My research focuses on the intersection of computer systems, algorithmic game theory, and economic mechanism design. My interdisciplinary work has introduced computer architects to a rich body of knowledge in economics and game theory. My dissertation marks the beginning of a paradigm shift in computer architecture toward more robust, game-theoretic analysis of shared systems. As a result, my work has been acclaimed by the computer architecture community. The Computational Sprinting Game [6] received the best paper award at ASPLOS’16 and was distinguished as one of the IEEE Micro Top Pick Honorable Mentions, and REF [8] was recognized as one of the IEEE Micro’s Top Picks from Computer Architecture Conferences in 2014.

Resource Management: A Game Theoretic Approach

To improve efficiency and amortize cost over more computation, resource sharing has become vital in high performance computing systems [15]. In such systems, the conventional wisdom assumes that users have to share, regardless of the management policy. With a wide range of computing options available, this assumption does not seem to hold for today’s self-interested users. These users selfishly pursue their individual performance without regard for others or the system. And if they dislike management outcomes, they will withdraw from the shared system. If they decide to share, they will try to game the management system by misreporting their resource demands to improve their performance, perhaps at the expense of others in the system.

Users’ selfish behavior is not just a theoretical assumption. Previous work in systems literature has reported real-world examples of strategic behavior [12, 13, 14], making it a real challenge facing systems architects. To address this challenge and study strategic behavior of self-interested users, game theory is known to be an effective tool. Drawing on game theory, my research encourages new thinking in designing management platforms robust to strategic behavior. My main contributions are management platforms at three different levels in datacenter systems: server processors [8, 7, 10], server racks [6, 5], and server clusters [2, 1].

Multi-resource Allocation in Server Processors

In a shared server processor, computer architects encounter two challenges – sharing fairly and sharing multiple resources. To address these challenges, I have proposed Resource Elasticity Fairness (REF) [8, 7], a fair, multi-resource allocation mechanism that provably guarantees four fundamental game-theoretic properties. First, REF provides sharing incentives, ensuring that users perform no worse than under an equal division of resources. Second, REF provides envy-freeness, ensuring that each user prefers her own allocation over other users’ allocations. Third, REF ensures Pareto efficiency, providing an allocation in which the system cannot improve a user’s performance without harming another’s. Finally, REF is strategy-proof when the number of users in a shared system is large, ensuring that users cannot improve their performance by misreporting their resource demands.

These properties are guaranteed when software preferences for hardware can be modeled by Cobb-Douglas utility functions. The Cobb-Douglas function accurately describes hardware performance for two fundamental reasons. First, it captures diminishing marginal returns in performance, a prevalent concept in computer systems. Second, the Cobb-Douglas function captures substitution effects, which are also typical – a user might trade off-chip memory bandwidth for last-level cache capacity. Using cycle-accurate simulations for diverse application suites, I showed that Cobb-Douglas utility functions are well suited to modeling user utility for hardware resources.

In a related work with researchers at Qualcomm, we studied power management architectures for many-core server processors [10]. Power management faces significant challenges due to the end of Dennard scaling. Supply voltages are no longer scaling, which means chip multiprocessors must contend with higher power densities [11]. Higher power density affects current draw, which is limited by the number of pads that connect the chip to its package. The demand for pads grows as the number of cores on a single chip increases. As a result, cores will increasingly compete for current, and current limits may constrain cores well before thermal limits.

Managing current draw is challenging due to three main reasons. First, current draw changes quickly due to microarchitectural activity. Second, current demands vary across cores, but cores share the same power rail in most chip multiprocessors. Third, current allocations should reflect software-defined proportional shares specified for each core. To address these challenges we have proposed PACMAN, a distributed current manager that rapidly allocates and enforces proportional current budgets to the components in a System on Chip (SoC).

In PACMAN, a combination of local and global hardware units work together to set current and voltage levels that (a) meet the specified component performance requirements, (b) maximize system throughput and resource utility, and (c) keep the system current draw, average power, and temperature under the rated maximums. Using RTL-validated simulations, we demonstrated that PACMAN is able to distribute power proportionally, in a work-conserving manner, while remaining within specified bounds. PACMAN’s fine-grained allocation improves performance by up to 16% for some high priority workloads. Moreover, we showed that PACMAN increases weighted system performance by more than 10% for most combinations of workloads and QoS settings.
Power Management in Server Racks

In a datacenter rack, power supply is shared between servers. Most of today’s servers are capable of computational sprinting by supplying extra power for short durations to enhance their performance. Although sprints improve servers’ performance, uncoordinated sprints could overwhelm the rack’s power supply and risk power emergencies. To maximize performance gains and minimize risks, systems architects face hard management questions – which servers should sprint and when should they sprint? To address these questions, in a joint work with Fan [6], we have designed the computational sprinting game. In equilibrium, the game produce several desiderata – performance optimality of individual servers, system stability, and distributed sprinting management.

The sprinting architecture, which specifies the sprinting mechanism as well as power and cooling constraints, defines rules of the game. The game assumes that each server is controlled by a self-interested user who decides whether to sprint. Since simultaneous sprints could lead to power emergencies, users have to account for competitors’ decisions before making any sprinting decision. When all users optimize their sprinting strategies against each other, the game reaches its equilibrium. To find an equilibrium, users make initial assumptions about system conditions and optimize their strategies. Doing so, they affect those same system conditions. Eventually, system conditions and users’ strategies converge to a stationary distribution and the game reaches its equilibrium.

We have shown that users’ equilibrium strategy is a simple threshold strategy – sprinting whenever utility gain exceeds a threshold. To find and maintain an equilibrium, we have proposed a computational sprinting management framework. Offline, the framework finds each user’s sprinting threshold. Online, users decide whether to sprint by comparing a sprint’s utility gain against their threshold. The framework permits a distributed sprinting enforcement, because in equilibrium, users have no incentives to change their strategies.

In a collaboration with undergraduate researchers [5], we studied the trade-off between sprinting power consumption and its performance benefits. We modeled and analyzed total cost of ownership and efficiency of sprinting policies. We showed that although some of the policies incur less costs, by neglecting users’ incentives, they fail to maximize efficiency gains. In contrast, the computational sprinting game achieves much higher cost-efficiency by providing incentives while maximizing individual performance and maintaining system stability.

Processor Core Allocation in Server Clusters

In many private datacenters, users share a non-profit server cluster and its capital and operating costs. In such datacenters, a cluster manager must ensure users receive their entitlements, which specify the minimum share of resources each user should receive relative to others. For instance, in an academic cluster that combines servers purchased by researchers, entitlements may specify shares in proportion to researchers’ financial contributions.

Entitlements for processor cores in a datacenter differ from those in a server. Within a server, time on processor cores is a divisible resource that can be proportionally divided between users. The idealized datacenter provides a similar abstraction—a warehouse-scale machine with logically divisible pool of cores. However, cores are physically distributed across servers. This is challenging because users deploy different jobs on different servers, which means their demands for cores vary across servers. To address this challenge, a classical approach enforces proportional shares on each server separately, allocating each user her demand or entitlement, whichever is smaller. When entitlement exceeds demand, excess cores are redistributed among other users according to their entitlements. Although simple and widely used, this approach does not guarantee datacenter-wide entitlements.

To guarantee datacenter-wide entitlements, in work with Llull [2], we have designed the Amdahl bidding mechanism. The mechanism’s centerpiece is the Amdahl utility function, which is derived from Amdahl’s Law to model users’ valuations for each server’s cores. Users receive budgets in proportion to their entitlements and spend their budgets bidding for processor cores according to their Amdahl utility function. The market sets prices based on bids and users respond to prices until, in equilibrium, all cores are allocated and allocations are optimal. Informally, budgets satisfy entitlements while bids shift more resources to more parallelizable workloads.

Market allocations are competitive with performance-centric ones. First, allocations incentivize sharing as each user always receives her entitlement and sometimes receives more. Second, allocations are Pareto-efficient, which means no other allocation can benefit one user without harming another. Third, the market is strategy-proof for highly competitive systems, which means no user can benefit by misreporting utility from processors. Finally, the market has low overheads as we have devised closed-form equations to calculate market allocations.

The Amdahl bidding mechanism facilitates the trade in resources between users with static demands within a single management epoch. To facilitate the trade in resources between users with dynamic demands across multiple epochs, in a joint work with Freeman and Conitzer [1], we have proposed two novel mechanisms. First, the T-period mechanism satisfies strategy-proofness and sharing incentives but with low efficiency. Second, the token mechanism satisfies strategy-proofness and guarantees at least a 50% approximation of sharing incentives, which means users receive at least half the utility they would have received by not participating in the mechanism. Through simulations on data gathered from Google clusters, we have shown that the performance of the token mechanism is comparable to that of state-of-the-art mechanisms that do not guarantee our game-theoretic properties. Further, although the token mechanism only guarantees a 50% approximation of sharing incentives, in practice, users receive at least 98% of their sharing incentives guarantee.
Other Contributions

Task Colocation. Colocating multiple tasks on a server increases efficiency, but introduces contention for shared resources. Resource contention affects co-runners differently; among co-runners, some experience more performance loss from contention than others. When similar tasks suffer similar performance losses, a colocation is said to be fair. To find fair colocations, in collaboration with Llull and Fan [4], we designed “Cooper”, a novel management system that uses stable matching mechanisms to colocate users’ tasks. Our results show that under Cooper’s colocations, more contentious users incur larger penalties from colocation. Moreover, most users colocate with their “preferred” co-runners, and only a few users benefit from not participating in the mechanism.

Dynamic Social Choice. Most of my work in resource management focuses private resources (i.e., resources that could be allocated to only one user). In a collaboration with Freeman and Conitzer [3], we studied allocation mechanisms for public resources, where the allocation decision could benefit multiple users at the same time. For instance, cache capacity could be a public resource in systems where many applications could access the same cached data. We examined axiomatic characteristics of two greedy mechanisms for online Nash social welfare maximization problem. On data gathered from real system applications, we showed that both mechanisms significantly outperform mechanisms that are state-of-the-art with respect to theoretical guarantees.

Managing Heterogeneity. To allocate processors in a datacenter with “big” and “small” processors, I have devised a novel token mechanism that frames the allocation problem as a repeated game with discrete rounds [9]. At each round, users request big processors and spend a token if their request is granted. Spent tokens are then redistributed among users who do not receive a big processor. I have formulated the game dynamics and optimized users’ strategies to produce an equilibrium. In equilibrium, allocations balance performance and fairness, outperforming fair mechanisms and being competitive with a performance maximizing mechanism.

Future Research Directions

The trend in datacenter computing is toward large hardware platforms shared by self-interested users. I intend to continue my research on the intersection of systems and economics to address new challenges in these platforms.

Resource Management. In the short term, I intend to design platforms that manage multiple resource types over time, which includes two directions, one in theory and one in practice. For the former, I intend to build on my work on dynamic proportional sharing to design mechanisms for multiple resource types. For the later, I intend to build on my work on hardware controllers for many-core processors to coordinate the management of multiple resources. Another interesting future direction is addressing uncertainty in users’ demands. My work so far has heavily relied on applications’ profiled data to devise strategies and guarantee game-theoretic properties. Assuming that applications are entirely profiled and their behavior is fully characterized seems optimistic. Stochastic games, online stochastic optimization, and control theory are the tools that I plan to use next to find optimal strategies in the presence of uncertainty. Finally, I intend to design and prototype a next-generation automated management framework for large-scale computing that guarantees game-theoretic desiderata. To this end, I will pursue implementation and integration of new components to current open-source engines (e.g., Apache Spark). This would inevitably create new theoretical and practical research questions.

Learning in Multi-Agent Systems. In the long term, I plan to explore the intersection of machine learning and multi-agent systems, and its applications to distributed resource management. Multi-agent systems focus on the joint behavior of intelligent agents. In management platforms, these agents could be deployed on behalf of users’ applications and the cluster manager. Machine learning could then be deployed to automate the inductive process for these agents. This automation is attractive, as agents discover on their own how to interact. I plan to design management platforms with learning agents and study how learning shapes expected emergent behavior.

Privacy and Security. My research has so far focused on studying rational users whose utility only models their own performance. Malicious users, however, have a different utility model which is not performance-driven – they derive satisfaction from hurting other users. In the presence of such users, other users may have costs and preferences for privacy and security. Guarding against malicious behavior and designing trustworthy systems which provide privacy and security are interesting research challenges, especially in distributed and mobile platforms. Addressing these challenges requires approaches that cut across different fields of computer science. In recent years, there have been a growing interest in the intersection of mechanism design and privacy and security. I plan to explore this area and its applications to computer architecture and distributed systems.

Cooperative Mobile Computing. We often carry multiple mobile devices, from smart watches to phones and tablets. These devices provide redundant computing resources. Idle devices could collaborate when multiple users share a common sensing and computation goal. Such a system leverages endogenous incentives, which arise naturally from self-interested participants. To facilitate collaborative mobile computing, I plan to explore cooperative multi-agent learning, which provides foundations for agents who cooperate to solve joint tasks. I expect this research direction to further introduce new challenges concerning security and privacy.
References


