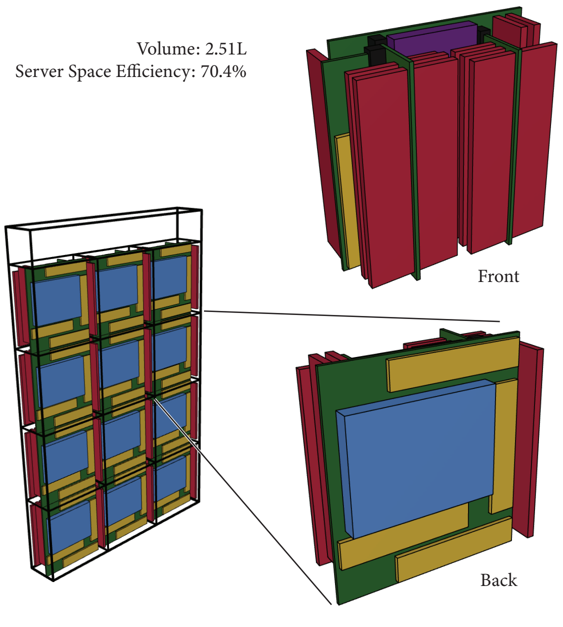
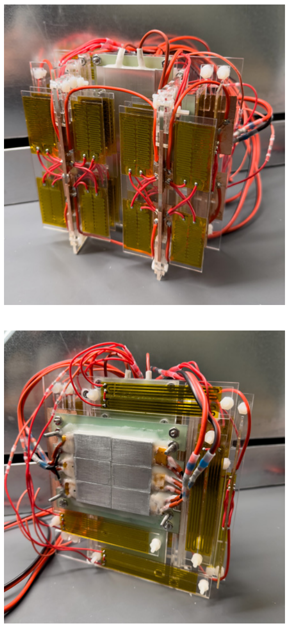
Computational Design of Dense Servers for Immersion Cooling

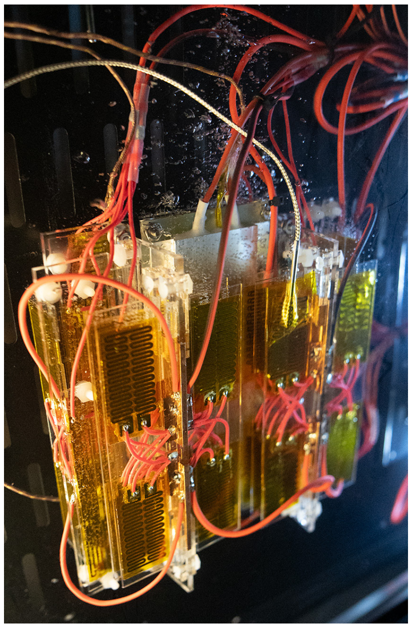
MILIN KODNONGBUA and ZACHARY ENGLHARDT, University of Washington, USA
RICARDO BIANCHINI and RODRIGO FONSECA, Microsoft, USA
ALVIN LEBECK, Duke University, USA
DANIEL S. BERGER, Microsoft, USA
VIKRAM IYER, University of Washington, USA
FIODAR KAZHAMIAKA, Microsoft, USA
ADRIANA SCHULZ, University of Washington, USA



(a) Density optimized server design



(b) Server prototype for thermal validation



(c) Server prototype running in an immersion tank

Fig. 1. Our computational design approach leverages a novel algorithm and domain-specific abstractions to maximize density and server packing in immersion tanks. The solution is validated through physical prototypes tested in an immersion cooling tank. In future applications of immersion cooling technologies, such designs have the potential to significantly reduce the carbon footprint of data center buildings.

The growing demands for computational power in cloud computing have led to a significant increase in the deployment of high-performance servers. The growing power consumption of servers and the heat they produce is on track to outpace the capacity of conventional air cooling systems, necessitating more efficient cooling solutions such as liquid immersion cooling. The superior heat exchange capabilities of immersion cooling both eliminates the need for bulky heat sinks, fans, and air flow channels while also unlocking the potential go beyond conventional 2D blade servers to three-dimensional designs. In this work, we present a computational framework to explore designs of servers in three-dimensional space, specifically targeting the maximization of server density within immersion cooling tanks. Our tool is designed to handle a variety of physical and electrical server design constraints. We demonstrate our optimized designs can reduce server volume by 25–52% compared to traditional flat server designs. This increased density reduces land usage as well as the amount of liquid used for immersion, with significant reduction in the carbon emissions embodied in datacenter buildings. We further create physical prototypes to simulate dense server designs and perform real-world experiments in an immersion cooling tank demonstrating they operate at safe temperatures. This approach marks a critical step forward in sustainable and efficient datacenter management.

Authors' Contact Information: Milin Kodnongbua, milink@cs.washington.edu; Zachary Englhardt, zacharye@cs.washington.edu, University of Washington, USA; Ricardo Bianchini, ricardob@microsoft.com; Rodrigo Fonseca, rofons@microsoft.com, Microsoft, USA; Alvin Lebeck, alvy@cs.duke.edu, Duke University, USA; Daniel S. Berger, daberg@microsoft.com, Microsoft, USA; Vikram Iyer, vsiyer@cs.washington.edu, University of Washington, USA; Fiodar Kazhamiaka, fkazhamiaka@microsoft.com, Microsoft, USA; Adriana Schulz, adriana@cs.washington.edu, University of Washington, USA.

ACM 1557-7368/2024/12-ART210
https://doi.org/10.1145/3687965

CCS Concepts: • **Computing methodologies → Shape analysis**; • **Hardware → Physical design (EDA)**.

Additional Key Words and Phrases: server design, fabrication-oriented design, immersion cooling, sustainable design

ACM Reference Format:
Milin Kodnongbua, Zachary Englhardt, Ricardo Bianchini, Rodrigo Fonseca, Alvin Lebeck, Daniel S. Berger, Vikram Iyer, Fiodar Kazhamiaka, and Adriana Schulz. 2024. Computational Design of Dense Servers for Immersion Cooling. *ACM Trans. Graph.* 43, 6, Article 210 (December 2024), 20 pages. https://doi.org/10.1145/3687965

1 Introduction

Datacenters are the fundamental infrastructure that provide many businesses with the computational power needed to function. As a consequence, datacenters collectively consume approximately 1% of the world's electricity [Masanet et al. 2020], and the deployment of new datacenters is escalating rapidly [Gooding 2024; Kearney et al. 2024]. The heat generated by this power draw is mitigated by sophisticated cooling systems that themselves consume 10–20% of the total power consumption of the servers [Barroso et al. 2018; Wang et al. 2024]. It is projected that the power consumed by servers in the near future will exceed the domain reasonably handled with air cooling technology [IEEE-HIR 2023], which is the currently the most common form of datacenter cooling. Consequently, datacenter operators are considering using alternatives such as liquid immersion cooling [ExxonMobil 2024; Jalili et al. 2021; Ramakrishnan et al. 2021], which involves immersing servers with non-conductive fluid that dissipates heat more effectively than air and uses less power.

This new landscape raises the question: How can servers be designed to take advantage of new cooling technologies? Air-cooled server designs are optimized for efficient airflow and heat transfer, and components are typically arranged on a single printed circuit board (PCB), i.e., a two-dimensional region. Existing prototypes of immersion-cooled servers resemble stripped-down versions of air-cooled servers with fans and heat sinks removed [Avalos et al. 2022; Jalili et al. 2021]. However, immersion liquid can flow freely across the server components, enabling an arrangement of servers in three-dimensional space without sacrificing thermal performance. This larger design space presents an opportunity to drastically redesign servers.

In this work, we propose a novel computational design method to optimize server layouts for immersion cooling, with a focus on maximizing density. Specifically, we aim to customize server PCBs and arrange components (e.g., CPUs, DIMMs, SSDs, etc.) in three-dimensional space to maximize the number of servers that can be housed within an immersion cooling tank. Denser server designs offer significant benefits as they reduce both land usage and the amount of cooling fluid required for immersion. These changes directly impact the embodied carbon associated with building construction and fluid manufacturing, and align with the sustainability goals of major datacenter operators.[Amazon 2021; Google 2021; Joppa 2021] Given the scale of modern datacenters, even small improvements in server density result in significant overall impact.

The challenges with this problem, however, are that (1) the number of possible arrangement of components and PCBs that form a server is combinatorially large (a typical server can have more than 30 components); (2) the number of ways such servers can be packed within the tank is exponential and depends on the designs of an individual server; and (3) designing a server, especially within an immersion liquid environment, requires multiple considerations, such as thermal properties of components, fluid interactions and phase changes, signal integrity, and serviceability, which can be difficult to measure and computationally expensive to evaluate. This is a complex problem that can be difficult for designers to explore; computationally, it implies a large-scale search and nested optimization loops with multiple performance objectives which are expensive to compute.

We address these challenges with three key contributions. First, we utilize domain-specific knowledge to simplify the problem. We observe that arbitrary orientations of components are undesirable because they complicate manufacturing Further, having all servers with unique designs arranged in complex patterns within the tank would not only increase manufacturing costs but also hinder serviceability as such arrangements make it difficult to remove and service servers and raises the cost of training engineers to service each unique design. From these considerations, we: (1) formulate a search space using axis-aligned components, (2) limit to one or two server design variations within the tank, and (3) stack servers vertically in a regular grid, facilitating easy removal of server stacks, referred to as serviceable units (SU). Further, experts in immersion cooling have observed that effective thermal performance can be achieved by ensuring adequate clearance around components and constraining their orientation. We use this domain knowledge to formulate thermal considerations as constraints that can be evaluated through cost-effective geometric checks instead of expensive fluid simulations.

With these simplifications, the problem becomes similar to the classic 3D bin packing problem (3D-BPP), which is already known to be NP-hard. However, the server design problem adds complexity with several additional constraints. First, server components cannot be placed freely in 3D space; they must be mounted on a PCB where wire traces deliver power and route signals between them. It is also required that all PCBs and components are connected to form a single system. This makes it non-trivial to apply existing 3D-BPP heuristics and algorithms, which rely on all items being independent, to search over the placements of components and PCBs. Second, physics laws regarding signal propagation impose constraints on the maximum wire length between two components and the number of PCB-to-PCB connections between them. This prohibits us from using existing 3D-BPP algorithms only on the set components and then augmenting the solution with a large number of PCBs to connect the components together because it is difficult to measure the wire length between components, which depend on the placements of both the components and PCBs, and ensure that they are within the limits.

We propose a novel tree representation of the server design, which guarantees connectivity by construction, and formulate the optimization of the server design as a mixed integer linear programming (MILP) problem. Given the quantity of each component and PCBs per server, our MILP formulation outputs the design decisions for each component and PCB (e.g., position, orientation, and which PCB they are connected to) such that it satisfies the thermal constraints (i.e., the minimum distance between two distinct components and the orientation) and the electrical constraints (i.e., the maximum wire length connecting certain components). Our formulation allows such constraints to be written linearly and thus allows us to leverage off-the-shelf MILP solver.

Lastly, we build physical prototypes that validate our model. Using a tank filled with two-phase coolant and prototypes that resemble the shape and power draw of server components, we conduct experiments to determine the minimum distance between components necessary for sufficient cooling. We also explore how the orientation of components might obstruct or slow the natural flow of the liquid, and measure the impact on cooling performance.

We evaluate our method across several CPU and GPU server specifications through comparisons with human-generated designs and two dimensional design baselines on the density metric, and report the estimated reduction in carbon emissions from using denser configurations in datacenters. We also build a prototype of a generated server design to validate that the servers in dense configuration can satisfy thermal requirements. The results show that our approach outperforms the 2D baselines with 25–52% reduction in volume, and our physical experiments shows that our generated design satisfies the thermal constraints.

2 Related Work

Server Designs. Reducing the operating costs of datacenters through improved server designs has been an interesting research topic. Frachtenberg et al. [2011] list design considerations for air-cooled servers and present a new server design with better airflow and thermal efficiency. Sakanova et al. [2019] optimize the angles and spacing of DIMMs in hybrid-cooled servers where only the CPUs are liquid cooled, using genetic algorithms and computational fluid dynamics (CFD) simulations to evaluate the objectives. Liu and Yu [2021] characterize the thermal performance of immersion-cooled servers across combinations of power draw and physical distance, using physical prototypes and CFD simulations to provide valuable data on the effects of physical layout on thermal performance. To the best of our knowledge, we are the first to present a computational approach that optimizes server designs in three-dimensional space for density in a liquid immersion cooling system.

Manufacturing-Oriented Design. Our work builds on a large body of work in computational tools to optimize designs for a given performance under manufacturing constraints. These tools have been applied in various domains, including architecture [Hafner and Bickel 2021; Ren et al. 2022; Vouga et al. 2012], garments [Narayanan et al. 2018; Pietroni et al. 2022], 3D-Printed shapes [Bächer et al. 2014; Lu et al. 2014; Prévost et al. 2013], robotics [Du et al. 2016; Kodnongbua et al. 2023], and mechanical toys [Coros et al. 2013; Skouras et al. 2013; Zhang et al. 2021], to name a few.

A fundamental challenge in server design for immersion cooling is the high cost of simulating cooling performance. This requires modeling the liquid's state changes and interactions with components, which is both computationally expensive and difficult to model accurately. Previous work has addressed the challenge of expensive evaluation through pre-computation [Schulz et al. 2017; Shugrina et al. 2015], approximating simulation results using machine learning [Kim et al. 2019; Otness et al. 2021; Umetani and Bickel 2018], or Bayesian optimization strategies [Piovarči et al. 2020; Tian et al. 2023]. However, these approaches struggle with the large search space and the high cost of gathering extensive physical data for training.

Our work leverages expert guidance to model liquid interactions as constraints and follows the general method of formulating a mixed integer programming (MIP) problem. This method has been applied in various applications, such as bespoke frame structure [Wang et al. 2023], LEGO sketch art [Zhou et al. 2023], knit graph representation [Mitra et al. 2023], ink selection [Ansari et al. 2020], and pop-up design [Huang et al. 2023]. We further validate the formulation with physical prototypes immersed in an actual tank.

