Optimizing performance per watt on GPUs in High Performance Computing: temperature, frequency and voltage effects

D. C. Price[a,*], M. A. Clark[a,b,c], B. R. Barsdell[a], R. Babich[b], L. J. Greenhill[a]

[a]*Harvard-Smithsonian Center for Astrophysics, MS 42, 60 Garden Street, Cambridge MA 01238 USA*
[b]*NVIDIA, 2701 San Tomas Expy, Santa Clara, CA 95050 USA*
[c]*Caltech MC 249-17 1200 East California Blvd, Pasadena CA 91125*

Abstract

The magnitude of the real-time digital signal processing challenge attached to large radio astronomical antenna arrays motivates use of high performance computing (HPC) systems. The need for high power efficiency (performance per watt) at remote observatory sites parallels that in HPC broadly, where efficiency is an emerging critical metric. We investigate how the performance per watt of graphics processing units (GPUs) is affected by temperature, core clock frequency and voltage. Our results highlight how the underlying physical processes that govern transistor operation affect power efficiency. In particular, we show experimentally that GPU power consumption grows non-linearly with both temperature and supply voltage, as predicted by physical transistor models. We show lowering GPU supply voltage and increasing clock frequency while maintaining a low die temperature increases the power efficiency of an NVIDIA K20 GPU by up to 37-48% over default settings when running xGPU, a compute-bound code used in radio astronomy. We discuss how temperature-aware power models could be used to reduce power consumption for future HPC installations. Automatic temperature-aware and application-dependent voltage and frequency scaling (T-DVFS and A-DVFS) may provide a mechanism to achieve better power efficiency for a wider range of codes running on GPUs.

Keywords: performance per watt, power efficiency, radio astronomy, HPC, GPU, DVFS

1. Introduction

Power efficiency — computational performance per unit of power used — is a crucial design factor within HPC. Power consumption is often a limiting

*Corresponding author
Email addresses: `dprice@cfa.harvard.edu` (D. C. Price), `mclark@nvidia.com` (M. A. Clark), `greenhill@cfa.harvard.edu` (L. J. Greenhill)

factor for HPC systems, with current generation petaflop machines already requiring power budgets >1 MW to operate. For example, the Piz Daint system at the Swiss National Supercomputing Centre consumes 1.76 MW of power, despite being the greenest petaflop machine on the Green500 list[1] (#5 overall, 7.7 PFLOPS theoretical maximum).

In order to build Exascale systems ($>10^{18}$ FLOPS), increasing the achieved performance per watt of high performance computing (HPC) hardware is of paramount importance for several reasons. Firstly and foremostly, the more power consumed, the more the system costs to operate. Secondly, generation and distribution of power is non-trivial on megawatt scales; for example, if we extrapolate power draw from Piz Daint, an Exascale machine would require $\sim$230 MW, thus requiring its own power plant. In addition, the waste heat of massive HPC systems poses an engineering challenge: this heat must be removed to avoid compute nodes overheating and failing. The cooling systems of large HPC installations often require significant amounts of power themselves.

As much as 50% of the total power consumption of HPC systems is consumed by infrastructure like cooling, power conditioning and lighting [1, 2]. Decreasing compute power consumption in turn decreases infrastructure power consumption and as such is the most promising way to increase overall power efficiency.

Machines based upon graphic processing units (GPUs) dominate the Green-500 list (June 2014), with all of the top 15 machines featuring NVIDIA Kepler GPUs. Indeed, two of the top 10 most powerful computers on the June 2014 Top500 list[2], Titan (#2) and Piz Daint (#6), are based on Kepler GPUs. As such, the question of how best to increase GPU power efficiency is pressing.

In this paper, we investigate how performance per watt can be optimized for an NVIDIA K20 GPU. We approach the problem by considering the physical processes that govern transistor performance; in particular, how temperature, supply voltage, and clock frequency affect power efficiency.

1.1. Power efficiency and GPUs in radio astronomy

The proposed all-sky imaging element of the Long Wavelength Array (LWA) [3], with $\sim 0.1\text{km}^2$ collecting area and the proposed Square Kilometre Array (SKA) telescope[3] will demand computation at peta- and exa-scale processing respectively (e.g. [4]). Both instruments would produce large enough volumes of data that a substantial fraction of processing must be performed in real-time and in close proximity to the telescope(s). Power efficiency is of particular concern within radio astronomy as most telescopes are in remote locations and must operate under stricter power limits. As such, custom hardware is often built to perform the required digital signal processing tasks. For example the EVLA WIDAR and ALMA correlators [5, 6] are very capable and power efficient

[1]http://www.green500.org/lists/green201406
[2]http://www.top500.org/lists/2014/06/
[3]http://www.skatelescope.org

signal processing systems, but they lack the flexibility afforded by architectures where processing is performed on general purpose computing platforms; they also took over a decade to design and implement. It has been shown that GPU-based signal processing systems for radio astronomy can be designed and deployed in a fraction of this time, see for example [7].

GPUs are well-suited to many of the signal processing tasks required in radio astronomy, such as correlation, time-series dedispersion and radio synthesis image processing [8, 9]. If power efficiency challenges can be met, then a GPU-based HPC system would be an attractive solution for SKA signal processing. In Magro et. al. [10], a GPU-based implementation of the SKA1-Low central signal processor (a subsystem of the full SKA) is calculated to require $\sim$335 kW, based on current NVIDIA Kepler GPU architecture. This assumes a GPU power efficiency of 12 GFLOPS/W; based on the results presented in Section 3 below, this is conservative: we report 18.3 GFLOPS/W for the `xGPU` cross-correlation code, after temperature-aware tuning of GPU supply voltage and frequency. Whether or not GPUs are considered for SKA1-Low signal processing hardware will depend upon demonstrating that GPUs can achieve acceptable power efficiency within the next few years.

