

## ARTICLES

# Distributed Chemical Computing Using ChemStar: An Open Source Java Remote Method Invocation Architecture Applied to Large Scale Molecular Data from PubChem

M. Karthikeyan\* and S. Krishnan

Digital Information Resource Center, Information Division, National Chemical Laboratory,  
Pune 411008, India

Anil Kumar Pandey

Munich Information Center for Protein Sequences (MIPS), Institute for Bioinformatics, Helmholtz Zentrum  
München German Research Center for Environmental Health (GmbH), Ingolstaedter Landstrasse 1, 85764  
Neuherberg (bei Munich), Germany

Andreas Bender

Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, Leiden University,  
Gorlaeus Laboratories, Einsteinweg 55, 2333 CC Leiden, The Netherlands

Alexander Tropsha\*

Laboratory for Molecular Modeling, School of Pharmacy and Department of Pharmacology,  
School of Medicine, University of North Carolina, Chapel Hill, North Carolina 27599

Received September 9, 2007

We present the application of a Java remote method invocation (RMI) based open source architecture to distributed chemical computing. This architecture was previously employed for distributed data harvesting of chemical information from the Internet via the Google application programming interface (API; ChemXtreme). Due to its open source character and its flexibility, the underlying server/client framework can be quickly adopted to virtually every computational task that can be parallelized. Here, we present the server/client communication framework as well as an application to distributed computing of chemical properties on a large scale (currently the size of PubChem; about 18 million compounds), using both the Marvin toolkit as well as the open source JOELib package. As an application, for this set of compounds, the agreement of log P and TPSA between the packages was compared. Outliers were found to be mostly non-druglike compounds and differences could usually be explained by differences in the underlying algorithms. ChemStar is the first open source distributed chemical computing environment built on Java RMI, which is also easily adaptable to user demands due to its “plug-in architecture”. The complete source codes as well as calculated properties along with links to PubChem resources are available on the Internet via a graphical user interface at <http://moltable.ncl.res.in/chemstar/>.

## INTRODUCTION

The distribution of computational tasks across multiple processors has been proposed and realized for decades. In recent years the “Screensaver Lifesaver” project,<sup>1,2</sup> lead by Oxford University academic Graham W. Richards, has achieved wide publicity, even shared by mainstream media.<sup>3</sup> The project, supported by United Devices and Intel and also by millions of users worldwide, is one of the world’s largest distributed computational project and was at its peak using the screensaver time of over 3.5 million computers worldwide

(450 000 years of central processing unit power) to evaluate 3.5 billion different molecules for their cancer-fighting potential. Following the Anthrax attacks on the Capitol in early 2002, this tremendous source of computing power has also been directed to fight the Anthrax scare. The aim was to identify compounds which inhibit binding of the Anthrax lethal factor to the so-called protective antigen heptamer in order to render the Anthrax toxin inactive.<sup>4</sup> More recently, in 2003, large scale docking against Smallpox protein targets was performed. In each case, suitable compounds have been (and still are) followed up for in vitro experimental testing and will hopefully lead to novel pharmaceuticals on the market. According to Richards, this was the first time in scientific computing that computing power was not a limiting

\* Corresponding authors. E-mail: m.karthikeyan@ncl.res.in (M.K.); alex\_tropsha@unc.edu (A.T.). Phone: +91 (20) 2590 2483 (M.K.); 919-966-2955(A.T.). Fax: + 91 (20) 2590 2850 (M.K.); 919-966-0204 (A.T.).

**Table 1.** Programming Models Suitable for GRID Computing (from ref 10)

model	examples	pros	cons
datagram/steam communication	UDP, TCP, multicast	low overhead	low level
shared memory, multithreading	POSIX threads, DSM	high level	scalability
data parallelism	HPF, HPC++	automatic parallelization	restricted applicability
message passing	MPI, PVM	high performance	low level
object-oriented	CORBA, DCOM, Java RMI	support for large-system design	performance
remote procedure call	DCE, ONC	simplicity	restricted applicability
high throughput	Condor, LSF, Nimrod	ease of use	restricted applicability
group-ordered	Isis, Totem	robustness	performance, scalability
agents	Aglets, Telescript	flexibility	performance, robustness

factor of scientific calculations and “We can do things that the major pharmaceutical companies cannot do”.<sup>5</sup>

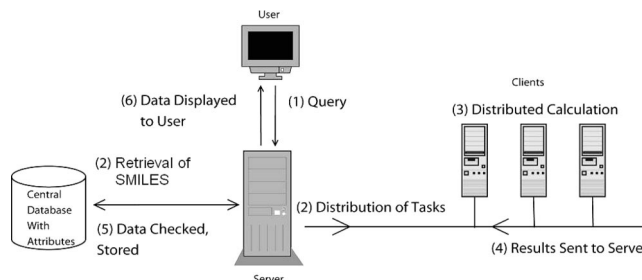
Launched in May 1999, the Internet distributed computing project SETI@home (Search for ExtraTerrestrial Intelligence at home) of the University of California at Berkeley started analyzing radio signals that might indicate a sign of intelligent life forms in other places in space. Signals collected by the world’s largest radio telescope, the Arecibo Radio Telescope in Puerto Rico, originally by far exceeded local processing capabilities. This fact created an interest in distributing data analysis tasks to a wider community. More than 40 gigabytes of data logged by the telescope have been distributed to clients worldwide with over two million people having had installed the SETI@home software at the peak of the project. The global network of 3 million clients provided processing power of about 14 teraFLOPS (14 trillion floating point operations per second), which corresponded to the world’s most powerful computer<sup>5</sup> in late 1999, exceeding the capabilities of the most powerful mainframe by a factor of nearly 10. SETI@home accumulated over 500 000 years of processing time within 18 months in 2000 and 2001, based simply on employing desktop computers’ spare CPU cycles. On the technical side, SETI@home is based on the Berkeley Open Infrastructure for Network Computing (BOINC) framework developed at UC Berkeley,<sup>6</sup> which is currently (as of January 2008) actively being used for dozens of distributed computing applications.<sup>7</sup>

There are several architectures for distributing a workload over multiple processors. One possible route is the utilization of a distributed operating system such as Amoeba<sup>8</sup> which enable the connection of heterogeneous hardware to a unit which seems like a standard single-processor machine to the user. The same operating system needs to be installed on every machine, on the other hand low-level interaction can be enabled, producing less computational overhead than higher-level data exchange. Also higher-level architectures such as the “GRID” can be employed,<sup>9,10</sup> making use of the Internet and the World Wide Web in order to share resources on an unprecedented scale and independent of location. Some of the more prominent approaches to distributing a workload over multiple processors are CONDOR,<sup>11</sup> PVM,<sup>12</sup> and CORBA, a more complete list is given in Table 1 (taken from ref 10). As can be seen in Table 1, every programming paradigm for distributed computing has different advantages and disadvantages which should define the route to follow in each given situation. The interesting question of which algorithms are actually amenable to distributed computing architectures was addressed recently by Fich and Ruppert.<sup>13</sup>

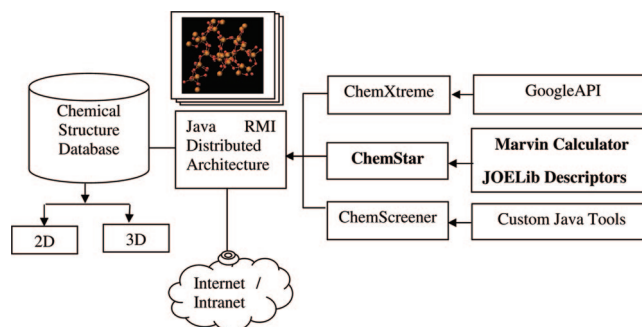
In this article, the first open source distributed computing and data analysis architecture based on Java remote method

invocation (RMI) is presented. A coarse-grained structure of the architecture is shown in Figure 1 that is discussed in detail in the Material and Methods section, where also a detailed discussion of the reasons for employing Java RMI in this work is provided. The main advantages of the present implementation of the system are its open source character and its flexibility to incorporate any kind of parallelizable distributed computing task. Accordingly, this Java-based distributed environment can be used creatively by the user to build his/her own scientific computing tools. It is thus not restricted to particular classes of algorithms (except that it must be possible to encode them in a parallel fashion). The “plug-in” character of the distributed environment presented here is illustrated in Figure 2.

