Front. Comput. Sci., 2012, 6(4): 339–346
DOI 10.1007/s11704-012-2071-5

RESEARCH ARTICLE

How much power is needed for a billion-thread high-throughput server?

Zhiwei XU (✉)

Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

Abstract With the advent of Internet services, big data and cloud computing, high-throughput computing has generated much research interest, especially on high-throughput cloud servers. However, three basic questions are still not satisfactorily answered: (1) What are the basic metrics (what throughput and high-throughput of what)? (2) What are the main factors most beneficial to increasing throughput? (3) Are there any fundamental constraints and how high can the throughput go? This article addresses these issues by utilizing the fifty-year progress in Little's law, to reveal three fundamental relations among the seven basic quantities of throughput (λ), number of active threads (L), waiting time (W), system power (P), thread energy (E), Watts per thread ω, threads per Joule θ. In addition to Little's law $L = \lambda W$, we obtain $P = \lambda E$ and $\lambda = L\omega\theta$, under reasonable assumptions. These equations help give a first order estimation of performance and power consumption targets for billion-thread cloud servers.

Keywords high-throughput computing, billion-thread servers, power consumption, waiting time and latency, performance formulation, Little's law

1 Introduction

Cloud servers, also known as warehouse-scale computers and datacenter computers [1], are widely used for Internet services, big data applications and cloud computing, and are the target of increasing research interest. The workloads and performance requirements for cloud servers are often different from those of high-performance computing (HPC). The recent objective of the latter is to sustain exascale (10^{18}) floating-point operations per second (flops) on workloads such as large-scale matrix calculations.

Instead, cloud servers need to provide for *high throughput computing* (HTC). For instance, a popular Internet service website (e.g., Google, Baidu, Taobao) needs to accommodate millions of active users, who generate billions of pageviews daily. All these billions of requests must be timely turned over to guarantee desired user experience. These websites also need to execute many data mining jobs, and each job (e.g., a Hadoop hive job) can generate thousands to millions of map-reduce tasks. All these tasks must be timely processed to enable the mining of tens of PB of data daily. These HTC needs are not captured by the HPC requirements favoring high flops.

Although much work has been done on HTC and cloud server research, and many HTC systems are in production use, three basic questions are still not satisfactorily answered.

- **Quantify throughput** This has to do with the basic metrics of HTC. We need a simple, universal, and quantitative definition of throughput that reflects and measures the essential nature of HTC. In other words, we need to answer: what throughput and high-throughput of what?
- **Identify beneficial factors** Which factors affect an HTC system's throughput? What are the main factors most beneficial to enhancing throughput? These factors should be quantifiable to facilitate measurement, analysis, and design optimization.
- **Understand fundamental constraints** Are there any

Received March 8, 2012; accepted June 3, 2012

E-mail: zxu@ict.ac.cn

fundamental constraints? How high can the throughput go? The constraints may be cost, space, form factors, programmability, reliability, etc. This article focuses on the fundamental constraints of power and energy consumptions.

A prevailing approach is to use benchmarks [2] to investigate the above three issues. Our work uses a complementary approach. We identify seven basic quantities of HTC, and utilize the fifty-year progress in Little's law [3,4], to reveal fundamental relations among the HTC basic quantities.

In Section 2, we formulate the performance and power consumption problem of HTC. Section 3 presents the basic assumptions and equations that reveal three fundamental relations among the seven basic quantities of HTC. Section 4 explains the implications of these equations when designing HTC systems. Section 5 estimates why a billion-thread cloud server may be needed in a few years, and the throughput and power consumption needs of a billion-thread cloud server. Section 6 reviews related work. Section 7 offers concluding remarks.

2 Performance and power consumption formulation of an HTC cloud server

A typical application scenario of a cloud server is illustrated in Fig. 1. Many user devices and/or physical object devices (PCs, smartphones, sensors nodes, etc.) interact with the cloud by issuing computing commands to the cloud. The cloud consists of various networks (e.g., the Internet, the mobile Web, Internet of Things) connecting the client devices to a cloud server (the HTC server). The commands reach the cloud server and are processed. Command processing forms the workload of the cloud server.

There are many types of computing commands, with different performance metrics. Table 1 lists several examples from production cloud services sites in China. The first three metrics mainly relate to on-line, customer-facing, request serving workloads, while the fourth has to do with backend, often off-line, data mining workloads. Although we only list numbers from Chinese service sites, workloads of international companies such as Google and Facebook show similar types and ranges.

