RZ 3461 (# 93871) 11/11/02
Computer Science 8 pages

Research Report

Creating Services with Hard Guarantees from Cycle-Harvesting Systems

Chris Kenyon and Giorgos Cheliotis

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Creating Services with Hard Guarantees from Cycle-Harvesting Systems

Chris Kenyon and Giorgos Cheliotis

Abstract— **Cycle-harvesting software on commodity computers is available from a number of companies and a significant part of the Grid computing landscape. However, creating commercial service contracts based on resources made available by cycle-harvesting is a significant challenge for two reasons. Firstly, the characteristics of the harvested resources are inherently stochastic. Secondly, in a commercial environment, purchasers can expect the providers of such contracts to optimize against the quality of service (QoS) definitions provided. These challenges have been successfully met in conventional commodities, e.g. Random Length Lumber, traded on financial exchanges and we draw inspiration from there. The essential point for creating commercially valuable QoS definitions is to guarantee a set of statistical parameters for each and every contract instance. In statistical terms this is the difference between guaranteeing the properties of what is delivered versus the source from which the delivery will be made. In this paper we describe an appropriate QoS definition, Hard Statistical QoS (HSQ), and show how this can be implemented for cycle-harvested resources using a hybrid stochastic-deterministic system. We present an architecture and algorithms to support HSQ contracts. We analyze algorithm behavior analytically using a distribution-free approach versus the expected proportion of deterministic resources required for an HSQ specification. For example, where slot lengths are log-Normally distributed we find that for hard guarantees on 8 quantiles with contract sizes 16 to 1024 slots, from 13% to 1% deterministic resources are required. Permitting oversampling is relatively inefficient leading to up to 61% of the stochastic resources being wasted in a typical case. Including downwards substitution reduces deterministic resource requirements by roughly half. We conclude that commercial service contracts based on cycle-harvested resources are viable both from a conceptual point of view and quantitatively for contracts of sufficient size.**

Keywords— **Cycle-stealing, cycle-scavenging, QoS, Grid computing.**

I. Introduction

BUSINESS models in Grid computing around buying and selling resources across budget boundaries (within or between organizations) are in their very early stages. Cycle-harvesting (or -scavenging, or -stealing) is a significant area of Grid and cluster computing with software available from several vendors (e.g. Platform Computing, Avaki, Data Synapse, United Devices, Entropia)[1]. However, creating commercial contracts based on resources made available by cycle-harvesting is a significant challenge for two reasons. Firstly, the characteristics of the harvested resources are inherently stochastic. Secondly, in a commercial environment, purchasers can expect the sellers of such contracts to optimize against the quality of service (QoS) definitions provided. These challenges have been successfully met in conventional commodities, e.g. Random Length Lumber (RLL), traded on financial exchanges (the Chicago Mercantile (CME, www.cme.com) in this case) and we draw inspiration from there. The essential point for creating a commercially valuable QoS definition is to guarantee a set of statistical parameters of each and every contract instance.

In this paper we describe an appropriate QoS definition, Hard Stochastic QoS (HSQ), and show how this can be implemented for cycle-harvested resources using a hybrid stochastic-deterministic system where dedicated resources are added. We give an analytic quantification of the efficiency of our implementation algorithm in terms of the proportion of deterministic resources required, and also analyze two extensions offering potential advanages. Thus we offer support for transforming what today is a process of saving wasted cycles to a process of offering these cycles as a new commercial service with QoS guarantees. This new business model also enables resource trading in an important area of Grid computing.

For RLL the contract definition used on the CME is shown in see Figure (2). The definition does include some error tolerance in the statistical metrics guaranteed at the contract level. We include an analysis of this in terms of a distance metric $\|.\|$ between a vector of guarantees and a realization together with a permitted error d.

We focus here more on the resources themselves rather than user jobs. Parallel or single-job scheduling versus stochastic resources is also important (e.g. [1], [2], [3], [4], [5]) but here we address the complementary question of resource characterization, i.e. QoS of resources rather than of jobs. This is significant when trade is at the resource rather than at the job level. Markets and busines models supported at different levels (of the software stack in this case) are to be expected just as in other fields and we focus on one of the lower layers here.

Implementations of Hard Stochastic QoS are important for cycle-harvesting because they offer a way to provide commercially valuable guarantees about these resources. (The definitions and terms used in this paper are collected together in the next section.) However, implementing HSQ by adding dedicated resources offers advantages beyond the basic guaranteeing at the sample level of population-level QoS metrics. Two extensions are possible: the QoS-metric-shaping and QoS-metric-extension. Shaping refers to offering HSQ on a metric that is different but still feasible from the cycle-harvesting population. Extension refers to offering HSQ on a metric that is not feasible from the cycle-harvesting population. We illustrate these concepts with a

The authors are both with IBM Research, Zurich Research Laboratory, Rüschlikon, Switzerland. E-mail: {chk|gic}zurich.ibm.com.

