Crossroads Workflows

LANL, NERSC, SNL

LA-UR-19-20837

February 4, 2019

1 Introduction

The Department of Energy (DOE) compute facilities operate as resources for the National Nuclear Security Administration (NNSA) and Office of Science. In this document we describe a subset of scientific workflows which are run on current platforms. There is significant commonality to how Los Alamos National Laboratory (LANL), Sandia National Laboratory (SNL), Lawrence Livermore National Laboratory (LLNL) and the National Energy Research Scientific Computing Center (NERSC) operate and conduct scientific inquiry with supercomputers. In addition to presenting the commonality between the sites, we believe that opportunities exist to significantly optimize the operation of our facilities and more rapidly advance our scientific goals. In describing how our scientific workflows move from inception to realization we believe that we can enable significant improvements in how computer architectures support real scientific workflows.

Briefly, in order for a team of scientists to access the large supercomputers at these facilities, the team must be granted an allocation. Allocations last for a fixed amount of time (usually 6 or 12 months), and a large number of teams will have overlapping allocations ensuring that a large amount of work is available in the scheduling queue. During the allocation, the scientists will construct workflows as described in this document. Many simultaneous allocations may use identical workflows with different data and scientific goals.

1.1 Computational Allocation Workflows

Included below are *workflow diagrams* that demonstrate two classes of computational science workloads: large-scale scientific simulation and data-intensive workflows (which are further divided into large-scale Uncertainty Quantification (UQ) and High Throughput Computing (HTC)). The large rectangle at the top of the figures shows the data sets generated throughout the workflow and the timescale during which the data is retained. In particular, we have differentiated three timescales of interest: temporary, campaign, and forever. Data generated on temporary timescales includes checkpoints and analysis data sets that are produced by the application, but the domain scientist will eventually discard (usually at job completion, or when a later data processing step completes). The campaign timescale describes data that is generated and useful throughout the execution of the entire scientific workflow. Once the allocation is complete the scientist will discard the data. Finally, we consider the data retention period labeled forever. Forever timescale data will persist longer than the machine used to generate the data. This includes a small number of checkpoint data sets; however, this is primarily the analysis data sets generated by the application and re-processed by the user/scientist. Note that the data lifetime categories do not simply correspond to storage tiers.

The lower portion of the diagram shows the *phases* of the workflow. A workflow phase is executed as a series of parallel and/or serial jobs submitted to the batch scheduler and executed when sufficient resources exist to execute the submitted jobs. The scheduler may maintain several job scheduling queues to ensure that high-value jobs, such as jobs constructed to complete the interactive analysis phase of the workflow, execute immediately (or perhaps during business hours when scientists will be at their terminals). Most phases of the workflow require executing tens and possibly thousands of jobs, with each job continuing the progress of prior jobs.

The workflow phases also describe a dependency between the submission of jobs into the scheduling queues. However, the strict ordering requirements are particular to the specific workflow phases. In some cases, a phase cannot begin until all of the prior phases are complete. In other cases, a phase cannot begin until the first *set of jobs* from the prior phases are complete; but there is no requirement that the prior phases be complete, only that some number of jobs are complete. We describe these dependencies in detail in sections 1.2 and 1.3. A series of workflow phases may be instantiated multiple times during an allocation, with each instantiation called a *pipeline*.

Finally, we include in the workflow phase an icon for the initiator of the jobs submitted during the phase. The stick person icon indicates that a human is "in the loop" and performing job submissions. An instrument icon indicates that an external instrument or automated process is performing the job submissions. We use this icon to indicate that the computational resources are coupled in near real-time to an on-going experiment or orchestration framework. For example, compute jobs are automatically submitted to the scheduling queue when data is generated by LBNL's Advanced Light Source (ALS) beamlines.

1.2 Simulation Science Workflow Overview

Figure 1 is a Crossroadsworkflow diagram that depicts the phases and data processing common to most simulation science workflows. In the initial phase we see that the scientist leverages a small input data set, typically curated over multiple allocations, to construct a set of initial simulation conditions. Although phase S1 is only depicted once in the diagram, it may occur multiple times (thus initiating multiple pipelines) during a proposal team's allocation. Further, the construction of the initial state (e.g. a complex 3-dimensional mesh) is sometimes a memory-intensive and parallel process that may use a moderate number of processors or it may require the entire machine. The typical constraint for initial state creation is acquiring the amount of memory to store the initial state and write it to file. Not surprisingly then, the generated input data set is usually a large portion of the memory of the processors allocated to phase 1, often 80% of the memory available to phase 1 jobs is written to storage.

Phase S2 is generally the most computationally intensive portion of the simulation science workflow. Given the initial state/mesh and a small text-based input deck of settings and configurations, the physics calculations begin. The number of processors allocated to the job depends on the physics application's computational and memory requirements. However, it is not uncommon for a project to use a coarse 2-dimensional mesh to create pilot data before running a much larger 3-dimensional mesh that spans a much larger portion of the machine. Restart dumps, or checkpoints, are created with a frequency high enough to ensure that when a job fails or finishes, a subsequent job can continue execution where the previous job progress ended. The ideal checkpoint interval, gamma (Γ) multiplied by the job mean time to interrupt (JMTTI) in the above diagram, has been estimated carefully by Daly [1], and most simulation applications create checkpoints with at least this frequency.

Although a full system job might run for 24 hours and generate a checkpoint dump hourly, checkpoints are generally overwritten using an odd/even scheme. Thus, when a phase S2 job

