A Service Model for Development and Test Clouds

Debdoot Mukherjee[1], Monika Gupta[1],
Vibha Singhal Sinha[1], and Nianjun Zhou[2]

[1] IBM Research – India
{debdomuk,monikgup,vibha.sinha}@in.ibm.com
[2] IBM TJ Watson Research Center
jzhou@us.ibm.com

Abstract. A Development & Test Cloud (DTC) enables IT service enterprises to host standardized configurations of just about any tool-set on cloud – the hosted software need not be designed for multi-tenancy and they may come from a multitude of vendors. However, since most enterprise software are available only under perpetual licenses, DTCs cannot become truly pay-per-use – customers of a DTC have to upfront purchase software licenses. This paper proposes a service model for a DTC vendor wherein the vendor purchases software licenses and recovers the cost from its clients based on their period of usage. Our model allows the vendor to maximize returns from a purchased license by using it in multiple projects separated in time. We set up an optimization problem to decide how best a DTC operator can invest in buying software licenses such that it gets maximum opportunity to resale purchased licenses. We conduct empirical studies to validate the feasibility and usefulness of our approach. Also, we enlist characteristics of tool-sets that make them profitable for the DTC vendor.

1 Introduction

Even as the economy recovers from the downturn, IT services enterprises continue to cut down on all forms of operational expenditure so that receding profit margins of services contracts do not affect their balance sheets adversely. All large companies with massive, geographically distributed workforces are upset with burgeoning IT support costs, under-par utilization of hardware resources and sub-optimal management of software licenses. Again, they wish to improve productivity of their personnel by equipping them with the latest developer toolsets, which often require advanced hardware configurations to run effectively. A cloud based service delivery environment addresses the above issues and offers many interesting possibilities toward shaping the next generation IT services enterprise. Using a high performance cloud platform for hosting development and test environments, not only reduces IT infrastructure and support costs drastically but also helps to streamline delivery by provisioning pre-configured, standardized toolsets and leads to significant improvements in developer productivity. Moreover, it empowers lines-of-businesses (LOBs) in an enterprise with extreme agility to contend changing market realities; they can easily scale up or scale down their IT infrastructure because they do not incur any capital expenditure to own hardware/software but simply pay a price as per their usage.

A *Development & Test Cloud (DTC)* comes across as a unique offering specifically designed to ensure application development and maintenance activities can move

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 523–531, 2011.

to the cloud. A DTC is a service environment that can automatically provision pre-configured, integrated sets of software on hardware configurations chosen by the user (See our technical report [5] for details on DTC use-cases, architecture and benefits). It can turn-around defect-free, ready-to-use development and testing environments within minutes; thus results in faster time-to-market of deliverables as well as lower idle times for project personnel. Initial pilots [7,5] of Development & Test Clouds have shown drastic reduction in provisioning overheads, elimination of configuration defects and improvement in developer productivity. However, current DTC implementations force their customers to upfront purchase licenses for most software. Very seldom, one finds software being rented in a pay-per-use manner – mostly limited to cases where the software comes from the cloud vendor itself. This poses a serious issue for enterprise application development since service engagements typically leverage software coming from a multitude of technology vendors. Clearly, in such a scenario, the promises of lower software costs and easier scaling of usage levels will not be realized – the licenses have to be purchased at the same rates as they are available for lifelong standalone use. Hence, the adoption of DTCs may be hit. In fact, a 451 Group report[1] and Lori [4] point out that old models of software licensing are entirely incompatible with cloud computing environments and this fact proves to be a severe roadblock for greater cloud adoption.

In this paper, we propose a service model whereby the DTC vendor purchases all software licenses and recovers the cost from its clients based on their period of usage. Our model allows the vendor to maximize returns from a purchased license by using it in multiple projects separated in time. We set up an optimization problem to decide how best a DTC operator can invest in buying software licenses such that it gets maximum opportunity to resale purchased licenses. Also, we empirically study characteristics of tool-sets that can lead to profitable DTC hosting.

The main contributions of the paper include:

1. Design of a service model for a DTC operator that optimally transforms costs incurred in buying software licenses to pay-per-use prices. The model guides the DTC vendor to purchase a set of software so that it can maximize returns (Section 2).
2. Empirical evaluation of the DTC service model to demonstrate its feasibility and an evaluation of heuristics that can decide whether a toolset is a preferred candidate for stocking in a DTC (Section 3).

