Information Batteries

Storing Opportunity Power with Speculative Execution

JENNIFER SWITZER, UC San Diego, USA
BARATH RAGHAVAN, USC, USA

Coping with the intermittency of renewable power is a fundamental challenge, with load shifting and grid-scale storage as key responses. We propose Information Batteries (IB), in which energy is stored in the form of information—specifically, the results of completed computational tasks. Information Batteries thus provide storage through speculative load shifting, anticipating computation that will be performed in the future.

We take a distributed systems perspective, and evaluate the extent to which an IB storage system can be made practical through augmentation of compiler toolchains, key-value stores, and other important elements in modern hyper-scale compute. In particular, we implement one specific IB prototype by augmenting the Rust compiler to enable transparent function-level precomputation and caching. We evaluate the overheads this imposes, along with macro-level job prediction and power prediction. We also evaluate the space of operation for an IB system, to identify the best case efficiency of any IB system for a given power and compute regime.

1 INTRODUCTION

Within the next twenty years humanity must eliminate fossil fuel use to avoid dangerous climate change [28, 37]. To do so requires renewable electricity generation. While there have been dramatic decreases in the cost of wind and solar, their intermittency in response to weather and solar irradiance is a widely-known issue [9, 10]. This intermittency is often out of phase with overall demands, so when renewable production is high, prices tend to be low (Figure 1).

The research community has explored myriad responses to this intermittency [3, 13, 22, 27, 33, 34, 51, 52, 55, 60], but has been stuck between the Scylla of dynamic load shifting and the Charybdis of expensive grid-scale storage. Dynamic load shifting at grid scale requires the rare combination of *flexible*, *large*, and *ubiquitous* loads. Grid-scale storage with proven technology requires nearby hydroelectric capacity or expensive battery arrays.

In this paper we aim for the best of both worlds: storage of surplus renewable production through the load shifting of computation with ***speculative execution***. Computation is near-infinitely divisible, flexible, large, ubiquitous, and can be stored cheaply. This approach, **Information Batteries** (IB), entails storing energy as completed precomputations that can, as we show, meet or exceed the end-to-end efficiency of grid-scale storage *using existing infrastructure*. Not all workloads or conditions will yield high efficiency with this approach, so a key aspect of our exploration is its limits.

Background. Even with relatively modest adoption of renewables and despite the inherent statistical multiplexing of large power grids, grid operators will soon face the problem of too much power. During the middle of the day in California it is now often the case that there is too much power being produced, largely due to solar, driving the price of electricity *negative* [9]. This problem has arisen with just 20% of California's electricity generation coming from solar (the renewable source with the greatest growth potential) [2].

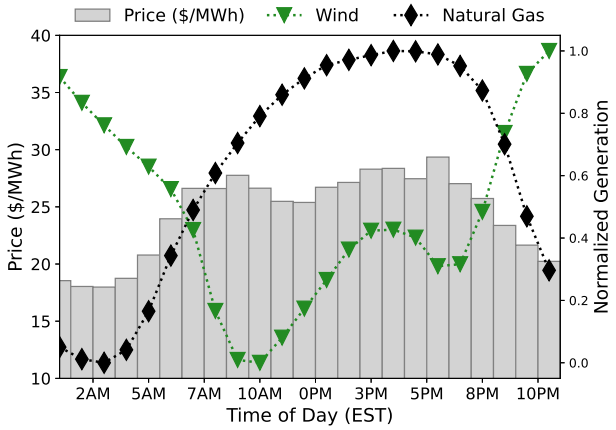


Fig. 1. Gas generation is flexible; wind generation requires wind, and is often out of phase with demand. Prices tend to be lower when wind generation is higher. Based on 2019 price and fuel mix data from MISO [36].

Today we face both power dumping *and* load shedding due to insufficient power during peak times. This disconnect in supply and demand, as renewable energy sources under-produce during low availability times and over-produce during high availability [58, 61], is the key challenge that must be solved to adopt renewables.[1]

An oft-proposed strategy to address this problem is to increase grid-scale storage via traditional energy storage systems such as lithium-ion batteries and pumped hydro [6]. However, such systems require a relatively high initial investment and have siting constraints; also, adding enough storage to soak up all excess production is prohibitively expensive [10]. Similarly, smart-grid advocates often observe that if the grid could simply signal to individual devices, such as appliances, when to consume power, this demand shifting could adapt to power availability. However, as with storage, demand shifting requires both grid upgrades—a proposition that a number of governments have balked at—and wide-scale adoption of new smart loads.

Approach. With Information Batteries, we propose the storage of energy as information, using the large and growing footprint of computing to perform both the functions of storage and load shifting. This approach hinges on three observations. First, data centers worldwide consume large amounts of electricity (250–500 TWh in 2018) and are projected to become even more power-hungry

Authors' addresses: Jennifer Switzer, UC San Diego, San Diego, California, USA, jfswitze@ucsd.edu; Barath Raghavan, USC, Los Angeles, California, USA, barathra@usc.edu.

[1] This negative-priced power and curtailed power (dumped power) are together referred to as *opportunity power* [10].

(840–3640 TWh in 2030 [1]).[2] Second, many computational tasks can be precomputed in whole or in part. Third, both power availability *and* compute demands are somewhat predictable, and thus it is possible to do *speculative* load shifting.

Rather than storing excess energy as a chemical (lithium-ion) or gravitational (pumped hydro) potential, an IB system stores this as information—completed computations. When excess renewable energy is available, an IB system uses this excess to perform precomputable, energy-intensive computations. The results of these computations are then stored for when they are needed. The duality of computation and energy is not new—it has been studied extensively in information theory, such as in the context of adiabatic computing [16]. However, this observation has not been raised to the level of grid-scale energy systems.

We study *the limits of Information Batteries* in enabling a renewable energy transition at a macro scale. For Information Batteries to be effective and worthwhile, they must 1) be less expensive than traditional storage systems, 2) shift significant computational work from grid to opportunity power, and 3) achieve this significant shift for some common workloads rather than new, esoteric workloads. As we show, it seems that this is only possible for some workloads and in some contexts.

