Deploying Checkpoint/Restart for Production Workloads at NERSC

Zhengji Zhao[1], Rebecca Hartman-Baker[1], and Gene Cooperman[2]

[1]NERSC at Lawrence Berkeley National Laboratory

[2]Northeastern University

State of the Practice Talks at SC20

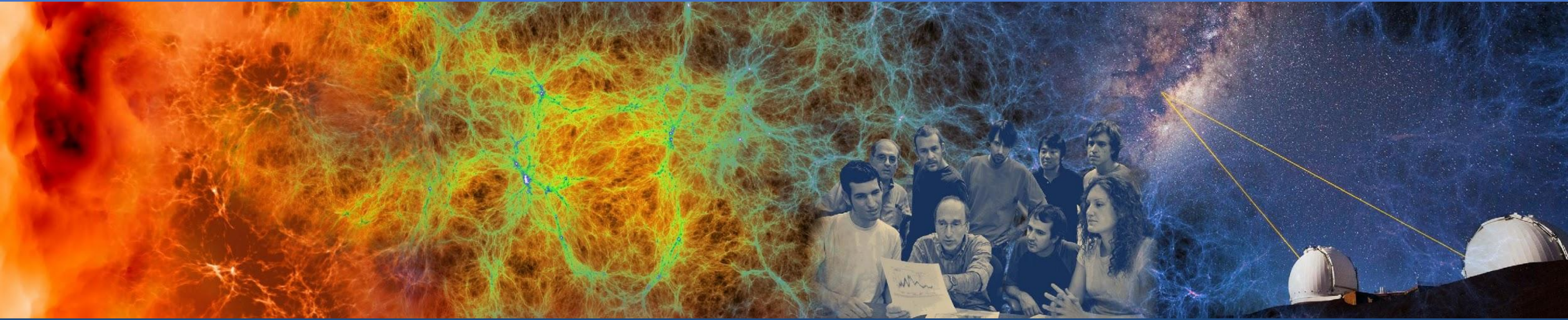
November 17 , 2020

Acknowledgements:
Rohan Garg, Twinkle Jain, and Prashant Chouhan at Northeastern University, Harsh Khetawat at North Carolina State University, and Tiffany Connors, Stephen Leak, and Christopher Samuel at NERSC.

Outline

- Introduction and Motivation for this Work
- DMTCP and MANA
- Enabling MANA/DMTCP for NERSC Workloads
- Promoting C/R Uptake by NERSC Users
- C/R Use Cases
- Summary and Future Work

Introduction and Motivation for this Work

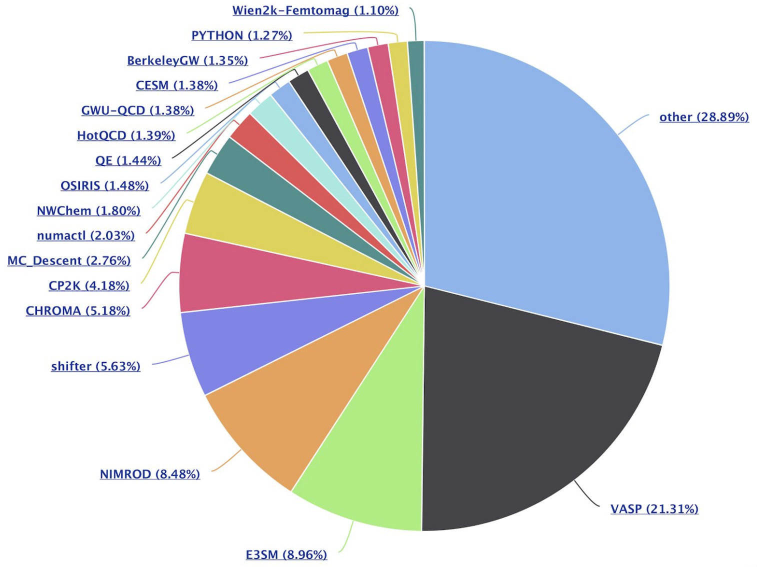


About NERSC

- Primary HPC Center for US DOE Office of Science
    - ~8,000 users, 900 projects, ~2500 papers/year acknowledging NERSC
- NERSC HPC systems:
    - **Cori**, ~30 PF Cray XC40 with 9600 Intel Xeon Phi KNL nodes & 2000 Intel Xeon Haswell nodes, Aries Dragonfly network, 2 PB Burst Buffer & 28 PB scratch file system
    - **Perlmutter**, arriving at end of 2020, ~150 PF Cray Cascade system with hybrid AMD CPU/NVIDIA GPU nodes

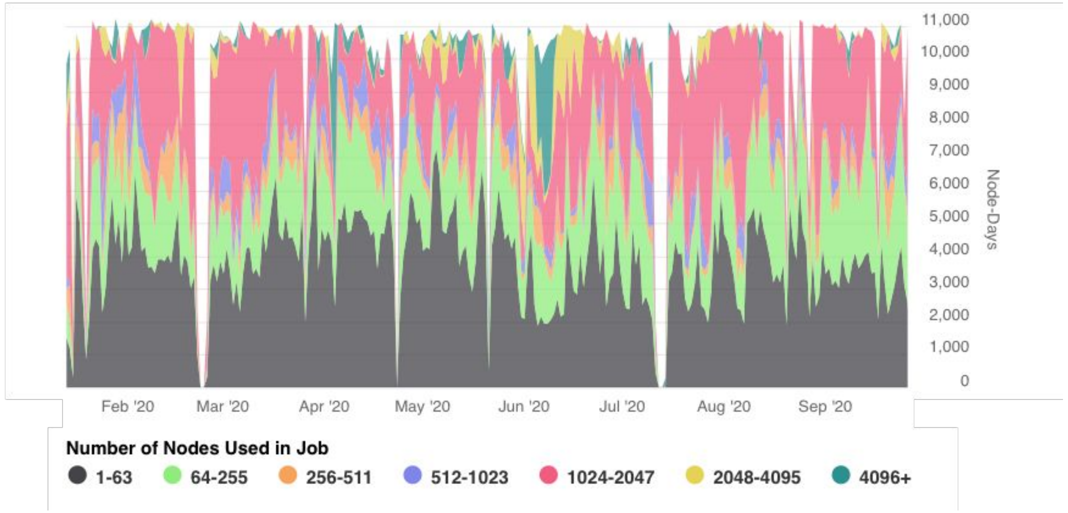
NERSC Production Workloads

Machine-Time Breakdown by Application
(1/14/2020 - 10/4/2020)



https://my.nersc.gov/application_usage_page_v2.php

Cori Job Size Chart
(1/14/2020 - 9/26/2020)



https://my.nersc.gov/jobsize.php

- In 2020, >22,000 different binaries run on Cori by >3,600 unique users
- Jobs run at all scales – from single node to full machine
- ~20 top applications account for >70% of computing cycles

What is Checkpoint/Restart (C/R)?

