Songklanakarin J. Sci. Technol.
41 (4), 828-837, Jul. – Aug. 2019

http://www.sjst.psu.ac.th

Original Article

Cache-as-a-Service for client-side cloud caching: Models and system

Chaturong Sriwiroj[1] and Thepparit Banditwattanawong[2*]

[1] *School of Information Technology, Sripatum University, Chatuchak, Bangkok, 10900 Thailand*

[2] *Department of Computer Science, Faculty of Science, Kasetsart University, Chatuchak, Bangkok, 10900 Thailand*

Received: 17 August 2017; Revised: 27 December 2018; Accepted: 4 April 2018

Abstract

Presently, cloud computing is being used to store large amounts of data for sharing among users. However, it imposes cloud data-out charges and data access delays on organizations. These problems can be relieved by using cloud caching to prevent repetitive data loading from clouds. This paper proposes technical and economic models and a system for operating a client-side shared cloud cache to be turned into a Cache-as-a-Service (CaaS). The models and the system are our novel contributions. Evaluation by simulation using two experimental data sets, representing large-sized and small-to-medium-sized cloud data, demonstrated that the technical model achieved cost-saving ratios up to 56.20% and delay-saving ratios up to 56.65%, and that the economic model recommended a monthly service charge of CaaS to be 24,599.70 USD on average based on a large-sized cloud data set and 657.93 USD on average based on a small-to-medium-sized cloud data set. The CaaS system gained a good level of user satisfaction.

Keywords: cloud computing, Cache-as-a-Service, economic service model, technical service model

1. Introduction

Presently, much data is stored and shared in clouds and is continuously and rapidly increasing for distributed sharing among users via standard protocol HTTP. As a consequence, this could saturate network bandwidth and lead to data access latency (Banditwattanawong & Uthayopas, 2013). Besides, it may also cause expensive cloud data loading when accounting for the transferred volume and data-out charge rate. This problem can be relieved by adopting client-side cloud caching (Figure 1) which replicates data and stores it on the user side in order to prevent repeated data loading from clouds every time there is a request for the same data. Although the client-side cloud caching algorithm (Bandit wattanawong, Masdisornchote, & Uthayopas, 2016) has been well established, it lacks service models to serve as a real working cloud service as a Cache-as-a-Service (CaaS) model.

*Corresponding author
Email address: thepparit.b@ku.th

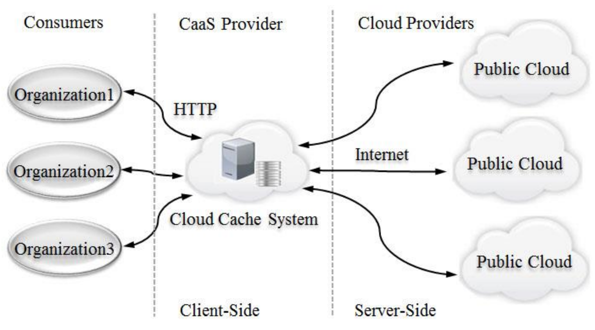

Figure 1. Deployment of client-side CaaS.

Research studies are available in the literature on the technical models for cloud cache such as SC2 (Chockler *et al.*, 2011), high performance model, and best value model (Han *et al.*, 2012). On the other hand, research has been reported that is related to economic service models for a cloud cache, such as an economic model for self-tuning (Dash, Kantere, & Ailamaki, 2009), a billing model (Chockler *et al.*, 2011), a dynamic pricing model (Kantere, Dash, Francois, Kyriako poulou, & Ailamaki, 2011), and a pricing model (Han *et al.*, 2012). However, all of these technical and economic service models belong to a server-side rather than client-side.

A few research studies were published on client-side cloud caching such as dynamic block-level caching that uses SSD on virtual host to improve virtual machine performance (Arteaga, Otstott, & Zhao, 2012), client-side hybrid caching file system (CFS) uses file-level disk caching and block-level memory caching to improve performance of random file access in cloud computing (Cao, Huang, Lei, Zhang, & Huang, 2012), client-side cloud cache replacement policy, namely Cloud, aims for optimal byte-hit ratio and a cost-savings ratio and improves delay-saving ratio (Bandit wattanawong & Uthayopas, 2014), and an intelligent cloud cache eviction approach, namely i-Cloud, that is capable of reducing public cloud data-out expenses, improving cloud network scalability, and lowering cloud service access latencies (Banditwattanawong *et al.*, 2016). However, these client-side cloud caching studies still lack technical and economic models to serve as a real-working client-side CaaS.

2. Proposed CaaS Models and System

Our proposed CaaS service models consist of a technical model and an economic model together with a proposed CaaS system which realizes the technical and economic models.

2.1 Technical model

Our proposed technical model represents a set of caching performance and privacy options for different consumer requirements. The model relies on two-dimensional elements: cached-data storage technologies, which can be random access memory [RAM], solid-state drive [SSD] or hard disk drive [HDD], and cached-data sharing requirements that are either shared cache space or isolated cache space. The isolated cache space is defined as a single cache space used to store data objects requested by a single consumer group. The shared cache space is defined as a single cache space used to store data objects requested by multiple consumer groups. Thus, a combination of these two dimensions yields six technical service options (Figure 2).

- Option 1: RAM isolated cache space is suitable for organizations who require very fast data access (based on RAM speed) and gives priority to data security and privacy.
- Option 2: RAM shared cache space is suitable for organizations who require very fast data access and have affiliation with other organizations able to share cached data without security and privacy concerns such as an enterprise with multiple branches can share cached data across the branches to economize service charge based on cost sharing.
- Option 3: SSD isolated cache space is a good choice for obtaining data access speed and data security and privacy with a moderate budget.
- Option 4: SSD shared cache space is for fast data access and multiple organizations able to share the same cached data.
- Option 5: HDD isolated cache space is appropriate for organizations who require data security and privacy with a limited budget.

