Computational Visual Media
https://doi.org/10.1007/s41095-0xx-xxxx-x

Research Article

PCLVis: Visual Analytics of Process Communication Latency in Large-Scale Simulation

Chongke Bi[1], Xin Gao[1], Baofeng Fu[1], Yuheng Zhao[2], Siming Chen[2], Ying Zhao[3], Yunhai Wang(✉)[4]

Abstract Large-scale simulations on supercomputers have become important tools for users. However, their scalability remains a problem due to the huge communication cost among parallel processes. Most of the existing communication latency analysis methods rely on the physical link layer information, which is only available to administrators. In this paper, a framework called PCLVis is proposed to help general users analyze process communication latency (PCL) events. Instead of the physical link layer information, the PCLVis uses the MPI process communication data for the analysis. First, a spatial PCL event locating method is developed. All processes with high correlation are classified into a single cluster by constructing a process-correlation tree. Second, the propagation path of PCL events is analyzed by constructing a communication-dependency-based directed acyclic graph (DAG), which can help users interactively explore a PCL event from the temporal evolution of a located PCL events cluster. In this graph, a sliding window algorithm is designed to generate the PCL events abstraction. Meanwhile, a new glyph called communication state glyph (CS-Glyph) is designed for each process to show its communication states, including its in/out messages and load balance. Each leaf node can be further unfolded to view additional information. Third, a PCL event attribution strategy is formulated to help users optimize their simulations. The effectiveness of the PCLVis framework is demonstrated by analyzing the PCL events of several simulations running on the TH-1A supercomputer. By using the proposed framework, users can greatly improve the efficiency of their simulations.

Keywords PCLVis, communication latency, large-scale simulation, visual analytics

1 Introduction

Large-scale simulations have found widespread utility across diverse fields, including fluid mechanics, aerodynamics, and aerospace. Leveraging the computational might of supercomputers for virtual simulations offers a promising means to conserve precious resources. These supercomputer systems comprise two foundational components: compute nodes and communication network infrastructure. Each computing node is equipped with its distinct private memory space, ensuring isolation from other nodes. In the context of large-scale simulations, supercomputers partition computational tasks into discrete processes for parallel execution across distinct compute nodes. These processes necessitate communication for synchronized operation and data exchange, highlighting the centrality of process communication.

However, the performance of simulation applications can be curtailed by bottlenecks in process communication[1, 2]. Communication delay events can unpredictably manifest at any juncture, prompting an imperative to dissect the underlying causes. Existing methods for communication latency analysis predominantly lean on the physical link layer, furnishing granular information encompassing start and end points, routing paths, and intermediate waypoints (routers). This information empowers the accurate identification of communication latency events. Nonetheless, the accessibility of physical link layer insights is confined to system administrators, limiting general users' access.

In the absence of access to physical link layer data, general

1 College of Intelligence and Computing, Tianjin University, Tianjin, 300072, China. E-mail: bichongke@tju.edu.cn, gao_xin_private@163.com, fubaofeng96@163.com.

2 School of Data Science, Fudan University, Shanghai, 200433, China. E-mail: 22110980020@m.fudan.edu.cn, simingchen@fudan.edu.cn.

3 School of Computer Science and Engineering, Central South University, Changsha, 410083, China. E-mail: zhaoying@csu.edu.cn.

4 School of Computer Science and Technology, Shandong University, Qingdao, 266237, China. E-mail: cloudseawang@gmail.com.

Manuscript received: xxxx-xx-xx; accepted: xxxx-xx-xx

Fig. 1 Visual analytics of the PCL events from a parallel application running with 1152 processes on a TH-1A supercomputer. (a) shows the spatio-PCL event clustering results obtained using our process-correlation-tree-based algorithm. The temporal evolution of each cluster can be viewed in (b1), and the details can be further interactively explored in (b2). (c) shows the constructed communication-dependency-based DAG graph, which helps users further explore the extracted PCL events. For each selected process, its former and latter connected processes are denoted by the red and blue curves, respectively. The leaf process can be further unfolded as shown in (c2), and the design of each glyph is shown in (c3). The possible PCL causes as summarized in (d) include (d1) poor process-to-processor mapping (too much intercommunication), (d2) poor communication pattern (unbalanced workload), and (d3) background traffic due to the application of other users. (e) shows the result obtained using an optimized process-to-processor mapping. The ratio between the intra- and inter-communications became larger than the original value reported in (d1).

users are relegated to investigating communication delays via Message Passing Interface (MPI) process communication data. To this end, we initiate our analysis by preprocessing the data. Specifically, we curate a communication event dataset through the scrutiny of parallel execution traces recorded by the Tuning Analysis Utility (TAU), yielding essential communication trace data. However, while MPI process communication data encapsulates the communication's origin (Source MPI rank) and destination (destination MPI rank), the absence of routing path and router details poses two fundamental challenges [3]. Firstly, pinpointing the latency region becomes intricate, given that delays often manifest at intermediary waypoints such as routers or routing paths. Secondly, unraveling the propagation path of communication delay events proves complex. The propagation path, integral to the routing path, remains elusive to general users devoid of access to the physical link layer.

To bridge this gap, we introduce 'PCLVis,' an innovative framework tailored for the wider supercomputer user base. PCLVis facilitates the visual analysis of process communication latency (PCL) events utilizing readily available MPI process communication data. First, a space-PCL event localization method is developed to localize processes involved in PCL events. All processes with a high correlation are classified into a single cluster by constructing a process-correlation tree. Second, the propagation path of PCL events is analyzed by constructing a communication-dependency-based directed acyclic graph (DAG), which can help users interactively explore a PCL event from the temporal evolution of a located PCL events cluster. In the end, A PCL events attribution strategy is also designed to help users optimize their simulations. And in summary, the graphics user interface of PCLVis is shown in Figure 1.

Based on the above works, PCLVis is evaluated via visual analytics of three types of communication data on a supercomputer. Results show that the proposed framework can help users greatly improve the efficiency of their simulations.

The main contributions of this work include:

- A process-correlation-tree-based spatio-PCL event locating method is proposed to help users locate those communication regions with high latency.
- A communication-dependency-based DAG is constructed to help users track the propagation path of PCL events.

- A PCL events attribution strategy is designed to help users optimize their simulations.

2 Related Work

This section reviews some existing work on communication delay analysis methods (Section 2.1), processes communication visualization methods (Section 2.2), and communication delay attribution methods (Section 2.3).

2.1 Communication Delay Analysis

Analyzing communication delays from a large-scale simulation is a challenging task due to the large data size. Some studies have attempted to reduce the data size by using clustering [4–7], compression [8–11], and some other algorithms [12, 13]. Nevertheless, the precision of analysis results does not satisfy the user requirements because some important features have been lost during the data reduction process. Several analysis tools can visualize the basic and statistical information of communication events, like the scalasca performance toolset [14], multi-threaded parallel application analysis tool [15], the TAU parallel performance tool [16], the HPCToolkit [17], the Craypat-cray X1 performance analysis tool [18], and some other tools [19–21]. However, they did not analyze communication delay events. Some researchers have proposed structure-based analysis methods. For instance, Hendriks et al. [22, 23] proposed critical-path analysis and distance analysis algorithms to help users track communication delays. Execution phases have also been used to track the communication state in different running phases [24–26]. Pattern matching methods [27–29] have been proposed to detect predefined communication patterns and have successfully extracted inefficient communication behaviors. Isaacs et al. [30–32] proposed a series of methods that can help users detect communication delays by extracting a logical structure from parallel tracking data. While these methods can help users detect communication delays, their accuracy cannot be guaranteed. To address this problem, this paper proposes a spatial-temporal communication delay locating method that can successfully locate all communication delays from large-scale communication data, thereby helping users further analyze these delays.

2.2 Process Communication Visualization

Process communication is a type of event sequence data. Therefore, this section not only examines process communication but also surveys the visualization of other event sequences. Several visualization tools have been developed, including VAMPIR [33], Paragraph [34], Paraver [35], Projections [36], which can visualize the performance of a simulation using a Gantt chart. However, these tools cannot support users in further exploring the details of process communication. They also cannot be directly used to analyze large-scale simulations due to the limitations in their scalability. Some improvement methods [37–39] have been proposed by making good use of sorting algorithms. While these Gantt-chart-based methods can be used to visualize event sequences, they still do not fully meet user requirements, especially for visualizing long event sequence data, including process communication.

To visualize a large-scale event sequence, flow-based methods can offer a visualization abstract for users [40]by aggregating event sequences [41]. Some scholars have attempted to reveal the sequence evolution pattern by using aggregated visualization methods, including simplification [42], flow design [43, 44], feature extraction [45], progression analysis [46], and some methods for a specific application [47, 48]. When visualizing a large-scale communication event sequence, scale-related issues are addressed by changing the vertical axis from processes to event duration [49, 50]. Isaacs et al. [51] reduced visual clutter by adopting a layered abstraction algorithm to visualize the logicalized process communication sequence. LBVis [52] applies an interactive visualization method to show the data transmission among different processes. While these methods can successfully visualize large-scale communication data, they do not offer communication dependency and state information to users, which are vital for understanding the evolution of communication latency. To address these limitations, this paper constructs a communication-dependency-based DAG that can help users track communication latency. A CS-Glyph is also designed to show the detailed state of a process.

2.3 Communication Delay Attribution

Communication delay attribution is important for users to improve the efficiency of their simulation. Most existing communication delay attribution methods mainly rely on analyzing the link layer log information [53–56]. In other words, these methods utilize the logs in routers through which messages have passed to check for possible communication latency. Fujiwara et al. [57] evaluated the transmission efficiency in a communication path using hop bytes and designed a re-routing and remapping algorithm to optimize the communication efficiency of a simulation. Li et al. [58] analyzed the performance of simulations in different workloads and routing strategies by using physical link layer information. Jha et al. [59] summarized the causes of communication latency by detecting network congestion using link layer information.

