Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon

Jennifer Switzer  
jfswitze@ucsd.edu  
UC San Diego  
La Jolla, CA, USA

Gabriel Marcano  
gmarcano@ucsd.edu  
UC San Diego  
La Jolla, CA, USA

Ryan Kastner  
kastner@ucsd.edu  
UC San Diego  
La Jolla, CA, USA

Pat Pannuto  
ppannuto@ucsd.edu  
UC San Diego  
La Jolla, CA, USA

1 INTRODUCTION

Manufacturing electronic devices is an energy-intensive process. For devices with lower utilization, such as consumer-class electronics or over-provisioned servers, manufacturing dominates the lifetime carbon footprint [17]. This is especially true in devices with short use cycles. In the United States alone, 150 million smartphones are discarded each year, amounting to one phone discarded per person every two years [8]. As a result, manufacturing accounts for 70 – 85% of the lifetime carbon footprint of a smartphone [17, 31].

Consumer electronics are also becoming increasingly powerful—the performance of recent smartphones rivals or exceeds that of an Intel Core-i3 processor (Figure 1). Yet, phones are often discarded despite being completely (or partially) operational. Otherwise functional electronic devices are retired prematurely due to technical, style, or planned obsolescence [33]. Compare this to other high-priced devices with entire ecosystems of functional obsolescence, e.g., cars, which are resold until they are “driven into the ground.”

Of course, the performance of computing devices improves much faster than that of a modern automobile. What should the lifetime target be for compute? Should we also run every electronic device “into the ground?” If so, which applications should run on these older, less-capable devices? Can we aggregate devices to perform equivalent larger-scale computing? And perhaps most importantly, how do we assess whether it is worth it: when are there carbon savings to be had by scavenging devices from the “junkyard?”

To answer these questions, we introduce a new carbon-aware performance metric. Computational Carbon Intensity (CCI) measures the lifetime carbon impact of a device versus the lifetime useful compute it performs. CCI quantifies the value of extending the service lifetime of computational devices.

We apply CCI to old servers, old laptops, and old smartphones. While each device type shows potential as carbon-saving hardware, we find that used smartphones (repurposed as general-purpose compute nodes) offer the best potential for carbon impact.

Smartphones are an attractive target for repurposing for several reasons. First, there is a remarkably large number of them. Between 60-70% of smartphones are neither thrown out nor recycled [6, 38]. If even 10% of the devices decommissioned in the last five years were available for repurposing, we would have 75 million new compute nodes. Second, a smartphone comes with a wide array of valuable components: increasingly powerful processors, a robust uninterruptible power supply (batteries), and a diverse array of networking hardware, which includes local area connectivity (WiFi and Bluetooth) and long-range, high-performance uplink (cellular modems).
Figure 1: The capabilities of recent smartphones meet or exceed that provided for modern microservices. The three plots show the performance (according to the GeekBench score [21]), number of cores, and memory for the five most popular Android phones released each year since 2013. A GeekBench score of 1 is equivalent to an Intel Core i3 processor. Solid lines indicate the mean. The shading shows the minimum and maximum ranges. The memory plot has two lines corresponding to the minimum- and maximum-memory configurations available. The horizontal dotted lines show the capabilities of different Amazon EC2 T4g instance sizes as of August 2021, plotted for context.

Can a smartphone meet the performance provided by purpose-built cloud servers? We argue that it can. Figure 1 compares the performance, the number of cores, and memory in legacy smartphones to AWS T4g instances, which use Arm-based AWS Graviton2 processors. T4g instances target burstable general-purpose workloads like microservices, e-commerce platforms, small databases, development environments, and virtual desktops [5]. The analysis indicates that even a several-year-old smartphone could be valuable for this class of modern cloud service.

We put this idea to the test. We benchmark discarded smartphones, some of which are now a decade old. We then build out a cloudlet of Pixel 3A smartphones, purchased on eBay for $65 USD. We measure the performance of the cloudlet using DeathStarBench, a benchmarking suite for cloud microservices [19].

The main contributions are:

(1) Defining CCI, a new metric to quantify the lifetime carbon impact of a device and our ability to amortize that impact (Section 3)
(2) Comparing the carbon and performance trade-offs of reusing old servers, laptops, and smartphones to identify the phone as a first-priority platform to explore for reuse (Section 3)
(3) Performing a design space exploration of how to architect legacy phones into a ‘junkyard computer’ (Section 4)
(4) Building a real-world prototype cloudlet constructed from reused devices and benchmarking its performance using modern microservice applications from the DeathStarBench suite (Section 6).

2 RELATED WORK
We are not the first to highlight the computational capability of wimpy cores [3], nor the first to suggest that phone-class hardware could be used [23, 43], or re-used [27, 50], in cloud computing contexts. What differentiates this work is centering the design and evaluation on the amortized lifetime carbon impact and empirical testing on recovered hardware.

2.1 Recycling is not the Answer
E-waste recycling of consumer electronics recovers less than 50% of their materials [52] and has a negative local impact on people and the environment [7, 28]. The inclusion of precious metals in ICs provides an incentive for recycling, but the presence of toxic chemicals means that the recycling process is energy-intensive and hazardous [37]. Wealthy countries in North America and the EU exacerbate global health disparities by shipping large amounts of E-waste to developing nations such as China and India. Just three towns in China are estimated to process 11.5% of the world’s E-waste [9]. Extensive studies of one of these towns revealed dangerous levels of pollutants in the surrounding environment [39] and elevated levels of heavy metal poisoning among residents.

2.2 Computational Efficiency Metrics
While efficiency metrics are useful for many applications, they do not capture the environmental and human cost of compute. Greater energy efficiency does not always imply a lower carbon footprint, especially when efficiency is achieved via fast device turnover.

