Energy-Efficient Variable-Flow Liquid Cooling in 3D Stacked Architectures

Ayse K. Coskun[†], David Atienza[‡], Tajana Simunic Rosing[⋆], Thomas Brunschwiler[#], Bruno Michel[#]
[†]Electrical and Computer Engineering Department, Boston University, USA.
[‡]Embedded Systems Laboratory (ESL), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
[⋆]Computer Science and Engineering Dept. (CSE), University of California San Diego, USA.
[#]IBM Research GmbH, Zurich Research Laboratory, Switzerland.

Abstract— **Liquid cooling has emerged as a promising solution for addressing the elevated temperatures in 3D stacked architectures. In this work, we first propose a framework for detailed thermal modeling of the microchannels embedded between the tiers of the 3D system. In multicore systems, workload varies at runtime, and the system is generally not fully utilized. Thus, it is not energy-efficient to adjust the coolant flow rate based on the worst-case conditions, as this would cause an excess in pump power. For energy-efficient cooling, we propose a novel controller to adjust the liquid flow rate to meet the desired temperature and to minimize pump energy consumption. Our technique also includes a job scheduler, which balances the temperature across the system to maximize cooling efficiency and to improve reliability. Our method guarantees operating below the target temperature while reducing the cooling energy by up to 30%, and the overall energy by up to 12% in comparison to using the highest coolant flow rate.**

I. Introduction

3D integration is a recently proposed design method for overcoming the limitations regarding the delay and power consumption of the interconnects. An important challenge in 3D circuits is the elevated temperature. Vertical stacking increases the thermal resistances and makes it difficult to remove the heat using conventional cooling methods. Liquid cooling is a potential solution to address the high temperatures in 3D chips, due to the higher heat removal capability of liquids in comparison to air. Our focus in this work is developing energy- and performance-efficient thermal management techniques for joint control of job scheduling and liquid flow rate in 3D systems.

Liquid cooling is performed by attaching a cold plate with built-in microchannels, and/or by fabricating microchannels between the layers of the 3D architecture. Then, a coolant fluid (i.e., water or other fluids) is pumped through the microchannels to remove the heat. The heat removal performance of this approach, called interlayer cooling [4], scales with the number of tiers. The flow rate of the pump can be altered dynamically, but as there is a single pump connected to the system, the flow rates among the channels are the same—as the channel dimensions are identical. One obvious way to set the flow rate is by matching it with the worst-case temperature. However, the pump power increases quadratically with the increase in flow rate [4], and its contribution to the overall system energy is significant. Also, over-cooling may cause dynamic fluctuations in temperature, which degrade reliability and cooling efficiency. Through runtime system analysis and intelligent control of the flow rate, it is possible to determine the minimum flow rate to remove the heat and maintain a safe system temperature. In addition, by maintaining a target temperature value throughout the execution, we can minimize the temperature variations. Note that, while reducing the coolant flow rate, it is necessary to maintain the temperature at a level where the temperature-dependent leakage power does not revert the benefits achieved with lower-power pumping.

In a 3D system, cores located at different layers or at different coordinates across a layer may significantly vary in their rates for heating and cooling [7]. This variation is due to the change in thermal resistance, which is a function of the unit's location. Therefore, even when we select an energy-efficient flow rate for the coolant, large temperature gradients across the system may still exist. Conventional multicore schedulers, e.g., dynamic load balancing, do not consider such thermal imbalances. To address this issue, we propose a temperature-aware load balancer, which weighs each core's workload with the core's thermal properties and uses this weighted computation to balance the temperature. This paper's contributions are the following:

- We show in detail how to model the effects of the liquid flow on temperature. Our model is based on the liquid cooling work of Brunschwiler et al. [4]. We improve the previous liquid cooling model in [6] with a finer-grained computation of the heat spread, and by using model parameters verified by finite element simulation. We integrate our modeling infrastructure in HotSpot [19].
- We propose a controller for adjusting the liquid flow rate dynamically to maintain a target temperature while minimizing the pump power consumption. Our controller forecasts maximum system temperature, and uses this forecast to proactively set the flow rate. This way, we avoid over- or under-cooling due to delays in reacting to the temperature changes.
- We integrate the controller with a novel job scheduler that computes the current workload of each core as a function of the core's thermal properties. The scheduler addresses the inherent thermal imbalances in multicore 3D systems and reduces the frequency of large thermal gradients.
- On 2- and 4-layered 3D systems we experiment with, we see that our method achieves up to 30% reduction in cooling energy, and 12% reduction in system-level energy in comparison to setting the flow rate at the maximum value, while we maintain the target temperature. We also show that temperature-aware load balancing reduces the hot spots and gradients significantly better than load balancing or reactive thread migration.

The rest of the paper starts with an overview of the prior art. Section III describes the thermal model for 3D systems with liquid cooling. In Section IV, we provide the details of the flow rate controller and job scheduler. The experimental results are in Section V, and Section VI concludes the paper.

