The Interplay of Software Bloat, Hardware Energy Proportionality and System Bottlenecks

Suparna Bhattacharya
Indian Institute of Science

Karthick Rajamani
IBM Research

K. Gopinath
Indian Institute of Science

Manish Gupta
IBM Research

Abstract

In large flexible software systems, bloat occurs in many forms, causing excess resource utilization and resource bottlenecks. This results in lost throughput and wasted joules. However, mitigating bloat is not easy; efforts are best applied where savings would be substantial. To aid this we develop an analytical model establishing the relation between bottleneck in resources, bloat, performance and power.

Analyses with the model places into perspective results from the first experimental study of the power-performance implications of bloat. In the experiments we find that while bloat reduction can provide as much as 40% energy savings, the degree of impact depends on hardware and software characteristics. We confirm predictions from our model with selected results from our experimental study.

Our findings show that a software-only view is inadequate when assessing the effects of bloat. The impact of bloat on physical resource usage and power should be understood for a full systems perspective to properly deploy bloat reduction solutions and reap their power-performance benefits.

1. INTRODUCTION

Power-efficient software is being proposed as an important tool [9, 12] in energy optimization of server systems. Meanwhile, in the software engineering domain, researchers have observed that large framework based applications can suffer from runtime inefficiencies due to "software bloat" [6, 10]. "Bloat" is the resource overhead induced by the presence of excess functionality and objects, typically due to the use of highly general components standardized around deeply layered frameworks.

Excess resource usage from bloat could lead to energy inefficiency and reduced performance [10]. Energy savings from reducing bloat are anticipated via both direct and indirect effects (e.g. server consolidation opportunity) of reducing excess resources. However, reducing bloat is non-trivial, particularly because its origin is linked to the same software development trends [6] that have been extremely successful in fueling the growth and widespread impact of redeployable software. Hence, there is a need to develop approaches to assess cost-benefit implications of reducing bloat and focus these efforts where they matter the most.

While reduction of bloat is expected to (obviously) have an impact on power-performance, we find that the inherent slack in large scale IT solution architectures and the presence of elements in the system that are not energy proportional may require this intuition to be qualified.

Is the effect of bloat more pronounced or less pronounced in the presence of energy proportional hardware? Should we expect an increase or a decrease in peak power consumption with lower bloat? In two different experiments [2] on the same hardware platform, but involving different kinds of bloat, one with Apache DayTrader and another with a microbenchmark, we obtained opposite answers. Even with the same workload and same de-bloating optimization, we have seen wide variations in the effects with different hardware and software characteristics. There is, therefore, a need for empirical and analytical studies that validate intuitions and provide a deeper understanding of the relationships between measures of bloat and energy consumption.

The complexity of modern software and system layers make it impractical to compute the exact power-performance impact of run-time bloat (reduction) analytically. We show that an analytical model is still useful in reasoning about implications of reducing bloat. Our analytical model highlights non-intuitive aspects of the complex behavior by abstracting certain relationships at bottleneck zones. This creates a foundation for reasoning quantitatively about the impact of bloat reduction on system power, performance and energy-efficiency.

Contributions:.
(1) We develop an analytical model for studying the implications of runtime bloat on power and energy efficiency under different situations. The model takes into account bottlenecks in the system as well as the energy proportionality characteristics of hardware in the system. We introduce the notion of equiperformance power reduction to characterize the impact, in addition to peak power comparisons. (2) The findings put into perspective our results from the first experimental study of the joint power-performance implications of software bloat, across a range of hardware and software configurations on modern server systems. The experimental results confirm the presence of effects predicted by our model in real systems. (3) Our analysis shows that a whole system perspective is required to properly evaluate benefits of bloat reduction solutions.

2. QUANTIFYING POWER-EFFICIENCY IMPACT OF BLOAT: A SIMPLE ABSTRACT MODEL

Applications use a variety of hardware resources on any given system, e.g., processor cores, on-chip caches, off-chip memory and disk storage. An imbalanced use of these resources can cause a performance bottleneck at one resource and under utilization of others. For example high cache miss rates caused by profligate use of objects due to bloat can cause under-utilization of the processor cores. The power efficiency characteristics differ in nature and magnitude across resources types, e.g., CPU with a super-linear power versus load characteristic when using dynamic voltage and frequency scaling (DVFS) versus main memory with a linear power characteristic and high standby power. This difference leads to a very different impact on energy-efficiency for bloat reduction depending on which resource(s) is impacted by bloat and which are under utilized.

