J Grid Computing (2015) 13:561–585
DOI 10.1007/s10723-015-9353-8

Scaling *Ab Initio* Predictions of 3D Protein Structures in Microsoft Azure Cloud

Dariusz Mrozek · Paweł Gosk · Bożena Małysiak-Mrozek

Received: 7 October 2014 / Accepted: 9 October 2015 / Published online: 12 November 2015

Abstract Computational methods for protein structure prediction allow us to determine a three-dimensional structure of a protein based on its pure amino acid sequence. These methods are a very important alternative to costly and slow experimental methods, like X-ray crystallography or Nuclear Magnetic Resonance. However, conventional calculations of protein structure are time-consuming and require ample computational resources, especially when carried out with the use of *ab initio* methods that rely on physical forces and interactions between atoms in a protein. Fortunately, at the present stage of the development of computer science, such huge computational resources are available from public cloud providers on a pay-as-you-go basis. We have designed and developed a scalable and extensible system, called Cloud4PSP, which enables predictions of 3D protein structures in the Microsoft Azure commercial cloud. The system makes use of the Warecki-Znamirowski method as a sample procedure for protein structure prediction, and this prediction method was used to test the scalability of the system. The results of the efficiency tests performed proved good acceleration of predictions when scaling the system vertically and horizontally. In the paper, we show the system architecture that allowed us to achieve such good results, the Cloud4PSP processing model, and the results of the scalability tests. At the end of the paper, we try to answer which of the scaling techniques, scaling out or scaling up, is better for solving such computational problems with the use of Cloud computing.

Keywords Bioinformatics · Proteins · 3D protein structure · Protein structure prediction · Tertiary structure prediction · Ab initio · Protein structure modeling · Cloud computing · Distributed computing · Scalability · Microsoft Azure

1 Introduction

Protein structure prediction is one of the most important and yet difficult processes for modern computational biology and structural bioinformatics [21]. The practical role of protein structure prediction is becoming even more important in the face of the dynamically growing number of protein sequences obtained through the translation of DNA sequences coming from large-scale sequencing projects. Needless to say, experimental methods for the determination of protein structures, such as X-ray crystallography or Nuclear Magnetic Resonance (NMR), are lagging behind the number of protein sequences. As a consequence, the

This work was supported by Microsoft Research within Microsoft Azure for Research Award. A part of the work was performed using the infrastructure supported by POIG.02.03.01-24-099/13 grant: "GeCONiI - Upper Silesian Center for Computational Science and Engineering".

D. Mrozek (✉) · P. Gosk · B. Małysiak-Mrozek
Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
e-mail: dariusz.mrozek@polsl.pl

number of protein structures in repositories, such as the world-wide Protein Data Bank (PDB) [3], is only a small percentage of the number of all known sequences. Therefore, computational procedures that allow for the determination of a protein structure from its amino acid sequence are great alternatives for experimental methods for protein structure determination, like X-ray crystallography and NMR.

Protein structure prediction refers to the computational procedure that delivers a three-dimensional structure of a protein based on its amino acid sequence (Fig. 1). There are various approaches to the problem and many algorithms have been developed. These methods generally fall into two groups: (1) physical and (2) comparative [35, 72]. Physical methods rely on physical forces and interactions between atoms in a protein. Most of them try to reproduce nature's algorithm and implement it as a computational procedure in order to give proteins their unique 3D native conformations [43]. Following the rules of thermodynamics, it is assumed that the native conformation of a protein is the one that possesses the minimum potential energy. Therefore, physical methods try to find the global minimum of the potential energy function [41]. The functional form of the energy is described by empirical force fields [10] that define the mathematical function of the potential energy, its components describing various interactions between atoms, and the parameters needed for computations of each of the components (see Section 2.1 for details). The

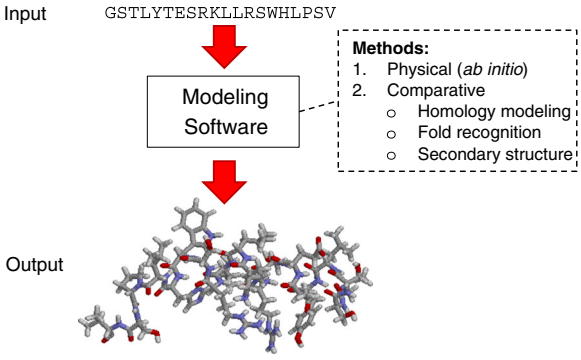

Fig. 1 3D protein structure prediction: from amino acid sequence (Input) to 3D structure (Output). Modeling software uses different methods for prediction purposes: **1** physical, **2** comparative

potential energy of an atomic system is composed of several components depending on the force field type. Methods that rely on such a physical approach belong to so-called *ab initio* protein structure prediction methods. Representatives of the approach include I-TASSER [68], Rosetta@home [42], Quark [69], and many others, e.g. [66] and [73]. In the paper, we will focus on this approach.

On the other hand, comparative methods rely on already known structures that are deposited in macromolecular data repositories, such as the Protein Data Bank (PDB). Comparative methods try to predict the structure of the target protein by finding homologs among sequences of proteins of already determined structures and by "dressing" the target sequence in one of the known structures. Comparative methods can be split into several groups including: (1) homology modeling methods, e.g. Swiss-Model [2], Modeller [14], RaptorX [32], Robetta [36], HHpred [63], (2) fold recognition methods, like Phyre [34], Raptor [70], and Sparks-X [71], and (3) secondary structure prediction methods, e.g. PREDATOR [18], GOR [19], and PredictProtein [56].

Both physical and comparative approaches are still being developed and their effectiveness is assessed every two years in the CASP experiment (Critical Assessment of protein Structure Prediction). It must also be admitted that *ab initio* prediction methods require significant computational resources, either powerful supercomputers [53, 60] or distributed computing. The proof of the latter give projects such as Folding@home [62] and Rosetta@Home [9] that make use of Grid computing architecture.

A promising alternative seems also to be provided by Cloud computing, which allows hardware infrastructure to be leased in the pay-as-you-go model. Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [47]. In practice, Cloud computing allows applications and services to be run on a distributed network using a virtualized system and its resources, and at the same time, allows to abstract away from the implementation details of the system itself. Cloud computing emerged as a result

of requirements for the public availability of computing power, new technologies for data processing and the need for their global standardization, becoming a mechanism allowing control of the development of hardware and software resources by introducing the idea of virtualization. The use of cloud platforms can be particularly beneficial for companies and institutions that need to quickly gain access to a computer system which has a higher than average computing power. In this case, the use of Cloud computing services can be faster in implementation than using the owned resources (servers and computing clusters) or buying new ones. For this reason, Cloud computing is widely used in business and, according to Forbes, the market value of such services will significantly increase in the coming years [46].

1.1 Related Works

Cloud computing evolved from various forms of distributed computing, including Grid computing. Both, Clouds and Grids provide plenty of computational and storage resources, which is very attractive for scientific computations performed in the domain of Life sciences. Therefore, both computing models became popular in scientific applications for which a large pool of computational resources allows various computationally intensive problems to be solved. In the domain of Life sciences there are many dedicated cloud-based and grid-based tools that allow us to solve problems, such as whole genome and metagenome sequence analysis [1], gene detection [45], identification of peptide sequences from spectra in mass spectrometry [44], analysis of data from DNA microarray experiments [33], mapping next-generation sequence data to the human genome and other reference genomes, for use in a variety of biological analyses including SNP discovery, genotyping and personal genomics [13, 57], gene sequence analysis and protein characterization [28], 3D ligand binding site comparison and similarity searching of a structural proteome [24], molecular docking [4, 8], protein structure similarity searching and structural alignment [25, 50–52], and many others.

In the domain of protein structure prediction, it is worth noting two cloud-based solutions. The first one is an open-source Cloud BioLinux [38], which is a publicly accessible Virtual Machine (VM) that enables scientists to quickly provision on-demand infrastructures for high-performance bioinformatics computing using cloud platforms. Cloud BioLinux provides a range of pre-configured command line and graphical software applications, including a full-featured desktop interface, documentation and over 135 bioinformatics packages for applications including sequence alignment, clustering, assembly, display, editing, and phylogeny. The release of the Cloud BioLinux reported in [31] consists of the already mentioned PredictProtein suite that includes methods for the prediction of secondary structure and solvent accessibility (profphd), nuclear localization signals (predictnls), and intrinsically disordered regions (norsnet).

The second solution is Rosetta@Cloud [27]. Rosetta@Cloud is a commercial, cloud-based pay-as-you-go server for predicting protein 3D structures using *ab initio* methods. Services provided by Rosetta@Cloud are analogous to the well-established Rosetta computational software suite, with a variety of tools developed for macromolecular modeling, structure prediction and functional design. Rosetta@Cloud was originally designed to work on the Cloud provided by Amazon Web Service (AWS). It delivers a scalable environment, fully functional graphical user interface and a dedicated pricing model for using computing units (AWS Instances) in the prediction process.

Great progress in scaling molecular simulations has also been brought by scientific initiatives that make use of the Grid computing architecture. Laganà et al. [39] presented the foundations and structures of the building blocks of the *ab initio* Grid Empowered Molecular Simulator (GEMS) implemented on the EGEE computing Grid. In GEMS the computational problem is split into a sequence of three computational blocks, namely INTERACTION, DYNAMICS, and OBSERVABLES. Each of the blocks has different purpose. *Ab initio* calculations determining the structure of a molecular system are performed in the first block. The main efforts to implement GEMS on the EGEE Grid were made to the second block, which is responsible for the calculation of the dynamics of the molecular system. The authors show that for various reasons the parallelization of molecular simulations is possible through the application of a simple parame-

ter sweeping distribution scheme, which is also used in our system.

WeNMR [67] is an example of a successful initiative for moving the analysis of Nuclear Magnetic Resonance (NMR) and Small Angle X-Ray scattering (SAXS) imaging data for atomic and near-atomic resolution molecular structures to the Grid. The WeNMR makes use of an operational Grid that was initially based on the gLite middleware, developed in the context of the EGEE project series [16]. Although, WeNMR focuses on NMR and SAXS, its web portal provides access to various services and tools, including those used for the prediction of biological complexes (HADDOCK [11]), calculating structures from NMR data (Xplor-NIH [58], CYANA [22] and CS-ROSETTA [61]), molecular structure refinement and molecular dynamics simulations (AMBER [6] and GROMACS [65]). A typical workflow is based on Grid job pooling. A specialized process is listening for Grid job packages that should be executed in the Grid, another process is periodically checking running jobs for their status, retrieving the results when ready, and finally, another process performs post-processing of results before they are presented to the user. In the paper, we will present the system that uses a similar workflow scheme, with prediction jobs that are divided into many prediction tasks. These tasks are enqueued in the system for further execution by processing units that work in a dedicated role-based and queue-based architecture that we designed.

Gesing et al. [20] report on a very important security issue while carrying out computations in structural bioinformatics, molecular modeling, and quantum chemistry with the use of Molecular Simulation Grid (MoSGrid). MoSGrid is a science gateway that makes use of WS-PGRADE [15] as a user interface, gUSE [30] as a high-level middleware service layer for workload storage, workload execution, monitoring and others, UNICORE [64] as a Grid resources middleware layer, and XtreemeFS [26] as a file system for storage purposes. Various security elements are employed in particular layers of the MoSGrid security infrastructure with the aim of protecting intellectual property of companies performing scientific calculations.

