Constrained Optimization Modelling on Resource Allocation for Heterogenous Distributed Systems[*]

Teng Yu[†]

1st Jan. 2016

[*]M.Res. Research Project, Department of Computing, Imperial College London
[†]Corresponding email: *t.yu15@imperial.ac.uk*

Abstract

This work intends to improve the performance of resource allocation process on heterogeneous distributed systems by designing a novel allocation model. It provides a novel approach to involve order theory to model the allocation process and constraints by research on the relations between different resource allocation requests(RAr) and mapping with the structure inside the cluster of servers, then designing ranking functions on their topology as a metric for optimisation instead of viewing the underlining problem as a multi-dimensional bin-packing instance and framing the process as a special Linear Programming formulation which is common in the literature. It focuses on the modelling process and invokes well-known algorithms in experiment to compare its performance with other models. Finally, we discuss that this approach can be easily extended to apply on not only the fields within distributed systems (such as for cloud computing platform), but also in the wide fields of computer science such as for Information Retrieval and HPC.

Acknowledgements

First thanks to my parents. Sincere thanks to Dr. Mark Stillwell for his countless helpful comments and advise during this project. Thanks Prof. Alexander L Wolf for his supervision and my personal tutor Dr. Sadri Fariba for advice. Thanks for the Departmental Scholarship funded by the Department of Computing, Imperial College London

Warm thanks the following friends for their helps and interesting discussions related to this project: Joel Choo (Imperial College London), Shale Xiong (Imperial College London), Jialiang Wang (Harvard University), Bichen Shi (University College Dublin), Huayi Ji (University College Cork), Bojia Ma (City University of Hong Kong) and Hongyu Yang (Eindhoven University of Technology).

Contents

1 Introduction

Resource Allocation is a core problem for designing high-performance heterogeneous distributed systems. From a classical point of view, most of approaches in the literature modelled this process as a multi-dimensional bin packing problem[8, 17, 19, 15, 9] whilst powerful heuristics has been developed during the last several decades[13, 4, 3].

While by considering the nowadays real-world heterogeneous distributed systems, at least one new constraint should be added and one significant constraints has been broken compared with the classical case which also lead to the motivation of the work presented in this report: First, in addition of single resource allocation request (RAr) can contain multiple dimensions such as asking for some CPU capacity and memory space simultaneously, there can also be relations between RArs or say different RArs are not independent; Second, single RAr can ask for the capacity which must be provided by multiple servers simultaneously and then, this also leads to the fact that the difference between RArs may be shown by the relation between servers they needed for. In this case, if we still try to model the scenario as a typical multi-dimensional bin packing problem and solve by linear programming solver, we find the first constraint really increase the complexity of the model as we needed either to adding more dimensions to handle with it[15, 2, 7] or update old dimensions and constraints to show the relations[19] whilst the second condition thoroughly influence the performance of the solver as not only different balls can be put in one bins, different bins can also be used for one balls.

The novel approach presented in this report share a new light on this problem by focusing on the relations between different RArs and viewing the allocation process as a mapping between the topology of RArs and clusters of servers instead of an action which putting balls into bins. Compared with others pervious approach which also involving set-variables to build up the model[16], we consider more theorems from order and lattice theory to construct the efficient model. We first define the relations between the RArs and use partially ordered set (*poset*) to represent the initial topology. Then the most interesting achievement we obtained is by originally applying the Birkhoff's representation theory[5] to model this initial topology which precisely transfer the un-modular initial structure to a modular lattice through a bijection. Then, it is easy to design or invoke some ranking functions to label the nodes in this new modular structure and obtains the metrics we needed to analysis the RArs. We use the MPC-X[20] device by Maxeler Technologies as a concrete industrial example to present our result and show the generality of our theoretical analysis. we have implemented a prototype program based on our model by Java-choco solver[14] to compare the performance with the classical bin packing linear-programming model and receive satisfied result, while the main part of model implementation and real data comparison remained to be the future work of this project.

The remaining sections are generated as follow: section 2 detailed presents the background and related work of resource allocation for distributed systems and show the state-of-the-art. We illustrate the essential knowledge of order and lattice theory used in this work in section 3. The original research is described from the section 4 which gives the definitions of relations between RArs and build up the initial topology of basic independent RArs. Section 5 shows the more general and complicate RArs topology which considering the new conditions. Section 6 analysis our model through the MPC-X device and illustrate the approach by applying Birkhoff's theory, then show the ranking function. The evaluation architecture and results are presented in section 7. We give conclusions and describe the future work in section 8.

2 Background and Related Work

Modelling on resource allocation problem for variant distributed systems has been widely research in the literature[17, 18, 19, 15, 7, 2, 9, 22, 23, 11, 13, 8]. In this section, we illustrate the work by Stillwell, Vivien and Casanva in 2010 [17] first as it can be viewed as a common base of the background and related work in a basic homogenous case. Then we extend the discussion to the heterogeneous case by considering servers which can contains different types of resources as illustrated in [18] and considering the time-varying and shared-resource need as shown in [15, 7, 2]. After that, we presented the practical model designed in [23, 11] and briefly mentioned the recent work in [22] which considered the work-time issue for their model. We discussed the corresponding experimental models in [4, 13] and illustrated the novel modelling approach shown in [16] which motivated our work in this report.

2.1 Model on Homogenous System

The work in [17] detailed illustrated a typical approach to model the problem in a subset of general distributed systems, named the *shared hosting platforms*[1]. The authors view the allocation process as a multi-dimensional bin packing problem, model it as a Mixed Integer Linear Program then invoke different kind of algorithms to solve it and finally, evaluate the performance by some benchmark experiments. The main achievement they obtained is a formulation of this resource allocation problem in such a shared hosting platform for static workloads to make decisions when allocating hardware resources to service instances. As shown in **Figure 1**, they viewed the resource allocation as a bridge between the servers and services to construct the system model. They encapsulated homogeneous servers in cluster with high-speed switched interconnect and each server provided several resources (e.g., CPU, RAM space, I/O bandwidth, disk space) whilst they assumed each service consists of a single Virtual

[1]Shared hosting platforms can share cluster resources among services to achieve a trade-off between high utilisation and performance isolation[21]

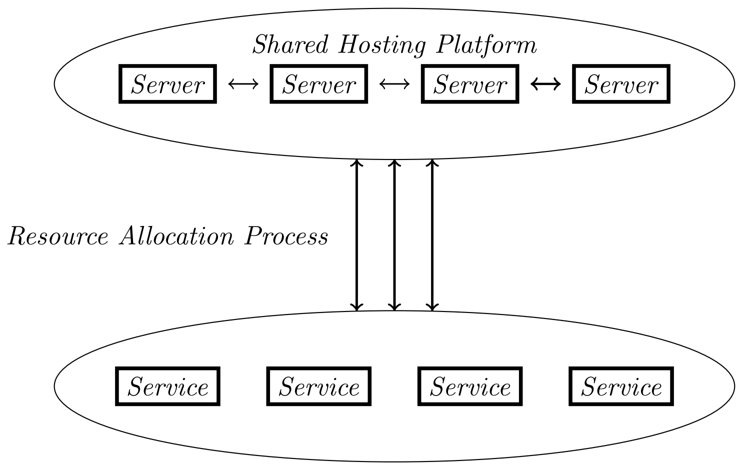

Figure 1: System Model

Machine(VM) instance[2]. As for resource allocation need, they classified as two types: *rigid* and *fluid*. A rigid need denote that a specific fraction of a resource is required. The service cannot benefit from a larger fraction and cannot operate with a smaller fraction. A fluid need specifies the maximum fraction of a resource that the service can use if alone on the server. The service cannot benefit from a larger fraction, but can operate with a smaller fraction at the cost of reduced performance. For each fluid resource need, they then defined the ratio between the resource fraction allocated and the maximum resource fraction potentially used. They named this ratio the *yield* of the fluid resource need. To compare yields across services with various minimum yield requirements, The *scaled yield* of a service is as follows:

$$\text{scaled yield} = \frac{\text{yield} - \text{minimum yield}}{1 - \text{minimum yield}}.$$

