# Throughput Computing and Workflows at NERSC

NERSC Users Group Special Interest Group on Experimental Facilities
June 11, 2020

Bill Arndt
Data Science Engagement Group
warndt@lbl.gov

# Agenda

- Motivation and Scope
- Resources Available at NERSC
- Throughput Computing Challenges
- How to Throughput Work at NERSC

# Why workflows are hard on HPC

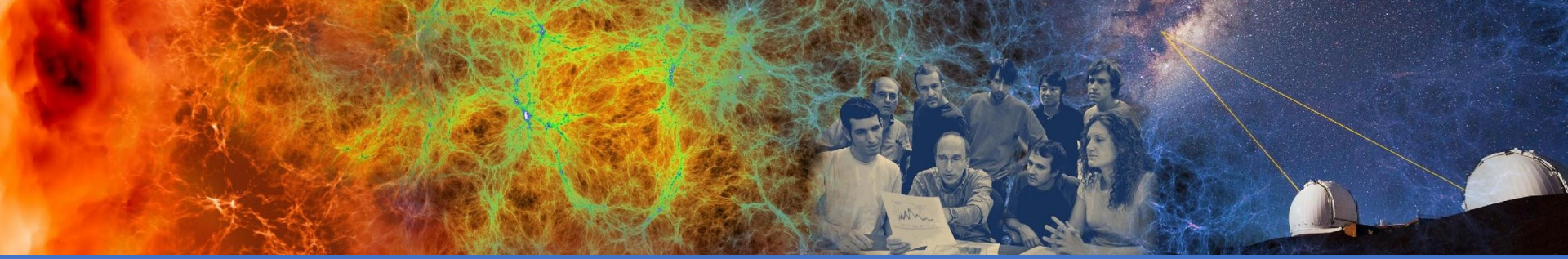
- Not part of HPC culture
- No consensus definition of what “workflow” means
- Choice overload of tools
  - Diversity among use cases and infrastructure
  - More than 300 choices, none of them feel right so I’ll make my own...
  - ...now more than 301 choices
  - Choosing the wrong tools can be disasterous
- Users often neglect to anticipate or plan for it

# What are workflows?

- A *workflow* is a problem best solved by inserting automation between user action and interfaces* to computation and data resources**.
  - *Interfaces like: Slurm commands, shell on a login node, HSI, Globus, IRIS, NEWT
  - **Resources like: Cori compute nodes, storage, network bandwidth and data transfer, identity management
- *Workflow Management Tools* (WMT) are the software systems that perform that automation.

# Some Generalized Examples

- Run one application thousands of times
- Chain together several different applications
- Application has a 2% chance of crashing and needing rerun
- Rerun this application every month



# Resources Available at NERSC for Workflows

# What NERSC is doing to support workflows

- Specialized infrastructure, software, and support
- Workflows Working Group
  - Formed September 2019 - Laurie Stephie (DAS), Bjoern Enders (DSEG), Bill Arndt (DSEG)
  - Thourough evaluation of many WMT ongoing
  - Documentation and guidance refresh
  - Outreach to users, facilities, tool developers, and infrastructure providers

