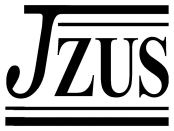
Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn



The dynamic power management for embedded system with Poisson process*

CHEN Tian-zhou (陈天洲), HUANG Jiang-wei (黄江伟), DAI Hong-jun (戴鸿君)

(*School of Computer Science, Zhejiang University, Hangzhou 310027, China*)

E-mail: tzchen@zju.edu.cn; hjw@zju.edu.cn; dahogn@zju.edu.cn

Received Nov. 15, 2004; revision accepted Feb. 3, 2005

Abstract: The mass of the embedded systems are driven by second batteries, not by wired power supply. So saving energy is one of the main design goals for embedded system. In this paper we present a new technique for modelling and solving the dynamic power management (DPM) problem for embedded systems with complex behavioural characteristics. First we model a power-managed embedded computing system as a controllable Flow Chart. Then we use the Poisson process for optimisation, and give the power management algorithm by the help of Dynamic Voltage Scaling (DVS) technology. At last we built the experimental model using the PXA 255 Processors. The experimental results showed that the proposed technique can achieve more than 12% power saving compared to other existing DPM techniques.

Key words: DPM, Flow Chart, Poisson process
doi:10.1631/jzus.2005.AS0070 **Document code:** A **CLC number:** TP303^{+}.3

INTRODUCTION

Embedded systems are driven by second batteries, not by wired power supply. Many methods have been proposed for reducing the energy consumption of individual components. These methods predict the periods when a component is idle or under-utilized (Cai and Lu, 2004). The idling component is turned off or the performance scaled down to reduce energy consumption during these periods. This is called energy management or dynamic power management. High power consumption reduces the battery service life. The goal of low-power design for battery-powered devices is thus to extend the battery service life while meeting performance requirements. Incorporating a dynamic power management scheme in the design of an already-complex system is a difficult process that may require many design iterations and careful debugging and validation (Huang *et al*., 2004a; 2004b).

A simple and widely-used technique is the "time-out" technique, which turns on the component when it is to be used and turns off the component when it will not be used for some pre-specified length of time. Srivastava *et al*. proposed a predictive power management strategy, which uses a regression equation based on the previous "on" and "off" times of the component to estimate the next "turn-on" time (Viredaz and Wallach, 2003). Some past researchers introduced a more complex predictive shut-down strategy with better performance. However, these heuristic techniques could not handle components with more than two ("ON" and "OFF") power modes, not handle complex system behaviors; and they cannot guarantee optimality. Qiu *et al*.(2000) has presented a new modelling method using Generalized Stochastic Petri Nets to solve this problem.

As proposed in (Paleologo *et al*., 1998), a power-managed system can be modelled as a discrete-time Markov decision process by combining the

*Project (No. 2003AA1Z2120) supported by the Hi-Tech Research and Development Program (863) of China

stochastic models of each component. Once the model and its parameters are determined, an optimal power management policy for achieving the best power-delay trade-off in the system can be generated. Qiu and Pedram (1999) improved it by modelling the power managed system using a continuous-time Markov decision process (CTMDP).

Previous works based on Markov decision processes only describe modelling and policy optimization techniques for a simple power managed system containing one Service Provider (SP) providing services (e.g. computing, file access, etc.), one Service Queue (SQ) that buffers the service requests for the SP, and one Service Requestor (SR) generating requests for SP. It is relatively easy to construct the stochastic models of the individual components because their behavior is rather simple, but a significant effort is required to construct the joint model of SP and SQ mainly because of the required synchronization between state transitions of SP and SQ (Qiu *et al*., 2000).

In this paper, we target the more complex power-managed systems as shown in Fig.1 (Huang *et al*., 2004a; 2004b). The example depicts a typical multi-server (distributed computing) system. Note that we are only interested in the system behavior related to the power management. The system contains multiple SPs with their own Local SQs (LSQ). There are multiple SRs generating the tasks (requests) that need to be serviced. The Request Queue (RQ) buffers the requests generate by SRs. The Request Dispatcher (RD) decides about which SP should service which request. Different SPs may have different power/performance parameters. In real applications, the RD and LSQs can be part of the operating system, while SPs can be multiple processors in a multi-processor computing system or number of networked computers of a distributed computing system.

