Probabilistic Model Checking and Power-Aware Computing[*]

Marta Kwiatkowska Gethin Norman David Parker
School of Computer Science, University of Birmingham
Birmingham, B15 2TT, UK

Abstract

Power-aware computing aims either to maximise the performance of a system under certain constraints on its power consumption and dissipation or, dually, to reduce power consumption in order to meet desired performance or throughput targets. This area is currently gaining importance due to the increasing usage of portable, mobile and hand-held electronic devices. In this paper we illustrate the applicability of probabilistic model checking, a formal verification technique for the analysis of systems which exhibit stochastic behaviour, to the field of power-aware computing. We use the probabilistic model checking tool PRISM on two case studies in this application domain: dynamic power management and dynamic voltage scaling.

1 Introduction

In recent years, the world of computing has witnessed a shift in emphasis from the traditional setting of servers, workstations and desktop machines towards embedded, ubiquitous and pervasive computing. This move from wired to wireless computing and the resulting increased prominence of devices such as handheld computers, wireless sensors and biomedical appliances has made the issue of battery performance and power savings a crucial one. As a result, much research has been done in the area of low-power design, power management and the balance between computation and communication power. One approach which has gained significant attention in the last few years is *system-level power management*, characterised by operating system controlled power saving measures based on the observation of application characteristics. Two distinct flavours of system-level power management are *dynamic power management* and *dynamic voltage/frequency scaling*.

Dynamic power management (DPM) is a way to save energy in devices which, under operating system control, can be switched either on and off or between several *power states* of varying power consumption. Due to the importance of the minimising power consumption in today's embedded systems, a lot of work has been initiated in both the component manufacturing industry and the systems design industry.

Dynamic voltage/frequency scaling (DVS/DFS) is used in real-time embedded systems to achieve a compromise between battery life and performance. The technique is used to schedule a number of tasks which must be executed periodically. Each task has an associated period and a worst-case execution time. The voltage of the system can also be varied during scheduling, which has the effect of reducing the power consumption of the system. This will, however, slow down the execution of the current task. The aim is to schedule tasks and voltage changes in such a way that power consumption is minimised whilst ensuring that all tasks are executed within their deadlines.

In this paper, we illustrate the usefulness of *probabilistic model checking* as a technique for analysing the performance of both of these two approaches to system-level power management. A detailed account of the work concerning DPM schemes appears in [4].

2 Probabilistic Model Checking and PRISM

Probabilistic model checking is a technique for the automatic verification of finite state systems that exhibit probabilistic behaviour. These include randomised algorithms, which use probabilistic choices or electronic coin flipping, and unreliable or unpredictable processes, such as fault-tolerant systems or communication networks.

To perform probabilistic model checking, one first constructs a probabilistic model of the system which is to be analysed. This model is usually a labelled transition system which defines the set of all possible states that the system can be in and the transitions which can occur between these states, augmented with information about the likelihood that each transition will take place.

Properties of the system which are to be verified are then specified, typically in probabilistic temporal logic. These allow reasoning about a wide range of properties such as: "the probability of system shutdown eventually occurring"; "the probability that the video frame will be delivered within 5ms", or "the expected number of failures in the first hour of operation of the system". A probabilistic model checker applies algorithmic techniques to analyse the state space of the probabilistic model and ascertain numerical values corresponding to these properties.

[*]This work was supported by EPSRC grants GR/N22960 and GR/S11107.

PRISM [3, 6] is a probabilistic model checker developed at the University of Birmingham. It supports analysis of three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), and Markov decision processes (MDPs). These probabilistic models are specified in the PRISM language, based on the Reactive Modules formalism of Alur and Henzinger [1]. The basic components of this language are *modules* and *variables*. A system is constructed as the parallel composition of a set of modules. A module's state is determined by a set of finitely ranging variables and it's behaviour is given by a set of guarded commands of the form:

```
[] <guard> → <command>;
```

The guard is a predicate over the variables of the system and the command describes a transition which the module can make if the guard is true. A command is specified by defining the new values of the variables of that module. This means that a module can *read* all of the variables in the system but only *write* to its own variables. In general, behaviour is probabilistic and a command takes the form:

```
<prob> : <action> + ··· + <prob> : <action>
```

where `<prob>` is a probability when the model is a DTMC or MDP and a non-negative, real value (taken to be the parameter of an exponential distribution) when it is a CTMC. In addition, the pair of square brackets at the start of a guarded command can contain a label. Actions from different modules with the same label take place synchronously. See [6] for more details.

