Journal of
CHEMICAL INFORMATION
AND MODELING

Application Note

pubs.acs.org/jcim

AceCloud: Molecular Dynamics Simulations in the Cloud

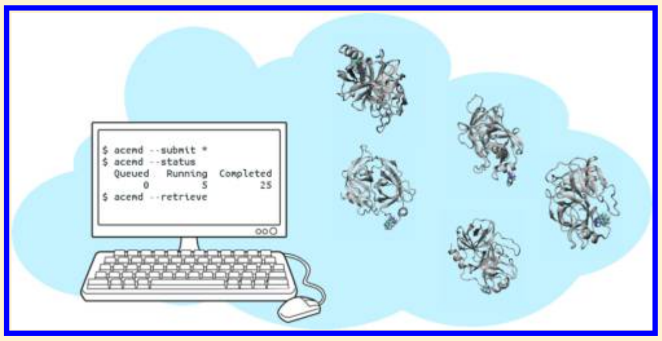
M. J. Harvey*,[†] and G. De Fabritiis*,[‡],[§]

[†]Acellera, Barcelona Biomedical Research Park (PRBB), C/Dr. Aiguader 88, 08003 Barcelona, Spain

[‡]Computational Biophysics Laboratory (GRIB-IMIM), Universitat Pompeu Fabra, Barcelona Biomedical Research Park (PRBB), C/Dr. Aiguader 88, 08003 Barcelona, Spain

[§]Institució Catalana de Recerca i Estudis Avançats, Passeig Lluis Companys 23, 08010 Barcelona, Spain

ABSTRACT: We present AceCloud, an on-demand service for molecular dynamics simulations. AceCloud is designed to facilitate the secure execution of large ensembles of simulations on an external cloud computing service (currently Amazon Web Services). The AceCloud client, integrated into the ACEMD molecular dynamics package, provides an easy-to-use interface that abstracts all aspects of interaction with the cloud services. This gives the user the experience that all simulations are running on their local machine, minimizing the learning curve typically associated with the transition to using high performance computing services.



```
$ acemd --submit *
$ acemd --status
  Queued  Running  Completed
       0        5         25
$ acemd --retrieve
```

INTRODUCTION

In recent years classical molecular dynamics simulation has begun to prove an increasingly powerful complement to experiment, as the time scales accessible through simulation approach biologically relevant time scales.

These developments are underpinned by progress in both simulation technology and methodology. In particular, the exploitation of specialized accelerator processors (known as GPUs, belying their origin in the graphics processing field) has proven transformative for the field. A modern GPU-based molecular dynamics (MD) code—such as ACEMD,[1] GROMACS,[2] or Amber[3]—is able to perform at simulation rates on a single low-cost GPU that may otherwise require several nodes of a dedicated high performance computer (HPC) cluster. This effective decoupling of MD from the established HPC space has led to substantial (over an order of magnitude) reduction in cost per unit of MD sampling.

This marked reduction in cost of MD sampling, yet without substantially increasing the time scales accessible within any one simulation, has prompted the development of methods based upon the analysis of ensembles of large numbers of independent simulations. Markov modeling techniques, for example, have been demonstrated to be able to recover equilibrium properties and kinetic processes that occur over time scales far longer than that of any one simulation.[4] Such studies are computationally expensive, and special-purpose infrastructures, for example GPUGRID[5] and Folding@Home,[6] have been constructed to support these activities by exploiting GPU computing resources provided by volunteers.

Recourse to unconventional sources of computing resources has been necessary because, despite GPUs now being well-established as a significant part of the contemporary HPC landscape, they have limited penetration in established HPC facilities.[7] In large part, this is attributable to the need for software to be substantially refactored in order to run on a GPU. On a mixed-workload system where only a faction of capacity is consumed by GPU-capable software, or in the absence of funding for software development, for example, it can be difficult to justify the cost of deploying GPUs. This has important implications for the broader uptake of GPU-dependent ensemble-MD-based analytical methods: evaluation of these techniques is limited to groups that already access substantial GPU resources.

