STUDIA UNIV. "BABEŞ-BOLYAI", INFORMATICA, Volume XLIII, Number 2, 1998

DATA DISTRIBUTIONS FOR PARALLEL PROGRAMS

VIRGINIA NICULESCU

ABSTRACT. The data distributions have a serious impact on the time complexity of parallel programs. This paper presents *simple* distributions of arrays and matrices and introduces *set* distributions of same kinds of data. The *set* distributions are used when the number of processors is greater than the data input size. The work-load properties of these distributions and their impact on the number of communications are discussed. In order to illustrate the possibilities to chose the best distribution for a program, some examples are presented.

1. INTRODUCTION

Many models of parallel programs use distributed data objects, like arrays or matrices. Each part of a data object is assigned to a process. In a parallel program, processes can independently perform operations on their data until a global information is needed. The communication processes are responsible for combining and collecting global information via message passing. This causes synchronisation points in the parallel program, which usually lead to waiting times. The impact of distributions on the time complexity of a parallel program is therefore an important issue.

A model for parallel programs design can consider a parallel program as a number of cooperating parameterised processes with similar structures. These processes are imperative programs. A parallel programs is specified using parameterised functional specifications, for each process, with parameterized preconditions and postconditions (given in a Hoare form):

$$\{Q.q\}S.q\{R.q\},\ 0 \leq q < p.$$

Such a specification forms the starting point for a parallel program derivation; a formal construction of parameterised processes constituting a parallel program [1].

Here are discussed *static* distributions, and the possibilities to chose the best distribution for a parallel program, before developing it. In a static distribution the assignments does not change during the execution of the program. There are presented two kind of distributions, classified on the ratio between the data size n, and the processes number p:

- If $n \geq p$ there are *simple* distributions defined by single-valued functions

- If $n < p$ there are *set* distributions defined by set-valued mappings.

1.1. **Notations.** A notation for quantifications is used:

$$(\odot k : Q : E)$$

where $\odot$ is a quantifier $(\sum, \forall, \max, \ldots)$, k is the list of bound variables, Q is the predicate describing the domain of the bound variables, and E is an expression. Function application is denoted by an infix, left associative dot '.' operator. The lambda calcul is used in the definition of the distribution functions. The set $(\forall i : 0 \leq i < n : i)$ is denoted by $\overline{n}$. The integer division and remainder use the symbols / and \, respectively.

2. Simple Distributions

The simple distributions are characterized by:

- the number of elements assigned to one process,
- the data distribution on processes.

If the data input is a vector, the simple distribution is called one-dimensional distribution, and if it is a multi-dimensional array, the distribution is called Cartesian distribution [1].

2.1. One-dimensional Distributions.

defintion 1. *$D = (\delta, A, B)$ is called a distribution, if A and B are finite sets, and δ is a mapping from A to B. Set A specifies the set of data objects (an array with n elements), and the set B specifies the set of processes, which is usually $\overline{p}$. The function δ assign each index $i : 0 \leq i < n$ (and its corresponding element) to a process number.*

Well-known ways of distributing an array are: every element to one unique process (*identity*), assigning p equally-sized consecutive array segments (*linear*) and assigning elements cyclically (*wrap*):

$$\begin{array}{lcl} identity & = & ((\lambda\ i \cdot i), \overline{n}, \overline{p}) \\ linear & = & ((\lambda\ i \cdot i/(n/p), \overline{n}, \overline{p}), \text{ provided that } p \mid n\ . \\ wrap & = & ((\lambda\ i \cdot (i \backslash p), \overline{n}, \overline{p}). \end{array}$$

defintion 2. *Given is a distribution $(\delta, \overline{n}, \overline{p})$, the set of elements from $\overline{n}$ assigned to process q, $0 \leq q < p$, is given by $O.q$:*

$$O.q = (\forall i : i \in \overline{n} \wedge \delta.i = q : i).$$

Counting the cardinalities of the sets $O.q$ is meaningfull in time complexity analysis. That gives a good indication of the amount of work per process. The sets $O.q$ form a partition of $\overline{n}$.

defintion 3. *The maximum/minimum number of data objects assigned to a process for a distribution $(\delta, \overline{n}, \overline{p})$ are defined by:*

$$\begin{aligned} Ma(\delta) &= (max\ q : 0 \leq q < p : |O^{\delta}.q|) \\ Mi(\delta) &= (min\ q : 0 \leq q < p : |O^{\delta}.q|). \end{aligned}$$

defintion 4. *A distribution $(\delta, \overline{n}, \overline{p})$ is called w-balanced, $w \geq 0$ iff:*

$$Ma(\delta) - Mi(\delta) \leq w$$

A distribution is called homogeneous if $w = 1$, and is called perfect if $w = 0$.

