2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing

Virtual Machine Coscheduling: A Game Theoretic Approach

Jaspal Singh Dhillon, Suresh Purini and Sanidhya Kashyap
International Institute of Information Technology
Hyderabad, India
Email: {jaspal.iiith,suresh.purini,sanidhya.iiith}@gmail.com

***Abstract*—When multiple virtual machines (VMs) are coscheduled on the same physical machine, they may undergo a performance degradation. The performance degradation is due to the contention for shared resources like last level cache, hard disk, network bandwidth etc. This can lead to service-level agreement violations and thereby customer dissatisfaction. The classical approach to solve the coscheduling problem involves a central authority which decides a coschedule by solving a constrained optimization problem with an objective function such as average performance degradation. In this paper, we use the theory of stable matchings to provide an alternate game theoretic perspective to the coscheduling problem wherein each VM selfishly tries to minimize its performance degradation. We show that the coscheduling problem can be formulated as a Stable Roommates Problem (SRP). Since certain instances of the SRP do not have any stable matching, we reduce the problem to the Stable Marriages Problem (SMP) via an initial approximation. Gale and Shapley proved that any instance of the SMP has a stable matching and can be found in quadratic time. From a game theoretic perspective, the SMP can be thought of as a matching game which always has a Nash equilibrium. There are distributed algorithms for both the SRP and SMP problems. A VM agent in a distributed algorithm need not reveal its preference list to any other VM. This allows each VM to have a private cost function. A principal advantage of this problem formulation is that it opens up the possibility of applying the rich theory of matching markets from game theory to address various aspects of the VM coscheduling problem such as stability, coalitions and privacy both from a theoretical and practical standpoint.**

We also propose a new workload characterization technique for a combination of compute and memory intensive workloads. The proposed technique uses a sentinel program and it requires only two runs per workload for characterization. VMs can use this technique in deciding their partner preference ranks in the SRP and SMP problems. The characterization technique has also been used in proposing two new centralized VM coscheduling algorithms whose performance is close to the optimal Blossom algorithm.

***Keywords*-virtual machine coscheduling; workload characterization; game theory; stable marriages problem;**

I. Introduction

Virtualization technology provides for composing elastic virtual machines (VMs) by virtualizing physical resources like processors, main memory, storage and networking devices [1], [4]. Each independent application can run on one or more VMs which can be configured to specifically suite the application needs. Application isolation, security requirements and service-level agreements can be achieved with relative ease when compared with multiple applications running on a single big monolithic physical machine (PM). Although application isolation in terms of the software requirements is easy, performance isolation is not guaranteed automatically as virtual machines running on the same physical machine can interfere with each other. For example, if two memory intensive VMs are assigned different cores on a dual core physical machine, they contend for the same Last Level Cache (LLC) polluting each others working set. This will adversely impact the performance of the applications running on those VMs. In general, a VM can undergo performance degradation if it contends for the same resource required by another VM. Performance degradation can be measured using metrics such as response time for client-server applications and running time for compute intensive applications.

Consequently, placement of VMs on PMs is a critical task as it directly affects the extent of performance degradation of a VM. Virtual machine coscheduling problem aims at finding a map between VMs and PMs while optimizing certain performance metric. Two such notable metrics are average and maximum performance degradation over all the VMs. The number of PMs required to satisfy the service-level agreements (SLAs) of all the VMs is yet another performance metric. If we consider multiple performance metrics simultaneously, then the coscheduling problem becomes a multi-objective optimization problem and there can only be pareto-optimal solutions.

The classical approach to solve the VM coscheduling problem involves a central authority which decides the VM mapping based on certain global objective function. This is currently the case in data centers and public cloud providers such as Amazon EC2 [28] and Microsoft Azure [8]. An alternate approach is to consider a distributed coscheduling algorithm wherein each VM chooses its own partner VM on mutual consent. Each VM can have its own preference list of partners based on a private benefit function. This problem can be seen as the well-known Stable Roommates Problem (SRP) [6], [18] from game theory and combinatorics. Irving [18] proposed a $O(n^2)$ time centralized algorithm to determine a solution for an instance of SRP if there exists one. We can circumvent the instability issues in the SRP by reducing the coscheduling problem to Stable Marriages Problem (SMP) [15], [19] through an initial approximation

CFP13UCC-USB/13 $26.00 © 2013 IEEE
DOI 10.1109/UCC.2013.47

described in this paper. Gale and Shapley [13] proved that every instance of an SMP has a stable matching and gave a polynomial time algorithm to construct the same. Gale and Shapley's SMP algortithm, and Irving's SRP algorithm can be run in a distributed fashion [7] by the participating VMs wherein each of the VMs need not reveal their partner preferences. Our formulation of the VM coscheduling problem as an SRP or an SMP opens up the possibility of using the rich theory of two sided matching market design [23].

