Cloud Benchmarking For Maximising Performance of Scientific Applications

Blesson Varghese, Ozgur Akgun, Ian Miguel, Long Thai and Adam Barker

Abstract—*How can applications be deployed on the cloud to achieve maximum performance?* This question is challenging to address with the availability of a wide variety of cloud Virtual Machines (VMs) with different performance capabilities. The research reported in this paper addresses the above question by proposing a six step benchmarking methodology in which a user provides a set of weights that indicate how important memory, local communication, computation and storage related operations are to an application. The user can either provide a set of four abstract weights or eight fine grain weights based on the knowledge of the application. The weights along with benchmarking data collected from the cloud are used to generate a set of two rankings - one based only on the performance of the VMs and the other takes both performance and costs into account. The rankings are validated on three case study applications using two validation techniques. The case studies on a set of experimental VMs highlight that maximum performance can be achieved by the three top ranked VMs and maximum performance in a cost-effective manner is achieved by at least one of the top three ranked VMs produced by the methodology.

Index Terms—cloud benchmark, cloud performance, benchmarking methodology, cloud ranking

1 Introduction

The cloud computing marketplace offers a wide variety of on-demand resources with a wide range of performance capabilities. This makes it challenging for a user to make an informed choice as to which Virtual Machines (VMs) need to be selected in order to deploy an application to achieve maximum performance. Often it is the case that users deploy applications on an ad hoc basis, without understanding which VMs can provide maximum performance. This can result in the application under performing on the cloud and consequently increasing running costs. This paper aims to address the above problem.

The above problem is addressed by benchmarking to measure the performance of computing resources [1], [2], which is employed on the cloud [3], [4], [5]. Typically, cloud benchmarking is performed independently of the application that needs to be deployed, and does not consider bespoke requirements an application might have. Our research advances the start-of-the-art by proposing an application aware benchmarking methodology that accounts for the requirements of the applications.

We hypothesise that by taking into account the requirements of an application, along with benchmarking data collected from the cloud, VMs can be ranked in order of performance and cost effectiveness so that a user can deploy an application on a cloud VM, which will maximise performance. In this paper, the focus is on scientific High-Performance Computing (HPC) applications and maximum performance is defined as the minimum execution time of an application. Our motivation for choosing scientific applications is that the cloud is becoming an alternate computing platform to do HPC without owning supercomputers [6], [7]. However, on the cloud, applications will need to use a pay-as-you-go model for running on cloud VMs that share physical nodes with other applications. This is in contrast to grants and quotas which are available for utilising tightly coupled nodes of supercomputers which are less likely to be shared with other applications. Deploying long running applications in an ad hoc manner on the cloud will result in under performance and increased running costs. In this context, a method to determine cloud VMs that can maximise the performance of the application before it is deployed is required given the wide variety of choices offered by providers to do HPC on the cloud.

We present a six step benchmarking methodology in order to determine the VMs that can maximise the performance of scientific applications on the cloud. A user provides as input a set of weights that describe the memory and process, local communication, computation, and storage requirements of the scientific application to be deployed on the cloud. These requirements are mapped onto a set of four aggregate groups or eight fine grain groups that capture the memory and process, local communication, computation, and storage attributes of the VMs. The groups are obtained by benchmarking the cloud VMs. Based on the user's knowledge of the application (developers and domain experts are well acquainted with their applications), either a set of four abstract weights or eight fine grain weights are provided as input. The value of each weight ranges from 0 to 5, where 0 signifies

- *E-mail: varghese@qub.ac.uk, {ozgur.akgun, ijm, ltt2, adam.barker}@st-andrews.ac.uk*

that the memory and process, local communication, computation, or storage groups represented by the weight has no relevance to the application, and 5 indicates that the group is important to the application for achieving maximum performance. These weights along with the benchmarked data are used to generate rankings that take performance and cost into account.

For the purposes of verifying our hypothesis, the methodology is validated on three scientific HPC applications; the first is used in financial risk, the second employed in molecular dynamics, and the third as a mathematical solver. The memory and process, local communication, computation and storage requirements of these applications are known beforehand. The applications are embarrassingly parallel and in this research we parallelise them on the multiple cores of a single VM. Two techniques, namely comparative validation and enumeration-based validation are used to validate the rankings produced by the methodology. The validation study demonstrates that the methodology can select cloud VMs that maximise the performance of the application. If such a methodology is not adopted, the application will result in higher running costs. The contributions of this paper are the development of a benchmarking methodology for selecting VMs that maximises the performance of scientific applications on the cloud, and the validation of the methodology against real world applications.

The remainder of this paper is organised as follows. Section 2 presents the cloud benchmarking methodology proposed in this paper for maximising an application's performance on the cloud. Section 3 considers the Amazon Elastic Compute Cloud (EC2) VMs employed in this paper. Section 4 presents the benchmarks used in the methodology. Section 5 presents the aggregate and fine-grain cloud rankings generated by the methodology for different weights. Section 6 considers three case study applications to validate the cloud benchmarking methodology. Section 7 presents a discussion on the work related to the research presented in this paper. Section 8 concludes this paper.

2 Cloud Benchmarking Methodology

A six step cloud benchmarking methodology that determines which cloud Virtual Machines (VMs) can maximise application performance on the cloud is proposed. The six steps are: (1) capture attributes of cloud VMs, (2) group attributes of cloud VMs, (3) benchmark cloud VMs, (4) normalise attribute groups, (5) provide weights to groups, and (6) rank cloud VMs. Only the last two steps are necessary to be performed for each application deployment on the cloud. The first four steps are performed infrequently (only if the underlying infrastructure has changed in a private cloud or periodically for a public cloud).

Individual attributes of cloud VMs are firstly evaluated and then grouped together. The user provides a set of weights (or the order of importance) for the groups based on the knowledge of the requirements of the application to be deployed on the cloud. The weights along with the attributes of the group for each VM are used to generate a score resulting in a VM rank. Two sets of ranks are generated; the first ranking is solely based on the performance of the VMs and the second ranking considers both performance and cost. We hypothesise from the first set of ranks that the VMs with the highest ranks can maximise application performance on the cloud, and the highest ranks from the second set can maximise application performance in a cost-effective manner.

Given $i = 1, 2, \cdots, m$ different VMs, the cloud benchmarking methodology we propose is as follows:

Step 1: Capture Attributes

The attributes of a VM that describes it are firstly selected based on experience with VMs and physical machines. For example, (a) attributes such as the number of integer, float and double addition, multiplication and division operations that can be performed in one second on a VM can describe its computational capacity, or (b) attributes such as the number of sequential and random read, write and delete operations that can be performed in one second on the storage of a VM can describe its file I/O. Assume that there are $j = 1, 2, \cdots, n$ attributes of a VM. Then, $r_{i,j}$ is the value associated with the j^{th} attribute on the i^{th} VM.

Step 2: Group Attributes

The attributes of the VM are then grouped into categories based on whether they are related to memory and process, local communication, computation or storage. For example, (a) attributes such as the bandwidth of memory read and write operations and of communication using pipes, AF Unix socket and TCP are grouped together as the local communication group, or (b) attributes related to the latencies of the main and random access memory and the L1 and L2 cache can be grouped as the memory group. Each attribute group is denoted as $G_{i,k} = \{r_{i,1}, r_{i,2}, \cdots\}$, where $i = 1, 2, \cdots m$, $k = 1, 2, \cdots, p$, and p is the number of attribute groups.

Step 3: Benchmark Virtual Machines

Based on the attribute groups a set of benchmarks are evaluated on all potential cloud VMs. The benchmarks evaluate the attributes of the cloud VMs as closely as possible to the underlying hardware employed [2], [8], [9]. The benchmarks are grouped as memory and process, local communication, computation and storage evaluation groups, based on *Step 2*. Standard benchmark tools are run on the cloud VM or on an observer system collecting the results. The focus is on evaluating attributes that are closer to the hardware, such as frequencies (number of operations in a second) latencies (micro seconds or nano seconds) and bandwidths (Megabytes per second). The value of each attribute, $r_{i,j}$ is obtained in this step.

Step 4: Normalise Groups

The value of the attributes are normalised to rank the performance of VMs for an attribute group. The group based rank can provide a view of the relative performance of the VM under each group. For example, a VM that is well ranked in the memory group may poorly perform in the storage group. The normalised value [10] of each attribute $\bar{r}_{i,j} = \frac{r_{i,j}-\mu_j}{\sigma_j}$, where μ_j is the mean value of attribute $r_{i,j}$ over m VMs (the mean value is the sum of all values of an attribute for m VMs and dividing that sum by m) and σ_j is the standard deviation of the attribute $r_{i,j}$ over m VMs (the standard deviation is the sum of the squares of the difference between the value of each attribute and mean for m VMs and dividing that sum by m). The resultant normalised attribute group, is denoted as $\bar{G}_{i,k} = \{\bar{r}_{i,1}, \bar{r}_{i,2}, \cdots\}$, where $i = 1, 2, \cdots m$, $k = 1, 2, \cdots, p$, and p is the number of attribute groups.

