Characterization of Scientific Workflows

Shishir Bharathi Ann Chervenak Ewa Deelman Gaurang Mehta Mei-Hui Su
Karan Vahi
USC Information Sciences Institute
Marina del Rey, CA
{shishir, annc, deelman, gmehta, mei, vahi}@isi.edu

Abstract

Researchers working on the planning, scheduling and execution of scientific workflows need access to a wide variety of scientific workflows to evaluate the performance of their implementations. We describe basic workflow structures that are composed into complex workflows by scientific communities. We provide a characterization of workflows from five diverse scientific applications, describing their composition and data and computational requirements. We also describe the effect of the size of the input datasets on the structure and execution profiles of these workflows. Finally, we describe a workflow generator that produces synthetic, parameterizable workflows that closely resemble the workflows that we characterize. We make these workflows available to the community to be used as benchmarks for evaluating various workflow systems and scheduling algorithms.

1. Introduction

The availability of a library of scientific workflow benchmarks would be extremely valuable for the development and comparison of workflow management systems. Such benchmarks would allow direct comparisons of different workflow systems and would facilitate the evaluation of scheduling and data management algorithms within workflow systems. To date, few scientific workflows have been made available for general use, including providing access to their data and codes. The Montage astronomy workflow [1] is one exception that has been widely used to evaluate workflow algorithms and systems. However, access to a wider variety of scientific workflow benchmarks is needed, since the optimizations performed by workflow planners, execution systems and schedulers may be more suited to some workflow patterns and structures than others.

The goal of our work is to provide an initial library of scientific workflows for use by the research community. We hope this effort will lead to the contribution of workflows from other scientific communities for use in the development of new and existing workflow systems.

In this paper, we provide detailed characterizations of five scientific workflows that include massively parallel workflows that process large amounts of data, pipelined applications that split up input datasets and operate on different chunks in parallel, and workflows that have a relatively fixed structure and perform identical analyses on multiple input datasets. The workflows are taken from diverse application domains such as astronomy, biology, gravitational physics and earthquake science. While we do not claim that these workflows represent the full spectrum of scientific workflows, we believe this work is a useful first step in characterizing that spectrum.

This paper is organized as follows. We begin in Section 2 with an overview of the scientific workflows that we characterize. In Section 3, we discuss the workflow characterization process, identify structures common to many workflows and describe how these are composed to generate complex workflows. In Section 4, we characterize in detail the scientific workflows described in Section 2. We highlight the data and computational requirements of each workflow and discuss how these may change when smaller or larger datasets are processed by similar workflows. In Section 5, we discuss our workflow generator, which creates synthetic, parameterizable workflows with similar characteristics to the characterized scientific workflows. We make these synthetic workflows available to the application community for study and comparison in the DAX (Directed Acyclic Graph in XML) format, which can be directly ingested by some workflow management systems such as Pegasus [2] or converted into input formats for other workflow systems. Finally, we discuss related work in Section 6 and outline directions for future work in Section 7.

2. Overview of scientific workflows

We now briefly describe the scientific workflows that we have characterized.

2.1. Montage

Montage [1] was created by the NASA/IPAC Infrared Science Archive as an open source toolkit that can be used to generate custom mosaics of the sky using input images in the Flexible Image Transport System (FITS) format. During the production of the final mosaic, the geometry of the output is calculated from the geometry of the input images. The inputs are then re-projected to be of the same spatial scale and rotation. The background emissions in the images are then corrected to be of the same level in all images. The re-projected, corrected images are co-added to form the final mosaic.

The Montage application has been represented as a workflow that can be executed in Grid environments such as the TeraGrid [3].

2.2. CyberShake

The CyberShake workflow is used by the Southern California Earthquake Center (SCEC [4]) to characterize earthquake hazards in a region using the Probabilistic Seismic Hazard Analysis (PSHA) technique. Given a region of interest, an MPI based finite difference simulation is performed to generate Strain Green Tensors (SGTs). From the SGT data, synthetic seismograms are calculated for each of the ruptures that were predicted. Once this is done, spectral acceleration and probabilistic hazard curves are generated.

CyberShake workflows resulting in a total of more than 800,000 jobs have been executed using the Pegasus Workflow Management System (Pegasus-WMS) on the TeraGrid. Additional details are available in Deelman et al [5] and Callaghan et al [6].

2.3. Epigenomics

The USC Epigenome Center is currently involved in the mapping of the epigenetic state of human cells on a genome-wide scale.

The Epigenomics workflow is essentially a data processing pipeline that uses the Pegasus Workflow Management System to automate the execution of the various genome sequencing operations. The DNA sequence data generated by the Illumina-Solexa [7] Genetic Analyzer system is split into several chunks that can be operated on in parallel. The data in each chunk is converted into a file format that can be used by the Maq [8] system. The rest of the operations involve the filtering out of noisy and contaminating sequences, mapping sequences into the correct location in a reference genome, generating a global map and then identifying the sequence density at each position in the genome. This workflow is being used by the Epigenome Center in the processing of production DNA methylation and histone modification data.

