End to End Workflow Monitoring Panorama 360

George Papadimitriou
georgpap@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

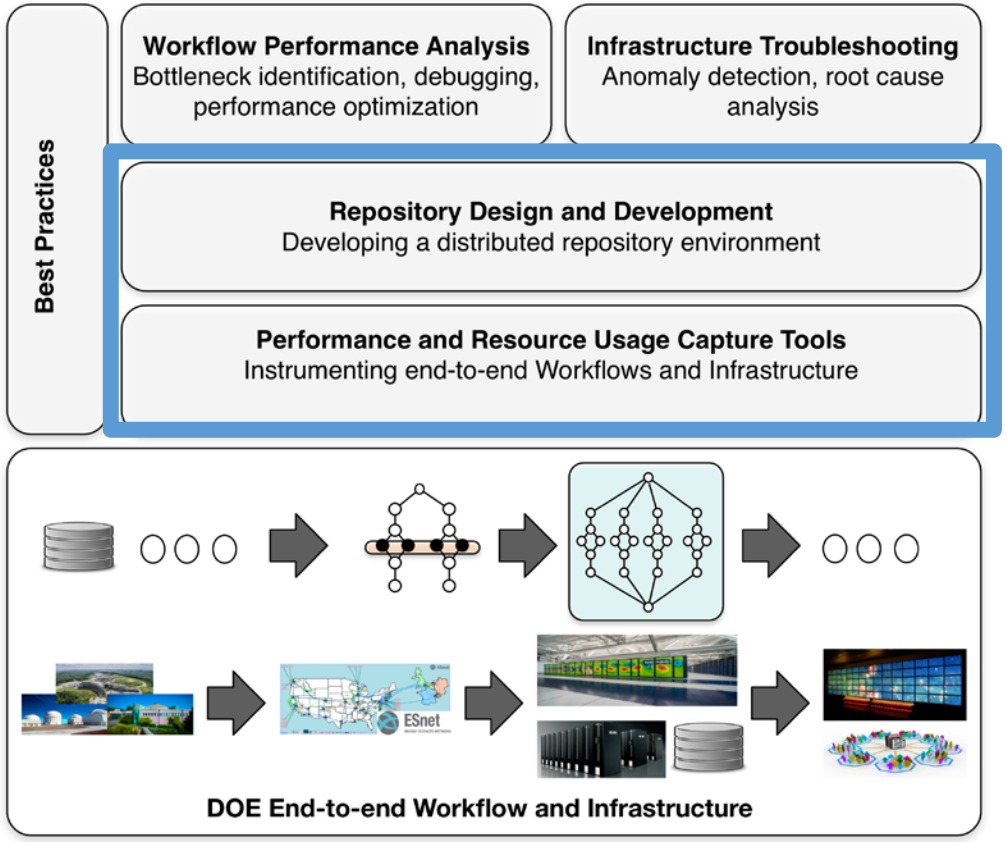
http://pegasus.isi.edu

Panorama 360: Project Overview

- Leverage the Pegasus WMS to structure, execute and monitor workflow execution
- Characterize performance: instrument data capture, summarize data, and publish results
- Create an open access **common repository** for storing end-to-end workflow performance and resource data captured using a variety of tools

***Open for external contributors**

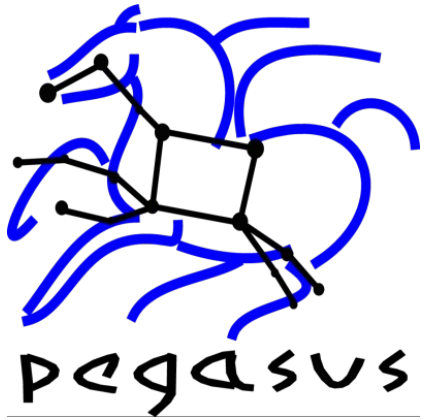
- Apply and develop ML techniques for workflow performance analysis and infrastructure troubleshooting
- Record findings, distill best practices, and share and refine them with other program teams



Panorama 360 Overview.

Data Sources: Application and Infrastructure

- **Pegasus Stampede** events regarding the workflow and its status
- **Pegasus Kickstart Online** collects resource usage traces with frequency as low as 1 second in real-time
- **Darshan** collects file access statistics (eg. POSIX, MPI-IO) during the execution
- **Globus** collects transfer statistics and general information about the transfer (throughput, file transfer errors etc.)



