IET Cyber-Physical Systems: Theory & Applications

Research Article

Hierarchical resource allocation and consolidation framework in a multi-core server cluster using a Markov decision process model

ISSN 2398-3396
Received on 26th June 2017
Revised 29th September 2017
Accepted on 11th October 2017
doi: 10.1049/iet-cps.2017.0060
www.ietdl.org

Pu Zhao[1], *Xue Lin*[1] ✉, *Yanzhi Wang*[2], *Shuang Chen*[3], *Massoud Pedram*[3]

[1]*Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA*
[2]*Electrical Engineering & Computer Science, Syracuse University, Syracuse, NY, USA*
[3]*Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA*
✉ *E-mail: xuelin@ece.neu.edu*

Abstract: This paper investigates a service level agreements (SLAs)-based resource allocation problem in a server cluster. The objective is to maximise the total profit, which is the total revenue minus the operational cost of the server cluster. The total revenue depends on the average request response time, whereas the operating cost depends on the total energy consumption of the server cluster. A joint optimisation framework is proposed, comprised of request dispatching, dynamic voltage and frequency scaling (DVFS) for individual cores of the servers, as well as server- and core-level consolidations. Each DVFS-enabled core in the server cluster is modelled by using a continuous-time Markov decision process (CTMDP). A near-optimal solution comprised of a central manager and distributed local agents is presented. Each local agent employs linear programming-based CTMDP solving method to solve the DVFS problem for the corresponding core. On the other hand, the central manager solves the request dispatch problem and finds the optimal number of ON cores and servers, thereby achieving a desirable tradeoff between service response time and power consumption. To reduce the computational overhead, a two-tier hierarchical solution is utilized. Experimental results demonstrate the outstanding performance of the proposed algorithm over the baseline algorithms.

1 Introduction

Nowadays cloud computing has emerged as a new computing paradigm that enables ubiquitous, convenient, on-demand network access to a large amount of remote, distributed, shared computing resources [1, 2]. In cloud computing, various kinds of computing resources are provided to users as services and the users have access to computing resources (e.g. networks, servers, storage, applications, and services) based on their needs from anywhere on earth [3]. The cloud computing has been known as software as a service, infrastructure as a service, and platform as a service [4–6].

In the cloud computing paradigm, there are basically two parties, the cloud service providers (CSPs) and the clients, who act their own parts through providing and usage of computing resources. While enjoying the convenient on-demand access to computing resources or services, the clients need to pay for such accesses. CSPs can make profits by providing services and charging the clients. Clients can avoid the costs associated with 'in-house' provisioning of computing resources and have access to a larger pool of computing resources than they can possibly own by themselves. There are many different aspects to evaluate the cloud computing service quality, which can be optimised with various methods. For example, the computing resource and fairness of the allocation are taken into account in a framework based on game theory [7]. An online mechanism based on auction is proposed to determine the resource allocation by considering both the incentives of clients and the incentives of the CSPs [8]. Wang *et al.* [9] advocates a joint optimisation framework to integrate virtual machine assignment and traffic engineering for achieving energy efficiency. To consider various aspects of cloud service and ensure the quality of service delivery, the clients need guarantees from CSPs, which are brokered between the CSPs and the clients as service level agreements (SLAs) specifying the quality of the delivered services from CSPs to clients. Usually, SLAs cover guaranteed amounts of computing power, storage space, and network bandwidth, and also availability and security aspects of the provided services.

The backbones of the cloud computing paradigm are supported by data centres and server clusters with massive computing resources, which are monitored and maintained by CSPs [10–12]. In order to meet SLAs on the quality of delivered services, CSPs often over-provisioning their computing resources being prepared for the worst scenarios, when demands of services are gusty [11]. Although over-provisioning of computing resources can guarantee the quality of services, it increases the operation incurred costs of the data centres or server clusters. For example, as the ever-increasing energy consumption of CSPs emerges as a significant problem, energy efficient frameworks for resource allocation in cloud computing become imperatively essential to prevent server overheating and reduce carbon emissions [13, 14]. Under such circumstances, optimal provisioning of the cloud computing resources and effective allocation has become an active research topic. The aim is to reduce the operation incurred cost of the servers (and also the environment impacts) while satisfying the brokered SLAs between CSPs and clients. The authors in [15–18] have conducted research on this topic by formulating and solving the optimal request dispatch and resource allocation in the cloud computing paradigm. In the past decade, the more general problems of resource allocation and management have been actively investigated, for example, in electronic commerce scenarios as in [19], in autonomic computing scenarios as in [20, 21], in grid computing scenarios as in [22, 23], and in clusters of servers scenarios as in [24].

In this work, we aim at solving the optimisation problem of cloud computing resource allocation with SLA. We are considering a multi-server, multi-client cloud computing framework. The server cluster contains possibly heterogeneous servers and the clients generate service requests for processing in the server cluster. The clients can be any software applications, which have computation, storage, and communication requirements to be satisfied by the server cluster. Moreover, according to the SLAs,

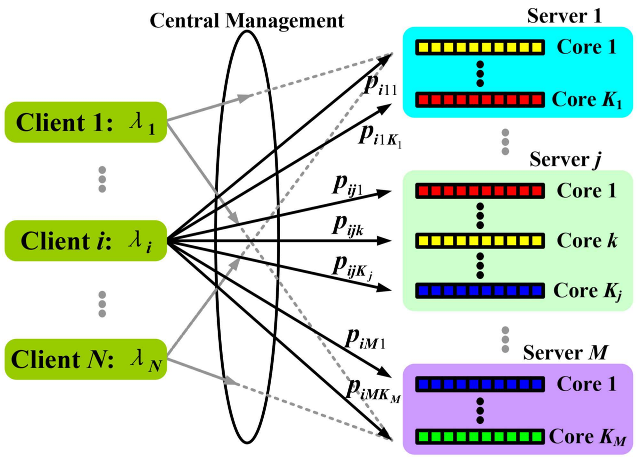

Fig. 1 *Architecture of the request dispatching and DVFS problem in the cloud computing system*

the clients have constraints on the response time of their service requests and pre-defined utility functions are used to express such constraints. The potentially heterogeneous servers in the server cluster are comprised of numbers of cores, which could be again heterogeneous. The CSP of the server cluster earns revenue from charging the clients. The cost to the CSP is the operating cost of the ON servers. Then the profit made by the CSP is the revenue minus the cost.

Different from prior works, in this work we are using a joint optimisation framework, in which techniques such as the optimal request dispatching, the dynamic voltage and frequency scaling (DVFS) in each core, as well as server- and core-level consolidations, are employed in the optimisation. In our formulation, each DVFS-enabled core is modelled as a continuous-time Markov decision process (CTMDP), where we choose clock frequency (and hence, the corresponding power supply voltage) as the action for service request processing, because we can achieve a tradeoff between shorter execution time of service requests and lower power consumption of the cores by the appropriate frequency control.

We present a near-optimal solution to the cloud computing resource allocation problem. Our solution algorithm consists of a central resource manager and several distributed local agents. The distributed local agents are employed to parallelise the solutions for single cores and decrease the optimisation time of the overall cloud computing resource allocation problem, thereby significantly increasing the solution scalability. In our solution, a distributed local agent needs to solve the DVFS problem for the local core, which are implemented by a linear programming-based Markov decision process solving method [25] that judiciously selects the most appropriate clock frequency (and therefore voltage) based on the number of waiting requests. On the other hand, the central manager solves the request dispatch problem based on the optimisation results from the distributed local agents. In order to reduce the computational overhead, the central manager is realised as a two-tier hierarchical controller, in which the first control tier associates each client with a selected set of servers, i.e. service requests generated from the client can only be dispatched to cores of the chosen set of servers. In the second control tier, the request dispatch problem is solved based on the results of the first tier. Finally, the central manager performs core- and server-level consolidations by finding the optimal numbers of ON cores and ON servers for request processing. In this way, we aim at achieving a desirable balance between the static and dynamic power consumptions of the server cluster [26, 27]. Experimental results demonstrate that the proposed near-optimal hierarchical resource allocation and consolidation algorithm can consistently achieve better performance compared with the baseline algorithms.

