Scaling an ACCRE research project to Production at the Clinic

A Hodge-podge of Cool ACCREesque and Pythonic Tips and Tricks

Chris Moth  June 28, 2018
Vanderbilt Program in Personalized Structural Biology

https://my.vanderbilt.edu/psbp/

**Acknowledgements**: Chuck Sanders, Jon Kropski, Christine Lovly, John Phillips, John Newman, Joy Cogan, Carlos Arteaga, Rizwan Hamid, Brett Kronke, and many more!
* I am Earth's leading authority on florist shop automation...

Daisy POS System

Daisy, the industry's original floral management system.

Teleflora's Daisy POS System is the preferred choice of over 3,000 florists throughout the United States and Canada.

Here's why:

- High-speed credit card processing
- High-speed Dove Network®
- Advanced credit card security
- Address Verification Service (AVS)
- Multi-payment capability
- Ability to collect customer information for e-mail marketing

* ... with a Ph.D. in Chemistry who lives in easy bicycle range
Ultimate Global Dream: Accurately predict the impact of DNA mutations on protein structure and function. (and explain disease)

3.2 x 10^9 DNA base pairs
Found on 23 pairs of chromosomes

1.5% of DNA is 21,000 Genes coding for proteins: Structure, Function, Networks (Metabolism, Transport....)
Whole Genome/Exome Sequence Data are Plentiful

Counts of Sequenced Humans

- 2014: 229,000
- 2015: 422,000
- 2016: 952,000
- 2017: 1,620,000

Source: Illumina
PDB Statistics: Overall Growth of Released Structures Per Year

X-ray / NMR / Cryo-EM ....And a lot more models too!
How does our ACCRE “pipeline” software attempt to help?

On Monday, Dr. asks, “SIRI, what do we know about Felicia’s mutations?”

Siri says, “I can help, but I need a .csv file”

<table>
<thead>
<tr>
<th>gene</th>
<th>mRNA transcript Refseq ID</th>
<th>protein mutation</th>
<th>mRNA Isoform-specific Uniprot Protein ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 GBA</td>
<td>NM_001005741</td>
<td>R502C</td>
<td>P04062-1</td>
</tr>
<tr>
<td>1 KCNRG</td>
<td>NM_173605</td>
<td>R33C</td>
<td>Q8N5I3-1</td>
</tr>
<tr>
<td>2 KIF5C</td>
<td>NM_004522</td>
<td>N855I</td>
<td>O60282-1</td>
</tr>
<tr>
<td>3 UTP14C</td>
<td>NM_021645</td>
<td>I187N</td>
<td>Q5TAP6</td>
</tr>
<tr>
<td>4 APOBR</td>
<td>NM_018690</td>
<td>E361D</td>
<td>Q0VD83-4</td>
</tr>
<tr>
<td>5 RELN</td>
<td>NM_173054</td>
<td>M709V</td>
<td>P78509-2</td>
</tr>
</tbody>
</table>

On Monday, Dr. asks, “SIRI, what do we know about Felicia’s mutations?”

Siri says, “I can help, but I need a .csv file”
PDBMap Processing Pipeline

Structure Processing
- UniProt ID
- UniProt ID Mapping
- ModBase Summary
- ModBase IDs
- PDB Bank
- PDB IDs
- PDB Structures
- Biological Assemblies
- Processing with Biopython
- Structures
- Models

Data Processing
- VCF
- BED
- PEDMAP
- Ensembl Variant Effect Predictor
- Variant Transcript Consequence Annotation
- rsIDs
- Chain-Aligned Ensembl Transcripts
- Alignment (SIFTS, Biopython)
- Matching Ensembl Transcripts