1.2. Power leakage

Understanding the factors that affect power efficiency is vital for optimizing systems to minimize power consumption. While the architecture of digital computational devices differ, the physical processes that underlie power usage are common across all architectures. These processes can be broadly broken into two categories: static and dynamic. That is, total power usage $P_{\rm sys}$ is given by

$$P_{\rm sys} = P_{\rm static} + P_{\rm dynamic}. \tag{1}$$

The dynamic power is the power consumed in switching logic states, and for a single logic component is given by

$$P_{\rm dynamic} = C V_{\rm dd}^2 f_{\rm clock},$$

where C is the load capacitance, $V_{\rm dd}$ is the voltage swing and $f_{\rm clock}$ is the switching frequency. For a chip with many logic components, the dynamic power is the sum of the contributions of all $N_{\rm c}$ components:

$$P_{\rm dynamic} = \sum_{\rm n=1}^{N_{\rm c}} C_{\rm n} V_{\rm n}^2 f_{\rm n}, \tag{2}$$

which for devices with a single clock domain (i.e. switching frequency), voltage swing $V_{\rm dd}$, and identical logic components simplifies to $N_{\rm c} C V_{\rm dd}^2 f_{\rm clock}$.

Static power, also known as *leakage power*, is consumed regardless of transistor switching and is due to current leakage. Briefly, there are four main sources of leakage current[4] in a CMOS transistor [11]

[4]Conversion to power is given simply by $P = VI$.

1. **Reverse-biased junction leakage current** (I_{rev}): leakage from source or drain to the substrate through reverse-biased diodes when a transistor is off.
2. **Gate-induced drain leakage** (I_{GIDL}): high field effect at drain junctions of MOS transistors results in electrons being collected by the drain and holes being swept out to the substrate, resulting in GIDL current.
3. **Gate direct-tunneling leakage** (I_{GDTL}): quantum tunneling of electrons through oxide insulation to the substrate.
4. **Subthreshold leakage** (I_{sub}): current resulting from conduction between source and drain through a transistor channel when gate-to-source voltage (V_{GS}) is below the threshold voltage (V_{thr}). The amount of leakage is strongly temperature-dependent.

More detailed discussion of these mechanisms can be found in [11–15].

For sub-micrometer processes (i.e. most current-generation compute architectures), subthreshold leakage is the dominant mechanism, and is the reason why Dennard scaling has broken down. Dennard scaling [16] states that as transistors get smaller their power density stays constant and power use remains proportional to area. However, as transistor size decreases, subthreshold leakage increases, which increases power leakage and chip heating. The breaking of Dennard scaling has led to chip manufacturers focussing on multi-core processors instead of frequency scaling to increase performance.

It is informative to consider an analytical expression for subthreshold leakage. As shown in [15], I_{sub} of a MOS device can be expressed as

$$I_{\text{sub}} = A_{\text{s}} \frac{W}{L} \left(\frac{kT}{q} \right)^2 e^{\frac{q(V_{\text{GS}} - V_{\text{thr}})}{nkT}}, \tag{3}$$

where A_{s} is a technology-dependent constant, W and L are device's effective channel width and length, and n is the transistor's subthreshold swing coefficient. The quantity kT/q is the thermal voltage, where k is Boltzmann's constant, q is the charge of an electron, and T is temperature. The threshold voltage V_{thr} is also a (non-linear) function of temperature, decreasing with increasing temperature.

Equation 3 predicts that subthreshold leakage current exhibits a non-linear temperature dependence, proportional to $I_{\text{sub}} \propto T^2 e^{-b/T}$, where b is a positive constant. Here, the exponent is necessarily negative, as $(V_{\text{GS}} - V_{\text{thr}}) < 0$ (by definition of subthreshold), $q/k \approx 11605$ K/V, and subthreshold swing $n \geq 1$; it follows that Equation 3 monotonically increases with temperature. This implies that power efficiency of a transistor increases with decreasing temperature, due to suppression of subthreshold leakage. One therefore expects to see performance per watt of GPUs improve as die temperature is lowered.

1.3. Maximizing power efficiency

Maximizing power efficiency η_{pow}, requires simultaneous optimization of power consumption and computational performance. Equations 1, 2 and 3, suggest that to minimize power usage we should push voltages, clock frequencies, and temperatures as low as possible. In tension with this, compute performance, operations per second (N_{OPS}), increases linearly with clock frequency. That is, maximum power efficiency is given by

$$\eta_{\mathrm{pow}} = \frac{N_{\mathrm{OPS}}}{P_{\mathrm{total}}} = \frac{N_{\mathrm{OPS}}}{P_{\mathrm{dynamic}} + P_{\mathrm{static}}}. \tag{4}$$

For a simple chip with full utilization of N_{c} identical compute components, each performing one operation per clock cycle, with a clock frequency f_{clock},

$$N_{\mathrm{OPS}} = N_{\mathrm{c}} f_{\mathrm{clock}}, \tag{5}$$

and we have

$$\eta_{\mathrm{pow}} = \frac{N_{\mathrm{c}} f_{\mathrm{clock}}}{N_{\mathrm{c}} C V_{\mathrm{dd}}^2 f_{\mathrm{clock}} + P_{\mathrm{static}}},. \tag{6}$$

Equation 6 shows that power efficiency is increased when voltage is decreased. Due to the P_{static} term in the denominator, efficiency also increases with clock frequency.

For a complex chip such as a GPU this formalism is a simplification; there are also chip-specific tolerances and thermal dissipation limits that must be considered. Another consideration is that frequency and voltage are generally scaled together, not separately. This is primarily as the speed at which a digital circuit can switch states from low to high — the gate delay time t_{delay} — is

$$t_{\mathrm{delay}} \propto \frac{V_{\mathrm{dd}} T^{\mu}}{(V_{\mathrm{dd}} - V_{\mathrm{thr}})^{\xi}}, \tag{7}$$

where ξ and μ are technology-dependent constants [12]. For 65-nm process, Liao et. al. [12] find $\mu = 1.19$ and ξ=1.2. The temperature dependence of Equation 7 arises as temperature affects carrier mobility and threshold voltage. At higher frequencies, there is more dynamic power usage, so die temperature will in turn increase, forcing higher V_{dd} to maintain suitable t_{delay} (which in turn increases power usage and die temperature).