[1]www.platform.com, www.avaki.com, www.datasynapse.com, www.ud.com, www.entropia.com

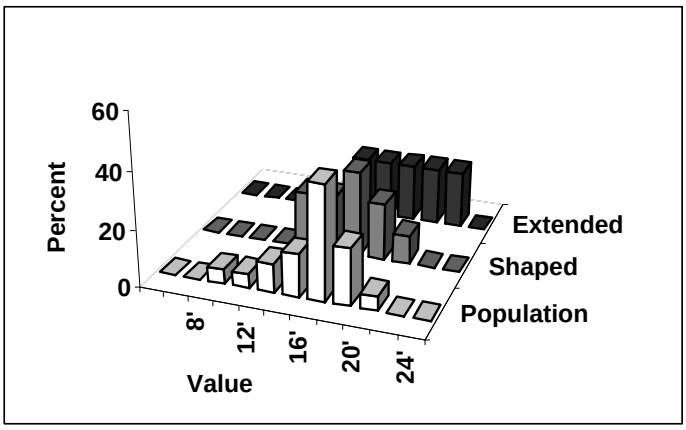

Fig. 1. Probability distribution functions (pdf) for example resource. The Population entry gives the observed pdf for the whole resource population. The Shaped and Extended pdfs describe possible additional pdfs that can be guaranteed via HSQ (see text for details).

simple example.

Consider a population-level QoS metric which has a (long-run observed) probability distribution function (e.g. length of uninterrupted time on a machine, which we call a "slot") as shown in Figure (1), "Population" entry. A simple HSQ implementation would guaranteed this distribution, say at an appropriate level of quantization, for each sample (of a specified number of slots, i.e. a contract). An HSQ implementation capable of shaping could also guarantee the distribution shown as "Shaped", again for every sample (i.e. contract). An HSQ implementation capable of extension could also guarantee the pdf "Extended". This is possible by using added dedicated resources noting that the infeasible slots from the population are fulfilled wholly from these dedicated resources. Whereas the feasible slots are in general fulfilled from a combination of both sources.

Time and deadlines play an important part in resource contract guarantees. With a sufficiently loose deadline HSQ can be supported with only cycle-harvested resources. However in general this is extremely inefficient because resources must be provisionally used and then if they do not fit the requirements, discarded. We will analyze this in detail.

This paper is organized as follows: in the next section we define the notion of resource and HSQ; after that previous work; the following section contains quantifies the need for HSQ; after we describe the method for supporting HSQ including system architecture, algorithms and algorithms quantification and examples; and the last section concludes.

II. Terms and Definitions

We collect together some concepts used throughout the paper in this section:

- *Stochastic resource*: cycle-harvested machine (time available only when machine is idle)
- *Deterministic resource*: dedicated machine (time available upon request)
- *Slot*: uninterrupted time on a machine, sometimes also called simply resource
- *Population*: the entire set of slots on stochastic resources
- *Sample*: a set of slots taken from the population to fulfil a contract

Definition 1 (Stochastic QoS) [SQ]. This is a QoS parameter or set of QoS parameters that is based on statistical properties of some QoS metric.

Definition 2 (Hard Stochastic QoS) [HSQ]. This is a QoS parameter or set of QoS parameters that:

1. is based on statistical properties of some QoS metric and that
2. is guaranteed with certainty

Note that in the implementation we will focus on guaranteeing quantiles of a non-deterministic QoS property. Guaranteeing quantiles versus moments of a distribution has the advantage that any given distribution can be described intuitively. Our proposed method can also be applied towards guaranteeing moments of a distribution, although this is not addressed in this paper.

III. Previous Work

In many IT fields, most notably networking, it is common to design QoS mechanisms that are based on statistical measurements, but the most popular approaches either target statistical guarantees for aggregates of demand (network flows) [6] or require the reservation of resources for each individual flow [7]. The difference here is that the statistical properties are guaranteed at the contract level where contracts are for small numbers of entities and independently for every single contract, with certainty. In networking statistical guarantees commonly deal with packets and hence only consider large samples, say 10^6, whereas we are interested in guarantees for samples three or more orders of magnitude less.

Outside of IT metrics the idea of guaranteeing statistical parameters is known. For example, in commodity markets this is found. Consider the section of a contract for Random Length Lumber (RLL) from the Chicago Mercantile Exchange Inc. (CME) rulebook (http://www.cmerulebook.com)

The CME gives no advice as to how this is to be guaranteed, only that it must be. This is an example of the sort and strength of guarantees required for commodities to have tradable value in a commercial environment.