2.4. LIGO Inspiral Analysis Workflow

The Laser Interferometer Gravitational Wave Observatory (LIGO) is attempting to detect gravitational waves produced by various events in the universe as per Einstein's theory of general relativity.

The LIGO Inspiral Analysis Workflow [9] is used to analyze the data obtained from the coalescing of compact binary systems such as binary neutron stars and black holes. The time-frequency data from any event for each of the three LIGO detectors is split into smaller blocks for analysis. For each block, the workflow generates a subset of waveforms belonging to the parameter space and computes the matched filter output. If a true inspiral has been detected, a trigger is generated that can then be checked with triggers for the other detectors. Several additional consistency tests may also be performed.

2.5. SIPHT

The bioinformatics project at Harvard University is conducting a wide search for small untranslated RNAs (sRNAs) that regulate several processes such as secretion or virulence in bacteria.

The sRNA identification protocol using high-throughput technology (SIPHT) program [10] uses a workflow to automate the search for sRNA encoding-genes for all of the bacterial replicons in the National Center for Biotechnology Information (NCBI) database. The kingdom-wide prediction and annotation of sRNA encoding genes involves a variety of individual programs that are executed in the proper order using Condor DAGMan's [11] capabilities. These involve the prediction of Rho-independent transcriptional terminators, BLAST (Basic Local Alignment Search Tools) comparisons of the inter genetic regions of different replicons and the annotations of any sRNAs that are found.

3. Characterization of workflows

Next, we describe basic concepts and terminologies that we use to characterize scientific workflows.

In Figure 1, we show some of the basic structures or components of scientific workflows. The final workflow is usually composed of several such components. Similar components have been considered for control flow structures, for example by the Workflow Patterns Initiative [12].

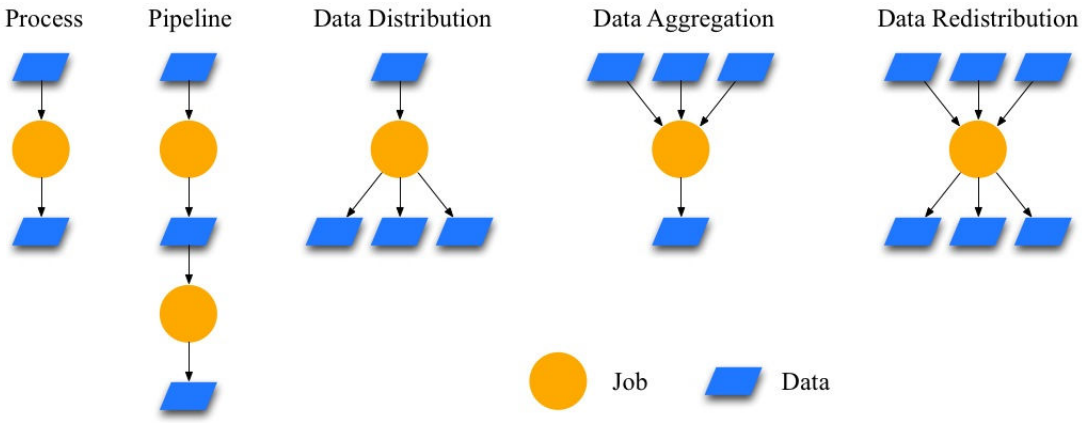

Figure 1. Basic workflow structures

However, in this paper our discussion is restricted to a study of how data partitioning and aggregation impacts the overall structure and execution of scientific workflows.

The simplest structure is the *process* structure that operates on some input data to produce an output. Several such data processing jobs can be combined sequentially to produce the *pipeline* structure. This structure can be found in several workflows and therefore, we describe it as a unique structure. In this case, each job in the pipeline operates on the output of the previous stage and the output produced is fed as input to the next stage in the pipeline. *Data distribution* jobs serve two purposes: they may either produce output data that are consumed by multiple jobs or they may operate on large datasets and divide or "chunk" them into smaller subsets to be processed by other jobs in the workflow. Since the latter usage is quite common, we also refer to it as *data partitioning* in the rest of this paper. If the partitioning involves computation in addition to creating smaller chunks of data, data partitioning jobs may consume a lot of time on the compute resource. However, partitioning leads to increased parallelism in the later stages of the workflow, and indeed this is the main reason for partitioning the data. *Data aggregation* jobs aggregate and process the outputs of several individual jobs and generate a combined data product. As data aggregation jobs operate on several individual data items, they can potentially consume a lot of time on compute resources. Additionally, such jobs may represent a reduction in the parallelism of the workflow. In some cases, data aggregated from a previous stage are redistributed to multiple jobs in a following stage. Even though *data redistribution* jobs represent a potential bottleneck, the parallelism is once again increased in future stages. Such data redistribution jobs can be found in several scientific workflows and represent a synchronization point from the data processing perspective.

4. Characterization of example workflows

In this section, we provide characterizations of each of the workflows mentioned in Section 2 and describe how jobs in each workflow relate to the simple workflow structures described in Section 3. Scientific workflows executed on the Grid are quite large in some cases. To better depict individual jobs and their relationships, we visualize each workflow type using smaller synthetic workflows generated using our workflow generator. However, note that the execution times and data sizes we refer to in this section are obtained from actual executions of the workflows on the Grid.

