Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Fork-Join program response time on multiprocessors with exchangeable join*

WANG Yong-cai†, ZHAO Qian-chuan, ZHENG Da-zhong

(*Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing 100084, China*)

†E-mail: wangyongcai@mails.tsinghua.edu.cn

Received April 2, 2005; revision accepted Jan. 9, 2006

Abstract: The Fork-Join program consisting of K parallel tasks is a useful model for a large number of computing applications. When the parallel processor has multi-channels, later tasks may finish execution earlier than their earlier tasks and may join with tasks from other programs. This phenomenon is called exchangeable join (EJ), which introduces correlation to the task's service time. In this work, we investigate the response time of multiprocessor systems with EJ with a new approach. We analyze two aspects of this kind of systems: exchangeable join (EJ) and the capacity constraint (CC). We prove that the system response time can be effectively reduced by EJ, while the reduced amount is constrained by the capacity of the multiprocessor. An upper bound model is constructed based on this analysis and a quick estimation algorithm is proposed. The approximation formula is verified by extensive simulation results, which show that the relative error of approximation is less than 5%.

Key words: Exchangeable join, First come first served (FCFS), Fork-Join, Multiprocessor, Response time
doi:10.1631/jzus.2006.A0927 **Document code:** A **CLC number:** TP3

INTRODUCTION

With the prevalence of distributed computing and parallel programming languages (Barry and Allen, 1998), performance evaluation of the parallel execution systems becomes important. In this work we derive bounds and an approximation of the mean response time of a particular type parallel program: program with Fork-Join tasks and executed in multiprocessor with first come first served (FCFS) policy. This kind of program is general in large-scale simulation and numerical computation, when the individual computing tasks have no direct meaning, until the finishing of all the parallel components of a program. Early studies mainly focused on the Fork-Join programs in single processor systems. For the correlation introduced by the join operation and the explosion of the state space, only for programs with two tasks, analytical distribution was obtained (Flatto, 1985; Wright, 1992). For programs with more than two tasks, various bounds or approximation techniques were proposed. These results provide effective performance estimation methods for single processor Fork-Join systems. Recently, multiprocessor Fork-Join systems attract more and more studies due to the popularization of the multithreading parallel programming. Multiprocessor system with processor sharing policy was studied in (Towsley *et al*., 1990) and multiprocessor system with FCFS policy was studied in (Baccelli and Liu, 1990; Ko and Serfozo, 2004). But these studies did not reveal two important properties of the multiprocessor systems, which make them essentially different from the single processor systems. These properties are named "Exchangeable Join" (EJ) and "Capacity Constraint" (CC) in this paper. Fig.1 shows the behavior of EJ and CC.

Fig.1 shows an instant when the multiprocessor system processes sequential Fork-Join programs with

* Project supported by the National Natural Science Foundation of China (Nos. 60274011 and 60574067), and the Program for New Century Excellent Talents in University (No. NCET-04-0094), China

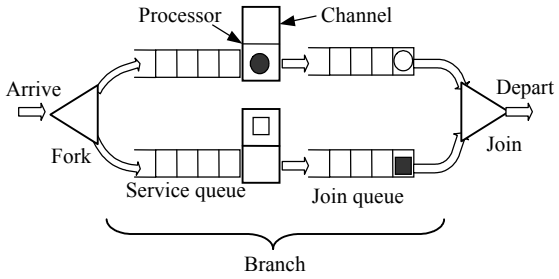

Fig.1 Exchangeable join behavior in Fork-Join system with two processors with each processor having two channels

FCFS policy. The system has two branches. Branch means the parallel path in which the tasks are processed. Two Fork-Join programs: a white one and a black one arrive sequentially and are split into tasks at the fork point. We indicate the tasks in the first branch by circles and in the second branch by squares to clearly express them. Each branch contains a multiprocessor with every multiprocessor having two channels. Because of the randomness of task execution, the black task in the second branch may finish its service earlier than the white square, and will join with the finished white circle and depart as a finished program. So the white and black tasks have exchanged their partners. We call this phenomenon "Exchangeable Join" (EJ), which cannot appear in single processor systems. Another character is that the exchangeable join is limited by the number of channels. In Fig.1, since each multiprocessor contains only two channels, a new arrival task has to wait if all the two channels are occupied by the early tasks, so it can never exceed more than two prior tasks. We call this character of multiprocessor system "Capacity Constraint" (CC). In single processor system, CC is extreme in the sense that a later task can never finish execution earlier than any prior task.

In this paper, we analyze how these two characters affect the mean response time of the Fork-Join programs. We introduce an upper bound model and derive an analytical approximation formula for the mean response time T. Because of the exchangeable join, the join process and the service process are dependent. The mean response time can no longer be easily formulated as the sum of the mean durations of these two phases. We apply sample path (Taha and Stidham, 1999) based analyzing method to deal with the correlation difficulty (Nelson and Tantawi, 1988) and express the mean response time of multiprocessor system. We prove that EJ can effectively reduce Fork-Join program response time, but the exchangeable level is restricted by the capacity of the multiprocessors. Based on these findings, a single processor system is appropriately constructed and is proposed as the upper bound model, which is proved by the sample path comparison. After feature analysis of the bounding error, an approximation formula is proposed for the mean response time.. Extensive simulations are performed to verify the bounds and the approximation results, which showed that the relative approximation error is less than 5 percent for various parameter settings.