In simulation there are various variance reduction techniques known that we build on here and adapt in several ways [8], [9]. These describe mathematical methods of how to reduce the variance of various measurements of interest (e.g. output simulation parameters), or to increase repeatability or comparability, in a stochastic setting. It is not known today how to apply, implement, or adapt these for cycle scavenging. Nor have these techniques been previously applied in the area of guaranteeing quality of service. This is what the current paper addresses.

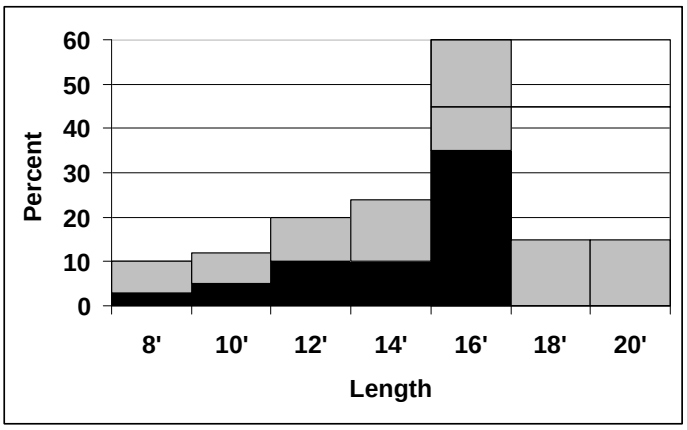

Fig. 2. Probability distribution function (pdf) guaranteed under Chicago Mercantile Exchange rules for Random Length Lumber (see http://www.cmerulebook.com for full details). The pdf is discrete with guaranteed minimum and maximum percentages in each length category. For the 16'+18'+20' combined category there are additional limits (45% and 60%) complementing the individual category limits.

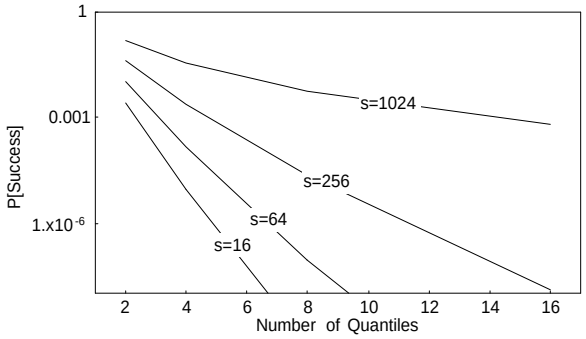

Fig. 3. Probability of successfully guaranteeing quantiles by random sampling for increasing contract size $s = 16, \ldots, 1024$ with an error tolerance d of 1% of the contract size (Equation(4)). We may expect HSQ creation methods to be required for all probabilities less than, say, 95%. Note that the plot is semi-logarithmic.

IV. Problem Quantification

We have stated that HSQ is valuable but we have not yet quantified the mismatch between sample-level guarantees (i.e HSQ) and population level guarantees. Is special support for HSQ necessary or is random sampling from a guaranteed population close enough to HSQ anyway? Clearly we can find situations at both extremes if we look hard enough. Given a large enough sample size one would imagine that we could provide any arbitrary level of HSQ. However, this is only correct provided the HSQ required is not taken relative to the sample size — if it is then quite the opposite result holds. For an HSQ that is not defined relative to the sample size the interesting question is: what sample size is sufficient?

What we will do is show first that the problem arises with very high probability when quantiles are guaranteed (as for the RLL definition that is in commercial use) and we will do this using a distribution-free analysis method. Secondly, for a given example distribution, we will plot the magnitude of the mismatch for the first two moments against the sample size.

A. Existence

Suppose that q quantiles are guaranteed and the sample size is also q. What is the probability that we have one sample in each quantile of the distribution? This is equivalent to the famous Birthday Problem (given a class of students, what is the probability that any two have the same birthday) where the number of quantiles is the number of students and the number of days in a year. The probability of providing the required HSQ (i.e. 1 sample in each quantile) is thus:

$$P[\text{success}] = \left(\frac{q!}{q^q}\right) \tag{1}$$

If we have more samples than quantiles, say nq samples and q quantiles and we allow a difference d between the actual vector of quantiles and the desired vector of quantiles (which is just n entries in each quantile) as measured according to some norm $||.||$ then the probability of success is

$$\begin{aligned} P[\text{success}] &= \sum_{||\vec{k}-\vec{n}||\leq d} \frac{(nq)!}{k_1!k_2!\ldots k_q!q^{nq}} && (2)\\ &\approx \int_{||\vec{k}-\vec{n}||\leq d} N(\vec{n}, n_{\{i=j\}} - n/q) && (3)\\ &= IG(q/2, (d^2/(n-n/q))/2) && (4) \end{aligned}$$

where of course $\sum k_i = nq$, we are simply summing over the cases of interest, using a least-squares (L_2) norm, and using a multinomial probability with equal probabilities of success for each outcome, i.e. quantile (the reader can check that it reduces to the previous formula for $n = 1$, $k_i = 1\ \forall i$).