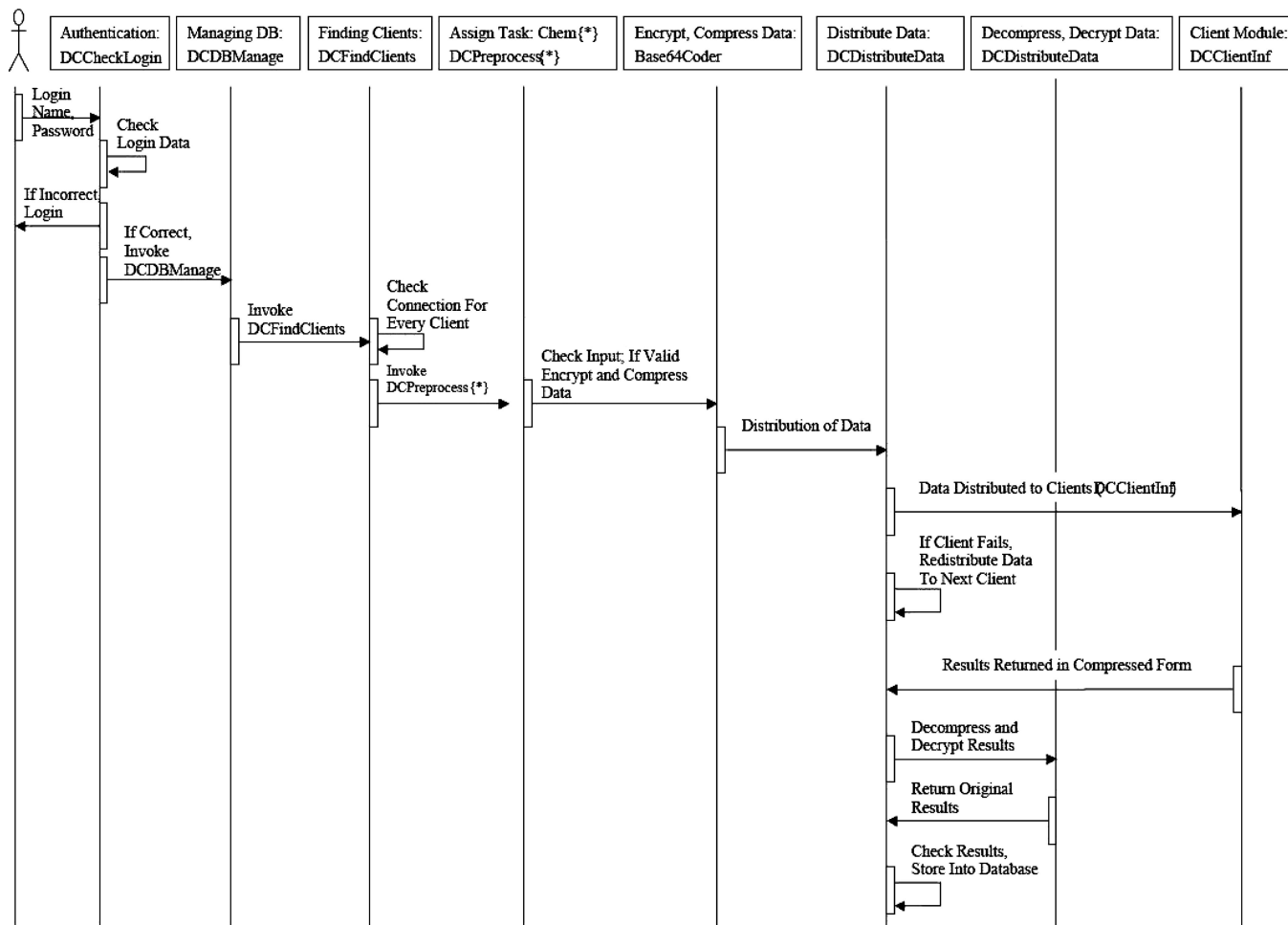
Earlier, we published the application of the Java RMI architecture to distributed data harvesting from the Internet in combination with the Google application programming interface (API), an application termed ChemXtreme.<sup>14</sup> While in both cases computational or searching tasks are carried out in a distributed fashion, the kind of data generated and their application is very different. In ChemXtreme the user



**Figure 1.** Communication between server and an, in principle, unlimited number of clients on a coarse-grained level. Each communication step is checked for possible errors, such as software, network, or hardware failures.



**Figure 2.** Core Java RMI-based “open distributed architecture” and its applications in cheminformatics. While already three different plug-ins for vastly different tasks (distributed searching, computing, and in silico synthesis) have been implemented, the architecture is adaptable to virtually every computational task that is amenable to being parallelized.



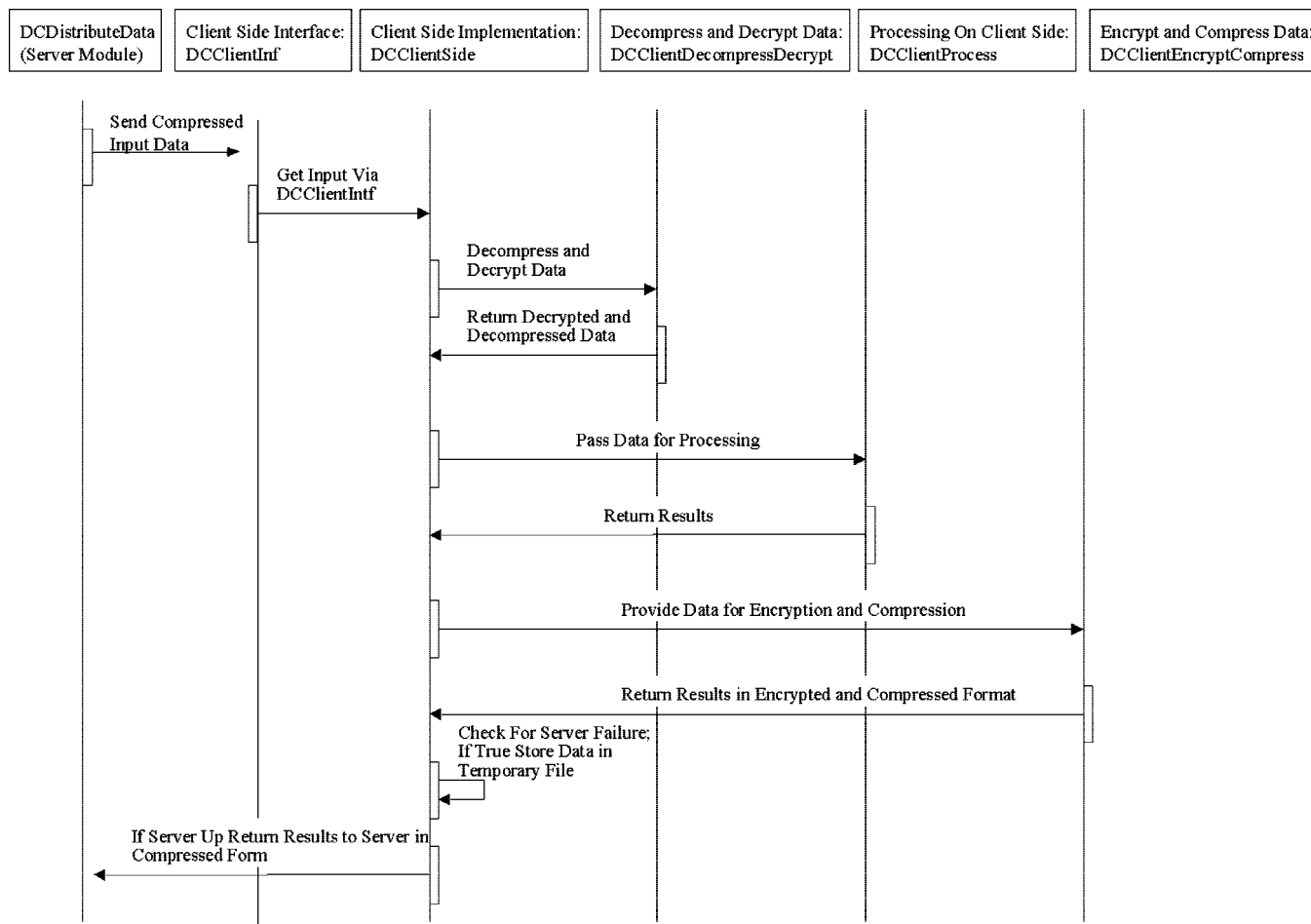
**Figure 3.** Computational steps employed for distributed searching, computing, and library generation on the server side. DCPreprocess is specific to the particular task employed; currently, the \* symbol denotes the modules ChemStar, ChemXtreme, and ChemScreener.

can specify the name(s) of a compound (alternatively, its CAS number) and all information publicly (on the Internet) available about this compound is extracted from the Internet using multiple clients and stored in a reusable format. Since Google<sup>15</sup> indexes a vast number of web pages (a number that currently stands at more than 8 billion), the Google API was used to identify relevant URLs located anywhere on the Internet.

