A storage system fit for Exascale

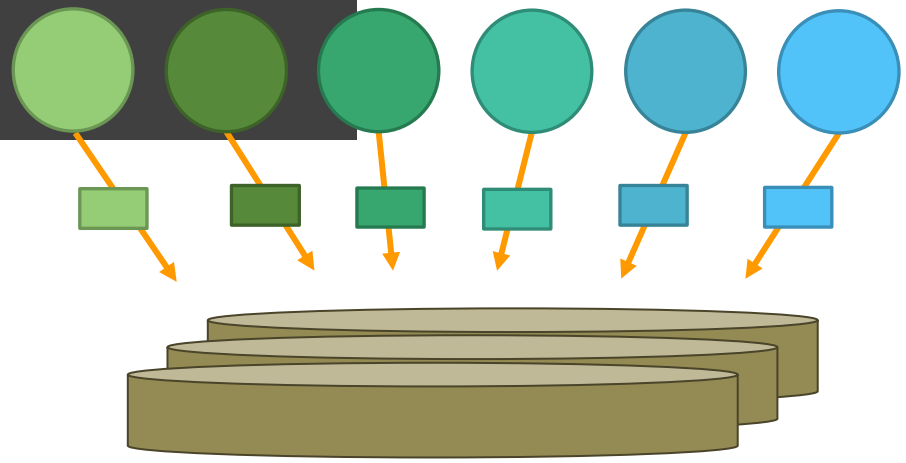
Kathryn Mohror
Salishan 2022

El Capitan's I/O system is designed to meet the needs of today's and future workloads

- As HPC system compute capabilities increase, jobs generate data at a faster rate
- HPC workloads are changing and stressing the I/O system in new ways
- We designed the I/O system of El Capitan to support today's workloads and with an eye to the future
- Design features
  - Hierarchical storage system with near-node local storage called Rabbits and a parallel file system
  - Will handle I/O for traditional use cases, e.g, checkpoint/restart, reading input files, writing output
  - Rabbit design enables future-looking I/O support, e.g., in situ analytics, complex workflows, AI

Traditional HPC workloads have experienced I/O slowdowns for more than a decade

```
int main(int argc, char* argv[]) {
  MPI_Init(argc, argv);

  read_input();

  for(int t = 0; t < TIMESTEPS; t++)
  {
    /* ... Do work ... */
    /* ... Synchronization ... */

      write_output();

  }

  MPI_Finalize();
  return 0;
}
```



- Processes perform file operations simultaneously, increasing load and contention on the parallel file system
- Result is slow I/O times that can be 10s of minutes with high variability
- I/O times as fractions of execution times are increasing with faster compute rates
- Mitigations
  - Easy: do less I/O
  - Easier: use burst buffers
  - More challenging: adopt data reduction or management methods, e.g., checkpoint/restart libraries or in situ analysis

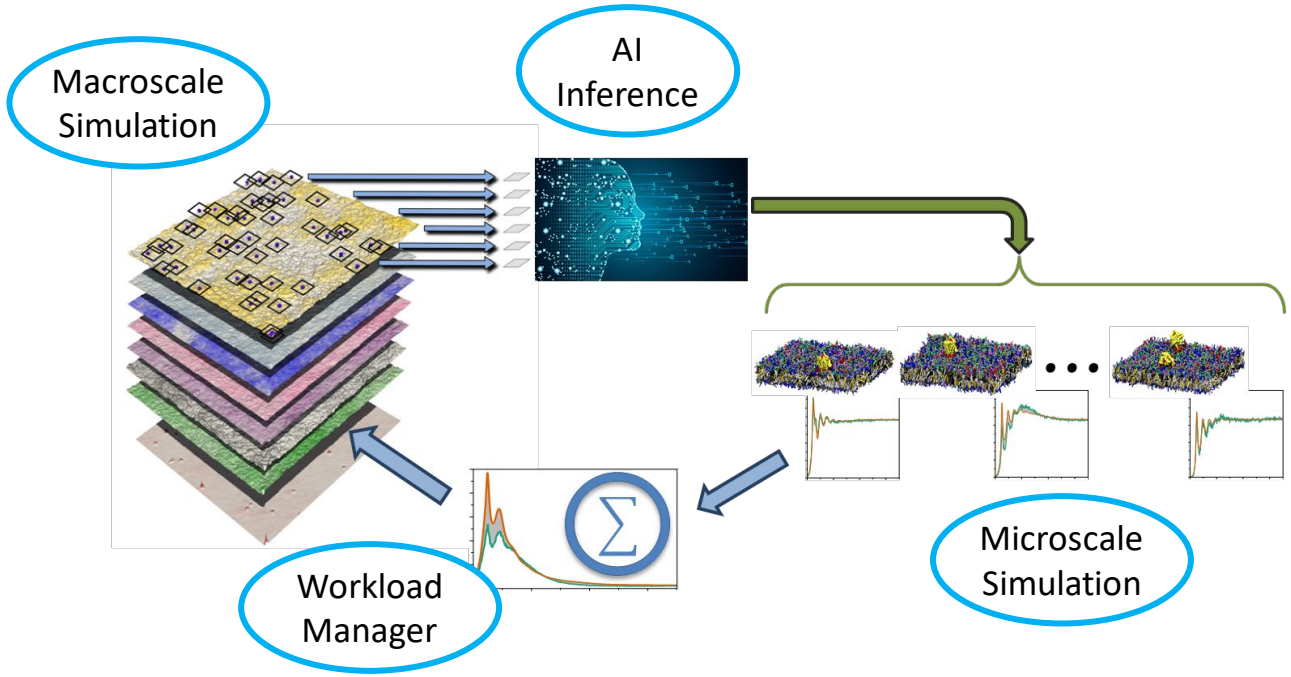
Machine learning training workloads are stressing the I/O system in new ways

- Machine learning training performs a large number of small, random reads
- Parallel file systems are not designed for this workload
- Example: LBANN crashed Sierra with a large-scale training run with 1000 trainers
- Mitigations
  - Easier: Use burst buffers
  - More challenging: in-memory data management (e.g., LBANN, DeepIO); specialized user-level file systems

LBANN: Livermore Big Artificial Neural Network Toolkit: https://github.com/LLNL/lbann

Complex workflows are struggling with I/O slowdowns too

- Complex workflow I/O patterns are problematic for parallel file system. Smaller, temporary files passed between stages create bottleneck
- Example: MuMMI workflow took 30 minutes for data scans for workflow feedback.
- Mitigations
  - Easy: do less I/O
  - Easier: shared burst buffers
  - More challenging: adopting in situ analysis, adopting other communication strategies like database or queuing software