We construct an analytical model using operational laws[1] of queueing theory ([5]) to understand the power performance implications of reducing bloat.

[1]hence, general enough to hold without any assumptions about the distribution of interarrival or service times

When a resource becomes the performance bottleneck because of bloat, reduction of bloat can increase power consumption because of increase in throughput. The varied and sometimes non-linear power characteristics of resources with load can make the impact of bloat reduction on power difficult to assess when compared across different throughput levels. To enable comparison on an equal footing before and after bloat reduction, we also analyze comparisons of power at *equiperformance* levels with and without bloat.

Definition An *equiperformance power comparison* between multiple alternatives compares power measurements taken at the same performance point for each alternative[2]. An equiperformance power comparison also serves as a power-efficiency comparison for that constant performance level.

2.1 The Model

Let $R_i, i = 1..N$, be various types of resources with service demands (time) D_i [5] in the software without bloat. Let $b_i, i = 1..N$, be the overhead due to bloat introduced in each of these resources in the bloated software, changing the service demands to $D_i(1 + b_i)$.

Using asymptotic bounds based on bottleneck analysis [5] to approximate achievable performance, peak throughput is given by $X = min(1/D_i)$ whereas peak throughput with bloat, $X_b = min(1/((1 + b_i)D_i))$.

Let $P_i(U_i)$ be the power consumed by resource R_i with utilization U_i (the exact relationship between utilization and power can be different for different resources). Utilization U_i of resource R_i is D_iX [5], when running the non-bloated software and $D_i(1 + b_i)X_b$ with the bloated software.
Non-bloated power, $P = \Sigma(P_i(U_i)) = \Sigma(P_i(D_iX))$
Power with bloat, $P_b = \Sigma(P_i(D_i(1 + b_i)X_b))$

Power efficiency (perf/watt metric) without bloat,

$$E = X/P = \frac{min(1/D_i)}{\Sigma(P_i(D_iX))}$$

Power efficiency with bloat,

$$E_b = X_b/P_b = \frac{min(1/((1 + b_i)D_i))}{\Sigma(P_i(D_i(1 + b_i)X_b))}$$

We are primarily interested in quantifying potential improvements from bloat reduction. Hence, we define the metrics of interest for relative throughput, peak power, power-efficiency and equiperformance power of the non-bloated software *normalized with respect to that of the bloated software.*

[2]typically at the peak performance point for the lowest performing alternative

Relative throughput with bloat reduction (> 1 is good),

$$\phi_x = (X/X_b) = \frac{min(1/D_i)}{min(1/((1+b_i)D_i))} \quad (1)$$

Relative peak power with bloat reduction (< 1 is good),

$$\phi_p = \frac{P}{P_b} = \frac{\Sigma(P_i(D_iX))}{\Sigma(P_i(D_i(1+b_i)X_b))} \quad (2)$$

Relative power-efficiency with bloat reduction ($>$ 1 is good),

$$\phi_e = \frac{X/P}{X_b/P_b} = (X/X_b)\frac{\Sigma(P_i(D_i(1+b_i)X_b))}{\Sigma(P_i(D_iX))} = \frac{\phi_x}{\phi_p} \quad (3)$$

Relative equiperformance power with bloat reduction (i.e. comparing power consumed by original and bloated software at the same throughput X_b) (< 1 is good)

$$\phi_q = \frac{\Sigma(P_i(D_iX_b))}{\Sigma(P_i(D_i(1+b_i)X_b))} \quad (4)$$

2.1.1 *Effect of degrees of energy proportionality*

An energy proportional hardware resource consumes power in proportion to actual resource utilization (a desirable property of server systems). In the presence of certain power management schemes (e.g. DVFS), this relationship may be super-linear; we characterize this with an exponent which we call the degree of energy proportionality of the resource.

Let us model the power consumed by each resource as $P_i(U_i) = a_iU_i^{\alpha_i} + c_i$, where α_i = degree of energy proportionality of resource R_i.

Let $P_{static} = \Sigma(c_i)$, be the static power of the hardware system, $P_{dyn} = \Sigma(a_iU_i^{\alpha_i})$ be the total dynamic (load-dependent) power consumption. (α_i would be zero if resource R_i is non-energy proportional)

Let $L_i = a_iU_i^{\alpha_i}$ be the load-dependent power for resource R_i. Thus $P_{dyn} = \Sigma(L_i)$, $P = P_{static} + P_{dyn}$

Relative peak power impact with bloat reduction:

$$\phi_p = \frac{\Sigma(L_i(D_iX_b)\phi_x^{\alpha_i}) + P_{static}}{\Sigma(L_i(D_iX_b)(1+b_i)^{\alpha_i}) + P_{static}}$$

Defining f_i = fraction of load dependent power consumed by resource R_i when running bloated software (wrt total system power), f_s = fraction of static power consumed with the bloated software ($f_s + \Sigma f_i = 1$), the above can be re-written as:

$$\phi_p = \Sigma(f_i(\frac{\phi_x}{1+b_i})^{\alpha_i}) + f_s \quad (5)$$

As for the relative equiperformance power,

$$\phi_q = \Sigma(\frac{f_i}{(1+b_i)^{\alpha_i}}) + f_s \quad (6)$$

2.2 System Bottlenecks and Bloat: A Curious Interaction

Consider the situation where bloat primarily affects the demand for a single resource. Let R_k be the bloated resource, then $b_k > 0$ and $b_i = 0, \forall i \neq k$. We show how the impact of bloat reduction depends on where the primary bottleneck is relative to the bloat site. Table 1 summarizes the impact on performance, peak power, and equiperformance power.

Bloat at non-bottleneck resource.

When bloat does not affect the bottleneck resource, $\phi_x = 1$, i.e. there is *no change in performance with bloat reduction.* Substituting in equation 5, we obtain:

$$\phi_p = \frac{f_k}{(1+b_k)^{\alpha_k}} + \Sigma_{i\neq k}f_i + f_s \leq 1$$

Peak power *decreases* with bloat reduction, showing a higher improvement when the bloated resource has a steeper (larger α_k) power-to-load characteristic and consumes a higher fraction of system power (larger f_k).

Since there is no change in performance $\phi_e = 1/\phi_p$, i.e., the relative power-efficiency improves with bloat reduction. And $\phi_q = \phi_p$, i.e., the relative equiperformance power is the same as the relative power at peak performance.

Bloat at Bottleneck Resource.

If bloat affects the bottleneck resource, i.e. $k = argmin(1/D_i)$, then $\phi_x = 1 + b_k > 1$, i.e. *throughput improves with bloat reduction*, maximum improvement being $1 + b_k$ when bloat is eliminated. Substituting in equation 5

$$\phi_p = f_k + \Sigma_{i\neq k}(f_i(1+b_k)^{\alpha_i}) + f_s \geq 1$$

Peak power *increases* or remains the same depending on the power characteristics of the *non-bloated resources*, in contrast with the previous case. Reducing bloat allows more productive use of the bottleneck resource, improving peak throughput. If this increase in throughput increases the usage of resources that were under-utilized earlier because of the bloat-affected bottleneck, and this increases their power consumption, then the power consumed by the application at peak throughput can increase.

$$\phi_e = (1+b_k)/\phi_p <= 1 + b_k$$

Relative power efficiency improvement is less than or equal to the throughput gain. A steeper energy proportionality characteristic of the other (non bloated) resources lowers the efficiency improvement from bloat reduction, especially if their power consumption is significant compared to the power con-

	Rel Peak perf ϕ_x	Rel power at peak perf. ϕ_p	Rel equi-perf power ϕ_q
Bloat at non-bottleneck resource	1 (no improvement)	$\frac{f_k}{(1+b_k)^{\alpha_k}} + \Sigma_{i \neq k} f_i + f_s \leq 1$ (same if $\alpha_k = 0$, else decreases)	$\frac{f_k}{(1+b_k)^{\alpha_k}} + \Sigma_{i \neq k} f_i + f_s \leq 1$ (same if $\alpha_k = 0$, decreases o/w)
Bloat at bottleneck resource	$1 + b_k$ (improves)	$f_k + \Sigma_{i \neq k}(f_i(1+b_k)^{\alpha_i}) + f_s \geq 1$ (same if $\alpha_i = 0, \forall i \neq k$, else increases)	ditto as above
Bloat reduction shifts bottleneck	$1 + b_{eff}$ (improves, but less)	$f_k(\frac{1+b_{eff}}{1+b_k})^{\alpha_k} + \Sigma_{i \neq k}(f_i(1+b_{eff})^{\alpha_i}) + f_s$ (can increase or decrease or stay same)	ditto as above

Table 1: Effect of bloat reduction in different scenarios when bloat affects a single resource R_k

sumption of the bloat-impacted bottleneck resource. The highest improvement from bloat reduction occurs in the case when the non-bottleneck resources are non-energy proportional (when $\alpha_i = 0, \forall i \neq k$).

