Optimal Reliability-Constrained Overdrive Frequency Selection In Multicore Systems

Andrew B. Kahng[†‡] and Siddhartha Nath[†]

[†]CSE and [‡]ECE Departments, University of California at San Diego, USA
{abk, sinath}@ucsd.edu

Abstract—In leading-edge process technologies, reliability is a first-class constraint for both IC design and system operation. For multicore systems, reliability affects task scheduling decisions since it constrains both performance and throughput. Previous works on reliability-constrained task scheduling have two basic limitations: either they cannot guarantee lifetime (e.g., that the chip can deliver useful performance over 10 years), or they cannot guarantee lower bounds on "acceptable performance" or "acceptable throughput" for the entire chip lifetime. In this work, we formulate and solve a new *maximum-value, reliability-constrained overdrive frequencies* (MVRCOF) problem that guarantees prescribed lower bounds on "acceptable performance" and "acceptable throughput" in multicore systems, without exceeding prescribed lifetime budget for any core. Our formulation maximizes value of overdrive frequencies for each number of active cores. We develop a solver for the MVRCOF problem and present optimal and heuristic solutions that determine the execution times of each core in each combination of simultaneously active cores, such that cores wear out in a balanced manner over the chip lifetime. These solutions deliver maximum value within the specified chip lifetime, and can be used for reliability-constrained scheduling policies. Our heuristic method can be 3.3% worse than our optimal method, but can converge up to 10× faster. Further, our solutions improve the objective function value by between 2.2% and 17.4% when compared to existing reliability-constrained task scheduling policies that provide lifetime guarantees.

I. Introduction

Modern systems with multicore processors typically operate at multiple operating modes, e.g., nominal and overdrive (a.k.a. Turbo) [28]. Applications running on multicore systems have different requirements for the number of cores used at any given moment, as well as for corresponding operating modes. For example, in a system with eight cores, applications A and B may have very different usage of these cores over time. Figure 1(a) conceptually illustrates the time-varying usage of cores by applications A and B. Each application uses at most four cores simultaneously over the course of its execution. When memory and I/O resources are not a bottleneck, *the operating system scheduler packs tasks using some or all of the available processing cores in the multicore system*. Figure 1(b) illustrates how the scheduler might pack executions of A and B across eight cores (cf. Packed A, B shown in green color in Figure 1(b)). A key observation is that when all of the cores are not simultaneously active, task scheduling on a subset of the available cores can be adjusted so that the cores wear out in a balanced manner and meet lifetime as well as performance requirements.

Mean time to failure (MTTF) is a measure of the lifetime of a core. MTTF of a core degrades due to reliability mechanisms such as electromigration, time-dependent dielectric breakdown, stress migration, thermal cycling, bias temperature instability, hot carrier injection, etc. [15] [18]. Below, we use the following terminology.

- *power-on-hours* (POH) denotes the *effective* number of lifetime hours consumed. POH is a measure of a given core's lifetime degradation, and differs from the total number of hours for which the core operates, due to operating conditions, e.g., frequency and temperature;
- *nominal temperature* is the temperature at which the MTTF degradation of a core is the same as the number of hours it is active;
- *nominal frequency* is the frequency at which the temperature of a core attains its nominal value;
- *overdrive frequency* is the frequency due to overclocking the cores;[1]
- *acceleration factor* (AF) denotes the increased MTTF degradation due to operating at higher temperatures [27] [32]. AF is the ratio of original MTTF (at nominal temperature) to actual MTTF due to operating at a higher than nominal temperature.[2]

To meet performance and throughput requirements, cores operate at overdrive frequencies, and hence at higher than nominal temperatures. Thus, overdrive modes can result in cores' actual MTTFs becoming significantly smaller than original MTTFs. MTTF degradation can lead to two challenges. (1) All the cores in a multicore system can fail even before all assigned tasks are completed. A common strategy is to dynamically adapt overdrive frequencies so that all tasks are completed within the system's lifetime [8] [15]. (2) To reduce MTTF degradation, overdrive frequencies must be reduced and can violate a minimum "acceptable performance" requirement for tasks. We use Black's Equation [3] to calculate the MTTF due to electromigration. We consider a core to be reliable as long as the POH of the core is $\leq$ MTTF.

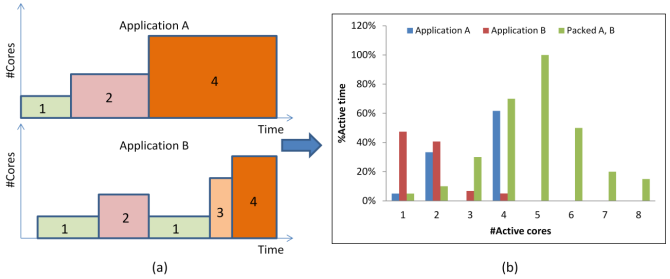

Fig. 1. Core usage profiles of (a) individual applications A and B, and (b) after scheduler packs execution using eight cores. B starts after A. The numbers in the colored boxes refer to the number of cores active.

[1]At overdrive frequencies, the core's temperature is higher than nominal.

[2]$AF = 1$ when a core is active at its nominal temperature, and $AF > 1$ at higher than nominal temperatures. This captures the well-known acceleration of wearout with higher operating temperature [27].

In this work, we formulate and solve a new *maximum-value, reliability-constrained overdrive frequencies* (MVRCOF) optimization problem that, unlike existing works, **guarantees prescribed levels of "acceptable performance" and "acceptable throughput"**. To the best of our knowledge, ours is the first approach that guarantees minimum performance and throughput requirements under reliability constraints. Our MVRCOF formulation maximizes the value (or, the advantage) of operating active cores at overdrive frequencies. The optimization is performed *offline* and is subject to four types of constraints: a lower bound on lifetime of each core, completion of all tasks within the system's lifetime, and upper bounds on instantaneous power and temperature. We develop a solver for the MVRCOF problem that determines the duration each combination should remain active so that all cores have balanced wearout. If no feasible solution exists, the scheduler may be notified to modify the task profile. To find the optimal solution, we perform exhaustive search over all overdrive frequencies within upper and lower bounds and all combinations of simultaneously active cores.[3] To make our flow scalable and efficient, we perform a *one-time characterization* of temperature and wearout for each combination of active cores, at each overdrive frequency. To our understanding, the MVRCOF solution is suitable for task migration in datacenters and other multicore systems [23].

Our contributions are the following.

- We propose a new MVRCOF formulation to maximize the value of operating multiple cores in overdrive frequencies under reliability constraints and given weights (relative importance) for overdrive and nominal frequencies in different modes.
- Our formulation guarantees satisfaction of prescribed lower bound constraints on both "acceptable performance" and "acceptable throughput" across all combinations ($C(N,m)$) of active cores. (m is the number of active cores, $m = 1,2,...,N$). To the best of our knowledge, we are the first to make minimum performance and throughput guarantees under electromigration reliability constraints.
- We determine optimal overdrive frequencies for each m out of N available cores. These frequencies can be maintained throughout the lifetime of the multicore system.[4]
- We develop both exhaustive (discretized) search for the optimal solution, as well as an approximate heuristic. In practice, our heuristic is within 3.3% of optimal across multiple testcases and converges up to $\sim$10$\times$ faster than the exhaustive search.[5]
- Our approaches determine the execution times of each combination of active cores; this can be used by OS schedulers to assign tasks to cores while maintaining required MTTF for all cores. (As we discuss below, this implies balanced wearout of cores.) Although we validate our solutions using four, six and eight cores, our method can be applied to systems with more than eight cores.
- We empirically demonstrate that our optimal solutions improve the objective function value by between 2.2% and 17.4% when compared to the existing reliability-aware task scheduling approaches of [4] and [15].

The remainder of this paper is organized as follows. Section II describes related work. Section III establishes notation and formulates our optimization problem. Section IV details our optimal, heuristic and baseline flows, Section V describes our methodology to generate testcases, Section VI describes validation of our flows and reports results, and Section VII outlines future work and concludes the paper.

