Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

X. Spezialfälle: Adaptiver Speicherschutz

Wolfgang Schröder-Preikschat

28. Juni 2022



Gliederung

Einleitung

Problemstellung
Sicherheitsdimension
Prozessorarchitektur
Störungsquelle

Lösungsansatz
Adressraumkonzept
Prozessisolation

Intermittierende Prozessisolation Prinzip

Zusammenfassung





- **gemeinsamer Hauptspeicher** (*shared memory*) bildet die typische Basis für mehr-/vielkernige Prozessoren
 - dabei verfügt jeder einzelne Rechenkern über eine eigene MMU
 - wichtiger Bestandteil jeder MMU ist der TLB
 - translation lookaside buffer, auch bekannt als Übersetzungspuffer



gemeinsamer Hauptspeicher (*shared memory*) bildet die typische Basis für mehr-/vielkernige Prozessoren

- für derartige Prozessoren sollten Anwendungen mehr-/vielfädige **Prozesse** ermöglichen, um in ihrer Leistung skalieren zu können [10]
 - verteilt über die Rechenkerne finden diese Prozesse gleichzeitig statt
 - sie bilden sogenannte gleichzeitige Prozesse
 - dabei teilen sich die Prozessfäden einer Anwendung denselben Adressraum



Einleitung

gemeinsamer Hauptspeicher (*shared memory*) bildet die typische Basis für mehr-/vielkernige Prozessoren

für derartige Prozessoren sollten Anwendungen mehr-/vielfädige Prozesse ermöglichen, um in ihrer Leistung skalieren zu können [10]

- bewirkt eine Fadenaktion eine Adressraumänderung, ist diese auf den Rechenkernen der anderen Fäden der Anwendung nachzuziehen
 - erzwungenes Unterbrechen von Prozessoren für TLB-Konsistenzaktionen
 - ein wahres "Niederschießen" von Rechenkernen: "*TLB shootdown*" [2]



gemeinsamer Hauptspeicher (*shared memory*) bildet die typische Basis für mehr-/vielkernige Prozessoren

für derartige Prozessoren sollten Anwendungen mehr-/vielfädige **Prozesse** ermöglichen, um in ihrer Leistung skalieren zu können [10]

bewirkt eine Fadenaktion eine Adressraumänderung, ist diese auf den Rechenkernen der anderen Fäden der Anwendung nachzuziehen

Adressraumisolation von Prozessen ist ein **Anwendungskriterium**, das nur im Bedarfsfall vom System durchgesetzt wird [5, 4]



Gliederung

Einleitung

Problemstellung
Sicherheitsdimension
Prozessorarchitektur
Störungsquelle

Lösungsansatz Adressraumkonzept Prozessisolation

Intermittierende Prozessisolation Prinzip

Zusammenfassung



- Sicherheit (security)
 - Schutz einer Entität vor ihrer Umwelt





- Sicherheit (security)
 - Schutz einer Entität vor ihrer Umwelt
- differenzierte Zugriffskontrolle
 - Text, Daten, Stapel, . . .
 - lesen, schreiben, ausführen, ...





- Sicherheit (security)
 - Schutz einer Entität vor ihrer Umwelt

- erreichbar durch:
 - Überwachung der Adressen
 - typisch für Mehradressraumsysteme
 - Randomisierung von Adressbereichen
 - möglich für Einadressraumsysteme
 - Typsicherheit der Programme
 - abhängig von Sprache/Kompilierer





- Sicherheit (security)
 - Schutz einer Entität vor ihrer Umwelt



- , benutzt" [9] Befehlssatzebene und...
 - Betriebssystemebene oder Kompilierer
 - je nachdem ob Typsicherheit <u>aller</u> Programme gewährleistet ist



- [Betriebs-]**Sicherheit** (*safety*)
 - Schutz der Umwelt vor einer Entität





- [Betriebs-]Sicherheit (safety)
 - Schutz der Umwelt vor einer Entität
- softwarebasiert
 - vollständige Interpretation der Befehle
 - CSIM virtuelle Maschine [6, S. 21]
 - Kompilierung
 - typsichere Programmiersprache





- [Betriebs-]**Sicherheit** (*safety*)
 - Schutz der Umwelt vor einer Entität
- softwarebasiert

- hardwarebasiert
 - MMU- oder MPU-basiert
 - partielle Interpretation der Zugriffe
 - im Ausnahmefall (trap)
 - durch das Betriebssystem





