A Hybrid Parallelization Approach for Cloud-enabled Metabolic Flux Analysis Simulation Workflows

Tolga Dalman[1], Michael Weitzel[1], Bernd Freisleben[2], Wolfgang Wiechert[1], Katharina Nöh[1]
[1]Institute of Bio- and Geosciences, IBG-1: Biotechnology 2 Forschungszentrum Jülich GmbH, Germany
[2]Department of Mathematics & Computer Science, University of Marburg, Germany

Abstract

Amazon's Hadoop MapReduce service is a a viable solution for Cloud-enabled scientific applications. In previous work we showed that writing scalable simulation workflows in the field of Systems Biology is easily possible. This work primarily aims at improving the performance of parallel simulations by exploiting the algorithm's structure. A hybrid parallel implementation of a Monte Carlo bootstrap workflow is presented that combines Hadoop MapReduce with shared memory parallelization techniques. Compared to a purely Hadoop-based implementation this hybrid approach is significantly faster while it simultaneously benefits from reduced memory requirements. A microbenchmark indicates 33% faster simulation times and 75% less memory consumption. Our solution is proven to be scalable with a realistic setup on up to 16 virtual Amazon nodes.

1 Introduction

1.1 Scientific Workflows and ^{13}C-MFA

In recent years, ^{13}C isotope-based Metabolic Flux Analysis (^{13}C-MFA) has evolved as a powerful diagnosis tool in Systems Biology for the quantification of intracellular reaction rates in microorganisms. Typical ^{13}C-MFA applications involve nested workflows comprising several intricate experimental and simulation tasks. The advent of high-throughput measurement techniques raise the need to efficiently evaluate the arising datasets. Thus, automation and High-Performance Computing (HPC) are challenging objectives in the context of ^{13}C-MFA. Beside the high-performance simulator 13CFLUX2, a BPEL-based scientific workflow framework that addresses the specific needs of ^{13}C-MFA is currently under development (cf. Figure 1).

1.2 Cloud Computing with Amazon ElasticMapReduce

The architectural design pattern MapReduce has gained interest as easy-to-use HPC technique in commercial and scientific applications. The MapReduce design pattern merely requires an application to provide two functions to the MapReduce framework: **map** and **reduce**. Pairwise independent datasets are processed embarrassingly parallel by the *map* function. Results from *map* are grouped in an application-defined manner and passed to the *reduce* tasks. Pairwise independence of the grouped intermediate results then allows concurrent execution of the *reduce* tasks. Moreover, MapReduce enables developers to concentrate on writing applications instead of implementing typical management tasks, e.g., parallelization, task distribution, data transfer, scheduling, or failure management.

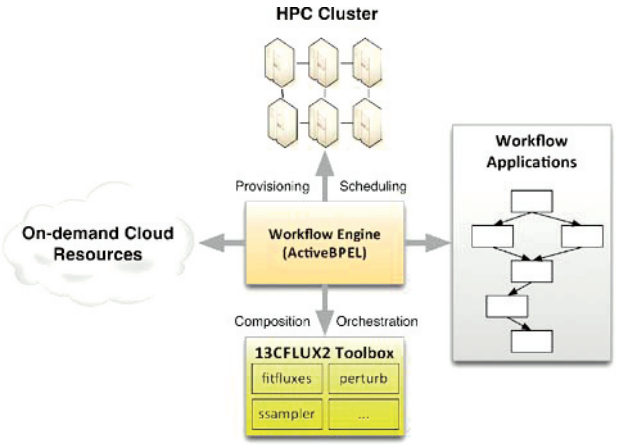

Figure 1: ^{13}C-MFA scientific workflow framework overview

The open-source Apache Hadoop MapReduce framework has emerged as the de-facto implementation of the MapReduce design pattern [1]. Amazon offers the Hadoop MapReduce service *ElasticMapReduce* on top of their virtualized Cloud computing offerings [12, 13]. Hence, by employing service-oriented technologies, integration into our BPEL-based workflow framework is easily possible [2-4].

2 Related Work

Hybrid parallelization paradigms are popular in the field of HPC [6, 7]. Hadoop provides convenient interfaces for Cluster-aware parallelization using the MapReduce architectural design pattern. Several recent publications reveal that scalable parallel MapReduce applications can be easily developed [7-9]. Other MapReduce research focusses on performance improvements, e.g., by exploiting the application's locality behavior or by modifying Hadoop's scheduling algorithms [10,11]. In the following, two applications in the field of bioinformatics are

presented that implement hybrid parallelization with MapReduce.

Qiu et al. provide a hybrid parallelization extension to the Twister MapReduce framework that combines MPI and conventional MapReduce parallelization. The authors present a multi-dimensional large-scale application in the field of metagenomics which can be processed on Cloud computing services [7]. They found out that hybrid parallelization is significantly faster than pure Hadoop MapReduce distributed computing.