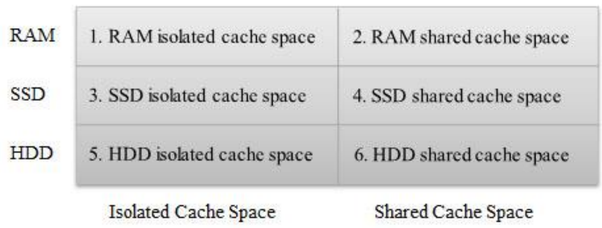

Figure 2. Proposed technical service model.

- Option 6: HDD shared cache space allows normal data access with a limited budget and full cached-data sharing.

2.2 Economic model

The proposed economic model serves as the service pricing scheme for the technical service options for CaaS providers. The model has been derived from two underlying platforms on which a CaaS system is deployed: colocation service and cloud Infrastructure as a Service (IaaS). Colocation-based and IaaS-based services incur different capital expenditures (CapEx), operating expenditures (OpEx) and probably different desired profit margins. The CapEx are the expenses incurred to initiate the business or service system to acquire or upgrade fixed long-term assets for operation. The OpEx are the expenses incurred in the course of ordinary business such as general and administrative expenses. OpEx are comprised of two parts: fixed cost (OpExf), which is a periodic constant cost, and variable cost (OpExv), which is a periodic cost that varies with operated service output. We have modeled the CapEx and the OpEx based on the total cost of ownership (Amazon Web Services [AWS], 2014). By taking into account these costing factors, a reasonable service charge has been derived in Equation 1.

$$PR = AC+PF \tag{1}$$

where PR is CaaS subscription pricing per month, AC is the total actual monthly cost, and PF is the desired monthly profit. Since AC is the summation of both CapEx and OpEx on a monthly basis, it is necessary to clarify the CapEx and OpEx of different CaaS deployment platforms which leads to two different economic options of the CaaS.

2.2.1 Option 1. CaaS running on colocation service

Based on Equation 1, the monthly price of CaaS is calculated from the summation of CapEx, OpEx, and desired profit. Specifically, CapEx comes from asset depreciation, while OpEx includes only amortized fixed cost and variable cost. Depreciation can be calculated using Equation 2.

$$D_d = ((C_d-R_d)/p_d)/12 \tag{2}$$

where D_d is the depreciation per annum of fixed asset d (which can be infrastructure, hardware, and software), C_d is the cost of fixed asset d, R_d is an estimated remaining value at the end of asset d's lifetime, p_d is the lifetime (years) of fixed asset d, and P is the maximum projected number of consumer sites that can be supported by the fixed assets. We can now derive CapEx per month for all assets involving a CaaS system by using Equation 3.

$$CapEx = \sum D_d/P \quad (3)$$

As a consequence, the first part of CaaS price derived from CapEx, i.e. the AC and PF of CapEx in Equation 1, is showed in Equation 4 where pm is the desired profit in percentage.

$$PRCapEx = CapEx+CapEx\times pm/100 \quad (4)$$

With respect to the amortized fixed cost, a basic formula for amortization is Equation 5 where A_{am} is the installment amount per month, P_{am} is an initial principal per month, e_{am} is an interest rate per month, q_{am} is the total number of installments, and $_{am}$ can be salary, colocation cost, hardware maintenance cost, and software maintenance cost.

$$A_{am} = (P_{am}\times e_{am}\times(1+e_{am})^{q_{am}})/((1+e_{am})^{q_{am}}-1) \quad (5)$$

Therefore, the amortized fixed cost (OpExf) per month per consumer site is Equation 6.

$$OpExf = \sum A_{am}/P \quad (6)$$

As a consequence, the second part of the CaaS price derived from fixed OpEx, i.e. the AC and PF of fixed OpEx in Equation 1, is shown in Equation 7.

$$PROpExf = OpExf+OpExf\times pm/100 \quad (7)$$

Additionally, as the third part, CaaS price must include variable OpEx, which is divided into two parts: a variable OpEx in case of cache miss and a variable OpEx in case of cache hit. For a data object o_i, t_{store} is a timestamp o_i is stored into a cache, t_{evict} is a timestamp o_i is evicted from a cache, Δt is elapsed time equal to $t_{evict} - t_{store}$, To_i is the transfer cost of o_i from a public cloud server, $S_c o_i(\Delta t)$ is the cost of using cache space per Δt for o_i, c is the member of a set {RAM,SSD,HDD}, p can be Isolated cache space or Shared cache space, $M_p o_i$ is the electricity cost of processing the cache miss of o_i, and Oo_i is the transfer-out cost of o_i from cache to an end user. Thus, the total cost of o_i's cache miss ($Cmiss_{c,p}o_i$) is given in Equation 8.

$$Cmiss_{c,p}o_i = To_i+S_c o_i(\Delta t)+M_p o_i+Oo_i \quad (8)$$

As a consequence, the part of CaaS price derived from variable OpEx in the case of cache miss (i.e., the AC and PF of variable OpEx in Equation 1 in the case of cache miss) is shown in Equation 9.

$$PRmiss_{c,p}o_i = Cmiss_{c,p}o_i+Cmiss_{c,p}o_i\times pm/100 \quad (9)$$

On the other hand, the total cost of a cache hit is simply Equation 10.