- **Checkpointing** is the action of saving the state of a running process to a checkpoint image file
- The process can be **restarted** later from the checkpoint file: continuing the execution from where it left off, on the same or different computer

Why Use C/R at NERSC?

- C/R is integral to many future plans at NERSC
    - e.g., supporting real-time workloads in 2025 timeframe
    - Early C/R adoption in your workload benefits you in the long term
- Enable long-running jobs
- Improved queue turnaround
    - A job split into smaller segments could complete before full-length job due to difficulty of scheduling long jobs
- Prepare for system failures, including PG&E Public Safety Power Shutdowns (California wildfires, extreme weather)

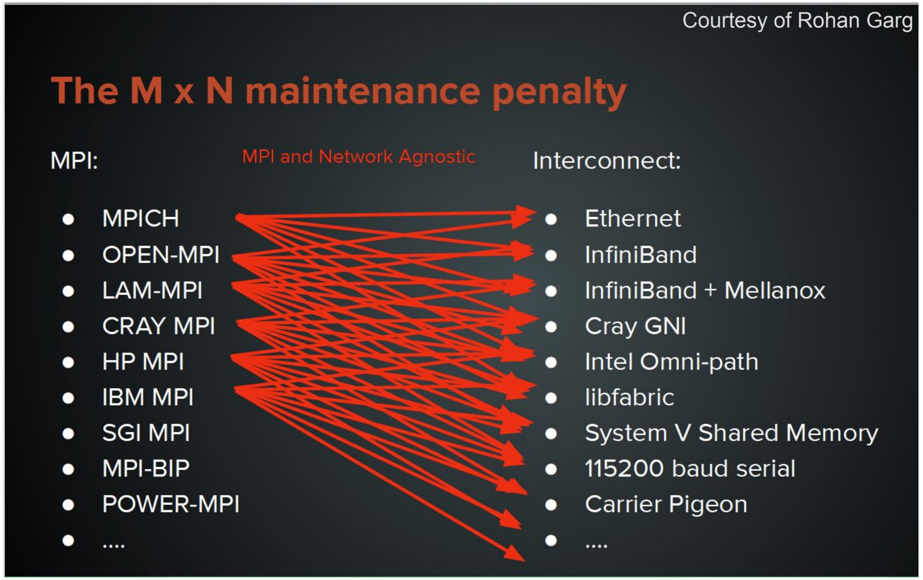
C/R Approaches

- Application-initiated checkpointing
  - Most applications at NERSC have some internal C/R support
  - Limited, inflexible (e.g., checkpoint only at end of iteration)
  - Burden for application developers

- Transparent checkpointing
  - Flexible, can stop/resume the execution at any point
  - No extra work for application developers
  - Vital for system-level checkpointing

But Transparent C/R in Production Is Hard!

- MPI+OpenMP applications dominate HPC workloads
- Developing and maintaining C/R tools for HPC applications is **labor-intensive and highly complex** because of
  - Ever-changing HPC systems
  - Diverse production workloads
- Major issue: maintainability of C/R tools over M (# of MPI implementations) × N (# of Networks)



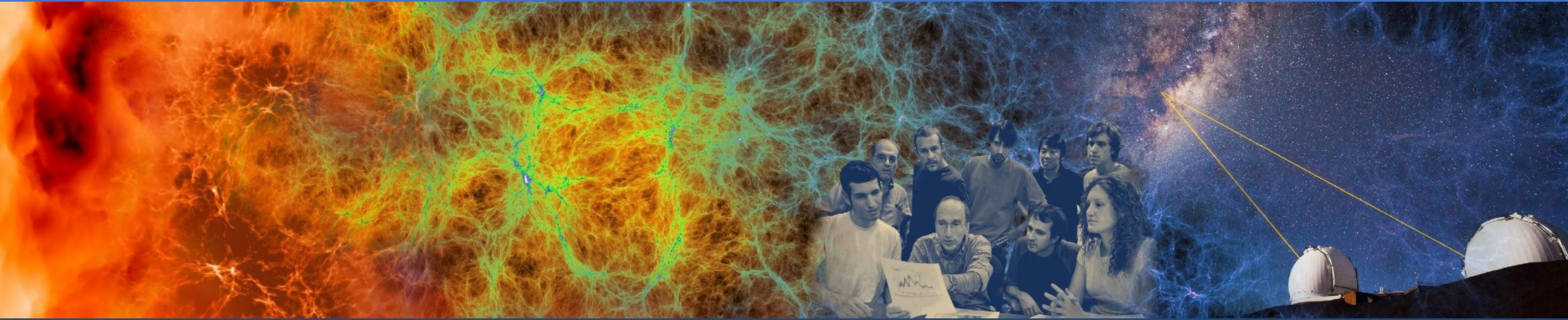
But Transparent C/R in Production Is Hard! (Cont.)

