Analysis of efficient synchronization in bulk-synchronous parallel discrete-event simulation [*]

Mauricio Marín
Computing Department, University of Magellan
Casilla 113-D, Punta Arenas, Chile
E-mail: mmarin@ona.fi.umag.cl

Abstract

Event dependencies associated with correct parallel discrete event simulations are preserved by using synchronization protocols. These protocols ensure that at the end of the simulation each event has taken place in chronological order in its respective logical process. To this end, the protocols apply a method to advance in the simulation time which can be synchronous or asynchronous. This paper presents an analysis of the relevant factors affecting the performance of known protocols for the bulk-synchronous parallel model of computing.

1 Introduction

Parallel discrete event simulation (PDES) [2] is performed by decomposing the simulation model into a set of logical processes (LPs), and distributing them onto the available processors. The constraint which ensures a final result equivalent to a sequential simulation of the same system, is that each LP must cause the event occurrences in *nondecreasing timestamp order*. Adherence to this constraint is not trivial since some sort of mechanism, called *synchronisation protocol*, must be introduced in order to synchronise the simulation of parallel events. Synchronization in this case refers to synchronization of events in the simulated time and it is achieved by imposing a specific method of advance in simulation time.

In this paper we present asymptotic expressions for efficient realisations of synchronization protocols for PDES. The operation of these protocols is based on two main methods of simulation time advance which we call *asynchronous* (ASYNC) and *synchronous* (SYNC) time advance respectively. Asynchronous time advance refers to simulations in which the synchronization of events is made locally, at LP level. The Chandy-Misra-Bryant (CMB) [10] and Time Warp (TW) [3] protocols are based on this approach. Synchronous time advance, on the other hand, refers to time stepped advance which is made by means of globally defined time intervals of fixed or variable length. We use the so-called *event-horizon* concept [14] as the realization of synchronous time advance. The YAWNS [11] and Breathing Time Buckets (BTB) [13] protocols are based on event-horizon time advance.

[*]Partially supported by Fondecyt project 1010723.

The synchronization protocols can be further characterized into conservative and optimistic ones. In conservative protocols (CMB, YAWNS) [10, 11] events are allowed to take place when there is certainty that no earlier events can take place in the respective LPs. This poses a difficult problem since determining when it is "safe" to process a given event requires information of the behavior of other LPs that might schedule events in the same place. This is in contrast with the approach adopted by optimistic protocols (TW,BTB) [3, 13] which simply processes available events and performs corrections whenever causality errors take place. However, state saving and roll-back procedures must be introduced in order to correct erroneous computations. These operations can have a significant cost both in memory and running time.

On the other hand, asynchronous conservative protocols block themselves when there is insufficient information to determine whether the available events are safe or not. This leads to the problem of deadlock occurrences which can introduce significant overheads associated with their detection and recovery. Also the need for keeping information about what events the LPs may schedule in the others, leads one to deal with the particular communication topology of the LPs. This makes these protocols less useful in simulations where the communication topology changes dynamically or the fan-in/fan-out of the communication is large (e.g., all to all).

Furthermore, synchronous conservative protocols base their operation upon global information in a manner that does not require special consideration of the communication topology. However, this "global information" by itself exacerbates their requirements of global synchronization in real and simulation time which can degrade the performance significantly. Similar argument applies to synchronous optimistic protocols. Thus it is not clear what kind of protocol is best suited for a given system.

This paper is concerned with the efficient realization of PDES on the bulk-synchronous parallel (BSP) model of computing [15, 9]. In the BSP model both computation and communication take place in bulk before the next point of global synchronization of processors. A BSP program is composed of a sequence of *supersteps*. During each superstep, the processors may only perform computations on data held in their local memories and/or send messages to other processors. These messages are available for processing at their destinations by the next superstep, and each superstep is ended with the barrier synchronization of the processors.

The running time cost of a superstep is calculated as follows. Let w be the maximum amount of local computation executed by any processor during a superstep s. Let h_s be the maximum number of messages sent by any processor during s, and h_r be the maximum number of messages received by any processor during s. Then the cost of the superstep s is given by $l + w + g \max\{h_s, h_r\}$ where g is the cost of passing one-word message through the communication network under a situation of continuous traffic, and l is the cost of barrier synchronizing the processors [9]. The cost of a BSP program is the sum of the costs of its supersteps.

