



Scientific Workflows on the Grid



Open Science Grid

Goals

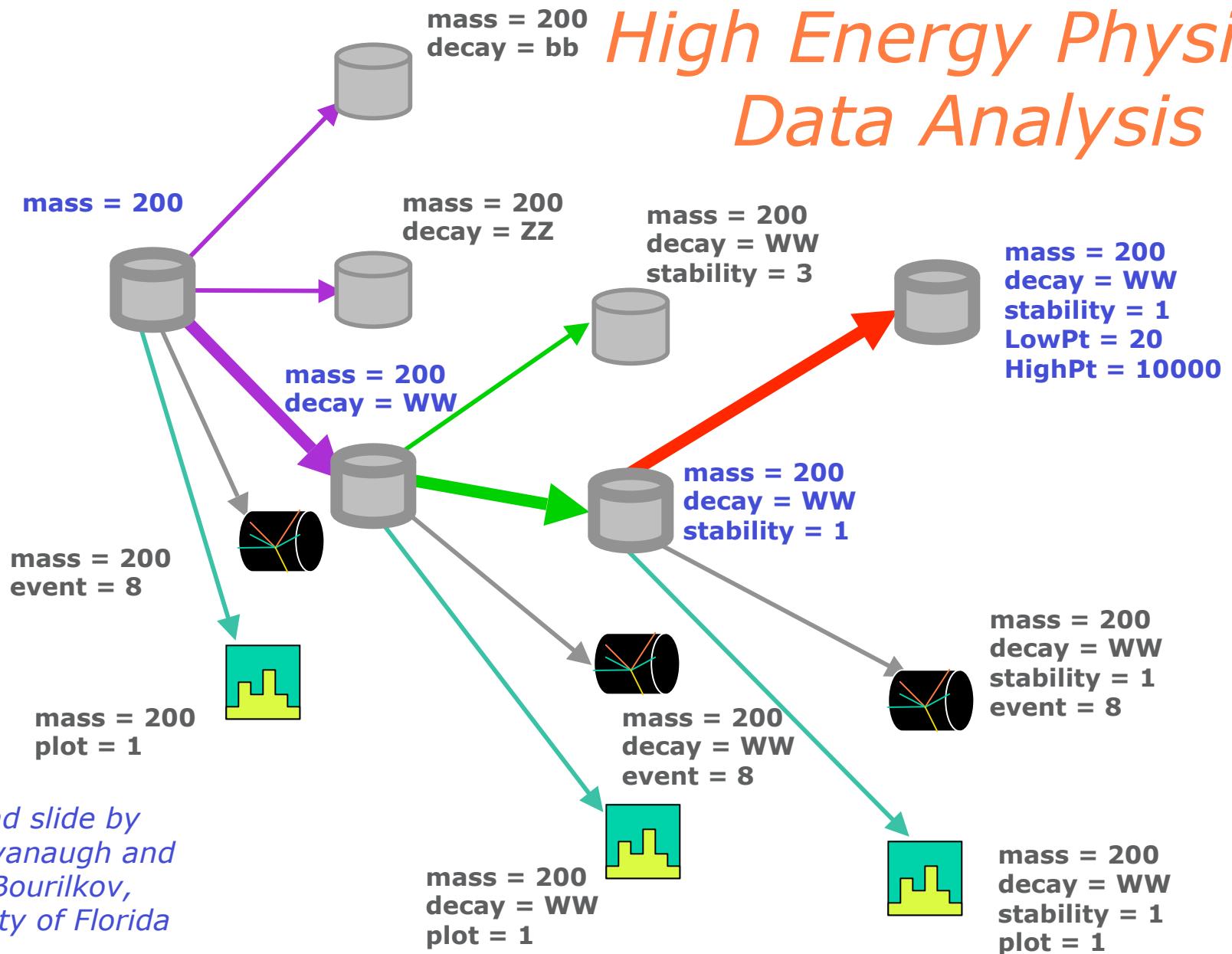
Enhance scientific productivity through:

- Discovery and application of datasets and programs at petabyte scale
- Enabling use of a worldwide data grid as a scientific workstation

Virtual Data and Workflows

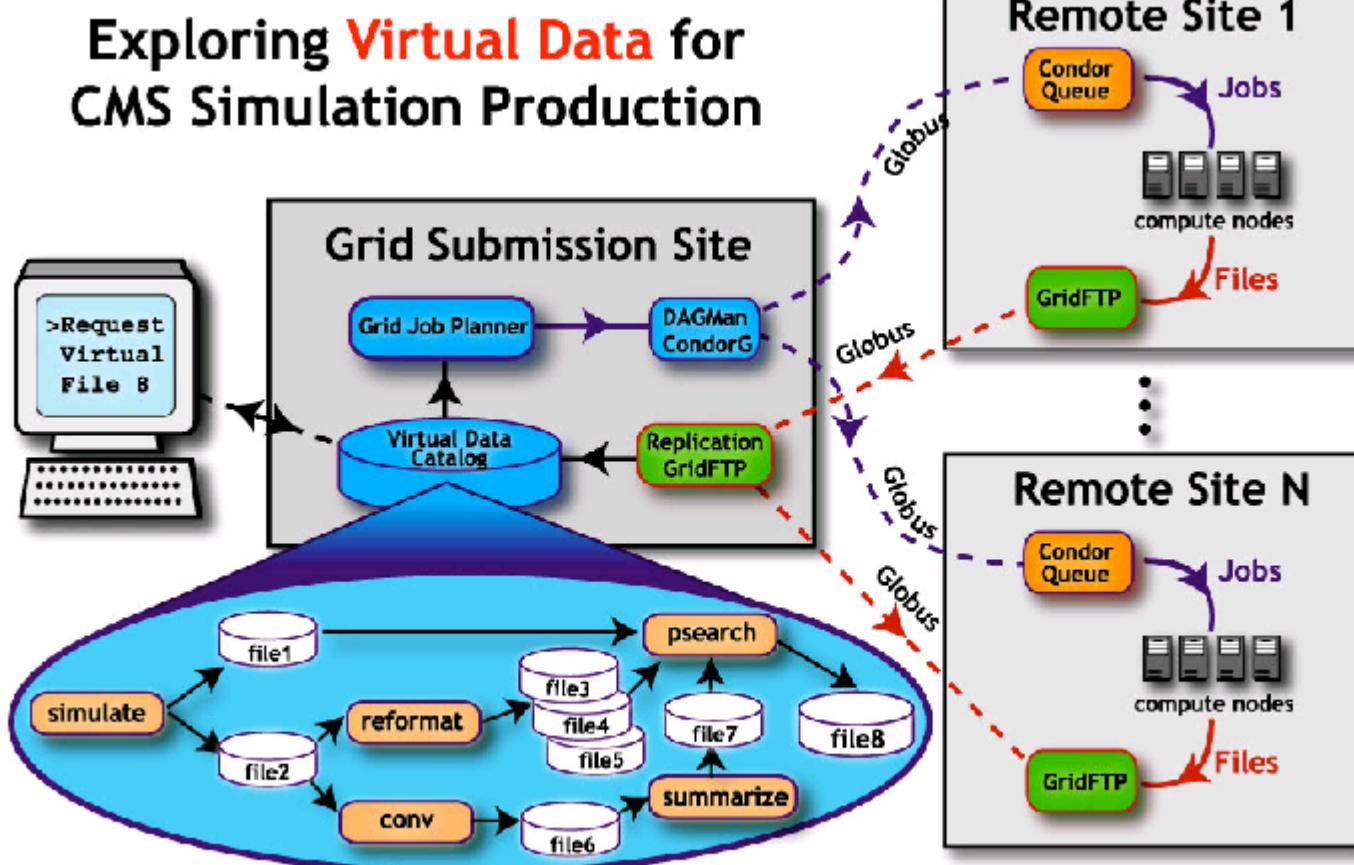
- Challenge is managing and organizing the vast computing and storage capabilities provided by Grids
- **Workflow** expresses computations in a form that can be readily mapped to Grids
- **Virtual data** keeps accurate track of data derivation methods and **provenance**
- Grid tools **virtualize** location and caching of data, and recovery from failures

Example Application: *High Energy Physics Data Analysis*



Work and slide by
Rick Cavanaugh and
Dimitri Bourilkov,
University of Florida

Executing a scientific workflow



What must we “virtualize” to compute on the Grid?

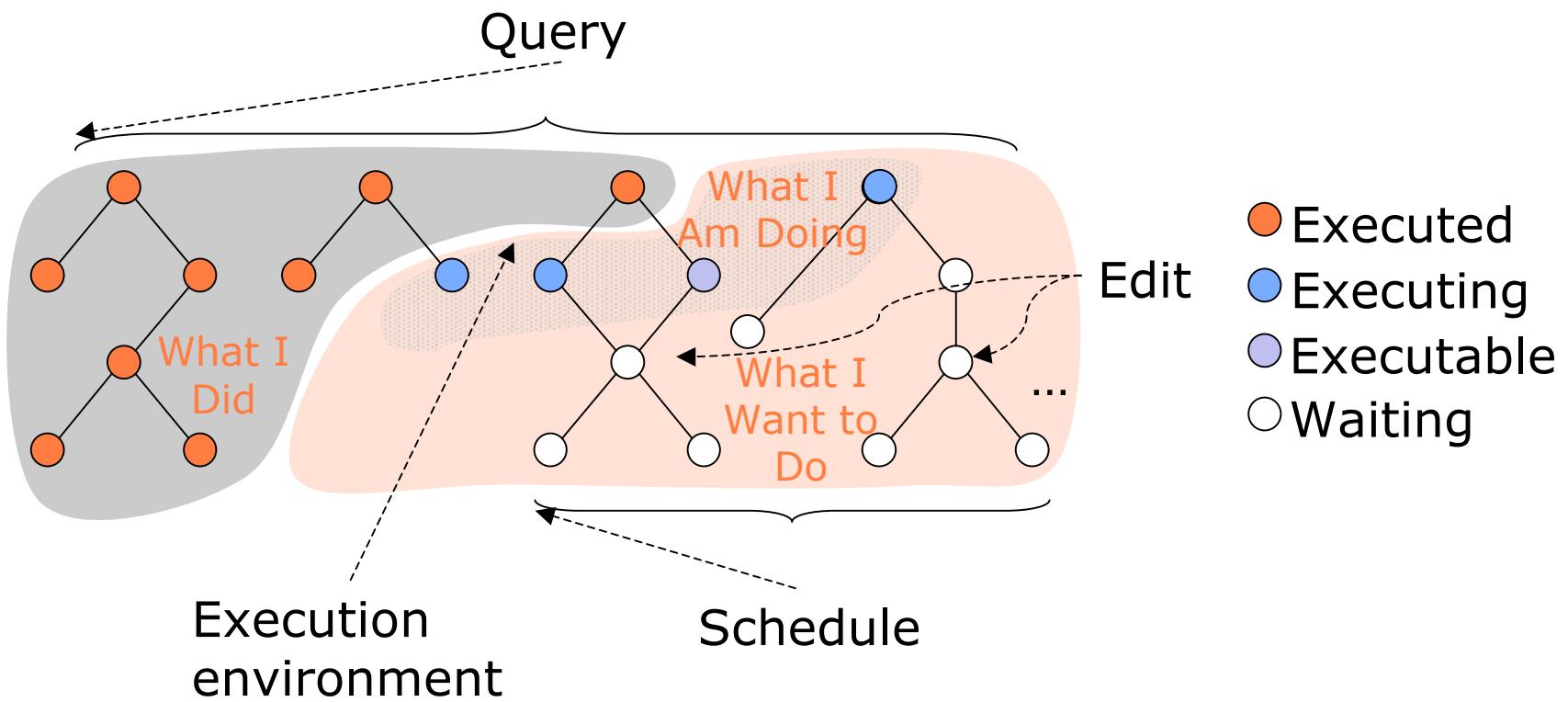
- Location-independent computing:
represent all workflow in abstract terms
- Declarations not tied to specific entities:
 - sites
 - file systems
 - schedulers
- Failures – automated retry for data server
and execution site un-availability