- Maintaining C/R codes for production use has **low priority compared to research**
- Long-term coordination between MPI library, kernel, & resource manager/scheduler developers has proven unsustainable
- Frequent OS updates on HPC systems can easily break C/R tools
- C/R tools incur runtime overheads & impose extra work upon users
  - Hinders the uptake of the C/R approach

Strategies to Enable C/R for NERSC Workloads

- DMTCP and MANA as C/R tool
  - DMTCP lives completely in user space
  - No need for coordination
  - A new implementation of DMTCP called **MANA** (MPI Agnostic Network Agnostic) addresses the critical $M \times N$ maintainability issue
- Promoting C/R uptake among NERSC Users
  - Variable-time job scripts, queue policies and user training
- Building an active and strong C/R Community
  - Promoting production-ready C/R tool development

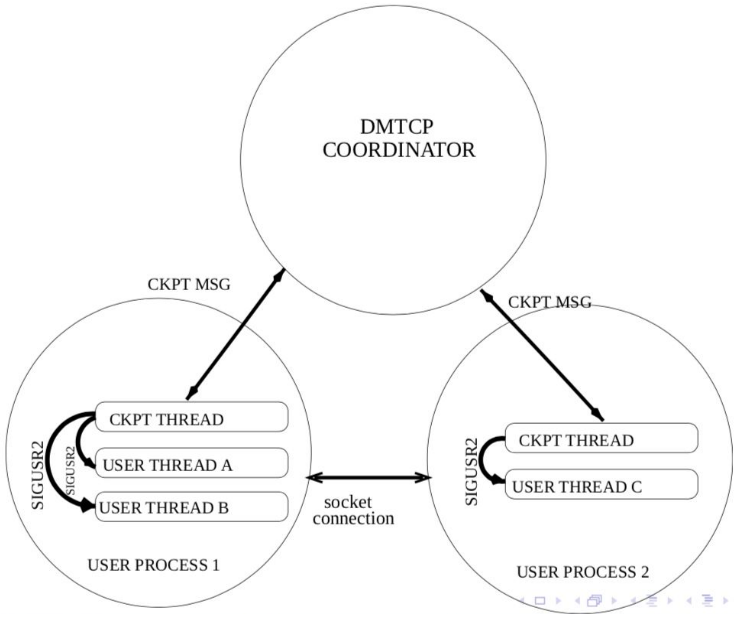
DMTCP and MANA



DMTCP: Distributed MultiThreaded CheckPointing

- DMTCP transparently checkpoints a single-host or distributed computation in user-space -- with no modifications to user code or to the O/S
- DMTCP supports a variety of applications, including MPI (various implementations over TCP/IP or InfiniBand), OpenMP, MATLAB, Python, C/C++/Fortran, shell scripts, etc.
- It is easy to extend DMTCP with plugins, e.g., MANA

DMTCP Architecture: Coordinated Checkpointing



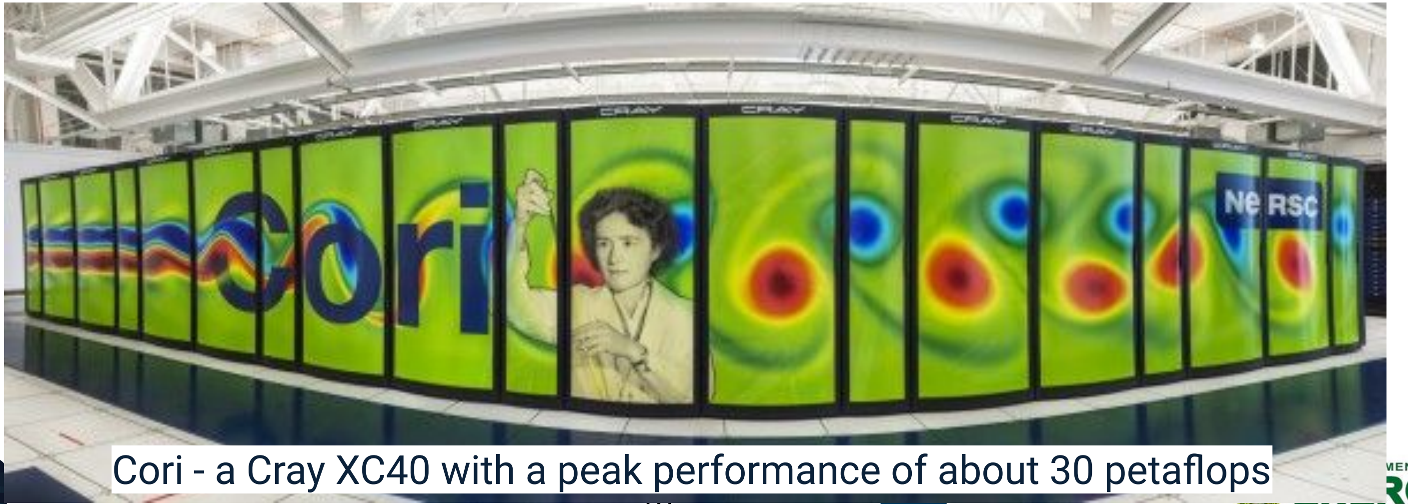
- One coordinator per computation
- One checkpoint thread per process, executing commands from the coordinator
- Either the checkpoint thread is active or the user thread, but not both at the same time
- Checkpoint files are backed up
- Everything is in user space, no admin privileges needed

DMTCP Commands

- **`dmtcp_coordinator`** - Coordinates checkpoints between multiple processes.
- **`dmtcp_launch`** -- Start a process under DMTCP control.
- **`dmtcp_restart`** -- Restart processes from a checkpoint image.
- **`dmtcp_command`** -- Send a command to the **`dmtcp_coordinator`** remotely

DMTCP on Cori at NERSC

- DMTCP works with serial/threaded applications on Cori
- Original DMTCP didn't work with Cray MPICH over Cray Aries network
- MANA for MPI workloads: a new plugin implemented in DMTCP



