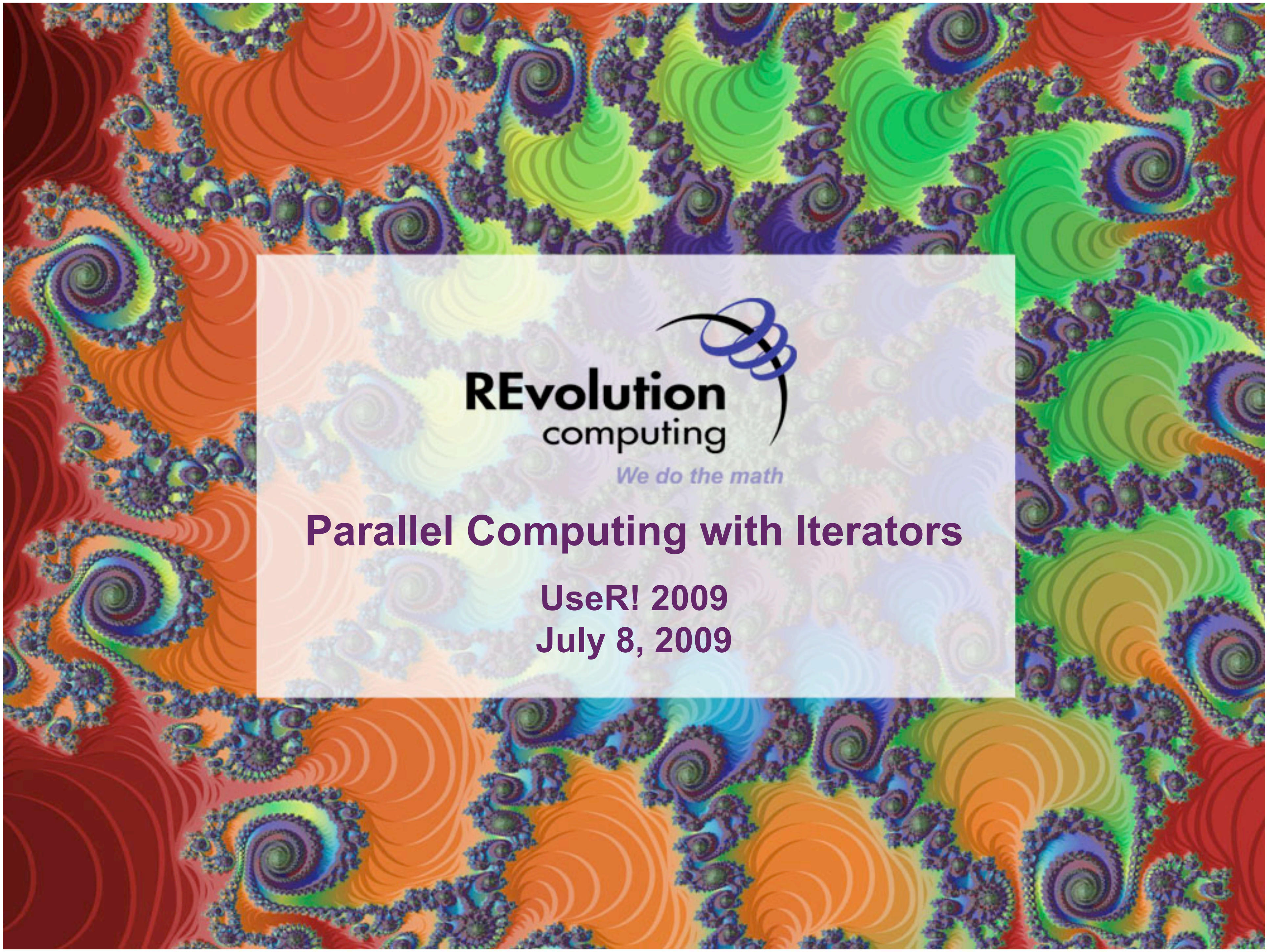


REvolution computing

We do the math

Parallel Computing with Iterators

UseR! 2009

July 8, 2009

Iterators

- `install.packages(“iterators”)`
- Generalized loop variable
- Value need not be atomic
  - Row of a matrix
  - Random data set
  - Chunk of a data file
  - Record from a database
- Create with: `iter`
- Get values with: `nextElem`
- Used as indexing argument with `foreach`

Numeric Iterator

```
> i <- iter(1:3)
> nextElem(i)
[1] 1
> nextElem(i)
[1] 2
> nextElem(i)
[1] 3
> nextElem(i)
Error: StopIteration
```

Long sequences

```
> i <- icount(1e9)
> nextElem(i)
[1] 1
> nextElem(i)
[1] 2
> nextElem(i)
[1] 3
> nextElem(i)
[1] 4
> nextElem(i)
[1] 5
```

Matrix dimensions

```
> M <- matrix(1:25,ncol=5)
> r <- iter(M,by="row")
> nextElem(r)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
> nextElem(r)
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    7   12   17   22
> nextElem(r)
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    8   13   18   23
```