After that, they specified the assumptions as summaried below:
1) Each service consists of only a single VM instance.
2) Each service has constant resource needs.
3) The yields of all fluid resource needs are identical.
4) Rigid resource need are independent from fluid resource needs
5) User concerns, or say higher level metrics are directly related to resource fractions allocated to services

Followed that, they specified the aim precisely as *Maximize the Minimum Yield* under the two constraints below:
1) The resource capacities of the servers not be overcome.

[2]The authors also discussed an extension of their result in multi-VM services in section6 of their paper

2) A service inside a single VM instance should be allocated to a single server.

Then, consider the problem formulation as a Mixed Integer Linear Program (MILP). Use N services indexed by i, H servers indexed by h which each provides d types of resources. Fractions of these resources can be allocated to services. For each service i, r_{ij} denotes its resource need for resource type j, as a resource fraction between 0 and 1. δ_{ij} is a binary value that is 1 if r_{ij} is a rigid need, and 0 if r_{ij} is a fluid need. Give $\hat{y}_i$ to denote the minimum yield requirement of service i, a value between 0 and 1. Define a binary variable e_{ih} that is 1 if service i runs on server h and 0 otherwise. And define y_{ih} the unscaled yield of service i on server h, which must be equal to 0 if the service does not run on the server. With these definitions the constraints of our linear program are as follows, with Y denoting the minimum yield:

$$\begin{array}{rcr}
\forall i, h & e_{ih} \in \{0,1\}\ , \quad y_{ih} \in \mathbb{Q} & (1) \\
\forall i & \sum_h e_{ih} = 1 & (2) \\
\forall i, h & 0 \leq y_{ih} \leq e_{ih} & (3) \\
\forall i & \sum_h y_{ih} \geq \hat{y}_i & (4) \\
\forall h, j & \sum_i r_{ij}(y_{ih}(1-\delta_{ij}) + e_{ih}\delta_{ij}) \leq 1 & (5) \\
\forall i & \sum_h y_{ih} \geq \hat{y}_i + Y(1-\hat{y}_i) & (6)
\end{array}$$

This work also provide a powerful section discussing different types of algorithms they applied to solve the resource allocation problem. They analysis and evaluate the trivial solution by solving the MILP directly, 49 greedy algorithms by combining different service sorting strategies and server picking options, a genetic algorithm based on GAlib library[12] and 4 vector packing algorithms[3]. Finally, they proved that the Choose Pack Algorithm with sorting vector lists by decreasing the sum of coordinates was the most efficient approach. We ignore the detailed algorithms analysis[4] and evaluation part which is far beyond the scope of this report as we focus on the modelling processing of resource allocation.

2.2 Model on Heterogeneous Systems

As a common basic case, the above work was easy to extend for heterogeneous platforms. As discussed in their following work[18], each resource provided in different servers contains two different types of capacities, elementary capacity and aggregate capacity in this heterogeneous platforms. The first capacity represents a single element in one resource dimension whilst the second denotes the total resource capacity counting all elements. For example, a server node comprises 2 cores and single memory will have different elementary and aggregate

[3]Best Fit, First Fit, Permutation Pack and Choose Pack

[4]Please refer to the Appendices section for a brief illustration of the most efficient algorithm, VP_CPSUM: Vector Packing algorithm using choose packing and sorting ordered by decreasing sum of coordinates

capacity for CPU but same capacity for memory; A resource allocation request may have different or same need for elementary and aggregate capacity. The aggregate capacity need may also not be an integer multiple of the corresponding elementary capacity need. Compared with the homogenous MILP model in [17], it used different vectors to represent *rigid* and *fluid* resource allocation need[5] instead of involving a binary indicator. It added one more constraint to handle the heterogeneous case by representing the aggregate resource capacities through the sum of resource used from all services on a server node.

While by considering other part of problems occurred in the heterogeneous systems such as shared-dimensions resource request and dynamic workload, Rai, Bhagwan, and Guha illustrated a novel approach implemented by their resource allocation solver, named *Wrasse* in [15].

Wrasse solver defined a specification language for resource allocation, which was expressive enough to encode a multitude of allocation problems without using any domain-specific abstractions. In terms of bin-packing with domain agnosticism, they assumed any ball could be assigned to any bin, balls consumed resources provided by bins and did not have different types of balls or bins map to in the problem domain as the basic case. Same as the modelling in [17, 18], Wrasse's abstraction for resource was still a single multi-dimensional resource vector and each bin's resource were mapped to different dimensions in that vector and encoded the constraints using a Boolean variable for each ball-bin combination then only a linear number of variables needed. In addition, by exploiting the language features that one ball could be put in only one bin, it used one integer variable for each ball and this variable was set to a value that represent its bin.

The novel approach to handle with shared resources was presented by adding new dimensions in the corresponding vectors for those resources to map to. For example, in a scenario in which there are n bins with k resources each, and m shared resources, the Wrasse resource vector has $(k * n) + m$ dimensions. Each dimension has a capacity, fixed in the problem specification. They also gave a discussion for considering the conflicting constraints in different dimensions: say a strategy that is optimised for performance may not necessarily meet fault-tolerance requirement. And trivially, if we viewed those conflicts as new shared resources constraints, we could handle them through the above way as well. A special mechanism to solve this problem more efficient was considering *friends* and *foes* groups: they assigned related balls to the same bin and invoked user-defined function to encourage it by feedback but it cannot direct the search algorithm to explore it. All balls in a friend group would be assigned to the same bin while at least one ball in a foe group would be assigned to a different bin. In addition, total difference could be achieved by using pair-wise

[5]The authors renamed those two types of resource allocation needs by *requirement* and *needs* separately in this paper

foe group. They illustrated network virtualization which extending simple VM placement problem with network bandwidth requirements as a concrete example. They illustrated two pervious example physics frameworks: SecondNet[7] which used the virtual data centre (VDC). Communicating VMs were placed on servers so that they did not exceed the capacities of network links commenting those servers. Each network link in the data centre was represented as resource dimension with the link capacity as the resource capacity; Oktopus[2] with Virtual Cluster(VC): a virtual star topology with N VMs: a single virtual switch connected all VMs and gave bandwidth requirement associated with each virtual link.