- [Betriebs-]Sicherheit (safety)
 - Schutz der Umwelt vor einer Entität
- softwarebasiert

hardwarebasiert

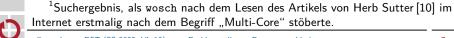


- , benutzt" [9] Befehlssatzebene und...
 - Betriebssystemebene oder Kompilierer
 - je nachdem ob Typsicherheit <u>aller</u> Programme gewährleistet ist



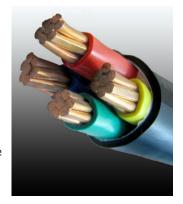
- Symbol¹ eines hochmodernen gekachelten 28-Kern-Prozessors:
 - 7 Kerne pro Kachel
 - cache-kohärent
 - homogen auf Kernebene
 - 4 Kacheln pro CPU
 - cache-inkohärent
 - heterogen auf Kachelebene







- "viel" geht weit über "mehr" hinaus
 - mehrere zehn Kerne als untere Grenze
 - Hunderte oder Tausende sind keine Utopie



¹Suchergebnis, als wosch nach dem Lesen des Artikels von Herb Sutter [10] im Internet erstmalig nach dem Begriff "Multi-Core" stöberte.





- globaler gemeinsamer Hauptspeicher
 - verteilt über alle Kacheln
 - woraus aber nicht zwingend global einheitliche Speicherkonsistenz folgt

¹Suchergebnis, als wosch nach dem Lesen des Artikels von Herb Sutter [10] im Internet erstmalig nach dem Begriff "Multi-Core" stöberte.



Gemeinsamer Arbeitsspeicher

- die Skalierbarkeit des Shared-Memory-Paradigmas steht und fällt mit der Art der Parallelität und der Programmierung [8]
 - grobkörnige Mitbenutzung und Botschaftenaustausch (message passing)
 zeigen etwa den gleichen Grad an Skalierung
 - feinkörnige Ansätze sind dagegen sehr stark durch die Beschränkungen des Gesetzes von Amdahl [1] begrenzt



- ein Problemfall, nicht zuletzt wegen des **TLB** (*translation lookaside buffer*)²
 - einer pro Rechenkern
 - die CPU fährt aber kein Kohärenzprotokoll



²Hieronymus im Gehäus, Albrecht Dürer, 1514: Hieronymus gilt im übertragenen Sinne als Schutzpatron der Übersetzer.

ein Problemfall, nicht zuletzt wegen des **TLB** (translation lookaside buffer)²

- erzeugt Interferenz innerhalb desselben gemeinsam genutzten Adressraums
 - gleichzeitige Prozesse
 - verschiedenen Rechenkernen zugeordnet





²Hieronymus im Gehäus, Albrecht Dürer, 1514: Hieronymus gilt im übertragenen Sinne als Schutzpatron der Übersetzer.

Überlagerung von Aktionen³ beim Zusammentreffen zweier oder mehrerer gleichzeitiger Prozesse

- als Folge des Zugriffs auf eine gemeinsam genutzte Ressource
- auch ausgelöst durch widersprüchliche Planungs-/Auswahlentscheidungen





³Hier bereits die parallele Ausführung eines Maschinenbefehls durch die CPU.

- Rauschen im Hintergrund der Ausführung von Programmen (background noise)
 - nichtfunktionales Merkmal
 - das jede Ausführungsumgebung besitzt



Betriebssystemrauschen (operating-system noise)



Rauschen im Hintergrund der Ausführung von Programmen (*background noise*)

- damit Behinderung eines Prozesses durch gleichzeitige äußere Einwirkungen
 - eines anderen Prozesses
 - auf derselben oder einer anderen CPU



→ **Betriebssystemrauschen** (operating-system noise)

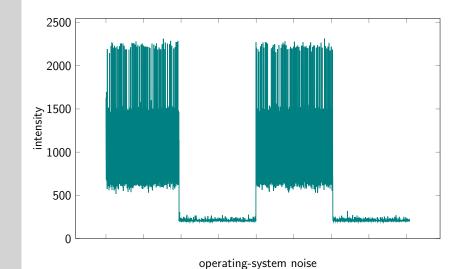


Rauschen im Hintergrund der Ausführung von Programmen (*background noise*)

 damit Behinderung eines Prozesses durch gleichzeitige äußere Einwirkungen

- eigentlich Normalität, sofern die Störung in Grenzen bleibt
- → Betriebssystemrauschen (operating-system noise)







Gliederung

Lösungsansatz Adressraumkonzept Prozessisolation





- ein **Programmiermodell** mit folgender Prämisse [3]:
 - ein global gemeinsam genutzter Adressraum verbessert die Produktivität
 - erfordert aber eine Unterscheidung zwischen "lokal" und "entfernt"
 - um Leistungsoptimierungen zu ermöglichen
 - um Skalierbarkeit auf groß angelegten parallelen Architekturen zu erreichen



ein **Programmiermodell**

- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse



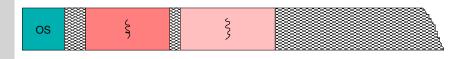
ein **Programmiermodell**

- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse
- $\begin{tabular}{ll} \hookrightarrow & Anforderung/Freigabe von Speicher bedingt Adressraumänderungen \\ \end{tabular}$



• ein **Programmiermodell**

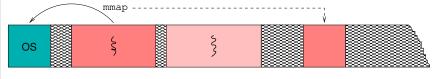
- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse
- Anforderung/Freigabe von Speicher bedingt Adressraumänderungen





ein **Programmiermodell**

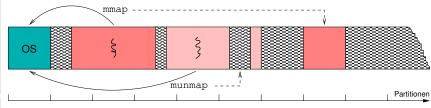
- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse
- → Anforderung/Freigabe von Speicher bedingt Adressraumänderungen





• ein **Programmiermodell**

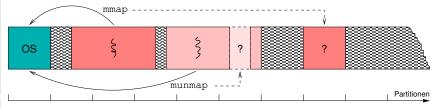
- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse
- Anforderung/Freigabe von Speicher bedingt Adressraumänderungen





ein **Programmiermodell**

- aus Betriebssystemsicht handelt es sich dabei um:
 - mehrere Hauptspeicherpartitionen, die sich kooperierende Prozesse teilen
 - in logischer Hinsicht ein gemeinsamer verteilter Hauptspeicher
 - ein einzelner Adressraum, in dem die kooperierenden Prozesse residieren
 - genauer: ein und derselbe logische/virtuelle Adressraum für diese Prozesse
- → Anforderung/Freigabe von Speicher bedingt Adressraumänderungen





Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen



- Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen
 - Attribute, erfasst über die Seitendeskriptoren der Adressräume
 - Anzahl Größe des gesamten Adressraums
 - Maß für die Gesamtzahl der benötigten Seitentabellen
 - 7ustand Ort einer Seite und die mit ihr verbundenen Rechte



- Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen
 - Attribute, erfasst über die Seitendeskriptoren der Adressräume

- wobei ein Seitendeskriptor eine logische und physische Erscheinung hat
 - logisch der Repräsentant im globalen virtuellen Adressraum
 - identifiziert/lokalisiert die Seite
 - → scheinbar nur einmal vorhanden



- Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen
 - Attribute, erfasst über die Seitendeskriptoren der Adressräume

• wobei ein Seitendeskriptor eine logische und physische Erscheinung hat

```
physisch - der Repräsentant im jeweils lokalen Speicher eines Prozessors
```

- identifiziert/lokalisiert den Seitenrahmen
- → wirklich mehrfach vorhanden, mit verschiedenen Werten



- Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen
 - Attribute, erfasst über die Seitendeskriptoren der Adressräume

- wobei ein Seitendeskriptor eine logische und physische Erscheinung hat
 - logisch der Repräsentant im globalen virtuellen Adressraum
 - identifiziert/lokalisiert die Seite → Referenz
 - → scheinbar nur einmal vorhanden
 - physisch der Repräsentant im jeweils lokalen Speicher eines Prozessors
 - identifiziert/lokalisiert den Seitenrahmen → Replikat
 - → wirklich mehrfach vorhanden, mit verschiedenen Werten



- Änderungen von Attributen global gemeinsam genutzter Adressräume müssen in konsistenter Art und Weise geschehen
 - Attribute, erfasst über die **Seitendeskriptoren** der Adressräume

• wobei ein Seitendeskriptor eine logische und physische Erscheinung hat

- jede **lokale Domäne** führt Buch über die in ihrem Gültigkeitsbereich vorliegenden Verwaltungsstrukturen
 - Bezugspunkt dabei ist die CPU, genauer: der einzelne Rechenkern
 - deren MMU zu programmieren und TLB zu verwalten ist



Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden



- Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden
 - (a) eine für gewöhnlich mehrstufige Seitentabelle pro Prozessexemplar
 - erfasst die Seitendeskriptoren des zugewiesenen Arbeitsspeichers



- Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden
 - (b) Liste der Gebrauchsstücke an Hauptspeicher pro Prozessexemplar
 - erfasst zugewiesene statische/dynamische (seitennummerierte) Segmente



Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden

- (c) der Übersetzungspuffer (TLB) pro CPU/Rechenkern
 - erfasst kürzlich erfolgte Übersetzungen von logischen/virtuellen Adressen



Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden

- (d) kernglobale Speicherverwaltung pro Kachel (tile)
 - erfasst verfügbare statische/dynamische (seitennummerierte) Segmente

Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden

- (e) eine globale Seitentabelle pro System
 - erfasst alle Maschinenprogramme plus Betriebssystem

Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden

- bei Adressraumänderungen ist dabei ein **verteilter Zustand** logisch zusammenhängender TLB/MMU Einheiten konsistent zu halten
 - relevant sind Aktionen, die den global gemeinsamen Adressraum betreffen



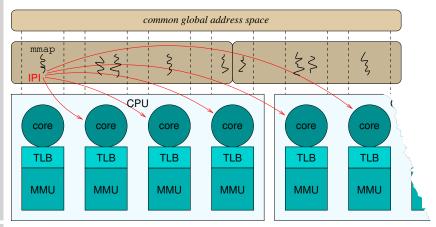
Prozessisolation "benutzt" bestimmte **funktionale Merkmale**, die typischerweise einem Betriebssystem zugerechnet werden

- (c) der Übersetzungspuffer (TLB) pro CPU/Rechenkern
 - erfasst kürzlich erfolgte Übersetzungen von logischen/virtuellen Adressen

- bei Adressraumänderungen ist dabei ein verteilter Zustand logisch zusammenhängender TLB/MMU Einheiten konsistent zu halten
 - relevant sind Aktionen, die den global gemeinsamen Adressraum betreffen
 - allerdings sind TLB/MMU nur lokal "von ihrem Kern aus" zugängig → per IPI (interprocessor interrupt) sind Kerne zur "Wartung" anzustoßen

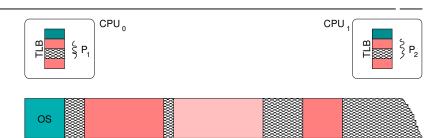


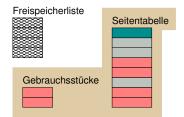
- auf Basis kernlokaler Ressourcen zur globalen Adressraumverwaltung
 - insbesondere MMU und TLB
 - ightarrow die nur lokal zugängig sind, aber von entfernt den Zustand ändern müssen





