DSC 2003 Working Papers
(Draft Versions)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

PVM kriging with R

Dr. Albrecht Gebhardt
Institut für Mathematik
Universität Klagenfurt
Universitätsstr. 65-67, A 9020 Klagenfurt
albrecht.gebhardt@uni-klu.ac.at

Abstract

Kriging is one of the most often used prediction methods in spatial data analysis. This paper examines which steps of the underlying algorithms can be performed in parallel on a PVM cluster. It will be shown, that some properties of the so called kriging equations can be used to improve the parallelized version of the algorithm. The implementation is based on R and PVM. An example will show the impact of different parameter settings and cluster configurations on the computing performance.

1 The classical form of kriging

One of the aims of geostatistical analysis is the prediction of a variable of interest at unmeasured locations. The prediction method which is used most often is kriging. It is based on the concept of so called regionalized variables $Z(\underline{x})$ with $\underline{x} \in D \subset \mathbb{R}^d, d = 2, 3$. Analysis usually starts with a spatial dataset consisting of measurements $Z(\underline{x}_i), i \in I$ at a grid of observation points $\underline{x}_i$. These values are now treated as a realization of the underlying stochastical process $Z(\underline{x})$ and modeled with a usual regression setup $Z(\underline{x}) = m(\underline{x}) + \varepsilon(\underline{x}), \mathbb{E}(\varepsilon(\underline{x})) = 0$. Of course it is not possible to make inference from only one realization. The idea of regionalized variables is now to partition the region D into nearly independent parts and to use them as different realizations. This is only valid if a stationarity assumption holds:

$$m(\underline{x}) \quad = \quad const,\ \underline{x} \in D \tag{1}$$

$$\mathrm{Cov}(Z(\underline{x}_i), Z(\underline{x}_j)) \quad = \quad C(\underline{h}),\ \underline{h} = Z(\underline{x}_i) - Z(\underline{x}_j),\ \underline{x}_i, \underline{x}_j \in D \tag{2}$$

That means that mean and covariance function $m(\underline{h})$ and $C(\underline{h})$ are assumed to be translation invariant. A more general assumption, called intrinsic stationarity, only demands that the variance of the increments of $Z(.)$ has to be invariant with respect to translation:

$$\mathrm{Var}(Z(\underline{x}_i) - Z(\underline{x}_j)) \quad = \quad 2\gamma(\underline{h}),\ \underline{h} = \underline{x}_i - \underline{x}_j,\ \underline{x}_i, \underline{x}_j \in D \tag{3}$$

Equation (3) introduces also the semivariogram $\gamma(\underline{h})$. It is connected with the covariance function via $\gamma(\underline{h}) = C(0) - C(\underline{h})$ if $C(\underline{h})$ exists. In simple cases $\gamma(.)$ depends only on $\mid \underline{h} \mid = h$. This leads to isotropic variograms and covariance functions. Both will later be used to determine the system matrix in the prediction step. Therefore it is necessary to estimate the semivariogram. The usual estimator in the isotropic case $\gamma(\underline{h}) = \gamma(h)$ with $h = |\underline{h}|$ is

$$\hat{\gamma}(h) \quad = \quad \frac{1}{2\,N(h)} \sum_{\underline{x}_i - \underline{x}_j \in L(h)} (Z(\underline{x}_i) - Z(\underline{x}_j))^2 \tag{4}$$

where $L(h)$ is an interval (lag) $[h_l, h_u]$ containing h and $N(h)$ denotes the number of pairs $(\underline{x}_i, \underline{x}_j)$ falling into the lag $L(h)$. Semivariogram functions have the property of negative semi-definiteness. This makes it necessary to fit a valid semivariogram function to the estimated $\hat{\gamma}(h)$. Often used semivariograms models are e.g. the spherical or the exponential model. Most semivariogram models are parametrized by the so called nugget parameter (it describes discontinuities at the origin), a range parameter (equals the correlation radius) and the sill parameter (maximum semivariogram value taken outside of the range).

