The Region Trap Library: Handling Traps on Application Defined Regions of Memory

Tim Brecht Harjinder Sandhu

University of Waterloo

Université York University

www.cs.uwaterloo.ca/~brecht

Problem

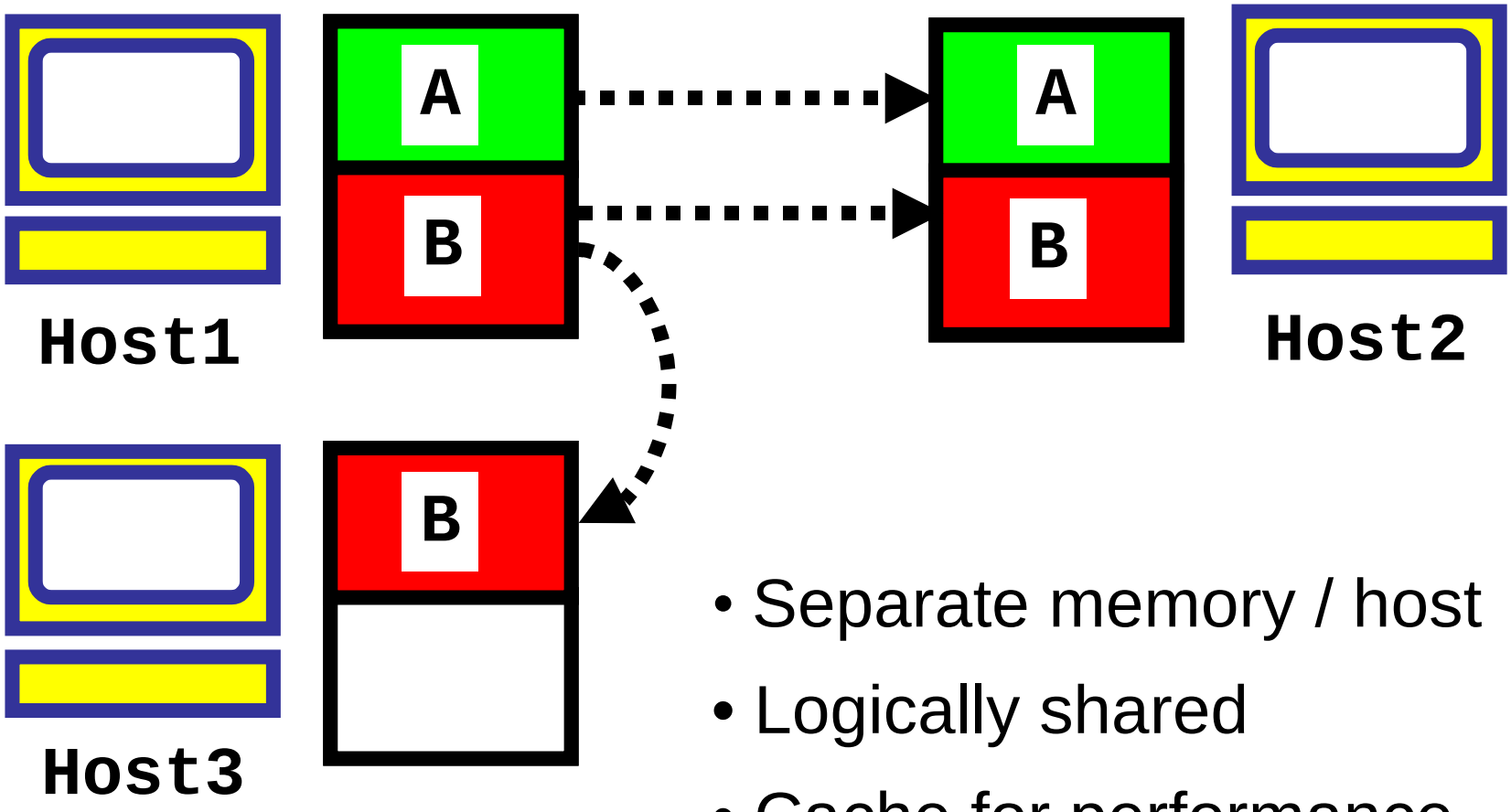
- Mismatch in granularity
  - Page size fixed (OS / Architecture)
  - Data Structure sizes vary (Application)
- Manage data at granularity of application

- Region = contiguous addresses of memory

Application Domains

- Distributed Shared Memory (DSM)
- Persistent Objects / Storage
- Garbage Collection
- Checkpointing