A requirement for any VM coscheduling algorithm (centralized or distributed) is the characterization of a VM workload and a model for its interference effects. In this paper we propose an efficient workload characterization technique which can be used by the VMs to arrive at preference lists in SRP and SMP formulations. Using the same characterization technique, we also propose two new centralized approaches for VM coscheduling.

II. Problem Definition

Given n VMs and m k-core PMs, the VM *coscheduling problem* is to assign each VM a core of a PM such that certain objective function is optimized. In this paper, we only consider dual core PMs. We can summarize the performance degradation of VMs, when coscheduled with one of the other VMs, by using an $n \times n$ matrix whose $(i,j)^{th}$ entry, denoted as d_{ij}, indicates the performance degradation of i^{th} VM when coscheduled with j^{th} VM. Note that the performance degradation matrix is not symmetric. Given a performance degradation matrix, we can consider one of the following two objective functions for optimization.

1) *sum-objective:* Find an optimal coschedule such that the sum of all the VM performance degradations is minimized.
2) *minmax-objective:* Find an optimal coschedule such that the maximum of all the VM degradations is minimized.

The problem of optimizing the sum-objective function [21] is equivalent to finding a minimum weight complete matching in an undirected graph with n vertices. Each vertex in the graph represents a VM and an edge (i,j) is annotated with the weight $d_{ij} + d_{ji}$. There exists a polynomial time algorithm to compute a minimum weight matching in a graph and it is called as the Blossom algorithm [11]. The weight of a matching is defined as the sum of all the edge weights in it. Scheduling constraints arising due to SLAs can be incorporated by removing the corresponding edges from the graph. Optimizing the sum-objective function when the number of cores per PM is greater than two is NP-hard though [21]. Optimizing minmax-objective function is equivalent to finding a minimal weight complete matching in an undirected graph where the cost of a matching is defined as the maximum of all the edge weights in it. Given an n vertex graph with each edge (i,j) assigned a weight $max(d_{ij}, d_{ji})$, optimizing the minmax-objective function is equivalent to finding a minimal weight complete matching in this graph. We are not sure if this graph problem has a polynomial time algorithm. It can be formulated as a 0-1 integer linear programming problem, though it does not make it necessarily NP-hard.

sum and *minmax* are global objective functions which a central authority like a data center administrator would like to optimize. Fairness to all the VMs is one of the guiding principles in defining a global objective function. In contrast, we can envisage a setting in which each of the VMs acts as a selfish agent and attempts to find a coschedule partner maximizing its personal benefit locally. In the next section, we formulate the coscheduling problem from this perspective.

III. Coscheduling from a Game Theoretic Approach

The *theory of stable matching* proposed by Gale and Shapley [13] found important applications in mechanism design for two sided matching markets such as college admissions and kidney exchange programs [25]. In this section, we show that the problem of VMs selfishly choosing their coschedule partners in a distributed fashion is equivalent to the SRP, and to the SMP via an initial approximation by a centralized authority.

A. Stable Roommates Problem

The *stable roommates problem* (SRP) can be informally described as identifying n stable roommate pairs from $2n$ college students. Each student has a preference list from the rest of the $2n-1$ students. A solution to an SRP instance is said to be unstable if there exists two students who prefer each other to their current roommates. Formally, let S be a finite set of students with even cardinality. For every student $s \in S$, let $\succ_s$ be a strict total order relation on set $S - \{s\}$. If $s_1 \succ_s s_2$, for $s_1, s_2 \in S - \{s\}$, then s strictly prefers s_1 over s_2.

Definition 1. *A bijective function $\mu : S \rightarrow S$ is said to be a matching if $\mu(\mu(s)) = s$, $\forall s \in S$.*

Definition 2. *Two students $s, s' \in S$ constitute a blocking pair (s, s') with respect to a matching μ, if and only if, $\mu(s') \succ_s \mu(s)$ and $\mu(s) \succ_{s'} \mu(s')$.*

Definition 3. *A matching $\mu : S \rightarrow S$ is said to be stable if there exists no blocking pairs with respect to the matching μ.*

Definition 4. *Every $s \in S$ has a cost function $C_s : S' \rightarrow \mathcal{R}$, where $S' = S - \{s\}$. The cost function C_s gives the cost to s when it is being matched with an $s' \in S'$.*

Definition 5. *The cost of a matching $\mu : S \rightarrow S$ is defined as*

$$\sum_{s \in S} C_s(\mu(s)).$$

Table I
AN SRP INSTANCE WITH NO STABLE MATCHING.

