Chapter 1

Introduction

1.1 Problem Statement

Cloud computing is a computing model offers a network of servers to their clients in a on-demand fashion. From NIST's definition [28], *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."* Basically, a web-based application provider can deploy applications on Cloud servers through a console as if these servers are at local. As applications start providing services, Cloud will automatically adjusts the capacity of servers in cope with fluctuating requests while the provider only focuses on application development.

Cloud computing has mainly three stakeholders [19] (see Figure 1.1): Cloud provider, Cloud user and End user. *Cloud providers* build data centers, provide maintenance and resource management on the hardware infrastructure. Their income come from Cloud users' rental of servers and Cloud providers' expense include upfront investment, energy consumption, and maintenance. *Cloud users* deploy their applications or services in Cloud. They make profit from selling their services to End users and pay the rental of resources to Cloud providers. *End users* request and pay for the applications.

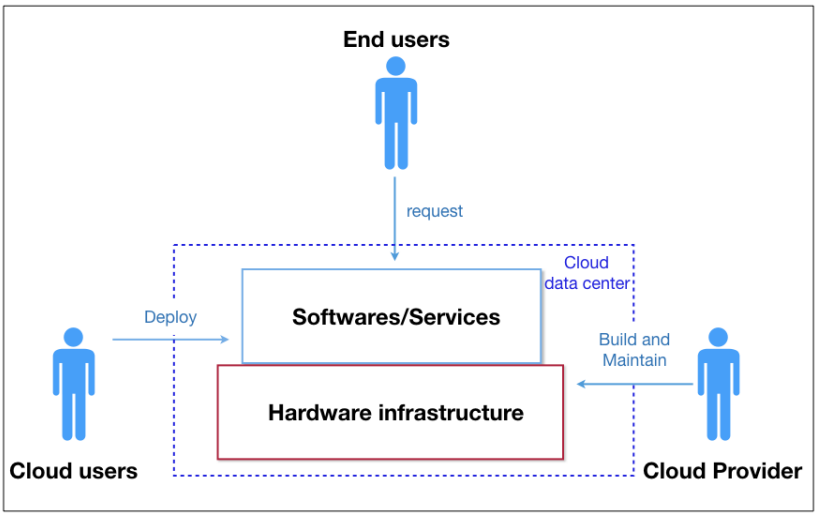

Figure 1.1: Stakeholders of Cloud computing

Cloud computing has completely reformed the software industry [7] by providing three

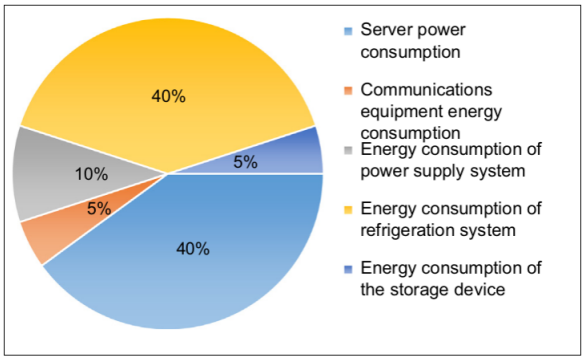

Figure 1.2: Energy consumption distribution of data centers [35]

major benefits to web-based software or web service providers. First, service providers do not need upfront investment in hardwares (e.g servers and networking devices) and pay for hardwares' maintenance. Second, service providers will not worried about the limited resources will obstruct the performance of their services when unexpected high demand occurs. The elastic nature of cloud can dynamic allocate and release resources for a service. In addition, software providers can pay as much as the resource under a *pay-as-you-go* policy. Third, service providers can publish and update their applications at any location as long as there is an Internet connection. These advantages allow anyone or organization to deploy their softwares on Cloud in a reasonable price.

From Cloud providers' perspective, they are trying to make the most profit on data centers. On one hand, cloud providers are trying to improve the quality of Cloud service to attract more service providers. On the other hand, they want to cut enormous energy consumption - as much as 25,000 households [23] - to lower the expense. Energy consumption is the major concern of Cloud providers. It is derived from several parts as illustrated in Figure 1.2. Regardless the energy consumption of refrigeration system (or cooling system), the majority are from servers. According to Hameed et al [16], servers are far from energy-efficient. The main reason for the wastage is that the energy consumption of servers remains high even when the utilization are low (see Figure 1.4). Therefore, a concept of *energy proportional computing* [3] raised to address the disproportionate between utilization and energy consumption. This leads to using virtualization technology to achieve server consolidation.