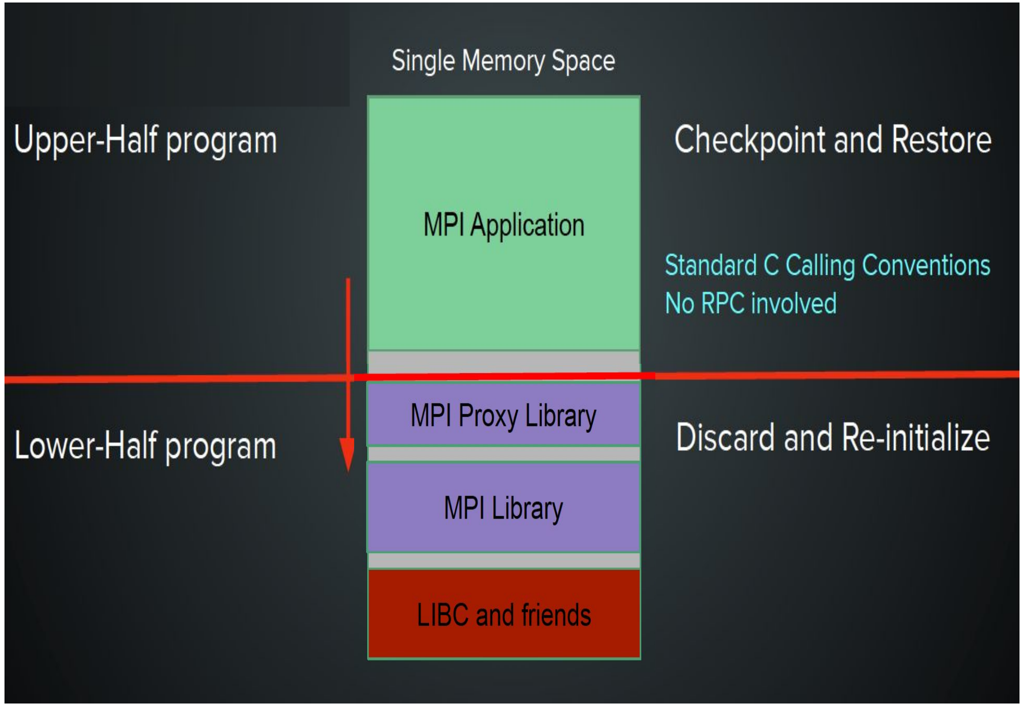
Cori - a Cray XC40 with a peak performance of about 30 petaflops

What is MANA?

- MANA for MPI: MPI-Agnostic Network-Agnostic Transparent Checkpointing
- MANA achieves MPI agnositic by *not* checkpointing MPI libraries and network agnostic by draining the network before checkpointing
  - A huge step forward towards ready-to-use C/R tools for future HPC platforms!
- For details on the novel MANA approach to checkpointing, see
  - Rohan Garg, Gregory Price, and Gene Cooperman, “MANA for MPI: MPI-Agnostic Network-Agnostic Transparent Checkpointing, High Performance Parallel and Distributed Computing (HPDC'19), 2019.
  - Gene Cooperman, “Checkpointing the Un-checkpointable: MANA and the Split-Process Approach”, MVAPICH User Group Meeting (MUG'19), Columbus, Ohio. video slides

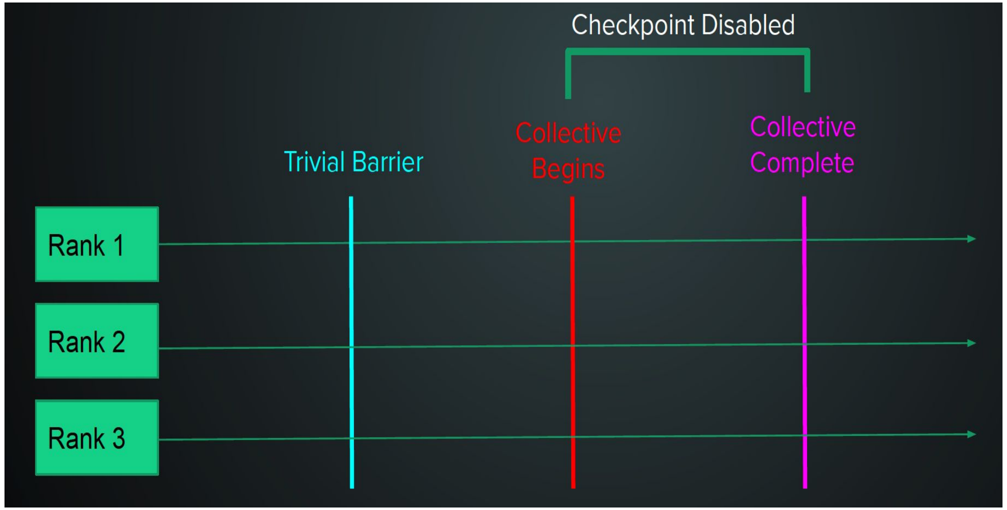
MANA: MPI Agnostic

- MANA employs a "split-process" approach
  - A single process contains two programs in its memory address space: upper-half and lower-half
  - Checkpoint upper-half only, & discard lower half
  - At restart time, lower half is re-initialized

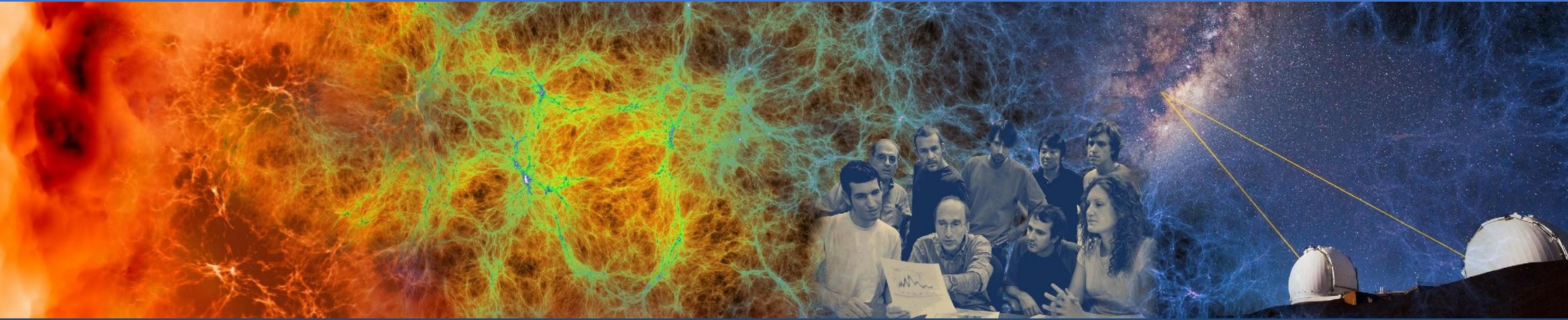


MANA: Network Agnostic

- MANA drains network before checkpointing
- For MPI collectives,
  - Preface all collective calls with a trivial barrier
  - When the trivial barrier is completed call the original collective
  - Prevent checkpointing in the middle of collective call



