Large Scale Docking in the Cloud

Benjamin Tingle and John J Irwin*

University of California San Francisco, Department of Pharmaceutical Chemistry, 1700 4th St, MC 2550, San Francisco CA 94158-2330

Corresponding author jji@cgl.ucsf.edu

Abstract

Molecular docking is a pragmatic approach to exploit protein structure for new ligand discovery, but the growing size of available chemical space is increasingly challenging to screen on in-house computer clusters. We have therefore developed AWS-DOCK, a protocol for running UCSF DOCK in the AWS cloud. Our approach leverages the low cost and scalability of cloud resources combined with a low-molecule-cost docking engine to screen billions of molecules efficiently. We benchmarked our system by screening 50 million HAC 22 molecules against the DRD4 receptor. We saw up to 3-fold variations in cost between AWS availability zones. Docking 4.5 billion lead-like molecules, a 7-week calculation on our 1000-core lab cluster, runs in less than a week in AWS for around $25,000, less than the cost of two new nodes. The cloud docking protocol is described in easy-to-follow steps and may be sufficiently general to be used for other docking programs. All the tools to enable AWS-DOCK are available free to everyone, while DOCK 3.8 is free for academic research.

Introduction

Molecular docking is a pragmatic approach to leverage structure for ligand discovery. The technique screens large libraries, rapidly scoring molecules for fit, ranking the database from best to worst for purchase and testing. Since 2015, the number of commercially accessible compounds has grown by over three orders of magnitude, leading to new opportunities to discover new chemistry for new biology. Ultra large scale docking has now been applied to over a dozen targets[1-11] and has discovered new compounds with activities often in the nM and occasionally in the sub-nM range[1, 3, 6, 12], often leading to molecules with interesting in vivo activities[3, 6, 12, 13]. A challenge to making this approach widely accessible has been the very size of the new libraries, where over 40 billion tangible molecules have been enumerated, and over 4.5 billion of these have structures calculated that are suitable for docking.

The cost of docking calculations for any target is roughly linear in the number of molecules docked, and is embarrassingly parallel, in that it can be spread across multiple processors without meaningful loss in efficiency. Still, whereas a screen of 138 million molecules against the DRD4 receptor[1] took less than 2 days on our 1000-core cluster, a screen of 4.5 billion molecules would take over two months; to finish in under two weeks would demand more resources than can often be mustered locally. As the database scales to 10 billion and beyond in the coming years, this capacity problem to obtain timely results will become only more pressing.

Amazon Web Services (AWS) is one of several cloud providers with the capacity to bring enormous resources to bear on docking screens, if it can be done economically and if the system can be mastered. Though in principle AWS, and related systems, are simply enormous fee-for-use clusters accessible to all (**Figure 1**), there are intricacies with their efficient use that are barriers to entry for the general user, and it was not certain to us that even used efficiently these off-site resources could compete with an on-site academic or industrial cluster or that a protocol suitable for occasional users and non-experts could be developed.

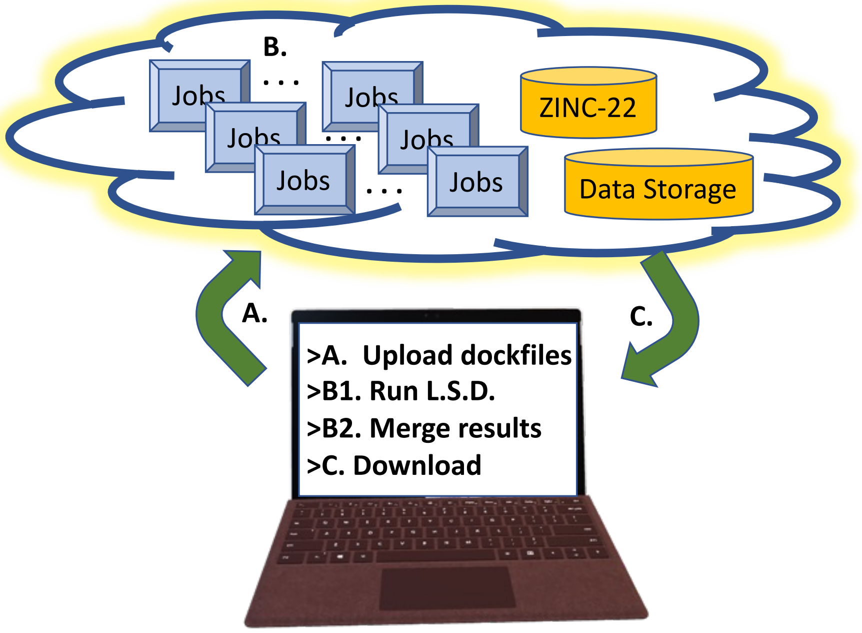


Figure 1. Docking in the Cloud. Though conceptually simple, cloud docking presents operational complexities that must be overcome. A. Upload receptor and software. The dockable database is pre-loaded. B. Large Scale Docking Jobs are run, potentially at immense scale, and the results merged to Top Hit Collections within cloud storage. C. The T.H.C. results are downloaded for prioritization and purchase.

