Star Clusters: From the Milky Way to the Early Universe
Proceedings IAU Symposium No. 351, 2019
A. Bragaglia, M. Davies, A. Sills & E. Vesperini, eds.

doi:10.1017/S1743921319007452

BEANS – distributed data analysis for numerical simulations

Arkadiusz Hypki[1], Mirek Giersz[2], Abbas Askar[3], Diogo Belloni[4,5] and Agostino Leveque[2]

[1]Astronomical Observatory Institute, Faculty of Physics, A. Mickiewicz University, Słoneczna 36, 60-286 Poznań, Poland
email: ahypki@amu.edu.pl

[2]Nicolaus Copernicus Astronomical Centre, Polish Academy of Sciences, Warsaw, Poland
email: mig@camk.edu.pl

[3]Lund Observatory, Department of Astronomy, and Theoretical Physics, Lund University, Box 43, SE-221 00 Lund, Sweden
email: askar@astro.lu.se

[4]National Institute for Space Research, São José dos Campos, Brazil
email: diogo.belloni@inpe.br

[5]Instituto de Física y Astronomía, Universidad de Valparaíso, Valparaíso, Chile

Abstract. BEANS is a tool for distributed data analysis. It provides web and command line interface for data analysis and plotting for huge datasets. BEANS is written in a general form and can be used in any field of research to analyze the data. The main purpose of BEANS is to provide to the community a versatile tool to store, analyze and then visualize any amount of scientific data (e.g. numerical simulation, observations).

Keywords. methods: data analysis, numerical, statistical – astronomical data bases: miscellaneous

1. Introduction

The amount of data which is produced e.g. by today's numerical simulations, can be quite substantial. The same applies to the amount of observational data. Thus, there is a strong need to present some alternative way to store huge amount of data, analyze it in a distributed way (using many independent nodes) and then extract some useful knowledge from the data. BEANS† software tries to solve the problem of managing huge amount of data (in terms of disk space) but also the problem of analyzing data coming from many different simulations (e.g. thousands of different numerical simulations, each consisting of multiple files). Managing that amount of information is not a trivial task.

BEANS was intended to help in managing and analyzing the data coming from numerical simulations done with the MOCCA code. The MOCCA code is currently one of the most advanced codes used to perform full dynamical and stellar evolution of real-size globular clusters (GCs) (Giersz *et al.* 2013 and Hypki *et al.* 2013)‡. It follows the evolution of GCs closely to NBODY codes but it is much faster. In this way, in the same amount of time, one can obtain results for many numerical simulations for various initial conditions. One simulation of a moderate size GC takes around 10-20 GBs and consists of about 20 files. Currently, there are over 3500 MOCCA simulations available for the analysis. BEANS is

† www.beanscode.net
‡ www.moccacode.net

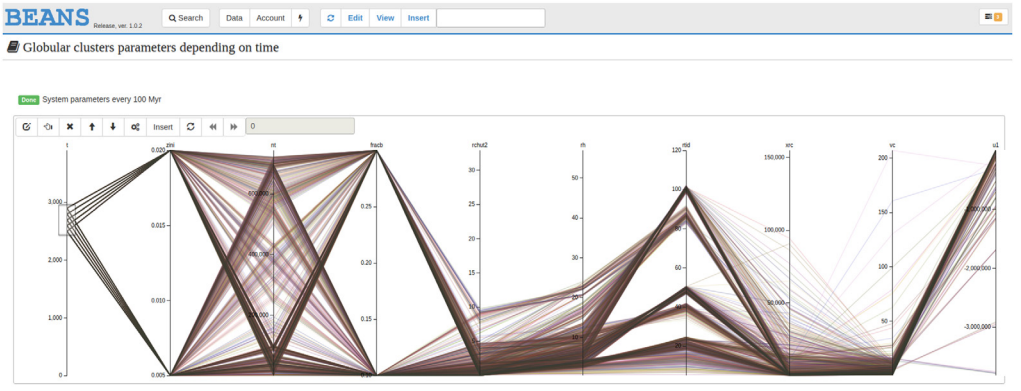

Figure 1. An example plot which demonstrates how one can join on one plot parameters of many MOCCA simulations for every 100 Myrs. The plot is interactive and one can use it to examine how global parameters of globular clusters change with time.

currently used to manage the MOCCA data. However, it is a general tool and can be used for data analysis in any field of research as long as the data can be presented in a tabular form.

2. Overview

BEANS software is built on the top of the tools which already proved their value in the industry. BEANS as the main database uses Apache Cassandra† – decentralized, automatically replicated, linearly scalable, fault-tolerant with tunable consistency open source database written in Java. BEANS uses Elastic‡ as an open source, distributed, full-featured search engine. For distributed data analysis BEANS uses Apache Hadoop¶ platform (Dean *et al.* 2008). Apache Hadoop provides low level approach for data analysis. Thus, BEANS uses Apache Pig as the main language for data analysis because it is a scripting language which internally transforms the scripts into series of low-level jobs run on the Apache Hadoop.

BEANS software organizes data into datasets with tables. A dataset is just a series of tables (like database in SQL terms). A table is a series of rows consisting of the same number of columns (like table in SQL terms). Any table can have any number of columns and any number of rows. Additionally, BEANS can handle any number of datasets with any number of tables per one dataset.

Typical data analysis in BEANS is done in a form of notebooks – a scrollable series of scripts, plots and other entities which provide self-consistent, compact and step-by-step analysis on a scientific subject.

More details about BEANS software, its features and capabilities one can find in Hypki (2018).

3. Examples

Examples are the best way to show the power of BEANS and all the components BEANS is using internally (e.g. Apache Pig).

