Parallel Cloud Computing Billing Model For Maximizing User's Utility and Provider's Cost-Efficiency

Mark Sandstrom
ThroughPuter, Inc.
mark@throughputer.com

ABSTRACT

Presented cloud computing billing model is designed for maximizing value-adding throughput of a multi-user parallel computing platform across a set of users of the platform. The model enables maximally demand driven computing capacity allocation while guaranteeing each user application its contract-based assured minimum capacity access level whenever actually demanded. The billing model thus facilitates providing maximized application processing throughput per unit cost across the set of user applications dynamically sharing the platform. Technically, for any given user contract, on any given billing assessment period, the model determines a level of the given contract's demand for capacity that is met by the capacity allocated to it, and assess billables for the contract based on such met demand and the contract's assured access to the capacity, as well as billing rates on individual billing assessment periods for the met demand and the contract's assured access for the capacity.

Categories and Subject Descriptors

C.2.4 [**Cloud computing**]

General Terms

Algorithms, Management, Performance, Design, Economics, Experimentation, Security, Standardization, Theory.

Keywords

Billing, throughput, optimization, cost-efficiency, performance.

1. INTRODUCTION

Computing systems will increasingly be based on large arrays of processing cores. Particularly in cloud computing, the many-core processing hardware will be shared among a number of software applications, which often belong to different users, while also individual software applications will increasingly be executing on multiple processing cores in parallel. As the processing loads and types of the applications for a given computing platform will vary over time, the particular set of application program processing tasks running on the processing cores of a given parallel computing platform will need to be dynamically updated, potentially highly frequently, in order to pursue optimized application program level as well as system wide processing throughput. To cost-efficiently enable such dynamic application task switching on a parallel computing platform, novel multi-user parallel computing architectures are needed to support dynamically assigning optimal sets of processing task instances for a given pool of parallel processing cores, and efficiently connecting the processing context of any given task instance to any core of the system (based on which task instance is assigned for execution at any core at any given time) as well as to facilitate efficient communication among the tasks of any given application program instance running on the many-core processing system. Moreover, innovations are needed regarding effective pricing and billing of user contracts, to increase the parallel computing cost-efficiency both for the users and provider of the computing service. Particular challenges to be solved include providing effective compute capacity service unit pricing model and billing techniques with appropriate incentives and tools to optimally spread application processing loads in time and space across the available parallel data processing resources, in order to pursue maximization of data processing throughput per unit cost for the users as well as maximization of profitability for the service provider.

2. CONTEXT

2.1 Manycore Processor Dynamically Allocated among Multiple Parallel Programs

The cloud computing billing model to address the above outlined innovation challenge is presented in the following in the context of a parallel processing system, referred to as a cloud processor, that is dynamically shared among a number of application programs, each of which has a variable number of ready-to-execute instances.

In the assumed operating context, each of the application program instances will typically further have a variable number of ready-to-execute tasks. However, in case of a multi-stage processing system, where for any given application each task is hosted on

a different processing stage, we can limit our study of the billing methods here to just one of the processing stages. Thus in the following discussion there will be just one task type per any of the applications at the processing system instance under study (e.g. the cloud processor per Fig. 1). In a system with multiple pipelined and/or parallel processing stages, the total billables for a given application will be the sum of its billables across all the processing stages. A suitable multi-stage parallel processing architecture to extend the scope of the herein discussed techniques is provided in [1].

A block diagram for such a cloud processor is shown in Fig. 1 below.