Infinite & Irregular sequences

```
iprime <- function() {
 lastPrime <- 1
 nextEl <- function() {
  lastPrime <<- as.numeric(nextprime(lastPrime))
  lastPrime
 }
 it <- list(nextElem=nextEl)
 class(it) <- c('abstractiter','iter')
it}
```

```
> require(gmp)
> p <- iprime()
> nextElem(p)
[1] 2
> nextElem(p)
[1] 3
```

REvolution computing
We do the math

Looping with `foreach`

```
install.packages("foreach")
```

```
foreach (var=iterator) %dopar% { statements }
```

- Evaluate `statements` until `iterator` terminates
- `statements` will reference variable `var`
- Values of `{ … }` block collected into a list

- Runs sequentially (by default) (or force with `%do%`)

Parallel Computing with Iterators

```
> foreach (j=1:4) %dopar% sqrt (j)

[[1]]
[1] 1

[[2]]
[1] 1.414214

[[3]]
[1] 1.732051

[[4]]
[1] 2
```

REvolution computing
We do the math

Combining Results

```
> foreach(j=1:4, .combine=c) %dopar% sqrt(j)
[1] 1.000000 1.414214 1.732051 2.000000

> foreach(j=1:4, .combine='+') %dopar% sqrt(j)
[1] 6.146264
```

- When order of evaluation is unimportant, use `.inorder=FALSE`

Referencing global variables

```
> z <- 2
> f <- function (x) sqrt (x + z)
> foreach (j=1:4, .combine='+') %dopar% f(j)
[1] 8.417609
```

- foreach automatically inspects code and ensures unbound objects are propagated to the evaluation environment

%dopar%

Modular parallel backends

- registerDoSEQ (default)
- registerDoMC (multicore)
- *registerDoNWS* (NetWorkSpaces)
- registerDoSNOW
- registerDoRMPI

REvolution computing
We do the math

Parallel Computing with Iterators

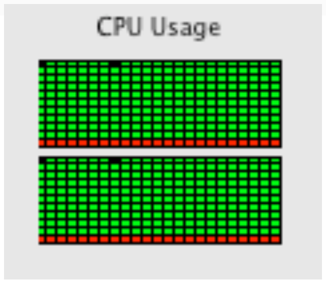
A simple simulation:

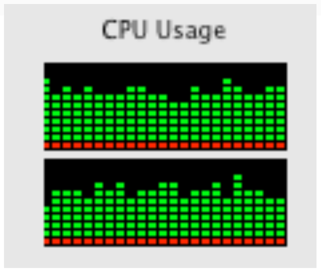
```
birthday <- function(n) {
  ntests <- 1000
  pop <- 1:365
  anydup <- function(i)
      any(duplicated(
         sample(pop, n, replace=TRUE)))
  sum(sapply(seq(ntests), anydup)) / ntests
}

x <- foreach (j=1:100) %dopar% birthday (j)
```

REvolution computing
We do the math

Parallel execution, dual-core MacOS X

```
> install.packages("doMC") # MacOS, Linux
> registerDoMC() # from Terminal, not R.app GUI
> system.time(
+ x <- foreach (j=1:100) %dopar% birthday (j)
+ )
    user  system elapsed
  19.849  19.793  28.886  # cf 40.471 sequential
```





REvolution computing
We do the math

Pitfalls to avoid

- Sequential vs Parallel Programming
- Random Number Generation
  - `library(sprngNWS)`
  - `sleigh(workerCount=8, rngType='sprngLFG')`
- Node failure
- Cosmic Rays (http://tinyurl.com/R-cosmic-rays)

Conclusions

- Iterators a useful programming construct generally
- Parallel computing is easy!
- Write loops with foreach / %dopar%
  - Works fine in a single-processor environment
  - doMC for multiprocessor MacOS & Linux systems
  - REvolution R and NetworkSpaces for Windows, clusters
  - Speed benefits without modifying code
- Easy performance gains on modern laptops / desktops
- Expand to clusters for meaty jobs
  - Appropriate unused PCs overnight!

Thank You!

- David Smith, Director of Community, REvolution Computing
- david@revolution-computing.com
- http://blog.revolution-computing.com

- Longer foreach example: **http://tinyurl.com/R-backtest**

REvolution computing
We do the math

Supporting the R Community

We are an open source company supporting the R community:

- **Benefactor of R Foundation**
- **Financial supporter of R conferences and user groups**
- **Zero-cost "REvolution R" available to everyone**
- **R Community website:** revolution-computing.com/community
  - **"Revolutions" Blog:** blog.revolution-computing.com
  - **Forum:** revolution-computing.com/forum
- **New functionality developed in core R to contributed under GPL**
  - **64-bit Windows support**
  - **Step-debugging support**
- **Promoting R use in the commercial world**

