Biology, one of the scientific fields that needs vast amounts of computing power, was one of the first

to take advantage of cloud computing. Molecular dynamics computations are CPU-intensive, whereas

protein alignment is data-intensive.

An experiment carried out by a group from Microsoft Research illustrates the importance of cloud

computing for biology research [223]. The authors carried out an “all-by-all” comparison to identify

the interrelationship of the 10 million protein sequences (4.2 GB size) in the National Center for

Biotechnology Information (NCBI) non redundant protein database using AzureBLAST, a version of the

BLAST23 program running on the Azure platform [223].

Azure offers VMs with four levels of computing power, depending on the number of cores: small

(1 core), medium (2 cores), large (8 cores), and extra large (>8 cores). The experiment used 8 core

CPUs with 14 GB RAM and a 2 TB local disk. It was estimated that the computation would take six to

seven CPU-years; thus, the experiment was allocated 3,700 weighted instances or 475 extra-large VMs

from three data centers. Each data center hosted three AzureBLAST deployments, each with 62 extra large

instances. The 10 million sequences were divided into multiple segments, and each segment was

submitted for execution by one AzureBLAST deployment.With this vast amount of resources allocated,

it took 14 days to complete the computations, which produced 260 GB of compressed data spread across

more than 400,000 output files.

A few observations and conclusions useful for many scientific applications running on Azure were

drawn after a post-experiment analysis. A first observation is that when a task runs for more than

two hours, a message will automatically reappear in the queue requesting the task to be scheduled,

thus leading to repeated computations; a simple solution is to check whether the result of a task has been generated before launching it. Many applications, including BLAST, allow for the setting of some

parameters, but the computational effort to find optimal parameters is prohibitive. A user is also expected

to decide on an optimal balance between the cost and the number of instances to meet budget limitations.

A number of inefficiencies were observed: many VMs were idle for extended periods of time; when

a task finished execution, all worker instances waited for the next task; and when all jobs use the

same set of instances, resources are either under- or over-utilized. Load imbalance is another source

of inefficiency; some of the tasks of a job take considerably longer than others and delay the job’s

completion time.

The analysis of the logs shows unrecoverable instance failures. Some 50% of active instances lost

connection to the storage service but were automatically recovered by the fabric controller. System

updates caused several ensembles of instances to fail.

Another observation is that a computational science experiment requires the execution of several

binaries; thus the creation of workflows, a challenging task for many domain scientists. To address this

challenge, the authors of [215] developed a general platform for executing legacyWindows applications

on the cloud. In the Cirrus system a job has a description consisting of a prologue, a set of commands,

and a set of parameters. The prologue sets up the running environment; the commands are sequences of

shell scripts, including Azure-storage-related commands to transfer data between Azure blob storage

and the instance.

After the Windows Live ID service authenticates the user, it can submit and track a job through the

portal provided by the Web role (see Figure 4.8). The job is added to a table called job registry. The

execution of each job is controlled by a job manager instance that first scales the size of the worker

based on the job configuration; then the parametric engine starts exploring the parameter space. If this

is a test run, the parameter-sweeping result is sent to the sampling filter.

Each task is associated with a record in the task table, and this state record is updated periodically

by the worker instance running the task. The progress of the task is monitored by the manager. The dispatch queue feeds into a set of worker instances. A worker periodically updates the task state in the

task table and listens for any control signals from the manager.

We continue our discussion of biology applications of the Azure infrastructure applied to a loosely

coupled workload for an ensemble-based simulation reported in [224]. A role in Azure is an encapsulation

of an application; as noted earlier, there are two kinds of role: (i) theWeb roles forWeb applications

and front-end code and (ii) the worker roles for background processing. Scientific applications such as

AzureBLAST use worker roles for the compute tasks and to implement their APIs, that provide a run

method and an entry point for the application and the state or configuration change notifications. The

applications use the Blob Storage (ABS) for large raw data sets, the Table Storage (ATS) for semistructured

data, and theQueue Storage (AQS) for message queues. These services provide strong consistency

guarantees, but the complexity is moved to the application space.

Figure 4.9 illustrates the use of a software system called BigJob to decouple resource allocation

from resource binding for the execution of loosely coupled workloads on an Azure platform [224]. This

software eliminates the need for the application to manage individual VMs. The results of measurements

show a noticeable overhead for starting VMs and for launching the execution of an application task

on a remote resource. Increasing the computing power of the VM decreases the completion time for

long-running tasks.