Cheminformatics Python Microservice (CPM): unifying access to open cheminformatics toolkits

Venkata Chandrasekhar[1], Nisha Sharma[1], Jonas Schaub[1], Christoph Steinbeck[1] and Kohulan Rajan[1*]

[1] Institute for Inorganic and Analytical Chemistry, Friedrich-Schiller-University Jena, Lessingstr. 8, 07743 Jena, Germany
* Corresponding author email: kohulan.rajan@uni-jena.de

Abstract

In recent years, cheminformatics has experienced significant advancements through the development of new open-source software tools based on various cheminformatics programming toolkits. However, adopting these toolkits presents challenges, including proper installation, setup, deployment, and compatibility management. In this work, we present Cheminformatics Python Microservice (CPM). This open-source solution provides a unified interface for accessing commonly used functionalities of multiple cheminformatics toolkits, namely RDKit, Chemistry Development Kit (CDK), and Open Babel. In addition, more advanced functionalities like structure generation and Optical Chemical Structure Recognition (OCSR) are made available through CPM based on pre-existing tools. CPM also enables developers to extend the functionalities easily and seamlessly integrate with existing workflows and applications. It is built on FastAPI and containerized using Docker, making it highly scalable. An instance of the microservice is publicly available at https://api.naturalproducts.net. The source code for CPM is publicly accessible on GitHub, accompanied by comprehensive documentation, version control, and continuous integration and deployment workflows. All resources can be found at the following link: https://github.com/Steinbeck-Lab/cheminformatics-python-microservice

Graphical abstract

Keywords

CDK, RDKit, Open Babel, cheminformatics, toolkits, microservice

Introduction

In recent years, cheminformatics has seen significant advancements largely due to open cheminformatics toolkits, large-scale chemical databases, and advances in available computing power [1,2]. As a result, large amounts of chemical data can be handled and analysed efficiently, benefitting research fields like chemistry, drug discovery, and material design. There are currently multiple prominent open-source cheminformatics toolkits available, which include RDKit [3], Open Babel [4], Chemistry Development Kit (CDK) [5,6], Indigo [7] and the recently developed Python-based Informatics Kit for Analysing CHemical Units (PIKAChU) [8]. A summary of the available tools and libraries, their native programming language, the latest version, and licence information are presented in Table 1.

Table 1: Summary of widely used open-source cheminformatics toolkits.

Toolkit	Programming Language	Latest Version	Licence	Source code
Open Babel	C++	3.1.1	GNU General Public License (GPL)	https://github.com/openbabel/openbabel
CDK	Java	2.8.0	GNU Lesser General Public License v2.1	https://github.com/cdk/cdk

RDKit	C++, Python	2023.03.2	BSD 3-Clause License	https://github.com/rdkit/rdkit
PIKAChU	Python	1.0.13	MIT License	https://github.com/BTheDragonMaster/pikachu
Indigo	C/C++	1.11.0	Apache License 2.0	https://github.com/epam/Indigo

All cheminformatics toolkits mentioned above offer ready-to-use routines for common tasks like data format conversions, descriptor calculations, or structure editing. On top of these, every toolkit has a specific set of more advanced functionalities like coordinate generation, substructure analysis, structure normalisation and individual feature-richness and handling. For this reason, researchers often use multiple toolkits in their tools/workflows [9,10]. They have to familiarise themselves with the specific requirements, syntax, and algorithms of each toolkit they want to utilise and to do this most efficiently, also be familiar with the underlying programming languages such as Python, Java, or C++. It is also essential to have a thorough understanding of chemical concepts, molecular representations, and computational algorithms.

In developing cheminformatics workflows, machine learning models, or web applications, researchers and software developers need to set up a proper working environment for the toolkits to operate in conjunction with their applications. These software tools can become cumbersome to use or unreproducible when not set up properly or undocumented, which is often the case due to the complexity of the setup, lack of documentation, and lack of good Research Data Management (RDM) practices [11]. Other challenges in developing cheminformatics software, databases, and web applications result from building on top of these toolkits due to factors such as various toolkit version management [12], dependencies management [13], and maintenance [14]:

- **Software Version Management:** The purpose of this process is to effectively manage the versions of software and toolkits throughout their development and maintenance cycle. The objective is to organise and track changes, facilitate collaboration, and ensure the stability and integrity of software projects, thus increasing productivity and streamlining development processes. This process can be time-consuming and requires careful planning and organisation.

- **Dependencies**: The majority of cheminformatics tools require several interdependent libraries to function. Managing these dependencies can be challenging since developers must ensure that each dependency is installed correctly and compatible with the other dependencies. Otherwise, this can lead to conflicts between dependencies, which may cause the software to malfunction.

- **Maintenance**: It is challenging to maintain software and databases as they require regular updates and bug fixes to remain up-to-date and functional. In the special case of open-source software, this usually requires a large community of active and committed users and developers.

Most cheminformatics open-source programming toolkits have a rather high entrance barrier due to the aspects presented above. Also, the quality of available documentation varies. This is especially critical for young researchers new to the field who have to spend a lot of setup time before being able to work on their research projects. Therefore, online tools for cheminformatics are becoming increasingly popular due to their usual ease of use, adequate documentation, and the fact that no or only little programming knowledge is required [15–18]. The use of web-based solutions or solutions that are driven by Application Programming Interfaces (APIs) [19] offers a smaller entrance barrier. In addition, these services are platform-independent and can be easily integrated with software utilities, databases, repositories, and cheminformatics data management workflows[17].

Due to compatibility and maintenance issues, incorporating these toolkits into application code can be challenging. To overcome this, two common software development techniques can be used: containerization and microservices. Microservices [20], also referred to as microservice architecture, is a collection of small, autonomous services that can be deployed, scaled, and maintained individually [21]. By leveraging well-defined APIs, each microservice carries out a specific function and interacts with other microservices. These services are characterised by their granular nature and employ lightweight protocols, enabling the independent execution of each service [22]. The native installation of such services on a system makes maintenance difficult in the long run. The service may not function as a result of Operating System (OS) level updates or environmental conflicts. In addition, in the event of a service failure, it is necessary to reboot the entire system. In order to address these issues, containerization can be used for the deployment of applications. Containers are lightweight and isolated environments that enable applications and their dependencies to run consistently across a variety of environments and systems independent of the OS [23]. In addition, this provides a consistent and reproducible execution environment that ensures that applications run in the same way across development, testing, and production environments. In this work, containerization was achieved using Docker [24]. Software components can be containerized using Docker and distributed publicly via the Docker Hub [25], a cloud-based registry provided by Docker that allows developers to store, share, and distribute Docker images.

This article presents Cheminformatics Python Microservice (CPM), an open-source solution for handling chemical data and performing various cheminformatics tasks by accessing multiple cheminformatics toolkits (CDK, RDKit and Open Babel). These tasks include generating high-quality chemical structure depictions and 3D conformers, calculating molecular descriptors and IUPAC names, and converting SMILES representations of chemical structures into other machine-readable formats. The microservice can be accessed through a unified REST (REpresentational State Transfer) [26] interface, which is made available as a public server for anyone to access via api.naturalproducts.net. Alternatively, it can be deployed locally or on a (private) cluster using the Docker image and can be deployed and infinitely scaled on a private cluster managed by Kubernetes via the Helm Charts provided in just a few steps [27]. These deployment options are designed to be user-friendly, requiring no prior knowledge of the underlying toolkits or their setup environment. They make CPM suitable for a wide range of applications in academic and industrial environments. The entire CPM source code is made available to the public on GitHub: https://github.com/Steinbeck-Lab/cheminformatics-python-microservice. Users and researchers are encouraged to submit feature requests and contribute to the microservice to ensure its continued growth.

Implementation

Cheminformatics Python Microservice (CPM) is developed using FastAPI, a web framework for generating RESTful APIs using Python. FastAPI was chosen for this project due to its speed, efficiency, and suitability for building advanced Application Programming Interfaces (APIs). It enables the quick creation of robust and scalable APIs. Docker is used for containerization, and semantic versioning principles are applied to track code changes and toolkit versions. In the CPM container, RDKit and Open Babel cheminformatics toolkits are accessed natively using Python, while the Chemistry Development Kit (CDK) is integrated using JPype [28]. CPM consists of five modules, namely *chem*, *convert*, *depict*, *ocsr*, and *tools*. Compliant with the OpenAPI [29] specification version 3.1.0, this work provides standard documentation, encourages interoperability, enables automatic code generation, simplifies validation, and integrates with a variety of tools and libraries to enhance the functionality of REST (REpresentational State Transfer) APIs. An overview of the CPM architecture can be seen in Figure 1.

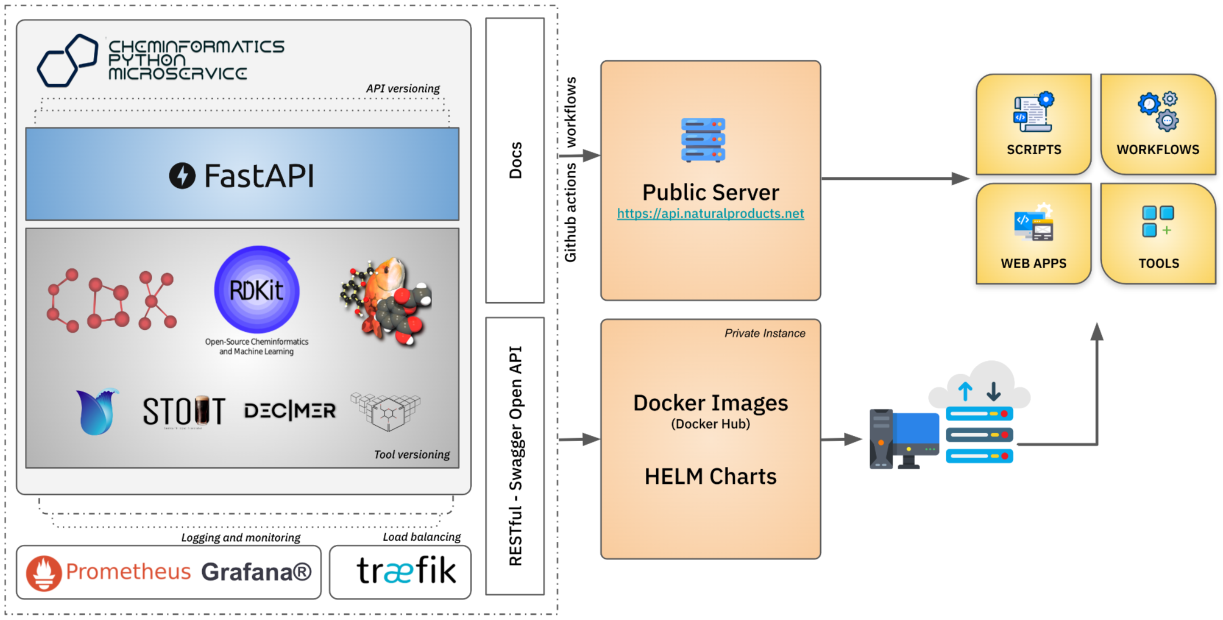


Figure 1: Overview of the Cheminformatics Python Microservice (CPM) architecture. The public server can be accessed via https://api.naturalproducts.net/latest/docs.

