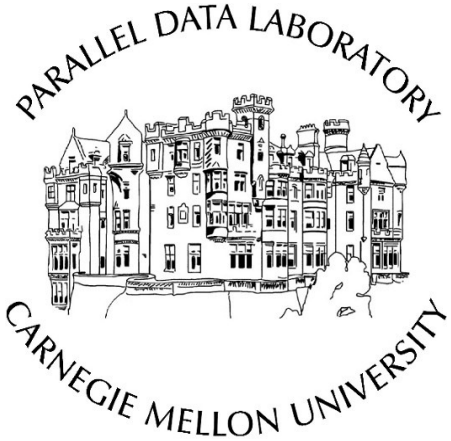


# Mimir: Finding Cost-efficient Storage Configurations in the Public Cloud

Hojin Park, Gregory R. Ganger, George Amvrosiadis
Carnegie Mellon University

CMU-PDL-22-102
Feb 2022

**Parallel Data Laboratory**
Carnegie Mellon University
Pittsburgh, PA 15213-3890

## Abstract

*Public cloud providers offer a diverse collection of block storage options with different costs and performance SLAs. As a consequence, it is difficult to select the right allocations for storage backends when moving data-heavy applications to the cloud. Mimir is a tool for automatically finding a cost-efficient virtual storage cluster (VSC) configuration for a customer's storage workload and performance requirements. Importantly, since no single allocation type is best for all workloads, Mimir considers all allocation types and even heterogeneous mixes of them. In our experiments, compared to state-of-the-art approaches that consider only one allocation type, Mimir finds VSC configurations that reduce cost by up to 81% for substantial storage workloads.*

**Acknowledgements:** We thank the members and companies of the PDL Consortium—Alibaba, Amazon, Datrium, Facebook, Google, Hewlett Packard Enterprise, Hitachi, IBM Research, Intel, Micron, Microsoft Research, NetApp, Oracle, Salesforce, Samsung, Seagate, and Two Sigma. Hojin Park is supported in part by Korea Foundation for Advanced Studies.

**Keywords:** Public cloud, Storage system, Automated resource provisioning

# Mimir: Finding Cost-efficient Storage Configurations in the Public Cloud

Hojin Park, Gregory R. Ganger, George Amvrosiadis

## 1 Introduction

Companies are increasingly moving data-heavy applications to the cloud, often replicating on-prem implementations of integrated data processing and storage backend systems on cloud instances. While researchers have introduced and studied effective approaches for auto-selecting cost-optimized VM instances for computation work [2, 10, 13, 35, 37], less attention has been paid to storage selection. For cold storage, there is usually a clear option (e.g., S3 in AWS or Blob Storage in Azure). For performant storage needs, however, the option-set is increasingly diverse in volume types, SLAs, and cost structures. Selecting the most cost-effective virtual storage cluster (VSC) configuration for a given data-heavy application deployment is likely beyond all but the most expert user.

Commonly, storage backends (e.g., distributed file systems or key-value stores) are built for use with block storage volumes providing traditional SSD or HDD interfaces. Selecting storage hardware for on-prem deployments is challenging [3, 4, 38], given the many options. The challenge in cloud deployments is similarly difficult, but differently so because of cloud SLA and cost structures. Using AWS as a concrete example, there are three types of block storage volume: local-SSD associated with a compute instance, remote-SSD that can be attached to any VM instance, and remote-HDD that can be attached to any compute instance. Making matters worse, each type has multiple allocation options with different costs and different SLA structures regarding cost as a function of performance and capacity required. For example, some charge per-GB with a number of IOPS per-GB, while others provide a specific capacity and performance for a given cost, and still others a given MiB/s per-TB-rented. Different customers will be best served by different choices, and the best choice can be a mix of multiple types.

Figure 1 illustrates the need to consider the many types and allocation options in selecting a VSC configuration. For each of three workloads on a distributed storage backend, it shows the cost for the best VSC configuration choice under each of three constraints: only considering local-SSD allocations, only considering remote storage (EBS) allocations (e.g., as is done by a recent auto-selector called Optimus-Cloud [29]), and considering arbitrary mixes of both storage types. Two crucial takeaways are visible: (1) the best single-type choice differs for different workloads, and (2) a mixture of allocation types is sometimes required to minimize cost.

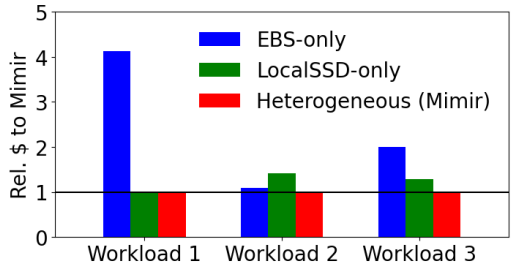


**Figure 1.** No single storage type is most cost-efficient for every workload, and a mix of storage types is sometimes best. The experimental setup and workloads (FR-A, FR-D, and SYN) are described fully in Sec4.1.

This paper presents **Mimir**, a tool for finding a cost-effective set of instance and volume allocations for a distributed storage backend used by a data-heavy application workload. Given high-level workload specifications and performance requirements, as might be produced by profiling an operational version of the system (whether on-prem or using an over-provisioned pre-Mimir configuration), Mimir considers potentially heterogeneous VSC configurations to find (for example) those shown as "Heterogeneous" in Figure 1.

Mimir casts VSC configuration selection as an optimization problem, like most prior tools for automated resource selection. Central to how Mimir achieves its goals is predicting the resources required for the given workload, including both the I/O throughput of the access pattern and the CPU+memory needs of the storage software. Predicting the resources required, rather than a workload's performance for any given instance type like most previous works [2, 24, 29], Mimir simplifies both predictions and exploration of heterogeneous VSC configuration options. It also allows Mimir to find good VSC configurations for workload mixes composed of multiple access patterns. With predicted resource requirements and analytically-formulated price-performance cost models of public cloud resources, Mimir determines cost-efficient VSC configurations using dynamic programming.

We evaluate Mimir using distributed storage backends and workloads inspired by discussions with engineers of a top customer relationship management (CRM) service. Our results show significant cost savings arising from Mimir's approach and its ability to consider diverse storage types. For example, compared to a state-of-the-art approach considering only EBS allocations, Mimir reduces cost by up to 81%. More generally, Mimir consistently and quickly finds cost-effective VSC configurations even when accounting for imperfect profiles and computation needs of the distributed storage software.

**Contributions.** This paper makes three primary contributions. First, it shows that finding cost-optimal cloud storage VSC configurations requires consideration of diverse allocation types. Second, it describes the architecture and algorithms that allow Mimir to find cost-effective VSC configurations for a distributed storage backend. Third, it demonstrates that Mimir can effectively explore AWS's diverse block storage offerings, reducing cost by up to 81% relative to state-of-the-art approaches. Mimir will be shared as open-source software.

## 2 Cloud Storage Configuration Challenges

This section motivates the need for tools like Mimir that automate the way virtual storage clusters are configured in the public cloud. Specifically, we examine two aspects of public clouds. First, we examine the diversity in performance and cost characteristics of different cloud storage types, which complicate the manual configuration process (§2.1). Second, we examine how the diverse characteristics of cloud storage types affect the overall cost of deploying Apache BookKeeper, a scalable storage service, in the public cloud (§2.2). Last, we survey related work and identify the difference between Mimir with previous works (§2.3).

### 2.1 Storage Characteristics in the Public Cloud

It is crucial to understand the characteristics of the public cloud storage types in order to configure storage systems atop virtual storage cluster in a cost-efficient manner. Mimir formulates the price and performance cost model with the analyzed storage characteristics in this section.

One of the public cloud storage types we use to build volumes for the distributed storage systems in this paper is *block storage*, such as AWS Elastic Block Store [7], Azure Disk Storage [8], and GCE Persistent Disk [18]. On AWS, there are five different types of block storage: local NVMe SSD, remote SSD (gp2, io1), and remote HDD (st1, sc1).

Local SSD is served as an SSD locally attached to some instance types, such as i3, c5d, and m5d. It delivers very high performance with low latency, but the attached volume capacity is fixed, and it can be an expensive option for the data that does not require high throughput. Unlike local SSD, users can attach any remote storage volumes (EBS) to the machines they need. The performance of remote storage types is defined as SLAs by the public cloud providers. For instance, AWS currently offers gp2 volumes at 3 IOPS per GiB of provisioned capacity, while it provides 40 MiB/s per TiB of provisioned capacity for st1 volumes.