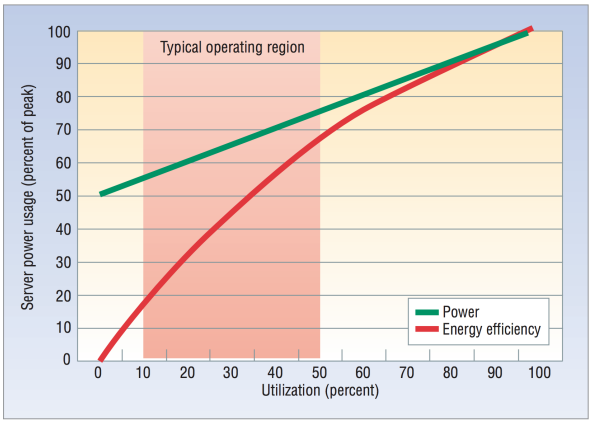

Figure 1.3: Disproportionate between utilization and energy consumption [3]

Virtualization [44] partitions a physical machine's resources (e.g. CPU, memory and

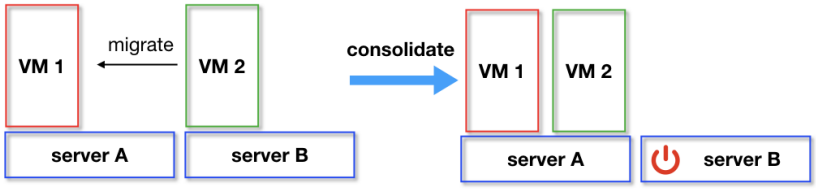

Figure 1.4: Server Consolidation by migrating VM 2 to server A [3]

disk) into several isolated units called virtual machines (VMs) where each VM allows an operating system running on them. This technology rooted back in the 1960s' and was originally invented to enable isolated software testing. VMs can provide good isolation which means applications running in co-located VMs within the same server do not interfere each other [39]. Soon, people realized it can be a way to improve the utilization of hardware resources: With each application deployed in a VM, a server can run multiple applications. Later after, a dynamic migration of VM was invented, which compresses and transfers a VM from one server to another. This technique allows resource management at runtime which inspires the strategy of server consolidation.

Server consolidation [51] resolves the low utilization problem by gathering applications or VMs into a fewer number of physical machines (PMs) (see Figure 1.4), so that the resource utilization of PMs are maintained at a high level. It dramatically improves hardware utilization and lowers server and cooling energy consumption.

Server consolidation is the core functionality involving in all Cloud resource management processes. Cloud resource management can be roughly separated into three phases [42, 29] (see Figure 1.5): application/VM Initialization, Dynamic resource management, and Static consolidation.

1. *Application/VM initialization* takes a list of incoming requests of applications or VMs as the input, based on their requested resource sizes, determines their allocation in servers. This phase can be seen as a static consolidation, where the requested VMs are consolidated into a minimum number of servers.

2. *Dynamic resource management* adjusts the allocation based on servers' states at any time. Normally, there are three scenarios when the dynamic management is conducted. First, an application has intensive requests causes the server overloading (Figure 1.5). In order to prevent the Quality of Service (QoS) dropping, an application is migrated to another server. This is called hot-spot mitigation [29]. Second, a server in a low utilization state should be emptied. Therefore, all the VMs inside are migrated to other active servers. This is a dynamic consolidation. Third, servers are discrepancy in utilization level. An adjustment is to migrate one or more VMs from high utilized servers to low ones. This is called load balancing.

3. A *static server consolidation* is conducted to improve the global energy efficiency at a certain time point, e.g. a fixed time interval. This is because Cloud data center has a highly dynamic nature with continuous arriving and releasing of VMs. Therefore, after the initial allocation, the energy efficiency keeps dropping. In comparison with initialization, static consolidation needs to consider the previous allocation in order to reduce the number of migration, for migration is a very expensive operation. In comparison with dynamic consolidation, static consolidation takes a set of VMs as input instead of one. Therefore, it is time consuming and often treated as a static problem.