Regarding cloud-based docking, there have been several studies that were encouraging. For instance, some groups have published docking pipelines suitable for running in the cloud [4, 7, 14-16]. Although available as open source, these generally lack a step-by-step protocol suitable for non-experts. Commercial solutions to docking in the cloud are also available, such as Orion (OpenEye, Santa Fe NM). This platform is currently made available at-cost to the academic community.

Here we report on a study that investigates efficient use of AWS for docking, with a step-by-step protocol for community use. The approach is tailored to our own software but is sufficiently general that it should be easily adaptable to any docking software and perhaps even other workflows.

Results

In principle, a cloud system like AWS provides almost unlimited resources for virtual screens that would otherwise be compute-bound, as we now find with ultra-large library docking. In practice, these calculations can be unaffordable versus in-house screens, unless optimized for cost and throughput. We investigated three variables that impact the speed and cost of a large library calculation: the availability zone where the calculation is run, the resource allocation strategy, and the bid percentage, i.e. how much we are prepared to pay to avoid temporarily surrendering cores to others willing to pay more for them. We examined the impact of each of these variables on the cost of a screen of a set of 50 million mid-sized (HAC 22), lead-like molecules against the DRDRD4 receptor. The results of the experiments are summarized below (**Table 1**).

Zone	**Cost (esd)**	**Variant**	**Observation**
us-east-2	$94(5)		Lowest cost
us-east-1	$186(6)		200%
us-west-1	$170(5)		181%
us-west-2	$220(7)		230%
eu-north-1	$152(5)		162%
sa-east-1	$280(10)		300%
us-east-2	$127(23)	Bid=20%	Worse than bid=100
us-east-2	$115(12)	Bid=50%	Worse than bid=100
us-east-2	$103(10)	Strategy=SCO	Not significant
us-east-2	$103(10)	Strategy=BFP	Not significant

Table 1. Cost to dock 50 million molecules, HAC 22, to the DRD4 receptor, standard protocol. (see **Methods**). Variant describes the variable that changes in addition to Zone where the job is run. Cost is USD, esd is the estimated standard deviation in the cost based on three runs.

We wondered whether docking in different availability zone regions would help with throughput, by adding more CPUs, and what the impact on total cost would be. AWS currently supports over 17 availability zone regions worldwide with more in development, many with multiple data centers. To consider the best one for docking, the prospective user should consider both the differential cost of the CPUs in different zones and the cost of data transport between regions. We began by considering the relative on-demand and spot prices of two representative instance types in six representative zones (**Table 2**). On-demand means a machine is reserved exclusively for our use until we surrender it, while spot means we are willing to be interrupted at any time, with a 2-minute warning. To support this use, DOCK 3.8 was modified to be interruptible and restartable to and be compatible with the AWS conventions for doing this. This data suggests that us-east-2 should be the cheapest zone to run spot calculations in, and that the next cheapest should be eu-north-1, perhaps somewhere between 20 and 40% more expensive than us-east-2.

Zone	**C5.18xlarge On-demand**	**C5.18xlarge Spot**	**C5.large On-demand**	**C5.large Spot**
us-east-1	306	117 (+44%)	8.5	3.54 (+82%)
us-east-2	306	81.2	8.5	1.95
us-west-1	382 (+25%)	111 (+36%)	10.6 (+25%)	3.08 (+60%)
us-west-2	306	116 (+43%)	8.5	3.26 (+67%)
eu-north-1	328 (+7%)	98.3 (+21%)	9.1 (+7%)	2.73 (+40%)
sa-east-1	472 (+54%)	179 (+220%)	13.1 (+54%)	4.7 (+241%)

Table 2. Cost of on-demand and spot instances on AWS for two representative compute-optimized instance types. US cents (¢) per hour. C5.18xlarge has 72 cores and C5.large has 2 cores.