II. Related Work

Accurate thermal modeling is critical in the design and evaluation of systems and policies. HotSpot [19] is an automated thermal model, which calculates transient temperature response given the physical and power consumption characteristics of the chip. To reduce simulation time even for large multicore systems, a thermal emulation framework for FPGAs is proposed in [1]. Dynamic thermal management in microprocessors has been introduced to ensure chip temperature does not exceed critical values. Activity migration [12] and fetch toggling [19] are examples of such techniques. Kumar et al. propose a hybrid method that combines clock gating and software thermal management [13]. The multicore thermal management method introduced by Donald et al. [9] combines distributed dynamic voltage scaling (DVS) with process migration. For multicore systems, temperature-aware task scheduling [8] achieves desirable thermal profiles at low performance cost.

Most of the prior work in thermal management of 3D systems address design stage optimization, such as thermal-aware floorplanning (e.g. [11]) and integrating thermal via planning in the 3D floorplanning process [17]. In [24], the authors evaluate several policies for task migration and DVS. Our prior work proposes a temperature-aware scheduling method specifically for air-cooled 3D systems [7], taking into account the thermal heterogeneity of the different layers in the system.

The use of convection in microchannels to cool down high power density chips has been an active area of research since the initial work by Tuckerman and Pease [23]. Their liquid cooling system can remove 1000 W/cm^2; however, the volumetric flow rate and the pressure drop are large. More recent work shows how back-side liquid cold plates, such as staggered microchannel and distributed return jet plates, can handle up to 400 W/cm^2 in single-chip applications [3]. The heat removal capability of interlayer heat-transfer with pin-fin in-line structures for 3D chips is investigated in [4]. At a chip size of 1 cm^2 and a $\Delta T_{jmax-in}$ of 60 K, the heat-removal performance is shown to be more than 200 W/cm^2 at interconnect pitches bigger than 50 μm. Finally, previous work in [2], [15] describe how to achieve variable flow rate for the coolant.

Prior liquid cooling work in [6] evaluates existing thermal management policies on a 3D system with a fixed-flow rate setting, and also investigates the benefits of variable flow using a policy to increment/decrement the flow rate based on temperature measurements, without considering energy consumption. This paper's contribution is a controller design to provide sufficient coolant flow to the system with minimal cooling energy. Our management policy combines this controller with a novel job scheduler to prevent thermal variations, and further improves the cooling efficiency without affecting performance. We also improve the liquid cooling model in [6] with a finer-grained computation of the heat spread to the microchannels, and by using model parameters verified by finite-element simulation.

III. Modeling of 3D Systems with Liquid Cooling

Modeling the temperature dynamics of 3D stacked architectures with liquid cooling consist of: (A) Forming the grid-level thermal R-C network, (B) Detailed modeling of the interlayer material between the tiers, including the through-silicon-vias (TSVs) and the microchannels, and (C) Modeling the pump and the coolant flow rate. We assume forced convective interlayer cooling with water [4] in this work, but our model can be extended to other coolants as well.

Figure 1 shows the 3D systems targeted in this paper. A target system consists of two or more stacked layers (with cores, L2 caches, crossbar, and other units for memory control, buffering, etc.), and microchannels are built in between the vertical layers for liquid flow. The crossbar contains the TSVs that provide the connection between the layers. The microchannels, which are connected to an impeller pump (such as [14]), are distributed uniformly, and fluid flows through each channel at the same flow rate. The liquid flow rate provided by the pump can be dynamically altered at runtime. In the rest of this section, we provide the details of the thermal modeling infrastructure we developed for the 3D system.

A. Grid-Level Thermal Model for 3D Systems with Liquid Cooling

Similar to 2D chips, 3D thermal modeling is performed using an automated tool that forms the R-C circuit for given grid dimensions. In this work, we utilize HotSpot v.4.2. [19], which includes 3D modeling capabilities. The existing model in HotSpot assumes that the interlayer material between stacked layers has homogeneous thermal characteristics, represented by a thermal resistivity and a specific heat capacity value. The extension we have developed in this work provides a new interlayer material model to include the TSVs and the microchannels.

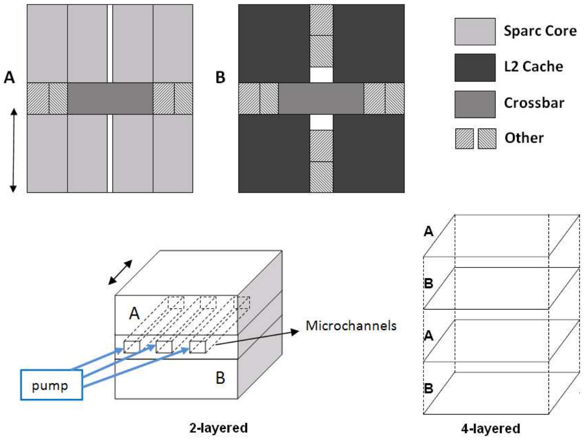

Fig. 1. Floorplans of the 3D Systems.

