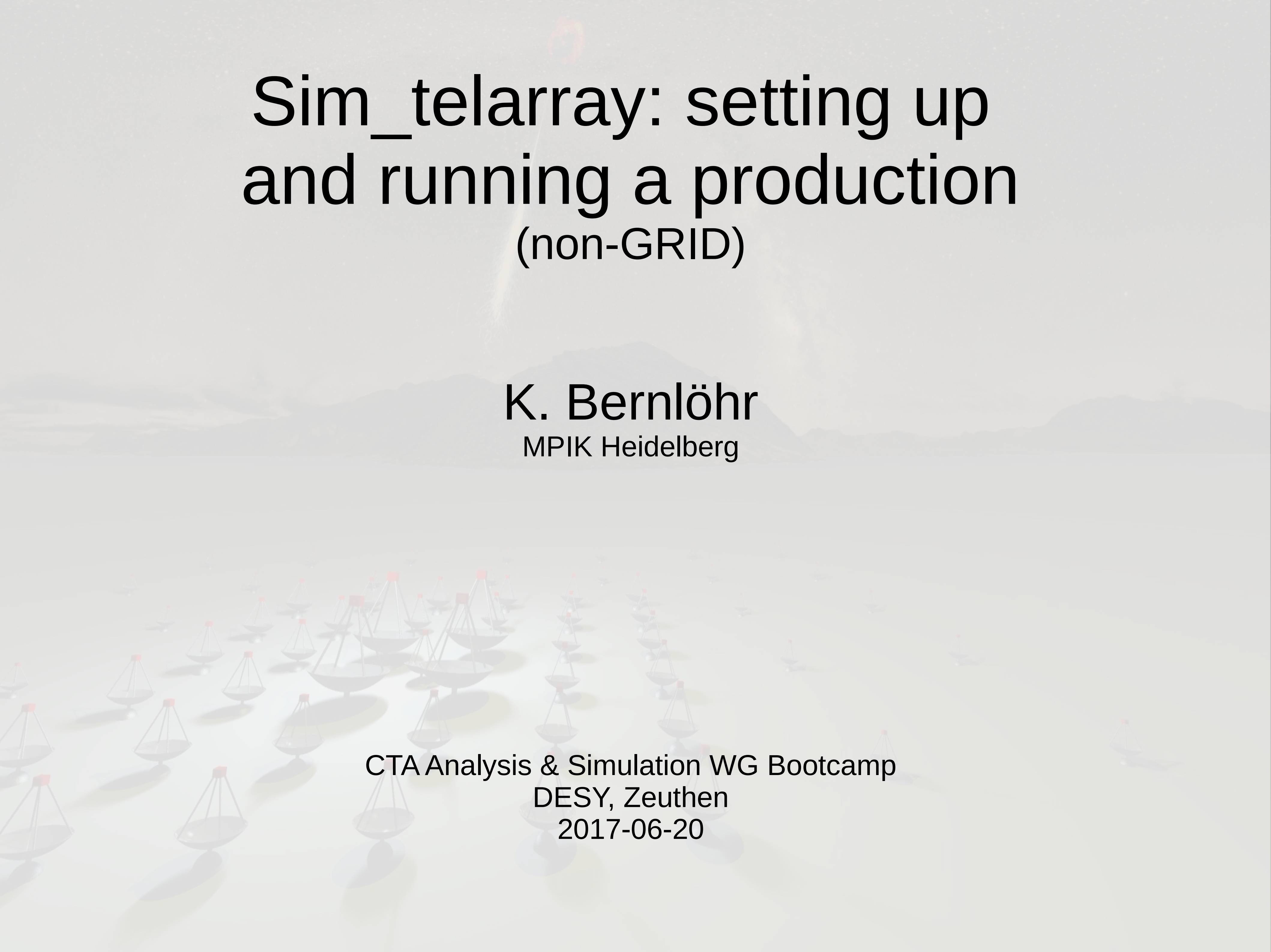


Sim_telarray: setting up and running a production

(non-GRID)

K. Bernlöhr

MPIK Heidelberg

CTA Analysis & Simulation WG Bootcamp
DESY, Zeuthen
2017-06-20

Before we start

- I hope you downloaded, as instructed yesterday, corsika7.5_simtelarray.tar.gz, unpacked it, and installed it with:

```
./build_all baseline qgs2
```