Figures 2 and 3 illustrate the characteristics of 1 TiB of gp2 volume and 1 TiB of st1 volume, in which the performance of each volume is 3000 IOPS and 40 MiB/s, respectively, and local SSD attached at i3.xlarge. We generated the test workloads with the `fio` benchmark [16] varying the access pattern (random/sequential), read ratio, and I/O unit size.

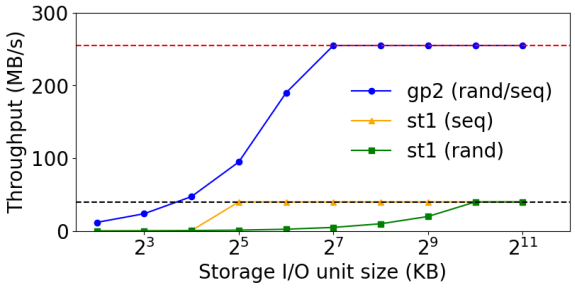


**Figure 2.** Performance characteristics of public cloud storage types by I/O unit size. Both storage types have throughput limits defined by AWS (horizontal lines).

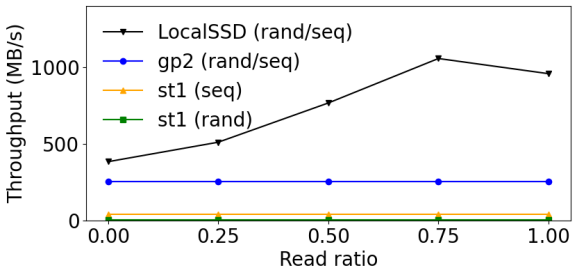


**Figure 3.** Performance characteristics of public cloud storage types by workload read ratio. EBS volumes are not affected by the read ratio, while local SSDs are.

The performance of gp2 is defined in IOPS. As Figure 2 shows, as the I/O unit size increases, the throughput of gp2 also increases up to 250 MiB/s, which is the maximum single gp2 volume throughput limited by SLA. In the case of st1, performance is defined in MiB/s, but shows lower throughput for the workloads with random access patterns and I/O units less than 1 MiB [22]. st1 has a throughput limit at 40 MiB/s for 1 TiB st1 volume, in which the limit can be up to 500 MiB/s for the larger st1 volume. The performance of gp2 is the same for both random and sequential data access patterns, while st1 shows better performance for sequential data access than random access.

In Figure 3, the throughput of EBS volumes is not affected by the read ratio of the storage workload as the read ratio is not included in their performance SLAs. The local SSD, however, shows much higher throughput than EBS, and we observed that the throughput is not affected by the I/O unit size for requests larger than 32KB.

We have measured the local SSD performance of all the machines we used as candidates of the cost-optimal VSC. Profiling is not a consuming process in terms of time or cost because there are not many machines to profile (28 instance types in our experiment), and profiling need only be performed once. There are other volume types (`io1`, `sc1`) we also considered, but we omit them for brevity.

### 2.2 Use Case: Apache BookKeeper Deployment Cost

Next, we give a motivating example demonstrating the potential savings of careful machine configuration for an application. Inspired by discussions with engineers from a large customer rights management (CRM) company shifting from on-prem to cloud, we look at Apache BookKeeper. Apache BookKeeper [23] is a storage system designed for high scalability, fault-tolerance, and low-latency. It stores data as streams of log entries in sequences called ledgers, and the ledgers become immutable once the ledger is closed. The primary data access pattern of the storage server (Bookie server) is sequential writes and random reads. The concept of ledgers and entries of Apache BookKeeper is similar to that of the SSTables used in many storage systems, including RocksDB [1].

Now we give a motivating example of reducing cost by exploiting heterogeneous resource allocations. Figure 4 shows the resource utilization of the Bookie server running on i3.2xlarge, with a 1.9 TiB local SSD. The workload is write-only and requires 1.8 TiB of data capacity and 360 MiB/s of write throughput. After 40 seconds, we add a 600GiB EBS volume and increase both requirements by 30% so the workload's required throughput per TiB remains the same.

For the first 40 seconds, 67% of CPU is idle on average while the storage bandwidth of the local SSD is fully utilized. Provisioning another instance with a local SSD would double costs. Attaching an EBS volume to the original instance, however, allows us to store 30% more data while paying only 12% additional cost, which reduces the cost per data size by 15%, and this heterogeneous allocation allows the workload to utilize 15% more idle computing power.

Therefore, it is crucial to accurately predict how much resources (e.g., CPU, memory, storage bandwidth, etc.) are required for the given workload characteristics to configure cost-efficient heterogeneous virtual storage clusters (§3.3). Also, though we restrict to a single instance type and one workload characteristic in this example, if we consider more instance types and workload characteristics together, the gain from the heterogeneity compared to the homogeneous allocation increases (§4.6).

### 2.3 Related Work

**Configuring storage and VMs in public cloud.** Many previous works [10, 27, 34, 40, 42] aim to optimize virtual cluster configuration in public clouds for various workloads. Previous works [2, 24, 25] find near-optimal cloud storage and VM configuration for data analytics workloads, guaranteeing the performance and minimizing the cost. However, the data service workloads we target have different nature from the data analytics workloads, e.g., data service workloads are long-running, rather than transient, and cannot classify data into input/output and intermediate data, which are common in data analytics applications.

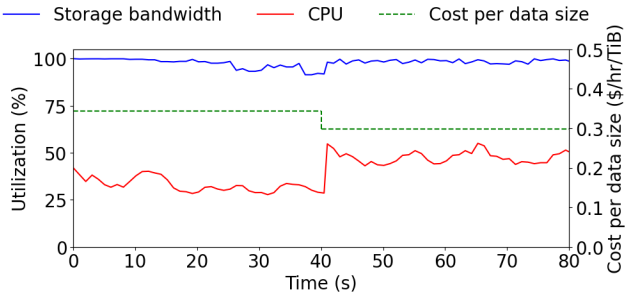


**Figure 4.** Reducing the cost per data size by exploiting heterogeneous machine allocation. When Bookie server uses only local SSD, CPU is underutilized. By attaching an EBS volume it can store 30% more data paying only 12% additional cost, i.e., reduce the cost per data size by 15%.

OptimusCloud [29] jointly optimizes database and VM configurations to find cost-efficient VSC configurations for distributed databases. We consider OptimusCloud as the state-of-the-art to compare with Mimir, but OptimusCloud seeks a cost-efficient configuration while considering only the EBS storage type, which we show could be a costly configuration compared to a VSC using both local SSD and EBS volume types (§4.6).

**Performance prediction on VMs.** Numerous previous systems [9, 12, 14, 15, 30–33] studied the method of predicting workload performance on VMs. PARIS [41] uses hybrid offline/online data collection and trained a random forest model using the collected data to predict the workload performance on VMs. Ernest [39] predicts the performance of large-scale data analytics workloads using profiled data and statistical performance modeling. Auto-configuration systems [24, 29] also predict the workloads' performance on VMs using machine learning techniques, such as collaborative filtering and gradient boosting tree.

In contrast, our approach predicts resources required for the given workload performance instead of predicting workload performance on VMs. Still, we can use similar data profiling techniques and prediction approaches that previous works proposed, such as gradient boosting tree.

## 3 Mimir Design

Figure 5 shows the overall workflow of Mimir. First, Mimir takes as input information about multiple user-defined workloads (§3.1). Each workload can have a different data access pattern, such as the data request rate, data access locality, and read and write request ratio. Then, our *Resource profiler* profiles each workload and collects data of how many resources are required (ContainerSpec) to cost-efficiently run each workload on the machines in the cloud (§3.2). Using the collected data, the *Resource predictor* learns how to convert each workload specification into the right size of the container to run (§3.3). In Mimir, each storage server runs on a docker container for resource isolation because multiple

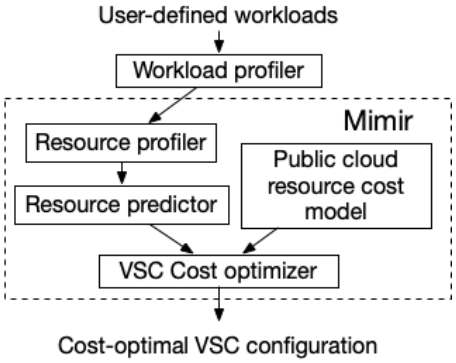


**Figure 5.** Mimir's workflow for optimizing the price of public cloud resources. Mimir profiles user-specified workloads and learn how much resources (e.g., CPU, memory) are required for each of the workloads. The VSC Cost optimizer uses this trained module and cost model of public cloud resources to find the cost-efficient VSC configuration in the public cloud.

storage servers running on a single machine must be guaranteed the allocated resources. Lastly, the *VSC Cost optimizer* uses the *Resource predictor* and the cost model of the public cloud resources to find the cost-efficient virtual storage cluster configuration of the distributed storage system (§3.4).

