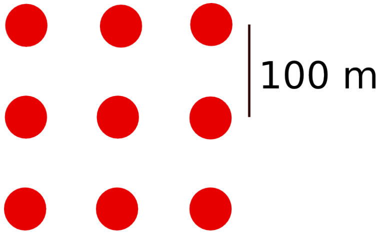
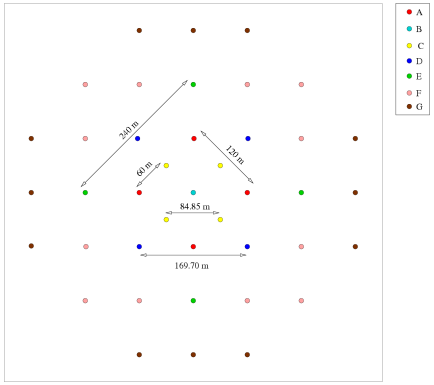
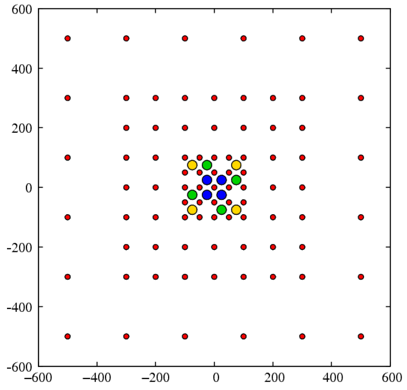
Notes on the eventio format and the `CORSIKA / sim_telarray` tool chain in CTA simulations

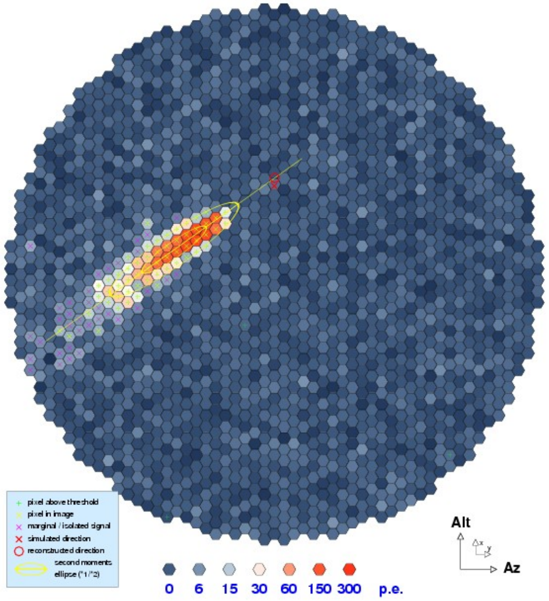






Konrad Bernlöhr

MPIK Heidelberg & HU Berlin



CTA-MC meeting, Paris, 2008-03-27

Outline

- Introduction: CORSIKA / sim_telarray tool chain.
- Data format.
- Running CORSIKA and/or sim_telarray.
- Keeping track of things: database tools.
- Analysis

The CORSIKA / sim_telarray tool chain

- Shower simulation with **CORSIKA**.
  - Using the IACT option for machine-independent output based on the `eventio` library.
  - Telescopes are just defined there as fiducial spheres.
  - Output may be saved to disk or piped directly into one or multiple telescope simulations (which has basically no I/O bottlenecks).

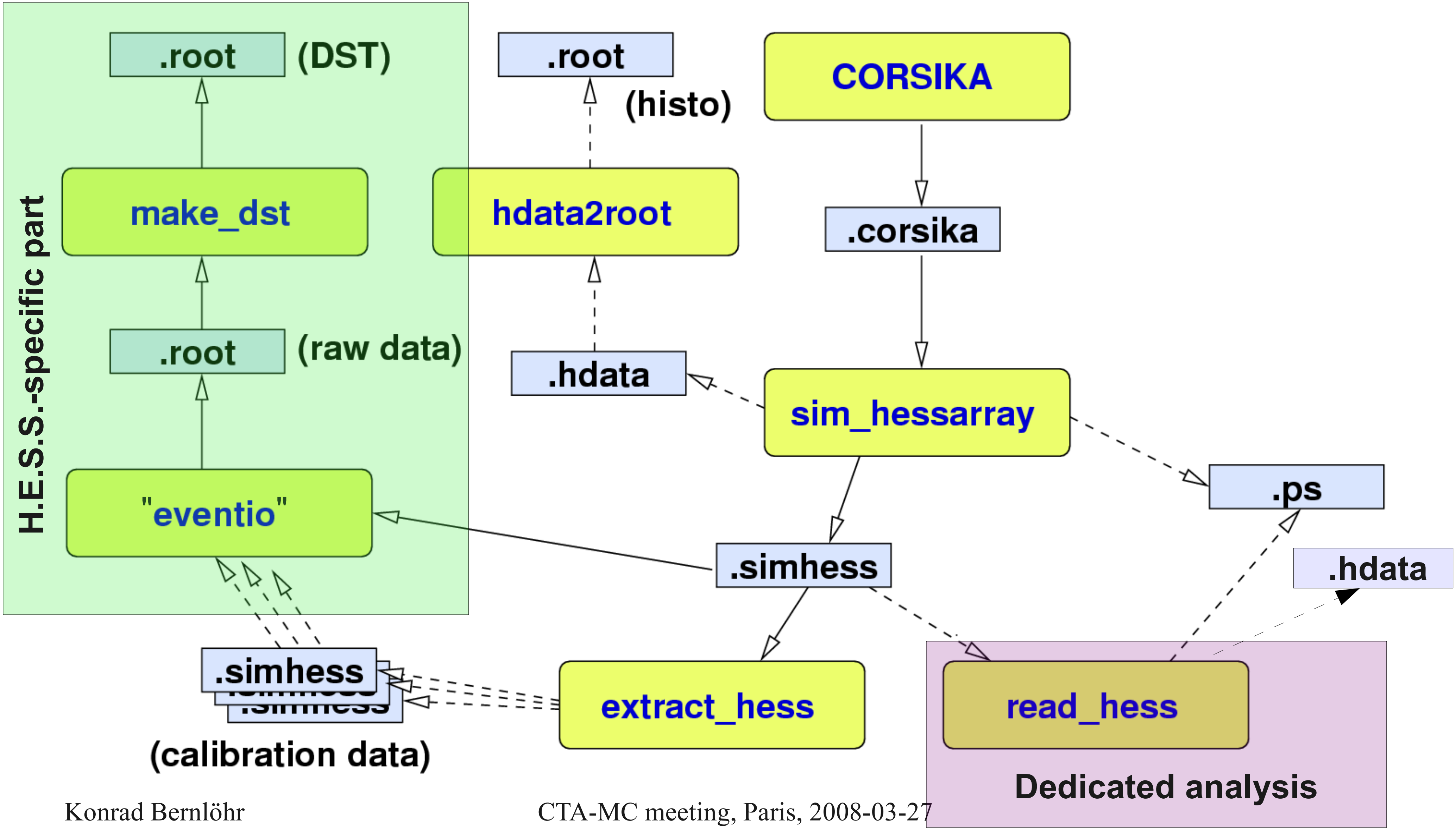
The CORSIKA / sim_telarray tool chain

- Telescope simulation with **sim_telarray** (also called sim_hessarray).
- Developed for and tested on HEGRA and H.E.S.S.
- Includes ray-tracing, night-sky background (+ stars), all sorts of electronic pulse shapes, switching behaviour of discriminators or comparators, ...
- Pixel read-out with one or two gains, ...
- Fully configurable at run-time.

The CORSIKA / sim_telarray tool chain

- **sim_telarray** output also eventio-based:
  - Including raw data integrated or in sample mode.
  - With second moments based analysis results from integrated reconstruction or external reconstruction program.
  - Histograms can be converted to ROOT (or PAW).
- Data conversion tools available (e.g. -> H.E.S.S.).
- Dedicated analysis program for quick development.

The CORSIKA / sim_hessarray processing pipeline



The underlying eventio format

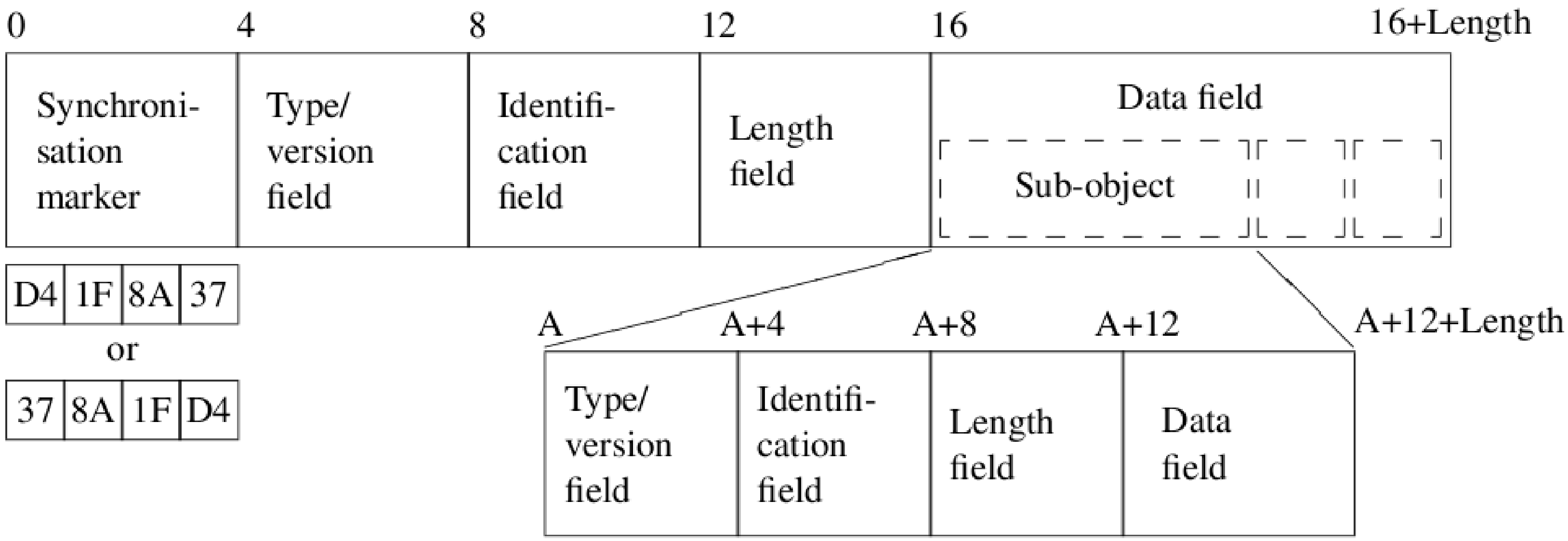
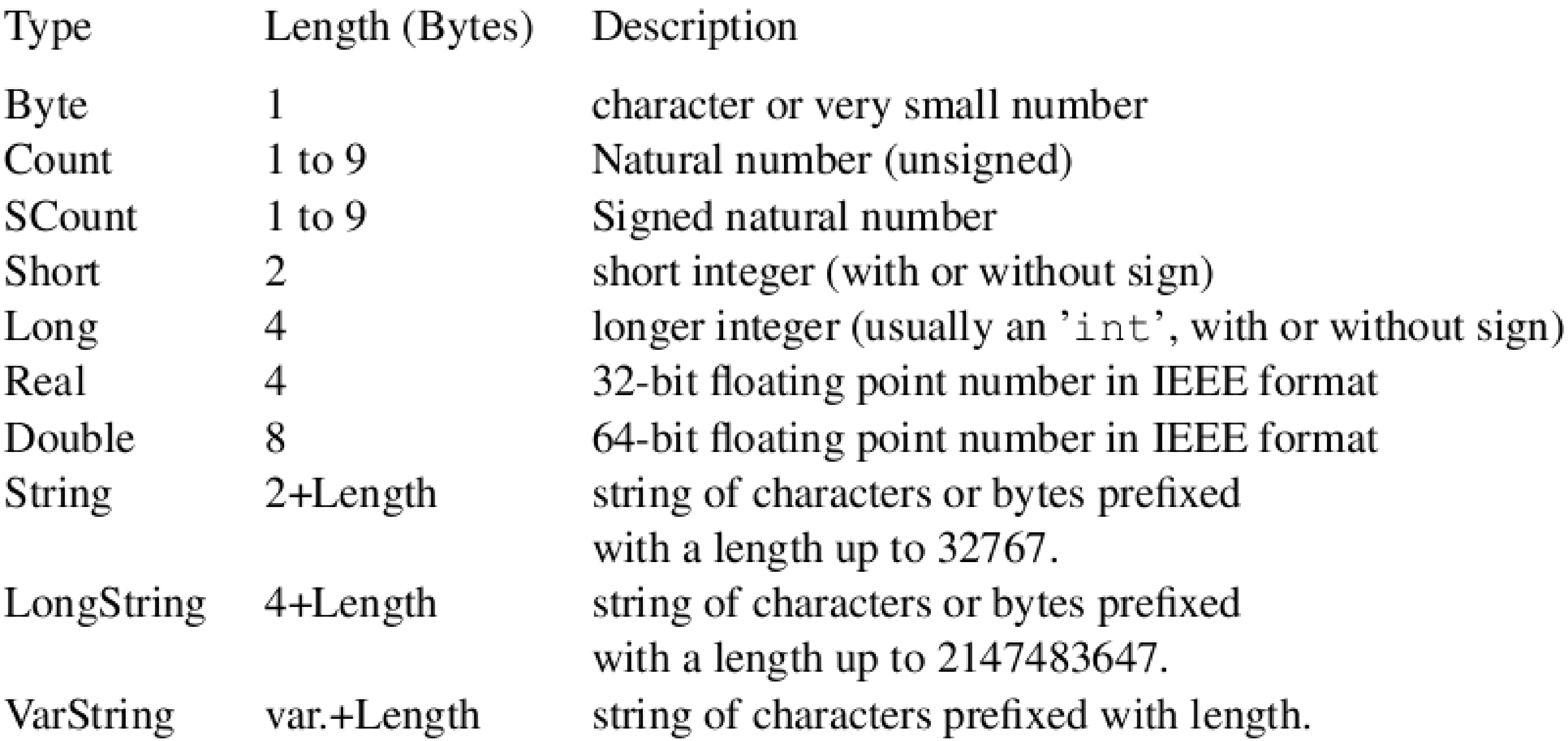


Figure 1: Schematic structure of a top-level object and a sub-object

Data format is machine-independent and hierarchical.
For each object/data type, a pair of read/write functions is needed (a bit like ROOT custom streamers).
Access is sequential. File I/O with automatic (de)compression.

Elementary data types with eventio



Type	Length (Bytes)	Description
Byte	1	character or very small number
Count	1 to 9	Natural number (unsigned)
SCount	1 to 9	Signed natural number
Short	2	short integer (with or without sign)
Long	4	longer integer (usually an 'int', with or without sign)
Real	4	32-bit floating point number in IEEE format
Double	8	64-bit floating point number in IEEE format
String	2+Length	string of characters or bytes prefixed with a length up to 32767.
LongString	4+Length	string of characters or bytes prefixed with a length up to 2147483647.
VarString	var.+Length	string of characters prefixed with length.

Data from CORSIKA IACT interface

- Full inputs file (control 'cards').
- Telescope definitions and event-by-event offsets.
- CORSIKA run+event headers+trailers.
- Vertical/longitudinal shower profiles (part.,e,μ,Ch.l.)
- Photon or photo-electron bunches of a full telescope array at a time (either 16 or 32 bytes per bunch). Fixed bunch size, but with excessive amount of light in a telescope an automatic thinning sets in. Arrival times always in telescope mid-plane.

Data in sim_telarray output file (current hessio production format)

- Run header / MC run header / var. telescope conf.
- History incl. CORSIKA inputs, command line and configuration used.
- Calibration data (optionally: calibration events)
- MC showers (incl. profiles) / MC events.
- Events: 'raw' data (sample mode or integrated), Hillas parameters, reconstructed showers.
- Statistics histograms.

sim_telarray raw data and its limitations

- Can hold sample-mode (typically GHz) or integrated signal (ADC sum, typically over 16 ns).
- One or two gains per pixel.
- Zero suppression and suppression of low-gain channel are possible but rarely used.
- Currently limited to 4095 pixels in hessio format.
- Only ADC counts, no h/w or pulse shape timing.
- For saving main memory with more or less static struct layout, limits set in io_hess.h header file.

Lifting the limitations ... (or: the future)

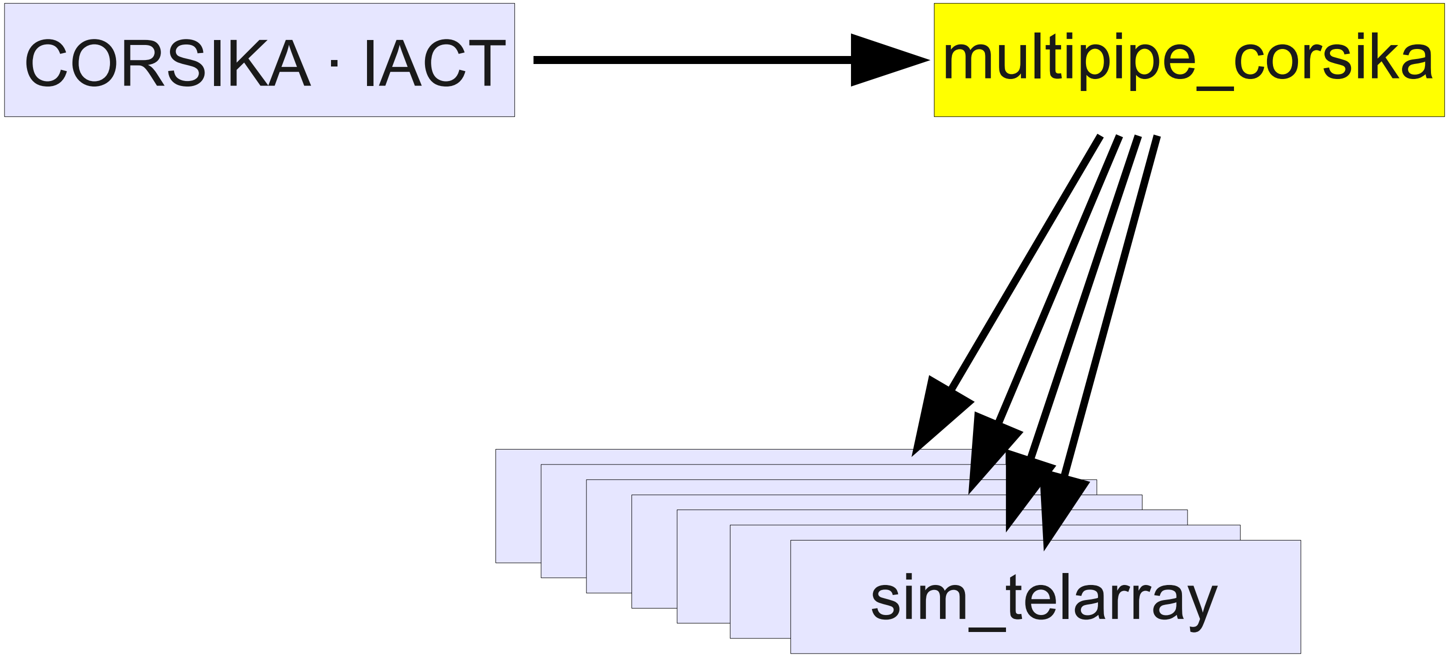
- A new format (name ??), also eventio based, but
  - all C++ (except low-level eventio engine),
  - dynamic (vectors, valarrays, ...) to adapt to actual needs.
  - more flexible (any no. of gains, arbitrary auxilliary data (optional) per pixel, per telescope, ...)
- Convertor from hessio to new format is working but interface into sim_telarray not yet ready.
- Conversion to HESS format may not be possible when the additional flexibility is used.

Running CORSIKA / sim_telarray

Several ways possible:

- First CORSIKA writing a data file and then run sim_telarray for each desired configuration.
- Run CORSIKA with IACT output directly piped into sim_telarray. Only one configuration possible!
- CORSIKA IACT output into multipipe_corsika program and that feeding several times sim_telarray with different configurations.

multipipe_corsika



Failure of individual pipes are tolerated, as long as one pipe remains.
Programs are finished when no pipe is left.

Running with multipipe_corsika

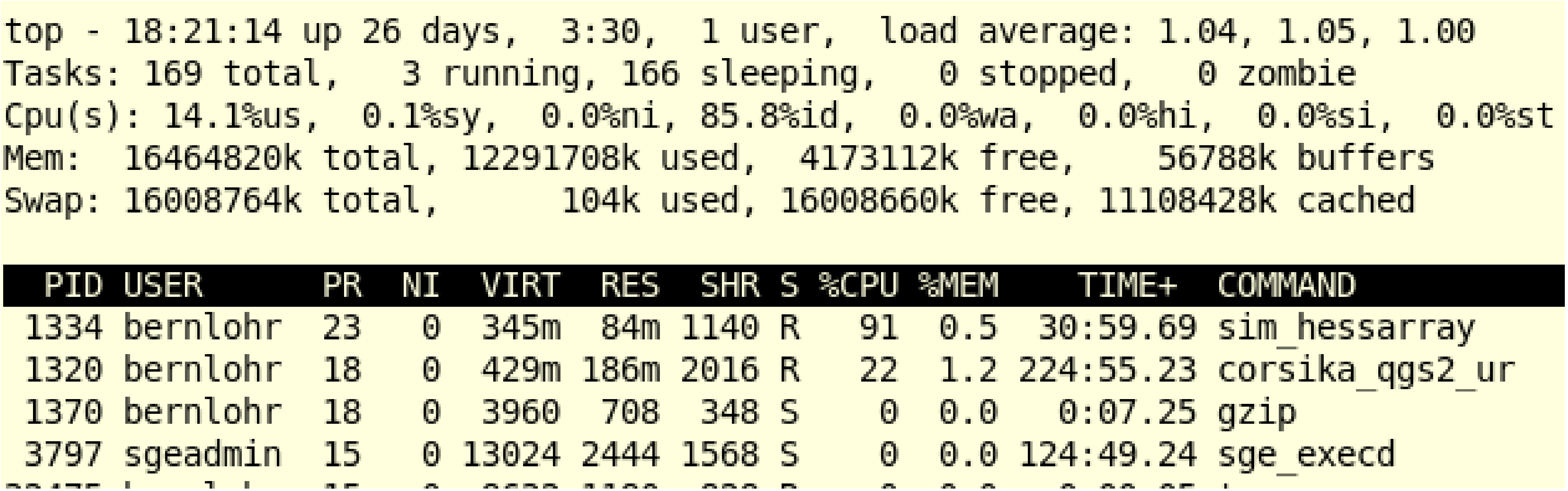
Advantages:

- No I/O bottlenecks because most data just moved in main memory.
- Adjusting of sim_telarray setup is easier thanks to CORSIKA_... environment variables.

Disadvantage:

- Higher main memory requirements when running CORSIKA + multipipe_corsika + n * sim_telarray.

Minimum memory requirements



```
top - 18:21:14 up 26 days,  3:30,  1 user,  load average: 1.04, 1.05, 1.00
Tasks: 169 total,   3 running, 166 sleeping,   0 stopped,   0 zombie
Cpu(s): 14.1%us,  0.1%sy,  0.0%ni, 85.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16464820k total, 12291708k used,  4173112k free,    56788k buffers
Swap: 16008764k total,      104k used, 16008660k free, 11108428k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1334 bernlohr  23   0  345m  84m 1140 R   91  0.5  30:59.69 sim_hessarray
 1320 bernlohr  18   0  429m 186m 2016 R   22  1.2 224:55.23 corsika_qgs2_ur
 1370 bernlohr  18   0  3960  708  348 S    0  0.0   0:07.25 gzip
 3797 sgeadmin  15   0 13024 2444 1568 S    0  0.0 124:49.24 sge_execd
```

In a typical HESS-1 simulation each sim_hessarray copy typically needs ~100 MB, CORSIKA ~200 MB. In big showers perhaps twice as much. But with 100 telescopes, this could well be a gigabyte per process.

Adapting to limited main memory

CORSIKA:

- Max. number of bunches per telescope. Thinning by powers of two when too many bunches received.
- Max. number of bunches kept in main memory while processing a shower. Extra bunches go to temporary files.
- Max. eventio buffer size. Program is terminated when that maximum is exceeded.

Adapting to limited main memory

sim_telarray:

- Telescopes are configured before reading any data. Currently compile-time limits with some pre-processor selected defaults:
  - `make DEFINES="-DCTA"` (up to 100 large tel., ...)
  - `make DEFINES="-DHESS_PHASE_1"` (<=16 tel.)

  or such.
- Need matching hessio library.

Setting up CORSIKA inputs

Possible ways:

- Hand-written files: no good random no. seeds.
- Using hand-written files as templates, replacing run no. and random no. seeds with generated values:
  - ▶ corsika_autoinputs
- Combining configuration fragments from a database, replacing optional variables:
  - ▶ corsika_config

Sample scripts: corsika_autoinputs

```
cd ${CORSIKA_DATA} || exit 1

${SIM_TELARRAY_PATH}/bin/corsika_autoinputs \
  --run ${CORSIKA_PATH}/corsika \
  -p ${CORSIKA_DATA} \
  ${CORSIKA_PATH}/INPUTS_cta_run20974
```

Sample scripts: corsika_config

```
corsika_config -C simulation --redirect-log \
   -c HESS_Phase_1_pipe \
   -k Showers=100000 \
   -k Primary=gamma \
   -k Theta=10 \
   -k Phi=166 \
   -k SimHessPath="$HESSROOT/sim_telarray"
```

Keeping track of things: database tools

- On-line and (more or less) off-line tools.
- On-line tool: corsika_config
  - Needs MySQL database from batch node.
- Various off-line tools reading e.g. CORSIKA inputs file, “.dbase” file, sim_telarray output file, ...
  - No database access from batch node needed.
  - Database entry can be generated at any later time.

Analysis

- We want to be flexible and (almost) everyone of us (plus other interested parties) should be able to work with the MC data - with his/her tools of choice.
- Options with hessio format data:
  - read_hess (quick and, hopefully, not too dirty own development). Mostly C, no external dependencies.
  - conversion to HESS data format (“Sash”, ROOT-based) and analysis with HESS software.
  - conversion to MAGIC format, under development, ...

read_hess

```
Syntax: /home/konrad32c/hess/hessio/bin/read_hess [ options ] [ - |
   input_fname ... ]
Options:
   -p ps_filename  (Write a PostScript file with camera images.)
   -r level        (Use 10/5 tail-cut image cleaning and redo
   reconstruction.)
                    level >= 1: show parameters from sim_hessarray.
                    level >= 2: redo shower reconstruction
                    level >= 3: redo image cleaning (and shower
   reconstruction
                                with new image parameters)
                    level >= 4: redo amplitude summation
                    level >= 5: PostScript file includes original and
                                new shower reconstruction.
   -v              (More verbose output)
   -q              (Much more quiet output)
   -s              (Show data explained)
   -S              (Show data explained, including raw data)
   ...
```

And many more options. Options in red not available with the reduced version read_hess_nr.

read_hess example

```
opts=”-r 2 -u -q ${cuts} ....” # e.g.: --not-telescope 5,6,7,8
gamma_dst="../DST/gamma_${t1}_${t2}_-2.57.simhess-dst.gz"
proton_dst="../DST/proton_${t1}_${t2}_-2.70.simhess-dst.gz"
electron_dst="../DST/electron_${t1}_${t2}_-3.30.simhess-dst.gz"

read_hess ${opts} --powerlaw -2.57 --auto-lookup "${gamma_dst}"
read_hess ${opts} --powerlaw -2.57 --auto-lookup "${gamma_dst}"
read_hess ${opts} --powerlaw -2.57 "${gamma_dst}"

read_hess ${opts} --powerlaw -2.70 --theta-scale 6 "${proton_dst}"
read_hess ${opts} --powerlaw -3.30 --theta-scale 6 "${electron_dst}"
```

Analysis with HESS tools

- In principle two ways feasible:
  - hessio raw data format
    ⇒ HESS Sash raw data format (with "eventio")
    ⇒ HESS Sash DST format
    (with analyse or process_run or ...)
    Well tested and regularly used for HESS MC data.
  - hessio raw data format
    ⇒ hessio DST format (with "read_hess")
    ⇒ HESS Sash DST format (with "eventio", needs mod.)
    Would not require to duplicate raw data, ...