To address this problem we have developed AceCloud, a service that provides online, on-demand access to computing resources for molecular dynamics simulation. AceCloud takes advantage of the broader computing industry trend of "cloud" computing[8]—effectively the flexible rental of computing resources—to lower the barrier to entry for performing large-scale molecular dynamics simulations. The capital setup costs of establishing in-house GPU resources are negated and replaced by ongoing operational costs that are proportional to usage.

AceCloud natively supports the popular molecular dynamics codes ACEMD and GROMACS, as well as permitting the easy execution of other third-party software, such as Amber or NAMD. The current implementation of AceCloud targets the Amazon Web Services (AWS)[9] Cloud computing platform.

To minimize the learning curve associated with cloud computing, AceCloud is accessed through simple extensions to the ACEMD molecular dynamics code. This interface abstracts all interaction with the supporting cloud infrastructure, and emulates, as far as is practicable, the experience of running simulations locally on one's own machine. This focus on ease-of-use minimizes the need for the user to change

Received: February 15, 2015
Published: April 7, 2015

Table 1. AceCloud Client Command-Line Arguments

argument	description
`--command [command]`	Give syntax and description of an ACEMD input file command.
`--device [devlist] [inputfile]`	Run a simulation on the specified comma-separated list of GPUs. If no input file name is specified, `input` is assumed.
`--submit [[ --project prj-name] directory...]`	Run jobs on Amazon EC2. Submit each directory to Amazon as a separate simulation. Each simulation is allocated to the project *prj-name* that precedes it in the argument list. If no directory is specified, the current working directory will be used.
`--status [ prj-name...]`	Return the status of all jobs in project summarized by project, or all jobs in the project if `--long` is specified.
`--retrieve [prj-name...] [-loop] [-incremental]`	Retrieve the results of all completed simulations, or only those in the specified projects. Results are placed in the originating directories. Make a single pass then exits unless `--loop` is specified. Add `--incremental` to retrieve partial results from running simulations.
`--delete [--project prj-name] [dir-list]`	Delete simulations and pending output of the specified project. If *dir-list* is specified only those named simulations will be deleted. If no *dir-list* is specified, everything under the project *prj-name* will be deleted.
`--spots`	Show current Amazon Spot pricing.
`--configure`	Set Amazon configuration options. `--configurehelp` for info.
`--verbose`	Add as a qualifier to the above to produce verbose output.
`--xml`	Add as a qualifier to the above to produce structured output.

or adapt existing working practices, a burden often encountered when transitioning from personal desktop to HPC systems.

In this manuscript we present the design and implementation of AceCloud along with examples of use.

DESIGN AND IMPLEMENTATION

AceCloud is designed to execute and manage large ensembles of molecular dynamics simulations on third party cloud resources. The user experience is intended to closely replicate that of command-line program execution, without exposing the details of interaction with the cloud service. All cloud operations (data transfer, virtual machine instantiation, etc.) are mediated by the client, giving the appearance of completely local execution.

Currently, AceCloud can target Amazon Web Services. AWS is presently (as of January 2015) the only major cloud provider that makes available GPU-based computing resources.

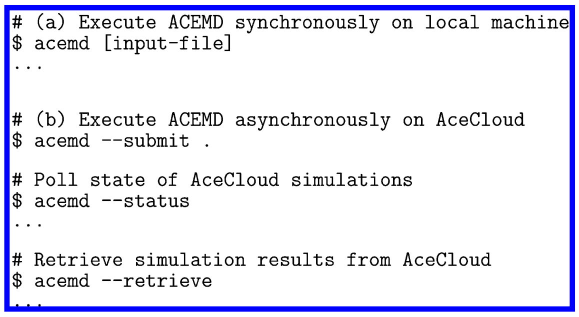
The following sections detail the user client and user experience and give an overview of the AWS features used by AceCloud and the implementation of the cloud-side computing service.

