sys-sage

A library for capturing HPC systems' topology and attributes

Stepan Vanecek

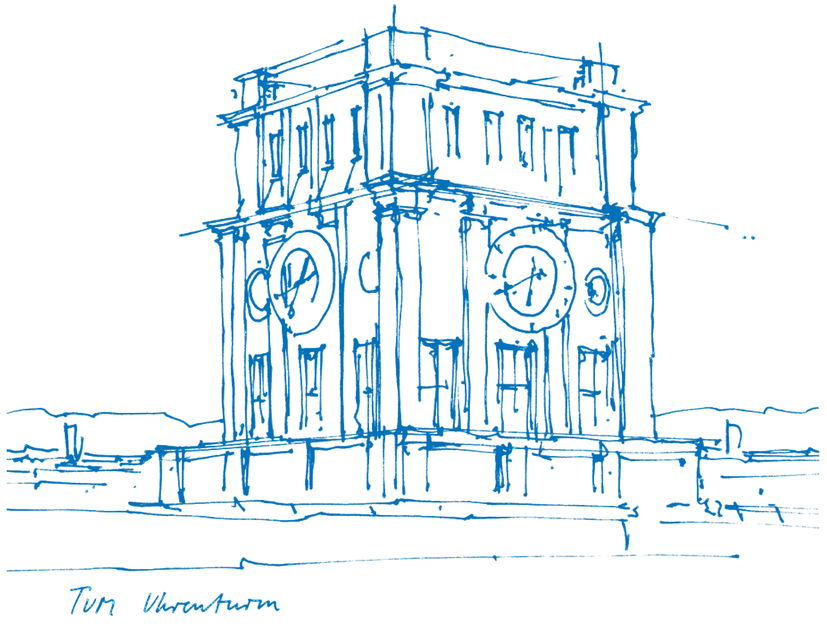
stepan.vanecek@tum.de

Research Group Prof. Martin Schulz

Chair of Computer Architecture and Parallel Systems

Technical University of Munich

Scalable Tools Workshop 2022, 20.06.2022



sys-sage

- Software library with a (C++) API to query system topology information
- Currently under development
  - Core functionality implemented
  - More data collection and functionality under development

Main goal: Store, update, and provide all relevant information about

- Hardware topology,
- **Dynamic system state**/configuration,
- System capabilities, and
- Other data related to HW

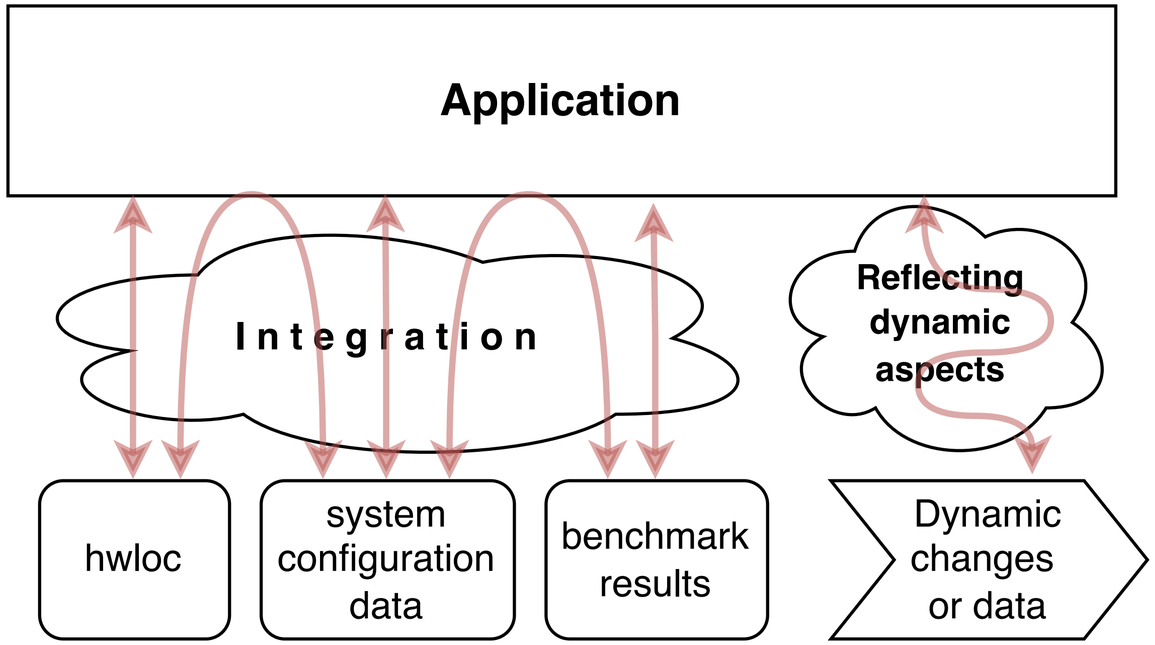
from **different data sources** logically connected to each other.

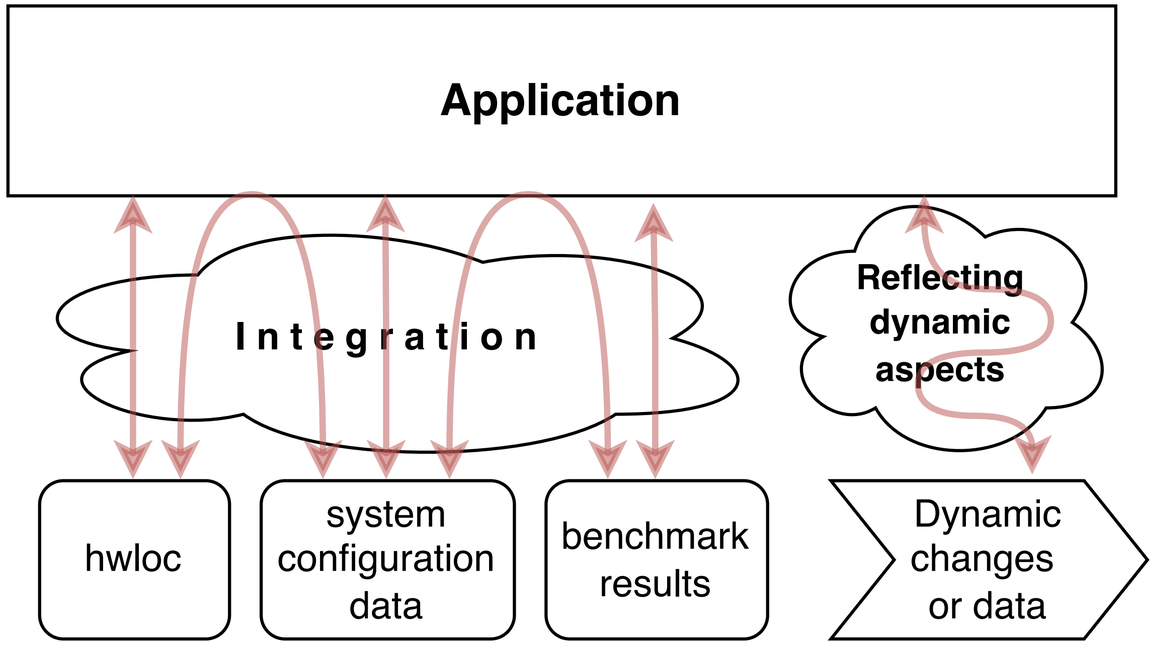
sys-sage vs. hwloc

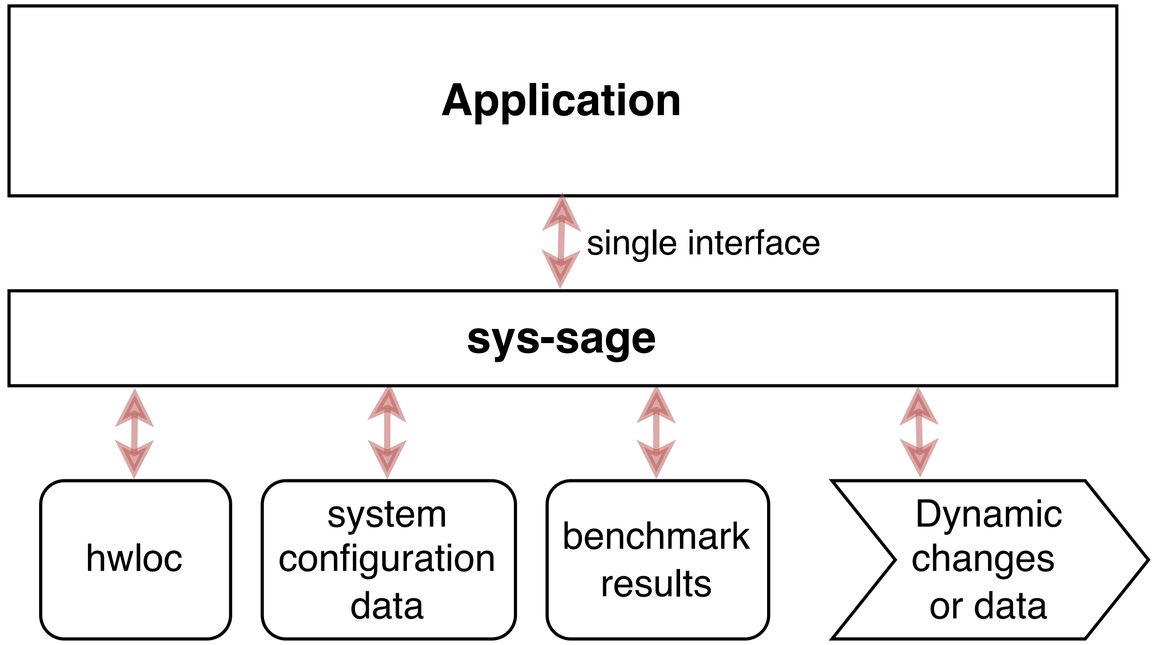
Extension to hwloc (*hwloc is not a mandatory part*)

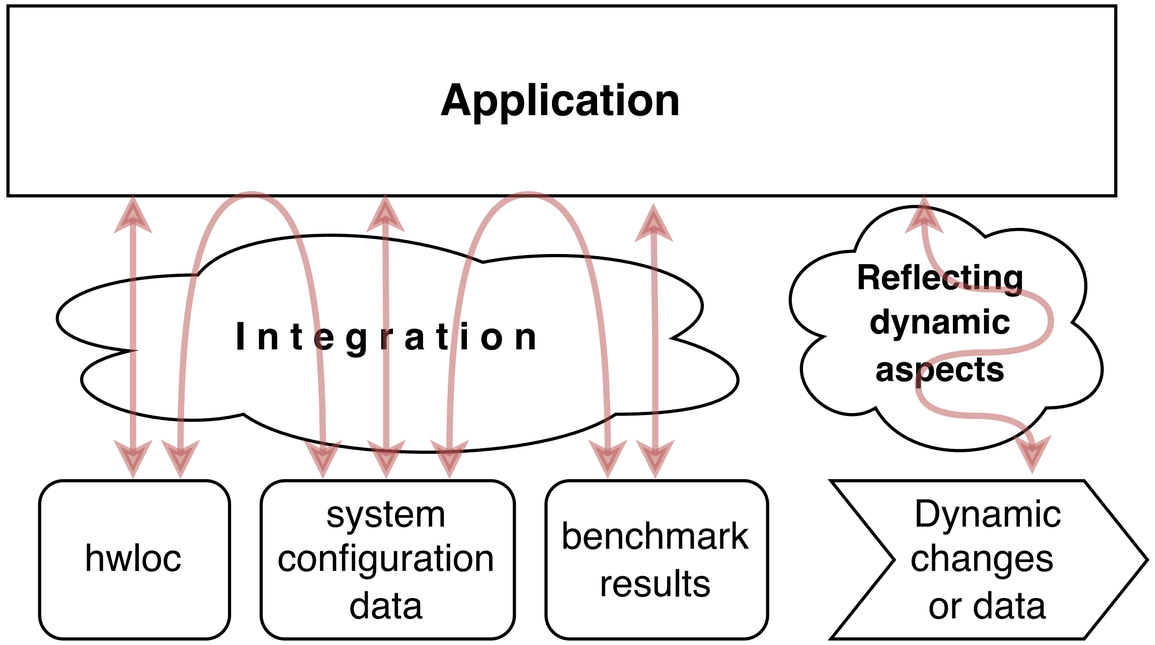
hwloc is limited to:

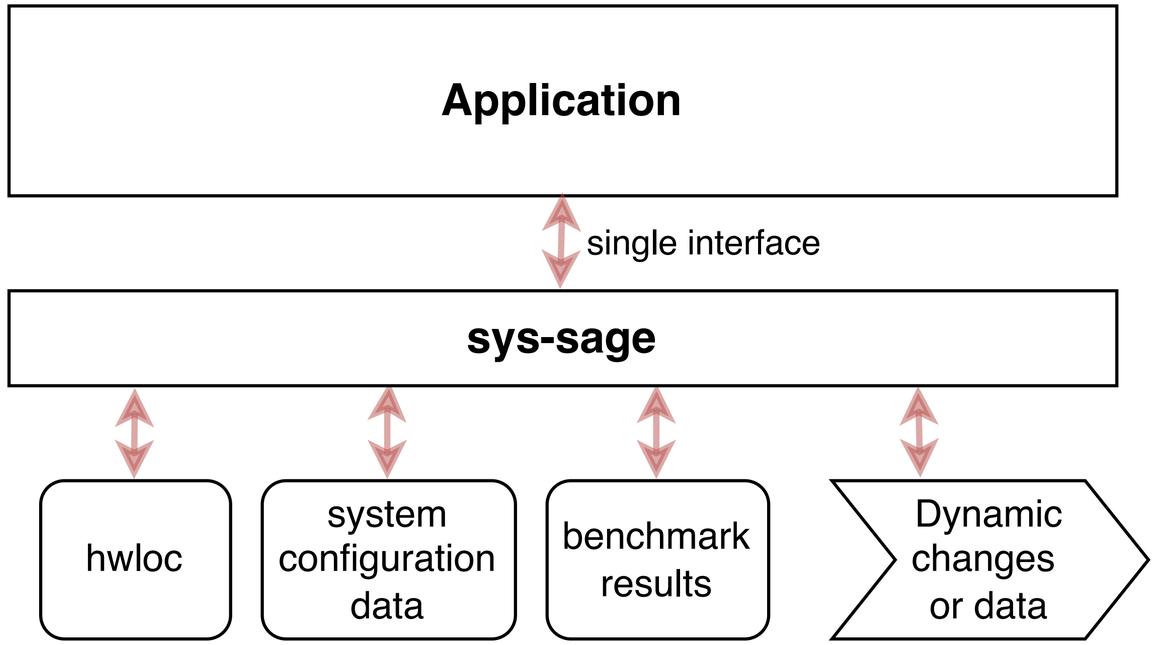
- Static data (only given HW topology)
  - Modern systems are not strictly hierarchical anymore
  - Information regarding data movement capabilities is missing
- Mainly CPU-centric
- Difficult to incorporate complementary information


Application
Integration
Reflecting dynamic aspects
hwloc
system configuration data
benchmark results
Dynamic changes or data


Application
Integration
Reflecting dynamic aspects
hwloc
system configuration data
benchmark results
Dynamic changes or data


Application
single interface
sys-sage
hwloc
system configuration data
benchmark results
Dynamic changes or data





sys-sage serves as a backend for HW-related-data

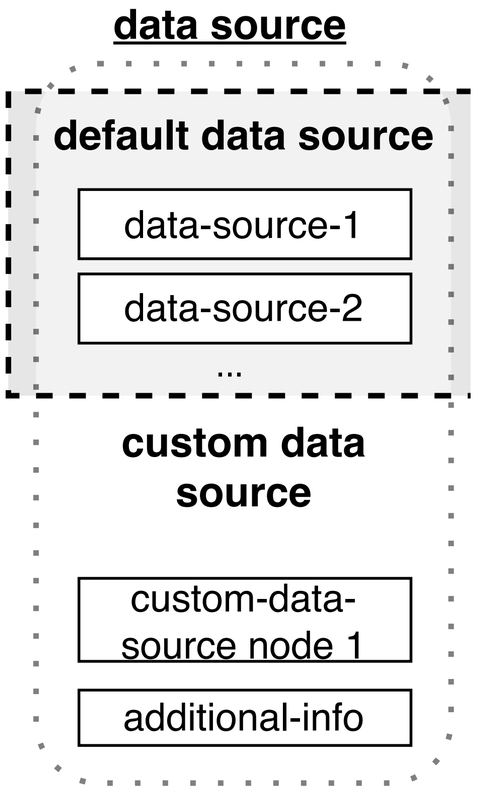
Internal representation logically connects information from different sources

What sys-sage Addresses

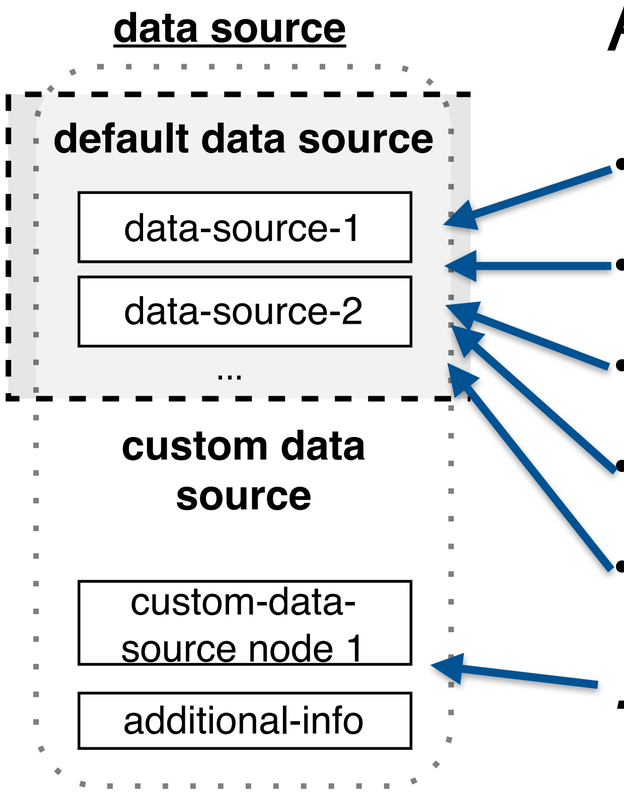
1. Dynamic aspects of modern HPC systems
   - Data movement information
   - Variable system characteristics
2. Support of heterogeneous components
   - CPU, GPU (more to come)
   - Interconnects / buses connecting the components
3. High variability
   - Different set of information needed in different use-cases
   - Arbitrary data can be added to already existing representation
   - sys-sage can store/maintain/provide arbitrary information out-of-the-box

Possible Areas of Usage

- Job / thread scheduling, co-scheduling multiple applications
- Autotuning tasks / applications
- Data management on heterogeneous memory systems (allocation decisions)
- Power management
- Performance optimization, performance modelling tools

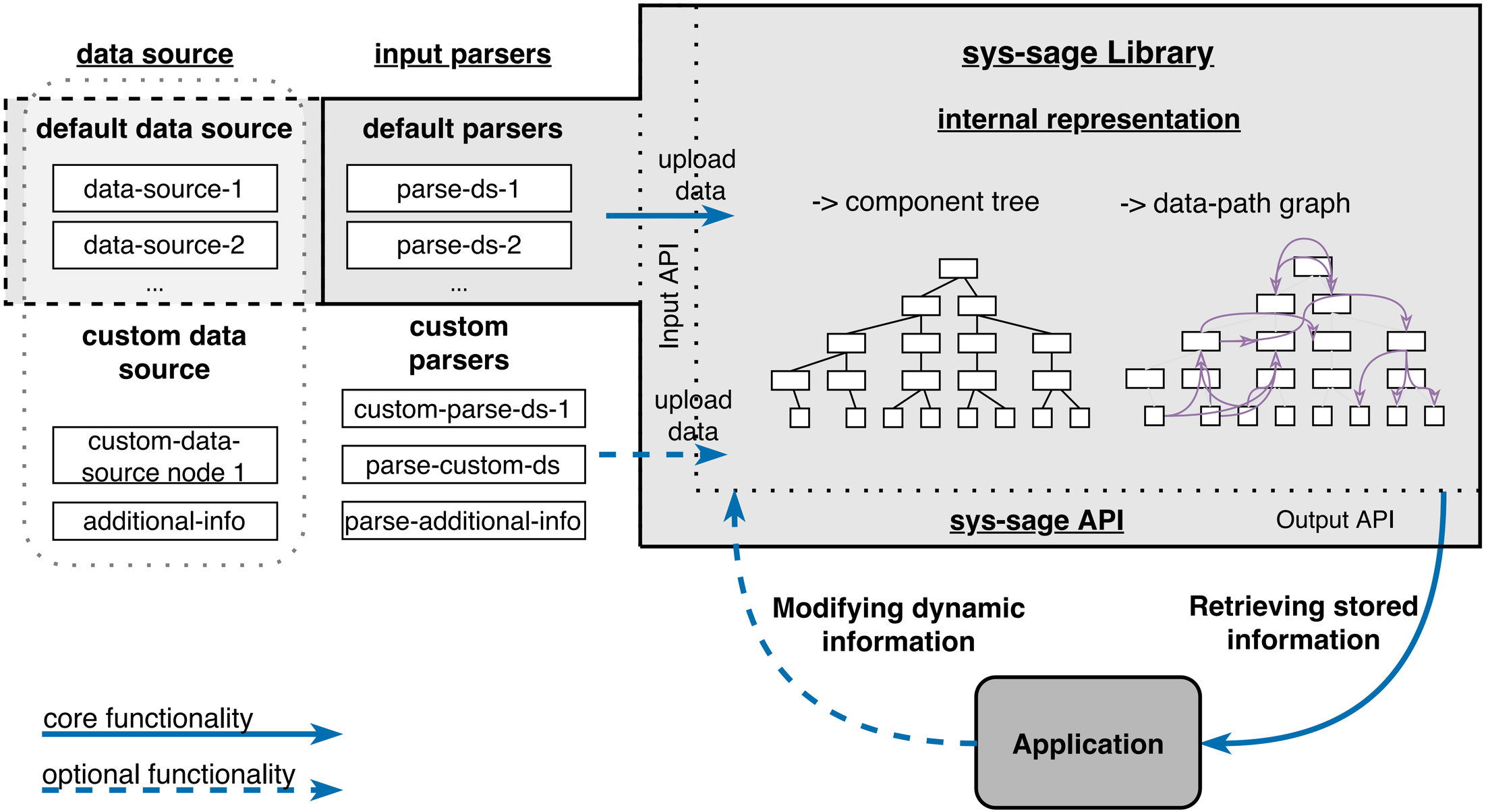

data source
default data source
data-source-1
data-source-2
...
custom data source
custom-data-source node 1
additional-info

Available Data (more to come)



- Uploading HWloc Topology output
- NUMA memory and cache bandwidth/latency benchmark
- Cache-partitioning aware available L3 cache size
- PCIe bandwidth+latency (WIP)
- GPU HW topology information (WIP)

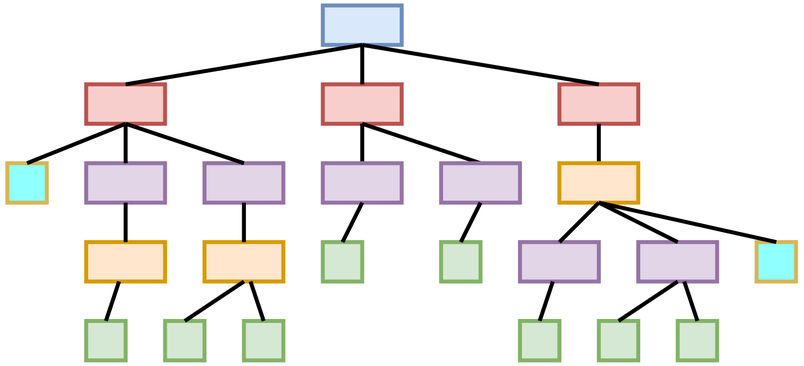
…any custom piece of information may be added using the API


data source
input parsers
sys-sage Library
default data source
data-source-1
data-source-2
...
default parsers
parse-ds-1
parse-ds-2
...
internal representation
upload data
-> component tree
-> data-path graph
Input API
custom data source
custom-data-source node 1
additional-info
custom parsers
custom-parse-ds-1
parse-custom-ds
parse-additional-info
upload data
sys-sage API
Output API
Modifying dynamic information
Retrieving stored information
Application
core functionality
optional functionality

Representation of an HPC System

- 2 main concepts
  - Component Tree
  - Data-path Graph

Representation of an HPC System



Component Tree

- Composed of *Components*
- Hierarchical representation (hwloc)
- Easy orientation
- Components contain rather static information (id, size, attributes)
- No restrictions on the hierarchy / *Component Types*

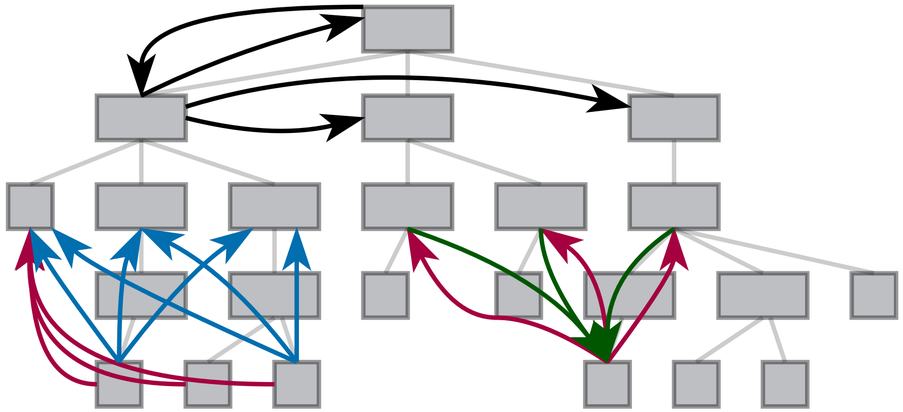
Various *Component Types*:

- Topology (root)
- Node
- Chip (socket)
- Subdivision
- NUMA
- Cache
- Core
- Thread (PU)
- Memory
- Storage

Representation of an HPC System

Data-path graph

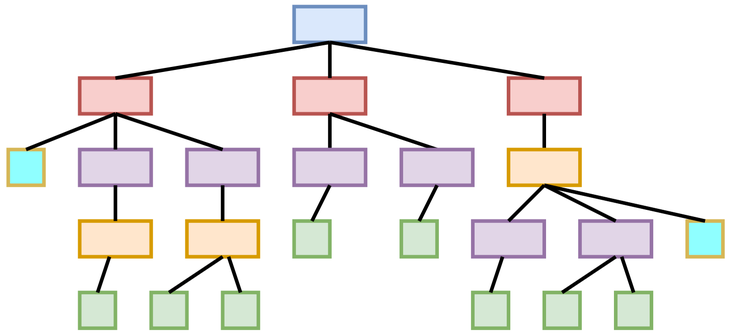
- Connects two arbitrary *Components*
- Utilizes the *Component Tree* elements
- Mainly dynamic information
- *Data-paths* may contain arbitrary information
  - bandwidth, latency
  - cache partitioning settings
  - PEBS memory samples
  - …



Representation of an HPC System

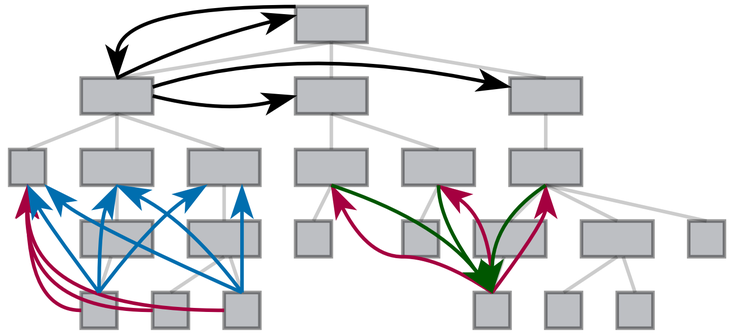
Component Tree

- Hierarchical representation (hwloc)
- Mandatory structure
- *Components* contain rather static information (id, size, attributes)



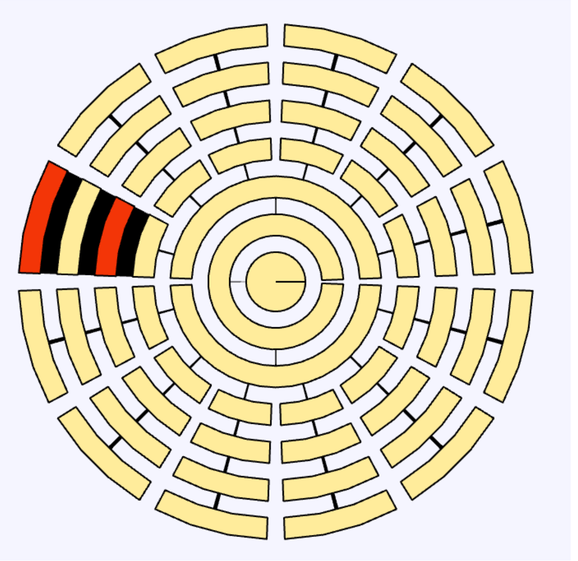
Data-path Graph

- Connects two arbitrary *Components*
- Orthogonal to the *Component Tree*
- Optional
- Expresses relation of 2 HW elements



Use-case 1: MemAxes

- Visualize data access characteristics
- Grouping PEBS samples by core and cache/NUMA region



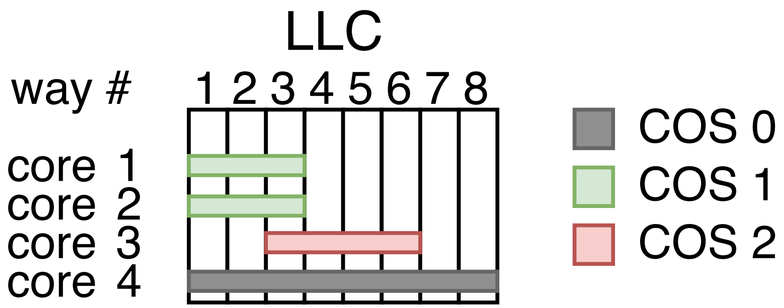
- *Component Tree* from hwloc
  - Contains CPU cores+caches+NUMA regions
  - Used by MemAxes for a system visualization
- (WIP) Represent memory samples as *Data Path* objects
  - source: cache/NUMA region that provided the data
  - target: issuing core
  - all sample data stored in the *Data Path* as well
    - (latency, source code line, timestamp,…)

Use-case 2: Cache Partitioning

- Retrieving the available L3 cache size for each core
  - A core may have access only to subset of L3 (different subset for each core possible)
  - “Static” hwloc total L3 size not valid anymore

- *Component Tree* from hwloc
  - Contains CPU HW topology (cores+caches+sockets…)
- L3 cache partitioning as *Data Paths*
  - source: each core
  - target: L3 cache
  - store # of open cache ways for given core

    ➥ calculate available cache size



Some More Possible Use-cases…

- Compare on-node characteristics within a multi-node system to find outliers
- Map OMP threads / tasks to HW threads
  - This will help connect HW samples / counters to SW threads
- Use sys-sage to decide where to allocate compute- and memory intensive threads
  - Even threads within a NUMA region have different characteristics
- Use autotuning strategies to optimize application performance on given node using data stored in sys-sage
  - e.g. react to different cache prefetching strategies

sys-sage

Check the GitHub repo:

https://github.com/stepanvanecek/sys-sage

Install with spack:

spack install sys-sage

Get in touch with us:

stepan.vanecek@tum.de

martin.w.j.schulz@tum.de

Acknowledgements:

DEEP-SEA
EU Grant #955606
BMBF #16HPC014

SPONSORED BY THE

Federal Ministry
of Education
and Research