Transactions on Mass-Data Analysis
of Images and Signals
Vol. 12, No 1 (2021) 3-11
© P-ISSN1868-6451, E-ISSN 2509-9353
ISBN 978-3-942952-88-0

ibai Publishing
www.ibai-publishing.org

Discrete Event Model of Pandemic Data Processing

Calin Ciufudean

Stefan cel Mare University, Suceava, 720229, Romania
ciufudean.calin@gmail.com

Abstract. Big data processing issues is a key response against the critical situations like pandemic one. Processing pandemic data is directly related to control the asynchronous flow of big data therefore this paper is focused on modeling such technical issue using discrete event formalisms, i.e. Petri nets. The techniques of the usual Petri nets require that the designer should be able to represent the Petri structures on the basis of the structural and functional properties of the modeled systems. Applications that are limited to simple systems have already shown that this goal is not easy to accomplish. Therefore, all these approaches have a more academic value, although they provide a schematic framework for the construction of practical models. A Petri net unitary scheme is preferred to represent the overall and control plans for better integration of the final data. A theoretical model is proposed here.

Keywords: Big Data, Pandemic, Petri Nets, Data Slots, Discrete Event Model, Critical Data String.

1 Introduction

Yes, we all know that information, respectively data can save lives. More than ever we prove it nowadays. This means data processing issues is a key response against the critical situations like pandemic one [1 - 4]. Processing pandemic data is directly related to control the asynchronous flow of big data therefore this paper is focused on modeling such technical issue using discrete event formalisms, i.e. Petri nets. We assume that reader is familiarized with the techniques of the most user friendly formalism for modeling and analysis of discrete event systems, i.e. Petri nets, or we refer the reader to [5 - 8]. The techniques of the usual Petri nets require that the designer should be able to represent the Petri structures on the basis of the structural and functional properties of the modeled systems. Applications that are limited to simple systems have already shown that this goal is not easy to accomplish.

Therefore, all these approaches have a more academic value, although they provide a schematic framework for the construction of practical models. An alternative approach is based on an object representation that constitutes at a certain level of ab-

straction invariable, selected functional properties of the represented components or systems. From our observations, the operator of the system tends to focus on the overall functionality of the system, which also involves the functionality of the constituent parts and the interactions between them. Such an approach to the problem is encountered in the case of representation with the help of objects that render typical components of automatic systems. This approach will allow a quick graphical representation and an analysis resulting from this graphical representation.

An optimal modeling will allow the assembly of the system through an interconnection between the available components. The analysis of both general and application-specific properties must be done in such a way as to automatically produce information that is easy to design in the operation of the system. The object representation approach requires a classification of objects for a particular field of application. In addition, emphasis must be placed on the invariable functional properties of these objects represented to some degree of abstraction. This will allow a rapid construction of some system models and, implicitly, an analysis focused on the major functional properties of the system, as well as on its global functioning. Knowledge of the global operation is very useful in the sketch stages of a system, helping to select those alternatives that best meet the functional requirements. The level of detail can be extended in later stages of design, to overlap with specific applications.

Basically, in our approach, we deal with the level of the control plan of an assembly structure of data slots for ensuring real time processing and real time building of the panoramic picture of the on-going process. We mention that our proposed model is an academic, theoretical for instance, one.

We assume that data slots are pieces of the assembly and they are delimited, partitioned, each corresponding to a distinct operation, and therefore the control plan requires information about the system resources which influences the design of the control system. A Petri net unitary scheme is preferred to represent the overall and control plans for better integration of the final data.

We will try to analyze these problems using a Petri net as an analysis and representation scheme for the whole, as follows:

1) We present a Petri net to notice both the overall plans of the ensemble and the representations of the lower levels of control of the ensemble, thus offering a unique, hierarchical representation of the ensemble.

2) We show that our Petri net model, i.e Big Data Petri Nets (BDPN) has the necessary properties from the perspective of the whole.

3) We present a method to build the level of data processing control from its global plans [9, 10].

4) We present a method for analyzing the Petri representation of the control plan in order to optimize the system performance.

The remainder of this paper is organized as follows. In section 2, the proposed BDPN is expatiated, and section 3 analyses the throughput of BDPN. Finally, the conclusions and future development of our system are given in section 4.