$a \rightarrow$	b	c	d
$b \rightarrow$	c	a	d
$c \rightarrow$	a	b	d
$d \rightarrow$	a	b	c

If we assume that each VM has a strict preference order over the rest of the VMs as coschedule partners, then any centralized or distributed algorithm for the SRP can be used to obtain a stable VM coschedule.

Definition 6. *We call the cost function associated with a VM i as canonical if $C_i(j) = d_{ij}$, where d_{ij} is the degradation of VM i when coscheduled with VM j.*

A VM can keep its cost function private and use it to construct its preference list. Each VM acts as a selfish agent trying to obtain a partner occurring early in its preference list. There exists SRP instances which does not have any stable matching. Table I depicts an SRP instance with no stable matching [13]. Irving [18] gave a $O(n^2)$ time algorithm to construct a stable matching for an SRP instance if there exists one. Brito et al. [7] gave a distributed algorithm for computing the stable matching wherein the individual agents need not reveal their preference lists to any central authority. Whenever an SRP instance does not have a stable matching, we can opt for constructing a matching with minimal number of blocking pairs. However, this is an NP-hard problem and not even approximable within an $\epsilon > 0$ factor unless P=NP [6].

1) Price of Anarchy: In game theory, *price of anarchy* quantifies the social cost due to the selfish behaviour of agents. For a given performance degradation matrix M, let P_M be the optimal value of the sum-objective function. Let P'_M be the cost of some stable matching wherein each VM uses the canonical cost function. Then the price of anarchy (PoA) is defined as

$$PoA = \frac{P'_M}{P_M}.$$

We show that the PoA is not bounded by any constant. Consider the following performance degradation matrix corresponding to four VMs; let $0 < \delta < 1$ and $\gamma > 1$.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & \gamma & \gamma \\ 1 & 0 & 1-\delta & \gamma \\ \gamma & 1-\delta & 0 & 1 \\ \gamma & \gamma & 1 & 0 \end{bmatrix} \end{array}$$

The diagonal entries does not have any significance as we do not schedule a VM with itself. The matching corresponding to the optimal sum-objective function value is $\{(a,b),(c,d)\}$ and its cost is 4. And, $\{(a,d),(b,c)\}$ is the only stable matching with an associated cost of $2\gamma + 2(1-\delta)$. We can see from the following limit that the PoA is not bounded by any constant.

$$\lim_{\gamma\to\infty} PoA = \lim_{\gamma\to\infty} \frac{2\gamma + 2(1-\delta)}{4} = \infty$$

Using the above example, we can construct an ensemble of degradation matrices such that the PoA for that ensemble is not bounded by any constant.

B. Stable Matching Problem

Stable matching problem (SMP) is similar to the SRP except that the set S is divided into two disjoint sets (men and women) of equal size. Each person accepts a match with only a person from the opposite sex. Gale and Shapley [13] proved that there exists atleast one stable matching for every SMP instance. Formally, let M and W be the set of men and women respectively with equal cardinality. Every $m \in M$ has a strict total ordering relation $\succ_m$ on W. If $w_1 \succ_m w_2$, for $w_1, w_2 \in W$, then that means m strictly prefers w_1 over w_2. Analogously, there is a strict total order relation $\succ_w$ on M for every $w \in W$.

Definition 7. *A stable matching $\mu : M \rightarrow W$ is a bijective function such that there exists no $m \in M, w \in W$ such that $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$.*

We cannot use the SMP directly to formulate the VM coscheduling problem, as any VM can partner with any other VM. However, we propose a heuristic function in Section V-B which partitions the set of available VMs into two sets M and W of equal size. After the initial partition we can apply any centralized or distributed algorithm [7] to solve the SMP instance. The initial step by a central authority to reduce the coscheduling problem into the SMP problem can be seen as an instance of *Stackelberg games* [12].

1) Price of Anarchy: We show that the PoA is not bounded by any constant even in the SMP formulation of the coscheduling problem. Consider the following performance degradation matrix for 4 VMs, such that $0 < \delta < 1$ and $\gamma > 1$.

$$\begin{array}{c} \\ a \\ c \\ b \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & c & b & d \end{array} \\ \left[\begin{array}{cc|cc} 0 & \gamma & 1 & \gamma \\ \gamma & 0 & 1-\delta & 1 \\ \hline 1 & 1-\delta & 0 & \gamma \\ \gamma & 1 & \gamma & 0 \end{array}\right] \end{array}$$

The above performance degradation matrix is exactly same as the one we considered for the SRM problem in the SectionIII-A1. So the optimal sum-objective function value is still 4. Let the VMs a and c are in the partition M, and the rest of the two VMs are in the partition W. Then the only existing stable matching is $\{(a,d),(b,c)\}$ with an associated cost of $2\gamma + 2(1-\delta)$. Similar to the SRP example, we can see that $\lim_{\gamma\to\infty} PoA = \infty$, proving that the PoA is not bounded by any constant. We can embed this performance

degradation matrix as a sub-matrix of any $n \times n$ degradation matrix to show that the PoA is not bounded by any constant as the number of VMs grow asymptotically.