The remainder of this paper is organised as follows. Section 2 presents the overall system modelling. Section 3 formulates the cloud computing resource allocation problem. Section 4 provides the near-optimal solution including the distributed local agents, central manager, and resource consolidation. Section 5 presents the experimental results and Section 6 concludes the paper.

2 Cloud computing system model

2.1 System modelling: overall

The target resource allocation system consists of a set of N clients, a server cluster of M potentially heterogeneous servers, and a central resource management node, as shown in Fig. 1. The central resource manager is the information exchange point between the clients and the server cluster and has the control over the whole system.

In the cloud computing system, a client is indexed with i ($1 \le i \le N$) and a server is indexed with j ($1 \le j \le M$). Each jth server in the server cluster consists of K_j potentially heterogeneous cores, which are indexed with k. Clients generate service requests to be processed by potentially multiple cores in more than one servers in the server cluster. The service requests generated from the ith client will be assigned to the kth core in the jth server with probability p_{ijk}, which are the optimisation variables in the request dispatch and DVFS optimisation framework.

We assume that the service requests generated by each ith client follow a Poisson process with an average generating rate of λ_i, which can be predicted based on the client's past behaviour pattern. This assumption enables an analytical form of the response time. According to the properties of the Poisson distribution and the above assumption, service requests generated from the ith client and dispatched to the kth core in the jth server also follow a Poisson process with an average rate of $p_{ijk} \cdot \lambda_i$ [28]. The average service request arrival rate of the kth core in the jth server is then given by $\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i$.

2.2 CTMDP service queue (SQ) modelling

The SQ in each core can be modelled as a CTMDP, which is a controllable continuous-time Markov process satisfying the Markovian property [25]. The CTMDP is a more detailed and accurate model than the generalised processor sharing model [29]. A CTMDP can be constructed and represented by a set of states $\boldsymbol{S}$ and a finite set of actions $\boldsymbol{A}$. The state transition rates are determined by actions $a \in \boldsymbol{A}$. The controller is charged by a cost rate function $c(s, a)$ when it is at state $s \in \boldsymbol{S}$ and action $a \in \boldsymbol{A}$ is chosen. A policy $\pi = \{(s, a) | s \in \boldsymbol{S}, a \in \boldsymbol{A}\}$ is a set of state–action pairs for all states of the CTMDP. We use notation $\pi(s) = a$ to specify that action a is chosen for state s according to policy π. We assume the stationary policy class. We would like to derive the optimal policy that minimises the total expected cost.

Given a CTMDP with n states, we define its parameterised generator matrix $\boldsymbol{G}(a)$ by an $n \times n$ matrix. Each entry $\sigma_{s,s'}(a)$ in $\boldsymbol{G}(a)$ is the transition rate from state s to state s' when action a is chosen, calculated as

$$\sigma_{s,s'}(a) = \frac{\delta_{s,s'}(a)}{\tau_{s,s'}(a)}, \quad \text{for } s \neq s', \tag{1}$$

where $\tau_{s,s'}(a)$ is the average transition time from state s to state s' when action a is chosen. $\delta_{s,s'}(a) = 1$ if s' is one of the destination states of state s under action a, and $\delta_{s,s'}(a) = 0$ otherwise. More details can be found in [25, 30] about CTMDP.

Fig. 2 models the SQ in each kth core in the jth server ($1 \le j \le M, 1 \le k \le K_j$) as a stationary CTMDP. The state set is $\boldsymbol{S} = \{0, 1, 2, \ldots, Q\}$, where Q is the maximum length of the queue. The action set is $\boldsymbol{A} = \{f_{\min}, \ldots, f_{\max}\}$, where an action is a specific execution frequency of the core. If a service request coming from a client is dispatched to this SQ, the state of the SQ is increased by one unless the SQ is full already. Independent of the action chosen, the SQ transits from state s to state $s + 1$ with the rate of $\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i$. After a service request has been processed by the core, the index of the SQ state is autonomously decremented by one. The transition rate $\mu_{jk}(a)$ from state s to state $s - 1$, as shown

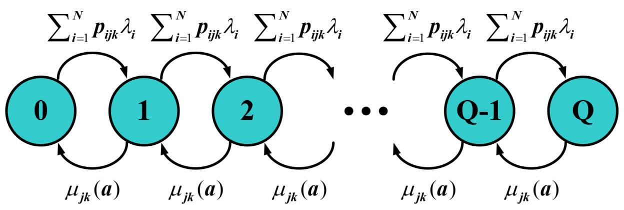

Fig. 2 *CTMDP model of a SQ*

in Fig. 2, is determined by the action (execution frequency) a chosen by the core. A higher execution frequency will result in a shorter average request service time, and thereby, a larger transition rate $\mu_{jk}(a)$, and vice versa.

The average *response time* of a service request in the kth core of the jth server, denoted by $\bar{R}_{jk}$, is the sum of the average waiting time in the SQ and the average request processing time. $\bar{R}_{jk}$ depends on the policy π_{jk} chosen in the core and the dispatching probability vector $\boldsymbol{p}_{jk} = (p_{1jk}, p_{2jk}, \dots, p_{Njk})$, denoted by $\bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$, with reasons shown in the following. Let $\tilde{p}_{0jk}, \tilde{p}_{1jk}, \dots, \tilde{p}_{Qjk}$ denote the steady-state probabilities of the CTMDP states 0 to Q, respectively, which can be derived from the policy π_{jk} and the arrival rate $\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i$ using standard queueing theory method [31]. Based on the Little's law [31], $\bar{R}_{jk}$ is given by

$$\bar{R}_{jk} = \frac{\sum_{s=0}^{Q} \tilde{p}_{sjk} \cdot s}{\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i}. \tag{2}$$

Power consumption of each core is comprised of two parts: a *dynamic power consumption* part for the core is active (i.e. when it is processing service requests) and a static power consumption part as long as the core is turned ON (i.e. no matter whether the core is active or idle.) The dynamic power consumption $P^c_{dy,jk}$ is a superlinear function of the execution frequency a of the core, denoted by $P^c_{dy,jk}(a)$ [32–34]. A higher execution frequency will result in larger dynamic power consumption. Given the policy π_{jk} for the kth core in the jth server, the execution frequency $a = \pi_{jk}(s)$ is chosen at each state $1 \le s \le Q$ (Please note that the dynamic power consumption is zero when $s = 0$.). Then the average dynamic power consumption $\bar{P}^c_{dy,jk}$ is given by

$$\bar{P}^c_{dy,jk} = \sum_{s=1}^{Q} \tilde{p}_{sjk} \cdot P^c_{dy,jk}(\pi_{jk}(s)). \tag{3}$$

Hence, similar to $\bar{R}_{jk}$, the average dynamic power consumption $\bar{P}^c_{dy,jk}$ also depends on π_{jk} and $\boldsymbol{p}_{jk}$, denoted by $\bar{P}^c_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$. On the other hand, the static power consumption of each kth core in the jth server is a constant value as long as the central manager keeps the core ON. We use $\bar{P}^c_{st,jk}$ to denote the static power consumption in the sense that it is an average value. The effect of power consumption is captured by the cost rate function, due to the fact that power consumption is directly related to the energy cost. More details of the cost rate function can be found in Section 4.