Then it provided another novel mechanism to model the dynamic workload. Say that assigning a ball to a bin must increase the resource utilization along the appropriate resource dimensions while the static resource utilization cannot model resource utilization that arises from the dynamic assignment of balls to bins. This led to the need of designing a function of the dynamic assignment. One way was also applied user-defined imperative function. There were several practical considerations when designing such a function: Soft constrains: over-constrained scenario may accept solutions where a small fraction of constrains are violated: Give a probability with which the constraint must hold; Pinning: for ongoing process, where new balls are added, old balls evicted over time: Wrasse allows the problem specification to 'pin' some balls to bins so as to not perturb already assigned balls in the system; Number of bins: taking the maximum of the total usage divided by total capacity across all bin-specific resources gives us a lower-bound. High-level operation: for each bin, considers all unallocated balls in parallel. A example of this case was the Microsoft Assessment and Planning Tool (MAP)[1] which took input VM time-varying requirements as a function of time. For each original resource, it created one resource dimension per time-slot. Updated the scalar resource utilization and time-varying utilization similarly. Server utilization maight exceed temporally capacity by at most 10%.

They used real-time pair-wise bandwidth constraint as a concrete example for the case when both shared resources need and dynamic workload had to be considered.They captured the above features to design an utilization function: First stored the traffic matrix and routing information, Wrasse maintained a single variable (per dimension) for the current resource utilization; When a VM was to be placed on a solver, then for every other VM on different server, added the bandwidth requirement between this VM and the other VM on path only up to the lowest common ancestor. For other not been placed, added the requirement between this VM and the other VM on the path from this server to the root of the tree. For every VM that was placed on the same server, subtracted the bandwidth between the VMs for all links from this server to the root.[6]

[6]For more details of the heuristics they applied in the searching process, please refer to the Appendices section in this report. The interesting approach in this part is beyond the scope of this report.

They also discussed using GPU to make best use of the corresponding hardware: Instead of using CPU, threads computed by GPU can ideally execute the same code-path (on different data), collaborate to increase sharing of the limited on-chip memory and avoid expensive synchronisation and data-dependence.

Several limitations occurred in Wrasse solver: First Wrasse cannot determine non-existence: Wrasse was sound, but not complete; Second say minimization: Wrasse did not guarantee a 'minimal' solution; Third: Wrasse's modelling was still a non-trivial transformation based on the underlining NP-complete Bin-packing problem.

While from a more practical point of view, Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud[23, 11] analysis a more complicate system model, or say architecture, to simulate the resource allocation scenario which is more closed to the really world distributeted infrastructures. Instead of using bins and balls to represent, this model composed by Application Environment (AE), Local Decision Module (LDM), Global Decision Module (GDM) and datacenter which contains physical machines and VMs. Briefly, we can say that AE is faced to the application associated with specific performance goals; An application-specific LDM is associated with each AE to evaluate the process baed on the current workload using service-level metric and generate a utility function of the resource allocation; A GDM is used to interact with each LDM and the real-datacenter. It is the core of this system which contains the Constraint solver to determine the management actions based on the input of LDM's utility functions and datacenter's system-level performance metrics. It works like a black box compared with LDM. We prefer to refer to the fig.2 in [23] of the system graph.

There are two utility functions in LDM: a fixed service-level function and a dynamic resource-level function which is communicated with GDM and updated for every iteration. VM allocation vectors in LDM are used for building up the upper bound constraints given by each application.As for the GDM: It involves two sequential process, one for determine VM allocation vectors for each application (VM Provisioning) and then for placing VMs and PMs and achieving optimisation (VM Packing). Beyond the constraints formulations based on the CPU and Memory, an interesting approach in VM Provisioning is using a coefficient to allow the administrator to trade-off between the fulfilment of the performance goals and the cost of operating the required resources. Another interesting method is illustrated during the optimisation process in VM Packing: To minimise the number of migration required to reach the new VM-to-PM assignment, or say minimise the reconfiguration to provide a strategy with few interim steps and maximum degree of parallelism. In conclusion, their work presents an automatic virtual resource management system for practical distributed infrastructures. The main advantages include: it can automate the dynamic provisioning and placement of VMs; support for heterogeneous applications and workloads; support for arbitrary application topology.

By considering the working-time issue in the resource allocation problem, Shahin Vakilinia, Mustafa Mehmet Ali, and Dongyu Qiu's recent work [22] model the process as a job-scheduling scenario instead. It considers the randomness by involving Poisson processing during service time. Then it solves the new issues aroused by this model which including probability distribution and job blocking. It analysis the different cases for VMs, job sizes and release time, respectively to build their efficient model.

2.3 Model for Experimental Environment

The next important part for modelling the resource allocation is designing a efficient experimental model. Alberto Caprara and Paolo Toth's pervious work in [4] used classes of randomly generated instances to model the 2-dimensional case. In detail, they used two variables c,d to represent the bins capacities on different dimensions, respectively. Then used Wj and Vj to represent the weights of item j on different dimensions, respectively. Finally used u.d.[a,b] to represent the uniformly distribution of the value for weights of item in the interval [a,b]. There were eight dimensions in the experimental environment. The dimensions in the first six classes are independently sampled. In classes seven and eight, the dimensions are correlated. They achieve positive correlation by setup the domain of the Vj to be a monotonic function from Wj while achieve negative correlation by setup it to be a inverse function in domain.

Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder in [13] provided a more powerful approach to extend the pervious model on 2-dimensional case to multi-dimensions by setting any additional pairs of dimensions (2i-1) and (2i) correlated as dimension 1 and 2, while independent with the other dimensions. As for generate negative correlation across all dimensions: Random variables to denote the random distribution of balls in each dimension, totally 2 times of the dimensions. Multiply with an random coefficient from [10,40] and over two. Further noise by adding a random value from [0,1] and ignored the overweight item.

As for defining metrics on the experimental model, Mark Stillwell in [17] sampled the resource needs from a normal probability distribution and gave the following two metrics: Failure rate (fr) :the percentage of instances for which it fails to find a solution; Distance from bound (dfb): the difference between the achieved minimum yield and lower bound.

2.4 Model from Set Theory

Instead of the pervious work we mentioned above, Paul Shaw in [16] give a new approach by modelling the underlining problem in resource allocation: the bin packing problem from a view of *Set Theory*.

The motivation of this approach is by considering a dedicated constraint and the fact that using pruning and propagation rules with lower bound method to significantly reduce search on traditional one dimensional bin packing problem. The new constraint it obtained is by considering the feature of candidate set of each bin. Trivially, there is a bound that a ball whose size is larger than the bin capacity should not be viewed as in this bin's candidate set. It gives some more precisely mathematical notations to represent the relation and generate this condition to make it dynamically during the searching process. In detail, it gives the formulas of single item elimination and commitment. A important fact it mentioned is that any bin load is equal to the total size to be packed, minus the loads of all other bins as it is the only rule which communicates information between different bins.