In this work, we present the calculation of chemical properties on a large-scale data set in a distributed fashion. For a list of (in this case study) over 11 million structures extracted from PubChem molecular properties were calculated on the client side via Marvin plugins<sup>16</sup> distributed by ChemAxon<sup>17</sup> as well as the open source JOELib<sup>18</sup> whose source code is available freely from SourceForge. The properties calculated here, namely lipophilicity (expressed as log P) and total polar surface area (TPSA), are important determinants of oral absorption,<sup>19</sup> hence of crucial importance to drug discovery. A well-known form of calculating the polar surface area quick yet reliably was published by Peter Ertl and co-workers<sup>20</sup> which formed the basis for the prediction of oral absorption. As an application of the distributed computing environment, log P and TPSA values were compared between both packages, and also to MOE (Molecular Operating Environment) as an external reference. Descriptor (or more general program) validation is an integral part of software development. Similar analyses, comparing the

output of several programs which attempt to predict a physicochemical property, have recently been performed on the CDK project,<sup>21</sup> where the implementation of the XlogP descriptor was compared to its MOE equivalent.<sup>22</sup>

All the computed data along with source code, documentation, performance analysis, and instructions for using the source code can be downloaded from a dedicated web resource (Moltable<sup>23</sup>). Not only the underlying Java RMI client/server architecture is implemented as an open source project, also all harvested (ChemXtreme) or calculated (ChemStar) chemical, biological, and physical molecular data are stored in a local database for free access over the Internet.<sup>23</sup> The idea of this publicly accessible database is to make descriptors, which can often only be calculated employing proprietary—and expensive—software to the broad scientific society. While in the work presented here the calculation of descriptors is limited to those provided by the Marvin calculator plugins and JOELib descriptors, the underlying ORACLE database structure is easily extendable to a virtually unlimited number of descriptors, along with a virtually unlimited number of compounds for which these are calculated (both only limited by the hard disk space available). This development is paralleled by the advent of publicly accessible repositories of biological activity data, linked to molecular structures, such as PubChem,<sup>24</sup> ChemBank,<sup>25</sup> or ChEBI.<sup>26</sup> The combination of publicly available molecular descriptors with publicly available molecular



**Figure 4.** Computational steps employed for distributed searching on the client side.

**Table 2.** Time Taken in Seconds for 500 Structures on Single Clients and Distributed Sets of 100 Molecules Each on Five Clients and Sets of 50 Molecules Each on Ten Clients, Respectively

number of structures	number of clients	MWT (<300) time (s)	MWT (400–500) time (s)	MWT (600–700) time (s)	MWT (800–900) time (s)	MWT (>1000) time (s)
500	1	78–81	328–653	1328–1335	2730–3211	6257–12412
500	5	9–20	61–149	167–344	261–632	509–3563
500	10	5–14	28–106	62–178	123–350	181–1969

biological data is hoped to facilitate the dynamic generation of QSAR and QSPR models on a much larger scale than known before. Those data can be used to evaluate, e.g., molecular representations such as fingerprints on broad data sets, which have often been a limiting factor in benchmarking studies of this kind.

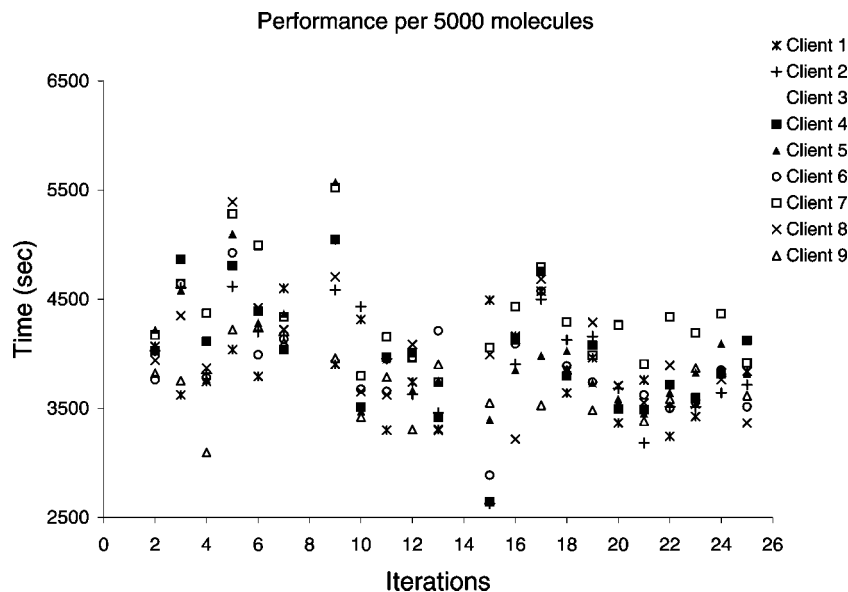
Very recently two systems related to the approach presented here have been published, namely a “Database-Centric Virtual Chemistry System”<sup>27</sup> and the “OpenMolGrid”.<sup>28,29</sup> The first of the two approaches<sup>27</sup> describes an Oracle-based data repository which forms the central cornerstone on which, in a mixed environment, external computing resources (both parallel and sequential ones) can be built. In the current publication, only pseudocode for the integration of OpenBabel is presented; also, while there Oracle functions are heavily employed for keeping track of jobs and results, the whole queuing system in our case consists of open source Java code which is shown to be interfaced with packages such as Marvin tools and the JOELib. The OpenMolGrid<sup>28,29</sup> on the other hand is an open source, GRID enabled system for distributed molecular

design and engineering applications which implements also packages relevant for performing QSAR/QSPR-type calculation with descriptor packages such as CODESSA. Recent publications of the environment have been published for toxicity predictions<sup>30</sup> and, in fact, some of the goals of the work presented here such as making molecular precomputed information publicly available via databases are also shared by this project. As opposed to OpenMolGrid, the work currently presented shows technological differences which also determine its applicability: While the OpenMolGrid is based on the UNICORE middleware<sup>31</sup> with the requirement for appropriate plugins, we are employing a Java RMI-based approach that enables a very flexible and user-definable integration of computing resources. Examples are given for the Marvin toolkit as well as JOELib.

## MATERIAL AND METHODS

**(a) Hardware and Software Used.** Operating systems employed on the server as well as the client side include Windows 95, Windows 98, Windows 2000, and Windows





**Figure 5.** Time needed for the property calculation of log P and TPSA values for 5000 molecules. Due to the different hardware of the clients, different times are needed for the calculation tasks; in addition, timing depends on the total load of the system which differs from time to time.

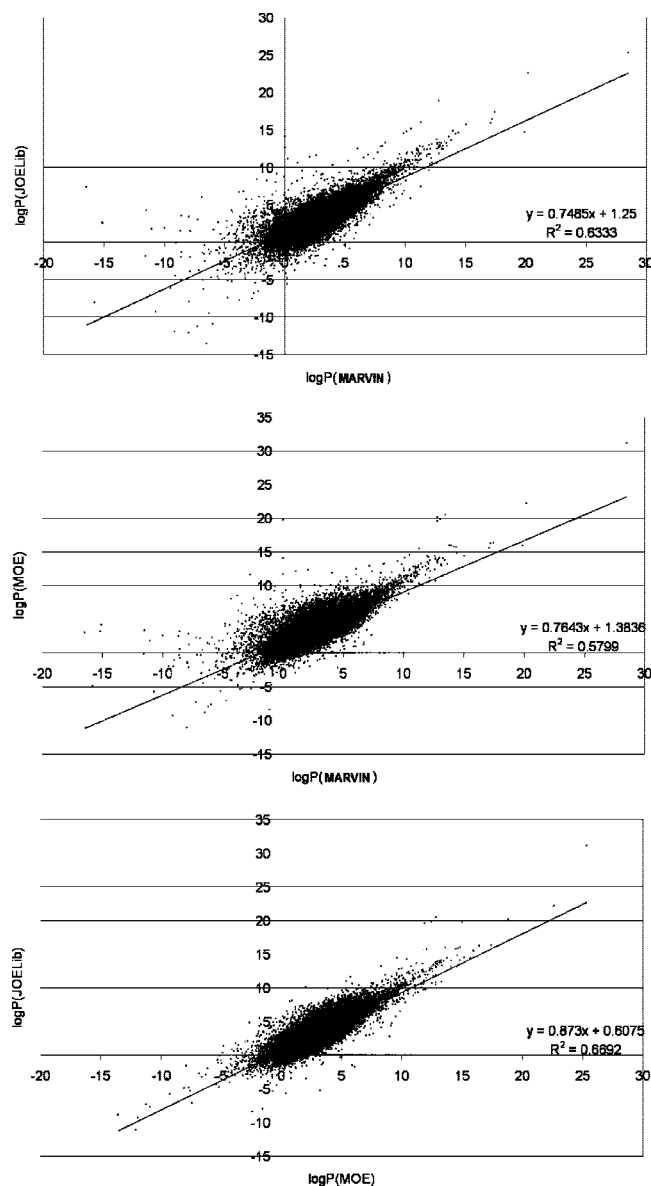
NT. In addition, on the server side, Java<sup>32</sup> and, of the databases, Oracle,<sup>33</sup> MySQL,<sup>34</sup> or MS Access are required. On the client side, only Java and the desired client side computing or searching modules have to be installed. In the network used here, the server was a Pentium IV (P-IV) machine running Windows XP with 2 GB RAM and a 80 GB of hard disk space. A wide range of clients was employed, and most of them are P-IV computers with 1 GB of RAM. This heterogeneous hardware structure did not pose any problem since the computational tasks assigned were tailored to the hardware ("thin" vs "thick" clients). This environment was tested in a network of 25 computers with similar configuration and operating systems which are also usually used for academic teaching/training.