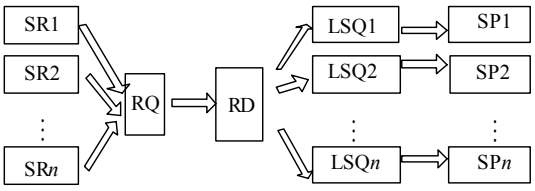


Fig.1 Multi-server and multi-requester/distributed-computing system

The complexity of the modelling problem for the above system is high not only because of the increased number of components, but also because of the complex system behaviors that are present. For example, we need to consider the synchronization of LSQs and SPs, the synchronization of the SRs and RQ, the synchronization of the RQ and LSQs, the dispatch behavior of the RD, and so on. In this situation, the complex system's behaviors are a major part of the system model. The modelling techniques introduced above become powerless because they only offer stochastic models for individual components and require that global system behaviors be captured manually. Obviously, we need new DPM modelling techniques for large systems with complex behaviors. (Qiu *et al*., 2000).

We analyze the whole process of Request Dispatch. After some research, we break the whole process into three major parts as shown in Fig.2 (Huang *et al.*, 2004a; 2004b).

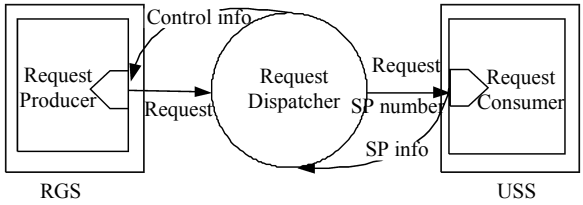


Fig.2 Three major parts of the whole process

The first part is Request Producer (RP) which includes the SRs and RQ used for producing request. The RP's behaviors cannot be controlled by us. The second part of the process includes the Request Dispatcher (RD) used for dispatching the request to a suitable SP. We name this part Request Transfer (RT). The last part called Request Consumer (RC), includes the SPs and LSQs, used for offering service. We can send the request to the suitable SP and adjust the SP's states. Because of the buffer between RP and RT is limited, we can find that the relationship of RP and RT is that of producer and consumer, such as RT and RC. The core part of this model is the RD, whose work of RD is finding the suitable SP, and sending the request to it. In this work, we will introduce a new transferring policy for RD using Poisson distribution.

MODEL ANALYSIS AND OPTIMIZATION

As the processor may not be fully utilized all the time, the variation in system load can be exploited to reduce power dissipation. The processor can be turned off or made to operate at lower speed when the processor has no or little work to do. Some processors allow the voltage to be dynamic adjusted. This is called Dynamic Voltage Scaling (DVS). The relationship among the power consumption rate (P), supply voltage (V_s) and clock frequency (f) can be described by the following formula:

$$P=C\times f\times V_s^2 \tag{1}$$

where C is the switched capacitance (Zhang and Chanson, 2003). It follows that if we decrease the switched capacitance (C), the power consumption will be reduced.

Analyze the Request Producer; the numbers of requests in non-overlapping intervals are independent for all intervals. The probability of exactly one request change in a sufficiently small interval $h\equiv 1/n$ is $p=vh\equiv v/n$, where v is the probability of one change and n is the number of trials. The probability of two or more requests change in a sufficiently small interval h is essentially 0 (Huang *et al*., 2004a; 2004b).