Followed that, it illustrated the new model of the problem from the view of *Set Theory*: As we know the goal is to assign each variable an element from its domain, then the solution can be viewed as an assignment. During the solution construction, we obtain a partial assignment which is a set of current domains at each step. Then it can views the optimisation process as a domain reduction on variables. And it clearly illustrates that its constraint includes one parameter which is a vector of n element (n is the number of balls) indicating the index of the bin into which it will be placed. It also mentions that it is NP-complete to achieve generalised arc consistency in packing constraint, so instead it describes a not-complete algorithm by considering *neighbouring subsets* in candidate sets. It means two sets are neighbouring if there is no other subset in the candidate set whose items sum to a value strictly between themselves. It implement an linear time (even can be computed in constant time by dynamically updating) algorithm *NoSUM* to determine if there is such a interim subset exist. Then it is trivially to prune the problem instance by invoking this algorithm. The new lower bound it involved is by splitting the balls in candidate set into subsets using a constant number and apply the result to partial solutions. In conclusion, this new constraint cut search by orders of magnitude.

This approach really motivated our work in this report: our basic approach is also by modelling the relation of bins and balls by viewing one as the value in another's domain. In addition to just modelling the underlining abstract bin packing problem from the view of relations between different Sets, we try to model the actual resource allocation process as well and focus on the relations between different resource allocation requests and physical servers topology. Our approach extends the pervious work by applying Order and Lattice Theory instead of just the Set theory as we model more practical constraints including non-independent requests and share-server requests by focusing on the *ordering* inside the model's topology. We first illustrated the essential knowledges used in this project in the following section.

3 About Order and Lattice Theory

In this section, we present the essential underlining theorems used in this work from Order and Lattice Theory. We only indicate the definitions of those theories and statements of the theorems ignoring all the verification to make the report concisely. For detail illustration of the content, we refer to Davey and Priestly's well-known book[5].

3.1 Partial Orders

Definition 1 *Let P be a set. An partial order on P is a binary relation $\leq$ on P such that, $\forall\ x,\ y,\ z \in P$,*
(i) $x \leq x$, (ii) $x \leq y$ and $y \leq x$ imply $x = y$, (iii) $x \leq y$ and $y \leq z$ imply $x \leq z$.

Definition 2 *A set P equipped with an order relation $\leq$ is said to be an partially ordered set. We use the shorthand poset in this paper.*

The cover-relation between elements of a poset P is defined as follows:

$$\forall x, y \in P. x \prec y \Leftrightarrow x \leq y \text{ and } \nexists u. x \leq u \leq y.$$

We say that "y covers x" or "x is covered by y". Also, the notation $x \succeq y$ is used to indicate that x covers y, or x is equal to y.

Definition 3 *A map f from poset P onto poset Q is called an order-isomorphism iff,*

$$\forall x, y \in P, x \leq y\ if\ and\ only\ if\ f(x) \leq f(y)\ in\ Q.$$

Then we say poset P and Q are isomorphic iff there exist an order-isomorphism map between P and Q.

Definition 4 *Let X be a set. The power-set $\mathcal{Q}(X)$, consisting of all subsets of X, is ordered by set inclusion:*

$$\forall A, B \in \mathcal{Q}(X),\ A \leq B\ if\ and\ only\ if\ A \subseteq B.$$

Definition 5 *Let P be a poset and let $S \subseteq P$. An element $x \in P$ is an upper bound of S if $s \leq x$ for all $s \in S$. A lower bound is defined dually. The least element of the set of all upper bounds of S is called the least upper bound and the greatest lower bound is defined dually as well.*

Definition 6 *Let P be a poset and let $S \subseteq P$. We say S is a lower set of P iff $\forall x \in S$ and $y \leq x$, then $y \in S$.*

3.2 Lattices and Valuations

Definition 7 *Let P be a poset and let $S \subseteq P$. An element $x \in P$ is an upper bound of S if $s \leq x$ for all $s \in S$. A lower bound is defined dually. The least element of the set of all upper bounds of S is called the least upper bound and the greatest lower bound is defined dually as well.*

We use x $\sqcup$ y (read x join y) to represent the least upper bound of {x,y}, whilst use x $\sqcap$ y (read x meet y) to represent the greatest lower bound of {x,y}. We use $\bigvee$S to represent the join of S and $\bigwedge$S to represent the meet of S.

Definition 8 *Let L be a non-empty ordered set.*

If $x \sqcup y$ and $x \sqcap y$ exist for all x, $y \in L$, then L is called a Lattice.

Definition 9 *Given a lattice L. An element $x \in L$ is join-irreducible iff:*

$$x = a \sqcup b \text{ implies } x = a \text{ or } x = b \text{ for all } a, b \in L$$

Definition 10 *A function f on a lattice L, $f : L \to \mathcal{R}_0^+$ is a valuation iff*

$$\forall x, y \in L.\, f(x \sqcap y) + f(x \sqcup y) = f(x) + f(y).$$

Definition 11 *A lattice $L = (P, \sqsubseteq)$ is modular iff*

$$\forall x, y, z \in L.\, x \sqsubseteq z \Rightarrow x \ \sqcup \ (y \sqcap z) = (x \sqcup y) \sqcap z.$$

The following equivalence holds in any modular lattice L:

(i) $\forall x, y \in L.\, x \succ x \sqcap y \Leftrightarrow x \sqcup y \succ y.$

(ii) $\forall x, y, z \in L.\, x \succ y \Leftrightarrow x \sqcup z \succeq y \sqcup z.$

3.3 Birkhoff's Representation Theory

Definition 12 *Any finite modular[7] lattice L is isomorphic to the lattice of lower sets of the partial order of the join-irreducible elements of L.*

4 Basic RArs

In this section, we discuss the relation between basic resource allocation requests (RArs). We assume RArs are independent between different dimensions in the theoretical case and focus on the relation of RArs in each dimension to build up their foundation model.

[7]In Birkhoff's original definition, it used *distributed lattice* instead and give a corollary saying any distributed lattice is modular. We used the followed result directly in this report as the detail mathematical description of the original theorems and corollary with the corresponding proof is far away beyond the range of this work.

4.1 RArs Relations

Consider to handle with resource that have difficult-to-represent capacities. User's request may be special on different processing element (heterogeneous)or say the relation between them. Then we find the capacity of a cluster is the set of all allocation request that it can service. Try to give a relation between RArs:

Definition 13 *Given the relation $\preceq$ between RAr:*

$$\forall \alpha \in a\ cluster\ device, \exists A, B \in RAr, such\ that\ A \preceq B \ \Leftrightarrow \alpha(B) \mapsto \alpha(A).$$

We say $\alpha(B)$ is true iff α can service B. Then we can define a parallel relation between RArs as follow:

Definition 14

$$\forall A, B \in RAr, A \npreceq B \ \wedge B \npreceq A \ \Leftrightarrow A \parallel B.$$

Followed by defining relations between user's requests, we can try to build a closed topology involving those requests. It will be more likely a lattice as we can ordered those requests by inclusion through the relation we just defined. As the topology inside a cluster device can be viewed as a littice as well by considering that the intersection of subset servers in cluster, it leads to a possibility to achieve mappings during the allocating process.

A trivial approach from the order theory is to consider Birkhoff's representation theory, which says modular lattice can be represented by down-sets of its corresponding partial order composed by join-irreducible elements [5]. Then it is easy to view the RArs as downset of the lattice of cluster topology. Then it remains to create a metric or say ranking function to compare the performance between different mapping.

