Industrial & Application Paper

Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure

Antony S. Higginson
Oracle Advanced Customer Services
Manchester, United Kingdom
antony.higginson@oracle.com

Clive Bostock
Oracle Product Development
Manchester, United Kingdom
clive.bostock@oracle.com

Norman W. Paton
University of Manchester
Manchester, United Kingdom
norman.paton@manchester.ac.uk

Suzanne M. Embury
University of Manchester
Manchester, United Kingdom
suzanne.m.embury@manchester.ac.uk

ABSTRACT

Capacity planning is an essential activity in the procurement and daily running of any multi-server computer system. Workload placement is a well known problem and there are several solutions to help address capacity planning problems of knowing *where*, *when* and *how much* resource is needed to place workloads of varying shapes (resources consumed). Bin-packing algorithms are used extensively in addressing workload placement problems, however, we propose that extensions to existing bin-packing algorithms are required when dealing with workloads from advanced computational architectures such as clustering and consolidation (pluggable), or workloads that exhibit complex data patterns in their *signals*, such as seasonality, trend and/or shocks (exogenous or otherwise). These extentions are especially needed when consolidating workloads together, for example, consolidation of multiple databases into one (*pluggable databases*) to reduce database server sprawl on estates. In this paper we address bin-packing for singular or clustered environments and propose new algorithms that introduce a time element, giving a richer understanding of the resources requested when workloads are consolidated together, ensuring High Availability (HA) for workloads obtained from advanced database configurations. An experimental evaluation shows that the approach we propose reduces the risk of provisioning wastage in *pay-as-you-go* cloud architectures.

1 INTRODUCTION

When employing I.T. to satisfy requirements, regardless of the type, combination or configuration, one thing has always been prevalent, which is the question of consumption, whether it is prior to a migration, re-platform, upgrade or installation. Understanding what resources are required over a period of time is key to the management of all I.T. systems. As hardware specifications increase, for example, in performance and capacity, the actual physical infrastructure decreases but with the adoption of virtualisation, the problem of server sprawl, arguably stays the same. The trade off between system separation conflicts with *true* consolidation that requires the workloads to be combined together or share the same infrastructure. Knowing how best to *fit* workloads together is a problem that has always been an important problem to solve.

Bin-packing is a well understood concept and used extensively in many fields of business. In computing, bin-packing can be used to place smaller workloads into larger infrastructure to establish how resources should be assigned to a set number of tasks. However, bin-packing requires additional constraints when dealing with advanced system architectures such as clustering and elasticising workloads once placed.

In this paper, we tackle placement of different workloads from advanced databases configurations into complex cloud architectures, and make the following contributions:

- We identify the challenges and opportunities presented by advanced architectural features, when placing database workloads into complex cloud infrastructrures.
- We present a new bin-packing algorithm for provisioning database workloads that takes into account fine-grained monitoring information and advanced architectures such as clustered Oracle databases that enable High Availability (HA).
- We evaluate the algorithms in experiments that involve the placement of diverse workloads into real world-cloud architectures.

2 PROBLEM DESCRIPTION

This section provides more detail on the problem to be solved, which is multifaceted. When the placement problem is broken down it becomes apparent that these facets are inter-related, that is to say, all parts of the problem need to be addressed together, rather than just individual elements. Before describing the problem, we provide some details on how workloads are executed on complex advanced database architectures and the relevent metrics.

Clustering. Clusters are groups of servers (also known as nodes) that are managed together, participating to act as one system, usually to provide high availability, as shown in Fig 1. These clusters reduce downtime and outages by allowing another server to take over should an outage occur or maintenance be required. Database clustering is the running of a database across multiple servers while accessing shared storage, for example, database datafiles that are stored on a SAN (Storage Area Network), NAS (Network Attached Storage) or a disk array. In Fig 1 we can see that the user's wish to access the *HR*, *Sales* or *Call Centre* applications from any web server. The Net Services layer of the technology stack, handles client access and directs users connections to the node where the service is running, for example, *Sales* is directed to run from node 2. A heartbeat between the nodes ensures cluster integrity, should one of the nodes no longer react or

produce a heartbeat, the service fails over and user connections are handled by the remaining nodes. This type of architecture facilitates a 24*7 SLA and is common in enterprises today.

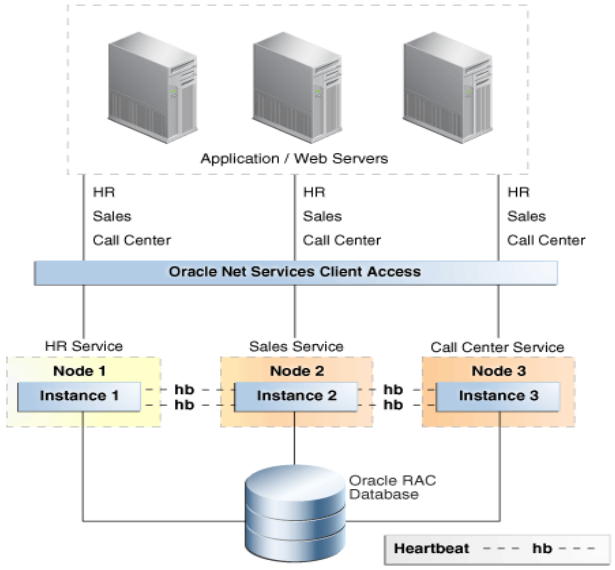


Figure 1: Oracle Database with Oracle RAC Architecture [6]

Workloads. Workloads consist of collections of tasks submitted by users. These tasks can be small units of work or individual pieces of SQL such as Data Modification Language statements (DML) that perform inserts, updates and deletes, that serve a web application, for example by way of an Online Transaction Processing system (OLTP). Other workloads consist of larger units of work such as batch jobs that aggregate information in the database periodically, for example, aggregation of sales data from hourly into daily, weekly or monthly, which is a common feature in data warehousing (OLAP). Another type of workload is a Data Mart that can be described as somewhere in-between OLTP and OLAP. Data marts consist of a combination of smaller DML OLTP units of work with medium OLAP type aggregations. Data Marts can be a subset of a large data warehouse that are subject-orientated such as sales, HR or Call Centre that are an aggregation of days and weeks rather than months or years.

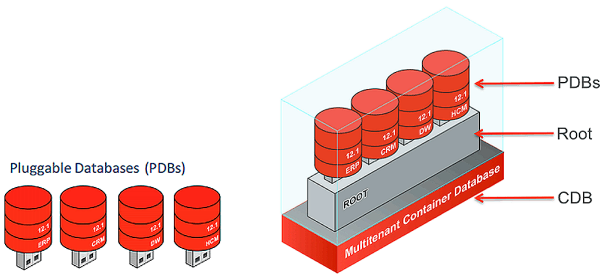


Figure 2: Oracle Database Multitenant Architecture [5]

These units of work vary in the amounts of resources consumed, and when analysed in a time series format, the task often dictates when the consumption of resources consumed to satisfy the task takes place. When these tasks are analysed via a trace we see those tasks exhibiting different patterns in their resource consumption. For example, in Fig 3 we have four workloads for CPU side by side. The first task, OLTP, shows a progressive trend with subtle repeating patterns (Seasonality). The next two tasks are OLAP showing a more definitive pattern of repeating tasks with little trend.

