The Characteristic Parameters of the NWT Computer System in the Local Memory Access*

Shigeki HATAYAMA*1

ABSTRACT

The NWT computer system available at the NAL since February 1993 comprises two system administrators,*n* processing elements (where *n* was 140 at the beginning, and is 166 at present) and a crossbar network, and operates as a distributed-memory message-passing MIMD computer. Each processing element itself is a vector computer. This paper reports measurements of two pairs of the characteristic parameters, (τ_∞, $n_{1/2}$) and (τ_∞, $s_{1/2}$), of the NWT with SIMD computing on a single processing element and with MIMD computing in the local memory access, respectively. Furthermore the significance of the obtained results is interpreted, and several hardware parameters are estimated. The obtained results in this paper apply only to the NWT system software during the period April to June 1993 after which substantial improvements have been achieved with the NWT performance.

Key words: Vector processing, Parallel processing, Local memory access, SISD/SIMD/MIMD computing, Synchronization overhead, Maximum performance, Half-performance length, Half-performance grain size, NWT.

概　　要

1993 年 2 月に運用を開始した航空宇宙技術研究所の数値風洞(NWT)計算機システムは 2 台の制御計算機，*n* 台の要素計算機(*n* は運用開始時において 140 台，現在は 166 台)並びにクロスバーネットワークから構成されていて，分散メモリ型メッセージパッシング方式の MIMD 計算機として動作する。各要素計算機自体はベクトル計算機である。本稿ではこの NWT の運用開始初期の 1993 年 4 月から 6 月の間に測定した，NWT の基本的処理性能を表す特性パラメータ(τ_∞，$n_{1/2}$)であり，他はローカル・メモリ・アクセスにおける NWT の並列処理性能を表す特性パラメータ(τ_∞，$s_{1/2}$)である。ここで，τ_∞ は最大処理性能を，$n_{1/2}$ は $\tau_\infty/2$ を達成するに必要なベクトル長を，$s_{1/2}$ は $\tau_\infty/2$ を達成するに必要なグレイン・サイズを表す。更に，得られた結果の意味について考察するとともに，いくつかのハードウエア・パラメータについて推定する。なお，本稿で示した値はその後の NWT システム・ソフトウエアの改善努力によって，大幅に改善されたが，その成果は稿をあらためて報告する。

1. INTRODUCTION

A parametric description of computer performance has been used to characterize the dependence of vector (SIMD) computer performance on vector length, using the parameters (τ_∞, $n_{1/2}$) [1, 2, 4, 9, 10, 11]. Similarly the degradation of performance due to the synchronization overhead that is incurred when programming with MIMD computing has been described by the prameters (τ_∞, $s_{1/2}$), and measured on several parallel computers [5, 6, 7, 8, 9, 12].

In this paper we apply the same techniques to the

* **平成 10 年 4 月 8 日受付** (received 8 April 1998)
*1 Computational Science Division

performance study of the Numerical Wind Tunnel (NWT) computer system available at the National Aerospace Laboratory since February 1993. This computer system comprises two system administrators, n processing elements (where n was 140 at the beginning, and is 166 at present) and a crossbar network, and operates as a distributed-memory message-passing MIMD computer. The language to bescribe parallel processing is the NWT Fortran, which is based on the Fortran 77 enhanced with the compiler directives. Almost all of the compiler directives are comments in the Fortran 77 specification. Main memories of the NWT are physically distributed across the processing elements, but to ease programming, the logical model of the NWT assumed a hierarchical memory parallel computer system for programming offers the virtual global space (or global memory) shared by the selected processing elememts to users [13]. On the other hand, each local space (or local memory) is the memory specific to each processing element. A barrier synchronization is performed for the selected processing elements that are running in parallel. After all the selected processing elements running in parallel reach the barrier point, the selected processing elements proceed over the barrier point together.

One of the purposes of this paper is to measure the parameters (τ_∞, $n_{1/2}$) of each processing element which itself is a vector computer. The definiton of the (τ_∞, $n_{1/2}$) benchmark and measurements of the parameters are given in Section 2. The other purpose is to measure the parameters (τ_∞, $s_{1/2}$) of the NWT with MIMD computing in the local memory access. The definition of the (τ_∞, $s_{1/2}$) benchmark and measurements of the parameters are givn in Section 3. The significance of the obtained results is discussed in Section 4, and several hardware parameters are estimated. The obtained results in this paper apply only to the NWT system software during the period April to June 1993. The results on an improvements to the NWT system software will be reported in another paper. By the way the benchmark program was run with other users on the system, but we will not very fail to ensure the most consistent measurements for the study in case of the local memory access. The terminology in this paper follows one in [9].

2. THE (τ_∞, $n_{1/2}$) BENCHMARK

The test problem which we call the (τ_∞, $n_{1/2}$) benchmark is dyadic and triadic operations executed on a single processing element as follows:

dyads : $A(I) = B(I) * C(I)$,

triads : $A(I) = B(I) * C(I) + D(I)$.

These Fortran codes are replaced by vector instructions by the vectorising compiler. The dyadic case uses only a single vector pipeline with all vectors stored in main memory. The triadic operations involve the chaining of two vector instructions and the simultaneous use of both the floating-point multiply and add pipelines with all vectors stored in main memory.

The time, t, to perform an operation on a vector of length, n, is measured and fitted by least-squares to the straight line described as follows:

$$t = a_0 + a_1 n. \tag{1}$$

From this equation we can obtain the characteristic parameters with SISD and SIMD computings on a single processing element as follows:

$t_0 = a_0$ = set-up time with SISD computing ($m$sec)
= time for null job when $n = 0$,

$t_0 = a_0$ = set-up plus pipeline start-up time with SIMD computing (μsec),

$p_0 = t_0^{-1}$ = specific performance (Mflop/sec),

$\tau_\infty = a_1^{-1}$ = maximum or asymptotic performance (Mflop/sec),

$n_{1/2} = a_0/a_1$ = half-performance length (flop) = lost amount of arithmetic operations that could have been done at the maximum rate of τ_∞,

where the set-up time is the time that is required to compute the first and last addresses for memory access based on the ratio of vector-register length to each vector on dyadic or triadic operations plus other overheads.

The actual or average performance, t, can be computed from

$$t = \tau_\infty \, pipe \, (n/n_{1/2}), \tag{2}$$

where

$$pipe\ (x) = 1/(1+x^{-1}) = pipeline\ function. \quad (3)$$

The vector breakeven length, n_b, which is the vector length above which the vector processing takes less time to perform the operation on a vector than the scalar processing, can be obtained from

$$n_b = n_{1/2}/(R_\infty - 1), \quad (4)$$

where

$$R_\infty = \frac{maximum\ vector\ processing\ rate}{maximum\ \ scalar\ processing\ rate}. \quad (5)$$

Table 1 shows measurements of the characteristic parameters of the NWT with SISD and SIMD computings on a single processing element. Figure 1 shows the timing relations and actual processing rates as a function of vector length with SIMD computation on a single processing element.

3. THE (τ_∞, $s_{1/2}$) BENCHMARK

The test problem which we call the (τ_∞, $s_{1/2}$) benchmark is dyadic and triadic operations defined in Section 2. The work of a dyadic and triadic operation is divided as evenly as possible between the selected number of processing elements, *pe*, for which two pairs of the compiler directives, PARALLEL REGION/END PARALLEL and SPREAD DO/END SPREAD, are inserted in Fortran codes. The SPREAD DO and END SPREAD are also barrier synchronization points in a parallel program. In order that the required data is available in the local memory of each processing element, furthermore, an EQUIVALENCE which is the statement to be coincided the location of the storage of the corresponding data assigned in the global and local memory, respectively, is declared. By this function, we can avoid to suffer from communication overheads due to the data transfer between the global and local memory spaces [3].

We measure the elapsed time, t, for a single parallel section as a function of the amount of computational work, s, in the parallel section measured in floating-point operations and fit the results by least-squares to the expression

$$t = a_0 + a_1 s. \quad (6)$$

From this equation we can obtain the characteristic parameters with MIMD computation in the local memory access as follows:

- $t_0 = a_0$ = set-up plus pipeline start-up plus synchronization time (μsec)
  = time for null job when $s = 0$, which is dependent on both the hardware and software (i.e., operating system and compiler),
- $\pi_0 = t_0^{-1}$ = specific performance (Mflop/sec),
- $\tau_\infty = a_1^{-1}$ = maximum or asymptotic performance (Mflop/sec),
- $s_{1/2} = a_0/a_1$ = synchronization parameter or half-performance grain size (flop)
  = amount of maximum possible arithmetic operations lost during synchronization,

where the time that is required to specify the starting and ending values of I, of which each processing element takes charge of the vector processing, is included in synchronization overheads.

The actual or average performance, τ, is given by the pipeline function as follows:

$$\tau = \tau_\infty\ pipe\ (s/s_{1/2}). \quad (7)$$

The scheduling parameter or efficiency of scheduling, E_{pe}, is defined by

$$E_{pe} = \frac{t_1}{t_{pe}pe}, \quad (8)$$

where

- t_1 = time to perform all the work on one processing element,
- t_{pe} = time to perform the work when it is shared amongst the *pe* processing elements.

Perfect scheduling occurs when $E_{pe} = 1$.

The breakeven grain size, s_b, above which it is faster to suffer the synchronization overhead and split the job between the *pe* processing elements than to avoid synchronization altogether by using a single processing element, can be computed from

$$s_b = s_{1/2}/(pe - E_{pe}^{-1}). \quad (9)$$

Table 2 shows measurements of the characteristic parameters of the NWT with MIMD computings in

the local memory access. Figure 2 shows the timing relations and actual processing rates as a function of the amount of arithmetic operations when pe = 1 ~ 128.

4. DISCUSSIONS

In this section we interpret the significance of the results obtained in the previous sections, and estimate several hardware parameters.

