Describing Cloud Usage with Excess Entropy

Charles Loboz
Windows Azure
Microsoft Corporation

ABSTRACT
The growing complexity and size of computer systems calls for improved methods to describe and analyze them. We propose to use excess entropy for that purpose. Excess entropy can be applied to quantify imbalance and concentration of processing. It also allows considering systems composed of subsystems. We demonstrate the utility of excess entropy approach by applying it to the description of the elements of load in Windows Azure compute clusters. Using excess entropy we locate unusual situations in the division of load between clusters. We also apply excess entropy to describe the imbalance of load in time.

Categories and Subject Descriptors
Performance of Systems

General Terms: Management, Performance.

Keywords: capacity planning, resource usage, entropy, imbalance, probability distributions.

1. INTRODUCTION
A modern server has over 2000 performance counters which are relevant to the description of its state and usage – that applies to both Windows and Unix servers. For one server we can select a smaller subset of counters requiring monitoring, but if we have several servers running different applications – the size of the monitoring set grows quickly. In addition the complexity of computer systems is growing, too. We started computer system performance analysis and capacity planning with a single mainframe. Then we have moved to multiple mainframes and groups of servers. That was followed by multiple virtual machines running on a single server. The current stage – cloud computing - is, in effect, an operating system controlling execution of processing on clusters of servers and cluster groups.

We need to consider both traditional system descriptors as well as the new ones arising from the handing server groups and virtualization. Examples of new descriptors include estimating effects of competition for disk bandwidth between multiple virtual machines running on the same server and sharing physical disks - or similar competition for network rack switches between virtual machines deployed to the same rack of servers.

Performance analysts and capacity planners have to deal with information explosion in two different dimensions. The first one is related to the scale of modern web services, when datacenters containing tens of thousands of servers are providing hundreds of services – thus we have data from a single server multiplied 100,000 times (or more). Most complexity in this dimension is coming from the number of servers. The second dimension is the virtualization and cloud artifacts – consideration of deployment strategy for virtual machines, consideration of migration options for virtual machines to other servers or clusters and management of whole clusters of servers.

To manage this information explosion we need descriptors of overall system usage that are on higher conceptual level than direct performance counters, like processor utilization, number of disk operations, memory bytes used, packets transferred through a network and other performance counters of this type.

An example of such higher-level descriptor is Performance Impact Factor (PIF) introduced in [7]. Processor utilization reported by the system monitor can be misleading – a daily average utilization of 0.2 may come from a server which is working evenly all day and could handle much more work – or from a server which is severely overloaded for four hours per day with disastrous consequences to response time and service level agreement. Differentiating between these extremes (and all intermediate situations) requires looking at daily load chart – but this is an impractical option for 100,000 servers. PIF is weighted average of processor (or disk or network card) utilization designed to signal how the load profile may affect the server response time. In effect PIF transforms the data from performance counter space to performance impact space. That simplifies analysis of a large number of servers, because PIF is a one-number summary and captures the information not easily discernible from the original counters.

Another example of a higher-level descriptor is Capacity Usage Factor (CUF) introduced in [8]. CUF starts with the notions of software work and computer capacity to perform that work and combines it with other system descriptions like cost or power . CUF is a generalization of utilization coefficient accommodating differences between processors (and disks, and network cards). It allows comparison of usage levels between servers with different hardware and between groups of such machines.

In this paper we propose another higher-level descriptor based on excess entropy. We argue that excess entropy can be used to describe imbalance between use of similar components (hardware or software) in the system. Alternatively and equivalently excess entropy can be viewed as describing the concentration of use of a system or a subsystem.

A parallel paper [9] applies excess entropy to describe usage of Windows Azure storage clusters by individual customer accounts. This paper applies excess entropy to selected Windows Azure compute clusters and cluster groups.

DataCloud-SC'11, November 14, 2011, Seattle, Washington, USA.

2. EXCESS ENTROPY, DISORDER, IMBALANCE AND CONCENTRATION

The notion on entropy has been introduced in thermodynamics in mid-19-century by Clausius. Twenty years later Boltzmann expanded the notion of entropy to statistical mechanics, which started the interpretation of entropy as that of a 'disorder'. In particular entropy describes how the system state is positioned among the all possible system states.

In economics the distribution of various forms of wealth in the society was a topic of interest and various quantitative measures of economic inequality were proposed over the years [2,3,4,5]. In the end the measures based on entropy were introduced and became the mainstream.

2.1 Imbalance in computer systems

This subsection introduced the concept of imbalance and illustrates its use on a simple example of utilization of a disk set.

Imbalance in various forms is frequently one of the symptoms of suboptimal or inefficient usage of a server or group of servers. For example a server with several disks will perform sub-optimally if one of the disks is handling all the requests and other disks are doing nothing - majority of the requests will be heavily queuing on one disk resulting in high overall application response time.

Discovering the imbalance of usage between two disks is trivial - we only need to compare two numbers. Estimating imbalance when larger number of disks is involved is easy in extreme cases - we can just look at utilization charts. However, when the imbalance is less extreme, differentiating between different situations is difficult. Figure 1 shows example utilization of three disk sets - deciding which set is more imbalanced is not an easy task even when the number of disks in each set is the same. Another practical issue occurs when the number of disks on our servers is not identical - how do you compare imbalance level of 4-disk server with 6-disk server?

Estimation and comparison of imbalance gets more complicated when we have hundreds or thousands of servers. Eyeballing multiple charts is not an option - we need quantitative measure of imbalance to estimate how many servers need disk balancing and prioritize the task to handle the most imbalanced servers first. Also, with that number of servers there is more chance that some of them will have different number of disks than others.

The issue of imbalance in computer systems usage is not limited to disk use. It can be applied to groups of servers processing transaction load or web queries, to a set of network devices, use of various applications performing similar service and so on.

2.2 The measures of imbalance

For a system described by set of values $(x_1, x_2, \dots x_n)$ the *excess entropy* (EE) is computed by:

$$EE = \frac{1}{n}\sum_{i=1}^{n} \frac{x_i}{x_{avg}} * \ln(\frac{x_i}{x_{avg}})$$

Excess entropy is the difference between the maximum entropy possible in the system and the entropy present in the system. The maximum value of EE is *ln(n)* - when all elements but one are equal; the maximum EE depends thus on the set size.