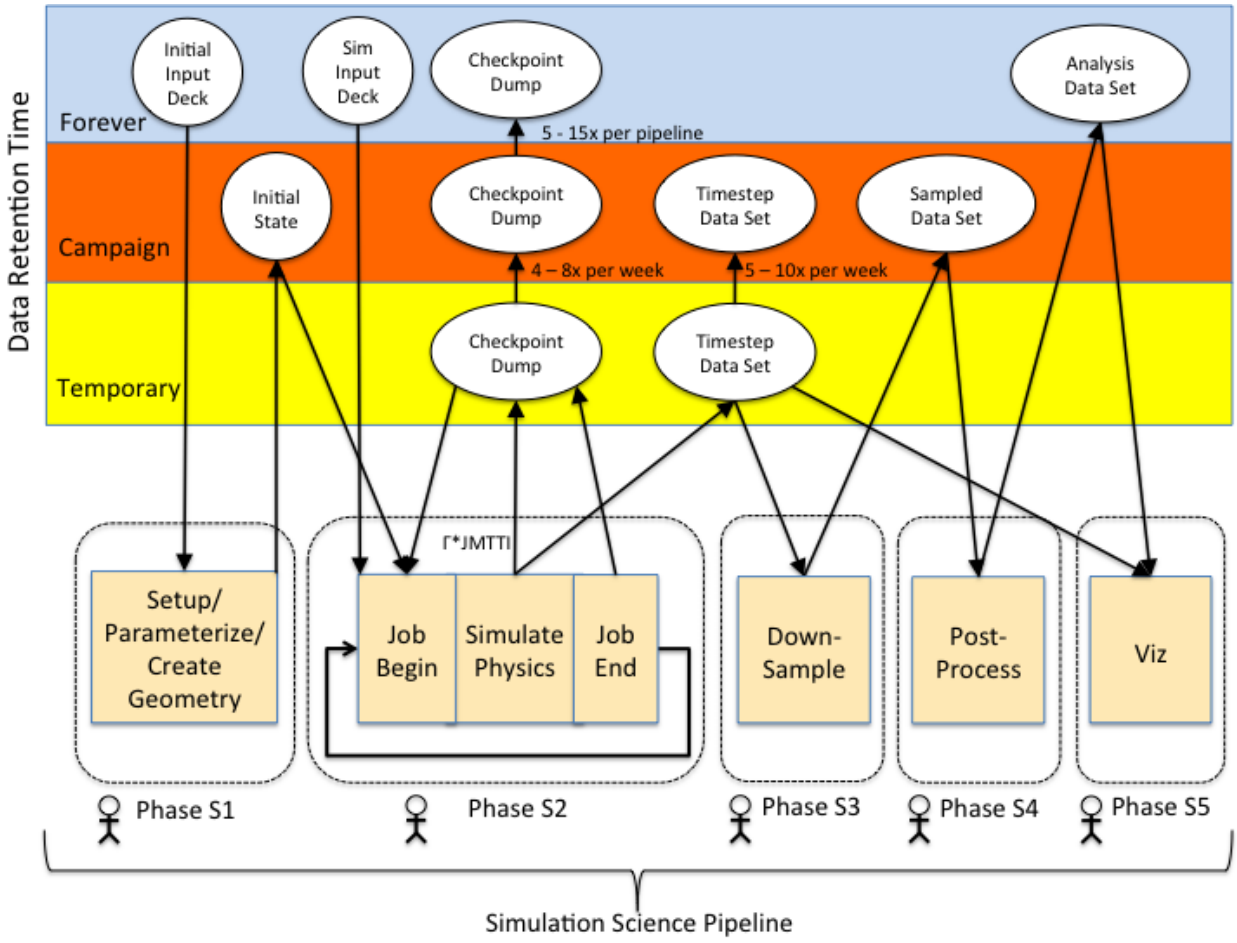

Figure 1: An example of an APEX simulation science workflow.

successfully terminates, 3 checkpoints should exist: an odd checkpoint, an even checkpoint, and an end-of-job checkpoint. As a simulation progresses over the course of several months, a large number of checkpoints will be generated and overwritten, and a large number of checkpoints will also be deleted, rather than retained. Over the course of an allocation, scientists will often retain 4 - 8 checkpoints for several weeks and possibly for a few months. This enables the scientist to rollback a week or month of computation in case an anomaly appears later in the simulation. In the above diagram we show that 4 - 8 checkpoints may be selected for retention each week over the course of an allocation. Additionally, checkpoints can often be analyzed for progress, and 5 - 15 checkpoints may be retained forever so that portions of the simulation results can be re-calculated later for verification purposes.

Phase S2 also results in the generation of analysis data sets. These data sets are generated at evenly spaced intervals in *simulated time*, but are typically not created uniformly throughout the life of the project or campaign. That is, the number of calculations required to construct analysis data sets varies over the duration of the simulation. The output data sets are often large, and composed of many files to enable multiple types of analysis and analysis tools. Again, to enable deep analysis of anomalies, a rolling window of un-sampled timestep data dumps are likely to be maintained during the campaign to enable scientists to examine previously generated results at full resolution.

Phase S3 shows the common task of down-sampling the analysis data. By their very nature, all analysis data sets are critical to the scientists, however, once a data set is down-sampled, the data at the original resolution is typically too large to be retained (often in the range of 5 - 20% of the total allocated memory per data set), rather as we described in Phase S2, only a sliding subset of the original timesteps are retained throughout the campaign to enable a rollback and examine anomalies. Down-sampling and analysis tasks in general are typically I/O intensive rather than compute intensive, and thus use the small numbers of compute nodes required to achieve the necessary memory footprint and adequate I/O bandwidth.

Phase S4 shows another common simulation science task, data post processing. Unlike phase S3, which is performed as the analysis data sets are generated, phase S4 requires all of the analysis data sets to exist prior to beginning phase S4. In this process the generated analysis data sets are formatted such that the regions of interest can be examined by visualization tools. This may involve concatenating portions of the data into separate data sets that favor visualization in isolation, or concatenating all of the data for visualization together. All post-processed data is vital to the scientist and will be retained beyond the lifetime of the campaign.

Phase S5 shows the phase where the scientist uses tools such as Ensight, ParaView and VisIt to visualize and analyze the simulated and processed timestep data. Interactive visualization typically occurs as soon as an analysis data set is generated, and then continues on through the life of the simulation and data processing phases.

A simulation science workflow pipeline is typically composed of each phase in the workflow. An example of a simulation pipeline could be a series of lower resolution simulations that identify an area of interest within the simulation space, followed by an extremely detailed, high-resolution simulation that better explores the regions of interest.

1.3 HTC and UQ Workflow Overview

In this section we describe HTC and UQ workflows. HTC workflows consist of a large number of low concurrency independent tasks and UQ workflows consist of an ensemble of simulation science workflows (see Section 1.2) with an extra analysis step over the entire ensemble. We group these workflows together because both classes of workflow generally use similar software infrastructure to efficiently execute and collect results from an ensemble of runs. Figure 2 shows the key features of these workflows. We will describe a UQ workflow with phases U1, U2 and U3 and a HTC workflow with phases H1, H2 and H3.

1.3.1 UQ

Phase U1 represents two sub-stages: 1) generating a single input mesh for all member of the ensemble and 2) generating a single input deck for the UQ software and different input decks for each ensemble member.

Phase U2 shows the action of running the ensemble of N simulations. A popular UQ framework for achieving this task is Dakota [2]. This framework supports launching an ensemble in a single batch job with a unique MPI communicator per member or in multiple batch jobs. Ensembles at Sandia currently consist of 1 to 100 members and 20 to 5000 MPI ranks per member. As before, checkpoint dumps are written periodically, however, there is now a checkpoint dump per ensemble member. This means there must be enough temporary storage to retain $3 \times N$ checkpoint dumps each on the order of several GB. Analysis data sets are also written throughout the run. A new workflow characteristic not previously covered is temporary files for communicating information between the ensemble and Dakota. These files provide a way to use pre-existing applications within