Meanwhile, Taffet et al. [60] attributed communication delays to three causes, namely, poor process-to-processor mapping (placement of MPI ranks on physical cores), poor communication patterns, and background traffic due to the application of other users.

A communication pattern illustrates how processes send and receive data to one another. Some example patterns include the many-to-one and nearest-neighbor communication patterns. All of these patterns belong to an unbalanced workload [61]. A poor communication pattern can lead to congestion and communication latency. Demmel et al. [62] reduced latency events using the communication avoidance algorithm. A poor process-to-processor mapping only extends the message transmission path, thereby increasing communication latency. Yan et al. [63] greatly improved communication performance by designing a good process-to-processor mapping. Background traffic is considered a job interference [64]. The network of a supercomputer is shared by many users, thereby restricting the performance of their simulations.

The majority of the above methods rely on link layer log information, which is only available to the administrators of a supercomputer. Therefore, this paper designs PCLVis as a process-information-based communication delay attribution method to help users analyze the causes of communication latency and improve the efficiency of their simulations.

3 OVERVIEW OF PCLVis

This section summarizes the user requirements (Section 3.1) and presents the PCLVis design (Section 3.2).

3.1 User Requirements

The requirements presented in this section were gathered through comprehensive interviews with a diverse group of supercomputer users. With insights derived from these interviews, we have distilled the users' needs into the following synthesized set of requirements:

R1: Locate high latency region. In large-scale supercomputer simulations, users face a complex communication environment. They need an intuitive way to understand communication dynamics throughout the simulation, enabling comprehensive analysis of design mechanisms. However, the sheer volume of communication data exchanged among processes presents a challenge. Variability in latency due to supercomputer background traffic adds complexity, making it impractical to set a fixed latency threshold. To address these issues, our system efficiently locates all latency instances in extensive communication data.

R2: Explore temporal evolution of latency. In extensive, long-duration simulations spanning days or weeks, users aim to understand how communication delays evolve over time in specific regions. This understanding helps identify and grasp periods of increased latency, enabling timely adjustments and effective responses during the entire simulation. Our system is designed to facilitate the extraction of crucial latency periods, empowering users to delve into comprehensive analyses of communication latencies over time.

R3: Communication delay attribution. Users require an efficient and user-friendly approach to assess communication latencies and their origins. This is vital for enabling ordinary users to engage with the simulation effectively. Our system meets this need by clearly depicting communication latency dependencies and summarizing their potential sources, making it easier for users to comprehend logical connections and identify underlying causes.

R4: Offer possible optimization schemes. Following the analysis of delay causes, users seek actionable strategies to counter communication delays within diverse timeframes and regions. They anticipate receiving initial optimization approaches or guidance to initiate effective improvements. In response to this, our system is designed to offer a range of potential optimization schemes, empowering users to proactively address communication delays through informed decisions.

3.2 System Overview

According to the abovementioned requirements, we design PCLVis for a visual analysis of PCL events in a large-scale simulation.

To analyze communication delay using MPI process communication data, we first preprocess the data. Specifically, we built a communication event dataset by analyzing parallel execution traces collected by the Tuning Analysis Utility (TAU) to obtain communication trace data [16]. During the runtime of the simulation program, TAU intelligently detects functions, methods, and code blocks, thereby capturing essential communication tracing data. The properties of a communication event are as follows:

- *MPI Rank*: Unique process to which the event belongs.
- *Type*: Two types of communication events (i.e., send or receive).
- *Timestamp*: Wall clock time of the event.
- *Source*: Source MPI rank of the event.
- *Destination*: Destination MPI rank of the event.
- *Message Size*: Size of message within the event.

PCLVis has three main parts, namely, spatial latency, temporal latency, and attribution. First, the spatial latency part offers information about communication latency within regions. In this part, a spatio-PCL event locating method is

proposed to help users locate those communication regions with high latency (**R1**). Second, the temporal latency of these communication regions is analyzed by DAG. We offer an evolution view to display the latency abstraction over time. Users can select a period of high latency to further explore the communication latency among processes in DAG (**R2**). Third, the attribution part offers three views to display information about the process of communication states. Users can analyze the causes of communication latencies using our proposed attribution strategy (**R3**) and then optimize their simulations using an appropriate method that is provided through our system. Such methods may include optimizing the physical mapping of nodes, adjusting the communication patterns, and choosing the running time with better background traffic (**R4**).

4 LOCATING SPATIAL PROCESS COMMUNICATION LATENCY EVENTS

This section introduces our spatio-PCL events locating method (**R1**). A process-correlation tree is constructed to divide all processes into different clusters (Section 4.2), and several criteria for defining a PCL event and the latency of a communication region are prepared (Section 4.1).

4.1 Criteria for Defining Communication Latency Within a Region

Given that a supercomputer is shared among users, the theoretical transmission speed cannot be used to define a PCL event. At the same time, there is no proper way to describe a communication delay standard in the given region. Therefore, this paper designs a statistical method for defining a PCL event in a region.

In the following tasks, we define two latency criteria (transmission time) for intra-node and inter-node communication. The difference between these criteria lies in whether two communication processes are in the same physical nodes or not given that the transmission speed of inter-node communication is much lower than that of intra-node communication [65].

First, we separately sample intra-node and inter-node messages and then collect as many as 10,000 messages for each sampling message-size. Second, we sort the messages with the same size in an ascending order. ***Latency criteria*** is defined as the median transmission time for a particular message-size. The latency criteria defined by our method are shown in Figure 2. As can be seen in the figure, the message transmission time linearly increases along with message size, and the inter-node latency criteria are much higher than the intra-node criteria.

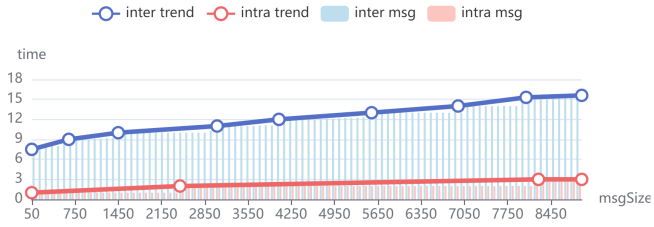


Fig. 2 Latency criteria of PCL events. The x-axis represents message size, whereas the y-axis represents time. The blue and red lines denote the latency criteria of intra-node and inter-node messages, respectively. The message size is sampled with an interval of 50 bytes.

Utilizing these latency criteria, the communication latency of a message can be calculated across three perspectives: spatial latency, temporal evolution, and PCL dependency, employing Equation 1.

$$latency_{msg} = \frac{t_{msg}}{gt_{msg}} \tag{1}$$

Where t_{msg} is the real transmission time of msg, and gt_{msg} is the latency criteria of msg. If $latency_{msg} > 1$, then msg is considered delayed. A larger $latency_{msg}$, corresponds to a longer delay. A high PCL event is defined as the communication event with the larger $latency_{msg}$.

The latency of a communication region can be calculated as

$$RegionLatency = \frac{1}{n}\sum_{i=1}^{n} latency_{msg_i} \tag{2}$$

Where n is the number of messages contained in the current communication region.

4.2 Process-Correlation-Tree-Based Spatial Clustering

This section describes the process-correlation-tree-based spatial clustering method 4.2.1 and presents the visualization of communication regions extracted by our clustering method and the communication latency regions extracted by our own latency criteria (Section 4.2.2)(**R1**).

4.2.1 Clustering Method

We define a ***region*** as a collection of multiple processes, which satisfies the idea that the communication inside the region is much more intensive than that outside the region. Therefore, PCL can be defined as a situation wherein the processes in the same region affect the communication latency of one another.

To generate communication regions, we introduce a metric called ***Process-Correlation***, which is used in our clustering method. This metric reflects how closely any pair of processes communicate with each other. Processes communication relationships are calculated by a tree structure. For process

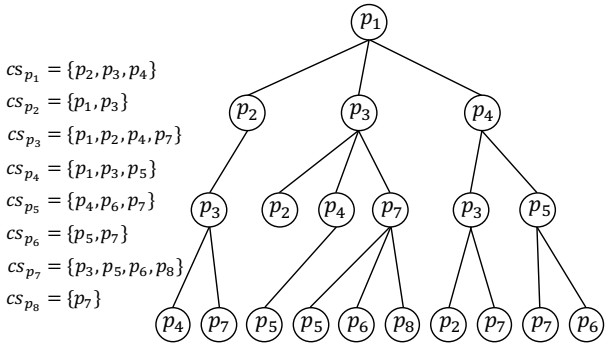


Fig. 3 The symmetry of process correlations in the process-correlation tree. (a) Process-correlation tree. The dataset on the left describes the process of communication. cs_{p_i} denotes the collection of processes communicating with p_i. The tree on the right describes the correlation of cs_{p_1} with the other processes. The red edge represents the path connecting cs_{p_1} and cs_{p_2}. (b) Process-correlation tree describing the correlation between cs_{p_1} and cs_{p_2}.

p, cs_p is defined as the communication collection of p. All processes in cs_p must send/receive messages to/from p. We build a tree ($tree_p$) for every process p as follows:

Take p as the root node of $tree_p$, and take the processes in cs_p as the children of p. For each child of p, we generate children for them in the same way as we did for p. We repeat the above step to construct the tree. The following discipline ensures that we can construct a unique tree for p: the process ID on the path from the root node to any leaf node is not repeated. That is, the ancestor and descendant nodes of node i cannot be the same. Figure 3(a) shows the process-correlation tree of p_1.

