Evolutionary Game Theoretic Power Capping for Virtual Machine Placement in Clouds

Yi Cheng-Ren
Dept. of Computer Science
University of Massachusetts
Boston, Boston, MA, USA
yiren001@cs.umb.edu

Junichi Suzuki
Dept. of Computer Science
University of Massachusetts
Boston, Boston, MA, USA
jxs@cs.umb.edu

Athanasios V. Vasilakos
Computer Science Dept.
Kuwait University
Safat 13060, Kuwait
th.vasilakos@gmail.com

Shingo Omura
OGIS International, Inc.
San Mateo, CA 94404, USA
omura@ogis-international.com

Ryuichi Hosoya
OGIS International, Inc.
San Mateo, CA 94404, USA
hosoya@ogis-international.com

ABSTRACT

This paper studies a multiobjective evolutionary game theoretic framework for application placement in clouds that support a power capping mechanism (e.g., Intel’s Runtime Average Power Limit–RAPL) for CPUs. Given the notion of power capping, power can be treated as a schedulable resource in addition to traditional resources such as CPU time share and bandwidth share. The proposed framework, called Cielo, aids cloud operators to schedule resources (e.g., power, CPU and bandwidth) to applications and place applications onto particular CPU cores in an adaptive and stable manner according to the operational conditions in a cloud, such as workload and resource availability. This paper evaluates Cielo through a theoretical analysis and simulations. It is theoretically guaranteed that Cielo allows each application to perform an evolutionarily stable deployment strategy, which is an equilibrium solution under given operational conditions. Simulation results demonstrate that Cielo allows applications to successfully leverage the notion of power capping to balance their response time performance, resource utilization and power consumption.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods; C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed applications

Keywords
Cloud computing, power-aware virtual machine placement, power capping, multiobjective optimization, evolutionary game theory

1. INTRODUCTION

Dynamic Voltage and Frequency Scaling (DVFS) is a major method of choice for investigating the tradeoff between power consumption and performance in cloud applications. Power capping is an emerging alternative to DVFS [15]. Instead of managing the CPU’s frequency directly, the user simply specifies a time window and a power consumption bound. The CPU guarantees that its average power consumption will not exceed the specified bound over each window. Both the window size and bound can be modified at runtime. This mechanism treats power as a schedulable resource and allows cloud operators to control the exact amount of power that each CPU consumes.

Given the current availability of power capping mechanisms from major CPU manufacturers, such as Intel’s Runtime Average Power Limit (RAPL), this paper focuses on an application placement problem for cloud operators to schedule resources (e.g., power, CPU and bandwidth) to applications and place applications onto particular CPU cores according to the operational conditions in a cloud, such as workload and resource availability. This paper investigates two important properties of application placement in clouds:

- **Adaptability**: Adjusting the locations of and resource allocation for applications according to operational conditions so that they can keep expected levels of performance (e.g., response time) while maintaining their resource utilization (e.g., CPU utilization and power consumption).
- **Stability**: Minimizing oscillations (non-deterministic inconsistencies) in making adaptation decisions.

Cielo is an evolutionary game theoretic framework for adaptive and stable application placement in clouds that support a power capping mechanism for CPUs. This paper describes its design and evaluates its adaptability and stability. In Cielo, each application maintains a set (or a population) of deployment strategies, each of which indicates
the location of and resource allocation for that application. Cielo theoretically guarantees that, through a series of evolutionary games between deployment strategies, the population state (i.e., the distribution of strategies) converges to an evolutionarily stable equilibrium, which is always converged to regardless of the initial state. (A dominant strategy in the evolutionarily stable population state is called an evolutionarily stable strategy.) In this state, no other strategies except an evolutionarily stable strategy can dominate the population. Given this theoretical property, Cielo aids each application to operate at equilibria by using an evolutionarily stable strategy for application deployment in a deterministic (i.e., stable) manner.

Simulation results verify this theoretical analysis; applications seek equilibria to perform evolutionarily stable strategies except an evolutionarily stable strategy can dominate the population. Given this theoretical property, Cielo aids each application to operate at equilibria by using an evolutionarily stable strategy for application deployment in a deterministic (i.e., stable) manner.

