2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 979-8-3503-0436-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/PerComWorkshops59983.2024.10503179

Improving the Usefulness of Context Information for IoT Applications: A Middleware-based Approach

Shakthi Weerasinghe, Kanaka Sai Jagarlamudi, Arkady Zaslavsky, Seng W. Loke, Kevin Lee, Guang-Li Huang
School of Information Technology
Deakin University
Geelong, Australia
{syweerasinghe, k.jagarlamudi, seng.loke, kevin.lee, belinda.huang}@deakin.edu.au

***Abstract*— Smarter pervasive and ubiquitous applications are on the rise. They leverage contextual information to make actions more relevant and dependable to the users and the Internet of Things (IoT) ecosystem. The adoption of context is making large leaps in time- and safety-critical applications like autonomous driving. While the sources of context information are also rapidly increasing to support this trend, they fail to provide consistent quality, performance and vary largely in features, e.g., presentation and interfaces. In large IoT ecosystem, untangling these complexities and responding to each context request with useful context information is expensive and not scalable. We investigate and propose a novel mechanism to provide Context Management Systems (CMS) with (a) adaptive context caching for reusing and repurposing context information for fast concurrent access, and (b) cost- and quality-aware context provider selection to supplement caching with means to maximize effectiveness. Unlike any previous work, our feature-rich strategy is capable of concurrently optimizing a CMS for Quality of Context, Quality of Service, and cost efficiency, factors that dictate the usefulness of context. We demonstrate how our mechanism adapts to variations in the IoT ecosystem, context providers, and context query loads in near real-time while efficiently scaling. A large-scale autonomous vehicle management use case is tested using our prototype. Our mechanism outperforms existing benchmarks, i.e., up to 43% reduction in penalty costs and up to 99% reduction in processing and retrieval costs, while providing 27% fresher context.**

Keywords*— *Context Management Systems, Adaptive Context Caching, Near-Real Time Adaptation, Quality Aware-Selection

I. Introduction

Distributed and pervasive context-aware applications and systems relying on the Internet of Things (IoT) use diverse data to enhance user experience [1]. For example, Google Maps can suggest restaurants based on distance, traffic, and crowdsourced reviews. Context Management Systems [2] or CMSs are middleware that integrate third-party data providers (e.g., sensors) and provide diverse contextual information to applications. Once a CMS receives context requests, it selects and invokes relevant data providers, then uses the retrieved raw IoT data (also referred to as low-level context) to infer "useful" (i.e., relevant, and meaningful) high-level context information for the context consumer.

Unfortunately, context providers (i.e., sources of context) often struggle to meet the Quality of Context (QoC) standards, e.g., age, errors, incompleteness, and/or unreliability [2], [3], which determine the usefulness of context to consumers and selecting QoC adequate data streams from billions of sources is an integral part of completing context requests. Maintaining QoC for continuous monitoring and delivery of context requires a large computing capacity for storage, inferencing, and refreshing. But, Quality of Service (QoS) has a reciprocal effect on QoC given processing and network delays increase the age of context. Therefore, with the growing demand for context information, it is infeasible to effectively scale a CMS using existing techniques such as traditional data caching [4] without compromising QoC and performance efficiency.

In this paper, we propose a novel mechanism that involves a cooperating adaptive context caching mechanism and a cost- and quality-aware context provider selector to concurrently optimize a CMS for QoC, performance-efficiency, and cost-efficiency. The adaptive context caching mechanism aims to minimize costs by maximizing performance efficiency. The adaptive context provider selector, ConCQEng [18], focuses on maximizing QoC while minimizing the Cost of Context [8]. This improves the QoC of cached context by delivering high-quality context values to inferencing processes, which are cached for reuse and repurposing. ConCQEng is optimized based on feedback from context caching algorithms, with adaptive context caching (ACOCA) periodically assessing the impact of recent selections on QoC, performance-, and cost-efficiencies. The novel contributions of this paper are as follows:

1) proposes a mechanism to concurrently maximize Quality of Context, Quality of Service, and cost efficiency of a Context Management System (CMS) in near real-time,

2) shows the applicability of adaptive context caching and cost- and quality-aware context provider selection to handle the complexities involved with handling the enormity of the IoT ecosystem, and,

3) demonstrates the logical coherency [5] between different algorithms in a CMS, helping to solve the inherent complex multi-attribute and multi-objective problems.

The rest of the paper proceeds as follows. Section II introduces our motivating scenario. Section III discusses the related work in optimizing and scaling CMSs. Section IV introduces our proposed mechanism's modules, algorithms, and architecture. Section V discusses the results obtained from our prototype, comparing them against the benchmarks and Section VI concludes our discussion with further directions in the research area.

II. Motivating Scenario

Let us consider the autopilot-connected navigation system based on autonomous vehicles in Melbourne, Australia, that guides vehicles to "suitable" parking spots based on context information from various data sources (Fig.1). The suitability of context information depends on the context consumer's interests, such as arrival time at the destination, access road situation, competition for parking spots, weather conditions,

979-8-3503-0436-7/24/$31.00 ©2024 IEEE

and speed. The autopilot plans a route based on these suggestions but may adapt if there is road congestion or other factors that affect the vehicle's performance.