2) Optimality: It is a well-known fact that in the Gale-Shapley algorithm for the SMP problem, if men are the proposers, then each of them gets the best partner when compared to any stable matching. This version of the Gale-Shapley algorithm is called as *men-optimal*. However, each of the women gets the worst partner when compared to any stable matching. So the VMs in the men partition will be favoured in the men-optimal algorithm. The central authority could do the partition in such a way that the impact of this asymmetry in the Gale-Shapley algorithm is minimized. We can define a sex-neutral version of the SMP problem and it turns out that any reasonable notion of sex-neutrality makes the problem NP-complete.

3) Strategy Proof: In the game theory terminology, Gale-Shapley algorithm is an example of a *mechanism* which gives a stable matching for a given preference list. It is possible that a VM can use an alternate preference list than its true one, to manipulate the Gale-Shapley algorithm to obtain better partner. We say that a mechanism is *strategy proof* if using the actual preference list is the dominant strategy for all the participating entities. It has been proved that there exists no *strategy proof mechanism* for the SMP problem [23]. However, the men-optimal Gale-Shapley algorithm is strategy proof for men, in the sense, they are better off using their actual preference lists.

IV. Workload Characterization

Any centralized (like Blossom) or distributed placement algorithm which requires the computation of the performance degradation matrix is infeasible in practice as it requires $\Theta(n^2)$ VM coruns, where n is the number of VMs. In this section, we propose a workload characterization technique which requires $\Theta(n)$ VM runs. In the next section, we discuss its application in the stable matching setting; and also develop two new centralized VM coscheduling algorithms.

Our characterization technique is specifically aimed at a combination of compute and memory intensive workloads. Other types of workloads such as I/O bound or network bound are not considered. A two level characterization approach is adopted. The first level involves a coarse grained workload classification. At the second level, we order a collection of workloads based on their LLC usage by observing the performance degradation they would induce on a coscheduled *sentinel program*.

A. Workload Classification

We classify a given workload into one of the following three categories.

1) *Compute Intensive (CI):* Compute intensive workloads have small working sets which completely fit into cache and hence low CPI.
2) *Memory Intensive (MI):* Memory intensive workloads have large working sets when compared to the available cache size. Due to this they have high cache misses per kilo instructions (MPKI) and thereby higher CPI.
3) *Mixed (MX):* This class consists of programs which cannot be clearly classified as either compute or memory intensive.

Table II
SPEC CPU2006 workload classification (*cpiUL*=0.7, *mpkiLL*=6.0)

Name	CPI	MPKI	CLass
416.gamess	0.42100	0.00574	CI
464.h264ref	0.47000	0.09310	CI
456.hmmer	0.51500	0.01937	CI
400.perlbench	0.53200	0.53667	CI
462.libquantum	0.91500	8.99225	MI
450.soplex	1.27300	9.54089	MI
433.milc	1.31400	17.82290	MI
458.sjeng	0.70200	0.44136	MX
434.zeusmp	0.71400	2.29561	MX
481.wrf	0.76400	1.27042	MX
435.gromacs	0.78000	0.10849	MX

Algorithm 1 depicts our approach for the classification of workloads into CI, MI or MX types. The two parameters of the algorithm, *cpiUL* and *mpkiLL*, are obtained by analyzing a diverse collection of workloads on a target architecture. This is an offline one-time process. In this paper, we used SPEC CPU2006 [16] collection of programs to determine these parameters. We first separate the SPEC CPU2006 programs with relatively high MPKI when compared with the rest and classify them as memory intensive. This can be done algorithmically by clustering programs based on the MPKI dimension alone. Among the MPKIs corresponding to the memory intensive programs, we set *mpkiLL* to the lowest value. The rest of the programs in the SPEC CPU2006 are sorted according to their CPI. The first half of the sorted list is classified as compute intensive and the second half as mixed. The parameter *cpiUL* is set to the CPI value of the last program in the first half. An alternate way to choose *cpiUL* is by clustering the programs into two groups using their CPIs. Table II shows the classification of some of the workloads from SPEC CPU2006 suite on an Intel i3-2120 processor with 4 GB of RAM. The CPI and MPKI for any workload can be easily measured using the hardware performance counters available in most modern processors.

