Reducing Technical Debt with Reproducible Containers

Tanu Malik
2019 BSSw Fellow
Assistant Professor
School of Computing
DePaul University
Chicago, IL

IDEAS-ECP Webinar, November 4th, 2020
Who am I

My expertise is:
Databases and distributed computing
Data provenance: history and lineage of data and software
Computational reproducibility: Repeating and recreating some one else’s work

Systems built: http://sciunit.run

I want to know more about:
Reproducibility case studies in HPC and how containers are used.

Problems I’m currently working on:
Provenance alignment: Using provenance to highlight sources of irreproducibility
State maintenance in lineage graphs: Making Jupyter Notebooks reproducible
Outline

PART 1: How technical debt affects reproducibility?

PART 2: If reproducible containers provide a start?

PART 3: Guidance and summary
PART 1: How technical debt affects reproducibility?
Monetary debt
Monetary debt meets the objective “sooner”
Technical debt\(^1\) is no different

\(^1\)A metaphor introduced by Ward Cunningham in 1992.
Technical debt\(^1\) is no different

\(^1\)A metaphor introduced by Ward Cunningham in 1992.
Technical debt is no different.

![Graph showing productivity over time with technical debt and journal deadline notations.]

Good scientific software

Poor scientific software

Technical debt

Journal deadline

Productivity

Time
Dimensions of Technical Debt

- Poor quality code
- Poor design
- Environment debt
- Documentation debt
- Testing debt
Consequence of Mismanaged Debt
Consequence of Mismanaged Debt
Dimensions of Scientific Technical Debt

• Poor quality code
• Poor design
• Environment debt
• Documentation debt
• Testing debt

Dimensions of Scientific Technical Debt

- Poor quality code
- Poor design
- Environment debt
- Documentation debt
- Testing debt
Bad bugs: The worst disasters caused by software fails

5 June 2013

Clever software can make our lives easier but a glitch can have disastrous consequences. In the past decades, computer bug catastrophes have caused deaths and disrupted lives on a large scale. Sally Adee takes us through six major software fails.

https://www.newscientist.com/gallery/software-bugs
A Scientist's Nightmare: Software Problem Leads to Five Retractions

Greg Miller
+ See all authors and affiliations

Science 22 Dec 2006:
Vol. 314, Issue 5807, pp. 1856-1857
DOI: 10.1126/science.314.5807.1856
Critiqued coronavirus simulation gets thumbs up from code-checking efforts

Influential model judged reproducible – although software engineers called its code ‘horrible’ and ‘a buggy mess’.

https://www.nature.com/articles/d41586-020-01685-y
Cost of Scientific Technical Debt
Supercomputing Artifact Description and Evaluation Initiative

https://sc20.supercomputing.org/planning-committee/
Lack of artifacts will reject a paper

- Unacceptable AD/AE: 1
- Submissions (Phase 2) with VG/E AD/AE: 24
- Submissions (Phase 2): 43
- Submissions (Phase 1) with VG/E AD/AE: 5
- Per reviewer: 80
- Submissions (Phase 1): 380

IDEAS-ECP Webinar, November 2020
Technical debt incurs burden

• Reproducibility is an after thought.
• Identifying files for an application is a challenge
• Missing workflows
  • Really, that data/algorithms should be part of the bundle?
• “Sticks” from reviewers work
  • Authors who have not taken AD/AE process seriously do submit additional work
• Time consuming task
  • No tools to check if everything relevant for the publication is submitted
• No mapping of experiments to content in the paper.
  • No infrastructure for efficiently verifying claimed results
PART 2: Do reproducible containers provide a start?
Reproducibility ecosystem

- Github
- OpenData.gov
- Figshare
- Zenodo.org
- Package managers
- AWS
- Sharing images via the cloud
- Docker.com

An introduction to Docker for reproducible research
C Boettiger - ACM SIGOPS Operating Systems Review, 2015 - dl.acm.org
Docker: Using containers from build to run

- Build images that captures applications requirements.
- Manually commit or use a recipe file.