2. PROBLEM STATEMENT

This section formulates an application deployment problem where M hosts are available to operate N applications. Each application is designed with three-tiered servers (Fig. 1). Using a certain hypervisor, each server is assumed to run on a virtual machine (VM) atop a host. A host can run multiple VMs. They share resources available on their local host.

Each message is sequentially processed from a Web server to a database server through an application server. A reply message is generated by the database server and forwarded in the reverse order (Fig. 1). This paper assumes that different applications utilize different sets of servers. (Servers are not shared by different applications.) And each host runs multi cores processor to allocate different applications.

The goal of this problem is to find evolutionarily stable strategies that deploy N applications (i.e., N x 3 VMs) on M hosts so that the applications adapt their locations and resource allocation to given workload and resource availability with respect to four objectives described below. (All objectives are to be minimized.)

CPU allocation: A certain processing core time share (in percentage) is allocated to each VM. (The processing core share of 100% means that a core is fully allocated to a VM.) It represents the upper limit for the VM’s processing core utilization. This objective is computed as \( \sum_{i=1}^{M} c_i \) where \( c_i \) denotes the processing core time share allocated to the \( i \)-th tier server in an application.

Bandwidth allocation: A certain amount of bandwidth (in bits/second) is allocated to each VM. It is the upper limit for the VM’s bandwidth consumption. This objective is computed as \( \sum_{i=1}^{M} b_i \) where \( b_i \) denotes the bandwidth allocated to the \( i \)-th tier server in an application.

Response time: This objective is indicated as the time required for a message to travel from a web server to a database server: \( T^p + T^w + T^c \) where \( T^p \) denotes the total time for an application to process an incoming message from a user at three servers, \( T^w \) denotes the waiting time for a message to be processed at servers, and \( T^c \) denotes the total communication delay to transmit a message among servers. \( T^p \), \( T^w \) and \( T^c \) are estimated with the M/M/M queuing model, in which message arrivals follow a Poisson process and a server’s message processing time is exponentially distributed.

\( T^p \) is computed as follows where \( T_i^p \) denotes the time required for the \( i \)-th tier server to process a message.

\[
T^p = \sum_{i=1}^{3} T_i^p
\]  

\( T^w \) is computed as follows.

\[
T^w = \frac{1}{\lambda} \sum_{i=1}^{M} \rho_0 \frac{\rho t}{(1 - \rho t)}
\]  

where \( a_t = \lambda_t \cdot T_i^p / f_t \), \( \rho_t = a_t / (\sum_{\lambda=1}^{M} \rho_0 + 1) \).

\( O \) is the total number of processing cores that a host uses to allocate applications. \( \lambda \) is the message arrival rate for an application (i.e., the number of messages the application receives from users in the unit time). \( \lambda = \frac{1}{\sum_{i=1}^{M} \lambda_i} \) (Currently, \( \lambda = \lambda_1 = \lambda_2 = \ldots = \lambda_M \)). \( \rho_0 \) is the processing core utilization of the \( i \)-th tier server. \( f_{\text{max}} \) and \( f_i \) are the maximum CPU frequency and the CPU frequency of a host that the \( i \)-th tier server resides on.

\( T^c \) is computed as follows where \( B \) is the size of a message (in bits).

\[
T^c = \sum_{i=1}^{2} T_i^c \approx \sum_{i'=2}^{3} \frac{B \cdot \lambda_{i'}}{b_{i'}} \quad i' = t + 1
\]

Power Consumption: This objective indicates the average of power consumption (in W) that one host consumes.

\[
\lambda \sum_{\lambda=0}^{M} \rho_n \sum_{\lambda=0}^{M} \lambda_i \cdot T_i^p / f_{\text{max}} = \left( P_{\text{idle}} + P_{\text{ho}} \right) \cdot \frac{b_{\text{ho}}}{f_{\text{max}}}
\]

\( P_{\text{idle}} \) and \( P_{\text{ho}} \) denote the power consumption of a host \( h \) when its processing core \( o \) utilization is 0% and 100% at the frequency of \( f_{\text{ho}} \) respectively.