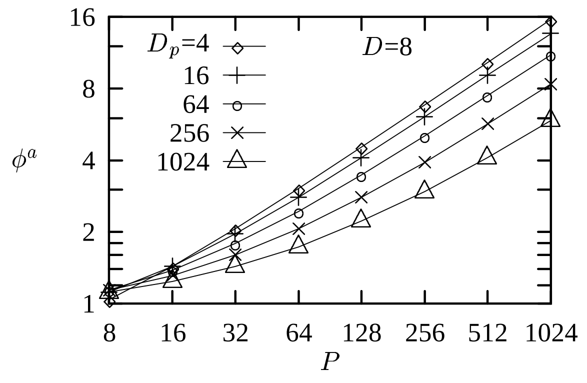
We present a comparison of BSP synchronization protocols which is based on the results presented in [5, 6, 8]. In particular, we determined expressions for the number of supersteps per unit simulation and load balance for both approaches to time advance. These expressions belong to idealized BSP protocols. From them we derive running time expressions for actual conservative and optimistic synchronization protocols. Section 2 reviews results presented in [5] which show the comparative performance of conservative and optimistic protocols. Section 3 presents asymptotic expressions which show the scalability of the different protocols. Section 4 presents conclusions and validate them with data from actual implementations of the synchronization protocols.

2 Comparison of protocols

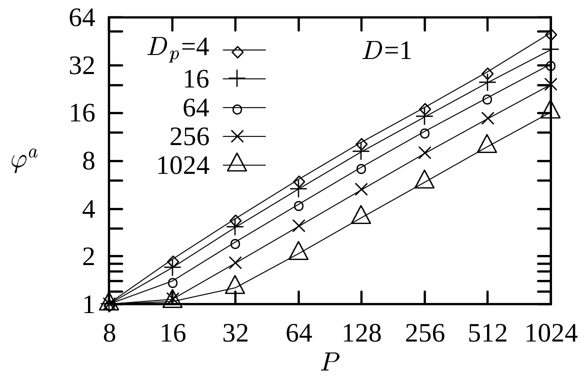
In [7, 8] we derived regression analysis formulae which allows one to establish comparisons between known synchronization protocols. In particular, in [5] we compared the protocols YAWNS [11, 1], BTB [13] and TW (BSP-TW as proposed in [8, 7]). Simulation models are represented by the PHold work-load [2].

In PHold there are P processors hosting D_p LPs each. Initially, D events are scheduled in each LP, so that the constant event population is $N = D\,D_p\,P$. The ocurrence of each event causes the scheduling of a single new event. The time increment between consecutive events follows a negative exponential probability distribution. After an event e_i takes place at LP p_i the following event e_j is scheduled to take place at an LP p_j. The LP p_j is selected uniformly at random from all the N_p LPs of the system with $N_p = D_p\,P$. Thus with probability $1 - 1/P$ the event e_j takes place in a different processor. This represents a fully connected communication topology among LPs.

In the optimistic protocols we increased the cost of processing each event in $\varphi \geq 1$ units in order to include the effect of state saving. Roll-backs cause re-simulation of events thus we consider that this operation increases the



(a)



(b)

Figure 1. Points in which SYNC protocols outperforms ASYNC protocols.

total number of simulated events in $\phi\,N$ events with $\phi \geq 1$. Figure 1 shows the points at which SYNC protocols outperform ASYNC protocols.

Figure 1.a shows ASYNC ϕ values (ϕ^a) for which BTB outperforms TW. Each point is the ϕ^a value that solves the costs equation BTB $-$ TW $= 0$ for the case $\phi^s = 1$, $\varphi^a = \varphi^s = 2$, $r = 1$ and $D = 8$ ([5]). The figure shows that the SYNC protocol outperforms the ASYNC protocol for rather unrealistic ϕ^a values.

Figure 1.b shows asynchronous φ^a values for the solution of YAWNS $-$ TW $= 0$ with $\phi^a = 2$, $r = 1$ and $D = 1$ (we use $D = 1$ instead of $D = 8$ to emulate a queuing network work-load since PHold cannot be simulated with YAWNS). Similar to BTB, for large P the φ^a values are rather unrealistic.

The above experiments assumed unfavorable scenarios for the ASYNC approach. This makes it evident that SYNC protocols can only outperform ASYNC protocols under conditions unlikely to be found in practice. In section 4 we provide empirical results on actual implementations of the protocols which validate this claim.

