Flexible Event Handling for an Extensible OS

Przemyslaw Pardyak

the SPIN group
Department of Computer Science and Engineering
University of Washington, Seattle, USA

SPIN

- extensible microkernel
- application specific services
- user's code downloaded into the kernel

Extensibility in SPIN

- dynamic linking
- type-safe language
- event-based interactions
- interfaces to internals

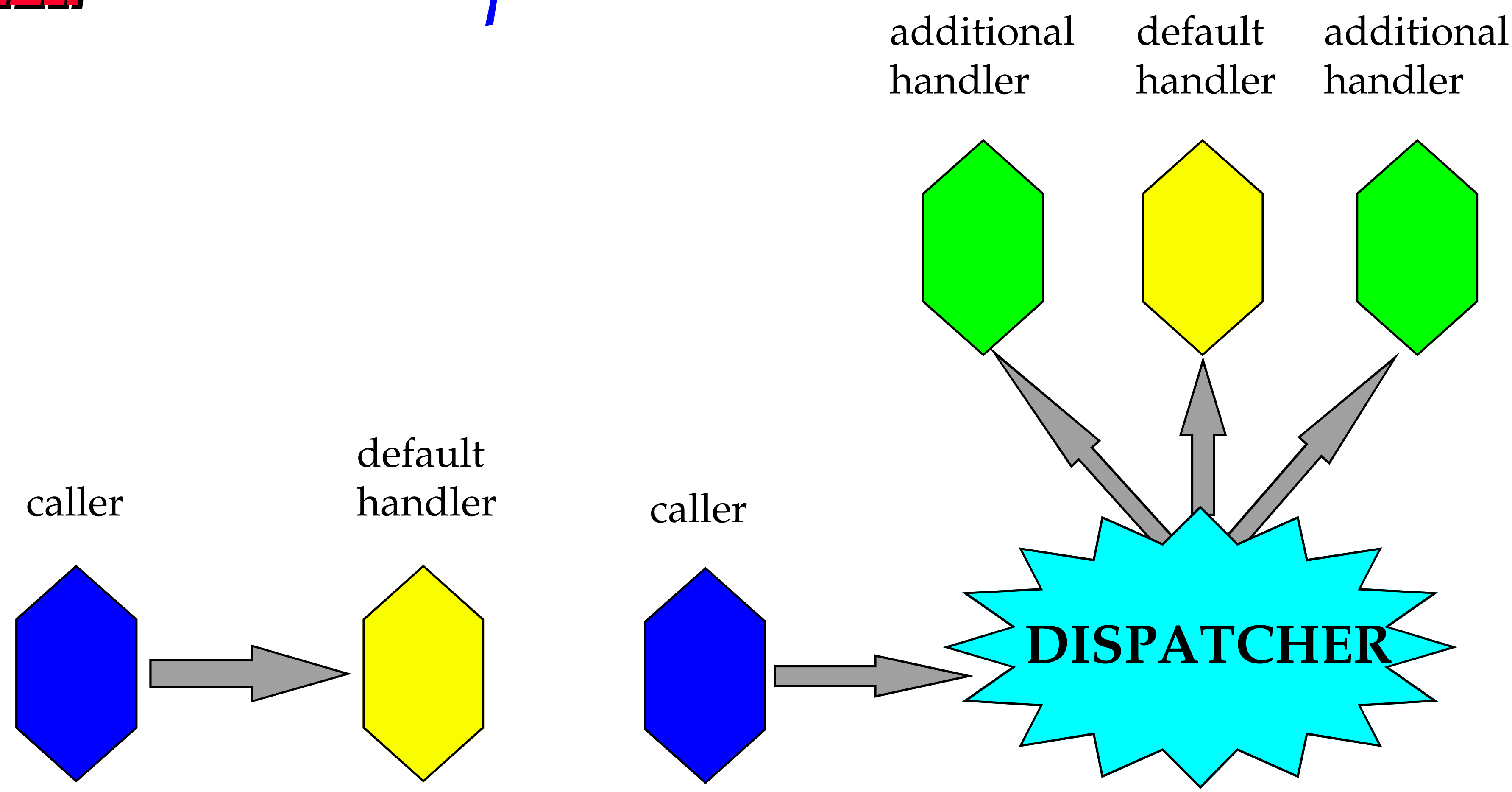
Events for extensibility

- name space for extensions
- indirect invocation
- separation of callers and callees
- late binding
- any number of callees (multicast)