We performed our standard docking calculation (DRD4 receptor, HAC 22, 50 million molecules) in six zones to test price sensitivity. Each experiment was run three times at randomly selected times over a period of two weeks to assess variability as the jobs compete with the background of all other jobs. Consistent with the spot pricing we were quoted, us-east-2 was clearly the cheapest to run docking calculations. Eu-north-1 came in second, costing a 50% premium, us-west-1 at a 70% premium, and us-east-1 at twice the cost of the baseline us-east-2 pricing. The most expensive zones, as predicted by the spot prices we selected, were sa-east-1 at 300% and us-west-2 at 230% of us-east-2. While the spot price estimates captured much of this, the estimates derived from quoted prices were optimistic in our hands, making us-east-2 even more competitive than we had predicted.

In dollar terms the numbers are sobering. Thus, whereas it costs 94 dollars to dock our 50 million molecule test database in our test conditions, implying a cost of $8,460 to dock 4.5 billion. Eu-north-1 and us-west-1 are tied for second place among the zones we tested with a cost of $160-$170, thus about 70% to 80% more than the cheapest us-east-2. Us-east-1 comes in fourth place and costs about twice as much as us-east-2, even though us-east-1 is where ZINC-22 is hosted in S3 and is where our S3 bucket of results resides. Us-west-2 comes in fifth at $220, thus about 2.3X the cost of running in us-east-2. Finally, sa-east-1, with $280 or about 3X the cost of us-east-2.

Data transport costs, while generally minor, could be surprising for some zones. Notably, transporting results back from sa-east-1 to us-east-1 where our S3 bucket lives cost about 15c per GB, for 50 million molecules about $20. Not a lot, $20, but at 4.5 billion scale that would be $1800 in data transfer costs alone. Other zones we tested are considerably cheaper to transport back from, about 1/10th as much, or $180 for a full screen. This cost could be reduced by merging results locally within the remote zone, and only transporting the final top 100,000 scoring molecules of each 50 million set back to us-east-1.

Spot instances use spare compute capacity that is available for less than the price of a fixed reservation. A lower ("spot") price is available, with the proviso that we may have to give up the CPU on short notice if another user is prepared to pay more. We figured that if we capped our bid at less than the recommended 100% of the CPU cost, we might save some money, but perhaps at the expense of being evicted more frequently, possibly incurring additional cost. We tested this tradeoff by docking using the following maximum bid levels: 100% (default), 50%, and 20%. Because of the random element of contention with other jobs in real time, we performed each run 5 times to obtain mean and standard deviation value estimates (esd). Surprisingly to us, we found (**Table 1**) that bidding less than 100% was almost always more expensive than bidding 100%. We saw a savings of 20% in a single run at 50% bid, but for most runs, lower maximum bids, whether of 20%, 50%, or even 90%, were generally 20-30% more expensive. This appears to be due to more frequent evictions, which add to the cost. As a result, bidding 100% (the default) was the best choice, on average, and is used in our protocol.

Another choice we explored is the resource allocation strategy. We wondered how it affected the price, and the speed of scheduling jobs. The three choices are BEST_FIT, BEST_FIT_PROGRESSIVE, and SPOT_CAPACITY_OPTIMIZED. These different options affect how machines are selected for running jobs, and use different strategies that explore the tradeoff between cost and throughput in response to current usage on the system[17]. Our experimental results (**Table 1**) failed to distinguish these three methods despite repeated attempts. We decided to use BEST_FIT_PROGRESSIVE as our default allocation strategy.

We were curious to compare the cost and speed of running in AWS with the use of our own departmental cluster (**Table 3**). By assuming some values, we could estimate costs of various idealized configurations. We considered three configurations as follows: a small cluster (128 cores in a single machine) corresponding to a small lab, a medium sized cluster, 10 units of 128 cores each, similar to our own lab cluster, and a bigger cluster, 40 units of 128 cores, similar in magnitude to our shared cluster at UCSF. We assumed 128 cores cost $15,000, a reasonable estimate given recent experience and current market conditions. It is easy to re-do the calculations with your own costs. We assumed the all-in cost to keep one server running is

2/3 of its cost per year, thus we budget \$10,000 annually for electricity, cooling, rack space, and system administration time. Within the university obtaining such values can be challenging. In each configuration we buy 1 PB of disk storage to hold 4.5 billion molecules of ZINC-22 in db2.tgz format with some additional storage for docking results. We assumed three scenarios using AWS: a) one large scale docking job per year, b) 10 LSD runs per year, and c) 38 LSD runs per year, corresponding to the largest cluster we considered.

We looked at the dollar cost per docking job, the wall clock time to execute a single large-scale screen, and the number of docking jobs that can be executed per year.

The time to execute a large-scale docking screen (4.5 billion molecules. 1s/mol) varied considerably between our various models. On a single 128-core machine, it takes more than a year to run. On ten times as many (1280 cores) it takes 41 days, almost six weeks. On 4 times as many again, 5120 cores, a full screen takes under 11 days. Compare these figures with AWS, where an average of 4000 cores gets a 4.5 billion molecule docking job run in 13 days.