Bin Packing. Related to our problem are well studied two and three dimensional bin packing problems (BPP). Common exact algorithms use branch-and-bound [Martello et al. 1998], mixed-integer programming formulation [Chen et al. 1995], and column generation [Eley 2005]. Due to NP-hardness of the problems, many placement heuristics [Crainic et al. 2008] and meta-heuristic approaches have been proposed, such as genetic algorithms [Wu et al. 2010] and guided local search [Faroe et al. 2003]. We also refer readers to the surveys by Lodi et al. [2002]; Zhao et al. [2016] for 2D- and 3D-BPP approaches. More recently, deep reinforcement learning methods have been proposed to an solve online 3D-BPP where all items to be packed are unknown before start [Hu et al. 2017, 2020; Zhao and Xu 2022]. Moreover, the problem of packing irregular 3D objects have been studied [Cui et al. 2023; Lamas-Fernandez et al. 2023; Liu et al. 2015]. The challenge with designing servers in 3D, even though we model components as cuboids, is that components cannot just be placed anywhere in space. They must obey certain structures and constraints such as connections between components and wire lengths. While placement of components on the same PCB can be considered a 2D-BPP, the arrangements of multiple PCBs in 3D space is difficult due to interactions between components on different PCBs whose degrees of freedom is restricted to 2D. This makes existing heuristics and approaches non-trivial to apply to our problem. Our proposed MILP formulation encodes these constraints, enabling us to efficiently solve the server component packing problem while adhering to the structural and spatial restrictions of 3D server designs.

Chip Design and Floorplanning. Another similar problem is the VLSI floorplanning problem that tries to minimize the chip area given a set of modules to be placed and relevant electrical constraints. This problem can be seen as variations of 2D-BPP; hence, many meta-heuristics approaches such as simulated annealing and particle swarm optimization approaches have been proposed. We refer to the survey by Laskar et al. [2015] for a complete overview. More recently, a deep reinforcement network approach has been propose for chip design [Mirhoseini et al. 2021]. Three-dimensional floor planning has also been explored using simulated annealing [Ababei et al. 2005; Cong et al. 2004] and differential evolutionary approaches [Rani et al. 2013]. However, 3D chips are typically stacked 2D layers and they can only have a few layers due to thermal constraints, making these approaches non-applicable to our problem.

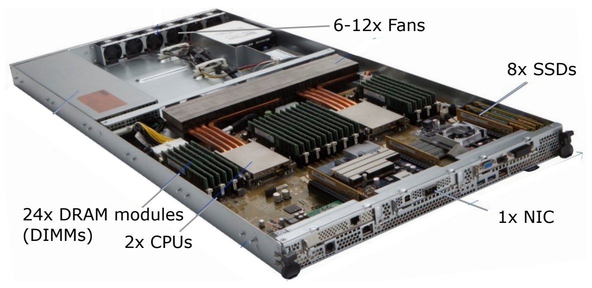


Fig. 2. **Typical Cloud Server** Components in today's blade servers are arranged on a two-dimensional PCB. When air-cooled, large heat sinks, fans, and additional space are required at the back.

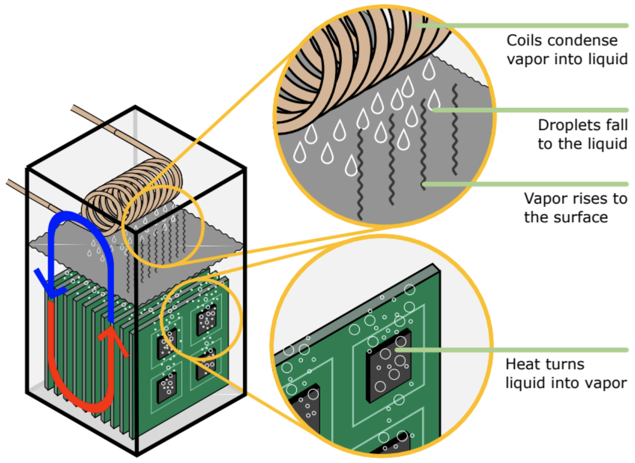


Fig. 3. **Two-phase immersion cooling.** Servers immersed into the dielectric liquid that changes phase. Vapor rises to the top where it recondenses back to liquid form. Figure rights belong to LiquidStack.

3 Background

3.1 Servers

Physically, servers are an assembly of server components enclosed in a chassis. The main components of a server includes CPUs, memory cards (DIMMs), networking cards (NICs), solid state drives (SSDs), and optionally GPUs. Other components such as voltage regulators deliver necessary power to the main components. These components are assembled on one or more PCBs whose traces deliver power and facilitate communication. Components are attached to a PCB through connectors such as PCIe slots, CPU sockets, and DIMM slots; and PCBs typically have many layers for traces to be routed. A server can have multiple PCBs to arrange components in a certain form factor. Power supply units can be mounted locally or connected via a cable. For example, a 2-socket server can include 2 CPUs (hence, 2-socket), 12 DIMMs per CPU, 4 SSDs per CPU, and one NIC for the server (see Figure 2).

This paper only focuses on the arrangement of main components and PCBs which define the high level geometry and form factor of the server. We do not optimize for PCB trace routing and placement of other small components.

3.2 Server Cooling

Datacenter servers require active cooling to maintain the temperature of server components within their recommended operating range. Maximum temperatures are in the range of 80–100°C and vary by components [Intel 2023; Lee et al. 2015; Ye et al. 2018]. The most prevalent technology is *air-cooling*, where servers are outfitted with fans that blow cool air across the server components and their heat sinks—components made of thermally conductive material that are attached to components to help absorb and disperse heat (see copper pipes and aluminium heat sink in Figure 2). Integrated circuits (ICs)—found on CPUs, GPUs, NICs, DIMMs, and SSDs—produce the majority of heat.

The projected power density offuture GPUs and CPUs exceeds what air cooling technologies can handle efficiently [Fan et al. 2018; Jalili et al. 2021; Kheirabadi and Groulx 2016; Sun et al. 2019]. Air-cooling is efficient up to ≈60 W/cm^2 [IEEE-HIR 2023; Refai-Ahmed et al. 2020], beyond which required air temperature set-points, fan speeds, and heat sink sizes make air cooling more costly than alternative cooling solutions. For reference, the power density of an NVIDIA H100 GPU chip exceeds 80 W/cm^2. Other factors, such as 3D chip designs used for high-bandwidth GPU memory [Hilson 2024; IEEE-HIR 2023], motivate the need for better cooling technologies.

Consequently, there is an active development of alternative cooling technologies from partially liquid-cooled to immersive cooling [IEEE-HIR 2023]. Pumped liquid cooling, such as cold plates, operates by pumping coolant directly to high-power ICs—typically CPUs and GPUs [Norrie et al. 2020]—while other relatively low-power components are cooled with air. In immersive cooling, servers are immersed in a tank filled with non-conducting liquid. Single-phase cooling (1P cooling), which has been used in datacenters [Zhong 2019], uses dielectric fluid such as mineral oil and a pump to circulate liquid within the tank and through a radiator cooling loop. Two-phase liquid immersion cooling (2P cooling) uses a special liquid that has a low boiling point (50–60°C) and cools server components by dissipating heat via a phase change. Such fluid has a significantly higher heat capacity than air [Ramakrishnan et al. 2021], and cooling occurs without a pump.

Our study focuses on server design in the context of 2P cooling. In 2P cooling, servers are immersed in dielectric fluid within a closed tank; as fluid makes contact with a hot component, it changes phase (vaporizes) if the component temperature exceeds the fluid's boiling temperature, creating vapor bubbles. This process is referred to as the *nucleation* of vapor bubbles. The vapor floats to the top of the tank, where it interacts with condenser coils and precipitates (see Figure 3). This process allows for the temperature of server components to be maintained within a few °C of the fluid's boiling temperature. 2P cooling is expected to support power densities in the hundreds of W/cm^2 [3M 2023; IEEE-HIR 2023].

4 Server Design Abstractions and Approach

4.1 Server Design Considerations and Abstraction

Server designs are influenced by thermal and electrical considerations, and in a datacenter setting with many thousands of servers, maintenance and space constraints are major considerations. Our method and the server design space are highly motivated by these considerations.

Manufacturability considerations: Given that server components typically come in rectangular shapes and the manufacturing of PCBs is limited to two-dimensional planes, we model components and PCBs as cuboids. Furthermore, tanks tend to be cuboids. In the context of packing cuboid objects to a cuboid container, we simplify the problem by only considering orthogonal orientations of components. While this simplification may not lead to the optimal packing, it is important for ease of manufacturing and serviceability.

Thermal considerations: In air-cooled servers, cold air is pulled by fans across the server components, from the front to the back of the rack. Consequently, servers are designed to minimize interference with the flow of air across the ICs, resulting in the ‘blade’ designs (see Figure 2) that have been in use for decades [Shaw 2016; Shaw and Goldstein 2014]. Air cooling requires large heat sinks to expand the surface area of hot components and large gaps between components to allow for flow, which limits the density of servers. The blade design has persisted for servers that are partially liquid-cooled, e.g., with cold-plates.

In 2P cooling, due to the fluid’s remarkable thermal properties, heat sinks are not required or can be significantly smaller than what is used for air cooling [Dymyd et al. 2020]. However, the cooling performance is affected by the power density, the surface texture, and the orientation and arrangement of components. Nucleation of vapor bubbles on hot surfaces can hinder the flow of fluid that returns to cool the surfaces. At high enough power density, the vaporization can occur at a faster rate than the inflow rate of the fluid, causing a phenomenon known as the *dry-out effect* where the components can no longer be cooled sustainably. Textured surfaces such as micro-meshes can alleviate this effect by forming smaller sized bubbles that quickly detach and float to the top of the tank [Zhang et al. 2020]. Surface orientation also affects the maximum power density before the dry-out effect [3M 2023]. Moreover, the flow of vaporized fluid from hot components can brush against downstream components and affect liquid contact [An et al. 2018]. Hence, the arrangement of components can impact the cooling performance if the vapor flow is restricted or accumulates on a surface (e.g., trapped within a concave structure).

We consulted a set of 2P cooling experts from a major cloud provider with significant experience in the thermal characteristics of immersion-cooled servers. These experts suggest that leaving a gap between components would be sufficient for cooling as long as there is no vapor trapping. We therefore constrain components to be at least some distance apart, which we model by adding to the component dimension, and allow our model to take that value as a parameter since it will depend on component specification and thermal properties of the liquid. In addition, we constrain all PCBs to be oriented vertically to allow for the unimpeded flow of fluids and vapors, although our model can also express horizontally oriented PCBs and components. Our model can also be extended to 1P cooling by leaving larger gaps between components and by modeling larger components to accommodate for heat sinks. For non-immersive cooling technologies that deliver fluids directly through pipes, future work is needed to model the pipes and their connectivity to represent the designs for optimization.

Electrical considerations: Components must communicate with one another through traces (wires) routed inside a series of PCBs to form a single connected system. To ensure valid behaviors, components can only tolerate a certain amount of signal loss defined by the industry standards (e.g., JEDEC DDR5) for that device. The signal loss is mainly due to the trace length between two components and also other factors such as the trace width, PCB materials, and the number of turns. These factors constrain where components can be placed. In this work, we constrain component placement by limiting the maximum trace length calculated using L1 distance along the PCBs connecting two components. Although we do not consider other factors that affect signal loss which can be implementation specific, they can be accounted for by adding a safety factor to the distance constraints. While it is difficult to guarantee the feasibility to route actual traces within the L1 constraints without knowing what components may be in the way, server PCBs often have many layers (often more than 10) to route crossing traces and accommodate the issue to some extent. Modelling trace routing precisely is a subject for future work.

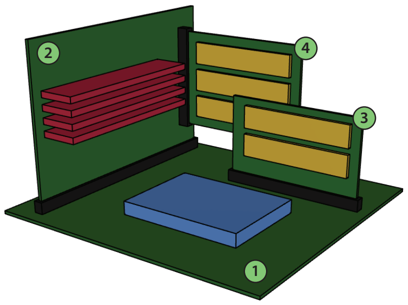
Serviceability considerations: At the scale of datacenters, server failures and subsequent maintenance is a regular occurrence that requires dedicated technicians [Lyu et al. 2023]. Easy access to servers and impact on surrounding servers are important aspects of the maintenance process. In conventional air-cooled blade designs, individual servers can be removed from racks by sliding them out of their enclosure with minimal impact on surrounding servers. Maintenance procedures, such as replacing failed components, are straightforward as all components are readily accessible. In 1P cooling, since servers are cooled by the liquid, they need to be turned off before being removed from the tank. Therefore, to service a particular server, one would need to shut off and pull out any servers on top or in the way of removing the faulty server. In 2P cooling, tanks only operate when they are sealed, meaning that all servers in a tank have to be turned off before they can be serviced. However, accessible arrangements of servers help reduce the service time and the time the tank has to be opened which can cause leaks and losses of fluid/vapor. In addition, each maintenance procedure requires technicians to be trained. Therefore, it is favorable to have fewer server variations and less complicated processes to service a server from its container and a component from a server. In this work, we limit to one or two server design variations within the tank, and we stack servers vertically forming a serviceable units (SU). We further simplify the space by having all SUs be identical and arranged in a regular grid within the tank.

Space considerations: In datacenters, servers are arranged in racks with standard dimensions, such as those in the Open Compute Rack specification [Shaw 2024]. This standardization governs the dimensions of the server chassis and how many servers can fit enabling a consistent datacenter layout and equipment re-use across multiple generations of server hardware. Standardization also impacts the density of server deployments, as space within a rack slot can be wasted when its over-sized for the necessary hardware. 2P cooling tanks have few existing standards for enclosure, and some prototypes inherit the rack standards for their dimensions [Jalili et al. 2021]. In this work, we propose two formulations to optimize server designs where the tank dimension is given and not given.

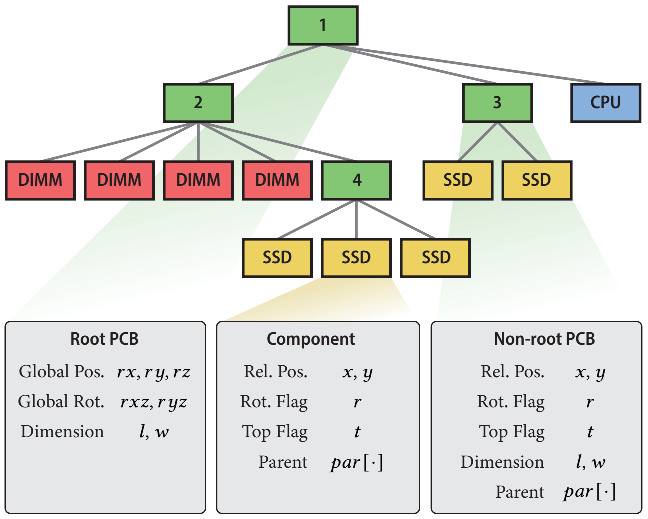
In summary, we make five key design decisions from the aforementioned considerations. First, we model components and PCBs as cuboids and only consider orthogonal orientations. Second, we constrain components with a minimum separation, which we model by adding to the component dimension. Third, we optionally constrain PCBs to be vertically oriented. Fourth, we constrain the maximum wire length between two components along the PCBs according to the limits of the wire signal integrity. Fifth, we limit to one or two design within the tank. We stack servers vertically forming an SU that can be tiled in a regular grid within the tank. Note that while only considering axis-aligned and cuboid components can be a limitation of our model due to linearity, the remaining design decisions only define one possible configuration of the model that deems appropriate for 2P immersion cooling and can be adjusted to suit specific requirements.

