doi:10.20944/preprints202109.0275.v1

Big Data Approach to Large Scale Molecular Dynamics Simulations: Necessity and Inevitability for Drug Design

B.S. Sanjeev* and Dheeraj Chitara

Dept. of Applied Sciences, Indian Institute of Information Technology - Allahabad, Prayagraj 211012, U.P., India. sanjeev@iiita.ac.in

Abstract. Molecular Dynamics (MD) simulations model motion of molecules in atomistic detail and aid in drug design. While simulations on large systems may require several days to complete, analysis of terabytes of data generated in the process could also be time consuming. Recent studies captured exciting and dramatic drug-receptor interactions under cell-like complex conditions. Such advances make simulations of biomolecular interactions more realistic, insightful, and informative and have potential to make drug design more realistic. However, currently available resources and techniques do not provide, in reasonable time, a comprehensive understanding of events seen in simulations. We demonstrate that big data approach results in significant speedups, and provides rapid insights into simulations performed. Advancing this improvement, we propose a scalable, self-tuning, and responsive framework based on Cloud-infrastructure to accomplish the best possible MD studies with given priorities and within available resources.

Keywords: Cloud infrastructure · Spark · Molecular Dynamics simulations · Drug design.

1 Introduction

Molecular Dynamics (MD) simulation is a powerful technique to study molecules in atomistic detail. It models the movement of atoms, ions and molecules subject to forces acting on them. This *evolution* of the system is captured as a sequence of *frames*. A frame holds the necessary details of the system (such as coordinates of atoms) corresponding to the time it represents. Processing such data provides insights into various facets of the simulated system.

In biological systems, MD simulations allow one to study a variety of phenomena such as protein folding [20], stability [14] and intermolecular interactions [3]. In particular, they are of great value in drug design and drug discovery, providing vital inputs on ligand docking, virtual screening, dynamics of drug-bound protein/receptor complexes, allosteric modulation, and role of water molecules in drug binding [5, 16]. Software, such as AMBER [17], are available to carry out MD simulations. The *length* of a simulation (in terms of time) needs to be optimal for meaningful interpretation of data. Smaller systems generally attain equilibrium faster, and larger systems require longer simulations. This places severe stress on simulating large systems, often resulting in runtimes that range between a few days to weeks at a time. Analysis of voluminous data produced in the process presents another challenge that can benefit from big data approach. While designing a drug *in silico* in itself is not an expensive task, the final hurdle that besets the acceptance of the drug involves the clinical trails. Here, the environment of the drug-receptor complex is different from the theoretical models. As recent a study shows, developing a new drug costs pharmaceutical companies about a billion dollars in investments [21]. Evidently, it is critical to improve efficiency of drug development at the theoretical stage itself.

In this paper, we demonstrate the necessity for a new framework in the study of MD simulations. Section 2 introduces the chosen simulated systems and the parameters for study. Section 3 demonstrates that big data approach can be adopted to the analysis of MD simulation data to gather rapid insights. Based on this advance, we present a Spark-based, self-tuning, flexible and scalable framework, deployable on public as well as on private Cloud infrastructure, that provides speedy insights of immense value in drug design. Section 4 concludes the paper.

2 Materials and Methods

Simulation data from two biological systems were used in this study. They included proteins, polynucleotides (such as miRNA and mRNA), drugs, and other biomolecules. In a nutshell, proteins are polymers made of 20 types of units called amino acids (see **Fig. 1**). miRNA and mRNA are polymers made of 4 types of units called nucleotides and are hence called polynucleotides. Both proteins and polynucleotides take different shapes in 3-D space which allow them to perform biologically meaningful activities. Certain calculations on proteins use only C_α atoms to represent the corresponding amino acids. Often two or more polymers of amino acids combine to form a single functional protein.

Of the two systems that were selected for the study, the first consisted of a human protein called Argonaute2 (Ago2) that is bound to one miRNA and another mRNA [4]. The second system consisted of Main protease (M^{pro}) of SARS-CoV-2 bound to 3 different drugs, viz., Elbasvir, Glecaprevir, and Ritonavir [18]. As **Figure 2** schematically shows, M^{pro} is a dimer (ie., a protein made of two amino acid polymers). Hence, for reasons beyond the scope of this work, 5 units of these 3 types of drugs were *docked* (ie., bound) to M^{pro}. While docking study identifies *where* drugs may bind on a protein, MD simulations help to identify *if* such a complex is viable. A biological cell packs a large variety of moelcules in a small volume, giving rise to a phenomenon called molecular crowding. This effect significantly influences dynamics and interactions between molecules [7, 11, 24]. However, both experimental and theoretical studies routinely neglect the effect of other (crowder) molecules on the system of interest due to complexities and challenges involved in such exercises. To study the effect of crowding, M^{pro} system had 8 *crowder proteins* and 170 other *crowder* metabolites as part of its environment. Ago2, however, was simulated in aqueous environment.

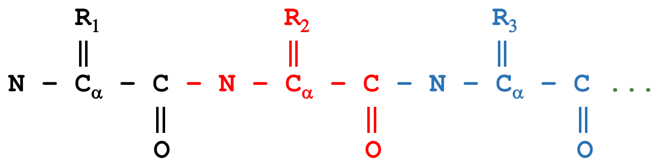

Fig. 1. Protein is a polymer made of 20 types of amino acids. Chemically, they differ at C_α atoms shown as R_is. Hydrogen atoms are not shown.