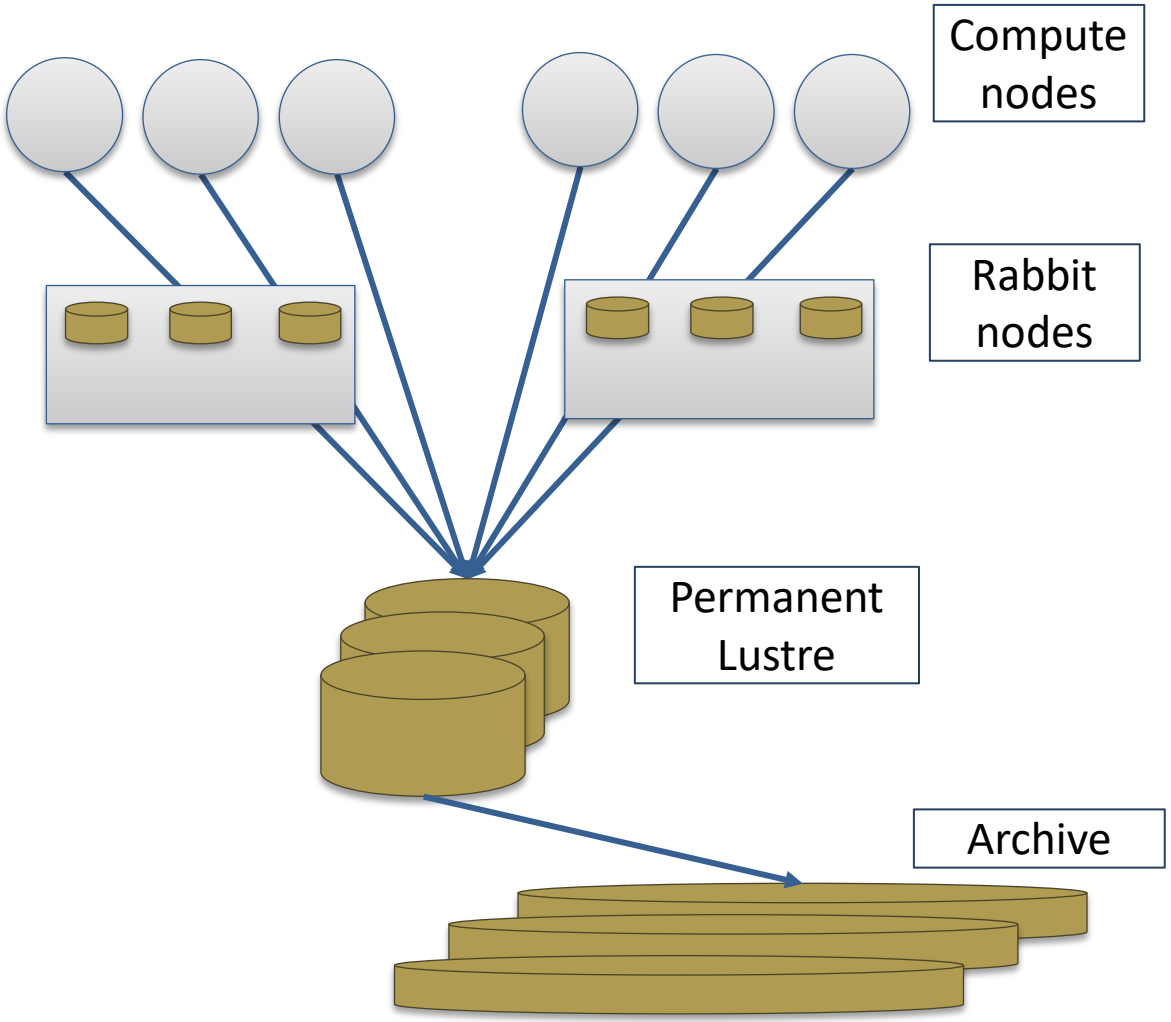
MuMMI: Machine-Learned Modeling Infrastructure: https://github.com/mummi-framework, Di Natale, SC'19

Our storage systems are hindering the development of new computing paradigms

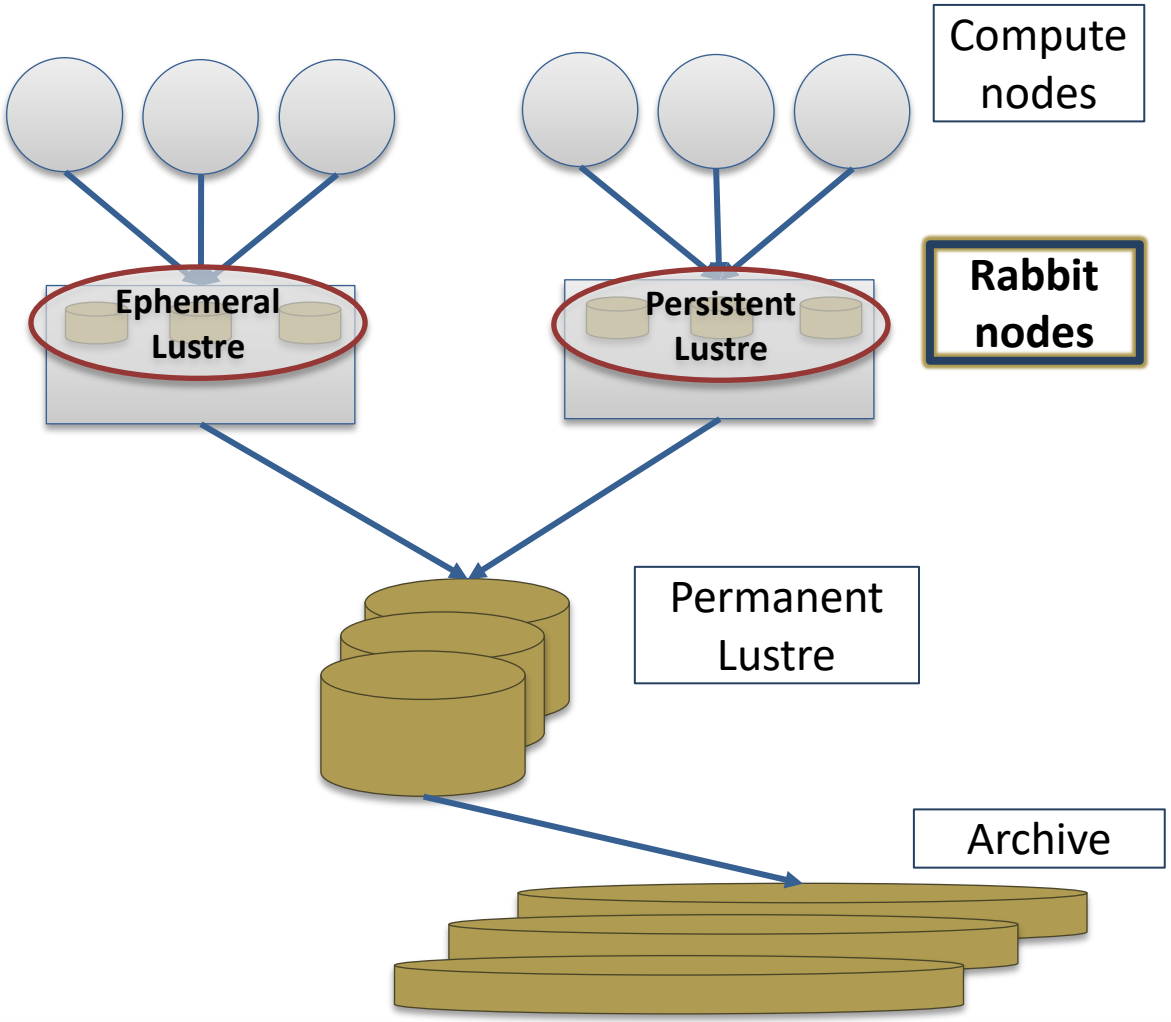
- Mitigation strategies are either less than satisfactory…
- or require significant developer time to implement and still need work to get right

How can we make it easier for domain scientists to try new computing paradigms on our systems?