2 DTC Service Model

We propose a service model for a Development & Test Cloud (DTC) offering that helps a DTC vendor to decide which software *appliances*[2] to stock on a DTC and how to

[1] http://www.informationengineer.org/2010/02/06/the-451-groups-cloud-computing-outlook-for-2010.html

[2] An *appliance* is a common set of software that when installed and tuned to a certain configuration can support development and testing activities across service engagements of a particular kind. For example, SOA engagements may always use an integrated appliance consisting of certain software from Websphere stack.

price them (per unit usage) in order to run the operations most profitably. We suggest the following scheme whereby the DTC provider purchases licenses and the end-users pay a just fee per their usage:

1. A DTC provider purchases licenses of different kinds of software and collects the same in *license pools*.
2. Every time an appliance is provisioned for a client, each software in the appliance is assigned with a license available in its pool. A fee for license usage, which computed as per the proposed model, is bundled into the appliance cost. The user may have to pay a premium price (higher than the fixed rate, possibly close to the license cost) only if there are no licenses available in the pool for a particular software.
3. The licenses are returned back to the pool after the appliances get de-provisioned.

Now, the DTC vendor wishes to keep just enough licenses in the pool to serve demand for appliances at any point of time. Also, it is desirable that a license once purchased finds use in several projects over the course of time. Greater license reuse across projects separated in time will bring down the fees paid by the end-user and enhance the DTC vendor's profitability. Thus, we have the following problem:

> **Problem Definition:** *How can the DTC enterprise effectively invest a fixed amount of capital to buy licenses of software present in common appliances and then appropriately price the appliances in a pay-per-use model, based on available demand forecasts?*

At first sight, one may relate the above problem to the standard problem of inventory management [6] – how much goods do you stock in your inventory so that you do not run out of materials when you need them? Turns out, the drivers for these two problems are quite different! The reason for maintaining an inventory of goods is to avoid shortage costs. The time taken to refill stock after placing an order is generally significant, thus replenishment is ordered in advance. For purposes of our problem, it can be assumed that an order for a new license is served instantaneously and thus, shortage costs are not applicable. However, the strategy of purchasing licenses every time a provisioning request arrives is not optimal. We want licenses to be reused to increase profitability; therefore, we wish to invest in purchase of only those licenses for which we expect sufficient future demand. Again, buying licenses in advance may help save money if price increases are common. We set up an optimization problem to determine the number of licenses of each kind of software that should be purchased in order to maximize the return on investment for the DTC vendor. Solving such a problem also helps us ascertain the price that can be set for each appliance or software usage.

We consider a finite set, $A = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$, composed of appliances that are sought after in typical service engagements. An appliance is a set of software, $\alpha_i = \{S_1, S_2, \ldots S_m\}$, with pre-built configurations commonly used in a particular form of engagement. It is assumed that engagements using a certain appliance, α_i, have similar duration. If not, new appliances are created in A such that we enforce the standard deviation of durations of all projects using a single appliance to be small. Furthermore, we conjecture that solutioning teams in service enterprises have the engagement pipeline data, which gives demand forecasts for each engagement type.

Our service model works with the following inputs:
Δ_i : Mean project duration of engagements using α_i
D_i : Demand for α_i as a function of time
T : Time period for which all price calculations are made
F : Capital that may be invested in license purchase during time interval $[0, T]$

We introduce the notion of a license unit for an appliance. One license unit for α_i includes one license each for every software S_j contained in it. Again, in our model, time t can take up discrete values in the time interval $[0, T]$. In practice, a time period T of a quarter or a year may be discretized in terms of the different weeks or months in them. Suppose,
χ_i : Number of license units of α_i purchased at $t = 0$
$L_i(t)$: Number of license units of α_i available in pool at time t
$U_i(t)$: Number of license units of α_i taken out of pool for use in projects starting at time t

As mentioned before, at any point of time, license units move out from the pool and get assigned to projects. Again, the license pool gets augmented by licenses from projects that have just ended. Thus, we can write:

$$L_i(t+1) = \begin{cases} L_i(t) + U_i(t - \Delta_i) - U_i(t), & t \geq \Delta_i \\ L_i(t) - U_i(t), & 0 < t < \Delta_i \\ \chi_i, & t = 0 \end{cases} \tag{1}$$

Solving the above recurrence relation we get:

$$L_i(t) = \begin{cases} \chi_i - \sum\limits_{t'=t-\Delta_i}^{t-1} U_i(t'), & t \geq \Delta_i \\ \chi_i - \sum\limits_{t'=0}^{t-1} U_i(t'), & 0 < t < \Delta_i \end{cases} \tag{2}$$

