HIGH PERFORMANCE COMPUTING PAPER REVIEW

12M56106

Naoki Yatsu

Review Paper

2013 IEEE International Congress on Big Data

Milieu: Lightweight and Configurable Big Data Provenance for Science

You-Wei Cheah, Beth Plale
School of Informatics and Computing Indiana University, Bloomington, IN

Richard Canon, Lavanya Ramakrishnan
Lawrence Berkeley National Laboratory, Berkeley, CA

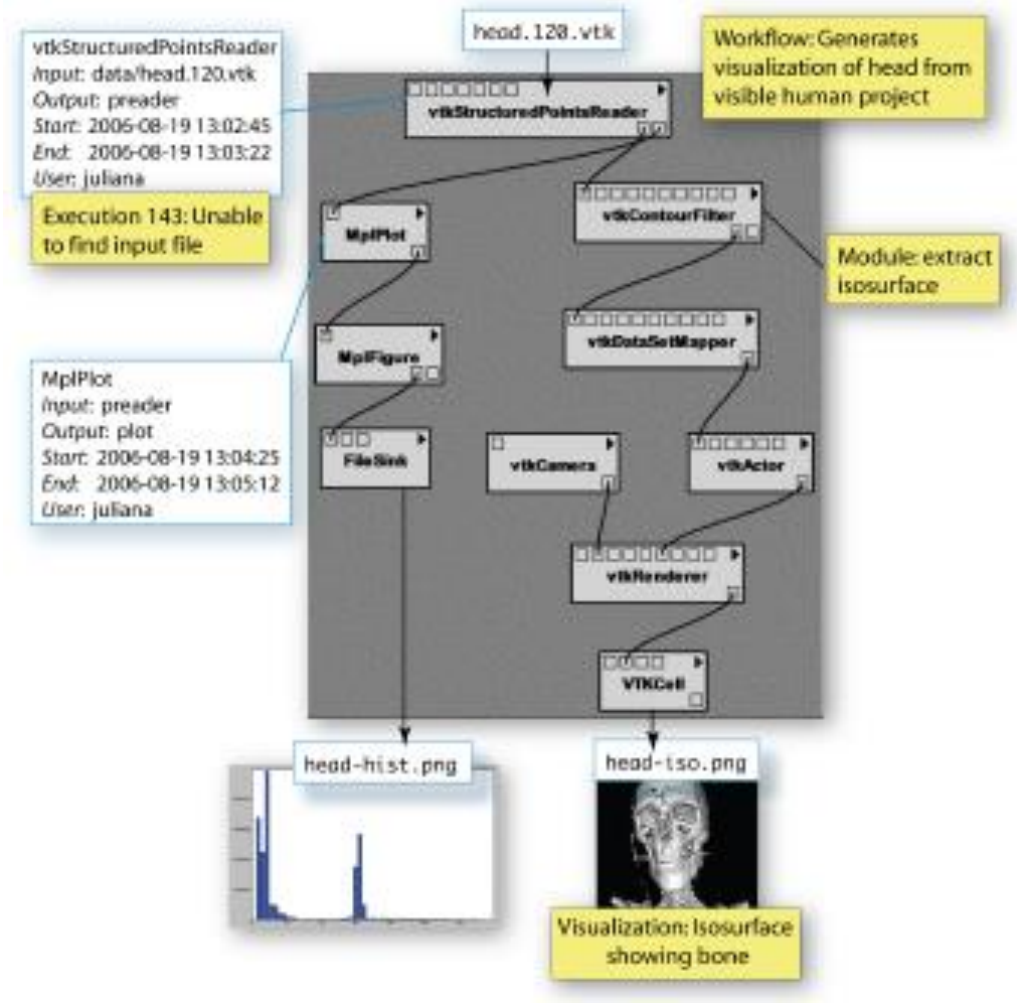
Contents

Contents

INTRODUCTION

- The volume and complexity of data produced and analyzed in scientific collaborations is growing exponentially. It is important to track scientific data-intensive analysis workflows to provide context and reproducibility as data is transformed in these collaborations.
- Provenance has traditionally been collected at the workflow level often making it hard to capture relevant information about resource characteristics and is difficult for users to easily incorporate in existing workflows.
- Milieu, a framework focused on the collection of provenance for scientific experiments in High Performance Computing systems.

Scientific Workflow



One of Open Problem -
Information management infrastructure.

With the growing volume of raw data, workflows and provenance information, there is a need for efficient and effective techniques to manage
these data.