Nonetheless, the default clock-voltage combination has been shown to be conservative on some GPUs (see [17]). This is particularly true on the NVIDIA Kepler architecture Tesla class cards: both the Tesla K20 and GeForce GTX 780 Ti have GK110 processing units, but the base clock for the K20 is 705 MHz, while the GTX 780 is 863 MHz. One reason for this is differences in TDP (thermal design power), and targeted market; power consumption is of concern for HPC but less so for the gaming market.

1.4. GPU power measurement and modeling

The simplified power efficiency formula presented in Equation 6 is not immediately applicable to GPUs, which feature hierarchical memory, different clock domains, multiple instructions, and dynamic control of voltage and clock frequency (DVFS). As such, there have been many analyses at higher abstraction levels that quantify the power characteristics of GPU hardware and provide models that predict power usage [17–28]. While approaches and resulting power models differ, the general findings of this corpus can be summarized as follows.

Table 1: Comparison of Tesla-class NVIDIA Kepler GPUs that use the GK110 chipset.

	Tesla K20	Tesla K20x	Tesla K40
Chipset	GK110	GK110	GK110B
CUDA cores	2496	2688	2880
Base processor clock	705 MHz	732 MHz	745 MHz
Memory size (GDDR5)	5 GiB	6 GiB	12 GiB
Memory bandwidth	208 GB/s	250 GB/s	288 GB/s
Thermal Design Power (TDP)	225 W	235 W	235 W

Source: http://www.nvidia.com/object/tesla-servers.html

- Power consumption is dependent not only on the type of GPU, but also upon the kernels running on the GPU.

- GPU performance is either memory-bound, or compute-bound; this is well described by the roofline model [29]. Different code optimizations are required for each case.

- Execution of compute instructions require less energy than memory access. For example, a multiply-add (MAD) instruction uses 7-15 times less energy than L1 memory access on an NIVDIA G80 [19].

- Distant (e.g., off-chip) memory access consumes more power than local memory access. These costs can be minimized by exploiting the memory hierarchy.

Our work differs in that we consider temperature, voltage and frequency as independent variables over which to optimize performance per watt. That is, we consider power efficiency $\eta_{\text{pow}} = \eta_{\text{pow}}(V_{\text{dd}}, f_{\text{clock}}, T)$. While voltage and frequency have previously been explored in GPU DVFS studies [17, 28, 30], we explore a larger parameter space. Apart from in Hong et. al. [23], temperature effects on GPU power efficiency have been ignored. This is detrimental to GPU power model accuracy and to achieving optimal power efficiency, as discussed in Liao et. al. [12, 31]. We show that the simplified linear model of Hong et. al. [23] is not sufficient for predicting power usage on current generation GPUs.

We show experimentally that significant power efficiency gains (up to 48%) can be made on current generation hardware by being aware of the physical processes underlying GPU power usage. The remainder of this paper is organized as follows. In Section 2, we introduce the hardware and software used to find optimal power efficiency on an NVIDIA K20 GPU. Our results are then presented in Section 3; this is followed by discussion (Section 4) and conclusions (Section 5).

2. Materials and Methods

2.1. Hardware overview

The work presented here was conducted on "GreenGPU", a custom-built computer system. GreenGPU consists of a Gigabyte GA-Z68MX motherboard with an Intel i7-2600 CPU, 16 GiB of DDR3 RAM, and an NVIDIA Tesla K20 GPU. The default heatsink of the K20 was replaced with an EK-FCTK20 water block, and a Swiftech water cooling system (MCP655) was installed. The specifications of the K20 are shown in Table 1, along with a comparison of other Kepler Tesla class cards that use the GK110 chipset on 28-nm process. The operating system used for testing was 64-bit Linux Ubuntu 12.04 LTS, with NVIDIA GPU driver version 319.37 installed. A Windows 7 partition was also installed in order to run Windows-only GPU firmware modification tools.

2.2. Clock and voltage management

To control the clock frequency and voltage of the K20 GPU, we used three tools: `nvidia-smi`[5], `GPU-Z`[6] and `Kepler BIOS Tweaker`[7]. The `nvidia-smi` utility, or NVIDIA System Management Interface, is a command line utility that uses the NVIDIA Management Library[8] (NVML) for management and control of NVIDIA devices. The `nvidia-smi` tool allows for the GPU core frequency to be altered; the allowed values are dependent upon the GPU (Table 2). In addition to the Tesla-class Kepler GPUs, some GeForce-class cards also use the GK110 chipset; these cards generally have larger TDPs, higher clockrates, and less memory. For example, the GTX 780 Ti has a TDP of 250 W, a base clock of 875 MHz, and 3 GiB GDDR5. The `nvidia-smi` tool also allows for power draw and GPU die temperature to be read from GPU sensors, giving an accurate way to measure temperature and power, with differences between power and temperature reliable to within ± 1 W and $\pm 1°$C. The reported power is the full-board power consumption, which includes memory and voltage regulators.

For finer grain control over core voltage and frequency, and so that we could tune these as independent parameters, we used the `GPU-Z` tool v0.7.7 and `Kepler BIOS Tweaker` tool v1.27. `GPU-Z` is a utility that displays GPU specifications and operating parameters, and allows for GPU firmware to be downloaded from the GPU. The `Kepler BIOS Tweaker` tool allows for modification of the parameters within GPU firmware, such as voltage and clock frequency. While benchmarking was run on the Ubuntu partition of GreenGPU, these two programs were run on the Windows 7 partition. Note that flashing firmware using tools such as `Kepler BIOS Tweaker` will void warranty and can potentially cause damage to the GPU.