Step 5: Weight Groups

For a given application, some attribute groups may be more important than the others. This is known to domain experts and application developers who are familiar with the application. For example, (a) a financial risk simulation for computing Value-at-Risk and ingests 2 GB of data may not have large storage related requirements but will need efficient computation and large I/O bandwidth between the processor and memory, hence, having a greater weight for the local communication group or (b) for a simulation requiring 500 Gigabyte space, the storage group is relevant. To capture this, after normalising, a weight for each attribute group is provided by the user, which is defined as W_k. In this paper, W_k can take values from 0 to 5, where 0 indicates that the group has no relevance for that application and 5 indicates that the group has the highest importance for the application.

Step 6: Rank Virtual Machines

The score of each VM is calculated as $S_i = \bar{G}_{i,k}.W_k = \sum\limits_{i'=1}^{i_{max}} \bar{r}_{i,i'}.Wk$, where max is the number of attributes in the attribute group. For attributes in the group where a higher value should denote a higher score, for example bandwidth or frequency, the weight used is Wk. On the other hand when a lower value should denote a higher score, for example latency, the weight used is $-Wk$.

The scores are ordered in a descending order for generating Rp_i which is the ranking of the VMs based solely on performance. The performance ranks are used in one of the validation techniques presented in Section 6.2.

An additional ranking of the VMs, Rc_i, based on both performance and costs is generated. To obtain this rank, C_i/S_i are ordered in ascending order, where C_i is the cost in \$/hour of the VM. The motivation for including costs in the methodology is determining the highest value-for-money VMs instead of merely finding the best performing VMs.

3 Cloud Virtual Machines

The Amazon Elastic Compute Cloud (EC2)[1] platform is chosen since it is publicly available and offers a variety of VMs with different performance capabilities. Different categories of previous generation EC2 VMs, referred to as instances, are offered as general purpose, memory optimised, cluster compute optimised, storage optimised, and GPU instances[2]. Table 1 shows the specification of the underlying hardware of the instances which are used for benchmarking (instances with more than 15 GiB are chosen to facilitate smooth running of the case study applications). The region and availability zones of the instances are US East N. Virginia (us-east-1) and US West Oregon (us-west-2).

TABLE 1
Amazon Elastic Compute Cloud (EC2) instances employed for benchmarking

Instance Type	Virtual CPUs (vCPU)	Mem. (GiB)	Processor	Clock Speed (GHz)	Cost/Hr ($) [18]
`m1.xlarge`	4	15.0	Intel Xeon E5-2650	2.00	0.480
`m2.xlarge`	2	17.1	Intel Xeon E5-2665	2.40	0.410
`m2.2xlarge`	4	34.2	Intel Xeon E5-2665	2.40	0.820
`m2.4xlarge`	8	68.4	Intel Xeon E5-2665	2.40	1.640
`m3.xlarge`	4	15.0	Intel Xeon E5-2670	2.60	0.500
`m3.2xlarge`	8	30.0	Intel Xeon E5-2670	2.60	1.000
`hi1.4xlarge`	16	60.5	Intel Xeon E5620	2.40	3.500
`hs1.4xlarge`	16	117.0	Intel Xeon E5-2650	2.00	4.600
`cc1.4xlarge`	16	23.0	Intel Xeon X5570	2.93	1.300
`cc2.8xlarge`	32	60.5	Intel Xeon X5570	2.93	2.400
`cr1.4xlarge`	32	244.0	Intel Xeon E5-2670	2.60	3.500

The general purpose instances are `m1` and `m3` instances, the memory optimised instances are `m2` and `cr1` instances, the compute optimised instances are `cc1` and `cc2` and the storage optimised instances are `hi1` and `hs1`. All instances are abstracted over their respective processors. Each virtual CPU (vCPU) of the `m3`, `cr1` and `cc2` instances is a hyperthread on a core of the underlying processor.

4 Cloud Benchmarks

The experimental setup for obtaining the attributes of VMs by benchmarking and then grouping them are presented in this section. The attributes $r_{i,j}$ of Step 3 in Section 2 are obtained and then grouped to obtain $G_{i,k}$.

[1] http://aws.amazon.com/ec2/
[2] https://aws.amazon.com/ec2/previous-generation/

4.1 Setup

Three tools, namely `bonnie++`, `lmbench` and `sysbench` are employed for benchmarking. The `bonnie++`[3] tool is used for file system benchmarks. The time and latency for reading data from file and writing data to file, sequential and random create, read and write operations, and number of seeks performed in a second can be benchmarked.

The `lmbench`[4] tool provides latency and bandwidth information on top of a wide range of memory and process related information [19]. Context switching times and VMs latencies are also obtained from this tool.

The `sysbench`[5] tool, commonly referred to as the Multi-threaded System Evaluation Benchmark, is also used for obtaining benchmark metrics related to the CPU and the file I/O performance. It is popular for understanding a system under data intensive loads.

All experiments to gather the benchmark metrics considered in the following section were performed eight times consecutively during a seven week period in September and October 2013. The instances with over 23 GiB memory could not be easily benchmarked. This is because file sizes used for benchmarking need to be at least twice as large as the memory to avoid caching of the file in memory which produces incorrect benchmark metrics. For this reason the size of the files used for benchmarking were up to 750 GiB nearly times the size of the highest memory for `bonnie++` and `sysbench` to obtain more accurate results. A number of instances cannot accommodate large files due to the limited storage available. Additional Amazon Elastic Block Storage (EBS)[6] volumes had to be attached to the instances to successfully benchmark the instances.

Table 2 shows the time and costs incurred for running the benchmarks on different VMs. This highlights the overhead not only on the time spent for obtaining the benchmarks but also on the total costs for using the resources, including the instances and storage.

TABLE 2
Time and cost for executing benchmarks

VM	Time (hrs) × Cost ($/hour)	Cost ($) for 8 executions
m1.xlarge	2.5 × 0.48	9.6
m2.xlarge	2.5 × 0.41	8.2
m2.2xlarge	5.0 × 0.82	32.8
m2.4xlarge	10.0 × 1.64	131.2
m3.xlarge	2.0 × 0.5	8.0
m3.2xlarge	2.5 × 1.0	20.0
hi1.4xlarge	7.5 × 3.5	210.0
hs1.8xlarge	14.0 × 4.6	515.2
cc1.4xlarge	5.0 × 1.3	52.0
cc2.8xlarge	6.0 × 2.4	115.2
cr1.4xlarge	10.0 × 3.5	280.0

4.2 Attributes and Groups

Two groupings are used to combine the attributes gathered from the cloud. The first provides an aggregated view of the attributes, whereas the second provides a more detailed view of the benchmarks as shown in Table 3. The aggregate set G_{agg}, comprises four groups, denoted as $G_{agg} = \{G_1, G_2, G_3, G_4\}$. The fine grain set G_{sub}, in which each aggregate group is decomposed into two sub-groups, is denoted as $G_{sub} = \{G_{1,1}, G_{1,2}, G_{2,1}, G_{2,2}, G_{3,1}, G_{3,2}, G_{4,1}, G_{4,2}\}$. Figure 1 shows the values obtained for a set of sample attributes. A low value of latencies and operation time and a high value of bandwidth and operations in one second indicates good performance of a VM.

TABLE 3
Aggregate and Fine-grain Groups

Group	Description	
G_1	Memory and process	
	$G_{1,1}$	Process latency
	$G_{1,2}$	Memory latency
G_2	Local communication	
	$G_{2,1}$	Local communication latency
	$G_{2,2}$	Local communication bandwidth
G_3	Computation	
	$G_{3,1}$	Integer operations
	$G_{3,2}$	Floating point operations
G_4	Storage	
	$G_{4,1}$	File I/O bandwidth
	$G_{4,2}$	File I/O frequency

4.2.1 *Memory and Process Group*

This group, denoted as G_1 captures the performance and latencies of the memory and process related operations. The L1 and L2 cache latencies of the instances are shown in Figure 1(a) and the main memory and random memory latencies of the instances are shown in Figure 1(b).