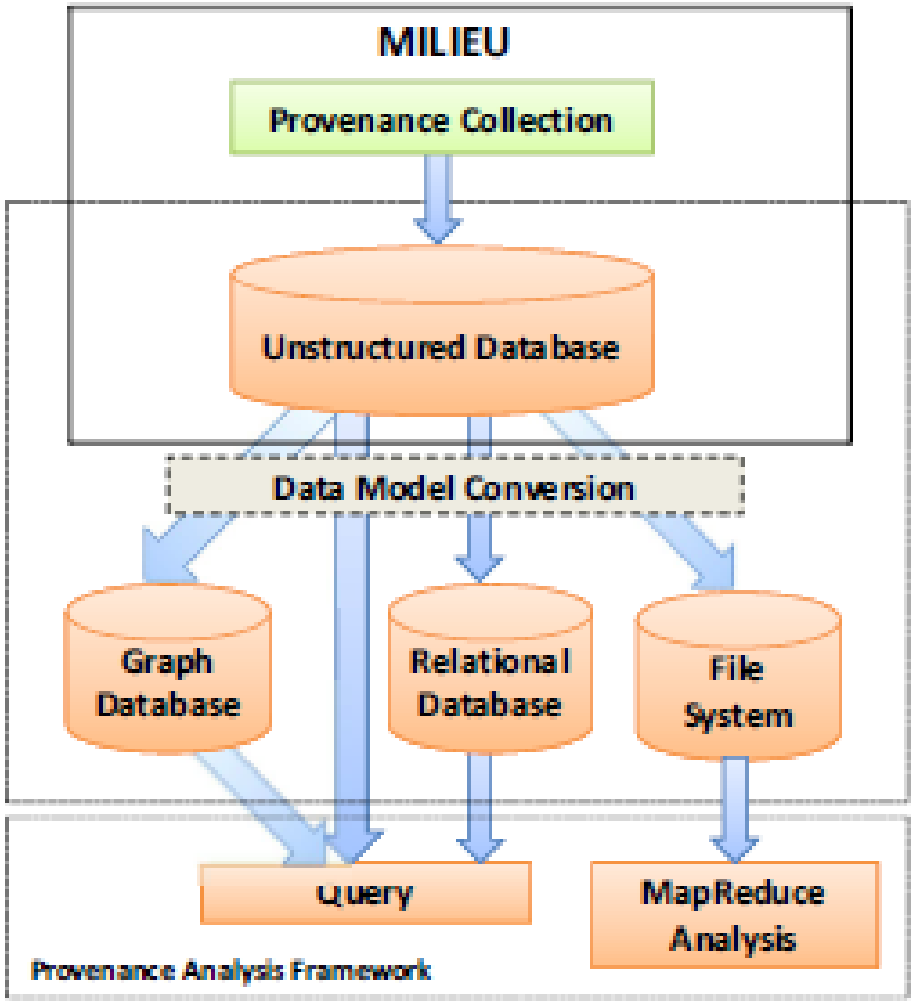
S. B. Davidson and J. Freire. Provenance and Scientific Workflows: Challenges and Opportunities. (2008)

Contributions of this Paper

- We present an architecture and implementation of Milieu: a provenance collection and storage framework for scientific applications running on HPC systems.
- We evaluate Milieu on two large systems at the National Energy Research Scientific Computing Center (NERSC) including a petascale system and show that the provenance overhead is minimal and within the normal variation experienced by the applications on the systems.
- We present and evaluate our query interface for provenance collection and provide a framework for provenance analysis support.

Contents

Tiered Provenance System



Milieu is the first tier where provenance data is collected in unstructured or semi-structured format.

Fig. 1: Overview of Provenance Collection and Analysis Framework. Milieu is the first tier where provenance data is collected in unstructured or semi-structured format. The second tier is more focused on provenance analysis and appropriate storage options might be used. This architecture allows us to provide the best optimized storage model for collection and specific analysis types.

Design Goals

- Support for various provenance data models
- Low overhead
- Semi-transparent collection
- Support user annotations
- Staged provenance levels
- Scalability