2 The BDPN network model

A point in a location of a BDPN indicates that a data slot represented by that place exists in the current state of the whole, and transitions model operations. In the initial state of the BDPN, all locations representing input parts not connected to the Petri net will be marked by one dot. Two sets of places: P_{In} and P_{Out} represent the initial, respectively the output stages of the BDPN. When completing a specific processing task of a data slots, the BDPN will return to the original marking so that a different sequence of tasks can be performed for the next data slot, i.e. the BDPN is a reversible net and that allows performing specific tasks involved in pandemics, such as epidemiological investigations. We introduce the transition t_{Reset} to explicitly model the network reversal from the final marking (M_{fin}) to the initial one (M_0) of the BDPN. Transition t_{Reset} has the set of P_{In} locations as output locations and P_{Out} locations as input locations (stages). The temporal measures of the data loads on the BDPN are modeled by associating a time indicating the duration of the process at each transition. The asynchronous nature of the appearance of the data slots is modeled by attaching duration to each marking. When the BDPN is initialized, the time mark associated with the initial network markup (e.g. corresponding to P_{In} locations) represents the arrival time of the input data slots.

A transition is validated if there is at least one point at each transition entry place. The last time-stamp of these sets of marking points is called the temporized-transition validation time. During the combustion of any transition, the points in the marking that produces the combustion can no longer be used for other transitions, although they remain in place until the combustion is completed. The data resources, i.e. points placed in output locations, have the same time-stamp, which is the time when the combustion is completed and is equal to the validation time at which the combustion duration is added. The notation $d_m(p)$ means the value of the time-stamp associated with a point in the location p.

As the BDPN model is the Petri net vivacity, it will not crash, and the reversibility property ensures that all data slots are unique. Therefore, the BDPN model provides an accurate simulation and representation of processing big data. We will show that BDPN can model the level of controlling big data processing therefore the representation of controls can be constructed correctly in real time using a single, unitary model of the data flow plan, processing data plan, and control plan which eliminates time-consuming translation between representation schemes. Also we mention that optimizes an important issue of mapping an overall plan to generate a control plan with minimal costs while managing issues such as timing and allocation of resources.

We will show that BDPN models control plan by preserving certain desired properties from the perspective of the whole. In a serial asynchrony data system the key indicator may be the data processing rate.

We propose a general model for evaluating the throughput of big data systems. We assume that there is a critical data string in the representation of the control net of the BDPN, responsible for the system throughput.

A critical data string is a sequence of transitions fire starting at a input location of BDPN and ends at a output location. A critical data string determines a bottleneck in the BDPN. In a real system, the critical data string is determined by several factors, such as: the nature of data, the data flux, the duration of data processing including

verification of data sources, etc. We denote by $t_{m(slot)}$ the time required to process a data slot d_{slot} in a BDPN model, i.e. $d_{slot} \in P_{Out}$ location. We consider two types of critical data strings:

a) Critical data string due to a task (CDST).

b) Critical data string due to a subassembly of BDPN (CDSS).

a) Critical data string due to a task (CDST).

The critical data string in a control net may be the sequence of transitions fire whose task (or transition, in terms of Petri nets) is the most time consuming. This is true only if other factors, such as the arrival time of the various data slots, have no effect on the overall time [11, 12].

In a control BDPN the sequence of transitions associated with processing the critical data string is: $t_1, t_2, \ldots, t_q$. We use the notation $t_{crit} = \{t_1, t_2, \ldots, t_q\}$ to denote the critical data string. The time of execution of the tasks in the bottleneck and, from here, the total time of execution of the BDPN is:

$$d_{crit} = d(t_1) + d(t_2) + \cdots + d(t_q) \tag{1}$$

where $d(t_i)$ is the firing time of transition t_i.

b) The critical data string due to a subassembly of BDPN (CDSS).

Alternatively, the critical data string in a BDPN can be defined by the late arrival of a d_{slot} and therefore the tasks affected by this d_{slot} define the critical data string. The throughput of the BDPN is determined by the sequence $t_{i,crit}$, where $t_{i,crit}$ is the sequence of transitions associated with processing the critical data string depending of delayed d_{slot}. The time associated with such a critical data string is denoted $d_{i,crit}$.

3 Throughput analysis of BDPN model

We assume that a critical data string is related to the time required to perform a task (or sequence of tasks) and is not due to the late arrival of the constituent parts (subassemblies). If the critical data string is a CDST, then processing time of a data slot is [13 - 15]:

$$d_{CDST} = d_{crit} + T_0 \tag{2}$$

where T_0 is the initiation time of the control net, i.e. BDPN.

