Data center disaggregation: when and how much?

Thomas Begin
Université Claude Bernard Lyon 1, ENS de Lyon, Inria, CNRS
Lyon, France
thomas.begin@univ-lyon1.fr

Alexandre Brandwajn
Jack Baskin School of Engineering University of California, Santa Cruz
Santa Cruz, USA
alexb@soe.ucsc.edu

Alain Tchana
ENS de Lyon, Inria, CNRS Université Claude Bernard Lyon 1
Lyon, France
alain.tchana@ens-lyon.fr

Abstract—**Disaggregation has been proposed by both the industry and academia as an approach to reduce resource fragmentation in cloud data centers. With disaggregation the entire rack is viewed as a single machine in which all resources are remotely accessible from anywhere in the rack. Because a large number of resources may be used remotely, the average access time to resources tends to be elongated compared to a non-disaggregated rack organization. This elongation represents a technology penalty of disaggregation and impacts application performance. This paper presents a model to capture basic trade-offs between virtual machine (VM) lifetime elongation and improved memory mutualization in a rack with disaggregation. Our results show that disaggregation has the potential to significantly improve the maximum VM launch throughput provided the technology penalty remains moderate. Additionally, our model provides guidance for the number of servers to combine through disaggregation.**

Index Terms—**Cloud Computing, Virtualization, Disaggregation, Data Center Architecture, Analytical Model, Performance Evaluation**

I. Introduction

The use of cloud computing has skyrocketed over the last decade, due to its attractive cost model. Virtualization is the keystone technology making cloud computing possible through flexible allocation/de-allocation of resources (in the form of Virtual Machines, "VMs") and increased server usage. Unfortunately, current virtualization techniques operate at physical machine granularity, which often leads to resource fragmentation and underutilization and hence considerable financial cost for operators. For example, Microsoft estimates that a 1% reduction in fragmentation within its Azure cloud platform would result in hundreds of millions of dollars in savings [1].

Resource fragmentation is mainly due to a combination of two factors: (1) the multi-dimensionality of server resources (CPUs, memory, disk, network, etc.) and (2) the diversity of types of virtual machine instances offered by operators to cover a large spectrum of customer needs.

Among the approaches that have been proposed to reduce fragmentation, *rack disaggregation* [2]–[8] appears to be the most promising candidate. It consists in enabling PMs of a same rack to share their hardware resources (CPUs, main memory, persistent storage, etc.) so that the whole set of distinct resources in the rack can be treated as if it was one single, large resource that can be accessed by different PMs. Hence, in this model, a data center can be seen as a set of very large machines (the racks), that can each be dynamically partitioned into VMs with flexible resource allocations.

Because in a disaggregated rack organization VMs can access a large number of potentially remote resources, the average access time to resources such as RAM can be expected to be longer than in a classical rack without disaggregation. We refer to this elongation as the technology penalty of disaggregation. The extent of this *technology penalty* is expected to depend on the specific design and implementation choices for the disaggregated rack [8]–[10]. In particular, Gao et al. [2] studied the impact of network latency in a disaggregated rack.

Overall, it appears that disaggregation may not always improve the general behavior of a data center [11], [12]. In fact, the technology penalty of disaggregation is symptomized by the increase of VM lifetime, which subsequently may prevent a VM launch request because of a lack of memory. In this paper we propose a high-level model to assess the potential performance impact of disaggregation and to determine to what extent it is beneficial. By considering on the one hand memory as the limiting resource for VM packing [3], [13] and on the other hand increased memory latency and VM lifetime in disaggregation as its technology penalty, our model captures the overall impact of disaggregation using the number of successful VM launch requests per time unit as the performance metric. Additionally, our goal is to determine the "right" number of physical machines to combine through disaggregation. Our high-level model concentrates on the specific disaggregation trade-offs discussed above. It does not purport to model a specific cloud system or a specific implementation. It re-purposes a published model of resource sharing [14] with an existing analytical solution. We believe that, for the purposes of our study, it is the right tool compared to general cloud simulation models requiring a much greater level of detail [15]–[17].

II. Memory utilization Model

We focus our work on memory allocation to VMs at launch time assuming no memory over-commitment and infinite capacity for other resources. We consider two rack organization types: classic, in which each physical machine (PM)'s memory is private, and disaggregated, in which the memory banks of all PMs are mutualized and seen as a single memory pool.

