Integrating Segmentation and Paging Protection for Safe, Efficient and Transparent Software Extensions

Tzi-cker Chiueh
Ganesh Venkitachalam
Prashant Pradhan

Computer Science Department
State University of New York
Stony Brook, NY 11794-4400

http://www.ecsl.cs.sunysb.edu/palladium.html

Introduction

- **Dynamic extensibility emerges as the major research theme and product trend**

  Extensible operating systems: Windows NT
  Extensible database systems: Informix, DB2, Oracle
  Extensible applications: Adobe's Premiere, Apache Web Server
  Active Networking

- **Component-based software development methodology**

  A single application consists of components produced by multiple vendors ==> Whose bugs cause application malfuction?

- **Need an Intra-address space protection mechanism to quarantine erroneous or malicious software components**

Palladium

- **A Linux-based system that supports safe user-level and kernel-level software extensions using Intel X86 architecture's segmentation and paging hardware**
- **Provide the same level of protection as using separate address spaces**
- **Fastest protection domain switching: 142 CPU cycles for a null protected procedure call and return**
- **Minimal changes required to existing programming tools and conventional linear-address-space programming model**

X86 Virtual Memory Hardware

- **Virtual Address: 16-bit segment selector and 32-bit offset**

  $\textit{Virtual Address} \xrightarrow{\textbf{Segmentation}} \textit{Linear Address} \xrightarrow{\textbf{Paging}} \textit{Physical Address}$