Client Software and User Experience. The AceCloud client is implemented as an extension to the ACEMD molecular dynamics simulation code. ACEMD has a command-line interface, with new AceCloud functions being accessed via a new set of command-line switches (Table 1). The default behavior of ACEMD is to perform a molecular dynamics simulation on the machine on which it is being run, starting from appropriate input files found in the current working directory. The addition of the command line argument `--submit` is sufficient to cause ACEMD to instead submit the job for execution on AceCloud (Figure 1 shows an example).

Whereas local execution is synchronous, with control not returning to the user until the simulation is completed, AceCloud jobs are asynchronous. Control is immediately returned to the user after submission, and additional commands are used to monitor job status and retrieve results.

When a job is submitted to AceCloud, the whole contents of the directory specified in the submission (omitting any large XTC or DCD trajectory files) are transferred to the cloud and form the basis for the input for an AceCloud job.

AceCloud jobs are, by default, named after the directory in which their input resided. Jobs can be further grouped into



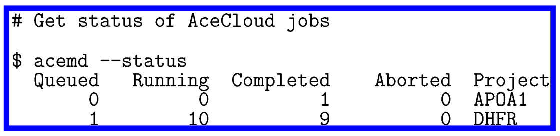
```
# (a) Execute ACEMD synchronously on local machine
$ acemd [input-file]
...

# (b) Execute ACEMD asynchronously on AceCloud
$ acemd --submit .

# Poll state of AceCloud simulations
$ acemd --status
...

# Retrieve simulation results from AceCloud
$ acemd --retrieve
...
```

Figure 1. Comparison of the commands required to run ACEMD (a) locally on GPUs directly attached to the machine in use and (b) remotely on AceCloud. In the former, ACEMD executes synchronously, running to completion before control is returned to the user. When running via AceCloud, in contrast, execution is asynchronous: control is returned to the user immediately, and the progress of the run must be monitored.

projects using an arbitrary user-supplied identifier. Grouping into projects allows collective operations to be easily performed—for example bulk deletion or retrieval of results. The {*project*, *directory*} tuple is constrained to be unique for all extant AceCloud jobs.

The status of all AceCloud jobs can be seen by using the `--status` option; this gives a summary by project of the number of jobs pending, running and completed, analogously to a `qsub` listing within a conventional cluster environment. An example of an AceCloud status view is given in Figure 2.

Because AceCloud jobs run asynchronously, results of jobs must be retrieved by pulling them back with the `--retrieve` option. The default behavior of this option is to place the results for finished simulations back to the directories from which they were originally created. Results of finished simulations are stored by AceCloud (in S3) until the user



```
# Get status of AceCloud jobs

$ acemd --status
  Queued   Running   Completed    Aborted  Project
       0         0           1          0  APOA1
       1        10           9          0  DHFR
```

Figure 2. Example of querying the status of AceCloud jobs. In this instance, 21 jobs are extant, grouped into 2 projects, APOA1 and DHFR. The DHFR project contains 20 jobs, 10 in progress, 9 completed, and 1 pended to run.

performs a retrieval, and they are automatically deleted afterward.

For long-running simulations, partial results can be obtained using the additional argument `--incremental`. This instructs the client to use SSH and `rsync` to synchronize the local copy of the output with the current progress of the simulation. `rsync` is used to ensure that successive retrievals only copy new or changed data, thus minimizing network bandwidth usage.

Running a retrieval as a nonterminating background process, with the options `--incremental`, provides the user experience most similar to having the simulations running locally. Output appears incrementally in the job directories soon after its creation, without any further user interaction with AceCloud being necessary.

