BioDCV: a grid-enabled complete validation setup for functional profiling

Silvano Paoli [1] Davide Albanese Giuseppe Jurman [2]
Stefano Merler Roberto Flor Annalisa Barla [3]

ITC-irst, Trento, Italy

James F. Reid

INT/IFOM-FIRC, Milano, Italy

Stefano Cozzini

INFM/Democritos, Trieste, Italy

Cesare Furlanello *

ITC-irst, Trento, Italy

Abstract

BioDCV is a distributed computing system for the complete validation of gene profiles. The system is composed of a suite of software modules that allow the definition, management and analysis of a complete experiment on DNA microarray data. The BioDCV system is characterized by high throughput computing needs in order to build predictive classification models and extracting the most important genes. In this paper we describe the porting of BioDCV to computational grids running LCG [13] /EGEE [12] middleware. Performances are evaluated on a set of 6 cancer microarray datasets of different sizes and complexity, and compared with results on a standard Linux cluster facility.

Key words: Bioinformatics, pattern recognition, feature selection, DNA microarray

1 Introduction

BioDCV is a gene profiling system for the molecular diagnosis of disease, implementing the development of predictive classifiers based on gene signatures within the complete validation protocol. Outcomes of the system are a predictive model, the straighforward evaluation of its accuracy, lists of genes ranked for importance, identification of patient subtypes. Molecular oncologists from medical research centers and collaborating bioinformaticians are currently the target end-users of BioDCV. It is applied to gene expression studies (e.g. microarrays), one of the fastest growing research area in life sciences.

The current gridification stage of BioDCV is described in this paper. Our application runs in the Egrid [14] virtual organization within the Italian grid facility INFN/Grid-it since March 2005. BioDCV uses all the elements of Globus/EDG/LCG-2 middleware: computing elements, storage elements, some worker nodes. The LCG-2 2.4 middleware was installed in July 2005 (from Egrid live-cd) and it is connected through Egrid's testbed of ICTP located at Trieste (Italy).

In the experiments we describe here, BioDCV was evaluated as a grid application on Cancer6, a set of gene expression cancer datasets with different numbers of cases (min 35 – max 327) and genes (min 1993 – 24481). We also include a comparison with performance on a Linux cluster of 8 Xeon 3.0 GhZ CPU units.

The interest in grid-enabling BioDCV regards the parallelization of tasks and subtasks within gene profiling. Complete validation schemes require an intensive replication of a basic classification task on resampled versions of the dataset. The scheme must ensure that no selection bias effect is contaminating the experiment [1,2]. The cost of this caution is high computational complexity. In a binary classification problem with 50 cases and 20 000 genes, a practitioner willing to implement this scheme will have to develop about 5×10^5 base models, which may become 2×10^6 in case the experiment is replicated with randomized output labels. Although the application can be

* Corresponding author
Email addresses: silpaoli@itc.it (Silvano Paoli), albanese@itc.it (Davide Albanese), jurman@itc.it (Giuseppe Jurman), merler@itc.it (Stefano Merler), flor@itc.it (Roberto Flor), barla@itc.it (Annalisa Barla), james.reid@ifom-ieo-campus.it (James F. Reid), cozzini@democritos.it (Stefano Cozzini), furlan@itc.it (Cesare Furlanello).
[1] Supported by AIRC
[2] Supported by the FUPAT post-graduate project "Algorithms and software environments for microarray gene expression experiments"
[3] Supported by AIRC

modularized, the high throughput input/output generated by loading data and by storing intermediate model parameters and model responses may severely stress a distributed computing facility. It was recently shown that diagnostic quantitaties should also be conserved from the process because novel forms of semi-supervised analysis may be derived, such as the identification of subtypes within cancer groups [3].

To deal efficiently with this data throughput on standard Linux clusters, we integrated the BioDCV system with the SQLite3 database management libraries. The application data are distributed at each job: each node operates on a local copy of the data stored as a SQLite3 file, which is also used to aggregate single job results. After all tasks have been completed, the local db files are retrieved and unified together (usually offsite). It is interesting to test this database-backed solution on a grid facility: this could allow the assessment of the relative computing costs of the fraction of task needed for data management with respect to the effective computing times.

The scientific objective is a first large scale comparison of prognostic gene signatures from breast cancer microarray datasets realized by a complete validation system and run in Grid. The models will constitute a reference experimental landscape for new studies. The gene signatures resulting from each datasets are analyzed for stability and they may be integrated. The comparisons presented in this paper demonstrate the factibility of this approach on public data as well as on original microarray data from IFOM-Firc.