We take a distributed-systems perspective, and evaluate the extent to which an IB storage system can be made practical through augmentation of compiler toolchains, key-value stores, and other important elements in modern hyper-scale compute. Our ability to shift computational load to opportunity power hinges on the accuracy of our predictive engine. If we cannot predict upcoming requests with at least reasonable accuracy, our system will not have results available for when they are requested, and may end up wasting opportunity power performing computations that are never requested. Furthermore, we must be able to accurately predict the future availability of opportunity power, so that we can effectively schedule our computations to take advantage of it. Finally, we must ensure that the cost of retrieving cached results is significantly smaller than the cost to compute them; otherwise, it is more efficient to perform the computation on demand.

Contributions. This paper makes three contributions:

(1) We introduce the idea of Information Batteries that provide a new speculative load shifting mechanism to address grid-scale renewable energy intermittency.
(2) We explore the design space of Information Batteries and show that in some common power and compute regimes there is the potential for an IB system to deliver efficiencies better than the best grid-scale storage available while in others IB systems provide little benefit.
(3) We implement a proof-of-concept IB system by augmenting the Rust compiler to enable transparent function-level precomputation and caching. We evaluate overhead, along with macro-level job prediction and power prediction.

[2]While some companies, such as Google, balance their data center power usage with renewable energy power purchase agreements, this still addresses only *average* power generation, not the peaks and troughs of generation.

2 CONTEXT

In this section we motivate the need for and feasibility of Information Batteries. We focus on two renewable energy markets: the Midcontinent ISO (MISO), which operates in the Southern and Midwestern United States and parts of Canada, and the California ISO (CAISO), which covers all of California. This allows us to narrow our scope while still considering both wind-dominant (MISO) and solar-dominant (CAISO) markets [35].

2.1 Availability of opportunity power

Opportunity power in CAISO and MISO is significant, growing, and often available. Current estimates place the yearly combined opportunity power of CAISO and MISO in 2017 between 7–20 TWh per year [9, 10]. In MISO, opportunity power is available 99% of the time (meaning opportunity power is available *somewhere* 99% of the time, since prices are location-dependent), and often in intervals of >100 hours [10]. In CAISO, some solar generators experience 3.3 hours of opportunity power per day [9].

Solar and wind energy are projected to be the fastest growing sources of electricity generation in the U.S., and currently account for 10% of total U.S. electricity generation [7, 18]. As solar and wind generation grow, so too will the amount of curtailed and negative priced power. Indeed, [9] measures the compound annual growth rate of opportunity power in CAISO to be 40%. Assuming 1.5 TWh of opportunity power in CAISO in 2017 (a conservative estimate) and a constant growth rate, CAISO alone could provide 22 TWh of opportunity power by 2025, enough to power all of Los Angeles.

2.2 Limitations of traditional energy storage

Energy storage is a simple response to overproduction. However, current battery prices make this untenable; grid-scale lithium-ion storage costs $356 per kWh today [38], not including installation costs. A naïve analysis of CAISO and MISO data, assuming 1.5 TWh/year and 6 TWh/year of opportunity power, respectively, yields a conservative estimate of $35 million to add one hour of storage to CAISO, and $140 million to add one hour of storage to MISO. A more complex analysis [10] suggests that adding grid-scale storage provides diminishing returns, and that adding 50 hours of storage to MISO would cost $50–400M *per wind generation site*, on par with the cost of the turbines themselves.

2.3 Non-computational load shifting

Existing non-computational flexible loads include manufacturing facilities, EV charging, and adaptive home appliances [15, 52].

Household or otherwise small-scale (but widespread) loads are a popular demand shifting target, such as in smart homes with or without storage [3, 22, 27, 33, 51, 55, 60] or in smart buildings more generally [34]. EV charging is a growing, flexible load [13]. Such load shifting requires accurate prediction capabilities [30, 32] such as of renewable generation and weather [5, 46, 47].

2.4 Computational load shifting

Some have considered to simply store power in data centers using the batteries in those facilities rather than shifting load [19, 21, 59]. Prior work has suggested shifting the loads themselves to leverage surplus power [11], for example by examining the price for computation (or power itself) in different regions [31, 41].

Data centers have more than enough capacity to soak up opportunity power. American data centers consume 70TWh/year, 1.8% of the country's total energy consumption [49]. With opportunity power in CAISO and MISO estimated at 7–20TWh/year [9, 10], opportunity power has the potential to provide between 10–30% of the energy needed by data centers.

Prior work also considered scheduling large-scale compute tasks under a variety of constraints. Speculative execution has been long known in computing [8, 50], but has not been applied in this context. Google recently announced their "carbon-intelligent computing platform" that attempts to reduce carbon emission of datacenters by aligning time-insensitive tasks with periods of high renewable energy availability [42]. In this paper, we will refer to this and similar techniques as time-shifted compute.

3 INFORMATION BATTERIES

Next we address the challenges of: 1) properties of computational tasks, hardware, and energy grids that make them more or less suited to Information Batteries, 2) the key elements of an IB system, and 3) the best case scenario/theoretical limits of IB systems.

3.1 Prediction

There is an element of speculation to any demand shifting: the assumption that the work done will be useful at some future time. However, most flexible loads are general tasks that will almost certainly be useful (e.g., charging an EV, washing clothes), and are therefore non-speculative. Precomputation, on the other hand, is speculative. Given the infinite space of possible computations, more work is needed to identify computational work that will be useful at some point in the future. The ability to predict future computations—to perform *task prediction*—is therefore key.

3.2 Granularity

Prior work on flexible loads has focused on high-granularity tasks—washing clothes, charging a car, even time-shifting large-granularity computational tasks [42]. Micro demand shifting is comparatively understudied. IB systems allow for tuneable load granularity, since computational tasks can (at least in theory) be broken into different-sized tasks (although data dependencies and complexity considerations may in practice favor large tasks).

3.3 Speculative demand shifting

The IB approach of demand shifting differs from prior approaches in two ways: the load itself is not pre-existing (as it is *speculative*), and the granularity at which we shift varies. Computation can be speculatively executed at many granularities, from whole-system to individual instructions.

Speculative load shifting uses energy to store not energy but *information*. This is different from conventional load shifting, which does not require storage (as the load itself has been shifted). However, since data storage is far less expensive than energy storage, this requirement is a minor imposition.