We outline a high-level analytical model whose goal is to capture the salient effects of these two rack organization choices in order to assess quantitatively the potential for performance improvement/degradation due to disaggregation. Figure 1 shows an example of memory utilization with three classes of VMs. We use the maximum number of VMs that can be launched (packed, activated) per time unit as the relevant performance metric.

To account for different workloads, we consider three classes of VMs with respect to their memory size requirements: "small", "medium" and "large", denoted by b_s, b_m and b_l, respectively. Additionally, VMs are characterized by their expected lifetime t_x ($x = s, m, l$, according to the VM class). We study different "mixes" of these VM classes with different proportions of each class.

We denote by N the number of PMs in the rack, and we let B_j be the amount of memory available on jth PM, ($j = 1, \dots, N$) for VM launch requests. Thus, in the disaggregated architecture, the VMs share a single memory pool of size $B = \sum_{j=1}^{N} B_j$. With the classic architecture, each PM and the VMs it hosts are limited to the PM's private memory. While disaggregation has the clear benefit of consolidating the memory of all PMs, it potentially leads to an overhead due to the utilization of remote memory. This *technology penalty* of disaggregation is represented in our model by the increase of the VM lifetime by a given percentage denoted by α. The expected task execution time with disaggregation is correspondingly elongated.

Our model represents VM launch requests (initiated by cloud tenants) as generated by a large set of request sources. In our model, shown in Figure 2, an arriving request needs to find enough memory available to be satisfied. In the case of a disaggregated architecture this is the residual memory available in the global memory pool. In the case of the classical architecture, it is the residual memory available at the individual PM. When enough memory is available to satisfy a VM launch request, the amount of memory corresponding to the VM class is held by the VM during its entire lifetime. Otherwise, the VM launch request is rejected and will be retried some time later.

Clearly, for a given rack memory, the availability of sufficient memory to launch a new VM of a given size depends on factors such as the rate of VM launch requests, the lifetime of VMs and their memory requirements. The proposed model captures interactions between these factors.

We now briefly describe the probabilistic model used. A resource of total quantity R is shared by c classes of requests. Referring to class j, $j = 1, \dots, c$, we denote by r_j the amount of resource required by a single request and by $1/\mu_j$ the mean time the request holds the resource. There are N_j sources generating class j requests and the mean time for a source to generate a new request upon completion or rejection of a preceding request is denoted by $1/\alpha_j$. Arriving requests that don't find enough resource left are rejected and the corresponding source will generate a new request after a mean time of duration $1/\alpha_j$. The resource holding times and the times to generate a new request are assumed to be exponentially distributed. It is worthwhile mentioning that, in steady state, loss system models tend to be insensitive to distributions of resource holding times, as well as of the request generation/inter-arrival times under rather general conditions [18].

We consider the above model in the steady state and we let $p(m_1, \dots, m_c)$ be the corresponding probability that there are m_j requests of class j, $j = 1, \dots, c$, using the resource. The state-transition diagram for our model is illustrated in Figure 4. Customary performance metrics can then be calculated as follows. The attained throughput for class j can be computed as

$$\theta_j = \sum_{m_1,\dots,m_c} p(m_1, \dots, m_c) m_j \mu_j.$$

The offered throughput can be written as

$$\phi_j = \sum_{m_1,\dots,m_c} p(m_1, \dots, m_c)(N_j - m_j)\alpha_j.$$

The probability that a request is rejected for lack of resource can be expressed as

$$(\phi_j - \theta_j)\phi_j.$$

As shown in [14], the model has a simple closed-form solution for the steady-state probabilities $p(m_1, \dots, m_c)$, which enables a fast solution to the model. We study the model for given attained throughputs, which are used to determine the rates α_j (an implementation of the model is available for download [19]). We set the numbers of request sources such that $N_j >> [R/r_j]$ so that their precise values matter little. As mentioned before, for simplicity, we limit the number of classes in our application of the model to three. We map VM launch requests to arriving requests in the model and VM lifetime to resource holding times. The total resource amount corresponds to the available memory size: $R = B$ or $R = B_i$, depending on the architecture considered. In the case of a classic architecture, we assume that the sources are divided evenly across the PMs. We use this model to assess the respective maximum numbers of VMs per time unit that can be launched in the two architectures considered under the condition that an overwhelming majority of VM launch requests finds enough available memory for the VM to launch successfully. For the purposes of this study, we take it to be at least 99% of VM launch requests.