We approximate Equation (2) first with a multi-Normal(mean, variance) in Equation (3)[10] and then express the probability volume within $||.|| < d$ using an incomplete Gamma function $IG(a, x) = \gamma(a, x)/\Gamma(a)$ to obtain Equation (4)[11]. Note that the last step is valid whatever the covariance structure of the multi-Normal.

Figure (3) gives examples of Equation (4). We consider the probability of satisfying HSQ for different contract sizes from $s = 16$ through $s = 1024$ guaranteeing increasing numbers of quantiles. The error level d permitted is taken as 1% of the contract size. Clearly 1% of contract sizes of 16 or 64 is non-integer but the multi-Normal approximation can handle this. Note however that for small numbers of samples per quantile that the multi-Normal is only approximate. As expected larger contracts (more samples per quantile) make it easier to attain the fixed QoS requirement. However the probability of satisfying the quality requirements with random sampling is basically zero. Interested readers can use Equation (4) to generate other examples as needed.

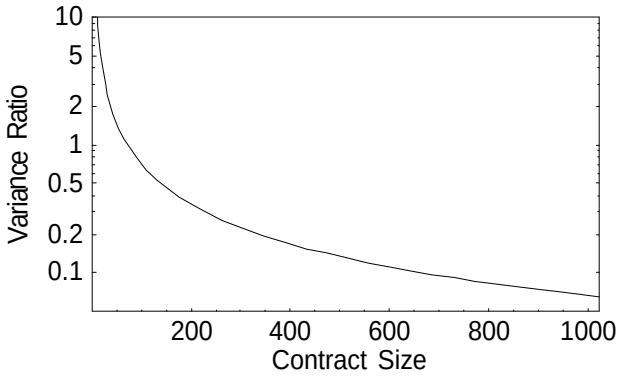

Fig. 4. Variance of the observed variance for random sampling from a Normal(4,1) distribution, Equation(5). The ideal variance ratio is zero.

This analysis is valid when quantiles are guaranteed for *any* distribution: the analysis has no dependence on the underlying distribution. However it does treat different quantiles as though no substitution between them were possible. Whilst this is certainly true in some cases, in other cases getting a longer time a machine may be perfectly acceptable: i.e. downward substitution will be acceptable in some situations.

B. Magnitude

In the section above we showed how often we may expect HSQ to be required in a distribution-free manner. In this section we examine, for a particular distribution, the magnitude of the error for a distribution-sensitive metric. Specifically, if the underlying population distribution is Normal and we take a sample at random what can we expect of the first two moments as opposed to what we would like to guarantee in this example, i.e. the population moments.

We can calculate the variances of the observed quantities under random sampling since analytic formulae are available for the first two moments of the Normal distribution. Ideally, with HSQ, the variances of the moments of the samples would be zero: all samples would have the guaranteed moments exactly. However in practice we will have:

- variance of the observed mean
$$= \sigma^2/n$$
- variance of the observed variance
$$= 2(n-1)(\mu^4 + 2n\mu^2\sigma^2 + n\sigma^4)/n^3 \quad (5)$$

The variance of the observed mean converges linearly to that of the population and that of the observed variance does so quadratically. However, although the rates of convergence are encouraging it is worthwhile to plot Equation(5) for an example. Let us take the case where we have a mean of 4 with a variance of 1, see Figure (4).

If the value of the cycle-scavenged resource slots is sensitive to the observed variance (or spread) then just relying on random sampling is potentially a very poor approach, as shown by Figure (4). The variance of the observed variance is high, even with relatively large sample sizes (n=256) and a relatively tight underlying distribution (coefficient of variation 0.25, this is ratio of standard deviation to mean). We need something better.

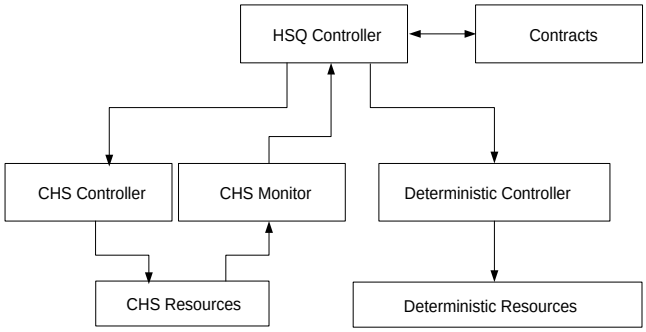

Fig. 5. HSQ System Architecture combining Cycle Harvested System with deterministic (dedicated) resources with appropriate monitoring and control.