The *chem* module offers various functions such as descriptor calculation, stereoisomer enumeration, HOSE code [30] generation, NPLikeness score calculation [31], ClassyFire classification [32], molecular structure standardisation [33], and a preprocessing pipeline for the upcoming version of COCONUT database as explained in the results section. The *convert* module provides conversions from SMILES [34] to molecule string representations such as InChI [35,36], InChIKey [36], canonical SMILES [37], CXSMILES [38], SELFIES [39,40], and IUPAC names generation using the Smiles TO iUpac Translator (STOUT) Version 2.0 toolkit [41]. Additionally, MOL files can be generated with 2D and 3D coordinates from SMILES input. Using the *depict* module, one can generate 2D depictions of chemical structures using various settings, including an option to generate 2D representations with stereochemical annotations following the Cahn–Ingold–Prelog [42] (CIP) sequence rules. RDKit or CDK can be used to generate 2D representations. The depictions generated are displayed in an SVG format and may be scaled to fit the needs of the user. It is also possible to generate a 3D depiction [43] using this module, which is useful as a chemical structure display option for databases or as a teaching aid. The underlying 3D conformer is generated using RDKit. The *OCSR* module incorporates Deep lEarning for Chemical ImagE Recognition (DECIMER) modules [44,45] for translating images of chemical structures into machine-readable SMILES representations. These can be accessed via HTTP *POST* requests. Finally, the *tools* module is designated as a miscellaneous collection of advanced cheminformatics tools. It offers a function to generate chemical structures from a molecular formula given as input using the SURGE chemical graph generator [46]. Other functions of the *tools* module can be used to identify glycosidic moieties in input molecules and to remove them in order to generate the aglycone structure. These routines are implemented based on the Sugar Removal Utility (SRU) [47].

The functionalities offered by each main module described above can also be implemented via independent Python functions, enabling users to access CPM natively without the REST interface. Users can import and utilise them like any other package directly in their own Python code. Individual toolkit wrapper modules provide access to the three cheminformatics toolkits RDKit, CDK, and Open Babel. RDKit and Open Babel are natively accessible, while CDK, a Java library, has functionalities ported to Python using JPype. It is possible to extend the functionalities provided by cheminformatics toolkits in the future by utilising these wrapper modules. Separating them into individual modules is necessary to achieve granular control over the functions. This also ensures that the entire system will not be broken if one module is affected. The Python functions are documented separately, and the documentation can be accessed via: https://cheminformatics-python-microservice.readthedocs.io/en/latest/?badge=latest.

To ensure accurate data reproduction, a strong versioning system is essential. Best practices for research data management (RDM) [48,49] recommend documenting software and component versions separately, especially for tools like CPM with multiple dependencies. CPM uses user-friendly multi-level versioning to record API and software dependencies, avoiding confusion for users. The CPM codebase undergoes bi-annual major releases, with corresponding documentation provided for the underlying toolkits, tools, and environment dependencies for each release. The API versioning changes are released only when significant changes to the API endpoints have been made. It is possible to update the underlying cheminformatics toolkits whenever new releases are published without having to update the entire code base since the REST API remains unchanged. The CPM can be logged, monitored, and visualised using Prometheus [50] and Grafana [51] in a standalone or distributed environment. CPM can also be deployed using the Continuous Integration and Continuous Deployment (CI/CD) pipeline via GitHub Actions [52]. With CI/CD, code integration, testing, and application deployment are automated, which fosters collaboration, minimises tasks, and enables timely feedback.

Results and Discussion

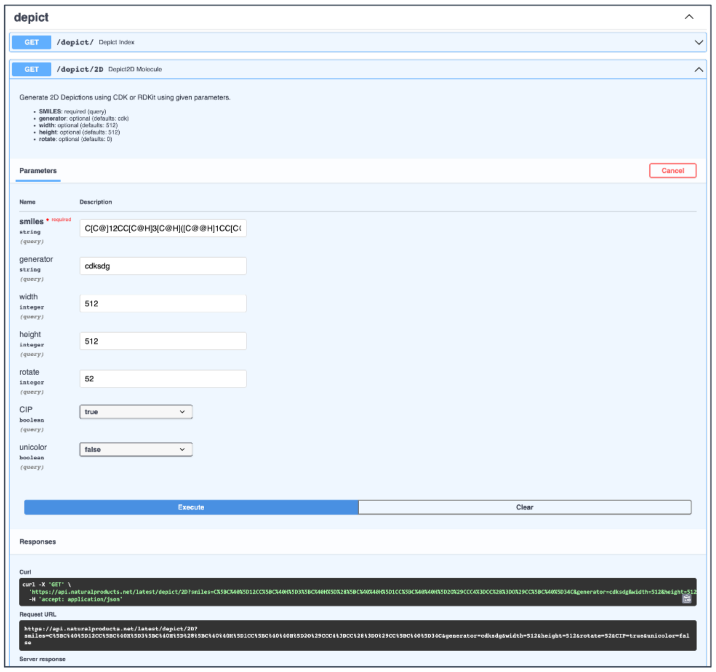
The Cheminformatics Python Microservice (CPM) provides straightforward access to open-source cheminformatics toolkits and libraries like RDKit, CDK, and OpenBabel, as well as deep learning models like Deep lEarning for Chemical ImagE Recognition(DECIMER) and Smiles TO iUpac Translator (STOUT) for OCSR applications. It offers a diverse selection of important functionalities needed by cheminformaticians on a daily basis, accessible via a RESTful interface. Additionally, CPM includes a number of widely used packages, including the ChEMBL [53] curation pipeline [33], CDK-based sugar removal utilities [47], and the open-source structure generator surge [46]. These tools facilitate the management of chemical data, format conversions, and descriptor calculation, simplifying the handling of substantial data volumes. Furthermore, they enable the development of adaptable, scalable, maintainable, and reproducible cheminformatics applications.

The main modules of CPM are well-documented and can be accessed via the following link: https://api.naturalproducts.net/. In order to obtain the data generated by the microservice, each module uses either a *GET* or a *POST* HTTP request method [54]. Most of the services provided by the *chem, convert, and depict* modules can be accessed using SMILES as an input format for molecular structures. Where the specific functionality provided by the microservice can be achieved in a similar manner with multiple toolkits, the user has the option of employing the toolkit of their choice via an additional parameter, including RDKit, CDK and Open Babel. In the *chem* module, users can input a SMILES string and perform the desired calculations. It is possible to use the *standardize* functionality via a *POST* method for the purpose of standardising molecules using the ChEMBL structure curation pipeline. The *convert* module enables users to convert SMILES representations into machine-readable formats of their choice, utilising the *GET* method. Meanwhile, the *depict* module allows users to generate customised 2D depictions, offering options for colour or black and white. The stereochemical annotations on the 2D depictions are generated using the Cahn-Ingold-Prelog(CIP) priority rules. To accomplish this, the Java package called *centres* [55,56] is utilised in conjunction with CDK. Additionally, the *depict* module produces interactive 3D depictions, further enhancing the user experience. Figure 2 illustrates the direct application of the depict module to generate a 2D depiction with CIP annotation in colour on a scale of 512 x 512 pixels and a 52° rotation. An example of the generated image with CIP annotations could be accessed directly by following this link:

https://api.naturalproducts.net/latest/depict/2D?smiles=C[C@]12CC[C@H]3[C@H]([C@@H]1CC[C@@H]2O)CCC4=CC(=O)CC[C@]34C&width=512&height=512&rotate=52&CIP=true

2D representations are produced in the form of SVG images, whereas 3D representations are returned as HTML files containing embedded JSMol objects [57,58].

A:



depict

GET /depict/ Depict Index

GET /depict/2D Depict2D Molecule

Generate 2D Depictions using CDK or RDKit using given parameters.

- **SMILES**: required (query)
- **generator**: optional (defaults: cdk)
- **width**: optional (defaults: 512)
- **height**: optional (defaults: 512)
- **rotate**: optional (defaults: 0)

Parameters

Name	Description
smiles * required string (query)	C[C@]12CC[C@H]3[C@H]([C@@H]1CC[C(
generator string (query)	cdksdg
width integer (query)	512
height integer (query)	512
rotate integer (query)	52
CIP boolean (query)	true
unicolor boolean (query)	false

Responses

Curl

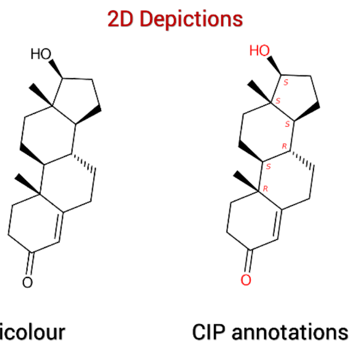
```
curl -X 'GET' \
  'https://api.naturalproducts.net/latest/depict/2D?smiles=C%5BC%40%5D12CC%5BC%40H%5D3%5BC%40H%5D%28%5BC%40%40H%5D1CC%5BC%40%40H%5D2O%29CCC4%3DCC%28%3DO%29CC%5BC%40%5D34C&generator=cdksdg&width=512&height=512
  -H 'accept: application/json'
```

Request URL

```
https://api.naturalproducts.net/latest/depict/2D?
smiles=C%5BC%40%5D12CC%5BC%40H%5D3%5BC%40H%5D%28%5BC%40%40H%5D1CC%5BC%40%40H%5D2O%29CCC4%3DCC%28%3DO%29CC%5BC%40%5D34C&generator=cdksdg&width=512&height=512&rotate=52&CIP=true&unicolor=fal
se
```

Server response

B:



C:

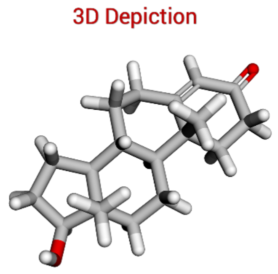


Figure 2: A: The screenshot showcases the output of the depict module, displaying the depiction options for the input molecule Testosterone given as a SMILES string. B: The output of the depict module with Unicolor and CIP annotations for 2D depictions, and C: the interactive 3D depiction [59].

Through REST API calls, the CPM *tools* and *OCSR* modules offer easy and convenient access to the underlying functionality of the implemented tools. For example, the *tools* module allows users to utilise the open-source chemical structure generator SURGE or *surge* to generate chemically valid structures based on a provided molecular formula. A list of SMILES representations is returned for each chemical structure produced by the structure generator corresponding to the given molecular formula. In order to address the resource-intensive nature of the chemical structure generation process, a maximum limit of 10 heavy atoms per molecular formula has been imposed. This restriction promotes efficiency and applies exclusively to the public instance. The *tools* module also allows users to access the Sugar Removal Utility (SRU) to detect and remove linear and circular sugar moieties in/from input structures. Users can access the DECIMER toolkit through the *ocsr* module, which enables identification, segmentation, and translation into machine-readable representations of chemical structure depictions from the scientific literature.

Currently, the CPM is available to the public via https://api.naturalproducts.net/latest/docs and in the back end, the container is running on a compute server with the processor Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and 16GB of RAM.