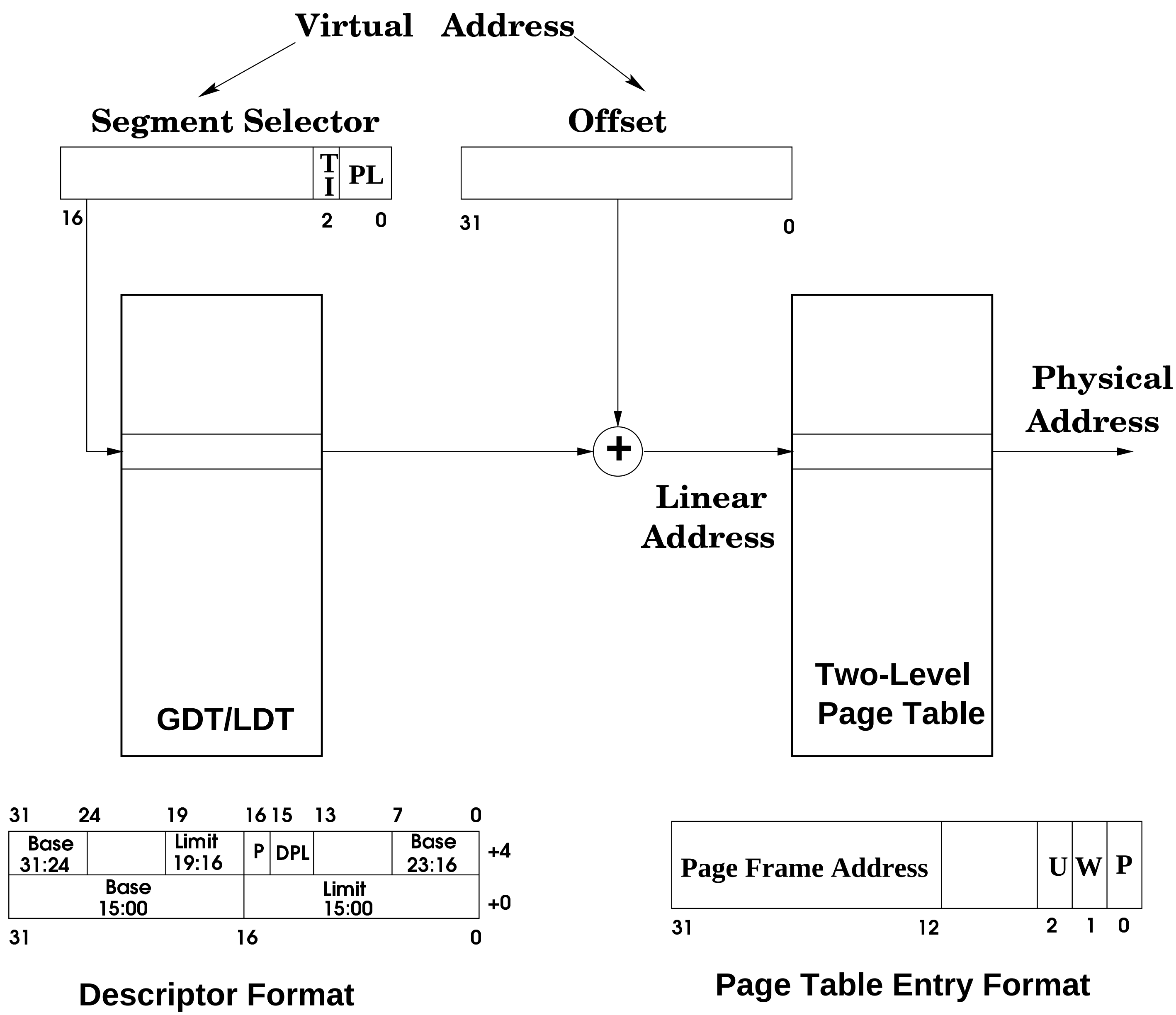
- **Segment-level Protection Check**

  4 Segment Protection Levels (SPL)
  Segment Limit

- **Page-level Protection Check**

  2 Page Protection Levels (PPL)
  Read/Write Permission

X86 Virtual Memory Hardware



X86 Virtual Memory Hardware

- **Mapping between SPL and PPL**

  SPL 0, 1, 2 ⟶ PPL 0

  SPL 3 ⟶ PPL 1

- **Control transfer among protection domains**

  `lcall call-gate-ID` `lret`