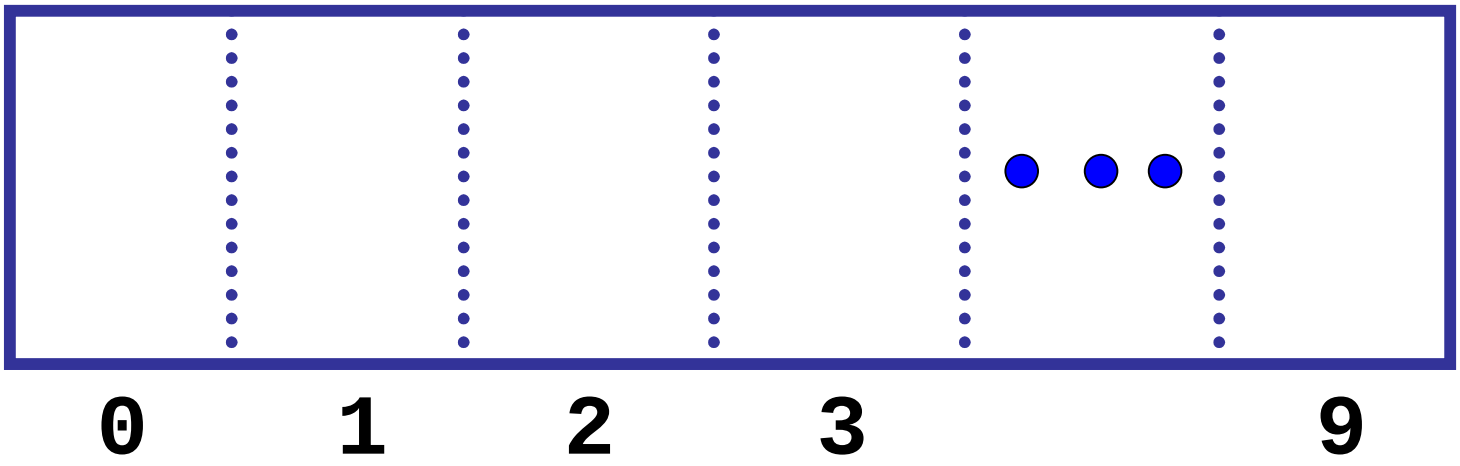
Example Domain: DSM



- Separate memory / host
- Logically shared
- Cache for performance

Data Larger than a Page

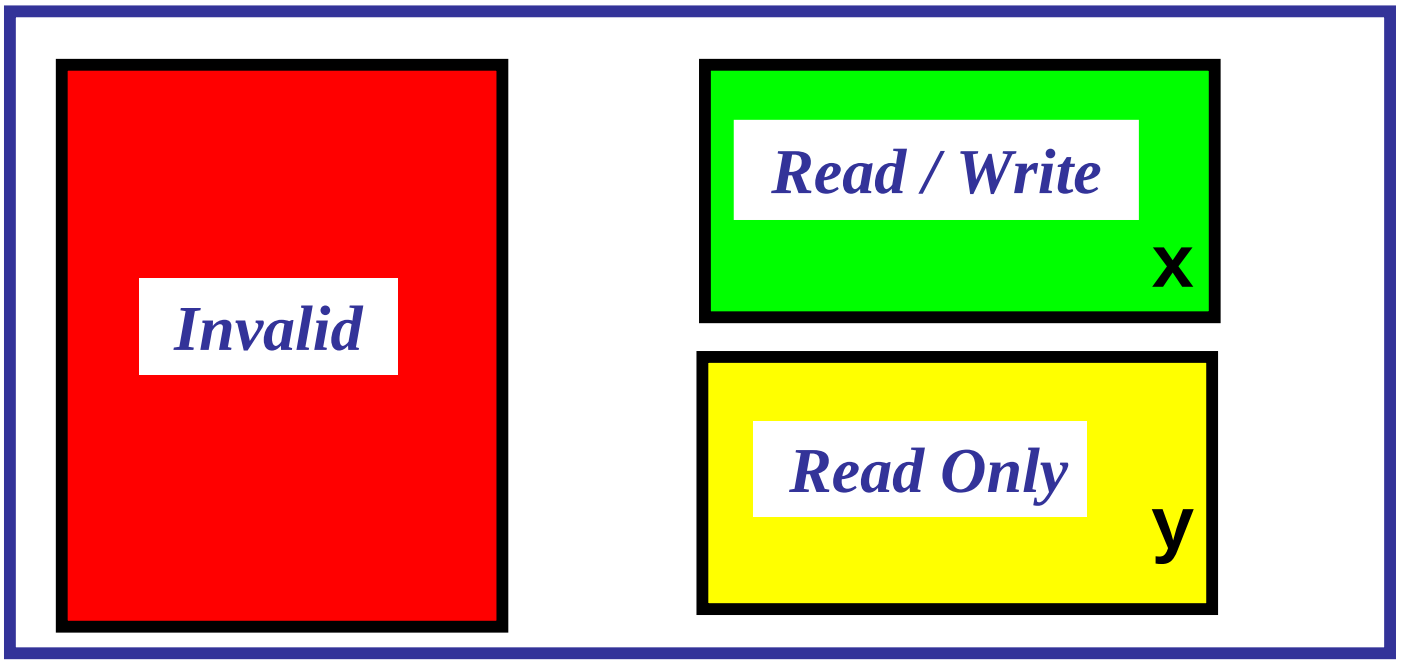
```
char array[40000];
```



- One page fault per page
- Action taken for each page

Data Smaller than a Page

```
char array[40000]; int x; int y;
```



- Can’t control faults when objs on same page
- “FALSE FAULTS” - determine if wanted fault

Region Trap: Goals

- User-level traps on regions not pages
- Similar to VM functions (**`mprotect`**)
- Wide Use:
  - C library *(no lang dependent tricks)*
  - Standard OS *(no kernel mods)*
  - Arch independent mechanisms

    (Linux/Pentium, AIX/PowerPC, Solaris/Sparc)

Requirements

- Control & detect access to regions
- Distinguish read and write traps
- Call handler on access (per region)
- Continue execution

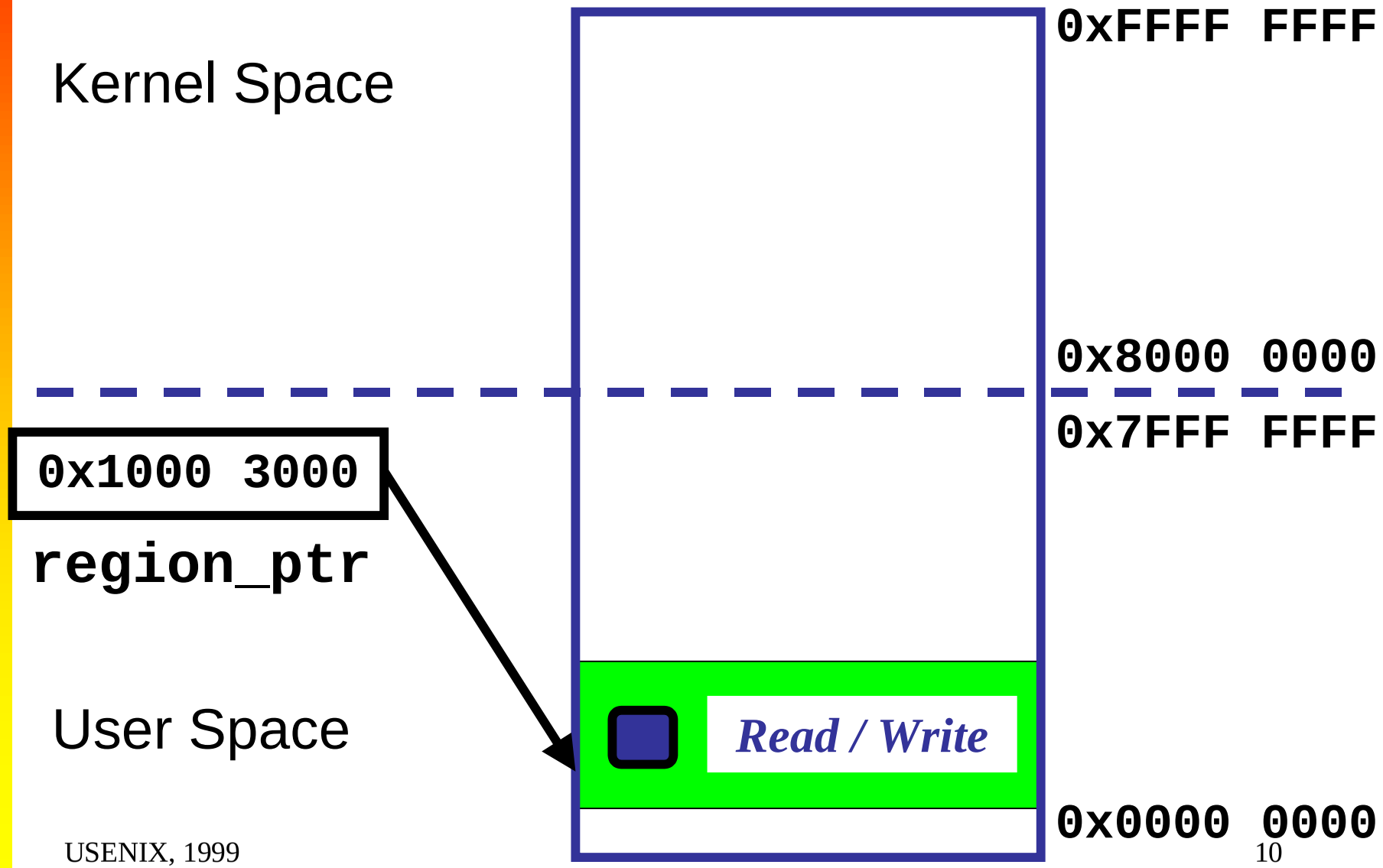
- Can't use **`mprotect`** (page based)
- Trap and call handler ⇨ ? Use signals

