Utilising Amazon Web Services to provide an on demand urgent computing facility for climate*prediction*.net

Peter Uhe*†, Friederike E. L. Otto*, Md Mamunur Rashid†, David C.H. Wallom†
* Environmental Change Institute,
University of Oxford,
Oxford, United Kingdom

† Oxford e-Research Centre,
University of Oxford,
Oxford, United Kingdom

Email: Peter.Uhe@ouce.ox.ac.uk

***Abstract*—climate*prediction*.net has traditionally been an activity that requires a large amount of computing resources from its volunteer network, whilst allowing a time-frame of weeks to months for simulations to be returned for each project. However, there is an increasing trend of projects requiring results in shorter and shorter timescales. Under no project is this clearer than in the World Weather Attribution (WWA) initiative, where we are aiming to provide in near to real-time an answer to how anthropogenic climate change has altered the frequency of occurrence of a particular type of extreme weather event, either as it happens or as soon after as is practical. As such we need the ability to run simulations on alternate resources when volunteer resources will not provide results within the necessary timeframe.**

This paper describes a workflow to distribute ensembles of climate*prediction*.net simulations in the Amazon Elastic Compute Cloud, to provide urgent compute capability for projects such as WWA. We propose a method of optimizing the use of cloud resources to minimize cost while maximising throughput. A case study is presented to provide a proof of concept of this methodology. As such, this is a clear example of beneficial utilisation of cloud resources to supplement those available through our volunteer community.

I. Introduction

A. World Weather Attribution and the need for urgency

climate*prediction*.net (CPDN, [1]) is a climate modelling project using volunteer computing to run very large ensembles of climate simulations. It is run within the Berkeley Open Infrastructure for Network Computing (BOINC) framework, and tens of thousands of climate simulations are computed every year, with the results sent back to CPDN servers to be analysed by the researchers on the project. Using volunteer computing has been extremely successful since the start of CPDN in 2003. However, we are seeing increased interest in different aspects of climate change and its impacts, and new applications of CPDN are required to provide many different scenarios. This includes time critical operations that were never part of the design of BOINC and are a challenge for systems that rely on the voluntary nature of resource donation.

The most prominent time critical activity currently undertaken in the CPDN framework is the international World Weather Attribution (WWA) initiative[1]. The aim of this project is to answer questions about the role of anthropogenic climate change in the likelihood and intensity of extreme weather events, within one or two weeks after the event. This provides scientific evidence when these events are subject to international debates in the media, which may be otherwise driven by opinion.

In order to be able to provide such real-time event attribution analysis, model simulations are performed ahead of an extreme weather event occurring by utilising seasonal forecast sea surface temperatures [2]. For some events, very large ensembles of simulations are required to identify subtle but robust changes in the occurrence of, e.g., extreme rainfall events. However, we do not have the capacity to run high resolution simulations over the whole globe in advance. Depending on the nature of the event and where it has occurred, we may need to run additional simulations over a specific region to improve our statistics on the event.

In addition to the computational requirement of this activity, we need to consider the aim of addressing the questions people impacted by the event are asking. Knowing what the impacts are requires information to be provided while the event is unfolding (this can be as short as a couple of days, up to weeks). Therefore, for events where we want to look at specific impacts which are not part of our standard model output, we may want to submit new simulations with output designed to address those impacts, and hence enhance the public discussion about extreme weather events and the role

[1] http://www.climateprediction.net/weatherathome/world-weather-attribution/

of anthropogenic climate change. These types of submissions fit into the urgent computing paradigm as we have a hard deadline of when these results are made public and the event is still fresh in the public consciousness, and after which time new results will not be used.

B. Conventional solutions to provisioning resources for urgent work

For the case where we require additional simulations and volunteer computing is unable to produce results within the required timeframe, we can look to commonly applied approaches to urgent computing.

The most common method by which any particular domain is able to provide an urgent computing capable facility is either through the provision of dedicated systems for this task or the over provisioning of a system that is in use for other non-urgent tasks [3]. An example of where this occurs is in the area of disaster response and management such as earthquake monitoring. This has a number of disadvantages around efficiency, as it is expensive to maintain a dedicated resource that is infrequently used. This may be appropriate for life or death disaster management, but is not suitable for a research project like CPDN.

