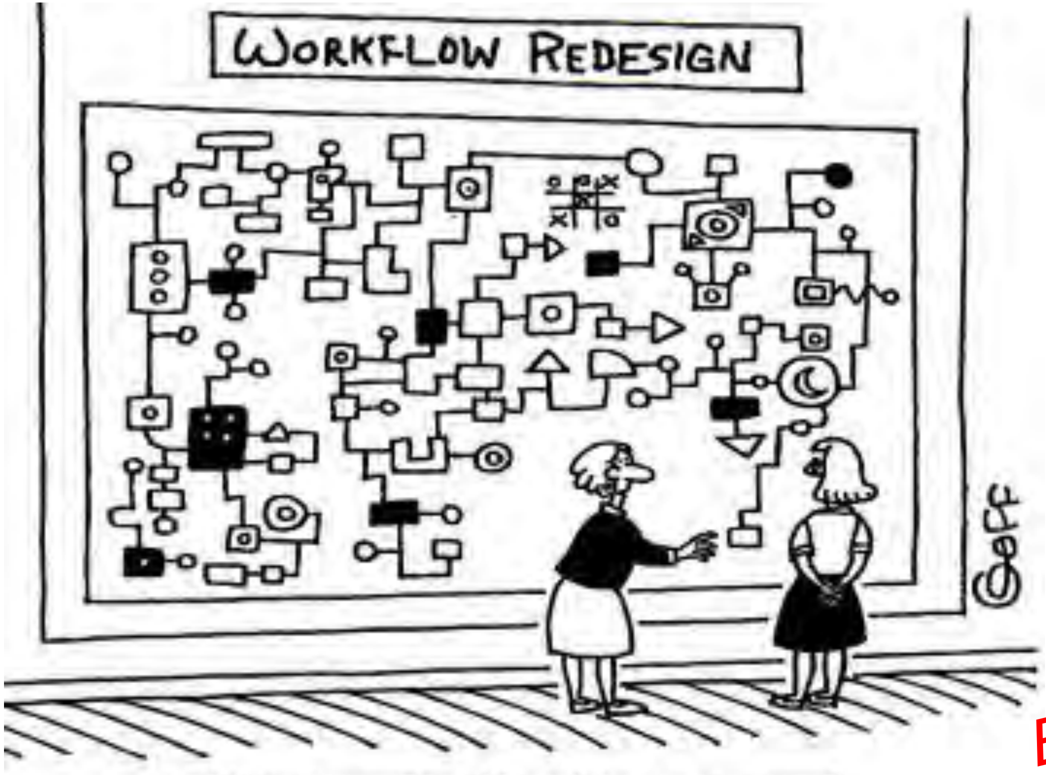
Workflow Analysis – An Approach to Characterize Application and System Needs

MSST 2016

Dave Montoya

May 3, 2016



"And this is where our ED workflow redesign team went insane."

Why are we discussing workflow?

Exascale is driving tighter integration!

Economics are changing the landscape!

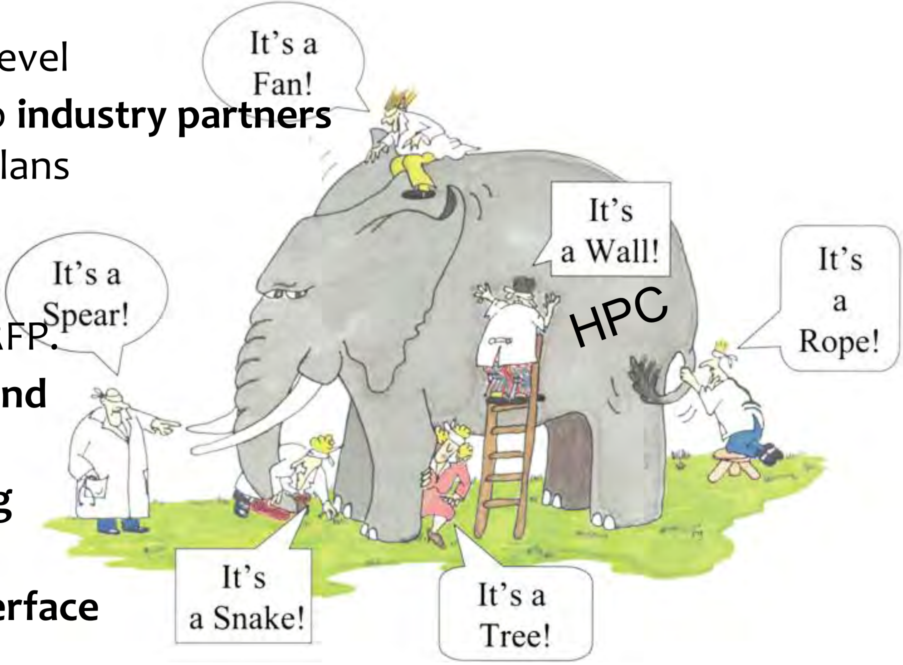
Premise:

Everything is a workflow!
- and interact with other workflows

Initial Focus - The Application Stack (as it pertains to the Data Stack)

However there are others:
- Data Stack - detail
- System Stack
- others

What are the Application Workflows?



