

Developing web-based and parallelized biostatistics/bioinformatics applications: ADaCGH as a case example

Ramón Díaz-Uriarte

Statistical Computing Team
Structural and Computational Biology Programme
Spanish National Cancer Centre (CNIO)

`rdiaz02@gmail.com`

`http://ligarto.org/rdiaz`

Statistical Computing 2007, Schloss Reichartshausen

Ultimate goal

Develop a framework/set of examples that will allow to quickly turn methodological developments into parallelized web-based applications.

Ultimate goal

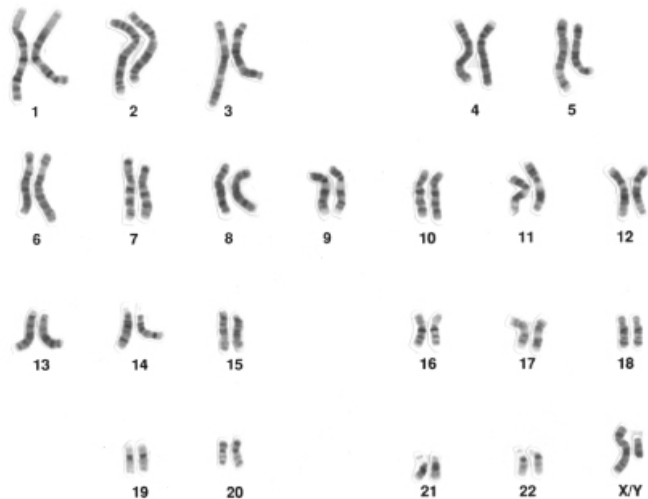
Develop a framework/set of examples that will allow to quickly turn methodological developments into parallelized web-based applications.

(Ultimate goal of the talk: walk through one particular instance)

Outline

- 1 aCGH analysis tools: end user's needs
- 2 Implementation
 - Parallelizing R code
 - Web-based application
- 3 Issues
- 4 Work in progress

Chromosomes



From Wikipedia; original source <http://www.genome.gov/Pages/Hyperion/>

Bioinformatics/biostatistics needs

Accessible, user-friendly, applications for biomedical researchers.

- Statistical rigor and currently accepted and state-of-the-art methods
- Short user wall time: use (hardware/software) resources rarely available to individual biomedical researchers

Bioinformatics/biostatistics needs

Accessible, user-friendly, applications for biomedical researchers.

- Statistical rigor and currently accepted and state-of-the-art methods
- Short user wall time: use (hardware/software) resources rarely available to individual biomedical researchers

Relevant also for statisticians

- Decrease in user wall time: simulations and method comparisons

Requirements for user-oriented aCGH analysis applications

Statistical rigor and currently accepted and state-of-the-art methods

Reviews by Willenbrock and Fridlyand (2005), Lai et al. (2005). R/BioC packages for CBS, HMM, GLAD, CGHseg(*); BioHMM, PSW. R-code for wavelet-based. Java for ACE. Use R

Decreased user wall time Parallelization

User friendliness Web-based interface

Decreased user wall time Web-based interface

1 aCGH analysis tools: end user's needs

2 **Implementation**

- Parallelizing R code
- Web-based application

3 Issues

4 Work in progress

R code

- Code available for most procedures (but none parallelized)
- Many computations are embarrassingly parallelizable:
 - ▶ Segmentation: for most methods, independently for each array*chromosome unit. Can be done concurrently over all array*chromosomes.
 - ▶ Some steps (e.g. post segmentation merging): at the array level
- Figures (with annotations): can be parallelized.

Parallelizing R code

- (Implement missing functionality/methods in R/C)
- Using MPI via the R packages Rmpi and papply
- Simple mechanism that uses send, receive, and broadcast
- Load balanced
- Use wrappers over “mid level” functions in corresponding package: ease updating (papply: easy debugging)
- Parallelize:
 - ▶ arrays
 - ▶ arrays by chromosomes
 - ▶ (or a combination of both)