3 Asymptotic trend

In the following we present expressions that describe the asymptotic cost of the protocols as the model size and number of processors scale up simultaneously. In order to include the CMB protocol, we consider a less demanding work-load composed of a queuing network with toroidal topology. We simplify the expressions by assuming that load balance is optimal, namely we assume that the number of LPs (events) per processor is large enough. Constant factors are omitted, and we scale up the model by setting a constant number of LPs per processor and increasing the number of processors.

In the queuing network each server is under high load so that it is never idling. The service times are exponential with mean one so that simulating the model up to the time T_E requires the processing of about $2\,N_p\,T_E$ events (arrival+completion) with N_p being the total number of LPs. Let us define $D_p = N_p/P$ where P is the number of processors.

The queuing model is defined so that there are D_p LPs per processor, and both P and D_p are perfect squares. For each service entry one message is sent to other LP reporting the arrival of the job to be serviced (job pre-sending). In each processor there are $\approx 4\sqrt{D_p}$ LPs that send messages to LPs located in other processors with probability 1/4. Thus the total number of events per unit simulation time is $N_e = 2\,D_p\,P$, and the total number of messages per unit simulation time is $N_m = \sqrt{D_p}\,P$. We assume that the cost of processing each event is a constant $C_e = 1$.

YAWNS

The total number of supersteps per unit simulation time required by YAWNS is $\approx \frac{4}{5}\sqrt{N_p}$ since the queuing model tends to behave like PHold with one event per LP [6]. We can reduce these supersteps by assuming that the time window is calculated considering the messages that are sent to other processors only (i.e., we exclude the messages among co-resident LPs). A total of $\sqrt{D_p}\,P$ messages are sent to other processors per unit simulation time, so that $S_p^s \approx \frac{4}{5}\,D_p^{1/4}\,P^{1/2}$. Thus the total BSP cost per unit simulation time of YAWNS is given by YWS $= (C_e\,N_e)/(E_f^s\,P) + (N_m/(E_f^s\,P))\,g + S_p^s\,l + S_p^s \cdot$Min-reduction. Assuming $E_f^s = 1$ and min-reduction with cost $[1+g+l]\lg P$ (the alternative method has cost $P + P\,g$), the asymptotic BSP cost of YAWNS is

$$\begin{aligned}\text{YWS} = \quad & [D_p + D_p^{1/4}\,P^{1/2}\,\lg P] + \\ & [D_p^{1/2} + D_p^{1/4}\,P^{1/2}\,\lg P]\,g + \\ & D_p^{1/4}\,P^{1/2}\,(1+\lg P)\,l\,.\end{aligned}$$

CMB

We assume CMB with null-messages [10, 8] and YAWNS' lookahead, and (given the large lookahead) we also assume that CMB approximates the supersteps of the asynchronous time advance approach (SYNC) within a constant factor. The supersteps per unit simulation time for ASYNC is $S_p^a \approx \ln\ln(D_p^{1/2}\,P)$ [6] (at most this value since this S_p^a is for PHold with one event per LP and fully connected topology, and a total of $D_p^{1/2}$ causality threads migrate to other processors in each unit of simulation time).

The total asymptotic cost per unit simulation time of computation plus communication of normal events is $D_p + D_p^{1/2}\,g$. However, in each superstep CMB must process null events and null messages. D_p null events are processed in each processor per superstep, and at most $D_p^{1/2}$ null messages are sent to other processors per superstep. These operations are perfectly balanced. Thus the asymptotic BSP cost of null events and null messages is $D_p\,S_p^a + D_p^{1/2}\,S_p^a\,g$, and thereby the asymptotic BSP cost of CMB is given by

$$\begin{aligned}\text{CMB} = \quad & D_p\,(1 + \ln\ln(D_p^{1/2}\,P)) + \\ & D_p^{1/2}\,(1 + \ln\ln(D_p^{1/2}\,P))\,g + \\ & \ln\ln(D_p^{1/2}\,P)\;l\,.\end{aligned}$$

CTW

Similar arguments to that used in CMB can be applied to conservative time windows with shortest paths (CWT) [8], which attemps to reduce the null-messages traffic of CMB by increasing lower bounds for the time of the next event in each LP. CTW must execute the shortest paths algorithms in each superstep. The cost of this algorithm is $D_p \lg D_p$. Thus the asymptotic BSP cost of CTW is

$$\begin{aligned}\text{CTW} = \quad & D_p\,(1 + \lg D_p\,\ln\ln(D_p^{1/2}\,P)) + \\ & D_p^{1/2}\,(1 + \ln\ln(D_p^{1/2}\,P))\,g + \\ & \ln\ln(D_p^{1/2}\,P)\;l\,.\end{aligned}$$