4.1 Consideration with SIMD computation

(1) The peak hardware performance of the processing element

For some reason that is not understood, the slower rates for the long-vector than ones for the short-vector have been measured, obviously shown in Figure 1. Figure 3.1 shows the timing relations as a function of the short-vector length. These are fitted by least-squares to

$$dyads{:}t = 7.401766*10^{-1} + 2.372605*10^{-3}n, \quad (10)$$

$$triads{:}t = 8.066134*10^{-1} + 1.756215*10^{-3}n, \quad (11)$$

from which we obtain the best measurements of the maximum performance of the NWT with SIMD computing as follows:

$$dyads : \tau_\infty = 4.214777*10^2 \ (Mflop/sec), \quad (12)$$

$$triads : \tau_\infty = 5.694064*10^2 \ (Mflop/sec). \quad (13)$$

Figure 3.2 shows the time chart of the vector pipelines, where $t_g = 0$ under the ideal situation. Since the arithmetic has been repeated 10^3 times to evade the problem on accuracy of the wall clock used measurements, we can think that the results are the values obtained in a situation after the second operation of Figure 3.2. Hence we can express the relations between the peak hardware performance, $\tilde{\tau}_\infty$, and τ_∞ as follows:

$$dyads : \tau_\infty = \frac{t_{VR}}{2t_{VR}} \tilde{\tau}_\infty, \quad (14)$$

$$triads : \tau_\infty = \frac{t_{VR}}{3t_{VR}} \tilde{\tau}_\infty, \quad (15)$$

where

t_{VR} = time to execute the elements on the vector register, from which we obtain

$$dyads : \tilde{\tau}_\infty = 8.429554*10^2 \ (Mflop/sec), \quad (16)$$

$$triads : \tilde{\tau}_\infty = 1.708219*10^3 \ (Mflop/sec). \quad (17)$$

Therefore we can estimate the average peak hardware performance per one arithmetic pipeline of the processing element as follows:

$$\tilde{\tau}_\infty = 8.485325*10^2 \ (Mflop/sec\ per\ one\ arithmetic\ pipeline). \quad (18)$$

If T_p = thickness of the pipelines, the machine clock period, τ, can be computed from

$$\tau = T_p / \tilde{\tau}_\infty, \quad (19)$$

from which if $T_p = 8$, we can estimate the average machine clock period as follows:

$$\tau = 9.428041*10^{-3} \ (\mu sec). \quad (20)$$

(2) The set-up plus pipeline start-up time

From (10), (11) and Figure 3.2, we can obtain the following relations:

$$dyads : 7.401766*10^{-1} = (s_d + l_L + l_M - 1 + n_{VR}/T_p)\tau, \quad (21)$$

$$triads : 8.066134*10^{-1} = (s_t + l_L + l_M - 1 + n_{VR}/T_p)\tau, \quad (22)$$

where

$t_d = s_d\tau$ = time that is required to compute the first and last addresses for each vector on dyadic operations plus other overheads (μsec),

s_d = amount of machine clock periods lost during a dyadic set-up

$t_t = s_t\tau$ = time that is required to compute the first and last addresses for each vector on triadic operations plus other overheads (μsec),

s_t = amount of machine clock periods lost during a triadic set-up over-

head,

$t_L = l_L\tau$ = start-up time of the load pipeline (μsec),

l_L = number of stages in the load pipeline,

$t_M = l_M\tau$ = start-up time of the multiply pipeline (μsec),

l_M = number of stages in the multiply pipeline,

$t_{VR} = n_{VR}\tau/T_p$ = elapsed time on the vector register (μsec),

n_{VR} = number of the elements on the vector register.

If $n_{VR} = 128$, $T_p = 8$ and (20) is used, the following relations are estimated:

$$dyads : s_d + l_L + l_M \doteq 64, \tag{23}$$

$$triads : s_t + l_L + l_M \doteq 71. \tag{24}$$

(3) Remark 1

We now consider the reason why the slow rates for the long-vector are observed. A plausible one seems that there is nonzero gap, t_g, shown in Figure 3.2. The value of t_g can be estimated from the equations

$$dyads : \tau_\infty = \frac{t_{VR}}{t_g + 2t_{VR}}\,\tilde{\tau}_\infty, \tag{25}$$

$$triads : \tau_\infty = \frac{t_{VR}}{t_g + 3t_{VR}}\,\tilde{\tau}_\infty, \tag{26}$$

where

$t_g = s_g\tau$ = elapsed time due to a gap (μsec),

s_g = amount of machine clock periods lost due to a gap.

Using (18), (20) and the values of τ_∞ in Table 1 (b), if $n_{VR} = 2048$, we obtain

$$dyads : s_g \doteq 46, \tag{27}$$

$$triads : s_g \doteq 20. \tag{28}$$

(4) Remark 2

Supposing two load pipelines could be available at the same time, the time chart of the vector pipelines turns into Figure 3.3, and hence we are to obtain the following values of about twice as large as maximum performance of (12) and (13):

$dyads$ = 842 ($Mflop/sec$),

$triads$ = 1138 ($Mflop/sec$).

Under this supposed condition, therefore, the maximum performance for the dyadic operations is equal very nearly to the estimated peak hardware performance of (16), and one of the triadic operations two third of (17).

4.2 Consideration with MIMD computation in the local memory access

(1) The complete formulas for the timing relations as a function of *pe*

Figure 3.4 shows the variation of maximum performance τ_∞ in Table 2 as a function of the number of processing elements *pe*. Solid lines show the linear fits

$$dyads : \tau_\infty = 3.858962*10^2\ pe\ (Mflop/sec), \tag{29}$$

$$triads : \tau_\infty = 5.468374*10^2\ pe\ (Mflop/sec). \tag{30}$$

Similarly Figure 3.5 shows the variation of half-performance grain size $s_{1/2}$ in Table 2 as a function of *pe*. Solid lines show the linear fits

$$dyads : s_{1/2} = -2.497182*10^3 + 1.325439*10^4\ pe\ (flop), \tag{31}$$

$$triads : s_{1/2} = -3.451546*10^3 + 1.878137*10^4\ pe\ (flop). \tag{32}$$

When pe =1, the value of $s_{1/2}$ is about 35 times as large as one of $n_{1/2}$ in Table 1 (b).

Figure 3.6 shows the variation of set-up plus pipeline start-up plus synchronization time t_0 (= a_0 in Table 2) as a function of *pe*. Since $t_0 = \tau_\infty^{-1}\ s_{1/2}$ from (6), these approximate fit functions are obtained from (29), (30), (31) and (32) as follows:

$$dyads : t_0 = 3.434703*10^1 - 6.471124*10^0/pe\ (\mu sec), \tag{33}$$

$$triads : t_0 = 3.434544*10^1 - 6.311832*10^0/pe\ (\mu sec). \tag{34}$$

The above parameters give rise to the complete formulas for the timing relations

$$dyads : t = 3.434703*10^{1} - \frac{6.471124*10^{0}}{pe} + \frac{2.591370*10^{-3}}{pe} s \ (\mu sec), \quad (35)$$

$$triads : t = 3.434544*10^{1} - \frac{6.311832*10^{0}}{pe} + \frac{1.828697*10^{-3}}{pe} s \ (\mu sec). \quad (36)$$

Figure 3.7 shows the variation of the processing time as a function of *pe*, for a wide range of the amount of arithmetic *s*.

(2) The degree of degradation due to synchronization overheads

The degree of degradation of the maximum rate, $d_{m\tau}$, due to synchronization overheads is given by

$$d_{m\tau} = pipe \ (s/s_{1/2}). \quad (37)$$

Furthermore we define the degree of degradation of the actual rate of the parallel processing in the local memory access to the actual rate of the vector processing, d_{plv}, as follows:

$$d_{plv} = \frac{pipe \ (s/s_{1/2})}{pipe \ (n/n_{1/2})}, \quad (38)$$

where $s = n$, and the values of $s_{1/2}$ and $n_{1/2}$ is ones in Table 2 and Table 1 (b), respectively. Then we can approximately express the actual rate of the parallel processing in the local memory access, τ_{pl}, by

$$\tau_{pl} = d_{plv}\tau_{v}\, pe, \quad (39)$$

where τ_{v} is the actual rate of the vector processing. It is clear that $d_{plv} = 1$ when s, $n \rightarrow \infty$. Table 3 shows the values of d_{plv} for $pe = 1$, 128 and several values of s. The values of d_{plv} for other *pe* take intermediate ones between both values for $pe = 1$ and 128 shown in Table 3.

(3) Remark

Under such a condition as mentioned in Subsection 4.1 (4), we could expect that the variation of maximum performance as a function of *pe* would become as follows:

$$dyads : \tau_{\infty} = 771 \ pe \ (Mflop/sec), \quad (40)$$

$$triads : \tau_{\infty} = 1093 \ pe \ (Mflop/sec). \quad (41)$$

Hence if we could improve bottlenecks in the local memory access by means of one addition of the load pipeline without any arithmetic enhancement, not only we could enjoy the maximum rate for dyadic operations close upon the peak arithmetic performance, but also we could realize for triadic operations the enormous capability of 140 *Gflop*/*sec* when $pe = 128$.

4.3 Comparison

We note the followings:

(i) The value of a_0 (= t_0) for dyads in Table 1 (b) is smaller than 0.75 μsec (CRAY X-MP, p.143 of [9]), 0.82 μsec (CRAY-1, p.143 of [9]) and 1.48 μsec (CRAY-2, p.204 of [9]). The ratio of τ_{∞} for dyads in Table 1 (b) to τ_{∞} in Table 2.1 or Table 2.7 of [9] is 5.57 (CRAY X-MP), 17.72 (CRAY-1) and 6.96 (CRAY-2), and hence the value of $n_{1/2}$ for dyads in Table 1 (b) becomes about a 5.5-fold value of CRAY X-MP, a 16.0-fold value of CRAY-1 and a 3.5-fold value of CRAY-2 of $n_{1/2}$ in Table 2.1 and Table 2.7 of [9], because $n_{1/2}$ is proportional to τ_{∞}.

(ii) The value of a_0 (= t_0) for $pe = 2$ and dyads in Table 2 is smaller than 45 μsec (TASKS on 2-CPU CRAY X-MP, p.146 of [9]), but larger than 28 μsec (LOCKS) and 14 μsec (EVENTS). The ratio of τ_{∞} for $pe = 2$ and dyads in Table 2 to τ_{∞} in Table 2.2 of [9] is 6.02 (TASKS) and 5.59 (LOCKS and EVENTS), and hence the value of $s_{1/2}$ for $pe = 2$ and dyads in Table 2 becomes about a 4.3-fold value of TASKS, a 6.2-fold value of LOCKS and a 12.5-fold value of EVENTS of $s_{1/2}$ in Table 2.2 of [9], because $s_{1/2}$ is proportional to τ_{∞}.