3 Problem formulation

We use $U_i(R)$ to denote the non-increasing utility function of the ith client as a function of the average response time R. We use x_j as a pseudo-Boolean integer to indicate whether the jth server is ON ($x_j = 1$) or OFF ($x_j = 0$). When the jth server is turned ON, it incurs an (average) idle power consumption of $\bar{P}^s_{id,j}$. We use y_{jk} as another pseudo-Boolean integer to represent if the kth core in the jth server is ON ($y_{jk} = 1$) or OFF ($y_{jk} = 0$). As the cores are within servers, when $x_j = 0$, we have $y_{jk} = 0$ for all $1 \le k \le K_j$. For the optimisation problem, we use p_{ijk}'s, π_{jk}'s, x_j's, and y_{jk}'s as optimisation variables. The rest of the parameters are either constants or associative to these optimisation variables.

We formulate the overall request dispatch, DVFS, and consolidation problem for the server cluster over a time period T as a profit maximisation problem as follows:

Find the optimal p_{ijk}'s, π_{jk}'s, x_j's, and y_{jk}'s.

Maximise

$$T \cdot \sum_{i=1}^{N} \lambda_i \cdot U_i(\bar{R}_i) - \text{Price} \cdot T \cdot \times \sum_{j=1}^{M} \left(x_j \cdot \bar{P}^s_{id,j} + \sum_{k=1}^{K_j} y_{jk} \cdot \left(\bar{P}^c_{st,jk} + \bar{P}^c_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk}) \right) \right) \tag{4}$$

where $\bar{R}_i = \sum_{j=1}^{M} \sum_{k=1}^{K_j} p_{ijk} \cdot \bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ is the average response time of the service requests of the ith client.

Subject to

$$x_j \in \{0,1\}, \quad \text{for} \quad \forall j \in \{1,2,\dots,M\}, \tag{5}$$

$$y_{jk} \in \{0,1\}, \quad \text{for} \quad \forall j \in \{1,2,\dots,M\} \quad \text{and} \quad \forall k \in \{1,\dots,K_j\}, \tag{6}$$

$$0 \le p_{ijk} \le 1, \quad \text{for} \quad \forall i,j,k, \tag{7}$$

$$\sum_{j=1}^{M} \sum_{k=1}^{K_j} p_{ijk} = 1, \quad \text{for} \quad \forall i \in \{1,2,\dots,N\}, \tag{8}$$

$$\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i < \mu_{jk}(f_{\max}), \quad \text{for} \quad \forall j,k, \tag{9}$$

$$y_{jk} \ge \frac{\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i}{\mu_{jk}(f_{\max})}, \quad \text{for} \quad \forall j,k, \tag{10}$$

$$x_j \ge \frac{\sum_{k=1}^{K_j} y_{jk}}{K_j}, \quad \text{for} \quad \forall j \in \{1,2,\dots,M\}, \tag{11}$$

where Price is the unit energy price during the current time period. Besides the above constraints, the constraints in the CTMDP formulation for each core should also be satisfied.

The details of the above formulation are as follows. The first term in the objective function (4) denotes the total revenue obtained from processing the service requests, and the second term corresponds to the total energy cost for operating the server cluster in consideration. Constraints (5)–(7) specify the ranges or domains of these variables. Constraint (8) ensures that all service requests generated by a client are processed by some certain core. Constraint (9) specifies the upper limit of the average service request arrival rate to a core, i.e. the arrival rate should be smaller than the maximum average service processing rate of that core when the core is running at the maximum execution frequency. Constraints (10) and (11) specify the ON cores and servers, respectively, based on the amounts of allocated resources.

The overall request dispatch, DVFS, and consolidation problem is integrated with a set of CTMDP optimisation subproblems (i.e. finding the optimal DVFS policy for each core based on the CTMDP model of the core.). The overall problem is then a mixed integer non-linear programming problem, which cannot be solved using conventional convex optimisation methods since the objective function is neither convex nor concave even if the optimal values of Boolean variables x_j's and y_{jk}'s are given. Moreover, the number of optimisation variables, especially that of p_{ijk}'s, grows superlinearly with the increase of M and N, which will significantly increase the solution complexity. This is another important issue that should be addressed by the proposed optimisation algorithm.

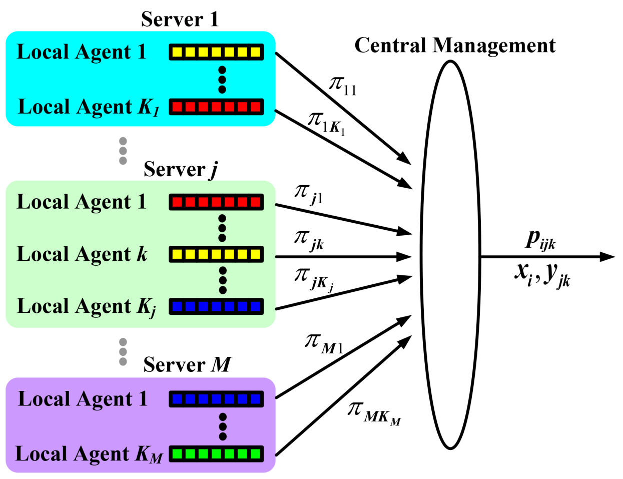

Fig. 3 *Flow chart of the centralised resource management algorithm with a set of distributed local agents*

4 Optimisation methods

As stated before, the joint optimisation problem of request dispatch, DVFS, and consolidation presented in the previous section is a hard problem due to the non-convexity of the objective function, the existence of Boolean variables, and the inclusion of a set of CTMDP optimisation subproblems within the original problem formulation. In addition, the superlinear increase in the number of optimisation variables with the increase in M and N is another important concern. Simple problem solvers cannot solve this problem except for the case of very small input sizes by executing exhaustive search or by using heuristic optimisation methods such as simulated annealing or genetic algorithm. In this section, we present a near-optimal solution for the joint optimisation problem with low computational complexity, properly addressing all the above-mentioned issues.

If we use a linear-form decreasing utility function in the optimisation problem, i.e. $U_i(R) = \beta_i - \alpha_i \cdot R$, which is similar to work in [16], then the objective function (4) can be transformed into the objective function (12) as shown in the following:

$$T \cdot \sum_{i=1}^{N} \lambda_i \cdot \left(\beta_i - \alpha_i \cdot \sum_{j=1}^{M} \sum_{k=1}^{K_j} p_{ijk} \cdot \bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})\right)$$
$$-\text{Price} \cdot T \cdot \tag{12}$$
$$\times \sum_{j=1}^{M} \left(x_j \cdot \bar{P}^{s}_{id,j} + \sum_{k=1}^{K_j} y_{jk} \cdot \left(\bar{P}^{c}_{st,jk} + \bar{P}^{c}_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})\right)\right).$$

In our proposed two-tier near-optimal solution, we use a distributed decision-making process instead of a centralised one, due to the relatively high complexity of a centralised solution. By using distributed decision making, we can increase the parallelism of the solution and thereby reducing the computation time significantly with only limited amount of communication efforts. In other words, it significantly increases the scalability of the solution. The distributed decision making is based on the following observation.