When placing workloads, we must treat each workload separately by extracting the *peak* values of each metric from each database instance on each node in each time interval, placing them on the target node. This is simple enough as long as the workloads are singular (independent of each other, and run on a single node). When the workload is clustered it becomes more complicated because, when placement commences, we must place the workloads while ensuring that we do not compromise HA. Therefore, for one clustered workload to be placed all workloads in the cluster must be placed. If we place a clustered workload without its sibling, we risk reducing a clustered workload to a singular workload, the consequence being that we lose high availability, thus compromising SLA's. The same understanding is required for pluggable databases [5], where a database may be detached from a singular database instance and plugged into another clustered database instance. This highlights that simple bin-packing routines are not suitable.

Consolidation. Consolidation can be described as running several workloads on a shared collection of nodes. There are several drivers for consolidation, such as system simplification or reducing server sprawl, whether that is the number of servers, clusters, databases or workloads. Server spawl can be described as a situation where servers are not being used to their full capacity, leading to significant wastage in terms of space, power and cooling, which can end up costing organizations substantial amounts of money.

Consolidation of databases has become easier with the development of pluggable databases where a database can be detached from a singular or Clustered Container Database (CDB), and plugged into another container DBMS that already has multiple plugged in databases. Detaching and attaching pluggable databases allows the pluggable database (and its associated data files) to be relocated to another server and be managed by another database instance (DBMS). This is shown in Fig 2, adding a further layer of abstraction when working in conjunction with HA. Each node in the cluster also houses a clustered container and within each clustered container there are pluggable databases. This architecture removes the support overhead of the database instance serving one database when one database instance can serve multiple plugged in databases while achieving HA. However, extracting the metric consumption on an instance with multiple pluggable databases residing together is challenging as the metric consumption is cumulative to the container. In this pluggable architecture, one must first separate the resource consumption for each pluggable, treating the pluggable database as a singular database workload.

Problem Statement. The problem to be solved can be considered as follows. Assume we are given a collection of *Workloads*, some of which are clustered, and a collection of computational *Nodes*. Each workload has a *time-varying* demand for resources defined using several metrics, and each server has a capacity defined using the same metrics. The task is to assign the Workloads to the Nodes, such that the demands placed by the workloads

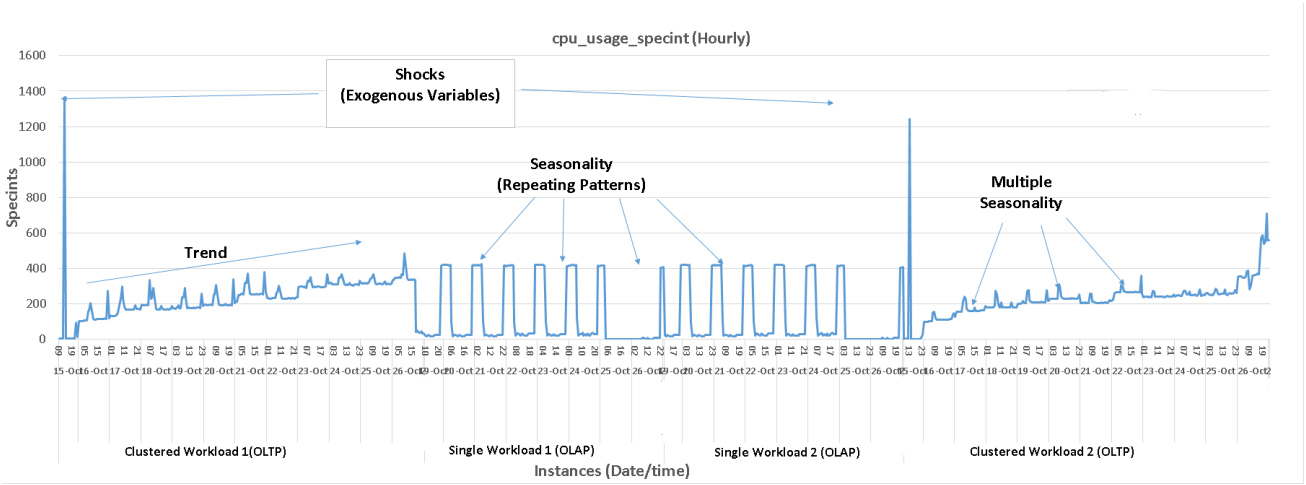


Figure 3: CPU Usage: Complex data structures.

are always within the capacity of the nodes, while respecting the constraints imposed by the clustered workloads.

3 RELATED WORK

Resource allocation or workload(s) placement in cloud environments is a well understood problem and there have been extensive studies and surveys undertaken as Hameed, Khoshkbarforoushha *et al* [16] and Bhavani and Guruprasad [4] both allude to in their survey's from 2014. Furthermore Singh and Chana [22] produced an extensive survey in 2016 that concluded that resource provisioning is a challenging job and there is a need for more research into optimal resource usage as this leads to improving the resources consumed with the aim of reducing wastage. This problem is ever more apparent in cloud computing with users accessing any shape of resources (vectors) from anywhere with the requirement of still being optimised. Vectors can be described as multiple metrics making up a shape rather than one singular metric. Most research has looked at consolidation from a Virtual Machine (VM) perspective to satisfy Quality of Service (QoS), and these are often single dimensions not vector dimensions. For example, placing workloads based on priority or assigning workloads to a VM via a migration to move problematic workloads should they exhaust a pool of resources as suggested by Yu, Wui *et al* [24]. In this survey, they identify several key problems of trying to figure out which applications can be consolidated together. Bin-packing algorithms do not take into consideration the divergent types of applications assigned to singular servers.

Other authors such as Sen, Ramachandra [21] and Zhang, Martin *et al* [25] view database workloads as controllable through the internal features of the database system via resource managers. However, they highlight the difficulties of the approach on cloud platforms where the resources are shared in the infrastructure. Halfpap and Schlosser [15] used an heuristic linear programming model to solve a placement problem by dissecting a database and where appropriate, replicating the data across multiple nodes. In doing so, this placement technique shared the workload from being cumulative on one node to being distributed between multiple nodes while keeping response times of the requests from said database optimum.

Most computing placement problems seem to centre on the provisioning of a VM (IaaS or PaaS) and not a workload SaaS or DBaaS. We identified in our previous work [18] that VM's mask the true signal of what the workload is actually consuming and that forecasting future resource requirements with a view to provisioning requires careful consideration of the nuances within the signal. Masdari and Khoshnevis in their 2019 survey [20] identified the techniques used to perform accurate forecasts of the resources being consumed as a precursor to provisioning. However, they stopped short of proposing how to *place* the workloads together once the forecasted future requirements are obtained.

When it comes to vector placement, using the bin-packing approach, several proposals have been put forward on prevalent customer use cases. For example, Wang, Hayat *et al* [23] looked at the scenario of a customer having multiple applications that make up a system. They propose a collection of algorithms based on a customers SLA as an important requirement to ensure business continuity. As application code or SQL is executed, and response times elongate this invariably lead to outages, provisioning the applications optimally by keeping these application response times as low as possible. Aydin, Muter *et al* [2], looked at another use-case for a VM placement problem with an added dimension of time. They looked for efficiencies in the power consumed by VM's fire-ups, with the aim of reducing energy consumption of data centres by minimising the number of servers and their fire-ups.