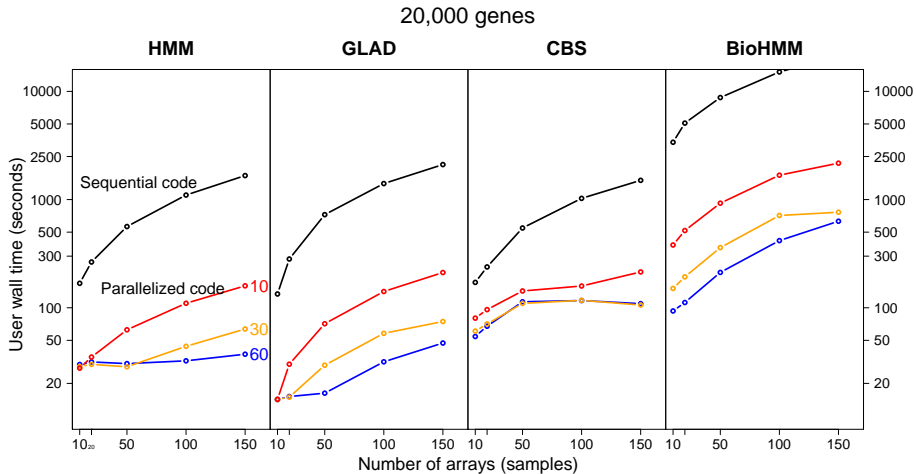
What do we gain?

- Are speed improvements really worth the effort?
- Over what range of problems do we see improvements?
- With what hardware can we see improvements?

Scenario for benchmarks

- Cluster: 2 master nodes, 30 computing nodes
- Computing node: 2 dual-core AMD Opteron (2.2 GHz) CPUs, 6 GB RAM
- Debian GNU/Linux, stock kernel (2.6.8), R (2.4.1), LAM/MPI (7.1.2).
- Ethernet
- Shared storage: NFS. Using same ethernet switch and network cards as MPI

What do we gain?



It works ...

- Speed-ups by factors of 15x (CBS), 30x (BioHMM), 45x (GLAD, HMM)
- Some are disappointing ($\ll 60$)
- R package ADaCGH available from CRAN.

... two details ...

- How many Rslaves per node (our case: 120 vs. 60 slaves)
- Parallelization: over arrays or arrays*chromosome or a combination?
 - ▶ Most cases: at least final merging step cannot be at the array*chromosome level
 - ▶ array*chromosome: much more communication (synchronization and sending initial jobs)

 - ▶ dual cores
 - ▶ cache
 - ▶ communication overhead: Ethernet

... two details ...

- How many Rslaves per node (our case: 120 vs. 60 slaves)
- Parallelization: over arrays or arrays*chromosome or a combination?
 - ▶ Most cases: at least final merging step cannot be at the array*chromosome level
 - ▶ array*chromosome: much more communication (synchronization and sending initial jobs)

 - ▶ dual cores
 - ▶ cache
 - ▶ communication overhead: Ethernet

Is it really worth it to spend a lot of time with these?

... two details ...

- How many Rslaves per node (our case: 120 vs. 60 slaves)
 - Parallelization: over arrays or arrays*chromosome or a combination?
 - ▶ Most cases: at least final merging step cannot be at the array*chromosome level
 - ▶ array*chromosome: much more communication (synchronization and sending initial jobs)

 - ▶ dual cores
 - ▶ cache
 - ▶ communication overhead: Ethernet
- Is it really worth it to spend a lot of time with these?
- ▶ hardware changes
 - ▶ method improvements

What do we gain?

Are speed improvements really worth the effort? Yes (your effort: typing “R CMD INSTALL ADaCGH”).

Over what range of problems do we see improvements? At least from 10 to 150 arrays and 10,000 to 40,000 spots/genes.

With what hardware can we see improvements? At least with clusters from small (5 two-CPU nodes) to medium (30 nodes).

