Balancing Performance, Resource Efficiency and Energy Efficiency for Virtual Machine Deployment in DVFS-enabled Clouds: An Evolutionary Game Theoretic Approach

Yi 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

Chonho Lee
Nanyang Technological University
Singapore
leechonho@ntu.edu.sg

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

Katsuya Oba
OGIS International, Inc.
San Mateo, CA 94404, USA
oba@ogis-international.com

ABSTRACT

This paper proposes and evaluates a multiobjective evolutionary game theoretic framework for adaptive and stable application deployment in clouds that support dynamic voltage and frequency scaling (DVFS) for CPUs. The proposed framework, called Cielo, aids cloud operators to adapt the resource allocation to applications and their locations according to the operational conditions in a cloud (e.g., workload and resource availability) with respect to multiple conflicting objectives such as response time performance, recourse utilization and power consumption. Moreover, Cielo theoretically guarantees that each application performs an evolutionarily stable deployment strategy, which is an equilibrium solution under given operational conditions. Simulation results verify this theoretical analysis; applications seek equilibria to perform adaptive and evolutionarily stable deployment strategies. Cielo allows applications to successfully leverage DVFS 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, multiobjective optimization, evolutionary game theory

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2605693.

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 $\sum_{t=1}^{3} c_t$ where c_t denotes the CPU time share allocated to the t-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_t^p denotes the time required for the t-th tier server to process a message.

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

T^w is computed as follows.

$$T^w = \frac{1}{\lambda} \sum_{t=1}^{3} \frac{\rho_t^2}{1-\rho_t} \quad \text{where } \rho_t = \lambda_t \frac{T_t^p}{c_t \frac{f_t}{f_{max}}} \tag{2}$$

λ is the message arrival rate for an application (i.e., the number of messages the application receives from users in the unit time). $\lambda = \frac{1}{3}\sum_{t=1}^{3} \lambda_t$. (Currently, $\lambda = \lambda_1 = \lambda_2 = \lambda_3$.) ρ_t is the CPU utilization of the t-th tier server. f_{max} and f_t are the maximum CPU frequency and the CPU frequency of a host that the t-th tier server resides on.

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

$$T^c = \sum_{t=1}^{2} T^c_{t \to t'} \approx \sum_{t'=2}^{3} \frac{B \cdot \lambda_{t'}}{b_t}, \quad t' = t+1 \tag{3}$$

Power Consumption: This objective indicates the total power consumption (in W) by the hosts that operate three VMs in an application.

$$\sum_{t=1}^{3} \{P_{idle}^{f_t} + (P_{max}^{f_t} - P_{idle}^{f_t}) \cdot c_t \cdot \frac{f_t}{t_{max}}\} \tag{4}$$

$P_{idle}^{f_t}$ and $P_{max}^{f_t}$ denote the power consumption of a host that the t-th tier server resides on when its CPU utilization is 0% and 100% at the frequency of f_t, respectively.

Cielo considers a CPU capacity constraint: $w_i \leq 1$ for all M hosts. w_i is the total CPU share allocated to the i-th host. The violation of this constraint is computed as:

$$c^v = \sum_{i=1}^{M} (I_i \cdot (w_i - 1)) \tag{5}$$

$I_i = 1$ if $w_i > 1$. Otherwise, $I_i = 0$.

Cielo also considers a bandwidth capacity constraint: $y_i \leq 1$ for all M hosts. y_i is the total bandwidth allocated to the i-th host (in percentage). The violation of bandwidth constraint is computed as:

$$b^v = \sum_{i=1}^{M} (I_i \cdot (y_i - 1)) \tag{6}$$

$I_i = 1$ if $y_i > 1$. Otherwise, $I_i = 0$.

