

Deterministic Computation Models for Parallel Machines

By 2020, five years after you get your Ph.D., embedded processors will sport 4,096 cores, server CPUs will have 512 cores and desktop chips could use 128 cores. Universally, there will be a shift from existing sequential algorithms to parallel in domains spanning control systems, cryptography and process-intensive computation. But more importantly, there is a need to design and develop parallel algorithms and parallel computation architectures with deterministic operation. Given a large number of stochastic data-dependent input streams and a spatio-temporal variation of core utilization, the goal is to devise a class of algorithms and complementary hardware architectures for mission and safety-critical environments where unreliable software is not an option.

Phase I: Algorithm Design for Parallel Machines

A core focus will be on algorithm design where time and space utilization are first-class citizens. While classical algorithm evaluation with Big-O notation focus on asymptotic performance, we are interested in the *deterministic* and absolute computation requirements where processing adaptively scales to additional cores to maintain the timeliness in completing computation. Scalable multi-core algorithm design will need to incorporate new ideas including scheduling, cost encoding and composition. **Scheduling** of tasks in multi-core systems is important and essential to real-time distributed stream processing. Given non-deterministic (but bounded) input, algorithms will need to schedule millions of threads at runtime to satisfy time and space computation constraints. In addition to the control flow, algorithms will need to incorporate where and when computation blocks are executed. Algorithms will need to **encode** the time/space/quality cost within their design so they scale seamlessly between different hardware configurations without rework and adapt to different input and real-time requirements. Finally, for mixed-criticality systems, such as flight control and cabin control that run on the same hardware in aircrafts, **composition** algorithms are required to bound the complexity of computation at runtime to optimize functional requirements (maintain a control law) and para-functional requirements such as safety, reliability and fault tolerance.

Phase II: On-line and Adaptive Stochastic Optimization

Current approaches to optimization for machine learning and on-line model building from data does not scale to more than 10's of input streams. This is largely a problem with (a) the complexity of optimization algorithms and (b) the tradeoff between concurrency and determinism. We will focus on incremental and dynamic algorithms that combine both on-line data and model information (built from the previous k iterations). While traditional scheduling algorithms assume all processes are compliant to the central scheduler, we will investigate cases where processes have both varying degrees of non-compliance and parts of the system model that are more well-defined than the central scheduler. This is a new class of real-time algorithms that move from computer-computer execution to a more realistic and general cyber-physical communication and computation model.

Phase III: Hardware Architecture Design

After sufficiently developing a base of real-time parallel algorithms for stochastic inputs, we will take a bottom-up approach and design new hardware architecture to aid the algorithms. This will involve developing custom multi-core processors on programmable hardware. Our focus will be to showcase a Datacenter-on-Chip that is a generic architecture for a large class of data dependent real-time problems.

Phase IV: Algorithm Mapping to Application and Tools

Our overall goal is to design distributed parallel machines where processes and responsibilities move fluidly between cores and datacenters. We will map the algorithms and architecture to a spectrum of applications in control systems, time-based cryptography and safety-critical computation. The ultimate question we want to answer is that given 1M cores, what performance guarantees can we provide for these applications and how can we achieve them.

+Possible co-advised with Prof. Rajeev Alur, top-cited professor in CIS, focus on formal methods for Multi-core Computation

**Of course this is a thought exercise, I would expect you to define your own thesis.*