These projects prove that computations performed in bioinformatics require ample computational resources that are available through the use of Grid or Cloud computing. In the paper, we present a newly developed Cloud4PSP (Cloud for Protein Structure Prediction) system. Cloud4PSP is a cloud-based computational framework for 3D protein structure prediction. Cloud4PSP was designed and developed for the Microsoft Azure commercial Cloud. In the paper, we show the architecture of the Cloud4PSP, its unique features, and the advantages of using the queue-based architecture instead of direct communication. We also present the processing model used by the system for those who would like to extend the system with new prediction methods. Finally, we show the scalability of the system based on the designed architecture on the example of sample prediction processes performed with the use of the *ab initio* prediction method designed by S. Warecki and L. Znamirowski [66].

The paper presents this system, which complements the collection of available Cloud-based and Grid-based tools by:

- sharing a dedicated architecture of a Cloud-based, scalable and extensible system focused on protein structure prediction, oriented to the specificity of the process,
- showing the benefits of building the system with the use of roles and queues,
- providing a processing model that involves the creation of prediction jobs with accompanying collections of prediction tasks,
- delivering a ready-to-deploy Azure solution, whose good scalability in the Azure cloud was proved by a series of tests.

1.2 Microsoft Azure

Microsoft Azure is a commercial cloud platform that delivers services for building scalable web-based applications. Microsoft Azure allows the development, deployment and management of applications and services through a network of data centers located in various countries throughout the world. Microsoft Azure is a public Cloud, which means that the infrastructure of the Cloud is available for public use and is owned by Microsoft selling cloud services. Microsoft Azure provides computing resources in a virtualized form, including processing power, RAM, space and

appropriate bandwidth for transferring data over the network, within the Infrastructure as a Service (IaaS) service model [47]. Moreover, within the Platform as a Service model [47], Azure also delivers a platform and dedicated cloud service programming model for developing applications that should work on the Cloud.

In Fig. 2 we can see types of services and resources provided to a developer of the application deployed to Microsoft Azure cloud. Below we briefly describe the Microsoft Azure cloud resources used in our project:

- Compute: Compute/Cloud Services represent applications that are designed to run on the Cloud and XML configuration files that define how the cloud service should run. The Microsoft Azure programming model provides an abstraction for building cloud applications. Each cloud application is defined in terms of component roles that implement the logic of the application. Configuration files define the roles and resources for an application. There are two types of roles that can be used to implement the logic of the application:
  - **Web role** - is a virtual machine (VM) instance used for providing a web based front-end for the cloud service;
  - **Worker role** - is a virtual machine (VM) instance used for generalized development that performs background processing and scalable computations, accepts and responds to requests, and performs long-running or intermittent tasks.
- Storage: Microsoft Azure also provides a rich set of data services for various storage scenarios. These data services enable storage, modification and reporting on data in Microsoft Azure. The following components of Data Services can be used to store data: **BLOBs** that allow the storage of unstructured text or binary data (video, audio and images), Tables that can store large amounts of unstructured non-relational (NoSQL) data, and the **Azure SQL Database** for storing large amounts of relational data, and HDInsight, which is a distribution of Apache Hadoop, based on the Hortonworks Data Platform.
- Messaging: Messaging mechanisms allow for effective communication between components

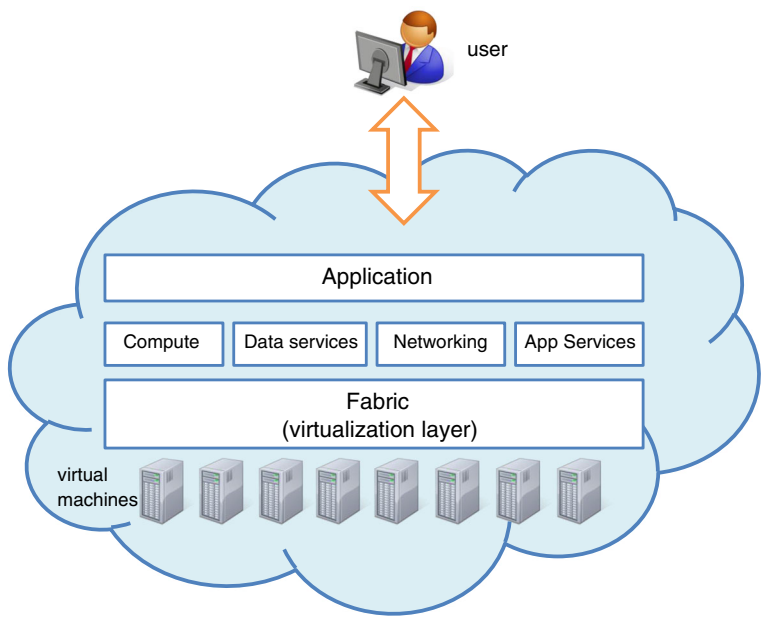

Fig. 2 Application deployed to Microsoft Azure cloud, which serves as a virtualized infrastructure, platform for developers, and gateway for hosting applications

and processes running in the whole cloud service. Queues are a general-purpose technology that can be used for messaging in a wide variety of scenarios, including: communication between web and worker roles in multi-tier Azure applications (as in our project), communication between on-premises applications and Azure-hosted applications in hybrid solutions, communication between components of distributed applications running on-premises in different organizations or departments of an organization. Microsoft Azure provides two types of queue mechanisms: **Azure Queues** and Service Bus Queues. Azure Queues, which are part of the Azure Storage infrastructure, enable reliable, persistent messaging within and between services. Service Bus queues are part of a broader Azure messaging infrastructure that supports queuing as well as other communication patterns.

– Fabric - the entire compute, storage (e.g. hard drives), and network infrastructure, usually implemented as virtualized clusters, constitutes a resource pool that consists of one or more servers (called *scale units*).

The basic tier of the Microsoft Azure platform allows us to create five classes of virtual machines with different parameters and computational power (number of cores, CPU/core speed, amount of memory, efficiency of I/O channel). Table 1 shows a list of features for available computing units.

Microsoft Azure is a commercial Cloud. Costs associated with the use of the Azure platform depend on the number of resources used (including compute units, storage, network traffic and bandwidth), the time by which these resources are used, the number of deployed applications, and the usage of other services that are enabled (e.g., HDInsight, Stream Analytics, Active Directory). The most popular and flexible payment plan is based on pay-as-you-go subscriptions.

2 Methods

Cloud4PSP is an extensible service that allows for *ab initio* prediction of protein structures from amino acid sequences. It was intentionally designed to work in the Azure cloud. At the moment, Cloud4PSP uses only the Warecki-Znamirowski (WZ) method [66] as a sample procedure for protein structure prediction. However, the collection of prediction methods can be extended by using available programming interfaces. The information on possible extensions will be provided in Section 2.4.

In terms of efficiency, the mentioned WZ method is slower than popular optimization methods like gradient-based steepest descent [7], Newton-Raphson [12], Fletcher-Powell [17], or Broyden-Fletcher-Goldfarb-Shanno (BFGS) [59]. However, it finds a global minimum of the potential energy with higher probability and has a lower tendency to converge on and get stuck in the nearest local minimum, which is not necessarily the global minimum of the energy function [66]. The method is based on the modified Monte Carlo approach, which is a better solution. In the section, we give details of the prediction method, architecture and processing model used by Cloud4PSP, and details of how the system is scaled in

Table 1 Available sizes of Microsoft Azure virtual machines (VMs) for Web and Worker role instances [48]

VM/server type	Number of CPU cores	CPU core speed (GHz)	Memory (GB)	Disk space (GB)	Cost ($)/hr
ExtraSmall	Shared core	1.0	0.768	19	0.02
Small	1	1.6	1.75	224	0.08
Medium	2	1.6	3.5	489	0.16
Large	4	1.6	7	999	0.32
ExtraLarge	8	1.6	14	2039	0.64

order to handle a growing amount of work related to the prediction process.

2.1 Prediction Method

The Warecki-Znamirowski (WZ) method for protein structure prediction models a protein backbone by (1) sampling numerous protein conformations determined by a series of ϕ and ψ torsion angles of the protein backbone (Fig. 3), and (2) finding the conformation with the lowest potential energy, which minimizes the expression:

$$\min_{\phi_0,\psi_0,\phi_1,\psi_1,...,\phi_{n-1},\psi_{n-1}} E(\phi_0, \psi_0, \phi_1, \psi_1, ..., \phi_{n-1}, \psi_{n-1}), \quad (1)$$

where: E is a function describing the potential energy of the current conformation, n is the number of amino acids in the predicted protein structure, and ϕ_i, ψ_i are torsion angles of the i^{th} amino acid.

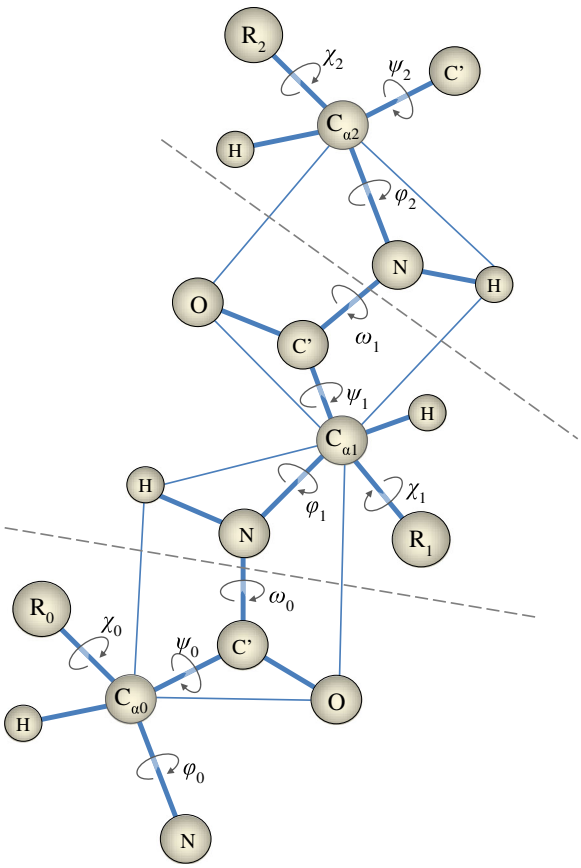

Fig. 3 Overview of protein construction. Visible atoms forming the main chain of the polypeptide. Side chains are marked as R_0, R_1, R_2

The basic assumption of the method is that the peptide bond is planar (see Fig. 3), which restricts the rotation around the $C' - N$ bond, and the ω angle is essentially fixed to 180 degrees due to the partial double-bond character of the peptide bond. Therefore, the main changes of the protein conformation are possible by chain rotations around the ϕ and ψ angles, and these angles provide the flexibility required for folding the protein backbone.

The following potential energy function E is used in the calculations of energy for the current conformation (configuration A^N of N atoms, determined by a series of ϕ and ψ torsion angles in (1)):

$$\begin{aligned} E(A^N) = & \sum_{j=1}^{bonds} \frac{k_j^b}{2}(d_j - d_j^0)^2 + \sum_{j=1}^{angles} \frac{k_j^a}{2}(\theta_j - \theta_j^0)^2 \\ & + \sum_{j=1}^{torsions} \frac{V}{2}(1 + \cos(p\omega - \gamma)) \\ & + \sum_{k=1}^{N} \sum_{j=k+1}^{N} \left(4\varepsilon_{kj} \left[\left(\frac{\sigma_{kj}}{r_{kj}} \right)^{12} - \left(\frac{\sigma_{kj}}{r_{kj}} \right)^{6} \right] \right) \\ & + \sum_{k=1}^{N} \sum_{j=k+1}^{N} \frac{q_k q_j}{4\pi\varepsilon_0 r_{kj}}, \end{aligned} \quad (2)$$

where: the first term represents the bond stretching component energy and: k_j^b is a bond stretching force constant, d_j is the distance between two atoms (real bond length), d_j^0 is the optimal bond length; the second term represents the angle bending component energy and: k_j^a is a bending force constant, θ_j is the actual value of the valence angle, θ_j^0 is the optimal valence angle; the third term represents torsional angle component energy and: V denotes the height of the torsional barrier, p is the periodicity, ω is the torsion angle, γ is a phase factor; the fourth term represents van der Waals component energy described by the Lennard-Jennes potential and: r_{kj} denotes the distance between atoms k and j, σ_{kj} is the collision diameter, ε_{kj} is the well depth, and N is the number of atoms in the structure A^N; the fifth term represents electrostatic component energy and: q_k, q_j are atomic charges, r_{kj} denotes the distance between atoms k and j, ε_0 is a dielectric constant, and N is the number of atoms in the structure A^N.

