Taskserver Design

Jan Tepelmann Marcel Noe

System Architecture Group
Universität Karlsruhe (TH)

System Design & Implementation, 2008

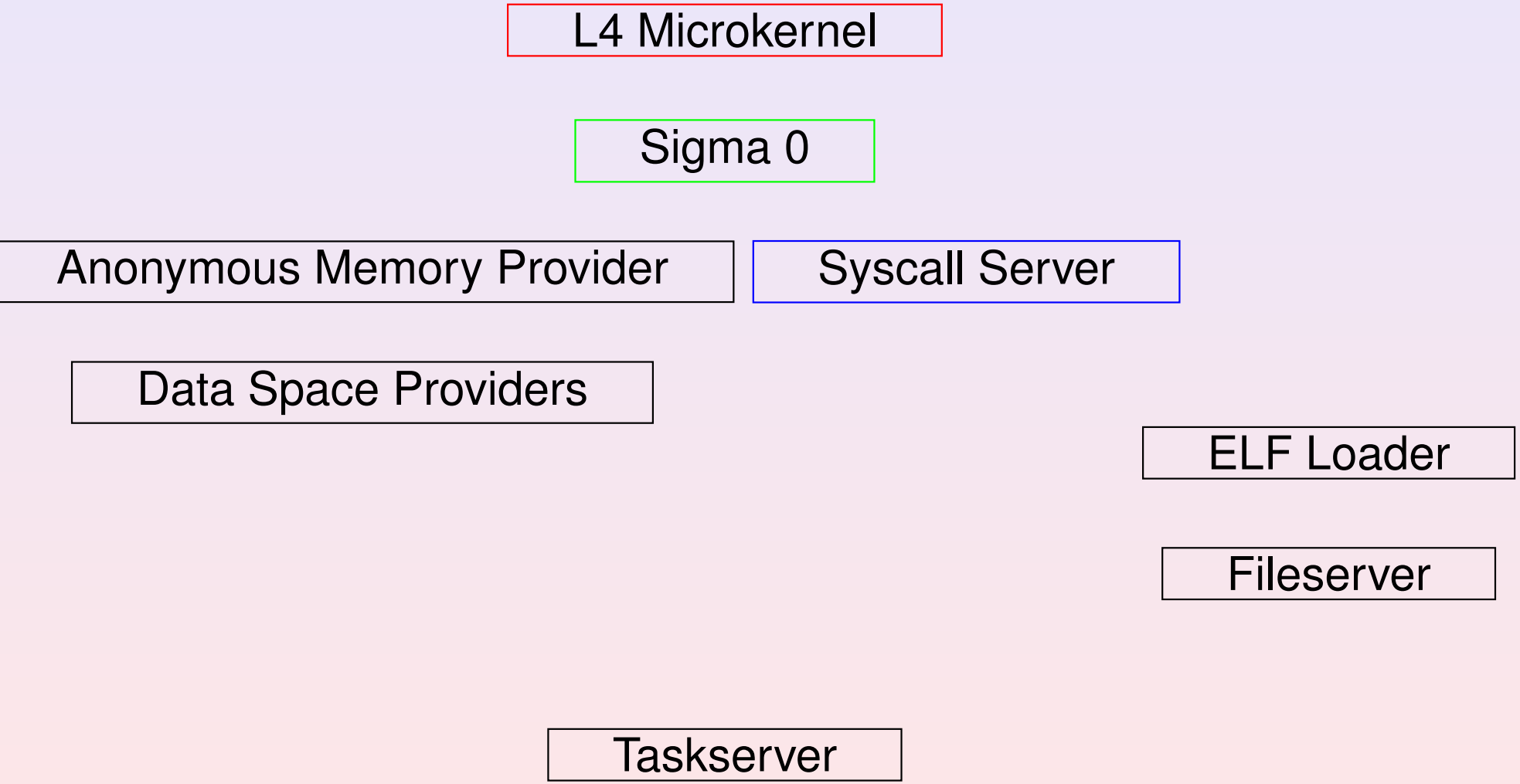
Outline

Outline

Outline

Outline

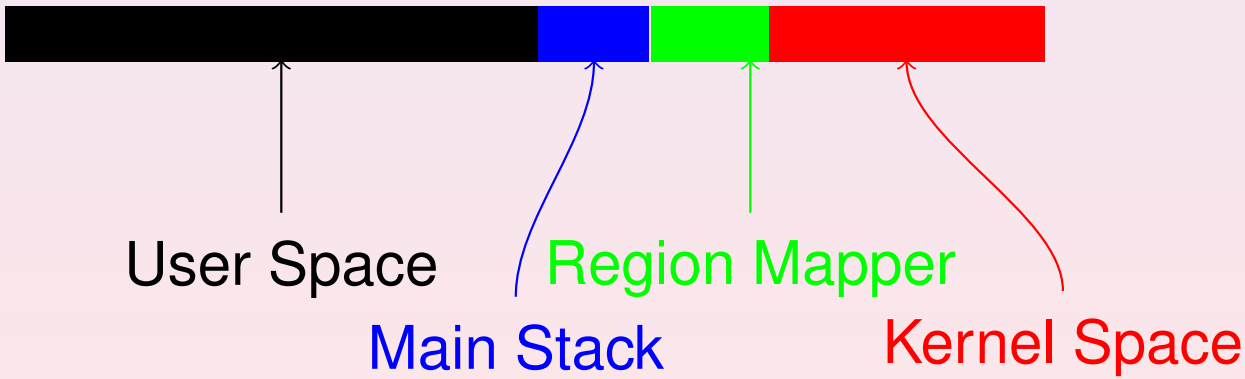
System Components



Outline

Sawmill Inspired Data Spaces

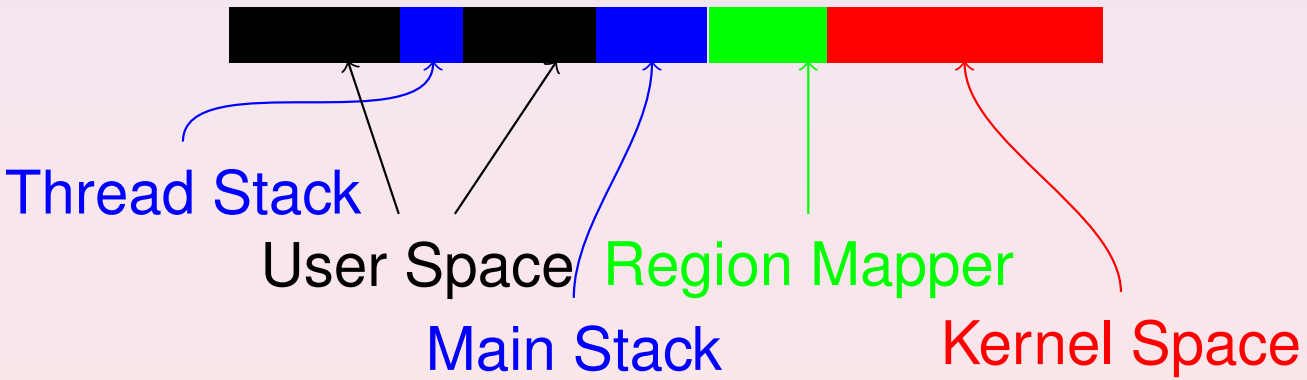
- Every address space has got it's own managing thread, called *region mapper*.
- *ragion mapper* resides at the end of user address space, just below kernel.
- *ragion mapper* holds mapping between *VM Region* and *Data Space Provider*



Outline

Stack Positioning

- *Main program stack* is created just below *Region Mapper*, growing down, towards *heap*
- For every additional thread, *stack space* is allocated from *heap*, surrounded by *read only pages* to detect overflow
- *Thread stacks* are created by *region mapper*



Outline

New

```
L4_ThreadId_t New(in String path, in String args);
```

- *Task server* asks *memory server* to create a new *address space*
- *Task server* creates a new *region mapper* inside new *address space*
- *Task server* sends message to *Region mapper*, telling it the path of the image to load
- *Region mapper* asks *ELF-Loader* (or *PE-Loader* or whatever) to map image into its *address space*
- *Region mapper* starts mapped program inside new thread

Fork

```
L4_ThreadId_t Fork();
```

- *Task server* asks *memory server* to create a new *address space*
- *Task server* creates a new *region mapper* inside new *address space*
- *Task server* asks *memory server* to map old *User space* and *Stack* into new *address space*
- *Task server* sends message to *region mapper*
- *Region mapper* resumes operation in new address space

Exec

```
L4_ThreadId_t Exec();
```

- *Task server* kills alll *threads* inside address space except *region mapper*
- *Task server* sends message to *Region mapper*, telling it the path of the image to load
- *Region mapper* asks *ELF-Loader* (or *PE-Loader* or whatever) to map image into its *address space*
- *Region mapper* starts mapped program inside new thread

Kill

```
Void Kill(in L4_ThreadId_t tid);
```

- If *TID* is a region mapper: Kill all *threads* in *address space*
- Else kill *thread* specified by *TID*

StartThread

```
L4_ThreadId_t StartThread(in L4_Pointer_t ip, in
L4_Pointer_t sp, in L4_Word_t stackSize);
```

- *Task server* tells *syscall server* to create a new *thread* inside specified *address space*
- *Task server* tells *region mapper* to start thread
- *Region mapper* creates *thead stack* and sends start message to *thread*

Outline

SetStatisticInterval

```
void SetStatisticInterval(in unsigned int
interval);
```

- Sets interval in which statistics are collected

SetTimeslice

```
void SetTimeslice(in unsigned int timeslice);
```

- Sets length of timeslice

SetPriority

```
unsigned int SetPriority(in unsigned int
priority);
```

- Sets priority of *thread* identified by *TID*

SetPreemptionDelay

```
L4_Word_t SetPreemptionDelay(in L4_ThreadId_t
tid, in L4_Word_t sensitivePrio, in L4_Word_t
MaxDelay);
```

- Sets preemption delay of *thread* identified by *TID*

Ping

```
unsigned int Ping(in L4_ThreadId_t tid);
```

- Sends *ping message* to *region mapper* of *address space* to check if it is still responsive

Outline

Statistics

- Collected statistics are accessed via virtual filesystem

Questions?

- Please feel free to ask *questions* or give *comments*!