$$Chit_{c,p}o_i = M_p o_i+Oo_i \quad (10)$$

Subsequently, the part of CaaS price derived from variable OpEx in the case of cache hit, i.e. the AC and PF of variable OpEx in Equation 1 in the case of cache hit, is shown in Equation 11.

$$PRhit_{c,p}o_i = Chit_{c,p}o_i+Chit_{c,p}o_i\times pm/100 \quad (11)$$

The part of CaaS price derived from total OpEx is shown in Equation 12.

$$PROpExv = \left(\sum_{l=1}^{m} PRmiss_{c,p}o_i + \sum_{j=1}^{h} PRhit_{c,p}o_i\right)/P \quad (12)$$

where m is the total number of cache misses (detected in a CaaS system) per month and h is the total number of cache hits (detected in a CaaS system) per month. Finally, based on Equation 1, the monthly price of CaaS deployed by means of colocation can be derived from Equation 13.

$$PR = PRCapEx+PROpExf+PROpExv \quad (13)$$

2.2.2 Option 2. CaaS running on infrastructure as a service

In this CaaS economic option, CapEx is composed of merely software license costs while OpExf consists of the amortized costs of salaries and IaaS subscription fee. The OpExv is composed of the costs of cache misses and hits. The variable costs of cache misses and hits can be calculated with Equations 8 and 10, respectively, and they can be summed up in Equation 14.

$$OpExv = \left(\sum_{l=1}^{m} Cmiss_{c,p}o_i + \sum_{j=1}^{h} Chit_{c,p}o_i\right)/P \quad (14)$$

Therefore, the monthly price of CaaS deployed over IaaS can be derived from Equation 15.

$$PR = [(CapEx+OpExf+OpExv)]+ [((CapEx+OpExf+OpExv)\times pm)/100] \quad (15)$$

2.3 CaaS system

This section describes the analysis, design and implementation of our CaaS system, which operates according to the proposed models.

2.3.1 System analysis and design

This subsection explains the transformation of CaaS system requirements into a blueprint. We defined user requirement specification based on not only the technical and economic models but also a service supply chain according to Josyula, Orr, and Page (2011) (Figure 3). The service supply chain consists of 6 steps starting from consumer subscription to CaaSes from a catalogue until they gain provisioned services matching their needs. All of the service usage and change requests are logged and can be monitored by a CaaS provider.

Once we had a clear blueprint set, we employed the Unified Modeling Language (UML) and Entity-Relationship diagrams to conduct the analysis and design. Figure 4 presents a user case diagram.

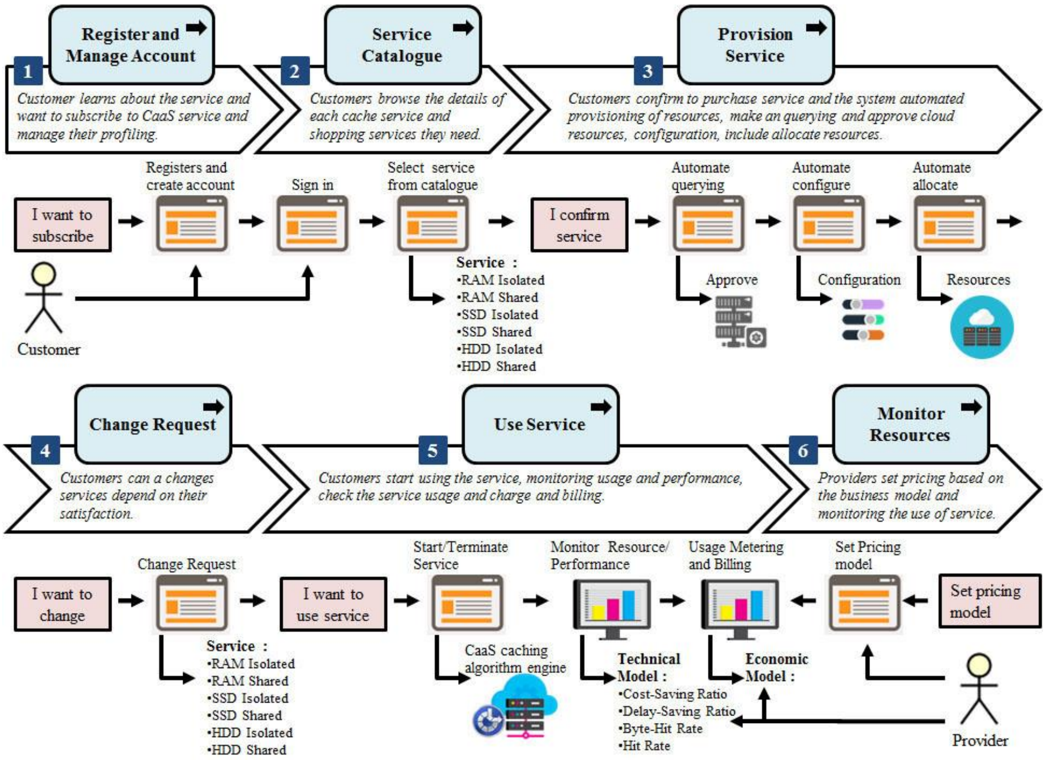

Figure 3. CaaS supply chain.

