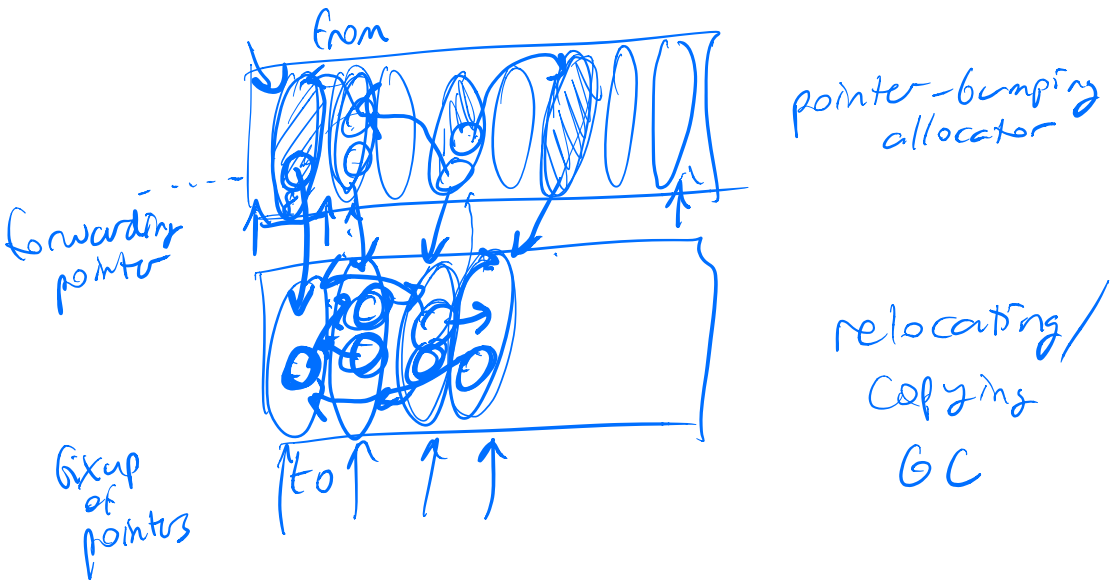
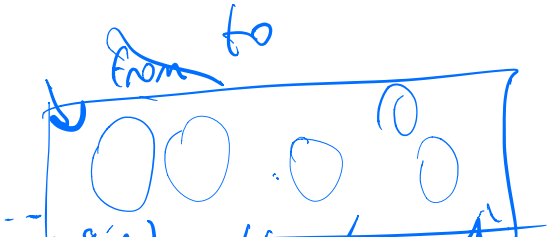
GC
FS

