A Case for Micro-Cellstores: Energy-Efficient Data Management on Recycled Smartphones*

Stavros Harizopoulos
HP Labs
Palo Alto, CA, USA
stavros@hp.com

Spiros Papadimitriou
Google Research
Mountain View, CA, USA
spapadim@gmail.com

ABSTRACT

Increased energy costs and concerns for sustainability make the following question more relevant than ever: can we turn old or unused computing equipment into cost- and energy-efficient modules that can be readily repurposed? We believe the answer is yes, and our proposal is to turn unused smartphones into micro-data center composable modules. In this paper, we introduce the concept of a Micro-Cellstore (MCS), a stand-alone data-appliance housing dozens of recycled smartphones. Through detailed power and performance measurements on a Linux-based current-generation smartphone, we assess the potential of MCSs as a data management platform. In this paper we focus on scan-based partitionable workloads. We show that smartphones are overall more energy efficient than recently proposed low-power alternatives, based on an initial evaluation over a wide range of single-node database scan workloads, and that the gains become more significant when operating on narrow tuples (i.e., column-stores, or compressed row-stores). Our initial results are very encouraging, showing efficiency gains of up to 6×, and indicate several promising future directions.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

Modern smartphones have the computational power of a 5-year-old PC, but at a fraction of the size and energy consumption (110x smaller volume than a standard 1U server, and 200x less peak power). More than 1 billion cellphones are shipped yearly; in 2010, according to IDC, over 300 million of those were smartphones (a 74.4% increase over 2009). Smartphones have a typical consumer refresh cycle of two to three years. Over the next few years, we expect a total of one billion smartphones to become obsolete; the aggregate computational power of these phones is similar to that of *all* 500 top supercomputers in the world combined—but at a fraction of their energy needs. How can this power be harnessed?

A place where cost- and energy-efficient computing units could be utilized at large numbers is a modern data center. Data center operating costs are characterized by a continuously growing energy cost component [5, 2]. Power and cooling costs are soon expected to surpass the (amortized) cost of purchasing servers. Demand for new and bigger data centers is on the rise, fueled by both consumer and enterprise applications. However, could a significantly underpowered device support applications that typically run on high-end servers? In this paper, we argue that for certain classes of enterprise data management problems, such as data warehousing and analytics, there are several emerging trends that lend themselves to a micro-data center design based on underpowered hardware (also known as "wimpy nodes" in the literature [3]). These trends are (a) MPP-style processing (massively parallel processing), (b) column-oriented and compressed data which ease pressure on the memory/network buses, and (c) offering reliability through replication instead of expensive hardware solutions.

Our proposal is to repurpose old or unused smartphones and use them to assemble units, called Micro-Cellstores, that contain dozens of interconnected smartphones which collectively act as a data-appliance mini-cluster. A concept dia-

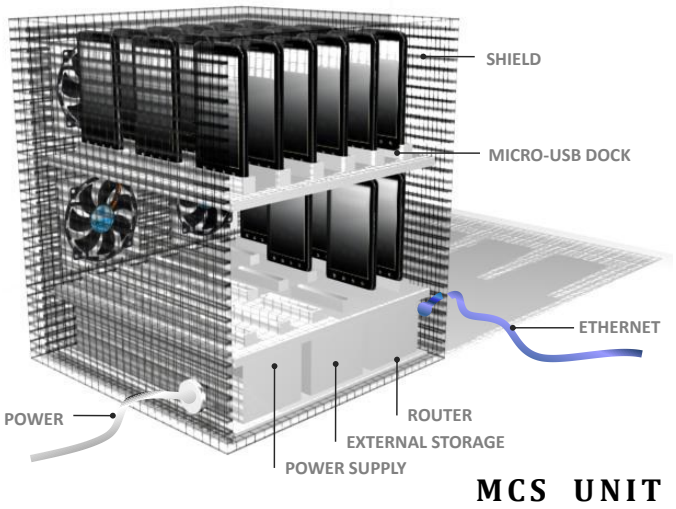

Figure 1: Micro-Cellstore Architecture.

*The views contained herein are the authors' only and do not necessarily reflect the views of Hewlett-Packard or Google.

DaMoN 2011

gram of a Micro-Cellstore is shown in Figure 1. There are several interesting questions around architecting and manufacturing Micro-Cellstores that are beyond the scope of this paper, such as: What are viable methods for networking? How can batteries and power management features be leveraged? What is the right ratio of phones, routers/hubs, and external storage? Is cooling a problem?