The El Capitan Storage System

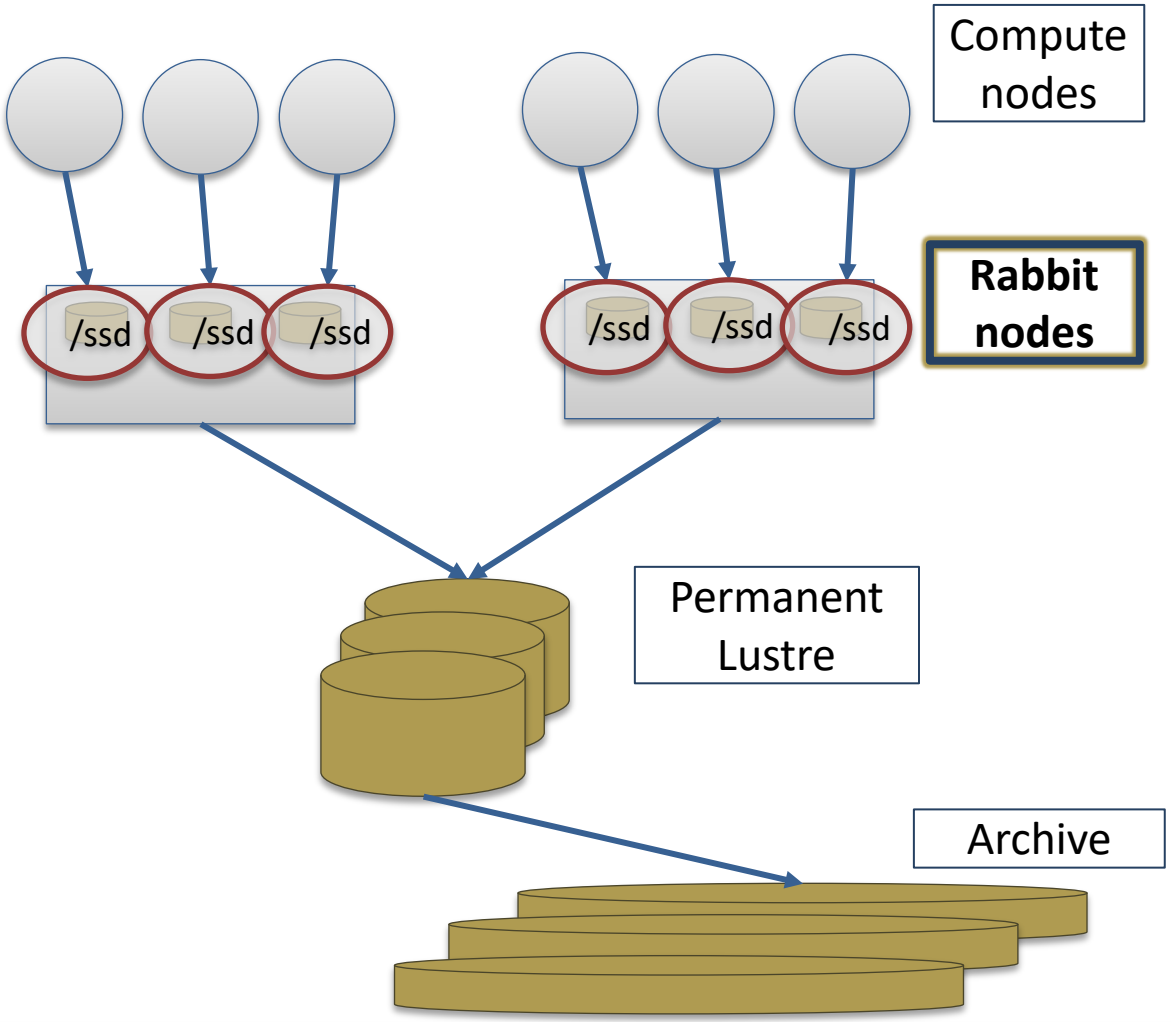


The El Capitan Storage System



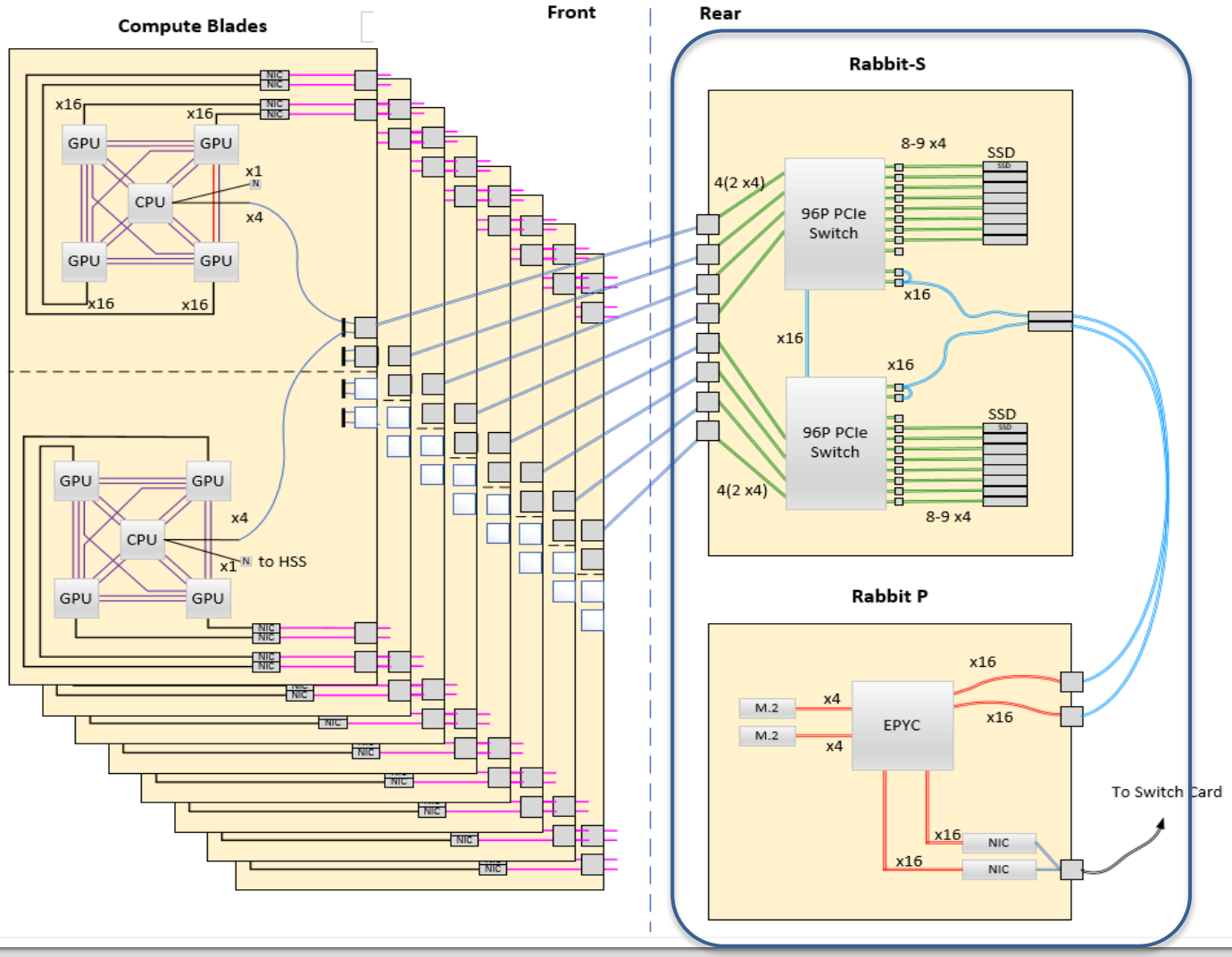
- **Ephemeral Lustre** is a temporary Lustre instance that runs over Rabbit storage and exists for a single job
- **Persistent Lustre** is a Lustre instance that runs over Rabbit storage and exists across multiple job lifetimes (e.g., shared project space)

