Conflicts between 2 vector transfers in Cray-XMP computers

R. Butel

IRIA
CENTRE DE ROCQUENCOURT

Rapports de Recherche

N° 418

CONFLICTS BETWEEN 2 VECTOR TRANSFERS IN CRAY-XMP COMPUTERS

Rémi BUTEL

Juin 1985

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tel. (3) 954 90 20

Conflicts between 2 vector transfers in Cray-XMP computers

Rémi Butel

INRIA

RESUME :

Cette note examine un aspect du fonctionnement de la mémoire des Cray XMP, et complète l'analyse de ce système de mémoire effectuée par Cheung [1].

Cette étude correspond principalement au cas où deux accès simultanés à la mémoire sont effectués par un seul processeur pour des transferts de blocs vectoriels avec incréments différents. Ce genre de situation est d'une part assez fréquent dans les codes scientifiques, et d'autre part conduit à une dégradation de performance importante, qui peut aller jusqu'à diminuer la vitesse de ces transferts simultanés d'un facteur 2.

L'analyse a été faite à partir de simulations et de tests réels.

On ne s'est intéressé qu'au cas d'un seul processeur, sans prendre en compte les dégradations de performances résultant des accès simultanés à la mémoire par un ou plusieurs autres processeurs de l'ordinateur, et des transferts d'entrée-sortie.

ABSTRACT :

This paper examines an aspect of the Cray-XMP memory system behaviour, and completes the analysis of this memory system done by Cheung [1].

This study is mainly focused on the case where two simultaneous accesses are done between memory and a CPU for vector block tranfers with different strides. This kind of situation is in fact frequent in scientific programs, and leads to a noticeable degradation of performance of the transfer speed, which may be decreased in some very bad but rare cases by a factor two. The analysis was done by means of simulations and real runs.

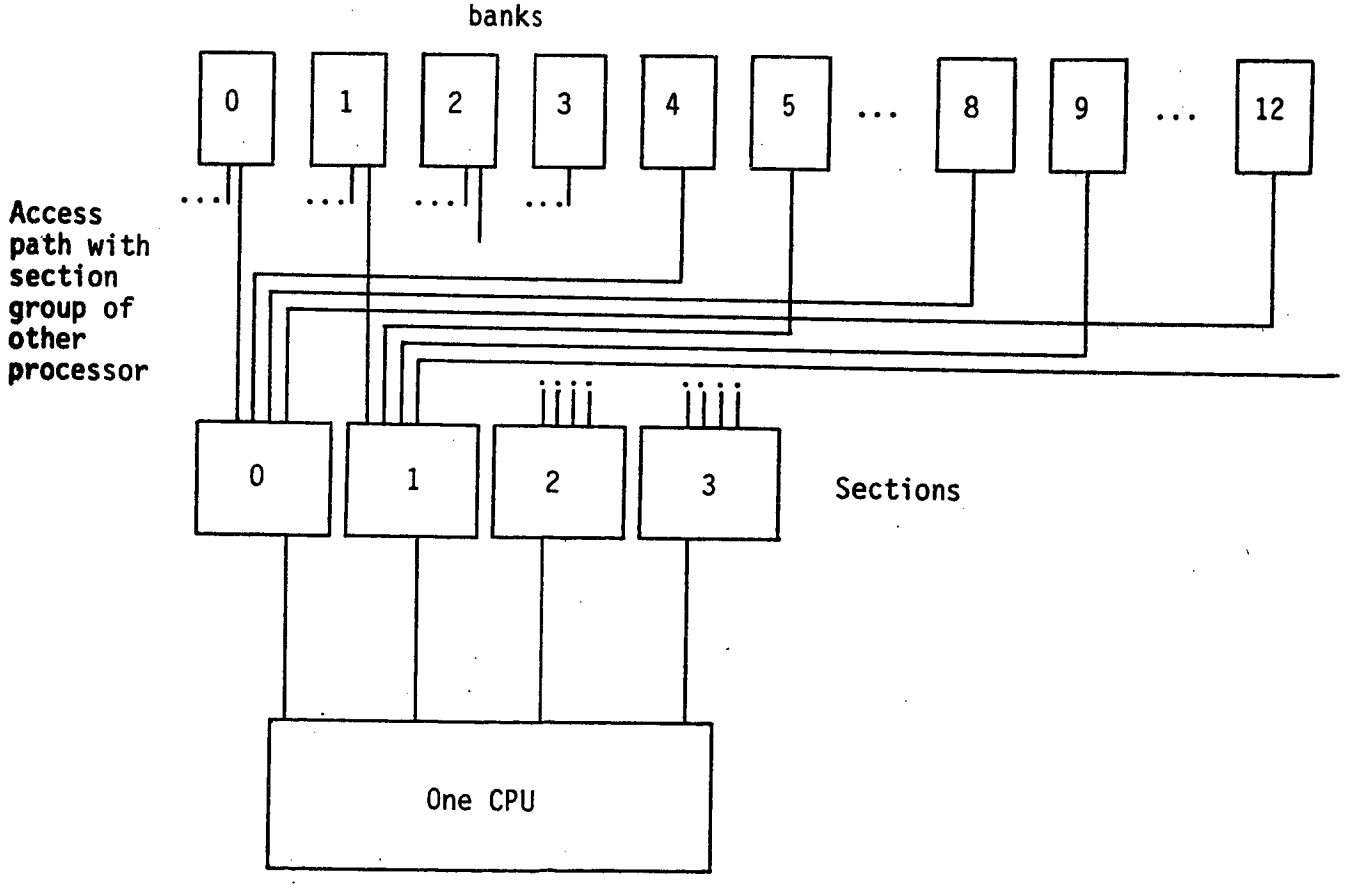
The case of a single CPU is the only studied ; the other degradations of performance resulting of the interaction between CPU's or of I/O transfers are not examined.

I Memory system working principles

I.1 Presentation

The Cray-XMP memory systems architecture and functionning are described in Cray publications [2],[3]. We will give here their principal features.

The memory is composed of chips whose cycle time is 4 cycles of CPU. In order to keep maximum performances and speed of transfer, the memory is divided in interleaved banks (their number is a power of two in order to make easy the adressing) ; these banks are grouped in sections of 8 banks.(1) These sections are independant for each CPU. Each CPU has four access ports to the memory, one of them is devoted for I/O transfers. It is possible to simultaneously carry out two vectorial load transfer and one vectorial store transfers (and then - in the best case - to carry out a diadic operation at the rate of on result per CPU cycle, operands loading and results storing included).



(1) 4 for the Cray XMP-22

Each vectorial block transfer instruction is allocated to one of the three "vectorial" ports (two ports for loading transfers, one for storing transfers). Each time a port is allocated, it becomes reserved until all elementary access request of the transfert are completed. These elementary requests (one word each) are individually and serially scanned, in order to detect memory access conflicts, and no elementary request of a transfer can be executed before the previous would be (for the same transfer). So the requests are executed serially inside a transfer, at the maximum rate of one by CPU cycle (we shall now say cycle instead of CPU cycle).

The simultaneity of access to memory between different instructions has four consequences :

- There is a possibility of conflicts between different instructions for gaining access to the same bank at the same time, or to the same section inside a CPU (as for a bank, at a given time there is only one access allowed to a section, but section reservation is only one cycle).
- The rate of one access per cycle can not be guaranteed.
- A priority mecanism has been introduced to solve the simultaneous access conflicts.
- When necessary, the serial execution of block transfers can (and must) be ensured by new hardware instructions.

I.2 Conflict resolution scheme

Conflicts can occur between elementary requests

- of a same instruction (case A)
- of different instructions executing on the same CPU (case B)
- or on different CPU (case C).

These conflicts are of three kinds :

- Simultaneous access to the same bank ("Simultaneous bank conflict" in [2]) , (in cases B, C).
- Access to a bank reserved for 4 cycles by a previous access ("Busy bank conflict") , (in cases A, B, C).
- Simultaneous access to the same section ("Section conflict") , (in case B).

