ChatTwin: Toward Automated Digital Twin Generation for Data Center via Large Language Models

Minghao Li
minghao002@e.ntu.edu.sg
Nanyang Technological University
Singapore

Ruihang Wang
ruihang001@e.ntu.edu.sg
Nanyang Technological University
Singapore

Xin Zhou
1020161201@jxstnu.edu.cn
Jiangxi Science and Technology
Normal University

Zhaomeng Zhu
zhaomeng.zhu@ntu.edu.sg
Nanyang Technological University
Singapore

Yonggang Wen
ygwen@ntu.edu.sg
Nanyang Technological University
Singapore

Rui Tan
tanrui@ntu.edu.sg
Nanyang Technological University
Singapore

ABSTRACT

Digital twin has been applied in various industrial fields to represent physical systems. However, the design of high-fidelity digital scenes is challenging in that it often requires intensive manual processes and domain expertise to edit the 3D models or description documents. To reduce human efforts, this paper proposes ChatTwin, a conversational system that leverages the power of GPT-4 to automate the generation of scene description documents for digital twins. ChatTwin assists scene generation by i) segmenting user-input prompts, ii) generating scenes with segmented prompts, and iii) optimizing the generated content. Specifically, the Segment-and-Generate (SG) workflow decomposes the long-text generation into several subtasks and reduces the complexity of the original task. The evaluation through our data center digital twin system shows that ChatTwin outperforms other baselines in terms of generation accuracy and efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Large Language Model, Automated Digital Twin, Data Center

ACM Reference Format:
Minghao Li, Ruihang Wang, Xin Zhou, Zhaomeng Zhu, Yonggang Wen, and Rui Tan. 2023. ChatTwin: Toward Automated Digital Twin Generation for Data Center via Large Language Models. In *The 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '23), November 15–16, 2023, Istanbul, Turkey*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3600100.3623719

1 INTRODUCTION

Digital Twin refers to a virtual model that represents an object or system throughout its lifecycle [4]. On top of physical data and prior knowledge, the digital model is advantageous in assisting predictive analysis and decision making. These advantages have spawned various applications to create digital replicas of the physical assets in industrial fields, such as data center [15], manufacturing [9], and smart grid [6]. Existing industrial digital twins utilize *textual* description documents to build the digital twins. The Azure Digital Twins Definition Language (DTDL) [10] uses a JSON-LD-based language to define certain contents, such as properties, telemetry, and commands. The Universal Scene Description (USD) in Omniverse [11] provides a rich set of features for large-scale 3D content creation and collaboration. These description documents provide a standard and interpretable way to define digital twins. However, the construction of the digital twins is an arduous process. Generating a reliable description file often requires several domain experts to manually operate the 3D models or edit the template files. As the scales of the designed system increase, such as industry-grade data centers with hundreds to thousands of equipment, manually editing the file becomes labor-intensive and error-prone. Therefore, novel methods to automate scene generation would be highly desirable.

The pretrained large language models (LLMs) have shown remarkable performance in a wide range of tasks [16]. A potential application of LLMs is to generate the hierarchical description document based on textual *prompts*. However, directly applying the LLMs to generate the scene description documents for the digital twin faces two challenges. First, the length of the description document is proportional to the complexity of the target scene. The description file may consist of thousands of lines for a system with multiple components. Generating such a long text in a one-shot manner is difficult and inefficient with the pretrained LLMs. Second, the pretrained LLMs used for domain-specific tasks are prone to *hallucination*. In such cases, the generated contents may look correct but physically implausible without verification.