### 3.1 User-defined workloads

Mimir takes user-defined workload specifications as an input. Table 1 shows the five attributes we use to describe workload characteristics in Mimir. They are divided into two categories: performance requirements and data access pattern.

Data capacity and data request rate are the attributes of the performance requirements that should be satisfied for the given workloads. Performance requirements are also used as profiling knobs and they are proportional to the size of *workload fraction*. For example, if a user defines a workload with 1 TiB of data capacity and 10K QPS (i.e., queries per second) of data request rate, we expect 3K QPS is required for the 300 GiB of the given workload's data.

The attributes of the data access pattern describe the behavior of the workloads: temporal/spatial data access locality, percentage of the read operations, and distribution of data size stored in the storage backend. Unlike the performance requirements, Mimir expects the attributes of the data access pattern to remain the same for any *workload fraction* and uses this assumption in the *Resource profiler* to generate a set of *workload fractions* to profile. As a future work, Mimir will monitor the actual characteristics of the *workload fraction* on runtime and give feedback to these assumptions to reconfigure the VSC configuration.

Many previous works [29, 36] have supported elastic right-sizing cloud resources at runtime by predicting future workload characteristics or in a reactive manner. But elasticity is orthogonal to our work. Instead, we focus on finding the potentially heterogeneous cost-efficient VSC configuration for the mixture of static workloads with different characteristics.

### 3.2 Resource profiler

Mimir should know the cost-efficient container size to allocate for the storage servers, i.e., Mimir should be able to allocate enough resources to satisfy the performance requirements of the user-defined workloads while not provisioning more resources than the workload needs. However, many factors make it challenging to compute the right size of *ContainerSpec* for the arbitrary workload specification analytically. Read/write amplification inherent in the storage servers depends on the implementation and data access pattern; memory size of the storage servers and read/write ratio of workloads affect the necessary storage throughput and computing power for the container to meet the performance requirements. None of these factors can be precisely formulated without the storage system experts and should be reformulated for every storage system to be used. Instead of formulating the right size of *ContainerSpec*, Mimir, therefore, 1) profiles and collects data using the *Resource profiler* and 2) predicts the cost-efficient size of *ContainerSpec* using the *Resource predictor* trained with the collected data.

The *Resource profiler* runs the workloads defined by users on a benchmark machine to collect the data of the cost-efficient size of containers for the given workload specifications. When profiling, Mimir uses the performance requirement attributes as knobs to get multiple data points. The attributes of the *ContainerSpec* we use are: the number of CPUs, memory size, storage bandwidth, storage capacity, and network bandwidth. To get enough profiling data, we chose i3.4xlarge of AWS as the benchmark machine, which has the local SSD with the highest single-storage performance (1900GB NVMe SSD), and sufficient memory and computing resources in profiling our evaluation workloads.

Note that there are multiple suitable *ContainerSpecs* for a single workload specification. For example, a read-intensive workload with a high degree of data access locality requires less storage volume performance and computing power with larger memory size because of memory caching. Figure 6 shows how different the required resources are according to the memory size, even for the same workload specification. So the *Resource profiler* profiles the workload with different memory sizes to take into account multiple *ContainerSpecs* in the optimization algorithm.

Algorithm 1 shows the *Resource profiler*'s logic to collect the data of the right size of *ContainerSpecs* for the multiple *workload fractions* with different performance requirements. The *Resource profiler* first measures the maximum performance $p$ of the workload on the benchmark machine with the given data access pattern (Line 4 of Algorithm 1). Then, it generates a set of $N$ different *workload fractions* (i.e., workloads with $1/N*p$, $2/N*p$, ..., $N/N*p$ performance requirements)

| Type | Attribute | Description | Units |
|---|---|---|---|
| Performance requirement (Profiling knobs) | data capacity | total size of data stored in the storage system | GB |
| | data request rate | rate of read and write requests arrive at the storage system | QPS |
| Data access pattern | data request size | mean or distribution of the requested data size | Byte |
| | read/write ratio | ratio of the read and write request rates | |
| | access locality | pattern of data access locality | |

**Table 1.** The workload characteristic attributes. The performance requirement attributes are the knobs used by Mimir in profiling to get multiple profile data points, while the data access pattern attributes are not changed.


**Figure 6.** The cost-efficient container sizes for the same workload with different memory sizes. The larger the memory size of the container running the storage server, the less other resources such as storage bandwidth and computing power are required.

**Algorithm 1** Profiling logic of the *Resource profiler*

```
1:  W: User-defined workload specification
2:  BM: Benchmark machine
3:  procedure PROFILE(W)
4:      p ← MEASUREMAXPERF(W, BM)
5:      S ← WORKLOADFRACTIONSETTOPROFILE(W, p)
6:      D ← {}
7:      for wf in S do
8:          u ← MEASURERESOURCEUTILIZATION(wf, BM)
9:          while ¬ ISCONTAINERRIGHTSIZE(wf, u) do
10:             u ← UPDATECONTAINERSPEC(wf, u)
11:         end while
12:         D[wf] ← u
13:     end for
14:     return D
15: end procedure
```

to profile on the benchmark machine (Line 5 of Algorithm 1). We used $N = 10$ in our experiments. For each *workload fraction* in the set, the *Resource profiler* finds the right container size (Line 7-13 of Algorithm 1). It first measures the average resource utilization while running the *workload fraction* on the benchmark machine. However, the container allocated with the average resource utilization may not meet the performance requirements, or it may have been allocated more resources than necessary. So, the *Resource profiler* repeatedly updates the candidate container size by measuring the storage server performance and resource utilization in the running container until it finds the cost-efficient size of the container. As simple rules of updating the container size iteratively (Line 10 of Algorithm 1), if the current container size satisfies the workload requirements and the average utilization values of some resources are less than the threshold (we used 80%), it is considered those resources are allocated more than necessary. So the profiler reduces their resource allocations. If it does not satisfy the workload requirements, Mimir increases the allocation of the resources with the average utilization higher than the threshold (we used 90%), judging them as bottleneck resources.

### 3.3 Resource predictor

Based on the data profiled by the *Resource profiler*, Mimir predicts the cost-efficient size of containers for the given workload characteristic. Currently, Mimir provides an implementation using interpolation, but other prediction models, such as a gradient boosting tree [17], could be used instead.

The *Resource profiler* profiled $N$ different *workload fractions*, in which each requires the performance of the $i/N \times p$ (where $i = 1 \cdots N$ and $p$ is the maximum performance on the benchmark machine). Thus, the *ContainerSpecs* for the workload requiring performance less than $p$ can be computed using interpolation. As we noted, we use the large enough instance types as a benchmark machine (i.e., the machine that can profile up to large $p$) in order to use the interpolation for the *workload fractions* that require high performance.

The interpolation approach allows accurate prediction of the right size of the *ContainerSpecs* as the *Resource profiler* profiles enough data. However, this approach requires a full profiling process when the new workload comes in, which requires additional profiling time and cost.