AceCloud has built-in support for ACEMD and GROMACS simulations and support for running arbitrary third-party code (for example Amber or NAMD). When instantiated, the AceCloud VM will determine whether to run ACEMD or GROMACS based on the presence of specific files (`input` for ACEMD, `topol.tpr` for GROMACS). If neither of these are present, a file called `run.sh` will be executed. This provides a mechanism by which the user may run arbitrary code on an AceCloud instance.

The MD codes Amber and NAMD, for example, are not included within AceCloud because of license restrictions. A user in possession of a suitable license may run their own copy by placing the relevant executable, supporting libraries and simulation input files in a directory, along with a `run.sh` crafted to execute it. An example of running Amber this way is shown in Figure 3.

```
# Contents of submission directory
# Including AMBER executable, libraries
# And simulation input

$ ls .
libcurand.so.4
md12.x
mdin
pmemd.cuda
prmtop
run.sh

# Contents of run.sh for running AMBER

$ cat run.sh

#!/usr/bin/bash
# Set environment to find library files
# submitted as part of the job
export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH

# Execute AMBER
./pmemd.cuda -O -p prmtop -c md12.x  -i mdin

# End
```

Figure 3. Example of running an arbitrary application via AceCloud, in this case the `pmemd` program from Amber.[3] An input directory is prepared containing the `pmemd` binary itself, along with a supporting library, the input files for the simulation itself, and a `run.sh` script written to run the program.

AceCloud on Amazon Web Services. AceCloud takes advantage of four major components of Amazon Web Services: the *Elastic Compute Cloud (EC2)*, *Simple Storage Service (S3)*, *Virtual Private Cloud (VPC)*, and *Identity and Access Management (IAM)*.

EC2 provides an *Infrastructure-as-a-Service (IaaS)* presentation of compute resources. The resource is provided as an instantiation of a virtualized machine (VM) running atop physical hardware, within which the user is free to deploy an arbitrary operating system image (known as an Amazon Machine Instance, or AMI).

EC2 provides numerous classes of VM, each representing a different set of resources (e.g., number of CPU cores, amount of memory and storage, network connectivity) optimized for particular use cases. Interaction with EC2 for provisioning, monitoring, and control of VMs is accomplished manually via an interactive Web console[10] or programatically via a web services RESTful API.[11,12]

EC2 instances are provisionable on demand and most frequently billed at fixed hourly rates. Of particular note is the availability of *spot pricing*. Instances obtained at spot prices are charged at a variable rate that ostensibly[13] reflects underlying demand. If the spot price rises above the limit set by the procurer when requesting an instance, those instances are liable to be terminated and the resources reallocated to an entity prepared to pay the higher rate. If the user of the instance is able to tolerate this potential inconvenience, spot pricing can be substantially more cost-effective than standard on-demand pricing.

S3 is an object storage system accessible via a web services interface. Items of data stored within S3 are referenced by a unique key and are treated as read-only, atomic objects. Objects may be grouped into *buckets*, analogous to file system directories, which exist in a namespace global to all AWS users. S3 provides an access control mechanism enabling buckets and objects to be publicly read—writable or limited in access to appropriately authenticated entities. Where objects are subject to access control, *presigned URLs* may be created. These contain a cryptographic signature that allow limited read or write access in the absence of credentials.

Although S3 is substantially different to a conventional storage system,[14] it nevertheless provides (in the case of AceCloud) a useful storage area for simulation state that is persistent and independent of the EC2 instance life-cycle.

AceCloud Compute Instances. Within the AceCloud design each user-submitted job is allocated its own instance of the AMI that persists only for the duration of the job. Notably, there is no AceCloud server process: the client interacts only with AWS web service end points and instances.

When the AMI is instantiated it boots a Linux operating system image (derived from Amazon Linux) and immediately downloads the job input from S3 via the URL provided in instance metadata and executes the requisite program. Once the run is complete any output generated is stored in S3 via the specified URL and the instance shuts down and is terminated.

