Snow White Clouds and the Seven Dwarfs

Stephen C. Phillips, Vegard Engen and Juri Papay
IT Innovation Centre
University of Southampton
Southampton, U.K.
{scp, ve, jp}@it-innovation.soton.ac.uk

***Abstract*—With increasing availability of Cloud computing services, this paper addresses the challenge consumers of Infrastructure-as-a-Service (IaaS) have in determining which IaaS provider and resources are best suited to run an application that may have specific Quality of Service (QoS) requirements. Utilising application modelling to predict performance is an attractive concept, but is very difficult with the limited information IaaS providers typically provide about the computing resources. This paper reports on an initial investigation into using Dwarf benchmarks to measure the performance of virtualised hardware, conducting experiments on BonFIRE and Amazon EC2. The results we obtain demonstrate that labels such as 'small', 'medium', 'large' or a number of ECUs are not sufficiently informative to predict application performance, as one might expect. Furthermore, knowing the CPU speed, cache size or RAM size is not necessarily sufficient either as other complex factors can lead to significant performance differences. We show that different hardware is better suited for different types of computations and, thus, the relative performance of applications varies across hardware. This is reflected well by Dwarf benchmarks and we show how different applications correlate more strongly with different Dwarfs, leading to the possibility of using Dwarf benchmark scores as parameters in application models.**

Keywords: application benchmarking; QoS; application modelling; performance prediction, Dwarfs, BonFIRE, Amazon EC2.

I. INTRODUCTION

Today, different Infrastructure-as-a-Service (IaaS) providers describe their infrastructure offerings in different ways and do not necessarily provide very much information, if at all, about the infrastructure being offered. For instance, Amazon EC2 describes (and prices) their infrastructure in terms of Amazon EC2 Compute Units (ECU) as well as the number of virtual cores and RAM size. A machine providing the capability of one ECU is said to be equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. Given the limited and heterogeneous information provided by IaaS providers, how can anyone know what resources they will need to execute their application with a particular quality of service (QoS)? If the application is already adapted for the IaaS provider's system then it may be possible to just try the application out and measure its performance, scaling the deployment as required. But what if the application is not yet adapted, or what if you want to choose between several IaaS providers?

We want to be able to predict the performance of an application given a general description of the hardware provided by the IaaS provider. The first challenge is to find some generic and sufficiently informative way of describing the hardware resources. Such a description should enable prediction of application performance and we hypothesise that the Dwarf benchmarks [1-3] are a good candidate. "A dwarf is an algorithmic method that captures a pattern of computation and communication" [1]. A set of many dwarf benchmark scores can be thought of as a more detailed performance description than the well-known pair of SPECint and SPECfp [4] scores commonly used for measuring super-computer performance. The second challenge is to model the application in such a way that its performance can be predicted on a range of infrastructure specifications (described as scores on Dwarf benchmarks).

We therefore need two things:

1. A description of each candidate IaaS provider's resources in terms of benchmark scores.
2. A model that can predict the performance of the application given the benchmark scores.

In this paper we describe a benchmark suite to measure the performance of virtualised hardware. Ultimately we could imagine each IaaS provider describing the performance of their resources in terms of a standard set of benchmark scores or even couching service level agreements (SLAs) in those terms. Alternatively, a Platform-as-a-Service (PaaS) provider may measure the performance of many IaaS providers, adding to one of many possible services that could be offered.

Once the (virtualised) hardware is described in terms of benchmark scores, these scores must be related to application performance through an application models. These models can help in many ways:

- Making better provisioning decisions: deploying the infrastructure resources required for a given application QoS rather than over-provisioning.
- Making better application scheduling decisions: knowing the application runtime with a good reliability permits more intelligent scheduling.
- Determining the optimal application configuration: the performance of complex applications and business or industrial data processing workflows with many components can be greatly affected by their configuration such as buffer sizes and number of threads.

This work has been carried out in BonFIRE, an EC supported 7th Framework Programme ICT project (FP7- 257386)

- Tracking uncertainty in business processes: many processes are non-deterministic; predicting the likelihood of completing tasks allows for the management of risk.

In this study we use virtualised resources from Amazon EC2 and from BonFIRE. EC2 requires no introduction, but BonFIRE may be unknown to some readers. BonFIRE [5] offers a multi-site testbed with heterogeneous cloud resources, including compute, storage and networking resources, for large-scale testing of applications, services and systems targeting the Internet of Services community.

II. BACKGROUND

A. Computer Benchmarks

There are many general benchmark scores that may be used to predict application performance. For instance, at the most basic level, the SPECint and SPECfp [4] benchmarks measure the integer and floating point arithmetic performance. The LINPACK [6] and the more recent LAPACK [7] benchmarks measure the performance of a computer when performing linear algebra operations common in much scientific software. We are concerned with correlation application performance to benchmarks. For instance, one would expect the performance of a chess program would be closely correlated with the SPECint score and a numerical scientific computation would correlate well with the SPECfp score. However, Seltzer et al. [8] and Zhang [9] argue that application benchmarking is important since standard benchmarks can be uninformative and misleading. The closer the benchmark resembles the application, the better the correlation will be, which is important to our aim of predicting application performance.