The estimator used by kriging for prediction at a location $\underline{x}_0 \in D$ has the linear form

$$\hat{Z}(\underline{x}_0) \quad = \quad \underline{\lambda}^\top \underline{Z} \tag{5}$$

with the kriging weights $\underline{\lambda} = (\lambda_i)_{i \in I}$ and the data vector $\underline{Z} = (Z(\underline{x}_i))_{i \in I}$. Depending on the modeling assumptions regarding $m(\underline{x})$ two variants of kriging can be differentiated: Ordinary kriging which uses a constant trend model $m(\underline{x}) = const$ and universal kriging which models $m(\underline{x})$ with a parameter-linear setup $m(\underline{x}) = \underline{\theta}^\top \underline{f}(\underline{x})$ with a parameter vector $\underline{\theta} \in R^p$ and a set of regression functions $f_j(\underline{x}), j = 1, \ldots, p$. The kriging weights $\underline{\lambda}$ are now chosen by minimizing the prediction variance $\sigma_K(\underline{x}_0) = \mathrm{Var}(\hat{Z}(\underline{x}_0))$ under the condition of unbiasedness $\mathbb{E}(\hat{Z}(\underline{x}_0)) = Z(\underline{x}_0))$, which is also called universality condition in this context and evaluates to $\Sigma_{i \in I} \lambda_i = 1$ for ordinary kriging and $\mathbf{F}^\top \underline{\lambda} = \underline{f}_0$ for universal kriging using the design matrix $\mathbf{F} = (\underline{f}(\underline{x}_i))_{i \in I}^\top$ and $\underline{f}_0 = \underline{f}(\underline{x}_0)$.

Minimizing $\sigma_K(\underline{x}_0)$ finally leads to the kriging equation

$$\begin{pmatrix} \mathbf{C} & \mathbf{F} \\ \mathbf{F}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \underline{\lambda} \\ \underline{\theta} \end{pmatrix} = \begin{pmatrix} \underline{c}_0 \\ \underline{f}_0 \end{pmatrix} \tag{6}$$

with the covariance matrix $\mathbf{C} = (C(\underline{x}_i - \underline{x}_j))_{i,j \in I}$ and the vector $\underline{c}_0 = (C(\underline{x}_0 - \underline{x}_i))_{i \in I}$. Ordinary kriging can be regarded as special case of universal kriging with $p = 1$, $\underline{f}_1(\underline{x}) = 1$ and $\mathbf{F} = \underline{1} = (1)_{i \in I}^{\top}$.

Because usually correlation vanishes if the distance of two points raises it is only necessary to take points within a certain distance around a prediction point into account for building the kriging equation system. This area around prediction points is called search neighborhood and its radius corresponds to the range parameter of the semivariogram. More details can be found in Cressie [1991].

Kriging prediction for the location $\underline{x}_0$ consists now of three steps:

1. Estimation of the semivariogram γ
2. semivariogram model fitting
3. Solving equation (6) to determine the kriging weights $\underline{\lambda}$ and calculation of $\hat{Z}(\underline{x}_0)$.

Usually prediction is carried out not only for one point $\underline{x}_0$. The final output of kriging are prediction maps, that means kriging is performed for each point of a discrete prediction grid. If the goal is to produce high quality prediction maps then dense prediction grids have to be used. This clearly enlarges the computational burden. Therefore it can be helpful to consider alternative approaches like parallel programming in this context.

2 Parallelizing spatial prediction

An algorithm can only be successfully reprogrammed in a parallelized manner if it contains blocks which can be executed independently of each other. Now it is necessary to examine the three steps of kriging prediction mentioned above for their potential to be executed in parallel. It will be assumed that this prediction has to be performed at the points of a discrete grid.

