Computing and Informatics, Vol. 35, 2016, 941–962

ENERGY WALL FOR EXASCALE SUPERCOMPUTING

Zhiyuan Wang, Yuhua Tang, Juan Chen

State Key Laboratory of High Performance Computing
National University of Defense Technology
Changsha, Hunan, 410073, China
e-mail: yhtang62@163.com, juanchen@nudt.edu.cn

Jingling Xue

School of Computer Science and Engineering
University of New South Wales, Australia

Yun Zhou, Yong Dong

School of Computer
National University of Defense Technology
Changsha, Hunan, 410073, China

Abstract. "Sustainable development" is one of the major issues in the 21st century. Thus the notions of green computing, green development and so on show up one after another. As the large-scale parallel computing systems develop rapidly, energy consumption of such systems is becoming very huge, especially system performance reaches Petascale (10^{15} Flops) or even Exascale (10^{18} Flops). The huge energy consumption increases the system temperature, which seriously undermines the stability and reliability, and limits the growth of system size. The effects of energy consumption on scalability become a growing concern. Against the background, this paper proposes the concept of "Energy Wall" to highlight the significance of achieving scalable performance in peta/exascale supercomputing by taking energy consumption into account. We quantify the effect of energy consumption on scalability by building the energy-efficiency speedup model, which integrates computing performance and system energy. We define the energy wall quantitatively,

and provide the theorem on the existence of the energy wall, and categorize the large-scale parallel computers according to the energy consumption. In the context of several representative types of HPC applications, we analyze and extrapolate the existence of the energy wall considering three kinds of topologies, 3D-Torus, binary n-cube and Fat tree which provides insights on how to mitigate the energy wall effect in system design and through hardware/software optimization in peta/exascale supercomputing.

Keywords: Energy consumption, scalability, exascale computing, energy-efficiency speedup, energy wall

Mathematics Subject Classification 2010: 68M01

1 INTRODUCTION

Currently, the scalable parallel computing has become the common approach for achieving high performance, and the system size scales up rapidly for meeting the growing demands. Despite great strides made by supercomputing systems, much of scientific computation's potential remains untapped, "because many scientific challenges are far too enormous and complex for the computational resources at hand" [1]. Planned exascale supercomputers (capable of an exaflop, 10^3 petaflops, or 10^{18} floating point operations per second) in this decade promise to overcome these challenges by a revolution in computing at a greatly accelerated pace [2]. However, exascale supercomputing is faced up with a number of challenges, including technology, architecture, energy, reliability, programmability and usability. In [3], we have discussed the reliability challenge for exascale supercomputing. This treatise is aimed at addressing the energy challenge in building scalable supercomputing systems, particularly those at the peta/exascale levels.

According to the top ten supercomputers in Top500 list [4] published in November 2015, the highest system power is 17 808 KW of Tianhe-2, and its yearly power bill will be more than 25 million US dollars. It shows that current high performance systems consume huge energy, and the energy consumption may continue to climb up as the performance increases, which is unsustainable and brings about serious challenges of stability, cooling, and so on.

"Sustainable development" is one of the major issues in the 21^{st} century. Thus the notions of green computing, green development and so on show up one after another. As the large-scale parallel computing systems develop rapidly, the effect of energy consumption on system scalability becomes a growing concern.

Against the background, we quantify for the first time the concept of "Energy Wall". We highlight the significance of achieving scalable performance in peta/exascale supercomputing by taking energy consumption into account.

For the parallel system, we present the energy-efficiency speedup model, define quantitatively the energy wall, give the existence theorem of the energy wall, and categorize it according to the energy consumption as red scalable system, yellow scalable system or green scalable system. Finally, we take 3D-Torus, binary n-cube and Fat tree topologies as examples for demonstrating our "Energy Wall" theory.

2 THE ENERGY WALL

Table 1 provides a list of symbols, their meanings, and where they are defined in the paper.

Symbol	Meaning	Definition
P	The number of cores	Section 2.1
E_P^R	The overall energy consumption of the system	(1)
E_P^E	Effective energy consumption of the overall system	Section 2.1
E_E	Effective computing energy consumption	(1)
E_E^C	Communication energy consumption of network	(1)
E_W^S	Redundant computing energy for serial part of application	(1)
E_W^I	Idling energy consumption for computation	(1)
E_W^C	Idling energy consumption of network	(1)
E_P^W	Extra energy consumption of introducing parallelization	(1)
S_P	Traditional speedup	(2)
E_P	Energy consumption speedup	(3)
$E(P)$	Factor of parallel energy consumption	(4)
V_P^E	Energy efficiency	(5)
S_P^E	Energy-efficiency speedup	(6)
S_{Amdahl}^E	Amdahl energy-efficiency speedup	(7)
$S_{Gustafson}^E$	Gustafson energy-efficiency speedup	(8)
U_P^E	Efficiency of parallel energy consumption	Section 2.2

Table 1. List of main symbols, their meanings, and definitions

Section 2.1 describes the composition of energy consumption of the system. Section 2.2 introduces a new energy-efficiency speedup model. Section 2.3 quantifies the energy wall based on our energy-efficiency speedup model.

2.1 Energy Composition

Energy consumption of nodes and interconnection network are two major components of high performance computing system [5], and each of them is further divided into static energy consumption and dynamic energy consumption. Only certain operating components may generate the dynamic consumption. For example, the node dynamic energy consumption occurs when the node is computing, and the network dynamic energy consumption occurs when the data is transmitted, while the static energy consumption always exists even when nodes or network resources are idle.

Node energy consumption mainly consists of the energy consumption of Effective computing E_E, the energy consumption of redundant computing for serial part of parallel application E_W^S (the serial part of parallel application needs only one process to complete, and other processes should wait until the finish of the serial part or compute serial part simultaneously, which consumes much "Wasted" energy consumption), and the energy consumption of idling E_W^I. Network energy consumption mainly comes from network resources (e.g. routers) consisting of energy consumption of Communication E_E^C and that of idling E_W^C. Shared-memory-based communications among different cores within a single computation node (CN) do not incur energy consumption of network resources, and therefore we treat energy consumed by such inter-core communications as a portion of energy consumption of the effective computation E_E on the corresponding node. In conclusion, the dynamic energy consumption of redundant computing for serial part E_W^S plus the static energy consumption, i.e. cores and network resource idling energy consumption $E_W^I + E_W^C$, is the waste when the system is running.