**(b) Justification for Java and Java RMI.** Java has been chosen as a programming language for a variety of reasons. The main ones are listed in the following:

- Java is an object oriented language.
- It is secure as well as portable (platform independent).
- Java is capable of handling multithreaded and dynamic algorithms.
- It provides good support for implementing graphical user interfaces (GUIs).
- By means of Java, RMI client/server interaction can be implemented very efficiently.

Java RMI (remote method invocation) has been employed for implementing client/server communication. In addition to the advantages of RMI over the sockets and remote procedure calls (RPCs) mentioned in the introduction, RMI facilitates rapid prototyping and eliminates the need to employ an additional programming language for the interface definition language (IDL) component. All code that handles communication is generated directly from the class files. RMI uses a registry server for registering and locating objects, but in addition handles on objects can be accessed in order to locate subsequent objects, rendering access of remote systems straightforward. All marshaling of parameters and return values is taken care of by the system; in turn, the programmer can focus on application design without spend-

ing too much time about client/server communication. Calls to remote methods appear to the programmer identical to calls to local methods, the only difference being how to obtain an initial reference to an object (which is usually performed through a registry service), and the fact that objects passed as parameters or returned as return values are passed by value rather than passed by reference. This means that an object returned by a remote method call that is subsequently modified locally will not affect the original object. This abstraction makes application development using Java RMI generally straightforward. It also removes the burden of memory management from the programmer since the underlying system already provides distributed garbage collection. While Java for example supports sockets for this purpose, which are in general a flexible and sufficient method, sockets require client and server to employ applications-level protocols for message exchange. The design of protocols on this level was found to be rather cumbersome in the situation presented here. An alternative to sockets are remote procedure calls (RPCs) which provide the programmer with the option to call remote procedures in the same way local procedures are called, when in fact the arguments of the call are transmitted to the (remote) target of the call. The problem with RPC calls is that they do not work well in distributed object settings, where communication between program-level objects is needed, which reside in different address spaces. In order to provide appropriate semantics of object invocation distributed object systems require remote method invocation, RMI, in case of Java provided in a straightforward fashion via Java RMI. With Java RMI a local stub object manages the invocation of the actual remote object. Java RMI system takes advantage of the Java virtual machine and thus also employs the Java object model in most cases. In addition, Java RMI provides very flexible reference styles to remote objects such as persistent and nonpersistent references, among others. All of these characteristics render programming in RMI a comparably straightforward task and a sensible choice for developing pure Java client/server, peer-to-peer, or agent-based applications.



**Figure 6.** Correlation of lipophilicity values generated via JOELib, Marvin, and MOE. While overall good agreement between the values can be observed (see Table 4 for details), the outliers are mainly formed by rather non-druglike molecules.

**Table 3.** Correlation Matrix ( $r^2$ ) and RMSE values (in parentheses) for calculated log P and TPSA Values Based on over 500 000 Randomly Selected Molecules

log P $r^2$ (RMSE)			
	MOE	Marvin (JChem)	JOELib
MOE	1	0.5799 (1.737)	0.6692 (1.273)
Marvin (JChem)		1	0.6333 (1.612)
JOELIB			1
Total Polar Surface Area $r^2$ (RMSE)			
	MOE	Marvin (JChem)	JOELib
MOE	1	0.7982 (17.02)	0.7831 (18.50)
Marvin (JChem)		1	0.9885 (6.101)
JOELIB			1

While this is not the first application of Java RMI for chemical computing, to the best knowledge of the authors it is the first open source application in the area. Previous

examples where Java RMI was used for chemical computing include the Virtual Computational Chemistry Laboratory (VCCLAB)<sup>35,36</sup> with its central servers at the GSF Forschungszentrum für Umwelt und Gesundheit (National Research Centre for Environment and Health) in Munich, Germany, but which distributes its calculations throughout Europe, such as the universities of Portsmouth, Erlangen, Moscow, Kyiv, and Milano.<sup>37</sup> In this case, VCCLAB taps the external expertise (as well as computational tools) at every research group at the above universities for tasks such as data reduction (Portsmouth), 3D coordinate generation (Erlangen), and descriptor generation (Moscow, Kyiv, and Milano) and provides results to the user via a single entry point. The integration across software applications as well as physical locations by VCCLAB is truly remarkable, and this project certainly represents one of the most fruitful applications of Java RMI in the distributed chemical computing area.

It should be mentioned that the Java implementation by Sun<sup>32</sup> as well as its libraries used to be until recently (November 2006) a nonfree implementation of this programming language.<sup>38</sup> One has generally to take care not to rely on libraries with restricted licenses, in order to provide a truly “free” (free as in speech) software which is not restricted by license limitations of its environment—paths originally taken by for example Kaffe<sup>39</sup> or, on the library side, by the GNU Classpath<sup>40</sup> project. The current implementation of the distributed computing approach presented here uses the Java environment provided by Sun directly and due to its recent license conversion into open source (GPLv2) we believe that this solution should be sustainable for most applications.

**(c) Idea Behind Remote Method Invocation (RMI).** The communication between server and clients based on RMI can be represented as a four-layer model. Layer 1 represents the “application layer”, which comprises the actual implementation of the client and server application. This is also the layer from where high-level calls are made to access (invoke) remote objects. Layer 2 is the “proxy layer”, also called the “stub/skeleton layer”. This layer is the one the application residing on layer 1 interacts with directly. All calls to remote methods and the marshalling of parameters and return objects are done through these stub/skeleton proxies. Layer 3 is the “remote reference layer” that is responsible for dealing with the semantics of remote invocations. This layer is responsible for handling replicated objects and for performing implementation-specific tasks with remote objects. Layer 4 as the lowest-level layer is the “transport layer” which is responsible for setting up connections and handling the transport of data from one machine to another.

**(d) Client/Server Communication Employing Java RMI in ChemStar, ChemXtreme, and ChemScreener.** The basis flowchart for ChemStar, ChemXtreme, and ChemScreener distributed calculations is shown in Figure 1 in a general fashion and in more detail in Figures 3 and 4. First on the server side (Figure 3), after login and database invocation, the number of available clients on the local area network (LAN) is determined which also need to run the required client software. On the server side, the contents to be distributed to the clients is defined, which for example includes a list of structures for which properties are to be computed in case of ChemStar. This data snippet is encrypted

**Table 4.** Most Extreme Outliers for log P/TPSA Calculations between Marvin/JChem, JOELib, and MOE

property	algorithms	PubChem CID	predictions	difference
log P	JOELib vs Marvin	3436508	Marvin = -45.29; JOELib = -0.81	44.48
		3017566	Marvin = -39.49; JOELib = 2.98	42.47
		3014312	Marvin = -39.31; JOELib = 3.10	42.41
		3017565	Marvin = -39.49; JOELib = 2.64	42.13
	MOE vs Marvin	3023616	Marvin = -39.31; JOELib = 2.75	42.06
		3436508	MOE = -0.05; Marvin = -45.29	45.24
		3017566	MOE = -0.70; Marvin = -39.49	38.78
		3014312	MOE = -0.80; Marvin = -39.31	38.52
	JOELib vs MOE	3023616	MOE = -0.85; Marvin = -39.31	38.47
		3014311	MOE = 6.80; Marvin = -39.31	38.47
		5238565	JOELib = -36.62; MOE = 8.47	45.09
		5238564	JOELib = -33.31; MOE = 8.81	42.12
		6479690	JOELib = 6.17; MOE = -20.95	27.12
		6335027	JOELib = -19.27; MOE = 6.13	25.40
		6335025	JOELib = -19.27; MOE = 6.13	25.40
TPSA	JOELib vs Marvin	444237	JOELib = 834.73; Marvin = 632.33	202.40
		3354082	JOELib = 834.73; Marvin = 632.33	202.40
		5018468	JOELib = 754.08; Marvin = 576.98	177.10
		6367229	JOELib = 416.67; Marvin = 242.37	174.30
	MOE vs Marvin	419780	JOELib = 687.15; Marvin = 522.48	164.67
		6338588	MOE = 3230.48; Marvin = 3532.16	301.68
		5497141	MOE = 3255.95; Marvin = 3557.63	301.68
		6914530	MOE = 2871.76; Marvin = 3140.92	268.16
	JOELib vs MOE	6338587	MOE = 2871.76; Marvin = 3140.92	168.16
		5497139	MOE = 2895.40; Marvin = 3163.56	168.16
		6338588	JOELib = 3659.51; MOE = 3230.48	429.03
		5497141	JOELib = 3659.51; MOE = 3255.95	403.56
		6914530	JOELib = 3254.12; MOE = 2872.76	381.36
		6338587	JOELib = 3254.12; MOE = 2872.76	381.36
		5497139	JOELib = 3254.12; MOE = 2895.4	358.72

(base 64) and LZW/ZIP-compressed for safe and reliable information distribution over the network. Threads are then generated on the server for each client and the encrypted and compressed data generated in the previous step is distributed to the clients. On the client side, the received data is decompressed, decrypted and processed. Processing in this context means that first the computations defined on the server-side are executed on the client, depending on the particular computational task.