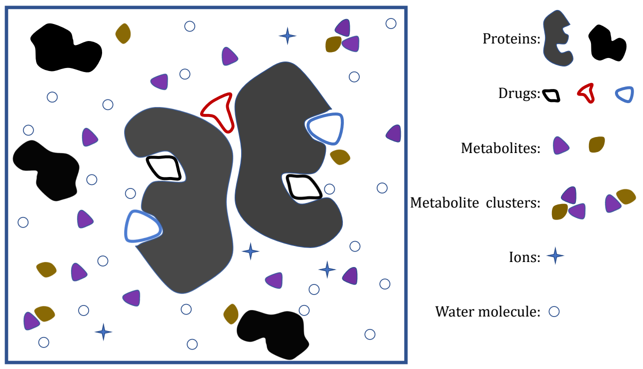

Fig. 2. A schematic model of M^{pro} system with 185,255 atoms. The more realistic model for simulation involved not only system of interest (M^{pro} protein bound to 5 drugs of 3 types) but also the *crowded* environment. This environment consisted of (other) proteins, metabolites seen naturally in cells, ions, and water. Some metabolites were seen strongly interacting with drugs during the course of simulation.

Analyzing dynamics of the simulated system involves processing numerous frames of data generated by MD simulation. Details of the system at a given time (such as coordinates and sizes of the simulated box) are captured through a frame associated with it. Sequence of frames thus reproduce evolution of system modelled by simulation. While simulations run over hundreds of nanoseconds (ns), simulated data is typically stored (as frames) at intervals of 1 picosecond (ps). The composition of the two chosen systems is given in Table I. In M^{pro} complex, 8 streptococcal protein Gs (each with 56 amino acids) were used as protein crowders along with 170 metabolites of 10 different types. A system should have a net charge of 0, hence ions were added as necessary. Details such as number of atoms, total frames, and the sizes of input data are given in Table II. Other parameters are covered in later part of this section.

Table 1. Composition of MD simulations of Ago2 and M^{pro} complexes is given in terms of amino acids (aa.) for proteins and in terms of nucleotides (nt.) for miRNA/mRNA. Besides ions, crowders (Cr.) comprising of proteins and metabolites also were incorporated in M^{pro} system.

System	Component	Composition	# atoms
Ago2 Complex	Protein	801 aa.	12914
	miRNA	21 nt.	656
	mRNA	10 nt.	323
	Cl^- ion	4	4
M^{pro} Complex	Protein	608 aa.	9257
	Drugs	5	540
	Cr. proteins	8	6840
	Cr. metabolites	170	3524
	K^+ ion	190	190

Table 2. Overview of the two systems used for analysis. From all candidate water networks (WNw^c), only those with minimum occurrence (ie., 40 %) were filtered (WNw).

Item	Ago2 complex	M^{pro} complex
atoms	175,105	185,255
polar atoms (P_i)	2,613	4,076
water molecules (W_i)	53,736	54,968
frames	500,000	200,000
WNw^c	488,698	3,261,310
WNw	1,000	1,317
Spark partitions	7,143	2,000
data size (GB)	1,050	445

Apache Spark is a unified analytics engine, developed for big data applications analytics [23, 8]. The Spark-based suite of programs, called SparkTraj, was written in Scala [12] as proofs-of-concept implementation for scalable computation on MD data. Three parameters were computed for the evaluation of performance, viz., (a) radius of gyration (RoG), (b) molecular contacts (MCon), and (c) water networks (WNw). CPPTRAJ [15] is the standard accompanying tool of AMBER package. It was used as benchmark implementation for serial version to compute radius of gyration (RoG) of C_α atoms. MCon is the total pairs of atoms that are within a cutoff distance of 5 Å, with no equivalent serial implementation. Ankush [19] is a program to find water networks. The preprocessing overheard associated with each frame in serial version was neglected for benchmarking. The three parameters are of interest for different reasons as discussed in Section III, under **Benchmarked Parameters**.

Radius of Gyration: The compactness of a protein can be measured using radius of gyration, which would increase if the protein begins to unfold. In the current work, this was equivalent to finding the root-mean-squared deviation of C_α atoms of either Ago2 or M^{pro}.

$$\text{RoG} = \sqrt{\tfrac{1}{N}\sum_{i=1}^{N}(r\_\bar{r})^2}$$

Molecular Contacts: Typical MD simulations consider only biomolecule(s) of interest apart water molecules and ions while cellular environment comprises of many other molecules, influencing the dynamics of system of interest. Under such circumstances, it would be necessary to find other compounds (such as metabolites) getting in proximity with molecules of interest. As shown in **Figure 3**, this information was captured using MCon algorithm that finds the (unique) pairs of atoms within a distance of 5 Å.

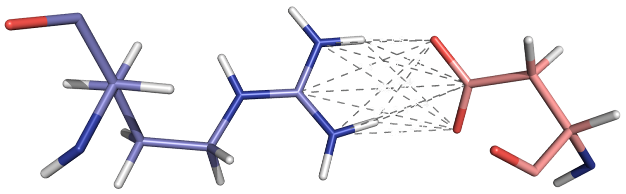

Fig. 3. Due to crowded environment within cells, molecules often bump into one another. The number of pairs of atoms that are within a distance of 5 Å is called as molecular contacts (MCon) which reflect proximity of molecules. The figure shows a few such contacts represented with dotted lines.

