HPC workload characterization using eBPF

Shubh Pachchigar, Brandon Cook, Brian Friesen
NERSC

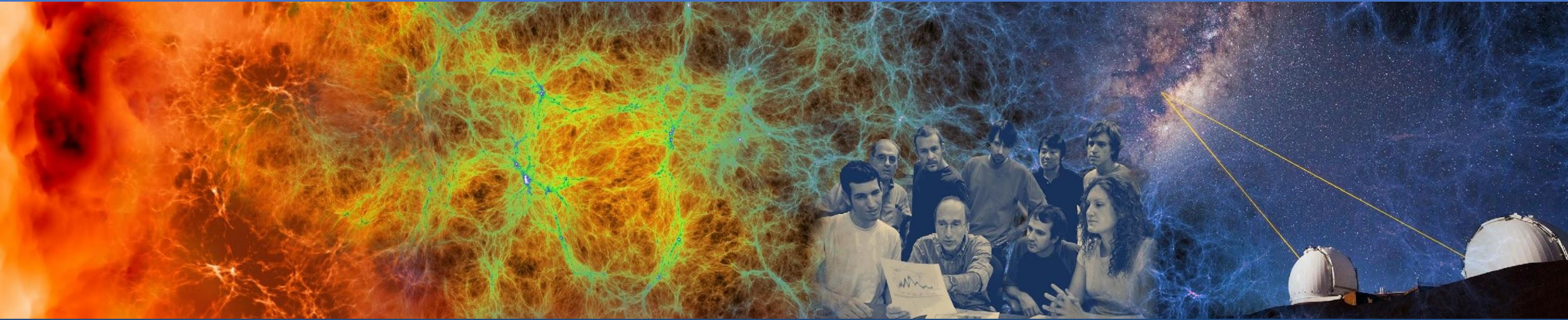
About me

- Graduate Student in Computer Science at San Francisco State University.
- Intern in the Programming Environments and Models (PEM) Group at NERSC since 2024.
- Previously worked with Performance analysis of scientific workflows in containerized workflows.
- Two publications at SC24 in Atlanta, GA.

Agenda

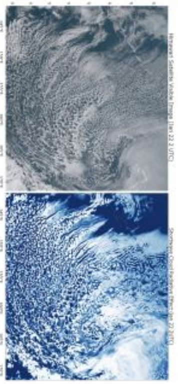
- NERSC workload evolution
- NERSC Filesystems
- Problem Statement
- Development and Testing Setup
- Introduction to eBPF and LDMS
- Software Design
- Overhead Analysis
- Future Plans

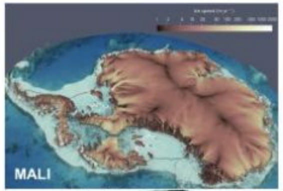
NERSC Workload Evolution

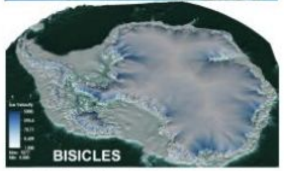


NERSC supports a wide range of workflows

Climate

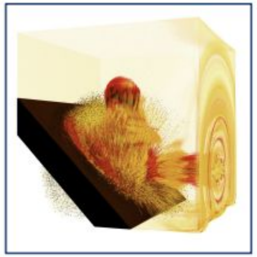


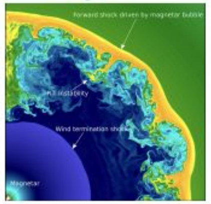




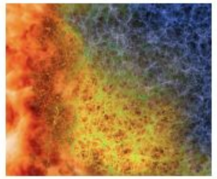
Simulated Antarctic ice sheet 200 years after all floating ice shelves are instantaneously removed.