Properties of the models constructed in PRISM are expressed in a language which subsumes the well known probabilistic temporal logics PCTL and CSL. PRISM has been used to analyse a wide range of case studies, including probabilistic algorithms for anonymity, contract signing, leader election and consensus; and performance analysis of various queueing systems, communication networks and manufacturing systems (see [6] for references).

3 Dynamic Power Management (DPM)

In this section, we outline how probabilistic model checking has been used to provide a detailed analysis of the stochastic DPM schemes of [7]. The approach of [7] is based on constructing a CTMC model of a dynamic power management system from which, for a given constraint, an optimisation problem is constructed. The solution to this problem is the optimum randomised power management policy satisfying this constraint. The system model is shown in Figure 1 and comprises: a Service Provider (SP), which represents the device under control; a Service Requester (SR), which issues requests to the device; a Service Request Queue (SRQ), which stores requests that are not serviced immediately; and the Power Manager (PM), which issues commands to the SP, based on observations of the system and a stochastic DPM policy. Constraints placed on this system in order to derive an optimum policy take the form of a bound on the average number of requests awaiting service.

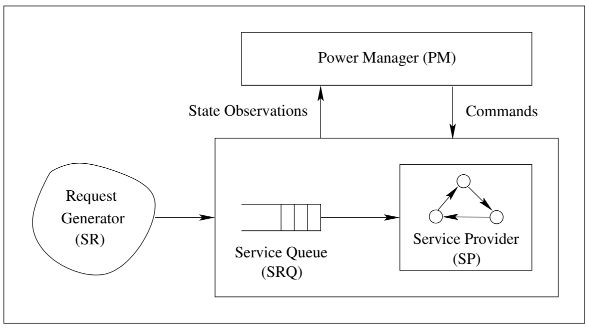


Figure 1. The system model

constraint	policy
av. queue size $\leqslant 5$	if queue is full and SP is sleeping, then go to idle if queue is empty and the SP is in idle, then - go to sleep with probability 0.075140 - stay in idle with probability 0.924860
av. queue size $\leqslant 1$	if queue is full and SP is sleeping, then go to idle if queue is empty and the SP is in idle, then - go to sleep with probability 0.008963 - stay in idle with probability 0.991037
av. queue size $\leqslant 0.1$	if queue is full and SP is sleeping, then go to idle if queue is empty and the SP is in idle, then - go to sleep with probability 7.39e-04 - stay in idle with probability 0.999261

Table 1. Policies under varying constraints

Model Construction.

We have designed generic models of the the DPM system in PRISM and then used this model to construct an optimisation problem whose solution is the optimal policy. Table 1 shows our derived optimal policies for the 3 state device from [7]. this calculation is done by generating the matrices in PRISM and formulating and solving the linear optimisation problem in MAPLE symbolic solver. Once the optimal policy is found, based on the generic description, we construct a model of the system corresponding to this policy. Figure 2 show the PRISM language description of the module corresponding to the PM, operating under the second DPM policy from Table 1.

```
module PM

  pm : [0..1]; //0 - sleep to idle and 1 - idle to sleep

  [sleep2idle] q=QMAX  →  pm'=pm; // sleep to idle
  // probabilistic choice when queue becomes empty
  [serve] q=1  →  0.008963  :  pm'=1; // go to sleep
  [serve] q=1  →  0.991037  :  pm'=0; // stay in idle
  [serve] !q=1  →  pm'=pm; // otherwise loop
  [idle2sleep] pm=1  →  pm'=0; // idle to sleep
  [req] true  →  pm'=0; // reset p when queue no longer empty

endmodule
```