The metrics in Table 1 are largely application level, even business level, macro metrics. There are many other similar metrics, such as the dollars per month per user metric used as a score in the Cloudstone benchmarks [2].

Designing an HTC cloud server needs to consider more

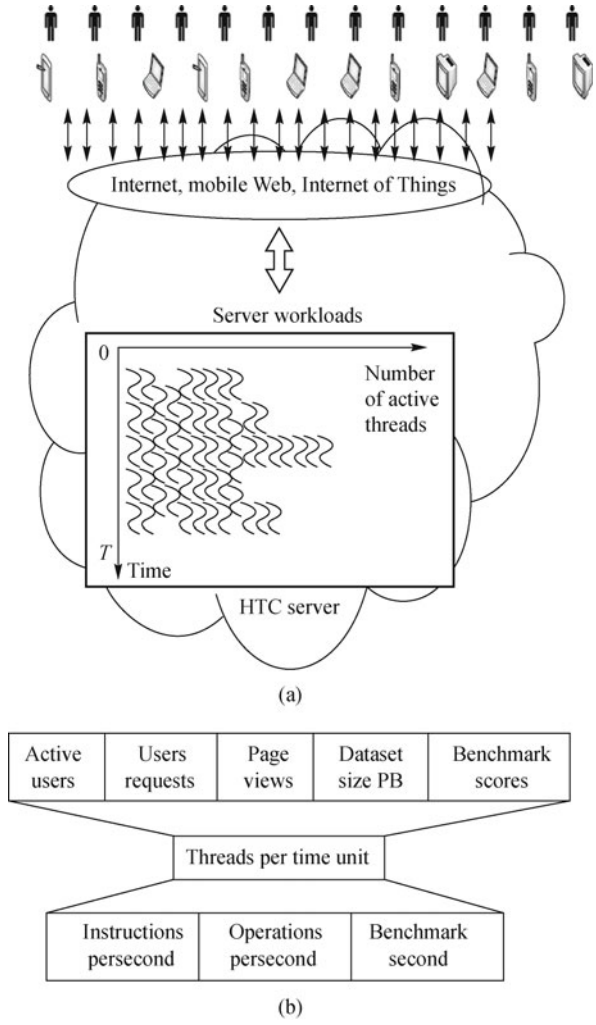

Fig. 1 Formulation of performance and power consumption of an HTC cloud server, (a) A typical HTC cloud server application scenario, (b) Abstraction of the threads per time unit metric following the Internet hourglass model

Table 1 Cloud workload metrics of Baidu, Taobao, and Tencent

	2011 value	Projected value for 2017
AU (million)	80–500	Hundreds
PV (billion)	4–10	20–70
RT (second)	1–6	Sub seconds to 1 second
DS (PB)	A few to tens	Hundreds to 1 000

AU: number of active users
PV: number of daily pageviews
RT: typical response time of a request
DS: size of datasets scanned daily

than such macro metrics. We also need to consider micro metrics such as instructions executed per second, floating-point or integer operations per second, typical transactions per second, queries per second, I/O operations per second, etc.

With so many performance metrics, it is difficult to define and investigate the design space of HTC servers. To address

this problem, we borrow the hourglass model from the Internet community: TCP/IP serves as the neck of the hourglass, which supports many application protocols above and many network technologies below. In other words, we need to develop a "neck" metric for HTC cloud servers.

As shown in Fig. 1, the main metric abstraction in our HTC server model is the executed *threads per time unit* metric, where the time unit can be second, minute, hour, day, month, or year. Any macro metric can become the thread metric, e.g., processing a page request means executing hundreds even thousands of threads; executing a thread in turn corresponds to executing many instructions or operations.

The term thread is used here in its broad meaning, embracing from large grain operating system threads to small grain accelerator threads [5]. In the rest of the paper, we use notations of reference [3], to better utilize the results of Little's law. The main difference is that we use *threads* for *items* in [3].

We define terminology and notations below in Definition 1. These quantities are notational and are valid for any HTC server systems without making any assumptions. We explicitly summarize our assumptions in Assumption 1 below.