II. Related Work

We taxonomize prior works on task scheduling for multicore systems as reliability-constrained (RC) or non-reliability-constrained (NRC). RC task scheduling policies can be further classified as those that make system "lifetime guarantees" (LG) and those that make "no lifetime guarantees" (NLG). Existing RC-LG policies apply (1) dynamic power management (DPM), (2) dynamic thermal management (DTM), or (3) dynamic reliability management (DRM). Such works are "performance-guaranteeing" (PG) if they guarantee lower bounds on "acceptable performance". Table I classifies existing works and our work according to the foregoing taxonomy.

TABLE I
Classification of existing works and our work.

Work	NRC	RC				
		NLG	LG			
			DPM	DTM	DRM	PG
Reiss et al. [13]	✓					
Karpuzcu et al. [9]		✓				
Mihic et al. [12]			✓			
Rosing et al. [15]			✓			
Rong et al. [14]			✓			
Coskun et al. [4] and [5]				✓		
Srinivasan et al. [18]					✓	
Karl et al. [8]					✓	
Our work					✓	✓

NRC and RC-NLG policies

Reiss et al. [13] present analysis of NRC task scheduling policies in Google datacenters. Each core maximizes throughput by operating at its worst-case temperature. The "BubbleWrap" work of Karpuzcu et al. [9] proposes a *DVSAM-Perf* policy to maximize performance in the presence of delay degradation due to bias temperature instability. DVSAM-Perf is an example of RC-NLG policy. *We demonstrate below that NRC and RC-NLG policies do not guarantee that all scheduled tasks will complete execution within the lifetime of the system.*

RC-LG policies

Mihic et al. [12] and Rosing et al. [15] perform detailed system modeling to devise a DPM policy to minimize total system energy subject to all tasks in a task graph completing execution within the multicore system's lifetime. Rong et al. [14] propose a DPM policy to minimize system energy subject to meeting all task deadlines within the multicore system's lifetime. Both these policies adjust voltage/frequency settings of a core to ensure all tasks complete execution within the system's lifetime.

Coskun et al. [4] propose five DTM policies to minimize thermal hotspots subject to completion of all tasks in a task graph while the multicore system meets its lifetime. In [5], Coskun et al. devise four DTM task scheduling and migration policies. They propose that tasks should be scheduled on cores

[3]We implicitly consider that all possible overdrive frequencies are feasible for a given block implementation. The relationship between feasible range of overdrive frequencies and the area, mix of different-Vt cells in implementation, etc. of a block is an open direction for future work.

[4]This implicitly assumes that each core in the system can adapt (e.g., by supply voltage scaling) to aging in order to maintain a given target (nominal or overdrive) operating frequency.

[5]Our method is based on offline simulations. Comparison of our results with measured data remains a direction for future work.

towards the periphery of the chip. However, in neither [4] nor [5] do the proposed policies guarantee any minimum frequency of a core.

Srinivasan et al. [18] develop *RAMP*, a reliability simulator, and propose a DRM policy to adjust voltage/frequency settings of cores to maximize the lifetime of the multicore system. Karl et al. [8] use a proportion-integral-derivative (PID) control system to determine the maximum voltage/frequency setting of a core to complete a given task. However, again, these policies do not guarantee any minimum frequency of a core.

Counterexamples for NRC, RC-NLG and RC-LG policies

We now demonstrate the suboptimality of existing policies using a simple counterexample. We consider a system with four cores and MTTF of seven years (= 61320h) for each core. The nominal frequency and temperature are respectively 1.5GHz and 358K, the maximum frequency is 3.0GHz, and the maximum temperature is 398K. We assume that the scheduler assigns tasks for each (m (= the number of active cores), nominal execution time, overdrive execution time) 3-tuple as follows: (1, 1000h, 3000h), (2, 2000h, 5000h), (3, 3000h, 8000h), and (4, 2000h, 5000h). Last, we assume that overdrive-mode tasks require a *minimum overdrive frequency* or "acceptable performance" of 1.8GHz.

Using *HotSpot* [17] as detailed in Section VI below, for the above instance optimal (discretized) overdrive frequencies can be found using exhaustive search for each m respectively as 2.85GHz, 2.3GHz, 1.8GHz and 1.8GHz.[6] However, NRC and RC-NLG policies will operate always at 3.0GHz and 398K, inducing an acceleration factor AF = 9.77 relative to the nominal operating temperature of 358K. Assuming execution of tasks are balanced across the four cores, Figures 2 and 3 respectively illustrate suboptimalities of NRC and RC-LG policies for a system with four cores, where each core has an initial lifetime of seven years (61320h) and the tasks listed in the previous paragraph are as assigned by the scheduler. The figures show how time progresses along with execution of nominal and overdrive tasks, starting with $m = 1$ and followed by $m = 2, 3$ and $m = 4$ for Figure 3. For example, for $m = 1$ in both figures, the duration of nominal tasks $E_{nom} =$ 1000h, each core executes 250h at a nominal frequency $f_{nom} =$ 1.5GHz, and the corresponding AF is 1.0, hence the value of power-on-hours, POH, is 250h$\times$1.0 = 250h for each core. All POH values are shown with a negative sign to indicate effective lifetime consumed. Further, for $m = 1$, the duration of overdrive tasks $E_{OD} =$ 3000h, each core executes 750h at an overdrive frequency $f_{OD} =$ 3.0GHz, and the corresponding AF is 9.77, hence the POH is 750h$\times$9.77 = 7372.5h.

With NRC policies, Figure 2 shows that the effective lifetime of each core at the end of $m = 3$ nominal tasks is 24947.5h ("Lifetime remaining" in the figure). This is because each core consumes 1220h + 24425h after executing 1000h in nominal and 2500h in overdrive modes for $m = 2$, and 3150h after executing 2250h in nominal mode for $m = 3$. For balanced execution of $m = 3$ overdrive tasks, each core requires 6000h of execution; this requires the lifetime of each core to be 6000h$\times$9.77 = 58620h, which exceeds the effective lifetime of 24947.5h. (Moreover, this is without resourcing any tasks for $m = 4$.) This simple example shows that *existing NRC policies are not optimal, in that they cannot guarantee that cores will complete all tasks before failing*.

[6]We describe in Section IV the use of *HotSpot* [17] to simulate temperatures (from which wearout acceleration factors (AFs) are derived) for different frequencies and combinations of active cores.

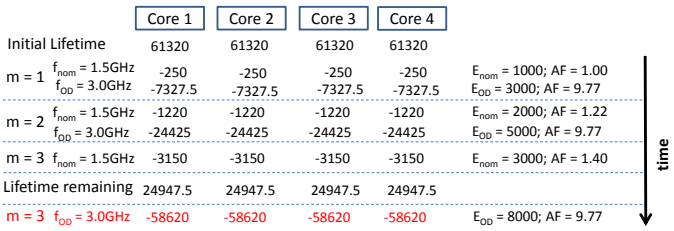

Fig. 2. Suboptimality of NRC policies. E_{nom} and E_{OD} respectively indicate the nominal and overdrive execution times. Lifetime of all cores are completely used up by the end of nominal mode execution in $m = 3$. Thus, tasks requiring $m = 3$ overdrive mode execution, and all tasks requiring $m = 4$, cannot be completed. All times are in hours.

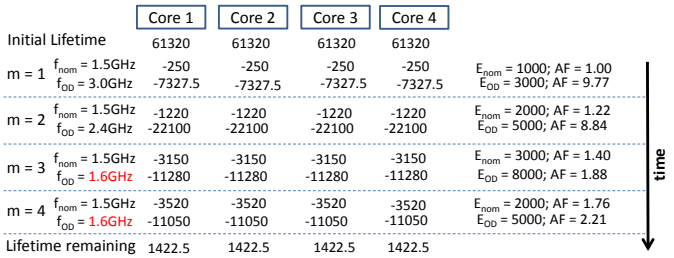

Fig. 3. Suboptimality of RC-LG policies. E_{nom} and E_{OD} respectively indicate the nominal and overdrive execution times. Tasks requiring $m = 3$ and $m = 4$ overdrive mode execution must operate at an overdrive frequency of 1.6GHz instead of the required 1.8GHz. All times are in hours.