Use of CPM in the COCONUT database

CPM is extensively employed in the upcoming version of the COCONUT (COlleCtion of Open Natural prodUcTs) database that is currently under development. With the CPM *depict* module, it becomes feasible to present all the natural product structure data entries within COCONUT in both 2D and 3D formats (Figure 3). CPM also includes the generation of molecular descriptors and further preprocessing steps for data submission to the COCONUT database.

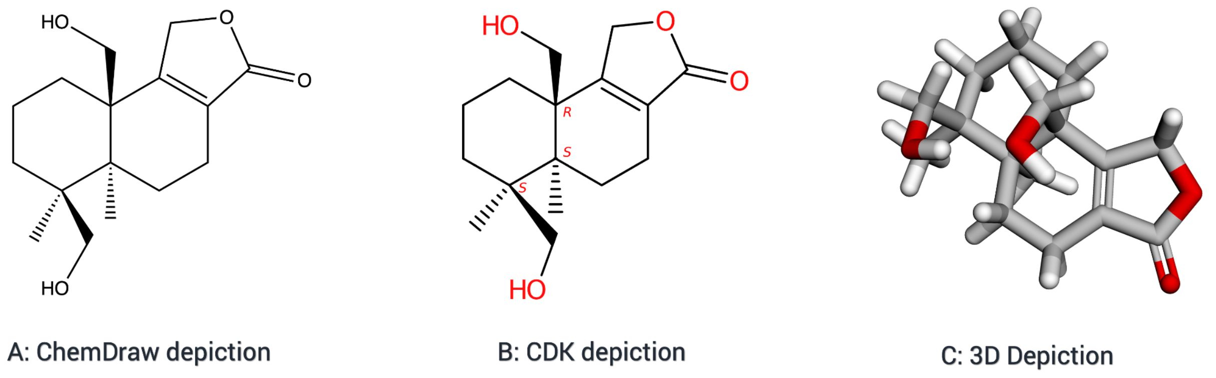
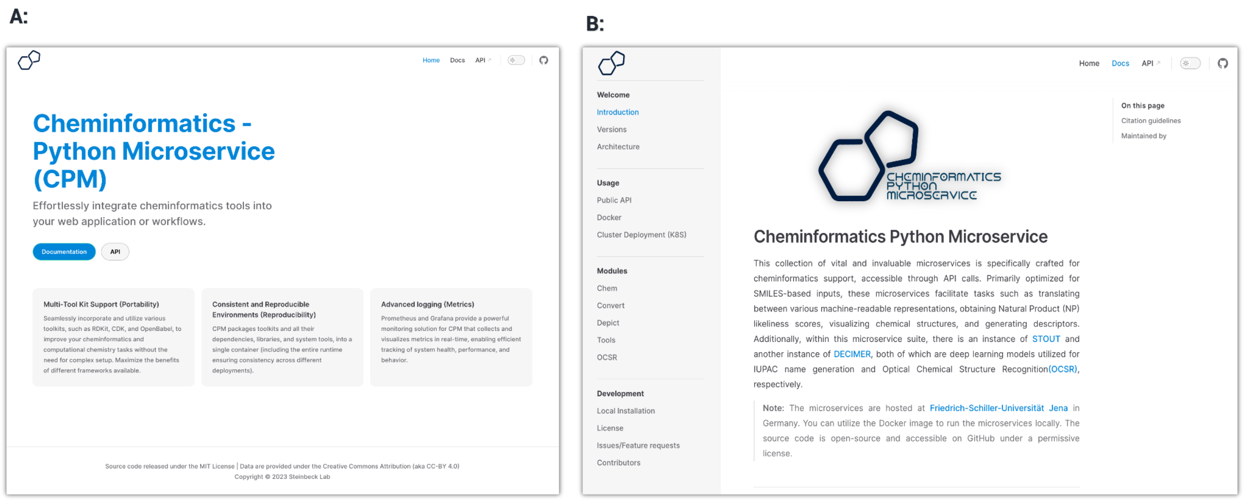


Figure 3: The molecule astellolide W [60] was extracted from the referenced article using DECIMER. The structure is depicted using ChemDraw (A) and CDK (B) for 2D representations, while the 3D depiction (C) was created using CPM.

Documentation

CPM offers detailed documentation alongside its source code, ensuring that users can easily access and navigate CPM without a high entry barrier. The documentation provides clear guidance on how to effectively utilise, deploy, install, and manage licensing information associated with CPM. Figure 4 offers a glimpse of the documentation that is deployed using GitHub Pages and can be accessed at the following URL:
https://steinbeck-lab.github.io/cheminformatics-python-microservice.



Cheminformatics - Python Microservice (CPM)

Effortlessly integrate cheminformatics tools into your web application or workflows.

Multi-Tool Kit Support (Portability)

Seamlessly incorporate and utilize various toolkits, such as RDKit, CDK, and OpenBabel, to improve your cheminformatics and computational chemistry tasks without the need for complex setup. Maximize the benefits of different frameworks available.

Consistent and Reproducible Environments (Reproducibility)

CPM packages toolkits and all their dependencies, libraries, and system tools, into a single container (including the entire runtime ensuring consistency across different deployments).

Advanced logging (Metrics)

Prometheus and Grafana provide a powerful monitoring solution for CPM that collects and visualizes metrics in real-time, enabling efficient tracking of system health, performance, and behavior.

Source code released under the MIT License | Data are provided under the Creative Commons Attribution (aka CC-BY 4.0)
Copyright © 2023 Steinbeck Lab

Welcome
Introduction
Versions
Architecture

Usage
Public API
Docker
Cluster Deployment (K8S)

Modules
Chem
Convert
Depict
Tools
OCSR

Development
Local Installation
License
Issues/Feature requests
Contributors

CHEMINFORMATICS PYTHON MICROSERVICE

Cheminformatics Python Microservice

This collection of vital and invaluable microservices is specifically crafted for cheminformatics support, accessible through API calls. Primarily optimized for SMILES-based inputs, these microservices facilitate tasks such as translating between various machine-readable representations, obtaining Natural Product (NP) likeliness scores, visualizing chemical structures, and generating descriptors. Additionally, within this microservice suite, there is an instance of STOUT and another instance of DECIMER, both of which are deep learning models utilized for IUPAC name generation and Optical Chemical Structure Recognition(OCSR), respectively.

Note: The microservices are hosted at Friedrich-Schiller-Universität Jena in Germany. You can utilize the Docker image to run the microservices locally. The source code is open-source and accessible on GitHub under a permissive license.

On this page
Citation guidelines
Maintained by

Figure 4: Screenshots of the CPM landing page (A) and detailed documentation (B)

Performance and scalability

CPM is specifically designed to scale at large, in a local deployment as well as an orchestrated cluster. CPM can be configured to run on multiple workers when deployed independently. If deployed through Docker Compose or over a Kubernetes cluster, CPM can be auto-scaled infinitely to handle incoming requests. Helm Chart or the Docker Compose file provided in the CPM codebase enables scaling without any additional setup. Prometheus and Grafana are used to monitor the request count over time. These logs and other performance indicators will allow to facilitate the scaling of CPM workers based on user demand in the future.

To determine the scalability of the CPM server, stress testing has been performed using Vegeta [61]. To determine the maximum throughput the CPM can handle when deployed on a machine with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and 16GB of RAM requests in small increments were added using Vegeta and the delivered throughput was measured until a limit was reached (Figure 5). This stress test indicated that the CPM reached its maximum capacity when handling approximately 2500 requests per second for echo requests, after which the success rate began to decline. When queried with a task converting a SMILES string into a molecular structure with 2D coordinates using RDKit, Figure 5 clearly demonstrates that CPM can effectively handle a wide range of requests, ranging from 50 to 500 per second. With an increase in the input molecule size and number of requests per second, this rate starts to deteriorate. A detailed description of the stress test can be found in the supplementary information.

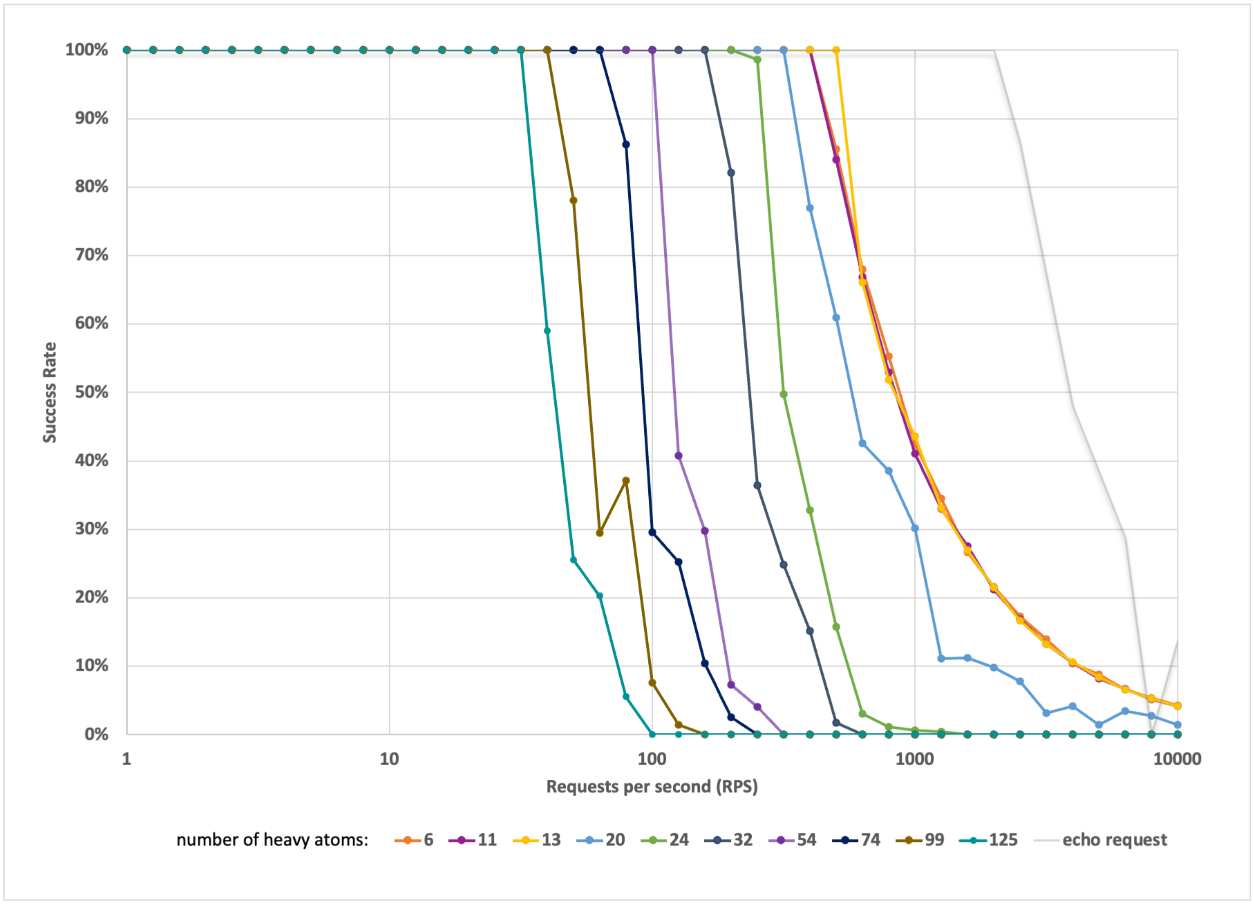


Figure 5: A snapshot of performance comparing the success rate with an increase in the number of requests per second as well as with an increase in the number of heavy atoms in the input. The grey line represents the maximum number of requests the server could process (echo request).