Definition 1 Referring to Fig. 1, we are interested in the following quantities and notations. Brief explanations are included with some notations.

1. The HTC server is observed in the time period $[0, T]$, where $0 < T < \infty$. In practice, T can be hours, days, weeks, months, or years.
2. Let there be a total of N threads in the time period $[0, T]$, denoting them as $\tau_1, \tau_2, \ldots, \tau_N$, that arrive and then depart the server system. In general, denote by $\Lambda(T)$ the cumulative number of threads arriving in $[0, T]$.
3. When a thread τ_i $(0 < i \leqslant N)$ arrives at the server at time t_i, it becomes and stays active until time $t_i + w_i$ when it departs the server. In other words, we do not have sleeping threads. The case when a thread follows the active-sleep-active pattern is modeled as two threads. We do not differentiate the processing time and the waiting time of a thread in the server. The total time spent by thread τ_i in the server system is called the waiting time of the thread (also called the latency of the thread), denoted by w_i.
4. Let $\Psi(t)$ denote the set of active threads at time t. Let $n(t) = |\Psi(t)|$ be the number of active threads at time t.
5. Denote by $p_\tau(t)$ the power consumption of thread $\tau \in \Psi(t)$ at time t. The total power consumption of the server system at time t is $P(t)$ Watts, which is a function of the power consumptions of all active threads at t.

$$\begin{aligned} P(t) &= f(p_\tau(t) | \tau \in \Psi(t)) \\ &= f(p_{\tau_1}(t), p_{\tau_2}(t), \ldots, p_{\tau_N}(t)). \end{aligned} \tag{1}$$

The total energy consumed by the server is $\int_0^T P(t)\mathrm{d}t$.

Assumption 1 Following the argument of the Little's law [3], we can make the following assumptions:

1. $n(0) = n(T) = 0$. This is the case of Theorem LL1 in [3], referring to the assumption when the queue is empty at 0 and T. In our HTC server situation, this assumes that the HTC server is inactive at both 0 and T. That is, all threads are processed between 0 and T.
2. At any time t between 0 and T, the total power consumption of the server system is

$$P(t) = \sum_{\tau \in \Psi(t)} p_\tau(t). \tag{2}$$

Equation (2) assumes that the total system power consumption at time t is equal to the sum power consumption of all active threads. It implies that any inactive threads do not consume power, and power consumption is additive.

3. The second assumption is not really valid in practice, as the total system power is not exactly the sum of the power consumption of the active threads. We need to consider other factors, such as power consumed by the system itself and by thread creation, synchronization, and communication. To resolve such issues, we assume that in addition to working threads, we add a number of systems threads to the set of all threads. The power consumed by the server system itself, by inter-thread communication, etc., is assigned to such system threads. In other words, $\Psi(t)$ denotes the set of active working threads and active system threads at time t. This technical treatment allows Eq. (2) to approximate many real cases.

3 Relations among basic quantities

From the formulation in Section 2, we can more formally define seven basic quantities regarding the performance and power consumption of an HTC server, and derive three fundamental relations (expressed as three equations) among these seven quantities.

Definition 2 We are now ready to define five average quantities of an HTC server system, called average parallelism L, average throughput λ, average latency W, average system power P, and average thread energy E, over time period $[0, T]$ and the thread set $\{\tau_1, \tau_2, \ldots, \tau_N\}$. When there is no confusion, we will call them parallelism, throughput, latency, system power, and thread energy.

$$L = \int_0^T n(t)\mathrm{d}t/T = \text{average parallelism over } [0, T]$$
$$= \text{average number of active threads over } [0, T].$$

$$\lambda = \lim_{T\to\infty} \Lambda(T)/T = \text{average arrival rate over } [0, T]$$
$$= \text{average number of arrived threads over } [0, T].$$

$$W = \lim_{N\to\infty} (\sum_{i=1}^{N} w_i)/N = \text{average latency of a thread}$$
$$= \text{average waiting time of a thread over all threads } \{\tau_1, \tau_2, \ldots, \tau_N\}.$$

$$P = \lim_{T\to\infty} \left(\int_{t=0}^{T} (\sum_{\tau\in\Psi(t)} p_\tau(t))\mathrm{d}t \right)/T$$
$$= \text{average system power over } [0, T].$$

$$E = \lim_{N\to\infty} \left(\sum_{i=1}^{N} \int_{t=0}^{\infty} p_i(t)\mathrm{d}t \right)/N$$
$$= \text{average energy consumed by a thread over all threads } \{\tau_1, \tau_2, \ldots, \tau_N\}.$$

