University of Zurich[UZH]

High Performance Computing Lecture 1

Douglas Potter
Cesare Cozza
Mark Eberlein

Organization

- 14 Lectures and 14 Exercise Classes
  - Wednesday: Concepts and overview
  - Friday: Hands-on exercises
  - 6 ECTS = 8+ hours per week
- Microsoft Teams: ESC401 team
  - https://teams.Microsoft.com/ (or application)
- Laptop required
  - CSCS Account – follow email instructions
  - Need an “ssh” terminal
    - Linux & OS X: already there
    - Windows: putty or WSL2
    - Part of the first exercise sheet ☺

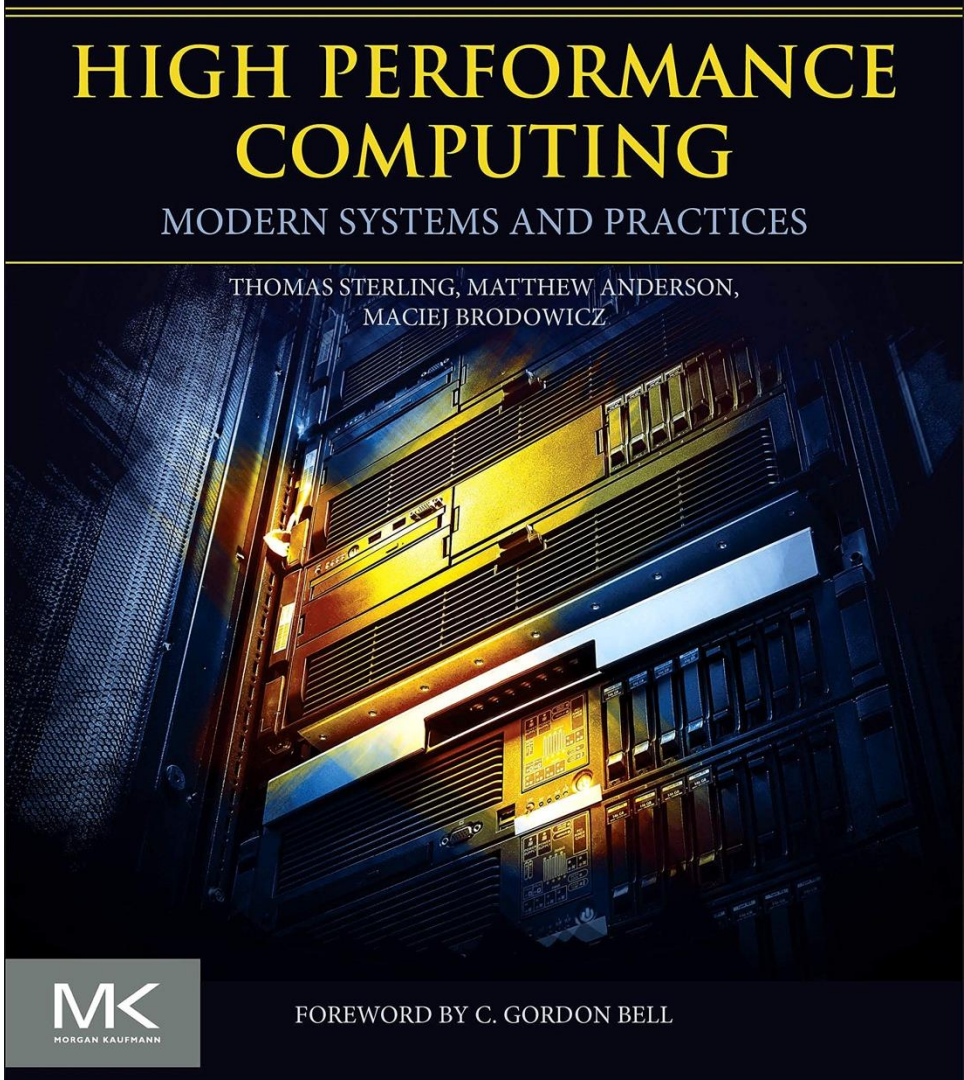
Prerequisites and Grading

- No prerequisites, but,
  - Having experience with C would help
  - Having written scripts (Python, BASH, etc.)
- Goals
  - Understand High Performance Computing
  - Understand how to make things faster
  - Use HPC resources
  - Change a serial code to a parallel code
- Grading
  - Must get 60% on Exercises
  - Examination on January 14th (tentative)
    - Good results on the exercises can improve grade

Optional

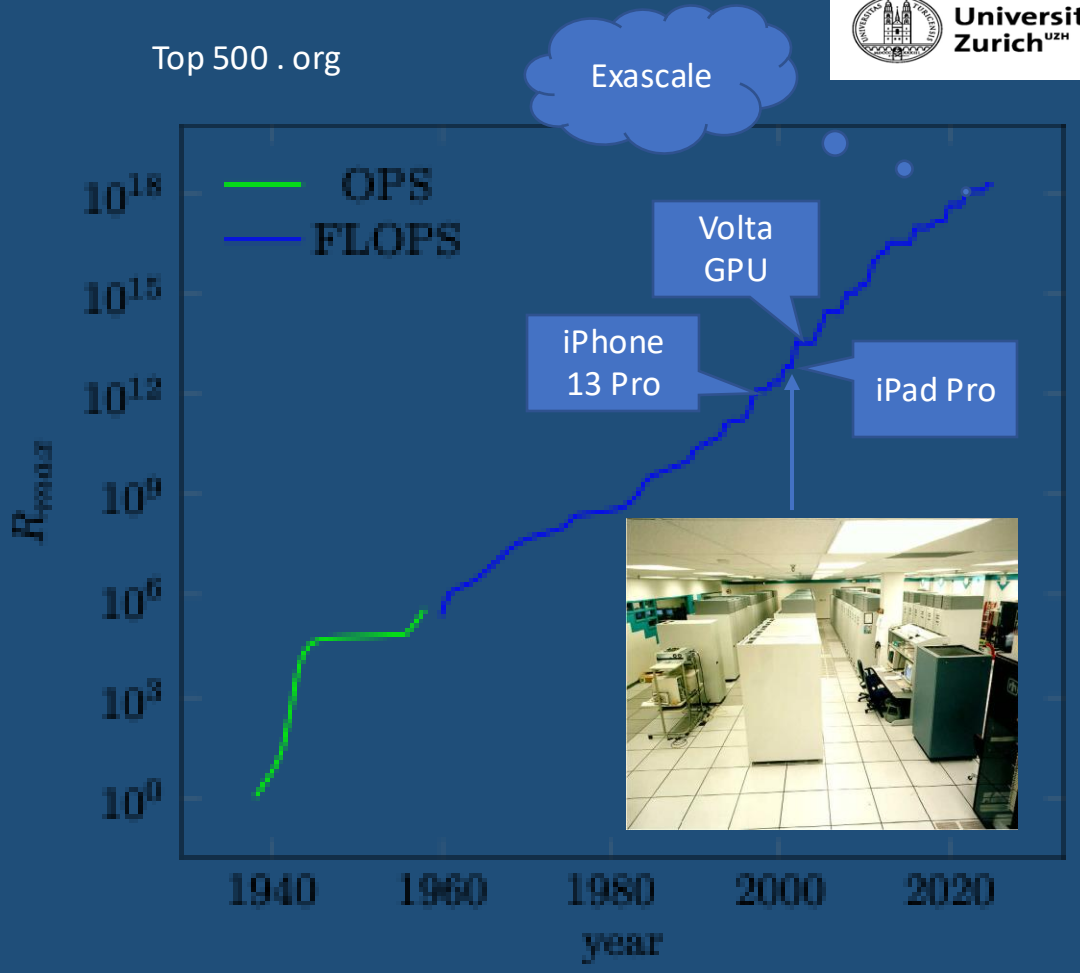
- Not required for the course
- Much greater detail on HPC topics
- Available in the Library
- … and on Kindle

- Other resources listed in Teams
- … including these slides



What is High Performance Computing?

- Time to solution
- Normal progression
  - Laptop
  - Workstation
  - Server
  - Special Hardware
  - Supercomputer
- More and more?
  - FLOPS generally
  - I/O maybe
  - Memory capacity
  - Memory speed



Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
2	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P , NUDT National Super Computer Center in Guangzhou China	3,120,000	33,862.7	54,902.4	17,808
3	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray/HPE Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272
4	**Gyoukou** - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz , ExaScaler Japan Agency for Marine-Earth Science and Technology Japan	19,860,000	19,135.8	28,192.0	1,350
5	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray/HPE DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209
6	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , IBM DOE/NNSA/LLNL United States	1,572,864	17,173.2	20,132.7	7,890

