DUNE HEP-SCORE status

Andrew McNab

University of Manchester

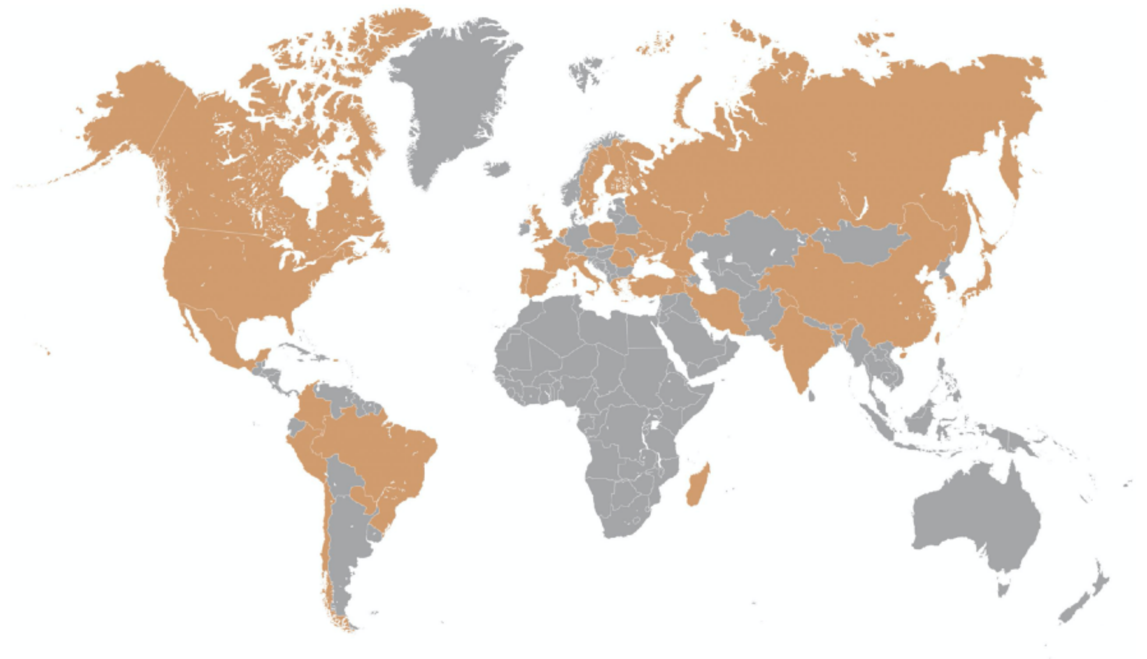
DUNE

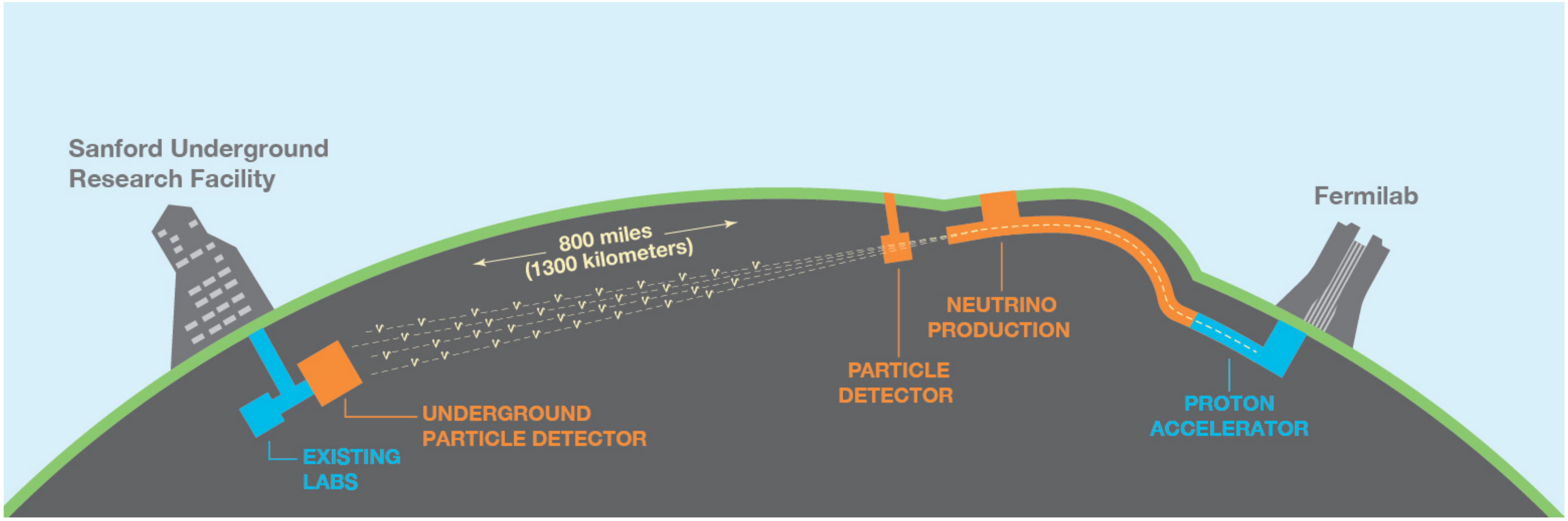
Worldwide LHC-experiment scale computing but centred on FNAL