Alternatively supercomputer time on clusters that are available for academic research may be purchased to run these urgent simulations. However, supercomputers generally run tasks using queues. If the cluster is utilized at or near capacity it may be difficult to submit thousands of tasks simultaneously without incurring significant queuing time, unless they are given priority over other users [3]. Additionally, supercomputers are generally optimized for high-performance computing (HPC) problems, such as large tasks using many CPU cores in a connected manner rather than many small individual tasks like CPDN.

Within the supercomputing domain there have been a number of different methods applied to attempt to bypass the problems of blocking caused by standard queueing methodologies. Examples include SPRUCE [4], where a token type of system was applied, allowing users to request elevated priority status on TeraGrid [5] connected resources. This pre-empted other tasks within the system, ensuring a high rate of throughput. This system was used a number of times for different urgent computing problems but still required a large infrastructure to manage the urgent and pre-empted tasks, to ensure that all tasks were run to completion. Alongside this are other specialised infrastructure configurations that are specifically designed to support urgent computing rather than as an addition to other types of service. An example of this is the CLAVIRE platform [6] which has constructed a specialised middleware system to support urgent computing. The biggest problem with both of these systems is though that they use large scale extremely costly computing infrastructure rather than the volunteer network with which we have had such success in CPDN.

One resource that fits the many small task paradigm is cloud computing. Commercial cloud providers provide highly scalable resources which are designed for use by many separate users and hence are more closely aligned with running many small jobs than large compute jobs as may be required on a supercomputer. Cloud resources are also scalable and on demand which meets our requirement for urgent computing. An example of urgent cloud computing work is in dust storm modelling where low resolution forecasts are run regularly, but then during a severe event, this is scaled up to run high resolution forecasts [7]. In addition, the scalable and on demand nature of the cloud means that there is no need to suspend other tasks to allow for our tasks to run (which may be the case on dedicated/traditional HPC machines). So we can optimise simulation run time and cost independently of others demands.

C. CPDN configuration using BOINC

The standard BOINC system relies on a set of central servers with a network of volunteer systems (hosts) operating as computational engines. Within this network, workunits (the unit of computational activity within BOINC systems) are generated within the central services, distributed and then received back in these services. There is also a degree of fault tolerance within the BOINC system where if a workunit crashes on a given computer it is resent out to another host to run (up to a maximum number of failures for a given workunit).

Any of the resources above could be used as hosts that compute simulations for CPDN. The main requirement is that the client software can be installed and the clients can communicate with the central CPDN servers. In the following sections, we introduce how the CPDN workflow can be utilised in a cloud environment and give an example of how cloud computing could supplement the existing CPDN set-up. The paper concludes with an estimation of funds required to optimally combine the advantages of a volunteer user base and cloud resources, to meet the computing challenges of a time critical research project in the focus of international media.

II. Workflow for using cloud resources for CPDN

For ease of deployment, avoiding costly development of new systems and integration with the current workflow, we will only consider the deployment of cloud resources in conjunction with the existing BOINC design. The cloud resource considered here is the Amazon Web Services (AWS), Elastic Compute Cloud (EC2) service[2]. EC2 has previously be tested running CPDN simulations in [8].

As the BOINC infrastructure is already in place, the simplest mechanism for running CPDN in a cloud system is to configure cloud virtual machine instances as BOINC clients. These run as per any other volunteer and upload results back to CPDN servers. This avoids setting up an alternate control system for the CPDN simulations, but requires management of cloud resources to spin up the required computing power to run a certain number of workunits, then shut them down once they are no longer required.

[2]https://aws.amazon.com/ec2/

A. Deploying a mixed cloud and volunteer solution or stand alone system

We have a choice of whether to use cloud resources attached to the existing project or to set up a dedicated BOINC project for simulations sent to the cloud. Either of these options may be preferable depending on the particular use case.

A shared project is most appropriate in the case where we are supplementing free volunteer resources with some additional cloud resources to provide a guaranteed minimum return of simulations in a tight timeframe. However, with a shared system, we may not be able to make optimum use of Amazon EC2 resources due to BOINC mechanisms preventing volunteers from requesting work too frequently resulting in time when EC2 instances are idle. There are also costs to upload data out of Amazon (currently around $90 per TB). However, this should be able to be waived if sending results back to a university server via a National Research and Education network[3].

