Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A cost-effective approach to improving performance of big genomic data analyses in clouds

Christopher Smowton [a], Andoena Balla [b], Demetris Antoniades [b], Crispin Miller [a], George Pallis [b], Marios D. Dikaiakos [b], Wei Xing [a,*]

[a] Cancer Research UK Manchester Institute, University of Manchester, United Kingdom
[b] Department of Computer Science, University of Cyprus, Cyprus

ARTICLE INFO

Article history:
Received 30 June 2015
Received in revised form
7 November 2015
Accepted 10 November 2015
Available online 17 December 2015

Keywords:
Big data
Clouds
Performance

ABSTRACT

With the rapidly growing demand for DNA analysis, the need for storing and processing large-scale genome data has presented significant challenges. This paper describes how the Genome Analysis Toolkit (GATK) can be deployed to an elastic cloud, and defines policy to drive elastic scaling of the application. We extensively analyse the GATK to expose opportunities for resource elasticity, demonstrate that it can be practically deployed at scale in a cloud environment, and demonstrate that applying elastic scaling improves the performance to cost tradeoff achieved in a simulated environment.

1. Introduction

The applications of biological computing are numerous and growing: from patient diagnosis to drug discovery and design, large-scale computing has never been more important in the medical field [1,2]. Historically, biological computing tasks have usually been executed using high-performance computing (HPC) facilities. These facilities offer dependable, homogeneous resources to their users. However, they can be inflexible compared to public cloud computing resources and impose high fixed costs.

Biological applications are well placed to take advantage of the elasticity permitted by a move into the cloud in concert with a suitable elasticity controller. Modern biological applications can be time-sensitive: for example, genetic mutation identification may be applied to determine the appropriate therapy for a particular patient. In this context the user cannot tolerate high or unpredictable latency. Because HPC clusters cannot usually be adaptively resized, it can be difficult to guarantee when the cluster is oversubscribed. Applications may also exhibit varying resource demand at different stages of their execution, which leads to suboptimal performance or wasted resources, depending on whether we provide for its maximum, average or minimum demand. Executing the application in a flexible cloud environment enables dynamic resource provisioning, which can overcome both of these problems.

A key challenge when moving an application into the cloud is to identify its detailed performance characteristics. Whilst in an HPC context applications may be harmlessly overprovisioned, making efficient use of the cloud environment, where resources are hired rather than bought, requires that we identify precisely what the application needs and when it needs it.

In this paper we address this challenge, deploying the Genome Analysis Toolkit (GATK) [3] application as a cloud service, using CELAR [4], an elasticity provisioning platform developed in collaboration between the authors and other partners, and SCAN [5], a prototype tool for executing biological applications under CELAR's governance, developed by the Manchester authors as part of the CELAR project. We have previously described SCAN's design and implementation [5]; here we focus on demonstrating its utility using the GATK as an example application.

We profile the GATK to identify its resource constraints and elasticity requirements, and then employ SCAN to deploy the application in the cloud environment. We describe the algorithms we developed to determine how and when to hire cloud resources, and measure the benefits obtained by using them as compared to a more straightforward cloud execution of the same workload. Our results indicate that moving such an application into the cloud can improve overall application performance and can significantly

* Corresponding author.
E-mail addresses: christopher.smowton@cruk.manchester.ac.uk (C. Smowton), andoena@cs.ucy.ac.cy (A. Balla), danton@cs.ucy.ac.cy (D. Antoniades), crispin.miller@cruk.manchester.ac.uk (C. Miller), gpallis@cs.ucy.ac.cy (G. Pallis), mdd@cs.ucy.ac.cy (M.D. Dikaiakos), Wei.Xing@manchester.ac.uk, wei.xing@cruk.manchester.ac.uk (W. Xing).

http://dx.doi.org/10.1016/j.future.2015.11.011
0167-739X/© 2015 Elsevier B.V. All rights reserved.

reduce the cost. We also describe our plans and rationale for SCAN's future development. The contributions of this work are:

1. We present a method for the effective deployment of the GATK cancer detection application to an elastic cloud environment using SCAN and CELAR, and propose CELAR policy to adaptively scale the number of workers used in response to changing demand on the GATK.
2. We provide an extensive performance analysis and elasticity profile of the GATK, showing how we can harness cloud resources most efficiently.
3. We conduct both simulation and real testbed experiments to examine the benefits of running the GATK in an elastic cloud environment.
4. We examine and justify the possible directions for SCAN's enhancement and further development.

