An Approach for Performance Evaluation of Batch-sequential and Parallel Architectural Styles

Golnaz Aghaee Ghazvini
Computer Engineering Department,
MSc student, Young Research Club,
Islamic Azad University,
Njafabad Branch,
Esfahan, Iran
Aghaee.golnaz@sco.iaun.ac.ir

Sayed Mehran Sharafi
Computer Engineering Department,
Islamic Azad University,
Najafabad Branch,
Esfahan, Iran
Mehran_sharafi@iaun.ac.ir

Sima Emadi
Computer Engineering Department,
Islamic Azad University,
Maybod Branch,
Yazd, Iran
emadi@maybodiun.ac.ir

Abstract— **Software architecture is considered one of the most important indices of software engineering today. Software Architecture is a technical description of a system indicating its component structures and their relationships, and is the principles and rules governing designing. Software Architecture can be utilized to materialize most of the important quality attributes in the system; and these qualities should be evaluated at architectural level. Therefore, to what extent software architectural design has been successful depends on the quality attributes of the system. One of the most important quality attributes is the performance. Usually an architect takes into consideration in software architectural design is to use software architectural styles. An architecture style is a set of principles which an architect uses in designing software architecture. Since software architectural styles have frequently been used by architects, these styles have a specific effect on quality attributes. If this effect is measurable for each existing style, it will enable the architect to evaluate and make architectural decisions more easily and precisely. In this paper an effort has been made to introduce a model for investigating this attribute in Parallel and Batch-sequential styles. So, our approach initially models the system as Discrete Time Markov Chain or DTMC, and then extracts the parameters to predict the response time of Batch-sequential and Parallel style. Then, in order to evaluate the systems whose architectures include both styles, we will generalize our model.**

keywords-Software architecture; Discrete Time Markov Chain; Batch-sequential style;Parallel style; Performance attribute;

I. INTRODUCTION

Component-Based software engineering provides an opportunity for better quality and increased productivity in software development by using reusable software components [3]. One of the most important quality attributes in software architecture is performance. Early performance analysis and measurement approaches for a component-based software system can help software architects to evaluate their systems based on the performance specification created by component developers [4]. During the last decades, there have been many approaches for evaluating the performance attributes of component-based systems. These approaches have been classified into formal and informal models. Classical formal models such as queuing networks [3], stochastic process algebras [6] stochastic Petri nets [7] and automata [8] can be used to model and analyze component-based software systems. However, these approaches do not specifically consider performance evaluation of architectural styles using Markov chain. An architectural style is a combination of architectural constraints that restricts the roles / features of architectural components and allows relationships among these components within any architecture that conforms to that style. [9] Architects use software architectural styles in designing software architecture. Common styles are Batch-sequential, Pipe and Filters, Call and Return and also Fault tolerance. In a batch-sequential style, components are executed in a sequential manner. This means that only a single component is executed in any instance of time. For example, a bank performs a batch of transactions update to a master file in sequence. A parallel style has a set of components running concurrently; a fault tolerant style has a set of back-up components compensating for the failure of the others; call and return style has some components, calling the other components at an indefinite number of times [10, 11]. In this paper two of the most common styles used in the architectural design of most systems were selected, and a model is offered to evaluate the performance attribute in these styles quantitatively. Then, in order to evaluate the systems whose architectures include both styles, we will generalize our model. Our approach consists of modeling the software architecture as a Discrete Time Markov Chain (DTMC), and the DTMC model is then analyzed to get performance attributes of the systems. The rest of the paper is divided as follows: section II introduces performance evaluation of Batch-sequential and Parallel styles. Section III illustrates a software architecture that contains sequential and parallel architectural styles. Example and future works are presented in section IV and V.

II. PERFORMANCE EVALUATION OF BATCH-SEQUENTIAL AND PARALLEL STYLES:

In this section, considering the multiplicity of performance parameters, the parameter of 'response time' which is one of the most important parameters has been

978-1-4244-8371-6 /10/$26.00 © 2010 IEEE

selected. The model is offered for quantitative evaluation of this parameter in architectural styles.