4.2 Representing Design Space

We will now describe the design space of the server, its representation, and the decision variables. As with traditional server designs, we consider designs where each component is connected to one PCB, a PCB is connected to another PCB if at all, and every component is connected as a single assembly. From these considerations, we propose to represent a server design using a rooted tree where each internal node corresponds to a PCB and each leaf node corresponds to a component because a tree guarantees that all components are connected and there is a unique path between any two nodes by design. We will use the term *element* to collectively refer to a PCB or a component. The parent of a non-root node indicates which PCB an element is mounted to. Recall that we model elements as axis-aligned boxes. To describe how a non-root element is attached to its parent, we store its position ($x, y \in \mathbb{R}_{\geq 0}$) with respect to its parent's coordinate frame, whether it is rotated by 90 degrees ($r \in \{0, 1\}$), and whether it is on top of its parent ($t \in \{0, 1\}$). In addition, for PCB, we store its length, width, and thickness ($l, w, h \in \mathbb{R}_{\geq 0}$). For components, we store its padded dimension ($l, w, h \in \mathbb{R}_{\geq 0}$). As a convention, we use right-hand axis ordering and use length (l), width (w), and height (h) to describe dimension in x, y, and z axis, respectively (see Figure 5a). Each PCB can be parallel to one of the three canonical planes xy, xz, and yz (see Figure 5 (b-d)). The orientation of a non-root PCB must be different from its parent and is also governed by the rotation flag r to select the remaining two planes. The component can be oriented in two ways with its height (h) being always orthogonal to the PCB plane. Optionally, to control the global orientation and position of the entire server, we include the orientation of the root PCB using one-hot



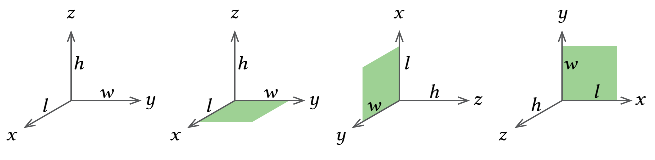
(a) Example Server Design



(b) Tree Representation and Decision Variables

Fig. 4. **Server Design Representation.** (a) an example server design; (b) the tree representation of the design where each node represents a PCB or a component and edges represent the connections. Listed below are decision variables for each node types.

variables ($rxz, ryz, rxy \in \{0, 1\}$) and the position of the root PCB ($rx, ry, rz \in \mathbb{R}$). Figure 4 shows an example of a server design (a) and the corresponding tree representation (b).



(a) Axis Convention (b) *xy* PCB (c) *xz* PCB (d) *yz* PCB

Fig. 5. **Convention.** (a) Global coordinate frame and axis convention. (b-d) Coordinate frame of a PCB oriented in different directions.

In our settings, we assume that we have a predetermined number of PCBs and components. All variables described earlier, with the exception of component dimensions and PCB thickness, are considered decision variables. The parent of each non-root element is another decision variable which we represent using a one-hot encoding of PCBs ($par[p] \in \{0, 1\}$ denotes whether p is the parent of the current element). The parent of a component can be any PCB. To avoid cycles, the parent of a PCB is restricted PCBs with lower index than itself. Note that it is possible to have a PCB without any children. This is desirable because an optimizer can choose to use *fewer* and up to the given number of PCBs without changing the problem definition.

4.3 Method Overview

Our approach is as follows. Given a server specification (the number of components and PCBs), the dimension of each component, and the dimension of the immersion tank, our method generates (1) the server design(s) and (2) the packing of servers inside the tank that maximizes the packing density. We formulate this problem as a bi-level optimization problem where we first partition the tank to a regular grid with different number of rows and columns along the length and width of the tank, where each grid cell is a SU that can be pulled up to be serviced. For each grid partitioning, we optimize for a set of one or two servers in the SU that minimizes the height (Section 5). Finally, we enumerate through all possible grid partitioning whose cell are of sufficient size and return one that maximizes the total number of servers in the tank.

Alternatively, our formulation can be adjusted to search for a dense design without constraints on maximum length and width. As 2P immersion cooling tech is a less-mature technology compared to air cooling, there are no prolific standards for tank dimensions, making it practical to customize tank dimensions and server placement for the generated design. We refer to this as the *standalone* formulation.

Since the design space scales exponentially with the number of elements, for servers with many elements we propose to create *bundles*—i.e., to group a set of components placed next to one another and treat them as a new component. In principle, a bundle can have any number of components, but for our experiments, we propose to only group components of the same type. Our insight is that while a server can have many components, there are only a few component types each of which with many replicas (e.g., a server could have more than 20 DIMMs). Components within a bundle share the same dimensions and will pack well together. They also share the same wire length constraints so they would otherwise be placed in close proximity without bundling. We refer to Appendix D for details.

To facilitate manufacturing, we model a *connector* for each PCB-to-PCB connection, a structure that physically holds the child PCB orthogonally. The position and dimension of the connector are only determined by the position and dimension of the child PCB. We constrain that connectors do not overlap with any other element except for their child PCB. The formulation of connectors, written in more detail in Appendix E, is independent of the formulation of the main problem, and so the remainder of the exposition will proceed without connectors.

5 Server Design Optimization

We formulate the problem of finding the densest design for a set of servers as an MILP. We assume the following information is given: (1) length L_{SU} and width W_{SU} of the serviceability unit (SU), (2) the number of servers and the number of PCBs and components in each server, (3) padded component dimensions, (4) the height of a PCB, and (5) wire length constraints between component u and v (d_{uv}). Each server is represented by a tree as described earlier where we have a predetermined set of nodes, each component node has a predetermined padded dimension, and each PCB node has a predetermined thickness. To recall, the decision variables for each server include: (1) translation rx, ry, rz and orientation rxz, ryz of the root PCB; (2) position x, y, rotation flag r, and top flag t of non-root elements; (3) dimension l, w of PCBs; and (4) one-hot variables of parent of non-root elements $par[p]$.

We define a *global bounding box* to be a box that encloses all the servers. The *global bounding box* spans $[-L_N, L_P]$, $[-W_N, W_P]$, and $[-H_N, H_P]$ in x, y, and z-axis, respectively, where L_N, L_P, W_N, W_P, H_N, $H_P \in \mathbb{R}_{\geq 0}$ are variables.

For packing within the SU, we constrain the length and width of the *global bounding box* to be the given length and width of the SU:

$$L_N + L_P = L_{\mathrm{SU}} \tag{1}$$
$$W_N + W_P = W_{\mathrm{SU}} \tag{2}$$

and minimize the height: $H_N + H_P$, which directly corresponds to volume (hence, density).

Without the SU constraints, which we refer to as the *standalone* formulation, we minimize the sum of dimensions: $L_N + L_P + W_N + W_P + H_N + H_P$, as a linear proxy to density[1].

We constrain all elements u to be inside the global bounding box. For this, we need to compute the positions and dimensions of each element in the global coordinate frame from the tree representation. Let $gx, gy, gz \in \mathbb{R}$ be the corner (with the lowest coordinate) and $gl, gw, gh \in \mathbb{R}$ be the dimension of an element in the global coordinate frame. This can be computed using a series of affine transformations based on the relative positions and orientation flags along each node to the root of the tree. We will describe how we linearize these variables and other non-trivial variables introduced in this section later in Section 5.3. Through out this paper, we will use a subscript to denote which element a variable belongs to. For example, gx_u is the global x-coordinate of element u. We can now write the bounding box constraints as:

$$-NL \leq gx_u \tag{3}$$
$$-NW \leq gy_u \tag{4}$$
$$-NH \leq gz_u \tag{5}$$
$$gx_u + gl_u \leq L \tag{6}$$
$$gy_u + gw_u \leq W \tag{7}$$
$$gz_u + gh_u \leq H \tag{8}$$

We also constrain that no two elements u and v overlap. This constraint is a disjunction between six cases—u to the left, right, top,

[1]Due to linearization, such objective does not necessarily correlate with density and would steer an optimizer towards solutions where the dimensions are similar.

bottom, front, or back of v, and can be written as:

$$\bigvee\{gx_u + gl_u \le gx_v,\; gx_v + gl_v \le gx_u,\; gy_u + gw_u \le gy_v,\; gy_v + gw_v \le gy_u,\; gz_u + gh_u \le gz_v,\; gz_v + gh_v \le gz_u\} \tag{9}$$

Let $pl, pw, ph \in \mathbb{R}$ be the dimension of an element written in the coordinate frame of its parent. We constrain an element u to be within its parent p.

$$x_u + pl_u \le l_p \tag{10}$$

$$y_u + pw_u \le w_p \tag{11}$$

Lastly, we constrain the wire length between components u and v to be within the specified limit d_{uv}

$$D(u,v) \le d_{uv} \tag{12}$$

where $D(u,v)$ is the wire length between u and v on the tree.

5.1 Actualization to 3D Geometry

We now describe the calculation of global positions (gx, gy, gz) and dimensions (gl, gw, gh) in more details. For the root PCB, this is trivial. $gx, gy, gz = rx, ry, rz$ and $gl, gw, gh = l, w, h$. However, since the position and orientation of other elements are written relative to their parents, we need to compute the transformation matrix for each PCB that transforms the PCB coordinate frame to the global frame. Let us first assume that we know the global orientation of the current PCB, which we will denote by one-hot variables $Txy, Txz, Tyz \in \{0,1\}$, and the global translation of the PCB Tx, Ty, Tz. We can write the transformation matrix as:

$$\mathbf{T} = \begin{bmatrix} Txy & Txz & Tyz & Tx \\ Tyz & Txy & Txz & Ty \\ Txz & Tyz & Txy & Tz \end{bmatrix} \tag{13}$$

We can now compute the global position and dimension for each element:

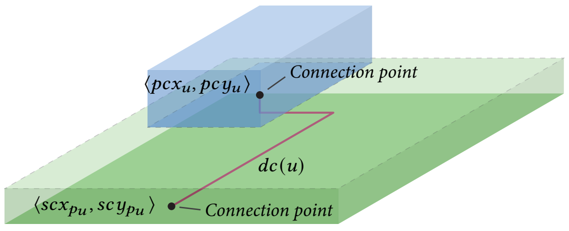
$$\begin{bmatrix} gx & gy & gz \end{bmatrix}^T = \mathbf{T}_p \begin{bmatrix} x & y & pz & 1 \end{bmatrix}^T \tag{14}$$

$$\begin{bmatrix} gl & gw & gh \end{bmatrix}^T = \mathbf{T}_p \begin{bmatrix} pl & pw & ph & 0 \end{bmatrix}^T \tag{15}$$

where $\mathbf{T}_p$ is the transformation matrix of the parent, pl, pw, ph is the dimension, and x, y, pz is the position in the parent frame. We can determine pl, pw, ph based on the rotation flag r of the element. For example, for a component, $pl = l$ if $r = 1$ and $pl = w$ otherwise. Similarly, pz is determined based on the top flag t. The transformation matrix of each PCB is calculated based on the matrix of its parent $\mathbf{T}_p$, its rotation flag r, its position x, y, and its top flag t. Please see Appendix A for complete details.

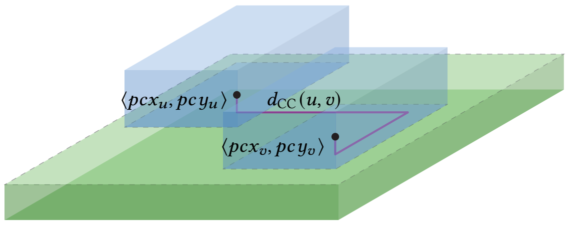
5.2 Wire Length

Every element has a point on the mounting surface where the connection with its parent occurs. We call this point the *connection point*, which can be written in the frame of the parent $pcx, pcy \in \mathbb{R}$ or its own frame $scx, scy \in \mathbb{R}$. This point can be any linear combination of the four corners defining the mounting surface. We choose to use the centroid of the surface without loss of generality.



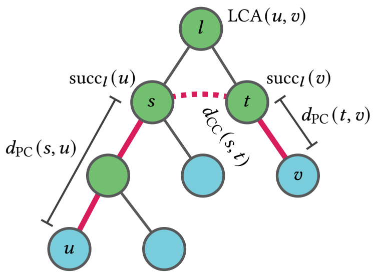
$\langle pcx_u, pcy_u \rangle$
Connection point
$dc(u)$
$\langle scx_{p_u}, scy_{p_u} \rangle$
Connection point

(a) Wire length of an element and its parent



$\langle pcx_u, pcy_u \rangle$
$d_{CC}(u,v)$
$\langle pcx_v, pcy_v \rangle$

(b) Wire length between two elements with the same parent



(c) Wire length between component u and v

Fig. 6. **Wire length.** Diagrams showing the computation of wire length between (a) an element to its parent; (b) two elements on the same PCB; and (c) any two elements.

Let us first consider the case when two elements lie on the same PCB (Figure 6b). In this case, the connection can be made through the shared parent. We define $d_{CC}(u,v)$ as the wire length between element u and v if they share the same parent p. This is the L1 distance between two connection points plus the PCB thickness so the traces lies in the middle of the PCB:

$$d_{CC}(u,v) = |pcx_u - pcx_v| + |pcy_u - pcy_v| + h_p \tag{16}$$

In the general case, we need to compute a path across parent PCBs. Figure 6a shows the wire connection between *element* u and its parent p_u with the trace lies in the middle of the PCB. The wire length between them $dc(u)$ is the L1 distance between the two connection points:

$$dc(u) = |pcx_u - scx_{p_u}| + |pcy_u - scy_{p_u}| + h_{p_u}/2 \tag{17}$$

Since there is a unique path between any two *elements* u and v through their lowest common ancestor (LCA) l, the wire length between them $D(u,v)$ can be written as a sum of the wire length between two adjacent nodes along the unique path except for node l whose two children (s and t) along the path can be connected directly (see Fig. 6c).