- Produce a CORSIKA file with

```
echo ‘gzip > ${CORSIKA_DATA}/run${CORSIKA_RUN}.corsika.gz’ \
>> sim_telarray/multi/multi_cta-prod3-paranal-baseline.cfg
NSHOW=10 EMIN=0.1 EMAX=3 CSCAT=400 NSCAT=10 \
./prod3_paranal_baseline_run
```

- Move the CORSIKA file for later use:

```
mv Data/corsika/run*.corsika.gz std/sim_telarray/x.corsika.gz
```

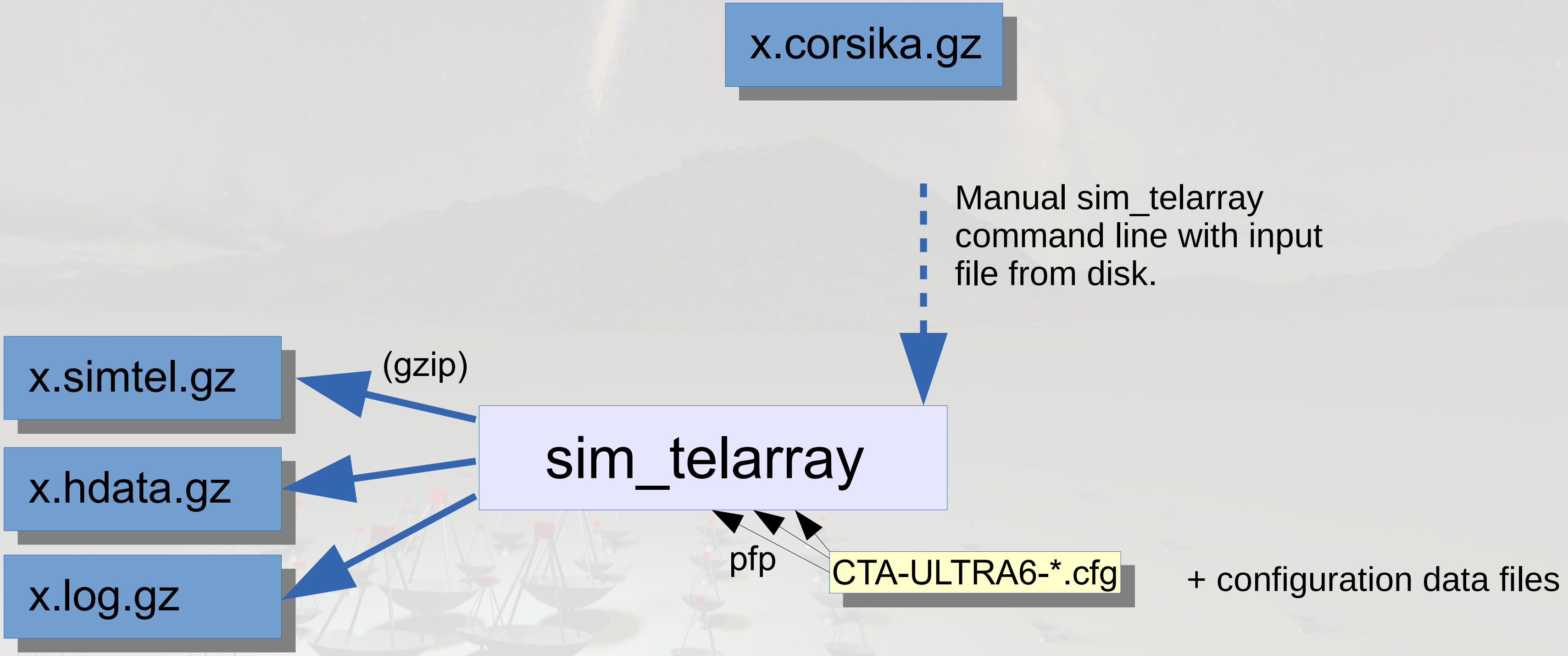
- cd std/sim_telarray

Not discussed here

- The largest effort in setting up an entirely new production (e.g. prod-2 -> prod-3) are new telescope configurations, including
  - optics definitions
  - all sorts of optical and sensor efficiencies
  - pulse shapes, amplitudes, noise, ...

  and in particular the evaluation of the best trigger settings (e.g. which threshold for which combination of pulse shaping and clipping results in the lowest energy threshold for gamma rays?)