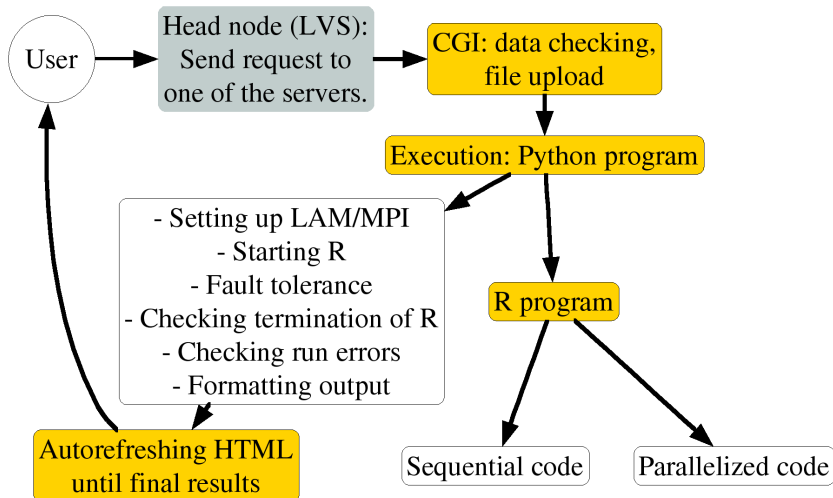
- Smaller clusters: more cost effective (10 Rslaves lead to almost 10x speed increase).
- “Single node clusters” less communication overhead. E.g.: workstations with two dual-core CPUs.

Requirements for user-oriented aCGH analysis applications

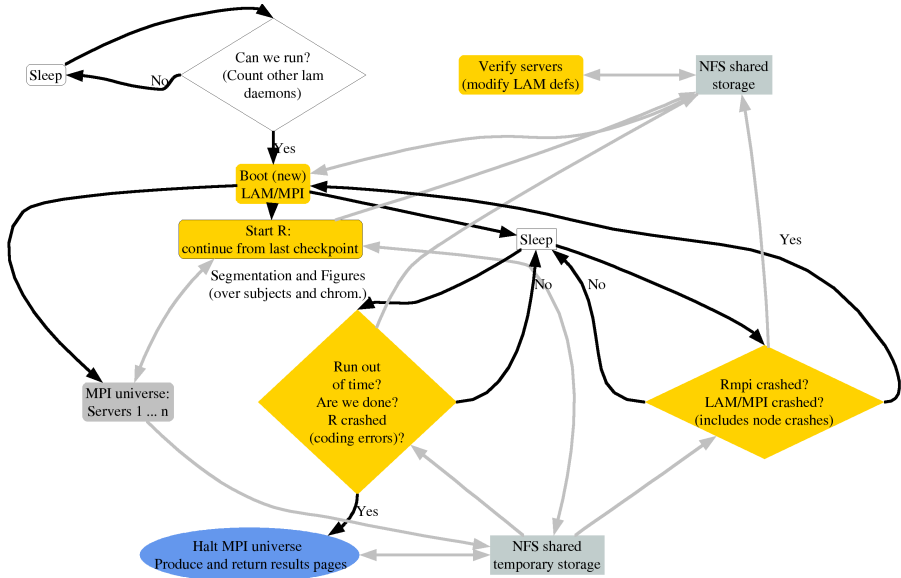
- Parallelization of state-of-the art, validated methods
- Web-based interface for user-friendly access and transparent parallelization

Web-based application (I)

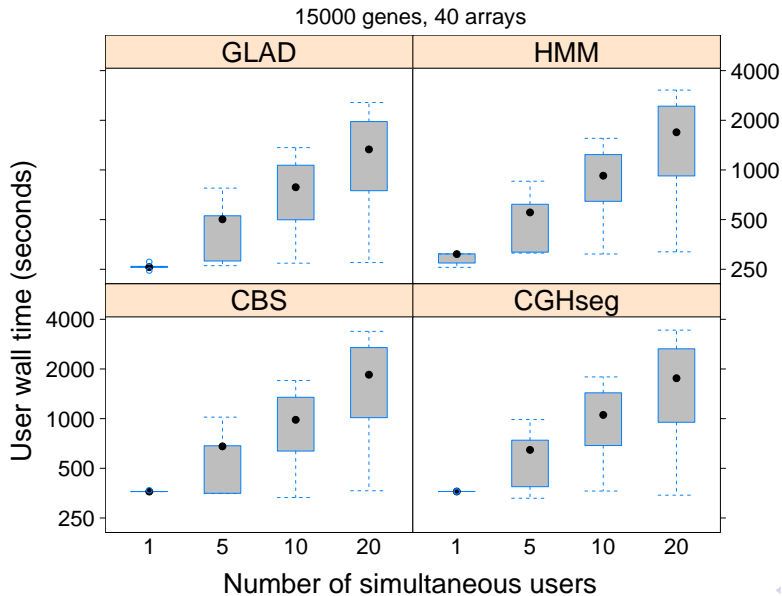
<http://adacgh.bioinfo.cnio.es>



Web-based application (II)