III. Results

We start by configuring our three VM classes as follows: $b_s = 2$ GB, $b_m = 4$ GB, $b_l = 8$ GB. We consider the following three "mixes" of these classes: in mix$_1$ 50% of VMs are small, while medium and large VMs represent each 25%; in mix$_2$, medium VMs represent 50% while the other two classes count for 25% each; in mix$_3$ large VMs represents 50% while small and medium VMs each account for 25% of the mix. In other words, small, medium and large VMs dominate in mix$_1$, mix$_2$

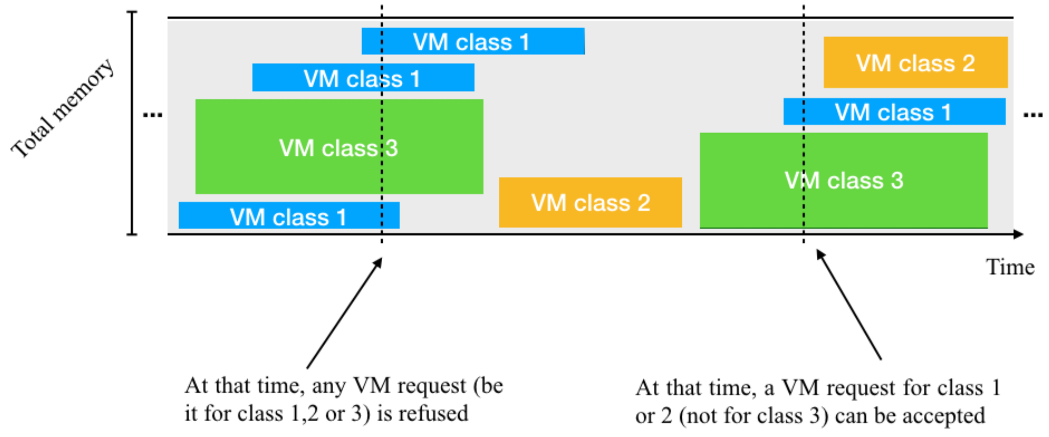

Total memory
VM class 1
VM class 1
VM class 3
VM class 1
VM class 2
VM class 2
VM class 1
VM class 3
Time
At that time, any VM request (be it for class 1,2 or 3) is refused
At that time, a VM request for class 1 or 2 (not for class 3) can be accepted

Fig. 1: Example of the utilization of a PM memory with three classes of VMs.

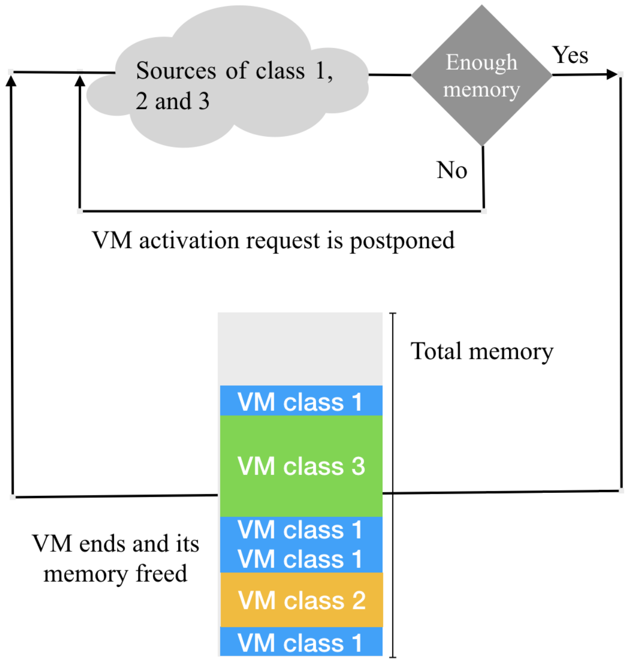


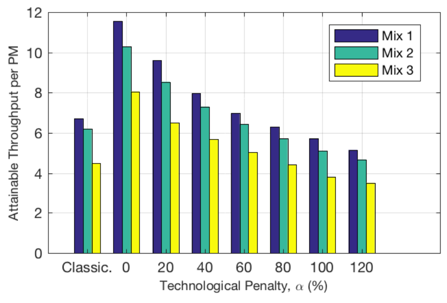
Fig. 2: Memory acquisition model with three classes of VMs.

and mix$_3$, respectively. We chose VM memory requirements and their distribution in mix$_1$ according to [20].