[5]https://developer.nvidia.com/nvidia-system-management-interface
[6]http://www.techpowerup.com/gpuz/
[7]http://www.softpedia.com/get/System/Benchmarks/Kepler-BIOS-Tweaker.shtml
[8]https://developer.nvidia.com/nvidia-management-library-nvml

Table 2: Supported core and memory clock pairs for the K20

GDDR5 Freq.	GPU Core Freq.	GPU Core Voltage	
(MHz)	(MHz)	State ID	(mV)
2600	758	V5	987.5-1112.5
	705	V4*	950-1062.5
	666	V3	925-1050
	640	V2	912.5-1025
	614	V1	900-1000
324	324	V0	875 - 875

*Default value

2.3. xGPU cross-correlation code

For benchmarking and power efficiency testing, we used the `xGPU` CUDA code[9] [9]. `xGPU` computes the cross-correlation of time-series data of N inputs and is used for interferometric synthesis imaging in radio astronomy, see for example [7]. It is virtually identical to the BLAS routine CHERK — Complex Hermitian Rank K update — where the $T \times N$ matrix, corresponding to time series data (T dimension) from N antennas is multiplied by its complex conjugate, producing an $N \times N$ Hermitian matrix. The problem is compute-bound because the compute complexity scales as N^2T, whereas the memory traffic scales as $N(T + N)$, assuming perfect caching. The `xGPU` code differs from regular CHERK as it contains domain-specific tweaks: it is designed to process 8-bit integer input, only stores the lower triangle of the correlation matrix, and uses smaller tiles to improve performance for small-N. `xGPU` also has an additional parameter corresponding to the number of frequency channels to process; the problem is trivially parallelizable over frequency channels, so this can be thought of as a batching parameter.

We use the `xGPU` application to do this investigation, partly because that is our domain of interest; however, we also note that given it is compute-bound, it is well suited to our investigation: `xGPU` uses a multi-level tiling algorithm to minimize memory accesses from all memory spaces, and thus achieves a high percentage of peak throughput on the CUDA architecture. Thus, when running this algorithm, most of the power is consumed by the floating point units, and this increases the validity of the simple model in Section 1.2.

`xGPU` has two different modes that were of particular use for this work. The first mode is a benchmark, which computes various performance metrics achieved, such as FLOPS, for a given set of compile-time parameters. The output of the GPU code is also compared against CPU code for validation. The second mode is a power diagnostic loop, in which `xGPU` is fed dummy data and run in an infinite loop, so as to keep the GPU running continuously.

We achieved compute-bound performance using the following compile-time

[9]https://github.com/GPU-correlators/xGPU

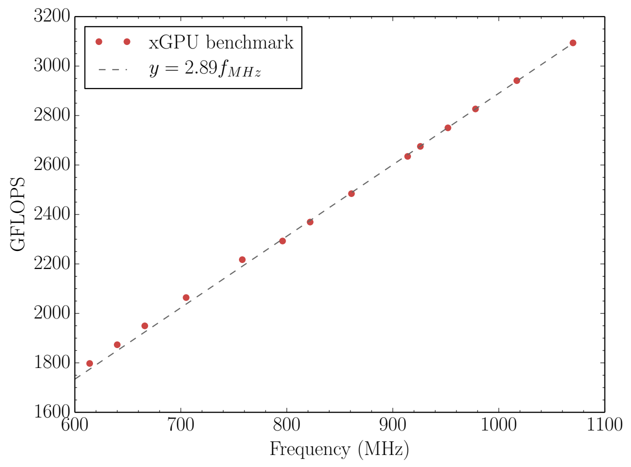


Figure 1: FLOPS achieved running `xGPU` codes on an NVIDIA K20, as a function of GPU core clock frequency.

parameters: `NINPUTS=8192 NSTATION=4096, NFREQUENCY=2, NTIME=2000, NTIME_PIPE=1000, SHARED_ATOMIC_SIZE=8`.

2.4. Performance profiling method

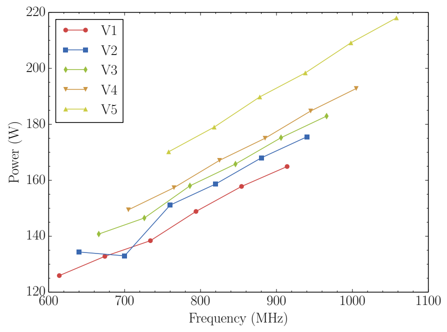
The main parameters used for testing power efficiency in this work were GPU die temperature, GPU core voltage and clock frequency. We used `Kepler BIOS Tweaker` and `nvidia-smi` to modify the GPU core voltage and clock frequency, then we used `xGPU` to benchmark performance. Attempts to vary the memory clock frequency resulted in the GPU being inoperable, so no memory clock adjustments were conducted.

Thermal control of the GPU die was achieved by running `xGPU` in a power loop, while controlling the flow of water through the water cooling system. In order to continuously monitor the temperature and power draw, we used a Python script to parse the output of `nvidia-smi` and to log timestamped power usage and temperature data to file every second. By running this script in tandem with `xGPU`, we tested the performance of the K20 GPU over a variety of core frequency and voltage combinations.

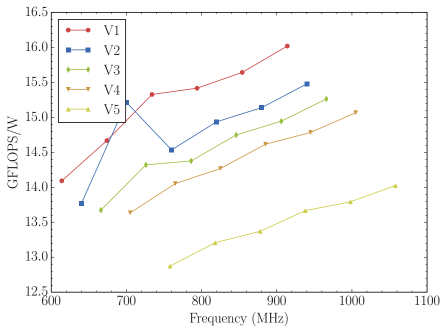
3. Results

3.1. `xGPU` benchmarking

The single-precision computational performance in FLOPS for `xGPU` is shown in Figure 1 for a variety of different clock frequencies between 614-1070 MHz. Note that temperature and voltage do not affect achieved FLOPS. The maximum performance of 3094 GFLOPS was achieved at a clock speed of 1070 MHz.



(a) Measured GPU power usage.



(b) Measured GPU power efficiency.

Figure 2: GPU power usage and efficiency for `xGPU` code running on a K20 GPU, for default voltages (V1-V5) with frequency offsets of 0-300 MHz over default f_{clock} settings (see Table 2).