The paper is organized as follows. In Section 2, we formulate the expression of the mean response time by sample path analysis; in Section 3, the effects of EJ and CC are analyzed and an upper bound model is proposed and proved; in Section 4, after feature analysis of the bounding error, an analytical approximation formula is proposed; extensive numerical results are presented to validate the bounds and approximation formula in Section 5; conclusion and future research work directions are given at the end of this paper.

FORMULATION OF THE RESPONSE TIME

We consider a system with K multiprocessors with every multiprocessor containing c identical channels. Fork-Join programs enter the system according to a Possion process with parameter λ. Every program is forked into K tasks and the ith task is processed independently by the ith multiprocessor with FCFS policy. The service time of a task is an exponential random variable with distribution parameter μ.

We are interested in the execution time of the program. According to the EJ of the tasks, the response time can no longer be simply formulated as the time interval from the program's arrival to the program's departure. Fig.2 illustrates the sample path of the ith Fork-Join program executed in the two-branches-two-channels system as shown in Fig.1. Let a_i be the time point when the program arrives at the system. At the same time, the program is split into

two tasks. At time points $b_{i,j}$, $c_{i,j}$ and $d_{i,j}$ (j=1,2), the ith task in the jth branch begins its service, finishes service and departs respectively. $w_{i,j}=b_{i,j}-a_i$ is the waiting time of the task and $s_{i,j}=c_{i,j}-b_{i,j}$ is its service time of the task. The sojourn time of the task in the jth branch is the sum of the waiting time and the service time.

$$S_{i,j}=w_{i,j}+s_{i,j}. \quad (1)$$

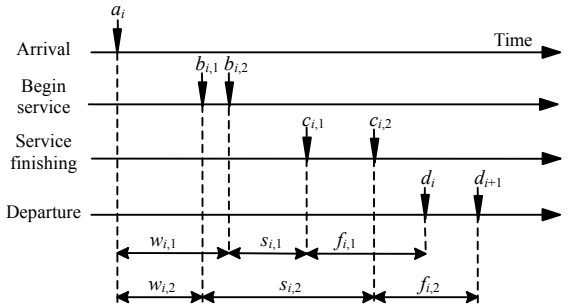

Fig.2 Sample path of Fork-Join program execution process

The distribution of task sojourn time can be calculated by queueing theory (Medhi, 1990). Since $d_i \neq d_{i+1}$, tasks split from the same program depart separately. The arriving program and the departure program are not identical in component. $f_{i,1}=d_i-c_{i,1}$ and $f_{i,2}=d_{i+1}-c_{i,2}$ are the waiting times in the join queue. To formulate the response time of a program, we take the time interval between the ith arriving program and the ith departure program as the response time of the ith program, denoted by T_i, as shown in Fig.3.

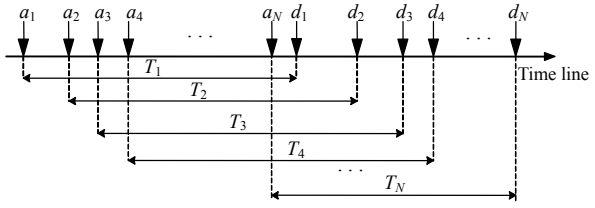

Fig.3 Formulation of the response time of programs

Given all the sample paths a_i, $b_{i,j}$, $c_{i,j}$ and $d_{i,j}$ of the N programs (i=1,...,N; j=1,...,K), to express the mean response time, we need to reorder the tasks finishing time sequence of every branch into ascending order to describe the real task finishing sequence, denoted by $\{\hat{c}_{1,j},\hat{c}_{2,j},...,\hat{c}_{N,j}\}$ (j=1,2,...,K). The mean response time of programs can be formulated as

$$E(T)=\lim_{N\to\infty}\frac{1}{N}\left(\sum_{i=1}^{N}\max(\hat{c}_{i,1},...,\hat{c}_{i,K})-\sum_{i=1}^{N}a_i\right). \quad (2)$$

In Eq.(2), $\max(\hat{c}_{i,1},...,\hat{c}_{i,K})$ is the finishing time of the ith program. Since Eq.(2) contains reordering and maximizing operations, and the task finishing times are associated (Nelson and Tantawi, 1988), $E(T)$ is difficult to calculate.

UPPER BOUND MODEL

Denote the multiprocessor Fork-Join system as P_c^K, where c is the capacity of the multiprocessors and K is the number of branches. Since the mean response time of P_c^K is very difficult to obtain, we try to build relations between P_c^K and some familiar systems.