- Begin to understand what we are doing at a larger level
- Providing computational and data use workflows to **industry partners** working toward developing exascale architecture plans
  - Fast Forward/Design forward projects
- Provide use cases to provide **vendors** for platform purchasing efforts. Cray, IBM, others. NNSA ATS-3 RFP.
- Provide a taxonomy for **code development teams and users** to discuss aspects of system
- Provide map of **use cases for production computing groups** to better tune the environment
- Form a base understanding for development of **interface points** across the HPC environment
- **Documenting** how a system works for understanding and training
- Establish **a map** for workflow performance assessment efforts
- Etc. - There are others

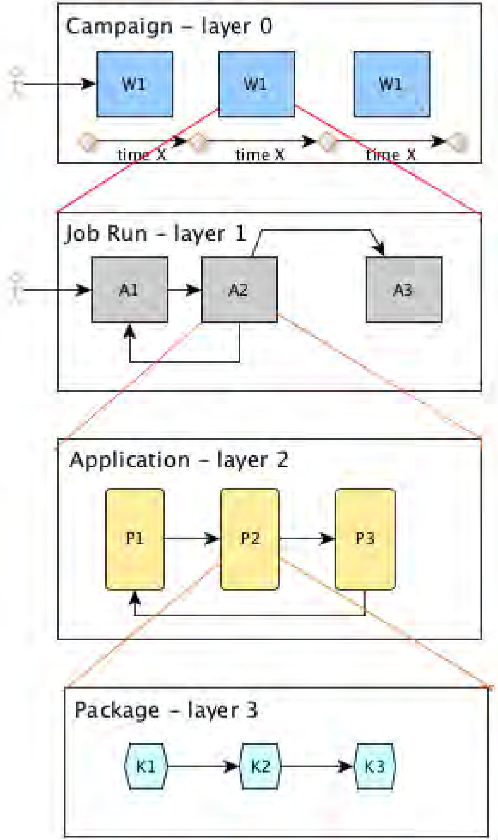
Workflow Layers within the Application Execution Stack

Layer 0 – **Campaign / Pipeline layer**. Process through time of repeated Job Run layer jobs with changes to approach, physics and data needs as a campaign or project is completed. Working through phases.

Layer 1 – **Job Run layer**. Application to application that constitute a suite job run series, which may include closely coupled applications and decoupled ones that provide an end-to-end repeatable process with differing input parameters. This is where there is user and system interaction, constructed to find an answer to a specific science question. Layer 0 and 1 are from the perspective of a end user.

Layer 2 – **Application layer**. Within an application that may include one or more packages with differing computational and data requirements. Interacts across memory hierarchy to archival targets. The subcomponents of an application {P1..Pn} are meant to model various aspects of the physics; Layer 1 and 2 are the part of the workflow that incorporates the viewpoint of the scientist.

Layer 3 – **Package layer**. This describes the algorithm implementation and processing of kernels within a package and associated interaction with various levels of memory, cache levels and the overall underlying platform. This layer is the domain of the computer scientist and is where the software and hardware first interact.