- Push an image to DockerHub, a hosted registry, or a private Docker Registry.
- Share Images

- Use Docker Engine to pull images down and execute a container from the image.

https://www.exascaleproject.org/event/contHPC
Containers provide constrained resource isolation

- **CPU**
- **Memory**
- **Filesystem**
- **Network**
Authors must program a Dockerfile

```bash
# fetch node v4 LTS codename argon
FROM node:argon

# Request samplename build argument
ARG samplename

# Create app directory
RUN mkdir -p /usr/src/spfx-samples
WORKDIR /usr/src/spfx-samples

#Install app dependencies
RUN git clone https://github.com/SharePoint/sp-dev-fx-webparts.git .
WORKDIR /usr/src/spfx-samples/samples/$samplename

# install gulp on a global scope
RUN npm install gulp -g

# RUN ["npm", "install", "gulp"]
RUN npm install
RUN npm cache clean

# Expose required ports
EXPOSE 4321 35729 5432

# Run sample
CMD ["gulp", "serve"]
```

FROM buildkit-export AS buildkitd-oci_only
COPY --from=buildkitd-oci_only /usr/bin/buildkitd-oci_only /usr/bin/
ENTRYPOINT ["buildkitd.oci_only"]

```
# Copy together all binaries for contained worker mode
FROM buildkit-export AS buildkitd-contained_only
COPY --from=runc /usr/bin/runc /usr/bin/
COPY --from=buildkitd-contained_only /usr/bin/buildkitd-contained_only /usr/bin/
ENTRYPOINT ["buildkitd.contained_only"]

FROM alpine AS containerd-runtime
COPY --from=runc /usr/bin/runc /usr/bin/
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/containerd* /usr/bin/
COPY --from=containerd /go/src/github.com/containerd/containerd/bin/ctr /usr/bin/
VOLUME /var/lib/containerd
VOLUME /run/containerd
ENTRYPOINT ["containerd"]

FROM buildkit-$(BUILDKIT_TARGET)
```
Containers do not reduce technical debt

- Declarative encapsulation of dependencies for isolated execution
  - E.g. various shell utilities and library versions unknown to user
Automatic Encapsulation of Dependencies: The Sciunit
Key Idea: Identify dependencies during program execution

- Captures application dependencies during executions
- Repeats executions (with guarantees) within isolated environments
Sciunit: Audit

- Audit uses `ptrace` to observe dependencies and environment variables
  - Identifies binaries, libraries, scripts, and environment variables that application is dependent on.
- Dependencies are copied into a directory in the filesystem
- Inclusion of data files is optional
  - user may or may not want to package based on the size of the dataset.

Audits provenance during execution time


Auditing and Maintaining Provenance in Software Packages. In International Provenance and Annotation Workshop (IPAW), 97-109, 2014
Sciunit: Share as a Zip file or Docker container

- Sciunit Containment
- Computational Artifacts (from websites)
- Sciunit Log
- Provenance Graph
- Identification of Inputs, Outputs, Processes, Dependencies
- Documentation
- Docker File


IDEAS-ECP Webinar, November 2020
Sciunit: Repeat

- Sciunit uses namespace isolation during repeat
- Redirection of each call into the package
Sciunit steps and external requirements

1. Create

2. Share

3. Repeat

IDEAS-ECP Webinar, November 2020
Network-enabled Sciunit: Audit

Possible with Network-enabled Sciunit

Note:
1. Identify remote host & copy Sciunit to it
2&3. Configure & run task with Sciunit
4. Retrieve & manually merge

IDEAS-ECP Webinar, November 2020
Network-enabled Sciunit: Repeat on single node

**Note:**
1. Repeat all computations at root node.
2. Network system calls are supplied through the content data captured during the original audit.
Network-enabled Sciunit: Repeat on multiple nodes

Requirements:
1. Identical number of nodes
2. Descriptions of new hostnames or IP addresses

IDEAS-ECP Webinar, November 2020
Docker contains more digital artifacts than it needs for its use. At the time, in comparison with the kernel, libraries, dependencies, input files and so on. Most of an image, Docker must add all information about the Linux base kernel as well as libraries, dependencies, and input files that are touched during the execution, significantly reducing the container sizes. Moreover, it only containerizes the digital artifacts (libraries, dependencies or files) that are touched during the execution, whereas supports incremental recomputation at the container level, automatically memoizing function calls, whereas Python interpreter that speeds up script execution times by 40× longer time to build an image for VIC. As previously mentioned, to build a container, Docker takes longer to build an image than the original database; the larger the application, the longer it may take to build a container. This is because in time during auditing comes from the fact that FIE (307MB) and VIC (1.2GB) respectively. The increase in time of extra-time to build the containers for IQE (84KB), with SciInc projects with normal run-times, auditing and re-execution times of different projects.