To address the above challenges, we present ChatTwin, which utilizes the power of Generative Pretrained Transformer 4 (GPT-4) [12] to facilitate the generation of comprehensive scene description documents specifically designed for the data center digital twin. ChatTwin features two important designs for generating more reliable results. First, we introduce a Segment-and-Generate (SG) workflow that decomposes the long-text generation into a series of subtasks and thus reduces the likelihood of generating erroneous or inconsistent scene descriptions. Second, we solve an integer programming problem to optimize the imperfect file based on several predefined rules. The post-process eliminates factual errors and improves the overall quality of the generated documents. We conduct

BuildSys '23, November 15–16, 2023, Istanbul, Turkey

ACM ISBN 979-8-4007-0230-3/23/11.
https://doi.org/10.1145/3600100.3623719

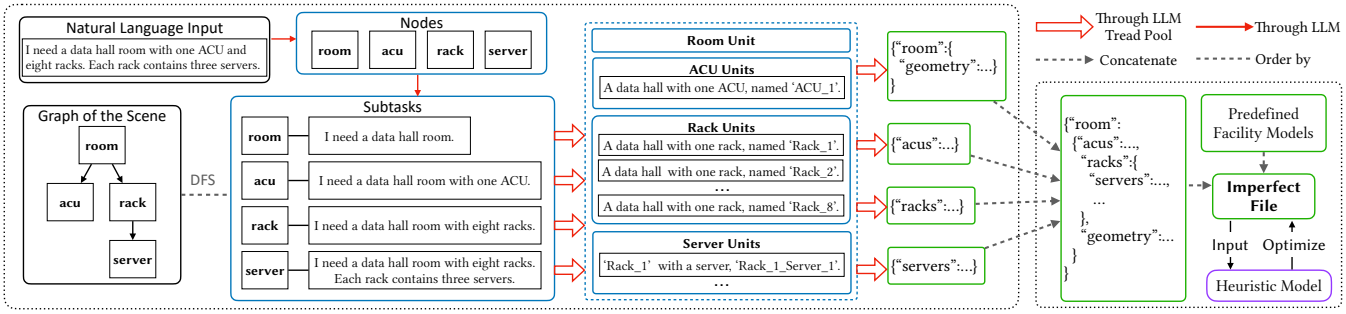

Figure 1: The workflow of ChatTwin in steps. 1) Segmentaion: We process the natural language (NL) task to graph-based sub-tasks and later decompose them into units. 2) Generation: By the instructions of each sub-task, we generate an imperfect file. 3) Optimization: We heuristically post-process it by rules.

experiments on our data center digital twin to evaluate the accuracy and efficiency of the SG compared with baseline prompting methods. Experimental results show that ChatTwin improves the generation accuracy by 55% and 23% compared with zero-shot and few-shot learning, respectively. In terms of efficiency, the designed SG workflow also surpasses the two basic prompting methods by generating 45% and 12.5% more tokens per second.

The contributions of ChatTwin are summarized as follows: 1) We propose the SG workflow and develop ChatTwin for long-text description document generation; 2) We implement ChatTwin and show the performance of LLM technologies to empower the generation of digital twin description files.

2 RELATED WORK

Text-to-3D has become an active research field and spawned various applications in the digital twin. Set-the-Scene [3] develops an agent-based training framework to fill the gap from controllable text to 3D synthesis. Text2Room [8] generates immersive 3D meshes with textures for room-scale environments based on given text prompts. Different from the 3D object generation, the contents in digital twins are predefined in a certain fixed structure and set with parameters for simulation purposes. Therefore, utilizing text-to-text technologies to generate scene descriptive documents (i.e. DTDL, USD) is more suitable for generating digital twins.

3 CHATTWIN OVERVIEW

This paper focuses on the data center as the core application scenario of the ChatTwin design, which consists of the components like, room, air conditioner unit (ACU), rack, and server. Figure 1 shows the overview workflow of ChatTwin, where the left part illustrates the document generation and the right part reveals the post-process of document optimization. To address the challenges of long-text generation in LLMs, we introduce the SG to empower automatic description file generation.