Our focus in this paper is exploring the types of data management workloads that can be efficiently supported by MCS units, and compare the energy-efficiency of appliances based on smartphones (ultra-wimpy nodes) against other low-power alternatives. Our contributions are the following:

- Detailed power-profile characterization of a modern smartphone (Nexus S, released in Q4 2010).
- Power efficiency measurements for partitionable, scan-intensive database workloads on smartphones and two types of wimpy platforms.
- Introducing the case for Micro-Cellstores (MCS).

The rest of the paper is organized as follows. In Section 2 we cover related work, including recent proposals for "wimpy" architectures. Section 3 motivates Micro-Cellstores and Section 4 details the characteristics of modern smartphones. Section 5 carries out our benchmarking and analysis of various single-node, scan-based database workloads. We conclude in Section 6.

2. RELATED WORK

Energy concerns are important enough to often dictate where data centers are built. A growing number of efforts to improve the energy efficiency of clusters and data centers include holistic redesigns that treat a data center as a single computer [4, 16], cluster workload consolidation to meet power constraints and reduce energy requirements [15, 13, 12], and considerations of low-power architectures [3, 19]. In this section we briefly discuss recent efforts in improving the energy efficiency of database applications.

Energy efficiency in databases. Traditionally, database systems have been optimized for performance, ignoring power-related costs. However, the proliferation of scale-out architectures has forced data management systems to consider energy as equally important to performance. Early research studies argued for the redesign of several key components such as the query optimizer, the workload manager, the scheduler and the physical database design [7, 11, 9, 20]. Many of these suggestions assumed that, like cars, computer systems had different optimal performance and energy efficiency points. However, a subsequent detailed study on the energy efficiency of a single database server [18] found that, because of the start-up power draw, the highest performing configuration was also the most energy efficient. That study did not consider multi-node configurations or low-power hardware. In this paper, we investigate the latter.

Non-server architectures. In an effort to improve the energy efficiency of clusters, a number of studies have also considered the use of low-power "wimpy" nodes consisting of low-power storage (SSDs) and processors (mobile CPUs) [3, 19, 14, 17]. Primarily, these designs target computationally "simple" data processing tasks that are extremely partitionable, such as key-value workloads [3]. For such workloads, wimpy clusters were shown to be more energy efficient compared to traditional clusters built using more power-hungry server nodes. However, this result may not hold in scenarios such as database workloads which often exhibit sub-linear scale-up characteristics, especially when full cluster cost is considered [14].

Our work in this paper can be viewed as in the same category as the above-mentioned wimpy-node architectures. To our knowledge, we are the first to characterize the energy-efficieny of modern smartphones when running database-style tasks. Throughout this paper, we define energy efficiency of a workload as the ratio of the query completion rate (e.g., scans per hour) to the average power consumed by the system.

3. A CASE FOR MICRO-CELLSTORES

There are three main emerging trends in enterprise data management that lend themselves to a micro-data center design based on underpowered hardware:

- Massively Parallel Processing (MPP): Parallel DBMSs typically adopt the shared-nothing paradigm for scaling out (rather than scaling up) to deal with increasingly larger data volumes. For queries that scale linearly with the number of nodes in a cluster, an underpowered cluster could reach acceptable performance levels by using more nodes.
- Column-oriented and highly compressed data: Column-stores have emerged as the prevalent architecture for high-performance data management. Operating on columnar, highly compressed data eases pressure on the memory and network/IO buses (which are typically under-specced in a smartphone, due to concerns over manufacturing cost).
- Reliability through replication: Modern systems increasingly rely on replication for providing reliability, rather than on expensive hardware-based solutions. Such techniques are particularly suitable for smartphones which do not compare well to server-grade components with respect to reliability.

Furthermore, recent work has demonstrated running MapReduce jobs on a network of smartphones [6]. Micro-Cellstores are inspired by the above observations, combined with the expected abundance of used smartphones in the future (as explained in the introduction).

In the concept of Figure 1, the proposed housing structure contains standardized micro-USB connectors and, possibly, WiFi routers for connectivity. We also expect that there will be some form of storage directly connected to the router/hub. Furthermore, batteries may be leveraged in interesting ways, e.g., to provide uninterrupted operation even under intermittent power availability, to charge during off-peak hours at possibly cheaper rates, or to smooth out the cluster's power profile. Studying the tradeoffs between the different types of networking, deciding the best use for external storage, and exploring ways to harness the batteries are beyond the scope of this paper.