2 BioDCV

Let $\{(x_i, y_i)\}_{i=1}^{N}$ be a training set, where $x_i \in \mathbb{R}^d$ are expression values of d genes and $y_i \in \{-1, 1\}$ are class labels, i.e. clinical outcomes. Let $D^\star = \{i_1, \ldots, i_{d^\star}\}$, with $d^\star \leq d$, be the set of genes potentially related to the clinical outcomes, i.e. we suppose that a function $F : \mathbb{R}^{d^\star} \mapsto \{-1, 1\}$ exists. The aim is first to identify $D^\star$ and then to construct an approximation $F^\star$ of F.

To approximate the unknown function F and to determine the set of the relevant genes $D^\star$, we use Support Vector Machines (SVM) [4] coupled with a feature selection strategy. Our strategy is based on a variant of the classical Recursive Feature Selection (RFE) [5], namely Entropy-based RFE [6]. These studies require a careful and computationally demanding methodology. In particular, it is crucial to use a complete validation process to avoid the selection bias problem [7].

BioDCV is comprised of the modules described below.

2.1.1 The ONF Procedure

Given a training set TR, this procedure is applied to select the optimal number of features based on a ranking method. A resampling procedure is iterated K times, producing each time a $(\mathrm{TR}_k, \mathrm{TS}_k)$ split of TR. A feature ranking procedure is applied to TR_k. Then, n subsets are created with the first F_i

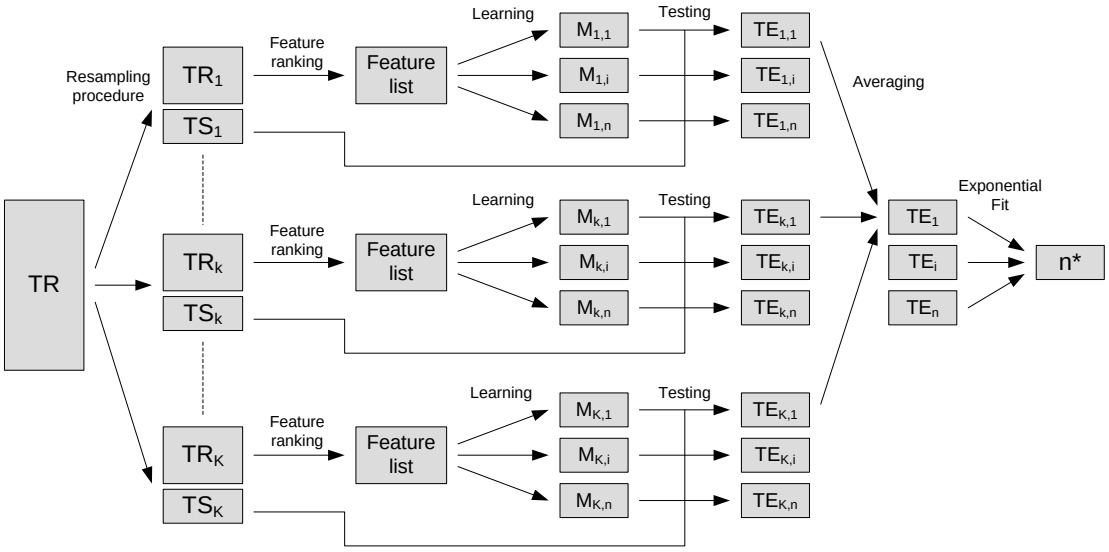

Fig. 1. The ONF procedure

features of the *feature list* (*i.e.* $F_1 = 1, F_2 = 5, F_3 = 10, \ldots, F_n = 1000$). Therefore, for each k a model family ($M_{ki}, \quad i = 1, \ldots, n$) is produced, one for each increasing value of F_i. The M_{ki} models are evaluated on the TS_k test data, computing TE_{ki} test errors, and we obtain the average error curve $\mathrm{TE}_i = \frac{1}{K}\sum_{k=1}^{K} \mathrm{TE}_{ki}$. An exponential fit is applied to TE, and the n^* value leading to saturation in terms of the exponential curve is returned as the ONF result. The complete scheme of the procedure is shown in Figure 1.