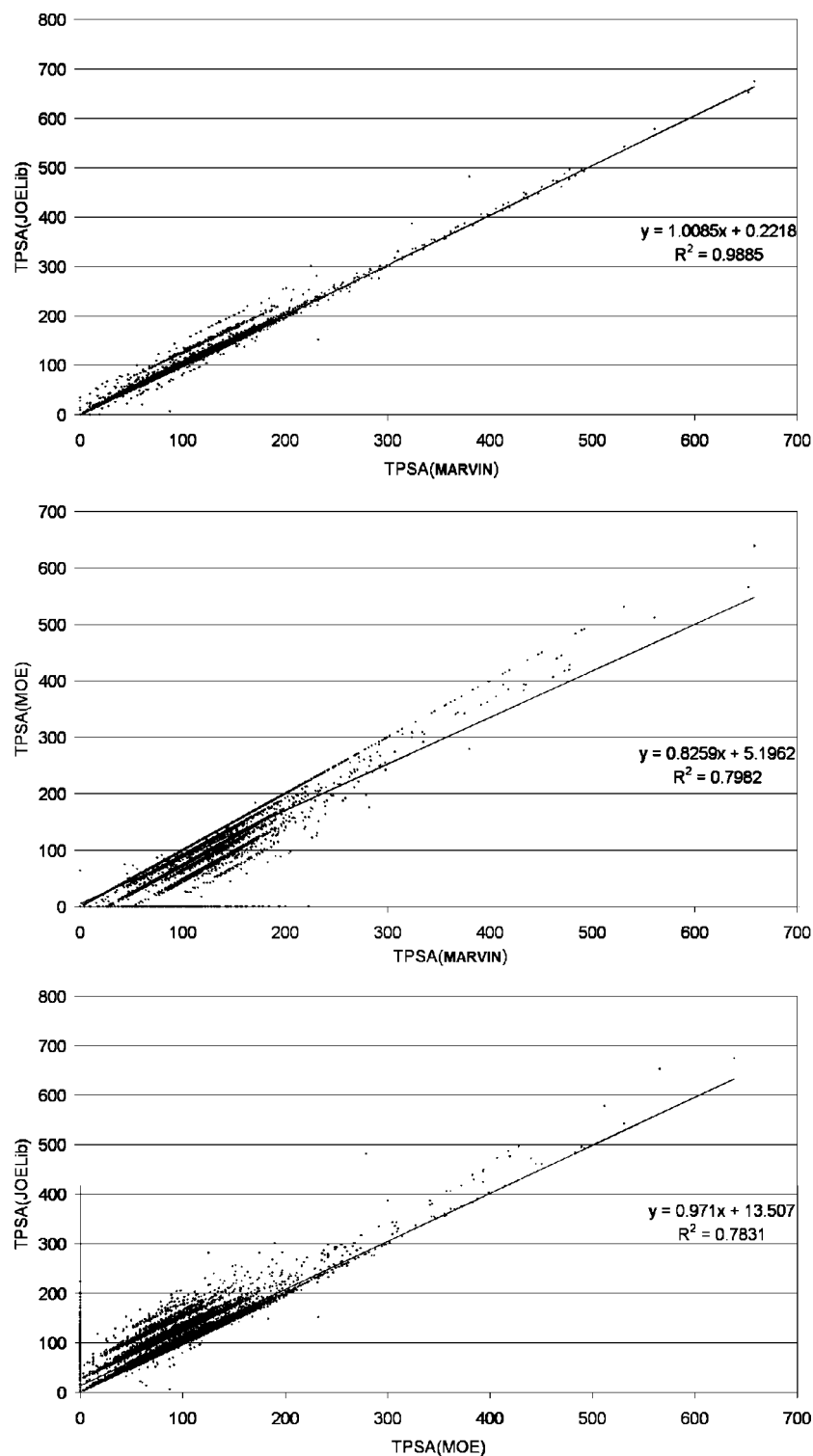
Computed results are encrypted on the client side, compressed and then sent back to the server within a certain time limit as defined in the configuration file at the server side. The server finally decompresses and decrypts the results received from the clients, checks the results for validity, and stores them in a permanent database. In order to facilitate performance analysis, timestamps are employed at every step where data are sent or received. These features are optionally enabled in the core DCE (Distributed Computing Environment) components. The configuration file with input SQL query is in the plain text format can be easily edited for submitting the job to the server without modify the program.

The choice of calculation tools on the client side is very flexible and currently in case of ChemStar includes Marvin<sup>16</sup> and JOELib tools.<sup>16</sup> Currently possible properties which can be calculated include predictions of  $pK_a$ , log P and log D, polar surface area, charge distribution, polarizability, topology analysis, H-bond acceptor and donor properties, the prediction of protonation microspecies (from  $pK_a$  prediction), Hückel analysis, refractivity, and elemental analysis.

**(e) Error Detection and Fault Tolerance.** The system also captures a wide variety of possible errors and, depending on the particular problem, attempts alternate routes. In case of server failure while calculations are taking place on the

clients the clients will continue performing calculations and will store the calculated results locally. In user-definable intervals, server availability is checked. As soon as the server recovers and a communication server process is started the clients transfer the results stored locally back to the server. In case of client failure, which includes total client failure as well as not transmitting results back to the server within a user-definable time period, the input data of that client is redistributed to the next available client.

**(f) Java Class Structure Employed.** The Java class structure employed is shown in Figure 3 for the server side and in Figure 4 for the client side. Server-sided (Figure 3) login to the system is provided via DCCheckLogin, which requires the input of a username as well as a password and allows login as admin, normal user or guest. Control is handed over to DCServer which gives the user the option to perform the desired computing task (ChemStar, ChemX-treme, ChemScreen, or any user-defined task). At the same time DCServer invokes the database handler, DCDBManage, which also requires database specific data such as the name of the database and the port employed for establishing communication. DCServer then establishes a list of clients which are able to participate in the computing or information harvesting task, respectively, via DCFindClients. Depending on the configuration, the appropriate DCPreProcessInput task is called, each of which requires different input data. In this case the molecules in SMILES format from local database at the server side standard input format to perform Marvin and JOELib calculations using DCPreprocessCompute (ChemStar). In each case, Base64Coder will be called next which takes the input data, compresses and encrypts it, and forward the encrypted and compressed output to DCDistributeData.



**Figure 7.** Correlation of TPSA values generated via JOELib, Marvin, and MOE. While overall good agreement between the values can be observed (see Table 4 for details), the outliers are due to different definitions of “polar” atoms, such as sulfur functional groups.

DCDistributeData distributes information to the clients and, after the computing task is finished, information is collected from the clients by Base64Coder.

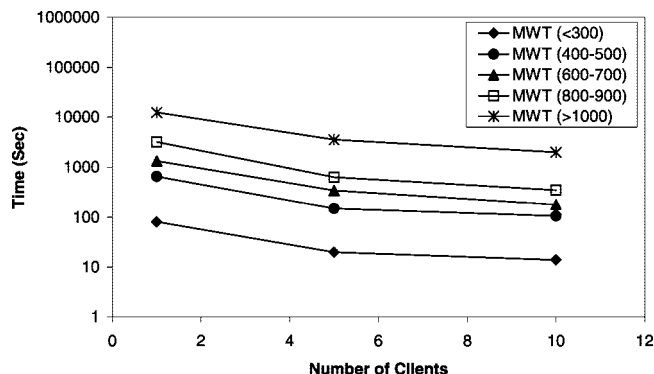
On the client-side (Figure 4), the structure of classes is slightly simpler, since in principle only information needs to be decompressed and decrypted, next it is processed, and finally results are encrypted and compressed again and transmitted back to the server. The first of these steps is handled by Base64Coder, which receives an encrypted and

compressed string and gives back the decrypted as well as decompressed form of it. Next, ProcessData is invoked, which performs the actual computation task.

