Future Generation Computer Systems 29 (2013) 582–590

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Cloud MapReduce for Monte Carlo bootstrap applied to Metabolic Flux Analysis

Tolga Dalman [a], Tim Dörnemann [b], Ernst Juhnke [b], Michael Weitzel [a], Wolfgang Wiechert [a], Katharina Nöh [a,*], Bernd Freisleben [b]

[a] Institute of Bio- and Geosciences 1: Biotechnology 2, Forschungszentrum Jülich, Wilhelm-Johnen-Straße, D-52428 Jülich, Germany

[b] Department of Mathematics & Computer Science and Center for Synthetic Microbiology, University of Marburg, Hans-Meerwein-Straße 3, D-35032 Marburg, Germany

ARTICLE INFO

*Article history:*
Received 7 March 2011
Received in revised form
27 July 2011
Accepted 17 October 2011
Available online 4 November 2011

*Keywords:*
Metabolic Flux Analysis
Cloud computing
Scientific workflows
Hadoop
MapReduce
Monte Carlo bootstrap

ABSTRACT

The *MapReduce* architectural pattern popularized by Google has successfully been utilized in several scientific applications. Up until now, *MapReduce* is rarely employed in the field of Systems Biology. We investigate whether a *MapReduce* approach utilizing on-demand resources from a Cloud is suitable to perform simulation tasks in the area of Metabolic Flux Analysis (MFA). An Amazon ElasticMapReduce Cloud implementation of the parallel, parametric Monte Carlo bootstrap in the context to $^{13}$C-MFA is presented. The seamless integration of the application into a service-oriented, BPEL-based scientific workflow framework is shown. A comparison of a straightforward *MapReduce* implementation using the Hadoop streaming interface on various Amazon *ElasticMapReduce* instance types and a single CPU core computation approach reveals a speedup of 17 on 64 Amazon cores. I/O operations on many small files within the *Reduce* step were identified as the limiting step. By exploiting the Hadoop Java API, making use of built-in data types and tuning problem-specific Hadoop parameters, the I/O issues could be resolved. With the revised implementation, a speedup of up to 48 could be achieved on 64 Amazon cores. To investigate the runtimes of a realistic $^{13}$C-MFA analysis, 50,000 Monte Carlo samples with a typical metabolic network model have been performed on 20 virtual nodes in 24 h and 23 min with a total cost of $384. Our work demonstrates the possibility to perform scalable Systems Biology applications using Amazon's Cloud MapReduce service.



## 1. Introduction

Cloud computing is increasingly demonstrating its benefits for processing large data sets in many scientific application fields. Recent efforts in modern software framework development unite scientific workflow orchestration and high-performance computing (HPC) technologies to establish complex, domain-specific applications. By relying on web service technology, the design of parallel applications to solve time-consuming simulation problems becomes an attractive option.

The focus of our work is to evaluate the benefits of Cloud computing technology in the context of Metabolic Flux Analysis with $^{13}$C-isotope tracers ($^{13}$C-MFA). $^{13}$C-MFA is a modern approach in Metabolic Engineering and Systems Biology enabling model-based estimation of intracellular reaction rates [1]. The overall $^{13}$C-MFA workflow consists of various steps following a classical *model building cycle*: starting from the definition of the model structure, the model is parameterized. Unknown model parameters should be inferable from given measurement configurations (*identifiability analysis*) and if so, the parameters have to be estimated with maximum precision and accuracy (*parameter fitting*). Statistical quality measures then provide a handle to the fitted parameters' certainty. The model is validated by new experimental data typically unraveling a number of model deficiencies. Optimally, by a planned experimental design, this process is repeated until the validation step is considered to be satisfactory. Although seemingly straightforward, $^{13}$C-MFA studies can become complex by two critical aspects:

1. Several tasks and sub-workflows are computationally challenging: due to the nonlinear nature of the parameter estimation problem, quite a number of difficulties may arise, such as convergence to local solutions rather than to a global one, the characteristics of the objective function may be either flat or rugged in the neighborhood of a solution, occurrence of underdetermined models etc. Hence, typically long-running global optimizations relying on heuristics are applied to find the optimal solution.
2. Many tasks need to be executed in an iterative or recursive fashion, and thus, the total number of workflow steps is not known beforehand: for example, non-identifiable parameters

* Corresponding author. Tel.: +49 2461 61 9294; fax: +49 2461 61 3870.
*E-mail addresses:* t.dalman@fz-juelich.de (T. Dalman), doernemt@informatik.uni-marburg.de (T. Dörnemann), ejuhnke@informatik.uni-marburg.de (E. Juhnke), m.weitzel@fz-juelich.de (M. Weitzel), w.wiechert@fz-juelich.de (W. Wiechert), k.noeh@fz-juelich.de (K. Nöh), freisleb@informatik.uni-marburg.de (B. Freisleben).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.10.007

have to be eliminated from the model before parameter fitting. This is to be done by the modeler rather than by a "fixed protocol". Identifiability, however, depends on the parameter values that have yet to be determined by the fitting approach.

Contemporary scientific workflow research projects are addressing these challenges [2] and have inspired the development of a service-oriented scientific workflow framework to manage and organize the modeling cycle for $^{13}$C-MFA [3]. Moreover, to satisfy the specific requirements of $^{13}$C-MFA, particular emphasis has to be put on flexible workflow orchestration, efficient on-demand resource provisioning, and user-friendly access to the entire spectrum of $^{13}$C-MFA applications.

*MapReduce* is a domain-independent programming model for processing data in a highly parallel manner [4]. Recently, several commercial and scientific applications have been realized using the *MapReduce* architectural pattern [5]. Applications such as data mining, image processing, and pattern recognition have successfully used *MapReduce* to solve computationally challenging problems. In the field of Systems Biology, this architectural pattern has rarely been employed. Up to now, applications with *MapReduce* are predominantly restricted to classical bioinformatics tasks, such as phylogenetic clustering or the analysis of DNA sequencing data [6–8].