Generating a Trap

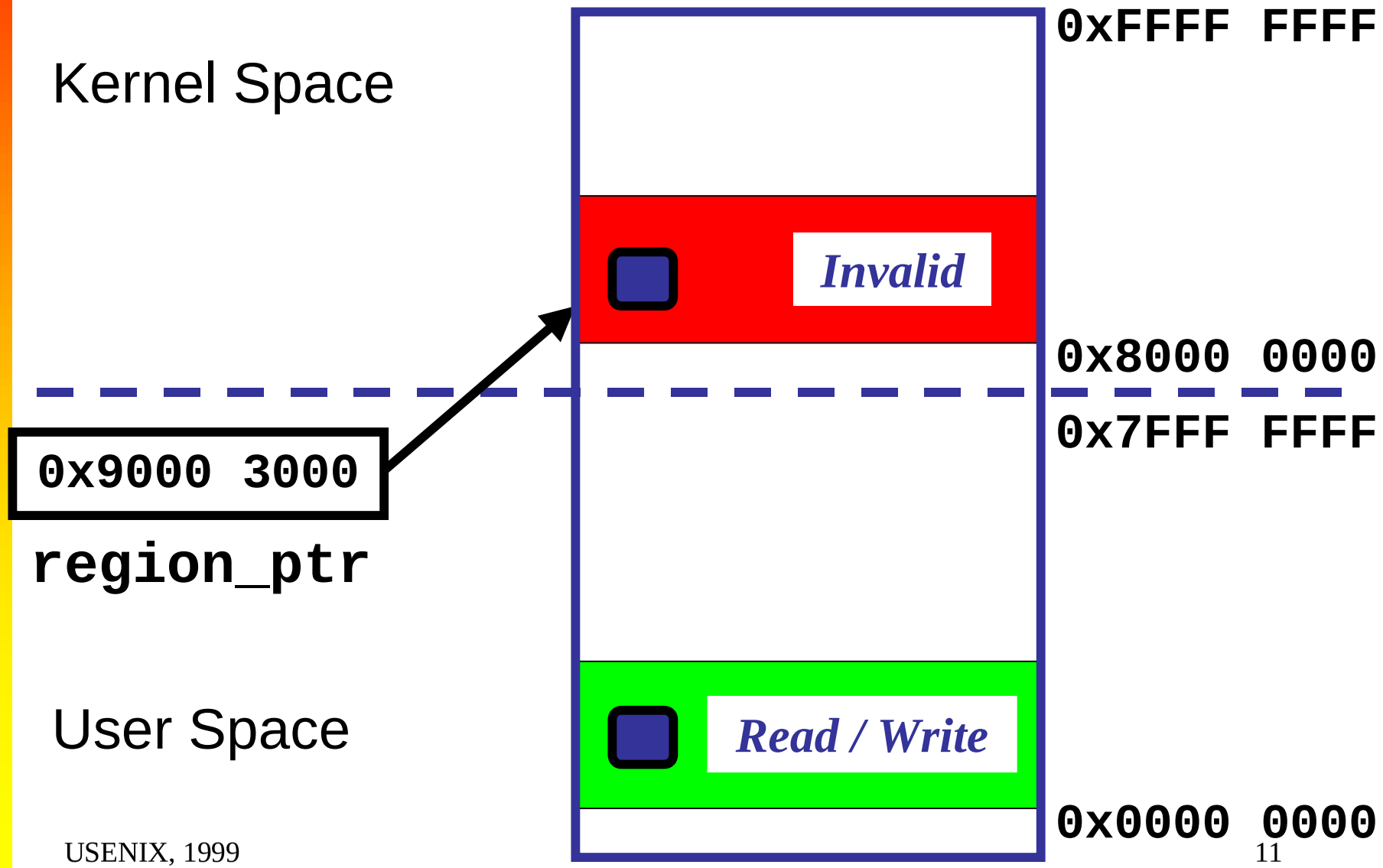
- How to generate **SIGSEGV** ?

```
char *A = malloc(N);

A[0] = ‘a’;
A[1] = ‘b’;
```