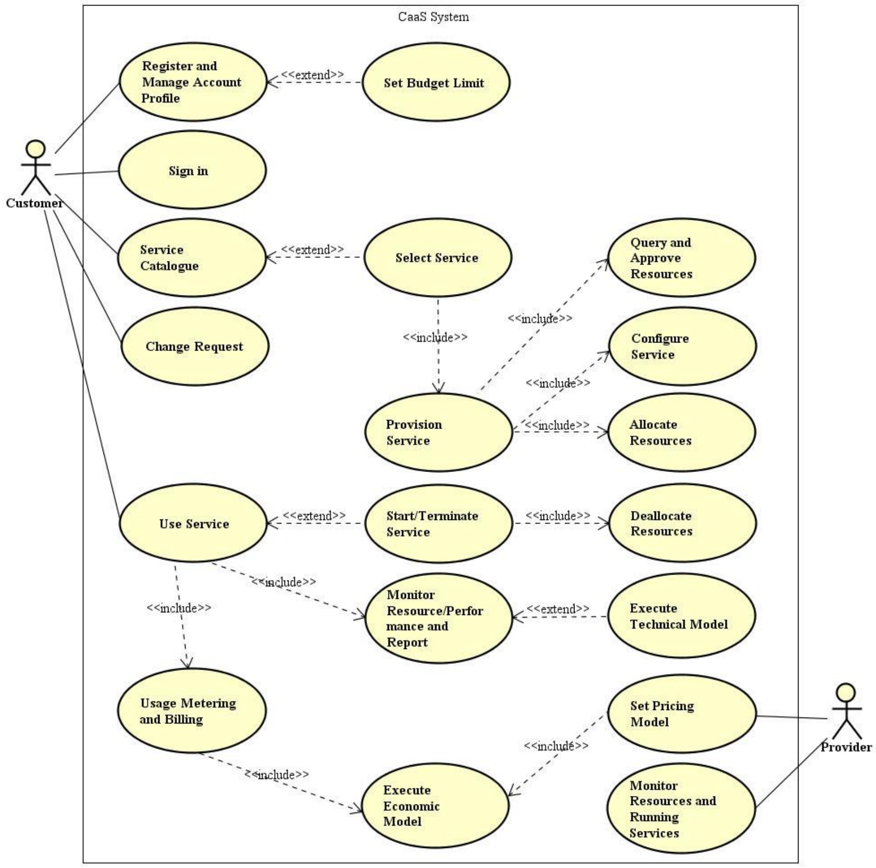

Figure 4 User case diagram for our CaaS system.

We also came up with the architecture of the IaaS-based CaaS system depicted in Figure 5.

The architecture is comprised of virtual machines, each of which acts as an independent caching server. One of the virtual machines is responsible for web-based CaaS management, and according to subscribing customer volume, the virtual machines can scale out or in.

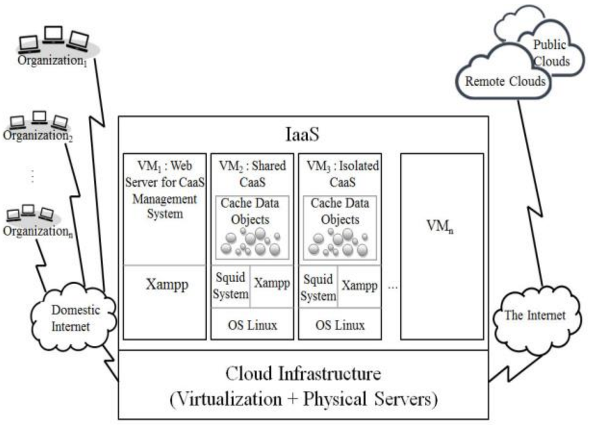

Figure 5. CaaS architecture based on IaaS.

3. Evaluation

Evaluation of both proposed models was conducted by means of trace-driven simulation. As for the CaaS system evaluation, we engaged thorough user-acceptance testing using questionnaires.

3.1 Simulation data sets

The traces were produced from a single pre-processed HTTP trace obtained from IRCache (National Laboratory for Applied Network Research [NLANR], 2016) collected from users in Boulder, Colorado, USA for 31 days. The preprocessing cleaned trace to remain only the records of requests to top 50 domains (to emulate the total number of cloud-hosted intranet services (Banditwattanawong *et al.*, 2016)) and adjusted object sizes within the trace proportional to the original sizes the scenarios separately. The pre-processing produced two distinct traces.

The preprocessing trace representing the large-sized object scenario, generated a data set with total requested object size equal to realistic organization at bandwidth consumption through 10 Gbps Metro Ethernet with 50% average downstream bandwidth utilization for 8 work hours a day. Thus, the total amount of cloud data-out transfer was 4,570.31 TB per year (260 workdays per year) or 380.86 TB per month. This size of total transferred objects is divided across the new object sizes in each preprocessed trace. The objects requested in the new trace include the files listed in Table 1.

The preprocessing trace representing the small-to-medium object scenario generated the other data set based on the original HTTP trace by adjusting the original object sizes to produce requests to the files listed in Table 2. Thus, the total size of objects requested from the cloud was 6,697.69 GB per month.

Table 1. Characteristics of the large-sized object data set.

Object types	Object size (bytes)	No. of objects	Utilized cache space (bytes)
High-definition large videos	53,687,091,200	617	6,979,321,856,000
Animations	21,474,836,480	1,702	11,617,886,535,680
Virtual machine image files	16,106,127,360	10,632	71,301,825,822,720
Standard-definition videos	5,046,586,572.8	35,237	119,518,309,803,622

Table 2. Characteristics of the small-to-medium-sized object data set.

Object types	Object size (bytes)	No. of objects	Utilized cache space (bytes)
Standard-definition large videos	5,046,586,572.8	522	2,634,318,191,106
Audios	314,572,800	8,552	2,690,226,585,600
High-definition image raw images	26,214,400	51,857	1,359,400,140,800
High-definition images	7,340,032	53,592	393,366,994,944
Miscellaneous cloud data	1,048,576	108,985	114,279,055,360

3.2 Performance metrics

To evaluate the technical model, we measured its realized performance using the following four metrics.