Exchangeable join and pessimistic bound

One of these familiar systems is similar to P_c^K. It has all the same structure and parameters as P_c^K, except that the EJ is not allowed. We denote this system U_c^K.

Using common random generator technique (Glasserman and Yao, 1992), the same task service finishing time sequences can be generated for both P_c^K and U_c^K : $\{c_{i,k}\}$ (i=1,2,...,N; k=1,2,...,K). The difference between the response time of P_c^K and that of U_c^K is only caused by the join process. With EJ, tasks in P_c^K need not wait for their siblings. The waiting time at the join queue can be saved. Fig.4 gives an illustration. It shows the service finishing time sequence of a system with three branches and six programs. Without EJ, the sum of departure times is 207 compared to 197 with EJ. The exchange of $c_{1,3}$ with $c_{2,3}$ and the exchange of $c_{4,1}$ with $c_{5,1}$ have reduced the sum of departure times 2 and 8 respectively. U_c^K

evidently has larger mean response time than P_c^K : $T_u \geq T$. In view of this, we point out an important character of the EJ behavior.

Theorem 1 Given sample path of task finishing time, every time an exchange in join occurs, the system mean response time will be reduced to less than that without the exchange in join.

Proof of Theorem 1 can be found in Appendix A.

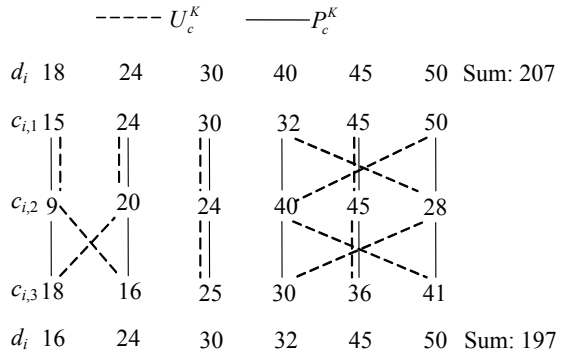

Fig.4 Effects of exchangeable join on response time

Capacity constraint and exceeding probability

Exchange in join will occur if a later task finishes its service earlier than tasks that arrived earlier. This can be recognized as the early tasks are exceeded by the later task. But what is the probability that this occurs? Can a later task exceed all the tasks before it? To answer these questions, we define the "Exceeding Probability".

Definition 1 Exceeding Probability: the probability that a task finishes its service earlier than one or more of its prior tasks.

Exceeding probability is denoted by P_e. P_e determines the frequency of occurrence of exchanges in join. Since the task cannot be executed when all the channels are occupied, P_e is constrained by the multiprocessor's capacity. Let us denote $P_e(X=n;c)$ as the exceeding probability when a task finds there are n customers in the processor with capacity c and denote $P_e(c)$ as the overall exceeding probability for the processor with capacity c. Since a new task cannot be executed unless there is an idle channel in the multiprocessor, $P_e(X=n;c)$ can be derived based on the memoryless property of the exponential distribution.

$$P_e(X=n;c)=\begin{cases}1-(1/2)^n, & n\leq c-1,\\ 1-(1/2)^{c-1}, & n>c-1.\end{cases} \tag{3}$$

It is easy to see that $P_e(X=n;c+1)\geq P_e(X=n;c)$. The system with larger capacity has larger exceeding probability when systems are in the same state. The stationary probability that there are n tasks in the multiprocessors can be calculated by the queueing theory:

$$P(X=n;c)=\begin{cases}\dfrac{\rho^n c^n}{n!}P_0(c), & n\leq c-1,\\ \dfrac{\rho^n c^c}{c!}P_0(c), & n>c-1,\end{cases} \tag{4}$$

where $P_0(c)=\left[\sum_{n=0}^{c-1}\dfrac{c^n\rho^n}{n!}+\dfrac{c^c\rho^c}{c!(1-\rho)}\right]^{-1}$ is the probability that the multiprocessor is idle. Combining $P_e(X=n;c)$ and $P(X=n;c)$, the expression of exceeding probability can be derived as a function of the processor capacity c and the utilization ρ.

$$\begin{aligned}P_e(c)&=\sum_{n=0}^{\infty}P_e(X=n;c)P(X=n;c)\\ &=\sum_{n=0}^{c-1}(1-(1/2)^n)\frac{\rho^n c^n}{n!}P_0(c)+\sum_{n=c}^{\infty}(1-(1/2)^{c-1})\frac{\rho^n c^c}{c!}P_0(c)\\ &=1-\sum_{n=0}^{c-1}(1/2)^n\frac{\rho^n c^n}{n!}P_0(c)-\sum_{n=c}^{\infty}(1/2)^{c-1}\frac{\rho^n c^c}{c!}P_0(c)\\ &=1-P_0(c)\left[\sum_{n=0}^{c-1}(1/2)^n\frac{\rho^n c^n}{n!}+(1/2)^{c-1}\frac{\rho^n c^c}{c!(1-\rho)}\right].\end{aligned} \tag{5}$$