Suppose a parallel system consists of P cores and the number of cores per node is a constant. A parallel program runs on the system with one process per core, resulting in a total of P processes.

Figure 1 a) shows the energy consumption of an ideal parallel computing system which runs without any waste of energy, i.e. all energy consumption are used for effective operation (computing and communication), as the number of cores increases from 1 to P. E_P^E is the energy consumption of this ideal system, which includes energy consumption of computation E_E and communication energy consumption of network E_E^C, i.e. $E_P^E = E_E + E_E^C$. E_E is the energy consumption of sequential version of the parallel application executed by a single core, which does not change with the system size.

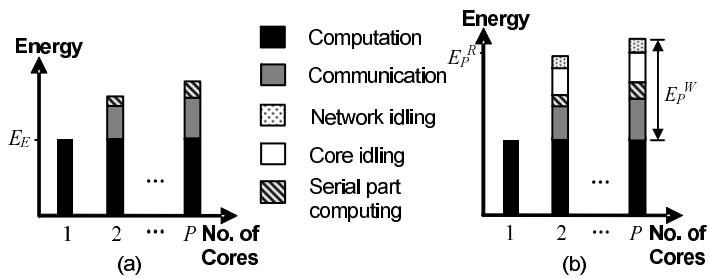

Figure 1. The increase of energy consumption of parallel system as its size scales up. a) Energy consumption of an ideal parallel system. b) Energy consumption of a parallel system in practice.

Figure 1 b) shows the energy consumption of a parallel system in practice, where E_P^R represents the total energy consumption during the period of Running the pro-

grams, including both dynamic and static energy consumption. Then we obtain the energy composition of the system as follows:

$$E_P^R = E_E + E_E^C + E_W^S + E_W^I + E_W^C = E_E + E_P^W. \tag{1}$$

As shown in Figure 1 b) and Equation (1), we define E_P^W. According to Equation (1), E_P^W has four components, representing the extra energy consumption of introducing parallelization by increasing the system size. It means that the increase of performance rarely comes without energy consumption. In other words, when the system performance increases, especially approaches exascale, the effect of energy consumption on scalability should be taken into account. Thus, energy is integrated into traditional speedup model to measure the scalability change of parallel computing.

2.2 Energy-Efficiency Speedup

The increase of computing performance is at the cost of energy consumption. As usual, the workload of parallel computing is measured by FLoating-point OPerations (FLOP) or Instruction Counts (IC). Without loss of generality, we measure the system workload by FLOP, and the computing performance is measured by FLoating-point OPerations per Second (FLOPS).

Given an application G, traditional speedup is the ratio of the computing performance of the parallel execution to that of the serial execution. Since the FLOP of both parallel and serial executions are identical, i.e. $FLOP_1 = FLOP_P$, the traditional speedup may be expressed as

$$S_P = \frac{FLOPS_P}{FLOPS_1} = \frac{FLOP_P/T_P}{FLOP_1/T_1} = \frac{T_1}{T_P}, \tag{2}$$

where T_j $(j = 1, P)$ is the time for completing workload $FLOP_j$ on j core(s) of the system, and $FLOPS_j$ is the computing performance on j core(s) of the system. The efficiency achieved is $U_P = \frac{S_P}{P}$.

Similarly, energy consumption speedup is defined as:

Definition 1 (Energy Consumption Speedup E_P)**.** Given an application G running on a P-core system, the energy consumption speedup is the ratio of energy consumption of parallel execution E_P^R to that of sequential version of the parallel application executed by a single core E_1^R.

$$E_P = \frac{E_P^R}{E_1^R}. \tag{3}$$

Energy consumption speedup reflects the increase of energy consumption of the system as the system size is scaled up. According to Section 2.1, the energy

consumption of effective computing $E_E = E_1^R$. According to Equation (3), energy consumption speedup can be rewritten as

$$E_P = \frac{E_P^R}{E_E} = \frac{E_E + E_P^W}{E_E} = 1 + \frac{E_P^W}{E_E} = 1 + E(P), \tag{4}$$

where $E(P)$ is called the factor of parallel energy consumption. Since E_P^W is the extra energy consumption after introducing parallelization to improve performance, $E(P)$ reveals the relationship between extra energy consumption for parallelization and the energy consumption of effective computing.

Next, we give the definition of energy efficiency and then build up the energy-efficiency speedup model.

Definition 2 (Energy Efficiency V_P^E)**.** Given an application G running on a P-core system, the energy efficiency is the computing performance gained from an unit of energy.

$$V_P^E = \frac{FLOPS_P}{E_P^R} = \frac{FLOP_P}{T_P E_P^R}. \tag{5}$$

According to the definition of energy efficiency, the energy-efficiency speedup model is built up as follows.

Definition 3 (Energy-Efficiency Speedup S_P^E)**.** Given an application G running on a P-core system, the energy-efficiency speedup is the ratio of energy efficiency of parallel execution V_P^E to that of sequential version of the parallel application executed by a single core V_1^E.

$$S_P^E = \frac{V_P^E}{V_1^E}.$$

According to Equations (3) and (5), the energy-efficiency speedup can be written as

$$S_P^E = \frac{V_P^E}{V_1^E} = \frac{FLOP_P/(T_P E_P^R)}{FLOP_1/(T_1 E_E)} = \frac{T_1/T_P}{E_P^R/E_E} = \frac{S_P}{E_P} = \frac{S_P}{1 + E(P)} = PU_P U_P^E, \tag{6}$$

where $U_P^E = \frac{1}{1+E(P)}$ is called *efficiency of parallel energy consumption*, which reflects the increase efficiency of computing scalability gained from the unit of energy. Thus energy-efficiency speedup S_P^E is the ratio of traditional speedup to energy consumption speedup, which is quantified by the speed-up of performance by consuming an unit of energy.

Different from traditional speedup, energy-efficiency speedup S_P^E integrates both energy consumption and computing performance of parallel system from the perspective of scalability. The variation of S_P^E is determined by both the application characteristic and the system architecture. Typically, if the proportion of the computation overhead is larger than that of the communication overhead during the parallel program running on the system, the application is computation-intensive, otherwise it is communication-intensive. For example, if there are lots of

communication operations, e.g. point-to-point communication for communication-intensive applications, the traditional speedup S_P increases slowly when the system size is scaled up, even worse, then decreases and approaches zero if there are massive collective communications, e.g. one-to-all or all-to-all broadcasting. Generally, the scalability of computation-intensive applications is better than that of the communication-intensive applications, hence many researches focused on the optimization of communication such as overlapping computations and communications.