The El Capitan Storage System



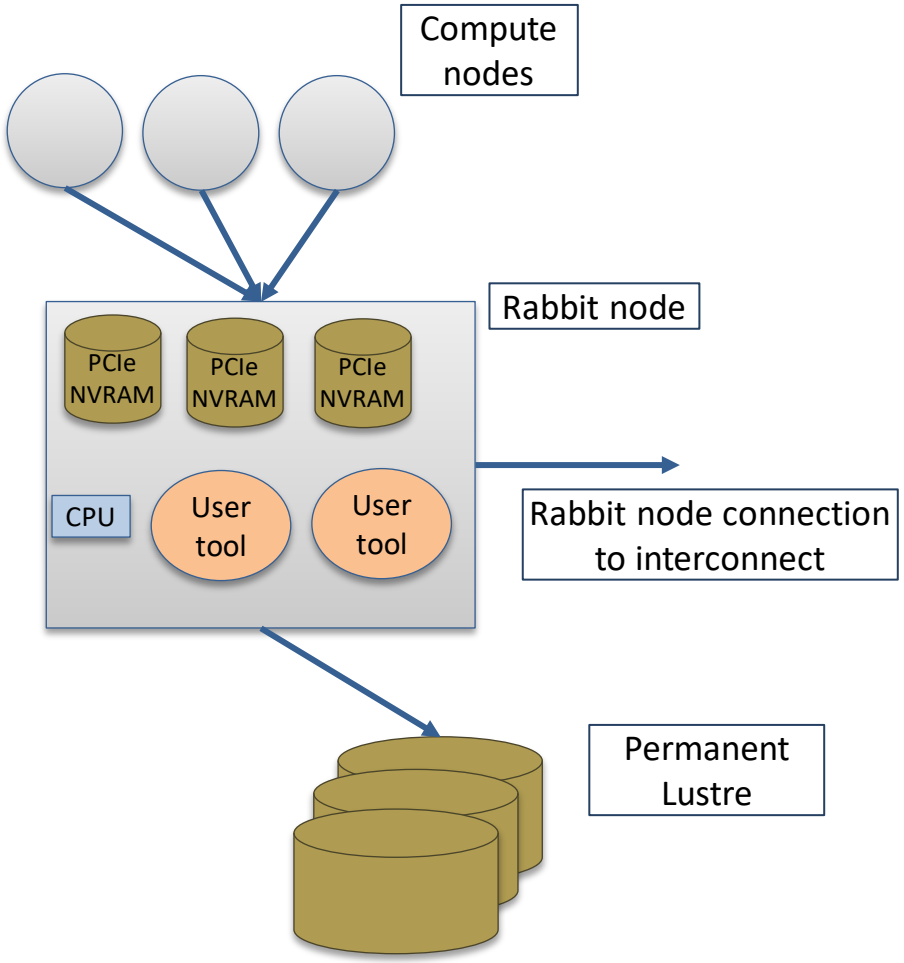
- **Node-local storage** will be presented as a temporary file system on rabbit storage, private to each compute node (just like on Sierra today)
- Node local storage exists for a single job

The El Capitan Rabbit Architecture



- One Rabbit blade per compute chassis
- Each Rabbit houses 18 SSDs (16+2 spare) with PCIe connections to every compute node via PCIe switch (*Rabbit-S*)
  - 2TB capacity per compute node
- Each Rabbit contains 1 AMD EPYC CPU (*Rabbit-P*)
- Rabbit blades are connected to the high-speed interconnect

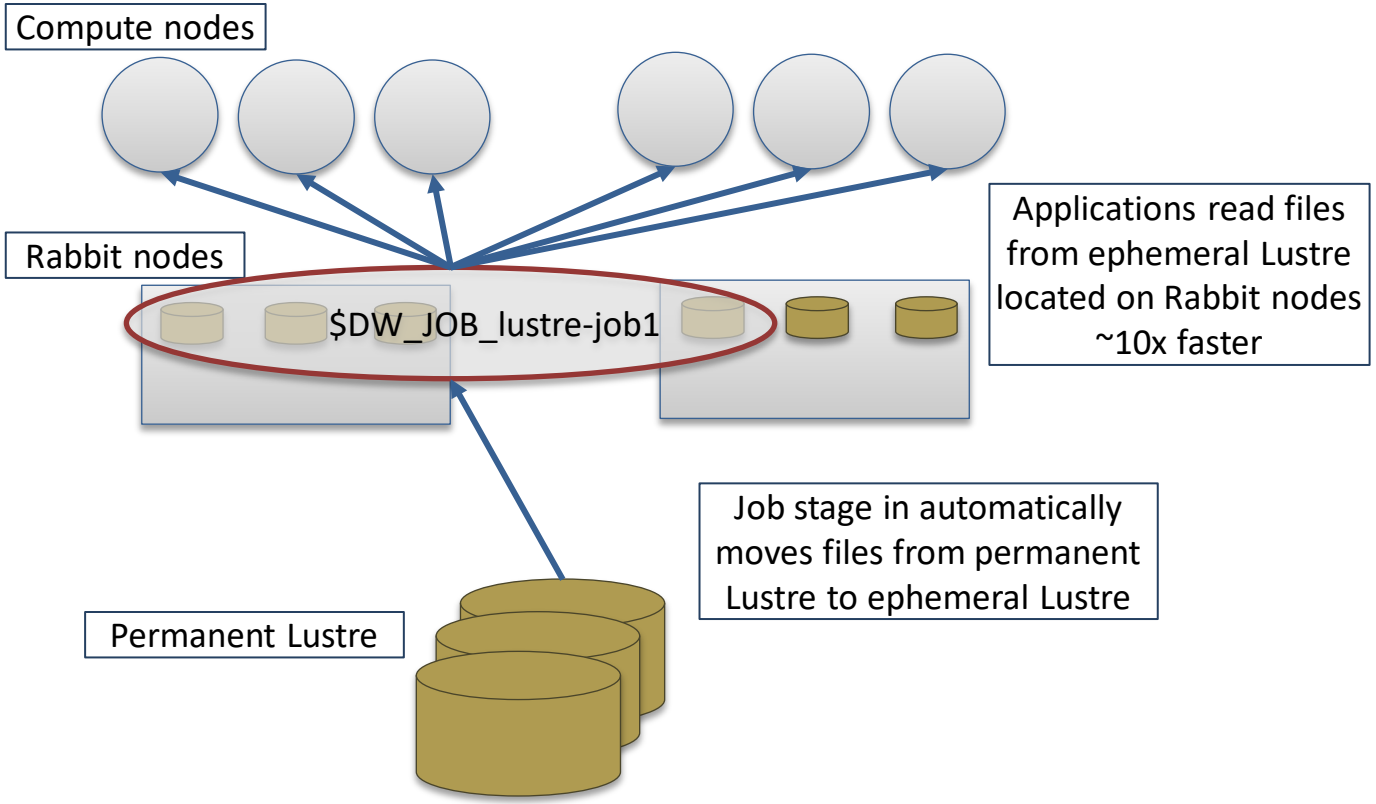
The El Capitan Rabbit High-Level Design



- User tools can run in containers on the Rabbits, e.g., I/O libraries, checkpoint/restart libraries, in situ analysis, and access the storage allocations for the job
- Rabbits have connection to system interconnect and PCIe connection to local compute nodes
  - User tools can communicate between Rabbits
  - User tools can also communicate between compute nodes and Rabbits

