Migratory Trash Clouds

Emily Ruppel Alexei Colin Brandon Lucia
{eruppel,acolin,blucia}@ece.cmu.edu
Carnegie Mellon University

1. Introduction

Reliable cloud computing is too expensive, as evidenced by a simple look at a bill from all of the cloud instances spun up to generate the data for this year's ASPLOS submission. The last several decades of Moore's law have yielded chips with exponential performance increases and costs dropping annually, but the need for more computation has arguably outpaced the improvement in cost per dollar, especially with the commoditization of computing at the warehouse-scale [1]. A major cost behind commoditized computing at warehouse-scale is power management, distribution, and storage. Cooling and power distribution add 50% to 100% to the cost of operating servers [12, 13]. Moreover, servers require maintenance of both their software and their hardware, which add additional cost in peoples' time and hardware components that need replacing. Often, the goal of all of this cooling, power distribution, maintenance, and hardware is to provide uptime, which translates to availability and reliability for customers. We observe that availability and reliability are often not important and we advocate for jettisoning power system, hardware, and software support for both.

Instead, we propose a model of commoditized computing that is opportunistic, unreliable, and sometimes unavailable, but incurs little of the cost associated with conventional commodity bulk computing solutions. We present a design for a computing system that implements this model using a geo-distributed collection of *low-reliability* clusters of discarded, then reclaimed hardware (e.g., old smartphones), powered through harvested renewable energy sources, like the sun. Repurposing outdated hardware minimizes the upfront cost of the computing system and reduces the environmental impact of the discarded goods. Using a re-newable power supply minimizes recurring operating costs, however presents the challenge of computing on an *intermittently-powered* processor. To ensure forward progress and guard against unrecoverable data corruption due to arbitrary power failures, the runtime software will leverage existing techniques for interruptible computation, such as two-phase commit [7], logging [14], and programming techniques from intermittent computing [10, 3, 11, 2, 4].

Each cluster is (sporadically) connected to other clusters and potentially to the Internet. Inputs and outputs for the computation on the cluster are transmitted either directly to individual nodes via their existing cellular modems, or through a per-cluster gateway with a specialized radio (e.g., LoRa [5]). Low power, long range communication [5] between clusters allows workloads to migrate to clusters with greater energy availabilty. To eliminate the high energy cost of high-bandwith communication, however, the workload can follow energy availability not by migrating to a different cluster but by physically moving the cluster itself towards the energy source. We envision *mobile* clusters as a payload aboard a perpetually aloft, long-haul solar drone glider. The drone can be programmed to fly above the clouds and follow the sun, providing best-effort continuous availability for the cluster.

Migratory Trash Clouds provides a best-effort guarantee to the customer, because clusters can only compute when a power source is available. Assuming solar power, Migratory Trash Clouds provide only the guarantee that a job will complete eventually, as the weather permits. Migratory Trash Clouds fill a similar niche to the one filled by Amazon EC2 Spot instances: compute power is transiently available when resources are available and when resources disappear, the compute power disappears with them. In the case of Migratory Trash Clouds, the limiting resource is the energy that is extractable from the environment.

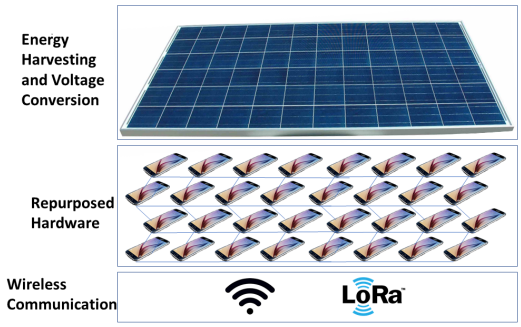
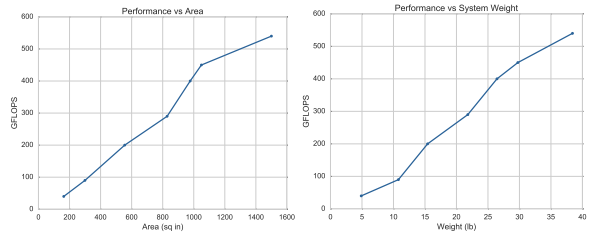