A more recent approach to benchmarking is the Dwarf taxonomy first introduced by Colella in 2004 [3], which has been further developed at UC Berkeley [1, 2]. Dwarves have been proposed as a higher level of abstraction than the plethora of benchmark tests that exist and are intended to capture known computational patterns. Initially, 7 Dwarves were proposed for scientific computing applications, which were extended to 13 in [1] to cover SPEC and EEMBC [10] tests, as well as three additional computing areas: machine learning, database software, and computer graphics and games. The current 13 Dwarves are given in Table I.

TABLE I. THE THIRTEEN DWARFS. THE DWARFS USED IN THIS STUDY ARE MARKED WITH AN ASTERISK (*).

Dwarf name	Description
Finite State Machines	XML transformation and video compression
Combinatorial	Logical functions, e.g., encryption
Graph Traversal*	Decision Tree, searching, quicksort
Structured Grids*	Regular grids, can be automatically refined
Unstructured Grids*	Irregular grids, finite elements and nodes
Dense Matrices*	Matrix to matrix operation
Sparse Matrices*	Matrix to vector operations with sparse matrices
Spectral*	Fast Fourier Transformations
Dynamic Programming	Hidden Markov Models, sequence alignment
Particles*	Interactions between particles
MapReduce (Monte Carlo)*	Independent data sets, simple reduction at end Ray-tracing, which is available in PARSEC
Backtrack and Branch & Bound	Constraint optimisation, simplex algorithm
Graphical Models	Hidden Markov Models and Bayesian Networks

The list of Dwarves is not final, which Asanovic et al. [1, 2] do not claim; they do however stress the importance of the abstraction so that the list does not grow too large. Che et al. [11] have proposed parallel benchmarks based on the Dwarf taxonomy, but argue that the Dwarf taxonomy alone may not be sufficient to capture the behaviour in some applications. Furthermore, in a recent study, Kaltofen [12] has identified a need for Dwarves to cover symbolic computation.

The TORCH project (Testbed for Optimization ResearCH) [13, 14] has identified several kernels for benchmarking purposes, including a subset of the 13 Dwarfs listed in Table I above, which can be downloaded from [15]. The current collection contains kernels from: Graph Traversal, Structured Grids, Dense Matrices, Sparse Matrices, Spectral, Particles and MapReduce (Monte Carlo). For each Dwarf, several algorithms are included in the suite which are different in the implementation detail, but nevertheless are all part of a higher level Dwarf. This suite has been adopted in our study, which is detailed further in Section III.

Alongside performing benchmarking of Cloud resources, we are also concerned with monitoring the effective performance of VMs as this may vary over time depending on load on the underlying hardware. Therefore, we are interested in benchmarking resources over time, as the observed variation in performance can be taken into account when predicting application performance.

B. Application Modelling

A generic application model (Fig. 1) takes as input a description of the expected static application workload, a description of the resources (physical or virtual) used to execute the application (including the resource reliability) and a description of any expected user interactions which contribute to the workload or otherwise affect the process. Using some mathematical process, the model makes a prediction about the application performance.

To give a concrete example from the EC IST IRMOS project [16, 17], where some of this work stems from, consider a web server hosting an e-learning application. The workload would describe the number of participants in an e-learning session, the resource description would be the networks connecting the application to the users and the virtual hardware deployed for the application, the reliability would describe the QoS of the virtual hardware and networks (propensity to crash,

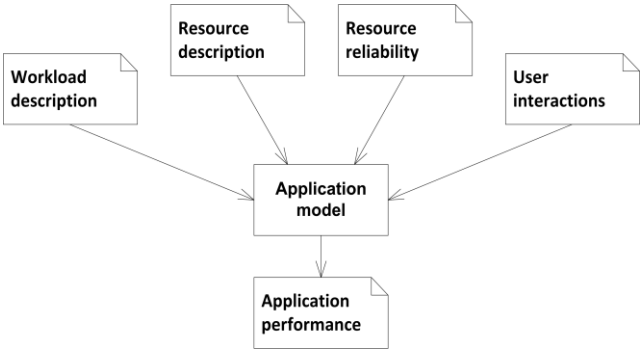

Figure 1. Generalised application model.

latency, etc) and the user interactions would be a statistical description of the frequency and magnitude of the interactions between the users and the application. Using this data and a mathematical model, the average response time of the web server can be computed and the appropriate resources allocated in order to achieve a certain QoS for the users.