Running sim_telarray by itself


x.corsika.gz
Manual sim_telarray command line with input file from disk.
x.simtel.gz
(gzip)
x.hdata.gz
x.log.gz
sim_telarray
pfp
CTA-ULTRA6-*.cfg
+ configuration data files

Pointing direction, site altitude, file names etc. are needed on command line and should match the CORSIKA (or LightEmission) settings.
Log file (stdout/stderr output) by default just on screen.
Perhaps need to set PATH, LD_LIBRARY_PATH, SIMTEL_CONFIG_PATH environment variables beforehand.

Example command line

Example produced with prod3_paranal_baseline_run:

```
./bin/sim_telarray -c cfg/CTA/CTA-ULTRA6-demo.cfg \
   -DFLASHCAM -DGCTS -DNUM_TELESCOPES=99 -Icfg/CTA \
   -C Altitude=2150 -C iobuf_maximum=1000000000 \
   -C maximum_telescopes=99 \
   -C atmospheric_transmission=atm_trans_2150_1_10_0_0_2150.dat \
   -C telescope_theta=20.0 -C telescope_phi=180 \
   -C power_law=2.50 \
   -C histogram_file=/home/konrad/Test-nb-061/pa-
baseline/Data/sim_telarray/cta-prod3-
demo/0.0deg/Histograms/gamma_20deg_180deg_run1___cta-prod3-
demo_desert-2150m-Paranal-baseline.hdata.gz \
   -C output_file=/home/konrad/Test-nb-061/pa-
baseline/Data/sim_telarray/cta-prod3-
demo/0.0deg/Data/gamma_20deg_180deg_run1___cta-prod3-demo_desert-
2150m-Paranal-baseline.simtel.gz \
   -C random_state=auto \
   -c cfg/CTA/CTA-ULTRA6-Paranal-baseline.cfg \
   -C show=all -
```

Override initial configuration file

Example reads in pipe from standard input

Example command line

We adapt that into:

```
./bin/sim_telarray \
   -DFLASHCAM -DGCTS -DNUM_TELESCOPES=99 -Icfg/CTA \
   -C Altitude=2150 -C iobuf_maximum=1000000000 \
   -C maximum_telescopes=99 \
   -C atmospheric_transmission=atm_trans_2150_1_10_0_0_2150.dat \
   -C telescope_theta=20.0 -C telescope_phi=180 \
   -C power_law=2.50 \
   -C histogram_file=x.hdata.gz \
   -C output_file=x.simtel.gz \
   -C random_state=auto \
   -c cfg/CTA/CTA-ULTRA6-Paranal-baseline.cfg \
   -C show=all x.corsika.gz
```

A little more automatic

- The “generic_run.sh” script knows all past production configurations and many tests.
- You need to pass it a configuration name (in case of symlink deduced from name), a few extra hints, and the input file name, e.g.:

```
cfg=cta-prod3-demo \
 extra_defs=’-DFLASHCAM -DGCTS’ \
 extra_config=’-c cfg/CTA/CTA-ULTRA6-Paranal-baseline.cfg’ \
 extra_suffix=’-2150m-Paranal-baseline-test’ \
 ./generic_run.sh x.corsika.gz
```

- Output still goes to directory `Data/sim_telarray/cta-ultra6/0.0deg/Data` and you find two files there (with/without “-test”).

A little more automatic

- The generic_run.sh script relies on CORSIKA_* env. variables for part of setup. Normally, they are set beforehand – but with input from a file, they are internally set from data in the file with