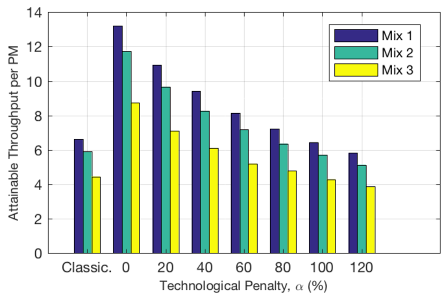
As is the common case [1], we assume that all PMs of the rack have the same amount of memory $B_j = 64$ GB for $j = 1, \ldots, N$. We start by setting the expected lifetime of a VM in absence of disaggregation equal to 1 time unit (e.g., 1 hour) for all VM classes, namely $t_s = t_m = t_l = 1$. Since disaggregation can be expected to carry a technology penalty caused by remote memory accesses, we use our model to study the attainable VM launch throughput with and without disaggregation for different values of the technology penalty, varying from α =0% to α =120%. A penalty of around 40% compared to the classical architecture was suggested in [12]. As mentioned before, attainable VM launch throughput refers to the expected maximum number of VM launches per time unit where for VMs of all classes at least 99% of launch requests find the memory they need. Although the choice of 99% is arbitrary, studies with other values show similar qualitative behavior.

Figures 3a, 3b and 3c show the attainable VM launch throughput per PM for N = 2, 4 and 8 PMs, respectively, as a function of the technology penalty. In each figure we have also included the attainable throughput without disaggregation, labeled "classic". We observe that the attainable throughput can be significantly improved through disaggregation (upwards of 100%, depending on the mix). For a given number of PMs combined through disaggregation, not unexpectedly, the amount of improvement depends on the technology penalty. If the latter is high enough, the disaggregated architecture may actually perform less well than the classical architecture. The precise value of the technology penalty for which disaggregation stops outperforming the classical architecture may vary with the VM mix and the number of PMs disaggregated but, for the parameter values considered in our study, appears to be around a technology penalty of 100-120%. This is illustrated in Figure 5. Intuitively, in general, one can expect the technology penalty to increase with the number of PMs combined through disaggregation due to a larger number of remote resources shared by a larger number of VMs..

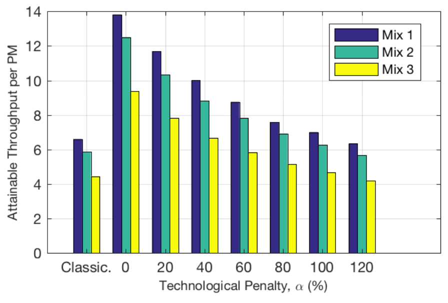
In Figure 6a we show the percent improvement in attainable throughput for a technology penalty of 40% as a function of the number of PMs combined through disaggregation. We can observe a clear pattern of "diminishing returns" at around N=6 PMs. Indeed, doubling the number of PMs from 2 to 4 results in over twofold increase in the percentage improvement for the attainable gain, while doubling again from 4 to 8 PMs leads to a much smaller increase. This trend seems to intensify after 8 PMs. The percent improvement in attainable VM launch throughput for other values of the technology penalty (for



(a) With 2 PMs.



(b) With 4 PMs.



(c) With 8 PMs.

Fig. 3: Attainable VM launch throughput per PM.

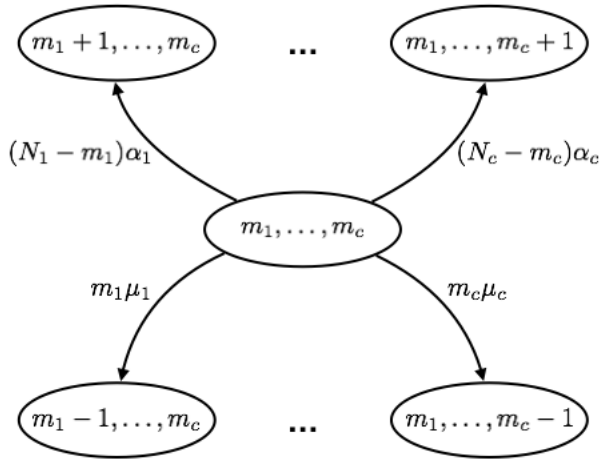


Fig. 4: Outgoing transitions for state $(m_1, \ldots, m_c)$ in the Markov chain.