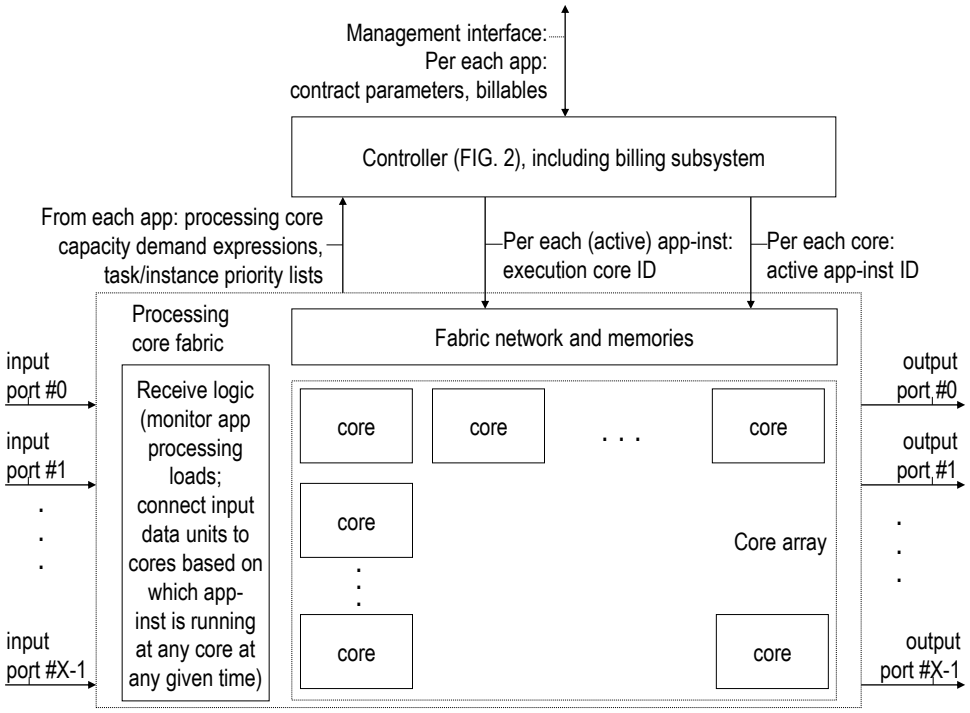

Figure 1. Cloud processor system architecture.

The dynamic management of the parallel processing resources (core slots in the core array) at a cloud processor per Fig. 1 is done by its controller, which (per Fig. 2) periodically allocates the cores among the application programs sharing the manycore processor, and assigns one of the processing core slots for each application instance selected execution on a given core allocation period. The core allocation period (CAP), e.g. 1 microsecond (can be even shorter if desired when the controller is implemented by hardware logic), is used as the elementary billing counter time tick for the billing assessment periods (BAPs) for the application programs sharing the given cloud processor.

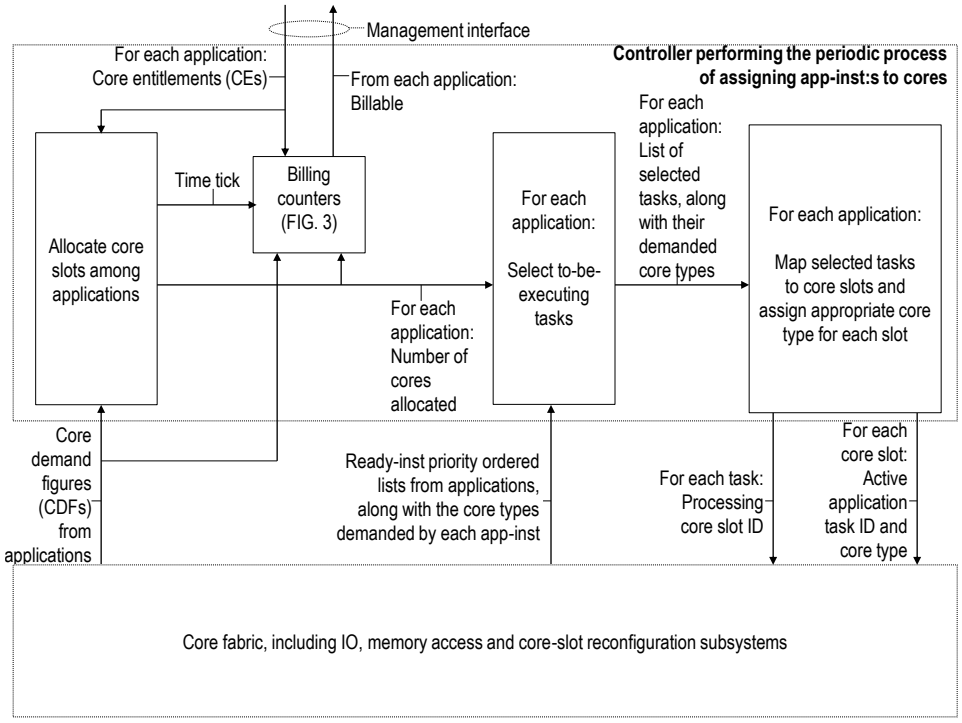

Figure 2. Cloud processor controller, including the context for the billing subsystem.