Cielo also considers four constraints:

Core capacity constraint: \( o_i \leq L_{U} \) for all \( M \) hosts. \( L_{U} \) is the maximum processing core capacity in one host, \( o_i \) is the total processing core share allocated to the \( i \)-th host. The violation of this constraint is computed as:

\[
C_{U} = \sum_{i=1}^{M} \left( I_i \cdot (o_i - L_{U}) \right)
\]

\( I_i = 1 \) if \( o_i > L_{U} \). Otherwise, \( I_i = 0 \).

Bandwidth capacity constraint: \( b_i \leq L_{B} \) for all \( M \) hosts. \( L_{B} \) is the maximum bandwidth capacity in one host, \( b_i \) is the total bandwidth allocated to the \( i \)-th host. The violation of bandwidth constraint is computed as:

\[
C_{B} = \sum_{i=1}^{M} (I_i \cdot (b_i - L_{B}))
\]

\( I_i = 1 \) if \( b_i > L_{B} \). Otherwise, \( I_i = 0 \).

Response time constraint: \( r_i \leq L_{R} \) for all \( N \) applications. \( L_{R} \) is the upper limit of response time for applications, \( r_i \) is the response time of the \( i \)-th application. The violation of response time upper limit constraint is computed as:
\[ C_R = \sum_{i=1}^{N} (I_i \cdot (r_i - L_R)) \] (7)

\[ I_i = 1 \text{ if } r_i > L_R. \text{ Otherwise, } I_i = 0. \]

Energy consumption constraint: \( e_i \leq L_E \) for all \( M \) applications. \( L_E \) is the upper limit of energy consumption for hosts, \( e_i \) is the energy consumption of \( i \)-th host. The violation of energy consumption upper limit constraint is computed as:

\[ C_E = \sum_{i=1}^{N} (I_i \cdot (e_i - L_E)) \] (8)

\[ I_i = 1 \text{ if } e_i > L_E. \text{ Otherwise, } I_i = 0. \]

Figure 1: Three Tiers of Web, Application and Database Servers

3. EVOLUTIONARY GAME THEORY

In a conventional game, the objective of a player is to choose a strategy that maximizes its payoff. In contrast, evolutionary games are played repeatedly by players randomly drawn from a population. This section overviews key elements in evolutionary games: evolutionarily stable strategies (ESS) and replicator dynamics.

3.1 Evolutionarily Stable Strategies (ESS)

Suppose all players in the initial population are programmed to play a certain (incumbent) strategy \( k \). Then, let a small population share of players, \( x \in (0,1) \), mutate and play a different (mutant) strategy \( \ell \). When a player is drawn for a game, the probabilities that its opponent plays \( k \) and \( \ell \) are \( 1-x \) and \( x \), respectively. Thus, the expected payoffs for the player to play \( k \) and \( \ell \) are denoted as \( U(k, x\ell + (1-x)k) \) and \( U(\ell, x\ell + (1-x)k) \), respectively.

Definition 1. A strategy \( k \) is said to be evolutionarily stable if, for every strategy \( \ell \neq k \), a certain \( \bar{\bar{x}} \in (0,1) \) exists, such that the inequality

\[ U(k, x\ell + (1-x)k) > U(\ell, x\ell + (1-x)k) \] (9)

holds for all \( x \in (0, \bar{\bar{x}}) \).

If the payoff function is linear, Equation 9 derives:

\[ (1-x)U(k,k) + xU(k,\ell) > (1-x)U(\ell,k) + xU(\ell,\ell) \] (10)

If \( x \) is close to zero, Equation 10 derives either

\[ U(k,k) > U(\ell,k) \text{ or } U(k,k) = U(\ell,k) \text{ and } U(\ell,\ell) > U(\ell,k) \] (11)

This indicates that a player associated with the strategy \( k \) gains a higher payoff than the ones associated with the other strategies. Therefore, no players can benefit by changing their strategies from \( k \) to the others. This means that an ESS is a solution on a Nash equilibrium. An ESS is a strategy that cannot be invaded by any alternative (mutant) strategies that have lower population shares.

