Energy Considerations for Divisible Load Processing

Maciej Drozdowski*

Institute of Computing Science, Poznań University of Technology,
Piotrowo 2, 60-965 Poznań, Poland
Maciej.Drozdowski@cs.put.poznan.pl

Abstract. In this paper we analyze energy usage in divisible load processing. Divisible load theory (DLT) applies to computations which can be divided into parts of arbitrary sizes, and the parts can be independently processed in parallel. The shortest schedule for divisible load processing is determined by the speed of computation and communication. Energy usage for such a time-optimum schedule is analyzed in this paper. We propose a simple model of energy consumption. Two states of the computing system are taken into account: an active state and an idle state with reduced energy consumption. Energy consumption is examined as a function of system parameters. We point out possible ways of energy conservation. It is demonstrated that energy can be saved by use of parallel processing.

Keywords: Energy-efficient computing, performance evaluation, divisible loads.

1 Introduction

Divisible load theory (DLT) is a new parallel processing paradigm applicable in computations which can be divided into parts of arbitrary sizes and processed independently on remote computers. In other words, the DLT assumptions are relevant to computations with fine granularity and negligible data dependencies. Processing big volumes of data is an example of divisible computation. Consider searching for patterns in medical screening photographs. The set of photographs can be partitioned with granularity of one picture. If the number of pictures is big, then the resolution of partitioning the whole dataset is fine. The photographs can be analyzed independently of each other. Other examples of divisible computations include processing measurement data (e.g. SETI@home), image and video processing, linear algebra, search for combinatorial objects (e.g. distributed.net). Divisible load theory originated in the late 1980s [1, 3] as a way to strike a compromise between the communication delays and the gains from faster parallel processing. Surveys of DLT, and its practical applicability can be found in [2, 4, 7].

* Partially supported by grant N N519 1889 33 of Polish Ministry of Science and Higher Education.

Energy consumption of contemporary data centers, and supercomputing facilities is becoming a limiting factor to their growth [6]. Buying power is becoming more and more expensive. For example, the average power usage of the first 5 supercomputers from the current (June 2009) top500 list [9] is over 3.2MW. At the current prices in Poland (≈0.3PLN/kWh) this would cost ≈23.3k PLN daily, and ≈8.5M PLN annually (resp. ≈5.5k, ≈2M EUR). Thus, careful and economic use of energy is an indispensable element of the future high performance computing.

The cost of divisible load processing in general has been analyzed in [5, 8]. The cost considered in [5, 8] could be monetary as well as energy. In both publications two problems were analyzed: to find the minimum cost schedule for a given schedule length, or to find minimum schedule length for a given cost limit. Both papers took into account the cost of computation only. In [8] the sequence of activating heterogeneous computers minimizing the cost was proposed, but computation startup costs were ignored. In [5] it was shown that the above problems are computationally hard (strictly **NP**-hard) when the startup time is non-negligible. Yet, for a given sequence of communications between the processors and the load distributor the problem can be solved by reduction to linear programming. In this paper we assume that performance is the primary criterion. Therefore, the shortest schedules are used. Energy consumption is examined for such time-optimum schedules. Moreover, we concentrate on a more energy-specific representation of the costs of load processing. It is assumed that both the communication and the computations use energy. The costs of computation initiation (startup costs) are taken into account. Similarly to [11] we assume that computing system can be in two states: idle and active. In the idle state power usage is reduced.

The rest of this paper is organized as follows. In the next section we formulate the mathematic model of divisible load processing both with respect to the timing and to the energy cost. In section 3 results of performance evaluation are presented. In section 4 we provide conclusions and discuss lessons learned.

2 Problem Formulation

In this section we outline construction of the optimum length schedule, as well as its energy consumption. Words computer, processor will be used interchangeably. A processor consists of a CPU, memory, and network interface. The CPU and the network hardware can work in parallel such that simultaneous communication and computation is possible. The topology of the processor interconnection is a star (a.k.a. single level tree). In the center of the star resides a processor P_0 called originator (also called master, server) which distributes the load to the remaining processors $P_1, \ldots, P_m$ (called slaves, workers). The star topology may represent a computer cluster in a local area network or a set of computers interconnected in the grid infrastructure. The computing environment is homogeneous.