4.2 RArs Model

Consider there is a common foundation of all RArs which is a non resource needed request, we represent it as $\perp$.

Say there are some kind of basic RArs which hold only one dimension of resource need. For instance, one RAr, named A_1, may only need some CPU capacity for computation while another, named A_2, just need some memory capacity to record information. It is easy to verify that not all cluster devices which can service A_1 can service A_2 and vise versa. So in this case, A_1 and A_2 are parallel while $\perp \preceq A_1$,A_2. Then it is also easy to extend this instance to the real case that there will be a large number n of different dimensions of RArs, and we have the following two conditions:

$$1)\ \forall i, j \in n, A_i \parallel A_j$$

$$2)\ \forall i \in n, \perp \preceq A_i$$

Now we consider the union of those RArs A_i. Some more complexed RArs may contain multi-dimensional resource allocation requests, which can be viewed as request different one-dimensional RArs simultaneously and this can be achieve by the union of A_i. In this way, we can define a n-dimensional RAr as follows:

Definition 15 *Given A^n denote a n-dimensional RAr, then:*

$$A^n = (\forall A_i\ i \in n) \bigwedge A_i.$$

Finally, we say any kind of resource allocation request can be viewed as a n-dimensional RAr and the topology of RArs model can be built up as a partially order set through the relations we just defined. We give an instance of two-dimensional RAr topology in **Figure 2**. We use $A_1(1)$ to represent a one-dimensional RAr which only require 1GHz CPU and $A_2(1)$ to represent another one-dimensional RAr requiring 1GB memory. Then $A^2(1,1)$ is the union of them and represent a two-dimensional RAr. For easy illustration, we suppose 1GHz CPU and 1GB memory are two atom (or say entry) requests in this instance. As there may be 1.5GHz CPU request in a RAr, we use dash line to denote the relation which contains intern nodes and full line to denote the closed relation.

It is not a supervise that we find the result topology is actually a modular lattice: Each pair of nodes in this topology has a least upper bound and a greatest lower bound whilst all nodes satisfy the modular law presented in Definition 11. We ignore the mathematical proof of this result.

Note: Instead of the theoretical analysis as above, another way of generating the topology of resource allocation requests, or more generally say *informations*, is through real data experiment. One instance of this approach can be refer to [10] which tested the web-service requests and also resulted a poset.

5 General RArs

We begin to analysis the general case of RArs topology in this section. In the first subsection, we extend the work in last section by reducing the constraint of independence between different dimensions in a RAr. Then in the next subsection, we consider the scenario where a RAr ask for service from multiple servers simultaneously.

5.1 Non-independent RArs Model

Instead of the two independent dimensions we considered in above section, now let's focus on the case when there are two related dimensions in a single RAr. For instance, we can assume a RAr which need 2*Mb/sec* network link's capacity and 3*Mb/sec* I/O bandwidth. While it is easy to say any RAr containing

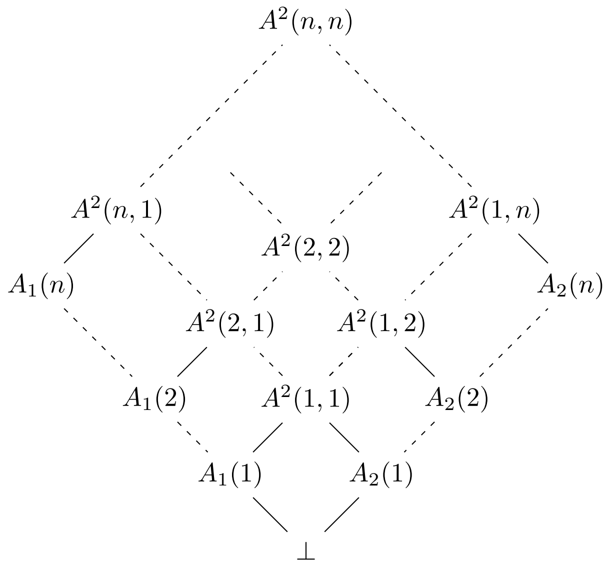

Figure 2: 2-dimensional(CPU, Memory) RAr topology

network link's capacity need will always need at least the same I/O bandwidth. Then, we say those two dimensions in a RAr are not independent. We use $B_1(1)$ to represent a one-dimensional RAr which only require 1*Mb/sec* network link's capacity and $B_2(1)$ to represent another one-dimensional RAr requiring 1*Mb/sec* I/O bandwidth. Then $B^2(1,1)$ is the union of them and represent a two-dimensional RAr. Same as the above section, we suppose 1*Mb/sec* network link's capacity and 1*Mb/sec* I/O bandwidth are two atom requests in this instance. We formalise the relation between those two RAr's dimensions as below:

$$\forall i \in \mathcal{N}^+, B_2(i) \preceq B_1(i)$$

Then it leads to the following corollary:

$$\forall i \in \mathcal{N}^+, B_1(i) = B^2(i,i)$$

Where $\mathcal{N}^+$ denotes the set of positive integers. The basic topology of RArs with those two dimensions is shown in **Figure 3**. Compared with the topology in **Figure 2** with two independent dimensions, this topology is a really general case of a lattice instead of a modular lattice. Say if we use three nodes: $B^2(1,2)$, $B_1(2)$ and $B_2(3)$ as a triple, then this triple will not satisfy the modular law: $B^2(1,2) \preceq B_1(2)$ but $B^2(1,2) \sqcup (B_2(3) \sqcap B_1(2)) \neq (B^2(1,2) \sqcup B_2(3)) \sqcap B_1(2)$. This topology will contains more such non-modular triples or non-modular sublattice if adding the values in each dimension.

$$\begin{tikzcd}
 & & B^2(2,3) \arrow[dl, no head] \arrow[dr, no head] & \\
 & B_1(2) = B^2(2,2) \arrow[d, no head] & & B_2(3) \arrow[ddl, no head] \\
 & B^2(1,2) \arrow[dl, no head] \arrow[dr, no head] & & \\
B_1(1) = B^2(1,1) \arrow[dr, no head] & & B_2(2) \arrow[dl, no head] & \\
 & B_2(1) \arrow[d, no head] & & \\
 & \perp & &
\end{tikzcd}$$

Figure 3: 2-dimensional(Network Link, I/O Bandwidth) RAr topology

5.2 Multi-server RArs Model

We consider the case when a RAr can ask for resources provided by different servers simultaneously in this section. From a traditional bin-packing point of view, not only one bin can contains multiple balls but also one ball can be separated and put into different bins in this scenario. Instead of based on some example RArs, we analysis the relation through the general multi-server RArs directly.