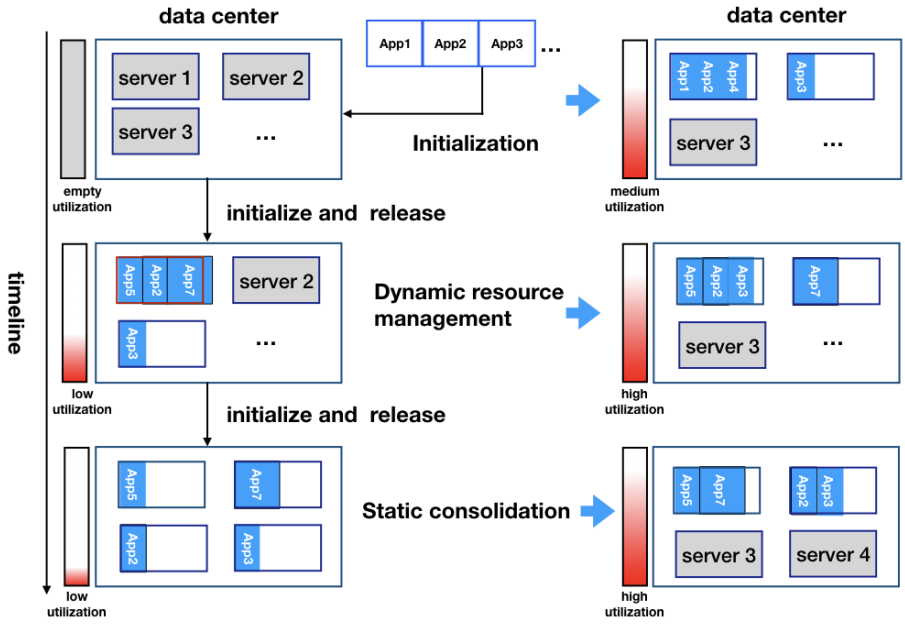

Figure 1.5: Cloud data center resource management involves three steps: initialization, dynamic resource management, and static consolidation. Grey means closed servers.

By the nature of Cloud resource management, server consolidation techniques can also be categories into static and dynamic methods [47, 45]. Static consolidation is a time consuming process which is often conducted off-line in a periodical fashion. It provides a global optimization to the data center. Dynamic consolidation adjusts servers in real time. It often allocates one VM at a time. Therefore, it can be executed quickly and often provides a local optimization to the data center.

Despite the usefulness of server consolidation, it is a difficult task. Traditional server consolidation is conducted manually [10] with vast data analysis and interviews with application owners. It is fraught and extremely difficult to reach a global optima state. Later on, server consolidation is often considered as a global optimization problem where its goal is to minimize the overall energy consumption. It is often modeled as a bin-packing problem [25] which is a well-known NP-hard problem meaning it is unlikely to find an optimal solution of a large problem.

In static methods, deterministic methods such as Integer Linear Programming [41] and Mixed Integer Programming [46] are often considered. However, it is well-known that they are very time-consuming for a large scale problem. More research proposed heuristic methods to approximate the optimal solution such as First Fit Decreasing (FFD) [32], Best Fit Decreasing (BFD) [4]. Manually designed heuristics are designed to tackle the special requirements such as a bin-item incomplete scenario [14] and Multi-tier Applications [22, 24]. Although these greedy-based heuristics can quickly solve the consolidation problem, as Mann's research [25] shown, server consolidation is a lot more harder than bin-packing problem because of multi-dimension, many constraints. Therefore, a simple greedy-based heuristic (e.g FFD) leads to a bad performance.

Evolutionary Computation (EC) is commonly used to solve combinatorial optimization

problem [15], therefore, it is particular useful in solving the static consolidation problem. Many EC techniques including Genetic Algorithm (GA) [48], Ant Colony Optimization (ACO) [13, 27], Particle Swarm Optimization (PSO) [20] have been used in solving this problem. EC algorithms show their advantages in the following aspects. Firstly, EC algorithms are good at solving multi-objective problems because of their population-based nature. And static consolidation problem often involves two or more objectives (e.g energy efficiency and migration cost). Secondly, they can provide near-optimal solutions within a reasonable amount of time.

Dynamic consolidation problem requires fast decision-making and global optimization. Previous most research proposed human designed greedy-based dispatching rules or heuristics such as a First-Fit-based approach [5], Modified Best Fit Decreasing [4], and a two-stage heuristic [52]. One of the major problem for human designed heuristics is that if any inherent component gets changes, then the designed heuristic may not work as it was expected [40]. EC algorithms are also seldom considered in this scenario because most EC methods need more time to search through solutions space.