Analysis
- Scikit-Learn
- Ripley’s K
- PathProx

Visualization
- Chimera
- IntersectBED
- Transcript Matching
- Data and Annotation Intersection
<table>
<thead>
<tr>
<th>Gene</th>
<th>mRNA transcript</th>
<th>protein</th>
<th>mRNA Isoform-specific Refseq ID</th>
<th>mutation</th>
<th>Uniprot Protein ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 GBA</td>
<td>NM_001005741</td>
<td>R502C</td>
<td>P04062-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 KCNRG</td>
<td>NM_173605</td>
<td>R33C</td>
<td>Q8N5I3-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 KIF5C</td>
<td>NM_004522</td>
<td>N855I</td>
<td>O60282-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 UTP14C</td>
<td>NM_021645</td>
<td>I187N</td>
<td>Q5TAP6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 APOBR</td>
<td>NM_018690</td>
<td>E361D</td>
<td>Q0VD83-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 RELN</td>
<td>NM_173054</td>
<td>M709V</td>
<td>P78509-2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To output.....
• We store and process a lot of data (from all over) in lots of places
• The pipeline launches 25-ish SLURM jobs per patient mutation
• The pipeline creates a summary report for each mutation, and case
• Cool 3D graphics

Questions…. ?
RETURNING February 9-10, 2019

DESCRIPTION

Guess who's back?! PyTennessee is back again, and this year we're going to celebrate exactly the same way we did last year! We'll have excellent keynotes, engaging and useful talks, good food, and great company. This year we're getting together on February 10th and 11th, 2018 at the Nashville School of Law. We will have three talk tracks, 1 tutorial track, 2 keynotes, lightning talks, and a Young Coders class. We'll provide breakfast, lunch, and snacks onsite with vegetarian options available provided by Sifted.co, all served with a side of Southern hospitality. Included in your ticket price is your PyTN...
A textbook mistake

#!/usr/bin/env python2.7
include os

if not os.path.exists("dirname"):
    os.makedirs("dirname")
Reproducible 😊 Random Numbers

#!/usr/bin/env python2.7
#=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-#
# Slurm Parameters
#SBATCH --mail-user=chris.moth@vanderbilt.edu
#SBATCH --mail-type=end
#SBATCH --time=00:01:00
#SBATCH --mem=50MB
#SBATCH --account=capra_lab_csb
#SBATCH --output=0random.stdout_stderr
#=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-#
import random, datetime, calendar, time
secondsSinceJanuary1970 = calendar.timegm(time.gmtime())
random.seed(secondsSinceJanuary1970)

firstRandomInteger = random.randint(1,1000000)

brilliantReportConclusion = \\
  "With seed %d, the reproducible answer is: %d"% \\
  (secondsSinceJanuary1970,firstRandomInteger)

print brilliantReportConclusion

with open("0random.txt","a") as f:
    f.write(brilliantReportConclusion + "\n")
I submit: `$ sbatch --array=0-9 0random.py`

I got: Only one line of stdout from SLURM:

```
$ cat 0random.stdout_stderr

With seed 1530136846, the reproducible answer is: 224123

Why only one line of stdout?
```

10 lines were written directly to `0random.txt` by the 10 `0random.py` invocations:

```
With seed 1530135527, the reproducible answer is: 448695
With seed 1530135527, the reproducible answer is: 448695
With seed 1530135528, the reproducible answer is: 103138
With seed 1530135586, the reproducible answer is: 911719
With seed 1530135586, the reproducible answer is: 911719
With seed 1530135588, the reproducible answer is: 634809
With seed 1530135598, the reproducible answer is: 224123
With seed 1530135598, the reproducible answer is: 224123
With seed 1530135598, the reproducible answer is: 224123
...`

Why isn't `0random.txt` jumbled? Why are there 10 lines in that file, but not stdout? How might you fix the duplicate answers?
Are microsecond-resolution seeds good enough?

Please write down your birth month and day  MM-DD format.
#!/usr/bin/env python2.7

# See https://slurm.schedmd.com/job_array.html
# See https://www.vanderbilt.edu/accre/documentation/parallel/
# Slurm Parameters
#SBATCH --mail-user=chris.moth@vanderbilt.edu
#SBATCH --mail-type=end
#SBATCH --time=00:01:00
#SBATCH --mem=50MB
#SBATCH --account=capra_lab_csb
#SBATCH --array=0-9
#SBATCH --output=1bigtext.out
#SBATCH --open-mode=append

import os

# set array_task_id = $SLURM_ARRAY_TASK_ID
array_task_id = os.environ.get("SLURM_ARRAY_TASK_ID")

for i in range(999999):  # Write million lines of text
    print "Task ",array_task_id,"line ",i," is my favorite row of data"

print "Line 1000000 - Great job!"

# Let's go!
$ sbatch 1bigtext.py
Submitted batch job 1073649
WOW! The Redhat7 cluster can get REALLY FAST!

# 1.5 seconds later:
$ squeue -u mothcw

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
</table>

# UUGH – What happened to my job??????
$ ls -l
-rw-r--r-- 1 mothcw capra_lab 488,888,670 Jun 28 04:55 1bigtext.out

# ALWAYS DO (AUTOMATED) QUALITY CONTROL ON OUTPUT - *ESPECIALLY ACCRE*
$ wc -l 1bigtext.out
10,000,000 1bigtext.out Super! We got 10,000,000 lines of output

$ cat -n 1bigtext.out | head -5
  1 Task 1 line 0 is my favorite row of data
  2 Task 1 line 1 is my favorite row of data
  3 Task 1 line 2 is my favorite row of data
  4 Task 1 line 3 is my favorite row of data
  5 Task 1 line 4 is my favorite row of data

First few lines of file look great. Task 1 launched first. Cool.
Quality control (of 10,000,000 line file) continued.......  

