

A Short Story of Efficiently Using Two Open-Source Applications on Stampede

Ritu Arora

Texas Advanced Computing Center

June 16, 2015

Email: rauta@tacc.utexas.edu



Part-1

- **ECSS Project:** Exploration and Triage of Complex Archaeological Collections
- **PI:** Jessica Trelogan, Institute of Classical Archaeology, UT Austin
- **Project Status:** Successfully completed last year (June 2014)



THE INSTITUTE OF CLASSICAL ARCHAEOLOGY

The Institute of Classical Archaeology (ICA) carries out multi-disciplinary archaeological research, conservation, and cultural resource management projects in the territory of ancient Greek colonies in southern Italy and on the Black Sea coast of Ukraine. ICA was established as a research unit in the College of Liberal Arts at the University of Texas, Austin, in 1974. Over the past 30 years, with major support from public and private sources, ICA has developed long-term projects to explore the agricultural hinterlands of ancient **Metapontum** and **Croton** in Italy and the territory and urban area of ancient **Chersonesos** in Crimea, Ukraine. Principal collaborators include the Archaeological Superintendencies of Calabria and Basilicata and the National Preserve of Tauric Chersonesos at Sevastopol. ICA's publications and research have brought it international recognition as a leader in the study of rural populations in the Greek and Roman world.

Chersonesos on UNESCO World Heritage List

Croton • Metapontum • Chersonesos



**russskaya
versia**

Project Characteristics

- Non-Traditional HPC users engaged, training of stakeholders and precise documentation along with workflow automation were the key to success
 - Information on how to create a portal account to running the steps in their workflow on Stampede needed to be as clearly documented as possible
 - The training document was written so that anyone from the PI's group can be productive at using the workflow on Stampede in the least amount of amount
 - The effort involved in developing the documentation and training the PI's group motivated us to organize workshops at the IEEE BigData 2014 and IEEE BigData 2015 conferences
 - Helped us better understand the challenges that a non-traditional HPC user can face while using the Supercomputing resources
 - Trained > 40 non-traditional HPC users at the workshop last year

Extracting Metadata Using DROID

The screenshot shows the 'File profiling tool (DROID)' page on The National Archives website. The page has a dark header with the National Archives logo and a search bar. A breadcrumb trail indicates the path: Home > Information management > Our services > Digital Continuity Service > File profiling tool (DROID). A left-hand navigation menu lists various services, with 'File profiling tool (DROID)' highlighted. The main content area is titled 'File profiling tool (DROID)' and contains a paragraph explaining that DROID is a free software tool for profiling file formats. It also provides links to download the latest version for free and to contact the digital continuity team for more information or a live demo.

The National Archives

You are here: [Home](#) > [Information management](#) > [Our services](#) > [Digital Continuity Service](#) > File profiling tool (DROID)

File profiling tool (DROID)

DROID stands for Digital Record Object IDentification. It's a free software tool developed by The National Archives that will help you to automatically profile a wide range of file formats. For example, it will tell you what versions you have, their age and size, and when they were last changed. It can also provide you with data to help you find duplicates. Profiling your file formats helps you to manage your information more effectively. It helps you to identify risks (and therefore plan mitigating actions). It can also help you to save money, for example by supporting data reduction.

You can [download our latest version of DROID for free](#). For previous versions go to <http://droid.sourceforge.net/>. For more information, see our [PRONOM resource](#).

If you are interested in using DROID at your organisation, would like a live DROID demo, or are experiencing any problems using DROID, contact us at digitalcontinuity@nationalarchives.gsi.gov.uk.