Segment. Instead of outputting all letters of a stringified JSON at once, the SG first cuts the natural language input $\mathcal{I}$ into several parts denoted as $\mathcal{T} = \{t_1, t_2, t_3, \ldots, t_i, \ldots, t_n\}$, where $\mathcal{T}$ is the set of all subtasks by users and t_i represents each subtask. In the scene of the data center digital twin, the facilities are always in a nested structure. For example, the racks are placed in a room and the servers are installed in a rack slot. We denote the facilities as $V = \{v_1, v_2, v_3, \ldots, v_n\}$, where each facility v_i is associated with its parent facility v_p and the directed edge $e_j = (v_p, v_i)$. Their nested tree structure can be predefined by a hierarchical representation $G = \{V, E\}$, where $E = \{e_1, e_2, e_3, \ldots, e_j, \ldots, e_n\}$. Then, t_i can be extracted from the original input $\mathcal{I}$ by its corresponding node facility v_i and its parent node facility v_p as $t_i = E(v_i, v_p, \mathcal{I})$, where t_i is also written in natural language. We continue to break down each subtask t_i into units $U_i = \{u_{i,1}, u_{i,2} \ldots u_{i,k}\}$, and obtain the unit set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ with all the n subtasks. The segmented prompts reduce the complexity of the original task and are expected to improve the generation quality and efficiency.

Generate. We can get a dictionary of subtasks $\mathcal{T}$ with each facility as the key and its description as the value from the above task segmentation. For example, for an original input *"a data center with ten racks"*, the segmented subtasks are identified with the "rack" key associated with the value *"a data center with one rack"* repeated ten times. With the relationships, we solve the tree-structured subtasks $\mathcal{T}$ by order of deep-first search (DFS). A prompt p_i for each unit u_{ik} of subtask t_i yields the corresponding string segment j_{ik}. With predefined prompts for each type of facility, we can simultaneously generate each corresponding segment individually with the thread pool technology. Consequently, we can combine these segments to the string segment j_i for the subtask t_i. After solving all subtasks, we can obtain a set of JSON segments $\mathcal{J} = \{j_1, j_2, j_3, \ldots, j_i, \ldots, j_n\}$, which will be concatenated together by the predefined file structure.

Optimization. Although we can get the scene description document with the right hierarchical structure, the detailed values are always unreasonably set to zero or a random value due to the poor mathematical reasoning ability of the language model [5]. Through the above SG process, we can get an imperfect scene description document with predefined facility models in the correct structure that fits the original input. To obtain a reasonable digital scene, we optimize the geometric values with several predefined rules.

Given the nested structure of the scene description file, the proposed method can be extended to other building-related scene (e.g. data center buildings) generation with the proper description document template and pre-designed rules.

4 IMPLEMENTATION OF CHATTWIN

We apply our ChatTwin to a self-developed data center digital twin. It can build a digital twin of a real-world data center hall that hosts ACUs and racks containing multiple servers. We first introduce the structure of the scene description document and the heuristic approach to optimize the imperfect file.

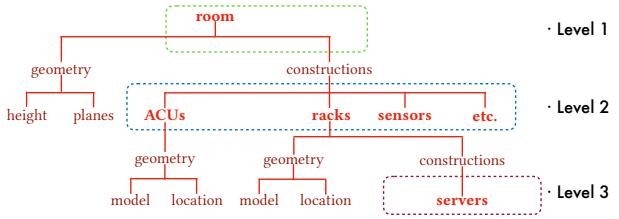

Figure 2: The architecture of the JSON-formatted *dcfile*.

4.1 Document Generation

Similar to existing digital twin systems, the digital twin of a data center is predefined with a textual file termed as *dcfile*. The structure of the *dcfile* is illustrated in Figure 2, which utilizes three levels to represent all facilities in the data center. Specifically, the top level defines only one item, i.e., the "Room" object, which refers to the data hall. The "Room" has several children, i.e., "Racks", "ACUs", "Sensors", etc., which is defined at the second level. In this paper, we focus on the locations of racks and ACUs for optimization in the ChatTwin design. The third level defines the "Servers" object that is accommodated under the "Racks". We first generate the *dcfile* that satisfies the predefined structure and the NL-task requirements, i.e., the number of desired facilities to generate. Then, with the generated file, we obtain the geometry values for each facility and the data hall by solving a domain-specific optimization problem.