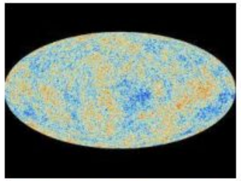
Plasma Physics

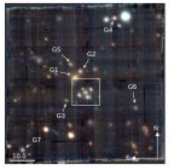


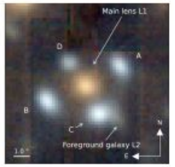


Astrophysics

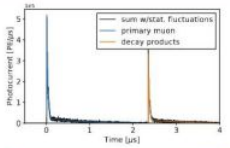


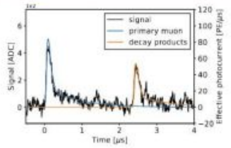


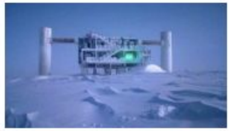


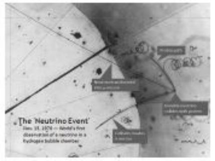


Particle Physics

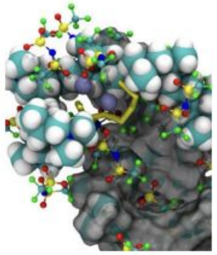


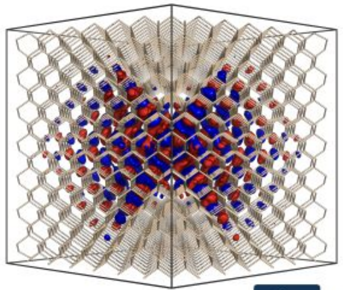




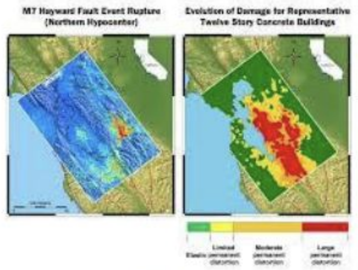


Materials Science





Geology

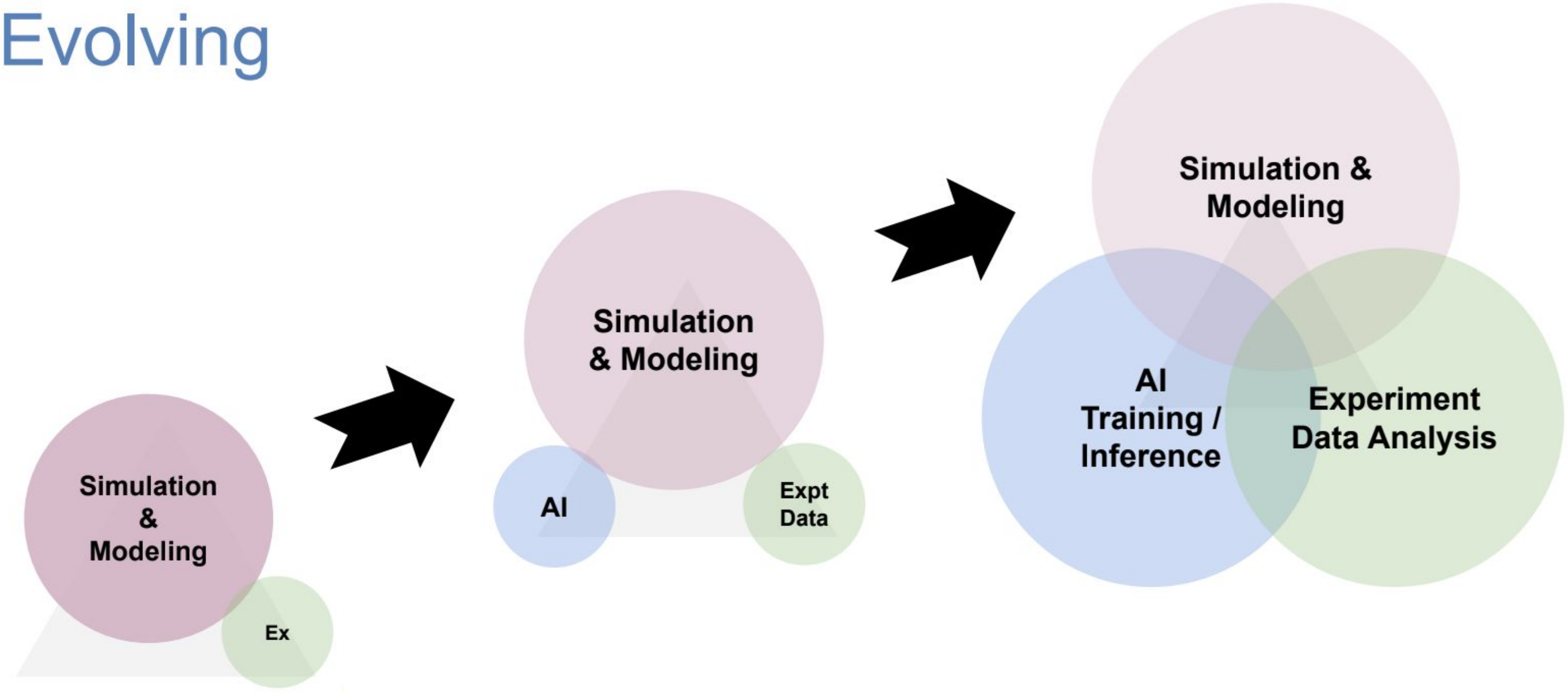

M7 Hayward Fault Event Rupture (Northern Hypocenter)
Evolution of Damage for Representative Twelve Story Concrete Buildings
Limited
Moderate
Large

NERSC

BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

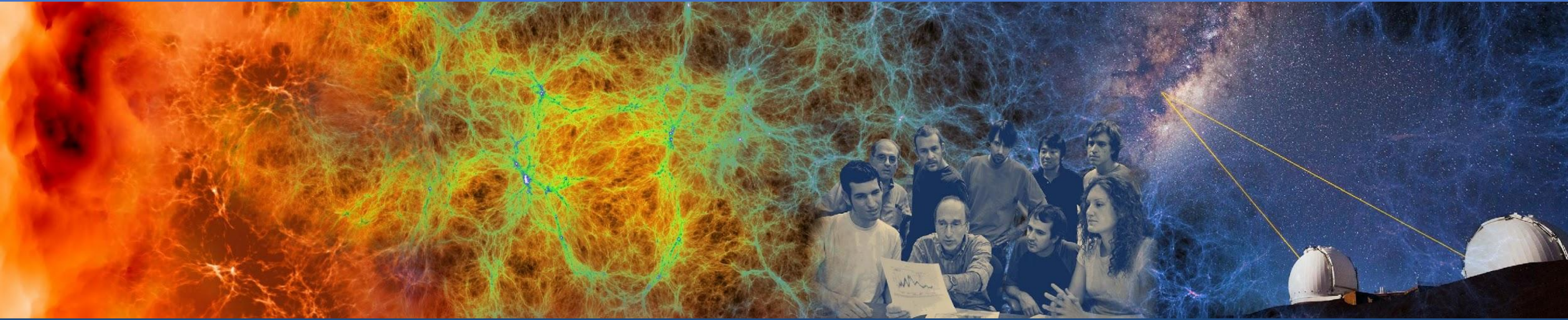
HPC Facility Workload Balance is Evolving



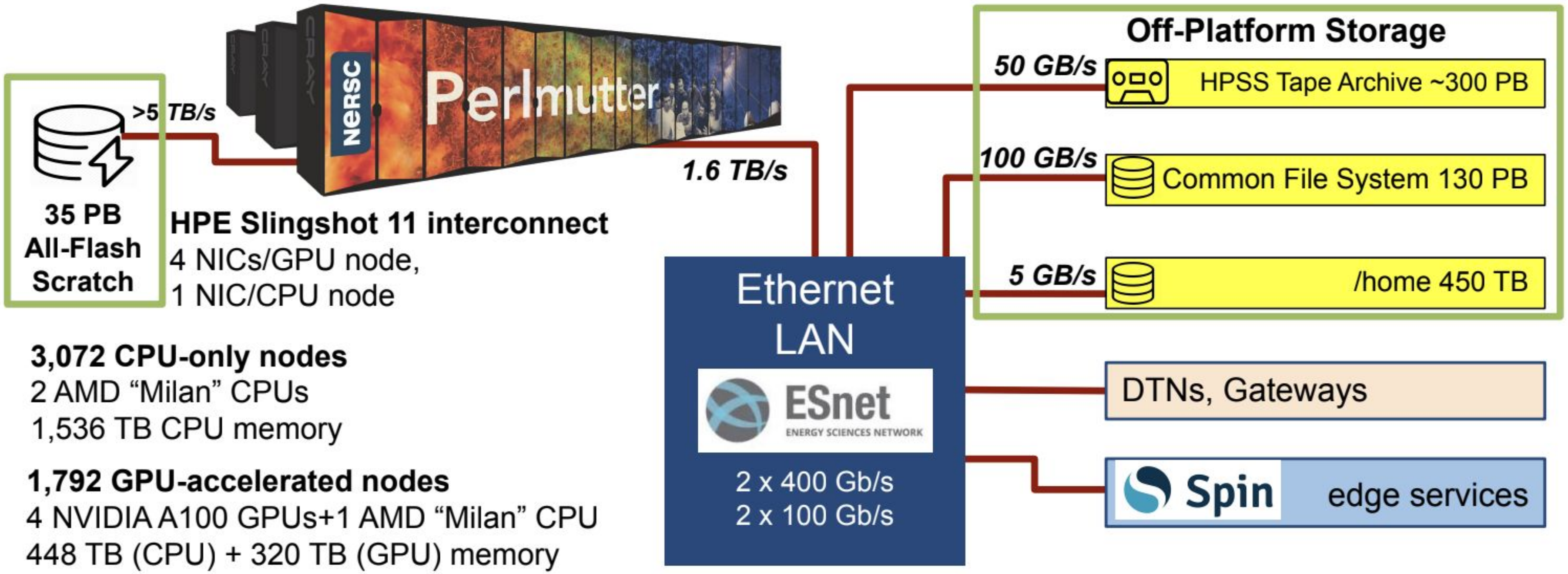
Containers at NERSC

- With rising data analysis and workload pattern evolving with AI, we see an increase in use of containers performing **intense I/O** operation.
- Supported container runtimes at NERSC: podman-HPC, Shifter and Apptainer*
- Although there is an **increase** in use of **container** yet tools are **limited** for **observability**