We expect MCSs based on cheaply acquired, used smartphones to be environmentally sustainable, minimizing total exergy cost. While our primary metric for efficiency in this paper is power consumption, it should be noted that cost-efficiency may have a favorable impact on our proposal, since

Year	Model	CPU	RAM	Storage (int./ext.)	WiFi
1996	Nokia 9000	33MHz AMD Elan x486	2MB	6MB	–
2002	Sony P800	156MHz ARM9	?	16MB	–
Q2 2007	iPhone	412MHz[1] ARM	128MB	4, 8, or 16GB	b/g
Q4 2008	HTC Dream	528MHz MSM7201A (ARM11)	192MB	256MB / microSD	b/g
Q2 2009	iPhone 3GS	600MHz[2] S5PC100 (Cortex-A8)	256MB	8, 16, or 32GB	b/g
Q4 2009	Motorola Droid	550MHz[3] OMAP3430 (Cortex-A8)	256MB	512MB / microSD	b/g
Q1 2010	Nexus One	1GHz QSD8250 (Snapdragon)	512MB	512MB / microSD	b/g/n
Q2 2010	iPhone 4	1GHz[4] Apple A4 (Cortex-A8)	512Mb	16 or 32GB	b/g/n
Q4 2010	Nexus S	1GHz S5PC110 (Hummingbird)	512MB	16GB iNAND	b/g/n
Q1 2011	HTC Thunderbolt	1GHz MSM8655 (Snapdragon)	768MB	8GB / microSD	b/g/n
Q2 2011	Droid Bionic	1GHz **dual-core** Tegra 2	512MB	2GB / microSD	b/g/n
Q2 2011	Galaxy S II	1GHz **dual-core** Exynos412 or Tegra 2	1GB	16 or 32GB	a/b/g/n

Table 1: Smartphone model feature summary.

smartphones would be otherwise discarded. A broader goal of this paper is to increase awareness of environmentally sustainable solutions for computing infrastructures, and the MCS concept is aimed towards that end. The rest of the paper focuses on specific aspects of the suitability of MCSs (which consist of ultra-wimpy nodes) for database workloads.

4. MODERN SMARTPHONES

Although the concept of what we today recognize as a "smartphone" is almost two decades old [1], until very recently the dominating characteristic of a "smartphone" was the "phone." Functionality was rather rudimentary and computing power was limited. The fairly recent explosion in the availability of reasonably fast wireless data networks has spurred demand for more capable computing devices, and vice versa, creating a virtuous cycle.

The current concept of a smartphone as an always-connected computing device that runs sophisticated applications was brought into the mainstream by the Apple iPhone, which was released four years ago. Since then, new, more powerful models are constantly introduced. Table 1 summarizes some key features of various smartphone models. Especially during the past two years, the smartphone space has witnessed exponential growth. Both CPU clock speeds and RAM capacity have roughly doubled in that time. Figure 2 illustrates the clock and memory trends over time.

In 2011, several companies are expected to introduce smartphones with dual-core CPUs. Furthermore, this trend does not show signs of slowing down. The ARM Cortex-A9 core design, on which these planned devices are based, supports up to four cores on the same chip and clock speeds up to 2GHz. Furthermore, since they are aimed at mobile devices, these designs focus on maintaining power consumption characteristics while increasing performance. Therefore, power efficiency should increase even further over time.

4.1 Experimental methodology

We collected measurements on a Samsung/Google Nexus S smartphone, running Android 2.3.3 (GRI40, with Linux kernel 2.6.35). We wrote logging software that records the following:

- Battery statistics, including battery level (%) and voltage, by listening for `BATTERY_CHANGED` broadcast events.
- CPU load statistics, by polling `/proc/stat` at a user specified interval (by default 60 seconds).