Two additional derivative quantities are defined as Watts per thread ω and threads per Joule θ.

$$\omega = E/W = \text{Watts per thread},$$
$$\theta = 1/E = \text{threads per Joule}.$$

The quantity ω can be understood as the average power consumption needed per thread. It is similar to the CPI (cycles per instruction) metric in computer architecture literature [6], but in the energy domain. The quantity θ can be understood as the number of threads executed per Joule of energy consumption, a measure of energy throughput, as λ is the time throughput.

With all definitions and assumptions in place, we are ready to prove the main results relating the quantities. We first restate two existing results in prior literature [3,7,8], in the context of HTC, with threads replacing items.

Theorem 1 (Little's law and a generalization)

1. If limits λ and W exist and are finite, L exists and is finite, and
$$L = \lambda W. \quad (3)$$
2. For each item i, define a (cost) function $f_i(t)$ where $\int_0^\infty |f_i(t)|\mathrm{d}t < \infty$ and $p_i(t) = 0$ for $t \notin [t_i, t_i + w_i]$. Assume the technical condition $w_i/t_i \to 0$ as $i \to \infty$ holds. Define
$$H = \lim_{T\to\infty} \left(\int_{t=0}^{T} \left(\sum_{i=1}^{\infty} f_i(t) \right) \mathrm{d}t \right)/T,$$
$$G = \lim_{N\to\infty} \left(\sum_{i=1}^{N} \int_{t=0}^{\infty} f_i(t)\mathrm{d}t \right)/N.$$
Then, if limits λ and G exist and are finite, H exists and is finite, and
$$H = \lambda G. \quad (4)$$

Now we are ready to present our main theorems.

Theorem 2 Assume an HTC server as defined by Fig. 1 and Definitions 1 and 2. The following two equations hold over the five basic quantities average parallelism L, average throughput λ, average latency W, average system power P, and average thread energy E.

$$L = \lambda W. \quad (5)$$
$$P = \lambda E. \quad (6)$$

Proof Equation (5) follows almost directly from Little's law (Theorem 1). We only briefly show why "limits λ and W exist and are finite" in an HTC server system.

First, we note that in practice, time t in $[0, T]$ is always finite. An HTC server system always has a finite life time, and any observation time period is finite. During any specific $[0, T]$, the (cumulative) number of arrived threads $\Lambda(T)$ is finite and definite. Thus, $\Lambda(T)/T$ has a finite and definite value.

Second, to allow T to approach infinity when an HTC server system has a finite life time, we can play the mathematical trick of assuming an infinite repetition of the behavior of the HTC server. The same finite limit of $\Lambda(T)/T$ prevails. Thus limit λ exists and is finite.

By similar arguments, we can show that limit W exists and is finite.

Intuitively, Eq. (6) follows directly from Theorem 1(2), by using P for H and E for G. But the proof of Eq. (6) is more involved. We first note that the cost function $f_i(t)$ of Theorem 1 in our case is the power consumption function $p_i(t)$. The condition $\int_0^\infty |p_i(t)|\mathrm{d}t < \infty$ naturally holds since any thread has a finite life time and consumes finite power and finite energy. The condition $p_i(t) = 0$ for $t \notin [t_i, t_i + w_i]$ naturally holds since only an active thread consumes power. The technical condition $w_i/t_i \to 0$ as $i \to \infty$ holds following the above

trick of assuming an infinite repetition of the behavior of the HTC server.

The system power of an HTC server is $P(t) = \sum_{\tau\in\Psi(t)} p_\tau(t)$ (see Assumption 1(2)), not obviously $\sum_{i=1}^{\infty} p_i(t)$ as used in limit H. We need to show that these two expressions are the same for the HTC server case. The reason is that $p_i(t) = 0$ for $t \notin [t_i, t_i + w_i]$. That is, inactive threads do not consume power. Thus,

$$\sum_{i=1}^{\infty} p_i(t) = \sum_{\tau\in\Psi(t)} p_\tau(t) + \sum_{\tau\notin\Psi(t)} p_\tau(t) = \sum_{\tau\in\Psi(t)} p_\tau(t) + 0.$$

With all these pieces together, and applying Theorem 1, we have Eq. (6).

