IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 8, 2019, accepted April 2, 2019, date of publication April 15, 2019, date of current version May 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911125

On-Orbit Event-Based Conformance Checking Using GPU

NAN LI
University of Chinese Academy of Sciences, Beijing 100049, China
Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China
e-mail: linan@csu.ac.cn

ABSTRACT In the past few years, China's space science and application has entered the stage of the space station with large-scale space science experiments. The number of flight missions related to space science grows rapidly, meanwhile, the corresponding payloads from various fields of science become more complicated. This paper breaks through the traditional management mode in which the payloads of space science are usually monitored by space-ground telemetry links and proposes an approach to utilize workflow net models to monitor the experimental processes of on-orbit payloads in real time. This approach can effectively offload the pressure of space-ground communication as well as the pressure of ground monitor stations. In this approach, spectral graph clustering is used in decomposing a workflow net model when the model is too huge to be processed in parallel on one single GPU (graphic processing units) device, and hybrid parallel computing algorithms are designed to carry out conformance checking operations based on observed events from on-orbit payloads in real time. The algorithms are implemented with a gather-apply-scatter model in CUDA 9.0 (compute unified device architecture) on Jetson TX2i module and are benchmarked. The performance of the algorithms is acceptable for practical use on orbit.

INDEX TERMS Petri nets, parallel processing, parallel algorithms, partitioning algorithms, network theory (graphs).

I. INTRODUCTION

With the development of process modeling theory and data mining technology, process mining technology is widely applied in many fields, e.g. financial industry [1], manufacturing industry [2], business management [3]. Moreover, this technology is also applied in the field of scientific research to assist different scientists in coping with complicated experiment workflows (a.k.a. scientific workflows) [4], [5], such as Astrophysics, Heliophysics and Biomedicine. Scientific workflows (a.k.a. data-intensive workflows) are routinely used in the majority of data-driven research disciplines, to serve computational, experimental, and observational sciences. The related experiments are often exploiting rich and diverse data resources and are conducted on large-scale, parallel, distributed and heterogeneous computing platforms [6], [7]. As described in [8], scientific workflow system provides an easy-to-use environment for individual application scientists and helps them to create their own workflows; it helps the scientists to execute their workflows and view their results in real time; it simplifies the process of sharing and reusing workflows between the scientists; it enables scientists to track the provenance of the workflow execution results and the workflow creation steps. The research hotspots related to conventional scientific workflows include the scheduling strategies of workflow tasks and data movement, programming and usability of workflow tools, workflow execution monitoring and the validation of workflow executions [7]. Nevertheless, to the best of our knowledge, the scientific workflows for space science experiments on orbit have never been focused.

For the past few years, China's space science and application has entered the stage of the space station with large-scale space experiments. Thus, it poses a great challenge to the management of space science experiments, especially experiment process monitoring. Commercial-Off-The-Shelf (COTS) devices are competing with conventional radiation-hardened (rad-hard) components by offering the various advantages [9], such as "higher performance, ease of maintenance, faster/cheaper development, off-the-shelf

The associate editor coordinating the review of this manuscript and approving it for publication was Muhamamd Aleem.

development systems and processing/communications libraries," etc. Thus, COTS devices are widely used in space science experiments on orbit. An emerging trend toward COTS adoption is presented in European Space Agency (ESA) roadmaps [10], while National Aeronautics and Space Administration (NASA) is moving aggressively to accept COTS chips in space. However, COTS devices can be severely harmed by the harsh environmental conditions of high vacuum, extreme temperatures, and high levels of ionizing radiation, or the vibrations during launch [9]. Since a malfunction inside of one COTS device may interrupt the process of the corresponding space science experiment or have an uncertain impact on the experimental results, various methods should be studied and implemented to detect the discrepancies between the process of the on-orbit experiment and the process designed by scientists in real time. These monitoring or detecting methods can guarantee the validation of the experimental results on orbit and significantly improve the efficiency of tele-science. The provenance data [11] of the experiment on orbit can be compared with the corresponding experiment on the ground or used to aid the design of similar experiments for subsequent space missions. There are many space science experiments which have the potential demand, such as the experiment of thermocapillary convection in liquid bridge on Tiangong-2 [12], the scientific condensation experiment on TZ-1 [13], the experiment of solidification and crystal growth on SJ-10 [14], etc.

During the development process of space science payloads and related on-orbit supporting platforms, software developers produce many design materials in addition to software products, including documents, models, test vectors and results. It is significant to utilize these materials to supply the maintenance service when the corresponding devices or systems are running on orbit. Extending the ideas of conventional scientific workflow management, the design materials above can be fully utilized by process mining technology. The two most prominent process mining tasks are [3]:

i. Process discovery: extracting a process model from observed behavior recorded in an event log.
ii. Conformance checking: diagnosing and measuring discrepancies between observed behavior and modeled behavior.

In the phase of design and implementation on the ground, space science payloads and related on-orbit supporting platforms could be developed while being synchronized with the design of workflow net models. On one hand, the process discovery methods and tools can extract the workflow net model from the remote-control commands and operating statuses of payloads or platforms. On the other hand, designers can build workflow net models to guide the designing of payloads or platforms. When the payloads and supporting platforms are running on orbit, the corresponding workflow net models and conformance checking algorithms execute synchronously to monitor the events of space science experiments for scientists. The procedures are shown in **Figure 1** and **Figure 2**.

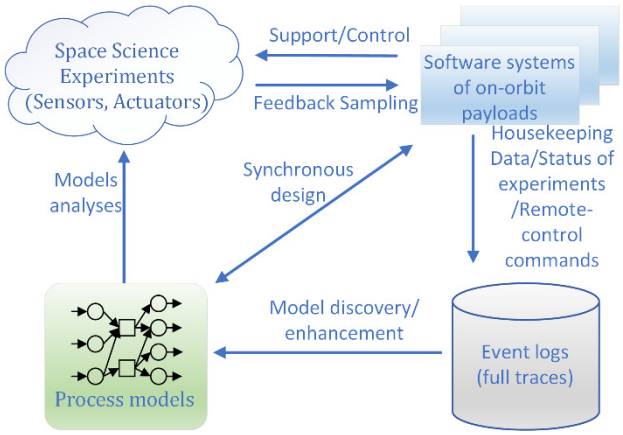

FIGURE 1. Design and implementation work on the ground.

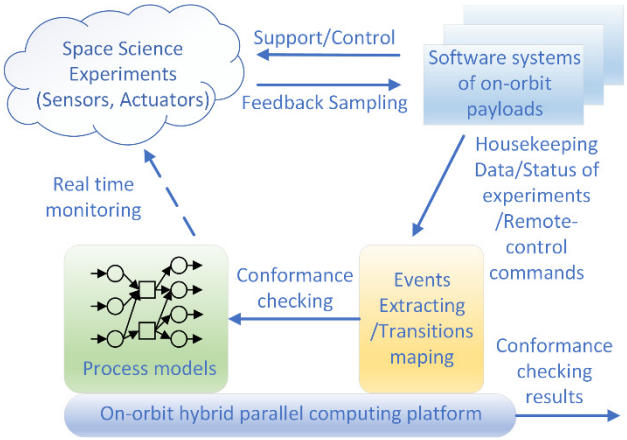

FIGURE 2. On-orbit monitoring operations.

This paper focuses on accelerated computing algorithms for conformance checking operations, on the assumption that workflow net models have been extracted in advance. It means that the accelerated computing algorithms are required to determine whether the observed behaviors ("control flows") which are captured in real time conform to the process model. In the scenarios on orbit, the workflow net models of space science experiments are given in advance. Only a portion of an observed event trace is captured from every sampling period, and there are multiple event traces changing in real time. This paper proposes a set of hybrid parallel computing algorithms for conformance checking based on workflow nets, supposing the duplicate transitions and invisible transitions of the workflow nets have been preprocessed on the ground. Workflow net models are treated as graph data processed with vertex-centric model which is shown in [15], [16]. As workflow net is a subclass of petri net, the input matrix and output matrix of workflow net model [17] and the related intermediate results are stored in texture memory and surface memory of GPU for high performance accessing as it is suggested in [18]. A single observed event (or multiple observed events) is captured by CPU and

is sent to GPU. Then the mark of the model is computed in an optimized paralleled manner to determine fitness of the currently observed event in the order of event occurrence. When the input and output matrixes with the related intermediate results are too huge to be processed in parallel on one single GPU device, spectral graph clustering method [19], [20] is used to partition the model into small components beforehand in a coarse-grained manner. Then all the model components are suggested to be dynamically loaded into different GPU devices before conformance checking tasks, so as to avoid unnecessary model data exchanging operations and enhance real time performance. This paper concentrates on the conformance checking algorithms on one single GPU device, and the algorithms are benchmarked on NVIDIA Jetson TX2i System-on-Module. The scheduling strategy of several model components among multiple GPU devices is not the focus of this paper, because it can be easily designed based on [21]–[22] and the decomposing method proposed in this paper.

The main contributions of this paper are as follows:

i. This paper proposes a feasible approach to monitor the working status of the on-orbit space science payloads in real time.
ii. This paper takes advantage of spectral graph clustering method to decompose the workflow net models for conformance checking. This method can adapt the huge net models to multiple GPU devices while keeping the degree of subnets' coupling at a low level. The small subnets resulting from decomposition are merged into a new subnet with proper size. Then these subnets with proper size can be assigned to different streams inside of one single GPU device or multiple GPU devices, so as to enhance the resources utilization ratio of each single GPU device. The implementation of the decomposition method of this paper is simpler than SESE decomposition method [22], [23], and the decomposition results of this method are more optimized than SESE decomposition method.
iii. Gather-Apply-Scatter model with parallel computing power of GPU is firstly utilized to accelerate the conformance checking operations in real time. The performance of conformance checking is significantly improved.

This paper is organized as follows. Section II presents related work. Section III illustrates the preliminaries about petri net and workflow net, spectral graph clustering theory, and graph processing strategy with GPU. The spectral graph clustering method for decomposing workflow net models is described in section IV. The vertex-centric conformance checking algorithms and certain programming advice in CUDA 9.0 are presented in section V, and pseudo codes of the algorithms are presented in section VIII. The algorithms above are benchmarked against the dataset of [24] in section VI. Section VII concludes this paper.

II. RELATED WORK

Over the last decades, the management systems inside of satellites and spacecraft have evolved from ''pre-programmed automata performing a priori known tasks and unable to react against unforeseen events, to smart embedded systems able to take pre-programmed decisions on event occurrence or able to react against context changes'' [25]. On the International Space Station (ISS), there is a software tool commonly referred to as ''Timeliner'', which is employed to ensure the scientific and engineering data generated by the systems onboard are handled properly [26]. During a ground Loss of Signal (LOS) data outage, payload developers have no insight into their experiments state, and benefit greatly from ''Timeliner'' running on ISS to perform telemetry monitoring and commanding operations. Besides, there are similar systems in European missions, which support On-Board Control Procedures (OBCPs) [27]. The on-board part of the OBCP execution environment consists of one or more OBCP engines which provide monitoring and control services for interfacing with the ground or onboard services. The monitoring approach of this paper is designed based on the combination of the above concepts and scientific workflow technologies, to serve for space science payloads on orbit. The state-of-the-art literature related to this paper is as follows.

In the perspective of fault management methods [28], the monitoring approach of this paper is similar to analytical redundancy [29], [30]. It is obviously that hardware redundancy on orbit is very costly in terms of size, weight, and complexity, and physically redundant is adopted in only a few of the most critical components. Thus, analytical redundancy is considered as a powerful alternative means of ensuring functional reliability. Analytical redundancy uses a mathematical model of the system together with some estimation techniques, and involves comparing the behavior of a system with a model of its expected behavior. The models corresponding to different analytical redundancies can be executed on a common hardware platform on orbit. It is just like the case in this paper that several models are running on GPU platform and monitoring the corresponding space science experiments on orbit. However, the monitoring approach of this paper can just detect the discrepancies between the control-flow model and the operating status of the related space science experiments. Fault isolation and recovery methods are beyond the scope of this paper.

In the perspective of runtime monitoring [31], the monitoring approach of this paper is based on a lightweight formal method (event-based conformance checking on workflow nets), compared to exhaustive verification methods such as model checking, which may be infeasible to apply for complex practical systems. Runtime monitoring involves a collection of approaches to evaluate formal specifications on traces of systems in order to verify the correctness of the system. Thus, runtime monitoring is also known as runtime verification, which is an area of formal methods that studies

the dynamic analysis of execution traces against formal specifications [32]. The runtime monitoring or runtime verification techniques play an important role in many application domains, such as distributed systems; hybrid and embedded systems; hardware; security & privacy; transactional information systems; contracts & policies; huge, unreliable or approximated domains [32]. These techniques are usually used to enhance the reliability of onboard or on-orbit systems, and related to health management in the field of aerospace. For instance, an approach of integrating responsive runtime monitoring of temporal logic system safety requirements with model-based diagnosis and Bayesian network-based probabilistic analysis, is proposed to implement the health management function for unmanned aerial systems [33]. The researchers of German Aerospace Center (DLR) present a formal approach for log-analysis and monitoring for the DLR ARTIS framework so as to implement intelligent system health management [31]. A conformance checking method based on colored Petri nets is used to implement a satellite network control protocol signaling dynamic conformance verification framework, which can capture runtime protocol signaling from protocol execution environment and assure that the interaction behaviors of protocol nodes conform to the expectation of protocol specification [34]. To the best of our knowledge, the combination of on-orbit runtime monitoring and process mining technology in our paper is a novel research subject.

