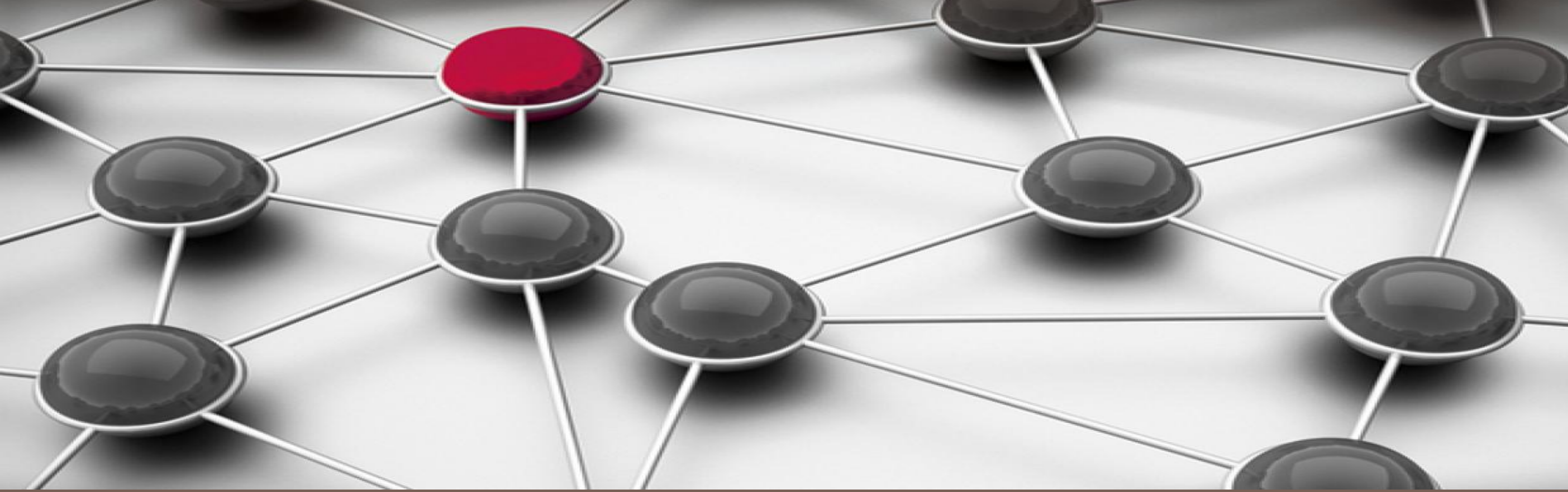


PARALLELIZING R FUNCTIONS

LIFEWATCH GREECE PROJECT

LifeWatch
E-Science European Infrastructure for Biodiversity and Ecosystem Research

HCMR, Crete – June 2014

LifeWatch Greece

The Team

- Anastasis Oulas
  PhD – Molecular Biology and Biomedicine
- Nikolas Pattakos
  MSc – High Performance Computing
- Theodore Patkos
  PhD – Computer Science

- The Problem

The Problem

- The R language is
  - single-threaded
  - memory bound
- Until recently the largest square matrix was ≈ $46*10^3 \times 46*10^3$
- The objective is to take advantage of *parallel computing* solutions to
  - overcome memory barriers (big data segmentation)
  - perform task segmentation (multi-cores, cluster computing)