Observation 1: When the values of p_{ijk}'s, x_j's, and y_{jk}'s are given, maximising the objective function (12) in a centralised way is equivalent to minimising the following objective function at each kth core in the jth server with $x_j = 1$ and $y_{jk} = 1$

$$\sum_{i=1}^{N} \left\{\alpha_i \cdot \lambda_i \cdot p_{ijk} \cdot \bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk}) + \text{Price} \cdot \bar{P}^{c}_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})\right\}. \tag{13}$$

Based on the above observation, we propose a near-optimal solution for the resource allocation problem, comprised of a centralised resource management algorithm and a set of distributed local agents. The flow chart of the proposed algorithm is shown in Fig. 3. In this algorithm, each local agent solves the CTMDP optimisation subproblem (i.e. the DVFS problem) for the corresponding core, to find the optimal policy π_{jk}. The optimal π_{jk} should achieve a desirable tradeoff between the average response time and power consumption. Furthermore, we effectively use lookup tables to reduce the online computation overhead of each local agent. The central management algorithm, on the other hand, solves the request dispatch problem and finds the optimal p_{ijk} values, which determines the most appropriate server(s) and core(s) for request dispatch. In order to reduce the computation overhead, the central management algorithm is a two-tier hierarchical solution, in which the first control tier associates each client with a selected set of servers, i.e. service requests generated from the client can only be dispatched to cores of the chosen set of servers. In the second control tier of the central management algorithm, the request dispatch problem is finally solved based on the results of the first tier. In this way, the computation complexity of the central management algorithm can be significantly reduced through reducing the number of optimisation variables, with negligible performance degradation.

Finally, the central management algorithm also performs server-level and core-level consolidations, i.e. finding the optimal values of x_j's and y_{jk}'s, in the outer loop of the algorithm in order to achieve a desirable balance between the static and dynamic power components in the server cluster. We elaborate the details of the near-optimal solution in the following three subsections.

4.1 Local agents

We consider the local agent of the kth core in the jth server ($x_j = 1$ and $y_{jk} = 1$) given the p_{ijk} values. We denote this local agent as the (j, k)-agent. The local agent finds the optimal policy in order to minimise the objective function (13). We properly set the cost rate function in the CTMDP such that the objective function (13) is minimised when we solve the CTMDP subproblem. Based on the Little's theorem [31], we have the following theorem.

Theorem 1: Suppose that the cost function in the CTMDP is given by $c(s, a) = w_1 \cdot |s| + w_2 \cdot P^{c}_{dy,jk}(a)$ when the system is in state $s \in \boldsymbol{S}$ and we choose an action (i.e. execution frequency level) $a \in \boldsymbol{A}$, where $|s|$ is the number of service requests waiting or being processed in the SQ, and w_1, w_2 are relative weights greater than or equal to zero. We minimise $w_1 \cdot \left(\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i\right) \cdot \bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk}) + w_2 \cdot \bar{P}^{c}_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ when we solve the CTMDP subproblem.

Proof: We consider the CTMDP operation in the time interval from time 0 to T. By finding the optimal policy, we are essentially minimising the integral of the cost function $c(s, a)$ in this time interval with respect to time t. Suppose that T is a large enough value. Then according to the Little's law [31], the integral of $|s|$ equals $T \cdot \left(\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i\right) \cdot \bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$, where $\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i$ is the average request arrival rate of the core of interest. On the other hand, the integral of $P^{c}_{dy,jk}(a)$ equals to $T \cdot \bar{P}^{c}_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$, where $\bar{P}^{c}_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ is the average dynamic power consumption of the core. Combining these two observations, we have proved Theorem 1. □

Based on Theorem 1, we set the cost rate function in the CTMDP optimisation to be

$$c(s, a) = \frac{\sum_{i=1}^{N} \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i} \cdot |s| + \text{Price} \cdot P^{c}_{dy,jk}(a). \tag{14}$$

By employing this cost rate function, we minimise the objective function (13) when we solve the CTMDP optimisation subproblem, which will result in an optimal tradeoff between the average response time and power consumption.

The CTMDP optimisation subproblem is formulated as a linear programming problem as follows:

$$\min_{\{f_s(a)\}} \left(\sum_s \sum_a f_s(a) \cdot \gamma_s(a) \right), \quad (15)$$

where $f_s(a)$ denotes the frequency that the system enters state s and action a is chosen in that state. $\gamma_s(a)$ is the expected cost when the system is in state s and action a is chosen. $\gamma_s(a)$ is calculated as

$$\gamma_s(a) = \tau_s(a) \cdot c(s, a), \quad (16)$$

where $\tau_s(a) = 1/\sum_{s' \neq s} \sigma_{s,s'}(a)$ is the expected duration of time when the system will stay in state s when action a is chosen, and $\sigma_{s,s'}(a)$ is the average transition rate from state s to state s' when action a is chosen, as defined in (1).

The linear programming problem is solved for variables $f_s(a)$'s while satisfying the following constraints:

$$\sum_a f_s(a) = \sum_{s' \neq s} \sum_{a'} f_{s'}(a') \cdot p_{s',s}(a'), \quad \text{for} \quad \forall s, \quad (17)$$

$$\sum_s \sum_a f_s(a) \cdot \tau_s(a) = 1, \quad (18)$$

$$f_s(a) \geq 0, \quad \text{for} \quad \forall s \in \boldsymbol{S}, \quad (19)$$

where $p_{s,s'}(a)$ denotes the probability that the system will next come to state s' if it is currently in state s and action a is chosen. Constraints (17)–(19) capture the properties of a CTMDP. More specifically, constraint (17) addresses the balance condition for the system in the steady state. Constraint (18) normalises the sum of the steady-state probabilities in all states. Constraint (19) limits each frequency value to be non-negative. Next we make an important observation about the CTMDP optimisation subproblem.

Observation II: The CTMDP optimisation subproblem for the (j, k)-agent is determined (and fixed) with given average request arrival rate $\sum_{i=1}^N p_{ijk} \cdot \lambda_i$ and parameters

$$\frac{\sum_{i=1}^N \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^N p_{ijk} \cdot \lambda_i}$$

and Price in the cost rate function (in fact, the ratio between

$$\frac{\sum_{i=1}^N \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^N p_{ijk} \cdot \lambda_i}$$

and Price will be sufficient).

Please note that the objective function and the constraints of the CTMDP optimisation subproblem are all linear functions of the optimisation variables $f_s(a)$'s. Hence, we utilise standard linear programming solver such as the MOSEK [35] to solve the CTMDP optimisation subproblem. We find the optimal policy π_{jk} subsequently after deriving the $f_s(a)$ values, using the method described in [25].

In order to facilitate the further optimisations in the central resource manager, we define an equivalent M/M/1 queue for the CTMDP of the kth core in the jth server with policy π_{jk}. The average service request processing rate of the equivalent M/M/1 queue is given by

$$\mu_{eq,jk} = \sum_{i=1}^N p_{ijk} \cdot \lambda_i + \frac{1}{\bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})}, \quad (20)$$

which implies that the equivalent M/M/1 queue has the same average service request response time $\bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ as the CTMDP when the average service request arrival rate is given by $\sum_{i=1}^N p_{ijk} \cdot \lambda_i$. Similarly, we assume that the dynamic power consumption of the core is $P^c_{eq,dy,jk}$ when processing service requests. Please note that the kth core in the jth server may not be able to execute exactly with average request processing rate of $\mu_{eq,jk}$, because it can only execute at a discrete set of frequencies. In this case, we can derive the $P^c_{eq,dy,jk}$ value through intrapolation.

From Observation II, we know that the equivalent values $\mu_{eq,jk}$ and $P^c_{eq,dy,jk}$ only depend on the average request arrival rate $\sum_{i=1}^N p_{ijk} \cdot \lambda_i$ and the ratio between

$$\frac{\sum_{i=1}^N \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^N p_{ijk} \cdot \lambda_i} \text{ and Price}.$$

In order to reduce the online computation overhead, we propose to build a lookup table for each (j, k)-agent in an offline manner. The input variables of the lookup table are the average request arrival rate $\sum_{i=1}^N p_{ijk} \cdot \lambda_i$ and the ratio between

$$\frac{\sum_{i=1}^N \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^N p_{ijk} \cdot \lambda_i} \text{ and Price}.$$

The entries stored in the lookup table are the corresponding (optimised) $\mu_{eq,jk}$ and $P^c_{eq,dy,jk}$ values. In this way, each (j, k)-agent only needs to calculate

$$\sum_{i=1}^N p_{ijk} \cdot \lambda_i \quad \text{and} \quad \frac{\sum_{i=1}^N \alpha_i \cdot p_{ijk} \cdot \lambda_i}{\sum_{i=1}^N p_{ijk} \cdot \lambda_i}/\text{Price}$$

online and index the lookup table to instantaneously determine the $\mu_{eq,jk}$ and $P^c_{eq,dy,jk}$ values.