  Switch to the stack associated with the destination SPL

- **Only supports transfer starting from more privileged level to less privileged level and back**

- **On a process switch, page-table base address register is reloaded and TLB is flushed**

Palladium's Extension Programming Model

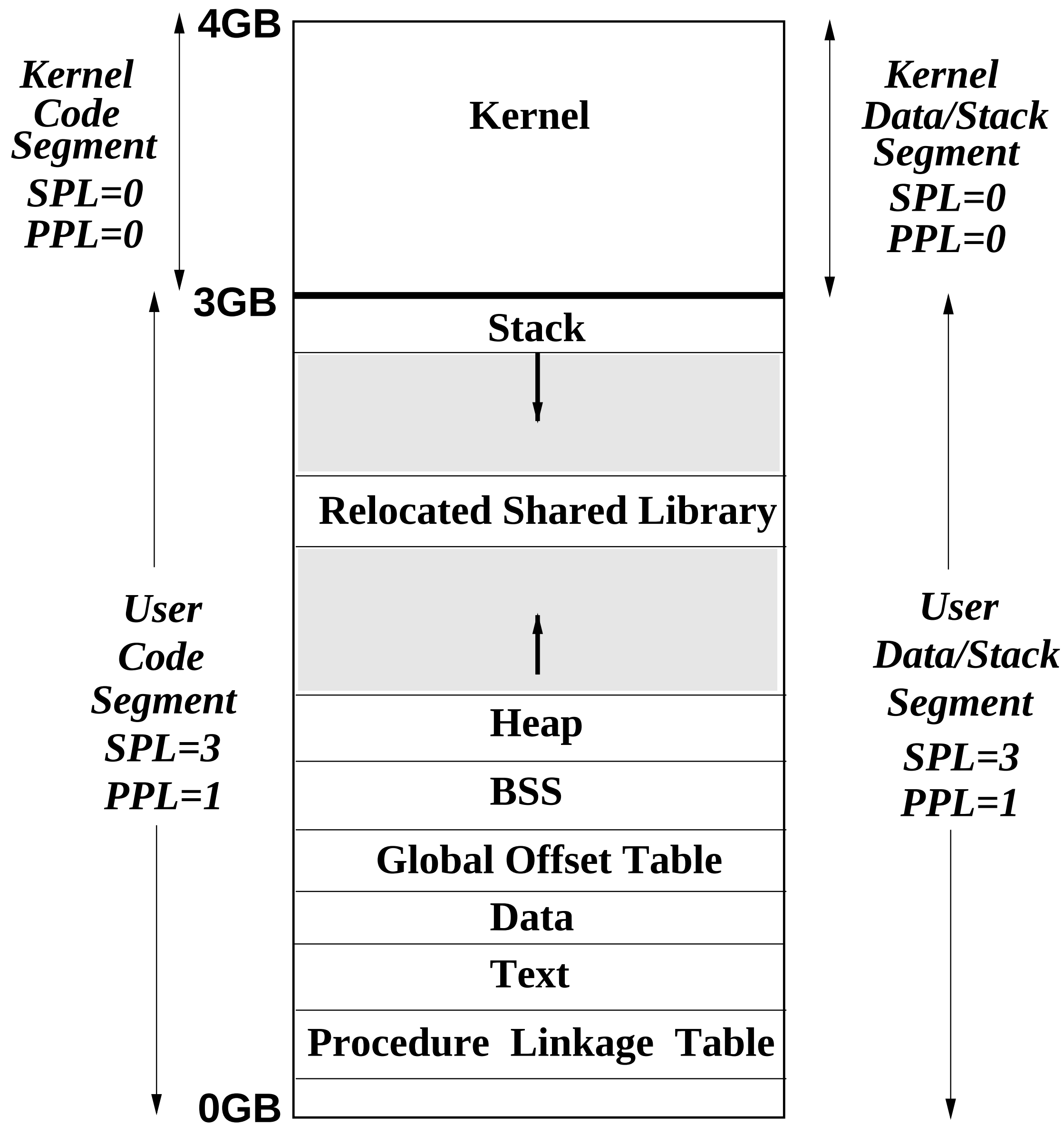
- **A main program (kernel or extensible application) is protected from its dynamically-linked extension modules, but not vice versa**

- **Extensions are protected function calls. Among extension modules, only safety-strength but not security-strength protection**

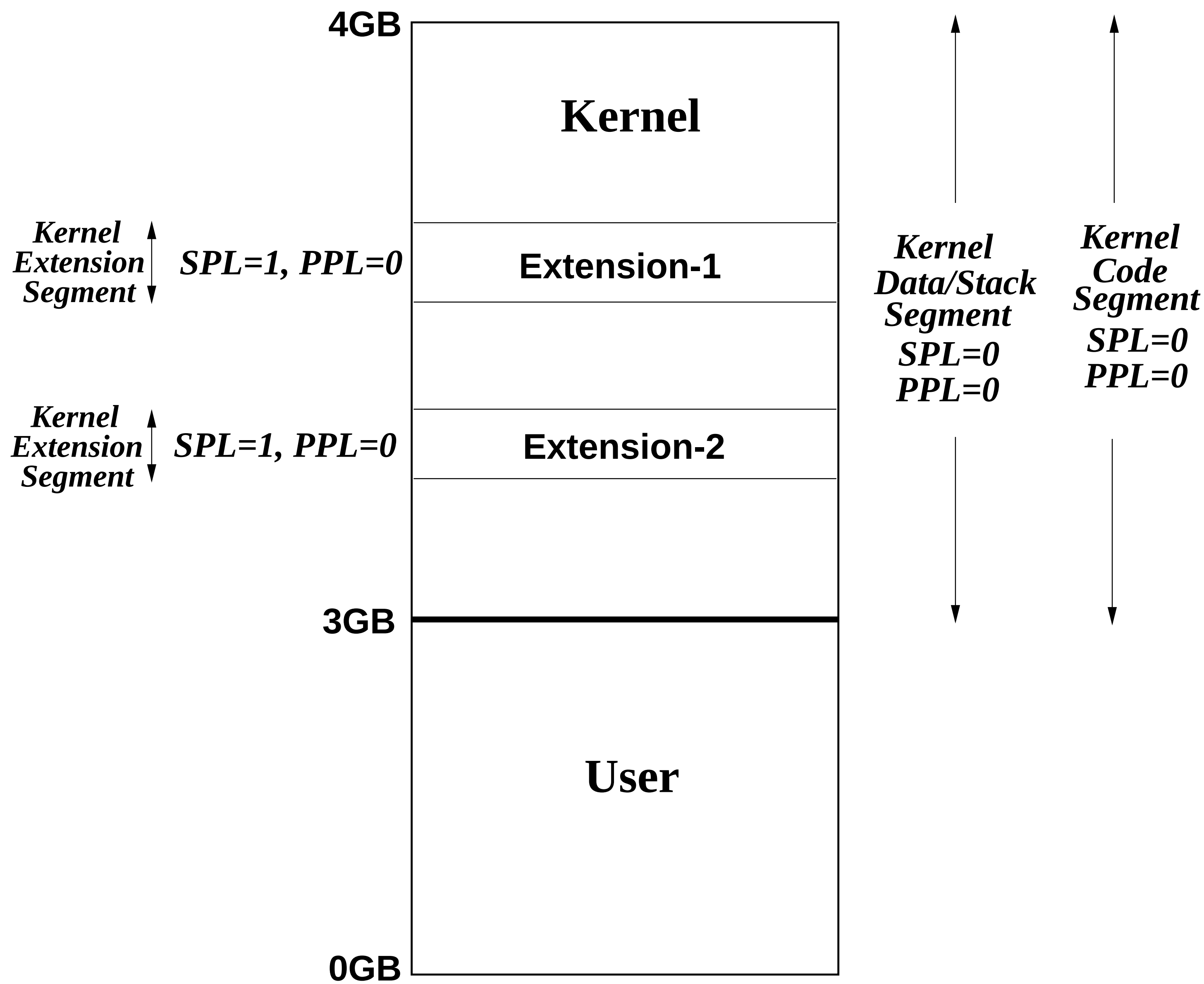
- **Shared data regions between protection domains are available to reduce data copying**