DARSHAN
HPC I/O Characterization Tool

globus

Pegasus Kickstart Online Traces

- In the Panorama branch pegasus-kickstart supports fine-grained monitoring capabilities by invoking a helper tool called pegasus-monitor.
- pegasus-monitor can pull resource usage statistics for workflow running tasks within a predefined time interval.
- Minimum interval is limited to **1 second**
- The statistics are being read from the **/proc** entry of the running task and among them include:
  - Number of processes and threads
  - stime and utime
  - Bytes read and written
  - iowait


```
{
  "site": "condorpool",
  "@version": "1",
  "xformation": "individuals",
  "bread": 0,
  "ts": 1552869746.009779,
  "rss": 25004,
  "procs": 5,
  "pid": 29304,
  "bwrite": 74547200,
  "threads": 6,
  "exe": "/var/lib/condor/execute/dir_29120/pegasus-4.9.2panorama/bin/pegasus-kickstart",
  "iowait": 0,
  "vm": 311864,
  "brecv": 0,
  "bsend": 0,
  "event": "kickstart.inv.online",
  "wf_uuid": "0a114f53-f208-44da-9b5c-9c30edcf49c6",
  "dag_job_id": "individuals_ID0000001",
  "utime": 8.4,
  "rank": 0,
  "task_id": "ID0000001",
  "syscr": 1088,
  "hostname": "workers1-1",
  "rchar": 1376495,
  "wchar": 7964288,
  "condor_job_id": "14988.0",
  "syscw": 47652,
  "stime": 1.57,
  "type": "kickstart",
  "wf_label": "1000genome-20190317T235728Z",
  "@timestamp": "2019-03-18T00:42:27.355Z"
}
```

Darshan

- Darshan is an HPC lightweight application-level I/O profiling tool that captures statistics on the behavior of HPC I/O operations.
- It captures data for each file opened by the application, including I/O operation counts, common I/O access sizes, cumulative timers, etc.
- I/O behavior is captured for POSIX IO, MPI-IO, HDF5, and Parallel netCDF data interface layers.
- It also captures a set of job-level characteristics, such as the number of application processes, the job's start and end times, and the job unique identification provided by the scheduler.

- *In Panorama we only expose accumulated performance data from the STDIO and POSIX modules*

- *Reference:* https://www.mcs.anl.gov/research/projects/darshan/


```
{
  "@version": "1",
  "darshan_log_version": "3.10",
  "STDIO_module_data": {
    "agg_perf_by_slowest": 0.94878,
    "shared_files": {
      "time_by_cumul_io_only": 0.000728,
      "time_by_open": 0,
      "time_by_slowest": 0.004884,
      "time_by_open_lastio": 62.774539,
      "time_by_cumul_meta_only": 0
    },
    "total_bytes": 848131,
    "agg_perf_by_open": 0.954247,
    "agg_perf_by_open_lastio": 0.012713,
    "unique_files": {
      "slowest_rank_io_time": 0.847622,
      "slowest_rank_meta_only_time": 0.004305,
      "slowest_rank": 0
    },
    "agg_perf_by_cumul": 0.953428
  },
  "uid": "1003",
  "@timestamp": "2018-07-18T19:22:36.788Z",
  "type": "stampede",
  "start_time": 1531941678,
  "exe": "/shared/software/NAMD_2.12_Linux-x86_64-MPI/namd2 equilibrate_200.conf",
  "monitoring_event": "darshan.perf",
  "end_time_asci": "Wed Jul 18 19:22:22 2018",
  "xwf__id": "e8c30de5-6bb2-4b38-a000-2b719e688e38",
  "end_time": 1531941742,
  "nprocs": 8,
  "run_time": 65,
  "event": "stampede.task.monitoring",
  "sched__id": "10469.0",
  "job__id": "namd_ID0000002",
  "POSIX_module_data": {
    "agg_perf_by_slowest": 18.470956,
    "shared_files": {
      "time_by_cumul_io_only": 0,
      "time_by_open": 0,
      "time_by_slowest": 0,
      "time_by_open_lastio": 0,
      "time_by_cumul_meta_only": 0
    },
    "total_bytes": 403752,
    "agg_perf_by_open": 18.470956,
    "agg_perf_by_open_lastio": 18.470956,
    "unique_files": {
      "slowest_rank_io_time": 0.020846,
      "slowest_rank_meta_only_time": 0.019316,
      "slowest_rank": 0
    },
    "agg_perf_by_cumul": 18.470956
  },
  "compression_method": "ZLIB",
  "start_time_asci": "Wed Jul 18 19:21:18 2018",
  "job_inst__id": 7,
  "jobid": "1547",
  "metadata": {
    "h": "romio_no_indep_rw=true;cb_nodes=4",
    "lib_ver": "3.1.6"
  }
}
```

Globus

- Globus is a research data management service, built on top of gridftp.
- It can be used to transfer files for your own computations or share files with the community.
- For every transfer request Globus creates logs containing transfer statistics, such as:
  - *Request and Completion time*
  - *Source and Destination*
  - *Transfer rate*
  - *Number of failures*
- *Reference: https://www.globus.org*

