Dressing up data for R

Hannes Mühleisen

DSC 2017

Problem?

- People push large amounts of data into R
- Databases, Parquet/Feather …
- Need native SEXP for compatibility
- R has no abstraction for data access
  - `INTEGER(A)[i] * INTEGER(B)[j]` etc.
- Data possibly never actually used

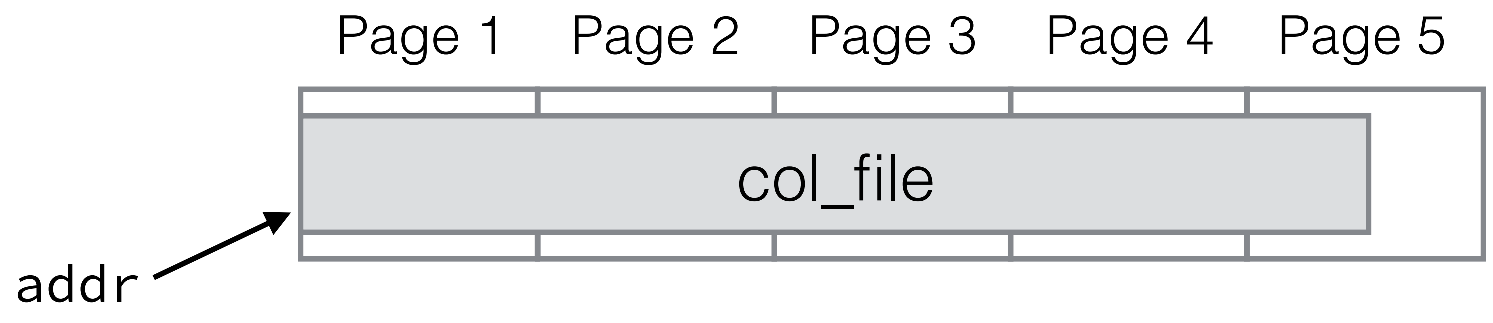
Sometimes lucky

- Perfectly compatible bits:
    - `int my_int_arr[100];`
    - `double my_dbl_arr[100];`
- Doctor `SEXP` header in front of data and good to go
- Implemented in MonetDBLite with custom allocator
    - Next version on CRAN will have this

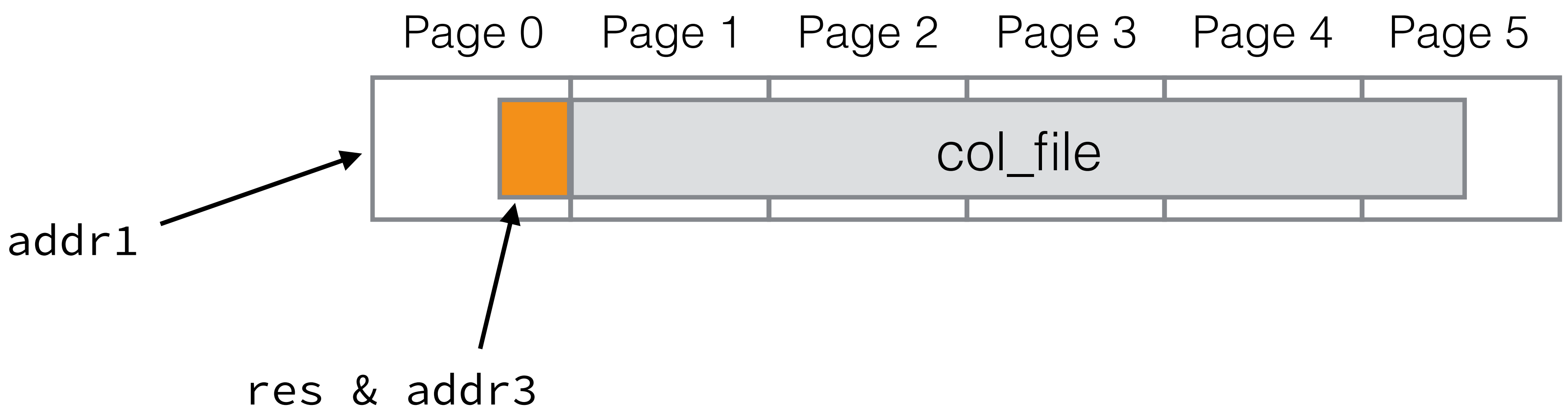
https://github.com/hannesmuehleisen/MonetDBLite