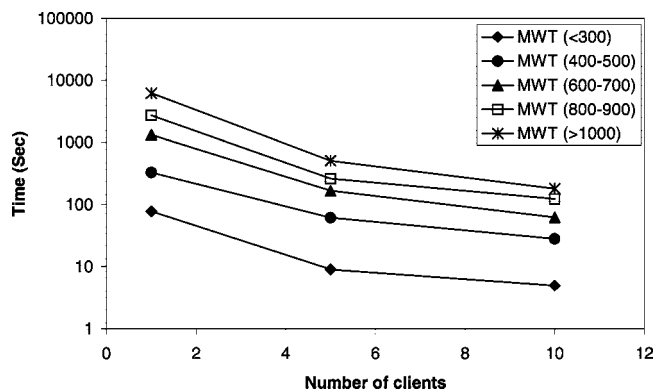
The snippet of computing code at the client side is shown below:

```
DCClientComputeJOELib(String smiles, String rid)
JOEMol mol = a.ReadSMILES (smiles, b.getIOType
("SMILES"), c.getIOType("SDF"),pid);
ComputeDescriptors comdescs = new ComputeDescriptors();
```





**Figure 8.** Maximum time taken for 500 molecules with different range of molecular weight with increasing number of clients. All properties for the molecules (TPSA and log P) were calculated simultaneously for each structure.



**Figure 9.** Minimum time taken for 500 molecules with different range of molecular weight with increasing number of clients. All properties for the molecules (TPSA and log P) were calculated simultaneously for each structure.

```
String computedstrJOELib = comdescs.logP + . . . ; com-
descs.tpsa + """"; DCClientComputeJChem(String SMI-
LES,String pid) { . . .
```

```
ReadLicenceFile rlf = new ReadLicenceFile(); rlf.jchem-
ReadLicence();
```

```
logPPlugin plugin = new log PPlugin(); plugin.validate-
(rlf.logPPlugin); plugin.setMolecule(target); plugin.run();.. }
Output_data = computed_strJChem+"""" + computed-
strJOELib;
```

Finally, results are encrypted and compressed again via Base64Coder and transferred back to the server.

**(g) Limitations and Further Testing.** While the system is very flexible and achieved very good performance in our setting, there are also some limitations of it. It is necessary to scan the available network (Internet or Intranet) for the registered and active clients which will perform computation. The list of registered and active clients in plain text format is read by the server and each client is tested for data transport features at particular port designated for RMI-based communication. Current version of the source code does not support dynamic recognition of new active clients after start distributing the data to clients. However this can be achieved by suitably modifying the source code at the server side. If the client fails after receiving the data from server does not respond within a set time limit results will be lost for the small set of input data and no efforts has been made to recover data from client side. Then the same task has to be given to another client by monitoring the overall process of computing at the server side. Since each calculation is

independent for individual molecules, rerouting the failed data set to any other active client at different time does not affect the overall computational activities. So far the project has undergone thorough testing on the Windows operating system only. Some changes to the program are required if the server as well as the clients are running a Unix/Linux operating system.

Computed data for 12 million molecules are available from <http://moltable.ncl.res.in/chemstar/>. The updated source code and executable programs of the Java RMI-based architecture can be obtained from one of the authors (M.K.) directly.

## RESULTS AND DISCUSSION

The work presented here is a proof-of-concept study for performing distributed chemical computing and searching tasks via Java RMI. The system was also stable in day-to-day use across the different computational tasks performed.

ChemStar was found to represent a stable as well as flexible solution for distributing computing tasks over multiple machines. In order to perform molecular properties calculation using Marvin calculator plugins special license has been obtained from ChemAxon and JOELib program being distributed freely over the Internet it was integrated with ChemStar with prior intimation to the developer. In the case study presented here, molecular properties were calculated for over 11 million PubChem structures using multiple clients without problems and minimal overhead.

Scaling of the method is presented in Table 2 for 500 structures on single clients and distributed sets of 100 molecules each on five clients and sets of 50 molecules each on ten clients respectively. All these calculations were repeated at least three times for molecules with ranges of molecular weight for performance analysis. It can be seen that favorable scaling can be achieved, with the time needed for each individual calculation being roughly inversely proportional to the number of clients used. For example, when going from 1 to 5 to 10 clients for molecules with MW < 300, the minimum time needed to calculate 500 structures decreases from 78 to 9 to 5 s, respectively, while the maximum time needed for this calculation decreases from 81 to 20 s to 14 s. In both cases roughly inversely proportional scaling can be observed, which is also true for compounds with higher molecular weight (see Table 2). The overhead present seems thus to be tolerable in the application presented. It is observed that the time taken for computing increases with increasing molecular weight and decreases with number of clients due to distribution of input data as shown in Table 2, which is unsurprising. Since the hardware employed here comprises computers anywhere from Pentium II to Pentium IV, the performance of each individual machine varies considerably (see Figure 5). While the time required for the calculation of properties varies by a factor of about 4 between different machines, due to the nature of the task scheduler and the complexity of the molecules as seen from the data, the time taken increases with increase in molecular weight and this does not represent a problem for the distributed computing environment.

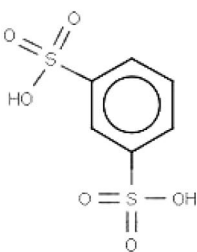
While the ChemStar environment presented here is generally applicable to the distributed calculation of chemical properties, a case study comparing descriptor calculations using three different cheminformatics packages shall be

Address [http://moltable.nci.res.in/chemstar/view\\_pcd\\_data\\_515821.jsp?start=1](http://moltable.nci.res.in/chemstar/view_pcd_data_515821.jsp?start=1)

ChemStar(Computed Data Browser)

NCL ID : 1

PubChem ID : [7388 \[link out\]](#)



[Browse \[1 - 515821\]](#)

First << Prev < Next > Last >>

Between 1 - 515821  [Go](#)

JCHEM		MOE	
Molecular Formula	C6H6O6S2	LogP	-0.263
Exact Mass	237.9606	TPSA	108.74
Atom Count	14	<b>JOELIB</b>	
Heavy atom Count	14.0	LogP	1.656
H present	false	HBA	6
LogP	1.0426	HBD	0
TPSA	125.5	n_BadAtoms	0
Mass	238.2402	n_CF3	0
Oxygen Atom Count	6	n_N	0
Nonisotope Count	6	n_NO2	0
Oxygen Isotope Count	0	n_O	6
Isotope Formula	C6H6O6S2	n_S	2
Red 255 Green 0 Blue 0		n_SO2	0
ChargeText		n_SSSR	1
	[-0.148 (0.239), -0.043, -0.169, -0.169, 0.055, 0.058 (0.064), -0.004 (0.062), 0.058 (0.064), 0.055, 0.101]	n_X	0
Composition	C (30.25%), H (2.54%), O (40.29%), S (26.92%)	Rotatables Bonds	2
		MOL Weight	236.226
		TPSA	131.16

**Figure 10.** Web interface to browse the MOE-, Marvin-, and JOELib-based computed data for over 11 million molecules along with link out option to PubChem resources.

presented here. Namely, a comparison of log P and total polar surface area (TPSA) calculations is performed, based on the packages MOE,<sup>41</sup> the Marvin calculator,<sup>16</sup> and JOELib.<sup>18</sup>

For the log P calculations, results are presented in Figure 6 and Table 3 for a collection of 11 million molecules from PubChem. Lipophilicity (log P) values calculated by MOE and JOELib show a correlation coefficient of  $r^2 = 0.71$  (Table 3) at an RMSE of 1.273 units, while the correlation between values calculated by JOELib and those calculated by Marvin is lower, at  $r^2 = 0.58$  (RMSE of 1.612). Comparing lipophilicities of MOE and Marvin log P values, a correlation coefficient of  $r^2 = 0.57$  is observed (RMSE 1.737 units). Overall considerable differences can be observed between the different approaches which can partly be explained by the different algorithms and training data sets used. While the algorithm employed by MOE is unpublished (but available in SVL code with the MOE license), Marvin and JOELib implement two different algorithms for the calculation of log P, namely those by Viswanadhan et al.<sup>42</sup> (in Marvin) and by Wildman and Crippen<sup>43</sup> (in JOELib). The implementation in Marvin can more precisely be referred to as a modification of the algorithm by Viswanadhan et al. with the following changes:<sup>44</sup>

The algorithm described in the paper was modified at several points. Many atomic types were redefined to accommodate electron delocalization. Contributions of ionic forms were added. The logP value of zwitterions are calculated from the logD value at the isoelectric point. The effect of hydrogen bonds on logP is considered if there is a chance to form a six membered ring between suitable donor and acceptor atoms. New atom types were introduced especially for sulfur, carbon, nitrogen, and metal atoms.<sup>44</sup>