Notes :

This last kind of conflict, which can only arises between differents requests of a same CPU, does not exist in Cray-1 computers.

The reservation duration of a section is one cycle, and so delay all other instructions requesting access to this section.

The priority of an elementary request is given by the following rules :

- Between differents processors :

  In Cray XMP-22 and XMP-24, the priority permutes every 4 cycles.

- Inside a processor :

  - odd stride transfers instructions have priority on those with even stride and scatter-gather transfers,
  - with same stride parity, the requests of the first instruction initiated have priority,
  - scatter-gather transfers always have the lowest priority (these transfers exist only for Cray XMP-48).

Notes :

When there are conflicts, results cannot be delivered at each cycle, but chaining yet occurs, in opposite with Cray-1S.

On a processor, as only one instruction can be initiated at each cycle, and as each vectorial transfer instruction must be preceded by the initialisation of the address register A0, a minimum of two cycles separates each vectorial transfer instruction initiation.

When only two instructions conflict, there never will be priority permutation, either because strides have different parities, or because there is an initiation order.

II Principles of the memory system simulator

II.1 Working principles

A simulation of one or two vector transfer instructions can be done, possibly with chaining, in reading or writing mode for the second. These instructions will be noted A and B.

The two instruction simulations are "initiated" independantly, and delay between these two initiations can be specified.

For each stride pair tested, the simulation is done for all initial configurations, i.e. for all initial bank gap (this gap is of course taken modulo the total number of banks). Therefore we obtain minimum, maximum and mean times of execution for each stride pair.

The non unit stride is to be taken only between 1 and NB the total number of bank of the computer(1) , all others strides give identical results. So we have only NB * NB * 2 simulations to run.(2)

When the second instruction is a writing one, we suppose that it is chained with the first instruction (always a reading transfer), either directly (copy of a vector for example), or via an operation ; behaviour of the instructions is not the same in this two cases. A new kind of instruction holding can happen in chained mode , when the operand is not ready. The simulator then prints the index of read and written vector register components (the index of the operand register can never be greater than the index of the last loaded register component).

The varying delay is used to take in account the effect of the execution of a vector operation between the load and the store (for example : B(i) = A(i) + Cte).

This mode of simulation will now be designated as the LS mode (Load-Store), whereas the other will be denoted LL mode (Load-Load).

The LL mode corresponds, for example, to the case where two operands are loaded simultaneously for an operation,

---

(1) 16 for XMP22, 32 for XMP24, 64 for XMP48

(2) On a Cray 1-S, for 64 banks, only 14 seconds of CPU, without printing the reservation tables !)

with no immediate storing of results (that do not match the case of a diadic operation, but applies to little more complicated operations). The LL mode also corresponds to the storing of a result of a precedent operation, simultaneous with the loading of an operand of a new operation.

The rules described in the paragraph I.2 are applied at each simulation cycle in order to determine the bank and section reservation, and the state changes of instructions A and B.

The choice of the delay in the LS mode is usually 17 (vector copy), 17+8 (floating add with scalar), or 17+9 (floating multiply with scalar). In the LL mode, the range of choices is greater, but the duration that separates two vector transfer instructions is at least two cycles, so we have focused our interest on this case, with a maximum overlapping between the transfers. In fact, due to the partial chaining capacity of the Cray-XMP, it is important to minimize the delay between the loading of the first operand and the start of the operation, start given by the availibility of the second operand.(1)

Moreover, unlike the cases examined by Cheung [1] - which were restricted to equal stride simultaneous transfers - the two instructions are systematicaly initiated at different times. The repartition of kind of conflicts arising is not the same as the one with simultaneous initiation. The delay of two cycles may in fact appear arbitrary, but others delays give slightly different results. In LS mode, with the chaining condition, the problem disappears.

It seems also that the most significative measure of performance is not the mean elementary request acceptance ratio, but the mean total time of completion of the two instructions (supposing that no other transfer operation occurs during their execution ; this is not a great restriction since the simulator is able to accept vector length multiple of 64). In LL mode, the total time directly gives the duration of operations chained on transfers.

The stride pairs tested were all of the form <1,i> or <i,1> (using Cheung's notation), we shall show that it is sufficient to cover all the cases in paragraph III.6. They were tested principaly in LL mode, because LR mode gives rise to very complicated and more difficult to explain conflicts.

---

(1) We can not assume that the compiler is able to minimize the delay between the two loadings, so this two cycles delay may be very rare in practice).

II.2 Outputs

The direct result of a simulation is a memory reservation table (Kogge , R1 [4]), giving at each CPU cycle and for each bank and section of the memory system a mark. This mark identifies the instruction an elementary request of which has reserved the bank or section.

A short explanation is added in case of request holding (giving the kind of conflict).

More analytic results presented for each stride pair are the following :

- mean duration, assuming an uniform repartition in the interval [0..NB-1] of the initial bank gap between the two instructions.
- minimum and maximum duration.
- mean duration of the first completed instruction.
- period of the memory reservation table pattern :
  All the initial bank gap cases develop in a periodic phase (called steady state phase by Cheung), and, inside this period, the same kind of situation can appear, with a circular shift modulo the total number of banks. The duration between this quasi-identical situations is a sub-period named P. In fonction of the initial bank gap the period may vary. The occurences of this phenomenon will be pointed out.
- the numbers Ra and Rb of elementary requests accepted during a sub-period, for instructions A and B.
- mean infered duration :
  The mean duration T can be approximated from the sub-period P and from the numbers Ra and Rb in the following way :

  We suppose here that the instruction A ends first.
  With VL the vector length (in 1..64) , we have

  $$T(\text{end A}) = VL * (P / Ra)$$
  $$T = T(\text{end A}) + \text{number of remaining B requests}$$
  $$= T(\text{end A}) + VL - VL * (P / Ra) * (Rb / P)$$
  $$= VL * (1 + (P - Rb) / Ra)$$

  More generaly, the formula is :

  $$T = VL * (1 + (P - \min(Ra,Rb)) / \max(Ra,Rb))$$

Notes :

1) Acceptance ratio during the periodic phase is :

$$(Ra + Rb) / (2 * P)$$

2) The formulas of infered mean time do not apply if conflicts occur between elementary requests of a same instruction (ie the supposed transfer rate for the lonely instruction is 1)