NERSC Filesystems



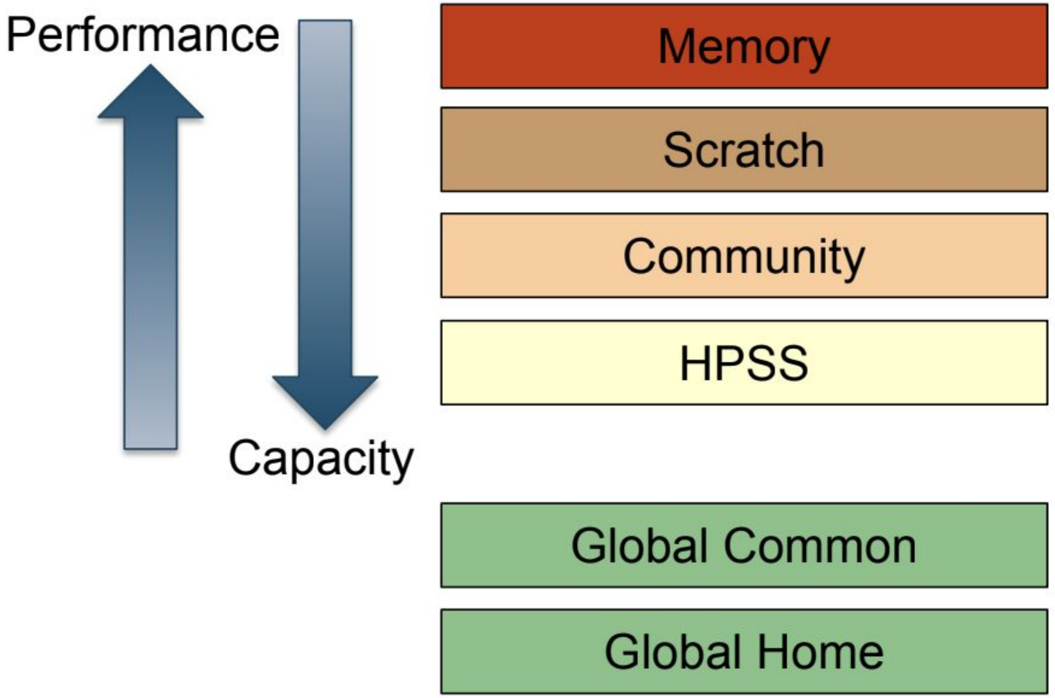
The System is a Sum of Many Parts!



3,072 CPU-only nodes
2 AMD "Milan" CPUs
1,536 TB CPU memory

1,792 GPU-accelerated nodes
4 NVIDIA A100 GPUs+1 AMD "Milan" CPU
448 TB (CPU) + 320 TB (GPU) memory

Simplified NERSC File Systems



36 PB SSD Perlmutter Scratch
Lustre 6 TB/s, temporary (purge)

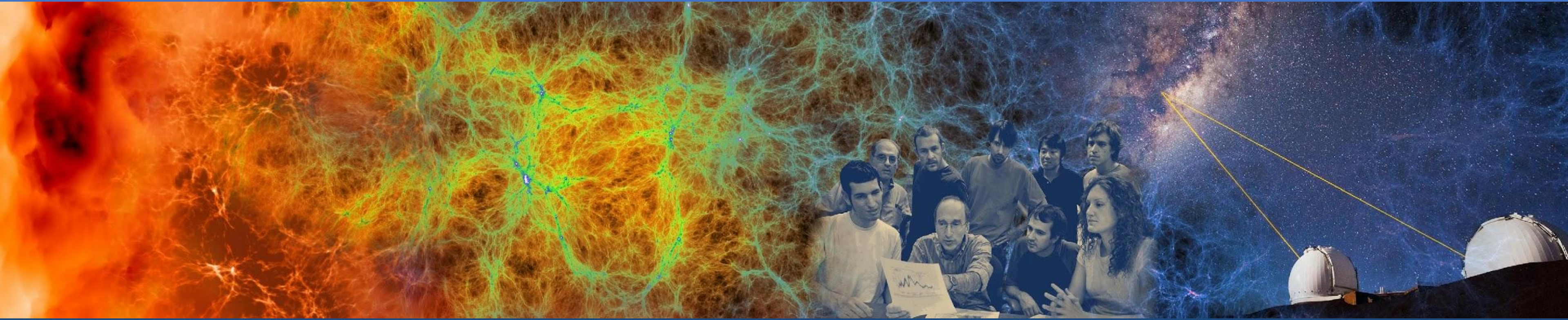
114 PB HDD Community
Spectrum Scale (GPFS)
150 GB/s, permanent

315 PB Tape HPSS Archive
Long Term

20 TB SSD Global Common Software
Spectrum Scale, Permanent
Faster compiling / Source Code

https://docs.nersc.gov/filesystems/

Problem Statement



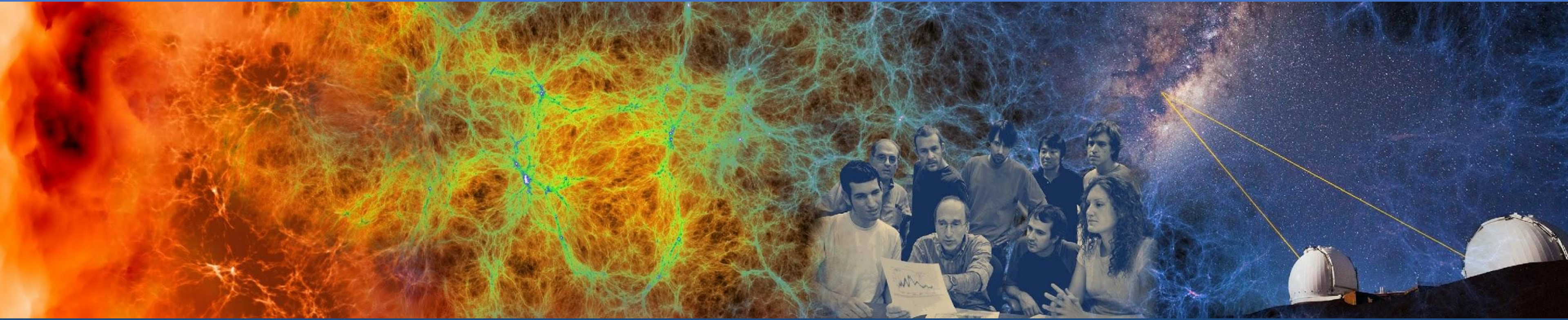
Problem Statement

- The HPC I/O stack is inherently complex.
- Efficient file system interaction are crucial for running containerized workflows at scale.
- Current solutions like Darshan have **no container visibility** yet.
- There's a growing need for deeper insights and more **granular visibility** into I/O behavior to drive meaningful improvements.

Current Practice: Darshan

- Darshan is a mature HPC I/O characterization tool.
- There are concerns with Darshan modifying binary executables and injecting custom code for instrumentation.
- Darshan does not understand the notion of containers or caching mechanisms in filesystems.