3.2 Replicator Dynamics

The replicator dynamics describes how population shares associated with different strategies evolve over time [19]. Let \( \lambda_t(k) \geq 0 \) be the number of players who play the strategy \( k \in K \), where \( K \) is the set of available strategies. The total population of players is given by \( \lambda_t(t) = \sum_{k=1}^{K} \lambda_t(k) \).

Let \( x_k(t) = \lambda_t(k)/\lambda_t(t) \) be the population share of players who play \( k \) at time \( t \). The population state is defined by \( X(t) = [x_1(t), \ldots, x_k(t), \ldots, x_K(t)] \).

Given \( X \), the expected payoff of playing \( k \) is denoted by \( U(k,X) \). The population’s average payoff, which is same as the payoff of a player drawn randomly from the population, is denoted by \( U(X, X) = \sum_{k=1}^{K} x_k U(k,X) \). In the replicator dynamics, the dynamics of the population share \( x_k \) is described as follows. \( \dot{x}_k \) is the time derivative of \( x_k \).

\[ \dot{x}_k = x_k [U(k,X) - U(X,X)] \] (12)

This equation states that players increase (or decrease) their population shares when their payoffs are higher (or lower) than the population’s average payoff.

Theorem 1. If a strategy \( k \) is strictly dominated, then \( x_k(t) \rightarrow 0 \) as \( t \rightarrow \infty \).

A strategy is said to be strictly dominant if its payoff is strictly higher than any opponents. As its population share grows, it dominates the population over time. Conversely, a strategy is said to be strictly dominated if its payoff is lower than that of a strictly dominant strategy. Thus, strictly dominated strategies disappear in the population over time.

There is a close connection between Nash equilibria and the steady states in the replicator dynamics, in which the population shares do not change over time. Since no players change their strategies on Nash equilibria, every Nash equilibrium is a steady state in the replicator dynamics. As described in Section 3.1, an ESS is a solution on a Nash equilibrium. Thus, an ESS is a solution at a steady state in the replicator dynamics. In other words, an ESS is the strictly dominant strategy in the population on a steady state.

Cielo maintains a population of deployment strategies for each application. In each population, strategies are randomly drawn to play games repeatedly until the population state reaches a steady state. Then, Cielo identifies a strictly dominant strategy in the population and deploys VMs based on the strategy as an ESS.

4. CIELO

Cielo maintains \( N \) populations, \( \{P_1, P_2, \ldots, P_N\} \), for \( N \) applications and performs games among strategies in each population. A strategy \( s \) is defined to indicate the locations of and resource allocation for three VMs in an application:

\[ s(a_i) = \bigcup_{i=1}^{2,3} (h_{i,t}, c_{i,t}, u_{i,t}, b_{i,t}, p_{i,t}), 1 \leq i < N \] (13)

\( a_i \) denotes the \( i \)-th application. \( h_{i,t} \) is the ID of a host that operates \( a_i \)'s \( t \)-th tier VM. \( c_{i,t} \) is the ID of the core inside the host \( h_{i,t} \) and \( b_{i,t} \) and \( p_{i,t} \) are the CPU and bandwidth allocation for \( a_i \)'s \( t \)-th tier VM. \( p_{i,t} \) denotes the power cap of host \( h_{i,t} \) core \( c_{i,t} \) where allocates \( t \)-th tier VM. This power cap is translated later to CPU p-state based on the table 3. Each core operates at the highest p-state required by its allocated VMs.