If we were to run a stand-alone system within AWS instead, we could also benefit by running the BOINC infrastructure within the cloud. The download and upload servers should be straightforward to set up in the cloud [8]. This would speed up data transfers and minimise transfer costs out of AWS. With data uploaded to AWS, we also have the potential for implementing instances within Amazon EC2 to analyse the results of the simulations, with direct access to the data. It would also be possible to make simulation results publically available directly from the cloud and thus simplify data access requirements enjoined by publicly funded research projects and many scientific publishers.

B. Design for controlling deployment of supplementary workunits using existing boinc infrastructure

Within Amazon EC2, spot fleets are the most cost-effective tool to manage resources[4]. These use EC2 spot instances which are significantly cheaper than on demand instances. Spot instances operate on a market price which is based on demand and differs across AWS regions and by availability zones (effectively which data centre the instances are housed in).

The main downside of spot instances is that if there is high demand and the price rises higher than the amount you bid, your instances can be terminated. However, the design of CPDN doesnt require specific simulations to be returned so can cope with a small percentage of instances terminating. When submitting simulations, a greater number are sent than required, to account for this loss. With this in mind, the chance of a simulation being interrupted is greater the longer it runs, so shorter simulations are better suited to use with spot instances. An alternate option to account for spot instance terminations is to save the state of terminating instances and restore the state into a new VM to continue the simulation. However, this option introduces a lot of additional complexity and is not particularly suited for our use case.

Spot fleets maintain a set capacity for a certain amount of time, and if instances are terminated, the spot fleet will automatically spin up new instances of whichever type and availability zone is cheapest at the time (spot fleets are confined to a single AWS region though).

It is important to note though that because of the massively scalable nature of cloud resources, if we want to run a large number of simulations as quickly as possible, it is more advantageous to run many instances simultaneously than running a smaller number of instances and have each instance completing simulations sequentially. Our chosen methodology for urgent tasks is therefore to spin up a spot fleet with the capacity to run a whole ensemble in one go. On each instance we start a set of workunits to fully utilize the instance, but prevent them from requesting additional workunits. The instances are terminated once all of its simulations are completed.

If additional simulations are required, new instances will be spun up (possibly a different instance type if the spot price has changed since the initial submission). The spot fleets capacity can be dynamically modified at any time to request more or less instances. The request can also be cancelled with or without terminating running instances (depending on whether we have reached a deadline after which new simulations will not be analyzed).

III. Benchmarking AWS instance types for the CPDN workload

Before deploying CPDN workunits to AWS through BOINC, we first want to optimize the use of resources. To do that we benchmark different Amazon EC2 instance types[5]. Amazon EC2 instance types are specified with a name such as ‘c4.large or ‘m3.xlarge. The first part of the name indicates the hardware configuration of the CPU e.g. c4 indicates a 4th generation compute optimised instance, m3 indicates a 3rd generation general purpose instance. The second part of the name indicates the number of vCPUs (hyper-threads) and amount of memory assigned to the instance e.g. ‘c4.large has 2vCPUs and 3.75GB memory, ‘m3.xlarge has 4 vCPUs and 15GB memory. We expect that all instances of the same type will have the same hardware configuration and hence similar performance.

The benchmarks were carried out running multiple copies of a stand-alone CPDN test workunit in parallel, with the number of simulations matching the number of vCPUs available on each instance type. The test workunit uses the weather@home model e.g. [9] with a 25km resolution regional domain over Europe, running a single day of simulation. This setup is consistent with a normal CPDN simulation except the test simulation does not connect to the CPDN servers and is a shorter length. For each instance types at least 4 benchmarks were run.

[3]https://aws.amazon.com/blogs/publicsector/aws-offers-data-egress-discount-to-researchers

[4]https://aws.amazon.com/blogs/aws/amazon-ec2-spot-fleet-api-manage-thousands-of-instances-with-one-request/

