1. INTRODUCTION

It is a challenging issue for cloud operators to deploy applications so that the applications can keep expected levels of performance (e.g., response time) while maintaining their utilization of resources (e.g., CPUs and bandwidth) and power consumption. In order to ensure these requirements, they are required to dynamically (re-)deploy applications by adjusting their locations and resource allocation according to various operational conditions such as workload and resource availability. This paper investigates two important properties of application deployment in clouds:

- **Adaptability**: Adjusting the locations of and resource allocation for applications according to operational conditions with respect to given objectives.
- **Stability**: Minimizing oscillations (non-deterministic inconsistencies) in making adaptation decisions.

Cielo is an evolutionary game theoretic framework for adaptive and stable application deployment in clouds that support dynamic voltage and frequency scaling (DVFS) 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 deployment strategies and adapt their locations and resource allocations to given operational conditions. Cielo allows applications to successfully leverage DVFS to balance their response
time performance, resource utilization and power consumption. In comparison to existing heuristics, Cielo outperforms a well-known multiobjective genetic algorithm, NSGA-II [4], while maintaining 74% less computational cost. It also exhibits 29% higher stability (lower oscillations) among different simulation runs than NSGA-II. Moreover, Cielo outperforms first-fit and best-fit algorithms (FFA and BFA), which have been widely used for adaptive cloud application deployment [1, 7, 15, 16].

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.)

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

- **CPU allocation:** A certain CPU time share (in percentage) is allocated to each VM. The CPU share of 100% means that a CPU is fully allocated to a VM. It represents the upper limit for the VM’s CPU utilization. This objective is computed as follows where $\lambda_i$ denotes the CPU utilization of the $i$-th tier server resides on when its CPU utilization is $\overline{\lambda_i}$, $\lambda_i$ is the upper limit for the VM’s CPU utilization. This objective is computed as follows where $\lambda_i$ denotes the CPU 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 $\sum_{t=1}^{3} b_t$ where $b_t$ denotes the bandwidth allocated to the $t$-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 $M/M/1$ 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 $T^p_l$ denotes the time required for the $t$-th tier server to process a message.

$$ T^p = \sum_{t=1}^{3} T^p_l $$

- $T^w$ is computed as follows.

$$ T^w = \frac{1}{\lambda} \sum_{t=1}^{3} \frac{\rho_t^2}{1 - \rho_t} \text{ where } \lambda = \frac{\lambda_1}{\rho_t} \frac{T_f^{\max}}{c_t} $$

- $\lambda$ is the message arrival rate for an application (i.e., the number of messages the application receives from users in the unit time). $\lambda = \frac{1}{\lambda_1}$, $\lambda_1$ is the CPU utilization of the $t$-th tier server.

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 overview 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 $\bar{x} \in (0, 1)$ exists, such that the inequality

$$ U(k, x \ell + (1-x)k) > U(\ell, x \ell + (1-x)k)$$

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

Figure 1: Three Tiers of Web, Application and Database Servers
If the payoff function is linear, Equation 7 derives:

$$\lambda \alpha_i(t) = \sum_{k=1}^{K} x_k(t) \lambda_k(t)$$

(8)

If $x$ is close to zero, Equation 8 derives either $U(k, k) > U(\ell, k)$ or $U(k, k) = U(\ell, k)$ and $U(\ell, k) > U(\ell, \ell)$

(9)

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 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 share.

3.2 Replicator Dynamics

The replicator dynamics describes how population shares associated with different strategies evolve over time [20]. Let $X(t) = \sum_{1}^{K} x_{k}(t)$ be the population share of player who play at $k$ at time $t$. The total population of players is given by $X(t) = \sum_{1}^{K} x_{k}(t)$. Let $x_{k}(t) = \lambda_{k}(t) / \lambda(t)$ be the population share of players who play at $k$ 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 $U_{k}(X, k)$. The population's average payoff, which is same as the payoff of a player drawn randomly from the population, is denoted by $U(X, X) = \sum_{k=1}^{K} x_{k}(t) \cdot U(k, X)$. In the replicator dynamics, the dynamics of the population share $x_{k}$ is described as follows.

$$\dot{x}_{k} = x_{k} \cdot [U(k, X) - U(X, X)]$$

(10)

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 $x_{k}(t) \rightarrow 0$.