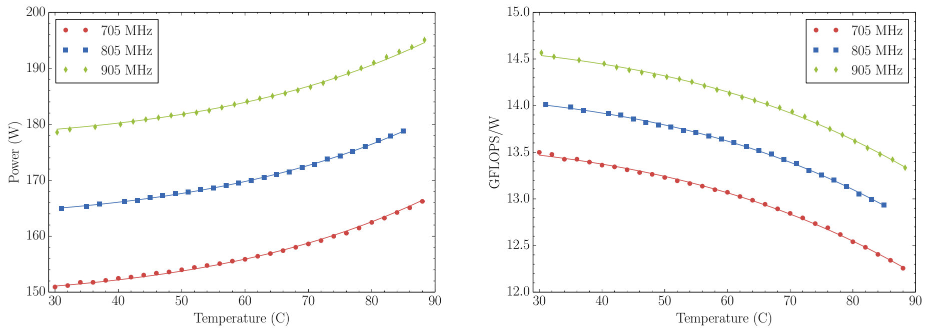
3.2. Overclocking at constant temperature with default voltages

After profiling the computational performance of `xGPU`, we compared power usage of the GPU at different (f_{clock}, V_{dd}) combinations. As shown in Table 2, the K20 has preset frequency-voltage combinations that can be selected with `nvidia-smi`. We applied frequency offsets of 0-300 MHz to these default values, in 60 MHz increments, and then measured the resulting power usage for the `xGPU` code (Figure 2a), and the corresponding power efficiency (Figure 2b). Voltage states are labelled V1-V5, with increasing voltage; for these data, the firmware voltage table was not modified. To account for temperature effects, we held GPU die temperature at 34$\pm$2°C.

The default voltage state (V4) with default frequency f_{clock}= 705 MHz yields a power efficiency of 13.6 GFLOPS/W. We find that the best power efficiency of 16.0 GFLOPS/W is achieved when using the lowest voltage state with f_{clock}= 914 MHz, an increase of 18%. The worst power efficiency is achieved when using the highest voltage level with its default f_{clock}=758 MHz. The dip in power usage at f_{clock}= 705 MHz when in the V2 state is likely due to the GPU selecting a low core voltage within the allowed range (see Table 2).

3.3. Temperature dependence of power efficiency for constant voltage state

Equation 3 predicts that subthreshold leakage current is proportional to $T^2e^{-b/T}$. To investigate this, we compared power usage of the GPU at various die temperatures (Figure 3a),where we have averaged multiple data into bins of $\pm$1°C. In Figure 3a, the clock frequency was set to 705, 805 and 905 MHz, with the default core voltage state (950-1062.5 mV). At all temperatures, power usage changes by a fixed $\sim$ 0.14 W/MHz. As clock frequency does not affect static power P_{static}, the offset between lines corresponds to the dynamic power P_{dynamic} component of the total power usage.



(a) Measured GPU power usage. (b) Measured GPU power efficiency.

Figure 3: GPU power usage and efficiency for `xGPU` code running on a K20 GPU, with default core voltage (V4). Power usage shows a strong non-linear temperature dependence; this decreases performance per watt as temperature increases.

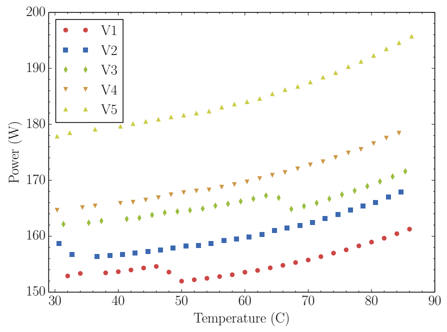
We also see a non-linear increase of power consumption as a function of temperature; the simple linear model as presented in [23] is not sufficient. If we take into account P_{dynamic}, we can fit a model , $P_{\text{static}} = aT^2e^{-b/T} + c$ to all three runs (solid lines). For temperature in Kelvin, a least-square fit yields $a = 1.00 \pm 0.23$, $b = 3209.7 \pm 83.7$, $c = (148.9 \pm 0.2,\ 162.7 \pm 0.2,\ 176.9 \pm 0.2)$ for 705, 805 and 905 MHz, respectively.

Power efficiency is improved as clock frequency is increased (Figure 3b), from 705 MHz to 805 MHz, and again to 905 MHz. If we compare worst case (705 MHz at 90°C) to best case (905 MHz at 30°C), we see an 18% difference in power efficiency. If we compare at constant T=30°C, the performance at 905 MHz is 14.6 GFLOPS/W, as opposed to 13.5 GFLOPS/W at 705 MHz; an 8.1% increase.

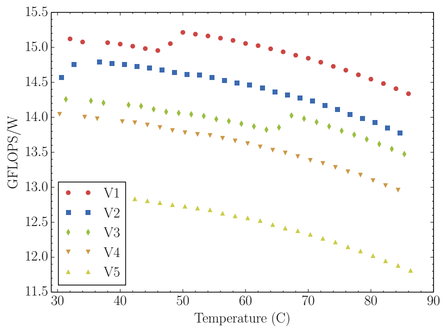
3.4. Constant frequency, modified voltage

The default voltage states of the K20 are not fixed voltages, but rather a range (Table 2). To investigate the effect of voltage on power efficiency, we reprogrammed the K20's firmware so that the GPU core voltages V1-V5 were fixed to 900, 912.5, 925, 950 and 987.5 mV, the lower bound of the default voltage ranges (Table 2). Power consumption as a function of temperature for the modified voltage levels V1-V5 is shown in Figure 4a, for a constant clock frequency of f_{clock} =800 MHz; this is converted into performance per watt in Figure 4b. We see that the highest core voltage state results in the worst power efficiency (at 30°C) of 12.6 GFLOPS/W, in comparison to 14.7 GFLOPS/W for the lowest voltage state, and that as voltage is increased, power efficiency decreases. This corresponds to a 16.7% difference in power efficiency between best and worst cases.

Apparent in Figure 4a are jumps in the power usage, where the reported power consumption drops unexpectedly. These drops are repeatable and occur



(a) Measured GPU power usage.



(b) Measured GPU power efficiency.

Figure 4: GPU power usage and efficiency for `xGPU` code running on a K20 GPU, with a core clock frequency of 800 MHz and varying voltage levels (see Table 2). Lower voltage levels correspond to higher power efficiency.

at different temperatures for different voltage states. We are uncertain as to the cause; however, `nvidia-smi` does not report any clock throttling and no decrease in performance (i.e. FLOPS) is seen. The altered voltage table (as written in the GPU's firmware) did not allow for different voltage states, and the K20 GPU does not employ temperature-dependent voltage scaling. As such, we conclude that this is due to an unknown off-chip (i.e. off-processor) effect. A possible explanation is that this is due to current-dependent efficiencies in power delivery of the regulators that supply the GPU die with power.