Power Usage Effectiveness (PUE) is a standard metric for datacenter efficiency that expresses the ratio of the total energy consumption of the datacenter to the energy consumption of ICT devices alone, with 1.0 being an ideal value. While PUE captures the operational efficiency of the datacenter’s computational devices and cooling system, it does not reflect the tremendous energy that went into manufacturing the facility or the equipment housed within.

To understand why this is important, consider two datacenters, A and B. For the same computational output, datacenter B consumes 10% more energy than A, i.e., $\text{PUE}_{(B)} = 1.1 \times \text{PUE}_{(A)}$. However, datacenter A achieves this efficiency by upgrading its servers at a rate that is 2× faster than datacenter B; i.e., datacenter A’s ICT manufacturing costs are 2× that of datacenter B. Let us assume that manufacturing is responsible for 20% of the carbon emissions associated with both datacenters and that operational energy accounts for the other 80%—a reasonable assumption for a server-class machine [10]. Then, the carbon footprint of datacenter B compared to that of datacenter A is:

$$\text{CO}_2e_{(B)} = 0.8 \times (1.1 \times \text{CO}_2e_{(A)}) + 0.2 \times (0.5 \times \text{CO}_2e_{(A)})$$

$$\text{CO}_2e_{(B)} = 0.98\text{CO}_2e_{(A)}$$

Where $\text{CO}_2e$ is the Global Warming Potential (GWP), or carbon footprint, in units of CO$_2$-equivalent weight. Despite having a higher
PUE, datacenter B has a slightly lower carbon footprint because it uses its servers longer.

Total Cost of Ownership (TCO) reflects the total dollar cost of equipment and operations over a period of time. It reflects how well initial purchases costs are amortized, and captures runtime efficiency via operational overhead. However, these economic costs are not always aligned with carbon costs.

2.3 Characterizing the Carbon Footprint of Computation

Several recent publications bring attention to the growing importance of carbon metrics in computing. Patterson et al. argue that researchers should consider the carbon emissions incurred by machine learning training [45]. Their formula for the carbon emissions of training is similar to our definition of compute emissions (\( C_C \), introduced in the next section) with the exception of scope: we calculate operational carbon for the entire lifetime of a system as opposed to a single model training event. In both cases, the energy source is important to the overall carbon footprint. Patterson et al. do not address embodied carbon.

Gupta et al. discuss the importance of accounting for embodied carbon [25]. Like us, they note that a significant fraction of smartphone carbon emissions come from manufacturing. They further note that recent optimizations have focused on maximizing performance rather than considering carbon footprint. Like us, they argue for longer device lifetimes: they state that amortizing the carbon footprint of mobile devices requires continuously operating them for at least three years. In follow-on work, Gupta et al. develop carbon-conscious metrics to enable architectural-level modeling of carbon consumption and describe the carbon benefits of repurposing smartphones as an SSD storage system [24].

Life-cycle assessment (LCA) seeks to characterize the lifetime environmental impact of a product across multiple metrics, including carbon. LCAs have been performed for smartphones, laptops, and servers [15, 17]. Manufacturing accounts for 70 – 90% of the lifetime carbon footprint for phones and laptops. For high-performance computing devices, manufacturing accounts for 20 – 50% of the lifetime carbon cost [15, 17, 31]. LCAs measure the total carbon footprint of a device across its lifetime; our approach (CCI) is different in that we consider the amortized carbon footprint of each unit of computational work. We use vendor-provided LCA numbers to parameterize our CCI calculation.

Raghavan and Ma characterize the embodied energy of the internet [46]. Subsequent work argues for the importance of including embodied energy in discussions of sustainable computing more generally [44]. With this work, we attempt to take a step towards fulfilling this mandate by providing architects and systems engineers the tools to express embodied energy quantitatively.

2.4 Repurposing Consumer Electronics for HPC

Rajovic et al. propose using mobile SoCs for high-performance computing (HPC) [47]. Their analysis finds that mobile SoCs are sufficiently performant for many applications and are more energy efficient than traditional HPC chips. Their follow-on work builds a mobile SoC-based cluster [48]. Their implementation uses new and isolated mobile SoCs and is not focused on reuse.

Shahrad and Wentzlaff propose a server built from decommissioned mobile phones [50]. While the work considers E-waste reduction as motivation, it does not look into quantifying the carbon impact. They present a design proposal but do not include an implementation or empirical evaluation.

Büsching et al. empirically test six Android phones over WiFi to evaluate cluster performance on LINPACK [11]. They do not consider other workloads, architectures, or management costs.

2.5 Repurposing Smartphones as Sensors

An orthogonal line of work is using smartphones in IoT and sensor networks [36, 53]. Mobile phone operating systems are optimized for interactive use, and when human interaction is completely removed, a very long tail of deployment-ending issues crops up. This experience led us to replace the standard mobile phone operating system with a more traditional technology stack for our smartphone-based system.

3 QUANTIFYING CARBON

The first step in repurposing devices is assessing their performance and carbon consumption tradeoffs versus a new system. Can we quantify the carbon benefit of repurposing devices? How much performance are we giving up by not using the latest technology? Which device provides the best tradeoff between performance and environmental impacts?

We study the performance and carbon benefits of building a server using different legacy devices. Our baseline is a PowerEdge R740—a modern server for which Dell has published a full LCA [10]. We aim to replicate the R740 functionality with repurposed devices. We compare four building blocks: a 15-year-old server (HP ProLiant DL380 G6), an 8-year-old laptop (Lenovo Thinkpad X1 Carbon G3), a decade-old smartphone (Nexus 4), and a 3-year-old smartphone (Pixel 3A).

3.1 Workload

We assume that the devices operate at the light-medium operating regime specified in Dell’s PowerEdge R740 LCA [10]. This operating regime has the following load profile:

- 100% load mode: 10% of the time.
- 50% load mode: 35% of the time.
- 10% load mode: 30% of the time.
- Idle mode: 25% of the time.