Nevertheless, the default configuration of CPM ensures excellent stability in handling user requests and effectively manages the computation required.

However, limitations are imposed on CPM public instance usage depending on the tools and routines deployed. For example, restrictions are imposed in the structure generator when dealing with molecules containing more than ten heavy atoms, as this service demands significant computational resources. Access to mass mining through the DECIMER endpoint is also restricted. Although these restrictions might be reconsidered in the future, depending on the user demand and computation resource availability.

Conclusion

The Cheminformatics Python Microservice (CPM) is a self-contained, web-based service that operates independently of the operating system. This can be used by researchers with no or limited programming experience to perform daily cheminformatics tasks effortlessly via the API hosted at https://api.naturalproducts.net. This service allows a diverse range of open-source cheminformatics toolkits to be accessed without requiring any software installation or environment setups. To the best of our knowledge, CPM is currently the sole cheminformatics microservice offering users access to multiple toolkits, as well as to additional tools such as the open-source structure generator surge, Sugar Removal Utility(SRU), and the DECIMER OCSR tools. CPM is designed to be user-friendly, easily extendable, deployable, and scalable. It can be accessed through the public API or hosted on private clusters or single machines.

The integration of multiple cheminformatics toolkits in a centralised platform enhances user accessibility to cheminformatics utilities. By using industry-standard monitoring, deployment, documentation, and code quality standards, this project demonstrates software development in a user-focused manner. By making the source code and the documentation completely open and public, we aim to make valuable contributions to the scientific community while also enabling the submission of feature requests for future enhancements. In future, CPM will continue to evolve to be an invaluable asset for cheminformaticians due to its inherent expandability, follows a clear semantic versioning system, bi-annual updates, and comprehensive documentation. These characteristics significantly facilitate data reproduction and reuse for researchers, thereby fostering better collaboration among peers.

List of abbreviations

2D - Two Dimensional
3D - Three Dimensional
APIs - Application Programming Interfaces
BSD - Berkeley Software Distribution
CC - Creative Commons
CDK - Chemistry Development Kit
CI/CD - Continuous Integration and Continuous Deployment
CIP - Cahn–Ingold–Prelog
COCONUT - COlleCtion of Open Natural prodUcTs
CPM - Cheminformatics Python Microservice
CPU - Central Processing Unit
CXSMILES - Chemaxon Extended Simplified Molecular Input Line Entry Specification
DECIMER - Deep lEarning for Chemical ImagE Recognition
GB - Gigabyte
GHz - Gigahertz
GNU - GNU's Not Unix
GPL - General Public License
HOSE - Hierarchically Ordered Spherical Environment
HTTP - Hypertext Transfer Protocol
InChI - International Chemical Identifier
IUPAC - International Union of Pure and Applied Chemistry
JDK - Java Development Kit

JPype - Java for Python
JSMol - JavaScript Molecular viewer
MIT - Massachusetts Institute of Technology
NPLikeness - Natural Product Likeness
OCSR - Optical Chemical Structure Recognition
OS - Operating System
PC - Personal Computer
PIKAChU - Python-based Informatics Kit for Analysing CHemical Units
RAM - Random Access Memory
RDM - Research Data Management
REST - REpresentational State Transfer
SELFIES - SELF-referencIng Embedded Strings
SMILES - Simplified Molecular Input Line Entry Specification
SRU - Sugar Removal Utility
STOUT - SMILES-TO-IUPAC-name translator
SVG - Scalable Vector Graphics

Availability and requirements

- **Project name:** Cheminformatics Python Microservice
- **Project home page:** https://github.com/Steinbeck-Lab/cheminformatics-python-microservice
- **Docker Image:** https://hub.docker.com/r/caffeinejena/cheminformatics-python-microservice
- **Helm Chart repo :** https://nfdi4chem.github.io/repo-helm-charts/
- **Helm Chart GitHub:** https://github.com/NFDI4Chem/repo-helm-charts
- **Current version:** v1.0.0
- **DOI of archived current release:** https://doi.org/10.5281/zenodo.8112749
- **Operating system(s):** Independent
- **Programming language:** Python 3, HTML
- **Requirements:**
    - API calls:
        - Internet connection and command line interface or a web browser
    - Run locally:
        - Docker - To use CPM as a Docker container
        - Conda environment - to use CPM natively without Docker
    - Dependencies (managed by Docker/Conda):
        - Python packages: uvicorn>=0.15.0,<0.16.0, fastapi>=0.80.0, fastapi-pagination==0.10.0, fastapi-versioning>=0.10.0, prometheus-fastapi-instrumentator, jpype1==1.4.1, jinja2, pandas, chembl_structure_pipeline, HOSE_code_generator @ git+https://github.com/Ratsemaat/HOSE_code_generator, websockets==10.4, pillow==9.4.0, opencv-python==4.7.0.68, matplotlib==3.4.3, scikit-image, pdf2image==1.16.2, IPython,

        pystow>=0.4.9, unicodedata2==15.0.0, efficientnet, tensorflow==2.12.0, pillow-heif==0.10.0, selfies>=2.1.1, httpx>=0.24.1, keras_preprocessing==1.1.2, decimer-segmentation>=1.1.2, STOUT-pypi>=2.0.5 and decimer>=2.2.0
        - Java: OpenJDK for Java 11
        - Java Libraries: CDK 2.8.0 , SRU 1.3.2 and Centres 1.0

- **License:** MIT
- **Documentation:**
    - Home page: https://steinbeck-lab.github.io/cheminformatics-python-microservice/
    - CPM API: https://api.naturalproducts.net/latest/docs
    - Python Documention: https://cheminformatics-python-microservice.readthedocs.io/en/latest/?badge=latest

- **Any restrictions to use by non-academics:** None

Declarations

Competing interests

The authors do not have any competing interests to declare.

Funding

Open Access funding enabled and organized by Projekt DEAL. This work was funded by the German Research Foundation under project number: 239748522- SFB 1127 ChemBioSys (Project INF) and NFDI4Chem under project number 441958208.

Authors' contributions

VC and KR initiated the project and developed the software. NS developed, automated and documented the work. JS helped with Java porting. KR designed the logo. VC, JS, and KR wrote the paper together. KR and CS supervised the study. All authors read and approved the final manuscript.

References

1. Ambure, P.; Aher, R.B.; Roy, K. Recent Advances in the Open Access Cheminformatics

Toolkits, Software Tools, Workflow Environments, and Databases. In *Methods in Pharmacology and Toxicology*; Methods in pharmacology and toxicology; Springer New York: New York, NY, 2014; pp. 257–296 ISBN 9781493935192.

2. Wegner, J.K.; Sterling, A.; Guha, R.; Bender, A.; Faulon, J.-L.; Hastings, J.; O'Boyle, N.; Overington, J.; Van Vlijmen, H.; Willighagen, E. Cheminformatics. *Commun. ACM* **2012**, *55*, 65–75, doi:10.1145/2366316.2366334.
3. Landrum, G.; Others RDKit: Open-Source Cheminformatics Software.(2016). *URL http://www. rdkit. org/, https://github. com/rdkit/rdkit* **2016**.
4. O'Boyle, N.M.; Banck, M.; James, C.A.; Morley, C.; Vandermeersch, T.; Hutchison, G.R. Open Babel: An Open Chemical Toolbox. *J. Cheminform.* **2011**, *3*, 33, doi:10.1186/1758-2946-3-33.
5. Willighagen, E.L.; Mayfield, J.W.; Alvarsson, J.; Berg, A.; Carlsson, L.; Jeliazkova, N.; Kuhn, S.; Pluskal, T.; Rojas-Chertó, M.; Spjuth, O.; et al. The Chemistry Development Kit (CDK) v2.0: Atom Typing, Depiction, Molecular Formulas, and Substructure Searching. *Journal of Cheminformatics* **2017**, *9*. doi: 10.1186/s13321-017-0220-4
6. 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, doi:10.1021/ci025584y.
7. Indigo Toolkit Available online: https://lifescience.opensource.epam.com/indigo/ (accessed on 25 June 2020).
8. Terlouw, B.R.; Vromans, S.P.J.M.; Medema, M.H. PIKAChU: A Python-Based Informatics Kit for Analysing Chemical Units. *J. Cheminform.* **2022**, *14*, 34, doi:10.1186/s13321-022-00616-5.
9. Brinkhaus, H.O.; Rajan, K.; Zielesny, A.; Steinbeck, C. RanDepict: Random Chemical Structure Depiction Generator. *J. Cheminform.* **2022**, *14*, 31, doi:10.1186/s13321-022-00609-4.
10. Zulfiqar, M.; Gadelha, L.; Steinbeck, C.; Sorokina, M.; Peters, K. MAW: The Reproducible Metabolome Annotation Workflow for Untargeted Tandem Mass Spectrometry. *J. Cheminform.* **2023**, *15*, 32, doi:10.1186/s13321-023-00695-y.
11. Ashiq, M.; Usmani, M.H.; Naeem, M. A Systematic Literature Review on Research Data Management Practices and Services. *Glob. Knowl. Mem. Commun.* **2022**, *71*, 649–671, doi:10.1108/gkmc-07-2020-0103.
12. Van Gurp, J.; Prehofer, C. Version Management Tools as a Basis for Integrating Product Derivation and Software Product Families. In Proceedings of the Proceedings of the Workshop on Variability Management-Working with Variability Mechanisms at SPLC; 2006; pp. 48–58.
13. Esparrachiari, S.; Reilly, T.; Rentz, A. Tracking and Controlling Microservice Dependencies. *ACM Queue* **2018**, *16*, 44–65, doi:10.1145/3277539.3277541.

14. Canfora, G.; Cimitile, A. SOFTWARE MAINTENANCE. In *Handbook of Software Engineering and Knowledge Engineering*; World Scientific Publishing Company, 2001; pp. 91–120 ISBN 9789810249731.

15. Huang, Y.-C.; Tremouilhac, P.; Nguyen, A.; Jung, N.; Bräse, S. ChemSpectra: A Web-Based Spectra Editor for Analytical Data. *J. Cheminform.* **2021**, *13*, 8, doi:10.1186/s13321-020-00481-0.

16. Jablonka, K.M.; Moosavi, S.M.; Asgari, M.; Ireland, C.; Patiny, L.; Smit, B. A Data-Driven Perspective on the Colours of Metal-Organic Frameworks. *Chem. Sci.* **2020**, *12*, 3587–3598, doi:10.1039/d0sc05337f.

17. Patiny, L.; Borel, A. ChemCalc: A Building Block for Tomorrow's Chemical Infrastructure. *J. Chem. Inf. Model.* **2013**, *53*, 1223–1228, doi:10.1021/ci300563h.

