A survey on self-adaptive cloud

Xiang Zhou, Fudan University
Chenjie Xu, Fudan University

Abstract—As the cloud system is becoming more and more popular and widespread these days, The complexity of cloud systems is also increasing in recent years, leading to increased effort for maintenance and configuration. Self-adaptive Cloud Systems (SACS) address this issue. Due to new computing trends, such as pervasive computing, miniaturization of IT leads to mobile devices with the emerging need for mobile-cloud convergence way of adaptation. Internet-ware is aiming to make a new definition of future software. Hence, we propose to analysis the previous adaptive cloud research and together with mobile computing in adaptation. This paper presents a taxonomy of Adaptive and Self-Managing Cloud Systems and a survey on SACS. Based on the taxonomy and the survey, we motivate a new idea on SACS.

Index Terms—Software Engineer, Cloud, adaptive, automatic, SAAS, PAAS, IAAS, mobile

1 Introduction

The complexity of current cloud systems is increasing. Due to the growing number of mobile devices as well as the widespread relatively high speed wireless network, users nowadays can almost access the service whenever and wherever they want, while at home, at work, or even on the way home/work. Some cloud systems are running distributed and also kind of integrate with several available, highly specialized cloudlet-like or mobile devices, e.g. distributed sensor, energy-free mobile, it gather the distributed data streams that operate in an ever-changing environment with flitter network resources and availability in order to make further decision. Furthermore, cloud systems are no longer restricted to centralized, tightly controllable areas with stable administrative responsibility, they are somewhat interconnected, leading to truly pervasive, global systems just like Smart Cities or the Internet of Things.

Developing, configuring, and maintaining such cloud systems is a very difficult, error prone, and time consuming task. One promising way to reduce this effort is self-adaptation. A self-adaptive cloud system (SACS) is able to automatically modify itself in response to changes in its operating environment [1]. The modification is done by adjusting attributes (parameters) or artifacts of the system in response to changes in the cloud system itself or in its environment. In recent years, SACS have seen an increasing level of interest in different research areas like Pervasive Computing, Autonomic Computing [35], and Nature-Inspired (Organic) Computing [37].

SACS provide self-management properties like self-configuration, self-healing in the presence of failures, self-optimization, and self-protection against threats [40]. For achieving adaptive behavior, basic cloud system properties are self-awareness and context-awareness [39]. Self-awareness describes the ability of a cloud system to be aware of itself, e.g. to be able to monitor its own resources(cpu, memory consumption), state, and behavior. Context-awareness means that the cloud system is aware of its environment, the so called context. According to the current situation, context can be any stuff that working around the target cloud system. it could be a machine, a place, or a mobile application which interact with the cloud system. The cloud system can also uses detector component or even sensor to collect information about its context and reasons about the information.

In this paper, we provide a structured overview of self-adaptation and approaches for SACS, and motivate the need for a new perspective on self-adaptation in pervasive computing systems. Our main contributions are as follows: First, we develop a taxonomy for self-adaptation cloud system that integrates existing views on self adaptation and specifically context-adaptive cloud systems, which are most relevant to pervasive computing. Second, we survey existing approaches for engineering SACS. Third, we discuss a new idea of SACS.

These contributions are directly reflected in the structure of the remaining part of the paper. In the next section, our taxonomy for self-adaptation is presented. In Section 3, we present the previous researches on SACS. Based on the taxonomy and the researches, in Section 4 we describe a new idea of a SACS. and finally, we have a conclusion for this paper.

Nov 1, 2015

2 Taxonomy

In this section, we summarize different aspects and perspectives on self-adaptation in SACS research and adaptation in general, and present them in a comprehensive taxonomy for self-adaptation. Our taxonomy is based on different perspective of self-adaptive cloud systems. 1 shows an overview of our taxonomy. In the remaining part of this section we discuss the different perspective of our taxonomy.

- *Xiang Zhou was with the Department of Software Engineering, Fudan University, China.*
  E-mail: 15110240026@fudan.edu.cn
- *Chenjie Xu was with the Department of Software Engineering, Fudan University, China.*

Manuscript received Nov 1, 2015; revised August Nov 1, 2015.

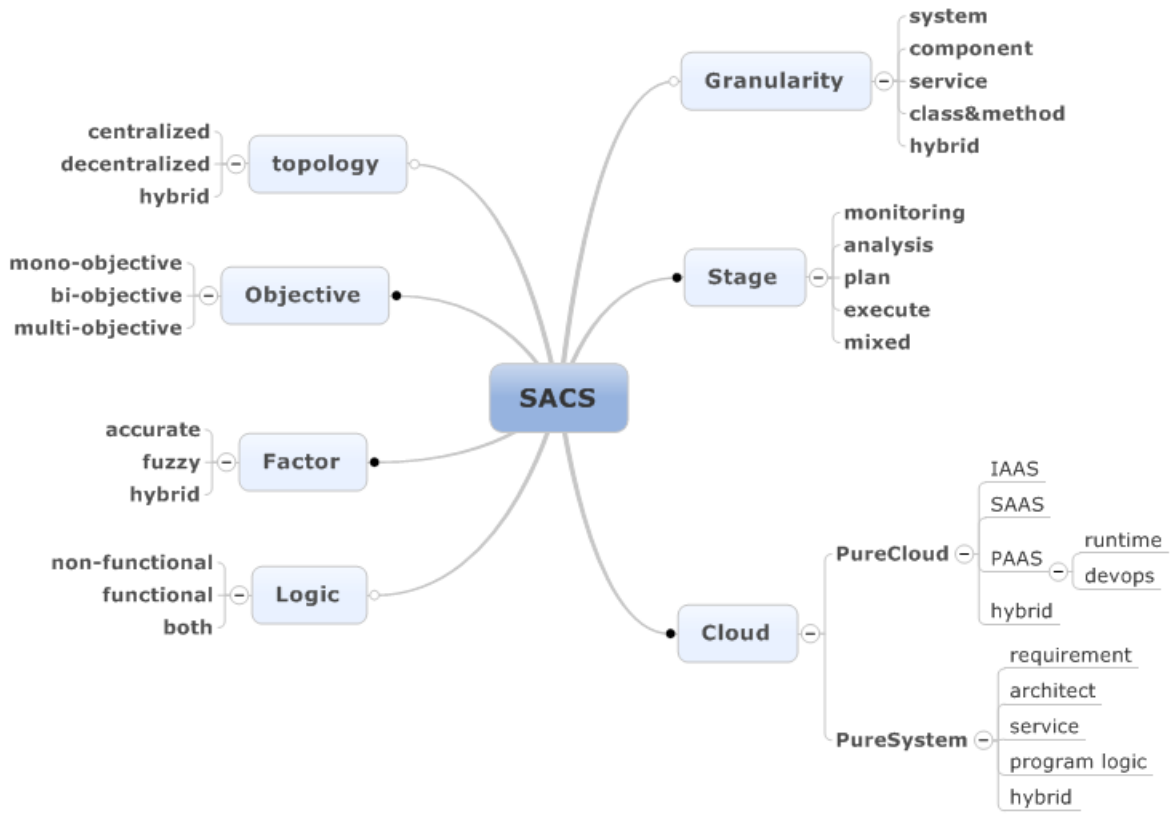

Fig. 1: Taxonomy of Self-Adaptive Cloud System

Before going into the detail of our taxonomy, we have to clarify that the "cloud" here is not referring to the narrow-sense cloud, besides, it not only include the server technology but also mobile stuff, which have compute and storage capability, so cloud here can be almost everything, no need to mapping to the IAAS, PAAS or even openstack stuff.

2.1 Stage

The Stage aspect is related to the MAPE-loop, as the Monitoring, Analysis, Plan, and Execution is the 4 standard steps in adaptive system, so each step can be a focus area for us to investigate in to make novelty progress for self-adaptive research. some researches are focus on improving monitoring step, e.g. [20], while some others are optimizing analysis step, the plan step is the most popular one, while the execution step is not widely touched.

2.2 Granularity

In general, a SACS is composed of two different elements: the managed element(s) and the self-adaption logic. the adaptation logic(the control unit) often stays stable, while the managed elements can be adapted. The managed elements are composed of various levels. It can be a entire cloud system such as a PAAS provider, or part of the cloud system such as a storage component, or even a method of a class. By the monitoring result, the adaption can be system requirement level, [5] or component/service level, [57] or even class/method level for very specific functional/performance tuning. [60] This is why we have a Granularity aspect.

