Workflow Analysis – A Starting Point
SOS20

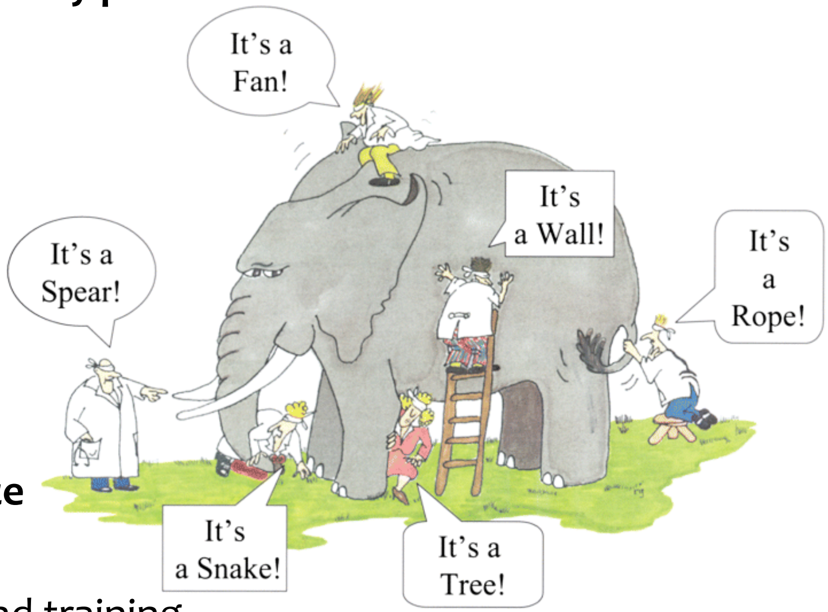
Dave Montoya

March 23, 2016

LA-UR-16-20222

What are Workflows and why do them?

- 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



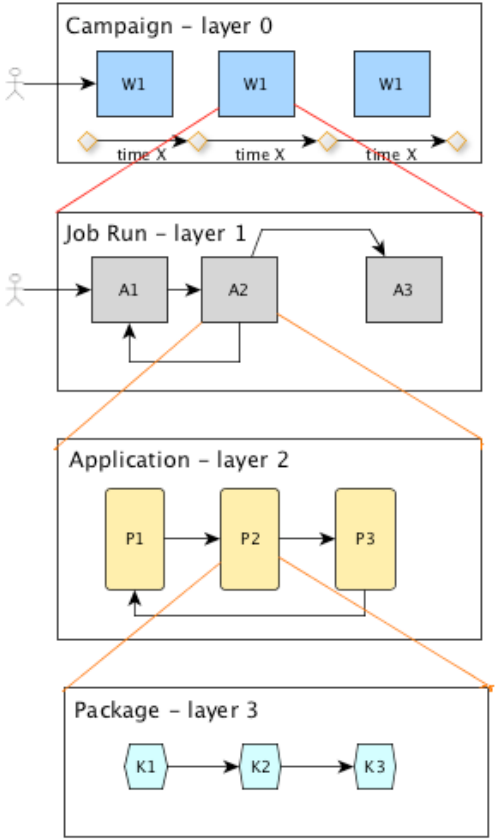
Workflow Layers within the Application Execution Stack

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.

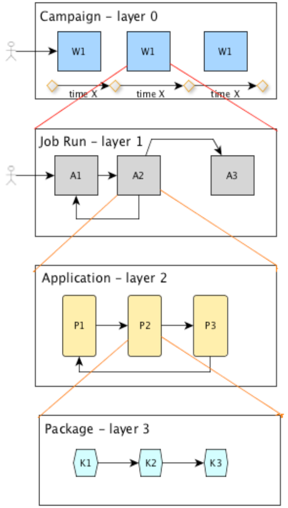
Layer 2 – **Application layer**. Within an application that may include one or more packages with differing computational and data requirements. Interacts across memory hierarchy to archival targets. The subcomponents of an application {P1..Pn} are meant to model various aspects of the physics; Layer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