A current common theme with placement in a cloud setting is server sprawl and provisioning VM's. Doddavula, Kaushik and Jain [12], suggested reducing server sprawl with the introduction of a vector packing algorithm. In their novel approach, they classify vectors based on resource consumption, and then through Matrix multiplication determine the possible combinations. By then applying rules, either the workload is full or a magnitude of *full* determine where the workload should reside with other workloads until the maximum of the target server threshold is reached. However, in a clustered environment this approach will face challenges as it's possible that several workloads running on the same cluster are *full* or a combination of *classifications* that could break their algorithm. Clustered workloads that required enforced SLA's where workloads of differing priority may run from x node in the cluster as explained in Section 2.

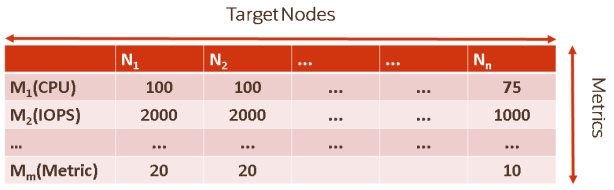


Figure 4: Nodes: Resource capacity.

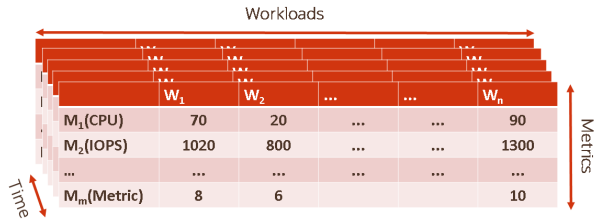


Figure 5: Workloads: Resource demand.

Notation	Interpretation
$Nodes = \{n_1, ..., n_n\}$	Computational Nodes.
$Workloads = \{w_1, ..., w_w\}$	Workloads.
$Metrics = \{m_1, ..., m_m\}$	Metrics (CPU, IOPS, etc).
$Times = \{t_1, ..., t_t\}$	Time intervals.
$Assignment(n_i) \rightarrow W$	The set of workloads $W \subseteq Workloads$ assigned to n_i.
$Capacity(n_i, m_j)$	The maximum capability of node n_i in relation to metric m_j.
$Demand(w_i, m_j, t_k)$	The peak demand that workload w_i has for resource of type m_j during time interval t_k.
$isClustered(w_i)$	True if w_i is in a clustered workload.
$Siblings(w_i)$	The set of workloads $s \subseteq Workloadss$ in the cluster of which w_i is a member.

Table 1: Notation for workload assignment.

4 BIN-PACKING

The basic bin-packing problem is the process of taking items of differing volumes and *packing* them into a finite number of *bins* in a way that minimises the number of bins used. The bin-packing problem is NP-complete as described by both Garey [13] and Korte *et al* [19], and thus approximate, heuristic, algorithms are often used in practice. There are many approaches to bin-packing, such as First-Fit Decreasing (FFD), Next-Fit (NF) and Best-Fit (BF) as discussed by Carter and Bays [3]. We look at First Fit Decreasing, and therefore, we treat all workloads being provisioned as having equal priority. Elastic Resource Provisioning (ERP) is assigning all workloads into one bin and elasticising the bin to fit around the workloads being placed, as described by Yu, Qiu *et al.* [24]. Our approach is to enhance FFD to tackle complex architectures such as clustered workloads and empirically evaluate them experimentally.

4.1 First Fit Decreasing Placement (FFD)

The notation used to describe bin-packing in this paper is listed in Table 1 and illustrated in Figures 4 and 5. The available resource is represented as a set of *Nodes*, each of which has a *Capacity* defined using a set of *Metrics*, that can include CPU, memory usage, logical IOs per second, etc. The bin-packing task is to assign a set of *Workloads* to the nodes. The *Demand* associated with each workload is defined in terms of the same metrics that are used to describe the nodes, but the *Demand* varies over a set of *Times*. The time-varying demand may be based on measured or predicted loads; our earlier work has explored the use of time series analysis techniques to model database workloads for capacity planning [18].

Sorting Workloads. First fit decreasing sorts workloads such that they can be assigned largest-first, and hence we need a notion of size; here we define order in terms of the demand across different metrics, normalising according to the total usage for each metric.

Overall demand for each metric (CPU, IOPS, MEMORY, STORAGE) is obtained from the demand of each workload:

$$overall\_demand(m) = \sum_{n_i \in Nodes} \sum_{w_j \in Workloads} \sum_{t_k \in Times} Demand(w_j, m, t_k) \quad (1)$$

Then the normalised demand of a workload w is the normalised sum of its demand over all metrics:

$$normalised\_demand(w) = (\sum_{m_j \in Metrics} \sum_{t_j \in Times} (Demand(w, m_j, t_k)/overall\_demand(m_k)) \quad (2)$$

Then the workloads can simply be sorted by their normalised demand. In practice, when assigning clustered workloads, clusters are considered in the order of the demand of their most demanding workloads, and then the workloads within a cluster are also sorted locally.

Node Capacity. The capacity of a node i for a metric m at time t is the original capacity of the node minus the resource usage of the workloads assigned to the node:

$$node\_capacity(n, m, t) = Capacity(n, m) - (\sum_{w_i \in Assignment(n)} Demand(w_i, m, t) \quad (3)$$

Fitting. A workload w can be added into a node n if there is capacity for all the metrics at all times.

$$fits(w, n) = \forall m \in Metrics \; \forall t \in Times \; Demand(w, m, t) \leq node\_capacity(n, m, t) \quad (4)$$

A clustered workload, *isClustered* from Table 1, sonsists of database instances (workloads) that are siblings of each other as shown in Fig 1. The node is the number of nodes on which W_i is to be run, and $w \in ClusteredWorkload$ is the set of workloads that need to be assigned to the target nodes discretely. A rule is enforced where all *Siblings* must be packed into discrete target nodes before the cluster is said to be packed. If at any point one of the *Siblings* fails to pack into a discrete target node then all

siblings are rolled back and the resources are released back to *node_capacity*.

5 APPROACH

Using techniques based on First Fit Decreasing (FFD) and the definitions from Section 4, our aim is to place database workloads into a target Oracle Cloud Infrastructure (OCI) [7] that supports clustered workflows. The aim is to achieving savings in costs, both financial *(pay-as-you-go)* and to release resources back to the cloud pool for utilisation elsewhere.

5.1 Workload Placement Algorithm

A high level description is shown in Algorithm 1. One of the major challenges of workload placement in computing estates that have adopted clustered configurations, is accounting for clustered workflows, that must be placed in their entirety or not at all. Identifying the workload is clustered and on how many nodes is key to placing the workload in the target cloud OCI configuration.