It is interesting to find that when we totally refer to the relations between different multi-server RArs, the topology is much more simple than the above cases as we obtained a trivial linear-order. As shown in **Figure 4**, we use $S(i)$ to denote a general RAr which ask for i server's resources simultaneously. The linear-order set we obtained is constructed by the trivial relation or say constraint between those RArs, say:

$$\forall i \in \mathcal{N}^+,\ S(i) \preceq S(i+1)$$

While the structure becomes complicate when we have to consider the relations between different servers. In the real industrial case, heterogeneous servers will provide different types or amount of resources even the number of them are the same. Further more, the difference between servers more also led by the physics topology they are allocated. For instance, two neighboured servers in a distributed system can provide resources with less network delay and enjoy more traffic tolerance than other pairs of servers. We refer to the next section as a concrete industrial example to illustrate our approach on a real-world distributed system by modelling it as a poset then transferring to a modular lattice.

$$\begin{tikzcd}
\vphantom{S} \arrow[d, dash, dotted] \\
S(3) \arrow[d, dash] \\
S(2) \arrow[d, dash] \\
S(1) \arrow[d, dash] \\
\perp
\end{tikzcd}$$

Figure 4: Multi-server RArs topology

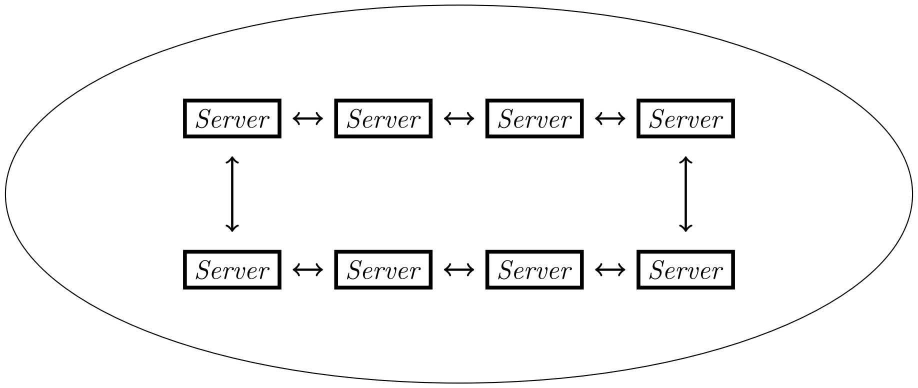

Figure 5: MPC-X device

6 MPC-X RArs

Now we consider the MPC-X[20] system with the additional constraints between RArs as a practical case. This is a real extended scenario that a RAr can request different server to work for it or say allocate to it simultaneously which break the classical constraints for any bin-packing problem model as they all limit that one ball can only be allocated to exactly one bin.

In this section, we first briefly describe the topology of MPC-X device and then model the RArs based on its special relations. Finally we discuss some potential approaches to build up metric.

6.1 MPC-X topology and RAr relations

An abstract MPC-X device contains eight servers arranged in a ring as shown in **Figure 5**. We assume the servers are homogeneous in a device. Then there are two types of RArs: one for singleton servers whilst another for adjacent servers.

We use S_n (n=1...8) to represent the first type RArs while A_n (n=1...8) to represent the second and in both of them, the index represent the number of server it requested.

Then it is easy to say the foundation relation below:

$$1)\ \forall i \in (1, ..., n-1), S_i \preceq S_{i+1}$$

$$2)\ \forall i \in (1, ..., n-1), A_i \preceq A_{i+1}$$

$$3)\ \forall i \in (1, ..., n), S_i \preceq A_i$$

$$4) \qquad A_1 = S_1$$

$$5) \qquad A_7 = S_7$$

$$6) \qquad A_8 = S_8$$

$$5) \qquad A_2 \preceq S_5$$

$$6) \qquad A_3 \preceq S_6$$

6.2 MPC-X RAr model

We give the RArs model of the MPC-X device below **Figure 6**, ignored the totally-ordered part for easy illustration. As mentioned in section 4.1, we apply Birkhoff's representation theory to map this partial order set to a downset lattice which can be referred to **Figure 7**. We use B_i to label the graph instead of using the downset label directly to make it more concisely. To illustrate the correspondence, for example, we know B_1=S_2 while B_{14} is the downset of $S_6 \cup A_4$ which contains $\{S_6, S_5, S_4, S_3, S_2, A_4, A_3, A_2\}$.[8]

Note: A general result we can inferred from this instance is that the RAr topology from any practical or say industrial distributed systems can be modelled as a modular-lattice by our representation and Birkhoff's theorem.

Let's then focus on the new modular-lattice model, actually it contains some really additional *interesting* nodes which we just ignored in our first MPC-X RAr topology. For instance, we still use the B_{14} we just mentioned above for illustration. There cannot be any single node in the pervious topology to exactly represent the scenario of the union of S_6 and A_4. (If we say A_6, then we find that its capacity is beyond our needs as a scenario in which six servers can work while five of them are adjacent can serve our need but it doesn't satisfy A_6.) On the other hand, the total sets of node in this new modular-lattice model can represent all the possible scenarios of RAr for the device based on its downward-close character.

[8]To keep the main body of this report concisely, we give the detailed table of the corresponding representations between those two topologies in the Appendices section

6.3 Metric on RAr model

At this step, we can begin to consider a metric on this lattice-model to represent the distance between different RArs to direct the allocation. A common metric on modular-lattice is a *valuation*. Recall the formula of valuation in definition 10. A function, or say labelling method which satisfying the valuation law will efficiently show the distance between different RArs as the sum of two different RArs' label will equal to the sum their least upper bound and greatest lower bound. In another word, valuation on the nodes of RArs' lattice topology keep the consistency of ability that to what level of RArs it can serve as shown in the corresponding downset.

A easy method to compute a valuation is by a ranking function. We refer to Gratzer's book [6] for the proof that the following ranking function satisfy the valuation law as it is beyond the range of this work.

Definition 16 *Let L be a modular lattice. The ranking f on L is a function* $f : L \to \mathcal{R}_0^+$ *that:* $\forall$ *x,y* $\in$ *L:*

$$(1)\ f(x) = f(y) + 1 \ \Leftrightarrow y \preceq x;$$

$$(2)\ f(x) = 0 \ \Leftrightarrow x = \perp.$$

This function is easy to compute on any modular lattice in polynomial time complexity, or say within O(nlogn) in which n is the amount of RArs. Based on a greedy approach, we can first define the ranking of a empty RAr to be zero as it must be a bottom in the topology and label the following nodes through binary search[9] in which we labeled by adding one to each child node from its parent node iterately.

7 Evaluation

We illustrate some basic experimental evaluations in this section. Say that we only implement some basic prototype programs at this step of the project. The whole program implementation by the novel model illustrated in this report refer to the future work of this two-term project.

7.1 Evaluation architecture

Two prototype programs are implemented in this step. The first one is a one-dimensional bin packing program by basic linear programming approach. It contains a tiny input size composed by 3 balls and 2 bins. We also implemented a extension vision of this model to handle two-dimensional bin packing problem which is trivially extended from the first one by parallel adding the second dimension. It contains a little larger input size compared with the first one which

[9]This is by the case that in modular lattice, one node will connect with at most two nodes in one direction.