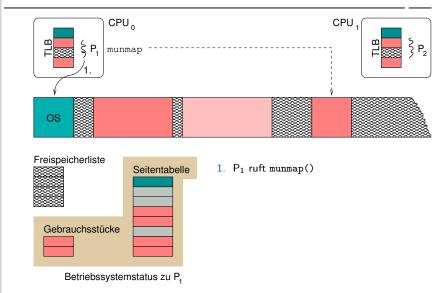




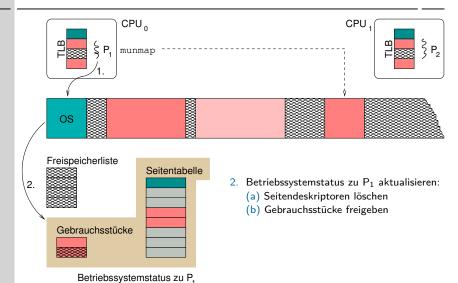


Betriebssystemstatus zu P,



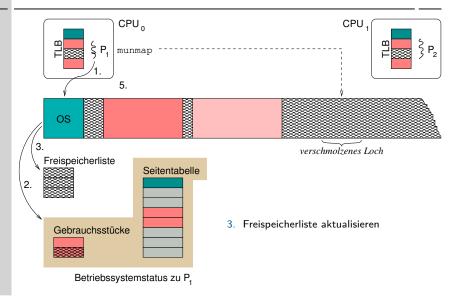




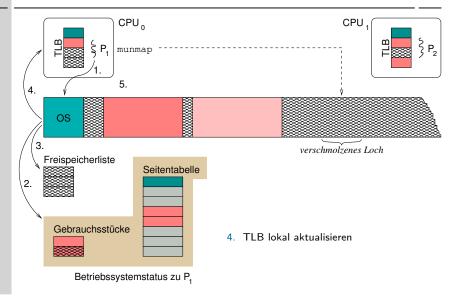




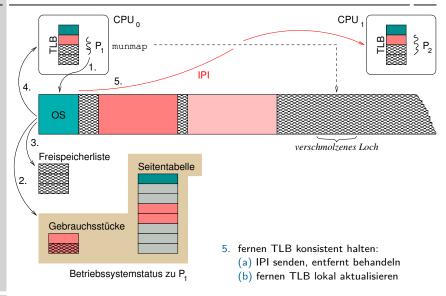
@ wosch



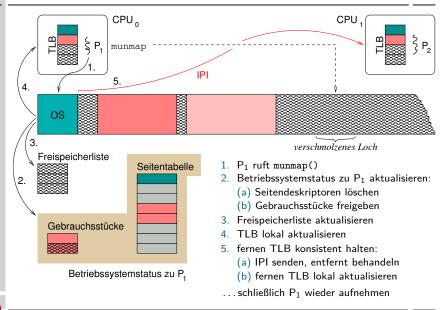














@ wosch

Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen



- Simultanverarbeitung mit einem Mehrprozessorsystem hat **Untiefen**
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte



- Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte
 - ist der betreffende Systemaufruf (z.B. munmap) der Abstraktionsebene des Fadens zuzurechnen, darf ein kritischer Abschnitt angenommen werden
 - → Bestandteil des nichtsequentiellen Programms, das den Fadens definiert



- Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

- "versteckt" sich der Systemaufruf jedoch in einer tieferliegenden Ebene, etwa innerhalb einer Bibliotheksfunktion, fehlt Wissen zur Prozesssynchronisation → und zwar "oben" wie auch "unten" → Annahme des schlimmsten Falls



- Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

• für einen über die Rechenkerne verteilter mehr-/vielfädiger Prozess ist der IPI ein **Gruppenruf** (*multicast*) oder **Rundruf** (*broadcast*)



- Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

- für einen über die Rechenkerne verteilter mehr-/vielfädiger Prozess ist der IPI ein **Gruppenruf** (*multicast*) oder **Rundruf** (*broadcast*)
 - dieser sollte mit der Anzahl der dem gleichzeitigen Prozess jeweils dynamisch zugewiesenen Rechenkerne skalieren



- Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

- für einen über die Rechenkerne verteilter mehr-/vielfädiger Prozess ist der IPI ein **Gruppenruf** (*multicast*) oder **Rundruf** (*broadcast*)
 - dieser sollte mit der Anzahl der dem gleichzeitigen Prozess jeweils dynamisch zugewiesenen Rechenkerne skalieren
 - zudem muss der rufende Betriebssystemkern Gewissheit darüber haben, dass sämtliche gerufenen "IPI handler" ihre Aufgaben haben erledigen können



- Simultanverarbeitung mit einem Mehrprozessorsystem hat **Untiefen**
 - der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

für einen über die Rechenkerne verteilter mehr-/vielfädiger Prozess ist der IPI ein **Gruppenruf** (multicast) oder **Rundruf** (broadcast)

→ skalierbarer verlässlicher Gruppenruf (scalable reliable multicast)



Simultanverarbeitung mit einem Mehrprozessorsystem hat Untiefen

 der eine Adressraumänderung auslösende Faden erzeugt eine Laufgefahr (race condition) durch zeitweilig inkonsistente TLB-Inhalte

• für einen über die Rechenkerne verteilter mehr-/vielfädiger Prozess ist der IPI ein **Gruppenruf** (*multicast*) oder **Rundruf** (*broadcast*)

• nicht zuletzt die durch den IPI verursachte **Störung** ferner Rechenkerne, die diese zu unbekannten Zeitpunkten mit unbekannter Frequenz trifft