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 closed connection between Nash equilibria and the steady state in the replicator dynamics, 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 location of and resource allocation for three VMs in an application:

$$s(a_i) = \bigcup_{i \in \{1,2,3\}} (h_{i,1}, c_{i,t}, b_{i,t}, f_{i,t})$$

(11)

$\alpha_i$ denotes the $i$-th application. $h_{i,t}$ is the ID of a host that operates $a_i$'s $t$-th tier VM. $c_{i,t}$ and $b_{i,t}$ are the CPU and bandwidth allocation for $a_i$'s $t$-th tier VM. $f_{i,t}$ denotes the CPU frequency of host $h_{i,t}$. Fig. 2 shows two example strategies for two applications $(a_1$ and $a_2)$ $(N = 2$ and $M = 3)$. $a_1$'s strategy $s(a_1)$ places the first-tier VM on host 1 ($h_{1,1} = 1$), which operates at 1 GHz CPU frequency and consumes 30% CPU share and 85 Kbps bandwidth for the VM $(c_{1,1} = 30$ and $b_{1,1} = 85)$. The second-tier VM is placed on host 2 ($h_{1,2} = 2$), which consumes 30% CPU share and 85 Kbps bandwidth for the VM $(c_{1,2} = 30$ and $b_{1,2} = 85)$. The third-tier VM is placed on host 2 ($h_{1,3} = 2$), which consumes 30% CPU share and 85 Kbps bandwidth for the VM $(c_{1,3} = 45$ and $b_{1,3} = 120)$. Given $s(a_1)$, $a_1$'s objective values for CPU allocation and bandwidth allocation are 105% (30 + 30 + 45) and 285 kbps (80 + 85 + 120).

**Algorithm 1** Evolutionary Process in Cielo

$$\begin{align*}
  1: & \quad g = 0 \\
  2: & \quad \text{Randomly generate the initial } N \text{ populations for } N \text{ applications: } P = \{P_1, P_2, \ldots, P_N\} \\
  3: & \quad \text{while } g < G_{\text{max}} \text{ do} \\
  4: & \quad \quad \text{for each population } P_i \text{ randomly selected from } P \text{ do} \\
  5: & \quad \quad \quad P_i^\prime \leftarrow \emptyset \\
  6: & \quad \quad \quad \text{for } j = 1 \text{ to } |P_i|/2 \text{ do} \\
  7: & \quad \quad \quad \quad s_1 \leftarrow \text{randomSelect}(P_i) \\
  8: & \quad \quad \quad \quad s_2 \leftarrow \text{randomSelect}(P_i) \\
  9: & \quad \quad \quad \quad \text{winner } \leftarrow \text{performGame}(s_1, s_2) \\
 10: & \quad \quad \quad \quad \text{replica } \leftarrow \text{replicate(winner)} \\
11: & \quad \quad \quad \quad \text{if random()} \leq P_{\text{mut}} \text{ then} \\
12: & \quad \quad \quad \quad \quad \text{replica } \leftarrow \text{mutate(winner)} \\
13: & \quad \quad \quad \text{end if} \\
14: & \quad \quad \quad P_i^\prime \leftarrow \{s_1, s_2\} \\
15: & \quad \quad \quad P_i \leftarrow P_i^\prime \cup \{\text{winner}, \text{replica}\} \\
16: & \quad \quad \text{end for} \\
17: & \quad \quad P_i \leftarrow P_i^\prime \\
18: & \quad \quad d_i \leftarrow \text{argmax}_{x \in P_i} x \\
19: & \quad \quad \text{while } d_i \text{ is infeasible do} \\
20: & \quad \quad \quad P_i \\setminus \{d_i\} \\
21: & \quad \quad \quad d_i \leftarrow \text{argmax}_{x \in P_i} x \\
22: & \quad \quad \text{end while} \\
23: & \quad \quad \text{Deploy VMs for the current application based on } d_i. \\
24: & \quad \quad \text{end for} \\
25: & \quad \quad g = g + 1 \\
26: & \quad \text{end while}
\end{align*}$$
The winner is replicated to increase its population share and mutually (Lines 7 to 9). The loser disappears in the population. The loser with respect to the objectives described in Section 2 (Lines 18 to 22). A strategy is said to be feasible if it never violate the CPU and bandwidth capacity constraints ($c^v = 0$ in Eq. 5 and $b^v = 0$ in Eq. 6). It is said to be infeasible if $c^v > 0$ or $b^v > 0$. Cielo deploys three VMs for an application in question based on the dominant strategy.

In performGame() (Algorithm 1), the selection of a winner depends on the dominance relationship between given two strategies and their feasibility. A strategy $s_1$ is said to dominate another strategy $s_2$ (denoted by $s_1 > s_2$) if:

- $s_1$’s objective values are superior than, or equal to, $s_2$’s in all objectives, and
- $s_1$’s objective values are superior than $s_2$’s in at least one objectives.