In recent years, virtualization technology has evolved to allow finer granularity resource management. A recent development of Container technique [38] has drove the attention of both industrial and academia. Container is an operating system level of virtualization which means multiple containers can be installed in a same operating system (see Figure 1.6). Each container provides an isolated environment for an application. In short, a VM is partitioned into smaller manageable units. This new concept starts a new service model called Container as a Service (CaaS) [34]. CaaS brings advantages for both Cloud customers and providers. From customers' perspective, CaaS has advantages of both IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) but without their disadvantages. On one hand - similar to PaaS - it frees the customers' from low level resource management so that they can focus on application development. On the other hand - similar to IaaS - it allows customers to customize their software environment without being constrained by platforms. For Clouds provide, traditional IaaS leads to the low utilization of resources.

The reasons for low utilization of resources with IaaS derive from two mechanisms: the separated responsibilities of resource selection for applications and resource allocation; The fixed types of VMs, where each type of VM represents a certain amount of resources (e.g. CPU, RAM, and Storage).

- Firstly, because of separated responsibilities, customers must estimate the quantity of resources. They tend to reserve more resources for ensuring the QoS at the peak hours [9]. This causes the low utilization.
- Secondly, because of the fixed size of VM and the one-on-one mapping of applications and VMs (see Figure 1.6 left hand-side), specific applications consume unbalanced resources lead to vast amount of resource wastage [43]. For example, computation intensive tasks consume much more CPU than RAM; a fixed type of VM may provide much more RAM than it needs. This also causes wastage.

In contrast, CaaS solves above two problems at a time. It allows Cloud providers to manage both the resource selection for applications and resource allocation; It also enables VM-resizing and many-to-one mapping between applications to VMs (Figure 1.6 right hand-side). Hence, Cloud providers have a complete control of resources which may lead to a better utilization of resources. In addition, IaaS Cloud runs many redundant operating systems and hypervisors. CaaS eliminates these redundancies by a single operating system with multiple containers.

Currently, vast amount of server consolidation methods are mostly VM-based which can not be directly applied on this problem, because two-level of bin-packing problems interact

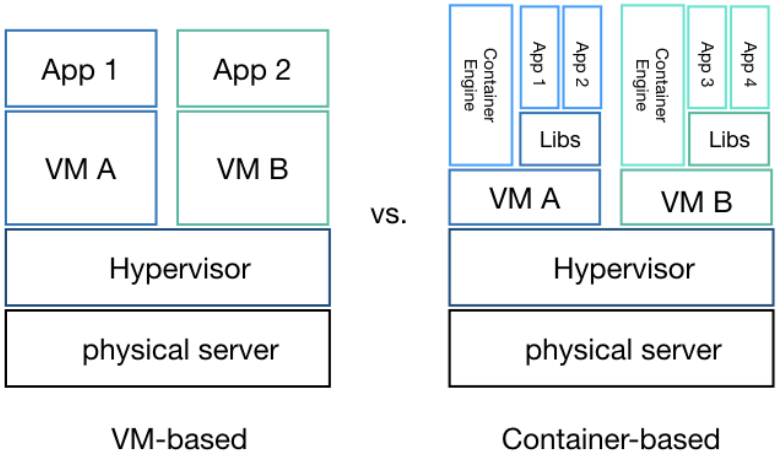

Figure 1.6: A comparison between VM-based and Container-based virtualization

with each other. Piraghaj [33] proposes a two-step procedure; it first maps tasks to VMs and then allocate containers to VMs. As Mann illustrated in [26], these two steps should be conducted simultaneously, otherwise it leads to local optimal. Other research [11, 17, 1] propose greedy-based heuristics on container allocation problem. They can be easily stuck at local optimal. This thesis, therefore, aims at providing an end-to-end solution for Container-based server consolidation which includes three stages: initialization, static container-based server consolidation and dynamic container placement.

1.2 Motivation

The container-based consolidation problem, similar to VM-based consolidation, can be seen as a continuous optimization procedure with three stages: initialization, off-line static joint allocation of container and VM, and on-line dynamic consolidation. Different stages have distinctive goals, therefore, they are considered as separated research questions. In this thesis, we aims at providing an end-to-end solution to all three problems. In addition, a scalability problem of static optimization is considered as an optional objective.

1. The initialization stage is first step in the continuous server consolidation. At this stage, a set of applications or containers is allocated to empty VMs and these VMs are allocated to physical servers. This seemingly two-step procedure is interconnected, therefore, should be conducted simultaneously. Previous research [19] focus on VM-based optimization and they cannot solve the two-level optimization problem.

   This stage will establish the fundamental concepts in studying the joint allocation of containers and VMs including new problem models: price and power model, new problem constraints, and optimization objectives. This study will be the first one that propose representations and solutions for the two-level initialization problem.