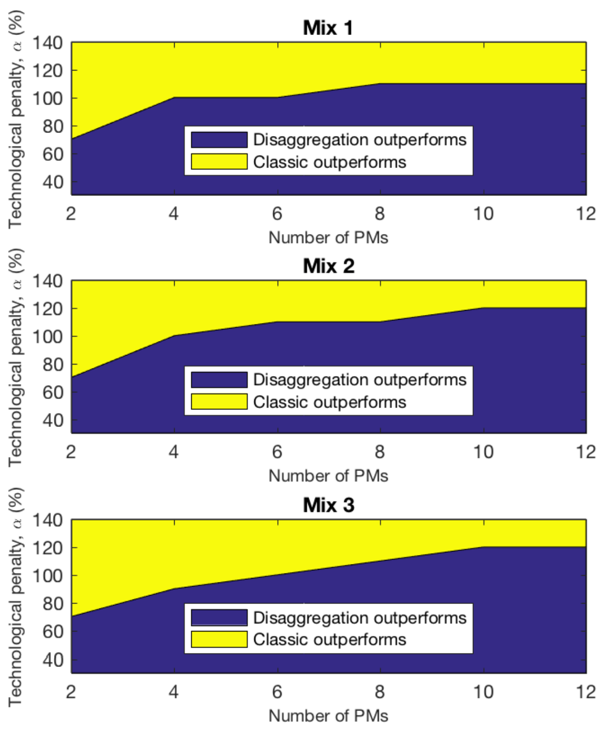


Fig. 5: Technology penalty for which disaggregation stops outperforming classical architecture.

which there is an improvement in the attainable throughput) appears to follow a similar pattern.

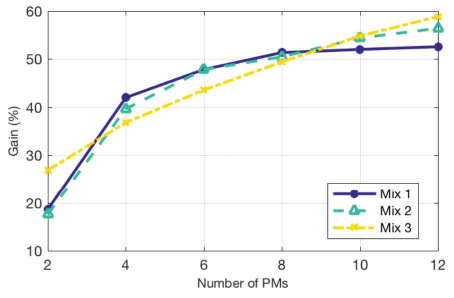
It is intuitively clear that the improvement in attainable launch throughput should tend to increase as the memory size of the VMs increases relative to the memory size of a single PM. In Figure 6b we show the results obtained for a memory size mix similar to our mix$_1$ but with 3 larger sizes for the third class of VMs: b_l=12, 16 and 20 GB, respectively. We observe that the percentage improvement in the attainable throughput can exceed 200% when b_l=20 GB. Interestingly, the diminishing returns appear to follow the same pattern as for our original value of b_l=8 GB.

So far we have assumed that the expected lifetime duration was the same for VMs of all classes. Figure 6c illustrates the improvement in attainable VM launch throughput for VMs whose memory requirement are those of our “mixes” considered in Figure 6a with different expected VM lifetimes. Specifically, we assume that t_s=0.5, t_m=1 and t_l=2 time units. In other words, small VMs have shorter lifetimes while large VMs also persist for the longest time. We observe a behavior similar to that seen in Figure 6a.

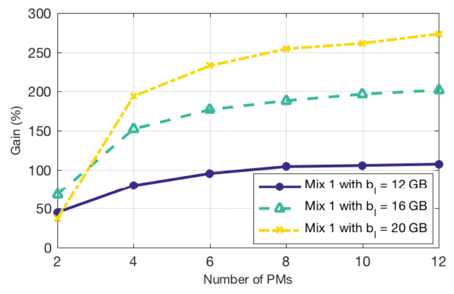
Overall, these results seem to suggest that the “good” number of PMs to disaggregate is between 4 and 8, and disaggregating a larger number might not be warranted, especially if the technology penalty increases with the number of PMs combined through disaggregation.