```
$ grep 'Great job' 1bigtext.out | cat -n
1  Line 1000000  - Great job!
2  Line 1000000  - Great job!
3  Line 1000000  - Great job!
4  Line 1000000  - Great job!
5  Line 1000000  - Great job!
6  Line 1000000  - Great job!
7  Line 1000000  - Great job!
8  Line 1000000  - Great job!
9  Line 1000000  - Great job!
10 Line 1000000  - Great job!
```

10 occurrences of the 1,000,000th line!

```
$ cat -n 1bigtext.out | grep Great
1897703  Line 1000000  - Great job!
2000000  Line 1000000  - Great job!
3000000  Line 1000000  - Great job!
4936824  Line 1000000  - Great job!
5000000  Line 1000000  - Great job!
6000000  Line 1000000  - Great job!
7000000  Line 1000000  - Great job!
8911412  Line 1000000  - Great job!
9000000  Line 1000000  - Great job!
10000000 Line 1000000  - Great job!
```

Oops, something ain’t right
What’s up around line: 4936824?

```
$ cat -n 1bigtext.out | grep -C5 4936824 # Look at 5 lines before and after a disconnect
4936819 Task  3 line 999994  is my favorite row of data
4936820 Task  3 line 999995  is my favorite row of data
4936821 Task  3 line 999996  is my favorite row of data
4936822 Task  3 line 999997  is my favorite row of data
4936823 Task  3 line 999998  is my favorite row of data
4936824 Line 1000000 - Great job! Task 3 finished OK
4936825 936824 is my favorite row of data
4936826 Task  4 line 936825 is my favorite row of data
4936827 Task  4 line 936826 is my favorite row of data
4936828 Task  4 line 936827 is my favorite row of data
4936829 Task  4 line 936828 is my favorite row of data
```

Task 4’s output had been interrupted by task 3 and now resumes.
Solution? File Locking on Linux. Just because it works on your desktop doesn’t mean it works on the cluster.

```python
#!/usr/bin/env python2.7
import struct, fcntl, os, datetime, time

# open a file
with open("lockdemo.txt","rw") as fd:
    # Attempt to exclusively lock lockdemo.txt
    print "Attempting lock at %.8s"%str(datetime.datetime.now().time())
    fcntl.flock(fd, fcntl.LOCK_EX)

    print "Acquired the lock at %.8s"%str(datetime.datetime.now().time())

lockHoldTime = 10
print "Holding the lock for %d seconds"%lockHoldTime
for i in range(lockHoldTime):
    print i
    time.sleep(1) # sleep one second