Timing Model. It is assumed that initially volume V of load resides on the originator. The load is sent from the originator to processors $P_1, \ldots, P_m$ in pieces

of sizes $\alpha_1, \ldots, \alpha_m$. To receive a piece of load a processor has to be *activated* first. The activation process may include transition of the computer hardware and software from the idle to the running state, loading the divisible application runtime environment (such as virtual machines and libraries), starting the application, allocating memory and bringing the application to the state of active waiting for the message with the piece of load. The time of activation is denoted S, and referred to as computation startup time. Sending α_i units of load to processor P_i takes time $\alpha_i C$. Computations for this amount of load take time $\alpha_i A$. It is assumed that the time of returning the results to the originator is very short and can be neglected. It is a common assumption in DLT made for the sake of simplicity in mathematical modeling [2, 4, 7]. We distinguish two cases depending on the participation in the computations of the originator. If the originator is dedicated solely to distributing work then it receives no load (Fig.1a). Otherwise, originator takes part in the computation and processes load $\alpha_0 > 0$ (Fig.2a).

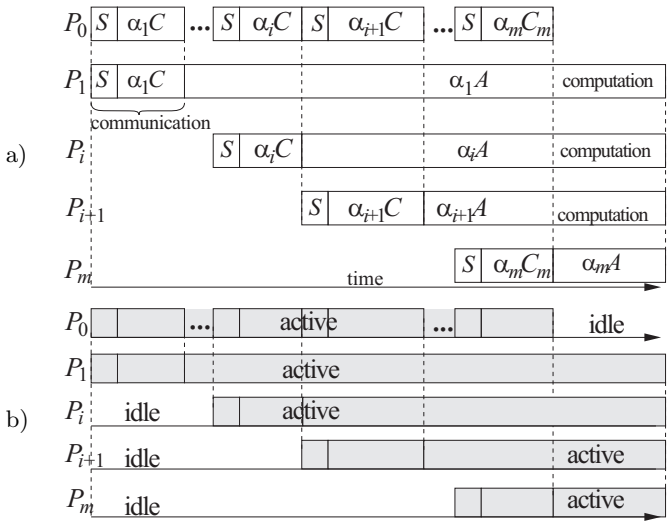

Fig. 1. Initiator *is not* computing. a) Communication and computation schedule. b) Power usage schedule.

The optimum schedule length is obtained by selecting sizes of load chunks $\alpha_1, \ldots, \alpha_m$, and α_0 if applicable. It has been shown in [2] that if the result returning time can be neglected, then for the minimum schedule length all processors must stop computations simultaneously.

Assume that the originator is not computing. The above observation leads to the system of linear equations, from which chunk sizes are derived (cf. Fig.1a):

$$A\alpha_i = S + (C + A)\alpha_{i+1} \quad \text{for} \quad i = 1, \ldots, m-1 \tag{1}$$

$$\sum_{i=1}^{m} \alpha_i = V \tag{2}$$

Let us denote as $\sigma = S/A$, and $\rho = 1 + C/A$. Then,

$$\begin{aligned}
\alpha_i &= \sigma + \rho\alpha_{i+1} = \\
&= \sigma + \rho(\sigma + \rho\alpha_{i+2}) = \sigma + \rho\sigma + \rho^2\alpha_{i+2} = \ldots \\
&= \sigma + \rho\sigma + \ldots + \sigma\rho^{m-i-1} + \rho^{m-i}\alpha_m = \\
&= \frac{\sigma(1-\rho^{m-i})}{1-\rho} + \rho^{m-i}\alpha_m.
\end{aligned} \tag{3}$$

for $i = 1, \ldots m$. From (2) and (3) we obtain

$$\begin{aligned}
V = \sum_{i=1}^{m} \alpha_i &= \sum_{i=1}^{m} \left(\frac{\sigma(1-\rho^{m-i})}{1-\rho} + \rho^{m-i}\alpha_m \right) = \\
&= \frac{\sigma}{1-\rho} \left(m - \frac{1-\rho^m}{1-\rho} \right) + \frac{\alpha_m(1-\rho^m)}{1-\rho}.
\end{aligned} \tag{4}$$

Consequently,

$$\alpha_m = \frac{V(1-\rho)}{1-\rho^m} - \frac{\sigma(m(1-\rho) - 1 + \rho^m)}{(1-\rho^m)(1-\rho)} \tag{5}$$

Note that in the above equation we have subtraction, and α_m may become negative if σ and m are sufficiently big. All such combinations of V, m, C, S, A that $\alpha_m < 0$ are rejected as infeasible. In practice $\alpha_m < 0$ means that the load V is too small to employ all m processors for the current communication parameters C, S and processing rate A. If the programmer still decided to use this number or more processors, then schedule length would grow instead of decreasing. Consequently, efficiency of the schedule would also unnecessarily decrease.