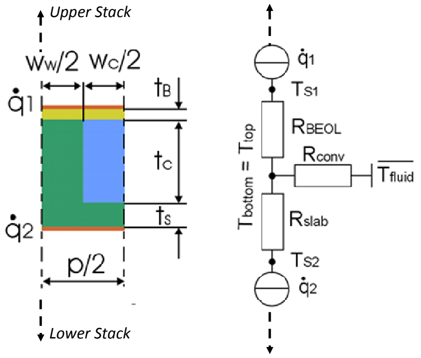

Fig. 2. Cross section of the 3D layers and the resistive network.

To model the heterogeneous characteristics of the interlayer material including the TSVs and microchannels, we introduce two novelties: (1) As opposed to having a uniform thermal resistivity value of the layer, our infrastructure enables having various resistivity values for each grid cell, (2) The resistivity value of the cell can vary at runtime. Item (1) enables distinctly modeling TSVs, the microchannels, and the interlayer material, while item (2) enables modeling the liquid coolant and dynamically changing flow rate. Thus, the interlayer material is divided into a grid, where each grid cell except for the cells of the microchannels has a fixed thermal resistance value depending on the characteristics of the interface material and TSVs. The thermal resistivity of the microchannel cells is computed based on the liquid flow rate through the cell, and the characteristics of the liquid at runtime. We use grid cells of $100\mu m$x$100\mu m$ in our experiments.

In a 3D system with liquid cooling, we compute the local junction temperature using a resistive network, as shown in Figure 2. In this figure, the thermal resistance of the wiring layers (R_{BEOL}), the thermal resistance of the silicon slab (R_{slab}), and the convective thermal resistance (R_{conv}) are combined to model the 3D stack. In the figure, the heat flux values ($\dot{q}$) represent the heat sources. This R-network can be solved to get the junction temperature (T_j). Note that the figure shows the heat sources and the resistances of only one layer, and heat will be dissipated to both opposing vertical directions (i.e., up and down) from the heat sources. For example, if there is another layer above the two heat-dissipating layers shown in the figure, $\dot{q}_1$ will also be dissipating heat towards the upper stack. Also, the network in

Figure 2 is a simplification and it assumes isothermal channel walls; i.e., top and bottom of the microchannel have the same temperature.

The typical junction temperature (T_j) response at uniform chip heat flux and convective cooling is a sum of the following three components: (1) The thermal gradient due to conduction (ΔT_{cond}); (2) the coolant temperature, which increases along the channel due to the absorption of sensible heat (ΔT_{heat}); and (3) the convective (ΔT_{conv}) portion, which increases until fully developed hydrodynamic and thermal boundary layers have been reached [4]. The total temperature rise on the junction, ΔT_j, is computed as the following:

$$\Delta T_j = \Delta T_{cond} + \Delta T_{heat} + \Delta T_{conv} \tag{1}$$

Thermal gradient due to heat conduction through the BEOL layer, ΔT_{cond} is computed with Equations 2 and 3. Note that ΔT_{cond} is independent of the flow rate. Figure 2 demonstrates t_B, and k_{BEOL} is the conductivity of the wiring layer.

$$\Delta T_{cond} = R_{th-BEOL} \cdot \dot{q_1} \tag{2}$$

$$R_{th-BEOL} = \frac{t_B}{k_{BEOL}} \tag{3}$$

Temperature change due to absorption of sensible heat is computed using Equations 4 and 5. A_{heater} is the area of the heater (i.e., total area consuming power), c_p is the heat capacity of the coolant, ρ is the density of the coolant, and $\dot{V}$ is the volumetric flow rate in the microchannel (in l/min). Equations 4 and 5 are valid for uniform power dissipation. For the general case, heat absorption in the fluid is calculated iteratively along the channel: $\Delta T_{heat}(n+1) = \sum_{i=1}^{n} \Delta T_{heat}(i)$, where n is the position along the channel.

$$\Delta T_{heat} = (\dot{q_1} + \dot{q_2}) \cdot R_{th-heat} \tag{4}$$

$$R_{th-heat} = \frac{A_{heater}}{c_p \cdot \rho \cdot \dot{V}} \tag{5}$$

Finally, Equation 7 shows how to calculate ΔT_{conv}. Note that ΔT_{conv} is independent of flow rate in case of developed boundary layers. h is dependent on hydraulic diameter, Nusselt number, and conductivity of the fluid [4]. As ΔT_{conv} is not affected by the change in flow rate, we compute this parameter prior to simulation and use a constant value during experiments. Figure 2 demonstrates w_c, t_c, and p parameters on the cross-section of the 3D system.

$$\Delta T_{conv} = (\dot{q_1} + \dot{q_2}) \cdot h_{eff} \tag{6}$$

$$h_{eff} = h \frac{2 \cdot (w_c + t_c)}{p} \tag{7}$$

The equations above give the ΔT_j for the unit cell shown in Figure 2; thus, we extend the computation to model multiple layers and multiple cells as well.