In Eq.(5), ρ can be fixed by varying μ with c, i.e., $\rho=\lambda/(c\mu)$, and then we can focus on the impact of c on the exceeding probability. Fig.5 uses the simulation data to verify the correctness of $P_e(c,\rho)$. The results

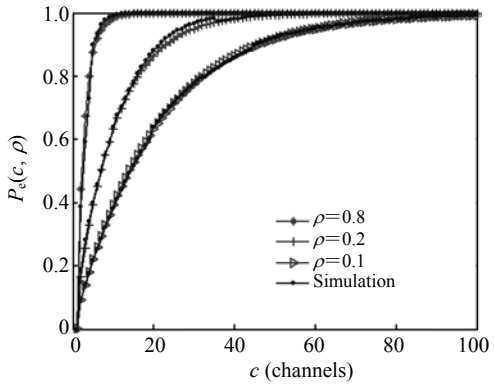

Fig.5 Verification of the exceeding probability formula

showed that Eq.(5) agrees very well with the simulation results. Since $P_0(c)$ is the reciprocal of a positive value, $P_0(c)>0$. It is easy to show that $P(X=n;c)>0$ and $P_e(X=n;c)>0$ when $c>1$ and $\rho>0$. So $P_e(c)>P_e(1)$ for all $c>1$ and $\rho>0$.

Upper bound model

We have investigated two important characters of the multiprocessor Fork-Join systems: EJ and CC. EJ occurs because tasks in different branches may exchange their siblings. This will reduce the waiting time of tasks at the join queue, so that the system response time can be reduced. The frequency that EJ occurs is determined by the exceeding probability.

Based on these findings, we propose an upper bound model B_c^K, whose mean response time is no less than that of the P_c^K system. This model is a single server Fork-Join system, whose parameters are specially tuned. B_c^K has the same task sojourn time as P_c^K, but has larger mean response time than P_c^K. Fig.6 shows the relationship between B_c^K and P_c^K.

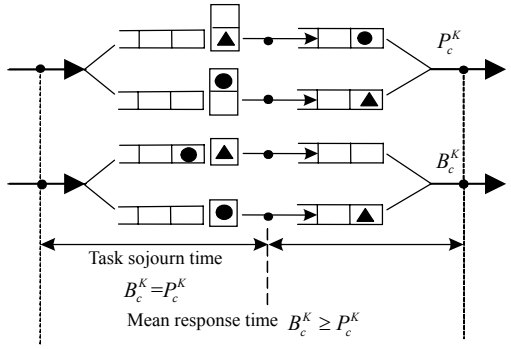

Fig.6 Sojourn time and response time comparison of P_c^K and B_c^K

The single channel of B_c^K limits the task finishing sequence. Exceeding can never happen and EJ is not permitted: $P_e(1)=0$. The parameters of the B_c^K system can be derived by queueing theory (Medhi, 1990). Program arriving rate is the same as P_c^K, so $\lambda_B=\lambda$. Mean service time of processors can be derived by Eq.(6), since B_c^K and P_c^K have the same mean task sojourn time:

$$\frac{1}{\mu_B-\lambda_B}=\frac{1}{\mu}+\frac{P_w}{c\mu(1-\rho)}, \tag{6}$$

where

$$P_w=\frac{(cp)^c}{c!(1-\rho)}p_0,\ p_0=\left[\frac{c^c\rho^c}{c!(1-\rho)}+\sum_{n=0}^{c-1}\left(\frac{c^n\rho^n}{n!}\right)\right]^{-1}.$$

So the mean service time of processors in B_c^K can be derived as:

$$\mu_B=\lambda+\left[\frac{1}{\mu}+\frac{P_w}{c\mu(1-\rho)}\right]^{-1}. \tag{7}$$

The relationship between B_c^K and P_c^K can be stated as:

Theorem 2 The mean response time of B_c^K is not less than that of P_c^K, i.e., $T_B \geq T$.

Intuitively, it is easy to verify that B_c^K and P_c^K systems have the same task sojourn times. The single server Fork-Join system will have longer mean response time since the tasks have to wait a longer time at the join queue. Proof of Theorem 2 can be found in Appendix B.