Zero-Copy in MonetDBLite

```
addr = mmap(col_file, len, NULL)
```



```
addr1 = mmap(NULL, len + PAGE_SIZE, NULL)

addr2 = mmap(col_file, len, addr1 + 4096)

addr3 = addr1 + PAGE_SIZE - sizeof(SEXPREC_ALIGN)

SEXP res =  allocVector3(INTSXP, len/sizeof(int), &allocator);
```



Demo 1
Stock R, MonetDBLite & zero-copy

```
library(“DBI”)
con <- dbConnect(MonetDBLite::MonetDBLite(), "/tmp/dscdemo")

dbGetQuery(con, "SELECT COUNT(*) FROM onebillion”)
# 1 1e+09

system.time(a <- dbGetQuery(con, "SELECT i FROM onebillion”))
#     user  system elapsed
#   0.032   0.000   0.033

.Internal(inspect(a$i))
# @20126efd8 13 INTSXP g0c6 [NAM(2)] (len=1000000000, tl=0)
1,2,3,4,5,...
```

Native R Vector
w. zero-copy!

Not always so lucky

- What if we have to actually convert?
  - Strings, `TIMESTAMP` to `POSIXct` etc.
  - `NULL/NA` mismatches
- More involved data representations
  - compressed, batched, hybrid row/col, …
- Need to convert all data before handing control over to R.
  - Can take forever, takes memory, non-obvious wait time

ALTREP

- Luke Tierney, Gabe Becker & Tomas Kalibera
- Abstract vectors, `ELT()/GET_REGION()` methods
- Lazy conversion!

```
static void monetdb_altrep_init_int(DllInfo *dll) {
   R_altrep_class_t cls = R_make_altinteger_class(/* .. */);
   R_set_altinteger_Elt_method(cls, monetdb_altrep_elt_integer);
   /* .. */
}

static int monetdb_altrep_elt_integer(SEXP x, R_xlen_t i) {
   int raw = ((int*) bataddr(x)->theap.base)[i];
   return raw == int_nil ? NA_INTEGER : raw;
}
```