Traditional speedup S_P varies mainly in three cases based on the types of applications, i.e. $\lim_{P\to\infty} S_P = \infty$, $\lim_{P\to\infty} S_P = C$, where C is a positive constant and $\lim_{P\to\infty} S_P = 0$, which we define as scalable, weak-scalable and unscalable applications, respectively.

Amdahl law [6] and Gustafson law [7] are two representative traditional speedups. When the system size is scaled up, Amdahl speedup is limited by the serial part of the program f, i.e. $\lim_{P\to\infty} S_{Amdhal} = \frac{1}{f}$ which quantifies the scalability for the weak-scalable applications, and Gustafson speedup increases linearly, i.e. $\lim_{P\to\infty} S_{Gustafson} = \infty$ which quantifies the scalability for scalable applications.

The Amdahl and Gustafson laws are readily to be generalized to Amdahl energy-efficiency speedup and Gustafson energy-efficiency speedup without loss of generality.

Amdahl energy-efficiency speedup may be defined as

$$S^E_{Amdahl} = \frac{S_{Amdhal}}{1 + E(P)} = \frac{PU^E_P}{1 + f(P-1)}. \tag{7}$$

Gustafson energy-efficiency speedup may be defined as

$$S^E_{Gustafson} = \frac{S_{Gustafson}}{1 + E(P)} = ((1-f)P + f)U^E_P. \tag{8}$$

Particularly, if $E^W_P = 0$, i.e., there is no extra energy consumption after introducing parallelization, then $E(P) = 0$ and $U^E_P = 1$, and the expressions of (7) and (8) are identical to those of the traditional Amdahl's and Gustafson's speedup models. However, the communication energy consumption of parallel computing cannot be ignored in practical scenario, therefore our works only consider the case where $E^W_P > 0$.

Taken Gustafson energy-efficiency speedup as an example, Figure 2 shows the trends of Gustafson speedup and Gustafson energy-efficiency speedup when the system size is scaled up. As shown in Figure 2, the energy-efficiency speedup S^E_P never exceeds the traditional speedup S_P because the efficiency of parallel energy consumption $U^E_P < 1$.

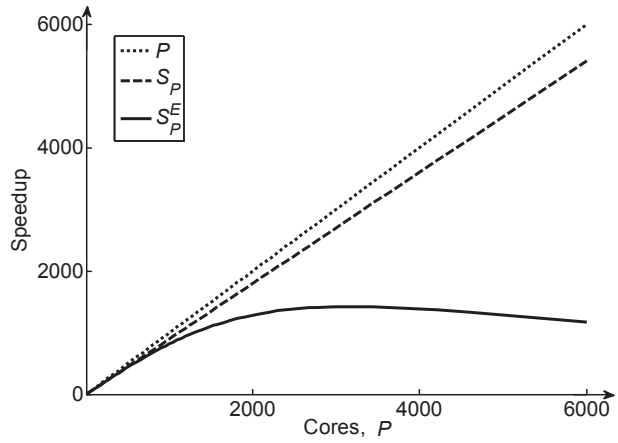

Figure 2. Gustafson speedup vs. Gustafson energy-efficiency speedup

2.3 Energy Wall

In Section 2.2, we discuss the relationship between energy consumption and scalability, where we try to answer how and to what extent the energy consumption limits the scalability. This section first gives the quantitative definition of energy wall, and then proposes the existence theory of energy wall along with its proof.

Definition 4 (Energy Wall)**.** Given an application G running on a P-core system, energy wall is the maximum of energy-efficiency speedup S_P^E, which is denoted by $\max S_P^E$.

By convention, we use $f(x) \succcurlyeq g(x)$ to denote the case that $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ is a positive constant or ∞, and $f(x) \succ g(x)$ if $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ is ∞. In addition, the operators $\preccurlyeq$ and $\prec$ are used in the standard manner. If $f(x) = \Theta(g(x))$, as is customary, the Θ notation[1] describes asymptotically both an upper bound and a lower bound.

Theorem 1 (Existence Theory of Energy Wall)**.** Given an application G running on a P-core system, energy wall exists if and only if $\lim_{P\to\infty} S_P^E = 0$.

Proof. By Definition 3, we explore the trends of energy-efficiency speedup S_P^E when S_P changes in different laws.

Case 1: $\lim_{P\to\infty} S_P = \infty$.

According to the definition of the increase factor of parallel energy consumption, $E(P)$ is the monotonically increasing function in P. By Definition 3, energy-efficiency speedup S_P^E has three curve shapes.

[1] Suppose $\mathfrak{R}(x)$ is the set consisting of all functions of x, $f(x) \in \mathfrak{R}(x)$ and $g(x) \in \mathfrak{R}(x)$. $f(x) = \Theta(g(x))$ if both $\lim_{x\to\infty} \frac{g(x)}{f(x)}$ and $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ are positive constants.

1. If $E(P) \succ S_P$, when the system size is scaled up, energy-efficiency speedup S_P^E first increases, then decreases and approaches zero.
2. If $E(P) = \Theta(S_P)$, when the system size is scaled up, energy-efficiency speedup S_P^E increases monotonically and approaches a positive constant.
3. If $\Theta(1) \prec E(P) \prec S_P$, when the system size is scaled up, energy-efficiency speedup S_P^E increases monotonically and approaches ∞.

$\Longrightarrow$ If energy wall exists, by Definition 4, then the maximum of energy-efficiency speedup exists, i.e., energy-efficiency speedup S_P^E first increases, then decreases and approaches zero.

$\Longleftarrow$ Since $\lim_{P\to\infty} S_P^E = 0$, energy-efficiency speedup S_P^E first increases and then decreases. So, the maximum of energy-efficiency speedup S_P^E exists, i.e., energy wall exists.

Case 2: $\lim_{P\to\infty} S_P = C$, where $C > 0$ is a positive constant.

Energy-efficiency speedup S_P^E has two curve shapes.

1. If $E(P) \succ S_P$, when the system size is scaled up, energy-efficiency speedup S_P^E first increases, then decreases and approaches zero.
2. If $E(P) \preccurlyeq S_P$, when the system size is scaled up, energy-efficiency speedup S_P^E increases monotonically and approaches a constant.