Water networks: A water network is a set of polar atoms (ie., oxygen/nitrogen atoms of solute shown as P_i in Table II) that are within 3.5 Å distance with a common water molecule. As shown in **Figure 4**, some networks such as $\{P_1,P_2\}$ are formed independently by different water atoms. Larger networks such as $\{P_a,P_b,P_c,P_d\}$ give rise to smaller networks such as $\{P_b,P_d\}$ and $\{P_a,P_c,P_d\}$. Occurrence (occ.) of a water network is defined as the number of frames in which a given network is seen. From all observed water networks (WNw^c in Table II), we *selected* for benchmarking water networks with a minimum occurrence of 40 % (WNw in Table II), though lesser cutoffs were used for the study. *Trajectory*

doi:10.20944/preprints202109.0275.v1

for each WNw, which shows its presence or the absence in each frame ordered by time, was computed. Using trajectory, Maximum Residence Time (MRT), defined as the highest number of sequential frames in which a particular WNw continues to exist, was also computed.

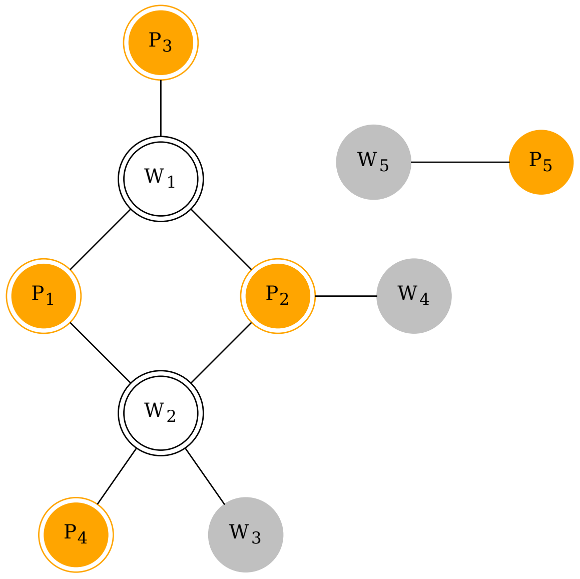

Fig. 4. Two nodes (ie., atoms) are connected if, (a) their distance is within 3.5 Å, and (b) if least one node is oxygen of water. A water network is a set of 2 or more nitrogen/oxygen atoms of solute connected to a common water molecule. The networks seen here are $\{P_1,P_3\}$, $\{P_2,P_3\}$, $\{P_1,P_2,P_3\}$ through W_1, $\{P_1,P_4\}$, $\{P_2,P_4\}$, $\{P_1,P_2,P_4\}$ through W_2, and $\{P_1,P_2\}$ independently through W_1 and W_2.

It is necessary to not just process but also to read data in parallel to obtain meaningful scaling of computation when size of data is large. To this end we used Apache Spark (referred also as 'Spark') with Scala. Spark is an open-source distributed general purpose framework that is designed for large-scale data analysis and offers implicit data parallelism with fault tolerance [8]. AMBER generates data in NetCDF format [9]. As Spark does not have native support for NetCDF format, SciSpark was used to read the simulation data. SciSpark is a framework developed by NASA and UCLA for applications in earth and space sciences [13]. The basic approach for computation we used was to read the MD data into SciSpark Resilient Distributed Datasets (RDDs) [22] first, followed by computation of necessary parameters through Spark Datasets and DataFrames. SciSpark supports HDFS [1] as well as native Linux file systems.

NVIDIA Tesla V100 based GPU nodes were used for simulations. For Spark calculations, a cluster with 20 data nodes was used. GRIDScaler SFA7700X appliance with 15 GB/s of throughput was the storage server for the two systems. GPU nodes and data nodes of the cluster have two Xeon Gold 6148 CPUs with 384 GB of RAM. The big data cluster supported Apache Spark (ver. 2.3.0) and Scala (ver. 2.11.8) for computations.

3 Results and Discussion

3.1 Spark-based processing of MD simulation data

AMBER18 was used for simulating the chosen systems, which saves output in one or more *mdcrd* files in NetCDF format. Based on the user input, one or more frames can be written in each mdcrd file. NetCDF format allows parallel reading of a single file. To work within the limitations of HDFS block size [2], only a limited number of frames per file were stored. Each mdcrd file of Ago2 and M^{pro} systems stored data of 10 frames and 50 frames respectively. The detailed description of a simulated system (such as names of atoms, various molecules, types of bonds, etc.) are stored separately in another file, called *parmtop*.

Apache Spark is a framework designed for massive in-memory data processing with lazy evaluation. This framework was used to process the simulation data. To implement efficient solutions, both Spark operations as well as sequence of their application should be carefully chosen. Sub-optimal approach may lead to *data shuffling* (ie., movement of massive data across nodes due to its redistribution) causing delays. It may even lead to expensive recomputations of Spark's RDD. Hence, we developed the solutions to circumvent these significant barriers. Further, generation and use of intermediate data written on storage, another potential rate-limiting step, was also avoided.