4.2 Heuristic Optimization

ChatTwin post-processes the imperfect *dcfile* by heuristic-based integer programming. In this paper, we consider a typical rectangular design of the data hall with width and length denoted by x_w and x_l. The bottom area of each hosted facility is denoted as s_i. We assume a hall room of a data center should be optimized to a maximum utilization denoted as $U = \frac{\sum_i^k s_i}{x_\mathrm{w} \times x_\mathrm{l}}$, where k is the total number of facilities. Then, we denote the total number of racks as n_rack and the total number of ACUs as n_acu, where we assume there is only one type of racks and ACUs, respectively. Given that each facility has four dimensions to determine its location, the incremental number of variables imposes high complexity for solving the optimization problem. Thus, the feasible region of this optimization problem is too broad, considering the locations of every rack and ACU as the variables. Due to the NP-completeness [1], we plan to solve the problem via a heuristic way by the following rules, i.e., constraints for the above problem.

Rule 1: We set the racks in columns with the same gap ξ_rack and use slots to determine the positions of the racks. The rack slot layout is shown as the left part of Figure 3. Thus, we can set the layout of the rack slots by:

$$a \times b + c = n_\mathrm{rack}, \tag{1}$$

where a represents the number of racks in a column, b represents the number of rows, and c represents the number of racks in the extra column. Since a, b, and c decide the layout of the racks, they are regarded as the variables of the optimization problem.

Rule 2: Racks should be placed in the central area as Figure 3, while ACUs should be evenly arranged around the group of racks with gap ξ_acu. The margin μ of the area of racks and the padding ν of the data hall should be adjusted by the operator's preferences.

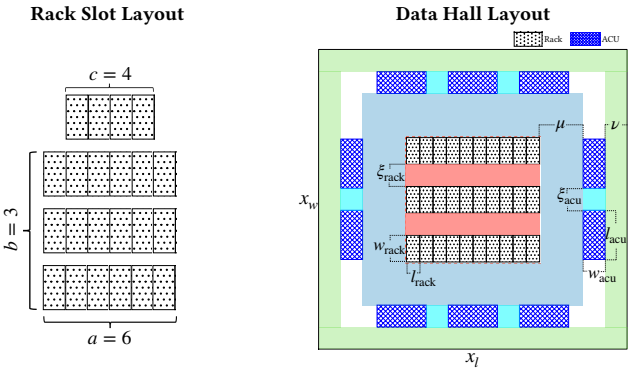

Figure 3: (left) The designed layout for racks in rows and columns. (right) The recommended layout from a large amount of real data hall room design.

The data hall room width x_w and length x_l can be calculated by the rack's width w_rack and length l_rack and the ACU's width w_acu as:

$$\begin{aligned} x_\mathrm{w} &= \begin{cases} a \times w_\mathrm{rack} + 2 \times (\mu + \nu + w_\mathrm{acu}), & a \geq c \\ c \times w_\mathrm{rack} + 2 \times (\mu + \nu + w_\mathrm{acu}), & \text{else} \end{cases} \\ x_\mathrm{l} &= (b+1) \times l_\mathrm{rack} + b \times \xi_\mathrm{rack} + 2 \times (\mu + \nu + w_\mathrm{acu}). \end{aligned} \tag{2}$$

Rule 3: We ensure there is no facility placed out of the hall. The hall shapes in rectangular without any extra pillars in the room. Considering the number of ACUs n_acu, we have the constraints of room width x_w and length x_l in Eq. (2), respectively, as:

$$\begin{aligned} x_\mathrm{w} &\geq m \times l_\mathrm{acu} + (m-1) \times \xi_\mathrm{acu}, \\ x_\mathrm{l} &\geq m \times l_\mathrm{acu} + (m-1) \times \xi_\mathrm{acu}, \end{aligned} \tag{3}$$

where $m = \lceil \frac{n_\mathrm{acu}}{4} \rceil$ means the largest possible number of ACUs on one side. From the three rules, we formulate a convex problem of integer programming:

$$\begin{aligned} &\underset{a,b,c \in \mathbb{Z}^+}{\text{minimize}} \; x_\mathrm{w} \times x_\mathrm{l} \\ &\text{subject to } (1), (3) \end{aligned} \tag{4}$$

We apply grid search to solve the optimization since the dimension of this problem is only three. In other words, the time complexity of the algorithm is $O(n^3)$. The thermal effects are influenced by the hyperparameters, including the gap between racks (ξ_rack), the gap between ACUs (ξ_acu), the margin (μ), and the padding (ν).