4.2 Optimal service request dispatch

In the optimal request dispatch problem to be solved by the central resource manager, we are given the values of x_j's, y_{jk}'s as well as the optimal policies π_{jk}'s for all the servers and cores from local agents. The objective is to find the optimal p_{ijk} values so as to maximise the objective function (12).

The main difficulty in this problem is that both $\bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ and $\bar{P}^c_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ are implicit functions of the optimisation variables p_{ijk}'s, as shown in Section 2.2. Hence, first we approximate $\bar{R}_{jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ and $\bar{P}^c_{dy,jk}(\pi_{jk}, \boldsymbol{p}_{jk})$ by using explicit functions of p_{ijk}'s. More specifically, we use the equivalent M/M/1 queue defined in Section 4.1 to approximate the CTMDP for each core. The approximate average request response time is $1/\left(\mu_{eq,jk} - \sum_{i=1}^N p_{ijk} \cdot \lambda_i\right)$ for each kth core in the jth server. The approximate average percentage of time this core is active (i.e. it has one or more service requests waiting or being processed) is $\sum_{i=1}^N p_{ijk} \cdot \lambda_i/\mu_{eq,jk}$ according to the M/M/1 queueing principles [31]. Hence, the approximate average dynamic power consumption is given by

$$\frac{\sum_{i=1}^N p_{ijk} \cdot \lambda_i}{\mu_{eq,jk}} \cdot P^c_{eq,dy,jk}. \quad (21)$$

Then the optimal request dispatch problem becomes as follows:

Find the optimal p_{ijk} values.

Minimise

$$\sum_{j=1}^M \sum_{k=1}^{K_j} \frac{\sum_{i=1}^N \alpha_i \cdot \lambda_i \cdot p_{ijk}}{\mu_{eq,jk} - \sum_{i=1}^N \lambda_i \cdot p_{ijk}} + \text{Price} \cdot \sum_{j=1}^M \sum_{k=1}^{K_j} \left(\frac{\sum_{i=1}^N p_{ijk} \cdot \lambda_i}{\mu_{eq,jk}} \cdot P^c_{eq,dy,jk} \right) \quad (22)$$

Subject to the following constraints:

```
**Input** $x_j$ and $y_{jk}$ values.
**Output** the $p_{ijk}$ values.
**Initialize** $p_{ijk}$ randomly and normalize.
**Do** (iteratively run the following procedure):
    The central resource manager issues commands to all the local agents with $x_j = 1$ and $y_{jk} = 1$ to start to perform CTMDP optimization.
    **For each** $(j,k)$-agent with $x_j = 1$ and $y_{jk} = 1$ (all the local agents act in parallel):
        Perform CTMDP optimization and find the optimal policy $\pi_{jk}$ (or simply the $\mu_{eq,jk}$ and $P^c_{eq,dy,jk}$ values) based on the $p_{ijk}$ values.
        Send optimization results back to the central resource manager.
    **End**
    The central resource manager performs the two-tier hierarchical request dispatching optimization and finds the optimal $p_{ijk}$ values based on the optimal policies $\pi_{jk}$'s (or the $\mu_{eq,jk}$ and $P^c_{eq,dy,jk}$ values) from local agents.
**Until** the solution converges.
```

Fig. 4 *Algorithm 1: the near-optimal solution with given ON servers and cores*

$$0 \le p_{ijk} \le 1, \quad \text{for} \quad \forall i, j, k, \tag{23}$$

$$p_{ijk} = 0, \quad \text{if} \quad y_{jk} = 0, \tag{24}$$

$$\sum_{j=1}^{M} \sum_{k=1}^{K_j} p_{ijk} = 1, \quad \text{for} \quad \forall i \in \{1, 2, \ldots, N\}, \tag{25}$$

$$\sum_{i=1}^{N} p_{ijk} \cdot \lambda_i < \mu_{eq,jk}, \quad \text{for} \quad \forall j, k. \tag{26}$$

The optimal service request dispatch problem maximises the sum of a set of linear fractional functions of the optimisation variables p_{ijk}'s. Effective algorithms exist in the literature [36] for finding a near-optimal solution of this kind of problem effectively. The fractional programming (FP) algorithm [36] can be applied to solve this optimisation problem.

However, the complexity in solving the above service request dispatch problem increases superlinearly with N, M, and K_j, which makes the solution not scalable. In order to address this issue, we propose a two-tier hierarchical solution for the request dispatch problem, in which the first tier associates each client with a selected set of servers, i.e. service requests generated from the client can only be dispatched to cores of the chosen set of servers. In the second tier of the central management algorithm, the request dispatch problem is finally solved, i.e. the p_{ijk} values are finally found, based on the results of the first tier. In this way, the computation complexity of the central management algorithm can be significantly reduced through reducing the number of optimisation variables, with negligible performance degradation. We discuss these two tiers in the following.

4.2.1 First tier: In the first tier, the central manager associates each client with a selected set of servers. Service requests that are generated from this client can only be dispatched to the set of servers in the subsequent tier. We define $p_{ij}^{\mathrm{Cl}\to\mathrm{S}}$ as the probability that the service requests generated from client i are dispatched to an ON server j (with $x_j = 1$), where the superscript 'Cl → S' denotes 'client to server'. In this tier the central manager derives the optimal $p_{ij}^{\mathrm{Cl}\to\mathrm{S}}$ values.

In order to limit the number of optimisation variables, we assume a straightforward mechanism to dispatch the arriving requests of each server j to its ON cores. Suppose that the kth core of server j is turned ON. Then a request that arrives at server j is dispatched to core k with probability $p_{jk}^{\mathrm{S}\to\mathrm{Co}}$, given by

$$\begin{aligned} p_{jk}^{\mathrm{S}\to\mathrm{Co}} &= \frac{\mu_{\mathrm{eq},jk}}{\sum_{k'=1}^{K_j} \mu_{\mathrm{eq},jk'} \cdot \boldsymbol{I}[y_{jk'} = 1]} \\ &= \frac{\mu_{\mathrm{eq},jk}}{\sum_{k'=1}^{K_j} \mu_{\mathrm{eq},jk'} \cdot y_{jk'}} \end{aligned} \tag{27}$$

where $\boldsymbol{I}[y_{jk'} = 1]$ is the indicator function which equals to 1 if the Boolean variable $y_{jk'} = 1$ is true (i.e. the k'th core of server j is turned ON.). The probability $p_{jk}^{\mathrm{S}\to\mathrm{Co}}$ is proportional to the equivalent request processing rate $\mu_{\mathrm{eq},jk}$ of the kth core and is independent of the optimisation variables $p_{ij}^{\mathrm{Cl}\to\mathrm{S}}$'s in the first tier. Then the optimisation problem in the first tier is formulated as follows.

Find the optimal $p_{ij}^{\mathrm{Cl}\to\mathrm{S}}$ values (please note that the number of optimisation variables has been significantly reduced compared with the original problem.).