V. Method

In the previous section we quantified the size of the problem, i.e. the ability of non-HSQ systems to provide HSQ. In this section we describe: a system architecture for supporting HSQ; algorithms for implementing HSQ; and a performance analysis of the HSQ algorithms.

A. HSQ System Architecture

The objective of this architecture is to provide hard stochastic quality of service for cycle harvesting systems in order to make their offerings (packages of available time) more commercially valuable. An architecture like this, at least to the point of combining stochastic and dedicated resources (but without the controller features as described here) is already present in some Grid systems such as Condor and an offering from Platform Computing. Thus this system is practically realizable provided the controller is implemented and the control and monitor functions are put in place with an appropriate scheduler.

The HSQ system architecture comprises five things:

1. a HSQ controller
2. a pool of stochastic (harvested) resources
3. a pool of deterministic (dedicated) resources
4. monitoring of the stochastic pool of resources
5. control of the stochastic resource pool

We assume that monitoring and control of the deterministic resources is present. This system is used to fulfill contract obligations as described by HSQ terms. We will next describe each part of the system shown in Figure (5) in detail.

The basic idea is that the HSQ Controller monitors the Cycle Harvesting System (CHS) resources, using the CHS Monitor, as applied to each Contract and takes one or more of two possible actions:

1. Sends an artificially "harvested resource end" to a particular CHS resource. Thus execution ends on that re-

source.
2. Diverts contract obligations from the CHS resources to the Deterministic Resources, under control of the Deterministic Controller, with appropriate instructions for their execution.

This system can deliver HSQ with 100% certainty for an arbitrary collection of contracts provided the HSQ controller is used as the access control for contracts. In addition it can deliver a parameterized approximation, where this is parameterized by the certainty level desired. Both of these can be delivered in a much cheaper fashion than would be possible with system designs that did not combine stochastic and of deterministic resources appropriately. This "appropriate" is described by the HSQ controller design.

To give a simple example the HSQ Controller can monitor obligations of contracts and when it realizes that a contract cannot be completed with certainty before the deadline specified in the contract (we assume that all contracts have a finish-by time defined) using the cycle scavenging system it transfers the rest of the contract fulfillment to the deterministic resources under its control. These resources are called deterministic precisely because they are under the exclusive control of the HSQ Controller. In general each statistical QoS metric that is guaranteed with certainty will result in at least one constraint, often several. When the HSQ Controller observes that any of the constraints resulting from a contract is reached, it transfers the contract fulfillment as described earlier. In general the constraints will be dynamic, i.e. they will change as time and events occur so they must be continuously calculated and updated.

Given that each contract results in a set of dynamic constraints the application to contract acceptance is direct. For a potential contract calculate the constraints that it gives rise to and if they are unfeasible do not accept the contract. If there is a window of possible acceptance for the contract then continue to monitor the dynamic constraints of the contract until either the window closes or the contract can be accepted.

B. HSQ Examples and Algorithms

In this section we provide examples of HSQ guarantees and the algorithms to deliver them. We pick two examples: a very simple case where only the number and average length of slots in a contract are guaranteed; a general case where the number and the quantiles of the distribution of slots are guaranteed. The first case serves to build intuition and the second as a realistic example.

B.1 Average Length Guarantee

The contract guarantees that for a set of s time-slots: the average of the time available per slot will be at least A; the contract will be fulfilled before T; and zero-length slots are permitted.

To fulfil the contract start up slots as soon as possible until s slots have been started. However, if at any point we have:

if (time left before T) = (sA/minimum(s, total number of slot instances available on deterministic resources)) - time made available)

then transfer to deterministic resources. When the total time made available = sA) finish. The contract has been fulfilled.

Basically what we do is to keep starting slots until the only way to satisfy the contract with certainty is to use deterministic resources — at which point they are used.

B.2 Quantile Guarantee

For simplicity we assume that the quantiles guaranteed reflect the population distribution, we are not considering shaped or extended distributions here. We assume that q quantiles are guaranteed, the contract ("sample") size is $s = nq$ with deadline T. harvest stochastic resources from and m_d dedicated machines. We consider here only resources from these machines that are not already committed to other contracts.

