Energy-aware scheduling for parallel evolutionary algorithms in heterogeneous architectures

Julio Ortega, Juan José Escobar, Antonio Díaz, Jesús González, Miguel Damas[1]

The availability of mechanisms such as dynamic voltage and frequency scaling (DVFS) and heterogeneous architectures including processors with different power consumption profiles allow scheduling algorithms aware of both runtime and energy. In this paper, we propose and evaluate a scheduling strategy that takes into account the relative weights of the workloads and the frequencies and voltages of the different processing cores in a given heterogeneous parallel architecture either to save energy without increasing the running time or to reach a trade-off among time and energy. The parallel algorithms considered to evaluate the proposed scheduling procedure are master-worker evolutionary algorithms whose fitness functions demand high computing times and distribute the fitness evaluation of the individuals among the available cores. As many useful bioinformatics and data mining applications present this profile, the proposed energy-aware scheduling approach could be frequently applied. The experimental results obtained by simulation show relevant energy savings, with values depending on the characteristics of the heterogeneous architecture and on the workload profiles.

1 Background models

Any scheduling procedure locates tasks on the available processors according to predictions about their computational cost and the corresponding energy consumption. Therefore, the procedure also needs information about the characteristics of processors in the system where the tasks are executed. In this section, the models on energy consumption and computing time required by the given tasks are described.

The energy model used by the proposed scheduling procedure is estimated from the power consumption equations corresponding to CMOS circuits that include the terms associated to capacitive, short-circuit, and leakage power. As the most part of previous works, and assuming the capacitive term as the most significant, we will use it to estimate the power consumption in a processor as:

$$P = \beta \times f \times V^2 \tag{1}$$

Where parameter β is related with the product of the number of transistors switching in the processor per clock cycle and the total capacitance load, f is the clock frequency of the processor, and V is the supply voltage. Therefore, the energy E_i consumed by a given task i that requires C_i clock cycles in a processor with a supply voltage V_i can be estimated from (1) by

$$E_i = \beta \times f \times V_i^2 \times \frac{C_i}{f} = \beta \times V_i^2 \times C_i \tag{2}$$

[1]Dept. of Computer Architecture and Technology, CITIC, University of Granada (Spain), jortega@ugr.es, jjescobar@ugr.es, afdiaz@ugr.es, jesusgonzalez@ugr.es, mdamas@ugr.es

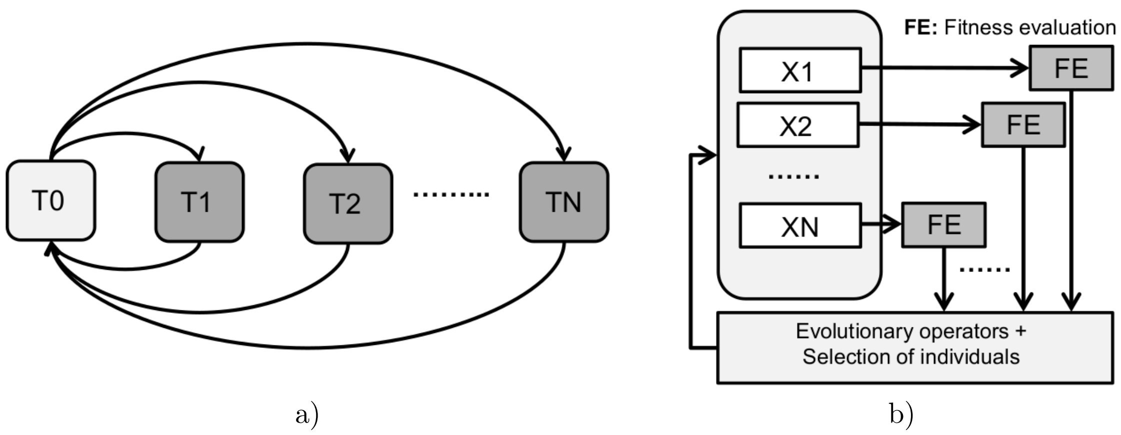

Figure 1: Task dependence graph considered (a), and evolutionary algorithm as example of application with such graph (b)