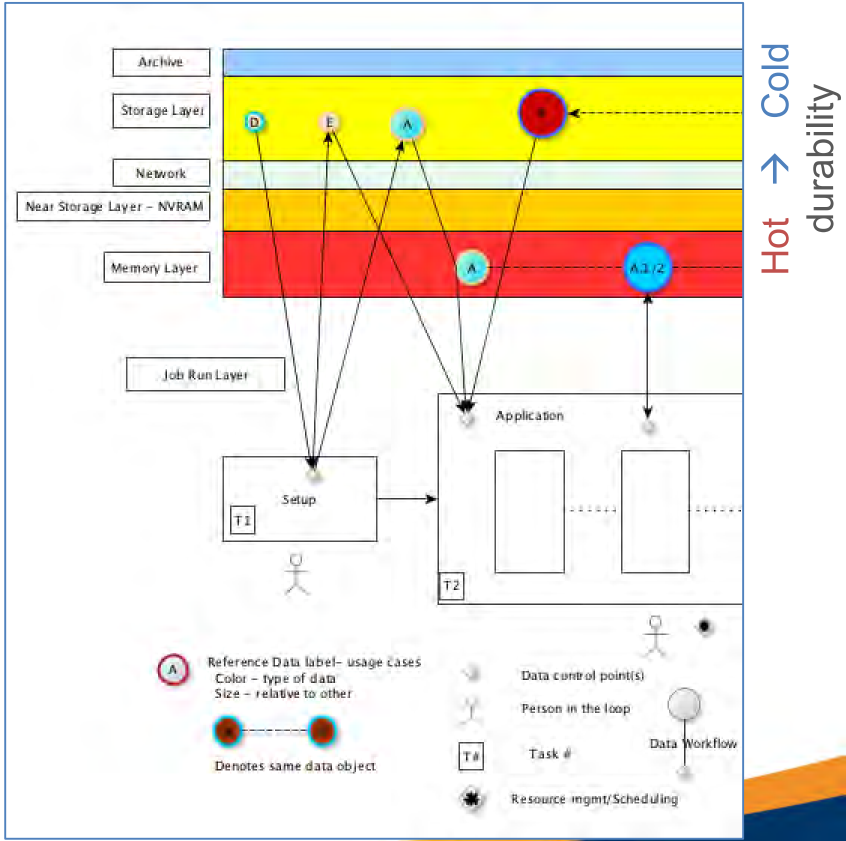
Started here

The Taxonomy

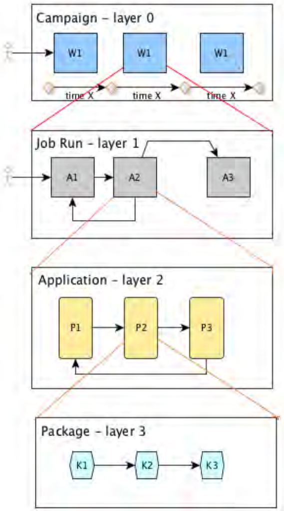
What is the Taxonomy?

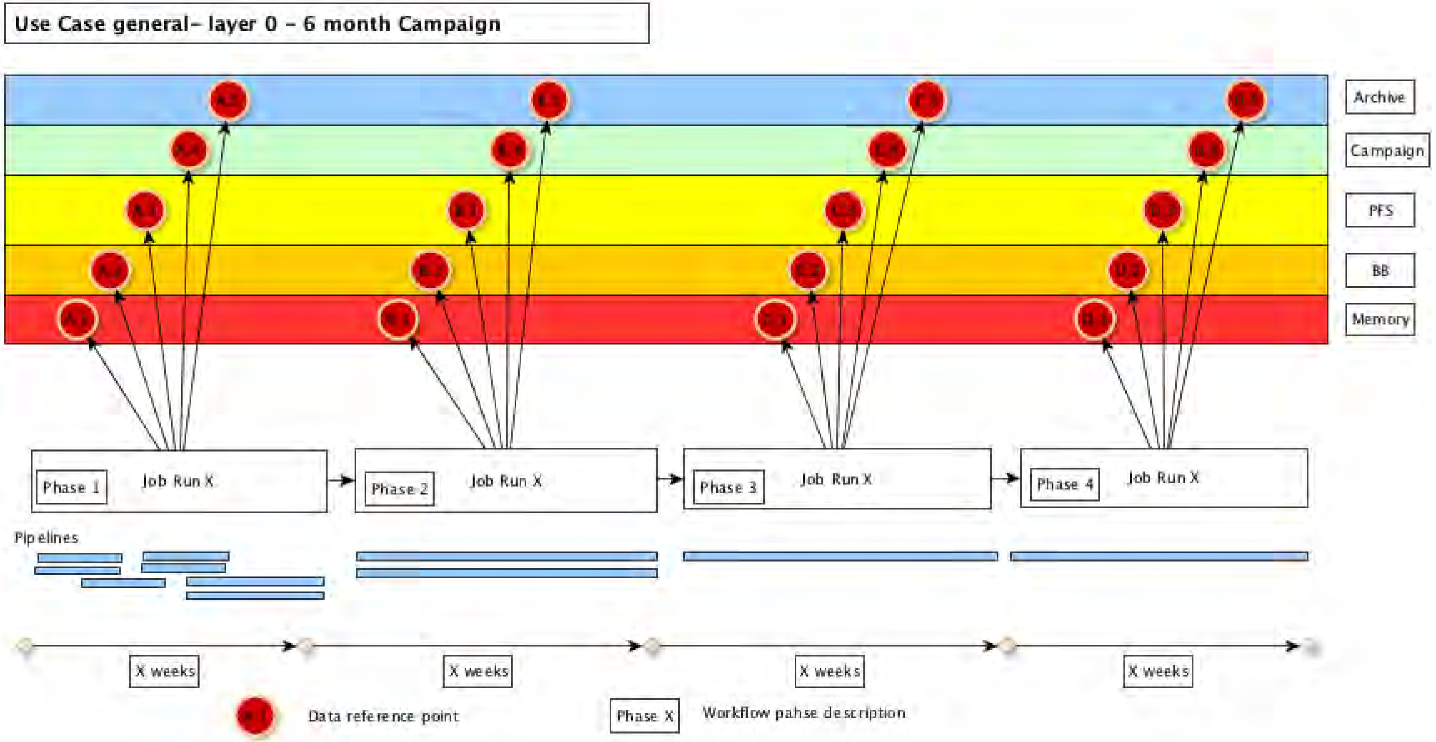
A description language:

- Wanted to capture flow - visually
- Incorporated data elements and data layers
- Defined a structure to describe relationships
- Templates to collect information
- Process to continue validation and reassessment