(iii) Hence all sorts of the start-up time of the NWT in the local memory access seem to be not very large, but we must continue to make efforts with the decrease of every kind of start-up time commensurate with an increased maximum performance.

We had been vigorously continuing subsequent improvements to the NWT system software, and accomplished an 1.2 to 1.3-fold decrease of synchronization overheads at April 1994, which we will report

in another paper.

5. CONCLUSIONS

The principal conclusion of this work is that the set-up, pipeline start-up and synchronization overheads, and maximum performance of the NWT computer system could be characterized by relatively simple timing equations for a wide range of problem conditions.

Generally speaking, a shorter machine clock period increases τ_∞ without changing $n_{1/2}$, but $s_{1/2}$ increases with τ_∞. A reduction in the set-up, pipeline start-up and synchronization time causes a reduction in both $n_{1/2}$ and $s_{1/2}$. An enlargement of the pipeline performance by some means excepting a shorter clock period involves an increase of $n_{1/2}$. Clearly it is hardly worth making a parallel section if the amount of work is much less than $s_{1/2}$, and really we would like to see parallel sections with $s > s_{1/2}$ if the parallel hardware is to be used efficiently. Hence, in order to shorten the processing time of user programs that are consisted of many parallel sections, each of which has a wide range of a different amount of computational work, not only hardware performance must be improved, but also it is very important that a total number of steps of the machine instruction inserted a user program by a compiler and operating system has to be reduced as much as possible.

In Section 2 we measured the set-up plus pipeline start-up overheads and maximum performance with SISD and SIMD computings of the NWT, and in Subsection 4.1 analysed to estimate several hardware parameters. The obtained results showed that for the short-vector length, performance of the memory access system harmonizes very much with one of arithmetic pipelines, but that for the long-vector length, a small amoumt of degradation of maximum performance is observed. In order to interpret a plausible cause of the latter phenomenon, we introduced a gap model, and computed an amount of gap. The measured value of $n_{1/2}$ seems to be a barely satisfactory one.

In Section 3 we measured the set-up plus pipeline start-up plus synchronization overheads and maximum performance with MIMD computation in the local memory access of the NWT, and in Subsection 4.2 analysed to derive the complete expression for the timing relations on dyadic and triadic operations as a function of *pe* and *s*, and to present the degree of degradation of the maximum rate due to synchronization overheads. The measured values of $s_{1/2}$ are rather large at the moment in this experiment yet.

Taking it as a whole, we think to have realized the enormous capability for the maximum performance of the NWT. On the other hand, all sorts of the start-up time of the NWT in the local memory access seem to be not very large, but we had to have been continuing to make every effort with the decrease of every kind of the start-up time commensurate with an increased maximum performance, and accomplisded an 1.2 to 1.3-fold decrease of synchronization overheads at April 1994, which we will report in another paper.

Most of the overheads incurred arise from the time spent in system software routines supporting user programs. Hence the obtained results in this paper apply only to the system software available at the NAL during the period April to June 1993.

6. REFERENCES

[1] D.A. Calahan, Algorithmic and architectural issues related to vector processors, *Proc. Int. Symp. Large Eng. Syst.* (Pergamon, New York, 1977) 327-339.

[2] D.A. Calahan and W.G. Ames, Vector processors : models and applications, *IEEE Trans. Circuits and Systems CAS-26* (1979) 715-726.

[3] S. Hatayama, The characteristic parameters of the NWT computer system in the global memory access, *NAL TR* (to be specified for publication).

[4] D. Heller, A servey of parallel algorithms in numerical linear algebra, *SIAM Rev. 20* (1978) 740-777.

[5] R.W. Hockney and D.F. Snelling, Characterizing MIMD computers: e.g. Denelcor HEP, in: M. Feilmeier et al., eds., *Parallel Computing 83* (North-Holland, Amsterdam, 1984) 521-526.

[6] R.W. Hockney, (τ_∞, $n_{1/2}$, $s_{1/2}$) measurements on the 2-CPU CRAY X-MP, *Parallel Computing 2* (1985) 1-14.

[7] R.W. Hockney, Performance characterization of the HEP, in: J.S. Kowalik, ed., *MIMD Computation: HEP Supercomputer and its Applications* (MIT Press, Cambridge, MA, 1985) 59-90.

[8] R.W. Hockney, Parametrization of computer performance, *Parallel Computing 5* (1987) 97-103.

[9] R.W. Hockney and C.R. Jesshope, *Parallel Computers 2* (Adam Hilger, Bristol, 1988).

[10] M.J. Jr. Kascic, Vector processing on the CYBER 200, in: C.R. Jesshope and R.W. Hockney, eds., *Infotech State of the Art Report 2* (Infotech Int. Ltd., Maidenhead, 1979) 237-270.

[11] P.M. Kogge, *The Architecture of Pipelined Computers* (McGraw-Hill, New York, 1981).

[12] D.J. Kuck, *The Structure of Computers and Computations Vol. 1* (Wiley, New York, 1978).

[13] H. Miyoshi, et al., Development and achievement of NAL Numerical Wind Tunnel (NWT) for CFD computations, *Proc. Supercomputing '94* (IEEE Computer Society Press, Washington, DC, 1994) 685-692.

Table 1 The characteristic parameters of the NWT with SISD and SIMD computings on a single processing element

(a) The values on the scalar processing

operation	$a_0(\mu sec)$	$a_1(\mu sec/flop)$	$r_\infty(Mflop/s)$	$r_0(Mflop/s)$	$n_{1/2}(flop)$
dyadic op.	$6.437200 * 10^{-1}$	$3.248445 * 10^{-2}$	$3.078396 * 10^{1}$	$1.553470 * 10^{0}$	$1.981625 * 10^{1}$
triadic op.	$7.752634 * 10^{-1}$	$2.102427 * 10^{-2}$	$4.756408 * 10^{1}$	$1.289884 * 10^{0}$	$3.687469 * 10^{1}$

(b) The values on the vector processing

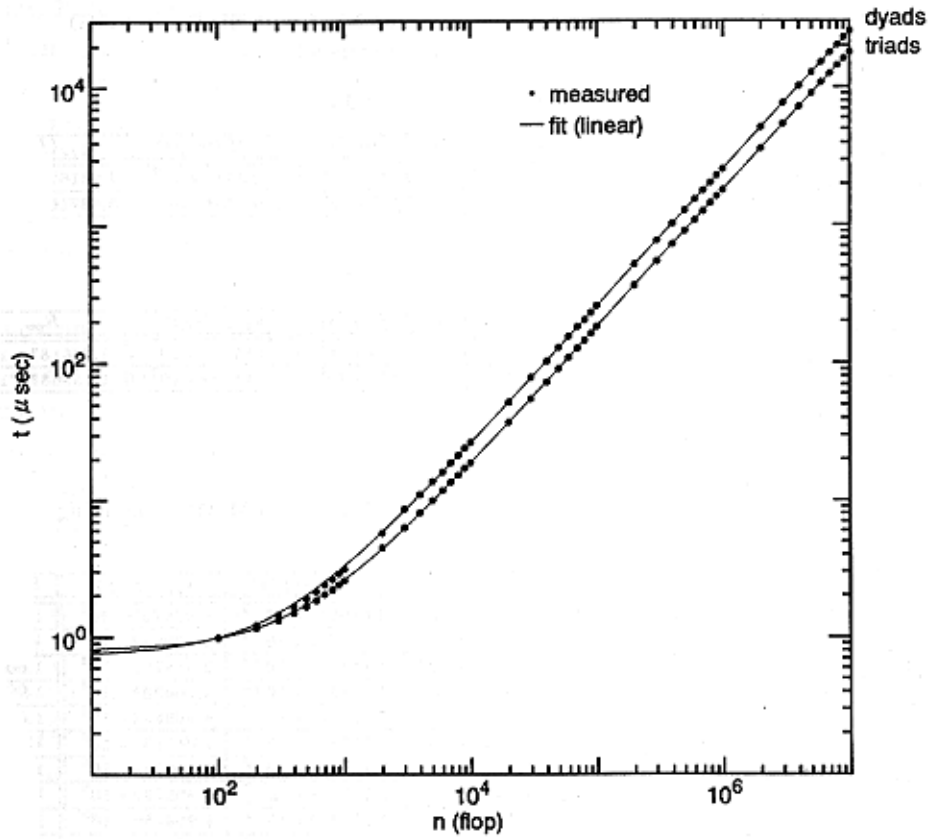
operation	$a_0(\mu sec)$	$a_1(\mu sec/flop)$	$r_\infty(Mflop/s)$	$r_0(Mflop/s)$	$n_{1/2}(flop)$	R_∞	$n_b(flop)$
dyadic op.	$7.401766 * 10^{-1}$	$2.565534 * 10^{-3}$	$3.897824 * 10^{2}$	$1.351029 * 10^{0}$	$2.885078 * 10^{2}$	$1.266187 * 10^{1}$	$2.473941 * 10^{1}$
triadic op.	$8.066134 * 10^{-1}$	$1.813552 * 10^{-3}$	$5.514041 * 10^{2}$	$1.239751 * 10^{0}$	$4.447699 * 10^{2}$	$1.159687 * 10^{1}$	$4.197182 * 10^{1}$

Table 2 The characteristic parameters of the NWT with MIMD computing in the local memory access