In the perspective of process mining technology, the monitoring approach of this paper is designed based on conformance checking of workflow nets, so as to determine whether the observed behaviors fit the expected behavior model of an on-orbit experiment. Replay-based conformance checking and align-based conformance checking are two primary strategies to cope with the conformance checking problems based on conventional process models, such as petri nets, petri nets with data, workflow nets. Thus, the two strategies are introduced as follows.

i. Replay-based strategy takes one trace as input at a time and play ''token games'' to determine the maximal prefix of the trace that can be parsed by the model. Paper [35] proposes the initial fitness approaches based on replaying log traces in a model in order to assess whether a trace can fit a model. Then small or medium-sized problem instances are undertaken [3], while the problems of industrial size cannot be handled due to computation complexity and performance limitation. When divide-and-conquer strategies are applied in petri net (workflow net) model [22], workflow nets are decomposed into Single-Entry Single-Exit (SESE) components [36], [37] which satisfy the definition of valid decomposition [21]. Then the captured event sequences are projected into different components to carry out conformance checking operations in lower computation complexity. Paper [23] also presents the conformance checking strategy based on SESE decomposition to cope with real time observed events. Nonetheless, there is no implementation detail for that real time strategy. Moreover, the replay-based strategy is also used in paper [38], where inexistent (so-called negative) events are inserted into the traces in the log. The traces extended with negative events are then replayed on the model to carry out conformance checking.

ii. Align-based strategy identifies, for each trace in the log, the closest corresponding trace parsed by the model and compute an alignment that shows the positions of discrepancy between these two traces. The result is a set of pairs of traces with discrepancies. Each pair contains a trace in the log that cannot match exactly any single trace in the model, together with the corresponding closest trace(s) produced by the model [39]. Paper [40] formalizes the notion of alignments between the observed behavior in an event log and the modeled behavior in a process model, and considers an alignment as a pairwise comparison between executed activities in the trace and the activities allowed by the model. In paper [40], instances of a petri net are constructed to match the prefixes of an observed event trace, A^* algorithm is executed to find a pair of net instance and event trace with optimal cost. The approach [40] is not suitable for online conformance checking because of the computational complexity of A^* algorithm. Formal language theory is used to speed up conformance checking operations in paper [41]. In paper [41], an Integer Linear Programming (ILP) algorithm is designed to find the solution from marking equation of petri net, while maximizing the similarity between the Parikh vectors of the solution and the observed event trace. Then the observed event trace is filtered and is used to compute an approximate alignment. The approach [41] requires a full firing sequence of the model and a full observed event trace as input, and it makes a compromise between computation complexity and quality. Paper [42] improve [40] in two aspects, pruning the search space of A^* algorithm with the estimated cost based on the current prefix and the incoming activity, proposing partially revert strategy to cope with disabled transitions in a petri net. Then an incremental framework is proposed for online conformance checking based on event streams. However, it is still difficult to determine an appropriate set of parameters (such as threshold values for pruning the search space of A^* algorithm, the size of the partially revert window) to guarantee the performance of the framework [42] can satisfy real-time constraint, while taking precision into consideration. The method proposed in paper [39] relies on the construction of an event structure from the model, an event structure from the log, and the computation of a synchronized product between these two event structures, from which a set of differences are extracted and verbalized. The computation of the partially synchronized product is

also based on A^* algorithm. The construction of event structures makes the approach [39] more suitable for offline conformance checking. Decomposition strategy [21] is used with align-based conformance checking in paper [43], and a new fitness metric is presented to facilitate the aggregation of conformance results from the costs of alignment based on subcomponents to an overall conformance result.

iii. The combination of replay-based strategy and align-based strategy is used in papers [44], [45]. The problem of process conformance checking is transformed to trace consistency analysis (based on trace dependence graphs) between the input traces and the corresponding reference traces of the process in paper [44]. The exhaustive computation of finding the reference trace is alleviated by process decomposition and trace replaying. Nonetheless, the computation complexity makes the approach [44] unsuitable for online scenarios. In paper [45], a bounded petri net is converted into a labeled transition system by constructing reachability graph of the petri net, and the cost value of each transition inside of the system is zero. Then regions are computed inside of the transition system, and extra transitions with positive cost values are added into the transition system to indicate deviations in conformance checking. The extended transition system is constructed beforehand, the concept of which is similar to align-based strategy. The observed event traces are replayed on the extended transition system to compute conformance checking online in a suboptimal manner. Meanwhile, it is noteworthy that the extended transition system occupies a lot of memory space.

Besides, there are a few research works based on declarative models, such as papers [46]–[48], in which the proposed approaches cope with constraint-based process models in the offline manner due to the computational complexity. With the development of data mining technology, MapReduce is used to accelerate conformance checking operations in papers [49], [50].

To recap, the align-based approaches are usually computationally more expensive compared with the replay-based approaches, and as the primary viewpoint of the process models, ''control flow'' is the research hotspot. While considering the real-time constraints of monitoring and the resource constraint of hardware on orbit, the monitoring approach of our paper is based on replay-based strategy with the fitness of ''control flow'' as focus. GPU devices are used instead of distributed systems, to accelerate the conformance checking operations. Moreover, it is notable that SESE decomposition is the dominating decomposition method in conformance checking. However, the use of SESE decomposition method [22], [23] has some limitation, because it cannot partition workflow nets into ''balanced'' subparts while considering the coupling degree of the subparts in according to the detailed analysis in subsection III.B and VIII.A of this paper.

III. PRELIMINARIES

A. PETRI NETS AND WORKFLOW NETS

1) MODEL DEFINITION

According to papers [22], [54], the basic definitions related to Petri net and event are presented as follows.

Definition 1 (Petri Net [22], [54]): A Petri net is a 3-tuple $PN = (P, T, F)$, where P is the set of places, T is the set of transitions, and $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the set of flows. Besides, a Petri net is also a directed bipartite graph, where the set of vertexes is $P \cup T$, the set of directed arcs (edges) is F. For a vertex v of a Petri net, $\cdot v$ is the predecessor set of v, with $\cdot v = \{v' | (v', v) \in F$, and $v \cdot$ is the successor set of v, with $v \cdot = \{v' | (v, v') \in F$.

Definition 2 (Petri Net Semantics [22], [54]): Let $PN = (P, T, F)$ be a Petri net. A marking M is a multiset (Bag) of places, i.e. $M \in Bag(P)$. A transition $t \in T$ is enabled in a marking M, denoted as $(PN, M)[t >$, if $\cdot t \leq M$. Firing transition t in M, denoted as $(PN, M)[t > (PN, M')$, results in a new marking $M' = M - \cdot t + t \cdot$, i.e., tokens are removed from $\cdot t$ and added to $t \cdot$. The firing rules of mark is related to input matrix and output matrix, which are defined in Definition 10.

Definition 3 (Transition Sequence [22]): A transition sequence $\sigma^t = \langle t_1, t_2, \ldots, t_n \rangle \in$ *Kleene closure*(T) of Petri net PN is represented as $(PN, M)[\sigma^t > (PN, M')$. It indicates that when the transitions of σ^t is fired in sequence, there is a set of markings $M_0, M_1, \ldots, M_n$ such that $M_0 = M$, $M_n = M'$ and $(PN, M_{i-1})[t_i > (PN, M_i)$ for $0 < i \leq n$. A marking M' is reachable from M if there exists a trace σ^t such that $(PN, M)[\sigma^t > (PN, M')$.

Definition 4 (Event, Trace, Event Log [3], [22]): Let $\mathcal{E}$ be the event universe, indicates the set of all possible event identifiers. Events may have various attributes, e.g., a timestamp, an activity, etc. A single event $e \in \mathcal{E}$, is usually corresponding to a transition in Petri Net. A trace σ is finite sequence of events, with $\sigma \in$ *Kleene closure*$(\mathcal{E})$. An event log is a multiset (Bag) of traces, i.e. $L = Bag($*Kleene closure*$(\mathcal{E}))$.

Definition 5 (Labeled Petri Net [3], [22]): A labeled Petri net $PN = (P, T, F, l, A)$ is a Petri net (P, T, F) , where $\mathcal{A}$ is a set of activity labels, $l \in T \nrightarrow \mathcal{A}$ is a labeling function, $A \subseteq \mathcal{A}$ is some universe of activity labels. Multiple transitions with the same activity label are duplicate transitions. A transition t with $l(t) = \tau$ is unobservable and is an invisible transition.

Definition 6 (Activity Sequence [22]): A sequence $\sigma_v = \langle a_1, a_2, \ldots, a_n \rangle \in$ *universe of Kleene closure*$(\mathcal{A})$ is an activity sequence of the Petri net PN, represented as $(PN, M)[\sigma_v \triangleright (PN, M')$ if and only if there is a transition sequence $\sigma^t = \langle t_1, t_2, \ldots, t_m \rangle \in$ *Kleene closure*(T) in PN such that $(PN, M)[\sigma^t > (PN, M')$ and $\langle l(t_i) | l(t_i) \neq \tau, 0 < i \leq m \rangle = \sigma_v$. The corner mark ''$v$'' of σ_v indicates that activities are visible.

Definition 7 (Workflow Net [3], [22]): A workflow net is a particular type of Petri net, denoted as $WN = (P, T, F, l, p_{source}, p_{sink})$, where p_{source} is a special place without incoming arcs, $\cdot p_{source} = \emptyset$, and p_{sink} is a special place without outgoing arcs, $p_{sink} \cdot = \emptyset$. WN is strongly connected.

Let M_{source} be the initial mark of WN, M_{sink} be the final mark of WN.

Definition 8 (Fitting Trace of Workflow Net): A trace $\sigma_v = \langle a_1, a_2, \ldots, a_n \rangle \in$ *universe of Kleene closure*$(\mathcal{A})$ fits a system $SN = (WN, M_{source}, M_{sink})$ if and only if a full firing sequence $(WN, M_{source})[\sigma^t > (WN, M_{sink})$ can be found such that $(WN, M_{source})[\sigma_v \triangleright (WN, M_{sink})$, where $\sigma^t = \langle t_1, t_2, \ldots, t_m \rangle \in$ *Kleene closure*(T), $\langle l(t_i) | l(t_i) \neq \tau, 0 < i \leq m \rangle = \sigma_v$.

Definition 9 (Free Choice [3]): A petri net model is free choice if any two transitions sharing an input place have identical input sets, i.e., $\cdot t_1 \cap \cdot t_2 \neq \emptyset$ implies $\cdot t_1 = \cdot t_2$ for any $t_1, t_2 \in T$.

2) MATRIX ANALYSIS

Definition 10 (Input Matrix, Output Matrix [17]): The input matrix is denoted as D^-. It is a $(n_t \times n_p)$ matrix, whose generic element d_{ij}^- is equal to the number of directed arcs from place p_j to transition t_i $(0 < i \leq n_t, 0 < j \leq n_p)$. d_{ij}^- is corresponding to $\cdot t_i$ with Definition 1. Similarly, the output matrix is denoted as D^+. It is a $(n_t \times n_p)$ matrix, whose generic element d_{ij}^+ is equal to the number of directed arcs from transition t_i to place p_j $(0 < i \leq n_t, 0 < j \leq n_p)$. d_{ij}^+ is corresponding to $t_i \cdot$ with Definition 1. n_t is the number of transitions inside the model, and n_p is the number of places inside the model.

Petri Net Semantics can be indicated by matrix manipulations as the following example.

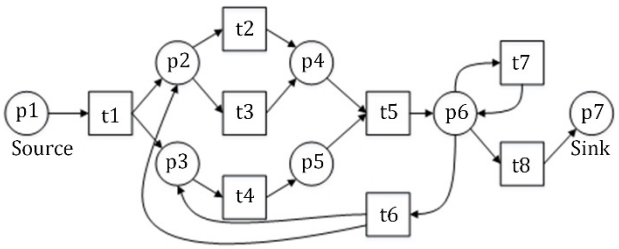

FIGURE 3. A workflow net with one initial token in p1.

$$D^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad D^+ = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 4. The input matrix and output matrix based on Figure 3.

Based on **Figure 3**, the input matrix and output matrix can be derived in **Figure 4**. The row index is $t1 \Rightarrow t8$ from top to bottom, and the column index is $p1 \Rightarrow p7$ from left to right.

Let $V(e_j)$ be a n_t-dimensional row vector in which all the entries equal to 0 except entry e_j $(0 < j \leq n_t)$ equals to 1. n_t is the number of transitions in workflow net. The vector is put into the following equation to simulate the firing rules of Definition 2. Entry e_j is corresponding to the transition t_j of workflow net. $(PN, M)[t_j > (PN, M')$, if $M \geq V(t_j)D^-$, where M is a n_p- dimensional row vector in which the entries indicate the number of tokens in each place. The corresponding equation is (1).

$$\begin{aligned} M' &= M - V(t_j)D^- + V(t_j)D^+ \\ &= M + V(t_j)(D^+ - D^-) \end{aligned} \tag{1}$$

Given a workflow net, a constant matrix D is obtained from $(D^+ - D^-)$. Taking a closer look at the one-step loop between $t7$ and $p6$ in **Figure 3** and the corresponding matrix D, it is clear that $d_{76}^+ - d_{76}^- = 0$ (the indices of D starts with 1 logically). So, the matrix D is insensitive to the event corresponding to $t7$, and it may generate some uncertainty in conformance checking when handling one-step loops like this. For the convenience of design, this paper calculates the equation in the form $M' = M - V(t_j)D^- + V(t_j)D^+$ to check whether the current marking M enables the firing of t_j. The multiplications such as $V(t_j)D^-$ and $V(t_j)D^+$ are replaced by search operations with t_j as the input. Given a workflow net, let M_0 be the initial marking of the model, and $t_i \rightarrow t_j \rightarrow t_k \rightarrow t_l \rightarrow t_j$ be a firing sequence (Definition 3), M_{cur} is the current marking after the firings. This paper calculates each transition in order and in paralleled manner in subsection V.C based on (2).

$$\begin{aligned} M_{cur} = M_0 &- V(t_i)D^- + V(t_i)D^+ \cdots - V(t_k)D^- \\ &+ V(t_k)D^+ \cdots - V(t_j)D^- \\ &+ V(t_j)D^+ \end{aligned} \tag{2}$$

B. PETRI NET DECOMPOSITION STRATEGIES

1) THE AIM OF DECOMPOSITION FOR CONFORMANCE CHECKING

The concept of divide-and-conquer is usually used in algorithm acceleration, and it also works in conformance checking. In according to Definition 17 of paper [21], a valid decomposition for conformance checking of petri net must meet the following requirements. The resulting subnets must "agree" on the original labeling function. Each place must appear in precisely one of the subnets, and each invisible transition must appear in precisely one of the subnets. Moreover, if there are multiple transitions with the same label, each of them must appear in precisely one of the subnets. Only unique visible transitions can be shared among different subnets. If the model is petri net with data [55], the data dependency should be considered in decomposition.

After decomposition, observed events of a sequence are transformed into transitions and projected into different subnets of the original petri net model. Then conformance checking is executed in the context of subnets in lower computational complexity. The result of the whole conformance checking is based on the results from the corresponding subnets. In an ideal state, the original petri net model should be decomposed into "balanced" subnets with less shared transitions, and these subnets can be processed in parallel.

On one hand, ''balanced'' means that the complexity or the size of each subnet approximates the same value, and the total time cost by conformance checking operations of these subnets in parallel can be effectively decreased. On the other hand, the scheduling overhead among different subnets can be effectively decreased when there are less shared transitions. In real time scenarios, the captured events are usually a part of a full trace, and these events are usually related to a limited number of subnets due to Principle of Locality. After these events are transformed into the transitions of the whole net model, the whole time consumed for conformance checking is determined by projecting operations which dispatch the transitions to the corresponding subnets, as well as conformance checking calculations in subnets. In the case of more shared transitions, more different subnets are tightly coupled with each other. It means that more subnets have to be scheduled to carry out conformance checking operations when the observed event sequence is captured in the current period, and the advantage of locality of events is not effectively utilized. Thus, the decomposition strategy for real-time conformance checking should consider the ''balanced'' factor, ''coupling'' factor as well as Principle of Locality at the same time.

Refined Process Structure Tree (RPST) based decomposition of workflow net is proposed in papers [22]–[24], taking advantage of Single-Entry Single-Exit (SESE) characteristic of RPST to reduce the number of shared transitions among subnets. However, the decomposition strategy [22]–[24] is affected by the structure of graph corresponding to the original petri net model, in addition to the valid decomposition constraints above. Thus, it is difficult to keep subnets ''balanced'' in an optimized loose-coupling status. Then the performance of conformance checking is often hindered by complicated subnets. There is a detailed analysis about RPST based decomposition (a.k.a. SESE decomposition) for integrity of this paper in subsection VIII.A.

Therefore, this paper utilizes spectral graph clustering method and merging strategies to decompose the conformance checking of workflow net so as to make the size of each subnet or each group of subnets suitable to be processed by one single GPU device, while considering the ''balanced'' factor, ''coupling'' factor as well as locality of the captured events.

2) SPECTRAL GRAPH CLUSTERING THEORY

Since workflow net is also a directed bipartite graph as shown in Definition 1, this paper transforms the directed bipartite graph into an undirected graph. Then spectral graph clustering method can be utilized to decompose the transformed net models. The theory of spectral graph clustering or spectral graph partitioning is widely applied in many fields, such as text categorization [56], image processing [57], and analyses of gene expression data [58]. This subsection introduces several basic notations and formulas of this theory [19], [20]. Then, the details about the procedure of decomposing is illustrated in section IV.