1) Partner Affinity: For each workload type in SPEC CPU2006, we identify a cluster leader. The cluster leader for compute intensive and mixed workloads is chosen by identifying the program whose CPI is closest to the cluster mean CPI. For memory intensive programs, the cluster leader is the one whose MPKI is closest to the cluster mean MPKI. Table III shows the coschedule partner preferences for each

Algorithm 1 Workload classification algorithm. *cpiUL* is the CPI upper limit and *mpkiLL* is the MPKI lower limit. These parameters are determined by an offline process.

```
if cpi ≤ cpiUL then
    Classify the workload as CI;
else if MPKI >= mpkiLL then
    Classify the workload as MI;
else
    Classify the workload as MX;
end if
```

Table III
PARTNER AFFINITIES

Type	1	2	3
CI	MX	CI	MI
MX	MX	CI	MI
MI	CI	MX	MI

workload type. This table is constructed by observing the performance degradation matrix of cluster leaders.

B. Characterization using the Sentinel Program

We define a *sentinel program* as one whose performance degradation is directly proportional to the LLC usage of the corunning workload. Hence, the performance degradation of the sentinel program will be small when corun with a compute intensive workload as against a memory intensive workload. The sentinel program could be either a synthetic or a natural program. We propose an algorithmic technique to choose an appropriate program as sentinel from a workload collection. This is a one-time job done offline. We use the SPEC CPU2006 workloads to identify the sentinel program.

Initially, we construct a performance degradation matrix, $M_{n\times n}$, where n is the number of programs in the workload collection. The i^{th} row of the matrix M summarizes the performance degradation of the i^{th} program with respect to the rest of the programs. Let $avg_{1\times n}$ be a row vector whose i^{th} entry contains the average of the i^{th} column of M. In other words, $avg[i]$ gives the average performance degradation induced by the i^{th} program on its coschedule partners. Intuitively, we would like to obtain this information by running the i^{th} program against the sentinel. Based on this observation, we identify the row vector of the matrix M which looks very similar to the avg vector and choose the corresponding row program as the sentinel. We constructively define the similarity metric between two row vectors $A_{1\times n}$ and $B_{1\times n}$ as follows.

1) Sort the elements of the two arrays A and B. Replace each element of the array with the index of its original position in the array.
2) The distance between the two vectors A and B is defined as the hamming distance between the two transformed arrays.

Given a set of workloads, we run those workloads against the sentinel program, and sort them according to the performance degradation they cause to the sentinel program. Programs occurring earlier in the list are more compute intensive and those occurring near the end are more memory intensive. In the next section, we show how we can use this sorted workload list along with their types in coscheduling algorithms.

V. APPLICATIONS OF WORKLOAD CHARACTERIZATION

Let $W_{1\times n}$ be the sorted list of workloads obtained using the sentinel program.

A. Preference Order Selection

In the SRP, each VM has to arrive at its preference list. A VM constructs its preference list by using the following two rules.

1) A VM whose workload type is preferred against another in accordance with the Table III occurs first in the list.
2) If two VMs have the same type, their order in the workload list W is preserved.

B. VM Partitioning in the SMP

In the SMP, a central authority partitions the VMs into two equal groups. The partitioning of the VMs is done by splitting the workload list W at the middle. Intuitively, workloads in the second half are *cache-hungry* and therefore it is a good idea not to schedule them together. After the initial partition, each VM finds a stable partner from the other partition. Each VM can arrive at its preference order in the same way as that of SRP except that the VMs from the same partition are eliminated from the preference list.

C. Coscheduling as Weighted Bipartite Graph Matching

We know that the coscheduling problem can be formulated as a matching problem in a complete graph. However, it requires $n(n-1)/2$ VM coruns to annotate the edges of the graph. We can reduce the number of VM coruns to $2n + n/2(n/2-1)/2$ by reducing the complete graph to a bipartite graph by eliminating the edges between the VMs belonging to the same partition (computed as in the SMP).

D. Naive Matching: A New Coscheduling Algorithm

We propose a new centralized coscheduling algorithm called Naive Matching (NM) which uses the workload list W and the type of each VM workload. The basic version of the NM algorithm pairs the workloads from the opposite ends of W giving rise to the coschedule $\{(W_1, W_n), (W_2, W_{n-1}), \cdots\}$. Intuitively, the NM algorithm attempts to create a balance by pairing a compute intensive workload with a memory intensive workload. In the basic version of the NM algorithm, the workload types