Theorem 3 The following equations hold:

$$L/W = P/E. \tag{7}$$

$$\lambda = L\omega\theta. \tag{8}$$

Proof Equation (7) follows by combining Eqs. (5) and (6).

Equation (8) can be proven by using Definition 2 and further manipulating Eqs. (5) and (6). We have

$$\lambda = L/W = L \cdot (E/W) \cdot (1/E) = L\omega\theta.$$

The above definitions and results involve infinity and limits. Following Little's later results and thoughts [3,4], we can consider the finite interval $[0, T]$ and further assume that $n(0) = n(T) = 0$ (Assumption 1(1)), to obtain more intuitive and simpler proofs of the main results.

Theorem 4 Assume an HTC server is observed over the finite interval $[0, T]$ and further assume that $n(0) = n(T) = 0$. Then the following equations hold:

$$L = \lambda W.$$
$$P = \lambda E.$$
$$\lambda = L\omega\theta.$$

Proof Following the argument of the Little's law [3], we have

$$L = \int_0^T n(t)\mathrm{d}t/T.$$
$$\lambda = N/T.$$
$$W = \int_0^T n(t)\mathrm{d}t/N.$$
$$P = \int_0^T P(t)\mathrm{d}t/T.$$
$$E = \int_0^T P(t)\mathrm{d}t/N.$$

Thus

$$L = \int_0^T n(t)\mathrm{d}t/T = (N/T) \cdot \left(\int_0^T n(t)\mathrm{d}t/N\right) = \lambda W.$$

$$P = \int_0^T P(t)\mathrm{d}t/T = (N/T) \cdot \left(\int_0^T P(t)\mathrm{d}t/N\right) = \lambda E.$$

$$\lambda = L/W = L \cdot (E/W) \cdot (1/E) = L\omega\theta.$$

4 Discussions and design implications

Equations (5), (6), and (8) are the three main equations governing the average performance and power consumption metrics of an HTC server system. They not only can be used to analyze an HTC server system, but also provide several insights into HTC server design. In particular, they can be used to address the three issues raised at the beginning of this paper.

- **Quantify throughput** We give a simple, universal, and quantitative definition of throughput λ (average number of threads arrived per second over $[0, T]$). With the additional assumption that the HTC server is empty at 0 and T (i.e., $n(0) = n(T) = 0$), the throughput metric λ measures the average number of threads processed (i.e., have both arrived and then departed) per second over $[0, T]$. Now, given a workload mix, we can set or measure the throughput goal of an HTC server. Similar to the HPC community's next goal of exascale systems, we may start to measure current HTC systems and set the goal for future HTC systems to be, say, a throughput of a billion threads per second by year 2017 and a trillion threads per second by year 2027, all within realistic latency and power consumption constraints.
- **Identify beneficial factors** Which factors affect an HTC system's throughput? Equations (5) and (6) identify four factors: parallelism L, latency W, system power P, and thread energy E. Equation (8) further identifies two more derivative factors: Watts per thread ω and threads per Joule θ. What are the main factors most beneficial to enhancing throughput? The answer provides the basis for a reasonable HTC server design methodology. From Eq. (5), i.e., Little's law, there are two main approaches to enhancing throughput λ: increasing the number of active threads L or decreasing the latency W. Table 1 reveals that in practice, it does not hold that the smaller W the better. W only needs to be small enough to satisfy the desired user experience. There seems to be some marginal effect. Thus it

makes sense to maximize L within the constraint of a small enough W. Equation (8) reveals another reason to increase L in the system architectural design of an HTC server. That is, increasing L does not necessarily increase system power P. In addition, in micro architecture design and technology selections, we want to increase the threads per Joule θ quantity but control the Watts per thread ω.

- **Understand fundamental constraints** Equations (5), (6), and (8) also point out fundamental constraints when designing an HTC server. For instance, suppose an HTC server can support at most 1 000 active threads. Then its throughput can be no more than 1 000/W threads per second. Suppose an HTC server has a maximal system power of 1 MW. Then its throughput can be no more than $1/E$ million threads per second.