IV. Conclusions

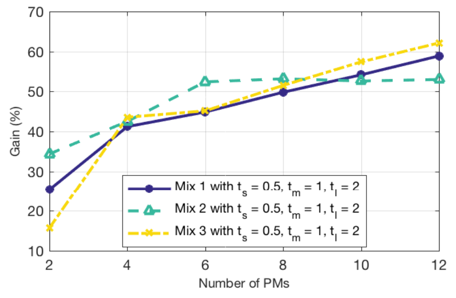
Based on the observation that memory tends to be the limiting resource in current cloud systems, we present a high-level model of VM launch / memory allocation in disaggregated architecture. The model possesses a simple analytical solution.



(a) VM configuration inspired by [20].



(b) VMs of larger memory sizes.



(c) VMs of heterogeneous lifetimes.

Fig. 6: Increase (in %) in attainable PM launch throughput for a technology penalty of 40% as suggested in [12].

We use our model to study the potential improvement in attainable VM launch throughput that could be expected from disaggregation. We present the results obtained for several mixes of VMs with different memory size requirements.

Our results indicate that disaggregation has the potential to significantly improve the attainable VM launch throughput. The caveat is that the technology penalty, caused by the increase of the average memory access time, must remain moderate (say, no more than about 50%). Clearly, this elongation in average resource access time will translate in at least a commensurate elongation in the expected end-user task execution time. The percentage improvement in the attainable VM launch throughput per PM exhibits a clear pattern of diminishing returns so that, for the parameter values considered in our study, disaggregation of some 4 to 8 PMs seems the right option. Our results are a potentially important element of guidance for system architects and cloud operators in the design and deployment of rack disaggregation. In our future work, we plan to focus on experimental validation of our model , which may lead to incorporating other PM resources in our model.

References

[1] O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich, and T. Moscibroda, "Protean: VM allocation service at scale," in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, Nov. 2020, pp. 845–861. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/hadary

[2] P. X. Gao *et al.*, "Network Requirements for Resource Disaggregation," in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[3] V. Nitu *et al.*, "Welcome to Zombieland: Practical and Energy-Efficient Memory Disaggregation in a Datacenter," in *Proc. of ACM EuroSys*, 2018.

[4] E. Amaro *et al.*, "Can Far Memory Improve Job Throughput?" in *Proc. of ACM EuroSys*, 2020.

[5] J. Gu *et al.*, "Efficient Memory Disaggregation with INFINISWAP," in *Proc. of USENIX NSDI*, 2017.

[6] Z. Ruan *et al.*, "AIFM: High-Performance, Application-Integrated Far Memory," in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

[7] J. Zhang *et al.*, "GiantVM: A Type-II Hypervisor Implementing Many-to-One Virtualization," in *Proc. of ACM VEE*, 2020.

[8] Y. Shan *et al.*, "LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation," in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[9] K. Lim *et al.*, "Disaggregated Memory for Expansion and Sharing in Blade Servers," in *Proc. of ISCA*, 2009.

[10] M. Amaral, J. Polo, D. Carrera, N. Gonzalez, C.-C. Yang, A. Morari, B. DAmora, A. Youssef, and M. Steinder, "Drmaestro: orchestrating disaggregated resources on virtualized data-centers," *Journal of cloud computing*, vol. 10, no. 1, pp. 1–20, 2021.

[11] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, "Network support for resource disaggregation in next-generation datacenters," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013, pp. 1–7.

[12] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in *Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 249–264.

[13] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," *ACM SIGARCH computer architecture news*, vol. 37, no. 3, pp. 267–278, 2009.

[14] A. Brandwajn and A. K. Sahai, "Aspects of the solution of some multiclass loss systems," *Performance Evaluation*, vol. 17, no. 2, pp. 141–154, 1993.

[15] "CloudSim," http://www.cloudbus.org/cloudsim/, 2021.

[16] "SimGrid," 2021. [Online]. Available: https://simgrid.org/

[17] "DiME," https://github.com/networkedsystemsIITB/DiME/blob/master/README.md, 2021.

[18] J. Kaufman, "Blocking in a shared resource environment," *IEEE Transactions on communications*, vol. 29, no. 10, pp. 1474–1481, 1981.

[19] "Matlab implementation ," https://github.com/memory-disaggregation/model, 2021.

[20] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2017, pp. 153–167.