The following algorithm (Q1, next page) is one of the simplest possible, it just keeps starting slots up to a maximum number s until either it realizes it won't finish in time or its used up all its permitted slot starts and then finishes off with deterministic resources. Note that s may be different from $n \times q$, to allow for oversampling.

The algorithm uses two extra methods which we do not describe here since they depend on the existence and features of an external scheduler for the deterministic resources. Method isSchedFeasible() just asks the deterministic scheduler whether it can finish off successfully in the time remaining (returning either True or False). Method RunSched() commands the deterministic scheduler to actually deliver the remaining slots in the contract. ϵ is used as a safety measure, to aim to finishing just before the deadline given that we have the loop still to do before we next check this condition. Both methods also use the error metric and the permitted error to choose the cheapest way to finish the contract but we have not brought out these details explicitly.

C. HSQ Algorithm Performance Analysis

We motivate our performance analysis with a business question. Given that resource packages are being traded and the owner of the HSQ system sees an Ask (i.e. a request for resources) with a price and a deadline what is the cost to support that request? If it is below the price then the HSQ system owner can respond directly to the Ask. If not then the owner can post an alternative price.

There are two interesting performance metrics: percentage of deterministic resources required relative to the whole contract; and percentage of stochastic resources wasted because of duplication. In our setup and with the algorithm Q1 we have little direct control of the second metric because we do not permit downward substitution, this would be a useful extension (but outside the current scope). Thus we will concentrate on the first metric which we call $R(s, T)$, clearly this is a random variable and we calculate

Algorithm 1 [Q1](s) To provide q quantiles of distribution D with n entries per quantile, before deadline T tolerating an error level d as measured by metric $\|.\|$ using at most s stochastic resources

Set number of started, finished slots to zero, $n_s = 0$, $n_f = 0$
if NOT(isSchedFeasible(T − (elapsed time + ϵ), $s - n_f$, $\vec{n} - \vec{k}$)) **then**
  Report "Schedule infeasible"
  Stop
end if
while isSchedFeasible(T − (elapsed time + ϵ), $s - n_f$, $\vec{n} - \vec{k}$)) **do**
  if A slot is available and ($n_s \leq s - 1$) **then**
    Start a slot, n_s++
  end if
  if A slot finishes **then**
    n_f ++, record which quantile the slot populated, discarding excess: k_i++
  end if
end while
RunSched(T − elapsed time, $s - n_f$, max($\vec{n} - \vec{k}, \vec{0}$)
Stop.

its expected value $E[R(s,T)]$. This expectation is most important metric when a system is supporting many contracts and thus benefits from the lack of correlation between them. At least for simple algorithms in the style of Q1 the correlation between their effects will be low thus making this the main figure of merit.

Clearly, for a sufficiently tight deadline only deterministic resources will be used, for example where the deadline is equal to the length of a slot in the last quantile, so $R(s,T)_{\max}(Q1) = 1$. A mixture could be used but this would require a modification of Q1 which we are not considering.

For a sufficiently loose deadline we can run $Q1(s)$ and it will even have spare time. This gives the proportion of deterministic resources required for Q1, $R(s,T)_{\min}(Q1)$:

$$E[R_{\min}] = \sum P[\text{observation}]R[\text{observation}] \sum\left(\frac{s!}{q^s\prod k_j!}\right)\left(\frac{\sum_j max(n-k_j,0)l_j}{\sum_j nl_j}\right) \quad (6)$$

where the first sum is taken over all $\{k_1,\ldots,k_q|\sum k_j = s\}$, i.e. over all possible observations where there are s samples, an observation is just a set of s random samples; l_j is the length of the slot for the j^{th} quantile.

We can evaluate Equation (6) for a specific example distribution to get an idea of the average loose deadline behavior of algorithm Q1. This should not be taken as the minimum proportion of deterministic resources since Q1 is a very simple algorithm. Hence the average loose deadline behavior of Q1 may be usefully indicative of what is achievable in practice even with tighter deadlines for more sophisticated algorithms. Of course deadline tightness is also relative to the number of machines (or slots processes) dedicated to each contract etc. Examining the loose deadline behavior factors out these influences.

We model stochastic resource slot length as having a log-Normal distribution with parameters (1.0,0.4) so the mean is 3.0 and the variance is 1.5. Figure (6) plots the expected proportion of deterministic resources required to support increasing QoS requirements for increasing contract sizes. For, say, 8 quantiles the required proportion varies from 4% to 25% for contract sizes of 1024 and 16 respectively.