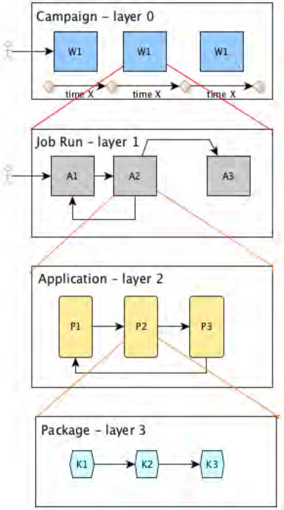
Layer 0 – Campaign/Pipeline Timeline – Use Case

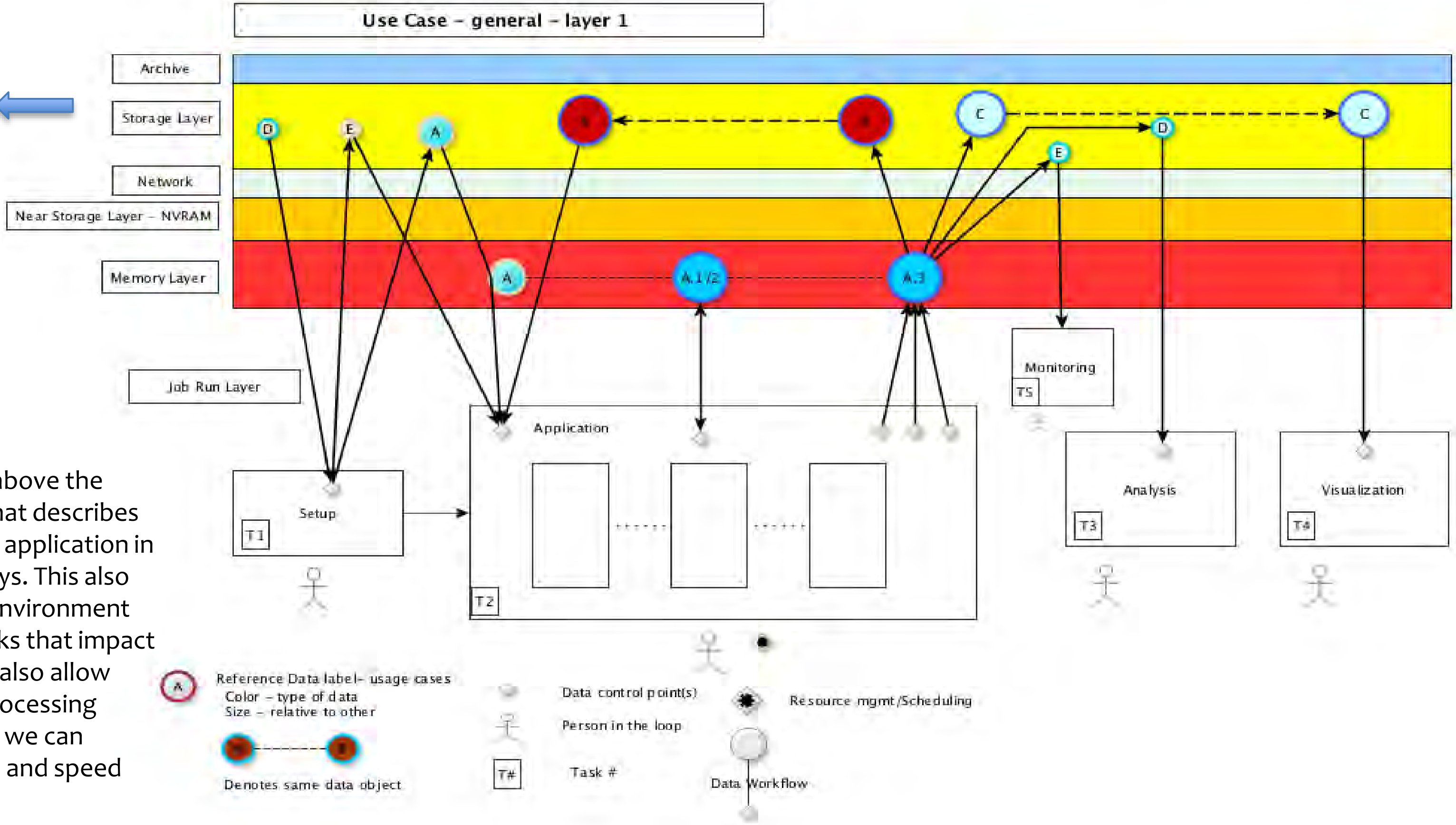




The Campaign / Pipe Line Series workflow layer is used to describe how job sequences are run within a project pipeline complete studies, also across campaign periods to identify impact through time. It is implementations of the Job Run (layer 1) workflows that are structured complete a problem set or solution across a time period.

