WP1 - parsers, a whole lot of them...

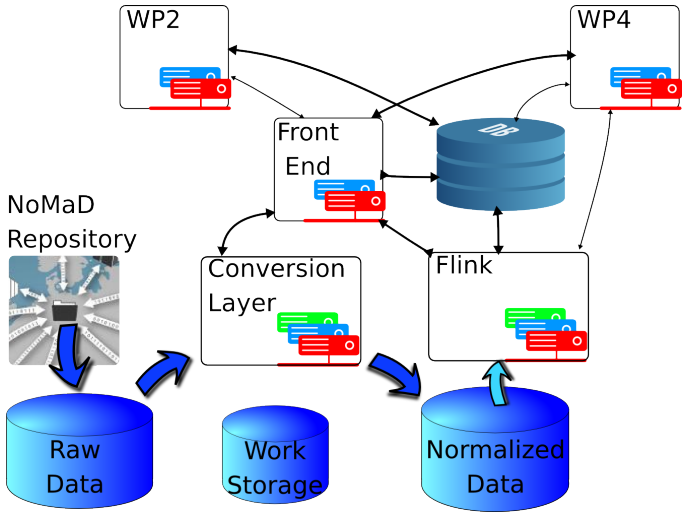
Fawzi Mohamed

27.10.2015

NOMAD
NOVEL MATERIALS DISCOVERY

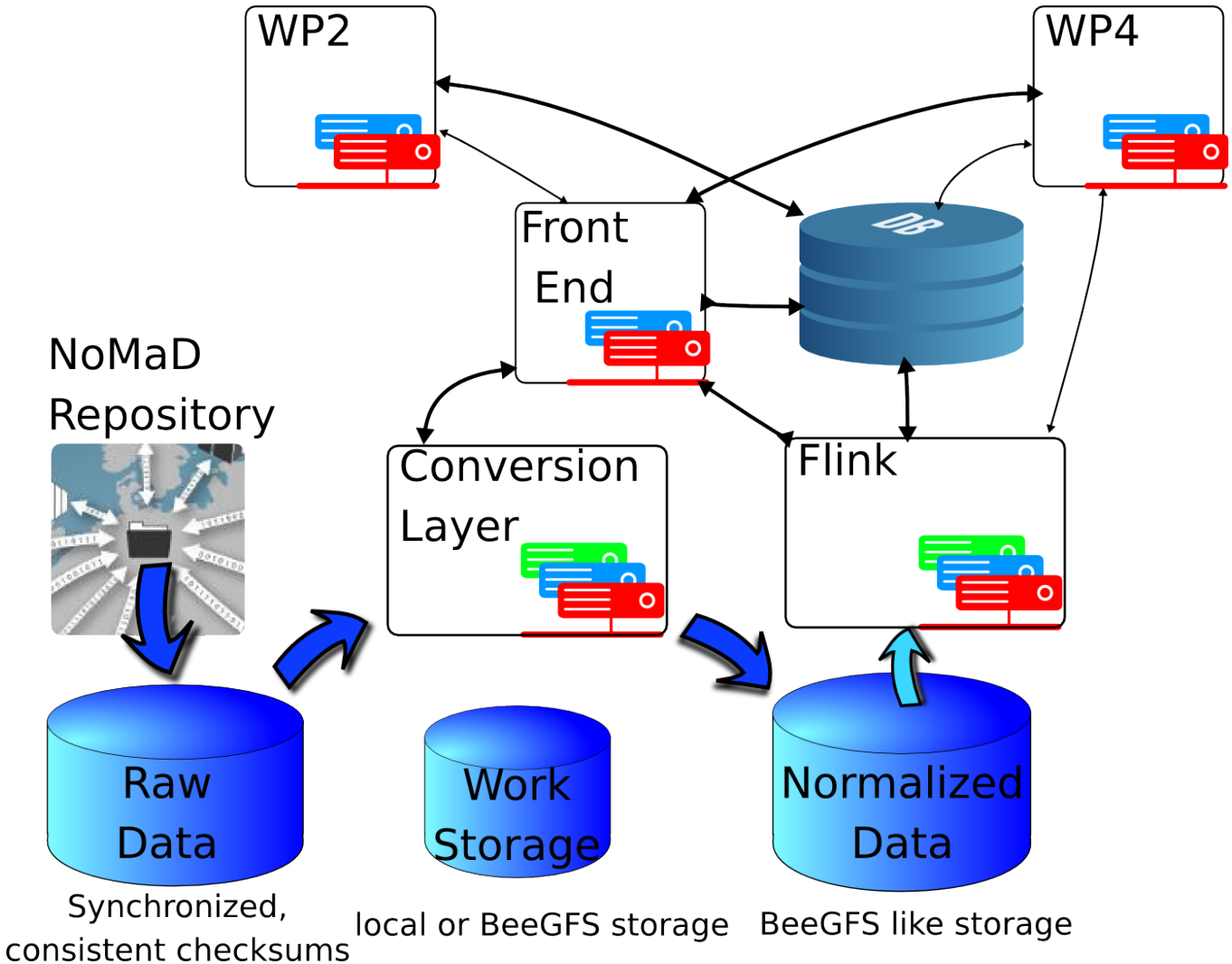
NOMAD-DB

- code independent representation
- Flink for map/reduce and more advanced queries
- enable big data analysis between and across different codes



NOMAD
NOVEL MATERIALS DISCOVERY

High-level architecture



Map & reduce

- way to express some algorithms that makes them easy to parallelize, became popular after Google article
- can work well on distributed data