Development and Test Setup

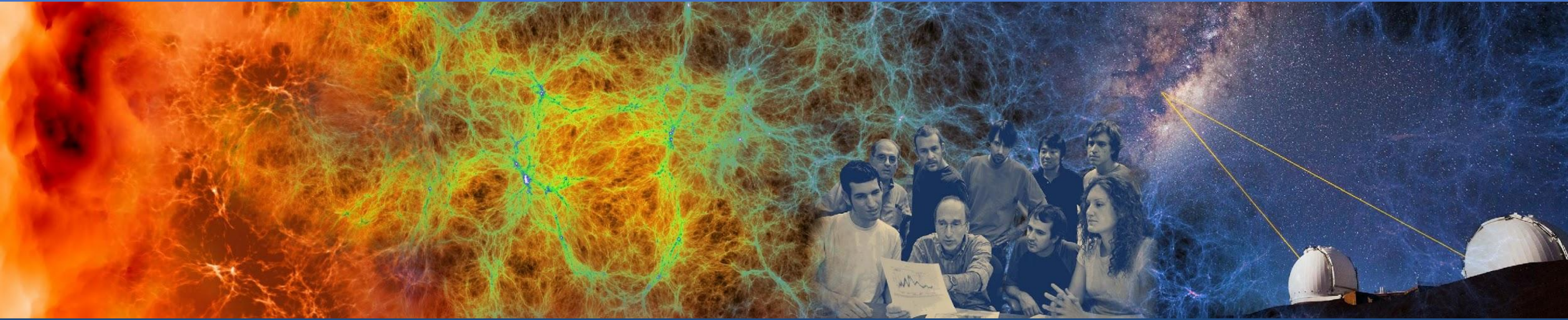


- AWS EC2 instance for Development
  - OS: SUSE Linux Enterprise Server 15 SP6
  - Kernel: 6.4 vanilla
- Perlmutter TDS (Alvarez/Muller) for Testing
  - OS: SUSE Linux Enterprise Server 15 SP5
  - Kernel: HPE fork of 5.14

Small Note about Cray ld errors with BCC

Presence of `libclang` in `/etc/ld.so.cache` due to cray PE optimizations can lead to build errors with eBPF programs. Rebuilding the cache without `cray-pe.conf` should avoid this issue.

Introduction to eBPF and LDMS



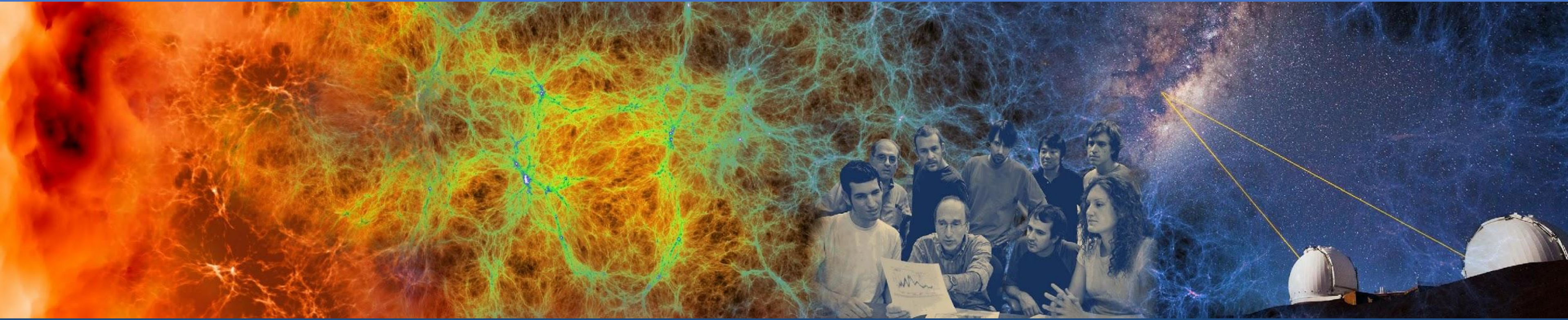
What is eBPF?

- eBPF allows us to run **sandboxed** programs in a privileged context.
- eBPF is used to safely and efficiently **extend** the **capabilities** of the kernel without requiring to change kernel source code.
- eBPF programs are **event-driven** and are run when the kernel or an application passes a certain **hook** point.
- Pre-defined hooks include system calls, function entry/exit, kernel tracepoints or network events

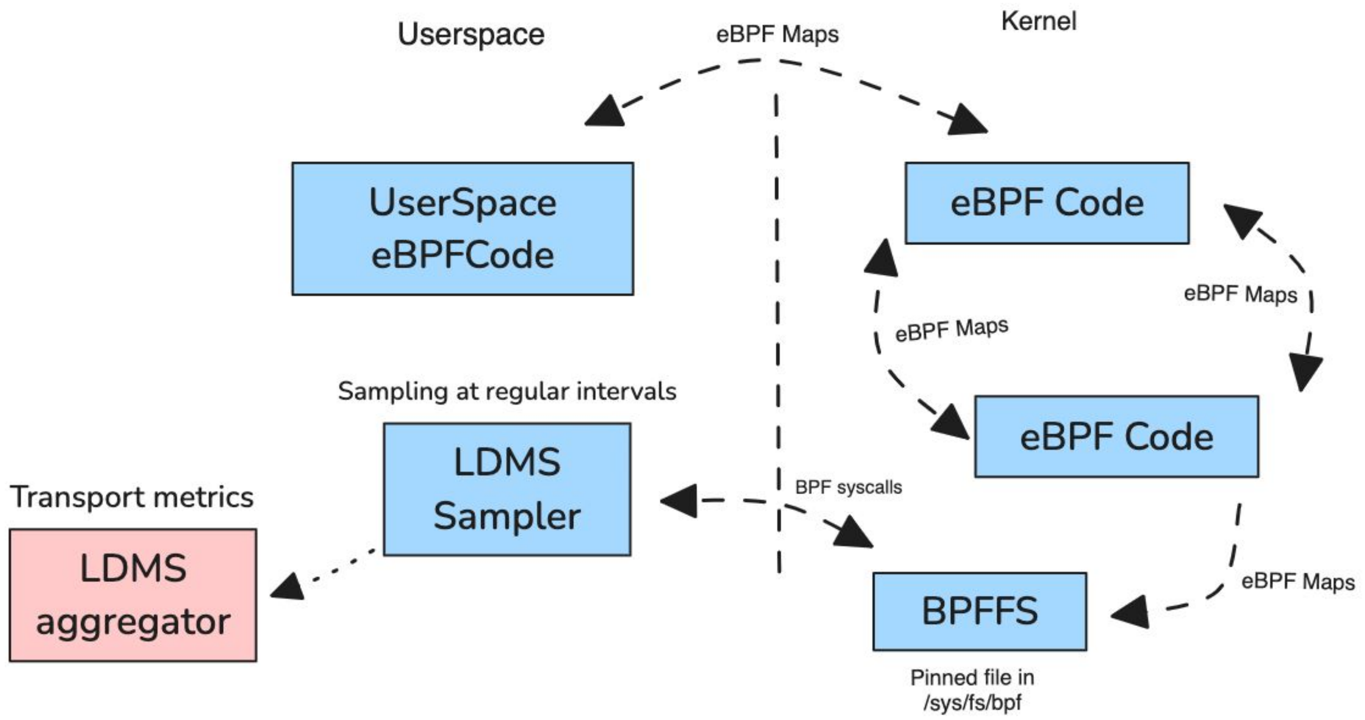
What is LDMS?

- LDMS is a lightweight framework for collecting, aggregating, and transporting system metrics in HPC environments.
- It gathers data from diverse sources like hardware counters and system logs, enabling real-time monitoring with low overhead and scalability for large systems.
- LDMS project started at Sandia Labs with collaborators from LLNL, NERSC and many industry vendors.