Alternatively, for a CDSS the critical data string may be associated with the delay of the data slot,. If Tpi is the arrival time of part data slot then the processing time is:

$$d_{CDSS} = \max_{1 \le i \le m} (d_{crit} + T_{pi}) \tag{3}$$

where m represents the number of entries in the BDPN (the number of data slots in the system).

Combining relations (2) and (3) the optimal control plan that has the minimum time for processing data is given by the next relation [16]:

$$d_{BDPN\ optim} = \min_{k} max_{1 \le i \le m}(d_{crit} + T_{0i}; d_{crit} + T_{pi}) \quad (4)$$

where k = number of possible control BDPNs.

We consider a BDPN modeling the process of m data slots: p_1, p_2, …, p_m.; each data slot is processed in corresponding m stages, as shown in Fig.1. Each stage has two inputs (except the first one): one input from the system input bus and one input from the previous stage. For the first stage both inputs are from the system bus and are characterized by the group $\{T_{piS}, I_{iqS}\}$, where T_{piS} is the first arrival time of part p_i at stage s of the pipe, I_{iqS} is the interval between the arrivals of two identical parts p_i in iterations (q-1) and q at stage s of the pipe-line. We assume that $I_{i1S} = I_{i2S} = \ldots = I_{imS}$, i.e. the interval between the arrivals of part p_i at stage s is always the same. The BDPN is characterized by:

T_{0S} = initialization time for stage s;

$d_{Written\ q}$ = time required for stage s to execute tasks in iteration q. This is the time associated with CDST tasks for this stage;

d_{Sreset} = reset time of stage s after the completion of the q-th data processing;

L_{qS} = time required to data in iteration q from stage s to stage s + 1;

We make the following simplifying hypotheses:

L_{qS} = L; the transport time of the product being assembled from one stage to the next is the same for all stages and for all iterations of the system;

$d_{Scrit\ q} = d_{Scrit}$; the time required to perform tasks in CDST in stage s is the same, regardless of the iteration q;

$d_{Sreset\ q} = d_{reset}$; the reset time of all stages in the pipe is the same.

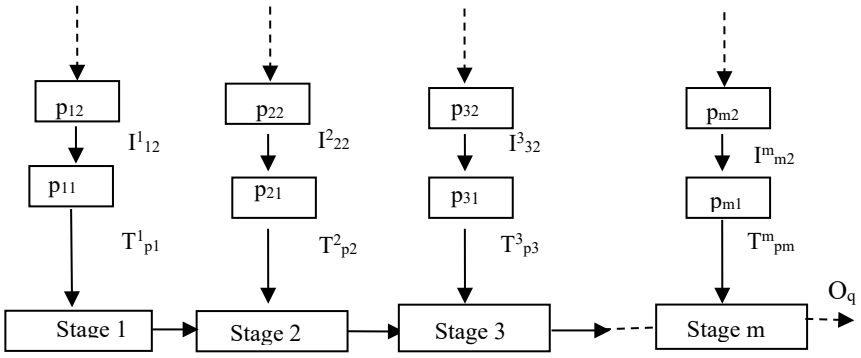

Fig.1. BDPN structure.

Fig. 1 shows the control plan of the BDPN model, where locations p_1, p_2, ..., p_m represent the m data slots of the data processing at each iteration. In the initial marking of the BDPN network, the locations p_1, p_2, ..., p_m correspond to the times T_{p1}, T_{p2}, ..., T_{pm} representing the arrival times of the subassemblies corresponding to the respective stage. All locations T_{0S} will also be marked, the time T_{0S} representing the initialization time for stage s, where s=1, 2, …, m. When we deal with systems which require complex data processing we will merge BDPNs models as the q^{th} iteration of

the system starts in stage s only after the completion of the q^{th} iteration in stage (s-1), i.e. the transition I_{s+1} fires (see Fig.2).