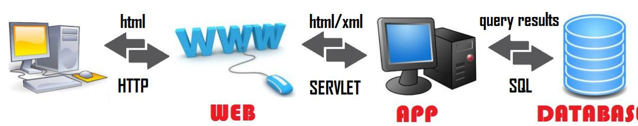


Figure 1: Three Tiers of Web, Application and Database Servers

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 ℓ. When a player is drawn for a game, the probabilities that its opponent plays k and ℓ are $1-x$ and x, respectively. Thus, the expected payoffs for the player to play k and ℓ 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*

$$U(k,\ x\ell + (1-x)k) > U(\ell,\ x\ell + (1-x)k) \tag{7}$$

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

If the payoff function is linear, Equation 7 derives:

$$(1-x)U(k,k) + xU(k,\ell) > (1-x)U(\ell,k) + xU(\ell,\ell) \quad (8)$$

If x is close to zero, Equation 8 derives either

$$U(k,k) > U(\ell,k) \text{ or } U(k,k) = U(\ell,k) \text{ and } U(k,\ell) > U(\ell,\ell) \quad (9)$$

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 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 [20]. Let $\lambda_k(t) \geq 0$ 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 $\lambda(t) = \sum_{k=1}^{|K|} \lambda_k(t)$. Let $x_k(t) = \lambda_k(t)/\lambda(t)$ be the population share of players who play k at time t. The population state is defined by $X(t) = [x_1(t), \cdots, x_k(t), \cdots, x_K(t)]$. 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 $U(X,X) = \sum_{k=1}^{|K|} x_k \cdot U(k,X)$. 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.

$$\dot{x}_k = x_k \cdot [U(k,X) - U(X,X)] \quad (10)$$

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* $x_k(t)_{t\to\infty} \to 0$.

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 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, $\{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$, 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:

$$s(a_i) = \bigcup_{t \in 1,2,3} (h_{i,t}, c_{i,t}, b_{i,t}, f_{i,t}), \quad 1 < i < N \quad (11)$$

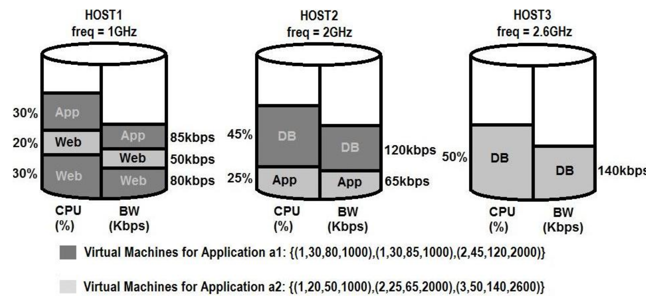


Figure 2: Example Deployment Strategies

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. $c_{i,t}$ 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 80 Kbps bandwidth for the VM ($c_{1,1} = 30$ and $b_{1,1} = 80$). The second-tier VM is placed on host 1 ($h_{1,2} = 1$), 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 operates at 2 GHz CPU frequency and consumes 45% CPU share and 120 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

```
1:  g = 0
2:  Randomly generate the initial N populations for N applications: P = {P_1, P_2, ..., P_N}
3:  while g < G_max do
4:      for each population P_i randomly selected from P do
5:          P'_i ← ∅
6:          for j = 1 to |P_i|/2 do
7:              s_1 ← randomlySelect(P_i)
8:              s_2 ← randomlySelect(P_i)
9:              winner ← performGame(s_1, s_2)
10:             replica ← replicate(winner)
11:             if random() ≤ P_m then
12:                 replica ← mutate(winner)
13:             end if
14:             P_i \ {s_1, s_2}
15:             P'_i ∪ {winner, replica}
16:         end for
17:         P_i ← P'_i
18:         d_i ← argmax_{s∈P_i} x_s
19:         while d_i is infeasible do
20:             P_i \ {d_i}
21:             d_i ← argmax_{s∈P_i} x_s
22:         end while
23:         Deploy VMs for the current application based on d_i.
24:     end for
25:     g = g + 1
26: end while
```

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 (x_s) 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 ($c^v = 0$ 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 $s_i \succ s_j$) iif:

- 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 $s_1 \succ s_2$.
- Both s_1 and s_2 are infeasible, and s_1's constraint violation is lower than s_2's ($c^v_{s_1} + b^v_{s_1} < c^v_{s_2} + b^v_{s_2}$).