2. Server consolidation can be considered as static or dynamic problem [47]. A *static server consolidation* is conducted to improve the global energy efficiency at a certain time point, e.g. a fixed time interval. This is because Cloud data center has a highly dynamic nature with continuous arriving and releasing of VMs. Therefore, after the initial allocation, the energy efficiency keeps dropping. Similar with VM-based consolidation, migration cost is one of the key consideration. So the problem is considered

as a multi-objective problem with minimization of migration cost as well as keeping a good energy efficiency. Previous research only consider each consolidation as an independent process. Therefore, although in current consolidation, the migration is minimized. It may lead to more migration in the future. We are going to consider the robustness of consolidation and propose a novel time-series-aware server consolidation which takes the previous consolidations and the future consolidation into consideration.

3. Dynamic consolidation is another method to maintain a high energy efficiency. Unlike static consolidation is conducted periodically in a global scale, dynamic consolidation is applied on single VM or container at any time point. At large, data center system monitors all the states of servers for overloaded and underloaded servers. Once an overloaded server is detected, one of the VM or container running inside the server will be migrated to other machine so that the applications do not suffer from a performance degradation; for an underloaded server, all its applications will be moved to other servers so that it can be turned off. In summary, the main goal for dynamic consolidation is to optimize the global energy consumption as well as prevent overloading. In a container-based environment, it involves three steps .

   - *When to migrate?* refers to determine the time point that a physical server is overloaded.
   - *Which container to migrate?* refers to determine which container need to be migrated so that it optimize the global energy consumption.
   - *Where to migrate?* refers to determine which VM and host that a container is migrated to.

   Specifically, we focus on the third question: dynamic placement problem. To solve a dynamic placement with large number of variables, heuristics and dispatching rules are often used [36, 37, 12, 4]. In this scenario, a dispatching rule is considered as a function that determines the priorities of servers that a container can be placed. However, dynamic placement is much complex than bin-packing problem [25]. Because of its dynamic nature, human designed heuristics do not work well when the environment has changed [40]. Multi-objective genetic algorithm (GA) [48] has been applied. However, GA is too slow for dynamic problem.

   We intend to develop a hyper-heuristic method - Genetic Programming (GP) technique [2] or artificial immune system [18]- to learn from the best previous allocation and automatic evolve dispatching rules to solve this problem. GP has been applied in generating dispatching rules for bin-packing problem [6, 40] and other scheduling problems [31]. The results have shown promising results.

4. Cloud data center typically has hundreds of thousands servers and more. Large scale of static server consolidation has always been a challenge since it takes large amount of variables into consider. Many approaches have been proposed in the literature to resolve the problem. There are mainly two ways, both rely on distributed methods, hierarchical-based [21, 30] and agent-based management systems [49]. The major problem in agent-based systems is that agents rely on heavy communication to maintain a high-level utilization. Therefore, it causes heavy load in the networking. Hierarchical-based approaches are the predominate methods. In essence, these approaches are centralized methods where all the states of machines within its region are collected and analyzed. The major disadvantage of hierarchical-based approaches is that it only provides local solutions. In fact, it is infeasible and unnecessary to check

all the states of machines since the search space is too large and most machines do not need a change. This idea motivates a way to improving the effectiveness is to reduce the number of variables so that the search space is narrowed. In this thesis, we are going to investigate the way to eliminate the redundant information.

1.3 Research Goals

The overall goal of this thesis is to propose an end-to-end server consolidation approach that considers all three stages: Initialization, Off-line Static Joint Allocation of Container and VM, On-line Dynamic Container Placement Problem. In addition, the static allocation normally involves with large amount of variables which is particular difficult to optimize. We also going to propose a method to solve this problem. These approaches combine element of AI planning, to ensure the objectives and constraint fulfillment, and of Evolutionary Computation, to evolve a population of near-optimal solutions. The research aims to determine a flexible way in which planning and EC can be combined to allow the creation of solutions to solve server consolidation problems. As discussed in the previous section, the research goal can be achieved in the following objectives and sub-objectives.

1. The initialization Problem,
   Currently, most research focus on VM-based server consolidation technique. They often modeled this problem as a vector bin-packing problem [50]. Container adds an extra layer of abstraction on top of VM. The placement problem has become a two-step procedure, in the first step, containers are packed into VMs and then VMs are consolidated into physical machines. These two steps are inter-related to each other.