3.4 Computational loads

Information Batteries are well-suited to tasks with high predictability and a large potential speedup. Potential speedup is the ratio between run time and IB cache latency. Given the same cache latency, longer running tasks will have higher potential speedup. Applications that fit these requirements include:

(1) **Machine learning.** OpenAI notes that AI workloads, particularly training, are growing exponentially [39]. Such workloads are ideal for Information Batteries due to their size, latency insensitivity, and high predictability.
(2) **Video transcoding.** Video streaming now accounts for 75% of web traffic [4], so video transcoding—the process of converting a video from one resolution to another—has become an important cloud workload. Video transcoding has the potential to be highly predictable, since consumer behavior drives which videos are requested.
(3) **Large-scale data analytics.** Companies like Facebook collect a huge amount of data [24], the analysis of which is both time and resource intensive. In many cases, this work can be performed asynchronously.

In addition to predictability of macro workloads, the IB approach can precompute some or all of the fragments of expected jobs and then reassemble these fragments on demand. Thus predictability extends not only to whole jobs but sub-job fragments. Furthermore, with careful binary and execution trace analysis, it is likely possible to perform computations using speculatively precomputed fragments of *different* jobs, as many workloads have some commonality. Indeed, assembling whole computations out of fragments of code is well studied in the literature in very different contexts, such as in the case of Return-oriented Programming [43, 45]. Exploring the efficiency of fragment precomputation and reassembly is worthy of study but beyond the scope of this paper.

3.5 Grid properties

For an IB system to be successful, renewable generation must be large, long-lasting (e.g., with high duty cycles), and predictable. This is influenced by consumption patterns, the power mix of the grid, the physical locations of power sources, and the amount of traditional (battery) storage available.

3.5.1 Renewable energy availability. The grid power mix impacts energy availability and pricing. Energy generation from fossil fuels can be scaled up or down to meet demand, hydro generation is relatively stable and flexible, and intermittent renewables (primarily wind and solar) produce according to environmental conditions. These different patterns have an impact on pricing: when generation is higher than demand, prices drop, as expected.

3.5.2 Power prices. Negative power prices are the best indicator of uneconomic production. Prices are used as a proxy for renewable oversupply because, unlike renewable production data, they tell us something about the current balance of supply and demand. If renewable production is high, but so is consumer demand, then adding more demand will be less useful. In order to maximize the benefits of the system, we should prioritize scheduling work for periods of time when renewable production is high *and* demand is low, e.g. when prices are low.

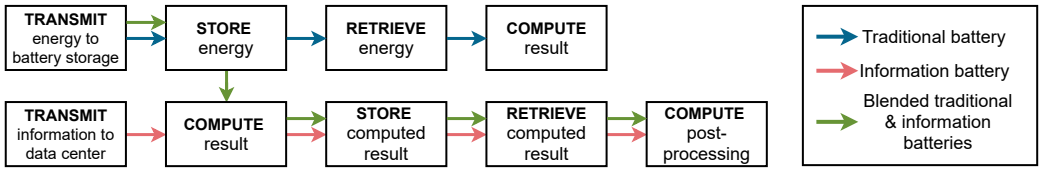


Fig. 2. Energy flows differently through Information Batteries and traditional batteries.

3.6 Framework

The key idea behind Information Batteries is quite simple: when renewable energy is available in excess, we use it to speculatively perform computation. The challenge is in determining *what* computation to perform, *where* and *when*, and *how* these computations should be done to make it efficient to retrieve their results later. The energy expended to perform the computation is therefore stored as the result of a computational task.[3]

4 DESIGN

Next we describe the key elements of an IB system, with a focus on two goals: to maximize the shift of compute to opportunity power, and to do so at a lower cost than traditional energy-storage systems.

4.1 Overview

We explore the components of an IB system, from compiler toolchains to ML-based tools for power and job prediction. We discuss how these could be combined in a production system.

Compiler Toolchain. Application binaries must be augmented to allow for run-time caching and retrieval of results.

Key-Value Store. The system must include an efficient key-value store for run-time caching and retrieval. In particular, the latency of precomputed result retrieval from the key-value store must be significantly less than the latency of computing that result directly.

Job Scheduler. A management system must be in place for scheduling incoming jobs during periods of opportunity power, and returning results once they are available. In particular, jobs must be prioritized according to their latency sensitivity and run time. The job scheduler must therefore be aware of the target workload.

Predictors. Information Batteries rely on speculative execution, and therefore require accurate predictions of future conditions. Specifically, the IB system must be able to predict: 1) whether or not opportunity power will be available in the near future (e.g., what the price of energy will be) and 2) what jobs will be requested in the near future, so that these jobs can be procomputed.

4.2 Program instrumentation

Memoization is a well-known technique used to improve the performance of programs by caching the results of expensive computations for later use. Previous work has considered the benefits of program memoization at the trace level, the basic-block level, and the function level [14, 20, 25]. In this paper, we employ function-level memoization, but later propose generalizations to this.

[3]There is a long history of work in lower-level aspects of the relationship between energy and information, such as work on adiabatic computing [16].

```
cycles_trad = 0, cycles_op = 0
every 5 minutes:
  predict = model price prediction
  actual = actual price according to trace
  job_size = number cycles per job
  hit_rate = accuracy of task prediction
  if predict<0 and actual<0: #true positive
    cycles_op += job_size
    cycles_trad += (1 - hit_rate) * job_size
  elif predict<0 and actual>0: #false positive
    cycles_trad += job_size
                + (1 - hit_rate) * job_size
  elif predict>0 and actual<0: #false negative
    cycles_trad += overhead of memoize
  else: #true negative
    cycles_trad += overhead of memoize
```

Fig. 3. Logic of the IB simulator.

Prior work has shown that a subset of functions can benefit from software memoization at compile and load time [53]. These functions share the following characteristics: expensive, side-effect free, critical, and repetitive arguments [54].

We apply these memoization techniques to precompute results for memoizable functions in the program's critical path. These results obviate later execution with non-renewable power. We achieve this by instrumenting all application binaries to use a key-value store to memoize the results of function calls.

4.2.1 Compiler extensions. We consider one specific approach to program instrumentation—compiler extension—and highlight other possibilities later. In this approach, all source code is compiled using our customized compiler, which inserts the appropriate hooks to precompute and retrieve precomputed results.

For each call to a precomputable function, the compiler inserts a fetch instruction to first check whether the function has been precomputed, and return the precomputed result if it exists. Note that since the fetch occurs at run time, it is important that its overall latency is significantly lower than the run time of the function to be computed. It is therefore important both that our cache implementation be efficient, and that our definition of precomputable function excludes those with extremely short runtimes.[4]

4.2.2 Caching infrastructure. The IB cache must enable fast caching and retrieval of precomputed results. In addition, for deployability, we would like to use a generic key-value store for caching rather than a IB-specific system. Here, latency and hit rate are important because we do not wish cache retrieval to itself induce higher overheads than computation itself. In addition, the memory footprint of the cache is important; results that require too much storage will increase the cost of the cache infrastructure itself.

