The Bioinformatics Data Management

Last review date	Reviewer
2009-09-15	Marco Bencivenni
	Enrico Fattibene

Table of Contents

The Bioinformatics Data Management

A transparent mechanism of attaching legacy interfaces to grid I/ O systems.

Introduction

Bioinformatics applications need both high- throughput computing and huge data storage but they were not designed with grid computing in mind, and thus they perform only simple local I/ O and have no facility for attaching to grid data systems. It is necessary to find a way to access data through familiar interfaces without changing applications. In this chapter it is explained how grid computing could be a viable solution to distribute and integrate large bioinformatics databases, and to make these distributed databases usable by legacy bioinformatics programs. To realize this purpose the EGEE data management system has been used for managing, replicating, and locating large database files. Also a customized version of Parrot (a tool for attaching existing programs to remote I/ O systems through the filesystem interface) has been used to connect legacy applications to the EGEE data management system in order to support a logical name space and thus freeing users from managing physical file locations.

Defining a Bioinformatics Namespace

The data files are registered in the EGEE replica management service (RMS) with the appropriate LFNs; and bioinformatics programs are registered tags in the experiment software management service (ESM) on several computing elements:

- databases: `lfn:// genomics_gpsa/ db/ dbname/ dbfiles`
- programs: `genomics_gpsa_program`

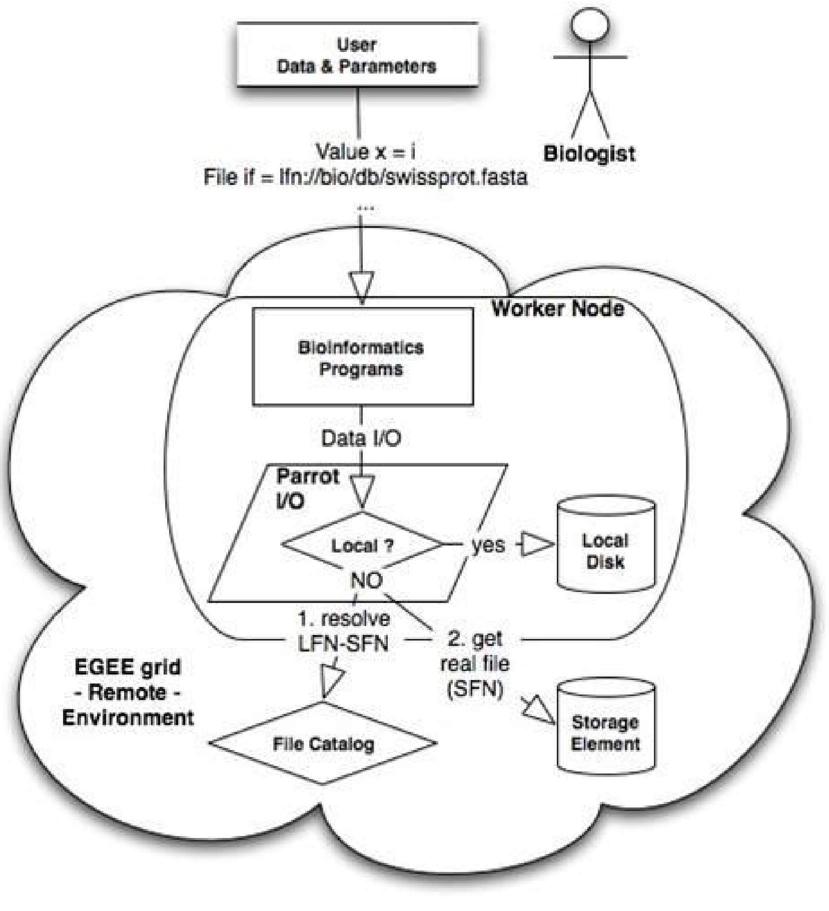
Jobs submitted with LFN and/ or ESM tags within the user’s submission file, will be sent according to matchmaking between these logical names and the free nodes hosting a physical replica.

Virtualizing remote I/ O on EGEE

A customized version of Parrot is an adapted tool for application usage on the EGEE middleware. The main change regards the file namespace understood by Parrot. Code has been added for the recognition of true URLs as the LFN is:

- `protocol:// hostname/ path/ to/ resource`
- `lfn:// genomics_gpsa/ db/ swissprot/ swissprot.fasta`

Parrot is also able to identify LFNs among the program command line arguments, to resolve names to locations (SFNs) and to get the corresponding SFN from a SE as described in the picture below.



Legacy application without remote I/ O

Legacy bioinformatics applications have no I/ O to directly access remote databases. They obtain a remote access to biological database in two ways: file replication on the local computer or remote I/ O. It means that the prerequisite databases have to be downloaded from the remote grid and pushed to the input stream of the program. This remote access is done by an agent separate from the bioinformatics program. This agent is able to:

- resolve grid filename(s) to locations - from LFN(s) to SFNs
- get the data from the best location - with the best protocol
- launch the execution of the program against these downloaded data.

Free storage space on the WN

In case of file replication to the local storage, the agent must make sure that there is enough free space on the local storage area of the computing node. Indeed a biological database may be several hundred megabytes. Thus this WN must be able to store this much data on its local disk because bioinformatics programs need to access the whole database in a single run. This constraint of free space will be also enhanced in the case of a job accessing several databases, or in the case of a WN with multiple CPUs, accessing multiple databases at the same time, but sharing the same storage space. In the case of remote I/ O with Parrot, the data are put directly on the standard input stream of the program, without caching them on the local disk. In some cases of special access (seek, ...), a cache may be needed if the protocol is not supporting such special I/ O primitives. For large files, the remote I/ O mode has the advantage, because (i) in most cases it doesn't have to worry about the free local space and (ii) it is too late to do this checking of the free space when the job is on the worker node. Indeed, it should be done before, at least at the job scheduling time. Thus the file copy mode implies to modify both the information system (to record the free space on WNs) and the scheduling mechanism (to include this “free space” element when the matching is done between the job requirements and the available WNs) whereas the remote I/ O mode works with the current information system and matching workload.