Modeling and Predicting Power Consumption of High Performance Computing Jobs

Article · December 2014

Source: arXiv

6 authors, including:

Scott Pakin
Los Alamos National Laboratory
113 PUBLICATIONS **2,815** CITATIONS

SEE PROFILE

Michael Lang
Los Alamos National Laboratory
96 PUBLICATIONS **1,369** CITATIONS

SEE PROFILE

Modeling and Predicting Power Consumption of High Performance Computing Jobs

Curtis Storlie[†], Joe Sexton[†], Scott Pakin[†], Michael Lang[†],
Brian Reich[‡], William Rust[†]

[†] Los Alamos National Laboratory
[‡] North Carolina State University

Abstract

Power is becoming an increasingly important concern for large supercomputing centers. Due to cost concerns, data centers are becoming increasingly limited in their ability to enhance their power infrastructure to support increased compute power on the machine-room floor. At Los Alamos National Laboratory it is projected that future-generation supercomputers will be power-limited rather than budget-limited. That is, it will be less costly to acquire a large number of nodes than it will be to upgrade an existing data-center and machine-room power infrastructure to run that large number of nodes at full power. That said, it is often the case that more power infrastructure is allocated to existing supercomputers than these machines typically draw. In the power-limited systems of the future, machines will in principle be capable of drawing more power than they have available. Thus, power capping at the node/job level must be used to ensure the total system power draw remains below the available level. In this paper, we present a statistically grounded framework with which to predict (with uncertainty) how much power a given job will need and use these predictions to provide an optimal node-level power capping strategy. We model the power drawn by a given job (and subsequently by the entire machine) using hierarchical Bayesian modeling with hidden Markov and Dirichlet process models. We then demonstrate how this model can be used inside of a power-management scheme to minimize the affect of power capping on user jobs.

Keywords: High Performance Computing; Power Consumption; Dirichlet Process; Hierarchical Bayesian Modeling; Hidden Markov; Power Capping.

Running title: Modeling and Predicting Power Consumption of High Performance Computers

Corresponding Author: Curtis Storlie, `storlie@lanl.gov`

1 Introduction

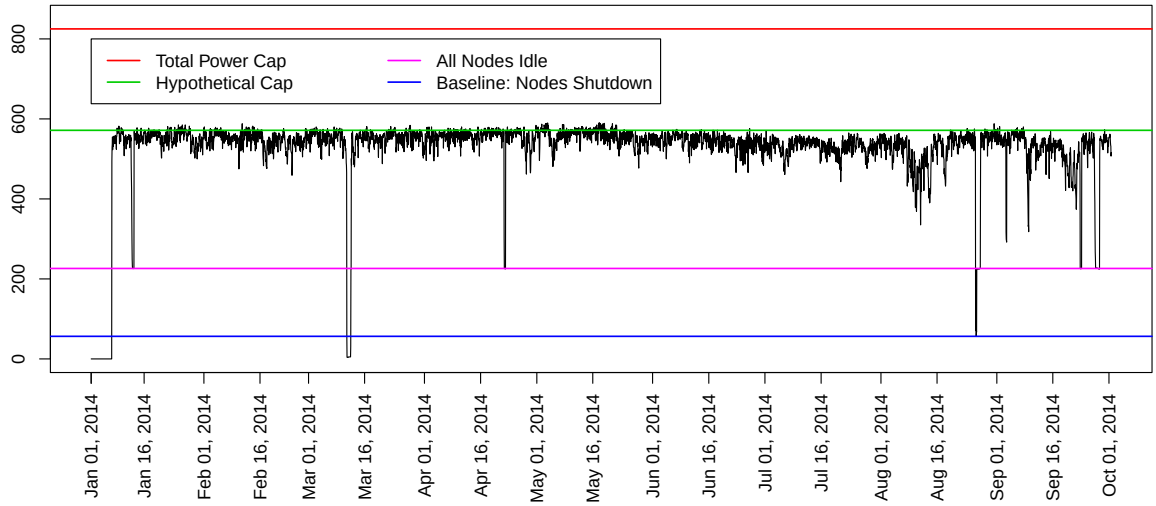
1.1 Power Concerns for Supercomputers

Power is becoming an increasingly important concern for large supercomputing centers (Kamil et al. 2008). Due to cost concerns, data centers are nearing their capacity to

enhance their power infrastructure to support increased compute power on the machine-room floor (Patki et al. 2013, Zhang et al. 2014). At Los Alamos National Laboratory it is projected that future-generation machines will be power-limited rather than budget-limited. That is, it will be less costly to acquire a large number of nodes than it will be to upgrade an existing data-center and machine-room power infrastructure to run that large number of nodes at full power. This is because the cost of power capacity follows a precipitous step function due to the need for construction work on the building and the installation of power substations, chillers, and other large investments. That said, it is often the case that there is a substantial amount of *trapped capacity* in existing supercomputing data centers (Pakin et al. 2013, Zhang et al. 2014). That is, more power infrastructure is allocated to existing supercomputers than these supercomputers typically draw. Trapped capacity is the difference between the infrastructure capacity allocated to a given machine (i.e., supercomputer) and the actual peak demand of that machine. For example, the electrical feeder to a rack of servers is typically sized to be able to feed all of the servers running simultaneously at maximum power draw. However, in normal operation the peak electrical demand of the rack may never exceed even half of the demand that was used to size the feeder. Yet the excess capacity is held in reserve "just in case" the worst-case demand were to occur. This reserve excess capacity is referred to as trapped capacity. Future systems will not have the luxury of this trapped-capacity cushion. In particular, node-level power capping (i.e., throttling performance to limit power consumption) will need to somehow be used to get the maximum performance out of the available power. The goal of this work is to use statistical modeling of job power to provide an optimal power capping strategy according to a given criterion (e.g., maximize throughput, minimize the worst user inconvenience, etc.).

Figure 1, for example, displays the power drawn from the Luna supercomputer at Los Alamos National Laboratory (LANL). It is apparent that the peak allocated power could be substantially reduced, or sufficiently more computing hardware (i.e., compute

Figure 1: Illustration of trapped-capacity for the Luna supercomputer at LANL.



nodes) could be safely added without increasing the power allocated to the machine. The trapped capacity of Luna is typical of most machines at LANL. The reason that there is typically so much trapped capacity for a given machine is that it is difficult to predict what the typical power draw will be for a machine prior to actually running jobs on it. The power infrastructure for a machine must be developed prior to having any of this information, and it is thus designed to accommodate a conservative estimate of a theoretical peak power draw from the machine.

Thus, we use as our example for this paper a hypothetical machine *Sol* with the same number of nodes and architecture as Luna, but with a smaller peak power allotment of 575 kW (i.e., the hypothetical cap provided in Figure 1). This is intended to mimic the power-limited scenario of the future, where the nodes could, in principle, draw more power collectively than is available to the machine. Luna is composed of 1540 compute nodes each with two sockets of 8-core, 2.6 GHz, Intel Xeon E5-2670 processors—a total of 24,640 processors at a combined peak performance of 539.1 TFlop/s. While a machine of the future will not necessarily have the same basic architecture as Luna, Luna was chosen as the template for this example because it had most of the data needed for this analysis readily obtainable. Further, this paper is about a proof of concept using power

data, independent of architecture. Machines of the future, including the newest machine on the horizon at LANL, Trinity, will also have the ability to set a hard cap for the power draw to each node (mainly via CPU throttling) at the expense of performance. Thus, we also assume that the hypothetical Sol machine has this node-level power-capping capability.

Suppose the hard cap for each node i was set to T_i such that $\sum_i T_i + B \leq T$ kW, where B is the *baseline* required to power the machine irregardless of whether the nodes are even powered on, and T is the peak power available. The baseline of $B = 56.5$ kW and $T = 820$ kW for the Luna machine are depicted in Figure 1. We assume the Sol machine has the same baseline, but introduce a more stringent hypothetical power cap of $T = 575$ kW. This is to mimic a power-limited future machine that in theory does not have enough power available to run all of its hardware at full throttle. However, if such a node-level capping constraint above were imposed, then there would not be an issue with going over the the power threshold and possibly tripping breakers, damaging nodes, paying exorbitant utility penalties, etc. This cap can also be adjusted for each node fairly quickly, although it remains unclear at the moment just how quickly the cap could be adjusted for a node on Trinity. Caps could easily be adjusted inside of a minute time frame, however, which coincides with the frequency of the data observations for this study. If more frequent observations are available to inform the caps, then the methodology described here can easily be applied to that time scale. The main question then becomes, what is the best way to choose/alter the hard cap for each of the various nodes as jobs are running on them?

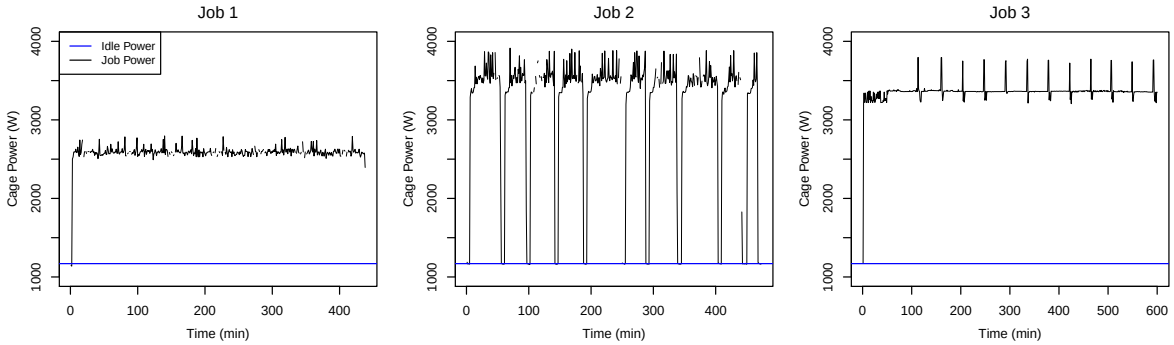
A very simplistic approach would be to let each node's cap be $T_i = (T - B)/N$ where N is the number of nodes in the machine. However, this approach would not allow the nodes that are being worked hard by a compute-intensive job to draw more power than those running closer to idle (e.g., because they are blocked on I/O). Instead, we seek to use the power data being collected for each node to make predictions for a short time

horizon and use these predictions to determine the best capping strategy for that time frame.

To accomplish the goal above, a stochastic process model is developed for the power drawn by the nodes running the same job. This model can then be updated as more data become available for a specific job and then used to predict (i.e., produce many realizations of) the power that may be drawn by each node running that job. These future power realizations for a job can be used to assess the detriment to performance due to a possible hard cap for the nodes running that job. And then, an optimal, machine wide, node capping strategy can be implemented.

In order to illustrate the concept, we assume that the distribution of jobs (and the power that they draw) on the hypothetical Sol machine is identical to that of Luna. Luna is *not* currently instrumented to collect power at the node level, which would be ideal. This is merely an instrumentation/cost issue and node-level power measurement will be available on the new Trinity machine and likely for all future machines. Luna does, however, currently support power monitoring at the *cage* (10-node) level. Therefore, all user jobs that took up an entire cage during the study period were selected for the analysis presented here. Figure 2 provides a few example cases of the cage level power draw over time from three production jobs in the dataset. Because of the restriction that these jobs must encompass a cage, this does not constitute a truly random sample of all jobs seen on Luna. However, most ($> 90\%$) of the nodes being used at a given time on Luna are used by such jobs, and they are typically the more compute-intensive and interesting jobs anyhow. Thus, for the purpose of "proving the concept" of efficient power capping in this work, we make the simplifying assumption that all jobs occurring on Sol come from the same population as jobs spanning a 10-node cage on Luna. Node-level power measurement on new and existing machines is forthcoming in any case, and an identical approach to that described here would apply directly to node-level data when these data become available. Further, this simplifying assumption, if anything, is conservative, since

Figure 2: Cage power over time for three distinct jobs.



larger jobs typically draw more power per node. Even still, as demonstrated in Section 4, the potential benefit of applying the proposed capping approach in practice is substantial.

The total power used by the Sol machine is then equal to the sum of the power draws for the 154 cages, plus the additional baseline level B that includes other more consistent power draws that are not recorded at the cage level (e.g., network switches, etc.). However, the baseline power draw is nearly constant (Pakin et al. 2013). Thus, the crux of this work is then to provide an accurate probabilistic characterization of the power drawn at the node level (or cage level in this case) by a given job.

1.2 Overview of the Statistical Approach

We propose a sophisticated statistical model for the power profile of a high performance computing (HPC) job. To accommodate the complex non-Gaussian features illustrated in Figure 2, we use a nonparametric Bayesian model for each job's time series. A hidden Markov model (HMM) describes the transitions between different regimes (or tasks within the job) and a correlated residual process allows for fluctuation within a task. Our approach builds on the emerging literature on hierarchical Dirichlet process hidden Markov (HDP-HM) models (Beal et al. 2002, Kottas & Taddy 2009, Lennox et al. 2010, Paisley & Carin 2010, Fox et al. 2011). In these flexible models, the number of potential states in the hidden Markov model is infinite, and the transition probabilities between states are modeled using the hierarchical Dirichlet process prior (Teh et al. 2006). Our

model is most similar to the sticky-HDP-HM model of (Fox et al. 2011), who specify a HDP-HM model with added probability on the staying in the same state.