Figure 2. PRISM module of the PM

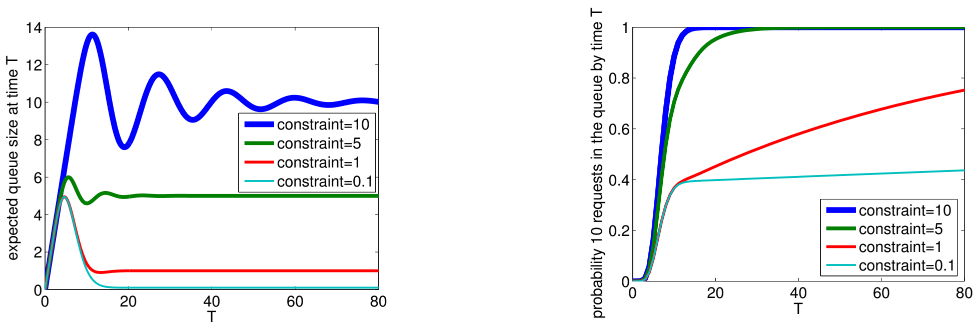


Figure 3. Analysis of different stochastic policies obtained with PRISM

performance constraint	performance measure	inter-arrival time distribution				
		exponential	deterministic	Erlang10	uniform	Pareto
0.1	average queue size	0.099999	0.124273	0.120996	0.112271	0.0226561
	average power consumption	0.960167	0.959464	0.959559	0.959812	0.9624347
1	average queue size	0.999999	1.244302	1.212106	1.125065	0.1297798
	average power consumption	0.928068	0.921180	0.922088	0.924544	0.9525747
5	average queue size	4.999999	5.685810	5.602779	5.371931	0.8491042
	average power consumption	0.785406	0.769478	0.771407	0.776775	0.8814583

Table 2. Average performance of different stochastic policies under various inter-arrival distributions

Model Analysis.

PRISM allows us to compute a wide range of performance measures including: the average number of requests awaiting service and the expected queue size within a given time bound. In Figure 3 we have given some examples of the results obtained with PRISM. In addition, more general distributions can be used to give a more realistic model of the inter-arrival time of requests. Table 3 illustrates the performance of this power manager under varying performance constraints, when the arrival process also varies from the ideal exponential ones. These numbers are computed using steady state probabilities and, in the non-exponential cases, the techniques presented in [2]. The results for the Pareto distribution are in general much smaller than the other distributions which is due to the Pareto distribution's *heavy tail*, which means that, in the long run, many requests will not arrive for a very long time, and hence in these cases the SP will serve all pending requests, and then the system will spend a long time with the queue empty.

4 Dynamic Voltage Scaling (DVS)

We have also used PRISM to model and analyse several dynamic voltage scaling schemes taken from [5]. In the classic model of real-time embedded systems, a set of tasks T_1,...T_n need to be executed periodically. Each task T_i has an associated period P_i and worst-case execution time (WCET) C_i. The task T_i is released every P_i time units and is required to complete its execution before some deadline which is typically defined as the end of its period. A real time scheduler must guarantee that tasks will meet their deadline given that both the task set is schedulable and no task exceeds its worst-case computation time.

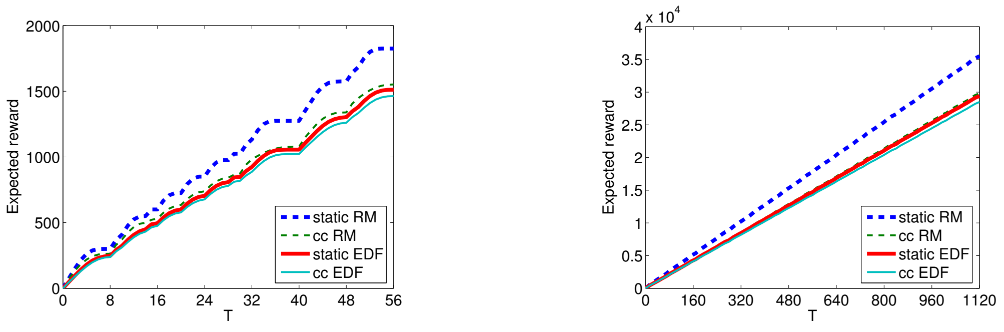
The real-time DVS schedulers in [5] are based on the following standard real-time schedulers:

- Rate Monotonic (RM) schedulers assign the task priority according to the periods - they always select the task with the shortest period that is ready to run.