The state model in this paper is based on Discrete Time Markov Chains (DTMC), so we Discuss Markov process and Discrete Time Markov Chains which is use to model the software of a system. Markov process is a stochastic process whose dynamic behavior is such that probability distributions for its future development depend only on the present state and not on how the process arrived in that state. [1,11] Let $\{x_k\}$ be a discrete time stochastic process which takes on values in a countable sets, called the state space.$\{x_k\}$ is called a Discrete Time Markov Chain (or simply a Markov chain , when the discrete nature of the index is clear) if :

$$P(X_K = i_k \mid X_{K-1} = i_{k-1}, X_{K-z} = i_{k-z}, \ldots) = P(X_K = i_k \mid X_{K-1} = i_{k-1})$$

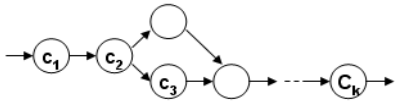
Where $i,j \in s$. For an application consists a number of components, we can present its software architecture using a DTMC .the state of the DTMC at an execution step is given by the component in execution of that step. Transitions between states represent transfer of control from one component to another.

A. Batch-Sequential Style

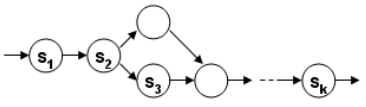
In a batch-sequential style, components are executed sequentially, In this type of architecture style, only one component is executed at any instance of time, the control flow is transferred to only one of its successors upon the completion of a component[10].

One of the examples of this style is modeled in fig.1 (a), where $c_1,c_2 \ldots c_k$ are software components in a machine, component c_2 transfer flow control to one of its branches subsequent components.

The transformation from the architecture to state model can be viewed as a mapping of one component to one state. The state model is shown in fig.1 (b).



(a)



(b)

Figure1 (a) batch-sequential style, (b) state model

For determining the model parameters, we assume that transition between adjacent components is possible. We could calculate the number of visit to each of the state i starting from state 1. By $V_i = q_i + \sum p_{ki}.v_k$ where, q_i is the probability of starting in state i,[4,13]

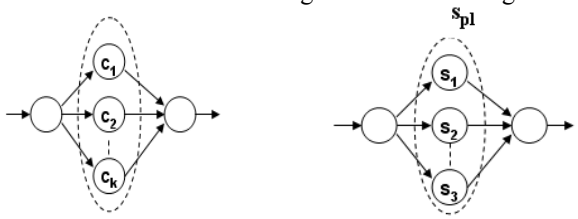
- The service time required to service one request by a software component that is using a standard Cpu and Disk are shown by cpu (i) and disk (i).
- m is the number of components on a machine.
- f_c and f_d are the rating factors of the Cpu and Disks respectively of each machine, which are present in the system [13].

Finally, we calculate the total Cpu and Disk service time given by equation 1:

$$Cpu - time = f_c \sum_{i \in m} v(i).cpu(i)$$
$$Disk - time = f_d \sum_{i \in m} v(i).disk(i) \quad (1)$$

B. Parallel Style

Parallel style has multiple components running concurrently, and in this way service time required is reduced An example of this style is shown in figure 2(a), where components $c_1,c_2 \ldots c_k$ in the dotted oval are running concurrently. These components cooperatively work on a partition of outputs produced by previous component, and synchronously release the control to the next subsequent component. Figure 2(b) shows the state model of figure 2(a). A set of cooperative concurrent components in software architecture is modeled to one single state in state diagram.