Although there is no runtime limit imposed on the instance, there is a limit of 1 GB for the total size of output data, imposed by the size of the underlying filesystem and object size limits in S3.

The EC2 Instance type currently used by AceCloud is `g2.2xlarge`, which provides eight threads of an Intel E5-2670 Xeon CPU, 15 GB of RAM, and a single Nvidia K520 GPU, based on the compute capability 3.0 GK104 silicon, with 1536 CUDA cores at 800 MHz. Performance data is given below.

Security. Ensuring data security is an important consideration when using external third-party computing resources or transferring data over public networks. AceCloud is designed with a minimal vulnerability surface: VM instances are configured to expose SSH as the only network-visible service

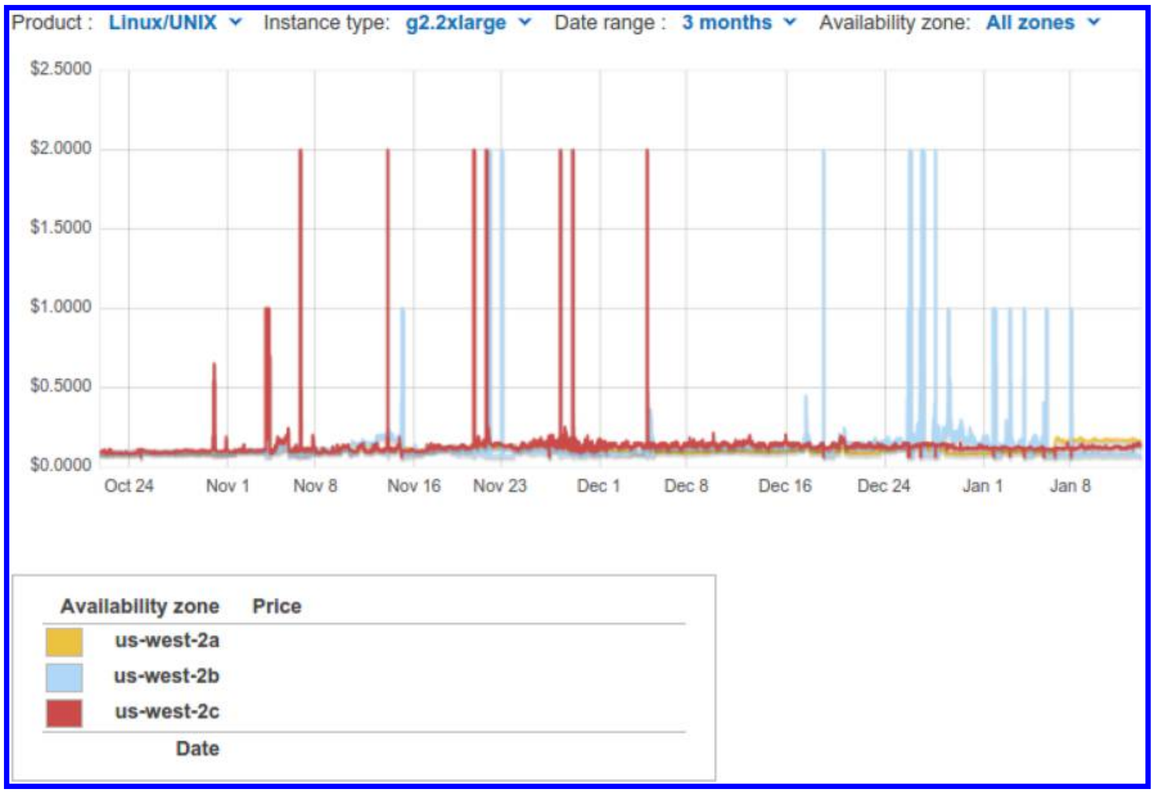


Figure 4. Historic spot pricing over a 3 month period for one AWS region. Pricing is most frequently at a low floor level below the fixed on-demand price, with occasional spikes reflecting peaks in demand (Reprinted with permission from AWS Web Console[21]).

and all data movement is performed via HTTPS (to and from S3) or via SSH connections (partial result retrieval). Each AceCloud client install automatically generates its own unique SSH keypair—no third-party SSH access is possible, nor is password authentication permitted.