2.3 Cloud

In Cloud aspect, we divide them into cloud specific adaptive and non-cloud related adaptive. the cloud specific refers to the widely-used cloud system: IAAS, PAAS, SAAS. the adaptive logic mainly focus on the cloud specific mechanism adaptation and always together with cloud related technology, e.g. hypervisor related stuff. for PAAS, there are also tow major direction, the runtime and the devops. runtime refers to the cloud process adaptation, while devops is mainly on development process and operation process, as PAAS is kind of a middle-layer between IAAS and SAAS, and its "customer" is usually the developer who always know a lot about IT technology, so it always face big challenge on various kinds of developers in different experience level. and others are non-cloud specific system, which can be any kind of software or hardware system [9]

2.4 Topology

Topology is not a new word, but it matters in adaptive cloud system, the majority of the current and previous research are based on centralized MAPE loop, Centralized self-adaptation with a central instance for control, Within a centralized adaptation logic, one sub-system implements the adaptation logic, is responsible for monitoring the context and all resources, and controls adaptation. Whereas a global maximum-like goal can be achieved due to the

central instance is aware of all information, however, this approach is always not suitable for large systems, as it has to face a large amount of information, and the tremendous calculation power required. For large systems, a central approach is hard to achieve because of the size of the system and real time constraints. [19] so the decentralized approach invented, [20] the MAPE can be host on any part of the system, each part of component can manage itself, and it can also communicate with each other to optimize. as the fully-decentralized system are often not well planned comparing to centralized system, there come out semi-decentralized system, which combine these two approach together, in order to address the disadvantage from both side, and also trying to adopt the advantage from both too. we can refer to 2

2.5 Objective

According to objective aspect, we divide it into mono-objective, bi-objective, and multi-objective, which means the goal of the adaptive logic. different adaptive logic in adaptive cloud system have different goals, but most of the previous research will concentrate on one or two goal of their adaptive result. There also have multi-goal optimization self-adaptive cloud system which covers a wide range of adaptive points.

2.6 Factor

Factor is mainly referring to the algorithm of self-adaptive cloud system, some are fuzzy-based or probability-based, while others are accurate value with programming logic.

2.7 Logic

There are always both functional and non-functional requirement in each software system, so does the self-adaptive cloud system, the adaptation result is to meet the functional requirement or the non-functional requirement is totally different approach. the non-functional approach is more-like a technical specific [46] while functional approach is including lots of business stuff in it, which lead to more complexity.

3 Survey

So far, we presented a consolidated taxonomy based on the characteristic of the existing research. Next, we are going to take a look at the different approaches of how a concrete adaptation is implemented. In this section, we categorize and present these approaches. Furthermore, we link the approaches to the dimensions of our taxonomy and the MAPE activities.

Our purpose is to present the diversity of the different approaches for SACS development. We will not discuss every detail of these approaches, what we propose here is to throw away a brick in order to get a gem, we will briefly discuss their strengths and weaknesses, and for the interested reader to referred to the cited works for more detailed information about the approaches.

From a cloud-related perspective, we'll begin with following aspects:

3.1 Pure System Specific

the pure system means it is not include cloud related technic, so go through those researches, we have different approaches:

- Requirement-based approaches
- Architecture-based approaches
- Service-oriented approaches
- Programming Logic
- Hybrid

3.1.1 Requirement-based approaches

In requirement-based section, what we mostly see is the goal-oriented system. e.g. [5] demonstrate the most crucial challenges in the engineering of Personalized Web-Tasking (PWT) systems can be addressed by implementing them as self-adaptive software. The Personalized Web-Tasking (PWT) proposes the automation of user-centric and repetitive web interactions to assist users in the fulfilment of personal goals using internet systems. it supports three levels of

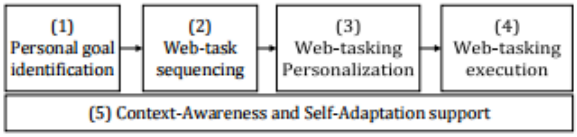

Fig. 3: Personalized Web-Tasking conceptual elements

automation: manual (i.e., the user explicitly specifies which web interactions correspond to a particular personal goal), assisted by recommendations (i.e., the system suggests web interactions to the user), and fully automated (i.e., the system executes web interactions on behalf of the user)

3.1.2 Architecture-based approaches

A software architecture is a set of software components, the relations between them, as well as the properties of both and denotes the high-level structure of software. Regarding adaptation purposes, software architectures are used for different activities. A SACS must be aware of its structure, which is called self-awareness. Software architectures can be used for representing the system structure and reasoning about adaptation levels. On the other hand, software architectures are used for the construction of SACS and defining responsibilities [38]. In this section, we present architecture-based approaches.

The Rainbow framework [57] is one of the most well-known frameworks for SACS. Driven by external control for self adaptation, the SACS is divided into an architecture layer and a system layer with the managed resources. The architecture layer has different components, which define adaptation plans. 4

As architect level is a kind of high level of a software system, but it also consist of the implementation of different functional unit detail, so some paper [12] also considered: What is the impact of the detail system changes on a systems architecture in a general case? it's fully of challenges to perfectly extracting the architecture of a system from its implementation artifacts, and fully of difficulties of tracking the architectural impact of implementation-level changes.

[10] want to schedule resources adaptively with the sole aim of reducing power consumption, based on a characterization of energy usage and resource utilization patterns

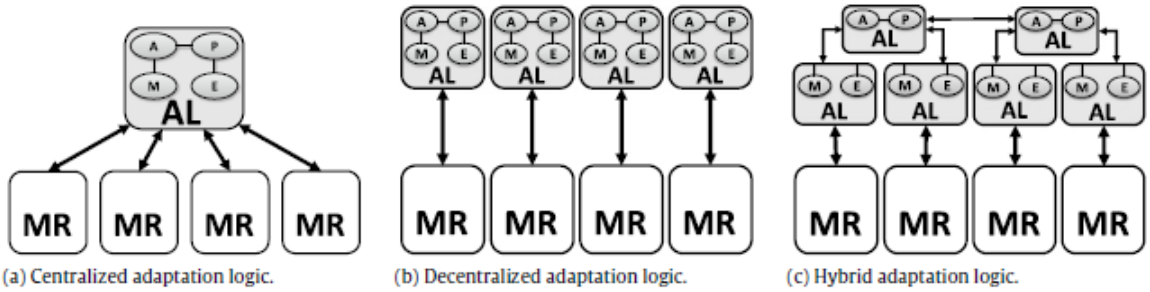

Fig. 2: Comparison of adaptation logic structures (AL = Adaptation Logic, MR = Managed Resources, M,A,P,E = MAPE functionality)

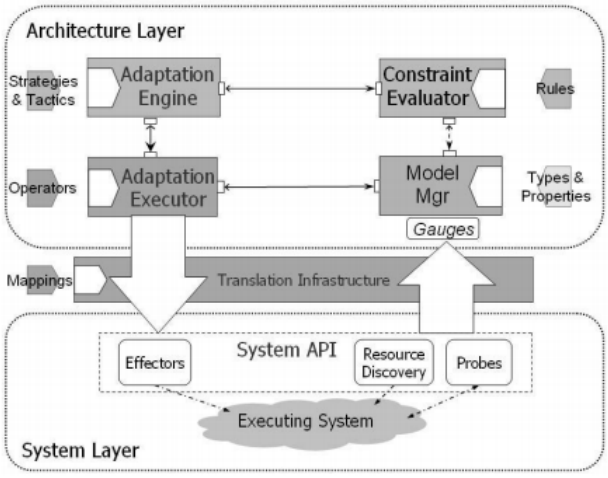

Fig. 4: The Rainbow framework

obtained by monitoring energy consumption in an enterprise data center. it propose an adaptive feature extraction method to classify resource utilization patterns from energy consumption data. It describes:

1) a classifier for energy consumption patterns
2) a reference model for adaptive energy management
3) an adaptive energy pattern extraction method (extract high value/cost process)

which is used to be able to dynamically identify the type of process that is running on a server, based solely on the energy consumption. This is needed to make decisions in the Adaptive and Planning components of a dynamic runtime model using the MAPE-K loop paradigm.

[31] also trying to evolving an Adaptive Industrial Software System to Use Architecture-Based Self-Adaptation, based on Rainbow framework.

[30] established several Controlled Experiment to compare the adaptive systems of using External Feedback Loops to Improve Design to not(just internal factors). which proves that applying external feedback loops can reduce control flow complexity and fault density, and improve productivity.

3.1.3 Service-oriented approaches

SOA is a popular word for decades, the services here are refer to the small, encapsulated, and autonomous units of software that fulfill a specific task. In Service-Oriented Computing (SOC) such services are used to "support the development of rapid, low-cost, interoperable, evolvable, and massively distributed functionalities", which is especially match the current PAAS application preferred architect. From webservice to restful api, the SOA mind is almost everywhere. The adaptation logic for SOA decides, which services should run. Therefore, dynamic exchange of services offers structural adaptation capabilities [58]. The problem is, how to build the adaptation logic and how to enable the exchange of services.

