Parallel Pre-processing of Affymetrix Microarray Data

Pietro Hiram Guzzi and Mario Cannataro

Department of Experimental Medicine and Clinic,
University Magna Græcia, 88100 Catanzaro, Italy
{hguzzi,cannataro}@unicz.it

Abstract. The study of genes is currently carried out by systematic analysis that relies on data produced by the microarray technology. The recent development of such technology and the increasing number of analysed samples result in an increased volume of raw data for each experiment. The time for preprocessing represents an important amount of the analysis, so the need to introduce tools for the efficient preprocessing arises. This paper presents a system able to manage and preprocess microarray data in a parallel way. First experimental results on Affymetrix data showing appreciable improvements in term of execution times are discussed.

1 Introduction

Biological processes within cells are carried out by genes that can be studied using microarray technology. For processing and studying DNA microarrays many different technologies as well as algorithms and tools have been introduced [2,3,7]. A typical workflow for analysing microarray data is structured on four main phases: *(i)* preprocessing, that comprises summarisation and normalization, *(ii)* annotation; *(iii)* statistical and data mining analysis, and *(iv)* biological interpretation.

Raw data generated from microarray platforms, e.g. Affymetrix Cel Files or Illumina Tagged Images, need to be preprocessed. The first step in preprocessing, known as summarisation, aims to recognise the position of different genes in raw images, associating different regions of pixels to the unique gene that generated them. Normalisation aims to correct the variation of gene expression in the same array due to experimental bias. Filtering reduces the number of investigated genes on the basis of biological considerations, e.g. genes of known functions, or considering statistical criteria (e.g. associated p-value). Finally, the annotation process associates each gene to a set of functional information, such as biological processes that are related to gene, and a set of cross reference database identifiers. Statistical and data mining analysis phases aim to identify biological meaningful genes, e.g. by finding differentially expressed genes among two groups of patients on the basis of their expression values.

The typical dimension of microarray datasets is growing for two main reasons: the dimension of files encoding a single chip and the number of the arrays involved in a single experiment are increasing. Let us consider, for instance, two

M.R. Guarracino et al. (Eds.): Euro-Par 2010 Workshops, LNCS 6586, pp. 225–232, 2011.

common Affymetrix microarray files (named CEL files): the older Human 133 Chip CEL file that has a dimension of 5 MB and contains 20000 different genes while the newer Human Gene 1.0 st that has a typical dimension of 10 MB and contains 33000 genes. A single array of the Exon family (e.g. Human Exon or Mouse Exon) can have up to 100 MB of size. Moreover the recent trend in genomics is to perform microarray experiment considering a large number of patients.

From this scenario, the need for the introduction of tools and technologies to process such huge volume of data in an efficient way arises. A possible way to develop the efficient preprocessing of microarray data is represented by the parallelization of existing algorithms on parallel computing, e.g. clusters. In the scenario we envision the whole computation is distributed onto different processors, that perform computations on smaller sets of data and results are finally integrated. Such scenario requires the design of new algorithms for summarisation and normalisation that take advantage of the underlying parallel architectures. Nevertheless, a first step in this direction can be represented on the replication on different nodes of existing preprocessing software that runs on smaller datasets.

This paper presents a software system based on a master-worker architecture for the parallel preprocessing of Affymetrix data. The core of the system is in fact based on a master node that distributes data on different nodes. Each node performs preprocessing on a subset of the dataset employing the Affymetrix Power Tools (APT)[1]. Finally, results are moved to the master node and are integrated.

Despite its relevance, the parallel processing of microarray data is a relatively new field. An important work is represented by affyPara [8] that is a Bioconductor package for parallel preprocessing of Affymetrix microarray data. It is freely available from the Bioconductor project. Compared to affyPara, our approach presents three main advantages: (i) the possibility to realize more summarisation scheme such as Plier, (ii) the easily extension to newer SNP arrays, (iii) it does not require the installation of Bioconductor platform.

The rest of the paper is structured as follows. Section 2 discusses the sequential preprocessing of Affymetrix data, Section 3 presents a parallel preprocessing algorithm and Section 4 presents a case study discussing the preprocessing of Affymetrix data. Finally Section 5 concludes the paper and outlines future work.

2 Preprocessing of Affymetrix Microarray Data

The preprocessing of microarray data can be structured as a pipeline of sequential steps, as data feeds along next steps, it becomes more and more refined. Each step can be performed by using different algorithms that are designed for each chip of different platforms. Usually, software tools are designed ad hoc for a vendor and are tailored to the properties of data; they do not allow the preprocessing in an general way.

[1] www.affymetrix.com

From a technical point of view CEL files store the results of the intensity calculations obtained from the pixel values of the raw image files (also referred as DAT files). The structure of the current version of CEL files (see Figure 1) is represented by a binary file where values are stored in little-endian format.