18. Patiny, L.; Zasso, M.; Kostro, D.; Bernal, A.; Castillo, A.M.; Bolaños, A.; Asencio, M.A.; Pellet, N.; Todd, M.; Schloerer, N.; et al. The C6H6 NMR Repository: An Integral Solution to Control the Flow of Your Data from the Magnet to the Public. *Magn. Reson. Chem.* **2018**, *56*, 520–528, doi:10.1002/mrc.4669.

19. Ofoeda, J.; Boateng, R.; Effah, J. Application programming interface (API) research. *Int. J. Enterp. Inf. Syst.* **2019**, *15*, 76–95, doi:10.4018/ijeis.2019070105.

20. Newman, S. *Building Microservices*; O'Reilly Media, 2015; ISBN 9781491950357.

21. Wolff, E. *Microservices: Flexible Software Architecture*; Addison-Wesley, 2017; ISBN 9780134602417.

22. Chen, L. Microservices: Architecting for Continuous Delivery and DevOps. In Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA); April 2018; pp. 39–397.

23. Containerization Explained Available online: https://www.ibm.com/topics/containerization (accessed on 22 June 2023).

24. Turnbull, J. *The Docker Book: Containerization Is the New Virtualization*; James Turnbull, 2014; ISBN 9780988820203.

25. Cook, J. Docker Hub. In *Docker for Data Science: Building Scalable and Extensible Data Infrastructure Around the Jupyter Notebook Server*; Cook, J., Ed.; Apress: Berkeley, CA, 2017; pp. 103–118 ISBN 9781484230121.

26. Sohan, S.M.; Maurer, F.; Anslow, C.; Robillard, M.P. A Study of the Effectiveness of Usage Examples in REST API Documentation. In Proceedings of the 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC); October 2017; pp. 53–61.

27. Gokhale, S.; Poosarla, R.; Tikar, S.; Gunjawate, S.; Hajare, A.; Deshpande, S.; Gupta, S.; Karve, K. Creating Helm Charts to Ease Deployment of Enterprise Application and Its Related Services in Kubernetes. In Proceedings of the 2021 International Conference on Computing, Communication and Green Engineering (CCGE); September 2021; pp. 1–5.

28. Nelson, K.E.; Scherer, M.K.; Others *JPype*; Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2020.

29. Initiative, O. Openapi Initiative, 2021. *. org/web/20210210172220/https://www. openapis. Org*.

30. Bremser, W. Hose — a Novel Substructure Code. *Anal. Chim. Acta* **1978**, *103*, 355–365, doi:10.1016/S0003-2670(01)83100-7.

31. Ertl, P.; Roggo, S.; Schuffenhauer, A. Natural Product-Likeness Score and Its Application for Prioritization of Compound Libraries. *J. Chem. Inf. Model.* **2008**, *48*, 68–74, doi:10.1021/ci700286x.

32. Djoumbou Feunang, Y.; Eisner, R.; Knox, C.; Chepelev, L.; Hastings, J.; Owen, G.; Fahy, E.; Steinbeck, C.; Subramanian, S.; Bolton, E.; et al. ClassyFire: Automated Chemical Classification with a Comprehensive, Computable Taxonomy. *J. Cheminform.* **2016**, *8*, 61, doi:10.1186/s13321-016-0174-y.

33. Bento, A.P.; Hersey, A.; Félix, E.; Landrum, G.; Gaulton, A.; Atkinson, F.; Bellis, L.J.; De Veij, M.; Leach, A.R. An Open Source Chemical Structure Curation Pipeline Using RDKit. *J. Cheminform.* **2020**, *12*, 51, doi:10.1186/s13321-020-00456-1.

34. Weininger, D. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36, doi:10.1021/ci00057a005.

35. Heller, S.R.; McNaught, A.; Pletnev, I.; Stein, S.; Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J. Cheminform.* **2015**, *7*, 23, doi:10.1186/s13321-015-0068-4.

36. The IUPAC International Chemical Identifier (InChI). *Chemistry International -- Newsmagazine for IUPAC* **2009**, *31*, 7–9, doi:10.1515/ci.2009.31.1.7.

37. Weininger, D.; Weininger, A.; Weininger, J.L. SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 97–101, doi:10.1021/ci00062a008.

38. Chemaxon Extended SMILES and SMARTS - CXSMILES and CXSMARTS Available online: https://docs.chemaxon.com/display/docs/chemaxon-extended-smiles-and-smarts-cxsmiles-and-cxsmarts.md (accessed on 22 June 2023).

39. Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation. *Mach. Learn.: Sci. Technol.* **2020**, *1*, 045024, doi:10.1088/2632-2153/aba947.

40. Krenn, M.; Ai, Q.; Barthel, S.; Carson, N.; Frei, A.; Frey, N.C.; Friederich, P.; Gaudin, T.; Gayle, A.A.; Jablonka, K.M.; et al. SELFIES and the Future of Molecular String Representations. *Patterns Prejudice* **2022**, *3*, 100588, doi:10.1016/j.patter.2022.100588.

41. Rajan, K.; Zielesny, A.; Steinbeck, C. STOUT: SMILES to IUPAC Names Using Neural Machine Translation. *J. Cheminform.* **2021**, *13*, 34, doi:10.1186/s13321-021-00512-4.

42. Cahn, R.S.; Ingold, C.; Prelog, V. Specification of Molecular Chirality. *Angew. Chem. Int. Ed Engl.* **1966**, *5*, 385–415, doi:10.1002/anie.196603851.

43. Rego, N.; Koes, D. 3Dmol.js: Molecular Visualization with WebGL. *Bioinformatics* **2015**, *31*, 1322–1324, doi:10.1093/bioinformatics/btu829.

44. Rajan, K.; Brinkhaus, H.O.; Sorokina, M.; Zielesny, A.; Steinbeck, C. DECIMER-Segmentation: Automated Extraction of Chemical Structure Depictions from Scientific Literature. *J. Cheminform.* **2021**, *13*, 20, doi:10.1186/s13321-021-00496-1.

45. Rajan, K.; Brinkhaus, H.O.; Isabel Agea, M.; Zielesny, A.; Steinbeck, C. DECIMER.ai - An Open Platform for Automated Optical Chemical Structure Identification, Segmentation and Recognition in Scientific Publications. *ChemRxiv* **2023**, doi:10.26434/chemrxiv-2023-xhcx9.

46. McKay, B.D.; Yirik, M.A.; Steinbeck, C. Surge: A Fast Open-Source Chemical Graph Generator. *J. Cheminform.* **2022**, *14*, 24, doi:10.1186/s13321-022-00604-9.

47. Schaub, J.; Zielesny, A.; Steinbeck, C.; Sorokina, M. Too Sweet: Cheminformatics for Deglycosylation in Natural Products. *J. Cheminform.* **2020**, *12*, 67, doi:10.1186/s13321-020-00467-y.

48. Wilkinson, M.D.; Dumontier, M.; Aalbersberg, I.J.J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.-W.; da Silva Santos, L.B.; Bourne, P.E.; et al. The FAIR Guiding Principles for Scientific Data Management and Stewardship. *Sci Data* **2016**, *3*, 160018, doi:10.1038/sdata.2016.18.

49. Hoyt, C.T.; Zdrazil, B.; Guha, R.; Jeliazkova, N.; Martinez-Mayorga, K.; Nittinger, E. Improving Reproducibility and Reusability in the Journal of Cheminformatics. *J. Cheminform.* **2023**, *15*, 62, doi:10.1186/s13321-023-00730-y.

50. Prometheus Overview Available online: https://prometheus.io/docs/introduction/overview/ (accessed on 23 June 2023).

51. Chakraborty, M.; Kundan, A.P. Grafana. In *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*; Chakraborty, M., Kundan, A.P., Eds.; Apress: Berkeley, CA, 2021; pp. 187–240 ISBN 9781484268889.

52. Chandrasekara, C.; Herath, P. Introduction to GitHub Actions. In *Hands-on GitHub Actions: Implement CI/CD with GitHub Action Workflows for Your Applications*; Chandrasekara, C., Herath, P., Eds.; Apress: Berkeley, CA, 2021; pp. 1–8 ISBN 9781484264645.

53. Gaulton, A.; Hersey, A.; Nowotka, M.; Bento, A.P.; Chambers, J.; Mendez, D.; Mutowo, P.; Atkinson, F.; Bellis, L.J.; Cibrián-Uhalte, E.; et al. The ChEMBL Database in 2017. *Nucleic Acids Res.* **2017**, *45*, D945–D954, doi:10.1093/nar/gkw1074.

54. Taneja, S.; Gupta, P.R. Python as a Tool for Web Server Application Development. *JIMS8I-International Journal of Information* **2014**.

55. Hanson, R.M.; Musacchio, S.; Mayfield, J.W.; Vainio, M.J.; Yerin, A.; Redkin, D. Algorithmic Analysis of Cahn-Ingold-Prelog Rules of Stereochemistry: Proposals for Revised Rules and a Guide for Machine Implementation. *J. Chem. Inf. Model.* **2018**, *58*, 1755–1765, doi:10.1021/acs.jcim.8b00324.

56. John, M. (2018). Centres: Perception and Labelling of Stereogenic Centres in Chemical Structures *(Version 1.0) [Computer software]*; Github.

57. Herráez, A. Biomolecules in the Computer: Jmol to the Rescue. *Biochem. Mol. Biol. Educ.* **2006**, *34*, 255–261, doi:10.1002/bmb.2006.494034042644.

58. Hanson, R.M.; Prilusky, J.; Renjian, Z.; Nakane, T.; Sussman, J.L. JSmol and the next-Generation Web-Based Representation of 3D Molecular Structure as Applied toProteopedia. *Isr. J. Chem.* **2013**, *53*, 207–216, doi:10.1002/ijch.201300024.

59. PubChem Testosterone Available online: https://pubchem.ncbi.nlm.nih.gov/compound/6013 (accessed on 23 June 2023).

60. Dai, G.; Sun, J.; Peng, X.; Shen, Q.; Wu, C.; Sun, Z.; Sui, H.; Ren, X.; Zhang, Y.; Bian, X. Astellolides R-W, Drimane-Type Sesquiterpenoids from an Aspergillus Parasiticus Strain Associated with an Isopod. *J. Nat. Prod.* **2023**, doi:10.1021/acs.jnatprod.3c00215.

