A Project Summary

A new generation of applications is becoming very important for high-performance computing, including collaborative design, interactive walkthroughs and large data visualization, and telepresence. They require tremendous resources including CPU, memory, storage, and audio/visual devices, and they have substantially different characteristics, performance goals and system interactions than traditional scientific applications. For example, they have extremely irregular and unpredictable data access needs and workload distributions, they interact more dynamically and with many more types of input/output sensors and devices, they involve dynamic user interaction and steering, and their goal is to deliver the best possible quality at a fixed output refresh rate rather than a solution of fixed quality in the minimum possible time. As computer architectures become more complex, it becomes increasingly difficult to develop such applications to achieve the desired performance. Three properties are critical: (i) high performance for rich interactive behavior, (ii) adaptability and isolation in all layers (i.e. the complexity and unpredictability demand that each layer of application or system software must adapt to the layers above and below it—through performance modeling and through runtime feedback and adaptation—and should try to shield the neighboring layers from each other’s complexity), and (iii) performance portability across component upgrades and across the different major types of platforms that may be used in such environments. Our goal is to develop the software building blocks, runtime systems and design methodologies to assist such application development.

To this end, this proposal includes four research tasks. The first is to develop methods for these complex and unpredictable applications and application-level libraries (e.g. a parallel rendering or audio/video library) to adapt to complex, multi-processor architectures and memory hierarchies. So far, such applications have been developed for particular platforms in ad hoc ways. This proposal proposes a software architecture that is adaptable in two ways: (i) it uses performance models for the software and hardware layers beneath it (including accelerators and I/O devices) to decompose, partition, and structure itself appropriately to begin with, and (ii) it responds to dynamic feedback from the layers above (the user or higher-level application) and below (the current system state) to maximize quality while guaranteeing interactive rates.

The second task is to develop methods for runtime system layers to (i) automatically choreograph computations and communication, shielding application/library software from some of the architectural complexity), and (ii) provide runtime feedback to the layers above to support adaptation. For (i), we will decompose programs into easy-to-program fine-grained computational components that provide hints about their data needs; our runtime system will then reorder the computations to interact well with complex memory and I/O hierarchies, hide communication latencies, and tolerate protocol overheads. For (ii), we will develop runtime instrumentation of system behavior and APIs for bidirectional information flow between layers.

The above system adaptation and shielding support will itself provide some performance portability across architectures. However, runtime system support will not be enough, and it is clear that application software must help. Our third task is to develop methods and generalizable guidelines to structure application or library programs so that they will perform well across the key types of parallel architectures and accelerators that will be used in these environments; that is, to achieve good performance portability. Parallel systems are no longer only custom-designed, but are also built with commodity systems and interconnects. Even shared address space systems (particularly attractive for these applications) have gone far beyond Symmetric MultiProcessor (SMP) design; Cache Coherent Non-Uniform Memory Access (CC-NUMA) architectures and software shared address space systems on clusters have become important platforms. Currently, ad hoc approaches are used to tune programs for each architecture. We will study how to structure a range of applications for performance portability in both major programming models, both through basic algorithmic, data access and communication structures as well as through run-time adaptability, and study tradeoffs between programming models on both types of scalable platforms for these complex applications.

The last task is to place system design and evaluation on a scientific footing, by developing benchmark suites and workload-driven evaluation methodologies for these emerging areas. All these aspects of the proposal will also be applied to the new generation of scientific and engineering applications, which share some key characteristics of the immersive applications and also provide a more familiar foundation.

The proposed research will be conducted on several infrastructures already in place at Princeton, including (i) a parallel immersive display wall system driven by a cluster of PCs, two other scalable PC and PC-SMP clusters using Myrinet built by the SHRIMP project (obtained via Intel donations and an NSF Research Infrastructure grant), and an NSF-funded 64-processor SGI Origin 2000 shared by three departments.