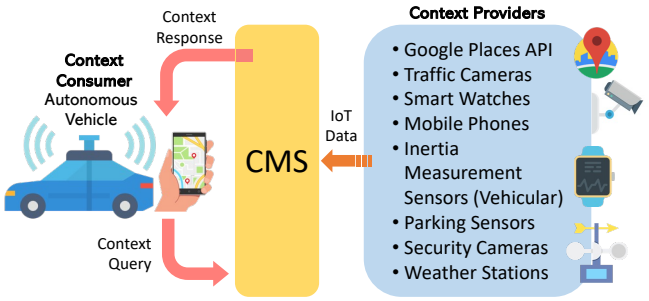


Fig.1. Suggesting parking spots for autonomous vehicles based on context about the environment and the passengers.

A. *Critical problems in the scenario*

The criticality of concurrently maximizing the quality of context, quality of service, and cost-efficiency using this scenario is three-fold (which are generalizable). Firstly, context information like situations is highly transient [6]. For example, the situation of an access road can immediately change from "light" to "heavy" traffic as a result of an accident such that a suggested parking spot is no longer recommendable. The problem is the lack of structure, potential errors, incompleteness, and unreliability of the Big IoT data. For example, assume there exists a lack of traffic camera coverage. The CMS would have to derive the road condition from selected indirect indicators that are unreliable, e.g., inferencing traffic level using roadside ambient noise level and smoke density.

Secondly, maintaining adequate Quality of Context (QoC) requires an expensive analytical pipeline, including selecting context providers, validating their selections, retrieving data, pre-processing, and inferring context, e.g., using Bayesian reasoning [1]. Yet, the context management processes need to adapt fast to transient situations. Assuming the CMS does not scale adequately to process context queries and notify all the autonomous vehicles about recently congested intersections in a timely manner (i.e., before merging with the congestion), the delivered context may become irrelevant. Thirdly, continuous monitoring and storage of data is hence very computationally expensive that perpetual scaling resources are cost-inefficient.

As it is evident from the above, it is not trivial to achieve high levels of *QoC* in a continuous near-real-time system that relies on context providers that fail to be consistent and are massively heterogeneous without compromising the cost- and performance-efficiency. Unlike any previous work in the literature, our novel mechanism addresses this real-world problem by delivering QoC adequate and cost-effective context in a timely manner.

III. Background

Context is "any information that can be used to characterize the situation of an entity, where an entity can be a person, place, an object that is considered relevant to the interaction" [7]. Useful context in a situation depends on meeting consumer QoC expectations. We also argued that the performance and cost-efficiencies of a CMS are related to QoC. Non-intrusive systems must be aware of all aspects of quality [8], making investigating mechanisms that maximize QoC, cost-, and performance-efficiency a pressing need.

Improvements in hardware (e.g., high-precision sensors) and network technologies (e.g., uRLLC) contribute to QoC, but their relevance and availability are uncertain due to hardware wear and heterogeneity in the IoT ecosystem [2]. So, relying on context providers for QoC is undesirable. Research on QoC-aware context provider selection focuses on provider features, reputation, and self-assessment results of cost- and performance-efficiency [9]. However, investigations on incorporating performance and cost characteristics of context providers for selection are necessary to make context information more accessible, e.g., trading-off precision for cost.

Our background analysis revealed that existing CMSs are limited in scalability due to outdated techniques (e.g., use of XML in Nexus [10] incurs significant processing and context sharing overheads), constraints in concurrent database access [11], and centralized architecture [12]. However, resource minimization [13] for cost is counterproductive to maximizing QoS and QoC, e.g., negative correlation between context refresh rate and QoC [14]. The literature lacks investigation into the trade-off between cost and other factors in each phase of context. Further, context caching is sparsely investigated, with authors developing techniques to maximize freshness [4], [14] and cache hit rate [15] but neglects the correlation of QoC with cost- and performance-efficiencies.

In summary, we understand that there is a significant gap in knowledge about, how to efficiently deliver high-quality context in a scalable manner, such that context is both, (a) "useful" – relevant and meaningful, and (b) easily accessible (i.e., cheap, and quickly accessible) to the consumer?

We intend to develop an answer to this research question in this paper. TABLE I summarize the notations and acronyms related to relevant concepts from previous work that are useful for further discussion.

TABLE I. Notations and Acronyms Related to Useful Concepts and Parameters Used in This Paper

Notation	Description
$Price_{res}$	Price per timely context response, e.g., \$0.50.
f_{thr}	Freshness threshold. $f_{thr} \in \mathbb{R} \cap [0,1]$.
RT_{max}	Maximum tolerated response latency for a CR.
$Prec$	Precision of a context value. $Prec \in \mathbb{R} \cap [0,1]$.
$Cost_{pen}$	Penalty per context response exceeding RT_{max}.

IV. Cost, Performance, and Quality-aware Context Management

This section outlines a two-component solution for optimising a CMS for QoC, performance, and cost. First, the Context Cost and Quality Engine (ConCQEng) selects cost-efficient context providers meeting QoC requirements. Second, the Adaptive Context Caching (ACOCA) mechanism reduces CMS reliance on ConCQEng for context and reuses the context [16] thereby improving the Quality of Service (e.g., reduced response time) and reducing costs. By cooperating, these solutions address the common challenge of achieving cost- and performance-efficiency while maintaining QoC. The following sub-sections will cover ConCQEng, the ACOCA mechanism, and their cooperative process.