fcntl.flock(fd, fcntl.LOCK_UN)
print "Released the lock at %.8s"%str(datetime.datetime.now().time())
```
It works! The 2\textsuperscript{nd} session waits!

Session 1 – launched one second ahead of sess 2
$ date ; ./2lockdemo.py
Wed Jun 27 18:12:00 CDT 2018
Attempting lock at 18:12:00
Acquired the lock at 18:12:00
Holding the lock for 10 seconds
0
1
2
3
4
5
6
7
8
9
Released the lock at 18:12:10

Session 2
$ date ; ./2lockdemo.py
Wed Jun 27 18:12:01 CDT 2018
Attempting lock at 18:12:01
Acquired the lock at 18:12:10
Holding the lock for 10 seconds
0
1
2
3
4
5
6
7
8
9
Released the lock at 18:12:20
ACCRE cannot lock files across nodes

$ hostname; date; ./2lockdemo.py
vmps08
Wed Jun 27 18:27:20 CDT 2018
Attempting lock at 18:27:20

Acquired the lock at 18:27:20 ☺
Holding the lock for 10 seconds
0
1
2
3
4
5
6
7
8
9
Released the lock at 18:27:30

$ hostname; date; ./2lockdemo.py
vmps09
Wed Jun 27 18:27:21 CDT 2018
Attempting lock at 18:27:21

Acquired the lock at 18:27:21 😞
Holding the lock for 10 seconds
0
1
2
3
4
5
6
7
8
9
Released the lock at 18:27:31 😞
#!usr/bin/env python2.7
import MySQLdb, MySQLdb.cursors
import ConfigParser;

config = ConfigParser.SafeConfigParser()
config.read('the secret text file')

# Get the super-secret database access keys.
sqlAccess = dict(config.items("Genome_PDB_Mapper"))

connectionList = []    # An empty list of database connections

while True:     # We're gonna be here a long time! ?
    # Connect to MySQL server on other side of campus
    cn = MySQLdb.connect(host=sqlAccess['dbhost'],
                            user=sqlAccess['dbuser'],
                            passwd=sqlAccess['dbpass'],
                            db=sqlAccess['dbname'])

    connectionList.append(cn)
print len(connectionList)

If you are ever unsure, then your resource is probably much more limited than you imagine
But Chris, this would NEVER happen to me!
How might you launch and manage 1,000 jobs? 10,000 jobs?

<table>
<thead>
<tr>
<th>Gene</th>
<th>mRNA transcript</th>
<th>protein</th>
<th>mRNA Isoform-specific Refseq ID</th>
<th>mutation</th>
<th>Uniprot Protein ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 GBA</td>
<td>NM_001005741</td>
<td>R502C</td>
<td>P04062-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 KCNRG</td>
<td>NM_173605</td>
<td>R33C</td>
<td>Q8N5I3-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 KIF5C</td>
<td>NM_004522</td>
<td>N855I</td>
<td>O60282-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 UTP14C</td>
<td>NM_021645</td>
<td>I187N</td>
<td>Q5TAP6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) Login to accre, prepare your clinic .csv file… Then..
2) Launch a master program which launches (shells):
   1 monitor process for each mutation which:
   - Launches all 50 jobs for each mutation &
   - Runs “scontrol” on all the jobs until exit() codes are returned from SLURM
Create .pdf reports when everything is finished.
Design Constraint: While any ACCRE file can be safely read by infinitely many processes, it must only be written by one process at a time.

Solution: The user runs four “ordinary” .py scripts:

1. **psb_plan.py**
   - ✓ Input the short mutation .csv and plan the work.
   - ✓ Output “workplan.csv” file (for each mutation)

2. **psb_launch.py** (must run on ACCRE)
   - ✓ Reads “workplan.csv”
   - ✓ Groups alike jobs and writes SLURM array scripts for each job group
   - ✓ Exeys “sbatch” to launch each SLURM script
   - ✓ Records new job IDs in “workstatus.csv” (if/when sbatch returns them)

2.5 User takes break: Attends a better ACCRE Talk

3. **psb_monitor.py** (User runs anytime she wants to check on things)
   - ✓ Read “workstatus.csv” to learn about running jobs.
   - ✓ For each of the 100s of jobs (which are reporting their progress to disk):
     - Read their 4 job status files in job-unique-directories: progress, info, failure, complete
   - ✓ Report state of incomplete jobs to user (on-screen)
   - ✓ Records updates back to “workstatus.csv”
   - ✓ Completed and Failed jobs are not interrogated in following runs.
   - ✓ No calls to ACCRE (scontrol, etc)

4. **psb_rep.py** (Final Report module)
   - ✓ Read “workplan.csv”, bark about incomplete jobs.
   - ✓ Generate final .html reports with whatever results have come in..
   - ✓ Optionally: Create slum script to create all reports.
Pipeline users must first adopt a “pristine” shell environment.