APPROXIMATION TECHNIQUE

We denote the bounding error of the mean response time as $E=T_B-T$. When we check the tightness of the upper bound, we find that E shows very typical dependent patterns on the parameters: c, K, and ρ. We fix two of the three parameters and increase another to see how the bounding error changes with the parameter. As shown in Figs.7 and 8, the bounding error E increases linearly with increasing ρ and c, and increases logarithmically with increasing K. It should be pointed out that since ρ needs to be fixed, when c changes, μ is adapted to keep ρ constant. The observed bounding error patterns help us to develop an analytical approximation formula for the mean response time of P_c^K system as follows. The error expression can be formulated as:

$$E=f(c\rho\lg K, c\rho, \rho\lg K, c\lg K, \lg K, c, \rho), \tag{8}$$

where f is a linear function of its variables, with coefficient vector $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$. We use the parameter scaling method to determine the values

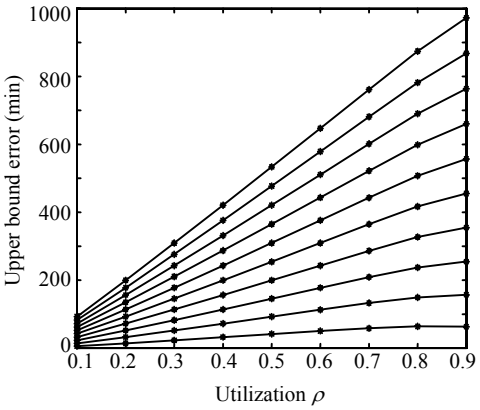

Fig.7 Upper bound error with increasing c and ρ when K is fixed. c=10, 20, 30, …, 100 respectively from bottom to top

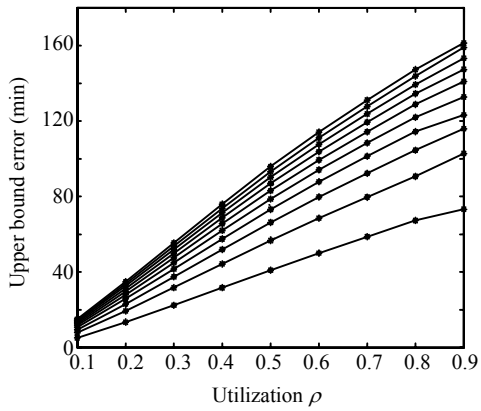

Fig.8 Upper bound error with increasing K and ρ when c is fixed. K=10, 20, 30, …, 100 respectively from bottom to top

of a_1~a_8, and obtain that $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$={3.15, 12.05, −0.16, 0.41, 24.10, −0.03, 4.00, 2.20}. As mentioned in Section 1, there are several available approximation results for single- server Fork-Join system. Combining the bounding error E with these formulae, we can give an analytical approximation formula for the mean response time of P_c^K system. We choose the approximation formula based on results of (Nelson and Tantawi, 1988):

$$T=\left[\frac{H_K}{H_2}+\frac{4}{11}\left(1-\frac{H_K}{H_2}\right)\rho_{\mathrm{B}}\right]\frac{(12-\rho_{\mathrm{B}})}{8}\cdot S-E, \quad (9)$$

where H_K is the harmonic series; $\rho_{\mathrm{B}}=\lambda/\mu_{\mathrm{B}}$; S is the mean sojourn time of tasks, when the mean service time is μ_{B} in the single server Fork-Join system; E is the upper bound error.

Remark 1 It is well known that the harmonic series increases logarithmically with increasing K, and since E also increases logarithmically with increasing K, thus the approximation Eq.(9) for the mean response time of multi-server Fork-Join system increases logarithmically with increasing K. This result tallies with the properties of the mean response time of the single-server Fork-Join system.

Remark 2 Eq.(9) is very easy to calculate. To evaluate a multiprocessor Fork-Join system with more than 20 branches and 10000 arriving programs by simulation, it will take hours or days to achieve static system performances for different parameters. With Eq.(9), it can be done within seconds.

NUMERICAL RESULTS

Multiprocessor Fork-Join system with EJ is a general model for a large range of applications. A good example is the engine repair system. Engines are disassembled into parts at first, and each part is processed in different repair shop. After that the finished parts are assembled and the same kind of parts can exchange with each other since every engine contains the same part. The object-oriented simulation model of the engine repair system is also this kind of system. Engines are regarded as programs and repairing process of parts is regarded as tasks. To verify the accuracy of the approximation formula, we build the engine repairing system simulation model to extensively compare the approximated mean response time with the simulation results.

Simulation is carried out with Flexsim™ (http://www.flexsim.com), which is a powerful simulation environment in performance analysis of queueing structure complex system. In simulations, we vary c, K and ρ over large ranges: $c\in$[2, 100], $K\in$[2, 128], $\rho\in$[0.1, 0.9]. Our aim is to check the performance of the approximation formula, to make sure it can effectively work for various multiprocessor environments. To approach the static status, we simulate 2000000 replications in calculating the mean response time of the system. We calculate the 99% confidence interval for the mean response time:

$Conf\{T-x\leq T\leq T+x\}$, $x=uv/\sqrt{m}$, in which u is the sampling variance, v=2.33 and m=2000000. Since we use very large replications to calculate the mean value, the confidence interval is nearly 0.1 percent of the mean value. The corresponding approximation result is calculated by Eq.(9).

Comparison of results

Due to limitation of space, only some results of comparison are summarized and presented. Fig.9a shows that when K is fixed at 5, with c varying from 2 to 100, ρ varying from 0.1 to 0.9, the approximation results accord very well with the simulation results. In Figs.9b and 9c, K is set to 25, 50 respectively. Eq.(9) still provides very accurate approximation.