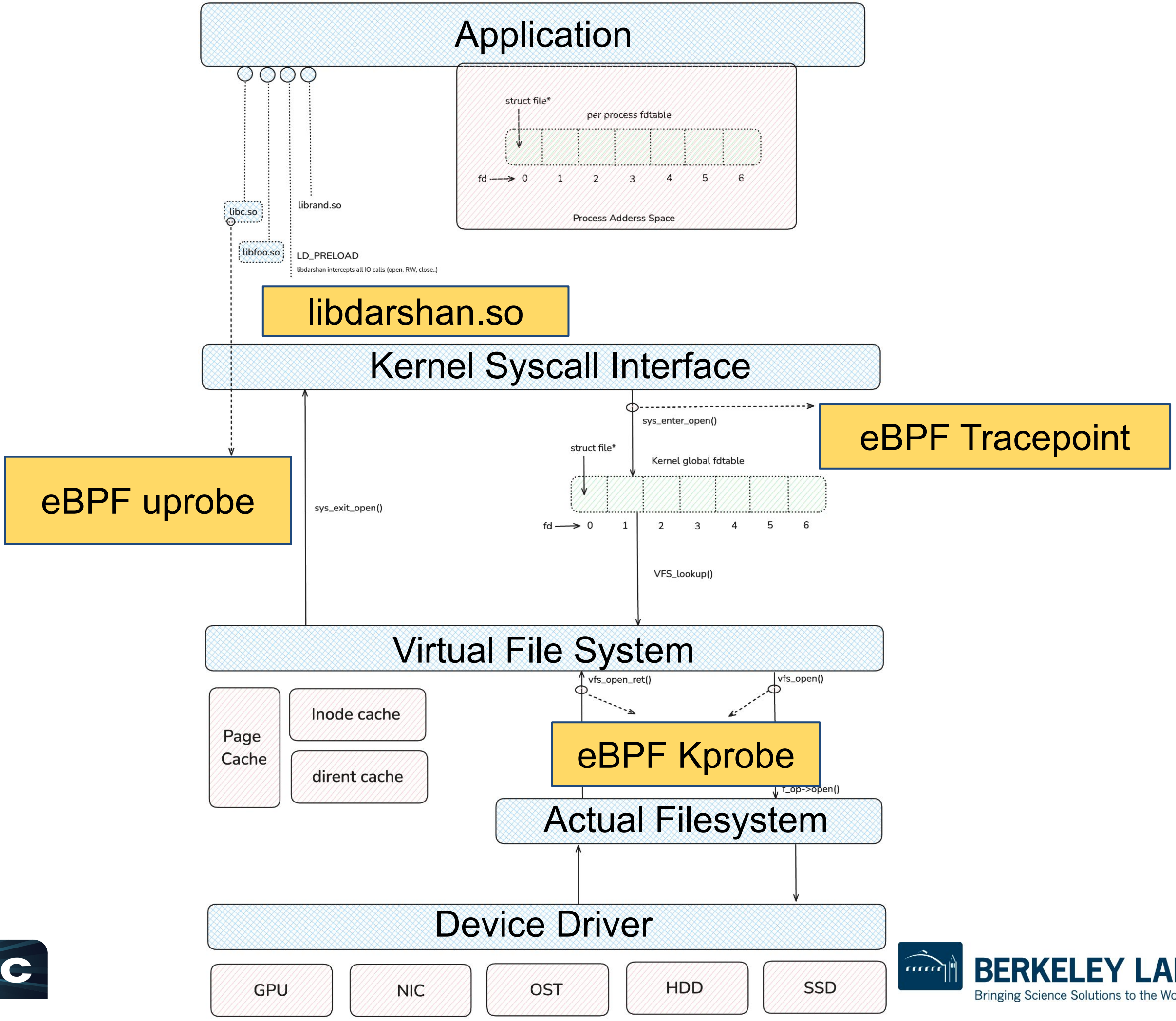
Software Design: Multiple Facets



Overall Architecture



Linux File-I/O Call Path (Simplified)



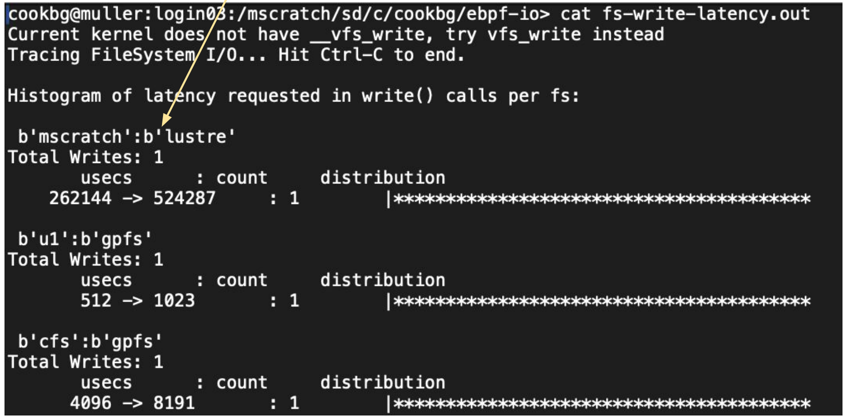
NERSC

BERKELEY LAB
Bringing Science Solutions to the World

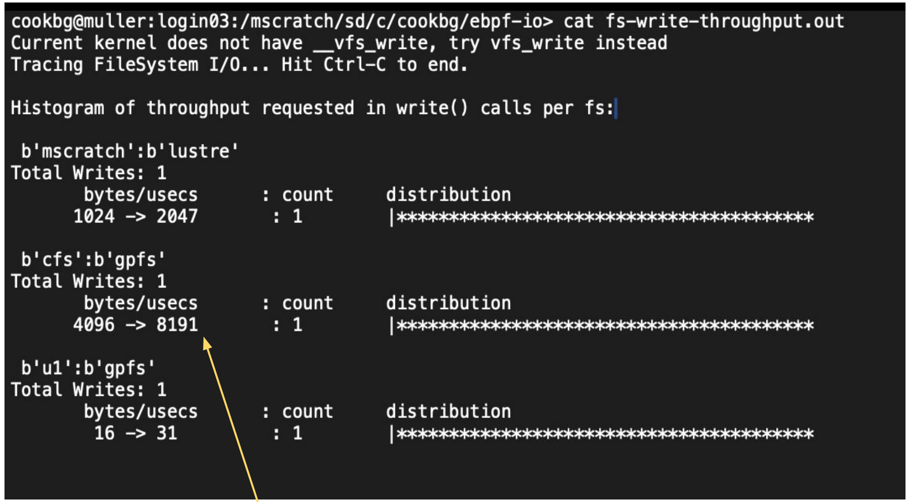
U.S. DEPARTMENT OF ENERGY | Office of Science