The Warecki-Znamirowski (WZ) algorithm minimizes the expression (2) for current values of variables defining degrees of freedom, referential values of the variables taken from force field parameter sets [37], and imposing constraints on torsion angles ϕ and ψ resulting from Ramachandran plots [55]. These constraints are derived based on the experimental observations of possible values of torsion angles for particular types of amino acids published in [23].

The full algorithm of Warecki-Znamirowski consists of two phases:

- Phase I: Monte Carlo phase
  In the first phase, random values of ϕ and ψ torsion angles are generated for each amino acid. The space of possible values for torsion angles ($360 \times 360°$) is restricted based on the type of amino acid and the Ramachandran plot for the particular type [23]. Protein conformation is then defined by a vector of ϕ and ψ torsion angles. When all angles are generated for the given amino acid chain, the energy of the conformation E is calculated. The whole process is repeated for a given amount of tries $iTotal$ (the $iTotal$ is specified by the user) and b_c best solutions are collected in a dedicated array. The result of the Monte Carlo phase is the array holding the best conformations of the given amino acid chain. These conformations represent approximations of valid solutions that might still need some adjustment.
- Phase II: Angles Adjustment phase
  The adjustment is performed in the second phase. In this phase, each conformation stored in the array of b_c best conformations is examined once again. The torsion angles ϕ_i and ψ_i ($i = 0, 1, ..., n-1$) of each conformation are modified one by one by given values $\pm k\Delta\phi$ and $\pm k\Delta\psi$. This simulates shaking the protein molecule and continuous motions of particular atoms, wherein the amplitude of the motions is determined by the $\pm\Delta\phi$ and $\pm\Delta\psi$ values. The k parameter is a positive and decreasing parameter during the consecutive steps of the minimization of the potential energy function. The shaking process is continued until the k parameter reaches a fixed value. After each modification of the torsion angle the conformational energy $E'(A^N)$ is calculated once again, and if it is lower than the current minimum $E'(A^N) < E_{min}$, the modified conformation and its energy are accepted and saved as a current best result. The whole adjustment process is performed for each of the b_c best conformations.

The number of random tries $iTotal$ that Phase I is repeated for is defined by the user. In Ref. [66] Warecki and Znamirowski give the formula by which users can estimate the minimal number of iterations that are needed to model the molecule containing n amino acids:

$$iTotal \approx \frac{\log(1-\alpha)}{\log(1-\xi^n)}, \tag{3}$$

where: α is the probability of finding the optimal solution with the accuracy $\xi \in (0; 1)$, which is a part of the range for each i^{th} pair of torsional angles, $i \in \langle 0, ..., n-1 \rangle$. For example, when modeling protein containing 5 amino acids ($n = 5$) and assuming the accuracy of localization of the optimal solution to be half of the range for each pair of torsional angles $\xi = 0.5$ with probability $\alpha = 0.9$, the number of iterations $iTotal = 73$. It is worth noting that with the same assumptions $iTotal = 2,357$ for $n = 10$ amino acids, $iTotal = 75,450$ for $n = 15$ amino acids, and $iTotal = 2,414,435$ for $n = 20$ amino acids. This shows that the number of possible conformations that should be explored grows quickly with the number of amino acids.

The best conformation is the one that is characterized by the lowest value of the potential energy E. Energies are calculated with the Tinker package [54] and AMBER96 [37] molecular mechanics potentials and the generalized Born continuum solvation model (GBSA) in the presence of side chains.

2.2 Cloud4PSP Architecture

Cloud4PSP parallelizes the WZ method by the distribution of the computations in the Cloud computing. Cloud4PSP has been developed to work on the Microsoft Azure public cloud. In the Microsoft Azure

cloud, applications can be delivered by deploying pre-configured virtual machines (IaaS) or as fully functional SaaS solutions. For the latter case, Microsoft provides a dedicated application programming interface (API) for developers of any software application that is intended to work on the Cloud. Internally, such an application is composed of a set of roles - a Web role providing a graphical user interface (GUI) in the form of a web site, and Worker roles implementing the application logic. The roles reside in and execute the application logic on virtual machines provided by Microsoft, as a cloud provider. These virtual machines come from Microsoft's standard gallery of virtual machines and provide various computational capabilities depending on their size (see Table 1).

Roles working in the Cloud4PSP system create a distributed set of processing units Ω_{PU} defined as follows:

$$\Omega_{PU} = \{r_W\} \cup \{r_{PM}\} \cup R_{PW}, \quad (4)$$

where r_W is the Web role responsible for the interaction with Cloud4PSP users, r_{PM} is the PredictionManager role distributing requests received from the Web role and preparing the workload, and R_{PW} is a set of m instances of the PredictionWorker role performing the prediction of protein structures in parallel.

The architecture of the Cloud4PSP consisting of the mentioned processing units is shown in Fig. 4. The Web role provides a front-end for users of the system, the PredictionManager role mediates the distribution of the prediction process and stores the results in the Azure SQL Database, and the PredictionWorker roles predict the protein structure according to the logic of the chosen prediction method. Control instructions and parameters are transferred through

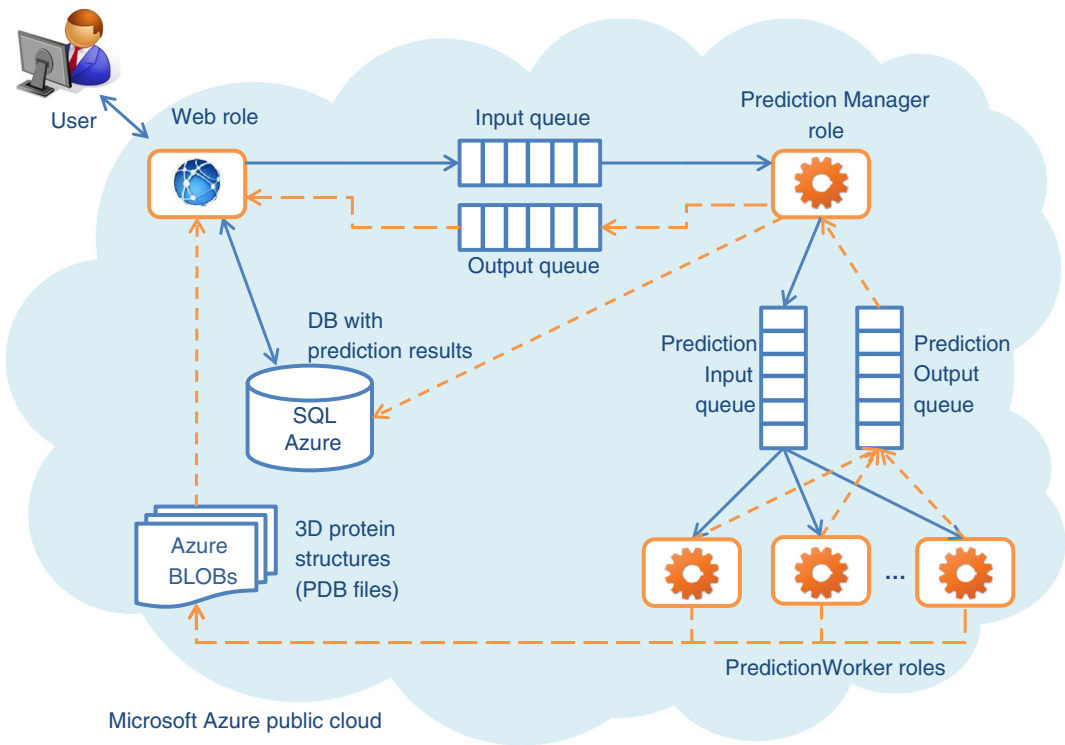

Fig. 4 Architecture of the Cloud4PSP - a cloud-based solution for *ab initio* protein structure prediction

appropriate queues. Roles have access to various storage resources, including Azure BLOBs (for PDB files with predicted 3D structures) and Azure SQL Database (for description of the results).

The Web role provides a Web site which allows users to interact with the entire system. Through the Web role users input the amino acid sequence of the protein whose structure is to be predicted. They also choose the prediction method (only one is available at the moment, but the system provides programming interfaces allowing us to bind new methods), specify its parameters, e.g. the number of iterations (random tries of the modified Monte Carlo method), and provide a short description of the input molecule.

The PredictionManager role distributes the prediction process across many instances of the PredictionWorker role. In other words, PredictionManager implements the entire logic of how the prediction process is parallelized. This may depend on the prediction method used. Algorithm 1 describes how calculations are scheduled and parallelized by the PredictionManager role with part of the code adapted to the specificity of the WZ prediction algorithm used. When the prediction is performed according to the WZ algorithm, the entire prediction process (in the system also called a *prediction job*) consists of a vast number of iterations (random selections of torsion angles) that, in the Cloud4PSP, are performed in parallel by many instances of the PredictionWorker role. The number of iterations is specified by the user through the Web role. In the configuration of the prediction job, the user specifies the number of tasks ($\#tasks$) and the number of tries performed in each task ($\#iterations$). This gives flexibility in choosing the stopping criterion and flexibility in configuring prediction jobs (and in consequence, the number of candidate structures after the prediction, since in this implementation of the parallel prediction only the best conformations generated in Phase I by each task are adjusted in Phase II and these tuned conformations are returned). The total number of iterations performed within the prediction job ($iTotal$) is a product of the number of tasks ($\#tasks$) and the number of tries performed in each task ($\#iterations$):

$$iTotal = \#tasks \times \#iterations. \tag{5}$$

The PredictionManager role creates a pool of $\#tasks$ tasks (lines 6–7). Descriptions of these tasks, which contain the information on the prediction algorithm, its parameters, random seed used to initialize a pseudorandom number generator for the modified Monte Carlo method, and the number of tries $\#iterations$, are stored in the PredictionInput queue (lines 11–12). Prediction tasks are then consumed by instances of the PredictionWorker role.

Instances of the PredictionWorker role act according to Algorithm 2. They consume messages from the Prediction Input queue (lines 2–4), execute the chosen prediction method, generate successive protein structures, calculate potential energies for them (lines 6–7), store protein structures in PDB files, and then save these files in Azure BLOBs (line 8). The set of PredictionWorker role instances R_{PW} is defined as follows:

$$R_{PW} = \{r_{PWi} | i = 1, ..., m\} \tag{6}$$

where r_{PWi} is a single instance of PredictionWorker role, and m is the number of PredictionWorker role instances working in the system.

Usually:

$$\#tasks \gg m. \tag{7}$$

Steering instructions that control the entire system and the course of action inside the Cloud4PSP are transferred as messages through the queueing system. There are four queues present in the architecture of the Cloud4PSP:

- Input queue - collects structure prediction requests (Prediction Job descriptions) generated by users through the Web site.
- Output queue - collects notifications of prediction completion for particular users (based on a generated token number).
- Prediction Input queue - transfers parameters of the prediction process for a single instance of the PredictionWorker role, among others: amino acid sequence provided by the user, the number of iterations $\#iterations$ to be performed, parameters of the prediction method.
- Prediction Output queue - transfers descriptions of results, e.g. PDB file name with protein structure and potential energy value for the structure, from each instance of the PredictionWorker role back to the PredictionManager role.