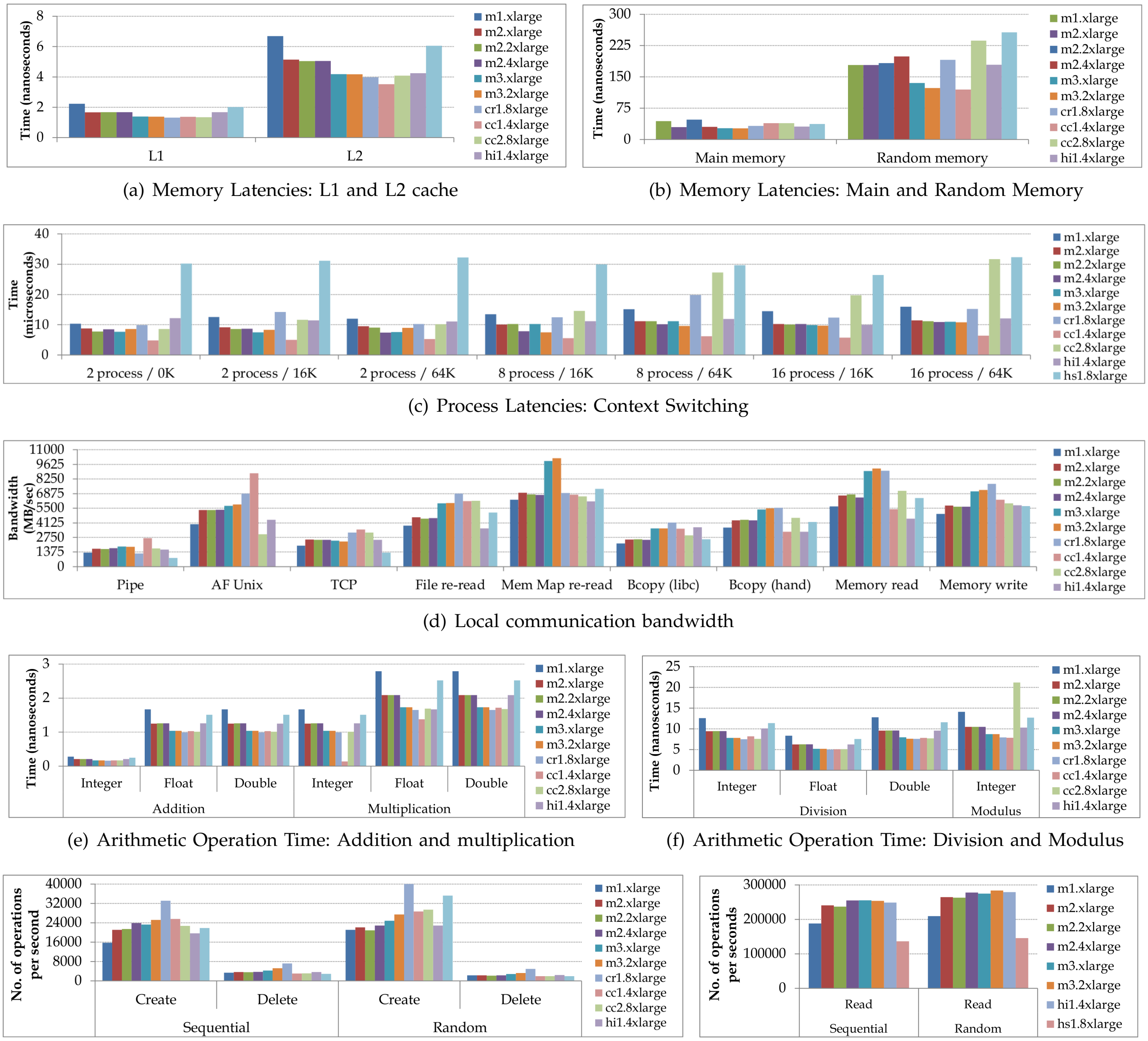
Context switching is computationally expensive and with tens or hundreds of switches per second a substantial cost is added to the processor. The latencies for context switching 2, 8 and 16 processes with 16 and 64 kilobyte sizes are shown in Figure 1(c) (For example, in the figure, 2 process / 64K indicates that the latency of context switching two processes is noted and the size of each process is 64K[7])

The aggregate group is divided into $G_{1,1}$ and $G_{1,2}$ sub-groups that capture all the process latencies and memory latencies respectively.

4.2.2 *Local Communication Group*

The bandwidth of both memory and interprocess communications are captured under the local communication group, denoted as G_2. Figure 1(d) shows

[3] http://sourceforge.net/projects/bonnie/
[4] http://lmbench.sourceforge.net/
[5] http://sysbench.sourceforge.net/
[6] http://aws.amazon.com/ebs/
[7] http://www.bitmover.com/lmbench/lat_ctx.8.html



(a) Memory Latencies: L1 and L2 cache

(b) Memory Latencies: Main and Random Memory

(c) Process Latencies: Context Switching

(d) Local communication bandwidth

(e) Arithmetic Operation Time: Addition and multiplication

(f) Arithmetic Operation Time: Division and Modulus

(g) File I/O Operations in one Second: Sequential and random create and delete

(h) File I/O Operations in one Second: Sequential and random read

Fig. 1. Sample benchmarks obtained from 11 Amazon Cloud VMs. Low value of latencies and operation time and high value of bandwidth and operations in one second indicates good performance of VMs.

memory communication metrics, namely the rate (MB/sec) at which data can be read from and written to memory, and interprocess communication metrics, namely the rate of data transfer between pipes and sockets.

The latencies in local communication are grouped as $G_{2,1}$ and the associated bandwidth under $G_{2,2}$.

4.2.3 Computation Group

The attributes captured in this group, denoted as G_3, are for benchmarking the performance of integer, single precision and double precision float operations such as addition and multiplication (refer Figure 1(e)) and division and modulus (refer Figure 1(f)). The computation benchmarks highlight consistently good performance of the `m3`, `cg1`, `cc1` and `cr1` instances across integer, float and double operations.

The integer operations and floating point operations are grouped into $G_{3,1}$ and $G_{3,2}$ respectively.

4.2.4 Storage Group

File I/O related attributes are grouped as the storage group, denoted as G_4, which considers the number of sequential create, read and delete and random create, read and delete operations as shown in Figure 1(g) and Figure 1(h). Overall, the best performer given the benchmarks obtained is the `cr1` instance. The `m3` instance is not too far behind in the file I/O performance.

The attributes that describe bandwidth and frequency of the file I/O operations are grouped as $G_{4,1}$ and $G_{4,2}$ respectively.

5 Cloud Rankings

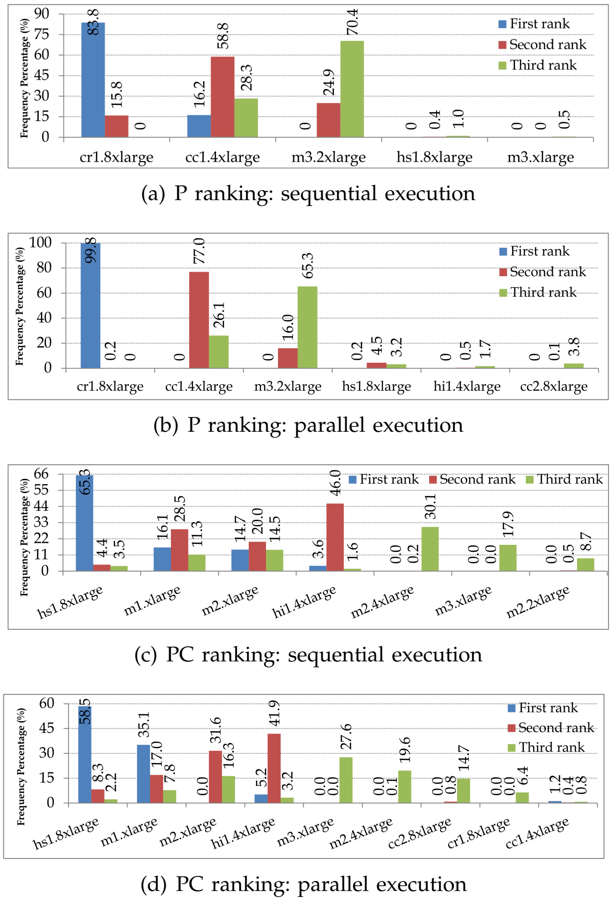
Given the two group sets G_{agg} and G_{sub} considered in the previous section, a user can provide a weight, W_k, for each group G_k. The set of weights corresponding to G_{agg} is $W_{agg} = \{W_1, W_2, W_3, W_4\}$ and the set of weight for G_{sub} is $W_{sub} = \{W_{1,1}, W_{1,2}, W_{2,1}, W_{2,2}, W_{3,1}, W_{3,2}, W_{4,1}, W_{4,2}\}$. The weights assigned by a user represents the importance of a group with respect to an application the user wants to deploy on a cloud instance. Each weight can take values between 0 and 5, where 0 signifies that the group represented by the weight has no relevance to the application and 5 indicates that the group is important to the application for achieving maximum performance.