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 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.
- $X(t) = \{x_1(t), x_2(t), \cdots, x_{|S^*|}(t)\}$ 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.
- $p^s_k = x_k \cdot \phi(F_s - F_k)$ 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:

$$\begin{aligned}\dot{x}_s &= \sum_{k\in S^*, k\neq s}\{x_s p^s_k - x_k p^k_s\} \\ &= x_s \sum_{k\in S^*, k\neq s} x_k\{\phi(F_s - F_k) - \phi(F_k - F_s)\} \quad (12)\end{aligned}$$

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

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: $X(t)_{t\to\infty} = \{x_1(t), x_2(t), \cdots, x_{|S^*|}(t)\}_{t\to\infty} = \{1, 0, \cdots, 0\}$. □

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 $x_1 = 1 - x_2 - \cdots - x_{|S^*|}$

$$\dot{z}_s = z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|s^*|} z_i \cdot c_{si}], \quad s, k = 2, ..., |S^*| \quad (13)$$

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

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

$$\begin{aligned}J_{sk} &= \left[\frac{\partial \dot{z}_s}{\partial z_k}\right]_{|Z=Z_{eq}} \\ &= \left[\frac{\partial z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|S^*|} z_i \cdot c_{si}]}{\partial z_k}\right]_{|Z=Z_{eq}} \quad (14) \\ & for\ s, k = 2, ..., |S^*|\end{aligned}$$

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

$$J = \begin{bmatrix} c_{21} & 0 & \cdots & 0 \\ 0 & c_{31} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{|S^*|1}\end{bmatrix} \quad (15)$$

$c_{s1} = -\phi(F_1 - F_s) < 0$ for all s; therefore, $Z_{eq} = \{0, 0, \cdots, 0\}$ is asymptotically stable. □

6. SIMULATION EVALUATION

This section evaluates Cielo's adaptability and stability through simulations. This paper uses a simulated cloud data center that consists of 100 hosts in a 10×10 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 1: Message Arrival Rate and Message Processing Time

Application type	1	2	3	4	5
Message arrival rate (λ)	110	70	40	20	10
Web server (T^p_1)	0.02	0.02	0.04	0.04	0.08
App server (T^p_2)	0.03	0.08	0.04	0.13	0.11
DB server (T^p_3)	0.05	0.05	0.12	0.08	0.11

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_{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

CPU frequency (f)	P^f_{idle}	P^f_{max}
1.0 GHz	34 W	68 W
1.8 GHz	51 W	80 W
2.0 GHz	55 W	84 W
2.2 GHz	66 W	89 W
2.4 GHz	90 W	97 W
2.6 GHz	96 W	108 W

Figs. 3 to 9 illustrate how Cielo evolves deployment strategies through generations and improve 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 miliseconds 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 in 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.

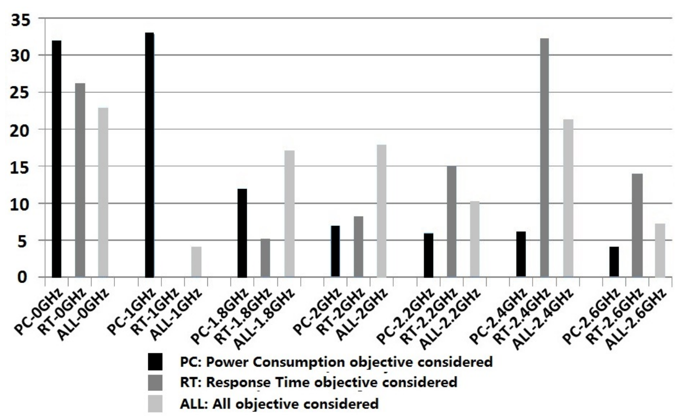
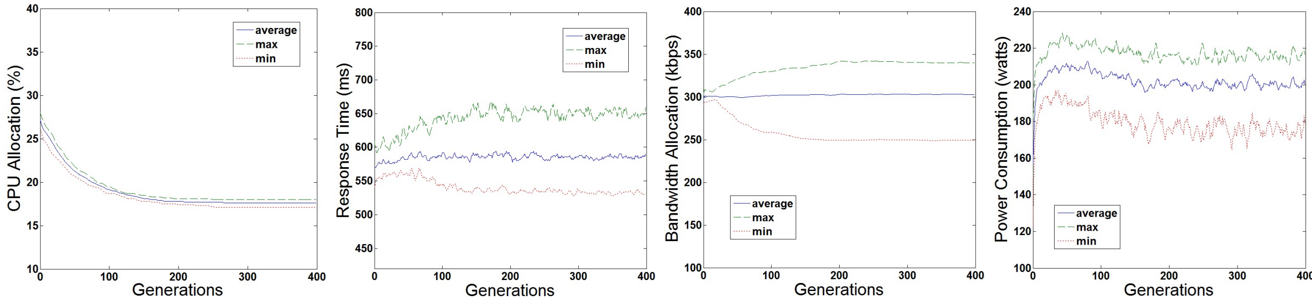


