Speedup Analysis of Data-parallel Applications on Multi-core NoCs

Xiaowen Chen[1,2,*], Zhonghai Lu[2], Axel Jantsch[2] and Shuming Chen[1]

Abstract — *As more computing cores are integrated onto a single chip, the effect of network communication latency is becoming more and more significant on Multi-core Network-on-Chips (NoCs). For data-parallel applications, we study the model of parallel speedup by including network communication latency in Amdahl's law. The speedup analysis considers the effect of network topology, network size, traffic model and computation/communication ratio. We also study the speedup efficiency. In our Multi-core NoC platform, a real data-parallel application, i.e. matrix multiplication, is used to validate the analysis. Our theoretical analysis and the application results show that the speedup improvement is nonlinear and the speedup efficiency decreases as the system size is scaled up. Such analysis can be used to guide architects and programmers to improve parallel processing efficiency by reducing network latency with optimized network design and increasing computation proportion in the program.*

Index Terms — speedup, communication, multi-core, NoC

I. INTRODUCTION

We are entering the multi-, many and even thousand core era [1] ! The rapid development of integrated circuit and computer architecture technology enables to integrate a number of cores on a single chip. These computing cores are potential to cooperate to obtain powerful performance. In this paper, we study the model of parallel speedup and its efficiency on multi-core Network-on-Chip (NoC) [3][4].

As is known, Amdahl's law [2] provides a simple, yet very useful method to analyze the potential speedup of a parallel system. We use Amdahl's law as the basic speedup model and add the analysis of network communication latency, since the network communication latency is becoming increasingly significant for the multi-core system. We consider the network communication latency model based on a regular NoC infrastructure. As a scalable solution of interconnecting so many cores, NoC has attracted significant attention over the last ten years since various buses do not scale well with the system size.

The focus of our analysis is the influence of network communication latency on the parallel speedup. We discuss in detail how the network topology, network size and traffic models (uniform or hotspot) affect the speedup. Our model also analyzes the effect of the computation/communication ratio. As matrix multiplication is a representative data-parallel application, we implement it on our Multi-core NoC platform to validate our speedup analysis. The theoretical analysis and the application results tell us that the speedup rises up nonlinearly, its efficiency goes down and the network impact is becoming more important when the network becomes larger and more complicated. Our study exhibits that optimizing network design and increasing computation proportion is a viable way to improve the performance.

The rest of the paper is organized as follows. Section II discusses related work. In Section III, we present the speedup model involving network communication latency and detail the analysis. In Section IV, the matrix multiplication is implemented on our Multi-core NoC platform and simulation results are discussed. Finally, we conclude in Section V.

II. RELATED WORK

Many researchers discuss variants of Amdahl's law for different purposes. In [5], Li and Malek discussed the effect of communication/computation ratio on the speedup, but their communication delay model is simple without considering the detail of interconnects. In [6], Paul revisited Amdahl's law on the single chip heterogeneous multiprocessor. In [7], Cho and Melhem presented the corollaries to Amdahl's law to study the interaction between parallelization and energy consumption. In [8], Hill and Marty offered a corollary of a simple model of multi-core hardware resources based on Amdahl's law but omitted the effect of network latency.

We note that few works discuss the effect of network communication latency on the speedup of Multi-core NoCs, which is the topic of this paper.

III. MODELS AND ANALYSIS

A. Problem Definition

The problem we consider is the parallel speedup in the context of multi-core packet-switched NoCs for data parallel applications. We give detailed analysis on communication latency. To facilitate constructing the models of communication latency and parallel speedup, k-ary-2-mesh/torus homogeneous are used and a program running on the multi-core NoCs is abstracted as a set of subtasks and communications. The communication denotes the interaction between two communicating nodes. A communication contains one or more packets transmitted in the network. The subtask denotes the non-communication processing (e.g. computation, memory access, etc.) between two successive communications. The models and analysis in this paper is based on the following three assumptions. 1) The requirement of computation time and communication time of the program assigned to each node is equal to the others. That is, the program in each node contains the same number of subtasks and communications. 2) The computation time of each subtask is also equal to the others. 3) The time of each communication is also equal to that of others.