The Input queue and Prediction Input queue are very important for buffering prediction requests and

Algorithm 1 PredictionManager role: Processing prediction jobs

```
1: while true do
2:     Check messages in the Input queue
3:     if exists a message then
4:         Retrieve the message and extract parameters of the prediction job (protein sequence, #tasks, prediction method, and its parameters)
5:         Save job description in SQL Azure database
6:         for i ← 1, #tasks do          ▷ This part of the code is specific to WZ-oriented Prediction Manager
7:             Create a prediction task
8:             Generate seed for the pseudorandom number generator
9:             Assign the seed to the prediction task
10:            Save prediction task description in SQL Azure database
11:            Encode prediction task (seed, sequence, prediction method, and its parameters) in the task description message
12:            Enqueue task description message in the Prediction Input queue
13:        end for
14:    end if
15:    Check messages in the Prediction Output queue
16:    if exists a message then
17:        Retrieve the message and extract results of the prediction task (name of the file with predicted structure, potential energy)
18:        Save task results in SQL Azure database
19:        if all tasks completed then
20:            Finalize the prediction job in SQL Azure database
21:        end if
22:    end if
23: end while
```

steering the prediction process. The prediction process may take several hours and is performed asynchronously. A user's requests are placed in the Input queue with the typical FIFO discipline and they are processed when there are idle instances of the PredictionWorker role that can realize the request. Many users can generate many prediction requests, so there must be a buffering mechanism for these requests. Queues fulfill this task perfectly. One prediction request enqueued in the Input queue causes the creation of many prediction orders (tasks) for instances of the PredictionWorker role. The Predic-

Algorithm 2 PredictionWorker role: Processing prediction task

```
1: while true do
2:     Check messages in the Prediction Input queue
3:     if exists a message then
4:         Retrieve the message and extract parameters of the prediction task (protein sequence, prediction method, and its parameters, including seed)
5:         Update task start time in SQL Azure database
6:         Execute prediction process according to the specified method
7:         Collect results of the prediction (name of the file with predicted structure, potential energy)
8:         Save protein structure to Azure BLOB under given filename
9:         Update task end time in SQL Azure database
10:        Serialize task execution results in the prediction output message
11:        Enqueue the message in the Prediction Output queue
12:    end if
13: end while
```

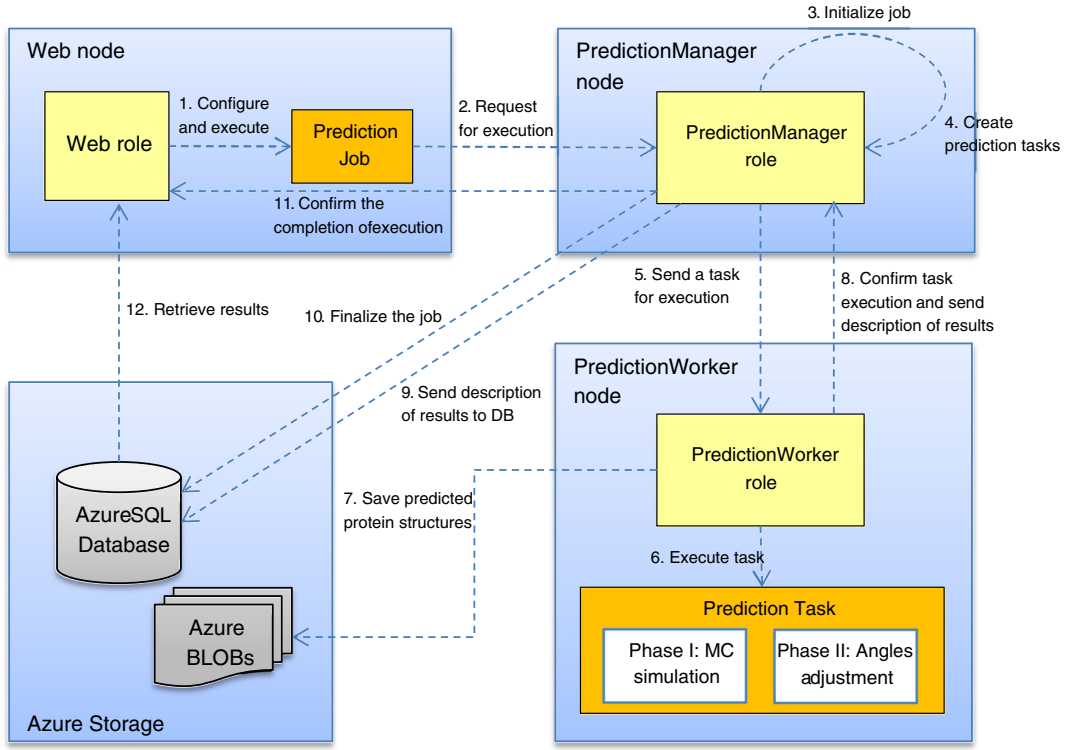

Fig. 5 Cloud4PSP processing model showing components and objects involved in the prediction process, and the flow of messages and data between components of the system

tionManager role consumes prediction requests from the Input Queue, generates many predictions tasks, each for performing the specified number of iterations by PredictionWorker roles available in the system, and generates task description messages to the Prediction Input queue. The Prediction Output queue is used by instances of the PredictionWorker role to return a description of results to the PredictionManager role. Such a feedback allows the PredictionManager role to control if all the prediction tasks have been completed. In the case of failure to realize of a prediction task, its completion can be delegated to another PredictionWorker instance. The Output queue allows the Web role to be notified that the whole prediction process is already completed.

2.3 Cloud4PSP Processing Model

While building the Cloud4PSP, we have designed and made available a dedicated processing model (Fig. 5) that is used by the Cloud4PSP and can be used by developers extending the system in the future. In the Cloud4PSP processing model, every prediction request generated by a user causes the creation of a Prediction Job (step 1). This is a special object with the information on the entire prediction that must be performed. This Prediction Job is submitted for execution (step 2), i.e., it is serialized to the message that is placed in the Input queue. The PredictionManager role consumes Prediction Jobs, initialize them (step 3) and, based on their description and available resources, creates so-called Prediction Tasks (step 4). These tasks are serialized to the messages that are sent to the Prediction Input queue (step 5) and are then consumed by successive PredictionWorker role instances. PredictionWorker role instances execute tasks according to the description contained in the retrieved message and using an appropriate prediction method (step 6), and finally save the predicted protein structures in the Azure BLOB storage service

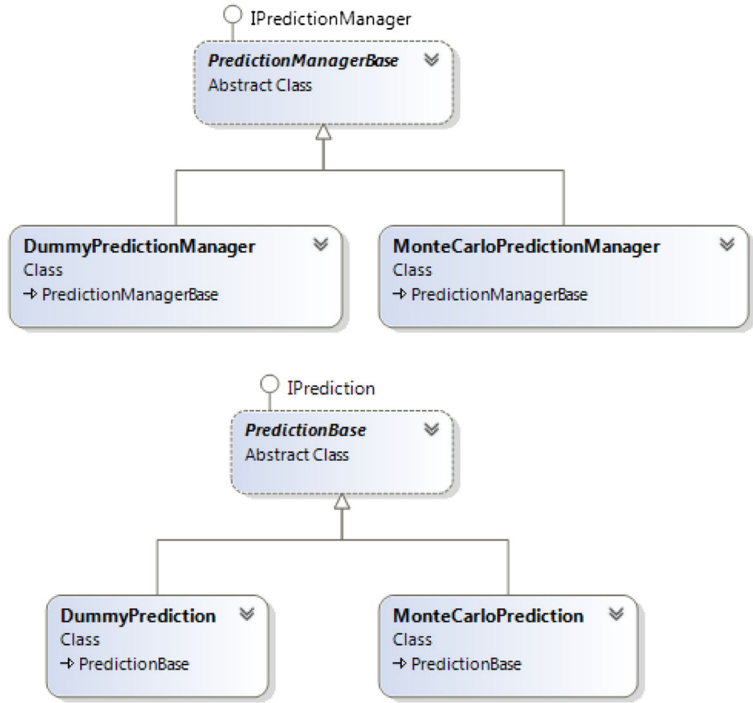

Fig. 6 Interfaces, their implementations for PredictionManager and PredictionWorker in the form of base classes, and inheritance for classes representing a particular prediction method. DummyPredictionManager and DummyPrediction represent the sample test manager and prediction method, MonteCarloPredictionManager and MonteCarloPrediction reflect implementations of the scheduling/parallelization and prediction procedure for the described WZ method

(step 7). After completing the Prediction Task, PredictionWorker confirms completion of the task and sends a description of the generated protein structures to the PredictionManager (step 8). The PredictionManager saves the results in the database managed by the Azure SQL Database (step 9). By using an additional register stored in the database, the PredictionManager maintains awareness of the progress of the Prediction Job execution. This emphasizes the importance of the confirmations obtained in step 8. When the PredictionManager states that all tasks have been completed, it finalizes the job by setting an appropriate property in the job's register in the database (step 10) and sends the notification to the Web role through the Output queue (step 11). The Output queue stores notifications together with the token number generated for the Prediction Job at the beginning of the prediction process, i.e., when submitting the job for execution. The Web role periodically checks the Output queue for messages reporting the completion of the Prediction Job and the statuses of prediction tasks in the database (step 12). Partial results (results of already finished tasks) are retrieved from the database. In this way, the user knows whether the prediction process is completed or is still in progress.

Such a processing model provides a kind of abstraction layer for developers of the system, hides some implementation details, e.g., implementation of communication and serialization of jobs and tasks, and finally, provides a framework for future extensions of the system.

2.4 Extending Cloud4PSP

Although Cloud4PSP has been designed to work on the Microsoft Azure Cloud, its architecture and workflow have a general character and can be also implemented on other Clouds. This, however, would require adoption of the programming model specific to the Cloud provider and the tailoring of elements of the architecture to available compute resources. The architecture of Cloud4PSP could also be adapted and extended to hybrid clouds by combining on-premises compute resources and Microsoft Azure cloud resources. This would require some additional effort, related to moving the PredictionManager to

the on-premises cluster or private cloud and parceling out prediction tasks across available resources. However, this would also allow for the use of public cloud resources to satisfy potentially increased demand especially for compute units. If the system needs some extra compute power, e.g., due to an excessively long and computationally demanding prediction process, part of the prediction job (some prediction tasks) could be scheduled for execution on the Azure cloud. Then, after the prediction process is completed these commercial compute resources could be released.

Cloud4PSP has been developed in such a way that it allows developers to extend its functionality by adding new modules. This is important, for example, in situations when new prediction algorithms have to be added to the system. This determines the openness of the system. Cloud4PSP has been mainly developed as a software, which places it in the SaaS layer of the cloud stack [47]. However, extensions are possible not only by using the processing model presented in Section 2.3, but also by a set of programming interfaces provided for almost all components of the system. In Fig. 6 we show interfaces, their implementations for PredictionManager (for parallelization logic) and PredictionWorker (for the prediction method) in the form of base classes, and inheritance for classes representing particular prediction methods. By using these base classes, advanced users and programmers can plug in their own prediction methods (by inheritance from the *PredictionBase* class) and their own computation scheduling algorithms (by inheritance from the *PredictionManagerBase* class)

2.5 Scaling the Cloud4PSP

The scalability of the system means that it is able to handle a growing amount of work in a capable manner or is able to be enlarged to accommodate that growth [5]. Cloud4PSP has this property. Cloud4PSP is a cloud-based, high performance and highly scalable system for 3D protein structure prediction. The scalability of the system is provided by the Microsoft Azure cloud platform. The system can be scaled up (vertical scaling) or scaled out (horizontal scaling). Microsoft Azure allows us to combine both types of scaling.