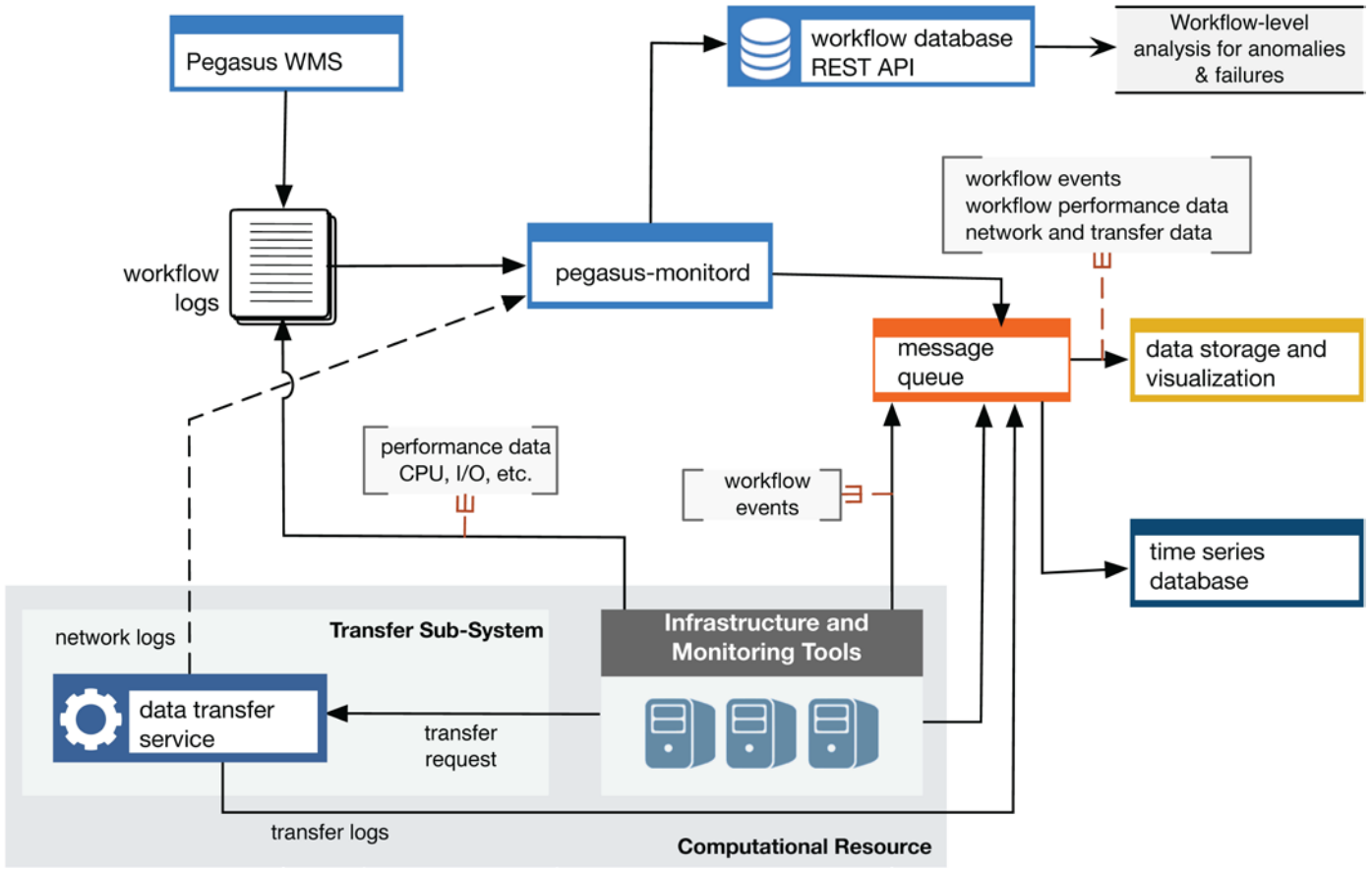

```
{
  "@version": "1",
  "label": "0a114f53-f208-44da-9b5c-9c30edcf49c6 - stage_in_local_local_2_2",
  "subtasks_pending": 0,
  "files_transferred": 2,
  "sync_level": null,
  "destination_endpoint_display_name": "georgepap#exo-master",
  "symlinks": 0,
  "nice_status_details": null,
  "completion_time": "2019-03-18 00:40:26+00:00",
  "source_endpoint_display_name": "georgepap#exo-data",
  "canceled_by_admin": null,
  "effective_bytes_per_second": 7919,
  "delete_destination_extra": false,
  "wf_uuid": "0a114f53-f208-44da-9b5c-9c30edcf49c6",
  "deadline": "2019-03-19 00:40:20+00:00",
  "fatal_error": null,
  "task_id": "68df23b8-4916-11e9-9e69-0266b1fe9f9e",
  "nice_status_expires_in": null,
  "nice_status": null,
  "recursive_symlinks": "ignore",
  "username": "georgepap",
  "owner_id": "b821e4e5-52df-41b7-a27c-4881691d259a",
  "bytes_transferred": 32248,
  "subtasks_expired": 0,
  "status": "SUCCEEDED",
  "history_deleted": false,
  "type": "TRANSFER",
  "@timestamp": "2019-03-18T00:40:31.932Z",
  "subtasks_failed": 0,
  "subtasks_total": 4,
  "preserve_timestamp": false,
  "is_paused": false,
  "directories": 0,
  "verify_checksum": false,
  "transfer_events": [↔],
  "DATA_TYPE": "task",
  "request_time": "2019-03-18 00:40:21+00:00",
  "files_skipped": 0,
  "subtasks_retrying": 0,
  "source_endpoint": "georgepap#exo-data",
  "faults": 0,
  "event": "transfer.inv.go",
  "dag_job_id": "stage_in_local_local_2_2",
  "subtasks_succeeded": 4,
  "subtasks_canceled": 0,
  "destination_endpoint": "georgepap#exo-master",
  "files": 2,
  "command": "API 0.10",
  "encrypt_data": false,
  "canceled_by_admin_message": null,
  "source_endpoint_id": "1baf5918-f813-11e8-8cda-0a1d4c5c824a",
  "nice_status_short_description": null,
  "destination_endpoint_id": "00a7def6-f813-11e8-8cda-0a1d4c5c824a",
  "bytes_checksummed": 0
}
```