### 3.4 VSC Cost optimizer

Mimir uses a nested optimization loop to minimize the cost of the virtual storage cluster while satisfying the performance requirements. Figure 7 shows an example of how the outer loop (OPTCLUSTER) and the inner loop (OPTSINGLEMACHINE) works. First, the outer loop uses dynamic programming to break the problem of finding the cost-efficient VSC configuration that can run the entire workload into the smaller

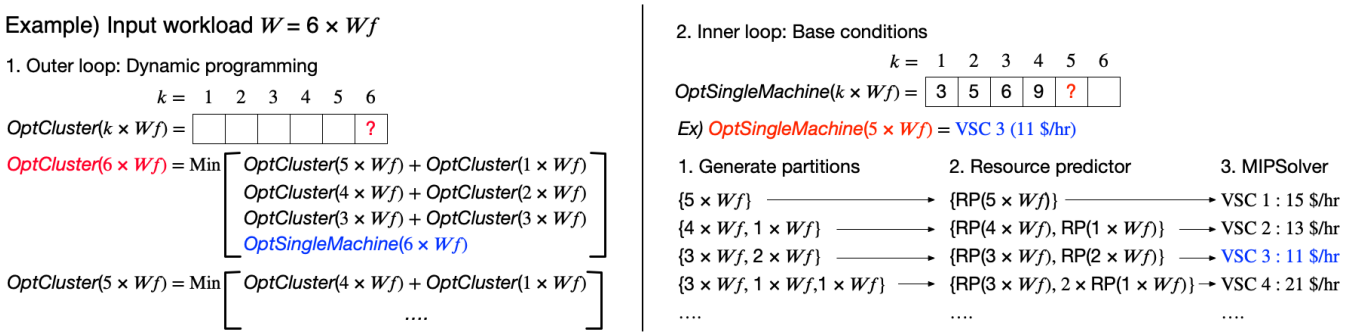


**Figure 7.** Example of the *VSC Cost optimizer*'s optimization algorithm. The outer loop uses dynamic programming to break the optimization problem into smaller problems, and the inner loop finds the answers for the base conditions.

**Algorithm 2** Optim. algorithm of the *VSC Cost optimizer*

```
1:  W: User-defined workload specification
2:  procedure OptCluster(W)
3:      S ← WorkloadFractionPairs(W)
4:      c ← ∞
5:      for Pair<W1, W2> in S do
6:          t ← OptCluster(W1) + OptCluster(W2)
7:          c ← min(t, c)
8:      end for
9:      t ← OptSingleMachine(W)
10:     c ← min(t, c)
11:     return c
12: end procedure
13:
14: procedure OptSingleMachine(W)
15:     D ← WorkloadFractionPartitions(W)
16:     c ← ∞
17:     for d in D do
18:         S ← ResourcePredictor(d)
19:         c ← min(MIPSolver(S), c)
20:     end for
21:     return c
22: end procedure
```

problems of finding the cost-efficient VSC configuration that can run the *workload fractions*. In order to obtain the values of the base conditions of the dynamic programming problem, the inner loop searches for the cost-efficient resource configuration of a single machine that can execute each *workload fraction*.

Algorithm 2 shows the pseudocode of the nested optimization loop. We will first explain the algorithm assuming that there is only a single workload $W$ as an input and then expand to the case where multiple workloads are given as an input.

**Outer loop: OptCluster.** The *VSC Cost optimizer* uses dynamic programming because our optimization problem has optimal substructure property and overlapping subproblems. Mimir first defines the *workload fraction unit* ($W_f$) of the given workload $W$, the smallest unit of the workload data stored in the same storage volume. The size of $W_f$ provides the trade-off between the search space size and the optimality of the solution. We empirically evaluated the trade-off and found that using the data size between 50-100 GiB for $W_f$ is generally good in our experiments, e.g., Mimir uses 100 GiB of data size and 1K QPS as $W_f$ for the workload requires 3 TiB of storage capacity and 30K QPS. Automatically adjusting the size of $W_f$ that considers both optimization time and solution optimality is an interesting research problem, and we left it as our future work.

If the size of $W_f$ is $1/N$ of $W$, the optimization problem of finding the cost-efficient VSC configuration for $W$ is:

$$\textsc{OptCluster}(W) = \textsc{OptCluster}(N \times W_f)$$

We will now explain the optimal substructure property of our optimization problem. Let's assume that we already know the cost-efficient VSC configuration for $W$, $VSC_{opt}$. If $VSC_{opt}$ has more than one machine, any subcluster of $VSC_{opt}$ must be the most cost-efficient VSC configuration for the *workload fraction* that the subcluster is running. Otherwise, we can find another VSC configuration cheaper than $VSC_{opt}$ by replacing the subcluster with the cheaper one, which contradicts our assumption of $VSC_{opt}$.

Using this optimal substructure property, we can use dynamic programming (Line 5-10 of Algorithm 2). We first divide the given workload ($W = N \times W_f$) into two *workload fractions*. That is, there are $\lfloor N/2 \rfloor$ number of different *workload fraction* pairs, $[i \times W_f, (N-i) \times W_f]$. Then the $\textsc{OptCluster}(W)$ is the minimum of $\textsc{OptCluster}$ of $\lfloor N/2 \rfloor$ pairs. As a base condition, all data of $W$ can be stored in a single machine:

$$\textsc{OptCluster}(N \times W_f) = \min\Big(\{\textsc{OptCluster}(i \times W_f) + \textsc{OptCluster}((N-i) \times W_f)\}_{i=1}^{i=\lfloor N/2 \rfloor}, \textsc{OptSingleMachine}(N \times W_f)\Big)$$

$\textsc{OptCluster}$ function can be recursively called, until input of $\textsc{OptCluster}$ becomes $W_f$.

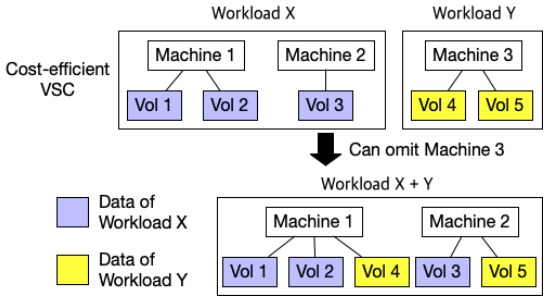


**Figure 8.** Example of cost benefits when optimizing multiple workloads together. In this example, Mimir can save the cost of Machine 3 by attaching Workload Y's storage volumes to the Workload X's virtual storage cluster.

Mimir can efficiently compute the dynamic programming using the memoization if the *VSC Cost optimizer* has all the results of the base conditions (i.e., OptSingleMachine) in advance.

**Inner loop: OptSingleMachine.** To compute the base conditions of dynamic programming, *OptSingleMachine* finds the cost-efficient VSC configuration of a single machine for the given *workload fraction* ($k \times W_f$). Within a single machine, there are Partition($k$) number of different ways to distribute the data into the storage volumes, where Partition($n$) equals the number of possible partitions of $n$. Then for each partition, Mimir generates a set of *ContainerSpecs* by predicting them for each *workload fraction* in the partition using the *Resource predictor*. As each set has the resource requirements of the *workload fractions*, Mimir uses mixed-integer programming (MIPSolver) to minimize the price of a single machine under the resource and the performance requirement constraints:

$$\underset{Machine,Storage}{\text{minimize}} \quad \text{Machine[Price]} + \sum_i \text{Storage[i][Price]}$$

$$\text{subject to} \sum_{CS} \text{CS[CPU,Mem,...]} \leq \text{Machine[CPU,Mem,...]}$$

$$\sum_{CS \in Storage[i]} \text{CS[Storage BW]} \leq \text{Storage[i][BW]}$$

Lastly, OptSingleMachine selects the partition with the smallest return value of the MIPSolver as the cost-efficient configuration of a single machine.

**Multiple workloads as an input.** When the input has more than one user-defined workload, running the optimization algorithm using all the workloads as an input at once can find more (or at least the same) cost-efficient results than using separate virtual storage clusters together after finding the cost-efficient VSC configuration for each (Figure 8). The same nested optimization algorithm can be used for multiple workloads. However, as the number of workloads increases, the complexity of the search space becomes infeasible.

The time complexity of the optimization outer loop for the set of workloads $\{W_i\}$, where each workload $W_i$ can be divided into $N_i \times W_{f_i}$, is proportional to the multiplication of $N_i$.

$$\text{TC of the outer loop} \propto \prod_i N_i$$

The time complexity of the inner loop is proportional to the multiplication of two values: the number of possible partitions of $k$, which is proportional to the exponential function of the square root of $k$ [5], and the optimization time of the MIPSolver.

$$\text{TC of the inner loop} \propto a^{\sqrt{k}} \times T_{\text{MIPSolver}} \text{ (a>1)}$$

Since the total time complexity is the product of these two time complexities, it increases exponentially with the number of workloads considered. To make it complexity feasible, we use *systematic sampling* and *pairwise workload optimization*.

In the inner loop, instead of computing MIPSolver for all the possible partitions, Mimir samples some of the partitions and find the minimum among them. The reason why we used systematic sampling rather than random sampling is that the order we generate partitions has a property that the adjacent partitions tend to have similar configurations. So by selecting every $n$th partition allows the Mimir to explore various configurations.