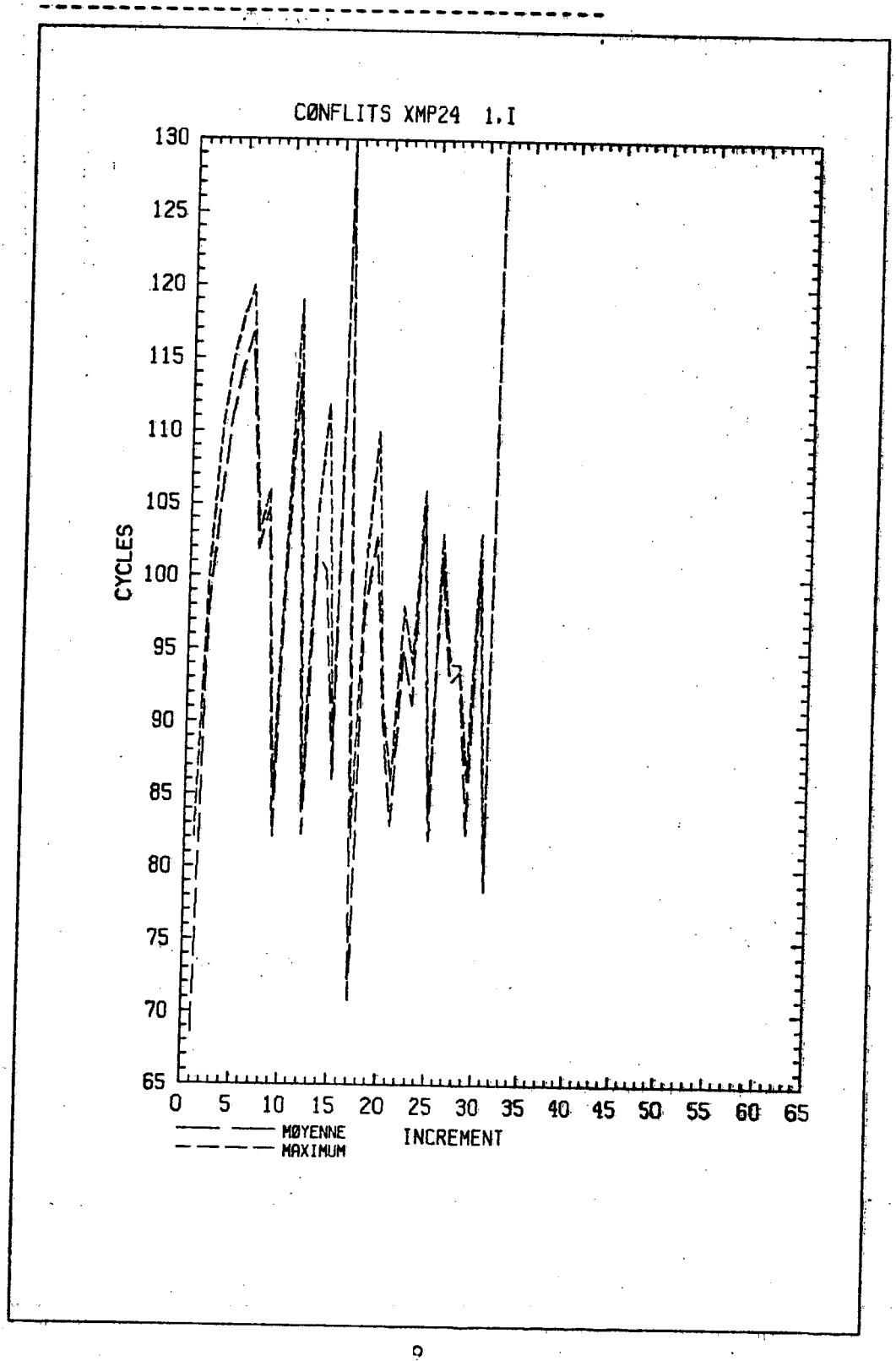
3) The validity of the formula is conserved in the pure transfer LS mode for any VL, since the use of the same vector register for A and B in the chained load-store operations implies that a double reservation on this vector, which ends at the end of B plus 5 cycles, giving then a synchronisation after each 64 elements transfer. This is not the case when an operation is chained between the two transfers, there is no more double reservation.

III Results

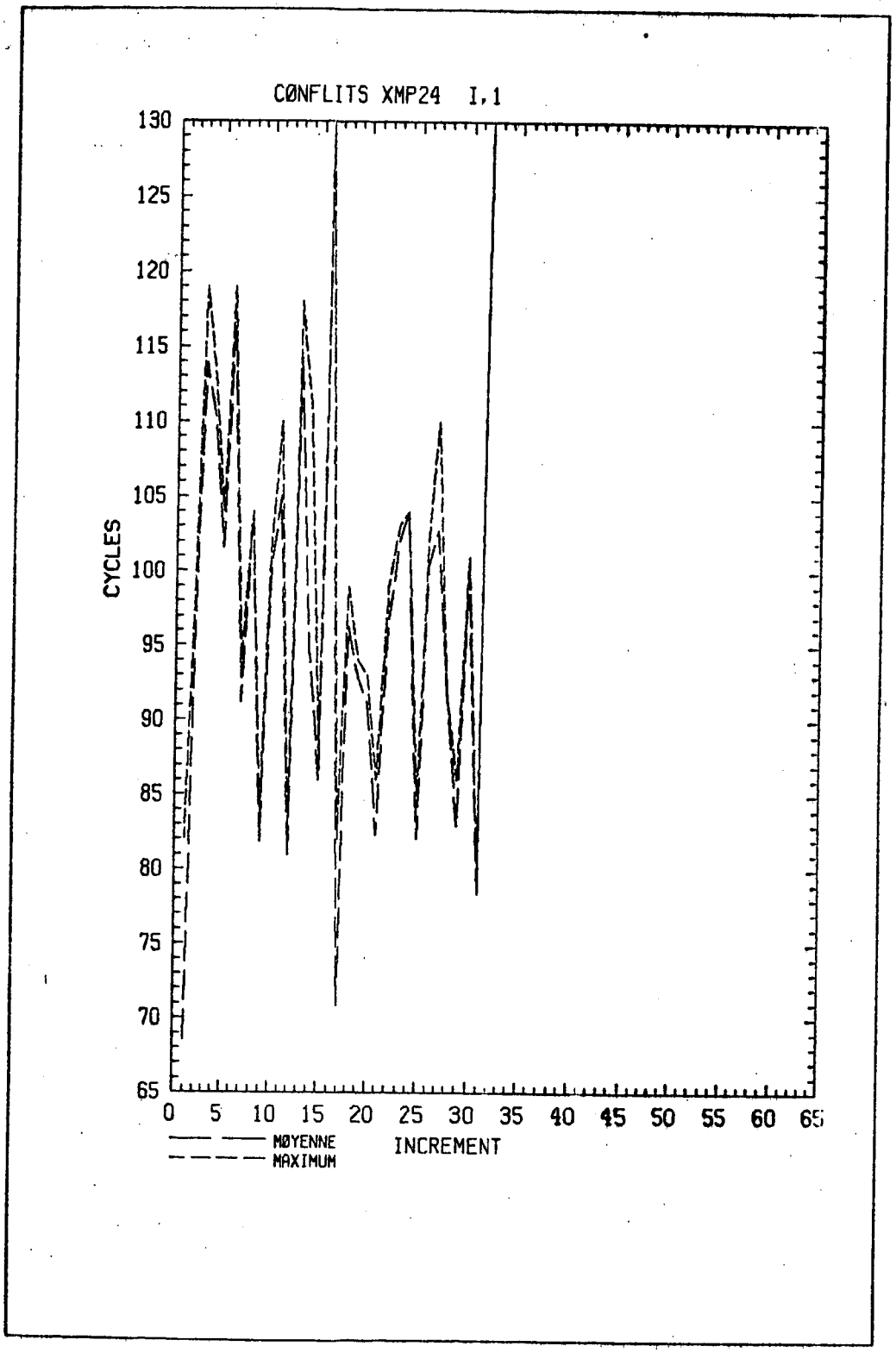
The notation <i,j> indicates that the first initiated load has stride "i" and the second stride "j".