TABLE I: Usecases descriptions.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code languages</td>
<td>R, Bash</td>
<td>C, C++, Python, C shell script, Fortran</td>
<td>Python</td>
</tr>
<tr>
<td>Source code files</td>
<td>29</td>
<td>97</td>
<td>5</td>
</tr>
<tr>
<td>Data files</td>
<td>14</td>
<td>11,481</td>
<td>5</td>
</tr>
<tr>
<td>Dependency files</td>
<td>659</td>
<td>357</td>
<td>112</td>
</tr>
<tr>
<td>Size of all files</td>
<td>306.6 MB</td>
<td>1.2 GB</td>
<td>22 MB</td>
</tr>
<tr>
<td>Normal run time</td>
<td>286.756 s</td>
<td>40.259 s</td>
<td>5.226 s</td>
</tr>
</tbody>
</table>


Sciunit (native) versus Docker sizes

As shown, in Figure 3, Docker images are 19X, 7X and 2.5X larger than SciInc and IncPy, respectively. This comparison measures the effectiveness of containerizing an application in terms of storage, creation time (i.e., containerizing or building time), and re-execution time. We evaluate three usecases: (i) Chicago Food Inspections Evaluation (FIE) [16], (ii) the Variable Infiltration Capacity (VIC) [17], and (iii) Incremental Query Execution (IQE) [18]. The detailed descriptions of FIE, VIC, and IQE are shown in Table I.
Sciunit audit and repeat times

![Bar chart showing time in seconds for different projects and tasks.

- Normal run
- Sci. Audit
- Sci. exec
- Doc. build
- Doc. exec

Selected projects:
- IQE
- FIE
- VIC

Time (in Seconds)

- IQE: 5, 6, 6, 8
- FIE: 287, 305, 288, 1235, 453
- VIC: 40, 70, 40, 631, 63

IDEAS-ECP Webinar, November 2020
Experiments

• NASA Parallel Benchmark:
  • Data transferred (~524 KB (class A) & ~268 KB (class B))

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>#Calls</th>
<th>Meta Audit</th>
<th>Content Audit</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB BT-MZ.A.2</td>
<td>20.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPB LU-MZ.A.2</td>
<td>15.74</td>
<td>190</td>
<td>~ 2.1%↑</td>
<td>~ 5.3%↑</td>
</tr>
<tr>
<td>NPB SP-MZ.A.2</td>
<td>14.84</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPB BT-MZ.B.2</td>
<td>83.95</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPB LU-MZ.B.2</td>
<td>71.02</td>
<td>190</td>
<td>~ 0.8%↑</td>
<td>~ 3.2%↑</td>
</tr>
<tr>
<td>NPB SP-MZ.B.2</td>
<td>59.12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• VIC

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Meta Audit</th>
<th>Content Audit</th>
<th>Replay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data retrieval</td>
<td>146.5±1.8</td>
<td>0.2%↑</td>
<td>134.5%↑</td>
<td>53.0±3.0</td>
</tr>
<tr>
<td>Other steps</td>
<td>varies</td>
<td>2% - 30% ↑ overhead</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sample Interaction of Sciunit

1. > sciunit create FIE
2. > sciunit exec ./FIE.sh ./DATA/weather_201710.Rds
   0. Download...
      1. Calculate violation matrix...
      2. Calculate heat map...
      3. Generate model data with ./DATA/weather_201710.Rds...
      4. Apply random forest model...
      5. Evaluation...
3. > sciunit list
   e1 Dec 4 12:44 ./FIE.sh ./DATA/weather_201710.Rds
4. > sciunit show
   id: e1
   sciunit: FIE
   command: ./FIE.sh ./DATA/weather_201710.Rds
   size: 306.6 MB
   started: 2017-12-04 12:44
5. > sciunit push
   ...
   Title for the new article: FIE
   new: 306.6 MB [01:05, 4.72MB/s]
6. > sciunit copy
   mSLTTj#