When the originator is not computing schedule length is

$$T(m) = S + (C + A)\alpha_1, \tag{6}$$

where α_1 is calculated from (3) and (5).

If the originator is computing (Fig.2a) then partitioning of the load can be derived in the same way as in the previous case, however, equations (1) and (2) start from $i = 0$. Equation (3) is valid for $i = 0, \ldots, m$. Analogously to (4), (5) we obtain

$$\alpha_m = \frac{V(1-\rho)}{1-\rho^{m+1}} - \frac{\sigma(m(1-\rho) - \rho + \rho^{m+1})}{(1-\rho^{m+1})(1-\rho)}. \tag{7}$$

Schedule length is

$$T(m) = S + A\alpha_0, \tag{8}$$

where α_0 is calculated from (7) and (3).

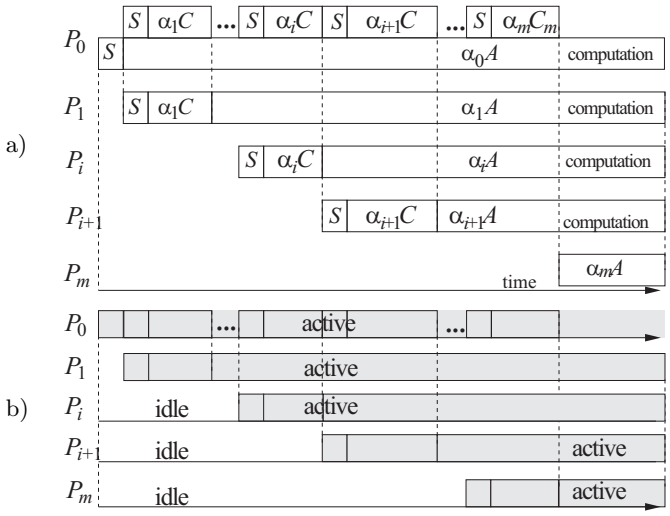

Fig. 2. Initiator *is* computing. a) Communication and computation. b) Power usage.

Energy Use Model. Now let us analyze the energy usage. Both the network and the processors may be either active or idle. It is assumed that active processors consume power P_C, and the active network equipment consumes power P_N. In the idle state power consumption is k times smaller, i.e. P_C/k, and P_N/k, for processors and for the network, respectively. To simplify mathematical formulae we divide the energy usage into two parts: the idle state energy, and the energy beyond the idle state consumed when processors and the network are running. During the whole schedule of length $T(m)$, the originator, m idle processors and the idle network consume energy

$$E_I = T(m)((m+1)P_C + P_N)/k. \tag{9}$$

The network is in the running state at the beginning of the schedule while distributing the load (cf. Fig.1b, and Fig.2b). Suppose the originator is not computing. Processor activation and the load distribution time is $\sum_{i=1}^{m}(S + C\alpha_i) = mS + CV$. The energy consumed above the network idle state is

$$E_{RN} = P_N \frac{k-1}{k}(mS + CV). \tag{10}$$

The originator is active during the whole load distribution time $mS + CV$, which results in energy consumption $P_C \frac{k-1}{k}(mS + CV)$. The remaining processors switch from the idle state to the running state when they are activated. Thus, processor P_i is active for $S + \alpha_i(C + A)$ units of time, consuming energy $P_C \frac{k-1}{k}(S + \alpha_i(C + A))$ above the idle state. The total computation energy

consumption beyond the idle state is

$$E_{RC} = P_C \frac{k-1}{k} \left(mS + CV + \sum_{i=1}^{m} (S + (C+A)\alpha_i) \right)$$
$$= P_C \frac{k-1}{k} (2mS + (2C+A)V). \quad (11)$$

Suppose the originator is computing. The time of communications is $\sum_{i=1}^{m}(S + C\alpha_i) = mS + C(V - \alpha_0)$. The energy consumed by the network beyond the idle state is

$$E_{RN} = P_N \frac{k-1}{k} (mS + C(V - \alpha_0)). \quad (12)$$

The processors together consume beyond the idle state

$$E_{RC} = P_C \frac{k-1}{k} \left(S + A\alpha_0 + \sum_{i=1}^{m} (S + (C+A)\alpha_i) \right) =$$
$$= P_C \frac{k-1}{k} \left((m+1)S + VA + (V - \alpha_0)C \right). \quad (13)$$

The total energy consumed in the computation is

$$E = E_I + E_{RN} + E_{RC}. \quad (14)$$

3 Performance Evaluation

In this section we analyze the amount of energy E necessary to achieve certain schedule length $T(m)$.