III.1 Mean and maximum duration graphs in LL mode

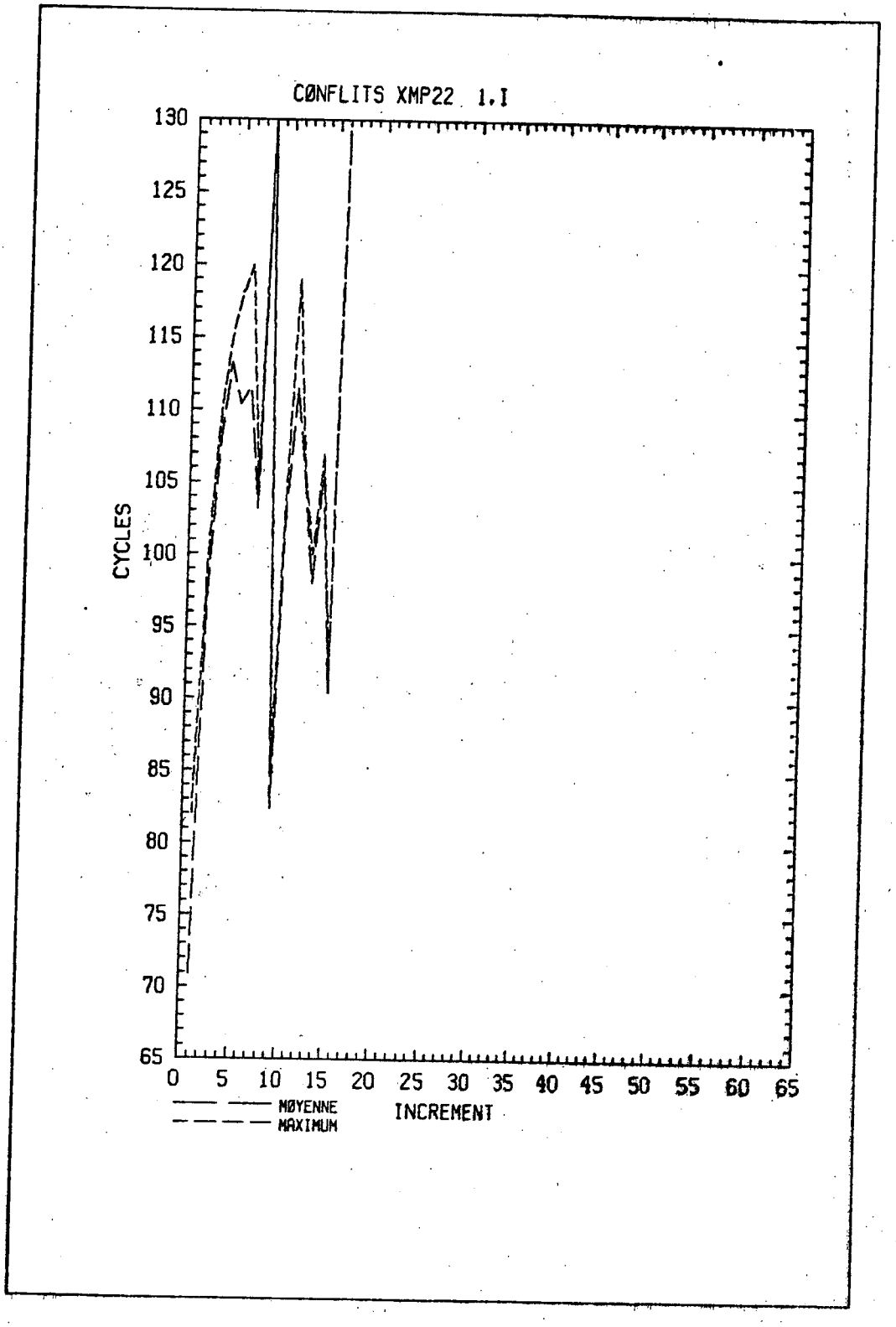
III.1.a Cray-XMP24 : <1,i> conflicts



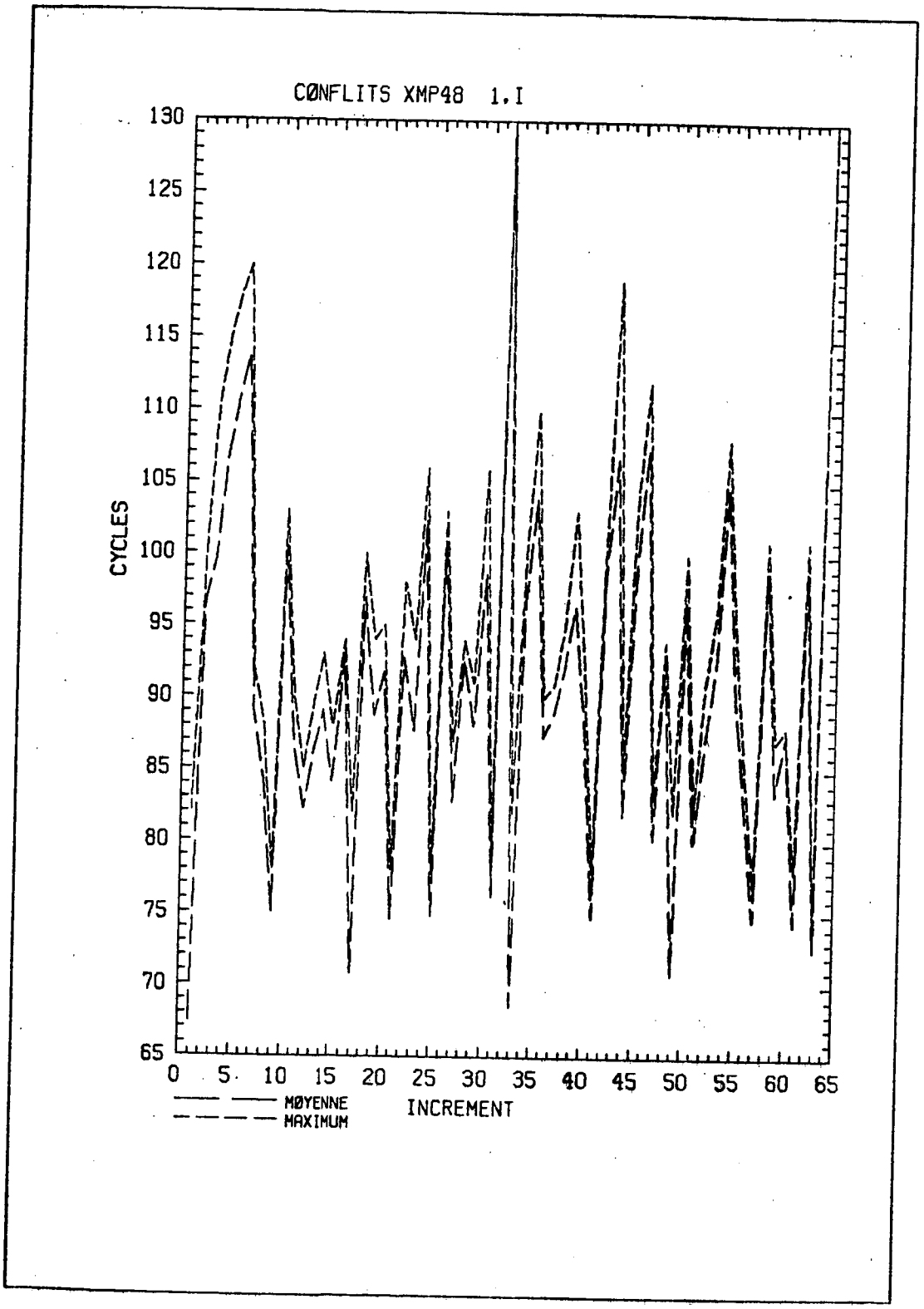
III.1.b Cray-XMP24 : <i,1> conflicts



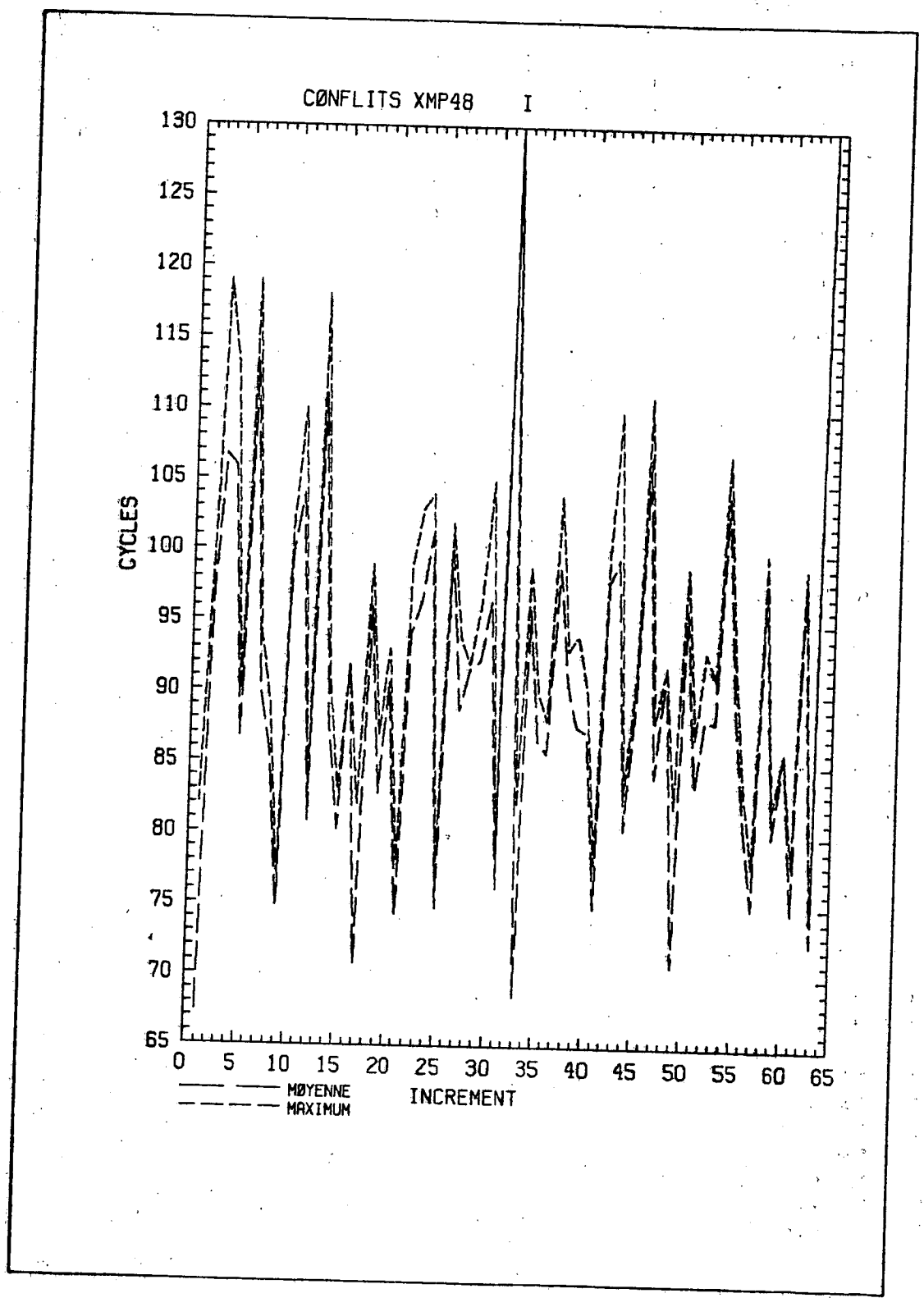
III.1.c Cray-XMP22 : <1,i> conflicts



III.1.d Cray-XMP48 : <1,i> conflicts



III.1.e Cray-XMP48 : <i,1> conflicts



III.2 Cray-XMP24 Reservation Table Patterns

<1,2> : The most frequent stride pair observed in programs

<1,3> & <3,1> : illustration of the pattern modification with initiation permutation

<1,7> : the second instruction completed before the first initiated

<1,10> : an example of very complex pattern and conflict chaining

<1,14> : pattern modification with the initial bank gap

<1,17> : a pair quasi equivalent to <1,1>

<1,31> : a both bank conflict case !

A sub-period is pointed out by "|" on the right side.

<1,2> ; initial banks 0,0 ; 99 cycles

```
  sect.  banks
                 0                   1                   2                   3
  0123   0123456789012345678901234567890123456789012345678901
1 A      - A
2  A     - AA
3   A    - AAA
4    A   - AAAA                                              -- bank conflict : B
5 A      -  AAAA                                             -- bank conflict : B
6 BA     - B AAAA                                            -- line conflict : B
7   A    - B  AAAA
8   BA   - B B AAAA                                          -- line conflict : B
9 A      - B B  AAAA
10 BA    -   B B AAAA                                        -- line conflict : B
11  A    -   B B  AAAA
12  BA   -     B B AAAA                                      -- line conflict : B
13 A     -     B B  AAAA
14 BA    -       B B AAAA                                    -- line conflict : B
15  A    -       B B  AAAA
16  BA   -         B B AAAA                                  -- line conflict : B
17 A     -         B B  AAAA
18 BA    -           B B AAAA                                -- line conflict : B
19  A    -           B B  AAAA                               |
                                                             -- line conflict : B |
```

```
<1,3>  : initial banks : 0,0 ; ... cycles

  sect.  banks

         0         1         2         3
  0123   01234567890123456789012345678901

 1 A     - A
 2  A    - AA
 3   A   - AAA                                  -- bank conflict : 3
 4    A  - AAAA                                 -- bank conflict : 3
 5 A     -  AAAA                                -- line conflict : 9
 6 BA    - B AAAA
 7   A   - B  AAAA                              -- bank conflict : 3
 8    A  - B   AAAA                             -- line conflict : 3
 9 A  B  - B  B AAAA
10  A    -    B  AAAA                           -- bank conflict : 3
11   A   -    B   AAAA                          -- line conflict : 3
12   BA  -    B  B AAAA                                                |
13 A     -       B  AAAA                        -- bank conflict : 3  |
14  A    -       B   AAAA                       -- line conflict : 3  |

<3,1>  : initial banks : 0,0 ; 112 cycles

  sect.  banks

         0         1         2         3
  0123   01234567890123456789012345678901

 1 B     - A
 2    A  - A  A
 3   A   - A  A  A                              -- bank conflict : 3
 4  A    - A  A  A  A                           -- bank conflict : 3
 5 A     -    A  A  A  A                        -- line conflict : 9
 6 B  A  - B     A  A  A  A
 7  BA   - BB      A  A  A  A
 8  AB   - BBB       A  A  A  A
 9 A  B  - BBBB        A  A  A  A
10 B  A  -  BBB           A  A  A  A
11  BA   -   BBB             A  A  A  A
12  AB   - A BBB                 A  A  A
13    B  - A  BBB                    A  A  -- bank conflict : A
14 A    - A  ABBB                        A  -- line conflict : 3
15 B    - A  A BBB                         -- bank conflict : A
16  B   -    A  BBB                        -- bank conflict : A
17   BA -    A  ABBB