Top500.org

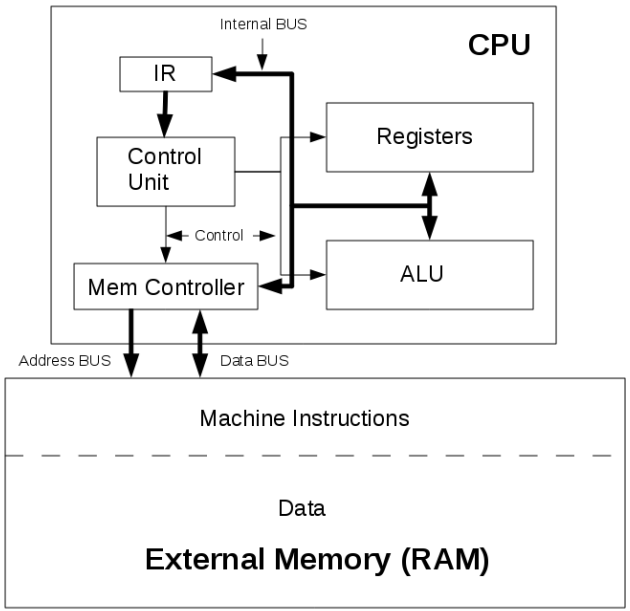
November 2017

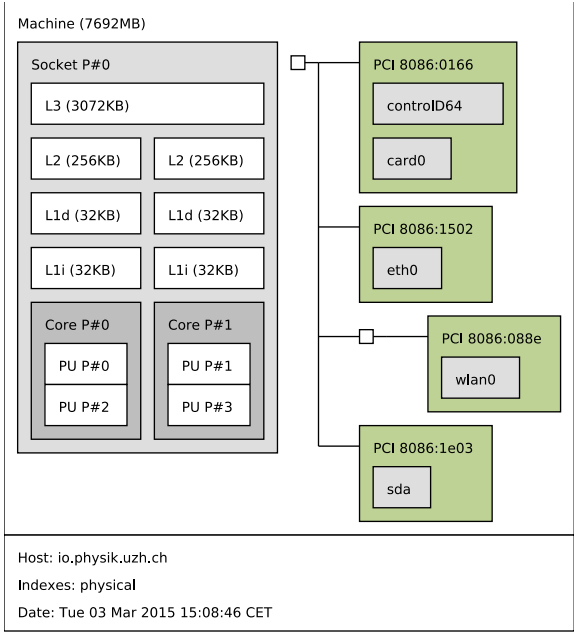
Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
6	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194
43	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21.23	27.15	2,384

Top500.org

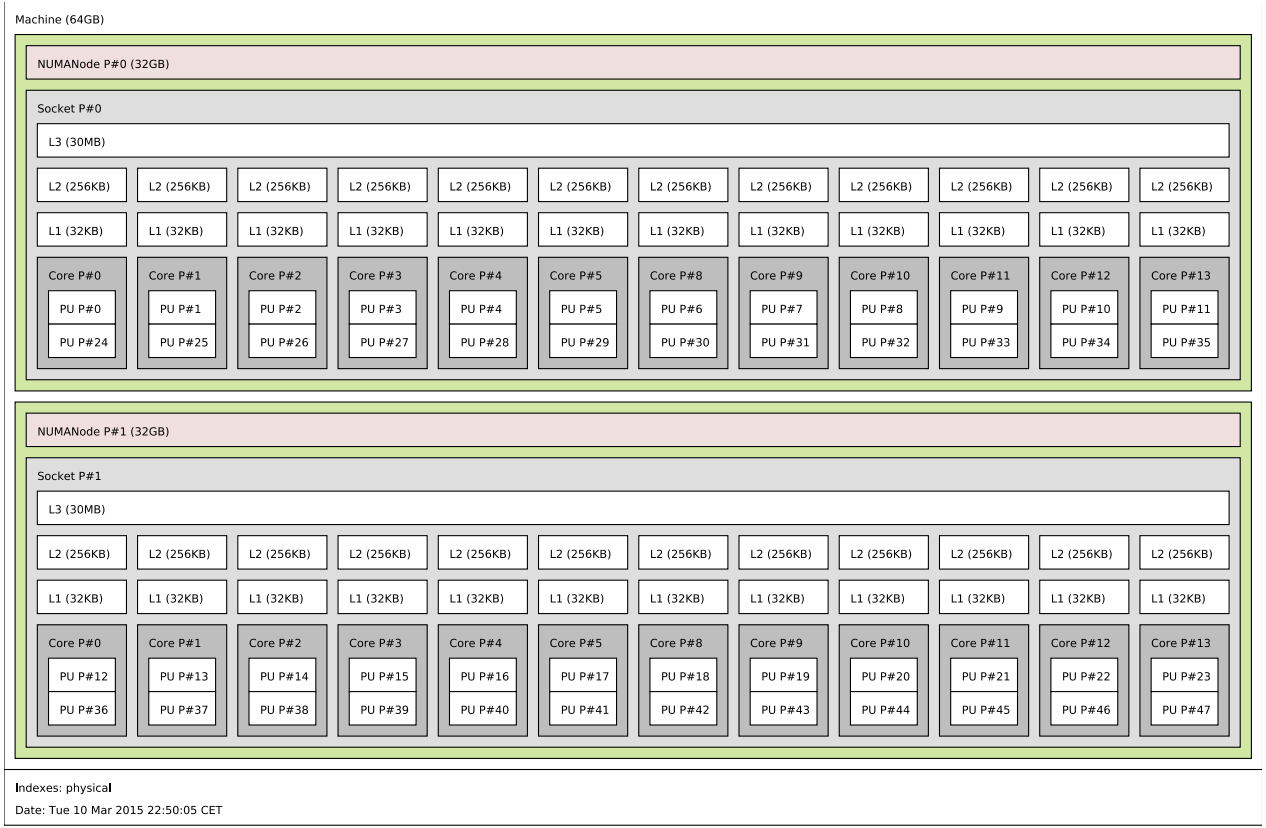
June 2024

What is a computer really?

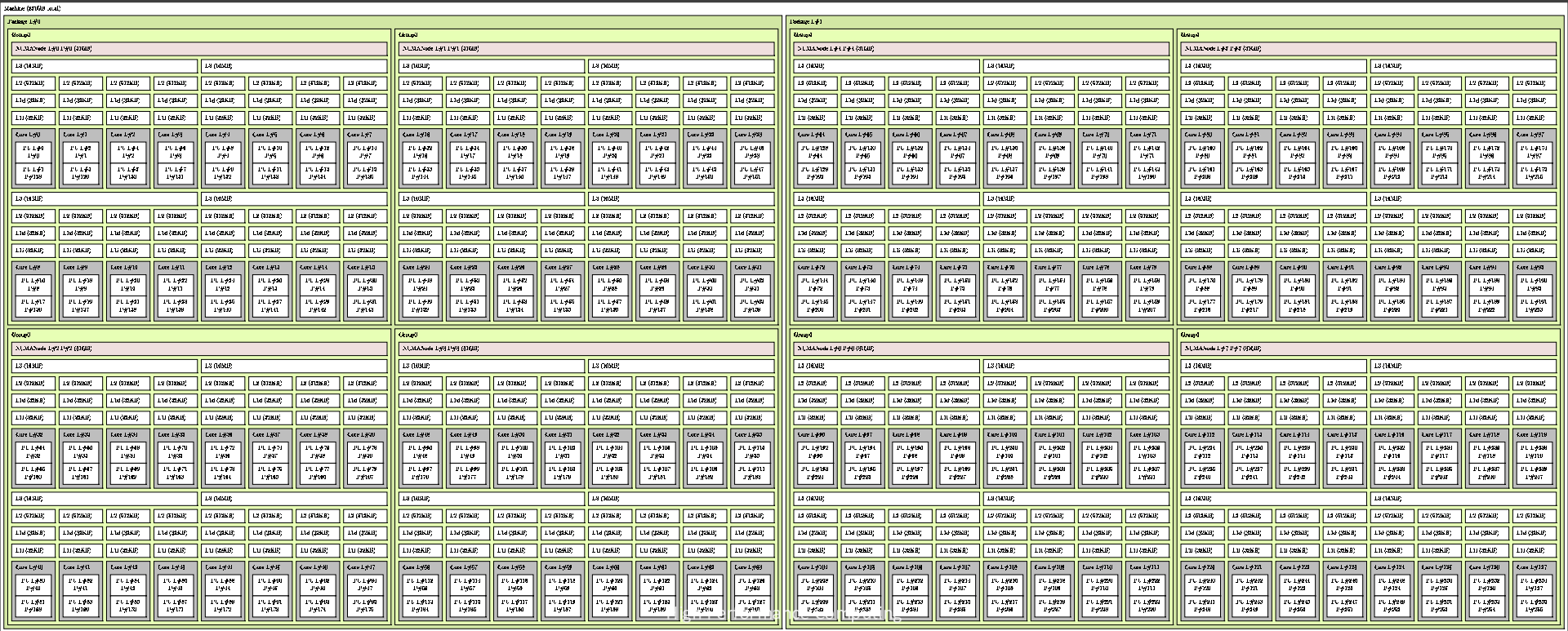




A 24 core NUMA node



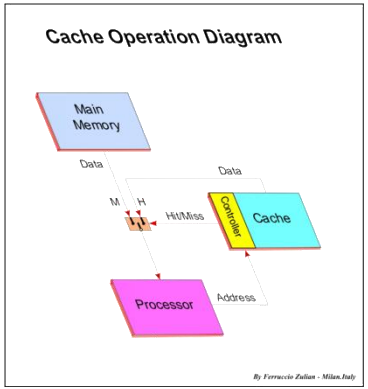
“Eiger” – 128 cores

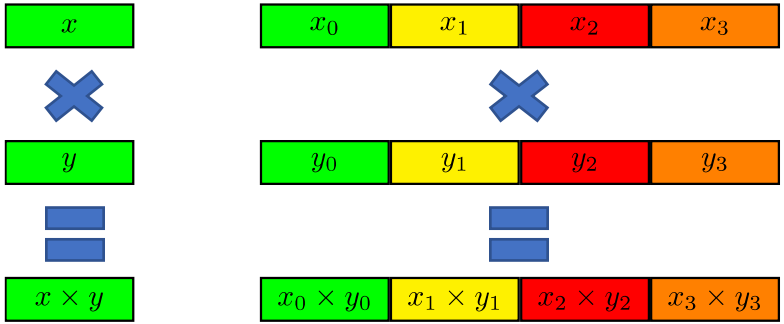


GPU Computing – P100 – 3584 “cores”