Demonstration on Perlmutter TDS

Mount Point: File system type

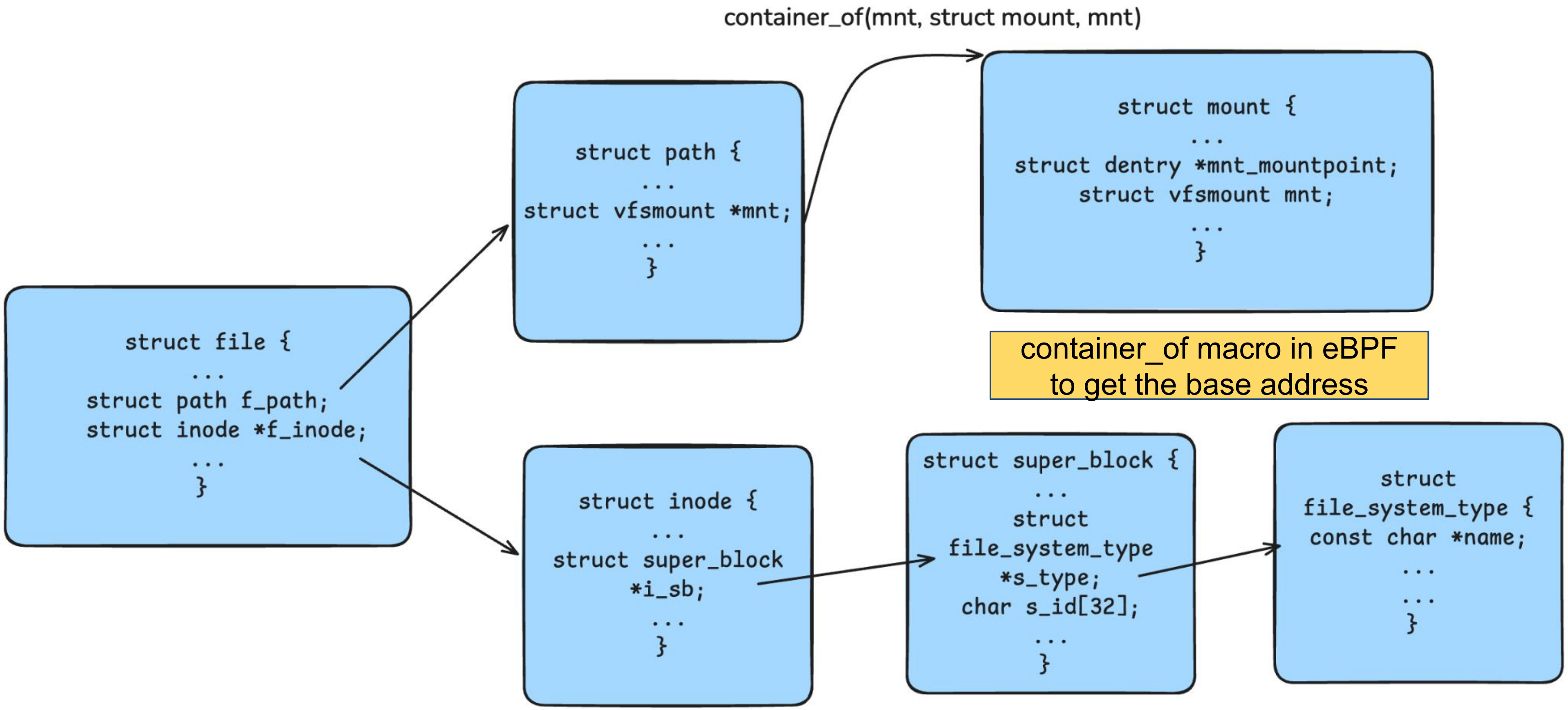


```
cookbg@muller:login03:/mscratch/sd/c/cookbg/ebpf-io> cat fs-write-latency.out
Current kernel does not have __vfs_write, try vfs_write instead
Tracing FileSystem I/O... Hit Ctrl-C to end.

Histogram of latency requested in write() calls per fs:

 b'mscratch':b'lustre'
Total Writes: 1
       usecs        : count     distribution
    262144 -> 524287     : 1        |****************************************

 b'u1':b'gpfs'
Total Writes: 1
       usecs        : count     distribution
       512 -> 1023       : 1        |****************************************

 b'cfs':b'gpfs'
Total Writes: 1
       usecs        : count     distribution
      4096 -> 8191       : 1        |****************************************
```



```
cookbg@muller:login03:/mscratch/sd/c/cookbg/ebpf-io> cat fs-write-throughput.out
Current kernel does not have __vfs_write, try vfs_write instead
Tracing FileSystem I/O... Hit Ctrl-C to end.

Histogram of throughput requested in write() calls per fs:

 b'mscratch':b'lustre'
Total Writes: 1
       bytes/usecs      : count     distribution
      1024 -> 2047        : 1       |****************************************

 b'cfs':b'gpfs'
Total Writes: 1
       bytes/usecs      : count     distribution
      4096 -> 8191        : 1       |****************************************

 b'u1':b'gpfs'
Total Writes: 1
       bytes/usecs      : count     distribution
        16 -> 31          : 1       |****************************************
```

Throughput of /global/cfs for a simple write



```
struct file {
...
struct path f_path;
struct inode *f_inode;
...
}
```

```
struct path {
...
struct vfsmount *mnt;
...
}
```

```
struct mount {
...
struct dentry *mnt_mountpoint;
struct vfsmount mnt;
...
}
```

```
struct inode {
...
struct super_block
*i_sb;
...
}
```

```
struct super_block {
...
struct
file_system_type
*s_type;
char s_id[32];
...
}
```

```
struct
file_system_type {
const char *name;
...
...
}
```

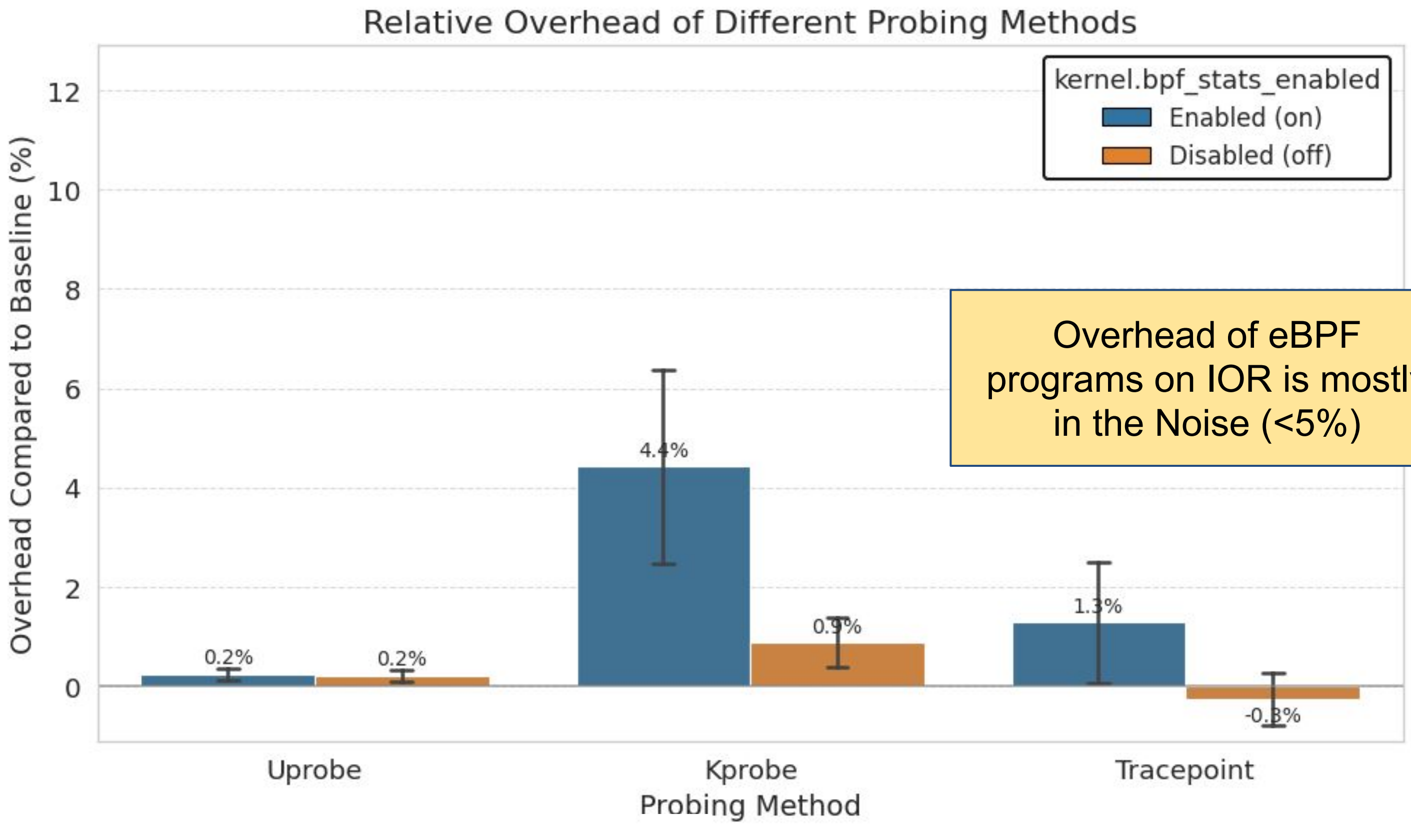
container_of macro in eBPF
to get the base address