Enabling MANA/DMTCP for NERSC Workloads



MANA on Cori

- Began as a research code, not ready to use for NERSC's production workloads on Cori
- In collaboration with the developers, NERSC interns worked on fixing bugs & added more features over the summer
- MANA has been tested with VASP (Fortran, MPI), Gromacs (C++, MPI+OpenMP), and HPCG (C++, MPI+OpenMP) codes
- Evaluated the checkpoint overhead with HPCG to get ready for large-scale deployment (up to 512 ranks) on both Lustre file system and Burst Buffer nodes

Fixing Bugs and Improving the Code

- MANA debugging was quite challenging
  - A memory design that works with all kinds of processors at all scales was difficult
  - DMTCP employs many low-level tricks
  - Missing informative messages upon errors
  - Some bugs appear only at large scale
- A new design was developed to avoid overlapping of the lower-half and upper-half memory
- Added support for hugepages memory, which is enabled by default on Cori

Work from Prashant Chouhan Ref [3]

Fixing Bugs and Improving the Code (cont.)

- Added extra messages for debugging
- Fixed bugs from running with VASP, Gromacs and HPCG
  - Applications froze after multiple checkpoints were taken in sequence
  - Unsupported MPI calls were discovered in some of the computations
  - Mismatched MPI ranks with Checkpoint files
- More issues to resolve
  - Applications hang during checkpointing sometimes
  - Assertion errors during restart when using 1024 ranks or more

Work from Prashant Chouhan and Harsh Khetawat

C/R Overhead Evaluation

- Working to scale-up MANA, so limited scaling tests were done
- C/R overhead was evaluated using HPCG with up to 512 MPI ranks on both Cori's Lustre file system (cscratch1) and burst buffer
  - HPCG was run with 512 MPI ranks, 8 OpenMP threads per task on 64 Haswell nodes on Cori

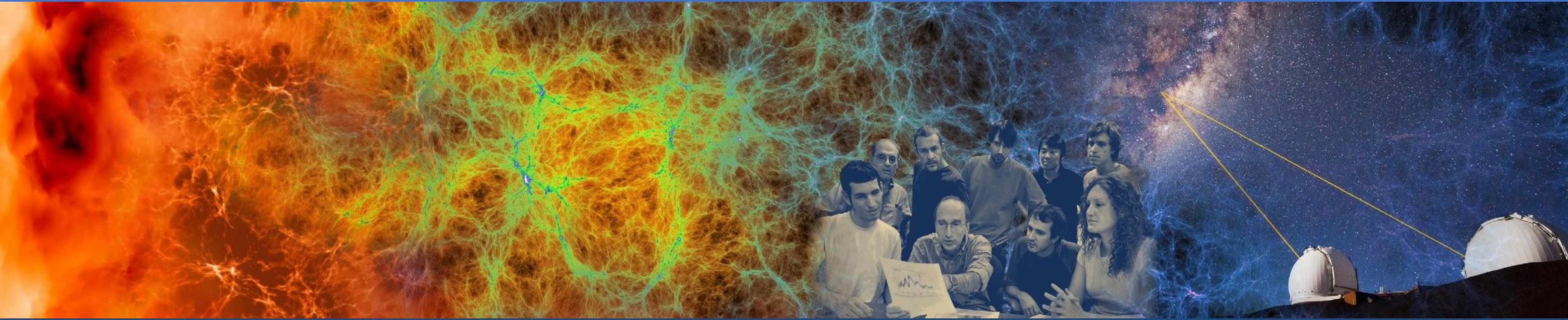
	Lustre File System	Burst Buffer	Total Memory Usage (GB)
Checkpoint Time (s)	640.3	30.1	5793.3
Restart Time (s)	110.3	39.2	

Work from Harsh Khetawat ref [4]

Next Steps for MANA for Production Deployment

- Enable MANA with more applications
  - Top 20 applications account for more than 70% of the machine time usage
- Scale up MANA for large-scale deployment
- Evaluate and improve C/R overhead for large scale jobs
  - Explore I/O options

Promoting C/R Uptake by NERSC Users



The “flex” Queue on Cori KNL

- To promote C/R adoption, we rolled out the “flex” queue with a 75% charging discount in April 2019
  - To compensate the runtime overhead incurred from C/R
  - Users have to specify a minimum time of 2 hours or less
  - A flex job can use up to 256 KNL nodes for up to 48 hours
- This creates opportunities for the Slurm scheduler to perform backfill, improving job throughput and increasing machine utilization

Variable-Time Job (VTJ) Scripts

- C/R imposes extra work for users
  - Resubmitting pre-terminated jobs multiple times until the job completes
- We developed the variable-time job (VTJ) scripts to simplify this process
- VTJ scripts add a few `sbatch` directives and `bash` functions in the Slurm job scripts, automatically splitting a long running job into multiple shorter ones, and self-resubmitting until the job completes

sbatch directives:

```
#SBATCH --time-min=2:00:00
#SBATCH --comment=48:00:00
#SBATCH --signal=B:USR1@<sig_time>
#SBATCH --requeue
```

bash functions

```
Requeue_job   #trap signals
func_trap     #action upon trap
parse_job     #process job info
```

Variable-Time Job (VTJ) Scripts (cont.)