We further consider what supporting datacenter infrastructure must be added to each configuration to provide the following:

- 1 Gbps networking capacity.
- 30 minutes backup power.
- Sufficient cooling to operate in 25 C ambient temperature.

3.2 Performance

Table 1 shows the performance of the five different devices across four applications from the Geekbench suite.1 The PowerEdge Server has the highest performance, followed by the old server, old laptop, 1We generally eschew performance benchmarks such as SPEC due to their unavailability and inaccessibility for mobile platforms. To date, Geekbench is the most commonly used measure in studies that focus on a broader array of platforms, particularly those which include mobile platforms [25, 56].
Table 1: The GeekBench performance on four benchmarks across the five devices. 'Single' represents the single-core performance, and 'multi' represents the multi-core performance. We consider the latter to be representation of the total computational power of the device. N is the number of devices that would be required to build a system with approximately the same computational power as a single PowerEdge R740 (baseline). This is calculated by dividing the PowerEdge multicore throughput by that of the reused device.

| Device                  | SGEMM (Gflops) | PDF Render (Mpixels/sec) | Dijkstra (MTE/sec) | Memory Copy (GB/sec) |
|-------------------------|----------------|--------------------------|-------------------|----------------------|
|                         | Single Multi N | Single Multi N           | Single Multi N    | Single Multi N       |
| PowerEdge R740 Server   | 2015           | 77.2 2,070 1             | 109.1 3,140 1     | 3.58 80.2 1          |
| HP ProLiant DL380 G6    | 2007           | 14.2 104.2 20            | 74.2 528.4 6      | 2.43 16.9 5          |
| Lenovo Thinkpad X1 Carbon G3 | 2015      | 72.1 123.7 17            | 123.2 225.1 14    | 3.08 7.45 11         |
| Pixel 3A Smartphone     | 2019           | 8.84 39.0 54             | 38.9 147.0 22     | 1.08 4.44 19         |
| Nexus 4 Smartphone      | 2012           | 1.95 8.12 256            | 14.1 40.8 77      | 0.654 2.21 37        |

3.3 Energy Consumption

Device power draw plays a prominent role in the overall carbon footprint. We perform a CPU stress test on all devices to characterize their power draw in different regimes. We turn off WiFi and Bluetooth on the phone and laptop for the stress test. We turn off the phone screens and turn the laptop’s brightness as low as possible. We run the Linux command n times, where n is the number of cores the device has and use cpulimit to restrict the CPU usage of each process. The phone’s power draw is measured via a USB power meter, which sits between the device and the power source. The laptop measurements use a plug-through power meter. We query the HPE iLO interface for the ProLiant servers.

Table 2 presents a summary of the results. The servers have the highest power draw, followed by the laptop and smartphones.

Table 2: Power (Watts) given CPU usage. Subscripts indicate CPU usage percentage. $P_{avg}$ calculated for light-medium workload.

| Device      | $P_{100}$ | $P_{50}$ | $P_{10}$ | $P_{idle}$ | $P_{avg}$ |
|-------------|-----------|----------|----------|------------|-----------|
| PowerEdge   | 510       | 369      | 261      | 201        | 308.7     |
| ProLiant    | 280       | 213      | 181      | 169        | 199.1     |
| Thinkpad    | 24        | 16.2     | 8.5      | 3.4        | 11.47     |
| Pixel 3A    | 2.5       | 1.9      | 1.4      | 0.8        | 1.54      |
| Nexus 4     | 3.6       | 2.7      | 1        | 0.7        | 1.78      |

3.4 Carbon Cost

To quantify the environmental cost of proposed solutions beyond operational energy, we need new metrics that properly capture the carbon footprint over the system’s lifetime. An ideal carbon metric:

1. Rewards operational energy efficiency.
2. Rewards manufacturing efficiency.
3. Rewards the reuse of already-manufactured devices.
4. Reflects computational work achieved per unit carbon.

Existing efficiency metrics, e.g., PUE, fulfill point #1 but fail to capture points #2-4. Carbon footprint—a measure of the greenhouse gas emissions of a device over a standard lifetime, reported by companies like Apple [4, 5] and Dell [10, 15]—does not satisfy point #4. Thus, we must define a metric that captures all four points.

**Computational Carbon Intensity (CCI)** is the CO$_2$-equivalent released per unit of computation work. We calculate this metric across the entire lifespan of our devices to get their amortized carbon intensity. The general formula is as follows:

$$CCI = \frac{\sum_{\text{ops}}^\text{lifetime} \text{CO}_2\text{e}}{\sum_{\text{ops}}^\text{lifetime}}$$ (1)

The numerator can be further broken up into the carbon associated with manufacturing, compute, and networking:

$$CCI = \frac{C_M + C_C + C_N}{\sum_{\text{ops}}^\text{lifetime}}$$ (2)

$C_M$ is the carbon associated with the manufacture of the device (often called the "embodied carbon"). It is a single upfront cost at the beginning of the cluster’s lifetime. We source our values for $C_M$ from the life cycle assessment literature.

$C_C$ is the carbon associated with compute. Equivalently, it is the total carbon footprint of the energy required to perform the computations over the lifetime of the device, calculated as:

$$C_C = \sum_{\text{lifetime}} C_{\text{grid}} \times E$$ (3)

where $C_{\text{grid}}$ is the carbon intensity of the grid, in units of CO$_2$e/joule, and $E$ is the cluster’s energy consumption, in joules. At the light-medium workload, this works out to:

$$C_C = C_{\text{grid}} \times \sum_{\text{lifetime}} 0.10 \times P_{100}\% + 0.35 \times P_{50}\%$$

$$+ 0.30 \times P_{10}\% + 0.25 \times P_{idle}$$ (4)

$C_N$ is the carbon associated with networking:

$$C_N = \sum_{\text{lifetime}} C_{\text{grid}} \times f_{\text{net}} \times E_{\text{net}}$$ (5)

where $f_{\text{net}}$ is the rate at which data is sent and received, in bytes/second, and $E_{\text{net}}$ is the energy intensity of networking, in joule/byte. CCI sums these three carbon components and divides by the total number of operations computed in the device’s lifetime.
CCI (mgCO\textsubscript{2}/ flop) 

CCI (mgCO\textsubscript{2}/MTE) 

CCI in practice. With this definition, we can do an initial, per-device CCI calculation for the servers, laptop, and phones.