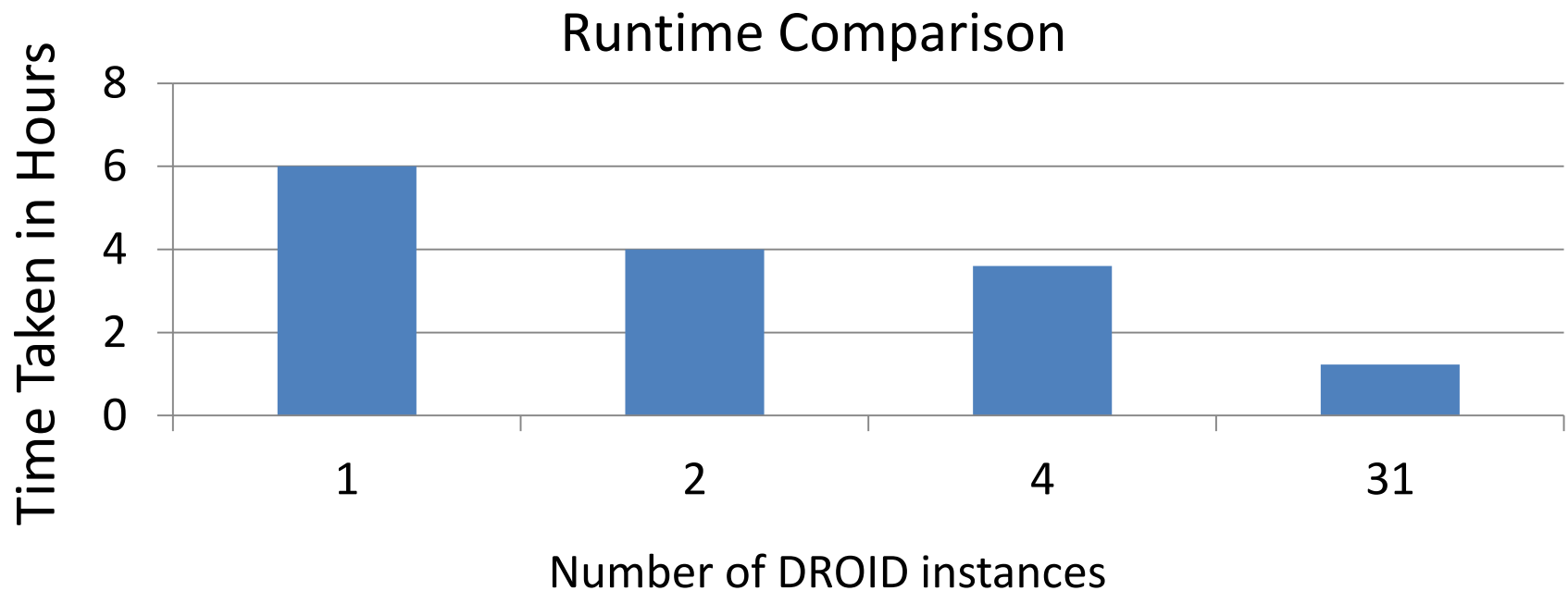
Use DROID to manage digital continuity

Using DROID can help you to manage a specific information risk - the risk to digital continuity. DROID helps you to gather information you need to understand your information assets. This can help you to define your digital continuity requirements, assess where your information is at risk and plan mitigating action.

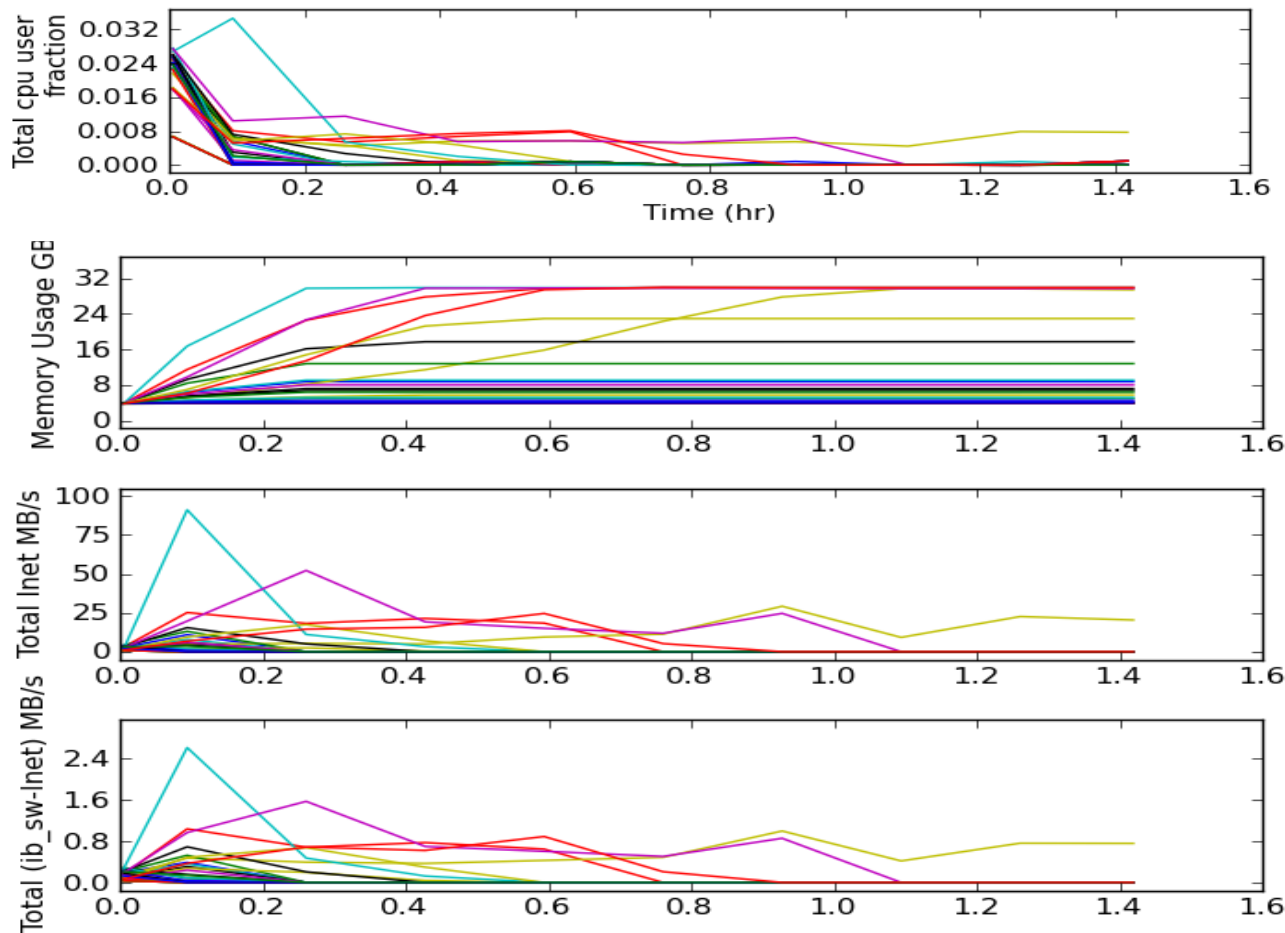
- Running one DROID instance to profile a subset of the 4.3 TB archeological data collection used to take > 2 days on a local server
- **6 to 8 hours** on a single node on Stampede