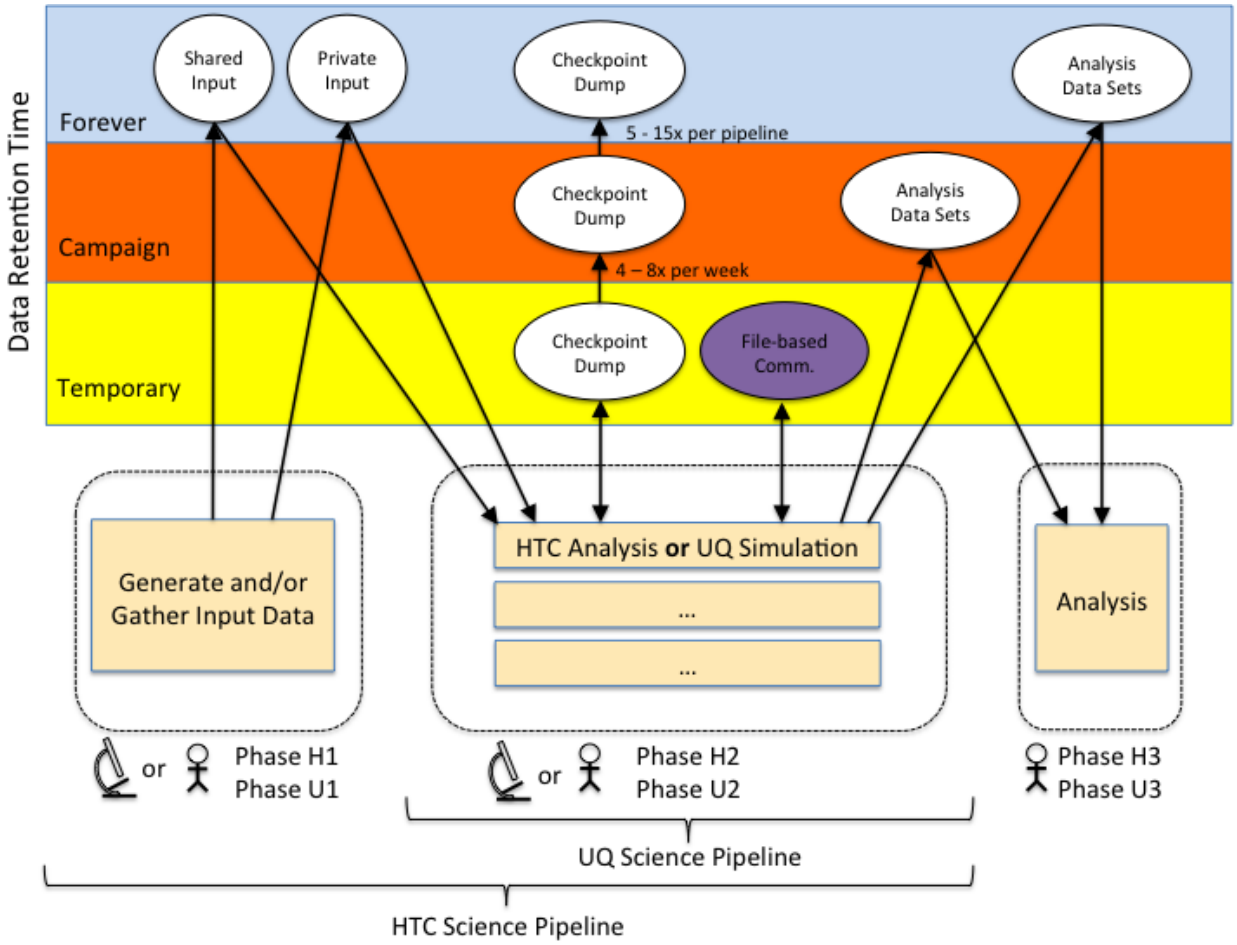

Figure 2: Example HTC and UQ workflow

the Dakota framework. To use this feature, a user must write glue code so that relevant information can be extracted from unmodified simulation data files and passed to Dakota UQ functions. Dakota also provides an API to support tighter coupling without temporary files, however, there will always be a requirement for loose coupling through files. In future, it is possible that the UQ will be embedded in the simulation to improve reuse of data and provide more opportunities to make use of fine-grained parallelism [3].

Phase U3 encapsulates phases 3-5 in Section 1.2. Typical tasks include analyzing the data files for parameter variation or statistical patterns.

1.3.2 HTC

Phase H1 shows the collection of data from a scientific experiment. The data from the experiment, labeled as Shared Input and Private Input, is sent from the experimental facility to compute facilities like NERSC, often over ESnet. The experiment is generally repeated a large number of times resulting in many input files and an aggregate size on the order of a PB.

Phase H2 shows the HTC analysis on the experimental input files. The high throughput nature arises because the same analysis is often repeated many times on different input data. For example, in the Sky Survey workflow the data analysis is run on small batches of about 60 images at a time

out of a collection of millions of images. Workflow Management Software (WMS) is used to manage the execution of the independent tasks. WMS includes custom user scripts and larger frameworks such as Swift, Fireworks and Tigres [4]. The number of concurrent tasks launched by the WMS depends on how many input data sets are ready for analysis and the number of cores used for each application.

The HTC analysis is often accomplished using a collection of different applications, where the output from one application is used as input for next application. This means that file-based communication happens both between dependent analysis applications and between the WMS and each analysis application. The applications in Phase H2 rarely have checkpoint/restart capability, however, resiliency is generally provided at a higher level by the WMS. For example, if Phase H2 consists of 3 applications and the 2nd application fails then the WMS will automatically restart the 2nd application from the initial state. This is possible because the applications in H2 often complete in less than a day (longer running applications are generally submitted to different queues, e.g. at NERSC there is a throughput queue ("thruput") which provides a job limit of 168 hours).

Phase H3 represents final analysis on the processed data. Many projects at NERSC provide web portals so that users can download the raw data from Phase H1, interact with analyzed data from Phase H2, and look at systematically collected provenance information. This phase may be repeated many times by different users over the course of many years. As such the input data, highest value analyzed data and provenance information must be saved "forever". It is therefore important that this data remain easy to access and available in a short enough time-period to avoid user frustration.

1.3.3 Commonalities

The features provided by Dakota and WMS include

- **Efficient scheduling** of an ensemble of independent runs on a supercomputer. The work per run is often variable and so dynamic load balancing strategies are implemented to improve overall throughput. Queue wait time can also be significant and so multiple runs are often bundled together to improve overall throughput.
- **Collecting results** from each run in the ensemble.
- **Recovery from failures** when some or all of the individual runs die.
- **Detailed monitoring** of ensemble progress and saving provenance information.

2 Mapping Workflows to Machine Architectures

In this section we describe ways to map the two example workflows to the Trinity [5] system architectures. We also discuss how alternative architectures featuring emerging technologies may provide opportunities for new, more efficient workflow mappings in the future.

2.1 Trinity Workflow Mapping

The Trinity systems provide a storage system which contains both SSD and HDD tiers. The SSDs are used to implement a Burst Buffer (BB) and the HDDs provide the storage for the Parallel File System (PFS). The product used to implement a BB on commodity SSDs is named Cray DataWarp. Details about DataWarp can be found in [6]. We briefly describe the functionality here.

The SSD storage in Cray DataWarp nodes is a scheduled resource that is dynamically assigned to compute jobs based on the capacity requested in the user batch script. The storage can be assigned for the lifetime of the compute job: a job reservation, or independently of the compute job: a persistent reservation. The raw storage can be configured in a scratch mode, where files are explicitly moved between tiers using a command line or C API, or a cache mode, where file pages move transparently between tiers according to a user-defined caching policy. Finally, the user may choose a striped or private access mode where files are either striped across multiple DataWarp nodes or exist in a single DataWarp node. Most combinations of storage lifetime, configuration and access mode are supported.

We also describe the longer-term storage interactions that result in data being written out of and restored into the Trinity PFS. These interactions can occur to both a disk-based capacity tier called an Inter-Campaign Storage System and an Archival Storage System using magnetic tape. These long-term storage tiers are currently accessed using nodes external to the supercomputer via file transfer agents.

2.1.1 Simulation Science Workflow

Phase S1 - Creating the Initial State Creating the initial state requires a small (typically not larger than 100MB), text-based input file describing relevant initial state parameters. This file may be staged into the BB and read into the tasks, or simply read directly from the PFS, into the phase S1 processes. The processes construct the initial state at a useful resolution (and partitioned across the processes), and write the resulting initial state into the parallel file system.

Phase S2 - Physics Simulation Prior to beginning a simulation job, the initial state and the most recent checkpoint will be migrated into the BB. The job will then begin by reading a small configuration file from the PFS, the initial state from the BB, and a checkpoint (if one exists) from the BB. At the optimal interval, the job will write checkpoints into the BB using the odd/even scheme described earlier. If the job ends prior to hitting a scheduler imposed wall-clock limit, the job will also write an end-of-job checkpoint into the BB. During the simulation execution, the BB control plane will copy checkpoint dumps (approximately one-third) from the BB into the PFS. At job completion, the most recently written checkpoint resident in the BB will be copied into the PFS.

Analysis data generated during phase S2, timestep data sets, will also be written into the BB (assuming the data set is large enough to benefit from using the BB). All analysis data sets will be migrated into the PFS, and the checkpoint used at the beginning of a simulation job cannot be deleted from the PFS until the analysis data generated by that job is safely stored in the PFS (due to the burst buffer favoring performance rather than reliability [7]).