Minimise

$$\begin{aligned} &\sum_{j=1}^{M} \sum_{k=1}^{K_j} \frac{\sum_{i=1}^{N} \alpha_i \cdot \lambda_i \cdot p_{ij}^{\mathrm{Cl}\to\mathrm{S}} \cdot p_{jk}^{\mathrm{S}\to\mathrm{Co}}}{\mu_{\mathrm{eq},jk} - \sum_{i=1}^{N} \lambda_i \cdot p_{ij}^{\mathrm{Cl}\to\mathrm{S}} \cdot p_{jk}^{\mathrm{S}\to\mathrm{Co}}} \\ &+ \mathrm{Price} \cdot \sum_{j=1}^{M} \sum_{k=1}^{K_j} \left(\frac{\sum_{i=1}^{N} p_{ij}^{\mathrm{Cl}\to\mathrm{S}} \cdot p_{jk}^{\mathrm{S}\to\mathrm{Co}} \cdot \lambda_i}{\mu_{eq,jk}} \cdot P^c_{\mathrm{eq},dy,jk} \right) \end{aligned} \tag{28}$$

Subject to the following constraints:

$$0 \le p_{ij}^{\mathrm{Cl}\to\mathrm{S}} \le 1, \quad \text{for} \quad \forall i, j, \tag{29}$$

$$p_{ij}^{\mathrm{Cl}\to\mathrm{S}} = 0, \quad \text{if} \quad x_j = 0, \tag{30}$$

$$\sum_{j=1}^{M} p_{ij}^{\mathrm{Cl}\to\mathrm{S}} = 1, \quad \text{for} \quad \forall i \in \{1, 2, \ldots, N\}, \tag{31}$$

$$\sum_{i=1}^{N} p_{ij}^{\mathrm{Cl}\to\mathrm{S}} \cdot \lambda_i < \sum_{k'=1}^{K_j} \mu_{\mathrm{eq},jk'} \cdot \boldsymbol{I}[y_{jk'} = 1], \quad \text{for} \quad \forall j. \tag{32}$$

Similar to the optimal request dispatch problem, in the first tier the central manager also maximises the sum of a set of linear fractional functions of optimisation variables $p_{ij}^{\mathrm{Cl}\to\mathrm{S}}$'s. We exploit the FP algorithm from [36] for solving this optimisation problem.

4.2.2 Second tier: In the second tier, the central manager solves the request dispatch problem, i.e. it finds the optimal values of p_{ijk}'s. The central manager minimises the objective function (22) subject to the constraints (23)–(26). In this tier, the central manager only focuses on the p_{ijk} values with $p_{ij}^{\mathrm{Cl}\to\mathrm{S}} > 0$, i.e. it sets an additional constraint that $p_{ijk} = 0$ if $p_{ij}^{\mathrm{Cl}\to\mathrm{S}} = 0$. In this way, the number of optimisation variables is significantly reduced compared with the overall request dispatch problem.

We integrate the request dispatch optimisation performed by the central manager with the local agents that solve the CTMDP subproblems, and thereby, we derive an iterative distributed algorithm with the ON servers and cores (i.e. with the given x_j and y_{jk} values.). Algorithm 1 (see Fig. 4) shows the pseudo-code of the proposed near-optimal solution.

```
Initialization: Turn on all servers and cores.
Do the following procedure:
    Run Algorithm 1 on the set of ON servers and cores and
    calculate the total profit.
    Turn off a subset of $L$ cores or a server with the
    minimum average service request arrival rate(s), and set
    the corresponding $x_j$ and $y_{jk}$ values to zero.
Until the total profit of the server cluster stops rising.
```

Fig. 5 *Algorithm 2: distributed near-optimal solution of the whole resource allocation and consolidation problem*

```
Initialization: Turn on all servers and cores.
Do the following procedure:
    Run Algorithm 1 on the set of ON servers and cores and
    calculate the total profit.
    Log the total profit of the server cluster under the current
    consolidation level.
    Turn off a subset of $L$ cores or a server with the
    minimum average service request arrival rate(s), and set
    the corresponding $x_j$ and $y_{jk}$ values to zero.
Until no solution can be found when running Algorithm 1
(because the number of ON servers and cores is not enough
for request dispatch and resource allocation.)
Find the optimal solution (in terms of profit) throughout the
whole consolidation process.
```

Fig. 6 *Algorithm 3: a more advanced distributed near-optimal solution of the whole resource allocation and consolidation problem*

4.3 Core- and server-level consolidations

In this section, we perform core-level and server-level resource consolidation to further improve energy efficiency of the overall server cluster. We perform the optimal service request dispatching and DVFS of the local agents as the inner loop [i.e. Algorithm 1 (Fig. 4) as the inner loop], while we determine the optimal set of ON servers and cores in an outer loop. We start from turning ON all the servers and cores and then selectively turn off a subset of cores or a server in one execution of the outer loop. The outer loop will terminate if no more gain in the total profit of the server cluster can be achieved. In this way, we will reach a near-optimal tradeoff between the dynamic and static power consumptions through effective consolidation methods. Algorithm 2 (see Fig. 5) provides the pseudo-code of the proposed procedure with the combination of optimal service request dispatching and DVFS [i.e. Algorithm 1 (Fig. 4) as the inner loop]. We also propose a more advanced algorithm for core- and server-level consolidations motivated by the Kernighan–Lin heuristic [37], with details shown in Algorithm 3 (see Fig. 6).

5 Experimental results

In this section, we perform several experiments with Matlab (fmincon function is utilised for solving convex optimisation) to test the proposed whole framework including request dispatching, DVFS, and resource consolidation, and compare with some baseline algorithms to investigate the effectiveness of the proposed framework.

The first experiment is performed for a server cluster with 15 heterogeneous servers and each having a uniformly distributed number of cores between 4 and 6. There are ten clients considered in the system. For the rest of parameters, we use normalised values instead of real values for the system simulation. We first set the average service request generating rate of each client as a uniformly distributed random number between 3 and 5 (This may be changed for later experiments.). We set the minimal average service request processing rate of each core (when running at its minimal execution frequency) as a uniformly distributed number between 1.0 and 2.5. We assume each core perform at five different execution frequencies, which are $1\times$, $1.25\times$, $1.5\times$, $1.75\times$, and $2\times$ of its minimal execution frequency, respectively. These five levels of execution frequencies correspond to five different actions in the CTMDP-based DVFS model. According to the server core's execution frequency setup, the different average service request processing rates of each core are therefore $1\times$, $1.25\times$, $1.5\times$, $1.75\times$, and $2\times$ of its minimum average service request processing rate, respectively.

We assume the (instantaneous) dynamic power consumption of each core is proportional to the square of its execution frequency (or equivalently, proportional to the square of its average service request processing rate.). On the other hand, we assume the (average) static power consumption of each core is half as the dynamic power consumption when that core is running at its minimum execution frequency level. The maximum queue length Q is set to be 20. For the utility functions, we assume each α_i value is a uniformly distributed random number between 1 and 1.5, and each β_i value is set as 10. We change the unit energy price Price in the experiments and derive different results on the total profit in the server cluster.

There is no existing work that addresses both the optimal request dispatching to the server cluster, optimal DVFS control of each core, and core-level and server-level resource consolidations. Therefore, we use three baselines that have no CTMDP-based DVFS control ability for the individual cores. Specifically, in baseline 1, we set each of the cores to run at its maximum possible execution frequency such that its average service request response time is minimised. For baseline 2, we set each of the cores to run at its minimum possible execution frequency such that its dynamic power consumption is minimised. We will make sure the service request processing rate of the core is still higher than the average service request arrival rate. In baseline 3, we set the cores to run at their medium execution frequencies, which are $1.5\times$ of the minimum frequencies, to achieve a tradeoff between response time and power consumption. For the request dispatching, we assume all the three baseline systems distribute the service requests from the clients to all the cores in the server cluster with equal probabilities. In addition, we add baseline 4 that also distributes service requests to all the cores in the system with equal probabilities and performs the CTMDP-based DVFS control for individual cores. Besides, baseline 5 utilises a near-optimal solution of the joint optimisation problem based on the Hungarian algorithm for the assignment problem, as well as convex optimisation techniques, in a way that is similar to the constructive partitioning algorithm in very large scale integration computer-aided design [17]. The near-optimal solution consists of two steps. The first step performs server and core assignment based on Hungarian algorithm and convex optimisation techniques. The second step implements effective service request dispatching and resource allocation, based on the results of the first step.