There are different methods that can be adopted for modelling, and a combination of several methods may be necessary. For example, in the work introduced above [16, 17], Discrete Event Simulation (DES) was used to model the requests to the e-learning service as events passing through different parts of the modelled system experiencing certain delays depending on the amount of requests and capabilities of the underlying hardware. To compute the innermost processing time of components in such a model, you need a method of estimating this based on a description of the hardware. In [17], this was done based on extensive benchmarking of the application on the known hardware the application could be run on, which was used to train an Artificial Neural Network (ANN).

In this paper we focus on the challenge of calculating the innermost computation time and ignore the problems of varying application workload and user interactions. To enable such prediction to succeed on unseen hardware, it is necessary to have a uniform description of hardware performance. We hypothesise that this is achievable with Dwarf benchmarks.

III. METHOD

We have adapted several pieces of software (the 'Dwarfs') and integrated them into an automated framework for measuring the performance of a machine: the benchmark suite. In addition we have integrated three applications which are also executed and timed so that we can investigate the correlations between application performance and Dwarf score.

A. Benchmarks

We have adopted the Dwarf benchmarks available in the TORCH benchmark suite [13, 14]. This suite is not complete, according to the list of Dwarfs suggested by Asanovic et al. [1, 2], which is highlighted in Table I. We have added one more Dwarf, Unstructured Grid, using the Computational Fluid Dynamics (CFD) software OpenFOAM [18].

In Addition to the Dwarf benchmarks, we are interested in comparing the results with integer and floating point benchmarks. Therefore, we have included Dhrystone [19, 20] and Livermore Loops [21, 22].

Most of the Dwarfs are comprised of multiple algorithms. Workloads of each algorithm have been carefully chosen to require a reasonable runtime without using more memory than available in a 1GB Linux machine. See Table II for an overview of the chosen workloads and their execution times on the reference machine. For Dhrystone and Livermore, we do not have control of the workload, and thus use these benchmarks 'out of the box'.

Please note that workloads presented here are used for score calculation. As mentioned earlier, we are benchmarking with a range of workloads. This enables us to fit a polynomial curve to describe the variation in runtime versus the varying workload, with the intention that this data may be useful when predicting the performance of untested application workloads. However, this part of the work is not the focus of this paper, and is therefore not discussed further.

No benchmark makes use of more than one processor core. This means that the Dwarf scores are invariant as the number of cores and quantity of memory assigned to a VM are increased (if other factors are kept constant).

TABLE II. DWARF WORKLOADS AND EXECUTION TIMES ON THE REFERENCE MACHINE.

Dwarf	Algorithm	Workload	Time (ms)
Graph Traversal	Quicksort	10000000	3176
Structured Grid	3D Central Differences	60	420
	3D Divergence	60	258
	3D Curl	60	356
	3D Gradient	60	350
	3D Laplacian	60	197
	SpMV Laplacian	60	253
Unstructured Grid	OpenFOAM	8100	25617
Dense Matrices	LU	200	557
	QR	100	1753
Sparse Matrices	SPMV	2000	908
	SPTS	2000	826
	CG	2000	697
Spectral	Stockham FFT	1048576	2073
	Cooley Tukey FFT	1048576	3727
	Four Step FFT	1048576	1978
Particles	2D N-Body Cutoff	2000	871
	2D N-Body Barnes Hut	2000	9729
	3D N-Body Cutoff	1000	256
	3D N-Body Barnes Hut	1000	3967
MapReduce	Quasi-Monte Carlo integration	20	4526
Dhrystone	No Register	N/A	5443082
	With Register		7155492
	With Optimisation		9208880
Livermore	Livermore	N/A	1080

B. Applications

For this initial investigation, the following CPU-bound applications have been chosen:

- Gromacs v. 4.0.7 [23]: a molecular dynamics package;
- FFmpeg v. 0.6.2 [24]: a video transcoder;
- Blender v. 2.49.2 [25]: a 3D renderer.

All three pieces of software are open source, cross-platform and are licensed under the GPL or LGPL licenses.

For Gromacs, two different workloads have been chosen. Both configurations run 2500 molecular dynamics steps of a box of 16896 water molecules with a 2fs time-step and periodic boundary conditions. However, one configuration uses a spherical cut-off for the electrostatic calculations and the other one uses the particle mesh Ewald (PME) method. It is expected that these different algorithms for approximating the same physical property will correlate differently with the Dwarfs. The computations take 255 seconds and 758 seconds respectively on the reference machine.

The chosen FFmpeg computation is the transcoding of the "Big Buck Bunny" video [26], which is nearly 10 minutes in

duration. The video was transcoded from M4V (h264 encoded) to OGV (libtheora encoded), also changing the frame size from 640x360 to 480x270. The original sound is also changed from AAC to FLAC during the transcoding. This takes nearly 300 seconds on the reference machine.