Offenbetrieb

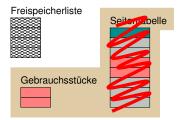




Offenbetrieb ...störfrei



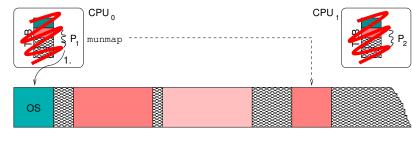


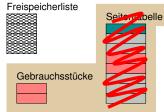


Betriebssystemstatus zu P₁



Offenbetrieb ...störfrei



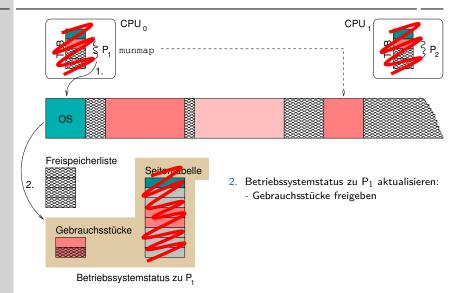


1. P₁ ruft munmap()

Betriebssystemstatus zu P₁

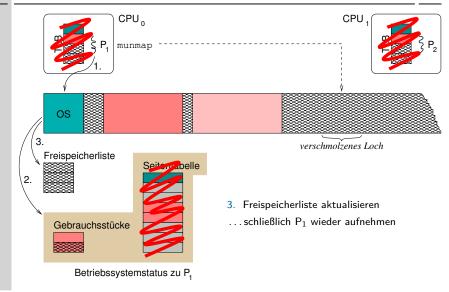


Offenbetrieb ...störfrei





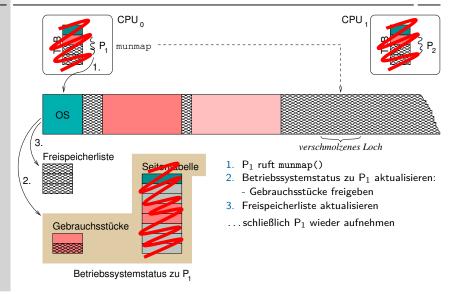
Offenbetrieb ... störfrei





© wosch

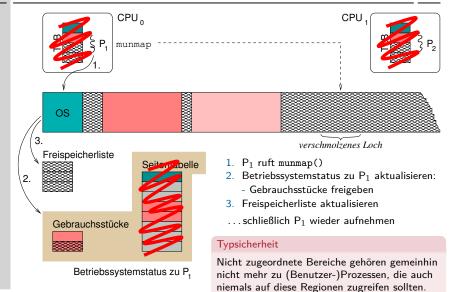
Offenbetrieb . . . störfrei





@ wosch

Offenbetrieb störfrei





Es war einmal ein free()...

...das Haldenspeicher an das Betriebssystem zurückgab und damit dafür sorgte, ungenutzten Hauptspeicher aus den Adressraum des Prozesses ausblenden zu können — das **Kontextwissen** vermittelte.



Es war einmal ein free()...

...das Haldenspeicher an das Betriebssystem zurückgab und damit dafür sorgte, ungenutzten Hauptspeicher aus den Adressraum des Prozesses ausblenden zu können — das **Kontextwissen** vermittelte.



Was soll's ... interne Fragmentierung in Seiten [7, S. 16] wird auch toleriert!



Es war einmal ein free()...

...das Haldenspeicher an das Betriebssystem zurückgab und damit dafür sorgte, ungenutzten Hauptspeicher aus den Adressraum des Prozesses ausblenden zu können — das Kontextwissen vermittelte.



die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird



- die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird
 - angenommen ein Prozess fordert 42 Bytes an dynamischen Speicher an, den das Betriebssystem nur in Einheiten von 4096 Bytes vergibt



- die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird
 - angenommen ein Prozess fordert 42 Bytes an dynamischen Speicher an, den das Betriebssystem nur in Einheiten von 4096 Bytes vergibt
 - weiter angenommen, der angeforderte dynamische Speicher wird an einer Seitengrenze im Prozessadressraum verfügbar gemacht



- die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird
 - angenommen ein Prozess fordert 42 Bytes an dynamischen Speicher an, den das Betriebssystem nur in Einheiten von 4096 Bytes vergibt
 - weiter angenommen, der angeforderte dynamische Speicher wird an einer Seitengrenze im Prozessadressraum verfügbar gemacht
 - sei char* daaaf Zeiger auf einen verfügbar gemachten Speicherbereich und int(daaaf)/4096 = 0, also ein seitenausgerichteter Wert



die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird

• so ist die Ausführung der Anweisungsfolge

```
daaaf = malloc(42);
if (daaaf)
for (i = 0; i < 0815; i++)
daaaf[i] = '*';</pre>
```

für alle Werte von i physisch gültig, obwohl nur $i \in [0,41]$ logisch gilt



die CPU greift wortweise auf den Hauptspeicher zu, der allerdings vom Betriebssystem seitenweise zur Verfügung gestellt wird

• so ist die Ausführung der Anweisungsfolge

```
daaaf = malloc(42);
if (daaaf)
for (i = 0; i < 0815; i++)
daaaf[i] = '*';</pre>
```

für alle Werte von i physisch gültig, obwohl nur $i \in [0,41]$ logisch gilt

■ das heißt, die ungültigen Zugriffe auf daaaf [j] mit $j \in [42, 4095]$ werden keinen Fehler liefern, da dieser Bereich noch zur gültigen Seite zählt



Gliederung

Einleitung

Problemstellung

Sicherheitsdimension

Prozessorarchitektur

Störungsquelle

Lösungsansatz

Adressraumkonzept

Prozessisolation

Intermittierende Prozessisolation Prinzip

Zusammenfassung



■ Übergänge zwischen verschiedenen **Betriebsmodi** ermöglichen



- Übergänge zwischen verschiedenen **Betriebsmodi** ermöglichen
 - MMU-basierten Schutz des Prozessadressraums ausschalten
 - Seitentabelle (für alle Fäden) des betreffenden Prozesses deaktivieren
 - sofern nötig, nur die globale (systemspezifische) Seitentabelle aktiv lassen