We define generated type A as success and $N(t)$ as the sum of event that occur. Given a Poisson process, the probability of obtaining exactly n successes (type A) in N trials is given by the limit of a binomial distribution (Balakrishnan and Basu, 1996)

$$P_p(n\mid N)=\binom{N}{n}p^n q^{N-n}=\frac{N!}{n!(N-n)!}p^n(1-p)^{N-n} \tag{2}$$

where binomial coefficient is $\binom{N}{n}$. The equation shows the distribution of n successes is a function of the expected number of successes $v=np$. Instead of the sample size N for fixed p, the equation becomes (Weisstein, 2000)

$$\begin{aligned}P_{v/n}(n\mid N)&=\binom{N}{n}p^n q^{N-n}\\&=\frac{N!}{n!(N-n)!}(v/n)^n(1-v/n)^{N-n}\end{aligned} \tag{3}$$

Letting the sample size N become infinity, the distribution then approaches:

$$\begin{aligned}P_v(n)&=\lim_{N\to\infty}P_v(n)\\&=\lim_{N\to\infty}\frac{N(N-1)\ldots(N-n+1)}{n!}\frac{v^n}{N^n}\left(1-\frac{v}{N}\right)^N\left(1-\frac{v}{N}\right)^{-n}\\&=\lim_{N\to\infty}\frac{N(N-1)\ldots(N-n+1)}{N^n}\frac{v^n}{n!}\left(1-\frac{v}{N}\right)^N\left(1-\frac{v}{N}\right)^{-n}\\&=1\cdot\frac{v^n}{n!}\cdot e^{-v}\cdot 1=\frac{v^n e^{-v}}{n!}\end{aligned} \tag{4}$$

which is known as the Poisson distribution (Papoulis, 1984; Pfeiffer and Schum, 1973), in the limit of the number of trials is infinity. Note that the sample size N has completely dropped out of the probability function, which has the same functional form for all values of v.

In the Poisson distribution, the λ is the probability that the event occurred. We cannot get λ at the beginning. We should adjust the λ dynamics by the data we had gathered. Then we can use the equation

$$\lambda_{i+1}=E(x)=\sum_k^i k\frac{\lambda^k e^{-\lambda}}{\lambda!} \tag{5}$$

to adjust λ.

For example, when the system is starting, we give the probability that the event occurred (generating request) as 0.5 (λ=0.5). By the time, we can adjust the λ base the data the system has collected. Then we can adjust the hardware and software environment of SP to improve the response rate, reduce the states changing of modules and improve the power utilization (Huang *et al*., 2004a; 2004b).

Every request in the request queue $R=\{R_1, R_2, \ldots, R_N\}$ has two associated parameters, F, and V. F is the lowest frequency the task needs, and V is the lowest voltage the task needs. Each task can be described using the duple (F, V).

When one request with the associated parameters (F, V) has been finished, the system needs to select a request to the SP. The selecting algorithm is explained in detail after we introduce a few definitions.

Definition 1 If $F_i>F_j$ and $V_i>V_j$, then we said (F_i, V_i)>(F_j, V_j).

Definition 2 We define the difference of (F, V) is (F_i,

V_i)–(F_j, V_j)=(F_i –F_j)×(V_i^2 –V_j^2).

Definition 3 We define the (F_i, V_i)=(F_j, V_j) when F_i=F_j and V_i=V_j.

When the queue was not empty, the RD select the request R_i with (F_i, V_i) that (F_i, V_i)>(F, V) and the result of (F_i, V_i)–(F, V) is the minimal in all requests to run on the SP.

When the queue is empty, the RD will calculate the λ using Eq.(5) for all requests that had happened. If the request has the maximal λ, we think that request will be happened in the next sufficiently small interval.

If (F_i, V_i)>=(F, V), the system can maintains the status until the next request occurs. If (F_i, V_i)<(F, V), the system can adjust the frequency and core voltage to F_i and V_i ahead. The algorithm is shown in Fig.3.

```
if is_empty(RQ) {
        for (all requests that had happened) {
          Calculate (λi)
    }
    Find the requesti with maxλ
    if ((Fi, Vi) >= (F, V)) {
        Return; //Keep_status
    }
    else {
         Adjust_status(Fi, Vi);   //adjust the status
    }
}
else {
        for (all the requests in the RQ)          {
          if ((Fi,Vi)–(FCurrent_small,VCurrent_small)=<0)
                          Current_small=i;
    }
    Adjust_status(Fi, Vi);    //adjust the status
    Dispatch (Ri);
}
```

Fig.3 The algorithm of RD

EXPERIMENTAL MODEL

Our experimental model is very simple. Our target system is a simple distributed computing system. The MSS (Multi-USS) contains two USSs and two RGSs. The RGSs generate requests in the request queue. The RQ capacity is 6. The two USS have different power consumptions and different service speeds. The SQ capacity is 2.