Mapping the Science Process to workflows

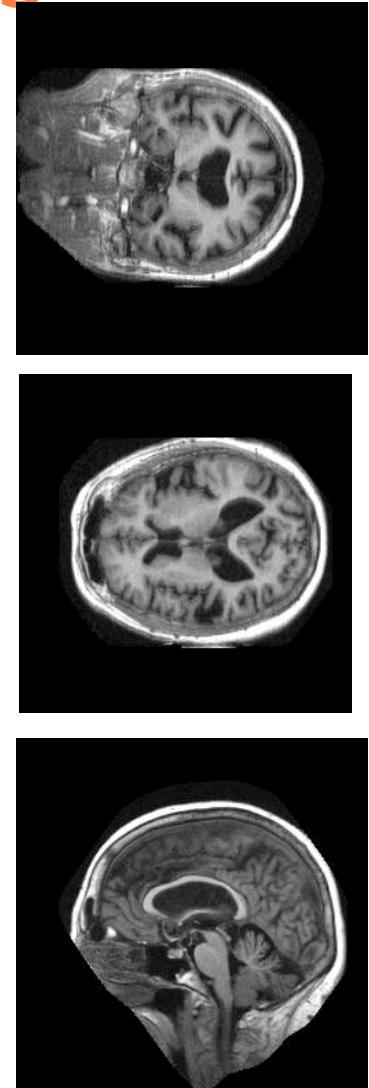
- Start with a single workflow
- Automate the generation of workflow for sets of files (datasets)
- Replicate workflow to explore many datasets
- Change Parameters
- Change code – add new transformations
- Build new workflows
- Use provenance info

How does Workflow Relate to Provenance?

- Workflow – specifies what to do
- Provenance – tracks what was done

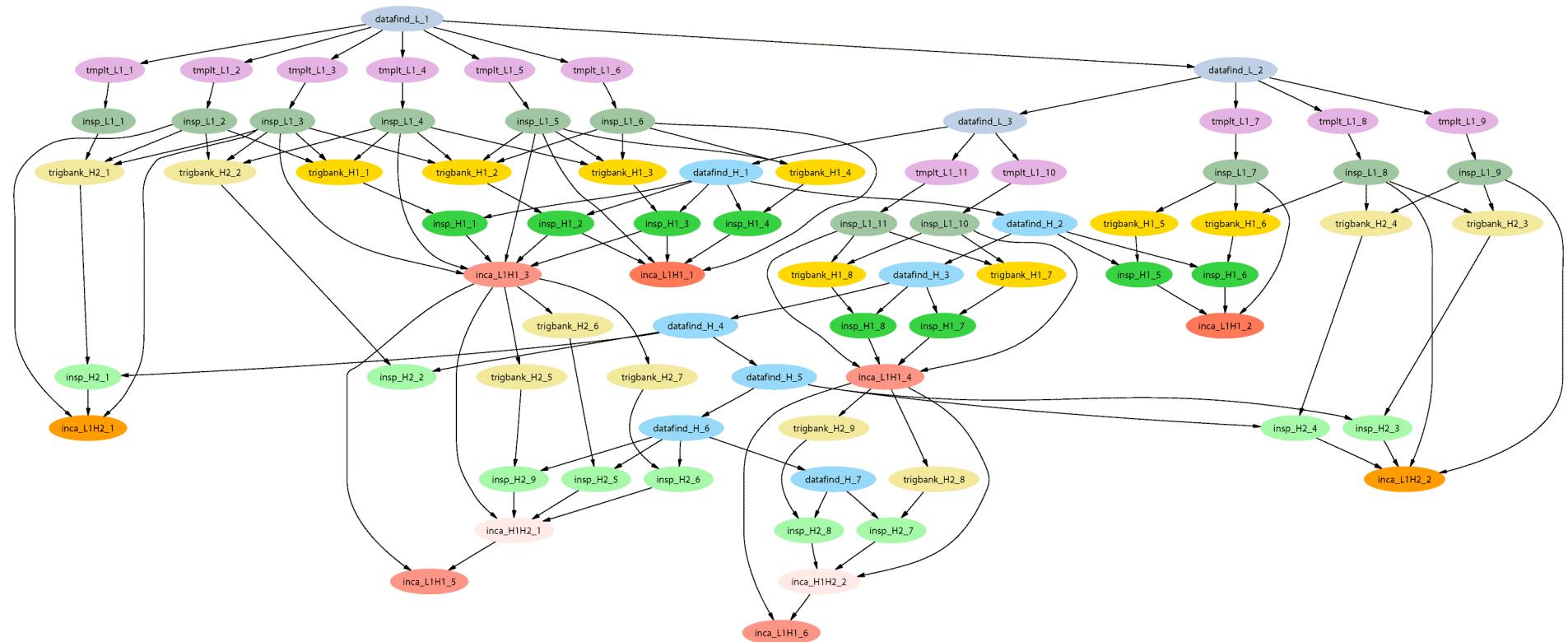


Functional MRI Analysis



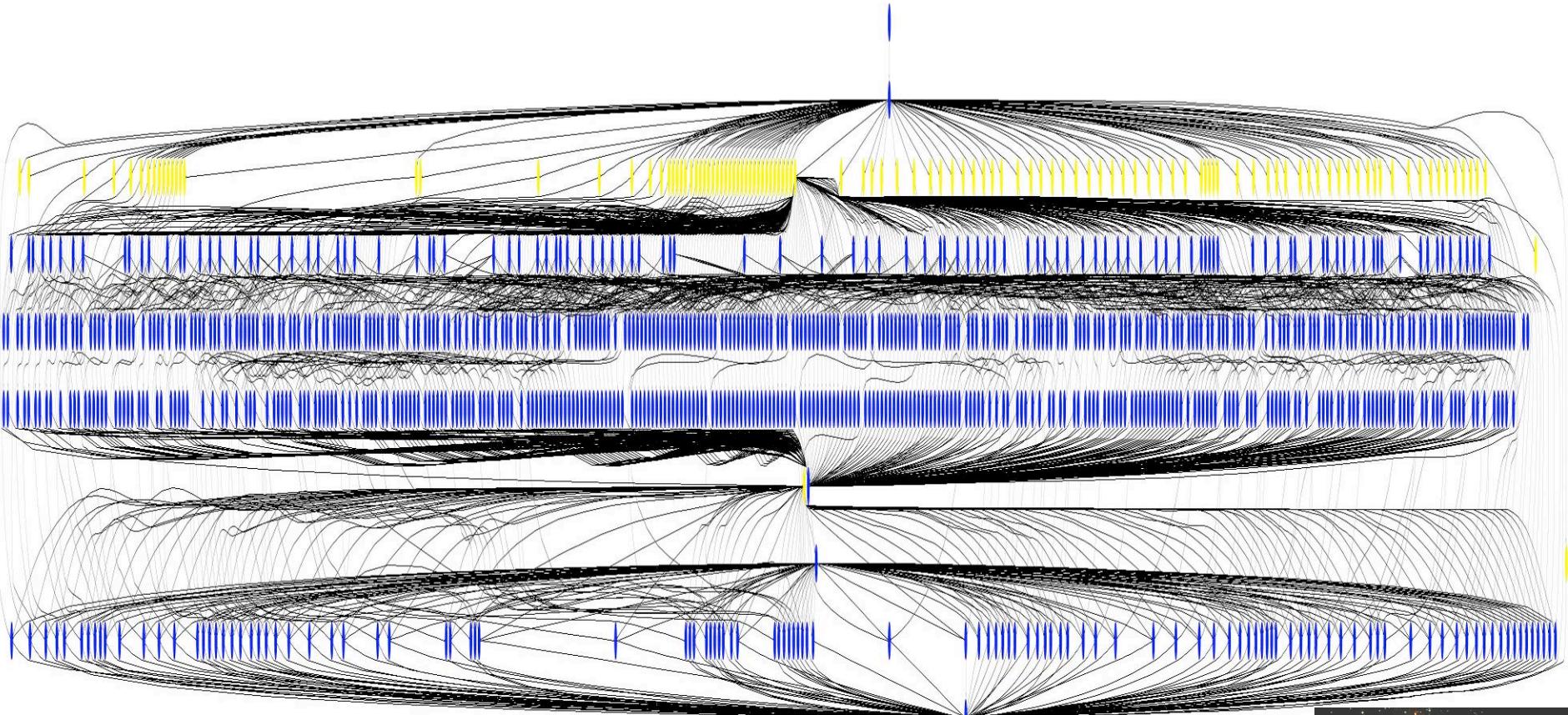
Workflow courtesy James Dobson, Dartmouth Brain Imaging Center

LIGO Inspiral Search Application



*Inspiral workflow application is the work of Duncan Brown, Caltech,
Scott Koranda, UW Milwaukee, and the LSC Inspiral group*