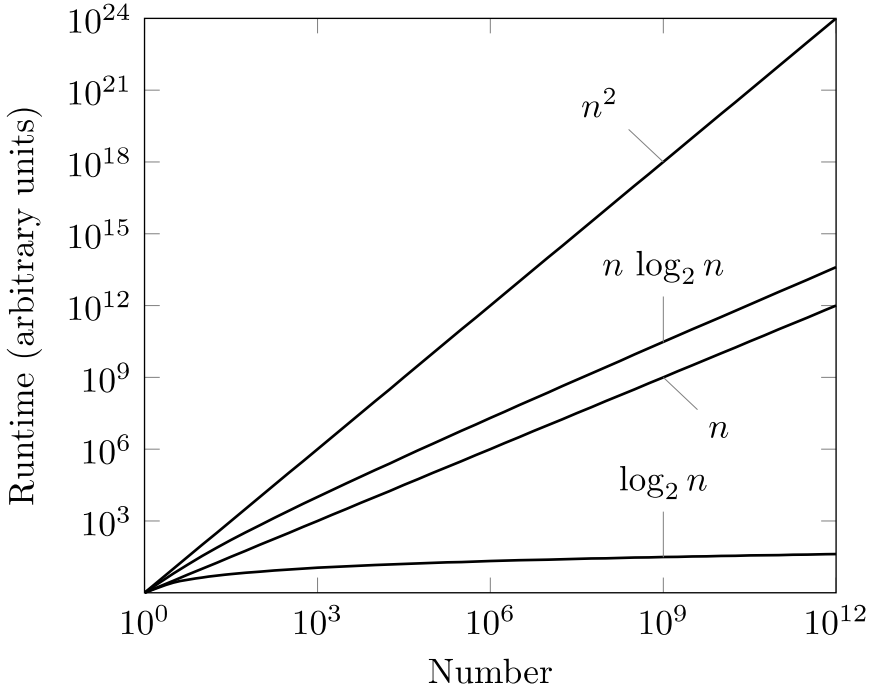
How do we make “it” faster?

- Increase the clock speed
  - 1 MHz to 4 GHz in around 35 years
  - 12 doublings of performance
- Optimize the hardware
  - Faster memory
  - Cache
  - Cycles per operation
- SIMD: AVX or SSE





Algorithmic Changes



	Thousand	Million	Billion	Trillion
Log N	10 ns	20 ns	30 ns	40 ns
N	1 us	1 ms	1 s	17 minutes
N log N	10 us	20 ms	30 s	11 hours
N^2	1 ms	17 minutes	32 years	32 mega-years

How do we make “it” faster 2

More is better?

- Add additional CPUs
- Increase the number of cores
- Add additional nodes

How do we use all this?

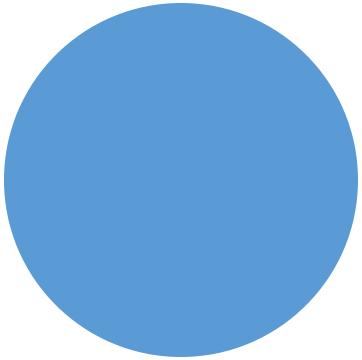
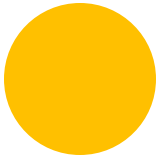
- Embarrassingly parallel
- Goal: truly parallel

Working on the “Same Problem”

- Multiple threads
- Message passing (MPI or local)
- ScienceCloud or other Cloud services
- Batch Clusters
- GPU Acceleration (+ more later)
  - Dedicated programs: Genga
  - To handle specific domains: RAMSES
  - Or as an accelerator: pkdgrav3 – Hybrid code
  - Also, GRAPE and Intel Phi

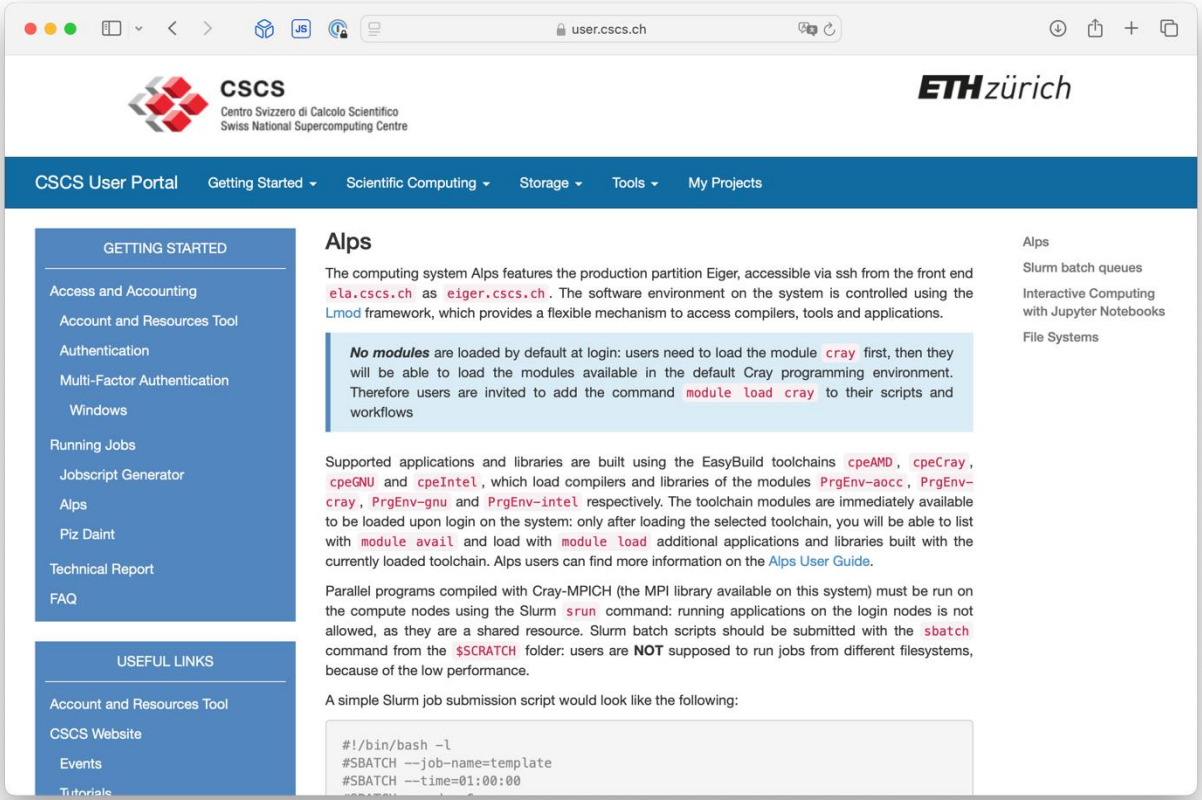
Super Computers

- A bunch of nodes together
- Special Interconnect (Network)
- Far away from you
- Runs Unix / Linux (usually)
- Non-interactive (usually)

Connecting to the Supercomputer

High Performance Computing



Online Documentation

Teletype

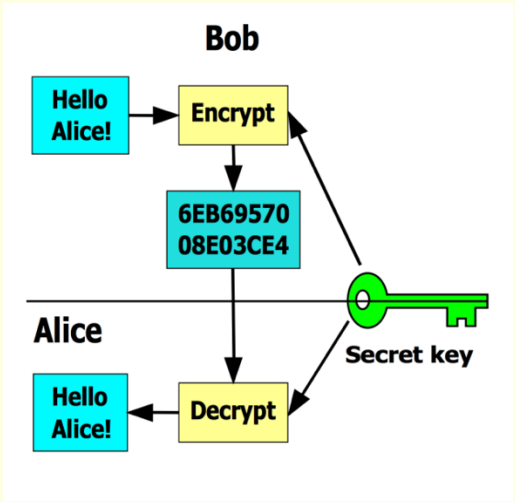
- Circa 1963
- MODEM
- Paired
- Paper tape
  - Writer
  - Reader

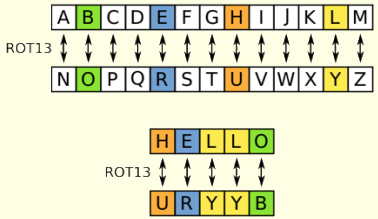
By Jason Scott

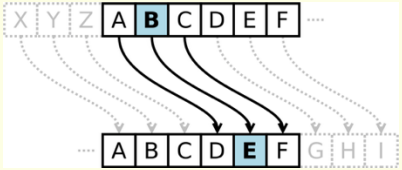
Terminals

- VT100 (1978)
- Direct connection
- RS-232 originally
- Later networked
- TELNET protocol (TELetype NETwork)
- SSH protocol (Secure SHell)
  - Encryption