As the search space complexity increases exponentially to the number of workloads, Mimir runs the optimization algorithm for up to two workloads at once for all the pairwise workload combinations. For example, if there are six different workloads as an input, rather than give six of them at once to the optimization function, run the pairwise workload optimization $\binom{6}{2}$ times and find the total cost-efficient VSC configuration using them.

Both approaches provide the trade-off between the optimization execution time and the solution's optimality. We could not directly evaluate the trade-off because the search space is infeasible without these approaches. But, we show Mimir can find cheaper VSC configuration using these approaches when multiple workloads are considered as an optimization input at once (§4.6).

## 4 Evaluation

We evaluate Mimir using a synthetic benchmark and Facebook's RocksDB key-value workloads [11]. We first describe our experimental setup (§4.1) and baselines that have different resource constraints to be used in sensitivity evaluation (§4.2). Then we evaluate Mimir with experiments to answer the following questions:

- Given a set of workload characteristics, can Mimir use profiling data to determine a cost-efficient container size? (§4.3)
- Given a set of workload characteristics and insufficient profiling data, can Mimir predict a cost-efficient container size accurately? (§4.4)

- How crucial selecting a data distribution for finding the cost-efficient virtual storage cluster (VSC)? (§4.5)
- Can Mimir find a cost-efficient VSC to satisfy the requirements of different workloads? (§4.6)
- Do the configurations proposed by Mimir waste resources, i.e., miss out on potential cost savings? (§4.7)

### 4.1 Experimental setup

**Cluster configuration and software.** We evaluated Mimir using AWS EC2 US-East-1. We used 55 different instance types on AWS for the candidate instance types of the cost-efficient VSC configuration (Table 2). The instance types we experiment with include all types of AWS instances except ones with GPUs, i.e., instances that are general purpose (m5, m5d), compute optimized (c4, c5, c5d), memory optimized (r5, r5d), and storage optimized (i3). For the candidate storage types, we used local SSD (i.e., the high performance SSD already attached at i3, c5d, r5d) and remote EBS volume types (gp2, io1, st1, sc1). Note that any new type of instance or storage type (e.g., gp3, io2) can be easily added to the resource candidate pool. We ran our optimization algorithm on a Xeon E5-2670 2.60GHz CPU with 64 GiB DDR3 RAM, using Gurobi 9.0.1 solver [19]. We used Apache BookKeeper 4.11.0 as the storage backend where upon our key-value workloads were run, as described in §2.2.

| Category | Instance type (# of instance sizes) |
|---|---|
| general purpose | m5 (6), m5d (6) |
| compute optimized | c4 (5), c5 (8), c5d (8) |
| memory optimized | r5 (8), r5d (8) |
| storage optimized | i3 (6) |

**Table 2.** AWS instance types we used as the candidate machine types. We selected at least one instance type from each category.

**Workloads.** We evaluated Mimir using two benchmarks on top of Apache BookKeeper: a read-only synthetic benchmark and a set of workloads similar to the Facebook RocksDB key-value workloads described in the previous work [11]. Detailed information of each benchmark is in Table 3.

Our read-only synthetic benchmark (**SYN**) is comprised of two workloads: high-throughput workload (`SYN-H`) and low-throughput workload (`SYN-L`). For the BookKeeper configuration parameters, we used 64 KB of entry size (i.e., data request size) and 2 MB of ledger size, which are the average value one of the CRM companies uses in their BookKeeper storage cluster. It also represents the key-value workloads that have large value sizes which are common in real-world [6, 21, 26, 28]. For the performance requirements, `SYN-H` and `SYN-L` require 200 MiB/s and 50 MiB/s per TiB of data capacity, respectively, and both have 3 TiB of data. Lastly, they read randomly from data in the storage cluster.

Facebook presented the detailed characteristics of their key-value workloads [11] in their storage cluster which uses RocksDB as their backend storage engine. They described three production use cases, which are UDB, ZippyDB, and UP2X, and we selected UDB to evaluate Mimir. Because UDB has six workloads that have different characteristics to each other, we can evaluate Mimir for the complicated realistic benchmark. To evaluate UDB-like workload on Apache BookKeeper, we implemented our own benchmark (**FR**) on Apache BookKeeper that has similar characteristics as Facebook described. Our benchmark has the same data size (i.e., entry size in BookKeeper) distribution stored in the storage cluster, data access locality and count distribution, and average Put/Get request ratio. We used the same distributions presented by Facebook, which are General Pareto Distribution [20] for value size distribution and a simple power model for access count distribution, to reproduce the similar workload characteristics. However, we implemented only `Put` and `Get` operations, as semantics of other RocksDB deviate significantly from available operations in Apache BookKeeper. Also, Mimir does not support dynamic re-configuration of VSC, so we omitted diurnal patterns of data request rate and used the maximum data request rate of the UDB workloads instead. We described the detailed attributes of six workloads, `FR-A` to `FR-F`, on Table 3 and you can refer the paper [11] to see the detailed information of data size distribution, and data access locality and count distribution.

### 4.2 Baseline

We chose three resource restrictions as baselines to demonstrate the importance of constructing a heterogeneous VSC configuration for cost-efficiency. We found the optimization results under the resource restriction of each baseline using the *VSC Cost optimizer* of Mimir and compared them to the result without any resource constraint.

**SingleInstance-only.** The simplest way to configure a VSC on the public cloud is to select one instance type and provision the same instance type as many times as necessary. User can easily decide the number of machines to provision by measuring the storage server performance of the selected instance type. However, as this approach has a single dimension (i.e., the number of machines) in the search space, it cannot search enough candidates of the optimal solution. In our evaluation, we used `i3.xlarge` on AWS as the instance type because it is categorized as a storage optimized instance and provides high-performance local SSD as a storage.

**LocalSSD-only.** Another way to configure the VSC is to use only instance types with local SSDs. In addition to storage optimized instance types, some compute or memory optimized instance types, such as `m5d`, `c5d` and `r5d`, also have local SSD. As local SSD provides high storage performance cost-effectively, it is a good candidate configuration for the storage system. However, it can be an expensive option that provisions more IOPS than the workload actually needs.

**EBS-only/OptimusCloud-like.** The VSC consists of only EBS volumes. EBS volumes can persist data independently

| Benchmark | Workload | Capacity | Request rate (QPS) | Request size | Read request ratio | Access locality |
|---|---|---|---|---|---|---|
| Synthetic benchmark | H | 3 TiB | 9600 | 64 KB | 1.0 | Random access |
| | L | 3 TiB | 2400 | 64 KB | 1.0 | |
| Facebook RocksDB benchmark | A | 3 TiB | 40K | 120 B | 0.86 | Same as described in [11] |
| | B | 600 GiB | 20K | 3 B | 0.0 | |
| | C | 800 GiB | 80K | 17 B | 0.81 | |
| | D | 200 GiB | 40K | 5 B | 0.0 | |
| | E | 400 GiB | 100K | 20 B | 0.29 | |
| | F | 800 GiB | 160K | 19 B | 0.14 | |

**Table 3.** Two benchmarks used to evaluate Mimir. Synthetic benchmark has two workloads: throughput-intensive (`SYN-H`) and capacity-intensive workload (`SYN-L`). Facebook RocksDB benchmark consists of six real-world workloads with different workload characteristics [11]. We have implemented the workloads atop Apache BookKeeper.

from the status of the instance, and users can provision the volume capacity as much as they need. However, to support the workload requires high-performance, it is less cost-efficient option compared to the local SSD. OptimusCloud [29] chose to restrict the volume type to EBS volume because of its persistent characteristic, but our result shows that using only EBS volumes as a storage type can be very expensive compared to the cost-efficient virtual storage cluster with no resource restriction. So we use the terms `EBS-only` and `OptimusCloud-like` interchangeably.

**Heterogeneous.** Finally, for heterogeneous VSC configuration, Mimir optimizes without any resource restriction. The cluster can have any instance and storage type, in which the search space includes both search spaces of `LocalSSD-only` and `EBS-only`, and mix of two.

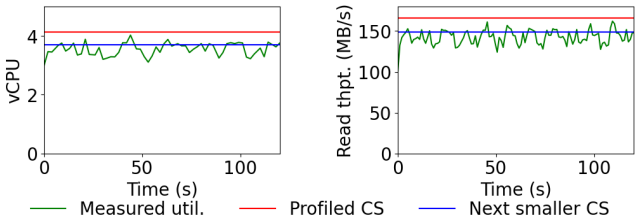


**Figure 9.** The resource utilization of `FR-A`'s *workload fraction*. The resource utilization (green line) shows that the storage server utilizes 85% of the vCPU and storage read throughput of the allocated resources according to the *ContainerSpec* (red line). If any resource allocation reduces to the amount of the next smaller *ContainerSpec* (blue line), the storage server cannot satisfy the performance requirements.