A. ConCQEng

The architecture and the process of the ConCQEng are shown in Fig.2. The process, as described in [17], involves selecting context providers based on their estimated QoC and cost outcomes. It then measures [3] and validates [18] the quality and cost of context from these providers to ensure compliance with the specific requirements of the context request [17].

Quality and Cost of Context-Aware Selection. In Step 1, ConCQEng receives context requests (*CR*s) with QoC requirements and price constraints from the CMS. In Step 2, the Relative Reputation Processor (RR-processor) selects potential context providers based on precomputed *QoC* adequacy rate (Q_{CP_i}) and veracity (V_{CP_i}). They represent the likelihood a context provider meeting the *QoC* specifications, where$V_{CP_i}$is the trustworthiness of provider's QoC parameters based on which the Q_{CP_i} is measured. Selected providers are forwarded to the Assurance Processor. It selects and invokes the most cost-efficient providers among those selected by the RR-processor in Step 3.

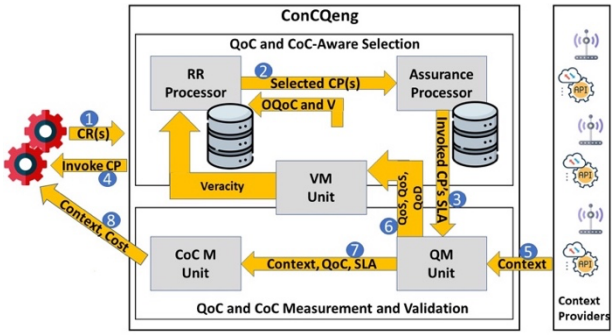


Fig.2. High-level architecture of ConCQEng.

Equations (1) and (2) are used to determine a provider's QoC-adequacy and CoC-effectiveness. Indexes $i \in \{1,2,\dots I\}$ denote the context providers that are compatible to complete a *CR*. Indexes $j \in \{1,2,\dots J\}$ denote different criteria for cost-efficiency. The overall *QoC* ($OQoC$) is the average of all *QoC* metrics in the provider response. *RR* is the average relative reputation of the context provider in producing context that satisfies the requesting context consumer. Relative reputation (*RR*) is the average reputation of a provider meeting the quality requirements based on consumer feedback. w_j represents the weight that criteria j has on cost-efficiency (e.g., context cost and penalties for QoC violation). Q_{ij} represents the value of criteria j related to CP_i. M_{ij} represents the normalised value of criteria j on CP_i computed using (3) depending on whether the criterion's ideal value is maximum or minimum for cost-efficiency. $\max Q_j$ and $\min Q_j$ signify the highest and lowest values of criterion j across all providers.

$$Q_{CP_i} = \frac{OQoC + RR}{2} \tag{1}$$

$$C_{CP_i} = \sum_{j=1..n} w_j(M_{ij}) \tag{2}$$

$$M_{ij} = \begin{cases} \dfrac{Q_{ij}}{\max\limits_{\forall j} Q_j}; j \text{ needs to be maximized} \\ \dfrac{\min\limits_{\forall j} Q_j}{Q_{ij}}; otherwise \end{cases} \tag{3}$$

Quality and Cost of context measurement and validation. Upon receiving context from the providers (Step 5), the Quality Measurement (QM) unit evaluates their *QoC* metrics using the model in [3]. The $OQoC$ in (1) is calculated. ConCQEng assesses the V_{CP_i} of providers to ensure the accuracy of *QoC* parameters. Then, the QM unit forwards the evaluated *QoC* along with *QoS* and Quality of Data metrics to the Veracity Measurement (VM) Unit (Step 6). The VM Unit evaluates V_{CP_i} based on providers' performance in QoS (e.g., response latency) and Quality of Device (e.g., precision), which influence *QoC* metrics. This evaluation is then sent to the RR-processor for further analysis.

$$V_{CP_i} = \sum_{n=1}^{N} \left[QW_i(RM_{i,n}) + SW_i(SM_{i,n})\right] \tag{4}$$

$$RM_{i,n} = \begin{cases} \dfrac{QoS_{i,n}}{\max\limits_{\forall p} QoS_p}; QoS \text{ needs to be maximized} \\ \dfrac{\min\limits_{\forall p} QoS_p}{QoS_{i,n}}; otherwise \end{cases} \tag{5}$$

$$SM_{i,n} = \frac{R_{i,n}}{\max\limits_{\forall n} R_{i,n}} \tag{6}$$

$QW_{i,n}(RM_{i,n})$ and $SW_{i,n}(SM_{i,n})$ represent the *QoS* and Quality of Device (QoD) metrics of a context provider against a *QoC* metric $n \in \mathcal{N} := \{1,2,\dots N\}$. $QW_{i,n}$ and $SW_{i,n}$ are the weights of *QoS* and *QoD* metrics. $RM_{i,n}$ is the normalised value of a *QoS* metric from $p \in \mathcal{P} := \{1,2,\dots P\}$. $\max QoS_p$ and $\min QoS_p$ represent the maximum and the minimum values of each QoS metric measured of all potential context providers that could be used to respond to a *CR*. $SM_{i,n}$ represent the normalised compatibility level of a context provider to meet a *QoC* parameter based on device features (e.g., resolution). $R_{i,n}$ is the compatibility level and $\max R_{i,n}$ is the maximum compatibility level among all context providers for a *QoC* metric.