Contents

Architecture of Provenance Framework

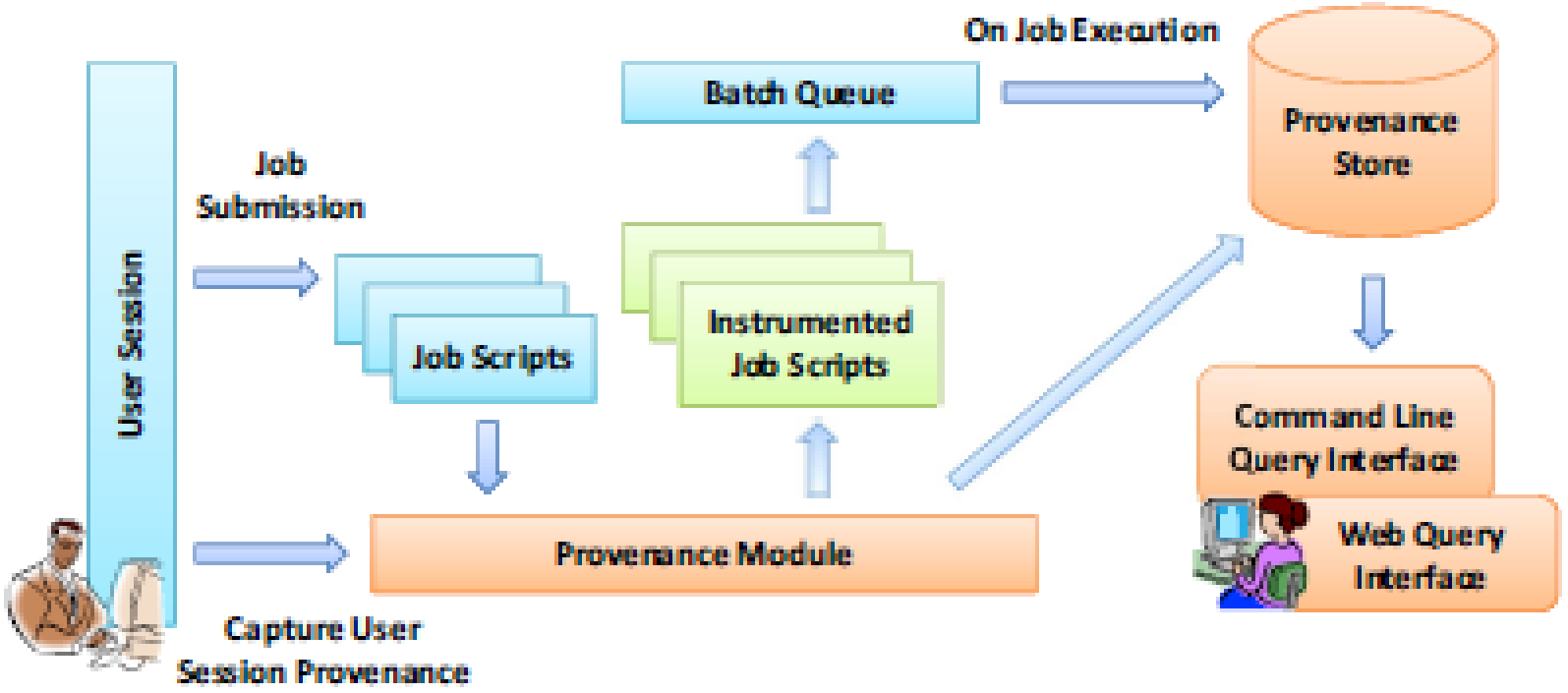


Fig. 2: Architecture of provenance framework. The activation of the collection mechanism triggers user session provenance and provenance from job script executions to be captured, which is stored in a MongoDB data store. The stored provenance can be retrieved either through a command line interface or via a web query interface.

A. Collection

- The provenance capture of a user session is collected in its entirety by our framework. For the provenance collection, we devised three levels of provenance capture:
- *Level 1* - Basic Provenance.
  - Includes information about the script, the outputs of the job run, basic environment about where the job was submitted and user annotations.
  - Essentially provenance that a scientific user would be interested in
- *Level 2* - Level 1 and more detailed resource information for the computation.
  - More of interests to system administrators and/or for detailed analysis of resource usage for a particular class of problems.
- *Level 3* - Level 2 with the addition of provenance in the form of detailed traces of I/O calls for commands in the job script
  - To be used rarely to collect detailed I/O information for data sets and/or debugging.

B. Storage

- The captured provenance is diverse and varied. Thus, the data store must be capable of supporting the storage of semistructured provenance documents. In our current implementation, we use MongoDB, a NoSQL data store for our storage needs.
- MongoDB is a scalable, high-performance data store that is open-source and developed in C++.
- In our system, captured provenance is grouped around job IDs (unique identifiers assigned by the batch queue system) or file identifiers (location based).

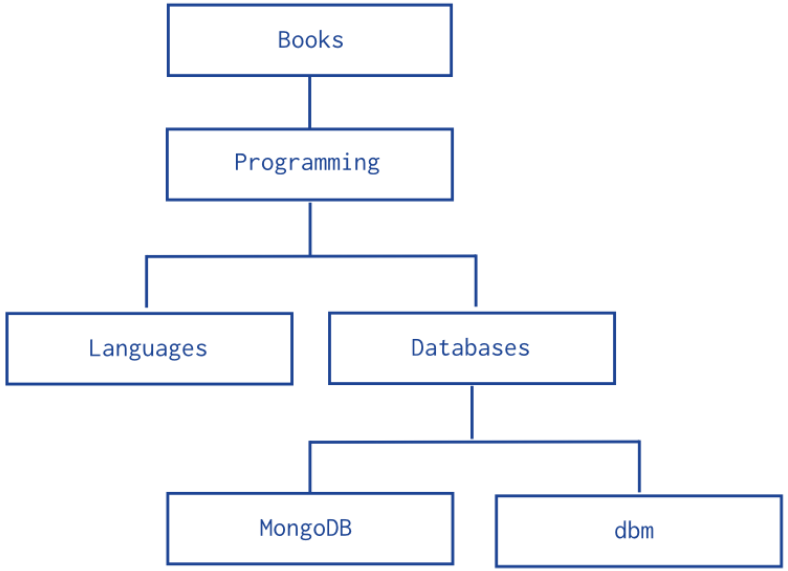
What is MongoDB

MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++, MongoDB features:

- Document-Oriented Storage
- Replication & High Availability
- Querying
- Map/Reduce
- GridFS

…

Figure 3: Data model with embedded fields that contain all related information.



http://www.mongodb.org/ - MongoDB-data-models-guide

Contents

A. Collection

- The Milieu collection module is available to the user by loading the module for provenance.
- In order to specify the calls that the user wants to strace, the tag *record trace provenance* has to be added in front of the intended call. Additional user provenance can also be captured in the form of name value pairs by adding a line *record provenance name value* in the users job scripts, as shown in the listing.
- When a user is done with their session, the captured provenance is stored in a file, which is then put into storage on our provenance Mongo database (MongoDB).
- We implemented our collection tools using a mixture of Python and Shell scripts. In all cases, the Shell scripts were used to interact at the front end, since this is native in a HPC environment.

```
#!/bin/csh
#PBS -j oe
#PBS -q regular
#PBS -N GTCcloud
#PBS -l walltime=00:20:00
#PBS -l nodes=8:ppn=8
#PBS -S /bin/csh
module load python
python $USER/provenance/source/python_scripts/
    timestamp.py $USER/provenance/source/
    python_scripts/dbconfig.py run.pbs 'begin
    script' yocheah $PBS_JOBID
qstat -f $PBS_JOBID > $USER/GSCRATCH/GTC/qstat
    .$PBS_JOBID

set verbose
cd $PBS_O_WORKDIR
cp ./gtc.input.64MPItasks.cloud gtc.input
record_provenance ``Edit date'' ``2012-01-01''
strace -tt -o $USER/GSCRATCH/GTC/strace.out
    .1.31543.20120927185010 mpirun -n 64 ./
    gtcmpi
rm gtc.input
cp $PBS_JOBID.OU 'pwd'/output
    .31543.20120927185010
echo '$USER/provenance/source/bin/
    qsub_file_insert hopper12 yocheah
    $PBS_JOBID 3 $USER/provenance/source/
    python_scripts/file_insert.py $USER/
    provenance/source/python_scripts/dbconfig.
    py $USER/GSCRATCH/GTC/qstat.$PBS_JOBID
    $USER/GSCRATCH/GTC/strace.out
    .0.31543.20120927185010 $USER/GSCRATCH/GTC
    /strace.out.1.31543.20120927185010 $USER/
    GSCRATCH/GTC/strace.out
    .2.31543.20120927185010 $USER/GSCRATCH/GTC
    /output.31543.20120927185010'
python $USER/provenance/source/python_scripts/
    timestamp.py $USER/provenance/source/
    python_scripts/dbconfig.py run.pbs 'end
    script' yocheah $PBS_JOBID
```

Listing 1: Example of an instrumented PBS script. Highlighted texts are instrumentation added or modified by the provenance framework or user.

```
$ cmdline_pc
# begin user session
...
# end user session
$ exit
```

B. Storage

- We group all provenance documents under a single provenance collection. For a single job ID, multiple provenance documents may exist in MongoDB.
- Mandatory fields (“columns” in the traditional sense) for each document include an associated job ID, user ID and timestamp.
- The provenance store also has information about the systems. The system information is more structured and will contain more fields, such as the host name, IP address, user environment variables, etc.
- For file management, we use the MongoDB GridFS specification for storing the actual file contents due to its ease of use, metadata and large file support.

C. Query and Access

- Milieu provides two query interfaces a) command-line and, b) web interface, that allows the user to query and make simple edits to the MongoDB provenance storage.

```
[Provenance Query$] jobid . (field,walltime) (
    value,00:10:00)
```

Listing 3: Example of a query to return all job IDs that have walltime 00:10:00 in the command line.

```
[Provenance Query$] file MILC
```

Listing 4: Example of a query to return all filenames that contain MILC.

Contents

A. Testbed Setup

- 3 applications from the NERSC-6 application benchmark suite.
- A range of problem sizes and core counts were used to capture mid-range to largescale codes.
- Our experiments were evaluated on two NERSC systems, namely Carver and Hopper, a petascale system.

Carver: IBM iDataPlex system with 1202 compute nodes.
Node - 2 two-core Westmere 2.67GHz processors with 48GB of memory
4 eight-core Nehalem-EX 2.00GHz processors with 1TB of memory
All nodes are connected using 4X QDR InfiniBand technology.

