mmap + indexing

crazy big data, crazy fast

Presented May 27, 2010
Inaugural Chicago RUG

Jeffrey A. Ryan
insight algorithmics, inc

The Problem

Data is unlimited
Memory is finite

... and R uses a memory model

The Options

Get more memory

The Options

Get more memory

expensive and limited

The Options

Get more memory

expensive and limited

Use an external data store

Oracle, MySQL, sqlite, Postgresql, Berkeley DB, Redis, voltDB,Vertica, monetDB, ...

The Options

Get more memory

expensive and limited

Use an external data store

(very) expensive and complex

Oracle, MySQL, sqlite, Postgresql, Berkeley DB, Redis, voltDB, Vertica, monetDB, ...

The Options

Get more memory

expensive and limited

Use an external data store

(very) expensive and complex

Oracle, MySQL, sqlite, Postgresql, Berkeley DB, Redis, voltDB, Vertica, monetDB, ...

What if we changed how R sees your data instead?

mmap

OS system call

very low level API - you see what the C call sees

virtually map files into memory on demand

similar (but different) to the R packages *ff* and *bigmemory*

mmap

mmap	R	C	bytes
raw()	raw	unsigned char	1
char()	raw	char	1
uchar()	raw	unsigned char	1
int8()	integer	signed char	1
uint8()	integer	unsigned char	1
int16()	integer	signed short	2
uint16()	integer	unsigned short	2
int24()	integer	three byte int	3
uint24()	integer	unsigned three byte int	3
int32()	integer	int	4
integer()	integer	int	4
real32()	double	single precision float	4
real64()	double	double precision float	8
double()	double	double precision float	8
cplx()	complex	complex	16
complex()	complex	complex	16
char(n)	character	fixed-width ascii	n+1
character(n)	character	fixed-width ascii	n+1
struct(...)	list	struct of above types	variable

mmap

```
> # 2-byte (int16)
> # 4-byte (int32 or integer)
> # 8-byte float (real64 or double)

> record.type <- struct(short=int16(),int=int32(),double=real64())
> record.type
struct: (short) integer(0)
        (int) integer(0)
        (double) double(0)
> nbytes(record.type) # 14 bytes in total
[1] 14

> m <- mmap(tmp, record.type)
> m[1]
$short
[1] 1

$int
[1] 366214

$double
[1] -1.382365
```

indexing

provide database style indexing and search tools
for R based data objects

column store + binary search + bitmap indexing + mmap

indexing

extend data.frame to use indexes (fast searching)

build in support for disk-based access (unlimited data)

R interface (painfully simple)