A strategy $s_1$ is said to win over another strategy $s_2$ if:

- $s_1$ is feasible and $s_2$ is infeasible.
- Both $s_1$ and $s_2$ are feasible, and $s_1 > s_2$.
- Both $s_1$ and $s_2$ are infeasible, and $s_1$’s constraint violations is lower than $s_2$’s ($c^v_1 + b^v_1 < c^v_2 + b^v_2$).

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 equilibrium, 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 $x_i(t)$ is the population share of a strategy $s \in S$. \( \sum_{s \in S^*} x_i(s) = 1 \).
- $F_s$ 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.
- $p_k^s = x_k \cdot (F_s - F_k)$ 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:

$$
\dot{x}_s = \sum_{k \in S^*, k \neq s} \{x_k p_k^s - x_s p_s^k\}
= x_s \sum_{k \in S^*, k \neq s} x_k (\phi(F_s - F_k) - \phi(F_k - F_s))
$$

(12)

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

THEOREM 2. The state of a population converges to an equilibrium.

Proof. It is that 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: $X(t)_{t \rightarrow \infty} = \{x_1(t), x_2(t), \ldots, x_{|S^*|}(t)\}_{t \rightarrow \infty} = \{1, 0, \ldots, 0\}$.

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

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

$$
\dot{z}_s = z_s [c_{s1}(1 - z_s) + \sum_{i=2,i \neq s} z_i \cdot c_{si}], \ s, k = 2, \ldots, |S^*|
$$

(13)

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, $Z_{t \rightarrow \infty} = Z_{eq} = \{0, 0, \ldots, 0\}$ 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

$$
J_{sk} = \left[ \frac{\partial z_s}{\partial z_k} \right]_{Z = Z_{eq}}
= \left[ \frac{\partial z_s [c_{s1}(1 - z_s) + \sum_{i=2,i \neq s} z_i \cdot c_{si}]}{\partial z_k} \right]_{Z = Z_{eq}}
$$

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

(14)

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

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

(15)

$c_{k1} = -\phi(F_1 - F_k) < 0$ for all $s$; therefore, $Z_{eq} = \{0, 0, \ldots, 0\}$ is asymptotically stable.

6. SIMULATION EVALUATION

This section evaluates Cielo’s adaptability and stability 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 ($T_1$)</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 ($T_2$)</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 ($T_3$)</td>
<td>0.05</td>
<td>0.05</td>
<td>0.12</td>
<td>0.08</td>
<td>0.11</td>
</tr>
</tbody>
</table>

This paper assumes each host is equipped with an AMD Opteron 2218 CPU, which has six frequency and voltage
operating points (P-states). Table 2 shows the power consumption at each P-state under the 0% and 100% CPU utilization [10, 17]. This setting is used in Eq. 4 to compute power consumption objective values.

In Cielo, the number of strategies is 100 in each population. Mutation rate ($P_m$ in Algorithm 1) is set to 0.01. The maximum number of generations ($G_{\text{max}}$ in Algorithm 1) is set to 400. Every simulation result is the average with 20 independent simulation runs.

### Table 2: P-states in AMD Opteron 2218

<table>
<thead>
<tr>
<th>CPU frequency (f)</th>
<th>$P_{\text{idle}}$</th>
<th>$P_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 GHz</td>
<td>34 W</td>
<td>68 W</td>
</tr>
<tr>
<td>1.8 GHz</td>
<td>51 W</td>
<td>80 W</td>
</tr>
<tr>
<td>2.0 GHz</td>
<td>55 W</td>
<td>84 W</td>
</tr>
<tr>
<td>2.2 GHz</td>
<td>66 W</td>
<td>89 W</td>
</tr>
<tr>
<td>2.4 GHz</td>
<td>90 W</td>
<td>97 W</td>
</tr>
<tr>
<td>2.6 GHz</td>
<td>96 W</td>
<td>108 W</td>
</tr>
</tbody>
</table>

Figs. 3 to 9 illustrate how Cielo evolves deployment strategies through generations and improves their objective values with respect to a given objective(s). Figs. 3 to 6 show the changes of objective values over generations when one of four objectives is considered. For example, Cielo considers the CPU allocation objective in Fig. 3.

In Fig. 3, CPU allocation decreases through generations because it is considered as the objective. Its average reaches 17.6% in the last generation, which is the best performance among Figs. 3a, 4a, 5a and 6a. The other objective values do not improve because they are not considered.

In Fig. 4, bandwidth allocation improves over time because it is considered as the objective. Its average reaches 185 Kbps in the last generation. This is the best performance among Figs. 3c, 4c, 5c and 6c. The improvement in bandwidth allocation contributes to the increase of response time because these two objectives conflict with each other.