If the phase 2 job resulted in an interesting checkpoint state, the checkpoint will also be copied from the PFS into longer-term storage (at LANL this is a disk-based capacity tier, while at NERSC it is a tape archive). Additionally, the most recent versions of full-resolution analysis data are likely to be stored in long-term storage.

Phase S3 - Downsampling Prior to executing a downsampling job, the full-resolution data sets to be sampled are fetched into the BB, and the downsampled data set is written into the BB, and then migrated into the PFS. Once the migration to PFS is complete, the down-sampled data set will be written to long-term storage, and removed from the PFS when space reclamation occurs.

Phase S4 - Post-processing Post-processing data sets may require the existence of a large number of the earlier data sets, thus to create or append to the post-processing data set, multiple data sets will be fetched into the BB prior to post-processing. When a large enough number of input data sets and the post-processed data set are in the BB, a job will execute to re-process the input data sets, and either append to or freshly create a post-processed data set. At completion of post-processing, the post-processed data set will be stored in both the PFS and long-term storage (often in replicated fashion) to ensure that the data is both protected and available for analysis.

Phase S5 - Visualization The post-processed data set should be in a format that enables efficient analysis and visualization. The visualization job will specify which portions of the data set must be fetched into the BB from the PFS to begin analysis (e.g. a small number of timesteps to begin with), and then as the scientist examines the data, additional portions of the data set will be read into memory directly from the PFS.

2.1.2 UQ and HTC Workflows

Phase U1 - Generation of UQ Input Data The U1 state creation is identical to the S1 initial state creation, with the exception that private input decks for each subsequent U2 pipeline phase are also created within the BB, and migrated into the PFS for improved reliability.

Phase U2 - UQ Simulation Phase U2 UQ simulations are virtually identical to phase S2 science simulations with the exception that one of the initial input sets that is initially read by a newly started simulation job is resident in a BB persistent reservation so multiple jobs may simultaneously read the data set. The generation of checkpoints and and analysis data sets occurs within BB job-level reservations, and uses the BB, PFS, and long-term storage resources in the same manner as a phase S2 simulation job.

Phase H1 - Gather Input Data The shared and private input data sets are placed according to the needs of phase H2. They are also archived into long-term storage. We describe placements appropriate for the bio-informatics and ALS workflows run at NERSC.

In bio-informatics workflows, a user generally copies reference genome lookup tables (a shared data set) from the globally accessible "project" file system to the local high performance "scratch" file system. The data set will be accessed many times within each workflow pipeline and also by multiple workflow pipelines. If all workflow pipelines will be run in a single batch job then staging the data set into the BB could be deferred until phase H2, otherwise, a dummy job could be submitted to stage the data set into a persistent reservation of BB storage accessible by multiple batch jobs.

In the ALS workflow, experimental data (a private data set) is automatically copied from the ALS to the "project" file system using NERSC Data Transfer Nodes (DTNs). A daemon running on a login node is used to automatically submit phase H2 jobs to a real-time queue when new data arrives. A future version of the daemon will also likely stage the experimental data into the BB, where, we anticipate use of a persistent reservation of BB storage to ensure that the BB can be used on-demand in real-time workflows. At the current time, the stage-in involves copying data from "project" to "scratch" and then "scratch" to the BB which adds latency to the near real-time workflow.

Phase H2 - HTC Analysis Input data not already resident in the BB will be staged into the BB at the start of the job. Intermediate data for file-based communication will remain on the BB unless the WMS provides restart capability. All final analysis data will be staged out to the PFS.

Phases U3 and H3 - Analysis Phases U3 and H3 encompass all of the storage interactions described for phases S3, S4, and S5, with the additional caveat that the number of pipelines per workflow for HTC and UQ are usually greater than the number of pipelines per simulation workflow, and while the amount of data analyzed is typically similar, the number of BB reservations, data set fetches and migrations, and discrete data sets analyzed is normally greater than in simulation science workflows and controlled by a workflow manager.

2.2 Emerging Technology Workflow Mapping

The authors expect disruptive technology to emerge for multiple technology areas within HPC system architectures. In this section, we describe opportunities for improved workflow mappings to hypothetical emerging technologies.

Large Memory Footprint Nodes Workflows may contain phases that have high aggregate memory footprint or high memory footprint per node requirements. These stages would benefit from nodes which provide large memory-to-compute ratios. For example, the concurrency of the initial state generation (phases S1 and U1) is often determined by the number of nodes needed to provide a given aggregate memory rather than a compute requirement. As such, it may be helpful to run this stage at a lower concurrency rather than wait in the queue for a large fraction of the machine to become available. This could increase utilization of the machine and allow the scientists to make early progress on the workflow. Large memory footprint nodes are also helpful when an application does not scale beyond a given number of processors, i.e. it is at the strong scaling limit, or does not support distributed memory parallelism. This often includes visualization of data sets (stages S5, H3 and U3) and community codes in HTC workflows (stage H2) that were never originally designed for supercomputers. The ability to statically or dynamically provision larger memory nodes could potentially make better use of machine resources and increase scientist productivity.

Reduced Tier Storage During the acquisition of Cori and Trinity, no examples of burst buffers existed upon which the authors could reliably base their expectations of the proposed burst buffer architecture. While the Cori and Trinity systems will be the first large-scale systems to deploy a burst buffer within the DOE computing complex, a great deal of smaller scale deployments and research has explored methods for leveraging burst buffers. In our analysis of the relevant scientific workflows and emerging storage media, it has become clear that opportunities may exist to dramatically accelerate scientific workflows by maintaining larger amounts of data in a high-performance storage tier. For example, if all of the data generated by HPC and HTC applications within the application's allocation can be stored in a fast storage tier and accessed from anywhere within the machine, there would be no need to provide a scratch storage system external to the platform. Instead, during an allocation and at allocation completion, newly created data could be migrated off of the machine's fast storage onto the Inter-Campaign Storage System (i.e. to a capacity tier that provides longer term retention) via some network pathway.

As the length of allocations is primarily a facility-specific and facility-controlled policy, if a machine with a single, extremely fast storage tier could also be provided with several weeks or months worth of capacity to satisfy a shortened allocation duration, scientists may be able to

realize a significant acceleration in their workflows executing on future architectures. Unfortunately, the exact amount of high-performance capacity required to accelerate the workflows is currently unknown; however, the authors believe that an adequately sized high-performance storage tier may significantly lessen the requirements for a scratch file system, provided that the workflows described in this document could be mapped onto that storage architecture (and its control-plane functionality).

3 Workflow Consistency Requirements

In this section we discuss the data consistency requirements imposed by the scientific workflows described in this paper. We believe that relaxing POSIX compliance is one way to accelerate scientific workflows. This could be implemented system-wide or per workflow through an API. We describe the consistency requirements of individual storage use cases below.

3.1 Analysis

Analysis workflow stages are generally run after the simulation or experiment has completed. This has the implication that data consistency could be delayed until the end of the data producing workflow stage using close-to-open consistency. This is the mechanism used in the Network File System (NFS) and has the effect that consistency is only guaranteed after returning from a close function call. We believe that these semantics are appropriate for interactive data analysis stages in which analysis tasks may be run on different compute nodes by many different users. For other data analysis tasks, e.g. automated post-processing workflow stages, it may be possible to further relax the consistency requirements by either reducing the scope of consistency to a subset of compute nodes or users, or further delaying the time until consistency. For example, global consistency is not needed if the analysis workflow stage is run in the same batch submission job as the data producing workflow stage and the lifetime of the analysis data is temporary. As another example, data consistency could be delayed until an analysis workflow stage submitted in a separate job has started (possibly using job dependencies).