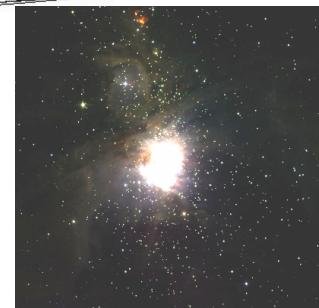
Example Montage Workflow



~1200 node workflow, 7 levels

<http://montage.ipac.caltech.edu/>

Mosaic of M42 created on
the Teragrid using Pegasus



Blasting for Protein Knowledge

BLAST compare of complete nr database for sequence similarity and function characterization

gi 23499780 gnl REF_tigr BRA0013	gi 16080231 ref NP_291080.1	44.27	253	131	1	15	257	8	2603.7	e-53	209.9
gi 23499780 gnl REF_tigr BRA0013	gi 123098409 ref NP_591875.1	43.48	253	133	2	16	256	5	2673.8	e-50	199.9
gi 23499780 gnl REF_tigr BRA0013	gi 14837187 ref ZP_00294182.1	44.92	256	125	2	14	256	7	2591.1	e-49	198.4
gi 23499780 gnl REF_tigr BRA0013	gi 152005400 gb AAY125342.1	44.75	257	126	2	15	258	3	2561.9	e-49	197.6
gi 23499780 gnl REF_tigr BRA0013	gi 14864015 ref ZP_00317908.1	44.49	245	134	1	13	257	5	2476.1	e-48	192.6
gi 23499780 gnl REF_tigr BRA0013	gi 130348881 gb AWA2934.1	39.53	253	138	3	18	257	5	2552.0	e-43	177.6
gi 23499780 gnl REF_tigr BRA0013	gi 19655222 gb AFV33939.1	40.64	251	138	1	17	256	10	2602.7	e-43	177.2
gi 23499780 gnl REF_tigr BRA0013	gi 127356808 gb AAQ07757.1	43.03	251	130	4	18	256	11	2602.4	e-41	170.6
gi 23499780 gnl REF_tigr BRA0013	gi 112597924 gb AAU16899.2	46.70	182	96	1	62	243	5	1856.8	e-39	162.5
gi 23499780 gnl REF_tigr BRA0013	gi 146363338 ref ZP_00260793.1	39.58	240	136	2	14	253	6	2361.8	e-36	154.5

REF_tigr BRA0013	gi 39933731 ref NP_946007.1	34.90	255
REF_tigr BRA0013	gi 40782600 ref ZP_00279106.1	35.92	245
REF_tigr BRA0013	gi 41407534 ref NP_960370.1	36.09	266
REF_tigr BRA0013	gi 40851585 ref ZP_00305793.1	32.39	247
REF_tigr BRA0013	gi 15966306 ref NP_386659.1	36.50	263
REF_tigr BRA0013	gi 17548526 ref NP_521866.1	36.36	264

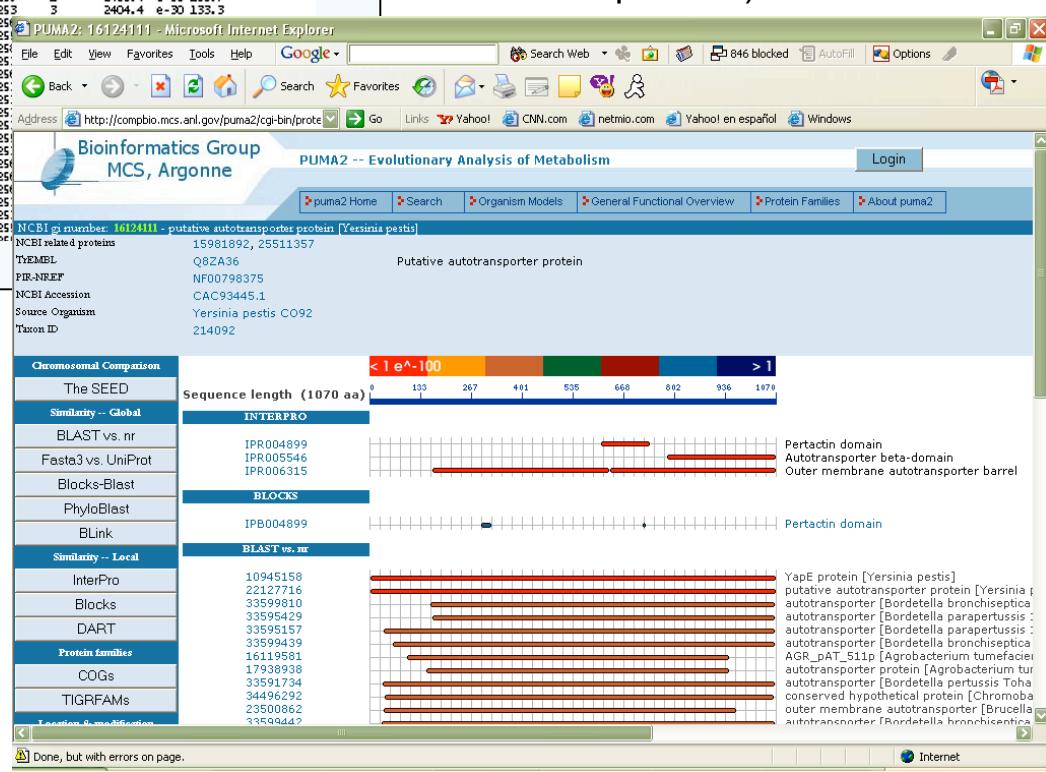
gi 23499780 gnl REF_tigr BRA0013	gi 151891730 ref NP_074421.1	38.87	247	136	7	18	256	1	2403.4	e-30	133.7
gi 23499780 gnl REF_tigr BRA0013	gi 1145881 gb AAA23739.1	33.87	248	147	3	13	253	3	2404.4	e-30	141.4
gi 23499780 gnl REF_tigr BRA0013	gi 125028334 ref NP_739388.1	35.20	250	147	4	15	259	3	2404.2	e-32	140.2
gi 23499780 gnl REF_tigr BRA0013	gi 21220953 ref NP_626732.1	38.52	257	138	6	12	259	3	2404.2	e-32	139.4
gi 23499780 gnl REF_tigr BRA0013	gi 14631405 ref ZP_00214636.1	33.86	254	153	2	12	259	3	2404.0	e-31	137.9
gi 23499780 gnl REF_tigr BRA0013	gi 141406852 ref NP_959688.1	33.63	238	149	2	16	259	3	2404.0	e-30	135.2
gi 23499780 gnl REF_tigr BRA0013	gi 15864471 ref NP_229523.1	35.69	255	144	5	12	259	3	2404.0	e-30	134.4
gi 23499780 gnl REF_tigr BRA0013	gi 123470090 ref ZP_00125423.1	35.20	250	145	4	12	259	3	2404.0	e-30	134.0
gi 23499780 gnl REF_tigr BRA0013	gi 148347237 ref NP_0027371.1	34.03	257	146	4	12	259	3	2404.0	e-30	133.3
gi 23499780 gnl REF_tigr BRA0013	gi 468476551 ref ZP_00214615.1	36.08	250	145	9	12	259	3	2404.0	e-30	132.5
gi 23499780 gnl REF_tigr BRA0013	gi 28615110 gb AAQ54587.1	35.29	250	142	4	12	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 27378783 ref NP_720312.1	35.29	251	143	3	14	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 11208336 sp P50198 LIDN_PSEP	34.23	250	143	4	12	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 133594148 ref NP_881792.1	34.17	240	148	5	18	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 133598116 ref NP_885759.1	34.17	240	148	5	18	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 2738226 ref NP_773825.1	34.32	271	151	5	5	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 13995569 ref NP_951520.1	32.14	252	150	2	13	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 116760071 ref NP_455688.1	33.06	248	149	3	13	259	3	2404.0	e-30	131.5
gi 23499780 gnl REF_tigr BRA0013	gi 36958919 gb AAU087344.1	33.74	246	136	6	19	259	3	2404.0	e-30	131.5

Analysis on the Grid

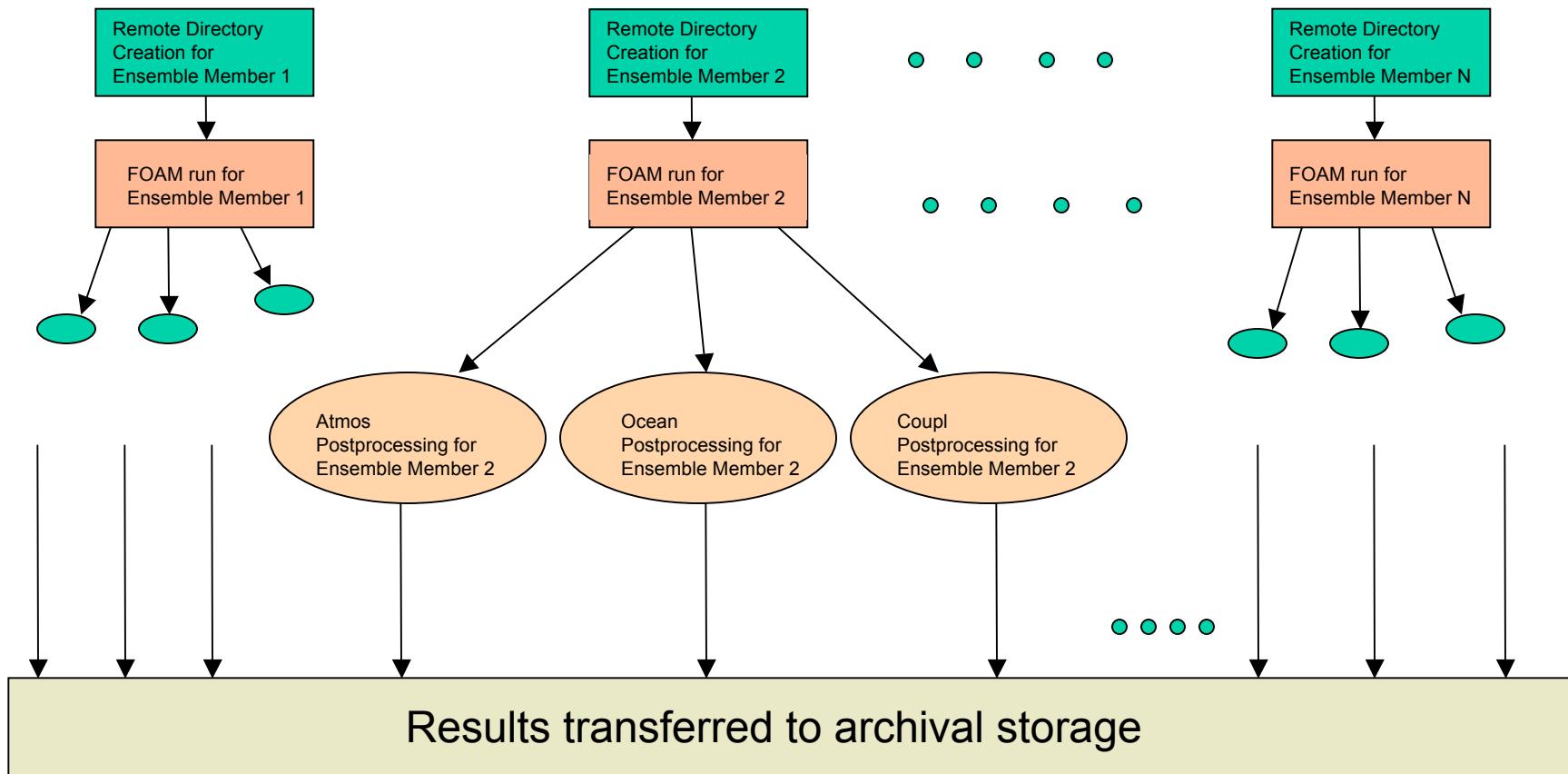
The analysis of the protein sequences occurs in the background in the grid environment. Millions of processes are started since several tools are run to analyze each sequence, such as finding out protein similarities (BLAST), protein family domain searches (BLOCKS), and structural characteristics of the protein.