To formalize, let $\text{succ}_l(u)$ be the immediate child of l on the unique path between element l and component u if u is in the subtree of l and let $d_{\text{PC}}(s,u)$ be the wire length between element s and component u if u is in the subtree of s. We can write:

$$d_{\text{PC}}(s,u) = \sum_{u'} dc(u') \tag{18}$$

where u' is the element between u and $\text{succ}_s(u)$.

We can now write the wire length between any two elements u and v as:

$$D(u,v) = d_{\text{PC}}(s,u) + d_{\text{CC}}(s,t) + d_{\text{PC}}(t,v) \tag{19}$$

where $s = \text{succ}_l(u)$, $t = \text{succ}_l(v)$, and $l = \text{LCA}(u,v)$.

5.3 Linearization

Formulating this problem as an MILP is challenging because (1) affine transformations requires multiplication of variables which can be non-linear; and (2) the tree structure is not fixed, making the computation of wire length and global positions non-trivial. In this section, we describe how we formulate these variables as linear equations and constraints. We also refer the reader to the MIP linearization guide [FICO 2017] for common operations such as taking absolute, minimization, and disjunction.

While there exists many instances of multiplications in our formulation, they are always in the form: $b_1 \cdot x_1 + \ldots b_K \cdot +x_K$ where x_k are real variables, b_k are boolean variables, and $\sum_k b_k = 1$. In words, the expression takes the value of the x_k whose corresponding b_k is set to true. To linearize, we introduce an auxiliary variable y and impose the following constraints for each k:

$$y \le x_k + M * (1 - b_k)$$
$$y \ge x_k - M * (1 - b_k)$$

where M is a sufficiently large number (also known as the big M). We call this operator a *multiplexer* denoted by $\text{mux}(b_1 \to x_1, \ldots, b_K \to x_K)$. Note that if none of b_k is true, y is unconstrained.

Using the *multiplexer*, we rewrite gx in (14) as:

$$gx = \text{mux}(Txy_p \to x, Txz_p \to y, Tyz_p \to pz) + Tx_p \tag{20}$$

We can rewrite other variables in a similar manner.

In many instances, we refer to some variables of a parent (e.g., l_p and Tyz_p) when the parent of each node is a decision variable. To linearize, we use the *multiplexer* to select the appropriate value based on the parent one-hot variables. For example, the length of the parent of u is expressed as:

$$l_p = \text{mux}(par_u[p_1] \to l_{p_1}, \ldots, par_u[p_P] \to l_{p_P}) \tag{21}$$

where $p_1, \ldots, p_P$ are possible parents of u.

5.3.1 *Wire Length.*

The computation of wire lengths involves multiple non-trivial expressions, specifically, the LCA of two elements l and the successor of l towards component u ($\text{succ}_l(u)$). Let us suppose for now that we know l.

To compute $\text{succ}_l(u)$, we introduce auxiliary variables $\text{succ}_l[u,j] \in \{0,1\}$ which is 1 if and only if $\text{succ}_l(u) = j$ We only consider when $j = u$ and j is a possible descendent of l (i.e., all PCBs with indices greater than l). We can write:

$$\text{succ}_l[u,u] = \text{par}_u[l] \tag{22}$$
$$\text{succ}_l[u,j] = \text{par}_j[l] \wedge \text{in\_subtree}[j,u] \tag{23}$$

where in_subtree$[j,u]$ is an auxiliary boolean variable indicating whether component u is in the subtree of PCB j, which can be written as:

$$\text{in\_subtree}[j,u] = \bigvee_{j'} \text{succ}_j[u,j'] \tag{24}$$

for applicable j' (i.e., $j' = u$ and j' is a possible descendent of j). This recursive expression only refers to PCBs that are descendent of itself, so there is no cyclic dependency.

Let $d_{\text{PC}}[s,u]$ be the linearization of $d_{\text{PC}}(s,u)$. We write:

$$d_{\text{PC}}[s,u] = \begin{cases} dc(u) & \text{succ}_s[u,u] \\ d_{\text{PC}}[j,u] + dc_j & \text{succ}_s[u,j] \text{ for applicable } j \end{cases} \tag{25}$$

To compute $d_{\text{PC}}(s,u)$ where $s = \text{succ}_l(u)$, we introduce auxiliary variables $d'_{\text{PC}}[l,u]$ to represent the expression. This can be written as:

$$d'_{\text{PC}}[l,u] = \begin{cases} 0 & \text{succ}_l[u,u] \\ d_{\text{PC}}[j,u] & \text{succ}_l[u,j] \text{ for applicable } j \end{cases} \tag{26}$$

In the calculation of $d_{\text{CC}}(s,t)$ where $s = \text{succ}_l(u)$ and $t = \text{succ}_l(v)$, we need pcx_s, pcx_t, pcy_s, and pcy_t. Let $pcx'[l,u]$ be an auxiliary variable representing $pcx_{\text{succ}_l(u)}$. We can write

$$pcx'[l,u] = \begin{cases} pcx_u & \text{succ}_l[u,u] \\ pcx_j & \text{succ}_l[u,j] \text{ for applicable } j \end{cases} \tag{27}$$

We define $pcy'[l,u]$ in a similar manner. We can write $d'_{\text{CC}}[l,u,v]$ which is the linearization of $d_{\text{CC}}(s,t)$ as:

$$\begin{aligned} d'_{\text{CC}}[l,u,v] = &\,|pcx'[l,u] - pcx'[l,v]| \\ &+ |pcy'[l,u] - pcy'[l,v]| + h_p \end{aligned} \tag{28}$$

Since the LCA l is not predetermined, we need to iterate through each PCB, compute the wire length assuming it is the LCA, and take the minimum. We can now linearize the wire length between component u and v, $D(u,v)$, as:

$$\begin{aligned} \underset{l}{\text{min\_en}}\{&(\text{in\_subtree}[l,u] \wedge \text{in\_subtree}[l,v]) \\ &\to d'_{\text{PC}}[l,u] + d'_{\text{CC}}[l,u,v] + d'_{\text{PC}}[l,v]\} \end{aligned} \tag{29}$$

where min_en is the operator that takes the minimum of only values whose predicate is true. The predicate ensures that the proceeding variables are well defined. The linearization of the operator is described in Appendix B.

6 Results

We evaluate our method over different server specifications through comparison with flat 2D motherboard designs and expert generated designs. We conduct physical experiments to validate our abstraction of thermal constraints, specifically the gap between components and the orientation of PCBs. Finally, we build a server prototype for one of our optimized server designs with custom heaters designed to simulate real server components to validate the thermal interactions between the fluid, heat source, and the geometry of the server. In addition, we show different standalone designs without tank restriction and show the potential of customizing tank dimension to further improve density.

Our key findings include an optimized 1-socket server design which reduces volume by 31.5% over the 2D baseline (an optimized placement of components that resembles air-cooled designs, with one PCB and all components placed on one side), and the ability to pack 60 units in a single tank, compared to 27 units using the baseline design. We further show in physical tests that this generated design is thermally stable in a real immersion tank maintaining temperatures of 60°C or less for all components over 10 hrs.

6.1 Server Specifications and Experimental Setup

We hand-crafted three server specifications for our evaluation similar to common air cooled configurations (Fig 2) used by major cloud providers. The *1-Socket* specification includes one CPU, 12 DIMMs, 7 SSDs, and one NIC. The *2-Socket* specification consists of two CPUs each connected to 12 DIMMs (24 total) and 4 SSDs (8 total), and one NIC connected to the first CPU. Lastly, the *2-Socket 8-GPU* specification has the same components as the *2-Socket* plus 4 GPUs connected to each CPU (8 total).

Every specification has 4 PCBs unless otherwise specified. We use dimensions of publicly available components on the market. We follow the dimensions of an AMD Genoa package and attached boiler plate (120.3 × 90.4 × 13.0 mm) for CPUs, DDR5 standards (160.0 × 6.0 × 35.0 mm) for DIMMs, M.2 22110 (22.15 × 110.15 × 3.88 mm) for SSDs, Open Compute standards (167.65×68.9×11.5 mm) for NICs [OCP Server Workgroup 2022], and an Nvidia H100 PCIE cards without heat sinks (264.0 × 13.0 × 111.0) for GPUS [TechPowerUp 2023]. We maintain 6 mm gaps between components based on expert recommendations, use 3 mm for PCB thickness (h_p), 153 mm for maximum wire length of CPU-DIMM connections, and 254 mm for CPU-NIC, CPU-SSD, and CPU-GPU connections. For some runs with many server components, we bundle multiple DIMMs together in groups of 2, 4, 6, and 12. All optimization problems were solved using Gurobi 10.0.3 on a machine with a 48-core CPU and 384 GB of RAM.

6.2 Metrics and Baselines

Reducing server volume and increasing density affects multiple metrics including space efficiency, volume of immersion cooling liquid required, datacenter floor space, and subsequent carbon savings that result from improvements to those metrics. We describe each of these in detail below.

Space Efficiency: To standardize the comparison of different designs, we define *Server Space Efficiency* (E_{server}) as the ratio between the theoretical minimum volume (V_{min}) and the actual volume of the server's global bounding box. V_{min} is a lower bound computed as the sum of component volumes, the minimum gaps around components, and half the thickness of the PCB for components with a connecting edge, that is: $\sum_c l_c \cdot w_c \cdot (h_c + h_p/2)$, where c is a component and l_c, w_c, h_c is the padded dimension of c. The *Tank Space Efficiency* (E_{tank}) is defined in terms of V_{lb}, the maximum number of servers that can be packed in a given tank (N), and tank volume (V_{tank}):

$$E_{tank} = \frac{V_{min} N}{V_{tank}} \tag{30}$$

Liquid Volume: The volume occupied by a server design, V_s can be expressed as the sum of component and PCB volumes. The volume of liquid, V_l needed to immerse the servers is calculated as:

$$V_l = V_{tank} - V_s N \tag{31}$$

Carbon emissions: Carbon emissions are typically measured in CO2-equivalents, or CO2e, which normalize emissions to the greenhouse effect of 1 kg of CO2. The emissions of datacenter buildings include carbon embodied in the production of the construction materials—notably steel and cement—the quantities of which have an approximately linear relation to floor space. Space savings are estimated by computing the area used by tanks needed to fit a given number of baseline servers into a datacenter, and re-scaling that area as fewer tanks are needed for denser servers. It is estimated that for a datacenter with 50,000 servers, a 1% reduction in server volume corresponds to the reduction of 17,082 kgCO2e, equivalent to the carbon sequestration of 71 acres of U.S. forests for 1 year [EPA 2024] (see Appendix C for details).

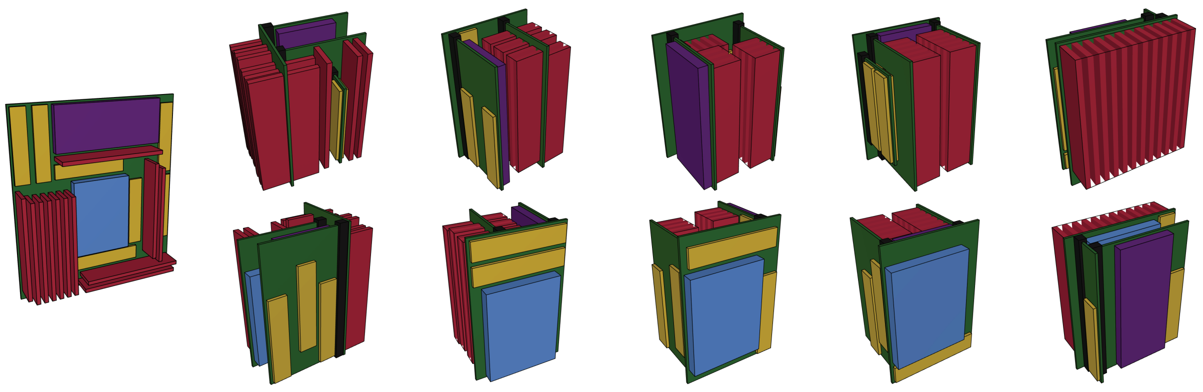
The carbon emissions associated with the immersion fluid include emissions from the manufacturing of fluid and leakage into the atmosphere. We note that the associated carbon reductions are directly proportional to reduction in V_l, although data for the manufacturing kgCO2e/L values is not publicly available. We also note that the global warming potential of some fluids is high if they are released into the atmosphere, e.g., over 9,600 kgCO2e/kg for FC-3284 liquid [Dunham 2013], or 16,400 kgCO2e/L. However, fluid leakage is expected to represent a very small fraction of used liquid, and it is unclear how server design affects the rate at which leakage may occur. Hence, we refrain from reporting any carbon savings associated with immersion fluid.

Baseline. We compare our results to a traditional 2D motherboard design, which serves as most existing prototypes for immersion-cooled servers. To account for our assumptions, our 2D baseline is obtained using the standalone formulation (without the SU constraints) with one PCB and the same set of components which we constrain to be on one side of the PCB. We compute the improvements of the mentioned metrics based on relative savings in volume over the baselines.

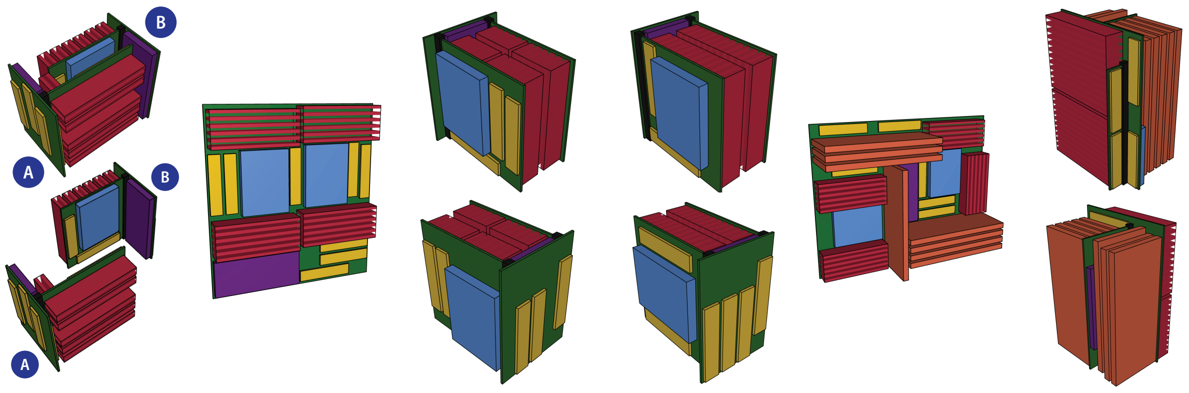
6.3 Standalone Experiment

We first evaluate our standalone formulation where we optimize a set of servers without the SU dimension constraints. Each optimization is run without a time limit and stops after no better solution is found after 6 hours[2]. Figure 7 features the optimized server designs and the baselines of the three specifications, some annotated with their DIMM bundle configurations. Note that the reported 'time to solution' excludes the 6-hour stopping criteria period. We refer to Fig. 18 (left and middle) in the appendix for the optimization curves.