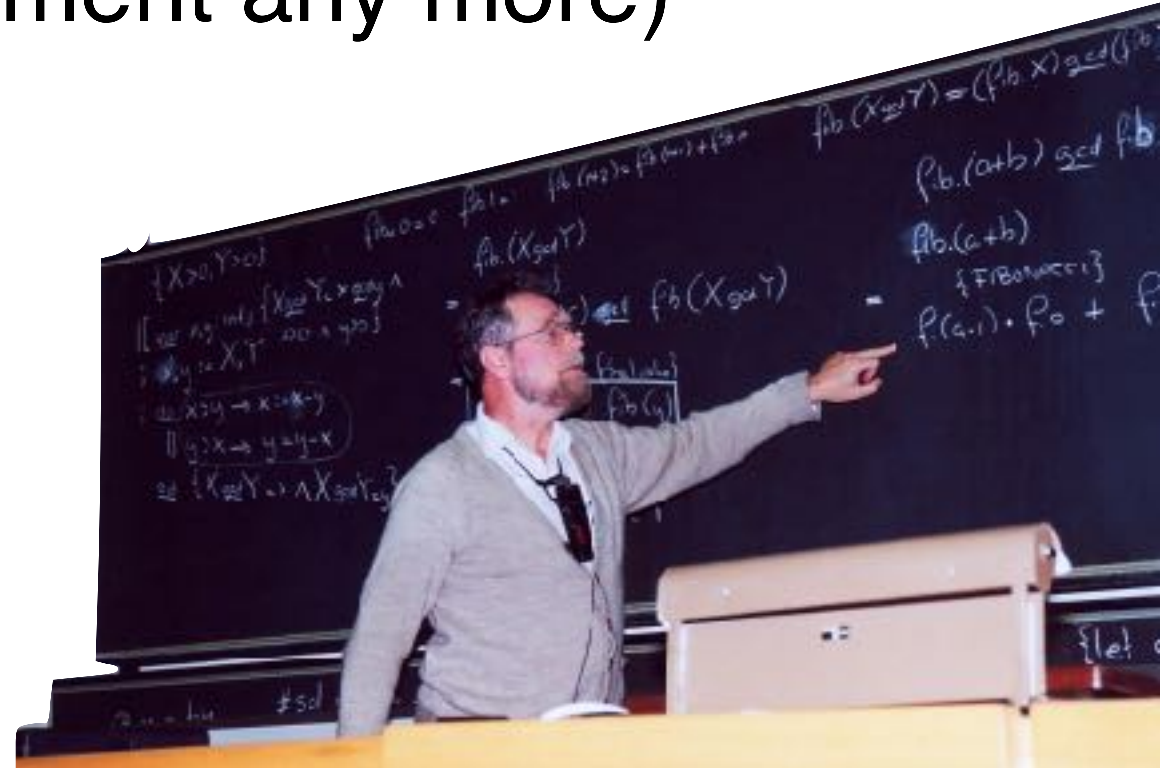
Demo 1
ALTREP, MonetDBLite & zero-copy

```
library(“DBI”)
con <- dbConnect(MonetDBLite::MonetDBLite(), "/tmp/dscdemo")

dbGetQuery(con, "SELECT COUNT(*) FROM onebillion”)
# 1 1e+09

system.time(a <- dbGetQuery(con, "SELECT i FROM onebillion”))
#     user  system elapsed
#   0.001   0.000   0.001

.Internal(inspect(a$i))
# @7fe2e66f5710 13 INTSXP g0c0 [NAM(2)] BAT #1352 int ->
integer
```

ALTREP-wrapped MonetDB Column

`DATAPTR()` considered harmful

- Most base R / some popular packages will be patched for `ALTREP`, but not many (prediction)
- Still get surprising waits / memory overload / … when `DATAPTR()` is called
  - (Just not at the obvious moment any more)



`DATAPTR()` considered harmful

- Example: survey package

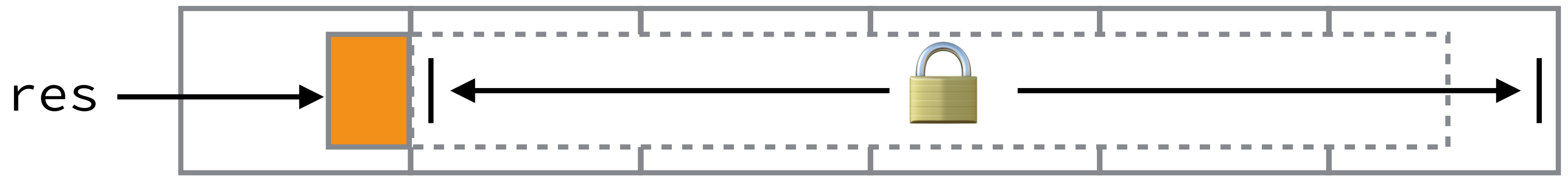
```
svrepdesign.default() →
 drop(as.matrix(na.fail(weights))) →
 complete.cases(object) →
 .External(C_compcases) →
⚡ INTEGER(u)[i]
```