Let $G = (V, E)$ be an undirected graph with n vertexes, where V is a vertex set $V = \{v_1, v_2, \ldots v_n\}$, E is an edge set $E = \{\langle v_i, v_j\rangle | v_i, v_j \in V\}$. Let ω_{ij} be the weight of $\langle v_i, v_j\rangle$, and $\omega_{ij} = \omega_{ji}$. Let W be the corresponding adjacency matrix.

$$W_{ij} = \begin{cases} \omega_{ij}, & if\ \langle v_i, v_j\rangle \in E \\ 0, & otherwise \end{cases} \tag{3}$$

When the vertex set V is partitioned into two subsets V_1 and V_2, the *cut* between them is defined as follow. There is a coefficient ''1/2'', because W_{ij} is calculated twice in the undirected graph.

$$cut\ (V_1, V_2) = \frac{1}{2} \times \sum_{v_i \in V_1, v_j \in V_2} W_{ij} \tag{4}$$

An extended version of *cut* for k vertex subsets is defined here. The mark ''$i < j$'' in (5) is used to prevent the double counting of the vertex subsets.

$$cut\ (V_1, V_2, \ldots, V_k) = \sum_{i<j, 1\le i, j\le k} cut\left(V_i, V_j\right) \tag{5}$$

Let $\mathfrak{D}$ be the diagonal ''degree'' matrix of G, with $\mathfrak{D}_{ii} = \sum_{j=1}^{n} W_{ij}$. Based on [59], the *Ncut* is defined as follow, where $\bar{V}_i$ is the complement of V_i in graph G, $Vol\ (V_i) = \sum_{v_j \in V_i} \mathfrak{D}_{jj}$.

$$Ncut\ (V_1, V_2, \ldots, V_k) = \sum_{i=1}^{k} \frac{cut(V_i, \bar{V}_i)}{Vol(V_i)} \tag{6}$$

The *Ncut* minimization problem is to partition the vertex of G into ''reasonably large'' groups while keeping the edges between different groups have very low weights. Given two different partitions with the same *cut* value (when $k = 2$), *Ncut* is smaller for the more balanced partitioning. From random walks point of view, the partition in this manner will have the property that the random walk does not have many opportunities to jump between clusters [20].

Although the *Ncut* minimization problem is NP-complete, the real relaxation of the problem can be transformed into a generalized eigenvalue problem as [20].

$$Lz = \lambda \mathfrak{D} z \tag{7}$$

where L is the Laplacian matrix of G, $L = D - W$. z is a generalized eigenvector related to eigenvalue λ. To partition the vertex set V into k subsets, it needs to compute the first k smallest generalized eigenvalues corresponding to the first k generalized eigenvectors $z_1, \ldots z_k$ of (7). Let $Z \in \mathcal{R}^{n\times k}$ ($\mathcal{R}$ indicates real numbers.) be the matrix containing the vectors $z_1, \ldots z_k$ as columns. For $i = 1, \ldots, n$, let $r_i \in \mathcal{R}^k$ be the vector corresponding to the i-th row of Z. After clustering the points $(r_i)_{i=1,\ldots,n}$ in $\mathcal{R}^k$ with the k-means algorithm, the result subsets $V_1, \ldots, V_k$ is determined.

There are various methods for choosing the number k of subsets, but no one of them can solve problem perfectly [20].

Hence, this paper utilizes the eigenvector corresponding to the 2nd smallest eigenvalue λ_2 to bipartition G in a recursive manner [19].

C. GRAPH PROCESSING MODELS USING GPU

Due to massive degree of parallelism and the high memory access bandwidth, GPU technology has been widely utilized in big data processing acceleration, where graph processing is a typical application. This paper treats a whole workflow net or one decomposed subnet of a whole workflow net inside one single GPU device as a directed bipartite graph and concentrates on conformance checking acceleration with graph processing models.

$$W = \begin{bmatrix} a & b & 0 & 0 \\ 0 & c & d & 0 \\ e & 0 & f & g \\ 0 & h & 0 & s \end{bmatrix}, \; C = [0 \quad 1 \quad 1 \quad 2 \quad 0 \quad 2 \quad 3 \quad 1 \quad 3]$$
$$R = [0 \quad 2 \quad 4 \quad 7 \quad 9]$$
$$Value_{array} = [a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad s]$$

FIGURE 5. Example CSR representation of adjacency matrix W: Column-indices array C, row offsets array R, non-zero values $Value_{array}$.

In order to improve the utilization ratio of the memory space in GPU while keeping graph data easy to be accessed, data layout of graph should be compact and regular. There are four typical data layout models [15], [16]: Adjacency Matrix and Adjacency List, Vector Graph (V-Graph), Compressed Sparse Row (CSR). Adjacency Matrix and CSR are used in this paper to store workflow net models and other related data. The adjacency matrix is a square matrix W. A non-zero element ω_{ij} of the matrix indicates there is an edge from the i-th vertex to the j-th vertex in an unweighted graph. A non-zero element ω_{ij} stands for the weight of the edge in a weighted graph. In order to achieve both compact storage and regular memory access, some graph algorithms make use of the CSR format. As illustrated in **Figure 5**, the column-indices array C is formed from the set of the adjacency lists W_i concatenated into a single array of m integers. For example, $W_0 = \langle a, b\rangle$, $W_1 = \langle c, d\rangle$. The row offsets array R contains $n + 1$ integers, and entry $R[i] (0 \leq i < n)$ is the index of the adjacency list W_i in C. $Value_{array}$ is used to store the non-zero values of W, and the position of non-zero values inside W can be computed by R and C. $R[n]$ is the length of C and $Value_{array}$. It is worthy of notice that the indices of R, W, C, and $Value_{array}$ used here are usually start with 0 while being compatible with the addresses of physical memory.

Vertex-centric model is frequently-used in existing graph systems on GPU devices. Programmers need to define a few functions that are executed on each individual vertex. There are two popular parallel graph programming models, namely Gather-Apply-Scatter (GAS) and Bulk Synchronous Parallel (BSP) [16]. GAS model is used in this paper to accelerate conformance checking operations. The operations occur in three phases:

Gather Phase: Each vertex aggregates the information from the adjacent vertices and edges by using the user-defined gather function.

Apply Phase: Each vertex updates its state using the gather result. The values or statuses of the vertex is updated in this phase by calling the apply function.

Scatter Phase: The values or statuses of the vertex is scattered to its adjacent vertices and edges.

IV. WORKFLOW NET DECOMPOSING BASED ON SPECTRAL GRAPH CLUSTERING METHOD

A. COPING WITH DUPLICATE TRANSITIONS AND INVISIBLE TRANSITIONS

There are several non-deterministic factors affecting detection of non-fitting situations in conformance checking: invisible or duplicate transitions of petri net (workflow net) models, incorrectly captured events or abnormal behaviors. As it is analyzed in [22], the duplicate transitions and invisible transitions (Definition 5) need to be pre-processed in order to alleviate the negative effects in valid conformance checking decompositions. This paper suggests that the duplicate transitions and invisible transitions of the workflow net models are eliminated by pre-processing methods on the ground. For example, the more detailed operating status of the monitored payloads can be added into the housekeeping data to distinguish the duplicate transitions or match the invisible transitions. It means that the context containing each duplicate transition or each invisible transition can be uniquely determined. Then there is a one-to-one correspondence between the transitions of the workflow net models and the observed events in conformance checking operations. Since the number of the non-deterministic factors affecting conformance checking is decreased, the precision of conformance checking is enhanced.

B. DECOMPOSING WORKFLOW NET WITH NCUT

In this paper, one workflow net is transformed into an undirected bipartite graph (namely G) in the following manner. It is supposed that the weight of each directed edge of the directed bipartite graph is 1. If there are two directed edges $v_i \to v_j$ and $v_j \to v_i$ between v_i and v_j $(i, j > 0, i \neq j$, such as $p6, t7$ in **Figure 3**), they are merged into $\langle v_i, v_j\rangle$ inside of G, and ω_{ij} is the sum of the weights of $v_i \to v_j$ and $v_j \to v_i$. If there is only one directed edge $v_i \to v_j$ or $v_j \to v_i$ between v_i and v_j, then it is transformed into $\langle v_i, v_j\rangle$, and ω_{ij} is the weight of $v_i \to v_j$ or $v_j \to v_i$. Based on Definition 10 and (3), the adjacency matrix of the undirected graph G transformed from a workflow net is:

$$W = \begin{bmatrix} 0 & (D^- + D^+) \\ (D^- + D^+)^T & 0 \end{bmatrix} \tag{8}$$

It is shown that the eigenvector corresponding to the 2nd smallest eigenvalue of the generalized eigenvalue problem (7) provides a real relaxation to the discrete optimization problem of finding the minimum $Ncut$ producing two subnets. The problem (7) can be further transformed into the singular value

decomposition (SVD) problem while taking advantage of the property of undirected bipartite graph [19]. Here is the computation of diagonal "degree" matrix $\mathfrak{D}$.

$$\mathfrak{D} = \begin{bmatrix} \mathfrak{D}_1 & 0 \\ 0 & \mathfrak{D}_2 \end{bmatrix} \tag{9}$$

In the equation above, $\mathfrak{D}_1(i,i) = \sum_{j=1}^{n_p} (D^- + D^+)_{ij}$, $\mathfrak{D}_2(i,i) = \sum_{j=1}^{n_t} (D^- + D^+)_{ij}^T$, where n_p is the number of places, and n_t is the number of transitions in according to Definition 10.

Based on the property of Laplacian matrix, there is the following equation:

$$\begin{aligned} L &= \mathfrak{D} - W \\ &= \begin{bmatrix} \mathfrak{D}_1 & -(D^- + D^+) \\ -(D^- + D^+)^T & \mathfrak{D}_2 \end{bmatrix} \end{aligned} \tag{10}$$

Then equation (7) can be rewritten as follow [19], where $\mathbb{A} = (D^- + D^+)$.

$$\begin{bmatrix} \mathfrak{D}_1 & -\mathbb{A} \\ -\mathbb{A}^T & \mathfrak{D}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \mathfrak{D}_1 & 0 \\ 0 & \mathfrak{D}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{11}$$

Since there is no isolated vertex inside the undirected bipartite graph corresponding to workflow net, $\mathfrak{D}_1$ and $\mathfrak{D}_2$ are nonsingular. Then the equations set corresponding to (11) is as follow, where $u = {\mathfrak{D}_1}^{1/2} x$ and $v = {\mathfrak{D}_2}^{1/2} y$.

$$\begin{cases} {\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2} v = (1-\lambda) u \\ {\mathfrak{D}_2}^{-1/2} \mathbb{A}^T {\mathfrak{D}_1}^{-1/2} u = (1-\lambda) v \end{cases} \tag{12}$$

Then u is substituted with v in equations set (12), the following equations can be derived.

$$\begin{aligned} &\left({\mathfrak{D}_2}^{-1/2} \mathbb{A}^T {\mathfrak{D}_1}^{-1/2}\right) \left({\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2}\right) v \\ &\quad = (1-\lambda)^2 v \Longrightarrow \\ &\left({\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2}\right)^T \left({\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2}\right) v \\ &\quad = (1-\lambda)^2 v \end{aligned} \tag{13}$$

It is clear that the second equation of (13) defines the singular value decomposition (SVD) of the matrix $\left({\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2}\right)$. u and v are the left and right singular vectors respectively with $(1-\lambda)$ as the corresponding singular value. Thus the 2nd largest value of $(1-\lambda)$ should be computed with the corresponding left and right singular vectors, namely, u_2 and v_2. Then the second eigenvector of L is given by:

$$z_2 = \begin{bmatrix} {\mathfrak{D}_1}^{-1/2} u_2 \\ {\mathfrak{D}_2}^{-1/2} v_2 \end{bmatrix} \tag{14}$$

From the above, the procedure for partitioning the undirected bipartite graph of workflow net is as shown in the Algorithm 1.

In according to the constraint of valid decomposition for conformance checking of workflow net, unique visible transitions are shared by the boundaries of different subnets. Thus, when G is partitioned into two parts, the corresponding

Algorithm 1 Bipartition

Input:

<1> The undirected bipartite graph transformed from workflow net, denoted as G.

Output:

<1> The vertex set V_1 and vertex set V_2. Let V be the vertex set of G, $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$.

<1> Let $\mathbb{A}$ be the matrix with n_t rows and n_p columns, $\mathbb{A} = \left(D^- + D^+\right)$, where, D^- and D^+ is the input matrix and output matrix of the corresponding workflow net being partitioned in according to Definition 10.

<2> Form $\mathbb{A}_n = \left({\mathfrak{D}_1}^{-1/2} \mathbb{A} {\mathfrak{D}_2}^{-1/2}\right)$.

<3> Compute the singular vectors of $\mathbb{A}_n$, namely u_2 and v_2 which are corresponding to the 2nd largest singular value of $\mathbb{A}_n$.

<4> Compute z_2 by (14).

<5> Carry out k-means algorithm on the column vector z_2. The two-value of each element inside the result column vector indicates that the corresponding vertex of G belongs to V_1 or V_2.

<6> Output V_1 and V_2.

subnets can be founded by the following method, namely shared-transitions complementation. Based on Definition 1, each edge of G has a "transition" vertex as one endpoint and a "place" vertex as the other endpoint. For example, there is only one edge $\langle v_i, v_j \rangle$ between V_1 and V_2, $v_i \in V_1$, $v_j \in V_2$. If the type of v_i is "place" in the original workflow net, the shared transition between the subnet corresponding to V_1 and the subnet corresponding to V_2 is v_j. It means that the subnet corresponding to V_1 should be constructed based on the copy of v_j and V_1. Without loss of generality, let ${V_1}^p$ be the set of vertexes inside of V_1, and the type of the vertexes is "place"; let ${V_1}^t$ be the set of vertexes inside of V_1, and the type of the vertexes is "transition"; let ${V_1}^{p\_t}$ be the set of transitions which are directly connected to the places of ${V_1}^p$ by the arcs (edges) inside of the original workflow net. Then shared-transitions complementation of the subnet corresponding to V_1 is to add the shared transitions of the set $({V_1}^{p\_t} - {V_1}^t)$ to V_1 and construct the subnet conforming to the constraint of valid decomposition.

According to the implementation of conformance checking algorithms in subsection V.E, the number of transitions and the number of places inside each subnet resulting from decomposition must not be greater than the maximum number of the concurrent threads in one single GPU device respectively. Then the whole procedure of decomposing workflow net with *Ncut* is as shown in the Algorithm 2.

There are three cases worthy of analyses.

Firstly, the above algorithm cannot decompose the subnets, of which $n_t > \mathbb{N}_{thread}$ *and* $n_p = 1$. In this case, we suggest

Algorithm 2 Decomposition With *Ncut*

Input:

<1> The workflow net WN, where, D^- and D^+ is the input matrix and output matrix of the workflow net being partitioned in according to Definition 10.

<2> The maximum number of the concurrent threads in one single GPU device, namely $\mathbb{N}_{thread}$. $\mathbb{N}_{thread} \geq 1$.

Output:

<1> The collection of subnets $\mathbb{D}_{WN} = \{SN_1, \ldots, SN_k\}$ corresponding to WN. SN_k is a system net defined by [22].

<1> $SN = WN$, $\mathbb{D}_{WN} = \{SN\}$.

<2> Let n_t be the number of transitions inside SN^i, n_p be the number of places inside SN^i.