[2]6 hours was used due to time constraints. In general, the design process for a datacenter server is much longer, and we find it is reasonable to allow the algorithm run for several weeks to find denser configurations in practice



	1-Socket Baseline	1-Socket	1-Socket (2-DIMM Bundles)	1-Socket (4-DIMM Bundles)	1-Socket (6-DIMM Bundles)	1-Socket (12-DIMM Bundles)
Volume (L)	3.37	2.54	2.35	2.48	2.34	2.31
Sum Dim. (mm)	598.2	424.7	409.9	415.2	405.8	416.2
Space Eff. (%)	52.7	69.7	75.3	71.4	75.9	76.7
Time to Sol. (hr)	1.7	9.4	0.6	0.0	1.2	0.2



	2-Serv 1-Socket (2-DIMM Bundles)	2-Socket Baseline (6-DIMM Bundles)	2-Socket (6-DIMM Bundles)	2-Socket (12-DIMM Bundles)	2-Socket 8-GPU Baseline (6-DIMM Bundles)	2-Socket 8-GPU (12-DIMM Bundles)
Volume (L)	4.83 (2.42 each)	5.21	3.87	3.82	19.88	9.51
Sum Dim. (mm)	520.2	732.3	476.5	474.0	968.0	654.1
Space Eff. (%)	73.5	59.1	79.5	80.6	39.9	83.5
Time to Sol. (hr)	9.6	0.1	10.5	0.6	3.7	6.1

Fig. 7. **Standalone Results.** Multiple server configurations optimized without SU constraints. The results show substantial space savings can be achieved for diverse configurations within reasonable computation times.

Our algorithm generates significantly denser designs than the baselines. For *1-Socket*, our baseline takes 3.37 L while our 3D designs (without bundles) only takes 2.54 L, a 24.6% reduction in volume. For *2-Socket*, our baseline takes 5.21 L while ours takes 3.87 L (25.7% reduction). Lastly, for *2-Socket 8-GPU*, our baseline takes 19.88 L while our design occupies 9.51 L (52.2% reduction).

We also vary the number of bundled DIMMs for comparison. We notice that bundles with more DIMMs resulted in slightly denser designs and the time to converge typically decreases (see Figure 7 *x-Socket (y-DIMM Bundle)*). This observation can be attributed to the limited time used as the convergence stopping criteria and its disproportional effect on configurations with more components; bundling components reduces the search space and tends to helps the solver find near-optimal configurations faster. In some experiments, we also observe the designs with smaller bundles shares some resemblance to those generated with larger bundles. For example in *1-Socket (2-DIMM Bundles)*, the three 2-DIMM bundles are placed next to one another resembling that of *1-Socket (6-DIMM Bundles)*, and similarly in *2-Socket (6-* and *12-DIMM bundles)*. These results suggests that while bundling artificially constrains the search space and should produce worse optimal solutions, it can be used as a heuristics to reduce the time needed to arrive at approximate solutions.

In addition, we optimize a set of two *1-Socket* servers shown in *2-Serv 1-Socket (2-DIMM Bundles)*. Theoretically, one can expect a better optimal result due to the potential to interleave some components across servers to fill in unused space. Our approach indeed finds an interesting design with two distinct L-shaped servers arranged in the bounding box. However, the average volume is not better that those optimized individually. This might be due to the larger design space, and exploring it would take longer than the time allotted for the experiment.

We note that our standalone formulation optimizes for the sum of dimensions rather than optimizing for density directly due to its non-linearity. Nevertheless, our approach offers significant improvements over the baseline, achieving 70% space efficiency or more for all specifications. Our best result for *1-Socket* takes 2.31 L, which is a 31.5% reduction in volume. At the scale of 50,000 server datacenter described above and in Appendix C, this translates to 538,083 kgCO2e reduction from space savings.

6.3.1 Number of PCBs. While increasing the number PCBs allows for a better theoretical optimal solution, the search becomes slower due to a larger design space and might not find the better solution in reasonable time. We run experiments using 1-Socket specifications with different number of PCBs (2 to 7) without time limit and stop after 6 hours without new solutions. The generated designs are presented in Fig. 19 and the optimization curves are plotted in Fig. 18 (right) in the appendix. In terms of the objective (sum of dimensions), we observe a significant improvement by going from 2 to 3 PCBs and a slight improvement from 3 to 4 PCBs, which are due to the more flexible design space. However, the improvement diminishes beyond 4 PCBs because the optimizations are slower (see the optimization curves) and do not find better solutions within the set stopping criteria. Moreover, we notice that in the configurations with more than 4 PCBs, the optimized designs actually use no more than 4 PCBs. The result either suggests that there are diminishing returns from adding more PCBs or more likely that the MILP approach is no longer effective in handling large number of elements. Therefore, we empirically choose 4 PCBs for all other experiments as it offers a balanced trade-off between the rich design space and search time.

6.4 Tank Packing

We evaluate our end-to-end method with the *1-Socket* specification. We use a tank of dimension 435 × 785 × 482 mm, which is the dimension of a 19-inch deep 17U rack. We run the optimization for every grid partitioning where the SU length and width are at least 70mm. For each optimization, we set the time limit to 6000s and stop if no new solution is found after 2000s. Figure 8 (a) shows the *1-Socket* design optimized for the tank. The design allows us fit 60 total servers within the tank in a regular grid tiling achieving the tank space efficiency of 64.6%.

We also show an experiment optimizing two servers within the SU in Figure 8 (b). The optimized design does not feature special form factor and appears as two cuboid servers stacked together. The average server volume is worse and so is the tank packing, which is the similar behavior found in the standalone experiments.

Figure 8 (c) shows an attempt to fit the best standalone design (*1-Socket (12-DIMM Bundles)*) in a regular grid tiling. While this design takes slightly less volume (2.31 L) than the tank-optimized design in Fig 8 (a) of 2.33 L, it only allows 54 (compared to 60) servers to be packed. Trying to fit other standalone variations of *1-Socket*, not shown here, resulted in even fewer servers packed. These results demonstrate the need for our tank packing algorithm to maximize tank space efficiency.

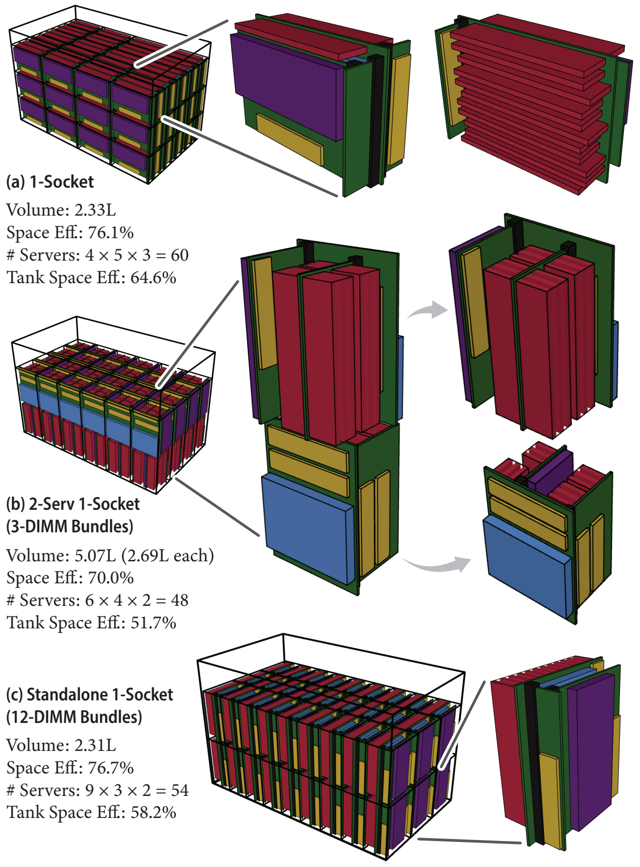
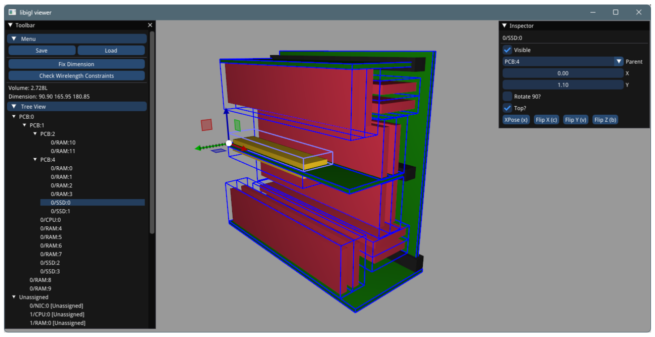

(a) 1-Socket
Volume: 2.33L
Space Eff.: 76.1%
# Servers: 4 × 5 × 3 = 60
Tank Space Eff.: 64.6%
(b) 2-Serv 1-Socket (3-DIMM Bundles)
Volume: 5.07L (2.69L each)
Space Eff.: 70.0%
# Servers: 6 × 4 × 2 = 48
Tank Space Eff.: 51.7%
(c) Standalone 1-Socket (12-DIMM Bundles)
Volume: 2.31L
Space Eff.: 76.7%
# Servers: 9 × 3 × 2 = 54
Tank Space Eff.: 58.2%

Fig. 8. **Tank Packing.** Designs optimized to fit the maximum number of servers in a 19" 17U tank. (a) 1-socket design, (b) Two servers within the SU, (c) Standalone 1-socket server described above tiled in a grid, which achieves lower packing efficiency.

6.5 Expert Comparison

To create a human-generated baseline for dense server designs, we recruited two candidate 'experts' to propose designs. Since server design for immersion cooling is still in its early stages and there are no/few professionals specialized in this area, we recruited experts from related fields: one specializing in the design of small-scale compact electronic devices and the other with extensive experience in 2P immersion fluid and cloud datacenter servers. The experts were provided with a CAD-like user interface (UI) to design the densest *1-Serv 2-Socket* server (Figure 9), and asked to optimize the volume of the global bounding box. The UI restricts to the designs in the space explored by our algorithm and flags element overlaps and wire length constraint violations to assist the user in creating a feasible design.



```
Volume: 2.728L
Dimension: 90.90 165.95 180.85

PCB:0
  PCB:1
    PCB:2
      0/RAM:10
      0/RAM:11
    PCB:4
      0/RAM:0
      0/RAM:1
      0/RAM:2
      0/RAM:3
      0/SSD:0
      0/SSD:1
    0/CPU:0
    0/RAM:4
    0/RAM:5
    0/RAM:6
    0/RAM:7
    0/SSD:2
    0/SSD:3
  0/RAM:8
  0/RAM:9
Unassigned
  0/NIC:0 [Unassigned]
  1/CPU:0 [Unassigned]
  1/RAM:0 [Unassigned]

0/SSD:0
[✓] Visible
PCB:4        Parent
0.00         X
1.10         Y
[ ] Rotate 90?
[✓] Top?
```

Fig. 9. **Expert comparison UI.** CAD-like UI used for user study with expert designers that enforces server design constraints.

The expert-generated designs are featured in Figure 10. The reported time spent on the task is 1.5–3 hours over several sessions. The first design (a) has the volume of 4.97 L, and another design (b) is meant to stacked with interlocking DIMMs. Considering the overlap, the average volume is estimated to be 4.99 L. Both designs have a higher server space efficiency than the 2-Socket 2D baseline (5.21 L), but lower than the design generated by the optimization algorithm (3.82 L). This result, although not conclusive, suggests that a computational design tool based on our algorithm can help designers discover denser server configurations. To manage complexity, we focused our experiments solely on the standalone formulation rather than on full tank packing which adds an additional dimension to the design problem.

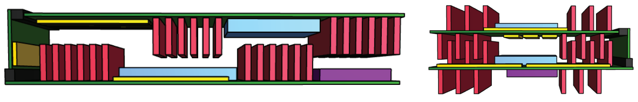


Fig. 10. **Human expert designs.** 1-Serv 2-Socket server designs generated by human experts. Both designs require higher volume than our optimized result.

6.6 Validation of Thermal Constraints

We also conduct physical experiments to validate expert's suggestions about the gap between components and the orientation of components. Existing studies on this topic [Liu and Yu 2021] use CFD simulations with power draws of 500–3000 W; our study complements this analysis with datapoints from physical experiments at 200 W, within the range of power drawn by currently-deployed server CPUs. First, we study the effect of the gap between two components on the temperature at stable state. Figure 11 (a) shows the test setup with two heater blocks facing each other with variable gap and heaters below to generate ambient heat and bubbles. Each heater, modeling a CPU, is an aluminum block of size $62 \times 13 \times 77$ mm inserted with a thermocouple and four cartridge heaters set to dissipate a total of 200 W. The ambient heater is set to dissipate 200 W.

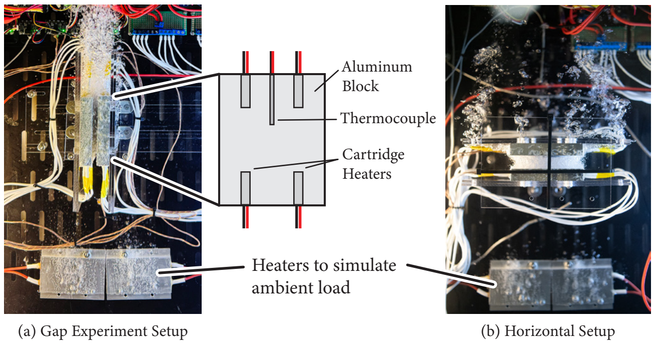


Fig. 11. **Heater configuration tests.** (a) Two CPU heaters composed of aluminum blocks with heaters and thermocouples used to evaluate the minimum gap required between components. (b) Tests to evaluate fluid flow in horizontal configuration.

We run the experiment with gap values of 0, 1, 5, 6, and 10mm. Figure 12 (inset) shows the temperature of the heaters and the fluid over time with a gap of 6 mm until it stabilizes, and (main) shows the stable temperatures achieved at different gaps. The results shows that with a gap, regardless of how small, the temperature stabilizes at around 55°C whereas if there is no gap (the two heater blocks touch), we see an increase in the stable temperature at around 60°C. We hypothesize that this is due to the reduction in the surface area. Note that we did not thermally insulate the sides and the back of the aluminum blocks, so the heat dissipated on the two facing surfaces are effectively less than 200W.