In Step 7, the QM unit sends the context from providers alongside *QoC* metrics and applicable Service Level Agreements to the CoC measurement unit for the final CoC calculation. Finally, ConCQEng relays these metrics along with the CoC to the CMS. After the CMS infers the context, determines a situation, and responds to the context consumer, the consumer provides feedback to the ConCQEng via the CMS. This feedback is aggregated to RR_{CP_i} and subsequently Q_{CP_i} using (4).

B. Adaptive Context Caching (ACOCA)

Based on the arguments in [5], context undergoes four stages in the "lifecycle" [16]. In each stage, a cost and performance-efficient context caching strategy has to address the 5Ws – *what* context to cache, *when*, *where*, *how* to cache, and *who* caches context information? The high-level architecture of ACOCA designed to address these questions is shown in Fig.3.

There are three key components to the adaptive context cache controller [16], in which the Context Prediction and Estimation Engine outputs the collection of directives addressing the 5Ws. The cache operations manager writes, updates, and evicts context in the cache memory. The cloud resource monitor is the master to all resource utility monitors

deployed in each stateful cache memory instance. They profile the virtual context cache for utility. Context resource management also performs adaptative cache memory scaling.

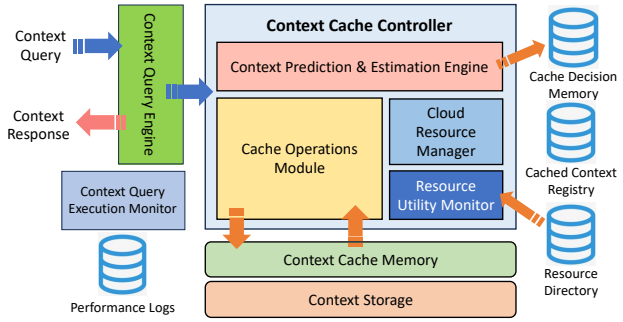


Fig.3. High-level architecture of ACOCA mechanism [16].

Let us discuss how the 5Ws are addressed in this architecture. Where, who, and how questions are intuitively addressed based on the design principles of the CMS. Consider the Context-as-a-Service (CoaaS) [12] platform that we used in our prototype. It is a Cloud-based system and stateful cloud cache memory instances (e.g., Redis-based) determine *where* to cache. The context cache controller that performs the caching decisions (*who*) is, hence, centralized. We chose to cache context entities (*how*) based on our previous work [16] using JSON-LD. As discussed in [5], addressing these problems is more complex involving edge or fog-based CMSs. For example, deciding what edge node to cache and what context is a research area that is yet to be investigated.

What context to cache and when? We developed an adaptive value-based system to make generic selective context caching decisions. Context entities are selected for caching using a binary decision criterion given by:

$$cache(i) = \begin{cases} true\,; conf(i) > \vartheta \\ false\,; otherwise \end{cases} \quad (7)$$

where $conf(i)$ is the confidence to cache a context entity i. ϑ is the adaptive decision threshold where $\vartheta: \theta \rightarrow \mathbb{R}^{+}$. θ is the area under the Z-distribution of recent historical $conf(i)$ values and left of z_θ. We calculate z_θ using θ and $\vartheta = \overline{Conf} + \sigma_{conf} \times z_\theta$. Consider $v(i)$ is the feature vector of a candidate context i,

$$v(i) = \{AT_i, CE_i, RE_i, Unreli_i, Cmpx(i)\}$$

AT_i is the access trend. CE_i is the caching efficiency. RE_i is the retrieval efficiency. $Unreli_i$ is the lack of reliability of the context providers from whom the data to derive the context is retrieved. $Cmpx(i)$is the average complexity of the context queries that this particular context is expected to be used in. $Cmpx(i)$ is calculated based on our novel concept of context query classes. It is aimed at statistically minimizing the uncertainty of context and their metadata (e.g., the probability distribution of applicable f_{thr}) for making caching decisions. Context query classes are identified by clustering context requests based on identifiable features [16].

We formulate our problem as follows:

$$\begin{bmatrix} AT_1 & CE_1 & RE_1 & Unreli_1 & Cmpx(1) \\ \vdots & & \vdots & & \vdots \\ AT_n & CE_n & RE_n & Unreli_n & Cmpx(n) \end{bmatrix} . \begin{bmatrix} \kappa \\ \mu \\ \omega \\ \delta \\ \rho \end{bmatrix} = \begin{bmatrix} conf_1 \\ \vdots \\ conf_n \end{bmatrix} \quad (8)$$

$\kappa, \mu, \omega, \rho, \delta$ are the weights assigned to each feature when making the context caching decision. So, $\{\kappa, \mu, \omega, \rho, \delta, \theta\}$ is the action set for an adaptive model. Action space is continuous $\mathbb{R} \cap [0,1]$.

How can we learn these weights to make optimal context caching decisions and adapt them to various scenarios? We employ a deep reinforcement learning model for this purpose. We chose the parametric model of Twin Delayed Deep Deterministic Policy Gradient (TD3) method in this process. So, the state space is defined using ten dimensions related to cost- and performance-efficiency metrics: cached context size, cache cost, earnings, penalties, retrieval cost, processing cost, probability of delay [6], hit rate, average cache lifetime ($\overline{CL}$), and average delay time ($\overline{DT}$) [16].

C. *Integration ACOCA with ConQEng*

Fig.4 illustrates the architecture and process of our mechanism. Cooperation between the ACOCA algorithms and ConCQEng for multi-objective and multi-attribute optimization occurs in three phases. They are as follows.

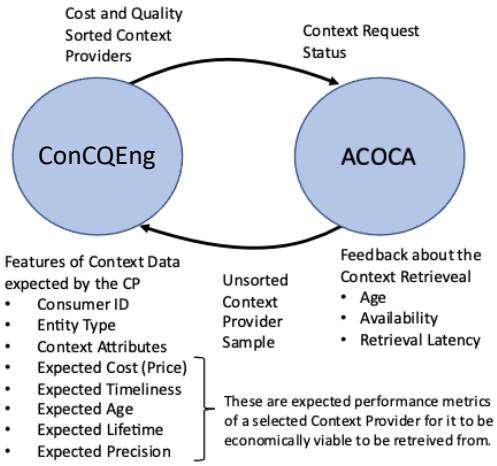


Fig.4. Process to concurrently mazximize QoC, performance-, and cost-efficiencies of the CMS.

Phase One. ACOCA uses ConCQEng to create a session for each context request, with a unique identifier attached to a process instance for cost- and quality-aware context provider selection. ACOCA shares the QoC, QoS, and cost-efficiency goals with ConCQEng to improve or preserve the current cost- and performance-efficiencies of the CMS. These goals filter out context providers that are either expensive, erroneous, unreliable, or slow to meet the objectives. ConCQEng uses ACOCA's feedback in phase three for this purpose.

Phase Two. Metadata about context providers based on spatial characteristics and QoC parameters are retrieved from the context providers registry. The CMS requests ConCQEng to prioritize filtered context providers based on cost and quality. ConCQEng executes multi-criteria decision-making and returns an ordered list of context providers. This phase is invoked when a cache miss occurs, context providers are no longer available or reliable for refreshing, or context providers used to refresh cached context are re-evaluated. Context providers may fail to be consistent, such as when a context

provider is more relevant to a situation than another, requiring eviction, interpretation from new data, and cached. This ensures consistency in achieving goals and ensuring efficient use of resources.

Phase Three. After each retrieval operation (including context refreshing), ACOCA sends feedback to ConCQEng about the effectiveness of selected context providers in achieving the set efficiency goals. Feedback contains the availability, completeness, age of context, and retrieval latency.

How are the efficiency objectives concurrently achieved in this process? ConCQEng's selected context providers ensure a high level of *QoC and* minimizes the cost of context retrieval. This improves caching ($CacheEff$) and retrieval efficiencies ($RetEff$) of the ACOCA cost model while minimizing unreliability. Timely context has a longer cache residence time, reducing the refresh rate and maximizing the hit rate of the context cache [6]. This maximizes *QoS* and improves the confidence on $QoCAdR$ values for context providers. This incremental effect on efficiency goals set by ACOCA contributes to ensuring that selected context providers are capable of satisfying the *QoC* needs of context requests. The improvement in *QoC*, *QoS*, and cost efficiency of the CMS for cached and delivered context information over multiple selection cycles contributes to a more efficient and reliable system.

V. Evaluations

In this section, our mechanism to concurrently optimize a CMS for quality, and cost of context along with quality of service is evaluated.

A. Experimental Setup

We implemented our mechanism and integrated with the Context-as-a-Service (CoaaS) platform [12] developed as a part of the bIoTope project (https://biotope-project.eu/) for the European Commission under the Horizon 2020 program. CoaaS and ACOCA are implemented in Java using the microservice architecture and can be found in GitHub[1]. Deep reinforcement learning models were developed using Python and TensorFlow libraries. Deep neural networks have an input, output, and two hidden layers each. The discount factor was set to 0.9 and the learning rates of the actor and critic networks were set to 0.001 and 0.002 respectively. NodeJS was used to implement ConCQEng. Runnable containers were deployed in a virtual machine (VM) hosted by Deakin University, which has 8GB RAM, 8 processor cores, 2593.33MHz clock speed, and runs Linux. The VM's specifications align with typical edge devices, and communication between ACOCA-ConCQEng occurs using RESTful APIs. A set of 92,720 context queries[2] was used to simulate the scenario, with the request rate (λ) conforming to a Poisson distribution and the complexity being normally distributed [16] where $\mu = 2.925$ and $\sigma^2 = 0.5407$. The study simulated forty context services using the IoT data simulator in this setup.

B. Benchmarks

With no previous working implementation of a *QoS*, *QoC*, and/or cost efficiency improvement strategy for CMSs in the literature, comparing empirical evidence of CMS's improvements using our mechanism is difficult. Context caching is different from data caching contributing to this rationale [19]. So, we benchmarked our mechanism against a CMS running in two modes: redirector – not storing any context, and context caching. Redirector mode was tested with and without ConCQEng denoted respectively as quality-aware (QA) and quality-oblivious (QO) redirectors. Context caching was similarly tested without ConCQEng. Hence, we attempt to establish the absolute advantage of our cooperative optimization strategy compared to integrating ACOCA or ConCQEng alone. Traditional replacement-based data caching (i.e., Least Recently Used) and traditional context-aware caching (that maximizes the hit rate) were used as external benchmarks.