<3> Search for one subnet $SN^i \in \mathbb{D}_{WN}$, while matching the condition that ($n_t > \mathbb{N}_{thread}$ *and* $n_p > 1$) or ($n_p > \mathbb{N}_{thread}$ *and* $n_t > 1$), go to step <4>. Otherwise, there is no subnet matching the condition, go to step <8>.

<4> Carry out transformation from SN^i to the undirected bipartite graph G.

<5> Call Bipartition Algorithm to bipartition G, output the vertex set V_1 and V_2.

<6> Construct subnets ${SN^i}_1$ and ${SN^i}_2$ in according to V_1 and V_2 with shared-transitions complementation method.

<7> $\mathbb{D}_{WN} = \mathbb{D}_{WN} - \{SN^i\} + \{{SN^i}_1\} + \{SN^i_2\}$. Go to step <3>.

<8> Output $\mathbb{D}_{WN} = \{SN_1, \ldots, SN_k\}$.

to increase $\mathbb{N}_{thread}$ or modify the WN model. For example, a large number of the original transitions connecting to the same original place can be divided into several groups while the number of the original transitions in each group is not greater than $\mathbb{N}_{thread}$. The original transitions of each group converge at one newly-inserted place, and the place connects to one newly-inserted transition. Then, several newly-inserted transitions converge at the original place.

Secondly, when $n_p = 2, n_t = 1, \mathbb{N}_{thread} = 1$, it is the corresponding workflow net consisting of two places (source place and sink place) which is connected by one transition. In the case when $n_p > \mathbb{N}_{thread}, n_t = 1, \mathbb{N}_{thread} > 1$, the soundness property [60] of the corresponding workflow net is violated. The workflow nets with violated-soundness property are out of the scope of this paper. Besides, the subnets created in the procedure of decomposing one workflow net are impossible to have the status "$n_p > \mathbb{N}_{thread}, n_t = 1$", and the proof is as follow. Without loss of generality, we suppose that there is one subnet with one transition and two places in the procedure of the workflow net decomposition. Since the subnet is not an independent workflow net which does not need to be decomposed, it has only one transition shared with other subnets. In this circumstance, the shared transition cannot be fired, or redundant tokens are produced by the firing of the shared transition. The soundness property is violated again.

Thirdly, the decomposition in this algorithm is different from the conventional spectral graph clustering, because step <6> may add shared transitions to the partitioned graphs to construct subnets. Then within one single subnet, the *Ncut* calculation is based on the weight of the edges connecting to the added transitions and the edges inside of the corresponding partitioned graph. Moreover, the subnets created by the decomposition based on the same workflow net (or subnet) may be different every time, because the k-means clustering method inside of the Bipartition Algorithm selects random centers in the partition vector z_2.

C. MERGING SMALL SUBNETS

In subsection IV.B, the Algorithm Decomposition with *Ncut* cannot avoid creating micro subnets. It means that the number of transitions and the number of places inside these subnets are far less than the maximum number of the concurrent threads in one single GPU device respectively. In order to enhance the use ratio of one single GPU device, Algorithm Greedy Merging is suggested to be executed after the Algorithm Decomposition with *Ncut*.

With this algorithm, the small subsets with the same shared transitions can be merged into big subnets, so as to utilize the concurrent threads of one single GPU device (explained in subsection V.E). The Greedy Merging Algorithm is as shown in the Algorithm 3. The objective of the algorithm is to make the number of transitions "n_t" and the number of places "n_p" inside the merged subnets approximate $\mathbb{N}_{thread}$ respectively as much as possible, while keeping the difference between n_t and n_p is as small as possible. The noteworthy feature of the algorithm is that the merging operations can decrease the number of shared transitions.

The subnets inside $\mathbb{D}_{merge}$ can be assigned to different streams of one single GPU device or multiple GPU devices to further improve the degree of parallelism (explained in subsection V.E). There is an example of decomposition based on **Figure 6**. The components of the corresponding RPST in **Figure 7** are tightly coupled with each other. It is clear that the decomposition based on this RPST will result in several polygons and bridges based on the analysis in VIII.A. On the contrary, when the algorithms of this section are utilized, the results are illustrated in **Table 1**. Since the feature information in the singular vectors of the corresponding undirected bipartite graph is not distinct, the decomposing result based on spectral graph clustering method may be barely satisfactory due to the random centers selection of k-means method. However, Algorithm Greedy Merging is an effective remedy.

V. VERTEX-CENTRIC CONFORMANCE CHECKING

A. TRANSFORMATION OF OBSERVED EVENTS

Since duplicate transitions and invisible transitions are eliminated by pre-processing in subsection IV.A on the ground,

Algorithm 3 Greedy Merging

Input:

<1> $\mathbb{D}_{WN} = \{SN_1, \ldots, SN_k\}$ which is corresponding to WN. SN_k is a system net defined by [22].

<2> The maximum number of the concurrent threads in one single GPU device, namely $\mathbb{N}_{thread}$.

Output:

<1> The collection of subnets $\mathbb{D}_{merge} = \{SN_1, \ldots, SN_j\}$ corresponding to WN.

<1> $\mathbb{D}_{merge} = \{\emptyset\}$.

<2> Search for one SN_i from $\mathbb{D}_{WN}$, while matching the condition that the number of vertexes inside SN_i is maximum among the subnets of $\mathbb{D}_{WN}$.

<3> Let $\mathbb{D}_{WN\_Adjacent}$ be the collection of subnets which share transitions with SN_i. If $\mathbb{D}_{WN\_Adjacent}$ is $\{\emptyset\}$, $\mathbb{D}_{merge} = \mathbb{D}_{merge} + \{SN_i\}$, $\mathbb{D}_{WN} = \mathbb{D}_{WN} - \{SN_i\}$, go to step <7>. Otherwise, go to step <4>.

<4> Let $\mathbb{D}_{WN\_Candidate}$ be the collection of 4-tuples which contain the result of step <5>. $\mathbb{D}_{WN\_Candidate} = \{\emptyset\}$. Each of the first three elements of the 4-tuple is used to store one single subnet respectively, and the last element is used to store a real number.

<5> For each SN_n inside $\mathbb{D}_{WN\_Adjacent}$, let SN_{merge} be the subnet which is the merging result of SN_i and SN_n (deleting duplicate transitions). If the number of places "n_p" inside SN_{merge} is not greater than $\mathbb{N}_{thread}$ and the number of transitions "n_t" inside SN_{merge} is not greater than $\mathbb{N}_{thread}$, $\mathbb{D}_{WN\_Candidate} = \mathbb{D}_{WN\_Candidate} + \{(SN_{merge}, SN_i, SN_n, n_p/n_t)\}$. Then, all elements of $\mathbb{D}_{WN\_Adjacent}$ are processed. If $\mathbb{D}_{WN\_Candidate}$ is $\{\emptyset\}$, $\mathbb{D}_{merge} = \mathbb{D}_{merge} + \{SN_i\}$, $\mathbb{D}_{WN} = \mathbb{D}_{WN} - \{SN_i\}$, go to step <7>. Otherwise, go to step <6>.

<6> Search for the 4-tuple inside $\mathbb{D}_{WN\_Candidate}$, while matching the condition that the real number of the tuple approximates 1 as much as possible. If there are several tuples matching the condition, select one of them arbitrarily. Let $({SN_{merge}}^{sel}, {SN_i}^{sel}, {SN_n}^{sel}, (n_p/n_t)^{sel})$ be the selected 4-tuple. $\mathbb{D}_{WN} = \mathbb{D}_{WN} - \{{SN_i}^{sel}\} - \{{SN_n}^{sel}\} + \{{SN_{merge}}^{sel}\}$.

<7> If $\mathbb{D}_{WN}$ is not $\{\emptyset\}$, go to step <2>. Otherwise, go to step <8>.

<8> Output $\mathbb{D}_{merge} = \{SN_1, \ldots, SN_j\}$.

the non-deterministic factors caused by duplicate transitions and invisible transitions are avoided. Without loss of generality, it is supposed that the observed events are extracted from housekeeping data or status and are mapped to the row indexes of the matrixes D^- and D^+ for further computing

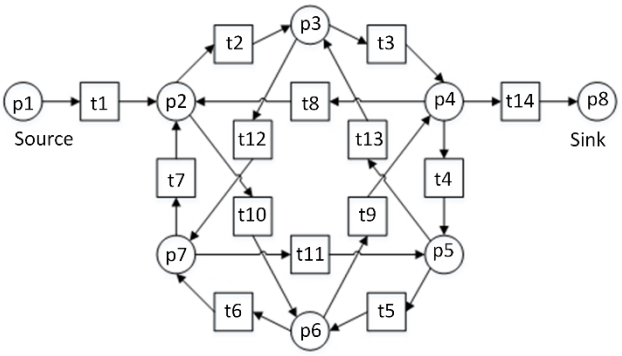

FIGURE 6. A workflow net containing a 4-connected graph structure.

(transformation from events to transitions). The procedure of the above transformation is omitted in this paper.

Then the results from transformation of the observed events on orbit are stored in $Buffer_{transition}$ (defined in subsection V.B) in the order of event occurrence. The following conformance checking algorithms are only coping with visible and non-duplicate transitions.

B. DATA LAYOUT OF WORKFLOW NET

In this paper, Jetson TX2i module with CUDA 9.0 is used as the hybrid computing platform. Supposing that a workflow net model has been decomposed to fit into a single Jetson TX2i module in subsection IV, the remaining of the section does not distinguish between the whole workflow net and the subparts of the net. The workflow net model is processed by the modified java program[1] on the host PC and transformed into an XML file. Then the XML file of workflow net model is loaded into Jetson TX2i module and processed by tinyXML2[2] to generate the matrixes D^- and D^+ from subsection III.A.2). The map from places of the workflow net to the column indexes of the above matrixes is founded, and the map from transitions of the workflow net to the row indexes of the above matrixes is also founded. The matrixes D^- and D^+ are stored in CSR format and the corresponding $Value_{array}$ (III.C) are omitted, because the non-zero values are either 0 or 1. For example, the CSR format of the matrixes from **Figure 4** are presented in **Figure 8**. It is worthy of notice that the indices of D^- and D^+ start with 1 logically, and the indices of C and R start with 0 in physical memory.

Because the workflow net model is not changed frequently on orbit, it may be beneficial to hold the structure of the model in constant memory [18]. Reading data from texture or surface memory instead of global memory can have several performance benefits. Therefore, C^-, C^+, R^-, R^+ are loaded into the texture memory of GPU. There is a small trick that C^- and C^+ are merged into a 2D cuArray[3] C_{array}, while R^- and R^+ are merged into another 2D cuArray R_{array}, such as **Figure 9**. Then different threads can calculate out the row

[1] https://github.com/jbpt/codebase

[2] https://github.com/leethomason/tinyxml2

[3] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

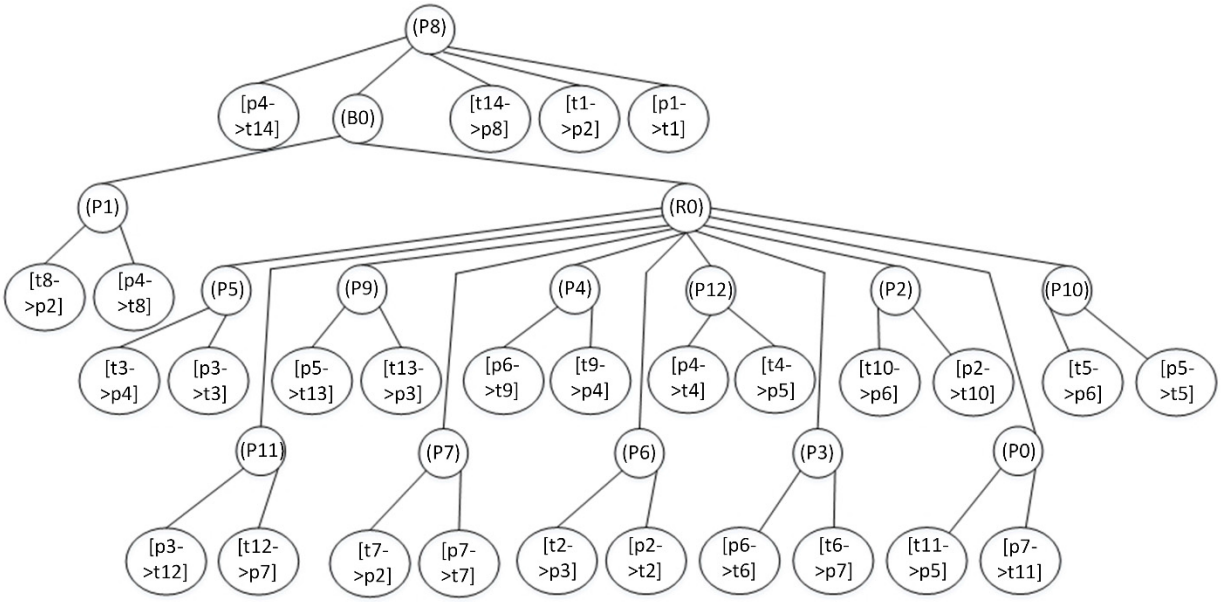

FIGURE 7. The RPST created by SESE decomposition method based on Figure 6.

TABLE 1. The decomposing result of Figure 6 using spectral graph clustering method.

Test Cases ($\mathbb{N}_{thread} = 6$)		Result Subnets (indicated as vertex set)
i.	Algorithm Decomposition with *Ncut* gives one proper result without merging.	{*t5, t6, t9, t10, p6*}, {*t4, t5, t11, t13, p5*}, {*t3, t4, t8, t9, t14, p4, p8*}, {*t1, t2, t7, t8, t10, p1, p2*}, {*t2, t3, t12, t13, p3*}, {*t6, t7, t11, t12, p7*}
ii.	Algorithm Decomposition with *Ncut* gives one barely satisfactory result without merging.	{*t1, p1*}, {*t3, t4, t8, t9, t14, p4, p8*}, {*t1, t2, t7, t8, t10, p2*}, {*t5, t6, t9, t10, p6*}, {*t4, t5, t11, t13, p5*}, {*t6, t7, t11, t12, p7*}, {*t2, t3, t12, t13, p3*}
iii.	Algorithm Greedy Merging is executed to process the result of ii.	{*t3, t4, t8, t9, t14, p4, p8*}, {*t1, t2, t7, t8, t10, p1, p2*}, {*t5, t6, t9, t10, p6*}, {*t4, t5, t11, t13, p5*}, {*t6, t7, t11, t12, p7*}, {*t2, t3, t12, t13, p3*}

$$C^{-} = [0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 5]$$
$$R^{-} = [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 6 \quad 7 \quad 8 \quad 9]$$
$$C^{+} = [1 \quad 2 \quad 3 \quad 3 \quad 4 \quad 5 \quad 1 \quad 2 \quad 5 \quad 6]$$
$$R^{+} = [0 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9 \quad 10]$$

FIGURE 8. CSR format of the matrixes from Figure 4.

$$C_{array} = \begin{bmatrix} 0 & 1 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 0 \\ 1 & 2 & 3 & 3 & 4 & 5 & 1 & 2 & 5 & 6 \end{bmatrix}$$
$$R_{array} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 6 & 7 & 8 & 9 \\ 0 & 2 & 3 & 4 & 5 & 6 & 8 & 9 & 10 \end{bmatrix}$$

FIGURE 9. Merged arrays of Figure 8.

indices of C_{array} and R_{array} directly based on their thread IDs without "if-else" branches. This trick can reduce branch divergence of the algorithms in subsection V.C.