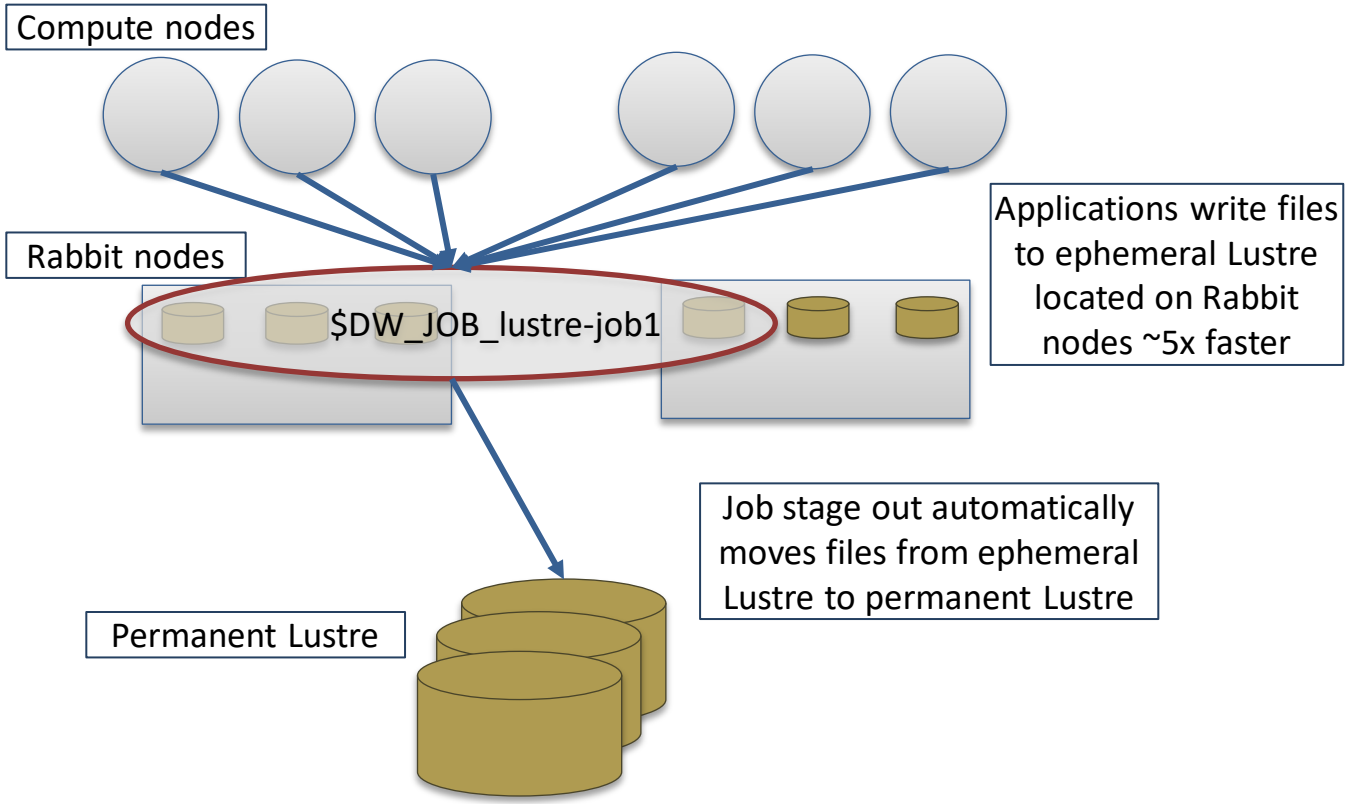
How can the Rabbits help with application input?

```
#DW jobdw type=lustre capacity=20TB name=lustre-job1
#DW stage_in source=/p/lustre1/<username>/data/ dest=$DW_JOB_lustre-job1
```



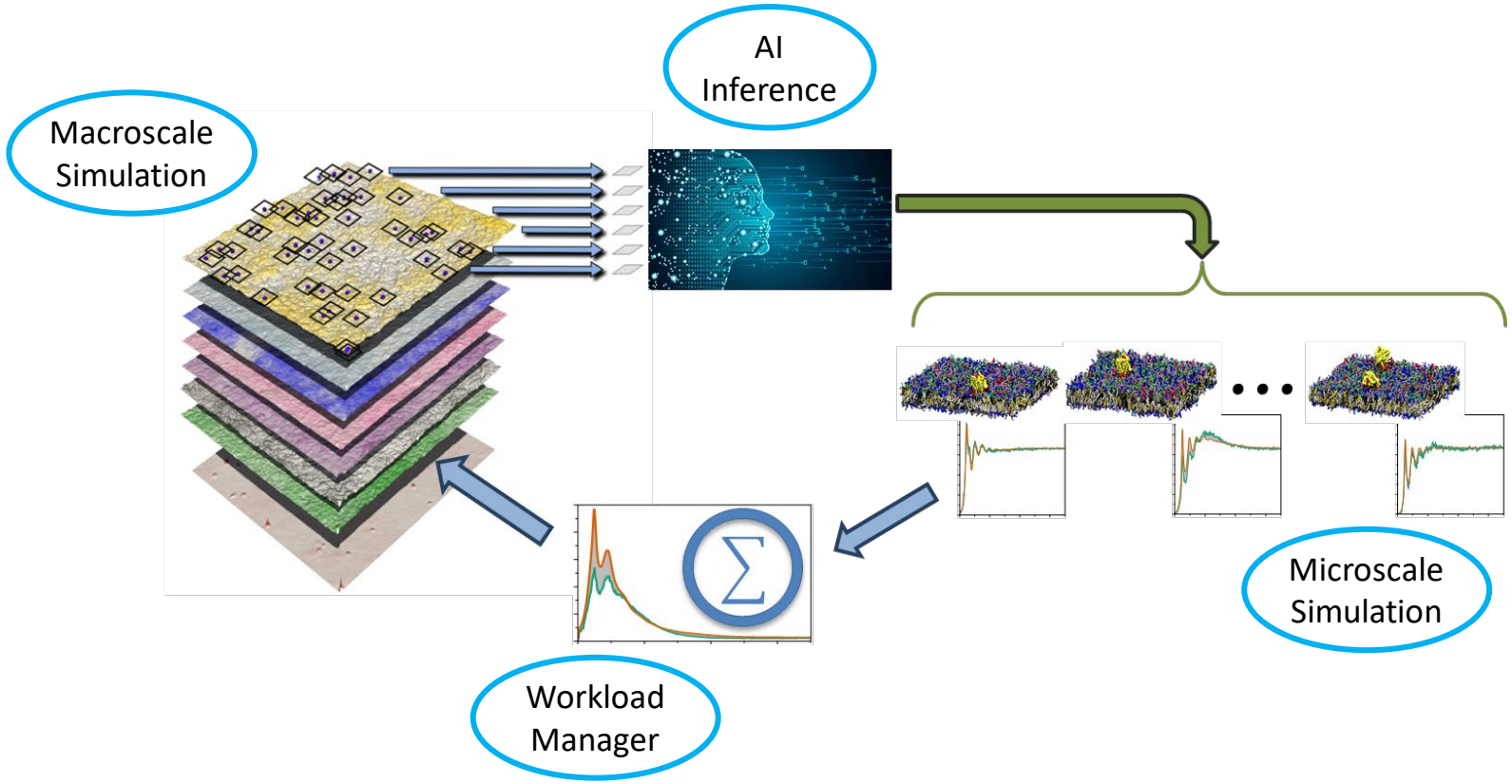
How can the Rabbits help with application output?

```
#DW jobdw type=lustre capacity=20TB name=lustre-job1
#DW stage_out source=$DW_JOB_lustre-job1 dest=/p/lustre1/<username>/data/
```