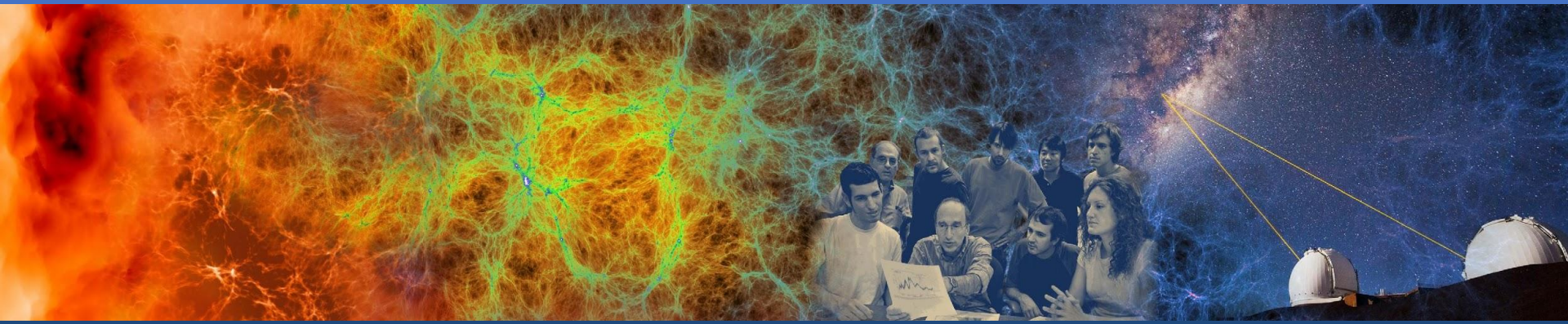
# Cori Workflow Nodes

- Cori has two service nodes specifically reserved for WMTs
  - Same environment as login nodes
  - Access is limited to approved users
  - Heavy compute not allowed
  - The preferred place for crontabs
  - Uptime same as Cori login nodes, prepare accordingly
- Gain access by submitting a request to NERSC support
  - Be prepared to describe your WMT and its resource footprint
  - Provide a list of users who need access to set up and maintain the WMT

# WMT Documentation and Guidance

- https://docs.nersc.gov/jobs/workflow-tools/
  - A work in progress; expanding and refining as our tool evaluation continues
  - Detailed information, examples, pitfalls, and suggestions regarding specific tools and use cases
- We *want* to get tickets about workflow management tools
  - Builds our experience and knowledge of what users need
  - Opprotunity to share that experience

# Throughput Computing Challenges



# Throughput Constraints: Slurm Performance

- The Slurm controller process and its database are a common bottleneck.
- Most Slurm commands incur some load or a database lock:
  - `sbatch`, `salloc`, `sinfo`, `scontrol`, `squeue`, `sqs`, `srun`
- Overloading Slurm degrades Cori for all users.
- Avoid issuing more than one Slurm command per second.
  - Beware of WMTs that do this under the hood

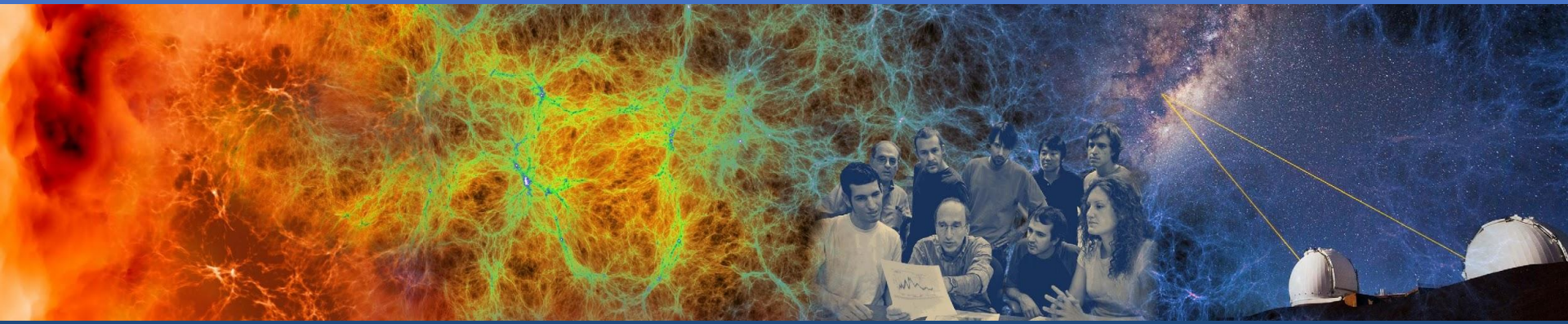
# Throughput Constraints: Queue Policy

- `MaxJobAccrue: 2`
  - Each user gets 2 jobs gaining priority.
  - Favors fewer jobs each requesting more resources and discourages many smaller jobs.
- Big rewards for packing many small tasks into fewer jobs requesting more nodes
- Don’t use Slurm task arrays

# Throughput Constraints: Filesystem Design

- HPC filesystems are designed to deliver maximum bandwidth to full-system sized jobs.
- High throughput workloads tend to use less total I/O bandwidth but many more operations.
- Some WMTs use filesystem locks or mmap commands that aren't available on all NERSC filesystems.
- Common source of scaling bottlenecks

# How To Throughput Work at NERSC



# `srun` Can be Used for Throughput

```
elvis@cori04:~/work> cat srun_tasks.sh
#/bin/bash
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -C haswell
srun -n 64 -c 2 payload.sh

elvis@cori04:~/work> cat payload.sh
#!/bin/bash
echo $SLURM_PROCID
```