As depicted in Figure 3, our dataset showcases a discernibly tree-like configuration among its constituent elements. Leveraging this inherent characteristic, we employ Equation 3 to calculate the interdependencies existing among processes.

$$R(p,q) = \sum_{v \in V, v.pid=q} \frac{1}{(v.depth)^2} \tag{3}$$

where p is the root node of $tree_p$, q is a process in cs_p, $R(p,q)$ is the process-correlation between p and q, V is the collection of all nodes in $tree_p$, $v.depth$ is the depth of node v, and $v.pid$ is the process ID of node v. A larger $R(p,q)$ indicates more communication between p and q. If i does not appear in $tree_p$, $R(p,i)$ is 0.

Table 1 Process-Correlation

	p1	p2	p3	p4	p5	p6	p7	p8
p1		1.36	1.5	1.36	0.47	0.22	0.58	0.11
p2	1.36		1.36	0.72	0.33	0.11	0.36	0.11
p3	1.5	1.36		1.47	0.72	0.47	1.11	0.25
p4	1.36	0.72	1.47		1.11	0.47	0.72	0.22
p5	0.47	0.33	0.72	1.11		1.25	1.36	0.36
p6	0.22	0.11	0.47	0.47	1.25		1.25	0.36
p7	0.58	0.36	1.11	0.72	1.36	1.25		1
p8	0.11	0.11	0.25	0.22	0.36	0.36	1	

Using Equation 3, the process-correlation between each pair of processes can be obtained as shown in Table 1. A larger value means more communication between two processes, and vice versa. When the value is 0, it means that the two processes have no communication. As shown in the table, although each process has a unique tree structure, the process-correlation value between a pair of processes is equal because the above tree structure describes the specific process communication relationship, whereas the process-correlation only describes the closeness of communication between any pair of processes.

The following example demonstrates the symmetry of process correlations in the process-correlation tree. According to the tree structure shown in Figure 3(a), the identifiable connection path between processes p_1 and p_2 becomes evident, visually highlighted by the presence of a red line Further, the process correlation tree of processes p_1 and p_2 can be drawn, as shown in Figure 3(b). Three paths from p_1 to p_2 are shown on the left, and three paths from p_2 to p_1 are shown on the right. Each path corresponds to the same process and depth because the physical node does not change during the process of communication. Therefore, the communication relationship between processes does not change whether a message is sent or received. According to Equation 3, $R(1,2)$ is equal to $R(2,1)$.

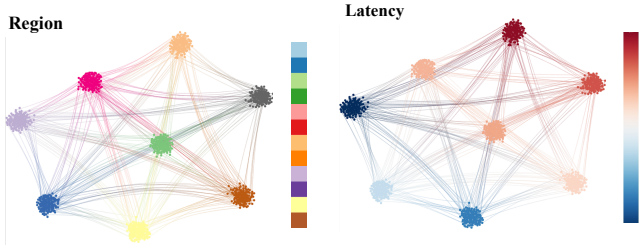
As depicted in Figure 3, there exists a strong hierarchical association among processes. Therefore, after obtaining the inter-process associations, we employ hierarchical clustering [66] to cluster the processes based on these associations. The clustering method follows a bottom-up strategy, iteratively merging nodes into increasingly larger clusters based on their similarity or distance. The specific algorithmic workflow for the communication cluster extraction method based on inter-process associations is outlined below:

1. Initially, all processes within the process set are treated as individual clusters, forming the leaves of the hierarchical structure.

2. Subsequently, based on the associations among the processes within a cluster, we compute a pairwise linkage metric between every pair of clusters. In this study, the average distance metric is adopted, as defined by Equation 3.

3. The two clusters with the maximum linkage metric are merged into a single cluster, forming a non-leaf node in the hierarchical tree.

4. Steps 2 and 3 are iteratively performed until the association between any two clusters falls below a threshold. In this



(a) (b)

Fig. 4 (a) The communication regions graph, where each communication region is encoded by a certain color. (b) The latency mode of the graph, where the color goes from blue to red, indicates that the latency increases from low to high.

study, the threshold is set to 2, and clustering ceases when there is only a single communication message between two clusters.

Applying the clustering method to the example data shown in Figure 3 forms two communication regions, namely, $Region_1 = \{P_1, P_2, P_3, P_4\}$ and $Region_2 = \{P_5, P_6, P_7, P_8\}$. All processes inside these regions communicate much more than those between $Region1$ and $Region2$.

4.2.2 *Visualization of Communication Regions*

A communication region graph is designed to illustrate the relationship between processes within a region and the relationship between regions. Figure 4(a) shows a total of 1024 processes, which are divided into 8 communication regions. A graph node represents a process, whereas a link between two nodes represents the communication between processes. Each region is represented by a color for users to intuitively understand the complex communication relationship. Our design aims to show the overall clustering results and not just a single process.

Through spatial clustering of process nodes, we find the process nodes with the closest communication and divide them into the same area to further visualize them. Users can select specific process node regions for detailed analysis according to the clustering results in the communication region graph.

To improve the visual experience of users, we use 2D force-directed edge building (FDEB), an edge bundling algorithm, to avoid cluttered connections. The system also provides many functions, including zooming in/out, translating, rotating, and dragging, for users to further observe and analyze the clustering results.

We also calculate the communication latency of each region using the message latency calculation method provided by the Equation 2. The communication latency of regions is characterized by the communication region graph. As shown in Figure 4(b), as the color goes from blue to red, the communication latency of a region goes from low to high. This graph synchronously displays the process communication delay information between different regions in real-time. Users can select those regions with high communication latency for analysis as shown in Figure 4.

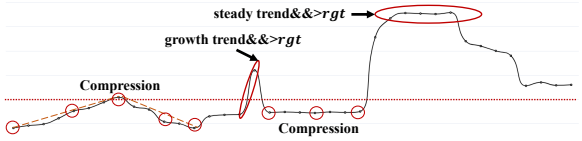


Fig. 5 Features of communication latency over time, which are used to generate temporal latency abstraction using the sliding window algorithm. The red line denotes gt_{region}, and the important *growth trend period* and*steady trend period* are circled by ellipses. The other features are compressed (reserve part of the data).

5 COMMUNICATION-DEPENDENCY-BASED DAG

This section discusses the construction of communication-dependency-based DAG Section 5.3 (**R2**), the temporal evolution of PCL events abstraction Section 5.1, and the communication dependency extraction method Section 5.2.

5.1 Temporal Evolution of Process Communication Latency Events Abstraction

A sliding-window-based algorithm is proposed to extract the features of PCL events within communication regions. This algorithm generates an abstraction of temporal evolution.

First, gt_{region} is defined as the average latency of all messages in the $region$ (Section 4.1) . Second, we define two important features to be extracted, namely, the *growth trend period* and *steady trend period*, as illustrated in Figure 5. For other unimportant periods, we reserve three timestamps to compress the time period, namely, start, mid, and end. Based on the features defined above, we use the sliding-window-based algorithm to generate the temporal latency abstraction.

The *growth trend period* refers to a time interval during which the delay data exhibits a sustained increasing trend. This could indicate changes within the communication network leading to a gradual rise in message delays. An example of this feature is depicted in Figure 5, showing a steady upward trend in delay data.

Conversely, the *steady trend period* implies that despite stable communication latency, delays are consistently maintained at a higher level, making it difficult to revert to lower values. The illustration in Figure 5 further clarifies this point, emphasizing fluctuations in message delays at a higher level.

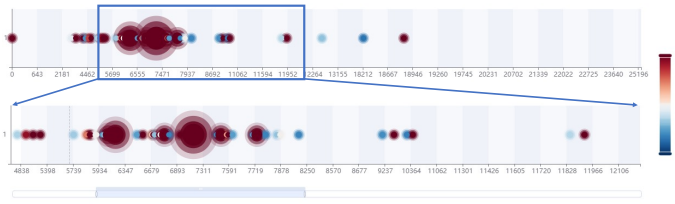


Fig. 6 Evolution of the latency of a communication region. The abstraction of communication latency of segments is generated based on temporal latency features. A slider component is used to select a certain period to analyze.

For time intervals of lesser importance, we opt to retain three timestamps denoting the start, midpoint, and end, allowing us to compress these periods effectively.

Building upon the definitions provided above, we employ a sliding-window-based algorithm to generate the abstraction of temporal latency. We believe that this approach better captures the features of events within the communication area, offering valuable insights for subsequent analyses.

We also design a temporal evolution view of region communication latency to help users locate those periods with high latency. The temporal evolution view of PCL events shows the overall latency abstraction over time, which can help users interactively explore these events. The DAG (Section 5.3) is constructed in the period that users have selected through the slider component below. The symbols in Figure 6 are defined as follows:

- *X-axis*: continuous time periods.
- *Circle*: Communication latency of a period.
- *Color of circle*: Communication latency of a region (Equation 2).
- *Size of circle*: Number of messages delayed in a region.
- *Arrow point*: Details of the communication latency.

5.2 Communication Dependency Among Processes

Based on the physical time order of communication events, the logical time is calculated using the Vector Clock approach [67]. This approach involves the following mathematical formulation:

For each event e, let $vec_p[e]$ represent the logical time associated with event e belonging to process p. Depending on the type of event:

1. In the case of a Send event from process i to process j, the logical time is updated as follows:

$$vec_p[e] = vec_p[e] + 1 \tag{4}$$

2. For a Receive event where process i receives from process j, the logical time is updated as follows:

$$vec_p[e] = vec_p[e] + 1 \tag{5}$$

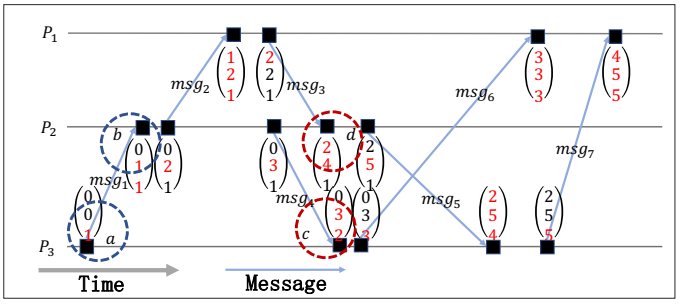


Fig. 7 Calculating the logical time using Vector Clock. Three processes are involved. The x-axis represents the time, whereas the blue line represents the process of communication.