1. > sciunit open mSLTTj#
   Opened Sciunit FIE
2. > sciunit list
   e1 Dec 4 12:44 ./FIE.sh ./DATA/weather_201710.Rds
3. > sciunit repeat e1
   ...
   0. Download...
   1. Calculate violation matrix...
   2. Calculate heat map...
   3. Generate model data with ./DATA/weather_201710.Rds...
   4. Apply random forest model...
   5. Evaluation...
4. > sciunit given '/tmp/weather_201801.Rds' e1 %
   ...
   1. Generate model data with '/tmp/weather_201801.Rds
   2. Apply random forest model...
   3. Evaluation...
5. > sciunit list
   e1 Dec 4 12:44 ./FIE.sh ./DATA/weather_201710.Rds
   e2 Dec 14 2:44 ./FIE.sh ./tmp/weather_201801.Rds
Container Limitations

- Container either include the data or exclude the data
  - The decision is binary but does not consider necessary and sufficient data
Container Debloating: MiDAS
Example

```c
void file_read(int bytes) {
    int fd, sz;
    char *c = (char *) calloc(bytes, sizeof(char));
    fd = open("test.txt", O_RDWR);
    lseek(fd, 100, SEEK_SET);
    sz = read(fd, c, bytes);
}
```

<table>
<thead>
<tr>
<th>File</th>
<th>Offsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.txt</td>
<td>100, 150</td>
</tr>
<tr>
<td>test.txt</td>
<td>100, 190</td>
</tr>
<tr>
<td>test.txt</td>
<td>100, 230</td>
</tr>
<tr>
<td>test.txt</td>
<td>100, 450</td>
</tr>
</tbody>
</table>
Example

void file_read(int bytes) {
    int fd, sz;
    char *c = (char *) malloc(bytes, sizeof(char));
    fd = open("test.txt", O_RDWR);
    lseek(fd, 100, SEEK_SET);
    sz = read(fd, c, bytes);
    
    test.txt, 100, 150
    test.txt, 100, 190
    test.txt, 100, 230
    test.txt, 100, 450
}
MiDAS: Minimizing DAtasetS

**WHAT**

Automatically identify & include ONLY relevant data chunks with application

**HOW**

Map high level user inputs to file offsets
Partial Evaluation & LLVM

• **Partial Evaluation**→ optimization technique to prune codebase
  • Uses static inputs to generate a specialized program to accept remaining dynamic inputs

```c
#include <math.h>
float compute_building_height(float building_distance){
    float viewing_angle = pi/4;
    float building_height =
        compute_opposite(building_distance, viewing_angle);
    return building_height;
}
float compute_opposite(float adjacent, float angle){
    float opposite = adjacent * tan(angle);
    return opposite;
}
```

(a) Original code

```c
float compute_building_height(float building_distance){
    float building_height =
        compute_opposite_specialized(building_distance);
    return building_height;
}
float compute_opposite_specialized(float adjacent){
    float opposite = adjacent * 1;
    return opposite;
}
```

(b) Specialized code
MiDAS

1. Instrumentation of Code
2. Code Execution
3. Data Chunk Extraction
4. Specialization

Original bitcode

Specialized bitcode with data chunks

specialization inputs

instrumented bitcode

execution traces

extracted data chunks

Instrumentation of Code

Code Execution

Data Chunk Extraction

Specialization
I/O Specialization

• Replace I/O call & \textit{preserve functionality}
  • Extracted file data in global variable $\rightarrow$ fileData
  • Copy from global variable to \textit{read} buffer $\rightarrow$ memset
  • Update all I/O call variables $\rightarrow$ return value of \textit{read}
  • I/O call instruction removed $\rightarrow$ read