indexing

the interface

create_index

load_index

[

vertical partitions

LZO compression

indexing

WAH bitmap compression

binary search

language agnostic storage

the technology

horizontal partitions

bitmap indexing

column store

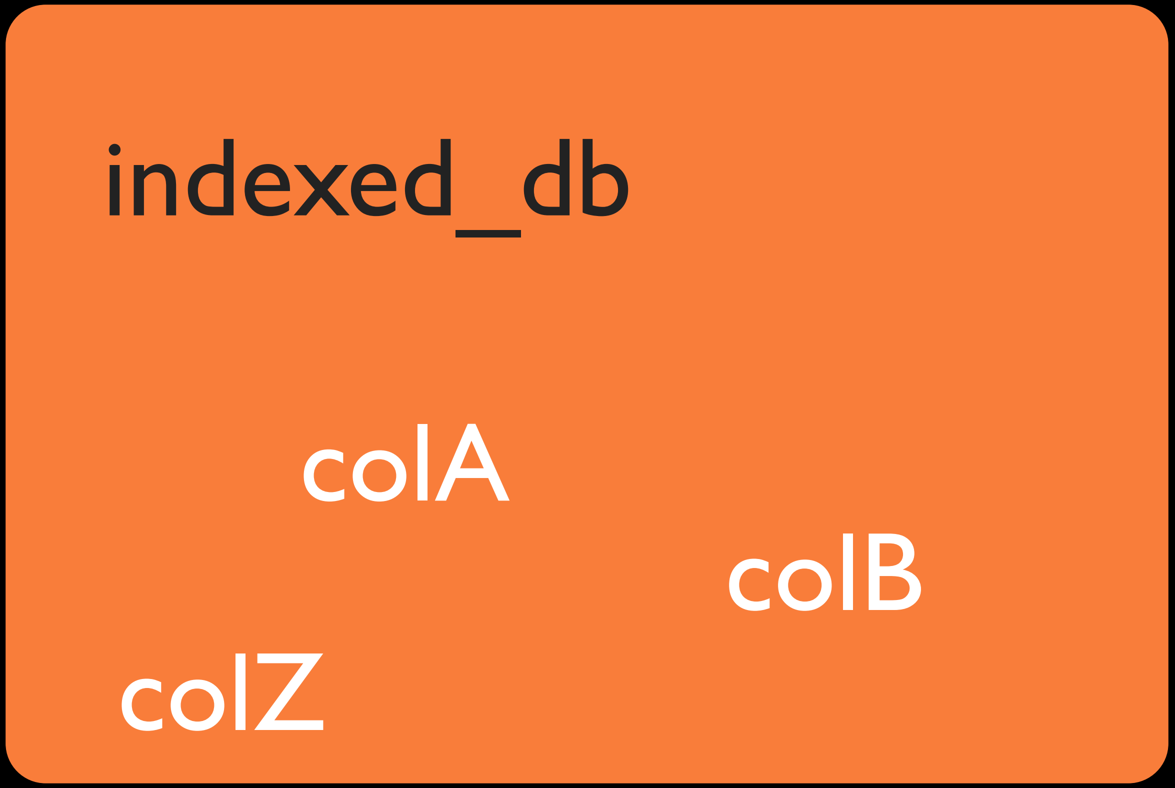
networked

query optimization

caching

mmap + indexing

indexed_db
is an
environment



indexed_db

colA

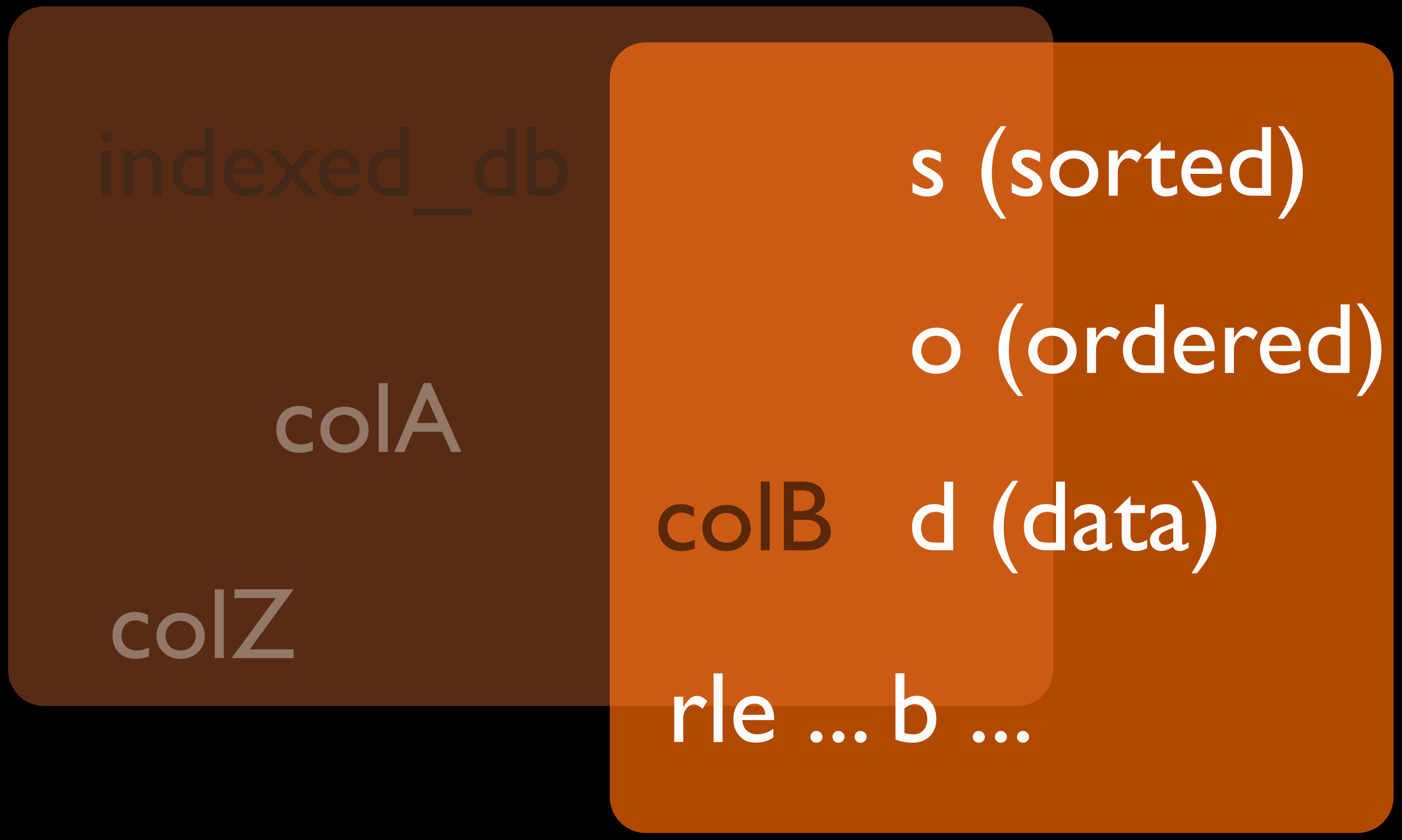
colB

colZ

colA - Z are
“columns”
of your data

“columns” are really objects (lists) in the environment

mmap + indexing



indexed_db

colA

colZ

colB

s (sorted)

o (ordered)

d (data)

rle ... b ...

lists contain the mmap objects to data on disk(s)

mmap + indexing

2 steps

create_index

any column or vector of data

returns the “indexed” environment

e.g.

```
Z <- rnorm(1e6)
db <- create_index(Z)
rm(Z)
```

[

use subsetting to magically extract data from disk using index (fast and friendly)

fancy *j* evaluation included

e.g.

```
db[Z < 0]
db[Z > 1 & Z < -3, Z]
db[Z < -3, mean(Z)]
```

mmap + indexing

Real World Example

67,000,000 equity option contracts
14+ columns

```
> system.time( db[symbols=="AAPL"] )
   user  system elapsed
  0.012   0.000   0.012

> db[symbols=="AAPL"]
91428 hits
```

ia

insight algorithmics inc.

Statistical software, training, and consulting

www.insightalgo.com