Comprehensive technical references for a cloud processor per the architecture diagrams in Figs. 1 and 2 are found in [1]-[3]. Taken together, the techniques per [1]-[3], pursue maximizing the value-adding user application processing throughput across all the applications sharing the given pool of parallel processing resources based on the realtime processing load demand variations of the individual applications, while providing for each application assured access to its contract based share of the processing capacity whenever actually demanded. Among the novel aspects of such cloud processors, of particular relevance to the billing functionality is the algorithm (see the leftmost box in the controller module of Fig. 2 for context) for periodically optimizing the allocation of the given pool of processing cores among the set of applications sharing such a pool.

2.2 Core Allocation Algorithm

Objectives for the core allocation algorithm include maximizing the processor core utilization (i.e., generally minimizing, and so long as there are ready app-insts, eliminating, core idling), while ensuring that each user application program (app) gets at least up to its entitled (e.g. a contract based minimum) share of the processor core capacity whenever it has processing load to utilize such amount of cores. Each app sharing a given manycore processor (Fig. 3) is specified its entitled quota of the cores, at least up to which number of cores it is to be allocated whenever it is able to execute on such number of cores in parallel. Naturally, the sum of the apps' core entitlements (CEs) is not to exceed the total number of core slots in the given processor. Each app on the processor gets from each run of the core allocation algorithm:

(1) at least the lesser of its (a) core entitlement (CE) and (b) core demand figure (CDF) worth of the cores; plus

(2) after condition (1) is met for all apps sharing the processor, as many additional cores to match its CDF as is possible while maintaining fairness among apps whose CDF is not fully met; plus

(3) the app's fair share of any cores remaining unallocated after conditions (1) and (2) are met for all the apps.

This algorithm allocating the cores to apps runs as follows:

(i) First, any CDFs by all apps up to their CE of the cores within the array are met. E.g., if a given app #P had its CDF worth zero cores and entitlement for four cores, it will be allocated zero cores by this step (i). As another example, if a given app #Q had its CDF worth five cores and entitlement for one core, it will be allocated one core by this stage of the algorithm. However, to ensure that each app-task will be able at least to communicate at some defined minimum frequency, the step (i) of

the algorithm allocates for each app, regardless of the CDFs, at least one core once in a specified number (e.g. sixteen) of the core allocation periods.

(ii) Following step (i), any processing cores remaining unallocated are allocated, one core per app at a time, among the apps whose CDF had not been met by the amounts of cores so far allocated to them by preceding iterations of this step (ii) within the given run of the algorithm. For instance, if after step (i) there remained eight unallocated cores and the sum of unmet portions of the app CDFs was six cores, the app #Q, based on the results of step (i) per above, will be allocated four more cores by this step (ii) to match its CDF.

(iii) Following step (ii), any processing cores still remaining unallocated are allocated among the apps evenly, one core per app at time, until all the cores of the array are allocated among the set of apps. Continuing the example case from steps (i) and (ii) above, this step (iii) will allocate the remaining two cores to certain two of the apps (one for each). Apps with zero existing allocated cores, e.g. app #P from step (i), are prioritized in allocating the remaining cores by this step (iii).

Moreover, the iterations of steps (ii) and (iii) per above are started from a revolving app ID# within the set, so that the app ID# to be served first by these iterations is incremented by one (and returning to 0 after reaching the highest app ID#) for each successive run of the algorithm.

Accordingly, all cores of the array are allocated on each run of the above algorithm according to apps' processing load variations while honoring their contractual entitlements. I.e., the allocating of the array of cores by the algorithm is done in order to minimize the greatest amount of unmet demands for cores (i.e. greatest difference between the CDF and allocated number of cores for any given app) among the set of apps, while ensuring that any given app gets its CDF at least within its CE met on each successive run of the algorithm.