Benchmarked parameters: Three parameters were chosen to constitute the proof concept implementations. The first was Radius of Gyration (RoG), a simple calculation that is equivalent to finding root-mean-square deviation of about 800 3-D coordinates. Considering that both systems have over 175,000 atoms, this is a trivial yet useful calculation of a parameter that reflects the compactness of a protein. From an algorithmic point of view, the time taken would reflect the data throughput available for Spark from the storage server. The second chosen parameter was Molecular Contacts (MCon) between two sets of atoms. Due to crowding, molecules bump into each other. This brings many atoms close to each other, and two atoms are said to have *contact* if they are within a distance of 5 Å. Large number of such contacts over a significant time signals proximity that may be of significance. To identify crowders (metabolites and proteins) that are in proximity of drugs during the simulation, MCon was deemed necessary. It was not available in CPPTRAJ or as any another software, to the best of our knowledge. Computing MCon required examination of 100 million possible contacts in each of 200,000 (M^{pro}) to 500,000 (Ago2) frames. Determining MCon requires a single pass over all frames. The third parameter chosen for benchmarking was the determination of Water Networks (WNws). Unlike typical parameters computed by CPPTRAJ (such as distance, hydrogen bonds, etc.), conceptually this requires two passes over each frame, with *each* pass producing significant amount of data. While the first traversal finds all possible networks followed by their occurrence, the second pass finds *trajectories* of all the selected water networks. The algorithms used for computation of the three parameters are given below.

Radius of Gyration: CPPTRAJ [15] was used to compute the radius of gyration for benchmarking of serial variant. It is the standard accompanying software for AMBER package and is used to compute a wide variety of parameters from simulation data. **Algorithm 1** shows the approach used for the Spark version. One should first note that if a system has N atoms, it has $3N$ coordinates (ie., 3-D coordinates x_1, y_1, z_1,...,x_N, y_N, z_N). Details of the system were read from the *parmtop* file (as discussed in the previous section). Then, the mdcrd files were read parallelly into a Resilient Distributed Dataset (RDD) called mdRDD loaded through SciSpark. mdRDD was then *flattened* into another RDD, called frRDD, so that every element was now a tuple that contained information of a particular frame in the form of ($time_i$, $box_{i,x}$, $box_{i,y}$, $box_{i,z}$, $x_{i,1}$, $x_{i,2}$, ... $x_{i,3N}$), where i is the frame number. Then, a Spark Dataset, with a method to find RoG for a frame, was created using frRDD. Invoking this method, RoGs of all frames are computed, data collected into a Scala list, and then sorted w.r.t. the time of the frames. Sorting by time was not done using Spark itself in any of the implementations for reasons discussed later in this section.

Algorithm 1: Calculation of Radius of Gyration

```
read topology from parmtop
create $mdRDD$ from mdcrds
obtain $time$, $RoG$ through:
    flattening $mdRDD$ into $frRDD$
    creating Dataset using additional parameters
    invoking Dataset's method for RoG
sort $RoG$ w.r.t. $time$
```

Molecular Contacts: Benchmarking of molecular contacts was not done serially as no equivalent implementation was available. As shown in **Algorithm 2**, much of the procedure to compute molecular contacts was similar to that of the radius of gyration, with two differences. The first difference was that the output from each frame was not a number (like RoG) but instead a string, which, when split, gave triplets (r_i,r_j,cnt) where r_ks were residue numbers and *cnt* was the number of contacts they shared. Secondly, while computing RoG was a (relatively) trivial task, MCon required finding contacts between two large sets of atoms. M^{pro} system, for instance, has 9,796 atoms for the M^{pro}-drug complex and crowder proteins and metabolites constitute 10,364 atoms. This translates to over 101 million pairs of atoms between them. Clearly, a brute force method of counting is prohibitively expensive. Consequently, only potentially feasible pairs of atoms were scanned. A 2-D illustration of the idea is shown in **Figure 5**. First, we scanned every atom from one set and placed them in all possible (ie., 9) squares where they may have contacts. Then, given an atom of interest from the other set, say x_{25}, we can immediately examine the list of atoms corresponding to the square. This approach allows us to linearize the computation time to find MCon as physically each square can only have a certain number of atoms at any time. In fact, the time taken for calculating MCon was comparable to that of RoG!

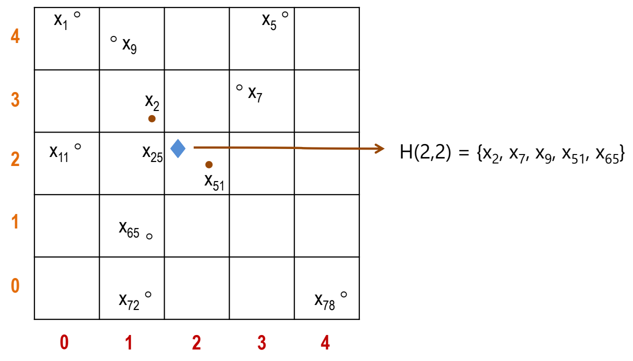

Fig. 5. Diamond shows the atom of interest, circles are other atoms, and only those represented with filled circles have relevant contacts. If we discretize space into squares with size cut-off distance (d) itself, examination of 9 squares is adequate to find potential contacts of an atom. A hash data structure, with tuples made of its discretized coordinates as key, can store all potential atoms that need to be examined for any atom in a given square. Same approach can be extended to 3-D for speedy computation of molecular contacts and water networks.

Water networks: Water networks were benchmarked against the serial version that works in two stages. In the first stage, it generates 2 files per frame; one that contains the contacts between polar atoms and water molecules, and the other between water molecules themselves. In the second stage, it processes these files to determine water networks. The first stage uses a C++ program and the second stage a Perl script. While this version worked well when it was originally developed, currently it is too inefficient for large systems with hundreds

Algorithm 2: Calculation of Molecular Contacts

```
read topology from parmtop
create $mdRDD$ from mdcrds
obtain $time, \{MCon\}$ through:
    flattening $mdRDD$ into $frRDD$
    creating Dataset using additional parameters
    invoking Dataset's method for $MCon$
sort $MCon$ w.r.t. $time$
```

of thousands of frames. Hence, as shown in **Algorithm 3**, a Spark version was developed to update it.