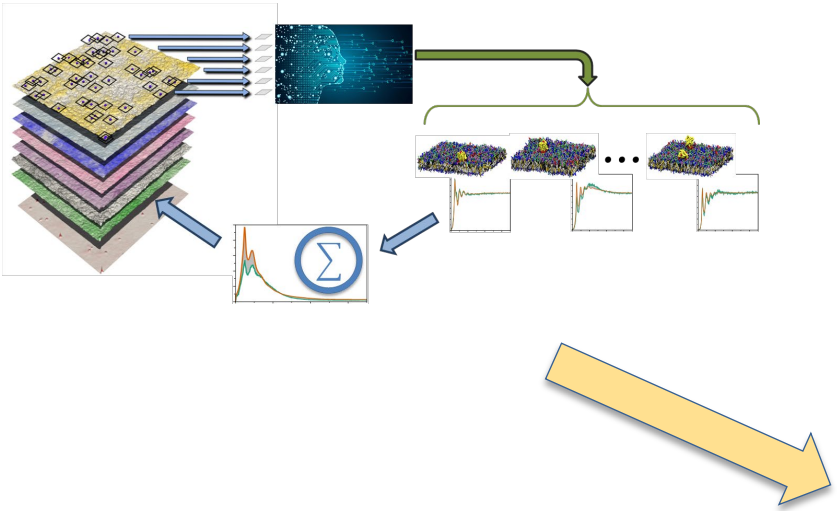
How can we make it easier for domain scientists to try new computing paradigms on our systems?

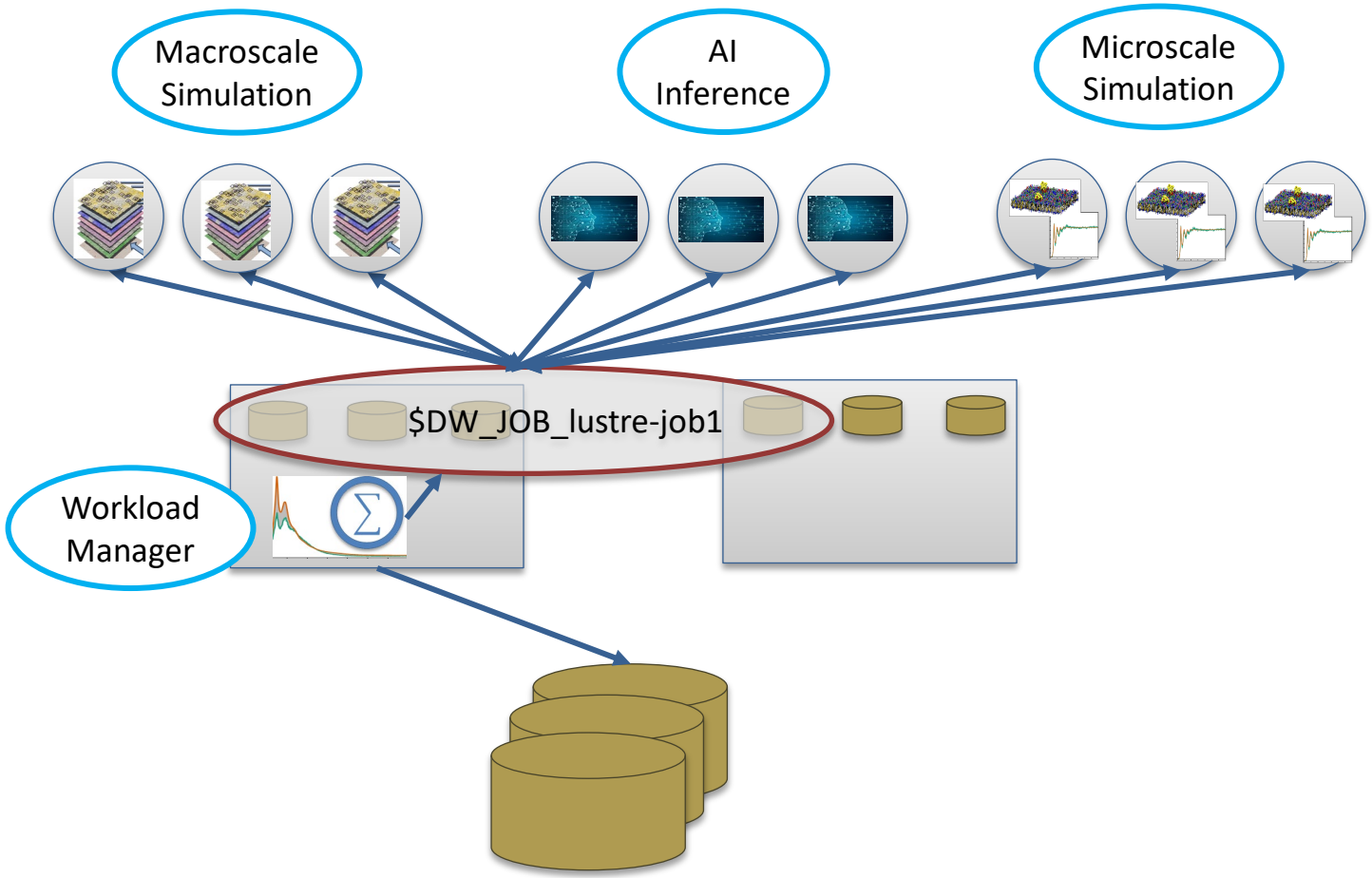
How can the Rabbits help complex workflows?