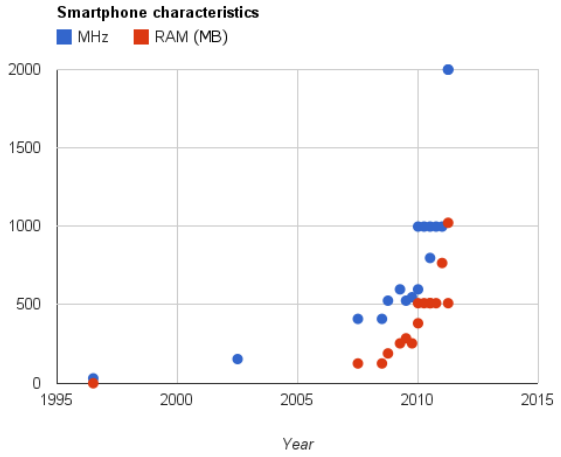

Figure 2: Smartphone clock speed and RAM over time.

- CPU frequency scaling statistics, by polling `/sys/devices/.../time_in_state`.
- Network connectivity changes, by listening for `CONNECTIVITY_ACTION` broadcast events.
- Network traffic statistics per interface, by polling `/proc/net/dev` (120 second interval, by default).
- Screen usage statistics, by listening for `SCREEN_ON` and `SCREEN_OFF` broadcast events.

Each logger is a separate component (Android service) that can be turned off when not needed. For each experiment, we only collect the statistics we need. Furthermore, we ran the device with logging fully enabled and compared baseline power consumption (see below) with logging fully disabled, and saw no measurable effect.

Log events are queued in memory and flushed to phone storage in batches (user parameter, typically 20 events). Statistics collected through broadcast event receivers are "pushed" only when something changes. For polled statistics we used Android alarm APIs. We kept logging to the minimum necessary and verified that it has no measurable effect on power consumption by comparing battery level drop with logging on and off, over a period of several hours.

During each experiment we acquired a *partial wakelock*, which prevents the CPU from sleeping (otherwise the O/S may power down the CPU when there is no user interaction,

even if processes are running). Beyond that, we kept the screen off, disabled all radios (cellular, WiFi, Bluetooth, and GPS) by setting the phone in airplane mode, and disabled all background services. Finally, before each run we charged the battery normally (i.e., no bump charging).

Since measuring battery capacity is difficult without specialized equipment, we used a fresh battery for our experiments. We converted ampere-hours to watt-hours using average voltage (time-weighted) during each experiment, based on battery voltage sensor values (typical range was 4 ± 0.03V). Android reports battery levels as integer percentages of capacity. On the Nexus S, a 1% drop corresponds to 15mAh or about 60mWh. We ensured that each experiment ran for at least one hour (much longer for idle power measurements), which implies an error of at most 10% (typical experiment power consumption was 0.7–1.2W).

The Hummingbird CPU in Nexus S uses dynamic frequency scaling (DVFS), supporting clock rates of 100, 200, 400, 800MHz and 1GHz. Effective clock rates were estimated by averaging frequencies, weighted by the fraction of time spent at each frequency based on O/S statistics in `/sys`. All experiments reported in this paper are with DVFS turned on, using the `ondemand` governor. This configuration is the most power-efficient overall. For CPU-bound workloads, the clock was indeed at or near its maximum. However, for disk-bound workloads, we observed that the O/S successfully scaled the CPU down to the minimum frequency that can handle the load (details ommited for space).

4.2 Characteristics

Table 2 summarizes some characteristic performance numbers for Nexus S. Sequential read bandwidths were measured by repeatedly reading a large enough array or file. Peak CPU power consumption was estimated at full load and 1GHz. Main memory and external storage bandwidths (361 and 25 MiB/s) are comparable to those of a mid-grade laptop (571 and 33 MiB/s; Intel SU9400 1.4GHz CPU and 500GB, 5400RPM drive). However, power consumption is an order of magnitude smaller (SU9400 TDP is 10W).

Description	Value
Memory read rate	361.9 ± 57.8 MiB/s
Storage read rate	24.8 ± 1.7 MiB/s
USB transfer rate	13.3 ± 0.8 MiB/s
CPU peak power	1090 ± 80 mW

Table 2: Nexus S characteristics.

We observed that CPU power consumption is effectively proportional to clock frequency [21] over a wide range of clock speeds. Figure 3 shows average power consumption versus effective clock frequency, with everything except the CPU turned off. Because the intercept is non-zero, power efficiency (mW/MHz) increases with clock rate.

5. ANALYSIS

Our goal in this section is to explore which types of data management workloads are best suited for MCS units, and compare the power consumption of smartphones to that of other, energy-efficient architectures, when running a range of parameterized workloads.