[59] present a novel approach targeted at the challenges of automatically composing an SOA software system in dynamic, unpredictable, and pervasive SOA software systems. it rely on the domain experts functional and Quality of Service (QoS) requirements expressed in an activity-oriented language to automatically generate an "optimal" architecture. Architecture generation consists of (1) selection of the appropriate service providers that satisfy the users activities, and (2) application of suitable behavior patterns to compose the services into a cohesive software system.

[35] is aiming to build and maintain an assembly of services that, besides functional requirements, is able to fulfill global quality of service (QoS) and structural requirements. A set of simulation experiments is used to assess the effectiveness of our approach in terms of convergence speed towards the optimal solution, and resilience to failures. The system we consider thus consists of a set S of N distributed services S = S1, . . . , SN . Services can be located anywhere and can communicate each other through a network. Each service has a provided interface, which is an interface through which it provides functionalities to clients. Also, each service has a set of required interfaces, that must be bound to the provided interfaces of other services. centered on the use of a gossip protocol to achieve decentralized information dissemination and decision making in order to makes it able to fulfill the goal of best global QoS.

[41] introduce a semantic similarity and semantic nearness service composition method. The objective of our work is to free service nodes from adherence to restrictive composition plans. It enables compositions to succeed in dynamic environments through a decentralized solution

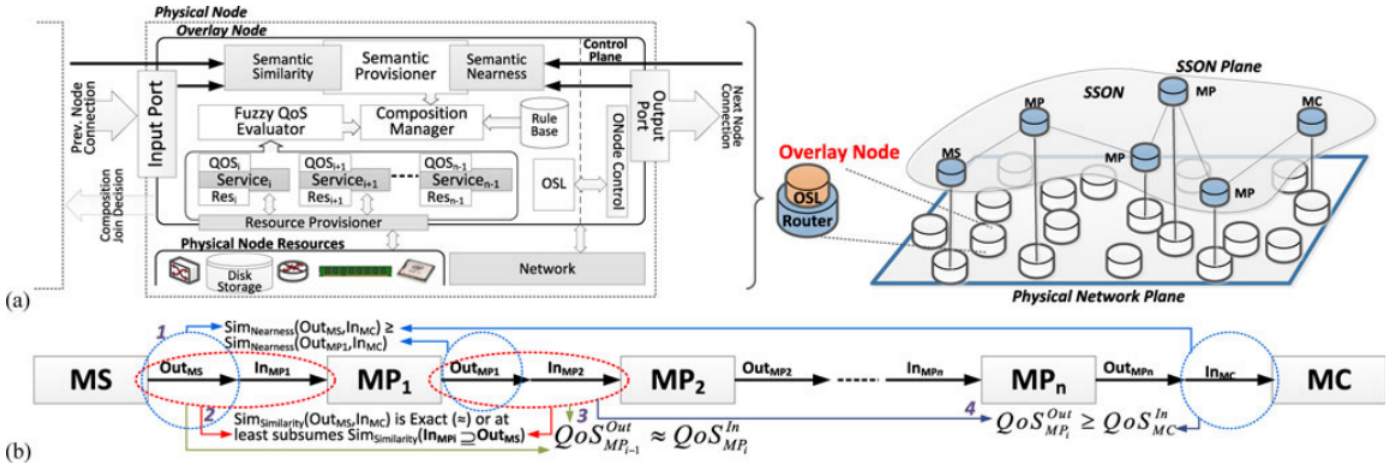

Fig. 5:
(a) System layers and overlay node architecture.
(b) Abstract SSON path for a successful service path composition
Media ports (MPs) to modify content and provide services such as caching, adaptation, and synchronization
service specific overlay networks (SSON) are established between media servers (MS) and users, or media clients (MC).
Goal: creating overlay networks that enable a controlled integration of advanced media routing, adaptation and caching mechanisms along the end-to-end media delivery path

even when nodes are unaware of nearby service nodes. 5

3.1.4 Programming Logic

Programming logic level is a lower level comparing to service-oriented level. It focus on the adaptation of class or even method stuff.

[60] is aiming to provide a self-management mechanism that can dynamically monitor and alter the execution of an existing application by selectively modifying certain procedures to execute remotely when it is necessary to improve safety and performance. for example: It automatically converts existing monolithic C applications into a distributed system semi-automatically based on (1) latency, (2) security. it should consider the confidential of the data and also the

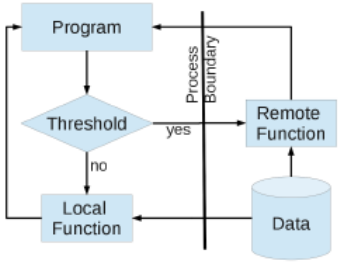

Fig. 6: Decision to move data to local server or to use remote function is based on threshold

latency of the network, to decide whether to execute the method locally or remotely.

[42] is a object level programming adaptive methodology. It proposes a library of self-adaptive containers which provide a ready route to developing scalable applications with low programmer overhead. Given an execution environment, the library flexibly adapts its use of data structures in an effort to meet programmer-specified service level objectives. It provides resource-aware container library which dynamically adjusts the underlying data structure associated with each container at execution time in order to satisfy resource constraints.

For example: deque (double-ended queue), list, and vector. set, multiset, map, and multimap.

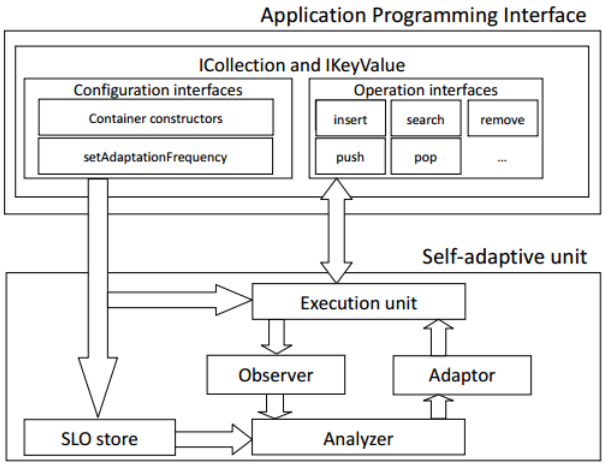

Fig. 7: The architecture of the library

The SLO Store: The SLO store holds all service level objectives laid down by the configuration interfaces. These objectives include per operation response times (insertion time, search time, and deletion time), maximum primary memory usage, and reliability. take a look at the following sample SLOs for library container:

1) 90 percentage of insertion times should be less than 1000ns, and 85percentage of search times should be

less than 1200ns.
2) Reliability should be higher than 0.99.
3) Memory consumption should be no more than 7.5GB

The Aspect-Oriented Programming (AOP) is also a example of programming logic aspect. the program is divided into distinct parts, called concerns. The goal is to use generic functionality in different classes (cross-cutting concerns). Therefore, logical concerns are separated from the concrete implementation [61]. One example of an AOP language is AspectJ, an extension to Java [62]. Whereas dynamic recomposition is often related to cross-cutting concerns as QoS, some paper propose AOP for enabling separation of concerns, which leads to simplified compositional adaptation. some paper show different possibilities to use AOP techniques for self-adaptation [63].

3.1.5 Hybrid

In Hybrid approaches, researcher are mostly trying to bring in the advantages from different approach, e.g. [1] propose a hybrid approach to combine the architect change and requirement change together. if architect modification can not meet the current adaptive requirement, it'll change the requirement(goal model) with the least influence, and then to see whether there are corresponding architect change match the changed requirement, and so on, to have a finite two-steps adaptive loop to match the goal.

there are also some special papers which focus on special automation, e.g. [6] propose an original application of dynamic software reconfiguration techniques to establish a moving target defense against browser fingerprint tracking. it aim at regularly changing the four most distinctive elements in a fingerprint: the operating system, the browser, the list of fonts and the list of plugins. The main objective here is to automatically modify the fingerprint a platform exhibits, in order to defense the browser(user) fingerprint tracking system on the internet, which can more or less, bring in more safety.

[32] start a comparative study between the requirement-based adaptive system and the architect-based system.