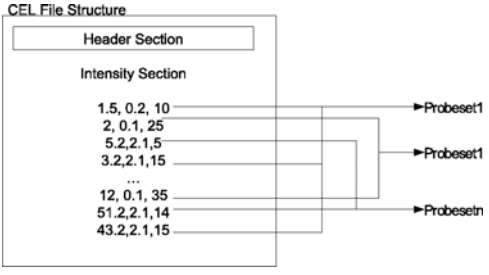


Fig. 1. The internal structure of a CEL file

Preprocessing of Affymetrix arrays start from multiple CEL files and produces as output a matrix whose element (i, j) represents the intensity of the i-th gene in the j-th sample. Preprocessing can be structured as: (i) background correction and quality control, (ii) normalisation, (iii) summarisation, and (iv) annotation.

Background correction aims to identify the background noise and to remove it [7]. Normalisation consists of reducing the bias among chips and within different regions of the same chip [2], aiming at removing non-biological variability within a dataset. Both biological and technical variations introduce artifacts and variability into the system. Summarisation combines multiple preprocessed probe intensities to a single expression value. All arrays employ more than one probe for each genes as introduced before. Summarisation takes into account all of the probes for the same gene and averages them by enhancing the signal-to-noise ratio.

All of these algorithms are based on several assumptions on the data distribution and they require a set of specific libraries in order to correctly access binary data. For Affymetrix arrays, there exist two main summarisation algorithms: Robust Multi-array Average (*RMA*) algorithm [6], and Probe Logarithmic Intensity Error (*PLIER*) algorithm [1].

Finally, a process known as *annotation*, associates to each probe its known annotations such as Gene Symbol or Gene Ontology [4] by matching probes to public databases or knowledge bases. Often annotation files are provided by the chip manufacturer and contain different levels of annotation, e.g. database identifier, description of molecular function, associated protein domains.

Affymetrix Power Tools are a set of command line programs provided by Affymetrix that implements different algorithms for preprocessing Affymetrix arrays. In particular apt-probeset-summarize implements summarization and normalization methods (e.g. RMA- RMA-SKETCH and PLIER) for expression arrays. APT-Tools are able to read a set of *CEL* files and produce a data matrix.

3 A Parallel System for Preprocessing Affymetrix Microarray Data

The preprocessing of Affymetrix files can be easily executed in a parallel way by considering the structure of RMA, RMA-SKETCH and PLIER algorithms. All the algorithms share a common execution scheme: (a) they initially find a raw value for the expression of the gene i starting from the binary file, (b) they merge all the genes of the chips in a single matrix, (c) then they normalize the expression value of each gene by considering the value of the gene i in the other arrays of the considered dataset. From this scenario the parallelization of the step (c) can be done by considering the split of the resulting matrix in different arrays.

A suitable architecture for the parallelisation of this algorithm is depicted in Figure 2. The master node is responsible for the invocation of the worker nodes, each worker node receives as input a copy of the whole dataset and a list of probeset to be preprocessed, then it performs their summarisation and normalization. Finally, it sends to the master node the subset of summarised probesets. Master node merges together results and builds the resulting matrix.

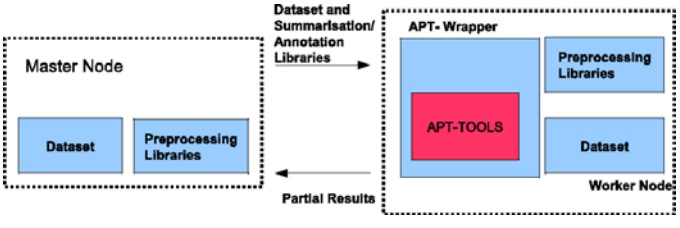


Fig. 2. Architecture of the system

Considering the APT-Tools, the parallelization can be performed by invoking APT on a subset of probesets specified in an input file (e.g. subset.txt), as explained in the following. The rest of the Section explains the execution of these steps on a cluster architecture as depicted in Figure 3:

- **Invocation of Master Node.** The Master node computes the number of needed nodes and the list of probeset for each node. We consider a simple load distribution strategy that assigns to each node the same load. Let N_{pbs} the number of probesets of a chip, and Nwk the number of desired workers, each worker will process $\frac{N_{pbs}}{Nwk}$ probes.
- **Data Distribution.** The Master Node replicates the whole dataset on each worker node and sends the list of probesets to be considered to each node.
- **Parallel execution of APT.** Each Worker node executes the APT-Tool on a subset of probesets and sends the resulting output to the Master node.
- **Integration of results.** The Master node, after the completion of each job collects the results and builds the final data matrix.