Using Blender, our intention was to render frames from the Big Buck Bunny video, but this required more RAM than feasible in practice for the machines we are interested in benchmarking. As with the benchmarks, we have taken care not to exceed a requirement of 1GB RAM. Instead, we render four frames of a bespoke animation, which takes approximately 930 seconds on the reference machine.

C. Benchmarking Process

Each algorithm in each dwarf is executed multiple times in succession with a broad range of workloads. The mean, standard deviation and coefficient of variation are calculated for each workload, and the means are used for subsequent analysis. Score calculation is performed only on the mean of the greatest workload, as specified previously in Table II. Executing the benchmarks multiple times smooths out the small local fluctuations in performance to give a more robust score. The number of executions has been set to 10 for this investigation due to pragmatic constraints on time. The total runtime of the benchmarks and the applications is approximately 75 minutes on the reference machine.

The runtimes of each individual algorithm would provide a representation of a machine's performance but these raw numbers are not ideal for three reasons: firstly the different runtimes for algorithms in the same Dwarf need to be combined, secondly the variation in magnitude for the different runtimes needs to be eliminated and thirdly the general expectation is that a higher number means a better score, but for runtimes smallest is best. For these reasons a final step is taken to calculate the Dwarf scores.

For a Dwarf composed of n algorithms, the Dwarf score D is:

$$D = \frac{1}{n}\sum_{i=1}^{n} 100\frac{T_i}{t_i}$$

where the runtime of algorithm i on a reference machine is T_i and on the machine being benchmarked is t_i. This formula expresses each algorithm's runtime as a percentage of a reference machine to normalise the scores and takes the mean of these sub-scores to create a Dwarf score. The reference machine is chosen to be relatively slow and would, by definition, score 100 for every Dwarf. Data from the reference machine is not otherwise included in this analysis.

The reference machine we have used for this investigation is a Dell Latitude D630, running native Ubuntu Natty. In terms of the benchmarking process we are conducting, and the subsequent analysis of correlations, the specification of the reference machine is irrelevant. It merely serves the purpose of providing some reference values from which scores are calculated. If these values were different, the scores would change, but the correlations would remain the same. However, using reference values from benchmarking results on a machine will allow us a more intuitive reference point, in which it is possible to say that a score of 200 is twice as "good" as the reference machine.

The application runtimes are treated in a similar way to obtain scores that may be correlated with the Dwarf scores. A single workload is executed for each application, and the runtime is then expressed as a percentage of the runtime on the reference machine, giving a baseline of 100 and a larger number for better performance.

To obtain some insights into the performance variation one might experience in the Cloud, all the benchmarks and applications have been executed ten times consecutively.

D. Computational Resources

The benchmark suite and applications have been executed on VMs deployed on a local machine, in the BonFIRE experimental facility [5] and on Amazon EC2. As already described, the benchmarks are invariant to the amount of memory and number of cores assigned to the VM and so although both facilities offer many instance types, only a few vary in anything more than memory or number of cores.

The machines benchmarked are shown in Table III. The CPU information is obtained from /proc/cpuinfo and memory information from /proc/meminfo in the Linux file-system. This information is not necessarily reliable as in theory a VM could be configured to report any data here. The data reported for

TABLE III. THE DIFFERENT MACHINES THAT HAVE BEEN BENCHMARKED AND THEIR BASIC SPECIFICATIONS.

	Machine	CPU	ECU	Cores	Speed (GHz)	Cache (MiB)	RAM (GiB)
	Local	Intel Core i7 M 640		1	2.8	4	1
BonFIRE	epcc-medium	AMD Opteron 2210		2	1.8	1	2
	ustutt-medium	Intel		2	2.7	4	2
	inria-medium	Intel Xeon 5148 LV		2	2.3	4	2
	ibbt-large	AMD Opteron 2212		2	2.0	1	4
Amazon EC2	micro	Intel Xeon E5430	≤ 2	1	2.7	6	0.6
	small (type 1)	Intel Xeon E5507	1	1	2.3	4	1.6
	small (type 2)	Intel Xeon E5430	1	1	2.7	6	1.6
	medium	Intel Xeon E5410	5	2	2.3	6	1.7
	large	Intel Xeon E5507	4	2	2.3	4	7.3
	xlarge	Intel Xeon E5645	8	4	2.0	12	14.7

EPCC, INRIA and IBBT is known to reflect the physical hosts but the data for UStutt ("Intel") is clearly incomplete (if not misleading).

The hypervisor in use on Amazon EC2 and the BonFIRE testbeds is Xen, except for IBBT where the VMs are deployed on physical nodes (no virtualisation used). The operating system used on the VMs deployed on the BonFIRE testbeds is Debian Squeeze, and on Amazon EC2, Ubuntu Maverick.

IV. RESULTS

The entire benchmark suite and applications have been executed on each VM instance shown in Table III. In addition, the benchmark suite and applications have been executed on nine further instances of the EC2 "small" VM to analyse any variation in performance.

