Operating system support for application-specific speculation

Benjamin Wester, Peter M. Chen, and Jason Flinn
{bwester,pmchen,jflinn}@umich.edu
University of Michigan

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 exact 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-specifiec 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

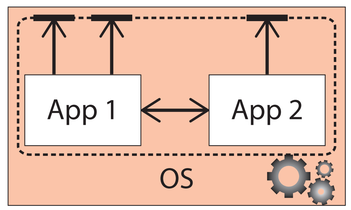
Defines the equivalence class of results that can safely be committed. When results are outside this class, the app can try to compensate for the differences, or it can abort the speculation.

How much should an application be able to customize its speculation?

No customization
Generic speculation
Simple to develop apps

Full customization
Flexible speculation
Complex to develop apps

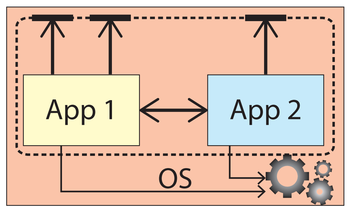
OS-Internal



Speculation is used by OS to **transparently** support existing applications.
+ Works on every app without changes
+ Wide scope: communicate via IPC, files

- Lacks semantic understanding of app
- Speculate only on **system call** operations

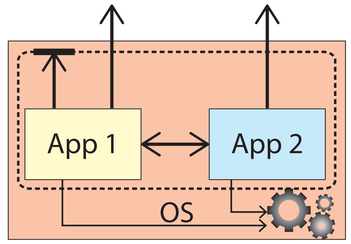
Invocable Service



Let applications invoke the OS to **define their own predictions** and speculative regions.
+ Speculate on **arbitrary app operations**
+ App can **reuse complex OS mechanism**

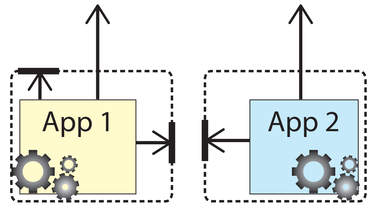
- Semantic information still limited
- Seculative external output **never allowed**
- Commit only on **identical** results

OS Mechanism with Application Policies



Give applications greater **control over their own safety** guarantees while speculative.
+ Speculate on arbitrary app operations
+ App can reuse complex OS mechanism
+ External output **allowd when safe**
+ Commit on **equivalent** results

Application Mechanism

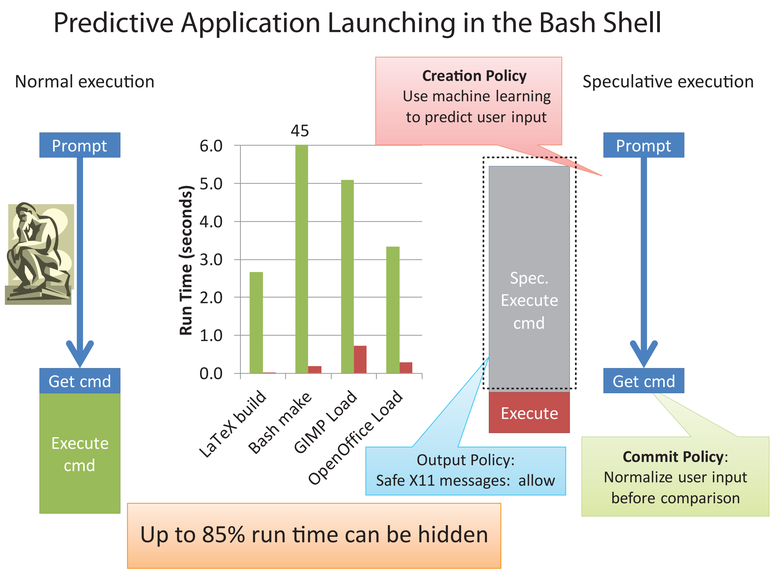


Application implements **its own mechanism** for speculative execution.
+ Everything can be customized

- **No reuse**: significant development
- **Scope limited** to application only

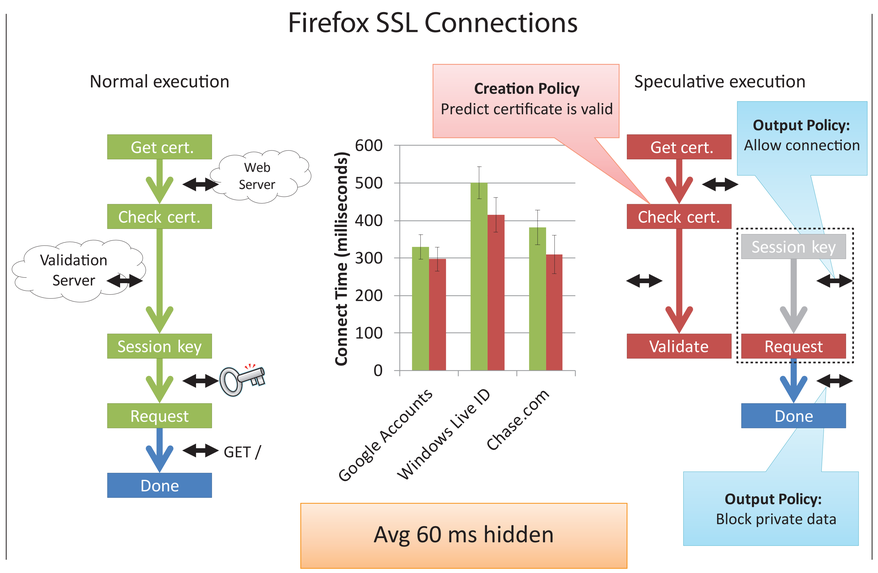
Application case studies

Predictive Application Launching in the Bash Shell



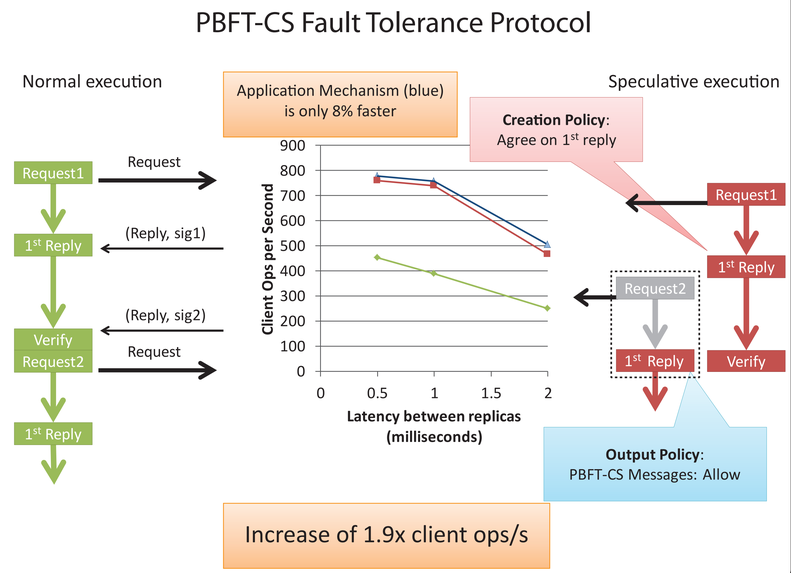
Up to 85% run time can be hidden

Firefox SSL Connections



Avg 60 ms hidden

PBFT-CS Fault Tolerance Protocol



Increase of 1.9x client ops/s