First example is shown in the Fig. 1. It is an interactive plot with a number of star cluster global properties (e.g. mass, initial N, binary fraction, core radius, central potential) on every 100 Myrs for hundreds of simulations with different initial conditions. The script

† http://cassandra.apache.org/
‡ https://www.elastic.co/
¶ http://hadoop.apache.org/

https://doi.org/10.1017/S1743921319007452 Published online by Cambridge University Press

which was used to produce the data needed by Fig. 1 consists of 6 lines and it is written in Pig Latin language (Apache Pig project).

```
rows = LOAD 'MOCCA/system' using UniTable();

rows = FILTER rows BY tphys < 14000;

rows = FOREACH rows GENERATE tbid, nt, DSPARAM(DSID(tbid), 'zini') as zini,
DSPARAM(DSID(tbid), 'fracb') as fracb, rchut2, r\_h as rh, rtid, xrc, vc, u1,
irc, tphys, FLOOR(tphys, 100.0) as t, tphys - FLOOR(tphys, 100.0) as diff;

rowsGrouped = GROUP rows BY (t, tbid);

rowsFlat = FOREACH rowsGrouped GENERATE group.tbid as tbid, group.t as t,
FLATTEN(rows), MIN(rows.diff) as minDiff;

rowsFlat = FILTER rowsFlat BY tphys - t $<$ minDiff + 0.001;

STORE rowsFlat into 'NAME "System every 1Gyr" SIMPLIFY' using UniTable();
```

In the first line of the script the data is read from all datasets which contain "MOCCA" word and from tables which contain "system" (in e.g. name, description). These are the files in MOCCA simulations which contain global parameters of GCs. The next line filters all rows and leaves only those which have time less then 14 Gyrs. The next line extracts some global values for GCs, some initial parameters, and a difference between real timestep and the closest 100 Myr threshold. The next line groups all rows over two parameters: time and ID of a table (every table has it own unique ID). Next, the rows are aggregated for each of the groups and minimal time difference between real time of a row and the closest 100 Myr threshold is computed. This is needed for the next line to filter the rows and leave only those which are closest to the 100 Myr steps. The last line of the scripts saves the data for the plot.

The script shows how one can analyze the data coming from many MOCCA simulations in a powerful way using just a few commands. The script produces the data which later on create an interactive plot. This allows to study in details how the globular clusters' parameters evolves with time (every 100 Myr). This is an example how one can create very informative, clean and compact plot which allows to search for some relations among multiple parameters on one plot.

The next example is presented below and shows how in several lines one can query all of stellar evolution files coming from MOCCA simulations in order to find binaries which consisted of main sequence – compact object and became after the stellar evolution step a double compact binary.

```
binEvol = LOAD 'MOCCA/binevol' USING UniTable();

-- check if before the interaction we have: compact-MS binary
binEvol = FILTER binEvol BY (mocca_GetOldStarType(id1i, *) <= 1
AND mocca_GetOldStarType(id2i, *) >= 10)
OR (mocca_GetOldStarType(id2i, *) >= 10
AND mocca_GetOldStarType(id1i, *) <= 1);

-- check if after the interaction we have: compact-compact
```

https://doi.org/10.1017/S1743921319007452 Published online by Cambridge University Press

```
binEvol = FILTER binEvol BY mocca_GetNewStarType(id1i, *) >= 10
AND mocca_GetNewStarType(id2i, *) >= 10;

binEvol = FOREACH binEvol GENERATE timevo as tphys, id1i AS id1, id2i AS id2,
mocca_GetOldSep(id1i, *) AS oldSep, mocca_GetNewSep(id1i, *) as newSep,
mocca_Summary(id1i, *) as summary;

STORE binEvol INTO ' NAME "compact binaries from binevol interactions"
SIMPLIFY ' USING UniTable();
```

In the first line of the script the data is read from all "MOCCA" simulations from tables containing "binevol". The next line filters all rows and leaves only those which contain binaries which initially contained some main sequence – compact object binaries. The script uses here several User Defined Functions (UDF) to make the script easier to understand. For example, UDF mocca_GetOldStarType(id1i, *) reads several columns from the given row and determines what is the stellar type of a star under the column *id1i* (first binary component). The other UDFs should be self-explanatory. The next line filters out the rows which do not have double compact object binaries after the interaction. The forth line extracts some example parameters of binaries (e.g. old and new semi-major axes of binaries) and the last line saves the results for further analysis.

4. Conclusions

BEANS software is an innovative way to analyze terabytes of data from MOCCA simulations. The already existing scripts are used to search for correlations in the data coming from thousands of MOCCA simulations, to build histories of exotic objects (e.g. blue stragglers, black holes), to search for dynamical interactions for a specific types of stars or binaries and many more. BEANS software becomes a central place to store huge amount of data, a repository for various scripts written in different languages (e.g. Apache Pig, AWK) and a place for easy to understand and very powerful set of notebooks which provide analysis on some astrophysical subjects.

Acknowledgements

AH was supported by Polish National Science Center grant 2016/20/S/ST9/00162. MG and AL were partially supported by the Polish National Science Center (NCN) through the grant UMO-2016/23/B/ST9/02732. AA is currently supported by the Carl Tryggers Foundation for Scientific Research through the grant CTS 17:113. DB was supported by the grants #2017/14289-3 and #2018/23562-8, São Paulo Research Foundation (FAPESP).

References

Giersz, M., Heggie, D. C., Hurley, J. R., & Hypki, A. 2013, *MNRAS* 431, 2184
Hypki, A. & Giersz, M., 2013, *MNRAS*, 429, 1221
Hypki, A. 2018, *MNRAS*, 477, 3076
Dean, J. & Ghemawat, S. 2008, *Commun. ACM*, 51, 107