Cloud4PSP can be scaled out and scaled up during the protein structure prediction process. Scaling out means changing the value of m in (6). Scaling up implies the change of the size of the PredictionWorker role:

$$size(r_{PWi}) \in \{XS, S, M, L, XL\}, \tag{8}$$

where: XS denotes ExtraSmall size, S - Small, M - Medium, L - Large, XL - ExtraLarge. The sizes of the virtual machines for PredictionWorker role instances are described in Table 1 and are consistent with the parameters of the computation units provided.

In the Cloud4PSP we also made the following assumption regarding all instances of the PredictionWorker role:

$$\forall_{r_{PWi}, r_{PWj} \in R_{PW}, i,j \in 1,\ldots,m, i \neq j} \quad size(r_{PWi}) = size(r_{PWj}). \tag{9}$$

The assumption of the same size for all instances of the PredictionWorker role is motivated by the ease of configuration and scaling of the system and by the ease of analysis of its scalability.

The number of instances m of the PredictionWorker role depends on the configuration of the Cloud4PSP. It can be measured in thousands. The m is limited only by the user's subscription for Microsoft Azure cloud resources and the number of compute units available in the Azure cloud.

3 Results

Prediction of 3D protein structures from the beginning with the use of the WZ method requires many Monte Carlo iterations, which takes some time. The number of iterations and therefore the total execution time depend on the size of the protein modeled, i.e., the number of residues in the input sequence. During our tests we successfully modeled a 20 amino acid long part of the nonstructural protein NSP1 molecule from the *Semliki Forest* virus with an RMSD of 0.99Å. The NMR structure of the molecule (PDB ID code: 1FW5) [40] from the Protein Data Bank is shown in Fig. 7 (top). The structural alignment and superposition of the modeled molecule and NMR molecule are presented in Fig. 7 (bottom). Modeling the 20 amino acid long part of the NSP1 protein was a sample computational procedure used for testing the efficiency of the designed Cloud4PSP architecture in the real (Microsoft Azure) cloud environment. Tests were performed with the use of twenty CPU cores. Two of these twenty cores, i.e., two compute units of the

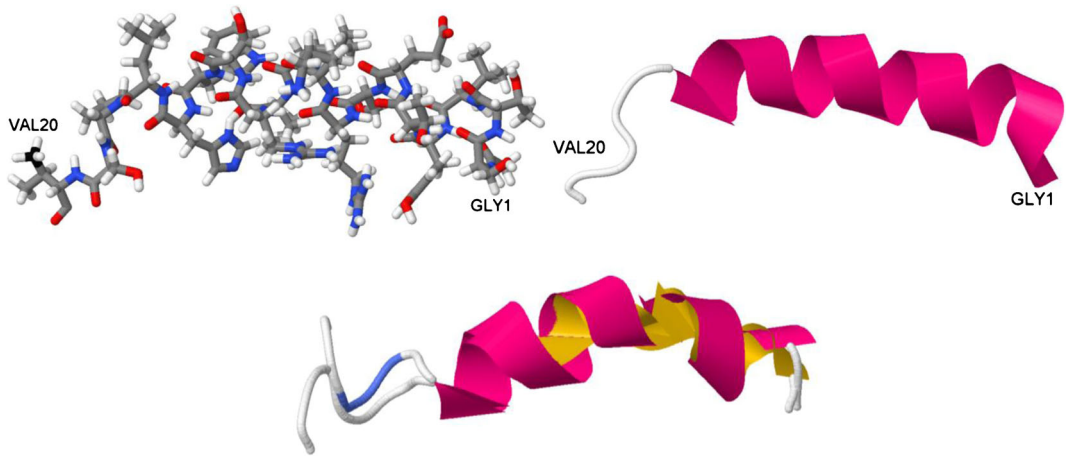

Fig. 7 Crystal structure of the part of the NSP1 molecule (PDB ID code: 1FW5) [40] from the *Semliki Forest* virus that was modeled during experiments: (*top left*) protein skeletal structure made by covalent bonds between atoms (*sticks* display mode in Jmol [29]), (*top right*) secondary structure showing mainly α-helical character of this fragment (*ribbon* display mode in Jmol), (*bottom*) structural alignment and superposition of modeled molecule and NMR molecule

Small size, were allocated to allow the activity of the Web role and PredictionManager role. The remaining eighteen cores were used for the activity of the PredictionWorker role instances and to test the scalability of the system.

In particular, we have tested:

- the vertical scalability of the system - the efficiency of the prediction process depending on the size of instances of the PredictionWorker role,
- the horizontal scalability of the system - the efficiency of the prediction process depending on the number of instances of the PredictionWorker role,
- the efficiency of the prediction process depending on the number of prediction tasks and the number of iterations performed by each task.

In all cases, the scalability has been determined on the basis of execution time measurements. We have carried out at least two replicas for each measurement. Then, the obtained results were averaged and in such a way they are presented in the following sections. Averaged values of measurements were also used to determine n-fold speedups for both scaling techniques.

3.1 Vertical Scalability

In the first phase of our tests, we wanted to check which of the sizes for the compute units offered by Microsoft Azure is most efficient. The basic tier for the Web and Worker roles consists of the following five sizes: ExtraSmall (XS), Small (S), Medium (M), Large (L), and ExtraLarge (XL). Their capabilities were described in Sect. 1.2 and briefly compared in Table 1. It is worth noting in Table 1 that beginning from the Small size up to the ExtraLarge size the amount of available memory and the number of cores doubles gradually. The ExtraSmall size provides one, shared core and is generally used when testing applications that should work on the Cloud, so as not to generate unnecessary costs for the testing process.

While testing vertical scalability, we changed the size of the PredictionWorker role from ExtraSmall (one, shared CPU core) to ExtraLarge (eight CPU

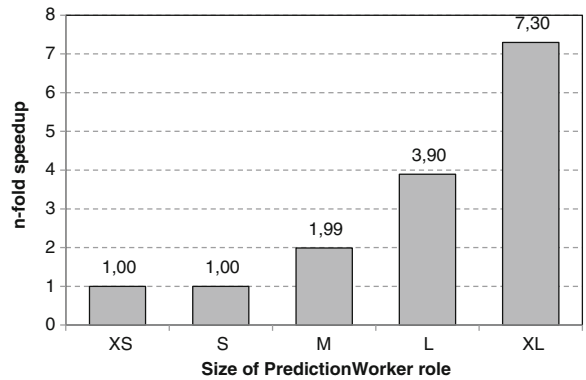

Fig. 8 Results of vertical scaling. Acceleration (n-fold speedup) of the 3D protein structure prediction (prediction job) as a function of the size of instances of the PredictionWorker role

Fig. 9 Results of vertical scaling. Average execution time for a single prediction task as a function of the size of the PredictionWorker role

Fig. 10 Results of horizontal scaling. Total execution time for a prediction job as a function of the number of instances of the PredictionWorker role (Small size)

cores). Only one PredictionWorker was used in this test. The Prediction job was configured to explore 80 000 random conformations ($iTotal$) that were performed in 16 prediction tasks. Multiple prediction tasks were assigned to those instances of PredictionWorker that possessed more than one CPU core, proportionally to the number of cores (according to the rule: one core - one task). The input sequence contained 20 amino acids of the NSP1 molecule. Each task was configured to generate 5 000 random protein conformations (performed 5 000 iterations per task) in Phase I of the WZ method, and to tune only one best conformation in Phase II of the method ($\Delta\phi = \Delta\psi =$ 8 degrees for $k = 3, 2, 1$ in successive adjustment iterations).

In Fig. 8 we show the n-fold speedup when scaling the system vertically. We observed that increasing the size of the PredictionWorker role from Small to Medium accelerated the prediction process almost two-fold. Above the Medium size (2 cores), the dynamics of the acceleration slowed down a bit - for Large size (4 cores) the acceleration reached 3.90 and for ExtraLarge size (8 cores) it reached 7.30.

When analyzing the results of tests, we deduced that the slowdown in the acceleration dynamics was an effect of two factors. The first factor was an overhead caused by the necessity of handling multiple threads. For example, on ExtraLarge-sized PredictionWorkers we ran eight prediction tasks concurrently (8 CPU cores = 8 threads = 8 prediction tasks). And the second factor was the cumulative I/O operations performed on a local hard drive by many threads of the PredictionWorker role. During computations, the PredictionWorker role makes use of various executable programs and additional files that must be read from the local hard drive. It also stores some intermediate results on the hard drive. Multiple threads of the role that run on the compute unit multiply these read/write operations and interfere with each other. Therefore, the more threads were running on the compute unit during our experiments, the more I/O operations were performed, which influenced the n-fold speedup. This is also visible when analyzing average execution times for prediction tasks, as shown in Fig. 9.

In Fig. 9 we can clearly see that execution of a single prediction task was the fastest when performed on Small-sized instances and the slowest on ExtraLarge-sized instances of the PredictionWorker role. However, ExtraLarge instances could host the execution of eight prediction tasks at the same time, while Small-sized instances could host only one. Therefore, the total execution time for the whole prediction job performed on ExtraLarge instances was 7.30 times shorter than the same job performed on a single Small instance.

3.2 Horizontal Scalability

In the second series of our performance tests, we checked the horizontal scalability of the Cloud4PSP. Tests were performed in the Microsoft Azure cloud. During this series of tests we increased the number of instances of the PredictionWorker role from one to 18. Instances of the PredictionWorker role were hosted on compute units of the Small size (one CPU core). The Small-sized virtual machine represents the cheapest

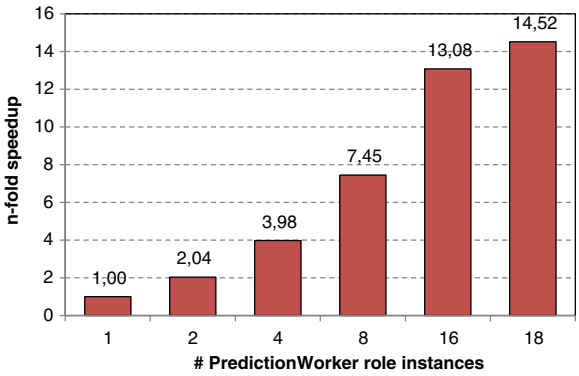

Fig. 11 Results of horizontal scaling. Acceleration (n-fold speedup) of the 3D protein structure prediction as a function of the number of instances of the PredictionWorker role (Small size)

computational unit, providing one unshared core, and is the smallest sized unit recommended for production workloads [49]. Moreover, as proved by the experimental results shown in Fig. 9, the size guaranteed the fastest processing of prediction tasks.

While testing the horizontal scalability, we used the same input sequence as in the vertical scalability tests. The sequence contained 20 amino acids of the NSP1 enzyme. The prediction was configured to explore 4 000 random conformations ($iTotal$) that were performed in 40 tasks. Each task was configured to generate 100 random protein conformations (performed 100 iterations per task) in Phase I of the WZ method, and to tune only one best conformation in Phase II of the method ($\Delta\phi = \Delta\psi = 8$ degrees for $k = 3, 2, 1$ in successive angle adjustment iterations).

In Fig. 10 we show total execution time for the whole prediction as a function of the number of instances of the PredictionWorker role. The results obtained proved that the total prediction time can be significantly reduced when increasing the number of compute units. Upon increasing the number of instances from one to 18, we reduced the prediction time from 9 h and 31 min to only 39 min.

Based on the execution time measurements that we obtained during the performance tests, we calculated the n-fold speedup for various configurations of the Cloud4PSP with respect to a single instance configuration. Figure 11 shows how the n-fold speedup changes as a function of the number of instances of the PredictionWorker role.

