Common Resource Calculation Model

(hints for discussion)

A. Sartirana (IN2P3)

The Context

WLCG Sys. Perf. and Cost Modeling WG.

Resource Calculation Model subgroup:

C. Biscarat, D. Bonacorsi, C. Bozzi, D. Costanzo, D. Düllmann, J. Flix, D. Giordano, J. Iven, D. Lange, A. Sartirana, M. Schulz, A. Sciaba.

Focused (of course) on the **definition of a general model for resource requirements.**

A Common Model

Goal: define a **common FW for modelling the computing requirements** of LHC (and HEP in general) exp.

- have a **general input schema** catching all the details of exp. plans;
- process these inputs into **forecasts for computing requirements;**

that is, we try to **standardize, generalize** and (if possible) **improve** what the exps already do with their "megatables".

Work in progress

We are **still** somehow in a brainstorming phase and we have **more** a list of **questions than answers**

- does this **makes sense**?
- which is a complete/suitable **set of inputs**?
- how to catch **CM details** (e.g. WAN access);
- how to properly describe **network needs**;
- how to describe **new architectures** contribution (e.g. GPGPU);
- time granularity, etc…

A starting point

To **fix ideas** we looked at something existing

https://github.com/kenbloom/resource-modeling

by K. Bloom, O. Gutsche and E. Vaandering
(**all credits** go to them …)

python model:

- # **events**, **cpu** and **storage** requirements;
- **inputs**: **physics** plans (data taking, mc needs) and **computing model** (tape vs disk, data types, sw improvements).

Used for DOE **US-CMS** long-time plans.

This is certainly **CMS-specific** but seems to be a **good starting point** for us.

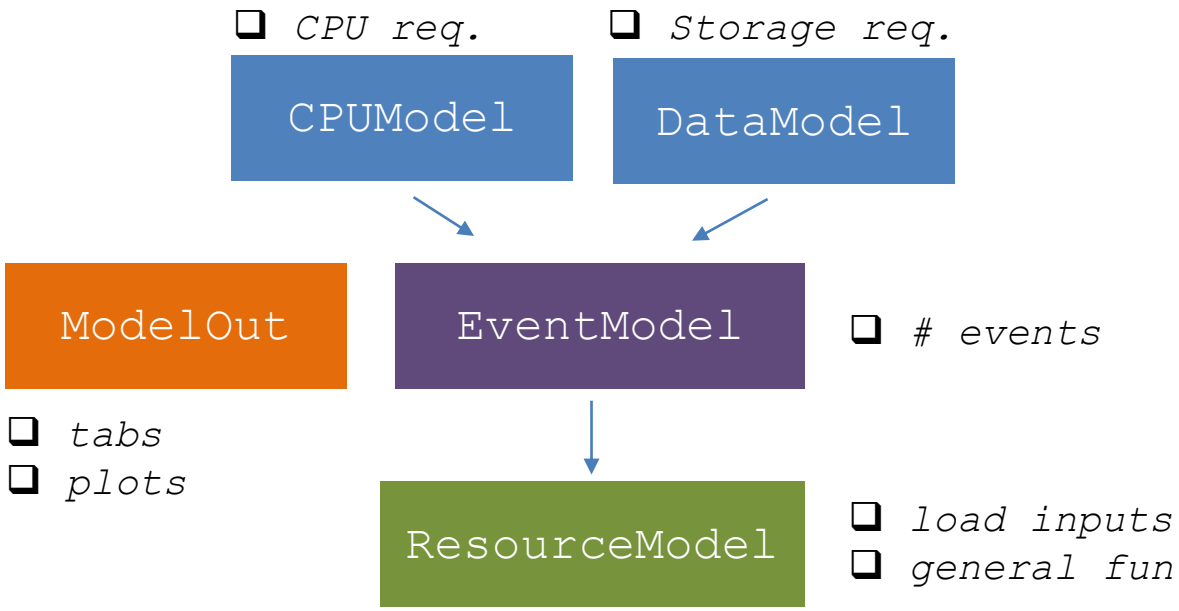
We are **refactoring** it a bit and trying to **generalize** (… so all errors are mine).

Our fork

We are **partially refactoring** the code (work in progress)

https://github.com/sartiran/resource-modeling/tree/wlcg-wg

in the master you can find the original version.



Also a CC7 dockerfile with all the good dependencies for quick start.

Inputs

The **inputs** of the model come as a list of **json files** which are loaded **hierarchically.**

```
./data.py model1.json model2.json …
```

a couple of default models are loaded "BaseModel.json", "RealisticModel.json".

Often inputs contain **time dependent lists.**

```
"parameter": {
    "2016": 123,
    "2026": 456,
    "2027": 678
},
```

This can be **looked up** for a given year returning the **value** and the **last matching year**…

… or it can be **interpolated** linearly.

Dev: allow time dep. values for **generic** time **granularity**

Data Events

This is quite straightforward.

For a given year look up in the model:

- `live_fraction`

```
"live_fraction": {
     "2016": 0.247,
     "2026": 0.165,
     "2027": 0.247},
```

- `trigger_rate`

```
 "trigger_rate": {
  "2016": 1000.0,
  "2026": 10000.0
 }
```

then:

```
# DATA EVENTS = SECONDS_PER_YEAR * LIVE_FRACTION * TRIGGER_RATE
```

MC Events

In the model we have the **mc fraction** for each year (to be looked up) for each **MC "Type" labelled by year**

MC Type

```
"mc_evolution": {
  "2017": {
    "2017": 1.9, "2024": 1.9, "2025": 0.5, "2026": 0.03,
    "2027": 0.0, "2050": 0.0},
  "2026": {
    "2017": 0.0108, "2024": 0.03, "2025": 1.0,
    "2026": 2.0, "2050": 2.0}
},
```

```
# MC_EVENTS = FRACTION[type][year] * # DATA_EVT
```

- **mc_year <= year:** data evts at `year`(or last non-shutdown year);
- **otherwise:** max between evts at `year` and at `mc_year`.

Resources per events

Now need to **translate events in resources**…

… this means, for an evt of a given data type/version, **data size and cpu time** (HS06*s) required to process/simulate it

… this is looked up from input

```
"tier_sizes": {
   "AOD": { "2017": 400000.0, "2026": 2000000.0},
   "GENSIM": {"2017": 1000000.0, "2026": 1000000.0},
…}
```

```
"cpu_time": {
  "data": {
     "RECO": {"2017": 250, "2026": 4000.0}},
  "mc": {
     "DIGI": {"2017": 100, "2026": 2500.0},
     …}}
```

Resources per events

For cpu take into accounts **improvement factors**

```
"improvement_factors": {…
  "software_by_kind": {
    "2017": {"2017": 1.05, "2024": 1.01, "2025": 1.0, "2050": 1.0},
    "2026": {"2017": 1.01, "2024": 1.1, "2025": 1.2, "2027": 1.1,
…}}},…
```

MC Type (hardcoded hack for data)

Dev: IF per data tier/type?

Product IF from «start» to proc. year

CPU Time = (Data CPU time)/(integrated SW improvement factor)

Possible dev: can't we get **more about evts. req. res**

- SW: **parallelizability**, **memory** usage, **IO**,…;
- CM: remote access, …;
- …

Storage Requirements

 # evts and evt size give us data production (per type, tier and year). We need the **storage occupation**

[this year, next-year, beyond-next-year]

```
"storage_model": {
  "disk_replicas": { "AOD": [0.2, 0, 0], "GENSIM": [0.1, 0, 0],…},
  "tape_replicas": { "AOD": [1, 1, 1], "GENSIM": [0, 0, 0], …},
  "versions": { "AOD": [1, 1, 0], "GENSIM": [1, 0, 0], …}
}
```

Dev: more flexible planning?
is this static?

For a **given year** we consider the **data produced at some previous year** per type (for mc sum over mc kind) and tier taking the **number of replicas and version on disk/tape** from the input above (taking the suitable value of the ntuple according to year – production_year).

Summing over all this we know how much data we have on disk/tape for the year.

CPU Requirements

Data **prompt-reco cpu requirements** for a given year

```
RECO_CPU_TIME = T0_FACTOR * EVENTS * RECO_EVENT_CPU_TIME
```

```
…
"t0_factor": 1.5,
…
```

Dev: time dependent?

express, repacking, AlCa, CAF functionality and skimming (scale with data). And a general "security factor" for peak days.

Processed almost **real time**

```
RECO_CPU_CAPACITY = RECO_CPU_TIME / BEAM_TIME
```

CPU Requirements

For **RE-RECO** everything is **hardcoded and CMS specific** based on how much we expect to re-reco of the current year and of the past year and in how much time.

```
RE_RECO_CPU_TIME = 1.25 * EVENTS * RECO_EVENT_CPU_TIME
```

```
RE_RECO_CPU_CAPACITY = EVENTS * RECO_EVENT_CPU_TIME/3*SEC_MONTH
```

Dev:

- this has to be **re-done** with a **general** computation: **input from other experiments** will help;
- define a suitable **set of parameters**. E.g. a time dependent table with **re-reco fraction for y and y-1** (general enough?).

CPU Requirements

MC cpu requirements is quite straightforward

```
MC_CPU_TIME[MC_KIND] = (GENSIM_EVENT_CPU_TIME +
DIGI_EVENT_CPU_TIME + RECO_EVENT_CPU_TIME)*EVENT[MC_KIND]
```

…and **summing over MC Kinds.**

We assume **MC runs all year long** so the capacity is computed dividing by the seconds in the year…

```
"new_detector_years
": [
     2017,
     2018,
     2026
 ],
```

… but in the **years with new detector we multiply by 2** the MC CPU required capacity as we assume we will have less time to make MC.

Crude and hardcoded. Use a time dictionary with (non std) time factors for activities.

CPU Requirements

The **CPU for analysis** is simply estimated by **multiplying for a factor** the sum of **all other activities** (time and capacity).

This is a bit **"reshaped" for the time up to HL-LHC:**
Take 2018 as base. Each year add in a year of the same size of the previous one (except for shutdown).

- no improvement factor: OK as it is IO bound;
- assumed to run over year.

```
"analysis_factor": 0.75,
…
"analysis_hllhc_planning"
: {
   "2019": 1.333,
   "2020": 1,
   "2021": 1,
   "2022": 1.25,
   "2023": 1,2,
   "2024": 1,52,
}
```

Factor wrt previous year

There is much to be done here. Looking at the similar thing for another exp may be a good starting point.

Questions/Thoughts

- Is this **useful**? Which are the minimal **requirements**?
  - general **format for input and output;**
  - **richer** required resources **description;**
- how can we define **NW resources requirements** in a sensible way?
  - some work in progress on this…;
- how can we make an event resources description richer?
  - more about what requires to process it: memory, parallelizability, (remote) IO,…

Backup Slides

Storage Requirements

To this we add a **`static`** **storage** according to values given in the input

```
"static_disk": {
    "Ops space": {
      "2017": 1.3e+16,
      "2026": 6.5e+16
    },
    "Run1 & 2": {
      "2017": 0
    }
  },
  "static_tape": {
    "Run1 & 2": {
      "2017": 1.21e+17
    }
  },
```

To this the "tier" while data type is "Other"

CPU Requirements

For **Long Shutdowns** we compute the CPU requirements as follows:

- reconstruction of the data of the 3 previous years;
- 3 times the MC amount of the previous year.

This over the whole year.

This is harcoded, CMS specific and has to be improved…

Expected Resources

```
"capacity_model": {
    "cpu_delta": {
      "2017": 300000.0,
      "2020": 800000.0
    },
    "cpu_lifetime": 5,
    "cpu_start": 1630000.0,
    "cpu_year": 2017,
…
"improvement_factors": {
    "disk": 1.1,
    "hardware": 1.1,
…}
```

This is not strictu sensu "resource requirements" but it is often looked at while computing resource requirements.

We start with "cpu_start", assume it was purchased linearly in the last "cpu_lifetime" years.

```
ADDED[YEAR] = DELTA[LOOKUP_YEAR]*
  FACTOR ** (YEAR – LOOKUP_YEAR)
```

```
CAPACITY[YEAR] = CAPACITY[YEAR - 1] + ADDED[YEAR] –
                 ADDED[YEAR - LIFETIME]
```

Same with disk and tapes.