AceCloud uses the AWS virtual private cloud (VPC) service to provide logical network isolation for EC2 instances. The VPC can be further configured to route all communications via a dedicated virtual private network (VPN) connection back to the user's local infrastructure. This VPC connection further secures the transmission of potentially sensitive data between client and cloud.

AceCloud requires read–write access to EC2, S3, and VPC services. Rather than assign full access to the user's AWS account, limited credentials can be generated using the Identify and Access Management (IAM) servce. These restrict the client to interacting only with the specified services. Multiple sets of such credentials can be created, allowing several users to independently use a single AWS account (e.g., for billing purposes).

To provide maximal protection to the Amazon account, EC2 instances never receive any (IAM) access credentials. Instead, presigned URLs generated and provided by the client are are used to allow the instance limited access to S3.

For users for whom compliance to industrial or regulatory standards is required, AWS complies to many of the major industry assurance programs, including SOC1-3, ISO27001, and ISO9001 (a full list is given in ref 15). AWS services are replicated in several different geographic regions. Users subject to regulatory issues may prefer to restrict their use to resources within certain territories for compliance purposes.

Cost-Optimization. As mentioned above, EC2 instances are available with fixed or spot pricing. For the instance type used by AceCloud, the spot price is often as much a 10 times lower than the on-demand price. Historic trends for the spot pricing for the three US regions are given in Figure 4, showing long periods of low pricing interspersed with peaks indicating transient peaks in demand.

Because the price differential between on-demand and spot is so marked, AceCloud is designed to use the latter. Furthermore, the client always attempts to allocate jobs to the cheapest zone within the AWS region it has been configured to use. The user may examine the spot pricing across the all AWS regions with the `--spots` option. Since spot pricing is volatile, the user sets a maximum bid price that they are prepared to pay for access. If the spot exceeds that threshold, instances are liable for termination, but costs remain controlled. While the spot remains below the set limit, the user's running instances incur charges at the prevailing spot price for the zone in which they are running.

The state of any instances that are terminated is lost. To guard against unnecessary loss of data, the best practice is to periodically perform incremental retrieval of results from long-running or important simulations.

MD Capabilities and Performance. AceCloud has built-in support for ACEMD[1] and unmodified GROMACS (Version 5.0).[2] ACEMD supports simulations parametrized with AMBER and CHARMM force fields, running in NVE, NVT, or NPT ensembles with the PME treatment of long-range electrostatics. Additionally, metadynamics simulations may be performed with PLUMED,[16] versions 1.3 and 2.0. The AceCloud version of GROMACS supports any simulation type that can be completely specified within a normal TPR-format input file.

The `g2.2xlarge` EC2 instance type is equipped with an Nvidia K520 GPU. This device is able to attain a rate of 105 ns/day on the DHFR benchmark (23 000 atoms, cutoff = 9 Å, dt = 4 fs, PME, NVT). This represents approximately 50% of the performance of the Tesla K40, the current top-end GPU at the time of writing. Despite the relatively low performance, the cost means they are still economic to use. Furthermore, for ensembles of simulations the overall throughput is of greater importance than that of any one individual simulation (provided that exceeds some acceptable minimum). The operation of both codes is in no way affected by the virtualized execution environment. Simulation outputs will be equivalent to that produced by execution on local physical resources, subject to the limits of reproducibility for the code in question.

USAGE EXAMPLE

AceCloud is well-suited to executing ensembles of simulations, such as those employed by De Fabritiis et al. in protein−ligand free binding and ab initio protein folding protocols.[4,17] An example of a 10-way ensemble of the benzamidine−trypsin model system is available for download from ref 18. This section provides a summary of the steps required to execute this ensemble on AceCloud.

