# HPC with Enhanced User Separation

**Andrew Prout**
**MIT Lincoln Laboratory Supercomputing Center**

**S-HPC @ SC24, Atlanta, GA**

**17 Nov 2024**



This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

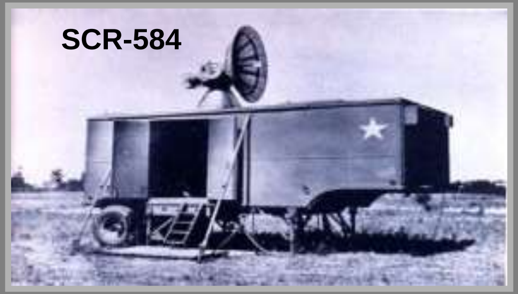
MIT LINCOLN LABORATORY
SUPERCOMPUTING CENTER

# Who We Are – a Little History


MIT Building 20

**Mission:** ***Development of radar systems and technology***

**Main projects:**
- **Surveillance radar**
- **Fire control radar**
- **Navigation systems**



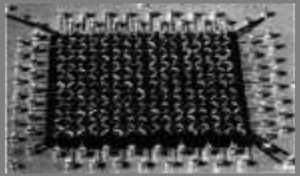

**4000 employees**
**Designed half of all US WWII radars**

**Est. 1951:** ***Air defense and technology development***

**Main projects:** **Semi-Automatic Ground Environment (SAGE)**

**Major Innovations:**


Real-Time Computing


Magnetic-core Memory


Light-pen CRT Interface



https://www.ll.mit.edu/sites/default/files/other/doc/2018-04/MIT_Lincoln_Laboratory_history_book.pdf

# MIT Lincoln Laboratory

## Department of Defense Federally Funded Research and Development Center

Massachusetts Institute of Technology

MIT Lincoln Laboratory

**Mission:** Technology in Support of National Security

**Key Roles:**
- System architecture engineering
- Long-term technology development
- System prototyping and demonstration

**FY23 Employees:** 4419

**FY23 Funding:** $1.39B

**Facilities:** 2.1M sq-ft

**Mission Areas:**
- Air and Missile Defense
- Homeland Protection
- Air Traffic Control
- Communication Systems
- Cyber Security
- Advanced Technology
- Space Control
- ISR Systems and Technology
- Tactical Systems
- Engineering
- Biotechnology & Human Systems

# History of Supercomputing at Lincoln Laboratory

1951 Whirlwind

1953 Magnetic-Core Memory Array

1956 TX-0

1958
- AN/FSQ-7 (Whirlwind II)
- Average Response Computer (ARC)
- CG-24
- TX-2

1962 Lincoln Instrument Computer (LINC)

1963 Sketchpad

1970 Fast Digital Processor (FDP)

1970 GENESYS

1974 Lincoln Digital Voice Terminal (LDVT)

1977 : Lincoln Digital Signal Processor (LDSP)

Late 1970s High-speed FFT pipelined processor

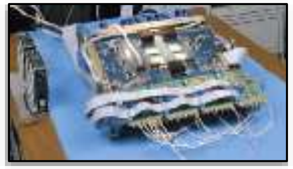
1978 Micro-Processor Based LPC Vocoder (LPCM)

1982 Compact LPC Vocoder

Early 1990s
- APT processor
- RAPTOR processor
- Space-Time Adaptive Processing Library (STAPL)

1992 Radar Surveillance Technology Experimental Radar (RSTER) processor

1999 Parallel Vector Library (PVL)



2002
- ISR Processing and Array Technology (IPAT) processor
- MatlabMPI

2003 pMatlab

2004 Knowledge-Aided Sensor Signal Processing and expert Reasoning (KASSPER) processor

2004
- pMapper gridMatlab
- LLGrid TX-2500

2007
- Parallel Vector Tile Optimizing Library (PVTOL)
- Real-Time Communication Layer (RTCL)

2012 D4M

2014 LLGrid TX-Green

2015 BigDAWG

2016 Lincoln Laboratory Supercomputing Center (LLSC)

1950 — 1960 — 1970 — 1980 — 1990 — 2000 — 2010



https://www.ll.mit.edu/r-d/cyber-security-and-information-sciences/lincoln-laboratory-supercomputing-center

# Lincoln Laboratory Supercomputing Center (LLSC) Role

**Mission Areas**

- Air and Missile Defense
- Homeland Protection
- Air Traffic Control
- Communication Systems
- Cyber Security
- Advanced Technology
- Space Control
- ISR Systems and Technology
- Tactical Systems
- Engineering
- Biotechnology & Human Systems