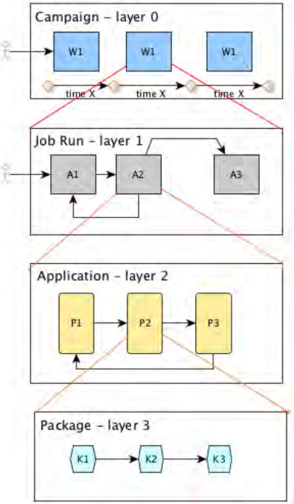
Layer 1 – Ensemble of applications – Use Case – example template

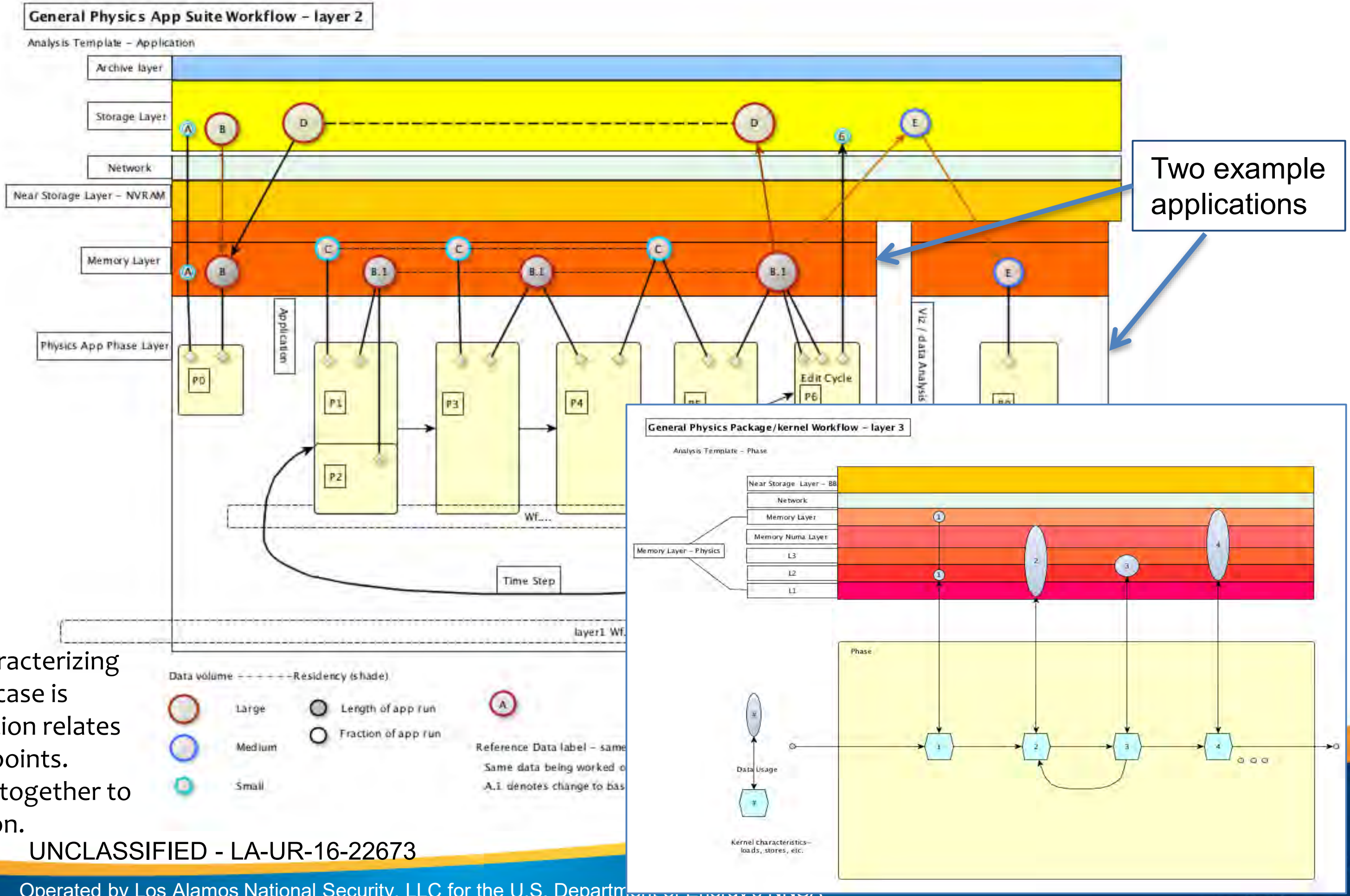




We described a layer above the application layer (2) that describes use cases that use the application in potential different ways. This also allowed the entry of environment based entities and tasks that impact a given workflow and also allow impact of scale and processing decisions. At this level we can describe time, volume and speed requirements.

Layer 2 – application characterization - example template

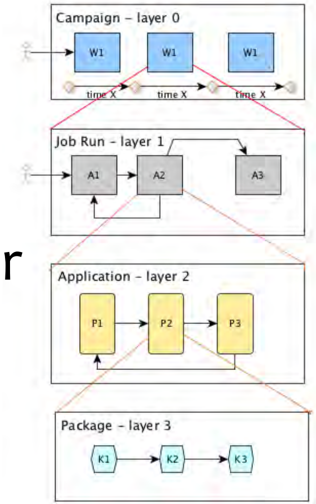




When looking at an application WF we started with what we called layer 2 – The Application Characterization layer.
Data elements were added to characterize relationships. This example shows 2 applications.