(a) (b)

Figure2 (a) parallel style (b) State model

Considering that all the parallel components are modeled to only one state s_{pl}, thus the visit count to state s_{pl} is calculated separately, through the equations 2:

$$v(s_{pl}) = q_{spl} + \sum p_{k.spl} v_k \quad (2)$$

We also consider the variable $T_{communication}$ to indicate the time spent on communication synchronization components that are executed in parallel. Finally for all parallel components, we can calculate the total Cpu and Disk service time given by equation 3:

$$Cpu - time = f_c[v(s_{pl}) \underset{i \in m}{MAX}(cpu(i)) + T_{communication}]$$
$$Disk - time = f_d[v(s_{pl}) \underset{i \in m}{MAX}(cpu(i)) + T_{communicat\,ion}] \quad (3)$$

III. SOFTWARE ARCHITECTURE FOLLOWING BATCH-SEQUENTIAL AND PARALLEL STYLE:

The application usually can not be performed completely in parallel, and the parts of the application should be forced to be implemented sequentially, therefore we assume a selective architecture, which is a combination of parallel and sequential styles. This means to run the program two kinds of machines will be used: a machine which contains parallel components, and the other one which contains sequential components. However, formation of these machines can change, considering the desired application. We assume that parallel components are allocated on a single machine, and any other components are allocated on a separate single machine, or if some components are allocated on a machine, all are executed in order and not run at the same time. We model the software system that combines Parallel and Sequential architecture using a DTMC [1]. A state can represent either a single component execution or a set of concurrent components executions. These concurrent components that execute on the machine h, cooperatively work on a partition of outputs produced by component c_{k-1} and synchronously release the control to the next initial component on the next machine. Components that are executed on sequential machines are assigned to individual states, and the set of concurrent components that are executed on parallel machine is modeled into one single state. Fig.3 (a) shows the architecture and Fig.3 (b) shows the DTMC model of fig.3 (a).

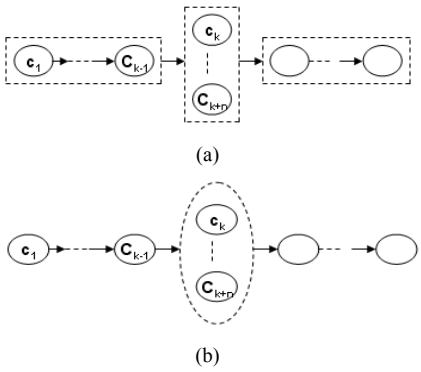


Figure3 (a) Architecture view (b) state model

In this architecture only the transitions between adjacent components are possible. On parallel machine h, the components are executed in parallel, so the required Cpu and Disk service time they needed will overlap; the maximum service time of them will eventually be considered. But in machine j≠h, total service time is considered, when components are executed sequentially. Cpu and Disk service time for this architecture is given by equation 4:

$$CPU-time(j)=\begin{cases} f_{cj}\sum_{i\in m} v(i).CPU(i) & j\neq h \\ f_{cj}[V_{spl}\ \underset{i\in m}{MAX}(CPU(i))+T_{com}] & j=h \end{cases}$$

$$Disk-time(j)=\begin{cases} f_{dj}\sum_{i\in m} v(i).Disk(i) & j\neq h \\ f_{dj}[V_{spl}\ \underset{i\in m}{MAX}(Disk(i))+T_{com}] & j=h \end{cases} \quad (4)$$

Also to find the mean time to complete of the application or response time, we consider μ_{cpui} as the mean Cpu time spent in state i and μ_{diski} as the mean Disk time in that state , E[cpu(i)]= μ_{cpui} , E[disk(i)]=$\mu_{disk\ i}$ and E[v_i]=x_i.

In the equation 5, we have not regarded $T_{Communications}$. So the following represent the expected mean time to complete of the application.

As v(i).cpu(i) is a function of random variables v(i) and cpu(i), so v(i).cpu(i), is a random sum, by using the expression for random sums.

$$E[v(i).cpu\ (i)] = E[v(i)].E[cpu\ (i)] = x_i \mu_{cpui}$$

$$E[cpu-time(j)]=\begin{cases} f_{cj}\sum_{i\in m} X_i.\mu_{cpui} & j\neq h \\ f_{cj}.v_{spl}\ \underset{i\in m}{MAX}(\mu_{cpui}) & j=h \end{cases}$$

$$E[Disk-time(j)]=\begin{cases} f_{dj}\sum_{i\in m} X_i.\mu_{diski} & j\neq h \\ f_{dj}.v_{spl}\ \underset{i\in m}{MAX}(\mu_{diski}) & j=h \end{cases} \quad (5)$$