3. Furthermore, for each process p, the logical time $vec_p[e]$ is updated as:

$$vec_p[e] = \max(vec_p[e], vec_p[j]) \tag{6}$$

The proposed discipline captures the communication dependency among processes. This discipline enhances the understanding of how events are interconnected within the context of logical time calculations.

We then define the communication dependency among processes based on two disciplines, which are described as follows with reference to Figure 7.

Discipline 1: For events a and b, b depends on a ($a \rightarrow b$). If $\forall i$, $1 \leqslant i \leqslant N$, then $V_a[i] \leqslant V_b[i]$. In Figure 7, events a and b are in blue circles. Given that $V_a[1] = 0 \leqslant 0 = V_a[1]$, $V_a[2] = 0 \leqslant 1 = V_a[2]$, $V_a[3] = 1 \leqslant 1 = V_a[3]$, we have $a \rightarrow b$ (i.e. $msg1_{send} \rightarrow msg1_{receive}$).

Discipline 2: Events c and d are concurrent events that do not depend on each other. If $\exists i$, $1 \leqslant i \leqslant N$, then $V_a[i] > V_b[i]$. Figure 7, events c and d are in red circles. $V_c[2] = 3 \leqslant 4 = V_d[2]$ and $V_c[3] = 2 > 1 = V_d[3]$, c and d are concurrent events.

Based on the extracted communication dependency, all communication events are ordered by logical time. A logical time-based communication events collection is performed to construct DAG in Section 5.3.1.

5.3 DAG of Communication State

The DAG is constructed using our proposed algorithm (Algorithm 1 in Section 5.3.1) based on the communication dependency described in Section 5.2. We introduce here our visual designs of the DAG (Section 5.3.2) as well as a new glyph called CS-Glyph for characterizing the process communication state, the visual layout, and the interactions.

5.3.1 DAG Construction

Based on the communication dependency of events (their logical time order) described in Section 5.2, we construct the DAG using Algorithm 1.

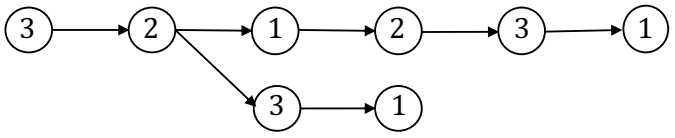


Fig. 8 The DAG of Figure 7 calculated by Algorithm 1. A node represents a process, the number in a node represents process id, and the links represent the messages between processes.

Figure 8 shows the DAG of communication events presented in Figure 7. A node represents a process, the number in a node represents the process ID, and the links represent the message between processes. The DAG clearly shows the communication latency among processes.

To further optimize the DAG, we introduce the load balance for processes. mc_p is defined as the sum of messages (sent and received) of process p during a period, whereas LB_p represents the load balance of p.

First, AD is calculated as

$$AD = \frac{\sum_{i=1}^{n} |mc_i - mc_{avg}|}{n} \tag{7}$$

where AD is the arithmetic average deviation, n is the number of processes communicated during this period, and mc_{avg} is equal to $\frac{1}{n}\sum_{i=1}^{n} mc_i$.

We calculate LB_p as

$$LB_p = \frac{|mc_p - mc_{avg}|}{AD} \tag{8}$$

5.3.2 *Visual Designs in DAG*

We designed CS-Glyph to show the communication state of processes in DAG. This glyph represents a node in DAG. As shown in Figure 9, the circular CS-Glyph consists of two parts, namely, an ellipse in the center and multiple bar charts in the border. The symbols in the CS-Glyph are defined as follows:

- *Ellipse in the center:* a process with an ID number.
- *Color of ellipse:* latency of compute node of a process.
- *Flatness of ellipse:* load balance of a process.
- *Greater flatness:* greater load imbalance of a process.
- *Bar charts inside the border:* messages received by a process.
- *Bar charts outside the border:* messages sent by a process.
- *Length of bar chart:* message transmission time.
- *Height of bar chart:* message size.
- *Color of bar chart:* latency of the process computed using Equation 1.

A hierarchical layout and edge-bounding of links are applied to DAG for a clearer display. As shown in Figure 9, the above designs can address the low rendering efficiency

Algorithm 1 DAG construction algorithm

Input: the set of communication events S (logical time in ascending order)
Output: DAG $G = (V, E)$
$process\ V = (pid, eventlist) \leftarrow \emptyset$;
$communication\ dependency\ E = edge(V_i, V_j) \leftarrow \emptyset$;
foreach ev in S **do**
  `// two types of event: Send and Receive`
  if ev is a Send event **then**
    if $\exists v \in V, v.pid = ev.pid$ **then**
      find the newest created node v_i (there may exist multiple nodes in V which satisfy the condition) such that $v_i.pid = ev.pid$, push ev into $v_i.eventlist$;
    end if
    else
      create a new node v, $v.pid = ev.pid$;
    end if
  end if
  else
    `// ev is a Receive event`
    if $\exists v_i \in V, v_i.pid = ev.pid, v_i.eventlist$ does not contain Send events **then**
      push ev into $v_i.eventlist$;
    end if
    else
      create a new node v, $v.pid = ev.pid$;
    end if
    if the Send event corresponding to ev is in $v_j.eventlist$ **then**
      create edge (V_i, V_j) adding into the E;
      `// edge` $V_i \rightarrow V_j$
    end if
  end if
end foreach

of large-scale process communication. In the vertical direction, the nodes of DAG are placed on different layers. In the horizontal direction, the nodes show a dependency relationship.

Some interactions are also offered to help users easily obtain the process communication state, hence facilitating their exploration of communication latency among processes. The interaction operation is discussed in detail as follows.

(1) **Overview and Zooming:** Users can effortlessly access the communication status information for the entire DAG by employing familiar actions like translating and

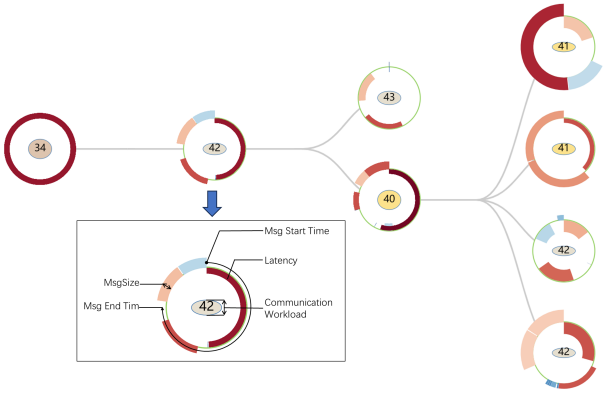


Fig. 9 Visual design of the CS-Glyph and layout of the DAG. The CS-Glyph is a circle with two parts of information, namely, an ellipse in the center and bar charts in the border. The ellipse represents the process with a process ID, and the bar charts represent the messages sent (outside the border) and received (inside the border).

zooming in/out. These interactions facilitate a holistic understanding of communication patterns and statuses across the system.

(2) **Process-Level Control:** For a more focused perspective, users have the ability to selectively view or conceal the communication status of specific processes. Achieved by simply double-clicking on a graph node, this action effectively folds or unfolds the connected nodes to the right, allowing users to tailor their view to their analytical needs.

(3) **Hover Interaction:** A seamless means of acquiring in-depth insights is through cursor hovering. By hovering over a node, users gain detailed information about the associated process. Simultaneously, this action also highlights the pertinent links, offering a comprehensive visualization of the communication dynamics surrounding that specific node.

These interactions enhance user engagement and empower efficient exploration of the communication status within the DAG, ultimately contributing to a more comprehensive understanding of the underlying data.

6 PCL Events Attribution Strategy

In this section, we analyze the characteristics of different PCL events causes. Furthermore, we summarize the attribution strategies (**R3**) and the optimization methods of different PCL events (**R4**).

A1 Poor process-to-processor mappings. This problem is caused by too much inter-node communication. Therefore, poor process-to-processor mapping occurs when DAG has many CS-Glyph pairs with ellipses that are colored differently. To address this problem, we present a view that helps users

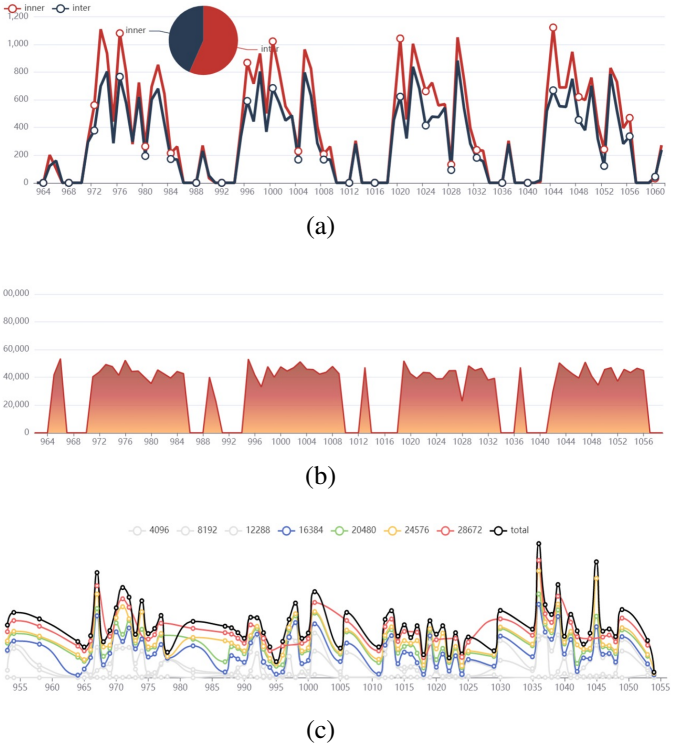


Fig. 10 (a) Mapping analysis view of a period. (b) Communication pattern analysis view of a period. (c) Background analysis view of a period.

analyze the mapping timing variation of their simulations. As shown in Figure 10(a), the mapping view consists of a line chart and a pie chart. The line chart shows the mapping for each duration, where the x-axis represents the duration, which can be defined by users, whereas the y-axis represents the number of inter- and intra-node communications. Meanwhile, the pie chart summarizes the mapping as a whole. The red component represents the count of inter-node communications. A high proportion of the red color denotes poor mapping.