  - User specifies a minimum and maximum time for the jobs
  - System then finds the best time slots for these jobs & automates the pre-terminated job resubmissions
  - User submits only one job script
- Allows better queue turnaround by utilizing backfill opportunities
- VTJ enables jobs of any length, e.g., weeks, months
- Can be used for applications with internal C/R support or checkpointing with external C/R tools
- Users running variable-time jobs in flex queue get 75% charging discount

Variable-Time Job Script with “flex” Queue

Original job script

```
#!/bin/bash
#SBATCH -J test
#SBATCH -q regular
#SBATCH -C knl
#SBATCH -N 2
#SBATCH --time=48:00:00
#SBATCH --error=%x-%j.err
#SBATCH --output=%x-%j.out

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4

srun -n32 -c16 --cpu_bind=cores ./a.out
```

Max walltime limit on Cori

```
func_trap() {
    $ckpt_command
    scontrol requeue ${SLURM_JOB_ID}
    scontrol update JobId=${SLURM_JOB_ID}
TimeLimit=${requestTime}
}
```

```
#!/bin/bash
#SBATCH -J test_vtj
#SBATCH -q flex
#SBATCH -C knl
#SBATCH -N 2
#SBATCH --time=48:00:00
#SBATCH --time-min=2:00:00  #the minimum amount of time the job should run
#SBATCH --error=%x-%j.err
#SBATCH --output=%x-%j.out
#
#SBATCH --comment=96:00:00  #desired time limit
#SBATCH --signal=B:USR1@300 #sig_time (300 secs) should match your checkpoint
overhead time
#SBATCH --requeue
#SBATCH --open-mode=append

# specify the command to use to checkpoint your job if any (leave blank if none)
ckpt_command=
max_timelimit=48:00:00

# requeueing the job if remaining time >0
. /usr/common/software/variable-time-job/setup.sh
requeue_job func_trap USR1

# user setting goes here
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4

# srun must execute in background and catch signal on wait command
srun -n32 -c16 --cpu_bind=cores ./a.out &

wait
```

Allow allocated time to vary

Specify desired run time; tracking remaining time for pre-terminated jobs

Triggers checkpoint and restart actions before terminating the job

Specify checkpoint command if any

Execute commands in func_trap upon receiving USR1 signal

VTJ Example 1: VASP Atomic Relaxation Jobs

```
#put any commands that need to run to continue
#the next job here
ckpt_vasp() {
    set -x
    restarts=`squeue -h -O restartcnt -j $SLURM_JOB_ID`
    echo checkpointing the ${restarts}-th job

    #to terminate VASP at the next ionic step
    echo LSTOP = .TRUE. > STOPCAR
    #wait until VASP to complete the current ionic step,
    #write WAVECAR file and quit
    srun_pid=`ps -fle|grep srun|head -1|awk '{print $4}'`
    wait $srun_pid

    #copy CONTCAR to POSCAR for next job
    cp -p CONTCAR POSCAR
    set +x
}

ckpt_command=ckpt_vasp
ckpt_overhead=300

#requeueing the job if remaining time >0
. /global/common/cori/software/variable-time-job/setup.sh
requeue_job func_trap USR1
```

```
#!/bin/bash
#SBATCH -q flex
#SBATCH -N 2
#SBATCH -C knl
#SBATCH -t 48:00:00
#SBATCH --time-min=2:00:00
#SBATCH --comment=48:00:00
#SBATCH --signal=B:USR1@300
#SBATCH --requeue
#SBATCH --open-mode=append
module load vasp/6.1.0-knl
export OMP_NUM_THREADS=4

#launching 1 task every 4 cores (16 CPUs)
srun -n32 -c16 --cpu-bind=cores vasp_std &


wait
```

VTJ Example 2: C/R Threaded Applications with DMTCP

Original Job Script

```
#!/bin/bash
#SBATCH -J test
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -C knl
#SBATCH -t 48:00:00
#SBATCH -o test-%j.out
#SBATCH -e test-%j.err

#user setting
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=64

./a.out
```

C/R Jobs with DMTCP Manual Resubmission

```
#!/bin/bash
#SBATCH -J test_cr
#SBATCH -q flex
#SBATCH -N 1
#SBATCH -C knl
#SBATCH -t 48:00:00
#SBATCH -o test_cr-%j.out
#SBATCH -e test_cr-%j.err
#SBATCH -time-min=2:00:00

#user setting
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=64

module load dmtcp
#checkpointing once every hour
start_coordinator -i 3600

#run job under dmtcp control
dmtcp_launch ./a.out
```

```
#!/bin/bash
#SBATCH -J test
#SBATCH -q flex
#SBATCH -N 1
#SBATCH -C knl
#SBATCH -t 48:00:00
#SBATCH -o test_cr-%j.out
#SBATCH -e test_cr-%j.er
#SBATCH -time-min=2:00:00

#for c/r with dmtcp
module load dmtcp

#checkpointing once every hour
start_coordinator -i 3600

#restart job from dmtcp checkpoint files
bash ./dmtcp_restart_script.sh
```

Variable-Time Job script for C/R with DMTCP

```
#!/bin/bash
#SBATCH -J test
#SBATCH -q flex
#SBATCH -N 1
#SBATCH -C KNL
#SBATCH --time=48:00:00
#SBATCH --error=test%j.err
#SBATCH --output=test%j.out
#SBATCH --time-min=02:00:00

#SBATCH --comment=48:00:00
#SBATCH --signal=B:USR1@300
#SBATCH --requeue
#SBATCH --open-mode=append

module load dmtcp nersc_cr
start_coordinator -i 3600 #checkpointing once every hour

#checkpoint/restart job
if [[ $(restart_count) == 0 ]]; then
    #user setting
    export OMP_NUM_THREADS=64
    export OMP_PROC_BIND=spread
    export OMP_PLACES=threads
    dmtcp_launch -j ./a.out &
elif [[ $(restart_count) > 0 ]] && [[ -e dmtcp_restart_script.sh ]]; then
    bash ./dmtcp_restart_script.sh &
else
    echo "Failed to restart the job, exit"; exit
fi

# requeueing the job if remaining time >0
ckpt_command=ckpt_dmtcp #additional checkpointing before pre-emption
requeue_job func_trap USR1

wait
```