Knowledge Base

PUMA is an interface for the researchers to be able to find information about a specific protein after having been analyzed against the complete set of sequenced genomes (nr file ~ approximately 3 million sequences)



FOAM: Fast Ocean/Atmosphere Model 250-Member Ensemble Run on TeraGrid under VDS

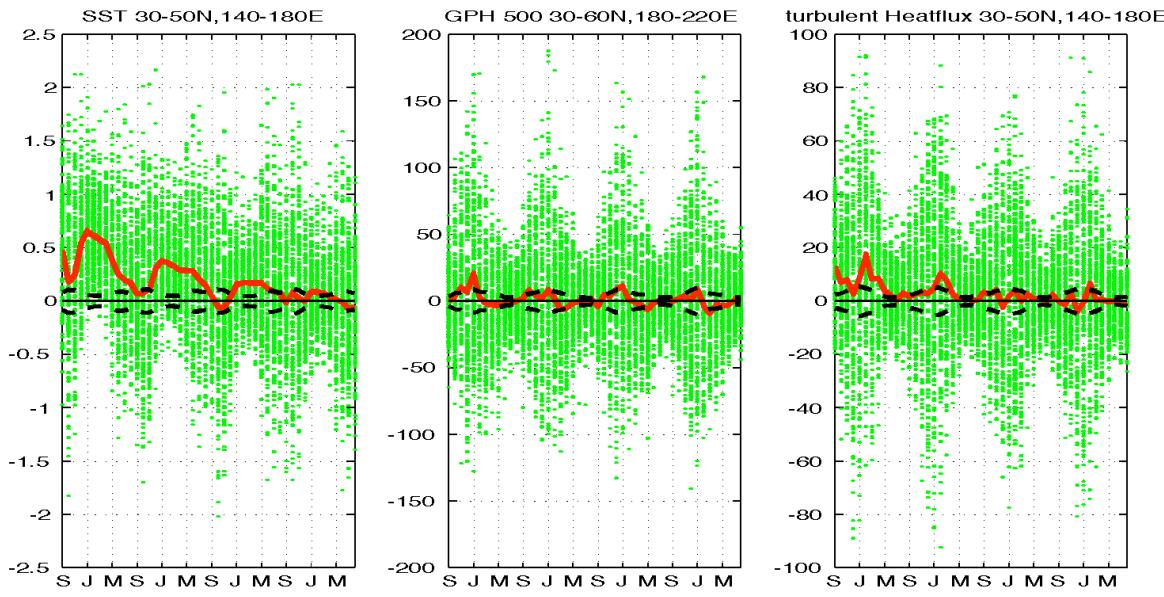


Work of: Rob Jacob (FOAM), Veronica Nefedova (Workflow design and execution)

FOAM: TeraGrid/VDS Benefits

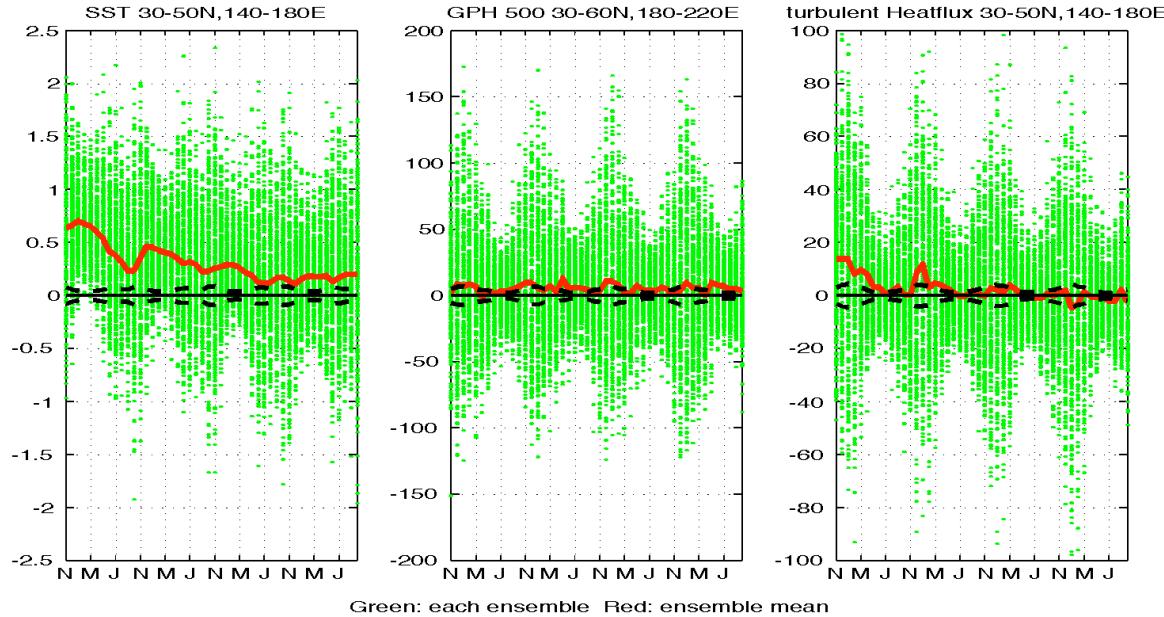
160 ensemble members = 2.5 months to run

*Climate
Supercomputer*



250 ensemble members = 4 days to run

*TeraGrid with
NMI and VDS*



Green: each ensemble Red: ensemble mean

*FOAM
application by
Rob Jacob,
Argonne; VDS
workflow by
Veronika
Nefedova,
Argonne
Visualization
courtesy Pat
Behling and
Yun Liu, UW
Madison..*

i2u2 - Leveraging Virtual Data for Science Education

Cosmics Data Interface - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Mail Print Word Excel People

Google Address http://quarknet.uchicago.edu:8085/cosmic/search.jsp?t=plot&f=view

Performance Study
Number of Events vs Time since Threshold (in seconds)

Performance Study
Number of Events vs Time since Threshold (in seconds)

Shower Study
Time since threshold (in seconds)

Flux Study
Flux (counts/s) vs Time (second)

Choose Raw Data Center - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Mail Print Word Excel People

Google Address http://quarknet.uchicago.edu:8085/cosmic/analyze.jsp?filename=67.2004.1027.0&t=flux

Cosmic Ray Collaboration Logged in as group: fermigroup Logout

Home Resources Upload Data Posters Site Index

Assessment View Data Performance Lifetime Flux Shower View Plots

Calculate the flux for your data file. Remember, flux = particles / time / area

Understand The Graph

You're analyzing... Chan1 events 1095 Chan2 events 3323 Chan3 events 1121 Chan4 events 3294

Click Analyze to use the default parameters. Control the analysis by expanding the options below.

Analysis Controls Channel Number 1

Plot Controls Bin Width (seconds) 60 X-min e.g. 1028/2004 3.00 X-max e.g. 1029/2004 18.00 Y-min Y-max Plot Size Medium Plot Title Flux Study Data: 10/27/2004 0:0:4 Detector(s): 67 Channel: 1

Analyze

staring.png teachers.png techstaff.png

Group: tea
Created: 0
View/Add

Global Positioning System
Scintillators
Cosmic Rays
DAQII Board
GPS
Photomultiplier Tube

The screenshot displays the Cosmics Data Interface within a Microsoft Internet Explorer window. At the top, there's a toolbar with standard browser buttons like Back, Forward, Stop, Refresh, and Home. Below the toolbar is a menu bar with File, Edit, View, Favorites, Tools, and Help. The address bar shows the URL: http://quarknet.uchicago.edu:8085/cosmic/search.jsp?t=plot&f=view. The main content area contains four subplots: 'Performance Study' (two side-by-side), 'Shower Study' (a 3D scatter plot), and 'Flux Study' (a line plot). Below these plots is a smaller window titled 'Choose Raw Data Center - Microsoft Internet Explorer' showing raw data analysis parameters. A large diagram at the bottom right illustrates the experimental setup, featuring a Global Positioning System (GPS) unit, Scintillators, Cosmic Rays, a DAQII Board, and a Photomultiplier Tube connected to a computer monitor displaying identification numbers.

VDS Applications

Application	Jobs / workflow	Levels	Status
ATLAS HEP Event Simulation	500K	1	In Use
LIGO Inspiral/Pulsar	~700	2-5	Inspiral In Use
NVO/NASA Montage/Morphology	1000s	7	In Use
GADU Genomics: BLAST,...	40K	1	In Use
fMRI AIR, freeSurfer prepro	100s	12	In Devel
QuarkNet CosmicRay science	<10	3-6	In Use
SDSS Coadd; Cluster Search	40K 500K	2 8	CS Research
FOAM Ocean/Atmos Model	2000 (core app runs 250 8-CPU jobs)	3	In use
GTOMO Image proc	1000s	1	In Devel
SCEC Earthquake sim	1000s		In use

Some workflow systems

- DAGman – part of Condor
- VDS – VDL+Pegasus
- Swift

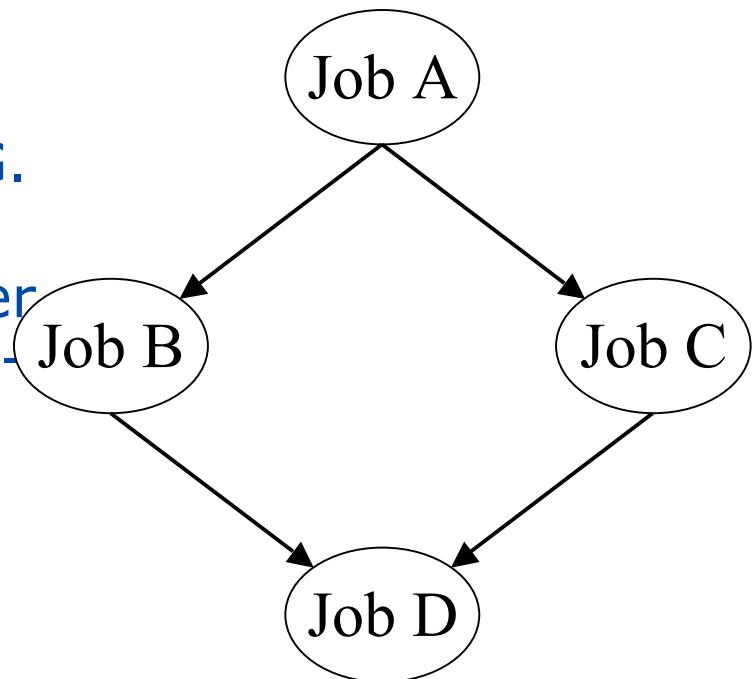
DAGMan

- Directed Acyclic Graph Manager

- DAGMan allows you to specify the *dependencies* between your Condor jobs, so it can *manage* them automatically for you.
 - By default, Condor may run your jobs in any order, or everything simultaneously, so we need DAGMan to enforce an ordering when necessary.
- Example: “Don’t run job “B” until job “A” has completed successfully.”