The conclusion from our counterexample: Operating at the maximum frequency for all overdrive tasks is not the optimal strategy, as it ignores lifetimes of cores.

With RC-LG policies, Figure 3 shows that the POH of each core is 250h + 7327.5h + 1220h + 22100h after executing 250h in nominal mode and 750h in overdrive mode for $m = 1$, and 1000h in nominal mode and 2500h in overdrive mode for $m = 2$. Cores can be run at the maximum value of 3.0GHz for $m = 1$, but cannot maintain this frequency for $m = 2, 3, 4$ if the required tasks are to be completed while maintaining the system's lifetime ("Lifetime remaining" is 1422.5h ≥ 0 in the figure). Executing the frequency assignment approach of RC-LG policies, we experimentally determine that overdrive frequency for $m = 3, 4$ can be at most 1.6GHz, which is less than the minimum required overdrive frequency of 1.8GHz as illustrated in Figure 3. The positive value of "Lifetime remaining" cannot be utilized to increase any of the overdrive frequencies further, as this violates lifetime requirements. Thus, our simple example shows that *existing RC-LG policies are not optimal, in that they cannot guarantee lower bounds on "acceptable performance"*. The conclusion from our counterexample: Executing overdrive tasks at the maximum frequency and decreasing the frequency to meet lifetime is not the optimal strategy, as it does not consider duration of all overdrive tasks across all combinations of active cores.

III. Problem Statement

We now formulate our MVRCOF optimization problem to maximize the value[7] of a set of overdrive frequencies, such that no core wears out before its prescribed lifetime requirement. We assume that the scheduler packs tasks from multiple applications and provides a final operating schedule as conceptually illustrated in Figure 4. Further, we do not consider the cost of task migration.

[7]By "value", we refer to the advantage of operating at overdrive frequencies. We maximize the advantage by maximizing the overdrive frequencies.

There are two kinds of inputs: (1) system description, and (2) task description. Table II lists the parameters used in our formulation and solution. Figure 5 shows the inputs and outputs of the MVRCOF problem graphically. We assume that the values for the system description parameters are given. The operating system scheduler provides values for the task description parameters depending on application demands for performance and the number of cores used. For example, applications that benefit from frequency overdrive are accounted for in $E_{OD,m}$ and $w_{OD,m}$ parameters, whereas applications that do not benefit from frequency overdrive are accounted for in $E_{nom,m}$, $f_{nom,m}$ and $w_{nom,m}$ parameters.

We define "acceptable performance" as $1.3 \times f_{nom,m}$, based on [28]; this does not compromise the generality of our conclusions. Further, we define "acceptable throughput" as the ability to complete all tasks within the multicore system's given lifetime.

TABLE II
PARAMETERS USED IN PROBLEM FORMULATION AND SOLUTION.

Parameter	Description	Type
N	#symmetric cores indexed by $i = 1, 2, \cdots, N$	System
P_{max}	maximum power consumption of any core	System
f_{max}	maximum frequency of any core	System
T_{nom}	core temperature at nominal frequency	System
T_{max}	maximum die temperature	System
$MTTF$	lifetime of each core	System
E_a	metal activation energy (0.7eV [7])	Physical
k	Boltzmann's constant (8.62×10^{-5}eV/K)	Physical
m	#simultaneously running, or *active*, cores, $1 \leq m \leq N$ (at the same nominal or overdrive frequency)	Task
l	operating mode of any core ($\{$nom, OD$\}$)	Task
$f_{nom,m}$	nominal frequency of m cores	Task
$w_{l,m}$	weights in objective function of achieved frequencies, $w_{l,m} \geq 0$, $\forall l, m$	Task
$E_{l,m}$	execution time in operating mode l with m cores	Task
P_i	power of the i^{th} core at any time	Variable
T_m	temperature of the die with m active cores at any time	Variable
$t_{j,m,l}$	execution times for the j^{th} combination of m cores	Variable
$b_{i,j,m}$	binary variable (1 when core i is present in the j^{th} combination of m cores, 0 otherwise)	Variable
$MTTF_i$	effective lifetime of the i^{th} core after it has been active	Variable
$POH_{l,i}$	POH of the i^{th} core in operating mode l	Variable
$AF_{i,j,m,l}$	AF of the i^{th} core in the j^{th} combination of m cores in operating mode l	Variable
$T_{i,j,m,l}$	temperature of the i^{th} core in the j^{th} combination of m cores in operating mode l	Variable
Δ_{fOD}	discrete amount by which overdrive frequency is increased	Variable
$f_{OD,m}$	overdrive frequency for m cores	Output
$v_{j,m,l}$	% total execution time for the j^{th} combination of m cores in operating mode l	Output
$u_{i,l}$	% of lifetime during which the i^{th} core is active in operating mode l	Output

MVRCOF Formulation

Given the above-described inputs, the problem to maximize the value of a set of overdrive frequencies is formally expressed as

$$\text{maximize} \sum_{m=1}^{N} \left(w_{OD,m} \cdot f_{OD,m} \cdot E_{OD,m} + w_{nom,m} \cdot f_{nom,m} \cdot E_{nom,m} \right)$$

subject to

$$\forall m, 1.3 \times f_{nom,m} \leq f_{OD,m} \leq f_{max} \quad (1)$$
$$\forall i, MTTF_i \leq MTTF \quad (2)$$
$$\forall i, P_i \leq P_{max} \quad (3)$$
$$\forall m, T_m \leq T_{max} \quad (4)$$

where

- Constraint (1) ensures lower bounds of "acceptable performance" is at least 30% more than $f_{nom,m}$ as well as restricts $f_{OD,m}$ values to be within the maximum frequency of the system.
- Constraint (2) ensures "acceptable throughput", i.e., tasks are completed within the system's lifetime.
- Constraints (3) and (4) ensure that the power of each core, P_i, and the temperature of the die with m active cores, T_m, are within maximum power and temperature upper bounds.[8] $MTTF_i$ is calculated using Equations (5) – (7) [27].

We use the execution times in the objective function to determine the duration over which cores in mode m execute at $f_{OD,m}$.

$$AF_{i,j,m,l} = exp\left(\frac{E_a}{k} \cdot \left[\frac{1}{T_{nom}} - \frac{1}{T_{i,j,m,l}}\right]\right) \quad (5)$$

$$POH_{l,i} = \sum_{m=1}^{N} \sum_{j=1}^{C(N,m)} t_{j,m,l} \cdot b_{i,j,m} \cdot AF_{i,j,m,OD} \quad (6)$$

$$MTTF_i = \sum_{l=\{nom,OD\}} POH_{l,i} \quad (7)$$

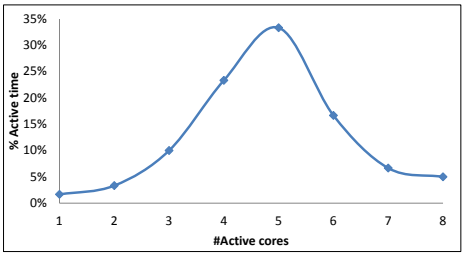

Fig. 4. Core usage profile from scheduler after packing tasks from multiple applications. The task profile represents typical datacenter workload from [2].

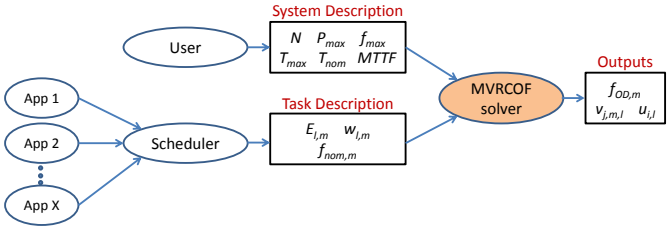

Fig. 5. Graphical representation of inputs and outputs of the MVRCOF problem.