Whenever a processor is idle, there is also a so called indirect energy consumption that for a given processor k can be estimated by

$$E_k^{\text{idle}} = \beta \times f \times V_{\text{idle}}^2 \times t_k \tag{3}$$

where V_{idle} is the supply voltage of the processor in its idle state, and t_k is the amount of time in which processor k has been in this state. The tasks have to be located on the processors included in a heterogeneous platform with p processors, $P_j(j = 1, ..., p)$. Each processor P_j can operate at different voltage supply levels (VSL), $V_{j,l}(l = 1, ..., \omega(j))$, corresponding to different clock frequencies frequencies $f_{j,l}(l = 1, ..., \omega(j))$.

2 A bi-objective scheduling procedure

This section describes a scheduling procedure that allocates processors and frequencies to tasks trying to minimize both runtime and energy consumption. It can be applied to parallel programs whose tasks dependence graphs are shown in Figure 1.a. In this graph, tasks $T1, T2, ..., TN$ can be executed in parallel after task $T0$, and after synchronizing themselves once they have finished, task $T0$ is executed again and generates another set of parallel tasks $T1, T2, ..., TN$ executed in parallel, and so on. Moreover, the runtime of task $T0$, is negligible with respect to the runtime of each parallel task $T1, ..., TN$. Many useful applications can be parallelized according to the dependence graph of 1.a. Indeed, 1.b schematizes an evolutionary algorithm. Each generation, the fitness of the individuals in the population has to be evaluated according to some performance procedure that could demand a costly computation. For example, in [1] evolutionary multi-objective optimization is applied to solve a feature selection problem in a BCI application. The individuals of the population correspond to different sets of features that define the components of the patterns to be classified. These sets of features have to be evaluated by the accuracy of the classifier once it has been adjusted by using the training patterns characterized by the selected features. The iterations required to train the classifier usually require a high amount of computing time.The fitness evaluation needs between 97.36% (with 30 000 individuals in the population) and 99.93% of the runtime (for 120 individuals).

To define the scheduling strategy, we take into account four parameters, t_{MAX}, t_{max}, t_{min} and t_{MIN}. These parameters can be obtained from the highest and lowest clock cycles values, C_i, required to complete the estimated workloads of the different tasks ($i = 1, ..., n$) and from the frequencies of the available processors, $f_{j,l}(j = 1, ..., p,\ l = 1, ..., \omega(j))$ as follows:

$$\begin{aligned}
t_{MAX} &= max(C_i(i = 1, ..., n))/min(f_{j,l}, (j = 1, ..., p,\ l = 1, ..., \omega(j))) &(4)\\
t_{max} &= max(C_i(i = 1, ..., n))/max(f_{j,l}, (j = 1, ..., p,\ l = 1, ..., \omega(j))) &(5)\\
t_{min} &= min(C_i(i = 1, ..., n))/min(f_{j,l}, (j = 1, ..., p,\ l = 1, ..., \omega(j))) &(6)\\
t_{MIN} &= min(C_i(i = 1, ..., n))/max(f_{j,l}, (j = 1, ..., p,\ l = 1, ..., \omega(j))) &(7)
\end{aligned}$$

The parameter t_{MAX} is the time required by the task with the heaviest workload when it is executed in a processor running at the lowest frequency, the parameter t_{max} is the time required by the task with the highest workload in a processor running at the highest frequency. This way t_{MAX} and t_{max} respectively represent the highest and lowest running times that the heaviest task would require in the present heterogeneous platform. In the same way, t_{min} and t_{MIN} are, respectively, the highest and lowest running times for the lightest task.

