Recent hardware trends suggest that within the next couple years, the average data analyst’s laptop will have an incredible amount of parallel hardware available—8 or more cores, each supporting hardware vectorized operations on 4 or more double-precision floating point values at a time (e.g. Intel’s SSE and AVX instruction sets or the experimental Larrabee hardware). At present R has only limited capabilities for taking advantage of this parallelism. For example, the `parallel` package, recently added to the standard distribution, provides the ability to execute code on multiple cores simultaneously; but it requires the user to explicitly parallelize their code, and, due to implementation overhead, is limited to relatively coarse-grained parallelism. Additionally, the current R execution model is not well matched to the performance details of coming hardware. For example, most well-written (vectorized) R code, as executed today, is bottlenecked by memory accesses and will see little or no performance improvements as hardware vector sizes increase.

Over the last two years we have developed Riposte, a new research runtime for the R language, which has allowed us to experiment with a wide range of novel performance designs. In its current form, Riposte uses a dual virtual machine design inspired by previous work in high-performance APL implementations. The first virtual machine, a carefully implemented threaded interpreter efficiently executes scalar and short vector R code. While this runs, a low-overhead tracing method dynamically extracts any long vector code which is run by a second virtual machine. This second VM uses an optimization technique called vector fusion to eliminate most memory accesses, removing the memory bottleneck and permitting us to profitably use hardware vector units. The VM then does just-in-time compilation and automatically executes the code across multiple cores. The result is a high-performance, parallel R runtime that efficiently executes standard R code without any manual parallelization by the user.

At a level relevant for performance-oriented R users, we will discuss current trends in hardware technology and performance constraints in the current R implementation. Building upon our experience developing Riposte, we will then describe possible paths toward a high-performance R.