Encryption



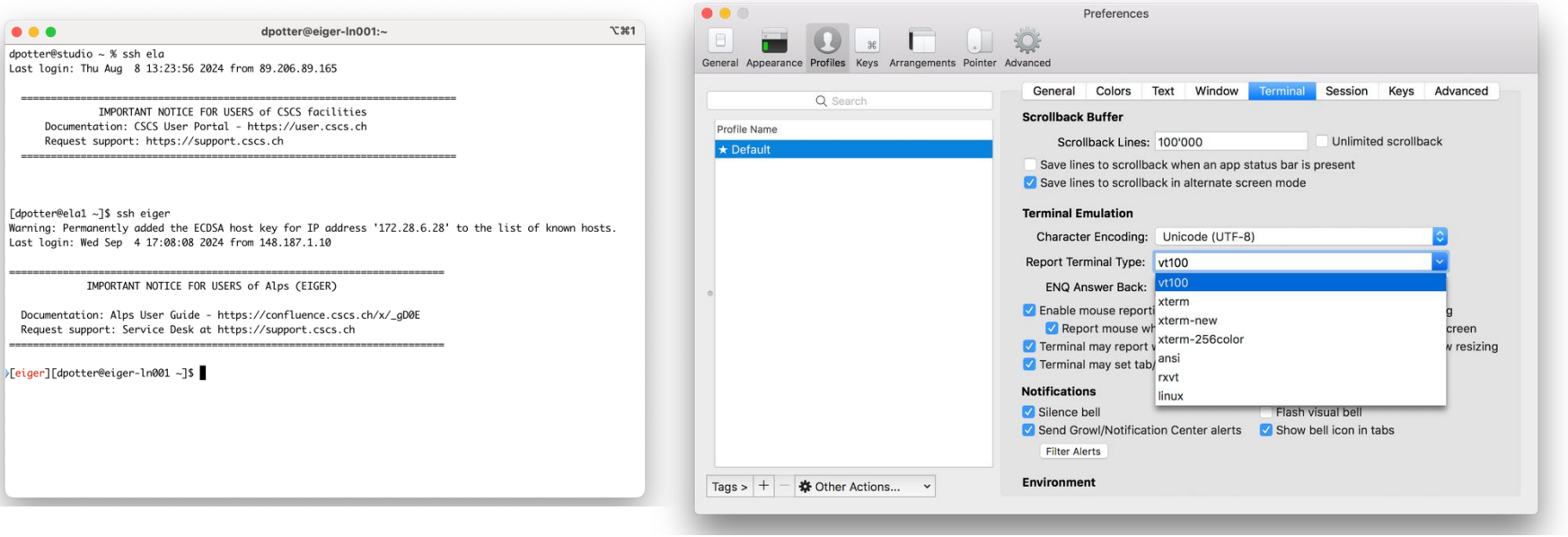




- First in ca. 1900 BC
- Julius Caesar 100 BC

- More complex keys
- E.g., ”One time PAD”

- Relies on sharing “keys”

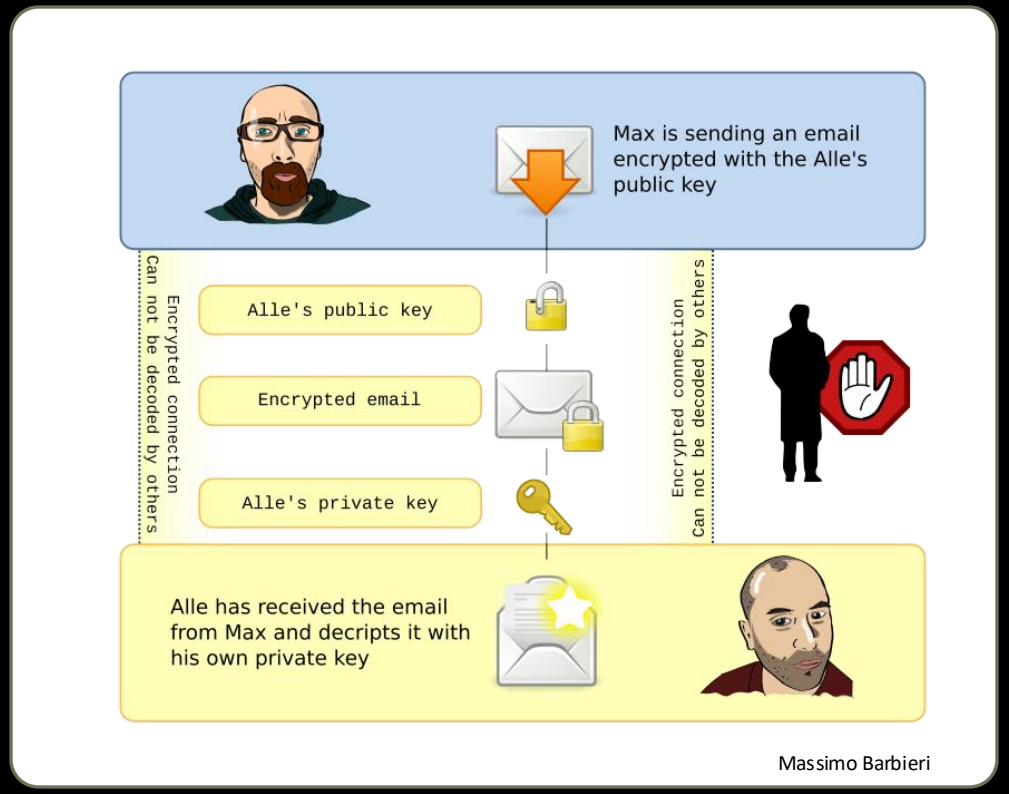


Secure Shell

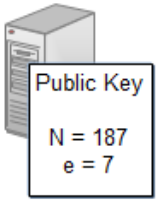
- We want to connect securely to remote systems…
- … but sharing keys in advance is cumbersome.

Public Key Encryption

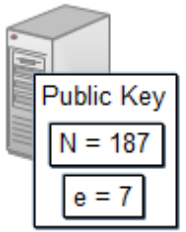
- Enter public keys
- Two keys: public & private
  - They are related.
- I “publish” my public key
  - And you publish yours.
  - Private keys we keep secret.
- Encrypt with a public key
  - Can decrypt with private key
  - I can send secret messages!
- Encrypt with a private key
  - Decrypt with public key
  - I can prove who I am!
- The details are more complicated of course.



- **PUBLIC KEY**
  - **2 giant primes, *p* and *q***
    - ***p* = 17, *q* = 11**
  - ***p* * *q* = N**
    - N = 187
  - **Pick another prime, *e***
    - *e* = 7



- **PRIVATE KEY**
  - **Private key = *d***
  - ***e* * *d* = 1(mod(p-1)*(q-1))**
  - **7 * *d* = 1(mod 16 * 10)**
  - **7 * *d* = 1(mod160)**
  - *d* = 23



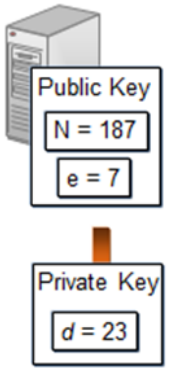
Private Key
d = 23

Factoring Prime Numbers

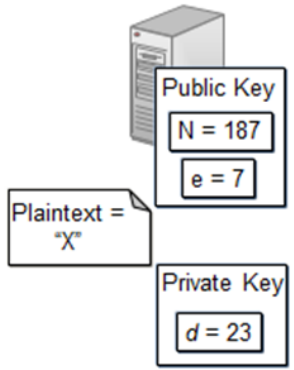
- Public key encryption depends on factoring primes being hard
- The security of the Internet depends on it
- Quantum computing introduces some concerns

- Let's send "X"
  - X = 88 ASCII
  - Encryption:
    - Ciphertext = $88^{e} (\text{mod } N)$
    - Ciphertext = $88^{7} (\text{mod } 187)$
    - Ciphertext = 11

Ciphertext
C = 11



- Decryption:
  - Plaintext = $C^{d} (\text{mod } N)$
  - Plaintext = $11^{23} (\text{mod } 187)$
  - Plaintext = 88
  - ASCII 88 = "X"
  - Plaintext = "X"