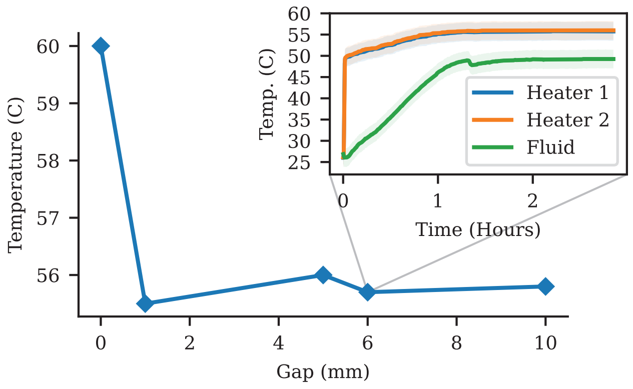


Fig. 12. **Heater gap evaluation.** (main) Stable temperatures achieved when varying heater distances showing gaps as small as 1 mm can maintain safe temperatures. (inset) Temperature stabilization data over 3 hrs for a gap of 6 mm for 2x 200 W CPUs.

In addition, we run the experiment where the two heaters are oriented horizontally to study the effect of impeded vapor path (see Figure 11 (b)). Here, we block the front, back, top, and bottom so that the majority of vapor has to travel some distance to the left and right of the cavity. The result shows that the temperature stabilizes at 57°C, which is slightly but not significantly higher than with vertical orientation. While further experiments using higher power and well insulated heaters are required to fully characterize the effect of component placements and orientations on cooling performance, our tool allows for customizing constraints with updated data.

6.7 Physical Validation

In addition to generating server designs, we go a step further to physically validate their thermal behavior to confirm such dense servers can operate in a real immersion cooling tank. To do this we build a physical prototype of the server using heaters which allow to simulate components with precise control and performing end-to-end experiments in an immersion cooling tank as shown in Figure 13 (LiquidStack DataTank 2U). The tank is filled with 3M FC-3284 fluid [3M2 2019], which has a boiling point of 50°C and density of 1.71 kg/L. Our goal is to show that with our optimized dense design, the temperature of each component remains stable within a safe range over an extended period of time.

We run the 1-Serv configuration with the tank dimension 519.75× 80.0×800.0 mm. Figure 1 shows the optimized design running in the immersion tank during which the temperature stabilizes. Our optimized design occupies 2.51 L, a 25.5% reduction in volume from the 2D baseline. At the scale of 50,000 server datacenter described above and in Appendix C, this translates to 435,591 kgCO2e reduction from space savings.

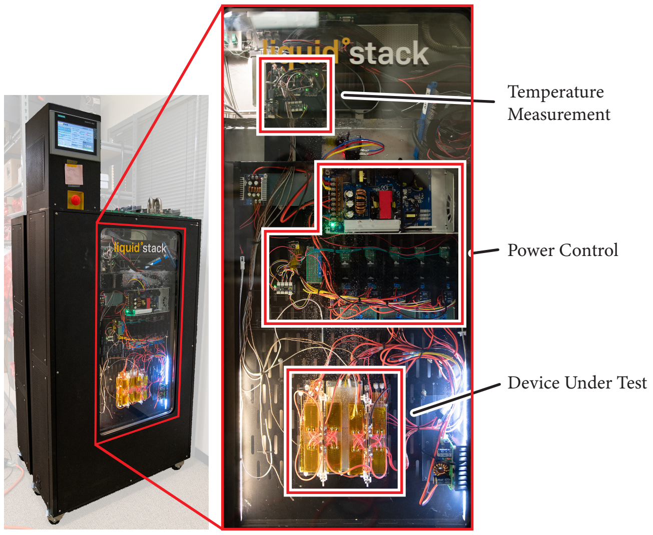


Fig. 13. **End-to-end Physical test.** Test setup for dense server prototype comprised of heaters with the same power and thermal profiles as real server components (Device Under Test). The prototype is placed in an immersion cooling tank with custom power control and sensing circuits.

We fabricate the prototype using laser-cut acrylic sheets to model PCBs and the structure of the components. We use resistive heaters of varying sizes to precisely control and simulate heat dissipation of components in different arrangements. These include arrays of thin film polyimide heaters to model DIMMs and aluminum blocks with variable power heaters to simulate CPUs, NICs, and SSDs.

Figure 13 shows the tank and our test setup. We design a custom power distribution system with a PID controller that delivers a constant 200 W for the CPU, 10 W for each DIMM (120 W total), 15W for each SSD (105 W total), and 80 W for the NIC, totaling 505 W. We verify the controller produces stable power outputs and that the heaters produce temperatures above the safe levels when uncooled in air. We attach a thermocouple to one component of each type and another to measure the fluid's temperature.

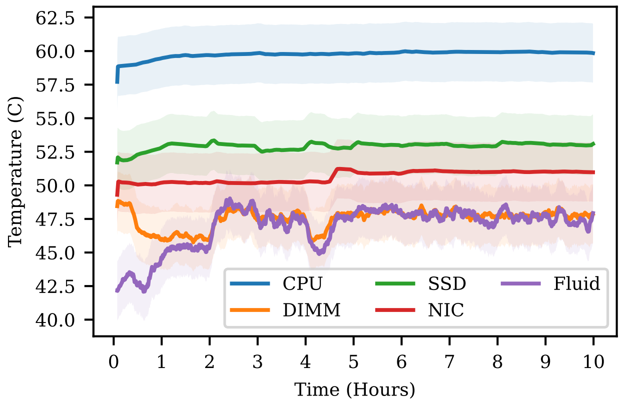


Fig. 14. **Temperature stabilization.** Ten hour experiment of the full server in the immersion tank above showing all components stabilize at safe temperatures ranges with small fluctuations.

Figure 14 shows the temperature of each component and the fluid over 10 hours. The temperature readings successfully stabilize at 59.8, 47.9, 53.1, 51.0, and 47.9°C for the CPU, DIMM, SSD, NIC, and fluid, respectively. We note that due to the limitation of our measurement equipment, the reported temperatures are within ±2.2°C of the actual values [Thermocoupleinfo 2024]. The result suggests that our generated server design can operate at safe and stable temperatures during long term use.

7 Limitations and Future Work

In presenting the first computational solution for designing servers for immersion cooling, our work opens up numerous avenues for future research. We describe limitations of our current tool and future directions below.

Scalability. The complexity of the design problem scales exponentially with the number of server components. Figure 15 plots the time it takes to find a solution with specific space efficiency across different number of elements. Empirically, the problem becomes difficult to solve within a reasonable time beyond 35 elements. Another problem with 90 elements, not shown on the curve, ran for over 10 days to obtain a solution with 20% space efficiency.

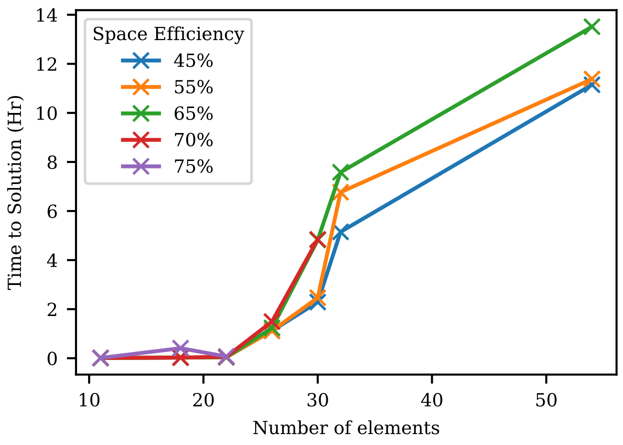


Fig. 15. **Computation time.** The time required to generate designs versus number of server components and space efficiency.

In our experiments, this limitation was mitigated by defining *bundles* which we hand-picked to effectively reduced the problem size. Future work could pursue the problem of selecting bundled components. Furthermore, tackling our problem with genetic algorithm, evolutionary algorithms, or learning-based methods, as with other exponentially scaling problems, is another interesting direction for future work.

Support for Multiple Objectives. Our work primarily focuses on optimizing density; however, there are other considerations that one might be willing to trade-off for density. For example, a server with easily accessible components is preferable to a slightly denser server where components have to be assembled in a specific order. Modeling considerations as a multi-objective optimization problem would lead to better designs overall.

Extending Design Space. Our current design space abstracts away many considerations. Modeling other small server components such as voltage regulators and capacitors is a direct extension as the algorithm scales to more elements. Further engineering effort to model other cooling technologies such as single-phase immersion, cold plates, and microfluidics [Van Erp et al. 2020] would be valuable in extending the scope of this work. Moreover, considering non-traditional PCBs, such as flexible PCBs or additive PCB manufacturing process is valuable in breaking the convention of what servers should look like.

8 Discussion and Conclusion

In this work, we propose the first computational approach to optimize server design for immersion cooling using a novel MILP formulation that achieves promising results. Our solution incorporates domain-specific considerations to define constraints on the layout while maintaining the flexibility to support a variety of server specifications. The MILP formulation can also be adapted to different thermal considerations, such as component orientations and gaps between parts, and can be extended to incorporate additional constraints.

To the best of our knowledge, ours is also the first public research that uses physical prototypes to test the effects of component distance on thermal performance. Our experiments, conducted with simple resistive heaters, are easily replicable, which we believe can lead to further research on the impact of distance and orientation to build deeper understanding of the interplay between cooling and design.

A fundamental contribution of our work is to open a new path towards reducing carbon emissions of datacenters. By demonstrating the potential gains from optimizing server design, our research encourages further research and funding for dense sustainable datacenters.

Importantly, the study makes several assumptions about carbon impact that must be verified to ensure environmental benefits are realized. One significant concern is the environmental impact of fabricating liquids for immersion cooling, including the carbon footprint of their production and the toxicity issues related to their safe disposal. While ongoing research aims to develop improved liquids for these applications, this work is still in progress. The choice between air cooling, 1P or 2P immersion cooling, and other technologies will depend on these developments.

It is important to note that, as with any other work that improves efficiency, our work can incentivize the use of more datacenters—an effect known as the Jevons Paradox. It is crucial to consider the trade-off between environmental impact and the value that datacenters can offer. We acknowledge that characterizing this trade-off is an intricate topic and further note the importance of considering the potential for misuse.

Fundamentally, efficient server packing can be studied for various cooling techniques. The flexibility of our tool to encode a variety of constraints allows this framework to be extended to future cooling solutions as this field continues to evolve. We hope this work will inspire further research into dense server design for a variety of cooling technologies and encourage continued exploration in this area.

Acknowledgments

This work was funded by NSF grants 2212049. We thank Yuxuan Mei, Benjamin Jones, James Noeckel, Amy Zhu, and Jerry Cao for their contributions during ideation. We would like to thank Devon Starks and Washington Kim for their assistance in operating immersion tanks. Special thanks to Teerapat Jenrungrot and Varich Boonsanong for their help in assembling the prototypes.

References

2019. 3M Fluorinert Electronic Liquid FC-3284. https://multimedia.3m.com/mws/media/64887O/3m-fluorinert-electronic-liquid-fc3284-en.pdf. (2019).

3M. 2023. Microporous Metallic Boiling Enhancement Coating. https://multimedia.3m.com/mws/media/563566O/3mtm-microporous-metallic-boiling-enhancement-coating-l-20227.pdf. (2023).

C. Ababei, H. Mogal, and K. Bazargan. 2005. Three-dimensional place and route for FPGAs. In *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, Vol. 2. 773–778 Vol. 2. https://doi.org/10.1109/ASPDAC.2005.1466456

Amazon. 2021. Energy Transition. https://aws.amazon.com/energy/sustainability/. (Accessed on 05/10/2024).

Xudong An, Manish Arora, Wei Huang, William C Brantley, and Joseph L Greathouse. 2018. 3D numerical analysis of two-phase immersion cooling for electronic components. In *2018 17th IEEE intersociety conference on thermal and thermomechanical phenomena in electronic systems (ITherm)*. IEEE, 609–614.

Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. 2020. Mixed integer ink selection for spectral reproduction. *ACM Trans. Graph.* 39, 6, Article 255 (nov 2020), 16 pages. https://doi.org/10.1145/3414685.3417761

Javier Avalos, Oscar Del Rio, and Oscar Farias. 2022. Air Cooling Server Conversion to Two Phase Immersion Cooling and Thermal Performance Results. In *2022 38th Semiconductor Thermal Measurement, Modeling & Management Symposium (SEMI-THERM)*. IEEE, 55–61.

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Luiz André Barroso, U Holzle, and P Ranganathan. 2018. The datacenter as a computer. *Morgan Claypool 2013* (2018).

C.S. Chen, S.M. Lee, and Q.S. Shen. 1995. An analytical model for the container loading problem. *European Journal of Operational Research* 80, 1 (1995), 68–76. https://doi.org/10.1016/0377-2217(94)00002-T

J. Cong, Jie Wei, and Yan Zhang. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* 306–313. https://doi.org/10.1109/ICCAD.2004.1382591

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. 2008. Extreme Point-Based Heuristics for Three-Dimensional Bin Packing. *INFORMS Journal on Computing* 20, 3 (2008), 368–384. https://doi.org/10.1287/ijoc.1070.0250 arXiv:https://doi.org/10.1287/ijoc.1070.0250

Qiaodong Cui, Victor Rong, Desai Chen, and Wojciech Matusik. 2023. Dense, Interlocking-Free and Scalable Spectral Packing of Generic 3D Objects. *ACM Trans. Graph.* 42, 4, Article 141 (jul 2023), 14 pages. https://doi.org/10.1145/3592126

Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational multicopter design. (2016).

Sarah Dunham. 2013. Mandatory Reporting of Greenhouse Gases: Notice of Data Availability Regarding Global Warming Potential Values for Certain Fluorinated Greenhouse Gases and Fluorinated Heat Transfer Fluids. https://www.federalregister.gov/d/2013-07977.

Lesya Dymyd, Leonard Ciubotaru, MichaelHelezen, Jimil M. Shah, Lentis Pai, Rolf Brink, Rick Payne, Jessica Gullbrand, and Nigel Gore. 2020. Design Guidelines for Immersion-Cooled IT Equipment. https://www.opencompute.org/documents/design-guidelines-for-immersion-cooled-it-equipment-revision-1-01-pdf.