Hopper: Cray XE6 peta-flop system consisting of 6384 nodes.
Node - 2 twelve-core AMD ‘Magny-Cours 2.1GHz processors with 64GB of memory.
The compute nodes are connected via a custom high-bandwidth,
low latency network provided by Cray in a 3D torus topology.

NERSC-6 application benchmark

Benchmark	Science Area	Algorithm Space	Base Case Concurrency	Problem Description	Lang	Libraries
CAM	Climate (BER)	Navier Stokes CFD	56, 240 Strong scaling	D Grid, (~0.5 deg resolution); 240 timesteps	F90	netCDF
GAMESS	Quantum Chem (BES)	Dense linear algebra, DFT	256, 1024 (Same as TI-09)	rms gradient, MP2 gradient	F77	DDI, BLAS
GTC	Fusion (FES)	PIC, finite difference	512, 2048 Weak scaling	100 particles per cell	F90	
IMPACT-T	Accelerator Physics (HEP)	Largely PIC, FFT component	256,1024 Strong scaling	50 particles per cell (currently)	F90	
MAESTRO	Astrophysics (HEP)	Low Mach Hydro; block structured-grid multiphysics	512, 2048 Weak scaling	64 x 64 x 128 gridpts per proc; 10 timesteps	F90	Boxlib
MILC	Lattice Gauge Physics (NP)	Conjugate gradient, sparse matrix; FFT	256, 1024, 8192 Weak scaling	8 x 8 x 8 x 9 Local Grid, ~70,000 iters	C, assem.	
PARATEC	Material Science (BES)	DFT; FFT, BLAS3	256, 1024 Strong scaling	686 Atoms, 1372 bands, 20 iters	F90	Scalapack

Table 6: Final Benchmark Selection Matrix together with problem sizes and concurrencies.

Workload

- GTC short for 3D Gyrokinetic Toroidal Code, is a 3-dimensional particle-in-cell (PIC) code used to study microturbulence in magnetically confined toroidal fusion plasmas. We used version 2 of the GTC code with a large problem size that involves 66,455,552 number of grid points as input on Hopper and 2 million grid points on Carver.
- MILC or MIMD Lattice Computation is used in part to study quantum chromodynamics (QCD), the theory of the subatomic “strong” interactions responsible for binding quarks into protons and neutrons and holding them together in the nucleus. We used version 7 of the MILC code in our experiments using the extra large input lattice size of 64x64x64x144 on Hopper. A smaller version using a lattice size of 32x32x16x18 with 2 quark flavors, four trajectories and eight steps per trajectory was used on Carver for our evaluation.
- The PARAllel Total Energy Code (PARATEC) benchmark code performs *ab-initio* quantum-mechanical total energy calculations using pseudopotentials, a plane wave basis set and uses an all-band (unconstrained) conjugate gradient (CG) approach for solving Density Functional Theory’s (DFT) Kohn- Sham equations. The version we used is based off the NERSC- 6 input that contains 6 conjugate gradient iterations and only 250 atoms in a diamond lattice configuration. This input does not allow any aggregate over the transposed data.

Conditions

- The applications were executed on Carver using 8 nodes with 8 processes per node. On Hopper, GTC was executed using 2048 cores and MILC was executed using 4096 cores.
- We conducted our experiments by comparing the different levels of provenance capture along with the base case of just running the application without any provenance capture.
- For each application, we perform 15 measurements without any provenance capture and also 15 measurements for each level of provenance capture (levels 1–3). We only performed 7 Hopper measurements for each scenario since jobs are more expensive than their Carver counterparts both resource-wise and timewise.
- All results were generated from job runs during normal production time on the NERSC machines discussed above.

B. Evaluation of Provenance Collection

- Evaluation of three applications on two of NERSC HPC systems: Hopper and Carver