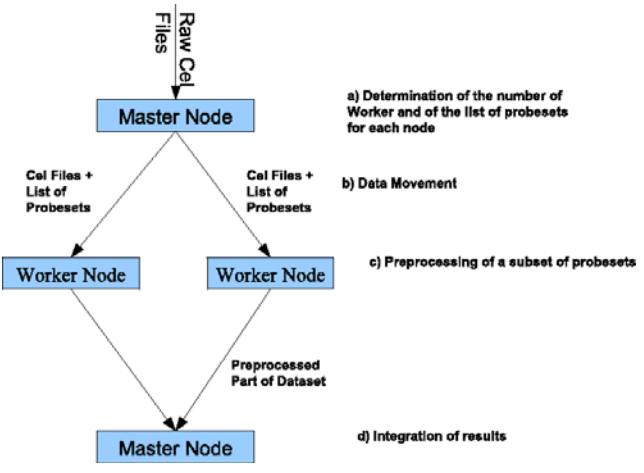


Fig. 3. Flow of data for the execution of parallel preprocessing

The following fragments of pseudo code show the main steps executed, respectively, by the master node and the worker nodes.

---

procedure: Master Node
Input: Dataset // Microarray Dataset (CEL Files), each file contains N_{pbs} probesets,
Input: N_{wk} **// Number of Workers**
Input: L_a**,** L_s**: Annotation and Summarisation Libraries**
begin
Computation of jobs and generation of the list of probesets for each node,
Each node will receive, $\frac{N_{pbs}}{N_{wk}}$ probesets; $Probeset[j], j = 1...N_{wk}$:Array;
Replication of Data and Libraries for each node
Delivery to each node the corresponding list of probes $Probeset[j]$ and the preprocessing parameters
FOR EACH $Workerj : send(Probeset[j], Parameters[j])$ $j = 1...N_{wk}$
Collection of partial results
Fusion of partial results
end

---

procedure: Worker Node
Input: (D,L_a,L_s,Preprocessed[j],Parameters[j])
// Preprocessing of dataset
$Results[j]$= $RunAPT(D, L_a, L_s, Probeset[j], Parameters[j])$
// Partial Results are moved to the Master Node
send($Results[j], MasterNode$)

The Master Node receives the preprocessing requests (a dataset and the number of desired workers) from the user, then replicates the dataset on the available worker nodes and sends to each of them the list of the subset of probesets to summarize and normalize. Each node will preprocess only a subset of probesets following the commands of the Master Node. With respect to the traditional invocation of APT-Tools, each Worker Node node preprocess only a subset of probesets so it employs trivially a smaller time. Each Worker Node is able to invoke an instance of the APT-Tools by using an ad hoc designed APT-Wrapper. After the jobs completion, the Master Node reads the output files sent by the worker nodes, and merges them together.

4 Case Study

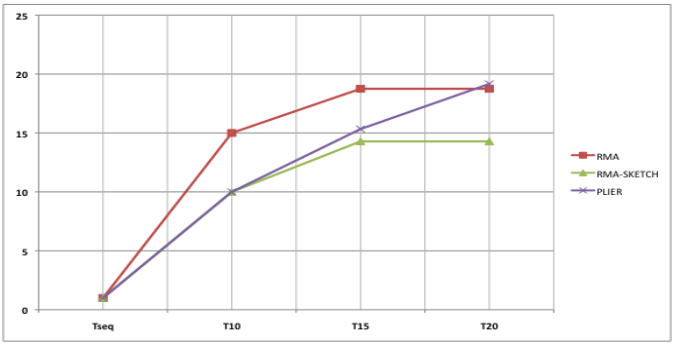
This section demonstrates the ability of the proposed system to preprocess microarray data in a parallel way showing a considerable improvement in term of consumed time. We considered a publicly available dataset of 21 Affymetrix HumanGene 1.0 st arrays and we split this dataset onto three subsets respectively of 7 (Dataset1 hereafter), 14 (Dataset2 hereafter), and 21 arrays (Dataset3 hereafter). The dimension of three datasets are respectively: 75Mb, 150Mb, and 220Mb.

For each dataset we measured the sequential execution time for preprocessing considering three possible preprocessing schemas: RMA, RMA-SKETCH and PLIER. Then we considered the execution time for the parallel preprocessing. The execution time of each preprocessing has to consider the overhead of data-movement among nodes as well as the execution time of the successive merging of the jobs of the nodes. The measured transfer time is considerably lower than the execution time, so we do not include it in this discussion. We measured the execution times of such preprocessing respectively considering an increasing number of worker nodes. Table 1, shows the comparison among the sequential, (T_{seq}) and parallel execution of preprocessing, (T_{10}, T_{15}, and T_{20}), for each dataset considering the RMA, RMA-SKETCH and PLIER algorithms.