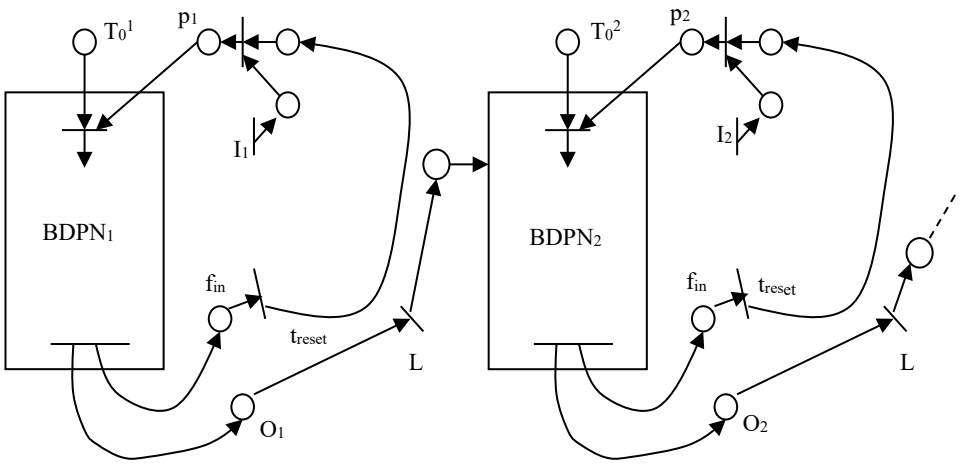

Fig.2. Merging BDPNs models.

Data processing time differs for each stage. We suppose there is a bottleneck in the system, respectively stage h, which requires the longest processing time. The time required for q iterations through stage h defines the processing rate of the system, i.e. stage h defines the critical data string.

The time required to process the q^{th} product is the time required for processing the critical data string plus the time required for the product to pass through the rest of the BDPN [17]:

$$d_m(O_q) = \max_{1 \le h \le m} [d_{h,crit} + (m-h)L + \left(d_{h+1,crit} + \cdots + d_{h(O_{q-1}),crit}\right)] \quad (5)$$

where: ($d_{h(Oq-1),\ crit}$) is the execution time associated with the processing data slot O_q-1 in stage h; (m-h)L = data transfer time interval (after stage h); $d_{h,\ crit}$ = the processing time in stage h.

The critical data string in stage h is a CDSS, because the data processed in the previous stage (h-1) determines the time required for stage h to perform its task.

Because h determines the critical data string, the data transfer time before stage h is irrelevant.

Next, we will determine the processing time for three possible cases, and we will assume that stage f appears earlier than stage h. The search space is reduced to only those situations that cause a bottleneck in the system. We can have three situations:

a) The initial narrow place in the system is caused by the time required to initialize the stage f. After initialization, the next narrow place is given by the data processing time in critical stage h. The processed data O_q is delivered by the stage h at the time [18]:

$$d_m(O_q) = \max_{1\le f\le h;1\le h\le m}[T_{0f} + d_{f,crit} + (h-f)L + (q-1)d_{h-1,crit} + (d_{f+1,crit} + \cdots + d_{h,crit})] \quad (6)$$

where: (h - f)L is the data transfer time interval.

Obviously, the reset duration of stage h (noted by the d_{reset} transition) must be taken into account. The initial narrow spot caused by stage f is $T_{0f} + d_{f,\,crit}$. For stage f the critical path is a CDST.

b) The initial narrow place in the system is due to the late arrival of a part at a stage f. At stage f the critical data string is therefore a CDSS. After the arrival of this data, the next narrow place is caused by the time required for assembly at stage h. We assume that z, which is one of the m stages of the system, is the input for stage f. From relations (5) and (6), we have:

$$d_m(O_q) = \max_{1\le z\le f;1\le f\le h;1\le h\le m}[T_{0z} + d_{z,crit} + (m-f)L + (q-1)d_{reset} + (d_{f+1,crit} + \cdots + d_{h,crit})] \quad (7)$$

c) The delay is completely due to the long time interval for the arrival of data slot z at stage f. The data processing line stops and waits for the arrival of the data slot at stage f. The critical data strings are all CDSS and from (5), we have:

$$d_m(O_q) = \max_{1\le z\le f;1\le f\le m}[T_{0z} + d_{z,crit} + (m-f)L + (q-1)I_{q-1,f} + (d_{f+1,crit} + \cdots + d_{m,crit})] \quad (8)$$

From formulas (6) ÷ (8), the first assembled product is delivered at time shown in the next relation:

$$O_1 = \max_{1\le z\le f;1\le f\le m}(T_{0f} + d_{f,crit}; T_{0z} + d_{z,crit}) + (m-f)L + (d_{f+1,crit} + \cdots + d_{m,crit}) \quad (9)$$

For the n^{th} system iterations, the data processing rate is:

$$Rate = [d_m(O_n) - d_m(O_1)]/(n-1) \quad (10)$$

Substituting the relations (7) ÷ (9) in the expression (10) we obtain the following data processing rate for case a) and b) above:

$$Rate = \max_{1\le h\le m} d_{h,crit} + d_{reset} \quad (11)$$

For case c):

$$Rate = \max_{1\le z\le f;1\le f\le m} I_{fz} \quad (12)$$

It follows from (11) and (12):

$$Rate = \max_{1\le z\le f;1\le f\le h;1\le h\le m}[I_{fz};(d_{h-1,crit}+d_{reset})] \quad (13)$$

The assembly rate for a given control plan is given by the relation (13). If there are k plans, the plan with the maximum rate will be the one that will deliver the O_q processed data in the minimum time. This is easily deduced from the relations (6) ÷ (8).