One simple modification of algorithm Q1 is to take more samples (oversample) than are numerically required in order to attempt to satisfy QoS requirements and reduce the quantity of deterministic resources needed. Figure (7), *top line* shows how oversampling reduces the determinitic resource requirements: the relationship is roughly linear on semi-log axes. For 8 quantiles and a contract size of 64 the deterministic resource requirements can be reduced from about 13% to 1% by oversampling 60%. This does however imply that roughly 61% of the stochastic slot time is wasted. Depending on the relative value of stochastic and deterministic resources this may be a viable strategy.

A further modification of algorithm Q1 is to permit downwards substitution as well as oversampling, i.e. a longer slot is an acceptable substitute for a shorter one (but not vice versa). This is shown by the *bottom line* in Figure (7). In general downwards substitution reduces the deterministic resource requirements by roughly half.

We have stated that we are analysing the loose deadline case: this raises the question "what is a sufficiently loose deadline?" which we can answer using renewal theory. Suppose that the inter-arrival finishing time of slots has a pdf $F(y)$, then the pdf for the s^{th} finish is given by the s-fold convolution of $F(y)$ with itself. The remaining time that the deterministic resources require is a function of the number of dedicated machines and their free time, say DR(requirements). However we can work out the distribution of the remaining slots from terms in the formula above to obtain for the special case of memoryless distributions (e.g. Poisson arrival processes) the total as:

$$T_{\text{loose}} = F*\ldots*F + \sum \frac{s!}{q^s\prod k_j!} DR(max(\vec{n}-\vec{k_j},\vec{0})) \quad (7)$$

In an intermediate situation between loose and tight deadlines we are interested in how many samples the algorithm is actually able to take before it turns to deterministic resources and of these samples how many are useful and what their distribution is. In this case the number of slots to get from the stochastic resources s was overoptimistic given the deadline of the contract. Clearly if we get a lot of samples within a given time we expect their average length to be low and vice versa. This is a straightforward extension of the analysis here using simulation.

VI. Summary and Conclusions

The motivation for this paper was the desire to see whether it was possible to turn cycle-harvested resources into something commercially valuable. Was it possible to

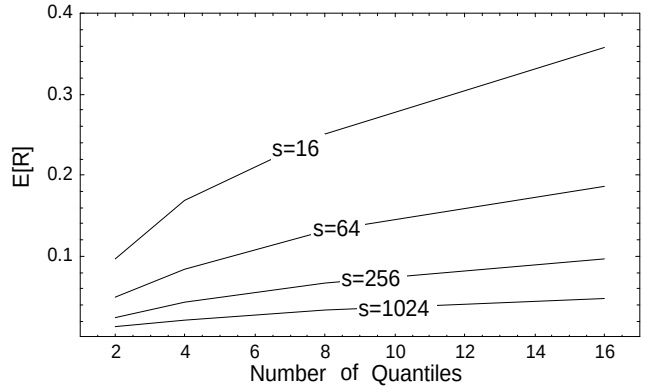

Fig. 6. Expected proportion of Deterministic resources required ($E[R]$, linear scale) to support hard QoS guarantees on delivered quantiles for increasing numbers of quantiles and increasing contract size

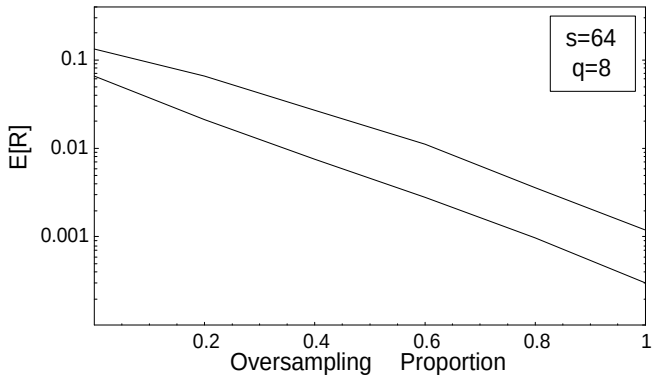

Fig. 7. Expected proportion of Deterministic resources required ($E[R]$, semi-log scale) to support hard QoS guarantees on delivered quantiles with oversampling for contract size $s = 64$ and $q = 8$ quantiles. Oversampling proportion of 1.0 means that $2 \times s = 128$ samples are used. *Lower line* shows the requirements when downwards substitution is permitted (see text for details).

define a contract that was valuable given that the underlying resources are inherently stochastic and that in a commercial environment we can expect the provider of a QoS guarantee to optimize against it? We wanted to avoid all arguments based on long term reputation because we were interested in having valuable (tradeable) individual contracts for cycle harvested resources — not the question of how to create valuable reputations for the providers.