[5]https://aws.amazon.com/ec2/instance-types/

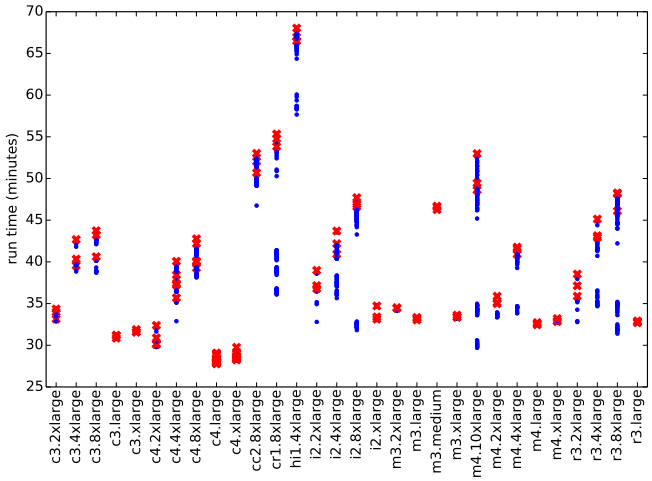

Fig. 1. Time taken to complete 1 day benchmark climate simulation using different EC2 instance types. Blue dots are completion times of each individual simulation and red crosses are the longest run time on each instance (e.g. how long that instance ran for).

On each instance, the individual simulations take different times to complete (depending on how much CPU time each process gets on the underlying physical machines). Fig. 1 shows the spread of run times of simulations (blue dots) with the run time of the instances (e.g. slowest running simulation on that instance) shown as a red cross. Optimising the balancing of load across the simulations may improve performance but is beyond the scope of this exercise.

Table I shows a list of instance types, the number of vCPUs and average run time required to complete all simulations on the instance (average time of crosses in Fig. 1). These combine to give the number of days of model simulation we can expect to compute in one instance hour. This shows a general trend of larger instances completing simulations more slowly. We assume this is to do with the way instances share the underlying CPU with other EC2 users. With smaller instances there is a greater chance the CPU is running at a lower utilization and our instances can scavenge extra CPU cycles.

Using the model days simulated per instance hour and the spot price of each instance, we can estimate the cost of running a year of simulation. Shown in Fig. 2 is this cost for a subset of instance types in each US region (where there are multiple Availability Zones in a region, the cheapest for each instance type is chosen).

When we submit simulations with a hard deadline, we also need to take into account the run length and ensure that on all instance types requested in our spot fleet the simulations will complete within the required time. In the analysis of Fig. 2, we discarded instance types which had a run time greater than 45 minutes in Table I (this will differ in real use cases). We also discarded the i2 instances which were significantly more expensive than other instance types.

TABLE I
LIST OF INSTANCE TYPES: NUMBER OF VCPUS, RUN TIMES (AVERAGE OF INSTANCE RUN TIME) AND MODEL DAYS COMPLETED PER INSTANCE HOUR BASED ON THIS RUN TIME AND NUMBER OF VCPUS.

Instance type	vCPUs	run time (min)	model days per instance hour
m3.medium	1	46.51	1.29
m3.large	2	33.20	3.61
r3.large	2	32.81	3.66
m4.large	2	32.59	3.68
c3.large	2	31.07	3.86
c4.large	2	28.19	4.26
i2.xlarge	4	33.75	7.11
m3.xlarge	4	33.45	7.17
r3.xlarge	4	33.39	7.19
m4.xlarge	4	32.99	7.28
c3.xlarge	4	31.68	7.57
c4.xlarge	4	28.64	8.38
i2.2xlarge	8	37.64	12.75
r3.2xlarge	8	37.18	12.91
m4.2xlarge	8	35.33	13.58
m3.2xlarge	8	34.48	13.92
c3.2xlarge	8	33.83	14.19
c4.2xlarge	8	30.87	15.55
hi1.4xlarge	16	67.16	14.29
r3.4xlarge	16	43.73	21.95
i2.4xlarge	16	42.27	22.71
m4.4xlarge	16	41.42	23.18
c3.4xlarge	16	40.86	23.49
c4.4xlarge	16	37.69	25.47
cr1.8xlarge	32	54.64	35.14
cc2.8xlarge	32	51.92	36.98
r3.8xlarge	32	47.51	40.41
i2.8xlarge	32	47.08	40.78
c3.8xlarge	32	42.53	45.15
c4.8xlarge	36	40.67	53.11
m4.10xlarge	40	50.35	47.67

A. *Optimizing cost of simulations*

The model days per instance hour in Table I is used as a weighting factor by the AWS spot fleet to determine which instance type is the most cost effective. The spot fleet is configured with the allocation strategy: 'lowestPrice, so when starting new instances, the spot fleet will dynamically determine which instance type is the cheapest (as in Fig. 2) and spin up instances of that type.