The first step, semivariogram estimation, has only to be executed once per data set and prediction grid. This holds also for the second step, semivariogram fitting. This step includes usually much interaction by the analyst, e.g. choosing the appropriate semivariogram model, which can not be done in an automated manner. But the last step consists of sequential repeated solving of kriging equation systems for each grid point. In this case also the condition of independence of the computations for different grid points holds. So clearly this steps qualifies as a candidate for parallel execution and it will be covered in the next sections.

A simple parallel version of kriging would consist of distributing the task of predicting at the points of the prediction grid to the members of some computing cluster. A supervising process has to manage the distribution of these tasks, to collect the results from the cluster members and to feed the cluster members with new grid points until prediction for all points is done. In an initialization step at the start of prediction all cluster members have to get the whole data set and

semivariogram parameters from the supervising process. This process should also be responsible for some error checking.

Another idea would be to divide the dataset into smaller parts and to feed these parts to the cluster members for prediction at one or more grid locations. This could help in situations where very huge datasets occur, which e.g. can not be handled at once because of resource limitations. The following sections will discussed the first idea in detail.

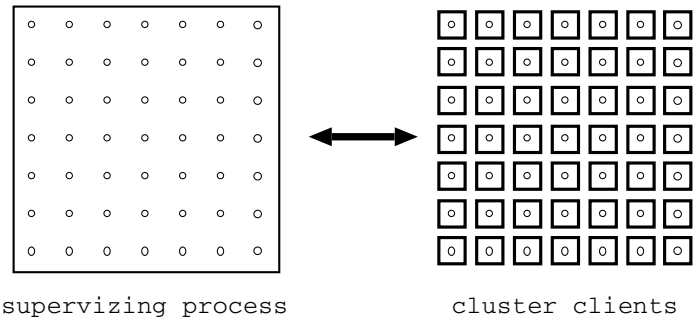

Figure 1: Schematic overview of parallel prediction on a grid

3 Improving parallel kriging computation

It can be helpful to search for properties, which allow a more effective way of parallelization. If we consider the kriging prediction grid and the above mentioned parallel version of the usually sequentially performed prediction on a grid, it turns out that the prediction at grid points, which are located very close to another, will yield very similar results. This is caused by the fact that both points share much of its neighbors and their distances to these neighbors will be very similar. In other words their search neighborhoods will be almost identical. Now this brings up the idea to handle both points $\underline{x}_1$ and $\underline{x}_2$ at once.

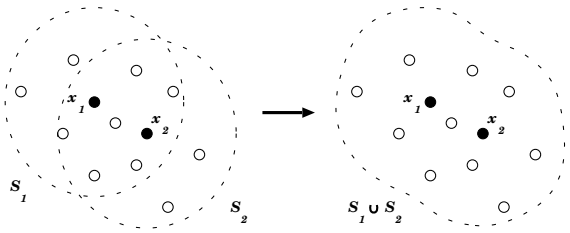

Figure 2: Merging search neighborhoods

Usually both krige systems are set up by establishing individual search neighborhoods $S_i = \{\underline{x}_i \,|\, \|\underline{x}_i - \underline{x}_{0i}\| \leq r\}$ and calculating the components $\mathbf{C}_{S_i}$, $\mathbf{F}_{S_i}$, $\underline{c}_{0i}$ and

$f(\underline{x}_{0i})$ $i = 1, 2$ of the system matrices. After merging both search neighborhoods into $S_i \cup S_2$ and using these points for prediction at $\underline{x}_1$ and $\underline{x}_2$ the system matrices of both kriging systems will coincide. The systems differ only in their right hand sides $\underline{f}_{0,i}$ and of course in their solution vectors $\underline{\lambda}_i$, $i = 1, 2$. Using the covariance matrix $\mathbf{C}_{S_1 \cup S_2}$, the design matrix $\mathbf{F}_{S_1 \cup S_2}$ and the vectors $\underline{c}_{0,i}$ and $\underline{f}(\underline{x}_{0,i})$, $i = 1, 2$, the following krige system can be established:

$$\begin{pmatrix} \mathbf{C}_{S_1 \cup S_2} & \mathbf{F}_{S_1 \cup S_2} \\ \mathbf{F}^{\top}_{S_1 \cup S_2} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \underline{\lambda}_1 & \underline{\lambda}_2 \\ \underline{\theta}_1 & \underline{\theta}_2 \end{pmatrix} = \begin{pmatrix} \underline{c}_{0,1} & \underline{c}_{0,2} \\ \underline{f}(\underline{x}_{0,1}) & \underline{f}(\underline{x}_{0,2}) \end{pmatrix} \tag{7}$$

It consists of a system of linear equations with multiple right hand sides and a solution matrix $\mathbf{\Lambda} = (\underline{\lambda}_1, \underline{\lambda}_2)$. Standard numerical libraries contain algorithms which can solve such systems simultaneously. This implementation uses the Fortran function `DGESV` from LAPACK.

3.1 Tiled grid kriging

The idea of collecting neighboring grid points into jobs, which should be performed by a single cluster member, will be called "tiled grid kriging". The prediction grid is now assumed to be a regular rectangular grid with grid spacing d_x and d_y. This grid is divided into rectangular subsets of equal number $t_x \times t_y$ grid points, called "tiles".

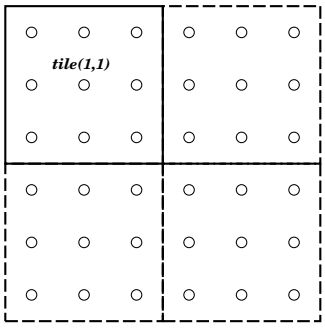

Figure 3: Tiled grid kriging

The idea of equation (7) can be extended to combine the search neighborhoods of m points $\underline{x}_i$. Now the covariance matrix $\mathbf{C}_{\cup_i S_i}$, the design matrix $\mathbf{F}_{\cup_i S_i}$ and the vectors $\underline{c}_{0i}$ and $f(\underline{x}_{0i})$ form the kriging system

$$\begin{pmatrix} \mathbf{C}_{\cup_i S_i} & \mathbf{F}_{\cup_i S_i} \\ \mathbf{F}^{\top}_{\cup_i S_i} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \underline{\lambda}_1 & \cdots & \underline{\lambda}_m \\ \underline{\theta}_1 & \cdots & \underline{\theta}_m \end{pmatrix} = \begin{pmatrix} \underline{c}_{01} & \cdots & \underline{c}_{0m} \\ \underline{f}(\underline{x}_{01}) & \cdots & \underline{f}(\underline{x}_{0m}) \end{pmatrix} \tag{8}$$

Again it is possible to solve this matrix equation simultaneously, e.g. using the `DGESV` routine with appropriate parameters, see Anderson and Others [1999].

The benefit of this approach is that now $m = t_x \times t_y$ points can be handled in one step which will increase computational efficiency significantly. Additionally this

will reduce communication and management overhead in the cluster setup because we can replace m data and result transfers with a single one.

3.2 Border effects

Unfortunately the "tiled grid" approach has the drawback of extending the so called border effect into the interior of the prediction grid. The border effect consist of errors caused by extrapolation at the borders of the prediction area. Now each "tile" can be viewed as such a prediction area and consequently the prediction error raises at the outer points of the tile. As a consequence the tile borders become visible in contour line plots of the resulting map, because the contour lines don't pass smoothly from tile to tile.

To reduce this effect it is necessary to extend the "tiled grid kriging" approach. The tiles are now allowed to overlap each other to some extent at their borders, let's say o_x points in x-direction and o_y points in y-direction. Now for the grid points where tiles overlap the prediction will be performed more then once. For these points the final prediction result will be calculated as arithmetic mean of all their results from different tiles. This operation has again to be done by the supervising process.