We can view EE as a description of how concentrated is the use of some resource – large values mean fewer users use most of the resource. When comparing systems with differing number of elements it is convenient to use normalized excess entropy:

$$nEE = \frac{1}{n * \ln(n)}\sum_{i=1}^{n} \frac{x_i}{x_{avg}} * \ln(\frac{x_i}{x_{avg}})$$

Examining the EE formula we can derive several properties:

- the ratio x_i/x_{avg} describes how much element x_i is above or below the average for the whole set. Thus EE involves only the ratios - not the absolute values of the elements.
- the EE is dimensionless – thus allows to compare imbalance between sets of substantially different quantities, for example when one set contains disk utilizations and another set contains response times.

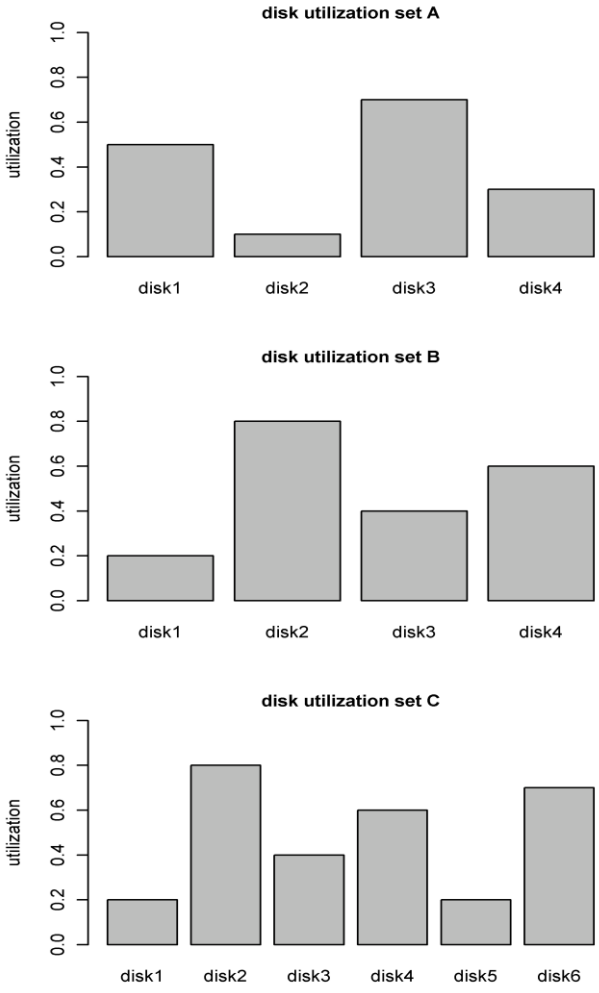

Figure 1 Example utilizations for sets of disks

- the minimum value of EE is zero - when all elements of the data set are identical. The maximum value of EE is.$\ln(n)$.

We can view EE as a description of how concentrated are the values in the data set – large values of *EE* mean fewer set elements contribute to the sum of all values in the set, small values mean more equal contribution.

Excess entropy was introduced in economics by Henri Theil to measure inequality of income and is known there as the *Theil index* [2, 3].

2.3 Sample application of EE

This section uses synthetic examples of groups of disks and their utilizations to develop some intuition, illustrate the calculation and interpretation of EE. Then we consider other (than utilization) bases for computing Imbalance coefficient and consider some operational characteristics of *nEE*.

We compute solutions for three cases: a set of four disks, testing whether tuning reduced imbalance, and comparing imbalance between two sets of disks of differing sizes.

2.3.1 Imbalance of a set of disks

We compute the imbalance coefficient for a set of four disks with utilization coefficients: (0.5, 0.1, 0.7, 0.3), where disk utilization is defined as fraction of time the disk is busy (see Figure 1, top picture).

Applying the EE formula we get the imbalance coefficient:

1/4 * [0.279 + -0.347 + 0.979 + -0.216]/log(4) = 0.126

The components of the sum also allow to quantify the input of each disk to the total imbalance of the whole set. This is different from the mean value – it gives the insight into the sizes of factors contributing to the nEE. We see that disk3 contributes the most: 0.979 while disk1 contributes only 0.07. It is obvious that disk3 is most utilized, so tuning should start with disk3 - we do not need the nEE to tell us that. The nEE computation also quantifies that - to reduce imbalance - working on reducing the load on disk3 is about three times more effective (0.973/0.279) than reducing load by the same amount on disk1.

2.3.2 Estimate imbalance before and after tuning

We analyze whether the tuning reduced the imbalance - in a 'live' system.

Assume that, after analyzing and rearranging the load and file layout, we repeat the utilization measurement and obtain following utilization coefficients: (0.2,0.8,0.4,0.6) [see Figure 1, top and middle pictures]. We want to estimate whether tuning had reduced the imbalance. The measurements are taken from the live system, so – after tuning - we have both different utilization distribution between disks as well as slightly different overall utilization.

Using the same formula we find that the imbalance coefficient after tuning is 0.077 - much lower than the original imbalance of 0.126. Thus nEE quantifies the improvement achieved by tuning. The imbalance looks rather similar on pictures; the difference in quantitative measure is much larger.

The naive approach here would be to use some measure only tangentially related to imbalance, for example standard deviation of utilizations. That creates a host of mathematical problems for further usage. Among others, it does not differentiate well - for both sets of utilizations the standard deviation is identical at 0.258 - yet the imbalance coefficients are quite different. So, at the minimum, the Imbalance Coefficient describes a different aspect of the system than standard deviation.

2.3.3 Imbalance when sets of disks are of different size

We compare the imbalance between two sets of disks, each set having different number of disks.

Assume we have two disk sets, one consisting of four disks with utilizations c(0.2,0.8,0.4,0.6) and another consisting of six disks with utilizations c(0.2,0.8,0.2,0.4,0.6,0.2,0.7) [Figure 1, middle and bottom pictures]. Applying the formula we see that the excess entropies for two disk sets are, respectively, 0.77 and 0.70 – suggesting that the first disk set is slightly more imbalanced than the second and should be tuned first.

Trying to assess the imbalance intuitively is more complicated when the number of elements differs in each set. Also, in the first and second examples we could have used *EE* instead of *nEE*. Here, to compare sets with differing numbers of disks, we had to use the normalized version of *EE*.