The remainder of this paper is structured as follows: in Section 2 we provide background information on CELAR, SCAN, and biological computing in general. In Section 3 we describe the methods we employed to deploy the GATK in a cloud environment. In Section 4 we describe our efforts profiling GATK in order to identify elastic behaviour. In Section 5 we study the cloud deployment's behaviour, both in a simulated cloud and in a real-world testbed. In Section 6 we describe the enhancement of SCAN's development, including a detailed exploration of what features are desirable and the implementation mechanisms. In Section 7 we discuss the wider lessons learned from working with the CELAR system. In Section 8 we describe related work, before concluding in Section 9.

2. Background

2.1. Genome Analysis Toolkit

The GATK is a collection of tools for the analysis of genetic sequencing information, with a particular focus on discovering differences between the genetic samples taken from a particular individual and the "standard" reference genome [3].

Together with a sequencing machine and pre-processing by other software tools that map the machine's output onto a reference genome, it can be used to determine whether a DNA sample taken from a patient exhibits genetic mutations known to cause cancer, and if it does, which specific variety of the disease. It is important that this analysis is carried out accurately and quickly, as varieties of cancer which appear similar but differ genetically can have very different responses to drugs and other treatments.

The GATK is usually used to construct analysis pipelines, subjecting input data to one GATK tool after another, with each intermediate result written out to disk. In this paper we focus on a particular pipeline that performs quality correction on input sequence data, before determining which locations differ from the reference genome and filtering the result by a standard list of recognised variations. Its input is provided as a BAM file, a bioinformatic file format that describes a series of short DNA (or RNA) sequences that were observed by the sequencing machine; it outputs a Variant Call Format (VCF) file that lists the chromosomes and offsets where the unfiltered variations were observed.

The GATK supports both thread- and process-level parallelism. Thread-level parallelism is provided internally by the GATK, without using any coordinating framework. The GATK's analyses are implemented according to the map-reduce pattern (although it does not support execution within a map-reduce coordinating framework such as Hadoop); the multiple threads are used to parallelise the map phase and/or parallelise a commutative reduce operator. Specific GATK analyses may support both, one, the other or neither form of thread parallelism; in the latter case they do not truly follow the map-reduce pattern as state is carried between sequential calls to the map function. GATK process-level parallelism requires external coordination, typically using the coordinator named Queue that is included with the GATK. It divides a given job into two or more sub-jobs, plus a gatherer job to merge their outputs back together. Process-level parallelism introduces more coordination overhead than thread-level, but can distribute work to multiple machines, and is likely to scale better to large numbers of cores as its parallelism is more coarse-grained.

2.2. CELAR

CELAR is a platform for automatic elastic scaling of cloud applications, developed by the authors and other partners as part of the larger CELAR project. We use CELAR to dynamically scale GATK analysis tasks in a cloud environment. Applications are described to CELAR in terms of their components and the relationships and dependencies between them. It then monitors the application's performance at runtime and adjusts its configuration. This might mean adjusting the application's parameters, or dynamically commissioning or retiring virtual machines (VMs) to adjust the compute resources available.

CELAR includes an intelligent decision-making component, which implements policy to meet targets such as request throughput, latency or running cost [6,7]. In this paper we propose and test appropriate policy to guide the scaling of the GATK application, though in future we intend to use CELAR's ability to adaptively learn constructive policy by observing the application, a possibility explored in more detail in Section 6.

2.3. SCAN

SCAN is a framework for running batch workloads in a cloud environment, with the optional assistance of a human or machine supervisor, such as CELAR, that takes scaling decisions and supplies resources. It was developed by the Manchester authors as part of the CELAR project; it has previously been elaborated in our technical report [5], but we will recapitulate its architecture briefly.

SCAN is designed to bring CELAR's complex analysis of application performance to bear on batch-job-oriented applications, such as most biological tools. To do this, it provides a bridge between CELAR's concept of a persistent, tunable application, and a batch processing system. Fig. 1 shows an overview of its architecture.

SCAN provides one or more work queues (the figure depicts the single-queue case), and manages a pool of worker machines per queue, dispatching tasks to the workers as they become available. Tasks may come from users, or be generated by a previous task. CELAR can dynamically adjust the number of workers associated with each pool. This version of SCAN assigns a single task per worker; however SCANv2 (discussed later in this paper) considers assigning multiple concurrent tasks per worker.

SCAN also provides information to CELAR, on which to base its decisions. It exposes the work rate that is achieved by each work queue, as well as low-level worker metrics, such as their CPU utilisation or I/O operation rate.

SCAN provides several distinct queues, and distinct worker pools, so that we can group tasks with similar performance characteristics together, thus helping CELAR to analyse their characteristics and assign appropriate resources. For example, if the developer deploying a new application against SCAN had reason to suspect that two kinds of tasks with radically different characteristics existed, they would be well advised to assign the two to different queues, so that CELAR can find an optimal configuration for each separately, rather than a compromise between the two.

Aside from the core batch processing system, SCAN also provides shared storage for use by jobs under its control, which our