Web-based: timings



Too many languages

Impedance mismatch problem:

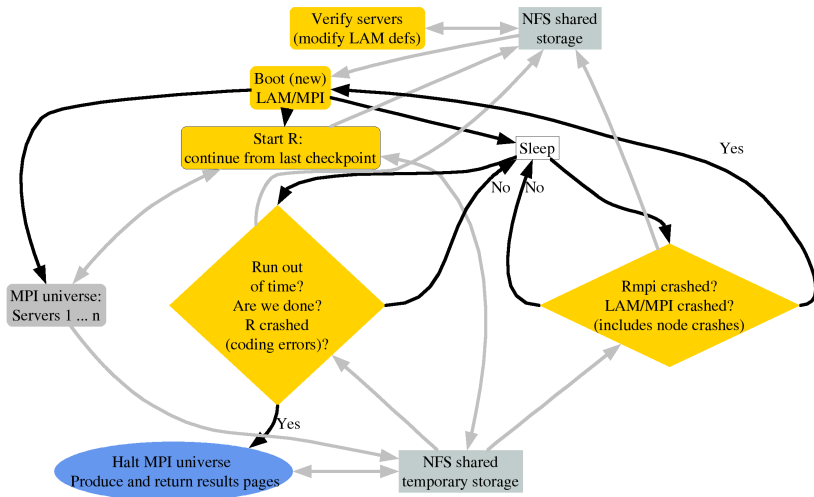
“Building Web-based applications requires the mastering of a number of languages/technologies (e.g. HTML, CSS, CGI, ASP, PHP, XML, etc..). Such languages and technologies were created to address different aspects on a by-need evolutionary manner. The result is a plethora of tools that are fitted together in an ad hoc fashion.” El-Ansary, Grolaux, Van Roy, Rafea (2005)

“Overcoming the Multiplicity of Languages and Technologies for Web-Based Development Using a Multi-paradigm Approach”.

- R and C
- HTML and Python: CGI, data entry, display
- Python (and others): control and monitor MPI
- Javascript: AJAX and figures

Fault tolerance and communication

- MPI: little fault tolerance
- Too much network traffic



Work in progress

Too many languages Use languages designed to overcome this problem: Hop, Links, QHTML.

Work in progress

Too many languages Use languages designed to overcome this problem: Hop, Links, QHTML.

Fault tolerance and too much traffic Alternatives to MPI?

- Linda and tuple spaces (also between-language funct.)
- PVM
- Roll-our-own based on Rserve
- Have Erlang control R processes?

Future work

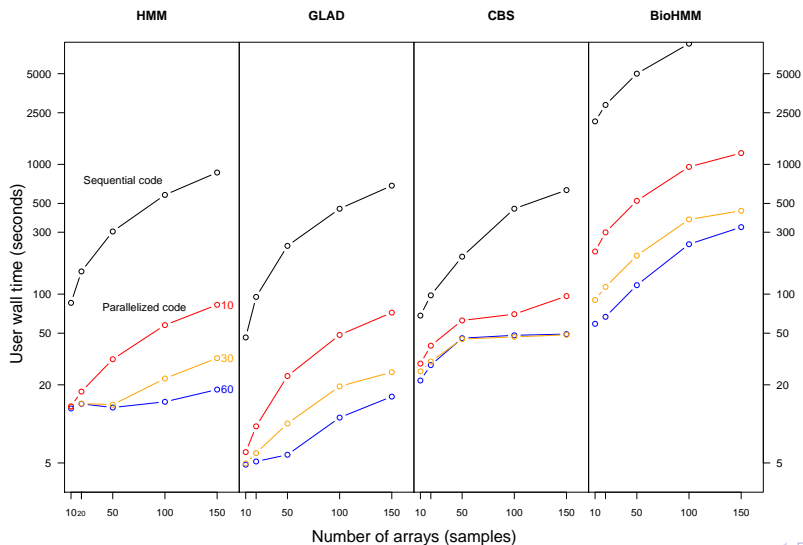
- Framework to compare parallelization alternatives (MPI, PVM, Linda, Rserve)
- Grain (e.g., array vs. array*chromosome)
- Diagnostics on bottlenecks