CCI satisfies point #1 as it accounts for operational energy consumption per instruction in the numerator (CCI\textsubscript{C}). It satisfies point #2 via the inclusion of C\textsubscript{M}. The total carbon cost is amortized per instruction; thus, point #4 is satisfied. To satisfy point #3, we stipulate that when reusing a device, the carbon cost of manufacturing is considered already paid, i.e., C\textsubscript{M} = 0.

Figure 2 shows the CCI over the lifetime of the five devices. Lower CCI values are better. Only the PowerEdge server incurs manufacturing carbon cost C\textsubscript{M}; repurposed devices have C\textsubscript{M} = 0. Generally speaking, the phones have the best (lowest) CCI except under consideration by the CPU usage to get operations per second, then multiply by the lifetime to get the total operations computed.

We assume that throughput scales linearly with CPU usage, e.g., ops\textsubscript{50\%} = 0.5 * ops\textsubscript{100\%}. This simplification is necessary when extrapolating from microbenchmarks; when benchmarking a full system (Section 6), we no longer make this assumption. Under the light-medium workload described previously, the average operations per second (ops) is calculated as:

\[
\text{ops}_{\text{avg}} = 0.10 \times \text{ops}_{\text{100\%}} + 0.35 \times \text{ops}_{\text{50\%}} + 0.30 \times \text{ops}_{\text{10\%}} \tag{6}
\]

Figure 2 shows the CCI over the lifetime of the five devices. Lower CCI values are better. Only the PowerEdge server incurs manufacturing carbon cost C\textsubscript{M}; repurposed devices have C\textsubscript{M} = 0. Generally speaking, the phones have the best (lowest) CCI except under the SGEMM benchmark. Here, the specialized computational hardware available on the laptop outweighs its higher power draw.

Assumptions and limitations. It is difficult to account for every possible impact of a computing system. For instance, CCI, as it stands, does not account for the carbon footprint of human labor and transportation beyond initial manufacture.

For our analysis, we assume that any device being repurposed is essentially ‘free’ in terms of manufacturing carbon, i.e., C\textsubscript{M} = 0.\footnote{Although as we will see below, when building more complex systems the carbon cost of added peripherals must be considered.} An alternate analysis considers the initial manufacturing cost of the repurposed device, amortized by the work done in its first life. This alternate CCI formula is:

\[
\text{CCI} = \frac{C_M + C_{CI(1^{st})} + C_{N(1^{st})} + C_{CI(2^{nd})} + C_{N(2^{nd})}}{\sum \text{ops}(1^{st}) + \sum \text{ops}(2^{nd})} \tag{7}
\]

Where C\textsubscript{CI(1\textsuperscript{st})} is the operational carbon footprint of the first life, C\textsubscript{CI(2\textsuperscript{nd})} is the operational carbon footprint of the second, etc. While this alternate form might be helpful for certain analyses, we find that it is difficult to reason about the operational carbon footprint and operations computed in a device’s ‘first life.’ In our case, we are working with used devices from eBay and have no visibility into their first lives.

Another limitation of CCI is that it focuses on computing and does not capture other types of reuse, e.g., it does not account for whether or not the camera, microphone, and networking capabilities are being reused. Consumer devices contain many subcomponents supporting different compute requirements: CPUs, GPUs, custom hardware accelerators (audio, encryption, AI), networking hardware, power supplies, and batteries. Ideally, every component is reused, but in reality, not all are required. For example, devices in a server role may not need a display or audio components.

Reuse Factor is a metric to account for the usage of the subcomponents. It weights each component by its carbon footprint to estimate the embodied carbon of the components of the device that are being reused. It is calculated as:

\[
\text{RF} = \frac{\sum \text{reused } C_{M(i)}}{C_M} \tag{8}
\]

Table 3 gives our best attempt at capturing the component-wise carbon footprint of a smartphone. Coarse-grained ratios are sourced from [17], which reports a 77\% contribution from ICs, 10\% from the
Table 3: Working estimates for the fractional carbon intensity of various Nexus 4 subcomponents.

| Category | Contribution | Includes |
|----------|--------------|----------|
| Compute  | 25%          | 12.5 kgCO₂ |
|          |              | 2GB RAM (Samsung K3PE0E00A), Snapdragon S4 Pro |
|          |              | LG and WiFi chips: Qualcomm MSM8925M, Murata SS988001, Qualcomm WTR1605L, Broadcom 20793S |
| Network  | 15%          | 7.5 kgCO₂ |
|          |              | Screen |
| Battery  | 15%          | 7.5 kgCO₂ |
|          |              | Battery, power management (Qualcomm PM8921, Avago ACPM-7251) |
| Display  | 10%          | 5 kgCO₂ |
|          |              | Sensors |
| Storage  | 10%          | 4 kgCO₂ |
|          |              | 8GB flash |
| Sensors  | 5%           | 3 kgCO₂ |
|          |              | Accelerometer, camera, audio codec |
| Other    | 20%          | 10 kgCO₂ |
|          |              | PCB, Chassis, packaging, other ICs |

display, 3% from the battery, and the rest from other components, including the PCB. The contribution from ICs was then further broken down using images from a Nexus 4 teardown and assuming that the relative contribution of each chip scales according to size.