```bash
~$ source /dors/capra_lab/users/psbadmin/psb_prep.bash
Preparing PDB Pipeline PATH and environment settings
This file should only be sourced from a bash or sh shell

Prepending to PATH:
  /dors/capra_lab/users/psbadmin/bin
  /dors/capra_lab/users/psbadmin/pdbmap
  /dors/capra_lab/users/psbadmin/pathprox
  /dors/capra_lab/opt/ensembl-tools-release-87/scripts/variant_effect_predictor
  /dors/capra_lab/opt/ensembl-tools/release-87/scripts/id_history_converter
  /dors/capra_lab/bin/vcftools/bin
  /dors/capra_lab/bin/vcftools/perl

Perl 5 detected
Prepending to PERL5LIB:
  /dors/capra_lab/opt/vcftools_0.1.12b/perl
  /dors/capra_lab/opt/bioperl-live
  /dors/capra_lab/opt/src/ensembl/modules
  /dors/capra_lab/opt/src/ensembl-compara/modules
  /dors/capra_lab/opt/src/ensembl-variation/modules
  /dors/capra_lab/opt/src/ensembl-funcgen/modules

Prepending to PYTHONPATH:
  /dors/capra_lab/users/psbadmin/pdbmap

PSB Pipeline  host:vmps11  user:mothcw
~$
```
For each mutation, psb_plan.py create a “workplan.csv” file, a self-contained listing of:
1) A unique key for each job
2) The $ command needed and options: command line, config files, working directory, output directory, and more.

2 of 4: Read workplan.csv and launch the processes listed therein by

1) Grouping like processes
2) Creating .slurm files for sets of like processes
3) Submitting (sbatch) the slurm jobs
4) Recording job IDs (where sbatch returns them) in workstatus.csv

The best code is the code that _writes_ code →
#SBATCH output=../../../../stdout/ANKRD11_NM_001256183_T269K_%A_%a.out
#SBATCH --array=0-23

```
#SBATCH

echo "SLURM_ARRAY_TASKID="$SLURM_ARRAY_TASKID

echo "SLURM_JOBID="$SLURM_JOBID

echo "SLURM_JOB_NODELIST"="$SLURM_JOB_NODELIST"

echo "SLURM_NNODES"="$SLURM_NNODES"

# echo "SLURMTMPDIR=""$SLURMTMPDIR"

echo "SLURM_SUBMIT_DIR = ""$SLURM_SUBMIT_DIR"

source /dors/capra_lab/users/psbadmin/psb_prep.bash
cd /dors/capra_lab/projects/psb_collab/UDN/UDN664435
if [ $? != 0 ]; then
echo Failure at script launch: Unable to change to directory
/dors/capra_lab/projects/psb_collab/UDN/UDN664435
exit 1
fi

case $SLURM_ARRAY_TASK_ID in
0)
    pathprox2.py -c /dors/capra_lab/users/psbadmin/config/global.config -u .. /sheehajh.config
    ENSP00000367581.2_1 NM_001256183 T269K --chain=A --add_cosmic --add_exac --radius=D --
    sqlcache=/dors/capra_lab/projects/psb_collab/UDN/UDN664435/ANKRD11_NM_001256183_T269K/sqlcache --
    overwrite --outdir
    /dors/capra_lab/projects/psb_collab/UDN/UDN664435/ANKRD11_NM_001256183_T269K/ENSP00000367581.2_1_A_
    PathProxCOSMIC --uniquekey ANKRD11_NM_001256183_T269K_ENSP00000367581.2_1_A_PathProxCOSMIC
    ;;