Table I lists the parameters used in the computations, and provides the values for the constants, which are taken from prior liquid cooling work [4]. Note that the flow rate ($\dot{V}$) range provided in the table is per cavity (i.e., the interlayer cavity consisting of all the microchannels in one layer), and this flow is further divided into the microchannels.

We compute the flow rate dependent components whenever the flow rate changes. Heat flux, $\dot{q}$ (W/cm^2), values change as the power consumption changes. Instead of reacting to every instant change in power consumption of the cores, we re-compute the $\dot{q}$ values periodically to reduce the simulation overhead.

Considering the dimensions and pitch requirements of microchannels and TSVs, we assume there are 65 microchannels in between each two layers (in each cavity), and there are cooling layers on the very top and the bottom of the stacks. Thus, there are 195 and 325 microchannels in the 2- and 4-layered systems, respectively.

TABLE I. PARAMETERS FOR COMPUTING EQUATION 1

Parameter	Definition	Value
$R_{th-BEOL}$	Thermal resistance of wiring levels	Eqn.(3) 5.333 $(K \cdot mm^2)/W$
t_B	See Figure 2	$12\mu m$
k_{BEOL}	Conductivity of wiring levels	$2.25W/(m \cdot K)$
$R_{th-heat}$	Effective thermal resistance	Eqn.(5)
A_{heater}	Heater area	Area of grid cell
c_p	Coolant heat capacity	$4183J/(kg \cdot K)$
ρ	Coolant density	$998kg/m^3$
$\dot{V}$	Volumetric flow rate	0.1-1 l/min per cavity
h	Heat transfer coefficient	$37132W/(m^2 \cdot K)$
w_c	See Figure 2	$50\mu m$
t_c	See Figure 2	$100\mu m$
t_s	See Figure 2	$50\mu m$
p	See Figure 2	$100\mu m$

In our target systems shown in Figure 1, we assume the TSVs are located within the crossbar. Placing the TSVs in the central section of the die provides an advantage on the thermal design as well, as TSVs reduce the temperature due to the low thermal resistivity of Cu. We assume there are **128 TSVs** within the crossbar block connecting each two layers. Feasible TSVs for microchannels of $100\mu m$ height and $100\mu m$ pitch have a minimal pitch of $100\mu m$ as well due to aspect ratio limits. We assume each TSV occupies a space of $50\mu m$x$50\mu m$, and the TSVs have a minimum spacing requirement of $100\mu m$.

Previous work has studied granularity and accuracy of TSV modeling [6]. The study shows that using a block-level granularity for TSVs, i.e., assigning a TSV density to each block based on the functionality of the unit, constitutes a reasonable trade-off between accuracy and simulation time. Thus, based on the TSV density of the crossbar, we compute the joint resistivity of that area combining the resistivity values of interlayer material and Cu. We do not alter the thermal resistivity values for the regions without TSVs or microchannels. We assume that the effect of the TSV insertion to the heat capacity of the interface material is negligible, which is a reasonable assumption considering the total area of TSVs constitutes a very small percentage of the total area of the material.

B. Modeling the Pump and Liquid Flow Rate

All the microchannels are connected to a pump to receive the coolant. We assume a 12V DC-pump, *Laing DDC* [14], which has suitable dimensions, flow rates, and power consumption for this type of liquid cooling. The power consumption of the pump across the *five* flow rate settings we use is shown in Figure 3 (right y-axis). The pressure drop for these flow rates changes between 300-600 mbar [14]. We assume that the total flow rate of the pump is equally distributed among the cavities, and among the microchannels. DC pumps typically have low efficiency. Also, the flow rate in the microchannels further decreases because the the pressure drop in the small microchannels is larger than its value in the pump output channel. In this work, we assume a global reduction in the flow rate by 50% to account for the loss due to all of these factors. In Figure 3, we show the per cavity flow rates for the 2- and 4-layered 3D systems after applying the reduction factor.

IV. JOINT FLOW RATE CONTROL AND JOB SCHEDULING

In this section, we provide the details of our energy-efficient thermal management technique for 3D systems with liquid cooling. The goals of our technique are: (1) Tuning the liquid flow rate to meet the heat removal demand of the current workload and reducing the energy consumption; (2) Minimizing the thermal imbalances across the chip to reduce the adverse effects of variations on reliability, performance, and cooling efficiency. To achieve these goals, we propose joint flow rate control and job scheduling. Figure 4 provides

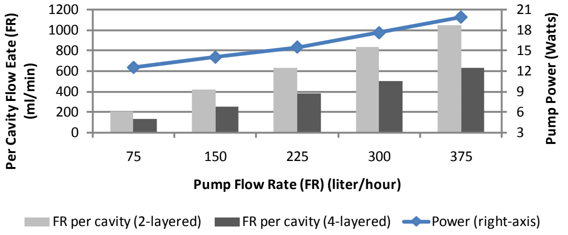

Fig. 3. Power consumption and flow rates of the pump (based on [14]). Per cavity flow rates reflect 50% efficiency assumption.