In previous work [18] we showed that, for a single-node DBMS server, the most energy-efficient configuration is typically the fastest one. In follow-up work-in-progress [10] we found that the same holds for low-power non-server architectures, such as laptops and desktops, and that laptops can be more energy efficient than servers for several types of database engine operations, such as scans, sorts, and joins. Therefore, for the purposes of this paper, we only compare against two energy-efficient platforms: a mini-desktop and a laptop with an Ultra Low Voltage (ULV) processor.

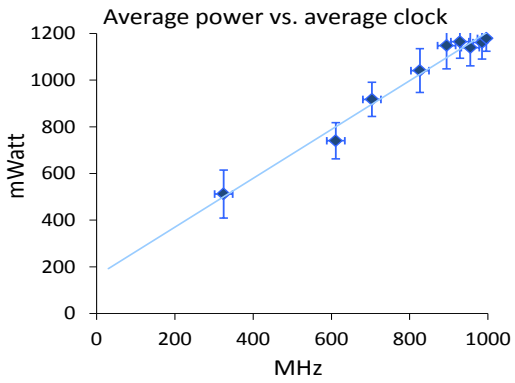

Figure 3: Averge clock frequency vs. power consumption.

In Section 5.1 we discuss what data management workloads are a natural fit for MCS units; in Section 5.2 we present the experimental setup and our results. We offer implications of our results in the concluding section.

5.1 Workload suitability for MCS

Low-power (or "wimpy") architectures trade single-node computational speed for higher energy efficiency. Compared to a server node, a task will run significantly slower on a wimpy node, but it will also consume much less energy. To make up for slower individual nodes, wimpy archictectures are typically positioned to run scale-out software infrastructure, with many more nodes than a server-based installation. For throughput-intensive tasks that can scale linearly with node count, this strategy is a win. Using more wimpy nodes increases total throughput, without changing energy efficiency (power and performance increase at the same rate). That result was also verified experimentally [19].

Recent work, however, pointed out that several complex parallel DBMS workloads exhibit sub-linear scalability, and therefore the energy efficiency of wimpy-node architectures may degrade as node count increases [14]. That analysis used total cost for purchasing and operating servers over a period of several years and showed that server-based clusters can be more cost-efficient than wimpy clusters, depending on the complexity of the workload. In our context, the final cost of an MCS unit is not clear, as it will depend on whether recycled phones come with a price tag. Thus, we compare only operating costs, i.e., energy efficiency.

The performance and power characterization of Nexus S from the previous section pointed that, while smartphones are extremely low-power devices, disk and network I/O speed as well as RAM size can be up to two orders of magnitude less than servers, or one order of magnitude less than previously proposed wimpy architectures. Therefore, we are interested in workloads with the following properties:

- Partitionable across a large number of nodes.
- Minimal network transfer.

Figure 4: Energy consumption per GB (lower is better), for a wide-tuple scan on four different configurations (Nexus S always operates on disk data).

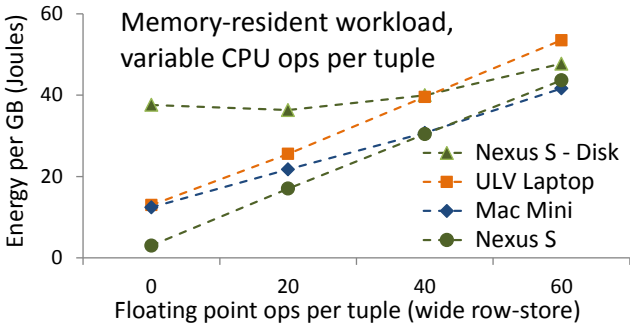

Figure 5: Energy consumption per GB (lower is better), for increasingly compute-intensive scans.

- Favor increased CPU processing cycles per byte read from disk.
- Do not require large in-memory structures.

Workloads that have the above properties include database scans that may be followed by additional (partitionable) operations (such as filtering, projection, aggregation) and certain types of joins that can run in a single-pass and involve small network transfers (e.g., when the inner table shuffled across all nodes also includes a highly selective predicate). In the rest of this paper, we focus on database scans without network transfers, with varying tuple widths (covering row/column-stores with lightweight compression), and with varying degree of CPU processing per tuple.