Prerequisites. AceCloud access is provided via the current free release of **ACEMD**,[1] which should be downloaded and installed. An Amazon Web Services account is also required; these can be created via http://aws.amazon.com/.

AceCloud requires a subscription to the *AceCloud Molecular Dynamics* Product in the Amazon Web Service Marketplace. The product page is found at https://aws.amazon.com/marketplace/pp/B00Q5ECSOG. Follow the instructions there to subscribe, but do not launch any instances.

Finally, the client must be provided with access credentials for the Amazon account. These are obtainable from ref 19. The client is configured with these credentials using the option `--configure auth`, e.g.

```
$ acemd --configure auth XXXX-KEY-XXXX   XXXX-SECRET-XXXX
```

Running Simulations. Submit each of the 10 simulation inputs using the `--submit` argument:

```
$ acemd --submit --project bentryp *
```

To monitor the progress, use `--status`:

```
$ acemd --status
  Queued    Running   Completed     Aborted   Project
       0         10           0           0   bentryp

$ acemd --status --long
 Project       Status        Name
 bentryp       RUNNING       /home/tmp/Benzamidine-Trypsin/0
 bentryp       RUNNING       /home/tmp/Benzamidine-Trypsin/1
         ...
 bentryp       RUNNING       /home/tmp/Benzamidine-Trypsin/9
```

Jobs will progress from *Queued* through *Running* to *Completed*. Once all jobs are in the completed state, the output can be retrieved back to the submission directories with `--retrieve`. To obtain partial output from uncompleted simulations, append the argument `--incremental`. `--verbose` can be added to see the operations in progress.

```
$ acemd --retrieve
[Ctrl-C to terminate]
$ ls 0
log
output.xtc
...
```

These simulations are each configured to run for 20 ns. The computing resources currently used by AceCloud can execute these simulations at a rate of 65 ns/day, meaning that the ensemble will complete within 8 h. This throughput exceeds that attainable on a workstation equipped with 4 Tesla K40 GPUs (≈11 h).

DISCUSSION

Molecular dynamics is an established method for in silico investigation of biomolecular processes. In previous work[20] we have made the case that, as a result of technological and methodological developments, MD is poised for transition from a research context into routine industrial and commercial use.

In logistical terms MD simulation remains challenging to employ: the scale and specialization of the computing resource required to support state-of-the-art methods[17] put it beyond casual or occasional use of any group without access to dedicated high performance computing resources. To mitigate this problem, and promote the wider uptake of MD in general, we have developed AceCloud. By exploiting "cloud computing"—the broad trend in the IT sector toward flexible deployment of computing resources and services through third-party providers—we are able to provide low-cost, on-demand access to MD simulation as a robust, production-quality service.

We believe AceCloud will be a useful tool for users of molecular dynamics simulation in both industry and academia and provide a platform for innovation in higher-level protocols, for example adaptive sampling.[17] Furthermore, AceCloud is an exemplar in delivering access to scientific computing—generally considered the purview of specialized high performance computing centers—through a general-purpose cloud resource provider. The operational benefits that accrue from cloud use are, we contend, substantial, and this mode of deployment will become an increasingly common way of delivering resource-intensive computation to end-users across a wide range of scientific disciplines.

AUTHOR INFORMATION

Corresponding Authors

*E-mail: m.j.harvey@acellera.com (M.J.H.).
*E-mail: gianni.defabritiis@upf.edu (G.D.F.).

Notes

The authors declare the following competing financial interest(s): The authors declare a financial interest in Acellera Ltd.

ACKNOWLEDGMENTS

G.D.F. acknowledges support by the Spanish Research program (ref. BIO2011/27450).

REFERENCES

(1) Harvey, M. J.; Giupponi, G.; de Fabritiis, G. ACEMD:Accelerating Biomolecular Dynamics in the Microsecond Time Scale. *J. Chem. Theory. Comput.* **2009**, *5*, 1632−1639.