As we look forward, we expect increased use of in-situ and in-transit analysis by scientific workflows. In this model data analysis tasks are run concurrently with the data producing task. This will result in different consistency requirements to the post-processing analysis described earlier. The ADIOS framework [8] already supports in-situ and in-transit analysis through Dataspaces, DIMES and Flexpath data transport methods. These transport methods ensure data consistency by synchronizing access to data stored in DRAM; files and POSIX consistency are replaced with data in memory and memory consistency. The same memory consistency ideas can of course be applied to data allocated in NVRAM memory.

3.2 Checkpoint/Restart

The consistency requirements of the checkpoint/restart use case depends on the method used to construct the checkpoint. Single file checkpoints with multiple writers require byte-level coherence during data generation, but do not require full POSIX-consistency. File-per-process checkpoint construction techniques require only close-to-open consistency during data generation. As checkpoints are never written after successful creation, once a checkpoint is created it is immutable, and is subsequently accessed read-only. Restarting a failed job immediately (within the same scheduler reservation) with a completed immutable checkpoint should be possible provided the job was allocated with spare nodes.

3.3 Input

Input files must support the same semantics as checkpoint files during creation, and are similarly immutable following creation, with the one difference being that we expect input files to be read repeatedly. Examples of input files include input decks and tabular data (such as equation of state databases).

3.4 Out-of-core

Out-of-core files are only used within an application. This is best supported by memory consistency.

3.5 Other

The data volumes for the use cases in this category are typically very small. However, the use cases often need strict POSIX compliance.

3.5.1 Diagnostics:

Most workflow stages produce various ASCII text files logging computation progress. Users often monitor these files using `tail -f` to ensure the workflow stage is progressing as expected. It is also common for time-stepping simulations to dump integrated quantities for each mesh variable. A user will often plot these quantities while the simulation is running to detect possible errors as early as possible. Finally, users often simply run `ls -trl` in the simulation directory to ensure the simulation has not deadlocked. All of these diagnostic and monitoring tests depend on POSIX access, but work acceptably with relaxed consistency models provided the file attributes are immediately updated.

In future we expect WMS to increase in popularity. This will simplify the monitoring process and potentially reduce the need for POSIX compliance because all interactions with monitoring data will happen in a well-defined way.

3.5.2 Semaphore files:

Semaphore or lock files are used to communicate information between independent processes through the file system. They are often used by experimental workflows to initiate the automated analysis of new experimental data. For example, the ALS workflow makes use of software named Spade which is run on computers at the ALS facility and Data Transfer Nodes (DTNs) at NERSC. The Spade process at ALS sends new beamline data followed by a semaphore file to NERSC using a file transfer tool. The Spade process at NERSC continually monitors for the presence of new semaphore files in the globally accessible project file system. POSIX compliance is necessary to ensure that the payload data is consistent as soon as the semaphore file arrives – however the actual capacity required for this use-case is quite small and could be satisfied with a small dedicated POSIX resource. The exact use of semaphore files in Spade is described in [9].

3.5.3 Legacy applications, install and run scripts, makefiles:

Finally legacy applications, install and run scripts, and makefiles often depend on POSIX access (including advisory locking), but due to the serial nature of these tasks, close-to-open consistency and similar models have typically proven adequate.

4 Crossroads and NERSC Workflow Summary

In this section we provide a table describing the characteristics of many of the identified Crossroads workflows as well as a set of NERSC workflows. The table values are a continuing work in progress, but represent the information we have currently collected to begin describing the scientific workflows for future supercomputers.

4.1 Workflows table

The rows in the table are described below and the table is shown in Figure 3.

- **Workflow**: The name of the workflow. The LANL workflows are EAP, LAP, Silverton and VPIC; the NERSC workflows are ALS (which represents reconstruction of microtomography beamline data generated by ALS beamline 8.3.2), CESM, GTS, HipMer, Materials (which includes Quantum Espresso and BerkeleyGW applications), MILC and Sky Survey; the SNL workflows are typical UQ workflows enabled by Dakota. LLNL workflows are still in the process of being captured. The computational characteristics of applications in the workflows are mostly captured by the Crossroadsbenchmarks [10].

- **Workflow Type**: The type of workflow. The values *Sim*, *UQ* and *HTC* represent simulation science, uncertainty quantification and high throughput computing workflows, respectively. All data for UQ workflows is given per ensemble member.

- **Workload Percentage**: The percentage of laboratory compute time consumed by this workflow. In the case of Tri-labs, each of the Tri-labs members are allocated one-third of the total available compute time. Note that the ALS and HipMer HTC workflows require much more storage capacity than the modest compute allocation would suggest.

- **Representative Workload Percentage**: The percentage of total facility compute time consumed by this workflow and other similar workflows. For example, the Gyrokinetic Tokamak Simulation (GTS) workflow is assumed to represent the compute and storage needs of all nuclear fusion Particle In Cell (PIC) workflows run at NERSC. Similarly, the EAP workflow is 60% of LANL's workflows, but as LANL is allocated only one-third of Cielo, EAP is only 20% of the total Cielo workload. Note that the NERSC representative workload percentage only adds to 48.2%. We crudely estimate that simulation science workflows account for 85% of the total NERSC workload; HTC workflows account for the remaining 15%.

- **Wall time (hours)**: The number of wall time hours it takes to run a workflow pipeline on Edison (NERSC) and Cielo (Tri-labs) using the routine number of cores. Recall that multiple pipelines are concurrently active during the allocation time period, and that allocations are 6 months in duration for Tri-labs, and 12 months in duration for NERSC.

- **Hero Run Cielo Cores**: The number of cores the code has been demonstrated to scale to on Cielo. This number is useful for determining the scale of the absolute largest data sets that have been generated on Cielo, and understanding how they may grow for a larger machine.

- **Routine Number of Cielo Cores**: The number of cores needed to run *today's* average problem size on Cielo. A workflow often uses multiple concurrencies, however, we simplify things by just showing the number of cores used for the bulk of the computation. Note that occasional "hero" runs may require much higher concurrencies, e.g. HipMer will be used to

assemble a small number of genomic data sets an order of magnitude larger than has been shown in the table.

- **Number of Workflow Pipelines per Allocation**: The number of pipelines run for each workflow during a single DOE allocation. In our analysis we assume that a single DOE allocation at NERSC is available for 12 months, while a Tri-labs allocation is 6 months in duration. The data for UQ workflows is presented as $N \times M$, where N is the number of ensembles and M is the number of ensemble members.
- **Anticipated Increase in Problem Size by 2020**: The anticipated growth of data sets by 2020. For example, if a simulation is expected to use 8x more grid points by 2020 then this number will be 8. This number could be very low if the project plan is to incorporate additional physics or more detailed physics models into a simulation rather than increase mesh resolution.
- **Anticipated Increase in Number of Workflow Pipelines per allocation by 2020**: The anticipated growth in throughput by 2020. For example, if the number of simulations in a UQ ensemble is expected to grow by 2x then this number will be 2. This is an extremely important number to quantify the expected growth of experimental workflows run at NERSC.
- **Storage APIs**: The storage APIs used during the workflow. POSIX means that read and write system calls were used (though there is no requirement for POSIX consistency), HDF5, NetCDF and pNetCDF indicate use of a high-level I/O library, and MPI-IO indicates use of the MPI-IO interface.
- **Routine Number of Analysis Datasets**: The number of analysis data sets typically generated per workflow pipeline. This number is wholly dependent upon the problem being studied by the user, and thus is an approximation averaged across many pipelines. Under no circumstances should this be considered a maximum value for a pipeline.
- **Routine Number of Analysis Files**: The actual number of files generated per workflow pipeline. This value may be less than the number of data sets if multiple application data sets are appended to the same file. It may be larger if an application performs file per process output.
- **Checkpoint Style**: A description of the number of checkpoint files in each checkpoint data set. N to 1 results in a single checkpoint file, N to N results in a checkpoint file per process, N to M means the number of checkpoint files produced is configurable, and explicit use of a value for M means that M files are always produced. The checkpoint data set size can be obtained by dividing the temporary checkpoint data retained row by three (see the odd, even, end of job checkpoint strategy described in Section 1.2)
- **Files Accessed/Created per Pipeline**: The number of files created and/or accessed during a routine workflow pipeline. Individual workflow pipelines may use different numbers of files based on the scale and problem space. This attempts to capture the number of files routinely created and/or accessed during a single representative pipeline. It does not include the number of checkpoint files produced for resilience purposes (because this would change according to JMTTI and machine bandwidth).
- **Data description (95% of storage volume)**: A description of how the majority of application data is organized in the file. This can include multidimensional arrays and variable length records.

