The relationship between UML processes and the corresponding host processes for each mode follows from this. Figure 9.3 shows these relationships.

`tt` mode really only exists on x86 hosts. The x86_64 and S/390 ports were made after `skas0` mode was implemented, and they both use that rather than `tt` mode. Because of this, in the following discussion about `tt` mode, I will talk exclusively about x86. Also, the discussion about address space sizes and constraints on UML physical memory sizes are confined to x86, since this issue affects only 32-bit hosts.

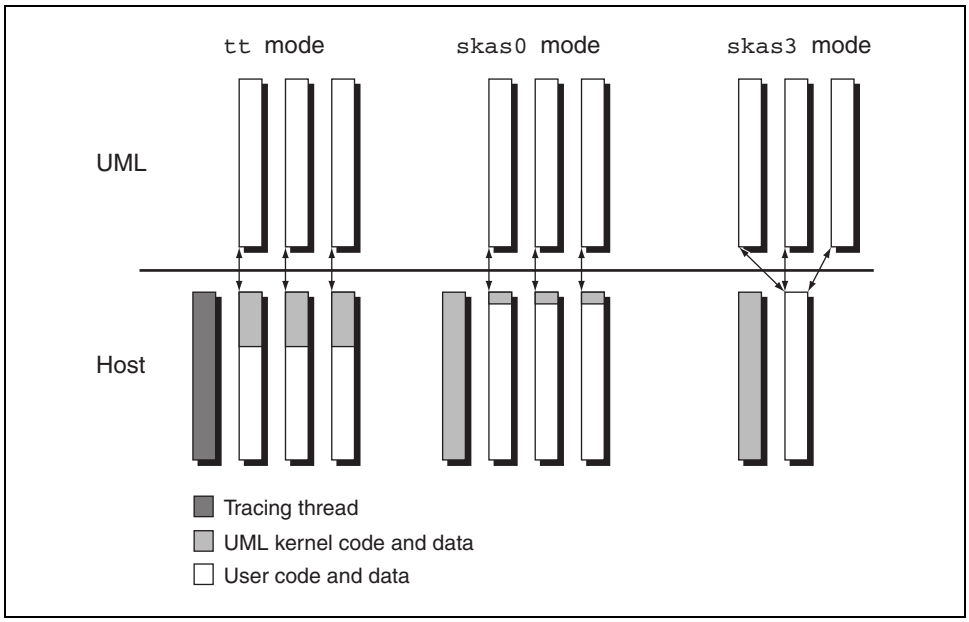

Figure 9.3 Comparison of the three UML execution modes. `tt` mode has a separate host thread (the tracing thread), which controls the execution of the other threads. Processes and threads within the UML instance have corresponding threads on the host. Each such host process has the UML kernel mapped into the top of its address space. In `skas3` mode, there is no separate tracing thread—this role is performed by the kernel thread. There is a single process on the host in which all UML processes run. `skas0` mode is a hybrid of `tt` mode and `skas3` mode. Like `skas3` mode, there is no tracing thread and there is a separate kernel thread in which the UML kernel lives. Like `tt` mode, each UML process has a corresponding host process.