C. Results and Discussions

Twenty context consumers with different *QoC* and *QoS* needs were randomly generated for testing. The set of *QoC* and *QoS* parameter values comply with the feasible SLA criteria [6]. Table II summarizes these parameter values for all consumers. $Price_{res}$ and $Cost_{pen}$ are defined in Dollars. RT_{max} is defined in milliseconds and let $f_{thr}, Prec \in \mathbb{R}_0^1$.

In the Figures below, dark blue represents our mechanism, orange represents CMS integrated with ACOCA only, grey represents context-aware data caching (referred to as Popularity), yellow represents traditional data caching that uses LRU eviction policy, light blue represents a CMS operating in redirector mode integrated with ConCQEng (referred to as QA-Redirector), green represents a CMS operating in redirector mode only (referred to as QO-Redirector) and pink represent a CMS operating in database mode (as is the case with the state-of-the-art CMSs).

TABLE II. Summary of QoC and QoS parameter values defined by consumers in the statical setup.

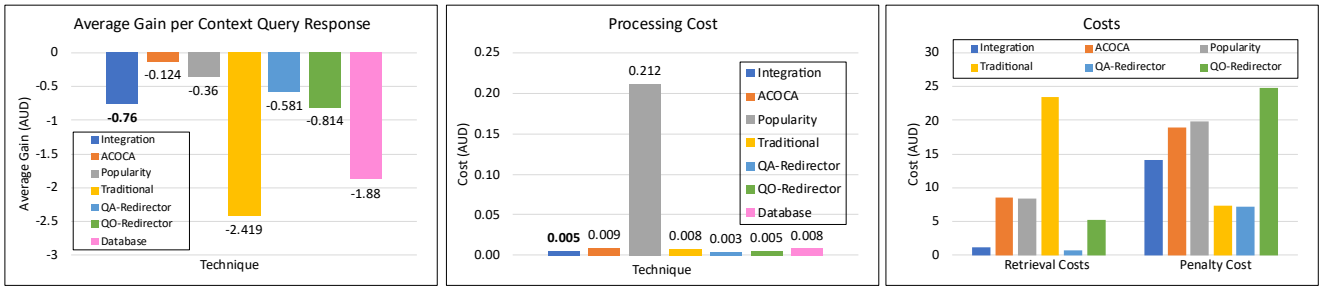
	$Price_{res}$	f_{thr}	RT_{max}	$Cost_{pen}$
Average	0.5198	0.74	2217	0.6925
S.Dev.	0.3844	0.1071	618.80	0.4347
Min	0.0070	0.5	14400	0.1000
Max	1.4000	0.9	3600	2.0000

Cost Efficiency. Fig.5 compares the average gain, processing cost per context query response, retrieval cost, and penalty cost per window of benchmarks. The average gain was minimal using only ACOCA, followed by context-aware caching (which we will refer to as popularity-based caching). Based on Fig.5(b), popularity-based caching outlies all other benchmarks based on overhead cost. It proves why popularity-based data caching is not fully applicable to caching context information. ACOCA results in CMS being 95% more profitable than traditional caching, 85% more profitable than redirector mode, and 93% more profitable than database mode used by state-of-the-art CMSs.

Our integration resulted in the minimum processing cost despite the overhead introduced by ACOCA and ConCQEng to the CMS. The retrieval cost was minimal using our mechanism, resulting in a 99.5% reduction compared to most state-of-the-art CMSs. The minimized retrieval cost is due to ACOCA cooperating with ConCQEng to cache and refresh context information from cheaper and reliable providers. Our mechanism also incurred the least penalty cost compared to ACOCA, popularity-based caching, and traditional redirector

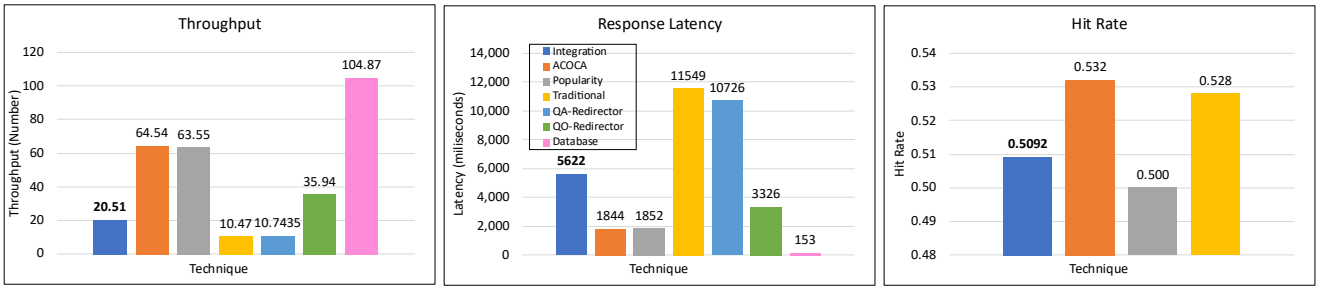
[1] Available at: https://bit.ly/reactive-acoca

[2] Available at: http://bit.ly/3h7yuSU



(a) (b) (c)

Fig.5. Comparison of (a) average gain, (b) processing cost per context query response, and (c) retrievals cost and penalty cost per window.