24

Many options for extended observability

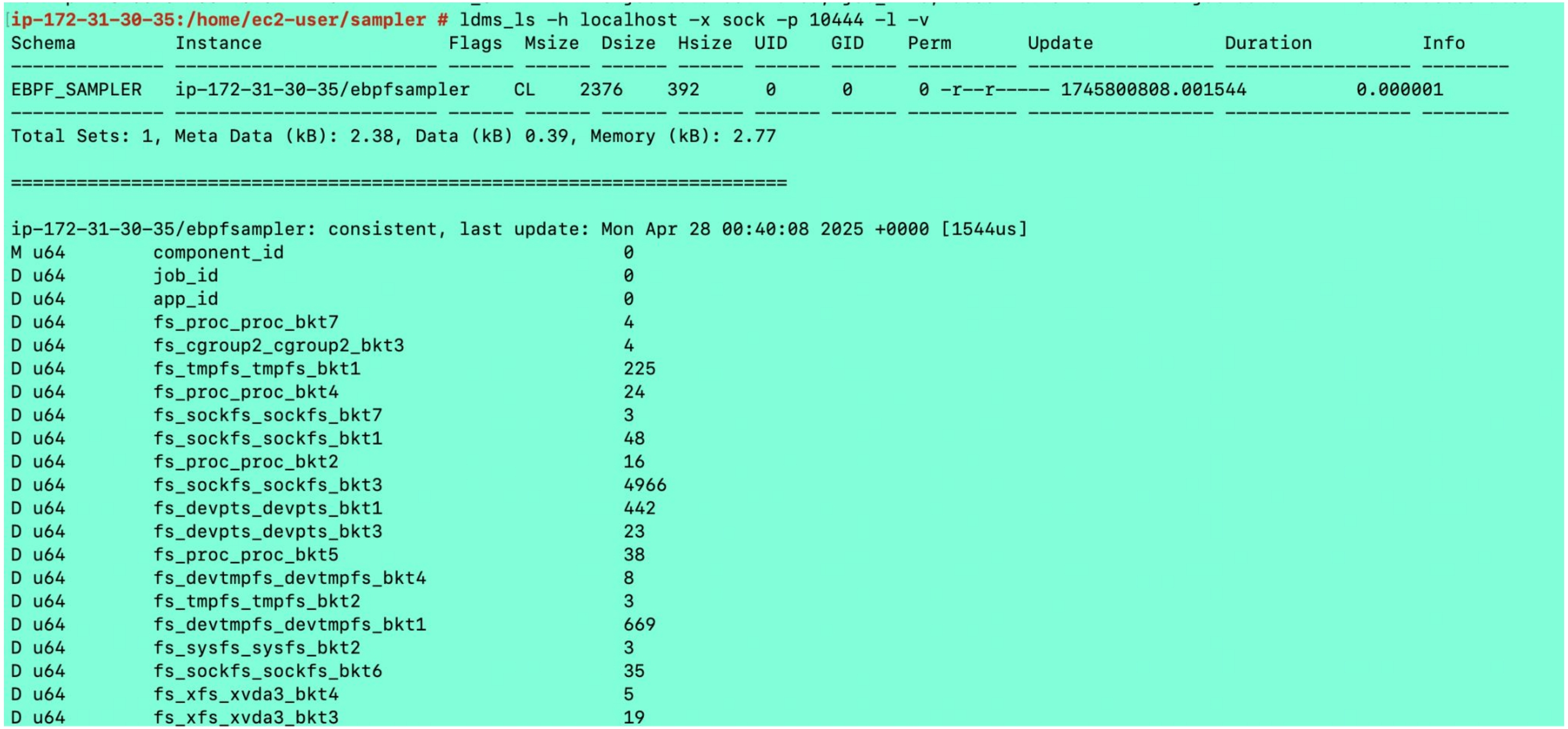
- Read/Write Latency
- Volume
- Read/Write Throughput
- IOPS
- Hook into any shared object symbol
- Hook into any place in the kernel*

Overhead Analysis

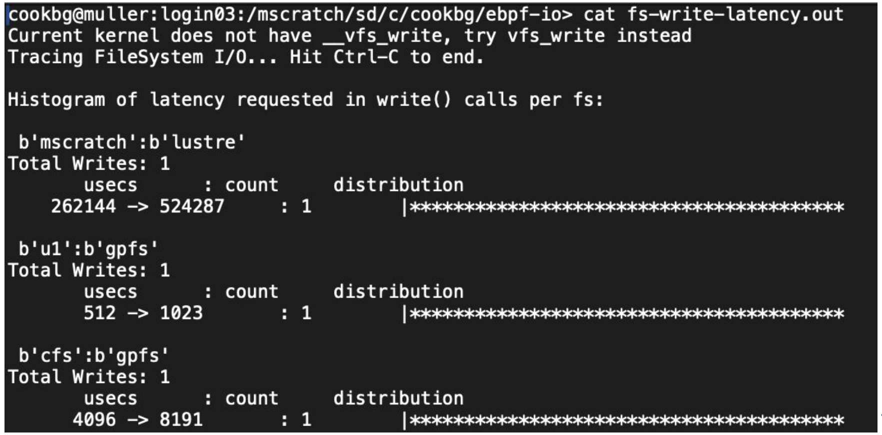


Overhead of eBPF programs on IOR is mostly in the Noise (<5%)