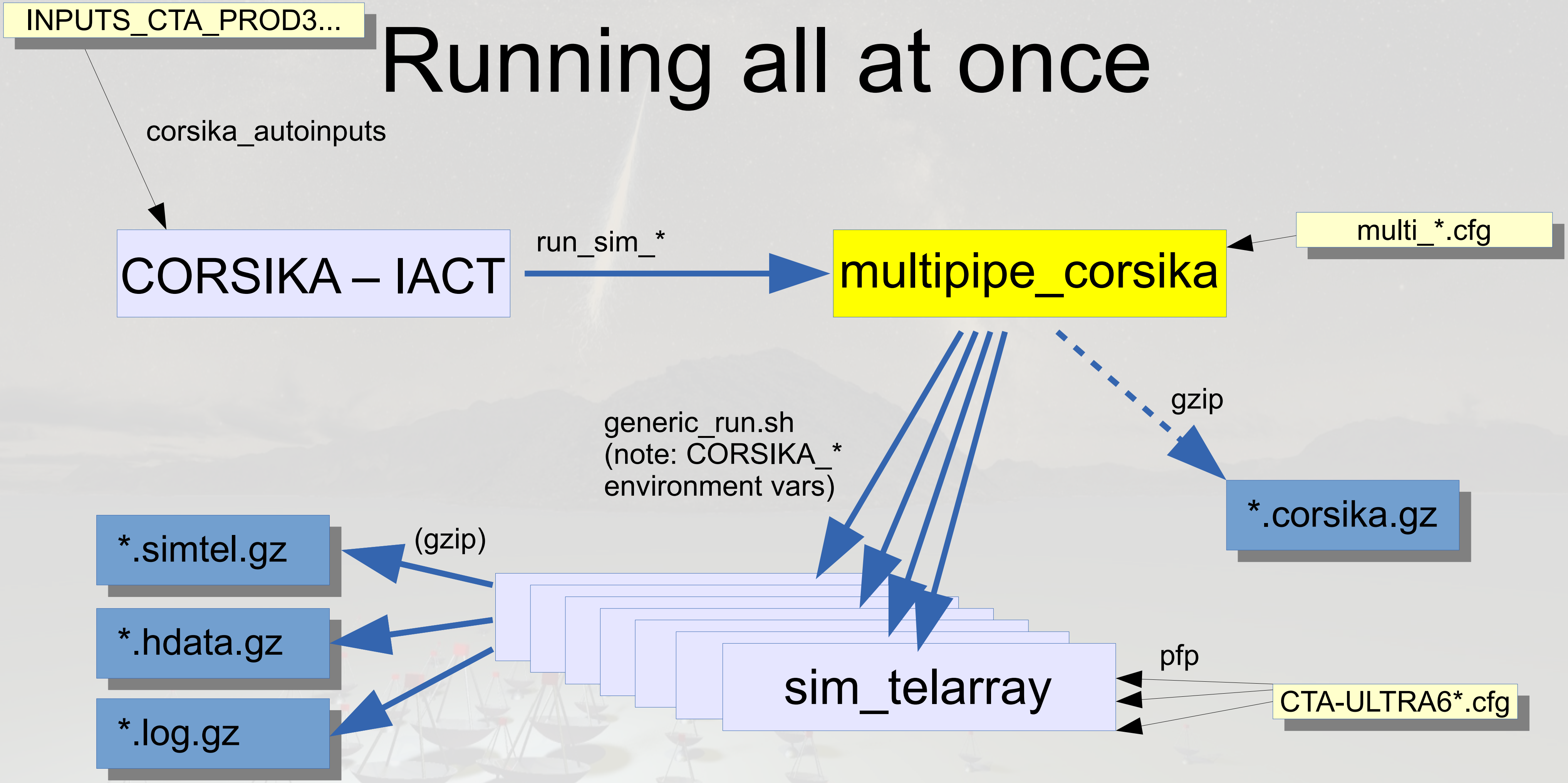
  ```
  bin/extract_corsika_tel --header-only --only-telescopes 1-999 x.corsika.gz
  ```

- But where does the second simtel.gz file (the one without “-test”) come from??

Complete production scripts

- The ‘build_all’ unpacks quite a number of production scripts – suitable for local clusters / batch systems only (MPIK: SGE6), not for GRID.
- You find them in directory example_scripts.
- Symlinks only for most relevant ones, e.g.
  prod3_run, prod3b_run, prod3_lapalma[3]_run,
  prod3_muon_run, prod3_paranal_baseline_run, ...
- Using prod3_paranal_baseline_run as example.
  - One of the simpler productions.
  - Prod3 & prod3b had intermediate CORSIKA output and had to merge sim_telarray outputs.

Running all at once



Failure of individual pipes are tolerated, as long as one pipe remains.
Programs are finished when no pipe is left.
To have processing in more sequential order, set CORSIKA_MULTIPIPE_SEQUENTIAL environment variable.

Top-level production script

For example protons, z=20° North with Paranal baseline:

```
#!/bin/bash

if [ -x ${HOME}/bin/init_batch_job ]; then
   . ${HOME}/bin/init_batch_job
fi

prod3site="Paranal"
if [ ! -z "${PROD3_SITE}" ]; then
   prod3site="${PROD3_SITE}"
else
   export PROD3_SITE="$prod3site"
fi

# Syntax: prod3_paranal_baseline_run \
#           site primary from_direction zenith_angle

export NSHOW=50000
./prod3_paranal_baseline_run "$prod3site" proton North 20
```

Production script in more detail

What is happening under the `prod3_paranal_baseline_run` production script?