(a) (b) (c)

Fig.6. Comparison of (a) throughput, (b) response latency, and (c) hit rate per window.

mode, down by 25%, 29%, and 43% respectively. But why our mechanism is less profitable? The reason can be found in Fig.6. Hence, we will discuss this problem in the next sub-section.

Performance Efficiency. Fig.6 compares the throughput, response latency, and hit rate per window among benchmarks. The CMS observed the maximum throughput, hit rate, and minimum response time using ACOCA, which showed a higher hit rate than the benchmarks. Despite being lightweight and cheaper to operate, ACOCA and ConCQEng increased response latency. Quality-aware redirectors are 70% lower in throughput and 222% higher in response latency compared to traditional redirector modes. Cost and quality-aware context provider selection introduces processing time overhead, resulting in penalties incurred by the CMS.

To minimize average gain, the authors integrated ConCQEng sparingly into the ACOCA and context query execution process. ACOCA periodically calls ConCQEng to refresh the priority of selected context providers for retrieval and refreshing based on newer and current context. This process improves cost-efficiency and QoC but increases the probability of context cache misses. ACOCA evicts context from previous prioritized providers when priorities change, improving cost-efficiency and QoC. However, the probability of context cache misses increases, reducing the hit rate from 0.532 to 0.509 using our mechanism, resulting in a decrease in the hit rate. The response latency reduction using ACOCA is 1482ms, while the increase using ConCQEng is 7400ms. The few context queries that cache miss result in significant penalties, resulting in the reduction of average gain despite the minimum penalties incurred by the mechanism.

Improvement in QoC. Fig.7 shows the age of context in cache, excluding popularity-based caching as it aims to cache context having longer lifetimes (i.e., non-transient data). The database mode of operation is excluded as it ignores context freshness and synchronizes refresh with provider sampling rates. Khargharia et al.'s context caching algorithm [15] also ignores context age. Database mode provides the most aged context information for context queries, with an average age of 24411ms, 66 times higher than our mechanism. Traditional data caching shows the minimum age of context due to frequent context replacements and proactive refresh, which contribute to outlier retrieval costs compared to benchmarks. This trade-off cannot be accepted.

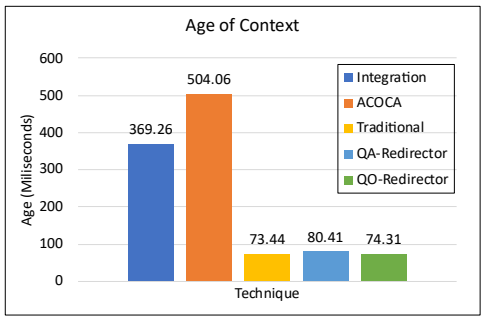


Fig.7. Average age of context information used in context response.

We expected the redirector modes to result in the least age since context is retrieved ad hoc for each query compared to context caching which reuses and repurposes the context retrieved for other context queries. Accordingly, our proposed mechanism outperforms traditional context-aware caching and ACOCA, due to greater context selection efficiency and the ability to retrieve context from providers that provide the freshest data to infer and cache the context.

VI. Conclusions

This paper proposes a mechanism to optimise cost- and performance-efficiencies, and Quality of Context (QoC) in Context Management Systems (CMSs). Our novel solution which comprises an Adaptive Context Caching (ACOCA) mechanism and a cost and quality-aware context provider mechanism called ConCQEng, effectively maximizes QoC while reducing context access and processing costs, a key goal for an effective CMS. Our evaluation using the autonomous

car parking scenario revealed that a cooperative deployment of ACOCA and ConCQEng shows that CMS responds with 27% fresher context information compared to using only ACOCA. Our mechanism is the most cost-efficient among the benchmarks, incurring the minimum retrieval, penalty, and processing costs, e.g., up to 43% reduction in penalty costs and up to 98% reduction in processing costs. We also showed our technique incurs 99% less context retrieval costs compared to the database mode. Future work will focus on developing a fully distributed context caching system for CMSs to further enhance scalability.

ACKNOWLEDGMENT

Support for this publication from the Australian Research Council (ARC) Discovery Project Grant DP200102299 is thankfully acknowledged.

REFERENCES

[1] I. Ullah, J.-B. Kim, and Y.-H. Han, 'Compound Context-Aware Bayesian Inference Scheme for Smart IoT Environment', *Sensors*, vol. 22, no. 8, p. 3022, Apr. 2022, doi: 10.3390/s22083022.

[2] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, 'Context Aware Computing for The Internet of Things: A Survey', *IEEE Commun. Surv. Tutor.*, vol. 16, no. 1, pp. 414–454, 2014, doi: 10.1109/SURV.2013.042313.00197.

[3] A. Manzoor, H.-L. Truong, and S. Dustdar, 'Quality of Context: models and applications for context-aware systems in pervasive environments', *Knowl. Eng. Rev.*, vol. 29, no. 2, pp. 154–170, Mar. 2014, doi: 10.1017/S0269888914000034.