Figure 10: # of hosts at each P-state in the last generation

7. RELATED WORK

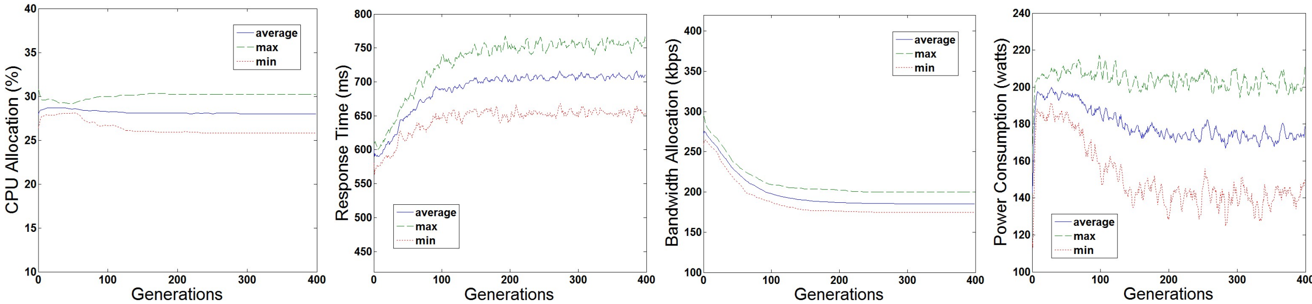
Numerous research efforts have been made to study heuristic algorithms for application placement problems in clouds (e.g., [1,3,7,13,15,16,21,23]). Most of them assume single-tier application architecture and considers a single objective. For example, in [3,13,21,23], only energy saving is considered

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



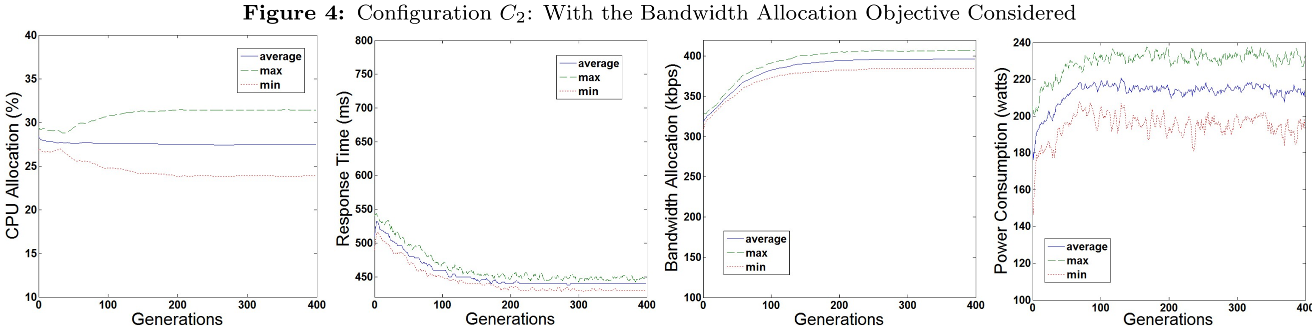
(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