The remaining elements of the cloud processor architecture per Fig. 1, including its controller process per Fig. 3, are described more comprehensively in [1]-[3]. In summary, the cloud processor hardware provides, besides the processing cores, and the dynamic core allocation per above, the functionalities of monitoring applications' processing loads and their contractual core capacity entitlements, prioritizing and selecting application task instances for execution, mapping selected task instances for processing on their assigned cores and accordingly dynamically configuring the memory and IO access subsystems (and on programmable hardware, the core slot types), and arranging the inter-task communications. While all these hardware-automated functionalities are essential to enable the dynamic parallel program execution, only the core allocation algorithm per above impacts the billing functionality which is the focus of this paper.

2.3 Billing System

2.3.1 Overview

The presented billing techniques are designed for maximizing the value-add of the application processing throughput of a multi-user parallel computing platform across a set of users of the service provided with the platform. These billing techniques, for any given user contract among the contracts supported by the platform, and on any given billing assessment period, determine a level of a demand for the capacity of the platform associated with the given contract that is met by a level of access to the capacity of the platform allocated to the given contract, and assess billables for the given contract based on 1) such met demand and 2) a level of assured access to the capacity of the platform associated with the given contract, as well as 3) billing rates, applicable for the given billing assessment period, for (a) the met demand and (b) the level of assured access associated with the given contract.

A logic block diagram billing subsystem for the cloud processor per Figs. 1 and 2 is presented in Fig. 3 below.

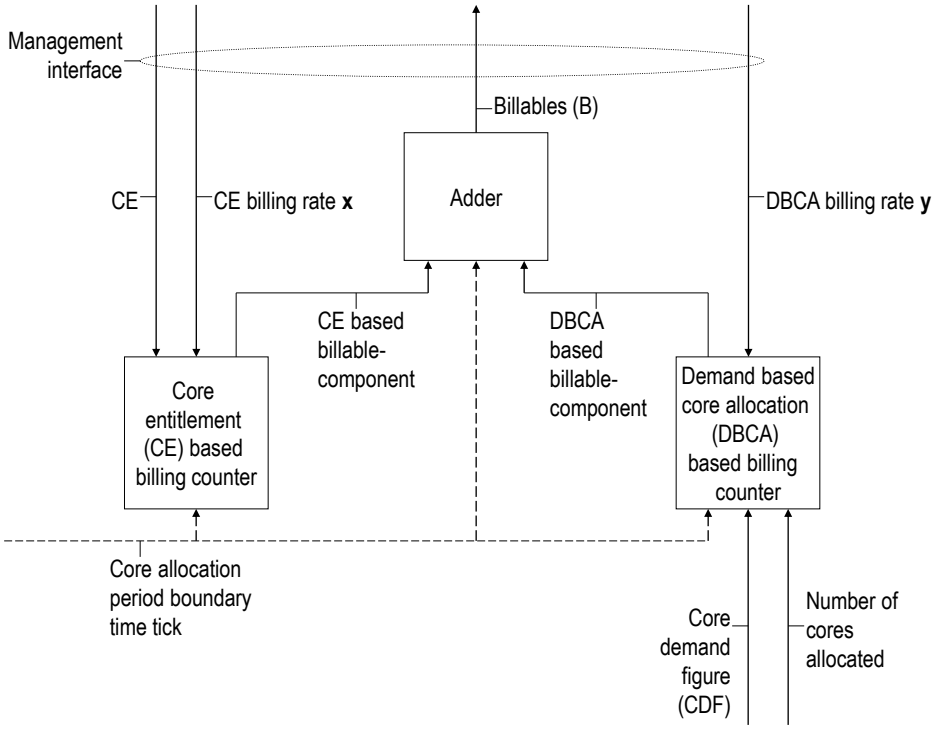

Figure 3. Billing subsystem of the cloud processor controller.

2.3.2 Objectives

The presented cloud processor billing techniques target maximizing: i) the on-time data processing throughput per unit cost for the users of a given processing system per Fig. 1, and ii) the revenue over a period of time for the service provider operating such a system of a certain total cost. Accordingly, these techniques have the following objectives:

1) Maximizing, at given billing rates for demand-based core allocations (DBCAs) for a billing assessment period (BAP), the total volume of demand-based core allocations for the programs configured for a given system per Fig. 1. Herein, DBCA refers to an amount of cores allocated to a program to meet that program's core demand figures (CDF) on the given BAP (i.e., any cores allocated for a program beyond the CDF of the program are not counted as demand based core allocations). DBCA for a given program on a given core allocation period (CAP) is taken as the lesser of the CDF and allocated core count of the program.