Acknowledgements

- O. M. Rueda, A. Alibés, A. Cañada, E. R. Morrissey, M. .L. Neves.
- Funding from Fundación de Investigación Médica Mutua Madrileña and Project TIC2003-09331-C02-02 of the Spanish Ministry of Education and Science
- Ramón y Cajal Programme of the Spanish Ministry of Education and Science
- L. Hsu, D. Grove, T. Price, O. Lingjaerde for code and discussion, S. Weston for answers about Linda, and LAM/MPI developers for help with MPI.
- The R users and developers for a vibrant statistical computing community and amazing platform

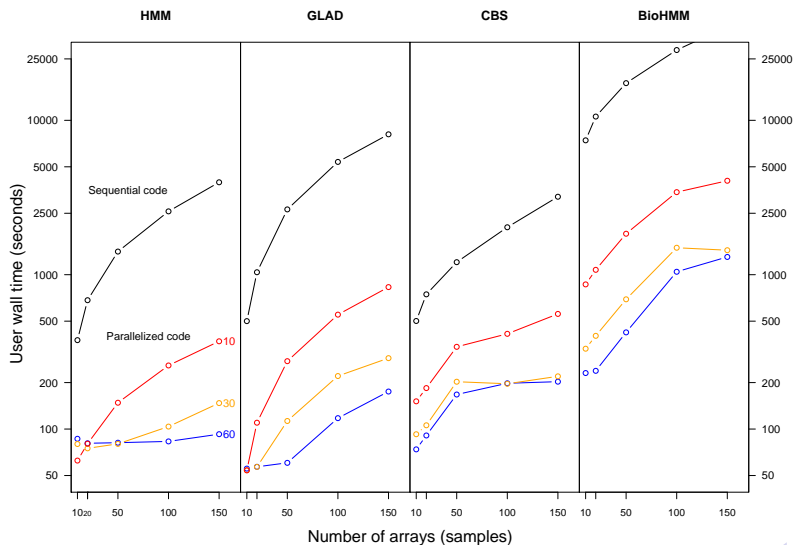
Does it work? (II)

10,000 genes

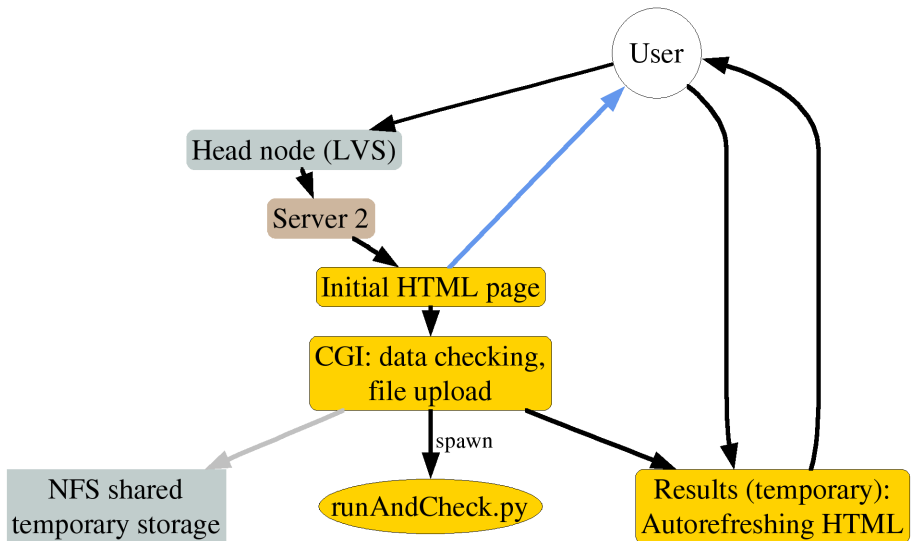


Does it work? (III)