In analyzing the scores from the 10 EC2 small instances, an anomaly was noticed: six of the instances performed significantly differently to the other four. Looking at the /proc/cpuinfo data revealed that these two groups of machines claimed to be different CPU models, with the smaller group having a faster clock speed and larger cache. These different machines have been labeled as "small (type 1)" and "small (type 2)" in Table III.

Fig. 2 shows the mean Dwarf scores for the two groups of EC2 small instances. Using Welch's *t* test for populations with unequal variances and assuming a Gaussian distribution, we find that the scores for all the Dwarfs apart from Structured Grid and Sparse Matrix are significantly different between the two groups. This demonstrates a statistically different performance between machines all described as just "small" and "1 ECU" by Amazon. Furthermore, although type 2 has the faster clock speed we see that it performs worse on the MapReduce, Particle and Spectral Dwarfs and worse on the Livermore and Dhrystone tests. This is counter to what would be expected even if the clock speed and cache sizes were known. There are clearly some non-obvious factors influencing the performance which the Dwarf scores are sensitive to.

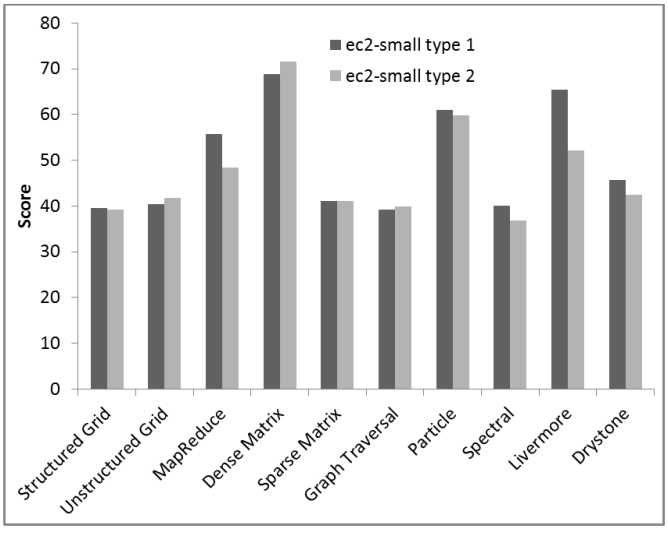

Figure 2. Comparison of mean Dwarf scores for the two types of EC2 small instances.

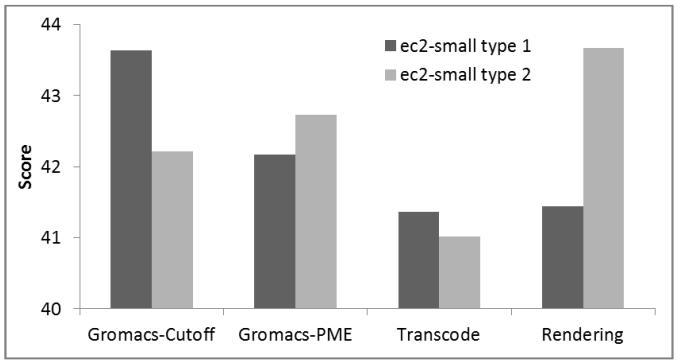

Figure 3. Mean application scores for the two types of EC2 small instances. Note the truncated *y*-axis.

These hidden factors also influence the performance of the applications, shown in Fig. 3. We see that the difference is statistically significant for all four application tests, with the Gromacs-Cutoff and Transcoding tests performing better on the seemingly slower machines (type 1).

Separating out the two EC2 small instance types, we can go on to analyse the behavior of the Dwarfs and applications across all the 11 machine types tested. Fig. 4 illustrates the range of scores for the Structured and Unstructured Grid Dwarfs and also demonstrates that some machines are better at one Dwarf than another. This example shows that the two Dwarfs are discriminating between different hardware types and are measuring in some way the unknown underlying factors.

Taking this a step further, we can look at the correlation of every Dwarf and application with every other Dwarf and application. We expect all pairs to correlate strongly. The interesting data is in the margins of these correlations – just how strongly do they correlate?

Fig. 5 shows the correlation matrix for all Dwarfs and applications, using Pearson's sample correlation coefficient. A key property of the Pearson's correlation coefficient is that it is invariant to the location and scale of the two data-sets. This means that the base-line values taken from the reference machine to compute the Dwarf and application scores do not affect these results. A perfect positive correlation has a value of one and is coloured bright red. An uncorrelated data-set

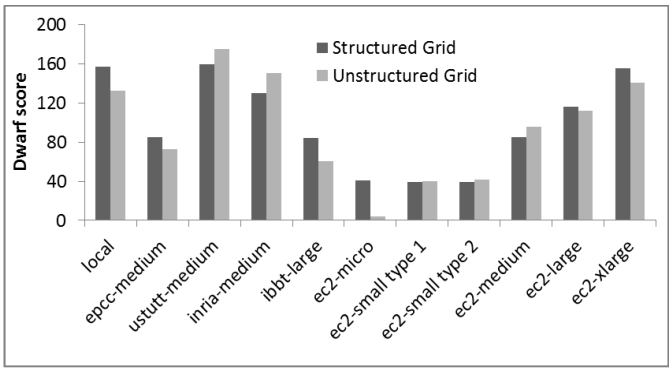

Figure 4. Comparison of Structured and Unstructured Grid Dwarf scores.