O1 Optimize the node mapping scheme. To optimize the mapping, users can map the processes that frequently communicate with one another to the same physical node. Our system prompts users to minimize the communication transmission between physical nodes as much as possible to reduce the hop count of message transmission and the risk of congestion. The system can also recommend high-efficiency mapping by using the graph partitioning algorithm.

A2 Poor communication patterns. Generally, communication patterns can describe the process communication load balance. The flatness of ellipse in the center of CS-Glyph represents this information. A higher flatness means poor communication patterns.

As shown in Figure 10(b), we provide an area chart to display the communication load balance of the simulation. In this view, the x-axis represents the duration, whereas

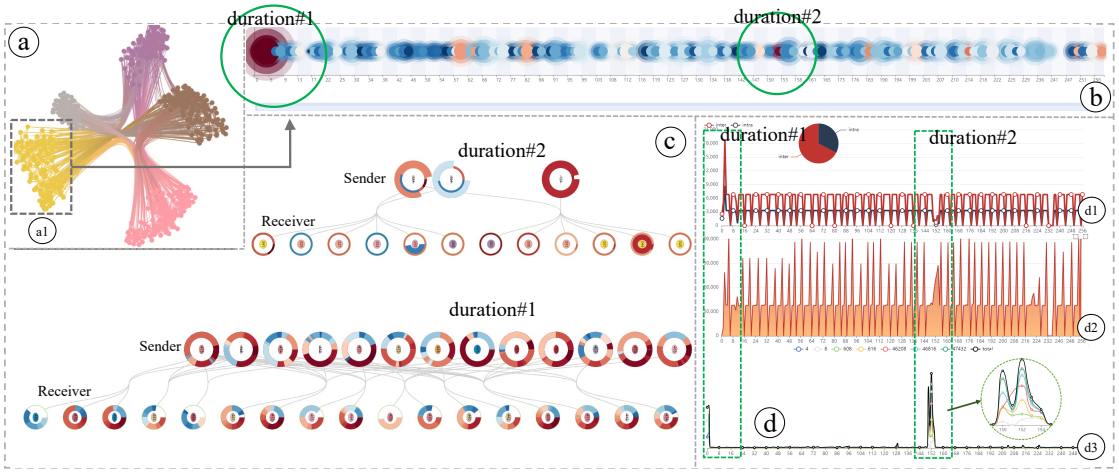


Fig. 11 Visualization of MiniFE running with 512 processes on TH-1A. (a) is the spatial view showing the process of communication of regions. (b) is the evolution view showing the temporal latency abstraction of a region. (c) is the DAG view showing the communication latency among processes in detail. (d) is the attribution view with three parts, namely, mapping analysis (d1), communication pattern analysis (d2), and background traffic analysis.

the y-axis represents the average communication load of a process, which we calculate using Equation 8. The area chart is rendered with a gradient color ranging from yellow to red, with red indicating that the communication load in the current duration is imbalanced.

O2 Optimize the simulation communication algorithm. Fixing a poor communication pattern requires modifying the communication algorithm of the simulation. Users can employ combined communication to avoid some poor point-to-point communication patterns, hence improving parallelism and further reducing the traffic for a single process.

A3 Background traffic. Background traffic varies by time. If the transmission time of the same size message varies dramatically in the inter-node communication, then we consider this transmission time to be influenced by background traffic. In the DAG, the bar chart with the same height across all CS-Glyphs has different colors.

The view in Figure 10(c) records the variance in the transmission time of different messages and reflects the influence of background traffic. In this view, the x-axis represents the duration, whereas the y-axis represents the average transmission time of different message sizes for each duration.

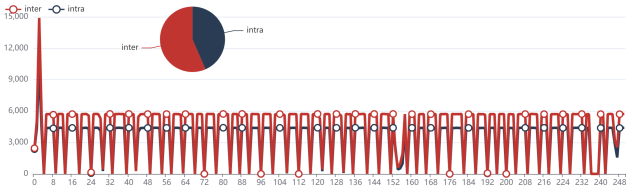


Fig. 12 Statistics of inter- and intra-node communication after remapping. The number of inter-node messages was reduced from 1385144 to 1159653.

We calculate and then normalize the average message transmission time. We also color the line as gray where the y value shows the least fluctuation. A greatly fluctuating y value indicates serious background traffic.

O3 Optimize the simulation running environment. When performance degradation is caused by background traffic, users do not need to modify the algorithm or communication mapping. Our system will prompt these users to just wait for the congestion to ease and run the simulation again.

7 EXPERIMENTAL RESULTS

We analyze in this section the causes of PCL events in two simulations. We invite three experts to evaluate the system and provide detailed feedback.

7.1 MiniFE

MiniFE is a proxy application for unstructured implicit finite element codes. We run this application with 512 processes on TH-1A. However, this program has poor performance. As shown in Figure 11, we use PCLVis to analyze the PCL events in MiniFE to help users improve the communication performance of this parallel program.

Figure 11(a) shows the 5 communication regions of MiniFE. The processes within each of these regions communicate with one another intensively. We select the yellow region from Figure 11(a1), which has many PCL events. Figure 11(b) shows the PCL temporal evolution of the yellow region. The color of durations #1 and #2 is unusually red, which confirms the presence of communication latency in these durations. Our system only has one DAG for users at the same time. Nevertheless, to better display the effect and

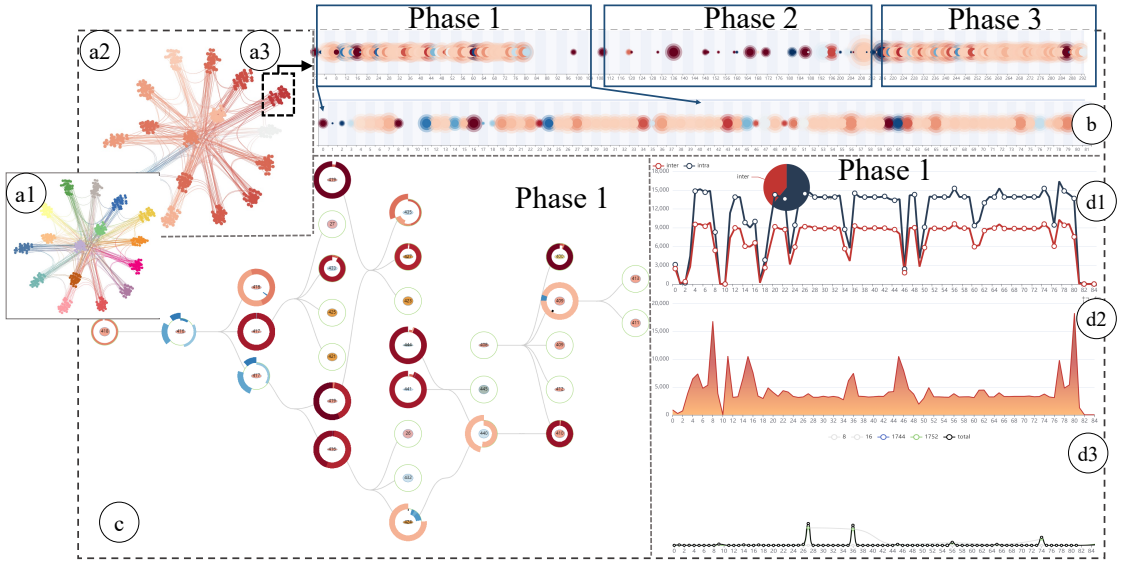


Fig. 13 Visualization of NPB CG running with 512 processes. (a1) shows the communication latency of regions, where each region is encoded by a certain color. (a2) is the latency mode of (a1). (a3) is a region with high latency. (b) is the latency evolution of region a3, which has three phases. (c) is the DAG graph of *Phase1*, which shows the communication latency among processes. (d) is the attribution view of *Phase1*. Three views are displayed to analyze the causes of PCL events, namely, the mapping (d1), communication pattern (d2), and background traffic (d3).

save space, Figure 11(c) flips the DAG for display and places the DAGs of durations #1 and duration #2 together.

Analysis of *Duration #1*:

To analyze the causes of PCL events in duration #1, we display the communication details of the process inside the yellow region during duration #1 in Figure 11(c-duration #1). The upper CS-Glyphs represent the sender, whereas the lower CS-Glyphs represent the receiver. The bar chart in each CS-Glyph is colored red, thereby indicating that a significant amount of time has been spent on communication. These CS-Glyphs have the same characteristics. Specifically, the ellipses of CS-Glyph pairs are flat and have different colors, which suggests that those processes belonging to different nodes communicate with one another intensively. We combine with Figure 11(d) for in-depth analysis. In Figure 11(d1), the proportion of inter-node communication is larger than that of intra-node communication, and the number of internode communications in duration #1 rapidly increases. The message transmission time is affected by background traffic when the messages need to be transmitted across inter-node links. Figure 11(d3) shows that the transmission time increases dramatically in duration#1, whereas Figure 11(d2) shows that the communication load balance of MiniFE has no significant impact on performance. Therefore, the PCL events in duration #1 are caused by poor process-to-processor mapping.