The other observation was that characterizing at this level was too general –a use case is necessary to assess how an application relates to specific environment and stress points.
Data collection templates were put together to collect and document the description.

Why are the Layers Important?

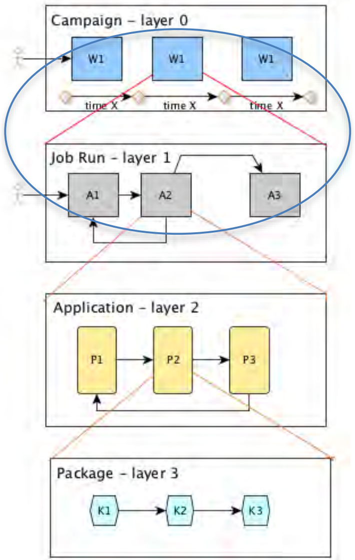


- Provides context – A Holistic View
  - Where do I fit in the big picture and what am I used for
  - What do I need and what constraints do I have
- If assessment is done across all layers – you can identify where there are bottlenecks, economic and resource utilization opportunities
- Allow for communication (people/machine) based on the layer(s) you are assessing

Initial effort – Information for Workflow Whitepaper for Crossroads (2020) RFP

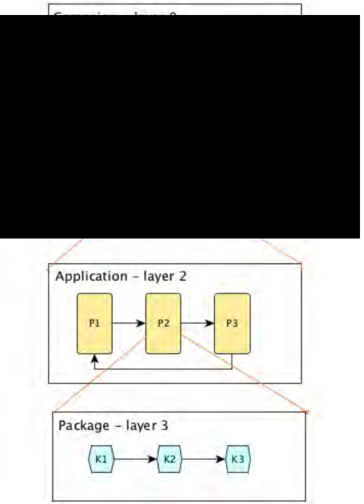
Characterizing what is happening in the Wild..

What are Users really doing?



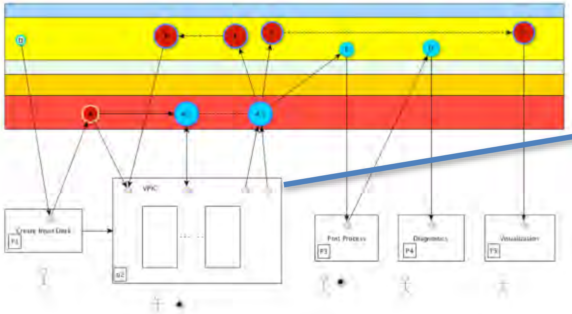
- Focused on Campaign 9 on Cielo, 8/29/15 – 2/29/16
- Characterized layer 0 and 1 with LANL users
- Included project suites – EAP, LAP, Silverton, VPIC

Page excerpts from VPIC workflow collection process



Use case – VPIC – Cielo campaigns,
Layer 1, Key Instigator: Lin Yin

Name:	VPIC – Vector Particle-In-Cell. It is a fixed regular mesh, fixed time step, Cartesian grid <x,y,z>, Job slice = time slice, memory/node = particles/cell (10s - 25K). Given simulation is ~ 1 week with multiple runs. Runs are put in at 6 hr segments mainly to get better throughput. Has excellent weak scaling and can scale well on any machine. Also parameters for run can be used to utilize memory available.	
Campaign usage model	VPIC normally included on Cielo campaigns, ~10% of cielo allocation. In normal 6 month campaign it will use initial 4 months for C1 runs, which are at 1000s of core level of which there may be 10 – 15 simulation runs at approximately 1 week per run. The last 2 months, C2 runs, are at the 70,000-core scale with approximately 2 simulation runs.	
Applications Involved	• VPIC setup • VPIC • Time slice analysis - IDL • Ensight	•
Environment:	Cielo for P1, P2, P3. P4 can be local machine, P5 on viz resources.	
Usage / Scale	Parameter: __ Typical: _X_ Hero: X Other:	