2.1.2 The OFS-M Procedure

Given a training set TR, a feature ranking method produces a list of ranked features, from which an optimal feature set OFS of size n^* is selected. Based on ONF procedure, a model M is developed by a suitable learning method. The accuracy of OFS-M is validated by the VAL procedure. A scheme for OFS-M is shown in Figure 2.

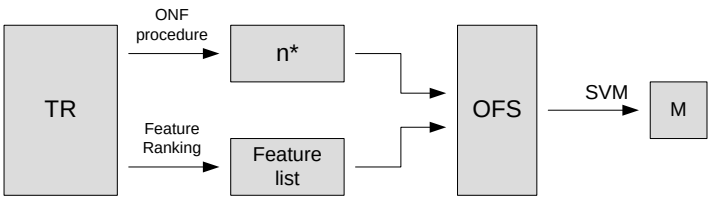

Fig. 2. The OFS-M procedure

2.1.3 The VAL Procedure

In VAL, the OFS-M procedure is validated on B replicated experiments (runs) using a resampling scheme. TE^b error results from testing the model with n^* features on the test set, which minimizes the risk of data overfitting. The procedure returns the expected test error

$$\mathrm{ATE} = \frac{1}{B}\sum_{b=1}^{B}\mathrm{TE}^b.$$

A scheme for the VAL procedure is shown in Figure 3.

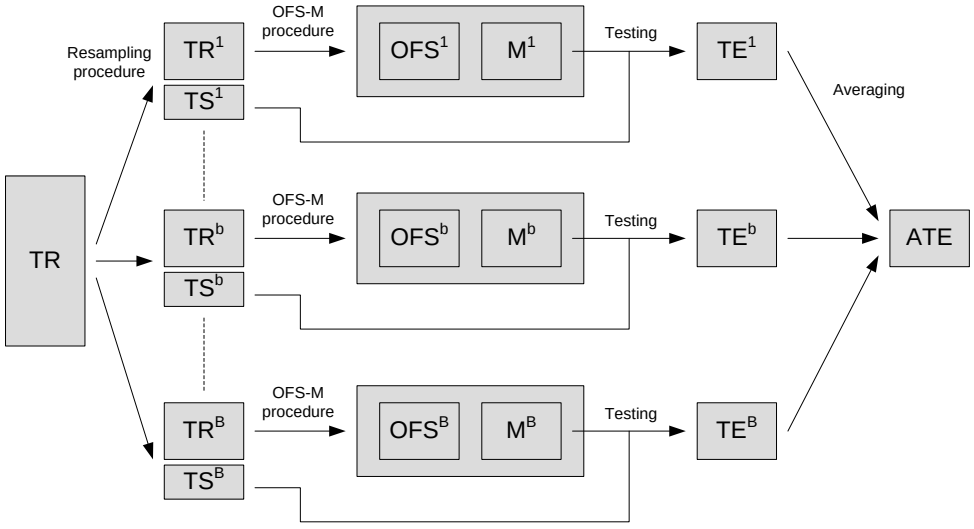

Fig. 3. The VAL procedure

2.1.4 Sample-tracking

The semisupervised procedure implemented in BioDCV is based on an analysis of the effects of the feature selection and ranking process on each individual sample. Given a complete validation setup (such as the one described in [6]), for each sample s we define the sample-tracking profile as the function of the number of features k as $E_s(k) = W(s,k)/N(s)$, where $N(s)$ is the number of runs in which s belongs to the test set and $W(s,k)$ is the number of runs in

which s belongs to the test set and it is wrongly classified when the model is built with k features. The sequences $E_s(k)$ may be studied as an estimate of the classification error as a function of the size of the feature set. Sample curves of easy-to-classify points quickly reach zero, while curves not far from the no-information error rate should correspond to hard-to-classify points. A profile lying systematically above the no-information error rate indicates a typical outlier behaviour. A complete description and examples of the sample-tracking procedure may be found in [3].

2.2 Implementation

A single experiment is formed by three main steps (summarized in 4), which correspond to the three main programs:

`exp` prepares the experiment. Given a configuration file and a matrix of data (and eventually an additional test) it builds the setup database.

`run` performs the validation procedure with $b = \mathit{initial\text{-}run}, \ldots, \mathit{final\text{-}run}$. Given the setup database, it builds one local database, and for each b it builds one flat text file containing the ranking procedure weight. This is the program which works in parallel.

`tracking` joins the setup database and the local databases in a unified database containing the entire experiment and performs the semi-supervised procedure.