Reading eBPF maps by LDMS

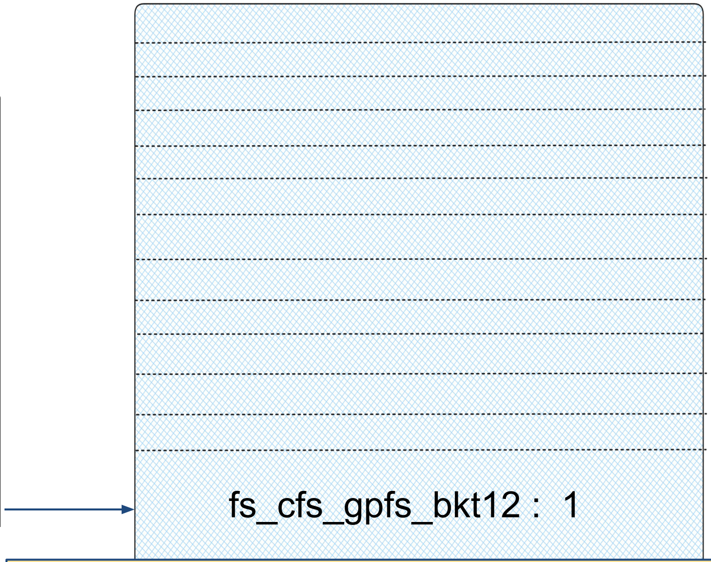


```
ip-172-31-30-35:/home/ec2-user/sampler # ldms_ls -h localhost -x sock -p 10444 -l -v
Schema          Instance                  Flags  Msize  Dsize  Hsize  UID    GID    Perm       Update            Duration          Info
--------------- ------------------------- ------ ------ ------ ------ ------ ------ ---------- ----------------- ----------------- --------
EBPF_SAMPLER    ip-172-31-30-35/ebpfsampler    CL     2376     392       0       0      0 -r--r----- 1745800808.001544          0.000001
--------------- ------------------------- ------ ------ ------ ------ ------ ------ ---------- ----------------- ----------------- --------
Total Sets: 1, Meta Data (kB): 2.38, Data (kB) 0.39, Memory (kB): 2.77

=======================================================================

ip-172-31-30-35/ebpfsampler: consistent, last update: Mon Apr 28 00:40:08 2025 +0000 [1544us]
M u64        component_id                           0
D u64        job_id                                 0
D u64        app_id                                 0
D u64        fs_proc_proc_bkt7                      4
D u64        fs_cgroup2_cgroup2_bkt3                4
D u64        fs_tmpfs_tmpfs_bkt1                    225
D u64        fs_proc_proc_bkt4                      24
D u64        fs_sockfs_sockfs_bkt7                  3
D u64        fs_sockfs_sockfs_bkt1                  48
D u64        fs_proc_proc_bkt2                      16
D u64        fs_sockfs_sockfs_bkt3                  4966
D u64        fs_devpts_devpts_bkt1                  442
D u64        fs_devpts_devpts_bkt3                  23
D u64        fs_proc_proc_bkt5                      38
D u64        fs_devtmpfs_devtmpfs_bkt4              8
D u64        fs_tmpfs_tmpfs_bkt2                    3
D u64        fs_devtmpfs_devtmpfs_bkt1              669
D u64        fs_sysfs_sysfs_bkt2                    3
D u64        fs_sockfs_sockfs_bkt6                  35
D u64        fs_xfs_xvda3_bkt4                      5
D u64        fs_xfs_xvda3_bkt3                      19
```



```
cookbg@muller:login03:/mscratch/sd/c/cookbg/ebpf-io> cat fs-write-latency.out
Current kernel does not have __vfs_write, try vfs_write instead
Tracing FileSystem I/O... Hit Ctrl-C to end.

Histogram of latency requested in write() calls per fs:

 b'mscratch':b'lustre'
Total Writes: 1
       usecs        : count     distribution
    262144 -> 524287     : 1        |****************************************

 b'u1':b'gpfs'
Total Writes: 1
       usecs        : count     distribution
       512 -> 1023       : 1        |****************************************

 b'cfs':b'gpfs'
Total Writes: 1
       usecs        : count     distribution
      4096 -> 8191       : 1        |****************************************
```

LDMS Sampler reading
from eBPF Maps using BPF syscall



Format:
fs_<mnt>_<fstype>_bkt<num>: <cnt>

Future plans

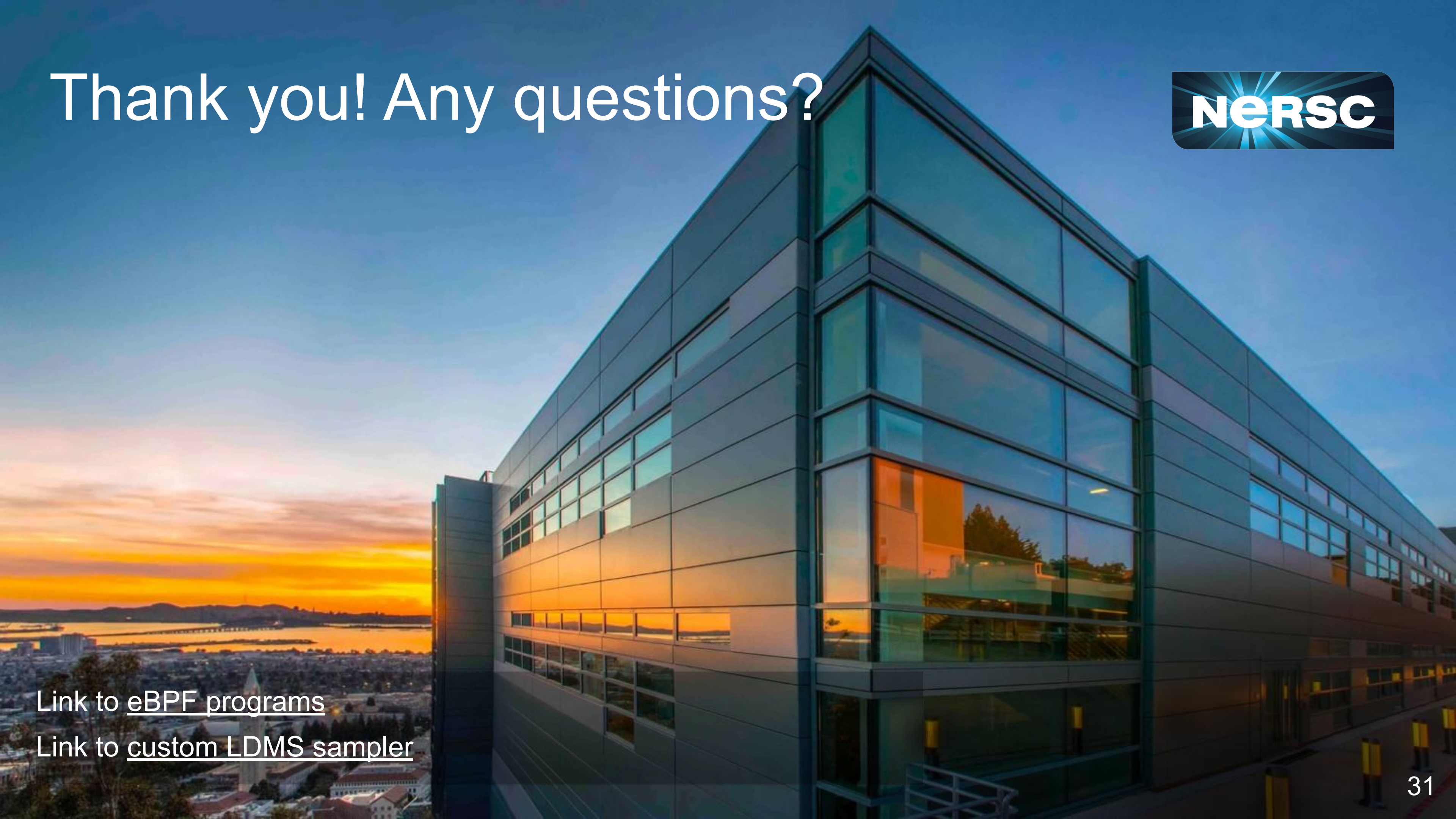
- Testing on real workloads
- Deployment on TDS, then Perlmutter
- BCC to Libbpf conversion
- Users CA, Security review and Scale testing
- Upstream eBPF LDMS sampler code
- Web view for NERSC Users
- Bulk analytics for Staff

Acknowledgment

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DEAC02-05CH11231.

Thanks to Ershaad Basheer, Lisa Gerhardt, Dhruva Kulkarni, Justin Cook and specially to Brandon Cook and Brian Friesen for help with this effort.

Thank you! Any questions?



Link to eBPF programs

Link to custom LDMS sampler