4.1. Montage

In Figure 2, we show a relatively small (20 node) Montage-like workflow generated by the workflow generator. The number of inputs processed by the workflow may increase over time as more images of a particular region of the sky are available. As such, the structure of the workflow changes to accommodate the increase in the number of inputs, which also translates to an increase in the number of computational jobs.

The number of mProjectPP jobs (which re-project the input image) is equal to the number of input FITS images processed. The outputs are the reprojected image and an "area" image that consists of the fraction of the image that belongs in the final mosaic. These are then processed together in subsequent steps. An mDiffFit job computes a difference for each pair of overlapping images. The number of mDiffFit jobs in the workflow therefore depends on how the input images overlap. The difference images are then fitted using a least squares algorithm by the mConcatFit job. The mConcatFit job fits the description of a data aggregation job, introduced in Section 3, and is also a computationally intensive job. Next, a correction to be applied

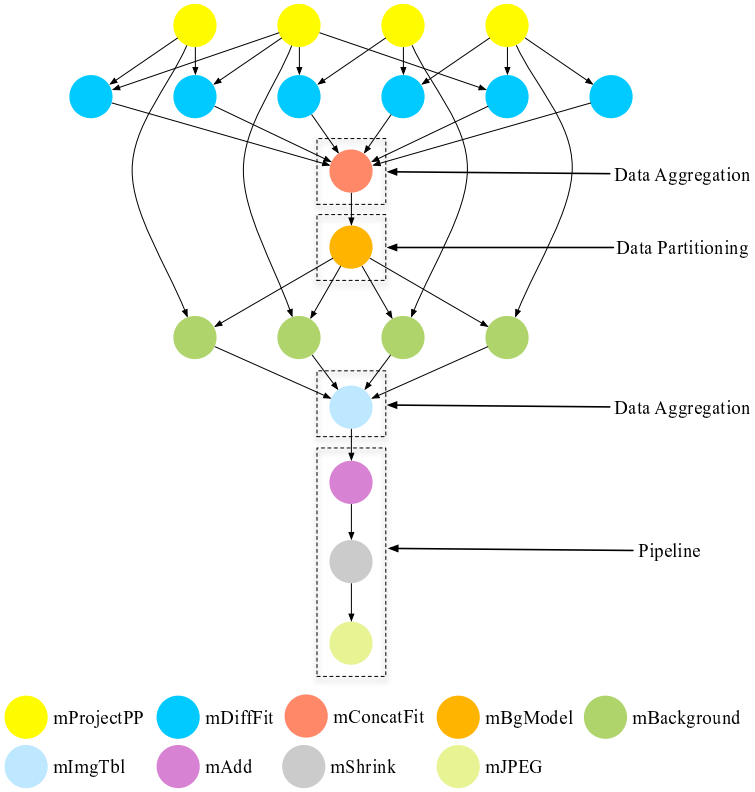

Figure 2. Montage workflow

to each image to obtain a good global fit is determined by the mBgModel job. The background correction is applied to each individual image by the mBackground jobs. The *mConcatFit* and *mBgModel* jobs are individually data aggregation and data partitioning jobs respectively. However, together they can also be considered as a data redistribution point. Note that there is not a lot of data being partitioned in this case. Rather, the same background correction is applied to all images. The mImgTbl job aggregates metadata from all the images and creates a table that may be used by other jobs in the workflow. As such, it represents a simple data aggregation step. The mAdd job co-adds all the reprojected images to generate the final mosaic in FITS format as well as an area image that may be used in further computation. The mAdd job is the most computationally intensive job in the workflow. The size of the FITS image is reduced by the mShrink job by averaging blocks of pixels. The shrunken image is then converted to JPEG format by the mJPEG job.

In Table 1, we provide the runtimes from an execution of a 1.0 degree Montage workflow on the Grid. We provide the total sizes (i.e. the sum of the sizes of all files) of inputs and outputs consumed and generated by each job. Note that the same input data item may be consumed by multiple jobs.

4.2. CyberShake

In Figure 3, we show a small CyberShake-like workflow. The workflow, while relatively simple in structure, can be used to perform significant amounts of computation on extremely large datasets. Strain Green Tensor (SGT) data generated from finite simulations are maintained in the form of large "master" SGT files for x and y dimensions. Such master SGT data is generated for a number of sites, each of which represent the impact of an earthquake hazard as measured at a given location. The ExtractSGT jobs in the workflow extract the SGTs pertaining to a given $\langle source, rupture \rangle$ pair from the master SGT files for the site. ExtractSGT jobs may therefore be considered as data partitioning jobs. Synthetic seismograms are generated for each variation of the $\langle source, rupture \rangle$ pair by the SeismogramSynthesis jobs. Peak intensity values, in particular the spectral acceleration, are calculated by the PeakValueCalcOkaya jobs for each synthetic seismogram. The resulting synthetic seismograms are collected and compressed by the ZipSeismograms and ZipPeakSA jobs to be staged out and archived. These jobs may be considered as simple data aggregation jobs, although they are not followed by further processing.