Carver

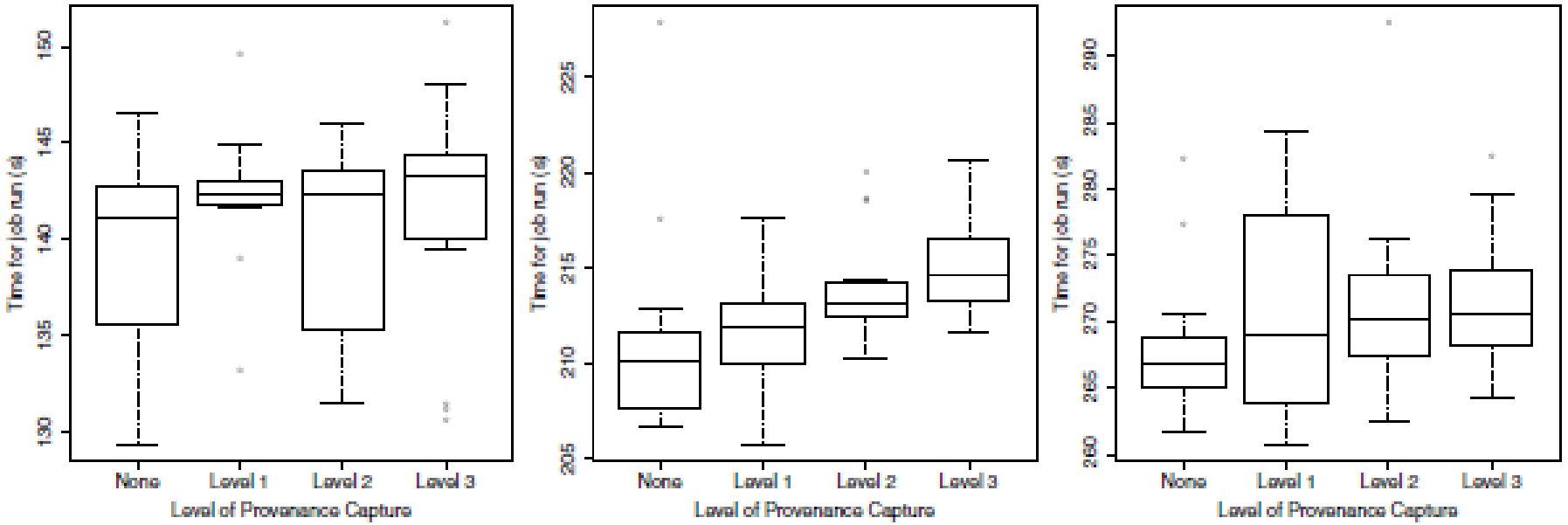
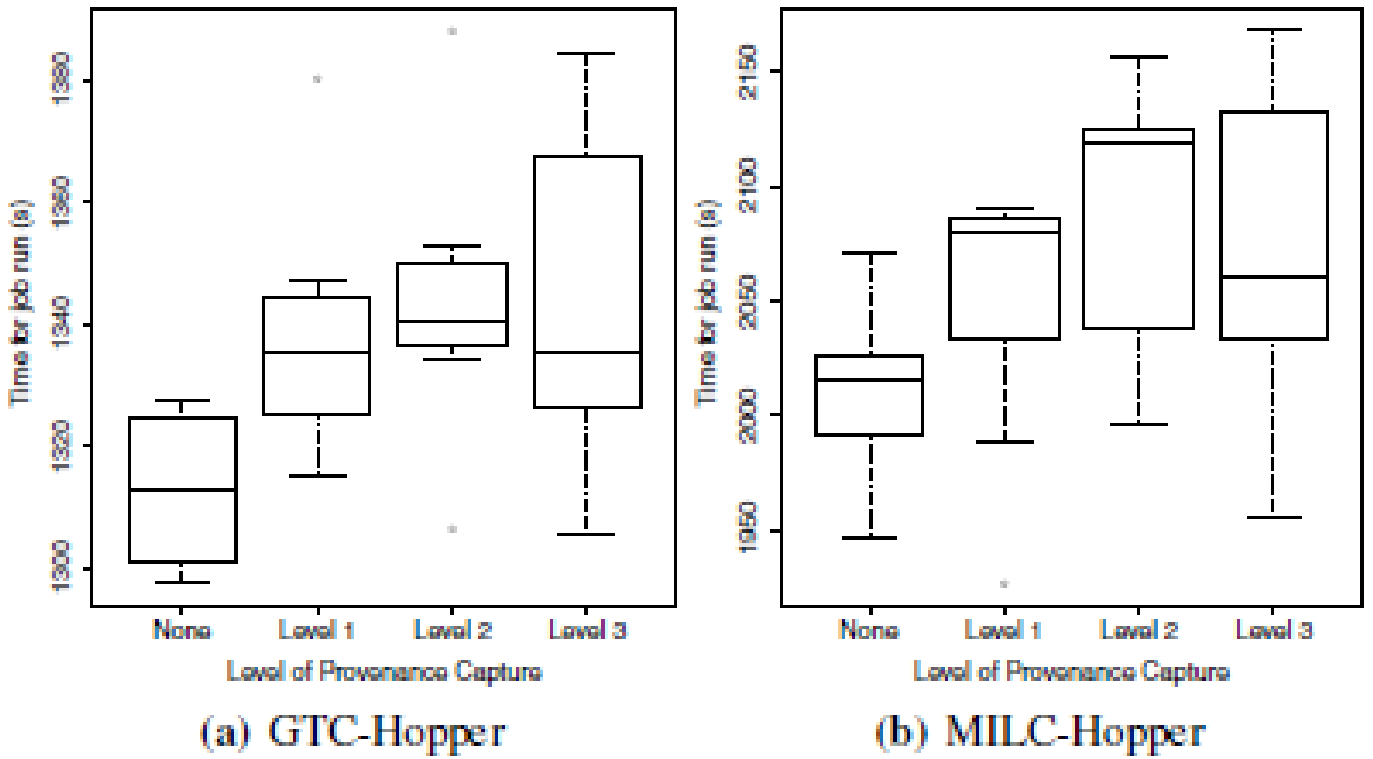


Fig. 3: Timings for base run and 3 different levels of provenance capture on Carver. Provenance capture does not yield significant overhead.