- Set up some environment variables.
- Pre-process the CORSIKA inputs template with the particle type, etc.
- corsika_autoinputs books a new run, creates a working directory for it with symlinks to necessary files etc., final touches to CORSIKA inputs (RUN, SEED, version dependent fixes),
  - then starts CORSIKA in working directory.

Production script in more detail

What is happening with the CORSIKA output?
(See corsika-run/INPUTS_CTA_PROD3-Paranal-baseline-template-20deg)

- IACT interface output with TELFIL parameter:
  TELFIL |${SIM_TELARRAY_PATH}/run_sim_cta-prod3-paranal-baseline
  - Note: output to pipe (‘|’)
  - Location of script set by environment variable
  - Given script is symlink to (historical name ...)
    run_sim_hessarray_generic
  - Script takes the ‘cta-prod3-paranal-baseline’ part of the script name to start
    bin/multipipe_corsika -c multi/multi_cta-prod3-paranal-baseline.cfg

Production script in more detail

What does multipipe_corsika do now?

- Looks into first four data blocks (run header, copy of inputs file, telescope positions, first event header) to set up some CORSIKA_... env. vars.
- Every non-empty, non-comment line in the given configuration file is a command for popen(). Open them all.
- Write the preserved first four blocks to output(s).
- All remaining data copied to output as-is.
- Output pipes can work in parallel or sequentially.
- Continue until end of data or all pipes failed.

Production script in more detail

What is in our multipipe_corsika config here?
(std/sim_telarray/multi/multi_cta-prod3-paranal-baseline.cfg)

- Comment:
  ```
  # Simulation with Paranal baseline layout (3HB9: 4 LSTs, 25 MSTs, ...
  ```
- Command for generic_run.sh to sim_telarray:
  ```
  env offset="0.0" cfg="cta-prod3-demo" extra_defs="-DFLASHCAM -DGCTS" extra_config="-c cfg/CTA/CTA-ULTRA6-Paranal-baseline.cfg" extra_suffix="-${CORSIKA_OBSLEV}m-${PROD3_SITE}-baseline" ./switch_sim.sh pa-baseline ./generic_run.sh
  ```
- Appended: additional copy of output:
  ```
  gzip > ${CORSIKA_DATA}/run${CORSIKA_RUN}.corsika.gz
  ```
- You can run as many output pipes in parallel as your main memory/batch system/admin allows.

Production script in more detail

And what does the ‘-DFLASHCAM -DGCTS’ do?
(std/sim_telarray/cfg/CTA/CTA-ULTRA6-Paranal-baseline.cfg)

- Actually nothing (default anyway) but ...

```
...
#elif TELESCOPE < 30
% Default MST type here is FlashCam.
# if NECTARCAM != 0
#  include <CTA-ULTRA6-MST-NectarCam.cfg>
# elif SCT != 0
#  include <CTA-ULTRA6-SCT.cfg>
# else
#  include <CTA-ULTRA6-MST-FlashCam.cfg>
# endif
...
```