18    B -       ABBBB                      -- bank conflict : A
19 B    -       A BBBB                     -- bank conflict : A
20  B   -       A  BBBB                    -- bank conflict : A
21   A  -          ABBB                    -- line conflict : 9
22   B  -          A BBB                   -- bank conflict : A
23    B -          A  BBB                  -- bank conflict : A
24 BA   -          A  ABBB                                      |
25  B   -             ABBBB                -- bank conflict : A  |
26   B  -             A BBBB               -- bank conflict : A  |
27    B -             A  BBBB              -- bank conflict : A  |
28 A    -                ABBB              -- line conflict : 9  |
29 B    -                A BBB             -- bank conflict : A  |
30  B   -                A  BBB            -- bank conflict : A  |
```

```
<1,7>  : initial banks : 0,0 ; 193 cycles

  sect.  banks

         0         1         2         3
  0123   01234567890123456789012345678901

 1 A     - A
 2  A    - AA
 3   A   - AAA                                          -- bank conflict : B
 4    A  - AAAA                                         -- bank conflict : B
 5 A     -  AAAA                                        -- line conflict : B
 6 BA    - B AAAA
 7   AB  - B  AAAAB
 8   B   - B   AAAB       B                            -- bank conflict : A
 9  B    - B    AAB       B       B                    -- bank conflict : A
10 B     -       AB       B       B       B            -- bank conflict : A
11    A  -        A       B       B       B            -- line conflict : B
12 A  B  -    B   AA              B       B
13  AB   -    B   AAAB            B       B
14  B    -    B   AAAB       B                         -- bank conflict : A
15 B     -    B    AAB       B       B                 -- bank conflict : A
16    B  -         AB       B       B       B         -- bank conflict : A
17   A   -          A       B       B       B         -- line conflict : B
18   BA  -        B   AA              B       B
19 AB    -        B   AAAB                    B
20 B     -        B   AAAB       B                    -- bank conflict : A
21    B  -        B    AAB       B       B            -- bank conflict : A
22   B   -   B         AB       B       B            -- bank conflict : A
23  A    -   B          A       B       B            -- line conflict : B
24  BA   -   B      B   AA              B
25 B  A  -   B      B   AAAB
26    B  -        B   AAAB       B                    -- bank conflict : A
27   B   -        B    AAB       B       B            -- bank conflict : A
28  B    -      B         AB       B       B         -- bank conflict : A
29 A     -      B          A       B       B         -- line conflict : B
30 BA    -      B      B   AA              B
31   AB  -      B      B   AAAB
32   B   -             B   AAAB       B               -- bank conflict : A
33  B    -   B         B    AAB       B               -- bank conflict : A
34 B     -   B     B         AB       B               -- bank conflict : A
35    A  -   B     B          A       B               -- line conflict : B
36 A  B  -   B     B      B   AA
37  AB   -        B      B   AAAB
38  B    -               B   AAAB       B            -- bank conflict : A
39 B     -     B         B    AAB       B            -- bank conflict : A
40    B  -     B      B         AB       B            -- bank conflict : A
41   A   -     B      B          A       B            -- line conflict : B
42   BA  -     B      B      B   AA                                        |
43 AB    -               B      B   AAAB                                  |
44 B     - B                    B   AAAB               -- bank conflict : A |
45    B  - B     B              B    AAB               -- bank conflict : A |
46   B   - B     B      B             AB               -- bank conflict : A |
47  A    - B     B      B              A               -- line conflict : B |
```