- The overhead associated with GTC, MILC and PARATEC to be insignificant compared to the base case with no provenance capture.
- Overhead is within the normal variability (well within 2%) that these application already experience on these systems.

Hopper



(a) GTC-Hopper (b) MILC-Hopper

Fig. 4: Timings for base run and 3 different levels of provenance capture on Hopper. Provenance capture shows a slight overhead (2–3%) compared with the base case. Median timings for level 3 are lower than level 1 and 2 but have more spread.

- For GTC (Figure 4a), we note that the provenance capture accounts for about 22–28 seconds (overhead of slightly over 2%). Figure 4b shows the performance of MILC on Hopper with different provenance levels. The same trend occurs for MILC on Hopper with an overhead of about 46–104 seconds (2–5% overhead).

Analysis of Overhead

TABLE I: Duration taken for individual scripts for provenance capture of a GTC job on Hopper (level 2) and Carver (level 3).

Script Name	Duration (s)	
	Hopper	Carver
qsub-prov	6.05	3.00
– qsub_pc.py[1]	(0.08)	(0.07)
timestamp.py (begin)	0.25	0.05
qsub_file_insert	7.77	0.30
– file_insert.py[2]	(0.43)	(0.03)
timestamp.py (end)	0.16	0.11

[1,2] Duration of qsub_prov.py and file_insert.py accounted within qsub-prov and qsub_file_insert respectively.

- From the results of both tables, we observe that our provenance module contributes very little to the overhead of the end-to-end execution of these applications.
- The Hopper overhead is a little higher than Carver since Carver is more directly connected to the file system server than Hopper.

C. Evaluation of Provenance Query

- Evaluate the performance of querying the stored provenance through a few queries.
  - Regular Expression
  - Limited set queries
  - Exact indexed queries

Regular Expression

- Query for a regular expression job ID query by iterating through all available documents in MongoDB and returning all documents
- Query performance generally increases linearly as the number of documents that are being stored in the database increases.
- If there are queries that need to query all data in a database, they will benefit from a MapReduce framework that will allow scalability.

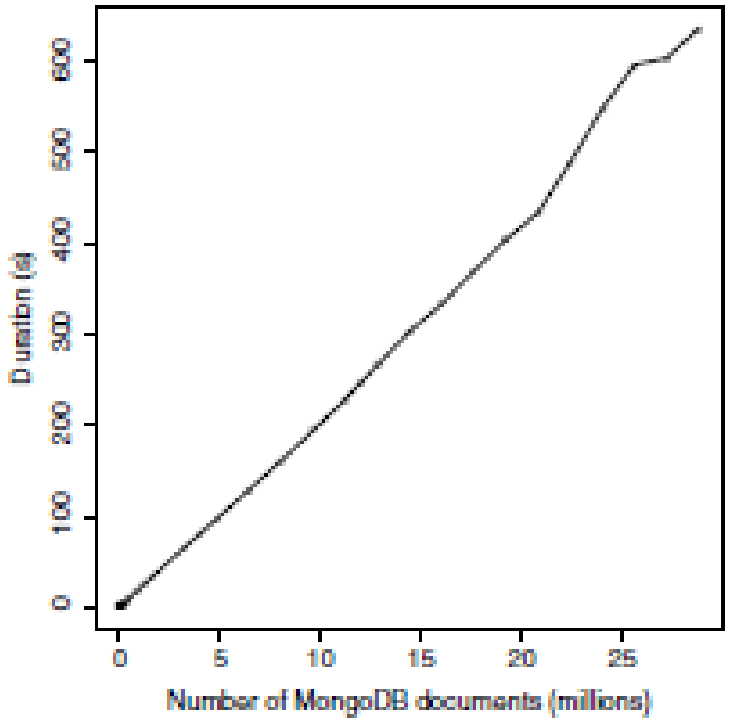


Fig. 5: Plot of query duration vs number of MongoDB documents returned. The return time for each query is generally linear with the number of results returned.

Limited set queries

- Regular expression query of a single job ID for a MongoDB database of different sizes.
- The outliers in this figure are the first queries issued against MongoDB.
- These first queries are retrieved and cached in memory, resulting in subsequent identical queries having less significant response times, approximately half the response times of the first queries.