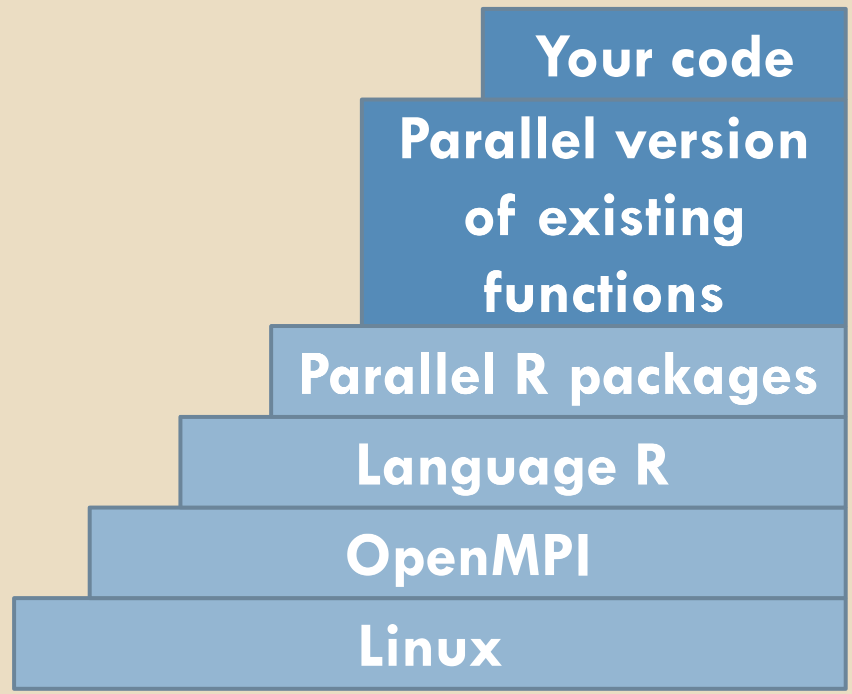
The HCMR Cluster

- The HCMR Biocluster comprises two queues
  - BigMem: Intel E5-2667 12 cores@2.9 with 385GB RAM
  - Batch: Intel X5680 96 cores@3.33 with 48GB ram

- The Approach

The Approach

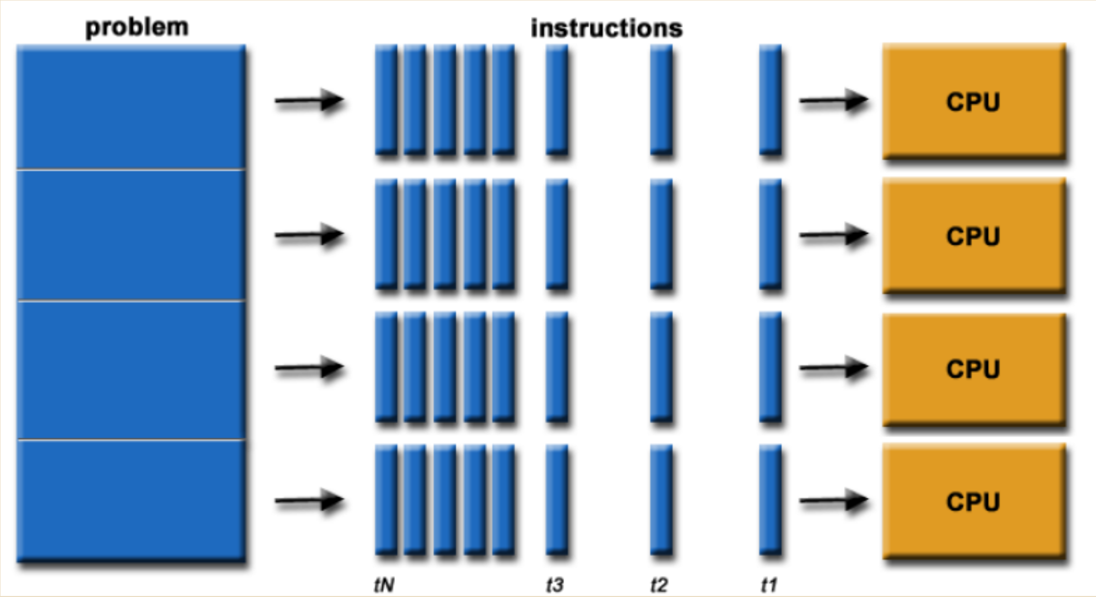
- We use Rstudio and experiment with several R packages, such as
  - pbdR, bigmemory
  - RMPI, pbdMPI
  - Parallel, multicore, snow
- over Linux and OpenMPI implementations.



LifeWatch Greece

The Approach

- We focus on the parallelization of functions at two levels of abstraction
  - Primitive functions (outer product, matrix multiplication etc)
  - General functions (taxa2dist, taxondive, simper, pca etc.)



Example 1: Outer product

- The outer product of vectors is used by many functions

$$[a_1, a_2, \ldots, a_N] \otimes [b_1, b_2, \ldots, b_N] = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \ldots & a_1 b_N \\ a_2 b_1 & a_2 b_2 & \ldots & a_2 b_N \\ \vdots & & \ddots & \vdots \\ a_N b_1 & a_N b_2 & \ldots & a_N b_N \end{bmatrix}$$

- If the vectors are big, the matrix does not fit in memory.