The cost to execute a large-scale docking screen also varied. Thus, a single machine with 128 cores costs \$25.5K for a single screen. This is high partly because of the upfront cost of the disk required to hold ZINC-22. Increasing this cluster 10-fold so 1280 cores reduced the cost to only \$15.6K per large scale screen, albeit with a higher upfront investment of \$300K in year 1 (all in, including space and expert/sysadmin). Quadrupling this cluster to 5120 cores does not decrease the cost per job. On AWS, we are able to access about 4000 cores in varying price categories, ranging from about \$10,000 to over \$30,000. We have set a price of \$25K per 4.5 billion molecules, which allows running in several higher priced zones. Running in the most economical zones can reduce this cost by half or more, but at correspondingly slower throughput. In short, investing in a sufficiently larger in-house cluster, with its associated support costs, may often be more cost-efficient than AWS, but it does come with a substantial investment, while AWS may be spun up or down at will, on a job-by-job basis, without affecting per-job costs.

Item	Local 128 cores	Local 1280 cores	Local 5120 cores	AWS
Initial cost	$65K	$200K	$650K	$0
Annual cost	$10K	$100K	$400K	$0
Cost/dockjob	$0	$0	$0	$25K
Dockjobs/yr	0.9	9	36	28*
Days/dockjob	407	41	10.5	13

Config.	Year 1	Year 2	Years 1-5	#jobs/5yrs	Cost/dockjob
local 128	$75K	$10K	$115K	4.5	$25.6K
Local 1280	$300K	$100K	$700K	45	$15.6K
Local 5120	$1050K	$400K	$2650K	180	$15.6K
AWS 1/yr	$25K	$25K	$125K	5	$25K
AWS 10/yr	$250K	$250K	$1250K	50	$25K
AWS 28/yr	$700K	$700K	$3500K	140	$25K

Table 3. Comparison of the cost of running docking jobs in AWS vs on an in-house cluster.

Equipped with the results of these experimental tests to optimize our protocol, we compiled a step-by-step procedure in five stages to run jobs efficiently in AWS, as follows (**Box 1**):

Stage 1. Assessment of the cost and difficulty of setting up a cloud account for running large scale docking. Setting up an AWS account for the first time and preparing all the settings manually is a complex task with many steps, and numerous pitfalls. We investigated ways to simplify this process and solved it by writing a script and a protocol that reduces the hundreds of steps needed to just seven. As a result, it now takes only 10 minutes to set up an AWS account that is fully ready to run large scale docking. The step-by-step procedure is given in the **Supporting Information S1** and online[18]. Setting up an account requires setting up a payment method but incurs only minor charges. This step includes creating an S3 “bucket” to contain the project files and creating a docker image to run the calculations.

Stage 2. Assessment of the cost and difficulty of preparing a job for large scale docking. Preparing to run a large-scale docking calculation has three steps. The docking model must be prepared and parameterized, the part of chemical space to be screened must be selected, and the data must be uploaded to AWS. Docking model preparation can be done automatically using TLDR (tldr.docking.org, free but registration required), using the dockopt module on any of the DUDEZ targets (dudez2022.docking.org). Chemical space may be selected and downloaded using the 3D Tranche Browser in ZINC-22 (cartblanche22.docking.org). And all the relevant files can be uploaded to AWS using the aws command line tool. All of these steps are described in the **Supporting Information S2**.

Stage 3. Assessment of the cost and difficulty of running a large-scale docking screen. Once all of the required inputs are ready, launching a docking calculation is relatively simple. We have developed some recommended best practices that help to keep the calculation organized and allow late-stage decisions about how to organize the results. A script and a protocol describing this step is given in the **Supporting Information S3**. This step is where almost all of the cost of Large Scale Docking is incurred. Our protocol recommends starting with a fraction of the database to dock to test that the calculation is producing desired results. In this way, $50 can be spent early to produce final results, which can be checked before more funds are committed. If the results are not satisfactory, refinements may be made until the results are as sought. Then a full screen, costing perhaps tens of thousands of dollars, can be launched knowing that the calculation is working correctly *in situ*.

Stage 4. Assessment of the cost and difficulty of merging the final results and preparing the result for manual review and prioritization for purchase. When docking is complete, the user may combine the docking results to yield one or a few lists of top scoring ligands for clustering and review. (See **Supporting Information** S4.)