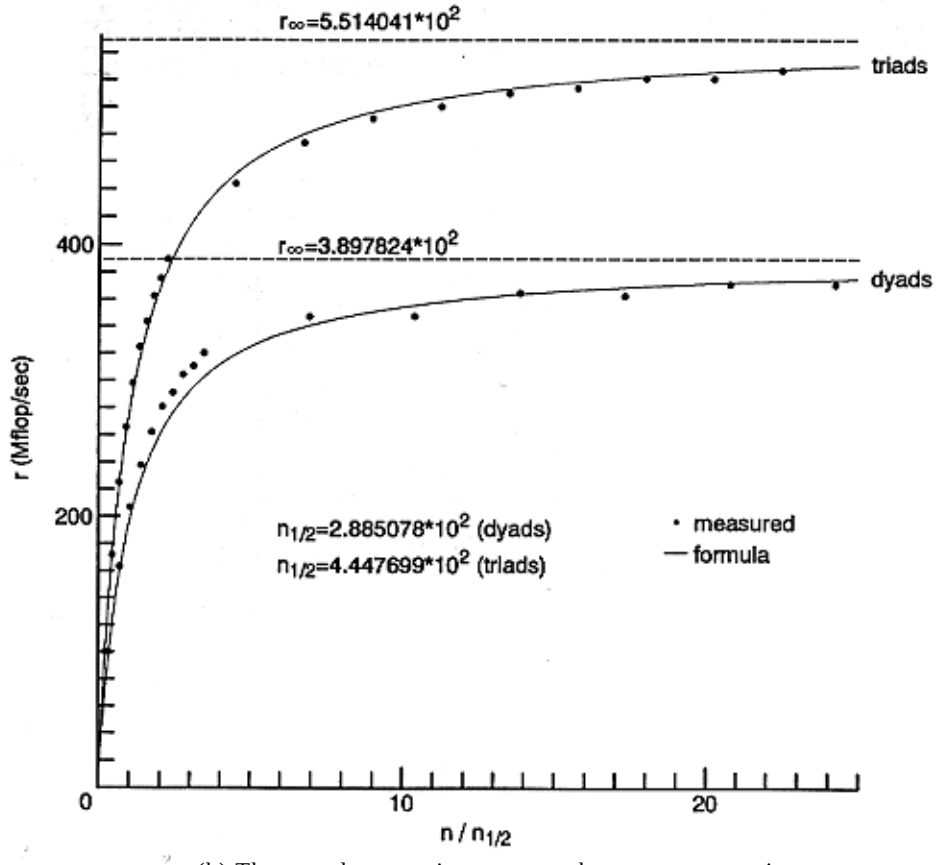
pe	operation	$a_0(\mu sec)$	$a_1(\mu sec/flop)$	$r_\infty(Mflop/s)$	$r_0(Mflop/s)$	$n_{1/2}(flop)$	E_{pe}	$n_b(flop)$
1	dyadic op.	$2.765320 * 10^{1}$	$2.570666 * 10^{-3}$	$3.890043 * 10^{2}$	$3.616218 * 10^{-2}$	$1.075721 * 10^{4}$	1.00	------
	triadic op.	$2.765320 * 10^{1}$	$1.803883 * 10^{-3}$	$5.543597 * 10^{2}$	$3.616218 * 10^{-2}$	$1.532982 * 10^{4}$	1.00	------
2	dyadic op.	$3.176100 * 10^{1}$	$1.278260 * 10^{-3}$	$7.823135 * 10^{2}$	$3.148515 * 10^{-2}$	$2.484706 * 10^{4}$	1.00	$2.484706 * 10^{4}$
	triadic op.	$3.176100 * 10^{1}$	$9.072138 * 10^{-4}$	$1.102276 * 10^{3}$	$3.148515 * 10^{-2}$	$3.500939 * 10^{4}$	1.00	$3.500939 * 10^{4}$
4	dyadic op.	$3.171855 * 10^{1}$	$6.382189 * 10^{-4}$	$1.566861 * 10^{3}$	$3.152729 * 10^{-2}$	$4.969854 * 10^{4}$	1.00	$1.656618 * 10^{4}$
	triadic op.	$3.171855 * 10^{1}$	$4.499608 * 10^{-4}$	$2.222416 * 10^{3}$	$3.152729 * 10^{-2}$	$7.049181 * 10^{4}$	1.00	$2.349727 * 10^{4}$
8	dyadic op.	$3.173214 * 10^{1}$	$3.209723 * 10^{-4}$	$3.115534 * 10^{3}$	$3.151379 * 10^{-2}$	$9.886255 * 10^{4}$	1.00	$1.412322 * 10^{4}$
	triadic op.	$3.173214 * 10^{1}$	$2.257243 * 10^{-4}$	$4.430183 * 10^{3}$	$3.151379 * 10^{-2}$	$1.405792 * 10^{5}$	1.00	$2.008274 * 10^{4}$
16	dyadic op.	$3.149133 * 10^{1}$	$1.608211 * 10^{-4}$	$6.218090 * 10^{3}$	$3.175477 * 10^{-2}$	$1.958159 * 10^{5}$	1.00	$1.305439 * 10^{4}$
	triadic op.	$3.149133 * 10^{1}$	$1.138919 * 10^{-4}$	$8.780256 * 10^{3}$	$3.175477 * 10^{-2}$	$2.765019 * 10^{5}$	1.00	$1.843346 * 10^{4}$
32	dyadic op.	$3.318714 * 10^{1}$	$8.012446 * 10^{-5}$	$1.248058 * 10^{4}$	$3.013215 * 10^{-2}$	$4.141949 * 10^{5}$	1.00	$1.336113 * 10^{4}$
	triadic op.	$3.318714 * 10^{1}$	$5.663572 * 10^{-5}$	$1.765670 * 10^{4}$	$3.013215 * 10^{-2}$	$5.859754 * 10^{5}$	1.00	$1.890243 * 10^{4}$
64	dyadic op.	$3.392477 * 10^{1}$	$4.062954 * 10^{-5}$	$2.461263 * 10^{4}$	$2.947699 * 10^{-2}$	$8.349780 * 10^{5}$	1.00	$1.325362 * 10^{4}$
	triadic op.	$3.392477 * 10^{1}$	$2.865816 * 10^{-5}$	$3.489408 * 10^{4}$	$2.947699 * 10^{-2}$	$1.183773 * 10^{6}$	1.00	$1.879005 * 10^{4}$
128	dyadic op.	$3.447933 * 10^{1}$	$2.024458 * 10^{-5}$	$4.939594 * 10^{4}$	$2.900288 * 10^{-2}$	$1.703139 * 10^{6}$	1.00	$1.341054 * 10^{4}$
	triadic op.	$3.447933 * 10^{1}$	$1.428595 * 10^{-5}$	$6.999885 * 10^{4}$	$2.900288 * 10^{-2}$	$2.413513 * 10^{6}$	1.00	$1.900404 * 10^{4}$

Table 3 The degree of degradation of the actual rate of the parallel processing in the local memory access to the actual rate of the vector processing

s	pe	$d_{plv}(dyads)$	$d_{plv}(triads)$
6,400	1	0.389837	0.314994
	128	0.003912	0.002829
12,800	1	0.555605	0.470844
	128	0.007628	0.005459
100,000	1	0.905481	0.870935
	128	0.055619	0.039962
1,000,000	1	0.989643	0.985340
	128	0.370047	0.293084
10,000,000	1	0.998954	0.998514
	128	0.854496	0.805610