TW

Even with small lookahead BSP TW can achieve similar supersteps to the asynchronous approach to time advance (ASYNC) [6]. However, roll-backs increase the number of event re-simulations. Let us assume that each event is re-simulated, on average, $\phi - 1$ times ($\phi \geq 1$). Like the other protocols, we assume that load balance is optimal (i.e., $D_p \gg 1$). We also assume that state saving increases the cost of each event in $\varphi - 1$ units ($\varphi \geq 1$). Thus the asymptotic BSP cost of TW is given by

$$\text{TW} = \varphi\,\phi\,D_p \;+\; (2\,\phi - 1)\,D_p^{1/2}\,g \;+\; \ln\ln(D_p^{1/2}\,P)\;l\,.$$

The parameters φ and ϕ provide information about the trade-off of using optimistic simulation (see section 2).

BTB

The BTB protocol can be seen as the optimistic counter part of the YAWNS protocol. Thus the incremental cost due to state saving and rollbacks applies. However, BTB does not generate anti-mesagges.

$$\begin{aligned} \text{BTB} = \quad & \varphi\,\phi\,[D_p + D_p^{1/4}\,P^{1/2}\,\lg P] + \\ & [D_p^{1/2} + D_p^{1/4}\,P^{1/2}\,\lg P]\,g + \\ & D_p^{1/4}\,P^{1/2}\,(1 + \lg P)\,l\,. \end{aligned}$$

Validation

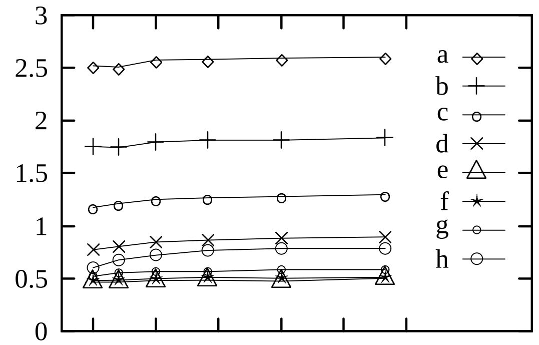
Figure 2 shows an experimental validation of the number of supersteps (S_p) for the SYNC and ASYNC protocols (i.e., S_p^s and S_p^a respectively). In the figure, the curves a, b, c, and d show S_p^s values obtained with the SYNC strategy divided by a function defined below, whereas the curves e, f, g, and h show S_p^a values obtained with the ASYNC strategy. In all experiments a total of eight events are scheduled initially in each LP (high load).

Figure 2.a shows data for experiments in which D_p is kept fixed and P is varied. The x-axis is $\sqrt{P}$ and the experimental values of S_p^s have been divided by $\sqrt{P}$. The four curves for each strategy are data for D_p= 128, 32, 8 and 2 (top to bottom). The number of processors is P= 4, 8, 16, 32, 64 and 128. The figure shows $S_p^s = O(\sqrt{P})$ whereas S_p^a seems to be constant for large D_p.

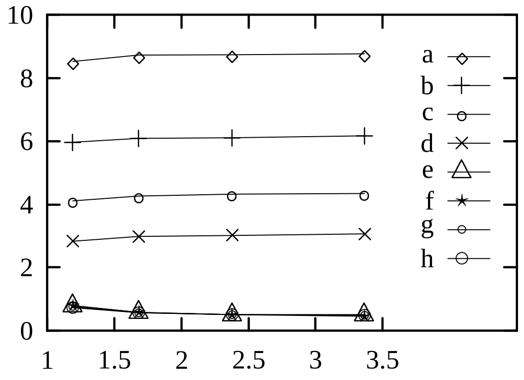
Figure 2.b presents the inverse case, namely P is kept fixed and D_p is varied. The x-axis is $D_p^{1/4}$ and the experimental values of S_p^s have been divided by $D_p^{1/4}$. In this case, the curves are data for P= 128, 64, 32, and 16 (top to bottom). D_p= 2, 8, 32 and 128. The figure shows $S_p^s = O(D_p^{1/4})$ whereas S_p^a seems to decrease with D_p.