In Fig. 7, we show the normalised total profit versus the unit energy price resulting from the proposed system and the five baseline systems. First, it is obvious that our proposed system consistently outperforms the five baseline systems. Specifically, if we check the normalised profit at the unit energy price of 1.8, we can observe 148.6, 55.0, 73.3, 30.5 and 12.4% higher value from the proposed system than the baseline systems 1–5, respectively. Also observable from Fig. 7 is that baseline 1 has better performance when the unit energy price is lower. This is because baseline 1 minimises the cost penalty by running every core at its maximum execution frequency and when the unit energy price is low the energy cost is much lower than the cost penalty. Differently, baseline 2 has better performance when the unit energy price is high, because in this case the energy cost becomes the dominating factor. The performance difference between the proposed algorithm and baseline 4 is due to the optimal service request dispatch method in the proposed algorithm. Baseline 5 has lower total profit than the proposed method due to the applications of multiple near optimal methods in multiple phases in two steps.

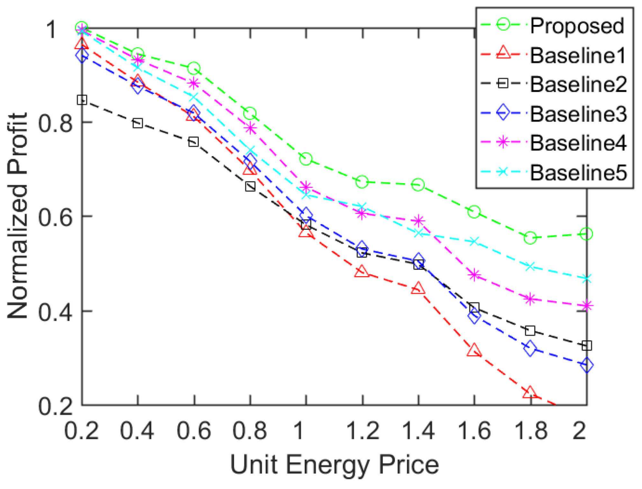

Fig. 7 *Normalised total profit versus the unit energy price of the proposed near-optimal algorithm and five baseline algorithms when there are 15 heterogeneous servers and 10 clients in the system*

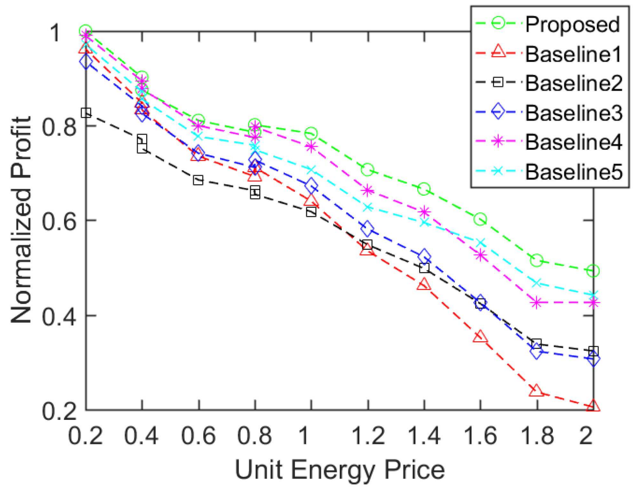

Fig. 8 *Normalised total profit versus the unit energy price of the proposed near-optimal algorithm and five baseline algorithms when there are 20 heterogeneous servers and 15 clients in the system*

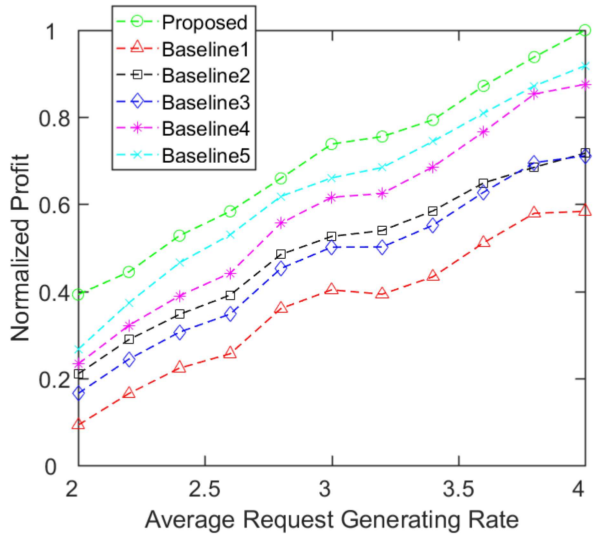

Fig. 9 *Normalised total profit versus the average service request generating rate of the proposed near-optimal algorithm and five baseline algorithms*

We also note that as the unit energy price increases, the near optimal method of baseline 5 begins to outperform baseline 4 at a certain point. The reason is that the near optimal service request dispatch of baseline 5 can achieve quite efficient energy allocation as the energy cost keeps increasing.

In the second experiment, we consider a larger-scale test case with 20 heterogeneous servers and 15 clients in the system. Fig. 8 illustrates the normalised total profit versus the unit energy price of the proposed near-optimal algorithm and five baseline algorithms. We can observe from Fig. 8 that the proposed near-optimal algorithm consistently outperforms the five baseline algorithms. When the unit energy price is 2.0, the total profit obtained by the proposed algorithm is 140.0, 52.2, 60.6, 15.6, and 11.6% higher than baseline 1, baseline 2, baseline 3, baseline 4, and baseline 5 respectively. The analysis for the first experiments also hold here as the explanation of results from the second experiments, demonstrating the effectiveness and consistency of our proposed algorithm.

Next, we investigate the effect of the average service request generating rate of the clients on the normalised total profit among the proposed system and the baseline systems. For this experiment, we use a fixed unit energy price Price of 1.6 while changing the average service request generating rate of the clients. This is for investigating the effectiveness of performing server-level and core-level consolidations. Fig. 9 illustrates the normalised total profit versus the average service request generating rate of the clients on the proposed system and five baseline systems. The proposed framework consistently outperforms the five baseline systems as observed in Fig. 9. When the average service request generating rate is 3.0, the total profit obtained by the proposed algorithm is 83.1, 40.3, 47.2, 19.8 and 11.8% higher than baseline 1, baseline 2, baseline 3, baseline 4, and baseline 5, respectively. On the other hand, when the average service request generating rate is 4.0, the total profit obtained by the proposed algorithm is 71.3, 39.4, 40.6, 14.2 and 8.9% higher than baseline 1, baseline 2, baseline 3, baseline 4, and baseline 5, respectively. Both the proposed framework and baseline 5 performs server-level and core-level consolidations and assignments, thus achieving higher total profit due to reduction on energy consumption. Since baseline 5 utilises multiple near-optimal methods in multiple phases, baseline 5 suffers from some total profit degradation compared with the proposed framework. We find that the optimal number of ON cores will increase if the average service request generating rate is larger. The optimal number of ON cores is 41 when the average request generating rate is 2.0, and is 62 when the average generating rate is 4.0. The reason is that fewer cores are required for request processing when the average service request generating rate is lower and the static power consumption in the server cluster is reduced in this case.