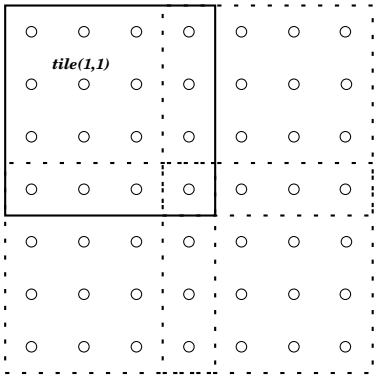

Figure 4: Tiled grid kriging with overlapping tiles

Of course this approach will loose some of the efficiency gains from the last section. Finally it will be the task of the user to find a trade-off between computational effort and output quality.

4 Implementation with PVM and R

The Implementation is based on both R and PVM. The functions for parallel kriging are implemented in a R library called `pvmkrige`[1]. Figure 5 gives a schematic

[1] available at `ftp://ftp-stat.uni-klu.ac.at/pub/R/contrib`

overview of the interactions between R and PVM. This idea has bee implemented with help of H. Tschofenig and N. Samonig (see Tschofenig [2001] and Samonig [2001]).

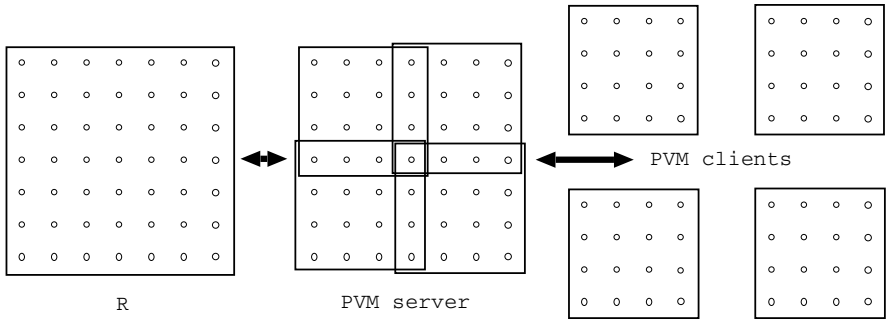

Figure 5: Schematic overview of PVM kriging with R

The advantage of using R and PVM consists in the convenient way to extend existing software with self written routines. PVM consists of some management programs and an API to its base library. It is ported to many computer platforms and so it makes it possible to use very heterogeneous hardware as a large virtual cluster over standard network connections. For more details see e.g. Geist and Others [1995].

The R interface function `krige.pvm.tiles` prepares the data set and passes it together with variogram, grid and tile parameters to a PVM master process called `krige_server`. This PVM program establishes the PVM cluster and starts the PVM client programs `krige_client` on each cluster node. Then it sends the data set and variogram parameters once to each client. After this is done, the PVM master starts dividing the grid into tiles according to the parameters for tile sizes and tile overlapping given from R. Then it establishes a work queue of tiles and subsequently feeds the clients with tile parameters and waits for their results. This is repeated until the work queue is empty. It has also to detect if some clients crash or timeout and has to re-send those tiles to another client. After it got all results back it calculates the arithmetic mean at points where tiles overlap and returns the predicted grid to R.

The library `pvmkrige` is based on the R library `sgeostat` and uses the same object types and print and plot methods. The main part of the `krige_client` programs is the Fortran call to `DGESV` for solving the kriging matrix equations. It is planned to re-implement parts of the `krige_server` program using the R library `rpvm`.

5 Example

The example uses the same data set as the examples for the functions in the R library `sgeostat` and `gstat`. It represents zinc data from a soil measurement campaign in a flood plain of the river Meuse (Netherlands). For comparison with sequential kriging another R library was used. This is the `rgeostat`[2] library, which extends the `sgeostat` library with a Fortran based kriging function `krige.grid`. Additionally two R functions `krige.tiles` and `krige.tilesov` implementing sequential versions of tiled grid kriging exist. The first comparison covers only non PVM kriging showing the impact of switching over to tiles, see Figure 6.

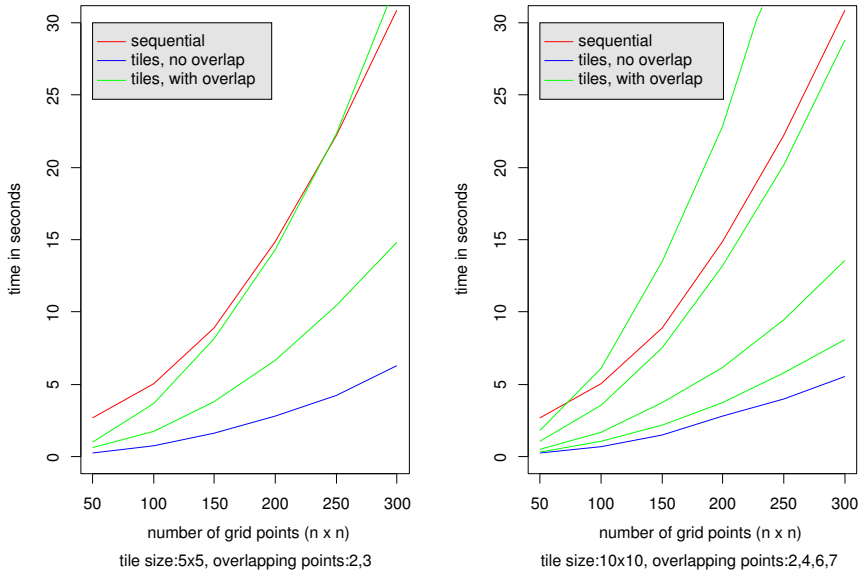

Figure 6: Comparison of different tile setups

Figure 7 shows the impact of different cluster setups. It can also be seen that greater overlap parameters slow down the prediction. Unfortunately these results are not comparable with Figure 6. The reason is the heterogenity of the PVM cluster in use. While the results shown in Figure 6 where produced on a recent Alpha workstation (UP 2000) the PVM cluster consists mainly of very old and slow machines (also Alpha) connected partially only over 10Mbit ethernet. Figure 8 shows a screenshot of XPVM displaying the PVM trace log of one kriging calculation. What can be seen are the single tile computing steps and the communication between the PVM master process and its clients. Finally figure 9 shows the kriging maps produced by PVM kriging with and without overlapping tile parameters.