Before presenting the details of the simulations let us examine a general relationship between the system and application parameters A, C, S, P_N, P_C, V, the number of used processors m, processing time $T(m)$, and energy E. As mentioned in the previous section the number of processors m that can be exploited depends on A, C, S, V. A general tendency in divisible load processing is that with growing C, S the number of usable processors decreases because communication delays increase and preclude effective use of many processors. On the other hand, with growing A, V the number of usable processors increases because relative contribution of communication delays to the schedule length decreases [2, 4, 7]. Since the reduction in processing time $T(m)$ comes from applying more processors, and the number of usable processors is limited, also the reductions in $T(m)$ are limited. Increasing C, S results in narrower range of processor numbers m where $T(m)$ is reduced. Conversely, increasing A, V widens the range of $T(m)$ reductions. Note that in the following charts $T(m)$ will be shown on the horizontal axis. Now let us examine energy as determined by equations (9) - (13). Intuitively, it can be expected that shorter schedules engage more processors, and hence, should be more costly in energy. Indeed, in all the above equations energy consumption grows with the processor number m. Beyond m, energy consumption depends on constants V, A, C, S, P_C, P_N, k. Optimizing them for minimum

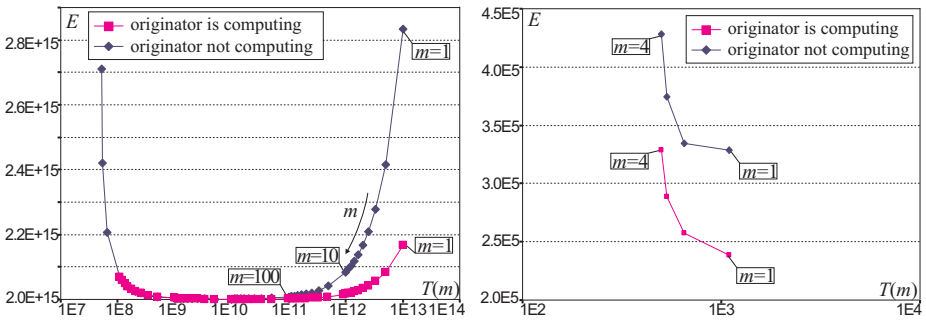

Fig. 3. Energy E vs. schedule length $T(m)$ for $A = 1, C =$ 1E-6, $S =$ 1E2, $P_N = 50, P_C = 200, k = 3$. a) $V =$ 1E13, b) $V =$ 1E3.

power usage is beyond the scope of this paper. Let us now proceed to the results of the simulations. In the following figures we present dependence of the total energy used as defined in (14) versus schedule length defined in equations (6), or (8).

In Fig.3 energy consumption vs. processing time is shown. Values of the parameters used in Fig.3 can be interpreted as follows. Processing one unit of load takes 1s ($A = 1$), transferring it from the originator to the remote processor takes 1μs ($C =$ 1E-6), computation startup time is 100s, the network equipment uses only 50W of power in the active state ($P_N = 50$), a computing processor uses 200W of power ($P_C = 200$), in the idle state power usage is three times smaller ($k = 3$). Let us remind that $T(m)$ is not a real independent variable, because both $T(m)$ and E change as a result of using more processors m. Surprisingly, E as a function of $T(m)$ has a minimum. With increasing processor number execution time is decreasing, as could be expected, but initially also the energy used is decreasing. This behavior of E dependence on $T(m)$ can be explained by several phenomena. Let us assume that the originator is not computing. Note that in (9) the idle state energy depends on $T(m)$. With growing processor number m, execution time $T(m)$ decreases. Therefore, E initially decreases with decreasing $T(m)$. Most of this reduction can be attributed to shorter network and initiator idle state. The relative difference between the highest and the lowest energy consumption in the above experiments ranged from 30% to 40% (originator is not computing). The extent of energy savings may be surprising, considering their source. However, it is a result of long computation time when the communication system remains idle. On the other end of the diagram E is not growing to the infinity because increasing m leads to $\alpha_m < 0$ in equation (5) which means that it is impossible to activate all the processors with the given V. As noted in the previous section we reject such cases as infeasible. A wide plateau of energy usage in Fig.3a results from approximately equal effect of decreasing $T(m)$ in (9) and increasing component mS in (10), (11). With