Data Sources: Problems

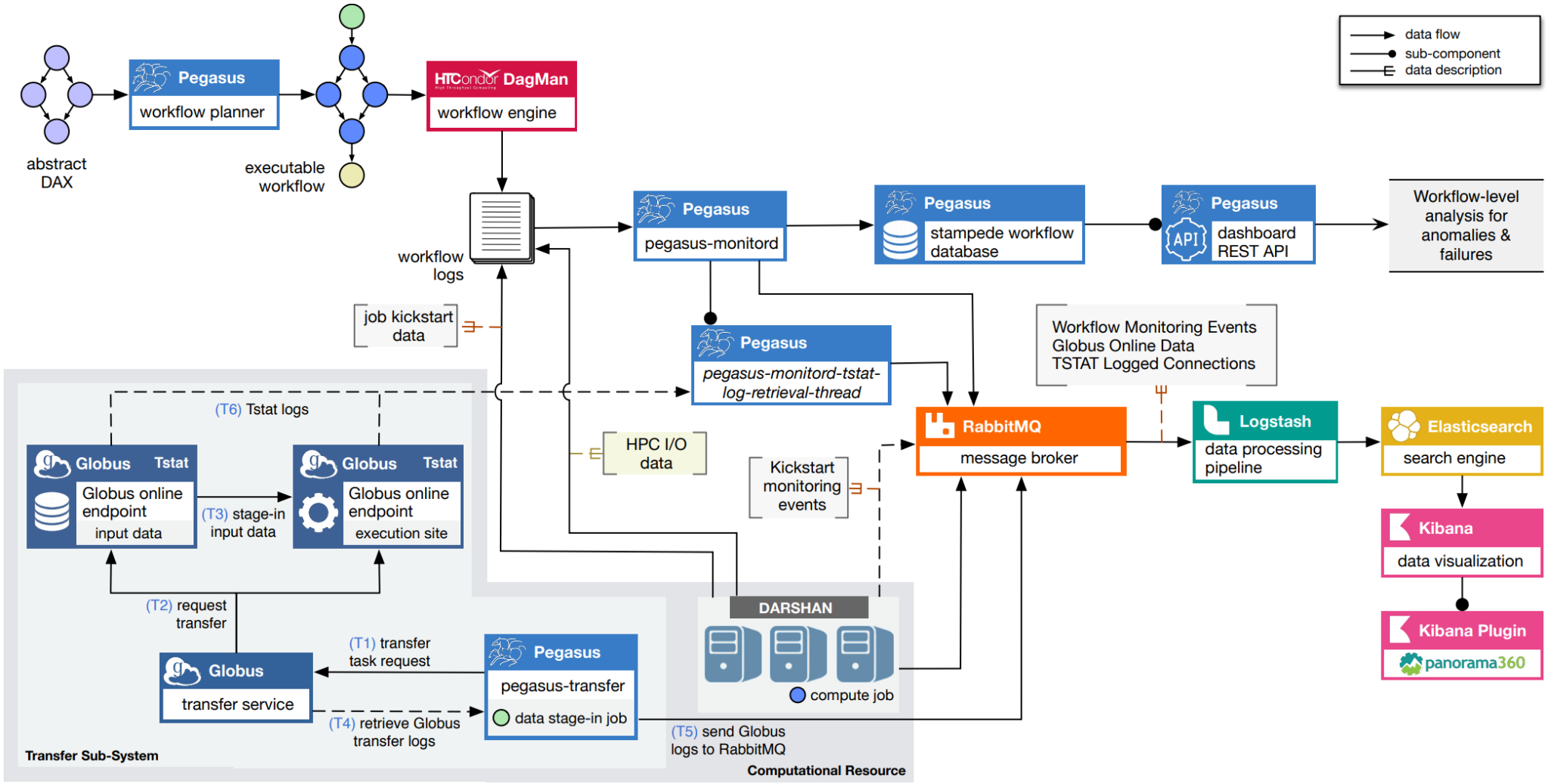
- They are scattered across multiple locations (Eg. execution site, cloud service, pegasus logs)
- They don't contain metadata about the workflow, and it's very hard to locate and match them in the future
- Captured data don't have a common format
  - Pegasus Kickstart logs are in XML format
  - Pegasus Stampede events are in JSON format
  - Pegasus Kickstart online logs are in JSON format
  - Globus logs are in JSON format
  - Darshan logs are in binary format