In this paper we presented an appropriate QoS definition for cycle-harvested resources, Hard Statistical QoS (HSQ), based on statistical QoS metrics guaranteed at the contract level. That is, these metrics are guaranteed for each and every contract. Statistically this is the difference between guaranteeing the properties of a sample versus guaranteeing the properties of the population from which the sample was drawn. In business terms this means that each contract can have a consistent value — not just a collection of contracts or a particular provider.

We have shown how HSQ can be implemented for cycle-harvested resources using a hybrid stochastic-deterministic system. We presented an architecture, and algorithms, to support HSQ contracts. We analyzed the algorithm behavior analytically using a distribution-free approach in terms of the expected proportion of deterministic resources required to support a given HSQ level. Thus our analysis can be applied whatever the details of a particular system.

For a particular HSQ example where time slot lengths were log-Normally distributed we found that to provide hard guarantees on 8 quantiles for contract sizes of 16 to 1024 slots from 13% to 1% additional deterministic resources were required. Permitting oversampling, for example for a contract size of 64, was relatively inefficient, leading to up to 61% of the stochastic resources being wasted. Including downwards substitution (i.e. accepting longer time slots as valid substitutes for shorter ones) reduced deterministic resource requirements by roughly half.

Our analysis and method relies on being able to discard time slots that fall outside a contract definition. Most jobs currently run on cycle-harvesting systems are self contained (even when, for example, created by parameter sweep scripts) so this is valid. If each job changed some global state in an unpredictable and irreversible way then this would not be a correct analysis. Examples of this nature include certain transaction processing and web applications which do not have roll-back facilities.

We have not fully exploited the potential interactions between contracts, nor have we considered any structure within a contract as may be required for particular classes of parallel programs. Instead we have simply considered a situation where a provider is supporting many contracts so that the provider only needs to have the average quantity of deterministic resources for HSQ on a per contract basis. This assumes that the slot lengths of the stochastic resources are not correlated. In practice we will expect correlation when considering short time-windows, e.g. situations like "overnight" or "morning" or "afternoon". However for longer time-windows, or time-windows within a single situation the correlation structure may be unimportant. A further analysis could extend the current work to cases where there are arbitrary correlation structures, modelling and simulating these in a copula framework [12].

We conclude that commercial service contracts with hard statistical QoS guarantees (HSQ), based on cycle-harvested resources are viable both from a conceptual point of view and quantitatively with only small (1% – 10%) requirements for deterministic (dedicated) resources.

References

[1] A.L. Rosenberg, "Optimal schedules for cycle-stealing in a network of workstations with a bag-of-tasks workload," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, no. 2, pp. 179–191, February 2002.

[2] E. Heymann, M.A. Senar, E. Luque, and M. Livny, "Evaluation of strategies to reduce the impact of machine reclaim in cycle-stealing environments," IEEE 1st International Symposium on Cluster Computing and the Grid, May 2001, pp. 320–328.

[3] K.D. Ryu and J. Hollingsworth, "Exploiting fine grained idle periods in networks of workstations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 683–699, 2000.

[4] A.L. Rosenberg, "Guidelines for data-parallel cycle-stealing in networks of workstations, ii: On maximizing guaranteed output," IEEE 10th Symposium on Parallel and Distributed Processing, April 1999, pp 520–524.
[5] S.T. Leutenegger and X-H. Sun, "Limitations of cycle stealing for parallel processing on a network of homogeneous workstations," *Journal of Parallel and Distributed Computing*, vol. 43, pp. 169–178, 1997.
[6] J. Bennet, K. Benson, J-Y Le Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Kalmanek, K. Ramakrishnam, and D. Stiliadis, "An Expedited Forwarding PHB," Internet Draft, http://www.ietf.org, 2001.
[7] S. Schenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," Internet Draft, http://www.ietf.org, 1997.
[8] A.M. Law and D.W.Kelton, *Simulation Modeling and Analysis*, chapter 11, Variance-Reduction Techniques, pp. 581–617, McGraw-Hill, New York, 3rd edition edition, 1999.
[9] J.C. Hull, *Options, Futures, and Other Derivatives*, chapter 18, Numerical proceedures, pp. 414–418, Prentice Hall, New York, 5th edition edition, 2002.
[10] A. Carter, "Le cam distance between multinomial and multivariate normal experiments under smoothness constraints on the parameter set," Technical report, University of California, Santa Barbara.[http://www.pstat.ucb.edu/carter], 2001.
[11] C. Walck, "Handbook on statistical distribution for experimentalists," Internal Report SUF-PFY/96-01, Stockholm, 2000, [http://www4.tsl.uu.se/ tord/Stat_Grad/suf9601.ps], pages 96–102.
[12] R.B. Nelsen, *An introduction to copulas*, vol. 139 of *Lecture Notes in Statistics*, Springer, New York, 1998.