- **User-level extensions make system calls through hosting applications; kernel-level extensions are allowed to access only selective core kernel services**

Linux's Virtual Address Space



Kernel Extension Mechanism



Kernel Extension Mechanism

- **Allow multiple extension segments, each of which can hold multiple extension modules that are loaded dynamically via** `insmod`

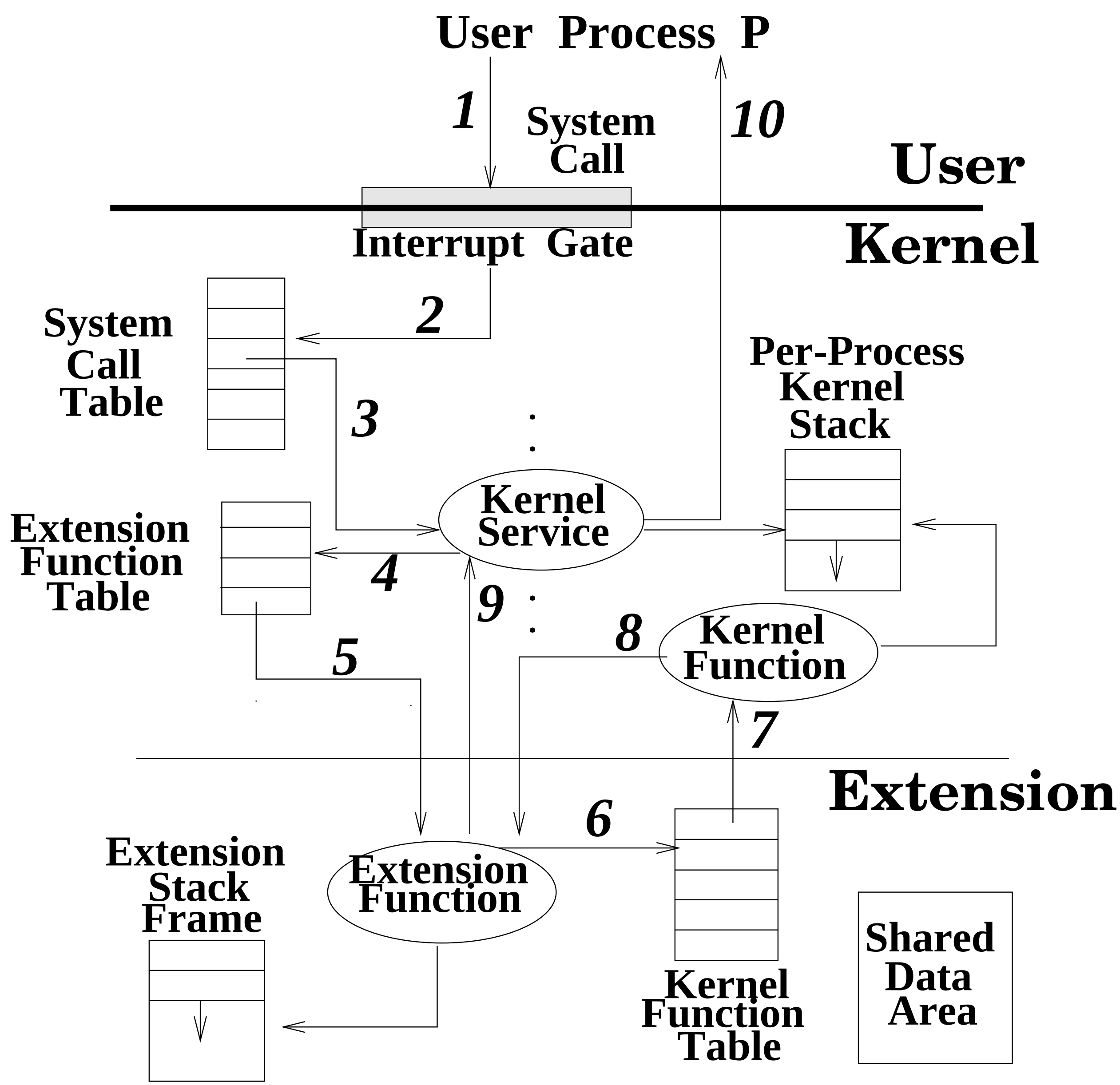
- **One stack per extension segment. Modules loaded into the same segment cannot run concurrently**