5 EVALUATION AND RESULTS

This section evaluates the effectiveness of ChatTwin and compares its performance with two baselines.

Baselines. For scene description documents generation, we compare our solution with two common types of prompting baselines: 1) Zero-shot prompting We provide the whole template for generating description documents without any clues and implements. 2) Few-shot prompting Besides providing the whole template of the scene description document, few-shot prompting provides several examples for different cases.

Implementations. We use the cutting-edge GPT-4 through the API provided by OpenAI as the backbone language model. The settings are applied to all baselines and our approach, including zero-shot, few-shot, and SG prompting for generation. We first create

Table 1: The success rates and the average token lengths of generated files of zero-shot, few-shot, and SG.

Methods	Success Rate	Avg. L_{token}
Zero-shot	32%	1656.80
Few-shot	64%	2180.15
SG	87%	3712.02

100 data hall descriptions in natural language without labeling the corresponding *dcfile*. Then, we compare the SG with the other two baseline prompting methods using the following metrics.

Metrics. In this paper, we report the *success rate* and *generation efficiency* across 100 descriptions as the metrics. The descriptions only focus on the rack, ACU, servers, and room in three levels. An example description is *"I want a data hall room with four ACUs and ten racks. Each rack contains two servers"*. The success rate is defined as the ratio of generated files with correct facilities. The generation efficiency is denoted by $\eta = \frac{L_{\text{token}}}{S}$, where S is the makespan and L_{token} is the generated token length measured by TikToken [13].

Results. We next show the results in this section to evaluate the ability of ChatTwin to generate the scene description document in the aforementioned digital twin system.

According to Table 1, the SG workflow achieves the highest success rate 87% compared with zero-shot and few-shot, which only have 32% and 64%, respectively. The SG can also generate longer and more complete files than the baselines, which outperforms 124% and 70% tokens on average. The efficiency of the SG workflow is revealed in Figure 4 compared with zero-shot and few-shot prompting methods. In terms of efficiency, some samples generated using the few-shot method outperform the SG method. This is because the additional time required for the segment process is redundant when generating short textual documents. With the help of the thread pool in ChatTwin, task units can be executed simultaneously, which brings 45.0% and 12.5% speed acceleration compared to the zero-shot and few-shot methods, respectively. ChatTwin can mitigate the influence of segmenting process of SG and generate the scene description file with a much higher success rate and completeness.

We also evaluate the efficiency of generating large digital twin scenes. Since zero-shot and few-shot are not effective for small data halls, we only evaluate SG. We gradually increase the number of racks from 10 to 50 and servers from 5 to 20 accommodated by each rack in the certain description. The result, shown in Figure 5, indicates that with the increasing number of racks and servers, the average makespan becomes longer. Besides, the SG can handle the scenario when $n_{\text{rack}} \leq 30$, since n_{server} becomes an insignificant factor to the makespan. When the user wants to generate a large industrial scene, SG also performs well when $n_{\text{rack}} > 30$. The maximum makespan is still under 350 seconds, which is taken by the task of building the scene of a huge data hall with 16 ACUs and 50 racks containing 1,000 servers in total. Such a scale captures the largest data hall rooms in the current industry-grade data centers [14]. The corresponding *dcfile* consists of 289,127 characters.