Apart from attribution, PCLVis can also provide some possible optimization methods. To address the poor process-to-processor mapping, we propose a remapping algorithm based on graph partition and then remap MiniFE to obtain a better mapping and to reduce the 225491 messages being transmitted across inter-node links (Figure 12).

Analysis of *Duration #2*:

Figure 11(c-duration#2) shows the communication dependency and state of PCL events in duration #2. Compared with Figure 11(c-duration#1), this view has fewer links, and the height of the bar charts is higher. Figure 11(d) shows that the number of communication decreases (Figure 11(d1)), but the transmission time of large messages greatly increases to 47462 bytes (Figure 11(d3)), hence suggesting that the transmission of large bytes in MiniFE leads to communication latency. Therefore, the PCL events in duration #2 are caused by background traffic. When the program is disturbed by background traffic, the system will prompt users to change the job node or adjust the simulation running time.

7.2 NAS Parallel Benchmarks Conjugate Gradient

We then analyze the PCL of **NAS parallel benchmarks conjugate gradient (NPB CG)** using PCLVis. These benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels. We use the conjugate gradient as our kernel, which we run with 512 processes on TH-1A. We easily identify the process mapping problem using PCLVis. To clearly illustrate other causes of latency,

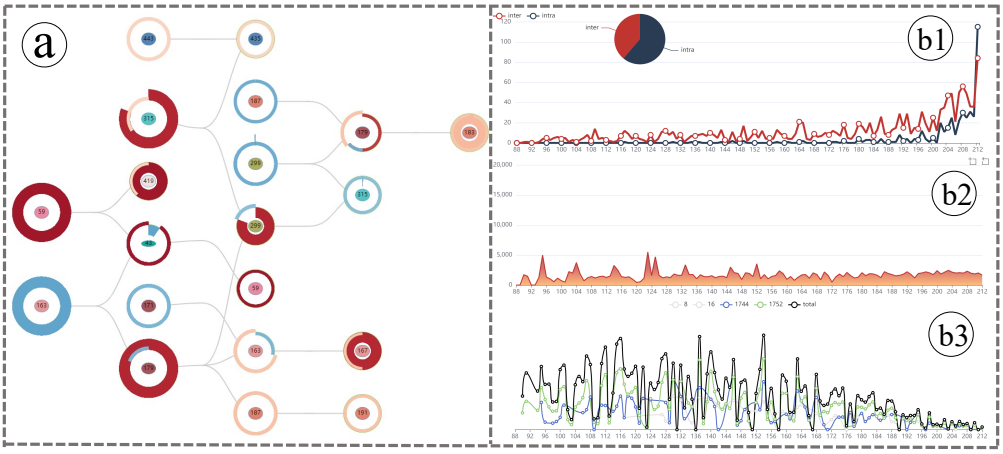


Fig. 14 Visualization of *Phase2* in Figure 13. (a) is the DAG graph of *Phase2*. Many CS-Glyphs have dark red bar charts, which indicate high communication latency. Three views are displayed to analyze the causes of PCL events, namely the mapping (b1), communication pattern (b2), and background traffic (b3).

we optimize the mapping using a graph-based mapping algorithm as a pre-processing step. Therefore, this section will not discuss the mapping problem in detail.

The visualization result is shown in Figure 13. As shown in Figure 13(a1), the 512 processes are divided into 16 communication regions. The communication among processes within each region is similar to that in others. We then switch to the latency mode (Figure 13(a2)), and find that the communication latency of one region (Figure 13(a3)) is higher than that of others. Therefore, we select the specific region with higher communication latency for further analysis.

We initially observe the temporal evolution of communication latency in region a3.Figure 13(b) shows 3 communication phases, namely, *Phase1*, *Phase2* and *Phase3*, which we analyze separately as follows.

Analysis of *Phase1*:

As shown in Figure 13(c), many CS-Glyphs in the DAG have a dark red color. To obtain additional information about the communication latency among different processes, we compare these dark red CS-Glyphs with the light-colored CS-Glyphs and find that the ellipses in the former are much flatter than those in the latter, thereby suggesting that the communication load of the dark red processes is very unbalanced. The DAG also suggests a potential problem with the communication pattern due to the highly unbalanced communication load.

To identify the cause of communication latency, we use the attribution view for further analysis. As shown in Figure 13(d1), the intra-node communication is much greater than the inter-node communication because we have already improved the process mapping. Meanwhile, Figure 13(d2) shows that the average load balance of processes greatly varies over time. Multiple peaks can be found in Figure 13(d2), which indicate a poor communication pattern. The lines in Figure 13(d3) change smoothly with a low value, which indicates low background traffic in this phase, thereby proving that communication latency is mainly caused by poor communication patterns.

Therefore, our system prompts the users to modify the simulated communication algorithm to improve the communication pattern.

Analysis of *Phase2*:

Figure 14(a) shows many CS-Glyphs in the DAG with a dark red color. We initially check the ellipses in these CS-Glyphs and find that these ellipses have low flatness, thereby confirming that the dark red processes have a balanced communication load. In other words, this particular phase exhibits a good communication pattern. We also examine the bar charts (representing messages) of the same height (representing message size) and find that messages of the same size have different colors, thereby suggesting that some background traffic may interfere with the communication in our simulation.

To identify the cause of communication latency, we use the attribution view for further analysis. We observe Figure 14(b2) directly because the process mapping has already been improved. We find that the average load balance of processes steadily varies over time and maintains a low value. This communication pattern can be considered good. Furthermore, the lines exhibit significant changes with high values as shown in Figure 14(b3), thereby confirming a high level of background traffic in this phase and further substantiating that the latency is primarily caused by background traffic.

Our system then prompts the users to just wait for the congestion to ease and run the simulation again. After analyzing *Phase3*, We find that it is similar to *Phase1*. Therefore, we do not present here the visualization results for *Phase3*.

8 Discussion

In this section, we have extended invitations to three domain experts to conduct an in-depth analysis of their simulations through the utilization of our PCLVis system. The synopsis of their feedback and our subsequent discussions pertaining to the experts' input are presented as follows:

8.1 Expert feedback on the PCLvis system

The Flowchart of PCLVis System

Like a traffic jam, all experts want to know "**where**", "**how**", "**what**", and "**why**". **Where** is the latency? **How** is the latency evolved? **What** happened in the processes with latency? **Why** does the latency occur?

Our flowchart has fully answered these four questions: the spatial view for locating latency; the evolution view for showing the evolution of the latency; the DAG view for analyzing the details; attribution view for the reason of the latency.

The Spatial View

All experts hope to find the worst latency region from their large-scale simulations on supercomputers. Also similar to the traffic jam, the latency would occur in a region, but not one process, because it can propagate from one process to another. For locating the latency region, all experts are happy with our spatial view design. Here is one of their comments: "It's so great. The complicated process-communication network with more than 1,000 processes has been instantly visualized into several logic regions. I can also find the worse latency region in the latency mode."

In relation to the clustering algorithms employed within the spatial view, the majority of experts express contentment with methodologies capable of categorizing processes based on their pertinence. Nonetheless, there exists an expert who articulates a desire for heightened temporal efficiency within the current algorithm. This viewpoint seamlessly converges with our forthcoming endeavors and serves as a focal point for our future pursuits.

The Evolution View

The experts also want to know the evolution of one latency region. This is because it can help to trace the source of the latency. The experts were satisfied to use the sliding window for finding the period of interest by observing the colored circles, which represent the latency. Meanwhile, they are also like our temporal latency abstraction strategy, which can successfully hide all unimportant temporal features. By using it, they can quickly locate the most important period for further analysis.

The DAG View

Experts also try to understand the details of the latency among several processes, like their dependencies, load balance, and so on. They highly rate our DAG view with the following 3 features: 1) It is a good idea to use DAG for visual analytics of communication dependencies; 2) Interactive exploration is a good choice for resolving the scalability problem; 3) The CS-Glyph has been well designed to show enough information for users, including load balance, send/receive messages, latency, and so on.

The Attribution View

Attribution is the most important one for users. All experts found the reasons for the latency in their large-scale simulations by using our PCLVis system.

- **Expert A**: from the result in Figure 1, I found that there are so many inter-node process communications in the attribution view ($d1$). This is a poor process-to-processor mapping. With guidance from the system-recommended graph division, I have successfully optimized his simulation. The ratio between intra-node communication and inter-node communication has been improved from $252545/281753$ to $325064/209234$.
- **Expert B**: Through the system's communication pattern analysis and DAG view, it is found that a large number of load-unbalance cases exist, which is the main reason for the communication latency of my simulation. This is difficult to detect by using existing tools. By checking and revising the simulation code for the unbalanced process IDs, the efficiency of my simulation has been improved. One possible suggestion is to list all possible breakpoints for users, although it may be out of the scope of this system.
- **Expert C**: My situation is that the efficiency of my simulation is unstable. I have checked my code again and again but found nothing. With the help of PCLVis, the reason is the communication latency caused by the fluctuation of background traffic.

8.2 Experts Feedback on Current System Limitations

Experts acknowledged the system's effective latency analysis, noting its assistance in resolving communication delays. However, they foresee greater convenience through real-time communication latency analysis. This enhancement will allow for easier use and analysis.

- **Expert**: At present, the PCLvis system can help me analyze and locate the latency region in my simulation. But it'd be even cooler if I could perform real-time analysis and latency region location on my running simulations.