User Trainings and “flex” Queue Usage

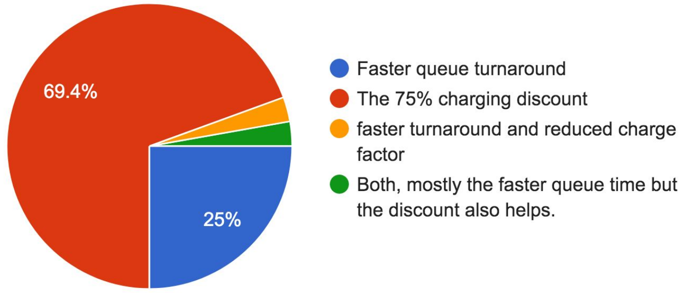
- Hosted multiple hands-on user trainings on variable-time job scripts and DMTCP. In 2020,
  - Flex usage was 2.8% of total Cori KNL cycles, enabling over 90% system utilization
  - 155 distinct users, 32 of them used >10,000 node hours
  - 50% of the top flex users ran variable-time jobs (per user survey)
  - 15 distinct DMTCP users with their serial/threaded workloads
- Top applications run in the flex queue include
  - VASP, NIMROD, MC_Descent, Shifter, Python, GPAW, CGYRO, QE, CHROMA, dftfe, disco_cEDM_v2.6, asap-python, OSIRIS, main, LAMMPS, E3SM, BerkeleyGW, CP2K, NWchem, etc.

User Surveys on “flex” Queue and VTJ Scripts

- Used user surveys to find out if “flex” queue serves the designated purpose (a faster turnaround) and meet users’ needs (May 2019)
  - 50% survey participants said “no or don’t know” - increased the flex queue priority based on user feedback as well as wait time monitoring data
  - Charging discount was helpful incentive
  - Issues: too short min time, VTJ is not easy to use, unpredictable start time

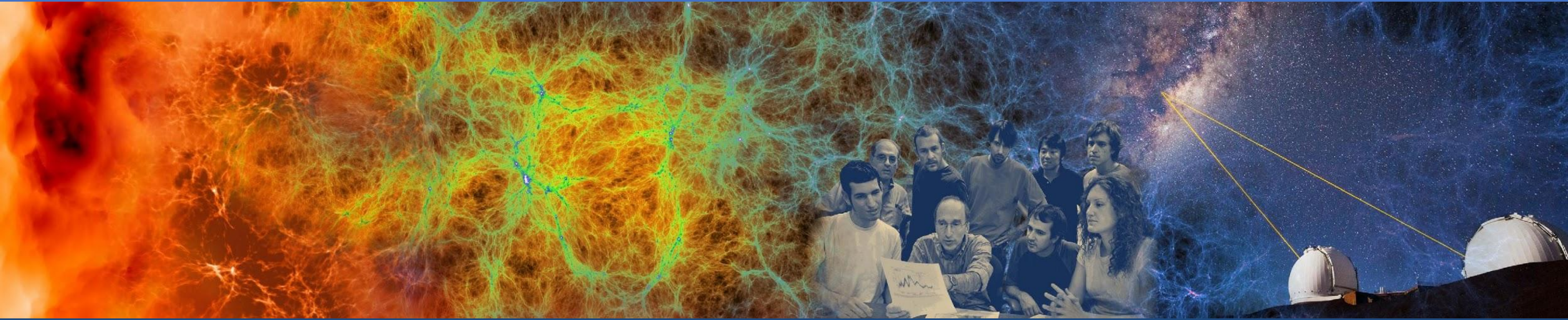
What is the main reason you used the flex QOS?

36 responses



- It works fine and I really like the 75% discount of the flex QOS!
- It's amazing, and I'm really grateful for the discount, which has allowed me to study so many more simulations. It's crazy, because I can sometimes get a 48 hr timeslot on flex quicker than a 2 hr times lot on knl_low or knl_regular. Thanks!
- It would be nice to remove the restriction of 2 hours minimum. Please don't remove this queue.
- Increase minimum time limit?
- The minimum job time is smaller than scavenger, which makes it less appealing.
- The flex QOS and NERSC's pushing of DMTCP has been great. I write a lot of my own code, and while it isn't too terrible to write restart/checkpoint friendly code, having DMTCP do that for me along with handling many of the pitfalls of using manually restarting code (I always have strange things happen, like that 1 in a million code quitting when it is reading to restart file) has been awesome for my workflow. I eagerly await the MPI version of DMTCP to use that with MPI compatible code too.

Use Cases in NERSC Workloads



Long Running Serial/Threaded Workloads

- SPADES - a genome assembler
  - A production workload of The Joint Genome Institute at the LBNL
  - Written in Python and C++, parallelized with OpenMP
- DMTCP has enabled long running SPAdes production workloads on Cori, and variable-time job scripts have made checkpointing and restarting jobs more manageable
  - Fixed several bugs exposed by the SPAdes workflow: a large number of threads, ~TB checkpoint image, deleted temporary files required at restart
  - The largest cases with about a terabyte memory usage suffer from high checkpoint overhead; we are looking into various I/O options