   (a) *Modeling*
   Previous VM-based models do not consider two-level allocation structure, therefore, our first sub problem is to propose a description of model for the initialization problem. In order to achieve this goal, we will first review the related models including VM-based placement models and bi-level optimization models. Furthermore, we are going to consider the differences and design the constraints and other characteristics.

   (b) *Representation*
   Based on this new model, we are going to discover a representation that suitable for this problem.

   (c) *New operators and searching mechanisms*
   In order to utilize Evolutionary Computation (EC) to solve this problem, we are going to discover searching mechanisms according to the nature of problem. In order to achieve this goal, we will design several new operators.

2. Off-line Static Joint Allocation of Container and VM Problem,
   A static allocation can be seen as a resource scheduling problem. A schedule is robust if it is able to absorb some degree of uncertainty in tasks duration while maintaining a stable solution [8]. Cloud resource management is a continuous process, after each static allocation, the system should be able to maintain a stable status with the least adjustment. The development of static allocation approach has three sub-objectives.

   (a) *Design a robustness measure*
   Previous studies only use simple measurement which counts the migration number between two static consolidation. This measurement aims at minimizing the

number of migration in a static placement process. It may cause more migration in the next consolidation. Therefore, it needs a time-series aware measure of the robustness of system. A data center should be both consolidated as well as robustness after consolidate. Therefore, in this objective, the first sub-problem we are going to solve is to propose a robustness measure.

(b) *Design an allocation method consider previous allocation*
Based on the robustness measure, we will first design an allocation method which takes previous allocation into account. It has two optimization objectives, maximize the robustness and also minimize the energy consolidation.

(c) *Design a time-series-aware allocation method*
Last but not the least, we will generalize the previous sub-objective to a more general one: design a time-series-aware allocation method which takes several allocation into consider.

3. On-line Dynamic Container Placement Problem with a GP approach,

   (a) Representation
   In order to utilize a hyper-heuristic method such as GP to solve the problem, the first step is to design a representation.

   (b) Construct Functional Set and Primitive Set for the problem
   As the basic component of a dispatching rule, primitive set contains the states of environment including: status of VMs, features of workloads. The functional set contains the operators which combines low level features.

   (c) Develop GP-based methods for evolving Dispatching rules

4. Large-scale Static Consolidation Problem

   (a) Propose a preprocessing method to eliminate variables
   Current static consolidation takes all servers into consider which will lead to a scalability problem. In this objective, we will propose a method that categorizes servers so that only a small number of servers are considered. This approach will dramatically reduce the search space. The potential approaches that can be applied in this task are various clustering methods.

1.4 Published Papers

During the initial stage of this research, some investigation was carried out on the model of container-based server consolidation.

1. Tan, B., Ma, H., Mei, Y. and Zhang, M., "A NSGA-II-based Approach for Web Service Resource Allocation On Cloud". *Proceedings of 2017 IEEE Congress on Evolutioanry Computation (CEC2017).* Donostia, Spain. 5-8 June, 2017.pp.

1.5 Organisation of Proposal

The remainder of the proposal is organised as follows: Chapter **??** provides a fundamental definition of the Container-based server consolidation problem and performs a literature review covering a range of works in this field; Chapter **??** discusses the preliminary work carried out to explore the techniques and EC-based techniques for the initialization problem;

Chapter **??** presents a plan detailing this projects intended contributions, a project timeline, and a thesis outline.

Bibliography

[1] ANSELMI, J., AMALDI, E., AND CREMONESI, P. Service Consolidation with End-to-End Response Time Constraints. *EUROMICRO-SEAA* (2008), 345–352.

[2] BANZHAF, W., NORDIN, P., KELLER, R. E., AND FRANCONE, F. D. Genetic programming: an introduction, 1998.

[3] BARROSO, L. A., AND HÖLZLE, U. The Case for Energy-Proportional Computing. *IEEE Computer 40*, 12 (2007), 33–37.

[4] BELOGLAZOV, A., ABAWAJY, J. H., AND BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Comp. Syst. 28*, 5 (2012), 755–768.

[5] BOBROFF, N., KOCHUT, A., AND BEATY, K. A. Dynamic Placement of Virtual Machines for Managing SLA Violations. *Integrated Network Management* (2007), 119–128.

[6] BURKE, E. K., HYDE, M. R., AND KENDALL, G. Evolving Bin Packing Heuristics with Genetic Programming. *PPSN 4193*, Chapter 87 (2006), 860–869.