Unlike the previous two algorithms, the study of water networks conceptually requires two passes over all frames. In the first pass, the set of all water networks ever seen, called candidate water networks (WNw^c), were found. In the second pass, networks were further filtered based on their occurrence (WNw). RDDs have a tendency to recompute. To prevent recomputations from slowing down the process, we cached water networks found for each frame in the first pass. Once networks of interest were selected, their trajectories (WNwTraj) and maximum residence times (MRT) were computed from the cached data. Network trajectories can be stored if desired. However, we found that it was good enough to just have the summary featuring occurrence, MRT, and the window in which MRT was observed. As discussed in the case of MCon, the hash-list approach described through **Figure 5** was applied to find interacting polar atoms of each water. First, all polar atoms were scanned and placed in lists corresponding to cuboids where they may have contacts. Then, for each water, the corresponding list of its hash was scanned. Once all connected atoms for any particular water were known, all subsets (ie., networks) with at least 2 elements were also created. We found that when sampled only 20 % of the frames of Ago2 at regular intervals, the occurrence was almost identical to that of 100 % sampled data.

Algorithm 3: Water network calculations

```
read topology from parmtop
create $mdRDD$ from mdcrds
create and cache $nwDF$ through:
    flattening $mdRDD$ into $frRDD$
    creating Dataset through additional parameters
    obtaining $time, \{WNw^c\}$ through Dataset's method
create $\{WNw\}$ using $nwDF$ and occ. cutoff
compute $WNwTraj$ using $WNw$ and $nwDF$
```

A vital aspect of calculating distance between atoms involves taking into account the *periodic boundary condition* of the simulation box. It means that during simulations, a molecule does not face a wall at the edge of the box. Instead, it can move transparently from one side, say from right in **Figure 2**, and appear from the left. (The rationale is beyond the scope of this work.) Hence, the two metabolite clusters seen here are not far away but are much closer. The size of the simulation box itself may change during simulation and hence frame-wise box lengths are stored in mdcrd files.

3.2 Benchmarks and insights

The chosen systems for the proof-of-concept implementations, viz., Ago2 and M^{pro} complexes, were benchmarked. Given the necessity for consistency besides and substantial resource requirements, three variants were tested on the DDN storage solution that is GPFS-based. These were, (a) serial version (serial), (b) Spark-version with data on native GPFS filesystem (gpfs), and (c) Spark-version with data on HDFS (hdfs) filesystem. HDFS storage was available from the same GPFS server through a connector. Times of completion for the three parameters are given in **Table III**.

Table 3. Time taken for completion of any job in serial mode (serial) is the longest, compared to Spark when input was read from native GPFS filesystem (gpfs) or HDFS filesystems (hdfs). All times are in seconds.

Parameter	Ago2 complex			M^{pro} complex		
	serial	gpfs	hdfs	serial	gpfs	hdfs
RoG	8,342	1,653	324	1,931	584	131
MCon	-	-	-	-	584	176
WNw	508,750	1,664	500	251,010	614	218

It is clear from **Table III** that performance differs significantly when the input is read from HDFS, instead of the gpfs filesystem. In particular, *speedup*, as defined in **Equation (1)**, shows that close to 5 times speedier computation can be done if input is read from HDFS instead of gpfs filesystem.

$$speedup = \frac{time_{serial}}{time_{spark}} \tag{1}$$

Speedups over 1000 were obtained for water network calculation. As the serial version could not be run over the entire trajectory, after confirming with shorter runs (with up to 20 % of data), linear scaling was applied to estimate the time required for computations. MCon, another important parameter that requires only a single pass over each frame and is critical for the analysis of crowder-based simulations, performed faster than that WNw. While it could be argued that serial version for water network could be improved, it is evident that the times for completion of MCon and WNw are comparable to that of RoG, with all of them requiring just a few minutes to complete their tasks. While CPPTRAJ required 8,342 s to compute *RoG*, Spark (hdfs) for *WNw* was completed within 500 s, i.e., 16 faster for Ago2 over 1 TB of data with 500,000 frames! The RoGs of the two systems are shown in **Figure 6**). The values show that no unfolding of proteins occurred during the simulation.

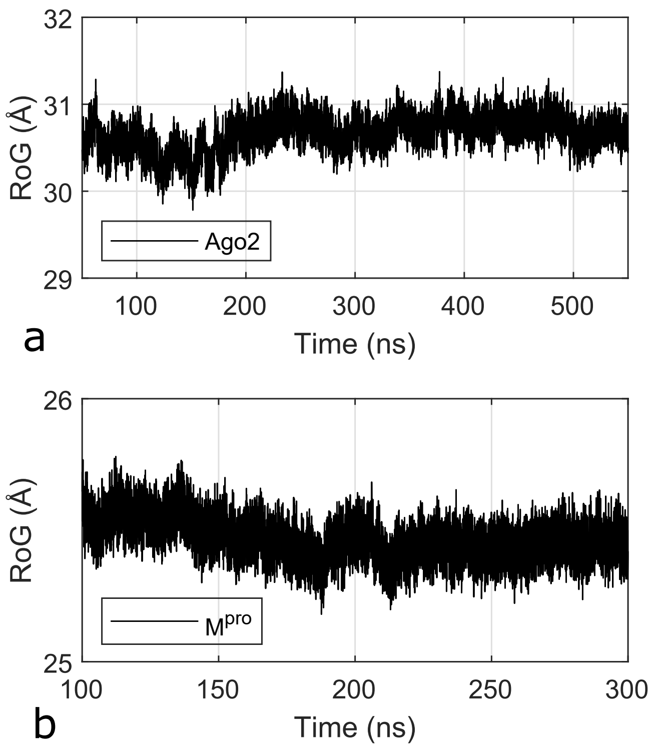

Fig. 6. The radii of gyration for Ago2 and M^{pro} were stable, indicating that no unfolding of these proteins happened during course of simulations.