- **Kernel extension modules can access selective core kernel services such as kmalloc**

- **Kernel service functions called by kernel extensions execute in the context of the kernel stack of the triggering user process or the ‘‘Idle’’ process**

- **Shared data region allocated in extension segment**

Kernel Extension Mechanism



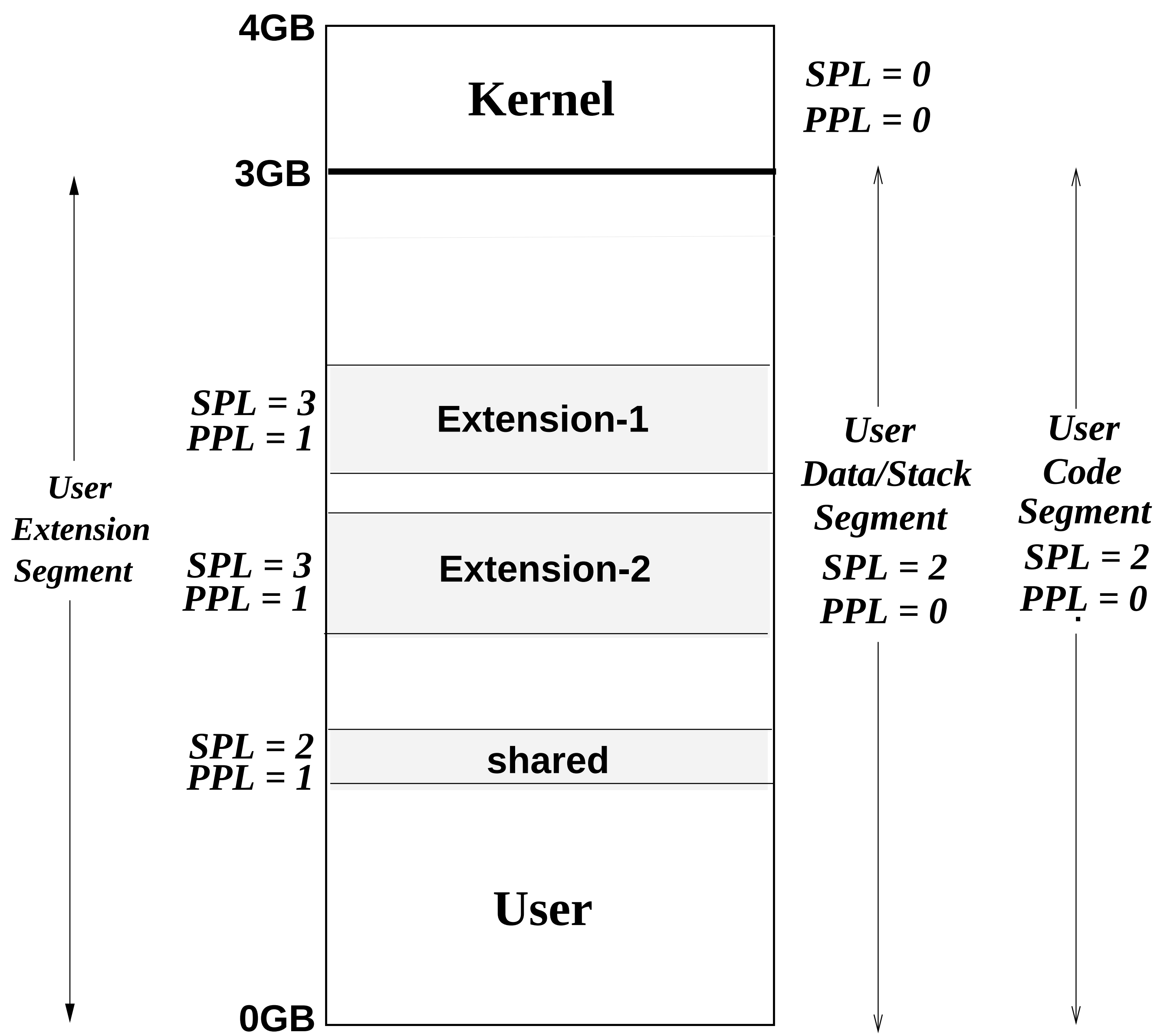
User-Level Extension Mechanism

Why the segmentation approach is not good?

- **Passing data/code pointers between protection domains requires swizzling because of different base addresses**
- **Gcc and ld need to be modified, because they assume a flat linear address space**
- **Difficult to support stateful shared library routines such as fprintf()**

Solution: Combining page-level and segment-level protection checks

User-Level Extension Mechanism



Programming Interface

- Use seg_dlopen, seg_dlsym and seg_dlclose to load access and close dynamically-loaded modules
- Call init_PL in the beginning to be safely extensible
- Use set_range to expose shared library code pages
- Use set_call_gate to package application service functions that user extensions can invoke
- Use xmalloc rather than malloc
- Invoke gcc with a specific linker script to ensure that Global Offset Table be placed on a separate page

Implementation Issues

- **X86 architecture's lcall goes from less-privileged level to more-privileged level, and lret for the other direction**