Figure 3: Configuration C_1: With the CPU Allocation Objective Considered



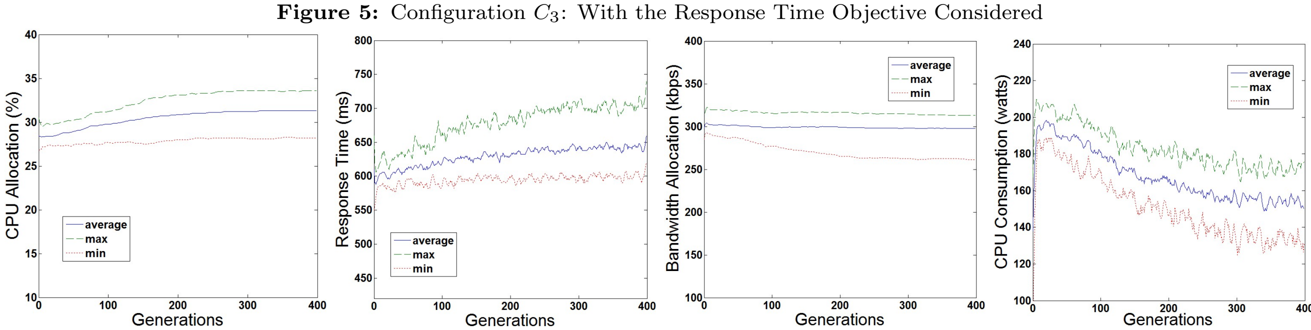
(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

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



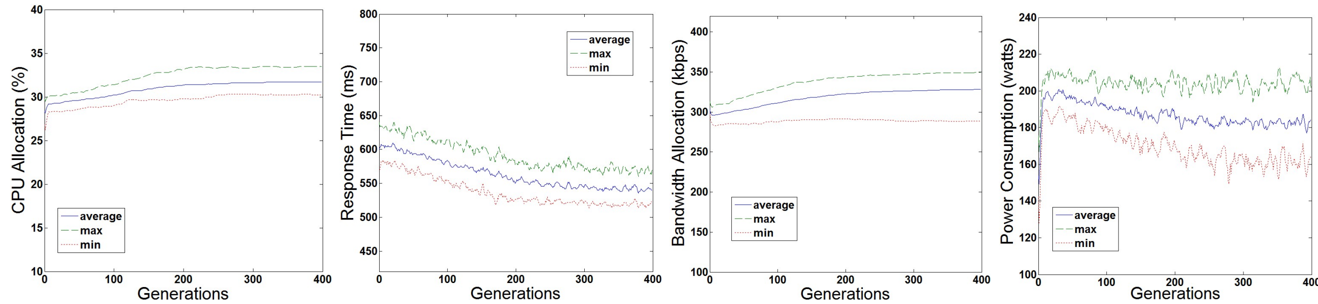
(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

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



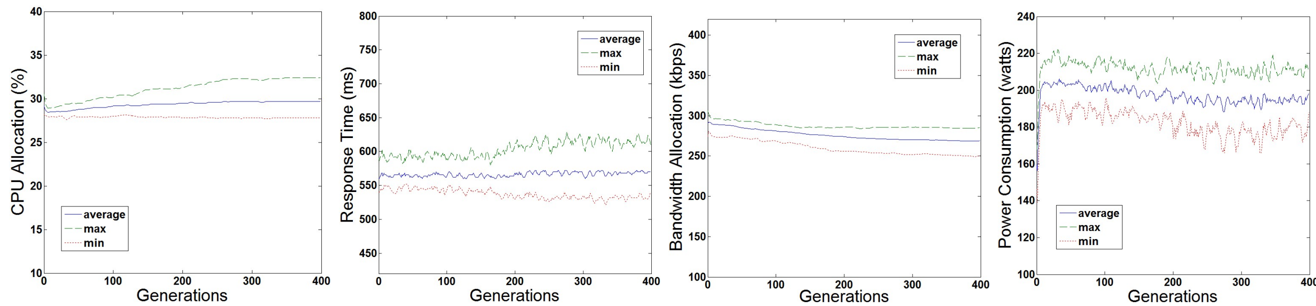
(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