Firstly, `exp` builds the setup database. The setup database contains the matrix of the original data and the indices of all b replicates, so that each `run` can build $(\mathrm{TR}_b, \mathrm{TS}_b)$ pairs and continue the experiment.

Secondly, given a setup database, `run` executes the validation procedure for $b = \mathit{initial\text{-}run}, \ldots, \mathit{final\text{-}run}$ and builds a local database. It is important that all the runs $b = 1, \ldots, B$ are completed before joining all the local databases. For example, if we want to execute an experiment with $B = 100$ external replicates, having 10 CPUs, we can submit 10 `run` with $\mathit{initial\text{-}run} = 1, 11, 21, 31, \ldots, 91$ and $\mathit{final\text{-}run} = \mathit{initial\text{-}run} + 9$. Moreover, `run` returns a flat text file for each b: this file contains the weights generated by the classifier that are associated with each feature for each feature step.

At the end, `tracking` joins the local databases and inserts them into the setup database creating the complete database; it then performs the sample-tracking procedure. The *replicates* table of the setup database is replaced by the *replicates* table union of the local databases.

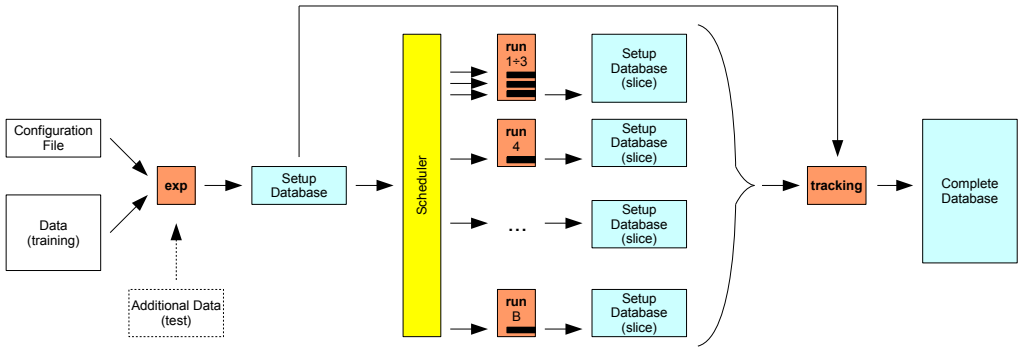

Fig. 4. The VAL procedure

2.3 Porting to the Grid

To guarantee speed, slim and robust code, and relational access to data and model descriptions, BioDCV was written in C and interfaced with SQLite (http://www.sqlite.org), which supports concurrent access and transactions useful in a distributed environment where the learning, tuning and evaluation tasks may be replicated for up to a few million models. We recently ported our application to grid systems, namely the Egrid [14] computational grids. The Egrid infrastructure is based on Globus/EDG/LCG2 middleware and is integrated as an independent virtual organization within the Grid.it, the INFN production grid. The porting requires just two wrappers, one shell script to submit jobs and one C MPI program. Three basic elements of a Grid.it infrastructure are used:

`Storage Element (SE)` stores user data in the grid and makes it available for subsequent elaboration;

`Computing Element (CE)` is where user tasks are delivered for elaboration and this is usually a front end to several Worker Node (WN);

`Worker Node (WN)` machines where the grid user programs are actually executed.

When the user submits a BioDCV job to the grid, the grid middleware looks for the CE and the WNs required to run the parallel program. As soon as the resources (CPUs in WNs) are available, the shell script wrapper is executed on the assigned CE. This script distributes the microarray dataset from the SE to all the involved WNs. It then starts the C MPI wrapper which spawns several instances of the BioDCV program itself. When all BioDCV instances are completed, the wrapper copies all outputs including model and diagnostic data from the WNs to the starting SE. Finally, the process outputs are returned, allowing the reconstruction of a complete data archive for the study.

3 Experimental design and evaluation metrics

In this section, we present in detail the two experiments we designed to measure the performance of the BioDVC parallel application in two different computing available environments: a standard Linux cluster and a computational grid.

In Benchmark 1, we study the scalability of our application as a function of the number of CPUs. Two DNA microarray datasets are considered. The benchmark is executed on a Linux clusters formed by 8 Xeon 3.0 CPUs and on the EGEE grid infrastructure ranging from 1 to 64 Xeon CPUs.