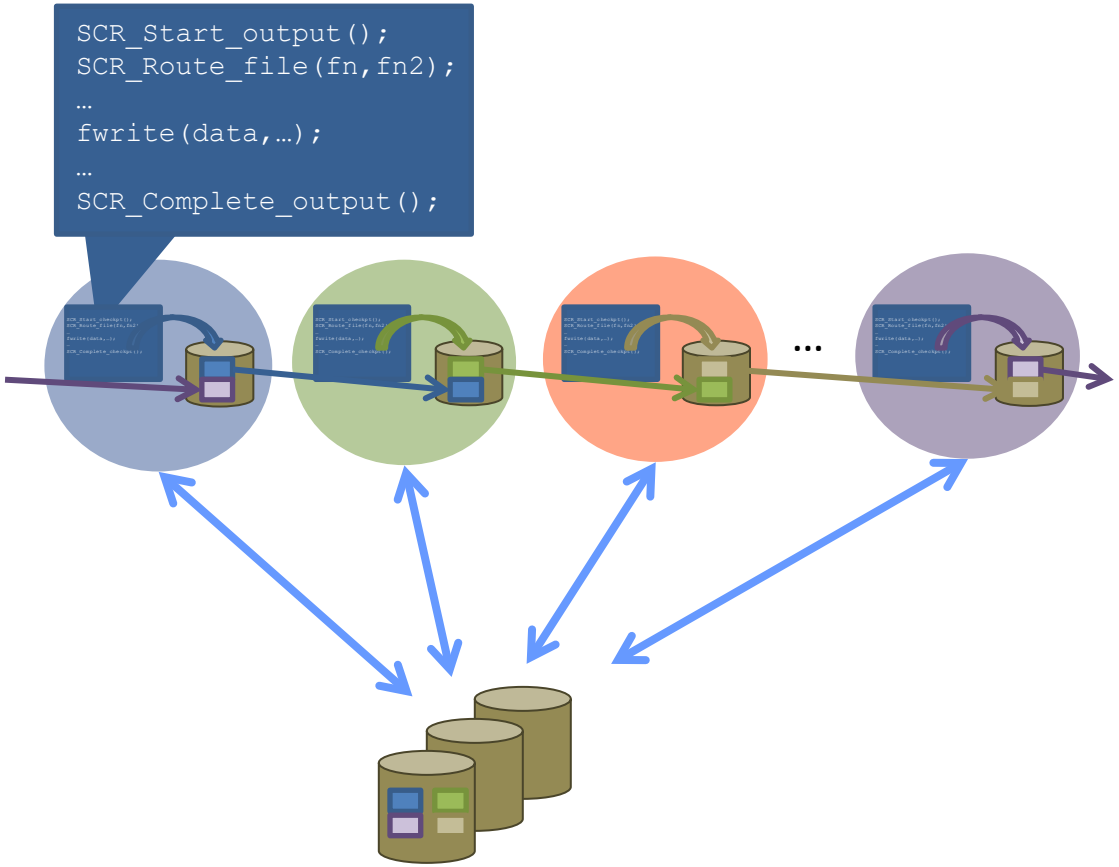
"A Massively Parallel Infrastructure for Adaptive Multiscale Simulations: Modeling RAS Initiation Pathway for Cancer" Di Natale et al., SC'19

How can the Rabbits help complex workflows?



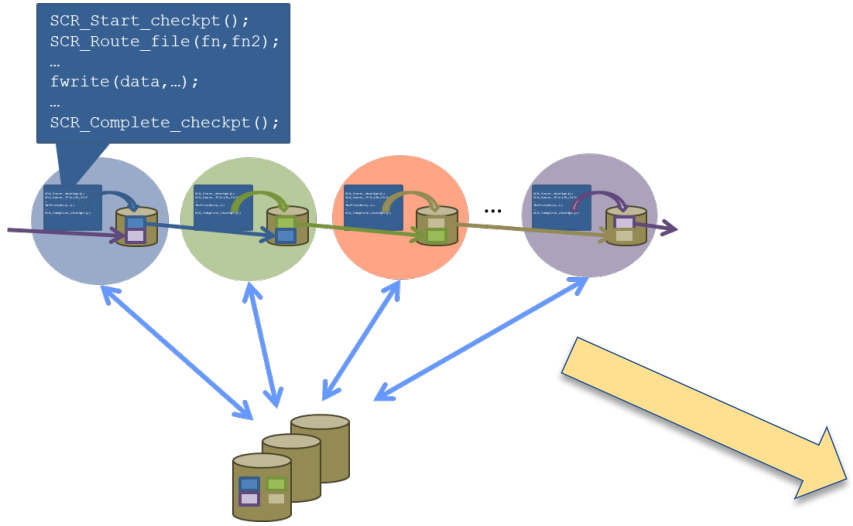


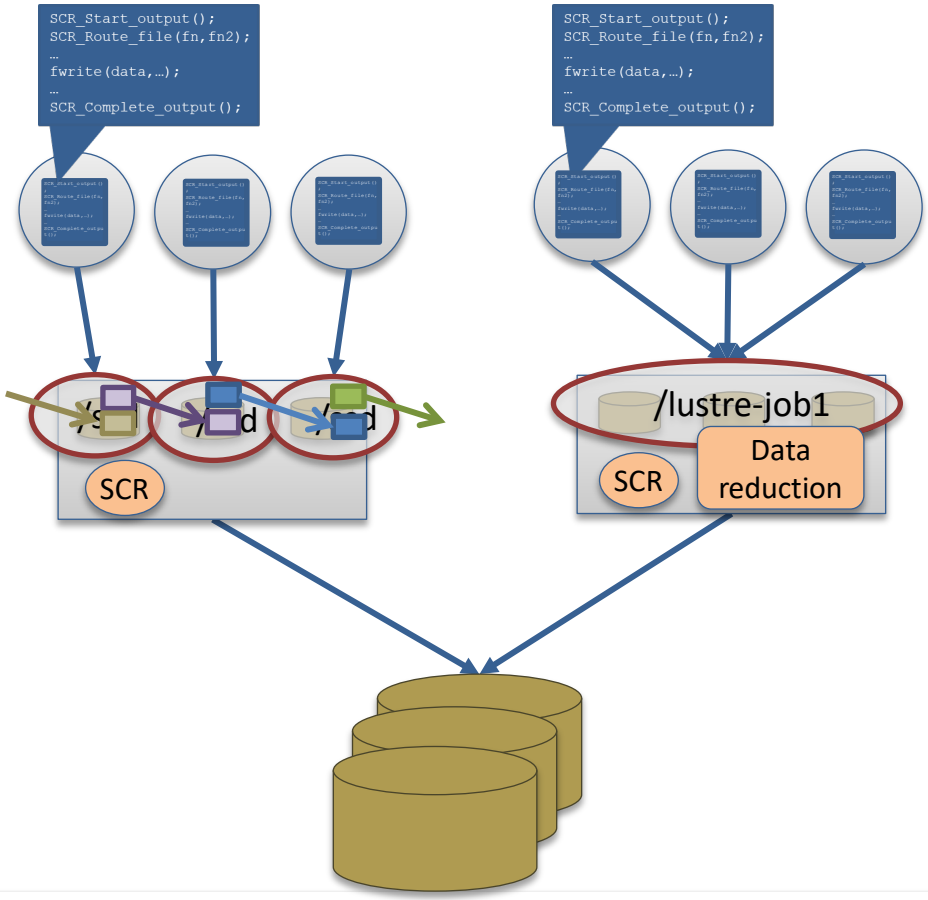
How can the Rabbits help I/O tools like SCR?