Original LLVM assembly code segment

\[
%94 = \text{load i32, i32* } %9, \text{ align 4}
%95 = \text{sext i32 } %94 \text{ to i64}
%96 = \text{call i64 } \text{@read(i32 } %92, \text{i8* } %93, \text{i64 } %95) \\
%97 = \text{trunc i64 } %96 \text{ to i32}
\text{store i32 } %97, \text{i32* } %13, \text{ align 4}
%98 = \text{load i32, i32* } %12, \text{ align 4}
\]

I/O Specialized LLVM assembly code segment

\[
%94 = \text{load i32, i32* } %9, \text{ align 4}
%95 = \text{sext i32 } %94 \text{ to i64}
%96 = \text{bitcast } [17 \text{ x i8}]* @\text{fileData to i8}*
\text{call } \text{void } \text{@llvm.memcpy.p0i8.p0i8.i64(i8* } %93, \text{i8* } %96, \text{i64 } %95, \text{i32 1, i1 false)}
%97 = \text{alloca i64}
\text{store i64 } %95, \text{i64* } %97
%\text{loadRetVal } = \text{load i64, i64* } %97
%98 = \text{trunc i64 } %\text{loadRetVal} \text{ to i32}
\text{store i32 } %98, \text{i32* } %13, \text{ align 4}
%99 = \text{load i32, i32* } %12, \text{ align 4}
\]
Specializing I/O Calls in Scientific Libraries

Python apps
Data access
NetCDF4-python module
Interface
NetCDF C lib
Fast I/O processing & storage
HDF5 C lib
Build from source, instrument, specialize I/O calls
Results | Percentage of File Accessed

- Larger files generated from 30 MB NetCDF data file
- Rewriting data for multiple timesteps
- Data accessed corresponding to *temperature* attribute

<table>
<thead>
<tr>
<th>Total Size</th>
<th>Accessed Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 MB</td>
<td>6.6 MB</td>
</tr>
<tr>
<td>700 MB</td>
<td>154 MB</td>
</tr>
<tr>
<td>1.4 GB</td>
<td>0.3 GB</td>
</tr>
<tr>
<td>9 GB</td>
<td>1.98 GB</td>
</tr>
<tr>
<td>12.8 GB</td>
<td>2.82 GB</td>
</tr>
</tbody>
</table>

Applications often access only a subset of a large dataset.
PART 3: Summary and Guidance
Summary

• Technical debt affects reproducibility of scientific claims.
  • Process for evaluation of scientific claims is being rethought.
  • Artifact description and evaluation are becoming part of conferences

• Better reliability is needed.
  • Containers will be a prominent choice but their reliability is poor
    • Dependencies must be specified
    • Inefficient to use
    • No guarantees for execution verification
    • Not meant for interactive programs

• New light-weight methods: Sciunit, MiDAS
Use Sciunit for your next paper submission!

1. Tools downloaded ~850 times (tracked using pip)
2. 8 active contributors to the project
3. Actively used in geoscience disciplines that develop computational models and data-analytic pipelines

Website: [http://sciunit.run](http://sciunit.run)
Issues and contribution: pr@sciunit.run
Guidance for Improving Reproducibility
Guidance for Improving Reproducibility

https://bssw.io/items?topic=reproducibility

Guidance for Improving Reproducibility

- Concurrency
- Hardware
- Bugs outside the application
- Algorithmic randomness
- Application complexity
- Execution State

Identify sources of irreproducibility
Guidance for Improving Reproducibility

Metadata: Provenance, Annotations, Snapshots

- Concurrency
- Hardware
- Bugs outside the application
- Algorithmic randomness
- Application complexity
- Execution State

https://bssw.io/items?topic=reproducibility
Guidance for Improving Reproducibility