- **Amount of data retained**: The capacity needed to store data used and produced by a single workflow pipeline. The capacity is given in units of percentage of system memory on Cielo. Use Equation 1 to convert the capacity from a percentage of system memory (Cm) to GiB (Cg) for a *single* problem size run on Nc cores. Cielo has 2 GiB of DRAM memory per core (M).

$$Cg = \frac{Cm}{100} \times M \times Nc \tag{1}$$

  The duration that data must be stored is categorized into 3 groups: during a single pipeline, during a campaign, and forever. This is then further subdivided into different storage use cases: checkpoint/restart, analysis, read-only input, and out-of-core. For simplicity, we assume *all* data generated during the workflow initially passes through the burst buffer.

  - **Temporary**: The capacity needed to store data which only needs to exist for the duration of a single workflow pipeline. Examples include temporary checkpoint files for resilience and restarting a simulation, temporary files for exchanging data between different workflow stages, and applications using out-of-core algorithms.
  - **Campaign**: The capacity needed to store data which must exist for the duration of a workflow allocation. Examples include additional files for ad-hoc or unplanned data analysis.
  - **Forever**: The capacity needed to store data which must be saved forever. This is the long-term storage requirement of the data used and produced in a single DOE allocation.

  The UQ data decorated with '*' is shared between ensemble members.

	Crosssroads workload									NERSC workload							
		LANL			SNL			LLNL									
Workflow	EAP	LAP	Silverton	VPIC	Dakota A	Dakota S	pF3D	R15	R20	ALS	CESM	GTS	HipMer	Materials	MILC	Sky Survey
Workflow type	Sim	Sim	Sim	Sim	Sim/UQ	UQ	Sim	UQ	SIM	HTC	Sim	Sim	HTC	Sim	Sim	HTC
Workload percentage	60	5	15	10	10	10	10	20	30	0.05	1.31	2.43	0.03	0.85	2.09	0.17
Representative workload percentage	20	2	5	3	3	3	3	6	10	0.10	6.00	6.00	0.10	19.00	11.00	6.00
Wall time (hours)	262.4	64.0	128.0	157.2	100.0	100.0	2304.0	76.8		0.50	240.00	48.00	4.00	41.70	6.00	4.00
Hero Run Cielo Cores	65536	32768	131072	70000	131072	65536				100	30000	132096	15360	12000	131072	24
Routine Number of Cielo cores	16384	4096	32768	30000	8192	4096	2048	4096	1024	100	8000	16512	960	2400	4096	24
Number of workflow pipelines per allocation	30	10	6	4	10 x 100	30 x 300	2	100		10760	8	36	100	100	1000	21000
Anticipated increase in problem size by 2020	1 to 2x	1 to 2x	1 to 2x	1 to 2x	4 to 8x	1.25 to 1.5x		1x		1x	4x	5x	1x	10 to 25x	34x	1x
Anticipated increase in workflow pipelines per allocation by 2020	1x	1x	1x	1x	2 to 8x	2 to 4x		10x		5x	3x	3x	50x	1x	1x	2.38
Storage APIs	POSIX	POSIX	POSIX	POSIX	HDF5 or NetCDF	HDF5 or NetCDF	POSIX	POSIX	POSIX	HDF5 and POSIX	NetCDF or pNetCDF	HDF5	MPI-IO and POSIX	HDF5 and POSIX	POSIX	POSIX
Routine number of analysis datasets	100	100	225	150						14	251635	2400	5	5	23	62
Routine number of analysis files										19464	20797	1328	201602	5	661	205
Checkpoint style	N to 1	N to 1	N to 1	N to N	N to N	N to N	N to N	N to N	N to N		N to 8	N to 64				
Files accessed/created per pipeline										23563	22717	1328	201604	6	680	385
Data description (95% of storage volume)										Dense 3D arrays	Dense 2D and 3D arrays	Dense 2D array	Variable len. ASCII records	Dense 2D arrays	Variable len. binary records	Dense 2D array
Data retained per Pipeline (percentage of memory)	268.00	510.00	463.00	360.25	5.87	32.54				285.63	1054.02	15.45	100.54	27.08	73.88	11.57
Temporary	30.00	75.00	285.00	222.75	0.02	30.00				147.68	0.32	0.67	34.34	4.17	73.28	2.16
Analysis			5.00	200.00						126.57			34.34	4.17		2.16
Checkpoint	30.00	75.00	210.00	18.75	0.02	30.00					0.32	0.67				
Input			70.00	5.00						21.10						
Out-of-core															73.28	
Campaign	170.00	170.00	100.00	115.00	2.00					21.10					0.44	
Analysis	80.00	70.00	30.00	60.00	2.00					21.10					0.44	
Checkpoint	90.00	100.00	70.00	50.00												
Input				5.00												
Forever	68.00	265.00	78.00	22.50	3.85	2.54				116.84	1053.70	14.78	66.20	22.92	0.17	9.41
Analysis	25.00	250.00	8.00	10.00	0.85	2.04				106.29	1027.96	14.78	0.36	22.92	0.08	0.62
Checkpoint	40.00	10.00	70.00	12.50							25.74					
Input	3.00	5.00			3.00*	0.50*				10.55			65.83	0.00	0.08	8.79

Figure 3: Crossroads and NERSC workflow summary table. This data may change as we continue our discussions with domain scientists

4.2 Example uses of table data

Q1. How much temporary storage is needed to run a typical VPIC workflow pipeline?

A single VPIC simulation job requires 222.75% of memory as temporary storage. At 30,000 cores the capacity required to store 222.75% of memory is 130.51 TiB.

Q2. How much storage is used for productive I/O (i.e. not defensive I/O) in a typical VPIC workflow pipeline?

The total storage used for checkpoint/restart for resilience purposes is 18.75% of memory at 30,000 cores. This means that 200.10 TiB (211.08 TiB - 10.99 TiB) of storage is used for productive I/O.

Q3. How much long-term storage is needed to archive all data from the ALS workflow per allocation?

There are 10,760 workflow pipelines executed per year and the data saved forever corresponds to 116.84% of memory at 100 cores. This is equal to 233.68 GiB of data per workflow pipeline. This adds up to 2.40 PiB of data per campaign and consists of 0.22 PiB of beamline data and 2.18 PiB of analyzed data.

Q4. How is the Sky Survey workflow expected to change by 2020?

The growth in problem size is 1.00x and growth in throughput is 2.38x. The problem size is expected to remain constant because the resolution of the sky images will not change. This is because the resolution of Dark Energy Camera, which is used to collect the images, will not change. The throughput is increasing because the number of images is expected to grow from 1.26 million to 3 million.

Q5. How does the storage need of the HipMer workflow change depending on whether multiple workflow pipelines are scheduled simultaneously or consecutively?