Consider for example, an application that is memory intensive requiring limited storage and the user wants to describe the application using a small set of weights. Aggregate weights can be chosen and W_1 can be set to 5 for representing the relevance of memory group and W_4 can be set to 0 for describing the irrelevance of the storage group. However, a user may understand the application in detail and opt for the larger weight set to describe the application. Now, if the memory latencies are more important than the process latencies, then $W_{1,2}$ is set to 5 and $W_{1,1}$ may be set to a lower value. W_{sub} is therefore useful in providing a more detailed description of the application to the benchmarking methodology.

Each weight can take six possible values (0-5), and therefore there are 6^4 and 6^8 different combination of weights possible for W_{agg} and W_{sub} respectively. However, a set with all zero values is of no real significance, and hence, the total number of different combinations of aggregate weights is $1,295$ and of fine grain weights is $1,679,615$. Using the benchmarking methodology all possible rankings for different weight combinations were generated. Performance (P) and performance-cost (PC) rankings for sequential and parallel execution were considered. When the performance of parallel execution is to be evaluated the methodology takes the number of vCPUs of the instance into account.

5.1 Aggregate Weight Space

Figure 2 shows the frequency of the instances appearing in the top three ranks in the aggregate weight space. When the P ranks are taken into account, it is evident that there are winners for the first, second and third ranks (refer Figure 2(a) and Figure 2(b)). This indicates that the benchmarking methodology is not highly sensitive to small variations in the weight combinations. So given an application that a user has limited knowledge about, the methodology is likely to produce ranks that will provide maximum performance.



(a) P ranking: sequential execution

(b) P ranking: parallel execution

(c) PC ranking: sequential execution

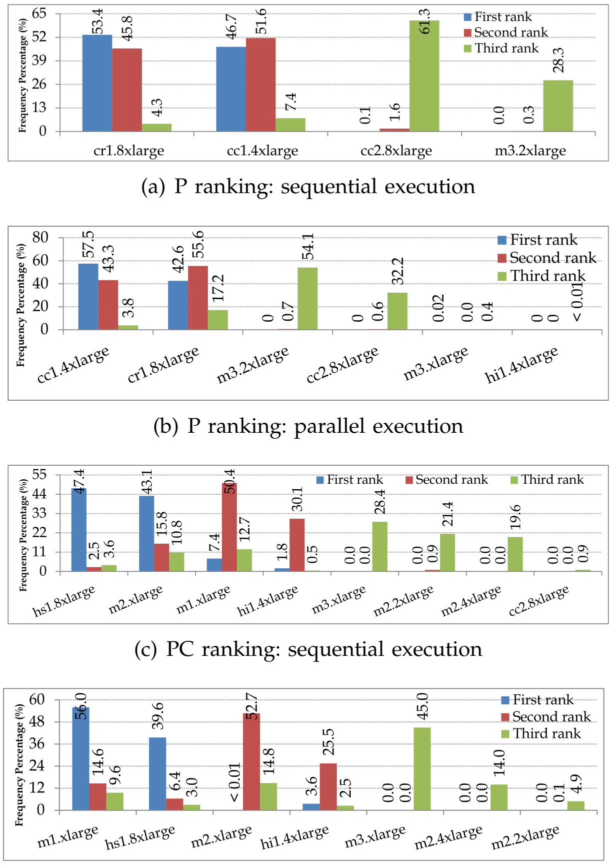
(d) PC ranking: parallel execution

Fig. 2. Frequency of instances appearing in the top 3 ranks for 1,295 combination of aggregate weights

When cost is taken into account with performance (PC), there is a more uniform spread across the top three ranks (refer Figure 2(c) and Figure 2(d)). We observe the following: (a) four instances appear in the first and second ranks in contrast to the former case with only one or two instances in the first and second ranks, (b) at least two instances that achieve first ranks have a good chance of winning, (c) two instances, `hs1` and `hi1` have better chances of winning when cost is considered, and (d) the `cr1` instance which is a clear winner in the P ranking has only a small chance of winning when cost is taken into account. The chances of winning are shared between instances when cost is accounted for. This makes the winning instances less obvious intuitively and the weights are relatively more sensitive compared to the P ranks.

5.2 Fine Grain Weight Space

Figure 3 shows the frequency of the instances appearing in the top three ranks in the fine grain weight space. Unlike the aggregate weight space, there are



(a) P ranking: sequential execution

(b) P ranking: parallel execution

(c) PC ranking: sequential execution

(d) PC ranking: parallel execution

Fig. 3. Frequency of instances appearing in the top 3 ranks for 1,679,615 combination of fine grain weights

more winners in this case. For example, the first and second ranks are almost equally shared between `cr1` and `cc1` instances and the third rank by `cg1` for sequential and parallel execution (refer Figure 3(a) and Figure 3(b)). As expected, the methodology produces a wider range of rankings when a larger number of groups and associated weights are considered.

The number of competing instances increases when cost is taken into account (refer Figure 3(c) and Figure 3(d)). At least two instances have a good chance of winning the first rank and at least three instances occupy the second rank. The third rank instances generally do not win first or second positions. The key difference from the aggregate weight space is the wide spread of winning instances which can be selected by the benchmarking methodology. There is a greater sensitivity to the set of weights supplied by the user but at the same time facilitates the selection of instances, which are both performance and cost effective, for an application.

6 Validation Study

In this section, we examine the benchmarking methodology for case study applications using validation techniques.

6.1 Case Studies

The benchmarking methodology we have proposed is suitable to predict the performance of embarrassingly parallel high-performance computing applications that execute on the multiple cores of the same VM. Three such case studies are chosen for validating the benchmarking methodology which can be executed on multiple VMs. However, in this research, we execute the applications on all virtual cores available on a single VM. The first application is a simulation used in the financial risk industry, the second application is a molecular dynamics simulations used by theoretical physicists, and the third is a mathematical solver employed in scientific applications. The applications have different requirements and are therefore chosen for validating the methodology.

6.1.1 Case Study 1: Financial Risk Analysis

Aggregate Risk Analysis [21], [22] is a simulation employed for computing key risk metrics such as Probable Maximum Loss (PML) [23]. The simulation is developed using C++ and parallelised on the CPU cores using OpenMP. The Boost library is used for implementing mathematical and statistical functions. The inputs to the simulation are one million catastrophic event trials and a collection of thousands of events and their corresponding losses which are obtained from the disk onto memory. A number of financial terms are applied to the loss associated with an event and aggregated to provide the PML for a contractual year. The simulation is embarrassingly parallel and can be executed on the virtual cores by sharing the input data between the threads on the core.

The simulation is memory intensive with numerous read and write operations and at the same time computationally intensive requiring a large number of float operations to be performed both to compute the risk metrics. The local communication between processes are less relevant. The simulation requires data transfer from the disk to memory initially making file operations moderately relevant.

6.1.2 Case Study 2: Molecular Dynamic Simulation

The second case study is a molecular dynamics simulation of short range interactions used by theoretical physicists of a complex system comprising N particles [24], [25]. The simulation is developed using C++ and OpenMP is leveraged for parallelism on the cores of the CPU. The simulation computes the trajectory of N particles and the forces they exert by solving a system of differential equations discretized into different time steps. In each time step, the position, velocity, the kinetic and potential energies of each particle are generated. It is assumed that if particles collide then they pass through each other. The simulations are performed on a three dimensional space for 10,000 particles and 200 time steps.

The simulation is computationally intensive followed by the memory and processor requirements along with the need for local communication. There are no file intensive operations in this simulation.

6.1.3 Case Study 3: Block Triagonal Solver

The Block Triagonal Solver, otherwise referred to as BT is a NASA Advanced Supercomputing (NAS) Parallel Benchmark (NPB) [26], version 3.3.1[8], which is used as a third case study. The BT solver is a pseudo application benchmark that represents a sample mathematical solver used in scientific applications. The Class C problem (defines the size and parameters of the problem) of the solver is used. Sequential and parallel programming models (in this paper, OpenMP is used) are used for an empirical analysis on the cloud VMs using NPB. The solver uses a grid size of $162 \times 162 \times 162$ for 200 iterations.

The solver is numerically intensive and memory and processor related operation is relevant, but does not take a precedence over computations. Local communications and file operations have little effect on the solver.

6.2 Validation Techniques

Two validation techniques are used to evaluate the benchmarking methodology. The first technique compares the ranking produced by the methodology and the ranking generated from empirical analysis of the application, which we refer to as 'Comparative Validation'. In the second technique, the entire space of weights is taken into account to make observations when comparing the aggregate and fine-grain weight spaces, which we refer to as 'Enumeration-based Validation'.