- Runs 64 tasks distributed over 2 nodes, 2 threads and 1 core each
- Use `$SLURM_PROCID` inside each task to uniquely distinguish its execution
- `--multi-prog` flag plus its config file can be used to make different task shapes in a single `srun`
- background `srun` is unreliable

# `srun` Tasks with Errors or Finishing Early

- If any task in an srun exits non-zero, every task in the srun is killed
  - add `--kill-on-bad-exit=0` to change that behavior
- If a node in the job fails, the entire allocation will be killed
  - add `--no-kill` to `sbatch` and `srun` to change this behvaior
- `srun --wait` flag controls if and when running tasks are killed if any of them finish before others
  - Our configuration by default applies `--wait=0` (wait for all tasks)

# GNU Parallel is Better than Shared QOS

```
elvis@cori07:~> seq 1 5 | parallel -j 2 'echo \
> "Hello world {}!"; sleep 10; date'
Hello world 1!
Thu Jun 11 00:21:00 PDT 2020
Hello world 2!
Thu Jun 11 00:21:00 PDT 2020
Hello world 3!
Thu Jun 11 00:21:10 PDT 2020
Hello world 4!
Thu Jun 11 00:21:10 PDT 2020
Hello world 5!
Thu Jun 11 00:21:20 PDT 2020
elvis@cori07:~>
```

- `module load parallel`
- Lots of advantages over `srun`
  - Run combinations of tasks in parallel and sequence
  - Easier input substitution
    - If you need it, *much* more power is available
  - No risk of Slurm overload
  - Packed jobs have massively reduced total queue wait
- ...but it only works on one node...

# Why not both?

```
elvis@cori04:~/work> cat srun_tasks_n.sh
#/bin/bash
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -C haswell
#SBATCH --ntasks-per-node 1
srun -K 0 -k -n 2 payload.sh $1

elvis@cori04:~/work> cat payload.sh
#!/bin/bash
cat $1 |                                \
awk -v NNODE="$SLURM_NNODES"            \
-v NODEID="$SLURM_NODEID"               \
'NR % NNODE == NODEID' |                \
parallel task.sh {}

elvis@cori04:~/work> sbatch srun_tasks_n.sh \
list_of_tasks_input.txt
Submitted batch job 2053142
```

- Use `srun` to run `parallel` on each node
- Arguments pass through the task input list
  - `awk` can round-robin distribute tasks to each node
- This pattern scales to use all of Cori
  - Look out for new bottlenecks of course

# “Two percent of my tasks will fail.”

```
elvis@cori04:~/work> cat payload.sh
#!/bin/bash
cat $1 |                                \
awk -v NNODE="$SLURM_NNODES"            \
-v NODEID="$SLURM_NODEID"               \
'NR % NNODE == NODEID' | parallel       \
--joblog logfile_$SLURM_NODEID.txt      \
task.sh {}

cat $1 |                                \
awk -v NNODE="$SLURM_NNODES"            \
-v NODEID="$SLURM_NODEID"               \
'NR % NNODE == NODEID' | parallel       \
--joblog logfile_${SLURM_NODEID}.txt    \
--resume-failed task.sh {}
```

- `--joblog` and `--resume-failed` can be used to track and rerun tasks with non-zero exit codes
- Don’t use `--retries`, it doesn’t do what it should
- Job log files ***must not*** be shared by multiple concurrently running instances of `parallel`

# Staggering Task starts

```
elvis@cori04:~/work> cat payload.sh
#!/bin/bash
sleep $((5*${SLURM_NODEID}))
cat $1 |                                    \
awk -v NNODE="$SLURM_NNODES"                \
-v NODEID="$SLURM_NODEID"                   \
'NR % NNODE == NODEID' | parallel           \
--delay 30 task.sh {}
```