Example 1: Outer product

- We segment the first vector p parts and each core will calculate $m=\lceil N/p \rceil$ rows

$$[a_1, a_2, |...|, a_N] \otimes [b_1, b_2, ..., b_N] = \left[\begin{array}{cccc} a_1 b_1 & a_1 b_2 & \dots & a_1 b_N \\ a_2 b_1 & a_2 b_2 & \dots & a_2 b_N \\ \hline \vdots & & \ddots & \vdots \\ \hline a_N b_1 & a_N b_2 & \dots & a_N b_N \end{array}\right]$$

Example 1: Outer product

- In special cases, we perform optimized allocation reducing the number of computations.

$$
[a_1, a_2, \ldots, a_N] \otimes [a_1, a_2, \ldots, a_N] = \left[\begin{array}{cccc}
a_1 a_1 & a_1 a_2 & \ldots & a_1 a_N \\
a_2 a_1 & a_2 a_2 & \ldots & a_2 a_N \\
\hline
\vdots & & \ddots & \vdots \\
\hline
a_N a_1 & a_N a_2 & \ldots & a_N a_N
\end{array}\right]
$$

$$
[a_1, a_2, a_3, |a_4, \ldots|, a_N] \otimes [a_1, a_2, a_3, a_4, \ldots, a_N] = \left[\begin{array}{cccccc}
a_1 a_1 & & & & & \\
a_2 a_1 & a_2 a_2 & & & & \\
a_3 a_1 & a_3 a_2 & a_3 a_3 & & & \\
\hline
a_4 a_1 & a_4 a_2 & a_4 a_3 & a_4 a_4 & & \\
\ldots & & & & & \\
\hline
a_N a_1 & a_N a_2 & a_N a_3 & a_N a_4 & \ldots & a_N a_N
\end{array}\right]
$$

Low- vs High-level Parallelization

- It can be much more efficient to focus on high-level functions, rather than low-level operations.

- For instance, we take advantage of the fact that data are already distributed and may need to be reused.

Example 2: Taxondive (vegan)

- The Taxondive fuction
  - ```
    del <-
      apply(comm,1, function(x)
                  sum(as.dist(outer(x,x))*dis))
    ```
    ```

Example 2: Taxondive (vegan)

- The Taxondive fuction
  - ```
    del <-
       apply(comm,1, function(x)
                 sum(as.dist(outer(x,x))*dis))
    ```

- `comm` is a MxN matrix, where typically M<<N

- As such, `outer(x,x)` is a NxN matrix, usually huge.

LifeWatch Greece
    ```

Example 2: Taxondive (vegan)

