Distributed Analysis Scheme

Xin SHI

12 May 2017

What is the problem?

- From hardware to software to physics … everything is changing … fast!
- For project leaders: hard to maintain stable team
- For newcomers: hard to get involved in short time

Two Parts

- Distributed Analysis Scheme
- Git Basic

Lesson from Linux Distribution

- Linux Distribution: Hundreds of distributions
  - Page Hit Ranking: Mint, Debian, Ubuntu ...
  - Mint based on Ubuntu, Ubuntu based on Debian !

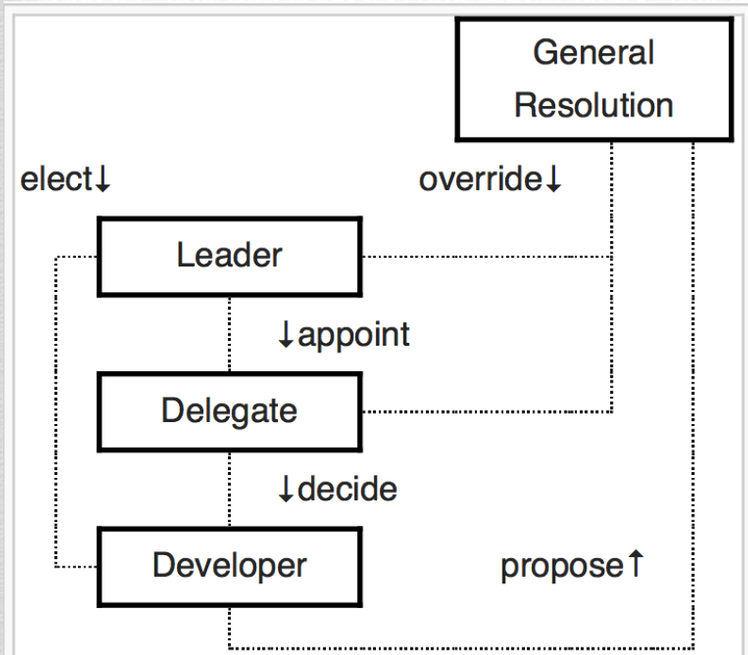
Page Hit Ranking

Data span: Last 6 months Go

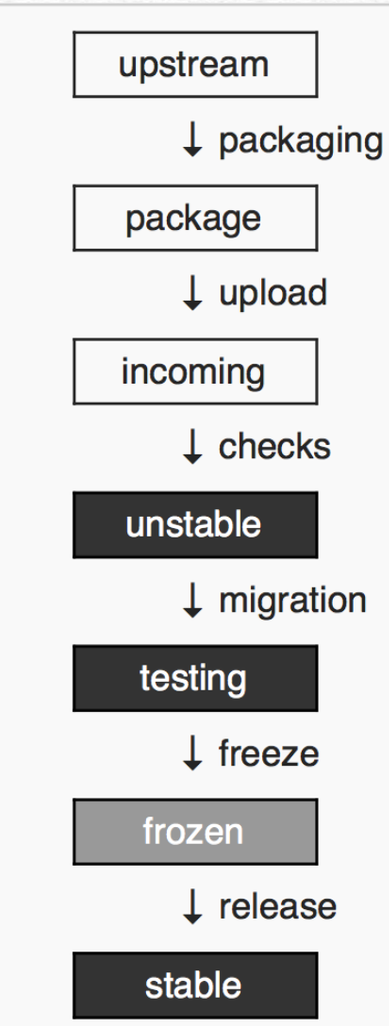
Rank	Distribution	H.P.D*
1	Mint	2947–
2	Debian	1764▼
3	Ubuntu	1648▼
4	openSUSE	1142▼
5	Manjaro	1109▼
6	Fedora	1071–
7	elementary	898▼
8	Zorin	828▼
9	CentOS	817▼
10	Arch	787▼
11	Mageia	717▼
12	PCLinuxOS	707▼
13	Ubuntu MATE	688▼
14	deepin	685▼
15	Slackware	601▼
16	Android-x86	594▼
17	LXLE	557▼
18	Antergos	518–
19	Lubuntu	511▼
20	Puppy	501–
21	FreeBSD	482▼
22	Lite	482▼
23	antiX	435–
24	ReactOS	430–
25	Solus	427▲
26	Tails	406▼
27	Xubuntu	391▼
28	Simplicity	387▲
29	KaOS	335▲
30	Kali	329–