Factoring Prime Numbers

SSH Key Types

RSA

- Slowly being phased out
- Relies on prime numbers
  - Integer factorization

$N = p \times q$

- Find p and q given N

ED25519

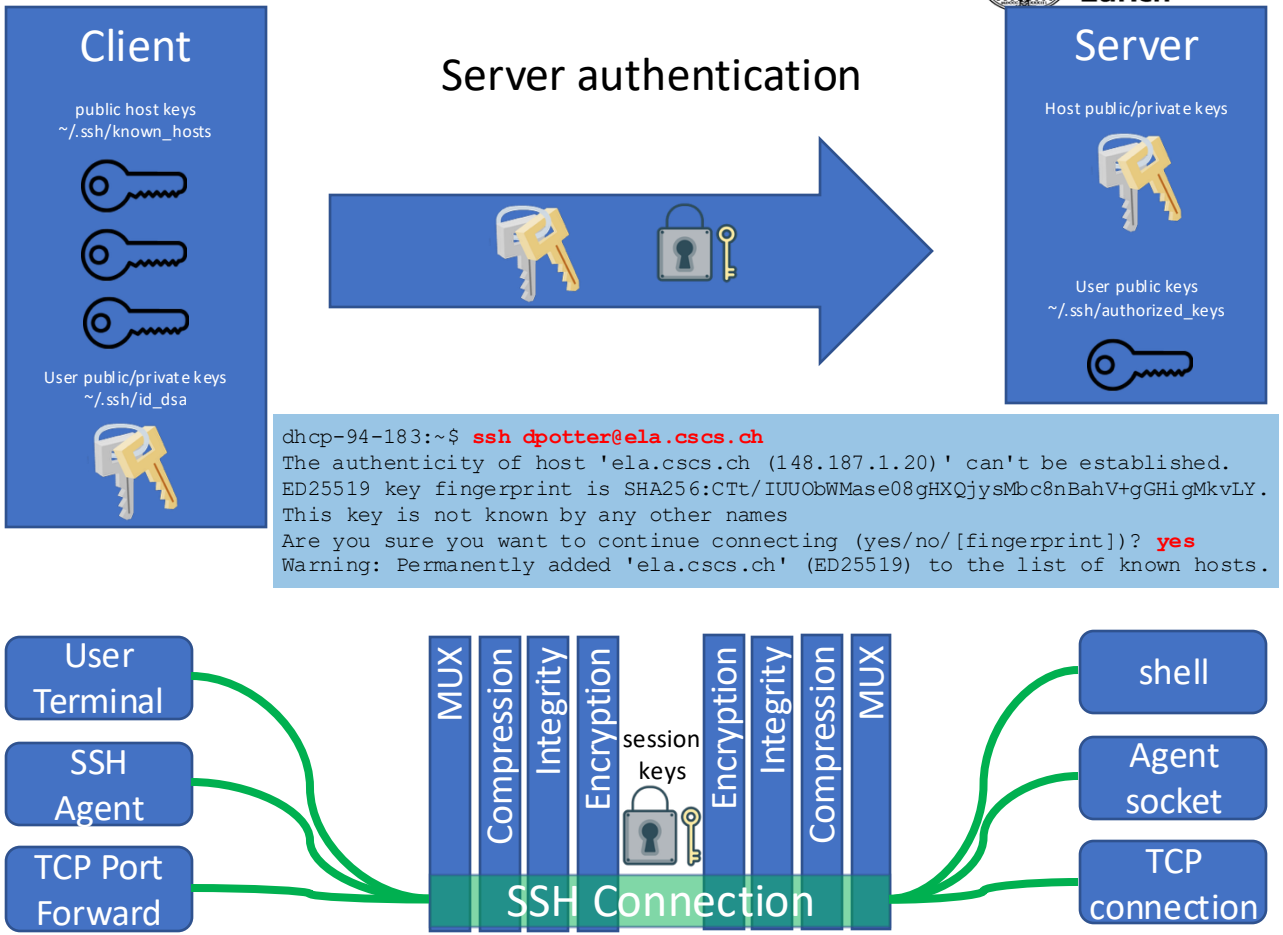
- Support on newer systems; faster
- Elliptical Curve
  - Discrete Logarithm problem

$x = a^b \, mod \, n$

- Solve for b given x, a, n

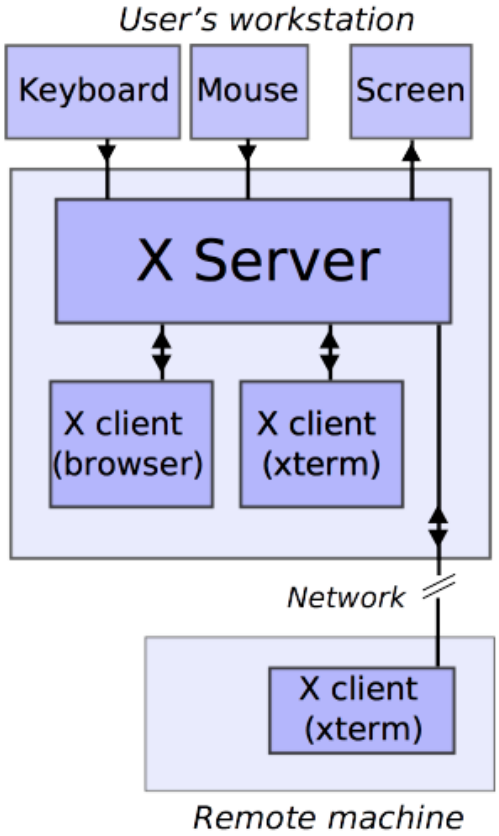
SSH Components

- Servers have public keys too!
- This way we can avoid man-in-the-middle attacks.
- Public host key is saved first time we connect to a host.
  - Thereafter it is verified.
- Session keys are exchanged for faster encryption & decryption.
- Traffic is “tunneled” through our connection.
  - Port forwarding for example.
- Public key can be uploaded
  - Login without a password.

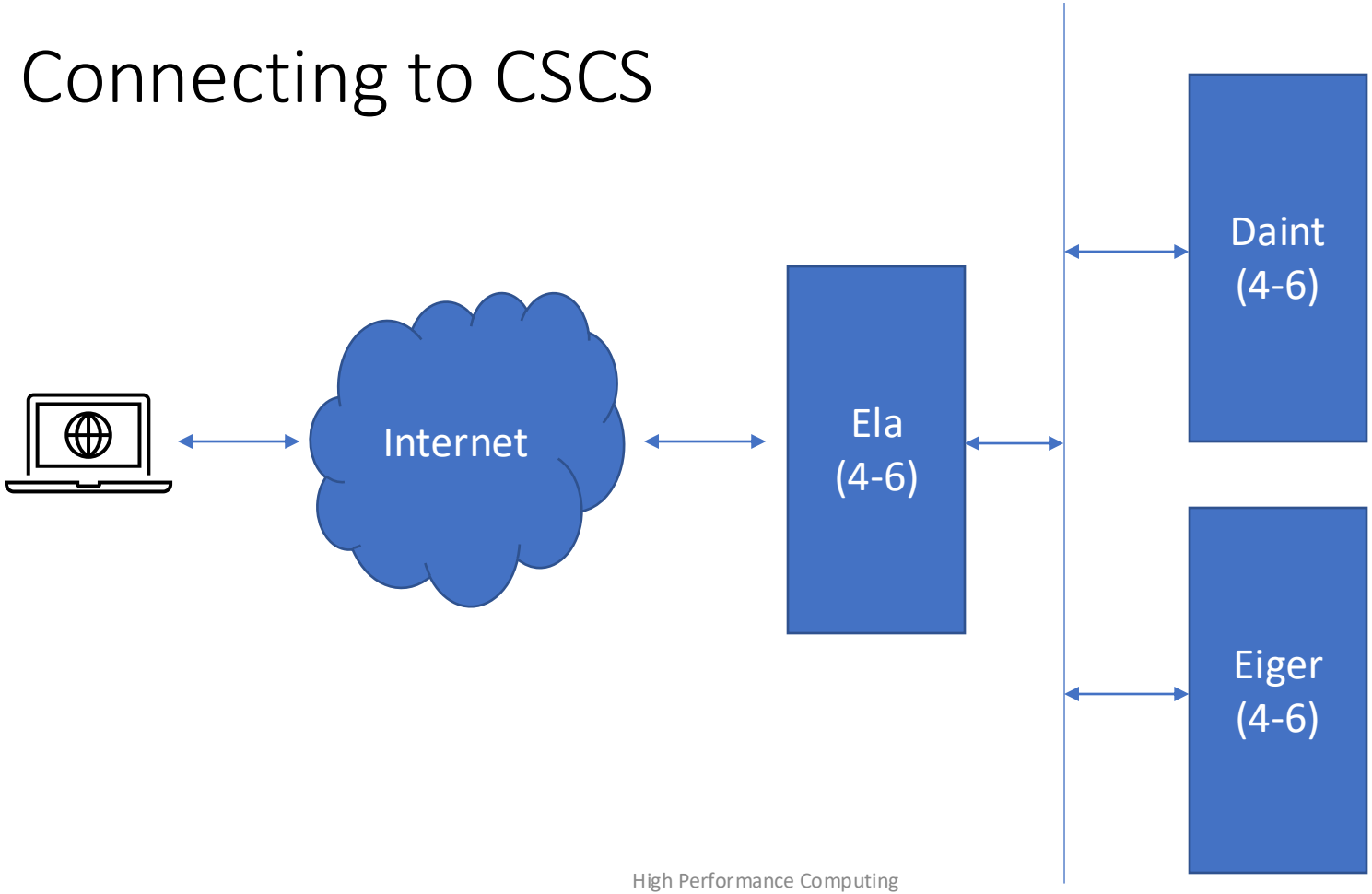


X11 for Graphical Interfaces

- Just “works” on Linux
- XQuartz on OS X
- VcXsrv on Windows
  - Not needed for WSL2 (Windows 11)
- DISPLAY environment variable
  - export DISPLAY=localhost:0
  - Just for windows probably
- “ssh” must tunnel usually
  - ssh -X hostname
  - ForwardX11 yes (config)



Connecting to CSCS



~/.ssh/config

```
Host ela
  Hostname ela.cscs.ch
  User dpotter
  ForwardX11 yes
  ForwardX11Trusted yes
  ForwardAgent yes
Host eiger
  ProxyJump ela
  User dpotter
  ForwardX11 yes
  ForwardX11Trusted yes
  ForwardAgent yes
```

```
dhcp-94-183:~$ ssh dpotter@ela.cscs.ch
[dpotter@ela3 ~]$
```

```
dhcp-94-183:~$ ssh ela
[dpotter@ela2 ~]$ ssh eiger
[eiger][dpotter@nid001080 ~]$
```