Stage 5. Assessment of the difficulty of cleaning up the AWS account when the calculation is complete. When the compounds have been ordered, the user may wish to reduce the disk space usages in AWS to reduce or avoid monthly charges. We propose several strategies, each with advantages and disadvantages. See **Supporting Information** S5.

A step-by-step protocol emerging from these studies is summarized in **Box 1**. While we exemplify use of AWS with DOCK 3.8, we believe that many of the steps described may be applied to other docking software and perhaps even software other than docking.

Box 1. Steps to run Large Scale Docking in the Cloud

Stage of Large-Scale Docking	Supporting Info	Online at https://wiki.docking.org/index.php/...
Set up Account	S1	AWS:Set_up_account
Upload files for docking	S2	AWS:Upload_files _for_docking
Submit docking job	S3	AWS:Submit_docking_job
Merge and download results	S4	AWS:Merge_and_download_results
Cleanup	S5	AWS:Cleanup

We have tested this protocol using the DUDEZ benchmark[19]. Ready-to-use inputs together with sample outputs are publicly accessible on our website, dudez2022.docking.org. Using these data, the user may test whether our protocol has been followed correctly.

Discussion

Three themes emerge from this work. First, large-scale docking at multi-billion molecule scale is now possible in the AWS cloud with UCSF DOCK 3.8—we suspect these procedures could be readily adapted to many docking programs. The calculation requires no commercial licenses for academics. Billions of commercially available molecules can be screened, typically for around twenty-five thousand dollars or less. No upfront investment in infrastructure and personnel to maintain it is required. Second, we have explored some of the cloud parameter choices for their effect on cost and speed of jobs. The best parameters we could find have been incorporated into our standard pipeline and best practices. Third new software tools and a detailed protocol are freely available to simplify all aspects of docking in the cloud, from configuring a cloud account, to prosecuting the screen, to merging and downloading the final results. We take up each of these points in turn.

Docking multiple billions of molecules is no longer limited to experts with large computer clusters and having expertise in cloud computing. Using the approach described here, any investigator can run large scale docking screens without assembling an expensive computer lab, without engaging costly consultants in cloud computing and without entangling alliances with commercial software vendors. Our approach does not use any graphical user interfaces, does not require any licenses other than to DOCK itself, and does not require licensing any third-party software not already included in DOCK. We focused on using tools that are native to AWS, adapting UCSF DOCK 3.8 to play within the constraints of that ecosystem. DOCK is now interruptable and restartable allowing the use of cheaper instances with spot pricing. Our protocol is intrinsically conservative, allowing a calculation to be tested at small scale at each step to avoid wasting money. There is scope for both improvisation and adaptation in our protocol because our approach is simple, modular and open.

There are numerous parameter choices to be made in the cloud that govern how the calculation is scheduled, where the data are stored, and how much the user is willing to pay to run jobs. We have examined three sets of choices empirically and have repeated the calculations to explore the variability of cost of using the shared, interruptible resource that the cloud is. By far the most important consideration is the availability zone, where we found variations of 300% in cost among zones. The best choices have been incorporated as defaults in our scripts and protocols. Surprisingly, writing data back to us-east-2 from eu-north-1 was both fast and economical, and storing the data in a single bucket in a single zone simplified the merging of the results at the end.

Our calculation of costs and times comparing different configurations of lab cluster with the cloud shows there are interesting tradeoffs to be had. It seems that buying and maintaining a home cluster of sufficient size (here, 1280 cores) is cheaper than renting (AWS), at least amortized over a 5-year period, if the machines are kept saturated (over the first year it is far more expensive, unamortized). An under-utilized on-premises cluster could easily become more expensive than AWS per job if not used efficiently. In short, investing in a sufficiently larger in-house cluster, with its associated support costs, may often be more cost-efficient than AWS, but it does come with a substantial investment, while AWS may be spun up or down at will, on a job-by-job basis, without affecting per-job costs. For occasional or uneven use, AWS provides tremendous flexibility to run jobs with no upfront cost, no long-term commitment, and reasonable cost.

Software tools and a protocol that use them have been developed and are freely available and ready to use. The software (other than DOCK itself) is available open source in github (see **Methods**). Our approach is not a black-box but a series of simplified steps, the results of which may be inspected and tested at each stage, and corrected if necessary. Parts of the protocol are highly tailored to AWS while others are general and could be used for other cloud providers. Although the scripts all use DOCK 3.8 as the docking engine, they could be easily adapted to other docking software and perhaps even other approaches. The software is written to be used either interactively or non-interactively, allowing for a range of experience by the target user. As much as possible, the protocol is outlined in both general, schematic terms as well as detailed step-by-step instructions. In this way, we hope the protocol will be useful for both beginners as well as for adaptation and improvisation by experts. We have provided test data for 40 DUDEZ systems with which our protocol can be used. We also provide some sample output data allowing users to test that they have followed our protocol correctly before embarking on their own research.

