

Efficiently Approximating Optimal Behaviors in Real-world Systems with AI

Jiayi Jane Chen

As computer systems become more complex and their operating conditions more diverse and dynamic, approaching performance optimality through manual design becomes increasingly difficult. Doing so requires understanding how workloads, system components, and policies¹ interact, and adapting decisions to workloads, environments, and objectives that vary across deployments and over time. For example, CDN and block storage caches can require very different designs because of differences in access patterns, resource constraints, and performance objectives.

AI creates an opportunity to rethink the system design process. Learning can uncover relationships that are difficult to capture analytically, while AI-driven policy optimization can automate specialization. However, today's systems often lack the interfaces and mechanisms needed to expose the information that effective optimization requires. Isolated workload measurements obscure dependencies, delayed feedback masks environmental changes, and performance reveals only the effects of deployed policies. The central challenge is to identify and represent this latent information, and use it to efficiently approach system-wide optimality within practical resource and latency constraints.

My research designs system abstractions and frameworks that integrate generalizable AI-based components, grounded in theoretical analysis, to efficiently approximate system-wide optimality under changing conditions. My key insight is that **latent information should be a first-class object in system design**. My dissertation demonstrates how exposing latent structure in workloads, system dynamics, and policy performance enables richer system models and better-informed policies. My ongoing and future research builds on this foundation to investigate how systems should be organized when applications and controllers explicitly produce, exchange, and act on latent representations.

This effort is fundamentally interdisciplinary. *I helped shape this research direction as part of the founding proposal team for the NSF Expedition on Learning-Directed Operating Systems (LDOS)*. My approach combines systems expertise to abstract real-world problems, theoretical analysis to guide optimal design, and AI-based approximations to bridge the gap between theoretical assumptions and practical constraints. I build and evaluate systems that efficiently use these approximations to materialize gains in real-world systems, and use theory to characterize their benefits and limitations.

Broader Impact. My research has demonstrated real-world performance gains with open-source artifacts [4, 5] and started follow-on collaborations across academia and industry. Darwin achieves up to a $1.71 \times$ hit-rate improvement on real Akamai production traces. Building on UNUM, we have ongoing collaborations with Amazon's Vega OS team to extend its state-estimation approach to Wi-Fi environments and with Collins Aerospace to explore predictability in satellite networks. Both Darwin and Tardis have been integrated into the widely used libCacheSim [9] cache simulator.

Elevating Latent Information to a First-Class Citizen in System Design

Approximating optimal system behavior requires an accurate system model that captures all information that can improve policy decisions, search methods that use available information to narrow the policy space and efficiently identify near-optimal policies, and safeguards that prevent catastrophic behaviors. These requirements impose new challenges for both AI methods and system abstractions. The fundamental trade-off is between representational power and computational efficiency. Rich AI models can improve system modeling and policy search, but their cost often exceeds runtime control budgets. The challenge is to determine when and how to use these models outside the critical path to improve decisions, and how to adapt lightweight AI techniques to capture the information needed for effective system optimization.

My key observation is that execution histories and policy behavior contain latent information that can guide better decisions without requiring expensive models on the critical path. Rather than treating AI as a monolithic replacement for system logic, my research develops AI techniques to efficiently extract this latent information, system abstractions that expose it for control while guarding against catastrophic behavior, and theoretical analyses that characterize the resulting benefits and limitations.

¹In this statement, system policies refer to the decision logic that selects system actions based on available information, objectives, and constraints, such as which task to schedule, what to cache, or when to transmit.

For system modeling, I developed compact representations that recover structure missing from conventional system measurements. TARDIS [6] compresses entity activity histories into sketches that expose temporal relationships, which proves to preserve next-activity probability mass under models of interleaved activity. UNUM [7] learns latent network state from measurement histories and, guided by control theory, separates state estimation from control. For policy search, I incorporated existing policy structure and application semantics to guide efficient exploration toward better policies. In Darwin [8], I designed cross-policy predictors to estimate the performance of non-deployed policies, enabling online bandit selection with theoretical guarantees on exploration cost. I also contributed to AGS [3] interface that exposes application execution semantics and scheduling objectives to guide scheduling policy search. In addition, I contributed to the system formulation of online batching [1] and the empirical observations motivating its stochastic analysis when finite lookahead can approach optimality. I developed the Guardrails [2] interface for specifying system properties and corrective actions, giving OS developers explicit mechanisms to monitor learned policies and respond to unacceptable behavior.