Thus the proof is similar to that of case $\lim_{P\to\infty} S_P = \infty$. Similarly, the energy wall exists if and only if $\lim_{P\to\infty} S_P^E = 0$.

Case 3: $\lim_{P\to\infty} S_P = 0$.

The energy-efficiency speedup $\lim_{P\to\infty} S_P^E = 0$ first increases, then decreases and approaches zero. So energy wall always exists.

□

This theorem has the following important implication. If $\lim_{P\to\infty} S_P = 0$, then $\lim_{P\to\infty} S_P^E = 0$ according to Equation (6). Thus, the energy wall always exists no matter how energy consumption changes. In the rest of the paper, we therefore focus on the cases when $\lim_{P\to\infty} S_P \neq 0$.

By Equation (6), the factor of parallel energy consumption $E(P)$ is the key factor of energy-efficiency speedup. According to the existence theory of energy wall, we derive the existence corollary of energy wall, which reveals the decisive effect of the factor on the existence of energy wall.

Corollary 1 (Existence Corollary of Energy Wall)**.** Suppose an application G that satisfies $\lim_{P\to\infty} S_P \neq 0$ runs on a P-core system. The energy wall exists if and only if $E(P) \succ S_P$.

Proof.

$\Longrightarrow$ If the energy wall exists, by Theorem 1, $\lim_{P\to\infty} \frac{S_P}{1+E(P)} = \lim_{P\to\infty} S_P^E = 0$. Since $\lim_{P\to\infty} S_P \neq 0$, we have $E(P) \succ S_P$.

$\Longleftarrow$ $E(P) \succ S_P$, then $\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{1+E(P)} = 0$. By Theorem 1, then the energy wall exists.

□

3 SYSTEM CATEGORIZATION

This section categorizes parallel systems by the different characteristics of energy-efficiency speedup when the system size is scaled up, to better understand the effects of energy consumption on scalability.

Section 3.1 provides a classification of supercomputing systems based on their energy consumption. Section 3.2 analyzes the existence of energy wall for different systems.

3.1 Categorizing Systems

When the large-scale parallel system is running, the energy-efficiency speedup exhibits different trends according to different application characteristics or different system architectures. Based on the different trends of energy-efficiency speedup, parallel systems are categorized as red scalable system, green scalable system and yellow scalable system, which are defined as:

Definition 5 (Red Scalable System)**.** Given an application G running on a P-core system, if energy-efficiency speedup satisfies $S_P^E \prec \Theta(1)$, then the system is called red scalable system.

Definition 6 (Green Scalable System)**.** Given an application G running on a P-core system, if energy-efficiency speedup satisfies $S_P^E \succ \Theta(1)$, then the system is called green scalable system.

Definition 7 (Yellow Scalable System)**.** Given an application G running on a P-core system, if energy-efficiency speedup satisfies $S_P^E = \Theta(1)$, then the system is called yellow scalable system.

By the definitions of system categorization and Equation (6), we give the intuitive explanations of the above three systems as follows:

For red scalable system

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{E_P} = \lim_{P\to\infty} \frac{S_P}{1+E(P)} = 0.$$

When the system size is scaled up, the increase speed of energy consumption is faster than that of the computing performance. So, the energy consumption of red scalable system is inefficient and not preferable, since there is large energy waste during system running. Unfortunately, almost all of the current parallel systems are red scalable system, which is analyzed in the Section 3.2.

For green scalable system

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{E_P} = \lim_{P\to\infty} \frac{S_P}{1+E(P)} = \infty.$$

The green scalable system is the energy-efficient parallel system, i.e. the increase speed of energy consumption is slower than that of the computing performance. It means that all energy consumption introduced by parallelization used for improving the computing performance is quite effective, which is the goal of future exascale supercomputing system.

For yellow scalable system

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{E_P} = \lim_{P\to\infty} \frac{S_P}{1+E(P)} = C.$$

where $C > 0$ and C is a constant.

The energy efficiency of yellow scalable system sits between that of red scalable system and green scalable system. When the system size is scaled up, the ratio of increase speed of energy consumption to that of computing performance approaches a constant. Yellow scalable system is a compromise in system and application design.

For ease of understanding, Figure 3 shows the variations of $E(P)$ for red scalable system, green scalable system and yellow scalable system and their corresponding energy-efficiency speedup S_P^E. Traditional speedup is substituted by Gustafson speedup $S_{Gustafson}$, and $E(P)$ varies in forms of P^2, P and $\lg P$, where lg is logarithmic function with base 10.

When $E(P)$ takes the form P^2

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_{Gustafson}}{1+P^2} = 0,$$

the corresponding system is red scalable system.

When $E(P)$ takes the form P

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_{Gustafson}}{1+P} = C,$$

where $C > 0$ is a constant. The corresponding system is yellow scalable system.

When $E(P)$ takes the form $\lg P$

$$\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_{Gustafson}}{1+\lg P} = \infty,$$

the corresponding system is green scalable system.