IV. AN EXAMPLE

An example of a component-based system that contains Batch-sequential and Parallel styles is used to validate the correctness of the above performance model. The architecture of this system is shown in fig(4)a ,with a total of 9 components, in this architecture ,components in the dotted oval run in parallel and are allocated on machine h, whereas the components in the dotted rectangle run sequentially and are allocated on the other sequential machines. We transform those identified architectures into state model. Since the component-to-state mapping can be many-to-one or one-to-one mapping, the total number of state in the state model can be different from the total number of component in the architecture. State model of this system shown in fig(4).b. Sequential components c_1 ,c_2 ,c_3 ,c_4 are mapped to separate state s_1,s_2,s_3,s_4 ,and parallel components c_5,c_6,c_7 are only mapped to one state s_5. Components c_8,c_9 are mapped to separate states.

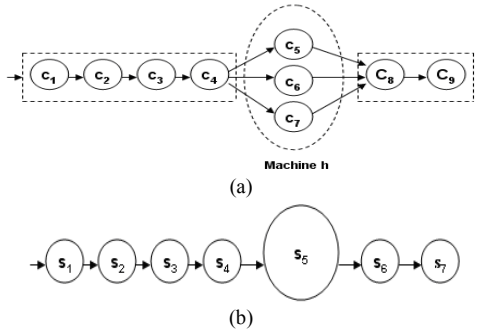


Figure4 (a) Architecture view (b) state model

The data about the software architecture is summarized in table1.The expected time spent by the application in component i per visit is already known, this time can either be obtained experimentally or may be known a priori. The expected number of visits to each state can be computed by solving the following system of linear equations, where q_i is the probability that the application starts in component i.

$V(i)=q_i +\sum p_{ki}\ v_k$

In this software architecture, we presume a transition between adjacent components are possible, and a transition to any component in a parallel component set is basically to the whole set. Therefore, the transition probability between adjacent states is equal to 1.In the following; we calculated the number of visits to state 1:

$$V_1 = q_1 + \Sigma\ P_{kl} V_k = 1 + 0 = 1$$

TABLE1.EXAMPLE SYSTEM EXECUTION BEHAVIOR

No .of Machines	3	No.of components onMc_2	3
No. of components on Mc_1	4	No.of components onMc_3	2
CPU time spent in components(in secs)			
CPU time in component$_1$	0.01	CPU time in component$_6$	0.30
CPU time in component$_2$	0.03	CPU time in component$_7$	0.20
CPU time in component$_3$	0.005	CPU time in component$_8$	0.04
CPU time in component$_4$	0.02	CPU time in component$_9$	0.01
CPU time in component$_5$	0.01		
Disk time spent in components(in secs)			
Disk time in component$_1$	0.01	Disk time in component$_6$	0.001
Disk time in component$_2$	0.003	Disk time in component$_7$	0.01
Disk time incomponent$_3$	0.002	Disk time in component$_8$	0.005
Disk time in component$_4$	0.02	Disk time in component$_9$	0.02
Disk time in component$_5$	0.01		
visit count to each of the state			
No. of visit to state 1	1	No. of visit to state 5	1
No. of visit to state 2	1	No. of visit to state 6	1
No. of visit to state 3	1	No. of visit to state 7	1
No. of visit to state 4	1		
Raring factor		**Time to synch**	
Rating factor(fcj) of CPUs	1	Synch time for parallel component	0.02

$$CPU-time(1) = fcj \sum_{i \in 4} v(i)CPU(i) = 1(0/01 + 0/03 + 0/005 + 0/02) = 0/065$$

$$Disk-time(1) = fdj \sum_{i \in 4} v(i).disk(i) = 1(0/01 + 0/003 + 0/002 + 0/02) = 0/035$$

As the formula in the previous section shows, we use MATLAB software to estimate the performance of this system with 9 components .The CPU and Disk-time for machine 1 are calculated as above, and service time required for the other machines is summarized in table 1. Finally, the total service time required for the application in this system is 0.795 sec.