2.3.4 Other bases for imbalance computation.

In the examples above we have used disk utilization to illustrate computations of imbalance. However, we can use other bases for computing imbalance. Selection of the base for imbalance computation depends on our goal.

In some cases other (than utilization) disk characteristics may be more relevant to the overall system performance - the number of input/output operations, the number of bytes transferred, operation time, queuing time or disk response time. The goal may be to assure balance of, say, queuing times or normalized (for transaction type or size) response times rather than utilizations.

The same formula can be used for all such calculations but the resulting nEE may have significantly different value when different bases are used. For example computing *nEE* using disk response times is likely to give larger nEE than for disk utilizations – as response times grow disproportionately when utilization is high.

We must note that even when the *nEE* computed for a disk set using various bases usually will have different values; the underlying meaning of *nEE* is still the same. $nEE = 1$ means total concentration of resource use or characteristic on element, $nEE = 0$ means perfectly equal distribution. If we want to achieve balance of response times, we should move files around to obtain the lowest response time *nEE* – even though that may result in high *nEE* for utilization.

2.3.5 Some operational characteristics of nEE

The nEE formula has a simple interpretation – it tells us about the concentration of use. High *nEE* means that most of overall disk use is concentrated in a small number of disks. Low *nEE* means that the disk use is spread more evenly. At the extremes, if only one disk had non-zero utilization, the *nEE* would have been equal to 1. If all the disks had the same utilization the *nEE* would have been zero. The underlying mathematical apparatus is to keep consistency on all intermediate situations.

The nEE spots the imbalance even at low levels of load - thus giving an early warning signal that, if the load grows, the imbalance can impact performance. For example with two disks at utilization levels 0.05 and 0.4 the overall system performance is unlikely to be affected and such server may easily be overlooked during analysis. Were the load on this server to grow by a factor of two, the utilization levels may grow to 0.1 and 0.8 - and that combination of level of utilization and imbalance is likely to impact the performance of the overall system. Thus the imbalance

coefficient works in a load-level-independent fashion and can be used as an early warning system.

That load-independence of nEE poses potential danger. The practical use of imbalance coefficient must take into account the noise level in the data. For example with four disks with utilizations (0.001, 0.001, 0.001, 0.009) we have high nEE of 0.4 - but at these load level the usage is effectively just noise - so there is little point in tuning effectively unused disks. At the same time, if the utilizations were like (0.001,0.001,0.001,0.1) – that is one disk was used in a non-trivial way and the others not at all – we could have had an imbalance situation. Fortunately, in most practical cases we can easily define the appropriate noise level for any basis and avoid mis-interpreting spurious high imbalance coefficients.

Computation of the level of imbalance may include adjustments for differences (if any) between disks in the disk set. For example, assume we have two disks with one being much faster than the other. Computing the imbalance metric using the number of operations per second would lead to equalizing the number of operations on both disks - which would be operationally incorrect - the faster disk can process more operations. A better approach here would be to consider the maximum capacity of each disk (in operations per second) and compute imbalance using the measured fractions of that capacity.

2.3.6 An aside on nEE and response time

This paper is about general definition and selected sample applications of excess entropy. The response time is not one of the applications described here because we did not have data available at the time of writing. Since response time is frequently the critical customer-side parameter so we will make few comments on response time and excess entropy.

Our examples so far were based on utilization – which has limited range from 0% to 100%. In addition utilization below, say, 1% can be considered a pure noise for most practical purposes.

In contrast the response time is much more multidimensional. We have to keep in mind that the response time is a combination of processing time and queuing time – so the imbalance in response time is a combination of these two effects.

Small response times are not noise – they are essential information. The range is much wider - even when we consider just the disk response time – it varies from less that 1ms (cached requests) to tens of seconds (similar to the range of variables like network transfers). If we consider averaging response time over long time periods with large number of individual response times in each period the average response time may be sufficient base for IC calculations. However, a single time period with a small number of response times and one of them being large may sway the average significantly. Thus using percentiles may be preferable to using averages when comparing disk response times – and keeping track of the number of events in the time period will be important, too.

Another aspect is that of specialization – in a system with multiple transaction types some transactions may be 'large' and addressing the information on specific disk – thus resulting in large average response time on that disk.

These elements are not newly introduced by the use of excess entropy – they apply to utilization and all earlier aspects of evaluating system performance – we need to be aware of the context in which any measure or system description is applied.

3. EXCESS ENTROPY AND CLUSTER LOAD

This section describes application of excess entropy to a load in a cluster group consisting of several Windows Azure compute clusters. We have hourly and daily data for several months. In particular we consider two load characteristics: (1) *ComputeHours* – the number of hours used by by virtual machines on the cluster; and (2) *Egres* – the transfer between the cluster and the outside world. [Values for both load characteristics are normalized to the largest value in the considered time period for confidentiality reasons. This does not affect any distributional computations; in particular it does not affect any excess entropy computations].

We start with examples of hourly load and high/low *EE* values and load configurations. Then we observe *EE* and load evolution over several months.

3.1 Hourly excess entropy for a cluster group

We consider first the network egress of six clusters over 20 hours.

Figure 2 shows (top picture) the load for each of the clusters between hours 5151 and 5170. The bottom picture shows the normalized excess entropy for the cluster group computed for each hour. The smallest *nEE*, at hour 5159, is about 0.02. Looking at the load of each of the clusters at the same hour we see the load of five clusters within the narrow range and the sixth cluster having large load.