6.2.1 Comparative Validation

This technique comprises the following three steps:

Step 1 - Application Benchmarking: the case study applications are used for an empirical analysis to validate the cloud benchmarking methodology. The application is firstly executed on all VMs. The time taken by an application to complete execution can be used as one criterion for evaluating performance.

Step 2 - Application-based Cloud Ranking: based on the performance of the VMs in the empirical analysis they are ranked. For example, in this paper, the ranks from the empirical analysis are based on performance evaluated in terms of the time taken for completing execution. Additional criterion such as the quality of result can be used if it is applicable to the application.

The values for each criterion for evaluating performance are normalised using $\bar{v}_{i,j} = \frac{v_{i,j} - \mu_j}{\sigma_j}$, where μ_j is the mean value of $v_{i,j}$ over m VMs and σ_j is the standard deviation $v_{i,j}$ over m VMs. The normalised values are used to rank the VMs Mp_i. If multiple criteria are used then a rank for each criterion is obtained. The multiple ranks then need to be aggregated into a single rank.

Step 3 - Cloud Ranks Comparison: the ranks R_i from the benchmarking methodology are compared against the validation ranks M_i. This comparison can be used for selecting one or more cloud resources that can maximise the performance of the application. If there are significant differences between the rankings of the VMs then the requirements of the application are re-evaluated and a different set of weights need to be assigned to the attribute groups.

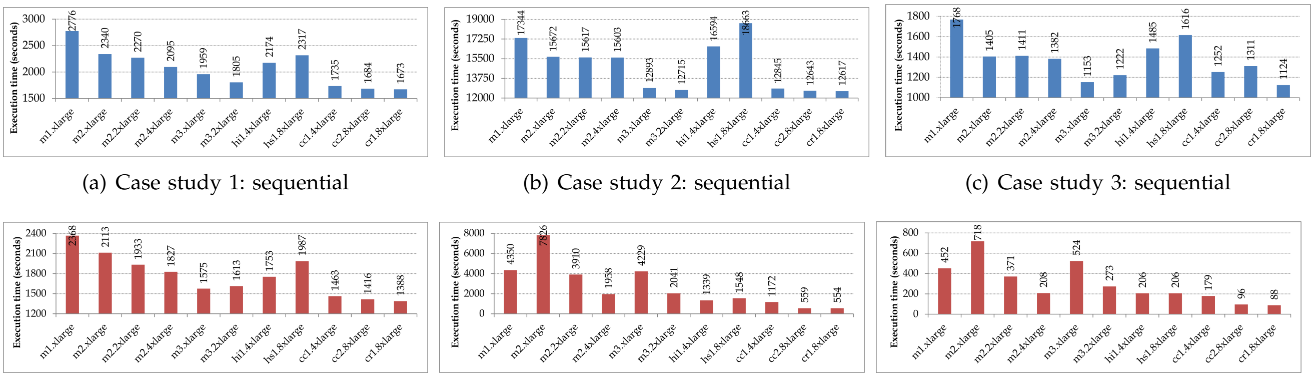
The three case study applications were executed on all instances shown in Table 1. The mean time taken to execute the applications ten times were obtained. Figure 4 shows the time taken for the sequential execution and parallel execution (the case study applications are executed on all the cores of the VM; 2 cores minimum and 32 cores maximum) of the applications respectively. For the P rankings, the instances are ranked based on their execution time; first rank for instance with lowest execution time. For the PC rankings, the ratio of the cost per hour of an instance (from Table 1) and the execution time (in this case is the measure of performance) is ordered in ascending order to generate the ranks. Again, the first rank is for the instance with the lowest ratio. The rankings of the methodology for each case study given a set of weights chosen in consultation with industry practitioners and domain experts were also generated. In this paper, the results using fine-grain rankings are considered. Aggregate weight-based ranking has been previously reported and $W_{agg} = \{5, 3, 5, 2\}$ for case study 1, $W_{agg} = \{4, 3, 5, 0\}$ for case study 2 and $W_{agg} = \{2, 0, 5, 0\}$ for case study 3 [3].

Table 4 shows the rankings for the financial risk application with $W_{sub} = \{3, 5, 3, 3, 2, 5, 2, 2\}$. There is a correlation of nearly 86% and 57% for sequential and parallel P ranks respectively. For the top four VMs (sequential performance) the benchmarking methodology points to the same instances observed as good performers in the empirical analysis. In the case of parallel performance we observe that the top four VMs are similar in both the benchmarking methodology and the empirical analysis. When costs are taken into account the correlation is improved to 93% and 91%.

Table 5 shows the rankings for the molecular dynamics simulation with $W_{sub} = \{3, 4, 3, 3, 2, 5, 0, 0\}$. There is a correlation of nearly 85% and over 67% for sequential performance and parallel P ranks respectively. For parallel performance the top five instances in the empirical analysis are pointed to by the benchmarking methodology.

Table 6 shows the rankings for the BT solver with $W_{sub} = \{2, 2, 0, 0, 5, 5, 0, 0\}$. There is 81% and 95% correlation for sequential performance. For parallel

[8] https://www.nas.nasa.gov/publications/npb.html



(a) Case study 1: sequential

(b) Case study 2: sequential

(c) Case study 3: sequential

(d) Case study 1: parallel

(e) Case study 2: parallel

(f) Case study 3: parallel

Fig. 4. Time taken for sequential and parallel execution (min 2 threads and max 32 threads) of the case studies

performance the rankings bear 95% correlation.

The percentage correlation (obtained by using the Pearson Product-Moment Correlation Coefficient) between the ranks obtained from the benchmarking methodology and the empirical analysis are shown in Table 7. There is high correlation between the ranks, which is an indication that the benchmarking methodology with fine-grain weights can produce P and PC ranks that are close to reality as verified through the case studies. The robustness of the methodology is confirmed through this validation exercise.

In the comparative validation method, the rankings calculated from the expert set of weights were compared against the empirical ranks. This method is limited for comparing the PC rankings in that two variables, namely time and cost, are used for obtaining the ranks. If any one of the variable has the same value, then the combination of performance-cost is not well captured. Additional variables will need to be captured to address this such that each VM is uniquely represented for a fairer benchmarking method than the method used to generate PC ranking in this paper. Nevertheless, additional variables will result in a more lengthy benchmarking process.

6.2.2 Enumeration-based Validation

Comparative validation cannot explain the distribution of rankings in the entire space of different sets of weights. Therefore, we present an exhaustive enumeration method that takes the entire space of weights into account.

The space of $1,295$ sets of aggregate weights and $1,679,615$ sets of fine grain weights are considered. The corresponding ranking of each set of weights is generated using the benchmarking methodology. A scoring mechanism is used for each calculated rank against the empirical ranks. A weighted hamming distance is used for ranking in Algorithm 1.

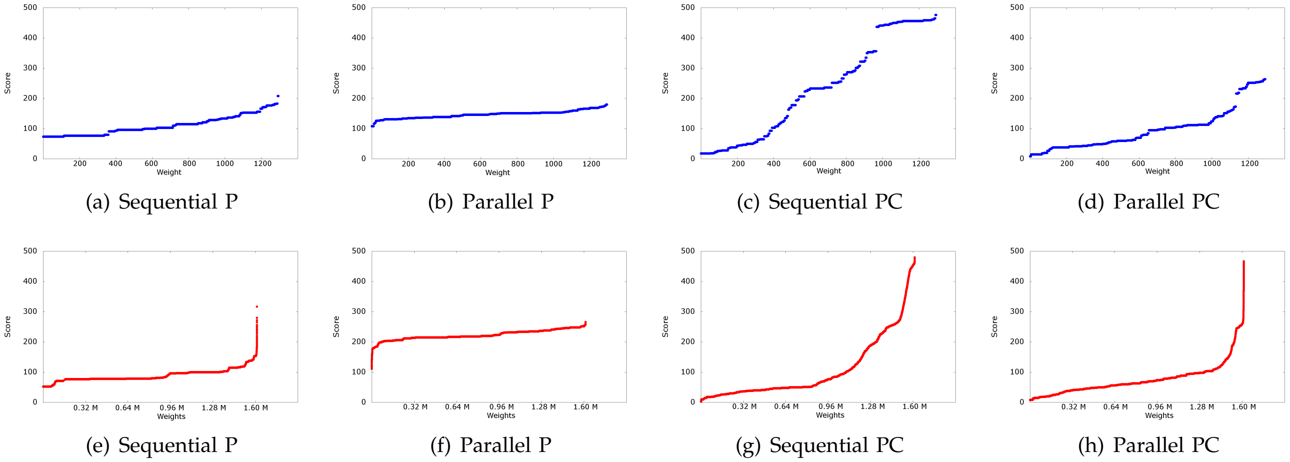
The input to the algorithm are two rankings - ER, which is the empirical rank and CR, which is the calculated rank and m, the number of cloud VMs. The distance between the empirical and calculated rank is computed as the sum of point-wise distances.