9 Conclusion

In this paper, we proposed the PCLVis framework to visually analyze the causes of PCL in a large-scale simulation. First, we developed a spatial PCL event locating method. Second, we designed a process-correlation-tree-based spatial clustering algorithm to generate communication regions. Third, we designed a communication-dependency-based DAG that can help users interactively explore the communication latency among processes in a communication region. We also designed a new glyph called CS-glyph to show the communication state of each process. Before constructing the DAG, we proposed a sliding-window-based method to generate an abstraction of PCL events over time, which displays the evolution of communication latency in a region. Finally, we developed a PCL event attribution strategy to help users improve the efficiency of their simulations. Several simulations running on the supercomputer TH-1A were analyzed using the proposed PCLVis system. Users greatly improved the efficiency of their simulations by following the recommendations of our PCLVis system. Furthermore, the temporal efficiency of PCLVis necessitates augmentation, serving as the focal point of our imminent pursuits. We aspire to achieve real-time analysis of PCL events.

Declarations

Availability of data and materials

The NAS parallel benchmarks conjugate gradient and MiniFE software packages used in this study are both publicly available under open source licenses.

The NAS parallel benchmarks can be downloaded from the following website: [https://www.nas.nasa.gov/software/npb.html].

MiniFE can be downloaded from the following website: [https://computation.llnl.gov/projects/minife/].

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Acknowledgements

This work was partially supported by the National Key R&D Program of China under Grand No. 2021YFE0108400, and partly supported by the National Natural Science Foundation of China under Grant No. 62172294.

Authors' contributions

Chongke Bi, Xin Gao, Baofeng Fu, Yuheng Zhao, Siming Chen, Ying Zhao, Yunhai Wang contributed to this paper in the following ways:

Chongke Bi: led and supported the design and implementation of the research, provided funding and equipment support for the research, and at the same time guided and revised the writing of the paper, and put forward important opinions and suggestions.

Xin Gao: participated in the design and implementation of the research, the writing and revision of the paper, and the later maintenance of the program.

Baofeng Fu: responsible for the implementation and maintenance of the research, participated in the writing and modification of the paper, and the maintenance of the program later.

Yuheng Zhao: participated in the experimental design, data collection, and made major contributions to the later debugging of the program.

Siming Chen: made important contributions to the later stage debugging of the program, and made important comments and suggestions on the guidance and revision of the manuscript writing.

Ying Zhao: participated in the research design, guided and revised the manuscript writing and made important comments and suggestions.

Yunhai Wang: guide and revise the manuscript writing, put forward important opinions and suggestions, and guide the author to complete the writing and publication of the paper.

References

[1] Martinez B, Montón M, Vilajosana I, Prades JD. The Power of Models: Modeling Power Consumption for IoT Devices. *IEEE Sensors Journal*, 2015, 15(10): 5777–5789, doi:10.1109/JSEN.2015.2445094.

[2] Kesavan SP, Fujiwara T, Li JK, Ross C, Mubarak M, Carothers CD, Ross RB, Ma KL. A visual analytics framework for reviewing streaming performance data. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, 2020, 206–215, doi: 10.48550/arXiv.2001.09399.

[3] Kuhn N, Lochin E, Lacan J, Mehani O, Boreli R. CLIFT: A Cross-Layer InFormation Tool for latency analysis based on real satellite physical traces. In *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2014, 182–189, doi:10.1109/ASMS-SPSC.2014.6934542.

[4] Nickolayev OY, Roth PC, Reed DA. Real-time statistical clustering for event trace reduction. *The International Journal of Supercomputer Applications and High Performance Computing*, 1997, 11(2): 144–159, doi: 10.1177/109434209701100207.

[5] Aguilera G, Teller PJ, Taufer M, Wolf F. A systematic multistep methodology for performance analysis of communication traces of distributed applications based on hierarchical clustering. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, 2006, 388–395, doi: 10.5555/1898699.1898872.

[6] Gamblin T, De Supinski BR, Schulz M, Fowler R, Reed DA. Clustering performance data efficiently at massive scales. In *Proceedings of the 24th ACM International Conference on Supercomputing*, 2010, 243–252, doi: 10.1145/1810085.1810119.

[7] Bahmani A, Mueller F. Efficient clustering for ultra-scale application tracing. *Journal of Parallel and Distributed Computing*, 2016, 98: 25–39, doi: 10.1016/j.jpdc.2016.08.001.

[8] Knüpfer A. A new data compression technique for event based program traces. In *International Conference on Computational Science*, 2003, 956–965, doi: 10.1007/3-540-44863-2_94.

[9] Lee CW, Mendes C, Kalé LV. Towards scalable performance analysis and visualization through data reduction. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, 2008, 1–8, doi: 10.1109/IPDPS.2008.4536187.

[10] Noeth M, Ratn P, Mueller F, Schulz M, De Supinski BR. Scalatrace: scalable compression and replay of communication traces for high-performance computing. *Journal of Parallel and Distributed Computing*, 2009, 69(8): 696–710, doi: 10.1016/j.jpdc.2008.09.001.

[11] Zhai J, Hu J, Tang X, Ma X, Chen W. Cypress: combining static and dynamic analysis for top-down communication trace compression. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, 143–153, doi: 10.1109/SC.2014.17.

[12] Xu Q, Subhlok J, Zheng R, Voss S. Logicalization of communication traces from parallel execution. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, 34–43, doi: 10.1109/IISWC.2009.5306796.

[13] Wu X, Mueller F. Elastic and scalable tracing and accurate replay of non-deterministic events. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, 2013, 59–68, doi: 10.1145/2464996.2465001.

[14] Geimer M, Wolf F, Wylie BJ, Ábrahám E, Becker D, Mohr B. The scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 2010, 22(6): 702–719, doi: 10.1002/cpe.1556.

[15] De Kergommeaux JC, Stein B, Bernard PE. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 2000, 26(10): 1253–1274, doi: 10.1016/S0167-8191(00)00010-7.

[16] Shende SS, Malony AD. The TAU parallel performance system. *The International Journal of High Performance Computing Applications*, 2006, 20(2): 287–311, doi: 10.1177/1094342006064482.

[17] Adhianto L, Banerjee S, Fagan M, Krentel M, Marin G, Mellor-Crummey J, Tallent NR. HPCToolkit: tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 2010, 22(6): 685–701, doi: 10.1002/cpe.1553.

[18] Kaufmann S, Homer B. Craypat-cray X1 performance analysis tool. *Cray User Group (May 2003)*, 2003.

[19] Mohr B, Wolf F. Kojak–a tool set for automatic performance analysis of parallel programs. In *European Conference on Parallel Processing*, 2003, 1301–1304, doi: 10.1007/978-3-540-45209-6_177.

[20] Wilke J. Structural simulation toolkit (SST) DUMPI trace library. *Accessed: Mar*, 2020, 31.

[21] Geimer M, Wolf F, Wylie BJ, Mohr B. A scalable tool architecture for diagnosing wait states in massively parallel applications. *Parallel Computing*, 2009, 35(7): 375–388, doi: 10.1016/j.parco.2009.02.003.

[22] Hendriks M, Vaandrager FW. Reconstructing critical paths from execution traces. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, 2012, 524–531, doi: 10.1109/ICCSE.2012.78.

[23] Hendriks M, Verriet J, Basten T, Theelen B, Brassé M, Somers L. Analyzing execution traces: critical-path analysis and distance analysis. *International Journal on Software Tools for Technology Transfer*, 2017, 19(4): 487–510, doi: 10.1007/s10009-016-0436-z.

[24] Casas M, Badia RM, Labarta J. Automatic phase detection of MPI applications. *Parallel Computing: Architectures, Algorithms, and Applications*, 2007, 38: 129–136.

[25] Chetsa GLT, Lefevre L, Pierson JM, Stolf P, Da Costa G. A user friendly phase detection methodology for HPC systems' analysis. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, 118–125, doi: 10.1109/GreenCom-iThings-CPSCom.2013.43.

[26] Gonzalez J, Gimenez J, Labarta J. Performance analytics: understanding parallel applications using cluster and sequence analysis. In *Tools for High Performance Computing 2013*, Springer International Publishing2014, 1–17, doi: 10.1007/978-3-319-08144-1_1.

[27] Kunz T, Seuren MF. Fast detection of communication patterns in distributed executions. In *CASCON*, 1997, 12, doi: 10.5555/782010.782022.

[28] Köckerbauer T, Klausecker C, Kranzlmüller D. Scalable parallel debugging with G-Eclipse. In *Tools for High Performance Computing 2009 - Proceedings of the 3rd International Workshop on Parallel Tools for High Performance Computing*, 2010, 115–123, doi: 10.1007/978-3-642-11261-4_8.

[29] Wolf F, Mohr B, Dongarra J, Moore S. Automatic analysis of inefficiency patterns in parallel applications. *Concurrency*

and Computation: Practice and Experience, 2007, 19(11): 1481–1496, doi: 10.1002/cpe.1128.

[30] Isaacs KE, Gamblin T, Bhatele A, Bremer PT, Schulz M, Hamann B. Extracting logical structure and identifying stragglers in parallel execution traces. In *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel programming*, 2014, 397–398, doi: 10.1145/2692916.2555288.

[31] Isaacs KE, Bhatele A, Lifflander J, Böhme D, Gamblin T, Schulz M, Hamann B, Bremer PT. Recovering logical structure from charm++ event traces. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015, 1–12, doi: 10.1145/2807591.2807634.

[32] Isaacs KE, Gamblin T, Bhatele A, Schulz M, Hamann B, Bremer PT. Ordering traces logically to identify lateness in message passing programs. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 27(3): 829–840, doi: 10.1109/TPDS.2015.2417531.

[33] Nagel WE, Arnold A, Weber M, Hoppe HC, Solchenbach K. VAMPIR: visualization and analysis of MPI resources. *Supercomputer 63*, 1996, 12(1): 69–80, doi: 10.1016/0166-5316(94)00030-1.

[34] Heath MT, Finger JE. ParaGraph: a tool for visualizing performance of parallel programs. In *Second Workshop on Environments and Tools for Parallel Sci. Comput*, 1994, 221–230.

[35] Pillet V, Labarta J, Cortes T, Girona S. Paraver: a tool to visualize and analyze parallel code. *Proceedings of WoTUG-18: Transputer and Occam Developments*, 1995, 44(1): 17–31.

[36] Kalé LV, Kumar S, Zheng G, Lee CW. Scaling molecular dynamics to 3000 processors with projections: A performance analysis case study. In *International Conference on Computational Science*, 2003, 23–32, doi: 10.1007/3-540-44864-0_3.

[37] Huang D, Tory M, Staub-French S, Pottinger R. Visualization techniques for schedule comparison. *Computer Graphics Forum*, 2009, 28(3): 951–958, doi: 10.1111/j.1467-8659.2009.01441.x.

[38] Jo J, Huh J, Park J, Kim B, Seo J. LiveGantt: interactively visualizing a large manufacturing schedule. *IEEE Transactions on Visualization and Computer Graphics*, 2014, 20(12): 2329–2338, doi: 10.1109/TVCG.2014.2346454.

[39] Han Y, Rozga A, Dimitrova N, Abowd GD, Stasko J. Visual analysis of proximal temporal relationships of social and communicative behaviors. *Computer Graphics Forum*, 2015, 34(3): 51–60, doi: 10.1111/cgf.12617.

[40] Wongsuphasawat K, Gotz D. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2012, 18(12): 2659–2668, doi: 10.1109/TVCG.2012.225.

[41] Krstajic M, Bertini E, Keim D. Cloudlines: compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 2011, 17(12): 2432–2439, doi: 10.1109/TVCG.2011.179.

[42] Monroe M, Lan R, Lee H, Plaisant C, Shneiderman B. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2013, 19(12): 2227–2236, doi: 10.1109/TVCG.2013.200.

[43] Gotz D, Stavropoulos H. Decisionflow: visual analytics for high-dimensional temporal event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 2014, 20(12): 1783–1792, doi: 10.1109/TVCG.2014.2346682.

[44] Liu Z, Kerr B, Dontcheva M, Grover J, Hoffman M, Wilson A. Coreflow: extracting and visualizing branching patterns from event sequences. *Computer Graphics Forum*, 2017, 36(3): 527–538, doi: 10.1111/cgf.13208.

[45] Du F, Shneiderman B, Plaisant C, Malik S, Perer A. Coping with volume and variety in temporal event sequences: strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics*, 2016, 23(6): 1636–1649, doi: 10.1109/TVCG.2016.2539960.

[46] Guo S, Jin Z, Gotz D, Du F, Zha H, Cao N. Visual progression analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 2018, 25(1): 417–426, doi: 10.1109/TVCG.2018.2864885.

[47] Xu P, Mei H, Ren L, Chen W. ViDX: visual diagnostics of assembly line performance in smart factories. *IEEE Transactions on Visualization and Computer Graphics*, 2016, 23(1): 291–300, doi: 10.1109/TVCG.2016.2598664.

[48] Mu X, Xu K, Chen Q, Du F, Wang Y, Qu H. MOOCad: visual analysis of anomalous learning activities in massive open online courses. In *EuroVis (Short Papers)*, 2019, 91–95.

[49] Muelder C, Gygi F, Ma KL. Visual analysis of inter-process communication for large-scale parallel computing. *IEEE Transactions on Visualization and Computer Graphics*, 2009, 15(6): 1129–1136, doi: 10.1109/TVCG.2009.196.

[50] Sigovan C, Muelder CW, Ma KL. Visualizing large-scale parallel communication traces using a particle animation technique. *Computer Graphics Forum*, 2013, 32(3pt2): 141–150, doi: 10.1111/cgf.12101.

[51] Isaacs KE, Bremer PT, Jusufi I, Gamblin T, Bhatele A, Schulz M, Hamann B. Combing the communication hairball: visualizing parallel execution traces using logical time. *IEEE Transactions on Visualization and Computer Graphics*, 2014, 20(12): 2349–2358, doi: 10.1109/TVCG.2014.2346456.

[52] Zhang J, Yang C, Li Y, Chen L, Yuan X. LBVis: interactive dynamic load balancing visualization for parallel particle tracing. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, 2020, 91–95, doi: 10.1109/PacificVis48177.2020.1029.

[53] Landge AG, Levine JA, Bhatele A, Isaacs KE, Gamblin T, Schulz M, Langer SH, Bremer PT, Pascucci V. Visualizing network traffic to understand the performance of massively parallel simulations. *IEEE Transactions on Visualization and Computer Graphics*, 2012, 18(12): 2467–2476, doi: 10.1109/TVCG.2012.286.

[54] Bhatele A, Gamblin T, Isaacs KE, Gunney BT, Schulz M, Bremer PT, Hamann B. Novel views of performance data to

analyze large-scale adaptive applications. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, 1–11, doi: 10.1109/SC.2012.80.

[55] Grant RE, Pedretti KT, Gentile A. Overtime: a tool for analyzing performance variation due to network interference. In *Proceedings of the 3rd Workshop on Exascale MPI*, 2015, 1–10, doi: 10.1145/2831129.2831133.

[56] Bhatele A, Jain N, Livnat Y, Pascucci V, Bremer PT. Analyzing network health and congestion in dragonfly-based supercomputers. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE2016, 93–102, doi: 10.1109/IPDPS.2016.123.

[57] Fujiwara T, Malakar P, Reda K, Vishwanath V, Papka ME, Ma KL. A visual analytics system for optimizing communications in massively parallel applications. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017, 59–70, doi: 10.1109/VAST.2017.8585646.

[58] Li JK, Mubarak M, Ross RB, Carothers CD, Ma KL. Visual analytics techniques for exploring the design space of large-scale high-radix networks. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, 193–203, doi: 10.1109/CLUSTER.2017.26.

[59] Jha S, Patke A, Brandt J, Gentile A, Lim B, Showerman M, Bauer G, Kaplan L, Kalbarczyk Z, Kramer W, et al.. Measuring congestion in high-performance datacenter interconnects. *Links*, 2020, 8(8): 4.

[60] Taffet P, Mellor-Crummey J. Understanding congestion in high performance interconnection networks using sampling. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, 1–24, doi: 10.1145/3295500.3356168.

[61] Haghi P, Guo A, Geng T, Skjellum A, Herbordt MC. Workload imbalance in HPC applications: effect on performance of In-Network processing. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, 1–8, doi: 10.5555/645820.669727.

[62] Demmel J. Communication avoiding algorithms. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, 2012, 1942–2000, doi: 10.1109/SC.Companion.2012.351.

[63] Baicheng Y, Zhang Y, Limin X, Yi Z, Bing W, Yao S. Lpms: a low-cost topology-aware process mapping method for large-scale parallel applications on shared HPC systems. In *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, 2019, 1–10, doi: 10.1145/3339186.3339208.

[64] Yang X, Jenkins J, Mubarak M, Ross RB, Lan Z. Watch out for the bully! job interference study on dragonfly network. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, 750–760, doi: 10.1109/SC.2016.63.

[65] Tanenbaum AS. *Distributed systems principles and paradigms*. 2007.

[66] Johnson SC. Hierarchical clustering schemes. *Psychometrika*, 1967, 32(3): 241–254, doi: 10.1007/BF02289588.

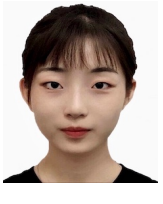
[67] Mattern F, et al.. *Virtual time and global states of distributed systems*, Univ., Department of Computer Science1988.

Author biography

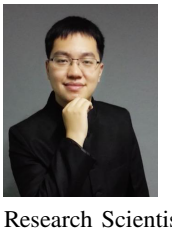
Chongke Bi received the B.Sc. (Eng.) degree and the M.Sc. (Eng.) degree from Shandong University, China, in 2004 and 2007, respectively, and the Ph.D. (Sci.) degree from the University of Tokyo, Japan, in 2012. After that, as a researcher in RIKEN, Japan, he was focused on research in the field of visual analysis of big data on supercomputer from 2012 to 2016. He is currently an associate Professor at Tianjin University. His current research interests include visualization and high performance computing.

Xin Gao received the B.Sc. (Sci.) degree from Qufu Normal University, China, in 2019, and the M.Sc. (Eng.) degree from Ludong University, China, in 2023. He is currently pursuing a Ph.D. (Eng.) degree from the Tianjin University, China. His current research interests include visualization and high performance computing.

Baofeng Fu received the B.Sc. (Eng.) degree from Qingdao University, China, in 2019, and the M.Sc. (Eng.) degree from Tianjin University, China, in 2022. Here research interests include visualization and high performance computing.

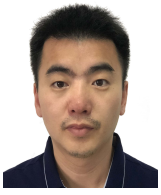


Yuheng Zhao Yuheng Zhao is currently a Ph.D student at the School of Data Science, Fudan University. Her main research interests include social media visualization and text visual analytics.



Siming Chen received the Ph.D. degree in computer and information technology from Peking University, Beijing, China, in 2017. He is currently an Associate Professor with the School of Data Science, Fudan University, Shanghai, China. Prior to this, he was a Research Scientist with Fraunhofer Institute IAIS and a Postdoc Researcher with the University of Bonn, Bonn, Germany. He has authored or coauthored more than 30 papers in visualization field, including more than 10 IEEE VIS/TVCG/EuroVis papers. His research interests include general visual analytics.

Ying Zhao is a Professor at the School of Computer Science and Engineering, Central South University, China. He received his Ph. D. degree in Computer Application Technology from Central South University in 2014. His research interests include visualization and visual analytics.



Yunhai Wang is a professor in the School of Computer Science and Technology at the Shandong University Qingdao campus. He leads the Interactive Data Exploration System (IDEAS) lab that aims to enhance people's ability to understand and communicate data through the design of automated visualization and visual analytics systems. His research focuses on the question of how can we automatically design an effective visualization that is best suited to pursue a given task on given input data.