This work has been supported partially by the FP7 EU project MOSART under contract number IST-215244 and the National 863 Program of China under Grant No. 2007AA01Z108.

[1]the authors are with the Microelectronics Institute, School of Computer Science, National University of Defense Technology, 410073, Changsha, P.R.China (email: xwchen@nudt.edu.cn; smchen@nudt.edu.cn).

[2]the authors are with the Department of Electronic, Computer, and Software Systems, KTH-Royal Institute of Technology, SE100-44, Stockholm, Sweden (email: xiaowenc@kth.se; zhonghai@kth.se; axel@kth.se).

[*]To whom correspondence should be addressed. Email: xiaowenc@kth.se

B. Notations

To facilitate the analysis, we first define a set of symbols.

i	the processor node No.
N	the number of processor nodes, $N = k^2$
k	topological ary, $k = \sqrt{N}$
s	the number of subtasks in the serial part of a program
p	the number of subtasks or communications in the parallel part of a program
τ_c	the computation time of a subtask
H	average hop count of transmitting a packet
τ_{1hop}	the time of transmitting a packet in one hop
τ_t	average time of transmitting a packet in the network
M	the number of packets in a communication
$T_t(i)$	the time of a communication issued by node i
T_T	the communication overhead of a program on the multi-core NoC
ρ	the ratio between computation time and communication time, $\rho = \tau_c/T_t(i)$
S	parallel speedup
E	speedup efficiency

C. Communication Latency Model

Communication latency contains two parts: minimal (non-contention) latency and contention latency.

The minimal latency is determined by the distance of the two communicating subtasks (nodes). We use hop count to calculate the latency. Table I lists the calculated hop count. We consider two representative traffic models (Uniform and Hotspot) and two popular NoC topologies (2D Mesh and 2D Torus). For uniform traffic, the two formulas are cited from [9]. For hotspot traffic, the corner node 0 is chosen as the hotspot node.

TABLE I
CALCULATION OF HOP COUNT

	k-ary-2-mesh	k-ary-2-torus
Uniform	$H = \begin{cases} \frac{2k}{3} & k \text{ even} \\ 2\left(\frac{k}{3} - \frac{1}{3k}\right) & k \text{ odd} \end{cases}$	$H = \begin{cases} \frac{2k}{4} & k \text{ even} \\ 2\left(\frac{k}{4} - \frac{1}{4k}\right) & k \text{ odd} \end{cases}$
Hotspot	$H = \frac{k^2}{k+1}$	$H = \begin{cases} \frac{k}{2} + \frac{k}{2(k^2-1)k} & k \text{ even} \\ \frac{k}{2} & k \text{ odd} \end{cases}$

The contention latency mainly depends on the routing and arbitration mechanism and the communication pattern of real applications. In general, it is difficult to quantify it exactly. The contention latency is included in τ_{1hop}. The architectural latency of τ_{1hop} is 1 cycle (in our experimental platform shown in Fig. 1), but τ_{1hop} is usually larger than 1 cycle when contention occurs.

With packet switching, the average time of transmitting a packet in the network is

$$\tau_t = H \cdot \tau_{1hop} \tag{1}$$

where H reflects the distance and τ_{1hop} the latency under contention.

In general, a communication contains one or more packets. There is at least one packet in a communication, so $T_t(i) \geq \tau_t$. For the worst case that packets are transmitted sequentially, i.e. a packet will not be transmitted until the previous one is finished, $T_t(i) = M \cdot \tau_t$. Since transmissions may exist simultaneously in the network, the time will be less. Hence, we have,

$$H \cdot \tau_{1hop} = \tau_t \leq T_t(i) \leq M \cdot \tau_t = M \cdot H \cdot \tau_{1hop} \tag{2}$$

The communication overhead of a program running on the multi-core NoCs is determined by $T_t(i)$. The program is parallelized on N nodes. If communications issued by each node have no dependence between each other, $T_T = min\{p/N \cdot T_t(i)\}$. In contrast, the worst case is that communications of each node happens sequentially one by one. So, we can get the inequality for T_T below,

$$min\left\{\frac{p}{N} \cdot T_t(i)\right\} \leq T_T \leq p \cdot T_t(i) \tag{3}$$

Considering (2) and (3), we can get,

$$max\{T_T\} = p \cdot M \cdot \tau_t = p \cdot M \cdot H \cdot \tau_{1hop} \tag{4}$$

which is the maximal communication overhead of the program for the worst case where all packets are transmitted in the network in a sequential way.