Note that Eqs. (5) and (6) may give the misleading indication that throughput λ is always linear with L, $1/W$, P, and $1/E$. This is not necessary true in practice. The reason is that L and W (or P and E) may change simultaneously. To increase L while maintaining a desirable W is a nontrivial research and engineering task.

5 Throughput and power consumption of a billion-thread cloud server

We use an example to illustrate how the above results can be utilized in analyzing and designing future HTC cloud servers.

Let us focus on the design of a high-end HTC cloud server that needs to effectively handle the top workloads of year 2017. Using the projection in Table 1 with some overdesign for future extendibility, we make the following assumptions for the cloud server.

- It needs to handle 100 billion pageviews per day.
- Each pageview request on average generates 10 000 threads that need to be executed. This assumes that future PVs will have richer content expressed with HTML5 constructs.
- The desired user experience requires that the average latency of a thread is no more than 20 milliseconds.

Can such a server be built with an average system power budget of 10 MW? For the daily average case, the throughput is 100 billion PVs times 10 000 threads per PV, equal to 1 million billion threads per day, or 11.574 billion threads per second.

To match the latency W of 20 milliseconds, the server needs to enable in average 11.574 times 0.02 billion active threads. That is, $L = 0.23$ billion threads.

With an average system power budget $P = 10$ MW, the energy consumed by a thread E needs to be controlled within 0.01 billion watts divided by 11.574 billion threads per second. That is, $E = 0.000\,864$ Joules per thread. Furthermore, the Watts per thread metric should be designed with the target of $E/W = 0.000\,864$ Joules per thread divided by 0.02 seconds. That is, the average power consumption per thread ω should be no more than 0.043 2 Watts.

To put this into perspective, a Google search request in 2010 consumed about 1 000 Joules at the server side [9]. If such a request needs 1 000–10 000 threads, that would be 0.1–1 Joules per thread. Thus, we need to improve the energy efficiency by two to three orders of magnitude within seven years.

All definitions and discussion in this paper relate to average values. In designing an HTC server, we also need to consider maximal or minimal values, as well as percentile constraints. In such cases, the definitions and results in this paper can be used as a first order approximation. Much further research is needed.

For instance, assume a peak-to-average ratio of five (this ratio is in the range of 2–10 in practice, according to data collected by the author from real service sites, where cost conscious sites adopt small L). The HTC cloud server defined above would need to handle over 1 billion active threads and a throughput of 58 billion treads per second. It would consume a peak power of 50 MW, if the energy consumed per thread can be controlled within 0.000 864 Joules.

6 Related work

The term HTC was probably first phrased by Miron Livny when developing the Condor software [10] in the 1980's. According to Livny, FLOPS (floating-point operations per second) is the yardstick of HPC systems, but "most scientists are concerned with how many floating-point operations per month or per year they can extract from their computing environment" [10], and such an environment is called an HTC system. Thus FLOPY (floating point operations per year) is the yardstick of HTC systems. Importantly, FLOPY $\neq 356 \times 24 \times 3\,600 \times$ FLOPS, and much research is needed.

In 2008, Ian Foster phrased the term many task computing (MTC) to bridge HTC and HPC, emphasizing the

use of "many computing resources over short periods of time to accomplish many computational tasks". Four MTC workshops were successfully held with the Supercomputing (SC08-SC11) conferences [11].

At the micro-architectural level, an HTC research example is throughput-oriented architectures, where many tasks are executed in parallel, and a task may be the execution of mere hundreds or even fewer instructions [5]. Throughput-oriented architectures focus on maximizing total throughput of parallel workloads, at the cost of increasing the latency of a single task.

Little's law ($L = \lambda W$) was originally presented 50 years ago [12]. A significant generalization is the formula $H = \lambda G$ [7,8]. Professor Little recently reviewed the 50 year progress and applications of Little's law [3] and shared his thoughts. Of particular interest is the following. Traditional mathematical treatments (sample path approach [13] or stationary proofs) produced many results. They often involve mathematic tools like limits and infinity (as shown in Definition 2 and Theorem 1). However, limits and infinity are not a must. Little presented two proofs of Little's law in [3] without involving limits and infinity. In his opinion, research considering systems with finite intervals $[0, T]$ "will often be valuable in practice" [3].

7 Conclusions

This paper focuses on analyzing and designing HTC cloud servers. It gives a performance and power consumption formulation, and proposes executed *threads per time unit* as the main performance metric of HTC cloud servers.