- Earliest-Deadline First (EDF) schedulers order tasks by their deadlines, giving highest priority to the released task with the most immediate deadline.

As in [5], we assume that the task deadline equals the period, scheduling and preemption overheads are negligible and the tasks are independent.

The first schedulers that [5] consider simply extend RM and EDF by selecting the lowest possible operating frequency that will allow the scheduler to meet all the deadlines of the task set. The frequency is set statically (and are therefore called "*static*") and is only changed when the task set is changed. In general, tasks are completed far sooner than their worst case execution time and in [5] extensions of RM and EDF are introduced which take advantage of this fact. In particular, when a task uses less than its WCET, the scheduler can use this fact to reduce the operating frequency. Note that, when a task is released for execution we do not know how much computation time it will require, and therefore at this point the algorithms make the conservative assumption that the task will require its worst case execution time. However, when a task completes, one can calculate the number of cycles that were not required by the task, and hence at this time the algorithms recalculate the frequency taking into account these unused cycles (hence the term "*cycle conserving*").



(a) Small time scale (b) Large time scale

Figure 4. Expected energy consumption for four different DVS scheduling schemes over time

Model Construction.

We have modelled a processor with three operating frequencies, 1, 0.75 and 0.5 and corresponding voltages of 5, 4 and 3. The task set is of size three with periods $P_1{=}8$, $P_2{=}10$ and $P_3{=}14$ and worst case execution times $C_1{=}3$, $C_2{=}3$ and $C_3{=}1$. To model the system in PRISM, we discretise time and, to simplify the modelling process, the division of one time unit is such that, for each of the possible frequencies, the number of discrete time steps required by a task to finish on its WCET is an integer. Furthermore, we suppose that the completion time distribution of a task is uniformly distributed between 1 and its WCET. For further details on the constructed models see the PRISM website [6]. The constructed models are MDPs in order to capture both probabilistic behaviour (the execution time of each task is random since only a worst-case figure is known) and non-determinism (there exist situations where the schemes do not specify which task to schedule). Since the model is an MDP we examine the worst-case behaviour of *any* implementation of each algorithm.

Model Analysis.

Figure 4 shows a comparison of "the maximum expected energy consumed by a given time bound" for four scheduling schemes outlined above (see [6] for more details). The actual cost measured is the square of the system's voltage, which is proportional to the energy consumed. The comparisons match those observed in [5], obtained through simulation.

5 Conclusions

We show that probabilistic model checking can be effectively used in the field of power-aware computing and in particular in the analysis of both dynamic power management policies and dynamic voltage scaling schedulers.

References

[1] R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15:7–48, 1999.

[2] M. Kwiatkowska, G. Norman, and A. Pacheco. Model checking expected time and expected reward formulae with random time bounds. *Computers & Mathematics with Applications*, 2005. To appear.

[3] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *Proc. QEST'04*, pages 322–323. IEEE, 2004.

[4] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta. Using probabilistic model checking for dynamic power management. *Formal Aspects of Computing*, 2005. To appear.

[5] P. Pillai and K. Shin. Real-time dynamic voltage scaling for low-powered embedded operating systems. *Operating Systems Review*, 35(5):89–102, 2001.

[6] PRISM web page (www.cs.bham.ac.uk/∼dxp/prism).

[7] Q. Qiu and Q. Wu and M. Pedram. Stochastic Modeling of a Power-Managed System: Construction and Optimization. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999.