In Benchmark 2, we characterize the BioDCV application with respect to different datasets, i.e. for different d and N values, where d is the number of features and N the number of samples for the complete validation experiment. A task for each single dataset is executed on the EGEE grid infrastructure using a fixed number of CPUs.

Performance is evaluated in terms of T_{tot} (total execution time), defined as the elapsed time of a full experiment (i.e., the sequential run of `exp` plus `run` plus `tracking`). This elapsed time comprises several different contributions, as stated in the following formula:

$$T_{tot} = L_i + U + E_0 + E_g + D + M + S \tag{1}$$

with:

L_i Experiment setup time at local submitter site. It includes the configuration of setup database, and data preparation. (i.e. `exp`)

U Time spent for uploading data and application to the grid facility, including delivery on Computing Element (CE).

E_0 Time elapsed before computing. This includes time spent waiting for node availability and time for data distribution onto worker nodes (WNs) within the facility.

E_g Computing time: a function of the number of obtained CPUs, characteristics of CPUs, algorithm parameters (`run`). It includes user time E_u, which is the time that the machines exclusively devote to the application.

D Time spent for data retrieval and download. This includes copying all results (SQL files) from the WNs to the starting SE, and their transfer to local site.

M Data merging time: reunification of retrieved data into an unique DB at the local site.

S Semisupervised analysis time: post-processing analysis at local site. The analysis provides the sample-tracking curves described in [3].

We note that E_0 (Time elapsed before computing) can be split in two contributions:

$$E_0 = E_0^q + E_0^l \tag{2}$$

where:

E_0^q Queueing time at grid site: the task is scheduled and waiting.
E_0^l Initial latency time at grid site.

It is therefore important to evaluate T_{tns} (Effective execution time), where the time spent on waiting is discarded:

$$T_{tns} = T_{tot} - E_0^q \tag{3}$$

and the E_l (Global latency time), the portion of non-computing time within the effective execution time, i.e. the overhead caused by the grid middleware:

$$E_l = T_{tns} - E_g \tag{4}$$

4 Results

Data presented here refer generally to one single run, due to resource availability on the computational grid. To estimate running variability, we repeat a few simulations for the smallest datasets: the resulting error bar is within 5%.

4.1 Benchmark 1: measuring speed-up

Benchmark 1 was performed on the two datasets: LiverCanc [8] (213 samples, ATAC-PCR, 1993 genes) and PedLeuk [9] (327 samples, Affymetrix, 12625 genes).

Table 1
Results for LiverCanc on MPBA cluster

N	L_i	U	E_0	E_g	E_u	D	M	S	T_{tot}
1	111	0	2	120136	120047	5	14	1278	121547
2	111	0	4	61039	60993	5	14	1278	62451
4	111	0	8	28267	28248	5	15	1279	29686
8	111	0	16	15318	15308	5	14	1278	16742

Table 2
Results for LiverCanc on Grid

N	L_i	U	E_0	E_g	E_u	D	M	S	T_{tot}	T_{tns}	E_0^l	E_0^q	E_l
1	111	36	234	102656	84709	355	8	1240	104641	104439	32	202	1783
2	111	36	225	42688	42573	355	10	1240	44665	44478	38	187	1790
4	111	36	87104	23255	21395	352	10	1240	112109	25074	75	87029	1819
8	111	36	155733	16098	10814	396	12	1240	173627	18075	184	155549	1977
16	111	36	177120	8874	6251	424	12	1240	187818	10998	300	176820	2124
32	111	36	279874	4296	2686	608	10	1240	286175	6894	593	279281	2598

Table 3
Comparing E_l for LiverCanc on Grid and MBPA Cluster

N	E_l Grid	E_l Cluster
1	1783	1411
2	1790	1412
4	1819	1419
8	1977	1434
16	2124	-
32	2598	-

In Tab. 1 and 2, the values for the terms in Eq. (1) for increasing number of CPUs (column N) are reported for the LiverCanc dataset on the MPBA cluster and on the grid respectively. The main difference between the two Tables is that while E_0 is negligible on the MPBA cluster, it increases exponentially with the number of CPUs on the grid. This effect is entirely due to the E_0^q time. Infact, when asking for 32 CPU MPI jobs on resources offered according to a "best effort policy", almost all the time is spent on queue.

The BioDCV core application (`run`) shows almost linear behaviour on execution time with respect to the number of CPUs (column E_G) on both the MPBA cluster and the computational grid.