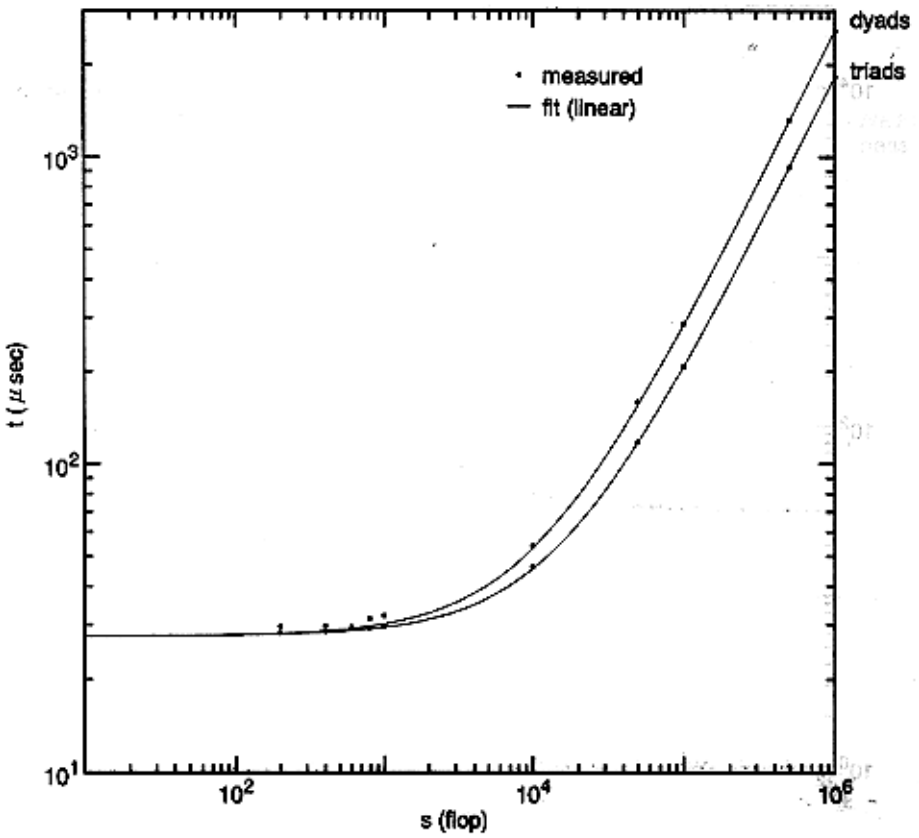


(a) The timing relations on the vector processing

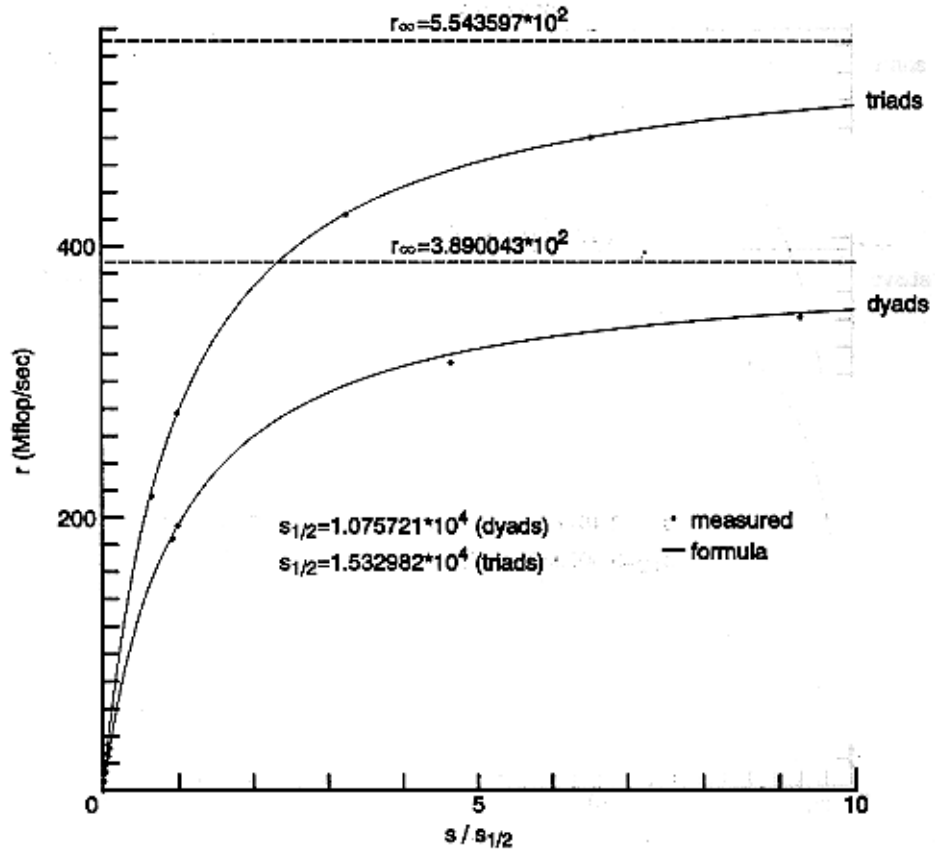


(b) The actual processing rates on the vector processing

Fig. 1 The (τ_∞, $n_{1/2}$) benchmark test on a single processing element

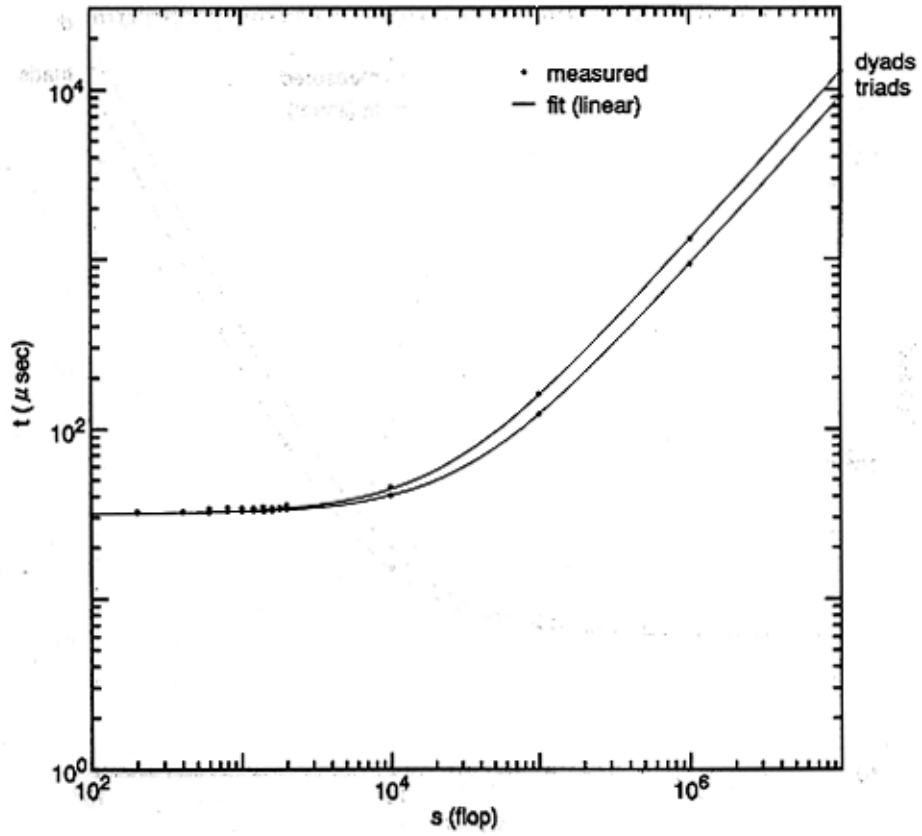


(a) The timing relation as a function of the amount of arithmetic

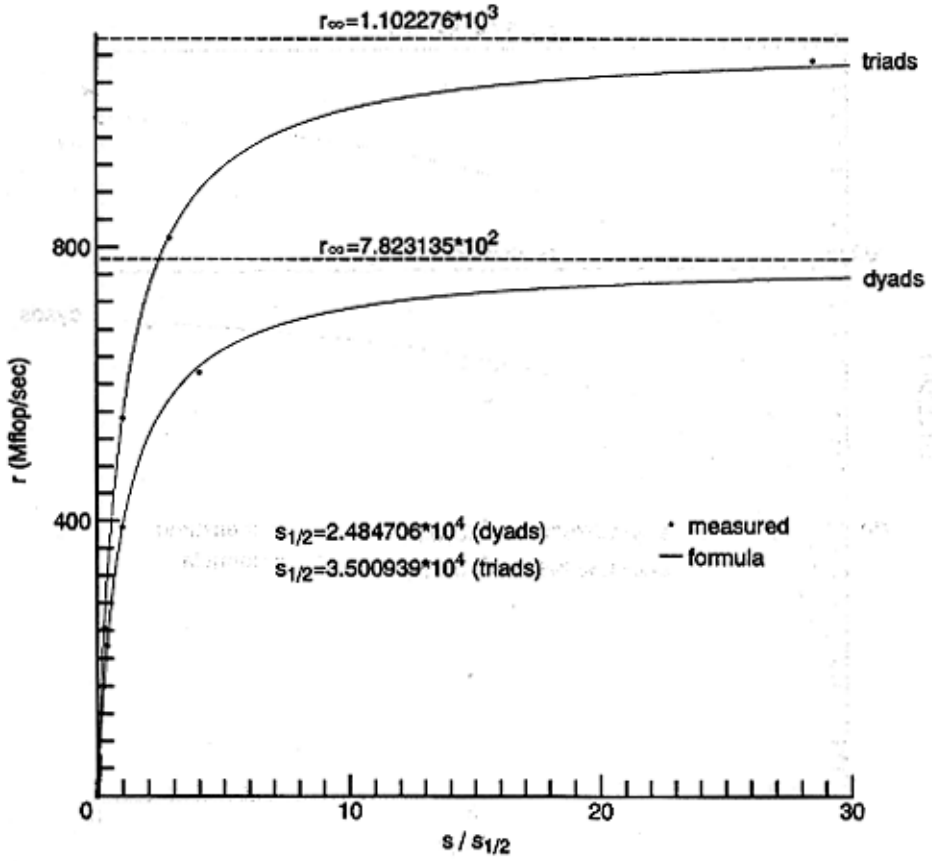


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.1 The (τ_∞, $s_{1/2}$) benchmark test (pe = 1)