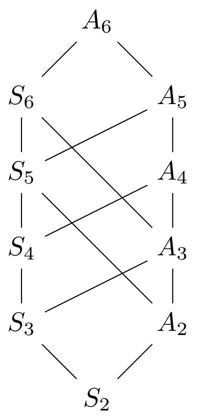

Figure 6: MPC-X RAr topology

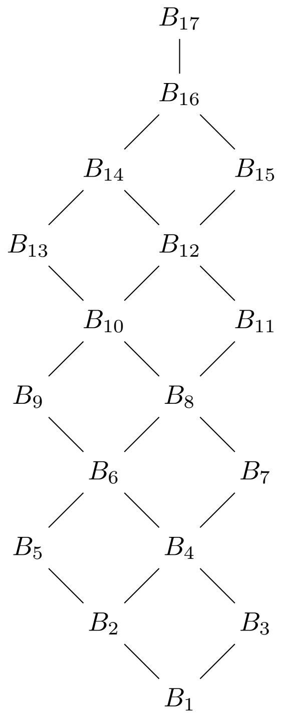

Figure 7: MPC-X RAr lattice-topology

Statistics Results by Novel Model							
No.	Building time	Initial propagation	Resolution	Nodes	Backtracks	Max depth	Constraints
1	0.0053s	0.010s	0.016s	22	20	21	4
2	0.0052s	0.009s	0.016s	22	20	21	4
3	0.0051s	0.010s	0.017s	22	20	21	4
4	0.0047s	0.010s	0.017s	22	20	21	4
5	0.0065s	0.010s	0.018s	22	20	21	4
6	0.0054s	0.009s	0.016s	22	20	21	4

we used as the basic input instance for multi-dimensional pin packing program.

Our new model is implemented from the third prototype program which is a one-dimensional bin packing program with the same tiny input size of program one, but the allocation is represented by a lattice which using the domain of bins to represent the balls and the values in the domain to represent the size of each bin.

We apply the same well-known greedy search algorithm as a benchmark of our following experiment.

All of the programs are implemented in Choco3 solver [14] by Java in Eclipse IDE (Vision: Luna Service Release 1a (4.4.1)). The experimental environment is in a MacBook Pro laptop with 2.8GHz Intel Core i7 CPU; 8 GB 1600 MHz DDR3 Memory; and OS X 10.9.2 Operation System.

7.2 Evaluation Results

We first compare the programs through trivial linear-programming bin packing model and our novel model. We show our solutions in the two tables which indicate the statistics results in trivial model and our novel model, respectively. When both of them find the same solution, the model's tuning-time related metrics: building time, Initial propagation and resolution used for program by novel model are all less than the program by trivial model. In addition, the number of constraints applied in our novel model are less than in trivial model.

The interesting results we find is that our model obtains the efficient solutions although it actually suffers more intern nodes and backtrack and reach larger max depth during the searching process through the same search algorithm. It is corresponding to the fact that the model by lattice when always search all the internal subset during the path even it cannot be the solution.

Statistics Results by Trivial Model							
No.	Building time	Initial propagation	Resolution	Nodes	Backtracks	Max depth	Constraints
1	0.0076s	0.013s	0.020s	4	2	3	5
2	0.0076s	0.014s	0.019s	4	2	3	5
3	0.0083s	0.010s	0.015s	4	2	3	5
4	0.0077s	0.012s	0.017s	4	2	3	5
5	0.0077s	0.011s	0.016s	4	2	3	5
6	0.0078s	0.011s	0.016s	4	2	3	5

8 Conclusions and Future work

This project provides a novel approach to model the resource allocation problem on heterogenous distributed systems by focusing on the relations between resource allocation requests (RArs) and servers. Modelling the resource allocation process as bin-packing problem has been common used in the literature and even viewing this underlining problem from the set theory point of view has been discussed in the pervious work. The novel feature of our work is originally applying Birkhoff's representation theorem from order and lattice theory to represent the topology of RAr as a modular structure which is much more efficient compared with just modelling them as sets and building up constraints through the values in domain. Instead of using linear-programming approach to solve the constrained optimisation problem, our model is constructed based on representation, or say mapping between the structures of RArs and cluster of servers. In this report, we have detailed illustrates the RAr models from the basic homogenous case to the general heterogeneous and practical case. We invoke ranking function on the lattice structure to give metric on the topology and compute the difference between RArs to direct the allocation.

The main achievement of this project is that we successfully model general heterogenous distributed systems, such as the MPC-X device to a modular lattice instead of a general unconstrained structure. Then we can give easy but efficient metric on RArs to such systems. Our novel model shares a new light on handling resource allocation problem and the underlining multi-dimensional bin-packing problem and specially suited for the case when there are multi-servers RArs and constraints between heterogenous servers.

We have evaluated the efficiency of our approach in the basic case by solving a trivial bin-packing problem while the implementation of a complicate programme to solve a real resource allocation problem through our model and compare the performance with traditional methods remains to be the future work. Other interesting extensions include applying our model on different practical distributed systems and HPC devices. As our model give a new metric on general RArs, it will also be possible to apply it on an Information Retrieval programme by defining efficient information order and distance.

The main limitation of our model currently is that it doesn't solve the work-time issue: Our model is based on the bin-packing underlining problem instead of a job-scheduling scenario. Recent work such as [22] model the resource allocation as job-scheduling to handle the work-time issue while our model focuses on the static scenario. It is possible to extend our model to solve job-scheduling problem as well but updating the relations and topology between different RArs may thoroughly reduce the efficiency of this model.

9 Appendices

9.1 VP_CPSUM algorithms

We give high level descriptions of VP_CPSUM algorithm in this subsection as it is the most efficient methods for the basic homogenous model. We first illustrate the VP_PPSUM.[10] below as it provides the basic structure for designing VP_CPSUM algorithm.

VP_PPSUM algorithm

1.place each of the N vectors in on of d!/ (d-ω)! Lists, where $\omega \in$[1,...,d]
⇒ Each list contains the vectors with a common permutation of their largest ω dimensions.
2.Vectors in each list are then sorted according by decreasing sum of the coordinates (SUM).
3.Filling bins with vectors, each time attempting to reduce the resource load imbalance in a bin.
⇒ When ω=2, first look in list (i,j) for the first vector that can fit in the bin, hope to reduce the resource imbalance.
⇒ If not, relaxes the ordering of the components and searches in other lists.
⇒ If no vector can fit in the current bin, then a new bin is added and the process is repeated until all vectors are placed in bins.

VP_CPSUM algorithm

A relaxation of the VP_PPSUM algorithm in that it does not enforce any ordering between the ω coordinates of vectors and thus need only d!/ω!(d-ω)! Lists[11].

[10]VP_PPSUM: Vector Packing algorithm using permutation packing and sorting ordered by decreasing sum of corrdinates

[11]Giving ω=2 leads to good empirical result has been shown in [8]

Bin-centric view

Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi WiederFrom in [13] illustrate the same algorithm[12] from a bin-centric view: Open only one bin at any time, place items into this bin from the largest suitable one. Close the bin when no item can be put in. Then there are two heuristics by involving random choosing. Grasp[k]: pick a random one from the best k instead of the bext one; Bubblesearch: the kth best is chosen with a propobability proportional to $(1-p)^k$, for a suitable p.

Algorithm Analysis

For fixed d and ω, both algorithm have complexity O(nlogn). In addition, as analysis in [13], for the case when some demands in a dimension always dominate the demands in the other dimension, VP_CPSUM is more robust. The dimensions which are not scare can be assigned smaller coefficients and have a smaller impact on the ordering.

9.2 Heuristics in Wrasse solver

This subsection illustrated some heuristics used during the searching processing in the Wrasse solver [15]. Those materials are interested for designing efficient resource allocation algorithm but not closed to the modelling process so we keep them in the Appendices.