Event handling

- events are raised to announce a change of state or to request a service
- extensions installed as event handlers
- handlers executed when the right event is raised
- execution may be conditional on event arguments and system state

Event dispatcher



Default handler only

Dispatched event raise

Performance issues

- problem:
  - inherent cost of flexibility
- solution
  - pay only if use it

How?

- remove it
- isolate it
- hide it

Implementation of the dispatcher

- dispatcher invoked only when necessary
- no cost in the default case
- optimized for a fixed number of known handlers
- event specific dispatching
- when clause optimizations

Example extension

event definition/default handler :

PROCEDURE *VM . PageFault* (*p* : INT)

a when clause:

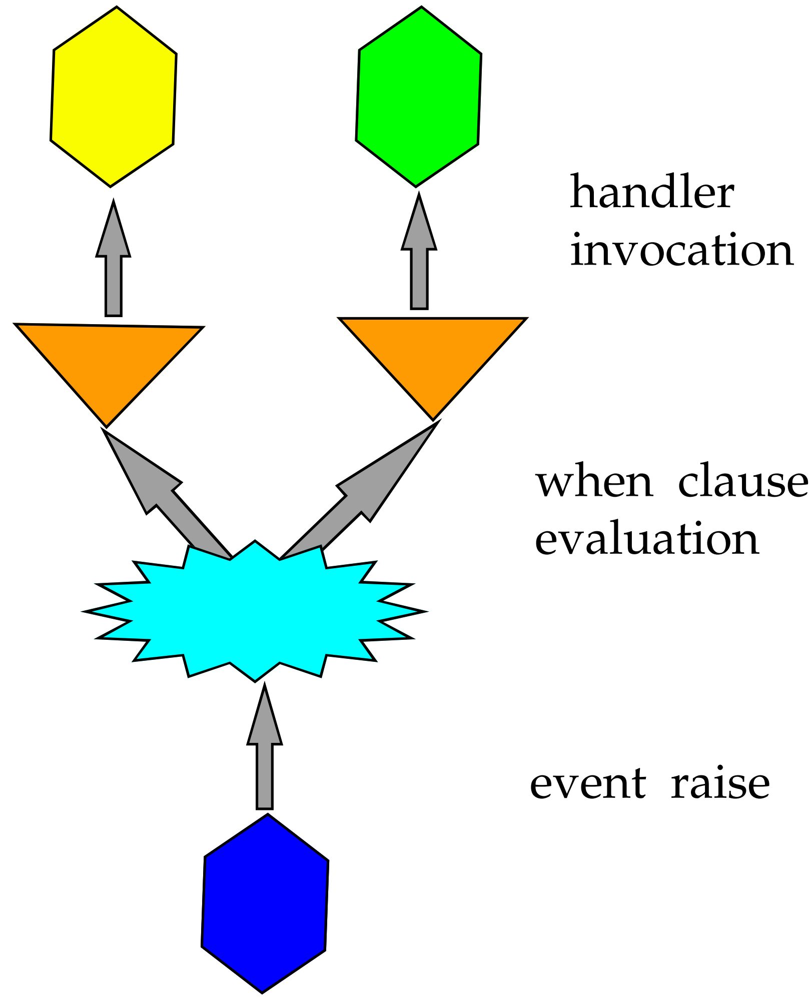
```
PROCEDURE whenClause (p: INT) : BOOL
   RETURN p IN myPages
END
```

a handler:

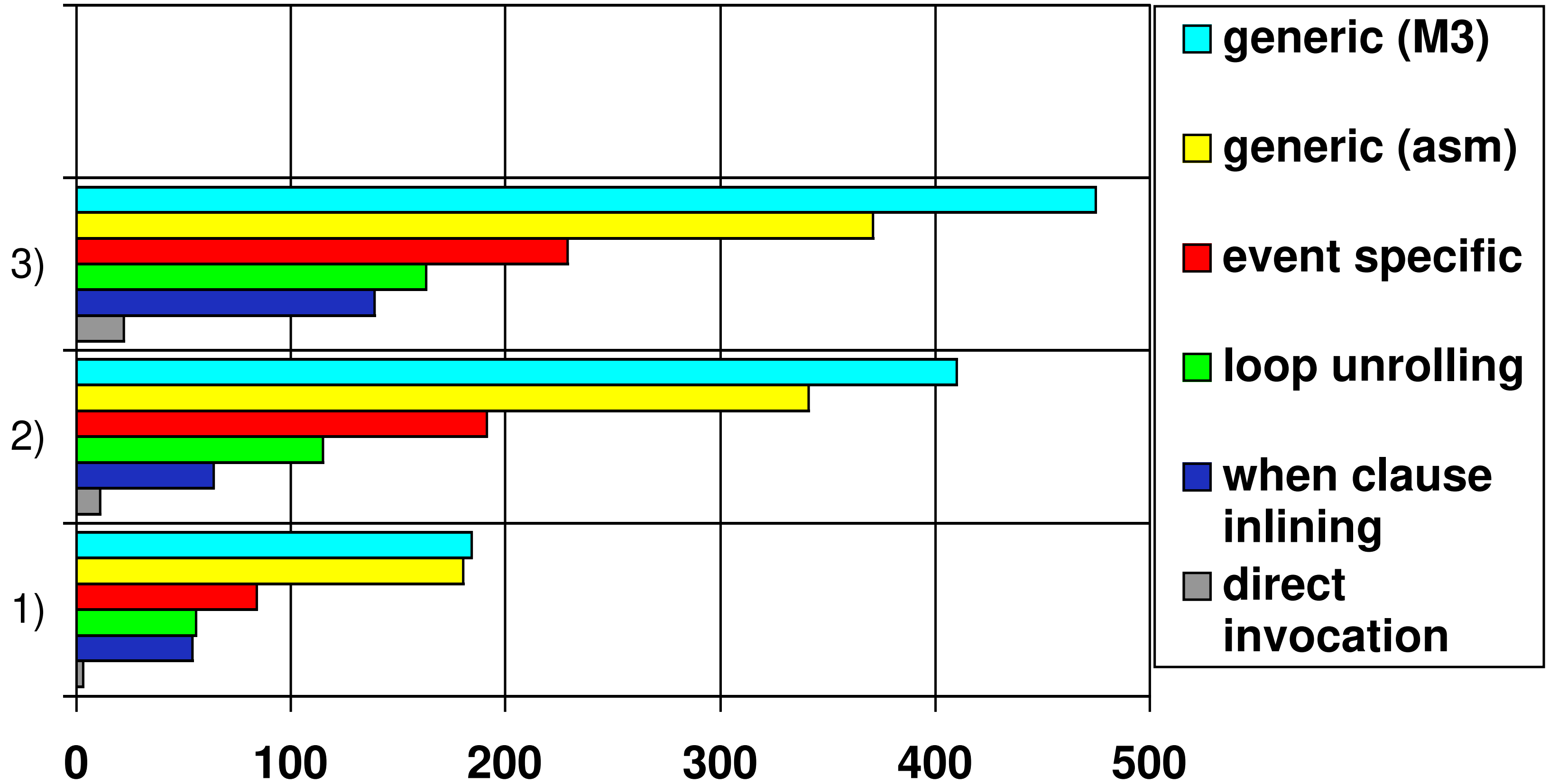
```
PROCEDURE myHandler ( p : INT )
   . . .
END
```

to install a new handler :

```
Dispatcher.Install ( VM.PageFault,
                     whenClause,
                     myHandler) ;
```



Initial measurements



measured on DEC 3000/300
all values in machine cycles

1) time to reach the first handler
2) total time with 2 handlers
3) total time with 4 handlers

Status

- first stage
  - side to side with Mach
  - threads, VM, IPC, events
- extensible from the first day!!!
- e-mail: pardy@cs.washington.edu