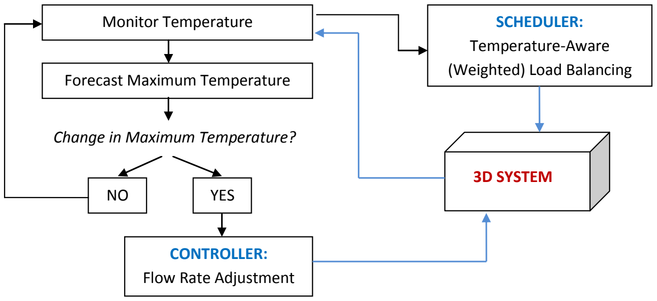

Fig. 4. Overview of the technique.

a flow chart of our method. We monitor the temperature at regular intervals for all the cores in the 3D system. Based on the forecasted change in maximum temperature, the controller is responsible for adjusting the coolant flow rate. In addition, the scheduler performs temperature-aware load balancing to reduce the thermal gradients.

Temperature Monitoring and Forecasting:

Monitoring temperature provides our technique with the ability to adapt the controller and job scheduler decisions. We assume each core has a thermal sensor. One way to utilize the thermal feedback is to react to temperature changes. A typical impeller pump like the one we use ([14]) takes around 250-300ms to complete the transition to a new flow rate. Due to the time delay in adjusting the flow rate, a reactive policy is likely to result in over-/under-cooling—the thermal time constant on a 3D system like ours is typically less than 100ms. Thus, for the liquid flow rate controller, we forecast temperature into the near future, and adjust the flow rate control on time to meet the heat removal requirement.

We use autoregressive moving average (ARMA) [5] to predict the *maximum temperature* for the next interval. Predicting maximum temperature is sufficient to pick the suitable liquid flow rate to apply, as the flow rate is fixed among the channels. Note that our job scheduler balances the temperature, therefore the temperature difference among cores is minimized. ARMA forecasts the future value of the time-series signal based on the recent history (i.e., maximum temperature history in this work), therefore we do not require an offline analysis. The prediction is highly accurate because of the serial correlation within most workloads and the slow change in temperature due to the thermal time constants. Furthermore, the rate of change of maximum temperature is typically even slower, resulting in easier prediction. In our experiments, we use a sampling rate of 100ms, and predict 500ms into the future.

If the trend of the maximum temperature signal changes and the predictor cannot forecast accurately, we reconstruct the ARMA predictor, and use the existing model until the new one is ready. Such cases occur when the workload dramatically changes (e.g., day-time and night-time workload patterns for a server). To achieve fast and easy detection, we apply the the sequential probability ratio test (SPRT) [10]. SPRT is a logarithmic likelihood test to decide whether the error between the predicted series and measured series is diverging from zero [10], [5]—i.e., if the predictor is no longer fitting the workload, the difference function of the two time series would increase. As the maximum temperature profile changes slowly, we need to update the ARMA predictor very infrequently.

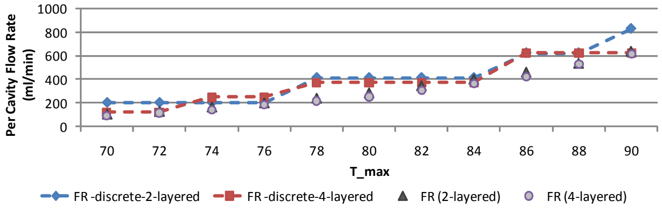

Fig. 5. Flow rate requirements to cool a given T_{max}.

Liquid Flow Rate Control:

The input to the controller is the predicted maximum temperature, and the output is the flow rate for the next interval. Then, considering that we have discrete flow rate settings for the pump, we first analyze the effect of each flow rate for both 3D systems (2- and 4-layered).

Figure 5 shows which flow rate (per cavity) should be applied when the maximum temperature is T_{max} so that the temperature is guaranteed to cool below the target operating temperature of 80^oC. In this figure, the dashed lines show the discrete flow rate settings, while the triangular and circular shaped data points refer to minimum rate to maintain the desired temperatures.

Based on this analysis, we see that for a given system and maximum temperature, we already know which flow rate setting is able to cool the system to the target temperature level. We set-up a look-up table indexed by temperature values, and each line holds a flow rate value. At runtime, depending on the maximum temperature prediction, we pick the appropriate flow rate from the table. As the maximum temperature prediction is highly accurate (well below 1^oC), this way we can adjust the cooling system to meet the changes in the heat removal demand on time. To avoid rapid oscillations, once we switch to a higher flow rate setting, we do not decrease the flow rate until the predicted T_{max} is at least 2^oC lower than the boundary temperature between two flow rate settings. The runtime overhead of using a look-up table based controller is negligible, considering that the cost is only limited to a look-up from a small-sized table.

Job Scheduling:

Our job scheduler is a temperature-aware version of load balancing. Dynamic load balancing is a common policy used in multicore schedulers today. While frequent load balancing eliminates contention and long thread waiting times in the queues, it does not consider the location of the cores. However, a core's thermal behavior is strongly correlated with where it is placed on the chip, and the power consumption of the neighboring units.