Rather than modeling a single time series, our application requires a joint analysis of multiple jobs. Fox et al. (2014) also analyze multiple time series using a beta process to share information across series. In our approach, each job is permitted to have different operating characteristics defined by job-specific parameters (e.g., mean time between state transitions and mean value in each state) modeled as draws from a parent distribution. This approach facilitates borrowing of strength across jobs to improve prediction for short series, yet allows flexibility to capture complex features as data accrues.

We then demonstrate how this model can be used inside of a power-management scheme to minimize the affect of power capping on user jobs. Such a scheme will be essential for HPC machines in the power-limited future. While sophisticated statistical models have been applied to the reliability of HPC machines (Storlie et al. 2013, Michalak et al. 2012), to the best of our knowledge this is the first effort to statistically model the power process of scientific computing jobs.

The rest of the paper is laid out as follows. Section 2 describes the relationship between job performance (i.e., execution time) and available power. Section 3 describes the proposed HDP-HM model for a job power process along with the estimation and prediction procedures. The job power model and updating procedure are then used to demonstrate an optimal node-level power capping strategy in Section 4. Section 5 concludes the paper. This paper also has online supplementary material containing Markov chain Monte Carlo (MCMC) estimation details.

2 The Effect of Power Capping on Job Performance

Before diving into a statistical model for the power profile of a job, it is first important to understand how a decrease in available power will affect job performance. Once the relationship between job performance (i.e., how long the job takes) and power is

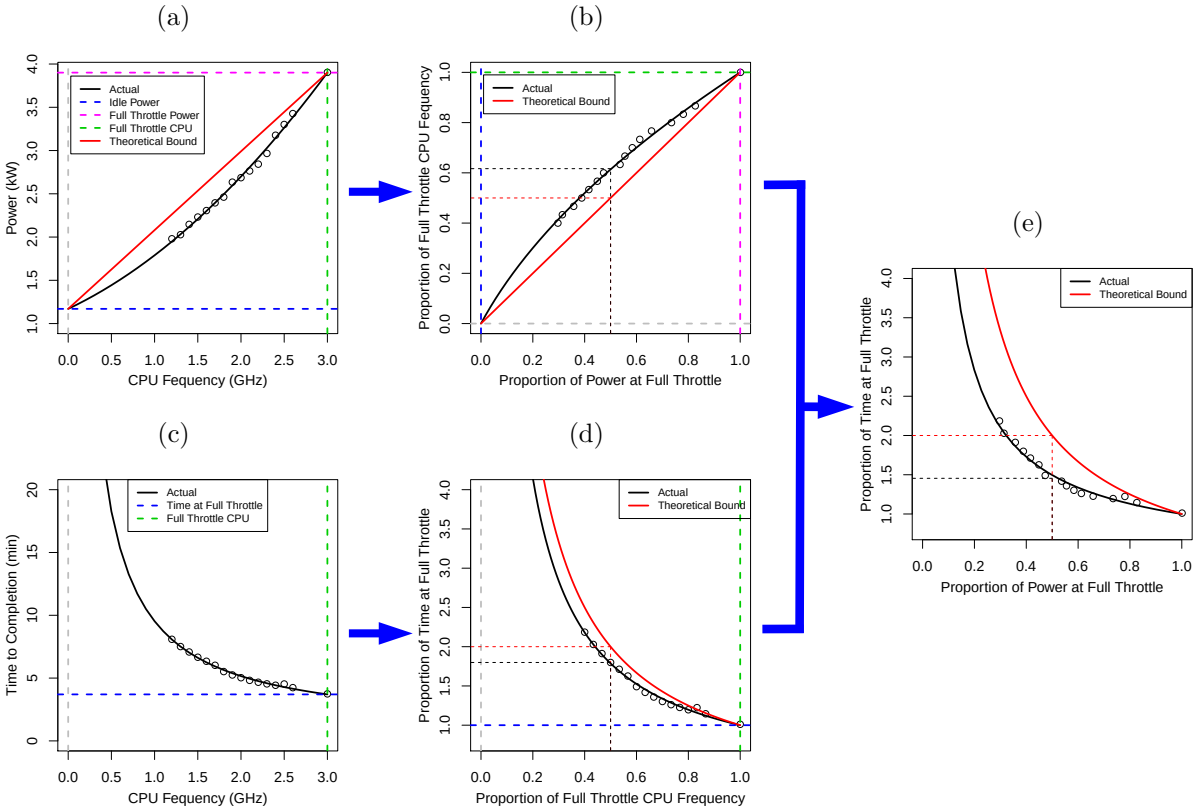
understood, the statistical model for power can then be applied to predict the possible degradation to job performance due to a given power cap.

The effect of power reduction via CPU throttling and its effect on performance has been previously investigated (Hsu & Feng 2005, Freeh et al. 2007, Ge & Cameron 2007, Pakin & Lang 2013). It is known and was demonstrated in Pakin & Lang (2013), on several benchmark programs, that power scales like CPU frequency squared. It is beyond the scope of this paper to develop a precise relationship between performance and power capping. Instead the simple logic portrayed in Figure 3 can be used to develop an upper bound on the performance degradation due to power capping.

Figure 3(a) displays Power by CPU frequency for the *POP* program, one of the benchmarks used in Pakin & Lang (2013). The POP program, along with the other programs used in that study, runs at a relatively constant power since it is essentially performing a single task, i.e., POP solves a pre-specified elliptic equation using a preconditioned conjugate gradient (PCG) solver (Dukowicz & Smith 1994). It is unlike a production job in that sense, since production jobs will cycle between different tasks and write output to disk at checkpoints, etc. However, POP would mimic the behavior exhibited by a production job during one of these homogeneous tasks.

Figure 3(a) provides an upper bound on the power draw as a function of CPU frequency. The derivation of this bound leverages the fact that power for a node (above and beyond idle power) scales like CPU frequency squared. Also, the idle power draw of a node (or cage in this case) is relatively constant and easy to measure. Thus, an upper bound on power draw is a straight line from idle draw I at CPU frequency $\chi = 0$, to the power needed, $P_{\max}$, when CPU frequency is at full throttle, i.e., $\chi = 3.0$ GHz in this case. Figure 3(b) shows the inverse of the relationship of that in Figure 3(a) on a proportional scale. That is, the proportion of full throttle CPU is given as a function of proportion of full throttle CPU power (i.e., above and beyond idle power), for both the POP program and the bound (which is now a lower bound for CPU frequency as a

Figure 3: Derivation of an upper bound on performance degradation: (a) Power by CPU frequency for the *POP* program. (b) Proportion of full throttle CPU (3.0 GHz) by Proportion of full throttle CPU power (i.e., above and beyond idle power). (c) Time to completion by CPU frequency for the POP program. (d) Proportion of time a full throttle CPU that is required to complete the job by proportion of full throttle CPU. (e) Relationships depicted in (c) and (d) are combined to produce the relation for the proportion of time to completion (if the full power needed were available) to the proportion of full power that is available. The resulting upper bound is simply an inverse relationship (i.e., reduce power by 50% of what the uncapped program would use for a task and the program would take 2× as long to complete that task).



function of power). Thus, if power is to be reduced by 50%, then CPU frequency could still be allowed to be (at least) 50% of full throttle. For the POP program in particular, CPU frequency could be ∼60% to obtain a 50% power reduction.

Figure 3(c) shows the time to completion by CPU frequency for the POP program. Figure 3(d) provides the proportional multiplier of the time needed to complete when at full throttle CPU as a function of CPU frequency (relative to full throttle CPU). For a completely compute bound program (i.e., simply executing instructions while all memory

is in local cache), the proportional time to completion would scale like the inverse of the CPU frequency (e.g., if the CPU frequency is reduced by 50% then the program would take 2× as long to complete). This is the upper bound illustrated by the red curve; most production jobs will not be 100% compute bound at any given time. The POP program is close to being 100% compute bound, but if CPU frequency is reduced by 50% it would take POP $\sim 1.8\times$ (i.e., less than the upper bound of 2×) as long to complete.

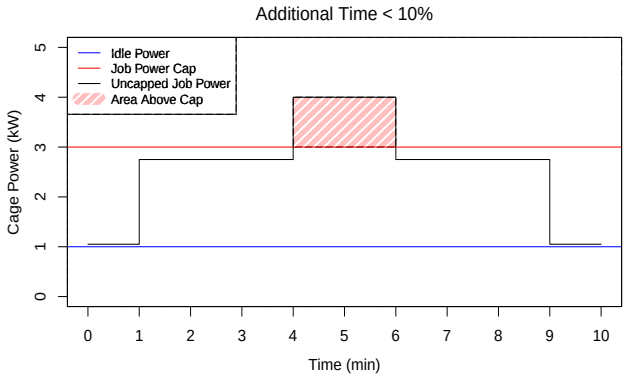
Finally, Figure 3(e) combines the relationships depicted in Figures 3(b) and 3(d) to produce the relation for the proportional multiplier of time to completion as a function of the proportion of the power (needed at full CPU throttle) that is available. The resulting upper bound is simply an inverse relationship, i.e., reduce power by 50% of what the uncapped program would use for a task and the program would take at most 2× as long to complete that task. Specifically, for the POP program, if the power were reduced by 50% of the full throttle power, then only $\sim 1.5\times$ more time would actually be needed to complete the task.

There are other possible means to limit power to a node beyond CPU throttling. However, the degradation to job performance would necessarily come via a CPU performance reduction. Thus, the bound on performance degradation provided here, i.e., assuming all power savings come at the cost of a CPU throttle down, would remain valid. Certainly, there would be some benefit to further exploring the performance by power relationship for the population of jobs running on a given cluster. However, that is beyond the scope of this paper, and the bound on performance degradation used here is not all that conservative as seen in Figure 3.

The upper bound relationship displayed in Figure 3(e) can be extended to provide a bound on performance degradation for a more heterogeneous program that does not run at constant power. Consider the program depicted in Figure 4, where it is known what the unrestricted/uncapped power draw would be over time. If a power cap were hypothetically introduced at 3 kW, in this case, the two minutes of computation required

to complete the task between 4 and 6 minutes in Figure 4 would be increased. The program wants 4 kW during that portion of the computation, but it is restricted to 3 kW, while idle draw is 1 kW, i.e., it is allowed only 2/3 of the power (above idle) that it needs during that time. Thus, according to the bound displayed in Figure 3(e) and discussed above, the program would require at most $1/(2/3) = 1.5\times$ as long to complete the task. In other words, that two minute task (at full power) would now require at most 3 minutes. The rest of the 8 minutes of the program's execution remains unaffected by the 3 kW cap. Thus, the program would take at most $8 + 3 = 11$ minutes to execute instead of 10 minutes, a 10% increase in compute time for the entire program execution.

Figure 4: Illustration of the Upper Bound on Performance Degradation.

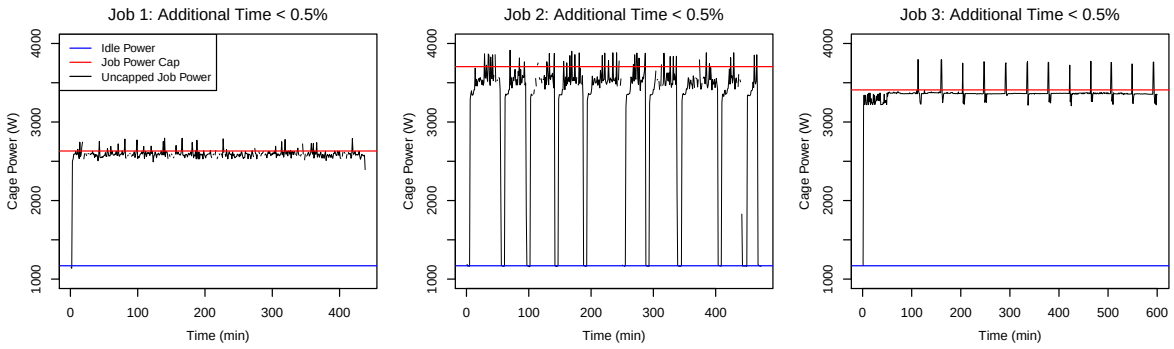


In general, suppose the unrestricted power profile over time t, for a given program, is known to be $P(t)$ and idle power is I. Using the same logic as above with a quadrature argument, the increase in time Δ resulting from a power cap of C would be

$$\Delta \leq \frac{\int |P(t) - C|_+ \, dt}{C - I}, \tag{1}$$

where $|x|_+ = x$ if $x > 0$ and zero otherwise. This relationship is used in Figure 5 to find the power cap that would produce at most a 0.5% increase in computation time for the three job examples from Figure 2. The cap leading to this 0.5% increase is much lower for Job 1 than for the other two jobs. The fact that this cap could be a lot lower for some jobs than for others is really the driving force of this paper. We would like to set the same cap for a group of nodes runnning the same job, in a manner such that no job receives a major performance degradation. In reality, the unrestricted power that a job will draw will not be known ahead of time, making it impossible to provide an optimal

Figure 5: Cage power over time for three distinct jobs with a power cap leading to an upper bound of 0.5% additional time to completion.



capping strategy for the current job mix on a machine directly as above. However, we next propose a statistical model for job power draw and use this model to provide an approximate predictive distribution of the future power profile for each job running on the machine. From this probabilistic description of the $P(t)$ for each job, one can devise an informed capping strategy to optimize a given criterion.