Of the computational jobs, seismogram synthesis jobs

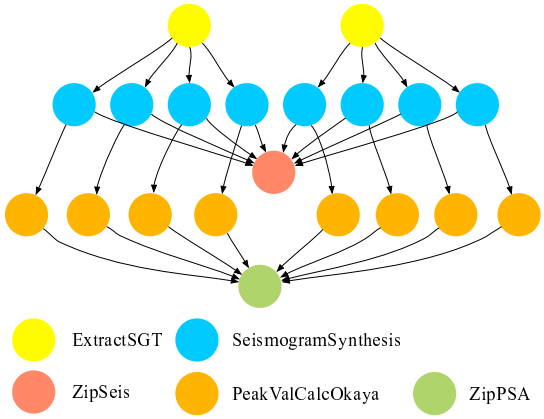

Figure 3. CyberShake workflow

are the most computationally intensive. However, when the workflow is executed on the Grid, due to the large sizes of the master SGT files, ExtractSGT jobs may also consume a lot of time on compute resources. Additionally, since a lot of data may be generated by the workflow, the ZipSeismograms and ZipPeakSA jobs may also consume a large amount of time in generating the compressed files to be staged out.

In Table 2, we provide execution times of jobs and data sizes from the execution of a subset of a relatively large (more than 800,000 jobs) CyberShake workflow on the Grid.

4.3. Epigenomics

In Figure 4, we show an example of a synthetic epigenomics workflow generated by our workflow generator. The epigenomics workflow represents a largely pipelined application with multiple pipelines operating on distinct chunks of data. This workflow is specific to data produced by the Illumina-Solexa genetic analyzer and the Maq mapping software. The overall input to the workflow is sequence data obtained for multiple "lanes" from the genetic analysis process. The information from each lane is split into multiple chunks by the fastqSplit jobs. The number of splits generated depends on the chunking factor used on the input data. The fastqSplit jobs represent data partitioning jobs in the workflow. The filterContams jobs then filter out noisy and contaminated data from each of the chunks. The data in each chunk is then converted to a format understood by later maq programs by the maq sol2sanger utility. For faster processing and reduced diskspace usage, the data is then converted to the binary fastQ format by maq's fastq2bfq utility. Next the remaining sequences are aligned with the reference genome by the map utility. The results of individual map processes are combined using one or more stages of mapMerge jobs, which are data aggregation jobs. The maqindex utility operates on the merged alignment file and retrieves reads about a specific region and the pileup utility displays the alignment in a specific "pileup" format.

As might be expected, the map jobs that align sequences with the reference genome are the most computationally intensive followed by the pileup jobs that work on the entire aligned output. The performance of other jobs in the pipeline mainly depends on the amount of data in each of the individual chunks.

To understand the above notions better, we refer the reader to Table 3 that lists job runtimes and sizes of files consumed and generated during an actual execution of the workflow.

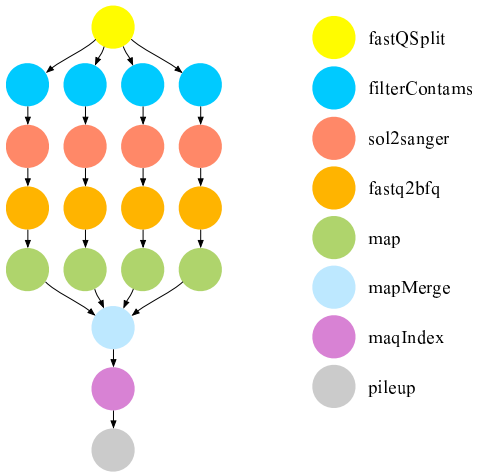

Figure 4. Epigenomics workflow

4.4. LIGO Inspiral Analysis Workflow

In Figure 5, we show an example of a synthetic workflow similar in structure to the LIGO Inspiral Analysis Workflow. In reality, an instance of the workflow is likely to have several hundreds of jobs. The *TmpltBank* jobs, which identify the continuous family of waveforms that belong to the parameter space for each block of the data, can all be executed in parallel once the input data from the LIGO detectors have been split into multiple blocks (each of 2048 seconds [9]). The output of a TmplBank job is a bank of waveform parameters that are used by the matched filtering code in an *Inspiral* job. The triggers produced by multiple Inspiral jobs are tested for consistency by inspiral coincidence analysis jobs, which are denoted by *Thinca* in the example. Since these jobs operate on data obtained from multiple jobs, they can be regarded as data aggregation jobs. The outputs of the Thinca jobs are inputs to the TrigBank jobs that generate template banks out of the triggers. These template banks are then used by the second set of Inspiral jobs, followed by

Table 1. Example of a Montage Execution Profile

Executed on viz-cluster at ISI - 8 nodes with dual Intel Xeon 2.4 GHz processors, 2 GB Memory