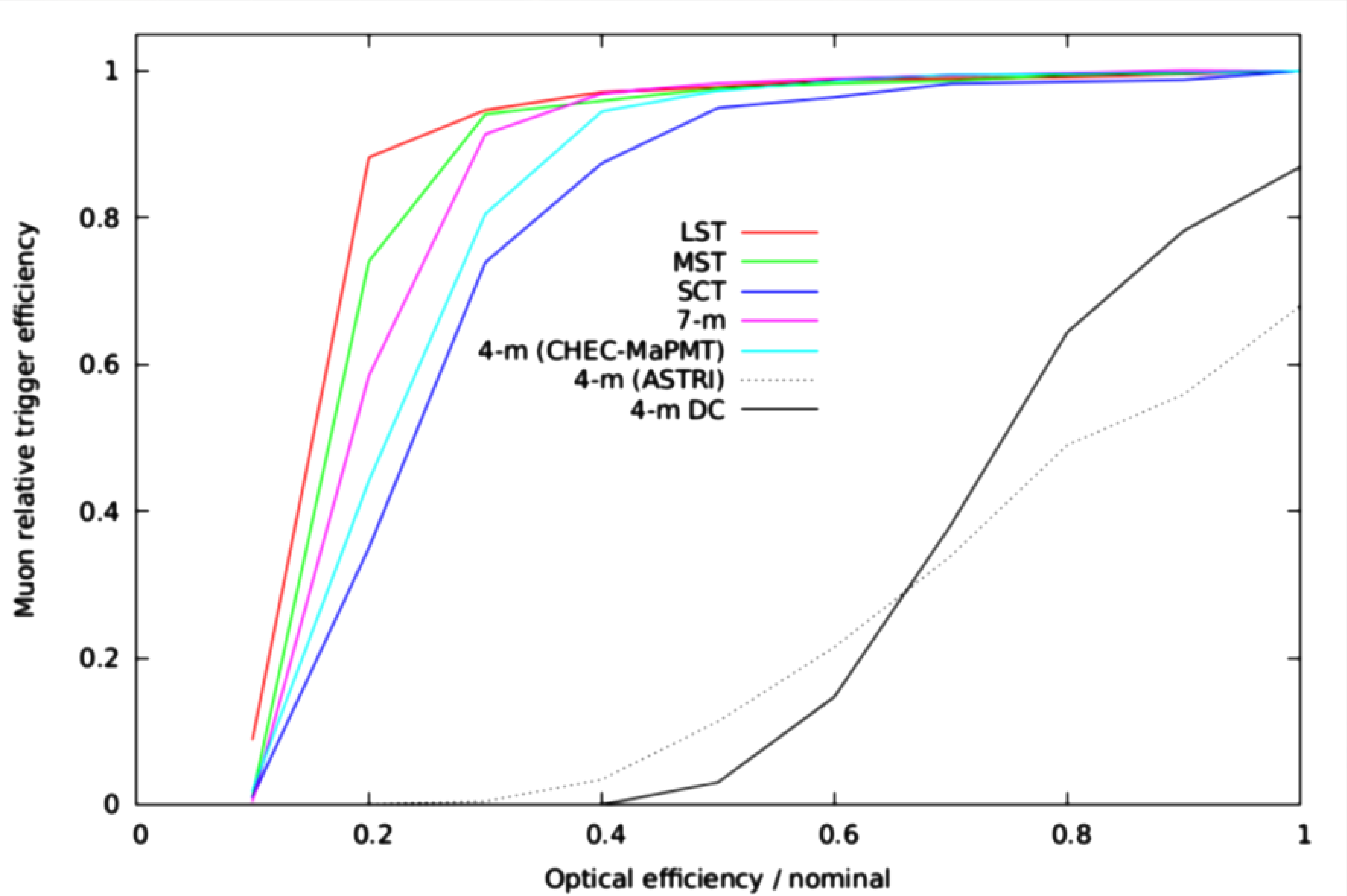
- Similar for SSTs ...

Other types of productions

- Muon ring trigger efficiency (for diff. opt. eff.):
  - prod3_muon_run [LST | MST | SST] site
  - For different optical efficiencies modify std/sim_telarray/multi/multi_cta-prod3-{l,m,s}st.cfg and add extra lines like:

    ```
    env offset="0.0" cfg="cta-prod3-sst-dc" extra_defs="-C
    MIRROR_DEGRADED_REFLECTION=0.8" extra_suffix="-$
    {CORSIKA_OBSLEV}m-${PROD3_SITE}-sst-dc-0.8"
    ./generic_run.sh
    ```
  - Just counting the lines with “has triggered” in the resulting log files is the simplest analysis for that.