NOMAD
NOVEL MATERIALS DISCOVERY

Flink 

- Flink, started here in Berlin
- One of the leading frameworks for data-flow and streaming optimization that improves on the map reduce approach
- Tries to support not just data-flow or stream processing but also iterative methods, graph processing and some machine learning algorithms

NoMaD Repository

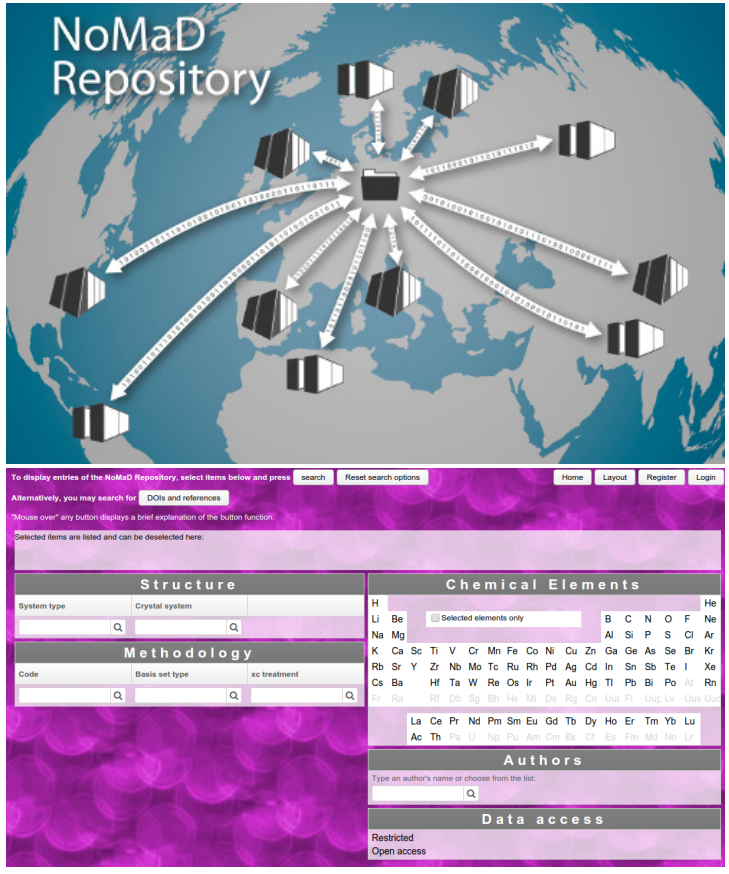
- `http://nomad-repository.eu/`
- Joint effort by the FHI (Matthias Scheffler), HUB (Claudia Draxl) and the MPCDF Garching (Stefan Heinzel).

- Lorenzo Pardini
- Fawzi Mohamed
- Hermann Lederer
- Johann-Christoph Freytag
- Christian Carbogno
- Thomas Zastrow
- Pasquale Pavone
- Luca Ghiringhelli
- Binyam Gebrekidan Gebre
- Former members:
  - Evgeny Blokhin

NOMAD
NOVEL MATERIALS DISCOVERY

NoMaD Repository

- source of data for the repository
- encourage data sharing, re-purposing and validation
- large amount of open access data
- 634'014 entries, OQMD is being added, materials project will follow



NOMAD
NOVEL MATERIALS DISCOVERY

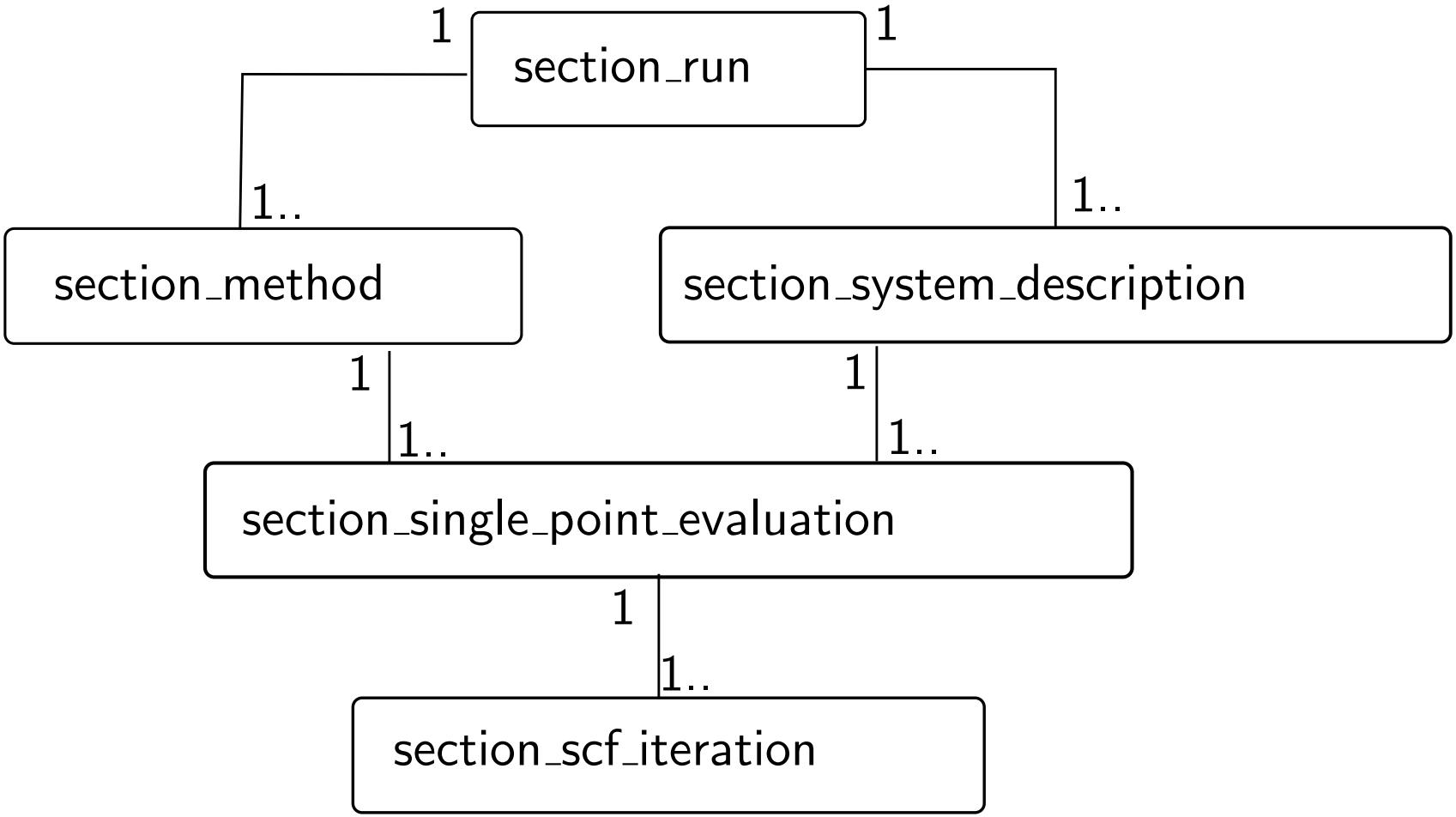
Parsers... for NOMAD

- extract information from simulation input and outputs to make it available for analysis
- information that is not extracted is invisible to us, a parser defines the data that can be analyzed
- to make the data processable in an automatic way it should be mapped to a clear model

Meta data: our conceptual model

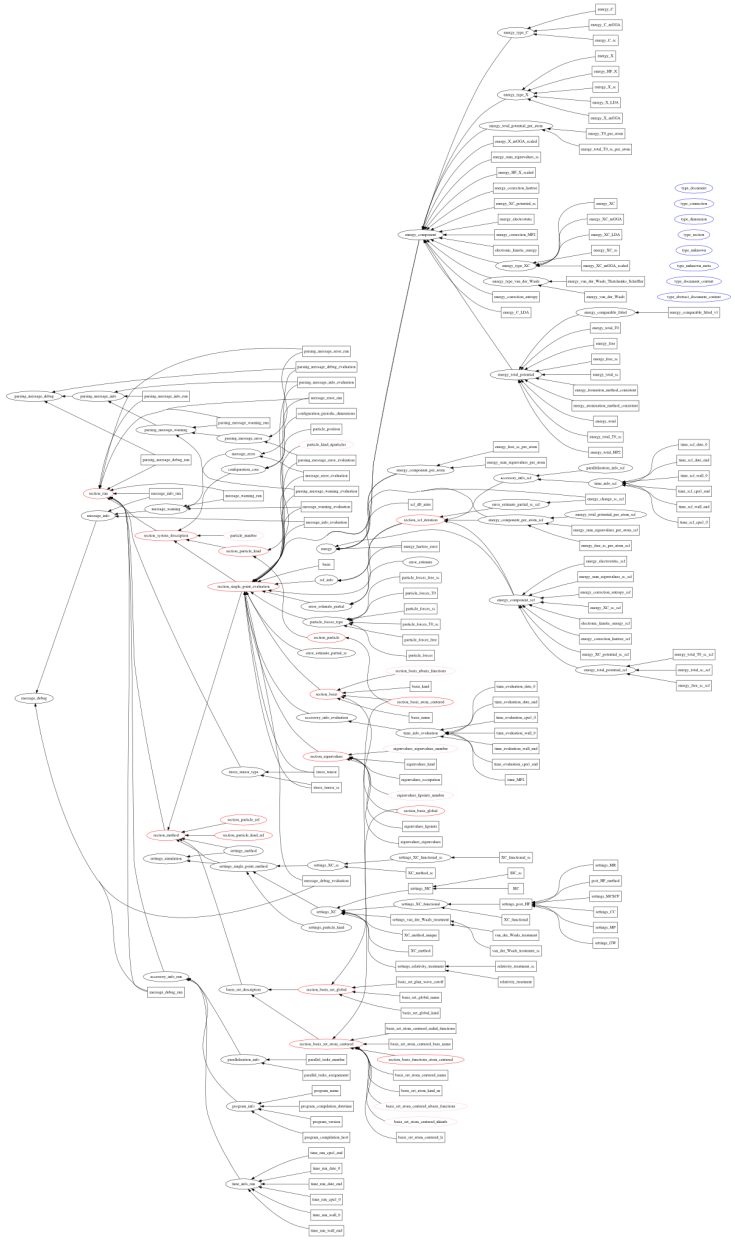
- define how the data that we extract is organized, and what it is
- important both for human and for the machine
- data values consist of simple data types and multidimensional arrays of them
- group together similar *types* making them inherit from the same type (all energies inherit from the *energy*)
- group together *values* with sections
- allow one to many relationships between sections

Common meta data: how to describe code independent quantities



NOMAD
NOVEL MATERIALS DISCOVERY

Common meta data



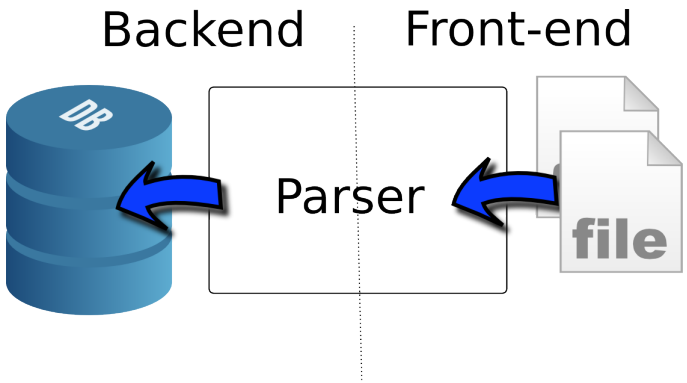
https://nomad-dev.rz-berlin.mpg.de/wiki/NomadMetaInfo

NOMAD
NOVEL MATERIALS DISCOVERY

What did we learn on parsers

- parsers should be fast because we want to apply them to large quantities of data (and re-parse regularly)
- parsers should be usable in various contexts
- code change in time, parsers need to evolve
- we will maintain and improve them for a while

Decoupling the parsers



- Independent systems are more robust
- can be changed or optimized independently
- can be reused in different contexts
- but the interface has to be chosen carefully, because it will dictate performance and complexity

The simplest kind of efficient parser

- push parser
- call back based
- can stream (avoids loading everything in memory)
- main problem:
- you cannot tell the parser to skip some info
- solve this by adding the possibility to tell the parser about which info you are interested in

The dangers of freedom

- what is not seen by parsers is not seen by analysis
- data reliability is one of the most challenging problems
- we do not want throwaway parsers, parsers should detect subtle problems
  - did the program encounter a strange situation during convergence
  - where there warnings? Do they get propagated or is it just a line somewhere in the output
  - where there multiple runs in the same file? are they detected correctly?
  - . . . and in the same directory? Are ancillary files really associated with the current run? What do creation dates say?. . .
  - contact with the code developers can help

declarative parsers

- try to describe the information that will be extracted
- we already have a way to do that: the meta data, we can extend it to describe code specific things too
- try to describe where to extract it
- example FHI-aims parser v3 written in python
- describe what should be done, but not how to do it: several ways to compile it into a real parser: adaptable and efficient
- close to documenting the thing to be parsed
- simpler for another person to change or optimize the parser (more optimization potential)

Declarative parsers problems

- difficult to describe transformations declaratively
- can be more tedious to write
- can be more difficult to debug (supporting tools can help here)
- possible solutions:
  - many derived quantities (like the normalized values) can be calculated at the section closing with a bit of caching
  - more complex normalization can be performed by another program.

The ideal parser

- starts with a declarative parser capable of parsing basically all information contained in an output
- optimizes it to extract the quantities required to calculate the code independent representation
- calculate the code independent quantities and return them
- can be reused in different contexts
- we can later decide that a quantity we ignored is now of interest.

WP1: not only parsers

- meta data tools
- getting raw data to parse, unique identifiers
- uncompress, find out which parser to use
- try to keep parsers minimal $\rightarrow$ common transformation in normalization step
- URI and interface to access pieces of data
- DB for meta data and references

Identifiers

- identifier (gid) uses a small prefix (depending on what was checksummed) + the first 28 characters (168 bits) of the base64 encoding of the SHA-512 digest to identify most things (files, metadata, normalized data...)
- this allows one to build uri (`nmd://gid/path`) that refer to single quantities, or files within an archive
- uri do not depend on where the file is stored: ready for distributed approach

NOMAD
NOVEL MATERIALS DISCOVERY

- FHI-MPG:
  - FHI-aims, VASP
  - *Quantum Espresso*, abinit, Dmol, Dmol^3, *CASTEP*
- HUB
  - exciting
  - WIEN2k, ELK, FLEUR, FPLO
- UB
  - Gaussian, GAMESS, NWChem, Molcas, CRYSTAL
  - *DL_POLY*, GULP
- KCL
  - onetep, *CASTEP*, *LAMMPS*, *DL_POLY*, LM Suite (TB-LMTO-ASA)
  - ASE related
- CAM
  - *CASTEP*, QUIP/libatoms/GAP, Molpro, *LAMMPS*
  - ASE related
- AALTO
  - cp2k
  - *VASP*, GPAW, *LAMMPS*, *Quantum Espresso*
  - Smeagol, Octopus, Crystal, BigDFT, SIESTA
- MPSD-MPG
  - *Quantum Espresso*, octopus?
- DTU
  - GPAW and ASAP
  - ASE: Elk, gromacs, MOPAC, SIESTA