IV. Optimal (Discretized) Solution Flow

We now present our flow to solve the MVRCOF optimization problem. We also present our heuristic flow and a baseline flow to compare RC-LG policies. To work around potential nonlinear constraints, we perform exhaustive search of $f_{OD,m}$ across all m by increasing values of $f_{OD,m}$ from $1.3 \times f_{nom,m}$ to f_{max} by Δ_{fOD}. We perform one-time characterization (∼9.5h on an Intel Xeon E5-2640 2.5GHz system) of temperature and AF for all discretized values of $f_{OD,m}$ for each combination of m out of N cores and generate a lookup table (LUT)[9] as follows.

1) Perform post-layout gate-level power simulation of a single core. We set the switching activity factor at

[8] P_i is obtained using post-layout power simulation of a core at $f_{nom,m}$ or $f_{OD,m}$. T_m is the die temperature obtained from *HotSpot* [17] simulations.

[9] The reader will notice that we are assuming that temperature is restored to its nominal value before task execution begins on each successive combination of active cores. We believe this assumption is reasonable as long as task assignments have relatively long durations, and given that, in a day, cores are utilized ∼60% of the time (i.e., system is idle for ∼9h) [13]. Our experiments indicate that the time for temperature to drop from 125^{o}C to 85^{o}C is ∼10s, and this is consistent with the results of [22]. Achieving a solution that is "history-aware" remains a direction for future work.

primary inputs in the nominal mode as 0.11 [29] and in the overdrive mode as 0.3 [16]; We also set the clock period as the inverse of the overdrive frequency in the SDC file [1].

2) Perform temperature simulation with power using *HotSpot* [17].[10]
3) Create a LUT entry for the i^{th} core if it is present in the j^{th} combination of m cores as ($f_{OD,m}$, temperature, AF). For each core and for each $f_{OD,m}$, there are at most $N \times C(N,m)$ entries. Figure 6 illustrates an example of a subset of a LUT for a system with three cores.
4) Increase $f_{OD,m}$ by Δ_{fOD} as long as $f_{OD,m} + \Delta_{fOD} \leq f_{max}$. Go to Step (1).

Core	(m, j)	$f_{OD,m}$ (GHz)	Temp (K)	AF
1	(1,1)	3.0	398	9.77
		2.95	392.5	7.34
		2.90	396.9	5.34
		• • •	• • •	• •
		1.80	362.0	1.29
• • •	• • •	• • •	• • •	• •
2	(2,1)	3.0	398	9.77
		2.95	393.1	7.58
		2.90	388.4	5.90
		• • •	• • •	• •
		1.80	367.6	1.82
• • •	• • •	• • •	• • •	• •
3	(3,1)	3.0	398	9.77
		2.95	397.5	9.53
		2.90	396.9	9.24
		• • •	• • •	• •
		1.80	377.4	3.42

Fig. 6. Example of a subset of a lookup table (LUT) for a three-core system.

Algorithm 1 shows details of our flow to determine the optimal values of $f_{OD,m}$. $f(.)$ in the algorithm indicates that the output value is a function of the input parameters $(.)$. Lines 1–18 initialize MTTF of all cores, overdrive frequencies across all m, the variable $b_{i,j,m}$ for each core, and the variables that store the optimal value of the objective function and the overdrive frequencies. Lines 19–46 contain the loops for the exhaustive search. Lines 22–34 comprise the innermost loop which, given an overdrive frequency, determines if all cores in all combinations can complete execution within the prescribed lifetime. If a core is present in a combination, Line 25 obtains its AF from the LUT, and Line 26 executes the following linear program (LP), which calculates $t_{j,m,nom}$ and $t_{j,m,OD}$ to balance wearout across all cores. Formally, the LP is expressed as

maximize c

subject to

$$\forall i, c \leq \sum_{l=\{nom,OD\}} POH_{l,i} \quad (8)$$

$$\forall i, \sum_{l=\{nom,OD\}} POH_{l,i} \leq MTTF \quad (9)$$

$$\sum_{j=1,\ l\in\{nom,OD\}}^{C(N,m)} t_{j,m,l} = E_{l,m} \quad (10)$$

where

- Constraint (8) ensures c is the minimum POH across all cores.
- Constraint (9) ensures that all tasks are completed within the multicore system's lifetime.
- Constraint (10) ensures that the sum of nominal and overdrive execution times across all combinations of m active cores meet the respective required execution times. $POH_{l,i}$ is obtained from Equation (6).

If an optimal solution exists for the LP, Line 27 calculates POH for the core using Equation (7) and Line 28 updates the core's effective MTTF, $MTTF_i$, based on $t_{j,m,l}$ and $E_{l,m}$ values. If all cores can complete execution within their lifetimes, Line 36 calculates the value of the objective function. Lines 38–41 store the overdrive frequencies and largest value of the objective function obtained up through the iteration *iter* (We compare *iter* across our testcases in Section VI). Lines 42–43 increments $f_{OD,m}$ by Δ_{fOD}. When all values of frequencies across all m have been tried, Lines 47–51 output the optimal solution or report infeasibility if no feasible solution exists.

Figure 7 shows our complete flow from generating the LUT to using Algorithm 1 and the LP to achieving optimal values of the objective function, $f_{OD,m}$ and $t_{j,m,l}$. We can now determine the optimal values of $v_{j,m,l}$ and $u_{i,l}$ from the values of $t_{j,m,l}$. Exhaustive search across all discretized values of $f_{OD,m}$ across all cores present in the j^{th} combinations of m achieves a discretized optimal solution. The time complexity of Algorithm 1 is $O(N^2 \cdot C(N,m) \cdot f_{steps})$, where $f_{steps} = (f_{max} - 1.3 \times f_{nom,m}) / \Delta_{fOD}$. By construction, our optimal flow guarantees "acceptable performance" because we search for optimal values of $f_{OD,m}$ that are at least 30% above the nominal frequency. The optimal flow also guarantees "acceptable throughput" because our LP always guarantees balanced wearout and Lines 29–31 of Algorithm 1 does not allow cores to operate at frequencies that cannot guarantee task completion within a core's effective lifetime.

Heuristic flow. We also develop a heuristic flow to solve the optimization by maximizing overdrive frequencies for each m in the order of decreasing $w_{OD,m} \times E_{OD,m}$. This is because maximizing $f_{OD,m}$ for the largest $w_{OD,m} \times E_{OD,m}$ causes large improvement in the objective function value. The flow is similar to the exhaustive search; however, rather than explore all modes in their numerical order, we start with the mode that has the largest product of $w_{OD,m} \times E_{OD,m}$ and obtain the maximum $f_{OD,m}$. The flow uses Lines 20–45 of Algorithm 1 to determine the maximum $f_{OD,m}$ for each m in decreasing order of $w_{OD,m} \times E_{OD,m}$. We compare the *iter* value (i.e., the number of iterations) from Line 43 between the optimal and heuristic solutions in Section VI.

Baseline (RC-LG) flow. To compare the solutions of our optimal and existing RC-LG policies, we describe a baseline flow in which we use frequency assignment approach in RC-LG policies. RC-LG policies assign the maximum frequency of a combination m as long as power, thermal upper bound and lifetime constraints are met [9] [13] [15]. The frequency is dynamically changed to meet lifetime requirements. The baseline flow is as follows.

- Choose core(s) with the maximum MTTF for execution and whose $MTTF_i \geq$ the required (nominal or overdrive) execution times.
- Find the maximum $f_{OD,m}$ subject to power, thermal upper bound constraints and $MTTF_i$ (when multiple cores are active, use the minimum $MTTF_i, \forall i$).[11]

[10] Thermal parameters for *HotSpot* [17] simulations are calibrated with Intel Xeon processors.