Algorithm 1: FitWorkloads

```
Input: Workloads (from Table 1)
       Nodes (from Table 1)
Result: Assignment(n) (from Table 1)
        NotAssigned
Assignment(n) = ∅ for all n ∈ Nodes
NotAssigned = ∅
foreach w ∈ Workloads ordered by normalised_demand
  (Equation 2) do
    assigned = false
    foreach n ∈ Nodes do
        if isClustered(w_i) then
            if w_i not already added to Assignment with
              cluster or included in NotAssigned then
                assigned =
                  FitClusteredWorkload(Siblings(w_i),
                  Nodes, Assignment, NotAssigned)
            if assigned then
                break
        else if fits(w,n) then
            Add w to Assignment(n)
            assigned = true
            reduce node Capacity (Equation 3) by
              Assignment
            break
    if (not assigned) then
        Add w to NotAssigned
Report on Workloads Assigned, NotAssigned and Nodes
  Capacity
```

Firstly we extract key information as inputs, ordering workloads by demand. Key configuration data is stored in a central repository [8] that stores whether a workload is clustered or not. If a workload is part of a cluster then we set a flag for that particular workload (represented by *isClustered* in Table 1), and the full set of workloads with which it is clustered can be obtained using *Siblings* in Table 1.

When placing workloads, if the workload w is from a single database instance then we simply check if the workload $fits$ (Equation 4) into an available node, and if so, add it to *Assignment* for that node. We report back to the user all workloads that have been fitted (by way of *Assignment*), and any that have not (by way of *NotAssigned*).

If, however, the workload is clustered we extract the related *Sibling* workloads in the cluster from a central repository; as we use the Oracle Enterprise Manager system [8] and the OEM intelligent agent to capture all performance and configuration data relating to the database instances. OEM utilises a database schema to hold information relating to the workloads, and databases instances, and we handle this via a Global Unique Identifier (GUID).

5.2 Fitting Clustered Workloads

The fitting of clustered workloads, aims to enforce high availability by placing *all* workloads in a cluster before it can report back that the workloads are fitted. For example, if the clustered workload has three nodes with a database instance running on each node as described in Fig 1, then it must place the workloads on three discrete target nodes or it will roll back what has already been placed. We show this in Algorithm 2.

Firstly we understand how many nodes make up the cluster (1,2,..,N) nodes, which gives an indication of how many target nodes are required. We cannot fit a clustered workload from three nodes into two target nodes, therefore, we perform a test to ensure that the requisite number of target nodes are available. If there are not enough target nodes then we stop otherwise we loop through all of the workloads ensuring that the siblings of the cluster are assigned to discrete target nodes. Each time an assignment takes place the amount of resource of the target node is reduced by the vector of the workload. Finally, we report on what workloads are assigned to each target node.

5.3 Evaluating the Placement

Once the workloads from all database instances have been assigned and placed in their target nodes. We overlay each workload by the time frequency (Hourly), allowing an understanding of the existing data signals (peaks and troughs) when the workloads are consolidated together. A simple groupby ($\sum$) per hour and per metric shows the newly consolidated data signal. In traditional bin-packing exercises, the *max_value* of a metric is taken and then bin-packing is based on that value, however, if a peak is singular, for example, without pattern then the prospect of over provisioning becomes apparent. When the new trace is displayed over an *X,Y* (Stacked), which we show in Section 7.2 and Fig 7. We can clearly see the consolidated workloads exhibit their complex traits such as seasonality, trend and shocks against the threshold limit of the bin. This simple approach allows us to understand and where possible, perform or feed into further elastication exercises that can be performed on the bin to fit the consolidated workloads more tightly.

6 EXPERIMENTAL SETUP AND WORKLOADS

Firstly we execute a selection of workloads (OLTP, OLAP and Data Mart) on different Oracle database configurations of varying versions (10g, 11g and 12C) on a Oracle Enterprise Linux Operating System or Oracle Exadata (clustered workload) [9]. Executing the workloads for 30 days allows key database features such as optimisers and caching to be warmed up or routine backups to take place, which all consume resources and influence

Table 2: Table of Experiments

Experiment	Workloads	Target Bins
Basic Single Database Instance	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal equal size
Basic Clustered Workloads	10 Workloads (10 RAC OLTP (5*2 Exadata nodes))	4 * OCI Bare Metal equal size
Basic different sized target bins	10 Workloads (10 OLTP, 10 OLAP and 10 DM)	4 * OCI Bare Metal unequal size
Moderate Combined (Clustered and Single Instance)	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM)	4 * OCI Bare Metal unequal size
Moderate scaling	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	4 * OCI Bare Metal equal size
Moderate different sized target bins	20 Workloads (4 * 2 node clustered + 5 OLTP, 6 OLAP and 5 DM	6 * unequal OCI Bare Metal
Complex (Scaling & different sized bins)	50 Workloads (10 * 2 node clustered + 10 OLTP, 10 OLAP and 10 DM	16 * unequal OCI Bare Metal

Algorithm 2: FitClusteredWorkload

```
Input: ClusteredWorkload, including sibling data (from
       Algorithm 1)
Nodes (from Table 1)
Assignment(n) (from Table 1)
NotAssigned (from Algorithm 1)
Result: assigned
foreach w ∈ ClusteredWorkload ordered by
  normalised_demand (Equation 2) do
    assigned = false
    if target_nodes are <= source_nodes then
        foreach n ∈ Nodes do
            if fits(w,n) then
                Add w to Assignment(n)
                assigned = true
                reduce node Capacity (Equation 3) by
                  Assignment
                break
        if (not assigned) then
            Remove all members of ClusteredWorkload
              from Assigned
            Add all members of ClusteredWorkload to
              NotAssigned
            increase node Capacity (Equation 3) by
              Assignment
            break
        not enough nodes to fit
Report on Workloads Assigned and NotAssigned
return assigned
```

the bin-packing routines by way of providing a different metric values between a cold and warm database. We also monitor the workloads to ensure the workloads are running smoothly without error. The workloads are executing Data Manipulation Language (DML) statements such as inserts, updates and deletes while performing large data aggregations, for example Business Intelligence (BI) reports through a Java environment consisting of a web container, giving us an N-Tier architecture using the Oracle load generator Swingbench [14]. This environmental setup is reflective of Database Management systems in use today.

We utilise the Oracle Enterprise Manager [8] system to execute and capture all performance and configuration data via an Intelligent Agent, which captures metrics such as CPU, IOPS, Memory and Storage used. The agent executes commands to retrieve the *max_values* of key metrics such as *sar*, *iostat*, and *memory* on the host and metrics specifically from the database are also obtained such as storage used, memory used, CPU % used and logical read/writes. The agent captures these metrics at 15 minute intervals and stores the values in a central repository. Aggregations on the data captured every 15 minutes are then performed providing a *max_value* for each metric for each database instance and host hourly, daily, weekly or monthly.

Storing the values in a central repository this way, enables the ability to align the metrics uniformly over consistent observations such as hourly in an overlay manner, allowing an easy comparison of all database instances. Using python open source libraries such as numpy and pandas we then executed the algorithms and empirically evaluated the placement of the workloads into target bins. We could use *average_values* from the metrics captured but we choose *max_values* for the simple reason of provisioning on an average will usually be lower than a *max_value* and if a VM hits 100% utilised it will panic and may cause an outage. Therefore, we always place on a max_value from a metric.