`mprotect()` to the rescue

- MMU can be programmed from user space
- Protects arbitrary memory areas against read/write
- Interrupt/Exception thrown when someone tries access
    - Exception can be caught..
- Can be used for (partial) lazy conversion

mprotect() for Lazy Conversion

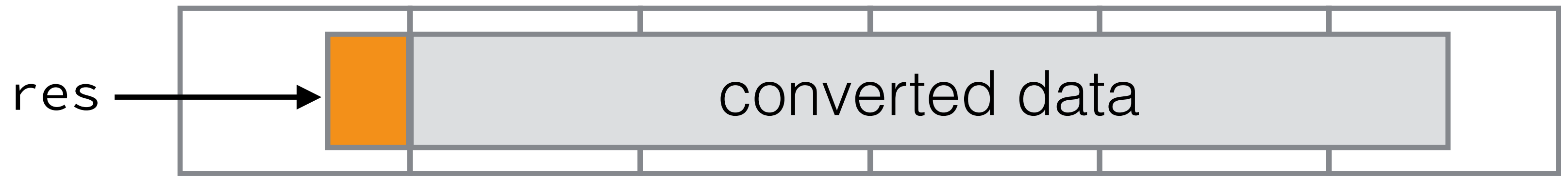
```
addr = mmap(NULL, len + PAGE_SIZE, NULL)
mprotect(addr + PAGE_SIZE, len , PROT_NONE)
SEXP res = allocVector3(…)
sigaction(SIGBUS, &sa, NULL);
```



```
int a = INTEGER(res)[42]
```

Signal handler gets memory address where fault occurred

```
convert(…)
mprotect(addr + PAGE_SIZE, len , PROT_READ)
```



Demo 3
ALTREP & MonetDBLite & Survey

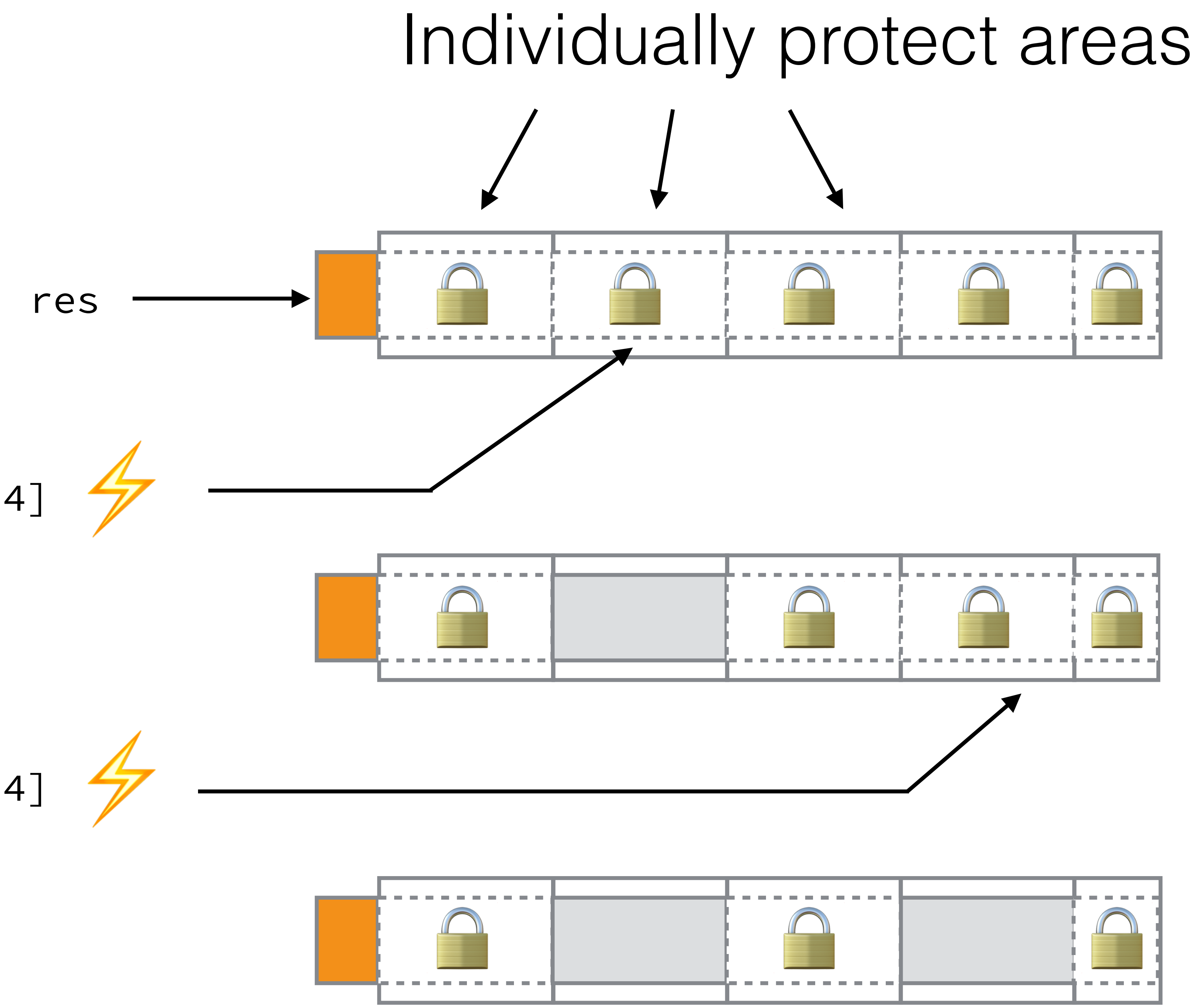
```
con <- dbConnect(MonetDBLite::MonetDBLite(), "/tmp/dscdemo")
s <- "alabama"

svydata <- dbReadTable(con, s)
# free

library(survey)
svydsgn <- svrepdesign(… , data = svydata)
# dataptr(1586)
# Got SIGSEGV at address: 0x110dcc000 for bat 1586
# …
```