[11] We perform exhaustive search for the maximum $f_{OD,m}$, starting from f_{max}. If a frequency can complete current and future tasks within the lifetime of all cores, then set this as the maximum $f_{OD,m}$. Otherwise decrease in 50MHz steps and until $f_{OD,m} = f_{nom,m}$.

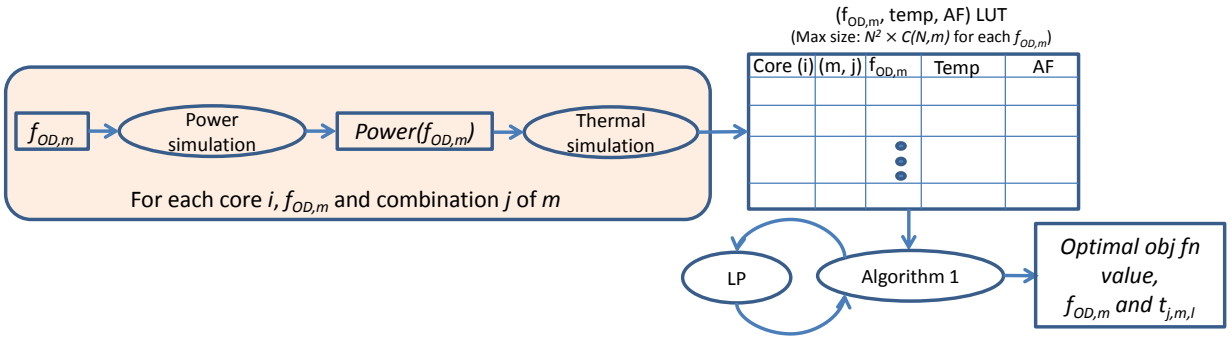

Fig. 7. Flow to obtain optimum solution. Algorithm 1 uses the lookup table (LUT) and repeatedly calls the LP solver.

V. Testcase Generation

To construct different testcases, we modify (1) $w_{nom,m}$ and $w_{OD,m}$, (2) $E_{nom,m}$ and $E_{OD,m}$, and (3) $f_{nom,m}$ values. While we make certain choices of parameter values in our experiments, these choices do not compromise the generality of our method and conclusions. Let the ratio $r_m = \frac{w_{nom,m}}{w_{OD,m}}$ be a random variable chosen uniformly in the interval [0.1, 10]. Then, $w_{OD,m} = \frac{1}{1+r_m}$ and $w_{nom,m} = \frac{r_m}{1+r_m}$.[12] For each m, we generate a value of r_m to obtain $w_{nom,m}$ and $w_{OD,m}$.

Figure 4 illustrates an example of the scheduler-determined total execution times in an eight-core system when $m = 1,2,...,8$ cores, respectively, are active. To model a similar skewed Gaussian distribution of $E_{tot,m} = E_{nom,m} + E_{OD,m}$ for the random variable m with mean μ, standard deviation σ and probability density function $f(m|\mu,\sigma)$, we assume the following.

- All cores begin execution with the same MTTF = {7, 10} years to represent typical server cores [19].
- $\mu = \lfloor \frac{N}{2} \rfloor + 1, \forall N$; μ is an integer as it equals m with the largest $E_{tot,m}$.
- $3\sigma = \max\left(N - \left(\lfloor \frac{N}{2} \rfloor + 1\right), \lfloor \frac{N}{2} \rfloor\right) = \lfloor \frac{N}{2} \rfloor$ because the execution times with fewer or more active cores are less than the ones which use approximately half of N [13]. σ does not need to be an integer because we only want to make sure that $|m - \mu| \leq 3\sigma$. So, $\sigma = \frac{1}{3} \cdot \lfloor \frac{N}{2} \rfloor, \forall N$.
- U is a uniformly random number in [0.35, 0.5], denoting the peak core utilization of Amazon EC2 datacenters, following core utilization data in [11]. Then, we have that $E_{nom,\mu} + E_{OD,\mu} = MTTF \times U$, and $E_{tot,m} = E_{tot,\mu} \times f(m|\mu,\sigma)$ when $m \neq \mu$.
- $r_{e,m}$ is a uniformly random number in [0.1, 0.5] following task priority information in [11], which denotes the ratio of $E_{nom,m}$ to $E_{nom,m} + E_{OD,m}$. Then, we have that $E_{nom,m} = r_{e,m} \times E_{tot,m}$, and $E_{OD,m} = E_{tot,m} - E_{nom,m}$.
- $f_{nom,m}$ takes on values between [1.5, 2.0]GHz in steps of 50MHz so that the maximum frequency is $\leq$ 3.0GHz [24].

Decomposition of task trace

We decompose packed tasks from Figure 4 to resemble realistic datacenter traces with the following assumptions. (1) For a given m, all cores execute either nominal or overdrive tasks. (2) In a day, nominal tasks run for ~20% (5h) and overdrive tasks run for ~40% (10h) [2] [10] [13]. The cores are idle for the remaining 9h. (3) Tasks are nonpreemptive. (4) Overdrive tasks are scheduled before nominal tasks.

We obtain $E_{nom,m}$ and $E_{OD,m}$ from above and calculate the per-day nominal and overdrive tasks for each m so that they are in the ratio of their total execution times.[13] Therefore, in a day, across all m, the sum of nominal tasks in a day adds up to 5h and the sum of overdrive tasks adds up to 10h. We now generate a trace by sequencing tasks as {overdrive, nominal} for each day such that execution times of all nominal and overdrive tasks respectively add to $E_{nom,m}$ and $E_{OD,m}$. The sequencing gives the notion that overdrive tasks must be completed before nominal tasks. We use these traces to validate our solutions, as we describe next.

VI. Validation and Results

To simulate a design to fill $25mm^2$, $38mm^2$ and $51mm^2$ die at ~70% utilization respectively with four, six, and eight vector processor-like cores, we create a floorplan file for *Hotspot* [17] that instantiates each processor core as 72× *jpeg_encoder* [31] cores. To obtain area and power for 72× *jpeg_encoder*, we perform synthesis, place-and-route, and power analysis of a single *jpeg_encoder* core and scale the area and power values by 72. We use 45nm commercial foundry libraries, synthesize the *jpeg_encoder* design using Synopsys *Design Compiler vG-2012.06* [26] and perform place-and-route using Cadence *SOC Encounter vEDI10.1* [25] with clock period set to 0.5ns. The post-layout netlist for a single *jpeg_encoder* core contains ~38K instances and the area of standard-cells is $0.06mm^2$. We then perform power analysis by varying supply voltage from 0.8V to 1.2V in steps of 10mV, varying the frequency (transcribed to clock period in the SDC [1] file) from 1.5GHz to 3.0GHz in steps of 50MHz. [14] We develop a solver using custom *Tcl* scripts to implement Algorithm 1 and the heuristic flow, and solve all our generated LP instances using *lp_solve* [30]. For all m, we set $f_{nom,m}$ = 1.5GHz and the minimum $f_{OD,m}$ to be 1.8GHz (i.e., 20% greater than the nominal frequency, following [28]). We set P_{max} to 30W [28] and T_{max} to 398K [27].

We report experimental setup and results with four[15], six and eight cores, and MTTF of each core set to seven years. Table III shows the $E_{nom,m}$, $E_{OD,m}$, $f_{nom,m}$, $w_{nom,m}$, and $w_{OD,m}$ values for each testcase. We name our testcases as N-testcase# where N indicates the number of cores. Testcase 6-II uses twice the $E_{OD,m}$ values from Testcase 6-I. Testcases 4-I and 4-II use different weights for $m = 1,4$. Our results enable (1) validation of our discretized optimal solutions, and (2) comparison of solutions from optimal, heuristic, and baseline (RC-LG) flows.

[12] $0.1 \leq r_m \leq 10$ allows us to obtain extreme values such as, $w_{nom,m} = 0.1$ and $w_{OD,m} = 0.9$, and $w_{OD,m} = 0.1$ and $w_{nom,m} = 0.9$.