61. Senart, T. *Vegeta: HTTP Load Testing Tool and Library. It's over 9000!*; Github.

Supplementary Information

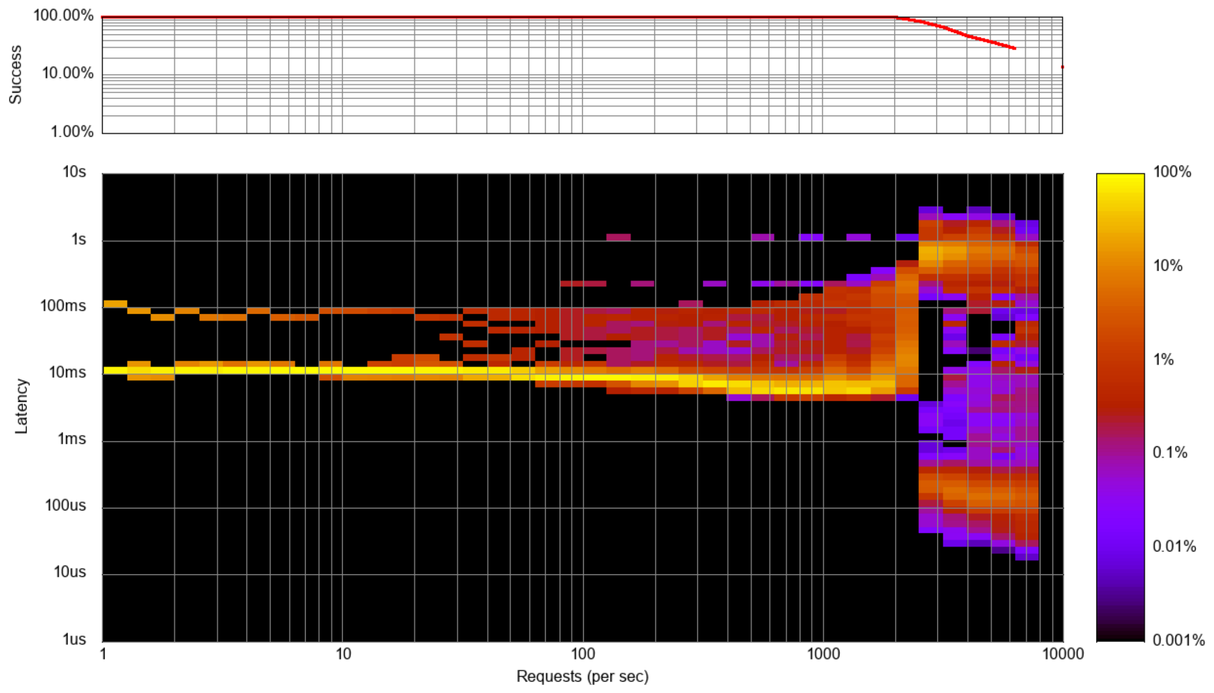
Performance testing: Latency distribution and success rate at each request rate

Ramping Load: To determine the maximum throughput the CPM (Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and 16GB of RAM) can handle, we used the Vegeta (https://github.com/tsenart/vegeta) tool to add load in small increments and measure the delivered throughput until a limit is reached. The results are then graphed to show the scalability profile. CPM's ***create 2D coordinates from SMILES*** endpoint is chosen as the test request, and the scalability profiles are generated for requests with SMILES starting from 6 heavy atom count to 125 heavy atoms (randomly sampled from the COCONUT database). The scalability profiles are then inspected visually to assess the server performance.

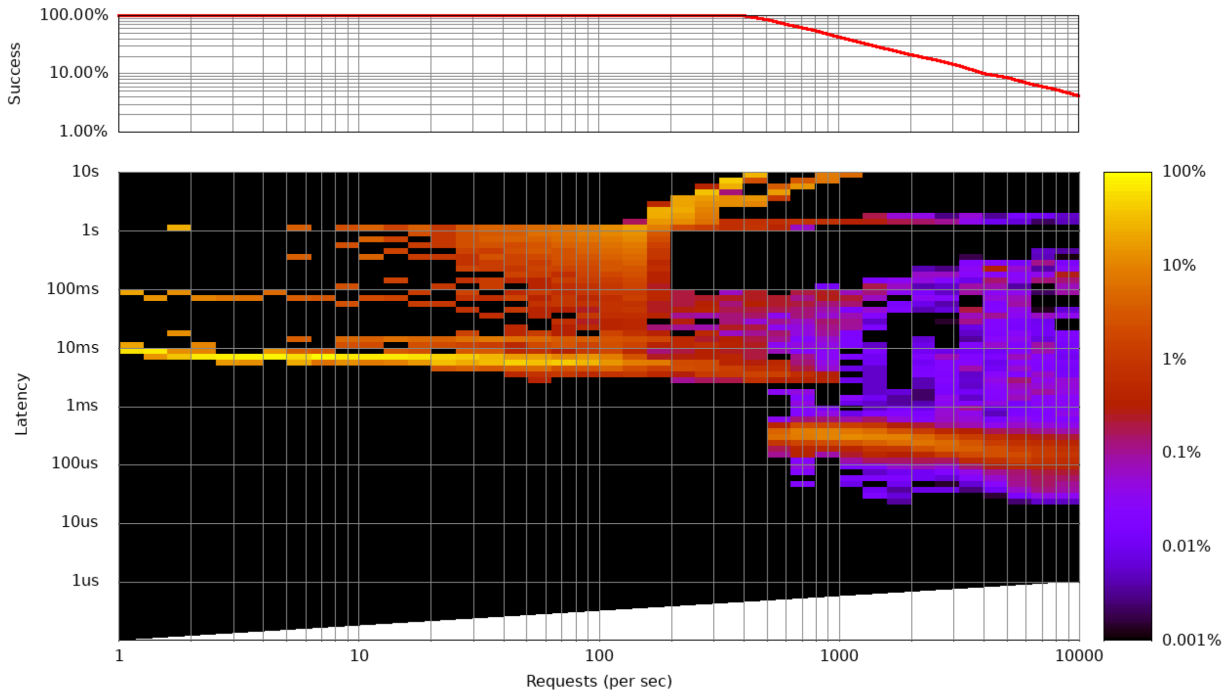
SMILES	Coconut ID	Heavy atom count
Echo request	-	-
O=C(N)C(F)Cl	CNP0031243	6
OC1CC2N(C)C(C1)C(O)C2	CNP0205916	11
O=C1C=C(O)C(C(=O)C1(C)C)(C)C	CNP0103604	13
OOC(C=CC#CC#CC(O)C=C)CCCCCCC	CNP0171228	20
O=C1NC(C)C2C(NC(=O)N(C)C2C3=CC=C(C=C3)[NH+]([O-])O)N1C	CNP0469758	24
O=C(C1=C(O)C=C(OC)C=C1OC2OC(CO)C(O)C(O)C2O)CCC3=CC=C(O)C=C3	CNP0267058	32
O=C1OC(C)CCCC(=O)CCCC=CC=2C=C(O)C(=C(O)C12)C(C3=CC(OC)=C(OC)C(OC)=C3)CC(=O)NC(C4=NC=5C=CC=CC5N4)C(C)CC	CNP0319430	54
O=C(O)C1OC(OC2CCC3(C)C(CCC4(C)C3CC=C5C6CC(C)(C)CCC6(C(=O)OC7OCC(O)C(O)C7OC8OC(C)C(OC9OCC(O)C(O)C9O)C(O)C8O)C(O)CC54C)C2(C)C)C(O)C(O)C1O	CNP0187011	74
O=C(O)CCC(NC(=O)C(NC(=O)C=1N=C(SC1)C(=O)C(C)CC)CC(C)C)C(=O)NC(C(=O)NCCCCC2NC(=O)C(NC(=O)C(NC(=O)C(NC(=O)C(NC(=O)C(NC(=O)C(NC2=O)CCCN)C(C)CC)CC=3C=CC=CC3)CC4=CN=CN4)CC(=O)O)CC(=O)N)C(C)C	CNP0170666	99
O=C(O)C1(OC2C(O)C(OC(CO)C2O)OC3C(O)C(OC(OC4C(OC(OC5C(O)C(O)C(OCC(NC(=O)CCCCCCCCCCCCCCC)C(O)CCCCCCCCCCCCCCC)OC5CO)C(O)C4OC6(OC(C(O)C(O)CO)C(NC(=O)C)C(O)C6)C(=O)O)CO)C3CC(=O)C)CO)OC(C(O)C(O)CO)C(NC(=O)C)C(O)C1	CNP0001893	125

Note: In the following plots,all axes are logarithmic. The complete performance results are available on Zenodo - https://zenodo.org/record/8124294