3.2 Cloud System Specific

As for the cloud related technic, it always divided into following aspects:

- IAAS-based approaches
- PAAS-based approaches
  - Runtime
  - DevOps
- SAAS-based approaches
- MCC
- Hybrid

3.2.1 IAAS

The IAAS is mostly a hypervisor MAPE loop, which can dynamic control the resources of CPU, memory, etc.. in runtime to make better benchmark or revenue. [9] is a good sample of the SACS on IAAS platform. It propose Hogna, a platform for deploying self-managing web applications on cloud. The platform enables the deployment of the applications based on the automation of a set of operations (starting instances, installing necessary software and configuring the instances, etc.), and then the continuous monitoring of the health of the applications. The gathered monitoring data is analyzed using a performance model and an action plan is created and executed. Any components involved (for monitoring, analyzing, planning and deployment changes) can be customized to fit the needs of the application and/or researcher. please refer to the figure 8

The [24] also provides a Volunteer clouds, where participants join and leave the platform and collaborate by sharing computational resources. The high complexity, dynamism and unpredictability of such scenarios call for decentralized self-* approaches. It present in its paper a framework for the design and evaluation of self-adaptive collaborative task execution strategies in volunteer clouds. considering one factor: Memory. 9

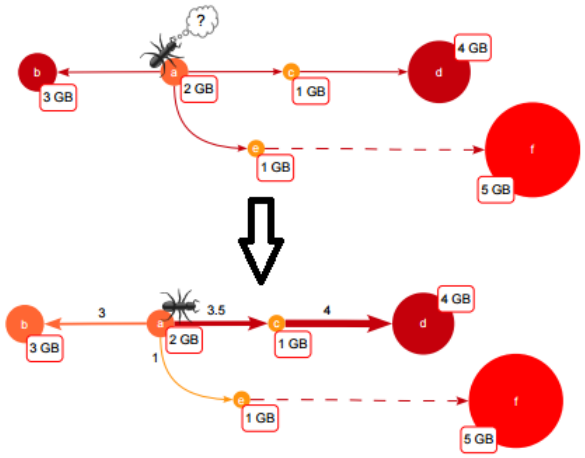

Fig. 9: Memory field supporting the ants quest

Considering three factors: Memory, CPU, distance. 10 Based on Ant Colony Optimization (ACO), to have the best result.

we can see the IAAS always provides the resource elasticity, but it is lack of the metrics for the elasticity itself: [4] propose a reliable metrics for quantifying the timing aspects and accuracy of elasticity. Based on these metrics, it also propose a novel approach for benchmarking the elasticity of Infrastructure-as-a-Service (IaaS) cloud platforms independent of the performance exhibited by the provisioned underlying resources. 11

Some researches are focus on the performance tuning, e.g. [3] said: it proposes an architecture for a self-adaptive prediction suite using an autonomic system approach. This suite can choose the most suitable prediction technique based on the performance pattern, which leads to more accurate prediction results, The hypothesis in this research is that prediction accuracy of auto-scaling systems can be increased by choosing an appropriate time-series prediction algorithm based on the performance pattern over time.

3.2.2 SAAS

Actually the SAAS is the most mature one among all the cloud types. It is directly faces the end user, always provide a login page, the end user login using their identical

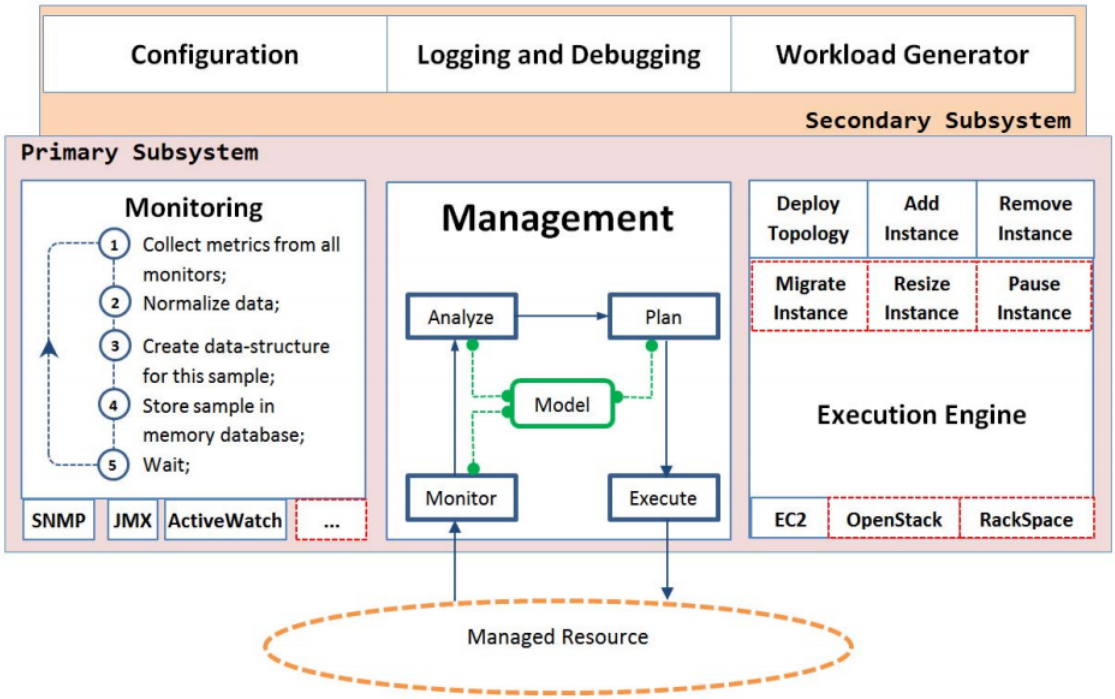

Fig. 8: The Hogna Frameworks Architecture

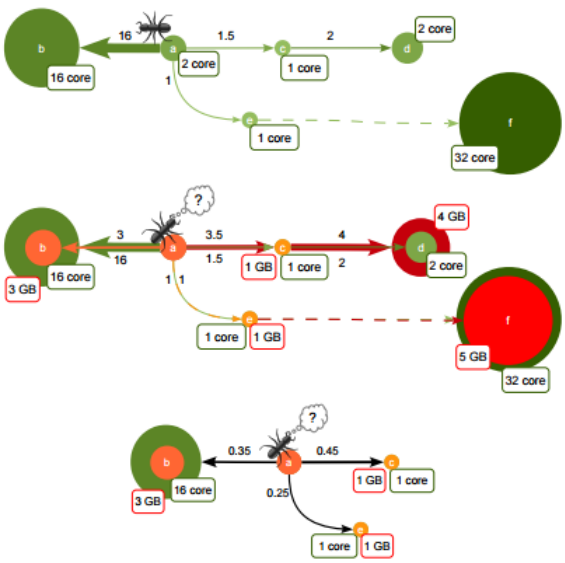

Fig. 10: Memory field overlap with CPU

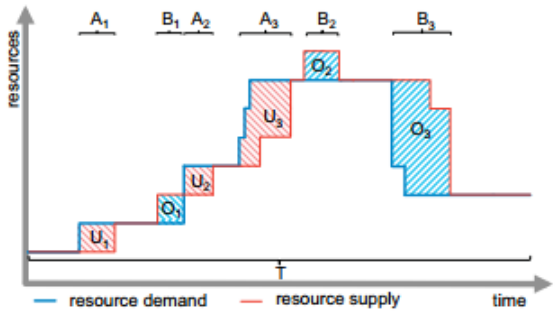

Fig. 11: Illustrating example for accuracy (Under, Over) and timing (A, B) metrics

id, then will get the system or functionality they want for their purpose. So this is the most convenient way of cloud providing, but also the least confidential way for the customer, e.g. the band system never running this way.

[21] propose a novel, dynamic and adaptive approach for implementing design diversity in the cloud market. The approach uses portfolio theory to construct a diversified portfolio of web service instances, which are traded from multiple/different cloud providers. Following will be considered in selection:

1) Availability. Availability Ai of a web service represents the probability that that web service is accessible.
2) Execution time. Execution time Ei measures the delay in seconds from the moment the web service is requested to the moment a reply is received.
3) Security. Security of web service Fi represents the level of security provision of the web service.

The weighted average profit comparing to the weighted average risk have the best result.

$$E_p = \sum_{i=1}^{m} w_i * a_i. \quad (1)$$

E represent the expect profit, w is the weight, a is the assets(service instance). The same equation as risks.

there are also rule based cloud resource elasticity for application on cloud [23]

3.2.3 PAAS

PAAS is the most immature technology comparing to IAAS and SAAS, as it is started just a few year ago, its progress is not that positive as IAAS. Not only because the cunning ”customer” (developer across various levels), but also it covers a wide range of IT server technology, including almost all the development language and container runtime, and the corresponding automatic testing, deploy, monitoring, bug fixing together with regression and move to production.

The Runtime and DevOps are two major part of PAAS system, so we can see the adaptive logic based on these two aspect.

Runtime:

[13] illustrate a method for eliciting, evaluating and ranking control points for web applications deployed in cloud environments. The proposed method consists of several phases that take high-level stakeholders adaptation goals and transform them into lower level MAPE-K loop control points: 12

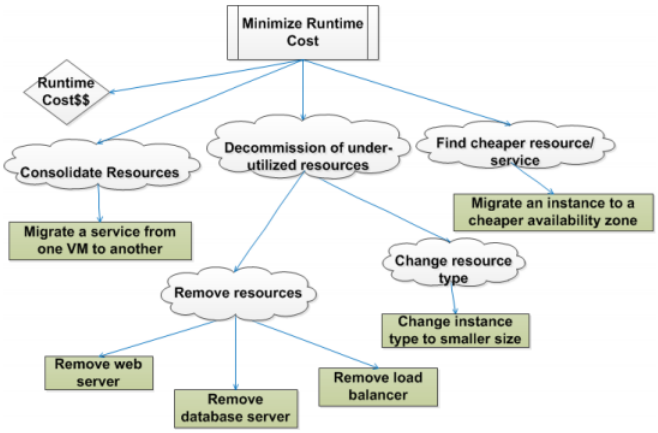

Fig. 12: Control Point Model for Runtime Cloud Cost

Show the elicited control points for meeting low cloud cost adaptation goals in PAAS cloud and application respectively. Therefore, these control points can be treated as catalogues for reuse.