Data Collection: End-to-End Workflow Execution Monitoring

- **Pegasus** apart from planning and running the workflow, orchestrates the data collection
- A **message queueing system** is used, to decouple the publishers from the datastore
- Flexible **search** and **visualization engines** are used to explore the data



Data Collection: Architecture Overview



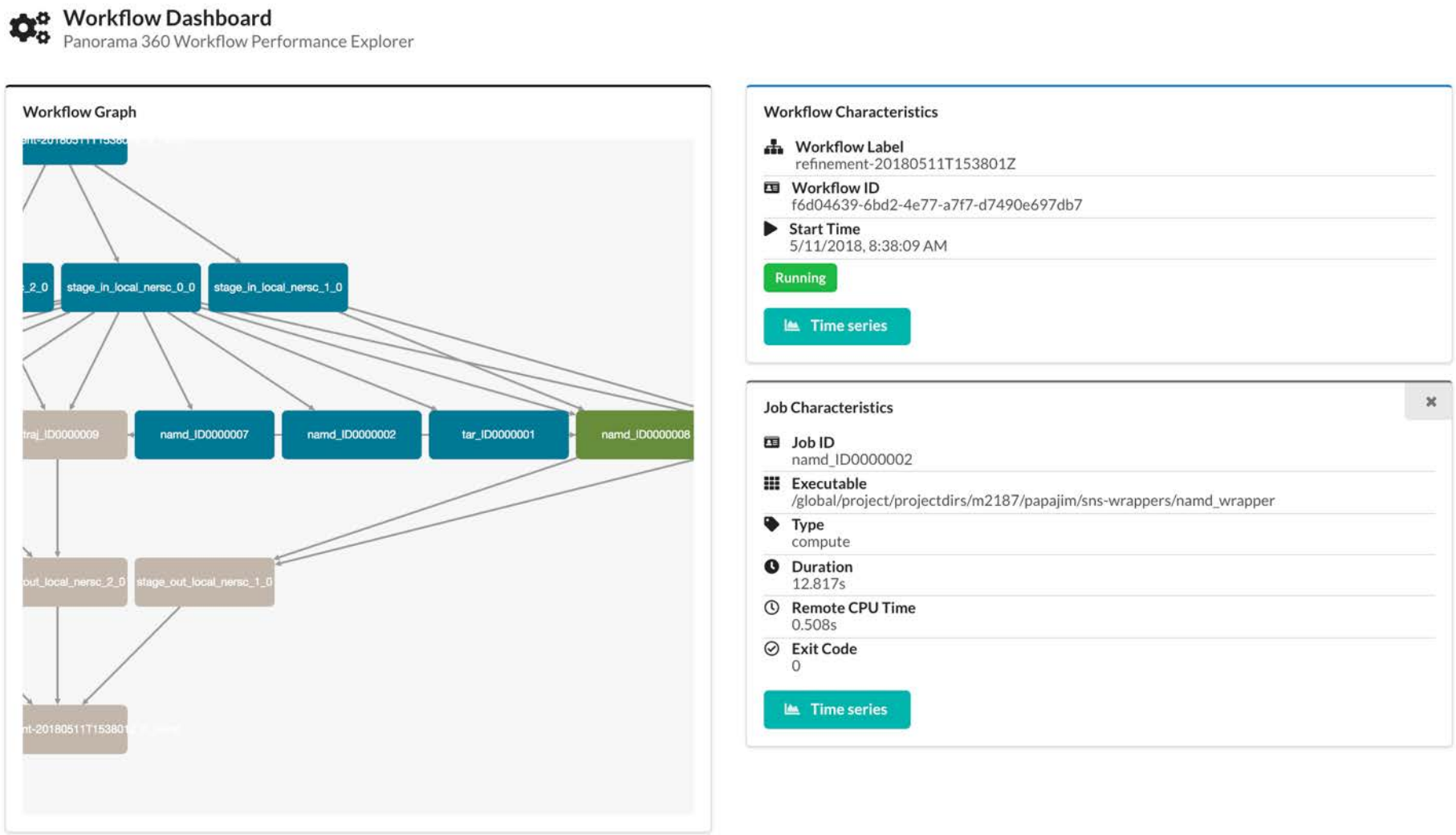
Data Collection: Tool enhancements and new tools