TABLE2. CPU TIME (IN SECS)

Service time spent in machine 1	0.1
Service time spent in machine 2	0.62
Service time spent in machine 3	0.075
Time to service one request in this system	0.795

V. CONCLUSION AND FUTURE WORK

In this paper, we discussed an analytical approach for performance evaluation of systems following Sequential and Parallel styles. For this purpose, we first constructed a DTMC model of the software system; this model provides us

with the visit counts to different states and can calculate the total service time (responsiveness) of the software system. Architectural styles have a specific effect on quality attributes and are used in designing software architecture .In this paper ,we focused on performance evaluation in sequential and parallel styles; Our future research includes performance evaluation on the other styles, such as Fault tolerance style and also Call and Return style.

REFERENCES

[1] K. Kant, "Introduction to computer system performance evaluation", McGraw-Hill, 1992vol..32, dec.1993.

[2] L.Bass, P.Clements, R.kazman, "Software Architecture in Practice",SEI Series in Software Engineering, Addison-Wesley,1998

[3] X.Wu, M. Woodside, "Performance Modeling from Software components", proceedings of the Fourth International Workshop on Software and Performance, vol. 29, Jan. 2004, pp.290-301.

[4] S. Gokhalea, , W. Eric Wong, J.R. Horganc and S.Trivedi, "An analytical approach to architecture-based software performance and reliability prediction", An International Journal of performance evaluation,ELSEVIER, vol 58, Dec.2004, pp. 391-412, doi:10.1016/j.peva.2004.04.003.

[5] S. Balsamo, V.D.N. Persone and P.Inverardi, "A review on queuing network models with finite capacity queues for software architectures performance prediction", An International Journal performance evaluation, ELSEVIER, vol. 51, Feb. 2003, pp.269-288, doi:10.1016/S0166-5316(02)00099-8.

[6] H. Hermanns, U.Herzog and J.P. Katoen, "Process algebra for performance evaluation", An International Journal Theoretical Computer Science, elsevier vol. 274, Mar. 2002, pp.43-87, doi:10.1016/j.peva.2009.07.007

[7] S. Emadi and F. Shams, "From UML component diagram to an executable model based on Petri Nets", Proc. the Third International Symposium on transformation Technology, Aug.2008, pp. 2780-2787, doi:10.1109/ITSIM.2008.4631945.

[8] M. sharafi, " Extending Team Automata to Evaluate Software Architectural Design", Proc. 32nd Annual IEEE International Computer Software and Applications Conference,(2008) , pp. 393-400

[9] K. Khodamoradi, J. Habibi and A. Kamandi, "Architectural Styles as a Guide for Software Architecture Reconstruction",13 th International CSI computer Science, Kish Island, Persian Gulf, Iran, (2008)

[10] W. Wang, D. Pan and M. Chen, "Architecture-based software reliability modeling", Journal of System and Software, vol. 79, Jan. 2006, pp. 132-146, doi: 10.1016/j.jss.2005.09.004.

[11] S. Ramamoorthy and S. P. Rajagopalan, "Component-Based Heterogeneous Software Architecture Reliability(COHAR) Modeling", International Journal on Computer Science and Engineering, vol. 02, 2010, pp. 1280-1285

[12] V .S. Sharma and K. S. Trivedi, "quantifying software performance, reliability and security: An architecture-based approach", Journal of Systems and Software, elsevier, vol. 80, 2007, pp. 493-509, doi:10.1016/j.jss.2006.07.021.

[13] V. S. Sharma, P. Jalote and K. S. Trivedi, "Evaluating Performance Attributes of Layered Software Architecture", Springer-Verlag Berlin Heidelberg, CBSE, LNCS, vol 3489, 2005, pp.66-81, doi: 10.1007/11424529_5.