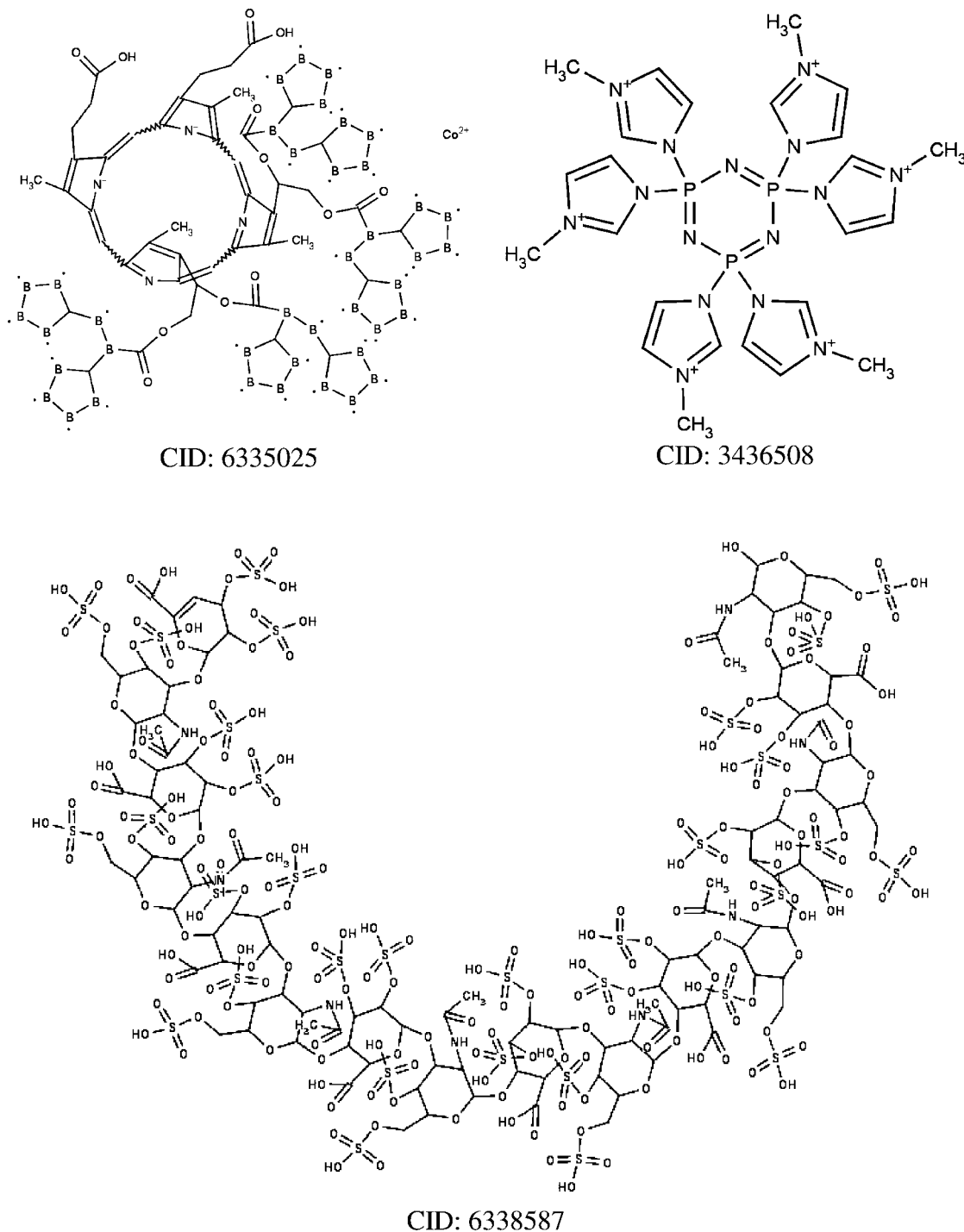
A complete description of the algorithm is provided on the ChemAxon Web site.<sup>45</sup>

For the TPSA calculations, results are presented in Figure 7 as well as Table 3 for the same collection of molecules. Generally, agreement between surface areas calculated by the different programs is much higher than agreement between lipophilicity values. TPSAs obtained from Marvin and JOELib show near-perfect correlation of  $r^2 = 0.98$

(RMSE 6.101 Å<sup>2</sup>), while the correlation between values calculated by MOE and those calculated by Marvin is lower, at  $r^2 = 0.91$  (RMSE 17.02 Å<sup>2</sup>). Comparing results of MOE and JOELib approaches a correlation coefficient of  $r^2 = 0.81$  is observed (RMSE 18.50 Å<sup>2</sup>). Interestingly the correlations of MOE TPSA values with both other approaches (JOELib and Marvin) show, apart from the largest part of the compounds being located on the diagonal (ideal) line, several shifted lines. Since this is not observed for the correlation of Marvin and JOELib results it might be the case that the MOE approach calculated significantly different other surface areas, compared to the other approaches. These results are particularly interesting since all of the software packages, MOE, Marvin as well as JOELib, are implementing Peter Ertl's TPSA algorithm<sup>20</sup> which should in principle lead to very similar (or identical) results. It can be seen from the structures which represent the largest outliers that certain fragments or structural features are consistently associated with them. These often include sulfur moieties, which were included in the original definition of the TPSA due to the improved correlation with human jejunum permeability.<sup>20,46</sup>

Another important factor that contributes to the assignment of predicted molecular properties to particular structures depends on the different standardization algorithms used. For example, different algorithms might perceive bond orders and implicit hydrogens differently, and the same might be true for aromatic ring systems for which no single, "universal" detection algorithm exists. Work on structure normalization has been presented before and only some more recent publications should be cited here.<sup>47,48</sup> It should be noted that the influence of different structure normalization algorithms on property prediction algorithms can easily be underestimated since this step is usually not part of the QSPR model itself but an "invisible" step preceding it. While keeping this factor in mind, the current work is focused on the whole process from structure to property prediction, using the package-inherent normalization algorithms in each case.

It can be seen that the wall clock time required to calculate the full set of molecular properties via Marvin and JOELib computing modules scales inversely with the number of



**Figure 11.** Sample list of some of the outlier molecules from the PubChem collection.

clients employed, indicating minimal computational overhead of the methods employed. Figures 8 and 9 show the maximum time taken and minimum time taken respectively in logarithmic scale by various clients with varying molecular weight. Here, times are measured as the interval between sending structure information from the server to the clients, until results are completely transferred back to the server. This benchmark can be seen as biased to some extent since different programming languages and architectures are compared. Still, by including every overhead from method invocation to network traffic, we get the most relevant benchmarking from the user perspective: The time until results are available on the user side. The variation in

computational time per molecule and the outliers in the computed data can be endorsed by the diversity and complexity of molecules in PubChem collection as shown in Figure 11.

The full set of molecular properties calculated was made publicly made available from a dedicated web resource.<sup>23</sup> In Figure 10, the graphical user interface is shown, where calculated properties for a sample compounds are displayed, distinguished by the different algorithms employed for their generation. Every molecule processed from the Pubchem compound collections was hyperlinked with the original Pubchem resource via the PubChem Compound Identifier (CID). This activity and the large volume of data would

complement the initiatives of Pubchem there by the users can explore the molecular data in both biological activity space originally provided by PubChem and also in the computed molecular descriptors space generated by distributed computing architecture integrated with Marvin Calculator Plugins and JOELib.

## CONCLUSIONS

This work presents the application of a Java RMI-based open source architecture to distributed chemical computing. As a case study, we calculated molecular properties for a set of 11 million molecular structures via MOE, Marvin, and JOELib and obtained a set of descriptors that were made publicly available.<sup>23</sup> As a side-product, it was found that both log P and TPSA values calculated differ significantly between the different approaches.

Due to the open source nature of the underlying Java RMI architecture, the user is able to modify it to his requirements. Currently, this system has demonstrated its capabilities for computing data as well as harvesting information from the Internet only for chemical data, though it can easily be extended to other chemoinformatics related computational tasks such as in-silico-based virtual screening, fast docking, text mining and other scientific fields such as informatics, physics or biology related areas. Testing of the system has so far been performed on WLAN as well as traditional LANs, and with minor modifications it can also be used on the Internet for large-scale distributed computing. Currently, Oracle and MySQL databases are compatible with the system and database support shall be extended greatly in the future. Presently, server and client nature are defined statically for a given computing project. This can be enhanced to define server and client nature in a dynamical way, such that the clients turn to become servers to distribute data to further clients. Furthermore, enhanced load leveling for clients of different performance is envisaged.

This initiative would complement the PubChem initiatives of keeping the molecular data open access along with other necessary data. QSAR (and related, e.g., classification) studies were until known often limited by the data sets and descriptors available (which often needed proprietary or commercial software for their calculation). With the advent of public databases containing biological information of molecules, such as PubChem, ChemBank, and ChEBI, we hope that the combination of those two factors, public descriptors as well as public biological data, contribute to large-scale chemoinformatics and computational chemistry applications in the future.