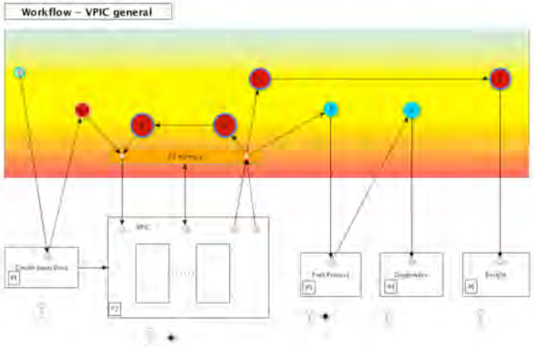
Data ref #: A – P2 P2 – A.1 – B/C/E	App: VPIC	Machine/scale/nodes: Run on Cielo C1 – 1000s cores C2 – 70,000 cores
App/Phase: -Description -Goal/approach -Why scale	Given simulation is ~ 1 week with multiple runs. Runs are put in at 6 hr segments mainly to get better throughput. Has excellent weak scaling and can scale well on any machine. Also parameters for run can be used to utilize memory available.	
Timing: -Walltime -Per job cycle -Interface pts	In normal 6 month campaign it will use initial 4 months for C1 runs, which are at 1000s of core level of which there may be 10 – 15 simulation runs at approximately 1 week per run. The last 2 months, C2 runs, are at the 70,000-core scale with approximately 2 simulation runs.	
Restart Dumps: -Timing -Number -Size -Total -# kept -Ideal?	2 checkpoints per 6 hour run, one file 9(B), 2GBs per node, uses odd/even process for retention, never saved to HPSS. Keep ~ 6 ckpts in arrears. Snapshot data (E), 8-10 GBs per node. Data analysis file (C) for visualization (ensight): • ~3GB files each for field, and each hydro species (at least 2 species/run) • ~10k files of each type/time slice. • Typically, ~400 time slices stored. • Data are stored on HPSS.	
Data: -Desc -Size -Access pattern	**Snapshot data** – collected per node, 8-10 GBs per, contains time slices and are stitched together and down selected at the end of a 6 hour run for analysis. Kept through a simulation run (1 week) and all stitched together at that time. Retention is HPSS a few years. On scratch, kept only about a month or two. **Data Analysis file** – same as Ensight sizes above, typically down-selected. Small portion will be archived to HPSS. File reduced through decimation [is this another task] - Data reduction is typically run serially in batch mode. Typically runs ~48 hours total for all files generated by a run.	
Mem/Storage: -Mem used -Storage used -Patterns	Memory - uses all. Parameters at initiation can be chosen to optimize memory available	
Issues		

Summary Application WFs - APEX RFP - WF whitepaper

1. LANL VPIC workflow

1.1. Description

VPIC is a three-dimensional, relativistic, electromagnetic particle-in-cell code, which uses an explicit time integration scheme to solve a variety of plasma physics problems on a structured mesh. VPIC is currently a mixture of about 40,000 lines of C and C++ code. VPIC currently exploits parallelism in three distinct ways. First, there is a distributed memory strategy, which uses asynchronous MPI calls to hide inter-node communication latencies. MPI can be used at the node level, core level or hardware thread level. Second, there is a thread level implemented with Pthreads, which will allow use of threads on a node at either the core level or hardware thread level. Finally, there is a vectorization level, which is implemented as a lightweight vector wrapper class, which can use either a portable implementation or a platform specific hardware intrinsic implementation. VPIC is specially designed to use single precision floating point calculations in order to optimize use of the available memory bandwidth. VPIC is used to perform simulations of astrophysical plasmas, laser-plasma interaction for ICF plasmas, relativistic laser-plasma interaction for laser-based accelerators, and first-principles studies of the mixing of dense plasma. At this time, VPIC only uses low level libraries such as MPI, Pthreads and vendor specific vector hardware intrinsics.



1.2. Campaign workload:

A normal simulation run is approximately 1 week made up of multiple job submittals. A 6 month campaign would include 4 months of smaller jobs (1000s of cores) and final 2 months target approximately 2 simulation runs of larger (70,000 core) jobs. Target 10% approximate use on ATS size machine.

APEX Workflows

LANL, NERSC, SNL

SAND2016-2371 O
LA-UR-15-29113

March 17, 2016

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

1

APEX WF Wh

erspective

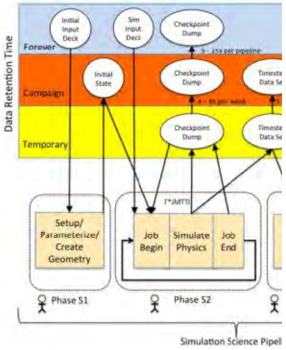


Figure 1: An example of an APEX simulat

Although a full system job might run for 24 hours and
checkpoints are generally overwritten using an odd/even
successfully terminates, 3 checkpoints should exist: an odd
an end-of-job checkpoint. As a simulation progresses over
number of checkpoints will be generated and overwritten, a
also be deleted, rather than retained. Over the course of an
4 - 8 checkpoints for several weeks and possibly for a few
rollback a week or month of computation in case an anoma
the above diagram we show that 4 - 8 checkpoints may be se
course of an allocation. Additionally, checkpoints can ofter
checkpoints may be retained forever so that portions of the
later for verification purposes.
Phase S2 also results in the generation of analysis data
evenly spaced intervals in *simulated time*, but are typically
life of the project or campaign. That is, the number of calc
data sets varies over the duration of the simulation. The
composed of many files to enable multiple types of analysis
deep analysis of anomalies, a rolling window of un-sampled

3

Anticipated increase in problem size by 2020	[redacted]	4 to 8x	1.25 to 1.5x		1x
Anticipated increase in workflow pipelines per allocation by 2020		2 to 8x	2 to 4x		10x
Storage APIs		DF5 or NetCDF	HDF5 or NetCDF	POSIX	POSIX
Routine number of analysis datasets					
Routine number of analysis files					
Checkpoint style		N to N	N to N	N to N	N to N
Files accessed/created per pipeline					
Data description (95% o storage volume)					
Data retained per Pipeline (percentage of memory)		5.87	32.54		
Temporary		0.02	30.00		
Analysis					
Checkpoint		0.02	30.00		
Input					
Out---of---core					
Campaign		2.00			
Analysis		2.00			
Checkpoint					
Input					
Forever		3.85	2.54		
Analysis		0.85	2.04		
Checkpoint					
Input		3.00*	0.50*		

vided the basis
ssions with
and is opening
ations with users
elopment teams

as we ask
al questions and
alidate

http://www.nersc.gov/research-and-development/apex/apex-benchmarks-and-workflo

Where is this taking us..