3.5. Tuning voltage and frequency

The best performance per watt is achieved when undervolting and overclocking the GPU, as predicted in Section 1.3 (Table 3). At 900 mV, `xGPU` code execution fails and the GPU froze when attempting to run the code at 1005 MHz. At 955 MHz, the code ran successfully but the output failed verification when GPU temperature was above 70°C; that is, it did not match the output of reference CPU code. No errors were found for temperatures below 70°C. At 875 mV, we achieved a maximum clock frequency of 905 MHz, but again found that GPU output did not pass verification for temperatures above 70°C. Due to the temperature dependence on power, temperature is held fixed at 50°C. The frequencies shown in Table 3a correspond to a fixed frequency offset of 312 MHz over the default K20 clock speeds. Similarly, the voltages shown in Table 3b correspond to the lower bound of the K20's default voltage levels V1-V5. Table 3c compares the V4 default (950-1062.5 mV) against a lowered voltage of 875 mV.

For $(V, f_{\text{clock}}, T) = (875\text{ mV}, 905\text{ MHz}, 30°\text{C})$, we achieved 18.3 GFLOPS/W for the `xGPU` code. For comparison, the K20 default of $(V, f_{\text{clock}}, T) =$ (950-1062.5 mV, 705 MHz, 30°C) yields 13.5 GFLOPS/W for the same code, degrading to 12.4 GFLOPS/W at 90°C. This means that by controlling GPU temperature, voltage, and clock frequency, we are able to increase performance per watt by 37-48% over default settings.

4. Discussion

Our results show that temperature has a nontrivial impact on GPU power efficiency. This is primarily due to leakage current, which scales in proportion to $T^2 e^{-b/T}$. Optimum power efficiency is achieved when GPU supply voltage is lowered and clock frequency is raised, while temperature is kept low. We find efficiency can be increased by as much as 48% on an NVIDIA K20 through this technique.

We have demonstrated a $\sim$30 W decrease on a GPU power consumption of 154 W by changing GPU core voltage state, a 20% reduction (Table 3c). Coupled with an increase in GPU clock frequency, we were able to increase performance from 2064 GFLOPS to 2636 GFLOPS for our code — a 28% increase — while simultaneously decreasing power usage by 10 W.

For large installations, even a small change in power efficiency can have significant cost benefits. For example, the Titan supercomputer has 18,688 K20X GPUs; if each GPU consumed 10 W less power, over 0.18 MW of power would be saved. At \$0.10/kWh, this equates to \$18.68/h, or roughly \$163k/yr. In the following subsections, we discuss the relevance of results presented here to future HPC installations and GPU design.

We have presented results from a single GPU. In actuality, the same chips from within the same process will have a distribution of values (dynamic power, leakage power, etc.), so a degree of conservatism is required in setting device parameters for mass production. Allowing clock frequency and core voltage to be set at run-time by the user, or adjusted automatically using dynamic voltage and frequency scaling techniques (DVFS), may provide a mechanism with which to boost power efficiency over conservative defaults.

4.1. Application-aware DVFS

When clock frequencies are chosen for a GPU, it is typical to choose clock frequencies that can support a wide range range of workloads. For example, codes such as `DGEMM` (double-precision general dense matrix multiply) consume more power than the single-precision `xGPU` code, but must still run within the TDP at default clock frequency. It follows that there will always be a significant boost in clock frequencies possible for applications that do not run close to the TDP limit. One could imagine a control system that automatically adjusts clock frequency and voltage, depending upon application and desired performance optimization (e.g. FLOPS/W or FLOPS). This would be a form of DVFS. Such an application-dependent frequency and voltage scaling system *(A-DVFS)* could offer a way to automatically boost power efficiency and performance of codes: by analyzing their power usage and adjusting clock rates and voltages according to a user's desired optimization. Such a system could also accommodate applications that require perfect load balancing or reduced system jitter by setting clock

Table 3: Power efficiency (at 50°C) and benchmarks for `xGPU` code as a function of voltage and clock frequency.

(a) Power efficiency for maximum attained clock frequencies at given voltage levels.

Clock Frequency (MHz)	Voltage (mV)	Power (W)	Benchmark (GFLOPS)	Power efficiency (GFLOPS/W)
1070	987.5	222.2	3094	13.9
1017	950.0	197.1	2940	14.9
978	925.0	186.9	2826	15.1
952	912.5	175.6	2750	15.7
926	900.0	168.5	2674	15.9
905	875.0	144.4	2636	18.3

(b) Power efficiency and benchmarks for a fixed 926 MHz clock at varying voltage levels.

Clock Frequency (MHz)	Voltage (mV)	Power (W)	Benchmark (GFLOPS)	Power efficiency (GFLOPS/W)
926	987.5	199.0	2674	13.4
926	950.0	183.1	2674	14.6
926	925.0	179.4	2674	14.9
926	912.5	172.0	2674	15.5
926	900.0	168.5	2674	15.9

(c) Comparison of power efficiency and benchmarks at default voltages to a lowered voltage of 875 mV, for various clock frequencies.

Clock Frequency (MHz)	Voltage (mV)	Power (W)	Benchmark (GFLOPS)	Power efficiency (GFLOPS/W)
705	950.0 - 1062.5	153.9	2064	13.4
805	950.0 - 1062.5	167.5	2330	13.9
905	950.0 - 1062.5	181.6	2636	14.5
705	875.0	122.1	2064	16.9
805	875.0	132.6	2330	17.5
905	875.0	144.4	2636	18.1[a]

[a]output from the GPU does not pass validation for temperatures >70°C.

frequencies uniformly across all devices used by the application, although the highest achievable frequency will be limited to that of the slowest device.