Übergänge zwischen verschiedenen **Betriebsmodi** ermöglichen

- MMU-basierten Schutz des Prozessadressraums einschalten
 - Seitendeskriptoren auf Basis der Gebrauchsstücke wiederherstellen
 - Seitentabelle (für alle Fäden) des betreffenden Prozesses aktivieren
 - lokale (prozessspezifische) Seitentabelle anwenden



- Übergänge zwischen verschiedenen **Betriebsmodi** ermöglichen
 - MMU-basierten Schutz des Prozessadressraums ausschalten
 - MMU-basierten Schutz des Prozessadressraums einschalten

- Ein- und Ausschalten ist eine "lärmende Aktion"
 - IPI als Gruppenruf (*multicast*) an die relevanten Prozessoren senden
 - den TLB der jeweiligen MMU spülen (flush)



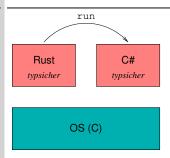
- Übergänge zwischen verschiedenen **Betriebsmodi** ermöglichen
 - MMU-basierten Schutz des Prozessadressraums ausschalten
 - MMU-basierten Schutz des Prozessadressraums einschalten

Ein- und Ausschalten ist eine "lärmende Aktion" — jedoch "selten"

- die MMU isoliert nur Prozesse typunsicherer Programme
 - Prozesse typsicherer Programme sind durch den Kompilierer isoliert
 - so ergeben sich zwei Momente für den Wechsel des Betriebsmodus'
 - Ladezeitpunkt: typunsicheres Programm per MMU im Adressraum isolieren
 - Entsorgungszeitpunkt: Isolation des typunsicheren Programms aufheben



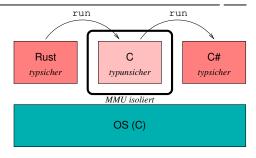
Betriebsmodi



Prozesse typischerer Maschinenprogramme im Offenbetrieb belassen



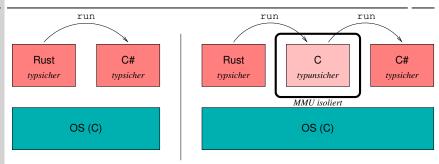
Betriebsmodi teils isoliert



- nur die Prozesse typunsicherer Maschinenprogramme beim Laden im eigenen Adressraum einschließen → Isolationsbetrieb
 - Schutz der Umgebung vor solchen Prozessen



Betriebsmodi offen vs. isoliert



- Prozesse typischerer Maschinenprogramme im Offenbetrieb belassen
- nur die Prozesse typunsicherer Maschinenprogramme beim Laden im eigenen Adressraum einschließen \sim Isolationsbetrieb
- Unabhängig von der Auslegung als MAS oder SAS, jedoch:
 - MAS physische Isolation der unsicheren Teile mittels MMU (hier)
 - SAS ggf. nur logische Isolation durch Adressraumrandomisierung



Gliederung

Einleitung

Problemstellung

Sicherheitsdimension

Prozessorarchitektur

Störungsquelle

Lösungsansatz

Adressraumkonzept

Prozessisolation

Intermittierende Prozessisolation Prinzip

Zusammenfassung





- erzwungenes Unterbrechen von Prozessoren in ihren Aktivitäten, um dort TLB-Konsistenzaktionen durchzuführen, ist massiv störend
 - systemweiter IPI als Folge einer Adressraumänderung eines Fadens
 - der IPI unterbricht die laufende Aktion des empfangenden Rechenkerns
 - TLB shootdown [2]



erzwungenes Unterbrechen von Prozessoren in ihren Aktivitäten, um dort TLB-Konsistenzaktionen durchzuführen, ist massiv störend

- diese Maßnahme forciert Überlagerungen beim Zusammentreffen der Unterbrechung mit einem gleichzeitigen Prozess
 - Interferenz, deren Zeitpunkt und Frequenz unbestimmt ist
 - verursacht durch einen "externen" Prozessfaden eines fernen Rechenkerns



erzwungenes Unterbrechen von Prozessoren in ihren Aktivitäten, um dort TLB-Konsistenzaktionen durchzuführen, ist massiv störend