It is also interesting to compare the Global latency time (E_l) for the grid against the latency time of the cluster. Since the MPBA cluster is a dedicated resource, the time spent on queue is zero. This means that on the cluster $E_l = T_{tot} - E_G$. These values are summarized in Table 3.

The results indicate that the overhead caused by the grid increases the total latency time by about 20%. The difference is almost entirely given by transferring data to-and-from local resources and the grid, as well as to-and-from, CE and WN within the grid.

Moreover, we note that latency constantly increases as the number of processors increases. This makes it impossible to scale the full experiment over a large number of processors. However, this is a second order problem because, at the moment, simulations with a large numbers of CPU is unfeasible due to the large queue times associated with them.

Table 4
Results for PedLeuk on MPBA cluster

N	L_i	U	E_0	E_g	E_u	D	M	S	T_{tot}
1	523	0	2	479577	479242	10	25	2584	482721
2	523	0	4	320534	320304	10	26	2584	323682
4	523	0	8	158337	158221	10	27	2584	161489
8	523	0	16	82855	82776	10	24	2584	86013

Table 5
Results for PedLeuk on Grid

N	L_i	U	E_o	E_g	E_u	D	M	S	T_{tot}	T_{tns}	E_0^l	E_0^q	E_l
4	523	232	3157	134461	115368	783	37	2575	141768	138695	64	3073	4234
8	523	232	62836	88622	65536	421	35	2575	155243	92574	167	62669	3952
16	523	232	250044	41047	27961	810	38	2575	295268	45567	330	249714	4520
32	523	232	333852	23082	14317	845	45	2575	361154	27831	529	333323	4749

Similar observations hold for the PedLeuk dataset (see Tables 4 and 5).

A graphical representation of the linear speed-up (we obtain a coefficient of around 1 for all cases) observed on both LiverCanc and PedLeuk datasets is shown in Fig. 5. The speed-up factor for n CPUs is defined as the user time E_u for one CPU divided by the user time E_u for n CPUs. Note that in the grid case, the baseline for the PeudLeuk dataset is the user time for 4 CPUs.

It is also illuminating to carefully check the E_g time spent by the `run` on cluster and grid resources. E_g is actually the wall-time, but we also measured user and system time by means of the /usr/bin/time function available on the systems. A fraction of time (E_g - E_u) is not directly consumed by our program: this time is related to the actual load present on the machine (WN) where the program is executed. This time is negligible for the cluster resource, where the node is fully dedicated to the application and no other process is competing for resources. On grid, the scenario is more complicated: in principle the WNs are fully dedicated to the parallel runs and the situation should resemble the cluster one. This is true only in some cases (e.g in the case of two CPUs and four uniform CPU runs for the LiverCanc example), while in other cases differences in time are considerable. It is not clear at the moment why this happens and this aspect needs to be further investigated.

4.2 Benchmark 2

Benchmark 2 was run on a suite of six microarray datasets (Cancer6): LiverCanc, PedLeuk, BRCA [4] (62 samples, cDNA, 4000 genes), Sarcoma [5] (35 sam-