The method above takes a simple view of optimization and chooses the cheapest instance type at the time of submission. This is most appropriate when the run times of simulations are short relative to the rate of change in the spot market. For longer simulations or where the volatility in the spot market is high, we may encounter issues where instances are terminated or result in a significantly higher cost than expected.

In these cases, the selection criteria would need to be modified to take the volatility into account. Possible alternate measures for instance type selection are lowest average price over the previous day (or appropriate time period). If terminations are more of a problem then choosing instance types with the lowest maximum spot price over a recent time period may be more appropriate. However, as spot-fleets only offer an allocation strategy which uses the instantaneous lowest price, these measures would also require developing more sophisticated control software to spin up and manage the instances.

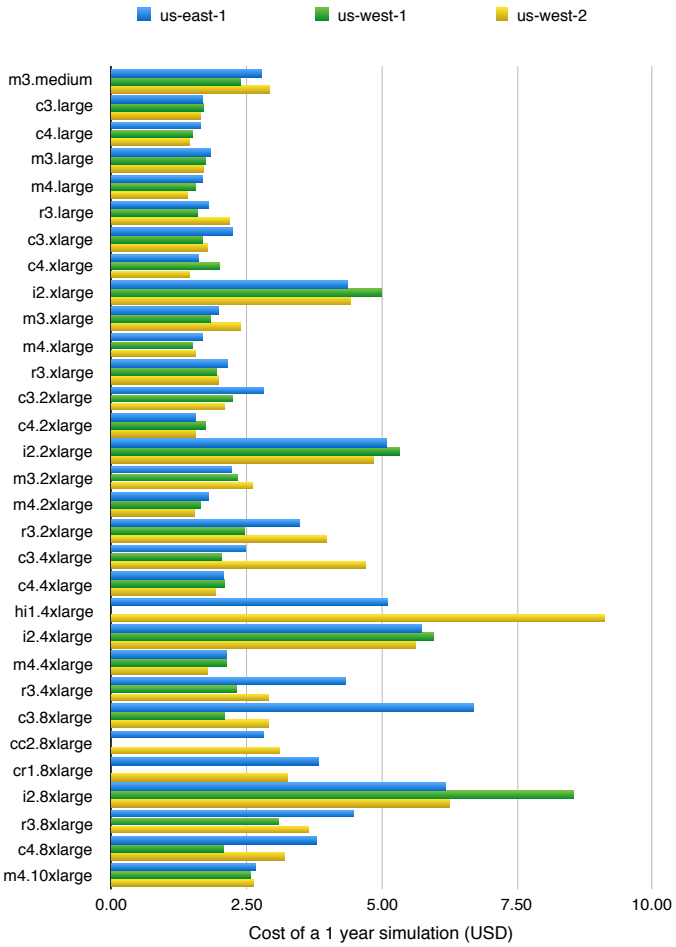

Fig. 2. Cost of 1 year of simulation in USD, by instance type and region (assuming using the cheapest availability zone in each region). Three AWS regions are shown: us-east-1 (blue), us-west-1 (green) and us-west-2 (yellow). Prices used are from the spot market as on 13/06/2016 13:45

All calculations above are made under the assumption that we are not a large enough user to use a significant proportion of the overall cloud and hence are not impacting the spot market price or instance availability (this is helped by considering as many as possible different instance types and using multiple regions).

B. Optimizing instance load

Another aspect to consider when running these instances is that each vCPU on an Amazon EC2 instance is actually a hyper-thread rather than a CPU core. So running an instance utilizing only half the hyper-threads may allow each simulation to have access to the full CPU cycles of one core and speed up the model run. The run times of simulations using c4 instance types under different loads are shown in Fig. 3. Utilizing an instance using 50% of the vCPUs or less results in run time of roughly 20 minutes which is significantly faster than using 100% of the vCPUs. However, as discussed in section III, Amazon EC2 instances are on hardware shared with other users which may be underutilizing the CPU. Because of this, the smaller instance types see a significant benefit of running