We noticed that employing two instances of the PredictionWorker role increased the speed of the prediction process more than two-fold. Adding more instances of the role proportionally accelerated the process. Finally, the acceleration ratio (n-fold speedup) reached the level of 14.52, when the number of instances of the PredictionWorker role was increased from one to 18. A slowdown of the acceleration dynamics was observed when using more than four compute units. This was caused by the uneven processing times of individual tasks – the average execution time per task was 840 s, with minimum 340 s and maximum 1722 s. Moreover, we have to remember that the execution times and n-fold speedup were measured for the whole system. And therefore, the execution times covered not only the execution of particular prediction tasks, but also processing and storing the results of these tasks by the PredictionManager role. Consequently, by increasing the degree of parallelism we raised the pressure on the PredictionManager, which serially processed incoming prediction results.

An interesting question is also why the processing times of tasks are so different. The answer lies in the prediction method itself, and specifically in its second phase. Phase I is fairly predictable in terms of the speed of generating random protein conformations and calculating energies for them (Fig. 12). The problem lies in the second phase, which can take much longer, depending on the length of the input sequences, tuning parameters and the course of tuning itself, which depends on the conformation generated in the first phase. We observed that for the tested input sequence Phase I took 5 % and Phase II took 95 % of the whole execution time.

3.3 Influence of the Task Size

Since in Phase II of the WZ method angle adjustment is only performed for the structure with the lowest energy, while performing our experiments we expected that the total execution time of the prediction process might depend on the configuration of the prediction tasks. We decided to verify how the task size (i.e. the number of iterations) and entire job configuration influence the whole prediction time and how strong that influence is. For this purpose, we decided to explore 4 000 random conformations ($iTotal$) in the prediction process, for the same input sequence as in the vertical and horizontal scalability tests. The sequence contained 20 amino acids of

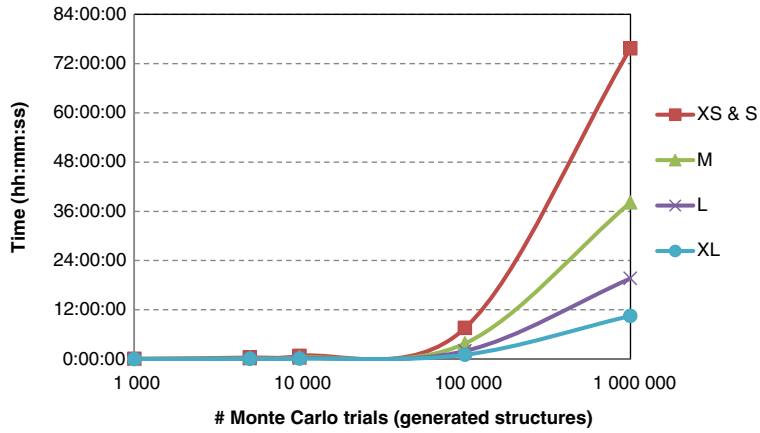

Fig. 12 Efficiency of a single compute unit in the first phase of the prediction process. Dependency between the execution time and the number of randomly generated structures (Monte Carlo trials) for various sizes of compute units

the NSP1 molecule. We tested nine different configurations of the prediction job. Leaving the number of random conformations ($iTotal$) constant, we appropriately selected the number of iterations per task and the number of tasks. Phase II of the WZ method was set up to tune only one best conformation for $\Delta\phi = \Delta\psi = 8$ degrees for $k = 3, 2, 1$ in successive angle adjustment iterations. Tests were performed using sixteen Small-sized instances of the PredictionWorker role.

The total execution times for various configurations of the prediction job are shown in Fig. 13. As can be observed, the most efficient is the configuration with the lowest number of tasks and the highest number of iterations ($16t \times 250i$). The prediction process for this configuration took 19 min and 25 s. The configuration with the largest number of small tasks ($800t \times 5i$) proved to be extremely inefficient. The prediction process for this configuration took 11 h and 42 min. The reasons for such a behavior of the system are the same as already discussed in the previous section. The most time-consuming phase of the WZ method is Phase II, which usually takes 90–98 % of the task execution time. By configuring the system to use many small prediction tasks, we multiplied the execution of the longest phase, which resulted in a long execution time for the whole prediction job.

For a small number of tasks it is advisable to choose the number of tasks as a multiple of the number of instances of the PredictionWorker role. For a larger number of tasks (if # tasks $> 2m$) the rule does not

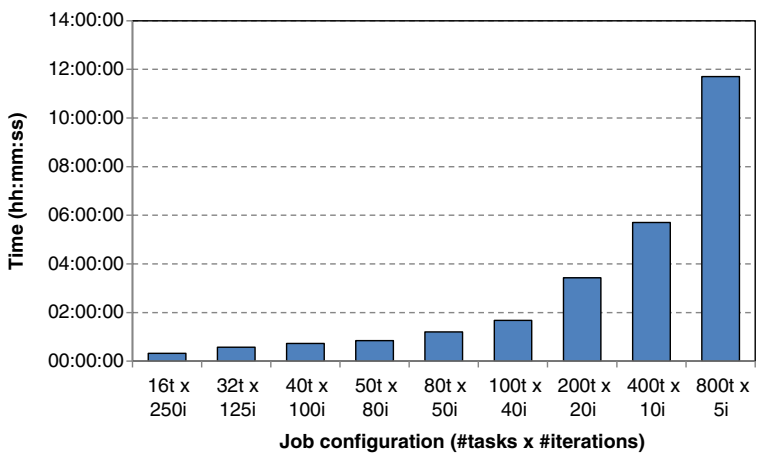

Fig. 13 Dependency between execution time and configuration of the prediction job ($\#tasks \times \#iterations$) for the constant number of randomly generated structures (Monte Carlo trials, $iTotal = 4000$) for sixteen instances of the PredictionWorker role

apply, since the execution times for particular tasks are different.

It is difficult to explicitly and clearly define the configuration of the prediction job. It depends on many factors, including the purpose of the prediction process, the size of the input molecule, and others. However, choosing a low number of tasks and a large number of iterations has the following consequences:

- the efficiency of the prediction process is higher,
- users obtain fewer candidate structures after the prediction is finished, and some good candidates can be missed,
- fewer prediction results are stored in the database and fewer 3D structures are stored in BLOBs, which lowers the long-term costs of storage in the Cloud,
- the cost of the computation process is lower due to higher efficiency.

On the other hand, choosing a large number of tasks and a low number of iterations has the following consequences:

- the efficiency of the prediction process is lower,
- users obtain more candidate protein structures after the prediction process, which increases the chance to get the global minimum of the energy function,
- more prediction results are stored in the database and more candidate 3D structures are stored in BLOBs, which increases the long-term costs of storage in the Cloud,
- the cost of the computation process is higher due to lower efficiency.

3.4 Scale Up, Scale Out or Combine?

Microsoft Azure allows us to scale out and scale up applications and systems deployed in the Cloud. The choice is left to the developer and administrator of the system. In both cases, the system must be properly implemented to utilize the resources of the multi-core compute unit (when scaling up) and to distribute, and then process, parts of the main job by many instances of the Worker role (when scaling out). The workload associated with the development of the system towards each of the scaling types is difficult to express in numbers. However, there are some technical reasons that speak in favor of particular scaling techniques. Moreover, we have conducted a series of tests in order to compare similar configurations of the Cloud4PSP architecture and answer which of the configurations is more efficient.

We tested various configurations of the Cloud4PSP by changing the size of the PredictionWorker role from ExtraSmall (one, shared CPU core) to ExtraLarge (eight CPU cores) and by changing the number of instances of the role from one to eighteen (if possible, depending on the size of the compute unit). The prediction was configured to explore 100 000 random conformations ($iTotal$) that were performed in 18 tasks. Multiple prediction tasks were assigned to instances that possessed more than one CPU core, proportionally to the number of cores (according to the rule: one core - one task). The input sequence contained 20 amino acids of the NSP1 enzyme. Each task was configured to generate 5 000 random protein conformations (performed 5 000 iterations per task) in Phase I of the WZ method, and to tune only one best conformation in Phase II of the method ($\Delta\phi = \Delta\psi = 8$ degrees for $k = 3, 2, 1$ in successive adjustment iterations).

We tested the following configurations of the Cloud4PSP with respect to resource consumption by the PredictionWorker role:

- one to eighteen ExtraSmall and Small compute units,
- one to nine Medium compute units,
- one to four Large compute units,
- one to two ExtraLarge compute units.

In Fig. 14 we show the number of prediction tasks completed within one hour as a function of the number of instances of the PredictionWorker role for various sizes of the instances. The results of the tests indicate a slight advantage of horizontal scaling over vertical scaling. Comparing both scaling techniques, we noted that the number of prediction tasks completed within one hour by eight Small (1-core) instances was higher than those completed by one ExtraLarge (8-core) instance. Similar behavior was observed when comparing four Small instances vs. one Large (4-core) instance, and two Small instances vs. one Medium (2-core) instance, but the differences were not so significant. We could draw the same conclusion based on the charts showing n-fold speedups for vertical scaling (presented in Fig. 8) and horizontal scaling (presented in Fig. 11). For example, the acceleration ratio was

slightly better for eight Small instances of the PredictionWorker role (7.45) than for a single, ExtraLarge (8-core) instance (7.30).

When combining both scaling techniques, our results again speak in favor of horizontal scaling with the use of Small-sized instances, e.g., eighteen Small instances vs. nine Medium instances, sixteen Small instances vs. two ExtraLarge instances, eight Small instances vs. two Large instances, etc. The differences are not significant, but they increase with the number of simulation hours. We must also remember that, although in Fig. 14 we present measurements for particular parameters of the prediction job, the entire experiment reveals the relationships between particular configurations of the Cloud4PSP. The actual execution times depend on the size of the input sequence, the number of Monte Carlo trials, the number of tasks, and the parameters of the WZ method.

It should also be noted that horizontal scaling is easier to implement. Once the Cloud4PSP system is designed to be scaled horizontally, scaling can be done at any moment after the system is published in the Cloud and only if the system requires higher resources (more compute units). Moreover, it can be done not only manually by the administrator of the system, but also automatically based on statistics of the system performance and resource usage indicators.

4 Discussion

Building Cloud computing services for the prediction of 3D protein structures, such as the presented Cloud4PSP, responds to the current demands for having widely available and scalable platforms solving these types of difficult problems. This is very important not only for scientists and research laboratories trying to predict a single 3D protein structure, but also for the biotechnology and pharmaceutical industry producing new drug solutions for humanity.

Cloud4PSP is such a platform, allowing a highly scalable prediction process in the Cloud, with all its advantages and drawbacks. Cloud4PSP was originally designed to be deployed in Microsoft's commercial cloud, where computing resources can be dynamically allocated as needed. As we have shown, the prediction process can be scaled up by adding more powerful computing instances, or scaled out by allocating more computing units of the same types. The results of our experiments have shown that scaling out by adding more Small-sized computing units turned out to be more effective and to give more development flexibility than scaling up (the acceleration rate for 8 cores was 7.45 when scaling out and 7.30 when scaling up). However, in the case of big molecules being modeled, we may be forced to use larger compute units, e.g. Medium, Large, or ExtraLarge, and to combine horizontal and vertical scaling. Delivering Cloud4PSP as a service offloads bio-oriented companies from maintaining their own IT infrastructures, which can now be outsourced from the cloud provider. This has another positive consequence - a low barrier to entry. Even a laboratory or a company that cannot afford to possess a high-performance computational infrastructure can outsource it in the Cloud and perform a prediction process starting with several computation units, then grow up, if needed, and scale its systemo at any time.