[4] original data from IFOM
[5] original data from IFOM

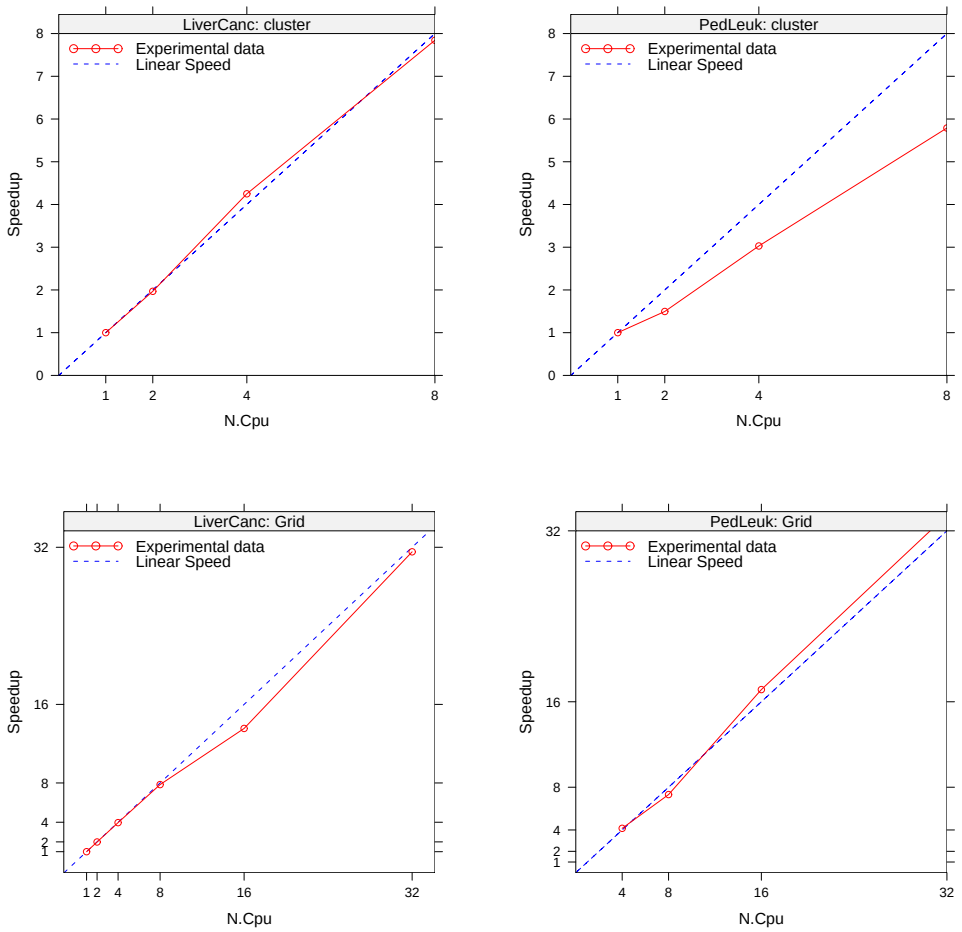

Fig. 5. Speed-up for Experiment 1

ples, cDNA, 7143 genes), Wang [10] (286 samples, Affymetrix, 17816 genes), Chang [11] (295 samples, cDNA, 25000 genes).

In Table 6, the quantities in Eq. (1) are reported. Time in seconds (s) refers to grid jobs executed on a fixed number of CPUs (32). In Fig. 6, the effective execution time T_{tns} is plotted versus the columns of the Table. It can be observed that T_{tns} increases linearly with the dataset dimension. In particular, the discriminating factor is the product of number of genes and number of samples (dN in Fig. 6 and in Tab. 6).

We can conclude that employing the grid facility is quite appealing whenever the queueing policy guarantees reasonable queueing times.

Table 6
Experiment 2 on Cancer6

	Dataset name	dN x 10e-7 ⋆	DB (MB)	L_i	U	E_g	D	M	S	T_{tns}
1	PL	4.1 (327,12625)	32	522	231	23082	845	45	2575	27831
2	Morishita	0.4 (213,1993)	3.7	111	36	4296	608	10	1240	6894
3	BRCA	0.2 (62,4057)	2.2	90	24	635	365	7	810	2534
4	Sarcoma	0.3 (35,7143)	2.2	149	25	386	356	9	1343	2887
5	Wang	0.5 (286,17816)	40	920	547	132279	947	38	2725	138355
6	Chang	0.7 (295,24481)	57	1399	625	107471	1061	56	3254	114546