<1,10> ; initial banks 0,0 ; 100 cycles

```
  sect.  banks

         0         1         2         3
  0123   01234567890123456789012345678901

 1 A     - A
 2  A    - AA
 3   A   - AAA                                    -- bank conflict : B
 4    A  - AAAA                                   -- bank conflict : B
 5 A     -  AAAA                                  -- line conflict : B
 6 BA    - B AAAA
 7   A   - B  AAAA                                -- line conflict : B
 8   BA  - B   AAAA  B
 9 A     - B    AAAA B                            -- line conflict : B
10 BA    -         AAAAB          B
11   B   -          AAAB          B          B   -- bank conflict : A
12   A   -           AAA          B          B   -- bank conflict : B
13 B  A  -           BAAA         B          B
14 A B   -           B AAA      B            B
15 BA    -           B AAAA     B         B
16   A   -           B  AAAA    B         B      -- line conflict : B
17   BA  -         B     AAAA   B         B
18 A     -         B       AAAA           B      -- line conflict : B
19  A    -         B        AAAA                -- bank conflict : B
20   A   -         B         AAAA               -- bank conflict : B
21    A  -                    AAAA               -- bank conflict : B
22 A     -                     AAAA              -- line conflict : B
23 BA    -                    B AAAA
24   A   -                    B  AAAA            -- line conflict : B
25   BA  -                    B   AAAA  B
26 A     -                    B    AAAA B        -- line conflict : B
27 BA    -       B                   AAAAB
28   B   -       B          B          AAAB      -- bank conflict : A
29   A   -       B          B           AAA      -- bank conflict : B
30 B  A  -        B         B           BAAA
31 A B   -    B             B           B AAA
32 BA    -    B          B              B AAAA
33   A   -    B          B              B  AAAA  -- line conflict : B
34   BA  -    B          B           B     AAAA
35 A     - A             B           B      AAA -- line conflict : B
36  A    - AA                        B       AA -- bank conflict : B
37   A   - AAA                       B        A -- bank conflict : B
38    A  - AAAA                                  -- bank conflict : B
39 A     -  AAAA                                 -- line conflict : B
40 BA    - B AAAA
41   A   - B  AAAA                               -- line conflict : B
42   BA  - B   AAAA  B
43 A     - B    AAAA B                           -- line conflict : B
44 BA    -         AAAAB          B
45   B   -          AAAB          B          B   -- bank conflict : A
46   A   -           AAA          B          B   -- bank conflict : B
```

<1,14> ; initial banks 0,? ; maximum conflict : 112 cycles

```
   sect.  banks

                0         1         2         3
   0123    0123456789012345678901234567890123456789

 1 A     - A
 2  A    - AA
 3   A   - AAA                                          -- line conflict : 3
 4    A  - AAAA                                         -- bank conflict : 3
 5 A     -  AAAA                                        -- bank conflict : 3
 6  A    -   AAAA                                      -- bank conflict : 3
 7   A   -    AAAA                                     -- line conflict : 3
 8   BA  -   B AAAA
 9 A     -   B  AAAA                                   -- line conflict : 3
10 BA    -   B   AAAA      B
11   A   -   B    AAAA     B                           -- line conflict : 3
12   BA  -        AAAA     B            3
13 A     -         AAAA    B            3              -- line conflict : 3
14  A    -          AAAA                3              -- bank conflict : 3
15   A   -           AAAA               3              -- bank conflict : 3
16    A  -            AAAA                            -- bank conflict : 3
17 A     -             AAAA                           -- line conflict : 3
18 BA    -            B AAAA                                                |
19   A   -            B  AAAA                         -- line conflict : 3  |
20   BA  -            B   AAAA        3                                     |
21 A     -            B    AAAA       3               -- line conflict : 3  |
22 BA    -         B         AAAA     3                                     |
23   A   -         B          AAAA    3               -- line conflict : 3  |
24    A  -         B           AAAA                   -- bank conflict : 3  |
25 A     -         B            AAAA                  -- bank conflict : 3  |
26  A    -                       AAAA                 -- bank conflict : 3  |
27   A   -                        AAAA                -- line conflict : 3  |
```

<1,14> initial banks 0,12 ; minimum conflict ; 86 cycles

```
                0         1         2         3
   0123    0123456789012345678901234567890123456789

 1 A     - A
 2  A    - AA
 3 B A   - AAA          B
 4   BA  - AAAA         B                  3
 5 A     -  AAAA        B                  3           -- line conflict : 3
 6 BA    -   AAAA  B    B                  3
 7   A   -    AAAA B                       3           -- line conflict : 3
 8   BA  -     AAAAB                 B
 9 B     -     BAAAB                 B                 -- bank conflict : A
10 A B   -     B AAA           B     B
11 BA    - B   B  AAA           B     B
12   A   - B   B   AAA           B                     -- line conflict : 3  |
13   BA  - B       AAAA  B     B                                             |
14 A     - B        AAAA B                            -- line conflict : 3  |
15 BA    -            AAAAB                 3                                |
16   B   -            BAAAB                 3          -- bank conflict : A  |
17 B A   -            B AAA           3     3                                |
18   BA  -         B   B  AAA           B     3                               |
```

```
<1,31> ; initial banks : 0,0 ; 78 cycles

  sect.  banks

         0         1         2         3
  0123   01234567890123456789012345678901

 1 A     - A                                                    -- bank conflict : 8
 2  A    - AA                                                   -- bank conflict : 8
 3   A   - AAA                                                  -- line conflict : 8
 4    A  - AAAA
 5 A     -  AAAA                                                                    3
 6 BA    - B AAAA                                                                  33
 7   AB  - B  AAAA                                                                333
 8  BA   - B   AAAA                                                              3333
 9 AB    - B    AAAA                                                           33333
10 BA    -       AAAA                                                        333333
11   AB  -        AAAA                                                     3333333
12   BA  -         AAAA                                                  33333333
13 AB    -          AAAA                                               B3333333
14 BA    -           AAAA                                           BB333333
15   AB  -            AAAA                                       BBB33333
16   BA  -             AAAA                                   BBBB333333
17 AB    -              AAAA                               BBBB  33333333
18 BA    -               AAAA                          BBBB    AAAA33333
19   AB  -                AAAABBBB                  BBBB      AAAAA3333
20       -                 AAABBB                BBBB        AAAAA333
21       -                  AABB              BBBB          AAAAA3333
22       -                   AB            BBBB            AAAAA3333
23   BA  -                   BA          BBBB              AAAAA3333
24 AB    -                   BBAA      BBBB                AAAAA3333
25 BA    -                   BBBAAA  BBBB                  AAAAA3333
26   AB  -                   BBBBAAAABBBB                  AAAAA3333
27   BA  -                     BBBB  AAAA                  AAAAA3333
28 AB    -                   BBBB      AAAA                AAAAA3333
29 BA    -                  BBBB         AAAA              AAAAA3333
30   AB  -                 BBBB           AAAA            AAAAA3333
31   BA  -               BBBB             AAAA          AAAAA3333
32 AB    -             BBBB                 AAAA        AAAAA3333
33 BA    -           BBBB                     AAAA      AAAAA3333
34   AB  -         BBBB                       AAAA    AAAAA3333
35   BA  -       BBBB                         AAAA  AAAAA3333
36 AB    - A   BBBB                           AAAAAAAAA3333
37 BA    - AA BBBB                             AAAAAAA3333
38   AB  - AAABBBB                               AAAAA3333
39       - AAABBB                                  AAA3333                -- both bank conflict
40       -  AABB                                   AA3333                 -- both bank conflict
41       -   AB                                    A3333                 -- both bank conflict
  ...
```