Interesting events were captured using MCon however. As **Figure 7** shows, multiple metabolites were seen interacting with drugs. The most significant interactions were seen with only two drugs, a Ritonavir and Glecaprevir. With further analysis based on this information, a cluster of 5 metabolites were identified that were seen blocking the movement of this drug, perhaps for the first time in any simulation! Ordinarily, such long-duration associations occur due to hydrogen bonds, salt bridges, π-π interactions, etc. Here, no long- lasting hydrogen bonds were seen as these metabolites were mobile, though still in the vicinity of the drug during the entire course of simulation! Such interactions need to be computed and a parameter such as MCon is very useful for the same.

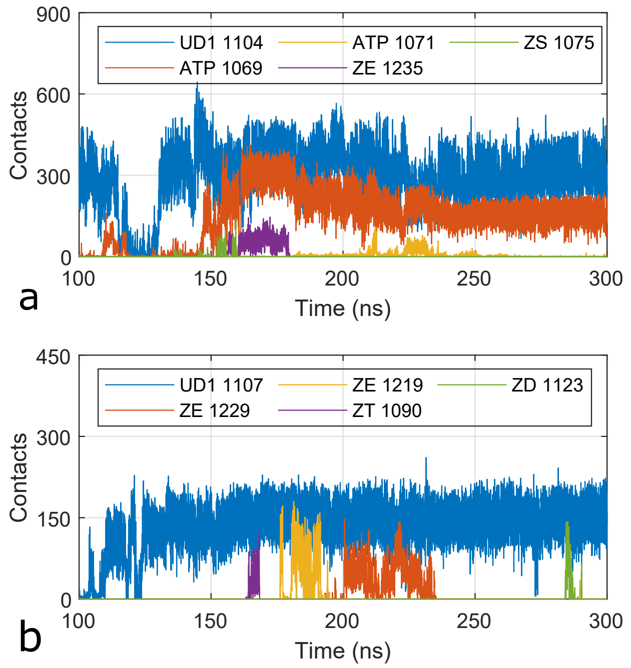

Fig. 7. MCon is useful to identify metabolites that are in proximity. Two out of five drugs (one of the two Ritonavirs and Glecaprevir) had consistent interactions with metabolites. Only the top 5 interacting metabolites are shown.

As already discussed in the Introduction, water networks (WNw), especially those that stabilize drug binding to proteins and other receptors, are of interest. With the current study, we have been able to quickly identify such water networks. As shown in **Figure 8**, water networks were found bridging protein residues (ILE 712, GLN 713, and GLY 714) in Ago2 complex with a nucleotide (C 806) of miRNA. Even more interestingly, water networks were also seen between drugs and M^{pro}. As shown in **Figure 8b**, Glecaprevir (GLC 612) is seen to be sharing a through-water network with two protein residues (ASP 593 and ARG 435).

Another crucial aspect of benchmarking software would be to find how the time for completion *scales* with size of the data. Ideally, processing time being same for each frame, increasing number of frames should lead to an increased time for completion by the same factor. For a large system of terabyte size such

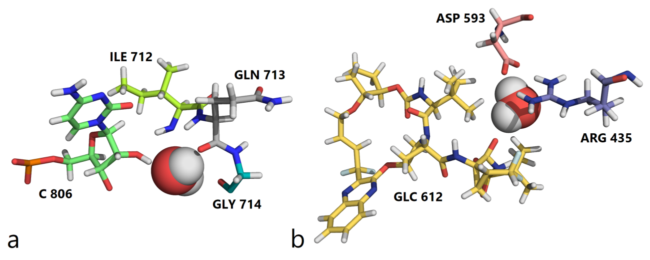

Fig. 8. The figure shows Water Network (WNw) found in each of the two systems. (a) WNw between a nucleotide (C 806) of miRNA and three amino acids of Ago2. (b) WNw present between a drug (GLC 612) and two amino acids of M^{pro}.

as Ago2, increasing input size by 10-fold (ie., 50K vs 500K frames) matched the expected performance where gpfs filesystem was used (see **Table IV**). In case of M^{pro} system (20K vs 200K frames), the escalations were much smaller. It was very interesting that the escalation of times were less than half in case of HDFS environment w.r.t. gpfs filesystem, exhibiting higher performance.

Table 4. With 10-fold increase of data frames, escalation in times of completion was almost by the same factor for the larger Ago2 system when data was read from gpfs filesystem, and by about half in case of HDFS. This increment was much lesser in case of M^{pro}, the smaller system.