Each workload generates complex data traces as shown in Fig 3, highlighting repeating patterns (seasonality), trend and shocks. Shocks are reflective of large IO operations, for example online database backups, and this can be seen in the metric IOPS. In Fig 3 we have displayed four workloads side-by-side for CPU usage that highlights these complex patterns, which are indicative of systems employed by most enterprises today. As workloads become larger in size, arguably the result is slower execution times and this is shown by the workloads exhibiting trend. Each experiment increases in complexity and scale as the number of workloads increase, which is described in Table 2. Each experiment and its results are discussed in detail in Section 7 where we produce charts that highlight the workloads consolidated together after being placed on a particular node. The target configuration is Oracle Cloud Infrastructure (OCI) [7] using the Bare Metal Configuration as shown in Table 3.

In the experiments we are only testing the database placement algorithms, as they are orthogonal to modelling. The placement algorithms do not know if the traces being inserted as inputs to the algorithms are actual or modelled, however, it is perfectly plausible that the inputs have first been predicted to obtain an

estimate of future resource consumption to model what a placement design may look like, which is a common planning exercise in any estate migration. Once the workloads are placed we also evaluate the consolidated workloads assigned to the target nodes to identify if there is any wastage that can be further eliminated by, for example, an elastication exercise.

Table 3: OCI Target Bin Configuration (BM.Standard.E3.128) [7]

Shape	Metric	Comments
Compute Shape	128 OCPU 2048 GB's Memory Block Storage	Equivalent to 980 SPECInts [10] per bin
Block Storage Shape	32 * 4TB Volumes 35,000 IOPS per vol	Equivalent to 1,120,000 IOPS per bin 128000GB Physical Storage
Network Shape	2* 50Gbps throughput Max 128 VNICS	65 per physical NIC

7 EXPERIMENTS AND ANALYSIS

We conducted several experiments that start off simple and then grow in complexity providing a detailed evaluation of the algorithms that reflect use cases of estates employed by customers today. For the purposes of saving space in the paper we have only included some of the most prominent charts and results that are interesting. Furthermore we do not describe all experiments listed in Table 2 to save space in the paper. The experiments we do show highlight the algorithms working to their full potential. We aim to answer the following questions:

(1) Minimum targets needed - What is the minimum number of target bins needed to fit all workloads across all vectors (metrics)?
(2) First Fit Decreasing Simple Placement - How do we place the workloads equally across equal sized bins?
(3) First Fit Decreasing Clustered Placement - If there are clustered workloads can we ensure that all clustered workloads are placed without compromising High Availability?
(4) Evaluating the placement - Once the workloads are placed (consolidated) together can we identify wastage with the aim to resize the target nodes, obtaining a tighter fit, reducing over provisioning?

We have included sample outputs from each experiment and the reader will note that these are not uniform in their outputs and this is because we wish to only highlight prominent outputs that answer questions the experiments pose. If we are to supply full sample outputs for each experiment we would quickly exhaust space in the paper. Also, we have used sample outputs as opposed to full UI screenshots, also saving space and showing the algorithms working. In our opinion, UI design although an important feature of any application is not as important, to this paper, as the algorithms working.

In all of our experiments we executed our algorithms multiple times with both adequate and inadequate target nodes and resources to ensure that the algorithms would place and reject workloads appropriately. In the experiments we report in this paper we took the average number of workloads we expect customers to provision per Oracle Cloud Architecture, i.e., customers

```
Can we fit all instances into minimum sized
bin for Vector CPU?

===== list

List of workloads
['DM_12C_1': 424.026, 'DM_12C_2': 424.026,
'DM_12C_3': 424.026, 'DM_12C_4': 424.026,
'DM_12C_5': 424.026,'DM_12C_6': 424.026,
'DM_12C_7': 424.026,'DM_12C_8': 424.026,
'DM_12C_9': 424.026, 'DM_12C_10': 424.026]

Target Bins 0
['DM_12C_1': 424.026, 'DM_12C_2': 424.026,
'DM_12C_3': 424.026, 'DM_12C_4': 424.026,
'DM_12C_5': 424.026,'DM_12C_6': 424.026]

Target Bins 1
['DM_12C_7': 424.026, 'DM_12C_8': 424.026,
'DM_12C_9': 424.026, 'DM_12C_10': 424.026]
```

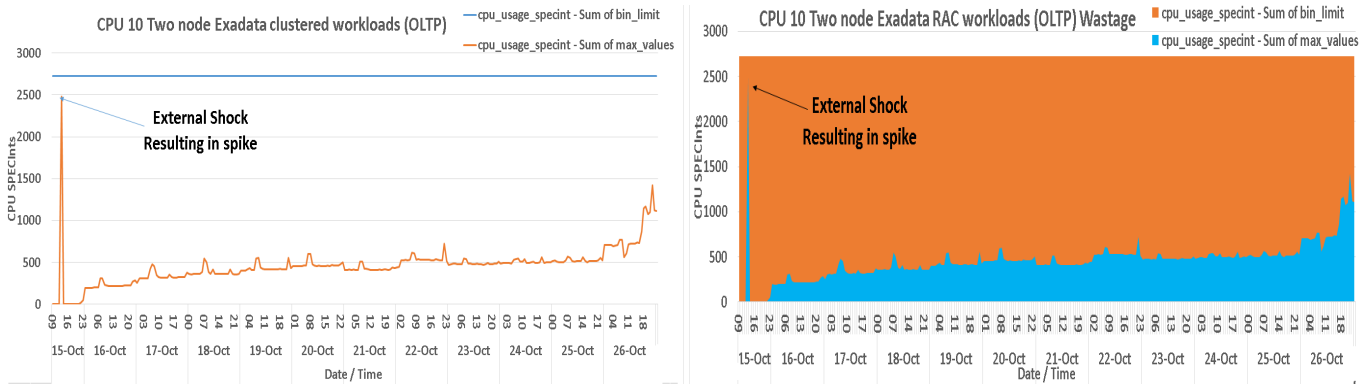
Figure 6: Sample output: Minimum Number of Nodes CPU

mostly provision a *1-to-1* relationship of an instance per VM. However, consolidation of workloads is rising as the technology allows; for example combining database workloads together is more achievable with the introduction of Pluggable databases [5] where one database can be detached from one instance and *'plugged'* into another instance. An instance includes the memory structures and optimiser that serves the database. The instance is then shared across multiple VM's that make up a physical cluster thus bin-packing multiple instances together is becoming more apparent.

7.1 Experiment (Basic) - Placement of Single Database Workloads (OLTP, OLAP & DM Workloads)

Overview. The first experiment involves placing 10 OLTP, 10 Data Mart and 10 Data Mart workloads from a source configuration of Oracle single database instances as described in Table 2, into four equally sized target OCI bins of a configuration described in Table 3. For the purpose of savings space in the paper have not shown these charts here. We show the sample outputs from the command line which we discuss in detail in *results*.

Results. In the sample outputs shown in Fig 6, we represent one metric (CPU) in the vector, although in our outputs, we cover all metrics in the vector. Each of the *max_values* taken from the hourly time period are listed as one list and then *placed* into the minimum number of bins. Each bin is represented within square brackets *'[]'*. In Fig 6, each workload is labelled by using a precursor to the value. For example workload DM_12C_1 identifies, DM representing the type of workload thus Data Mart, 12C representing the version of Oracle database the workload was executed with, and 1..10 being that actual workload. The values after the *':'* in Fig 6 is the *max_value*. As we show in Fig 6, we have successfully answered question 1, treating all workloads heuristically, what is the minimum number of target nodes required to fit my workloads?