What is a DAG?

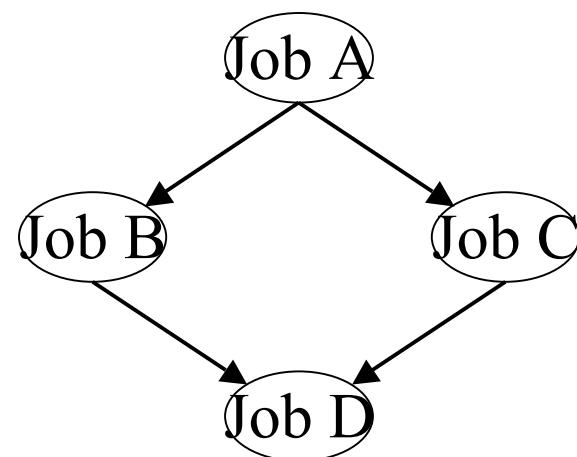
- A DAG is the **data structure** used by DAGMan to represent these dependencies.
- Each job is a “**node**” in the DAG.
- Each node can have any number of “parent” or “children” nodes - as long as there are **no loops!**



Defining a DAG

- A DAG is defined by a *.dag file*, listing each of its nodes and their dependencies:

```
# diamond.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```



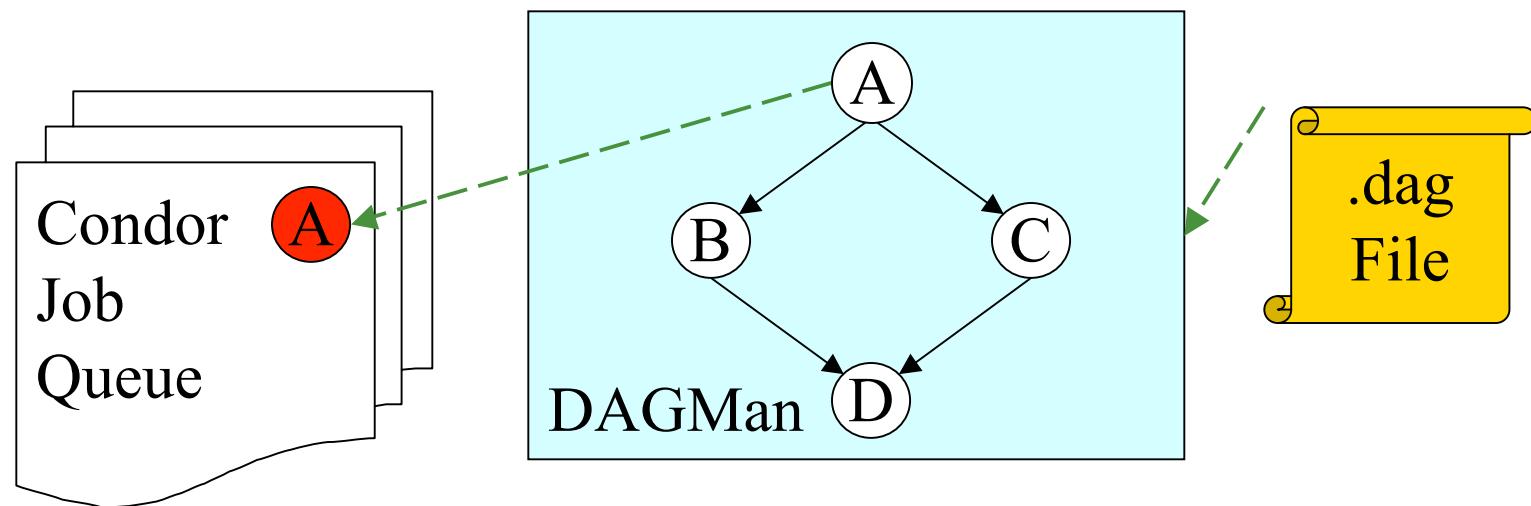
- each node will run the Condor job specified by its accompanying *Condor submit file*

Submitting a DAG

- To start your DAG, just run `condor_submit_dag` with your .dag file, and Condor will start a personal DAGMan daemon which to begin running your jobs:
 - ◊ `condor_submit_dag diamond.dag`
- `condor_submit_dag` submits a job with DAGMan as the executable.
 - This job happens to run on the submitting machine, not any other computer.
- Thus the DAGMan daemon itself runs as a Condor job, so you don't have to baby-sit it.

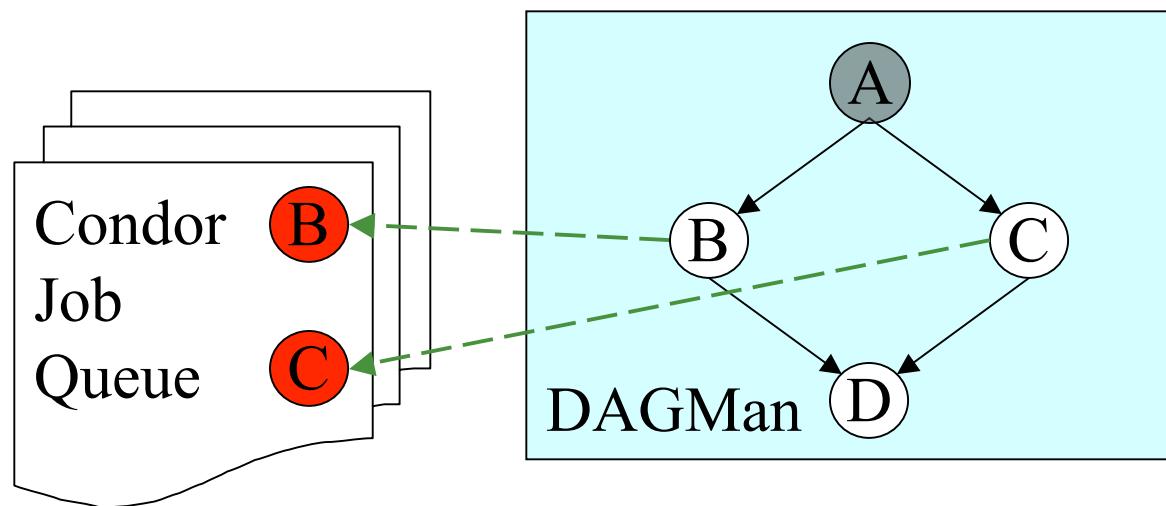
Running a DAG

- DAGMan acts as a “meta-scheduler”, managing the submission of your jobs to Condor based on the DAG dependencies.



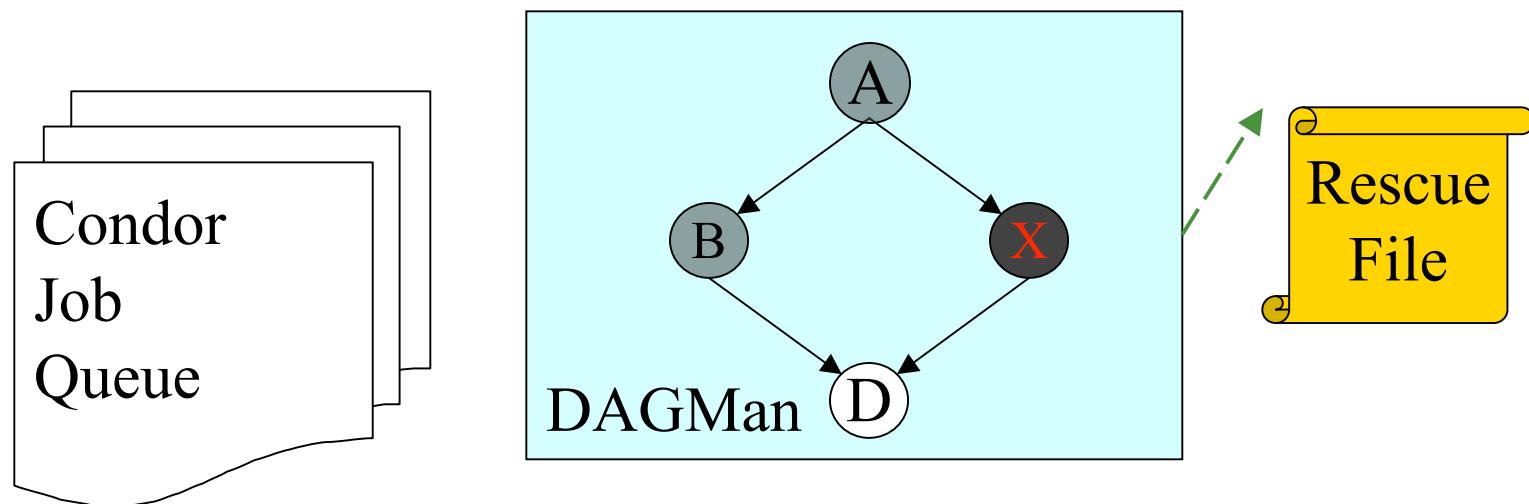
Running a DAG (cont'd)

- DAGMan holds & submits jobs to the Condor queue at the appropriate times.



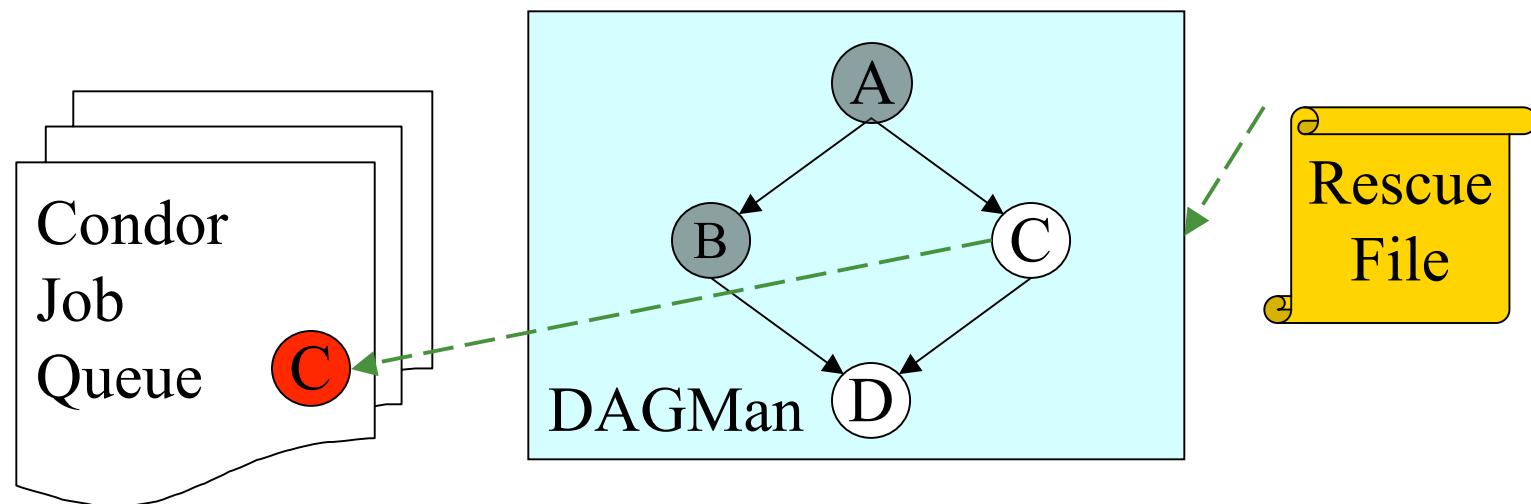
Running a DAG (cont'd)