[4] H.-P. Schwefel, M. B. Hansen, and R. L. Olsen, 'Adaptive Caching Strategies for Context Management Systems', in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, Athens, Greece: IEEE, 2007, pp. 1–6. doi: 10.1109/PIMRC.2007.4394813.

[5] S. Weerasinghe, A. Zaslavsky, S. W. Loke, A. Hassani, A. Abken, and A. Medvedev, 'From Traditional Adaptive Data Caching to Adaptive Context Caching: A Survey', *ArXiv221111259 CsHC*, p. 35, Nov. 2022.

[6] S. Weerasinghe, A. Zaslavsky, S. W. Loke, A. Medvedev, and A. Abken, 'Estimating the Lifetime of Transient Context for Adaptive Caching in IoT Applications', in *ACM Symposium on Applied Computing*, Brno, Czech Republic: ACM, Apr. 2022, p. 10. doi: 10.1145/3477314.3507075.

[7] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, 'Towards a Better Understanding of Context and Context-Awareness', in *Handheld and Ubiquitous Computing*, vol. 1707, H.-W. Gellersen, Ed., in Lecture Notes in Computer Science, vol. 1707. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 304–307. doi: 10.1007/3-540-48157-5_29.

[8] K. Fizza *et al.*, 'A Survey on Evaluating the Quality of Autonomic Internet of Things Applications', *IEEE Commun. Surv. Tutor.*, pp. 1–1, 2022, doi: 10.1109/COMST.2022.3205377.

[9] K. S. Jagarlamudi, A. Zaslavsky, S. W. Loke, A. Hassani, and A. Medvedev, 'Requirements, Limitations and Recommendations for Enabling End-to-End Quality of Context-Awareness in IoT Middleware', *Sensors*, vol. 22, no. 4, p. 1632, Feb. 2022, doi: 10.3390/s22041632.

[10] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm, 'Next century challenges: Nexus—an open global infrastructure for spatial-aware applications', in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seattle Washington USA: ACM, Aug. 1999, pp. 249–255. doi: 10.1145/313451.313549.

[11] G. Judd and P. Steenkiste, 'Providing contextual information to pervasive computing applications', in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, Fort Worth, TX, USA: IEEE Comput. Soc, 2003, pp. 133–142. doi: 10.1109/PERCOM.2003.1192735.

[12] A. Hassani *et al.*, 'Context-as-a-Service Platform: Exchange and Share Context in an IoT Ecosystem', in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Athens: IEEE, Mar. 2018, pp. 385–390. doi: 10.1109/PERCOMW.2018.8480240.

[13] M. Wagner, R. Reichle, and K. Geihs, 'Context as a service - Requirements, design and middleware support', in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Seattle, WA, USA: IEEE, Mar. 2011, pp. 220–225. doi: 10.1109/PERCOMW.2011.5766873.

[14] A. Medvedev, 'Performance and Cost Driven Data Storage and Processing for IoT Context Management Platforms', Doctoral Thesis, Monash University, Australia, 2020. Accessed: Jul. 27, 2021. [Online]. Available: https://bridges.monash.edu/articles/Performance_and_Cost_Driven_Data_Storage_and_Processing_for_IoT_Context_Management_Platforms/12423479

[15] H. S. Khargharia *et al.*, 'Probabilistic analysis of context caching in Internet of Things applications', in *2022 IEEE International Conference on Services Computing (SCC)*, Barcelona, Spain: IEEE, Jul. 2022, pp. 93–103. doi: 10.1109/SCC55611.2022.00025.

[16] S. Weerasinghe, A. Zaslavsky, S. W. Loke, A. Hassani, A. Medvedev, and A. Abken, 'Adaptive Context Caching for IoT-Based Applications: A Reinforcement Learning Approach', *Sensors*, vol. 23, no. 10, p. 4767, May 2023, doi: 10.3390/s23104767.

[17] K. Sai Jagarlamudi, A. Zaslavsky, S. W. Loke, A. Hassani, and A. Medvedev, 'ConQeng: A Middleware for Quality of Context Aware Selection, Measurement and Validation', in *Internet of Things*, vol. 13533, A. González-Vidal, A. Mohamed Abdelgawad, E. Sabir, S. Ziegler, and L. Ladid, Eds., in Lecture Notes in Computer Science, vol. 13533. , Cham: Springer International Publishing, 2022, pp. 211–225. doi: 10.1007/978-3-031-20936-9_17.

[18] K. S. Jagarlamudi, A. Zaslavsky, S. W. Loke, and K. Lee, 'Validating Quality of Context in Pervasive Computing Systems: Surf Life Saving Use Case', in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Atlanta, GA, USA: IEEE, Mar. 2023, pp. 577–582. doi: 10.1109/PerComWorkshops56833.2023.10150324.

[19] S. Weerasinghe, A. Zaslavsky, S. W. Loke, A. Medvedev, A. Abken, and A. Hassani, 'Context Caching for IoT-Based Applications: Opportunities and Challenges', IEEE Internet Things M., vol. 6, no. 4, pp. 96–102, Dec. 2023, doi: 10.1109/IOTM.001.2200247.