# Protección de procesadores SMT frente a ataques por contención: Equilibrio entre planificación de núcleos, aislamiento de recursos y seguridad en unidades de ejecución

Jan Marschatz Aguilar
Email: 2020026367@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Kun Kin Zheng Liang
Email: kzheng@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Abstract*—Simultaneous Multithreading (SMT) architectures offer higher throughput and resource utilization by allowing multiple threads to share a single physical core. However, shared resources in SMT processors also pose significant security risks, leading to a broad range of contention-based side-channel attacks. This paper provides a comparative analysis of three complementary mitigation strategies aimed at reducing or eliminating these vulnerabilities. The first, Partial-SMT (core scheduling), isolates sensitive processes by allocating them entire physical cores, thereby circumventing microarchitectural modifications. The second, SecSMT, proposes a comprehensive resource-partitioning scheme throughout the pipeline, ensuring fine-grained isolation of critical components such as caches, TLBs, and functional units. Finally, SMT-COP focuses on execution ports and functional units, preventing contention-based leaks by spatially or temporally partitioning these resources. Our detailed comparison includes the methods each strategy employs to evaluate both security and performance overhead, the trade-offs in complexity and adoption, and how these solutions manage to reclaim part of the SMT benefits while minimizing leakage. The discussion highlights the practical pathways to achieving robust SMT security in modern and future processor designs.**

***Palabras clave*—Multihilo simultáneo (SMT), núcleos físicos, canales de contención, particionamiento de recursos, ejecución especulativa, seguridad de microarquitectura**

## I. Introducción

La creciente proliferación de ataques de canal lateral basados en contención en sistemas multihilo simultáneos (SMT) ha posicionado este tema como un desafío crítico en el diseño de arquitecturas de computadores modernas. Estos ataques explotan la compartición de recursos microarquitecturales entre hilos co-ubicados en un mismo núcleo, permitiendo la fuga de información sensible mediante la observación de patrones de contención. Ante este escenario, la comunidad ha explorado estrategias tanto desde el nivel del sistema operativo, mediante políticas de gestión o restricción de compartición de núcleos, como desde el diseño de hardware, proponiendo soluciones que aíslan o particionan recursos críticos. Estas aproximaciones buscan mitigar riesgos sin comprometer drásticamente el rendimiento inherente al paralelismo a nivel de hilos, un pilar fundamental en procesadores contemporáneos.

En este contexto, este trabajo analiza y contrasta tres enfoques innovadores para abordar el problema: Partial-SMT, que propone una estrategia de programación a nivel de núcleos para limitar la compartición [1]; SecSMT, un método integral que aísla múltiples recursos del pipeline mediante particionamiento [2]; y SMT-COP, centrado en la protección específica de unidades de ejecución y puertos [3]. El objetivo principal es evaluar críticamente sus metodologías de evaluación, los mecanismos para suprimir canales de fuga, y sus resultados experimentales, identificando ventajas, limitaciones prácticas y oportunidades de mejora. Esta comparación busca ofrecer una visión holística que facilite la selección e implementación de soluciones adaptables a distintos entornos.

El análisis se estructura en torno a criterios sistemáticos, incluyendo el tipo de recursos protegidos (desde cachés y colas de instrucción hasta puertos de ejecución), el nivel de intervención requerido (modificaciones de software versus cambios en la microarquitectura), el impacto en el rendimiento medido como overhead, la complejidad de implementación en sistemas reales, y su aplicabilidad en contextos multiusuario con cargas heterogéneas. Estos aspectos permiten discernir equilibrios entre seguridad, eficiencia y practicidad, esenciales para diseños futuros.