Minimal latency, $T_t(i)$ and T_T are affected by the network size, network topology and traffic models: a) When the network size is scaled up, H, $T_t(i)$ and T_T increases due to the longer communication distance; b) Because torus has shorter average distance than mesh for the same network size, the distance latency of torus is less than that of mesh and the difference is more obvious when the network size is larger; c) Hotspot traffic has higher average hop count, and hence more minimal latency than uniform traffic, especially when the network size becomes larger. Besides, hotspot traffic causes much contention.

D. Speedup Model

$$S = \frac{(s+p) \cdot \tau_c}{s \cdot \tau_c + \frac{p}{N} \cdot \tau_c + T_T} \tag{5}$$

Equation (5) gives the speedup formula. The last product item in the denominator describes the communication overhead. In the following, we use two cases to analyze the speedup in detail.

Case 1: The network is k-ary-2-mesh and the traffic model is uniform. We adopt the maximum of $T_t(i)$ and the minimum of T_T in (5). $T_t(i)$ of each node is the same due to the three assumptions. Then we have

$$\begin{aligned} S &= \frac{(s+p) \cdot \tau_c}{s \cdot \tau_c + \frac{p}{N} \cdot \tau_c + min\left\{\frac{p}{N} \cdot T_t(i)\right\}} \\ &= \frac{(s+p) \cdot \tau_c}{s \cdot \tau_c + \frac{p}{N} \cdot \tau_c + \frac{p}{N} \cdot T_t(i)} \\ &= \frac{(s+p) \cdot \tau_c}{s \cdot \tau_c + \frac{p}{N} \cdot \tau_c + \frac{p}{N} \cdot M \cdot H \cdot \tau_{1hop}} \end{aligned} \tag{6}$$

$$= \frac{(s+p) \cdot \tau_c}{s \cdot \tau_c + \frac{p}{N} \cdot \tau_c + \frac{p}{\sqrt{N}} \cdot M \cdot \frac{2}{3} \cdot \tau_{1hop}} \tag{7}$$

When N increases, S increases. However, when M increases, S decreases yet, because the communication latency rises up. More contention results in the increase of τ_{1hop} and hence the decrease of S.

We define *Speedup efficiency* as *speedup* S divided by the

number of nodes, N. We have

$$E = \frac{S}{N} = \frac{(s+p)\cdot\tau_c}{s\cdot\tau_c\cdot N + p\cdot\tau_c + p\cdot M\cdot\frac{2}{3}\sqrt{N}\cdot\tau_{1hop}} \quad (8)$$

As the network size N increases, the speedup efficiency is slowing down. The communication latency deepens the decrease of the speedup efficiency.

We also refine Equation (6) to reflect the impact of the computation/communication ratio as follows:

$$S = \frac{(s+p)\cdot\rho}{s\cdot\rho + \frac{p}{N}\cdot\rho + \frac{p}{N}} = \frac{s+p}{s + \frac{p}{N} + \frac{p}{N}\cdot(\frac{1}{\rho})} \quad (9)$$

The speedup increases as the computation takes more proportion in the program when the system size is fixed. When $\rho \to \infty$, the effect of communication becomes insignificant.

