significant amount of parallelism that can be exploited in games and virtual worlds. If the amount of concurrency that we can exploit is greater than the amount of latency that we might introduce, the overall performance of the game or virtual world will be better.

Betting on the Future

Our reliance on multithreading from multiple cores is essentially a bet on the way processors will evolve in the future. Currently servers are built with processors offering between 2 and 32 cores; we believe that the future of chip design will center around even more cores rather than on making any existing core run at a higher clock rate. When we began this project some years ago, this bet seemed far more speculative than it now appears. At that time, we often presented our designs as an exercise in "what if," saying that we were experimenting with an architecture that would be viable if the performance of chips became more a function of the number of threads supported than the clock speed of a single thread. This is one of the advantages of doing such a project in a research lab, where it is acceptable to take a much higher risk in your approach to a design as a way of exploring an area that might turn out to be commercially viable. Current trends in chip design make the decision to build an architecture centered on multithreading look far more prescient than it appeared at the time the decision was made.‡

Even if we can get only 50% of perfect concurrency, we could hit a performance break-even point if we can reduce the penalty of using persistent storage to between 2 and 16 times that of main memory. We believe we can do better in both the dimension of concurrency and in the dimension of reducing the difference between accessing the persistent state and keeping everything in memory. But much will depend on the usage patterns of those building upon the infrastructure (which, as we noted earlier, are difficult to discover).

Nor should we think of minimizing latency as the only goal of the infrastructure. By keeping all the server game or world objects in the Data Store, we minimize the amount of data that would be lost in the event of a server failure. Indeed, in most cases a server failure will be noticed only as a short increase in latency as the tasks (which are themselves persistent objects) are moved from the server that failed to an alternate server; no data should be lost. Some caching schemes might result in the loss of a few seconds of play, but even this case is far better than the current schemes used by online games and virtual worlds, where occasional snapshots are the main form of persistence. In such infrastructures, hours of game play might be lost if a server crashes at just the wrong time. As long as latencies are acceptable, the greater reliability of the persistence mechanism used by Darkstar can be an advantage for both the developers of the system built on the infrastructure and the users of that system.

‡ Showing, once again, that very little is as important as luck in the early stages of a design.