Adicionalmente, se discuten los desafíos intrínsecos de los entornos SMT, donde componentes compartidos como unidades de ejecución especulativa, colas de instrucciones y cachés privadas amplían la superficie de ataque. Cada enfoque estudiado aborda estos riesgos mediante mecanismos distintos: aislamiento proactivo de recursos, detección de contención anómala o mitigación mediante particionamiento. Sin embargo, sus diferencias en granularidad y overhead subrayan la necesidad de soluciones adaptativas, capaces de equilibrar seguridad y rendimiento sin sacrificar la escalabilidad. Esta investigación no solo sintetiza avances recientes, sino que también orienta futuros desarrollos hacia arquitecturas resilientes y eficientes en escenarios de computación crítica.

## II. Metodologías de Evaluación Utilizadas

### A. Microbenchmarks específicos para saturar recursos

En primer lugar, muchos de los experimentos realizados en los tres trabajos se basan en microbenchmarks diseñados para ejercer presión sobre componentes específicos de la microarquitectura, como unidades de ejecución, puertos o buffers de instrucciones. Estos programas cortos buscan saturar el recurso concreto a fin de exponer posibles canales laterales y medir el potencial de fuga de información. Al centrarse en un solo bloque del pipeline, se puede analizar con precisión la latencia resultante cuando el atacante y la víctima comparten dicho recurso, lo que facilita cuantificar en bits por segundo la capacidad del canal encubierto. Asimismo, se obtiene una visión más clara de hasta qué punto las medidas de seguridad propuestas logran anular o reducir significativamente la contención entre hilos.

### B. Aplicaciones reales o macrobenchmarks

Para complementar el enfoque anterior, también se incluyen evaluaciones con aplicaciones que presentan comportamientos mucho más cercanos a entornos reales. En particular, los autores suelen recurrir a suites de rendimiento como SPEC CPU, así como a implementaciones de cifrado. La Figura 1 ejemplifica este enfoque mediante la partición espacial de recursos duplicados, un mecanismo clave para mitigar la contención en macrobenchmarks.

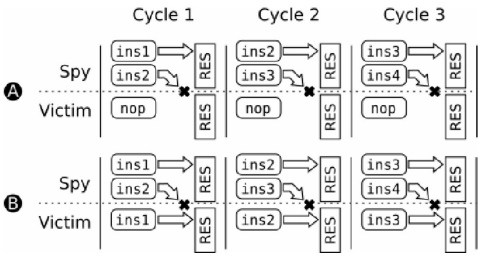


**Figura 1.** Partición espacial de recursos duplicados [3].

La inclusión de estos macrobenchmarks permite estimar el impacto que tienen las defensas propuestas en situaciones menos controladas, donde confluyen diversos patrones de acceso a memoria y a unidades funcionales. De este modo, los resultados obtenidos no se limitan únicamente a la reducción de la fuga de información, sino que también reflejan la experiencia de usuario y el coste adicional en términos de ciclos de reloj y de latencia global del sistema.

### C. Medidas de overhead y de "leakage" en bits/segundo

La comparación entre las distintas soluciones se basa, fundamentalmente, en dos ejes: por un lado, la pérdida de rendimiento (o overhead) que introduce cada mecanismo y, por otro, la capacidad de cada canal encubierto para transmitir datos. Este último se mide a menudo en bits por segundo, algo que proporciona un indicador cuantitativo de cuánta información puede filtrar el atacante por unidad de tiempo. Algunos estudios también recurren a factores de ralentización (slowdown) e incrementos de latencia para captar mejor el efecto de la defensa en el desempeño total del procesador. Al informar tanto la degradación de rendimiento como la merma en la fuga de datos, se establece un balance preciso entre seguridad y eficiencia.

### D. Modelos de amenazas contrastados

Por último, las metodologías de evaluación se ven influidas de forma decisiva por el modelo de amenazas que se asume en cada trabajo. Algunos autores examinan escenarios en los que el atacante está estrictamente confinado a un hilo colindante en el mismo núcleo y dispone de privilegios muy limitados. Otros, en cambio, consideran entornos más permisivos, donde el adversario puede orquestar tácticas sofisticadas para evadir contramedidas, o incluso explotar recursos compartidos adicionales (por ejemplo, branch predictors o TLB). Estos diferentes supuestos determinan qué tan exhaustivas deben ser las pruebas y permiten contextualizar la efectividad de las contramedidas en diversos grados de agresividad y sofisticación de la amenaza.

## III. Análisis Comparativo de los Enfoques

### A. Granularidad y Alcance del Aislamiento

La primera diferencia notable radica en el nivel en el que cada propuesta implementa su mecanismo de separación. En Partial-SMT (Core Scheduling) [1], la política de asilamiento se lleva a cabo a gran escala, a nivel de núcleos físicos completos, lo que impide directamente que la víctima y el atacante compartan simultáneamente un mismo núcleo. Esta medida, al ser muy efectiva en la supresión de canales de contención, deja a la aplicación en un entorno protegido sin necesidad de alterar la microarquitectura. Sin embargo, puede limitar la multiplexación de hilos de diferentes aplicaciones, lo cual se traduce en una menor utilización de recursos si no hay suficiente disponibilidad de núcleos libres.

SecSMT [2], por otro lado, adopta una visión integral de la microarquitectura e identifica cada subsistema compartido (caché de instrucciones, colas de carga/almacenamiento, TLB, puertos de ejecución, etc.) para imponer restricciones o particiones específicas en cada uno. Esto se traduce en un alto grado de control y, por ende, en una considerable capacidad de reducir fugas. No obstante, el alcance de este aislamiento requiere modificaciones complejas en el hardware para supervisar y repartir dinámicamente los recursos, algo que se vislumbra más aplicable en diseños futuros de procesadores.

Entre ambos extremos se sitúa SMT-COP [3], con un foco concreto en los puertos y las unidades de ejecución. Al restringir, bien de forma temporal o espacial, el acceso simultáneo a estos componentes, se logra proteger específicamente contra ataques muy recientes y efectivos (PortSmash o SMoTherSpectre), que basan su efectividad en la contención a nivel de ejecución. Si bien se reduce con ello la capacidad de fuga en la parte de ejecución del pipeline, se dejan parcialmente desprotegidos otros elementos como la TLB o las cachés, de modo que la cobertura en este sentido es menos amplia que en SecSMT. Esto posiciona a SMT-COP como una alternativa

más ligera de implementar, pero que no abarca todos los posibles canales laterales.

### B. Nivel de Intervención y Complejidad de Implementación

La complejidad de cada enfoque está íntimamente ligada al lugar en que se aplican las modificaciones. Partial-SMT [1] se ubica en un plano principalmente de software: mediante la extensión de las políticas de planificación y la utilización de librerías de usuario, se controla el uso exclusivo de un núcleo por parte de la aplicación sensible. Esta aproximación evita costos de rediseño de hardware, lo que la hace viable en entornos existentes (e.g., nubes o centros de datos) que no pueden esperar a nuevas generaciones de procesadores. Sin embargo, requiere una coordinación más estricta con el resto del sistema para no perjudicar demasiado la ocupación de los núcleos en tareas de menor seguridad.

SecSMT [2] implica la mayor complejidad, dado que sus particiones abarcan múltiples estructuras de la microarquitectura. Cada parte del pipeline comparte recursos de maneras diferentes, de modo que la partición debe adaptarse a la naturaleza de cada uno: algunos recursos pueden particionarse estáticamente, otros se multiplexan en el tiempo y algunos requieren supervisión permanente para evitar fugas indirectas. Estas modificaciones exigen un rediseño que involucra la unidad de planificación (scheduler), el frontend y el backend del procesador. Sin embargo, el gran atractivo de SecSMT es que, de implementarse, permitiría mantener SMT activo para la mayoría de aplicaciones, al tiempo que se reduce ostensiblemente la fuga de información.

En la propuesta de SMT-COP [3], la intervención de hardware se concentra en la lógica de emisión de instrucciones y en la asignación de puertos y unidades de ejecución. Esto conlleva cambios en el scheduler del procesador (particionado o multiplexado), pero no requiere necesariamente supervisar o repartir otros componentes como la TLB o la caché. Al ser más acotada, la modificación resulta menos costosa que la de SecSMT y puede suponer un escalón más realista en el camino hacia un SMT seguro: se gana protección en un ámbito crítico (la ejecución de instrucciones), aunque no se abordan todos los canales potenciales de fuga.

### C. Mecanismos de Activación Selectiva

Los tres enfoques contemplan, en mayor o menor medida, la posibilidad de activar o desactivar la protección según el contexto o las necesidades del sistema. En Partial-SMT, la planificación basada en núcleos se puede aplicar de manera estática solo a las aplicaciones marcadas como sensibles, de modo que el resto sigue compartiendo hilos lógicos con normalidad. Así, se evita un overhead universal e innecesario en cargas que no gestionan datos críticos.

SecSMT propone la adaptabilidad no solamente en qué aplicaciones se ven beneficiadas, sino también en el grado de partición o en la temporalidad de la misma. La idea es que el hardware pueda reconfigurar la asignación de recursos al detectar cambios en la demanda o potenciales eventos de riesgo. El reto radica en equilibrar el ritmo y la oportunidad de dicha reconfiguración, para que no impacte negativamente en el rendimiento global.

SMT-COP, por su parte, sugiere tres escenarios de activación: uno con la protección siempre activa, otro que se dispara al detectar manipulación de temporizadores (o intentos claros de medición) y un tercer modo que se invoca desde la aplicación. Este último caso permite, por ejemplo, encender la protección únicamente en la sección crítica de cifrado de una aplicación y desactivarla en el resto del programa. Dicho planteamiento concede flexibilidad y minimiza el overhead, pero a la vez deja en manos del software la responsabilidad de indicar con precisión cuándo se necesita la defensa.

### D. Impacto en Rendimiento y Overhead Global

Con respecto a la penalización en rendimiento, se observa un espectro de situaciones. Cuando se recurre a Partial-SMT [1], la degradación está vinculada a la inhabilitación de la coresidencia con hilos distintos: es decir, una vez que un núcleo se asigna de manera exclusiva, el sistema no puede explotar la multiplexación tan característica de SMT. Aunque esto elimina virtualmente la fuga de información dentro del núcleo, puede degradar la eficiencia total si hay muchos procesos compitiendo por pocos núcleos.

SecSMT [2], al abarcar todo el pipeline, puede llegar a introducir penalizaciones considerables si la partición o multiplexación de recursos no se planifica cuidadosamente. Sin embargo, el potencial de mantener un cierto grado de simultaneidad entre hilos es superior al de simplemente deshabilitar SMT o aislar totalmente un núcleo para un solo proceso. De allí que sus autores reporten cifras que, en el mejor de los casos, retienen un porcentaje importante del rendimiento de SMT a costa de una notable complejidad de implementación.

El caso de SMT-COP [3], se traduce en una degradación "moderada" (habitualmente en torno al 8-10 % en las publicaciones) cuando se analiza en escenarios que requieren protección continua. Esta merma, centrada en la limitación del uso de puertos o unidades de ejecución compartidas, se considera asumible en varios entornos críticos. Además, al ser posible activarlo selectivamente, el impacto real en cargas de trabajo mixtas puede resultar todavía menor.

### E. Cobertura de Vectores de Ataque y Robustez

Por último, es fundamental contrastar la efectividad de cada estrategia frente a la variedad de canales laterales que puede haber en un procesador SMT. En Partial-SMT, la idea de no compartir absolutamente nada dentro de un mismo núcleo cierra la puerta a cualquier canal de contención local (unidades de ejecución, caché L1 compartida, TLB). No obstante, persisten las interacciones entre núcleos a través de la caché LLC u otros subsistemas (aunque con un ancho de banda menor). Así, si un escenario no contempla ataques cross-core demasiado sofisticados, la protección es casi total. Como se observa en la Figura 2, el cronometraje de un hilo espía y una víctima bajo SMT-COP revela tiempos idénticos en ambas fases de ejecución, validando la neutralización de fugas por contención en unidades de ejecución.