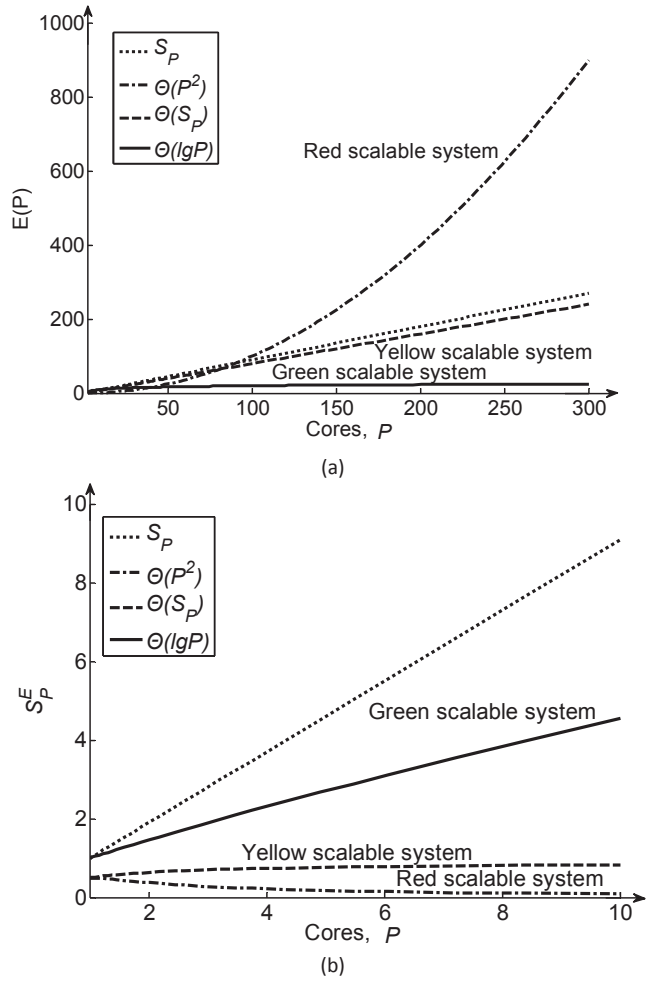

Figure 3. Trends of the increase factor of parallel energy consumption $E(P)$ and the energy-efficiency speedup S_P^E for three kinds of systems. a) $E(P)$ for three kinds of systems. b) S_P^E for three kinds of systems.

In Figure 3 a), when the system size is scaled up, the increased speed of the factor of parallel energy consumption $E(P)$ for red scalable system exceeds that of traditional speedup S_P, hence the gap between S_P and $E(P)$ keeps growing and the corresponding S_P^E in Figure 3 b) approaches zero. The increased speed of $E(P)$ for green scalable system is slower than that of S_P, where the gap between S_P and $E(P)$ widens as shown in Figure 3 a), and the corresponding S_P^E in Figure 3 b) approaches infinity. The trend of $E(P)$ for yellow scalable system is consistent with that of

S_P as shown in Figure 3 a), and the corresponding S_P^E in Figure 3 b) approaches a positive constant.

3.2 System Categorization and Energy Wall

Section 3.1 categorizes parallel systems into red scalable system, green scalable system and yellow scalable system. At the cost of unit energy consumption, the improvement of computing performance for these three systems are different, hence the variations of energy-efficiency speedup are different. This section explores the relationship of energy wall and the system categories.

Based on Corollary 1, we further derive the existence corollaries for different systems.

Corollary 2 (Existence of Energy Wall for Red Scalable System)**.** Suppose an application G that satisfies $\lim_{P\to\infty} S_P \neq 0$ runs on the red scalable system. The energy wall always exists.

Proof. By Definition 5, $\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{1+E(P)} = 0$ and $\lim_{P\to\infty} S_P \neq 0$, so $E(P) \succ S_P$. Based on Corollary 1, the energy wall exists. □

Corollary 3 (Existence of Energy Wall for Yellow/Green Scalable System)**.** Suppose an application G that satisfies $\lim_{P\to\infty} S_P \neq 0$ runs on the yellow or green scalable system. The energy wall does not exist.

Proof. By the yellow or green scalable system Definition 6 or 7, $\lim_{P\to\infty} S_P^E = \lim_{P\to\infty} \frac{S_P}{1+E(P)} \neq 0$, and so $E(P) \preccurlyeq S_P$. Based on Corollary 1, the energy wall does not exist. □

Existence of energy wall for red, yellow and green scalable systems are shown in Table 2, where it is shown that the red scalable system is unscalable, while the yellow and green scalable systems are scalable. The scalable system, i.e. green scalable system and yellow scalable system, is the design objective of system designers or application developers.

System Categorization	Condition	Energy Wall
Red scalable system	$E(P) \succ S_P$	Exists
Yellow scalable system	$\Theta(E(P)) = \Theta(S_P)$	Does not exist
Green scalable system	$E(P) \prec S_P$	Does not exist

Table 2. Existence of energy wall for categorizations

According to Table 2, we come to the conclusion that the relationship of S_P and $E(P)$ is the decisive factor on the existence of the energy wall. $E(P)$ is decided by the extra energy consumption after introducing parallelization for improving performance E_P^W. As shown in Figure 1 b), E_P^W consists of four parts (two kinds of energy

consumption of computation and two kinds of energy consumption of network). As we know, load imbalance accounts for idling, and in the following analysis, we assume the system is load balancing, i.e. $E_W^I = 0$. Therefore, we mainly concern the trends of other three energy consumption of E_P^W, i.e. the redundant energy of serial part in parallel application E_W^S, energy consumption of communication E_E^C and that of network idling E_W^C.

Suppose the execution time of serial part is fixed and denoted by T_P^f, the average dynamic power per core equals to the ratio of a single CN dynamic power to the number of cores per CN, denoted by p_1^C.

$$E_W^S = p_1^C \cdot T_P^f \cdot (P-1).$$

Thus, without regarding the energy consumption of network, $\Theta(E(P)) = \Theta(E_W^S) = \Theta(P)$. For the cases of $\lim_{P\to\infty} S_P = C(C > 0)$ and 0, $\Theta(E(P)) = \Theta(P) \succ \Theta(S_P)$, the parallel computing system is red scalable, and the energy walls always exist. For the scalable application, $S_P = S_{Gustafson} = \Theta(P)$, $\Theta(E(P)) = \Theta(P) = \Theta(S_P)$, and then energy-efficiency speedup $S_P^E = \Theta(1)$, the parallel computing system is yellow scalable and the energy wall does not exist according to Table 2. E_W^S is not the decisive factor of the existence of energy wall, while the energy consumption of network decides the existence of energy wall. Thus, we only analyze the influence of energy consumption of network on the existence of energy wall for scalable applications i.e. $S_P = S_{Gustafson} = \Theta(P)$ in the following case studies.

The energy consumption of network is mainly consisted of the energy consumption of network nodes(NNs) rather than the energy consumption of communication links, where the latter is negligible, hence we focus on analyzing the influence of energy consumption of NNs on the energy wall.

Currently, large-scale systems usually adopt optoelectronic hybrid network. For example, Tianhe-2 uses proprietary interconnect, called TH-Express-2 network which uses 3-level fat tree topology, and the active optical cables (AOCs) are used in cabinet-to-cabinet connection to decrease the communication latency at reasonable cost [8]. For 3-level fat tree TH-Express-2 network, all the switches are built by Network Router Chips (NRCs), and the messages transmit through the fabric in the second and the top levels of the fat tree, where the power of photoelectric conversion of NRCs is a constant (*No. of optical ports per NRC* $\times$ *power per optical port*) and should be taken into account in the energy consumption of NNs. In addition, the power of each NRC is also a constant. Therefore, the average power of each NN is constant function in P and the number of NNs N is the decisive factor of the existence of energy wall rather than the power of each NN from the perspective of scalability.