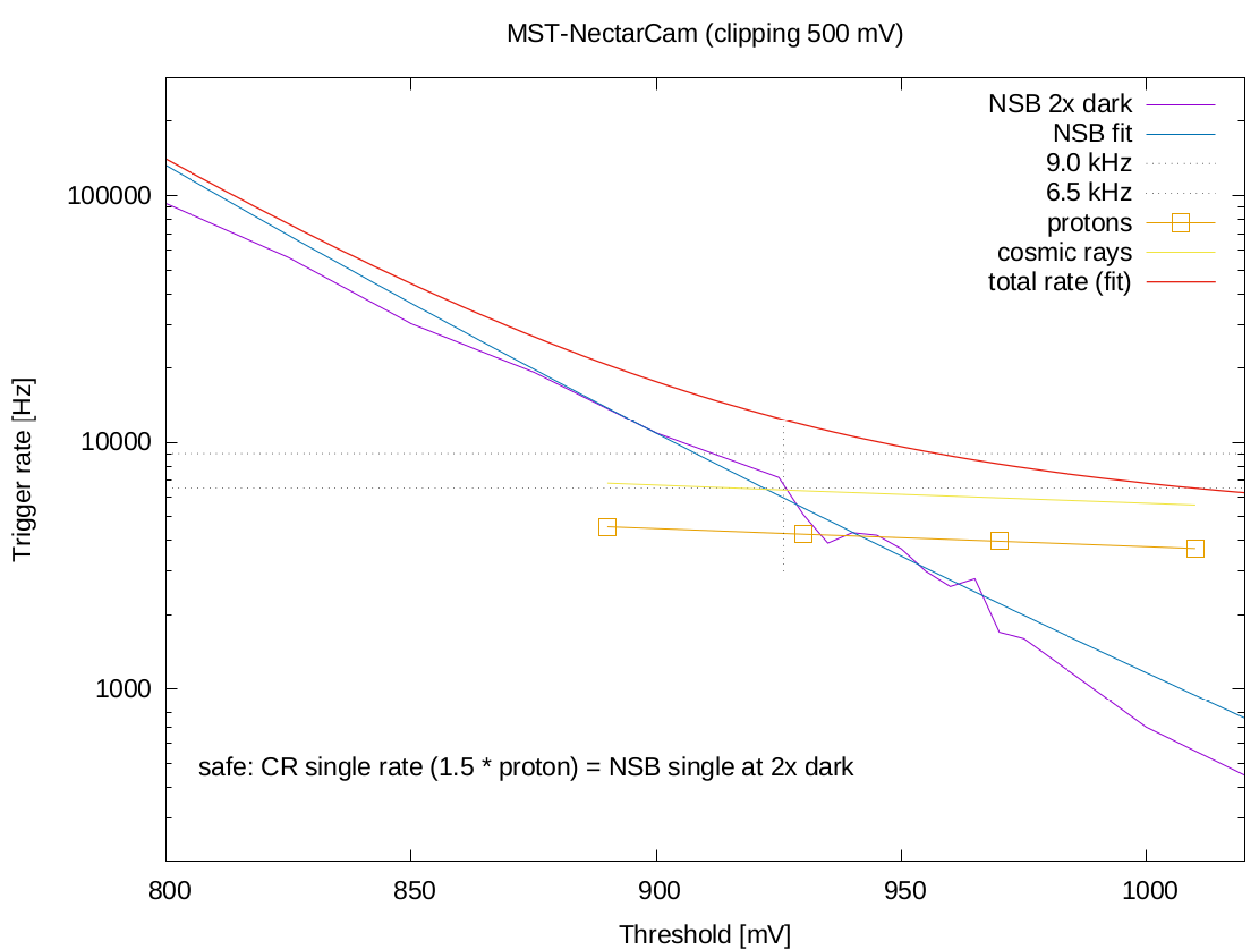
Muon trigger efficiencies (prod-2)



see 2014-01-21 CCF telecon

Other types of productions

- Simulations of NSB-only triggers (needs a CORSIKA output without actual Cherenkov light, e.g. dummy100000.corsika.gz)
  - See also 2013 Tutorial 1
  - Newer scripts see prod3-random-scripts.tar.gz
  - Takes a few CPU hours for each telescope type / NSB level / threshold / clipping / ...
  - Later to be complemented with proton simulations for a few thresholds (50-100 runs each) in the expected range for each telescope type / NSB level / clipping / ...
  - Evaluate safe / aggressive thresholds from those.


MST-NectarCam (clipping 500 mV)
NSB 2x dark
NSB fit
9.0 kHz
6.5 kHz
protons
cosmic rays
total rate (fit)
Trigger rate [Hz]
100000
10000
1000
800
850
900
950
1000
Threshold [mV]
safe: CR single rate (1.5 * proton) = NSB single at 2x dark

Conclusions

- I tried to illustrate the whole chain of scripts involved in a straight-forward production (Paranal baseline layout).
- Top-level scripts for other primary particles and other directions should be easy (change one line).
- The prod3/prob3b production scripts for Paranal were more tricky, with splitting-up CORSIKA output, separate simulations, and later merging the sim_telarray output.
- Showed two cases of auxiliary productions.