Bloat reduction shifts bottleneck.

If reducing bloat causes the bottleneck to shift from R_k to R_l, then, $1 < \phi_x < 1 + b_k$, i.e. throughput improves with bloat reduction (but to a lower extent than the previous case). Let us term $b_{eff} = \phi_x - 1$ as the effective bloat factor. Now the analysis for peak power and power efficiency are similar to the previous case, adjusting for b_{eff}.

Equiperformance power is impacted to the same extent in all the three cases above. Performance and equi-performance power are generally improved with bloat reduction. However, power at peak performance can increase with bloat reduction as a result of increased throughput following reduced pressure at a bottleneck resource.

2.3 Experimental observations discussion

We complement the conclusions from our model-based analysis below with some experimental observations (Figure 1). These observations are a subset of detailed multi-platform experiments for quantifying the impact of software bloat [2]. Three levels of cache resource pressure (bottleneck) are discussed in the observations, with bloat's impact on cache pressure increasing from SMT2 to SMT4 and further to SMT4 with half the original cache capacity (HC) on a Power 750 system running the SPECPower_ssj2008 benchmark modified to increase or decrease the extent of bloat. Bloat primarily impacts cache resources while the cores (which see a smaller impact from bloat) consume a higher fraction of power.

1. Peak power can *increase or decrease* with reduction in bloat depending on whether it affects the bottleneck resource or other resources. In the situation where bloat affects the bottleneck, *the degree of impact can depend on the steepness of energy proportionality characteristic of the resources which are not bloated* and the fraction of system power consumed by them.

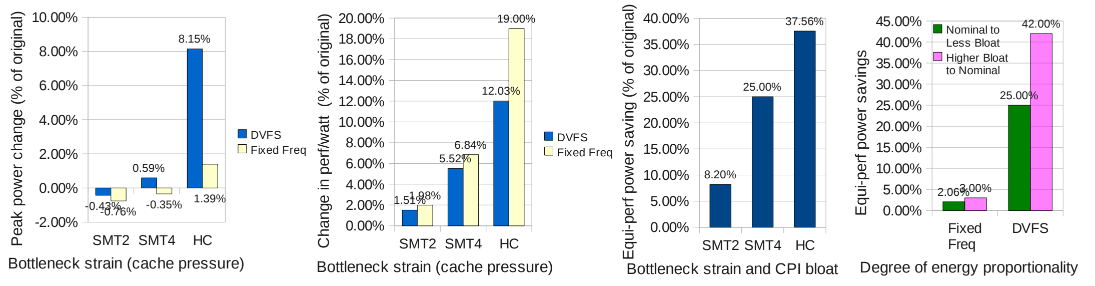
In our experiments, bloat induced cache pressure causes under-utilization of the compute resources for SMT4 and HC, hence throughput increases with bloat reduction. Figure 1(a) shows the impact of this on peak power - the effect is most pronounced when cores have a steeper power vs load characteristic (with DVFS) as predicted by our model. Reducing bloat also reduces memory references and consequently memory power, particularly in the SMT2 case; the effects are small as memory consumes a low fraction of system power on this system.

2. The energy-efficiency improvement from reducing bloat is likely to be most pronounced when bloat affects a bottleneck resource and the non-bottleneck resources are not very energy proportional. *Energy proportional hardware mitigates the effect of bloat on energy efficiency at peak performance* - Figure 1(b) confirms this showing the gains in energy efficiency are greater when running the cores at fixed frequency than with DVFS.

3. While energy efficiency improvement at peak performance is higher with non energy proportional resources, *the improvement at equal performance can be significantly higher for energy proportional hardware.* This can be seen as significantly higher **equiperformance power savings** for DVFS in Figure 1(c) compared to the energy efficiency improvements seen in (b). Figure 1(d) shows for two different levels of bloat reduction how having energy proportional resources (using DVFS) can yield significantly higher equiperformance energy savings.

Reducing bloat yields the greatest energy saving benefit when the bloated resource is the bottleneck and has a super-linear power vs load characteristic while the other resources in the system are non energy proportional.