With CloudBLAST, Matsunaga et al. provide a Hadoop implementation of the sequence alignment tool BLAST [8]. Their experiments on virtualized and non-virtualized resources reveal that the runtime of CloudBLAST outperforms a MPI version of BLAST (mpiBLAST). On 64 non-virtualized processors, CloudBLAST achieved a speedup of 57, while speedup factor 52.4 is measured with mpiBLAST. Thus, BLAST is easily parallelizable and therefore a perfectly suited Cloud application employing MapReduce or MPI-based distributed computing techniques.

3 ^{13}C-MFA and the Monte Carlo Bootstrap

Monte Carlo-based statistical analysis is an important building block not only within Systems Biology workflows. In previous work, we showed that it is possible to deploy service-enabled on demand MapReduce applications of the Monte Carlo Bootstrap (MCB) algorithm based on Amazon's ElasticMapReduce Cloud services [4]. In the following, we take the ^{13}C-MFA application as model system to describe the Monte Carlo algorithm with the high-performance toolkit 13CFLUX2, a simulator for the evaluation of ^{13}C-labeling experiments. Further details can be found elsewhere [2-5].

3.1 Parallel Monte Carlo Algorithm

Detailed knowledge of the metabolic reaction rates, so called fluxes, are vital to understand the functioning principle of microorganisms. However, the non-observable fluxes are parameters within a large-scale non-linear model that have to be inferred from measurement data by an iterative parameter fitting procedure. In order to interpret the quality of parameter determination, typically a MCB-based statistical analysis is utilized [14].

Here, a set of artificial measurement data is obtained by perturbing the original experimental data. A set of flux distributions is determined by repeating the fitting step with each of the measurement sets. Finally, the distribution of calculated fluxes is interpreted, e.g., by means of confidence intervals. A typical setup comprises *N*, independent parameter optimizations. For each simulation, *M* multi-start optimization processes are performed. Because the fitting steps are pair-wise independent, Monte Carlo bootstrap can be easily parallelized. Reasonable choices for sample sizes are suggested in the literature, i.e., *N* should be in the order of 10^3 or higher and *M* in the order of 10^1 to 10^2 [14].

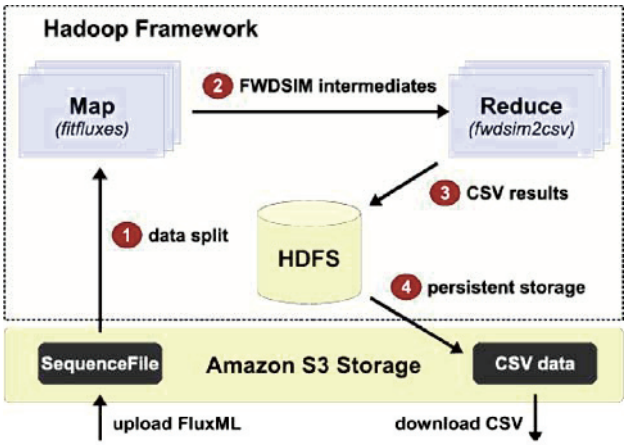

Figure 2: Hadoop MapReduce implementation of the Monte Carlo bootstrap using Amazon's ElasticMapReduce service. Input model data (in the XML-based FluxML format) is serialized and packed into the Hadoop SequenceFile datatype. The Hadoop framework processes the SequenceFile from the persistent Amazon S3 storage (steps 1 to 4). Likewise, CSV output data is copied to the Amazon S3 filesystem after job completion.

3.2 13CFLUX2 – High-Performance MFA Software Toolkit

The high-performance simulation toolkit 13CFLUX2 is a software suite to evaluate and analyze data from ^{13}C-labeling experiments. 13CFLUX2 consists of several highly optimized applications for simulating metabolic reaction networks, estimating non-measurable fluxes, applying statistical analysis tools, and contains several helper tools for data conversion and extraction. The implementation of the ^{13}C-MFA MCB algorithm utilizes the 13CFLUX2 tool *perturb* generating artificial measurements and the parameter estimation program *fitfluxes*. To combine the *N*×*M* XML-based result files, the CSV conversion tool *fwdsim2csv* is employed [2].

4 Design

4.1 FluxHadoop

In our first Java version of the ^{13}C-MFA MCB algorithm (FluxHadoop), *map* is implemented as parameter estimation step invoking the 13CFLUX2 program *fitfluxes*, while *reduce* converts the XML intermediate results to a CSV output format using the *fwdsim2csv* [4]. The number of simultaneous parameter estimation processes per map task is denoted as *k* (i.e., *k=1* for FluxHadoop). Figure 2 depicts the FluxHadoop implementation using Amazon's ElasticMapReduce service. To embed FluxHadoop into larger ^{13}C-MFA workflows, a WSRF service is implemented as glue layer between the Amazon Cloud API and the BPEL workflow framework.