III.3 (1,even) strides conflicts for Cray-XMP24 in LL mode

Stride	mean T	max. T	min. T	End 1	Per.	Ra	Rb	infer. T
1, 2	97.0	100	96	63.2	2	2	1	96
2, 1	96.1	99	95	65.3				
1, 4	111.0	115	107	63.2	4	4	1	112
4, 1	110.3	113	108	65.7				
1, 6	116.7	120	114	63.3	6	6	1	117.3
6, 1	116.3	119	115	65.7				
1, 8	104.7	106	102	77.6	10	8	5	104
8, 1	104.0	104	103	80.2				
1,10	100.9	103	98	66.9	17	16	8	100
10, 1	100.3	102	99	69.1				
1,12	82.2	85	81	63.8	4	4	3	80
12, 1	80.8	83	80	65.8				
1,14	100.4	112	86	67.1	10	10	3	108.8
					7	6	5	85.5
14, 1	94.7	111	85	71.7	7	5	6	85.5
					10	3	10	108.8
1,16	144.8	147	141	70.4	18	16	7	108(1)
16, 1	143.0	145	141	72.7				
1,18	97.1	100	95	63.4	2	2	1	96
18, 1	96.3	99	95	65.5				
1,20	89.9	91	89	70.2	5	4	5	89.6
					6	6	2	106.7
20, 1	91.1	93	89	76.2				
1,22	95.0	98	94	78.6	7	4	6	96
22, 1	96.4	99	94	75.4				
1,24	104.7	106	102	63.8	8	8	3	104
24, 1	104.0	104	103	66.5				
1,26	101.3	103	100	66.7	17	16	8	100
26, 1	100.4	102	99	68.8				
1,28	93.5	94	92	74	8	7	5	91.4
28, 1	92.0	92	92	76.1				
1,30	101.1	103	100	66.8	17	16	8	100
30, 1	100.0	101	99	69.3				
1,32	265	266	263	65.6	-	-	-	- (1)
32, 1	260	261	259	67.7				

III.4 (1,odd) conflicts for Cray-XMP24 in LL mode

Stride	mean T	max. T	min. T	End 1	Per.	Ra	Rb	infer. T
1, 1	68.5	82	66	65	1	1	1	64
1, 1	"	"	"		* 5	4	4	80
1, 3	105.0	110	101	63.2	3	3	1	106.7
3, 1	113.9	119	110	76.1	7	2	6	117.3
1, 5	114.5	118	111	63.3	5	5	1	115.2
5, 1	101.6	104	99	63.8	7	7	3	100.5
1, 7	101.8	103	100	79	6	3	5	102.4
7, 1	91.2	94	89	63.7	5	5	3	89.6
1, 9	82.0	85	80	77.3	6	5	5	76.8
9, 1	81.8	84	80	77.1	"	"	"	"
1,11	113.9	119	110	76.1	7	2	6	117.3
11, 1	105.0	110	101	63.2	3	3	1	106.7
1,13	101.6	104	99	63.8	7	7	3	100.5
13, 1	114.5	118	111	63.3	5	5	1	115.2
1,15	86.0	88	85	78.4	10	7	9	85.3
15, 1	86.0	88	85	78.4	7	5	6	85.3
1,17	70.8	82	66	66.9	1	1	1	64
17, 1	70.8	82	66	66.9	* 5	4	4	80
1,19	102.8	110	99	70.7	8	5	7	91.4
19, 1	92.8	94	91	71.3	* 6	6	2	106.7
1,21	82.9	86	79	77.1	20	16	16	76
21, 1	82.3	86	79	77.3	"	"	"	"
1,23	91.2	94	89	63.7	5	5	3	89.6
23, 1	101.8	103	100	79	6	3	5	102.4
1,25	8.18	84	80	77.1	5	4	4	80
25, 1	82.1	85	80	77.3	"	"	"	"
1,27	92.8	94	91	71.3	8	7	5	91.4
27, 1	102.8	110	99	70.7	* 6	6	2	106.7
1,29	82.3	86	79	77.3	20	16	16	80
29, 1	82.9	86	79	77.1	"	"	"	"
1,31	78.4	80	75	74.1	19	16	16	76
31, 1	78.4	80	75	74.1	19	16	16	76

The asterisk points out a change of the period ; the periods of the inversed stride pair can then be deduced from the conjugate pair results (see III.6 generalisation)

III.5 Comments

The performances are very variable.

The speed slackening goes from 0 % to 97 %, for LL mode as for LS mode.

In the Cray-XMP24 simulations we can observ the period modification for the cases (1,1) , (1,14) , (1,15) , (1,17) , (1,19) , (1,27).

As there is no priority modification between unit stride and even stride when initiation times are permuted, only transient phases are modified, and there are little differences between the two simulation <1,i> and <i,1> ; in fact, on or two cycles, excepted the <1,14> case already characterised by its two periods.

On the other hand, very strong differences exist for the conflicts <1,odd> and <odd,1>.

The simulations in LS mode showed similar differences with initiation order permutation, for even as for odd strides, in this case due to the chaining mechanism.

The correlation between infered mean time and the mean time both obtained by simulation is good. This result shows the minor importance of transient phases for this kind of conflicts in LL mode, fact also confirmed in general by a slight difference between minimum and maximum durations. Large differences are in fact revealing the period changes in function of the initial bank gap. The reciprocal is false, as shows the (1,15) pair, with two periods and the same infered mean time.

The <1,7> conflict illustrates the fact that same index elementary requests of two instructions are not satisfied in serial order on the Cray-XMP computers; here the second initiated instruction is completed before the end of the first one. To insure the serial order of these requests when necessary, new hardware instructions have been added :

CMR (Complete Memory References) is hold before issue until all vector blocks tranfers currently executed are completed,

and DBM (Disable Bidirectional Memory transfers) forbids simultaneous loads and stores.(1)

---

(1) These instructions are also very useful for synchronisation between processors.

III.6 Generalisation to the case of two any strides

If pairs of strides usually found in scientific codes are of type (1,i), it is however interesting having a look on the general case (i,j).

Preliminary result :

A power of two and an odd number are always relatively prime.

This result is at the basis of all the following discussion.

Cheung mentioned in his paper [1] that the case of two equal odd strides is equivalent to the <1,1> case, by renumbering of the banks.(1) The case of two equal even strides was also discussed ; there are two sub-cases :

- If the initial bank gap is odd, the two transfer instructions do not interfer, and are executed in parallel without any conflict (excepted the pathologic strides 16, 32, 48, ...)
- If the initial bank gap is even, conflicts occur of type (i/2,i/2) in a reduced memory system (half banks, half sections),

and more generaly, if 2 ** k (two at power k) divides the two strides, there are 2 ** k sub-cases, all of them of the first type except one.

These results can be generalised to the general case of two different strides.

If one of the stride is odd, then the case is isomorphic to a case with unit stride, again using renumbering ; from preliminary result, with odd "i" ,there always exists a number "k" such as i * k = j (modulo 32) since "i" is invertible, and so <i,j> and <1,k> are identical.