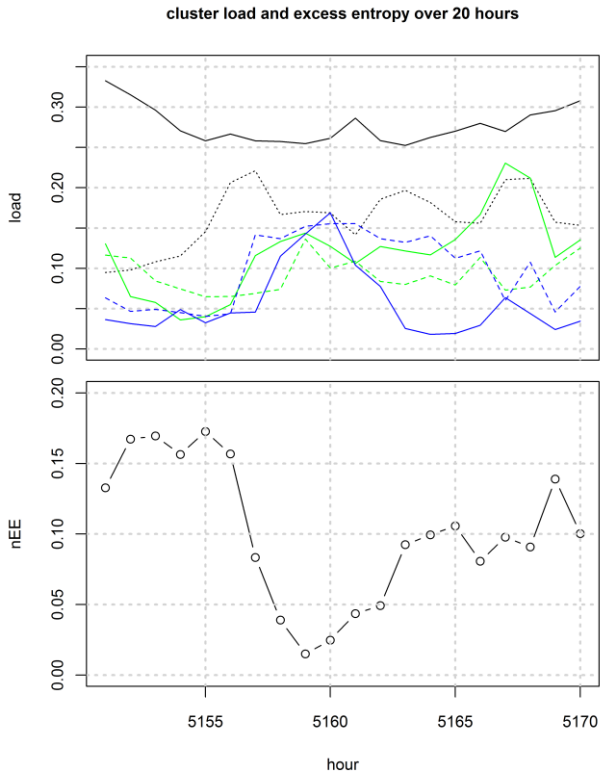

Figure 2 Cluster loads for six clusters (top picture) and excess entropy for the cluster group over 20 hours. Cluster loads are scaled so the maximum group usage over the whole measurement period is 1.0

Table 1 gives details for the evolution of load preceding the minimum excess entropy. Four hours before the minimum, at hour 5156 (first row) the load on clusters 3 to 4 was around 0.05 with clusters 1 and 2 having load over 0.2 – resulting in *nEE=0.16*.

Four hours later (last row) we had clusters 2 to 6 with the load ranging from 0.14 to 0.17 and cluster 1 with the load of 0.25, resulting in *nEE*=0.2 – eight time smaller than before. At the minimum value of *nEE* the load is spread almost evenly across all clusters.

Table 1 Normalized excess entropy and cluster loads for four hours preceding the minimum nEE at hour 5159

#hour	nEE	cl1	cl2	cl3	cl4	cl5	cl6
5156	0.16	0.27	0.21	0.06	0.06	0.04	0.04
5157	0.08	0.26	0.22	0.12	0.07	0.05	0.14
5158	0.04	0.26	0.17	0.13	0.07	0.12	0.14
5159	0.02	0.25	0.17	0.14	0.14	0.14	0.15

Figure 3 shows (top picture) the load for each of the clusters between hours 5291 and 5310. This time the maximum *nEE=0.4* is much larger than before.

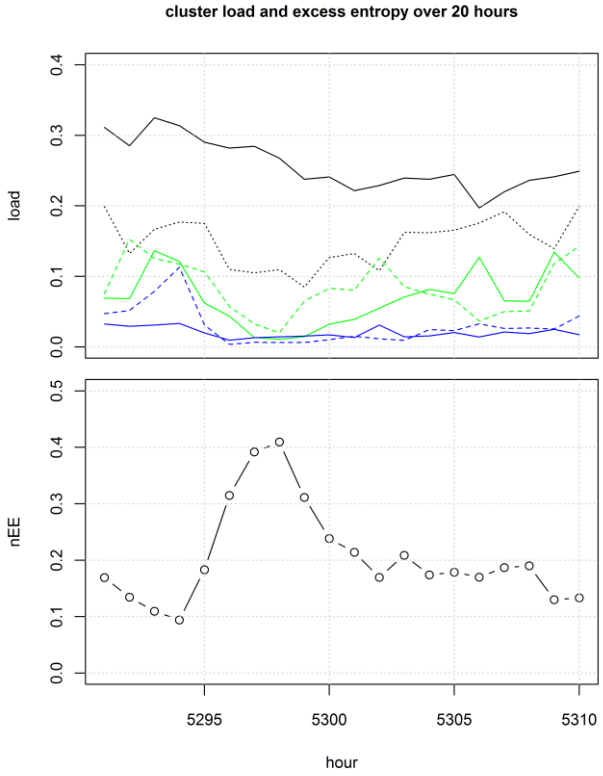

Figure 3 Cluster loads for six clusters (top picture) and excess entropy for the cluster group over 20 hours. Cluster loads are scaled so the maximum group usage over the whole measurement period is 1.0

Table 2 shows the details – the source of high *nEE* at hour 5298 is drop in load (to almost none) of clusters 3 to 7 with the load in clusters 1 and 2 being order of magnitude higher. Thus, at the maximum value of *nEE* the load is concentrated in two out of six clusters.

These examples illustrate that differences in load between clusters are translated by *nEE* into a numerical estimate of the imbalance or, (equivalently) concentration of the processing among clusters. What is intuitively obvious in extreme cases is translated into a single numerical value describing the degree of concentration.

Table 2 Normalized excess entropy and cluster loads for four hours preceding the maximum nEE at hour 5130

#hour	nEE	cl1	cl2	cl3	cl4	cl5	cl6
5294	0.09	0.31	0.18	0.12	0.12	0.03	0.11
5295	0.18	0.29	0.18	0.06	0.11	0.02	0.03
5296	0.31	0.28	0.11	0.04	0.06	0.01	0.00
5297	0.39	0.28	0.11	0.01	0.03	0.01	0.01
5298	0.41	0.27	0.11	0.01	0.02	0.01	0.01

3.2 Excess entropy and daily load

We consider excess entropy of a cluster group over 300 days. The cluster group started with five clusters and was extended later to seven clusters.

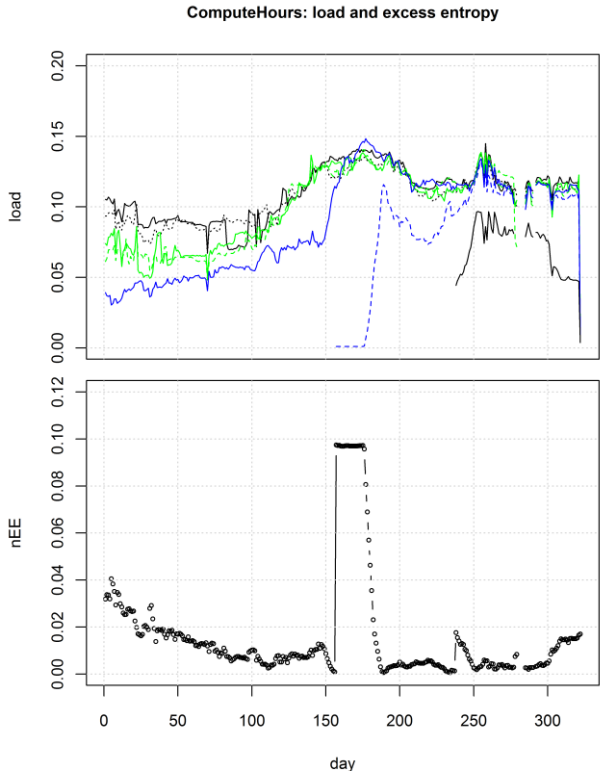

Figure 4 *ComputeHours* and excess entropy for a cluster group over 300 days.

Figure 4 shows the load for each cluster (top picture) in the number of compute hours used by virtual machines deployed on each cluster (the values are scaled so the maximum *ComputeHours* for the cluster group in the measured period is 1.0). The bottom picture shows the normalized excess entropy of the cluster group describing imbalance between clusters' load. In the first few days there are five clusters with loads differing by a factor of three. That disparity is gradually decreasing. One of the clusters is exhibiting very variable load around day 25 – and we can see that reflected in outliers on the *nEE* plot below. Overall the load is getting more evenly distributed and the *nEE* is tending down till the day 150. At that time a new cluster is introduced (dashed blue line) and remains unpopulated (but monitored) for few days. That creates a large jump in imbalance. Around the day

185 the load on that new cluster rapidly grows and the imbalance is rapidly coming down to the previous level.

Another new cluster is added around the day 240, but this time its load is non-zero from day one and grows quickly. The imbalance disturbance is much smaller and goes back to the values typical to this cluster group much faster.

The next Figure 5 shows the load and imbalance for the same date range and cluster group – but this time for *Egres*. Overall *Egress* load and excess entropy values are more volatile than *ComputeHours* (The 'quantum' of deployment is a virtual machine measured in hours, but the 'quantum' of *Egress* is a single byte; in additiona different uses tend to use network differently).

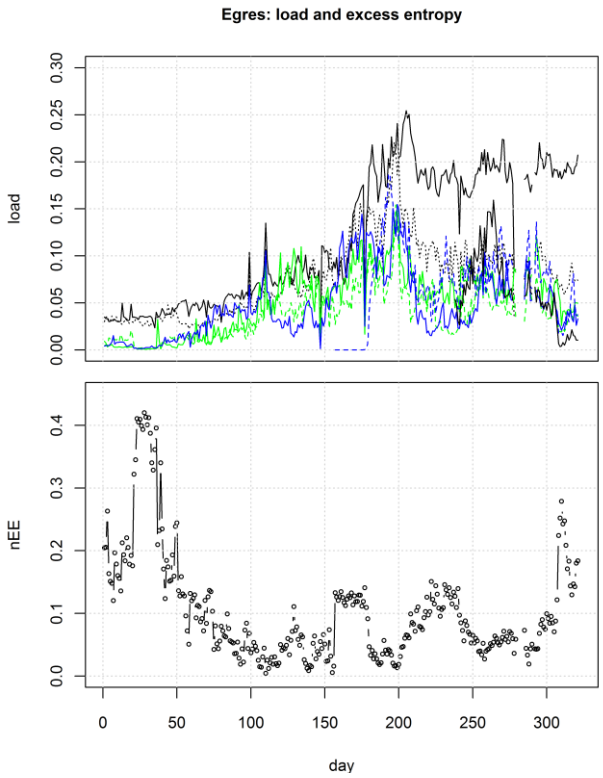

Figure 5 *Egres* and excess entropy for a cluster group over 300 days.

Volatility of *nEE* is not the only difference between these system characteristics. The overall imbalance for *Egress* is higher than the imbalance for *ComputeHours*. The *nEE* for *Egress* stayed mostly above 0.05, while for *ComputeHours* were below 0.02.

The disturbances in imbalance of *Egres,* caused by the introduction of new clusters, are still visible but relatively smaller than for *ComputeHours*. We have imbalance of 0.4 around day 25 and another high imbalance of 0.3 around the day 310 – and they are not related to the introduction of new clusters but caused by variations in the activity between the existing clusters.

The examples show that we can use excess entropy as a general indicator of disturbances in load distribution between clusters. Were the number of clusters small (like two-three) we could just look at the loads of individual clusters and notice unusual situations. However, even with 5-7 clusters spotting of unusual situations by looking at individuals loads is becoming problematic - excess entropy allows better aggregate description.

The examples above also show that we can compare imbalance of significantly different usage characteristics of the cluster (these characteristics are not even measured in the same units).

4. Excess entropy in time dimension

Previous section applied excess entropy to describe imbalance in load (concentration of load) in clusters of a cluster group at given moment in time. This application is similar to that described in the introduction – imbalance between server disks or evaluation of performance of some load balancer. We considered evolution in time of the imbalance between system elements – the 'natural' application of excess entropy.

Here we consider a non-obvious application of excess entropy to describe the *variability of load in time* – how imbalanced (concentrated) is the load within the day or week. Here we do not consider imbalance between usage characteristics of similar system elements. We consider one element – cluster group load – and compute excess entropy for given day using hourly load levels.

We analyze the total load of the cluster group (normalized to the maximum, as before).

4.1 Intra-day imbalance

First we consider the concentration of load within each day. We take the load in each of the 24 hours of the day and compute excess entropy for that day. We repeat the process for each day. That daily excess entropy characterizes the spectrum of hourly loads.

Figure 6 shows the daily load for the cluster group (top picture) and intra-day imbalance for each day (bottom picture). The daily load is not directly related to the intra-day imbalance – we see significant growth in daily load but with no corresponding growth in imbalance. We also see high imbalance for days 14, 20, and 76 – while the daily load on these days appears does not differ from the neighboring days..