Job	Count	Runtime		Inputs		Outputs	
		Mean(s)	Variance	Mean(MB)	Variance	Mean(MB)	Variance
mProject	45	13.59	6.00e-02	4.03	0	7.94	4.14e-04
mDiffFit	107	10.59	1.00e-02	15.88	1.37e-03	0.54	7.92e-02
mConcatFit	1	13.60	0	0.03	0	0.02	0
mBgModel	1	10.88	0	0.03	0	0.00	0
mBackground	45	10.74	3.00e-02	7.95	4.14e-04	7.94	4.14e-04
mImgtbl	1	10.69	0	357.27	0	0.01	0
mAdd	1	30.34	0	357.28	0	330.86	0
mShrink	1	12.26	0	165.43	0	6.62	0
mJPEG	1	10.96	0	6.62	0	0.32	0

Table 2. Example of a CyberShake Execution Profile

Executed on OSG cluster at PSU [13]

Job	Count	Runtime		Inputs		Outputs	
		Mean(s)	Variance	Mean(MB)	Variance	Mean(MB)	Variance
extract_sgt	19	355.31	1.80e+05	38,786.64	5.78e+00	441.97	9.85e+04
seismogram_synthesis	5,726	63.35	4.38e+02	791.46	5.20e+04	0.02	0
ZipSeismograms	1	942.37	0	0.00	0	2.05	0
PeakValCalc_Okaya	5,726	1.36	3.91e+00	0.02	0	0.00	0
ZipPeakSA	1	398.79	0	0.00	0	118.79	0

Table 3. Example of an Epigenomics Execution Profile

Executed on USC HPCC Cluster using 8 nodes with dual Intel quad core 2GHz cpus, 16 GB memory

Job	Count	Runtime		Inputs		Outputs	
		Mean(s)	Variance	Mean(MB)	Variance	Mean(MB)	Variance
fastqSplit	2	41.78	1.82e+02	462.20	6.92e+03	462.20	6.92e+03
filterContams	146	1.15	5.00e-01	6.33	2.37e-02	0.00	0
sol2sanger	146	0.24	1.00e-02	3.16	1.14e-02	2.41	6.55e-03
fast2bfq	146	0.39	2.00e-02	2.41	6.55e-03	0.53	2.01e-03
map	146	9,635.01	1.66e+07	2,964.24	2.01e-03	0.58	3.18e-03
mapMerge	3	23.54	3.26e+01	52.67	2.99e+02	48.79	3.38e+02
chr21 (maqIndex)	1	3.57	0	72.36	0	0.91	0
pileup	1	3,269.73	0	2,964.62	0	3.13	0

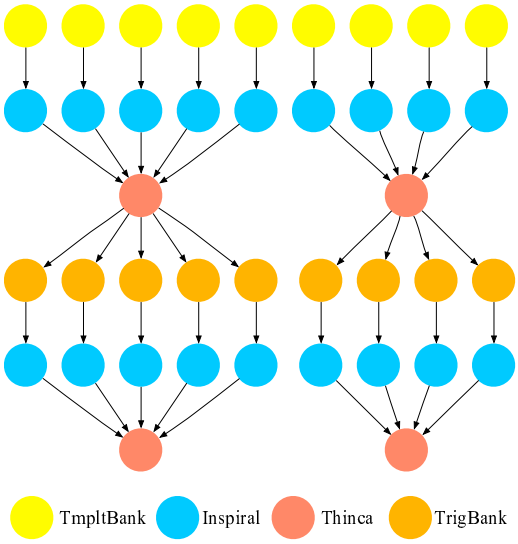

Figure 5. LIGO workflow

another set of Thinca jobs. Therefore, the first set of Thinca jobs can also be regarded as data redistribution jobs.

Inspiral jobs executing the matched filter to generate trigger are the most computationally intensive jobs in the workflow and there is considerable variance in the execution times of these jobs. Even though the Thinca jobs perform consistency checks on data aggregated from multiple Inspiral jobs, they are not as computationally intensive as the Inspiral jobs. For comparison purposes, we list runtimes for jobs from an actual execution in Table 4 and also provide the total sizes of inputs consumed and outputs generated by each type of job.

4.5. SIPHT

A SIPHT-like workflow is shown in Figure 6. All SIPHT workflows have almost identical structures, and larger workflows can be generated by composing independent smaller workflows. The only difference in the structures of any two instances is in the number of *Patser* jobs that depends on inputs describing transcription factor binding sites (TFBSs). The results of these Patser jobs are concatenated by the *Patser_concate* job, which is a data aggregation job. There are various *BLAST* jobs in the workflow that compare different combinations of sequences. The initial BLAST job (that does not depend on any other job) and the Blast_QRNA job operate on all possible partner intergenetic regions (IGRs) from other suitable replicons. Even though it is not apparent in Figure 6, these BLAST jobs operate on hundreds of data files and can therefore be regarded as data aggregation jobs. There are three jobs in the workflow that search for transcription terminators - *FindTerm*, *RNAMotif* and *Transterm*. The sRNA prediction is performed by the *SRNA* job that operates on the outputs of the above jobs as well as the output of the initial BLAST job. The output of this job is used by the other BLAST jobs. Therefore, the SRNA job may be regarded as a data redistribution job. The *SRNA_Annotate* job annotates candidate sRNA loci that were found, for multiple features such as its conservation in other bacterial strains, its association with putative TFBSs, and its homology to other previously identified sRNAs [10]. Therefore, this can be regarded as a data aggregation job.

Of all the jobs in the workflow, the BLAST jobs that compare sequences between several replicon pairs are the most computationally intensive. The next computationally intensive is the sRNA prediction job followed by the jobs that identify transcription terminators. In Table 5, we list the execution execution profile statistics collected from an earlier execution of the SIPHT workflow.

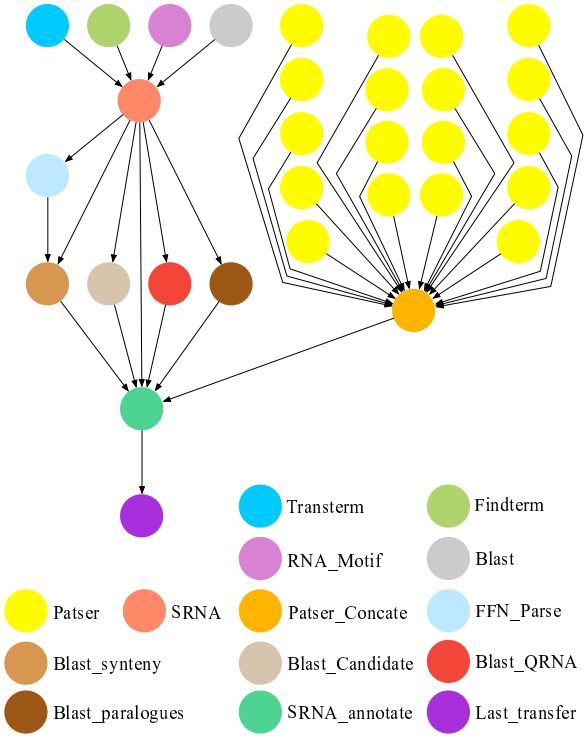

Figure 6. SIPHT workflow

5. Generation of workflows for simulation

To facilitate evaluation of workflow algorithms and systems on a range of workflow types and sizes, we have developed a workflow generator. This generator uses the information gathered from actual executions of scientific workflows on the Grid as well as our understanding of the processes behind these workflows to generate synthetic work-

Table 4. Example of a LIGO Inspiral Analysis Execution Profile

Executed on NCSA Teragrid [14]

Job	Count	Runtime		Inputs		Outputs	
		Mean(s)	Variance	Mean(MB)	Variance	Mean(MB)	Variance
tmpltbank	34	18.14	1.80e-01	70.81	1.62e-03	0.94	3.27e-04
inspiral	76	460.21	2.97e+05	74.68	2.17e-01	0.30	3.05e-01
thinca	14	5.37	6.00e-02	16.08	6.51e+00	0.03	1.72e-03
trigbank	42	5.11	0	13.62	2.19e-03	0.01	1.25e-04

Table 5. Example of a SIPHT Execution Profile

Executed on a heterogeneous Condor pool at U. Wisconsin, nodes with 4 2.7GHz processors and 2-4 GB memory

Job	Count	Runtime		Inputs		Outputs	
		Mean(s)	Variance	Mean(MB)	Variance	Mean(MB)	Variance
Transterm:2.0.5	1	32.02	0	4.65	0	0.41	0
Findterm	1	975.16	0	4.87	0	45.84	0
RNAMotif:3.0.4	1	43.93	0	4.64	0	1.00	0
Blast	1	3,331.16	0	258.41	0	4.31	0
Patser:3.0.1	17	1.32	1.90e-01	4.30	1.18e-08	0.01	1.96e-05
SRNA	1	306.53	0	22.09	0	5.12	0
Patser_concate	1	0.01	0	0.11	0	0.11	0
FFN_parse	1	1.40	0	4.40	0	0.83	0
Blast_synteny	1	33.05	0	3.30	0	1.35	0
Blast_candidate	1	5.76	0	1.37	0	0.01	0
Blast_QRNA	1	1,344.88	0	258.93	0	4.50	0
Blast_paralogues	1	4.54	0	1.20	0	1.24	0
SRNA_annotate	1	1.90	0	3.43	0	1.10	0

flows resembling those characterized in this paper. Our workflow generator supports various parameters for each of the workflow types that we have discussed, allowing for the generation of workflows of different scale for each type.

The generation of synthetic workflows involves two main steps: identification of individual jobs and their compositions followed by annotation of the workflow with execution times for computational jobs and sizes for data items. We now describe these steps in further detail.

The structure of the generated workflow is determined by the number of inputs to be processed, the number of jobs in the workflow and their composition. The type and compositions of the jobs that are created depend on the workflow type specified by the user. The number of inputs and the number of jobs in the workflow is determined by user specified parameters - either directly or as a function of the scale of the workflow to be generated. For example, in the case of Montage workflows, if the scale of the workflow is specified in terms of the degree of the final image, the generator will first estimate the number of inputs for a workflow of the given degree based on patterns observed in existing workflows of various degrees. The implementation of the workflow generator encodes our understanding of the various components and compositions of each workflow type, allowing us to describe the processing and combination of data items in a manner similar to that observed in real workflows. For example, with Montage workflows, the workflow generator creates an mProjectPP job for each input image, and for each pair of input images, an mDiffFit job is created to represent the calculation of the differences of the reprojected images. As in real Montage workflows, an mConcatFit job that represent the least squares fit of the reprojected images and a mBgModel job that represents the calculation of the background correction are also created. To be consistent with actual Montage workflows, for each mProjectPP job that generates the reprojected image, we include a corresponding mBackground job that represents the application of the background correction to each reprojected image. The final stages of the workflow, denoted by the pipeline structure in Figure 2, represent the creation of the final mosaic. These stages are common to all Montage workflows, and therefore, are also included by the workflow generator.