Algorithm 1 Weighted hamming distance scores

```
procedure DISTANCE–SCORES($ER$, $CR$, $m$)
    score = 0
    for each $i$ $R \in ER$ do
        $C = m - R + 1$
        $R' = rankOf(CR, i)$
        $score+ = C * |R - R'|$
    end for
end procedure
```

For example, consider that there are 11 VMs to be ranked and assume the empirical rank of one VM to be 4 and the calculated rank of the same VM to be 2, then the distance between them is 2. The coefficient C is $11-4+1 = 8$. The contribution to the $score$ is $8*2 = 16$. Assume the empirical rank of another VM to be 10 and the calculated rank to be 8, now the distance is again 2. However, the coefficient is significantly lower, $C = 11 - 10 + 1 = 2$ and its contribution to the score is 4. Using such a mechanism we ensure that VMs ranked closer to the top in the empirical ranking are given higher weights in comparison to those ranked lower. The weight decay is linear and varies between 1 and the number of VMs considered, in our case $m = 11$. Smaller scores indicate that the calculated rank is closer to the empirical rank.

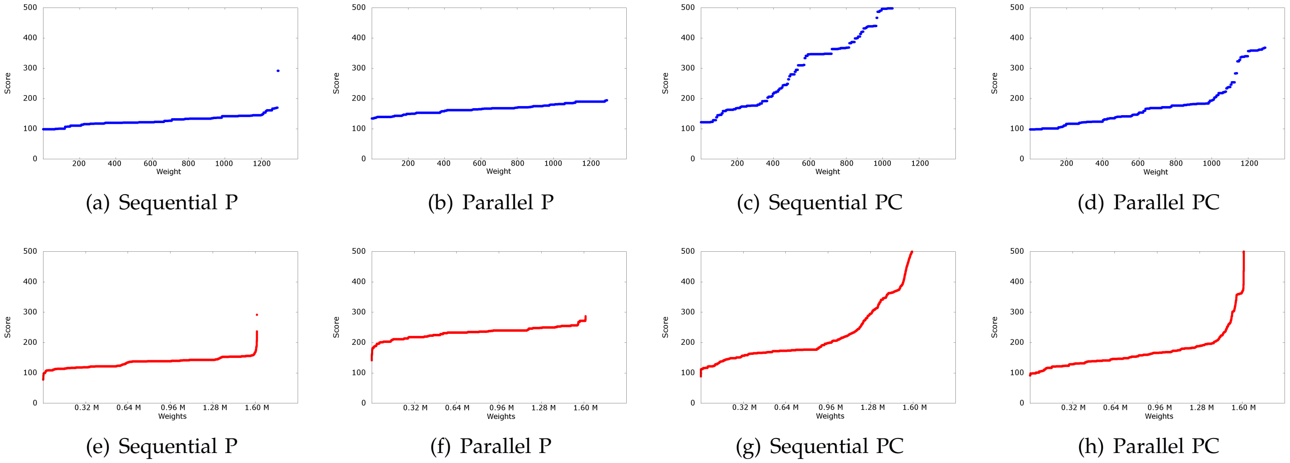
Figure 5, Figure 6 and Figure 7 show the space of different sets of weights and the scores of their rankings on the Y-axis. The X-axis are the different weights sorted by the scores. We make the following observations: (i) different set of aggregate weights produce rankings of distinct quality. This evidences that weights can discriminate between good and bad performance of VMs, (ii) the weights are more significant when costs are taken into account together with performance. The scores of the ranks are more diverse resulting in a discrimination of the weight space, (iii) using fine-grain weights results in increased number



(a) Sequential P (b) Parallel P (c) Sequential PC (d) Parallel PC

(e) Sequential P (f) Parallel P (g) Sequential PC (h) Parallel PC

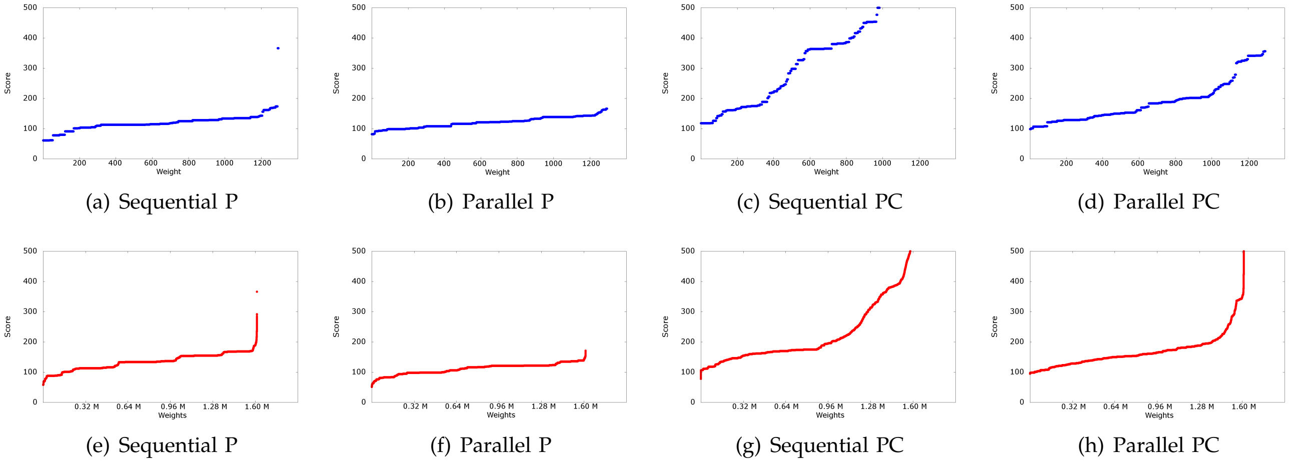
Fig. 5. Case Study 1 - Aggregate weights are shown in Figure 5(a) to Figure 5(d) and fine-grain weights are shown in Figure 5(e) to Figure 5(h). X-axis is the weights ordered by scores shown in the Y-axis.



(a) Sequential P (b) Parallel P (c) Sequential PC (d) Parallel PC

(e) Sequential P (f) Parallel P (g) Sequential PC (h) Parallel PC

Fig. 6. Case Study 2 - Aggregate weights are shown in Figure 6(a) to Figure 6(d) and fine-grain weights are shown in Figure 6(e) to Figure 6(h). X-axis is the weights ordered by scores shown in the Y-axis.



(a) Sequential P (b) Parallel P (c) Sequential PC (d) Parallel PC

(e) Sequential P (f) Parallel P (g) Sequential PC (h) Parallel PC

Fig. 7. Case Study 3 - Aggregate weights are shown in Figure 7(a) to Figure 7(d) and fine-grain weights are shown in Figure 7(e) to Figure 7(h). X-axis is the weights ordered by scores shown in the Y-axis.

TABLE 4
Fine-grain Rankings for Case Study 1, $W_{sub} = \{3, 5, 3, 3, 2, 5, 2, 2\}$

Amazon Instance	Sequential P Rankings		Parallel P Rankings		Sequential PC Rankings		Parallel PC Rankings	
	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis
m1.xlarge	10	11	11	11	2	1	1	2
m2.xlarge	9	10	9	10	1	2	2	1
m2.2xlarge	8	8	8	8	3	4	3	5
m2.4xlarge	7	6	7	7	9	7	5	6
m3.xlarge	5	5	6	4	4	3	4	3
m3.2xlarge	3	4	3	5	5	5	6	4
hi1.4xlarge	6	7	4	6	8	8	9	8
hs1.8xlarge	11	9	10	9	10	11	11	11
cc1.4xlarge	2	3	1	3	6	6	8	7
cc2.8xlarge	4	2	5	2	7	9	10	10
cr1.8xlarge	1	1	2	1	11	10	7	9

TABLE 5
Fine-grain Rankings for Case Study 2, $W_{sub} = \{3, 4, 3, 3, 2, 5, 0, 0\}$