Cost-saving ratio is the difference between the total data downloading cost without caching and with caching divided by the total data downloading cost without caching. The metric value is calculated using Equation 16.

$$[\sum_{i=1}^{n} c_i s_i r_i - [\sum_{i=1}^{l} c_k s_i h_i + \sum_{i=1}^{o} (c_i + c_k) s_i m_i]] \,/\, [\sum_{i=1}^{n} c_i s_i r_i] \tag{16}$$

where n is the number of total requests, c_i is data object i loading charge from a cloud provider to a CaaS system per object size, s_i is the size of object i, r_i is a request to object i, l is the total number of cache hits, c_k is a data loading charge from CaaS to a consumer premise per object size, k is a CaaS option (RAM, SSD, or HDD), h_i is the hit rate of object i, m_i is the miss rate of object i, and o is the total number of cache misses.

Delay-saving ratio refers to the difference between the total data downloading latency without caching and with caching divided by the total data downloading latency without caching. The metric value can be calculated using Equation 17.

$$[\sum_{i=1}^{n} d_i r_i - [\sum_{i=1}^{l} e_i h_i + \sum_{i=1}^{o} (d_i + e_i) m_i]] / [\sum_{i=1}^{n} d_i r_i] \quad (17)$$

where d_i is data object i loading delay from a cloud provider to a CaaS system and e_i is data object i loading delay from CaaS to a consumer premise.

Hit rate is the total number of hits per total number of requests (Podlipnig & Böszörmenyi, 2003) as shown in Equation 18.

$$\sum_{i=1}^{n} h_i \;/\; \sum_{i=1}^{n} r_i \quad (18)$$

Byte hit rate is the total size of hit objects per total size of requested objects (Podlipnig & Böszörmenyi, 2003) as in Equation 19.

$$\sum_{i=1}^{n} s_i h_i \;/\; \sum_{i=1}^{n} s_i r_i \quad (19)$$

3.3 Simulation tools

We adapted the proven preprocessor and simulator of (Banditwattanawong *et al.*, 2016) using Java SE Runtime Environment version 1.8.0_20-b26 and NetBeans version 8.0.2 Patch 2. All simulation sessions were conducted on a 64-bit Windows 7 laptop computer using an Intel CPU i7-3667 U 2.00 GHz, 6 GB RAM, and 1 TB hard drive.

3.4 Simulated scenarios

Four scenarios of combination between the CaaS deployment scenarios and the data consumption scenarios were used in our simulation.

- Scenario I: CaaS was deployed based on colo-cation and served a large-sized object data set.
- Scenario II: CaaS was deployed based on IaaS and served a large-sized object data set.
- Scenario III: CaaS was deployed based on colo-cation and served a small-to-medium-sized object data set.
- Scenario IV: CaaS was deployed based on IaaS and served a small-to-medium-sized object data set.

Furthermore, every scenario was simulated against a combination of practical factors shown in Figure 6 that consisted of two data-out charge types (non-uniform costs exist when using more than one cloud provider with different data-out charge rates), two service types, three CaaS options, one cache size, and the pricing policies applied to only the IaaS deployment scenario: cost sharing and profit sharing. Therefore, there were a total of 72 simulation sessions among the four scenarios.

3.5. System implementation

We mainly engaged Java, PHP, Yii2 framework (Makarov, 2013; Safronov & Winesett, 2014), XAMPP (Apa che, 2015), and modified Squid open source software (squid-cache.org, 2015). The modification of Squid was to include i-Cloud.

4. Results and Discussion

We report the simulation results based on the four scenarios. Since the economic model sets the appropriate prices, the simulation results reflect monthly CaaS sub-scription cost when serving entire certain data sets. The simulation results of the technical model are reported in the 4 performance metrics.

- Economic model based on Scenario I: Figure 7a illustrates that, if the consumer of a single cloud utilizes CaaS (RAM option and isolated cache space), a CaaS provider should bill 31,731.16 USD per month for the service usage in order to gain the desired profit. Figure 7b shows that, if the consumer of multi-provider clouds utilizes CaaS (SSD cache space sharing option), a CaaS provider should bill 21,405.69 USD per month for the service usage to gain the desired margin.
- Economic model based on Scenario II: Figure 7c demonstrates the costs on a profit-sharing basis for a consumer organization using a single cloud. Similarly, Figure 7d shows costs on a profit-sharing basis for a consumer using multiple cloud providers. Figure 7e illustrates CaaS usage costs based on the cost-sharing policy when a consumer organization utilizes a single cloud. Figure 7f shows costs on a cost-sharing basis for a consumer using multiple cloud providers.
- Economic model based on Scenario III: Figure 7g illustrates the appropriate costs for a consumer using a single cloud. Figure 7h shows the costs for a consumer using multiple cloud providers.

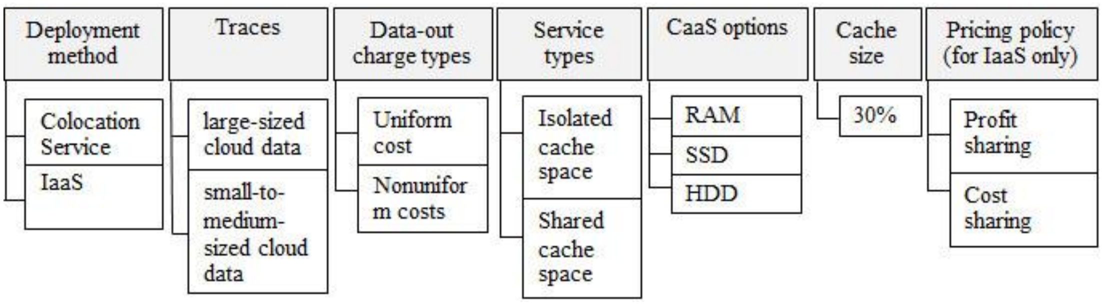

Figure 6. Practical factors to build the 72 simulation sessions.