### 4.3 Cost-efficiency of the profiled container size

**Observation 1:** *Mimir profiles a cost-efficient container size to run the storage server for the given workload characteristics. Any workload we tested utilizes at least 83% of the container resources allocated by Mimir.*

In this section, we evaluate how the *ContainerSpec* profiled by Mimir fits for the given workload. Figure 9 shows the example of how the container with the size of profiled *ContainerSpec* works for the *workload fraction* of `FR-A`. Data access pattern of the *workload fraction* we tested is the same as the one of the original workload, and the performance requirements in this example are 6K QPS of data request rate and 450 GiB of data capacity. The *ContainerSpec* profiled for this *workload fraction* is 4.1 vCPU and 166 MB/s read throughput of the storage volume. To evaluate, we ran a docker container with the profiled amount of resources and measured the CPU and storage throughput of the storage server running on the docker container. As Figure 9 shows, the storage server utilizes 85% of both allocated computing power and storage read throughput. We confirmed that the storage server running on the same container (i.e., container with 4.1 vCPU and 166 MB/s read throughput) satisfies the workload requirements, but the storage server on the next smaller container (i.e., container with 3.7 vCPU and 150 MB/s read throughput) following our container size update algorithm (§3.2) cannot meet the performance requirements. We also checked that even a container lack of a single resource could not satisfy the performance requirements. We ran the same experiment on 300 other *ContainerSpecs* we profiled and all the results showed the resource utilization higher than 83% of the resources allocated according to the profiled *ContainerSpecs*.

### 4.4 Resource predictor accuracy

**Observation 2:** *Mimir can predict the cost-efficient container size using interpolation with small percent error.*

We evaluate our *Resource predictor* using interpolation to see how accurately it predicts the right size of *ContainerSpec*. As a dataset, we use the *ContainerSpecs* that are profiled by the *Resource profiler* for the six workloads of `FR`.

As Mimir profiles multiple data points with different performance requirements for each workload, we used all the profiled data as a test dataset to evaluate the interpolation approach. For instance, the maximum data request rate we measured for the `FR-A` workload on `i3.4xlarge` with memory-to-data ratio of 1:16 is 20K QPS. So the *Resource profiler* profiled the right size of 10 different *ContainerSpecs* for the

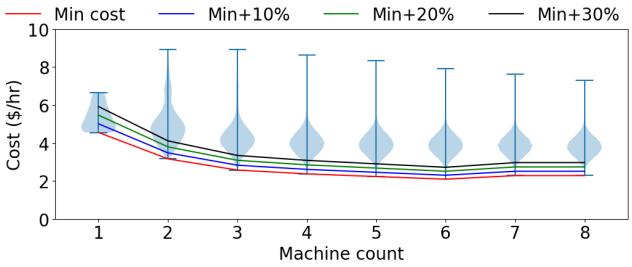


**Figure 10.** Violin plot of FR-F showing the distribution of the cost-efficient VSC configuration price for all possible data distributions. Only a tiny portion of data distributions can find the near cost-efficient VSC configuration.

*workload fractions* of FR-A with the performance requirements of 2K QPS, 4K QPS, ..., 20K QPS. So we evaluated how close the profiled *ContainerSpec* for 4K QPS to the interpolation result of two *ContainerSpecs* for 2K QPS and 6K QPS. Table 4 shows at most 12.9% error for predicting the cost-efficient container size of FR's six workloads using interpolation.

| Workload | Avg % error of the interpolation predictor | | | |
|---|---|---|---|---|
| | CPU | Read thpt. | Write thpt. | Net |
| FR-A | 4.0% | 1.1% | 7.4% | 2.0% |
| FR-B | 2.4% | 6.1% | 8.4% | 1.8% |
| FR-C | 3.1% | 0.8% | 4.8% | 1.1% |
| FR-D | 5.9% | 7.7% | 1.4% | 1.1% |
| FR-E | 2.5% | 0.6% | 2.7% | 1.0% |
| FR-F | 12.9% | 2.5% | 4.5% | 5.1% |

**Table 4.** Percent error of the *ContainerSpec* prediction using interpolation. The interpolation approach predicts the cost-efficient container size with small percent errors.

### 4.5 Analysis on ways of distributing data

**Observation 3:** *The cost-efficient VSC configuration differs greatly depending on how data is distributed.*

We evaluate how much data distribution affects the cost-efficiency of the VSC configuration. Here data distribution is a partition of the workload data. For example, consider a workload $W$ that defines its *workload fraction unit* ($W_f$) as 1/4 of the original size. Then, there are five different partitions (i.e., data distributions) of $W$, which are $\{4W_F\}$, $\{3W_F, W_F\}$, $\{2W_F, 2W_F\}$, $\{2W_F, W_F, W_F\}$, and $\{W_F, W_F, W_F, W_F\}$. The data can be stored in the arbitrary number of storage servers. So we fixed the number of machines to use and ran the optimization algorithm of Mimir for each data distribution.

Figure 10 is a violin plot of FR-F showing the distribution of the cost-efficient VSC configuration price for each data distribution with a fixed number of machines. Mimir finds the most cost-efficient VSC configuration with 6 nodes at the price of 2.09$/hr. For 6 nodes, out of 6043 possible data distributions, only two of them could find the VSC configuration cheaper than 1.1x of the minimum price, which is 2.3$/hr, i.e., except for the best data distribution, only one distribution can find the VSC configuration that costs less than 1.1x of the minimum price. Even for 1.2x and 1.3x of the minimum price, only 24 and 144 data distributions can find the VSC configuration cheaper than the respective prices. In other words, only 2.4% of all possible data distributions can unearth the VSC configuration cheaper than 1.3x of the minimum cost. As we demonstrated, although there are many data distributions and only a few of them can find near cost-efficient VSC configurations, Mimir successfully finds the cost-efficient one using its optimization algorithm.

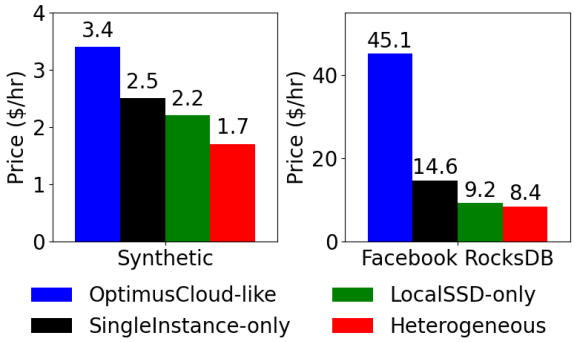


**Figure 11.** The sensitivity analysis of the optimization results of the two benchmarks, SYN and FR. Mimir finds the most cost-efficient VSC configuration under the Heterogeneous compared to the other baseline constraints.

### 4.6 Sensitivity analysis

**Observation 4:** *Mimir finds the most cost-efficient VSC configuration when there is no resource constraint because different workloads prefer different storage types to store data cost-efficiently.*

We conducted the sensitivity test of the optimization results to show how the cost-efficient VSC configuration differs between 1) the workloads with different characteristics and 2) different degrees of resource heterogeneity for the same workload.

Figure 11 shows price comparison of the optimization results with Heterogeneous and other baselines for both SYN and FR. Mimir successfully finds cheaper virtual storage cluster with Heterogeneous than the other baselines and it is up to 5.3x cheaper than the OptimusCloud-like constraint. In both benchmarks, Mimir finds more cost-efficient VSC configuration under LocalSSD-only constraint compared to the OptimusCloud-only constraint. However, it does not mean that every workload data in the benchmarks is more cost-efficient to be stored in local SSD than EBS volume.