---

(1) From the preliminary result, the application k -> i * k from Z/32 Z in itself is a one to one mapping when "i" is odd, and is the inverse of the wanted renumbering. 32 is the total number of bank, here for Cray-XMP24, and the results are similar for 16 and 64...

As illustration, one can see the exact similitude of results for stride pairs <1,3> and <11,1> ; <1,5> and <13,1> ; <1,7> and <23,1> ... in table III.3

Cases where the two strides are even again give rise to decoupled or reduced systems. Tables of performance are given here :

We can notice that the results for <2*i,2*j> are in average better than those for <i,j>, due to the decoupling, and that the <4*i,4*j> stride pairs give quasi-uniform results, due to the fact that the reduced memory system involves only one section !

Strides	mean T	max. T	min. T	End 1
2, 2	68.5	82	66	65
2, 4	81.9	100	66	63.2
2, 6	87	110	66	63.3
2, 8	89.6	115	66	63.3
2,10	88.2	118	66	67.7
2,12	90.5	120	66	69.8
2,14	84.5	105	66	74.3
2,16	147.8	69	128	73.9
2,18	74.3	85	66	70.4
2,20	84.7	105	66	65.8
2,22	88.8	119	66	71.5
2,24	89.4	114	66	64.1
2,26	82.1	100	66	77.4
2,28	89.5	114	66	70.3
2,30	78.1	92	66	74.2
2,32	256.5	258	255	65.3

Strides	mean T	max. T	min. T	End T
4,4	81.7	131	55	63
4,8	81.8	131	55	63
4,12	81.8	131	55	63
4,16	143.8	193	128	63
4,20	81.8	131	55	63
4,24	81.8	131	55	63
4,28	81.8	131	55	63
4,32	255.8	258	255	63

III.7 Real tests

I am indebted to A. Lichnewsky in performing some tests on the Cray XMP-48 of Cray Research Inc., at Mendota, US. The results of some of these tests are described and discussed here.

Two loops with a pair of simultaneous transfers were tested :

a pure load-store transfer (copy) :

```
      DO 1000 i = 1, ivl
        A(i * ia) = B(i * ib)
1000  CONTINUE
```

and a diadic operation :

```
      DO 2000 i = 1, ivl
        A(i * ia) = B(i * ib) + Cte
2000  CONTINUE
```

The arrays A and B were declared so as to have a zero initial bank gap.

The results are the global duration of 10000 executions of each loop, in seconds, given for vector lengthes of 256, 512 and 1024 (ivl). From these measures the loop overhead can be eliminated, and we can compute the mean number of cycles of one execution of the loop body. In comparison are given the number of cycles obtained from simulation, for zero initial bank gap, and for minimum and maximum durations. As two consecutive iteration of a 64 element loop may interfer, a simulation with vector length multiple of 64 must be used.

Note

The simulations showed a interesting behaviour of transfers in the case where, in LL mode, the acceptance ratio would be greater for the second request. In LS mode, this acceptance ratio can not be greater, or more accuratly, it can not remain greater during a long time : a relative lack of operand induced by this predominance of instruction B against instruction A occurs, and B becomes blocked, not by memory conflicts, but by the chaining mechanism. So the reservation table shows patterns where by turns B is predominant (and A slowed), or B is blocked, and A increasing its pace.

Results table

The first stride is the loading one, the second is the storing one

Loop 1

strides	T 256	T 512	T 1024	Overh.	cyc	0-0, min, max
< 2, 1>	.05788	.10650	.2038	.0092	512	440, 396, 472
< 1, 2>	.05027	.09142	.1736	.00918	433	416, 412, 432
< 3, 1>	.04726	.08528	.1614	.0092	400	468, 396, 477
< 1, 3>	.05866	.10820	.2070	.0092	521	440, 380, 460
< 7, 1>	.04498	.08071	.1522	.00923	376	396, 384, 411
<10, 1>	.05220	.09517	.1810	.0092	452	404, 396, 456
<17, 1>	.04231	.07541	.1415	.00925	348	340, 324, 383
<31, 1>	.04343	.07767	.1461	.0092	360	352, 348, 369
< 1, 1>	.04192	.07460	.1401	.1401	344	324, 324, 380

Loop 2

strides	T 256	T 512	T 1024	Overh.	cyc	0-0, min, max
< 2, 1>	.05022	.08936	.1677	.01105	412	484, 406, 499
< 1, 2>	.04870	.08634	.1616	.01107	395	404, 401, 411
< 3, 1>	.04707	.08317	.1554	.01095	380	424, 382, 454
< 1, 3>	.04973	.08710	.1616	.0126	393	432, 430, 435
< 7, 1>	.04601	.08097	.1510	.01110	368	373, 368, 379
<10, 1>	.05201	.09268	.1741	.01129	428	438, 407, 449
<17, 1>	.04187	.07270	.1343	.011	324	332, 332, 380
<31, 1>	.04697	.08346	.1564	.0105	384	363, 344, 368
< 1, 1>	.04183	.07273	.1342	.011	325	332, 332, 380

Except for two measures, all real test results fit in the simulation minimum-maximum duration interval, and in general well correspond to the zero initial bank gap simulation (almost always for the second loop). Only the accord with the <1,3> stride pair is bad.

The differences can possibly result from the simplifications done in the simulator, or, on another hand, from the way the compiler generated the code.

Inspite of this restriction, we can say that :

1) The accord is good for the first loop, yet better for the second. The simulation gives good indications for predicting behaviour of pairs of transfers.

2) The decreasing of performance is characteristic of the vector register double reservation, as mentioned earlier : the second loop is faster than the first !

Some tests written in CAL will be done to test the LL mode.

Conclusion

In the LL mode :

All the "races" between two vectorial transfer instructions lead to a periodic phase, the period of which can vary with initial bank gap and initiation delay between the two instructions.

A more or less complex chaining of conflicts occurs during this period phase, then giving a performance decrease in almost all cases, and in average (on strides, if this figure have a meaning) of about 50 % slower than the ideal <1,1> case, and sometimes close from the worst case (serial execution of instructions, see <1,6>).

This degradation is a quasi-random function of stride pairs, depending in addition of the activation order of instructions.

In the LS mode :

Resulting from the additional blocking mechanisms of chaining and vector double reservation, the behaviour is far more complex than the LL mode. No true periodic phase appears, but a quasi periodic comportement may exist however.

In the two modes, the behaviour changes when vector length is increased, as end of one transfer of the previous 64 loop iteration interfers with the other transfer beginning in the current iteration.

At a hardware level, the increasing total bank numbers from Cray-XMP22 to Cray-XMP48 gives medium improvement, due to the increasing of the sub-period in the periodic phase. Simulations were done with eight sections for a Cray-XMP48, and showed much more significant improvements (Cheug already mentioned the disparition, in this case, of the linked <1,1> conflict).

At a software level, concurrency between unit and non unit stride transfers should be avoided. However, these situations may arise in problems as important as spectral methods (with real operators applied to complex fields), or odd-even linear system solvers.

In conclusion, minimizing loads and stores remains a rule on Cray-XMP, with the supplementary constraint to use as often as possible unit strides.

Acknowledgements

I am very grateful to Professor Lichnewsky for the encouragements he provided to me and for the confirmations given by his tests.

The real tests were executed on the Cray-XMP48 at the Cray Research Inc. center at Mendota, US.

The simulations were tested at I.N.R.I.A. and were executed on the Cray 1-S of the C.C.V.R.

Finally this paper was also composed and edited on the DPS8 of I.N.R.I.A.

Bibliography

[1] Cheung, an analysis off the Cray-XMP memory system, in Proceedings of the 1984 International conference on Parallel Processing.

[2] CRAY X-MP COMPUTERS SYSTEMS, Mainframe Reference Manual, HR-0032

[3] CRAY X-MP 48 Hardware Reference Manual

[4] Kogge , 1981, The Architecture of Pipelined Computers, Mc Graw-Hill, New York

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique