

Fragment the Heap!

...let the compiler / VM implementors
deal with fragmentation!



Dr. Fridtjof Siebert
CTO, aicas
ISMM 2008, Tucson, 8. June 2008

Split Heap in small fixed-size blocks

Allocate only these blocks

- smaller objects use one block
- larger objects use graph of blocks

Let the VM/compiler emit code to access these objects!

Objects and Arrays

Block 1 (Header)

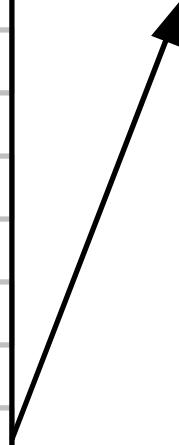
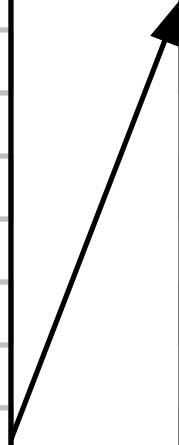
head
type
monitor
field 1
field 2
field 3
field 4
next

Block 2

field 5
field 6
field 7
field 8
field 9
field 10
field 11
next

Block 3

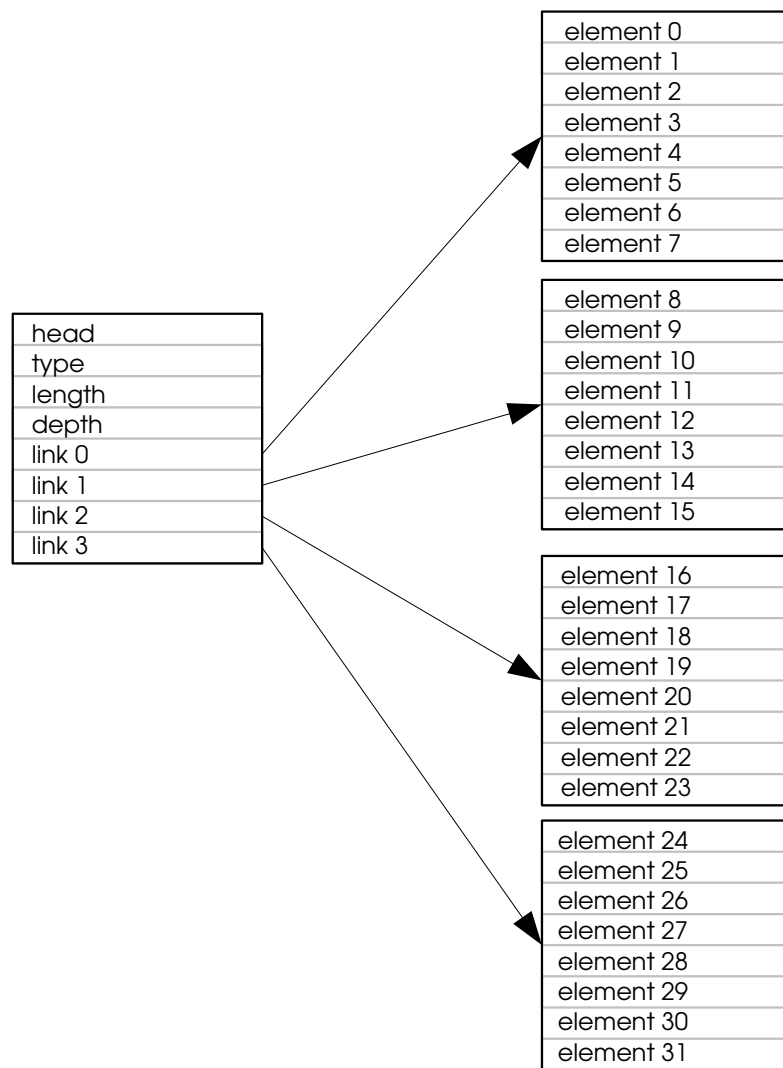
field 12
field 13
field 14
field 15
—
—
—
—



Objects and Arrays

head
type
length
depth
element 0
element 1
element 2
element 3

Objects and Arrays



element 97
element 98
element 99
element 100
element 101
element 102
element 103

element 104
element 105
element 106
element 107
element 108
element 109
element 110
element 111

element 112
element 113
element 114
element 115
element 116
element 117
element 118
element 119

element 120
element 121
element 122
element 123
element 124
element 125
element 126
element 127