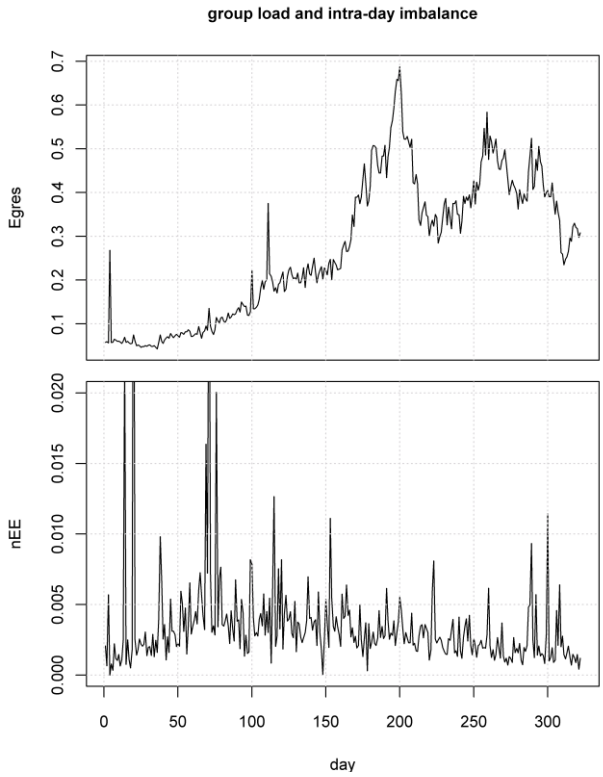

Figure 6 Total *Egress* for the cluster group and intra-day imbalance.

Figure 7 shows the intra-day hourly load history for two days with high imbalance (14, 76 – black lines) and two days with low imbalance (6, 321 - blue lines).

Hourly load history for high-imbalance days shows large jumps from the average. In contrast, low-imbalance days – one with high load, another with low load – show more even profile. Again, the load level is not related to the *nEE* – dashed blue line for day 321 shows the load at the level of 0.3 but the *nEE* for that day is similar to day 6 (solid blue line) with load level 0.06 – five times lower – this is because the excess entropy measures effectively the differences from the average relative to that average.

4.2 Systems of subsystems

An important property of entropy-based measures of inequality is that they are *composable*, that is the excess entropy of the total system can be computed as a composition of excess entropies of subsystems. Since most large systems are composed of subsystems, that composition capability gives an obvious advantage to excess entropy over other approaches (like Gini index, still popular in economics and elsewhere [4]).

Such composition capability is important as it allows applying imbalance considerations to any hierarchical system. Thus we are able to apply excess entropy to a data center consisting of a set of clusters and to a region consisting of a set of datacenters.

4.2.1 Composite Excess Entropy

We have a group consisting of M subgroups. Subgroup m with excess entropy EE_m used fraction f_m of the total resource used by the group. Also, avg_m is the average value in subgroup m, and avg_{tot} is the global average value. The formula for composite excess entropy (*CEE*) is [2,3]:

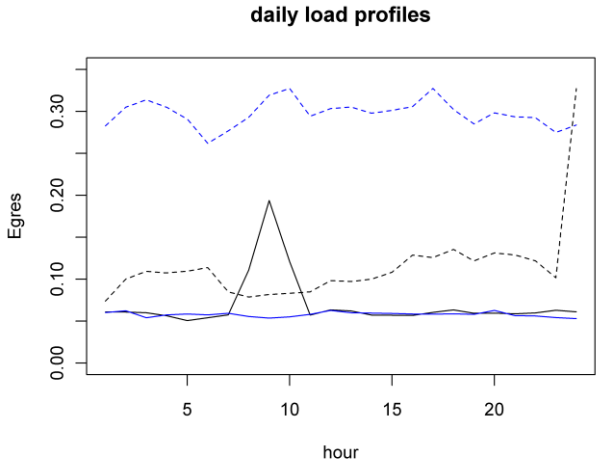

Figure 7 Intra-day load histories for selected days. Blue: days with low *nEE*, black: days with high *nEE*.

$$CEE = \sum_{m=1}^{M} f_m * EE_m + \sum_{m=1}^{M} f_m * \ln(\frac{avg_m}{avg_{tot}})$$

Thus the composite excess entropy of the group is the sum of (1) excess entropy within subgroups weighted by the fraction of resources used by these subgroups and (2) excess entropy between the subgroups weighted by the fraction of resources used by each subgroup. This approach allows us to decompose the imbalance of the total system into imbalances of its subsystems and identify which subsystems contribute most of the overall imbalance. In addition we can quantify whether contribution of a subsystem to overall imbalance is caused by internal imbalance within that subsystem or by the imbalance between that subsystem and other subsystems.

That allows us, for example, to compute the imbalance for the whole data center as a function of internal imbalance of individual clusters and differences between clusters[9]. In the process we can quantify how much each cluster contributes to the datacenter imbalance and whether that contribution is coming from the internal imbalance within that cluster or a difference between that cluster and other clusters in the data center.

4.2.2 Weekly and daily imbalance

We apply the composition of excess entropy to compute imbalance in weekly load – composed of imbalances in daily loads within that week.

We consider one week of hourly *Egress* load data for a week starting at day 95. Figure 8, top picture, shows daily load for each day with the hourly range in the day marked by grey bar and 10-90 inter-quantile range marked by a black bar. The bottom picture shows intra-day hourly load evolution with weekday number overlayed. We see that weekday 6 (day 100) was very different from other days having both higher average load and wider range.

we see (top picture) that *nEE* for days 99 and 100 was similar in size and much higher than for all other days if this week. However, their contribution to the weekly composite *EE* was different.

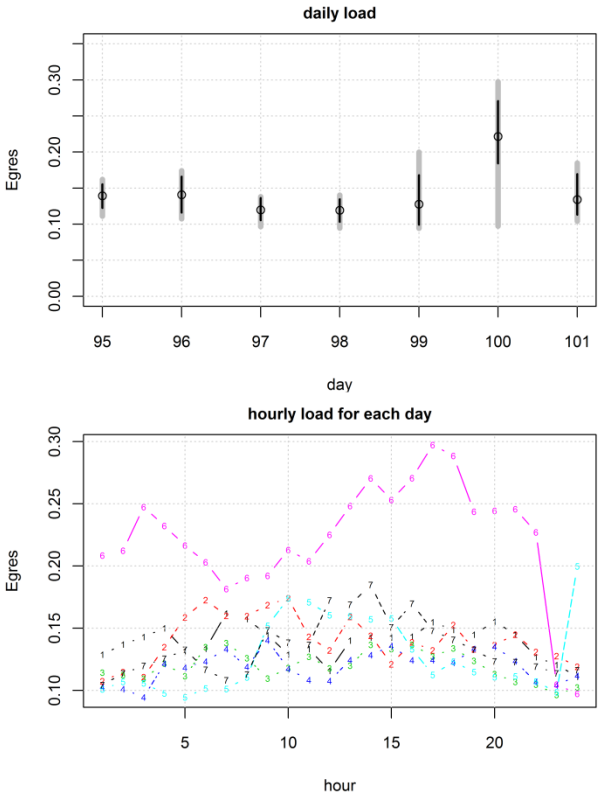

Figure 8 Daily and hourly load for week 14 (days 95 to 101)

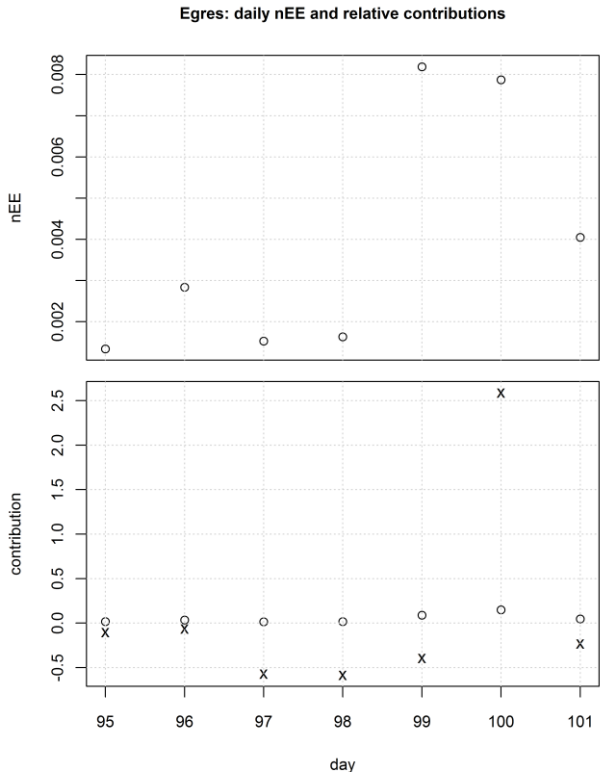

Figure 9 Excess entropy and composition of excess entropy for week 14; the circles in the bottom picture denote (scaled) contribution of intra-day imbalance, the crosses denote the scaled contributions of between-days imbalance.

Looking at various excess entropy characterizations in Figure 9 The bottom picture of Figure 9 shows contributions to *CEE* for each day (scaled, so the sum of all contributions in the week is equal to 1). The circles denote contributions from the first term in the *CEE* formula (inequality within subgroups/days), the crosses denote contributions from the second term (inequality between subgroups/days). Day 100 contribution to the weekly *CEE* is large and comes almost exclusively from the difference between this day and other days. Day 99 contribution to the weekly *CEE* is much smaller than that of day 100 - even though intra-day *EEs* for both days were similar. .

Overall the contribution to the weekly *CEE* from intra-day excess entropy is small – very close to zero for all days. The weekly *CEE* depends mostly in differences in load between days. Also, the first two days of the week do not contribute much in either dimension.

Figure 10 shows the same information as Figure 9, but for week 19 (days 137 to 143). We see that that weekday 2 had much larger intra-day imbalance than other days (top picture). Overall contributions from intra-day imbalance to the *CEE* (bottom picture) were small, similar to week 14. The largest contribution to the weekly *CEE* came, as before, from weekday 6 and it was almost entirely caused by the between-days imbalance.

Contribution pattern shown in the two sample weeks suggests that the weekday 6 may be different from other weekdays. We compute (scaled)) between-day contributions of weekday 1 to of of the 45 weeks and repeat the computation for weekday 6.

Figure 11 shows that the spectrum of between-days contributions for weekday 1 is shifted to the left (top picture) with median value of -0.54 , while similar spectrum for weekday 6 is shifted to the right (bottom picture) with median value 0.31, with similar differences between respective 25^{th} and 75^{th} percentiles.

So the differences between these two weekday days in contributions to weekly excess entropy is real and not caused by some outlier. The *Egres* processing by the cluster group is usually higher on weekday 6 than on weekday 1 and that is not related to intra-day volatilies.

Applying CEE we get additional, quantitative insights into contributions of individual subsystems (days) to the imbalance of the overall system.

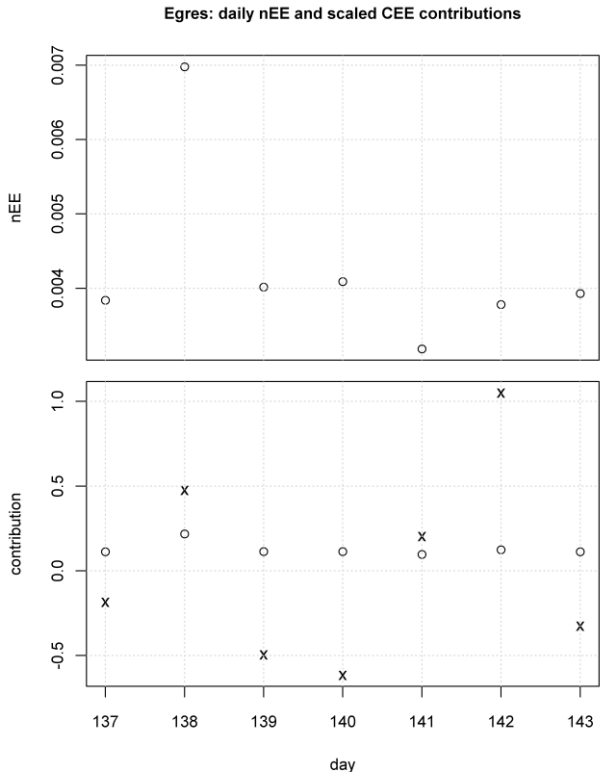

Figure 10 Excess entropy and composition of excess entropy for week 19; the circles in the bottom picture denote (scaled) contribution of intra-day imbalance, the crosses denote the scaled contributions of between-days imbalance