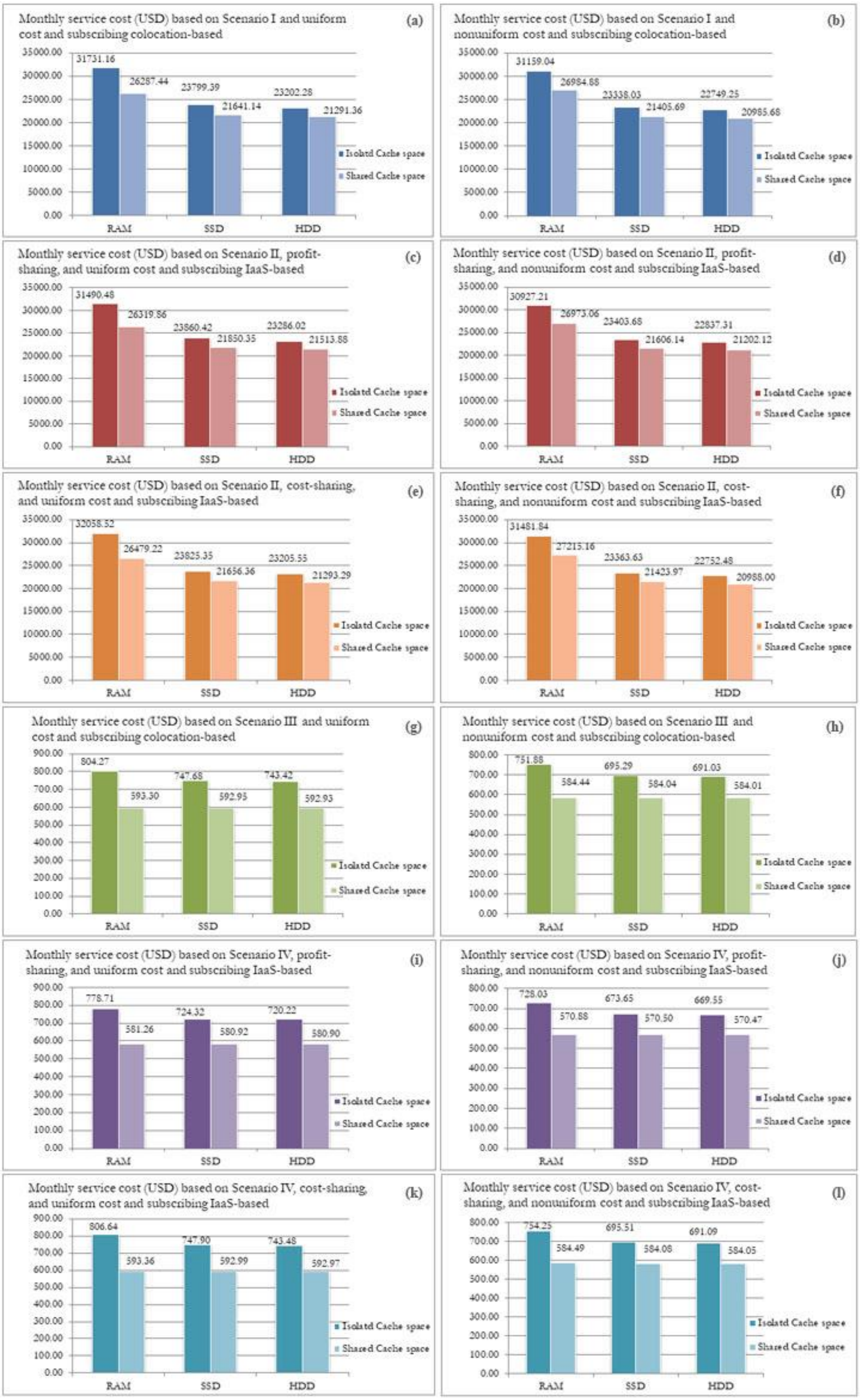

Figure 7. Monthly service subscription costs based on the scenarios.

● Economic model based on Scenario IV: Based on a profit-sharing policy, Figure 7i illustrates the appropriate CaaS subscription costs for a consumer using a single cloud. Similarly, Figure 7j shows the costs for a consumer using multiple cloud providers. On the other hand, with a cost-sharing policy, Figure 7k illustrates costs for a consumer using a single cloud. Similarly, Figure 7l shows costs for a consumer using multiple cloud providers.