Using WLCG experience wherever possible (eg CRIC for pledges)

protoDUNE SP+DP at CERN now

DUNE itself starts mid-decade




Sanford Underground Research Facility
800 miles (1300 kilometers)
Fermilab
NEUTRINO PRODUCTION
PARTICLE DETECTOR
PROTON ACCELERATOR
UNDERGROUND PARTICLE DETECTOR
EXISTING LABS

DUNE HEP-SCORE rough plan

Step 1: identify workloads

Step 2: produce proof-of-concept H-SC container

Step 3: finalise workloads and weighting

Step 4: publish production quality H-SC container

Identify workloads

- Within DUNE we discussed various options
  - Trivial examples from new user guides etc
  - Creating some custom recipes that are representative
  - Using recipes we already had
- Settled on the suite of Continuous Integration tests that we run every day
  - Well understood and maintained already
  - Exist for protoDUNE SP+DP and for DUNE far detector; for reconstruction, simulation, and analysis workloads
  - Depend on Geant4 and LArSoft
  - Software and data in cvmfs

Example CI workload

- This is a DUNE Far Detector reconstruction test
- Uses software from /cvmfs/dune.opensciencegrid.org, fermilab.opensciencegrid.org and larsoft.opensciencegrid.org
- Uses data from Large Scale CernVM-FS dune.osgstorage.org
- As written below, takes about a minute - to process one event

```
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh

setup dunetpc v09_13_00 -q e19:prof

lar --rethrow-all -n 1 --timing-db time.db \
       -o prodgenie_nue_dune10kt_1x2x6_reco_Current.root \
       --config ci_test_reco_dunefd.fcl \
       /cvmfs/dune.osgstorage.org/pnfs/fnal.gov/usr/dune/persistent/
stash/ContinuousIntegration/DUNEFD/detsim/
prodgenie_nue_dune10kt_1x2x6_detsim_Reference.root
```

DUNE HEP-SCORE rough plan

Step 1: identify workloads

Step 2: produce proof-of-concept H-SC container

Step 3: finalise workloads and weighting

Step 4: publish production quality H-SC container

Proof of concept container

- Want to go through all the steps to produce a DUNE HEP-SCORE container
  - Bring in all the dependencies
  - Identify any “surprises”
  - Identify any needed feature requests in the HEP-SCORE framework
  - But not polished up and complete in this first pass
- In short, put the the setup and lar commands in a dune-…-bmk.sh shell script
  - And add in the dependencies in the .spec file, Dockerfile.append etc

Proof of concept container

- This has been produced now
- Code in GitLab, in a fork of HEP-Benchmarks/hep-workloads
    - Including some fixes and hard-coded workarounds in the rest of the hep-workloads framework
- Some shortcuts for now (for example, using the dummy parseResults.sh from the HEP-SCORE CI tests)
- Needed to include cvmfs-config.cern.ch too to get config for the three OSG software cvmfs repositories mentioned before
- Results from running a copy of the container

```
cat dune-reco-fd_summary.json
{ "copies" : 2 , "threads_per_copy" : 1 , "events_per_thread" : 5 , "throughput_score" : 1 , "log":
"OK", "app" : {
  "version": "v0.1",
  "description": "DUNE Reco FD benchmark",
  "cvmfs_checksum": "9df32fdab6f7a60f861bc86f12b024f7",
  "bmkdata_checksum": "84f300e87df247972697163014808 0e0",
  "bmk_checksum": "0db7e39fa193ee63b547a25ff8347990",
  "containment": "docker"
} }
```

Issues identified

- v09 of the DUNE TPC package requires CentOS 7
  - This is currently not working in the HEP-SCORE framework
  - We just forced it to cc7:latest in Dockerfile.header (thanks to Domenico for advice)
- Hard-coded assumptions that all cvmfs repos end in .cern.ch
  - Like `sed -e 's@cvmfs-\([^\.]*\)\.cern\.ch.*@\1@'` for files like `cvmfs-dune.opensciencegrid.org.trace.log`
  - Changed to `sed -e 's@cvmfs-\(.*\)\.trace\.log.*@\1@'`
- CernVM-FS shrink wrap failed for dune.osgstorage.org data repository
  - Not set up on the OSG cvmfs server instance for shrink wrap?
  - Put the data file into git for now (30Mbytes)

DUNE HEP-SCORE rough plan

Step 1: identify workloads

Step 2: produce proof-of-concept H-SC container

Step 3: finalise workloads and weighting

Step 4: publish production quality H-SC container

Next steps

- Go back to DUNE software experts with a working example to show
  - Agree which subset of Continuous Integration tests to use in the benchmark
  - What parameters for each?
  - What weight to give each one if we choose to use more than one in the final published DUNE HEP-SCORE
  - What would we like to do as DUNE software evolves rapidly to the mid-decade?
- Help integrate the fixes for the issues identified already in the HEP-SCORE hep-workloads framework
- Produce a production quality DUNE HEP-SCORE

Summary

- Working through our plan to produce a HEP-SCORE benchmark
- Have identified a representative set of DUNE workloads to draw on (the DUNE CI tests)
- Have produced a proof-of-concept DUNE HEP-SCORE container, with the necessary dependencies
- This has identified some (easy to fix) issues with current HEP-SCORE hep-workloads framework
- Next step is to go back to DUNE software experts to finalise workloads and parameters to use