On the other hand, the important aspects of processing data in the Cloud are privacy and data protection. As rightly pointed out by Gesing et al. [20], both the input and output data of molecular simulations are sensitive and may constitute a valuable intellectual property. Therefore, they need to be stored securely. Cloud security is still an evolving domain and defensive mechanisms are implemented on various levels of the functioning of the cloud-based system. In order to protect the systems deployed in the Cloud, Microsoft Azure provides a reach set of security elements, including SSL mutual authentication mechanisms while accessing various components of the system, encryption of data stored in Azure Storage and encryption of data transmission, firewalled and partitioned networks to help protect against unwanted traffic from the Internet, and many others. Many of the security elements are explicitly, and many of them implicitly, used by Cloud4PSP, e.g., secure access to Azure BLOBs using HTTPS, secure authentications when accessing SQL Azure, where prediction results are stored, SSL-based communication between internal components. In terms of data privacy, Microsoft is committed to safeguarding the privacy of data and follows the provisions of the E.U. Data Protection Directive. Users of the Azure cloud, i.e., cloud application developers, may specify the geographic areas where the data are stored and replicated for redundancy. Also, the distribution and deployment model of the Cloud4PSP supports privacy and data protection. On the deployment level, various users and companies are

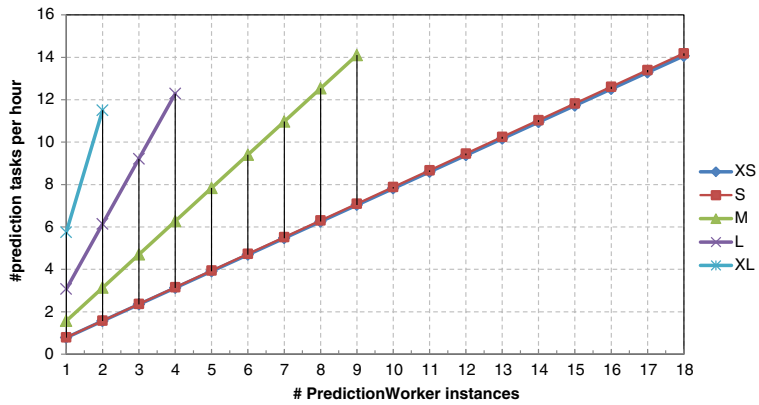

Fig. 14 Comparison of the efficiency of horizontal and vertical scaling, and the combination of both scaling techniques. The number of prediction tasks completed within one hour as a function of the number of instances of the PredictionWorker role for various sizes of the instances. Measurements for ExtraSmall (XS) and Small (S) sizes are similar

expected to establish their private Cloud4PSP-based clusters in the Azure cloud for their own simulations. This ensures separation of data and computations performed by two companies, for which the predicted 3D structures of proteins may be a highly-protected intellectual property. Finally, on the application level, Cloud4PSP provides additional security mechanism of 128-bit GUID[1]-based tokens for protecting access to results of prediction processes.

Comparing Cloud4PSP to other solutions mentioned in the Introduction section, we can draw the following conclusions. Firstly, Cloud4PSP, in terms of what it offers to the end user, is a typical Software as a Service solution (SaaS), and also an extensible computational framework, where advanced users may add their own prediction methods. This is the fundamental feature that differentiates our cloud-based software from the Cloud BioLinux. Cloud BioLinux provides a virtual machine and, in the context of the presented service stack, functions in the Infrastructure as a Service (IaaS) model, providing pre-configured software on a pre-configured Ubuntu operating system and leaving the users the responsibility for maintaining the operating systems and configuring the best way to scale the available applications. The second fundamental difference between the two solutions is the software used for structure prediction. Cloud BioLinux is rich in various software packages, allowing us to perform many processes in the domain of bioinformatics. However, in terms of the protein structure prediction, it represents a different approach to the problem. It relies on the PredictProtein and provides tools for the prediction of secondary structures of macromolecules, not 3D structures. Unlike Cloud BioLinux, Cloud4PSP is focused on *ab initio* prediction methods for tertiary structure.

Considering these two mentioned features, i.e. the service model and prediction methods, Cloud4PSP is more similar to Rosetta@Cloud and its AbinitioRelax application. In both systems prediction services are provided in the SaaS model. Although both systems use different prediction methods, the implemented prediction methods belong to the same *ab initio* class. Both systems are prepared to be published to the Cloud, giving users the ability to establish their private cloud-based clusters for 3D protein structure prediction, which is also important from the viewpoint of data protection. Rosetta@Cloud builds a cluster of compute units on Amazon Web Services (AWS) and Cloud4PSP works on the Microsoft Azure cloud. There are also some similarities and differences in the architectures of the two systems. Rosetta@Cloud uses a so-called Master Node that controls the distribution of workload among Worker Nodes. The Master Node in Rosetta@Cloud is a counterpart of the PredictionManager role in Cloud4PSP, and Worker Nodes are counterparts of instances of the PredictionWorker role. However, Rosetta's GUI for launching predictions is installed locally on the user's computer as Rosetta@Cloud Launcher, i.e., it works outside the cloud infrastructure, while the Cloud4PSP GUI is designed to be available through a web site (provided by the Web role working inside the cloud), and therefore, users do not have to install anything locally.

[1]GUID - Globally Unique Identifier

Moreover, Cloud4PSP uses queues to buffer prediction requests. Applying queues has several advantages. Since queues provide an asynchronous messaging model, users of Cloud4PSP need not be online at the same time. Queues reliably store prediction requests as messages until the Cloud4PSP is ready to process them. Then, Cloud4PSP can be adjusted and scaled out according to the current needs. As the depth of the prediction request queue grows, more computing resources can be provisioned. Therefore, such an approach allows money to be saved, taking into account the amount of infrastructure required to service the application load. Moreover, the queue-based approach allows load balancing - as the number of requests in a queue increases, more instances of the PredictionWorker role can be added to accelerate the prediction. In both systems the prediction process can be measured and end-users can be charged based on some metrics, such as the amount of computing instances, storage requirements, and data transfers.

On a critical note, we have to admit that, although Cloud4PSP implements a relatively new algorithm for 3D protein structure prediction, which is based on Monte Carlo simulations and structure refinement, at the current stage of development, it is unable to catch up Rosetta@Cloud in terms of the wealth of deployed software for predicting protein structures. However, the advantage of the WZ method used in the Cloud4PSP is that it eliminates the drawbacks of widely used gradient-based methods and this brings it closer to finding the global minimum of energy function.

Implementation of other prediction methods remains in the perspective of system development in the future. Advanced users may also extend the system by adding their own, new prediction methods. We also have to mention that our tests were performed for the prediction of small molecular structures. For large proteins, the number of iterations needed for modeling the structure increases exponentially, and the process requires more computational resources than those which we had during our tests.

5 Summary

Predicting protein structures from the beginning using the *ab initio* methods is one of the most challenging tasks for modern computational biology. This requires huge computing resources to allow us to perform calculations in parallel in order to reduce the prediction time. Cloud computing provides theoretically unlimited computing resources in a pay-as-you-go model. This is very important feature of the cloud architecture. Clouds provide an important computing power that can be provisioned on-demand and according to particular needs, which perfectly fits the character of the prediction process.

6 Availability

Cloud4PSP software is available at the project home page http://zti.polsl.pl/w3/dmrozek/science/cloud4psp.htm. The system can be used by the academic society for free, i.e., the software is free, but users have to lease the Cloud hardware infrastructure to use the system for their own costs. Users will have to configure and publish the system on the Azure cloud. Configuration details are provided at the Cloud4PSP project home page.

Further development of the system will be carried out by the Cloud4Proteins non-profit, scientific group (http://www.zti.aei.polsl.pl/w3/dmrozek/science/cloud4proteins.htm).

Acknowledgments We would like to thank Microsoft Research for providing us with free access to the computational resources of the Microsoft Azure cloud under the Microsoft Azure for Research Award program.

References

1. Angiuoli, S., Matalka, M., Gussman, A., Galens, K., et al.: CloVR: a virtual machine for automated and portable sequence analysis from the desktop using Cloud computing. BMC Bioinf. **12**, 356 (2011)
2. Arnold, K., Bordoli, L., Kopp, J., Schwede, T.: The SWISS-MODEL workspace: a web-based environment for protein structure homology modelling. Bioinformatics **22**(2), 195–201 (2006)
3. Berman, H., et al.: The Protein Data Bank. Nucleic Acids Res. **28**, 235–242 (2000)
4. Bertis, V., Bolze, R., Desprez, F., Reed, K.: From dedicated Grid to Volunteer Grid: large scale execution of a