Representing latent workload and system dynamics. System policies operate on partial, noisy observations of workloads and operating conditions. Measurements such as request arrivals, latency, throughput reveal individual or aggregated performance, but not the underlying relationships and dynamics that determine future behavior. Recovering this hidden structure can improve decisions, but maintaining full context from the detailed execution histories is too expensive for runtime control.

My work uses AI techniques to distill such latent factors into compact representations that provide richer context for policy decisions. I characterize the representations' value for control with theoretical analysis, and design general frameworks that enable various policies to use this context within practical resource and latency constraints.

TARDIS [6]: System policies often characterize entities individually, using statistics such as object access frequency or average workload resource usage. These statistics miss temporal relationships across entities: an access may signal demand for another object, while workloads may repeatedly experience overlapping demand peaks. TARDIS introduces a temporal relationship sketch that compresses activity histories into compact vectors whose similarity reflects recency-weighted temporal alignment. It's effective in recovering latent relationships and lightweight, requiring $O(\log N)$ state per entity and $O(\log N)$ computation per update or pairwise query, where N is the number of entities. TARDIS similarity is proved to captures nearly as much next-activity probability mass as an oracle that knows the latent process state. I develop a sketch interface and an asynchronous runtime that separates sketch maintenance from policy decisions. Integrating TARDIS into cache eviction, VM migration, and request load balancing demonstrates how the same relationship representation improves three distinct control tasks, reducing cache miss ratio by up to 12% and request tail latency by up to 50%.

UNUM @ NSDI'26 [7]: UNUM addresses a complementary information gap. Instantaneous and running-average measurements obscure latent factors such as application demand, path dynamics, and interactions with other controllers. Guided by the separation principle in control theory, I design UNUM to expose state estimation as a service. It introduces a shared representation layer that learns a high-quality state embedding from historical network observations, implicitly capturing latent dynamics in a shared state representation. I develop integration methods that expose the embeddings to downstream controllers, and an asynchronous system architecture that makes richer state estimation practical at runtime. Across real-world trace emulations, UNUM improves congestion control and adaptive bitrate selection rewards by 13.6% and 45.5%, respectively. By capturing the effects of controller interactions in shared state, UNUM also provides a foundation for implicit coordination toward system-wide optimality.

Guiding policy search with system and application knowledge. Learning effective system policies can require extensive exploration across a large decision space, and the resulting models may exceed runtime control budgets. My key insight is that systems already contain knowledge that can make policy search more efficient. Existing heuristics encode useful policy structure, while applications know their execution semantics, constraints, and objectives.

My work incorporates this knowledge into AI-based policy search to efficiently approach optimality. I develop methods that identify promising policy regions and exploit relationships among candidate decisions, theoretically characterize their benefits and limitations, and design frameworks that translate the resulting guidance into efficient online policies.

Darwin @ SIGCOMM’23 [8]: System has many heuristic policies that encode valuable system insight, but their effectiveness varies with workloads and optimization objectives. There is no one-size-fits-all policy. I developed Darwin to combine the adaptability of learning with the efficiency of heuristic control for CDN cache admission. Darwin identifies promising configurations of existing heuristics as workload-specialized experts offline, then selects among them online as traffic changes. To make this selection efficient, Darwin exploits a key insight: for the same problem, policies usually share a common structure, making their performance correlated. I introduced cross-expert prediction networks that use observations from the deployed policy to estimate the performance of alternatives. These predictions guide a neural-aided bandit algorithm, allowing observations from one expert to inform the evaluation of many candidates and reducing the need for direct exploration. This algorithm is proved to identify the best candidate policy with asymptotic expected stopping-time scaling independent of the number of policies to select from. Darwin framework supports multiple optimization objectives and improves object hit rate by up to 43% over evaluated baselines with minimal runtime overhead.