Case 2: The network is k-ary-2-torus (k is odd) and the traffic model is hotspot. We adopt the maximum of $T_t(i)$ and the maximum of T_T in (5). Then we have

$$S = \frac{(s+p)\cdot\tau_c}{s\cdot\tau_c + \frac{p}{N}\cdot\tau_c + p\cdot T_t(i)} \quad (10)$$

$$= \frac{(s+p)\cdot\tau_c}{s\cdot\tau_c + \frac{p}{N}\cdot\tau_c + p\cdot M\cdot H\cdot\tau_{1hop}}$$

$$= \frac{(s+p)\cdot\tau_c}{s\cdot\tau_c + \frac{p}{N}\cdot\tau_c + p\cdot M\cdot\frac{\sqrt{N}}{2}\cdot\tau_{1hop}} \quad (11)$$

Let $\frac{\partial S}{\partial N} = 0$, we can get $N = \sqrt[3]{\left(\frac{4\cdot\tau_c}{M\cdot\tau_{1hop}}\right)^2}$. Because $N \geq 2$ for the multi-core NoCs, we can obtain two results. i) When $\frac{4\cdot\tau_c}{M\cdot\tau_{1hop}} \leq 2\sqrt{2}$, S decreases with the increase of N. Parallelization degrades the performance rather than improve it. ii) When $\frac{4\cdot\tau_c}{M\cdot\tau_{1hop}} > 2\sqrt{2}$, S increases with the increase of N when N is smaller. However, S achieves its maximum when $N = \sqrt[3]{\left(\frac{4\cdot\tau_c}{M\cdot\tau_{1hop}}\right)^2}$. As N continues becoming larger, S yet decreases. In (11), we also can get that more contention results in the increase of τ_{1hop} and hence the decrease of S. Improving the computation time and reducing the communication time can make the maximum of S larger.

The speedup efficiency in this case is

$$E = \frac{(s+p)\cdot\tau_c}{s\cdot\tau_c\cdot N + p\cdot\tau_c + p\cdot M\cdot\frac{N\cdot\sqrt{N}}{2}\cdot\tau_{1hop}} \quad (12)$$

If the communication delay is ignored, $E \sim O(N^{-1})$. However, (12) suggests that $E \sim O(N^{-3/2})$. The communication latency deepens the decrease of the speedup efficiency.

Refine Equation (10) to Equation (13) below:

$$S = \frac{(s+p)\cdot\rho}{s\cdot\rho + \frac{p}{N}\cdot\rho + p} = \frac{s+p}{s + \frac{p}{N} + p\cdot(\frac{1}{\rho})} \quad (13)$$

As with *Case 1*, augmenting the computation part in the program can improve the speedup.

IV. Experiments and Results

A. Experimental Platform

To validate our analysis, we constructed a Multi-core NoC experimental platform as shown in Fig. 1. The Multi-core NoC has a 2-mesh/torus topology and its size is configurable. Each Processor-Memory (PM) node has a LEON3 processor, an enhanced memory controller plus a local memory. The enhanced memory controller extends the function of LEON3's own memory control model to support memory accesses from remote nodes via the network. The network performs dimension-order XY routing, and provides best-effort service. Moving one hop in the network takes one cycle.

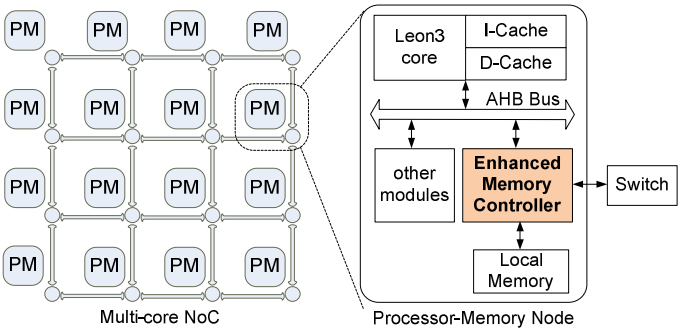

Fig. 1. The Multi-core NoC experimental platform

B. Application Example

We use matrix multiplication as the application example and perform experiments on various instances of the application described as follows:

1) Two types of data partition are realized to reflect the two traffic models. The one is "*Uniform*" meaning that the matrix data are equally separated into all nodes. The other is "*Hotspot*" meaning that the matrix data are only located in the corner node 0.
2) Both integer matrix and floating point matrix are implemented to vary the computation/communication ratio, ρ. For the same problem size and algorithm, floating point computation needs more time than integer computation and hence has bigger ρ.
3) The product of two matrix, A[64,1] and B[1,64], results in a C[64,64] matrix, on the Multi-core NoC with the network size varying from 1x1 (1), 2x2 (4), 4x4 (16), to 8x8 (64) and with both the mesh and torus topology. The total problem size is fixed and the problem size assigned to each node varies from A[***64***,1]*B[1,64], A[***16***,1]*B[1,64], A[***4***,1]*B[1,64] and A[***1***,1]*B[1,64].

C. Simulation Results

Effect of network size

Fig. 2 ~ Fig. 4 shows that the speedup increases and the speedup efficiency decreases, as the network size is scaled up.

Effect of traffic models

Fig. 2 plots the speedup and the speedup efficiency of the integer matrix multiplication for uniform and hotspot traffic versus the size of the 2-mesh Multi-core NoC from 1x1 (1), 2x2 (4), 4x4 (16) to 8x8 (64). The speedup for hotspot traffic on 4x4 (16) is only slightly bigger than that on 2x2 (4) and that on 8x8 (64) is even smaller. It's because larger network size, e.g. 8x8

(64), leads to heavy traffic load for the hotspot. For both uniform and hotspot traffic, the speedup efficiency slows down as the increase of the network size. Moreover, the speedup efficiency for hotspot traffic goes down very quickly, resulting in very small E with only 0.05 for hotspot on 8x8 (64).

We also estimate the theoretical speedup and its efficiency plotted in Fig. 2. The program of matrix multiplication can fully be parallelized, thus s = 0. The computation subtask on each node is c(i,j) = c(i,j) + a(i,k)*b(k,j). The computation time of such subtask is collected in our experiment. For "*Uniform*" data partition, a(i,k) and c(i,j) are located on the local node but b(k,j) is on the remote node, so M = 1. For "*Hotspot*" data partition, a(i,k), b(k,j) and c(i,j) are all located on the node(0,0), hence M = 4. It's difficult to statically estimate the contention latency, so we set τ_{1hop} to be 1 meaning that there is no contention latency in our theoretical speedup. Then, using the speedup formula (5) estimates the theoretical speedup. Fig. 2 shows that the real speedup and its efficiency are consistent as to the trend of the theoretical speedup and its efficiency. The real one is less than the theoretical one and the difference between them increases with the network size, because more contention is incurred.

Effect of network topology

Fig. 3 plots the speedup and the speedup efficiency of the integer matrix multiplication for k-ary-2-mesh topology and k-ary-2-torus topology versus the size of the Multi-core NoC with the uniform traffic from 1x1 (1), 2x2 (4), 4x4 (16) to 8x8 (64). As shown in the figure, the speedup increases but the speedup efficiency decreases, when the network size increases. The difference between the speedup for k-ary-2-mesh and that for k-ary-2-torus is becoming larger along with the increase of the network size, so is the difference for the speedup efficiency. For instance, S for k-ary-2-mesh and k-ary-2-torus is the same, 3.94, due to the equivalence of 2x2 mesh and 2x2 torus, while S for 8x8 mesh is 36.57 and S for 8x8 torus is 41.08. This is because the 8x8 torus has shorter communication distance than the 8x8 mesh.

Effect of computation/communication ratio

Fig. 4 plots the speedup and the speedup efficiency of both integer and floating point matrix multiplication versus the size of the mesh Multi-core NoC with the uniform traffic from 1x1 (1), 2x2 (4), 4x4 (16) to 8x8 (64). For the same network factors, the speedup and its efficiency for the floating point multiplication is higher than those for the integer multiplication. This is as expected because when the computation time increases, the portion of communication latency becomes less significant and thus achieving higher speedup.

V. CONCLUSION

In the paper, we built up the abstract model for speedup analysis of data-parallel applications on Multi-core NoCs by including detailed network communication factors into the speedup formulas. Our theoretical analysis and the real application results show the effect of network topology, network size, traffic models, and computation/communication ratio. Such results signify that optimizing network parameters and increasing computation proportion in the program can achieve higher performance.