**Vast Data & Computation**


OSINT | Weather | HUMINT | C2 | Ground | Maritime | Air | Space | Cyber

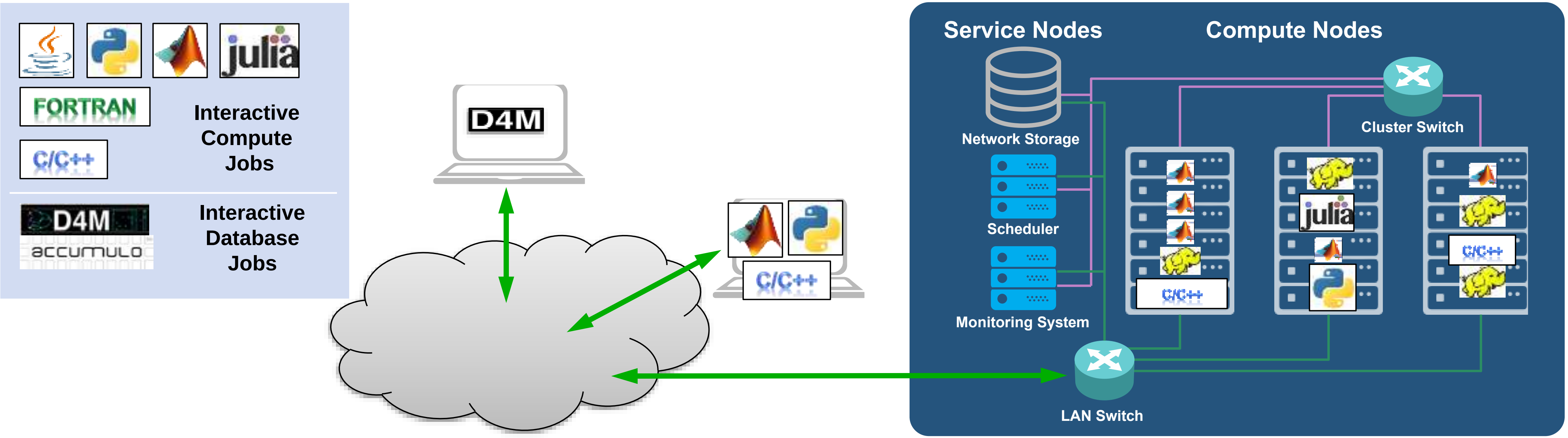
**LLSC develops & deploys unique, energy-efficient supercomputing that provides cross-mission**

- Data centers, hardware, software, user support, and pioneering research
- 100x more productive than standard supercomputing[1]
- 100x more performance than standard cloud[2]



[1]*Interactive Grid Computing at Lincoln Laboratory*, Bliss et al., LL Journal, 2006
[2]*Scalability! But at what COST?*, McSherry et al., HotOS XV, 2015

# Broaden the Definition: Interactive HPC




- LLSC provides a software platform that allows users to
  - Launch interactive compute jobs from their desktop
  - Share large volumes of project data
- The LLSC experience provides
  - Reference datasets pre-positioned in databases
  - Software modules and training to reduce user ramp up time

- Rapid Prototyping
  - Algorithm development
  - Data analysis
  - Machine learning training
- Application Steering
  - Real-time / streaming data analytics
  - Debugging/validation
- Visualization



N. T. Bliss, R. Bond, J. Kepner, H. Kim, and A. Reuther, "Interactive Grid Computing at Lincoln Laboratory," *Lincoln Lab. J.*, vol. 16, no. 1, p. 165, 2006.

# Outline

- **What's different about HPC?**
- **Path Forward**
- **Implementation**
- **Results & Conclusion**

# What's different about HPC?

- Every HPC user is a software developer
  - … but software development is not most users' primary domain of expertise!
  - Very few HPC users have workflows that don't require them to write code
  - This can present in many different ways: writing algorithms in Python/Julia/Matlab/Octave, setting up processing pipelines, performing analysis, creating multi-workflow orchestration via shell scripts, developing a complex distributed simulation using C and MPI …
- Some of the code is early prototype "version 0"
  - It's going to have bugs
  - It's not going to have any security built in (yet)
- Even venerable HPC libraries have little security built-in
  - MPI frameworks do not encrypt data or authenticate peer ranks
  - Efforts to extend them with security have seen little adoption[1,2]



1. *An Empirical Study of Cryptographic Libraries for MPI Communications*, Naser et al., IEEE CLUSTER 2019
2. *Scalable Cryptographic Authentication for High Performance Computing*, Prout et al, IEEE HPEC 2012

# What's different about HPC?

- Users are often required to run both software from large open source frameworks and proprietary closed-source programs
  - Neither are typically designed with a HPC environment in mind
  - Both have unique challenges making them difficult or impossible to modify to better suit this environment
- Software is not the product
  - In many cases the primary goal of running the program is to generate data that will appear in a plot
- Obvious security concerns running this code:
  - It's interacting with sensitive data
  - It's distributed using the HPC network
  - It's on a shared use system