- diese Maßnahme forciert Überlagerungen beim Zusammentreffen der Unterbrechung mit einem gleichzeitigen Prozess
 - Interferenz, deren Zeitpunkt und Frequenz unbestimmt ist
 - verursacht durch einen "externen" Prozessfaden eines fernen Rechenkerns
 - die Störungen sind problematisch für zeitabhängige gleichzeitige Prozesse
 - beeinträchtigen aber auch allgemein die Performanz paralleler Anwendungen
 - insb. Anwendungen, die "Gleichschritt" (lock-step) von Fäden erwarten



(3)

erzwungenes Unterbrechen von Prozessoren in ihren Aktivitäten, um dort TLB-Konsistenzaktionen durchzuführen, ist massiv störend

 diese Maßnahme forciert Überlagerungen beim Zusammentreffen der Unterbrechung mit einem gleichzeitigen Prozess

- speichersichere Prozesse vor zu erwartenden Störungen bewahren, die ein MMU-basierter Schutz zur Folge haben kann
 - nur speicherunsichere Prozesse durch die MMU überwachen lassen
 - Adressraumisolation von Prozessen adaptiv gestalten [4]



erzwungenes Unterbrechen von Prozessoren in ihren Aktivitäten, um dort TLB-Konsistenzaktionen durchzuführen, ist massiv störend

diese Maßnahme forciert Überlagerungen beim Zusammentreffen der Unterbrechung mit einem gleichzeitigen Prozess

speichersichere Prozesse vor zu erwartenden Störungen bewahren. die ein MMU-basierter Schutz zur Folge haben kann



Literaturverzeichnis I

[1] AMDAHL. G. M.:

Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities.

In: Proceedings of the AFIPS Spring Joint Computer Conference (AFIPS 1967 (Spring)), AFIPS Press, 1967, S. 483-485

- [2] BLACK, D. L.; RASHID, R. F.; GOLUB, D. B.; HILL, C. R.: Translation Lookaside Buffer Consistency: A Software Approach. In: Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems. New York, NY, USA: Association for Computing Machinery, 1989 (ASPLOS III). -ISBN 0897913000. S. 113-122
- [3] DE WAEL, M.; MARR, S.; DE FRAINE, B.; VAN CUTSEM, T.; DE MEUTER, W.: Partitioned Global Address Space Languages. In: ACM Comput. Surv. 47 (2015), Mai, Nr. 4, S. 62:1-27. http://dx.doi.org/10.1145/2716320. -
 - DOI 10.1145/2716320



Literaturverzeichnis II

- [4] Drescher, G.:
 - Adaptive Address-Space Management for Resource-Aware Applications, Lehrstuhl für Verteilte Systeme und Betriebssysteme, Department Informatik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Dissertation, 2021. https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/16511
- [5] DRESCHER, G.; SCHRÖDER-PREIKSCHAT, W.: Adaptive Memory Protection for Many-Core Systems. Version: 2016

http://dx.doi.org/10.4230/DagRep.6.10.120.

In: Adaptive Isolation for Predictability and Security Bd. 6.

Dagstuhl Publishing, 2016. –

DOI 10.4230/DagRep.6.10.120, S. 120-153:140

[6] GOLDBERG, R. P.:

Architectural Principles for Virtual Computer Systems / Harvard University, Electronic Systems Division.

Cambridge, MA, USA, Febr. 1973 (ESD-TR-73-105). – PhD Thesis



Literaturverzeichnis III

[7] Kleinöder, J.; Schröder-Preikschat, W.: Stapelverarbeitung. In: Lehrstuhl Informatik 4 (Hrsg.): Systemprogrammierung. FAU Erlangen-Nürnberg, 2015 (Vorlesungsfolien), Kapitel 7.1

[8] Kremenetsky, M.; Raefsky, A.; Reinhardt, S.: Poor Scalability of Parallel Shared Memory Model: Myth or Reality? In: Proceedings of the 2003 International Conference on Computational Science (ICCS 2003) Bd. LNCS 2660, Springer-Verlag Berlin Heidelberg, 2003 (Lecture Notes in Computer Science), S. 657-666

[9] Schröder-Preikschat, W.:

Hierarchien

In: Lehrstuhl Informatik 4 (Hrsg.): Betriebssystemtechnik — Adressräume: Trennung, Zugriff, Schutz. FAU Erlangen-Nürnberg, 2013 (Vorlesungsfolien), Kapitel 4

[10] SUTTER, H.:

Turn Toward Concurrency in Software—Your free lunch will soon be over. What can vou do about it?

In: Dr. Dobb's Journal 30 (2005), März, Nr. 3, S. 16–22