	Structured Grid	Unstructured Grid	MapReduce	Dense Matrix	Sparse Matrix	Graph Traversal	Particle	Spectral	Livermore	Drystone	Gromacs-Cutoff	Gromacs-PME	Transcode	Rendering
Structured Grid	1.00	0.94	0.95	0.95	0.97	0.93	0.94	0.94	0.91	0.95	0.85	0.89	0.90	0.93
Unstructured Grid	0.94	1.00	0.97	0.98	0.94	0.95	0.93	0.94	0.90	0.95	0.86	0.92	0.93	0.99
MapReduce	0.95	0.97	1.00	0.97	0.98	0.98	0.97	0.89	0.95	0.97	0.92	0.95	0.98	0.94
Dense Matrix	0.95	0.98	0.97	1.00	0.95	0.95	0.93	0.89	0.91	0.96	0.87	0.92	0.94	0.97
Sparse Matrix	0.97	0.94	0.98	0.95	1.00	0.95	0.98	0.91	0.93	0.97	0.90	0.93	0.94	0.91
Graph Traversal	0.93	0.95	0.98	0.95	0.95	1.00	0.93	0.91	0.88	0.91	0.83	0.88	0.97	0.90
Particle	0.94	0.93	0.97	0.93	0.98	0.93	1.00	0.89	0.97	0.98	0.95	0.96	0.94	0.92
Spectral	0.94	0.94	0.89	0.89	0.91	0.91	0.89	1.00	0.81	0.87	0.75	0.80	0.81	0.92
Livermore	0.91	0.90	0.95	0.91	0.93	0.88	0.97	0.81	1.00	0.97	0.97	0.98	0.93	0.91
Drystone	0.95	0.95	0.97	0.96	0.97	0.91	0.98	0.87	0.97	1.00	0.96	0.98	0.92	0.96
Gromacs-Cutoff	0.85	0.86	0.92	0.87	0.90	0.83	0.95	0.75	0.97	0.96	1.00	0.99	0.90	0.89
Gromacs-PME	0.89	0.92	0.95	0.92	0.93	0.88	0.96	0.80	0.98	0.98	0.99	1.00	0.93	0.93
Transcode	0.90	0.93	0.98	0.94	0.94	0.97	0.94	0.81	0.93	0.92	0.90	0.93	1.00	0.89
Rendering	0.93	0.99	0.94	0.97	0.91	0.90	0.92	0.92	0.91	0.96	0.89	0.93	0.89	1.00

Figure 5. Correlation matrix for all Dwarfs and applications across the 11 machine types benchmarked.

would have a value of zero. The weakest correlation is between the Gromacs-Cutoff application and the Spectral Dwarf (0.75) and this is coloured bright green.

Looking first at the upper-left section of the diagram, where the Dwarf-to-Dwarf correlations are shown, we see strong correlations across the board with the Spectral Dwarf being the most distinct, having correlations as low as 0.89 with several other Dwarfs including the Particle Dwarf. The data points for these two Dwarfs are plotted in Fig. 6, demonstrating that there is indeed a roughly linear relationship. The strong correlations between Dwarf scores are to be expected, but the variation in the strength of the correlations demonstrates that the Dwarfs are indeed measuring different properties of the machines.

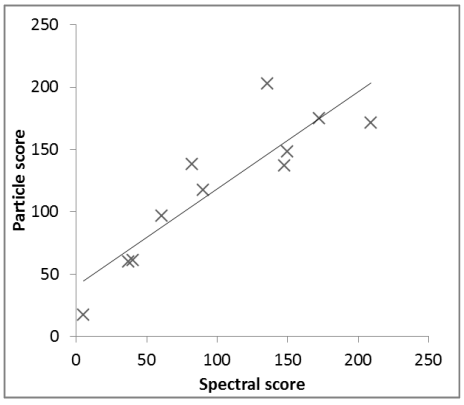

Figure 6. Comparing the Spectral and Particle Dwarf scores for the 11 machines with the trend line shown. The Pearson's correlation coefficient for this data-set is 0.89.

The bright red section of Fig. 5 near the bottom-right is perhaps surprising. This area shows that the Livermore and Dhrystone measures are strongly correlated, demonstrating that floating-point and integer performance scale linearly on the systems tested. The other strong correlations in this area show that the Gromacs-Cutoff and Gromacs-PME computations are very strongly correlated with the Dhrystone and Livermore measures and, as a consequence, correlate very strongly with each other. Note, this does not mean the two application tests are doing the same thing: the Gromacs-Cutoff application is three times faster consistently in this case (though less precise).

It is instructive to look at the variation in application performance across the 11 machines (see Fig. 7). The shape of the four bars varies quite markedly between different groups of machines, demonstrating that some hardware is better at executing some applications than others. The question is, can the Dwarf scores help to predict which application will perform well and which will not?

Returning to Fig. 5 we can inspect the correlations between the applications and the Dwarfs. As already mentioned, the Gromacs applications correlate most strongly with the Livermore and Dhrystone scores, but both also correlate very strongly with the Particle Dwarf. This is to be expected as both algorithms are predominantly sums over functions of particle

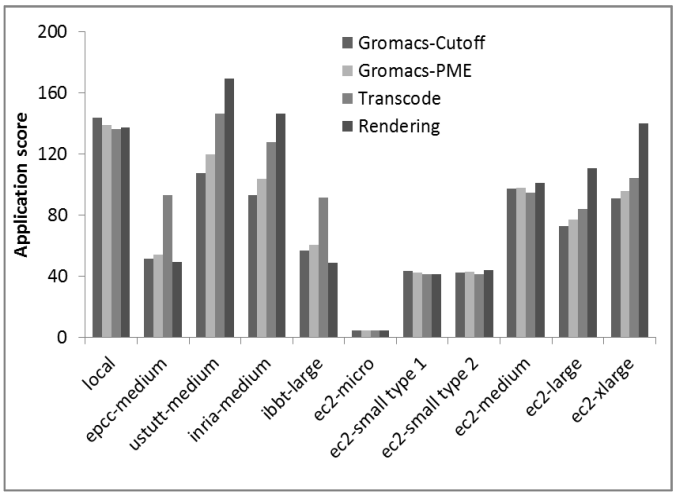

Figure 7. Comparison of application scores across the benchmarked machines.

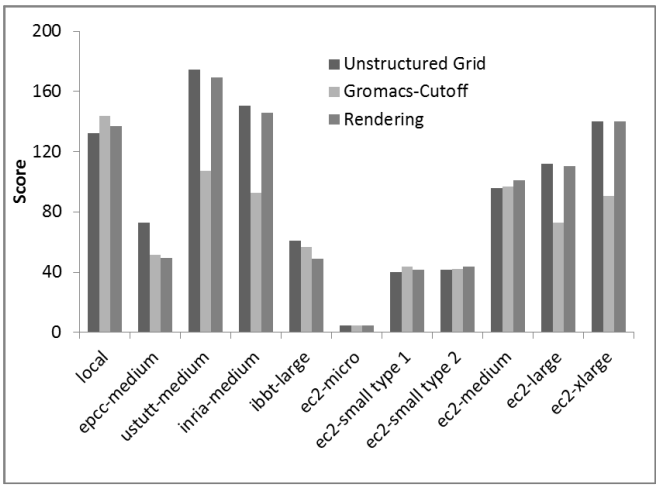

Figure 8. Comparison of the scores for Unstructured Grid, Gromacs-Cutoff and Rendering, showing that the Unstructured Grid Dwarf is a good predictor for Rendering performance but not for Gromacs-Cutoff.

pair distances, with the PME method performing the sum in Fourier space for the long-range portion. The use of the FFT in the PME application may be indicated by an increase in the correlation with the Spectral Dwarf compared to the Cutoff case (0.75 to 0.80).

Of the other applications, the Transcoding application correlates very strongly with the MapReduce and Graph Traversal Dwarfs (0.98 and 0.97 respectively) and the Rendering application correlates very strongly with the Unstructured Grid and Dense Matrix Dwarfs (0.99 and 0.97 respectively).

Fig. 8 gives an indication that these application-to-Dwarf correlations may indeed be able to provide useful predictions of application performance: the Unstructured Grid Dwarf and the Rendering application score clearly move together whereas the Unstructured Grid score is sometimes similar and sometimes very different to the Rendering application score.

V. Conclusion

We have executed a suite of eight Dwarfs, a floating point and integer benchmark and four applications multiple times on virtual machines deployed on eleven different physical host types.

Through analysis of the runtimes and logs of all these codes we have demonstrated that on Amazon EC2, the description of the virtual machines using just EC2 Compute Units (ECUs) and RAM size is not sufficient for predicting performance. Machines all described as "small" may have different clock speeds and cache size and even knowledge of this additional detail does not help in performance prediction as some applications run slower on the machines with a faster clock.

We have shown that the Dwarf scores are sensitive to the sometimes small (but hidden) differences between the architectures of the physical hosts that the virtual machines tested were deployed upon. By examining the correlations between different Dwarf scores we have demonstrated that different Dwarfs measure different aspects of computational performance.