Figure 1: Migratory Trash Clouds components

2. System Overview

Migratory Trash Clouds are a distributed collection of compute clusters, composed out of trash smartphones that were thrown away or donated, powered by a single solar panel with minimal power conditioning circuitry (e.g., voltage conversion and a decoupling super-capacitor). The clusters may be stationary or mounted on a solar-powered flying drone that can follow the energy source. Each cluster is composed of tens of devices powered directly by harvested solar energy and equipped with wireless radios for wifi, 4G, and LoRa. Within a cluster, one device is designated as cluster manager and monitors status information about the other local processors and the cluster's energy availability.

Figure 1 shows the physical design of a Migratory Trash Clouds cluster. A cluster features a 100W solar panel affixed to about 30 smart phones connected via a WiFi mesh and 4G and LoRa radios for long-range wireless communication. We assume that each phone is a device circa 2016 such as a Samsung Galaxy S6 with four ARM Cortex-A57 at 2.1GHz and four Cortex-A53 cores at 1.5GHz, 3GB of RAM and a 64GB off-chip flash memory. Each device is capable of roughly 10 GFlops [9] at a power consumption of around 2W. Performance per Watt of a Migratory Trash Clouds cluster is not competitive with emerging devices like the 1 TFLOP per 15W in the NVidia Jetson TX2 "Parker", however the cost of a cluster, i.e. a set of *upcycled* smartphones, is competitive. Assuming 2W per device and 77% efficiency of voltage conversion, a 100W solar panel delivers 66W, sufficient to continuously power 33 devices. In total, a cluster provides around 330 GFlops of performance for 66W of "free" power, collected from the environment.



(a) Area (b) weight

Figure 2: Migratory Trash Clouds area and weight per FLOP

To gather the "free" power, however, a fixed cost of a solar panel must be incurred. A high-end 130W panel costs approximately $300, and its cost is amortized across all elements in the cluster. We anticipate the panel cost and the device cost to dominate the price of a cluster. The panel also dominates area and volume of the cluster, occupying approximately 7.3 square feet. In our design, compute and radio hardware is situated underneath the solar panel, as the total area of 33 smart phones is approximately 4.1 square feet. The system weight must be minimized for mobile clusters, to fit within the payload of a solar-powered drone. The entire system weight is approximately 30 pounds, given 0.2 lb (100g) per phone if the battery is removed. Figure 2 shows the design space of performance versus cluster area and cluster weight based on the dimensions of commercially available solar panels [15].

3. Interruptible Computation

Intermittent computing addresses challenge of continuing computation across unpredictable frequent reboots in simple, embedded processors. Intermittent programming models work because they ask the programmer to write their program in *idempotent tasks* that are amenable to frequent restarts with low overhead [10, 3, 11, 16]. Intermittent computing follows in the footsteps of a long history of operating systems and databases work that developed a diversity of programming and execution models for various reliability benefits (e.g., transactions [8]). Recent work on Apache Spark [6] showed that, as in tiny intermittent systems, large-scale systems can also reap the benefits of idempotent computational tasks, making migration simple, allowing for arbitrary restarting, and eliminating often costly side effects by construction. Software in Migratory Trash Clouds can leverage these ideas to ensure reliable operation despite unpredictable weather and unreliable power. Moreover, with batteries removed (for weight reduction in mobile clusters) devices will operate intermittently, requiring software techniques for reliability and interruptibility.

4. Communication

Devices in a cluster must communicate with one another to compute and must communicate with off-cluster devices to collect inputs and produce outputs. Within a cluster, devices communicate using a WiFi mesh. Communication over long distances is a challenge because existing low power WANs have low bandwidth. For instance, LoRa [5] radios have line-of-sight (LOS) transmission range up to 15km, but data rates of only 50 kbps. The advantage of LoRa radios is that they add little additional power: transmit power is approximately 0.25W at maximum power and receive power is 0.03W [5].