We assume that we want to run ten workflow pipelines. If the workflow pipelines are run simultaneously then the storage requirement is 100.54% of memory at 960 cores all multiplied by 10 workflow pipelines. This is 18.85 TiB. If, on the other hand, the workflow pipelines are run consecutively then the storage requirement is 13.05 TiB (12.41 TiB for the long-term data products and 0.64 TiB as temporary scratch space for 1 pipeline at a time). The capacity difference could be significant if the user must schedule a fixed capacity of fast storage.

Q6. How much forever storage is needed for a typical Dakota-A UQ ensemble?

The example Dakota-A workflow uses an ensemble of 100 ensemble members. The data saved forever includes a single input mesh and multiple analysis files. The same input mesh is read by all ensemble members and has size 0.48 TiB. Each ensemble member also generates 0.136 TiB of analysis data. This adds up to 14.1 TiB (0.48 TiB + (0.136 TiB $\times$ 100))

5 Closing

From the above described workflows, the Crossroadsteam has mined many of the requirements used to draft our requirements for procuring compute resources. A careful reader of these workflows will likely find additional insights and generalizations that offer opportunities to improve workflow performance on APEX machines.

A Supporting Data

The following section contains the file system distributions from LANL's two Trinity file systems on August 15, 2018. This file distribution changes over time depending on the workflows currently active, but this is a valid starting point for understanding LANL's Trinity file system usage.

A.1 File Counts

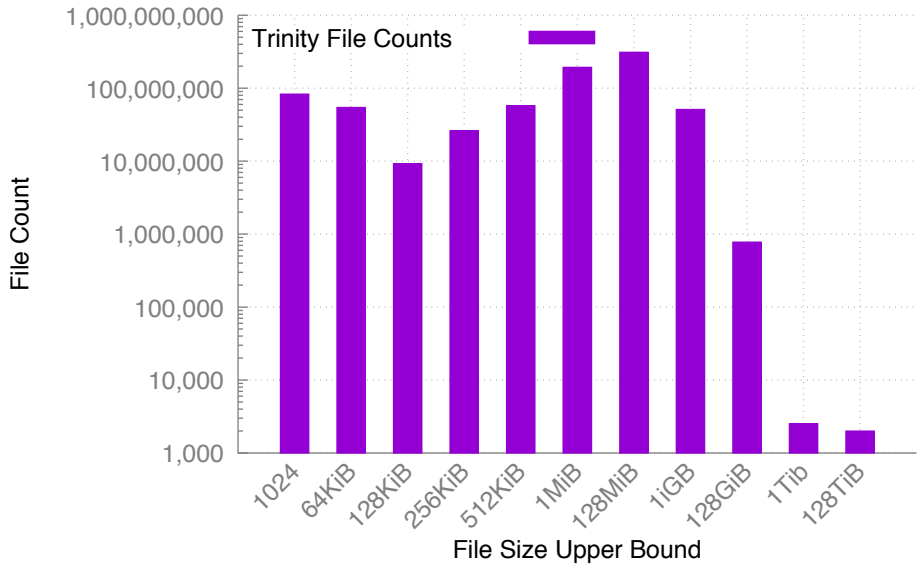

Figure 4: File counts binned by file size.

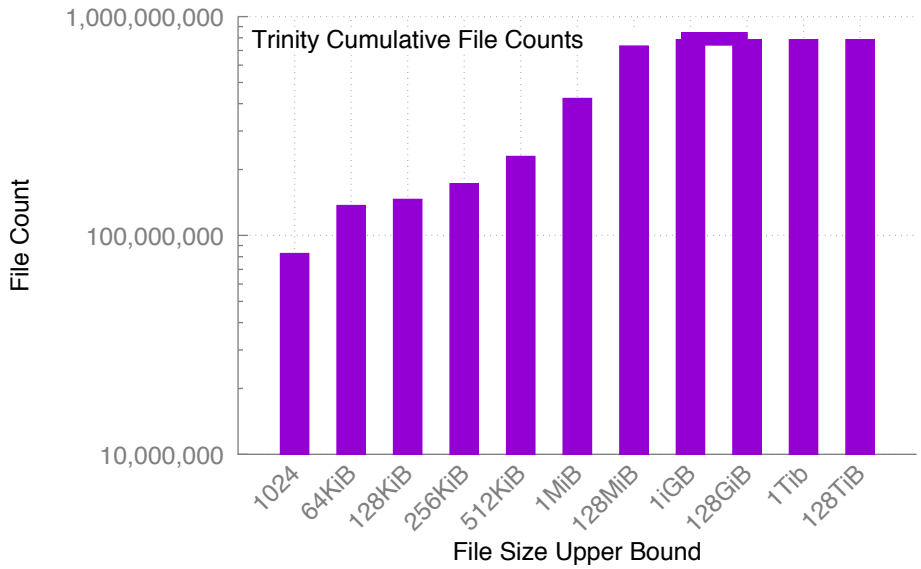

Figure 5: Cumulative file counts binned by file size.

In Figure 4 we see the total count of files binned by the file size. In the first bucket we see that the file count for files 1KB or smaller is less than 100,000,000 while the number of files sized between 1GiB and 128GiB is slightly less than 1,000,000. Although captured in a 1TiB to 128TiB bin during this data collection, the largest files at the time of this file system tree walk was approximately 38TiB. As efficiency on Trinity is further improved we will expect significantly larger files.

Figure 5 shows the same data but the bin counts are cumulative. That is the last bucket includes every file in the file system that is smaller than 128TiB. This count is approximately 751,000,000 total files across the two 38PiB file systems. Although not visualized here, the total number of file system directories was approximately 70,000,000. By examining the tick marks on the graph we can see that approximately 300,000,000 files are 1MiB or smaller.

A.2 File System Capacity Usage

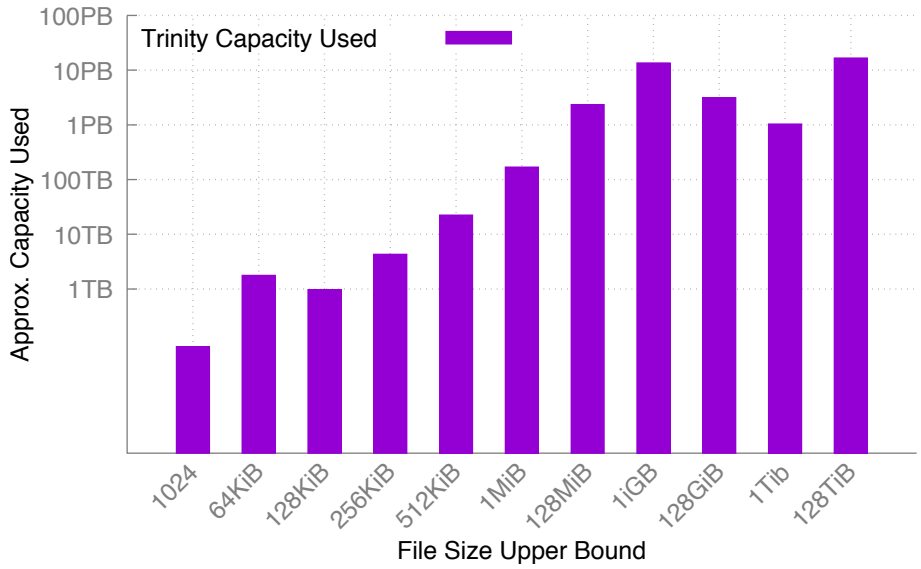

Figure 6: File system capacity used binned by the file size.