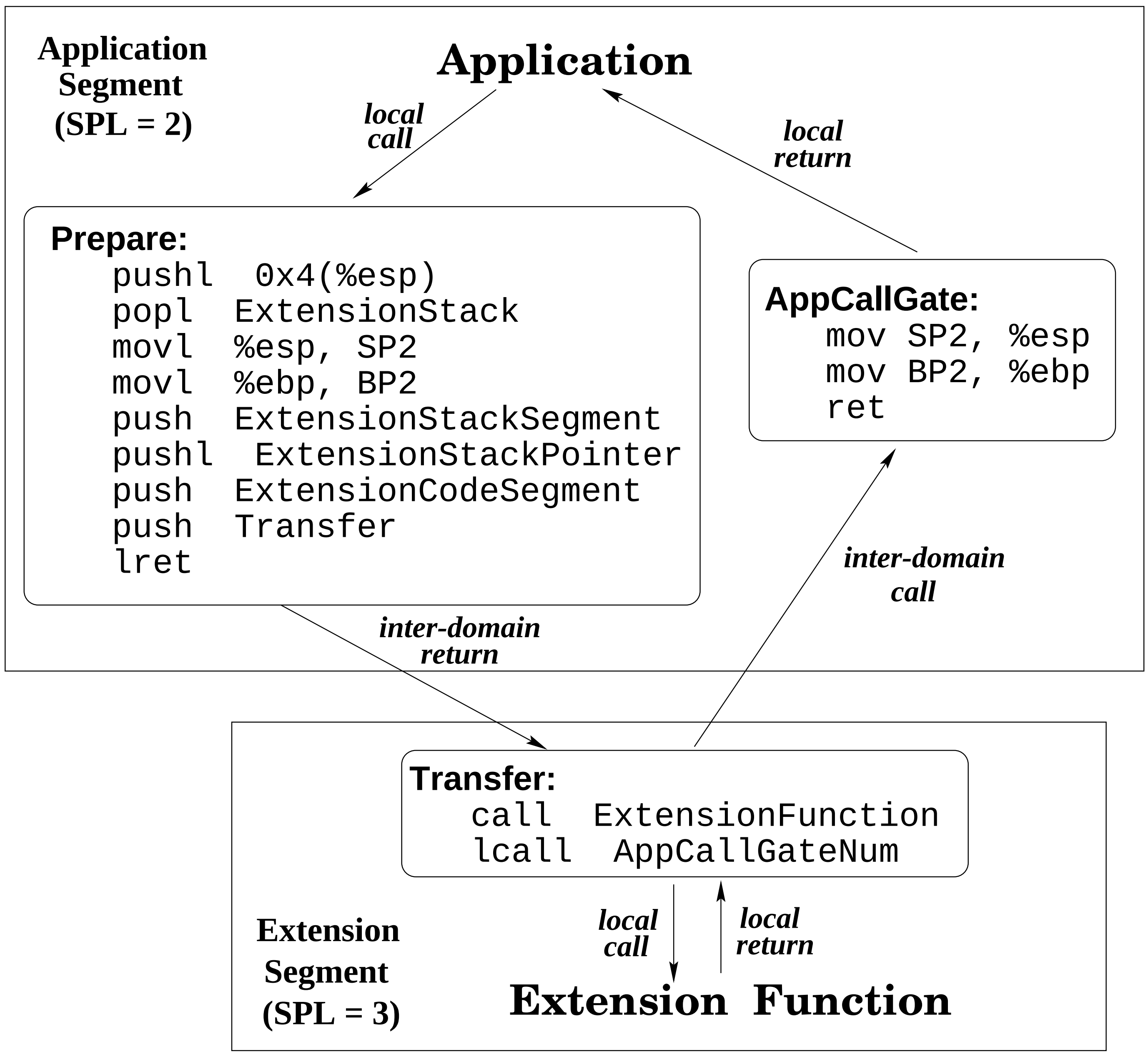
- **Gcc and ld do not know segments**

Solution: Add one level of indirection by dynamically generating code sequences to hide inter-domain control transfers and call/return semantic mismatch

- **Seg_dlsym returns modified function pointers**

- **Saving SP and BP at user space to avoid system calls**

Control Transfer



Kernel Support

- Additional protection check at page fault time based on calling code segment's SPL and faulted page's PPL

- System call check based on application's SPL and the calling code segment's SPL

- A lifetime timer is set at the beginning of invoking an extension to prevent ''infinite-loop'' bugs. Timer value is left to be a policy issue

- Deliver signals to user applications when protection faults arise or lifetime timers expire. No support for system state cleanup other than resource reclamation

Performance Results

Micro-Benchmark: Null protected procedure call

Component	Intra-Domain	Inter-Domain	Hardware
Setting up stack	2	26	5
Calling function	3	34	22
Returning to caller	3	75	44
Restoring state	2	7	5
Total Cost	10	142	89

Performance Results

Micro-Benchmark: Reverse-string function with different input string sizes, in micro-seconds

String Size (Bytes)	Unprotected Call	Palladium Call	Linux RPC
32	2.20	2.79	349.19
64	4.06	4.65	352.55
128	7.78	8.37	374.20
256	15.22	15.97	423.33