Better component-wise embodied carbon estimates are necessary to create more accurate models. The ACT is a tool for modeling architectural carbon footprint [24]. Currently, ACT only has estimates for ICs, which align with our estimates here, but does not consider all elements of the phone (e.g., battery, display, PCB, and chassis, which contribute \(\frac{1}{3}\)). Thus, ACT is complementary to our work, and we are excited by the potential of its component models to improve our more holistic models.

The reuse factor should be treated as a proxy for the extent to which a device has been reused. To see how the reuse factor is used, consider a cloudlet example: A smartphone is repurposed as a compute node. It is network-connected and is always in use. Code and results may use storage on the device and its battery as a UPS. In this scenario, the compute capabilities, networking, battery, and storage are all reused, while the display and sensors are not. This yields a reuse factor of 0.85.

4 DESIGNING THE JUNKYARD

We showed that repurposed smartphones have the potential to outperform a traditional server in terms of carbon per op. This section discusses how to transform discarded consumer smartphones into general-purpose compute nodes. We aim to answer the question: What does it take to make a server out of smartphones?

4.1 Cooling

Datacenter cooling can account for as much as 40% of operational energy consumption [26]. While we envision cloudlets rather than warehouses as a more realistic outcome for repurposed phones, we still anticipate packing heat-generating compute into limited space.

Phones are designed not to burn your hand when you use them, which means they have a strict upper limit on their thermal dissipation. They begin throttling around 40-50°C [32]. Thus, a well-behaved device should never overheat due to its operation. At around 60-70°C, the device will shut itself off to protect the user [35].

While collecting many smartphones in an enclosed space can create a high ambient temperature, we postulate that the individual thermal throttling of each device has the aggregate effect of limiting the air temperature reached.

Experimental Set-Up. Four Nexus 4s and one Nexus 5 are placed in a sealed 3 × 15 × 10.5 inch Syrofoam box. We run two scenarios: a CPU stress test, with a 90%+ CPU job running on all four cores, and a simulated light-medium workload. We log job throughput, device internal temperature, and air temperature in the box.

Results. Figure 3 summarizes performance and temperature results. We also calculate thermal power for both scenarios:

\[
P_T = \frac{c_p(\text{air}) \cdot m_{\text{air}} \cdot \Delta T_{\text{air}} \sum_{\text{phones}} c_p(\text{Si}) \cdot m_{\text{ph}} \cdot \Delta T_{\text{ph}}}{\Delta \text{time}}
\]

We calculate the thermal power prior to the shutdown of any devices. Total thermal power for the 100% load scenario is \(-13\) W (2.6 W/device), and \(-6\) W (1.2 W/device) for the light-medium scenario.

Some additional observations: a) The phones shut off at an internal temperature of 75-80°C. For most devices, this happened at an air temperature of 40°C. b) The Nexus 5 did not overheat in either scenario. c) As the temperature increased, performance decreased. d) Even under a 90%+ (CPU) load, the phones exhibited a thermal power (2.6 W) well below their 5 W thermal design point (TDP).

At the cloudlet scale, with 256 Nexus 4s running at 100%, we expect up to 666 W of thermal power. This is within the cooling abilities of two COTS server fans rated for 500 W, which would add 4 W of power per fan [16]. A rough estimate based on weight [40], and assuming a world energy mix during production, yields an embodied carbon of 9.3 kg per fan. We consider this added embodied carbon when estimating the carbon efficiency of cloudlet-scale smartphone clusters in Section 5.2.

Scaling Further. Aggressive thermal management reduces the cooling demand of a hypothetical smartphone datacenter but also limits its performance potential. A phone’s modest thermal dissipation capacity means that its effective sustained TDP is much lower than its transient peak TDP. Thus, phones are well-suited to bursty
workloads, but a poorer fit for more sustained compute without an aggressive adaptation of their thermal management systems.

4.2 Networking
There are many ways to network the phone cluster. We consider two general use cases: (1) the in-situ, edge compute case, where the cluster operates independently with access only to the cellular network and (2) the existing-infrastructure case, where the cluster has access to a pre-existing WiFi or wired network. The first case reflects applications in remote environments. The second might occur if the cluster was housed within an office building or datacenter.

Na et al. found that networking a co-located smartphone cluster over WiFi is not feasible; interference made wireless networking intractable beyond 30 devices [41]. Thus, in a datacenter-scale scenario, we expect that wired clusters of smartphones connected to network switches would be employed.

For smaller clusters operating away from external infrastructure, we propose a tree topology, with phones organized into sets of five, each with one hotspotted device. The hotspotted device communicates with the outside world via LTE and other devices via its WiFi network. WiFi is the more bandwidth limiting, e.g., the Nexus 4/5 smartphones have 150 Mbit/s for both the uplink and downlink [12]. The tree topology leads to a capacity of 18.5 Mbit/s downlink and uplink per device. While wired networking is best as the cluster scales, for smaller clusters (i.e., our ten-device cloudlet described in Section 6), we have found a wireless network to be sufficient.

4.3 Batteries & Smart Charging
A critical advantage of smartphones is their built-in batteries. By charging during periods of higher renewable (“greener”) energy production, we can reduce the operational carbon intensity of the cluster. We call this smart charging.

Smart charging opportunistically charges the devices whenever the grid-level carbon intensity falls below a tunable threshold. This threshold is based on historic grid characteristics and the charged device. We devised a heuristic for the California grid, but other regimes might require a different strategy. The device is charged if the battery level drops below 25%, regardless of grid conditions.

We evaluate this algorithm on publicly available data from the California Independent System Operator [29]. In California, the carbon intensity of the grid tends to be anti-correlated with solar production trends (Figure 4a). Since this pattern is relatively uniform across days, we set the threshold as the Pth percentile of the previous day’s instantaneous carbon intensities, where P is the percent of time the device will need to spend charging.