Figure 12 shows the storage preference of each workload of SYN and FR. First, SYN-H is the workload that requires high-performance (i.e., throughput-intensive workload) of the storage system. Thus, Mimir finds the virtual storage cluster that only uses local SSD for the cost-efficient solution

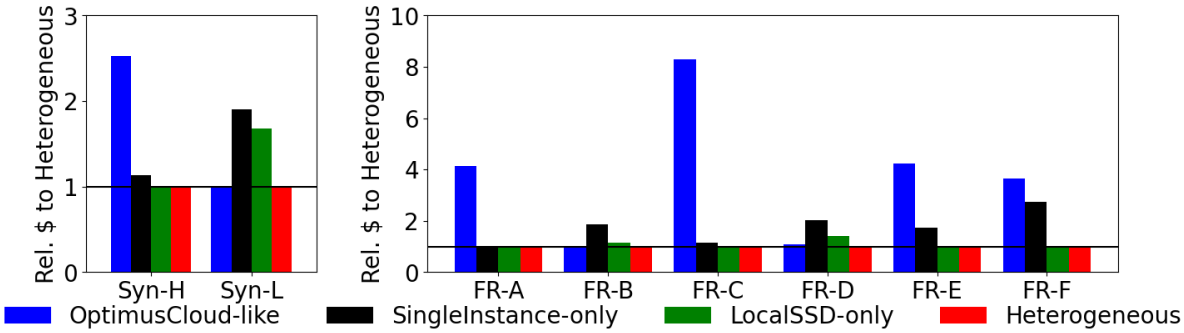


**Figure 12.** The sensitivity analysis of the optimization results of the workloads of the two benchmarks, SYN and FR. Generally, throughput-intensive workloads (e.g., SYN-H, FR-A,C,E,F) prefers local SSD as its storage volume type. In contrast, other workloads (e.g., SYN-L, FR-B,D) that do not require high throughput prefer EBS volume to local SSD. An i3 instance type is a costly option for some workloads (e.g., FR-B,D,E,F) that require the high computing power of the storage server, even if AWS categorized i3 as storage optimized instance type.

with Heterogeneous. On the other hand, if Mimir uses EBS volumes to store data that requires high storage performance, it should provision much higher storage capacity than it needs to provision enough volume IOPS. For example, in the cost-efficient VSC configuration of SYN-H under EBS-only, Mimir provisions total 15 TiB of gp2 volumes to store only 3 TiB of data to get enough volume IOPS. This configuration costs 2.5x higher price compared to the LocalSSD-only or Heterogeneous.

In contrast, SYN-L is the workload that does not require high-performance (i.e., capacity-intensive workload). So local SSD is an expensive storage type to store data of SYN-L, as it under-utilizes storage bandwidth of local SSD. The throughput of gp2 (i.e., 3 IOPS per provisioned GiB) is enough to support the workload. Figure 12 shows that the cost-efficient VSC configuration under LocalSSD-only costs 1.7x higher price than the one with EBS-only.

FR workloads with different characteristics also show different preferences on the volume type. FR-A,C,E,F require 4.13x, 8.3x, 4.2x, 3.6x higher price with EBS-only constraint than LocalSSD-only constraint, respectively, while FR-B,D require 1.1x, 1.3x higher price under LocalSSD-only. As Table 3 indicates, FR-B,D need lower data request rate and smaller data request size than the other workloads, which makes both workloads well suited to EBS volume type.

**Observation 5:** *Considering various instance types is also crucial to find the cost-efficient VSC configuration.*

Not only the volume type, but also the instance type is the important factor that affects the price of the virtual storage cluster. For example, FR-F workload requires the second highest storage system throughput per GiB of data among the workloads of FR, and Mimir finds more cost-efficient VSC configuration under the LocalSSD-only constraint compared to the EBS-only constraint. However, i3.xlarge, a storage optimized instance type (SingleInstance-only), is costly option for the FR-F workload. Instead, the cost-efficient VSC configuration uses c5d instance type under Heterogeneous and LocalSSD-only constraints, in which c5d is a compute optimized instance type that has small capacity of local SSD. This is because the the storage server for FR-F needs high computing power (i.e., CPU-intensive) as the workload requires high data request rate. Similarly, FR-B,D,E prefer m5d or c5d to i3 instance type.

**Observation 6:** *Considering two workloads together in the optimization algorithm can save cost up to 10.3% compared to using two clusters optimized for each.*

Lastly, we evaluate the pairwise workload optimization of the FR's workloads. We ran the *VSC Cost optimizer* for the $\binom{6}{2}$ number of pairwise combinations of the FR's workloads. Table 5 shows the selected combinations of the workloads that minimize the total price of the virtual storage cluster when the cluster should support all the workloads. (FR-B, FR-E) pair yields the highest financial gains, 10.3% lower price, when Mimir considers them together to optimize the virtual storage cluster. FR-B and FR-E are cost-efficient when data is stored in EBS volume and local SSD, respectively. Thus, Mimir finds the VSC configuration that remote EBS volumes for FR-B are attached to the machines for FR-E that have local SSDs. In this way, the cost of provisioning instances for FR-B could be saved. (FR-A, FR-D) pair also have the same property (i.e., they prefer different volume types), but there is no financial gain as no computing power left in the machines of FR-A to support additional workloads in the same machine. By the pairwise workload optimization, Mimir could save total 4% additional cost compared to using six individual virtual storage clusters optimized for each workload.

Despite the large search space for the resource heterogeneity and numerous factors to consider (e.g., complex workload and storage characteristic, many price and performance SLAs), Mimir could find the cost-efficient VSC configuration under Heterogeneous.

| Optimal Cost/Hour | $W_1$ | $W_2$ | $W_1 + W_2$ | Gain |
|---|---|---|---|---|
| $W_1$ = FR-A, $W_2$ = FR-D | $1.86 | $0.46 | $2.32 | 0% |
| $W_1$ = FR-B, $W_2$ = FR-E | $0.33 | $1.8 | $1.91 | 10.3% |
| $W_1$ = FR-C, $W_2$ = FR-F | $2.25 | $2.09 | $4.21 | 3% |

**Table 5.** Pairwise workload optimization of FR application. Mimir finds 10.3% cheaper VSC configuration when it optimizes the configuration for both workloads at once.

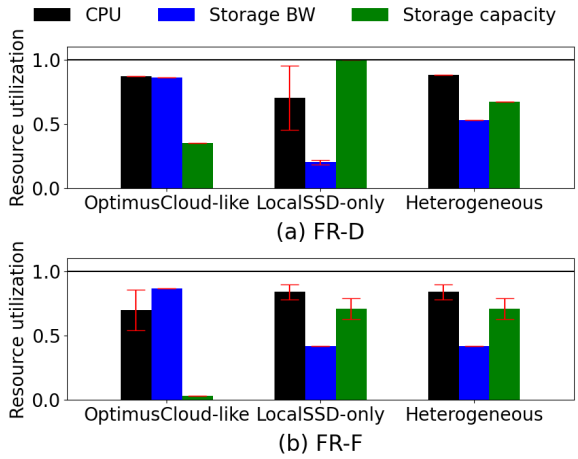


**Figure 13.** Resource utilization of CPU, storage bandwidth and storage capacity under various baseline constraints, evaluated with FR-D and FR-F workloads. The cost-efficient VSC configuration with no resource constraint (Heterogeneous) shows the highest and most balanced resource utilization.

### 4.7 Average resource utilization of cluster

**Observation 7:** *When a workload is run on the cost-efficient virtual storage cluster, it shows high and balanced resource utilization of machines in the cluster.*

We observed that the poorly configured VSC configuration often waste large amount of provisioned resources, which increases the total price of the virtual storage cluster. On the contrary, the cost-efficient VSC configuration shows high and balanced resource utilization.

Figure 13 shows the average resource utilization of the machines in the clusters, in which each VSC configuration is optimized with Mimir under various baseline constraints. We tested with two FR workloads, FR-D and FR-F, and we measured the utilization of CPU, storage bandwidth, and storage capacity (i.e., what fraction of provisioned storage capacity is actually used).

First, FR-D workload under EBS-only constraint uses only 35% of the provisioned volume capacity, because more EBS volume capacity is provisioned than the actual data size to satisfy the throughput requirement of the workload, as EBS volume's throughput is proportional to the provisioned volume capacity. With the LocalSSD-only, only 20% of the provisioned storage bandwidth is utilized because FR-D workload does not require high data request rate. Even though the machines with local SSD offer high-performance storage at a low price, using only the machine types that have local SSD is not the most cost-efficient way to configure the cluster if the storage bandwidth utilization is very low. So, without any resource restriction, Mimir finds the virtual storage cluster that uses both types to store data by attaching EBS volumes to the machines that have small local SSDs. As a result, it achieves more balanced resource utilization under the Heterogeneous constraint.