[7] BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J., AND BRANDIC, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems 25*, 6 (June 2009), 599–616.

[8] CANON, L.-C., AND JEANNOT, E. Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments. *IEEE Transactions on Parallel and Distributed Systems 21*, 4 (2010), 532–546.

[9] CHAISIRI, S., LEE, B.-S., AND NIYATO, D. Optimization of Resource Provisioning Cost in Cloud Computing. *IEEE Trans. Services Computing 5*, 2 (2012), 164–177.

[10] CHEBIYYAM, M., MALVIYA, R., AND BOSE, S. K. Server consolidation: Leveraging the benefits of virtualization. *Perform or Perish* (2009).

[11] DONG, Z., ZHUANG, W., AND ROJAS-CESSA, R. Energy-aware scheduling schemes for cloud data centers on Google trace data. *OnlineGreenComm* (2014), 1–6.

[12] FORSMAN, M., GLAD, A., LUNDBERG, L., AND ILIE, D. Algorithms for automated live migration of virtual machines. *Journal of Systems and Software 101* (2015), 110–126.

[13] GAO, Y., GUAN, H., QI, Z., HOU, Y., AND LIU, L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Comput. Syst. Sci. 79*, 8 (2013), 1230–1242.

[14] GUPTA, R., BOSE, S. K., SUNDARRAJAN, S., CHEBIYAM, M., AND CHAKRABARTI, A. A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints. *IEEE SCC* (2008).

[15] GUZEK, M., BOUVRY, P., AND TALBI, E.-G. A Survey of Evolutionary Computation for Resource Management of Processing in Cloud Computing [Review Article]. *IEEE Computational Intelligence ... 10*, 2 (2015), 53–67.

[16] HAMEED, A., KHOSHKBARFOROUSHHA, A., RANJAN, R., JAYARAMAN, P. P., KOLODZIEJ, J., BALAJI, P., ZEADALLY, S., MALLUHI, Q. M., TZIRITAS, N., VISHNU, A., KHAN, S. U., AND ZOMAYA, A. Y. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing 98*, 7 (2016), 751–774.

[17] HINDMAN, B., KONWINSKI, A., ZAHARIA, M., GHODSI, A., JOSEPH, A. D., KATZ, R. H., SHENKER, S., AND STOICA, I. Mesos - A Platform for Fine-Grained Resource Sharing in the Data Center. *NSDI* (2011).

[18] HOFMEYR, S. A., AND FORREST, S. Architecture for an Artificial Immune System. *Evolutionary Computation 8*, 4 (2000), 443–473.

[19] JENNINGS, B., AND STADLER, R. Resource Management in Clouds: Survey and Research Challenges. *Journal of Network and Systems Management 23*, 3 (2015), 567–619.

[20] JEYARANI, R., NAGAVENI, N., AND RAM, R. V. Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. *Future Generation Comp. Syst. 28*, 5 (2012), 811–821.

[21] JUNG, G., HILTUNEN, M. A., JOSHI, K. R., SCHLICHTING, R. D., AND PU, C. Mistral - Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. *ICDCS* (2010), 62–73.

[22] JUNG, G., JOSHI, K. R., HILTUNEN, M. A., SCHLICHTING, R. D., AND PU, C. Generating Adaptation Policies for Multi-tier Applications in Consolidated Server Environments. *ICAC* (2008).

[23] KAPLAN, J. M., FORREST, W., AND KINDLER, N. Revolutionizing data center energy efficiency, 2008.

[24] LI, B., AND JIANXIN, L. *An Energy-saving Application Live Placement Approach for Cloud Computing Environment*. ACM International Conference on Cloud Computing, 2009.

[25] MANN, Z. Á. Approximability of virtual machine allocation: much harder than bin packing.

[26] MANN, Z. Á. Interplay of Virtual Machine Selection and Virtual Machine Placement. In *Service-Oriented and Cloud Computing*. Springer International Publishing, Cham, Aug. 2016, pp. 137–151.

[27] MATEOS, C., PACINI, E., AND GARINO, C. G. An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. *Advances in Engineering Software 56* (2013), 38–50.

[28] MELL, P. M., AND GRANCE, T. The NIST definition of cloud computing. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, Gaithersburg, MD, 2011.

[29] MISHRA, M., DAS, A., KULKARNI, P., AND SAHOO, A. Dynamic resource management using virtual machine migrations. *IEEE Communications ... 50*, 9 (2012), 34–40.