[13] For example, if $E_{nom,1}$ is 10% of total execution time of nominal tasks across all m, then in a day the nominal task for $m = 1$ is 10% of 5h.

[14] We assume that for a given block implementation all the possible f_{OD} values are feasible by construction, i.e., the implementation ensures no timing violations when the frequency is set to f_{max}.

[15] The area of four cores with 70% utilization is $(4 \times 72 \times 0.06)/0.7 \approx 25mm^2$.

Alg. 1 Determination of optimal $f_{OD,m}$

Require: N, f_{max}, $f_{nom,m}$, $E_{nom,m}$, $E_{OD,m}$, $w_{nom,m}$, $w_{OD,m}$, ($f_{OD,m}$, temp) LUT, $MTTF$

for all $i = 1,2,...,N$ **do**
  $MTTF_i \leftarrow MTTF$
end for
for all $m = 1,2,...,N$ **do**
  $f_{OD,m} \leftarrow 1.3 \times f_{nom,m}$
end for
for all $m = 1,2,...,N$ **do**
  for all $j = 1,2,...,C(N,m)$ **do**
    for all $i = 1,2,...,N$ **do**
      if core $i \in$ combination j **then**
        $b_{i,j,m} \leftarrow 1$
      else
        $b_{i,j,m} \leftarrow 0$
      end if
    end for
  end for
end for
$iter \leftarrow 1$; $Bestval \leftarrow 0$; $Bestfod[] \leftarrow 0$
for all $m = 1,2,...,N$ **do**
  while $f_{OD,m} \leq f_{max}$ **do**
    $Val_{iter} \leftarrow 0$; $term \leftarrow 0$
    for all $j = 1,2,...,C(N,m)$ **do**
      for all $i = 1,2,...,N$ **do**
        if $b_{i,j,m} > 0$ **then**
          Get AF: $AF_{i,j,m,OD} \leftarrow f(f_{OD,m}, LUT, i, j, m)$
          Solve LP: $t_{j,m,l} \leftarrow f(AF_{i,j,m,l}, MTTF)$
          Calculate POH: $POH_i \leftarrow f(t_{j,m,l}, E_{nom,m}, E_{OD,m})$
          Update MTTF: $MTTF_i \leftarrow MTTF_i - POH_i$
          if $MTTF_i \leq 0$ **then**
            $term \leftarrow 1$
          end if
        end if
      end for
    end for
    if $term > 0$ **then**
      Calculate objective function value:
      $Val_{iter} \leftarrow w_{nom,m} \cdot f_{nom,m} \cdot E_{nom,m} + w_{OD,m} \cdot f_{OD,m} \cdot E_{OD,m}$
      if $Val_{iter} > Bestval$ **then**
        $Bestval \leftarrow Val_{iter}$
        $Bestfod[m] \leftarrow f_{OD,m}$
      end if
      $f_{OD,m} \leftarrow f_{OD,m} + \Delta_{fOD}$
      $iter \leftarrow iter + 1$
    end if
  end while
end for
if $Bestval > 0$ **then**
  Output $Bestval$ and $Bestfod[1,2,...,N]$
else
  Output no feasible solution exists
end if

A. Solution validations

To validate our solutions, we confirm the sensibility of how varying Δ_{fOD}, $w_{nom,m}$, and $w_{OD,m}$ affect the value of the objective function. Then, for Testcase 8-I, we present the optimal values of $v_{j,m,l}$ and $u_{i,l}$ which can potentially be used by the operating system scheduler for task migration.[16]

TABLE III
EXPERIMENT SETUP. EACH ROW SHOWS EXECUTION TIMES AND WEIGHTS FOR TWO VALUES OF m.

Testcase	m	$E_{nom,m}$ (Kh)	$E_{OD,m}$ (Kh)	$w_{nom,m}$	$w_{OD,m}$	$f_{nom,m}$ (GHz)
4-I	1, 2 3, 4	1, 2 3, 2	3, 5 8, 5	0.5, 0.3 0.2, 0.4	0.5, 0.7 0.8, 0.6	1.5
4-II	1, 2 3, 4	1, 2 3, 2	3, 5 8, 5	0.2, 0.3 0.2, 0.5	0.8, 0.7 0.8, 0.5	1.5
4-III	1, 2 3, 4	1, 2 3, 2	3, 5 8, 5	0.1, 0.2 0.9, 0.8	0.9, 0.8 0.1, 0.2	1.5
4-IV	1, 2 3, 4	1, 2 3, 2	3, 5 8, 5	0.1, 0.2 0.1, 0.2	0.9, 0.8 0.9, 0.8	1.5
4-V	1, 2 3, 4	1, 2 3, 2	3, 5 8, 5	0.9, 0.8 0.1, 0.2	0.1, 0.2 0.9, 0.8	1.5
6-I	1, 2 3, 4 5, 6	0.5, 0.7 1.2, 1.6 1.3, 0.9	1, 2 3.5, 5 4, 2.5	0.5, 0.45 0.3, 0.2 0.4, 0.4	0.5, 0.55 0.7, 0.8 0.6, 0.6	1.5
6-II	= 6-I	= 6-I	= 2×6-I	= 6-I	= 6-I	= 6-I
8-I	1, 2 3, 4 5, 6 7, 8	0.55, 0.45 0.65, 0.8 0.77, 0.73 0.74, 0.59	0.6, 0.8 0.9, 1.1 1, 0.95 0.82, 0.73	0.5, 0.45 0.3, 0.2 0.3, 0.35 0.4, 0.4	0.5, 0.55 0.7, 0.8 0.7, 0.65 0.6, 0.6	1.5

Impact of Δ_{fOD}. Table IV shows the optimal values of $f_{OD,m}$ and the value of our objective function for different values of Δ_{fOD} from our optimal flow. The objective function value varies by less than 0.3% when Δ_{fOD} varies from 200MHz to 50MHz. However, when $\Delta_{fOD} = 50$MHz, the solution requires $\sim$13$\times$ the number of iterations to converge.[17] Smaller values of Δ_{fOD} achieves higher values of the objective function, as expected. *Because the MVRCOF solver runs offline and runtime is not significant concern for our testcases, we use $\Delta_{fOD} = 50MHz$ to achieve higher value of the objective function.*

Impact of $w_{nom,m}$ and $w_{OD,m}$. Table V compares solutions of all testcases with four cores, 4-I – 4-V, from our optimal flow. The value of the objective function depends on the values of $E_{nom,m} \times w_{nom,m}$ and $E_{OD,m} \times w_{OD,m}$. In our experiments, we set $f_{nom,m} = 1.5$GHz for all m, so the value depends on $E_{OD,m} \times w_{OD,m}$. At smaller values of m, $f_{OD,m}$ is larger as compared to larger values of m because as m increases, overdrive frequency is limited by the maximum die temperature T_{max}. Specifically, because $E_{OD,m} \times w_{OD,m}$ is higher in 4-I – 4-IV for $m = 1,2,3$ than in 4-V, testcases 4-I – 4-IV yield a higher value of the objective function than 4-V. Testcase 4-IV has the highest value of $E_{OD,3} \times w_{OD,3}$, so it yields the highest value of the objective function even though $f_{OD,3}$ for 4-IV is the same as 4-III. *Therefore, larger values of $f_{OD,m}$ are achieved for smaller values of m and larger value of the objective function is achieved when $E_{OD,m} \times w_{OD,m}$ is large for these corresponding values of m.*

TABLE IV
OBJ FN VALUE VS. Δ_{fOD}. HIGHER OBJ FN VALUE IS BETTER.