[4]The definition of *precomputable function* is flexible, and may differ between workloads.

Parameter	Description
Cache latency	Storage and retrieval overhead
Cache hit rate	Accuracy of task predict model
Job length	Average job length
Price predict false positives	Fraction of time model mistakenly predicts negative-priced power
Price predict false negatives	Fraction of time model fails to predict negative-priced power

Table 1. Input parameters for the system-level IB simulator.

4.3 Scheduling

There are three main components to the IB manager: the scheduler, which receives computational tasks and schedules them to maximize grid power savings; the price predictive engine, which makes predictions about upcoming power prices; and the precomputation engine, which makes predictions about upcoming tasks, and performs precomputations on these predicted tasks whenever opportunity (negative-priced) power is available. If the incidence of negative-priced power is too low, the threshold of what is considered "negative" can also be set to some small positive number.

The scheduler receives tasks, and determines when and how they should be computed. Submitted tasks are of the form `source code, deadline`, where `deadline` indicates the latest timestamp at which the result is needed.

If opportunity power is currently available, the task can be computed immediately. Otherwise, the scheduler asks the renewable predictive engine when opportunity power will next be available. If the next available window of opportunity power is within the task's deadline, the task is scheduled.

Note that even if a task is not precomputable, the scheduler will still attempt to schedule it for a period of opportunity power. Thus any task with a generous enough deadline can be scheduled to use opportunity power. The scheduler also forwards the source code and the time it was received to the precomputation engine, which bases its predictions on this real-time stream of task requests.

The renewable predictive engine forms the core of the scheduler; the scheduler's overall effectiveness is dependent on the ability of the solar predictive engine to correctly predict opportunity power. An inaccurate solar predictor risks missing out on opportunity power, or erroneously scheduling tasks during regular grid power.

The precomputation engine is responsible for predicting upcoming tasks, pre-computing them, and caching the results. There are two main components of the precomputation engine: the task predictor and the precompute manager. The task predictor receives a continuous stream of task requests from the scheduler and uses a recurrent neural network to predict task requests.

The precompute manager consists of a single event loop that queries energy prices every 5 minutes until they fall below some small threshold. At this point, the manager requests 5 minutes worth of computational tasks from the task predictor. The precomputation engine functions in 5-minute increments since that is the smallest granularity at which energy prices are set in CAISO and MISO.

Before Instrumentation:

```
1  ...
2  fn square(a: u32) -> u32 { a * a };
3  let result = square(x);
4  ...
```

After Instrumentation:

```
1  ...
2  fn square(a: u32) -> u32 { a * a };
3  let result = memoize(square, "square", x);
4  fn memoize(f: fn(usize) -> usize, fname:
       String, a: usize) -> usize {
5      let cached = db.get(fname, a);
6      if cached.is_some() {
7          return cached.unwrap();
8      }
9      else {
10         let computed = f(x);
11         db.put(fname, a, computed);
12         return computed;
13     }
14 }
```

Fig. 4. Instrumented programs first check for precomputed, cached results and use those results if found.

4.4 Integration

Information Batteries are designed to work with existing data centers. Some, very limited processing power is reserved for the IB manager, which manages the scheduling of both real-time computational tasks and precomputation. A cluster of machines or VMs is designated for precomputation. The IB cache, which stores the results of these precomputations, is kept local for quick retrieval. No additional infrastructure is needed.

Although Information Batteries have thus far been described as an alternative to traditional energy storage, it is also possible to use them in conjunction with traditional batteries. Figure 2 illustrates the flow of energy in a traditional and information battery system, and how these two could be combined.

5 IMPLEMENTATION

Next we describe our proof-of-concept implementation of Information Batteries, which has three key components: 1) a Rust compiler augmentation for function-level precomputation, 2) a price prediction model for both CAISO and MISO, and 3) a function-level task prediction model. Benchmarking these components allows us to realistically parameterize our IB simulator.

5.1 Rust compiler instrumentation

We augment the Rust compiler to do function-level precomputation. This is accomplished at the MIR (Mid-Level Intermediate Representation) stage of the Rust compiler [29]. At this stage, the program has been converted into a control-flow graph (CFG) representation [44]. We implemented our instrumentation as an extra pass through the CFG, which we call the memoize pass.

The memoize pass replaces each function call with a call to our `memoize` function[5]. This takes as input a pointer to the original function, the function's name (fetched at compile time) and the original arguments to the function. `memoize` does the following: 1) checks if the function has been called with the particular arguments before,

[5]We require that the input binary include the definition of memoize

and 2) executes and caches the result of executing the function if not. We refer to this instrumentation as the `memoize` wrapper. For simplicity, we support only functions with the function signature `fn(u32)->u32`, but in practice other signatures could be supported.

Figure 4 shows the instrumentation of a simple program with `memoize` (code simplified for readability).

5.2 Cache

The performance of the precomputation engine is highly dependent on the cache implementation. Latency should be minimized as much as possible. For our proof of concept, we use pickleDB-rs [40] to implement a simple, local key-value store. However, any key-value store could be used in practice.

5.3 Price Predictor

We implemented our price models using TensorFlow, as Recurrent Neural Networks (RNN) with one LSTM layer and one dense layer. This is similar to techniques used for weather prediction [30, 57]. We collected training data from historical 5-minute Location Marginal Prices (LMPs) reported by MISO [36] and CAISO [23].

LMP is the dollar cost of supplying the next MW of power at a specific geographic region [48]. Since pricing is location dependent, the model makes predictions based on time of day and geography. Given a location and a 5-minute interval, the price predictor returns a prediction for the next hour's worth of LMP prices. We define opportunity power as being available any time the LMP prediction drops below some small, tune-able threshold.

5.4 Task Predictor

We implemented task prediction using TensorFlow, as an RNN with one LSTM layer and one dense layer. We collected training and validation data from several open-source Rust crates: Substrate [17], Iced [26], and Juniper [12]. For each crate, we generated a function-level trace on their provided example applications, and used these traces to train the model.