It is also possible to define energy consumption parameters, EC_{MAX}, EC_{max}, EC_{min}, and EC_{MIN}, that respectively correspond to the runtimes t_{MAX}, t_{max}, $_{tmin}$, and t_{MIN} as follows:

$$\begin{aligned}
EC_{MAX} &= \beta \times max(Ci(i = 1, ..., n)) \times [min(V_{j,l}, (j = 1, ..., p, l = 1, ..., \omega(j)))]^2 &(8)\\
EC_{max} &= \beta \times max(Ci(i = 1, ..., n)) \times [max(V_{j,l}, (j = 1, ..., p, l = 1, ..., \omega(j)))]^2 &(9)\\
EC_{min} &= \beta \times min(Ci(i = 1, ..., n)) \times [min(V_{j,l}, (j = 1, ..., p, l = 1, ..., \omega(j)))]^2 &(10)\\
EC_{MIN} &= \beta \times min(Ci(i = 1, ..., n)) \times [max(V_{j,l}, (j = 1, ..., p, l = 1, ..., \omega(j)))]^2 &(11)
\end{aligned}$$

The parameters t_{MAX}, t_{max}, t_{min}, and t_{MIN} verify that $t_{MIN} < t_{max} < t_{MAX}$ and $t_{MIN} < t_{min} < t_{MAX}$ while EC_{MAX}, EC_{max}, EC_{min}, and EC_{MIN} verify that $EC_{min} < EC_{MAX} < EC_{max}$ and $EC_{min} < EC_{MIN} < EC_{max}$. Therefore, given a task i with C_i clock cycles, that have been allocated to a processor with supply voltage V_j and frequency f_j, it is possible to define two indexes, Δt and ΔE, with values between 0 and 1, and respectively related with the contribution of the task allocation to the runtime and to energy consumption:

$$\begin{aligned}
\Delta t_{ij} &= \frac{\frac{C_i}{f_j} - t_{MIN}}{t_{MAX} - t_{MIN}} &(12)\\
\Delta E_{ij} &= \frac{K \times C_i \times V_j^2 - EC_{min}}{EC_{max} - EC_{min}} &(13)
\end{aligned}$$

To select a processor and the corresponding frequency for a given task, the scheduling algorithm uses a cost function that takes into account both the energy and runtime objectives through indexes Δt_{ij} and ΔE_{ij}. In our procedure we propose the cost function $\Delta(C_i, f_j) = \Delta t_{ij}^a \times \Delta E_{ij}^b$ with integers a and b greater than or equal to one. This cost function promotes allocations with low values of Δt_{ij} and ΔE_{ij}, and even lower values for one factor whenever the other factor grows. Depending on the relative values of a and b it is possible to give more relevance either to lower runtime or lower energy consumption. The proposed scheduling procedure is shown in Figure 2.

```
$C(i)(i=1,...,p)$    // Cycles of task $i$ to be allocated to a processor
$C_{max} = \max(C(i)(i=1,...,p))$;
$C_{min} = \min(C(i)(i=1,...,p))$;

// Frequency $i$-th of processor $j$-th ($FL$ frequency levels; $p$ processors)
$F(i,j)(i=1,...,FL; j=1,...p)$
$F_{max} = \max(F(i,j)(i=1,...,FL; j=1,...p))$;
$F_{min} = \min(F(i,j)(i=1,...,FL; j=1,...p))$;
Compute $t_{MAX}$, $t_{max}$, $t_{MIN}$, $t_{min}$, and $EC_{MAX}$, $EC_{max}$, $EC_{MIN}$, $EC_{min}$

$C(i)(i=1,...,p)$ is sorted verifying $C(j) \leq C(j+1)(j=1,...,p-1)$;
for $i = 1 : p$
    // Select processor to locate $C(i)$ and frequency
    for $j = 1 : p$
        if processor $j$ has not been previously selected
            for $k = 1 : FL$
                $\Delta(C(i), F(j,k)) = \Delta t^{a}_{i(j,k)} \times \Delta E^{b}_{i(j,k)}$;
            end;
        end;
    end;
    Select the frequency level of processor, $s$, not previously selected, for which
    is obtained the minimum value of $\Delta(C(i), F(j,k))$;
    Mark processor $s$ as selected;
end;
```

Figure 2: Description of the energy-aware procedure for scheduling

3 Performance evaluation

In this section, we analyze the performance of the proposed scheduling procedure. Table 1 describes the three configurations of eight processors with different clock frequencies we have used in our experiments. As can be seen, each configuration corresponds to a different level of heterogeneity. Indeed, *conf1* is homogeneous as all the processors have the same levels of voltage and frequency. The configuration *conf2* includes two different processors while *conf3* includes four. More specifically, Table 1 provides the relative speeds with respect to the one achieved at the highest frequency, which is 1 GHz in all the configurations.

We have also used three different profiles for the evolutionary algorithm. They are defined by the lowest and the highest values for the computing cost of the fitness evaluation tasks, and the lowest difference between two different computing costs. The cost weights for the individuals in the population are randomly selected. This way, the benchmark $b30{\times}100$ includes tasks with computing costs between 100 and 3 000 cycles with cost differences multiple of 100 cycles. The computing costs of the tasks in $b300{\times}100$ go from 1 000 to 30 000 cycles being 100 cycles the lowest difference between tasks, and finally, the tasks in $b30{\times}1\,000$ go from 1 000 to 30 000 cycles with lowest differences of 1 000 cycles between tasks. Each of these files ($b30{\times}100$, $b300{\times}100$ and $b30{\times}1\,000$) includes 100 configurations of task costs with the aforementioned characteristics. These configurations have been randomly selected by using a standard uniform distribution. This way, the averages for the increments in runtime and energy consumption correspond to 100 different

%	P1	P2	P3	P4	P5	P6	P7	P8
conf1	100	100	100	100	100	100	100	100
	80	80	80	80	80	80	80	80
	50	50	50	50	50	50	50	50
conf2	80	80	80	80	100	100	100	100
	64	64	64	64	80	80	80	80
	40	40	40	40	50	50	50	50
conf3	60	60	80	80	90	90	100	100
	48	48	64	64	72	72	80	80
	30	30	40	40	45	45	50	50

Table 1: Relative speeds (in %) in the processors of the configurations used in the experiments

		(a,b)									
		$(1,1)$		$(1,3)$		$(2,1)$		$(3,1)$		$(3,2)$	
Bench.	Conf.	Δt	ΔE	Δt	ΔE	Δt	ΔE	Δt	ΔE	Δt	ΔE
*b*30×100	*conf1*	100.00	−55.80	100.00	−55.80	**0.00**	**0.00**	**0.00**	**0.00**	100.00	−55.40
	conf2	74.68	−60.96	74.68	−60.96	**6.78**	**−10.84**	**6.78**	**−10.78**	75.19	−60.44
	conf3	58.61	−64.18	58.61	−64.18	148.80	−11.74	**24.41**	**−16.05**	59.71	−63.36
*b*300×100	*conf1*	100.00	−55.80	100.00	−55.80	**0.00**	**0.00**	**0.00**	**0.00**	100.00	−55.40
	conf2	75.23	−61.85	75.23	−61.85	**9.37**	**−10.42**	**8.81**	**−10.27**	76.19	−61.05
	conf3	54.76	−65.06	54.76	−65.06	148.50	−11.96	**24.23**	**−15.78**	57.42	−63.60
*b*30×1 000	*conf1*	100.00	−55.80	100.00	−55.80	**0.00**	**0.00**	**0.00**	**0.00**	100.00	−55.40
	conf2	73.59	−60.95	73.59	−60.95	**8.34**	**−10.21**	**7.62**	**−9.97**	73.71	−60.28
	conf3	53.81	−64.18	53.81	−64.18	143.70	−11.82	**21.85**	**−15.87**	54.97	−63.02

Table 2: Averages of increments in runtime and energy consumptions. The values corresponding to the lowest runtime increments are shown in bold characters

cases.

Table 2 shows the results obtained for the averages of the increments in runtimes and energy consumptions by the proposed scheduling procedure for the different benchmarks. These benchmarks have been executed on the three considered configurations described in Table 1. The increments corresponds to the differences in runtime and energy consumption with respect to a random scheduling of the tasks across the different processors. Several couples of values for the parameters a and b in $\Delta(C_i, f_j) = \Delta t_{ij}^a \times \Delta E_{ij}^b$ have been considered.

As can be seen, in the homogeneous configuration *conf1* only it is possible to save energy for the (a,b) couples $(1,1)$, $(1,3)$, and $(3,2)$, that also provide high decrements in the energy consumption in the heterogeneous configurations *conf2* and *conf3*. Nevertheless, in these last two configurations, the couples $(1,1)$, $(1,3)$, and $(3,2)$ also determine increments in the runtime higher than 50%. The (a,b) couples $(2,1)$ and $(3,1)$ produce neither increments in the runtimes nor decreases in the energy consumption whenever the homogeneous *conf1* is considered.

In the heterogeneous configuration *conf2*, the couples $(2,1)$ and $(3,1)$ allow increments in the runtime much lower than those obtained with $(1,1)$, $(1,3)$, and $(3,2)$ although the energy savings are also lower. In the other heterogeneous configuration, *conf3*, this behavior (i.e. lower increments in the runtime with decrements in the energy consumption) is only observed with couple $(3,1)$. The couple $(2,1)$ produces high increments in the runtime (higher than 140%). From Table 2 it is also apparent that given a parameter couple (a, b), the averages in the increments in runtime and energy consumption are similar for the three different distributions of task costs considered ($b30\times100$, $b300\times100$ and $b30\times1\,000$).

4 Conclusions

This paper proposes a scheduling procedure for evolutionary algorithms, with fitness functions requiring high runtimes to be evaluated, that takes into account not only runtime but also energy consumption. The procedure is based on dynamic voltage and frequency scaling (DVFS) and it is useful in heterogeneous architectures including processors with different power consumption profiles. It uses an approximate estimation of the cost of the fitness evaluation task, to build a cost function, $\Delta(C_i, f_j) = \Delta t_{ij}^a \times \Delta E_{ij}^b$, including the effect of energy consumption and speed through two parameters, a and b, that control the trade-off between these two measures. The simulation experiments we have accomplished have shown that by using the adequate combination of those two parameters it is possible to control the strength of each component, runtime and energy consumption. This way, the averages values for the increments of speed and energy consumption obtained across different configurations of task costs show that our procedure is able to reach energy savings of more than 10% with a runtime increment of about 9%. In many configurations of tasks it has been also observed energy savings of about 10% without any increase in the runtime. Given a configuration and a couple of parameters, a and b, similar averages of increments in runtimes and energy consumption have been observed across the considered distribution of task costs.

A lot of researching work still has to be completed. On the one side a more detailed characterization of the heterogeneous configurations of processors in terms of their capabilities to make possible task allocations with the best speed and energy consumption figures would be very useful. The performance evaluation by using profiles of task costs corresponding to real applications should be also completed, along with measuring the real values for energy savings and speeds in the execution of the parallel codes in the available heterogeneous platforms. This way, the usefulness of the consumption models we use in our scheduling procedure would be demonstrated.

Acknowledgements

This work was supported by project TIN2015-67020-P, funded by the Spanish “Ministerio de Economía y Competitividad” and European Regional Development Funds (ERDF).

References

[1] J. Ortega, J. Asensio-Cubero, J. Q. Gan, and A. Ortiz, *Classification of motor imagery tasks for bci with multiresolution analysis and multiobjective feature selection*, BioMedical Engineering OnLine, 15 (2016), p. 73.