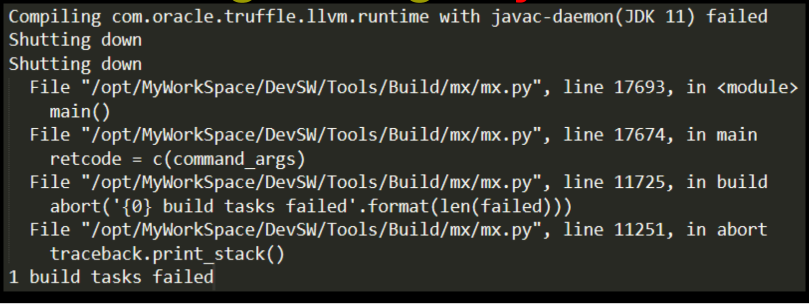
Practice

- https://github.com/AdoptOpenJDK/openjdk-jdk //OpenJDK11 src
- export JDK_BOOT_DIR=$YOUR_OPENJDK10_HOME
- reserve at least 6GB disk space
- on Laptop with Fedora 28 + Kernel 4.16.15 + GCC 8.1.1-1 + 8GB Memory (6GB DDR4 + 2GB Swap)
- cd $YOUR_OPENJDK11_SRCHOME and run the commands:

```
bash configure --disable-warnings-as-errors
make JOBS=4 images
```

#build GraalPython & GraalVM

- setup mx
- patching for avoid javaCompliance limitation
- Fail to build GraalVM with previously built OpenJDK 11, something wrong in javac?



```
Compiling com.oracle.truffle.llvm.runtime with javac-daemon(JDK 11) failed
Shutting down
Shutting down
  File "/opt/MyWorkSpace/DevSW/Tools/Build/mx/mx.py", line 17693, in <module>
    main()
  File "/opt/MyWorkSpace/DevSW/Tools/Build/mx/mx.py", line 17674, in main
    retcode = c(command_args)
  File "/opt/MyWorkSpace/DevSW/Tools/Build/mx/mx.py", line 11725, in build
    abort('{0} build tasks failed'.format(len(failed)))
  File "/opt/MyWorkSpace/DevSW/Tools/Build/mx/mx.py", line 11251, in abort
    traceback.print_stack()
1 build tasks failed
```

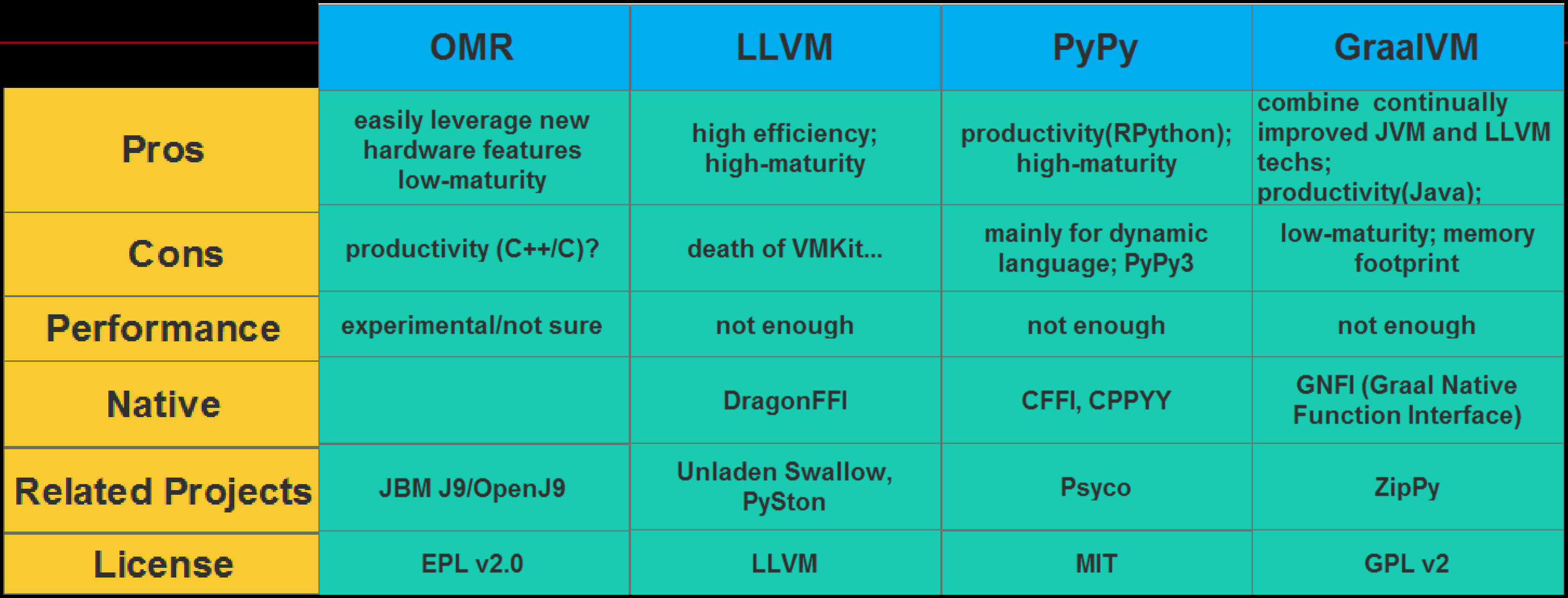
- Successfully build GraalPython via JDK 10