⋆ numbers of samples and genes in parentheses

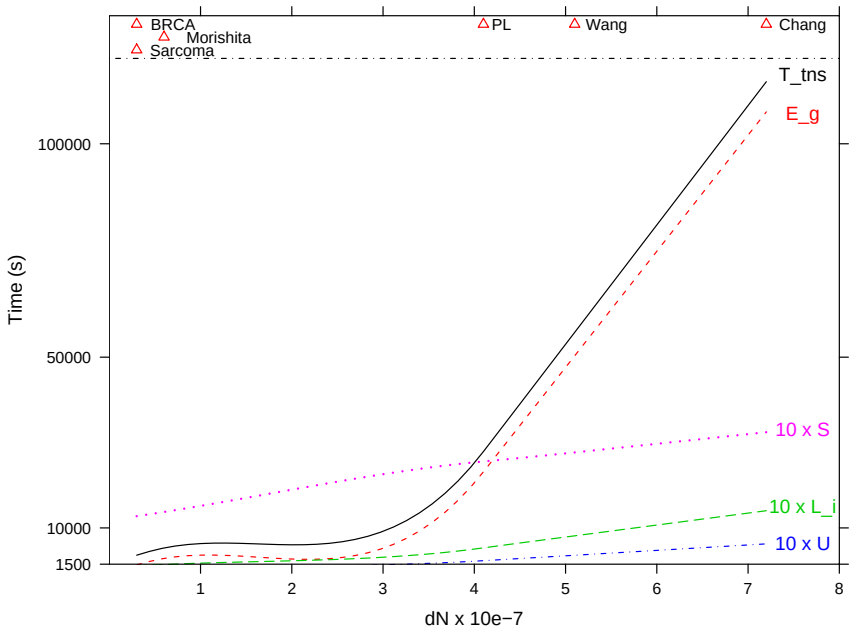

Fig. 6. Different measures of times vs. footprint of Cancer6

5 Discussion and Conclusions

The data presented in the previous section, which sum up to 139 CPU days within the Egrid [14] infrastructure, allows some observations about the general behavior of the BioCDV system on LCG [13]/EGEE [12] computational grids. The system actually fits well in the environment provided by the computational grid considering that it required little integration work in order to be ported to such an environment. The performance penalty payed with respect to a standard parallel run performed on local cluster is limited and it is mainly due to data transfer. This aspect could probably be improved if the advanced data management techniques included in the application were coupled with data grid tools more efficiently.

At the moment, the main drawback in using the grid is the long waiting time in case of the largest parallel simulations. In the case of the Egrid production grid, this issue is considerable due to the fact that the Egrid VO can access only one large grid facility site (the INFN.IT Padova site), where resources are made available on a best effort basis. The planned experiment is currently aiming at a very competitive scientific result and thus the experimental phase requires stability in resource availability.

Current limitations may be overcome by replacing the MPI approach with an ad-hoc software layer implementing a client-server model that submits N different single CPU jobs instead of one single job asking for N CPUs. We are currently investigating if and how this method can be applied to BioCDV systems.

BioDCV is an open source application and it is currently distributed under GPL. (SubVersion repository at `http://biodcv.itc.it`).

References

[1] C. Ambroise, G. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, Proc. Natl. Acad. Sci. USA. 99 (10) (2002) 6562–6566.

[2] R. Simon, M. Radmacher, K. Dobbin, L. McShane, Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification, J Natl Cancer Inst 95 (1) (2003) 14–18.

[3] C. Furlanello, M. Serafini, S. Merler, G. Jurman, Semisupervised learning for molecular profiling, IEEE/ACM Transactions on Computational Biology and Bioinformatics 2 (2) (2005) 110–118.

[4] C. Cortes, V. Vapnik, Support–vector networks, Machine Learning 20 (3) (1995) 273–297.

[5] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning 46 (2002) 389–422.

[6] C. Furlanello, M. Serafini, S. Merler, G. Jurman, Entropy-Based Gene Ranking without Selection Bias for the Predictive Classification of Microarray Data, BMC Bioinformatics (4) (2003) 54.

[7] R. Simon, E. Korn, L. McShane, M. Radmacher, G. Wright, Y. Zhao, Design and Analysis of DNA Microarray Investigations, Statistics for Biology and Health, Springer, 2004.

[8] J. Sese, Y. Kurokawa, M. Monden, K. Kato, S. Morishita, Constrained clusters of gene expression profiles with pathological features., Bioinformatics 20 (17) (2004) 3137–3145.

[9] Khan, J. et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial networks. Nature Medicine 7, 673-679 (2001).

[10] Y. Wang, J. Klijn, Y. Zhang, A. Sieuwerts, M. Look, F. Yang, D. Talantov, M. Timmermans, M. M. van Gelder, J. Yu, T. Jatkoe, E. Berns, D. Atkins, J. Foekens, Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer, Lancet 365 (9460) (2005) 671–679.

[11] H. Chang, D. Nuyten, J. Sneddon, T. Hastie, R. Tibshirani, T. Sorlie, H. Dai, Y. He, L. van't Veer, H. Bartelink, M. van de Rijn, P. Brown, M. van de Vijver, Robustness, scalability, and integration of a wound-response gene expression signature in predicting breast cancer survival, PNAS 102 (10) (2005) 3738–3743.

[12] Enabling Grids for E-sciencE (EGEE), http://public.eu-egee.org/ .

[13] Worldwide LHC Computing Grid, http://lcg.web.cern.ch/LCG/ .

[14] Egrid project, http://www.egrid.it .