3 Hierarchical Bayesian Model for Job Power

As discussed previously, a hierarchical model is assumed for the node-level power draw for a job. The model for an individual job is described in Section 3.1, and then the parent model governing the parameters of the individual job model is introduced in Section 3.2. Estimation of the parent model parameters and then the individual job parameters (i.e., updating) is discussed, both from a fully Bayesian and a simple, pragmatic perspective.

3.1 Statistical Model for an Individual Job

The model for the power drawn for a given job is a hidden Markov model that allows a job to switch between various regimes (or tasks, e.g., see Figure 2) and draw a different amount of power on average while performing each task. Specifically, the power draw observation the j^{th} job, $j = 1, \ldots, J$ at time t is represented as

$$x_j(t) = \sum_{k=1}^{\infty} \mu_{j,k} I_{\{\xi_j(t)=k\}} + z_j(t) + \varepsilon_j(t), \tag{2}$$

where (i) $\xi_j(t)$ is an indicator for a particular regime (i.e., task that is being performed) during the job, (ii) $\mu_{j,k}$ is the mean level power draw while job j is in the k^{th} regime, (iii) $z_j(t)$ is a stationary, mean zero, time dependent process that allows the power to fluctuate around the current mean level, and (iv) $\varepsilon_j(t) \overset{iid}{\sim} N(0, \tilde{\tau}^2)$ is a white noise measurement error, with common variance for all jobs.

For simplicity in the estimation procedure, we assume that t is discrete, since the power observations are recorded at regular one-minute intervals, $t = 1, 2, \ldots, T_j$. However, a continuous time analog is implied by the following discrete time model if, for example, predictions on a finer grid than every minute were desired.

From exploratory analysis of a homogeneous task (i.e., the POP program and similar), it was deemed appropriate to model $z_j(t)$ as an Ornstein-Uhlenbeck (O-U) process, i.e.,

$$z_j \sim GP\left(0, \sigma_j^2 \Gamma_{\rho_j}\right), \tag{3}$$

where $\Gamma_\rho(s,t) = \exp\{-\rho|s-t|\}$. This is equivalent to a first order auto-regressive model in regularly-spaced discrete time.

Finally, the regime indicator process $\xi_j(t)$ is assumed to be a Markov chain where the residence time in regime $\xi_j(t) = k$ is

$$T \sim \text{Geometric}(\lambda_{j,k}(1 - \pi_{j,k})). \tag{4}$$

The parametrization of the geometric rate with $\lambda_{j,k}(1-\pi_{j,k})$ above may seem redundant and unidentifiable at first glance. However, the characterization of state transition probabilities below identifies these parameters. The reason for the parametrization in (4) is that it allows for conjugate updates of the $\pi_{j,k}$ (see the Supplementary Material), while allowing for essentially the same model as if only $\lambda_{j,k}$ were used for the geometric rate.

Suppose that a transition out of state k occurs at time u, then it is assumed that

$$\Pr(\xi_j(u) = l \mid \xi_j(u-1) = k) \propto \begin{cases} \pi_{j,l} & \text{for } l \neq k \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

That is, when a transition occurs, a *new* regime is chosen with probability proportional to $\boldsymbol{\pi}_j = [\pi_{j,1}, \pi_{j,2}, \dots]'$, regardless of the state k from which the transition is being made.

The model for ξ_j above is a discrete time Markov chain (MC), but was intentionally parameterized analogously to a continuous time MC, (i.e., geometric in place of exponential residence time, then it moves on to a new regime). In (5), although the transition rates are assumed independent of the previous state, the residence times in each state are allowed different $\lambda_{j,k}$. Thus, this parametrization results in a transition probability matrix (TPM) for the discrete time MC with unequal rows in general. In particular, the diagonal of the TPM will generally be inflated, i.e., the $(k,l)^{\text{th}}$ element of the TPM is,

$$P_{k,l} = \Pr\left(\xi_j(t+1) = l \mid \xi_j(t) = k\right) = \begin{cases} \lambda_{j,k}\pi_{j,l} & \text{for } l \neq k, \\ \lambda_{j,k}\pi_{j,k} + (1 - \lambda_{j,k}) & \text{for } l = k. \end{cases} \tag{6}$$

The transition model in (6) is related to the sticky HDP-HMM of Fox et al. (2011). Fox et al. (2011) allow for a completely different set of transition probabilities from each current state (they consider a single chain, so we drop the dependence on j), i.e., $\text{Prob}(\xi(t) = l | \xi(t-1) = k) = \pi_{kl}$. The transition probabilities for each state, $\pi_k = (\pi_{k1}, \pi_{k2}, ...)$, are each modeled with a DP with inflated prior mass on the diagonals π_{kk} to encourage the process to stay in the current state. They use a hierarchical DP to pool information across states to estimate the transition probabilities.

The rationale for the proposed model is that the distribution of the residence time in a given regime can vary greatly between the regimes within the same job (refer back to Figure 2, for example). However, upon *leaving* a regime, it was not immediately clear from the data that the next regime was dependent on the previous regime. Most programs have only a few transitions from each of their *observed* regimes, which would make estimation of completely separate transition probabilities difficult for practical purposes. For a real job, the regime switching process is not even a MC at all, as it is largely deterministic (i.e., which task is computed next, when to write out a check point to disk,

etc.). However, the logic and hardware involved are complicated enough to make it look somewhat random. In any case, the proposed model adequately represents the regime changes for our purposes, see Section 3.5.

A subtle point is that there are 454 observations in the data set, but only 213 unique jobs. That, is, some jobs span multiple cages. In these cases, the jobs are assumed to share common parameters $\mu_{j,k}$, σ_j^2, ρ_j, $\lambda_{j,k}$, and $\pi_{j,k}$, but the values of the random processes ξ_j, z_j, and ε_j are treated as independent across replicates. Initial inspection of several replicate job observations, implied that the regimes were not quite in lock step with one another over time so that allowing for separate ξ_j was necessary. Still, it could be beneficial for estimation purposes to model the offset of the ξ_j for replicate jobs, i.e., introduce a dependence between their respective ξ_j. For the purpose of prediction of the entire machine in Section 4, it will be assumed that jobs spanning multiple cages (or ultimately nodes) do have identical ξ_j processes, which would produce a conservative prediction of the aggregate power drawn by such a job, e.g., all nodes running the same job would be assumed to be in the most power intensive regime at the same time.

3.2 Parent Model for the Job Parameters

There are several parameters in the model for a given job in Section 3.1, e.g., $\mu_{j,k}$, $\lambda_{j,k}$, σ_j, etc. Further, we wish to be able to make predictions, even for newly started jobs with very little or no data. A typical approach in such cases is to assume that job specific parameters come from a parent distribution. In the Bayesian paradigm, the parent parameters governing the parent distribution have prior distributions placed on them, and their posterior distribution is approximated via MCMC. Here we describe the details of the parent model, and defer the computational details to Section 3.3.

Parent Model for z_j Parameters. In the model for the power fluctuations z_j in (3), there are two job specific parameters, σ_j^2 and ρ_j. We assume these parameters for the j^{th} job are realized from a parent distribution as follows,

$$\sigma_j^2 \overset{iid}{\sim} \log N(\tilde{\mu}_\sigma, \tilde{\sigma}_\sigma^2),\ j = 1, \ldots, J,$$
$$\rho_j \overset{iid}{\sim} \log N(\tilde{\mu}_\rho, \tilde{\sigma}_\rho^2),\ j = 1, \ldots, J,$$

where $\log N$ is the log-Normal distribution. Here and throughout the paper, any parameters that are parent parameters receive a tilde above them in their notation to add clarity. A log-Normal distribution was chosen for σ_j^2 as opposed the popular conjugate choice of Inverse-Gamma, due to the fact that the Inverse-Gamma would have far too heavy of a tail to adequately represent the parent distribution of σ_j^2 variation among jobs. Power predictions of a *brand-new* job, for example, would be allowed to be significantly higher than realistic limits if using an Inverse-Gamma model for σ_j^2.

To make the model specification complete, a prior distribution is placed on the parent parameters $\tilde{\mu}_\sigma$, $\tilde{\sigma}_\sigma^2$, $\tilde{\mu}_\rho$, and $\tilde{\sigma}_\rho^2$. These parameters are assumed to be distributed as,

$$\tilde{\mu}_\sigma \sim N(M_\sigma, S_\sigma^2),\ \ \tilde{\sigma}_\sigma^2 \sim \text{IG}(A_\sigma, B_\sigma),\ \ \tilde{\mu}_\rho \sim N(M_\rho, S_\rho^2),\ \ \tilde{\sigma}_\rho^2 \sim \text{IG}(A_\rho, B_\rho), \tag{7}$$

where IG is the Inverse-Gamma distribution. Hyper-prior parameters (e.g., M_σ, S_σ^2, etc.) for all parent parameter prior distributions defined in (7) and below are always denoted by capital letters and a corresponding subscript. Values must be set for all such parameters in order to complete the model specification. For convenience, all of these parameters will be reviewed and specified for this application at the end of this section.

Parent Model for $\mu_{j,k}$. The mean level $\mu_{j,k}$ for the k^{th} regime of job j in (2) is assumed to come from a (possibly infinite) normal mixture model, i.e.,

$$\mu_{j,k} \overset{iid}{\sim} \sum_{m=1}^{\infty} \tilde{\omega}_m N(\tilde{\nu}_m, \tilde{\varsigma}_m^2),\ k = 1, 2, \ldots, \text{ and } j = 1, \ldots, J. \tag{8}$$

where $\sum \tilde{\omega}_m = 1$. The normal mixture model in (8) is assumed to be a Dirichlet process (Ferguson 1973, Ishwaran & James 2001, Lid Hjort et al. 2010). That is, the mixture probabilities follow a stick-breaking distribution (Sethuraman 1994), $\tilde{\boldsymbol{\omega}} = [\tilde{\omega}_1, \tilde{\omega}_2, \ldots]' \sim \text{SB}(\tilde{\gamma})$, or

$$\tilde{\omega}_m = u_m \prod_{n=1}^{m-1}(1-u_n) \tag{9}$$

where $u_m \overset{iid}{\sim} \text{Beta}(1, \tilde{\gamma})$, $m = 1, 2, \ldots$. A further hyper-prior is typically assumed on $\tilde{\gamma}$, i.e.,

$$\tilde{\gamma} \sim \text{Gamma}(A_\gamma, B_\gamma). \tag{10}$$

The remaining parent parameters for $\mu_{j,k}$ distribution are assumed to have the following prior distributions,

$$\begin{aligned} \tilde{\nu}_m &\overset{iid}{\sim} N(M_\nu, S_\nu^2),\ m = 1, 2, \ldots, \\ \tilde{\varsigma}_m^2 &\overset{iid}{\sim} \text{IG}(A_\varsigma, B_\varsigma),\ m = 1, 2, \ldots \end{aligned} \tag{11}$$

Parent Model for ξ_j ***Parameters***. The ξ_j process is governed by the parameters $\lambda_{j,k}$ and $\pi_{j,k}$, i.e., the regime transition rate and transition probabilities from (4) and (5), respectively. The $\lambda_{j,k}$ in the parametrization of the regime transition rates for each job are assume to come from the parent model,

$$\lambda_{j,k} \overset{iid}{\sim} \text{Beta}(\tilde{\alpha}_\lambda, \tilde{\beta}_\lambda),\ k = 1, 2, \ldots, \text{ and } j = 1, \ldots, J.$$

Similar to the model for $\tilde{\omega}_m$ in (9), it is assumed that that the transition probabilities for ξ_j in (5) come from a stick-breaking distribution. That is, $\boldsymbol{\pi}_j = [\pi_{j,1}, \pi_{j,2}, \ldots]' \sim \text{SB}(\tilde{\delta})$, or

$$\pi_{j,k} = v_{j,k} \prod_{l=1}^{k-1}(1-v_{j,l}) \tag{12}$$

where $v_{j,k} \overset{iid}{\sim} \text{Beta}(1, \tilde{\delta})$, $k = 1, 2, \ldots,$ and $j = 1, \ldots, J$.