(a) CPU OLTP RAC Workloads

(b) CPU OLTP RAC Wastage

Figure 7: RESULTS: Consolidated placed workloads & Potential Wastage

```
How many of the instances (Database Workloads)
can we get in 4 equal sized bins?

bin packed it looks like this
Target Bins 0
{'DM_12C_9': 424.026, 'DM_12C_5': 424.026,
'DM_12C_10': 424.026}
Target Bins 1
{'DM_12C_8': 424.026, 'DM_12C_4': 424.026,
'DM_12C_1': 424.026}
Target Bins 2
{'DM_12C_7': 424.026, 'DM_12C_3': 424.026}
Target Bins 3
{'DM_12C_6': 424.026, 'DM_12C_2': 424.026}
```

Figure 8: Sample output: Workloads placed equally across targets (CPU)

In Fig 8 we answer question 2, which is can we place the workloads equally across the target nodes? In the sample output shown in Fig 8 the target nodes are represented by brackets *'{}'*. Question 3 is not answered in this experiment as there are no clustered workloads however, we will answer this question in other experiments. Question 4 we ask in each experiment; evaluating the target nodes after placement can we resize the bins to obtain further savings? We have not shown these charts for this experiment due to the space available. We discuss this question in detail in Section 7.2 from charts a & b in Fig 7.

7.2 Experiment (Basic) - Placement of 10 Clustered workloads (RAC)

Overview. In this experiment, from Table 2, we focus on clustered workloads of a type OLTP, which are executed on an Oracle two node Exadata Machine [9]. There are 10 workloads, which equates to five two node clusters executed on an Oracle 11G version database. The workloads exhibit complex data structures as shown in Fig 3 such as seasonality, trend and exogenous shocks. In this experiment we are answering Question 3 of the first fit decreasing placement algorithm with the aim of enforcing High Availability. All workloads in a cluster must be placed or no workloads from the cluster will be placed as described in algorithm 2. The target bins are of type equal sized Oracle Cloud Infrastructure [7] and covered in Table 3.

Results. In the sample output shown in Fig 9, which we have cropped for ease of reading and space within the paper, we show the algorithm's command line output. This output is available in all experiments but we focus particularly on clustered workloads to highlight our placement algorithms working clearly. The target bin configurations are displayed along with the databases instances and their *max_values* for a given time period. The first block in Fig 9 titled *'Cloud Configurations'*, lists the target OCI bin vectors and their available space named OCI0,..,OCI6. The next block in Fig 9 titled *' Database instance / resource usage:'*, lists the source RAC database instances and their vectors, which we have also cropped for space within the paper by only showing four clustered databases instances (8 workloads).

We inform the user of the instances we are intending to place and as the algorithm is executed, providing a real-time decision of each instance being placed however, we have not shown this output here. Once executed we provide a summary of workloads placed, refused or rolled back, for example, if an workload from a clustered database instance has been placed but a sibling was not placed. It will require a rollback and the counter *'rollback count'* increments as shown in the block titled *'Summary'* in Fig 9. We observed in our tests that once an instance is rolled back, the resources are released and made available again, allowing a smaller vector size to be placed. We also provide a list of workloads that failed to fit due to lack of resources, reporting a list of failed instances to the user. In the block titled *'Cloud Target : DB Instance mappings:'*, we provide a mapping of the clustered workloads that were placed and their siblings, including which target node they are placed, note that no two instances from the same cluster are ever placed in the same target node; they are always placed discretely.

Evaluating the target nodes after placement, can we resize the bins to obtain further savings (Question 4)? Charts a & b in Fig 7 show several interesting points. Chart 7a shows an external shock that caused a spike and the FFD algorithm takes this *max_value* when placing workloads. When the workloads are consolidated together we can see trend as the line gradually rises. The available, target, resources are shown by the blue line and the large spike fits below the line. However, Chart 7b displays the potential CPU resources that will not be utilised (orange). Therefore,

```
Cloud configurations:
                     OCI0     OCI1 ...   OCI11 ...  OCI16
metric_column
cpu_usage_specint     2728     2728 ...    2364 ...  681.25
phys_iops          1120000  1120000 ...  560000 ...  280000
total_memory       2048000  2048000 ... 1024000 ...  512000

Database instances / resource usage:
RAC_1_OLTP_1 RAC_1_OLTP_2 RAC_2_OLTP_1 RAC_2_OLTP_2 RAC_3_OLTP_1 RAC_3_OLTP_2 RAC_4_OLTP_1 RAC_4_OLTP_2

metric_column
cpu_usage_specint  1,363.00   1,363.00    1,363.00    1,363.00   1,363.00    1,363.00    1,363.00   1,363.00
phys_iops         16,341.00  16,341.00   16,341.00   16,341.00  16,341.00   16,341.00   16,341.00  16,341.00
total_memory      13,822.00  13,822.00   13,822.00   13,822.00  13,822.00   13,822.00   13,822.00  13,822.00
USED_GB               53.47      53.47       53.47       53.47      53.47       53.47       53.47      53.47

SUMMARY                                 Cloud Target : DB Instance mappings:
=======                                 ====================================
Instance success: 8.                    OCI0 : RAC_1_OLTP_1, RAC_2_OLTP_2
Instance fails: 12.                     OCI1 : RAC_2_OLTP_1, RAC_3_OLTP_2
Rollback count: 0.                      OCI2 : RAC_3_OLTP_1, RAC_4_OLTP_2
Min OCI targets reqd: 10                OCI3 : RAC_4_OLTP_1, RAC_1_OLTP_2

Original vectors by bin-packed allocation:
                     OCI0    RAC_1_OLTP_1  RAC_2_OLTP_2
metric_column
cpu_usage_specint     2728     1,363.31     1,363.31
phys_iops          1120000    16,340.62     16,340.62
total_memory       2048000    13,822.21     13,822.21
```

Figure 9: Sample output: 10 RAC Workloads First Fit Decreasing High Availability Enforced

elasticising the target cloud node, and reassigning the resources would reduce wastage. In this experiment we have successfully placed clustered workloads proving our placement algorithms (Algorithm 2) and evaluated the consolidated workloads in their target nodes, successfully identifying potential wastage that may occur.

7.3 Experiment (Complex) - Placement of combined workloads (Clustered & Single Instance), varying sized bins at scale

Overview. This experiment is the most complex of all of the experiments we conducted. In this experiment we are answering the question of scale by placing a large number of workloads of varying size into different sized target nodes. The target nodes reduced in their available resources, which arguably, reflects the use case most customers face today when undertaking a migration exercise that involves procuring and placing workloads from on-premises advanced configurations such as clustering into cloud configurations. We tackle this experiment by running a combination of algorithms which are the following: -

- What is the minimum number of target nodes I require based on the size of my vectors?
- What is the maximum number of workloads I can fit into the available target nodes while keeping the integrity of the clustered workloads?
- Can we work at scale, which is a plethora of eclectic workloads into varying sizes of targets?