Now, we can only assign license units for an appliance only if there is demand for that appliance and there exist free units in its pool. Therefore, $U_i(t) \leq min[L_i(t), D_i(t)]$. The cost C_i of a license unit for appliance α_i is calculated as sum of license prices for each software $S_j \in \alpha_i$. Now, the returns derived by the DTC each time an appliance is used in a project are directly proportional to the cost of the appliance. We formulate an optimization problem in Equation 3 that seeks to maximize such returns. The constraints are: license purchases are limited to as many units as are permitted by the available capital, F; and existence of both demand and unassigned license units.

$$\begin{aligned} max. \sum_{\forall i} C_i \sum_{t=1}^{T} U_i(t) \\ s.t. \quad \sum_{\forall i} C_i \chi_i \leq F \\ U_i(t) \leq L_i(t) \ \forall i, t \in \{1, 2, \ldots T\} \\ U_i(t) \leq D_i(t) \ \forall i, t \in \{1, 2, \ldots T\} \end{aligned} \tag{3}$$

The optimization problem contains $n(T + 1)$ variables; where n is the number of appliances and T is the upper limit of the discrete time interval that we consider. For example, the variables for α_i are: $\chi_i, U_i(1), U_i(2), \ldots, U_i(T)$. All variables take up integer values only; so the problem is NP-complete like all integer programming problems.

Pricing Appliances: Once we solve the optimization problem in Eqn. 3, we can ascertain the price for using an appliance per unit time. This is computed by amortizing the total costs spent on licenses and configuration over the period of time when instances of that appliance find use. Additionally, support charges may be bundled; if support is important.

$$\text{Price of } \alpha_i \text{ per unit time} = \frac{\text{Configuration Cost} + C_i\chi_i}{\Delta_i \sum_{t=1}^{T} U_i(t)} \tag{4}$$

3 Experiments and Results

In this section, we experimentally show that our model is tractable and also validate its usefulness. Further, we design statistical tests to derive indicators that can help a DTC vendor to choose profitable appliances to stock in the cloud. First, we discuss the data and the setup used in our experiments and then we present the emprirical studies.

3.1 Experimental Data and Setup

Our service model is designed to source its inputs from the forecasted deal pipeline and the archives of financials for recent projects. Since, such data is highly confidential we resort to data synthesis to create data-sets for the purposes of this paper. Wherever possible we try to mimic real samples coming from services enterprises which engage in application development and maintenance. Table 1 shows how the different levers in our model are simulated for our evaluation (See [5] for further details).

To solve the integer linear program described in Equation 3, we used an IP solver engine called *Gurobi* available within a commercial optimization and simulation package[3]. All experiments were run on a machine with a configuration – 2.16 GHz, 2 GB RAM, Windows XP. The entire experimental data-set as well as the solution set-up in MS-Excel is available for download[4].

3.2 Study 1: Feasibility and Effectiveness of the DTC Service Model

Goals and Method: This study seeks to demonstrate the feasibility and validity of our approach on synthesized practical data-sets. We solve the optimization problem in Equation 3 on several data-sets that are produced as per Table 1. We vary the number of appliances and the available capital to generate different data-sets for this study.

After solving the optimization problem, we obtain values for the number of license-units to be purchased, χ_i, as well as the weekly allocations, U_i, for each appliance, α_i. Now, in order to quantify the value of our approach stemming from license reuse across projects, we compute the `Overall-Potential` score defined below:

$$\texttt{Overall-Potential (\%)} = \frac{\sum_{\forall i} C_i \sum_{t=1}^{T} U_i(t) - \sum_{\forall i} C_i\chi_i}{\sum_{\forall i} C_i\chi_i} \times 100$$

[3] http://www.solver.com

[4] http://researcher.watson.ibm.com/researcher/files/in-debdomuk/dtc-data.zip

Results and Analysis: Table 2 list various details of the solutions obtained to our optimization problem for 5 data-sets with different combinations of available capital, F and number of appliances, n.

Every run of the IP solver completes within 15 seconds and ends up with a globally optimal solution. The fact that the *Left-Over capital* ($F - \sum_{\forall i} C_i \chi_i$) is always less than the lowest license-unit cost available in the data-sets indicate the no more license allocations are possible beyond what is obtained in the solutions. *Unsatisfied Demand* points to cases where no allocations where possible despite presence of demand. Non-zero values for unsatisfied demand show that there were no trivial solutions to the IP.