We assume short threads, which is a common scenario in server workloads running on multiprocessor systems [9], [8]. For instance, in real-life workloads running on the UltraSPARC T1, the thread length (i.e., continuous execution time without any interrupt) has been reported to vary between a few to several hundred milliseconds [8]. Thus, since we consider threads with short lengths and similar execution time, we use number of threads for computing the job queue length of each core. Note that, depending on the available information, our approach can be extended for other workload metrics such as instruction count per thread.

To address the thermal asymmetries of cores in a 3D system, we introduce *Weighted Load Balancing*. We do not change the priority

TABLE II. WORKLOAD CHARACTERISTICS

	Benchmark	Avg Util (%)	L2 I-Miss	L2 D-Miss	FP instr
1	Web-med	53.12	12.9	167.7	31.2
2	Web-high	92.87	67.6	288.7	31.2
3	Database	17.75	6.5	102.3	5.9
4	Web & DB	75.12	21.5	115.3	24.1
5	gcc	15.25	31.7	96.2	18.1
6	gzip	9	2	57	0.2
7	MPlayer	6.5	9.6	136	1
8	MPlayer&Web	26.62	9.1	66.8	29.9

TABLE III. THERMAL MODEL AND FLOORPLAN PARAMETERS

Parameter	Value
Die Thickness (one stack)	$0.15mm$
Area per Core	$10mm^2$
Area per L2 Cache	$19mm^2$
Total Area of Each Layer	$115mm^2$
Convection Capacitance	140 J/K
Convection Resistance	0.1 K/W
Interlayer Material Thickness	0.02 mm
Interlayer Material Thickness (with channels)	0.4 mm
Interlayer Material Resistivity (without TSVs)	0.25 mK/W

and performance aware features of the load balancing algorithm, but only modify how the queue lengths are computed. Each core has a queue to hold the incoming threads, and the weighted queue length of a core is computed as:

$$l^i{}_{weighted} = l^i{}_{queue} \cdot w^i{}_{thermal}(T(k)) \qquad (8)$$

$l^i{}_{queue}$ is the number of threads currently waiting in the queue of core i, and $w^i{}_{thermal}(T(k))$ is the thermal weight factor. The weight factor is a function of the current maximum temperature of the system. For a given set of temperature ranges, the weight factors for all the cores are computed in a pre-processing step and stored in the look-up table. For example, consider a 4-core system, where the average power values for the cores to achieve a balanced 75^oC are p_1, p_2, p_3, and p_4, and $p_1 = p_4 > p_2 = p_3$. This means cores 2 and 3 should run fewer number of threads per unit time to balance temperature. Thus, we take the multiplicative inverse of the power values, normalize them, and use them as thermal weight factors.

V. EXPERIMENTAL RESULTS

The 3D multicore systems we use in our experiments are based on the 90nm UltraSPARC T1 processor [16]. The power consumption, area, and the floorplan of UltraSPARC T1 are available in [16]. UltraSPARC T1 has 8 multi-threaded cores, and a shared L2-cache for every two cores. Our simulations are carried out with 2-, and 4-layered stack architectures. We place cores and L2 caches of the UltraSPARC T1 on separate layers (see Figure 1). Separating cores and memory is a preferred design scenario for shortening wires between the cores and caches and achieving higher performance.

First, we gather workload characteristics of real applications on an UltraSPARC T1. We sample the utilization percentage for each hardware thread at every second using mpstat for half an hour, and record the length of user and kernel threads using DTrace [18]. We use various real-life benchmarks including web server, database management, and multimedia processing. The web server workload is generated by SLAMD [20] with 20 and 40 threads per client to achieve medium and high utilization, respectively. For database applications, we experiment with MySQL using sysbench for a table with 1 million rows and 100 threads. We also run the gcc compiler and the gzip compression/decompression benchmarks as samples of SPEC-like benchmarks. Finally, we run several instances of the mplayer (integer) benchmark with 640x272 video files as typical examples of multimedia processing. A detailed summary of the benchmarks is in Table II, showing average system utilization, cache misses, and floating point (FP) instructions. Misses and FP count are per 100K instructions. The workload statistics collected on the UltraSPARC T1 are replicated for the 4-layered 16-core system.

SPARC's peak power is close to its average value [16]. Thus, we assume that the instantaneous dynamic power consumption is equal to the average power at each state (active, idle, sleep). The active state power is taken as 3 Watts [16]. The cache power consumption is 1.28W per each L2, as computed by CACTI [22] and verified by the values in [16]. We model crossbar power by scaling the average power value according to the number of active cores and the memory accesses. To account for the temperature effects on leakage power, we use the polynomial model proposed in [21].

Many systems have power management capabilities to reduce the energy consumption. We implement Dynamic Power Management (DPM), especially to investigate the effect on thermal variations. We utilize a fixed timeout policy, which puts a core to sleep state if it has been idle longer than the timeout period (i.e., 200ms in our experiments). We set a sleep state power of 0.02 Watts, which is estimated based on sleep power of similar cores.