DevOps:

cloud computing offers a large number of benefits, such as lower capital costs and a highly agile environment. Yet, the development of software engineering practices has not kept pace with this change.

DevOps blurs the traditional boundaries between developers and IT by ideally bringing together development and operations to achieve a common goal. ”Writing the code” and ”making them work on a platform” are the kind of works done by different types of people who traditionally worked separately and implemented entirely different methodologies and workflows. Successful DevOps requires a change in mindset C all parties involved need to share responsibility for the success or the failure of the product.

DevOps is the most challenge part in PAAS platform, which have to provide the development support and operational support, it has to support offline plugin for development of its PAAS platform, or to support online IDE-like stuff. besides, it also have to combine all the software engineer life cycle activities, such as debugging, automatic testing, automatic deployment, automatic regression and automatic move to production, which means move to runtime for release.

There will a lot of problems in the whole software development life cycle, as PAAS is not well support the traditional local development environment support, the application architect and its related development language and tools has to accommodate with what the PAAS platform provided. e.g. Anticipate is very important, we always do not have well debug environment in DevOps platform, so we have to anticipant the potential functional defect and performance issue in order to prepare log and metrics such as performance benchmarks for production environment analysis instead of directly debugging on it [29].

CloudWave a smart middle for Devops in clouds, aiming to utilize the principles of DevOps to create an execution analytics cloud infrastructure where, through the use of programmable monitoring and online data abstraction, much more relevant information for the optimization of the ecosystem is obtained. [45] The approach dynamically adapts cloud services to their environment, resulting in improved service quality and optimized resource use. This is supported with an enhanced cloud monitoring that provides holistic analytics of IaaS and SaaS layer services running on the cloud, leading to CloudWaves innovative, automated adaptation of the infrastructure and application, as well as enabling DevOps-like data and interfaces for the developer. 13

Aiming to provide the means for efficient collaboration between development and operations personnel, the DevOps paradigm is backed by an increasingly growing collection of tools and reusable artifacts for application management. Continuous delivery pipelines are established based on these building blocks by implementing fully automated, end-to-end application delivery processes, which significantly shorten release cycles to reduce risks and costs as well as gaining a critical competitive advantage. Diverse application environments need to be managed along the pipeline such as development, build, test, and production environments.

[47] address the need for systematically specifying and maintaining diverse application environment topologies enriched with environment-specific requirements in order to implement continuous delivery pipelines. Systematically handle and resolve dynamic development requirements to establish suitable application environments (development, test, production, etc.) as the key building blocks of continuous delivery pipelines as below.

For specific application: 14

Across different applications: 15

3.2.4 MCC

The main benefits of cloud computing is reducing downtime and wasted expenditure for servers and other computer equipment. A given company is required to purchase

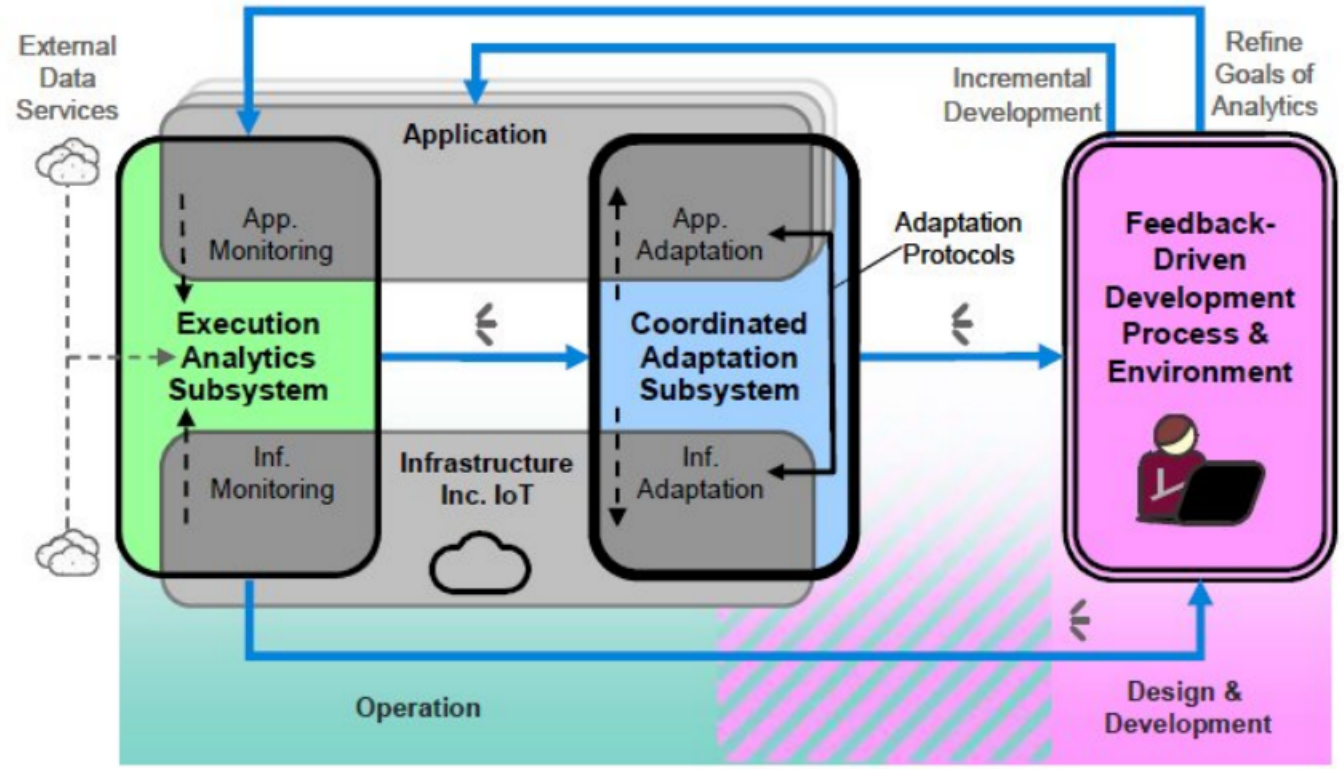

Fig. 13: CloudWave paradigm
monitoring: application data and infrastructure info monitoring
consolidating: application and infrastructure resource negotiating and consolidating
feedback loop: exploit all relevant runtime consolidating data together with operational feedback data
reconfigure: refine code and refine goal and metrics based on feedback
next loop with automatic feedback

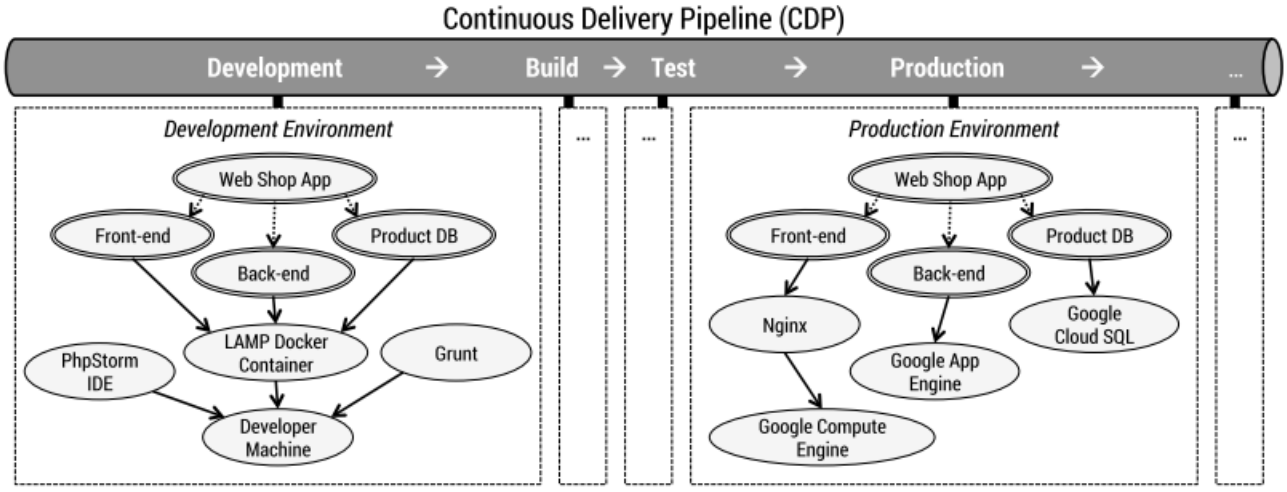

Fig. 14: continuous delivery pipeline (CDP) for the specific web shop application

the minimum amount of hardware necessary to handle the maximum points of stress on their system. To have the unnecessary maximum resource always leads to wasted money. For example, taobao.com, the normal access rate is less then 10 percent of its 11.11.