Table IV
SPEC CPU2000 WORKLOAD CLASSIFICATION

Name	CPI	MPKI	Class
164.gzip	0.786	0.08116	MX
168.wupwise	0.446	0.63273	CI
171.swim	1.018	3.72255	MX
172.mgrid	0.591	0.75447	CI
173.applu	0.68	1.09041	CI
175.vpr	1.122	2.94754	MX
176.gcc	0.697	0.70887	CI
177.mesa	0.486	0.18136	CI
179.art	1.235	2.50638	MX
181.mcf	6.052	68.91975	MI
183.equake	0.969	7.91738	MI
186.crafty	0.641	0.02144	CI
188.ammp	0.815	1.48509	MX
197.parser	1.023	1.04268	MX
200.sixtrack	0.529	0.03258	CI
253.perlbmk	0.94	0.92288	MX
254.gap	0.647	0.52449	CI
256.bzip2	0.701	0.73803	MX
300.twolf	0.883	0.01831	MX
301.apsi	0.73	1.72548	MX

and their partner affinities are not considered. The improved NM-k algorithm considers the partner affinities and attempts to find a match for a VM by considering a window of k VMs from the list W. The algorithm starts with W_n and creates a preference order list among the VMs W_1 to W_k as in the SRP. VM W_n is matched with the first VM in the preference order list. The two matched VMs are removed from the list W and the algorithm iterates until no more VMs are left unmatched.

VI. EXPERIMENTAL SETUP AND RESULTS

In this section, we give details of our experimental setup and compare the performance of the SRP, SMP, BGM and NM-k approaches with the Blossom algorithm. All experiments are conducted on a machine with a dual core Intel i3-2120 processor with 3MB of LLC. Each VM is allocated 2 GB of RAM and is pinned to one of the cores of the processor. The test workloads which run on these VMs consist of 20 programs from SPEC CPU2000. Table IV gives their CPI, MPKI and workload type as determined by the workload classification algorithm. In order to avoid over-fitting, the sentinel program (`482.sphinx3`) is extracted from the SPEC CPU2006 benchmark suite.

The following list summarizes different implementations of the coscheduling algorithms whose performance we compare in this section.

1) *Blossom:* Applies the Blossom algorithm on the actual degradation matrix. Theoretically this should give the best average VM degradation.
2) *BGM:* Implementation based on the bipartite graph matching algorithm as described in Section V-C.
3) *NM-1:* Naive matching with a window size 1.
4) *NM-2:* Naive matching with a window size 2.

Table V
PERFORMANCE SUMMARY OF VARIOUS COSCHEDULING ALGORITHMS. APD AND MAXD CORRESPOND TO THE AVERAGE AND MAXIMUM PERFORMANCE DEGRADATIONS MEASURED IN PERCENTAGE.

Model	APD	MaxD
Blossom	14.78	50.67
BGM	15.86	50
NM-1	17.08	61.11
NM-2	16.68	61.11
NM-3	15.75	61.11
SRP-A	24.86	71.05
SRP-S	27.50	163.37
SMP-Men-A	21.83	101.28
SMP-Men-S	20.02	55.56
SMP-Women-A	22.70	61.11
SMP-Women-S	24.29	169.23

5) *NM-3:* Naive matching with a window size 3.
6) *SRP-A:* SRP approach wherein each VM arrives at its preference list using the corresponding row in the performance degradation matrix.
7) *SRP-S:* SRP approach wherein each VM arrives at its preference list using the sentinel program and the workload type.
8) *SMP-Men-A:* Men optimal SMP approach wherein preference lists are computed similar to SRP-A.
9) *SMP-Men-S:* Men optimal SMP approach wherein preference lists are computed similar to SRP-S.
10) *SMP-Women-A:* Women optimal SMP approach wherein preference lists are computed similar to SRP-A.
11) *SMP-Women-S:* Women optimal SMP approach wherein preference lists are computed similar to SRP-S.

Figures 1 and 2 summarize the performance of various coscheduling algorithms with respect to the parameters average program degradation (APD) and maximum program degradation (MaxD). Table V presents the same data in a form suitable for easy comparison between APD and MaxD. The performance of the BGM, NM-1, NM-2, NM-3 approaches matches that of the Blossom's. This indicates the effectiveness of our characterization technique using the sentinel program. SRP, SMP and their variants perform decently with respect to the parameter APD when compared to Blossom's. However, the performance is really bad with respect to the parameter MaxD. This could be attributed to the social cost of selfish behaviour of the VMs wherein fairness to all the VMs is not at all a criteria. Also, the performance with respect to the parameter APD roughly remains the same irrespective of whether the preference lists are constructed using the actual degradation matrix or the sentinel program. However, with respect to the parameter MaxD, there is a substantial performance fluctuation. This probably indicates that the parameter MaxD is very sensitive to the changes in the preference order lists of the VMs.