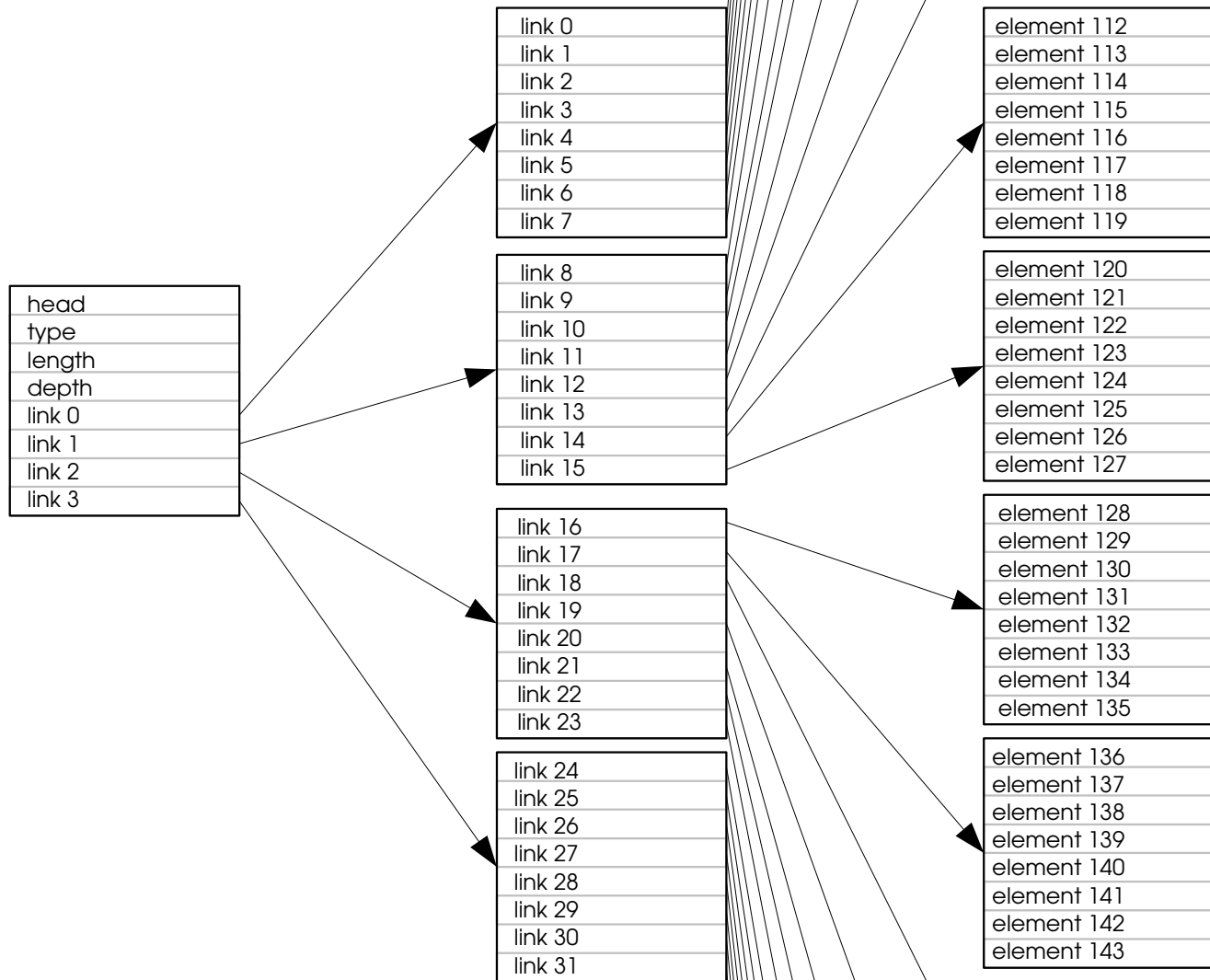
element 128
element 129
element 130
element 131
element 132
element 133
element 134
element 135

element 136
element 137
element 138
element 139
element 140
element 141
element 142
element 143

element 144

Fragment the Heap!

Objects and Arrays



Pro's

GC

- no compaction phase
- unique GC mark & sweep steps
- no tricks (pinning, arraylets, etc.)
- no coalescing
- only one free list

VM compiler

- no handles, forwarding ptrs, ...
- no pointer updates
- alloc never fails due to fragmentation

Pro's

Parallel systems

- very simple free list
- no locks, free list modifications via CAS
- simple stealing, blancing, etc. between processors
- David Bacon: Look at pathological case!



Any relevant Con's?