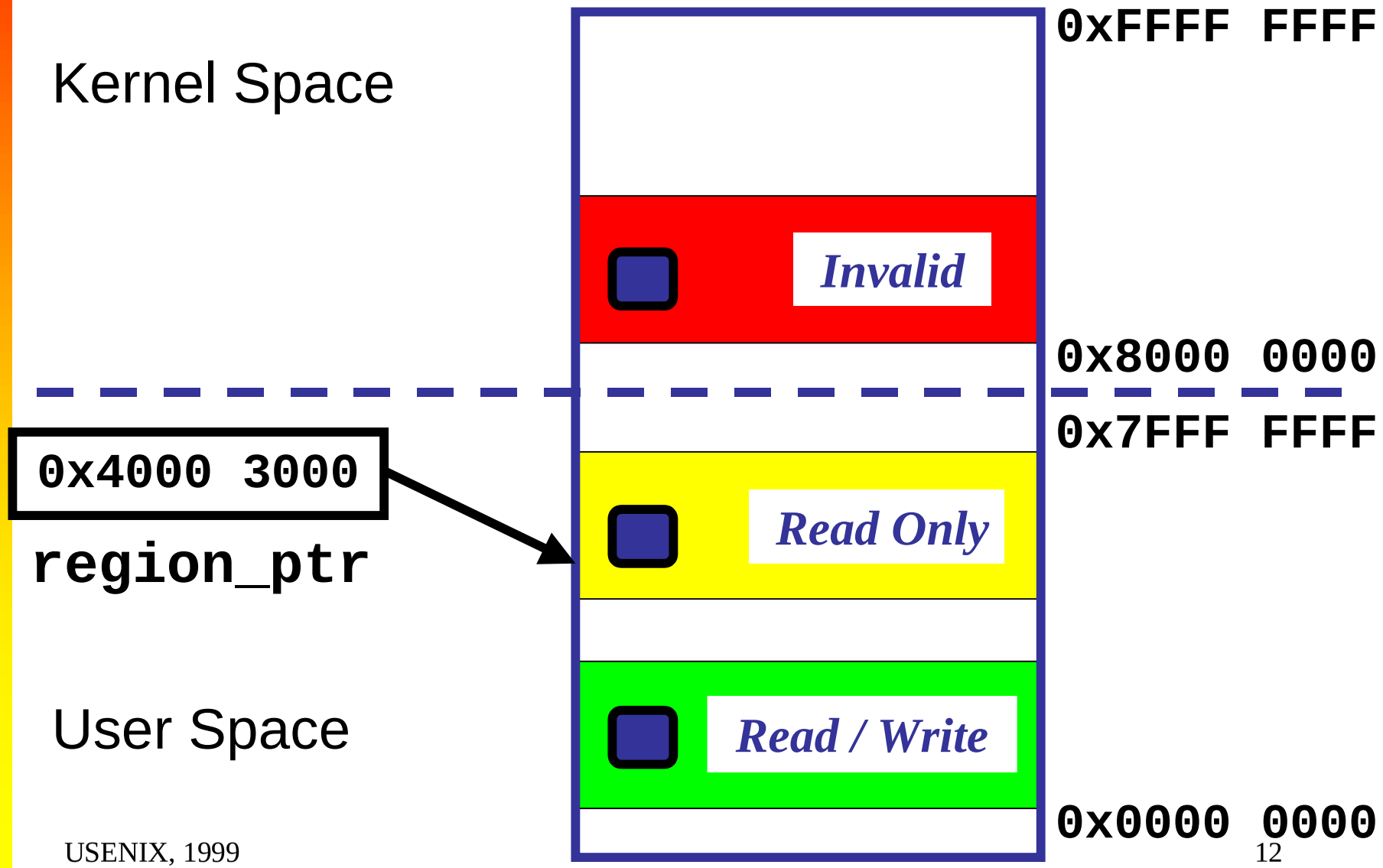
Trapping: Pointer Swizzling



Trapping: Pointer Swizzling



Trapping: Pointer Swizzling



Region Trap Library

```
 1  char A[N], *B, *C;
 2  B = reg_define(A, N, &B);
 3  C = reg_alloc(N, &C);
 4  reg_handler(B, B_handler);
 5  reg_handler(C, C_handler);
 6  reg_protect(B, PROT_NONE);
 7  reg_protect(C, PROT_READ);
 8  B[i] = C[i];  /* B traps */
 9  C[i] = i;     /* C traps */
10  A[i] = i;     /* does not trap */
```

Determining the Region

- AVL tree of regions (start and end addrs)
- If faulting addr lies inside a region

  Find region in AVL tree
- Otherwise real **`SIGSEGV`**

  Stop catching **`SIGSEGV`** and resume
- User can't use sigaction for **`SIGSEGV`**