We performed a statistical experiment to observe the

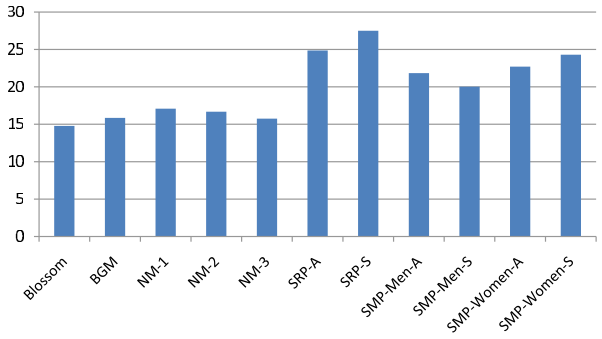

Figure 1. Average VM performance degradation (in percentage).

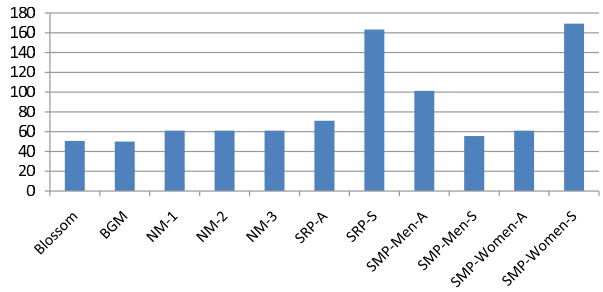

Figure 2. Maximum VM performance degradation (in percentage).

Table VI
TOP 5 SENTINEL PROGRAMS FROM SPEC CPU2006 AND THEIR FREQUENCIES

	Ratio		
Program	0.5	0.66	0.72
482.sphinx3	1808	3310	4305
403.gcc	1227	1571	1699
433.milc	760	640	442
445.gobmk	713	688	577
471.omnetpp	666	643	568
Sum of top 5	5174	6852	7591

variations in the choice of the sentinel extraction algorithm as a function of the workload collection. Towards that, we pick a random subset of the SPEC CPU2006 suite such that it consists of at least one workload from each of the CI, MX and MI types. The sentinel extraction algorithm chooses a sentinel program from the random subset. The experiment is repeated 10000 times for three different subset sizes. The fractional size of the subset in the three cases are 0.5, 0.66 and 0.72. Table VI shows the top five workloads frequently chosen as the sentinel. We can see that *482.sphinx3* is consistently being selected as the sentinel program for the majority of runs for the different ratios.

VII. RELATED WORK

Jiang et al. [20] showed that finding an optimal coschedule on a set of dual core machines is equivalent to finding a minimum weight perfect matching in a graph. They also proved that the problem becomes NP-complete if the underlying processors have more than 2 cores and proposed a set of heuristic algorithms for the same. Tian et al. [2] investigated the complexity of finding VM coschedules while minimizing the *makespan* of the schedule. The makespan of a schedule is the time taken by all the jobs to finish. To the best of our knowledge coscheduling problem has not been studied from a game theoretic perspective so far in the research literature. The closest resembling work is due to Suri et al. [26] who formulated the selfish load balancing problem as an atomic congestion game. Interestingly, they show that under certain assumptions the Nash assignment approaches the social optimum and the price of anarchy is bounded by a constant. Berenbrink et al. [5] and Adolphs et al. [3] studied the distributed version of the selfish load balancing problem.

There has been considerable amount of research on the workload classification and characterization problem. Zhang et al. [29] showed how to use hardware performance counters for workload modeling; and CPU scheduling to improve performance and fairness. Chang et al. [9] proposed a cooperative cache partitioning technique for multiplexing the cache resources to the threads running concurrently on a multi-core machine. The application of this technique in the context of multiple VMs running on a single machine is yet to be investigated. Huang et al. [17] proposed a software controlled mechanism for last level cache partitioning to reduce cache pollution. Dwyer et al. [10] proposed a machine learning based technique with a selection of hardware performance counters as features to estimate the performance degradation of a workload online. Tam et al. [27] proposed a thread clustering technique which uses performance counters to reduce cross-chip cache accesses. Snavely et al. [24] proposed a coscheduling strategy which involves an overhead-free sample phase and a symbiosis phase to determine an effective schedule on a simultaneous multithreaded (SMT) processor. Govindan et al. [14] proposed an elegant system called Cuanta in which a synthetic cache loader program is used to profile the cache usage behaviour of an application. Mars et al. [22] proposed a performance degradation prediction technique by computing a sensitivity curve for each workload by running them against a *bubble* program. The bubble program can be tuned to put varying levels of pressure on the cache.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we provide a new game theoretic perspective to the VM coscheduling problem using the theory of stable matchings. This alternate perspective opens up the possibil-

ity of using the tools and techniques from the rich theory of matching markets to address problems such as auction strategies for resources; coalitions and privacy preserving algorithms. We also proposed a new workload characterization technique using a sentinel program. The characterization technique requires only two runs of a workload, the first run by itself and the second run coscheduled with the sentinel program. The workload characterization technique helps the VMs to construct their preference lists in the stable matching setting. Using the same characterization technique we propose two new centralized coscheduling algorithms whose performance is very close to the optimal Blossom algorithm. The optimal algorithm requires the construction of a performance degradation matrix using a quadratic number of VM coruns.