Amazon Instance	Sequential P Rankings		Parallel P Rankings		Sequential PC Rankings		Parallel PC Rankings	
	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis
m1.xlarge	11	10	10	10	1	2	2	2
m2.xlarge	7	8	9	11	2	1	1	1
m2.2xlarge	9	7	8	8	4	4	4	4
m2.4xlarge	8	6	6	6	7	7	6	6
m3.xlarge	4	5	4	9	3	3	3	3
m3.2xlarge	3	3	7	7	5	5	5	5
hi1.4xlarge	6	9	3	4	10	9	10	8
hs1.8xlarge	10	11	11	5	8	10	8	9
cc1.4xlarge	2	4	5	3	6	6	7	7
cc2.8xlarge	5	2	1	2	9	8	9	10
cr1.8xlarge	1	1	2	1	11	11	11	11

TABLE 6
Fine-grain Rankings for Case Study 3, $W_{sub} = \{2, 2, 0, 0, 5, 5, 0, 0\}$

Amazon Instance	Sequential P Rankings		Parallel P Rankings		Sequential PC Rankings		Parallel PC Rankings	
	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis	From Benchmarking	From Empirical Analysis
m1.xlarge	11	11	11	11	1	1	2	3
m2.xlarge	7	8	8	10	2	2	1	1
m2.2xlarge	9	7	9	9	5	4	4	4
m2.4xlarge	8	6	7	7	7	7	7	7
m3.xlarge	5	2	5	6	4	3	3	2
m3.2xlarge	4	3	4	4	6	5	5	5
hi1.4xlarge	6	9	6	5	9	9	10	8
hs1.8xlarge	10	10	10	8	10	10	9	9
cc1.4xlarge	1	4	1	2	3	6	6	6
cc2.8xlarge	3	5	3	3	8	8	8	10
cr1.8xlarge	2	1	2	1	11	11	11	11

of options that are not available using aggregate weights, and (iv) fine-grain weights can produce better ranking options than aggregate weights.

7 Related Work

Benchmarking is performed using a set of standard tests for evaluating the relative performance of a computing resource [27], [28]. For example, Linpack is used to evaluate the performance of supercomputers for ranking the Top500[9] list [29]. Similar techniques can be employed to benchmark the cloud [2], [30], [31]. Cloud benchmarking considers the evaluation of the resources and the services [32], [33].

Unlike systems like the cluster or grid, the user has access to an abstract computing resource (limited access to the underlying hardware) on the cloud. Hence, it is important to understand the VM through resource benchmarking for obtaining maximum performance

[9] http://www.top500.org/

TABLE 7
Percentage correlation of rankings obtained using fine-grain weights from benchmarking methodology and empirical analysis

Case Study	Sequential P rank	Parallel P rank	Sequential PC rank	Parallel PC rank
1	93	87	93	91
2	85	67	96	97
3	81	95	95	95

when an application is deployed. Performance of the memory accessible on the VM, processor capabilities such as computations performed, and file operations are taken into account. Such benchmarks are useful when a user wants to exploit multiple CPUs offered by cloud VMs.

Service benchmarking provides insight into the reliability and variability of cloud services [34]. For example, network performance between cloud resources is crucial for workflows or web services [4], [35], [36], [37], [38], [39]. Sometimes, a hybrid of resource and service benchmarking is considered [40], [41]. In this paper, we assume that reasonable and seamless service is obtained on the public cloud and therefore only resource benchmarking is considered.

Current resource benchmarking research on the cloud is limited in three ways. Firstly, research reported in relevant literature considers a small sample of low-cost VMs [42], [43]. In reality, a wide variety of VMs with varying performance and cost are available from a cloud provider which are not taken into account. The cloud infrastructure has rapidly matured over the last few years and consequentially, there has been a significant improvement in the performance of the VMs offered on the cloud. In this paper, we have considered a variety of resources with different performance capabilities offered by the same provider.

Secondly, the requirements of applications that need to be deployed on the cloud are seldom mapped on to benchmarks (for example, [34]). Benchmarking without meaningfully interpreting the results based on the requirements of an application cannot be useful to a user [2], [31]. In this paper, the proposed benchmarking methodology considers the requirements of the application and the user assigns weights of importance to cloud attributes that best describes the requirements. The result is a set of VMs on which maximum performance can be achieved.

Thirdly, the benchmarking techniques need to incorporate methods to validate the benchmarks. This is an important issue to guarantee that the benchmarks obtained are acceptable. While a number of other issues related to benchmarking are addressed there is minimal focus in this direction (for example, [40], [43]). Empirical analysis is a straightforward way for validating benchmarks. While research reported in [13], [14], [15], [16], [17], [11], [12] perform empirical analysis on the cloud, the results are not employed for validating the benchmark results. In this paper, we employ a validation technique in which the benchmarks are validated through an empirical analysis using case study applications.

The research in this paper was motivated to address the above three challenges. For this, the cloud benchmarking methodology was developed, and in this research we have demonstrated the methodology on a set of resources with different performance capabilities, taken into account user requirements of an application, and validated the benchmarks using sample applications. The memory and process, local communication, computation and storage requirements of the application are known beforehand. These high-performance computing applications are embarrassingly parallel and executed on the multiple cores of a single VM.

The state-of-the-art is advanced by developing an application aware benchmarking methodology that captures a wide variety of requirements of an application to generate performance and cost based rankings of cloud VMs. The methodology has been rigorously validated in multiple ways for three scientific HPC applications. In contrast to our previous paper [3], the research presented in this paper improves on the benchmarking methodology by (i) generating both performance and cost based rankings, (ii) capturing the requirements of an application in a fine-grain manner using a set of eight weights, and (iii) validating the weight space using an enumeration technique, all of which were not considered previously.

8 Conclusions

In this paper, we proposed a six step benchmarking methodology for cloud VMs. We hypothesized that by considering the requirements of an application provided by a user as a set of weights (aggregate weights or fine-grain weights), along with benchmarking data collected from the cloud using the proposed methodology, VMs can be ranked in order of performance and cost effectiveness so that a user can deploy an application on a cloud VM, which will maximise performance. The hypothesis was validated on three real-world case study applications, namely a financial risk simulation, a molecular dynamics simulation and a mathematical solver, using comparative validation and enumeration-based validation techniques. The results show a good correlation between benchmarked and empirical rankings.

The class of problems we have targeted in this paper are scientific HPC applications. They are usually developed and maintained by a large community and there is a lot of knowledge on their requirements. As an application evolves over time some requirements may vary, but these are generally known to the developers. The weights we determined for the applications were in consultation with experts and developers from each of these communities which was readily known to them.

The benchmarking methodology we proposed can be used until a user is satisfied with the rankings generated. Once benchmarking is performed the entire weight space and rankings can be generated. The user can then provide different sets of weights (if not sure about certain weights) to obtain different rankings. Then the user can simply select the VMs which may appear in all the top ranks for different weights provided to ensure that performance is maximised.

Benchmarking the whole VM with large memory and storage is only possible at the expense of monetary costs and time. The benchmarking method proposed in this paper is limited in that way since it benchmarks a whole VM and thereby cannot be used in real-time. However, the benchmark data obtained from the method can be used with real-time methods. We are currently investigating container as a mechanism for benchmarking a small portion of a VM to reduce the time and costs required for cloud benchmarking to bypass the above limitation. Preliminary experiments indicate that container-based benchmarking can give similar results to benchmarking the whole VM. Using this method saves both time and money spent for benchmarking.

In the future, we intend to incorporate benchmarking useful for applications that span across multiple nodes and leverage multi-core systems with complex memory hierarchies, such as GPUs.

Acknowledgments

This research was pursued under the EPSRC grant, EP/K015745/1, a Royal Society Industry Fellowship and an AWS Education Research grant.

References

[1] Z. Li, L. O'Brien, H. Zhang and R. Cai, "On a Catalogue of Metrics for Evaluating Commercial Cloud Services," 13th International Conference on Grid Computing, 2012, pp.164-173.

[2] A. Iosup, R. Prodan and D. Epema, "IaaS Cloud Benchmarking: Approaches, Challenges, and Experience," Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, 2013.

[3] B. Varghese, O. Akgun, I. Miguel, L. Thai and A. Barker, "Cloud Benchmarking for Performance," Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science, 2014, pp. 535-540.

[4] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," Proceedings of the 1st ACM Symposium on Cloud Computing, 2010, pp. 143-154.

[5] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini and R. Biswas, "Performance Evaluation of Amazon EC2 for NASA HPC Applications," Proceedings of the 3rd workshop on Scientific Cloud Computing, 2012, pp. 41-50.

[6] J. J. Rehr, F. D. Vila, J. P. Gardner, L. Svec and M. Prange, "Scientific Computing in the Cloud," IEEE Computing Now, Vol. 12, No. 3, 2010, pp. 34-43.

[7] O. Niehorster, A. Brinkmann, G. Fels, J. Kruger and J. Simon, "Enforcing SLAs in Scientific Clouds," Proceedings of the IEEE International Conference on Cluster Computing, 2010, pp. 178-187.

[8] L. Gillam, B. Li and J. O'Loughlin, "Benchmarking Cloud Performance for Service Level Agreement Parameters," International Journal of Cloud Computing, Volume 3, No. 1, 2014, pp. 3-23.

[9] L. Gillam, B. Li, J. O'Loughlin and A. P. Tomar, "Fair Benchmarking for Cloud Computing Systems," Journal of Cloud Computing: Advances, Systems and Applications, Volume 2, No. 6, 2013.

[10] E. Kreyszig, "Advanced Engineering Mathematics," 8th Edition, John Wiley & Sons, 2000.

[11] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing," USENIX Login, Vol. 33, No. 5, 2008, pp. 18-23.

[12] A. Gupta and D. Milojicic, "Evaluation of HPC Applications on Cloud," HP Technical Reports, HPL-2011-132, 2011.

[13] T. Rabl, M. Frank, H. M. Sergieh and H. Kosch, "A Data Generator for Cloud-scale Benchmarking," Proceedings of the Second TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems, 2010, pp. 41-56.

[14] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia. J. Shalf, H. J. Wasserman and N. J. Wright, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, 2010, pp. 159-168.

[15] A. Gupta, L. V. Kale, D. S. Milojicic, P. Faraboschi, R. Kaufmann, V. March, F. Gioachin, C. H. Suen, B. -S. Lee, "Exploring the Performance and Mapping of HPC Applications to Platforms in the Cloud," Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing, 2012, pp. 121-122.

[16] Q. He, S. Zhou, B. Kobler, D. Duffy and T. McGlynn, "Case Study for Running HPC Applications in Public Clouds," Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, pp. 395-401.

[17] C. Luo, J. Zhan, Z. Jia, L. Wang, G. Lu, L. Zhang, C.-Z. Xu, N. Sun, "CloudRank-D: Benchmarking and Ranking Cloud Computing Systems for Data Processing Applications," Frontiers of Computer Science, Vol. 6, Issue 4, 2012, pp 347-362.

[18] Amazon Elastic Compute Cloud (EC2) instance pricing webpage: http://aws.amazon.com/ec2/pricing/ [Last accessed: 16 February 2014]

[19] C. Staelin, "lmbench - An Extensible Micro-benchmark Suite," Software - Practice and Experience, 2004.

[20] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, S. K. Weeratunga, "The NAS Parallel Benchmarks - Summary and Preliminary Results," Proceedings of the ACM/IEEE Conference on Supercomputing, 1991, pp. 158-165.

[21] A. K. Bahl, O. Baltzer, A. Rau-Chaplin, and B. Varghese, "Parallel Simulations for Analysing Portfolios of Catastrophic Event Risk," Workshop Proceedings of the International Conference of High Performance Computing, Networking, Storage and Analysis (SC), 2012.

[22] B. Varghese and A. Barker, "Are Clouds Ready to Accelerate Ad hoc Financial Simulations?" Proceedings of the International Symposium on Big Data Computing (BDC), 2014.

[23] G. Woo, "Natural Catastrophe Probable Maximum Loss," British Actuarial Journal, Vol. 8, 2002.

[24] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan and D. E. Shaw, "Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters," Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Article No. 84, 2006.

[25] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, K. Schulten, "Scalable Molecular Dynamics with NAMD," Journal of Computational Chemistry, Volume 26, Issue 16, 2005, pp. 1781-1802.

[26] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A.

Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, S. K. Weeratunga, "The NAS Parallel Benchmarks - Summary and Preliminary Results," Proceedings of the ACM/IEEE Conference on Supercomputing, 1991, pp. 158-165.

[27] K. Huppler, "The Art of Building a Good Benchmark," Performance Evaluation and Benchmarking (Editors: R. Nambiar and M. Poess), Lecture Notes in Computer Science, 2009, pp. 18-30.

[28] D. Kaeli and K. Sachs, "Computer Performance Evaluation and Benchmarking," Lecture Notes on Computer Science (LNCS 5419), Springer, 2009.

[29] J. J. Dongarra, P. Luszczek and A. Petitet, "The LINPACK Benchmark: Past, Present and Future," Concurrency and Computation: Practice and Experience, Vol. 15, 2013, pp. 803-820.

[30] B. Gregg, "Systems Performance: Enterprise and the Cloud," Prentice Hall, First edition, 2013.

[31] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl and C. Tosun, "Benchmarking in the Cloud: What it Should, Can, and Cannot Be," Selected Topics in Performance Evaluation and Benchmarking (Editors: R. Nambiar and M. Poess), Lecture Notes in Computer Science, 2013, pp. 173-188.

[32] V. Vedam and J. Vemulapati, "Demystifying Cloud Benchmarking Paradigm: An In Depth View," Proceedings of the 36th IEEE International Conference on Computer Software and Application, 2012.

[33] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware" Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, 2012.

[34] A. Lenk, M. Menzel, J. Lipsky, S. Tai and P. Offermann, "What are you paying for? Performance Benchmarking for Infrastructure-as-a-Service Offerings," Proceedings of the 4th International Conference on Cloud Computing, 2011, pp. 484-491.

[35] A. Iosup, N. Yigitbasi and D. Epema, "On the Performance Variability of Production Cloud Services," Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011, pp. 104-113.

[36] Y. Mei, L. Liu, X. Pu and S. Sivathanu, "Performance Measurements and Analysis of Network I/O Applications in Virtualised Cloud," Proceedings of the 3rd International Conference on Cloud Computing, 2010, pp. 59-66.

[37] A. Li, X. Yang, S. Kandula and M. Zhang, "CloudCmp: Comparing Public Cloud Providers," Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010.

[38] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox and D. Patterson "Cloudstone: Multi-Platform Multi-Language Benchmark and Measurement Tools for Web 2.0," Proceedings of Cloud Computing and its Applications, 2008.

[39] A. Li, X. Zong, S. Kandula, X. Yang and M. Zhang, "Cloud-Prophet: Towards Application Performance Prediction in Cloud," Proceedings of the ACM SIGCOMM 2011 conference, 2011, pp. 426-427.

[40] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware," Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, 2012.

[41] J. Schad, J. Dittrich, J. -A. Quiane-Ruiz, "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," Proceedings of the VLDB Endowment, Vol. 3, Issue 1-2, 2010, pp. 460-471.

[42] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," IEEE Transactions on Parallel and Distributed Systems, Vol. 22, Issue 6, 2011, pp. 931-945.

[43] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer and Dick Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," Cloud Computing, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2010, pp 115-131.

Blesson Varghese is a Research Fellow in Computer Science at the Queen's University Belfast. Previously, he worked at University of St Andrews and Dalhousie University. He obtained a PhD in Computer Science (2011) and MSc in Network Centred Computing (2008), both from the University of Reading, on international scholarships. His research interests are in developing and analysing parallel and distributed systems. More information is available from www.blessonv.com.

Ozgur Akgun is a Research Associate at the School of Computer Science, University of St Andrews. He received BSc in Computer Science from the Izmir University of Economics (2009) and PhD in Computer Science from the University of St Andrews (2014). His research focus is on high level modelling languages for constraint programming and their use for automated modelling. He is interested in applications of constraint programming in cloud computing and astronomy.

Ian Miguel is a professor of Computer Science at the University of St Andrews. He has researched Constraint Programming throughout his career. His PhD received the BCS/CHPC Distinguished Dissertation award, and between 2004 and 2009 he held a Royal Academy of Engineering/EPSRC Research Fellowship. Recently, he has focused on automated constraint modelling and constructing efficient constraint solvers. Ian is Principal Investigator of UK EPSRC EP/K015745/1, having previously been PI or CI of four other EPSRC grants and an EPSRC CASE for New Academics award, sponsored by Microsoft Research, totalling nearly GBP 3M.

Long Thai is a PhD student at School of Computer Science, St Andrews University. He received his BEng in Information Technology from Ho Chi Minh City International University, and his MSc in Internet and Distributed Systems from Durham University. His research focuses on resource provisioning and job scheduling on the cloud to satisfy performance and monetary cost requirements.

Adam Barker is a Senior Lecturer in Computer Science at the University of St Andrews, a Royal Society Industry Fellow, and an Honorary Fellow at the University of Edinburgh. He worked as a Research Fellow at the University of Oxford, University of Melbourne and University of Edinburgh. He holds a PhD in Informatics from the University of Edinburgh (2007). Adam's primary research interests concentrate on the effective engineering of large-scale distributed systems, covering areas such as cloud computing, big data infrastructure, data-intensive computing and service-oriented architecture.