challenges

- prone to break build
- deal with JDK, Truffle/Graal, LLVM…
- customize GraalPython to meet our need
- dynamically enable or reload Graal compiler at runtime
- …

Rethinking of Python Runtime

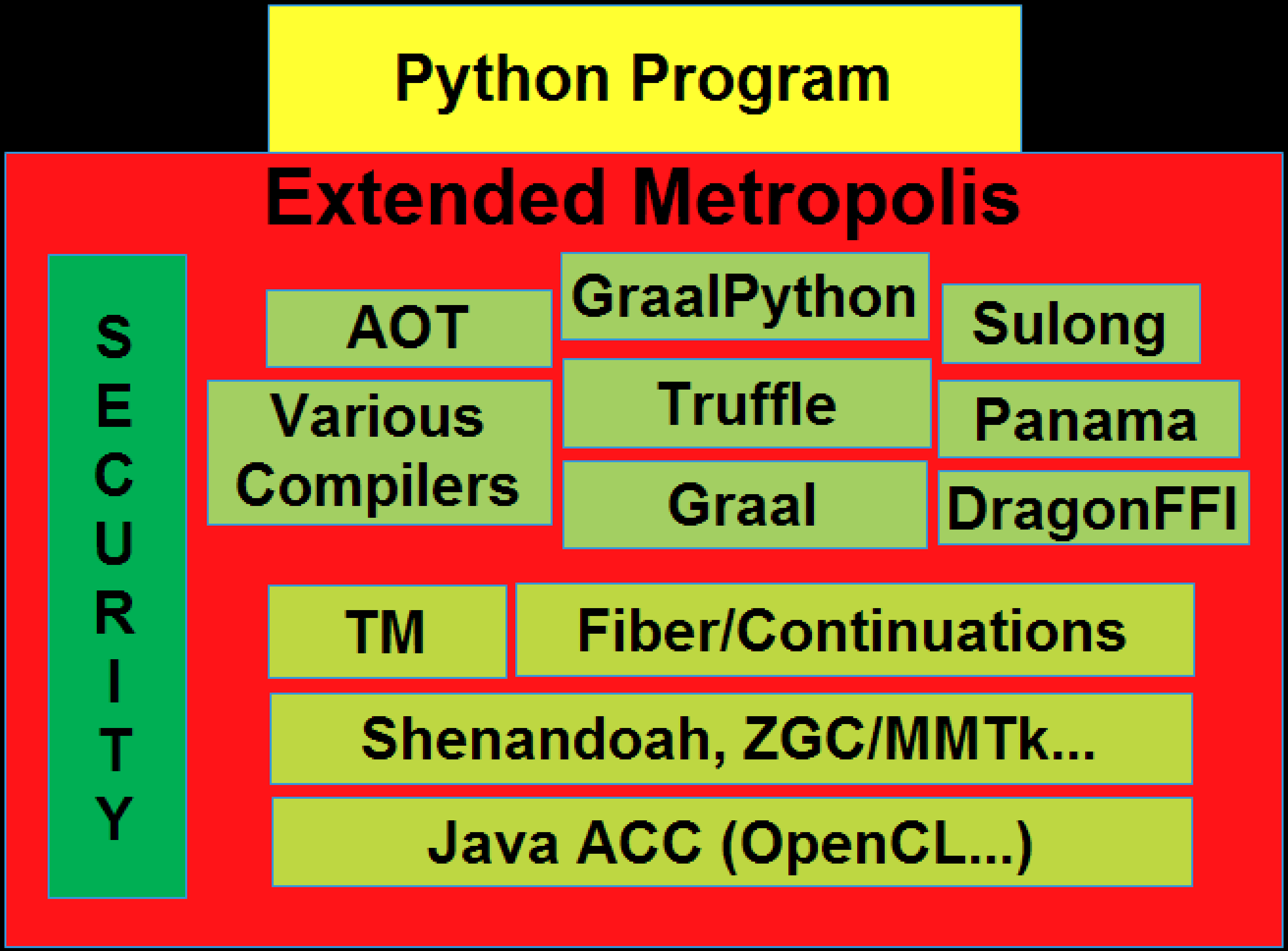
- **from my point of view, various Runtime Frameworks for Python implementation:**