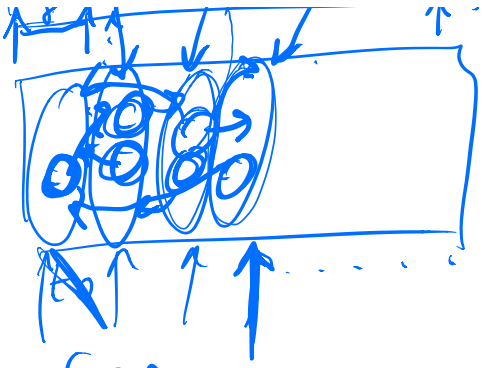
ref counting RC

mark-sweep MS

semispace SS







- 2x memory

From

+ implicitly COMPACTING the heap

+ reduces fragmentation

+ increases locality

- copying takes cycles

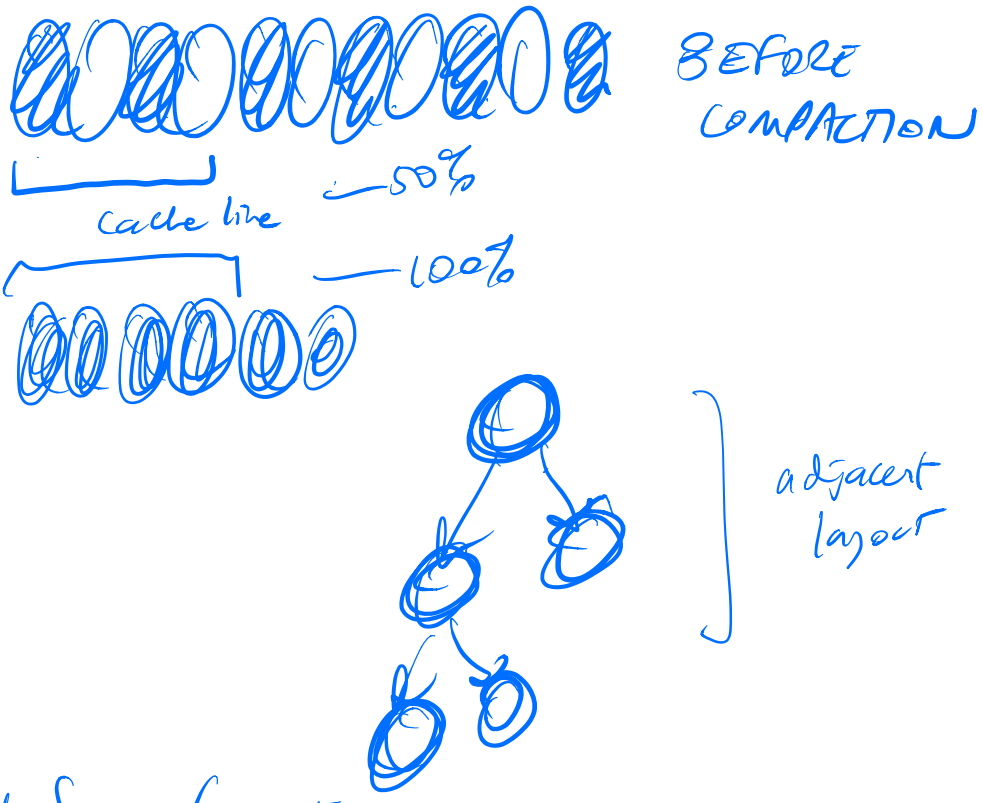
STW

STOP THE WORLD

- GC has collateral damage (locality)

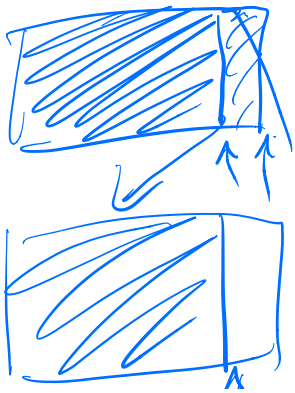
MSC conc.

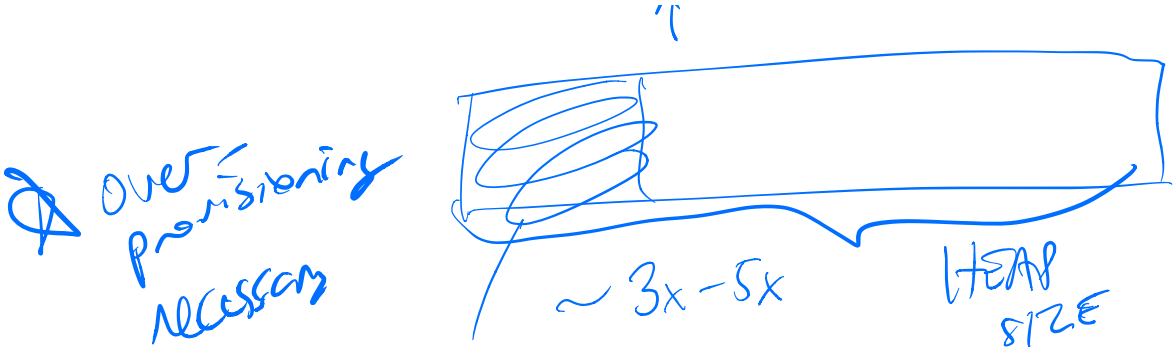
no slowdown (cache warmup)



