In our experience, even when we can get developers to talk about the loads placed on the server by their game or virtual world, they are often incorrect in their reports. This is not because they are attempting to maintain some commercial advantage by misreporting what their server actually does, but because they genuinely don't know themselves. There is very little instrumentation placed in game servers that would allow them to gather information on how the server is actually performing or what it is doing. The analysis of such servers is generally experiential at best. Programmers work on the server until it allows game play to be fun, which is achieved in an iterative manner rather than by doing careful measurements of the code itself. There is far more craft than science in these systems.

This is not to say that the servers backing such games and virtual worlds are shoddily constructed pieces of code or that they are badly built. Indeed, many of them are marvels of efficiency that demonstrate clever programming techniques and the advantages of one-time, special-purpose servers for highly demanding applications. However, the custom of building a new server for each game or world means that little knowledge of what is needed for those servers has developed, and there is no commonly accepted mechanism for comparing one infrastructure to another.

Parallelism and Latency

This lack of information about what is needed for acceptable performance in the server is of particular concern to the Darkstar team, as some of the core decisions that we have made fly in the face of the lore that has developed around how to get good performance from a game or virtual world server. Perhaps the most radical difference between the Darkstar architecture and common practice is the refusal in the Darkstar architecture to keep any significant information in the main memory of the server machine. The requirement that all data that lasts longer than a particular task be stored persistently in the Data Store is central to the functionality of the Darkstar infrastructure. It allows the infrastructure to detect concurrency problems, which in turn allows the system to hide those problems from the programmer while still allowing the server to exploit multicore architectures. It is also a key component to the overall scaling story, as it allows tasks to be moved from one machine to another to balance the load over a set of machines.

Storing the game state persistently at all times is heresy in the world of game and virtual world servers, where the worry over latency is paramount. The received wisdom when writing such servers is that only by keeping all of the information in main memory will the latency be kept small enough to allow the required response times. Snapshots of that state may be taken on occasion, but the need for interactive speeds means that such long-term operations must be done rarely and in the background. So it appears on the face of it that we have based our architecture on a premise that will keep that architecture from ever performing well enough to serve the needs of its intended audience.