5.2 Experimental setup and results

We used the database storage manager developed in [8] to run a series of database scans. This is a block-iterator engine that can operate on both row- and column-oriented data. We ran the same C++ code on all three platforms. On Android we used the Native Development Kit (NDK, release R5b). We experimented with the `LINEITEM` table from TPC-H, using the same simplifications as in [8].

We compare Nexus S to two low-power systems: a 2010 Apple Mac Mini and an HP Compaq 2710p Tablet laptop. The Mac Mini features an Intel Core 2 Duo processor, whereas the HP Laptop features an Ultra Low Voltage version of the same processor. Both systems have 2GB RAM and relatively slow disks: a 40MB/sec SATAII HDD in the Mac Mini and a 80 MB/sec PATA SSD in the HP Laptop (we also report projected results based on faster disks). Table 3 summarizes configuration and power consumption details for all three systems. "Idle" is the power consumption at zero load. For the Mac Mini and the Laptop we used a Brand Electronics 20-1850 CI to measure total system power. This power meter has ±1.5% accuracy and collects readings once a second. Each experiment was repeated multiple times to get stable power measurements.

System	CPU (cores)	Power	Disk
Mac Mini	2.4GHz (2)	7.1W – 26.8W	41MB/sec
Laptop	1.2GHz (2)	11.1W – 23.7W	78MB/sec
Nexus S	1GHz (1)	0.2W – 1.17W	25MB/sec

Table 3: CPU specs and idle/peak power consumption of tested systems.

In our first experiment, we used all systems in an out-of-the-box configuration, to scan `LINEITEM` from disk, using a row-store representation (tuple width is fixed at 144 bytes). The results are in Figure 4 (leftmost part). This figure shows energy per GByte of data read (lower is better) for four different configurations of the two Intel systems. The rightmost configuration corresponds to the same scan when the table fits entirely in memory, whereas for the two middle configurations we recompute the consumed energy assuming two faster disks: one with 250MB/sec bandwidth and one with 500MB/sec. For the Nexus S we always show the energy consumption for disk-resident data.

Figure 4 shows that Nexus S consumes significantly less energy than the wimpy platforms for a row-store wide-tuple scan in an out-of-the-box configuration. The Mac Mini and ULV Laptop perform very poorly because they have fairly slow disks (thus running time is high) while paying a substantial up-front penalty for idle power (for example, the Mac Mini operates at 9.9–11.4W out of a 7.1–26.8W range). However, under more realistic assumptions about typical disk configurations, the wimpy nodes become competitive. When the wimpy platforms operate on memory-resident data, they consume less energy than Nexus S reading from disk.

For the remaining experiments we always show main-memory measurements for the two Intel systems, as the best-case scenario for those systems.

Next, we keep the tuple width the same, but experiment with increased number of computations per tuple. Figure 5 shows energy per GByte of data read (lower is better) for varying computation intensity. For this experiment we modified the code, by injecting a number of floating point operations to emulate worst-case processing for each tuple (e.g., complex analytic workloads). Data set size and tuple width are fixed for all runs. For the Nexus S we show both disk- and memory-resident performance.

As Figure 5 shows, the Nexus S consumes less energy than the other two systems when operating on memory-resident data. However, the gap closes when there are tens of floating operations executed in-between tuple reads. The energy consumption of Nexus S on disk-resident data converges slowly to the consumption with memory-resident data. In this experiment we wanted to show a large range of possible operations (typical per-tuple transformations correspond to a few floating operations).

In our last experiment, we compare energy efficiency when evaluating a single predicate using four different table storage representations: wide row-store (144 bytes), narrow row-store (32 bytes), wide column-store (8 bytes), and narrow column-store (1 byte). In all cases we use data from `LINEITEM` to create various projections of different width. The narrow

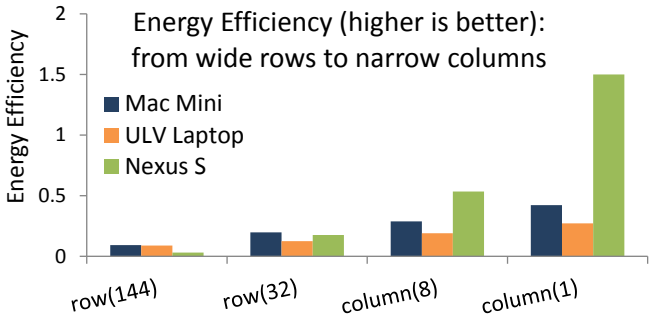

Figure 6: Energy efficiency (higher is better) for varying tuple widths and storage formats.

versions of the row and column tuples are also representative of compressed versions of the original tuples, as the additional overhead of lightweight compression is small [8]. Because we do not know how memory capacities will evolve, we compare the best-case scenario for the two Intel systems (memory-resident data) with the worst-case scenario for the Nexus S (disk-resident data); for memory-resident data, the efficiency of the Nexus S increases further by 1.2–2.6×.