Mark Sweep Compact

Generational Garbage Collection



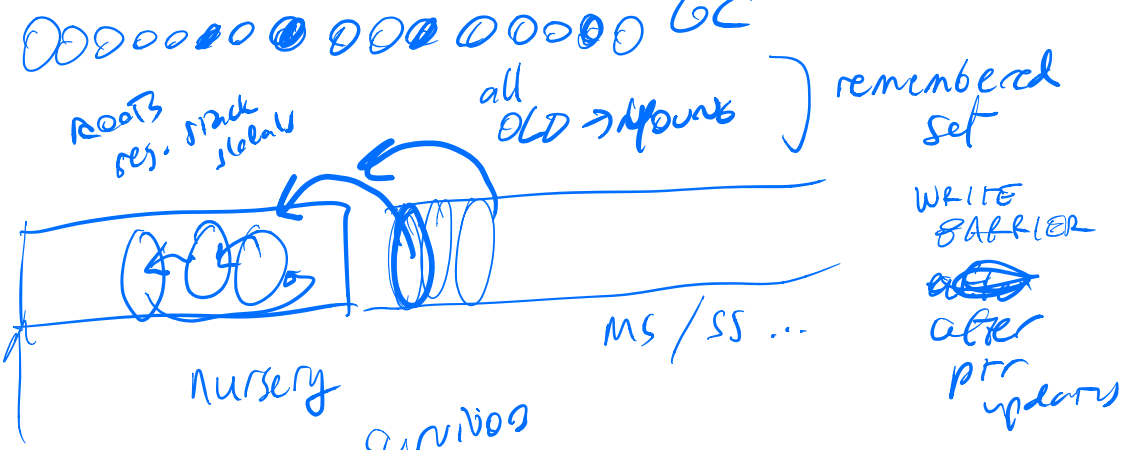


David Ungar
Craig Chambers — Self

pervasively OOP

dynamic compilation
JIT

GC

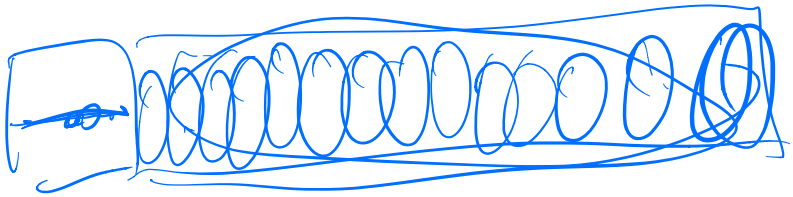


gen. hyp. — most OBJECTS DIE YOUNG

Immortal objects
pretenured — SPECULATED to be immortal

[MAP (F, ◯) ⇒ new objects
REDUCE (OP, ◯)

off heap



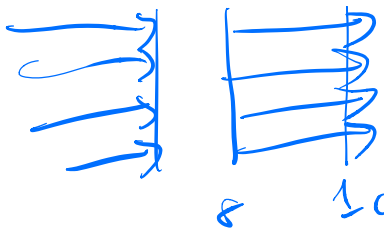
"filter" reduces rate of Full GCs

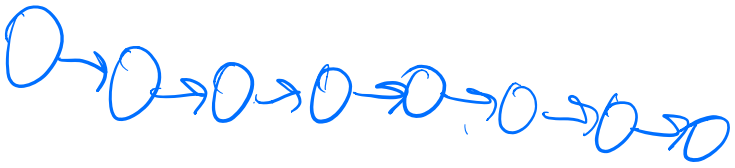
0s
|
Newgen GC

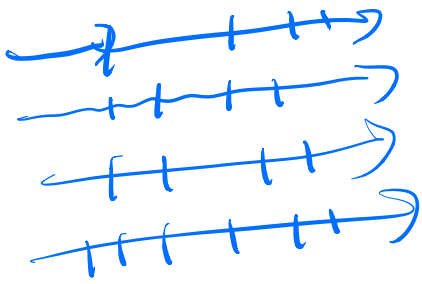
|
GEN

|
FULL STW GC
MS/SJ

GC — parallel
1 → ||||||





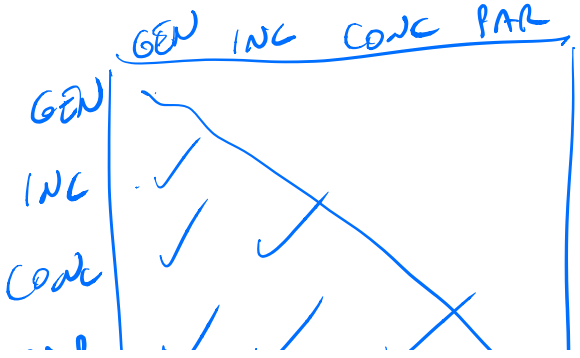


parallel
Concurrent GC
incremental GC

Eliot Moss
Rick Hudson

Sapphire
"Train" – Mature Object Space

incremental –
do GC a bit at a time



Conc –
no STW
program keeps
running
"mutator"

parallel – GC

for(i=0; ——→ {

x = new Foo

a[i] = x

}

escape analysis

static GC

FreeMe | compile-time GC

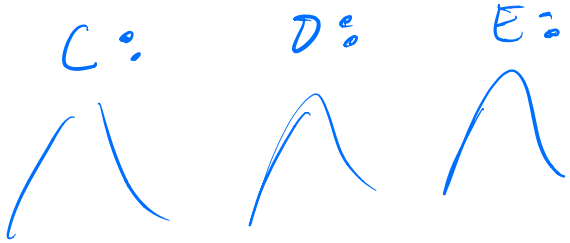
hierarchical file system

flat file system

HFS

DOS

UNIX VMS



POSIX FS

Creat

Open

Java C++

| /

POSIX FS API

close
read
write

OS

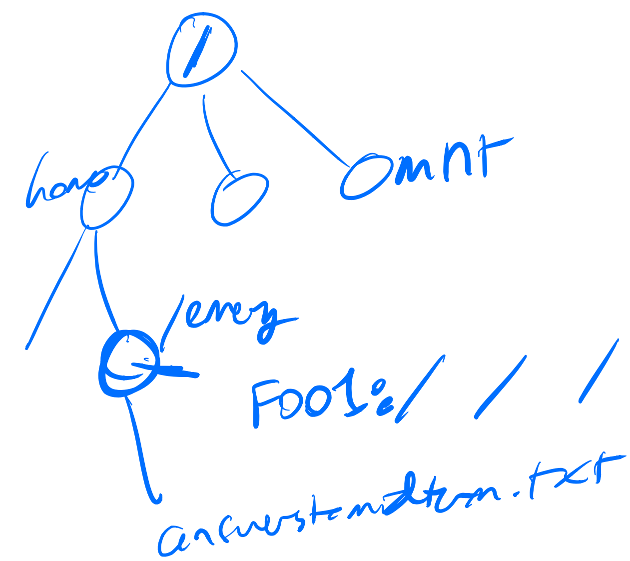
NTFS
FAT32
HFS+

ext3
ext4
ZFS

XFS
ReiserFS

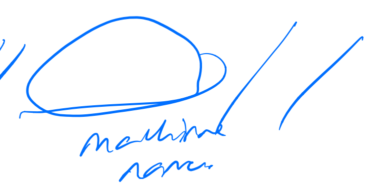
NFS

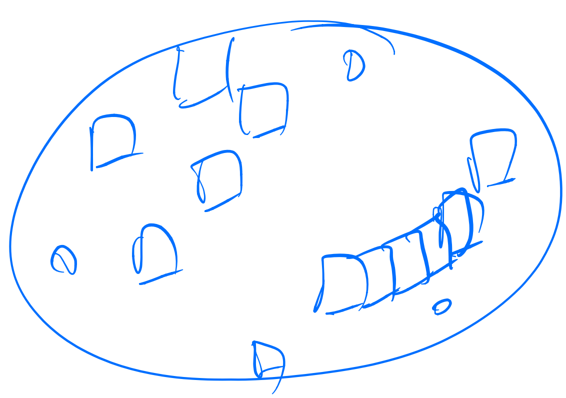