Indeed, the NVIDIA GPU Boost feature[10], launched with the K40 series GPU, allows users to select from two preset higher clocks though `nvidia-smi`, boosting performance for codes that run below TDP. GPU Boost is implemented differently on the GeForce-class gaming cards: core frequency is scaled to maintain card power consumption close to TDP. Adding similar dynamic frequency scaling functionality to server-class GPUs may increase both power efficiency and performance for codes with low power consumption.

4.2. Temperature-aware DVFS

Temperature and TDP limit the range of clock frequencies and supply voltage combinations. The default settings for GPUs are chosen specifically to ensure that neither temperature or TDP tolerances are exceeded for any application. In contrast, best power efficiency occurs when voltages are lowered and clock frequency raised in accordance with operating temperature.

A hypothetical temperature-aware voltage and frequency scaling system *(T-DVFS)* could raise and lower core voltages automatically, based on the GPU die temperature. If cooling systems maintained lower temperatures, the T-DVFS system would accordingly lower voltage, increasing power efficiency.

4.3. Cooling Systems

Optimization of overall power efficiency of a GPU-based HPC system depends also on the power consumed by cooling subsystems. In our tests, we used a water-based direct cooling system, as it is a more efficient cooling technique than forced air cooling; Januszewskia et al. report that water-based cooling systems can reduce the total power consumed by a server room by more than 15% [32]. For our single-node system, we achieve better power efficiency even without factoring in any power savings from the water-based cooling system.

For multi-node systems with greater power dissipation, how to achieve optimal performance is less clear. Should GPUs be only just kept within operational tolerances, or should they be maintained at a lower temperature? Let us start by considering only GPU die temperature as a free parameter to optimize. In this case, there is a direct trade-off between GPU power consumption and cooling system power consumption. For example, if a cooling systems consumed an extra 100 W per GPU to maintain a temperature of 30°C instead of 80°C, this would outweigh the ~30 W savings in GPU power consumption reported here, resulting in an overall decrease in power efficiency (see Section 3.3).

While warm water cooling techniques show great promise, it should be noted that energy savings within the cooling systems may not correspond to optimal overall power efficiency. An IBM Aquasar system demonstrated an exergetic

[10]http://www.nvidia.com/content/PDF/kepler/nvidia-gpu-boost-tesla-k40-06767-001-v02.pdf

efficiency increase of 34% through use of warm water (60°C) cooling [33], although power consumption of electronics increased by $7 \pm 1\%$ as the coolant temperature increases from 30°C to 60°C.

Things get more interesting once we start to consider that lower temperatures allow higher clock rates for a given supply voltage. In Section 3.5 above, we showed that the difference between worst-case default and best-case power efficiencies for the K20 GPU running `xGPU` code is 48%; we did not consider the power required for cooling systems, P_{cool}. If we modify Equation 6 to include P_{cool} and other infrastructure sources, we instead wish to optimize

$$\eta_{\mathrm{pow}}(V, f, T) = \frac{N_{\mathrm{OPS}}(V, f, T)}{P_{\mathrm{sys}}(V, f, T) + P_{\mathrm{cool}}(T) + ...}, \tag{8}$$

where the denominator is the sum of the power over the entire system. Here, we have explicitly written P_{sys} and P_{cool} as functions of temperature. Using Equation 8 as a basis for finding optimal power efficiency for a given code differs from past techniques as it considers the system as a whole, with regards to the fundamental physics that governs power usage of the underlying microarchitecture.

A novel aspect of Equation 8 is that it predicts that lowering temperature may lead to increased power efficiency, which appears somewhat in conflict to previous findings that report lower data center energy consumption at higher temperatures (e.g. [33, 34]). There are two main reasons this discrepancy arises. Firstly, general-purpose data centers focus on optimizing power usage effectiveness (PUE, [35]), as opposed to performance per watt, which is of more interest to HPC systems. PUE is defined as

$$\mathrm{PUE} = \frac{\text{total facility energy}}{\text{IT equipment energy}},$$

where total facility energy is the data center's total energy usage, and IT equipment energy is the sum of all computing, storage and network equipment energy usage. Unlike performance per watt, PUE does not directly consider the computational performance of a system. Secondly, previous comparisons between cooling methods do not account for temperature-dependent optimization of supply voltage and clock frequency.

5. Conclusions

One of the main challenges facing exascale HPC is dramatically reducing the power usage of large HPC systems. We have shown that temperature-aware optimization of core clock frequency and supply voltage can increase performance of a GPU code by up to 48% on an NVIDIA Tesla K20. This increase was achieved by increasing the GPU clock frequency and decreasing supply voltage while maintaining a die temperature of 30°C.

It is taken for granted that code must be optimized for different architectures in order to fairly compare compute performance. In contrast, when optimizing power efficiency for HPC systems, the effect of temperature upon optimal GPU core frequency and voltage is generally not considered. Two possible mechanisms that may allow for better performance per watt are temperature-aware and application-dependent frequency and voltage scaling (T-DVFS and A-DVFS). Incorporation of these or similar techniques into future GPU DVFS systems may yield greater computational performance with reduced power consumption by automatically tuning core frequency and voltage with consideration of both application code and thermal environment.

Acknowledgements

The authors acknowledge support from NSF grants PHYS-080357, AST-1106059, and OIA-1120587. BB thanks the NVIDIA internship program for support.

[1] B Przywara and S Weeren. Energy efficient servers in Europe. Technical report, The Efficient Servers Consortium, 2007.

[2] G I Meijer. Cooling Energy-Hungry Data Centers. *Science*, 328:318–319, April 2010.

[3] S W Ellingson, G B Taylor, J Craig, J Hartman, J Dowell, C N Wolfe, T E Clarke, B C Hicks, N E Kassim, P S Ray, L J Rickard, F K Schinzel, and K W Weiler. The LWA1 Radio Telescope. *Antennas and Propagation, IEEE Transactions on*, 61(5):2540–2549, 2013.

[4] P Chris Broekema, Rob V van Nieuwpoort, and Henri E Bal. ExaScale high performance computing in the square kilometer array. In *workshop on High-Performance Computing for Astronomy Data*, page 9, New York, New York, USA, 2012. ACM Press.