Our approach has limitations. Careful controls, when possible, including the DUDEZ benchmark, are an important way to assess a docking model and its likelihood for prospective screening success. Our automatic preparation of docking parameters currently requires actives and decoys to help optimize it, and even with these controls will not work well on every target. Some docking calculations can take substantially longer than others, based on the size of the binding site, degree of sampling, size of database screened, among other factors. This can result in widely varying costs of a large-scale docking calculation, including unexpectedly high costs. In this work we have estimated costs based on a throughput of 1 second per library molecule, which is typical for DOCK3.8, but can certainly vary. We recommend small-scale calculations to estimate total cost before attempting a run at scale. Suboptimal parameter choices can easily double the length (and cost!) of the calculation. Our protocol has been tested in six availability zones and should work in any AWS zone, but we have not been able to test it everywhere yet.

Notwithstanding these caveats, a new protocol for docking at scale in the cloud is now available. The step-by-step, detailed protocol requires no specialist expertise and in particular no prior experience with AWS. Their adoption should lower barriers to entry to large library docking in the community, bringing the large chemical space they represent to ever more interesting biological targets.

Acknowledgements

Supported by NIH GM133836 (to JJI). We thank OpenEye Scientific Software for a license to Omega and OEChem. We thank Chemaxon for a license to Jchem, and thank B. Shoichet for reading this manuscript.

The ORICID of J.J.I. is 0000-0002-1195-6417.

Conflict of Interest Statement

J.J.I. is a co-founder of Blue Dolphin Lead Discovery LLC, a contract research organization for molecular docking screens. He is also a co-founder of Deep Apple Therapuetics, Inc.

Software availability statement

All software for using DOCK in the cloud is freely available via github.com/docking-org/awsdock-scripts.

Methods

Scripts are written in Python 3.8. We used the docking protocol previously outlined[20] and our benchmark used the successful docking parameters developed in an earlier study for the DRD4 receptor[1]. We use the DockBlaster2.0 software for automated receptor preparation. All software for using DOCK in the cloud is freely available via github.com/docking-org/awsdock-scripts. This software may be forked.

Aws-setup tool: The aws-setup tool can be generalized to configure all sorts of cluster-based computation environments in the cloud. Our aws-setup container image by default comes with configuration for creating a docking environment, as well as a script to submit jobs to that environment (similar to our traditional SLURM submission script, subdock.bash).

License UCSF DOCK 3.8. We used UCSF DOCK 3.8.0 for all docking in this work. DOCK is free to academics at dock.compbio.ucsf.edu. For for-profit outfits, DOCK 3.8 is modestly priced (write to dock_industry@googlegroups.com for more information).

Select a database subset. We used ZINC-22 [21] as of September 2022 (4.5 billion lead-like molecules). To select a database subset, browse to cartblanche22.docking.org and select Tranches>3D. Then use the selection tool to select the desired region of chemical space. When you are ready, download the selection for db2.tgz format and select for the AWS platform.

Obtain sample data for DUDEZ for docking. Browse to dudez2022.docking.org and download either the tarball for each target, or all targets together.

Prepare docking parameters. Browse to tldr.docking.org/start/dockopt and upload the target information. Click “Go” and wait about an hour to be informed that the receptor preparation job is complete and that the results are ready for download. Sample data are available at dudez2022.docking.org.

Benchmarking calculations. We docked a database of around 50 million commercially available small molecules with heavy atom count of 22 for benchmarking. The actual molecules docked may be found at dudez2022.docking.org. The docking model used for the benchmarking was taken exactly from our previous work on dopmanine DRD4 [1] and is available for reference at dudez2022.docking.org.

AWS Pricing. The current spot pricing in each zone can viewed at https://aws.amazon.com/ec2/spot/pricing/ and compared with the full price of a reservation here https://aws.amazon.com/ec2/pricing/on-demand/. While these enable an estimate of the upper and lower bounds of a calculation, we found the actual price varied a lot, often unpredictably (**Table 2**).

Literature Cited

1. Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K.; Tolmachev, A. A.; Shoichet, B. K.; Roth, B. L.; Irwin, J. J., Ultra-Large Library Docking for Discovering New Chemotypes. *Nature* **2019**, 566, 224-229.

2. Gentile, F.; Yaacoub, J. C.; Gleave, J.; Fernandez, M.; Ton, A. T.; Ban, F.; Stern, A.; Cherkasov, A., Artificial Intelligence-Enabled Virtual Screening of Ultra-Large Chemical Libraries with Deep Docking. *Nat Protoc* **2022**, 17, 672-697.