networked
file system



mount
points

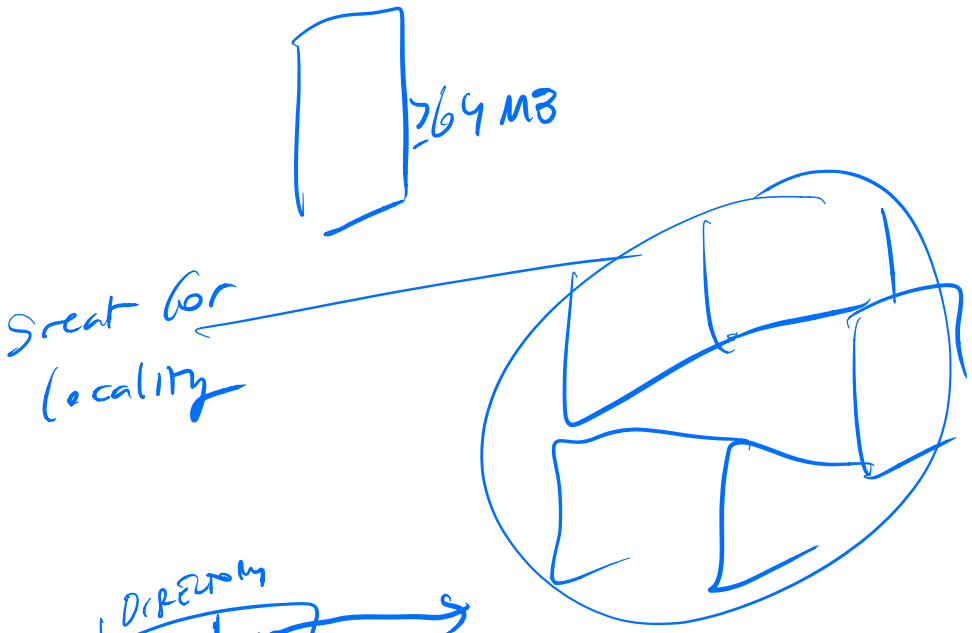
"http fs"

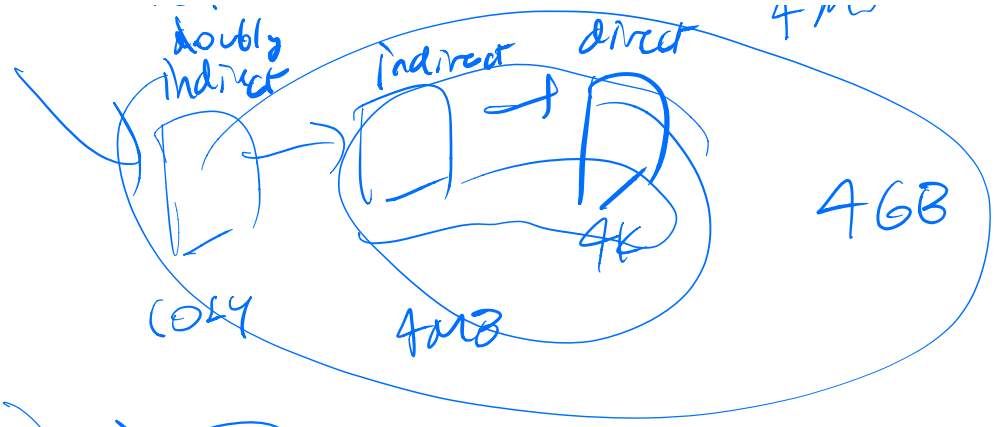
http://


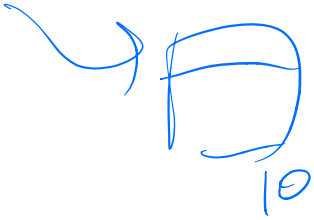


GFS

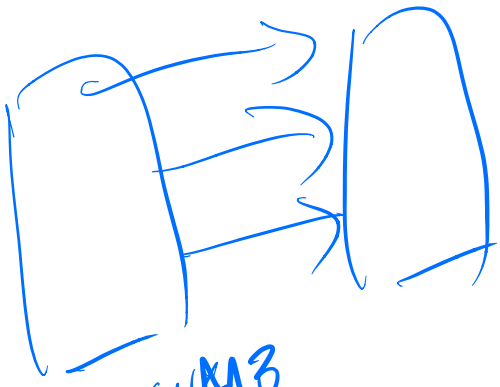
hdfs


~64 MB
Great for
locality
Directory
fn
4K
4K
direct block
indirect
4K
direct
4 bytes
4K
~1000
1024
4K





1024 · 4GB 4TB



64MB

16,000 * 64 MB

1024 TB

leases

chunkservers

"append only"