Summary and more cases

The above formulae are presented in the first two sections of table 1. YAWNS-2 uses the non-scalable $P + P\,g$ min-reduction. The third section of the table show the cost of the algorithms in accordance with the trend observed empirically (figure 2), namely constant number of supersteps per unit simulation time. Overall, the formulae of the table show that YAWNS requires more slackness to amortize its higher communication and synchronization costs. These results are for the toroidal network which has a sparse topology because it defines a constant number of input links to each LP. For a fully connected topology, the only feasible protocols are YAWNS and TW. This case is shown in the fourth section of the table.



(a)



(b)

Figure 2. SYNC and ASYNC supersteps for toroidal topology.

4 Final comments

The synchronous protocols require larger number of supersteps to complete the simulation. As a result they are strongly impacted by a moderately high cost of barrier synchronization of processors. These protocols are more suitable for communication topologies with large fan-in/fan-out. The need for performing periodical min-reductions can also be detrimental to the performance of the synchronous protocols in systems with sparse communication topology. The formulae presented in table 1 confirm these conclusions.

Figure 3 shows experimental results with actual implementations of the protocols. This is a very favorable case for conservative synchronous protocols since the number of processors and logical processes is small and the available lookahead is large (lookahead is a measure of the degree in which a logical process can predict the timestamp of events to arrive from downstream logical processes [2], this improves significantly the performance of conservative protocols). The results show that the general purpose optimistic protocol is able to achieve competitive performance.

References

[1] P.M. Dickens, D.M. Nicol, P.F. Reynolds, and J.M. Duva. “Analysis of Bounded Time Warp and comparison with YAWNS”. *ACM Trans. on Modeling and Computer Simulation*, 6(4):297–320, Oct. 1996.

[2] R.M. Fujimoto. “Parallel discrete event simulation”. *Comm. ACM*, 33(10):30–53, Oct. 1990.

[3] D.R. Jefferson. “Virtual Time”. *ACM Trans. Prog. Lang. and Syst.*, 7(3):404–425, July 1985.

[4] M. Y. H. Low. “Dynamic load balancing for BSP Time Warp”. In *35th Annual Simulation Symposium*, San Diego, California, April 2002.

[5] M. Marín. “Comparative analysis of a parallel discrete-event simulator”. In *XX International Conference of the Chilean Computer Science Society*, Santiago, Chile, Nov 2000 (IEEE-CS Press).

[6] M. Marín. “Asynchronous (time-warp) versus synchronous (event-horizon) simulation time advance in BSP”. In *Euro-Par’98 (Workshop on Theory and Algorithms for Parallel Computation*, pages 897–905, Sept. 1998. LNCS 1470.

[7] M. Marín. “Time Warp On BSP Computers”. In *12th European Simulation Multiconference*, June 1998.

[8] M. Marín. “Discrete-event simulation on the bulk-synchronous parallel model”. PhD Thesis, Programming Research Group, Computing Laboratory, Oxford Unversity, 1998 (anonymous ftp at ona.fi.umag.cl).

[9] W.F. McColl. “General purpose parallel computing”. In A.M. Gibbons and P. Spirakis, editors, *Lectures on Parallel Computation*, pages 337–391. Cambridge University Press, 1993.

[10] J. Misra. “Distributed discrete-event simulation”. *Computing Surveys*, 18(1):39–65, March 1986.

[11] D.M. Nicol. “The cost of conservative synchronization in parallel discrete event simulations”. *Journal of the ACM*, 40(2):304–333, April 1993.

[12] D.B. Skillicorn, J.M.D. Hill, and W.F. McColl. “Questions and answers about BSP”. Technical Report PRG-TR-15-96, Computing Laboratory, Oxford University, 1996. Also in *Journal of Scientific Programming*, V.6 N.3, 1997.

[13] J.S. Steinman. “SPEEDES: A multiple-synchronization environment for parallel discrete event simulation”. *International Journal in Computer Simulation*, 2(3):251–286, 1992.

[14] J.S. Steinman. “Discrete-event simulation and the event-horizon”. In *8th Workshop on Parallel and Distributed Simulation (PADS’94)*, pages 39–49, 1994.

[15] L.G. Valiant. “A bridging model for parallel computation”. *Comm. ACM*, 33:103–111, Aug. 1990.

[16] L.G. Valiant. “General purpose parallel architectures”. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North Holland, 1990.