In the case of mobile cloud computing an additional significant benefit is brought to the table. Many mobile devices have significant constraints imposed upon them because

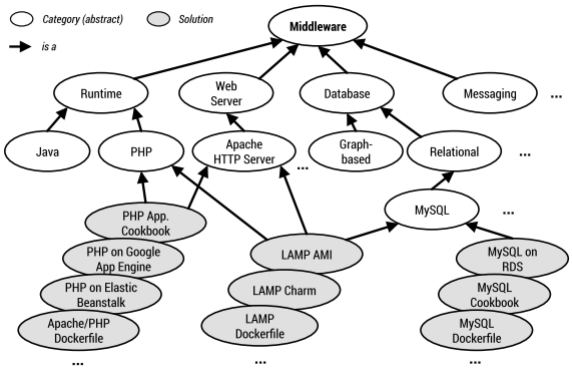

Fig. 15: across applications
LAMP: Linux + Apache + Mysql + PHP

of the importance and desirability of smaller sizes, lower weights, longer battery life and other features. This often severely constrains hardware and software development for these devices. Cloud computing allows devices to avoid these constraints by letting the more resource intensive tasks be performed on systems without these constraints and having the results sent to the device. Thus, cloud computing for mobile devices is a very appealing and potentially lucrative trend.

Mobile Cloud Computing (MCC) is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. The ultimate goal of MCC is to enable execution of rich mobile applications on a plethora of mobile devices, with a rich user experience. MCC provides business opportunities for mobile network operators as well as cloud providers. More comprehensively, MCC can be defined as ”a rich mobile computing technology that leverages uni?ed elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle.”

[2] presents the cost and energy aware service provisioning scheme for mobile client in mobile cloud, which includes two-stage optimization process. In the first-stage optimization process, the mobile cloud user gives the unique optimal payment to the cloud provider under the cost and energy constraint and optimizes its benefit. In the second-stage optimization process, mobile cloud provider runs multiple VMs to execute the jobs for mobile cloud users; the cloud providers also need to maximize the revenue.

The [27] focus on Improving the Energy Efficiency of Mobile Applications using adaptive Cloud Offloading.

But as the impact of unreliable network connections on mobile offloading has been experimentally confirmed. The execution of offloading tasks may suffer from long delays or even sometimes from failures. The [25] introduce an automated restart scheme. It aims at completing the job using restart with offloading. When the offloading task fails, the client may retry offloading or restart the task using the resources in the local device instead of those in the Cloud. 16 Here is the approaches: A) No restart, B) Infinite offloading

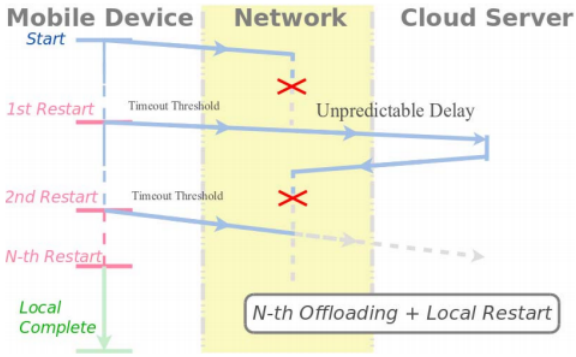

Fig. 1. Process of offloading with and without restart

Fig. 16: Process of offloading with and without restart

restart C) Only local restart n = 1, D) One offloading restart + one local restart n = 2, E) Two offloading restarts + one local restart n = 3. etc... Pick the one with best performance, for corresponding tasks, and tune it using historical data.

The [49] also use a genetic algorithm to automatically generate optimal application configurations, based on feature model, at runtime. This optimizes the configuration of the system at runtime according to the available resources. The approach does not entail excessive overhead, and helps the app coping with the resource constrained environment and optimizing its performance.

3.2.5 *Hybrid*

In this section, we'll share the adaptive research across the different cloud system. [20] illustrate a MonPaaS Architecture, it presents a novel monitoring architecture addressed to the cloud provider and the cloud consumers. the architecture offers a Monitoring Platform-as-a-Service to each cloud consumer that allows to customize the monitoring metrics. The cloud provider sees a complete overview of the infrastructure whereas the cloud consumer sees automatically her cloud resources and can define other resources or services to be monitored, refer to 17

it is based on openstack(IAAS), but implement as a extension of a PAAS system. The MonPaaS working together with the cloud controller via the message bus. it present a novel monitoring Platform-as-a-Service to enable both cloud provider and cloud consumer to monitor the cloud infrastructure resources and services.

4 A NEW IDEA ON SACS

The survey of approaches showed that the inclusion of context in most approaches is not sufficient. Whereas most approaches monitor the context, explicit alteration of context is not included in many approaches and the environment remains uncontrollable for the adaptation logic. As the context can not be well controlled.

A SACS is composed of managed resources and the adaptation logic. The surrounding context is used as input for the analyzing process. The context is not controlled by the adaptation logic of the SACS, e.g. it is not included

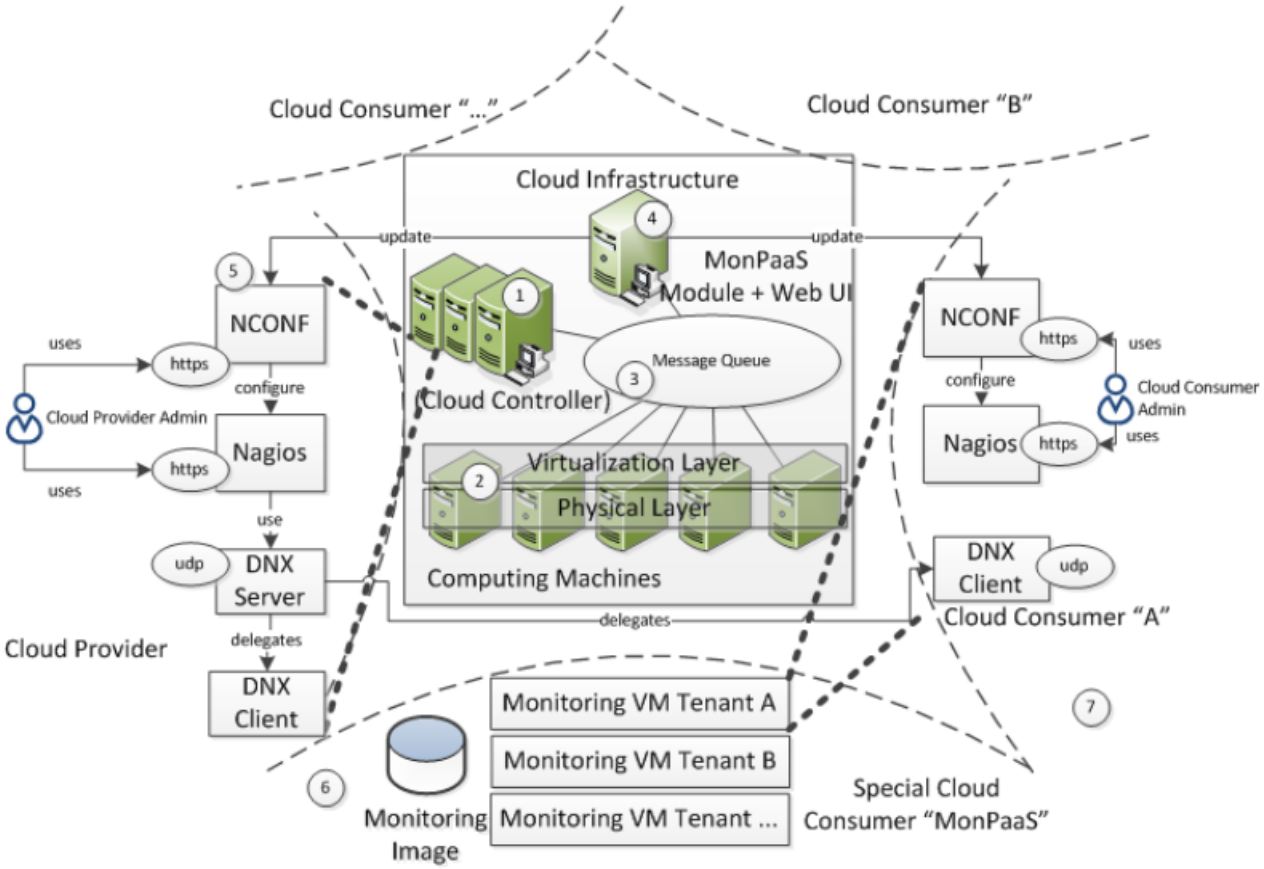

Fig. 17: MonPaaS Architecture: Nagios for monitoring, NConf is UI, DNX is for deployment of Nagios

in the planning activity. Nor the context's relationship is not considered by previous research, e.g. the location of the person maybe illustrate the possible situation(going to subway), and which is related to possible network situation, and the time e.g. 18:00 of Friday together with the role of the user may be inferred to the preference of the user in mobile usage of application.