Fig. 2 shows two example strategies for two applications \( (a_1 \) and \( a_2 \) \( (N = 2 \) and \( M = 2 \)). \( a_1 \)'s strategy \( s(a_1) \)
Algorithm 1 shows how Cielo seeks an evolutionarily stable strategy for each application through evolutionary games. In the 0-th generation, strategies are randomly generated for each population (Line 2). In each generation ($g$), a series of games are carried out on every population (Lines 4 to 24). A single game randomly chooses a pair of strategies ($s_1$ and $s_2$) and distinguishes them to the winner and the loser with respect to the objectives described in Section 2 (Lines 7 to 9). The loser disappears in the population. The winner is replicated to increase its population share and mutated with a certain rate $P_m$ (Lines 10 to 15). Mutation randomly chooses one of three VMs in the winner and alters its $h_i, t, c_i, t$ and $b_i, t$ values at random (Line 12).

Once all strategies play games in the population, Cielo identifies a feasible strategy whose population share ($s_i$) is the highest and determines it as a dominant strategy ($d_i$) (Lines 18 to 22). A strategy is said to be feasible if it never violate the CPU and bandwidth capacity constraints ($c^c = 0$ in Eq. 5 and $b^c = 0$ in Eq. 8). It is said to be infeasible if $c^c > 0$ or $b^c > 0$. Cielo deploys three VMs for an application in question based on the dominant strategy.

A game is carried out based on the superior-inferior relationship between given two strategies and their feasibility (performGame() in Algorithm 1). If a feasible strategy and an infeasible strategy participate in a game, the feasible one always wins over its opponent. If both strategies are feasible, they are compared with one of the following three schemes to select the winner.

- Pareto dominance (PD): This scheme is based on the notion of dominance [16], in which a strategy $s_1$ is said to dominate another strategy $s_2$ (denoted by $s_1 \succ s_2$) if both of the following conditions hold:
  - $s_1$’s objective values are superior than, or equal to, $s_2$’s in all objectives.
  - $s_1$’s objective values are superior than $s_2$’s in at least one objectives.

The dominating strategy wins a game over the dominated one. If two strategies are non-dominated with each other, the winner is randomly selected.

- Hypervolume (HV): This scheme is based on the hypervolume metric [24]. It measures the volume that a given strategy ($s$) dominates in the objective space:

\[
HV(s) = A \left( \bigcup \{ x' \mid s \succ x' \succ x_r \} \right)
\]  

\(A\) denotes the Lebesgue measure. $x_r$ is the reference point placed in the objective space. A higher hypervolume means that a strategy is more optimal. Given two strategies, the one with a higher hypervolume value wins a game. If both have the same hypervolume value, the winner is randomly selected.

- Hybrid of Pareto comparison and hypervolume (PD-HV): This scheme is a combination of the above two schemes. First, it performs the Pareto dominance (PD) comparison for given two strategies. If they are non-dominated, the hypervolume (HV) comparison is used to select the winner. If they still tie with the hypervolume metric, the winner is randomly selected.

If both strategies are infeasible in a game, they are compared based on their constraint violation. A strategy $s_1$ wins a game over another strategy $s_2$ if both of the following conditions hold:

- $s_1$’s constraint violation is lower than, or equal to, $s_2$’s in all constraints.
- $s_1$’s constraint violation is lower than $s_2$’s in at least one constraints.
5. STABILITY ANALYSIS

This section analyzes Cielo’s stability (i.e., reachability to at least one of Nash equilibria) by proving the state of each population converges to an evolutionarily stable equilibrium. The proof consists of three steps: (1) designing differential equations that describe the dynamics of the population state, (2) proving an strategy selection process has equilibria, and (3) proving the the equilibria are asymptotically (or evolutionarily) stable. The proof uses the following terms and variables.

- $S$ denotes the set of available strategies. $S^*$ denotes a set of strategies that appear in the population.
- $X(t) = \{x_1(t), x_2(t), \ldots, x_{|S^*|}(t)\}$ denotes a population state at time $t$ where $x_s(t)$ is the population share of a strategy $s \in S$. $\sum_{s \in S^*} x_s = 1$.
- $F_s$ is the fitness of a strategy $s$. It is a relative value determined in a game against an opponent based on the dominance relationship between them. The winner of a game earns a higher fitness than the loser.
- $p_{ik}^t = x_k \cdot \phi(F_s - F_k)$ denotes the probability that a strategy $s$ is replicated by winning a game against another strategy $k$. $\phi(F_s - F_k)$ is the probability that the fitness of $s$ is higher than that of $k$.