Wrasse invokes the user defined *dynamic* utilization function with the partial assignment. In briefly: first checks that capacity constraints are met. After that assign friend balls, check for group-consistency, if violated then left them as unallocated. When no more balls can fit, the solver mores on to the next bin.

As for the generic search algorithm it used: When picking balls and bins, different choices optimize for difference outcomes. To balance utilization across bins, it uses power of two sized bin-groups, exploit real-world spatial coherence in bins while being domain agnostic. Wrasse solver tries several size in parallel instead of requiring the user to specify in this process and searched the space by exploration: each thread-group independently exploring its own partial solutions as deep as possible. There were four reasons for choosing explore rather than exploit in the wrasse solver: 1. Define 'good' partial solution in a domain-agnostic manner is problematic; 2. We can retain early lucky decisions; 3. After several steps, we end up with a larger diversity of assignments; 4. Do not need addition heuristics.

Analysis by evaluation: For the VM placement problem, Wrasse is able to match the performance of the carefully tuned, production level placement heuristics.

[12]They named their algorithm as FFDAvgSum which means First Fit by Decreasing Average Sum and is the same as VP_CPSUM in [17]

For the network virtualization: Both heuristics find the smallest sub-tree in the network that can for a Virtual Cluster. Wrasse's parallel search for allocations provides high-quality solutions within a few seconds.

9.3 Table of MPC-X topology representation

We provide a detail table of the representation from initial RArs topology of MPC-X device to its corresponding modular lattice topology for easy understanding. Elements in the first column are nodes in the modular lattice after Birkhoff's representation. The second column contains the name of the corresponding downset which the first column denoted. The third column represents the corresponding initial RArs to the MPC-X device which the downset in the second column contains or say the node in the first column can server.

Node in lattice-topology	**Corresponding Downset**	**Containing RArs**
B_1	$S_2\downarrow$	$\{S_2\}$
B_2	$S_3\downarrow$	$\{S_3,S_2\}$
B_3	$A_2\downarrow$	$\{A_2, S_2\}$
B_4	$(S_3 \text{ U } A_2)\downarrow$	$\{A_2, S_3, S_2\}$
B_5	$S_4\downarrow$	$\{S_4, S_3, S_2\}$
B_6	$(S_4 \text{ U } A_2)\downarrow$	$\{A_2, S_4, S_3, S_2\}$
B_7	$A_3\downarrow$	$\{A_3, A_2, S_3, S_2\}$
B_8	$(S_4 \text{ U } A_3)\downarrow$	$\{A_3, A_2, S_4, S_3, S_2\}$
B_9	$S_5\downarrow$	$\{A_2, S_5, S_4, S_3, S_2\}$
B_{10}	$(S_5 \text{ U } A_3)\downarrow$	$\{A_3, A_2, S_5, S_4, S_3, S_2\}$
B_{11}	$A_4\downarrow$	$\{A_4, A_3, A_2, S_4, S_3, S_2\}$
B_{12}	$(S_5 \text{ U } A_4)\downarrow$	$\{A_4, A_3, A_2, S_5, S_4, S_3, S_2\}$
B_{13}	$S_6\downarrow$	$\{A_3, A_2, S_6, S_5, S_4, S_3, S_2\}$
B_{14}	$(S_6 \text{ U } A_4)\downarrow$	$\{A_4, A_3, A_2, S_6, S_5, S_4, S_3, S_2\}$
B_{15}	$A_5\downarrow$	$\{A_5, A_4, A_3, A_2, S_5, S_4, S_3, S_2\}$
B_{16}	$(S_6 \text{ U } A_5)\downarrow$	$\{A_5, A_4, A_3, A_2, S_6, S_5, S_4, S_3, S_2\}$
B_{17}	$A_6\downarrow$	$\{A_6, A_5, A_4, A_3, A_2, S_6, S_5, S_4, S_3, S_2\}$

9.4 User Guide

As illustrated in above section, we use Java-Choco [14] solver and Eclipse IDE to implement the experimental programme. We provide a concise guideline in this section about how to install and use it.

The Java 8 can be downloaded from its official website by Oracle through the following URL:

$$https://www.java.com/en/download/$$

The Eclipse IDE can be downloaded from the URL:

$$https://eclipse.org/downloads/$$

The Choco solver should be added in the project's referenced libraries and the jar file we used is *choco-solver-3.3.3-with-dependencies.jar* which can be free downloaded from the URL:

$$http://choco-solver.org/Download?q=releases$$

The source codes of our prototype programs are open and can be downloaded from the project account in Imperial College's Gitlab:

$$https://gitlab.doc.ic.ac.uk/ty215/MResProjectIC.git$$

References

[1] Microsoft Assessment and Planning Toolkit (MAP). *http://www.microsoft.com/map/.*

[2] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.

[3] Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 697–708. IEEE, 2006.

[4] Alberto Caprara and Paolo Toth. Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, 111(3):231–262, 2001.

[5] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order.* Cambridge university press, 2002.

[6] George Grätzer. *Lattice theory: foundation.* Springer Science & Business Media, 2011.

[7] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*, page 15. ACM, 2010.

[8] William Leinberger, George Karypis, and Vipin Kumar. Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In *Parallel Processing, 1999. Proceedings. 1999 International Conference on*, pages 404–412. IEEE, 1999.

[9] Siva Theja Maguluri, R Srikant, and Lei Ying. Heavy traffic optimal resource allocation algorithms for cloud computing clusters. *Performance Evaluation*, 81:20–39, 2014.

[10] Heikki Mannila and Christopher Meek. Global partial orders from sequential data. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 161–168. ACM, 2000.

[11] Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud. Autonomic virtual resource management for service hosting platforms. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8. IEEE Computer Society, 2009.

[12] GAlib: A C++ Library of Genetic Algorithm Components. *http://lancet.mit.edu/ga*, 2010.

[13] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. *research. microsoft. com*, 2011.

[14] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco3 Documentation.* TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014.

[15] Anshul Rai, Ranjita Bhagwan, and Saikat Guha. Generalized resource allocation for the cloud. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 15. ACM, 2012.

[16] Paul Shaw. A constraint for bin packing. In *Principles and Practice of Constraint Programming–CP 2004*, pages 648–662. Springer, 2004.

[17] Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. Resource allocation algorithms for virtualized service hosting platforms. *Journal of Parallel and Distributed Computing*, 70(9):962–974, 2010.

[18] Mark Stillwell, Frédéric Vivien, and Henri Casanova. Dynamic fractional resource scheduling versus batch scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):521–529, 2012.

[19] Mark Stillwell, Frederic Vivien, and Henri Casanova. Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 786–797. IEEE, 2012.

[20] Maxeler Technologies. *MPC-X, https://www.maxeler.com/products/mpc-xseries.* Maxeler Tech., 2015.

[21] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1, 2008.

[22] Shahin Vakilinia, Mustafa Mehmet Ali, and Dongyu Qiu. Modeling of the resource allocation in cloud computing centers. *Computer Networks*, 91:453–470, 2015.

[23] Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud. Sla-aware virtual resource management for cloud infrastructures. In *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*, volume 1, pages 357–362. IEEE, 2009.