Debian Develop Model

- Organization, lifecycle and **package** system



Simplified organizational structure



Flowchart of the life cycle of a Debian package

http://distrowatch.com/table.php?distribution=debian

Full Package List	unstable	testing	8.0	7.0
Package	**unstable sid**	**testing stretch**	**8.0 jessie**	**7.0 wheezy**
abiword (3.0.1)	3.0.1	3.0.1	3.0.0	2.9.2
alsa-lib (1.1.2)	1.1.2	1.1.2	1.0.28	1.0.25
ati-driver (15.201.1151)	--	--	14.301.1001	8.980
bash (4.3.30)	4.3	4.3	4.3	4.2
bind (9.10.4-P2)	9.10.3	9.10.3	9.9.5	9.8.4-P1
chromium (52.0.2743.116)	52.0.2743.116	52.0.2743.116	41.0.2272.118	26.0.1410.43
cups (2.1.4)	2.1.4	2.1.4	1.7.5	1.5.3
dhcp (4.3.4)	4.3.4	4.3.4	4.3.1	4.2.2
e2fsprogs (1.43.1)	1.43.1	1.43.1	1.42.12	1.42.5
firefox (48.0.2)	48.0	45.3.0	31.6.0	10.0.12
freetype (2.6.5)	2.6.3	2.6.3	2.5.2	2.4.9
gcc (6.2.0)	6.1.1	6.1.1	4.9.2	4.7.2
gimp (2.8.18)	2.8.16	2.8.16	2.8.14	2.8.2
glibc (2.24)	2.23	2.23	2.19	2.13
gnome-shell (3.20.4)	3.20.3	3.20.3	3.14.2	3.4.2
gnucash (2.6.13)	2.6.13	2.6.13	2.6.4	2.4.10
gnumeric (1.12.32)	1.12.32	1.12.31	1.12.18	1.10.17
grub (2.00)	2.02beta2	2.02beta2	2.02beta2	1.99
gtk+ (3.20.9)	3.20.9	3.20.9	3.14.5	3.4.2
httpd (2.4.23)	2.4.23	2.4.23	2.4.10	2.2.22
inkscape (0.91)	0.91	0.91	0.48.5	0.48.3.1
k3b (2.0.3a)	2.0.3a	2.0.3a	2.0.2	2.0.2
kmod (23)	22	22	18	9
libgnome (2.32.1)	2.32.1	2.32.1	2.32.1	2.32.1
libreoffice (5.2.0)	5.2.0	5.2.0	4.3.3	3.5.4
linux (4.7.2)	4.6	4.6	3.16.7	3.2.41

> 50, 000 packages!

https://en.wikipedia.org/wiki/Debian

Semantic Versioning

- Avoid “dependency hell” in software management
- Semantic Versioning: http://semver.org
- X.Y.Z = Major.Minor.Patch
  - Major: incompatible API changes
  - Minor: add backwards-compatible functionality
  - Patch: make backwards-compatible bug fixes

Three levels of programing language

Remember: the main goal of any “language” is for communication! One writes a piece of code not for himself, but for others to read as well! Unless one has strong reason, we suggest to use the following convention:

- Core algorithm: C++
- System level: python
- User level: bash

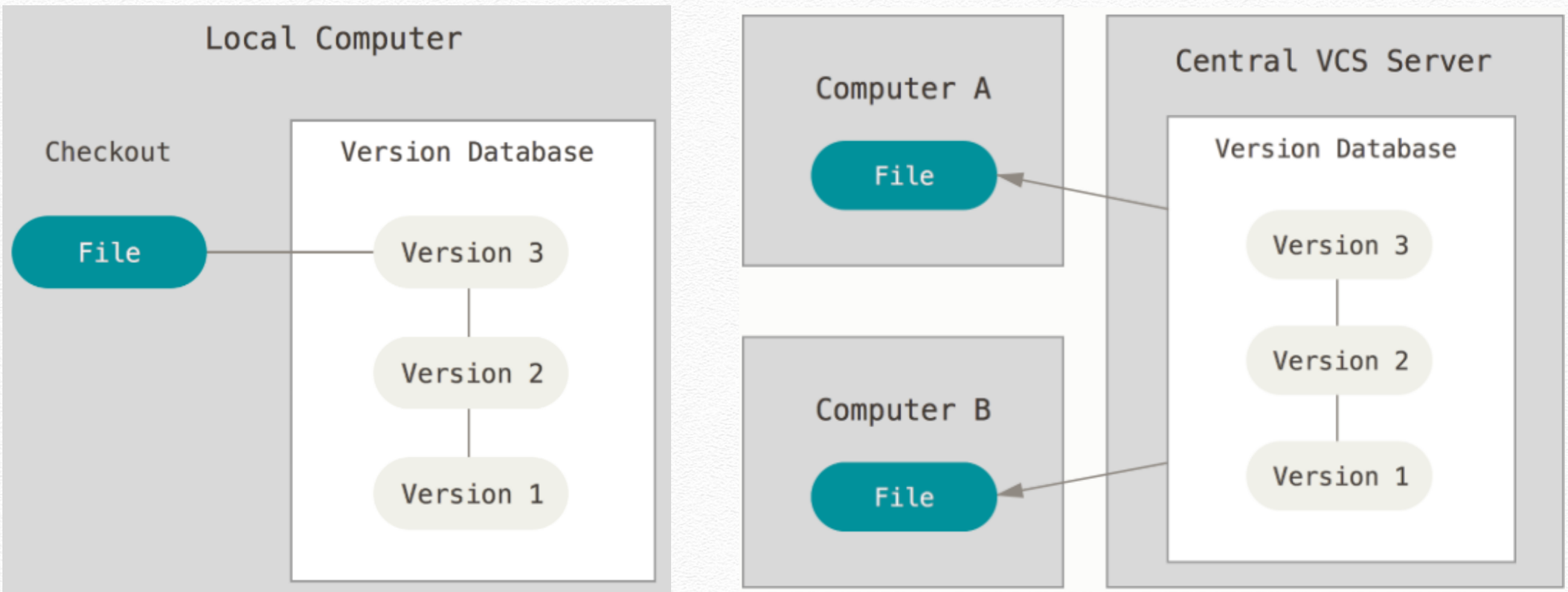
Always document well!

```
usage() {
    printf "NAME\n\tsubmit.sh - Main driver to submit jobs\n"
    printf "\nSYNOPSIS\n"
    printf "\n\t%-5s\n" "./submit.sh [OPTION]"
    printf "\nOPTIONS\n"
    printf "\n\t%-5s  %-40s\n"  "0.1.1"    "Run on signal samples"
    printf "\n\t%-5s  %-40s\n"  "0.1.2"    "Run on background samples"
    printf "\n\t%-5s  %-40s\n"  "0.1.3"    "Draw plots of signal and background"
    printf "\nDATE\n"
    printf "\n\t%-5s\n" "AUGUST 2016"
}
```