4.2 Hybrid Parallelization Approach

While FluxHadoop schedules $N \times M$ ^{13}C-MFA simulation tasks, the FluxHadoop2 combines Hadoop MapReduce with a shared memory parallelization approach. Thus, only N tasks are scheduled by the Hadoop framework. Each *map* task, implemented by the C++ program *multifitfluxes*, performs M multi-start optimizations, where $k \geq 1$ tasks are executed in parallel. *Multifitfluxes* is built on top of the 13CFLUX2 libraries.

Because *multifitfluxes* emits M flux vectors as HDF5 formatted data, these intermediate outputs need to be converted to CSV results. Hence, the command-line tool *hdf5tocsv* is developed to implement the FluxHadoop2 *reduce* step.

4.3 Hadoop Design Considerations

Multifitfluxes is a multi-process application, thus, all simulation cores of a compute node can be utilized by a single task. However, ^{13}C-MFA parameter estimation is a non-deterministic process, i.e., the runtime of each task is varying. Therefore, to prevent suboptimal load on all cores of a compute node, several *multifitfluxes* tasks are invoked by the Hadoop framework on each compute node.

5 Results

5.1 Hadoop Parameters

More than 200 adjustable parameters are provided to tune and customize the Hadoop framework [1]. We deliberately focus on modifying few parameters, since we do not optimize FluxHadoop and FluxHadoop2 for specific application cases or Hadoop versions. Instead, we strive for a universal framework that works sufficiently fast for compute-intensive ^{13}C-MFA workflow jobs. The adjusted Hadoop parameters are summarized in Table 1, defaults from Hadoop and Amazon are retained otherwise [1, 13].

Parameter	Value
mapred.child.java.opts	-Xmx512m
mapred.map.tasks.speculative.execution	false
mapred.max.split.size	262144
mapred.map.tasks	200
mapred.reduce.tasks	40
mapred.task.timeout	86400000

Table 1: Custom Hadoop parameter settings used for the runtime benchmarks with FluxHadoop and FluxHadoop2

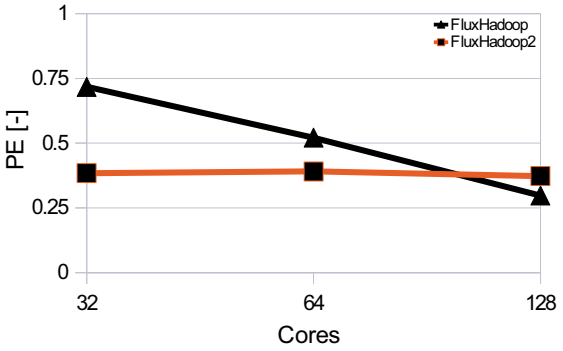

Figure 3: Runtime comparison of FluxHadoop and FluxHadoop2. The parallel efficiency is defined as $PE(P) = T_1 / PT_p$, where T_1 denotes the serial runtime and T_P the parallel runtime on P processing units.

5.2 Runtime and Memory Analysis

To demonstrate the advantages of the hybrid parallel FluxHadoop2 implementation, 20,000 samples of a small-scale network (20 dimensions, 2 model parameters) are simulated on a local two-node cluster (Quad Intel Xeon X7350, 2.93GHz, 128GB RAM) employing eight cores each. To utilize the nodes optimally, four *multifitfluxes* tasks per node are spawned with $k=20$ parallel parameter estimation processes each.

With *FluxHadoop2*, the total simulation time of this microbenchmark is 3 minutes and 44 seconds and 1.3 GB memory are consumed. Compared to the non-hybrid *FluxHadoop* implementation the runtime is decreased by 33% (5:39 minutes) and 75% less memory is required (5.2 GB).

5.3 Scalability Analysis

An experimental series employing a realistic metabolic network model (328 dimensions, 18 model parameters) is conducted. By calling the 13CFLUX2 programs *fitfluxes* and *fwdsim2csv* using a Linux shell script, the serial simulation runtime (T_1) of 20,000 Monte Carlo bootstrap samples is computed on an Amazon virtual node of type High-CPU extra large. Each node has eight CPU cores with 7 GB RAM and 20 EC2 compute units [12]. The total serial runtime T_1 is 302 hours.

Comparing FluxHadoop and FluxHadoop2, the simulation runtime is computed on four (T_{32}), eight (T_{64}), and 16 (T_{128}) virtual Amazon nodes. Three *multifitfluxes* map tasks per node are invoked with $k=8$ parameter estimation processes per task. To reduce the impact of runtime fluctuations on virtualized Cloud resources, the runtime measurements are verified by conducting each simulation at least twice. The runtime results are depicted in Figure 3.