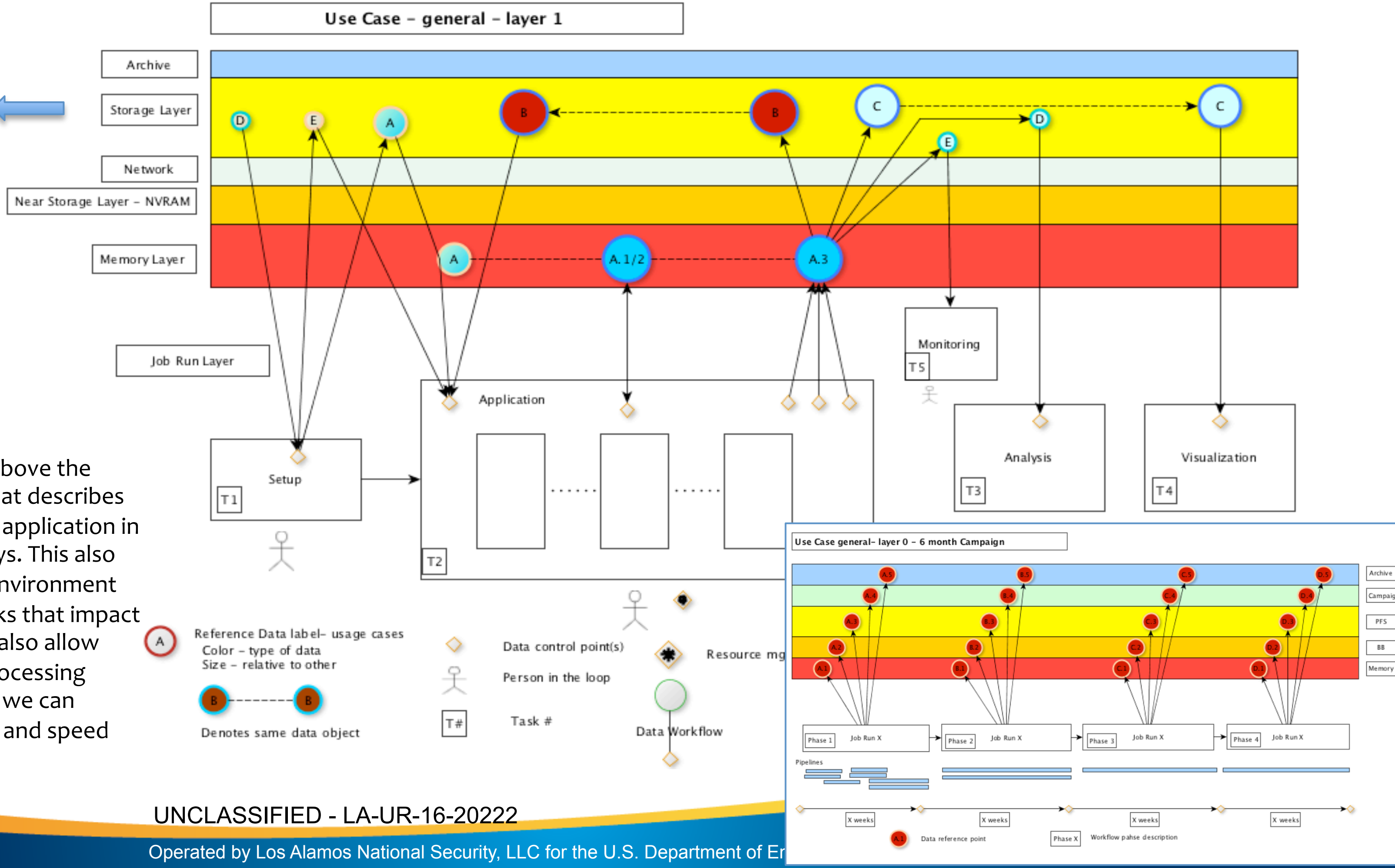
Layer 3 – **Package layer**. This describes the algorithm implementation and processing of kernels within a package and associated interaction with various levels of memory, cache levels and the overall underlying platform. This layer is the domain of the computer scientist and is where the software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



Started here

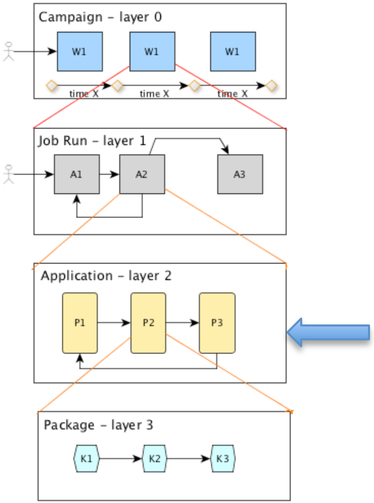
Layer 1 – Ensemble of applications – Use Case – example template