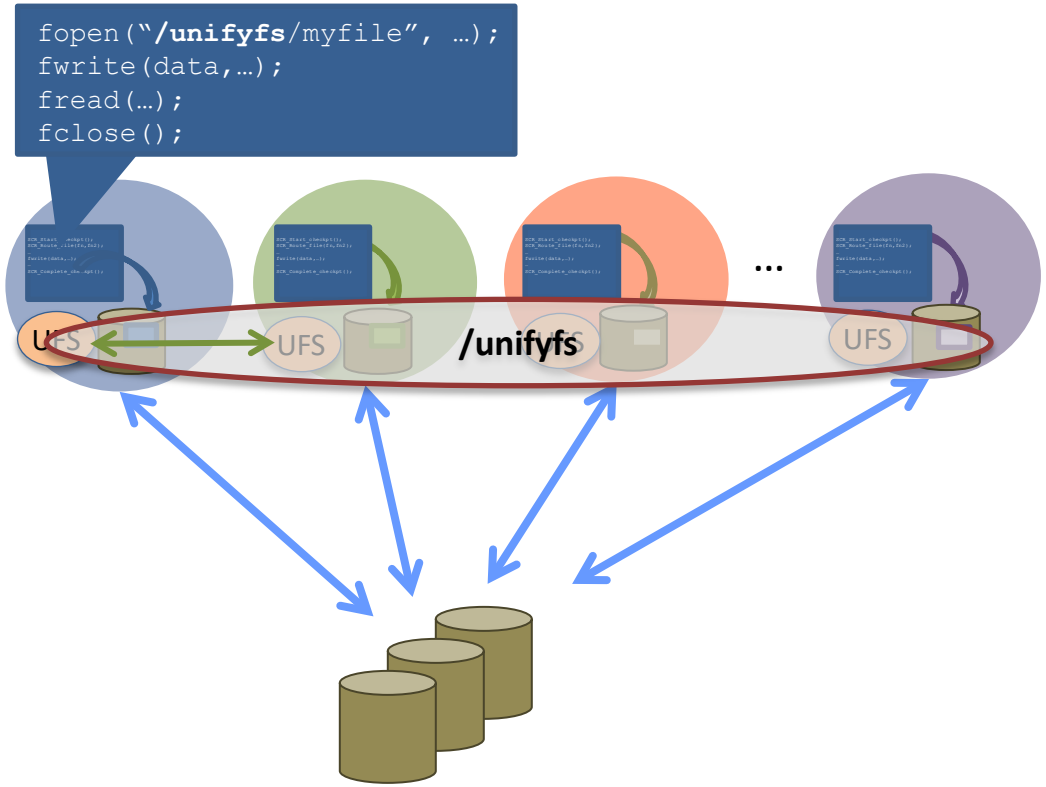
"Scalable Checkpoint/Restart Library: https://github.com/llnl/scr

How can the Rabbits help I/O tools like SCR?



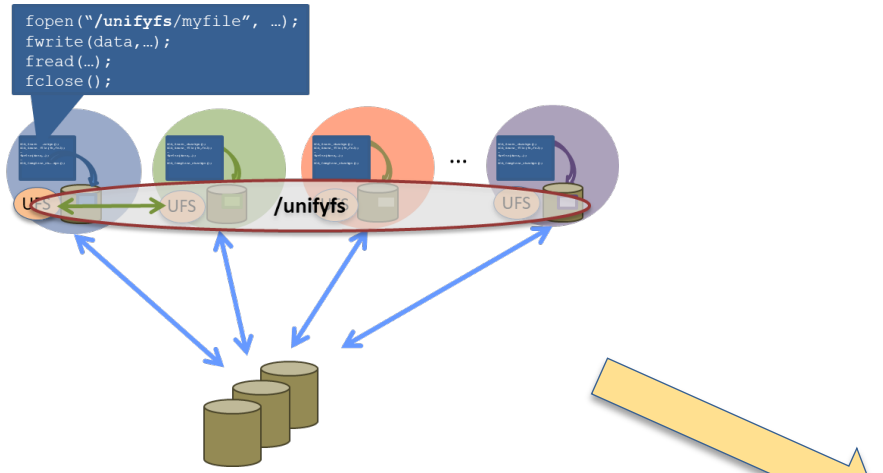


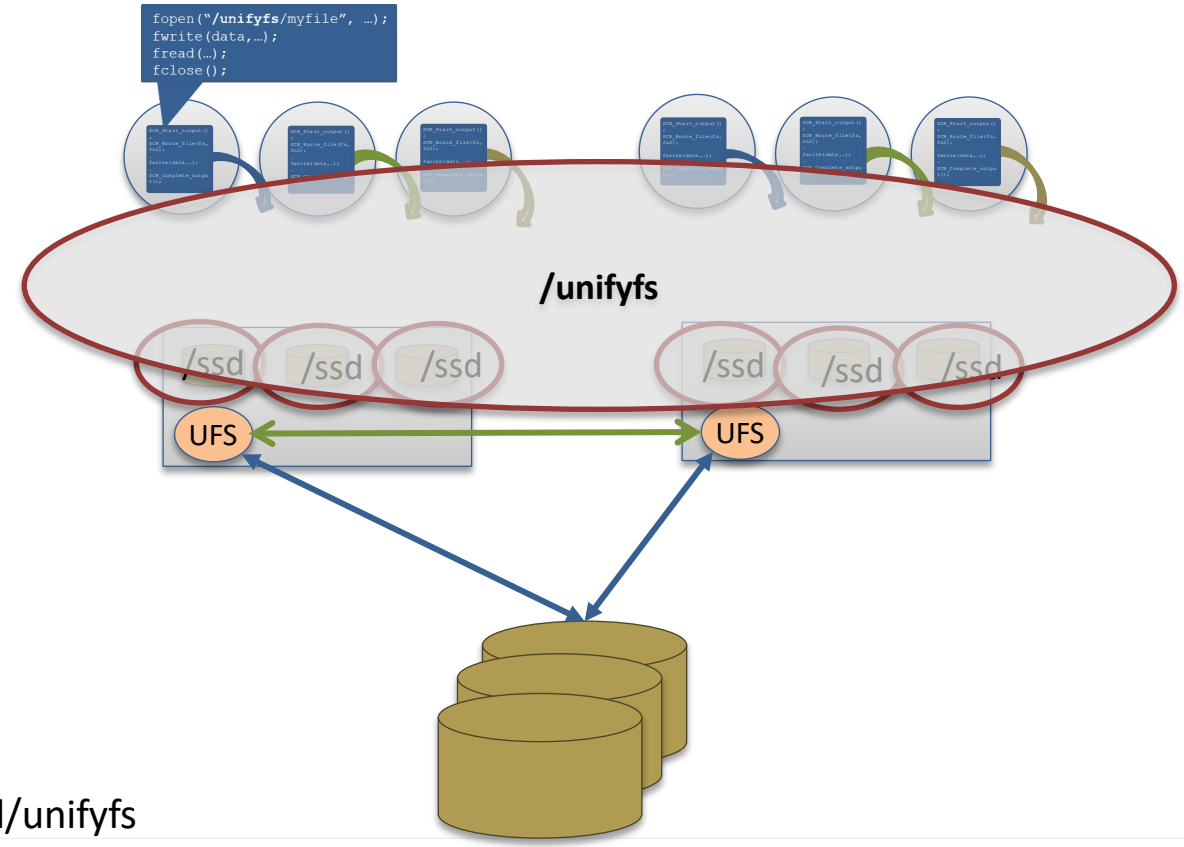
How can the Rabbits help I/O tools like UnifyFS?



"UnifyFS – A lightweight non-POSIX file system: https://github.com/llnl/unifyfs

How can the Rabbits help I/O tools like UnifyFS?





"UnifyFS – A file system for burst buffers: https://github.com/llnl/unifyfs

Applications can gradually adopt new strategies to support their I/O needs

1. Applications still just write/read to permanent Lustre (Will work but will not have scalable bandwidth compared to Rabbits.)

2. Applications allocate file systems on Rabbit using HPE-provided options, ~5x faster than permanent Lustre

3. Applications utilize software tools to provide even better performance and portability

4. Workflow, analytics frameworks, I/O middleware can be ported to the Rabbits and provide better performance and new capabilities to codes

Thank you!

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.