6 CONCLUSION AND FUTURE WORK

In this paper, we design ChatTwin to efficiently generate the description files, and reveal the ability of LLM in modeling and generating data center scenes of the digital twin. We found that there are many repetitive segments among units at the same level as shown in Figure 1. To fix the scale issue and accelerate the generation process in SG workflow, we will employ the concept of program-aided language model (PAL) [7] in our future work. In ChatTwin, the heuristic optimization has four hyperparameters, which are required to be manually tuned for thermal constraints. In future work, the optimization objectives will incorporate both thermal and geometry factors based on the given parameters of each facility. The optimization module of ChatTwin will also be compatible with the scale of the real-world data center that consists of multiple data halls, chiller plants, and other functional rooms. Besides, the performance of ChatTwin majorly depends on the GPT-4 API, which is updated opaquely. Some researchers claim that the performance of GPT-4 has been shrinking over time from March to June 2023 [2]. To get rid of network and performance fluctuations, we plan to fine-tune LLaMa-2 for our downstream tasks.

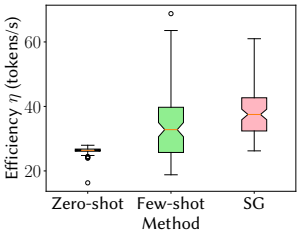

Figure 4: The efficiency of generating scene documents by zero-shot prompting, few-shot prompting, and SG.

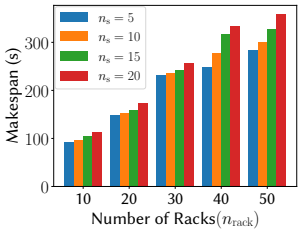

Figure 5: The average makespan of generating various numbers of racks and inside servers by SG.

REFERENCES

[1] J. Chen, W. Zhu, and M. M. Ali. 2010. A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning. *IEEE Trans. SMC* 41, 4 (2010), 544–553.
[2] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009* (2023).
[3] D. Cohen-Bar, E. Richardson, G. Metzer, R. Giryes, and D. Cohen-Or. 2023. Set-the-Scene: Global-Local Training for Generating Controllable NeRF Scenes. *arXiv:2303.13450* (2023).
[4] A. El Saddik. 2018. Digital twins: The convergence of multimedia technologies. *IEEE multimedia* 25, 2 (2018), 87–92.
[5] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. C. Petersen, A. Chevalier, and J. Berner. 2023. Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867* (2023).
[6] G. Gao, C. Song, T. G. T. A. Bandara, M. Shen, F. Yang, W. Posdorfer, D. Tao, and Y. Wen. 2021. FogChain: A blockchain-based peer-to-peer solar power trading system powered by fog AI. *IEEE IoTJ* 9, 7 (2021), 5200–5215.
[7] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig. 2023. Pal: Program-aided language models. In *ICML*. 10764–10799.
[8] L. Höllein, A. Cao, A. Owens, et al. 2023. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv:2303.11989* (2023).
[9] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. Digital Twin in manufacturing: A categorical literature review and classification. *IfacPapersOnline* 51, 11 (2018), 1016–1022.
[10] Microsoft. 2023. Azure Digital Twins - Conceptual Overview and Models. https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models.
[11] NVIDIA. 2023. NVIDIA Omniverse. https://www.nvidia.com/en-sg/omniverse/.
[12] OpenAI. 2023. GPT-4 Research. https://openai.com/research/gpt-4.
[13] OpenAI. 2023. Tiktoken, a fast BPE tokeniser for use with OpenAI's models. https://github.com/openai/tiktoken.
[14] J. Wan, X. Gui, S. Kasahara, Y. Zhang, and R. Zhang. 2018. Air flow measurement and management for improving cooling and energy efficiency in raised-floor data centers: A survey. *IEEE Access* (2018), 48867–48901.
[15] R. Wang, X. Zhou, L. Dong, Y. Wen, R. Tan, L. Chen, G. Wang, and F. Zeng. 2020. Kalibre: Knowledge-based neural surrogate model calibration for data center digital twins. In *ACM BuildSys*. 200–209.
[16] W. Zhao, K. Zhou, J. Li, et al. 2023. A survey of large language models. *arXiv:2303.18223* (2023).