The resulting model takes as input a series of function calls, and a specification, N, for how far in the future to predict. It returns a prediction for the next N function calls. This is similar to techniques used for text prediction [56]. Note that this approach is quite simplistic. We do not consider, for instance, the value of arguments to the function. Our implementation is intended as a small proof-of-concept only.

6 EVALUATION

Our evaluation is in two parts. First, we microbenchmark each component of our proof-of-concept implementation described in Section 5. We then use these microbenchmarks, combined with real price data from CAISO and MISO, to provide a realistic parameterization of a system-level simulation.

6.1 Microbenchmarking

Next we present our microbenchmarks of function-level memoization and price prediction.

6.1.1 Function-level memoization. Figure 6 shows the latency of function calls with memoization, as compared to traditional compute. There are two sources of added latency from memoization: (1) the cost to check the cache for a precomputed result, and to return it if it exists; (2) the cost to store a computed result.

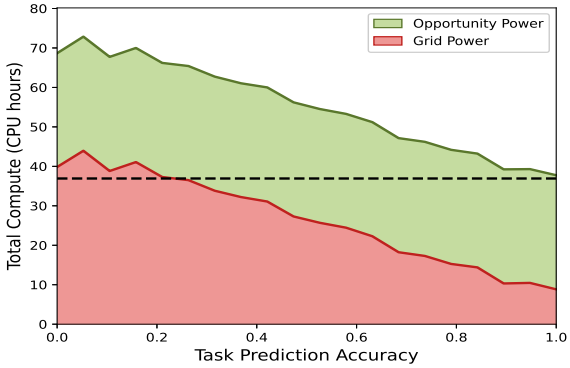


Fig. 5. While more accurate task prediction increases performance, energy can still be saved with relatively low (e.g. 30%) accuracies. Results are shown for machine cycles offloaded in simulation with varied task prediction accuracy and other parameters set based on microbenchmarking results. Simulation ran on MISO price data from 2019.

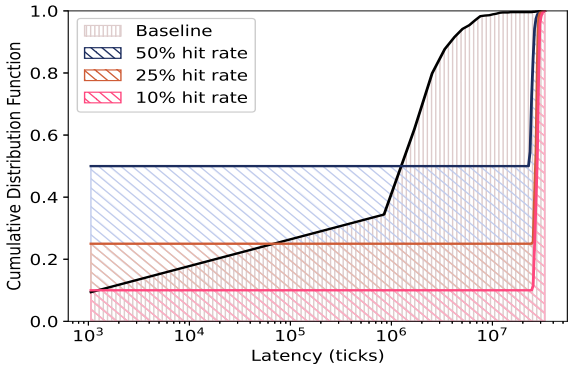


Fig. 6. Function call latency with function-level memoization at different hit rates for a 25ms job. The baseline (black line) refers to the uninstrumented function.

We measured these values for our proof-of-concept implementation. The results, summarized in Table 2, were used to parameterize our system-level simulations.

6.1.2 Price predictor. We measured the performance of the price predictor on both CAISO and MISO data. Table 3 summarizes the results.

6.1.3 Task predictor. Our naive task predictor had a top-1 accuracy prediction accuracy of 46% for predicting the next 10 function calls given the previous 10. This means that each individual function in the predicted sequence has a 46% chance of being correct.

6.2 System-level simulation

We built a system-level simulator to explore the performance of Information Batteries at scale. Our simulator takes as input a time-series of energy prices, and a set of parameters that define the performance of individual components of the system (Table 1).

The output of the simulator is a simulated 100-day run of an IB system, reporting 1) $\text{cycles}_{\text{avail}}$, the total amount of opportunity power that was theoretically available for compute, 2) $\text{cycles}_{\text{op}}$, the amount of opportunity power that was actually used for compute,

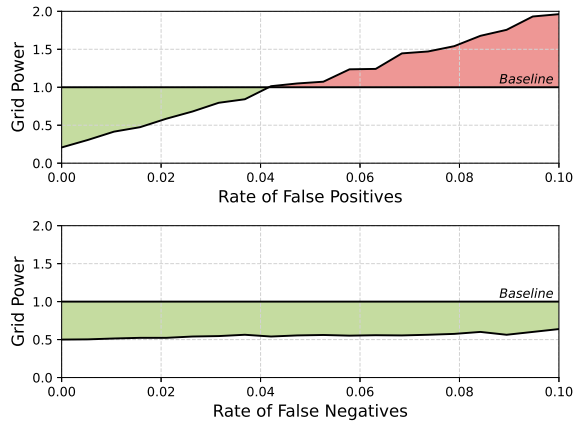


Fig. 7. The efficacy of the price predictor at different rates of false positives and false negatives. False positives—the mistaken identification of a period of opportunity power—can have a large impact on overall performance. False negatives—when the predictor fails to identify a period of opportunity power—have a significantly smaller effect. Efficacy here is defined as the amount of grid power used, compared to a baseline non-instrumented system. Information batteries are only worthwhile when this number is below the baseline (indicated with green fill).

and 3) $\text{cycles}_{\text{grid}}$, the amount of grid power that was used for compute. The run is considered successful if the amount of grid power used for compute is less than what would have otherwise been used in a traditional computing system.

For each 5-minute time interval, the simulator decides (based on the provided traces and the specified performance of the price predictor), whether or not to anticipate a period of negative-priced power. If negative-priced power is anticipated, it "schedules" jobs. It then records the amount of grid and opportunity power "used" in the interval. We summarize the logic of the simulator in Figure 3.

Our initial simulation results highlight the importance of accurate price prediction (Figure 7), and low cache latency (Figure 8), and relatively accurate task prediction (Figure 5). We use the results from microbenchmarking to realistically parameterize our system-level simulator according to the metrics described in Table 1.

We then evaluate the overall effectiveness of Information Batteries according to: 1) the efficiency of the system, in terms of the amount of processing offloaded from grid power to opportunity power, and 2) dollar cost relative to traditional batteries.

6.2.1 Energy savings. A key goal of the IB system is to make use of opportunity power that might otherwise be wasted. Thus, a central metric for any IB system is the amount of processing power offloaded from non-opportunity power to opportunity power.

We measure these savings in terms of the number of machine cycles offloaded. Figure 5 shows these results for a wind-dominant power profile (MISO). We parameterize the simulator as follows: (1) Memoization overhead of 1 ms (2) Task prediction accuracy of 50%; (3) For the price predictor, false positive and false negative rates of 0.1%. The length of tasks is varied.