FR-F workload, on the other hand, uses only 3% of provisioned volume capacity, which is much lower utilization compared to the FR-D, under EBS-only constraint. As FR-F requires very a high data request rate, 33x of the actual data size should be provisioned for the EBS volume capacity to get enough volume throughput. Therefore the cost-efficient VSC configuration only uses local SSD with no resource constraint, in which the resource utilization is also balanced using this configuration.

## 5 Conclusion

Mimir finds cost-efficient virtual storage cluster (VSC) configurations for distributed storage backends. Given workload information and performance requirements, Mimir predicts resource requirements and explores the complex, heterogeneous set of block storage offerings to identify the lowest-cost VSC configuration that satisfies the customer's need. Experiments show that no single allocation type is best for all workloads and that a mix of allocation types is the best choice for some workloads. Compared to a state-of-the-art approach, Mimir finds VSC configurations that satisfy requirements at up to 81% lower cost.

## References


[1] RocksDB. http://rocksdb.org/.

[2] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 469–482, Boston, MA, March 2017. USENIX Association.

[3] Guillermo A. Alvarez, Elizabeth Borowsky, Susie Go, Theodore H. Romer, Ralph A. Becker-Szendy, Richard A. Golding, Arif Merchant, Mirjana Spasojevic, Alistair C. Veitch, and John Wilkes. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Trans. Comput. Syst.*, 19:483–518, 2001.

[4] Eric Anderson, Michael Hobbs, Kimberly Keeton, Susan Spence, Mustafa Uysal, and Alistair Veitch. Hippodrome: Running Circles Around Storage Administration. In *Conference on File and Storage Technologies (FAST 02)*, Monterey, CA, January 2002. USENIX Association.

[5] George E. Andrews. *The Theory of Partitions.* Cambridge University Press, 1976.

[6] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. Workload Analysis of a Large-Scale Key-Value Store. SIGMETRICS '12, page 53–64, New York, NY, USA, 2012. Association for Computing Machinery.

[7] Amazon Elastic Block Store. https://aws.amazon.com/ebs.

[8] Azure Disk Storage. https://azure.microsoft.com/en-us/services/storage/disks/.

[9] Vasanth Balasundaram, Geoffrey Fox, Ken Kennedy, and Ulrich Kremer. A Static Performance Estimator to Guide Data Partitioning Decisions. In *Proceedings of the Third ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPOPP '91, page 213–223, New York, NY, USA, 1991. Association for Computing Machinery.

[10] Muhammad Bilal, Marco Canini, and Rodrigo Rodrigues. Finding the Right Cloud Configuration for Analytics Clusters. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, SoCC '20, page 208–222, New York, NY, USA, 2020. Association for Computing Machinery.

[11] Zhichao Cao, Siying Dong, Sagar Vemuri, and David H.C. Du. Characterizing, Modeling, and Benchmarking RocksDB Key-Value Workloads at Facebook. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 209–223, Santa Clara, CA, February 2020. USENIX Association.

[12] Surajit Chaudhuri, Vivek R. Narasayya, and Ravishankar Ramamurthy. Estimating progress of execution for SQL queries. In *SIGMOD '04*, 2004.

[13] Andrew Chung, Jun Woo Park, and Gregory R. Ganger. Stratus: cost-aware container scheduling in the public cloud. *Proceedings of the ACM Symposium on Cloud Computing*, 2018.

[14] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. *SIGPLAN Not.*, 49(4):127–144, February 2014.

[15] Salvatore Dipietro, Giuliano Casale, and Giuseppe Serazzi. A Queueing Network Model for Performance Prediction of Apache Cassandra. In *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS'16, page 186–193, Brussels, BEL, 2017. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[16] Flexible I/O Tester. http://freshmeat.sourceforge.net/projects/fio.

[17] Jerome Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29:1189–1232, 10 2001.

[18] Google Compute Engine Persistent Disks. https://cloud.google.com/compute/docs/disks.

[19] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[20] Jonathan R. M. Hosking and Jamie Wallis. Parameter and quantile estimation for the generalized pareto distribution. *Technometrics*, 29:339–349, 1987.

[21] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Ling Shen, Liu Tang, Yuxing Zhou, Menglong Huang, Wan Wei, Cong Liu, Jian Zhang, Jianjun Li, Xuelian Wu, Lingyu Song, Ruoxi Sun, Shuaipeng Yu, Lei Zhao, Nicholas Cameron, Liquan Pei, and Xin Tang. TiDB: A Raft-based HTAP Database. *Proc. VLDB Endow.*, 13:3072–3084, 2020.

[22] Amazon EBS volume types. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html#hard-disk-drives.

[23] Flavio P. Junqueira, Ivan Kelly, and Benjamin Reed. Durability with BookKeeper. *SIGOPS Oper. Syst. Rev.*, 47(1):9–15, January 2013.

[24] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. Selecta: Heterogeneous Cloud Storage Configuration for Data Analytics. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 759–773, Boston, MA, July 2018. USENIX Association.

[25] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. Pocket: Elastic ephemeral storage for serverless analytics. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 427–444, Carlsbad, CA, October 2018. USENIX Association.

[26] Chunbo Lai, Song Jiang, Liqiong Yang, Shiding Lin, Guangyu Sun, Zhenyu Hou, Can Cui, and Jason Cong. Atlas: Baidu's key-value storage system for cloud data. *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–14, 2015.

[27] Viktor Leis and Maximilian Kuschewski. Towards Cost-Optimal Query Processing in the Cloud. *Proc. VLDB Endow.*, 14:1606–1612, 2021.

[28] Yongkun Li, Zhen Liu, Patrick P. C. Lee, Jiayu Wu, Yinlong Xu, Yi Wu, Liu Tang, Qi Liu, and Qiu Cui. Differentiated Key-Value Storage Management for Balanced I/O Performance. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 673–687. USENIX Association, July 2021.

[29] Ashraf Mahgoub, Alexander Michaelson Medoff, Rakesh Kumar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. OPTIMUSCLOUD: Heterogeneous Configuration Optimization for Distributed Databases in the Cloud. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 189–203. USENIX Association, July 2020.

[30] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[31] S. Mitra, Shanka Subhra Mondal, Nikhil Sheoran, Neeraj Dhake, Ravinder Nehra, and Ramanuja Simha. DeepPlace: Learning to Place Applications in Multi-Tenant Clusters. In *APSys '19*, 2019.

[32] Kristi Morton, Magdalena Balazinska, and Dan Grossman. ParaTimer: A progress indicator for MapReduce DAGs. pages 507–518, 06 2010.

[33] Barzan Mozafari, Carlo Curino, and Samuel Madden. DBSeer: Resource and Performance Prediction for Building a Next Generation Database Cloud. In *CIDR*, 2013.

[34] Oliver Niehorster, Alexander Krieger, Jens Simon, and Andre Brinkmann. Autonomic Resource Management with Support Vector Machines. In *2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 157–164, 2011.

[35] Andrew Or, Haoyu Zhang, and Michael J. Freedman. Resource Elasticity in Distributed Deep Learning. In *MLSys*, 2020.

[36] Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. Auto-Scaling Web Applications in Clouds: A Taxonomy and Survey. *ACM Comput. Surv.*, 51(4), July 2018.

[37] Supreeth Shastri and David E. Irwin. HotSpot: automated server hopping in cloud spot markets. *Proceedings of the 2017 Symposium on Cloud Computing*, 2017.

[38] John D. Strunk, Eno Thereska, Christos Faloutsos, and Gregory R. Ganger. Using Utility to Provision Storage Systems. In *FAST*, 2008.

[39] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 363–378, Santa Clara, CA, March 2016. USENIX Association.

[40] Haoyu Wang, Haiying Shen, Qi Liu, Kevin Zheng, and Jie Xu. A Reinforcement Learning Based System for Minimizing Cloud Storage Service Cost. In *49th International Conference on Parallel Processing - ICPP*, ICPP '20, New York, NY, USA, 2020. Association for Computing Machinery.

[41] Neeraja J. Yadwadkar, Bharath Hariharan, Joseph E. Gonzalez, Burton Smith, and Randy H. Katz. Selecting the Best VM across Multiple Public Clouds: A Data-Driven Performance Modeling Approach. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, page 452–465, New York, NY, USA, 2017. Association for Computing Machinery.

[42] Peipei Zhou, Jiayi Sheng, Cody Hao Yu, Peng Wei, Jie Wang, Di Wu, and Jason Cong. MOCHA: Multinode Cost Optimization in Heterogeneous Clouds with Accelerators. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '21, page 273–279, New York, NY, USA, 2021. Association for Computing Machinery.