- The Taxondive fuction
  - ```
    del <-
       apply(comm,1, function(x)
                 sum(as.dist(outer(x,x))*dis))
    ```

- `del`, on the other hand, is a vector of length M, usually small.
    ```

Example 2: Taxondive (vegan)

- The Taxondive fuction
  - ```
    del <-
      apply(comm,1, function(x)
                sum(as.dist(outer(x,x))*dis))
    ```
  - ```
    dstar <-
      apply(comm,1, function(x)
           sum(dis*(xx
            <- as.dist(outer(x, x))))/sum(xx))
    ```
- We need to multiply both from the left and from the right.
- Outer is already distributed

LifeWatch Greece

- Experimental valuation

Experimental Evaluation

- Two different implementations of Taxa2Dist

Taxa2DistMPI

	1%			10%			50%		
(cores)	(sec)	(Mb)	(dimensions)	(sec)	(Mb)	(dimensions)	(sec)	(Mb)	(dimensions)
1	4.861	21.7127	1687 1687	203.793	2177.1769	16893 16893			
2	2.089	10.8628	844 1687	152.567	1088.653	8447 16893			
6	1.63	3.6296	282 1687	72.317	362.9274	2816 16893			
12	1.471	1.8149	141 1687	21.336	181.4638	1409 16893			
24	1.18	0.9139	71 1687	11.721	90.732	704 16893			

Taxa2Dist Ddmatrix

	1%			10%			50%		
1	4.424	21.7139	-	1218.76	2177.178	-			
2	8.795	10.864	-	1047.17	1088.783	-			
6	11.182	3.633	-	906.015	362.993	-			
12	4.926	1.8214	-						
24	4.834	0.9071	-						

LifeWatch Greece

Experimental Evaluation

- Scalability and performance

Taxa2DistMPI

	1%			10%			50%		
(cores)	(sec)	(Mb)	(dimensions)	(sec)	(Mb)	(dimensions)	(sec)	(Mb)	(dimensions)
1	4.861	21.7127	1687 1687	203.793	2177.1769	16893 16893			
2	2.089	10.8628	844 1687	152.567	1088.653	8447 16893			
6	1.63	3.6296	282 1687	72.317	362.9274	2816 16893			
12	1.471	1.8149	141 1687	21.336	181.4638	1409 16893			
24	1.18	0.9139	71 1687	11.721	90.732	704 16893			

Taxa2Dist Ddmatrix

	1%			10%			50%		
1	4.424	21.7139	-	1218.76	2177.178	-			
2	8.795	10.864	-	1047.17	1088.783	-			
6	11.182	3.633	-	906.015	362.993	-			
12	4.926	1.8214	-						
24	4.834	0.9071	-						

Experimental Evaluation

- The problem here is not time, but data size

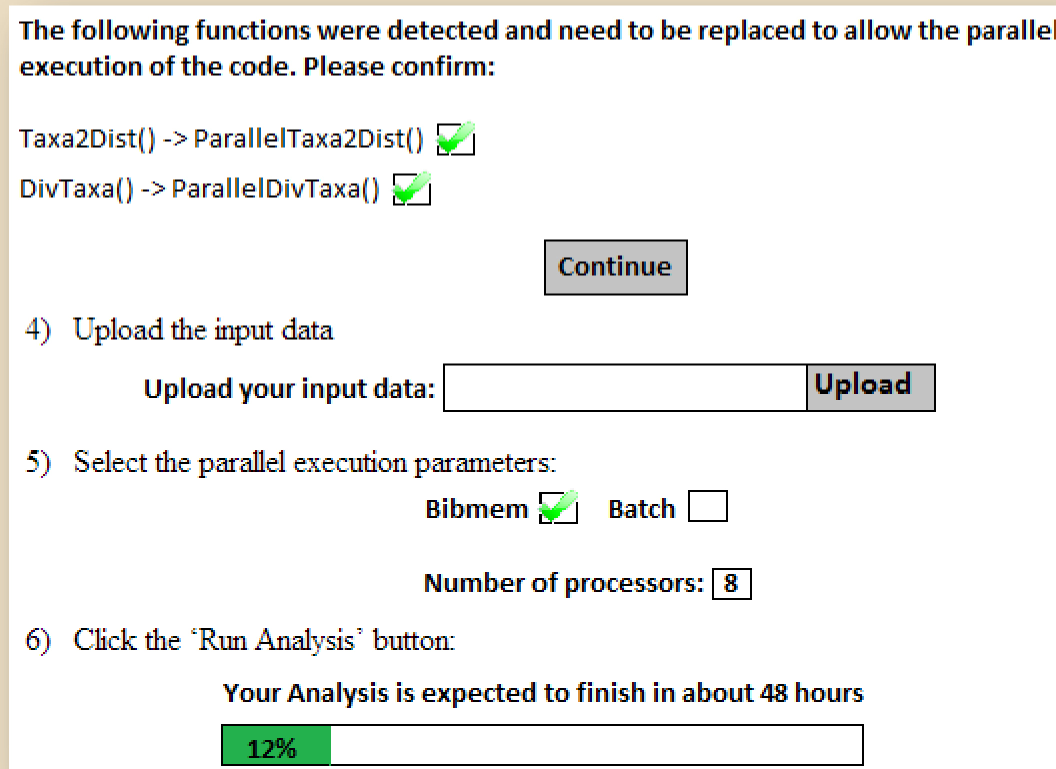
TaxonDive MPI

		1%			10%		50%
1	11.803	22.0245	1687 1687	978.5	2177.435	16893 16893	
2	5.906	11.0965	844 1687	882.771	1088.846	8447 16893	
6	3.064	3.6296	282 1687	274.484	363.078	2816 16893	
12	2.085	1.9833	141 1687				
24	1.665	0.9139	71 1687				

- Goals and Open Problems

The R Statistical Processing vLab

- From the UI
  - The user will upload R scripts
  - The system will recognize functions that can be parallelized
  - Information regarding expected time based on input data will also be displayed



Open problems

- It is not certain that the parallel solution will be more profitable in all cases
  - Our evaluation will offer evidence for better resource management

- We need to decide which functions to parallelize (the code for many functions is not always available to us)

- Thank you!