- pegasus-monitord: extended with JSON output format, the ability to pickup job related monitoring messages, and publish to amqp endpoints

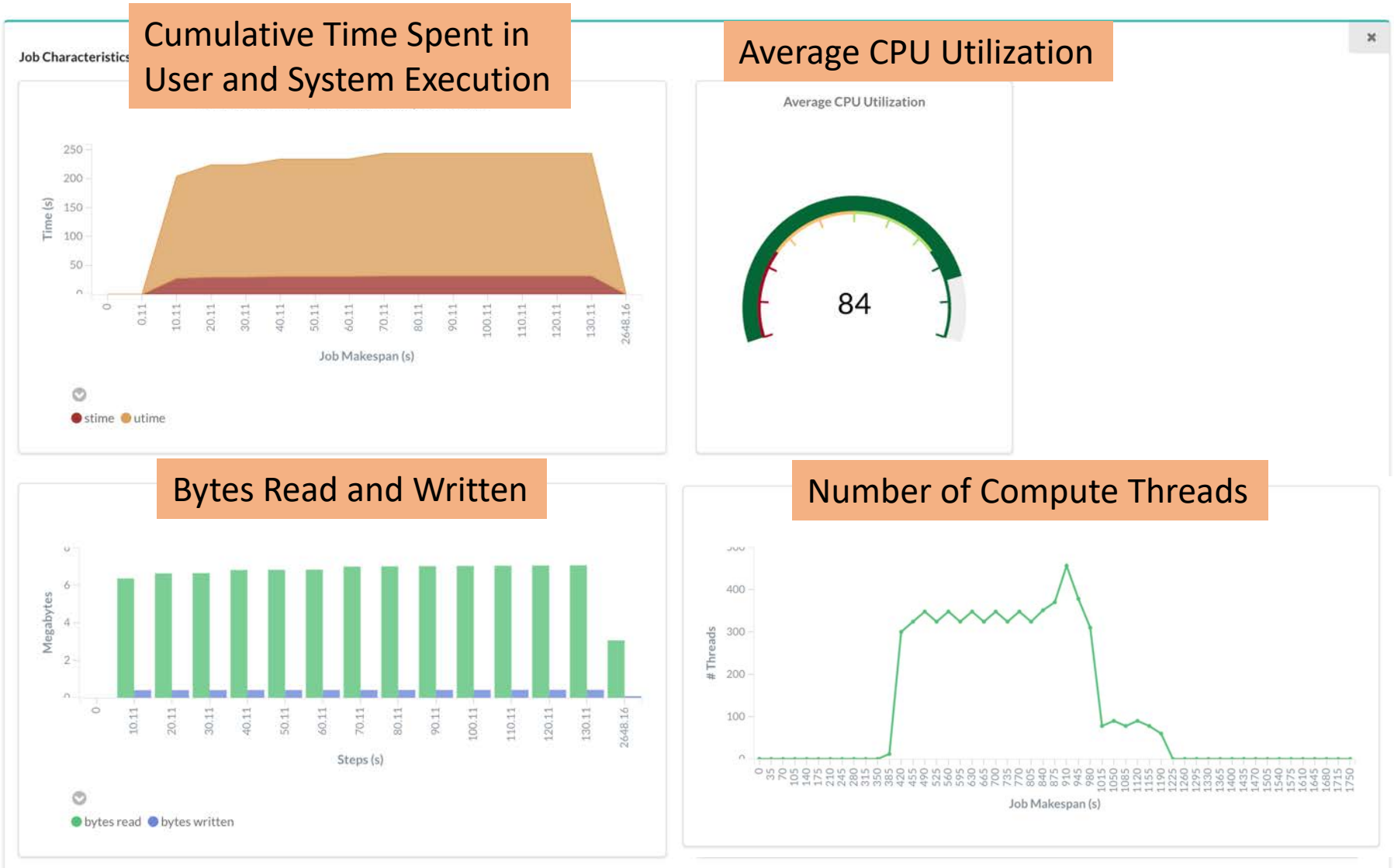
- pegasus-transfer: extended to support Globus transfers and the ability to publish statistics in json format to amqp endpoints

- pegasus-darshan: wrapper to darshan-parser, that pushes darshan logs in JSON format to pegasus-monitord

Visualization: Detailed Workflow and Job Characteristics



Visualization: Time Series Data of Workflow Performance



Repository: Organization

ElasticSearch Index	Description
panorama_transfer	Globus logs
panorama_kickstart	Pegasus-Kickstart online traces
panorama_stampede	Workflow Events and Darshan logs