Note that IBs always incur more machine cycles than traditional compute. This is to be expected, since the IB system must perform the same computational task with additional pre- and post-processing. However, in these scenarios, the grid power consumed by the IB system is less than that for traditional compute, since a large fraction of the compute was offloaded to opportunity power.

The overall energy savings of the IB system is the difference between the grid power consumed in the traditional compute scenario and the grid power consumed by the information batteries.

Cache Miss	0.87 μs
Cache Hit (Fetch)	2.2μs
Store	34 μs

Table 2. Average latency of cache hit, cache miss, and store.

	Mean-Absolute Error (Val)	Mean-Absolute Error (Train)
CAISO	0.1181	0.1402
MISO	0.0745	0.0614

Table 3. Performance of the price predictor model on validation and training data. The model performed best on MISO data, but the mean-absolute error was low for both datasets.

6.3 Cost of storage

In some scenarios, Information Batteries are more cost-effective than traditional energy storage systems. First we explain how to think about the storage capacity of an IB system. Then we compare potential IB systems with conventional grid-scale battery storage.

Unlike conventional batteries, Information Batteries have both a "charge rate" measured in Watts, corresponding to the power draw at which precomputation can be done by a given IB system (limited by data center capacity)—and a prediction time horizon that is not present in ordinary batteries. In some ways this makes an IB system incommensurable with conventional batteries.

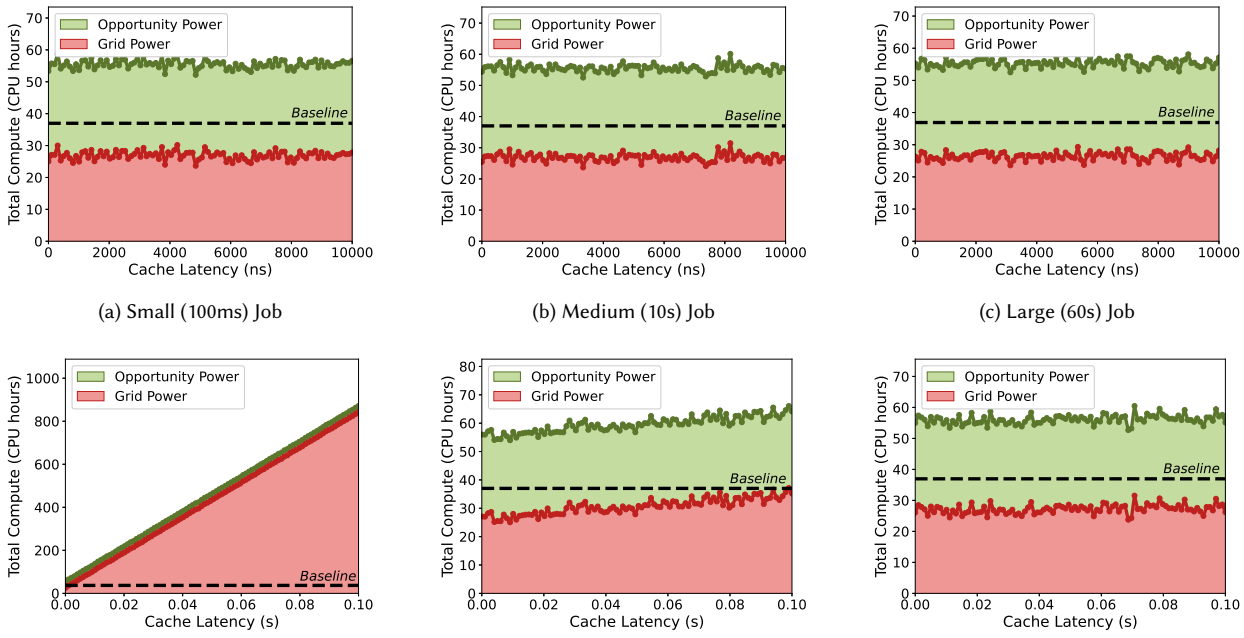
6.3.1 IB memoization overhead. Our function-level precomputation has an overhead of 34 μs for each put, 0.87 μs for each cache miss and 2.2 μs for each cache hit and retrieval on a 2.6 GHz Intel Core i7 CPU. This is extraordinarily efficient by virtue of its simplicity. For example, any job with just one second or longer run time would experience less than 1% overhead due to memoization.

6.3.2 Battery comparison. Consider for a moment a hypothetical IB system that has a one day time horizon, and can predict with perfect accuracy. The IB system can thus precompute a day's worth of tasks. Consider a hyper-scale data center that is 100 MW with this one-day prediction horizon; an IB system in this data center could store a monumental 2400MWh via precomputation alone.

It is rare for a data center to have full-day lookahead. Instead, we might more realistically have, on average, 90 minutes prediction ability with 90% accuracy.[6] With one-hour lookahead, such a data center could store 150 MWh, significantly more than most grid-scale battery-based storage projects. Given the negligible overhead of memoization, the key efficiency parameter is job prediction.

Using 90% as a canonical target efficiency, as it closely matches lithium-ion battery efficiencies, such an IB-enabled data center

[6]There is little public data on workloads and scheduling; our estimate here is based upon our experience working in such hyper-scale compute environments.



(a) Small (100ms) Job (b) Medium (10s) Job (c) Large (60s) Job

(d) Small (100ms) Job (e) Medium (10s) Job (f) Large (60s) Job

Fig. 8. Evaluating the effect of cache latency on energy savings. Results are presented as a stacked graph, with the red fill representing the amount of grid power used, and the green fill the amount of opportunity power. The system is only worthwhile if the amount of grid power used is below the baseline (e.g. if the red fill remains below the dotted line). Results presented for small (100ms), medium (10s), and large (60s) jobs for cache latencies of 0-10000 ns (top row) and 0-0.1 s (bottom row). Cache latency has a huge effect on performance, especially for shorter jobs.

would match the storage capacity of a 150,000 kWh lithium-ion storage array, which, at current lithium-ion prices of $356 / kWh [38] would cost $53.4 million.

7 CONCLUSION

In this work we have shown that Information Batteries have the potential to provide a cost-effective means to cope with growing renewable intermittency using large-scale computing infrastructure. Key to the IB approach is that it is not a general-purpose solution, but is likely to be effective for many common workloads.

7.1 Future directions

This paper merely introduces and explores one avenue of implementation and evaluation of Information Batteries. Much remains to be studied. In particular, we believe that there are three lines of worthwhile future research on this topic: improved prediction, improved integration into large systems, and support for the precomputation and recombination of fragments of computational tasks.

7.1.1 Prediction. The smart grid research community has done extensive work to improve prediction of price and power availability. In addition, the distributed systems community has done extensive work to characterize workloads in a wide range of settings. As we showed, as prediction accuracy improves, the efficiency of an IB system will improve, so there is substantial low-hanging fruit in incorporating state-of-the-art predictors.

7.1.2 Integration. While we frame our IB system prototype as an end-to-end system, any real-world deployment of this approach would necessarily omit the toy compute controllers we built for testing and instead integrate IB decision-making into an existing compute controller (e.g., a Kubernetes controller managing a whole data center). The criteria used by such data center operators to use the IB approach would necessarily be dependent upon their costs, business models, and the types of workloads they typically serve.

7.1.3 Computation. The efficiency of an IB system depends in large part on how well jobs can be accurately predicted and precomputed. But this precomputation need not be merely binary in nature—indicating whether a whole task should be precomputed or not—but instead can reflect a complex planning strategy that decomposes compute workloads into precomputable sub-units. There remains substantial work to be done on integrating ideas from related analyses performed in dramatically-different contexts, such as return-oriented programming, to identify which tasks can be meaningfully fragmented and then reassembled. In addition, efficient caching and retrieval of fragmented, precomputed results is challenging, as the greater complexity of fragmented precomputation, the more expensive retrieval is likely to be; this may require storage of program control-flow graphs along with compute fragments, so as to easily identify cached results that will meet the needs of new tasks. Finally, our exploration in this paper leveraged compiler support for program instrumentation, but in a real deployment it would be ideal to support unmodified program binaries.

REFERENCES

[1] Anders SG Andrae and Tomas Edler. 2015. On global electricity usage of communication technology: trends to 2030. *Challenges* 6, 1 (2015), 117–157.

[2] Solar Energy Industries Association. 2021. California Solar. https://www.seia.org/state-solar-policy/california-solar.

[3] Sean Barker, Aditya Mishra, David Irwin, Prashant Shenoy, and Jeannie Albrecht. 2012. Smartcap: Flattening peak electricity demand in smart homes. In *2012 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 67–75.

[4] Thomas Barnett, Shruti Jain, Usha Andra, and Taru Khurana. 2018. Cisco visual networking index (vni) complete forecast update, 2017–2022. *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation* (2018).

[5] Noman Bashir, Dong Chen, David Irwin, and Prashant Shenoy. 2019. Solar-TK: A Data-driven Toolkit for Solar PV Performance Modeling and Forecasting. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 456–466.

[6] R. H. Byrne, T. A. Nguyen, D. A. Copp, B. R. Chalamala, and I. Gyuk. 2018. Energy Management and Optimization Methods for Grid Energy Storage Systems. *IEEE Access* 6 (2018), 13231–13260.

[7] C2ES. 2020. Renewable Energy. https://www.c2es.org/content/renewable-energy/

[8] Fay Chang and Garth Gibson. 1999. Automatic I/O hint generation through speculative execution. USENIX.

[9] Andrew A Chien. 2018. *Characterizing Opportunity Power in the California Independent System Operator (CAISO) in Years 2015-2017.* Technical Report. Technical Report TR-2018-07. University of Chicago.

[10] Andrew A Chien, Fan Yang, and Chaojie Zhang. 2016. Characterizing curtailed and uneconomic renewable power in the mid-continent independent system operator. *arXiv preprint arXiv:1702.05403* (2016).

[11] Andrew A Chien, Chaojie Zhang, and Hai Duc Nguyen. 2019. Zero-carbon Cloud: Research Challenges for Datacenters as Supply-following Loads. (2019).

[12] Magnus Hallin Christian Legnitto, Christoph Herzog. [n.d.]. Crate Juniper. https://crates.io/crates/juniper.

[13] Santiago Correa, Lei Jiao, Aidas Jakubenas, Roby Moyano, Jesus Omana Iglesias, and Jay Taneja. 2020. Who's in Charge Here? Scheduling EV Charging on Dynamic Grids via Online Auctions with Soft Deadlines. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 41–49.

[14] A. T. da Costa, F. M. G. Franca, and E. M. C. Filho. 2000. The dynamic trace memoization reuse technique. In *Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00622)*. 92–99.

[15] Paul Denholm and Maureen Hand. 2011. Grid flexibility and storage required to achieve very high penetration of variable renewable electricity. *Energy Policy* 39, 3 (2011), 1817–1830.

[16] John S Denker. 1994. A review of adiabatic computing. In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*. IEEE, 94–97.

[17] Substrate DevHub. [n.d.]. Crate Substrate. https://crates.io/crates/pallet-product-registry.

[18] EIA. 2019. U.S. Energy Information Administration - EIA - Independent Statistics and Analysis. https://www.eia.gov/todayinenergy/detail.php?id=38053

[19] Íñigo Goiri, William Katsak, Kien Le, Thu D Nguyen, and Ricardo Bianchini. 2013. Parasol and greenswitch: Managing datacenters powered by renewable energy. *ACM SIGPLAN Notices* 48, 4 (2013), 51–64.

[20] A. Gonzalez, J. Tubella, and C. Molina. 1999. Trace-level reuse. In *Proceedings of the 1999 International Conference on Parallel Processing*. 30–37.

[21] Sriram Govindan, Anand Sivasubramaniam, and Bhuvan Urgaonkar. 2011. Benefits and limitations of tapping into stored energy for datacenters. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 341–351.

[22] Nidhi Hegde, Laurent Massoulié, Theodoros Salonidis, et al. 2011. Optimal control of residential energy storage under price fluctuations. *Energy* (2011).

[23] California ISO. [n.d.]. OASIS. http://oasis.caiso.com/mrioasis/logon.do.

[24] Changqing Ji, Yu Li, Wenming Qiu, Uchechukwu Awada, and Keqiu Li. 2012. Big data processing in cloud computing environments. In *2012 12th international symposium on pervasive systems, algorithms and networks*. IEEE, 17–23.

[25] Jian Huang and D. J. Lilja. 1999. Exploiting basic block value locality with block reuse. In *Proceedings Fifth International Symposium on High-Performance Computer Architecture*. 106–114.