According to this situation, we plan to take the advantage of the big data technology in order to predict the context change of from the monitoring information, we use three dimensions to make the adaptation of the SACS system, please refer to 18

It process its adaptive of SACS(cloud-cloudlet-mobile) based on three dimensions: Time, Location and Person(character). Following below MCC rules:

- weighted average response time of all users
- constraints: energy consumption(free am, critical p-m)
- Service-oriented approaches
- Programming Logic
- Hybrid

5 Conclusion

In this paper, we presented a taxonomy and a survey for self-adaptation cloud system, and a new idea on SACS. Based on literature research and combination of existing surveys, the taxonomy describes the SACS in different perspectives. The adaptation logic, which controls the adaptation, must be appropriately designed. Developers need to define the approach, adaptation decision criteria, and degree of decentralization, as well as the integration of the self-adaptation dimensions for monitoring and reasoning. For building SACS, different approaches can be found in literature. We classified the approaches in cloud perspectives: IAAS, PAAS, SAAS, MCC, and Hybrid. These categories can overlap. The new SACS idea introduce a three-dimension approach for optimize the SACS with more accurate and reasonable means.

Appendix A
TODO

Appendix one text goes here.

Appendix B
TODO2

Appendix two text goes here.

Acknowledgments

We would like to thank...

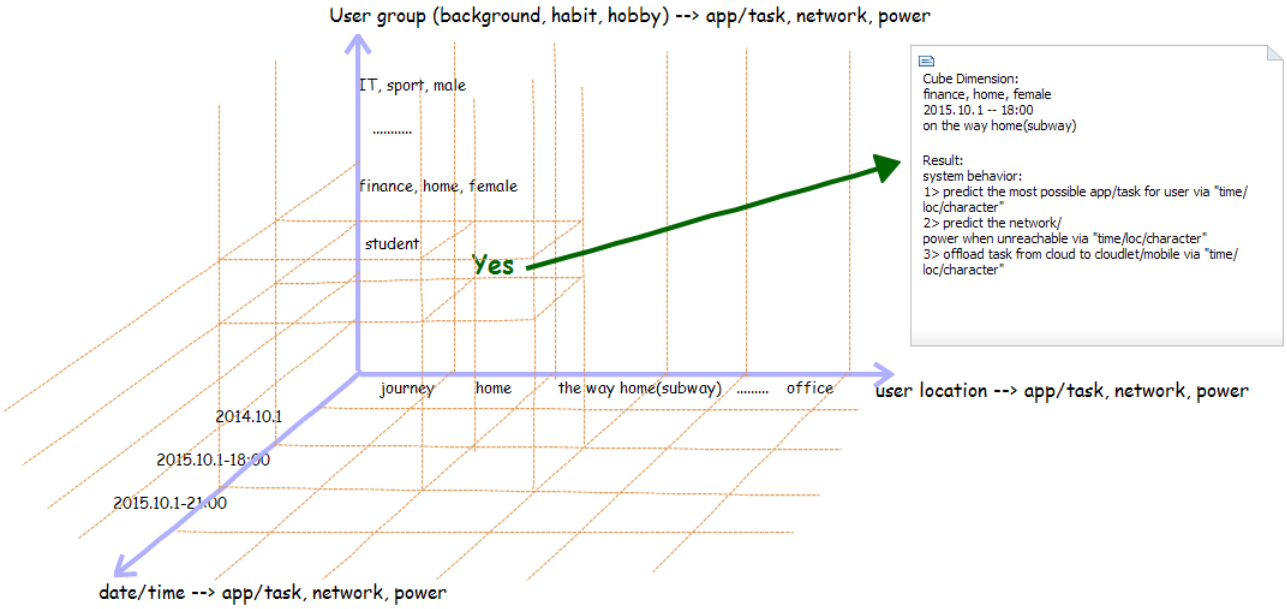

Fig. 18: Goal: Maximize the user satisfaction based on time/loc/characteristic

REFERENCES

[1] Bihuan Chen, Xin Peng, Yijun Yuz, Bashar Nuseibehz, and Wenyun Zhao, Self-Adaptation through Incremental Generative Model Transformations at Runtime, ICSE 2014.

[2] Woojoong Kim, Dong-Ki Kang, Seong-Hwan Kim, Chan-Hyun Youn: Cost Adaptive VM Management for Scientific Workflow Application in Mobile Cloud. MONET (2015)

[3] Ali Yadavar Nikravesh, Samuel A. Ajila, Chung-Horng Lung, Towards an Autonomic Auto-Scaling Prediction System for Cloud Resource Provisioning, ICSE 2015.

[4] Nikolas Roman Herbst, Samuel Kounev, Andreas Weber, BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments, ICSE 2015.

[5] Lorena Casta?eda, Norha M. Villegas, Hausi A. Mller, Self-Adaptive Applications: On the Development of Personalized Web-Tasking Systems, ICSE 2015.

[6] Pierre Laperdrix, Walter Rudametkin, Benoit Baudry, Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification, ICSE 2015.

[7] Luciano Baresi, Clment Quinton, Politecnico di Milano, Dipartimento di Elettronica, Dynamically Evolving the Structural Variability of Dynamic Software Product Lines, ICSE 2015.

[8] Nikos Dimokas, Kostas Kalogirou, Pavlos Spanidis, Evangelos Bekiaris: Building adaptive user interface using cloud computing. CSCESM 2015

[9] Cornel Barna, Hamoun Ghanbari, Marin Litoiu and Mark Shtern, Hogna: A Platform for Self-Adaptive Applications in Cloud Environments, ICSE 2015.

[10] Andreas Bergen, Nina Taherimakhsousi, Hausi A. Muller, Adaptive Management of Energy Consumption using Adaptive Runtime Models, ICSE 2015.

[11] Mark Harman, Yue Jia, William B. Langdon, Justyna Petke, Iman Hemati Moghadam, Shin Yoo, and Fan Wu, Genetic Improvement for Adaptive Software Engineering (Keynote), ICSE 2014.

[12] Nenad Medvidovic, Adapting Our View of Software Adaptation: An Architectural Perspective, ICSE 2014.

[13] Parisa Zoghi, Mark Shtern, Marin Litoiu, Designing Search Based Adaptive Systems: A Quantitative Approach, ICSE 2014.

[14] Erik M. Fredericks, Byron DeVries, Betty H. C. Cheng, Towards Run-Time Adaptation of Test Cases for Self-Adaptive Systems in the Face of Uncertainty, ICSE 2014.

[15] Eric Yuan, Naeem Esfahani, Sam Malek, Automated Mining of Software Component Interactions for Self-Adaptation, ICSE 2014.

[16] Faisal Alrebeish, Rami Bahsoon: Implementing Design Diversity Using Portfolio Thinking to Dynamically and Adaptively Manage the Allocation of Web Services in the Cloud. IEEE T. Cloud Computing 3(3): 318-331 (2015)

[17] Liliana Pasquale1, Carlo Ghezzi2, Claudio Menghi2, Christos Tsigkanos2, Bashar Nuseibeh1, Topology Aware Adaptive Security, ICSE 2014.

[18] Lorena Casta?eda, Norha M. Villegas, Hausi A. Mller, Self-Adaptive Applications: On the Development of Personalized Web-Tasking Systems, ICSE 2014.

[19] Danilo F. Mendon?a, Raian Ali, Genaína N. Rodrigues, Modelling and Analysing Contextual Failures for Dependability Requirements, ICSE 2014.

[20] Jose M. Alcaraz Calero, Jaime Gutierrez: MonPaaS: An Adaptive Monitoring Platformas a Service for Cloud Computing Infrastructures and Services. IEEE T. Services Computing 8(1): 65-78 (2015)

[21] Faisal Alrebeish, Rami Bahsoon: Implementing Design Diversity Using Portfolio Thinking to Dynamically and Adaptively Manage the Allocation of Web Services in the Cloud. IEEE T. Cloud Computing 3(3): 318-331 (2015)

[22] Tao Chen, Rami Bahsoon, Symbiotic and Sensitivity-Aware Architecture for Globally- Optimal Benefit in Self-Adaptive Cloud, ICSE 2014.