Suppose the average static power of a single node in the network is denoted by p_0, the average dynamic power of a single node in the network is denoted by p_1, and l is the total times of communication that the messages are transmitted through NNs. p_0, p_1, T_P and l are all constant functions in P.

NN of many current parallel systems integrates special hardware for routing and switching. The message from the source-NN to the target-NN may pass several intermediate NNs, each of which is called a *hop*. Thus, the trend of network energy consumption with P is decided by three coefficients N, hop_j (the total times of network hops for the j^{th} communication) and t_j^c (the time of j^{th} communication).

3D-Torus, binary n-cube and Fat tree are three commonly-used network topologies in large-scale parallel systems [9]. We use these three representative systems to verify and extrapolate the existence of the energy wall. The number of cores per CN is a constant, and denoted by CC. In 3D-Torus and binary n-cube topologies, N is equal to the number of CNs, i.e. $N = \frac{P}{CC} = \Theta(P)$, and m-branch Fat tree needs $N = \frac{P}{m \times CC} log_m \frac{P}{CC} = \Theta(P log_m \frac{P}{CC})$ NNs to connect computation nodes.

We list the energy consumption of network for three topologies in Table 3, where there are two terms in the formulas of energy consumption in network. The first term is derived from the static energy consumption of network. For the Fat tree topology, we have $S_P^E \prec \Theta(1)$ if we only consider the first term. If the parallel computing system is red scalable, the energy wall always exists. For 3D-Torus and binary n-cube topologies, we have $S_P^E \prec \Theta(1)$ (according to hop_j and t_j^c) or $S_P^E = \Theta(1)$, and the system is either red or yellow scalable.

Topology	Energy Consumption of Network	Categorization
3D-Torus	$p_0 \cdot T_P \cdot \Theta(P) + (p_1 - p_0) \sum_{j=1}^{l} hop_j \cdot t_j^c$	Yellow/Red
Binary n-Cube	$p_0 \cdot T_P \cdot \Theta(P) + (p_1 - p_0) \sum_{j=1}^{l} hop_j \cdot t_j^c$	Yellow/Red
m-branch Fat Tree	$p_0 \cdot T_P \cdot \Theta(P \cdot log_m \frac{P}{CC}) + (p_1 - p_0) \sum_{j=1}^{l} hop_j \cdot t_j^c$	Red

Table 3. Existence of energy wall for three commonly adopted network topologies

Furthermore, we take one-to-all broadcasting as an example, and analyze the energy consumption of one-to-all broadcasting on 3D-Torus and binary n-cube topologies respectively. One-to-all broadcasting occupies all network resources, then $hop_j = P$ and t_j^c satisfies at least $\succ \Theta(1)$. Thus, $E(P) \succ \Theta(P) = S_{Gustafson}$, and then energy-efficiency speedup $S_P^E \prec \Theta(1)$. It means that the system is red scalable, and the energy wall exists when there are one-to-all broadcasting on 3D-Torus or binary n-cube topology, even it has only once.

In conclusion, the systems with 3D-Torus, binary n-cube or fat tree topologies are all not green scalable system, because the function of static energy consumption makes $E(P)$ always $\succcurlyeq \Theta(S_P)$. In order to build the green scalable system, we should adopt appropriate methods, such as closing NNs technology with low overhead to avoid the linear or superlinear increase of network energy consumption.

Our energy wall theory provides us insights on reducing energy consumption for improved scalability in a number of principal directions.

- First, the relationship between energy consumption and scalability can be revealed. An energy-efficiency speedup model can be developed and the existence of the energy wall can be analytically predicted.

- Then, by performing an energy wall analysis, we revealed that the key scalability-limiting energy consumption is the network energy consumption. The 3D-Torus and binary n-cube topologies are better than fat tree from the perspective of scalability. So the appropriate topologies should be chosen in the system design in order to mitigate the effect of energy wall.
- Besides, according to the analysis of network energy consumption, the network static energy consumption cannot be ignored. The development of new low energy techniques can be guided by our theory to focus on optimizing the key scalability-limiting factors, such as network static energy consumption.

4 RELATED WORK

Speedup has been almost exclusively used for measuring scalability in parallel computing, such as Amdahl speedup, Gustafson speedup or memory-bounded speedup [6, 7, 10, 11]. These models are only concerned with the computing performance. When supercomputing evolves from high performance to high productivity, it is vital to rethink the performance metric by integrating computing performance and reliability, energy consumption, communication and so on [3, 12, 13, 14]. Recently, there are also some works that discussed the performance scalability and energy for multi-core systems using Amdahl's Law [15, 16]. Ge et al. [17, 18] imported varying power modes into Amdahl's speedup so that different power-performance pairings were applied and lower power was assumed to mean slower performance of a component. In comparison, we integrate the performance and energy consumption from the perspective of scalability, emphasize the energy efficiency of parallel system and then quantitatively define the energy wall. We also provide the existence theorems for the energy wall, which are significant for analyzing energy wall effects with different topologies for guiding system design and hardware/software development. That is the major difference of their works and ours.

Song et al. [19] proposed a system-level iso-energy-efficiency model for predicting energy-performance of data intensive parallel applications and they also demonstrated that this model is helpful for various application contexts and in scalability decision-making. For building energy efficiency model for parallel systems, iso-energy-efficiency model proposed by Song et al. is similar to ours. In comparison, our energy-efficiency speedup model focuses on the impact of the scalability on energy efficiency, and deduces the condition that makes the following theorem: with the increase of the system scale, energy-efficiency speedup has a maximum, i.e., energy wall exists. Song's work, to some extent, supports our conclusion.

The "power wall" is currently one of the major obstacles computer architecture is facing. The power wall proposed by [20, 21, 22] means that uniprocessor performance improvements have come to an end due to power constraints, and emphasizes instantaneous energy consumption or the average energy consumption for a period of time. However the accumulative energy consumption over a period of time is not reflected. Different from power wall, the energy wall is a synthesized concept,

mainly focused on the synthesize characteristics of the performance and energy. But, it should be also noted that the concepts of energy wall and those of power wall are not contradictory.