Methods for analyzing the metadata

- Concurrency
- Hardware
- Bugs outside the application
- Algorithmic randomness
- Application complexity
- Execution State

https://bssw.io/items?topic=reproducibility
Acknowledgements

Yuta Nakamura
Ph.D. student

Raza Ahmad
Research Engineer

Nithin Manne
M.S Student

Jason Chuah
M.S. student

Zhihao Yuan
Research Engineer

Ton That Dai Hai
Postdoctoral Associate

Jason Chuah
Ph.D student @UVA

IDEAS-ECP Webinar, November 2020
Acknowledgements

Ashish Gehani
SRI International

Ian Foster
UChicago & ANL

Dave Tarboton
Utah State

Jon Goodall
Univ. of Virginia

Scott Peckham
Univ of Colorado Boulder

Eunseo Choi
Univ of Memphis
Acknowledgements | Funding

NSF CNS-1846418, ICER-1639759, ICER-1661918
BSSw Fellowship
Bloomberg Foundation
DePaul Seed Grants
Questions

• tanu.malik@depaul.edu
Example

How to run pySUMMA locally

Installation and Usage

pySUMMA requires Python 3.6 and following packages:

- xarray 0.10.7: N-D labeled arrays and datasets in python
- numpy 1.16.1: the fundamental package for scientific computing with Python
- matplotlib 3.0.2: a Python 2D plotting library
- seaborn 0.9.0: statistical data visualization
- jupyterthemes 0.20.0: select and install a Jupyter notebook theme
- h5restclient 1.3.3: HydroShare REST API python client library
- ipyleaflet 0.9.2: A jupyter widget for dynamic Leaflet maps
- Linux Environment (VirtualBox 5.2.8)
  - Ubuntu-16.10 executable
  - Ubuntu-16.04 executable

Download and Install pySUMMA:

1) Download pySUMMA

```bash
$ git clone https://github.com/uwa-hydroinformatics/pysumma.git
```
### Listed Packages

<table>
<thead>
<tr>
<th>Listed Packages</th>
<th>Identified Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>xarray {0.10.7}</td>
<td>xarray {0.10.7}</td>
</tr>
<tr>
<td>numpy {1.16.1}</td>
<td>numpy {1.16.1}</td>
</tr>
<tr>
<td>matplotlib {3.0.2}</td>
<td>matplotlib {3.0.2}</td>
</tr>
<tr>
<td>hs-restclient {1.3.3}</td>
<td>hs-restclient {1.3.3}</td>
</tr>
<tr>
<td>ipyleaflet {0.9.2}</td>
<td>ipyleaflet {0.9.2}</td>
</tr>
<tr>
<td>seaborn {0.9.0}</td>
<td></td>
</tr>
<tr>
<td>jupyterthemes {0.20.0}</td>
<td></td>
</tr>
</tbody>
</table>

### Identified Sub-Packages

- Pygments {2.2.0}
- asyncio backcall {0.1.0}
- blinker {1.3}
- certifi {2018.10.15}
- cftime {1.0.2.1}
- geopandas {0.4.0}
- html http ipykernel {5.1.0}
- ipython-genutils {0.2.0}
- ipython {7.1.1}
- ipywidgets {7.4.2}
- jedi {0.13.1}
- jupyter-core {4.4.0}
- netCDF4 {1.4.2}
- pandas {0.23.4}
- parso {0.3.1}
- pexpect {4.6.0}
- prompt-toolkit {2.0.7}
- ptyprocess {0.6.0}
- pyparsing {2.3.0}
- pyssumma {0.1}
- pytz {2018.7}
- pynvml {17.1.2}
- requests-oauthlib {1.0.0}
- requests-toolbelt {0.8.0}
- tornado {5.1.1}
- traitlets {4.3.2}
- traittypes {0.2.1}
- wcwidth {0.1.7}

### Python Built-In Packages

- charget
- collections
- concurrent
cyl
- dateutil
dis
- functools
- email
- encodings
- idna
- importlib
- jinja2
- json
- logging
- markupsafe
- multiprocessing
- oauthlib
- pkg_resources
- pydoc_data
- requests
- sqlite3
- unittest
- urllib
- urllib3
- xml
Current and Future Work

• Developing Sciunit audit and repeat with checkpoint-restart
  • Compute- and data-analytic models that vary several parameters and are reexecuted multiple times to test their reproducibility.
  • Useful for Jupyter Notebooks

• Sciunit for reproducibility will provide provenance-based guarantees
  • Several cyberinfrastructure for Artifact Evaluation (OCCAM, CKFoundation)
  • Provenance-based guarantees are missing

• Developing MiDAS for different inputs