[23] Pooyan Jamshidi, Aakash Ahmad, Claus Pahl, Autonomic Resource Provisioning for Cloud-Based Software, ICSE 2014.

[24] Stefano Sebastio, Michele Amoretti, Alberto Lluch Lafuente, A Computational Field Framework for Collaborative Task Execution in Volunteer Clouds, ICSE 2014.

[25] Qiushi Wang, Katinka Wolter: Automated Adaptive Restart for Accelerating Task Completion in Cloud Offloading Systems. ICAC 2015: 157-158

[26] Nikolas Roman Herbst, Samuel Kounev, Andreas Weber, Henning Groenda: BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments. SEAMS@ICSE 2015: 46-56

[27] Young-Woo Kwon, Eli Tilevich: Facilitating the Implementation of Adaptive Cloud Offloading to Improve the Energy Efficiency of Mobile Applications. MOBILESoft 2015: 94-104

[28] Ivan Dario Paez Anaya1, Viliam Simko2, Johann Bourcier1, Noël Plouzeau, A Prediction-Driven Adaptation Approach for Self-Adaptive Sensor Networks, ICSE 2014.

[29] Jan Waller, Nils Christian Ehmke, Wilhelm Hasselbring: Including Performance Benchmarks into Continuous Integration to Enable DevOps. ACM SIGSOFT Software Engineering Notes 40(2): 1-4 (2015)

[30] Danny Weyns, M. Usman Iftikhar, and Joakim S oderlund, Do External Feedback Loops Improve the Design of Self-Adaptive Systems A Controlled Experiment, ICSE 2013.
[31] Javier Camara, Pedro Correia, Rogerio de Lemosy, David Garlanz, Pedro Gomesx, Bradley Schmerlz, and Rafael Ventura, Evolving an Adaptive Industrial Software System to Use Architecture-Based Self-Adaptation, ICSE 2013.
[32] Konstantinos Angelopoulos1, V?tor E. Silva Souza, Jo?ao Pimentel, Requirements and Architectural Approaches to Adaptive Software Systems: A Comparative Study, ICSE 2013.
[33] Alessio GambiCy, Daniel MoldovanC, Georgiana CopilC, Hong-Linh TruongC, and Schahram Dustdar, On Estimating Actuation Delays in Elastic Computing Systems, ICSE 2013.
[34] Siqian Gong, Beibei Yin, Wenlong Zhu, Kaiyuan Cai: An Adaptive Control Strategy for Resource Allocation of Service-Based Systems in Cloud Environment. QRS Companion 2015: 32-39
[35] Vincenzo Grassi, Moreno Marzolla, Raffaela Mirandola, QoS-Aware Fully Decentralized Service Assembly, ICSE 2013.
[36] Fangming Liu, Peng Shu, John C. S. Lui: AppATP: An Energy Conserving Adaptive Mobile-Cloud Transmission Protocol. IEEE Trans. Computers 64(11): 3051-3063 (2015)
[37] Gustavo G. Pascual, Monica Pinto, and Lidia Fuentes, Run-Time Adaptation of Mobile Applications using Genetic Algorithms, ICSE 2013.
[38] Matthew A. McCarthy, Lorraine M. Herger, Shakil M. Khan, Brian M. Belgodere: Composable DevOps: Automated Ontology Based DevOps Maturity Analysis. SCC 2015: 600-607
[39] Genci Tallabaci, V?tor E. Silva Souza, Engineering Adaptation with Zanshin: An Experience Report, ICSE 2013.
[40] Paulo Casanova, David Garlan, Bradley Schmerl, Rui Abreu, Diagnosing Architectural Run-Time Failures, ICSE 2013.
[41] Yousif Al Ridhawi, Ahmed Karmouch: Decentralized Plan-Free Semantic-Based Service Composition in Mobile Networks. IEEE T. Services Computing 8(1): 17-31 (2015)
[42] Wei-Chih Huang and William J. Knottenbelt, Self-Adaptive Containers: Building Resource-Efficient Applications with Low Programmer Overhead, ICSE 2013.
[43] Antinisca Di Marco, Paola Inverardi, Romina Spalazzese, Synthesizing Self-Adaptive Connectors Meeting Functional and Performance Concerns, ICSE 2013.
[44] Tzu-Chi Huang, Kuo-Chih Chu, Wei-Tsong Lee, Yu-Sheng Ho: Adaptive Combiner for MapReduce on cloud computing. Cluster Computing 17(4): 1231-1252 (2014)
[45] Dario Bruneo, Thomas Fritz, Sharon Keidar-Barner, Philipp Leitner, Francesco Longo, Clarissa Cassales Marquezan, Andreas Metzger, Klaus Pohl, Antonio Puliafito, Danny Raz, Andreas Roth, Eliot Salant, Itai Segall, Massimo Villari, Yaron Wolfsthal, Chris Woods: CloudWave: Where adaptive cloud management meets DevOps. ISCC 2014: 1-6
[46] Yue Tan, Cathy H. Xia: An Adaptive Learning Approach for Efficient Resource Provisioning in Cloud Services. SIGMETRICS Performance Evaluation Review 42(4): 3-11 (2015)
[47] Johannes Wettinger, Vasilios Andrikopoulos, Frank Leymann: Enabling DevOps Collaboration and Continuous Delivery Using Diverse Application Environments. OTM Conferences 2015: 348-358
[48] Mauro Andreolini, Michele Colajanni, Marcello Pietri, Stefania Tosi: Adaptive, scalable and reliable monitoring of big data on clouds. J. Parallel Distrib. Comput. 79: 67-79 (2015)
[49] G. G. Pascual, M. Pinto, and L. Fuentes. Run-time adaptation of mobile applications using genetic algorithms. SEAMS 2013: 73-82.
[50] Yu Wu, Chuan Wu, Member, IEEE, ACM, Bo Li, Fellow, IEEE, Linquan Zhang, Zongpeng Li, and Francis C. M. Lau, Senior Member, IEEE, Scaling Social Media Applications Into Geo-Distributed Clouds, IEEE Transactions 2014.
[51] Asif Qumer Gill, Adaptive Cloud Enterprise Architecture, World Scientific 2015.
[52] Jin Gou, Wang-Ping Guo, Feng Hou, Cheng Wang, Yi-Qiao Cai: Adaptive differential evolution with directional strategy and cloud model. Appl. Intell. 42(2): 369-388 (2015)
[53] Xiang T. R. Kong, Ji Fang, Hao Luo, George Q. Huang, Cloud-enabled real-time platform for adaptive planning and control in auction logistics center. Computers and Industrial Engineering 2015
[54] Antonio Brogi, Jos Carrasco, Javier Cubo, Elisabetta Di Nitto, Francisco Durn, Michela Fazzolari, Ahmad Ibrahim, Ernesto Pimentel, Jacopo Soldani, PengWei Wang, Francesco D'Andria: Adaptive management of applications across multiple clouds: The SeaClouds Approach. CLEI Electron. J. 18(1) (2015)
[55] Bing Yu, Yanni Han, Hanning Yuan, Xu Zhou, Zhen Xu: A cost-effective scheme supporting adaptive service migration in cloud data center. Frontiers of Computer Science 9(6): 875-886 (2015)
[56] Jong-Moon Chung, Yong-Suk Park, Jong-Hong Park, Hyoung Jun Cho: Adaptive Cloud Offloading of Augmented Reality Applications on Smart Devices for Minimum Energy Consumption. TIIS 9(8): 3090-3102 (2015)
[57] D. Garlan, S.-W. Cheng, A.-C. Huang, B.R. Schmerl, P. Steenkiste, Rainbow: architecture-based self-adaptation with reusable infrastructure, IEEE Comput. 37 (10) (2004) 46C54.
[58] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, K. Pohl, A journey to highly dynamic, self-adaptive service-based applications, Autom. Softw. Eng. 15 (3C4) (2008) 313C341
[59] D. Menasce, H. Gomaa, S. Malek, J.P. Sousa, SASSY: a framework for self-architecting service-oriented systems, IEEE Softw. 28 (6) (2011) 78C85.
[60] Andreas Bergen, Ya?g?z Onat Yaz?r, Hausi A. Muller, Yvonne Coady, RPC Automation: Making Legacy Code Relevant, ICSE 2013.
[61] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, Aspect-oriented programming, in: ECOOP97łObject-Oriented Programming, in: LNCS, vol. 1241, Springer, 1997, pp. 220C242.
[62] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W.G. Griswold, An overview of AspectJ, in: ECOOP 2001łObject-Oriented Programming, in: LNCS, vol. 2072, Springer, 2001, pp. 327C354.
[63] R. Haesevoets, E. Truyen, T. Holvoet, W. Joosen, Weaving the fabric of the control loop through aspects, in: Self-Organizing Architectures, in: LNCS, vol. 6090, Springer, 2009, pp. 38C65.