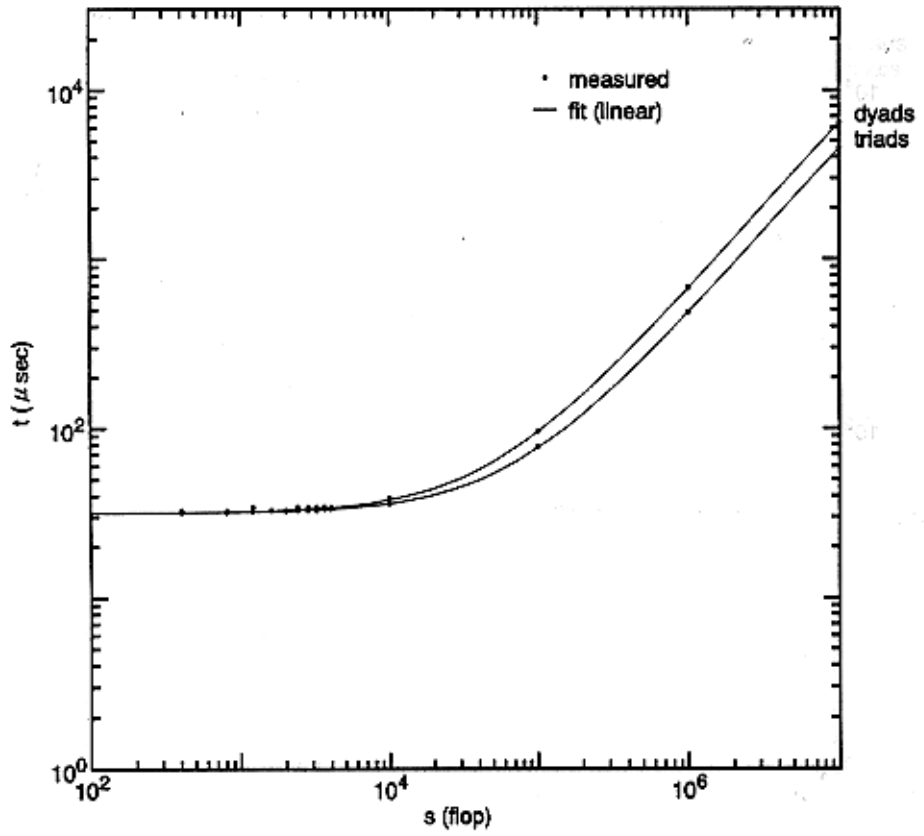


(a) The timing relation as a function of the amount of arithmetic

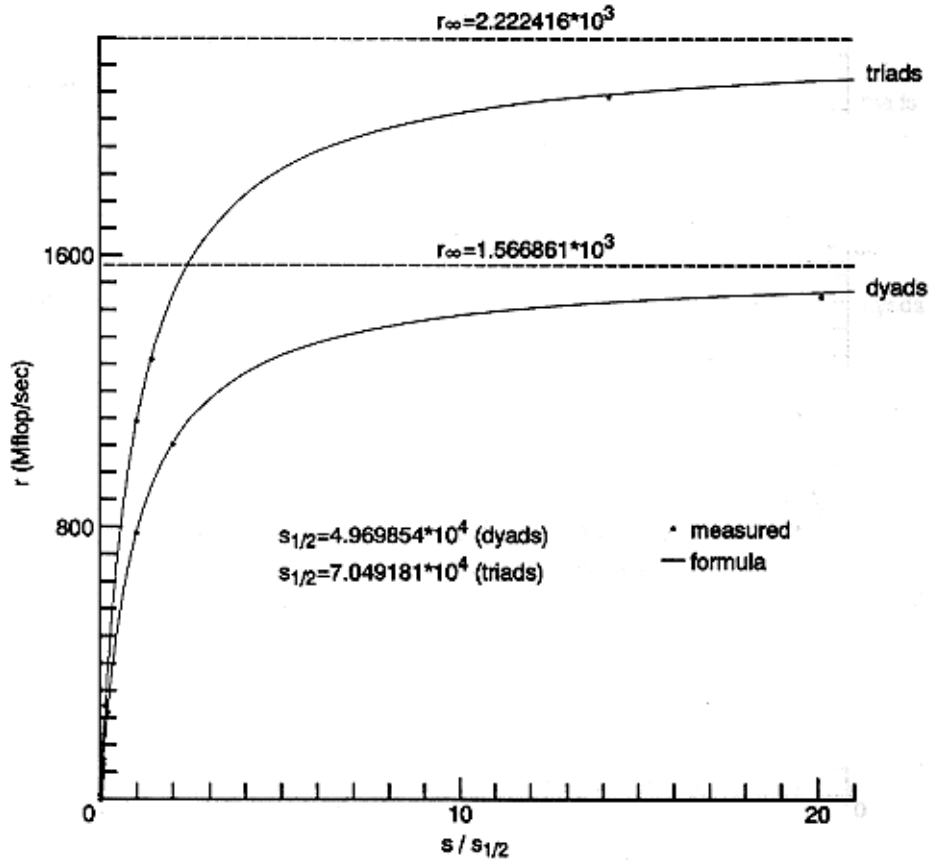


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.2 The (τ_∞, $s_{1/2}$) benchmark test (pe = 2)

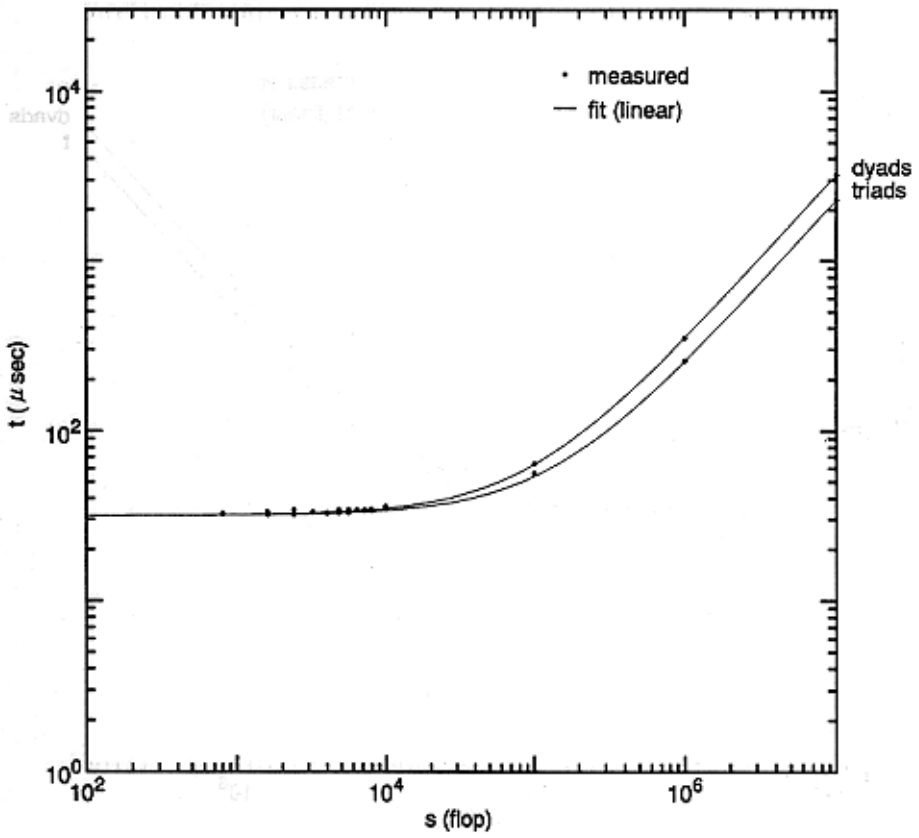


(a) The timing relation as a function of the amount of arithmetic

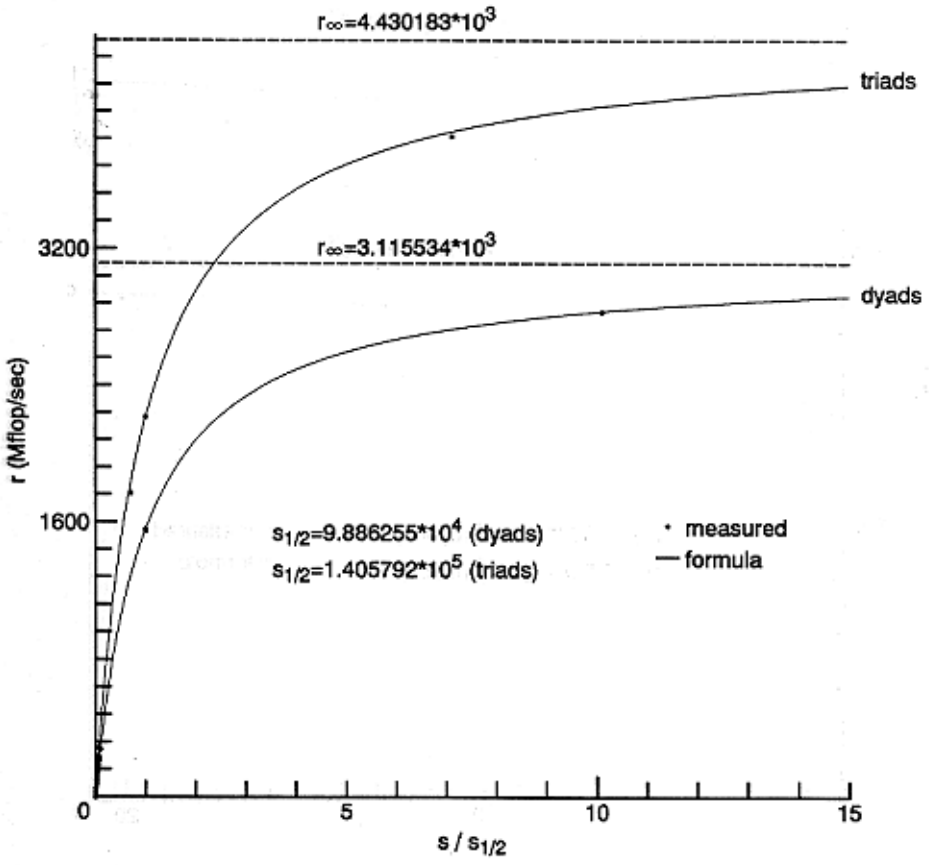


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.3 The (τ_∞, $s_{1/2}$) benchmark test (pe = 4)