Parameter	Ago2 complex		M^{pro} complex	
	gpfs	hdfs	gpfs	hdfs
RoG	9.2	4.2	6.0	2.3
MCon	-	-	6.6	2.6
WNw	8.9	4.7	6.0	2.9

To improve the performance of Spark implementation, many variations were considered and examined. For instance, sorting (by time) using Spark was less efficient due to data shuffling. So the data was first collected and was sorted directly using Scala. While we contemplated different ways to improve the performance further, it was not pursued to avoid possible over-engineering; reckon that time required to read mdcrds was at least 60 % of the total runtime.

To appreciate the immense value and contribution of such studies, it would be pertinent to reckon a few insights from the simulations of M^{pro} complex. Apart from metabolites and metabolite clusters, identified initially through MCon, the simulations in crowded environment yielded other fascinating insights as well. They include, possible preference of certain metabolites to particular sites, movement of a free amino acid in a probable (drug) binding site, and *crawling* of a drug over the surface from one pocket to another in presence of crowders (both proteins and metabolites). Such movement was not seen in other simulations and drugs remained stable at their docked sites during the course of simulations.

3.3 Framework for Cloud-based MD Simulation Service

The unique advantage of Cloud infrastructure over typical high-performance computing servers is the ability to scale resources on demand. However, the operating costs of Cloud infrastructure vary based on type, quantity and duration [6, 10]. We present a framework to expedite the best MD simulations studies possible within the available resources.

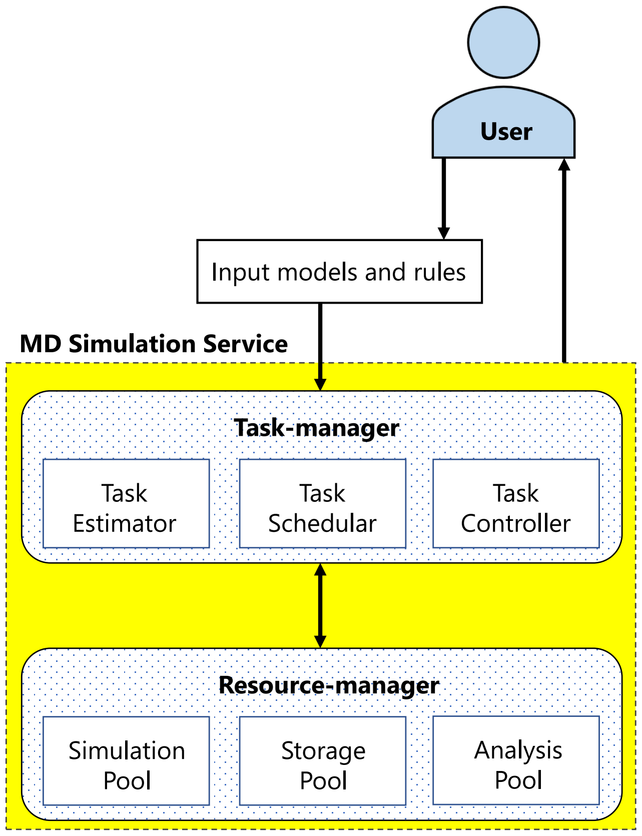

Fig. 9. Block diagram of the proposed framework for MD studies. Task-manager estimates task size, generates an execution plan, and in concert with Resource-manager, dynamically orchestrates tasks and resources. Periodic updates and final results are sent to user for information and feedback.

As shown in **Figure 9**, given the models to simulate and the accompanying rules (such as given priorities and available funds), Task Estimator first runs sample jobs to assess the time required on various platforms, determines the number, types of servers, infrastructure, and services that could be acquired within given constraints. Task Schedular then prepares a schedule of jobs to be submitted. In certain cases there could more specific rules. For instance, consider the case of *screening* of drug-candidates. A user may require only those drug-candidates that retained interactions with the target protein throughout the simulation(s). This could be monitored by Task Controller that can preemptively terminate simulations in which drugs lose interactions. Other reasons to pre-emptively discard a model from simulation studies include extreme changes in RoG, insufficient binding energy between drug and protein, etc. This ability to dynamically preempt simulations of poor drug-candidates allows users to channel the resulting savings into examination of additional drug-candidates or to enhance infrastructure for the computations. Task-manager updates Resource-manager on inputs, schedules, and resources to be deployed. Resource-manager then acquires, manages and frees appropriate software, services, or (virtual) hardware from Cloud. The necessary software and hardware resources for simulation, analysis, and storage together form corresponding pools. As computations progress, Task-manager updates user on the latest status of simulations/analysis (scheduled, underway, completed, or terminated), available results, deployed resources, and the projected time for completion of the work. If required, the user may opt to intervene and update inputs to alter or improvise the study undertaken, or expand its scope and allocated funding.

This above framework can be implemented by the Cloud provider (First-party API) or could be developed by third parties (Third-party API). Competent clients may develop solutions through their own efforts and deploy them either on public or private Clouds. In principle, it is of practical utility, and with relative ease, the solution can be deployed by providers of Cloud infrastructure to identify even those drug-receptor candidates that may work well in aqueous conditions but fail in physiological environment. This would definitely help in containing both time and phenomenal costs involved in the development of new drugs.

4 Conclusions

In this paper, using Apache Spark, we demonstrated that big data approach provides substantial speedups in MD studies. This is especially needed in the more cell-like environment involving a plethora of biomolecules. A responsive, scalable, and self-tuning framework that pairs Spark with the flexibility of Cloud infrastructure for the MD studies is presented. This framework enables users to optimally utilize the available resources. Insights obtained using the proposed approach were discussed. Complex MD studies, accompanied by such newer and more versatile tools, would bridge the gap between theoretical modelling and experimental observations. Such studies may soon become an imminent requirement in the capital intensive and time-constrained pharmaceutical industry. The relevance of such advances, especially in these pandemic times, cannot be overstated.