Testcase	m	Δ_{fOD}	$f_{OD,m}$ (GHz)	Obj fn value	$iter$
4-I	1, 2 3, 4	200MHz	3.0, 2.2 1.8, 1.8	32870	21
4-I	1, 2 3, 4	50MHz	2.85, 2.3 1.8, 1.8	32995	271
6-I	1, 2, 3 4, 5, 6	100MHz	2.8, 2.9, 2.6 1.8, 1.8, 1.8	28367.5	1500
6-I	1, 2, 3 4, 5, 6	50MHz	2.85, 2.9, 2.6 1.8, 1.8, 1.8	28392.5	10572
8-I	1, 2, 3 4, 5, 6 7, 8	100MHz, 50MHz	2.9, 2.9, 2.8 1.8, 1.8, 1.8 1.8, 1.8	12316.0, 12316.0	1845, 11254

[16] Testcase 6-II is infeasible because no value of $f_{OD,m}$ with lower bound ("acceptable performance") set to 1.8GHz can complete the tasks within the system's lifetime.

[17] On a Intel Xeon E5-2640 2.5GHz system, each iteration executes roughly in 0.7s.

TABLE V
OBJ FN VALUE VS. WEIGHTS AT $\Delta_{fOD} = 50$MHz.

Testcase	m	$f_{OD,m}$ (GHz)	Obj fn value	$iter$
4-I	1, 2, 3, 4	2.85, 2.3, 1.8, 1.8	32995	271
4-II	1, 2, 3, 4	2.95, 2.25, 2.05, 1.8	36175	408
4-III	1, 2, 3, 4	2.95, 2.35, 2.05, 1.8	28005	424
4-IV	1, 2, 3, 4	2.95, 2.35, 2.05, 1.8	41125	424
4-V	1, 2, 3, 4	3.0, 2.3, 2.0, 1.8	29600	410

Task migration policy based on optimal values of $v_{j,m,l}$ and $u_{i,l}$ for Testcase 8-I. Based on notations and terminologies described in Section III, Table VI shows values of $v_{j,m,l}$ for each active combination j of $1 \leq m \leq 8$ active cores in both nominal and overdrive execution modes. For example, $v_{11,2,OD} = v_{17,2,OD} = 25.8$ for $C(N = 8, m = 2)$. Out of total lifetime of seven years, each core is active as follows: core 1 for 12.91%, core 2 for 10.29%, core 3 for 11.46%, core 4 for 11.58%, core 5 for 12.44%, core 6 for 10.05%, core 7 for 9.66% and core 8 for 13.44%. As expected, as they enjoy better heat removal, the cores towards the periphery of the die (e.g., cores 1, 4, 5, 8) are active for longer duration (higher percentages of chip lifetime) than other cores [4].[18] *The operating system scheduler can use the optimal values of $v_{j,m,l}$ and $u_{i,l}$ to determine how long each core should execute at nominal and overdrive frequencies for balanced wearout.* Using the linear programming approach to determine the execution time of each combination leads to the times being very imbalanced across combinations. As part of our future work, we will explore techniques such as geometric programming which may lead to more balanced solutions.

B. Comparisons of optimal, heuristic, and RC-LG solutions

We compare our optimal solutions with (1) our heuristic solutions and (2) baseline (RC-LG) solutions [4] and [15]. Figures 8–10 show the results of these comparisons. Figure 8 shows that for our testcases, the heuristic solutions are at most 3.3% worse than the optimal solutions, but can converge in up to 10× fewer iterations as shown in Figure 9 when compared to the number of iterations in Table IV. Moreover, our solutions guarantee that all tasks in each combination of m execute at $f_{OD,m} \geq$ 1.8GHz, i.e., the solutions meet "acceptable performance" requirements. In addition, we guarantee "acceptable throughput" because all tasks complete within the multicore system's lifetime. Figure 8 shows that the baseline method's solutions can be up to 17.4% below optimal, even as they execute around 27% of the overall overdrive execution time below the minimum required frequency of 1.8GHz as shown in Figure 10. In other words, the baseline solutions do not meet "acceptable performance" requirements. For Testcase 6-II, the baseline flow achieves a solution within lifetime constraints, but unfortunately executes 75.5% of the overall overdrive execution time at frequencies below 1.8GHz. Runtime versus accuracy tradeoffs would determine a user's choice between optimal and heuristic methods.

VII. CONCLUSIONS

When scheduling tasks in multicore systems implemented in leading-edge IC technologies, reliability awareness is critical to achieving guaranteed lower bounds on performance and throughput. With this in mind, we have formulated and solved a new *maximum-value, reliability-constrained overdrive frequencies* (MVRCOF) problem. We show how an optimal (discretized) solution may be found using exhaustive search, and we propose a heuristic that maximizes the overdrive frequency in the order of user-specified weights for each operating mode. We develop a solver that serves as the foundation of both the optimal and heuristic flows. Our methods are the first to *guarantee* both acceptable performance and throughput, in that all tasks are executable for their entire duration at the optimal overdrive frequencies, without exceeding the total lifetime requirement of any core. Further, we determine optimal execution times of each core in each mode; these can be utilized by schedulers for balanced wearout of cores. Experimentally, across eight testcases on between 4 and 8 cores, our optimal overdrive frequencies achieve between 2.2% and 17.4% greater value than existing RC-LG policies [4] and [15] (the largest improvement is for the 8-core testcase). Our ongoing work seeks five improvements: (i) use of geometric programming to guarantee more balanced usage of core combinations in each operating mode; (ii) application of our methods to traces from actual server workloads, along with validation of our results on actual servers; (iii) extension of our optimizations to accurately comprehend the cost of task migration / context switching in the objective function; (iv) enhance our methods to achieve solutions that are temperature history-aware; and (v) the relationship between feasible range of overdrive frequencies and the area, mix of different-Vt cells in implementation, etc. of a block.

[18] To confirm the optimality of these solutions, we separately solve the dual program for each m, minimize $\sum_{i=N}^{2N} \lambda_i MTTF_i + \mu_1 E_{nom,m} + \mu_2 E_{OD,m}$ and verify that we obtain identical objective function values.

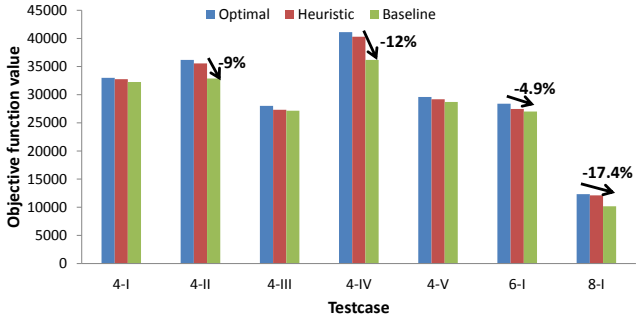

Fig. 8. Comparison of objective function values from optimal, heuristic, and b

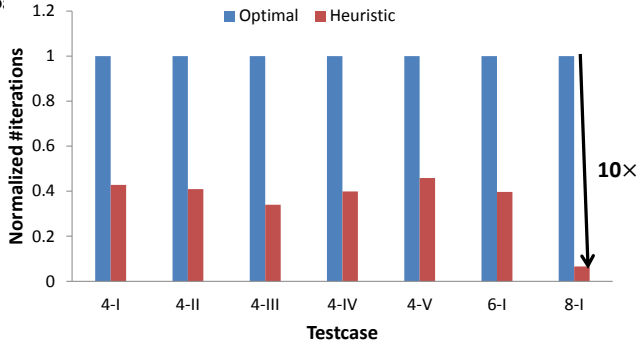

Fig. 9. Comparison of the normalized #iterations (normalized with respect to the #iterations of the optimal solution) between optimal and heuristic solutions for seven testcases that yield an optimal solution.

REFERENCES

[1] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*, Springer Books, 2009.

[2] O. Bilgir, M. Martonosi and Q. Wu, "Exploring the Potential of CMP Core Count Management on Data Center Energy Savings", *Proc. Workshop on Energy Efficient Design*, 2011.

[3] J. R. Black, "Electromigration Failure Modes in Aluminium Metallization for Semiconductor Devices", *IEEE Letters*, 57(9) (1969), pp. 1578-1594.

[4] A. K. Coskun, T. S. Rosing, K. A. Whisnant and K. C. Gross, "Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs", *IEEE Trans. on VLSI* 16(9) (2008), pp. 1127-1140.