[30] MOENS, H., FAMAEY, J., LATRÉ, S., DHOEDT, B., AND DE TURCK, F. Design and evaluation of a hierarchical application placement algorithm in large scale clouds. *Integrated Network Management* (2011), 137–144.

[31] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic Design of Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming. *IEEE Transactions on Evolutionary Computation 18*, 2 (2014), 193–208.

[32] PANIGRAHY, R., TALWAR, K., UYEDA, L., AND WIEDER, U. Heuristics for vector bin packing. *research microsoft com* (2011).

[33] PIRAGHAJ, S. F., CALHEIROS, R. N., CHAN, J., DASTJERDI, A. V., AND BUYYA, R. Virtual Machine Customization and Task Mapping Architecture for Efficient Allocation of Cloud Data Center Resources. *Comput. J. 59*, 2 (2016), 208–224.

[34] PIRAGHAJ, S. F., DASTJERDI, A. V., CALHEIROS, R. N., AND BUYYA, R. Efficient Virtual Machine Sizing for Hosting Containers as a Service (SERVICES 2015). *SERVICES* (2015).

[35] RONG, H., ZHANG, H., XIAO, S., LI, C., AND HU, C. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews 58* (May 2016), 674–691.

[36] SARIN, S. C., VARADARAJAN, A., AND WANG, L. A survey of dispatching rules for operational control in wafer fabrication. *Production Planning and ... 22*, 1 (Jan. 2011), 4–24.

[37] SHI, W., AND HONG, B. Towards Profitable Virtual Machine Placement in the Data Center. *UCC* (2011), 138–145.

[38] SOLTESZ, S., PÖTZL, H., FIUCZYNSKI, M. E., BAVIER, A. C., AND PETERSON, L. L. Container-based operating system virtualization - a scalable, high-performance alternative to hypervisors. *EuroSys 41*, 3 (2007), 275–287.

[39] SOMANI, G., AND CHAUDHARY, S. Application Performance Isolation in Virtualization. *IEEE CLOUD* (2009), 41–48.

[40] SOTELO-FIGUEROA, M. A., SOBERANES, H. J. P., CARPIO, J. M., HUACUJA, H. J. F., REYES, L. C., AND SORIA-ALCARAZ, J. A. Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect Representation. *Recent Advances on Hybrid Intelligent Systems 451*, Chapter 28 (2013), 349–359.

[41] SPEITKAMP, B., AND BICHLER, M. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on services ...* (2010).

[42] SVARD, P., LI, W., WADBRO, E., TORDSSON, J., AND ELMROTH, E. Continuous Datacenter Consolidation. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* (2015), IEEE, pp. 387–396.

[43] TOMÁS, L., AND TORDSSON, J. Improving cloud infrastructure utilization through overbooking. *CAC* (2013), 1.

[44] UHLIG, R., NEIGER, G., RODGERS, D., SANTONI, A. L., MARTINS, F. C. M., ANDERSON, A. V., BENNETT, S. M., KÄGI, A., LEUNG, F. H., AND SMITH, L. Intel Virtualization Technology. *IEEE Computer 38*, 5 (2005), 48–56.

[45] VERMA, A., AND DASGUPTA, G. Server Workload Analysis for Power Minimization using Consolidation. *USENIX Annual Technical Conference* (2009), 28–28.

[46] WANG, Y., AND XIA, Y. Energy Optimal VM Placement in the Cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD* (2016), IEEE, pp. 84–91.

[47] XIAO, Z., JIANG, J., ZHU, Y., MING, Z., ZHONG, S.-H., AND CAI, S.-B. A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory. *Journal of Systems and Software 101* (2015), 260–272.

[48] XU, J., AND FORTES, J. A. B. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. *GreenCom/CPSCom* (2010).

[49] YAZIR, Y. O., MATTHEWS, C., FARAHBOD, R., NEVILLE, S. W., GUITOUNI, A., GANTI, S., AND COADY, Y. Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis. *IEEE CLOUD* (2010), 91–98.

[50] ZHANG, J., HUANG, H., AND WANG, X. Resource provision algorithms in cloud computing - A survey. *J. Network and Computer Applications 64* (2016), 23–42.

[51] ZHANG, Q., CHENG, L., AND BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications 1*, 1 (2010), 7–18.

[52] ZHANG, Z., HSU, C.-C., AND CHANG, M. Cool Cloud: A Practical Dynamic Virtual Machine Placement Framework for Energy Aware Data Centers. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)* (2015), IEEE, pp. 758–765.