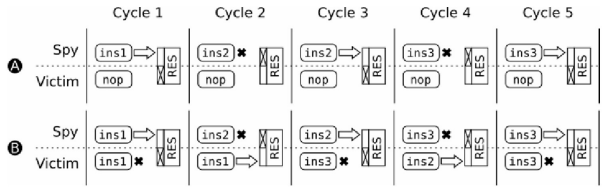


**Figura 2.** Cronometraje de un hilo espía y un hilo víctima ejecutándose bajo el esquema de particionamiento SMT-COP [3].

En cambio, SecSMT apunta a minimizar incluso la fuga que podría darse cuando, sí o sí, dos hilos ajenos deben compartir un núcleo. Se introduce un control muy fino en toda la ruta de ejecución, desde la captación de instrucciones hasta la fase de commit. Aunque esto brinda una protección amplia y reduce la ventana de ataques, el mantenimiento coordinado de tantas barreras de partición obliga a un rediseño significativo.

En la misma línea, SMT-COP se concentra en blindar las unidades de ejecución, asegurando que si se comparte el núcleo, la contención en puertos no sea ya un canal viable para filtrar información. Esta particularidad lo hace muy atractivo para frenar ataques muy divulgados y recientes que explotan port contention. Sin embargo, si la fuga se produce a través de la TLB o de un predictor de saltos, es preciso contar con otras defensas complementarias.

## IV. Resultados

La evaluación empírica de los tres enfoques analizados se centra en dos dimensiones críticas: la efectividad en la reducción de canales encubiertos y el impacto en el rendimiento del sistema. Por ejemplo, la Figura 3 compara el rendimiento de SPEC2006 bajo DDM y Partial-SMT, evidenciando cómo el aislamiento de núcleos afecta la eficiencia en cargas heterogéneas. Por otro lado, la Figura 4 contrasta el rendimiento de enfoques de pipeline Compartido, Participnado, Adaptativo y Asimétrico, destacando las ventajas de las particiones dinámicas. Esta dualidad permite discernir no solo la robustez de cada mecanismo de seguridad, sino también su viabilidad práctica en escenarios reales, donde el equilibrio entre protección y calidad de servicio es primordial.

En términos de mitigación de fugas, los experimentos con microbenchmarks revelan que, sin contramedidas, los atacantes pueden alcanzar tasas de extracción de información de cientos de kilobits por segundo (kbps). Al implementar las protecciones, todas las propuestas logran reducir drásticamente este riesgo. Partial-SMT, basado en programación exclusiva de núcleos, elimina por completo la contención intra-núcleo, anulando la fuga. Por su parte, SecSMT y SMT-COP, mediante particionamiento integral de recursos y restricción de puertos de ejecución, respectivamente, reducen el ancho de banda de los canales a niveles residuales o incluso insignificantes en ciertos casos. La Figura 5 demuestra visualmente cómo los canales encubiertos entre un hilo espía (T0) y un hilo troyano (T1) varían según el enfoque de pipeline utilizado, subrayando la eficacia del particionamiento adaptativo en SecSMT.

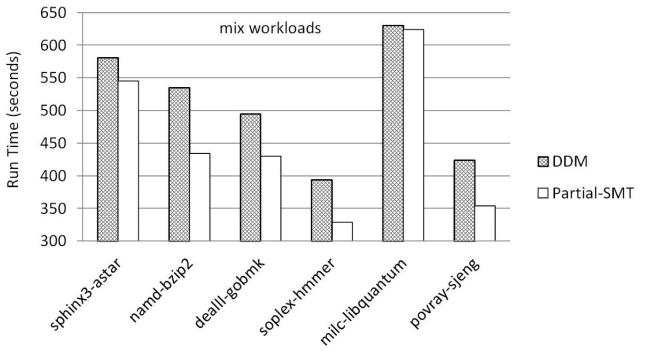


**Figura 3.** Rendimiento de los puntos de referencia SPEC2006 con DDM y Partial-SMT [1].

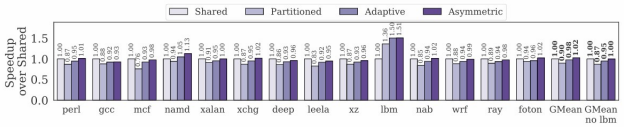


**Figura 4.** Comparación del rendimiento entre los enfoques de pipeline Compartido, Particionado, Adaptativo y Asimétrico [2].

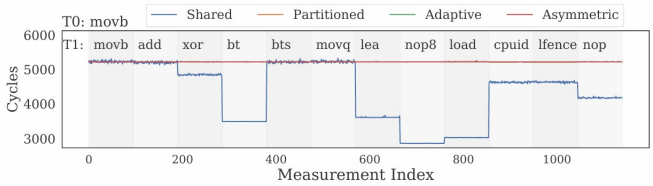


**Figura 5.** Canales encubiertos entre un hilo espía (T0) y un hilo troyano (T1) bajo enfoques de pipeline Compartido, Particionado, Adaptativo y Asimétrico [2].

No obstante, las diferencias emergen al analizar el overhead en cargas de trabajo reales. Partial-SMT presenta un impacto variable: en sistemas con alta concurrencia y núcleos limitados, el bloqueo exclusivo de núcleos para aplicaciones críticas penaliza el rendimiento, mientras que en entornos con pocas tareas sensibles, el costo es mínimo al aprovecharse el paralelismo SMT en otras cargas. SecSMT, al aislar recursos globalmente, sacrifica parte de la eficiencia inherente a SMT, aunque mantiene cierto grado de simultaneidad, posicionándose como un equilibrio entre la desactivación total de Hyper-Threading y una compartición insegura. En contraste, SMT-COP exhibe un overhead moderado (8-10 % en pruebas con suites como SPEC CPU o cifrado), atribuible a su enfoque focalizado en restringir unidades de ejecución, lo que lo hace viable para aplicaciones que priorizan seguridad sin comprometer excesivamente la velocidad. Este comportamiento se ilustra en la Figura 6, que muestra el overhead de rendimiento de SMT-COP en benchmarks de SPEC2006, confirmando su degradación moderada (8-10 %).

Un hallazgo transversal es la eficacia de las técnicas de activación selectiva de defensas. Los tres enfoques demuestran

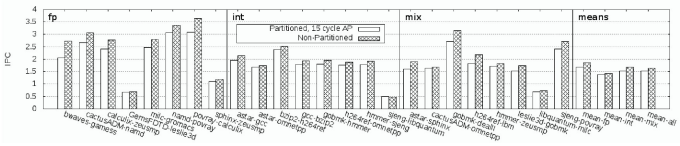


**Figura 6.** Sobrecarga de rendimiento de SMT-COP en los benchmarks de SPEC 2006 [3].

que habilitar protecciones únicamente en secciones críticas del código reduce el costo operativo. Esto sugiere que, con una delimitación precisa de los períodos sensibles, es posible alternar entre modos de alta seguridad y rendimiento optimizado, minimizando ventanas de exposición sin penalizar la productividad general. La Figura 7 ejemplifica este principio: la parte superior muestra aplicaciones ejecutándose con DDM, mientras la inferior ilustra cómo las aplicaciones críticas operan bajo Partial-SMT, evitando la compartición de núcleos durante fases sensibles.

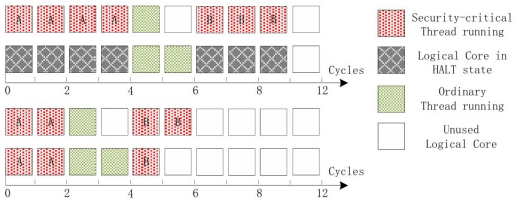


**Figura 7.** Parte superior: aplicaciones ejecutándose con DDM. Parte inferior: aplicaciones críticas de seguridad ejectándose con Partial-SMT [1].