`DATAPTR()` called, made protected area, area accessed, converted

Still problematic

- Surprising waits whenever conversion is required
    - User does not expect this
- Still whole vector needs to be pulled into virtual memory
    - Might not be possible, swap space usually quite small

Chunked Conversion



```
int a = INTEGER(res)[1234]
convert(1)

int b = INTEGER(res)[1234]
convert(4)
```

Generic Solution?

- Getting this right is hard, but not implementation-specific
    - No per-class `DATAPTR()`
    - Use `mprotect()`, signal handler & `GET_REGION()`
    - Use temporary mmap-ed file if needed
      (using OS' page cache)
- "`chunkrep`"
    - ALTREP vector wrapping library (PoC)
    - Never calls DATAPTR() on wrapped vector

`https://github.com/hannesmuehleisen/chunkrep`

Demo 4
“chunkrep”

```
a <- 1:10^8
b <- chunkrep::wrap(a)
.Internal(inspect(b))
# @7fae4ea7b640 13 INTSXP g0c0 [NAM(2)] CHUNKREP
#   @7fae4ef6efc8 13 INTSXP g0c0 [MARK,NAM(2)]  1 : 100000000 # (compact)

str(complete.cases(b))
# dataptr(), setting up 5 maps in [0x125671000, 0x13dd10fff]
# Signal for wrapped address: 0x125671000, belongs to chunk 0,
# converting [0:20480000]
# …
# Signal for wrapped address: 0x138ef1000, belongs to chunk 4,
# converting [81920000:100000000]
# logi [1:100000000] TRUE TRUE TRUE TRUE TRUE TRUE ...
```

`DATAPTR()` called,
made protected area,
areas accessed,
converted partially

R Wishlist

- Add non-contiguous `SEXPs` (ALTREP has those)
    - Header / data separation with pointer/callback
- Allow strings to live outside global hash table
- Export `sizeof(SEXPREC_ALIGN)` to C
- Support more than one interpreter per process
    - Perhaps start with outlawing C globals on CRAN

https://github.com/hannesmuehleisen/MonetDBLite

https://github.com/hannesmuehleisen/chunkrep