Speculative execution as an OS service for applications

Many different systems have shown that speculative execution is a useful technique for increasing parallelism and hiding latency. These systems typically implement their own mechanism tailored to their application's specific needs.

This work examines how to design a shared operating system service that provides speculative execution for easier use by applications. Our design splits common tasks (checkpointing and rolling back state, buffering output, and tracking dependencies) into a shared mechanism controlled by application-specific policies.

In designing this system, we explore:
• What should the split be between mechanism and policy?
• In what ways can an application want to customize its execution?

Speculation policies for applications

Creation
Makes arbitrary regions of application code visible to the OS. Describes how to predict the results of those regions.

Output
Describes how speculative output should be handled. An output attempt could result in:
1. Buffering the output.
2. Tainting the destination.
3. Halting the execution.
4. Allowing the output.

Commit
Defines the equivalence class of results that can safely be committed. When results are outside this class, the app can try to compensate for the differences, or it can abort the speculation.

Application case studies

Predictive Application Launching in the Bash Shell

Firefox SSL Connections

PBFT-CS Fault Tolerance Protocol