Contribution ID: **79** Type: **not specified**

Revisiting XSAVE: Lessons from 20 Years of Processor Context State Management

Friday, 20 September 2024 17:50 (20 minutes)

Prior to the XSAVE introduction, managing the processor's context state was handled on a per-feature basis. XSAVE normalized this by making state management independent of the CPU feature set. The XSAVE architecture has since evolved with optimizations, such as compacting the state format and tweaks for efficient reloads, resulting in a few XSAVE variants.

This monolithic approach to state management has accommodated the addition of around 10 features, expanding the overall state size to more than 10KB from the initial 1KB at the time of XSAVE's introduction about 20 years ago. Despite this growth, the unified approach has effectively prevented fragmentation and reduced the complexity that would arise from managing feature-specific state additions.

During the initial consideration of XSAVE, extensive discussions focused on the context format in the signal stack. It was emphasized that the new XSAVE format should be backward-compatible, and self-describing. Then, the XSAVE uncompacted format was adopted as part of user ABI, considering CPUID to universally provide the size and fixed offsets while trusting its proper extensions.

As the XSAVE architecture continued to embrace more feature states, some of these features were excluded in other CPU implementations. This uncovered a limitation in the uncompacted format, which proved inflexible in adapting to these dynamic changes. Unfortunately, the new compacted format cannot serve as a drop-in replacement for the user ABI, as it is incompatible with the uncompacted format.

This inflexibility has recently posed challenges in managing large states like AMX. Given this context, it is worthwhile to revisit the XSAVE story as a case study from both architectural and kernel perspectives. In the long run, it may be beneficial to discuss an alternative to the hardware format. Additionally, considering architectural mitigation could address the current limitations of the monolithic approach.

Primary author: BAE, Chang (Intel Corporation)

Presenter: BAE, Chang (Intel Corporation)

Session Classification: x86 MC

Track Classification: x86 Microconference