  `reg_handler(NO_REGION, handler);`

Continue Execution

- Change protection level

  Fix (swizzle) region pointer and aliases

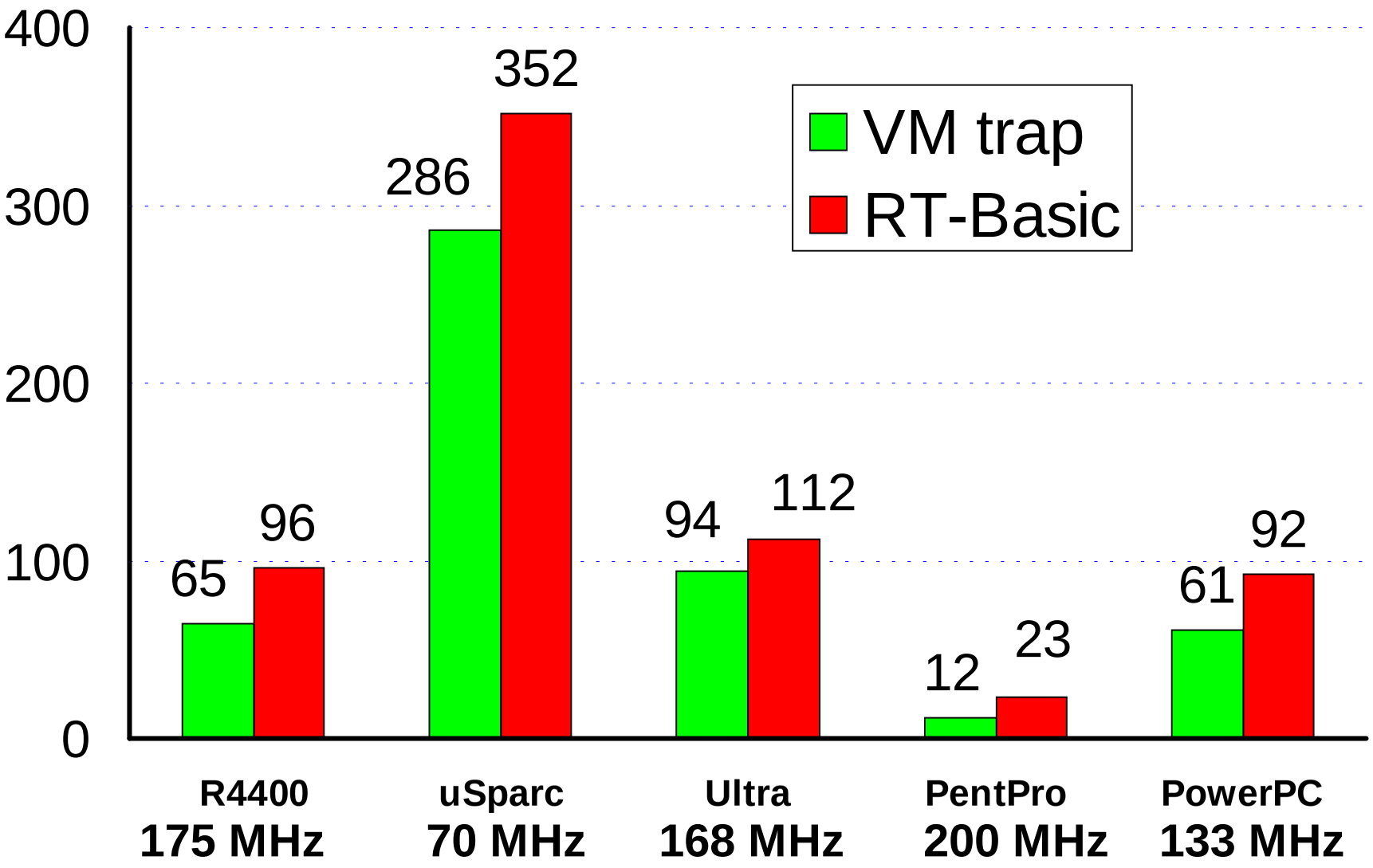
  ```
  ptr = reg_alloc(N,&ptr);
  reg_ptr(&newptr);
  ```

  Fix (swizzle) register

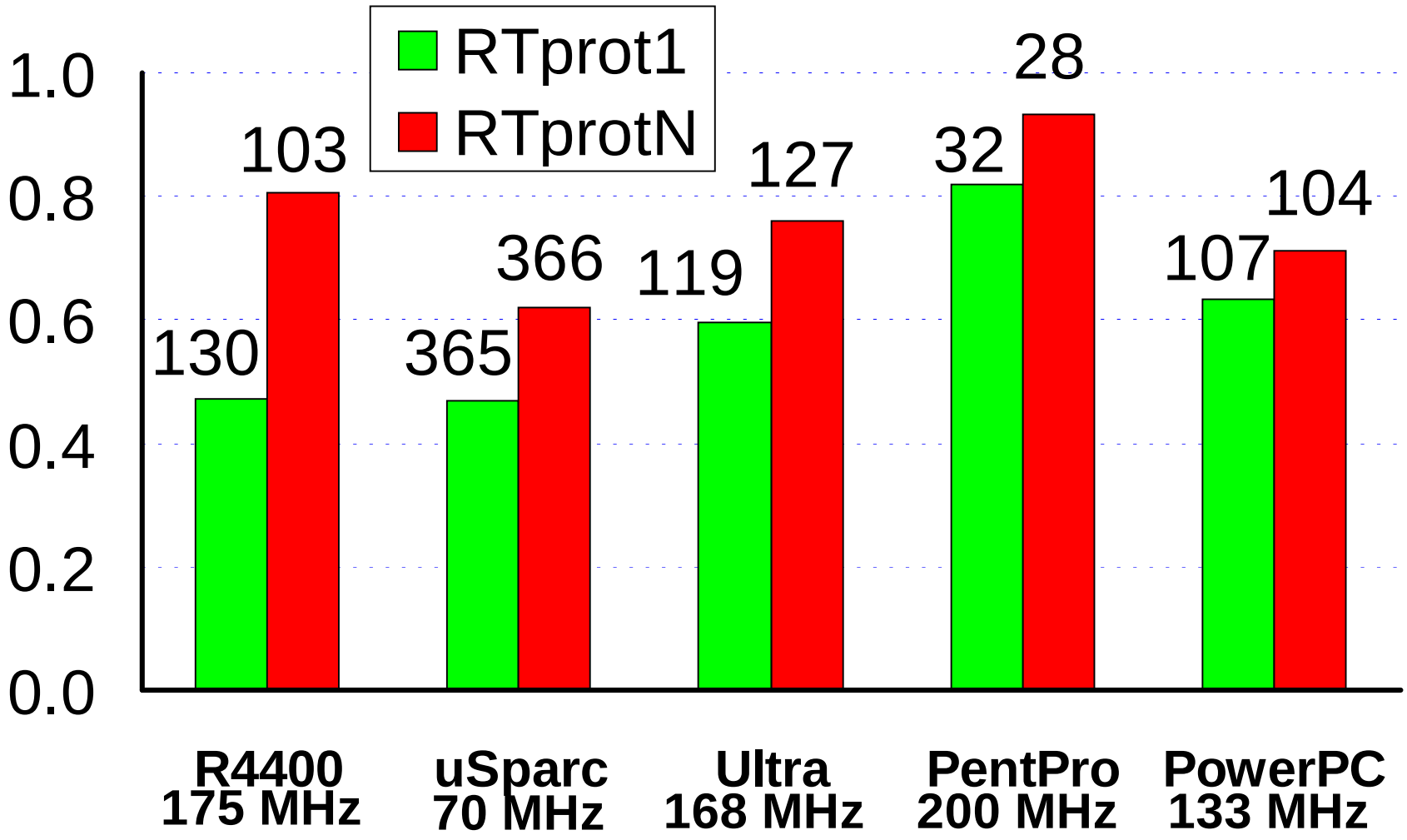
  ```
  ld r2, (r3)    st r2, (r3)
  ```