# Outline

- What’s different about HPC?
- Path Forward
- Implementation
- Results & Conclusion

# Path Forward

- **How do we manage this risk?**
- **Option 1: Make the code better**
  - **Focus on improving the most commonly-used software and development libraries and frameworks, providing easy to use security primitives**
  - **Training users to be better, security-focused programmers**

- **Challenges:**
  - **Doesn’t solve issue of where to run “version 0” of code**
  - **Doesn’t address large open source frameworks or closed-source commercial software**
  - **There is a daunting variety of software run on our system, we can’t fork everything**
  - **Still requires users to prioritize writing secure code, and use any primitives provided**
  - **We get new users all the time**

# Path Forward

- How do we manage this risk?
- Option 2: Make the HPC system itself better
  - Every software developer needs a coding sandbox, a safe testbed for the initial development of new code
    - For when you *know* the code still has bugs, including security-relevant bugs
    - Enabling fast exploration of capability
    - Not all coding efforts will turn into successful projects, some are intentional one-offs
    - Even for much more mature code, software designed with HPC in mind rarely fully considers security
  - Core security responsibility cannot be delegated to unprivileged users
    - If everyone is responsible for something, no one is

**Can we make a system where all core security concerns are addressed at the system level?**

# Outline

- What’s different about HPC?
- Path Forward
- Implementation
- Results & Conclusion

# Implementation

- Enhanced separation: Enforcing the separation between users, isolating them so they can't observe or interact with each other
- Several categories of cross-talk that need to be considered:
  - Processes / jobs (local / global)
  - Filesystem (local / shared)
  - Network & web forwarding
  - Accelerators (GPUs, etc)

# Implementation: Local Processes

**Restrict locally visible Linux process information: hidepid=2 on /proc/ mount**

- **Hides processes and command lines belonging to other users or system daemons**
- **Solves entire class of information leakage issues**
  - **Mitigated SLURM CVE-2020-27746 in advance: x11 authentication key exposed on command line**
- **Critical on shared nodes (login, data transfer)**

```
aprout@login-3:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
aprout    253759 3840877  0 15:47 pts/55   00:00:00 ps -ef
aprout   3840832       1  0 14:27 ?        00:00:02 /lib/systemd/systemd --user
aprout   3840877 3840860  0 14:27 pts/55   00:00:00 -bash
aprout@login-3:~$
```

**Better user experience: users only see things they should care about**

# Implementation: Scheduler Jobs (global processes)

- **Restrict globally visible scheduler information: SLURM privatedata configuration**
  - **Hides other users jobs, usage, scheduling and accounting information, etc.**
  - **Shares many of the same information leakage concerns as local processes**
  - **Many job properties could contain private information: name, command, working directory**

```
aprout@login-3:~$ sbatch -p xeon-p8 -a 0-2 --wrap '/bin/sleep 90'
Submitted batch job 27347457
aprout@login-3:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
        27347457_0   xeon-p8     wrap   aprout  R       0:01      1 d-18-8-2
        27347457_1   xeon-p8     wrap   aprout  R       0:01      1 d-18-8-2
        27347457_2   xeon-p8     wrap   aprout  R       0:01      1 d-18-8-2
aprout@login-3:~$
```

```
root@login-3:~# squeue | wc -l
1448
root@login-3:~#
```

**Better user experience: users only see things they should care about**

# Implementation: Filesystem

- **Goal: Users should be unable to share data with any other user**
  - **Except through intentional use of an approved project group**
- **User private groups: the default UNIX group for every user contains only themselves**
- **HPC File Permission Handler[1]: Linux kernel patches to restrict filesystem permissions**
  - **Security mask (smask): Block the use of world bits for unprivileged users**
    - **Similar to "umask 007", but immutable and enforced (even on chmod)**

```
aprout@login-3:~$ touch /tmp/test; chmod o+rwx /tmp/test
aprout@login-3:~$ ls -lah /tmp/test
-rw-rw---- 1 aprout aprout 0 Nov 10 14:56 /tmp/test
```

  - **Restrict file access control lists to group members only**
    - **Cannot grant permission to a group unless you're a member of said group**

```
aprout@login-3:~$ setfacl -m u:areuther:rwx /tmp/test
setfacl: /tmp/test: Operation not permitted
aprout@login-3:~$ setfacl -m g:areuther:rwx /tmp/test
setfacl: /tmp/test: Operation not permitted
aprout@login-3:~$ setfacl -m g:gridteam:rwx /tmp/test
aprout@login-3:~$
```