Amazon offers *ElasticMapReduce* (EMR), a *MapReduce* implementation, as a Cloud computing service [9]. EMR uses the de-facto standard implementation of the *MapReduce* framework, Apache Hadoop MapReduce (henceforth abbreviated as Hadoop). Since EMR interfaces are exposed as (web) services, a simple integration of on-demand resources into our service-oriented scientific workflow framework is possible.

In this paper, an approach is presented to realize the Monte Carlo Bootstrap (MCB) method as part of a BPEL workflow for $^{13}$C-MFA using *MapReduce* in Amazon's Elastic Compute Cloud. MCB is one of the most prominent methods for data analysis and quality assessment of model-based evaluations [10], and hence, is one important building block within the $^{13}$C-MFA modeling cycle [3,11,12]. Being a computationally expensive approach that requires the same calculation steps many times, it is well suited to be realized in a distributed environment with the *MapReduce* programming model.

Our Cloud-based Hadoop implementation of the MCB algorithm is evaluated in two experimental series. Firstly, the runtime behavior of our approach on metabolic network models that vary in size is analyzed. With the conventional definition of speedup (i.e., $T_1/T_p$, where $T_p$ is the runtime on $p$ cores), a value of 48 could be reached on 64 virtual Amazon cores. Secondly, to show that it is possible to scale our solution to even larger problems, a biologically meaningful large-scale and compute-intensive MCB study with 50,000 Monte Carlo samples is conducted. This simulation experiment is computed on 152 virtual Amazon cores in 24 h and 23 min, with a total cost of $384. Thus, by utilizing Hadoop, our Monte Carlo applications can be easily scaled on on-demand Cloud computing resources.

This paper is organized as follows. In Section 2, components of the scientific workflow framework including relevant $^{13}$C-MFA methods and simulation tools, *MapReduce*, as well as the overall software architecture are presented. The MCB method and design considerations for the Hadoop implementation are discussed Section 3. Section 4 covers implementation details of the MCB solution regarding service workflows, Hadoop and Amazon Cloud interfaces. Comparative runtime results are presented in Section 5. Related work from life sciences using the *MapReduce* framework is discussed in Section 6. Section 7 concludes the paper and outlines areas for future work.

## 2. The $^{13}$C-MFA computing architecture

### 2.1. Metabolic Flux Analysis and the simulator 13CFLUX2

Microorganisms convert substrates like sugars into products like amino acids. Understanding and optimizing this process is a challenging part of ongoing research in the field of Metabolic Engineering. Isotope-based Metabolic Flux Analysis is a powerful method for the accurate determination of reaction rates within living microorganisms [1]. Basically, this process consists of two steps:

1. *Carbon labeling experiment*: Substrates labeled with $^{13}$C at specific carbon positions are metabolized by the cells: through a complex network of reactions and driven by metabolic activity, the (isotopic) carbon atoms are distributed within the cell and characteristic labeling patterns emerge in intermediate metabolites. As soon as the labeling is equilibrated, samples are withdrawn from the bioreactor and analyzed. Isotopically labeled fractional enrichments are subsequently quantified with highly accurate measurement devices [13].
2. *Computer-based evaluation*: The measured fractional labeling enrichments are incorporated into an organism-specific network model that describes the fate of all carbon atoms. A nonlinear mathematical model is deduced that relates model parameters to intracellular reaction rates (so-called fluxes) and measurements. The in vivo fluxes are determined by solving an inverse, nonlinear least-squares problem. Finally, the quality of these estimations is assessed using statistical methods [1].

For computer-based evaluation, high-performance simulation tools are readily available that are well-suited for the evaluation of experimental data sets. In particular, the software 13CFLUX2 [14] is used, the successor of the widely established 13CFLUX toolbox [15]. 13CFLUX2 programs are implemented in a modular manner and compiled to run as command-line executables. Graphical interfaces are deliberately separated from the computational core components. Well-defined input/output semantics relying on XML-based documents, *FluxML* and *FWDSIM*, are used for describing and configuring models as well as measurements and for data exchange [14]. Thus, all ingredients are available to easily integrate 13CFLUX2 programs into workflows in order to build automated simulation tasks. Further details on the $^{13}$C-MFA methodology can be found in recent review papers [14–16].

### 2.2. MapReduce, Apache Hadoop and Amazon's Cloud

The *MapReduce* architectural pattern has evolved as a generic, domain-independent processing method for large amounts of data. Two functions, `map` and `reduce`, are required to be implemented by the user with the following prototypes [4]:

$$\texttt{map}\ (k1, v1) \rightarrow \mathit{list}\ (k2, v2)$$

$$\texttt{reduce}\ (k2, \mathit{list}(v2)) \rightarrow \mathit{list}\ (v2).$$

These interfaces are similar to those present in Lisp and other functional programming languages. *list* denotes a list of objects, $k1$ and $k2$ represent key types, $v1$ and $v2$ are value types. The input key/value pairs $(k1, v1)$ are pairwise independent, thus, `map` can be invoked in parallel for all pairs, yielding an intermediate list of mapped $(k2, v2)$ pairs. For each key $k2$, the corresponding values $v2$ are grouped and passed to the `reduce` function, which merges – or reduces – final result values to a list of type $v2$.

The open-source Apache *Hadoop* project has emerged as the de-facto standard implementation for the *MapReduce* programming model [5]. Providing custom `map` and `reduce` functions, *Hadoop* automatically manages parallel execution of these functions on traditional clusters as well as on-demand Cloud infrastructures.