- In case of a job failure, DAGMan continues until it can no longer make progress, and then creates a “*rescue*” file with the current state of the DAG.



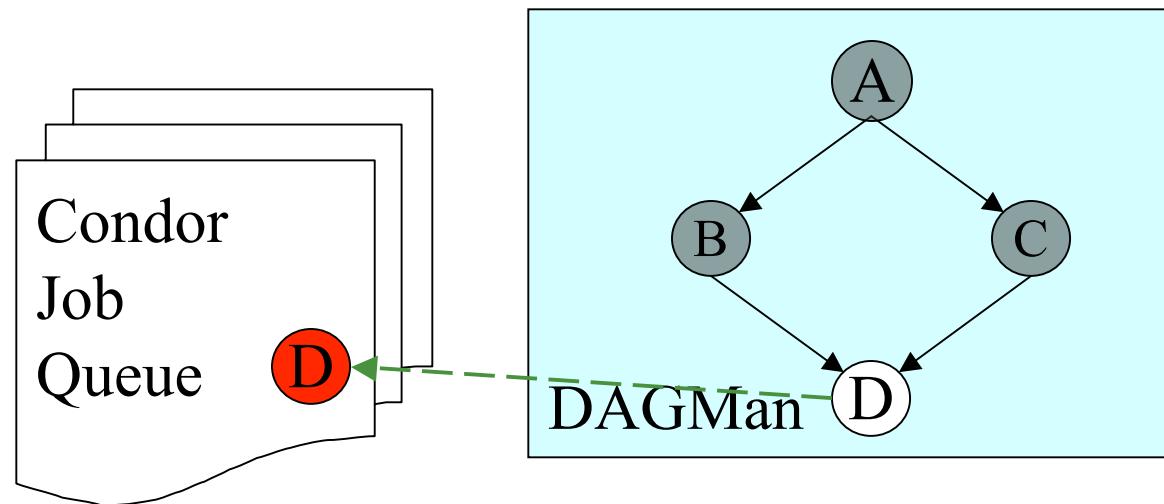
Recovering a DAG

- Once the failed job is ready to be re-run, the rescue file can be used to restore the prior state of the DAG.



Recovering a DAG (cont'd)

- Once that job completes, DAGMan will continue the DAG as if the failure never happened.



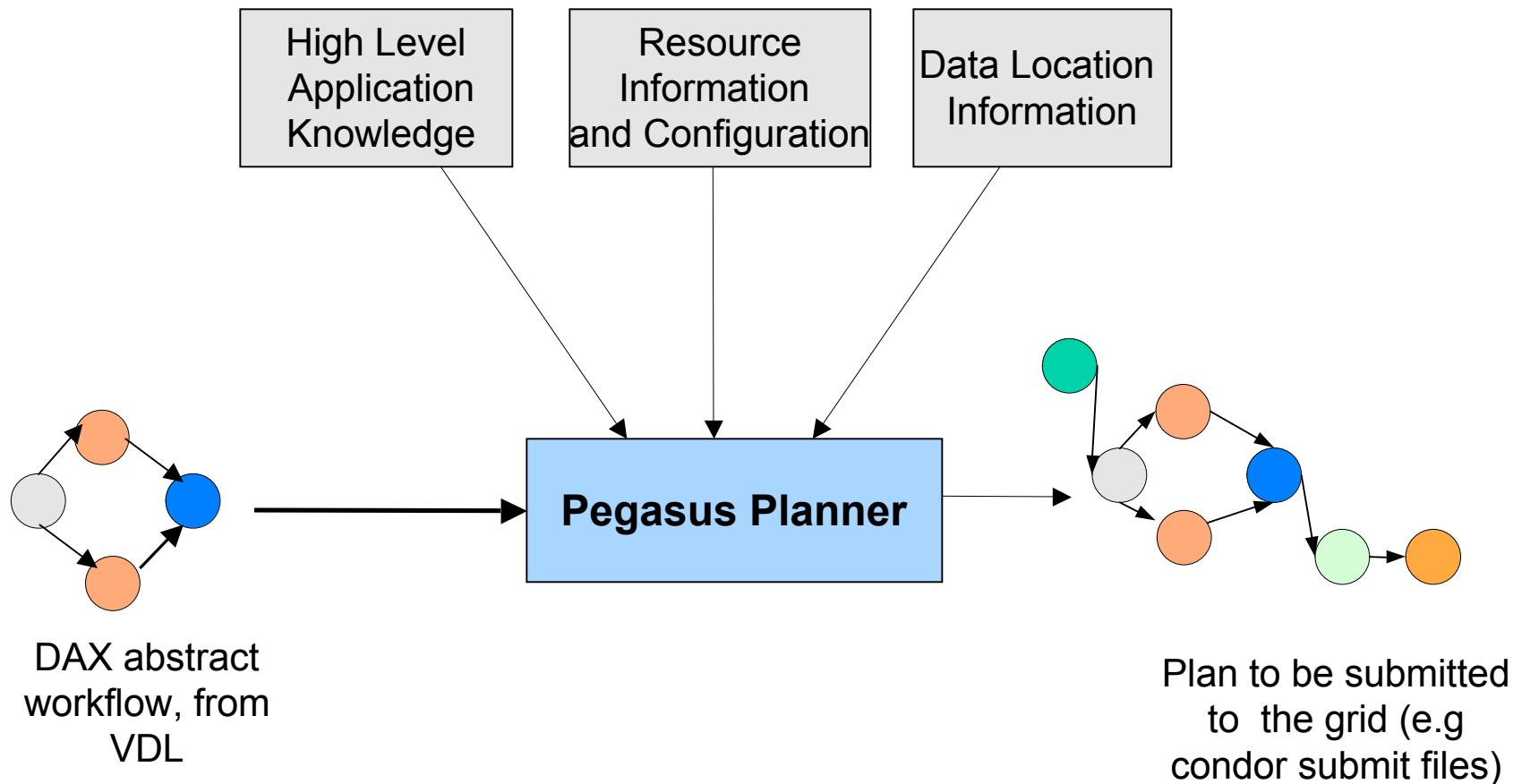
VDS

- **Virtual Data System**
 - Virtual Data Language (VDL)
 - > A language to express workflows
 - Pegasus planner
 - > Decides how the workflow will run
 - Virtual Data Catalog (VDC)
 - > Stores information about workflows
 - > Stores provenance of data

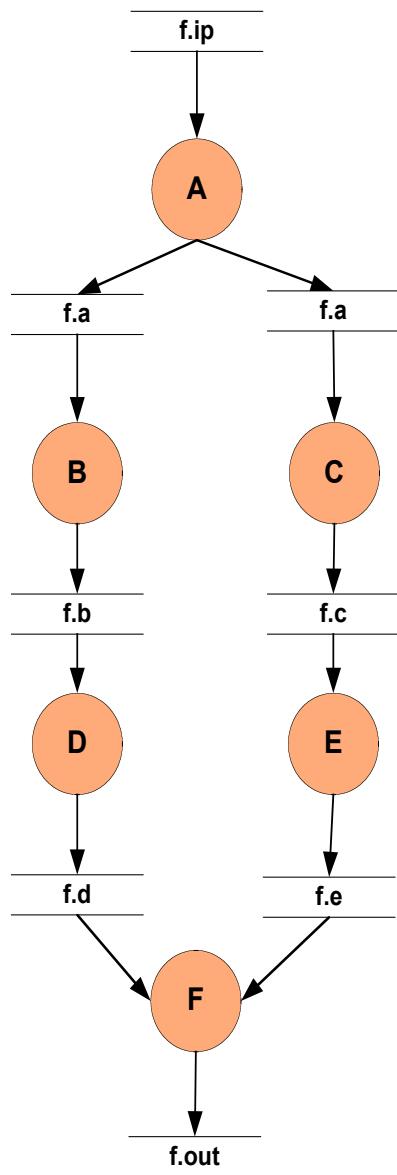
Virtual Data Process

- Describe data derivation or analysis steps in a high-level workflow language (VDL)
- VDL is cataloged in a database for sharing by the community
- Grid workflows are generated from VDL
- Provenance of derived results stored in database for assessment or verification

Planning with Pegasus

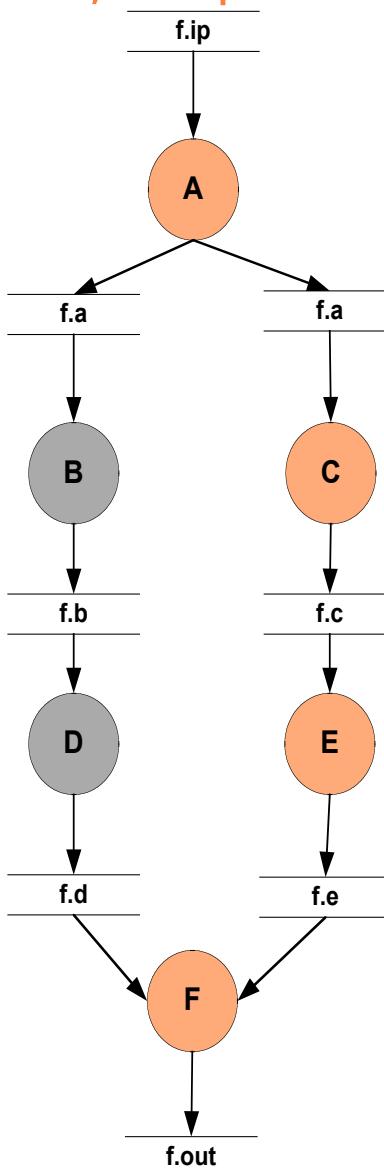


Abstract to Concrete, Step 1: Workflow Reduction

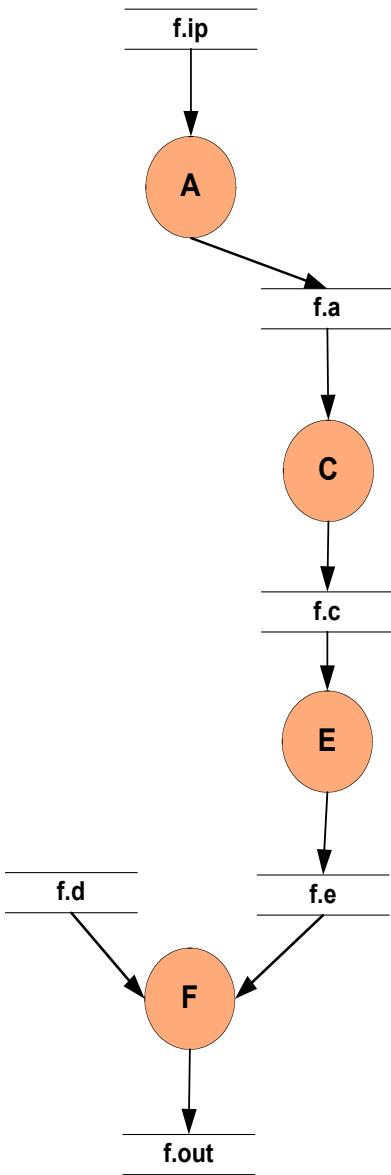


Abstract Workflow

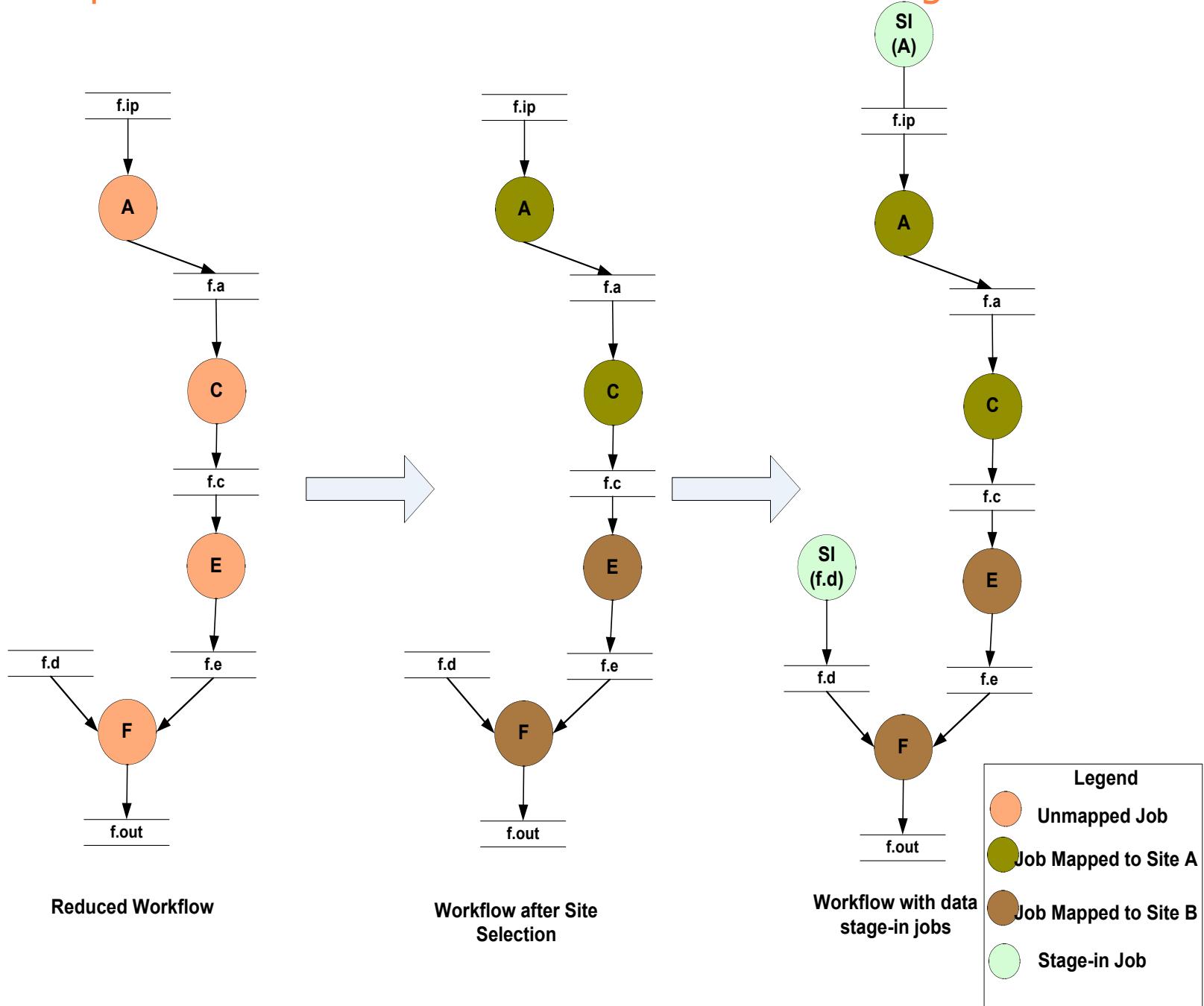
File $f.d$ exists somewhere.
Reuse it.
Mark Jobs D and B to delete



Delete Job D and Job B

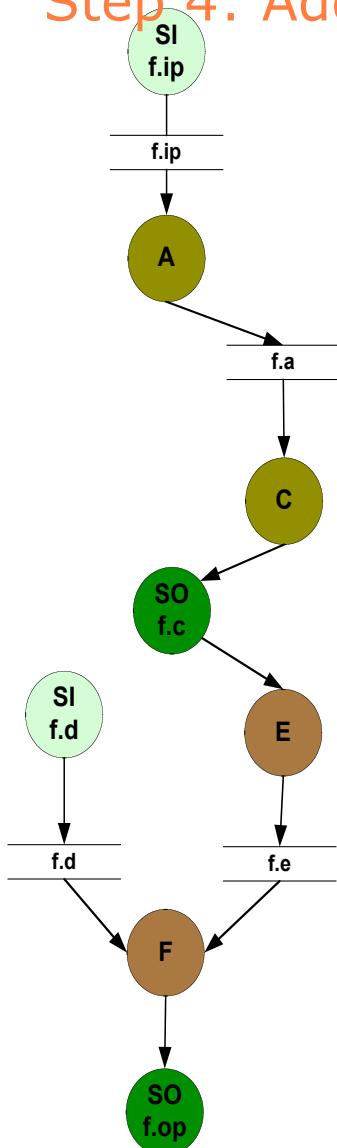