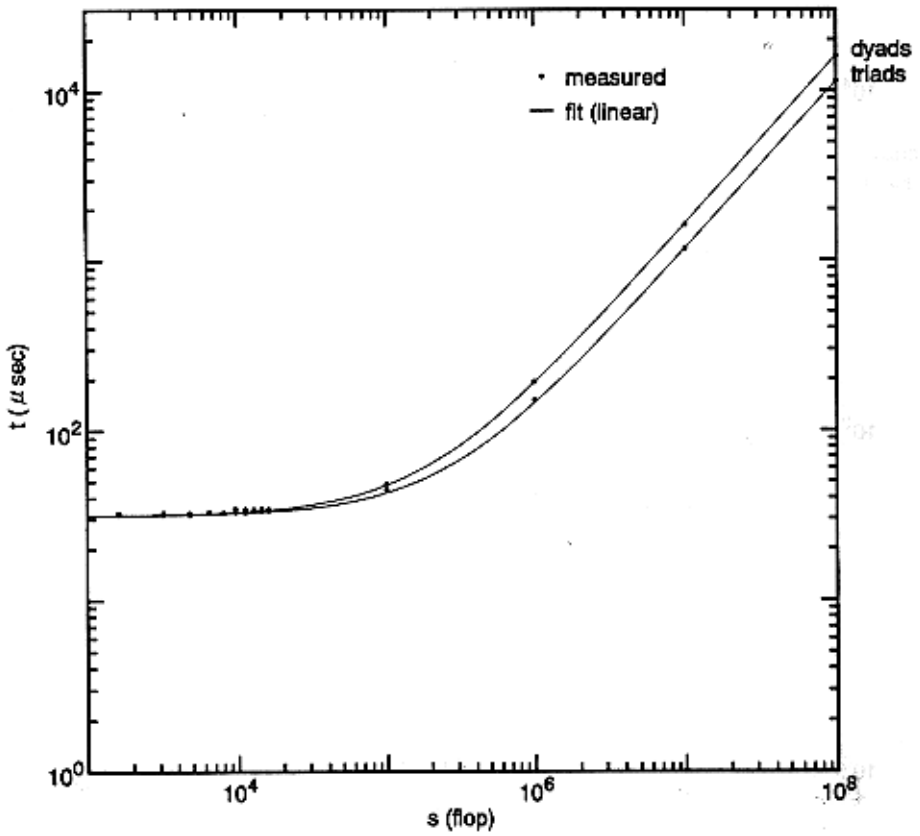


(a) The timing relation as a function of the amount of arithmetic

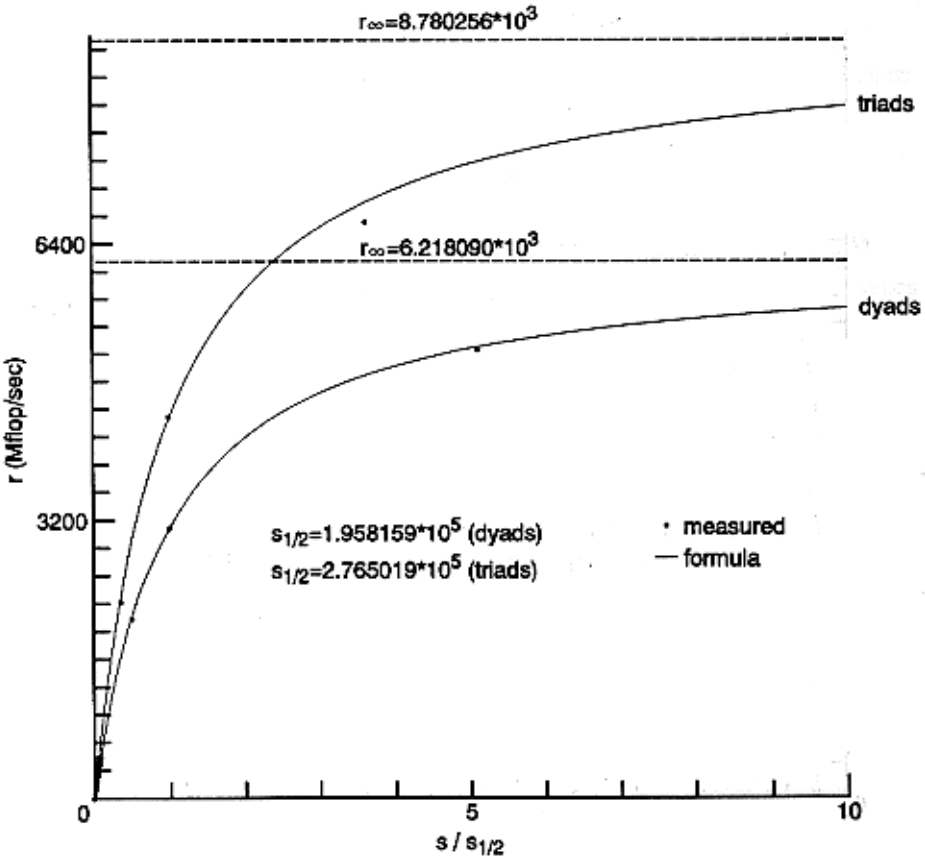


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.4 The (τ_∞, $s_{1/2}$) benchmark test (pe = 8)

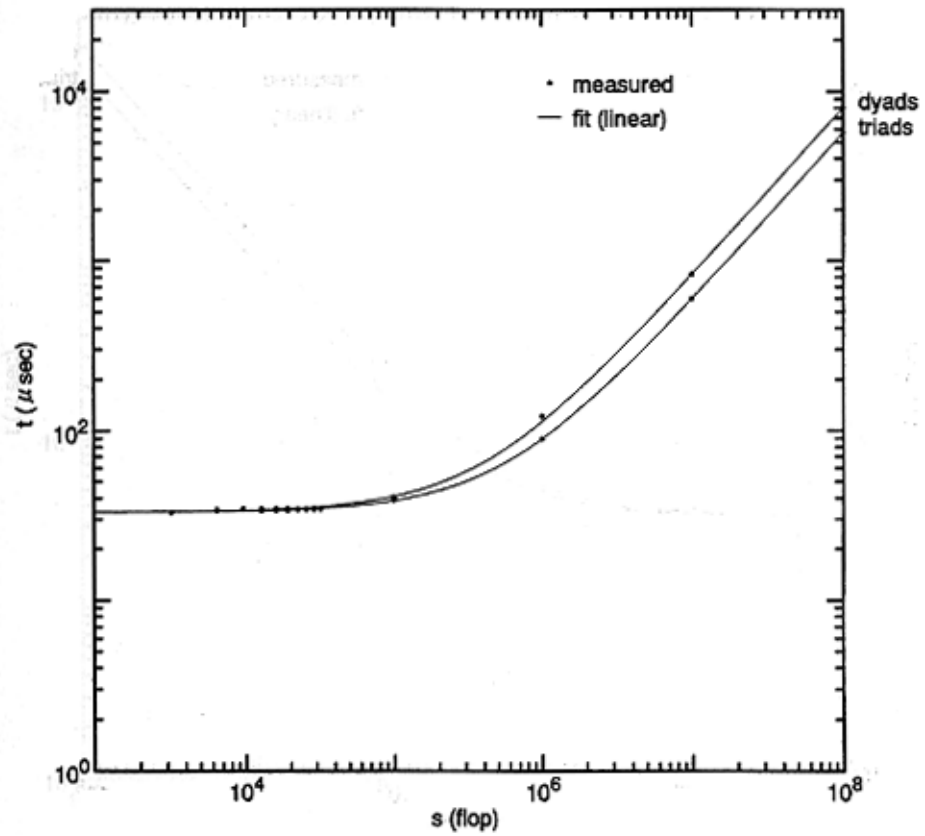


(a) The timing relation as a function of the amount of arithmetic

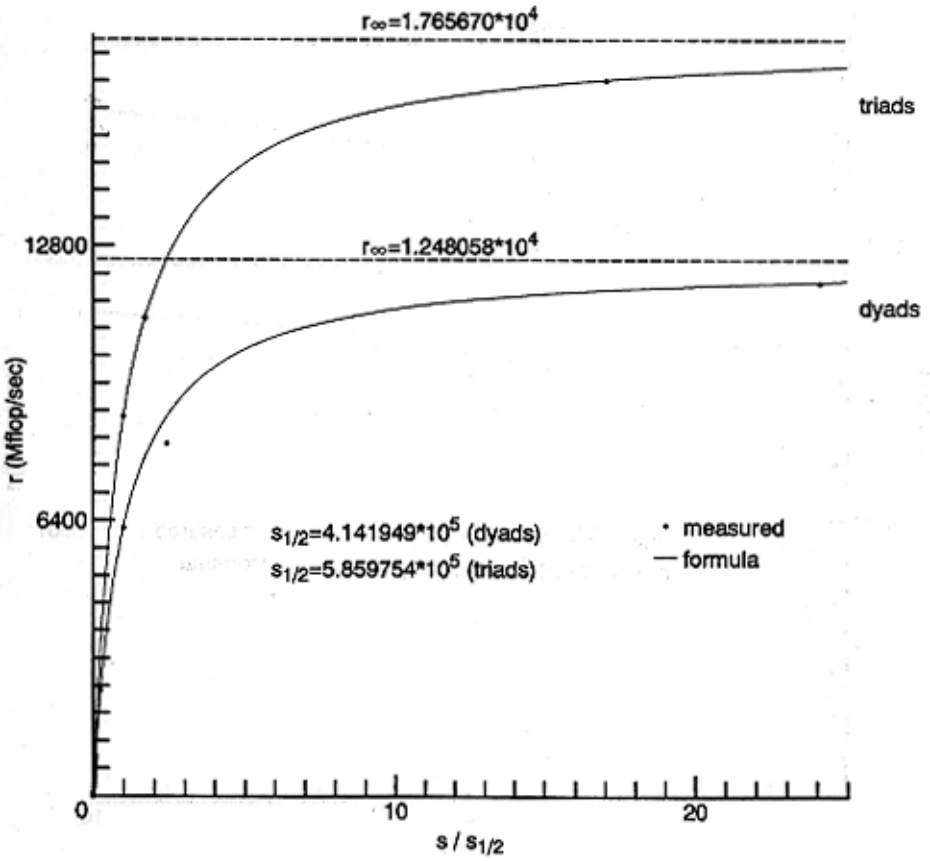


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.5 The (τ_∞, $s_{1/2}$) benchmark test (pe = 16)