REFERENCES

[1] "Kernel-based virtual machine." [Online]. Available: http://www.linux-kvm.org

[2] in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, W. Cirne, N. Desai, E. Frachtenberg, and U. Schwiegelshohn, Eds., 2013, vol. 7698.

[3] C. P.Adolphs and P.Berenbrink, "Distributed selfish load balancing with weights and speeds," in *Proceedings of the 2012 ACM symposium on Principles of distributed computing*.

[4] P.Barham *et al.*, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*.

[5] P.Berenbrink, T.Friedetzky, L. A.Goldberg, P. W.Goldberg, Z.Hu, and R.Martin, "Distributed selfish load balancing," *SIAM J. Comput.*, vol. 37, no. 4, pp. 1163–1181, Nov. 2007.

[6] P.Biró, D. F.Manlove, and E. J.McDermid, ""almost stable" matchings in the roommates problem with bounded preference lists," *Theor. Comput. Sci.*, vol. 432, pp. 10–20, May 2012.

[7] I.Brito and P.Meseguer, "Distributed stable matching problems with ties and incomplete lists," in *Proceedings of the 12th international conference on Principles and Practice of Constraint Programming*.

[8] B.Calder *et al.*, "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*.

[9] J.Chang and G. S.Sohi, "Cooperative cache partitioning for chip multiprocessors," in *Proceedings of the 21st annual international conference on Supercomputing*.

[10] T.Dwyer, A.Fedorova, S.Blagodurov, M.Roth, F.Gaud, and J.Pei, "A practical method for estimating performance degradation on multicore processors, and its application to hpc workloads," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*.

[11] J.Edmonds, "Paths, trees, and flowers," *Canadian Journal of Mathematics*, pp. 449–467, Feb.

[12] D.Fudenberg and J.Tirole, *Game Theory*. MIT Press, 1991.

[13] D.Gale and L. S.Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15.

[14] S.Govindan, J.Liu, A.Kansal, and A.Sivasubramaniam, "Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*.

[15] D.Gusfield and R. W.Irving, *The stable marriage problem: structure and algorithms*. Cambridge, MA, USA: MIT Press, 1989.

[16] J. L.Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[17] T.Huang, Q.Zhong, X.Guan, X.Wang, X.Cheng, and K.Wang, "Reducing last level cache pollution through os-level software-controlled region-based partitioning," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*.

[18] R. W.Irving, "An efficient algorithm for the stable roommates problem," *Journal of Algorithms*, vol. 6, no. 4, pp. 577 – 595, 1985.

[19] K.Iwama and S.Miyazaki, "A survey of the stable marriage problem and its variants," in *Informatics Education and Research for Knowledge-Circulating Society, 2008. ICKS 2008.*, 2008, pp. 131–136.

[20] Y.Jiang, X.Shen, J.Chen, and R.Tripathi, "Analysis and approximation of optimal co-scheduling on chip multiprocessors," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*.

[21] Y.Jiang, K.Tian, X.Shen, J.Zhang, J.Chen, and R.Tripathi, "The complexity of optimal job co-scheduling on chip multiprocessors and heuristics-based solutions," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 7, pp. 1192–1205, 2011.

[22] J.Mars, L.Tang, R.Hundt, K.Skadron, and M. L.Soffa, "Bubble-up: increasing utilization in modern warehouse scale computers via sensible co-locations," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*.

[23] A. E.Roth and M. A.Oliveira Sotomayor, *Two-sided matching : a study in game-theoretic modeling and analysis*, ser. Econometric society monographs. Cambridge, Mass.: Cambridge university press, 1992.

[24] A.Snavely and D. M.Tullsen, "Symbiotic jobscheduling for a simultaneous multithreaded processor," *SIGARCH Comput. Archit. News*, vol. 28, no. 5, pp. 234–244, Nov. 2000.

[25] T.Snmez and U.Unver, "Matching, allocation, and exchange of discrete resources," Boston College Department of Economics, Boston College Working Papers in Economics 717, 2009.

[26] S.Suri, C. D.Tóth, and Y.Zhou, "Selfish load balancing and atomic congestion games," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*.

[27] D.Tam, R.Azimi, and M.Stumm, "Thread clustering: sharing-aware scheduling on smp-cmp-smt multiprocessors," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*.

[28] G.Wang and T. S. E.Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *Proceedings of the 29th conference on Information communications*.

[29] X.Zhang, S.Dwarkadas, G.Folkmanis, and K.Shen, "Processor hardware counter statistics as a first-class system resource," in *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*.