2) Maximizing, at given billing rates for core entitlements (CEs), the number of core entitlements sold for user contracts supported by a given system per Fig. 1. CE herein refers to the number of cores up to which amount of cores of the shared array a given user program is assured to get its (CDFs) met by core allocations on successive runs of the algorithm.

These objectives reflect the utility for the users running their programs on a system per Fig. 1; the users are assumed to perceive value in, and be willing to pay for, assured access to their desired level of capacity of a given compute system and their actual usage of the platform capacity. Accordingly, the above objectives 1) and 2) are among principal factors driving the revenue for the operator of the given system per Fig. 1.

2.3.3 Billing Formula

Per Fig. 3, the billables (B) for the operator of the system from a given user contract is per the following equation: $B = x*CE + y*DBCA$ (Equation 1), wherein CE stands for core entitlement for the user, DBCA stands for the amount of core allocations to that user's program to meet its CDFs for the Core Allocation Periods (CAPs, e.g. 1 microsecond each) during the contract time period in question, and x and y are billing rates per the contract that convert CE and DBCA into monetary figures.

An advantage of this billing method is that a portion (i.e. the term y*DBCA) of the cost of the utility computing service for a user running its program on a system per Fig. 1 is based on the CDFs of the user's program (to the degree that CDFs are met by core

allocations). Therefore, each user of the system per Fig. 1 has an economic incentive to configure its programs so that they eliminate any CDFs beyond the number of cores that the given program is actually able to utilize at the given time. If so allowed for a given user contract, the system will generate the CDFs for the user automatically based on the input data load levels for the user program instances. Whether the CDFs are generated by a user programs or the system on their behalf, the users have the incentive to not automatically (cause a) demand (for) at least their CE worth of cores irrespective of on how many cores the given program is able to execute on in parallel at any given time. This incentive leads to increasing the average amount of surplus cores for runs of the core allocation algorithm i.e. cores that can be allocated in a fully demand driven manner (rather than in a manner to just meet the CDFs by each application for their CE figure worth of cores). Such maximally demand driven core allocation (which nevertheless allows guaranteeing each user application an assured, contract defined minimum capacity access level whenever actually demanded) facilitates providing maximized value-adding processing throughput per normalized cost across the set of user applications dynamically sharing the system per Fig. 1.

Moreover, either or both of the billing rates x and y for Equation 1 can be specified in the user contract to vary over time. The term x*CE can take a form of a sum such as x1*CE1 + x2*CE2, wherein, for example, x1 is the billing rate for a core entitlement during specified premium businesses hours (e.g. Monday-Friday 9am - 5pm at the local time zone of the given platform or user) and x2 the billing rate for a core entitlement outside the premium business hours, while CE1 and CE2 are core entitlements for the given user contract for the premium and non-premium hours, respectively. Naturally, there can be more than just two time phases with their respective billing rates. For instance, in addition to premium pricing during the business hours, also evening hours 5 pm - 1 am could have a different billing rate than 1am - 9am, and so forth, depending on the popularity of the compute capacity usage during any given hours of the day. Similarly, different days of the week, special calendar days etc. can have different billing rates, based on expected popularity of compute capacity on such days. Naturally, this discussion applies also the for the coefficient y of the term y*DBCA in Equation 1.

Per Fig. 3 (see also context from Figs. 1 and 2), digital hardware logic within the controller module functions as a billing counter for the contracts supported by a given system per Fig. 1. In the logic implementation for the billing subsystem functionality discussed herein, in addition to the billing rate values, the signals x and y, provide notifications of transitions of contract time phases at which the CE and DBCA billing rates (x and y) get new values. In such a logic implementation, DBCA based billing counter counts an average number of cores allocated to a given user program over the core allocation periods (CAPs) during a given billing assessment period (BAP) (i.e. time between two successive changes of the rate y, or the maximum BAP duration configured for the system), and multiplies this average DBCA amount with a total DBCA billing rate per core applicable for that BAP. Similarly, the CE based billing counter counts the average CE level for the given program (or simply takes any constant CE level for the time phase in question) for a given BAP for which the CE billing rate remains a constant, and multiplies that average (or simply constant) CE level with a total CE billing rate applicable for that BAP. At user billing intervals, the adder accumulates the series of billable components, so produced for such BAPs of constant billing rates to form the billables for the given program. For context, the typical CAPs consist of tens to thousands of processing logic clock cycles, thus lasting for microseconds or less, while the BAPs, at boundaries of which the billing rates change, may last from minutes to hours, comprising several millions to billions of CAPs. Finally, the user contract billing periods are typically calendar months, thus typically comprising tens to hundreds BAPs.