The mark of the workflow net is stored in the form of a row vector $Vector_{mark}$ with $Size_{place}$ elements, where $Size_{place}$ is the number of places in the model. $Vector_{mark}$ is exchanged between CPU and GPU, and it is stored in the surface memory of GPU for performance benefits. In the scenarios on orbit, housekeeping data is captured by periodic sampling, and it means the number of observed events per sample may be larger than one. Therefore, there is a row vector $Buffer_{transition}$ used as temporary storage for the transitions extracted from the housekeeping data, and the elements of the row vector will be processed by the conformance checking algorithms. The max number of elements of $Buffer_{transition}$ is defined as $Size_{buff\_tran\_max}$, and it is the max number of transitions which the hybrid computing platform can process in a single sampling period. The number of valid transitions inside $Buffer_{transition}$ is indicated as $Size_{buff\_tran}$, $0 \leq Size_{buff\_tran} \leq Size_{buff\_tran\_max}$. $Buffer_{transition}$ is transferred from CPU to GPU and kept in the global memory of GPU. Moreover, an important array is designed to store intermediate

token values, namely $Interm_{array\_token}$, where the max number of rows is $Size_{buff\_tran\_max} \times 2$ and the number of columns is $Size_{place}$. $Interm_{array\_token}$ resides in the surface memory of GPU for performance benefits with the row index ranging from 0 to $Size_{buff\_tran\_max} \times 2 - 1$.

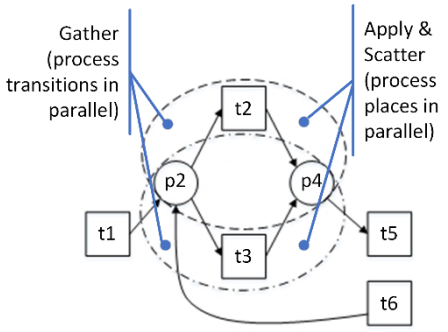

FIGURE 10. Applying GAS model to compute a part of workflow net of Figure 3.

C. GAS MODEL FOR WORKFLOW NET REPLAYING

Since a workflow net is also a directed bipartite graph, GAS model is adjusted to handle transition-centric computing of the bipartite graph. The concept of conformance checking algorithms is based on Definition 8 and (2). The three phases of the adjusted GAS model are explained with **Figure 10** and **Figure 11** to illustrate the transition-granular replay technique. It is supposed that two events are sampled in the current period. Then they are transformed into two visible transitions and are stored in $Buffer_{transition}$.

In the gather phase, the first element of $Buffer_{transition}$ is taken out, indicated as $t2$. The input arcs and the output arcs of transition $t2$ are searched out from C_{array} and R_{array}. Then the arcs are transformed into token flows and temporarily recorded in $Interm_{array\_token}$, where the token flows leaving places are stored in the even-indexed rows and the token flows entering places are stored in the odd-indexed rows. Supposing transition $t3$ is the successor of $t2$ in $Buffer_{transition}$, similar operations are carried out with $t3$ and the token flows are stored in the concatenating memory after $t2$ inside $Interm_{array\_token}$.

In the apply phase, $Vector_{mark}$ is added with the first even-indexed (e.g. the 0 index) row of $Interm_{array\_token}$, and the result is the number of tokens in the corresponding places, which is used to determine whether transition $t2$ is enabled as it is defined in Definition 2. If transition $t2$ is enabled to fire, the scatter phase is carried out. Otherwise, it means that a discrepant event is captured.

In the scatter phase, the modified $Vector_{mark}$ from the apply phase is added with the first odd-indexed (e.g. the 1 index) row of $Interm_{array\_token}$, and $Vector_{mark}$ is updated. If there are other token flows left in $Interm_{array\_token}$, such as the token flows of $t3$, jump to the apply phase and continue with next steps to process $t3$. Otherwise, all events sampled in the current period are in conformance with the workflow net.

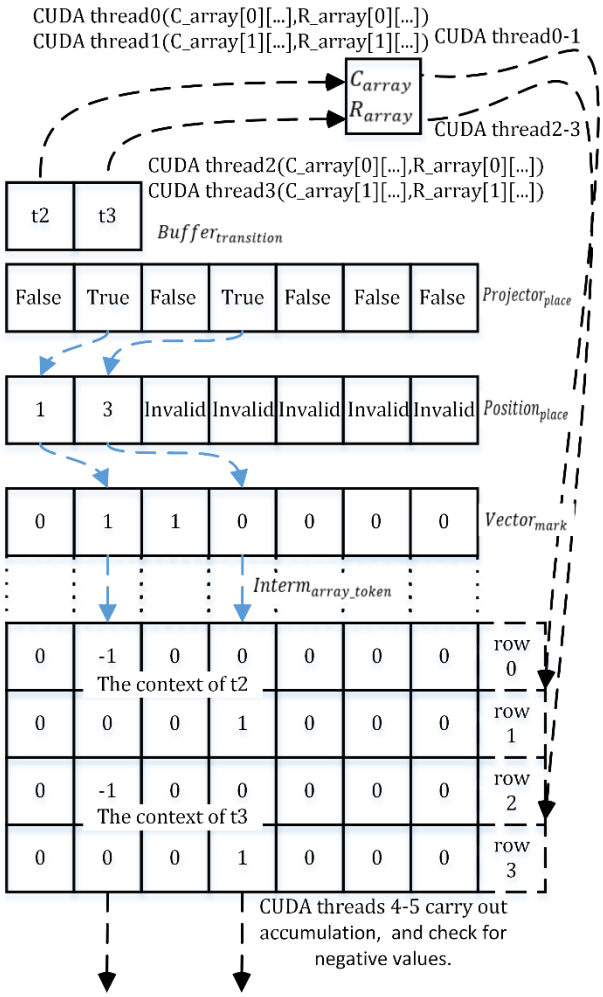

FIGURE 11. Data flow and CUDA thread diagram corresponding to Figure 10.

In real time conformance checking, the whole procedure is as follow. Each element of $Buffer_{transition}$ is processed by two CUDA threads to generate token flows in $Interm_{array\_token}$ during the gather phase. It means that the "transition" elements of $Buffer_{transition}$ are processed in parallel. One of the two CUDA threads handles C^- and R^-, while the other handles C^+ and R^+. Then the column positions containing new values inside different rows of $Interm_{array\_token}$ are projected into a row vector with $Size_{place}$ elements, namely $Projector_{place}$. Each element of $Projector_{place}$ is a variable of type Boolean to indicate whether the corresponding column of $Interm_{array\_token}$ contains new token flows. The gather phase algorithm is implemented in a single kernel program. When the kernel returns, $Projector_{place}$ is processed by CPU and is transformed into another row vector $Position_{place}$ with $Size_{place}$ elements. Each element of $Position_{place}$ either indicates the index of column which contains new token flows inside $Interm_{array\_token}$ or is invalid. The small contexts of the transitions are founded. The pseudo code of the gather phase algorithm is in subsection VIII.B.

During the apply-scatter phase in real time, $Position_{place}$ and $Vector_{mark}$ are loaded into the memory of GPU. Since the values of $Interm_{array\_token}$ resides in the surface memory of GPU, each CUDA thread reads out the token value from the element of $Vector_{mark}$ selected by $Position_{place}$, and then the CUDA thread accumulates the token value with the elements from the corresponding column inside $Interm_{array\_token}$ while updating $Vector_{mark}$ step by step.

It means that the related "place" columns of $Interm_{array\_token}$ and $Vector_{mark}$ are processed in parallel. When the number of tokens inside any place selected by $Position_{place}$ is negative, the apply-scatter phase is stopped, and the first discrepancy in the current period is captured. If no discrepancy is captured, each CUDA thread continue with accumulation and updating operations until all valid rows of $Interm_{array\_token}$ are processed. The algorithm for the apply-scatter phase is implemented in another single kernel program. The pseudo code of the apply-scatter phase algorithm is in subsection VIII.C.

Moreover, the flowchart of the assisting programs on CPU is in **Figure 12** and **Figure 13** with A, B, C as connection points.

D. COPING WITH DISCREPANCIES

In this paper, if the conformance checking algorithms are disrupted by one discrepant event of an event trace, it pauses until the next event trace comes so as to avoid the uncertain results of the following conformance checking operations caused by the inference based on free-choice structures (Definition 9) of the model. Then the algorithms resynchronize at the entry of the workflow net and continue with conformance checking operations of the new trace.

In the case of discrepancies, some remarks about replay strategy are given in paper [23], such as the forced firing of a non-enabled transition, restarting the replay of the full trace each time an event is added while allowing the replay program to revise earlier decisions. Unfortunately, the replay strategy [23] has deficiencies in the case when selecting one proper firing path from free-choice structures of workflow nets without the support of extra information. It is suggested that some prediction strategies can be designed to enable the conformance checking algorithms to cope with the discrepant events in real time. Then the discrepant event can be replaced by the predicted event and the conformance checking algorithms continue with the following checking operations for the remaining events in the same trace.

There are many techniques which can extract priori knowledge from event traces of workflow net models and make event-based predictions, such as constructing neural networks [51], training probabilistic models [52], [53]. Each of these mathematical and statistical methods above has its own advantages and strong points in extracting particular patterns from event sequences, but no one of them can cover all types of event sequences. Therefore, optimizations of the predictions should be studied based on application-specific knowledge, and a compromise between the accuracy and the robustness of conformance checking operations in real time should be made. The design of these advanced replay strategies is left for future work.

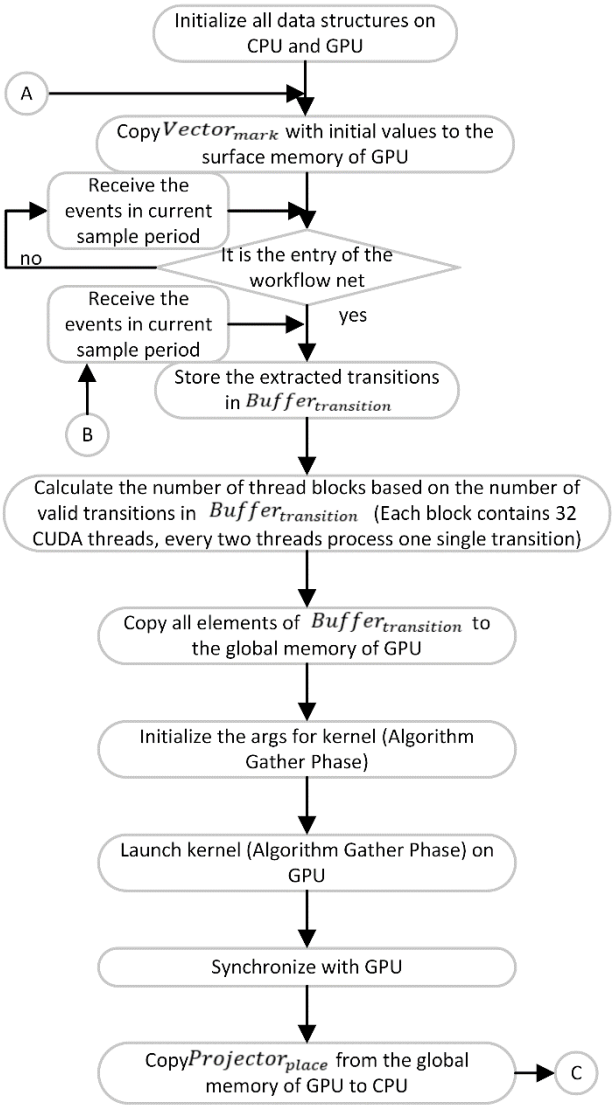

FIGURE 12. Flowchart of the assisting programs on CPU for the kernel algorithm gather phase.

E. DISCUSSION ABOUT CUDA PROGRAMMING OF THE ALGORITHMS

Firstly, with regard to one single workflow net (or subnet), the gather phase algorithm and the apply-scatter phase algorithm are executed in serial manner. The transitions corresponding to the observed events of the sampling period are processed with concurrent threads in the gather phase, and then the places connected to these transitions are processed with concurrent threads in the apply-scatter phase. It means the number of concurrent threads has an important influence on the performance of conformance checking operations.

In an ideal state, all the transitions (the number is indicated as n_t) of one single workflow net (or subnet) are supposed to

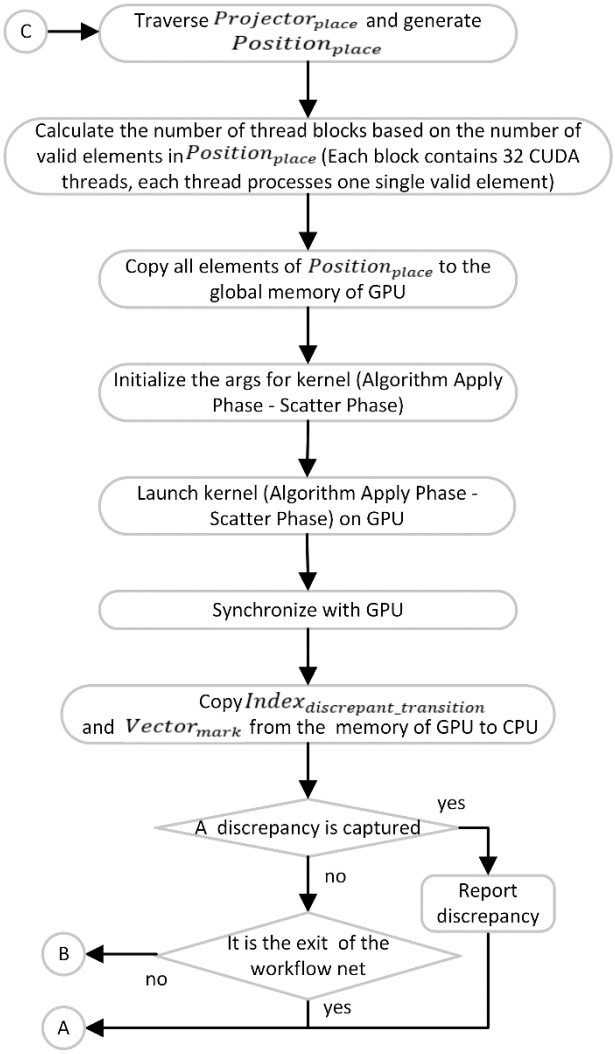

FIGURE 13. Flowchart of the assisting programs on CPU for the kernel algorithm apply-scatter phase.

be processed in parallel, and then all the places (the number is indicated as n_p) of one single workflow net (or subnet) are supposed to be processed in parallel. In according to the implementation of the algorithms, we need $2n_t$ concurrent threads to cope with transitions and n_p concurrent threads to cope with places. However, in one sampling period, there may be a small number of events which are captured, due to Principle of Locality. Then, the corresponding transitions are only related with a limit number of places. Thus, in subsections IV.B and IV.C, we make a compromise and decompose one workflow net into several subnets with the threshold $\mathbb{N}_{thread}$ (the maximum number of the concurrent threads in one single GPU device).

Secondly, GPU devices of NVIDIA Corp. often support Concurrent Kernel Execution. It means that the gather phase algorithm and the apply-scatter phase algorithm corresponding to one single workflow net (or subnet) can be assigned into one single stream, while other streams are launched with different kernels to process different workflow nets (or subnets). However, there is a few tips worthy of notice. Cooperative groups of CUDA 9.0 is used in this paper to alleviate synchronizing overhead between different thread blocks (namely, Grid Synchronization). It is observed that the kernel launched by the way of "cuLaunchCooperativeKernel" cannot execute concurrently with other streams. The kernel with the function of Grid Synchronization is executed exclusively until the kernel exits. Thus, it is suggested that the conventional kernel launch manner "$\lll \ldots \ggg$" and block wide synchronization strategy are utilized when the size of workflows net (or subnets) are small enough to be processed by one thread block. For instance, we can either run three same kernels (with three copies of data) concurrently with different streams to implement triple-modular redundancy, or run different kernels corresponding to different subnets concurrently with different streams. Moreover, the degree of parallelism among these streams is also determined by launch bound of these kernels and the available resources of GPU. The combining strategies of different streams to enhance the performance of GPU are left for future studies.