In California, the smart charging algorithm schedules charging when solar production is highest (middle of the day). The carbon savings vary depending on the device’s battery capacity and charging rate. The Pixel 3A, with a 3 Ah battery and 18 W charging rate, sees a median carbon reduction of 7.2% for the time period studied, while the ThinkPad X1 Carbon Gen3 laptop sees a median 4.0% reduction during the same time period. The laptop sees smaller savings than the Nexus phone because its much higher power consumption (11.4 W versus 1.54 W) offsets its larger battery capacity.

Batteries also provide a convenient source of backup power. For a Pixel 3A operating on a light-medium workload, a 25% charge will last just under 2 hours. The 25% minimum can be increased to provide a higher margin. Conversely, the minimum charge threshold can be decreased to prioritize carbon savings over backup power.

There is one complicating factor, however: Battery lifetimes. Smartphone batteries become unusable after about 2,500 cycles [18]. Consider a Pixel 3A with a light-medium usage pattern. The mean power draw of the device would be 1.54 W, giving a daily energy consumption of 133 kJ. The 3 Ah (45 kJ) battery included in the Pixel would require three full daily charges. After 833 days or 2.3 years, the battery would be unusable and have to be replaced.

The Pixel 3A’s battery has an embodied carbon of 2.00 kgCO₂e and a projected lifespan of 2.3 years. The Nexus 4’s 2.1 Ah battery and 1.8 W average power yields an embodied carbon of 1.11 kgCO₂e and a projected lifespan of 1.23 years. For any reused smartphone, the embodied carbon cost of replacing the battery C_M is:

\[ C_M = C_M(BATTERY) \times \frac{\text{lifetime}}{\text{battery lifetime}} \] (10)

This replacement is also costly in terms of human labor. We have replaced a Nexus 4 battery and found it takes about 10 minutes. For a server-equivalent cluster of 54 (Pixel 3As) or 270 (Nexus 4) phones, this would imply 9 (Pixel 3A cluster) - 45 (Nexus 4 cluster) hours of labor every 1.2-2.3 years. We believe this is reasonable; however, this upkeep may become prohibitive for larger systems. Removing and bypassing the battery completely might be worthwhile despite the loss of smart charging and using the battery as a UPS.

5 CHARACTERIZING CARBON
We consider the design factors from the previous section to evaluate the carbon savings of junkyard smartphone servers.

5.1 Sourcing Energy
First, we consider the energy source’s effect on the carbon intensity of new and reused devices. The carbon emissions associated with compute, C_C, are equivalent to the carbon footprint of the wallplug energy used to power the device. This can be expressed as:

\[ C_C = C_{\text{grid}} \times \sum \text{lifetime} \frac{P_{avg}}{\text{battery lifetime}} \] (11)

The carbon intensity of the grid (C_{grid}) is a measure of the amount of CO₂e released per kWh of energy provided and varies depending on the source of that energy. For instance, the carbon intensity of solar is 48 gCO₂e/kWh and is 602 gCO₂e/kWh for gas. In California, the mean carbon intensity of grid power is 257 gCO₂e/kWh. Figure 6 examines how variations in local energy supply affect CCI. Unsurprisingly, more renewable energy means a lower carbon footprint. We can get additional gains with smart-charging.

Figure 6 considers three power regimes. The first is an energy mix that models the California grid. The second is a theoretical regime in which solar energy is always available. While this is not realistic today, hyperscalers have announced plans for 24/7 carbon-free energy within the next decade [22].

The third is a theoretical, 100% carbon-free energy source. Note that such an energy source does not currently exist. Even solar, wind, and hydro have carbon costs associated with their generation, which is reflected in their carbon intensity. However, it provides a
Figure 4: Smart charging saves carbon. Results are shown for a representative day during the month of April 2021. The black curve indicates the carbon intensity of the California grid throughout the day, and the green shading gives the charging periods suggested by the smart charging algorithm. On this particular day, the Pixel 3A’s carbon emissions are reduced by 7.98%, and the ThinkPad’s carbon emissions are reduced by 4.28%. Over the course of the month as a whole, the Pixel 3A sees median carbon savings of 7.22%, with a standard deviation of 5.93%. The ThinkPad sees median carbon savings of 4.03%, with a standard deviation of 2.2%.

Figure 5: Cluster-level CCI for the smartphone clusters vs. an equivalent number of new PowerEdge servers, old Proliant servers, and old ThinkPad laptops. The six graphs have three benchmarks with two power regimes. The first row shows CCI for a California energy mix with smart charging for the laptops and smartphones. Smart charging requires a periodic replacement of the battery. The second row assumes access to solar power 100% of the time, obviating the need for charging and eliminating the batteries. In this case, $C_C$ is zero, and as Figure 6 shows (green lines), $C_M$ becomes the dominant factor for CCI.

5.2 Cloudlet-scale Computing
We now compare the carbon intensities of each previously introduced comparison point at the cloudlet scale. That is, we consider the number of devices and the peripherals necessary to make a cluster of smartphones, laptops, or old servers that meet the performance of a PowerEdge R740 server. We also consider the effect of networking (reflected by the $C_N$ term). Our cloudlets are:

1. A single PowerEdge R740.
2. 17 Lenovo ThinkPad Gen 3s, with 17 smartplugs added to achieve smart charging savings of 4%.
3. 20 ProLiant.
4. 54 Pixel 3A phones, with 20% designated as networking and management nodes. 54 smartplugs are added to achieve smart charging savings of 7%, and one server fan is rated for a 500 W TDP.
5. 256 Nexus 4 phones, with 20% designated as networking and management nodes. 256 smartplugs are added to achieve smart-charging savings of 7% and two server fans rated for a 500 W TDP.
We assume that items #1-3 are plugged into an already-existent wired network, while items #4-5 achieve network connectivity via the tree topology described in Section 4.2.