The dynamics of the population share of $s$ is described as:

$$\dot{x}_s = \sum_{k \in S^*, k \neq s} \{x_k p_{ik}^t - x_k p_{sk}^t\} = x_k \sum_{k \in S^*, k \neq s} x_k \{\phi(F_s - F_k) - \phi(F_k - F_s)\} \quad (15)$$

Note that if $s$ is strictly dominated, $x_s(t) \rightarrow 0$.

**Theorem 2.** The state of a population converges to an equilibrium.

**Proof.** It is true that different strategies have different fitness values. In other words, only one strategy has the highest fitness among others. Given Theorem 1, assuming that $F_1 > F_2 > \cdots > F_{|S^*|}$, the population state converges to an equilibrium: $X(t) \rightarrow \infty = \{x_1(t), x_2(t), \ldots, x_{|S^*|}(t)\} \rightarrow 1, 0, \ldots, 0 \}$.

**Theorem 3.** The equilibrium found in Theorem 2 is asymptotically stable.

**Proof.** At the equilibrium $X = \{1, 0, \cdots, 0\}$, a set of differential equations can be downsized by substituting $x_1 = 1 - x_2 - \cdots - x_{|S^*|}$:

$$\dot{z}_s = c_{sk} + \sum_{i=2, i \neq s}^{[S^*]} z_i \cdot c_{si}, \quad s, k = 2, \ldots, |S^*| \quad (16)$$

where $c_{sk} \equiv \phi(F_s - F_k) - \phi(F_k - F_s)$ and $Z(t) = \{z_2(t), z_3(t), \ldots, z_{|S^*|}(t)\}$ denotes the corresponding downsized population state. Given Theorem 1, $Z \rightarrow \infty = Z_{eq} = \{0, 0, \cdots, 0\}$ of $(|S^*| - 1)$-dimension.

If all Eigenvalues of Jacobian matrix of $Z(t)$ has negative real parts, $Z_{eq}$ is asymptotically stable. The Jacobian matrix $J$’s elements are

$$J_{sk} = \left[\frac{\partial z_s}{\partial z_k}\right]_{Z = Z_{eq}} = \left[\frac{\partial z_s}{\partial z_k}\right]_{Z = Z_{eq}} = \frac{\partial z_s}{\partial z_k}\left[c_{sk}(1 - z_s) + \sum_{i=2, i \neq s}^{[S^*]} z_i \cdot c_{si}\right] \cdot (17)$$

for $s, k = 2, \ldots, |S^*|$

Therefore, $J$ is given as follows, where $c_{21}, c_{31}, \cdots, c_{|S^*|1}$ are $J$’s Eigenvalues.

$$J = \begin{bmatrix} c_{21} & 0 & \cdots & 0 \\ 0 & c_{31} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{|S^*|1} \end{bmatrix}$$

$c_{sk} = -\phi(F_1 - F_s) < 0$ for all $s$; therefore, $Z_{eq} = \{0, 0, \cdots, 0\}$ is asymptotically stable. □

6. SIMULATION EVALUATION

This section evaluates Cielo through simulations. This paper uses a simulated cloud data center that consists of 100 hosts in a $10 \times 10$ grid topology ($M = 100$). The grid topology is chosen based on recent findings on efficient topology configurations in clouds [8,9]. This paper also assumes five types of applications. Table 1 shows the message arrival rate (the number of incoming messages per second) and message processing time (second) for each of the five application types. This configuration follows Zipf’s law. This paper simulates 40 application instances for each type (200 application instances in total; $N = 200$).

<table>
<thead>
<tr>
<th>Application type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message arrival rate ($\lambda$)</td>
<td>110</td>
<td>70</td>
<td>40</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Web server ($T^w_1$)</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>App server ($T^a_1$)</td>
<td>0.03</td>
<td>0.08</td>
<td>0.04</td>
<td>0.13</td>
<td>0.11</td>
</tr>
<tr>
<td>DB server ($T^d_1$)</td>
<td>0.05</td>
<td>0.05</td>
<td>0.12</td>
<td>0.08</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Figs. 3 to 5 illustrate how Cielo three variants evolve deployment strategies through generations and improve their objective values.