Michael Eley. 2005. *A bottleneck assignment approach to the multiple container loading problem.* Springer Berlin Heidelberg, Berlin, Heidelberg, 359–374. https://doi.org/10.1007/3-540-26686-0_16

EPA. 2024. Greenhouse Gases Equivalencies Calculator - Calculations and References. https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references. (Accessed on 05/19/2024).

ExxonMobil. 2024. ExxonMobil and Intel announce collaboration to bring data center immersion cooling solution to market. https://www.exxonmobilchemical.com/en/resources/library/library-detail/111994/collaboration_to_bring_data_center.

Yuehong Fan, Casey Winkel, Devdatta Kulkarni, and Wenbin Tian. 2018. Analytical design methodology for liquid based cooling solution for high tdp cpus. In *2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*. IEEE, 582–586.

Oluf Faroe, David Pisinger, and Martin Zachariasen. 2003. Guided Local Search for the Three-Dimensional Bin-Packing Problem. *INFORMS Journal on Computing* 15, 3 (2003), 267–283. https://doi.org/10.1287/ijoc.15.3.267.16080

FICO. 2017. MIP Formulations Quick Reference. https://www.fico.com/fico-xpress-optimization/docs/latest/mipform/dhtml/GUID-949C11F8-99C3-33A2-BD4E-3AB742A1B3E4.html Accessed on December 1, 2023.

Eitan Frachtenberg, Ali Heydari, Harry Li, Amir Michael, Jacob Na, Avery Nisbet, and Pierluigi Sarti. 2011. High-efficiency server design. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis* (Seattle, Washington) *(SC '11)*. Association for Computing Machinery, New York, NY, USA, Article 27, 27 pages. https://doi.org/10.1145/2063384.2063420

Matthew Gooding. 2024. US Data Center Power Consumption. https://www.datacenterdynamics.com/en/news/us-data-center-power-consumption/.

Google. 2021. 24/7 Clean Energy – Data Centers. https://www.google.com/about/datacenters/cleanenergy/. (Accessed on 05/10/2024).

Christian Hafner and Bernd Bickel. 2021. The design space of plane elastic curves. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.

Gary Hilson. 2024. AI-Driven HBM Uptake Is Power-Sensitive. https://www.eetimes.com/ai-driven-hbm-uptake-is-power-sensitive/.

Haoyuan Hu, Xiaodong Zhang, Xiaowei Yan, Longfei Wang, and Yinghui Xu. 2017. Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method. *ArXiv* abs/1708.05930 (2017). https://api.semanticscholar.org/CorpusID:23532462

Ruizhen Hu, Juzhan Xu, Bin Chen, Minglun Gong, Hao Zhang, and Hui Huang. 2020. TAP-Net: transport-and-pack using reinforcement learning. *ACM Trans. Graph.* 39, 6, Article 232 (nov 2020), 15 pages. https://doi.org/10.1145/3414685.3417796

Fei Huang, Chen Liu, Kai-Wen Hsiao, Ying-Miao Kuo, Hung-Kuo Chu, and Yong-Liang Yang. 2023. Image-Based OA-Style Paper Pop-Up Design via Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics* 29, 10 (2023), 4269–4283. https://doi.org/10.1109/TVCG.2022.3189569

Thermal Technical Working Group IEEE-HIR. 2023. Heterogeneous Integration Roadmap, Chapter 20: Thermal. https://eps.ieee.org/images/files/HIR_2023/ch20_thermalfinal.pdf. (2023).

Intel. 2023. Safe Operating and Storage Temperature of Intel® Xeon® Gold 6254 Processor. https://www.intel.com/content/www/us/en/support/articles/000089129/processors.html. (Accessed on 05/19/2024).

Majid Jalili, Ioannis Manousakis, Íñigo Goiri, Pulkit A Misra, Ashish Raniwala, Husam Alissa, Bharath Ramakrishnan, Phillip Tuma, Christian Belady, Marcus Fontoura, et al. 2021. Cost-efficient overclocking in immersion-cooled datacenters. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 623–636.

Lucas Joppa. 2021. Made to measure: Sustainability commitment progress and updates. https://blogs.microsoft.com/blog/2021/07/14/made-to-measure-sustainability-commitment-progress-and-updates/. (Accessed on 05/10/2024).

Laila Kearney, Seher Dareen, , and Deep Kaushik Vakil. 2024. US electric utilities brace for surge in power demand from data centers. https://www.reuters.com/business/energy/us-electric-utilities-brace-surge-power-demand-data-centers-2024-04-10/.

Ali C Kheirabadi and Dominic Groulx. 2016. Cooling of server electronics: A design review of existing technology. *Applied Thermal Engineering* 105 (2016), 622–638.

Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019. Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 59–70.

Milin Kodnongbua, Ian Good Yu Lou, Jeffrey Lipton, and Adriana Schulz. 2023. Computational design of passive grippers. *arXiv preprint arXiv:2306.03174* (2023).

Carlos Lamas-Fernandez, Julia A. Bennell, and Antonio Martinez-Sykora. 2023. Voxel-Based Solution Approaches to the Three-Dimensional Irregular Packing Problem. *Operations Research* 71, 4 (2023), 1298–1317. https://doi.org/10.1287/opre.2022.2260 arXiv:https://doi.org/10.1287/opre.2022.2260

Naushad Manzoor Laskar, Rahul kumar Sen, Prashanta Kumar Paul, and Krishna Lal Baishnab. 2015. A survey on VLSI Floorplanning: Its representation and modern approaches of optimization. *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (2015), 1–9. https://api.semanticscholar.org/CorpusID:16811830

Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu. 2015. Adaptive-latency DRAM: Optimizing DRAM timing for the common-case. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 489–501. https://doi.org/10.1109/HPCA.2015.7056057

Cheng Liu and Hang Yu. 2021. Evaluation and Optimization of a Two-Phase Liquid-Immersion Cooling System for Data Centers. *Energies* 14, 5 (2021). https://doi.org/10.3390/en14051395

Xiao Liu, Jia min Liu, An xi Cao, and Zhuang le Yao. 2015. HAPE3D—a new constructive algorithm for the 3D irregular packing problem. *Frontiers of Information Technology & Electronic Engineering* 16, 5 (2015), 380–390. https://doi.org/10.1631/FITEE.1400421

Andrea Lodi, Silvano Martello, and Michele Monaci. 2002. Two-dimensional packing problems: A survey. *Eur. J. Oper. Res.* 141 (2002), 241–252. https://api.semanticscholar.org/CorpusID:3240079

Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-last: Strength to weight 3D printed objects. *ACM Transactions on Graphics (ToG)* 33, 4 (2014), 1–10.

Jialun Lyu, Marisa You, Celine Irvene, Mark Jung, Tyler Narmore, Jacob Shapiro, Luke Marshall, Savyasachi Samal, Ioannis Manousakis, Lisa Hsu, Preetha Subbarayalu, Ashish Raniwala, Brijesh Warrier, Ricardo Bianchini, Bianca Schroeder, and Daniel S. Berger. 2023. Hyrax: Fail-in-Place Server Operation in Cloud Platforms. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, Boston, MA, 287–304. https://www.usenix.org/conference/osdi23/presentation/lyu

Silvano Martello, David Pisinger, and Daniele Vigo. 1998. The Three-Dimensional Bin Packing Problem. *Operations Research* 48 (02 1998). https://doi.org/10.1287/opre.48.2.256.12386

Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. 2020. Recalibrating global data center energy-use estimates. *Science* 367, 6481 (2020), 984–986.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, Will Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. 2021. A graph placement methodology for fast chip design. *Nature* 594 (2021), 207 – 212. https://api.semanticscholar.org/CorpusID:235395490

Rahul Mitra, Liane Makatura, Emily Whiting, and Edward Chien. 2023. Helix-Free Stripes for Knit Graph Design. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 75, 9 pages. https://doi.org/10.1145/3588432.3591564

Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James Mccann. 2018. Automatic machine knitting of 3D meshes. *ACM Transactions on Graphics (TOG)* 37, 3 (2018), 1–15.

Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman P Jouppi, and David A Patterson. 2020. Google's Training Chips Revealed: TPUv2 and TPUv3.. In *Hot Chips Symposium*. 1–70.

OCP NIC Workgroup OCP Server Workgroup. 2022. OCP NIC Specification 3.0. https://www.opencompute.org/wiki/Server/NIC. (Accessed on 05/19/2024).

Karl Otness, Arvi Gjoka, Joan Bruna, Daniele Panozzo, Benjamin Peherstorfer, Teseo Schneider, and Denis Zorin. 2021. An extensible benchmark suite for learning to simulate physical systems. *arXiv preprint arXiv:2108.07799* (2021).

Nico Pietroni, Corentin Dumery, Raphael Falque, Mark Liu, Teresa A Vidal-Calleja, and Olga Sorkine-Hornung. 2022. Computational pattern making from 3D garment models. *ACM Trans. Graph.* 41, 4 (2022), 157–1.

Michal Piovarči, Danny M Kaufman, David IW Levin, and Piotr Didyk. 2020. Fabrication-in-the-loop co-optimization of surfaces and styli for drawing haptics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 116–1.

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand: balancing shapes for 3D fabrication. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Bharath Ramakrishnan, Husam Alissa, Ioannis Manousakis, Robert Lankston, Ricardo Bianchini, Washington Kim, Rich Baca, Pulkit A Misra, Inigo Goiri, Majid Jalili, et al. 2021. CPU overclocking: A performance assessment of air, cold plates, and two-phase immersion cooling. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 11, 10 (2021), 1703–1715.

D. Gracia Nirmala Rani, S. Rajaram, and Athira Sudarasan. 2013. A novel 3D algorithm for VLSI floorplanning. In *International Conference on Communication and Electronics System Design*, Vijay Janyani, M. Salim, and K. K. Sharma (Eds.), Vol. 8760. International Society for Optics and Photonics, SPIE, 87601S. https://doi.org/10.1117/12.2012334

Gamal Refai-Ahmed, Hoa Do, Yaser Hadad, Srikanth Rangarajan, Bahgat G Sammakia, Vadim Gektin, and Tahir Cader. 2020. Establishing thermal air-cooled limit for High Performance Electronics Devices. In *2020 IEEE 22nd Electronics Packaging Technology Conference (EPTC)*. IEEE, 347–354.

Yingying Ren, Uday Kusupati, Julian Panetta, Florin Isvoranu, Davide Pellis, Tian Chen, and Mark Pauly. 2022. Umbrella meshes: elastic mechanisms for freeform shape deployment. *ACM Transactions on Graphics* 41, 4 (2022), 1–15.

Assel Sakanova, Sajad Alimohammadi, Jaakko Mcevoy, Sara Battaglioli, and Tim Persoons. 2019. Multi-objective layout optimization of a generic hybrid-cooled data centre blade server. *Applied Thermal Engineering* 156 (04 2019). https://doi.org/10.1016/j.applthermaleng.2019.04.071

Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive design space exploration and optimization for CAD models. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14.

Mark Shaw. 2016. Cloud Server Blade 2.0. http://files.opencompute.org/oc/public.php?service=files&t=62279808dbea0cf380632c3042246979&download&path=/V2/OCS_Open_CloudServer_Blade_v2.0.pdf.

Mark Shaw. 2024. Open Rack V3 Base Specification. https://www.opencompute.org/wiki/Open_Rack/SpecsAndDesigns.

Mark Shaw and Martin Goldstein. 2014. Open Compute: Cloud Server Blade 1.0. http://files.opencompute.org/oc/public.php?service=files&t=62279808dbea0cf380632c3042246979&download&path=/V1/Open_CloudServer_Blade_v1.0.pdf.

Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab forms: Customizable objects for fabrication with validity and geometry caching. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational design of actuated deformable characters. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli. 2019. Summarizing CPU and GPU design trends with product data. *arXiv preprint arXiv:1911.11313* (2019).

TechPowerUp. 2023. Nvidia H100 PCIE 80GB. https://www.techpowerup.com/gpu-specs/h100-pcie-80-gb.c3899. (Accessed on 05/19/2024).

Thermocoupleinfo. 2024. Thermocouple Accuracies. https://www.thermocoupleinfo.com/thermocouple-accuracies.htm. (Accessed on 05/19/2024).

Yunsheng Tian, Pavle Vanja Konakovic, Beichen Li, Ane Zuniga, Michael Foshey, Timothy Erps, Wojciech Matusik, and Mina Konakovic Lukovic. 2023. AutODEx: Automated Optimal Design of Experiments Platform with Data-and Time-Efficient Multi-Objective Optimization. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World.*

Nobuyuki Umetani and Bernd Bickel. 2018. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

Remco Van Erp, Reza Soleimanzadeh, Luca Nela, Georgios Kampitsis, and Elison Matioli. 2020. Co-designing electronics with microfluidics for more sustainable cooling. *Nature* 585, 7824 (2020), 211–216.

Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of self-supporting surfaces. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Jaylen Wang, Daniel S. Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, Jonathan Stern, Ricardo Bianchini, and Akshitha Sriraman. 2024. Designing Cloud Servers for Lower Carbon. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*.

Ziqi Wang, Florian Kennel-Maushart, Yijiang Huang, Bernhard Thomaszewski, and Stelian Coros. 2023. A Temporal Coherent Topology Optimization Approach for Assembly Planning of Bespoke Frame Structures. *ACM Trans. Graph.* 42, 4, Article 144 (jul 2023), 13 pages. https://doi.org/10.1145/3592102

Yong Wu, Wenkai Li, Mark Goh, and Robert de Souza. 2010. Three-dimensional bin packing problem with variable bin height. *Eur. J. Oper. Res.* 202 (2010), 347–355. https://api.semanticscholar.org/CorpusID:12413854

Ning Ye, Yangming Liu, Zhongli Ji, Dmitry Vaysman, In-Soo Yoon, and Hem Takiar. 2018. Thermal Challenges and Solutions of M.2 Solid State Drive. In *2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*. 979–983. https://doi.org/10.1109/ITHERM.2018.8419478

Ran Zhang, Thomas Auzinger, and Bernd Bickel. 2021. Computational design of planar multistable compliant structures. *ACM Transactions on Graphics (TOG)* 40, 5 (2021), 1–16.

Shiwei Zhang, Xingchi Jiang, Yuanjie Li, Gong Chen, Yalong Sun, Yong Tang, and Chin Pan. 2020. Extraordinary boiling enhancement through micro-chimney effects in gradient porous micromeshes for high-power applications. *Energy conversion and management* 209 (2020), 112665.