2.3.4 Usage Scenarios

The compute capacity provider operating a platform based on system(s) per Fig. 1 can offer different types of CE time profiles for different application types. For instance, a service provider operating the platform could sell four basic contract types with differing CE time profiles per examples of contract plans A, B, C and D in Tbl. 1 below:

	Plan	A	B	C	D	
	Contract type:	**enterprise**	**entertainment**	**batch**	**always on**	**Sum of CEs = cores needed for the below contract mix**
	Number of contracts	1	3	1	2	
CEs - time profiled:	- **business hours**	8	2	0	1	**16**
	- **evening hours**	1	4	0	1	**15**
	- **night hours**	0	2	8	1	**16**
	Max during 24h:					**16**
CEs - flat:	- **any hour**	8	4	8	1	**30**
Cost-efficiency gain of time profiled CEs vs. flat CEs:						***(30-16)/16 = 87.5%***

Table 1.

As illustrated in Tbl. 1, the capability to allow configuring compute capacity contracts with differing CE time profiles, particularly contract types with non-overlapping CE peaks on a given platform per Fig. 1, can be used both for improving the computing cost-efficiency for the users of the compute service provided through the platform as well as increasing the revenues that the compute capacity service provider is able to achieve with the platform of a certain cost of ownership. Either or both of the CE and DBCA billing rates can be set for different values on the different billing assessment periods (BAPs) within day, week, month, etc., in order to optimally even out the user program's collective processing load for a given system per Fig. 1 over time, and thereby, maximize the cost efficiency for the users of the computing service provided with the given platform and/or the revenue generation rate for the service provider operating the platform. For instance, in an example scenario, the CE billing rate on business days could be $0.08 per a core for the BAP of the business hours, $0.04 for the BAP of the evening hours, and $0.01 for the BAP of night hours, while DBCA billing rate, per the average number of demand based cores allocated to a given program over the eight hours of these daily BAPs, could be $0.04 for the business, $0.02 for evening, and $0.01 for night BAPs. These daily BAP billing rates can naturally be set to any other values as well, and can have differing values on different calendar days, as well as different week days (e.g. Monday-Friday versus Saturday-Sunday) can have non-uniform BAP phasing (e.g. Saturday-Sunday could replace the business hour BAP of Monday-Friday with 'extended' evening hour BAP), etc.

With the example values of Tbl. 1 for a mix (or 'basket') of enterprise, entertainment (including news etc.), batch job (overnight block data processing), and always-on type of applications, it can be seen that the capability to configure applications for a given platform per Fig. 1 with different CE time profiles enables the service provider operating the platform to support a given set of applications, with their collective CE requirements, with a significantly reduced system processing core capacity requirement, i.e., with a lower cost base for the revenues generated by the given set of user applications. With the numerical example shown in Tbl. 1, this system core utilization efficiency gain with time-profiled contract CEs compared to flat CEs enables a reduction from 30 to 16 cores needed for the provided mix of user contracts. In turn, this compute resource utilization efficiency gain through time profiled CEs reduces the cost of revenue for the utility computing service provider by an accordant factor. Put differently, the service provider's revenue per unit cost of the service provided (driven by the number of cores needed to support a given set of contracts) is multiplied accordingly.