REvolution computing
We do the math

A Taxonomy of Parallel Processing

- Multi-threaded processing (lightweight processes)
  - OpenMP / POSIX threads
  - Multiprocessor / Multicore
  - GPU processors (CUDA/NVIDIA ; ct/INTEL)
  - *Usually* shared memory
  - Harder to scale out across networks
  - Examples: multicore (Unix), threaded linear-algebra libraries for R (ATLAS, MKL)
- Multi-process processing (heavyweight processes)
  - *Usually* distributed memory
  - Easier to scale out across networks
  - Examples: SNOW, ParallelR, Rmpi, batch processing

Data File

```
> rec <- iread.table("MSFT.csv",sep=",", header=T, row.names=NULL)
> nextElem(rec)
  MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
1     29.91     30.25     29.4      29.86    76935100         28.73
> nextElem(rec)
  MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
1      29.7     29.97    29.44      29.81    45774500         28.68
> nextElem(rec)
  MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
1     29.63     29.75    29.45      29.64    44607200         28.52
> nextElem(rec)
  MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
1     29.65      30.1    29.53      29.93    50220200          28.8
```

REvolution computing
We do the math

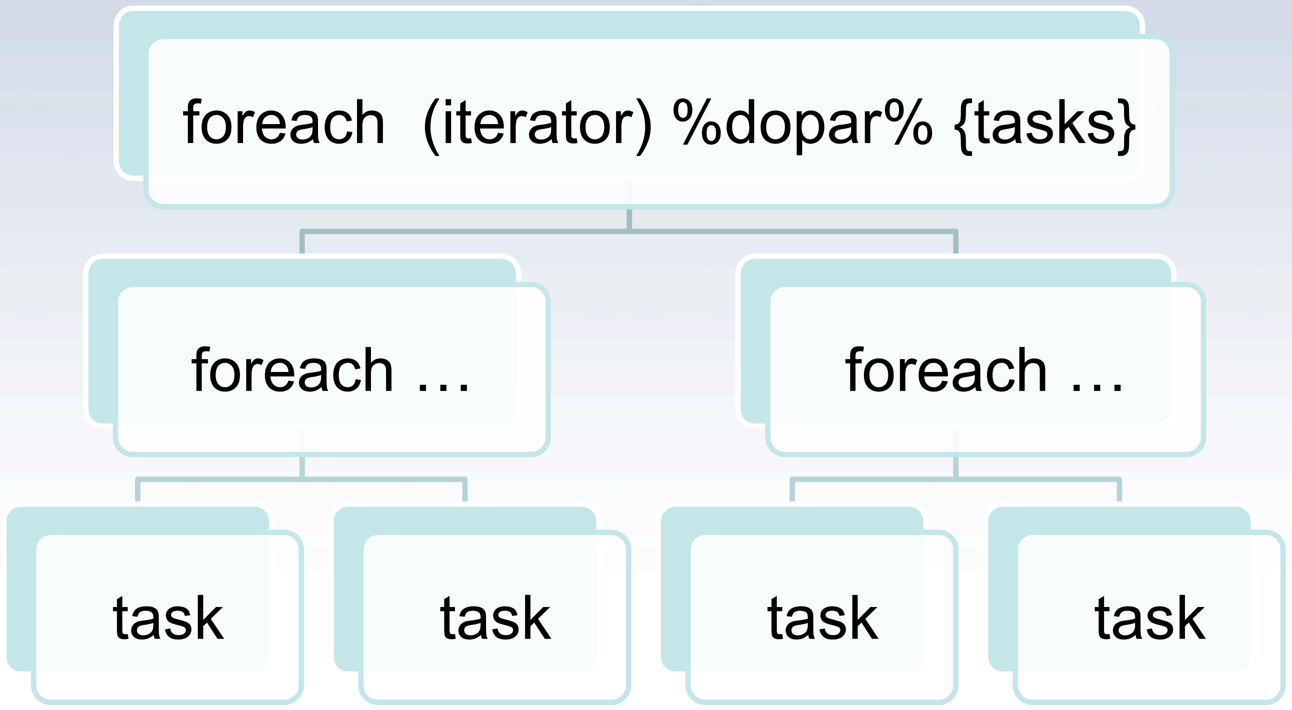
Database

```
> library(RSQLite)
> m <- dbDriver('SQLite')
> con <- dbConnect(m, dbname="arrests")
> it <- iquery(con, 'select * from USArrests', n=10)
> nextElem(it)
            Murder Assault UrbanPop Rape
Alabama       13.2     236       58 21.2
Alaska        10.0     263       48 44.5
Arizona        8.1     294       80 31.0
Arkansas       8.8     190       50 19.5
California     9.0     276       91 40.6
Colorado       7.9     204       78 38.7
Connecticut    3.3     110       77 11.1
Delaware       5.9     238       72 15.8
Florida       15.4     335       80 31.9
Georgia       17.4     211       60 25.8
```

Distributed Computing

CLUSTER (NWS)

SMP (MC)


foreach (iterator) %dopar% {tasks}
foreach …
foreach …
task
task
task
task