1)
    pathprox2.py -c /dors/capra_lab/users/psbadmin/config/global.config -u .. /sheehajh.config
    ENSP00000367581.2_1 NM_001256183 T269K --chain=A --add_pathogenic --add_exac --radius=D --
    sqlcache=/dors/capra_lab/projects/psb_collab/UDN/UDN664435/ANKRD11_NM_001256183_T269K/sqlcache --
    overwrite --outdir
    /dors/capra_lab/projects/psb_collab/UDN/UDN664435/ANKRD11_NM_001256183_T269K/ENSP00000367581.2_1_A_
    PathProxClinvar --uniquekey ANKRD11_NM_001256183_T269K_ENSP00000367581.2_1_A_PathProxClinvar
    ;;
```
Monitoring 1000 jobs

/dors/capra_lab/users/mothcw/accretalk$ sbatch arraydemo1.slurm
sbatch: error: slurm_receive_msg: Socket timed out on send/recv operation
sbatch: error: Batch job submission failed: Socket timed out on send/recv operation

... If you didn't get back a job number on stdout???

/dors/capra_lab/users/mothcw/accretalk$ sq

<table>
<thead>
<tr>
<th>JOBID</th>
<th>NAME</th>
<th>ST</th>
<th>TIME</th>
<th>TIME_LIMIT</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>28198084</td>
<td>IntersectModbaseExac</td>
<td>PE</td>
<td>0:00 2-00:00:00</td>
<td>(Priority)</td>
<td></td>
</tr>
<tr>
<td>28198221</td>
<td>[0- arraydemo1.slurm</td>
<td>PE</td>
<td>0:00</td>
<td>1:00</td>
<td>(Priority)</td>
</tr>
</tbody>
</table>
All 1,000 jobs have unique status directories. (info/progress/complete/fail)
Pipeline monitor for launched jobs. -h for detailed help.

Monitoring all jobs for 17 mutations
1 of 17: PLXND1    NM_015103.2 K694N
Recording all updates to /dors/capra_lab/users/psbadmin/bin/psb_monitor.py

5 of 7 jobs still incomplete:

Jobid:Flavor       Info
28220566:PLXND1_NM_015103.2_K694N_ENSP00000317128.4_3_A_PathProxCOSMIC   Configured
28220566:PLXND1_NM_015103.2_K694N_ENSP00000317128.4_3_A_PathProxClinvar    Initialized
28220566:PLXND1_NM_015103.2_K694N_Q9Y4D7-1_48_753_4gza.1.C_A_PathProxCOSMIC   Configured
28220566:PLXND1_NM_015103.2_K694N_Q9Y4D7-1_48_753_4gza.1.C_A_PathProxClinvar    Initialized
2820568:PLXND1_NM_015103.2_K694N_SequenceAnnotation    Begun

2 of 17: ANKRD11    NM_001256183 T269K
Recording all updates to /dors/capra_lab/users/psbadmin/bin/psb_monitor.py

25 of 37 jobs still incomplete:

Jobid:Flavor       Info
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.2_1_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.2_1_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.1_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.1_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.2_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.2_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.3_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.3_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.5_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_ENSP00000367581.5_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_Q6UB99_100_296_5le9.1.A_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_Q6UB99_100_296_5le9.1.A_A_PathProxClinvar    Configured
28220569:ANKRD11_NM_001256183_T269K_Q6UB99_101_296_5jhq.1.A_A_PathProxCOSMIC   Configured
28220569:ANKRD11_NM_001256183_T269K_Q6UB99_101_296_5jhq.1.A_A_PathProxClinvar    Configured

.... and so on....
Happy Birth-usec, SLURM jobs! (Likelihood of duplicate runs with usec seeds)

Imagine 100 jobs starting together within the same second, with 1,000,000 randomly distributed microsecond-resolution start times!)

What are the chances that 2 of the 100 jobs will have the same start time?

With 100 jobs there are \( (100 \times 99 / 2) = 4,950 \) unique pairs of jobs.

The chance of any 2 jobs having different the microsecond assignment = 
\[ (1 - 1/1,000,000) = (999,999/1,000,000) \]

The odds of _ALL_ 4,950 pairs having different microsecond launch-times is:
\[ (0.999999)^{4950} = 0.9951 \]

0.9951 is about 1/200. 1 in 200 batch submissions of (100 jobs) will see a duplicate microsecond assignment :)

And, when the new Redhat7 scheduler launches them all in 1/100 of a second the picture gets 100x worse for duplicate random seeds:

\[ (99,999/100,000)^{4950} = .952 \] (1 in 20 of the 100-job batch submissions must have a duplicate random seed)