6 Conclusion

In this paper, we consider the SLA-based resource allocation optimisation problem in the cloud computing framework. The objective is to maximise the total profit, which is the total revenue obtained from serving the clients subtracted by the energy cost of the server cluster. The total revenue depends on the average request response time for each client as defined in its utility function. Each core in the server cluster is modelled by a CTMDP. We propose a joint optimisation framework accounting for request dispatch, DVFS for individual cores in the server cluster, as well as core-level and server-level consolidations. The near-optimal solution is comprised of a central manager and distributed local agents. Each local agent employs linear programming-based CTMDP solving method to solve the DVFS problem for the corresponding core. On the other hand, the central manager solves the request dispatch problem and finds the optimal number of ON cores and servers for request processing. To reduce the computational overhead, the central manager is realised as a two-tier hierarchical controller. Experimental results demonstrate that the proposed near-optimal resource allocation and consolidation algorithm consistently outperforms the baseline algorithms.

7 Acknowledgments

This work was sponsored in part by a grant from the Software and Hardware Foundations of the National Science Foundation, and NSF awards CNS-1704662 and CCF-1733701.

8 References

[1] Hayes, B.: 'Cloud computing', *Commun. ACM*, 2008, **51**, (7), pp. 9–11

[2] Mell, P.: 'The NIST definition of cloud computing', 2011, NIST Special Publication 800-145

[3] Wang, B., Qi, Z., Ma, R., *et al.*: 'A survey on data center networking for cloud computing', *Comput. Netw.*, 2015, **91**, pp. 528–547

[4] Buyya, R.: 'Market-oriented cloud computing: vision, hype, and reality of delivering computing as the 5th utility'. 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid), 2009

[5] Pedram, M.: 'Energy-efficient datacenters', *IEEE Trans. Comput.-Aided Des.*, 2012, **31**, (10), pp. 1465–1484

[6] Armbrusk, M., Fox, A., Griffith, R., *et al.*: 'A view of cloud computing', *Commun. ACM*, 2010, **53**, (4), pp. 50–58

[7] Wei, G., Vasilakos, A.V., Zheng, Y., *et al.*: 'A game-theoretic method of fair resource allocation for cloud computing services', *J. Supercomput.*, 2010, **54**, (2), pp. 252–269

[8] Mashayekhy, L., Nejad, M.M., Grosu, D., *et al.*: 'An online mechanism for resource allocation and pricing in clouds', *IEEE Trans. Comput.*, 2016, **65**, (4), pp. 1172–1184

[9] Wang, L., Zhang, F., Vasilakos, A.V., *et al.*: 'Joint virtual machine assignment and traffic engineering for green data center networks', *ACM SIGMETRICS Perf. Eval. Rev.*, 2014, **41**, (3), pp. 107–112

[10] Chen, K., Hu, C., Zhang, X., *et al.*: 'Survey on routing in data centers: insights and future directions', *IEEE Netw.*, 2011, **25**, (4), pp. 6–10

[11] Barroso, L.A., Holzle, U.: 'The case for energy-proportional computing', *IEEE Comput.*, 2007, **40**, (12), pp. 33–37

[12] Katz, R.H.: 'Tech titans building boom', *IEEE Spectr.*, 2009, **46**, (2), pp. 40–54

[13] Polverini, M., Cianfrani, A., Ren, S., *et al.*: 'Thermal-aware scheduling of batch jobs in geographically distributed data centers', *IEEE Trans. Cloud Comput.*, 2014, **2**, (1), pp. 71–84

[14] Wang, L., Zhang, F., Aroca, J.A., *et al.*: 'GreenDCN: a general framework for achieving energy efficiency in data center networks', *IEEE J. Sel. Areas Commun.*, 2014, **32**, (1), pp. 4–15

[15] Wang, Y., Chen, S., Goudarzi, H., *et al.*: 'Resource allocation and consolidation in a multi-core server cluster using a Markov decision process model'. Proc. of Int. Symp. on Quality Electronic Design (ISQED), 2013

[16] Goudarzi, H., Pedram, M.: 'Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems'. Proc. of IEEE Int. Conf. on Cloud Computing (CLOUD), 2011

[17] Wang, Y., Chen, S., Pedram, M.: 'Service level agreement-based joint application environment assignment and resource allocation in cloud computing systems'. Proc. of IEEE Green Technologies Conf. (GreenTech), 2013

[18] Wei, G., Vasilakos, A.V., Zheng, Y., *et al.*: 'A game-theoretic method for fair resource allocation for cloud computing services', *J. Supercomput.*, 2010, **54**, (2), pp. 252–269

[19] Liu, Z., Squillante, M.S., Wolf, J.L.: 'On maximizing service-level-agreement profits'. 3rd ACM Conf. on Electronic Commerce (EC), 2001

[20] Zhang, L., Ardagna, D.: 'SLA based profit optimization in autonomic computing systems'. 2nd Int. Conf. on Service Oriented Computing, 2004

[21] Ardagna, D., Trubian, M., Zhang, L.: 'SLA based resource allocation policies in autonomic environments', *J. Parallel Distrib. Comput.*, 2007, **67**, (3), pp. 259–270

[22] Buyya, R., Murshed, M.: 'Gridsim: a toolkit for the modelling and simulation of distributed resource management and scheduling for grid computing', *Concurrency Comput. Pract. Exp.*, 2002, **14**, (13), pp. 1175–1220

[23] Krauter, K., Buyya, R., Maheswaran, M.: 'A taxonomy and survey of grid resource management systems for distributed computing', *Softw. Practice Exp.*, 2002, **32**, (2), pp. 135–164

[24] Chandra, A., Gongt, W., Shenoy, P.: 'Dynamic resource allocation for shared clusters using online measurements'. *Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2003, **31**, (1), pp. 300–301

[25] Puterman, M.L.: '*Markov decision processes: discrete stochastic dynamic programming*' (Wiley Publisher, New York, 1994)

[26] Srikantaiah, S., Kansal, A., Zhao, F.: 'Energy aware consolidation for cloud computing'. Workshop on Power Aware Computing and Systems (HotPower' 08), 2008

[27] Hwang, I., Kam, T., Pedram, M.: 'A study of the effectiveness of CPU consolidation in a virtualized multi-core server system'. Proc. of Int. Symp. on Low Power Electronics and Design (ISLPED), 2012

[28] Papoulis, A.: '*Probability, random variables, and stochastic processes*' (McGraw-Hill, 1991, 3rd edn.)

[29] Zhang, Z., Towsley, D., Kurose, J.: 'Statistical analysis of generalized processor sharing scheduling discipline'. ACM SIGCOMM'94 Conf. on Communications Architectures, Protocols and Applications

[30] Jung, H., Pedram, M.: 'Stochastic dynamic thermal management: a Markovian decision-based approach'. Int. Conf. on Computer Design (ICCD), 2006

[31] Kleinrock, L.: '*Queueing systems, volume I: theory*' (Wiley, New York, 1975)

[32] Burd, T., Pering, T., Stratakos, A., *et al.*: 'A dynamic voltage-scaled microprocessor system'. IEEE Int. Solid-State Circuits Conf., Digest of Technical Papers, 2000

[33] Dubois, M., Annavaram, M., Stenström, P.: '*Parallel computer organization and design*' (Cambridge University Press, MA, 2012)

[34] 'AMD Processor Power and Thermal Data Sheet'. Available at http://support.amd.com/us/Processor_TechDocs/40036.pdf, accessed 12 March 2015

[35] Andersen, E.D., Andersen, K.D.: 'The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm', in Frenk, H., Roos, K., Terlaky, T., Zhang, S. (Eds.): '*High Performance Optimization. Applied Optimization*', (Springer, Boston, MA, 2000), vol. **33**, pp. 197–232

[36] Chen, D.Z., Daescu, O., Dai, Y., *et al.*: 'Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications', *J. Comb. Optim.*, 2005, **9**, (1), pp. 69–90

[37] Kernighan, B.W., Lin, S.: 'An efficient heuristic procedure for partitioning graphs', *Bell Syst. Tech. J.*, 1970, **49**, (2), pp. 291–307