Running 31 Parallel Instances of DROID

Ran multiple DROID instances, on different sub-directories of the 4.3 TB data collection, using multiple nodes on Stampede



Running 31 Parallel Instances of DROID



Graphs show CPU usage, memory usage, and network bandwidth usage during parallel runs of DROID on Y-axis and time consumed on X-axis

One of the Issues with the Version of DROID Used

- Each DROID run creates a database profile
- Profile associated with a run is supposed to have a unique id
- During multiple simultaneous runs of DROID for distributing the metadata extraction, the jobs started crashing due to database conflicts
 - Same profile id was getting assigned to multiple profiles
- Upon looking into the DROID code it was found that it is using current system time in milliseconds for generating the seemingly unique profile id
 - This resulted in having multiple profiles to have the same id

A Very Simple and an Unideal Fix

- This bug in DROID was reported to the development team but we did not have time to fix the code and do the regression testing ourselves, and there were couple of other bugs too
- Therefore, a quick-fix approach was adopted
 - Staggered the time at which each DROID instance starts so that each of it gets a different profile-id
 - Wrote a wrapper bash script that submits multiple independent jobs depending upon the required number of independent DROID runs
 - Determining the amount of data to be processed by each DROID run is a separate step in the workflow prior to the step of running the wrapper script
 - Stakeholder gets to choose whether DROID needs to be run on certain directories or on all the directories in their complex, deeply-nested data collection
 - It was not ideal to run all the jobs as one set from a single job script because the data distribution for each DROID instance was not balanced and hence, certain DROID instances would finish running in 25 minutes while others may take more than 1 hour

Wrapper Script to Create and Submit Multiple DROID Jobs

```
#!/bin/bash
count=0
mkdir batchSubmit
cd batchSubmit
cat ../paramlist7 | while read cmd
do
    count=$((count + 1))
    cp ../test_independent_submission_1.sh submit"$count".sh
    cat submit"$count".sh | sed "s#commandGoesHere#$cmd#g" > test"$count".sh
    rm submit"$count".sh
    sleep 5
    sbatch test"$count".sh
done
```



The Script

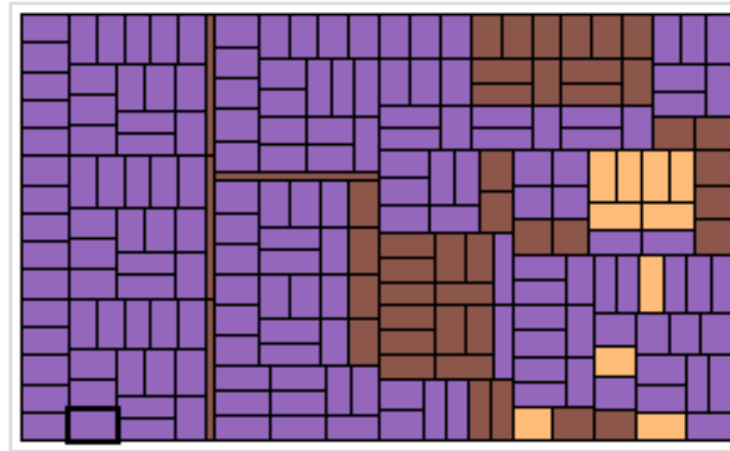
`“test_independent_submission_1.sh”` that
is Referenced in the Previous Slide

```
#!/bin/bash
#SBATCH -J droid # job name
#SBATCH -o droid.o%j # output & error file name (%j is jobID)
#SBATCH -n 1 # total number of tasks requested
#SBATCH -p normal # queue (partition) - normal, development, etc.
#SBATCH -t 3:00:00 # run time (hh:mm:ss)
#SBATCH -A icadigarchives
set -x
date
export JAVA_TOOL_OPTIONS="-Xmx20G -XX:-DisableExplicitGC"
cd ..
commandGoesHere
date
```