Thirdly, the type of the variables in the pseudo codes (VIII.B, VIII.C) of the CUDA algorithms are 32-bit integer. The 8-bit variables can also be used if necessary.

VI. EXPERIMENTAL RESULTS

The scheduling strategy of conformance checking with multiple GPU devices can be easily designed based on [21] and section IV. However, the actual performance of these tasks is usually affected by the communication speed between multiple GPU devices and the working speed of the scheduler communicating with these GPU devices. Therefore, the experiment for the conformance checking algorithm of this paper concentrates on one stream inside of one single GPU device to illustrate the advantages of transition-centric conformance checking in parallel.

Since Commercial off-the-shelf (COTS) embedded systems conforming to industrial grade are widely used on orbit, Jetson TX2i System-on-Module from NVIDIA Corporation is used as the platform for the transition-centric conformance checking benchmark. Jetson TX2i module contains a 256-core NVIDIA Pascal GPU with CUDA capabilities 6.2, and the maximum operating frequency of GPU is 1.12GHz. There are two CPU clusters inside the module including NVIDIA Denver 2 (Dual-Core) Processor (the maximum operating frequency is 1.95 GHz) and ARM® Cortex® - A57 MPCore (Quad-Core) Processor (the maximum operating frequency is 1.92 GHz). Moreover, there is a set of LPDDR4 memory inside the module, the characteristics of which includes a 128-bit DRAM interface, a memory bus with 1600 MHz as maximum frequency, 8GB memory space. The on-orbit hardware is designed while conforming to Open VPX standard, shown in **Figure 14**. As the tool nvpmodel is supplied by NVIDIA Corporation to manage the performance of Jetson TX2i module, the test cases of this paper are benchmarked with MAXP_CORE_ARM power mode. The time precision of the measurement is microsecond.

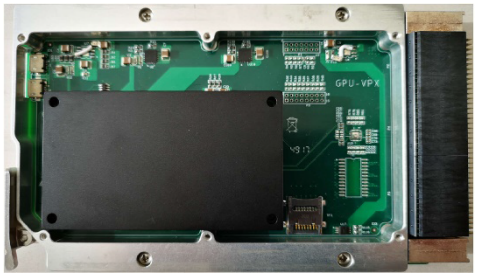

FIGURE 14. The on-orbit hybrid parallel computing platform based on Jetson TX2i, Open VPX 3U board (also compatible with Jetson TX1).

To illustrate the performance of the algorithms, the workflow net models for benchmark are from the dataset[4] [24]. There are two test cases as follows.

The model pr-11-14-1904-A14 contains 83 places and 114 transitions. The event log pr-11-14-1904-A14-01 contains 2000 traces. The length of each trace ranges from 12 to 17. There is no duplicate transition in the model, but there are four invisible transitions in the free-choice structures of the model. In this paper, the four invisible transitions are considered as visible transitions by the conformance checking algorithms. Then the original traces related to the four transitions are considered as the traces containing discrepant events. With regard to sampling period, two conditions are tested with the event log: a single trace is sampled in each period (indicated as Test Case A in **Table 2**); a single event is sampled in each period (indicated as Test Case B in **Table 2**).

The model pr-1-11-1244-A59 contains 71 places and 68 transitions. The event log pr-1-11-1244-A59-m17-l1-noise contains 2000 traces. The length of each trace ranges from 12 to 27. There is no duplicate transition or invisible transition in the model. With regard to sampling period, the two conditions above are tested with the event log. A single trace is sampled in each period (indicated as Test Case C in **Table 2**); a single event is sampled in each period (indicated as Test Case D in **Table 2**).

[4]https://data.4tu.nl/repository/uuid:b8c59ccb-6e14-4fab-976d-dd76707bcb8a

The processing time (indicated as $Time_{all}$ in **Table 2**) between the point when transitions are extracted by CPU and the point when the conformance checking results are returned to CPU is measured, including the computing time of the gather phase algorithm and the apply-scatter phase algorithm on GPU, the time of assisting programs of CPU. Besides, the computing time of the gather phase algorithm (indicated as $Time_{gather}$ in **Table 2**) and the computing time of the apply-scatter phase algorithm (indicated as $Time_{apply\_scatter}$ in **Table 2**) on GPU are also measured respectively. There are two run-time scenarios: a full trace of transitions is processed when each of the GPU kernels above is launched once (such as Test Case A and C); a transition is processed when each of the GPU kernels above is launched once (such as Test Case B and D). In **Table 2**, it is observed that the execution time of the algorithm for gather phase is usually longer than the algorithm for apply-scatter phase, because there are additional operations which access surface memory in the algorithm for gather phase, such as the initialization of $Interm_{array\_token}$.

When a full trace is processed in one single sampling period, the time of the conformance checking algorithms spent on each transition can be calculated by (15). $length_{current\_trace}$ is the number of events processed inside the trace of current sampling period before the first discrepancy is captured.

$$Time_{each\_event\_in\_trace} = Time_{all}/length_{current\_trace} \quad (15)$$

The average value of $Time_{each\_event\_in\_trace}$ is approximately 0.69 ms in Test Case C. Compared with $Time_{all}$ (average) of Test Case B and D inside **Table 2**, it indicates the advantages of parallel processing when multiple events are sampled.

The benchmark program makes use of the default stream of CUDA. All the task is processed within tens of milliseconds. It may cost more time when the program

TABLE 2. The measured performance parameters of four test cases.

Test Cases	Test Case A, (2000 traces)	Test Case B, (27008 events)	Test Case C, (2000 traces)	Test Case D, (34128 events)
$Time_{all}$ (max [a])	17.395 ms	18.691 ms	31.006 ms	12.309 ms
$Time_{all}$ (min [b])	5.675 ms	1.09 ms	3.778 ms	1.005 ms
$Time_{all}$ (average [c])	7.796 ms	3.556 ms	11.712 ms	3.275 ms
$Time_{gather}$ (max)	11.346 ms	17.58 ms	15.747 ms	11.743 ms
$Time_{gather}$ (min)	4.447 ms	0.904 ms	2.252 ms	0.816 ms
$Time_{gather}$ (average)	5.052 ms	3.109 ms	4.983 ms	2.830 ms
$Time_{apply\_scatter}$ (max)	8.669 ms	5.542 ms	16.799 ms	7.031 ms
$Time_{apply\_scatter}$ (min)	1.187 ms	0.17 ms	0.944 ms	0.183 ms
$Time_{apply\_scatter}$(average)	2.742 ms	0.446 ms	6.728 ms	0.443 ms

[a] It indicates the maximum value in the sample sequence.
[b] It indicates the minimum value in the sample sequence.
[c] It indicates the average value of the sample sequence and equals to the sum of sampled values divided by the number of elements inside the sample sequence.

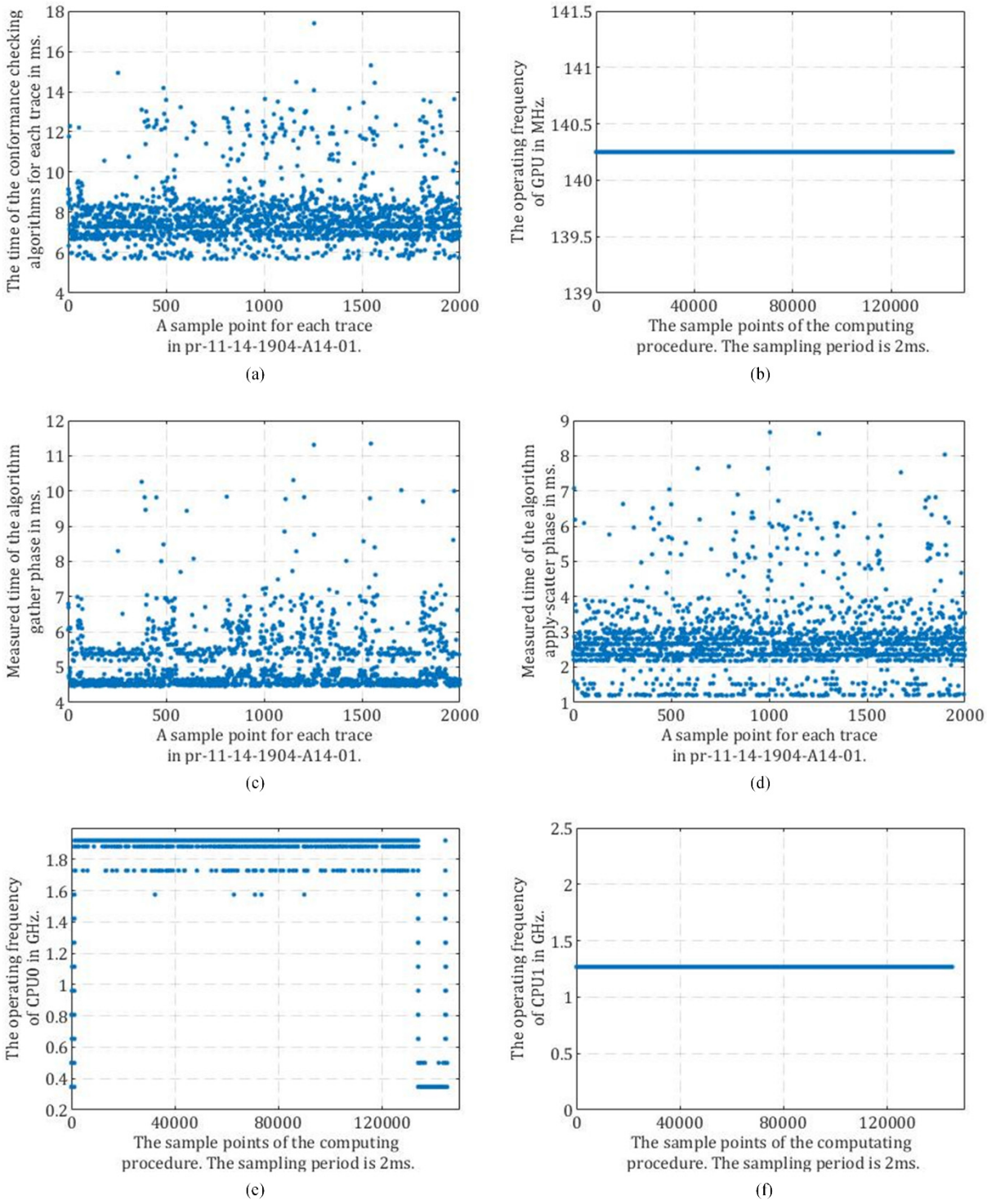

FIGURE 15. In test case A: (a) the execution time of the all algorithms; (b) the operating frequency of GPU; (c) the execution time of gather phase; (d) the execution time of apply-scatter phase; (e) the operating frequency of CPU (0); (f) the operating frequency of CPU (1).

processes the first trace or the first transition, because the GPU of Jetson TX2i module needs some warm-up operations. The warm-up operations are commonly used in the official CUDA samples from NVIDIA Corporation. The fluctuations of measured time during continuous tests are mainly caused by the following factors: scheduling operations of Linux and CUDA runtime, dynamic voltage and frequency scaling (DVFS) strategy of Jetson TX2i module.

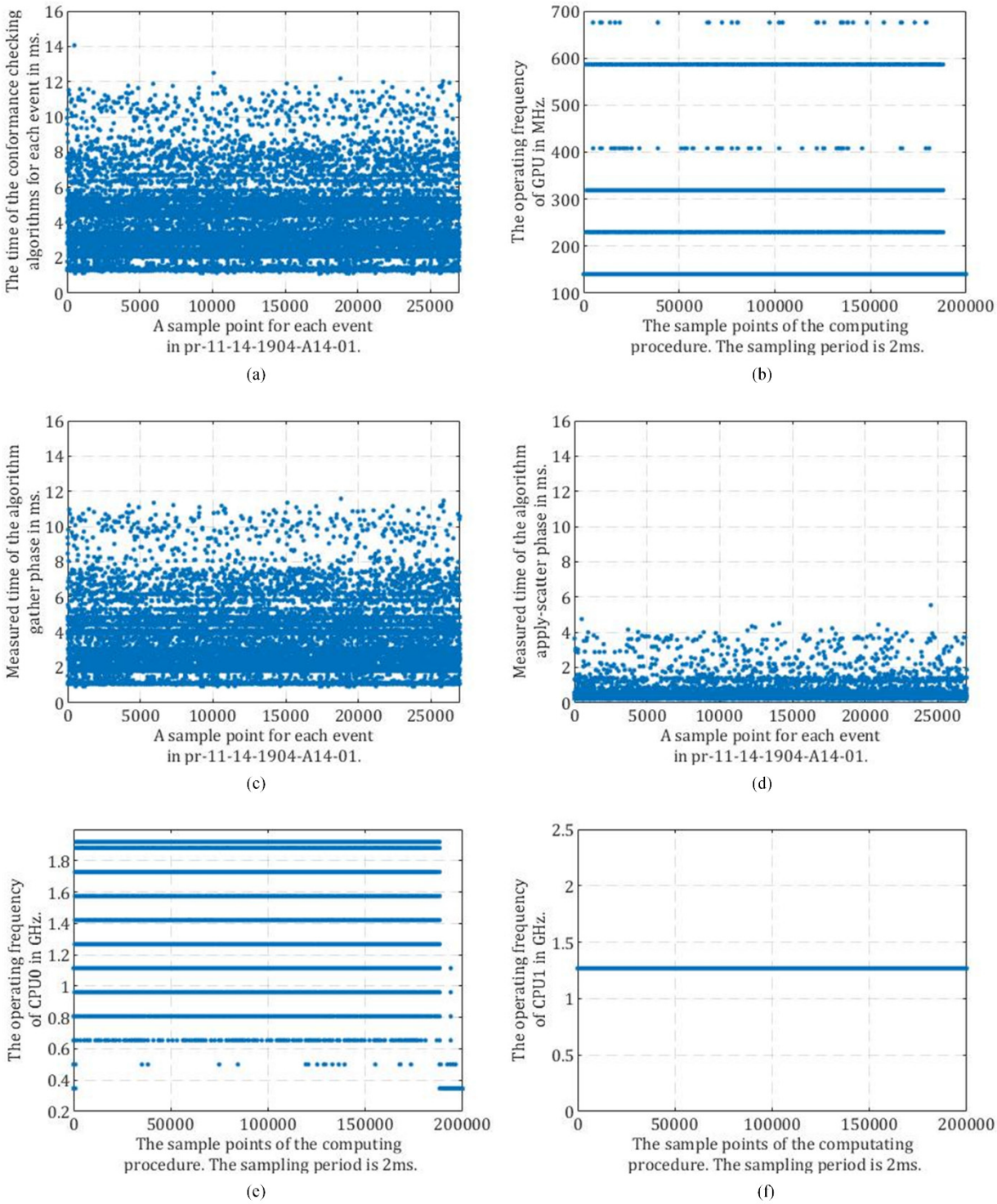

FIGURE 16. In test case B: (a) the execution time of the all algorithms; (b) the operating frequency of GPU; (c) the execution time of gather phase; (d) the execution time of apply-scatter phase; (e) the operating frequency of CPU (0); (f) the operating frequency of CPU (1).