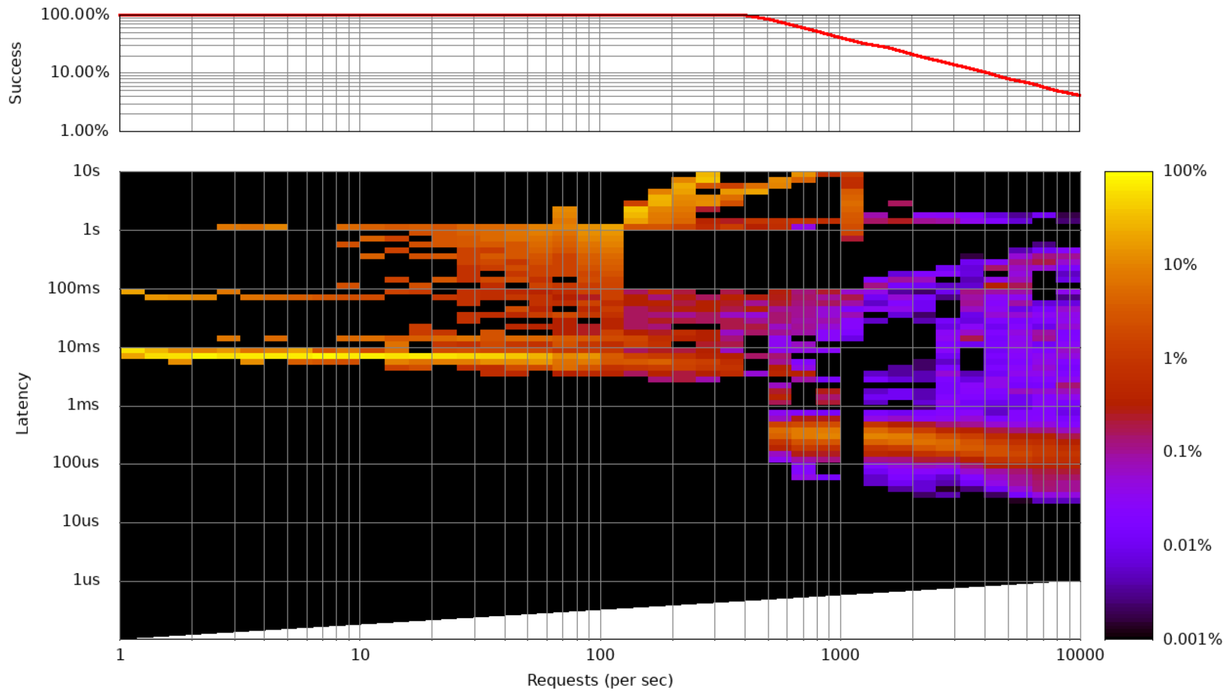
(A) Scalability profile - Echo request



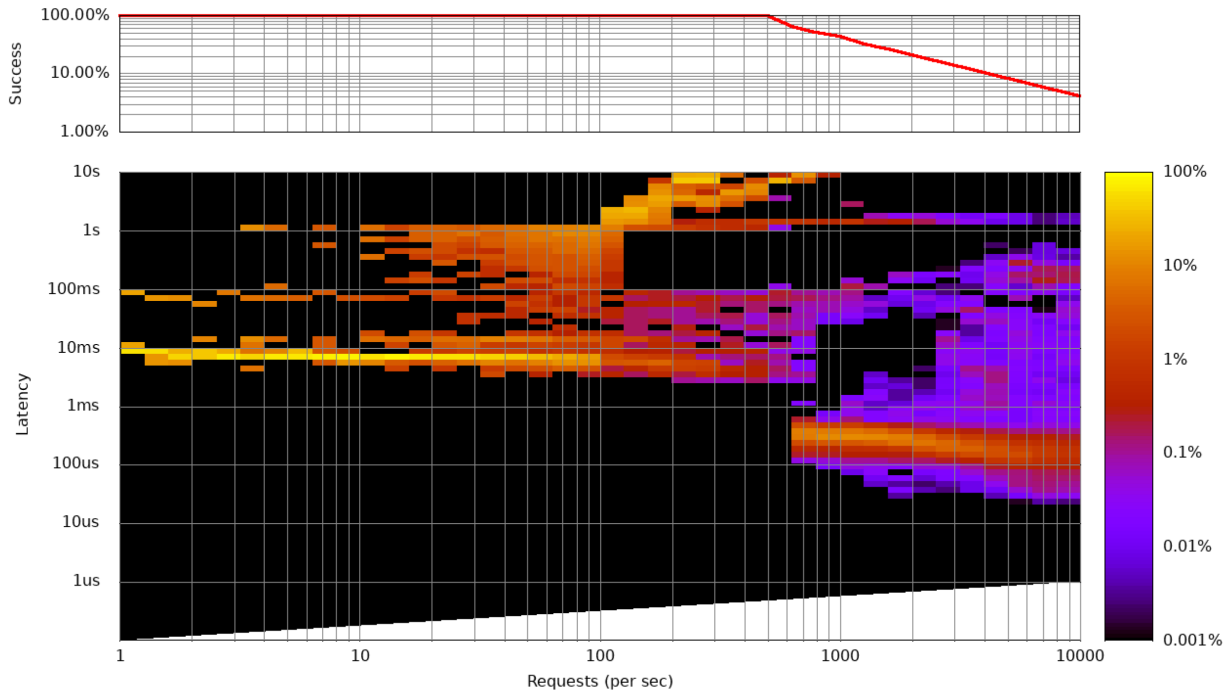
(B) Scalability profile - SMILES to Mol2D Conversion (6 heavy atoms)



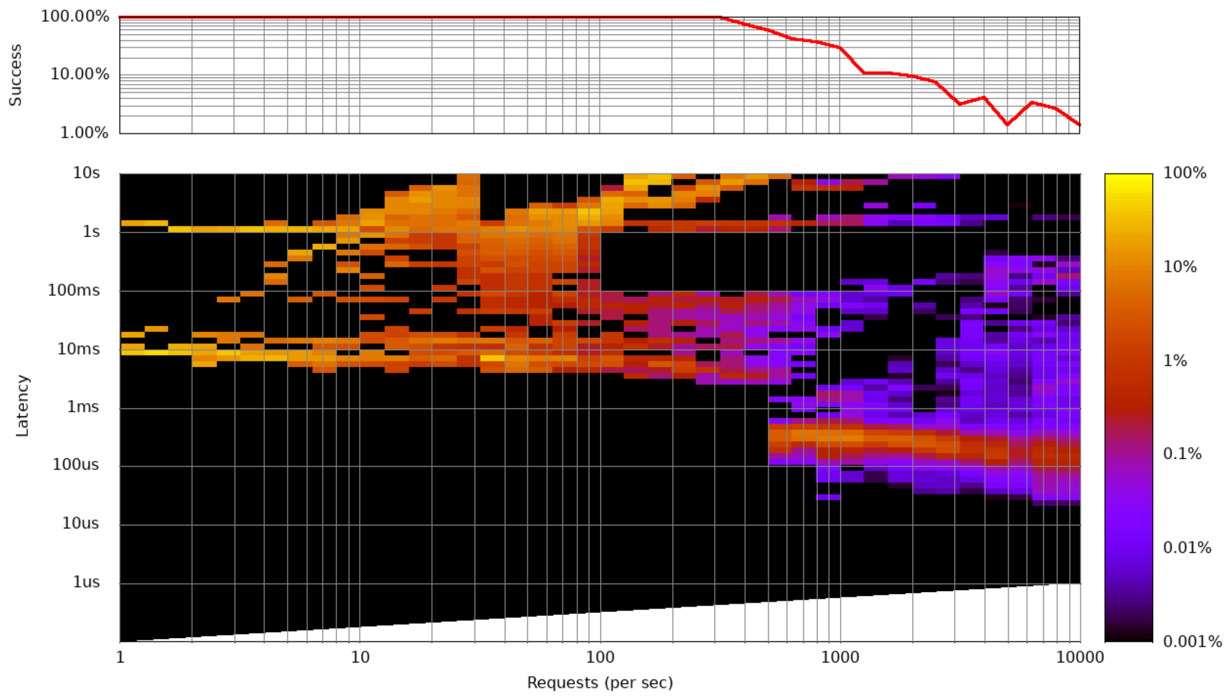
(C) Scalability profile - SMILES to Mol2D Conversion (11 heavy atoms)



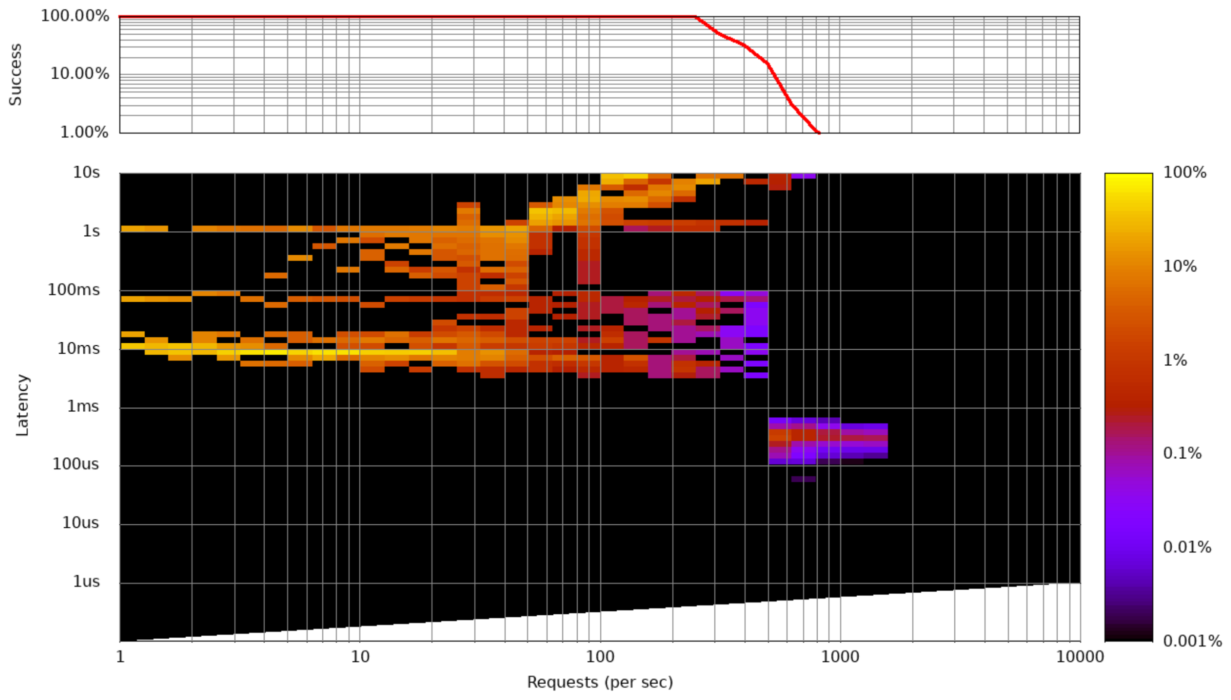
(D) Scalability profile - SMILES to Mol2D Conversion (13 heavy atoms)



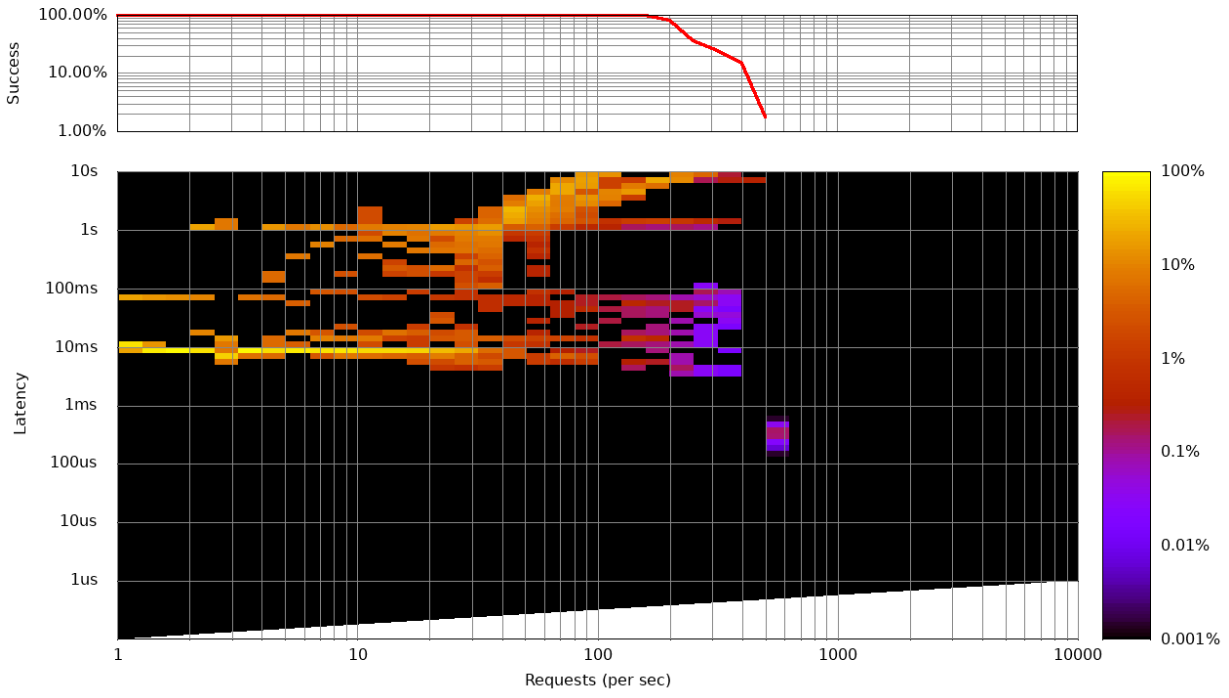
(E) Scalability profile - SMILES to Mol2D Conversion (20 heavy atoms)