After that, we fix c while varying K in the range K={2, 4, 8, 16, 32, 64, 128} and varying ρ in the range $\rho\in$[0.1, 0.9] to check how Eq.(9) works. Fig.10a summarizes the comparison results for c=5. It shows that the approximation is very accurate. In Figs.10b and 10c, we increase c to 50 and 100 respectively. The approximation formula shows good accuracy for large K and c. Even for K=128, c=100 and ρ=0.9, the approximation error is less than 5%.

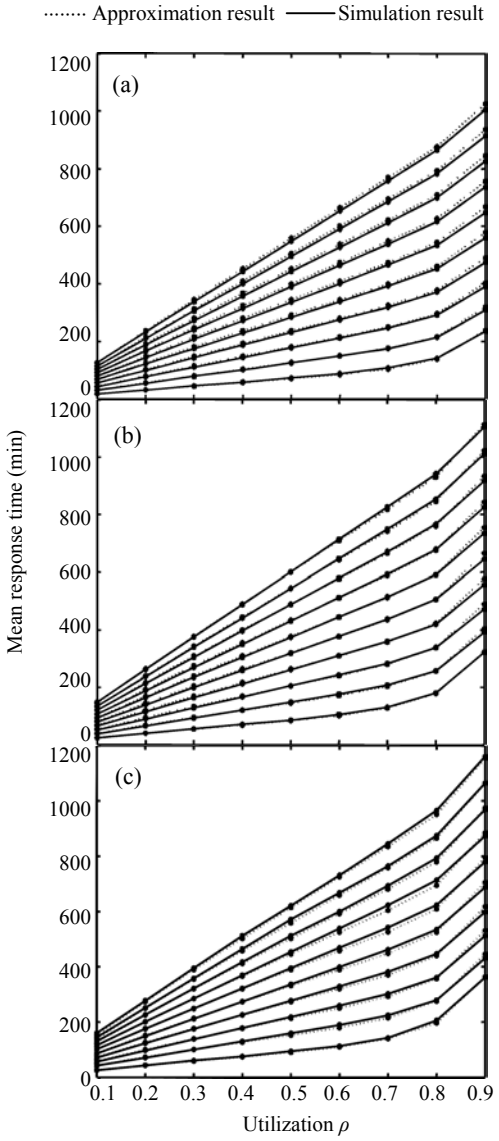

Fig.9 Approximation vs Simulation, c=10, 20, 30, …, 100 respectively from bottom to top. (a) K=5; (b) K=25; (c) K=50

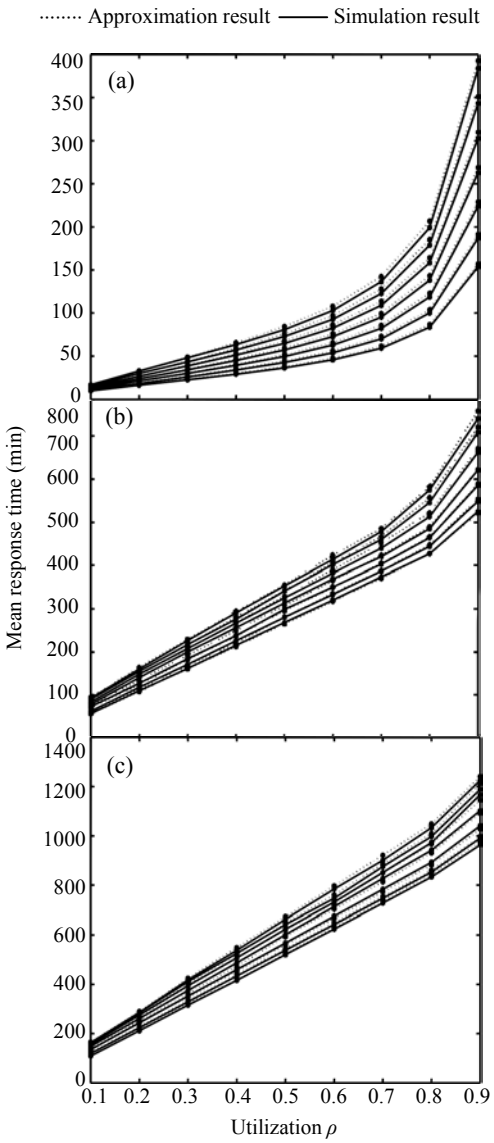

Fig.10 Approximation vs Simulation, K=2, 4, 8, ..., 128 respectively from bottom to top. (a) c=5; (b) c=50; (c) c=100

For all these parameter settings, the relative errors of the Approximation versus Simulation are recorded. Statistical results are summarized in Table 1, showing that under all parameter settings, the relative approximation error is always less than 5%. Numerical results validate that Eq.(9) can provide acceptable approximation for the mean response time of P_c^K system for various parameter settings.