We use HotSpot Version 4.2 [19] as the thermal modeling tool. We use a sampling interval of 100 ms, and all simulations are initialized with steady state temperature values. The model parameters are provided in Table III. Modeling methodology for the interlayer material to include TSVs and the microchannels has been described in Section III. In our experiments, we compare air-cooled and liquid-cooled 3D systems. For the conventional system, we use the default characteristics of a modern CPU package in HotSpot.

We assume that each core has a temperature sensor, which is able to provide temperature readings at regular intervals (e.g., 100ms). Modern OSes have a multi-queue structure, where each CPU core is associated with a dispatch queue, and the job scheduler allocates the jobs to the cores according to the current policy. In our simulator, we implement a similar infrastructure, where the queues maintain the threads allocated to cores and execute them.

We compare our technique to other well-known policies in terms of temperature, energy, and performance. **Dynamic Load Balancing (LB)** balances the workload by moving threads from a core's queue to another if the difference in queue lengths is over a threshold. LB does not have any thermal management features. **Reactive Migration** initially performs load balancing, but upon reaching a threshold temperature, which is set to 85^oC in this work, it moves the currently running thread from the hot core to a cool core. Our novel temperature-aware weighted load balancing method is denoted as **TALB**. We also compare liquid cooling systems with air cooling systems (denoted with *(Air)*). In the plots *Var* refers to variable flow rate and *Max* refers to with using a maximum (worst-case) flow rate.

Figure 6 shows the average percentage of time spent above the threshold across all the workloads, percentage of time spent above threshold for the hottest workload, and energy for the 2-layered 3D system. We demonstrate both the pump energy and the total chip energy in the plot. Note that, for the air-cooled system, there is also an additional energy cost due to the fans, which is beyond the focus of this work and not included in the plot. The energy consumption values are normalized with respect to the load balancing policy on a system with air cooling. We see that temperature-aware load balancing combined with liquid flow control achieves 10% energy savings on average in comparison to setting the worst-case flow rate. For low utilization workloads, such as gzip and MPlayer, the total energy savings (including both chip and pump energy) reach 12%, and the reduction in cooling energy exceeds 30%.

Figure 7 shows the average and maximum frequency of spatial and

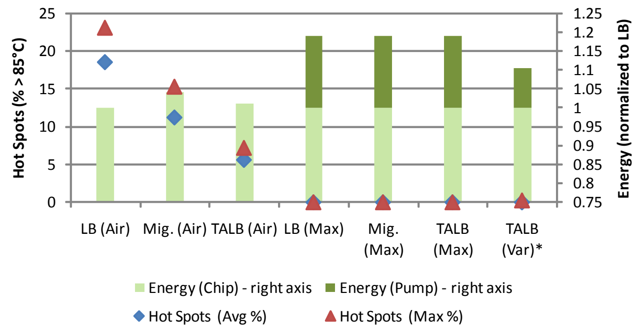

Fig. 6. Hot spots (left-axis) and energy (right-axis) for all the policies. (*) denotes our novel policy.

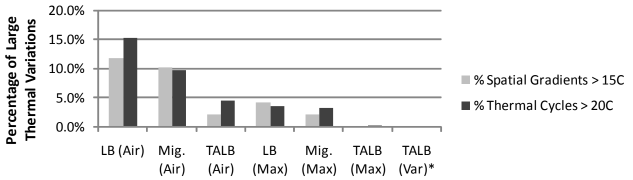

Fig. 7. Thermal variations (with DPM). (*) denotes our novel policy.

temporal variations in temperature, respectively, for all the policies. We evaluate the spatial gradients by computing the maximum difference in temperature among all the units at every sampling interval. Similarly, for thermal cycles, we keep a sliding history window for each core, and compute the cycles with magnitude larger than 20^oC. In the experiments in Figure 7, we run DPM in addition to the thermal management policy. Our weighed load balancing technique (TALB) is able to minimize both temporal and spatial thermal variations much more effectively than other policies.

For our multicore 3D systems, we compute throughput as the performance metric. Throughput is the number of threads completed per given time. As we run the same workloads in all experiments, when a policy delays execution of threads, the resulting throughput drops. Most policies we have run in this work have a similar throughput in comparison to default load balancing. Thread migration, however, reduces the throughput especially for high-utilization workloads because of the performance overhead of frequent temperature-triggered migrations. The overhead of migration disappears for the liquid cooled system, as the coolant flowing at the maximum rate is able to prevent all the hot spots, and therefore no temperature-triggered migrations occur. Figure 8 compares the policies in terms of energy and performance, both for the air and liquid cooling systems. For 3D systems with liquid cooling, our technique is able to improve the energy savings without any effect on the performance.