In Fig. 5, response time improves over time because it is considered as the objective. Its average reaches 440 milliseconds in the last generation, which is the best result among Figs. 3b, 4b, 5b and 6b. As response time decreases, bandwidth allocation and power consumption increases because they are conflicting with each other.

In Fig. 6, power consumption decreases over time because it is considered as the objective. Its average reaches 150 W in the last generation, which is the best performance among Figs. 3d, 4d, 5d and 6d. For reducing power consumption, Cielo attempts to collocate as many VMs as possible on some hosts and turn off the other hosts that operate no VMs. It also attempts to run hosts at lower CPU frequencies. Fig. 10 confirms this analysis. It shows the number of hosts at each P-state in the last generation. With power consumption considered, 32 hosts are turned-off (indicated as 0 GHz), and 33 hosts run at the lowest P-state (1 GHz). Only four hosts run at the highest P-state (2.6 GHz). In Figs. 6 and 10 demonstrate that Cielo successfully leverage DVFS to reduce power consumption. Note that response time increases as power consumption decreases because they are conflicting.

Figs. 3 to 6 demonstrate that Cielo successfully evolves deployment strategies so that applications improve their objective values with respect to a given objective(s).

In Figs. 7 and 8, two objectives are considered simultaneously. All objectives are considered simultaneously in Fig. 9. As these figures show, Cielo successfully balance objective values by following the trade-offs among given objectives.

Table 3 compares Cielo with a well-known evolutionary multiobjective genetic algorithm, NSGA-II [4]1, as well as existing heuristics, FFA (first-fit algorithm) and BFA (best-fit algorithm), which have been widely used for VM placement in clouds [1, 7, 15, 16]. The table shows the minimum, average and maximum objective values in the last generation. In all objectives, Cielo outperforms NSGA-II. 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%). On average, Cielo outperforms NSGA-II by 24.19%. With DVFS disabled, 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. BFA is the best in CPU allocation and the worst in power consumption because it is designed to deploy VMs on the hosts that maintain higher resource availability. With all five objectives considered, Cielo maintains balanced objective values between FFA and BFA while yielding the best performance in response time and bandwidth allocation.

Table 4 shows the variance of objective values that Cielo and NSGA-II yield at the last generation in 20 different simulation runs. A lower variance means higher stability (or higher similarity) in objective value results (i.e., lower oscillation in objective value results) among different simulation runs. Cielo consistently maintains higher stability than NSGA-II. Cielo’s average stability is 29.32% higher than NSGA-II’s. This result exhibits Cielo’s stability property (i.e., ability to seek evolutionarily stable strategies), which NSGA-II does not have.

Fig. 11 shows the time required for Cielo and NSGA-II 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., 400 generations), Cielo is 74.1% faster than NSGA-II.

### Figure 10: # of hosts at each P-state in the last generation

1 NSGA-II and Cielo use the same population size and the same limit of generations. All other configurations are borrowed from [4].

7. RELATED WORK

Numerous research efforts have been made to study heuristic algorithms for application placement problems in clouds (e.g., [1, 3, 7, 13, 15, 16, 21, 23]). Most of them assume single-tier application architecture and considers a single objective. For example, in [3, 13, 21, 23], only energy saving is considered...
Figure 3: Configuration $C_1$: With the CPU Allocation Objective Considered

Figure 4: Configuration $C_2$: With the Bandwidth Allocation Objective Considered

Figure 5: Configuration $C_3$: With the Response Time Objective Considered

Figure 6: Configuration $C_4$: With the Power Consumption Objective 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 [5, 12, 24], task allocation [18] and data replication [11]. In [5, 12, 24], 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., [19, 22]) and other stochastic optimization algorithms (e.g., [2, 6]) 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.

This paper reports a set of extensions to the authors’ prior work [14]. For the problem formulation, this paper considers two extra objectives (bandwidth allocation and power consumption) and an extra parameter in each deployment strategy (bandwidth allocation), all of which are not studied in [14]. This paper also considers constraints in CPU and bandwidth allocation and investigates a constraint-handling algorithm (Section 4), while no constraints are assumed in [14]. Moreover, this paper carries out larger-scale simulations with
8. CONCLUSIONS

This paper proposes and evaluates Cielo, an evolutionary game theoretic framework for adaptive and stable application deployment in DVFS-enabled clouds. It theoretically guarantees that every application seeks an evolutionarily stable deployment strategy, which is an equilibrium solution under given workload and resource availability. Simulation results verify that Cielo performs application deployment in an adaptive and stable manner. Cielo outperforms existing well-known heuristics in the quality, stability and computational cost of application deployment.

9. REFERENCES