	OMR	LLVM	PyPy	GraalVM
Pros	easily leverage new hardware features low-maturity	high efficiency; high-maturity	productivity(RPython); high-maturity	combine continually improved JVM and LLVM techs; productivity(Java);
Cons	productivity (C++/C)?	death of VMKit...	mainly for dynamic language; PyPy3	low-maturity; memory footprint
Performance	experimental/not sure	not enough	not enough	not enough
Native		DragonFFI	CFFI, CPPYY	GNFI (Graal Native Function Interface)
Related Projects	JBM J9/OpenJ9	Unladen Swallow, PySton	Psyco	ZipPy
License	EPL v2.0	LLVM	MIT	GPL v2

Future

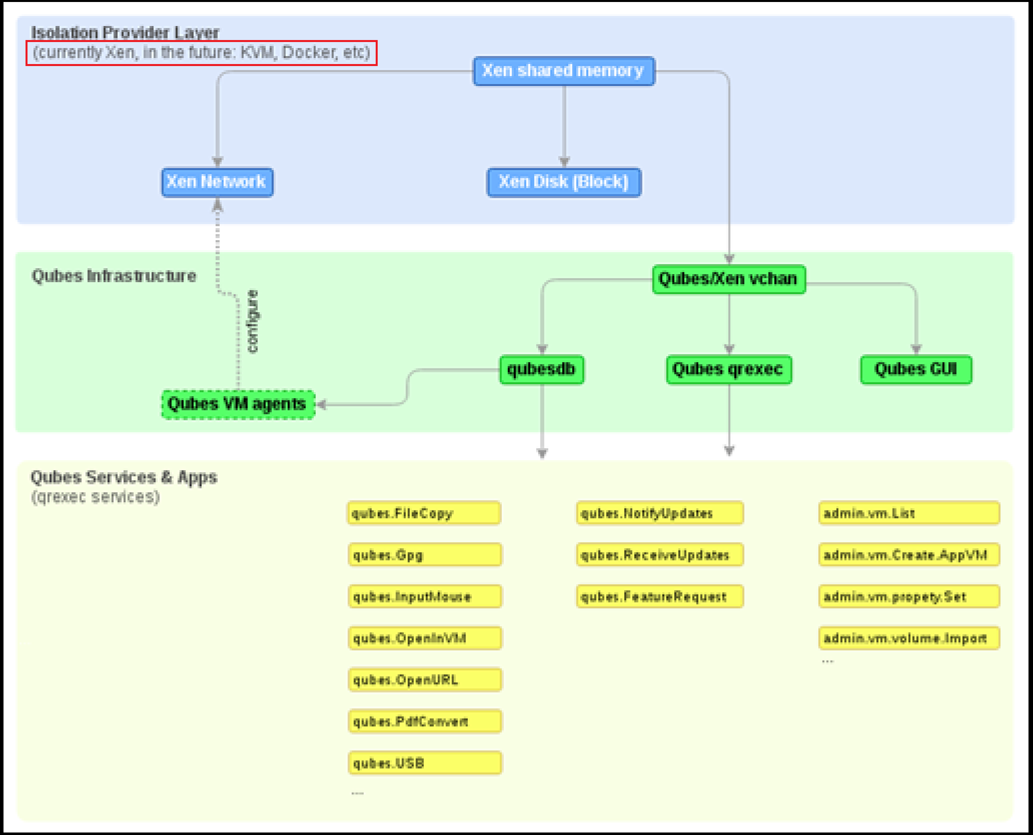
- http://openjdk.java.net/projects/metropolis/
- extend Project Metropolis and customize it for Python