The parent parameters $\tilde{\alpha}_\lambda$, $\tilde{\beta}_\lambda$, and $\tilde{\delta}$ have the following prior distribution,

$$\tilde{\alpha}_\lambda \sim \text{Gamma}(A_\lambda, B_\lambda),\ \ \tilde{\beta}_\lambda \sim \text{Gamma}(C_\lambda, D_\lambda),\ \ \tilde{\delta} \sim \text{Gamma}(A_\delta, B_\delta). \tag{13}$$

Prior Distribution for $\tilde{\tau}^2$. Lastly, the measurement error variance $\tilde{\tau}^2$ of the model in (2) is common to each job and is assumed to have prior distribution,

$$\tilde{\tau}^2 \sim \text{IG}(A_\tau, B_\tau) \tag{14}$$

Summary of the Hierarchical Model Parameters and Prior Specification. Table 1 summarizes the hierarchical model and provides the values used in the parent parameter prior specification. Relatively diffuse priors were used for most parameters with a few exceptions, e.g., the regime location distribution, the measurement error variance, and the O-U process parameters. Priors for these values were formulated based on data from the performance study of Pakin & Lang (2013).

Table 1: Summary of hierarchical model and the specification of the parent prior distributions.

Description	Job Parameter Model	Parent Prior	Specification
Variance of the O-U Process z_j	$\sigma_j^2 \overset{iid}{\sim} \log N(\tilde{\mu}_\sigma, \tilde{\sigma}_\sigma^2)$	$\tilde{\mu}_\sigma \sim N(M_\sigma, S_\sigma^2)$	$M_\sigma = 4$ $S_\sigma^2 = 1$
		$\tilde{\sigma}_\sigma^2 \sim \text{IG}(A_\sigma, B_\sigma)$	$A_\sigma = 10$ $B_\sigma = 5$
Range of the O-U Process z_j	$\rho_j \overset{iid}{\sim} \log N(\tilde{\mu}_\rho, \tilde{\sigma}_\rho^2)$	$\tilde{\mu}_\rho \sim N(M_\rho, S_\rho^2)$	$M_\rho = -2$ $S_\rho^2 = 9$
		$\tilde{\sigma}_\rho^2 \sim \text{IG}(A_\sigma, B_\sigma)$	$A_\sigma = 10$ $B_\sigma = 5$
Location of the k^{th} regime	$\mu_{j,k} \overset{iid}{\sim} \sum \tilde{\omega}_m N(\tilde{\nu}_m, \tilde{\varsigma}_m^2)$	$\tilde{\omega}_m \sim \text{SB}(\tilde{\gamma})$, $\tilde{\gamma} \sim \Gamma(A_\gamma, B_\gamma)$	$A_\gamma = 1$ $B_\gamma = 1$
		$\tilde{\nu}_m \overset{iid}{\sim} N(M_\nu, S_\nu^2)$	$M_\nu = 2000$ $S_\nu^2 = 10^6$
		$\tilde{\varsigma}_m^2 \overset{iid}{\sim} \text{IG}(A_\varsigma, B_\varsigma)$	$A_\varsigma = 1$ $B_\varsigma = 1$
Transition rate for k^{th} regime	$\lambda_{j,k} \overset{iid}{\sim} \text{Beta}(\tilde{\alpha}_\lambda, \tilde{\beta}_\lambda)$	$\tilde{\alpha}_\lambda \sim \Gamma(A_\lambda, B_\lambda)$	$A_\lambda = 1$ $B_\lambda = 1$
		$\tilde{\beta}_\lambda \sim \Gamma(C_\lambda, D_\lambda)$	$C_\lambda = 1$ $D_\lambda = 1$
Regime transition probabilities	$\boldsymbol{\pi}_j \overset{iid}{\sim} \text{SB}(\tilde{\delta})$	$\tilde{\delta} \sim \Gamma(A_\delta, B_\delta)$	$A_\delta = 1$ $B_\delta = 1$
Observation Error Variance	–	$\tilde{\tau}^2 \sim \text{IG}(A_\tau, B_\tau)$	$A_\tau = 10$ $B_\tau = 10$

3.3 Estimation of Model Parameters

The complete details of the MCMC algorithm, including full conditional distributions, etc., are provided in the Supplementary Material. However, an overview is provided here

to illustrate the main idea. The MCMC routine is a typical hybrid Gibbs, Metropolis Hastings (MH) sampling scheme (see Givens & Hoeting (2000), for example). Each MCMC iteration consists of the following two steps:

(i) Update job-specific parameters for each job, conditional on the parent parameters.

(ii) Update parent parameters, conditional on the job-specific parameters from (i).

This process is then repeated until the chain reaches steady state. Conditional on the parent parameters, the parameters for each job are independent across job, making the many (213 in this case) job specific updates easily parallelizable.

The job specific parameters sampled in the MCMC are

$$\Theta_j = \left\{ \{\xi_j(t)\}_{t=1}^{T_j}, \{\lambda_{j,k}\}_{k=1}^{K}, \{\pi_{j,k}\}_{k=1}^{K}, \{\mu_{j,k}\}_{k=1}^{K}, \{z_j(t)\}_{t=1}^{T_j}, \sigma_j^2, \rho_j \right\}, \; j = 1, \ldots, J.$$

For convenience of computation, the number of components in stick-breaking model for $\pi_{j,k}$ was capped at a finite value K, i.e., $k = 1, \ldots, K$. The value of $\pi_{j,K}$ was observed and K was increased until $\pi_{j,K}$ values were negligible for all jobs at $K = 10$. Because of the discrete representation of the job power process, the job specific parameters have relatively simple conjugate updates (details provided in the Supplementary Material). Two exceptions are σ_j^2 and ρ_j, which require MH updates. However, the proposal for σ_j^2 is provided by matching the moments of the log-normal prior to an inverse-Gamma, and producing the corresponding conjugate update. This proposal is then accepted or rejected in the usual MH fashion. This approach resulted in $> 80\%$ acceptance for all j along with the benefit that it requires no tuning. The ρ_j were updated via a random walk proposals. However, the random walk was conducted on the log scale, i.e., $\log(\rho_j^*) = \log(\rho_j + \epsilon)$ for a deviate $\epsilon \sim N(0, s^2)$. With the use of the log scale, a constant tuning parameter $s^2 = 0.25$ could be used for all jobs to achieve acceptances in the range of (30% - 55%).

The parent parameters sampled in the MCMC are

$$\Theta^P = \left\{ \tilde{\mu}_\sigma, \tilde{\sigma}_\sigma, \tilde{\mu}_\rho, \tilde{\sigma}_\rho, \{\tilde{\omega}_m\}_{m=1}^{M}, \{\tilde{\nu}_m\}_{m=1}^{M}, \{\tilde{\varsigma}_m^2\}_{m=1}^{M}, \tilde{\alpha}_\lambda, \tilde{\beta}_\lambda, \tilde{\delta}, \tilde{\gamma}, \tilde{\tau}^2 \right\}.$$

Most parent parameters have conjugate updates as well, with only α_λ and β_λ requiring MH updates. Proposals for α_λ and β_λ are obtained via a random walk with normal deviates on the log scale as described for the updates of ρ_j above. The proposal standard deviations were tuned to produce $\sim 40\%$ acceptance in both cases.

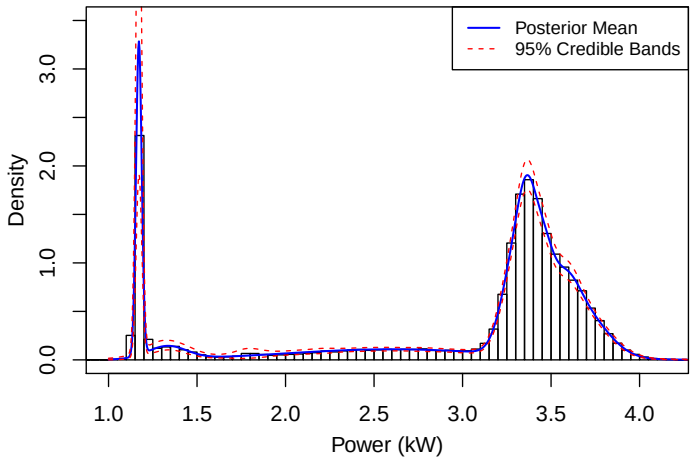
Five different MCMC chains with varying starting points were run out to 10,000 iterations. Based on trace plots of the parent parameters, all chains converged to the approximately the same posterior distribution (aside from label switching of the m index for $\tilde{\omega}_m$, $\tilde{\nu}_m$, and $\tilde{\varsigma}^2_m$) after about 2,000 iterations.

3.4 Updating a Given Job

The main goal of this work (i.e., intelligent node-level power capping) hinges on the ability to make predictions about the future power profile for a job, possibly given some previous measurements from that job. To facilitate this goal, we leverage the fact that the uncertainty in the parent parameters is negligible relative to the uncertainty present in estimating the job parameters for a given job. This should be intuitively clear since all jobs in the training set contribute all of their data toward estimation of ~ 30 parent parameters. And of these parameters, several of the $\tilde{\omega}_m$, $\tilde{\nu}_m$, $\tilde{\varsigma}^2_m$ for larger m values do not matter that much as they have little influence on the overall parent density for the regime level $\mu_{j,k}$ in (8). A posterior summary of the density for $\mu_{j,k}$ is provided in Figure 6. The solid curve is the posterior mean value of the density, while the dashed lines provide 95% (pointwise) credible bands. For reference a histogram is also drawn based on the sampled values of $\mu_{j,k}$ in the posterior.

The tight credible bands around the posterior mean in Figure 6 serve to illustrate the point above about negligible uncertainty in the parent parameters. Thus, when updating the parameters of a specific job for prediction purposes, an empirical Bayes approach is taken where the uncertainty in the parent parameters is ignored, i.e., their values are fixed at their posterior mean. Label-switching issues with the parameters of

Figure 6: Parent normal mixture distribution for the location parameters $\mu_{j,k}$ of the regimes in an individual job process.



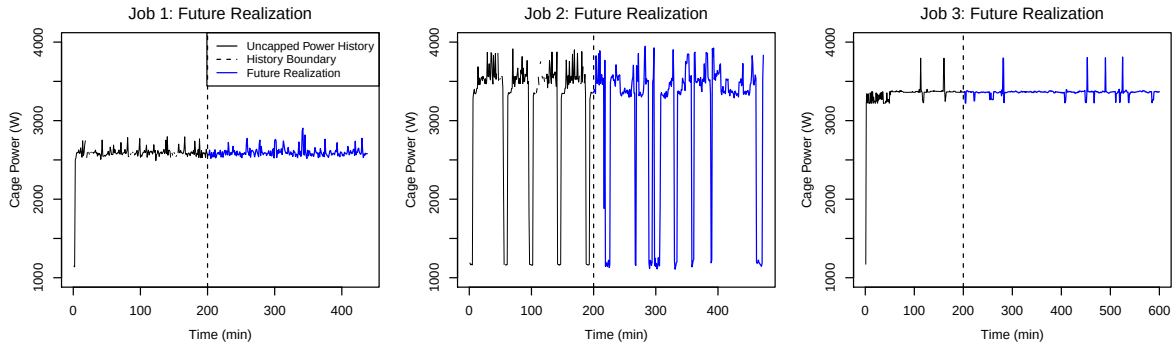
the normal mixture for $\mu_{j,k}$ would render their posterior mean unusable, however, the resulting overall density for $\mu_{j,k}$ is immune to label switching. Therefore the posterior mean density is evaluated on a fine grid as in Figure 6, then approximated with a best fitting normal mixture of 10 components to provide the fixed value of the parent normal mixture parameters $\{\tilde{\omega}_m, \tilde{\nu}_m, \tilde{\varsigma}^2_m\}_{m=1}^{10}$. The remaining parent parameters were fixed at their posterior mean values.

Once the parent parameters are fixed, a posterior distribution for the job parameters for a job given its previous power observations can be sampled by simply iterating step (i) of the MCMC algorithm in Section 3.3. One additional caveat is that in operation, the nodes will have a power cap, making some of the observations right-censored. However, this can easily be handled by simply sampling such observations conditional on the other parameters (and conditional on being greater than the cap) in the MCMC iterations. In this way, the rest of the algorithm remains unchanged, as if no censoring occurred.