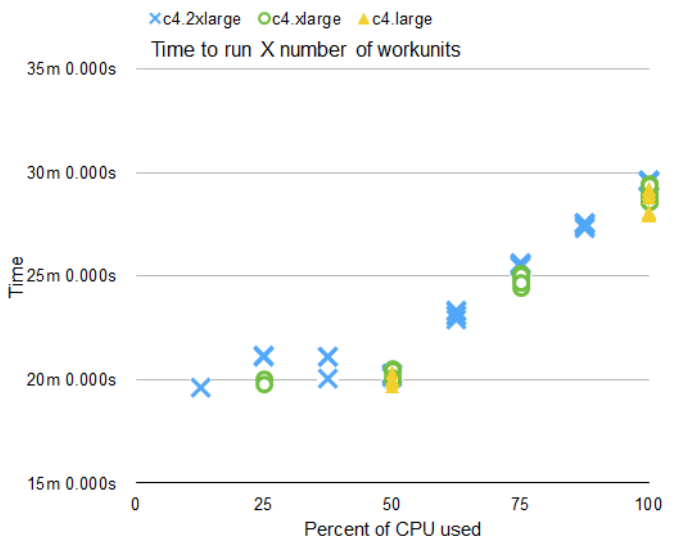

Fig. 3. Dependence of model run time on CPU utilization. This shows the change in run time as the percent of vCPUs used is increased. Three instance types are shown: c4.large, c4.xlarge and c4.2xlarge

on 100% of the vCPUs e.g. looking at Fig. 3 we see that a c4.2xlarge instance may run 4 simulations in 20 minutes using 50% of the vCPUs (and hence 8 simulations in 40 minutes), compared to 8 simulations in 30 minutes using 100% of the vCPUs. Here we have the notable exception of the c4.8xlarge which takes up a whole CPU and running on 100% of the vCPUs computes a simulation in roughly 40 minutes which is twice the time compared to the simulations using 50% of the vCPUs in Fig. 3. In that case it should be just as cost effective to run 18 simulations using half of the vCPUs as 36 simulations running at half the speed on all of the vCPUs.

However, if urgent computing takes a higher priority than cost (e.g. in the case where the deadline is too short for simulations to be completed when utilizing all the vCPUs), it would be necessary to run less workunits on each instance. Running on 50% of the vCPUs available, so each simulation has a dedicated CPU core, should result in the best performance in terms of simulation speed. If submitting simulations in this configuration, different benchmarks would be required to reflect the instance performance under this load.

IV. Case Study of a CPDN Ensemble Sent to Amazon EC2

As a proof of concept, we have sent an ensemble of 750, 13 month simulations to Amazon EC2 instances. These simulations cover a different region to the benchmarks (South America at 50km resolution compared to Europe at 25km resolution) so the run times of these longer simulations are not comparable to the benchmarks above. However we assume that the relative performance of the different instance types will be the same. We also note that the simulation results were uploaded to a climateprediciton.net server rather than remaining in AWS.

These simulations were distributed across two AWS regions: North Virginia and Oregon. The configuration of the spot fleets sent were as follows. The spot fleet instance weightings used the 'model days per instance hour normalised with the 'c4.large (fastest instance with 2 vCPUs) having a value of 2. This allowed the requested capacity of the spot fleet to be roughly equivalent to the number of vCPUs e.g. a requested capacity of 20 will result in at least 20 vCPUs being available. The maximum bid price was set for each instance type as 5 times the spot price at the time of submission to reduce the likelihood of terminations.

The instances were configured using a startup script which installed the relevant software and libraries and connected to the climatprediction.net BOINC project. This script then continued to monitor the state of the BOINC project and cleaned up and terminated the instances shortly after the simulations were complete.

In total 318 instances were spun up. The capacity of the spot fleets in each region were gradually increased over periods of roughly half an hour. This was aimed to reduce the likelihood of all instances being spun up with the same instance type in the same availability zone. There were two simulations which crashed about a tenth of the way through due to model instabilities. There were also 12 c3.large instances which were terminated within a few hours of the submission (this appears due to spikes in the spot market where the price reached 10 times the on demand price), but the remainder of instances continued to completion. An outcome of this case study is that for future ensembles, we will need to filter instance types to remove those which have recently had spikes in price. A simple solution to this would be to remove those instance types from the spot-fleet, however if may also be beneficial to investigate other strategies such as discussed in section III-A. Three instance types had simulations that ran to completion: c4.large, c4.xlarge and m3.large. The run times for these instances were: c4.large 101.6 hours (range: 100.3 to 104.75), c4.xlarge 102.493902439 hours (range: 91.1 to 106.9), m3.large 119.6 hours (range: 118.4 to 120.6). We note that the outlier of 91.1 hours for the c4.xlarge instance type occurred where one of the four simulations had crashed, leading to the remaining three simulations to complete more quickly. This highlights the consequence of underfilling instances on performance.