Happy-Ending: Information Visualization Using Tableau

FORMAT_NAME

-  Graphics Interchange Fo...
-  JPEG File Interchange Fo...
-  Portable Network Graphics



FORMAT_NAME

- ☐ (All)
- ☐ Null
- ☐ Cascading Style Sheet
- ☐ Extensible Markup Langu...
- ☒ Graphics Interchange Fo...
- ☐ GZIP Format
- ☐ Hypertext Markup Langu...
- ☐ Java Archive Format
- ☐ Java Script File

✓ Keep Only

✗ Exclude



EXT: jpg
FILE_PATH: /work/01698/rauta/tools/l_ics_2013.0.028/doc_icsxe/icsxe_gsg_files
/Linux_itac_files/image014.jpg
FORMAT_NAME: JPEG File Interchange Format
PARENT_ID: 167
TYPE: File
Count of ID: 1

Part-2

- **ECSS Project:** Modification of FLASH Astrophysics Code for Optimizing the Parallel I/O
- **PI:** None, project developed out of a consulting ticket
- **Project Characteristics:** Optimization of file I/O on Luster File System
- **Project Status:** Successfully completed last year (December 2014)

Project Background

- This project was created after an XSEDE user was blocked from running jobs on Stampede
- The user was running the FLASH astrophysics code using 7000+ cores when two Lustre servers got overloaded (and eventually unresponsive) during the step in which a large file was being read
- Such intensive I/O activity impeded other users trying to use the file-system
- The user was requested to set the stripe-count manually as a temporary workaround
 - Temporary because there is a chance that one could forget to change the default stripe count if the job is run from a new directory – hence, can get blocked again
- An ECSS project was suggested to modify the FLASH code so that a copy of the optimized code (generated at the end of the ECSS project) can be made available not only to this particular user but to other users as well

More Details Related to the Problem Scenario

- The user was reading a restart file just once in a job run using 7000+ cores, after which checkpoints are written every 10 minutes or so, till the end of the job
- The restart file was 250 GB in size with the default stripe of two
- All 7000+ cores trying to read the single file was not efficient and monopolized the two servers so much that they were unable to respond to other requests and became unresponsive

Performance While Using Different Stripe Counts

- Performance tests were done with the Flash astrophysics code on Stampede
 - Number of cores used in this testing: 7680, the size of checkpoint file that was read: 189GB
 - Please note: tests were done when the system was fully active and hence there was competition from other users while writing to the shared filesystem – therefore, the numbers presented here are not absolute best timings we could get on a quiet system
- It was found that with the stripe count of 80, the time taken for reading the file was lowest, while with stripe count 40, the time taken for writing was lowest
- The stripe count of 40 looked optimal as beyond that, even though the time for reading continues to go down but it goes down at a slower rate, however, the time for writing the checkpoint file increases noticeably in going up from stripe count 40 to 60

LFS Stripe Count #	Time for reading the checkpoint (in sec)	Time for writing the first checkpoint (in sec)
2	515.528	494.212
30	61.182	175.892
40	53.445	108.782
60	46.913	182.65
80	40.57	183.107

Solution

- Use `MPI_Info_set` function to set the striping factor
 - Create a String and assign value of `maxProcs`
 - As per our tests on Lustre File System, set stripe count to: `minimumOf (number of MPI processes, 80)`
- The key changes that were needed for this project were made in three files, two of which are listed below

`/source/IO/IOMain/hdf5/parallel/io_h5file_interface.c`

`/source/IO/IOMain/pnetcdf/io_ncmpi_interface.c`

```
int LEN = 10;

char stringProcs[LEN];

snprintf(stringProcs, LEN, "%d", maxProcs);

ierr = MPI_Info_set(FILE_INFO_TEMPLATE, "striping_factor", stringProcs);
```

Thanks!

Questions or Comments?