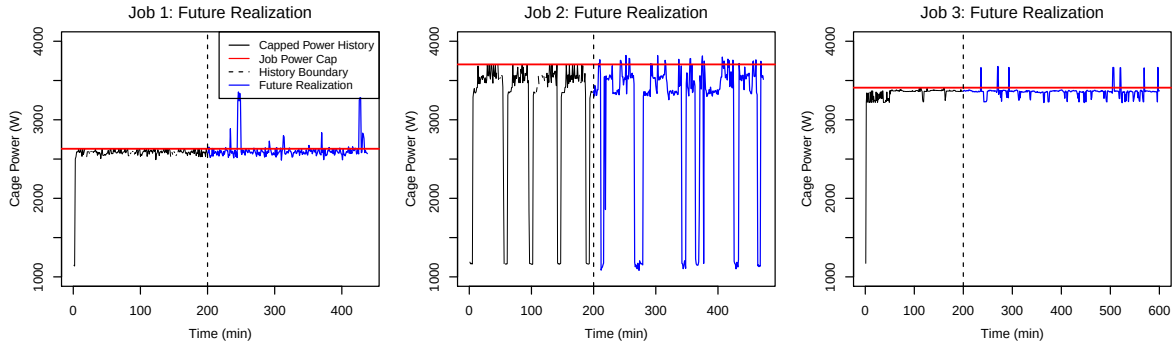
In practice, the many ($\sim$ 100 for Luna) jobs running at a given time on a machine will need to be updated simultaneously. However, this is once again easily parallelizable. Because of the simplicity of the updates for each of the job parameters, the MCMC routine for a single job is very fast. For example, to update a job that has been observed

Figure 7: Example realizations of the future after updating the 3 example jobs from Figure 2 (given 200 minutes of history). (a) Updated using uncapped data history. (b) Updated using capped (i.e., right censored) at the 95$^{\text{th}}$ percentile of the historical data.

(a) Future realization after updating with no power cap



(b) Future realization after updating with displayed power cap



for 200 minutes requires $\sim$ 1 minute for 10,000 MCMC iterations. Convergence in most cases happens very quickly as well (within the first 1,000 iterations). Since the intention is to make node-level power capping changes on the order of minutes, this approach is readily applicable in practice.

Figure 7 displays example realizations of the future power after updating the three example jobs from Figure 2 using 200 minutes of history. The realizations for the three jobs in Figure 7(a) are the result of using uncapped (i.e., uncensored) data to perform the updates. In contrast, the job updates for the realizations in Figure 7(b) were performed by artificially introducing a power cap and censoring the historical data at its 95$^{\text{th}}$ percentile (i.e., the horizontal line in Figure 7(b)). Thus, it is unclear to the estimation procedure how large the power draw could be when it hits this threshold, and it must borrow

strength from the parent model to make its predictions.

Although the job updates via MCMC are fast, the real-time job updates could possibly benefit from a sequential Monte Carlo (SMC) (a.k.a particle filtering) approach (Liu & Chen 1998, Pitt & Shephard 1999, Doucet et al. 2001, Del Moral et al. 2006). However, there are many fixed (over time) parameters in the job power model, e.g., $\pi_{j,k}, \mu_{j,k}$. The fact that these parameters are fixed plays a critical role in the model because a job typically reverts back to its previous regimes. SMC methods can be very challenging to apply in the presence of fixed parameters (Liu & West 2001). Some recent advances have been made in this area however (Storvik 2002, Andrieu et al. 2005, Polson et al. 2008). It is a subject of further work to explore an SMC alternative for job updates prior to full implementation of the proposed power capping approach.

3.5 Assessing Prediction Accuracy

The accuracy of the proposed method for the prediction of the performance degradation is assessed in Figures 8 and 9. Figure 8 displays degradation bound predictions (mean and 95% prediction bands) along with the actual degradation bound as a function of the node power cap for the three jobs displayed in Figures 2 and 7. The model was updated after 200 minutes of history in each case, and then predictions of the performance degradation for the next five minutes across a range of power caps were obtained. In principle, a job will run a little slower when a power cap is introduced, but the procedure used here implicitly assumes that a job was running at its uncapped (max) efficiency in its previous history. This phenomenon would need to be accounted for in the rates of transitions out of regimes in the model if severe power caps were to be introduced, i.e., caps leading to significant performance degradations. However, the goal of this work is to select a power cap such that minimal performance degradation occurs, i.e., at most in the $\sim 5\%$ range. In such cases, this small bias in regime transtion rates leads to negligible differences in prediction of degradation (particularly for a short time horizon).

Figure 8: Performance degradation (percent compute time increase) bound as a function of power cap for the three example jobs from Figure 2. Jobs were updated using 200 minutes of history with right censoring (i.e., power cap) at the 95$^{\text{th}}$ percentile of the historical data. Predictions of the performance degradation for a future time horizon of 5 minutes were obtained on a grid of potential power caps. Predictions are summarized by the posterior mean and 95% credible bands and compared to the *actual* degradation computed from the actual power draw in the next five minutes.

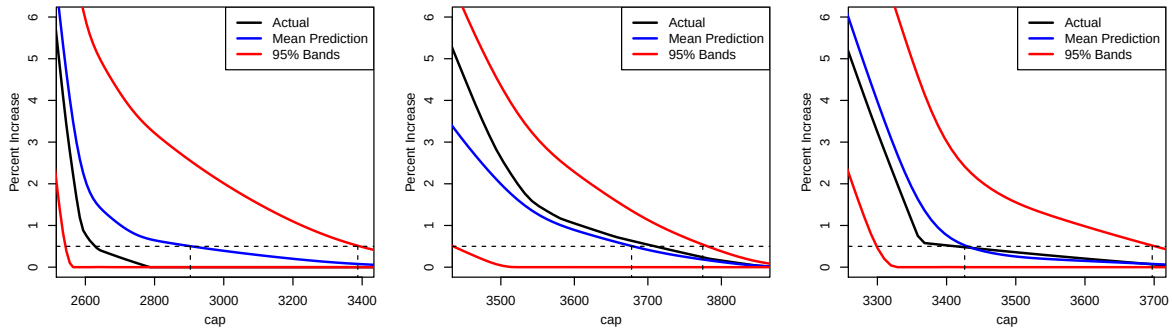
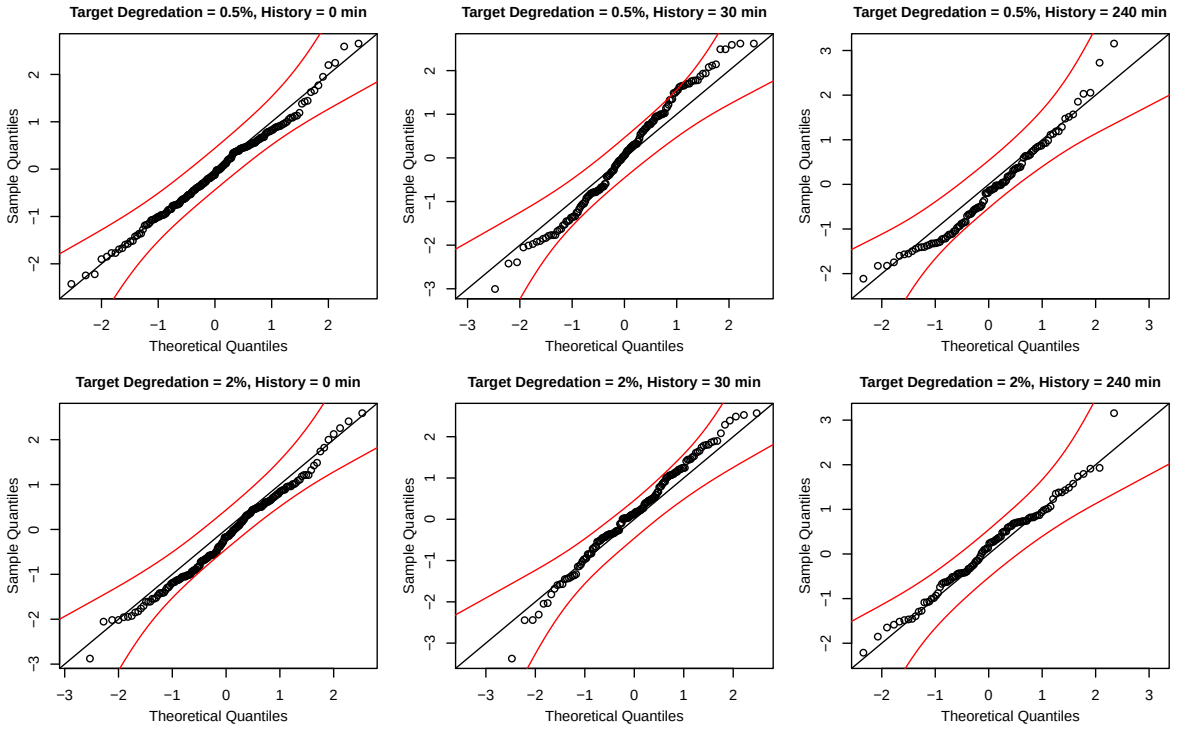


Figure 9: Normal Q-Q Plots resulting from prediction of performance degradation for each of the 213 jobs in the data set. Several prediction scenarios are considered by varying the targeted degradation (0.5% or 2%), and the history, i.e., the length of the time the job had been observed (0, 30, or 240 minutes). In each case, the historical data was assumed to be right censored by a power cap at the 95$^{\text{th}}$ percentile of the historical data.



The prediction accuracy across all 213 jobs in the dataset can be assessed via the Q-Q plots in Figure 9. The actual degradation is compared to the predictive distribution for a targeted degradation (0.5% or 2.0%) based on the mean prediction. The actual degradation for each job is converted to a Z-score via the respective predictive distribution. A normal Q-Q plot is then produced for the various prediction senarios of the amount of history used for updating (0, 30, or 240 minutes) and target degradation (0.5% or 2.0%). Simultaneous 95% confidence bands for each Q-Q plot were computed under the assumption that the data were independent standard normal deviates and are displayed for reference. In all cases, the Z-scores created from the predictive distribution fall inside the confidence bands, indicating that there is little to no sign of model inadequacy.

3.6 A Pragmatic Alternative to the Bayesian Model

An alternative to the full Bayesian model and MCMC is to use a simple, pragmatic approach to estimate the parameters of the model in (2) for each job. For example, the $\mu_{j,k}$ for the j^{th} job can be estimated using normal mixture clustering such as that in Fraley & Raftery (2002), provided by the `mclust` package in R. Conditional on the $\mu_{j,k}$, the value of the hidden ξ_j process could be simply inferred based on the most likely group membership from the normal mixture model. Once the the $\mu_{j,k}$ and ξ_j are assmed known, the parameters of the time to transition, transition probabilities, the AR(1) process and the observation error can be estimated via maximum likelihood estimates (MLEs). Let the estimate of all these parameters for the j^{th} job be denoted $\hat{\Theta}_j$. The parent distribution for each of the parameters could be taken to be the empirical distribution formed by the collection of $\{\hat{\Theta}_j\}$. The major advantage of this approach over the fully Bayesian approach is that it is incredibly simple and fast to implement. While it assumes the same model in (2), it makes far fewer assumptions regarding prior distributions. However, the major disadvantage is that it ignores the uncertainty in job specific parameter estimates and does not penalize toward the parent distribution to borrow strength when estimating

the parameters of a job that has a small number of observations.

A job may be updated in such a framework by calculating the conditional distribution of its parameters Θ^* given the new job's data X^*. This would involve calculating the likelihood $\mathcal{L}(\hat{\Theta}_j)$ of the new job's data for each parameter setting in the support of the parent distribution $\{\hat{\Theta}_j\}_{j=1}^J$. The conditional distribution would be given by

$$[\Theta^* \mid X^*] \propto \mathcal{L}(\hat{\Theta}_j) I_{\{\Theta^* = \hat{\Theta}_j\}}. \tag{15}$$

The disadvantage apparent in (15) is that the parent distribution was assumed to be the empirical (discrete) distribution of the $\hat{\Theta}_j$ resulting from the training data. If the training set contains enough data so that all jobs that will run on the machine in the future will be very similar to those seen in training, then this approach will work well. However, it will always be possible for the machine to see entirely new jobs. Still, this approach has the advantage of being free of many other assumptions about the parent distribution that have been made in the proposed Bayesian approach. The performance of this approach for the purpose of nodel level power capping is compared to that of the full Bayesian approach next in Section 4.

4 Optimal Power Capping Across an Entire Machine

In this section, a node-level power capping strategy is proposed and evaluated on a simulation study on the hypothetical Sol machine. The simulation setup assumes that all jobs on Sol encompass multiples of 10 nodes (i.e., a cage), i.e., since only cage level data was available. The 213 distinct Luna jobs in the data set were resampled and "launched" on the Sol machine in the sampled order until the machine no longer had room for the next job. Each job was required to cover the same number of cages as it did in reality on Luna. For example, if Sol had two cages idle, but the next job in the queue required three cages, it would have to wait until another job finished before it could start. Jobs finished after running the same amount of time as they really did on Luna. Once

a job finished, if there was then enough room for the next job in the queue, then it was launched at that time. If more than one job from the top of the queue would fit, then all such jobs were added. This process was run out to steady state ($\sim$ 1000 completed jobs). This queuing strategy is far simpler than the actual queuing system used at LANL, but it is only intended to provide a realistic job mix with which to test the capping strategies. At steady state, for example, $\sim$ 100 jobs will be running with varying start times and consequently a varying amount of time history.