Finally, the varying correlations between Dwarfs and applications suggest the Dwarfs will be useful in predicting application performance as part of an application model.

VI. Further Work

To progress further with this work, more data will be needed, both data of more distinct physical hosts and repeated measurements of the same physical hosts.

We are primarily interested in using the Dwarf scores as a predictor for application performance and will investigate this aspect further, comparing predictions from Dwarf scores with predictions possible from other data such as ECUs or clock speed. It may also be possible to use the data measuring the Dwarf performance for varying workloads to predict the variation in performance of applications as their workload is changed. This will feed into our related work on modelling quality of service terms for applications.

It may also be instructive to apply the benchmark suite to investigating other issues, such as the performance difference between a physical host and a virtual machine on the same host or whether there is any difference in performance between different hypervisors.

Finally, any model of an application executing in the cloud must take into account the variability of virtual machine performance as the load resulting from other users of the same physical host varies. Executing the benchmark suite on the same virtual machine type over an extended period will help measure these fluctuations. We also hope to make use of detailed infrastructure monitoring data from the BonFIRE facility describing the variation over time of CPU and memory resources allocated to each virtual machine.

References

[1] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley UCB/EECS-2006-183, 2006.

[2] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A View of the Parallel Computing Landscape," Communications of the ACM, vol. 52, pp. 56-67, Oct 2009.

[3] P. Colella, "Defining Software Requirements for Scientific Computing," DARPA HPCS Presentation, 2004.

[4] SPEC, "Standard Performance Evaluation Corporation," Available: http://www.spec.org/index.html, 2010.

[5] BonFIRE. (2011). EC FP7-ICT BonFIRE Project. Available: http://www.bonfire-project.eu/

[6] Netlib, "LINPACK," Available: http://www.netlib.org/linpack/, 2010.

[7] Netlib, "LAPACK -- Linear Algebra PACKage," Available: http://www.netlib.org/lapack/, 2010.

[8] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang, "The Case for Application-Specific Benchmarking," in Proc. 7th Workshop on Hot Topics in Operating Systems, 1999, pp. 102-107.

[9] X. Zhang, "Application-Specific Benchmarking," Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, 2001.

[10] EEMBC, "Embedded Microprocessor Benchmark Consortium," Available: http://www.eembc.org, 2010.

[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sceaffer, S.-H. Lee, and K. Skadron, "Parallel Benchmarks Inspired by Berkeley Dwarfs," in Proc. IEEE Int. Symp. on Workload Characterization, 2009.

[12] E. L. Kaltofen, "The ``Seven Dwarfs" of Symbolic Computation," North Carolina State University, Research Report SFB F013, 2010.

[13] A. Kaiser, S. Williams, K. Madduri, K. Ibrahim, D. Bailey, J. Demmel, and E. Strohmaier, "TORCH Computational Reference Kernels: A Testbed for Computer Science Research," EECS Department, University of California, Berkeley UCB/EECS-2010-144, 2010.

[14] A. Kaiser, S. Williams, K. Madduri, K. Ibrahim, D. Bailey, J. Demmel, and E. Strohmaier, "A Principled Kernel Testbed for Hardware/Software Co-Design Research," presented at the 2nd USENIX Workshop on Hot Topics in Parallelism (HotPar), 2010.

[15] TORCH. (2011). Testbed for Optimization ResearCH. Available: https://ftg.lbl.gov/projects/torch/

[16] IRMOS. (2010). EC FP7-ICT IRMOS Project. Available: http://www.irmosproject.eu

[17] T. Cucinotta, F. Checconi, G. Kousiouris, D. Kyriazis, T. Varvarigou, A. Mazzetti, Z. Zlatev, J. Papay, M. Boniface, S. Berger, D. Lamp, T. Voith, and M. Stein, "Virtulised e-Learning with Real-Time Guarantees on the IRMOS Platform," presented at the SOCA, 2010.

[18] OpenCFD Limited. (2011). OpenFOAM. Available: http://www.openfoam.com/

[19] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark," Communications of the ACM, vol. 27, pp. 1013-1030, 1984.

[20] P. Weicker, "Dhrystone benchmark: rationale for version 2 and measurement rules," ACM SIGPLAN Notices, vol. 23, pp. 49-62, 1988.

[21] [21] F. H. McMahon. (1993). Livermore Loops C Code. Available: http://www.netlib.org/benchmark/livermorec

[22] F. H. McMahon, "Livermore fortran kernels: A computer test of numerical performance range," NTIS report UCRL-53745, 1986.

[23] GROMACS (2011). GROningen MAchine for Chemical Simulations. Available: http://www.gromacs.org/

[24] FFMPEG (2011). Available: http://www.ffmpeg.org/

[25] Blender (2011). Available: http://www.blender.org/

[26] Blender (2011). Big Buck Bunny. Available: http://www.bigbuckbunny.org