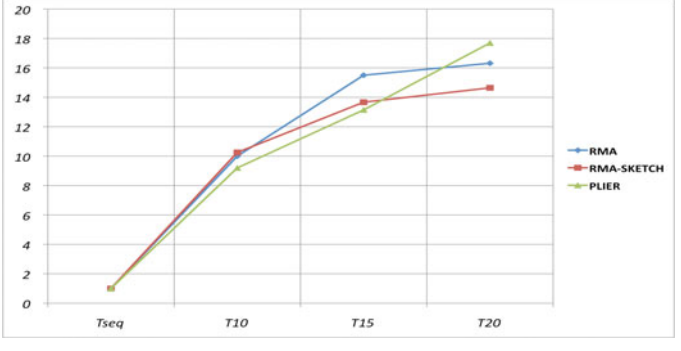
Table 1. Execution Times for Datasets

Algorithm - Times	Dataset1				Dataset2				Dataset3			
	T_{seq}	T_{10}	T_{15}	T_{20}	T_{seq}	T_{10}	T_{15}	T_{20}	T_{seq}	T_{10}	T_{15}	T_{20}
RMA	310	31	20	19	150	10	8	8	480	50	30	29
RMA-SKETCH	205	20	15	14	100	10	7	7	295	29	20	18
PLIER	460	50	35	26	230	23	15	12	710	80	46	46

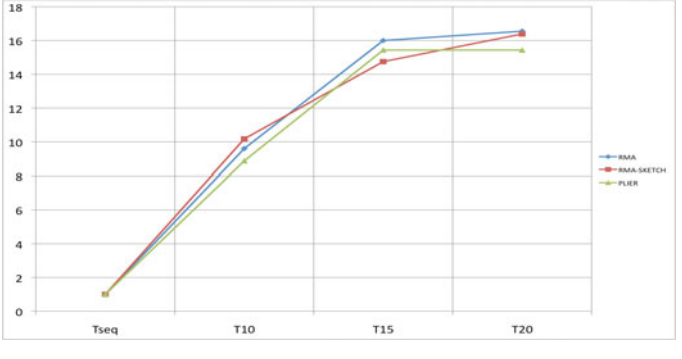
We measured also the speed-up for each dataset measuring the ratio $Speed-Up = \frac{T_{seq}}{T_{par}}$, where T_{par} is the time for the parallel processing considering the different number of workers. Figures 4(a), 4(b), and 4(c) depict the speed-up respectively for Dataset 1, 2, and 3.



(a) Dataset 1



(b) Dataset 2



(c) Dataset 3

Fig. 4. Speed-Up for Dataset 1,2, and 3

As evidenced in all the figures the parallel execution initially presents a super-linear [5] speed-up. This is due to the decrease of size of the number of probesets and to the consequently allocation in main memory of the problem. The super-linear speed-up is only present when 10 workers are used.

5 Conclusion

In this work we presented a system for the parallel preprocessing of microarray data based on a distributed architecture. The proposed system is able to collect data from user, run different preprocessing algorithms in a parallel way, and integrate results. Early experiments with the proposed system using publicly available Affymetrix data showed improvements with respect to execution times. This gain will be more evident considering the constant increase of the volume of data. Future work will focus on the complete realization of the system.

References

1. Affymetrix: Guide to probe logarithmic intensity error (plier) estimation, http://www.affymetrix.com
2. Fujita, A., Sato, J.R., Rodrigues, L.O., Ferreira, C.E., Sogayar, M.C.: Evaluating different methods of microarray data normalization. BMC Bioinformatics 7, 469 (2006), http://dx.doi.org/10.1186/1471-2105-7-469
3. Guzzi, P.H., Cannataro, M.: mu-CS: An extension of the TM4 platform to manage Affymetrix binary data. BMC Bioinformatics 11(1), 315+ (2010), http://dx.doi.org/10.1186/1471-2105-11-315
4. Harris, M.A., et al.: The gene ontology (go) database and informatics resource. Nucleic Acids Res. 32(Database issue), 258–261 (2004)
5. Helmbold, D.P., McDowell, C.E.: Modeling speedup (n) greater than n. IEEE Trans. Parallel Distrib. Syst. 1(2), 250–256 (1990)
6. Irizarry, R.A., Hobbs, B., Collin, F., Beazer-Barclay, Y.D., Antonellis, K.J., Scherf, U., Speed, T.P.: Exploration, normalization, and summaries of high density oligonucleotide array probe level data. Biostat. 4(2), 249–264 (2003), http://biostatistics.oxfordjournals.org/cgi/content/abstract/4/2/249
7. Rocke, D., Durbin, B.: A model for measurement error for gene expression arrays. J. Comput. Biol. 8(6), 557–569 (2001), http://citeseer.ist.psu.edu/447387.html
8. Schmidberger, M., Vicedo, E., Mansmann, U.: affypara. a bioconductor package for parallelized preprocessing algorithms of affymetrix microarray data. Bioinform. Biol. Insights 30(22), 83–87 (2009)