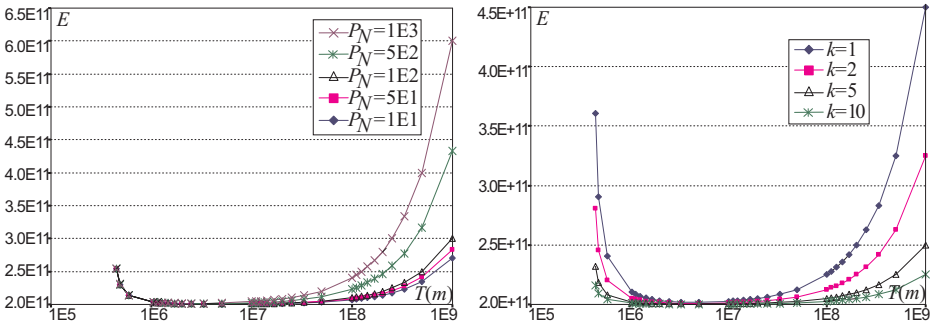

Fig. 4. Energy E vs. $T(m)$ a) for changing P_N at $k = 3$ b) for changing k at $P_N = 50$, and $A = 1, C =$ 1E-6, $S =$ 1E2,$P_C = 200, k = 3$. Originator is not computing.

decreasing problem size V the interval of $T(m)$ with nearly flat energy usage narrows until disappearing completely for $V =$ 1E3, as shown in Fig.3b. The above observations apply also if the originator is computing, though this case is more energy efficient. Consequently, energy savings are smaller. We further discuss the difference between the situation when the originator is computing, or not computing, at the end of this section.

In Fig.4a energy consumption vs. processing time is shown for various values of the network power usage P_N when originator is not computing. The bigger P_N is, the bigger the initial decrease of E with decreasing $T(m)$. On the other hand, when m is big, and $T(m)$ is near its minimum, E_{RC} is dominating in E, and all the functions end overlapping.

In Fig.4b energy consumption vs. processing time is shown for various values of the active to idle power usage ratio k. For instance, $k = 1$ represents the situation when power usage in idle state is no different than in the running state. This means that the whole energy consumption is described in equation (9). As it can be seen the biggest reduction in energy consumption takes place just for $k = 1$ which confirms that the energy savings result from shortening of the idle state. On the other hand, for $k > 1$ the energy savings are shallower in Fig.4b, but the total energy consumption is smaller than for $k = 1$. Let us note that one should not be confused that $k = 1$ is better than $k > 1$ because deeper energy reductions are not the same as smaller overall energy use.

When the originator is computing the dependencies of E on $T(m)$ for changing P_N, k are very similar. Therefore we do not present them here.

Let us now return to the difference between the cases when originator is and is not computing. By subtracting (12) from (10) we obtain the difference in the energy used by the network: $C\alpha_0 P_N(k-1)/k$. Analogously, from (11) and (13) the difference in the energy used by the running processors is $((m-1)S+C(V+$

$\alpha_0))P_C(k-1)/k$. The total difference in energy use is

$$\Delta E = P_N \frac{k-1}{k} C\alpha_0 + P_C \frac{k-1}{k}((m-1)S + C(V+\alpha_0)). \tag{15}$$

Startup times mS cannot dominate in the processing time because otherwise distributed processing would be counterproductive. Hence, $(m-1)SP_C(k-1)/k$ does not constitute a big difference. The remaining components are related to CV and $C\alpha_0$. These gains are especially noticeable if C is big, e.g. $C \approx A$. Moreover, if the originator is computing it is possible to save energy by not sending load α_0 for remote processing. In this case we have an additional computer which nearly immediately starts processing the load. The two cases are juxtaposed in Fig.5. It confirms that the difference between the two cases is big when $C \approx A$. For example, for $C = 0.5$ the difference in energy used is in the range of 130%, while for $C =$ 1E-3 it is not more than 30%.

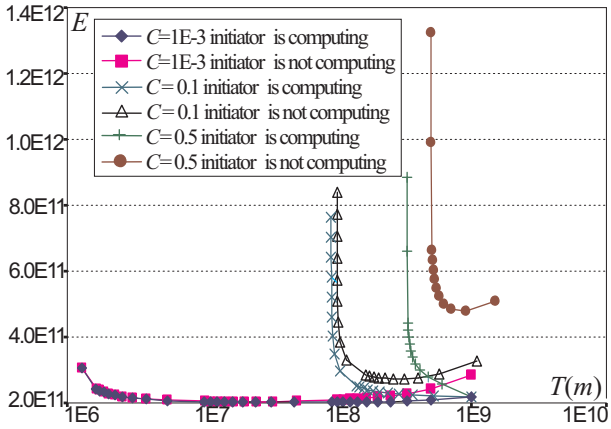

Fig. 5. Energy E vs. schedule length $T(m)$ for $A = 1, S =$ 1E2, $V =$ 1E9, $P_N = 50, P_C = 200, k = 3$ and changing C.

Let us observe that Fig.5 also demonstrates influence of communication rate C on potential energy savings. As it can be seen, for small C energy consumption initially decreases with increasing m, and hence decreasing processing time $T(m)$. The plateau of nearly flat energy consumption spans three orders of magnitude in $T(m)$. On the other hand, for big C energy consumption decreases only marginally, and then quickly grows with m (while $T(m)$ is nearly constant). It can be concluded that small C is essential for allowing reduction in energy consumption. It means that bandwidth must be high. This condition coincides with the requirements for effective communication in parallel applications.

4 Conclusions

In this paper we analyzed energy use in distributed processing of divisible loads. The energy consumed has been presented as a function of the execution time. Surprisingly, it appeared that this function has a minimum, and with decreasing processing time energy used is also decreasing. Hence, we have demonstrated that it is possible to save energy by parallel processing. We compared two ways of processing divisible loads: with and without computations on the load originator. It turns out that using the originator is more energy-efficient. Yet, the differences are apparent only if communication medium is slow.

Our analysis reveals that the savings come from shorter idle state of the communication subsystem. The network idle time is specific to divisible load processing. Similar idle intervals exist in other parallel processing models, e.g., bulk-synchronous processing [10]. Hence, also in other types of parallel applications reduction in network energy consumption should be possible. On the other hand, parallel applications which are communication intensive would have no such network idle time. Consequently, this kind of energy saving would not materialize. The analysis conducted in this paper points to a new way of economizing on energy which is often overlooked. Namely, communication network consumes energy, and also here considerable resources can be saved. Possibly, further savings may be achieved by grouping communications, and switching off the network when it is idle.

References

1. Agrawal, R., Jagadish, H.V.: Partitioning Techniques for Large-Grained Parallelism. IEEE Transactions on Computers 37, 1627-1634 (1988)
2. Bharadwaj, V., Ghose, D., Mani, V., Robertazzi, T.: Scheduling divisible loads in parallel and distributed systems. IEEE Computer Society Press, Los Alamitos CA (1996)
3. Cheng, Y.-C., Robertazzi, T.G.: Distributed computation with communication delay. Transactions on Aerospace and Electronic Systems 24, 700–712 (1988)
4. Drozdowski, M.: Scheduling for Parallel Processing. Springer-Verlag, London (2009)
5. Drozdowski, M., Lawenda, M.: The combinatorics of divisible load scheduling. Foundations of Computing and Decision Sciences 30, 297–308 (2005)
6. Katz, R.H.: Tech titans building boom. IEEE Spectrum 46 (INT), 36–49 (2009) `http://www.spectrum.ieee.org/feb09/7327`.
7. Robertazzi, T.: Ten reasons to use divisible load theory. IEEE Computer 36, 63–68 (2003)
8. Sohn, J., Robertazzi, T.G., Luryi, S.: Optimizing computing costs using divisible load analysis. IEEE Transactions on Parallel and Distributed Systems 9, 225–234 (1998)
9. TOP500 List, June 2009, `http://top500.org/list/2009/06/100`
10. Valiant, L.: A bridging model for parallel computation. Communications of the ACM 33, 103–111 (1990)
11. Woo, D.H., Lee, H.-H.S.: Extending Amdahl's law for energy-efficient computing in the many-core era. IEEE Computer 41, 24–31 (2008)