● Technical model based on Scenario I: The best performance in terms of a cost saving ratio was found in the HDD option and the shared cache space as shown in Figure 8a, while the best performance in terms of delay saving ratio

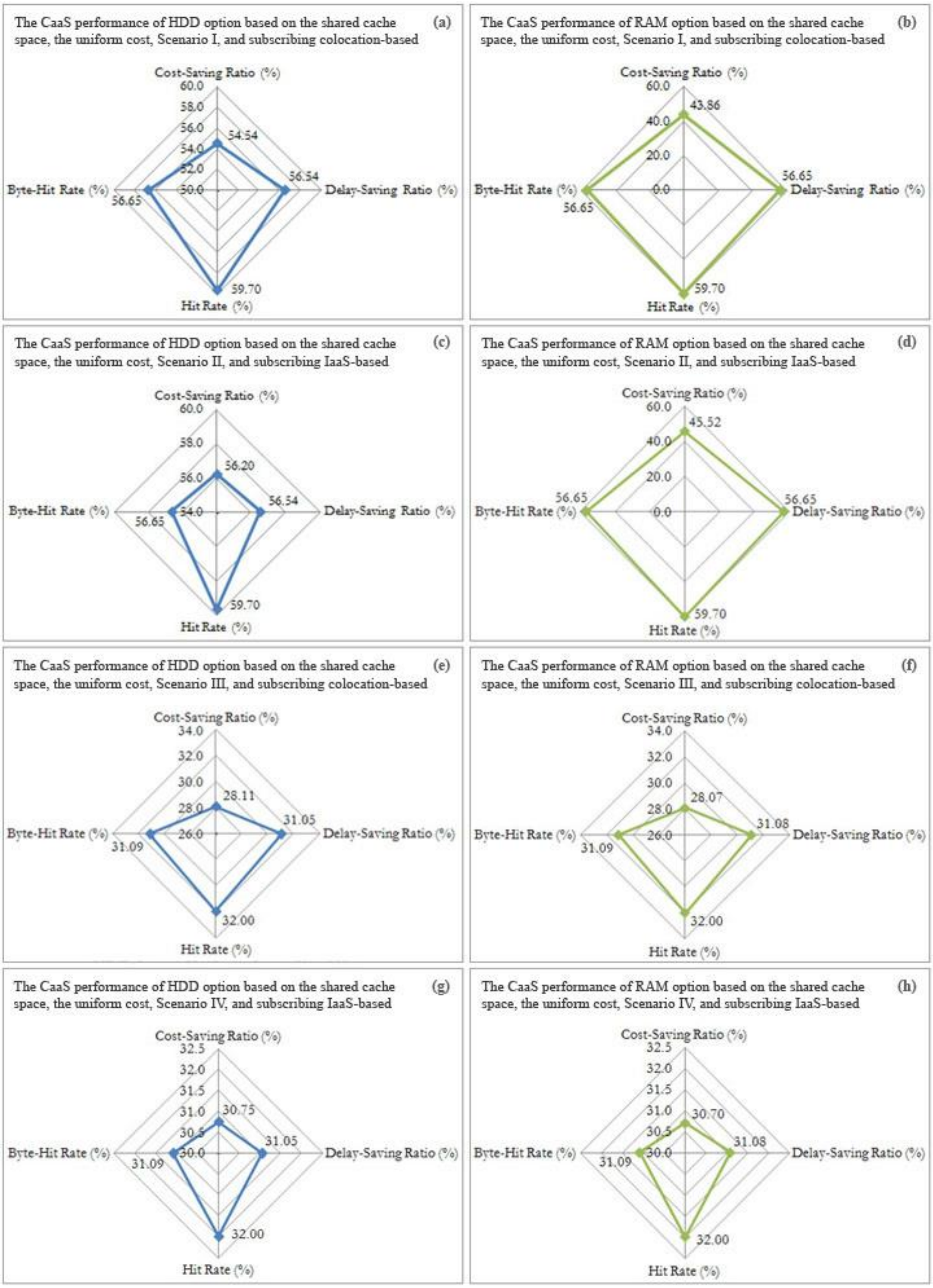

Figure 8. CaaS performance based on the scenarios.

were found in the RAM option and the shared cache space as depicted in Figure 8b.

- Technical model based on Scenario II: The best cost saving performances were found in the HDD option and the shared cache space as shown in Figure 8c, while the best performances in terms of delay saving ratio were found in the RAM option and the shared cache space as depicted in Figure 8d.
- Technical model based on Scenario III: The best performances in terms of cost saving ratio were found in the HDD option and the shared cache space as shown in Figure 8e, while the best performances in terms of delay saving ratio were found in the RAM option and the shared cache space as depicted in Figure 8f.
- Technical model based on Scenario IV: The best cost saving performances were found in the HDD option and the shared cache space as shown in Figure 8g, while the best performances in terms of delay saving ratio were found in the RAM option and the shared cache space as depicted in Figure 8h.

The first finding to be discussed comes from Figure 8c. The HDD shared cache space option had the highest cost-saving ratio due to its lowest cost per storage space unit, whereas Figure 8f shows that RAM shared cache space option had the lowest cost-saving ratio because of its highest cost per space unit. For the delay-saving performance, Figure 8b and Figure 8d prove that RAM shared cache space options outperformed the other options because RAM has the lowest access latency, whereas Figure 8e and Figure 8g showed that HDD shared cache space options had the worst delay-saving ratios because of the slowest access times of HDD. By synthesizing Figure 8, an economic model that had the cheapest service price was the shared cache space options rather than the isolated cache space options because the costs were shared by multiple user organizations.

In terms of cost effectiveness, the option that provided the highest benefit per cost value (where the benefit was a percent cost-saving ratio and the cost was monthly service cost) was, of course, the HDD shared cache space service option (cost effectiveness = 0.05292) based on the uniform cost and profit-sharing policy. Furthermore, the highest benefit per cost value when the benefit was percent delay-saving ratio and the cost was monthly service cost was the SSD shared cache space service option (cost effectiveness = 0.05447) based on the non-uniform cost and profit-sharing policy. The reason that the SSD shared cache space service option had the best cost effectiveness was because SSD has a slightly longer access delay than RAM but much faster than HDD resulting in the delay saving performance of SSD closer to that of RAM, while RAM incurs a highest cost per storage space or the highest monthly service cost.