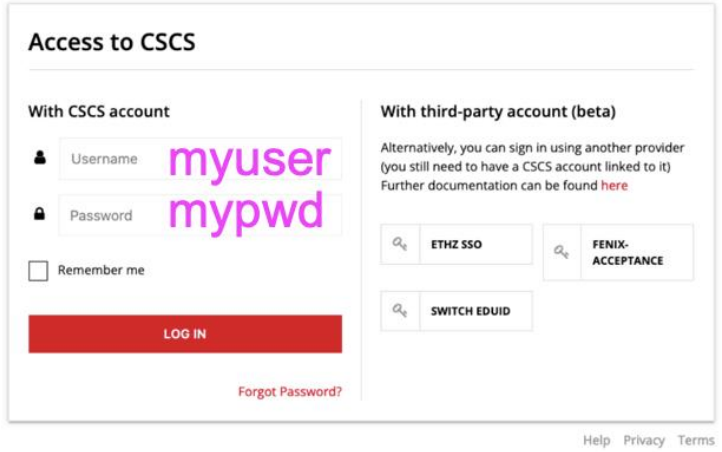
```
dhcp-94-183:~$ ssh eiger
[eiger][dpotter@nid001000 ~]$
```

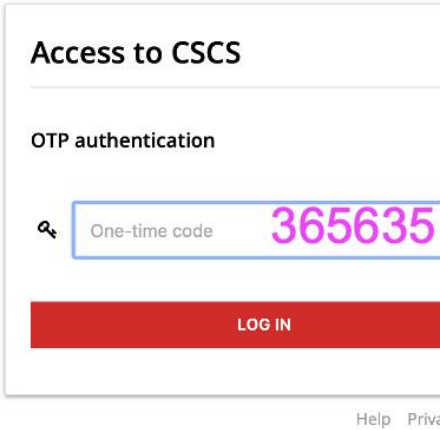
Putting it together

- SSH directory: ~/.ssh or $HOME/.ssh
- “config” file (man ssh_config)
- “known_hosts” – prevents “man in the middle”
- “authorized_keys” – keys allowed access
- “id_rsa” – your private key (default name)
- “id_rsa.pub” – your public key (default name)
- “ssh-keygen” – generate a public/private key pair
- “ssh-copy-id” – convenience program
- “ssh-agent” and “ssh-add”

Two Factor Authentication at CSCS

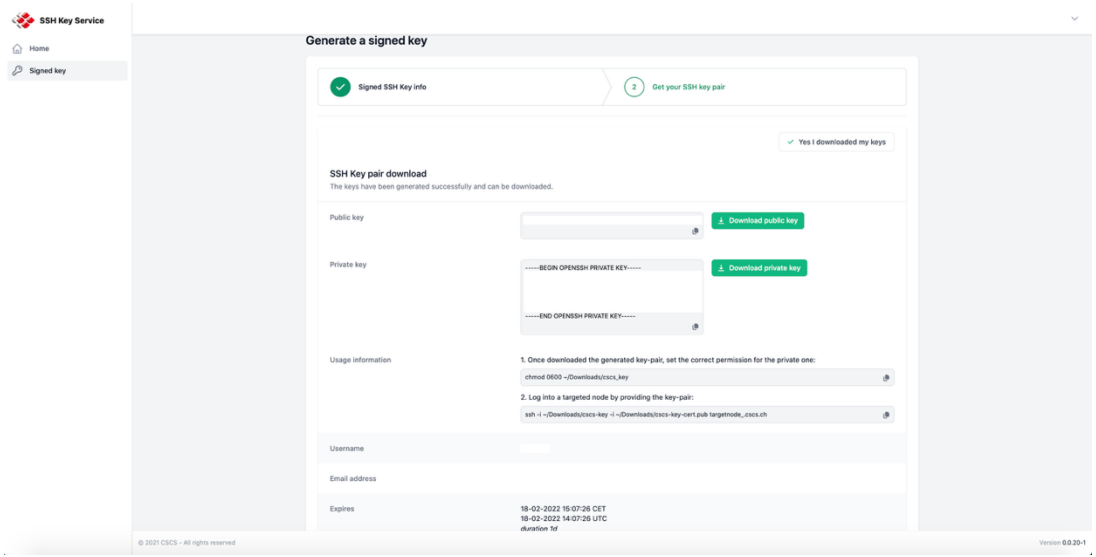
- New accounts use MFA
- You will need an Authenticator
  - Google Authenticator
  - Microsoft Authenticator
  - and so on
- You are used to this probably
  - Online services
  - Banks
  - Steam





Generate SSH keys with MFA

- CSCS requires MFA for keys
- The are valid 24 hours only!
- There is an online generator
- You can also run a script
- You need to provide
  - Username
  - Password
  - One Time Password (OTP)



Why do we use the Command Line?

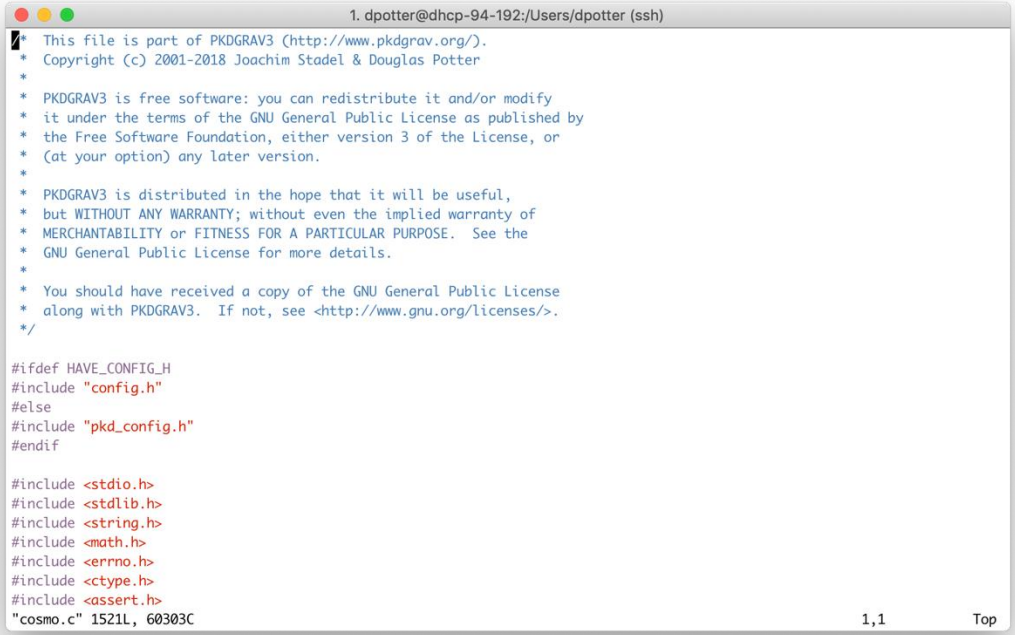
- Because we've always done it that way?
- Graphical User Interfaces are slow over distances
  - Web forms are possible (e.g., Jupyter Notebook)
- Need more flexibility
  - Build and run codes (ours and others)
  - Designed workflows.

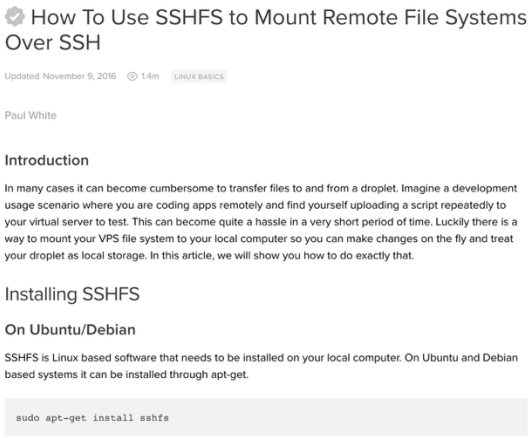
```
64      char achFile[256];                  /*DEBUG use MAXPATHLEN here (& elsewhere)? -- DCR*/
65      double dTime;
66      double E=0,T=0,U=0,Eth=0,L[3]={0,0,0},F[3]={0,0,0},W=0;
67      double dMultiEff=0;
68      long lSec=0;
69      int i,iStep,iStartStep,iSec=0,iStop=0;
70      uint64_t nActive;
71      int bKickOpen=0;
72      int bDoOpeningKick=0;
73      int bDoCheckpoint=0;
74      int bDoOutput=0;
75      int64_t nRungs[MAX_RUNG+1];
76      uint8_t uRungMax;
77      double diStep;
78      double ddTime;
79      int bRestore;
80
81      printf("%s\n", PACKAGE_STRING );
82
83      bRestore = msrInitialize(&msr,mdl,argc,argv);
84      msr->lStart=time(0);
85
86      /*
87      ** Establish safety lock.
88      */
89      if (!msrGetLock(msr)) {
90          msrFinish(msr);
91          return NULL;
92          }
93
94      /* a USR1 signal indicates that the queue wants us to exit */
95  #ifndef _MSC_VER
96      timeGlobalSignalTime = 0;
97      signal(SIGUSR1,NULL);
98      signal(SIGUSR1,USR1_handler);
```

Text Editors

High Performance Computing

“Vim”
(text only)





How To Use SSHFS to Mount Remote File Systems Over SSH

Updated November 9, 2016 1.4m LINUX BASICS

Paul White

Introduction

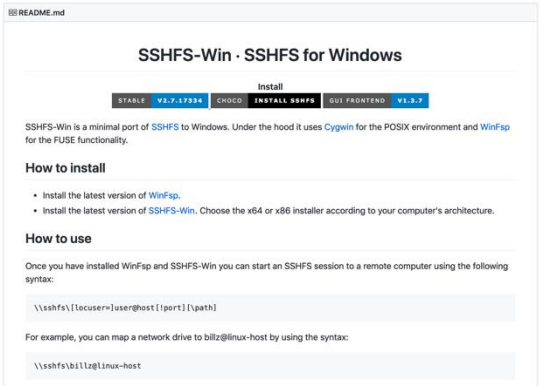
In many cases it can become cumbersome to transfer files to and from a droplet. Imagine a development usage scenario where you are coding apps remotely and find yourself uploading a script repeatedly to your virtual server to test. This can become quite a hassle in a very short period of time. Luckily there is a way to mount your VPS file system to your local computer so you can make changes on the fly and treat your droplet as local storage. In this article, we will show you how to do exactly that.

