Digs Documentation

Release 1.0a1

Tycho Marinus <main@tmarinus.nl>, Lucas van Dijk <info@return

April 05, 2016

Contents

1 Contents **1**
1.1 Introduction 1
1.2 Requirements 1
1.3 System Design 2

Bibliography **3**

CHAPTER 1

Contents

1.1 Introduction

In the past few years sequencing the genome has become a lot cheaper, due to next generation sequencing techniques. It is now a lot more viable to sequence the genome of a new patient, and for example compare it to a known reference genome. The human genome consists of three billion base pairs, and some plant genomes are sometimes an order of magnitude larger. So we are dealing with a huge amount of data, and the algorithms for mapping short reads on the reference genome, alignment, or de novo genome assembly can be quite computationally heavy. Furthermore, if you have mapped your reads to a reference genome, you will probably want to perform several kinds of analysis on your newly sequenced genome, for example check if there are any genes different compared to the reference.

Because you retrieve a lot of individual short reads from your sequencing step, you can map and align these reads independently to a reference genome. Thus, there are lot of possibilities for parallel computation. The idea is now to build a distributed system to handle this next generation sequencing pipeline, to perform some of this assembly and mapping algorithms in parallel, distributed over a set of "computational super nodes". This also brings the challenge to efficiently manage the corresponding datasets. Decap et al. already built a similar system using Hadoop *[decap2015halvade]*.

In this assignment we will focus on the distributed system, and initially only implement a subset of the steps instead of the whole NGS pipeline.

1.2 Requirements

In this section we discuss the following two use cases we have in mind for our distributed system: first a use case where the "life science" group in your organisation has performed a sequencing experiment, and you want to reliably store this data, the second use case being the bioinformatics group wanting to perform some analysis on this data.

1.2.1 Use Case 1: store the data of a sequencing experiment

The biological lab work is often done by a different group than the group performing the actual analysis on the data. Although the costs of sequencing have dropped dramatically recently, it's still relatively expensive ($1000 dollar per experiment). The data coming from such experiment should be stored for later analysis.

This brings the following challenges:

- The huge amount of data: a human genome with 60x read coverage depth can occupy easily 200 GB in its compressed FASTQ file format.
- The data needs to be stored persistently and reliably.

- The data needs to be accessible by other teams
- Analysis and other actions need to be performed on this dataset, and the results should be stored too.

1.2.2 Use Case 2: perform analyses on the data

When the data is safely stored in the database, an organisation probably wants to analyse this data. For example, map individual reads to a reference genome or locally align them, assemble a new genome from the individual reads, or check if this newly sequenced genome has any variant genes compared to the reference.

Most of these operations can take a lot of time, but due to the nature of the sequencing experiment (you get a lot of independent reads), it is possible to perform a lot of steps at the same time, but on different chunks of the data. Building a scalable distributed system for these kinds of pipelines could reduce the computational time significantly.

1.2.3 Requirements Prioritisation

Must Have

- Built a distributed system which implements a subset of the steps of a NGS pipeline: Burrows-Wheeler alignment and local alignment on independent reads, keeping the known best practices in mind *[auwera2013fastq]*.
- The data must be stored safely and reliably.
- Fault tolerant, when one of the nodes crashes it should not hinder the final results.

Should have

- Different scheduling policies for different workloads
- Multi-tenancy: let multiple teams perform different actions simultaneously.

Could Have

- Data-ownership: who can see which datasets

1.3 System Design

The general idea is to have a multi cluster architecture with centralized scheduling. There are three kind of nodes:

- Central manager nodes
- Data nodes
- Computational worker nodes

When clients want to store or retrieve data or perform some computational large job, it should first ask the central manager, and from there it receives further instructions.

Bibliography

[decap2015halvade] Decap, D., Reumers, J., Herzeel, C., Costanza, P., & Fostier, J. (2015). Halvade: scalable sequence analysis with MapReduce. *Bioinformatics*, 31(15), 2482-2488.

[auwera2013fastq] Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., del Angel, G., LevyMoonshine, A., ... & Banks, E. (2013). From FastQ data to highconfidence variant calls: the genome analysis toolkit best practices pipeline. *Current Protocols in Bioinformatics*, 11-10.