Performance Results

Macro-Benchmark: user-level extension

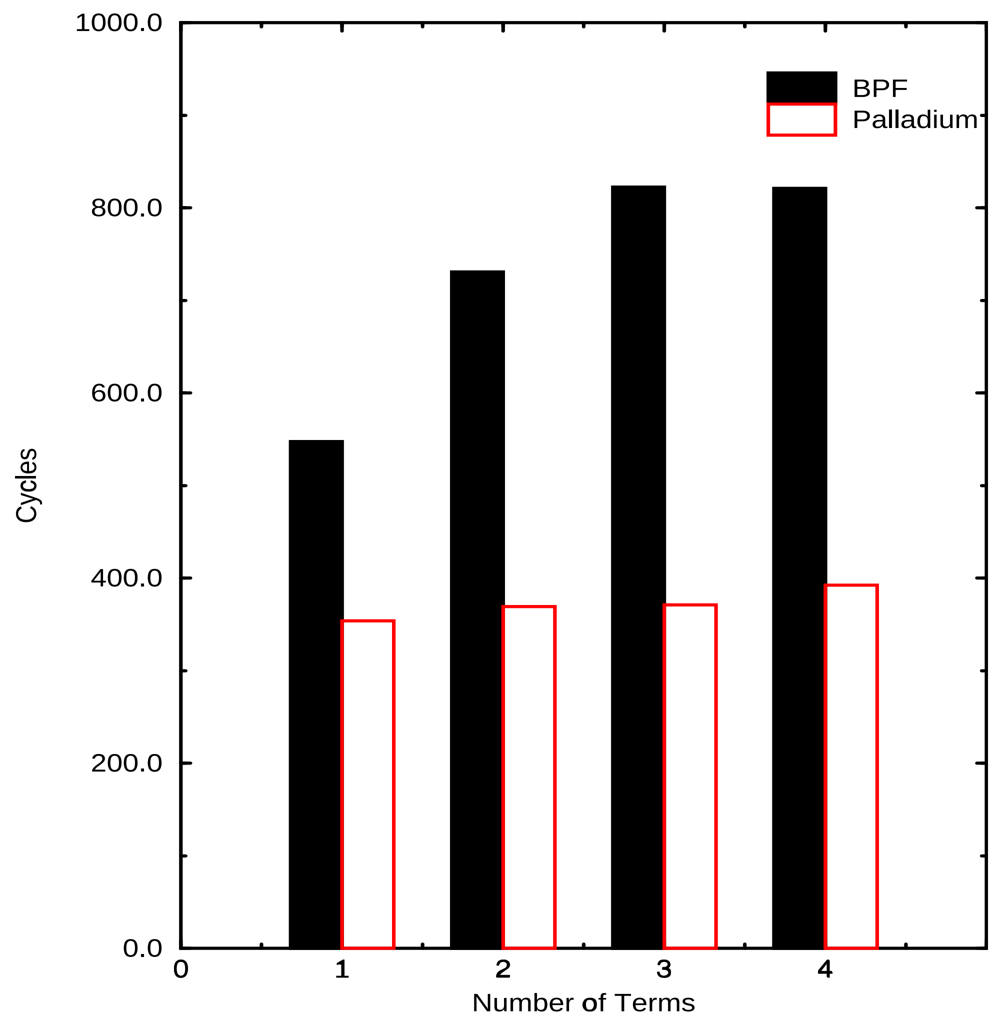
Fast CGI Invocation -- CGI script running as a protected function call, in requests/sec

HTML file size	CGI	FastCGI	LibCGI (Palladium)	LibCGI (unprotected)	Web Server
28Byte	98	193	437	448	460
1KByte	92	188	423	431	436
10KBytes	76	130	311	312	315
100KBytes	33	52	57	57	57

Performance Results

Macro-Benchmark:

Compiled versus Interpretive Packet Filtering



Conclusion

- **Palladium provides safe, efficient and transparent user-level and kernel-level software extensions**

- **The key idea is to exploit both paging and segmentation hardware feature in X86 architecture**

- **Future Work**

  Combine multi-threading with multiple protection domains

  Exploit segmentation hardware to implement other kernel services such as narrow-interfaced protected memory

  Debugger support for multi-segment programming

  More applications development experiences in database and 3D graphics applications