We use two different metrics for the power impact of bloat reduction, (i) power at peak achievable performance and (ii) power at equiperformance. Both perspectives are important. Energy efficiency at peak is likely to impact power provisioning in a data center. An improvement in this metric implies that a higher throughput per server is possible within its power budget, potentially reducing the number of servers that must be provisioned. A



(a) Impact of bloat reduction on peak system power

(b) Impact of bloat reduction on energy efficiency score for SPECPower_ssj2008

(c) Impact of bloat reduction on equi-performance system power with DVFS

(d) Equi-performance power savings with bloat reduction at different levels of bloat in SMT4 mode

Figure 1: Experimental results from varying temporary objects bloat in SPECPower_ssj2008 by enabling or disabling object reuse under different configurations on a Power750 system. Bottleneck strain is created by increasing the degree of hardware multi-threading from 2-way (SMT2) to 4-way(SMT4) and reducing the cache size to half (HC); degree of energy proportionality is varied by using a fixed frequency or enabling DVFS.

lower equiperformance power consumption, on the other hand, is likely to provide operational energy savings with workload/power management schemes which maintain the minimum operating point necessary to meet performance goals.

3. RELATED WORK

Mitchell and Sevitsky initiated studies of runtime bloat with their analysis of data transformations [8] in framework based Java applications and data structure health signatures [7]. [10] summarizes the state of the art in research on software bloat analysis and solutions. In our controlled experiments we apply object reuse for de-bloating excess temporaries; the technique has recently been automated [1]. We complement such efforts by undertaking the first study of the actual systems level power-performance impact of bloat reduction under different conditions.

Energy (and power) characterization of the Java runtime and applications have been conducted using real system power measurements by Contreras and Martonosi [3] on mobile platforms and more recently by Esmaeilzadeh et al[4]. However, those studies do not examine the impact of bloat on energy-efficiency. Zhao et al [11] analysed the implications of object allocation on scalability and performance of Java applications but power consumption was not a consideration in their work.

4. CONCLUSIONS

In this paper, we explore the intersection of two areas that have been researched separately so far: software bloat and power consumption. The degree to which bloat impacts power-performance is not always obvious and can vary widely with hardware and software configuration. We develop a simplified analytical framework to explain the impact of bloat on power-performance by relating it to resource pressure caused by bloat. We find that understanding the cause and resource impact of bloat (software perspective) as well the relative energy proportionality of hardware resources and how close they are to being a bottleneck (hardware perspective) are both critical to mitigation of software bloat as an avenue for power-performance optimization.

5. REFERENCES

[1] Bhattacharya, S., Nanda, M. G., Gopinath, K., and Gupta, M. Reuse, recycle to de-bloat software. In *ECOOP'11*.

[2] Bhattacharya, S., Rajamani, K., and Gupta, M. Power performance implications of software runtime bloat: A case study with specpower_ssj2008. Tech. rep., IBM Research, 2011.

[3] Contreras, G., and Martonosi, M. Techniques for real-system characterization of java virtual machine energy and power behavior. *IISWC'06*.

[4] Esmaeilzadeh, H., Blackburn, S., Yang, L., and McKinley, K. S. Power and performance of native and java benchmarks on 130nm to 32nm process technologies. In *MoBS'10*.

[5] Jain, R. Operational laws, ch 33. In *The Art of Computer Systems Performance Analysis, Wiley India Edition*.

[6] Mitchell, N., Schonberg, E., and Sevitsky, G. Four trends leading to java runtime bloat. *IEEE Software* (Jan 2010).

[7] Mitchell, N., and Sevitsky, G. The causes of bloat, the limits of health. In *OOPSLA '07*.

[8] Mitchell, N., Sevitsky, G., and Srinivasan, H. Modeling runtime behaviour in framework based applications. In *ECOOP'06*.

[9] Saxe, E. Power efficient software. *Communications of the ACM 53*, 2 (Feb 2010), 44–48.

[10] Xu, G. e. a. Software bloat analysis: Finding, removing, and preventing performance problems in modern large-scale object-oriented applications. In *FoSER'10*.

[11] Zhao, Y., Shi, J., Zheng, K., Wang, H., Lin, H., and Shao, L. Allocation wall: a limiting factor of java applications on emerging multi-core platforms. In *OOPSLA '09*.

[12] Zichen Xu, Y. T., and Wang, X. Exploring power performance tradeoffs in database systems. In *ICDE'10*.