4 Conclusion

We proposed a method of simulation and optimization of a big data system similar to pandemic data processing with the help of new class of Petri nets: Big Data Petri Nets (BDPN), which achieve the correct integration of subassemblies, so that the representation of control plans can be done directly from the model of the overall plan, thus guaranteeing an efficient coordination of the pandemic data process. As we have mention we deal with time intervals, i.e. we deal with time intervals different from one medical care unit to another one. We recommend, if our theoretical approach will be considered to be practical implemented, to adopt it to specificity of each hospital, clinic etc.

We also presented the technological process as a set of operations that can be improved by minimizing their execution time and, especially, by achieving an optimal variant in terms of logical interdependence between them, which results in the conditions of succession or simultaneity in the process of data processing.

Further development of the proposed approaches for pandemic data processing will be focused on developing fuzzy cooperative BDPN models.

References

1. Perner,P.: Maintenance of Medical and Engineering Systems by Big Data, Brescia Italy, Dipartimento di Matematica e Fisica, Università Cattolica del Sacro Cuore, Big Data 2019, (2019).
2. Perner,P.: New Ways to present the Results of a Data Mining Process to an Expert, Big data analysis and Data mining, June 20-21, (2018).
3. Perner, P.: Health Monitoring by an Automated System for Biological Hazardous Material in the Air, 2nd International Conference on Vision, Image and Signal Processing, ICVISP 2018, Las Vegas, USA, Aug. 27-28, (2018).
4. Perner, P.: Data Mining on Multimedia Data, Lecture Notes in Artificial Intelligence, Vol. 2558. Springer Verlag, Berlin Heidelberg New York, (2002).
5. Petri, C.A., Reisig, W.: Petri net, Scholarpedia, 3 (4): 6477, (2008).
6. Murata, T.: Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, **77** (4): 541–558, (1989).
7. Gao, X., Hu, X.: A Petri Net Neural Network Robust Control for New Paste Backfill Process Model, IEEE Access, vol.8: 18420–18425, (2020).
8. Koch, I., Reisig, W., Schreiber, F.: Modeling in Systems Biology - The Petri Net Approach. Computational Biology, 16, Springer, ISBN 978-1-84996-473-9, (2011).
9. Zhang, A., Wang, J., et al.: Data-Driven Computational Social Science: A Survey, Big Data Research, Elsevier, Volume 21, pages 100-145, (2020).

10. Neilson, J., Daniel, B.: Systematic Review of the Literature on Big Data in the Transportation Domain: Concepts and Applications, Big Data Research, Volume 17, Pages 35-44, (2019).
11. Bauer, D., Froese, F., et al.: Building and Operating a Large-Scale Enterprise Data Analytics Platform, Big Data Research**,** Volume 23, (2021).
12. Al-Jarrah, O. Y., Yoo, P. D., et al.: Efficient Machine Learning for Big Data: A Review, Big Data Research, Volume 2, Issue 3, Pages 87-93, (2015).
13. Lee, J. G., Kang, M.: Geospatial Big Data: Challenges and Opportunities, Big Data Research, Volume 2, Issue 2, Pages 74-81, (2015).
14. Jain, S., Shao, G., Shin, S. J.: Manufacturing Data Analytics Using a Virtual Factory Representation, International Journal of Production Research 55(18), pp. 5450-5464, (2017).
15. Bartlett, M. S., Stein, N., et al.: Big Data on the Shop-Floor: Sensor-Based Decision-Support for Manual Processes, Journal of Business Economics 88(5), pp.593-616, (2018).
16. Thomas, J.P., Nissanke, N., Baker, K.D.: A hierarchical Petri net framework for the representation and analysis of assembly, IEEE Transactions on Robotics and Automation, (1996).
17. Thomas, J.P., Nissanke, N.: Model for throughput evaluation in assembly manufacturing systems, Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, ETFA'95, (1995).
18. Van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer, ISBN 978-3-662-49850-7, (2016).