Table 1. Data-set Parameters

Time Period, T	54 weeks
Weekly Demand Function, $D_i(t)$	Gaussian data sample with μ, σ from [5,10], [0,5] respectively.
License-Unit Cost, C_i	Random from [$500, $3000].
Project Duration Δ_i **(weeks)**	Uniformly from {2, 4, 6, 8, 12, 16, 20, 24, 32, > 54}

Table 2. Validating Feasibility and Effectiveness

n	**20**	**50**	**100**	**100**	**100**
F	**$0.5 mn**	**$5 mn**	**$2 mn**	**$5 mn**	**$10 mn**
Variable (= $n[T+1]$)	1100	2750	5500	5500	5500
Time taken (in secs)	5.45	9.23	13.6	12.7	13.28
Total Objective	11088550	27654900	22558400	35874900	49131000
Left-Over Capital(in $)	200	50	0	250	0
Unsatisfied Demand (%)	23	16.6	66.9	46.1	24.4
Overall-Potential (%)	2117	453	1028	617.5	391

Our approach yields high values of `Overall-Potential` in all cases. This underscores the significant financial benefits that can stem from optimal license management and license resale in DTC environments. `Overall-Potential` subsides as we add availability of capital in our experiments. With less capital, our solution always allocates licenses to the most profitable appliances. When we add more capital, the solution serves relatively less profitable appliances, so the average potential decreases.

3.3 Study 2: Identifying Profitable Appliances

Goals: Optimal solutions to Equation 3 indeed help a DTC provider to decide the correct amount of stock to keep for each appliance. However, in many practical scenarios, one needs to decide whether an appliance is a *good* candidate to stock on cloud without complete information of the other appliances in contention. It is often desirable to be able to independently form an opinion on an appliance's *profitability* simply by studying its characteristics – either in isolation or with respect to high level trends observed in other appliances. This study aims to identify some characteristics of an appliance that can possibly serve as indicators of profitability. We organize the study in terms of 3 research questions; each of which tries to evaluate a parameter in our service model on whether its value can throw some light upon the preferredness of the appliance.

- **RQ1:** Investigate the effect of *project duration*, Δ_i, on an appliance's profitability.
- **RQ2:** Investigate the effect of *cost of a license-unit*, C_i, on an appliance's profitability.

– **RQ3:** Investigate the effect of *demand function*, D_i, on an appliance's profitability.

Method: We empirically evaluate the dependence of different parameters in our service model, Δ_i, C_i and D_i, on a quantitative measure of an appliance's profitability – the *Profit-Potential (PP)* of an appliance:

$$\texttt{Profit-Potential (\%)} = \frac{C_i \sum_{t=1}^{T} U_i(t) - C_i\chi_i}{C_i\chi_i} \times 100$$

In the context of any solution to our service model, the `Profit-Potential` for an appliance signifies the degree of license re-sale that can be effected for the appliance in the solution. Of course, greater the re-sale, higher is the profit. Now, for each RQ, we empirically measure the correlation of the parameter in question with `Profit-Potential`. We analyse every parameter in isolation, i.e., while studying the effect of a parameter, we generate the data such that the other parameters take identical values in all appliances being monitored. Next, we perform statistical tests to decide whether the correlation is significant at 0.01 level. If the correlation is found to be significant, then we conclude that the parameter is indicative of an appliance's profitability.

Statistical Analysis and Results: To address all 3 research questions, we create samples of data pertaining to 100 appliances. Table 3 shows the values taken up by different model parameters in the three cases and also summarizes the experimental results. For **RQ1**, we measure the correlation between the project-duration and the observed `Profit-Potential` for appliances to be -0.632, which is significant at 0.01 level. Since, the project duration demonstrates a strong negative correlation we can conclude that lower project duration means greater profitability for the DTC vendor. The result is intuitively true since lower the project duration, greater are the chances of license re-sale. For **RQ2**, we measure the correlation between the cost of a single license-unit and the observed `Profit-Potential` for appliances to be -0.17. Such a value for correlation is not significant, thus we cannot make conclusive statements about the effect of license-unit cost on appliance's profitability. For **RQ3**, we wish to determine whether uniformity in demand is good for profitability of an appliance. Thus, we measure the correlation between the standard-deviation of weekly demand for appliances across the one year period and the observed `Profit-Potential` for appliances to be -0.382, which is significant at 0.01 level. Since, lower deviation in demand leads to greater profitability, we can conclude that uniform demand for an appliance augurs well for the DTC vendor.