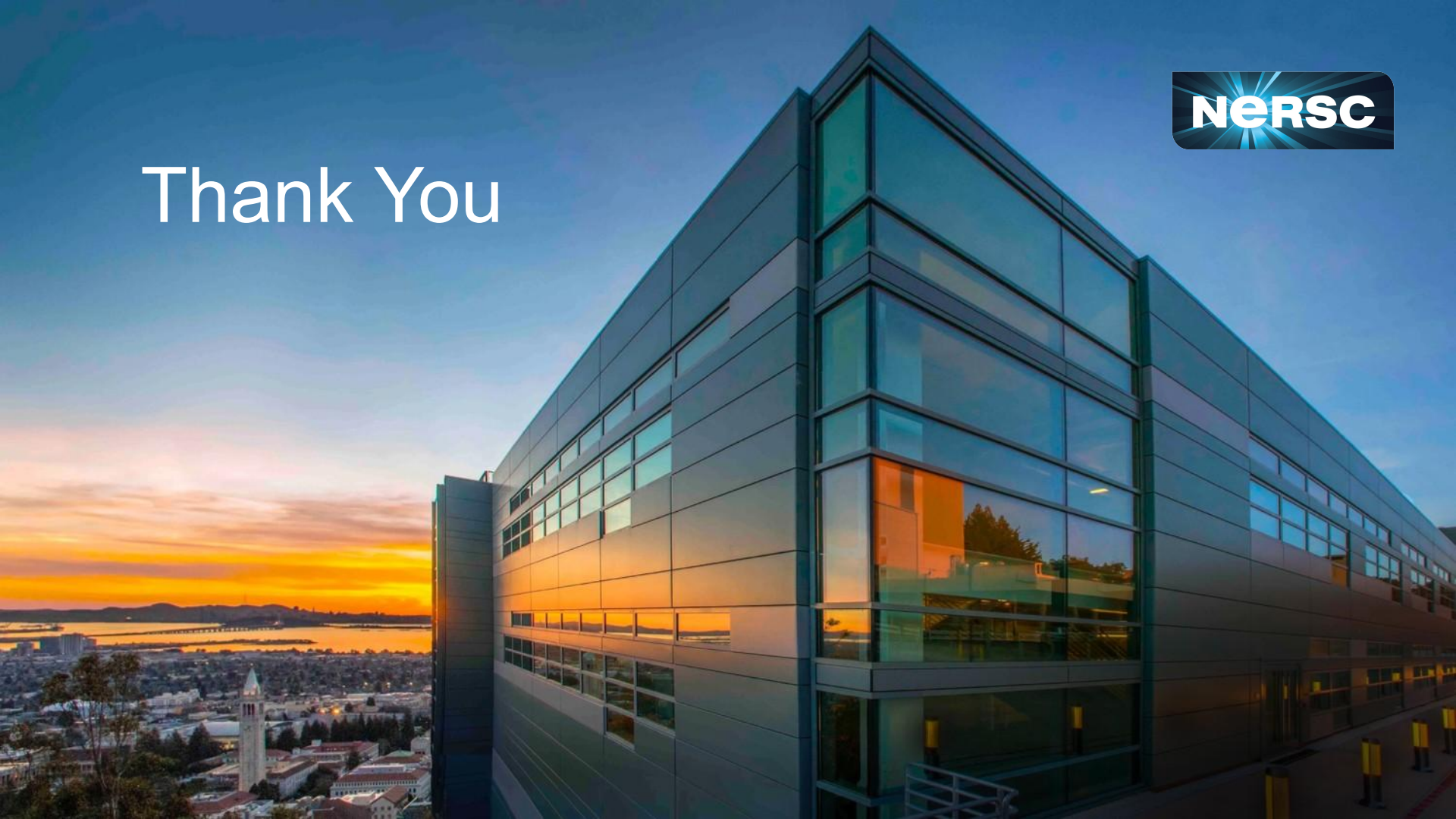
- Use to protect services or filesystems from being overwhelmed
  - use `sleep` to slow the rate that `parallel` commands begin
  - `--delay` flag limits the rate that each `parallel` issues new tasks

# Burst Buffer and Throughput Computing

- The Burst Buffer has excellent I/O operations capacity
  - Up to hundreds of metadata servers on Burst Buffer vs. two for Cori scratch
  - Nessecary to scale an I/O intensive HTC workload to hundreds of compute nodes or beyond

# Data Centric Workflow Management Tools

- “I have many different applications and data types chained together in a network of dependencies.”
- *Plenty* of options. Snakemake and Parsl are two good choices, among *many*
  - Documentation coming soon
- Pitfalls:
  - Many expect cloud responsiveness and can’t handle queue waiting or policies
  - Often lack job packing
  - Naive Slurm integration can use too many requests


NERSC
Thank You