Hang Zhao and Kai Xu. 2022. Learning Efficient Online 3D Bin Packing on Packing Configuration Trees. In *International Conference on Learning Representations*. https://openreview.net/forum?id=bfuGjlCwAq

Xiaozhou Zhao, Julia A. Bennell, Tolga Bektaş, and Kath Dowsland. 2016. A comparative review of 3D container loading algorithms. *International Transactions in Operational Research* 23, 1-2 (2016), 287–320. https://doi.org/10.1111/itor.12094 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12094

Yangfan Zhong. 2019. A large scale deployment experience using immersion cooling in datacenters. *Alibaba Group: Open Compute Project Summit* (2019).

Mingjun Zhou, Jiahao Ge, Hao Xu, and Chi-Wing Fu. 2023. Computational Design of LEGO® Sketch Art. *ACM Trans. Graph.* 42, 6, Article 201 (dec 2023), 15 pages. https://doi.org/10.1145/3618306

A Actualization to 3D Geometry

In this section, we describe in full details the derivation of the global position gx, gy, gz and dimension gl, gw, gh.

Suppose we know the transformation matrix $\mathbf{T}$ for each PCB (13). We can compute global position and dimension using (14) and (15) based on the transformation of its parent $\mathbf{T}_p$, the position (x, y, pz), and dimension (pl, pw, ph) in the parent frame.

For PCBs, the dimension in the parent frame (pl, pw, ph) can be written as:

$$\begin{bmatrix} pl \\ pw \\ ph \end{bmatrix} = \begin{bmatrix} w & h \\ h & l \\ l & w \end{bmatrix} \begin{bmatrix} r \\ 1-r \end{bmatrix} \tag{32}$$

For components, we write:

$$\begin{bmatrix} pl \\ pw \\ ph \end{bmatrix} = \begin{bmatrix} l & w \\ w & l \\ h & h \end{bmatrix} \begin{bmatrix} r \\ 1-r \end{bmatrix} \tag{33}$$

The z-coordinate in parent frame pz depends on the top flag t. This is the thickness of the parent PCB if it is on top or negative its own height otherwise:

$$pz = t \cdot h_p + (1 - t) \cdot -ph \tag{34}$$

We now describe the derivation of variables necessary to compute the transformation matrix, namely, the global orientation Txy, Txz, Tyz and global translation Tx, Ty, Tz of a PCB coordinate frame. For the root PCB, we can write $Txy, Txz, Tyz = rxy, rxz, ryz$ and $Tx, Ty, Tz = rx, ry, rz$.

For non-root PCBs, the orientation undergoes the following transformation that depends on the rotation flag r:

$$\begin{bmatrix} Txz \\ Tyz \end{bmatrix} = \begin{bmatrix} r & 0 & 1-r \\ 1-r & r & 0 \end{bmatrix} \begin{bmatrix} Txy_p \\ Txz_p \\ Tyz_p \end{bmatrix} \tag{35}$$

and $Txy = 1 - Txz_p - Tyz_p$. The global translation of a PCB frame can be written as:

$$\begin{bmatrix} Tx & Ty & Tz \end{bmatrix}^T = \mathbf{T}_p \begin{bmatrix} x & y & pz & 1 \end{bmatrix}^T \tag{36}$$

We have now provided necessary details to convert the tree representation into the 3D geometric representation.

B The min_en Operator

For the expression of the form:

$$\underset{x}{\text{min\_en}}\{pred(x) \rightarrow expr(x)\} \tag{37}$$

where $x \in \{x_1, \dots, x_N\}$, the min_en operator takes the minimum of only values ($expr(x)$) whose predicate ($pred(x)$) is true. To linearize, we introduce an auxiliary variable y to capture the results and $N + 1$ variables $d_i \in \{0, 1\}$, $1 \leq i \leq N + 1$ to indicate the argmin. The extra variable d_{N+1} is set when none of the predicate is true. We impose the following constraints for each i, $1 \leq i \leq n$:

$$y \leq expr(x_i) + M \cdot (1 - pred(i)) \tag{38}$$
$$y \geq expr(x_i) - M \cdot (1 - d_i) \tag{39}$$
$$d_i \leq pred(i) \tag{40}$$

We constrain $\sum_{i=1}^{N+1} d_i = 1$ and $y = Q$ if $d_{N+1} = 1$ (i.e., none of predicate is true) where Q is a large number not larger than M. We impose:

$$y \geq Q - M \cdot (1 - d_{N+1}) \tag{41}$$
$$y \leq Q + M \cdot (1 - d_{N+1}) \tag{42}$$

In our work, we set $Q = M = 100,000$ since $expr(x) \geq 0$ in our case (the wire length).

C Datacenter floorspace

There are no public reference values for datacenter floorspace for immersion cooling deployments that could be used to estimate the magnitude of carbon savings from reducing datacenter floorspace. Hence, we provide a bottom-up estimation using the space occupied by our 2D 2-socket baseline servers, noting that this underestimates the actual space required when all server components are accounted for. We assume that tanks and access aisles account for 80% of floorspace for a datacenter that hosts 50,000 servers. Tank dimensions are approximated based on published images of tank prototypes for datacenter settings [Jalili et al. 2021]

We assume that a single tank can hold 42 baseline servers, requiring 1,190 racks for 50,000 servers. Within a rack we assume servers are arranged in a single horizontal stack, with a total dimension of 44 mm × 42 = 1.848 m length and 0.355 m width. The tank is also assumed to have a 0.15 m padding full of electrical distribution and cooling components. Therefore, each tank occupies 1.41 m^2, and collectively occupy 1,678 m^2. Each tank is assumed to have an access aisle occupying the same area as the tank itself. Hence, with 1,190 tanks, the tanks and the access aisles occupy 3,356 m^2. The total datacenter space, of which tanks and aisles represent 80%, is estimated at 4,195 m^2. To simplify the calculation, we assume the reduction in the number of racks is proportional to the reduction in the volume of the server's global bounding box. Hence, with an embodied carbon of 509 kgCO2e/m^2, a 1% reduction in volume corresponds to a 33.56 m^2 reduction in floorspace, reducing 17,082 kgCO2e. For reference, each acre of forest in the US sequesters an average of 240 kg of CO2 per year [EPA 2024], so a 1% reduction in server volume corresponds to the carbon sequestration of 71 acres of forest for 1 year.

D Bundles

We define *bundles* as a group of components that are placed together which we treat as a new component type with combined dimensions and padding necessary. The wire length constraint is offset by the distance between the connection point of the bundle and the connection point of the component inside.

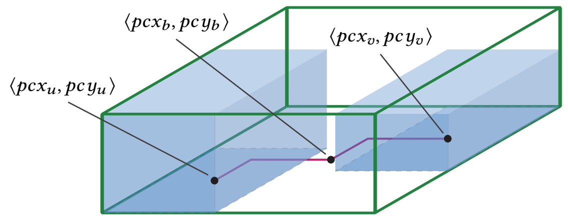

$\langle pcx_b, pcy_b \rangle$
$\langle pcx_v, pcy_v \rangle$
$\langle pcx_u, pcy_u \rangle$

Fig. 16. **Bundles.** Bundle b with two components u and v. The dimension of b is the minimal bounding box that covers u and v (green). The distance constraints is offset by the distance between connection point of bundle and that of u and v (magenta).

Let b be a bundle with component u and v (see Figure 16). Component u and v need to be placed as if they are on the same PCB and meet all overlap and wire length constraints (between u and v, if any). The dimension of b is then minimal bounding box that covers u and v (green). Let pc_u denote the connection point of u (i.e., $\langle pcx_u, pcy_u \rangle$). Suppose the original wire length constraints between u and some other component s is d_{us} and v with s is d_{vs}. The new distance constraint between b and s (d_{bs}) is then:

$$d_{bs} = \min(d_{us} - \|pc_u - pc_b\|_1, d_{vs} - \|pc_v - pc_b\|_1) \tag{43}$$

The formulation above extends to cases where bundles have more than two components.

E Connectors

A *connector* s is a physical equipment that holds the child PCB in place onto the parent PCB p. The dimension of the connector is defined by three values A, B, and C (see Figure 17).

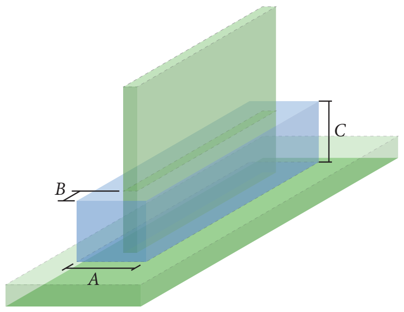


Fig. 17. **Connectors.** The connector s (blue) of the child PCB (green, top) is plugged on to the parent PCB p (green, bottom).

The connector dimension in its parent frame (pl_s, pw_s, ph_s) can be written as:

$$pl_s = \text{mux}(r \longrightarrow w + 2B, \neg r \longrightarrow A) \tag{44}$$

$$pw_s = \text{mux}(r \longrightarrow A, \neg r \longrightarrow l + 2B) \tag{45}$$

$$ph_s = \min(C, ph) \tag{46}$$

where l, w, r are the length, width, and rotation flag of the child PCB and ph is the height of the child PCB in its parent's frame.

We now derive the connector location in its parent's frame (px_s, py_s, pz_s).

$$px_s = x - \text{mux}(r \longrightarrow B, \neg r \longrightarrow (A - h_p)/2) \tag{47}$$

$$py_s = y - \text{mux}(r \longrightarrow (A - h_p)/2, \neg r \longrightarrow A) \tag{48}$$

$$pz_s = \text{mux}(t \longrightarrow h_p, \neg t \longrightarrow -ph_s) \tag{49}$$

where h_p is the PCB thickness and t is the top flag of the child PCB.

Global position and dimension can be obtained by multiplying with $\mathbf{T}_p$.

We constrain the connector to not overlap with other elements except for the child PCB (since they will always overlap by design).

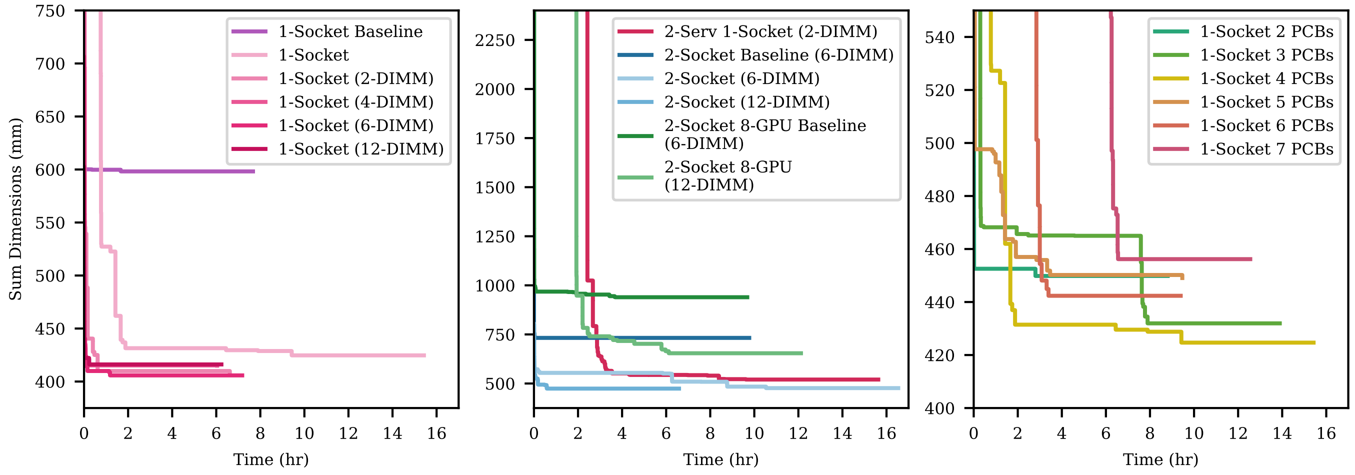
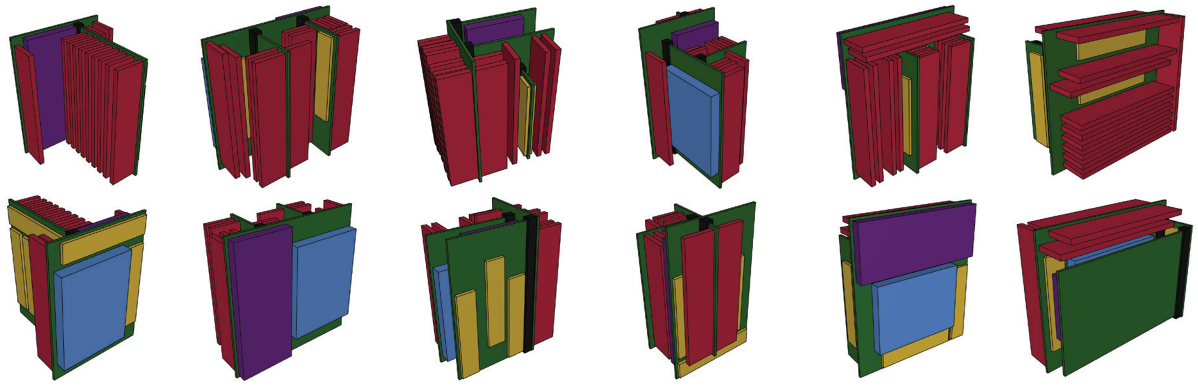


Fig. 18. **Optimization Curves.** Optimization objective versus time of configurations shown in Fig. 7 (left and middle) and 1-Socket configuration with different number of PCBs (right). All optimizations are run without time limit and terminated after a 6-hour period without no new solutions.



	1-Socket (2 PCBs)	1-Socket (3 PCBs)	1-Socket (4 PCBs)	1-Socket (5 PCBs)	1-Socket (6 PCBs)	1-Socket (7 PCBs)
Volume (L)	3.29	2.59	2.54	3.11	2.40	3.03
Sum Dim. (mm)	449.8	432.0	424.7	450.2	442.4	456.2
Space Eff. (%)	53.9	68.5	69.7	57.0	73.9	58.4
PCBs used (#)	2	3	4	4	3	3
Time to Sol. (hr)	2.8	7.9	9.4	3.5	3.4	6.6

Fig. 19. **Comparison between number of PCBs.** Optimized 1-Socket designs with different number of PCBs. The results show objective improvements as we add additional PCB up to 4 PCBs. Beyond 4 PCBs, the search becomes slower and lead to worse results using the same stopping criteria (see Fig. 18). Moreover, we observe the results do not use more than 4 PCBs.