Git Basic

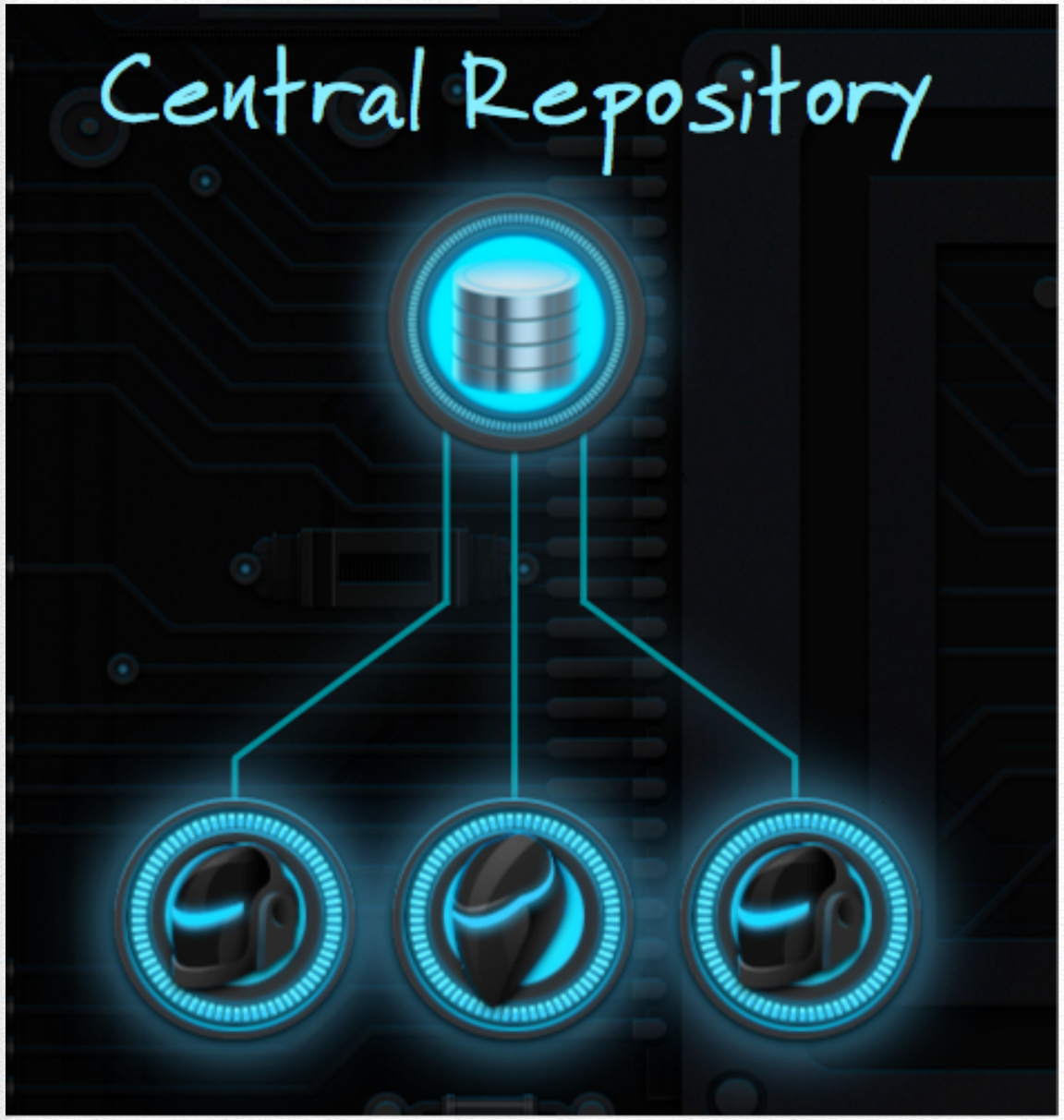
Why Version Control?

- Better debugging and collaborating



local central

central vs. distributed



Projects using Git

- Linux Kernel Source code (1.6GB)
  https://github.com/torvalds/linux

Linus Torvalds
torvalds

- LHC: CMS core software (570MB)
  https://github.com/cms-sw/cmssw

  - 2,183 forks / 617 contributors

basic git workflow

- Check out from remote: $ git clone remote.server/code.git
- Update: $ git pull
- Add file: $ git add abc.txt
- Commit: $ git commit -m “comment message” abc.txt
- Push to remote: $ git push

fast local operation

workflow in github.com

- Register on github.com (better with short id)
- Fork the repo
- Make your change / contribution
- Push to your own repo (forked repo)
- Create Pull Request
- Get approved and merged to the official repo
- CMS Example: HLT Photon + Jet in DQM code
  https://github.com/cms-sw/cmssw/pull/4946/

Further info

- Quick start: http://www.codeschool.com/courses/git-real

- In-depth: (中文) https://git-scm.com/book/zh/v2