3. Stein, R. M.; Kang, H. J.; McCorvy, J. D.; Glatfelter, G. C.; Jones, A. J.; Che, T.; Slocum, S.; Huang, X. P.; Savych, O.; Moroz, Y. S.; Stauch, B.; Johansson, L. C.; Cherezov, V.; Kenakin, T.; Irwin, J. J.; Shoichet, B. K.; Roth, B. L.; Dubocovich, M. L., Virtual Discovery of Melatonin Receptor Ligands to Modulate Circadian Rhythms. *Nature* **2020**, 579, 609-614.

4. Gorgulla, C.; Boeszoermenyi, A.; Wang, Z. F.; Fischer, P. D.; Coote, P. W.; Padmanabha Das, K. M.; Malets, Y. S.; Radchenko, D. S.; Moroz, Y. S.; Scott, D. A.; Fackeldey, K.; Hoffmann, M.; Iavniuk, I.; Wagner, G.; Arthanari, H., An Open-Source Drug Discovery Platform Enables Ultra-Large Virtual Screens. *Nature* **2020**, 580, 663-668.

5. Sadybekov, A. A.; Brouillette, R. L.; Marin, E.; Sadybekov, A. V.; Luginina, A.; Gusach, A.; Mishin, A.; Besserer-Offroy, E.; Longpre, J. M.; Borshchevskiy, V.; Cherezov, V.; Sarret, P.; Katritch, V., Structure-Based Virtual Screening of Ultra-Large Library Yields Potent Antagonists for a Lipid Gpcr. *Biomolecules* **2020**, 10.

6. Alon, A.; Lyu, J.; Braz, J. M.; Tummino, T. A.; Craik, V.; O'Meara, M. J.; Webb, C. M.; Radchenko, D. S.; Moroz, Y. S.; Huang, X. P.; Liu, Y.; Roth, B. L.; Irwin, J. J.; Basbaum, A. I.; Shoichet, B. K.; Kruse, A. C., Structures of the Sigma2 Receptor Enable Docking for Bioactive Ligand Discovery. *Nature* **2021**, 600, 759-764.

7. Lu, H.; Wei, Z.; Wang, C.; Guo, J.; Zhou, Y.; Wang, Z.; Liu, H., Redesigning Vina@Qnlm for Ultra-Large-Scale Molecular Docking and Screening on a Sunway Supercomputer. *Front Chem* **2021**, 9, 750325.

8. Gorgulla, C.; Cinaroglu, S. S.; Fischer, P. D.; Fackeldey, K.; Wagner, G.; Arthanari, H., Virtualflow Ants-Ultra-Large Virtual Screenings with Artificial Intelligence Driven Docking Algorithm Based on Ant Colony Optimization. *Int J Mol Sci* **2021**, 22.

9. Gorgulla, C.; Padmanabha Das, K. M.; Leigh, K. E.; Cespugli, M.; Fischer, P. D.; Wang, Z. F.; Tesseyre, G.; Pandita, S.; Shnapir, A.; Calderaio, A.; Gechev, M.; Rose, A.; Lewis, N.; Hutcheson, C.; Yaffe, E.; Luxenburg, R.; Herce, H. D.; Durmaz, V.; Halazonetis, T. D.; Fackeldey, K.; Patten, J. J.; Chuprina, A.; Dziuba, I.; Plekhova, A.; Moroz, Y.; Radchenko, D.; Tarkhanova, O.; Yavnyuk, I.; Gruber, C.; Yust, R.; Payne, D.; Naar, A. M.; Namchuk, M. N.; Davey, R. A.; Wagner, G.; Kinney, J.; Arthanari, H., A Multi-Pronged Approach Targeting Sars-Cov-2 Proteins Using Ultra-Large Virtual Screening. *iScience* **2021**, 24, 102021.

10. Sadybekov, A. A.; Sadybekov, A. V.; Liu, Y.; Iliopoulos-Tsoutsouvas, C.; Huang, X. P.; Pickett, J.; Houser, B.; Patel, N.; Tran, N. K.; Tong, F.; Zvonok, N.; Jain, M. K.; Savych, O.; Radchenko, D. S.; Nikas, S. P.; Petasis, N. A.; Moroz, Y. S.; Roth, B. L.; Makriyannis, A.; Katritch, V., Synthon-Based Ligand Discovery in Virtual Libraries of over 11 Billion Compounds. *Nature* **2022**, 601, 452-459.