En síntesis, los resultados revelan un compromiso inherente entre seguridad, rendimiento y complejidad. Para entornos con pocas aplicaciones críticas, Partial-SMT ofrece una solución pragmática y de bajo costo. En sistemas que requieren protección amplia sin renunciar al paralelismo SMT, SecSMT y SMT-COP representan alternativas complementarias: el primero prioriza un blindaje integral mediante particionamiento de recursos, mientras que el segundo mitiga ataques específicos con menor impacto en rendimiento. Esta diversidad de enfoques subraya la necesidad de adaptar las soluciones a las demandas particulares de cada escenario, donde la selección dependerá de la criticidad de los datos, la carga de trabajo y los recursos disponibles.

## V. Conclusiones

La comparación de los enfoques Core Scheduling, SecSMT y SMT-COP revela estrategias diferenciadas para mitigar fugas de información en arquitecturas SMT, cada una con ventajas y compromisos específicos. Core Scheduling emerge como una solución práctica a nivel de sistema operativo, al aislar aplicaciones no confiables en núcleos dedicados, eliminando la contención intra-núcleo a costa de limitar el paralelismo. En contraste, SecSMT propone un paradigma de hardware más ambicioso, particionando recursos del pipeline para ofrecer protección integral, lo que lo posiciona como un referente para futuros diseños que busquen preservar las ventajas de SMT sin exponer canales encubiertos. Por su parte, SMT-COP aborda vulnerabilidades concretas, como los ataques PortSmash o SMoTherSpectre, mediante restricciones escalables en unidades de ejecución, equilibrando eficiencia y seguridad con un overhead moderado.

En cuanto a su aplicabilidad, entornos virtualizados o en la nube podrían implementar Core Scheduling de inmediato a través de hipervisores para aislar máquinas virtuales críticas. Para diseñadores de hardware, SecSMT y SMT-COP ofrecen inspiración hacia procesadores con particiones más granulares, adaptadas a demandas de seguridad emergentes. Es previsible que sistemas de alta seguridad adopten enfoques híbridos: el sistema operativo aislaría aplicaciones sensibles, mientras el hardware aplicaría particiones dinámicas en casos generales, combinando rendimiento y protección.

No obstante, persisten retos significativos. La integración inteligente de mecanismos de software y hardware, como políticas de activación on-demand, requiere investigación para optimizar respuestas ante amenazas en tiempo real. Además, es crítico evaluar el impacto energético de las particiones de recursos, especialmente en microarquitecturas con mayor paralelismo, como cuando son de 4 hilos por núcleo. Paralelamente, el desarrollo de pruebas que combinen contención en ejecución con otras vulnerabilidades, como predicción de saltos, permitirá validar la robustez de estas soluciones frente a ataques multifacéticos.

En definitiva, la contención en SMT representa un desafío complejo, sin soluciones únicas. Los enfoques analizados demuestran que la elección entre ellos dependerá de factores como la criticidad de los datos, la carga de trabajo y los recursos disponibles. El futuro cercano probablemente testee arquitecturas adaptativas, donde la sinergia entre capas de software y hardware defina un nuevo estándar en el equilibrio entre seguridad, rendimiento y escalabilidad.

## Referencias

[1] X. Wu, Y. He, Q. Zhou, H. Ma, L. He, W. Wang, and L. Chen, “Partial-SMT: Core-Scheduling Protection Against SMT Contention-Based Attacks,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 378–385, IEEE, 2020.

[2] M. Taram, X. Ren, A. Venkat, and D. Tullsen, “SecSMT: Securing SMT Processors against Contention-Based Covert Channels,” in *31st USENIX Security Symposium (USENIX Security 22)*, (Boston, MA), pp. 3165–3181, USENIX Association, 2022.

[3] D. Townley and D. Ponomarev, “SMT-COP: Defeating Side-Channel Attacks on Execution Units in SMT Processors,” in *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 43–54, IEEE, 2019.