Figs. 3a, 4a and 5a show that CPU allocation decreases through generations. Cielo HV reaches 8.04% of average in
Table 3: P-states in Intel Core2 Quad Q6700

<table>
<thead>
<tr>
<th>p-state</th>
<th>CPU frequency (f)</th>
<th>P_{idle}</th>
<th>P_{max}</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>1.6 GHz</td>
<td>82.7 W</td>
<td>88.77 W</td>
</tr>
<tr>
<td>p2</td>
<td>1.867 GHz</td>
<td>82.85 W</td>
<td>92 W</td>
</tr>
<tr>
<td>p3</td>
<td>2.113 GHz</td>
<td>82.91 W</td>
<td>95.5 W</td>
</tr>
<tr>
<td>p4</td>
<td>2.4 GHz</td>
<td>83.1 W</td>
<td>99.45 W</td>
</tr>
<tr>
<td>p5</td>
<td>2.67 GHz</td>
<td>83.25 W</td>
<td>103 W</td>
</tr>
</tbody>
</table>

the last generation, which is the best performance among all three Cielo variants.

Figs. 3b, 4b and 5b show that BW allocation improves over generations. Cielo HV reaches 200.95 bps of average in the last generation, which is the best performance among all three Cielo variants.

Figs. 3c, 4c and 5c show that Cielo successfully saves energy consumption through generations. Cielo HV reaches 334.89 Watts of average in the last generation, which is the best performance among all three Cielo variants.

In Figs. 3d, 4d and 5d response time maintains almost stable through generations, because response time conflicts with all other objectives and Cielo trends to balance the trade-off among all the objectives. Cielo HV also reaches the best performance among all three Cielo variants with 26.11 ms of average in the last generation.

Table 5 compares Cielo three variants with two well-known heuristics algorithms, FFA (first-fit algorithm) and BFA (best-fit algorithm), which have been widely used for VM placement in clouds [1,6,13,14]. The table shows the minimum, average and maximum objective values in the last generation. In all objectives, Cielo HV outperforms Cielo PD and Cielo HV-PD. The largest difference is in the minimum bandwidth allocation with DVFS disabled (40%), and the smallest difference is in the maximum response time with DVFS enabled (16.60%). FFA yields the lowest power consumption because it is designed to deploy VMs on the minimum number of hosts, however it sacrifices the other objectives. Theoretically BFA should perform the best in CPU allocation because it is designed to deploy VMs on the hosts that maintain higher resource availability. However Cielo HV is able to find the dominant strategy which distributes CPU allocation among hosts even better than BFA. Cielo HV maintains balanced objective values in between FFA and BFA while Cielo yields the best performance in response time, CPU allocation and bandwidth allocation.

Table 4 shows the time required for Cielo three variants to execute given numbers of generations. Simulations were carried out with a Java VM 1.7 on a Windows 8.1 PC with a 3.6 GHz AMD A6-5400K APU and 6 GB memory space. For running a single simulation (i.e., 500 generations), Cielo HV runs 6 min 15 sec which is the fastest among all Cielo variant.

Fig. 6 illustrates how Cielo three variants evolve their hypervolume value through generations. Hypervolume value is the average computed using each application’s dominant strategy in each generation. The results confirms again Cielo HV outperforms its hypervolume performance among other Cielo variants.

From simulation results, Cielo HV outperforms over other two Cielo variants in all objectives performance value and execution time. Cielo PD and Cielo HV-PD use the notion of pareto dominance, which requires to make multi comparison among all objectives. Cielo HV instead uses just one comparison to decide the winner. Pareto dominance asks for the strictly dominant strategy, one strategy should outperforms in all objectives and survives through generations in order to become the dominant strategy. However in most of the cases strategies are tie using pareto dominance because objectives are conflicting with each other.