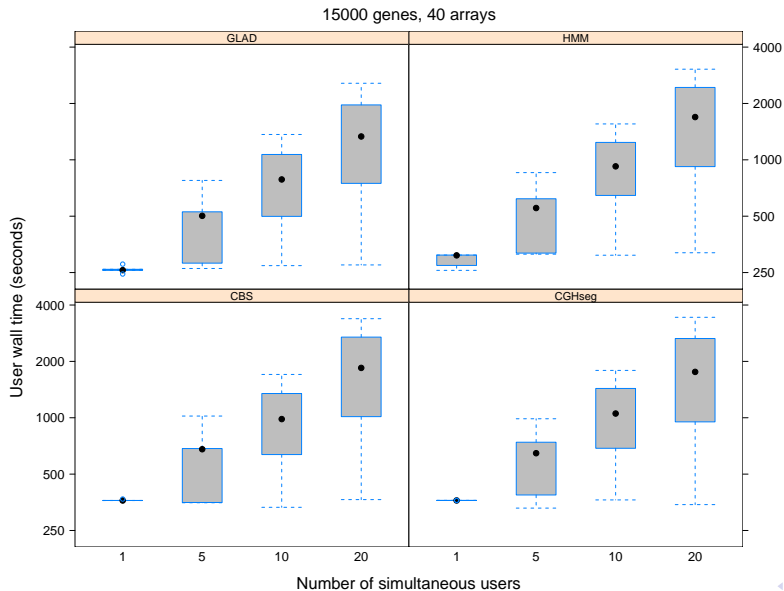
42,325 genes



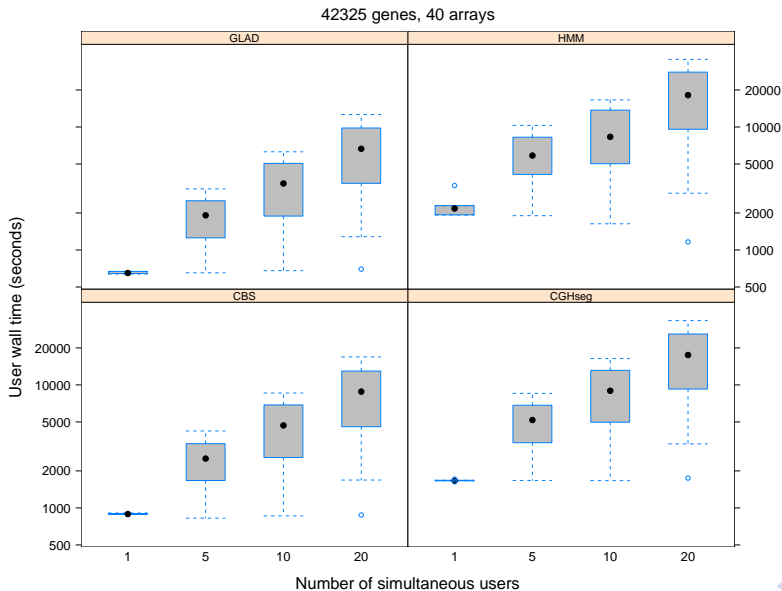
Web-based application (Appendix)



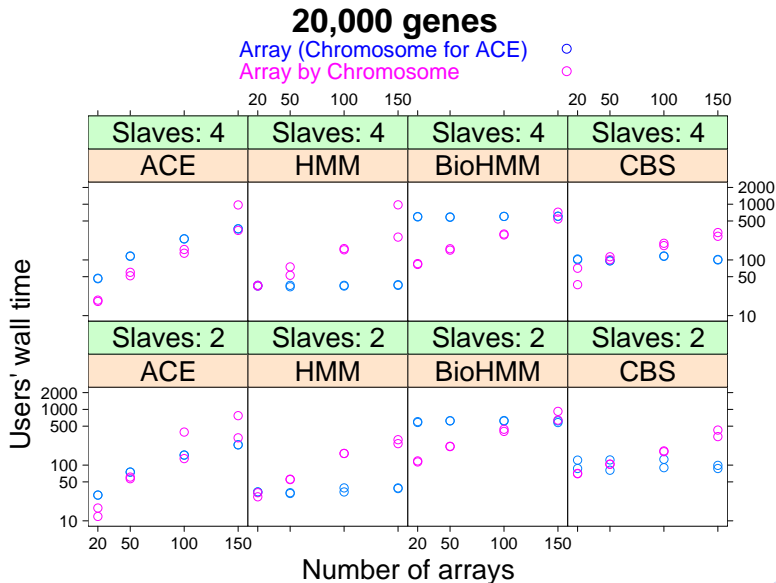
Web-based: timings (II)



Web-based: timings (III)



Number of R slaves



Number of R slaves

42,325 genes