Figure 6 shows the results. This time we compute the overall energy efficiency (higher is better) on the y-axis. While the two Intel systems always operate close to maximum CPU utilization, the Nexus S starts as disk-bound (for row-144) and becomes cpu-bound after row-32. For column-store scans, the Nexus S is significantly more energy-efficient than the other two systems—up to 6×, despite operating on disk-resident data. In accordance to [18], all systems become more efficient as they ran faster.

6. CONCLUSIONS

In this paper, we introduced the concept of a Micro-Cell-store (MCS) unit, a data appliance consisting of recycled smartphones. Through detailed power and performance measurements on a Linux-based current-generation smartphone, we assesed the potential of modern smartphones as a building unit for energy-efficient database appliances. Our results confirm that smartphones are overall a more energy efficient alternative, and further show that the gains become more significant for narrow tuples (i.e., column-oriented stores, or compressed row stores), achieving up to 6× improvement even when compared against other low-power options.

Our intention with the ideas presented in this paper is to motivate environmentally sustainable approaches, based on reusing and repurposing computing equipment. Towards this goal, there are several open questions around architecting MCS units and developing purpose-built data management software: How can total cost be competitive to that of traditional data centers? What other workloads could possibly run on this platform? What are the limits of scaling out data management tasks on wimpy or ultra-wimpy node architectures? Is there any benefit in combining MCS units with traditional servers to form hybrid data centers? If yes, what software changes would be needed? We hope these questions will motivate the research community to intensify their efforts on energy-efficient solutions.

7. REFERENCES

[1] IBM Simon. `http://en.wikipedia.org/wiki/IBM_Simon`.

[2] Report To Congress on Server and Data Center Energy Efficiency. In *U.S. EPA Tech. Report*, 2007.

[3] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: a fast array of wimpy nodes. In *SOSP '09*, 2009.

[4] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), 2007.

[5] C. Belady. In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling*, 23(1), 2007.

[6] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos. Misco: A MapReduce framework for mobile systems. In *PETRA*, 2010.

[7] G. Graefe. Database Servers Tailored to Improve Energy Efficiency. In *Software Engineering for Tailor-made Data Management*, 2008.

[8] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance tradeoffs in read-optimized databases. In *VLDB*, 2006.

[9] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy Efficiency: The New Holy Grail of Database Management Systems Research. In *CIDR*, 2009.

[10] W. Lang, S. Harizopoulos, M. A. Shah, J. M. Patel, and D. Tsirogiannis. Improving the Energy Efficiency of a DBMS Cluster. In *Submitted for publication*, 2011.

[11] W. Lang and J. M. Patel. Towards Eco-friendly Database Management Systems. In *CIDR*, 2009.

[12] W. Lang and J. M. Patel. Energy Management for MapReduce Clsuters. In *VLDB*, 2010.

[13] W. Lang, J. M. Patel, and J. F. Naughton. On Energy Management, Load Balancing and Replication. In *SIGMOD Record*, 2009.

[14] W. Lang, J. M. Patel, and S. Shankar. Wimpy Node Clusters: What About Non-Wimpy Workloads? In *DaMoN*, 2010.

[15] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower*, 2009.

[16] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. *SIGOPS Oper. Syst. Rev.*, 2008.

[17] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White. Low-power amdahl-balanced blades for data intensive computing. *SIGOPS Oper. Syst. Rev.*, 2010.

[18] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *SIGMOD '10*, 2010.

[19] V. Vasudevan, D. Andersen, M. Kaminsky, L. Tan, J. Franklin, and I. Moraru. Energy-efficient cluster computing with FAWN: workloads and implications. In *e-Energy '10*, 2010.

[20] Z. Xu, Y.-C. Tu, and X. Wang. Exploring Power-Performance Tradeoffs in Database Systems. In *ICDE*, 2010.

[21] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *CODES+ISS*, 2010.