In Figures 6 and 7 we use the same X-axis respectively, but the y-axis is approximate capacity used rather than file counts. We use the term approximate because we recognize that the Lustre file system used by Trinity may not report the exact number of Bytes allocated to a file in some cases. In figure 6 we show the capacity used for just files within that bucket. So files sized between 1TiB and 128TiB use approximately 10PB of disk capacity, or approximately 20% of the 54 total PBs in use when this data was collected.

The cumulative capacity provided in Figure 7 may be useful to show the effects of packing small files into inodes or onto alternative storage partitions, especially when combined with the cumulative file count data in Figure 5. For example, if all files 512KiB or smaller were moved to an alternative storage location the total capacity required would be less than 50TB and the number of files stored would be between 200 and 300 million. At present the effects of these types of optimizations are not well studied for large HPC platforms, but may offer significant benefits for some storage systems.

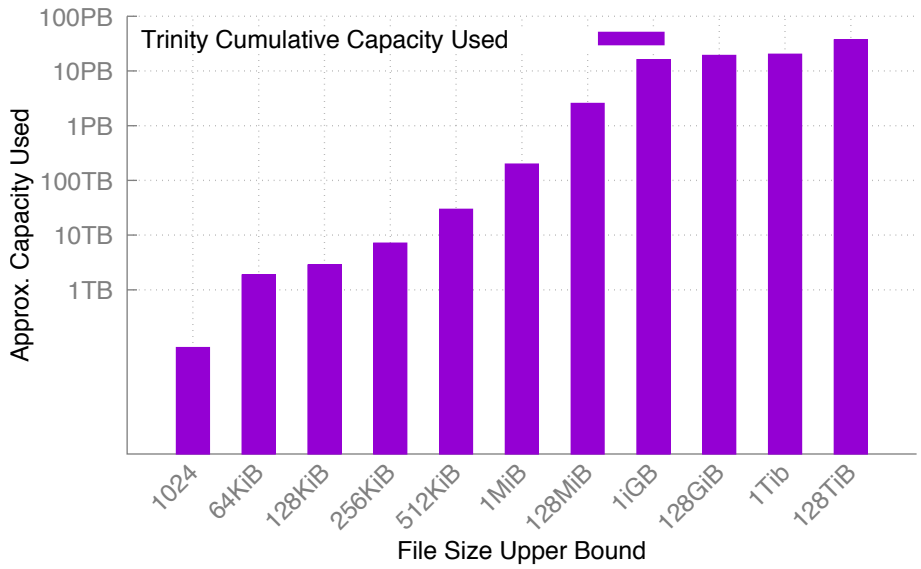

Figure 7: Cumulative file system capacity used binned by the file size.

B Glossary of Terms

To better understand this document we provide the following glossary of terms used within this document.

Allocation A fixed quantity of computational time granted to a single project to use in the execution of their workflow on the capability-class computer. An allocation is used by running the 10s - 1000s of parallel jobs needed to satisfy a workflow.

Analysis Dump Data written from a parallel application for the express purpose of scientific analysis or visualization. The data is typically domain dependent and is typically analyzed in conjunction with other analysis dumps from the same parallel application (e.g. each dump may represent the system state at each second of the simulated system, with a total of 200 seconds simulated during a campaign).

Campaign A Tri-labs centric method of aligning the start and end times of a set of granted allocations. For example, during Cielo Campaign 9, approximately 50 6-month allocations were started and completed. NERSC does not align allocations, therefore there is no NERSC equivalent to a campaign.

Checkpoint Dump Data written from a parallel application that allows an interrupted application to resume from the progress made up to the time of the data creation. Simulations read the checkpoint dump at the beginning of a job in order to leverage prior progress toward simulation completion.

Data Retention Time Describes the time interval during which the data must be made durable. For example, an application that produces checkpoint dumps requires that those dumps must exist long enough for the application to progress far enough to create a subsequent checkpoint dump, or the application completes successfully. In the described workflows we describe three relevant retention times: immediate, meaning the data is temporary in nature and will be

replaced with more up to date data as soon as possible; campaign, meaning the data is valuable throughout the allocation; and forever, indicating the data will outlive the life of the machine.

Gamma The optimal checkpointing interval for a computational job parameterized by both the time to create a checkpoint and the JMTTI. If JMTTI is 24 hours, Gamma=0.1 corresponds to an optimal compute interval between checkpoints of 2.4 hours.

Job A computational job runs on the supercomputer and consumes a portion of the allocation of compute time.

Pipeline A pipeline is an instantiation across some or all of the phases in a workflow. A scientific simulation workflow that performs 3 small-scale 2-d simulations with analysis followed by 1 large 3-d simulation with analysis would result in 4 total pipelines. For high-throughput workflows a pipeline would be limited to ensembles of simulation runs, or collections of analysis tasks. For example, a high-throughput climate workflow may use 10 initial sets of weather to seed 10 independent simulation runs, followed by a single analysis phase that averages the results from all 10 simulations. The climate workflow would have a total of 10 pipelines.

Proposal A research proposal is a document that provides a detailed description of the manner in which the supercomputer will be used to achieve scientific goals.

Simulation An execution of a parallel application that simulates physical phenomena. Simulations may take anywhere from hours to weeks or months to finish. A simulation will typically generate a series of restart dumps that allow the simulation to be composed into a series of jobs that execute on the supercomputer.

Tri-labs A designator for describing the three research labs primarily tasked with supporting nuclear safety: Lawrence Livermore National Laboratory (LLNL), Los Alamos National Laboratory (LANL), and Sandia National Laboratory (SNL). In this document, Tri-labs is used as a designation to describe the collection of users of one of of the proposed machines. On the Tri-labs machine, each of the three Tri-labs sites receives an identically sized machine allocation per campaign.

Workflow A description of the dependencies and frequencies of a series of inter-related computational jobs. A proposal typically describes the constituent workflows required to support the proposed scientific inquiry.

Workload A description of how workflows are executed on the supercomputer. The workflows may be co-scheduled or interleaved to the degree possible within the workflow constraints.

References

[1] J.T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303 – 312, 2006.

[2] B.M. Adams et al. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.3 user's manual. Technical Report SAND2014-4633, November 2015. Available at `https://dakota.sandia.gov/sites/default/files/docs/6.3/Users-6.3.0.pdf`.

[3] E.T. Phipps, M. D'Elia, H.C. Edwards, M. Hoemmen, J.J. Hu, and S. Rajamanickam. Embedded ensemble propagation for improving performance, portability and scalability of uncertainty quantification on emerging computational architectures. *CoRR*, abs/1511.03703, 2015.

[4] Data & Analytics - Workflow Tools. `https://www.nersc.gov/users/data-analytics/workflow-tools/`; accessed 13 January 2016.

[5] The Trinity Advanced Technology System. `http://www.lanl.gov/projects/trinity/`; accessed 13 January 2016.

[6] Cray. *DataWarp User Guide S-2558-5204*, September 2015. Available at `http://docs.cray.com/books/S-2558-5204/S-2558-5204.pdf`.

[7] John Bent, Brad Settlemyer, Nathan DeBardeleben, Sorin Faibish, Uday Gupta, Dennis Ting, and Percy Tzelnic. On the non-suitability of non-volatility. *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)*, 2015.

[8] ORNL. *ADIOS 1.9 User's Manual*, July 2015. Available at `http://users.nccs.gov/~pnorbert/ADIOS-UsersManual-1.9.0.pdf`.

[9] S. Patton, T. Samak, C.E. Tull, and C. Mackenzie. Spade: Decentralized orchestration of data movement and warehousing for physics experiments. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1014–1019, May 2015.

[10] Benchmark Distribution & Run Rules. `http://www.lanl.gov/projects/crossroads/benchmarks-performance-analysis.php`; accessed 18 July 2018.