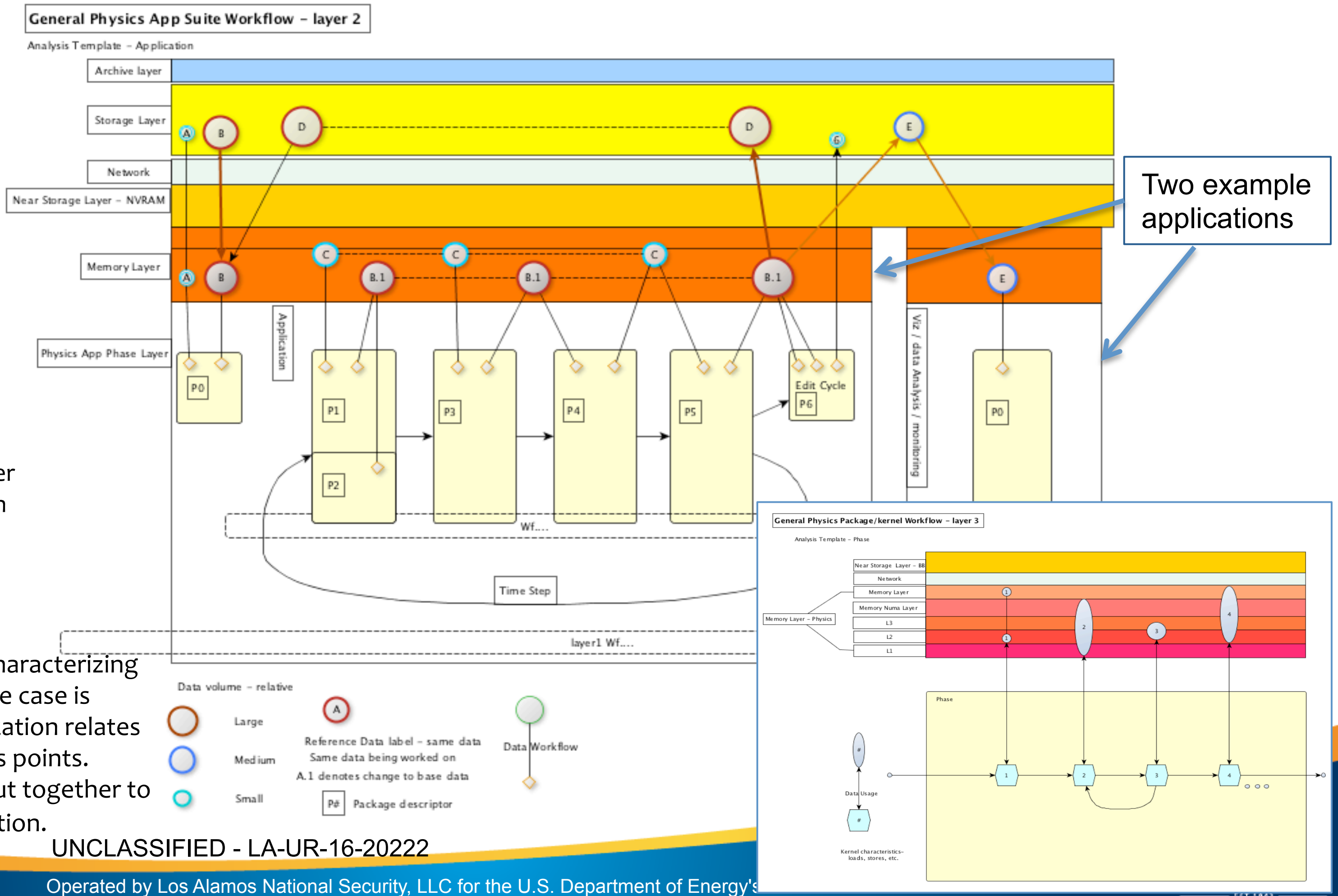




We described a layer above the application layer (2) that describes use cases that use the application in potential different ways. This also allowed the entry of environment based entities and tasks that impact a given workflow and also allow impact of scale and processing decisions. At this level we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