Installing SSHFS

On Ubuntu/Debian

SSHFS is Linux based software that needs to be installed on your local computer. On Ubuntu and Debian based systems it can be installed through apt-get.

```
sudo apt-get install sshfs
```



README.md

SSHFS-Win · SSHFS for Windows

Install

STABLE V2.7.17334 CHOCO INSTALL SSHFS GUI FRONTEND V1.5.7

SSHFS-Win is a minimal port of SSHFS to Windows. Under the hood it uses Cygwin for the POSIX environment and WinFsp for the FUSE functionality.

How to install

- Install the latest version of WinFsp.
- Install the latest version of SSHFS-Win. Choose the x64 or x86 installer according to your computer's architecture.

How to use

Once you have installed WinFsp and SSHFS-Win you can start an SSHFS session to a remote computer using the following syntax:

```
\\sshfs\[locuser=]user@host[!port][\path]
```

For example, you can map a network drive to billz@linux-host by using the syntax:

```
\\sshfs\billz@linux-host
```

Homebrew Formulae

sshfs

File system client based on SSH File Transfer Protocol

https://osxfuse.github.io/

/api/formula/sshfs.json (JSON API)

Formula code on GitHub

Current versions:

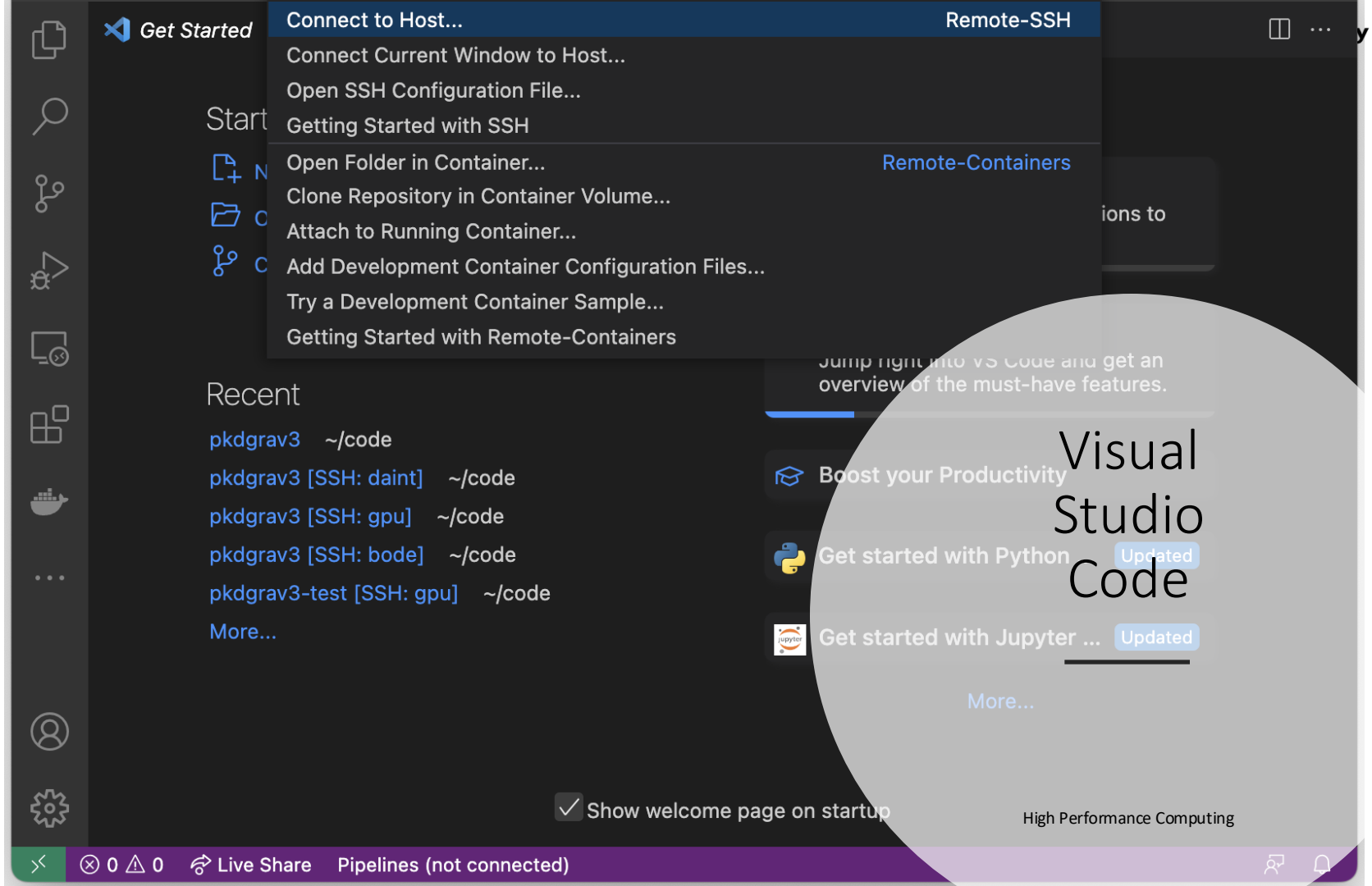
stable	✅	2.10
bottle	🍾	mojave, high_sierra, sierra, el_capitan, yosemite

Depends on:

glib	✅	2.58.3 Core application library for C

Remote Access to files

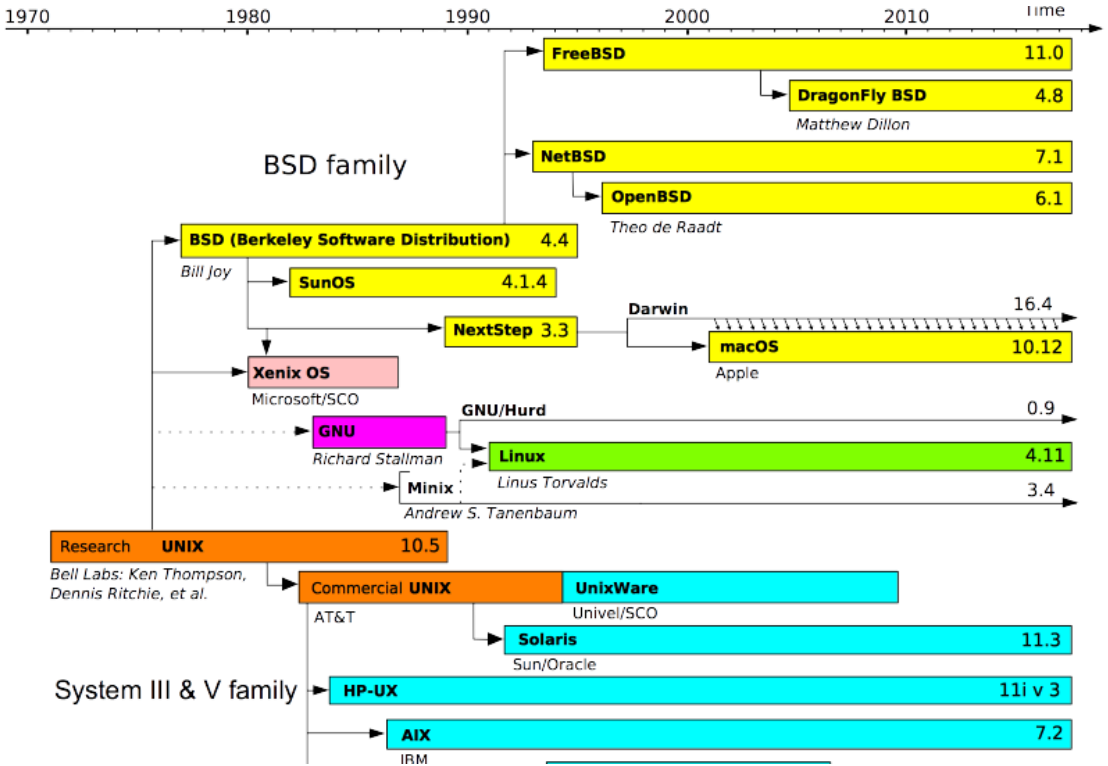

ela
parse.py
ela
master.py
master.c
master.h
{
// The tab key will cycle through the s
// Visit http://wbond.net/sublime_packa
// sftp, ftp or ftps
"type": "sftp",
"sync_down_on_open": true,
"sync_same_age": true,
"host": "ela",
"user": "dpotter",
//"password": "password",
//"port": "22",
"remote_path": "/users/dpotter",
//"file_permissions": "664",
//"dir_permissions": "775",
//"extra_list_connections": 0,
"connect_timeout": 30,
//"keepalive": 120,
//"ftp_passive_mode": true,
fio.c
fio.h
fof.c
fof.h
grav.h
grav2.c
gridinfo.cxx
gridinfo.hpp
group.c
group.h
groupstats.c
groupstats.h
Built into SublimeText
High Performance Computing