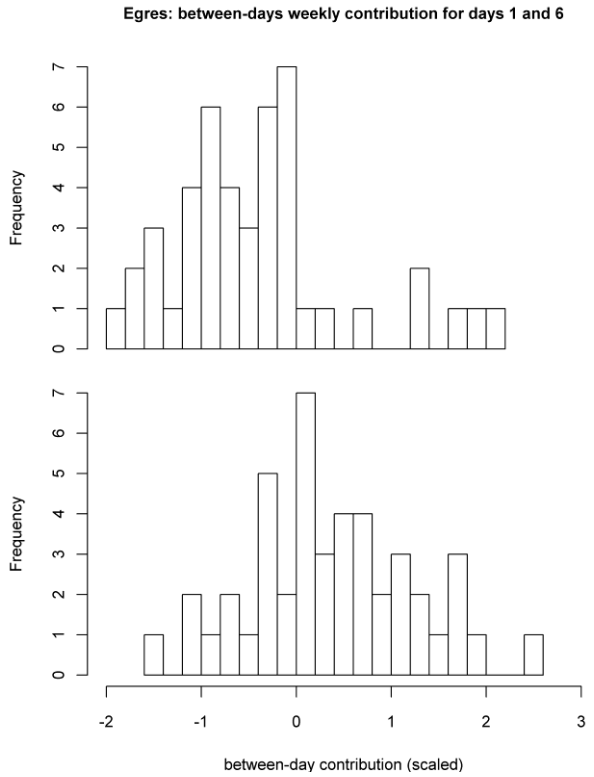

Figure 11 Distribution of between-days contribution to *CEE* from weekday 1 and weekdays 6.

5. Summary

As computer systems grow in complexity and size additional system descriptors need to complement the existing ones to manage the information explosion. We have introduced Excess Entropy (*EE*) as a way to characterize imbalance (or, equivalently, concentration of processing) in computer systems. A variant of *EE*, normalized excess entropy (*nEE*) enables comparison between different systems, even if these systems have different number of elements or are measured in different units. Another variant, composite excess entropy (*CEE*), allows hierarchical composition (and decomposition) of imbalance in subsystems into imbalance in overall system. *CEE* also allows discovery which subsystems contribute most to the overall imbalance – and why.

All variants of the *EE* are dimensionless and their normalized versions exist. That allows for quantitative comparison of usage profile between different types of resources – for example imbalance of *ComputeHours* can be compared with imbalance of *Egres*, even though both resources are measured in different units.

Excess entropy of subsystems can be composed into an excess entropy of the total system and quantify how much each subsystem contributes to the overall imbalance and whether their contribution comes from high internal imbalance or difference from other subsystems.

Operationally, the formulas for computation of all *EE* coefficients described here are straightforward. They can be implemented trivially in a spreadsheet and their computation time is trivial.

The useful operational interpretation of *nEE* is that of *concentration of resource usage*. Higher *nEE* values imply resource use by a small number of customers, lower *nEE* values indicate more even spread of resource use.

Excess entropy is not a silver bullet. It does not replace existing approaches to system description (like load, utilization, response time) – it also does not replace traditional statistics like average, standard deviation etc. Excess entropy adds to them and offers additional insights into the analyzed system. For example growth in load with growth in imbalance signals that only a small subset of clusters is getting more active. A growth in load with drop in imbalance signals broadly-based growth.

Imbalance describes only one aspect of the system – but this aspect has not been quantitatively analyzed earlier in computer systems (to the best of our knowledge).

The relevance of high or low imbalance in a system is heavily contextual. Depending on our goals and aspects of the system under considerations high imbalance can be good, bad or irrelevant. The change in imbalance, regardless of the level, signals change in the distribution of load, towards more (or less) concentrated.

Imbalance between server use in a group of servers managed by a load balancer is almost always a signal that something is wrong, so is imbalance in disk loads. However, when overall load is trivial then even high imbalance can be ignored – though we should check whether imbalance is growing with load or is independent of the load level. High imbalance in core usage for multicore servers may signal that affinity settings were used and a server working as intended – but it may also signal that an application is serializing on some software element, therefore its implementation should be reviewed.

Composite normalized imbalance gave us additional insights into the structure of imbalances in time, allowing us to identify unusual load on certain days as well as give a concise description of load variability in time.

The few applications of quantitative imbalance shown here represent only a small subset of the potential applications of this method.

For example we can consider relationships between imbalance of the hardware components of a server, a rack of servers, a cluster of servers, a datacenter, a region and the global datacenter set – thus composing the global from several hierarchical levels (instead of just two levels considered here). Another application would be analysis of imbalance at multiple time scales with detection at which time scale the imbalance picture starts to diverge – as this could be a signal of some underlying shift in the internal structure of usage and yield potentially useful information.

In this paper we have investigated application of imbalance to some aspects of resource usage in a Windows Azure compute clusters. In the companion paper [9] we demonstrated multiple applications of excess entropy to Windows Azure storage clusters. We have used EE there to describe the concentration of storage resource usage by a small subset of accounts and evolution of this concentration in time.

Description of systems by excess entropy can be applied to many areas – even a single server. It is, however, of particular interest when describing large systems like cloud computing – description by excess entropy scales up well with the number of elements and allows aggregated description of multi-element systems. The composability of system description by excess entropy also maps well into the complexity and hierarchical infrastructure of cloud computing.

6. REFERENCES

[1] F. Alexander Bais, J. Doyne Farmer, *The Physics of Information*, arXiv:0708.2837v2

[2] Cowell, F.A. (1995), *Measuring Inequality*, 2nd edition, Harverster Wheatsheaf, Hemel Hampstead

[3] Theil, H. (1967). *Economics and Information Theory*. Chicago: Rand McNally and Company

[4] Gastwirth, Joseph L. (1972). "The Estimation of the Lorenz Curve and Gini Index". *The Review of Economics and Statistics* (The MIT Press) **54** (3): 306–316. doi:10.2307/1937992. JSTOR 1937992.

[5] Sen, A. (1997). On Economic Inequality. Oxford: Clarendon Press.

[6] Loboz, C. "Cloud resource usage – extreme distributions invalidating traditional capacity planning models", *ScienceCloud'11*, June 8, 2011, San Jose, California, USA.

[7] Loboz, C., Lee, S., Yuan, J. "How do you measure and analyze 100,000 servers", Proceedings of the Computer Measurement Group 2009 International Conference, paper 9100

[8] Loboz, C., Lee S. "Capacity, usage, computer work – and daily analysis of 100,000 servers", Proceedings of the Computer Measurement Group 2010 International Conference, paper 5089

[9] Loboz, C. "Quantifying imbalance in computer systems" accepted for the Computer Measurement Group 2011 International Conference, paper 1132

[10] R project for statistical computing, www.r-project.org