**Manufacturing (C_M).** The inclusion of hardware peripherals adds a new term to C_M: C_M(p), which represents the carbon cost of the added peripherals. The cloudlet-scale C_M for the repurposed smartphone- and laptop-based systems is:

\[
C_M = N \times C_{M(batt)} + \frac{\text{lifetime}}{\text{battery lifetime}} + C_M(p)
\]

(12) where N is the number of devices.

**Networking (C_N).** We calculate the carbon footprint of networking as described in Equation (5):

\[
C_N = \sum_{\text{lifetime}} C_{\text{grid}} \times f_{\text{net}} \times E_{\text{net}}
\]

For the smartphones, E_{\text{net}} varies for 3G, 4G, and WiFi. From [1], we use 5 uJ/byte for WiFi, and 11 uJ/byte for LTE. The actual value of f_{\text{net}} will vary widely depending on the target workload. We assume 0.1 Gbps for each cloudlet.

**Compute (C_C).** At cloudlet scale, we add the operational carbon cost of any added peripherals. For the smartphone-based cloudlets, this will be the added power consumption of the server fans.

\[
C_C = N \times C_{C(device)} + C_{C(p)}
\]

(13) The results of this cloudlet-scale calculation are presented in Figure 5. We plot the results across three benchmarks: SGEMM, PDF Rendering, and Dijkstra, and for two power regimes, California mix and 100% solar. In the 100% solar regime, we remove the smartplugs from the equation since they are not needed for smart charging.

In all cases, the repurposed smartphones and laptops perform better than the new server, especially for shorter lifetimes. The old server performs the worst overall due to its relatively high energy consumption (2/3 that of the PowerEdge despite being significantly less powerful). For the SGEMM benchmark, the ThinkPad performs the best overall; for the other two benchmarks, the Pixel performs the best. The carbon savings become even more pronounced in the 100% solar regime, where embodied carbon dominates.

It is interesting to note that the Nexus 4 smartphone cluster, despite consuming more energy (456 W) than the new PowerEdge (309 W), is nonetheless still more carbon efficient for both the PDF Rendering and Dijkstra benchmarks. For the SGEMM benchmark, the Nexus 4 cluster is more carbon efficient for lifetimes less than 45 months. In other words, running the higher-powered Nexus 4 phone cluster is better than manufacturing a new server if that server will be in service for less than 45 months. The more efficient Pixel 3A smartphone cluster beats out the server every time.

## 5.3 Datacenter-Scale Analysis

Like PUE, CCI can be calculated at the datacenter scale and provides new insights not reflected by PUE alone.

To illustrate this, we estimate CCI and PUE for two 50MW datacenters: One built with clusters of Pixel 3A smartphones and one with PowerEdge R740 servers. With the PowerEdge at 308 W and the Pixel 3A cluster at 84 W (54 Pixel 3A phones on a light-medium workload), we provision each 50 MW datacenter with 170,000 units. We assume each smartphone cluster will take up 2U of space, which is enough to leave 75% of the space empty.

### 5.3.1 PUE

We make the simplifying assumptions of no personnel or windows and define PUE as:

\[
PUE = \frac{P_{IT} + P_{cooling} + P_{lighting}}{P_{IT}}
\]

(14) We follow the methodology in [42] to get estimates for P_{cooling} and P_{lighting} using the wattage and size of the PowerEdge and phone cluster. This gives a PUE of 1.32 for the smartphone-based design and 1.31 for the traditional server-based system. The smartphone-based design has a slightly larger PUE because it takes up more physical space, which means that P_{cooling} and P_{lighting} are higher.

### 5.3.2 CCI

For datacenter scale:

\[
CCI = \frac{C_M + PUE \times (C_C + C_N)}{\sum_{\text{lifetime}} \text{flop}}
\]

(15) Using the PUE results from above, and assuming a California energy mix and three-year lifespan, we get the macro-level CCI results given in Table 4.

## 6 JUNKYARD CLOUDLET

To demonstrate the feasibility of smartphone-based cloudlet computing we build out a proof-of-concept cluster of ten Pixel 3A smartphones and evaluate our junkyard prototype against a range of similarly-powerful AWS EC2 instances using the DeathStarBench benchmarking suite [19], which provides complete end-to-end microservice-based cloud applications.
6.1 Experimental Methodology

The testbed consists of ten Google Pixel 3A and Pixel 3A XL phones, initially released in 2019. We replace the native Android OS with Ubuntu Touch, an open-source mobile OS that provides a desktop-like experience [51]. To support Docker, which DeathStarBench relies upon, we further modify the Ubuntu Touch kernel to add several necessary modules, e.g., the BTRFS file system [34].

**Benchmarks.** We implemented two of the three publicly available end-to-end applications from the DeathStarBench suite. They are described below based on information from the DeathStarBench paper and public GitHub [14, 19].

- **HotelReservation** is a service that supports getting location-based hotel information and rates, recommending hotels based on user metrics, and placing reservations. It is built with Go and gRPC. The benchmark includes a mixed workload generator.

- **SocialNetwork** implements unidirectional follow relationships and supports creating text and media posts, reading posts, reading an entire user timeline, searching a database for users or posts, registering, logging in, and following. It is implemented in C++ and uses Thrift RPCs for communication. There are three workload generators: composing posts, reading user timelines, and reading home timelines.

We present results for the first two workload generators.

We attempted to deploy the third public DeathStarBench application (MovieReviewing) on the smartphone cloudlet but found that it did not scale well to multiple devices. The median and tail latency increased with an increasing number of devices. This has been observed by others [30], so we do not believe that it is a limitation of the smartphone platform but rather a property of the benchmark.

**AWS Experiments.** As a baseline, we test each benchmark on differently-sized AWS EC2 C5 instances, described as providing “cost-effective high performance” for “advanced compute-intensive workloads” [49]. The applications run on a single AWS machine. The client workload generator operates in another thread on the same machine to eliminate network latency between client and server. Since all microservices are deployed on the same node, there is also no network latency between microservice function calls.