Protocol	Comp.	Comm.	Sync.
YAWNS	$D_p + D_p^{1/4}\, P^{1/2}\, \lg P$	$D_p^{1/2} + D_p^{1/4}\, P^{1/2}\, \lg P$	$D_p^{1/4}\, P^{1/2}\,(1 + \lg P)$
YAWNS-2	$D_p + D_p^{1/4}\, P^{3/2}$	$D_p^{1/2} + D_p^{1/4}\, P^{3/2}$	$D_p^{1/4}\, P^{1/2}$
CMB	$D_p\,(1 + \ln \ln (D_p^{1/2}\, P))$	$D_p^{1/2}\,(1 + \ln \ln (D_p^{1/2}\, P))$	$\ln \ln (D_p^{1/2}\, P)$
TW	$\varphi\, \phi\, D_p$	$(2\,\phi - 1)\, D_p^{1/2}$	$\ln \ln (D_p^{1/2}\, P)$
CMB	D_p	$D_p^{1/2}$	1
TW	$\varphi\, \phi\, D_p$	$(2\,\phi - 1)\, D_p^{1/2}$	1
YAWNS	$D_p + D_p^{1/2}\, P^{1/2}\, \lg P$	$D_p + D_p^{1/2}\, P^{1/2}\, \lg P$	$D_p^{1/2}\, P^{1/2}\,(1 + \lg P)$
YAWNS-2	$D_p + D_p^{1/2}\, P^{3/2}$	$D_p + D_p^{1/2}\, P^{3/2}$	$D_p^{1/2}\, P^{1/2}$
TW	$\varphi\, \phi\, D_p$	$(2\,\phi - 1)\, D_p$	$\ln \ln (D_p\, P)$

Table 1.

The cost of computation, communication, and synchronization. The first section of the table presents formulae for YAWNS with $O(\lg P)$ and $O(P)$ min-reductions respectively for the toroidal topology. The second section presents results for CMB and TW under the assumption of PHold's S_p^a values adapted to the queuing network model and toroidal topology. The third section presents similar results but considering the $S_p^a = O(1)$ trend observed in figure 2. The fourth section presents results for a queuing network with fully connected topology and PHold's S_p^a values.

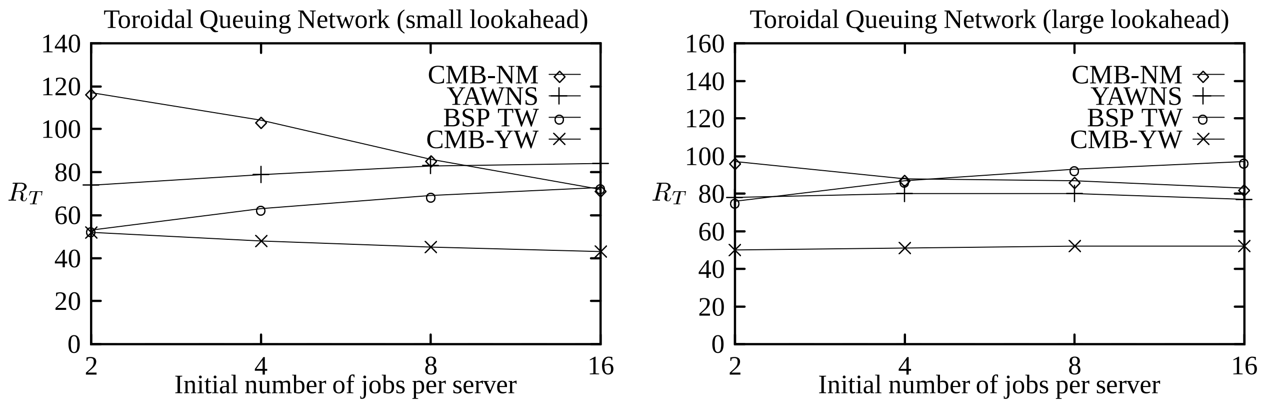


Figure 3.

Results for a BSPlib implementation of the protocols on a 4-processors SGI PowerChallenge computer. R_T= running time in seconds. Figure (a) shows values for a case in which the time of the next events are calculated by the sum $0.1 + X$ with X exponentially distributed and figure (b) for the case $1.0 + X$ (the constants 0.1 and 1.0 — called base lookahead — are exploited by the conservative protocol to achieve better performance). In the figures, CMB-NM is CMB with null-messages, whereas CMB-YW is CMB-NM extended with YAWNS' lookahead.