Energy consumption is one of the major factors that limits the development of a high efficiency computer system. Nowadays, low power optimization in the parallel system includes task scheduling, load balancing and so on [23, 24, 25, 26, 27, 28, 29]. The dynamic compiling technology compiles, alters and optimizes the execution sequence of the application, which is used for reducing the energy consumption of scientific computing and data processing [30, 31, 32].

Most of the above mentioned low power technologies are confined to the optimization of dynamic power consumption, which can reduce the value of energy wall, but cannot remove the wall fundamentally. The future low power technology should pay more attention to the network energy consumption, especially static network energy consumption, for the sake of removing the energy wall.

As the device size of computer design decreases, the number of transistors increases dramatically, which makes the static power (e.g. leakage current) exceed the dynamic power (e.g. dynamic switching) and dominate the power consumption of interconnection networks. Recently, many researches are focusing on the optimization of static power consumption from several aspects, such as buffer, arbitrating component, switch, link, network topology and routing algorithm.

A large number of buffers are the main power consumers of static leakage current in interconnection networks. Chen and Peh found that buffers consumed the majority of the static power consumption of network-on-chip and proposed several designs of power sensitive buffers [33]. Methods of power optimization of static leakage current for cache are also used in that of interconnection network. In [34], Hanson et al. evaluated the effects of several power optimization methods of cache used in interconnection network, such as Gated-VDD [35]. Matsutani et al. applied virtual channels to solve static power and dynamic power simultaneously [36, 37]. Topology optimization is a powerful approach to optimize static power consumption of interconnection networks, and an important method is to customize the topology of networks according to the characteristics of applications [38, 39, 40, 41]. Turnoff-based optimization methods reduce static power through turning off some of the routers, links and so on, or making them sleep [36, 42].

To sum up, nearly no holistic metrics or theories can be used to direct the development of system-level low power technology. The energy wall theory aims at improving the scalability, which directs the system-level low power technology.

5 CONCLUSION

This paper quantifies for the first time the concept of "Energy Wall" and proposes an energy wall theory that allows the effects of energy consumption on scalability of parallel computing systems to be understood and predicted analytically, particularly those at the peta/exascale levels.

The significance of this work is demonstrated by three representative topologies. Our work enables us to mitigate energy wall effects in system design (e.g. by choosing the appropriate topologies) and through applying the network static energy consumption optimizations in hardware/software approaches.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (NSFC) No. 61303068, No. 61221491 and No. 61303061.

REFERENCES

[1] Kothe, D. B.: Science Prospects and Benefits with Exascale Computing. Technical Report ORNL/TM-2007/232, Oak Ridge National Laboratory, 2007.

[2] Simon, H.—Zacharia, T.—Stevens, R.: Modeling and Simulation at the Exascale for Energy and the Environment. http://www.sc.doe.gov/ascr/ProgramDocuments/ProgDocs.html.

[3] Yang, X.—Wang, Z.—Xue, J.—Zhou, Y.: The Reliability Wall for Exascale Supercomputing. IEEE Transactions on Computers, Vol. 61, 2012, No. 6, pp. 767–779.

[4] Top 500, Website, http://www.top500.org.

[5] Shang, L.—Peh, L.-S.—Jha, N. K.: Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks. Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA '03), IEEE Computer Society, Washington, DC, USA, 2003.

[6] Amdahl, G. M: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. Proceedings of the 1967 Spring Joint Computer Conference (AFIPS '67), 1967, pp. 483–485.

[7] Gustafson, J. L: Reevaluating Amdahl's Law. Multiprocessor Performance Measurement and Evaluation, IEEE Computer Society Press, Los Alamitos, CA, USA, 1995, pp. 92–93.

[8] Liao, X. K.—Pang, Z. B.—Wang, K. F.—Lu, Y. T.—Xie, M.—Xia, J.—Dong, D. Z.—Suo, G.: High Performance Interconnect Network for Tianhe System. Journal of Computer Science and Technology, Vol. 30, 2015, No. 2, pp. 259–272.

[9] Jurczyk, M.—Siegel, H. J.—Stunkel, C.: Interconnection Networks for Parallel Computers. 1998.

[10] Sun, X. H.—Ni, L. M.: Scalable Problems and Memory-Bounded Speedup. Journal of Parallel and Distributed Computing, Vol. 19, 1993, No. 1, pp. 27–37.

[11] Sun, X. H.—Rover, D. T.: Scalability of Parallel Algorithm-Machine Combinations. IEEE Transactions on Parallel and Distributed Systems, Vol. 5, 1994, No. 6, pp. 599–613.

[12] SPECpower_ssj2008 (2008). http://www.spec.org/power_ssj2008/.

[13] Green 500. http://www.green500.org.

[14] Yang, X.—Du, J.—Wang, Z.: An Effective Speedup Metric for Measuring Productivity in Large-Scale Parallel Computer Systems. The Journal of Supercomputing, Vol. 56, 2011, No. 2, pp. 164–181.

[15] Woo, D. H.—Lee, H.-H. S.: Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era. IEEE Computer., Vol. 41, 2008, No. 12, pp. 24–31.

[16] Cho, S.—Melhem, R. G.: Corollaries to Amdahl's Law for Energy. Computer Architecture Letter, Vol. 7, 2008, No. 1, pp. 25–28.

[17] Ge, R.—Cameron, K. W.: Power-Aware Speedup. IPDPS, 2007, pp. 1–10.

[18] Cameron, K. W.—Ge, R.: Generalizing Amdahl's Law for Power and Energy. IEEE Computer, Vol. 45, 2012, No. 3, pp. 75–77.

[19] Song, S.—Su, C.-Y.—Ge, R.—Vishnu, A.—Cameron, K. W.: Iso-Energy-Efficiency: An Approach to Power-Constrained Parallel Computation. IPDPS, 2011, pp. 128–139.

[20] Kuroda, T.: CMOS Design Challenges to Power Wall. International Microprocesses and Nanotechnology Conference, Shimane, Japan, 2001, pp. 6–7.

[21] Meenderinck, C.—Juurlink, B.: (When) Will CMPs Hit the Power Wall? Euro-Par 2008 Workshops – Parallel Processing. Springer-Verlag Berlin, Heidelberg, 2009, pp. 184–193.