Ratios: c4.xlarge 1.008 times c4.large, m3.large 1.18 times c4.xlarge. This compares to from the benchmarks: c4.xlarge 1.016 times c4.large, m3.large 1.18 time c4.large. So for this example, the average run lengths of our longer tests give good agreement of the relative performance of instance types. In addition, from monitoring the usage of these instances, they appear to consistently use 100% of the CPU. As the CPU is the bottleneck for the model computation, we expect that the timings of our simulations to scale linearly with run length, further justifying the use of one day simulations as benchmarks.

V. Use case for urgent CPDN ensemble in Amazon EC2

The methodology for the attribution of extreme weather and climate-related events is based on the simulation of large ensembles of possible weather under present day and counterfactual climate conditions using a general circulation climate model [10]. Within the WWA project we typically run three month long simulations (at least one month before the event and one month after). These simulations use GloSEA5 seasonal forecast [11] sea surface temperatures as boundary conditions so it is possible to run them ahead of time before an event occurs (see [2]). However, as introduced above, we may not have simulations that describe the event in the most impact relevant way, i.e. an additional set of climate variables is required, or the ensemble is too small to obtain robust results. Hence cloud resources can be utilised to supplement the simulations with additional ensembles.

We can estimate the cost of running an ensemble such as this in Amazon EC2, using the benchmarks above. We take the cheapest instance type from Fig. 2 (m4.large in us-west-2). This takes 33 minutes per day of simulation (Table I), which is 49.5 hours for a three month simulation. Using the spot price as per Fig. 2, the compute cost for 10,000 workunits will be roughly $4700. We also have the additional cost of storage used by the instances while they are computing the simulations. Given an estimated disk usage of 4GB per workunits for 49.5 hours, we estimate this to be $275. This results in a minimum cost of $5000 to run such an ensemble. This costing will depend on the resolution and size of the region simulated (the example here is a 25km resolution regional model over Europe), a 50km resolution model for the same region would take less than half the time to compute and cost less than half the price to run.

Data storage of the simulation results is an additional cost which will depend on how much model output is requested for analysis and how long the output needs to be stored (for example data used in publications commonly need to be kept for at least 5 years). As of June 2016, data storage in AWS Simple Storage Service costs $30 per month per TB. If data is uploaded to other servers, this cost will vary.

As a comparison, a similar ensemble would cost over twice the amount run on a dedicated resource such as the University of Oxford Advanced Research Computing (ARC) high performance computer arcus-b[6]. The cost to run the above ensemble would be around $12,800, based on an average benchmark run time of 32:23 and cost per CPU core of £0.02 per hour. This was based on 7 benchmarks on arcus-b where 16 simulations were run in parallel on a 16 CPU core node (comparable to the AWS set up). The storage on ARC however is slightly cheaper than AWS at $27.40 per month per TB. Prices quoted here assume exchange rates between GBP(£) and USD($) as on 28/07/2016. Lastly we note that running on EC2 on-demand instances rather than spot instances would be

[6]http://www.arc.ox.ac.uk/content/services

much more expensive again (e.g. $54,000 given the m4.large instance as above).

VI. Conclusion

We have presented a workflow for including AWS cloud resources within the BOINC distributed computing framework to supplement existing volunteer computing resources for CPDN. This includes optimizing the resources by benchmarking a range of AWS EC2 instance types and using spot fleets to choose the cheapest instances at the time of submission. We assume that for a given instance type, the hardware configuration used by AWS will be the same. We do note that as our benchmarks seem to depend on instance load, and usage patterns of AWS may change over time, recalculating benchmarks at regular intervals is a good idea. In addition to this, the usage patterns will differ between region (and possibly availability zones) so calculating benchmarks for different regions and availability zones may improve our optimization.