Acknowledgment

We thank the Central Computing Facility of IIIT Allahabad for all the computations carried out. We thank Dr. Muneendra Ojha (IIIT-NR, Naya Raipur) and Dr. Jagpreet Singh (IIIT-A) for proof-reading the manuscript. BSS thanks Mr. Asari Ramprasad (IIIT-A) and Ms. Shivani Maheshwari (Postman Inc.) for useful discussions on SciSpark and Cloud infrastructure. Dr. Arumugam Madhumalar and Mr. Zahoor Ahmad Bhat (JMI, New Delhi) are also gratefully acknowledged for the help provided regarding M^{pro} simulations. SparkTraj was developed by one of the authors (BSS), and analysis was done jointly by the two.

Bibliography

[1] Borthakur, D.: The hadoop distributed file system: Architecture and design. Hadoop Project Website **11**(2007), 21 (2007)

[2] Borthakur, D., et al.: Hdfs architecture guide. Hadoop Apache Project **53**(1-13), 2 (2008)

[3] Bux, K., Moin, S.T.: Solvation of cholesterol in different solvents: a molecular dynamics simulation study. Physical Chemistry Chemical Physics (2020)

[4] Chitara, D., Sanjeev, B.S.: Buried water molecules in long scale molecular dynamics simulations. insights from 1.5 microsecond simulations on human argonaute2-rna complex. Manuscript Under Preparation

[5] De Vivo, M., Masetti, M., Bottegoni, G., Cavalli, A.: Role of molecular dynamics and related methods in drug discovery. Journal of medicinal chemistry **59**(9), 4035–4061 (2016)

[6] EC2, A.: Amazon Web Services. Available in: http://aws.amazon.com/ (2006)

[7] Ellis, R.J.: Macromolecular crowding: obvious but underappreciated. Trends in biochemical sciences **26**, 597–604 (2001)

[8] Karau, H., Warren, R.: High performance Spark: best practices for scaling and optimizing Apache Spark. " O'Reilly Media, Inc." (2017)

[9] Li, J., Liao, W.k., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., Zingale, M.: Parallel netcdf: A high-performance scientific i/o interface. In: SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing. pp. 39–39. IEEE (2003)

[10] Microsoft: Microsoft Azure. Available in: https://azure.microsoft.com/ (2014)

[11] Minton, A.P.: Molecular crowding: analysis of effects of high concentrations of inert cosolutes on biochemical equilibria and rates in terms of volume exclusion. In: Methods in enzymology, vol. 295, pp. 127–149. Elsevier (1998)

[12] Odersky, M., Spoon, L., Venners, B.: Programming in scala. Artima Inc (2008)

[13] Palamuttam, R., Mogrovejo, R.M., Mattmann, C., Wilson, B., Whitehall, K., Verma, R., McGibbney, L., Ramirez, P.: Scispark: Applying in-memory distributed computing to weather event detection and tracking. In: 2015 IEEE International Conference on Big Data (Big Data). pp. 2020–2026. IEEE (2015)

[14] Pikkemaat, M.G., Linssen, A.B., Berendsen, H.J., Janssen, D.B.: Molecular dynamics simulations as a tool for improving protein stability. Protein engineering **15**(3), 185–192 (2002)

[15] Roe, D.R., Cheatham III, T.E.: Ptraj and cpptraj: software for processing and analysis of molecular dynamics trajectory data. Journal of chemical theory and computation **9**(7), 3084–3095 (2013)

[16] Salo-Ahen, O.M., Alanko, I., Bhadane, R., Bonvin, A.M., Honorato, R.V., Hossain, S., Juffer, A.H., Kabedev, A., Lahtela-Kakkonen, M., Larsen, A.S.,

et al.: Molecular dynamics simulations in drug discovery and pharmaceutical development. Processes **9**(1), 71 (2021)

[17] Salomon-Ferrer, R., Case, D.A., Walker, R.C.: An overview of the amber biomolecular simulation package. Wiley Interdisciplinary Reviews: Computational Molecular Science **3**(2), 198–210 (2013)

[18] Sanjeev, B.S., Chitara, D., Arumugam, M.: Physiological models to study the effect of molecular crowding on multi-drug bound proteins: Insights from sars-cov-2 main protease. Journal of Biomolecular Structure and Dynamics. (Accepted)

[19] Sanjeev, B.: Ankush. Indian Institute of Science (2004)

[20] Shaw, D.E., Maragakis, P., Lindorff-Larsen, K., Piana, S., Dror, R.O., Eastwood, M.P., Bank, J.A., Jumper, J.M., Salmon, J.K., Shan, Y., et al.: Atomic-level characterization of the structural dynamics of proteins. Science **330**(6002), 341–346 (2010)

[21] Wouters, O.J., McKee, M., Luyten, J.: Estimated research and development investment needed to bring a new medicine to market, 2009-2018. Jama **323**(9), 844–853 (2020)

[22] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12). pp. 15–28 (2012)

[23] Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I., et al.: Spark: Cluster computing with working sets. HotCloud **10**(10-10), 95 (2010)

[24] Zimmerman, S.B., Minton, A.P.: Macromolecular crowding: biochemical, biophysical, and physiological consequences. Annual review of biophysics and biomolecular structure **22**, 27–65 (1993)