Figure 3: A Migratory Trash Clouds compatible drone

An alternative to LoRa is to use 4G radios, which have long range and high bandwidth. A key drawback to 4G, however, is the need for a per-device subscription for access to terrestrial infrastructure. We advocate for an alternative solution that does not require pay-per-byte infrastructure.

To avoid prohibitively low bandwidth and per-byte bandwidth costs, we envision deploying quad-copter *data drones* as a "station wagon full of hard drives" in the sky. A data drone can deliver a batch of input or collect a batch of output from an mobile (glider-attached) or stationary cluster via ad hoc wifi mesh at high data rate. Data drones are more efficient than radio when the total energy, money, and time cost required to transmit a workload via LoRa or 4G exceeds the cost of launching a drone. Quantitatively comparing these costs is an open research question.

5. Energy source tracking

A mobile Migratory Trash Clouds cluster can compute perpetually, by directing its carrier drone to follow the sun and stay above the cloud cover. During daylight hours the mobile clusters will always have access to energy. Each glider ensures that it always operates during daylight by flying west at the rate of the movement of the sun to ensure constant access to energy. With a well-planned trajectory, migration toward the power source allows a cluster to compute continuously.

6. Concluding Research Questions

Migratory Trash Clouds raises several interesting research questions that require additional exploration to scale out energy efficient, low cost computation.

- When is communication between nodes advantageous in a distributed system with very low data rate communication?
- How can cluster management be made robust to frequent failures in a highly energy constrained and resource limited settting?
- Can continuation strategies used in cloud computing be applied to improve the reliability of a cluster with unpredictable energy availability?
- Is continuous flight and computing possible with state-of-the-art solar drones?
- When are data drones beneficial? Is there a case where data rate, cost, and latency demand physical cluster access?

References

[1] Luiz Andre Barroso and Urs Hoelzle. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[2] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. An energy-interference-free hardware-software debugger for intermit-

tent energy-harvesting systems. *ACM SIGPLAN Notices*, 51(4):577–589, 2016.

[3] Alexei Colin and Brandon Lucia. Chain: tasks and channels for reliable intermittent programs. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 514–530. ACM, 2016.

[4] Alexei Colin, Emily Ruppel, and Brandon Lucia. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proceedings of the 51st International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2018.

[5] Adwait Dongare, Craig Hesling, Khushboo Bhatia, Artur Balanuta, Ricardo Lopes Pereira, Bob Iannucci, and Anthony Rowe. Openchirp: A low-power wide-area networking architecture. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 569–574. IEEE, 2017.

[6] The Apache Software Foundation. Lightening-fast cluster computing. https://spark.apache.org/, 2017.

[7] Jim Gray and Leslie Lamport. Consensus on transaction commit. *ACM Trans. Database Syst.*, 31(1):133–160, March 2006.

[8] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1992.

[9] Florence Ion. Benchmarked: The galaxy s6 is the fastest android phone. period. https://www.greenbot.com/article/2904384/benchmarked-the-galaxy-s6-is-the-fastest-android-phone-period.html, 2015.

[10] Brandon Lucia and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. In *ACM SIGPLAN Notices*, volume 50, pages 575–585. ACM, 2015.

[11] Kiwan Maeng, Alexei Colin, and Brandon Lucia. Alpaca: Intermittent execution without checkpoints. In *Proceedings of the 2017 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM, 2017.

[12] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: Eliminating server idle power. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, pages 205–216, New York, NY, USA, 2009. ACM.

[13] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": Temperature-aware workload placement in data centers. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pages 5–5, Berkeley, CA, USA, 2005. USENIX Association.

[14] Mendel Rosenblum and John K Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems (TOCS)*, 10(1):26–52, 1992.

[15] Inc SolarTech Power. F-series solar panels. http://www.solartechpower.com/fseries.html, 2018.

[16] Joel Van Der Woude and Matthew Hicks. Intermittent computation without hardware support or programmer intervention. In *Proceedings of OSDI'16: 12th USENIX Symposium on Operating Systems Design and Implementation*, page 17, 2016.