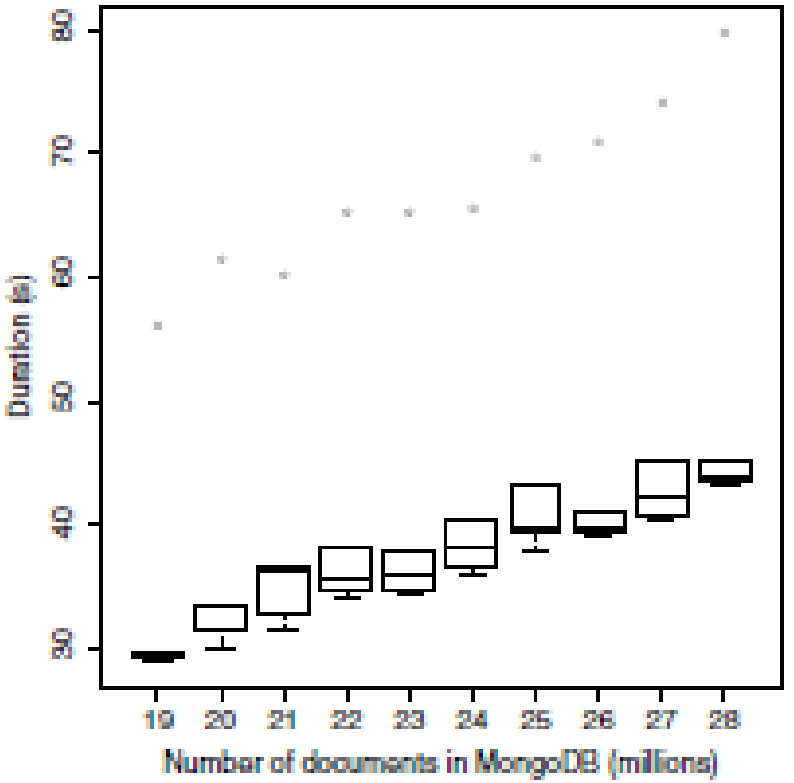


Fig. 6: Plot of query duration for a regular expression query of a single job ID vs number of MongoDB documents. Outliers reflect the timings of the first query issued. The query performance here is linear and the result size returned for each query is 16 documents.

Exact indexed queries

- The performance of the query is very fast with response times within 600–800 microseconds even for a large database with 24 million documents.

Contents

Uses of Provenance

- *Data Management*
  The captured provenance helps the user identify with ease where data objects are stored. Users can use Milieu to query the NoSQL data store using regular expressions to search previous job runs for a variety of information including data objects, status, etc.
- *Faults*
  The provenance can be used to determine if the job script terminated properly.

Flexibility

- Milieu provides support for multiple levels of provenance. This allows the scientist or system administrator to pick the level that is most suitable for their use case.
- The support for user annotations allows users to capture notes and metadata that are often lost. It should be noted that Milieu does not require users to make any changes to their job scripts.
- The support for user annotations allows users to capture notes and metadata that are often lost but is not required to collect other provenance information. Provenance initiation is user controlled but instrumentation is automated.

Operational considerations

- In our current implementation, all users are stored in a single collection. This optimizes the queries by system administrators performed across users.
- In future work, we plan to provide a start-time configuration to control per user collection if it is required in cases where the usage model might differ.
- Additionally, we operate MongoDB in a single-server mode. MongoDB allows sharding that allows us to easily scale up as data grows. MongoDB sharding is well-documented and evaluated and can be easily configured for MongoDB.

Scalable Storage

- The “big data” movement has seen a number of NoSQL data stores for horizontal scalability distributed over many servers. In this paper, we used Mongo database (MongoDB), a document-oriented data store, since it was well suited for semi-structured provenance documents.
- MongoDB is typically well suited for write-once, read-many times pattern data access, as is the case with Milieu. The NoSQL systems are evolving rapidly and it is possible that some other data store might provide additional features and/or better performance. Our architecture and methodology is not tied to the use of MongoDB and thus it will be possible to use other data stores with Milieu in the future.

Contents

CONCLUSION

- In this paper, we present the design, implementation and evaluation of Milieu, a minimally-intrusive, light-weight, multi-level provenance collection, storage and query framework. Milieu supports the collection of semi-structured provenance data from jobs and user commands on HPC systems.
- Our evaluation on NERSC production systems, including a petascale machine shows that the collection overhead is minimal.
- Milieu makes it possible to build a multi-tiered provenance architecture where storage for collection and storage for optimized queries can be separated. A multi-tiered architecture will be able to support a wider range of provenance queries and analyses that is difficult if not impossible today.

Impression

- This framework is interesting with the point that user can easily store provenance with a little overhead.
- It is needed to compare other framework to store provenance and data properties.