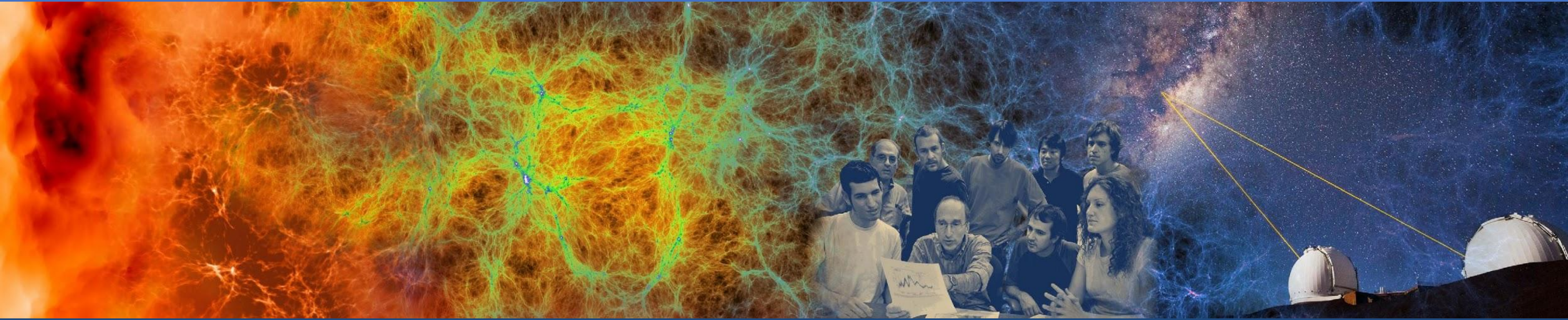
VASP: Rank #1 Application at NERSC

- VASP is a widely used materials science code
  - Written in Fortran 90, parallelized with MPI (V5) or MPI + OpenMP (V6)
  - Uses FFT and linear algebra libraries, e.g. MKL, FFTW
  - Uses >20% of computing cycles; 455 active users at NERSC
- Atomic relaxation jobs have been running with VTJ + internal C/R
  - MANA which is preferred: allowing predictable checkpoint overheads & checkpointing/restarting at any point of execution
- MANA enables VASP users to:
  - Run long-running RPA jobs that have no internal C/R support
  - Save machine time significantly for some long running VASP jobs, which spike in memory usage in their final computation stage.

Gromacs and MANA

- Gromacs is a widely used molecular dynamics (MD) code
  - Written in C++, parallelized with MPI + OpenMP
  - Uses FFT, LAPACK, etc.
  - Ranked #41 at NERSC; 44 users ran Gromacs in 2020
- With MANA working with Gromacs, users can now produce the exact same results as non-interrupted jobs for MD calculations at any point of execution
  - While Gromacs has internal C/R, MANA's C/R transparently saves all states, including random seeds. This makes it an ideal tool to restart chaotic MD simulations, for which trajectories diverge rapidly with even slight changes in restart data.

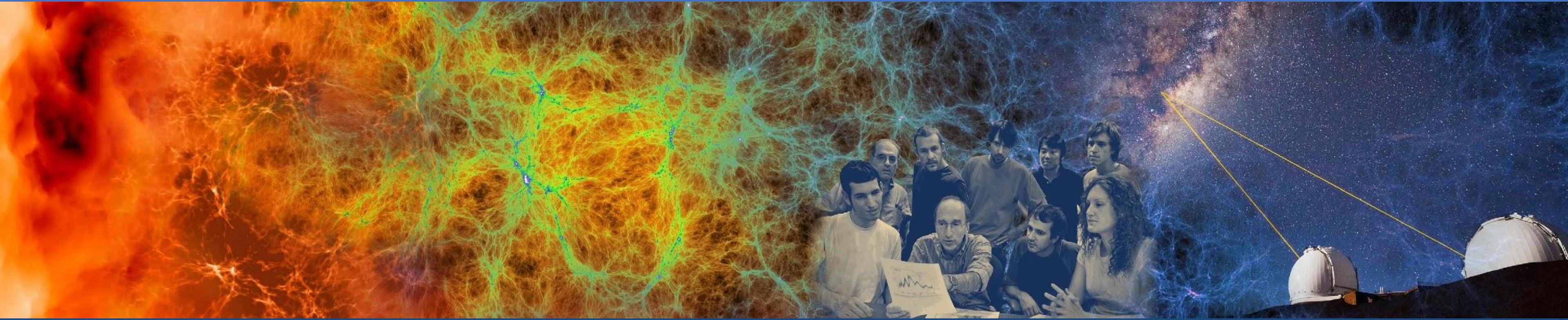
Community Building



First International Symposium on Checkpointing for Supercomputing (SuperCheck21)

- Date: **Feb 4-5, 2021** (8:00 am - 12:45 pm PST)
- Online symposium featuring the latest work in checkpoint/restart research, tools development, and production use
- Participation highly encouraged!
  - We especially encourage abstract submissions on adopting C/R tools in production workloads
- More details at https://ckpt-symposium.lbl.gov

Summary and Future Work



Summary of NERSC C/R Efforts

- Enabled DMTCP for NERSC serial/threaded applications
- Enabled MANA checkpointing tool for top applications at NERSC
  - VASP (Fortran, MPI), Gromacs (C++, MPI+OpenMP)
- Developed variable-time job scripts that split a long running job into multiple shorter ones and automate job resubmission
- Rolled out queue policies to help users adopt a C/R approach in their production workloads
- Building an active C/R community: SuperCheck21

Future Work

- Enable MANA for NERSC production workloads at all scales
  - Make C/R overhead manageable at scale
- Provide user training to help users to adopt MANA
- Implement a preempt queue to support real-time workloads
- Enable MANA for our next supercomputer, Perlmutter - an NVIDIA GPU system

References

1. R. Garg, G. Price, and G. Cooperman, “MANA for MPI: MPI-Agnostic Network-Agnostic Transparent Checkpointing”, Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, June 2019, Pages 49-60.
2. Gene Cooperman, “Checkpointing the Un-checkpointable: MANA and the Split-Process Approach”, MVAPICH User Group (MUG'19), Columbus, Ohio. Video Slides
3. Jason Ansel, Kapil Arya, and Gene Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop", IEEE International Parallel and Distributed Processing Symposium (IPDPS’09), Rome, Italy, May, 2009
4. Prashant Chouhan, *et al.*, “Providing Fault-Tolerance to NERSC Workloads Using MANA”, to be submitted to EuroMPI 2021
5. Harsh Khetawat, *et al.*, “Scale-up study with DMTCP/MANA on Lustre File Systems and Burst Buffer”, to be submitted to EuroMPI 2021
6. DMTCP: Website DMTCP code MANA code
7. NERSC Documentation
   a. NERSC website checkpoint/restart (DMTCP)
   b. Variable-time job scripts; github repo
8. Training Materials:
   a. Variable-time job scripts for VASP users
   b. Variable-time job scripts
   c. DMTCP for users running serial and threaded applications

Acknowledgements

- Rohan Garg, Twinkle Jain, and Prashant Chouhan at Northeastern University for developing/improving MANA and DMTCP
- Harsh Khetawat at North Carolina State University for evaluating the checkpoint/restart overhead on Cori
- Tiffany Connors, Steve Leak, and Chris Samuel at NERSC for the variable-time job script development and user support
- This work was supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231.
- The work of the third author was supported by National Science Foundation Grant OAC-1740218 and by a grant by Intel Corporation.

Thank you!