VI. Conclusion

Liquid cooling is a promising solution to overcome the elevated thermal problems of 3D chips, but intelligent control of the coolant flow rate is needed to achieve energy-efficiency. In this paper we have presented a novel controller that is able to select a minimal coolant injection rate to guarantee a bounded maximum temperature in 3D MPSoCs under variable workload conditions. Our method minimizes the energy consumption of the liquid cooling subsystem. The controller is integrated with a novel job scheduler which balances the temperature across the system to prevent the thermal variations and to improve cooling efficiency. Our experimental results show that the joint flow rate control and job scheduling technique maintains the temperature below the desired levels, while reducing cooling energy by up to 30% and achieving overall energy savings up to 12%.

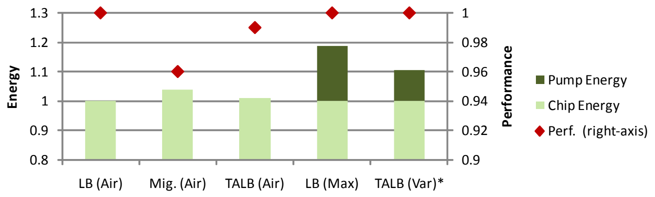

Fig. 8. Performance and Energy. (*) denotes our novel policy.

Acknowledgements

This research has been partially funded by the Nano-Tera.ch NTF Project CMOSAIC (ref. 123618), which is financed by the Swiss Confederation and scientifically evaluated by SNSF. This work has also been funded by Sun Microsystems, UC MICRO, Center for Networked Systems at UCSD, MARCO/DARPA GSRC, and NSF Greenlight.

References

[1] D. Atienza, P. D. Valle, G. Paci, F. Poletti, L. Benini, G. D. Micheli, and J. M. Mendias. A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. In *DAC*, 2006.

[2] A. Bhunia, K. Boutros, and C.-L. Che. High heat flux cooling solutions for thermal management of high power density gallium nitride HEMT. In *Inter Society Conference on Thermal Phenomena*, 2004.

[3] T. Brunschwiler et al. Direct liquid-jet impingement cooling with micron-sized nozzle array and distributed return architecture. In *ITHERM*, 2006.

[4] T. Brunschwiler et al. Interlayer cooling potential in vertically integrated packages. *Microsyst. Technol.*, 2008.

[5] A. K. Coskun, T. Rosing, and K. Gross. Proactive temperature balancing for low-cost thermal management in mpsocs. In *ICCAD*, 2008.

[6] A. K. Coskun, T. S. Rosing, J. Ayala, and D. Atienza. Modeling and dynamic management of 3D multicore systems with liquid cooling. In *IFIP/IEEE International Conference on VLSI (VLSI-SoC)*, 2009.

[7] A. K. Coskun, T. S. Rosing, J. Ayala, D. Atienza, and Y. Leblebici. Dynamic thermal management in 3D multicore architectures. In *Design Automation and Test in Europe (DATE)*, 2009.

[8] A. K. Coskun, T. S. Rosing, K. A. Whisnant, and K. C. Gross. Static and dynamic temperature-aware scheduling for multiprocessor socs. *IEEE Transactions on VLSI*, 16(9):1127–1140, Sept. 2008.

[9] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA*, 2006.

[10] K. C. Gross and K. E. Humenik. Sequential probability ratio test for nuclear plant component surveillance. *Nuclear Technology*, 93(2):131–137, Feb 1991.

[11] M. Healy et al. Multiobjective microarchitectural floorplanning for 2-d and 3-d ICs. *IEEE Transactions on CAD*, 26(1), Jan 2007.

[12] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *ISLPED*, pages 217–222, 2003.

[13] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha. HybDTM: a coordinated hardware-software approach for dynamic thermal management. In *DAC*, pages 548–553, 2006.

[14] Laing 12 volt DC pumps datasheets. http://www.lainginc.com/pdf/DDC3_LTI_USletter_BR23.pdf.

[15] H. Lee et al. Package embedded heat exchanger for stacked multi-chip module. In *Transducers, Solid-State Sensors, Actuators and Microsystems*, 2003.

[16] A. Leon et al. A power-efficient high-throughput 32-thread SPARC processor. *ISSCC*, 2006.

[17] Z. Li et al. Integrating dynamic thermal via planning with 3D floorplanning algorithm. In *ISPD*, pages 178–185, 2006.

[18] R. McDougall, J. Mauro, and B. Gregg. Solaris Performance and Tools. *Sun Microsystems Press*, 2006.

[19] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ISCA*, 2003.

[20] SLAMD Distributed Load Engine. www.slamd.com.

[21] H. Su et al. Full-chip leakage estimation considering power supply and temperature variations. In *ISLPED*, 2003.

[22] D. Tarjan, S. Thoziyoor, and N. P. Jouppi. CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories Palo Alto, 2006.

[23] D. B. Tuckerman and R. F. W. Pease. High-performance heat sinking for VLSI. *IEEE Electron Device Letters*, 5:126–129, 1981.

[24] C. Zhu, Z. Gu, L. Shang, R. P. Dick, and R. Joseph. Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Transactions on CAD*, 27(8):1479–1492, August 2008.