For a given job mix at steady state, the following scenario is considered. Cage level caps must be imposed so that the entire system is subject to a power cap of 575 kW. With a 56.5 kW baseline, this means the sum of the node caps (or cage caps in this case) must be 518.5 kW. All idle cages automatically receive a fixed cap of 1.2 kW (i.e., barely above idle power draw). All cages running the same job receive the same cap. Therefore, for simplicity, consider the cap vector to be optimized as $\boldsymbol{c} = [c_1, c_2, \ldots, c_{J^*}]'$ containing the caps for each of the J^* running jobs. Since depending on the job mix, a number (N_{idle}) of cages may be at idle, the constraint becomes $\sum_j c_j \leq 518.5 - 1.2N_{\text{idle}}$ kW. We consider three power capping strategies all based on the predicted performance degradation (bound) for the next five minutes:

(i) Minimize the weighted mean performance degradation. That is, find the cage level power cap vector $\boldsymbol{c}_{avg}$, where

$$\boldsymbol{c}_{avg} = \arg\min_{\boldsymbol{c}} \left\{ \frac{\sum_j N_j \mathrm{E}[D_j]}{\sum_j N_j} \right\},$$

where D_j is the performance degradation for the j^{th} job in the next five minutes, and N_j is the number of cages used by the j^{th} job.

(ii) Minimize the expected maximum performance degradation, i.e., find the cage level power cap vector $\boldsymbol{c}_{max}$, where

$$\boldsymbol{c}_{max} = \arg\min_{\boldsymbol{c}} \left\{ \mathrm{E}\left[\max_j D_j \right] \right\}$$

(iii) Set each cage running a job to have the same power cap,

$$\boldsymbol{c}_{naive} = \frac{518.5 - 1.2N_{\text{idle}}}{154 - N_{\text{idle}}}.$$

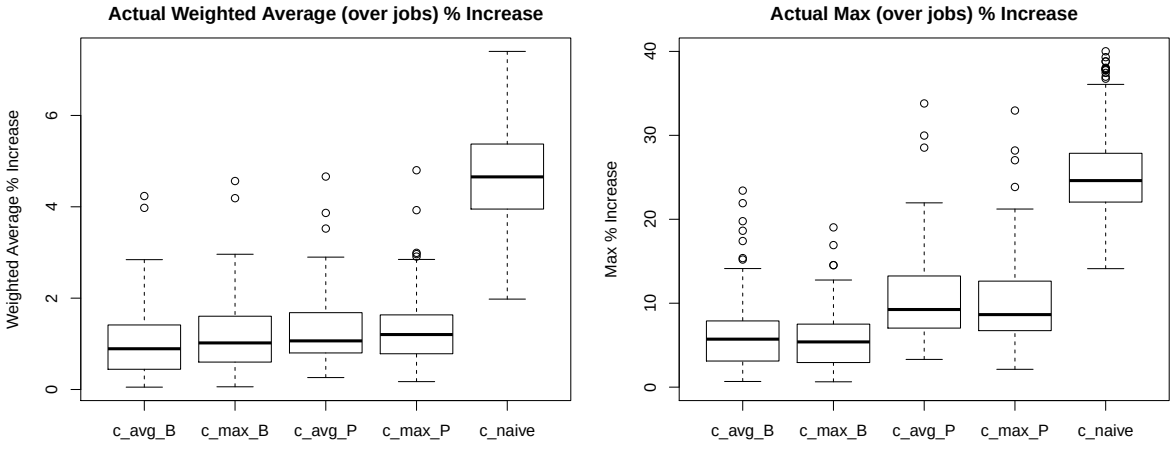
The first two strategies above were applied using both the fully Bayesian approach for estimation and updating and the simple pragmatic approach described in Section 3.6. In each case, 1000 realizations of the future job power draw were generated for each job. And then the `optim` function in R with the Nelder-Mead algorithm was used to find the optimal cap vector for each of the two criteria. To enforce the sum to $518.5 - N_{\text{idle}}1.2$ kW constraint, the cap vector $\boldsymbol{c}$ was reparameterized to a vector $\boldsymbol{c}^*$ where the value of the first element of $\boldsymbol{c}^*$ was fixed and the remaining $J^* - 1$ were allowed to vary freely. The mapping back to $\boldsymbol{c}$ is

$$\boldsymbol{c} = \boldsymbol{c}^* \left(\frac{518.5 - 1.2N_{\text{idle}}}{\sum_j c_j^*} \right).$$

Complete freedom for the other values could result in pathological/unrealistic choices for caps, so an exponential penalty was simply imposed in the optimization to discourage departure from reasonable limits, i.e., 1.2 kW $< c_j <$ 4.5 kW. Once this penalty was in place, all solutions were well inside of this range anyhow.

Five capping strategies were considered, (i) `c_avg_B` : $\boldsymbol{c}_{avg}$ using the full Bayesian approach, (ii) `c_max_B` : $\boldsymbol{c}_{max}$ using the full Bayesian approach, (iii) `c_avg_P` : $\boldsymbol{c}_{avg}$ using the pragmatic estimation approach in Section 3.6, (iv) `c_max_P` : $\boldsymbol{c}_{max}$ using the pragmatic estimation approach in Section 3.6, (v) `c_naive` : $\boldsymbol{c}_{naive}$. These five strategies were applied to the Sol machine once it had reached a steady state setting for job scheduling on 100 different randomly generated job mixes. The actual job performance degradation (bound) for the next five minutes was then calculated for each job and two summaries were calculated for each capping strategy: (i) the actual weighted average degradation over all running jobs and (ii) the actual maximum performance degradation across all running jobs. Figure 10 displays the resulting box plots of these two metrics from the 100 simulated scenarios for each of the five capping strategies.

Figure 10: Actual degradation results for the weighted average increase and the maximum increase, respectively, according to the five capping strategies. Boxplots are created from the degradation's that would have occurred for each of the 100 job mix realizations.



It is clear from Figure 10 that all of the proposed statistical capping strategies far outperform the naive (same cap for each job) approach. The naive approach results in a weighted average increase of $\sim$ 4% (on average over the 100 job mix realizations), whereas the weighted average increase for the statistical strategies is $\sim$ 1% on average. To put this in perspective, this 3% advantage of computational efficiency would be the equivalent of freeing up 4.5 cages (i.e., 45 nodes) on Sol for additional computation. The amount of overhead required to use this approach would be < 2 nodes for the full Bayesian approach and < 1 node for the pragmatic estimation approach.

The naive approach also produces a max increase of $\sim$ 24% on average over the 100 realizations and as much as a 40% increase in some cases. In contrast, the Bayesian statistical strategies keep the maximum increase at about 6% on average. As would be expected, `c_avg_B` and `c_avg_P` perform better than `c_max_B` and `c_max_P` on minimizing the weighted average increase. However, the opposite is true when using the observed maximum increase as the performance metric. All of these differences are statistically significant, even if not all that important, practically.

The results for the pragmatic approach are very competitive with the full Bayesian

approach at minimizing the weighted average increase. The full Bayesian approach, however, does a better job of minimizing the maximum job increase. This makes some intuitive sense, as accurate prediction of the expected maximum would rely more heavily on representation of uncertainty. And the Bayesian approach addresses much of the estimation uncertainty that the pragmatic approach inherently ignores. Still the pragmatic approach is an order of magnitude faster than the full Bayesian approach, and may remain in the discussion for implementation purposes, depending on the performance goal.

5 Conclusions & Further Work

A novel statistical model has been developed for the power used by a HPC jobs. To the best of our knowledge, this is the first attempt to statistically model this process. This model was then used to inform an intelligent node-level power capping strategy, with the intention that this could be used for systems in the near future that have a node-level capping mechanism. This approach has been demonstrated on a hypothetical machine, molded from a real machine at LANL. The results demonstrate that the proposed approach is about $5\times$ more efficient than the simple approach where all nodes receive the same power cap. In addition, the job power model introduced here could have applications beyond power capping, such as more intelligent scheduling, optimizing complicated power contracts with utilities, improving the power efficiency of jobs, etc.

There are two important areas where this approach could be further refined. In particular, the current model assumes that new jobs come from a large population of all jobs that could run on the machine. This results in a large amount of uncertainty when predicting the future of a new job prior to seeing any data from it. Thus, it may be beneficial to introduce another *user* level into the hierarchical model. In other words, a new job could come from a specific user's population, as opposed to the population of all possible jobs. Finally, as mentioned previously, the MCMC approach to update jobs is currently fast enough for practical application. However, it would still be pru-

dent to explore possible SMC solutions to job updating in order to further increase the computational efficiency.

References

Andrieu, C., Doucet, A. & Tadic, V. B. (2005), On-line parameter estimation in general state-space models, *in* 'Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on', IEEE, pp. 332–337.

Beal, M. J., Ghahramani, Z. & Rasmussen, C. E. (2002), 'The infinite hidden Markov model', *Advances in Neural Information Processing Systems* **14**, 577–584.

Del Moral, P., Doucet, A. & Jasra, A. (2006), 'Sequential Monte Carlo samplers', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(3), 411–436.

Doucet, A., de Freitas, N. & Gordon, N. (2001), *Sequential Monte Carlo Methods in Practice*, Springer Verlag, New York, NY.

Dukowicz, J. K. & Smith, R. D. (1994), 'Implicit free-surface method for the Bryan-Cox-Semtner ocean model', *Journal of Geophysical Research: Oceans (1978–2012)* **99**(C4), 7991–8014.

Ferguson, T. S. (1973), 'A Bayesian analysis of some nonparametric problems', *The Annals of Statistics* **1**(2), 209–230.

Fox, E. B., Hughes, M. C., Sudderth, E. B. & Jordan, M. I. (2014), 'Joint modeling of multiple related time series via the beta process with application to motion capture segmentation', *Annals of Applied Statistics* **8**(3), 1281–1313.

Fox, E. B., Sudderth, E. B., Jordan, M. I. & Willsky, A. S. (2011), 'A sticky HDP-HMM with application to speaker diarization', *Annals of Applied Statistics* **5**, 1020–1056.

Fraley, C. & Raftery, A. (2002), 'Model-based clustering, discriminant analysis, and density estimation', *Journal of the American Statistical Association* **97**, 611–631.

Freeh, V. W., Lowenthal, D. K., Pan, F., Kappiah, N., Springer, R., Rountree, B. L. & Femal, M. E. (2007), 'Analyzing the energy-time trade-off in high-performance computing applications', *IEEE Transactions on Parallel and Distributed Systems* **18**(6), 835–848.

Ge, R. & Cameron, K. W. (2007), Power-aware speedup, *in* 'Proceedings of the 21st IEEE Parallel and Distributed Processing Symposium'.

Givens, G. & Hoeting, J. (2000), *Computational Statistics*, 1st edn, Hoboken, NJ: John Wiley & Sons.

Havard Rue, L. H. (2005), *Gaussian Markov Random Fields: Theory and Applications*, Chapman & Hall/CRC.

Hsu, C.-H. & Feng, W. (2005), A power-aware run-time system for high-performance computing, *in* 'Proceedings of the ACM/IEEE Supercomputing Conference (SC'05)'.

Ishwaran, H. & James, L. F. (2001), 'Gibbs sampling methods for stick-breaking priors', *Journal of the American Statistical Association* **96**(453).

Kamil, S., Shalf, J. & Strohmaier, E. (2008), Power efficiency in high performance computing, *in* 'Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on', pp. 1–8.

Kottas, A. & Taddy, M. A. (2009), 'Markov switching Dirichlet process mixture regression', *Bayesian Analysis* **4**, 793–816.

Lennox, K. P., Dahl, D. B., Vannucci, M., Day, R. & Tsai, J. W. (2010), 'A Dirichlet process mixture of hidden Markov models for protein structure prediction', *Annals of Applied Statistics* **4**, 916–962.

Lid Hjort, N., Holmes, C., Müller, P. & Walker, S. G. (2010), *Bayesian Nonparametrics*, Cambridge University Press.

Liu, J. & Chen, R. (1998), 'Sequential Monte Carlo methods for dynamic systems', *Journal of the American Statistical Association* **93**, 1032–1044.

Liu, J. & West, M. (2001), Combined parameter and state estimation in simulation-based filtering, *in* 'Sequential Monte Carlo methods in practice', Springer, pp. 197–223.

Michalak, S., DuBois, A., Storlie, C., Quinn, H., Rust, W., DuBois, D., Modl, D., Manuzzato, A. & Blanchard, S. (2012), 'Assessment of the impact of cosmic-ray-induced neutrons on hardware in the Roadrunner supercomputer', *IEEE Transactions on Device and Materials Reliability* **12**(2), 445–454.

Paisley, J. & Carin, L. (2010), 'Hidden Markov models with stick-breaking priors', *IEEE Trans. Signal Processing* **59**, 3905–3917.

