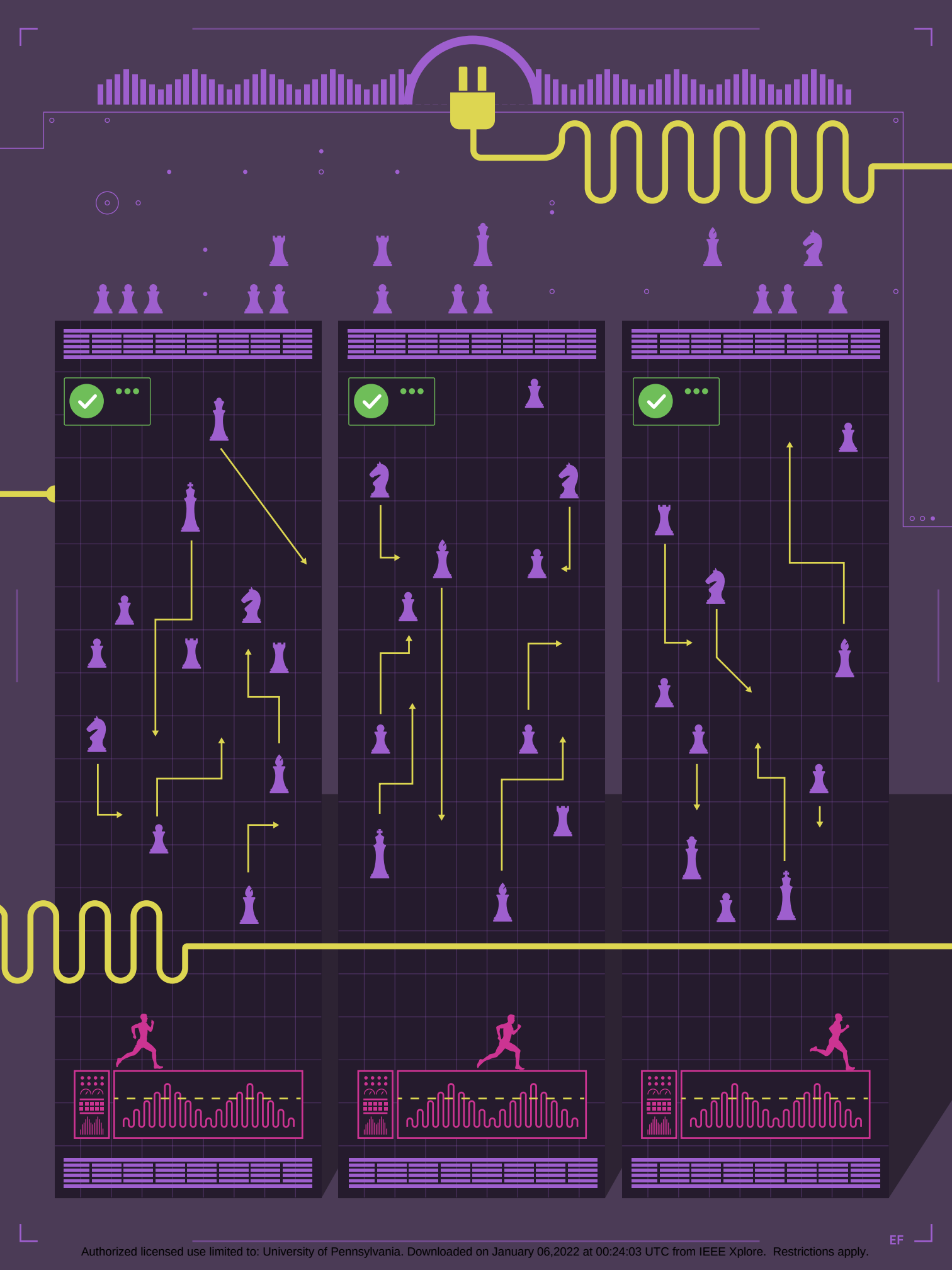
A WIN FOR GAME THEORY IN THE DATA CENTER

With the right strategy, colossal computing centers will waste less power

By Seyed Majid Zahedi & Benjamin C. Lee

WHEN YOU HEAR THE WORDS "data center" and "games," you probably think of massive multiplayer online games like *World of Warcraft*. But there's another kind of game going on in data centers, one meant to hog resources from the shared mass of computers and storage systems. • Even employees of Google, the company with perhaps the most massive data footprint, once played these games. When asked to submit a job's computing requirements, some employees inflated their requests for resources in order to reduce the amount of sharing they'd have to do with others. Interestingly, some other employees deflated their resource requests to pretend that their tasks could easily fit within any computer. Once their tasks were slipped into a machine, those operations would then use up all the resources available on it and squeeze out their colleagues' tasks. • Such trickery might seem a little comical, but it actually points to a real problem—inefficiency. • Globally, data centers consumed 205 billion kilowatt-hours of electricity in 2018. That's not much less than all of Australia used, and about 1 percent of the world total. A lot of that energy is wasted because servers are not used to their full capacity. An idle server dissipates as much as 50 percent of the power it consumes when running at its peak; as the server takes on work, its fixed power costs are amortized over that work. Because a user running a single task typically takes up only 20 to 30 percent of the server's

ILLUSTRATION BY Eric Frommelt



resources, multiple users must share the server to boost its utilization and consequently its energy efficiency. Sharing also reduces capital, operating, and infrastructure costs. Not everybody is rich enough to build their own data centers, after all.

To allocate shared resources, data centers deploy resource-management systems, which divide up available processor cores, memory capacity, and network resources according to users' needs and the system's own objectives. At first glance, this task should be straightforward because users often have complementary demands. But in truth, it's not. Sharing creates competition among users, as we saw with those crafty Googlers, and that can distort the use of resources.

So we have pursued a series of projects using game theory, the mathematical models that describe strategic interactions among rational decision makers, to manage the allocation of resources among self-interested users while maximizing data-center efficiency. In this situation, playing the game makes all the difference.

HELPING A GROUP OF RATIONAL and self-interested users share resources efficiently is not just a product of the big-data age. Economists have been doing it for decades. In economics, market mechanisms set prices for resources based on supply and demand. Indeed, many of these mechanisms are currently deployed in public data centers, such as Amazon EC2 and Microsoft Azure. There, the transfer of real money acts as a tool to align users' incentives (performance) with the provider's objectives (efficiency). However, there are many situations where the exchange of money is not useful.

Let's consider a simple example. Suppose that you are given a ticket to an opera on the day of your best friend's wedding, and you decide to give the ticket to someone who will best appreciate the event. So you run what's called a second-price auction: You ask your friends to bid for the ticket, stipulating that the winner pay you the amount of the second-highest bid. It has been mathematically proven that your friends have no incentives to misrepresent how much they value the opera ticket in this kind of auction.

If you do not want money or cannot make your friends pay you any, your options become very limited. If you ask your friends how much they would love to go the opera, nothing stops them from exaggerating their desire for the ticket. The opera ticket is just a simple example, but there are plenty of places—such as Google's private data centers or an academic computer cluster—where money either can't or shouldn't change hands to decide who gets what.

Game theory provides practical solutions for just such a problem, and indeed it has been adapted for use in both computer networks and computer systems. We drew inspiration from those two fields, but we also had to address their limitations. In computer networks, there has been much work in designing mechanisms to manage self-interested and uncoordinated routers to avoid congestion. But these models consider contention over only a single resource—network bandwidth. In data-center computer clusters and servers, there is a wide range of resources to fight over.

In computer systems, there's been a surge of interest in resource-allocation mechanisms that consider multiple resources, notably one called dominant resource fairness. However, this and similar work is restricted to performance models and to ratios of processors and memory that don't always reflect what goes on in a data center.

To come up with game theory models that would work in the data center, we delved into the details of hardware architecture, starting at the smallest level: the transistor.

Transistors were long made to dissipate ever less power as they scaled down in size, in part by lowering the operating voltage. By the mid-2000s, however, that trend, known as Denard Scaling, had broken down. As a result, for a fixed power budget, processors stopped getting faster at the rate to which we had become accustomed. A temporary solution was to put multiple processor cores on the same chip, so that the enormous number of transistors could still be cooled economically. However, it soon became apparent that you cannot turn on all the cores and run them at full speed for very long without melting the chip.

In 2012, computer architects proposed a workaround called computational sprinting. The concept was that processor cores could safely push past their power budget for short intervals called sprints. After a sprint, the processor has to cool down before the next sprint; otherwise the chip is destroyed. If done correctly, sprinting could make a system more responsive to changes in its workload. Computational sprinting was originally proposed for processors in mobile devices like smartphones, which must limit power usage both to conserve charge and to avoid burning the user. But sprinting soon found its way into data centers, which use the trick to cope with bursts of computational demand.

HERE'S WHERE THE PROBLEM ARISES. Suppose that self-interested users own sprinting-enabled servers, and those servers all share a power supply in a data center. Users could sprint to

SPRINTING'S UPS AND DOWNS

Computational sprinting allocates more processor cores and boosts their frequencies to speed algorithms such as Google's PageRank. However, it consumes more power and raises the processor temperature.

7×
NORMALIZED SPEEDUP

150%
NORMALIZED POWER

14 °C
AVERAGE TEMPERATURE INCREASE

increase the computational power of their processors, but if a large fraction of them sprint simultaneously, the power load will spike. The circuit breaker is then tripped. This forces the batteries in the uninterruptible power supply (UPS) to provide power while the system recovers. After such a power emergency, all the servers on that power supply are forced to operate on a nominal power budget—no sprinting allowed—while the batteries recharge.

This scenario is a version of the classic "tragedy of the commons," first identified by British economist William Forster Lloyd in an 1833 essay. He described the following situation: Suppose that cattle herders share a common parcel of land to graze their cows. If an individual herder puts more than the allotted number of cattle on the common, that herder could achieve marginal benefits. But if many herders do that, the overgrazing will damage the land, hurting everyone.

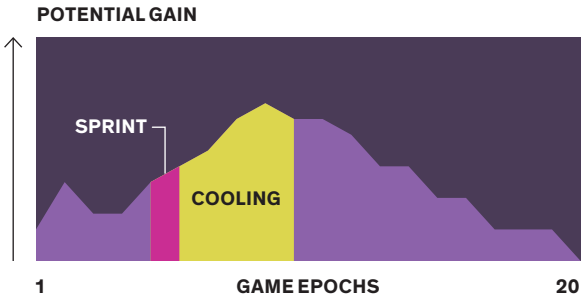
Together with Songchun Fan, then a Duke University doctoral candidate, we studied sprinting strategies as a tragedy of the commons. We built a model of the system that focused on the two main physical constraints. First, for a server processor, a sprint restricts future action by requiring the processor to wait while the chip dissipates heat. Second, for a server cluster, if the circuit breaker trips, then *all* the server processors must wait while the UPS batteries recharge.

We formulated a sprinting game in which users, in each round, could be in one of three states: active, cooling after a sprint, or recovering after a power emergency. In each epoch, or round of the game, a user's only decision is whether or not to sprint when their processor is active. Users want to optimize their sprinting to gain benefits, such as improved throughput or reduction in execution time. You should note that these benefits vary according to when the sprint happens. For instance, sprinting is more beneficial when demand is high.

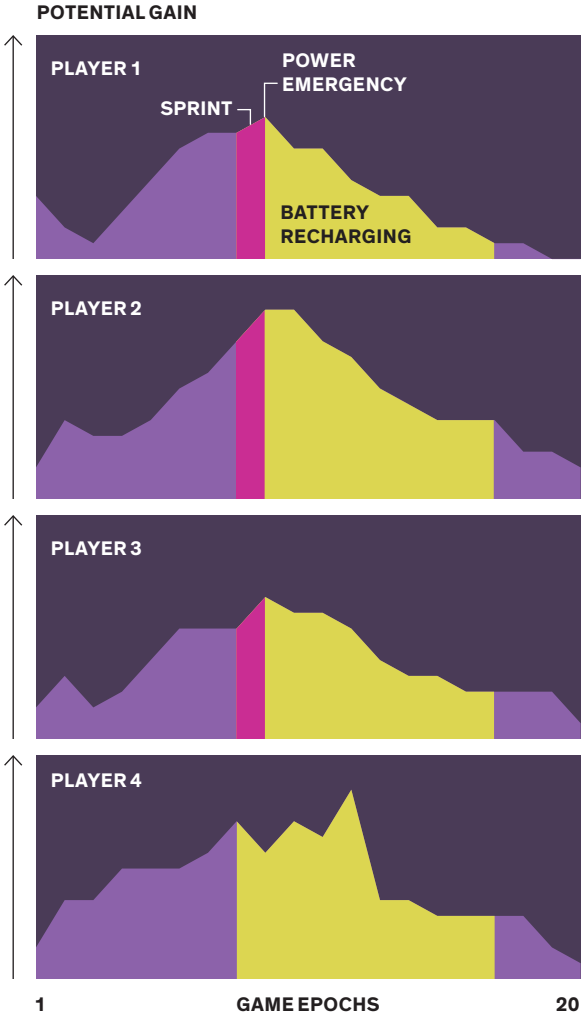
Consider a simple example. You are at round 5, and you know that if you sprint, you will gain 10 units of benefit. However, you'd have to let your processor cool down for a couple of rounds before you can sprint again. But now, say you sprint, and then it turns out that if you had instead waited for round 6 to sprint, you could have gained 20 units. Alternatively, suppose that you save your sprint for a future round instead of using it in round 5. But it turns out that all the other users decided to sprint at round 5, causing a power emergency that prevents you from sprinting for several rounds. Worse, by then your gains won't be nearly as high.

All users must make these kinds of decisions based on how much utility they gain and on other users' sprinting strategies. While it might be fun to play against a few users, making these decisions becomes intractable as the number of competitors grows to data-center scale. Fortunately, we found a way to optimize each user's strategy in large systems by using what's called mean field game analysis. This method avoids the complexity of scrutinizing individual competitors' strategies by instead describing their behavior as a population. Key to this statistical approach is the assumption that any individual user's actions do not change the average system behavior significantly. Because of that assumption, we can approximate

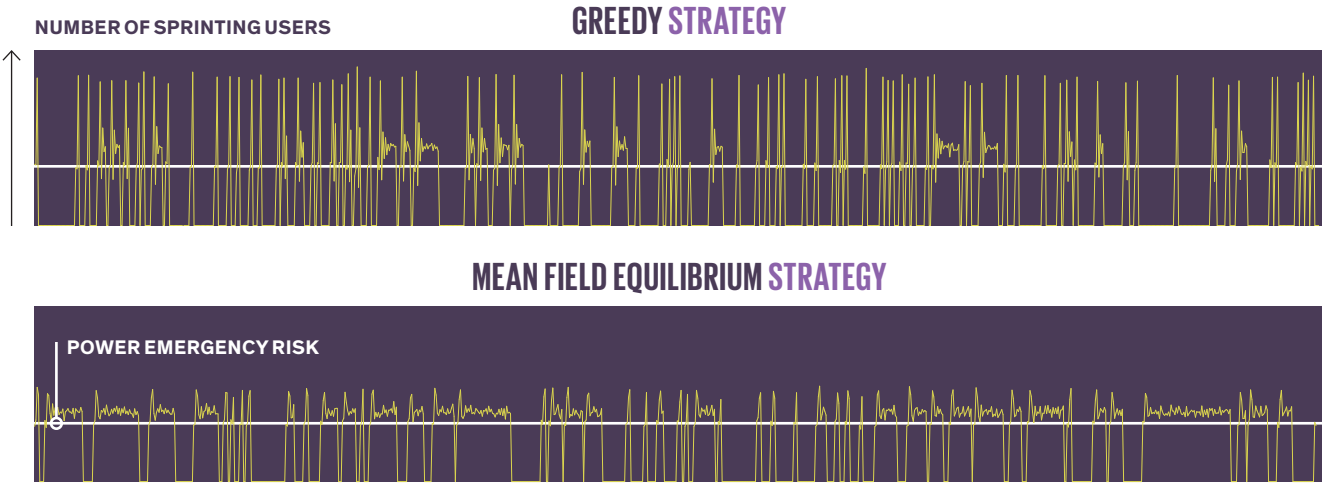
THE SPRINTING GAME



Players are users of a shared data center. If a player chooses to sprint during epoch 5 they achieve a certain gain, but they must wait several epochs while the processor cools before they can sprint again. Had they waited to sprint for a later round they would have accrued more units of gain.



If too many players choose to sprint at once, the increase in current can cause a power emergency. No one, not even player 4, who did not sprint, is allowed to sprint again until the computer cluster's uninterruptible power supply battery has recharged.



GREED ISN'T GOOD: Playing the sprinting game using the mean field equilibrium strategy gets more work done with fewer power emergencies than a "greedy" strategy.

the effect of all the other users on any given user with a single averaged effect.

It's kind of analogous to the way millions of commuters try to optimize their daily travel. An individual commuter, call her Alice, cannot possibly reason about every other person on the road. Instead she formulates some expectation about the population of commuters as a whole, their desired arrival times on a given day, and how their travel plans will contribute to congestion.

Mean field analysis allows us to find the "mean field equilibrium" of the sprinting game. Users optimize their responses to the population, and, in equilibrium, no user benefits by deviating from their best responses to the population.

In the traffic analogy, Alice optimizes her commute according to her understanding of the commuting population's average behavior. If that optimized plan does not produce the expected traffic pattern, she revises her expectations and rethinks her plan. With every commuter optimizing at once, over a few days, traffic converges to some recurring pattern and commuters' independent actions produce an equilibrium.

Using the mean field equilibrium, we formulated the optimal strategy for the sprinting game, which boils down to this: A user should sprint when the performance gains exceed a certain threshold, which varies depending on the user. We can compute this threshold using the data center's workloads and its physical characteristics.

When everybody operates with their optimal threshold at the mean field equilibrium, the system gets a number of benefits. First, the data center's power management can be distributed, as users implement their own strategies without having to request permission from a centralized manager to sprint. Such independence makes power control more responsive, saving energy. Users can modulate their processor's power draw in microseconds or less. That wouldn't be possible if they had to wait tens of milliseconds for permission requests and answers to wind their way across the data center's network. Second, the equilibrium gets more computing done, because users optimize strategies for timely sprints that reflect their own workload demands. And finally, a user's strategy becomes straightforward—sprinting whenever the gain exceeds a threshold. That's extremely easy to implement and trivial to execute.

THE SPRINTING POWER-MANAGEMENT PROJECT is just one in a series of data-center management systems we've been working on over the past five years. In each, we use key details of the hardware architecture and system to formulate the games. The results have led to practical management mechanisms that provide guarantees of acceptable system behavior when participants act selfishly. Such guarantees, we believe, will only encourage participation in shared systems and establish solid foundations for energy-efficient and scalable data centers.

Although we've managed to address the resource-allocation problem at the levels of server multiprocessors, server racks, and server clusters, putting them to use in large data centers will require more work. For one thing, you have to be able to generate a profile of the data center's performance. Data centers must therefore deploy the infrastructure necessary to monitor hardware activity, assess performance outcomes, and infer preferences for resources.

Most game theory solutions for such systems require the profiling stage to happen off-line. It might be less intrusive instead to construct online mechanisms that can start with some prior knowledge and then update their parameters during execution as characteristics become clearer. Online mechanisms might even improve the game as it's being played, using reinforcement learning or another form of artificial intelligence.

There's also the fact that in a data center, users may arrive and depart from the system at any time; jobs may enter and exit distinct phases of a computation; servers may fail and restart. All of these events require the reallocation of resources, yet these reallocations may disrupt computation throughout the system and require that data be shunted about, using up resources. Juggling all these changes while still keeping everyone playing fairly will surely require a lot more work, but we're confident that game theory will play a part. ■

POST YOUR COMMENTS AT spectrum.ieee.org/gametheory-apr2020