Note that in discussion herein regarding the example of Tbl. 1, also the flat CE reference, against which the cost-efficiency of the time profiled CE contracts are compared, is assumed to be implemented on a system

per Fig. 1 that supports the application load adaptive core allocation per Ch. 2.2 etc. dynamic parallel execution techniques per [1]-[3]. Since the described dynamic compute resource allocation with contract specified minimum system access level guarantees (to be met when so demanded) is not supported by conventional computing systems, the contracts supported with a platform per Fig. 1, i.e. contracts with the capability to burst to up to the full system core capacity while having a contract defined minimum assured level of access to the shared system capacity, provide a higher market value than conventional contract types, which provide either only a dedicated capacity share (but without a capability to dynamically, without user or platform operator involvement, burst beyond the dedicated cores) or a capability to burst (but without a contract defined minimum core count based access level that the user contract is guaranteed to get whenever needed).

Moreover, regarding Tbl. 1, please also note that CE level of 0 does not imply that such contract type would not allow the application under that contract to execute on its host system per Fig. 1 during the hours in question; instead, CE of 0 indicates that, while the application is not guaranteed to have its CDFs met for up to any specified minimum core count, it will still in practice get its demand based fair of share of the cores allocated to it after the CDFs of set of the applications up to their CE levels have been met (per Ch. 2.2). In fact, at times when there are no other user application expressing a positive CDF at a given system per Fig. 1, the application with CE of 0 will get its CDFs met all the way to the total core count of the array.

The 24 hour cycle for the CE time profiles per example of Tbl. 1 here is merely to illustrate the capability to facilitate efficient combining of applications with differing demand time profiles for compute capacity into a shared compute capacity pool. In various implementation scenarios, there can be, for instance, further variants of plans within the basic contract types (e.g. plans A through D per Tbl. 1) such that offer greater CE levels than the norm for the given base plan (e.g. plan A) at specified seasons or calendar dates of the year (either during the peak hours of the profile or throughout given 24 hour days) in exchange of lower CE levels than the norm for that base plan at other dates or seasons. Besides combining contracts with differing CE profiles within 24h cycles as illustrated in Tbl. 1 to dynamically share the same capacity pools, the system also facilitates combining the seasonally differing variants of contracts within a given plan type (i.e. variants with non-coinciding seasonal peaks in their CE profiles) in the same capacity pools for further capacity utilization efficiency gains, in addition to the 8-hour phases shown in Tbl. 1. Moreover, there can be variants of contract types within a given base plan that have finer time granularity in their CE profiles. For instance, among the contracts of type B, there can be a variant that offers greater than the standard CE level of the plan type for the night hours (e.g. 1am - 9am) at specific timeslots (e.g. for a news casts at for 15 minutes at 6am, 7am, 8am) in exchange of lower CE at other times during the night hours. The system facilitates efficiently combining these type of variants of contracts within a given base type with complementary peaks and valleys in their CE profiles also within a given (8 hour) phase of the 24h cycle. As well, this type of combining of complementary variants (either seasonally, within 24h cycles, etc.) of a given contract type can take place within the aggregate CE subpool of the contracts of the given base type. In the example shown in Tbl. 1, this type of intra contract type combining of complementary variants can thus take place e.g. among the three contracts of type B, whose aggregate CE level is, for instance, during the night hours worth 3*2 = 6 cores for each CAP. At systems per Fig. 1 with greater number of cores, there will normally be a greater number of applications of any given type sharing the systems (and a greater subpool of CEs for each contract type) than what is shown in the simple illustration example of Tbl. 1.

2.3.5 Hardware Implementation for High Resolution and Overhead Elimination

The direct hardware logic implementation of the user application billing counters per Fig. 3, including the hardware logic based subcounter for computing the CE based billables components for each given application on the successive CAPs and BAPs, enables supporting (in practical terms) infinitely fine granularity of CE time profiling for the contract types and their variants. Moreover, the capability to customize the contract and variant CE time profiles per their application specific demands for processing capacity, with the hardware logic based (down to clock cycle) fine granularity, determinism, accuracy and efficiency, enables the computing service provider operating a system per Fig. 1 to profitably sell highly competitively priced compute capacity service contracts, with the offered customizable CE time profiles accurately matching the processing capacity demands of any given application type. With

these capabilities of the system, the users with less time sensitive programs, for instance among the programs within a given base plan, have an incentive to shift their processing loads (at least in term of their core entitlements) to less busy times, to make room for CE peaks at more popular times for the applications than can afford the more pricier CEs at such times of high aggregate demand for CEs (specifically, high aggregate demand that would exist if the CE pricing adjustment techniques were not used). These system software overhead eliminating, fine granularity hardware logic based pricing adjustment, billables assessment and efficient compute platform sharing techniques per above facilitate both maximizing the users' net value of the compute service being subscribed to as well as the service provider's profitability.