[5] R Perley, P Napier, J Jackson, B Butler, B Carlson, D Fort, P Dewdney, B Clark, R Hayward, S Durand, M Revnell, and M McKinnon. The Expanded Very Large Array. *Proceedings of the IEEE*, 97(8):1448–1462, 2009.

[6] A Wootten and A Thompson. The Atacama Large Millimeter/Submillimeter Array. *Proceedings of the IEEE*, 97(8):1463–1471, 2009.

[7] J Kocz, L J Greenhill, B R Barsdell, G Bernardi, A Jameson, M A Clark, J Craig, D Price, G B Taylor, F K Schinzel, and D Werthimer. A Scalable Hybrid FPGA/GPU FX Correlator. *Journal of Astronomical Instrumentation*, 03(01):1450002, March 2014.

[8] B R Barsdell, D G Barnes, and C J Fluke. Analysing astronomy algorithms for graphics processing units and beyond. *Monthly Notices of the Royal Astronomical Society*, 408(3):1936–1944, November 2010.

[9] M A Clark, P L Plante, and L J Greenhill. Accelerating radio astronomy cross-correlation with graphics processing units. *International Journal of High Performance Computing Applications*, 27(2):178–192, May 2013.

[10] Alessio Magro, Kristian Zarb Adami, and Steve Ord. Suitability of NVIDIA GPUs for SKA1-Low. *arXiv.org*, 1407.4698v3, 2014.

[11] Farzan Fallah and Massoud Pedram. Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits. *IEICE Transactions on Electronics*, E88-C(4):509–519, April 2005.

[12] Weiping Liao, Lei He, and K M Lepak. Temperature and supply Voltage aware performance and power modeling at microarchitecture level. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(7):1042–1053, 2005.

[13] D Rittman. Nanometer Power Leakage. Technical report, Tayden Design, 2005.

[14] D Brooks, R P Dick, R Joseph, and Li Shang. Power, Thermal, and Reliability Modeling in Nanometer-Scale Microprocessors. *Micro, IEEE*, 27(3): 49–62, 2007.

[15] Yongpan Liu, R P Dick, Li Shang, and Huazhong Yang. Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. In *Design, Automation & Test in Europe Conference & Exhibition, DATE '07*, pages 1–6, 2007.

[16] R H Dennard, F H Gaensslen, V L Rideout, E Bassous, and A R LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, 1974.

[17] Xinxin Mei, Ling Sing Yung, Kaiyong Zhao, and Xiaowen Chu. A measurement study of GPU DVFS on energy conservation. In *Workshop on Power-Aware Computing and Systems*, pages 1–5, New York, USA, 2013. ACM Press.

[18] M Rofouei, T Stathopoulos, and S Ryffel. Energy-aware high performance computing with graphic processing units. In *Conference on Power aware computing and systems*, 2008.

[19] Sylvain Collange, David Defour, and Arnaud Tisserand. Power Consumption of GPUs from a Software Perspective. In *Computational Science–ICCS*, pages 914–923. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[20] R Suda and Da Qi Ren. Accurate Measurements and Precise Modeling of Power Dissipation of CUDA Kernels toward Power Optimized High Performance CPU-GPU Computing. In *Parallel and Distributed Computing, Applications and Technologies, International Conference on*, pages 432–438, 2009.

[21] S Huang, S Xiao, and W Feng. On the energy efficiency of graphics processing units for scientific computing. *Parallel & Distributed Processing (IPDPS), IEEE International Symposium on*, pages 1–8, 2009.

[22] Da Qi Ren and Reiji Suda. Investigation on the power efficiency of multi-core and GPU Processing Element in large scale SIMD computation with CUDA. In *International Conference on Green Computing (Green Comp)*, pages 309–316. IEEE, 2010.

[23] Sunpyo Hong, Hyesoon Kim, Sunpyo Hong, and Hyesoon Kim. *An integrated GPU power and performance model*, volume 38. ACM, New York, USA, June 2010.

[24] Y Jiao, H Lin, P Balaji, and W Feng. Power and Performance Characterization of Computational Kernels on the GPU. In *Green Computing and Communications (GreenCom), IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 221–228, 2010.

[25] Hitoshi Nagasaka, Naoya Maruyama, Akira Nukada, Toshio Endo, and Satoshi Matsuoka. Statistical power modeling of GPU kernels using performance counters. In *International Conference on Green Computing (Green Comp)*, pages 115–122. IEEE, 2010.

[26] K Kasichayanula, D Terpstra, P Luszczek, S Tomov, S Moore, and G D Peterson. Power Aware Computing on GPUs. *Application Accelerators in High Performance Computing (SAAHPC), Symposium on*, pages 64–73, 2012.

[27] Jianmin Chen, Bin Li, Ying Zhang, Lu Peng, and Jih-Kwon Peir. Statistical GPU power analysis using tree-based methods. In *International Green Computing Conference and Workshops (IGCC)*, pages 1–6, 2011.

[28] Rong Ge, R Vogt, J Majumder, A Alam, M Burtscher, and Ziliang Zong. Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU. In *Parallel Processing (ICPP), 42nd International Conference on*, pages 826–833, 2013.

[29] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, April 2009.

[30] Cedric Nugteren, Gert-Jan van den Braak, and Henk Corporaal. Roofline-aware DVFS for GPUs. In *International Workshop*, pages 8–10, New York, New York, USA, 2014. ACM Press.

[31] Weiping Liao and Lei He. Coupled Power and Thermal Simulation with Active Cooling. In *Power-Aware Computer Systems*, pages 148–163. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[32] R Januszewskia, L Gillyb, E Yilmazc, and A Auweterd. Cooling–making efficient choices. Technical report, Partnership for Advanced Computing in Europe, 2013.

[33] Severin Zimmermann, Ingmar Meijer, Manish K Tiwari, Stephan Paredes, Bruno Michel, and Dimos Poulikakos. Aquasar: A hot water cooled data center with direct energy reuse. *Energy*, 43(1):237–245, July 2012.

[34] n.d. URL `https://www.google.com/about/datacenters/efficiency/internal/#temperature`.

[35] D Azevedo, D A French, and E N Power. PUE^{TM}: A Comprehensive Examination of the Metric. Technical report, The Green Grid, 2012.