III. Future Evolution & Re-design

1) Official *Generalization*

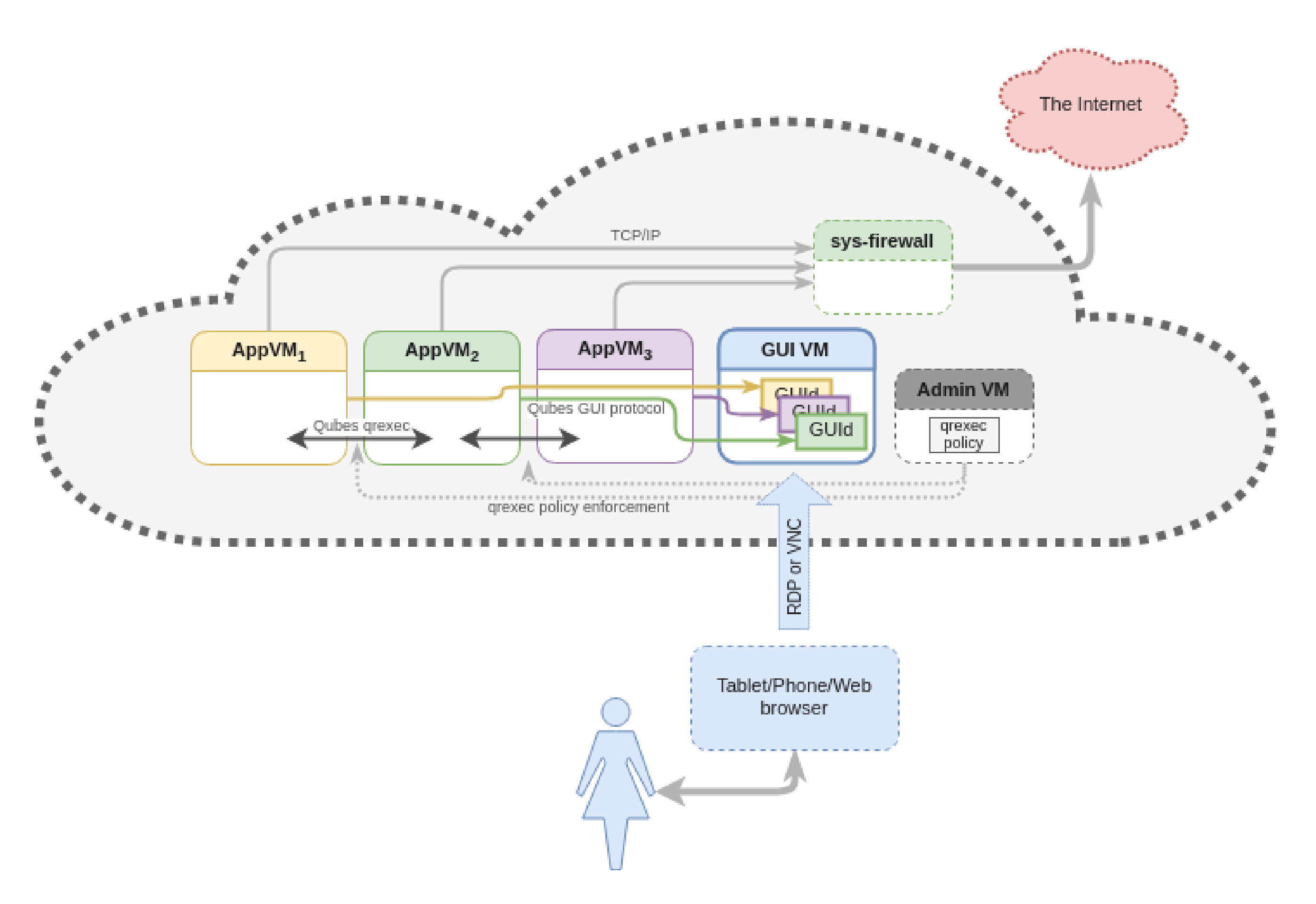
- **Generalizing the Qubes Architecture**



Qubes Remote Execution (qrexec)

Qubes Air

- https://www.qubes-os.org/news/2018/01/22/qubes-air/
- Qubes in the Cloud



2) Re-designing, Re-engineering, Re-inventing

- A customized Linux distribution for Dom0 with various optimization
- Integrate GraalVM-based customized runtime for accelerating Toolstack & Applications in Dom0/DomU
- Support Wayland display server in TemplateVM
- …

Q & A

Thanks!

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- https://en.wikipedia.org/wiki/Qubes_OS
- https://www.python.org
- http://llvm.org
- https://en.wikipedia.org/wiki/CPython
- https://en.wikipedia.org/wiki/Just-in-time_compilation
- https://github.com/dropbox/pyston
- …