Five basic quantities and two derivative quantities are identified for HTC cloud servers:

- Parallelism L = average parallelism over $[0, T]$ = average number of active threads over $[0, T]$.
- Throughput λ = average arrival rate over $[0, T]$ = average number of arrived threads over $[0, T]$.
- Latency W = average latency of a thread = average waiting time of a thread over all threads $\{\tau_1, \tau_2, \ldots, \tau_N\}$.
- Power P = average system power over $[0, T]$.
- Energy E = average energy consumed by a thread over all threads $\{\tau_1, \tau_2, \ldots, \tau_N\}$.
- Watts per thread ω.
- Threads per Joule θ.

Utilizing research results from Little's law and its generalizations, we derived three basic relations governing these quantities:

- $L = \lambda W$
- $P = \lambda E$
- $\lambda = L\omega\theta$

with the main assumption that the total system power at any time t is equal to the sum of the power consumptions of all active threads at time t.

These results can be used to suggest HTC cloud server design methodologies. A potentially viable approach is to increase parallelism L in the system architectural design, and to increase the threads per Joule quantity θ but control the Watts per thread quantity ω in micro architecture design and technology selections.

An example with data from real websites is used to illustrate how such relations (equations) can be used to estimate performance and power consumption. HTC cloud servers in year 2017 may need to handle a billion active threads, per-thread latency of 20 ms and a throughput of 58 billions threads per second. Such a server could be built within a system power budget of 50 MW, if the energy consumed per thread can be controlled within 0.000 864 Joules.

Acknowledgements This work was supported in part by the Strategic Priority Program of Chinese Academy of Sciences (XDA06010400), the National Basic Research 973 Program of China (2011CB302502) and the Guangdong Talents Program (201001D0104726115). The author is indebted to Dr. Lixin Zhang of National Laboratory of Computer Architecture, Dr. Wensong Zhang of Taobao, Dr. Yongqiang Zou of Tencent and Mr. Shiding Lin of Baidu, for stimulating discussions. The author thanks Mr. Lei Nie for typesetting and proofreading of this paper.

References

1. Barroso L, Hoelzle U. The datacenter as a computer: an introduction to the design of warehouse-scale machines. Synthesis Lectures on Computer Architecture, 2009, 4(1): 1–108
2. Cloudstone. http://radlab.cs.berkeley.edu/wiki/projects/cloudstone
3. Little J. Little's law as viewed on its 50th anniversary. Operational Research, 2011, 59(3): 536–549
4. Little J, Graves S. Little's law. In: Chhajed D, Lowe T J, eds. Building Intuition: Insights from Basic Operations Management Models and Principles. New York: Springer Science and Business Media LLC, 2008
5. Garland M, Kirk D. Understanding throughput-oriented architectures. Communications of the ACM, 2010, 53(11): 58–66
6. Hanlon C. A conversation with john hennessy and david patterson. ACM Queue, 2006–2007, 4(10): 14–22
7. Brumelle S L. On the relation between customer and time averages in

queues. Journal of Applied Probability, 1971, 8(3): 508–520

8. Heyman D, Stidham S J. The relation between customer and time averages in queues. Operational Research, 1980, 28(4): 983–994
9. Glanz J. Google details, and defends, its use of electricity. The New York Times, 2011
10. High-throughput computing. http://research.cs.wisc.edu/condor/htc.html. see also http://en.wikipedia.org/wiki/condor_high-throughput_computing_system and http://en.wikipedia.org/wiki/high-throughput_computing
11. Many-task computing. http://en.wikipedia.org/wiki/Many-task_computing
12. Little J. A proof for the queuing formula: $L = \lambda W$. Operational Research, 1961, 9(3): 383–387
13. El-Taha M, Stidham S. Sample-Path Analysis of Queueing Systems. Springer Netherlands, 1999

Zhiwei Xu is a professor and CTO of the Institute of Computing Technology of the Chinese Academy of Sciences. His research interests include network computing science and Internet operating systems. His prior industrial experience includs chief engineer of Dawning Corp., a leading high-performance computer vendor in China. He currently leads “Cloud-Sea Computing Systems”, a ten-year research project of the Chinese Academy of Sciences that aims at developing billion-thread computers with elastic processors by 2020. Xu holds a PhD from the University of Southern California.