The mean Likert scale values for the CaaS system on the performance and security, content, functional process, and ease of use aspects were 4.29, 4.44, 4.44, and 4.39, respectively. The average score of all aspects was 4.39, which meant that our CaaS system performed well at a good level. The overall 4.39 score resulted from the application of a well-known spiral software life cycle model by conducting requirement gathering, analysis, design, coding, testing, and evaluation phases in an iterative manner (Pfleeger & Atlee, 2009).

5. Conclusions

This paper presented three novel contributions: an economic model, a technical model, and a system. The economic model consisted of two economic options: colocation based and IaaS based. The technical model was comprised of six service options: (1) RAM isolated cache space; (2) RAM shared cache space; (3) SSD isolated cache space; (4) SSD shared cache space; (5) HDD isolated cache space; and (6) HDD shared cache space. The system shows the analysis, design and implementation of client-side CaaS for the first time in the field. Evaluation of the economic model was conducted with 72 simulation sessions. It was found that the highest monthly service cost was 32,058.52 USD, which existed in the scenario where CaaS (RAM isolated cache space) was deployed based on IaaS and served the large-sized cloud data set using the cost sharing and the uniform cost, whereas the minimum monthly service cost was 778.71 USD of the scenario where CaaS (RAM isolated cache space) was deployed based on IaaS and served the small-to-medium-sized cloud data set using the profit sharing and the uniform cost. Evaluation of the technical model was also based on the 72 sessions. The best cost-saving ratio (56.20%), which existed in the scenario CaaS (HDD shared cache space), was deployed based on IaaS and served the large-sized cloud data set using the uniform cost. The best delay-saving ratio (56.65%) lied in two scenarios: a scenario CaaS (RAM shared cache space) was deployed based on colocation and served the large-sized cloud data set using the uniform cost and a scenario CaaS (RAM shared cache space) was deployed based on IaaS and served the large-sized cloud data set using the uniform cost. Finally, we evaluated the system development by means of user acceptance test, which achieved an average score of 4.39. Our future work includes 1) to perform an industrial test of the system before real CaaS deployment and 2) to invent another model effectiveness metric that is independent of US dollars, which was precise only at the time this paper was written, i.e. when the US dollar exchange rate changes, the data must be re-applied in our economic model.

References

Amazon Web Services. (2014). *Total Cost of Ownership (TCO) comparison*. Retrieved from https://s3-eu-west-1.amazonaws.com/donovapublic/TCOOutput.pdf

Arteaga, D., Otstott, D., & Zhao, M. (2012). Dynamic block-level cache management for cloud computing systems. *Proceeding of the Conference on File and Storage Technologies.*

Banditwattanawong, T., & Uthayopas, P. (2013). Improving cloud scalability, economy and responsiveness with client-side cloud cache. *Proceeding of the 10th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology,* 1-6. doi:10.1109/ECTICon.2013.6559553

Banditwattanawong, T., & Uthayopas, P. (2014). A client-side cloud cache replacement policy. *ECTI Transactions on Computer and Information Technology*, *8*(2), 152-160.

Banditwattanawong, T., Masdisornchote, M., & Uthayopas, P. (2016). Multi-provider cloud computing network infrastructure optimization. *Future Generation Computer Systems*, *55*, 116-128. doi:10.1016/j.future.2015.09.002

Cao, L., Huang, L., Lei, K., Zhang, Z., & Huang, L. E. (2012). Hybrid caching for cloud storage to support traditional application. *Proceeding of the IEEE Asia Pacific Cloud Computing Congress (APCloudCC)*, 11-15. doi:10.1109/APCloudCC.2012.6486503

Chockler, G., Laden, G., & Vigfusson, Y. (2011). Design and implementation of caching services in the cloud. *IBM Journal of Research and Development*, *55*(6), 9-1. doi:10.1147/JRD.2011.2171649

Dash, D., Kantere, V., & Ailamaki, A. (2009). An economic model for self-tuned cloud caching. *Proceeding of the 25th International Conference on Data Engineering, IEEE,* 1687-1693. doi:10.1109/ICDE.2009.143

Han, H., Lee, Y. C., Shin, W., Jung, H., Yeom, H. Y., & Zomaya, A. Y. (2012). Cashing in on the Cache in the Cloud. *IEEE Transactions on Parallel and Distributed Systems*, *23*(8), 1387-1399. doi:10.1109/TPDS.2011.297

Josyula, V., Orr, M., & Page, G. (2011). *Cloud computing: Automating the virtualized data center.* Indianapolis, IN: Cisco Press.

Kantere, V., Dash, D., Francois, G., Kyriakopoulou, S., & Ailamaki, A. (2011). Optimal service pricing for a cloud cache. *IEEE Transactions on Knowledge and Data Engineering*, *23*(9), 889-915. doi: 10.1109/TKDE.2011.35

Makarov, A. (2013). *Yii application development cookbook*. Birmingham, England: Packt Publishing.

NLANR (2016, January). National Laboratory for Applied Network Research. Retrieved from http://www.ircache.net/

Pfleeger, S. L., & Atlee, J. M. (2009). *Software engineering: Theory and practice* (4th ed.). Cambridge, England: Pearson Press.

Podlipnig, S., & Böszörmenyi, L. (2003). A survey of web cache replacement strategies. *ACM Computing Surveys*, *35*(4), 374-398. doi:10.1145/954339.954341

Safronov, M., & Winesett, J. (2014). *Web application development with Yii 2 and PHP*. Birmingham, England: Packt Publishing.

Squid-Cache (2015) [Computer software]. squid-cache.org.

XAMPP (2015) (Windows) [Computer software]. Apache Friends.