The cardinalities of the $O.q$s are the same for the *linear* and *wrap* distributions. Both distributions assign a maximal number of elements to process 0:

$$Ma(linear) = Ma(wrap) = (n + p - 1)/p$$

A measure of the load imbalance is the difference between the $Ma(\delta)$ and $Mi(\delta)$. For certain class of computations the *wrap* distribution is a good candidate. The following example prove this.

example 1. *Given is a parallel program consisting of n steps. In step $k, 0 \leq k < n$ computations are done only for the first k elements of the program's arrays, (all arrays have length n and are distributed in the same way). Each array requires a constant number of elementary operations. The maximal number of elementary computations in step k, for any array distribution $(\delta, \overline{n}, \overline{p})$ is bounded from below by:*

$$(max\ q : 0 \leq q < p : |O^{\delta}.q \cap \overline{k}|) \geq (k + p - 1)/p).$$

This follows from:

$$\begin{aligned} & p * (max\ q : 0 \leq q < p : |O^{\delta}.q \cap \overline{k}|) \\ \geq & \quad \{calculus\} \\ & (\sum q : 0 \leq q < p : |O^{\delta}.q \cap \overline{k}|) \\ = & \quad \{O.q\ forms\ a\ partition\ of\ k\} \\ & |(\cup q : 0 \leq q < p : O^{\delta}.q \cap \overline{k})| \\ = & \quad \{calculus\} \\ & k. \end{aligned}$$

The lower bound is attained by the wrap *distribution.*

$$\begin{aligned} & |O^{wrap}.q|) \\ = & \quad \{definition\} \\ & |(\forall i : 0 \leq i < n \wedge i \backslash p = q : i)| \\ = & \quad \{calculus,\ range\ splitting\} \\ & |(\forall i : 0 \leq i < (n/p) * p \wedge i \backslash p = q : i)| + |(\forall i : (n/p) * p \leq i < n \wedge i \backslash p = q : i)| \\ = & \quad \{calculus\} \\ & n/p + |(\forall i : 0 \leq i < n \backslash p \wedge i \backslash p = q : i)| \\ = & \quad \{calculus\} \\ & (n + p - 1 - q)/p. \end{aligned}$$

The maximum value is obtained for $q = 0$. *Replacing* $n = k$ *gives the result.*

So, at any step in such a parallel program, the maximum number of computations in a process for any distribution is at least the maximum number of computations in a process using the wrap *distribution. Therefore, in this case,* wrap *distribution is one of the best.*

2.2. **Composition of Distributions.** Distributions can be composed to obtain new distributions.

defintion 5. *The composition of two distributions* $D0 = (\delta 0, \overline{m}, \overline{M})$, $D1 = (\delta 1, \overline{n}, \overline{N})$, *with* $M = n$ *is defined by:*

$$D1 \circ D0 = (\delta 1 \circ \delta 0, \overline{m}, \overline{N}).$$

The properties of a composed distribution can be obtained from the properties of its constituents.

A distribution called *wrap-of-linear* can be obtain , from $linear = (\delta^{linear}, \overline{n}, \overline{m}), m \mid n$, and $wrap = (\delta^{wrap}, \overline{m}, \overline{p})$ as folows:

$$wrap - of - linear = ((\lambda i \cdot (i/(n/m))\backslash p), \overline{n}, \overline{p}).$$

The number of elements assigned to a process $q, |O^{wrap-of-linear}.q|$, can be obtained by counting:

$$|(\forall i, j : \ 0 \leq i < m \wedge \delta^{wrap}.i = q \wedge 0 \leq j < n \wedge \delta^{linear}.j = i : \ j)|.$$

For homogeneous distributions $D0 = (\delta 0, \overline{m}, \overline{M})$, $D1 = (\delta 1, \overline{n}, \overline{N})$, with $M = n$:

$$Mi(\delta 1) * Mi(\delta 0) \leq |O^{\delta 1 \circ \delta 0}.q| \leq Ma(\delta 1) * Ma(\delta 0).$$

Composition of distributions does not always preserve homogeneity.

Practical applications of composed distributions are, for example, parallel programs using different data distributions. By introducing a parameter like m in *wrap-of-linear*, it is possible to trade off the load balance for each individual part and to avoid expensive redistributions during computation.