AGS [3]: Applications possess scheduling knowledge that general-purpose schedulers cannot readily use, yet translating that knowledge into effective policies requires accounting for uncertain runtimes and interactions among decisions. AGS exposes application scheduling knowledge through an interface for specifying events, available actions, and utility using lightweight symbolic execution semantics. Constraint solving derives hindsight-optimal schedules from execution traces, which initialize an online policy refined through evolutionary search to account for stochastic runtimes and modeling error. AGS reduces mean Age of Information by 47% relative to default scheduling in Autoware and matches a specialized learned scheduler’s average job completion time in DAG scheduling.

Characterizing information limits and safeguarding. Richer information and more efficient policy search do not alone ensure effective runtime decisions. Information’s value depends on workload dynamics and resource constraints, while learned policies with strong average performance can still violate system requirements. My work performs system-driven theoretical analysis of information limits, and introduces runtime safeguards that detect violations and trigger corrective actions.

Online batching with lookahead @ SIGMETRICS’27 [1]: Batching exposes a fundamental tension between serving requests immediately to reduce waiting and delaying service to form larger, more efficient batches. My empirical observations show that finite lookahead into future arrivals can bring online batching close to the offline optimum on real request traces. However, its benefits depend qualitatively on the arrival model. By incorporating insights from competitive analysis into queueing-theoretic steady-state bounds, we show that finite lookahead has inherent limitations under adversarial arrivals with bounded batch capacity, but narrows the optimality gap exponentially as the lookahead window grows under Poisson arrivals.

Guardrails @ HotOS’25 [2]: AI-based policies require explicit mechanisms to detect and respond to behavior that violates system expectations. I structured these requirements into a systematic, uniform abstraction of properties to monitor and corrective actions to take, and designed a declarative interface for specifying these properties, when to check them, and how to respond to violations.

Ongoing and Future Work: Rethinking System Design Around Latent Representations

My past work shows how to extract and use latent information to improve system models and policies within practical runtime budgets. Yet existing system boundaries restrict both the information available to system policies and the decision space to optimize, leaving a gap to system-wide optimality. For example, real-time data transmission needs to send fixed application data regardless of the network condition, resource managers may lack knowledge of application dependencies and intent, and geographically distributed controllers may act on delayed or inconsistent views. Optimizing across these boundaries requires richer information exchange, whose communication and processing costs can be prohibitive.

I view latent representations as *task-directed semantic compression*: compact encodings that preserve information relevant to particular objectives. These representations offer a way to make richer information exchange practical, creating opportunities for joint optimization across layers and components. I envision a systematic interface architecture through which applications and system policies explicitly produce, exchange, and act on these representations across the application-to-application data plane, the application-to-system control plane, and the control plane connecting system policies.

A common challenge is how to shape specialized latent representations that preserve the information needed for effective decisions at low communication and computation cost. I am investigating task-

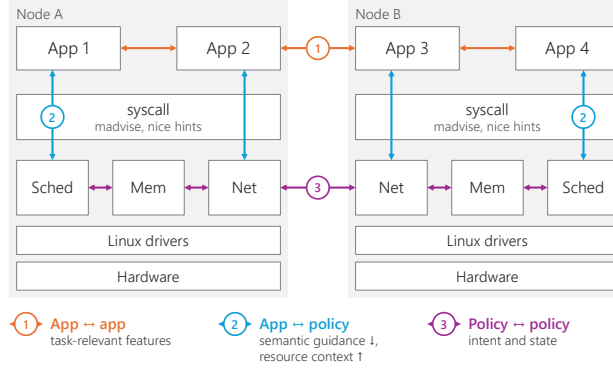


Figure 1: Latent representation as shared interface.

aware rate-distortion formulations to guide the design of latent representations that approach information-theoretic communication limits while bounding the loss in system performance. By co-designing representations, exchange protocols, and control mechanisms, I aim to further narrow the gap to system-wide optimality with limiting runtime overhead. I will establish formal guarantees for such system architectures on optimality and stability. At the same time, I will study how to support traditional system properties such as reliability, security, fairness as explicit constraints while supporting meaningful interpretability with latent representations.

Supporting real-time communication through latent representations. Today’s applications often transmit raw data streams through the Internet. For example, a robot with limited onboard compute may offload a large vision-based action model to an edge server, but continuously transmitting full camera observations can delay actions under bandwidth constraints. I plan to develop system support for such real-time communication through latent representations, where a lightweight onboard encoder could distill task-specific information to improve action quality and timeliness within practical bandwidth and compute budgets.