Table 1 Relative approximation errors for various *c*, *K* and ρ

K	ρ	Relative approx. error		
		c=5	c=50	c=100
5	0.1	0.1802	0.3502	0.2843
	0.3	0.1498	0.3487	0.4486
	0.6	0.2211	0.3932	0.3932
	0.9	0.3068	0.3742	0.3941
25	0.1	0.1925	0.2458	0.3002
	0.3	0.1830	0.3081	0.3748
	0.6	0.2884	0.3618	0.4618
	0.9	0.2934	0.3509	0.4128
50	0.1	0.1937	0.2905	0.3265
	0.3	0.2796	0.2898	0.3744
	0.6	0.2355	0.3802	0.4811
	0.9	0.2069	0.3260	0.4257

CONCLUSION

We study the mean response time of programs processed by multiprocessor Fork-Join systems. Two characters are analyzed:

(1) Exchangeable join behavior of tasks allows tasks exchanging their partners at join operation, which can reduce the synchronization delay of tasks. This character is shown to effectively reduce the program's response time.

(2) Capacity constraint of the multiprocessor restricts the exchangeable join phenomenon. The number of early tasks that could be exceeded by a later task is limited by the number of channels.

Based on these characters, we construct a single processor Fork-Join system to work as an upper bound of the multiprocessor system. This enables us to take advantage of the existing results on the single processor Fork-Join systems which have been extensively studied in (Nelson and Tantawi, 1988). After analyzing the patterns of the bounding errors, we use parameter scaling method to develop an analytical approximation formula for T. It exhibits logarithmic behavior with increasing K, which also holds for single- server systems. Extensive comparisons with the simulation results have validated that the approximation formula can be confidently applied for evaluation of multiprocessing systems. Further study is to generalize the result to general service time systems, and to implement the results in fast evaluation of real applications.

References

Baccelli, F., Liu, Z., 1990. On the execution of parallel programs on multiprocessor systems—a queuing theory approach. *Journal of the ACM*, **37**(2):373-414. [doi:10.1145/77600.77622]

Barry, W., Allen, M., 1998. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. Addison-Wesley Publisher, Boston.

Flatto, L., 1985. Two parallel queues created by arrivals with two demands II. *SIAM Journal on Adv. Appl. Prob.*, **45**:845-861.

Flatto, L., Hahn, S., 1984. Two parallel queues created by arrivals with two demands. *SIAM Journal on Appl. Math.*, **44**(5):1041-1053. [doi:10.1137/0144074]

Glasserman, P., Yao, D.D., 1992. Some guidelines and guarantees for common random numbers. *Management Science*, **38**(6):884-908.

Ko, S.S., Serfozo, R.E., 2004. Response times in M/M/s Fork-Join networks. *Adv. Appl. Prob.*, **36**(3):854-871. [doi:10.1239/aap/1093962238]

Medhi, J., 1990. Stochastic Models in Queueing Theory. Academic Press Inc.

Nelson, R., Tantawi, A.N., 1988. Approximate analysis of Fork/Join synchronization in parallel queues. *IEEE Transactions on Computers*, **37**(6):739-743. [doi:10.1109/12.2213]

Taha, M.E., Stidham, S., 1999. Sample-Path Analysis of Queueing Systems. Kluwer Academic Publishers.

Towsley, D., Rommel, C.G., Stankovic, J.A., 1990. Analysis of Fork-Join program response times on multiprocessors. *IEEE Transctions on Parallel and Distributed Systems*, **1**(3):286-303. [doi:10.1109/71.80157]

Wright, P.E., 1992. Two parallel processors with coupled inputs. *Adv. Appl. Prob.*, **24**(4):986-1007. [doi:10.2307/1427722]

APPENDIX A

P_c^K denotes the Fork-Join system allowing exchangeable join. It has K branches and every branch contains c identical processors. U_c^K has all the same parameters as P_c^K, except that the EJ is not allowed. d_i and D_i are used to denote the product departure time of P_c^K and U_c^K systems respectively.