Figure 6: Configuration C_4: With the Power Consumption Objective Considered



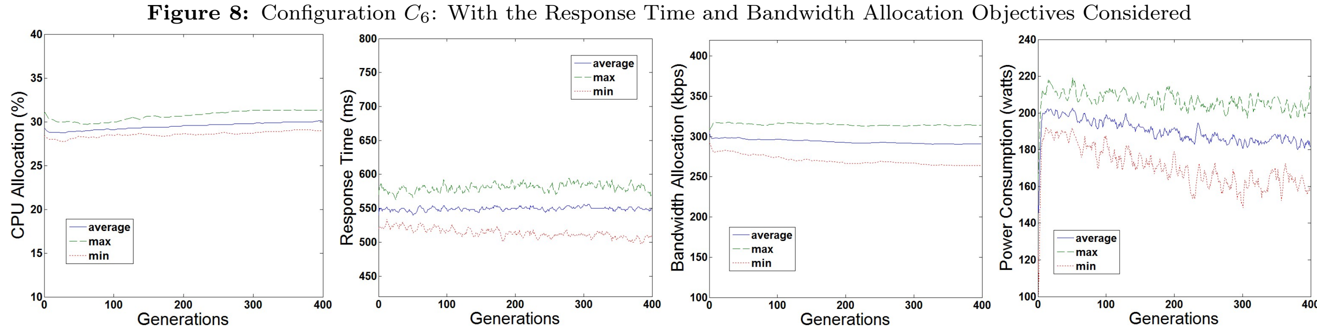
(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

Figure 7: Configuration C_5: With the Response Time and Power Consumption Objectives Considered



(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

Figure 8: Configuration C_6: With the Response Time and Bandwidth Allocation Objectives Considered



(a) CPU allocation **(b)** Response time **(c)** Bandwidth allocation **(d)** Power consumption

Figure 9: Configuration C_7: With All Four Objectives Considered

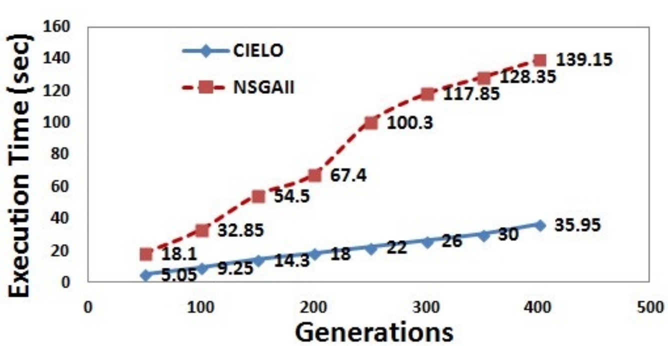


Figure 11: Computational Costs for Cielo and NSGA-II

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

Table 3: Performance of Cielo, NSGA-II, FFA and BFA

Objectives		Min	Avg	Max
Power consumption (W)	Cielo w/ DVFS	4,370	5,000	5,830
	Cielo w/o DVFS	8,806	9,090	9,280
	NSGA-II w/ DVFS	6,143	6,643	6,960
	NSGA-II w/o DVFS	10,032	10,033	10,036
	FFA w/o DVFS	4,380	4,380	4,381
	BFA w/o DVFS	10,090	10,103	10,127
CPU allocation (%/host)	Cielo w/ DVFS	68.4	70.4	72
	Cielo w/o DVFS	65.6	68.8	71.6
	NSGA-II w/ DVFS	85.5	86.1	86.4
	NSGA-II w/o DVFS	83.4	84.9	85.8
	FFA w/o DVFS	96.9	96.9	96.9
	BFA w/o DVFS	40.65	40.65	40.65
Bandwidth allocation (kbps/host)	Cielo w/ DVFS	430	440	450
	Cielo w/o DVFS	420	420	430
	NSGA-II w/ DVFS	530	540	550
	NSGA-II w/o DVFS	520	530	530
	FFA w/o DVFS	290	290	290
	BFA w/o DVFS	305	305	305
Response time (msec)	Cielo w/ DVFS	174.9	185.51	200.15
	Cielo w/o DVFS	172.21	190	204
	NSGA-II w/ DVFS	282.3	287	290.6
	NSGA-II w/o DVFS	287	289.37	290.75
	FFA w/o DVFS	206	206	206
	BFA w/o DVFS	203.75	203.75	203.75

Table 4: Stability of Objective Values in Cielo and NSGA-II

Objectives	Cielo	NSGA-II	Diff (%)
Power consumption (W)	150	199.3	24.73
CPU allocation (%)	17.6	28.7	38.67
Response time (ms)	440	540	18.51
Bandwidth allocation (kbps)	185.51	287.08	35.38
Average Difference (%)			29.32

more realistic configurations than [14], which uses only five hosts and simple topologies among them on a cloud that never support DVFS.