- Call installed handler with useful params
- Leave sig handler (re-execute instruction)

Micro-Benchmarks



Micro-Benchmarks



Related Work

- Explicit checking [Hoskings & Moss, 93]
  [Zekauskas *et al*., 94] [Scales *et al*., 96]
- Program Annotation
  [Sandhu *et al*., 93] [Bershad *et al*., 93]
- Unaligned traps [Thekkath & Levy, 94]
- MultiView [Itzkovitz & Schuster, 99]

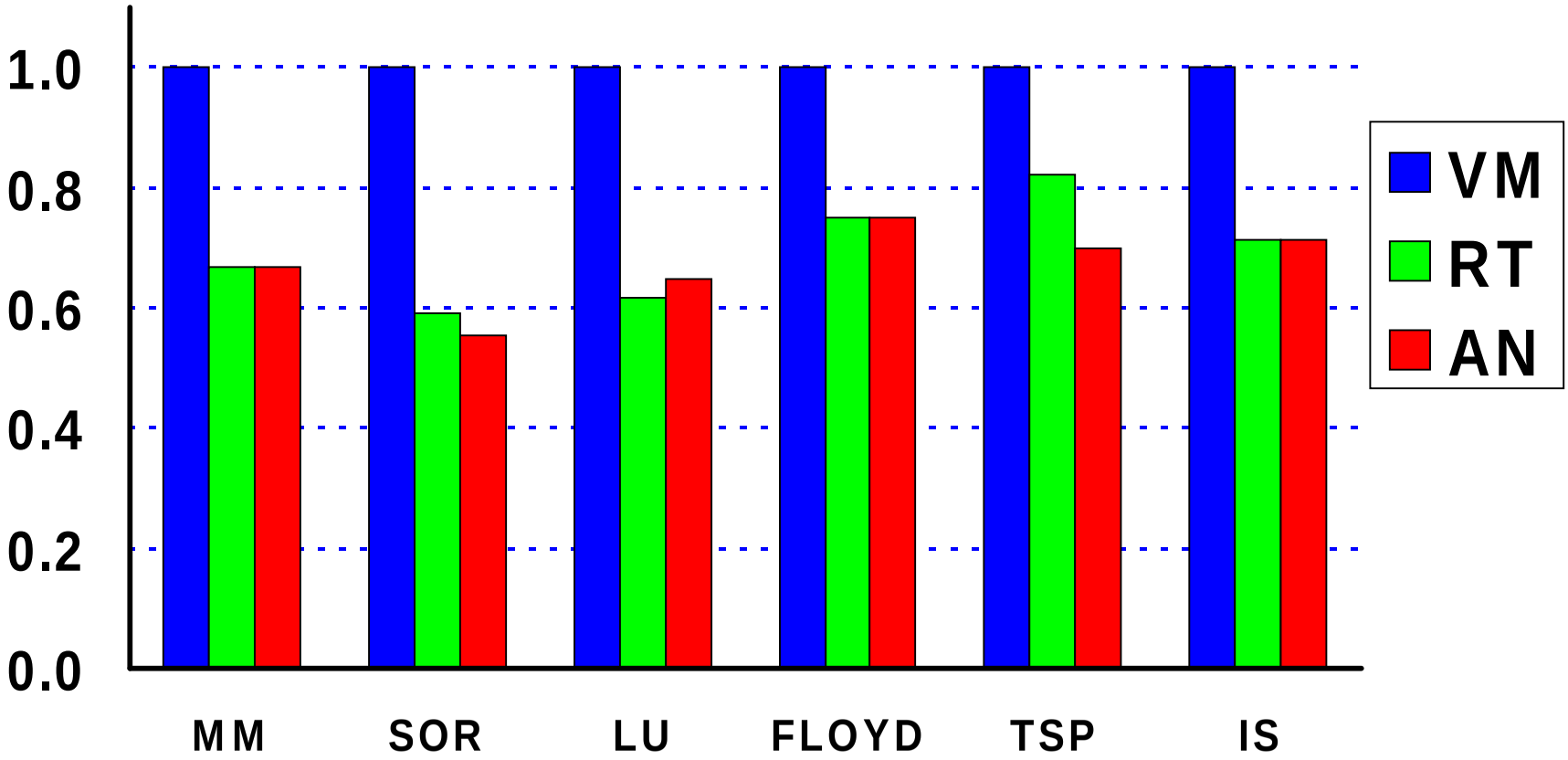
Example with Annotation

```
1  for (i=0; i<N; i++)
2  ReadAccess(B[i]);

3  for (i=start; i<end; i++) {
4    ReadAccess(A[i]);
5    WriteAccess(C[i]);
6     for (j=0; j<N; j++)
7      for (k=0; k<N; k++)
8       C[i][j] += A[i][k] * B[k][j];
9  }
```