Repository: Open access data

https://data.panorama.isi.edu

https://kibana.panorama.isi.edu

How to Deploy: Prerequisites

- HTCondor 8.6+:
  - https://research.cs.wisc.edu/htcondor/downloads/
- Pegasus Panorama:
  - Compile from source: https://github.com/pegasus-isi/pegasus/tree/panorama
  - Pre-compiled binaries: http://download.pegasus.isi.edu/pegasus/4.9.2panorama/
- Docker 17.02+:
  - https://docs.docker.com/install/
- Docker Compose:
  - https://docs.docker.com/compose/

How to Deploy: Monitoring Backend (RabbitMQ, ELK Stack)

- On a host that has Docker and Docker Compose installed, clone https://github.com/Panorama360/data-collection-arch
- Change to the cloned directory and execute the following command:

docker-compose up –d

- **Example:**

```
georgpap@iris:~/GitHub/panorama360/data-collection-arch$ docker-compose up -d
Creating network "panorama_net" with driver "bridge"
Creating panorama-rabbitmq      ... done
Creating panorama-elasticsearch ... done
Creating panorama-kibana        ... done
Creating panorama-logstash      ... done
georgpap@iris:~/GitHub/panorama360/data-collection-arch$
```

How to Deploy: Checking Services (RabbitMQ, ELK Stack)

Now the host should have RabbitMQ, Elasticsearch, Logstash, and Kibana running as Docker containers with their service ports exposed. Try to access them…

- RabbitMQ: http://<hostname or ip>:15672
- Elasticsearch: http://<hostname or ip>:9200
- Logstash: http://<hostname or ip>:9600
- Kibana: http://<hostname or ip>:5601

How to Deploy: Enabling Stampede Events

- In order to get pegasus-monitord to publish **all** of its events to an AMQP endpoint in JSON format, 3 properties must be specified in the workflow's properties file (eg. "pegasus.properties").
  - pegasus.monitord.encoding = json
  - pegasus.catalog.workflow.amqp.url = amqp://[username:password]@<hostname>[:port]/<exchange_name>
  - pegasus.catalog.workflow.amqp.events = stampede.*

- **Example:**

```
19  # Monitord Events
20  pegasus.monitord.encoding=json
21  pegasus.catalog.workflow.amqp.url=amqp://panorama:panorama@amqp.isi.edu:5672/panorama/monitoring
22  pegasus.catalog.workflow.amqp.events = stampede.*
```

- More about stampede events: https://pegasus.isi.edu/documentation/stampede_wf_events.php

How to Deploy: Enabling Transfer Events

- In order to get pegasus-transfer to publish transfer statistics from the Globus Transfer Service to an AMQP endpoint in JSON format, 2 profiles must be specified in the workflow's sites catalog (eg. "sites.xml"), under the site where pegasus-transfer is going to be invoked (eg. "local").
  - env.PEGASUS_TRANSFER_PUBLISH = 1
  - env.PEGASUS_AMQP_URL = amqp://[username:password]@<hostname>[:port]/<exchange_name>
- **Example:**

```
<!-- The local site contains information about the submit host -->
<site handle="local">
    <directory type="shared-scratch" path="${PWD}/work/scratch">
        <file-server operation="all" url="go://georgepap#exo-master/${PWD}/work/scratch"/>
    </directory>
    <directory type="local-storage" path="${PWD}/work/outputs">
        <file-server operation="all" url="go://georgepap#exo-master/${PWD}/work/outputs"/>
    </directory>
    <!-- These profiles tell pegasus-transfer to publish stats to an AMQP endpoint -->
    <profile namespace="env" key="PEGASUS_TRANSFER_PUBLISH">1</profile>
    <profile namespace="env" key="PEGASUS_AMQP_URL">amqp://panorama:panorama@amqp.isi.edu:5672/panorama/monitoring</profile>
</site>
```

How to Deploy: Enabling Kickstart Online Traces

- In order to get pegasus-kickstart to publish traces of resource usage statistics to an AMQP endpoint in JSON format, 2 profiles must be specified in the workflow's sites catalog (eg. "sites.xml") under the compute site.
  - pegasus.gridstart.arguments = -m <interval in seconds>
  - env.KICKSTART_MON_URL = rabbitmq://[USERNAME:PASSWORD]@<hostname>[:port]/api/exchanges/<exchange_name>/publish
- **Example:**

```
23  <site handle="condorpool" arch="x86_64" os="LINUX">
24    <!-- These profiles tell Pegasus that the site is a plain Condor pool -->
25    <profile namespace="pegasus" key="style">condor</profile>
26    <profile namespace="condor" key="universe">vanilla</profile>
27    <!-- These profiles tell pegasus-kickstart to publish stats to an AMQP endpoint -->
28    <profile namespace="pegasus" key="gridstart.arguments">-m 10</profile>
29    <profile namespace="env" key="KICKSTART_MON_URL">rabbitmq://panorama:panorama@amqp.isi.edu:15672/api/exchanges/panorama/monitoring/publish</profile>
30  </site>
```