Once a workflow of the proper structure has been generated, the next step is to estimate the sizes of the data items processed and generated by the workflow. In some cases, (e.g. input images in Montage workflows), all inputs may have the same size. This is duplicated in the workflow generator. The sizes of intermediate and output data products are estimated from the number and types of inputs being processed as well as the scale of the workflow. For example, the size of final mosaic for a 4.0 degree Montage workflow will be calculated to be approximately 16 times as large as the size calculated for a 1.0 degree workflow. Once this is done, the execution times associated with jobs are calculated using estimates from actual executions and scaling them appropriately, depending on the sizes of the input data items being processed. In this way, we ensure that execution times and data sizes associated with individual jobs are consistent with the execution times and data sizes associated with other jobs in the workflow as well as the input parameters specified during the generation of the workflow. For the runtimes of most jobs and the sizes of most data items (except when they are known deterministically), we generate variations in the estimated values using truncated normal distributions. Note that for some workflows, a larger workflow (in terms of jobs) may or may not result in larger execution times and data sizes. For example, each TmpltBank job in the LIGO Inspiral Analysis workflow always operates on a chunk of data representing 2048 seconds of information. The data sizes and execution times for all such jobs will be picked from the same distribution and will not vary with the number of other jobs in the workflow.

The generated workflows are represented in the Directed Acyclic Graph in XML (DAX) format, which is commonly used for workflows planned with Pegasus. The DAX format represents the abstract form of the scientific workflow by listing the jobs that are to be executed, the inputs and outputs for each of these jobs and the control and data flow dependencies between the jobs in the workflow. We include additional annotations that specify the sizes of data items and execution times for jobs in the workflow.

In addition to the workflow generator, we plan to provide a client program that can simulate the execution of each job in the workflow. Data items used in the workflow are typically files and can be created using utilities like "dd" on UNIX like systems. Together the workflow and the various clients can be used to model the executions of real world workflows on the Grid and may be used by different projects to benchmark their workflow systems. Note that the actual data and executables needed to execute these workflows belong to the respective scientific communities and are not provided with the synthetic workflows.

6. Related work

We now briefly discuss related work in the following contexts: characterization of workflow patterns, analysis of workloads, workflow generation and the creation of repositories for the sharing and reuse of scientific workflows.

In the area of identifying basic workflow components, Van der Aalst and Ter Hofstede have started The Workflow Patterns initiative to "provide a conceptual basis for process terminology" [12]. They categorize various perspectives such as control flow, data, resource and exception handling that need to be supported by workflow description and busi-

ness process modeling languages. On their website, they also maintain descriptions of common patterns for each perspective, such as sequence, parallel split, synchronization, etc. for the control flow perspective. In contrast, we focus more on how data is partitioned and aggregated leading to complex workflow structures.

The characterizations of workloads of different scientific applications were performed by Berry et al [15]. Their benchmarks derived from fluid dynamics, molecular dynamics and structural dynamics applications were mainly used to measure hardware performance metrics (e.g. FLOPS) on supercomputers. In this paper we characterize workflows used by scientific applications and describe how synthetic workflows may be used to measure the performance of workflow algorithms and systems.

Next, we describe related work in the area of characterizing workloads in distributed and Grid environments. The Parallel Workload Archive [16] and, more recently, the Grid Workload Archive [17] provide workloads from various parallel and Grid execution environments and can be used in simulations of such environments. These workloads are typically centered around the performance and utilization of computational resources, whereas our goal is to also provide an insight into the overall data processing needs of the application.

Iosup et al. [18] describe the system-wide, virtual organization-wide, and per user characteristics of traces obtained from four real Grids. Once again, such analyses provide significant insight into how Grid environments are used and allows users to develop better models of such environments. Our goals are complementary to such efforts in providing more insight into individual applications.

Finally, we describe efforts related to the creation of repositories aimed at the sharing of workflow components for use by various communities. De Roure et al [19] describe the ^{my}Experiment virtual research environment [20] that enables scientists to share and execute workflows as well as discuss different workflow management systems and best practices. ^{my}Experiment utilizes several Web 2.0 principles to build a community where users can contribute their workflows for reuse by other scientists. It also allows users to launch their workflows from the website using the Taverna [21] workbench. We expect our repository to be used mainly by researchers working on the development of existing and new workflow systems.

Von Laszewski and Kodeboyina [22] describe the framework of a Repository Service for Grid Workflow Components that supports storing and sharing of workflow components defined by the community. In addition, they describe the integration of the repository service into the Karajan [23] scripting language, which allows for items in the repository to be accessed and reused. Again, our focus is in providing synthetic workflows that can be used to evaluate various workflow systems as opposed to generating real workflows that can be used by scientific communities.

7. Conclusions and future work

In this paper, we identified basic data processing structures that can be used to compose complex scientific workflows. We characterized the workflows used by five diverse scientific communities, identified their data and computational needs and described how the workflows change with the amount of data being processed. We described a workflow generator that creates synthetic workflows similar to the ones we characterized and explained how these are annotated with runtimes of computational jobs and sizes of data items. Finally, we make these available to the workflow community for experimentation.

For the future, our main goal is to expand the repository of benchmarks with workflows from other scientific applications and make these available to the community. We hope that in time, the repository will expand to better represent the full spectrum of scientific workflows. The synthetic workflows described in this paper are available at http://vtcpc.isi.edu/pegasus/index.php/WorkflowGenerator.

Acknowledgments. We are grateful to the personnel of various scientific communities for providing detailed information about their workflows and statistics from executions of those workflows: Ben Berman from USC for the Epigenomics workflow, Bruce Berriman and John Good from IPAC, Caltech for the Montage workflows, Scott Callaghan, Thomas Jordan, Gideon Juve, Phil Maechling, David Meyers from USC for CyberShake workflows, Jonathan Livny from Harvard for SIPHT workflows, Duncan Brown, Patrick Brady, Scott Koranda from UW Milwaukee and Kent Blackburn, Britta Daudert, Chad Henna from Caltech for LIGO workflows.

This work was supported by the Department of Energy's SciDAC II program under grant DE-FC02-06ER25757 (Center for Enabling Distributed Petascale Science) and by the National Science Foundation under grant CCF-0725332 (SciFlow).

References

[1] "Montage: An astronomical image engine." [Online]. Available: http://montage.ipac.caltech.edu

[2] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. P. M.-H. Su, K. Vahi, and M. Livny, "Pegasus: Mapping scientific workflows onto the grid," in *Across Grid Conference*, 2004.

[3] G. B. Berriman, E. Deelman, J. Good, J. Jacob, D. S. Katz, C. Kesselman, A. Laity, T. A. Prince, G. Singh, and M. Su, "Montage: A grid enabled engine for delivering custom science-grade mosaics on demand," in *SPIE*, 2004.

[4] "Southern california earthquake center." [Online]. Available: http://www.scec.org

[5] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T. Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta, D. Okaya, K. Vahi, and L. Zhao, "Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example," in *e-Science*. IEEE Computer Society, 2006, p. 14.

[6] S. Callaghan, P. Maechling, E. Deelman, K. Vahi, G. Mehta, G. Juve, K. Milner, R. Graves, E. Field, D. Okaya, D. Gunter, K. Beattie, and T. Jordan, "Reducing time-to-solution using distributed high-throughput mega-workflows - experiences from SCEC CyberShake," in *Fourth IEEE International Conference on e-Science (e-Science 2008), 10-12 December 2008 in Indianapolis, Indiana, USA*.

[7] "Illumina." [Online]. Available: http://www.illumina.com/

[8] "Maq: Mapping and assemby with qualities." [Online]. Available: http://www.maq.sourceforge.net

[9] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, "A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis," in *Workflows for eScience*. Springer-Verlag, 2006.

[10] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas," *PLoS ONE*, vol. 3, no. 9, p. e3197, Sep 2008.

[11] "Dagman: Directed acyclic graph manager." [Online]. Available: http://www.cs.wisc.edu/condor/dagman

[12] "The workflow patterns initiative." [Online]. Available: http://www.workflowpatterns.com

[13] "The pleiades computational cluster." [Online]. Available: http://ligo.aset.psu.edu/pleiades/osg/

[14] "Ncsa teragrid linux cluster." [Online]. Available: http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/TGIA64LinuxCluster/

[15] M. W. Berry, "Scientific workload characterization by loop-based analyses," *SIGMETRICS Performance Evaluation Review*, vol. 19, no. 3, pp. 17–29, 1992.

[16] "Parallel workloads archive." [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[17] "Grid workload archive." [Online]. Available: http://gwa.ewi.tudelft.nl/pmwiki/

[18] A. Iosup, C. Dumitrescu, D. H. J. Epema, H. Li, and L. Wolters, "How are real grids used? the analysis of four grid traces and its implications," in *7th IEEE/ACM International Conference on Grid Computing*. IEEE, 2006, pp. 262–269.

[19] D. D. Roure, C. A. Goble, and R. Stevens, "Designing the myexperiment virtual research environment for the social sharing of workflows," in *eScience2007*. IEEE Computer Society, 2007, pp. 603–610.

[20] "The ^{my}experiment virtual research environment." [Online]. Available: http://www.myexperiment.org

[21] D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic Acids Research*, vol. 34, no. Web-Server-Issue, pp. 729–732, 2006.

[22] G. von Laszewski and D. Kodeboyina, "A repository service for grid workflow components," in *ICAS/ICNS*. IEEE Computer Society, 2005, p. 84.

[23] "Karajan." [Online]. Available: http://wiki.cogkit.org/index.php/Karajan