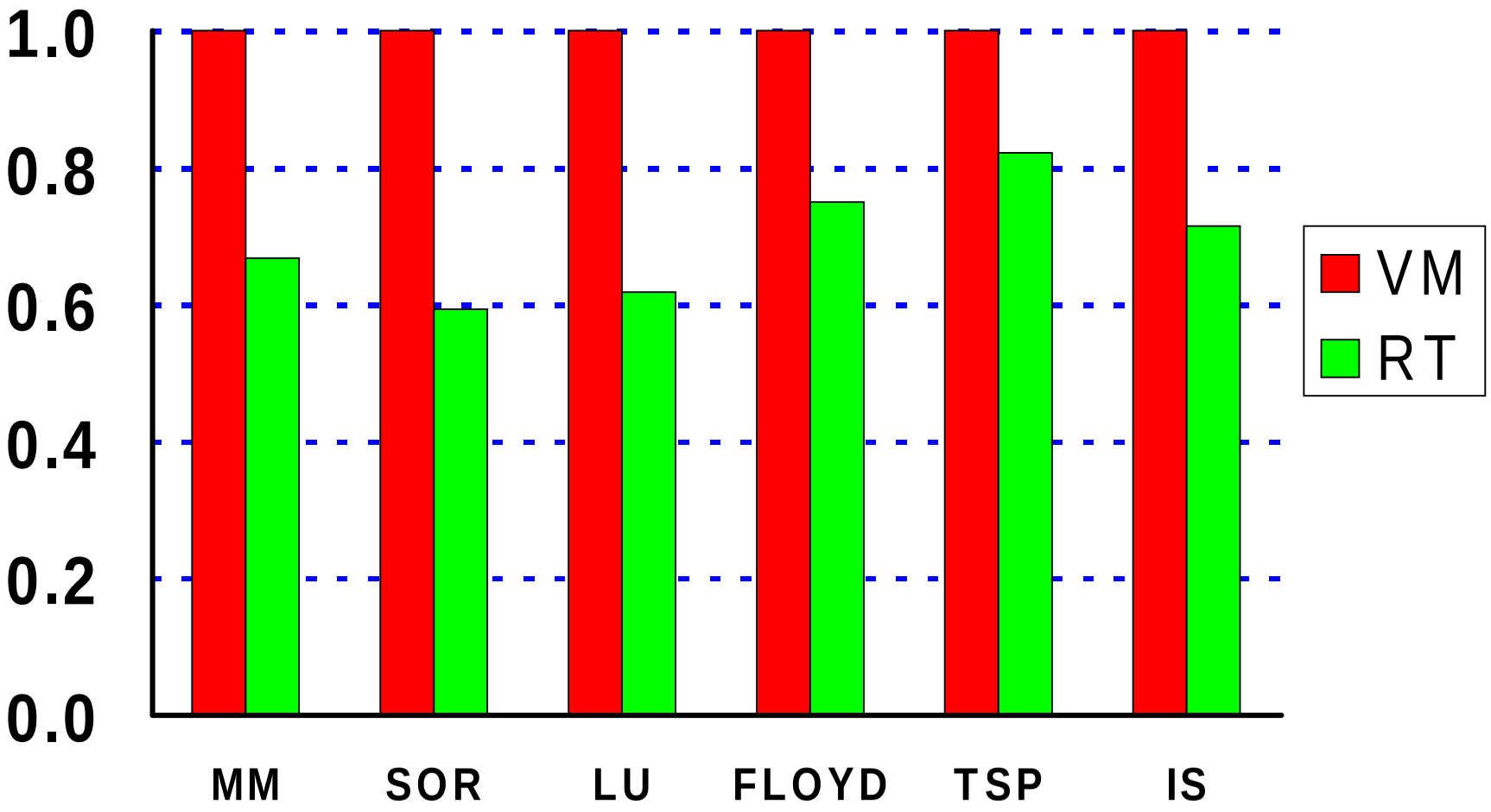
Example without Annotation

```
1  for (i=0; i<N; i++)
2  ReadAccess(B[i]);

3   for (i=start; i<end; i++) {
4     ReadAccess(A[i]);
5     WriteAccess(C[i]);
6       for (j=0; j<N; j++)
7         for (k=0; k<N; k++)
8           C[i][j] += A[i][k] * B[k][j];
9   }
```

Application Performance



Application Performance



Conclusions

- Control / detect access to independent regions
- Machine speed access to unprotected regions
- RTL works on many Architectures and OSes
- Prototype implementation quite efficient
- Significant performance gains for some apps

http://www.cs.uwaterloo.ca/~brecht