- Alternatively if we want to customize the monitoring interval per computational task we can specify the profile in the workflow's transformation catalog (eg. "tx.txt")

How to Deploy: Enabling Kickstart Online Traces (MPI Jobs)

- Usually MPI jobs are not launched by Pegasus-Kickstart. Thus, adding the gridstart.arguments profile doesn't have any effect.
- We can work around this by using a wrapper script for the MPI job, that invokes directly pegasus-monitor.
- We still need to specify **KICKSTART_MON_URL** in the sites catalog.

- **Example:**

```
1  #!/usr/bin/env bash
2
3  cd $PEGASUS_SCRATCH_DIR
4
5  mpirun pegasus-monitor -i 10 /shared/software/NAMD_2.12_Linux-x86_64-MPI/namd2 $@
6
```

How to Deploy: Enabling Darshan Logs (MPI Jobs)

- In case your MPI application wasn't compiled and statically linked with Darshan's library, we need to set a profile in the transformation catalog, adding the path of the library to LD_PRELOAD.

- We launch the application using a **wrapper script**, and as post job steps:
  - Build the darshan log path from the environmental variables
  - Invoke **pegasus-darshan** with the files as input

```
tr namd {
    site local-slurm {
        pfn "/shared/software/wrappers/namd_wrapper_slurm.sh"
        arch "x86_64"
        os "LINUX"
        type "INSTALLED"
        profile pegasus "exitcode.successmsg" "End of program"
        profile pegasus "memory" "2500M"
        profile globus "jobtype" "single"
        profile env "LD_PRELOAD" "/shared/software/darshan-3.1.6/lib/libdarshan.so"
    }
}
```

Transformation Catalog

```
#!/usr/bin/env bash

cd $PEGASUS_SCRATCH_DIR

mpirun pegasus-monitor -i 10 /shared/software/NAMD_2.12_Linux-x86_64-MPI/namd2 $@

#post job parse darshan output
DAY=$(date '+%d')
DAY=${DAY##0}
MONTH=$(date '+%m')
MONTH=${MONTH##0}
YEAR=$(date '+%Y')

darshan_base=/shared/darshan-logs/${YEAR}/${MONTH}/${DAY}
darshan_file=${darshan_base}/${SLURM_JOB_USER}_namd2_id${SLURM_JOB_ID}_*.darshan

for f in $darshan_file; do
   $PEGASUS_HOME/bin/pegasus-darshan -f "$f"
done
```

Wrapper Shell Script

How to Deploy: Enabling Darshan Logs (MPI Jobs)

- pegasus-darshan will output in stdout a monitoring payload, that will be picked by pegasus-monitord, which in its turn will publish it to the AMQP endpoint.

- *This can also be used as a generic way of adding new tools to this architecture*.


```
@@@MONITORING_PAYLOAD - START@@@
{
  &quot;monitoring_event&quot;: &quot;darshan.perf&quot;,
  &quot;payload&quot;: [
    {
      &quot;POSIX_module_data&quot;: {
        &quot;agg_perf_by_cumul&quot;: 14.667417,
        &quot;agg_perf_by_open&quot;: 14.667417,
        &quot;agg_perf_by_open_lastio&quot;: 14.667417,
        &quot;agg_perf_by_slowest&quot;: 14.667417,
        &quot;shared_files&quot;: {
          &quot;time_by_cumul_io_only&quot;: 0.0,
          &quot;time_by_cumul_meta_only&quot;: 0.0,
          &quot;time_by_open&quot;: 0.0,
          &quot;time_by_open_lastio&quot;: 0.0,
          &quot;time_by_slowest&quot;: 0.0
        },
        &quot;total_bytes&quot;: 403761,
        &quot;unique_files&quot;: {
          &quot;slowest_rank&quot;: 0.0,
          &quot;slowest_rank_io_time&quot;: 0.026253,
          &quot;slowest_rank_meta_only_time&quot;: 0.023997
        }
      }
    }
  ],
  &quot;ts&quot;: 1552878285
}
@@@MONITORING_PAYLOAD - END@@@
```

DEMO

panorama360

- GitHub: https://github.com/Panorama360
- Website: https://panorama360.github.io

USC Viterbi
School of Engineering
Department of Computer Science

George Papadimitriou
Computer Science PhD Student
University of Southern California

email: georgpap@isi.edu

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website
http://pegasus.isi.edu

Users Mailing List
pegasus-users@isi.edu

Support
pegasus-support@isi.edu

Pegasus Online Office Hours
https://pegasus.isi.edu/blog/online-pegasus-office-hours/

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments