No. 188
December 2014

Bits & Bytes

RECHENZENTRUM GARCHING

Garching Computing Center of the Max Planck Society and the Institute for Plasma Physics*
Boltzmannstraße 2, D-85748 Garching bei München

# Taco - A Metadata System for Hierarchically Structured Data Collections

Thomas Zastrow, Karin Gross, and Raphael Ritz

## Introduction

The Taco system (short for 'Tags and Components') is a software stack for discipline-independent handling of metadata, the additional descriptive information about some data in question. Here, metadata are structured as *key-value pairs* – the **tags** – that allow to specify simple attributes in a predefined way once configured. Tags can be grouped into **components** to express relatedness and to structure the user interface. Taco can be used in the humanities as well as in natural sciences which are dealing with large collections of texts, images, videos or other digital data. The Max Planck Institute for Ornithology was the first user of Taco and accompanied the development from the beginning.

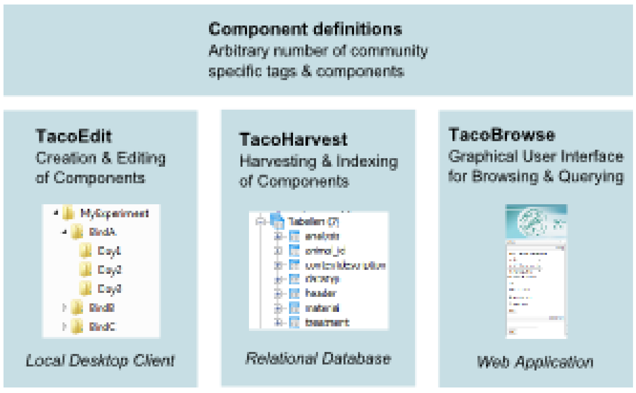

Component definitions
Arbitrary number of community specific tags & components
TacoEdit
Creation & Editing of Components
TacoHarvest
Harvesting & Indexing of Components
TacoBrowse
Graphical User Interface for Browsing & Querying
Local Desktop Client
Relational Database
Web Application

Figure 1: The intrinsic parts of the Taco system

Taco allows to assign metadata directly to the folders of a file system or any subtree thereof. Such a directory structure often represents meta information about the data it contains, but because this information is not bound to concrete datastreams of bits, it is often not included into traditional metadata formats which are used to describe data stored in files. Taco provides an alternative approach to make this – and additional – information explicit. Taco consists of several independent parts (Fig. 1): **TacoEdit** for the scientist who provides the information, **TacoHarvest** for the system administrator who collects and processes the data and **TacoBrowse** for the end user who can search and browse based on the information provided. All parts can be used to create a community-specific setup, using one and the same configuration.

## The configuration

The configuration file is the heart of the Taco system. It is a simple, human readable text file (in csv or Excel format) which contains the definitions of all the tags and components which are available in the Taco system. All three parts of Taco make use of the tags and components as defined in the configuration file:

- At every start, TacoEdit reads the configuration file and builds the graphical user interface around the tags and components it finds.
- TacoHarvest reads the configuration file to create the database schema out of the tags and components.
- TacoBrowse utilizes the configuration to offer different search possibilities to the user.

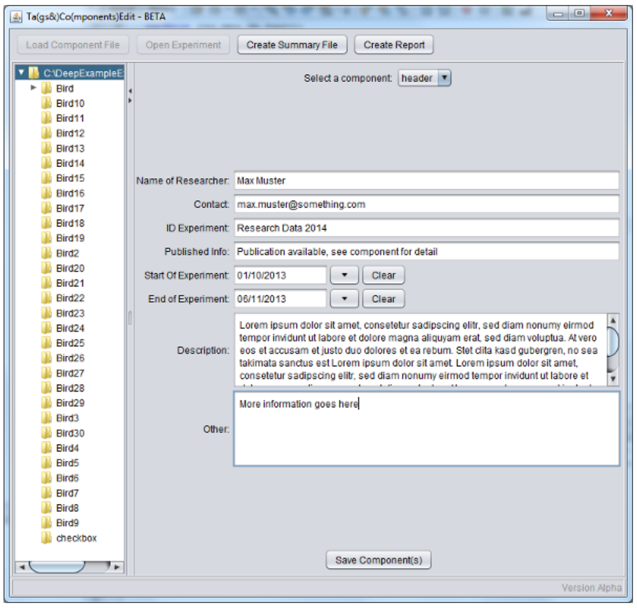
## TacoEdit

With TacoEdit, a user can create and edit metadata (cf. Fig. 2). TacoEdit is a desktop application which requires Java 7 and should run on any modern desktop system (Windows, Mac OS X, Linux). Your system administrator can help you installing and starting TacoEdit. A typical workflow with TacoEdit looks like this:

- Start the TacoEdit application
- Open a configuration file
- Open an experiment
- Assign a header component to the experiment top folder

*Tel.: +49(89) 3299-01, e-mail: benutzerberatung@rzg.mpg.de, URL: http://www.rzg.mpg.de/
**Editor: Renate Dohmen**

- Assign as many components as you like to subfolders of the header folder
- Create a summary file of your metadata and send it via e-mail to your system administrator
- Optional: create a report as overview of the metadata you have created



Select a component: header

Name of Researcher: Max Muster

Contact: max.muster@something.com

ID Experiment: Research Data 2014

Published Info: Publication available, see component for detail

Start Of Experiment: 01/10/2013

End of Experiment: 06/11/2013

Description: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et

Other: More information goes here

Figure 2: The desktop application TacoEdit used to enter the metadata values. Here the user has selected to specify the 'header' component which consists of tags such as 'Name of Researcher', 'Contact', etc.

## TacoHarvest

After the user has created metadata components, the second part of the tool chain can be applied to the folder hierarchy. It scans the folder recursively and extracts the metadata, compiling a tree of metadata components which typifies the whole metadata of a project. This tree of metadata can be exported to various formats, for example HTML. In addition, TacoHarvest can automatically generate a relational database schema from the predefined list of components and insert the concrete metadata via SQL statements. This database is the basis for the third part of the tool chain, TacoBrowse.

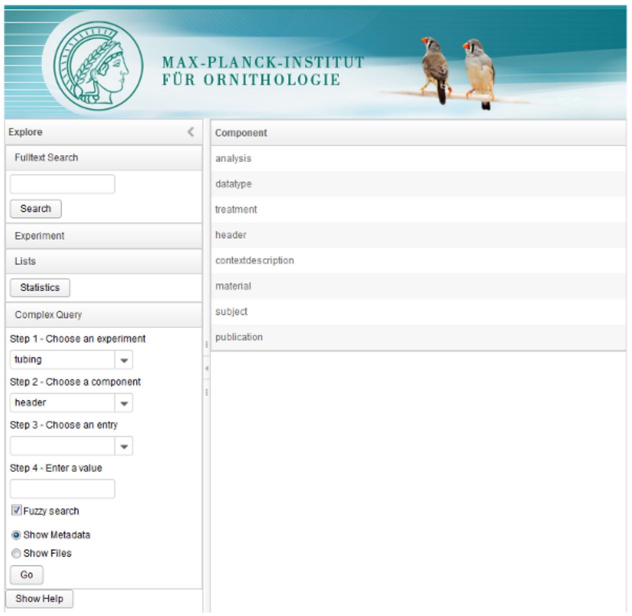
TacoHarvest consists of several command-line applications and needs a Java 7 installation. The basic usage is:

```
java -cp TacoHarvest.jar \
  de.mpg.rzg.tacoharvest.<application> <params>
```

where *application* is the application you want to execute, followed by some (optional) parameters.

## TacoBrowse

TacoBrowse is a web application for displaying and exploring (searching and browsing) the metadata (Fig. 3). For Max Planck Institutes the RZG offers to host this service.



Step 1 - Choose an experiment: tubing

Step 2 - Choose a component: header

✓ Fuzzy search

● Show Metadata

Figure 3: TacoBrowse, the web interface to search and explore the metadata

## Further information

The Taco system is licensed under LGPL. You can download the source code, precompiled Jar files, administrator and user manual and example configurations (Dublin Core) from the TACO download section. A list of frequently asked questions (FAQ) is also available. Further or specific questions can be asked at taco@rzg.mpg.de.

# HPC System Hydra

Ingeborg Weidl and Markus Rampp

## System availability and usage statistics

Comprehensive live statistics about the availability of the *hydra* system and usage details, such as number of jobs, job sizes, and GPU usage, have been made available for Max Planck users on the RZG website (access restricted to MPG subnets and authenticated users). The website shows the total number of available cores and GPUs and the distribution of job sizes (nodes, cores, GPUs) as functions of time, with a daily, weekly and monthly resolution. An example is shown in Fig. 4.

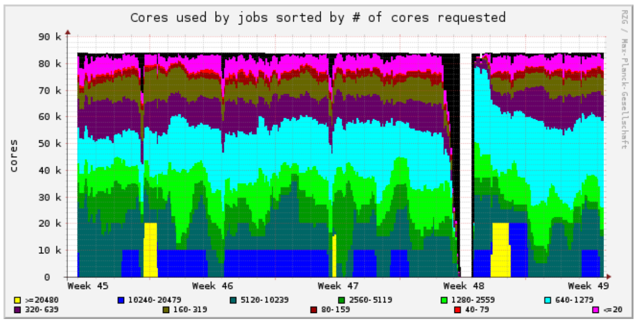


Figure 4: Availability (max: 82k cores) and usage (color-coded by the requested number of cores) of hydra during November 2014

## CPU and memory usage of jobs

Note that after completion of the phase II in autumn 2013 the majority of *hydra's* compute nodes offer 20 physical cores (2 x CPU Intel Xeon E5-2680v2, Ivy Bridge). Accordingly, the product of the `tasks_per_node` and `ConsumableCpus` requirements should equal 20 (or 40, if hyperthreading is desired). Thus, the relevant settings in a LoadLeveler batch script should read, for example

```
# @ node = 32
# @ tasks_per_node = 20
# @ resources = ConsumableCpus(1)
```

for a plain MPI job or

```
# @ node = 32
# @ tasks_per_node = 2
# @ resources = ConsumableCpus(10)
```

for a hybrid MPI/OpenMP job with 10 OpenMP threads per MPI task. The RZG webpage provides complete sample scripts for the LoadLeveler batch system.

Jobs which request less than 16 nodes are predominantly scheduled on the smaller Sandy Bridge partition (phase I) of *hydra* with 16 physical cores (2 x CPU Intel Xeon E5-2670, Sandy Bridge) per node. Hence, for such jobs it is recommended to request resources equivalent to 16 instead of 20 cores per node.

In order to detect underutilization of the CPU and/or memory issues, the RZG recommends to routinely check the CPU and memory utilization statistics of applications which are reported to standard output for each batch job. The following example shows the respective output of a job which utilized only a quarter of the CPU resources (possibly due to I/O and communication overhead), but stayed well below the 56 GB memory limit per node.

```
----------------------------------------------
Statistics for LoadLeveler job ll1-ib0.6666.0:

number of nodes: 1792
number of MPI tasks: 1792
number of OpenMP threads per task: 20

memory consumption(high water mark): 2383.4 GB
maximum memory per node: 1.33 GB
minimum memory per node: 1.11 GB
average memory per node: 1.17 GB

average CPU usage: 25.2 %
```

In addition users can collect communication (MPI) statistics such as communication patterns, message sizes, load imbalances by linking (or preloading) the mpitrace profiling library (see `module help hpct` and the Bits&Bytes article Performance Analysis with IBM HPC toolkit).

## New compiler and module defaults

With the latest maintenance of *hydra* (Nov. 26th, 2014) new compiler and module defaults were introduced. Most importantly the following defaults became effective:

- Intel compilers (C, C++, Fortran): **intel/14.0** (was: intel/13.1)
- Intel MKL (BLAS, LAPACK, ScaLAPACK, ...): **mkl/11.1** (was: mkl/11.0)
- GCC (C, C++, Fortran, ...): **gcc/4.8** (was: gcc/4.7)
- Nvidia CUDA for GPUs: **cuda/6.5** (was: cuda/5.5)

Dependent module software and libraries were rebuilt by the RZG using the new default compilers. In principle, all statically and dynamically linked user codes should continue to work after the update. Recompilation with the new default compilers is recommended, however.

In order to keep the list of modules as current and concise as possible a number of modules referring to outdated software were removed or marked as deprecated. The underlying software installation directories were not touched, however, and hence all codes which link such libraries continue to work provided either static or dynamic linking with `-rpath` was chosen. For reasons of reproducibility the latter methods should be preferred over setting the LD_LIBRARY_PATH at runtime, which would fail, e.g., if the LD_LIBRARY_PATH is derived from a module which was removed.

### Notes on the floating-point accuracy of Intel compilers

Intel compilers tend to adopt increasingly more aggressive defaults for the optimization of floating-point semantics. The current default is `-fp-model fast=1`. It is recommended to double-check the accuracy of simulation results by using more conservative settings (which might, however, come at the expense of computational performance) like `-fp-model precise` (which is usually a good compromise between performance and accuracy) or even the most conservative setting of `-fp-model strict`. More details can be found in compiler man pages (`man icc`, `man ifort` or `man icpc`) and in the Intel overview presentation Floating point accuracy and reproducibility.

### Ivy-Bridge specific instructions with Intel/14.0 compilers

The new default compiler, intel/14.0 (see above) is able to generate instructions which are specific to the Ivy Bridge micro architecture and which cannot be executed on Sandy Bridge CPUs. Although hardly relevant for most HPC applications, such instructions might be inadvertently generated if the `-xHost` compiler option is used during compilation on one of the *hydra-i* nodes (which host Ivy Bridge CPUs). This can cause the executable to fail if it gets executed on the Sandy Bridge partition of *hydra*. Unless the generation of Ivy-Bridge specific instructions is explicitly desired users are advised to either compile on the *hydra* login nodes (which host Sandy Bridge CPUs) and/or use the `-xavx` compiler option instead of `-xHost`. In any case the code can safely execute on both, the Sandy Bridge *and* the Ivy Bridge partition of *hydra*.

# Virtual Hosting Environment

Florian Kaiser

Extending the service hosting portfolio, the RZG operates a virtual hosting environment based on VMware vSphere. This approach allows for a quicker and more flexible allocation of resources (virtual machines) for small to medium-sized projects.

Currently, the platform is hosting a variety of services on more than 100 virtual machines. These range from services operated by the RZG like the GIT hosting service over services of the IPP (various web sites and web services) to projects by various Max Planck Institutes and national and international data projects like CLARIN, DARIAH, EUDAT and RDA Europe.

For the virtual machines, the user has the choice of either SuSE Linux Enterprise Server or Scientific Linux as the operating system. Generally, the operating system is managed and kept up-to-date by the RZG, while the user / service admin is granted root access to deploy and manage their service. Working together with other groups at the RZG, assistance can be offered each step of the way.

Note however, that the service is not intended as a self-service cloud hosting platform, but rather to offer individual solutions tailored to each project. Requirements are gathered together with the user, based on which individualized VMs are created and deployed.

To request resources on the virtualization environment, open a ticket in the RZG helpdesk.

# Extreme Scaling and Visualization of HPC Applications

Andreas Marek, Elena Erastova, and Markus Rampp

The RZG contributed two high-performance computing (HPC) codes of the Max Planck Society to the Extreme Scaling Workshop 2014 which was organized by the Leibniz Supercomputing Centre (LRZ) in summer 2014. Both codes have been parallelized and optimized in close collaborations of the HPC applications team of the RZG and Max Planck researchers. The supernova code VERTEX (Max Planck Institute for Astrophysics, Garching) achieved scalability up to the entire *SuperMUC* system (about 150,000 cores) and showed roughly 0.5 PFlop/s of sustained computing performance. With the DNS code NSCOUETTE (originally developed at the Max Planck Institute for Dynamics and Self-Organization, Göttingen) up to half of the *SuperMUC* system (64,000 cores) could be utilized with reasonable efficiency. For both codes scientifically relevant setups were used and newly implemented functionality was successfully tested. The results are summarized in the autumn 2014 issue of the 'Inside

Online' magazine of the Gauss Centre of Supercomputing (GCS).

Using the remote-visualization facilities of the RZG a number of complex visualizations of simulation results obtained with both codes were recently completed by the RZG visualization team (cf. Figs. 5 and 6).

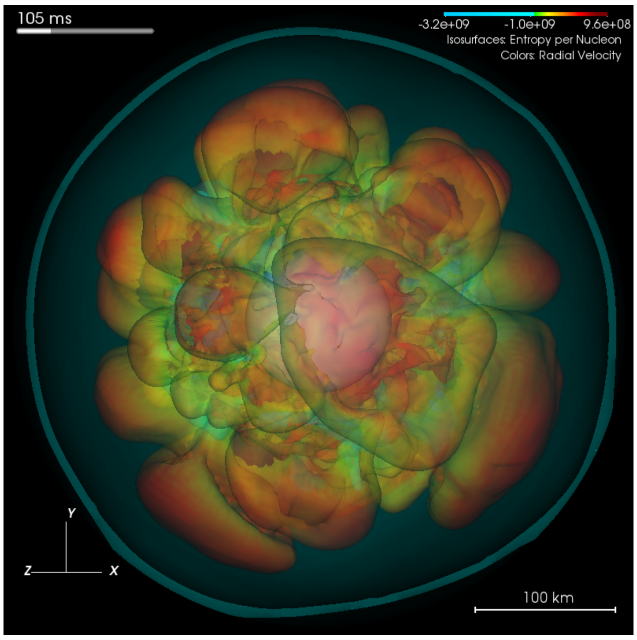


Figure 5: Visualization of a type-II supernova simulation performed with the VERTEX code

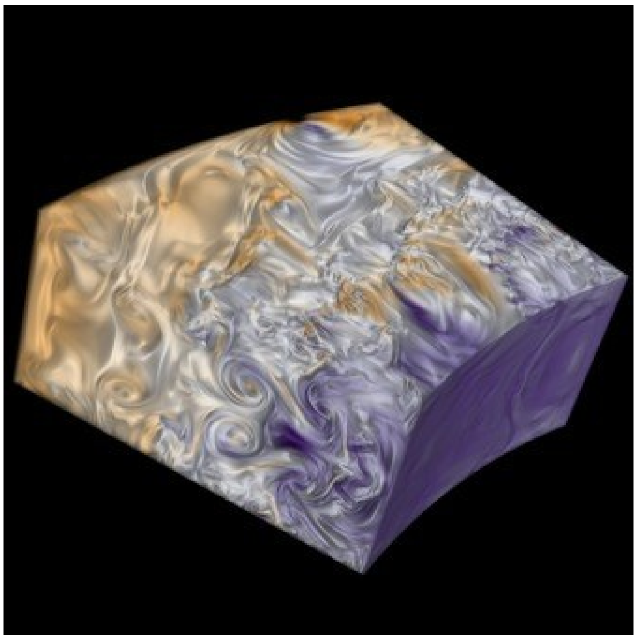


Figure 6: Visualization of turbulence in a rotating Keplerian disc performed with the NSCOUETTE code

## Overview on Events

Hermann Lederer

### Software training

The RZG has initiated that annual software training events take place for Max Planck scientists with respect to both Intel software (related to the Max Planck - Intel software license agreement with more than 30 participating institutes) and Nvidia GPU programming (related to the Max Planck supercomputer which is equipped with 676 Nvidia K20X GPUs). The Nvidia trainings are scheduled to be held regularly in spring at the RZG; the previous one took place in May 2014, the 2015 dates are in preparation. The Intel software training was initiated in 2012 at the RZG, and following up several courses have been given at different Max Planck Institutes, the latest one from October 22nd to 23rd at the MPI for Solid State Research in Stuttgart. Dates and locations for 2015 will be announced.

### HPSS User Forum

The annual HPSS User Forum of the worldwide HPSS user community took place in Munich in the last week of October 2014, organized by the RZG. HPSS (High Performance Storage System) is a highly scalable, cluster-based, hierarchical mass storage software solution designed to manage and access large amounts of data at high data rates. It is used by many of the worldwide leading scientific research centers. Around 60 participants from 29 institutions in three continents gathered for five days to share their experience with HPSS, present their current activities and future challenges, discuss open issues and hear from vendors about upcoming new technologies.

### International HPC Summer School

The International HPC Summer School on Challenges in Computational Sciences, which started in 2010 as a US European event, has been extended to include Japan and Canada. In 2014 it took place in Budapest with 80 participants (PhD students and post-docs), from which 30 came from European institutions and 4 from the Max Planck Society. The 2015 HPC Summer School will take place in Toronto from June 21st to 26th, hosted by 'Compute Canada'. A call for applications is planned for end of January 2015 and will then also be published on the RZG web page.