1. Source code available at: https://github.com/mit-llsc

# Implementation: Network – internal

- **Goal: only permit network connections between processes where client & server are running as the same user**
  - **With ability to extend to project groups on an opt-in basis**
- **No modification of end-user code**
  - **We'd tried providing cryptographic primitives before[1], very little adoption**
    - **Would require a mandate, and a never-ending "policing it" effort**
    - **Not a solution for closed source code**
  - **No ability for user to turn it off**
- **User-Based Firewall[2] (UBF) for TCP & UDP traffic**
  - **IPTables NetFilter Queue module (nfqueue) used to send new connections to userspace daemon for decision**
  - **Only "new" connections are sent, IPTables conntrack handles existing connections**
  - **ident[3]-like query sent to far system to get user information, same query run locally**
  - **Connection allowed if same user, or connector is a member of listener process primary group**
  - **Implicitly controls most IB/RDMA traffic: most frameworks use TCP connection for setup**

1. *Scalable Cryptographic Authentication for High Performance Computing*, Prout et al, IEEE HPEC 2012
2. *Enhancing HPC Security with a User-Based Firewall*, Prout et al, IEEE HPEC 2016. Source code available at: https://github.com/mit-llsc
3. *Identification Protocol*, US Department of Defense, RFC1413 Feb 1993

# Implementation: Network – external

- **Goal: Enable easy access to web-services running as jobs on the HPC cluster from end-user web browsers**
  - **With always-on enforced authentication provided by the system**
  - **Without TLS certificate warnings**
- **Point solutions existed, but require integration effort, increased attack surface, and often used incompatible authentication schemes**
  - **Multi-user solutions: JupyterHub, RStudio, …**
  - **One-offs: TensorBoard, VisualStudio Code Server, …**
- **Web application forwarding via HPC portal[1]**
  - **Allows users to forward access to web applications running as part of jobs**
  - **Avoids ad-hoc port forwarding through SSH, TLS certificate warnings, user misconfigurations**
  - **Authentication required to HPC Portal and UBF connection rules applied**
    - **Supports password-less smartcard-only systems**

1. *Enhancing HPC Security with a User-Based Firewall*, Prout et al, IEEE HPEC 2016. Source code available at: https://github.com/mit-llsc

# Implementation: Accelerators

- GPUs do not use a traditional security model for data resident in memory
  - No concept of data ownership, data segmenting within the GPU[1,2]
- Assign GPUs as a single-user resource
  - Not relevant when whole node scheduling when pam_slurm restrictions are in place
  - Modify permissions on relevant character special files in /dev/ to allow only the user private group of the user allocated that GPU via SLURM
  - GPUs not assigned to the user are not visible at all
- Clear GPU memory before reassignment
  - GPU has no implicit way to know when it’s being reassigned
  - Previous user’s data will remain in GPU memory, registers
  - Vendor-provided steps taken to clean GPU performed in SLURM epilog



1. *CUDA Leaks: Information Leakage in GPU Architectures*, Di Pietro et al, ACM TECS 2013
2. *Confidentiality Issues on a GPU in a Virtualized Environment*, Maurice et al, FC 2014

# Outline

- What’s different about HPC?
- Path Forward
- Implementation
- Results & Conclusion

# Results

- Opportunities for accidental data leakage between users are greatly reduced
  - A few paths still exist: file names in world-writable directories (e.g. /tmp, /var/tmp), abstract namespace unix domain sockets, direct IB verbs communication
- Enhances reliability as well
  - Even if users chose same port number for a network service, they can’t crosstalk and corrupt each others’ data
- Limits the damage of misbehaving code
  - Contains the “blast radius” of any issues to just that user’s account
- The user experience is enhanced because they don’t need to sort through irrelevant information about other users processes/jobs
- Compliance people are happier
  - No more blurring the line between who is responsible for security: it’s a system service

# Conclusion

- Every HPC user is a software developer
- Software development is not their primary domain of expertise, and never will be
- By enabling strong user separation at every point in the system, you protect the confidentiality and integrity of the data
- By reducing the burden on the user to worry about these things, the usability of the system is enhanced as well
- By making security a system-provided service, data owners can have increased confidence about having their data on a multi-tenant system

# Acknowledgements


- LaToya Anderson
- William Arcand
- William Bergeron
- David Bestor
- Alex Bonn
- Daniel Burrill
- Chansup Byun
- Vijay Gadepally
- Michael Houle
- Matthew Hubbell
- Hayden Jananthan
- Michael Jones
- Jeremy Kepner
- Piotr Luszczek
- Peter Michaleas
- Lauren Milechin
- Guillermo Morales
- Julie Mullen
- Albert Reuther
- Antonio Rosa
- Siddharth Samsi
- Jason Williams
- Charles Yee

# Contact

**Source Code: https://github.com/mit-llsc**