[22] Gioiosa, R.: Towards Sustainable Exascale Computing. 18th IEEE/IFIP VLSI System on Chip Conference, September 2010, pp. 270–275.

[23] Olsen, C. M.—Morrow, L. A.: Multi-Processor Computer System Having Low Power Consumption. Proceedings of the 2nd International Conference on Power-Aware Computer Systems (PACS '02), Springer-Verlag, Berlin, Heidelberg, 2003, pp. 53–67.

[24] Kadayif, I.—Kandemir, M.—Karakoy, M.: An Energy Saving Strategy Based on Adaptive Loop Parallelization. Proceedings of the 39th Annual Design Automation Conference (DAC '02), ACM, New York, NY, USA, 2002, pp. 195–200.

[25] Freeh, V. W.—Pan, F.—Kappiah, N.—Lowenthal, D. K.—Springer, R.: Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05), IEEE Computer Society, Washington, DC, USA, 2005.

[26] Pan, F.—Freeh, V. W.—Smith, D. M.: Exploring the Energy-Time Tradeoff in High-Performance Computing. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05), IEEE Computer Society, Washington, DC, USA, 2005.

[27] Freeh, V. W.—Lowenthal, D. K.: Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster. Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '05), ACM, New York, NY, USA, 2005, pp. 164–173.

[28] Springer, R.—Lowenthal, D. K.—Rountree, B.—Freeh, V. W.: Minimizing Execution Time in MPI Programs on an Energy-Constrained, Power-Scalable Cluster. Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '06), ACM, New York, NY, USA, 2006, pp. 230–238.

[29] Kappiah, N.—Freeh, V. W.—Lowenthal, D. K.: Just in Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC '05), IEEE Computer Society, Washington, DC, USA, 2005.

[30] Unnikrishnan, P.—Chen, G.—Kandemir, M.—Mudgett, D. R.: Dynamic Compilation for Energy Adaptation. Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design (ICCAD '02), ACM, New York, NY, USA, 2002, pp. 158–163.

[31] Son, S. W.—Chen, G.—Kandemir, M. T.—Choudhary, A. N.: Dynamic Compilation for Reducing Energy Consumption of I/O-Intensive Applications. Languages and Compilers for Parallel Computing (LCPC 2005), Lecture Notes in Computer Science, Vol. 4339, 2006, pp. 450–457.

[32] Wu, Q.—Martonosi, M.—Clark, D. W.—Reddi, V. J.—Connors, D.—Wu, Y.—Lee, J.—Brooks, D.: A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance. Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 38), IEEE Computer Society, Washington, DC, USA, 2005, pp. 271–282.

[33] Chen, X.—Peh, L.-S.: Leakage Power Modeling and Optimization in Interconnection Networks. Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED '03), 2003, pp. 90–95.

[34] Hanson, H.—Hrishikesh, M. S.—Agarwal, V.—Keckler, S. W.—Burger, D.: Static Energy Reduction Techniques for Microprocessor Caches. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, 2003, pp. 303–313.

[35] Hu, Z.—Buyuktosunoglu, A.—Srinivasan, V.—Zyuban, V.—Jacobson, H.—Bose, P.: Microarchitectural Techniques for Power Gating of Execution Units. Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED '04), 2004, pp. 32–37.

[36] Matsutani, H.—Koibuchi, M.—Amano, H.—Wang, D.: Run-Time Power Gating of On-Chip Routers Using Look-Ahead Routing. Proceedings of the 2008 Asia and South Pacific Design Automation Conference (ASP-DAC '08), 2008, pp. 55–60.

[37] Matsutani, H.—Koibuchi, M.—Wang, D.—Amano, H.: Adding Slow-Silent Virtual Channels for Low-Power On-Chip Networks. Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS '08), 2008, pp. 23–32.

[38] Jalabert, A.—Murali, S.—Benini, L.—De Micheli, G.: Xpipescompiler: A Tool for Instantiating Application Specific Networks on Chip. Proceedings of Design, Automation and Test in Europe Conference and Exposition, 2004, pp. 884–889.

[39] Srinivasan, K.—Chatha, K. S.—Konjevod, G.: An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks. Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design (ICCAD '05), 2005, pp. 231–237.

[40] Xu, J.—Wolf, W.—Henkel, J.—Chakradhar, S.: A Design Methodology for Application-Specific Networks-on-Chip. ACM Transactions on Embedded Computing Systems (TECS), Vol. 5, 2006, No. 2, pp. 263–280.

[41] Stensgaard, M. B.—Sparsø, J.: ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology. Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS '08), 2008, pp. 55–64.

[42] Powell, M.—Yang, S. H.—Falsafi, B.—Roy, K.—Vijaykumar, T. N.: Gated-V dd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. International Symposium on Low Power Electronics and Design, July, 2000, pp. 90–95.

Zhiyuan Wang received her B.Sc., M.Sc. and Ph.D. degrees from the National University of Defense Technology in 2003, 2005 and 2011 respectively. She is now Assistant Professor at the State Key Laboratory of High Performance Computing, National University of Defense Technology and the School of Computer, National University of Defense Technology. Her research interests focus on parallel and distributed systems, machine learning, data mining and robotics.

Yuhua Tang is now Full Professor at the State Key Laboratory of High Performance Computing, National University of Defense Technology and the School of Computer, National University of Defense Technology. Her research interests focus on computer architecture.

Juan Chen received her B.Sc. degree from the Department of Computer Science, Southeast University in 2001 and her Ph.D. degree from the School of Computer, National University of Defense Technology, China in 2007. She is now Associate Professor in the State Key Laboratory of High Performance Computing at NUDT, China. Her research interests focus on energy-aware HPC interconnection networks, and parallel software frameworks.

Jingling Xue received his B.Sc. and M.Sc. degrees in computer science from the Tsinghua University, China, in 1984 and 1987, respectively. He received his Ph.D. degree in computer science from the University of Edinburgh, United Kingdom, in 1992. He is currently Professor in the School of Computer Science and Engineering, University of New South Wales. His current research interests include programming languages, compiler optimisations, program analysis, high-performance computing and embedded systems. He is a senior member of the IEEE and a member of ACM.

Yun Zhou received his Ph.D. degree from National Unversity of Defense Technology. He is now Assistant Professor at the State Key Laboratory of High Performance Computing, National University of Defense Technology and the School of Computer, National University of Defense Technology. His research interests focus on machine learning, deep learning and robotics.