(2) Pronk, S.; Paill, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; C, S. J.; Kasson, P. M.; van der Spoel, D.; Hess, B.; Lindahl, E. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics* **2013**, *29*, 1−10.

(3) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An overview of the Amber biomolecular simulation package. *WIREs Comp. Mol. Sci.* **2012**, *3*, 198−210.

(4) Buch, I.; Giorgino, T.; de Fabritiis, G. Complete reconstruction of an enzyme-inhibitor binding process by molecular dynamics simulations. *Proc. Natl. Acad. Sci. U.S.A.* **2011**, *108*, 10184−10189.
(5) Buch, I.; Harvey, M. J.; Giorgino, T.; Anderson, D. P.; de Fabritiis, G. High-Throughput All-Atom Molecular Dynamics Simulations Using Distributed Computing. *J. Chem. Inf. Model.* **2010**, *50*, 397−403.
(6) Beberg, A. L.; Ensign, D. L.; Jayachandran, G.; Khaliq, S.; Pande, V. S. Folding@Home: Lessons from eight years of volunteer distributed computing. *IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009*, Rome, May 25−28, 2009; pp 1−8.
(7) Yates, J.; Hamilton, M.; Dhanoa, H.; Jenner, C.; Hong, N. C.; Hettrick, S.; Parchment, O.; Real, A.; Richards, A.; Burbidge, S. *Report of 2014 UK National E-Infrastructure Survey (HEIs and Reserch Institutes)*; 2014.
(8) Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; Zaharia, M. A View of Cloud Computing. *Comm. ACM* **2010**, *53*, 50−58.
(9) https://aws.amazon.com (accessed 12 Feb 2015).
(10) https://console.aws.amazon.com (accessed 12 Feb 2015).
(11) Fielding, R.; Taylor, R. N. Principled Design of the Modern Web Architecture. *ACM Trans. Internet Technol.* **2002**, *2*, 115−150.
(12) Amazon Web Services Inc. *AWS Documentation*. https://aws.amazon.com/documentation (accessed 15 January 2015).
(13) Ben-Yehuda, O. A.; Ben-Yehuda, M.; Schuster, A.; Tsafrir, D. Deconstructing Amazon EC2 Spot Instance Pricing. *ACM Trans. Econ. Comput.* **2013**, *1*, 1−20.
(14) The IEEE. The Open Group, The Open Group Base Specifications Issue 7, IEEE Std 1003.1. http://pubs.opengroup.org/onlinepubs/9699919799/, 2013; (accessed 15 January 2015).
(15) https://aws.amazon.com/compliance/ (accessed 12 Feb 2015).
(16) Bonomi, M.; Branduardi, D.; Bussi, G.; Camilloni, C.; Provasi, D.; Paitari, P.; Donadio, D.; Marinelli, F.; Pietrucci, F.; Broglia, R.; Parrinello, M. PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Comput. Phys. Commun.* **2009**, *180*, 1961−1972.
(17) Doerr, S.; de Fabritiis, G. On-The-Fly Learning and Sampling of Ligand Binding and High-Throughput Molecular Simulations. *J. Chem. Theory. Comput.* **2014**, *10*, 2064−2069.
(18) Harvey, M. J. ACEMD Simulation Inputs for Benzamidine−Trypsin Ensemble. 2015; DOI: 10.6084/m9.figshare.1290977.
(19) https://console.aws.amazon.com/iam/home?nc2=h_m_sc#security_credential (accessed 25 Feb 2015).
(20) Harvey, M. J.; De Fabritiis, G. High-Throughput Molecular Dynamics: The Powerful New Tool For Drug Discovery. *Drug. Discov. Today* **2012**, *17*, 1059−1062.
(21) https://console.aws.amazon.com/ec2/v2/home (accessed 16 April 2015).