Step 2: Site Selection & Addition of Data Stage-in Nodes

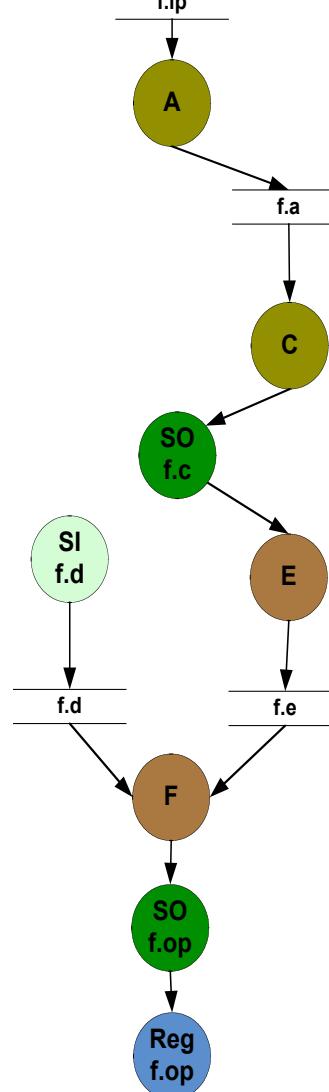
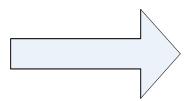


Step 3: Addition of Data Stage-out Nodes

Step 4: Addition of Replica Registration Jobs



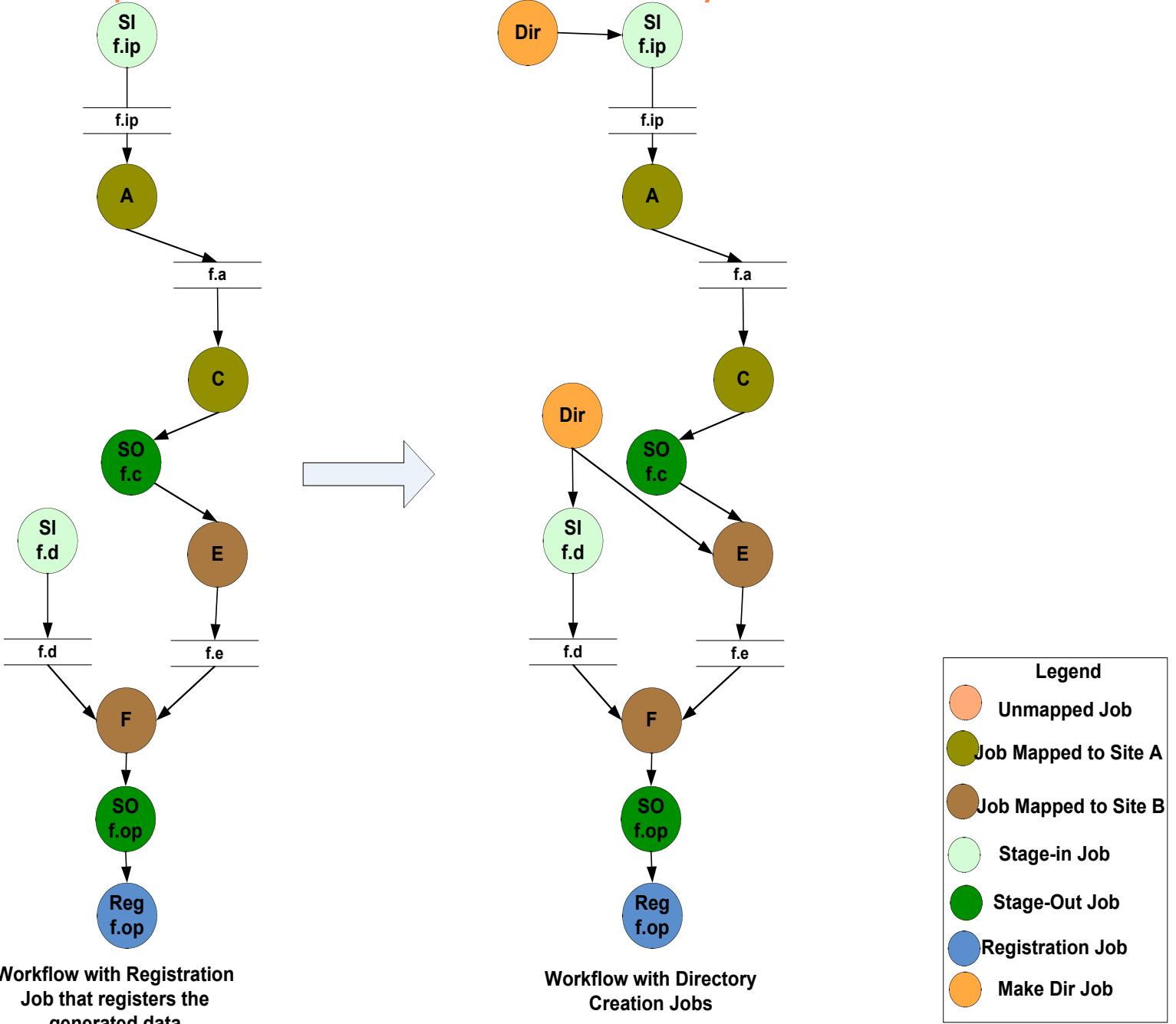
Workflow with Data Stage out Jobs to final output site



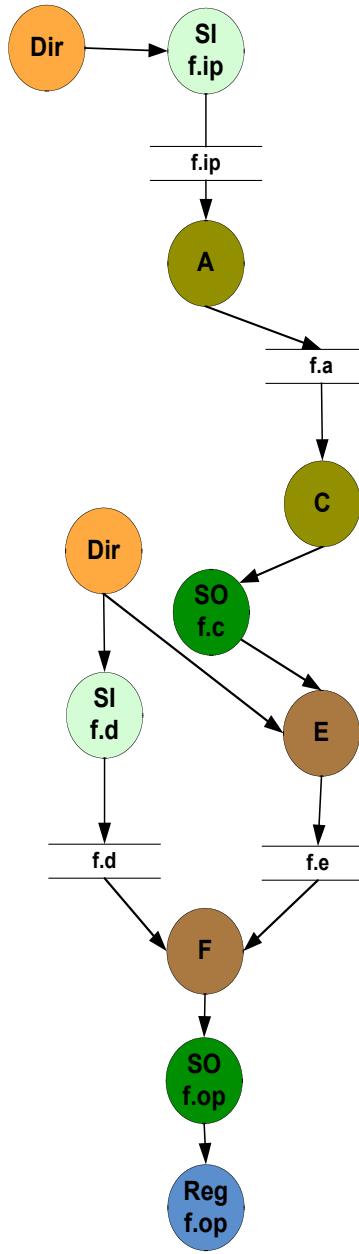
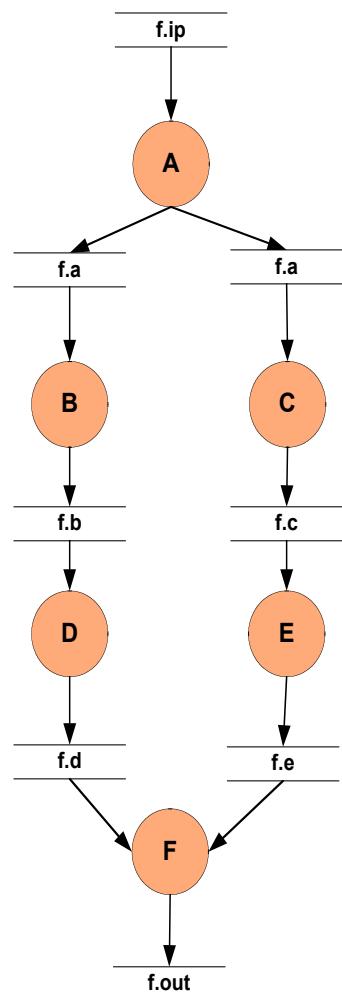
Workflow with Registration Job that registers the generated data

Legend	
	Unmapped Job
	Job Mapped to Site A
	Job Mapped to Site B
	Stage-in Job
	Stage-Out Job
	Registration Job

Step 5: Addition of Job-Directory Creation



Final Result of Abstract-to-Concrete Process



Legend	
Unmapped Job	(Orange)
Job Mapped to Site A	(Dark Green)
Job Mapped to Site B	(Brown)
Stage-in Job	(Light Green)
Stage-Out Job	(Dark Green)
Registration Job	(Blue)
Make Dir Job	(Orange)

Swift System

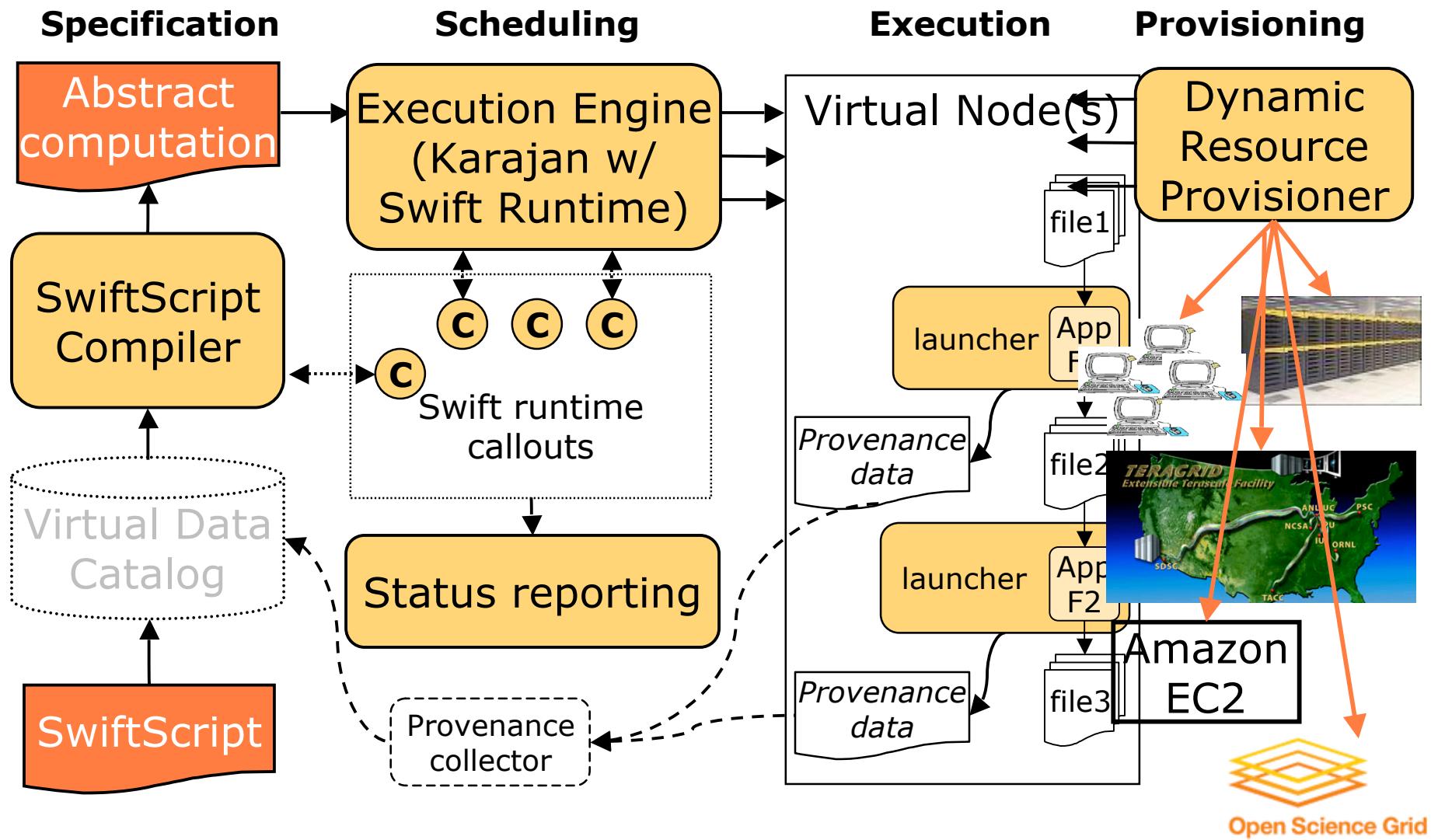
- Clean separation of logical/physical concerns
 - **XDTM** specification of logical data structures
- + Concise specification of parallel programs
 - **SwiftScript**, with iteration, etc.
- + Efficient execution on distributed resources
 - Lightweight threading, dynamic provisioning, Grid interfaces, pipelining, load balancing
- + Rigorous provenance tracking and query
 - Virtual data schema & automated recording
- **Improved usability and productivity**
 - Demonstrated in numerous applications

AIRSN Program Definition