bioinformatics application. J. Grid Comput. **7**(4), 463–478 (2009)
5. Bondi, A.: Characteristics of scalability and their impact on performance. In: 2nd International Workshop on Software and Performance, WOSP 2000, pp. 195–203 (2000)
6. Case, D., Cheatham, T., Darden, T., Gohlke, H., Luo, R., Merz, K.J., Onufriev, A., Simmerling, C., Wang, B., Woods, R.: The Amber biomolecular simulation programs. J. Comput. Chem. **26**, 1668–1688 (2005)
7. Chen, C., Huang, Y., Ji, X., Xiao, Y.: Efficiently finding the minimum free energy path from steepest descent path. J. Chem. Phys. **138**(16), 164122 (2013)
8. Chen, H.Y., Hsiung, M., Lee, H.C., Yen, E., Lin, S., Wu, Y.T.: GVSS: a high throughput drug discovery service of Avian Flu and Dengue Fever for EGEE and EUAsiaGrid. J. Grid Comput. **8**(4), 529–541 (2010)
9. Chivian, D., Kim, D.E., Malmström, L., Bradley, P., Robertson, T., Murphy, P., Strauss, C.E., Bonneau, R., Rohl, C.A., Baker, D.: Automated prediction of CASP-5 structures using the Robetta server. Proteins: Struct., Funct., Bioinf. **53**(S6), 524–533 (2003)
10. Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W., Kollman, P.A.: A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J. Am. Chem. Soc. **117**(19), 5179–5197 (1995)
11. De Vries, S., van Dijk, A., Krzeminski, M., van Dijk, M., Thureau, A., Hsu, V., Wassenaar, T., Bonvin, A.: HADDOCK versus HADDOCK: new features and performance of HADDOCK2.0 on the CAPRI targets. Proteins **69**, 726–733 (2007)
12. Edic, P., Isaacson, D., Saulnier, G., Jain, H., Newell, J.: An iterative Newton-Raphson method to solve the inverse admittivity problem. IEEE Trans. Biomed. Eng. **45**(7), 899–908 (1998)
13. Emeakaroha, V.C., Maurer, M., Stern, P., Łabaj, P.P., Brandic, I., Kreil, D.P.: Managing and optimizing bioinformatics workflows for data analysis in clouds. J. Grid Comput. **11**(3), 407–428 (2013)
14. Eswar, N., Webb, B., Marti-Renom, M.A., Madhusudhan, M., Eramian, D., Shen, M., Pieper, U., Sali, A.: Comparative Protein Structure Modeling Using MODELLER. Wiley, New York (2007)
15. Farkas, Z., Kacsuk, P.: P-GRADE portal: a generic workflow system to support user communities. Future Gener. Comput. Syst. **27**(5), 454–465 (2011)
16. Ferrari, T., Gaido, L.: Resources and services of the EGEE production infrastructure. J. Grid Comput. **9**, 119–133 (2011)
17. Fletcher, R., Powell, M.: A rapidly convergent descent method for minimization. Comput. J. **6**(2), 163–168 (1963)
18. Frishman, D., Argos, P.: Seventy-five percent accuracy in protein secondary structure prediction. Proteins **27**, 329–335 (1997)
19. Garnier, J., Gibrat, J., Robson, B.: GOR method for predicting protein secondary structure from amino acid sequence. Methods Enzymol. **266**, 540–53 (1996)
20. Gesing, S., Grunzke, R., Krüger, J., Birkenheuer, G., Wewior, M., Schäfer, P., et al.: A single sign-on infrastructure for science gateways on a use case for structural bioinformatics. J. Grid Comput. **10**, 769–790 (2012)
21. Gu, J., Bourne, P.: Structural Bioinformatics (Methods of Biochemical Analysis), 2nd edn. Wiley-Blackwell, Hoboken (2009)
22. Herrmann, T., Güntert, P., Wüthrich, K.: Protein NMR structure determination with automated NOE assignment using the new software CANDID and the torsion angle dynamics algorithm DYANA. J. Mol. Biol. **319**, 209–227 (2002)
23. Hovmöller, S., Zhou, T., Ohlson, T.: Conformations of amino acids in proteins. Acta Cryst. **D58**, 768–776 (2002)
24. Hung, C.L., Hua, G.J.: Cloud Computing for protein-ligand binding site comparison. Biomed. Res. Int., 170356 (2013)
25. Hung, C.L., Lin, Y.L.: Implementation of a parallel protein structure alignment service on Cloud. Int. J. Genomics **439681**, 1–8 (2008)
26. Hupfeld, F., Cortes, T., Kolbeck, B., Stender, J., Focht, E., Hess, M., et al.: The XtreemFS architecture - a case for object-based file systems in Grids. Concurrency Computat.: Pract. Exper. **20**(17), 2049–2060 (2008)
27. Insilicos: Rosetta@Cloud: macromolecular modeling in the Cloud. Fact Sheet. https://rosettacloud.files.wordpress.com/2012/08/rc-fact-sheet_bp5-en2a.pdf (2012). Accessed 9 March 2015
28. Jithesh, P., Donachy, P., Harmer, T., Kelly, N., Perrott, R., Wasnik, S., Johnston, J., McCurley, M., Townsley, M., McKee, S.: GeneGrid: architecture, implementation and application. J. Grid Comput. **4**(2), 209–222 (2006)
29. Jmol Homepage: Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org. Accessed 7 Sept 2015
30. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karóczkai, K., Márton, I.: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. J. Grid Comput. **10**(4), 601–630 (2012)
31. Kaján, L., Yachdav, G., Vicedo, E., Steinegger, M., Mirdita, M., Angermüller, C., Böhm, A., Domke, S., Ertl, J., Mertes, C., Reisinger, E., Staniewski, C., Rost, B.: Cloud prediction of protein structure and function with PredictProtein for Debian. BioMed Res. Int. **2013**(398968), 1–6 (2013)
32. Källberg, M., Wang, H., Wang, S., Peng, J., Wang, Z., Lu, H., Xu, J.: Template-based protein structure modeling using the RaptorX web server. Nat. Protoc. **7**, 1511–1522 (2012)
33. Kanaris, I., Mylonakis, V., Chatziioannou, A., Maglogiannis, I., Soldatos, J.: HECTOR: enabling microarray experiments over the hellenic Grid infrastructure. J. Grid Comput. **7**(3), 395–416 (2009)
34. Kelley, L., Sternberg, M.: Protein structure prediction on the Web: a case study using the Phyre server. Nat. Protoc. **4**(3), 363–371 (2009)
35. Kessel, A., Ben-Tal, N.: Introduction to Proteins: Structure, Function, and Motion. Chapman & Hall/CRC Mathematical & Computational Biology, CRC Press, Boca Raton (2010)
36. Kim, D., Chivian, D., Baker, D.: Protein structure prediction and analysis using the Robetta server. Nucleic Acids Res. **32**(Suppl 2), W526–31 (2004)
37. Kollman, P.: Advances and continuing challenges in achieving realistic and predictive simulations of the properties of organic and biological molecules. Acc. Chem. Res. **29**, 461–469 (1996)

38. Krampis, K., Booth, T., Chapman, B., Tiwari, B., et al.: Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community. BMC Bioinf. **13**, 42 (2012)
39. Laganà, A., Costantini, A., Gervasi, O., Lago, N.F., Manuali, C., Rampino, S.: COMPCHEM: progress towards GEMS a grid empowered molecular simulator and beyond. J. Grid Comput. **8**(4), 571–586 (2010)
40. Lampio, A., Kilpeläinen, I., Pesonen, S., Karhi, K., Auvinen, P., Somerharju, P., Kääriäinen, L.: Membrane binding mechanism of an RNA virus-capping enzyme. J. Biol. Chem. **275**(48), 37853–9 (2000)
41. Leach, A. Molecular Modelling: Principles and Applications, 2nd edn. Pearson Education EMA, Essex (2001)
42. Leaver-Fay, A., Tyka, M., Lewis, S., Lange, O., Thompson, J., Jacak, R., et al.: ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol. **487**, 545–74 (2011)
43. Lesk, A. Introduction to Protein Science: Architecture, Function, and Genomics, 2nd edn. Oxford University Press, NY (2010)
44. Lewis, S., Csordas, A., Killcoyne, S., Hermjakob, H., et al.: Hydra: a scalable proteomic search engine which utilizes the Hadoop distributed computing framework. BMC Bioinf. **13**, 324 (2012)
45. Lordan, F., Tejedor, E., Ejarque, J., Rafanell, R., Álvarez, J., Marozzo, F., Lezzi, D., Sirvent, R., Talia, D., Badia, R.M.: ServiceSs: an interoperable programming framework for the cloud. J. Grid Comput. **12**(1), 67–91 (2014)
46. McKendrick, J. Cloud computing market hot, but how hot? estimates are all over the map. http://www.forbes.com/sites/joemckendrick/2012/02/13/cloud-computing-market-hot-but-how-hot-estimates-are-all-over-the-map/ (2012). Accessed 24 Aug 2015
47. Mell, P., Grance, T.: The NIST definition of cloud computing. Special Publication 800-145. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (2011). Accessed 7 May 2015
48. Microsoft Azure Cloud Services Specification: Sizes for Cloud Services. https://azure.microsoft.com/pl-pl/documentation/articles/cloud-services-sizes-specs/. Accessed 7 May 2015
49. Microsoft Azure Cloud Services Specification: Sizes for virtual machines. https://azure.microsoft.com/pl-pl/documentation/articles/virtual-machines-size-specs/. Accessed 7 Sept 2015
50. Mrozek, D.: High-Performance Computational Solutions in Protein Bioinformatics. SpringerBriefs in Computer Science. Springer International Publishing (2014)
51. Mrozek, D., Kutyła, T., Małysiak-Mrozek, B.: Accelerating 3D protein structure similarity searching on Microsoft Azure Cloud with local replicas of macromolecular data. Parallel Processing and Applied Mathematics - PPAM 2015, Lecture Notes in Computer Science. Springer, Berlin Heidelberg (2015)
52. Mrozek, D., Małysiak-Mrozek, B., Kłapciński, A.: Cloud4Psi: cloud computing for 3D protein structure similarity searching. Bioinformatics **30**(19), 2822–2825 (2014)
53. Pierce, L., Salomon-Ferrer, R., de Oliveira, C., McCammon, J., Walker, R.: Routine access to millisecond time scale events with accelerated molecular dynamics. J. Chem. Theory Comput. **8**(9), 2997–3002 (2012)
54. Ponder, J.: TINKER - software tools for molecular design. Dept. of Biochemistry & Molecular Biophysics, Washington University, School of Medicine, St. Louis (2001)
55. Ramachandran, G., Ramakrishnan, C., Sasisekaran, V.: Stereochemistry of polypeptide chain configurations. J. Mol. Biol. **7**, 95–9 (1963)
56. Rost, B., Liu, J.: The PredictProtein server. Nucleic Acids Res. **31**(13), 3300–3304 (2003)
57. Schatz, M.C.: CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics **25**(11), 1363–1369 (2009)
58. Schwieters, C., Kuszewski, J., Tjandra, N., Clore, G.: The Xplor-NIH NMR molecular structure determination package. J. Magn. Reson. **160**, 65–73 (2003)
59. Shanno, D.: On Broyden-Fletcher-Goldfarb-Shanno method. J. Optimiz Theory Appl., 46 (1985)
60. Shaw, D.E., Dror, R.O., Salmon, J.K., Grossman, J.P., Mackenzie, K.M., Bank, J.A., Young, C., Deneroff, M.M., Batson, B., Bowers, K.J., Chow, E., Eastwood, M.P., Ierardi, D.J., Klepeis, J.L., Kuskin, J.S., Larson, R.H., Lindorff-Larsen, K., Maragakis, P., Moraes, M.A., Piana, S., Shan, Y., Towles, B.: Millisecond-scale molecular dynamics simulations on Anton. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, pp. 39:1–39:11. ACM, New York (2009)
61. Shen, Y., Vernon, R., Baker, D., Bax, A.: De novo protein structure generation from incomplete chemical shift assignments. J. Biomol. NMR **43**, 63–78 (2009)
62. Shirts, M., Pande, V.: COMPUTING: screen savers of the world unite! Science **290**(5498), 1903–4 (2000)
63. Söding, J.: Protein homology detection by HMM-HMM comparison. Bioinformatics **21**(7), 951–960 (2005)
64. Streit, A., Bala, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., et al.: Unicore 6 - recent and future advancements. JUEL, 4319 (2010)
65. Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A., Berendsen, H.: GROMACS: fast, flexible, and free. J. Comput. Chem. **26**, 1701–1718 (2005)
66. Warecki, S., Znamirowski, L.: Random simulation of the nanostructures conformations. In: Proceedings of International Conference on Computing, Communication and Control Technology, The International Institute of Informatics and Systemics, Austin, Texas, vol. 1, pp. 388–393 (2004)
67. Wassenaar, T.A., van Dijk, M., Loureiro-Ferreira, N., van der Schot, G., de Vries, S.J., Schmitz, C., van der Zwan, J., Boelens, R., Giachetti, A., Ferella, L., Rosato, A., Bertini, I., Herrmann, T., Jonker, H.R., Bagaria, A., Jaravine, V., Güntert, P., Schwalbe, H., Vranken, W.F., Doreleijers, J.F., Vriend, G., Vuister, G., Franke, D., Kikhney, A., Svergun, D.I., Fogh, R.H., Ionides, J., Laue, E.D., Spronk, C., Jurkša, S., Verlato, M., Badoer, S., Dal Pra, S., Mazzucato, M., Frizziero, E., Bonvin, A.M.: WeNMR: structural biology on the Grid. J. Grid Comput. **10**(4), 743–767 (2012)
68. Wu, S., Skolnick, J., Zhang, Y.: Ab initio modeling of small proteins by iterative TASSER simulations. BMC Biol. **5**(17) (2007)

69. Xu, D., Zhang, Y.: Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. Proteins **80**(7), 1715–35 (2012)
70. Xu, J., Li, M., Kim, D., Xu, Y.: RAPTOR: optimal protein threading by linear programming, the inaugural issue. J. Bioinform Comput. Biol. **1**(1), 95–117 (2003)
71. Yang, Y., Faraggi, E., Zhao, H., Zhou, Y.: Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. Bioinformatics **27**(15), 2076–2082 (2011)
72. Zhang, Y.: Progress and challenges in protein structure prediction. Curr. Opin. Struct. Biol. **18**(3), 342–348 (2008)
73. Znamirowski, L.: Non-gradient, sequential algorithm for simulation of nascent polypeptide folding. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M., Dongarra, J.J. (eds.) Computational Science - ICCS 2005, Lecture Notes in Computer Science, vol. 3514, pp. 766–774. Springer, Berlin Heidelberg (2005)