2.3. **Cartesian Distributions.** Distributions of multi-dimensional arrays can be modeled by Cartesian distributions. In the following, it is assumed that a m by n matrix is distributed across processes.

defintion 6. *The Cartesian product of two one-dimensional distributions* $D0 = (\delta 0, \overline{m}, \overline{M})$, $D1 = (\delta 1, \overline{n}, \overline{N})$, *is defined by:*

$$D0 \times D1 = (\delta 0 \times \delta 1, \overline{m} \times \overline{n}, \overline{M} \times \overline{N}).$$

where the function $\delta 0 \times \delta 1$ *assigns to array index pair a pair of process numbers.*

Formally:

$$\delta 0 \times \delta 1 = (\lambda i, j \cdot (\delta 0.i, \delta 1.j)).$$

The Cartesian product of two one-dimensional distributions uses a process pair as identification for a process. Cartesian distributions of matrices can be obtained

by distributing the rows of the matrix independently from the columns. Since p, the processes number is fixed, we can consider all decomposition such that $p = M * N$.

The set of elements assigned to each process by a Cartesian distribution $D0 \times D1$ of a m by n matrix can be defined in a similar way as in Definition 2. It has the following property:

$$O^{\delta 0 \times \delta 1}.(s,t) = O^{\delta 0}.s \times O^{\delta 1}.t,$$

with $0 \leq s < M$ and $0 \leq t < N$.

Consider two homogeneus distribution $D0$ and $D1$. Similar results hold for the Cartesian distribution $D0 \times D1$ as for composition, for instance:

$$Mi(\delta 0) * Mi(\delta 1) \leq |O^{\delta 0 \times \delta 1}.(s,t)| \leq Ma(\delta 0) * Ma(\delta 1)$$

Some examples of Cartesian distributions can be given, for $p = M \times N$ processes:

$linear^2$	=	$(\delta^{linear}, \overline{m}, \overline{M}) \times (\delta^{linear}, \overline{n}, \overline{N})$, with $M = N$; called also *block*.
row	=	$linear^2$ with $N = 1$.
$column$	=	$linear^2$ with $M = 1$.
$wrap^2$	=	$(\delta^{wrap}, \overline{m}, \overline{M}) \times (\delta^{wrap}, \overline{n}, \overline{N})$, with $M = N$; called also *grid*.
$wrap - row$	=	$wrap^2$ with $N = 1$.
$wrap - row$	=	$wrap^2$ with $M = 1$.

2.4. **Counting Communications.** During a communication, processes need values that are not available locally, values that have been assigned to some different processes, and hence have to be communicated. The distribution determines the total number of communications. It is also important that the communications are spread evenly across the processes in such way that many communications take place in parallel - if the communication network offers enough freedom to implement communication processes efficiently.

Given a program's postcondition and a distribution it is possible to count the *total* number of communications. The program's postcondition is split in p local postconditions according to the distribution used. With every local postcondition a process is associated that will establish it. If is it assumed that every datum is assigned to one unique process then the total number of postconditions that refer to a particular datum is a measure of the number of communications of that datum. However, it may happen that a subexpression containing several data occurs in different postconditions. One process can compute such a subexpression and store the result in a variable, which is communicated to the other processes. In this way, communication is reduced. This case is excluded from the following counting technique.

For the datum e it is introduced the quantity $Nocc.e$:

$Nocc.e =$ the number of local postconditions in which e occurs.

By summing over all e, it will be obtained the total number of communications $Ncom$:

$$Ncom = (\sum e :: Nocc.e - R.e)$$

where:

$$R.e = \begin{cases} 1, & \text{if } e \text{ occurs in the postcondition of the process that contained it} \\ 0, & \text{otherwise.} \end{cases}$$

The value of $Ncom$ is only determined by the way the program's postcondition is split up and the distribution used. The communication complexity is bounded from below by $(Ncom + p - 1)/p$ if a process can perform only one communication action at each moment.

This technique of counting communications allows for a comparison of the distributions on the basis of their communication overhead. Even if, the applicability of the technique is not possible when exist common subexpressions, it is useful especially because the results that can be obtained are independent of any communication network. To illustrate this technique, an example is given.

example 2. *Given are two matrices a and b, of dimensions $m \times o$, and $o \times n$, respectively. The problem is to compute the m by n matrix c, satisfying postcondition R:*

$$R: \ c = a \cdot b.$$

For matrix c is used a Cartesian distribution $D0 \times D1$, and are used $p = M * N$ processes; each process is identified by an ordered pair $(s,t), 0 \leq s < M, 0 \leq t < N$.

The local postconditon $R.s.t$ of process (s,t) is:

$$\begin{aligned} R.s.t \ : \ & (\forall i,j : 0 \leq i < m \wedge 0 \leq j < n \wedge \delta 0.i = s \wedge \delta 1.j = t \\ & : c(i,j) = (\sum k : 0 \leq k < o : a(i,k) * b(k,j)). \end{aligned}$$

Note that:

$$(\forall s,t : \ 0 \leq s < M \wedge 0 \leq t < N : \ R.s.t) \Rightarrow R.$$

In order to count the number of communications, quantities $Nocc.a(i,k)$ and $Nocc.b(k,j)$ are calculated:

$$\begin{aligned} & Nocc.a(i,k) \\ = & \\ & |(\forall s,t : 0 \leq s < M \wedge 0 \leq t < N \wedge \delta 0.i = s \wedge (\exists j :: \delta 1.j = t) : (s,t))|. \end{aligned}$$

If the $\delta 1$ is surjective:

$$\begin{aligned} & Nocc.a(i,k) \\ = & \quad \{\textit{defintion Nocc, } \delta 1 \textit{ is surjective } \} \\ & |(\forall s,t : 0 \leq s < M \wedge 0 \leq t < N \wedge \delta 0.i = s \wedge true : (s,t))| \\ = & \quad \{\textit{calculus}\} \\ & N * |(\forall s,t : 0 \leq s < M \ \wedge \delta 0.i = s : s)| \\ = & \quad \{\delta 0 \textit{ is a function}\} \\ & N. \end{aligned}$$

Similary: $Nocc.b(i,k) = M$, *if* $\delta 1$ *is surjective. Hence:*

$$\begin{aligned} & Ncom \\ = & \quad \{definition\ Ncom\ \} \\ & (\sum i,k : 0 \le i < m \wedge 0 \le k < o : Nocc.a(i,k) - 1) + \\ & (\sum k,j : 0 \le k < o \wedge 0 \le j < n : Nocc.b(k,j) - 1) \\ = & \quad \{calculus\} \\ & o * (m * (N-1) + n * (M-1)). \end{aligned}$$

Some observations can be made:

- *For* $M = N = p = 1$, $Ncom = 0$ *and no communications are necessary.*
- $Ncom$ *is independent of particular choices of* $\delta 0$ *and* $\delta 1$.
- *It is possible to determine* M *and* N, $p = M * N$ *such that* $Ncom$ *is minimal. All possible values* (M,N) *are integer points on the hyperbola* $p = M * N, 1 \le M, N \le p$, *and the values of* $Ncom$ *for fixed* m, n, o *lie on the line with a slope dependent on* $\frac{m}{n}$. *Hence, the minimal value for* $Ncom$ *depends on the ratio* $\frac{m}{n}$ *and in particular for* $m = n$, $Ncom$ *has a minimal value if* p *is square. This result confirm the results obtained by evaluation of parallel programs based on one-dimensional decomposition and two-dimensional decomposition* [2].

3. Set Distributions

When the number n of data input objects is smaller than the processes number, an element of data input can be assigned to more than one process.

defintion 7. *A set distribution for* n *data input objects on* $p(> n)$ *processes is defined by a set-valued mapping* $\delta : \overline{n} \dashv \overline{p}$; $\delta.i$ *represents the set of processes containing data object* i.

The set distributions are characterized by:

- the number of processes containing one data object,
- the data distribution on processes.

Examples of set distributions are $\overline{linear}$ and $\overline{wrap}$:

$$\begin{aligned} \delta^{\overline{linear}}.i &= (\forall k : 0 \le k < p/n : i(p/n) + k), \text{ if } n \mid p \\ \delta^{\overline{wrap}}.i &= (\forall k : 0 \le k < p/n : kn + i), \text{ if } n \mid p. \end{aligned}$$

Similar properties with those for the simple distributions can be established. But, because is not simple to work with a set of processes and, the chosen method to design parallel programs start from a global postcondition, which is split in parameterised postconditions, another approach is necessary. If data input is a n length vector, p is factorised in this way: $p = M * Q$; where $M \le n$. A simple distribution $(\delta, \overline{n}, \overline{M})$ is replicated Q times.

If data input is a m by n matrix, p the processes number is factorized in $p = M * N * Q$, where $M \le m$ and $N \le n$. A Cartesian distribution $D0 \times D1$ will be

replicated. In fact, a renumbering of processes is made, in order to simplify the specification of the set of processes that contain same data.