[2]also available at `ftp://ftp-stat.uni-klu.ac.at/pub/R/contrib`

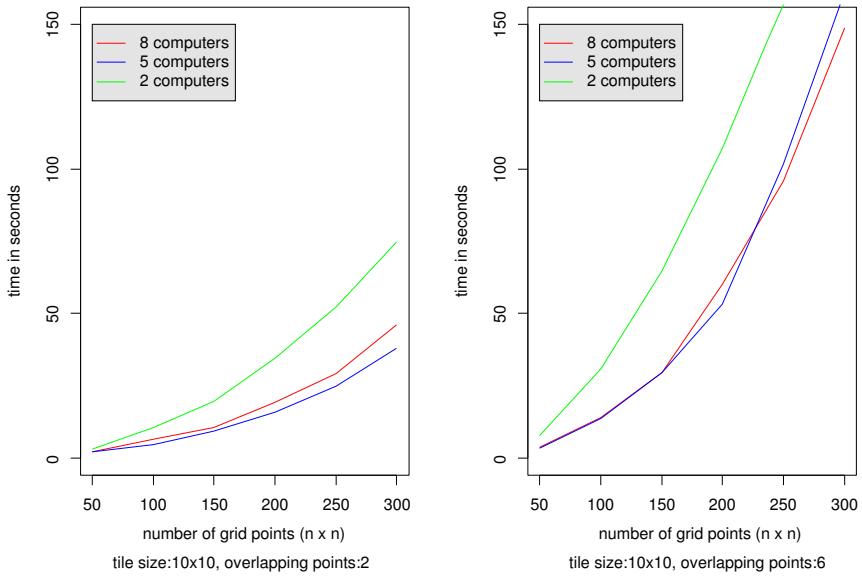

Figure 7: Comparison of different PVM cluster setups

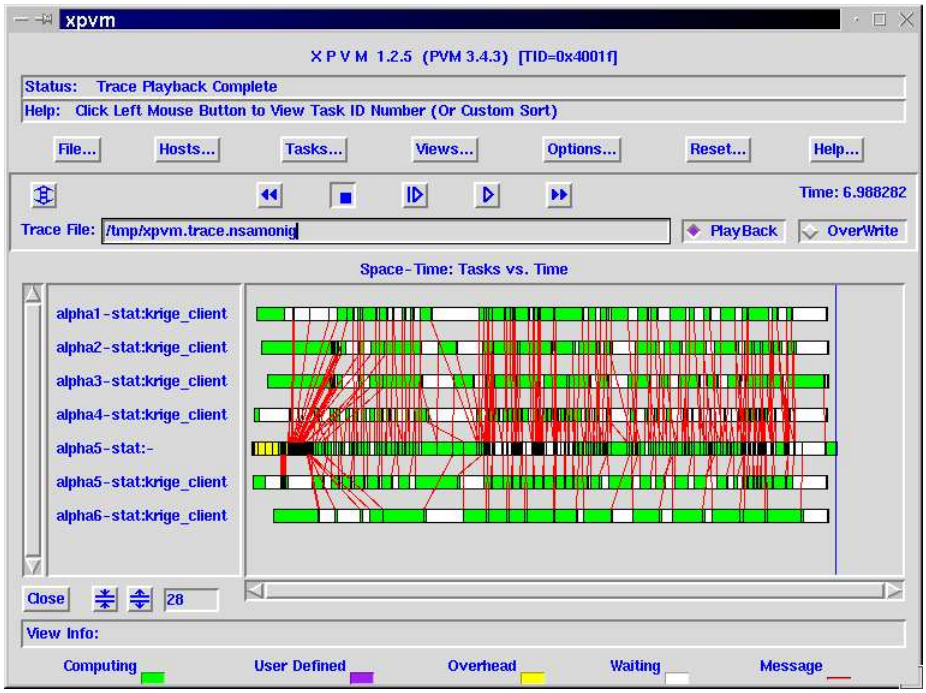

Figure 8: XPVM screenshot

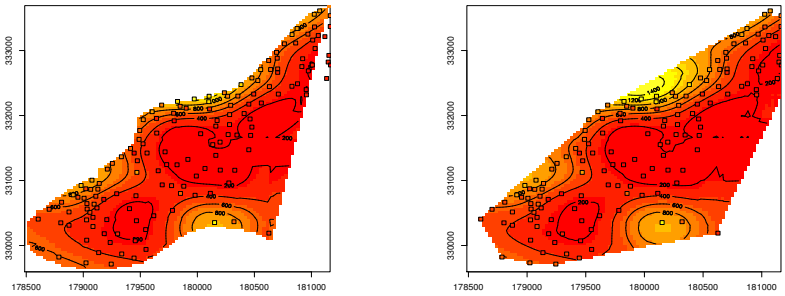

Figure 9: PVM tile kriging output, tile size 5×5, overlapping tile in right picture

References

E. Anderson and Others. *LAPACK Users' Guide*. SIAM Philadelphia, third edition, 1999.

N. Cressie. *Statistics for Spatial Data*. Wiley, New York, 1991.

Al Geist and Others. *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Networked Parallel Computing*. The MIT Press, Cambridge, Massachusetts, second edition, 1995.

Nadja Samonig. Parallel computing in spatial statistics. Master's thesis, University Klagenfurt, 2001.

Hannes Tschofenig. Anwendung der parallelen, virtuellen Maschine (PVM) für eine Geostatistikanwendung. Technical report, Universität Klagenfurt, 2001.