In test case A and B, 249 discrepancies are captured, which is caused by the invisible transitions in the free-choice structures of the model pr-11-14-1904-A14. In Figure 15 (b), it is observed that the operating frequency of GPU is 140.25 MHz most of the time, which is the minimum frequency of GPU in MAXP_CORE_ARM power mode. The fluctuations of the execution time in test case A are mainly caused by the algorithm for apply-scatter phase shown as Figure 15 (d),

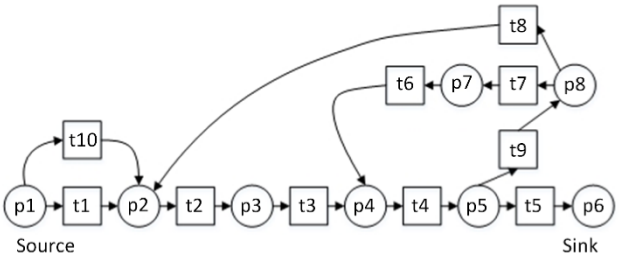

FIGURE 17. A workflow net with two loops.

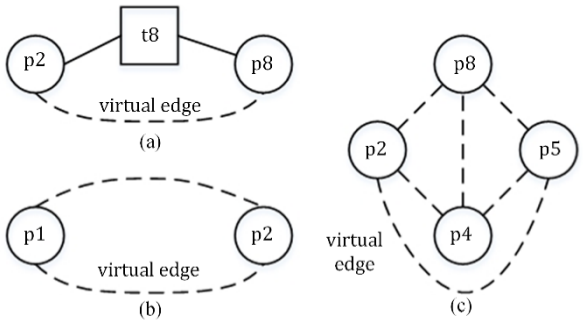

FIGURE 18. RPST components: node (P3) indicated as (a), node (B0) indicated as (b), node (R0) indicated as (c).

because the number of loop times for accumulation inside the algorithm is determined by different length of event traces in the log. There is a little deviation in the time measurement, which is introduced by the changing frequencies of CPUs. In MAXP_CORE_ARM power mode, it is observed that the CPU (0, 3, 4, 5)[5] are similar in frequency variation shown as Figure 15 (e), and CPU (1, 2) get the same frequency value shown as Figure 15 (f).

In **Figure 16**, DVFS strategy has a significant effect on the frequency of GPU, and it is observed that there are mainly six different levels of frequency. The highest frequency captured in **Figure 16** (b) is approximately 675.75 MHz and is a little more than the half of the maximum frequency of GPU in MAXP_CORE_ARM power mode. The fluctuations of the execution time of the gather phase in **Figure 16** (c) are mainly caused by different frequencies of GPU. As to the apply-scatter phase in **Figure 16** (d), the fluctuation of execution time is not obvious because only one transition is processed in every sampling period, and the execution time is short. The frequencies of CPUs in test case B are also obviously affected by DVFS strategy, and it also introduces a little deviation in the time measurement. In test case C and D, no discrepancy is captured, and other test phenomena are similar to test case A and B respectively.

It is said by NVIDIA Corporation that the DVFS scheme is not public to customer for patents protection, and the scheme provides the best performance based on the minimal amount of energy. We think that the boost of the frequencies of GPU relates to the intensity of kernel launches. Although the DVFS scheme may adjust the frequency of GPU to the minimum frequency, the execution time of the conformance checking algorithms is tens of milliseconds. Therefore, the hybrid parallel computing algorithms of this paper are acceptable for practical use in coping with the workflow nets with proper size.

[5]The serial numbers of CPU refer to the path "/sys/devices/system/cpu/" in Linux operating system on Jetson TX2i.

VII. CONCLUSION

The foremost contribution of this paper is a set of hybrid parallel computing algorithms for conformance checking of workflow nets in real time, and it can be used as an edge computing application to support on-orbit space science experiments. These algorithms take advantage of graph processing techniques to accelerate conformance checking operations and can be used in combination with the decomposition strategy based on spectral graph clustering method, so as to cope with workflow nets of various structures. The benchmark above indicates that the performance of the algorithms is

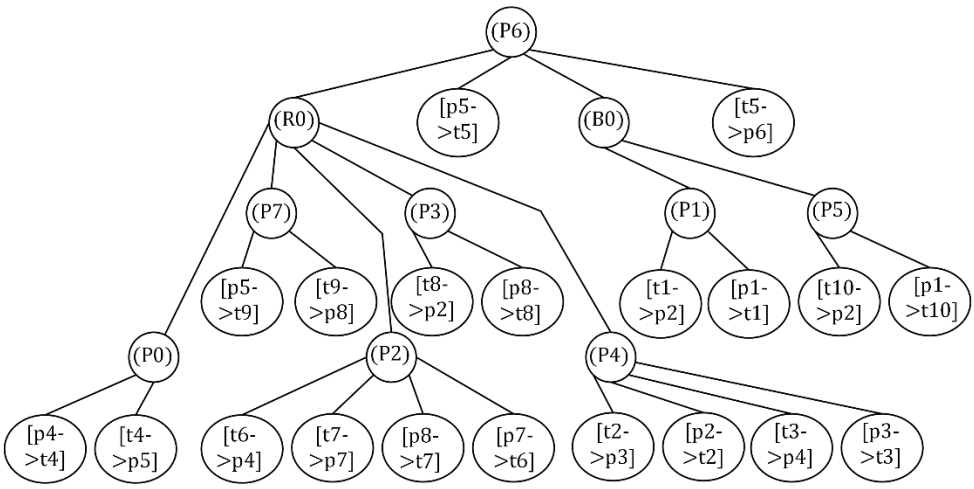

FIGURE 19. The RPST computed based on the workflow net of Figure 17.

acceptable for practical use on orbit. When there are enough resources in a single Jetson TX2i module, more CUDA streams can be created to monitor several space science payloads at the same time.

In future studies, more attributes of workflow net models will be taken into consideration to extend the current approach, such as time and other important statuses sampled from the on-orbit scenarios. The robustness of the conformance checking algorithms will also be improved. Moreover, Single Event Upset will be taken into account with the design of software protecting strategies, and the thermal effect on the performance of the algorithms will be estimated.

APPENDIX

A. ANALYSIS OF RPST BASED DECOMPOSITION

In paper [22], RPST of workflow net is constructed on the basis of papers [36], [37], and the subparts of RPST are partitioned out, conforming to the valid decomposition constraints. However, it is difficult to take into account both the ''balanced'' factor and ''coupling'' factor of subnets at the same time.

The procedure of RPST based decomposition for conformance checking is analyzed as follow. The main problem in computing RPST is to construct the tree of the tri-connected components from the graph structure of petri net (workflow net) [36], which leads to recursive cycle decompositions and merge operations [37]. A RPST is constructed from three types of tri-connected components [36]: maximal bonds, maximal polygons, rigid split components, and these components consist of several trivial edges and virtual edges. In the computation process of RPST, a petri net is considered as an undirected graph, and it is the same as subsection III.B.2). Trivial edges are mapped to arcs of the graph. A maximal bond indicated as (Bi) in **Figure 19** consists of 2 vertexes and $m \geq 2$ trivial edges (or virtual edges), and there is no other bond that shares a virtual edge with this bond. A maximal polygon indicated as (Pi) in **Figure 19** is a graph that has $m \geq 3$ vertexes and k trivial edges (or virtual edges) such that all nodes and edges are contained in a cycle, and there is no other polygon that shares a virtual edge with this polygon. A rigid split component indicated as (Ri) in **Figure 19** is a simple tri-connected graph (it may be a k-connected graph, $k \geq 3$, e.g. the RPST of **Figure 6**), where simple means that no pair of nodes is connected by more than one edge (or one virtual edge). Virtual edges are used to assist recursive cycle decomposition and merge operation [36], [37]. When RPST is constructed, virtual edges are deleted.

Figure 19 is an example RPST of **Figure 17**. The leaf node $[t5 \rightarrow p6]$ is a trivial edge. The non-leaf node (P3) is a maximal polygon including two trivial edges $[p8 \rightarrow t8]$, $[t8 \rightarrow p2]$, and an undirected virtual edge in dashed line between $p8$ and $p2$, shown as **Figure 18** (a). The non-leaf node (R0) is a rigid split component including six virtual edges in dashed line, shown as **Figure 18** (c). The non-leaf node (B0) is a maximal bond including two virtual

Algorithm 4 Gather Phase, in CUDA 9.0. Search Out the Input Arcs and Output Arcs Which Are Connecting the Transitions Inside $Buffer_{transition}$. There is a Small Trick to Alleviate Branch Divergence. The Threads With Even Number as the $thread_{rank}$ Access Even-Indexed (Row Index is 0) Rows of C_{array} and R_{array}. The Threads With Odd Number as the $thread_{rank}$ Access Odd-Indexed (Row Index is 1) Rows of C_{array} and R_{array}.

Input parameters : C_{array}, R_{array}, $Size_{place}$, $Buffer_{transition}$, $Size_{buff\_tran}$, $Vector_{mark\_value}$.
Output parameters: $Interm_{array\_token}$, $Projector_{place}$.
-$Vector_{mark\_value}$ is a row vector with two elements $\{-1, 1\}$.
-*surf*2*Dwrite*, *atomicExch*, *tex*1*D*, *tex*2*D* are CUDA APIs. cudaBoundary-ModeTrap is used by surface operations.

```
//CUDA 9.0 cooperative groups.
grid = cg:: this_grid();
if grid → thread_rank < Size_buff_tran × 2
    parallel for 0 ≤ i < Size_place
        //Initialize the elements of Interm_array_token
        //with zero.
        //Carry out surface memory operations.
        surf2Dwrite(0, Interm_array_token, i × 4, grid →
        thread_rank);
    //Obtain the access positions for the current thread.
    //It is the trick to alleviate branch divergence.
    Index_odd_or_even = grid → thread_rank mod 2;
    Index_buffer_transition = (grid →
    thread_rank − Index_odd_or_even) / 2;
    //Get the transition for the current thread.
    ID_transition = Buffer_transition[Index_buffer_transition];
    //Calculate the offset and range inside C_array based
    //on R_array. Carry out texture memory operations.
    offset_start = tex2D(R_array,
    ID_transition, Index_odd_or_even);
    offset_end = tex2D(R_array, ID_transition + 1,
    Index_odd_or_even);
    parallel for offset_start ≤ i < offset_end
        //Get the input arcs or output arcs of the current
        //transition. The IDs of the places connecting to
        //these arcs are recorded.
        ID_place = tex2D(C_array, i, Index_odd_or_even);
        //Get the value -1 or 1 for input arcs or output arcs.
        Mark_value = tex1D(Vector_mark_value,
        Index_odd_or_even);
        //Write the value -1 or 1 into the corresponding
        //position of Interm_array_token.
        surf2Dwrite(Mark_value, Interm_array_token,
        ID_place × 4, grid → thread_rank);
        //The atomic function are used to update
        //Projector_place in global memory. 0 × AA
        //indicates that the corresponding column of
        //Interm_array_token needs to be processed by
        //algorithm 5 in Apply-Scatter phase. 0 × 0
        //indicates that no operation is needed.
        atomicExch(&(Projector_place[ID_place]), 0xAA);
```

Algorithm 5 Apply-Scatter Phase in CUDA 9.0. Each Thread Computes One of the Columns Inside $Interm_{array\_token}$ Selected by the Vector $Position_{place}$. The Corresponding Elements of $Vector_{mark}$ is Added With the Elements of $Interm_{array\_token}$. The Order of Add Operations Is in Conformance With the Occurrence Sequence of Transitions. When the Intermediate Result of Any Add Operation Produces a Negative Value, a Discrepant Transition is Captured.

Input : $Size_{buff\_tran}$, $Vector_{mark}$, $Position_{place}$, $Count_{valid\_place}$, $Interm_{array\_token}$.
Output: $Index_{discrepant\_transition}$, $Vector_{mark}$.
-$Position_{place}$ contains the index of columns inside $Interm_{array\_token}$. The selected columns are processed.
-$Count_{valid\_place}$ is the number of elements with valid values inside $Position_{place}$.
-$Index_{\text{discrepant\_transition}}$ is the index of the discrepant transition inside $Buffer_{transition}$.
-$surf1Dread$, $surf2Dread$, $atomicMin$, $surf1Dwrite$ are CUDA APIs. cudaBoundary-ModeTrap is used by surface operations.

```
//CUDA 9.0 cooperative groups.
grid = cg :: this_grid();
if grid → thread_rank < Count_valid_place
    //Read out the current mark and prepare for add
    //operations. Carry out surface memory operations.
    place_thread_to_process =
    Position_place [grid → thread_rank];
    surf1Dread(Mark_value, Vector_mark,
    place_thread_to_process × 4);
parallel for 0 ≤ i < Size_buff_tran × 2
    if grid → thread_rank < Count_valid_place
        //Read out the value from the corresponding
        //column of Interm_array_token. Carry out surface
        //memory operations and add operations.
        surf2Dread(Mark_value_column, Interm_array_token,
        place_thread_to_process × 4, i);
        Mark_value = Mark_value + Mark_value_column;
        if i mod2 = 0
            //Check the mark value when tokens flow out
            //of the place.
            if Mark_value < 0
                Index_temp_transition = i/2;
                //The atomic function are used to update
                //Index_discrepant_transition in global
                //memory with valid values.
                atomicMin(&Index_discrepant_transition,
                Index_temp_transition);
        else
            //Check whether Vector_mark needs to be
            //updated. Then, update Vector_mark, and
            //carry out surface memory operations.
            if Mark_value_column > 0
                surf1Dwrite(Mark_value, Vector_mark,
                place_thread_to_process × 4);
    //Synchronize the threads of the whole grid. Exit
    //from kernel if a discrepant transition is captured.
    grid → sync;
    if Index_discrepant_transition has a valid value
        break;
```

edges in dashed line, shown as **Figure 18** (b). The RPST in **Figure 19** is constructed by merging the tri-connected components which shares the same virtual edge. Therefore, instances of the three types of tri-connected components are nested within each other.

A decomposition of petri net (workflow net) based on RPST is to find the biggest non-leaf nodes, the size of which is smaller than the proper threshold while conforming to the constraint of valid decomposition requirements [21]. The size of one node is the number of trivial edges contained in the node [22]. Since the structure of petri net (workflow net) has a strong impact on the number of trivial edges belonging to the non-leaf nodes in RPST, it is difficult to decompose the petri net (workflow net) models of complicated structure into balanced subnets. For example, if there is a node containing numerous trivial edges inside the RPST, and the node is indicated as a rigid split component, it is difficult to decompose the rigid split component into the children of the current node. Because the children of the node are interconnected with each other, and the subnets resulting from the decomposition are tightly coupled. In the worst case, the children of the rigid split component may be the maximal polygons or maximal bonds containing other rigid split components. Supposing that the size of some children of (R0) in the RPST of **Figure 19** is greater than the proper threshold while the size of other children is smaller than the threshold, the smaller children are decomposed as subnets. Then, the remaining part of R0 will be decomposed iteratively. Finally, the size of each subnet is divergent. Besides, when the boundaries of subnets contain the vertexes of type "place" (Definition 1), bridges [22] have to be added as subnets to circumvent the constraint of valid decomposition [21]. As a result, there will be a lot of shared transitions.

Since it can be observed that rigid split components are usually caused by multiple loops sharing the same arcs inside workflow nets, and these loops often relate to free-choice structures, it is suggested that model designers should preprocess these loops to reduce the difficulty of decomposition.