Table 5: Performance of Cielo, FFA and BFA

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU allocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(%/host)</td>
<td>Cielo HV</td>
<td>7.97</td>
<td>8.04</td>
</tr>
<tr>
<td></td>
<td>Cielo PD</td>
<td>19.38</td>
<td>19.69</td>
</tr>
<tr>
<td></td>
<td>Cielo PD-HV</td>
<td>19.16</td>
<td>19.51</td>
</tr>
<tr>
<td></td>
<td>FFA</td>
<td>96.1</td>
<td>96.1</td>
</tr>
<tr>
<td></td>
<td>BFA</td>
<td>10.64</td>
<td>10.54</td>
</tr>
<tr>
<td>Bandwidth allocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(bps/host)</td>
<td>Cielo HV</td>
<td>199.86</td>
<td>200.95</td>
</tr>
<tr>
<td></td>
<td>Cielo PD</td>
<td>224.54</td>
<td>228.59</td>
</tr>
<tr>
<td></td>
<td>Cielo PD-HV</td>
<td>223.62</td>
<td>225.3</td>
</tr>
<tr>
<td></td>
<td>FFA</td>
<td>445</td>
<td>446</td>
</tr>
<tr>
<td></td>
<td>BFA</td>
<td>425</td>
<td>425</td>
</tr>
<tr>
<td>Power consumption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(W)</td>
<td>Cielo HV</td>
<td>334.76</td>
<td>334.89</td>
</tr>
<tr>
<td></td>
<td>Cielo PD</td>
<td>338.99</td>
<td>339.22</td>
</tr>
<tr>
<td></td>
<td>Cielo PD-HV</td>
<td>339.08</td>
<td>339.19</td>
</tr>
<tr>
<td></td>
<td>FFA</td>
<td>43.82</td>
<td>43.83</td>
</tr>
<tr>
<td></td>
<td>BFA</td>
<td>338.66</td>
<td>338.73</td>
</tr>
<tr>
<td>Response time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(msec)</td>
<td>Cielo HV</td>
<td>25.35</td>
<td>26.11</td>
</tr>
<tr>
<td></td>
<td>Cielo PD</td>
<td>36.84</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Cielo PD-HV</td>
<td>30.64</td>
<td>30.93</td>
</tr>
<tr>
<td></td>
<td>FFA</td>
<td>173.45</td>
<td>173.45</td>
</tr>
<tr>
<td></td>
<td>BFA</td>
<td>192.92</td>
<td>192.92</td>
</tr>
</tbody>
</table>

7. RELATED WORK

Numerous research efforts have been made to study heuristics algorithms for application placement problems in clouds (e.g., [1,3,6,12–14,20,22]). Most of them assume single-tier application architecture and considers a single objective. For example, in [3,12,20,22], only energy saving is considered as the objective. In contrast, Cielo assumes a multi-tier application architecture and considers multiple objectives. It
is intended to reveal the trade-off relationships among conflicting objectives.

Game theoretic algorithms have been used for a few aspects of cloud computing; e.g., application placement [4,11,23], task allocation [17] and data replication [10]. In [4,11,23], greedy algorithms seek equilibria in application placement problems. This means they do not attain the stability to reach equilibria as Cielo does.

Several genetic algorithms (e.g., [18,21]) and other stochastic optimization algorithms (e.g., [2,5]) have been studied to solve application placement problems in clouds. They seek the optimal placement solutions; however, they do not consider stability. In contrast, Cielo aids applications to seek evolutionarily stable solutions and stay at equilibria.

8. CONCLUSIONS

This paper describes and evaluates Cielo, a multiobjective evolutionary game theoretic framework for adaptive and stable application placement in clouds that support a power capping mechanism for CPUs. It aids cloud operators to schedule resources to applications and place applications onto particular CPU cores according to the operational conditions in a cloud. It is theoretically guaranteed that Cielo allows each application to perform an evolutionarily stable deployment strategy, which is an equilibrium solution under given operational conditions. Simulation results verify that Cielo performs application deployment in an adaptive and stable manner. Cielo outperforms existing well-known heuristics: FFA or BFA.

9. REFERENCES