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

[1] H. Casanova, M. Stillwell, and F. Vivien. Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. In *Proc. IEEE Int'l Parallel & Distributed Processing Symposium*, 2012.

[2] X. Chang, B. Wang, L. Jiqiang, W. Wang, and K. Muppala. Green cloud virtual network provisioning based ant colony optimization. In *Proc. ACM Int'l Conference on Genetic and Evol. Computat*, 2013.

[3] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders. Blackbox prediction of the impact of DVFS on end-to-end performance of multitier systems. *ACM SIGMETRICS Performance Eval. Rev.*, 37(4), 2010.

[4] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proc. Conf. Parallel Problem Solving from Nature*, 2000.

[5] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos. Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing. *Elsevier Computer Comm.*, 30(3), 2007.

[6] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Computer and System Sciences*, 79(8), 2013.

[7] H. Goudarzi and M. Pedram. Energy-efficient virtual machine replication and placement in a cloud computing system. In *Proc. IEEE Int'l Conf. on Cloud Comput.*, 2013.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proc. of ACM SIGCOM*, 2009.

[9] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *Proc. of ACM SIGCOM*, 2008.

[10] B. Kerby. Managing data center power and cooling with AMD Opteron processors and AMD PowerNow! technology. Technical report, Dell Inc., 2007.

[11] S. Khan and I. Ahmad. A pure Nash equilibrium based game theoretical method for data replication across multiple servers. *IEEE T. Knowl. Data En.*, 21(4), 2009.

[12] S. U. Khan and C. Ardil. Energy efficient resource allocation in distributed computing systems. In *Proc. of Int'l Conf. on Distrib., High-Perf. and Grid Comp.*, 2009.

[13] D. Kliazovich, P. Bouvry, and S. U. Khan. DENS: data center energy-efficient network-aware scheduling. *Cluster Computing*, 16(1), 2013.

[14] C. Lee, J. Suzuki, A. V. Vasilakos, Y. Yamano, and K. Oba. An evolutionary game theoretic approach to adaptive and stable application deployment in clouds. In *Proc. IEEE Workshop on Bio-Inspired Algorithms for Distributed Systems*, 2010.

[15] X. Lia, Z. Qiana, S. Lua, and J. Wu. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Mathematical and Computer Modelling*, 58(5-6), 2013.

[16] F. Ma, F. Liu, and Z. Liu. Multi-objective optimization for initial virtual machine placement in cloud data center. *J. Infor. and Computational Science*, 9(16), 2012.

[17] J.-M. Pierson and H. Casanova. On the utility of DVFS for power-aware job placement in clusters. In *Proc. Int'l Conf. on Parallel Processing*, 2011.

[18] R. Subrata, A. Y. Zomaya, and B. Landfeldt. Game theoretic approach for load balancing in computational grids. *IEEE Trans. Parall. Distr.*, 19(1), 2008.

[19] H. A. Taboada, J. F. Espiritu, and D. W. Coit. MOMS-GA: A Multi-Objective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems. *IEEE Trans. Reliability*, 57(1), 2008.

[20] P. Taylor and L. Jonker. Evolutionary stable strategies and game dynamics. *Elsevier Mathematical Biosci.*, 40(1), 1978.

[21] von Laszewski, L. Wang, A. J. Younge, and X. He. Power-aware scheduling of virtual machines in DVFS-enabled clusters. In *Proc. IEEE Int'l Conf. on Clusters*, 2009.

[22] H. Wada, J. Suzuki, Y. Yamano, and K. Oba. E3: A multiobjective optimization framework for sla-aware service composition. *IEEE Trans. Services Computing*, 5(3), 2012.

[23] Q. Wang, Y. Kanemasa, J. Li, C. A. Lai, M. Matsubara, and C. Pu. Impact of DVFS on n-tier application performance. In *Proc. ACM Conference on Timely Results in Operating Systems*, 2010.

[24] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *J. Supercomputing*, 54(2), 2009.