[26] Héctor Ramón Jiménez. [n.d.]. Crate Iced. https://crates.io/crates/iced.

[27] Matthew P Johnson, Amotz Bar-Noy, Ou Liu, and Yi Feng. 2011. Energy peak shaving with local storage. *Sustainable Computing: Informatics and Systems* 1, 3 (2011), 177–188.

[28] Richard Kinley. 2017. Climate change after Paris: from turning point to transformation. *Climate Policy* 17, 1 (2017), 9–15. https://doi.org/10.1080/14693062.2016.1191009 arXiv:https://doi.org/10.1080/14693062.2016.1191009

[29] Steve Klabnik and Carol Nichols. [n.d.]. The Rust Programming Language. https://doc.rust-lang.org/book/.

[30] Zheng Liu and Clair J Sullivan. 2019. Prediction of weather induced background radiation fluctuation with recurrent neural networks. *Radiation Physics and Chemistry* 155 (2019), 275–280.

[31] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. 2013. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation* 70, 10 (2013), 770–791.

[32] Daud Mustafa Minhas, Raja Rehan Khalid, and Georg Frey. 2017. Load control for supply-demand balancing under renewable energy forecasting. In *2017 IEEE Second International Conference on DC Microgrids (ICDCM)*. IEEE, 365–370.

[33] Aditya Mishra, David Irwin, Prashant Shenoy, Jim Kurose, and Ting Zhu. 2012. Smartcharge: cutting the electricity bill in smart homes with energy storage. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*.

[34] Aditya Mishra, David Irwin, Prashant Shenoy, Jim Kurose, and Ting Zhu. 2013. Greencharge: Managing renewableenergy in smart buildings. *IEEE Journal on Selected Areas in Communications* 31, 7 (2013), 1281–1293.

[35] MISO. 2021. About MISO. https://www.misoenergy.org/about/.

[36] MISO. 2021. Markets and Operations. https://www.misoenergy.org/markets-and-operations/.

[37] Climate Change Mitigation. 2011. IPCC special report on renewable energy sources and climate change mitigation. *Renewable Energy* 20, 11 (2011).

[38] Kendall Mongird, Vilayanur Viswanathan, Jan Alam, Charlie Vartanian, Vincent Sprenkle, and Richard Baxter. 2020. 2020 Grid Energy Storage Technology Cost and Performance Assessment. *Energy* 2020 (2020).

[39] OpenAI. 2018. AI and Compute. https://openai.com/blog/ai-and-compute/.

[40] pickleDB rs. [n.d.]. Crate pickledb. https://docs.rs/pickledb/0.4.1/pickledb/index.html.

[41] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. 2009. Cutting the electric bill for internet-scale systems. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 123–134.

[42] Ana Radovanovic. 2020. Our data centers now work harder when the sun shines and wind blows. https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows

[43] Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. 2012. Return-oriented programming: Systems, languages, and applications. *ACM Transactions on Information and System Security (TISSEC)* 15, 1 (2012), 1–34.

[44] Rust. 2021. Guide to Rustc Development. https://rustc-dev-guide.rust-lang.org/.

[45] Hovav Shacham. 2007. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and communications security*. 552–561.

[46] Navin Sharma, Jeremy Gummeson, David Irwin, and Prashant Shenoy. 2010. Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems. In *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, 1–9.

[47] Navin Sharma, Pranshu Sharma, David Irwin, and Prashant Shenoy. 2011. Predicting solar generation from weather forecasts using machine learning. In *2011 IEEE international conference on smart grid communications (SmartGridComm)*. IEEE, 528–533.

[48] Vatsala Sharma, Pratima Walde, RK Saket, and Saad Mekhilef. 2020. Optimization of distributed generation size based on line sensitivity using transmission congestion cost. *International Transactions on Electrical Energy Systems* (2020), e12695.

[49] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. 2016. *United states data center energy usage report.* Technical Report. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

[50] Michael D Smith, Monica S Lam, and Mark A Horowitz. 1990. Boosting beyond static scheduling in a superscalar processor. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*. 344–354.

[51] Pirathayini Srikantha, Catherine Rosenberg, and Srinivasan Keshav. 2012. An analysis of peak demand reductions due to elasticity of domestic appliances. In *2012 Third International Conference on Future Systems: Where Energy, Computing and Communication Meet (e-Energy)*. IEEE, 1–10.

[52] Sun Sun, Min Dong, and Ben Liang. 2015. Distributed real-time power balancing in renewable-integrated power grids with storage and flexible loads. *IEEE Transactions on Smart Grid* 7, 5 (2015), 2337–2349.

[53] Arjun Suresh, Erven Rohou, and André Seznec. 2017. Compile-Time Function Memoization. In *Proceedings of the 26th International Conference on Compiler Construction* (Austin, TX, USA) *(CC 2017)*. Association for Computing Machinery, New York, NY, USA, 45–54. https://doi.org/10.1145/3033019.3033024

[54] Arjun Suresh, Bharath Narasimha Swamy, Erven Rohou, and André Seznec. 2015. Intercepting Functions for Memoization: A Case Study Using Transcendental Functions. *ACM Transactions on Architecture and Code Optimization* 12, 2 (June 2015), 18:18:1–18:18:23. https://doi.org/10.1145/2751559

[55] Jay Taneja, David Culler, and Prabal Dutta. 2010. Towards cooperative grids: Sensor/actuator networks for renewables integration. In *2010 First IEEE International Conference on Smart Grid Communications*. IEEE, 531–536.

[56] TensorFlow. [n.d.]. Text generation with an RNN. https://www.tensorflow.org/tutorials/text/text_generation.

[57] TensorFlow. [n.d.]. Time series forecasting. https://www.tensorflow.org/tutorials/structured_data/time_series.
[58] David Timmons, Jonathan M Harris, and Brian Roach. 2014. The economics of renewable energy. *Global Development And Environment Institute, Tufts University* 52 (2014), 1–52.
[59] Rahul Urgaonkar, Bhuvan Urgaonkar, Michael J Neely, and Anand Sivasubramaniam. 2011. Optimal power cost management using stored energy in data centers. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. 221–232.
[60] Ye Xu, David Irwin, and Prashant Shenoy. 2013. Incentivizing advanced load scheduling in smart homes. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. 1–8.
[61] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. 2011. Greenware: Greening cloud-scale data centers to maximize the use of renewable energy. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 143–164.