In summary, appliances that find use in projects having smaller durations and appliances having uniform demand functions are better prospects for stocking in a DTC.

4 Related Work

License management for cloud computing environments has been recognized as an open problem in literature. The 451 group report[3] and [4] identified restrictive license terms to be a major threat to cloud adoption. Dalheimer et. al. [2] propose GenLM, a license management framework that allows ISVs to manage their license usage in a distributed

Table 3. Identifying Profitable Appliances

	RQ1	**RQ2**	**RQ3**
Weekly Demand, $D_i(t)$	10	10	Gaussian with μ, σ from [5, 10] and [0,5] resp.
License-Unit Cost, C_i	1000	Random from [$500, $3000]	1000
Project Duration, Δ_i	{2, 4, 6, 8, 12, 16, 20, 24, 32, 54} weeks.	6 weeks	6 weeks
Available Captial, F	$10000000	$2000000	$2500000
Time Period, T	54 weeks	54 weeks	54 weeks
Study Variable, X	Δ_i	C_i	$\sigma(D_i)$
Pearson's-Correlation(X, PP)	-0.632	-0.17	-0.382
Significance at 0.01 level	Yes	No	Yes

world. The main idea of GenLM is to attach the license not to a node or a person but to issue licenses for the input datasets, thus allowing users to buy a per-job license and run the job on any suitable resource. However, such a model is only applicable for web applications and does not suit standalone software that run on desktops.

Cacciari et. al. [1] propose elasticLM - Licence as a Service (LaaS) for creating and managing software licenses. The framework enables a user to negotiate terms and policies with a license provider service to be able to procure a license token to execute an application. [8,9] discuss issues related to implementation of license management systems for Grid environments. But, none of these address how a LaaS can be run profitably.

5 Conclusions

We note that unavailability of enterprise software under usage based pricing models can potentially affect adoption of a *Development & Test Cloud (DTC)* – a delivery platform that promises revolutionary cost advantages and efficiency improvements for an IT service enterprise. As a counter, we propose a novel service model wherein the DTC acts as a proxy to help transform software costs from perpetual licensing to pay-per-use. We suggest that a DTC can centrally buy all software licenses and maintain them in license pools. All software present in an image are assigned licenses from the pools whenever a new instance is provisioned; the licenses return back to their respective pools on deprovisioning of the instance. We set up an optimization problem that can determine how many licenses for every software should a DTC buy in order to meet the forecasted demand in the most effective yet profitable manner. Our empirical studies show that it is feasible to obtain optimal solutions to the service model. Also, we find that stocking appliances that are used in projects with a small duration and the ones that exhibit uniform demand can lead to greater profitability of the DTC vendor.

References

1. Cacciari, C., D'Andria, F., Gozalo, M., Hagemeier, B., Mallmann, D., Martrat, J., Peréz, D., Rumpl, A., Ziegler, W., Zsigri, C.: Elasticlm: A Novel Approach for Software Licensing in Distributed Computing Infrastructures. In: 2nd IEEE International Conference on Cloud Computing Technology and Science, pp. 67–74 (2010)

2. Dalheimer, M., Pfreundt, F.: GenLM: License Management for Grid and Cloud Computing Environments. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 132–139 (2009)
3. IDC and Flexera. Inc. 2010 Key Trends in Software Pricing & Licensing Survey (2010)
4. MacVittie, L.: Cloud Computing's Other Achilles' Heel: Software Licensing (2009)
5. Mukherjee, D., Gupta, M., Sinha, V.S., Zhou, N.: Development & Test Cloud: A Next Generation Service Delivery Platform. IBM Technical Report No. RI11007 (2011), http://domino.research.ibm.com/library/cyberdig.nsf/index.html
6. Silver, E., Pyke, D., Peterson, R., et al.: Inventory management and production planning and scheduling, vol. 2. Wiley, New York (1998)
7. Singh, A., Hung, E., Balepin, I., et al.: IBM Technology Adoption Program Cloud Sandbox Internal Pilot (2009)
8. Dornemann, K., Freisleben, B.: Licensing the Use of Grid Services, Citeseer (2007)
9. Dong, X., et al.: Floating License Sharing System in Grid Environment. In: 1st International Conference on Semantics, Knowledge and Grid, p. 96 (2005)