FluxHadoop performs faster than FluxHadoop2 on four and eight virtual nodes. However, the parallel efficiency using P cores ($PE(P) = T_1 / PT_p$) of the new hybrid parallel approach is almost constant (varying between 0.37 and 0.39). On 16 Amazon

nodes, FluxHadoop2 performs 20% faster than FluxHadoop, i.e., 6 hours 20 minutes instead of 7 hours 57 minutes.

6 Conclusion and Outlook

The implementation of applications using the open-source MapReduce Hadoop framework is comparably easy and outstanding flexible [4]. Implementing the MapReduce Monte Carlo bootstrap (MCB) algorithm, Cloud services are seamlessly integrated into compute-intensive ^{13}C-MFA workflows. Moreover, once developed, the distributed Hadoop MapReduce application is easily deployed on a local cluster, or in the Amazon Cloud.

In this work, a scalability improvement over the original implementation of the MCB algorithm is presented. The hybrid parallelization approach is 33% faster than the original implementation in a microbenchmark employing a small-scale model, and up to 25% faster in a realistic simulation setup employing 16 virtual Amazon Cloud nodes. While the 13CFLUX2 programs have a very low memory footprint (at most 20 MB in our experiments), Java-based Hadoop map/reduce tasks require 325 MB. Being a shared memory application, *multifitfluxes* can utilize all compute cores on a node. Thus, less Hadoop tasks need to be spawned and the memory consumption is reduced by up to 75%.

Our experiments show that the parallel efficiency of the hybrid parallel approach is constant, i.e., runtime is halved by doubling the number of employed compute cores. Future work will investigate the low runtime performance on four and eight Amazon nodes of FluxHadoop2 compared to FluxHadoop.

In conclusion, the hybrid parallel ^{13}C-MFA MCB service paves the way to perform serialized statistical evaluations with even larger-scale networks in series. Since Hadoop is a versatile and scalable distributed computing framework, the development of compute-intensive applications is not only restricted to ^{13}C-MFA workflows. Thus, Cloud computing resources provide a vital technology for future Systems Biology applications.

References

[1] T. White, *Hadoop – The Definitive Guide*, O'Reilly Press, 2009

[2] T. Dalman, E. Juhnke, T. Dörnemann, M. Weitzel, K. Nöh, W. Wiechert, B. Freisleben, *Service Workflows and Distributed Computing Methods for ^{13}C Metabolic Flux Analysis*, Proc. of 7th EUROSIM Congress, 2010

[3] T. Dalman, P. Droste, M. Weitzel, W. Wiechert, K. Nöh, *Workflows for Metabolic Flux Analysis: Data Integration and Human Interaction*, Proc. of 4th ISoLA Conference, 2010

[4] T. Dalman, T. Dörnemann, E. Juhnke, M. Weitzel, M. Smith, W. Wiechert, K. Nöh, B. Freisleben, *Metabolic Flux Analysis in the Cloud*, Proc. of 6th e-Science IEEE Conference, 2010

[5] M. Weitzel, *High Performance Algorithms for Metabolic Flux Analysis*, PhD diss., Siegen Univ., 2009

[6] B. Chapman, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*, The MIT Press, 2007

[7] J. Qiu, J. Ekanayake, T. Gunarathne, J. Choi, S-H. Bae, H. Li, B. Zhang, T-K. Wu, Y. Ruan, S. Ekanayake, A. Hughes, G. Fox, *Hybrid Cloud and Cluster Computing Paradigms for Life Science Applications*, BMC Bioinformatics 2010, 11(Suppl 12):S3, 2010

[8] A. Matsunaga, M. Tsugawa, J. Fortes, *CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications*, Proc. of 4th IEEE Conference on eScience, 2008

[9] S. Matthews, T. Williams, *MrsRF: an efficient MapReduce Algorithm for Analyzing Large Collections of Evolutionary Trees*, BMC Bioinformatics 2010, 11 (Suppl 1):S15, 2010

[10] S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, L. Qi, *LEEN: Locality/Fairness-Aware Key Partitioning for MapReduce in the Cloud*, Proc. of 2nd CloudCom IEEE Conference, 2010

[11] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, I. Stoica, *Improving MapReduce Performance in Heterogeneous Environements*, Proc. of 8th USENIX Symposium on OSDI, 2008

[12] Amazon Web Services, http://aws.amazon.com/, last accessed 20-08-2011

[13] Amazon Web Services – Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/, last accessed 20-08-2011

[14] M. Chernick, *Bootstrap Methods: A Guide for Practitioners and Researchers*, 2nd Ed., Wiley Interscience, 2007