Layer 2 – application characterization - example template

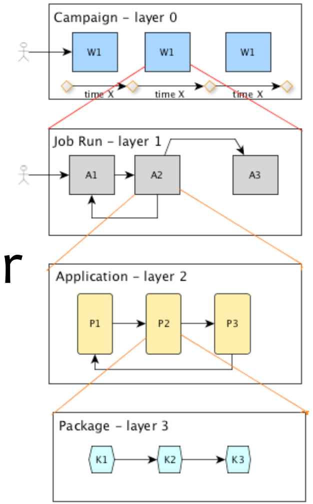




When looking at an application WF we started with what we called layer 2 – The Application Characterization layer.
Data elements were added to characterize relationships. This example shows 2 applications.

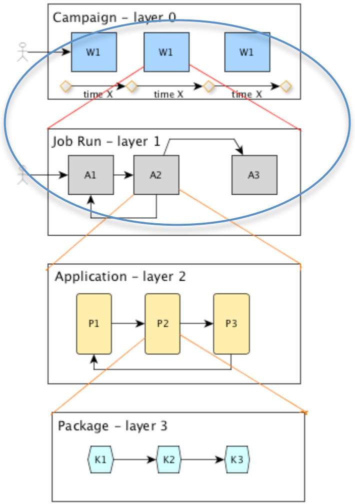
The other observation was that characterizing at this level was too general –a use case is necessary to assess how an application relates to specific environment and stress points.
Data collection templates were put together to collect and document the description.

Why are the Layers Important?



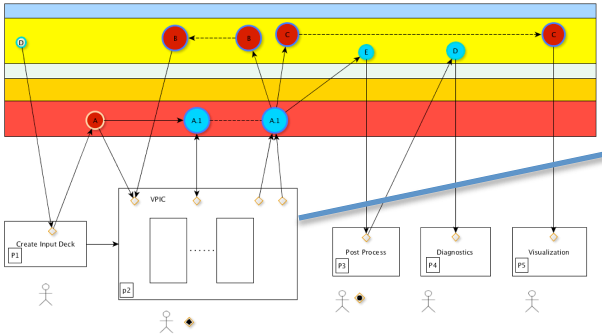
- Provides context – A Holistic View
  - Where do I fit in the big picture and what am I used for
  - What do I need and what constraints do I have
- If assessment is done across all layers – you can identify where the main bottlenecks are and resource utilization
- Allow for communication (people/machine)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

Page excerpts from VPIC workflow collection process



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

Name:	VPIC is xxxxx. 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 checkpoints in arrears. Data analysis file (C) for ensight at (table) size. Snapshot data (E), 8-10 GBs per node	
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 variable (table) **Data Analysis file** – Small portion will be archived to HPSS. File reduced through decimation [is this another task], size and Number (table)	
Mem/Storage: -Mem used -Storage used -Patterns	Memory - uses all. Parameters at initiation can be chosen to optimize memory available	
Issues		

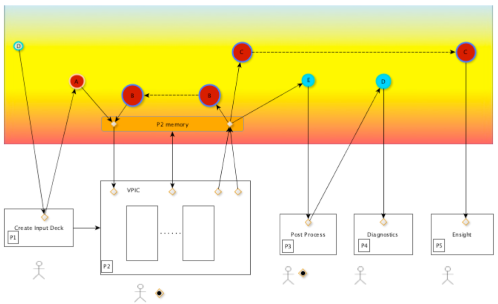
Summary WF paper

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.

Workflow – VPIC general



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.