Using common random generator, the same task finishing time sequence can be generated for P_c^K and U_c^K. We denote them as $\{c_{i,j}\}$ (i=1,2,...,K; j=1,2,...,N). It means the finishing time of the jth task in the ith branch. In U_c^K, EJ is not allowed, so the departure time of the ith program is always equal to the latest task's service finishing time D_i=max($c_{i,1}$, $c_{i,2}$,...,$c_{i,N}$). In P_c^K, with EJ, if $c_{j,k}$ exchanges with $c_{i,k}$ in join process, there must be $c_{j,k}<c_{i,k}$ when $i<j$. So max($c_{i,1}$,...,$c_{j,k}$,...,$c_{i,N}$)≤max($c_{i,1}$,...,$c_{i,k}$,...,$c_{i,N}$). We have $d_i \le D_i$. The exchange only affects d_i and d_j. Other departure times d_{i+1},...,d_{j-1} will remain unchanged. If $c_{i,k}$≤max($c_{i,1}$,...,$c_{j,k-1}$,$c_{j,k+1}$,...,$c_{i,N}$), there must be $d_j=D_j$. In this condition, we have $\sum_{i=1}^{N} d_i \le \sum_{i=1}^{N} D_i$. If $c_{i,k}$ ≥max($c_{j,1}$,...,$c_{j,k-1}$,$c_{j,k+1}$,...,$c_{j,N}$), there must be $d_j=c_{i,k}$. Since $c_{j,m}\ge c_{j,m}$, m=1,2,...,K; $m\ne k$ in other branches, there must be $c_{i,k}\ge$max($c_{i,1}$,...,$c_{i,k-1}$,$c_{i,k+1}$,...,$c_{i,N}$). Then there must be $D_i=c_{i,k}$, so $d_j=D_i$. Since d_i= max($c_{i,1}$,...,$c_{j,k}$,...,$c_{i,N}$), and $c_{j,m}\ge c_{j,m}$, there must be $d_j=D_j$. So in this condition we also have $\sum_{i=1}^{N} d_i \le \sum_{i=1}^{N} D_i$.

APPENDIX B

We use P_c^K to denote the Fork-Join system with exchangeable join, and use B_c^K to denote its upper bound model. Because the sojourn times of B_c^K and P_c^K are the same, their task sojourn time sequences can be generated with reverse function method. We denote them $S_i^j = F^{-1}(\mu_i^j)$ (i=1,2,...,K; j=1,2,...,N), in which $F(x)$ is the distribution function of task sojourn time and μ_i^j is a random variable which is uniformly distributed in (0,1) while i and j stand for the ith task in the jth branch. We prove upper bound in this way. If the sample path of P_c^K is feasible for B_c^K, B_c^K and P_c^K will have the same mean response time. Otherwise, B_c^K must have larger mean response time than the P_c^K system.

When the sample path of P_c^K is feasible in B_c^K system, it means that in B_c^K there is no later task finishing earlier than prior task. The mean response time of B_c^K and P_c^K are calculated in the same manner:

$$T = T_{\mathrm{B}} = \frac{1}{N}\sum_{j=1}^{N}[F^{-1}(\mu_1^j),...,F^{-1}(\mu_K^j)]. \tag{B1}$$

If some tasks' sojourn times are feasible in P_c^K system, but infeasible in B_c^K system, we prove $T_{\mathrm{B}}\ge T$ and the proof is divided into two cases.

Case 1 Only one branch contains infeasible sojourn time, say branch j.

In this case, there must be a later task exceeding a prior task: $d_i^j \le d_{i-1}^j$ in P_c^K. Although this is feasible in P_c^K, it is infeasible in B_c^K. Considering the service process of task i in the single processor, it has to wait until the departure of the (i−1)th task. We denote the sojourn time of task i in B_c^K as x_i^j. Comparing with $d_i^j \le d_{i-1}^j$ in P_c^K, if the infeasible condition occurs, there must be $x_i^j \ge d_{i-1}^j$, i.e., $x_i^j > F^{-1}(\mu_i^j)$. The waiting time of the ith task may affect the later tasks' sojourn times, which could only become longer for the postponing of the prior tasks. So the feasible sojourn time sequence of this task in B_c^K should be $x_k^j > F^{-1}(\mu_k^j)$ (k=i,i+1,…,K). The mean response time of B_c^K system can be calculated as:

$$\begin{aligned} T_{\mathrm{B}} &= \frac{1}{N}\sum_{j=1}^{N}\max\left(F^{-1}(\mu_i^1),...,x_i^j,...,F^{-1}(\mu_i^K)\right) \\ &\ge \frac{1}{N}\sum_{j=1}^{N}\max\left(F^{-1}(\mu_i^1),...,F^{-1}(\mu_i^K)\right) = T. \end{aligned} \tag{B2}$$

So $T_{\mathrm{B}}\ge T$.

Case 2 There are multiple branches containing infeasible sojourn times. Without loss of generality, we assume that the infeasibility happens in the first m branches. For the capacity limitation of the single processor, if $x_i^j \geq d_i^j$ occurs, all the later tasks have to be postponed. The sample path of B_c^K system can be obtained as $\{x_i^j, x_{i+1}^j, ...\}$ $(j=1,2,...,m)$, in which $x_k^j \geq F^{-1}(\mu_k^j)$ $(k=i,i+1,...,K)$. The mean response time of B_c^K can be calculated as:

$$\begin{aligned} T_{\mathrm{B}} &= \frac{1}{N}\sum_{j=1}^{N}\max\left(x_i^1,...,x_i^m,...,F^{-1}(\mu_i^K)\right) \\ &\geq \frac{1}{N}\sum_{j=1}^{N}\max\left(F^{-1}(\mu_i^1),...,F^{-1}(\mu_i^K)\right) \\ &= T. \end{aligned} \tag{B3}$$

In summary, for all the sample paths for P_c^K and B_c^K systems, we have proved that $T_{\mathrm{B}} \geq T$.