Realizing this requires utilizing representation learning, information theory and networked system design to jointly adapt representation generation, computation placement, and network transmission. I will explore task-aware rate-distortion formulations to characterize the communication needed to achieve a target level of decision quality under latency constraints. Supporting latent representations in the data plane also requires rethinking reliability and interpretability.

Rethinking the division-of-labor between application and systems. System interfaces expose only limited application knowledge through hints such as `madvise` and `nice`, leaving systems to infer how resource decisions affect application progress. With better AI capability, they can extract richer guidance from application semantics and execution histories. For example, application code or agentic workflow encode the task dependency and knowledge of execution needs. I plan to develop mechanisms and interfaces that expose this knowledge through compact latent representations to guide system policies, such as scheduling, memory management, network transmissions. Applications would take greater responsibility for shaping resource policies, while the system provides reusable mechanisms and arbitrates shared resources.

This division of responsibility requires richer representations that capture new semantics such as uncertainty or resource value, and together with system design that utilizes this guidance into effective policies. In addition, I will investigate enforceable contracts and fallback mechanisms that constrain application influence to preserve isolation, fairness, and resource limits even under inaccurate or adversarial guidance.

Enabling policy coordination. UNUM enables implicit coordination by sharing latent state representations across local controllers. Building on this foundation, I am exploring how controllers can account for the effects of one another’s decisions through two complementary directions.

Within memory management, I am examining severe performance failures caused by existing uncoordinated policies in Linux. I plan to use latent representations of current system state to estimate the costs and benefits of interacting policies, guiding when one policy should take priority. This would allow coordination to adapt to workload and resource conditions.

Across distributed controllers, coordination must also accommodate partial, delayed, and inconsistent views. I will investigate what information latent representations must preserve for effective coordination, and how update frequency and consistency requirements affect performance and stability. I plan to develop protocols that bound the effects of stale or conflicting information, connecting the communication cost of latent exchange to the achievable gap from joint optimality.

References

- [1] Dimitrios Christou, **Jiayi Chen**, Nihal Sharma, Aditya Akella, Shuchi Chawla, and Sanjay Shakkottai. Online batching with lookahead: Adversarial limits and stochastic gains. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2027.
- [2] Divyanshu Saxena*, **Jiayi Chen***, Sujay Yadalam, Yeonju Ro, Rohit Dwivedula, Eric H. Campbell, Aditya Akella, Christopher J. Rossbach, and Michael Swift. How i learned to stop worrying and love learned os policies. In *Proceedings of the 2025 Workshop on Hot Topics in Operating Systems*, pages 1–7, 2025. * Equal contribution.
- [3] Haoyu Li, **Jiayi Chen**, Yuning Wu, Aditya Akella, and Venkat Arun. Towards application-guided scheduling. In *submission*, 2027.
- [4] **Jiayi Chen**. Darwin repo. <https://github.com/Janecjy/Darwin>.
- [5] **Jiayi Chen**. UNUM repo. <https://github.com/ldos-project/UNUM>.
- [6] **Jiayi Chen**, Aryan Khatri, Dimitrios Christou, Ayaan Dhuka, Apia Okorafor, Haoyu Li, Nihal Sharma, Saarth Deshpande, Venkat Arun, Aditya Akella, Sanjay Shakkottai, and Shuchi Chawla. Tardis: Online temporal relationship sketch for systems control. In *submission*, 2027.
- [7] **Jiayi Chen**, Nihal Sharma, Debajit Chakraborty, Saurabh Agarwal, Jeffrey Zhou, Aditya Akella, and Sanjay Shakkottai. UNUM: A new framework for network control. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pages 1–18, 2026.
- [8] **Jiayi Chen**, Nihal Sharma, Tarannum Khan, Shu Liu, Brian Chang, Aditya Akella, Sanjay Shakkottai, and Ramesh K. Sitaraman. Darwin: Flexible learning-based cdn caching. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 981–999, 2023.
- [9] Juncheng Yang. libCacheSim repo. <https://github.com/lala11a/libCacheSim>.