WF Table from Crossroads RFP

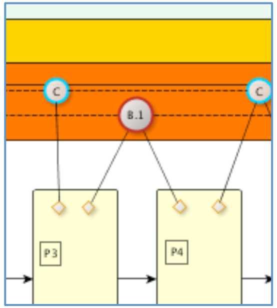
	Tri-Labs workload							
	LANL				SNL		LLNL	
Workflow	EAP	LAP	Silverton	VPIC	Dakota A	Dakota S	pF3D	R15
Workflow type	Sim	Sim	Sim	Sim	Sim/UQ	UQ	Sim	UQ
Workload percentage	60	5	15	10	10	10	10	20
Representative workload percentage	20	2	5	3	3	3	3	6
Wall time (hours)	262.4	64.0	128.0	157.2	100.0	100.0	2304.0	76.8
Hero Run Cielo Cores	65536	32768	131072	70000	131072	65536		
Routine Number of Cielo cores	16384	4096	32768	30000	8192	4096	2048	4096
Number of workflow pipelines per allocation	30	10	6	4	10 x 100	30 x 300	2	100
Anticipated increase in problem size by 2020	10 to 12x	8 to 12x	8 to 16x	8 to 16x	4 to 8x	1.25 to 1.5x		1x
Anticipated increase in workflow pipelines per allocation by 2020	1x	1x	1x	1x	2 to 8x	2 to 4x		10x
Storage APIs	POSIX	POSIX	POSIX	POSIX	HDF5 or NetCDF	HDF5 or NetCDF	POSIX	POSIX
Routine number of analysis datasets	100	100	225	150				
Routine number of analysis files								
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
Files accessed/created per pipeline								
Data description (95% of storage volume)								
Data retained per Pipeline (percentage of memory)	268.00	510.00	463.00	360.25	5.87	32.54		
Temporary	30.00	75.00	285.00	222.75	0.02	30.00		
Analysis			5.00	200.00				
Checkpoint	30.00	75.00	210.00	18.75	0.02	30.00		
Input			70.00	5.00				
Out---of---core								
Campaign	170.00	170.00	100.00	115.00	2.00			
Analysis	80.00	70.00	30.00	60.00	2.00			
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		
Analysis	25.00	250.00	8.00	10.00	0.85	2.04		
Checkpoint	40.00	10.00	70.00	12.50				
Input	3.00	5.00			3.00*	0.50*		

Data Perspective

This provided the basis for discussions with vendors and is opening conversations with users and development teams

What's Next

- **Continued validation of information** collected, additions to the layer 0 and 1 workflows – add table to the collection to characterize use cases
- **Begin to characterize layer 2** – This goes deeper into data movement, re-packaging, algorithm needs within packages defined within an application

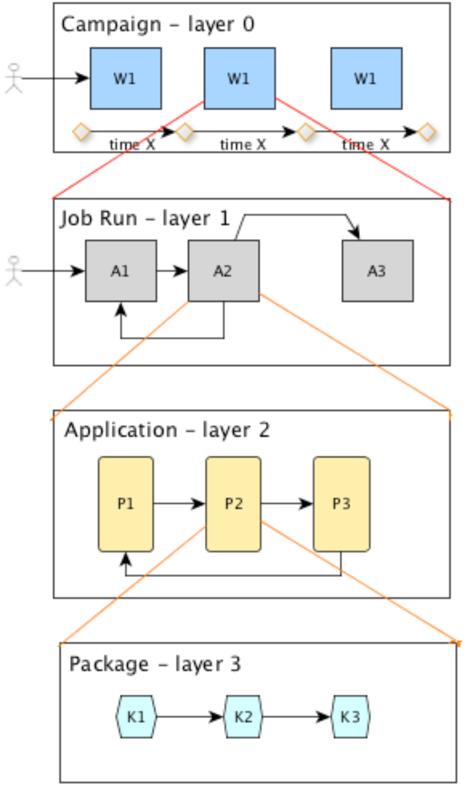


- **Validate workflow** – This ties to a sub-project focused on workflow performance. Includes performance gathering – monitoring data from application and system sources

Questions

Backup slides

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. Feeds up.	• 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. Feeds up.	• 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. Feeds up.	• 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.

LANL Workflow Performance / Monitoring

Prototype project:

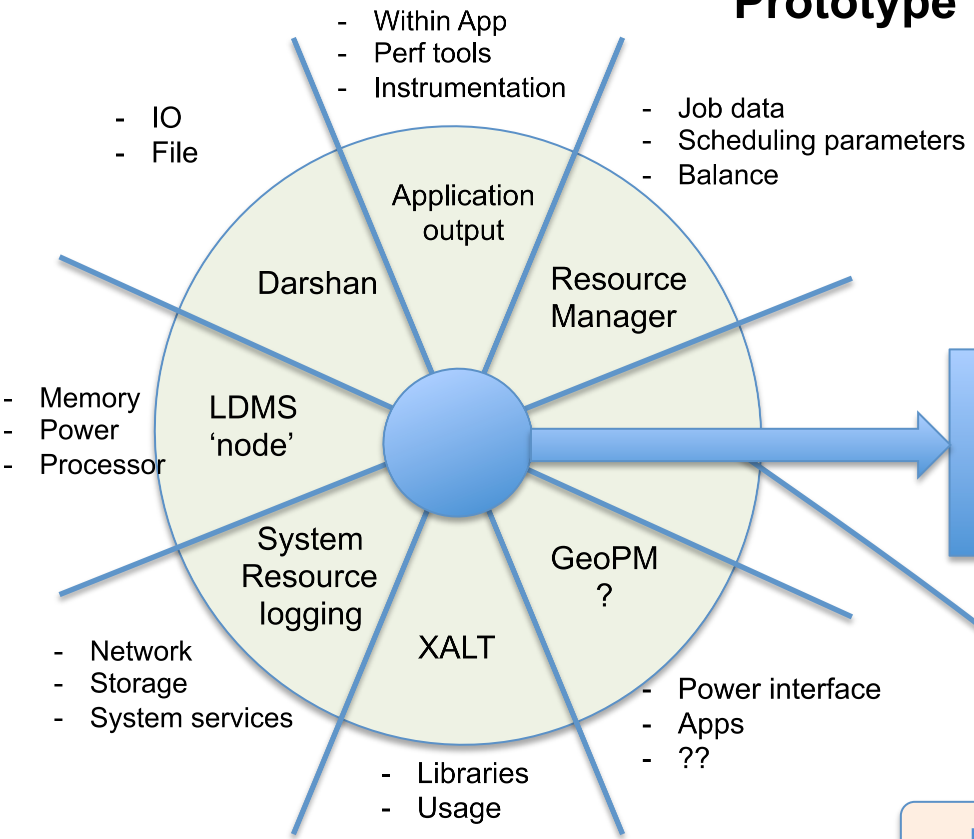
- Question Driven
  - What are we trying to answer or need to know
  - What data do we have and need
- Integration of developed collection and monitoring systems
- Assess collection points/tools
  - Data collected – Apps/System
  - Overhead/Conflict
  - Data analysis through integration

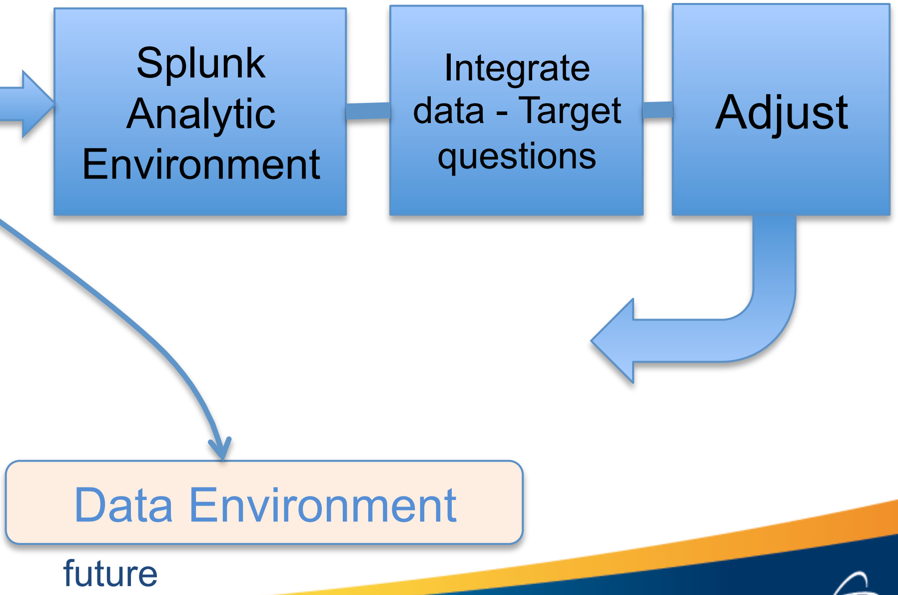
Focus Areas:

- System wide
- Application
  - Resource needs
  - Power
- Data Infrastructure
- IO
- Power
- Network
- Memory
- Reliability
- Interference

Prototype Workflow / Monitoring Environment

Try broad brush of Collection tools across the System, Scheduling, and Application Environment – Assess data, feed decisions, identify conflicts





Parting Thoughts

- The ***workflow taxonomy*** allows us to build a map and provides a collection process
  - 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
- The resulting view provides a data driven environment to drive balance and optimization decisions such as power and throughput