11. Fresnais, L.; Ballester, P. J., The Impact of Compound Library Size on the Performance of Scoring Functions for Structure-Based Virtual Screening. *Briefings in bioinformatics* **2021**, 22.

12. Fink, E. A.; Xu, J.; Hubner, H.; Braz, J. M.; Seemann, P.; Avet, C.; Craik, V.; Weikert, D.; Schmidt, M. F.; Webb, C. M.; Tolmachova, N. A.; Moroz, Y. S.; Huang, X. P.; Kalyanaraman, C.; Gahbauer, S.; Chen, G.; Liu, Z.; Jacobson, M. P.; Irwin, J. J.; Bouvier, M.; Du, Y.; Shoichet, B. K.; Basbaum, A. I.; Gmeiner, P., Structure-Based Discovery of Nonopioid Analgesics Acting through the Alpha(2a)-Adrenergic Receptor. *Science* **2022**, 377, eabn7065.

13. Kaplan, A. L.; Confair, D. N.; Kim, K.; Barros-Alvarez, X.; Rodriguiz, R. M.; Yang, Y.; Kweon, O. S.; Che, T.; McCorvy, J. D.; Kamber, D. N.; Phelan, J. P.; Martins, L. C.; Pogorelov, V. M.; DiBerto, J. F.; Slocum, S. T.; Huang, X. P.; Kumar, J. M.; Robertson, M. J.; Panova, O.; Seven, A. B.; Wetsel, A. Q.; Wetsel, W. C.; Irwin, J. J.; Skiniotis, G.; Shoichet, B. K.; Roth, B. L.;

Ellman, J. A., Bespoke Library Docking for 5-Ht(2a) Receptor Agonists with Antidepressant Activity. *Nature* **2022**, 610, 582-591.

14. Venkatraman, V.; Colligan, T. H.; Lesica, G. T.; Olson, D. R.; Gaiser, J.; Copeland, C. J.; Wheeler, T. J.; Roy, A., Drugsniffer: An Open Source Workflow for Virtually Screening Billions of Molecules for Binding Affinity to Protein Targets. *Front Pharmacol* **2022**, 13, 874746.

15. Ochoa, R.; Palacio-Rodriguez, K.; Clemente, C. M.; Adler, N. S., Dockecr: Open Consensus Docking and Ranking Protocol for Virtual Screening of Small Molecules. *J Mol Graph Model* **2021**, 109, 108023.

16. Labbe, C. M.; Rey, J.; Lagorce, D.; Vavrusa, M.; Becot, J.; Sperandio, O.; Villoutreix, B. O.; Tuffery, P.; Miteva, M. A., Mtiopenscreen: A Web Server for Structure-Based Virtual Screening. *Nucleic Acids Res* **2015**, 43, W448-54.

17. AWS Aws Allocation Strategies. https://docs.aws.amazon.com/batch/latest/userguide/allocation-strategies.html

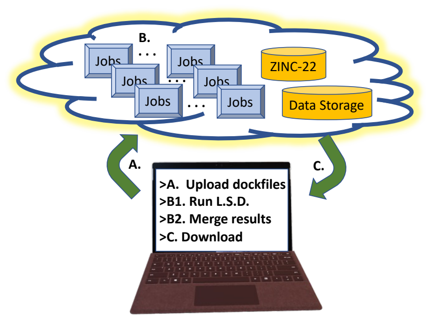
18. Aws Dock Enviroment Setup. https://wiki.docking.org/index.php/AWS:Set_up_account

19. Stein, R. M.; Yang, Y.; Balius, T. E.; O'Meara, M. J.; Lyu, J.; Young, J.; Tang, K.; Shoichet, B. K.; Irwin, J. J., Property-Unmatched Decoys in Docking Benchmarks. *J. Chem. Inf. Model* **2021**, 61, 699-714.

20. Bender, B. J.; Gahbauer, S.; Luttens, A.; Lyu, J.; Webb, C. M.; Stein, R. M.; Fink, E. A.; Balius, T. E.; Carlsson, J.; Irwin, J. J.; Shoichet, B. K., A Practical Guide to Large-Scale Docking. *Nat Protoc* **2021**, 16, 4799-4832.

21. Tingle, B.; Tang, K.; Castanon, J.; Gutierrez, J.; Khurelbaatar, M.; Dandarchuluun, C.; Moroz, Y. S.; Irwin, J. J., Zinc-22 - a Free Multi-Billion-Scale Database of Tangible Compounds for Ligand Discovery. . *ChemRxiv* **2022**.

TOC Figure



```
>A.  Upload dockfiles
>B1. Run L.S.D.
>B2. Merge results
>C. Download
```