```
(Run snr) functional ( Run r, NormAnat a,
    Air shrink ) {
    Run yroRun = reorientRun( r , "y" );
    Run roRun = reorientRun( yroRun , "x" ),
    Volume std = roRun[0];
    Run rndr = random_select( roRun, 0.1 );
    AirVector rndAirVec = align_linearRun( rndr, std, 12, 1000, 1000, "81 3 3" );
    Run reslicedRndr = resliceRun( rndr, rndAirVec, "o", "k" );
    Volume meanRand = softmean( reslicedRndr, "y", "null" );
    Air mnQAAir = alignlinear( a.nHires, meanRand, 6, 1000, 4, "81 3 3" );
    Warp boldNormWarp = combinewarp( shrink, a.aWarp, mnQAAir );
    Run nr = reslice_warp_run( boldNormWarp, roRun );
    Volume meanAll = strictmean( nr, "y", "null" )
    Volume boldMask = binarize( meanAll, "y" );
    snr = gsmoothRun( nr, boldMask, "6 6 6" );
}
```

```
(Run or) reorientRun (Run ir,
    string direction) {
    foreach Volume iv, i in ir.v {
        or.v[i] = reorient(iv, direction);
    }
}
```

Swift Architecture





Based on:
The Virtual Data System -
*a workflow toolkit for
science applications*

*OSG Summer Grid Workshop
Lecture 8
June 29, 2006*



Based on: Fast, Reliable,
Loosely Coupled Parallel
Computation

Tiberiu Stefan-Praun
Computation Institute

University of Chicago & Argonne National Laboratory
tiberius@ci.uchicago.edu



www.ci.uchicago.edu/swift

Based on: Lecture 5

Grid Resources and Job Management

Jaime Frey

Condor Project,

University of Wisconsin-
Madison

jfrey@cs.wisc.edu



Open Science Grid



Grid Summer
Workshop

Acknowledgements

*The technologies and applications described here
were made possible by the following projects and support:*

GriPhyN, iVDGL, the Globus Alliance, the Open
Science Grid and QuarkNet, supported by
The National Science Foundation



The Globus Alliance, PPDG, OSG and QuarkNet,
supported by the US Department of Energy,
Office of Science



Support was also provided by
NVO, NIH, and SCEC