- Workflow co-design – HPC integration team / vendor / code developer / user
- Validation……
- Continued characterization and collection of workflow data
- Scoping future workflow

→ Communication/ Understanding

We become further enlightened as we compare notes and track what we are doing

→ Reality..

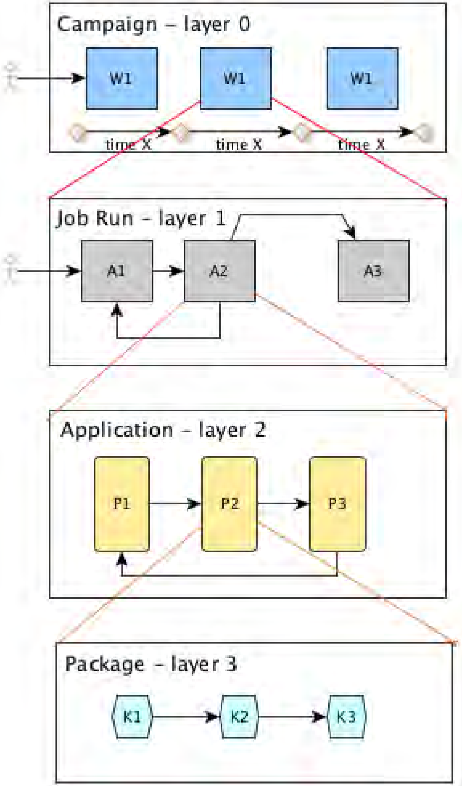
WF Performance

→ Monitoring…

→ Build on knowledge, roadmaps, and assess and track transition

Workflow Performance

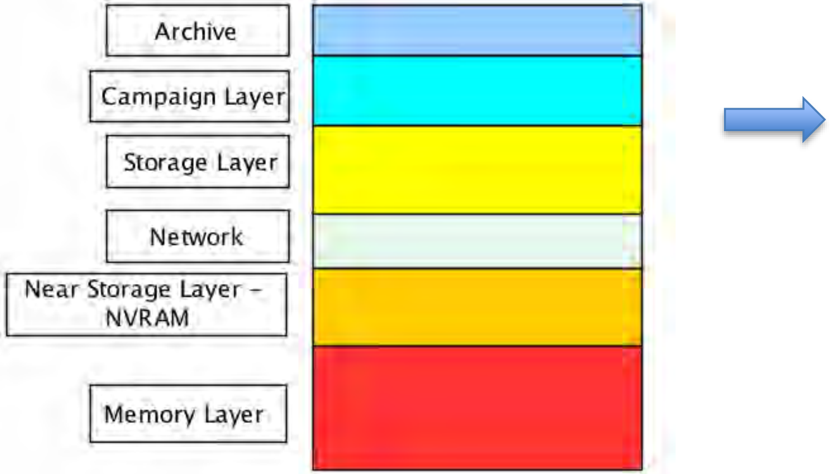
What are important metrics for each layer?



Collection approaches	For jobs
- Pull data from data bases summarized for historic runs	• Requirements across time. Scale, checkpoint, data read/written, Data needs over time, overall power, other.
- What is collected from each run – job level information. App and system – integrated and tracked.	• Requirements for job run. Data movement, checkpoint and local needs, data analysis process, data management. Multiple job tracking, resource integration into system.
- During run of app, mainly from within app- data, phases – integrated with system data for environmental perspective.	• Memory use, BB utilization, differences between packages in app, time step transition, analysis/preparation of data for analysis, IO, traces
- During run of app, mainly from within app – more intrusive collection. Performance, algorithm, architecture, compiler impact etc.	• Detailed measurements traditionally done through instrumentation and traditional tools such as TAU, HPC Toolkit, Open|SpeedShop, Cray Apprentice, etc. Focus on - MPI, threads, vectorization, power, etc.

How about the Data Stack?

Simple Data Stack from Application perspective



Is there a Taxonomy needed for the Data Stack

- Load and bleed APIs
- Distributed computation
- Transactions / resilience
- Global namespace
- Data Services based on usage models
- Data inflight ingestion and analysis
- Etc.

Parting Thoughts

- The ***workflow taxonomy*** allows us to build a map and provides knowledge requirements
  - Being done for Application stack - is there a similar one for system environment?
- The ***workflow performance*** allows us to identify collection points and identify data needs
- Technology roadmaps are driving a **transition in the economics** of computing infrastructures
- The resulting view provides a data driven environment to ***drive architecture and application decisions*** regarding balance and optimization

Thanks for Listening!

Questions..