The parallel programs is split into two stages. In the first stage all processes work to compute partial results, and, the second, in which partial results are combined. For a matrix input, the global postcondition is split in $M*N$ parameterised postconditions, in which some partial calculus occur. To calculate these partial values $M*N*Q$ partial parameterised postconditions are defined.

The counting communications technique used for the simple distributions can be used also for the set distributions, with the difference that for the datum e the value $R.e$ is the number of processes where it is assigned.

example 3. *We consider again, two matrices a and b of dimensions $m \times o$, and $o \times n$, respectively (with $p \leq m*n*o$). To calculate the m by n product matrix c, postcondition R must be satisfying:*

$$R: \ c = a \cdot b.$$

For matrix c is used a Cartesian distribution $D0 = (\delta 0, \overline{m}, \overline{M}) \times D1 = (\delta 1, \overline{n}, \overline{N})$, and are introduced $p = M*N*Q$ processes; each process is identified by an ordered pair $(s,t,r), 0 \leq s < M, 0 \leq t < N, 0 \leq r < Q$. Another distribution $D2 = (\delta 2, \overline{o}, \overline{Q})$ is used.

The parameterised postconditon $R.s.t.0$ of process $(s,t,0)$ is:

$$\begin{aligned} R.s.t.0 \ : \ & (\forall i,j : 0 \leq i < m \wedge 0 \leq j < n \wedge \delta 0.i = s \wedge \delta 1.j = t \\ & : c(i,j) = (\sum r : 0 \leq r < Q : c(i,j).r)). \end{aligned}$$

Values $c(i,j).r$ are calculated based on the following parameterised postconditons:

$$\begin{aligned} R^0.s.t.r \ : \ & (\forall i,j : 0 \leq i < m \wedge 0 \leq j < n \wedge \delta 0.i = s \wedge \delta 1.j = t \\ & : c(i,j).r = (\sum k : 0 \leq k < o \wedge \delta 2.k = r : \ a(i,k) * b(k,j))). \end{aligned}$$

Note that:

$$(\forall s,t : 0 \leq s < M \wedge 0 \leq t < N : R.s.t.0) \Rightarrow R.$$

For the first stage, the number of communications is determined by:

$$\begin{aligned} & Nocc.a(i,k) \\ = \ & |(\forall s,t,r : 0 \leq s < M \wedge 0 \leq t < N \wedge 0 \leq r < Q \wedge \delta 0.i = s \wedge \delta 2.k = r \wedge (\exists j :: \delta 1.j \\ & = t) : (s,t,r))| = 1, \end{aligned}$$

analogue $Nocc.b(k,j) = 1$. Therefore $Ncom = 0$.

For the second stage:

$$\begin{aligned} & Nocc.(c(i,j).r) \\ = & \\ & |(\forall s,t,: 0 \leq s < M \wedge 0 \leq t < N \wedge \delta 0.i = s \wedge \delta 1.j = t : s,t))| \\ = & \\ & 1, \end{aligned}$$

Hence $Ncom = mn(Q-1)$. *If* Q *is taken the small it is possible, the number of communications decreases, but the work-load of the processes increases* ($p = M * N * Q$).

It can be observed that $M \cdot N$ *communications can be performed in parallel. Therefore, the communication complexity depends on* $\frac{mn}{MN}(Q-1)$.

4. Conclusions

Static distribution of arrays and matrices are discussed. New complex distributions can be obtained by the composition and the Cartesian product. The Cartesian distributions have the property to consider rows and columns as entire units distributed across M ensembles of N processes, respectively. Additionally, M and N may vary under the constraint $p = M * N$. Quantities like the sizes of the set elements assigned to a process allow for a comparison of the load-balancing properties of distributions.

Counting communications enables evaluation of different distributions. The result obtained are independent of any communication network.

When the number of processes is greater than the number of input elements, another kind of distributions are constructed, based on set-valued mappings. The best way to arrange processes, can be chosen based on the distribution.

References

[1] Lol L.D. Loyens, A Design Method for Parallel Programs,Technische Universiteit Eindhoven,1992.

[2] I. Foster, Designing and Building Parallel Programs,1995.

[3] I. Chiorean, Calcul paralel, "Babeş- Bolyai" University, Cluj-Napoca, 1995 (in Romanian).

"Babeş-Bolyai" University of Cluj-Napoca, Department of Computer Science
E-mail address: gina@cs.ubbcluj.ro