Pakin, S. & Lang, M. (2013), 'Energy modeling of supercomputers and large-scale scientific applications', *Proceedings of the 4th International Green Computing Conference (IGCC 2013)* .

Pakin, S., Storlie, C., Lang, M., Fields III, R. E., Romero, E. E., Idler, C., Michalak, S., Greenberg, H., Loncaric, J., Rheinheimer, R., Grider, G. & Wendelberger, J. (2013), 'Power usage of production supercomputers and production workloads', *Concurrency and Computation: Practice and Experience (in press)* .

Patki, T., Lowenthal, D. K., Rountree, B., Schulz, M. & de Supinski, B. R. (2013), Exploring hardware overprovisioning in power-constrained, high performance computing, *in* 'Proceedings of the 27th international ACM conference on International conference on supercomputing', ACM, pp. 173–182.

Pitt, M. K. & Shephard, N. (1999), 'Filtering via simulation: Auxiliary particle filters', *Journal of the American Statistical Association* **94**(446), 590–599.

Polson, N. G., Stroud, J. R. & Müller, P. (2008), 'Practical filtering with sequential parameter learning', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(2), 413–428.

Sethuraman, J. (1994), 'A constructive definition of Dirichlet priors', *Statistica Sinica* **4**, 639–650.

Storlie, C. B., Michalak, S. E., Quinn, H. M., Dubois, A. J., Wender, S. A. & Dubois, D. H. (2013), 'A Bayesian reliability analysis of neutron-induced errors in high performance computing hardware', *Journal of the American Statistical Association* **108**(502), 429–440.

Storvik, G. (2002), 'Particle filters for state-space models with the presence of unknown static parameters', *Signal Processing, IEEE Transactions on* **50**(2), 281–289.

Teh, Y., Jordan, M. I., Beal, M. & Blei, D. (2006), 'Hierarchical Dirichlet processes', *Journal of the American Statistical Association* **101**, 1566–1581.

Zhang, Z., Lang, M., Pakin, S. & Fu, S. (2014), Trapped capacity: Scheduling under a power cap to maximize machine-room throughput, *in* 'Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing', IEEE Press, pp. 41–50.

Supplementary Material: "Modeling and Predicting Power Consumption of High Performance Computing Jobs"

A MCMC Computational Details

As mentioned in the main paper, each iteration of the MCMC algorithm for parent parameter estimation consists of (i) updating the job specific parameters Θ_j, $j = 1, \ldots, J$, and (ii) updating the parent parameters Θ^P. The MCMC algorithm for updating a new job given its data up until the current time consists of just iterating over (i) for that single new job. The full conditionals for Gibbs sampling and/or MH steps used are provided in Section A.1 for the elements of Θ_j, and in Section A.2 for the elements of Θ^P.

A.1 Updates for Job Specific Parameters

The entire collection of parameters to be sampled for the j^{th} job is

$$\Theta_j = \left\{ \{\xi_j(t)\}_{t=1}^{T_j}, \{\lambda_{j,k}\}_{k=1}^{K}, \{\pi_{j,k}\}_{k=1}^{K}, \{\mu_{j,k}\}_{k=1}^{K}, \{z_j(t)\}_{t=1}^{T_j}, \sigma_j^2, \rho_j \right\}, \; j = 1, \ldots, J.$$

As mentioned in the main paper, the number of components in stick-breaking model for $\pi_{j,k}$ was capped at a finite value K, for computational convenience. The value of $\pi_{j,K}$ was observed and K was increased until $\pi_{j,K}$ values were negligible for all jobs at $K = 10$. Before proceeding to the full conditionals for the elements of Θ_j the following equivalent model for the $\xi_j(t)$ is introduced because it allows conjugate updates for the $\pi_{j,k}$.

Assume that the time to a *possible* transition when in regime $\xi_j(t) = k$ is

$$T^* \sim \text{Geometric}\left(\lambda_{j,k}\right). \tag{A1}$$

At a possible transition time $T^* = u$ a regime label is chosen according to

$$\Pr(\xi_j(u) = l \mid \xi_j(u-1) = k) = \pi_{j,l}, \quad \text{for } l = 1, \ldots, K \tag{A2}$$

That is, at the time of a possible transition out of state k, a regime l is chosen with probability equal to $\pi_{j,l}$. The difference here is the possibility that $l = k$ (i.e., there is possibly no regime change at time u).

Proposition 1. *The model for $\xi_j(t)$ proposed in (A1) and (A2) leads to an equivalent model to that described in (4) and (5) in the main paper.*

It is relatively straight-forward to justify Proposition 1 by recognizing that the model for $\xi_j(t)$ described in described in (4) and (5) is a discrete time MC, so too is that described in (A1) and (A2), and they have the same probability transition matrix.

With this new representation for ξ_j, we also introduce a new latent variable $\phi_j(t)$ to be sampled in the MCMC; $\phi_j(t)$ is the indicator of whether or not time t was a *possible* transition time. This is done to allow for conjugate updates of $\lambda_{j,k}$, conditional on ϕ_j. A latent variable $\psi_{j,k} \in \{1, \ldots, M\}$ is also introduced, representing the index of the component in the normal mixture in (10) that produced $\mu_{j,k}$. This is to allow for conjugate updates of the $\mu_{j,k}$ and parent parameters of the normal mixture. The complete collection of parameters to be sampled for the j^{th} job is then

$$\Theta_j = \left\{ \{\xi_j(t)\}_{t=1}^{T_j}, \{\phi_j(t)\}_{t=1}^{T_j}, \{\lambda_{j,k}\}_{k=1}^{K}, \{\pi_{j,k}\}_{k=1}^{K}, \{\mu_{j,k}\}_{k=1}^{K}, \{\psi_{j,k}\}_{k=1}^{K}, \{z_j(t)\}_{t=1}^{T_j}, \sigma_j^2, \rho_j \right\}.$$

$\underline{\xi_j(t), \phi_j(t) \mid \text{rest}}$

Let *rest* denote the data for the j^{th} job and all parameters in Θ^P and in Θ_j except $\xi_j(t)$ and $\phi_j(t)$. Because of the Markov property of $\xi_j(t)$, conditional on all other parameters and the data (i.e., rest), $\{\boldsymbol{\xi}_j(t), \phi_j(t)\}$ only depends on $\{\xi_j(s) : s \neq t\}$ through $\xi_j(t-1)$ (for $t > 1$) and $\xi_j(t+1)$ (for $t < T_j$). That is,

$$\begin{aligned} \Pr\left(\xi_j(t) = k \mid \text{rest}\right) &\propto \Pr\left(\xi_j(t) = k \mid \xi_j(t-1)\right) \Pr\left(\xi_j(t+1) \mid \xi_j(t) = k\right) \mathcal{L}\left(x_j(t) - \mu_{j,k} - Z_j(t)\right) \\ &= P_{\xi_j(t-1),k} P_{k,\xi_j(t+1)} N(x_j(t) - \mu_{j,k} - Z_j(t); 0, \tau^2), \end{aligned} \tag{A3}$$

where $P_{k,l}$ was defined in (6) and $N(\cdot, 0, \tau^2)$ is the Gaussian density with mean 0 and variance τ^2. Once $\xi_j(t)$ is updated via (A3) the indicator $\phi_j(t)$ can be updated conditional on the rest **and** $\xi_j(t)$ as

$$\Pr\left(\phi_j(t) = 1 \mid \text{rest}\ ,\ \xi_j(t)\right) = \begin{cases} 1 & \text{if } \xi_j(t-1) \neq \xi_j(t) \\ \frac{\lambda_{j,k}\pi_{j,k}}{1-\lambda_{j,k}(1-\pi_{j,k})} & \text{if } \xi_j(t-1) = \xi_j(t) = k. \end{cases}$$

$\underline{\lambda_{j,k} \mid \text{rest}}$

Conditional on Θ^P all other parameters in Θ_j, $\lambda_{j,k}$ only depends on $\{\phi_j(t)\}_{t=1}^{T_j}$. Specifically,

$$\lambda_{j,k} \mid \text{rest} \sim \text{Beta}(a^*, b^*),$$

where $a^* = \alpha_\lambda + M_{j,k}$, $b^* = \beta_\lambda + N_{j,k} - M_{j,k}$, $M_{j,k}$ is the number of *possible* transitions generated from state k, and $N_{j,k}$ is the number of total time steps observed from state

k, i.e.,

$$\begin{aligned} M_{j,k} &= \sum_{t=1}^{T_j-1} \phi_j(t+1) I_{\{\xi_j(t)=k\}} \\ N_{j,k} &= \sum_{t=1}^{T_j-1} I_{\{\xi_j(t)=k\}} \end{aligned}$$

$\underline{\boldsymbol{\pi}_j = [\pi_{j,1}, \ldots, \pi_{j,K}]' \mid \text{rest}}$

There is a one to one correspondence between $\boldsymbol{\pi}_j$ and $\boldsymbol{v}_j = [v_{j,1}, \ldots, v_{j,K}]'$ in (12). Conditional on Θ^P and the rest of Θ_j, $\boldsymbol{v}_j$ depends only on $\{\phi_j(t)\}_{t=1}^{T_j}$, $\{\xi_j(t)\}_{t=1}^{T_j}$, and $\tilde{\delta}$. Specifically,

$$v_{j,k} \mid \text{rest} \overset{ind}{\sim} \text{Beta}(a_k^*, b_k^*),$$

where,

$$\begin{aligned} a_k^* &= \sum_{t=1}^{T_j} \phi_j(t) I_{\{\xi_j(t)=k\}} + 1 \\ b_k^* &= \sum_{t=1}^{T_j} \phi_j(t) I_{\{\xi_j(t)>k\}} + \tilde{\delta} \end{aligned}$$

$\underline{\mu_{j,k} \mid \text{rest}}$

Let $\boldsymbol{x}_j^{(k)}$ be the vector (ordered in time) of all $x_j(t)$ when $\xi_j(t) = k$. Define the vectors $\boldsymbol{z}_j^{(k)}$ and $\boldsymbol{\varepsilon}_j^{(k)}$ analogously. Then define $\boldsymbol{r}_j^{(k)} = \boldsymbol{x}_j^{(k)} - \boldsymbol{z}_j^{(k)} = \mu_{j,k} - \boldsymbol{\varepsilon}_j^{(k)}$, and let $n_{j,k}$ be the length of the $\boldsymbol{r}_j^{(k)}$ vector. Lastly for convenience of notation, let the current value of $\psi_{j,k}$ be denoted as m. All *observations* $\boldsymbol{r}_j^{(k)}$ are *iid* from the same normal distribution with

known variance and mean $\mu_{j,k} \sim N(\nu_m, \varsigma_m^2)$, resulting in a simple conjugate update, i.e.,

$$\mu_{j,k} \mid \text{rest} \sim N(\mu^*, \sigma^{2^*}),$$

where,

$$\begin{aligned} \mu^* &= \sigma^{2^*}\left(\frac{\nu_m}{\varsigma_m^2} + \frac{\sum_i r_{j,i}}{\tau^2}\right) \\ \sigma^{2^*} &= \left(\frac{1}{\varsigma_m^2} + \frac{\sum_i n_{j,k}}{\tau^2}\right)^{-1}. \end{aligned}$$

$\underline{\psi_{j,k} \mid \text{rest}}$

$\Pr(\psi_{j,k} = m \mid \text{rest}) \propto \tilde{\omega}_m N(\mu_{j,k}; \tilde{\nu}_m, \tilde{\varsigma}_m^2)$

$\underline{z_j(t) \mid \text{rest}}$

Let $r_j(t) = x_j(t) - \mu_{j,\xi_j(t)} = z_j(t) + \varepsilon_j(t)$, and let the time ordered vector of the $r_j(t)$ be denoted $\boldsymbol{r}_j$. All *observations* $\boldsymbol{r}_j$ are *ind* from a normal distribution with known variance and mean vector $\boldsymbol{z}_j \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{\Gamma}_{\rho_j})$, where $\boldsymbol{\Gamma}_{\rho_j}$ is correlation matrix for $\boldsymbol{z}_j$ formed by evaluating the correlation function in (3) at the observed time points for the j^{th} job.

$$\boldsymbol{z}_j \mid \text{rest} \sim N(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*),$$

where,

$$\begin{aligned} \boldsymbol{\mu}^* &= \boldsymbol{\Sigma}^* \boldsymbol{r}_j \\ \boldsymbol{\Sigma}^* &= \tau^2 \left(\boldsymbol{I} + \frac{\tau^2}{\sigma_j^2} \boldsymbol{\Gamma}_{\rho_j}\right)^{-1}. \end{aligned} \tag{A4}$$

It is far more efficient to use the kalman filter or use Gaussian Markov random fields

(GMRF) results (Havard Rue 2005) in this case as opposed to actually evaluating the inverse in (A4). We use the latter here, let $\boldsymbol{Q} = 1/\sigma_j^2 \boldsymbol{\Gamma}_{\rho_j}^{-1}$, i.e., the precision matrix for the prior on $\boldsymbol{z}_j$. This matrix is readily obtainable without a matrix decomposition due the Markov model imposed by the exponential covariance function. The precision matrix $(\boldsymbol{\Sigma}^*)^{-1}$ for the update of $\boldsymbol{z}_j$ is a sparse matrix with a bandwidth of 3. Efficient algorithms exist for generating a multivariate normal vector in such cases, see page 31 of Havard Rue (2005) for example.

MH update for σ_j^2

As mentioned in the main paper, the full conditional distribution of $\sigma_j^2 \mid$ rest does not have a convenient form with which to perform Gibbs updates. However, the MH ratio has a simple form which is easy to compute. The prior used is $\sigma_j^2 \sim \log N(\tilde{\mu}_\sigma, \tilde{\sigma}_\sigma^2)$. Proposals for σ_j^{2*} are made by identifying an a and b for an $\text{IG}(a, b)$ distribution with the same mean and variance as the log-normal prior. By assuming the prior for $\sigma_j^2 \sim \text{IG}(a, b)$, the update is conjugate. Thus we take the proposal to be this conjugate update, specifically,

$$\sigma_j^{2*} \sim \text{IG}(a + T_j/2 \ , \ b + \sum_t Z_j(t)^2).$$

That is, the proposals are independent of the current σ_j^2 value, let the density of the proposal be denoted $d(\sigma_j^{2*})$. The only portion of the full model likelihood that differs between the current value and the proposal is $\mathcal{L}(\boldsymbol{z}_j; \sigma_j^2, \rho_j)$ which is a multivariate normal density with mean 0 and covariance $\sigma_j^2 \boldsymbol{\Gamma}_{\rho_j}$. As with the update of $\boldsymbol{z}_j$, there are efficient means of evaluating this density (or the log-density) via GMRF results. The MH ratio is then

$$MH = \frac{\mathcal{L}(\boldsymbol{z}_j; \sigma_j^{2*}, \rho_j)\pi(\sigma_j^{2*})d(\sigma_j^2)}{\mathcal{L}(\boldsymbol{z}_j; \sigma_j^2, \rho_j)\pi(\sigma_j^2)d(\sigma_j^{2*})}.$$

MH update for ρ_j

As mentioned in the main paper, the ρ_j were updated via a MH random walk proposals. However, the random walk was conducted on the log scale, i.e., $\log(\rho_j^*) = \log(\rho_j + \epsilon)$ for a deviate $\epsilon \sim N(0, s^2)$. With the use of the log scale, a constant tuning parameter $s^2 = 0.25$ could be used for all jobs to achieve acceptances across all jobs in the range of (30% - 55%). Let the density of the proposal, given the current value of ρ_j be denoted $d(\rho_j{}^* \mid \rho_j)$. As with updates of σ_j^2 above, the only portion of the full model likelihood that differs between the current value and the proposal is $\mathcal{L}(\boldsymbol{z}_j; \sigma_j^2, \rho_j)$. The MH ratio is then

$$MH = \frac{\mathcal{L}(\boldsymbol{z}_j; \sigma_j^2, \rho_j^*)\pi(\rho_j^*)d(\rho_j \mid \rho_j^*)}{\mathcal{L}(\boldsymbol{z}_j; \sigma^2, \rho_j)\pi(\rho_j)d(\rho_j^* \mid \rho_j)}.$$

A.2 Updates for Parent Parameters

The entire collection of parent parameters sampled in the MCMC is

$$\Theta^P = \left\{\tilde{\mu}_\sigma, \tilde{\sigma}_\sigma, \tilde{\mu}_\rho, \tilde{\sigma}_\rho, \{\tilde{\omega}_m\}_{m=1}^M, \{\tilde{\nu}_m\}_{m=1}^M, \left\{\tilde{\varsigma}_m^2\right\}_{m=1}^M, \tilde{\alpha}_\lambda, \tilde{\beta}_\lambda, \tilde{\gamma}, \tilde{\delta}, \tilde{\tau}^2\right\}$$

$\tilde{\mu}_\sigma \mid$ rest

Conditional on the rest, $\tilde{\mu}_\sigma$ has a simple conjugate normal update,

$$\tilde{\mu}_\sigma \mid \text{rest} \sim N(\mu^*, \sigma^{2^*}),$$

where,

$$\begin{aligned} \mu^* &= \sigma^{2^*}\left(\frac{M_\sigma}{S_\sigma^2} + \frac{\sum_{j=1}^J \log(\sigma_j^2)}{\tilde{\sigma}_\sigma^2}\right) \\ \sigma^{2^*} &= \left(\frac{1}{S_\sigma^2} + \frac{J}{\tilde{\sigma}_\sigma^2}\right)^{-1}. \end{aligned}$$

$\underline{\tilde{\sigma}_\sigma \mid \text{rest}}$

Conditional on the rest, $\tilde{\sigma}_\sigma^2$ has a simple conjugate IG update,

$$\tilde{\sigma}_\sigma^2 \sim \text{IG}\left(\alpha_\sigma + \frac{J}{2}\ ,\ \beta_\sigma + \frac{1}{2}\sum_{j=1}^{J}\left[\log(\sigma_j^2) - \tilde{\mu}_\sigma\right]^2\right).$$

$\underline{\tilde{\mu}_\rho \mid \text{rest}}$

Conditional on the rest, $\tilde{\mu}_\rho$ has a simple conjugate normal update,

$$\tilde{\mu}_\rho \mid \text{rest} \sim N(\mu^*, \sigma^{2*}),$$

where,

$$\begin{aligned}\mu^* &= \sigma^{2*}\left(\frac{M_\rho}{S_\rho^2} + \frac{\sum_{j=1}^{J}\log(\rho_j)}{\tilde{\sigma}_\rho^2}\right)\\ \sigma^{2*} &= \left(\frac{1}{S_\rho^2} + \frac{J}{\tilde{\sigma}_\rho^2}\right)^{-1}.\end{aligned}$$

$\underline{\tilde{\sigma}_\rho \mid \text{rest}}$

Conditional on the rest, $\tilde{\sigma}_\rho^2$ has a simple conjugate IG update,

$$\tilde{\sigma}_\rho^2 \sim \text{IG}\left(\alpha_\rho + \frac{J}{2}\ ,\ \beta_\rho + \frac{1}{2}\sum_{j=1}^{J}\left[\log(\rho_j) - \tilde{\mu}_\rho\right]^2\right).$$

$\underline{\boldsymbol{\omega} = [\tilde{\omega}_1, \ldots, \tilde{\omega}_M]' \mid \text{rest}}$

There is a one to one correspondence between $\boldsymbol{\omega}$ and $\boldsymbol{u} = [u_1, \ldots, u_M]'$ in (9). Conditional on Θ^P and the rest of Θ_j, $\boldsymbol{u}$ depends only on the $\left\{\{\psi_{j,k}\}_{j=1}^{J}\right\}_{k=1}^{K}$ and $\tilde{\gamma}$. Specifically,

$$u_m \mid \text{rest} \overset{ind}{\sim} \text{Beta}(a_m^*, b_m^*),$$

where,

$$\begin{aligned} a_m^* &= \sum_{j=1}^{J}\sum_{k=1}^{K} I_{\{\psi_{j,k}=m\}} + 1 \\ b_m^* &= \sum_{j=1}^{J}\sum_{k=1}^{K} I_{\{\psi_{j,k}>m\}} + \tilde{\gamma} \end{aligned}$$

$\underline{\tilde{\nu}_m \mid \text{rest}}$

Let $\boldsymbol{\mu}^{(m)}$ be the vector of all $\mu_{j,k}$ when $\psi_{j,k} = m$, and let n_m denote the length of this vector. All *observations* $\boldsymbol{\mu}^{(m)}$ are *iid* from the same normal distribution with known variance and mean $\tilde{\nu}_m \sim N(M_\nu, S_\nu^2)$, resulting in a simple conjugate update, i.e.,

$$\tilde{\nu}_m \mid \text{rest} \sim N(\mu^*, \sigma^{2^*}),$$

where,

$$\begin{aligned} \mu^* &= \sigma^{2^*}\left(\frac{M_\nu}{S_\nu^2} + \frac{\sum_{i=1}^{n_m}\mu_i^{(m)}}{\varsigma_m^2}\right) \\ \sigma^{2^*} &= \left(\frac{1}{S_\nu^2} + \frac{n_m}{\varsigma_m^2}\right)^{-1}. \end{aligned}$$

$\underline{\tilde{\varsigma}_m^2 \mid \text{rest}}$

Conditional on the rest, $\tilde{\varsigma}_m^2$ has a simple conjugate IG update,

$$\tilde{\varsigma}_m^2 \sim \text{IG}\left(A_\varsigma + \frac{n_m}{2}\ ,\ B_\varsigma + \frac{1}{2}\sum_{i=1}^{n_m}\left(\mu_i^{(m)} - \tilde{\nu}_m\right)^2\right).$$

$\underline{\tilde{\gamma} \mid \text{rest}}$

Conditional on the rest, $\tilde{\gamma}$ depends only on the $\tilde{\omega}_m$ (or equivalently the u_m). One can equivalently think of the SB model as a prior for $u_m = \Pr(\psi_{j,k} = m \mid \psi_{j,k} > m-1) \overset{iid}{\sim} \text{Beta}(1, \tilde{\gamma})$, for $m = 1, \ldots, M$. A Gamma prior is conjugate for $\tilde{\gamma}$ in this model, thus, $\tilde{\gamma}$ has the update,

$$\tilde{\gamma} \sim \text{Gamma}\left(A_\gamma + M\ ,\ B_\gamma - \sum_{m=1}^{M-1}\log(1-u_m)\right).$$

$\underline{\tilde{\delta} \mid \text{rest}}$

Conditional on the rest, $\tilde{\delta}$ depends only on the $\pi_{j,k}$ (or equivalently the $v_{j,k}$). By a completely analogous argument, as that for the update of $\tilde{\gamma}$ above, $\tilde{\delta}$ has a conjugate Gamma update,

$$\tilde{\delta} \sim \text{Gamma}\left(A_\delta + JK\ ,\ B_\delta - \sum_{j=1}^{J}\sum_{k=1}^{K-1}\log(1-v_{j,k})\right).$$

$\underline{\tilde{\tau}^2 \mid \text{rest}}$

Let $r_j(t) = x_j(t) - \mu_{j,\xi_j(t)} - z_j(t)$. Then $r_j(t) \overset{iid}{\sim} N(0, \tilde{\tau}^2)$, and the inverse-Gamma prior

on $\tilde{\tau}^2$ is conjugate, leading to the simple update,

$$\tilde{\tau}^2 \sim \text{IG}\left(A_\tau + \frac{1}{2}\sum_{j=1}^{J} T_j \ , \ B_\tau + \frac{1}{2}\sum_{j=1}^{J}\sum_{t=1}^{T_j} r_j(t)^2\right).$$

MH update for $\tilde{\alpha}_\lambda$

As mentioned in the main paper, $\tilde{\alpha}_\lambda$ was updated via a MH random walk proposals. Again, the random walk was conducted on the log scale, i.e., $\log(\tilde{\alpha}^*_\lambda) = \log(\tilde{\alpha}_\lambda + \epsilon)$ for a deviate $\epsilon \sim N(0, s^2)$. A tuning parameter $s^2 = 0.01$ was used to achieve an acceptance rate of 40%. Let the density of the proposal, given the current value of $\tilde{\alpha}_\lambda$ be denoted $d(\tilde{\alpha}^*_\lambda \mid \tilde{\alpha}_\lambda)$. The only portion of the full model likelihood that differs between the current value and the proposal is

$$\mathcal{L}(\boldsymbol{\lambda}; \tilde{\alpha}_\lambda, \tilde{\beta}_\lambda) = \prod_{j=1}^{J}\prod_{k=1}^{K} \text{Beta}(\lambda_{j,k}; \tilde{\alpha}_\lambda, \tilde{\beta}_\lambda).$$

The MH ratio is then

$$MH = \frac{\mathcal{L}(\boldsymbol{\lambda}; \tilde{\alpha}^*_\lambda, \tilde{\beta}_\lambda)\pi(\tilde{\alpha}^*_\lambda)d(\tilde{\alpha}_\lambda \mid \tilde{\alpha}^*_\lambda)}{\mathcal{L}(\boldsymbol{\lambda}; \tilde{\alpha}_\lambda, \tilde{\beta}_\lambda)\pi(\tilde{\alpha}_\lambda)d(\tilde{\alpha}^*_\lambda \mid \tilde{\alpha}_\lambda)}$$

MH update for $\tilde{\beta}_\lambda$

The update for $\tilde{\beta}_\lambda$ was conducted in a completely analogous manner to that for $\tilde{\alpha}_\lambda$ above.