**Smartphone Testbed.** The smartphones are connected over WiFi. They run in Docker Swarm mode, which distributes microservices amongst the phones according to the dependencies defined in the docker-compose-swarm.yml file from the DeathStarBench GitHub [14]. To minimize network latency, all phones and the client are on the same local WiFi. Device communication has added latency since the microservices are spread amongst multiple devices.

6.2 DeathStarBench Performance

Figure 7 shows the performance of the phone cloudlet against various AWS EC2 instances. The smartphone cloudlet has a higher tail and median latency across all throughputs; however, it scales well to a large number of requests. This higher latency is expected, given the added network latency between devices in the swarm.

The latency requirements of the system would determine whether the smartphone cloudlet is an acceptable replacement for a cloud server. For instance, if the hotelReservation system required that median latency be no more than 50 ms and tail latency no more than 100 ms, the smartphone cloudlet could meet the specs and handle up to 4,000 queries per second. In this case, the performance of the cloudlet would approximate that of a c5.9xlarge EC2 instance, which costs $1.53 per hour.

The smartphone cloudlet costs $1,027.60 USD for three years of deployment, which includes the initial $70 USD cost per Pixel 3A and California energy prices. The c5.9xlarge EC2 instance costs $40,404 USD ($1.53 USD per hour) for the same deployment length.

Some interesting differences exist between the smartphone cloudlet performance on the different benchmarks. Normalized to the AWS equivalents, the smartphone cloudlet performs significantly better on the write-only SocialNetwork application than on the read-only application. This may be explained by the fact that the read workload involves the transfer of a user’s entire timeline – a large amount of data.
CPU Utilization. The CPU utilization of the smartphones (Figure 8) indicates two interesting points. First, the smartphones were generally not CPU-bound while hosting these microservices. Second, the utilization varied widely depending on the services being hosted by the device, with 6/10 of the devices making little use of their raw compute powers.

We ran the same experiment on the c5.9xlarge and found that it was similarly not CPU-bound: the CPU utilization was relatively constant at 30% utilization while running the SocialNetwork-Read workload at 4,500 QPS and 25% while running the SocialNetwork-Write workload at 2,000 QPS.

6.3 Carbon Performance

A Pixel 3A consumed approximately 1.7 W while hosting the hotel reservation system. Therefore, we assume that the ten-device system running at 17 W requires, at most, a single server fan. With this energy consumption, the phones would have to be charged 3.7 times a day, and batteries would have to be replaced every 1.9 years. Using the CCI methodology, Figure 9 shows the carbon intensity of the Pixel 3A cloudlet.

It is impossible to know the actual power consumption of the EC2 instances without having access to their custom hardware. We use the public dataset of rough estimates based on benchmarking very similar hardware [13] to approximate that a c5.9xlarge instance will use 140.7 W at 10% utilization, and 239 W at 50%. The CPU benchmarking indicates that the c5.9xlarge will likely be at 25-30% utilization while hosting SocialNetwork; for simplicity, we round down to 10% and use the 140.7 W as an estimate for c5.9xlarge. The

Figure 9: Carbon intensity of Pixel 3A cloudlet running the DeathStarBench applications compared to c5.9xlarge server.

The major takeaways of this work are:

1. For specific workloads, clusters of repurposed phones are cheaper and more carbon efficient than traditional servers.
2. More broadly, scavenging unwanted equipment shows excellent potential for building economic and carbon-efficient systems, especially when renewable energy is plentiful.
3. Sustainability has operational and manufacturing facets; manufacturing dominates as operating trends towards zero with cleaner energy mixes.
4. Accurate LCA information is essential for carbon-based analyses; it would be beneficial if more ICT manufacturers published this information, including cloud providers who build custom systems.

Our work highlights the need for more holistic analyses of the environmental impact of computing. With the substantial carbon cost of manufacturing and the difficulties of responsible recycling, the energy efficiency of a device may be the least significant component of its environmental and human impact.

8 ACKNOWLEDGEMENTS

This work was funded in part by a National Science Foundation (NSF) grant (CNS-2233894). We thank the anonymous reviewers and our shepherd for their aid in improving the paper. We would also like to thank Dean Tullsen for his feedback and encouragement.
Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon

[47] Nikola Rajovic, Paul M Carpenter, Isaac Gelado, Nikola Puzovic, Alex Ramirez, and Mateo Valero. 2013. Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC?. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 1–12.

[48] Nikola Rajovic, Alejandro Rico, Nikola Puzovic, Chris Adeniyi-Jones, and Alex Ramirez. 2014. Tibidabo: Making the case for an ARM-based HPC system. Future Generation Computer Systems 36 (2014), 322–334.

[49] Amazon Web Services. [n. d.]. Amazon EC2 C5 Instances. AmazonEC2C5Instances. Accessed: 2022-10-30.

[50] Mohammad Shahrad and David Wentzlaff. 2017. Towards deploying decommissioned mobile devices as cheap energy-efficient compute nodes. In 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17).

[51] ubports. 2021. Ubuntu Touch. https://ubuntu-touch.io/. Accessed: 2022-04-12.

[52] Emile Van Eygen, Steven De Meester, Ha Phuong Tran, and Jo Dewulf. 2016. Resource savings by urban mining: The case of desktop and laptop computers in Belgium. Resources, Conservation and Recycling 107 (2016), 53–64. https://doi.org/10.1016/j.resconrec.2015.10.032

[53] Trevor Zink, Frank Maker, Roland Geyer, Rajeevan Amirtharajah, and Venkatesh Akella. 2014. Comparative life cycle assessment of smartphone reuse: repurposing vs. refurbishment. The International Journal of Life Cycle Assessment 19, 5 (2014), 1099–1109.

Received 2022-07-07; accepted 2022-09-22