Get Started
Connect to Host... Remote-SSH
Connect Current Window to Host...
Open SSH Configuration File...
Getting Started with SSH
Open Folder in Container... Remote-Containers
Clone Repository in Container Volume...
Attach to Running Container...
Add Development Container Configuration Files...
Try a Development Container Sample...
Getting Started with Remote-Containers
Recent
pkdgrav3 ~/code
pkdgrav3 [SSH: daint] ~/code
pkdgrav3 [SSH: gpu] ~/code
pkdgrav3 [SSH: bode] ~/code
pkdgrav3-test [SSH: gpu] ~/code
More...
overview of the must-have features.
Boost your Productivity
Get started with Python Updated
Get started with Jupyter ... Updated
More...
Show welcome page on startup
Live Share
Pipelines (not connected)
Visual Studio Code
High Performance Computing

What is Linux / Unix

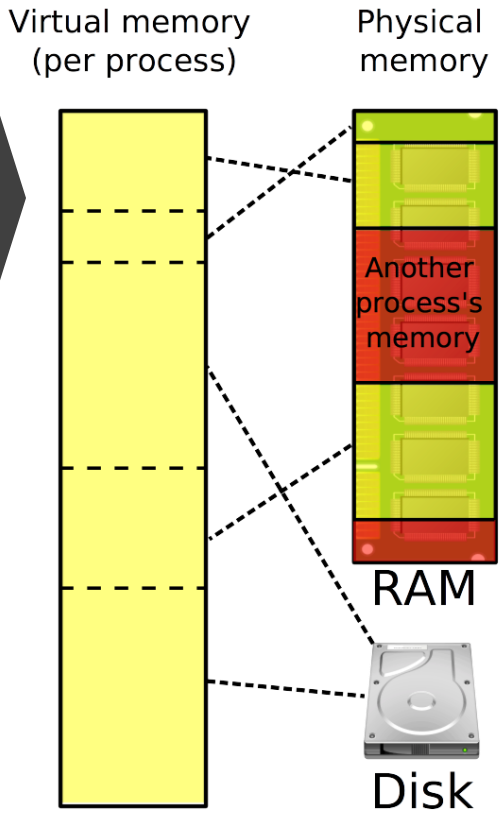
The HPC Operating System

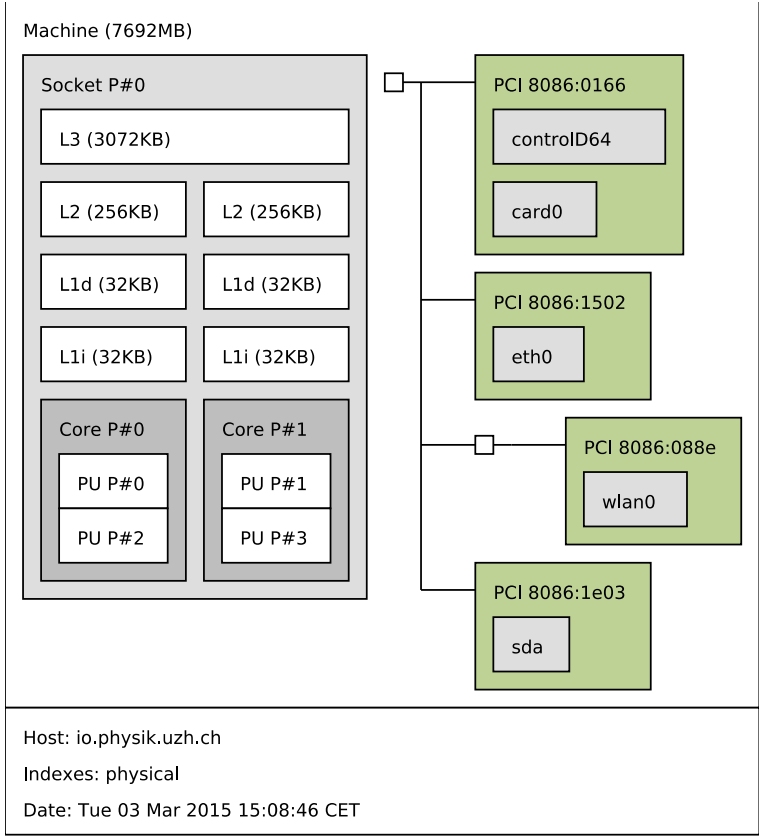
Unix History



Threads and Processes?

- Share memory?
- Share CPU?
- Registers?





“top”

dpotter@daint106:~ ⌥⌘1

```
top - 12:16:36 up 20 days, 4 min, 41 users,  load average: 0.08, 0.26, 0.54
Tasks: 498 total,   2 running, 496 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.9 us,  0.5 sy,  0.0 ni, 98.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 16247788 total,  7473776 free,  2657768 used,  6116244 buff/cache
KiB Swap: 10239996 total, 10091260 free,   148736 used.  8962412 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
17005 bp000253  20   0  181664   5192   1432 S   5.3  0.0   4:14.73 sshd: bp000253@notty
17006 bp000253  20   0  137856   2128   1204 R   2.3  0.0   3:42.73 rsync --server --sender -vlogDtpre.iLs . /sto+
 2194 root      20   0 1065348   3644    788 S   0.7  0.0 174:18.84 /opt/IBM/zimon/sbin/pmsensors -C /opt/IBM/zim+
14541 root       0 -20 5886420   1.5g 245092 S   0.7  9.8   1418:31 /usr/lpp/mmfs/bin/mmfsd
74403 dpotter   20   0  172752   3272   2108 R   0.7  0.0   0:00.19 top -c
74421 root      20   0  112864   4280   3256 S   0.7  0.0   0:00.02 sshd: [accepted]
  870 root      20   0   47732  10352   9948 S   0.3  0.1  54:34.11 /usr/lib/systemd/systemd-journald
 1498 root      16  -4   55520    492    372 S   0.3  0.0   9:21.19 /sbin/auditd
 1568 root      20   0  300824   1928   1236 S   0.3  0.0  40:46.86 /usr/bin/vmtoolsd
 1569 root      20   0   90500    288    284 S   0.3  0.0  31:32.09 /sbin/rngd -f
 2163 root      20   0 2008520  50284   5468 S   0.3  0.3   1199:18 /usr/share/metricbeat/bin/metricbeat -c /etc/+
 2188 root      20   0 3266008 129848   6244 S   0.3  0.8 126:12.44 /usr/bin/python2 /usr/lpp/mmfs/bin/mmsysmon.py
82198 root      20   0  112864   1612    592 S   0.3  0.0   8:09.58 /usr/sbin/sshd -D
93225 astagni   20   0  179760   3260   1444 S   0.3  0.0   2:45.77 sshd: astagni
120562 sobeck   20   0  179416   2656   1212 S   0.3  0.0   0:11.09 sshd: sobeck@pts/36
    1 root      20   0  194616   5816   2536 S   0.0  0.0 101:01.04 /usr/lib/systemd/systemd --switched-root --sy+
    2 root      20   0       0      0      0 S   0.0  0.0   0:04.52 [kthreadd]
    3 root      20   0       0      0      0 S   0.0  0.0   6:34.95 [ksoftirqd/0]
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [kworker/0:0H]
    7 root      rt   0       0      0      0 S   0.0  0.0   0:14.60 [migration/0]
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 [rcu_bh]
    9 root      20   0       0      0      0 S   0.0  0.0  30:06.72 [rcu_sched]
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [lru-add-drain]
   11 root      rt   0       0      0      0 S   0.0  0.0   0:11.75 [watchdog/0]
   12 root      rt   0       0      0      0 S   0.0  0.0   0:10.22 [watchdog/1]
   13 root      rt   0       0      0      0 S   0.0  0.0   0:14.83 [migration/1]
   14 root      20   0       0      0      0 S   0.0  0.0   7:34.07 [ksoftirqd/1]
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [kworker/1:0H]
   17 root      rt   0       0      0      0 S   0.0  0.0   0:10.16 [watchdog/2]
   18 root      rt   0       0      0      0 S   0.0  0.0   0:16.70 [migration/2]
   19 root      20   0       0      0      0 S   0.0  0.0   5:56.64 [ksoftirqd/2]
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [kworker/2:0H]
   22 root      rt   0       0      0      0 S   0.0  0.0   0:10.21 [watchdog/3]
   23 root      rt   0       0      0      0 S   0.0  0.0   0:15.60 [migration/3]
   24 root      20   0       0      0      0 S   0.0  0.0   6:39.99 [ksoftirqd/3]
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [kworker/3:0H]
   27 root      rt   0       0      0      0 S   0.0  0.0   0:10.40 [watchdog/4]
   28 root      rt   0       0      0      0 S   0.0  0.0   0:15.52 [migration/4]
   29 root      20   0       0      0      0 S   0.0  0.0   5:52.68 [ksoftirqd/4]
   31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 [kworker/4:0H]
   32 root      rt   0       0      0      0 S   0.0  0.0   0:10.42 [watchdog/5]
```

What have we learned?

- What is High Performance Computing
- What is a Supercomputer
- Some ways to make workflows faster
- What is a terminal emulator
- How to connect remotely
  - public key encryption and how to use it
  - Connection alternatives (X Windows, sshfs)
- What is the “Linux” operating system
- What is a “program” or “process”