3. CONCLUSIONS

The presented dynamic parallel cloud computing billing model enables combining the desired aspects of per-user dedicated and multi-user shared capacity based computing services. Each user is guaranteed its access to its contract-specified level of the processing capacity whenever actually demanded. However, the contract specified capacity entitlements are neither kept locked down to their associated programs (at times when the processing load associated with a given user program does not demand its entitlement worth of processing core capacity) nor are they any limits for maximum capacity available for their user programs (at times when the processing load of a given user program exceeds its entitlement worth of core capacity). In fact, the incentives that the billing model provides for the user programs to economize on their core capacity demand expressions (i.e. to only demand as much capacity as their current processing load demands, rather than at least their capacity entitlement worth of processing cores regardless of the actual processing load) lead to maximization of the portion of the system processing capacity available for realtime application processing load variation based capacity allocation, to match the processing capacity demand peaks of the user programs (beyond their capacity entitlement levels).

Accordingly, the presented billing techniques for parallel processing system capacity utilization and application processing performance (per normalized cost) optimization described in the foregoing provide the following fundamental advantages:

- Increased user's utility, measured as demanded-and-allocated cores per unit cost, as well as, in most cases, allocated cores per unit cost. Note that, compared to a case where the users would purely pay for their core entitlements (CEs), and as such have no direct incentive to ever demand less than their CE worth of cores, the billing method wherein a portion of the billables per a user is based on the user's demand-based-core-allocations (DBCAs) (Eq. 1; Fig. 3) during the billing assessment period, incentivizes the users to economize on their core demand figures (CDFs) (e.g. not demand their CE worth of cores unless the given user application is able to effectively utilize at the time such number of cores). In turn, this leads to there on average being more cores, per unit cost for a system per Fig. 1, to be allocated to meet CDFs above any given user's CE, when the given user's program is actually able to benefit from such bursting. Note also that cores allocated beyond the CDF of the user's application do not cost the user anything, while a users' program can gain performance benefit from receiving a greater than number of cores allocated to it than it demanded. Thus the described billing techniques (together with the dynamic parallel execution techniques per [1]-[3]) maximize the amount of utilizable parallel execution core capacity received by each given user application on systems per Fig. 1 per unit of cost of the computing service provided through such platform.
- Increased revenue generating capability for the service provider from CE based billables, per unit cost for a system per Fig. 1, through the ability to offer contract plans with mostly or fully non-overlapping CE peaks (such as in case with plans A through D per example of Tbl. 1). This enables increasing the service provider's operating cash flows generated or supported by a system per Fig. 1 of certain cost level. Also, compared to a given computing service provider's revenue level, this method reduces the provider's cost of revenue, allowing the provider to offer more competitive contract pricing, by passing on at least a portion of the savings to the customers (also referred to as users) running programs on the system per Fig. 1, thereby further increasing the customer's utility of the computing service subscribed to (in terms of compute capacity received when needed, specifically, number of cores allocated and utilized for parallel program execution) per unit

cost of the service. Consequently, this technique for optimally combining user contracts with complementary CE time profiles on a given system per Fig. 1 allows the service provider operating the system per Fig. 1 to increase the competitiveness of its compute capacity service offering among the prospective customers in terms of performance and price.

The presented pricing optimization and billing techniques, in particular when combined with dynamic parallel cloud computing techniques [1]-[3], thus are designed for maximizing the overall utility computing cost-efficiency, particularly for workflows requiring parallel execution for on-time processing throughput performance gain.

4. REFERENCES

[1] Sandstrom, M. 2013. US patent application #13959596. Program Execution Optimization for Multi-stage Manycore Processors.

[2] Sandstrom, M. 2012. US patent application #13717649. Application Load and Type Adaptive Manycore Processor Architecture.

[3] Sandstrom, M. 2014. US patent application #14318512. Concurrent Program Execution Optimization.