We use the Sistang board as SP and RP. The Sistang Board processor is Xscale PXA 255. Its frequency and Core voltage can be dynamic adjust by instructions.

In this setup, we compare the following methods:

Optimal USS power management+heuristic dispatch policy (Li *et al*., 2003), Time-out+heuristic dispatch. Greedy+heuristic dispatch (Qiu *et al*., 2000).

The comparisons of experimental results for the four methods described above are in the Table 1.

Table 1 Comparisons of experimental results

	vs. Greedy DPM policy	vs. Timeout policy	vs. Local optimal	Average saved
pA=0.2, pB=0.8	22.75%	13.49%	26.33%	20.856%
pA=0.4, pB=0.6	23.91%	14.99%	23.55%	20.816%
pA=0.5, pB=0.5	24.15%	15.87%	26.63%	22.21%
pA=0.6, pB=0.4	23.01%	14.94%	25.9%	21.28%
pA=0.8, pB=0.2	19.62%	13.65%	44.83%	26.03%

CONCLUSION

From the experimental results, we can prove that the Flow Charting mechanism can save more than 12% power compared to all the three methods. The improvement shows the importance of building an accurate system model and obtaining an optimal dispatch policy and power management policy for the SP's for the target distributed computing system.

We use the Flow Chart to model and solve the DPM problem for embedded systems with complex behavioral characteristics. It gives a new and easy way to solve the DPM problem with 12% power saved. The reality indicates that our approach has relatively better performance.

References

Balakrishnan, N., Basu, A.P., 1996. The Exponential Distribution: Theory, Methods, and Applications. Gordon and Breach, New York.

Cai, L., Lu, Y.H., 2004. Dynamic Power Management Using Data Buffers. School of Electrical and Computer Engineering, Purdue University.

Huang, J.W., Chen, T.Z., Lian, Y., Dai, H.J., 2004a. Dynamic Power Management of Complex System Using Flow

Model and Passion Process. IASTED International Conference on Power and Energy Systems (PES 2004).

Huang, J.W., Chen, T.Z., Ye, M.J., Yi, L., 2004b. Dynamic Power Management of Complex Systems Using Flow Model. The First International Conference on Embedded Software and System (ICESS 2004).

Li, D.X., Xie, Q., Pai, H., 2003. Scalable Modeling and Optimization of Mode Transitions Based on Decoupled Power Management Architecture. Irvine, Chou Center for Embedded Computer Systems University of California.

Paleologo, G.A., Benini, L., Bogliolo, A., Micheli, G.D., 1998. Policy Optimization for Dynamic Power Management. Proceedings of Design Automation Conference, p.182-187.

Papoulis, A., 1984. Poisson Process and Shot Noise. Ch.16 in Probability, Random Variables, and Stochastic Processes, 2nd Ed. McGraw-Hill, New York, p.554-576.

Pfeiffer, P.E., Schum, D.A., 1973. Introduction to Applied Probability. Academic Press, New York.

Qiu, Q.R., Pedram, M., 1999. Dynamic Power Management Based on Continuous-Time Markov Decision Processes. Annual ACM IEEE Design Automation Conference, Proceedings of the 36th ACM/IEEE Conference on Design Automation. New Orleans, Louisiana, United States, p.555-561.

Qiu, Q.R., Wu, Q., Massoud, P., 2000. Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets. Department of Electrical Engineering – Systems, University of Southern California.

Viredaz, M.A., Wallach, D.A., 2004. Power evaluation of a handheld computer. *IEEE Micro*, **23**(1):2003.

Weisstein, E.W., 2000. Poisson Distribution. http://mathworld.wolfram.com/PoissonDistribution.html.