## ACKNOWLEDGMENT

The authors thank the Director, NCL-Pune, for the support for this project. M.K. thanks the Department of Scientific and Industrial Research for sponsorship to MOLTABLE portal and Department of Science and Technology, New Delhi, India, for the award of the BOYSCAST fellowship and the Department of Biotechnology, New Delhi, India, for the award of the Long Term Overseas Associateship at the University of North Carolina at Chapel Hill. Ferenc Csizmadia (ChemAxon) and Jörg Wegner, author of JOELib, are thanked for their permission and cooperation during the implementation and testing of this project. M.K. thanks S. K.

Sidhu, S. G. Nandkumar, B. C. Sachin, I. Deepali, and S. Rashmi for technical support.

**Supporting Information Available:** Java RMI open source code of the distributed computing environment. This material is available free of charge via the Internet at <http://pubs.acs.org/>.

## REFERENCES AND NOTES

- (1) Richards, W. G. Virtual screening using grid computing: the screen-saver project. *Nat. Rev. Drug. Discov.* **2002**, *1*, 551–5.
- (2) Davies, E. K.; Glick, M.; Harrison, K. N.; Richards, W. G. Pattern recognition and massively distributed computing. *J. Comput. Chem.* **2002**, *23*, 1544–50.
- (3) BBC World News, Saturday, 18 September, 2004. <http://newswww.bbc.net.uk/2/hi/health/3597738.stm> (accessed January 31, 2008).
- (4) Glick, M.; Grant, G. H.; Richards, W. G. Pinpointing anthrax-toxin inhibitors. *Nat. Biotechnol.* **2002**, *20*, 118–9.
- (5) Top500. <http://www.top500.org/list/1999/11/> (accessed January 31, 2008).
- (6) Berkeley Open Infrastructure for Network Computing. <http://boinc.berkeley.edu/> (accessed January 31, 2008).
- (7) <http://boinc.berkeley.edu/projects.php> (accessed January 31, 2008).
- (8) Mullender, S. J.; van Rossum, G.; Tanenbaum, A. S.; van Renesse, R.; van Staveren, H. Amoeba—A Distributed Operating System for the 1990s. *IEEE Comput.* **1990**, *23*, 44–53.
- (9) Foster, I. The grid: A new infrastructure for 21st century science. *Phys. Today* **2002**, *55*, 42–47.
- (10) Foster, I.; Kesselman, C. Computational Grids. In *The Grid: Blueprint for a Future Computing Infrastructure*; Foster, I., Kesselman, C., Eds.; Morgan Kaufman Publishers: San Francisco, CA, 1998.
- (11) Thain, D.; Tannenbaum, T.; Livny, M. Distributed computing in practice: the Condor experience. *Concurrency Comput.: Pract. Experience* **2005**, *17*, 323–356.
- (12) Sunderam, V. S.; Geist, G. A.; Dongarra, J.; Manchek, R. The PVM Concurrent Computing System—Evolution, Experiences, and Trends. *Parallel Comput.* **1994**, *20*, 531–545.
- (13) Fich, F.; Ruppert, E. Hundreds of impossibility results for distributed computing. *Distrib. Comput.* **2003**, *16*, 121–163.
- (14) Karthikeyan, M.; Krishnan, S.; Pandey, A. K.; Bender, A. Harvesting Chemical Information from the Internet Using a Distributed Approach: ChemXtreme. *J. Chem. Inf. Model.* **2006**, *46*, 452–461.
- (15) Google, Inc., Mountain View, CA, USA. <http://www.google.com> (accessed January 31, 2008).
- (16) Csizmadia, F. JChem: Java applets and modules supporting chemical database handling from web browsers. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 323–4.
- (17) ChemAxon. <http://www.chemaxon.com> (accessed January 31, 2008).
- (18) JOELib—a java based computational chemistry package. <http://joelib.sourceforge.net/> (accessed January 31, 2008).
- (19) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **1997**, *23*, 3–25.
- (20) Ertl, P.; Rohde, B.; Selzer, P. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *J. Med. Chem.* **2000**, *43*, 3714–7.
- (21) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): an open-source Java library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493–500.
- (22) Fechner, U.; Guha, R. Descriptor Validation on the CDK Project. 1. XlogP. *CDK News* **2006**, *3*, 33–37.
- (23) <http://moltable.ncl.res.in/chemstar/> (accessed January 31, 2008).
- (24) PubChem Database. <http://pubchem.ncbi.nlm.nih.gov/> (accessed January 31, 2008).
- (25) ChemBank. <http://chembank.broad.harvard.edu/> (accessed January 31, 2008).
- (26) Chemical Entities of Biological Interest (ChEBI). <http://www.ebi.ac.uk/chebi/> (accessed January 31, 2008).
- (27) Lind, P.; Alm, M. A database-centric virtual chemistry system. *J. Chem. Inf. Model.* **2006**, *46*, 1034–9.
- (28) Sild, S.; Maran, U.; Romberg, M.; Schuller, B.; Benfenati, E. OpenMolGRID: Using automated workflows in GRID computing environment. In *Advances in Grid Computing—Egc 2005*; Springer-Verlag Berlin: Berlin, 2005; Vol. 3470, pp 464–473.



- (29) Sild, S.; Maran, U.; Lomaka, A.; Karelson, M. Open computing grid for molecular science and engineering. *J. Chem. Inf. Model.* **2006**, *46*, 9539.
- (30) Mazzatorta, P.; Smiesko, M.; Lo Piparo, E.; Benfenati, E. QSAR Model for Predicting Pesticide Aquatic Toxicity. *J. Chem. Inf. Model.* **2005**, *45*, 1767–1774.
- (31) Romborg, M. The UNICORE Grid Infrastructure. *Sci. Comp.* **2002**, *10*, 149–157.
- (32) Java. <http://java.sun.com/> (accessed January 31, 2008).
- (33) Bissantz, C.; Folkers, G.; Rognan, D. Protein-based virtual screening of chemical databases. 1. Evaluation of different docking/scoring combinations. *J. Med. Chem.* **2000**, *43*, 4759–67.
- (34) Blaney, J. M.; Martin, E. J. Computational approaches for combinatorial library design and molecular diversity analysis. *Curr. Opin. Chem. Biol.* **1997**, *1*, 54–9.
- (35) Tetko, I. V.; Tanchuk, V. Y.; Kasheva, T. N.; Villa, A. E. P. Internet Software for the Calculation of the Lipophilicity and Aqueous Solubility of Chemical Compounds. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 246–252.
- (36) Tetko, I. V.; Gasteiger, J.; Todeschini, R.; Mauri, A.; Livingstone, D.; Ertl, P.; Palyulin, V. A.; Radchenko, E. V.; Zefirov, N. S.; Makarenko, A. S.; Tanchuk, V. Y.; Prokopenko, V. V. Virtual computational chemistry laboratory--design and description. *J. Comput.-Aided Mol. Des.* **2005**, *19*, 453–63.
- (37) <http://www.vcclab.org/servers/> (accessed January 31, 2008).
- (38) <http://www.sun.com/2006-1113/feature/index.jsp> (accessed January 31, 2008).
- (39) Kaffe. <http://www.kaffe.org/> (accessed January 31, 2008).
- (40) GNU Classpath. <http://www.gnu.org/software/classpath/> (accessed January 31, 2008).
- (41) MOE (Molecular Operating Environment); Chemical Computing Group Inc.: Montreal, Quebec, Canada.
- (42) Viswanadhan, V. N.; Ghose, A. K.; Revankar, G. R.; Robins, R. K. Atomic Physicochemical Parameters for Three Dimensional Structure Directed Quantitative Structure-Activity Relationships. 4. Additional Parameters for Hydrophobic and Dispersive Interactions and Their Application for an Automated Superposition of Certain Naturally Occurring Nucleoside Antibiotics. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 162–172.
- (43) Wildman, S. A.; Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 868–873.
- (44) <http://www.chemaxon.com/marvin/chemaxon/marvin/help/calculator-plugins.html#logp> (accessed January 31, 2008).
- (45) <http://www.chemaxon.com/marvin/chemaxon/marvin/help/logPlogD.html> (accessed January 31, 2008).
- (46) Winiwarter, S.; Bonham, N. M.; Ax, F.; Hallberg, A.; Lennernas, H.; Karlen, A. Correlation of human jejunal permeability (in vivo) of drugs with experimentally and theoretically derived parameters. A multivariate data analysis approach. *J. Med. Chem.* **1998**, *41*, 4939–49.
- (47) Proschak, E.; Wegner, J. K.; Schuller, A.; Schneider, G.; Fechner, U. Molecular query language (MQL)--a context-free grammar for substructure matching. *J. Chem. Inf. Model.* **2007**, *47*, 295–301.
- (48) Guha, R.; Howard, M. T.; Hutchison, G. R.; Murray-Rust, P.; Rzepa, H.; Steinbeck, C.; Wegner, J.; Willighagen, E. L. The Blue Obelisk--interoperability in chemical informatics. *J. Chem. Inf. Model.* **2006**, *46*, 991–8.

CI700334F