Firstly we obtained an estimate on the minimum number of target nodes based on the *max_values* obtained for each metric within the vector from each workload. Taking the configurations based in Table 3, the number of nodes needed to place 50 workloads was: -

- CPU - On this metric the advice was 16 target bins
- IOPS - On this metric the advice is 10 target bins
- Storage - On this metric the advice is 1 target bin
- Memory - on this metric the advice is 1 target bin

From analysis of the workloads we identified that some of our workloads are CPU and IOPS *heavy*, therefore, allowing the algorithms to utilise 16 available target nodes was key to meeting the demands of the experiment bearing in mind it is a question of scale. Utilising 16 target nodes of OCI configurations of varying sizes, these being 10 target bins 100%, 3 being 50% and 3 25% available resource from Table 3. In Fig 9 under Section *"Cloud Configurations"*, which we have cropped to aid viewing we show OCI11 (50%) and OCI16 (25%). Placing our workloads heuristically on a first fit decreasing method, all workloads are treated equally, but focusing on enforcing High Availability as there is a combination of both single and clustered workloads.

Results. The results of this experiment were very much the same as the previous experiments in that the algorithms (Algorithms 1 and 2) worked as expected. All the algorithms fitted their workloads in a First Fit Decreasing manner. All the Algorithms evaluated the nodes once placement of the workloads took place to identify further efficiencies. However, what we wish to focus

```
Rejected instances (failed to fit):
metric_column  cpu_usage_specint  phys_iops  total_mem
RAC_1_OLTP_1          1,363.31   47,982.17   13,882.21
RAC_7_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_9_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_9_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_8_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_1_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_8_OLTP_1          1,241.99   47,982.17   12,723.78
RAC_10_OLTP_1         1,241.99   47,982.17   12,723.78
RAC_7_OLTP_2          1,241.99   47,982.17   12,723.78
RAC_10_OLTP_2         1,241.99   47,982.17   12,723.78
```

Figure 10: Sample output: Experiment 4 RAC workloads failed to fit

on in this experiment was the instances that failed to fit as shown in Fig 10.

From the sample output shown in Fig 10 it would suggest there is a random nature to the instances not being fitted and this is explained because of the following. When we first list the instances, their workloads, vectors and the amount of resource consumed we order them in descending order with the largest single instance being first then the largest RAC instances. By optimally sorting on size we avoid the algorithm rolling back already placed instances as the available target nodes exhaust their resources with siblings not been placed. We must treat the siblings of the clusters equally then sort order based on the size of the total cluster.

8 CONCLUSIONS AND FUTURE WORK

Summary. In this paper we evaluated the technique of Vector Bin-Packing utilising the First-Fit Decreasing algorithm against databases that employ advanced technologies such as clustering and pluggable databases with a view to fitting a variety of workloads into complex target cloud configurations such as Oracle Cloud Infrastructure with a Bare Metal configuration. When placing workloads into Cloud configurations because most cloud providers provision on multiple dimensions such as IOPS, Storage, CPU and Memory, a vector approach is required. If the *Cloud Consumer* is also a *Cloud Provider* then the vectors are likely to increase in number, covering other areas of cloud technology, for example Network throughput, Bandwidth or Virtual Network Interface Cards (VNIC) configuration to name but a few. The approach adopted provides the ability to place workloads on scaleable vectors, by increasing the number of metrics $[m_1, .., m_m]$.

We wanted to understand once the workloads were placed could we make further efficiencies? Given placement algorithms only take the *max_value* of a metric that is associated with its workload over time, once a workload migrates architectures, the *signal* changes, especially when analysed in a time series format as described from our earlier work [18]. Therefore, over-provisioning is possible without understanding if there are repeating patterns or trends within the signal. The charts in Fig's 7a and 7b, indicated by the colour orange shows that what, was initially, provisioned may not be used. Our approach identified this wastage.

- What is the maximum number of target nodes needed to consolidate my workloads?
- What size do I need those target nodes to be?
- How should those workloads be placed in the target nodes?
- Is the target node adequately sized once placement of the workloads takes place?
- Will placement of the workloads compromise my SLA's?

Given the popularity of advanced database features such as high availability and consolidation we had to extend the existing FFD bin-packing algorithms rather than simply mapping a database instance as a *1-to-1* mapping to a VM. By consolidating the workloads together, gave us additional complexities to take into consideration. For example, pluggable databases are still attached to a global database memory structure consuming resources. By treating a pluggable database as a single instance workload we were able to reduce complexity within the algorithms, allowing us to place pluggable databases. Our algorithms needed to be multi-faceted in that they can place simple, complex and very complex vectors attributed to any database workload regardless of the source database configurations. By treating pluggable and standby databases as a single instance workload allowed us to perform workload placement without introducing further notation in our formulas. A standby database will usually be in recovery mode applying all archivelogs from all nodes in the primary cluster therefore, a standby is a single instance which is more IO resource intensive than memory or CPU as we have shown in our earlier work [17].

Central Repository. Using an intelligent agent capable of Monitor Analyse Plan and Execute (MAPE) (Arcaini *et al* [1]) to identify, capture, store metric and configuration data centrally, allowed us to align the time series data of the workloads uniformly. An intelligent agent executes a command for example *sar* or *IOSTAT* at a particular time with the command results being stored in a central repository within a database schema. Aggregations are performed on the metric data to an hourly value and while this has the negative affect of smoothing the signal (averaging the time points) it allowed us to compare the workloads at any given time period easily, as shown in Fig 5, reducing the amount of data wrangling in the application layer by python libraries such as Pandas, Numpy etc.

Benchmarks. Comparing Servers with different performance speeds such as IOPS or CPU is a challenge and there we utilised benchmarks. SPECInt 2017 [10] was used to compare the workload consuming CPU on one architecture compared with another chip architecture. Storage benchmarks were also provided based on Transaction Processing Performance Council [11] benchmarks. However, RDBM systems utilise complex memory algorithms that often bypass fetch operations of the database therefore, logical reads were taken as the metric. However, given our approach and algorithms allows placement on a vector that is scaleable, other Metrics such as physical IOPS could be used if one chooses to do so.

Automation. With the manual approach of performing a workload placement exercise, technicians tend to adopt a spreadsheet approach when placing workloads into clouds. This approach can be cumbersome, for example, manually researching, converting the CPU (SPECint), IO speeds and Memory between the source and target architectures, so creating the spreadsheet is time consuming. Often these spreadsheets build in complexity and are bespoke to individual customers resulting in inflexibility, resulting in *'expert friendly'* analysis that only the author understands. We wanted to automate this process with the aim of reducing the

level of effort technicians spend manually building spreadsheets, reducing errors from miscalculations that may occur in bespoke spreadsheets and reduce the time to complete a placement plan from weeks/months to hours/days.

When we execute our algorithms we are effectively retrieving the configuration and performance data from a central repository. For example, extracting the CPU make, model and its SPECInt value that is obtained by the intelligent agent, therefore performing a comparison rather than manually researching this data. The Algorithm can then quickly place and store the placement design of the workloads as a *'plan'* in a normalised database schema, rather than having complex sets of bespoke spreadsheets. This approach worked well, allowing us to execute the placement algorithms in minutes rather than days or weeks.

