# Steven S. Lumetta—Research Agenda

Increasing hardware complexity has made application performance more difficult and more important to understand. Although compiler technology has made some progress in identifying and implementing optimizations, it has not kept pace with the rapid growth of peak processing power. The use of traditional languages accounts for much of the gap—efforts that tune language constructs to underlying performance models simplify the process of optimization, both for compilers and for programmers. For me, the most stunning example of this notion was the ease with which our parallel C dialect, Split-C, allowed us to obtain world-class application performance on MPP architectures like the CM-5 and the T3D.

My primary research interest lies in developing models of performance and a software environment that exposes those models, thereby enabling a programmer to reason about performance. Towards building such knowledge, I bring experience with a wide range of system components, including communication software, optimizing compilers, programming languages, development tools, and parallel algorithms. In my Master's work, for example, I developed a distributed, graphical debugger with integrated support for the Split-C language. My efforts significantly enhanced the programming environment for our CM-5 and later for our network of workstations (NOW).

The last few years have brought fruition to cluster-based computing, enabling the construction of scalable systems that simultaneously support both interactive and batch use by sequential and parallel jobs. A combination of engineering and economy is now pushing the basic computing unit in these clusters from a uniprocessor workstation to a symmetric multiprocessor, or SMP. My recent efforts have focused on these new SMP-based clusters, known as Clumps. Clumps further complicate the optimization process by logically extending the access latency hierarchy to include other processors' caches and memory in other SMP's. Applications capable of exploiting the tight coupling of resources within each SMP can obtain superior performance on a Clump. For some applications, however, naive use of a Clump degrades performance relative to a cluster of uniprocessors. Capturing the performance phenomena induced by the hierarchical nature of these architectures requires a sophisticated set of models.

In my dissertation work, I combine modeling, implementation, and measurement to develop a quantitative understanding of the key performance issues for Clumps. I have built runtime support software that allows an application to exploit the inherent advantages of tightly coupled resources. Avoiding the potential pitfalls of the architecture, often in the form of artificially correlated resource demands, requires further structure in the runtime system. I am currently investigating this problem. The system that emerges from my work will simplify the process of obtaining high performance on these machines.

In the next few years, I intend to build upon my thesis work to vertically integrate performance notions for cluster-based systems. I will generalize technology developed as part of the NOW project to construct a peer-to-peer software architecture that dynamically discovers and configures itself. When underlying resources or their availability changes, the system will notify applications through a well-defined interface. The same interface will serve to generate reports of persistent failures automatically. In creating this peer-to-peer system, I plan to restructure the models of resource discovery and allocation to provide a usable framework for understanding performance. Resources within the LAN will be virtualized—separated logically from physical hosts—so as to simplify migration and dynamic reconfiguration. The system will also improve performance predictability by isolating performance interactions for shared resources. Finally, the system design will include both debugging and performance interfaces to aid in the construction of effective tools.

To drive the construction of this new system, I plan to use it as the focus of a graduate-level class. Part of the appeal of an academic position lies in the exchange of ideas driven purely by intellectual value, in asking how our solutions evolve as aspects of a problem change underneath us. I feel strongly that scientific inquiry must at some point be free of the constraints of established techniques and commercial success. However, this freedom is at its best when tempered by the needs of real users. Hence I also intend to encourage computational scientists to use the system and to provide feedback as to the value of its components. I am also excited about the prospect of teaching undergraduate courses, particularly in the areas of concurrent programming and software development techniques, which I believe will be of increasing importance in the future.