When results are needed within a very tight timeframe, and it is not expected that volunteer resources (which are only typically available when participants computers are on and idle) can produce them within that timeframe, dedicated cloud resources may be able to meet this requirement. Cloud resources also have potential to add additional compute capability where the volunteer computing base has decreased, or the project computing requirement has increased. However the cost of cloud resources are significant, so at the present time we consider their use most suited to the urgent computing case that by design cannot be met by volunteer computing. This makes them a very useful resource, but we stress that they cannot in the foreseeable future replace the unique computing framework provided by our generous and dedicated volunteer base.

Acknowledgment

We would like to thank our colleagues at the Oxford eResearch Centre: A. Bowery and S. Sparrow for their technical expertise. We would like to thank the Met Office Hadley Centre PRECIS team for their technical and scientific support for the development and application of weather@home. This work was funded by Climate Central through the World Weather Attribution project. The compute resources were provided under the AWS Cloud Credits for Research Program.

References

[1] M. Allen, "Liability for climate change," *Nature*, vol. 421, pp. 891–892, 2003.

[2] K. Haustein, F. E. L. Otto, P. Uhe, N. Schaller, M. R. Allen, L. Hermanson, N. Christidis, P. McLean, and H. Cullen, "Real-time extreme weather event attribution with forecast seasonal SSTs," *Environmental Research Letters*, vol. 11, no. 6, p. 064006, 2016. [Online]. Available: http://stacks.iop.org/1748-9326/11/i=6/a=064006

[3] S. H. Leong, A. Frank, and D. Kranzlmüller, "2013 International Conference on Computational Science: Leveraging e-Infrastructures for Urgent Computing," *Procedia Computer Science*, vol. 18, pp. 2177 – 2186, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913005310

[4] P. "Beckman, S. Nadella, N. Trebon, e. P. W. Beschastnikh, Ivan", and J. C. T. Pool, *"SPRUCE: A System for Supporting Urgent High-Performance Computing"*. Boston, MA: Springer US, "2007", pp. 295–311. [Online]. Available: "http://dx.doi.org/10.1007/978-0-387-73659-4_16"

[5] C. Catlett, "The philosophy of teragrid: Building an open, extensible, distributed terascale facility," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, May 2002, pp. 8–8.

[6] V. Alexandrov, M. Lees, V. Krzhizhanovskaya, J. Dongarra, P. M. Sloot, S. V. Kovalchuk, P. A. Smirnov, S. V. Maryin, T. N. Tchurov, and V. A. Karbovskiy, "2013 international conference on computational science deadline–driven resource management within urgent computing cyberinfrastructure," *Procedia Computer Science*, vol. 18, pp. 2203 – 2212, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913005346

[7] Q. Huang, C. Yang, K. Benedict, S. Chen, A. Rezgui, and J. Xie, "Utilize cloud computing to support dust storm forecasting," *International Journal of Digital Earth*, vol. 6, no. 4, pp. 338–355, 2013. [Online]. Available: http://dx.doi.org/10.1080/17538947.2012.749949

[8] D. P. Montes, "climateprediction.net: A Cloudy Approach," Master's thesis, Universidade de Santiago de Compostela, June 2014.

[9] N. Massey, R. Jones, F. E. L. Otto, T. Aina, S. Wilson, J. M. Murphy, D. Hassell, Y. H. Yamazaki, and M. R. Allen, "weather@home–development and validation of a very large ensemble modelling system for probabilistic event attribution," *Q. J. Roy. Meteor. Soc.*, pp. n/a–n/a, 2015. [Online]. Available: http://dx.doi.org/10.1002/qj.2455

[10] N. Schaller, F. E. L. Otto, G. J. van Oldenborgh, N. R. Massey, S. Sparrow, and M. R. Allen, "The heavy precipitation event of May–June 2013 in the upper Danube and Elbe basins [in "Explaining Extremes of 2013 from a Climate Perspective"]," *Bull. Amer. Meteor. Soc.*, vol. 95, no. 9, pp. S69–S72, 2014.

[11] C. MacLachlan, A. Arribas, K. A. Peterson, A. Maidens, D. Fereday, A. A. Scaife, M. Gordon, M. Vellinga, A. Williams, R. E. Comer, J. Camp, P. Xavier, and G. Madec, "Global seasonal forecast system version 5 (glosea5): a high-resolution seasonal forecast system," *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 689, pp. 1072–1084, 2015. [Online]. Available: http://dx.doi.org/10.1002/qj.2396