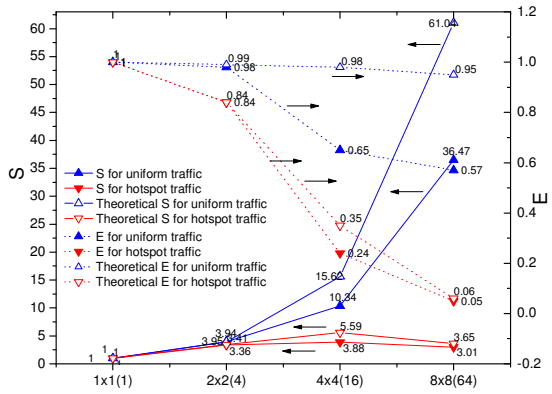

Fig. 2. Effect of traffic models

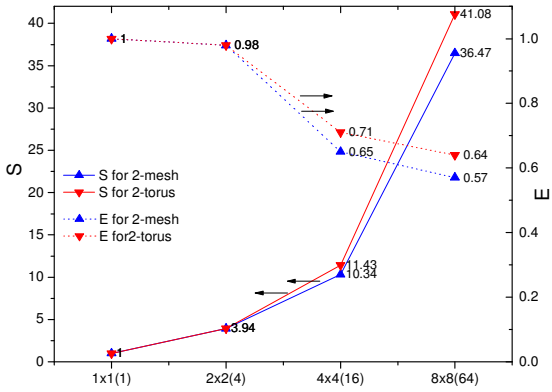

Fig. 3. Effect of network topology

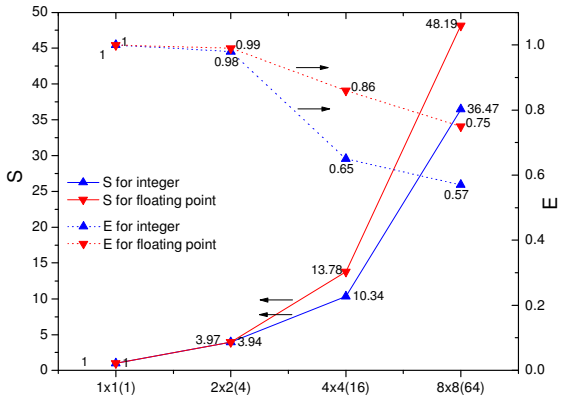

Fig. 4. Effect of computation/communication ratio

REFERENCES

[1] S. Borkar, "Thousand core chips -- a technology perspective," *Proc. ACM/IEEE 44th Design Automation Conf.(DAC)*, 2007, pp. 746-749.

[2] G. M. Amdahl, "Validity of the single-processor approach to achieving large-scale computing capabilities," *Proc. Am. Federation of Information Processing Societies Conf.*, AFIS Press, 1967, pp. 483-385.

[3] A. Jantsch and H. Tenhunen, "Networks on chip", *Kluwer Academic Publishers*, 2003.

[4] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip", *ACM Computing Surveys*, vol. 38, no. 1, pp. 1-51, Mar. 2006.

[5] X. Li and M. Malek, "Analysis of speedup and communication/ computation ratio in multiprocessor systems", *Proc. Real-Time Systems Symposium*, 1988, pp. 282-288.

[6] J. M. Paul, "Amdahl's law revisited for single chip systems," *Int'l J. Prarallel Programming,* vol. 35, no. 2, pp. 101-123, Apr. 2007.

[7] S. Cho and R. Melhem, "Corollaries to Amdahl's law for energy," *Computer Architecture Letters,* vol. 7, no. 1, pp. 25-28, Jan.-Jun. 2008.

[8] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *IEEE Computer,* vol. 41, no.7, pp. 33-38, Jul. 2008.

[9] W. J. Dally and B. Towles, "Principles and practices of interconnection networks", *Morgan Kaufmann Publishers*, 2004