Conclusion. In conclusion, we believe that there is a need for accurate workload placement especially when provisioning services such as IaaS, PaaS, DBaaS or SaaS, whether that is on-premise, remote or hybrid clouds. However, knowing which algorithm or collection of placement algorithms to use is key as one simply can not utilise a standard approach or an *off-the-shelf* technique when advanced workload configurations such as clustering are employed as our experiments show. We focused on the First-Fit Decreasing bin-packing method on advanced databases architectures such as Clustering and Pluggables and found that we needed to extend the FFD algorithm to accommodate sibling workloads within the cluster, especially when the cluster is consuming resources unevenly.

Future Work. During our experiments we found curious behaviour causing us to extend our new FFD Algorithm further, and this was attributed to ordering the workloads prior to placement, something we did not expect. The result was to insert steps 1 and 3 of algorithms 1 and 2 to include ordering in descending order with the largest workload being the first to be placed while also ordering the largest available resource target nodes being first. However, we did not account for the siblings when ordering clustered workloads. Therefore, we had to order the workloads *and* their siblings together. The reasoning for this is to account for the siblings in a cluster that consume resources unevenly. When clustered instance workloads are listed individually one workload within a cluster can be located considerably down the pecking order compared with its sibling if a simple ordering exercise takes place. Eventually the target nodes are exhausted of their resources and placement ceases. If the target node runs out of available resources before the sibling is placed then a rolling back exercised is performed.

Therefore, it is critical to order on the cluster and its siblings in descending order. This is more of a work around really rather than a solution. We are working on a solution to this problem but the likely answer will be to take the cumulative approach of the total amount of resources consumed per cluster and order based on number of nodes then resources consumed. We also intend to enable a user defined priority assignment function. Assigning a user defined priority to a workload allows for separation between live systems as some systems may be more critical than others. In our earlier work we leveraged Machine Learning (Supervised) coupled with Time Series Analysis [18] to predict future resource consumption of a workload and we see a combination of those techniques and placement to create a more informed choice when one is making decisions on what systems can be placed based on their future resource consumption.

REFERENCES

[1] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. 2015. Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 13–23. https://doi.org/10.1109/SEAMS.2015.10

[2] Nurşen Aydın, İbrahim Muter, and İlker Birbil. 2020. Multi-objective temporal bin packing problem: An application in cloud computing. *Computers and Operations Research* 121 (2020), 104959–.

[3] Carter Bays. 1977. A Comparison of Next-Fit, First-Fit, and Best-Fit. *Commun. ACM* 20, 3 (March 1977), 191–192. https://doi.org/10.1145/359436.359453

[4] Bhavani and Guruprasad. 2014. Resource Provisioning Techniques in Cloud Computing Environment: A Survey. *IJRCCT International Journal of Research in Computer and Communication Technology* 3, 7 (2014), 395 – 401.

[5] Oracle Corporation. 2017. *Oracle Multitenant Architecture.* https://www.oracle.com/uk/database/multitenant/.

[6] Oracle Corporation. 2020. *Oracle Database High Availability Overview and Best Practices, 19c.* https://docs.oracle.com/en/database/oracle/oracle-database/19/haovw/index.html.

[7] Oracle Corporation. 2021. *Oracle Cloud Infrastructure Compute Shapes.* https://docs.cloud.oracle.com/en-us/iaas/Content/Compute/References/computeshapes.htm.

[8] Oracle Corporation. 2021. *Oracle Enterprise Manager.* https://www.oracle.com/uk/enterprise-manager/technologies/.

[9] Oracle Corporation. 2021. Oracle Exadata Database Machine. *datasheet* (2021). https://www.oracle.com/database/technologies/exadata.html.

[10] Standard Performance Evaluation Corporation. 2017. *SPEC, CPU2017 Integer Result Intel Xeon Platinum 8168, 2.70 GHz.* https://www.spec.org/cpu2017/results/res2018q4/cpu2017-20181211-10244.html.

[11] Transaction Processing Performance Council. 2021. *Transaction Processing Performance Council.* http://www.tpc.org/information/benchmarks5.asp.

[12] S. K Doddavula, M Kaushik, and A Jain. 2011. Implementation of a Fast Vector Packing Algorithm and its Application for Server Consolidation. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. IEEE, 332–339.

[13] Michael R. Garey and David S. Jounson. 1979. *Computers and Intractability: A Guide to the theory of NP-Completeness.* Freeman.

[14] Dominic Giles. 2019. *SwingBench 2.2 Reference and User Guide.* http://www.dominicgiles.com/swingbench.html.

[15] S. Halfpap and R. Schlosser. 2019. Workload-Driven Fragment Allocation for Partially Replicated Databases Using Linear Programming. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1746–1749. https://doi.org/10.1109/ICDE.2019.00188

[16] Abdul Hameed, Abdul Hameed, Alireza Khoshkbarforoushha, Alireza Khoshkbarforoushha, Rajiv Ranjan, Rajiv Ranjan, Prem Prakash Jayaraman, Prem Prakash Jayaraman, Joanna Kolodziej, Joanna Kolodziej, Pavan Balaji, Pavan Balaji, Sherali Zeadally, Sherali Zeadally, Qutaibah Marwan Malluhi, Qutaibah Marwan Malluhi, Nikos Tziritas, Nikos Tziritas, Abhinav Vishnu, Abhinav Vishnu, Samee U Khan, Samee U Khan, Albert Zomaya, and Albert Zomaya. 2016. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* 98, 7 (2016), 751–774.

[17] Antony S. Higginson, Norman Paton, Suzanne Embury, and Clive Bostock. 2017. DBaaS Cloud Capacity Planning: Accounting for Dynamic RDBMS Systems that Employ Clustering and Standby Architectures. In *: Proceedings of the 20th International Conference on Extending Database Technology.* https://doi.org/10.5441/002/edbt.2017.89

[18] Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, and Suzanne M. Embury. 2020. Database Workload Capacity Planning Using Time Series Analysis and Machine Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 769–783. https://doi.org/10.1145/3318464.3386140

[19] Bernhard Korte and Jens Vygen. 2006. Bin-Packing. In *Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics 21.* Springer, 426.

[20] Mohammad Masdari and Afsane Khoshnevis. 2019. A survey and classification of the workload forecasting methods in cloud computing. *Cluster computing* 23, 4 (2019), 2399–2424.

[21] Rathijit Sen and Karthik Ramachandra. 2018. Characterizing Resource Sensitivity of Database Workloads. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 657–669.

[22] Sukhpal Singh and Inderveer Chana. 2016. Cloud resource provisioning: survey, status and future research directions. *Knowledge and information systems* 49, 3 (2016), 1005–1069.

[23] Zhuoyao Wang, Majeed M Hayat, Nasir Ghani, and Khaled B Shaban. 2017. Optimizing Cloud-Service Performance: Efficient Resource Provisioning via Optimal Workload Allocation. *IEEE transactions on parallel and distributed systems* 28, 6 (2017), 1689–1702.

[24] Tao Yu, Jie Qiu, B Reinwald, Lei Zhi, Qirong Wang, and Ning Wang. 2012. Intelligent Database Placement in Cloud Environment. In *2012 IEEE 19th International Conference on Web Services*. IEEE, 544–551.

[25] Mingyi Zhang, Patrick Martin, Wendy Powley, and Jianjun Chen. 2018. Workload Management in Database Management Systems: A Taxonomy. *IEEE transactions on knowledge and data engineering* 30, 7 (2018), 1386–1402.