TABLE VI

COMPARISON OF % EXECUTION TIME IN EACH CORE AND IN EACH ACTIVE COMBINATION WITH 8-I WITH Δ_{fOD} = 50MHZ. EXECUTION TIMES ARE SHOWN AS NOMINAL/OVERDRIVE.

	$C(8,2)$ == 28 combinations ($j = 1,2,...,28$)								$C(8,6)$ == 28 combinations ($j = 1,2,...,28$)							
	Cores								Cores							
j	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3	50/0	-	-	50/0	-	-	-	-	1.3/0	1.3/0	1.3/0	1.3/0	1.3/0	-	-	1.3/0
4	0/24.2	-	-	-	0/24.2	-	-	-	0/0.3	0/0.3	0/0.3	0/0.3	-	0/0.3	0/0.3	-
5	0/0	-	-	-	-	0/0	-	-	0/0	0/0	0/0	0/0	-	0/0	-	0/0
9	-	0/0	-	0/0	-	-	-	-	0/31	0/31	0/31	-	0/31	-	0/31	0/31
11	-	0/25.8	-	-	-	0/25.8	-	-	0/0	0/0	-	0/0	0/0	0/0	0/0	-
12	-	0/0	-	-	-	-	0/0	-	98.7/7.8	98.7/7.8	-	98.7/7.8	98.7/7.8	98.7/7.8	-	98.7/7.8
13	-	0/0	-	-	-	-	-	0/0	0/0.3	0/0.3	-	0/0.3	0/0.3	-	0/0.3	0/0.3
16	-	-	0/0	-	-	0/0	-	-	0/2	-	0/2	0/2	0/2	0/2	0/2	-
17	-	-	0/25.8	-	-	-	0/25.8	-	0/0	-	0/0	0/0	0/0	0/0	-	0/0
19	-	-	-	0/0	0/0	-	-	-	0/29.7	-	0/29.7	0/29.7	-	0/29.7	0/29.7	0/29.7
22	-	-	-	0/24.2	-	-	-	0/24.2	-	0/28.7	0/28.7	0/28.7	0/28.7	0/28.7	0/28.7	-
23	-	-	-	-	0/0	0/0	-	-	-	0/0	0/0	0/0	0/0	0/0	-	0/0
25	-	-	-	-	50/0	-	-	50/0	-	0/0	0/0	0/0	-	0/0	0/0	0/0
Sum	50/24.2	0/25.8	0/25.8	50/24.2	50/24.2	0/25.8	0/25.8	50/24.2	100/71.1	100/68.1	1.3/91.7	100/68.8	100/69.8	98.7/68.5	0/92	100/68.8

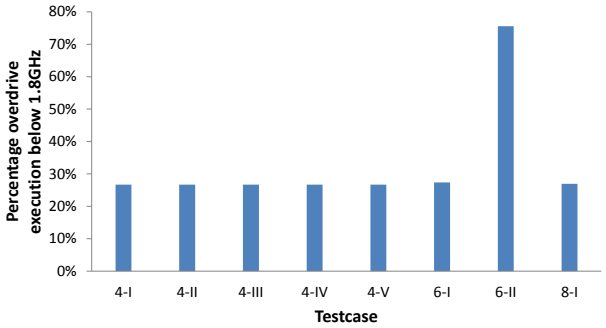

Fig. 10. Percentage of overdrive execution time below "acceptable performance" of 1.8GHz of baseline solutions. Across all testcases, the baseline solutions will execute 27% of the overall overdrive execution time below the minimum required frequency of 1.8GHz. Our optimal and heuristic solutions always *guarantee* execution at overdrive frequencies $\geq$1.8GHz.

[5] A. K. Coskun, R. Strong, D. M. Tullsen and T. S. Rosing "Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip", *Proc. SIGMETRICS*, 2009, pp. 169-180.

[6] L. Huang, F. Yuan and Q. Xu, "Lifetime Reliability-Aware Task Allocation and Scheduling for MPSoC Platforms", *Proc. DATE*, 2009, pp. 51-56.

[7] A. B. Kahng, S. Nath and T. S. Rosing, "On Potential Design Impacts of Electromigration Awareness", *Proc. ASP-DAC*, 2013, pp. 527-532.

[8] E. Karl, D. Blaauw, D. Sylvester and T. Mudge, "Multi-Mechanism Reliability Modeling and Management in Dynamic Systems", *IEEE Trans. on VLSI* 16(4) (2008), pp. 476-487.

[9] U. R. Karpuzcu, B. Greskamp and J. Torrellas, "The BubbleWrap Many-Core: Popping Cores for Sequential Acceleration", *Proc. MICRO*, 2009, pp. 447-458.

[10] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan and C. Kozyrakis, "Power Management of Datacenter Workloads Using Per-Core Power Gating", *IEEE Computer Architecture Letters* 8(2) (2009), pp. 48-51.

[11] H. Liu, "A Measurement Study of Server Utilization in Public Clouds", *Proc. Dependable Autonomic and Secure Computing*, 2011, pp. 435-442.

[12] K. Mihic, T. Simunic and G. D. Micheli, "Reliability and Power Management of Integrated Systems", *Proc. DSD*, 2004, pp. 5-11.

[13] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis", *Proc. SOCC*, 2012.

[14] P. Rong and M. Pedram, "Power-Aware Scheduling and Dynamic Voltage Setting for Tasks Running on a Hard Real-Time System", *Proc. ASP-DAC*, 2006, pp. 473-478.

[15] T. S. Rosing, K. Mihic and G. D. Micheli, "Power and Reliability Management of SoCs", *IEEE Trans. on VLSI* 15(4) (2007), pp. 391-403.

[16] R. Sasanka, S. V. Adve, Y.-K. Chen and E. Debes, "The Energy Efficiency of CMP vs. SMT for Multimedia Workloads", *Proc. ICS*, 2004, pp. 196-206.

[17] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan and D. Tarjan, "Temperature-Aware Microarchitecture", *Proc. ISCA*, 2003, pp. 2-13.

[18] J. Srinivasan, S. V. Adve, P. Bose and J. A. Rivers, "The Case for Lifetime Reliability-Aware Microprocessors", *Proc. ISCA*, 2004, pp. 276-287.

[19] W. Torell and V. Avelar, "Performing Effective MTBF Comparisons for Data Center Infrastructure", *APC Whitepaper 112*, 2005.

[20] S. Wang and J.-J. Chen, "Thermal-Aware Lifetime Reliability in Multicore Systems", *Proc. ISQED*, 2010, pp. 399-405.

[21] S. Zhang and K. S. Chatha, "Approximation Algorithm for the Temperature-Aware Scheduling Problem", *Proc. ICCAD*, 2007, pp. 281-288.

[22] X. Zhou, J. Yang, Y. Xu, Y. Zhang and J. Zhao, "Thermal-Aware Task Scheduling for 3D Multicore Processors", *IEEE Trans. on Parallel and Distributed Systems* 21(1) (2010), pp. 60-71.

[23] J. Mars and L. Tang, UC San Diego, *personal communication*, May 2013.

[24] *AMD FX.* http://www.engadget.com/2012/10/23/amd-fx-processor-refresh/ .

[25] *Cadence SOC Encounter User Guide.* http://www.cadence.com .

[26] *Design Compiler User Guide.* https://solvnet.synopsys.com/dow_retrieve/latest/dcug/dcug.html .

[27] *Failure Mechanisms and Models for Semiconductor Devices*, JEDEC JEP122G, 2011.

[28] *Intel Turbo Boost technology – On-Demand Processor Performance.* http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html .

[29] *ITRS 2011 Edition.* http://public.itrs.net .

[30] *lp_solve Reference Guide.* http://lpsolve.sourceforge.net/5.5/ .

[31] *OpenCores.* http://opencores.org/projects .

[32] *Thermal Reliability.* http://wdc.com/wdproducts/library/other/2579-001134.pdf .