Therefore, this paper proposes decomposing workflow net with spectral graph clustering method, or it is suggested that the RPST based decomposition is used in a coarse-grained manner guided by designers, when the size of workflow net model is too huge to be processed by a single GPU device.

B. CUDA STYLE PSEUDO CODE OF ALGORITHM GATHER PHASE

See Algorithm 4.

C. CUDA STYLE PSEUDO CODE OF ALGORITHM APPLY-SCATTER PHASE

See Algorithm 5.

REFERENCES

[1] M. Werner, "Financial process mining—Accounting data structure dependent control flow inference," *Int. J. Accounting Inf. Syst.*, vol. 25, pp. 57–80, May 2017. doi: 10.1016/j.accinf.2017.03.004.

[2] S. Y. Son *et al.*, Process mining for manufacturing process analysis: A Case study. presented at the AP-BPM, Eindhoven, the Netherlands, 2014. [Online]. Available: https://www.researchgate.net/publication/271910986_Process_Mining_for_Manufacturing_Process_Analysis_A_case_Study

[3] W. M. P. van der Aalst, "Process modeling and analysis, operational support, tool support," *Process Mining: Discovery, Conformance Enhancement Business Processes*. Berlin, Germany: Springer, 2011, pp. 31–46, pp. 241–258, pp. 261. [Online]. Available: http://www.processmining.org/book/start

[4] S. Olabarriaga *et al.*, "Scientific workflow management—For whom?" in *Proc. IEEE 10th Int. Conf. e-Science*, Sao Paulo, Brazil, Oct. 2014, pp. 298–305. doi: 10.1109/eScience.2014.8.

[5] A. Bolt, M. de Leoni, and W. M. P. van der Aalst, "Scientific workflows for process mining: Building blocks, scenarios, and implementation," *Int. J. Softw. Tools Technol. Transf.*, vol. 18, no. 6, pp. 607–628, 2016. doi: 10.1007/s10009-015-0399-5.

[6] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," *Future Gener. Comput. Syst.*, vol. 75, pp. 216–227, Oct. 2017. doi: 10.1016/j.future.2017.05.041.

[7] E. Deelman *et al.*, "The future of scientific workflows," *Int. J. High-Perform. Comput. Appl.*, vol. 32, no. 1, pp. 159–175, Apr. 2017. doi: 10.1177/1094342017704893.

[8] D. Watkins, D. Thomas, L. Hetherton, M. Bolger, and P. W. Cleary, "Workspace—A scientific workflow system for enabling research impact," presented at the MODSIM, Dec. 2017. [Online]. Available: https://mssanz.org.au/modsim2017/C3/watkins.pdf

[9] G. Lentaris *et al.*, "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation," *J. Aerosp. Inf. Syst.*, vol. 15, no. 4, pp. 178–192, Apr. 2018. doi: 10.2514/1.I010555.

[10] G. Furano and A. Menicucci, "Roadmap for on-board processing and data handling systems in space," *Dependable Multicore Architectures at Nanoscale*. Cham, Switzerland: Springer, Aug. 2018, pp. 253–281. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-54422-9_10. doi: 10.1007/978-3-319-54422-9_10.

[11] V. Cuevas-Vicenttín, S. Dey, S. Köhler, S. Riddle, and B. Ludäscher, "Scientific workflows and provenance: Introduction and research opportunities," *Datenbank-Spektrum*, vol. 12, no. 3, pp. 193–203, Nov. 2012. doi: 10.1007/s13222-012-0100-z.

[12] W. Zhang, "Progress of tiangong-2 space science and application mission," (in Chinese) *Space Int.*, vol. 456, pp. 18–23, Dec. 2016. [Online]. Available: http://www.cnki.com.cn/Article/CJFDTotal-GJTK201612006.htm

[13] S. Yan, X. W. Wang, Q. S. Liu, and Z. Q. Zhu, "Ground-based experimental results of TZ-1 matching scientific condensation experiment," (in Chinese) *Chin. J. Space Sci.*, vol. 36, no. 4, pp. 531–535, 2016. doi: 10.11728/cjss2016.04.531.

[14] Z. Yin, X. Zhang, J. Wu, X. Li, J. Yu, and Z. Yuan, "Solidification and crystal growth on the SJ-10 recoverable scientific experiment satellite," *Chin. J. Space Sci.*, vol. 38, no. 5, pp. 836–838, 2018. doi: 10.11728/cjss2018.05.836.

[15] T. Zhang, J. Zhang, W. Shu, M.-Y. Wu, and X. Liang, "Efficient graph computation on hybrid CPU and GPU systems," *J. Supercomput.*, vol. 71, no. 4, pp. 1563–1586, Apr. 2015. doi: 10.1007/s11227-015-1378-z.

[16] X. H. Shi *et al.*, "Graph processing on GPUs: A survey," *ACM Computing Surv.*, vol. 50, no. 6, Jan. 2018, Art. no. 81. doi: 10.1145/3128571.

[17] A. Bobbio, "System modelling with petri nets," *Systems Reliability Assessment*, Dordrecht, The Netherlands: Springer, 1990, pp. 103–143. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-94-009-0649-5_6#citeas. doi: 10.1007/978-94-009-0649-5_6.

[18] P. C. Yianni, L. C. Neves, D. Rama, and J. D. Andrews, "Accelerating petri-net simulations using NVIDIA graphics processing units," *Eur. J. Oper. Res.*, vol. 265, no. 1, pp. 361–371, Feb. 2018. doi: 10.1016/j.ejor.2017.06.068.

[19] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. KDD*, San Francisco, CA, USA, Aug. 2001, pp. 269–274. doi: 10.1145/502512.502550.

[20] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007. doi: 10.1007/s11222-007-9033-z.

[21] W. van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distrib. Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013. doi: 10.1007/s10619-013-7127-5.

[22] J. Munoz-Gama, "Conformance checking and diagnosis in process mining," Ph.D. dissertation, Dept. Software, UPC Barcelona Tech., Barcelona, Spain, 2014.

[23] S. K. L. M. V. Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, and J. Vanthienen, "Event-based real-time decomposed conformance analysis," in *Proc. OTM Conf.*, Amantea, Italy, vol. 8841, 2014, pp. 345–363. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-662-45563-0_20. doi: 10.1007/978-3-662-45563-0_20.

[24] J. Munoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Single-entry single-exit decomposed conformance checking," *Inf. Syst.*, vol. 46, pp. 102–122, Dec. 2014. doi: 10.1016/j.is.2014.04.003.

[25] A. Zolghadri, "The challenge of advanced model-based FDIR for real-world flight-critical applications," *Eng. Appl. Artif. Intell.*, vol. 68, pp. 249–259, Feb. 2018. doi: 10.1016/j.engappai.2017.10.012.

[26] R. Cornelius and J. Frank, "International space station (ISS) payload autonomous operations past, present and future," in *Proc. SpaceOps Conf.*, 2016, pp. 1–16. [Online]. Available: https://ti.arc.nasa.gov/publications/34626/download/

[27] *Spacecraft On-Board Control Procedures*, ECSS Standard ECSS-E-ST-70-01C, 2010.

[28] NASA, Washington, DC, USA. (2012). *Fault Management Handbook, NASA Technical handbook NASA-HDBK-1002, DRAFT 2*. [Online]. Available: https://www.nasa.gov/pdf/636372main_NASA-HDBK-1002_Draft.pdf

[29] S. Indra, L. Travé-Massuyès, and E. Chanthery, "A decentralized fault detection and isolation scheme for spacecraft: Bridging the gap between model-based fault detection and isolation research and practice," in *Proc. EUCASS*, vol. 6, 2013, pp. 281–298. doi: 10.1051/eucass/201306281.

[30] M. Tipaldi and B. Bruenjes, "Survey on fault detection, isolation, and recovery strategies in the space domain," *J. Aerosp. Inf. Syst.*, vol. 12, no. 2, pp. 235–256, Feb. 2015. doi: 10.2514/1.I010307.

[31] C. Torens, F. M. Adolf, P. Faymonville, and S. Schirmer, "Towards intelligent system health management using runtime monitoring," in *Proc. AIAA Inf. Syst.-AIAA Infotech @ Aerosp., AIAA SciTech Forum*, 2017, p. 0419. doi: 10.2514/6.2017-0419.

[32] C. Sánchez *et al.* (2018). "A survey of challenges for runtime verification from advanced application domains (beyond software)." [Online]. Available: https://arxiv.org/abs/1811.06740

[33] J. Schumann, K. Y. Rozier, T. Reinbacher, O. J. Mengshoel, T. Mbaya, and C. Ippolito, "Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems," *Int. J. Prognostics Health Manage.*, vol. 6, no. 1, pp. 1–27, Jun. 2015. [Online]. Available: http://works.bepress.com/ole_mengshoel/53/

[34] J. Zhu, P. Wei, S. Xie, and R. Lu, "A dynamic conformance checking method based on petri nets for satellite communication system network control protocol," in *Proc. 11th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Shanghai, China, Sep. 2015, pp. 444–451. [Online]. Available: http://59.80.44.47/toc.proceedings.com/30044webtoc.pdf. doi: 10.1049/cp.2015.0709.

[35] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, Mar. 2008. doi: 10.1016/j.is.2007.07.001.

[36] A. Polyvyanyy, J. Vanhatalo, and H. Völzer, "Simplified computation and generalization of the refined process structure tree," in *Proc. WS-FM*, Hoboken, NJ, USA, vol. 6551, 2010, pp. 25–41. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-19589-1_2. doi: 10.1007/978-3-642-19589-1_2.

[37] M. Mader, "Planar graph drawing," M.S. thesis, Dept. Comput. Inf. Sci., Univ. Konstanz, Konstanz, Germany, 2008.

[38] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, "Determining process model precision and generalization with weighted artificial negative events," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1877–1889, Aug. 2014. doi: 10.1109/TKDE.2013.130.

[39] L. García-Bañuelos, N. R. T. P. van Beest, M. Dumas, M. L. Rosa, and W. Mertens, "Complete and interpretable conformance checking of business processes," *IEEE Trans. Softw. Eng.*, vol. 44, no. 3, pp. 262–290, Mar. 2017. doi: 10.1109/TSE.2017.2668418.

[40] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. IEEE 15th Int. Enterprise Distrib. Object Comput. Conf.*, Helsinki, Finland, Aug./Sep. 2011, pp. 55–64. doi: 10.1109/EDOC.2011.12.

[41] F. Taymouri and J. Carmona, "A recursive paradigm for aligning observed behavior of large structured process models," in *Proc. BPM*, Rio de Janeiro, Brazil, vol. 9850, Sep. 2016, pp. 197–214. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-45348-4_12#citeas. doi: 10.1007/978-3-319-45348-4_12.

[42] S. J. van Zelst, A. Bolt, M. Hassani, B. F. van Dongen, and W. M. P. van der Aalst, "Online conformance checking: Relating event streams to process models using prefix-alignments," *International Journal of Data Science and Analytics*. Springer, May 2017, pp. 1–16. [Online]. Available: https://link.springer.com/article/10.1007%2Fs41060-017-0078-6#citeas. doi: 10.1007/s41060-017-0078-6.

[43] W. L. J. Lee, H. M. W. Verbeek, J. Munoz-Gama, W. M. P. van der Aalst, and M. Sepúlveda, "Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining," *Inf. Sci.*, vol. 466, pp. 55–91, Oct. 2018. doi: 10.1016/j.ins.2018.07.026.

[44] W. Song, H.-A. Jacobsen, C. Z. Zhang, and X. X. Ma, "Dependence-based data-aware process conformance checking," *IEEE Trans. Services Comput.*, to be published. doi: 10.1109/TSC.2018.2821685.

[45] A. Burattin and J. Carmona, "A framework for online conformance checking," in *Proc. BPM*, Barcelona, Spain, Sep. 2017, pp. 165–177. doi: 10.1007/978-3-319-74030-0_12.

[46] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *Proc. BPM*, Tallinn, Estonia, Sep. 2012, pp. 82–97. doi: 10.1007/978-3-642-32885-5_6.

[47] D. Borrego, I. Barba, and P. Abad, "Data-aware conformance checking for declarative business process models," *Information System Development*, Cham, Switzerland: Springer, Jul. 2014, pp. 269–282. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-07215-9_22#citeas. doi: 10.1007/978-3-319-07215-9_22.

[48] A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert Syst. Appl.*, vol. 65, pp. 194–211, Dec. 2016. doi: 10.1016/j.eswa.2016.08.040.

[49] I. S. Shugurov and A. A. Mitsyuk, "Applying mapreduce to conformance checking," in *Proc. Trudy ISP RAN / Proc. ISP RAS*, vol. 28, no. 2, 2016, pp. 103–122. doi: 10.15514/ISPRAS-2016-28(3)-7.

[50] D. Loreti, F. Chesani, A. Ciampolini, and P. Mello, "A distributed approach to compliance monitoring of business process event streams," *Future Gener. Comput. Syst.*, vol. 82, pp. 104–118, May 2018. doi: 10.1016/j.future.2017.12.043.

[51] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proc. CAiSE*, Essen, Germany, vol. 10253, 2017, pp. 477–492. doi: 10.1007/978-3-319-59536-8_30.

[52] J. Becker, D. Breuker, P. Delfmann, and M. Matzner, "Designing and implementing a framework for event-based predictive modelling of business processes," in *Proc. 6th Int. Workshop Enterprise Modelling Inf. Syst. Archit.*, Luxembourg City, Luxembourg, Sep. 2014, pp. 71–84.

[53] D. Breuker, M. Matzner, P. Delfmann, and J. Becker, "Comprehensible predictive models for business processes," *MIS Quart.*, vol. 40, no. 4, pp. 1009–1034, Dec. 2016. doi: 10.25300/MISQ/2016/40.4.10.

[54] T. Murata, "Petri Nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989. doi: 10.1109/5.24143.

[55] M. de Leoni, J. Munoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Decomposing alignment-based conformance checking of data-aware process models," in *Proc. OTM Conf.*, Amantea, Italy, vol. 8841, 2014, pp. 3–20. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-662-45563-0_1#citeas. doi: 10.1007/978-3-662-45563-0_1.

[56] B. Gao, T.-Y. Liu, G. Feng, T. Qin, Q.-S. Cheng, and W.-Y. Ma, "Hierarchical taxonomy preparation for text categorization using consistent bipartite spectral graph copartitioning," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 9, pp. 1263–1273, Sep. 2005. doi: 10.1109/TKDE.2005.147.

[57] W. Z. Kong, C. H. Sun, S. Q. Hu, and J. H. Zhang, "Automatic spectral clustering and its application," in *Proc. ICICTA*, vol. 1, May 2010, pp. 841–845. doi: 10.1109/ICICTA.2010.164.

[58] G. Li, Q. Ma, H. Tang, A. Paterson, and Y. Xu, "QUBIC: A qualitative biclustering algorithm for analyses of gene expression data," *Nucleic Acids Res.*, vol. 37, no. 15, p. e101, Aug. 2009. doi: 10.1093/nar/gkp491.

[59] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000. doi: 10.1109/34.868688.

[60] W. M. P. van der Aalst *et al.*, "Soundness of workflow nets: Classification, decidability, and analysis," *Formal Aspects Comput.*, vol. 23, no. 3, pp. 333–363, 2011, May 2011. doi: 10.1007/s00165-010-0161-4.

NAN LI received the B.S. degree from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2003, and the M.S. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2006. He is currently pursuing the Ph.D. degree with the Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences. He is currently a Doctoral Candidate with the University of Chinese Academy of Sciences. His current research interests include process mining, workflow net, parallel computing, embedded system, and software engineering.