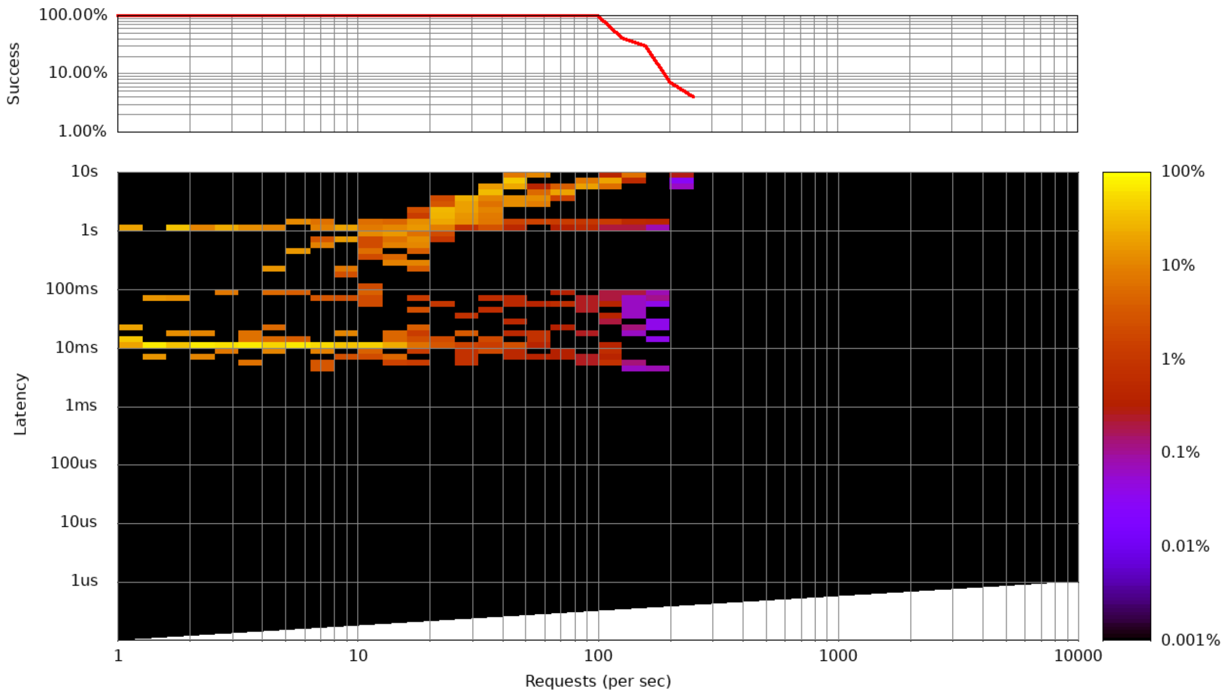
(F) Scalability profile - SMILES to Mol2D Conversion (24 heavy atoms)



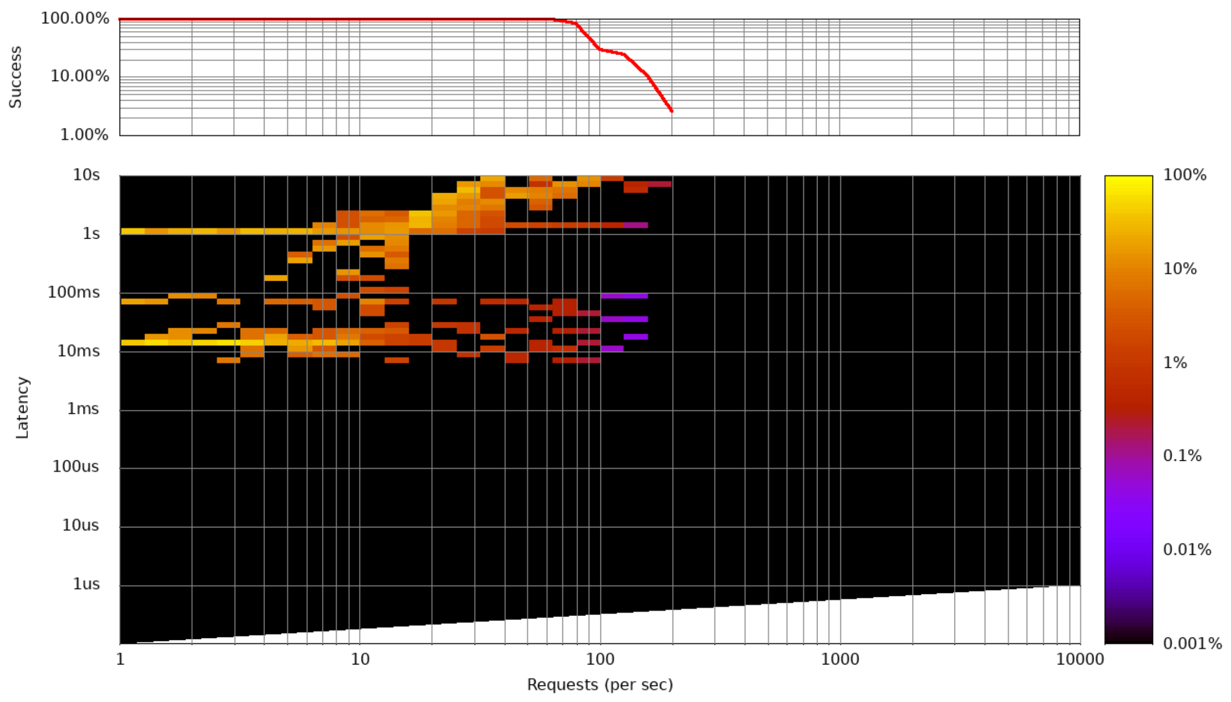
(G) Scalability profile - SMILES to Mol2D Conversion (32 heavy atoms)



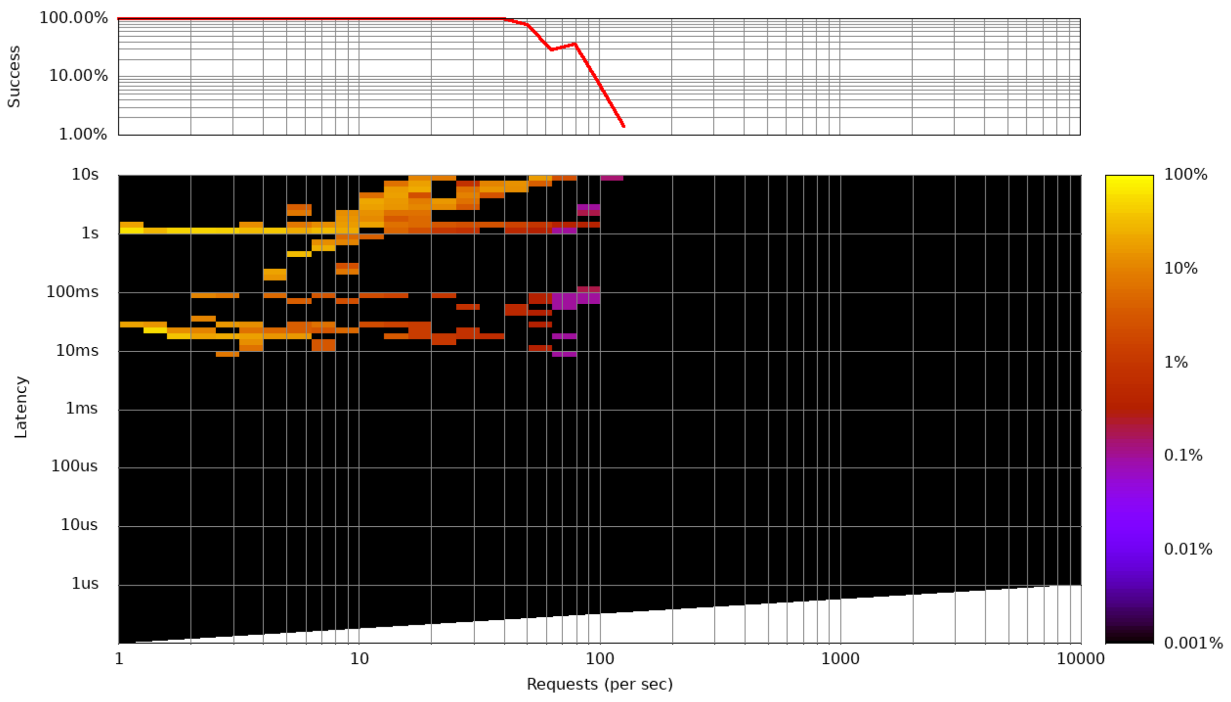
(H) Scalability profile - SMILES to Mol2D Conversion (54 heavy atoms)



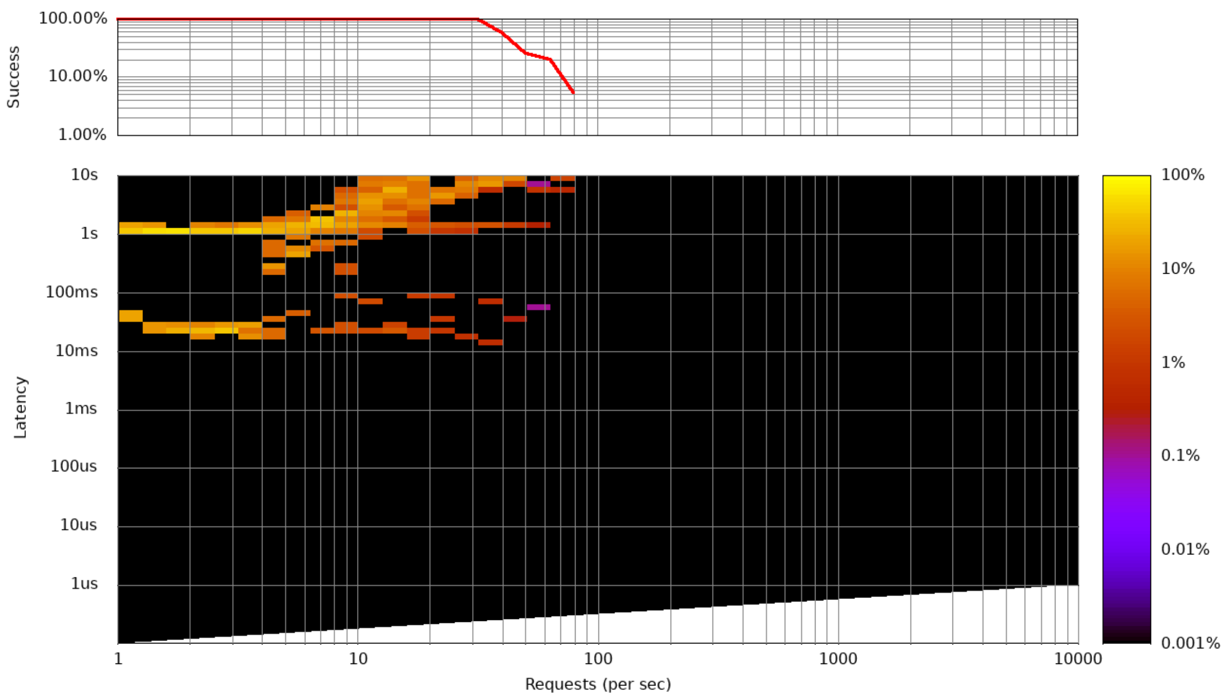
(I) Scalability profile - SMILES to Mol2D Conversion (74 heavy atoms)



(J) Scalability profile - SMILES to Mol2D Conversion (99 heavy atoms)



(K) Scalability profile - SMILES to Mol2D Conversion (125 heavy atoms)



100.00%
10.00%
1.00%
Success
10s
1s
100ms
10ms
1ms
100us
10us
1us
Latency
1
10
100
1000
10000
Requests (per sec)
100%
10%
1%
0.1%
0.01%
0.001%