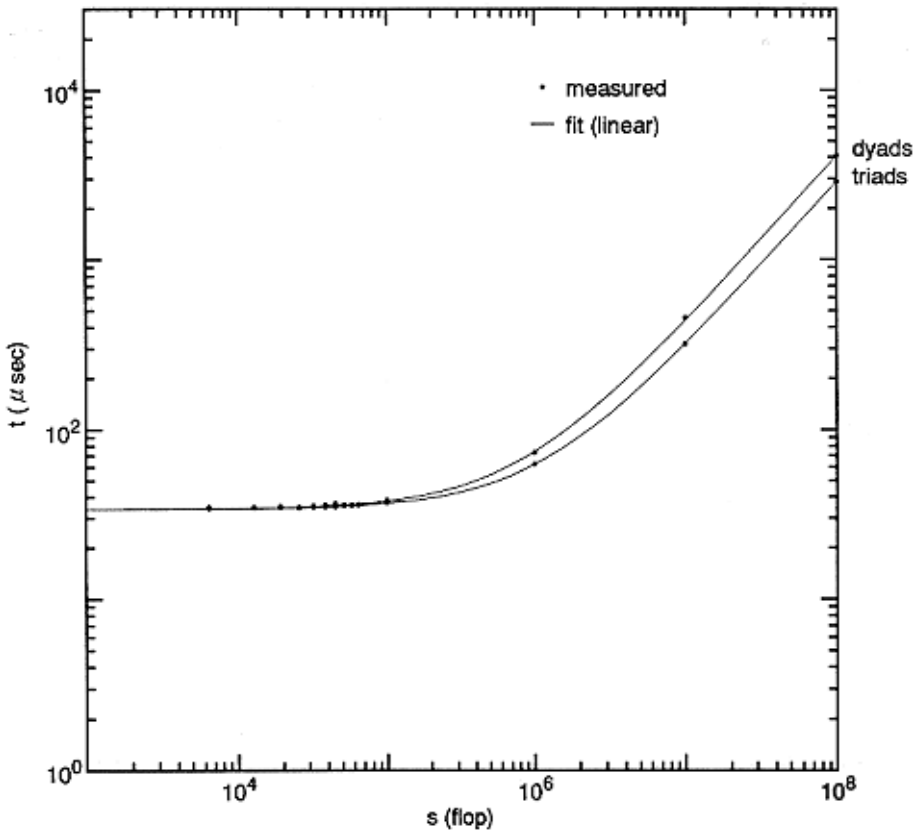


(a) The timing relation as a function of the amount of arithmetic

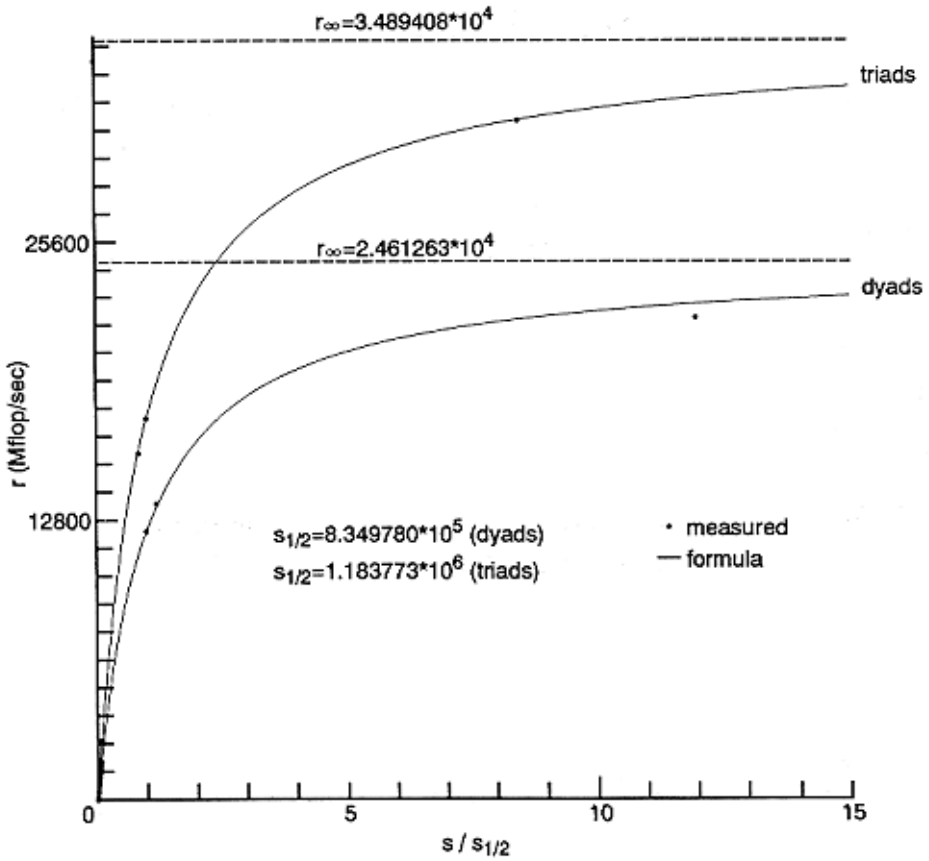


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.6 The (τ_∞, $s_{1/2}$) benchmark test (pe = 32)

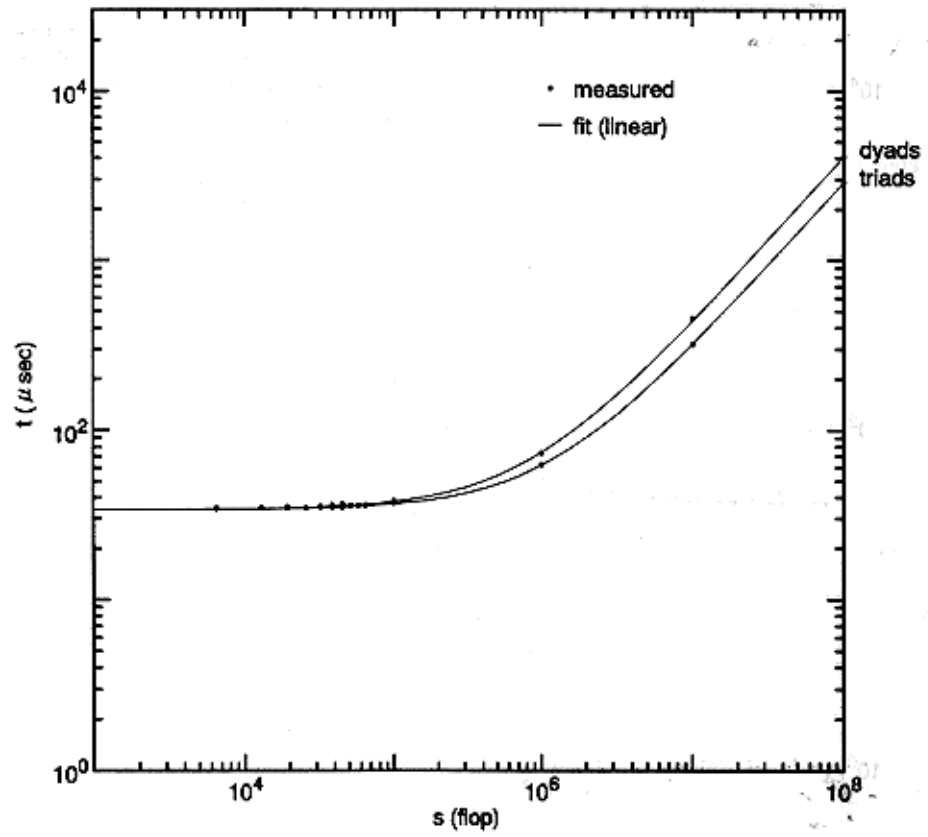


(a) The timing relation as a function of the amount of arithmetic

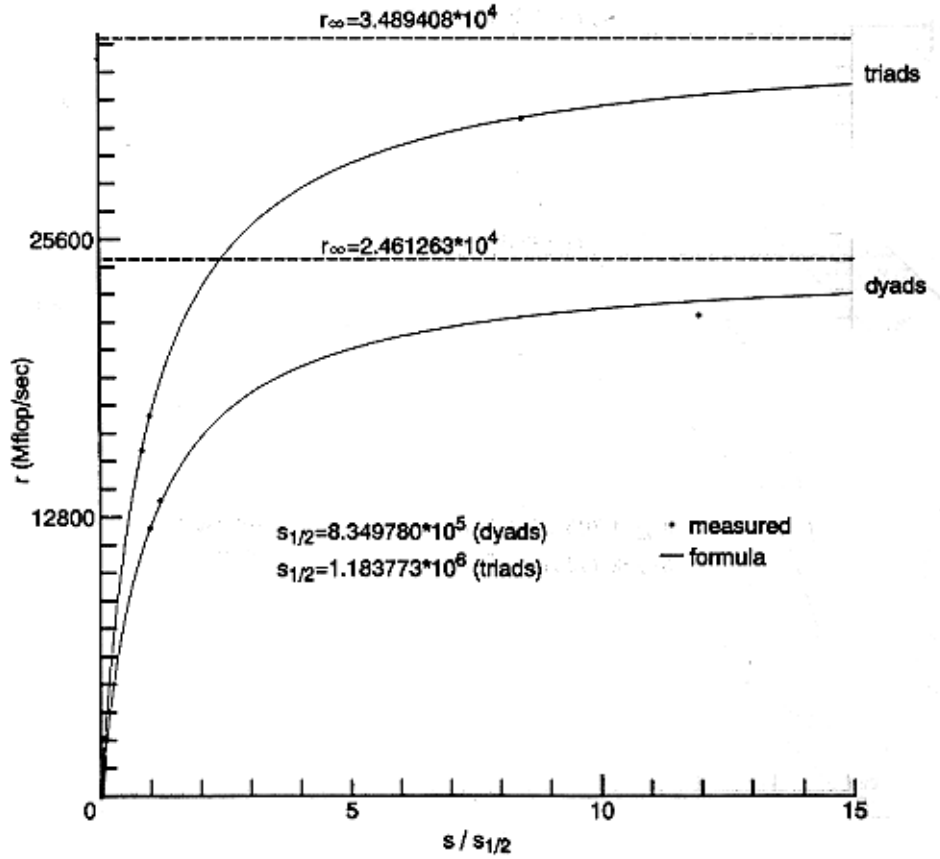


(b) The actual processing rate as a function of the amount of arithmetic

Fig. 2.7 The (τ_∞, $s_{1/2}$) benchmark test (pe = 64)



(a) The timing relations as a function of the amount of arithmetic



(b) The actual processing rates as a function of the amount of arithmetic

Fig. 2.8 The (τ_∞, $s_{1/2}$) benchmark test (pe = 128)

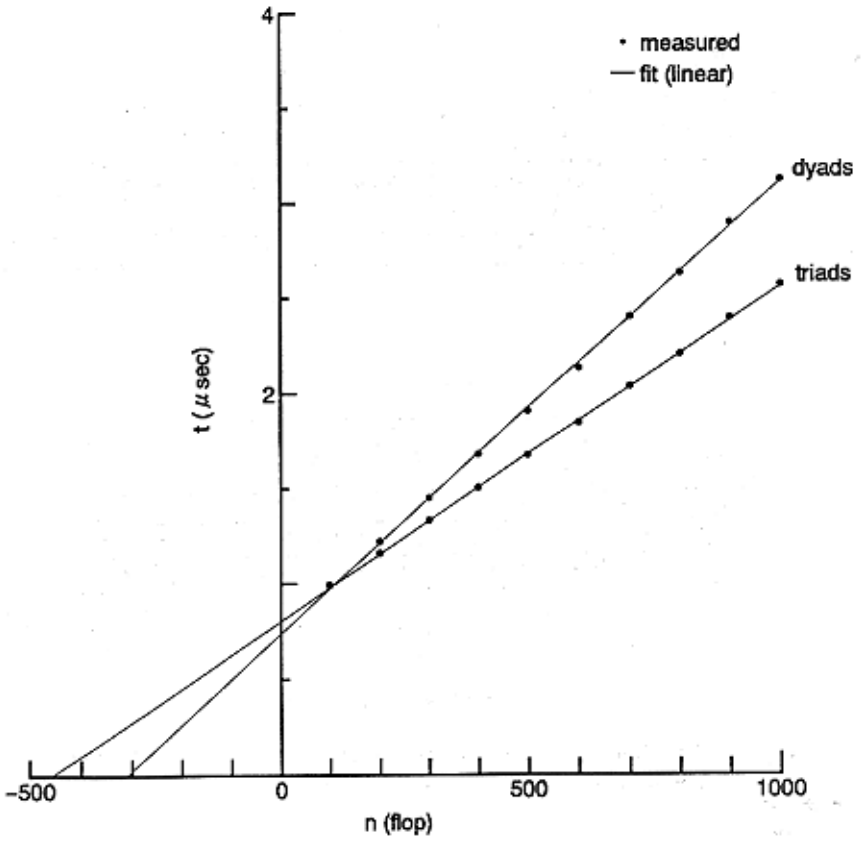
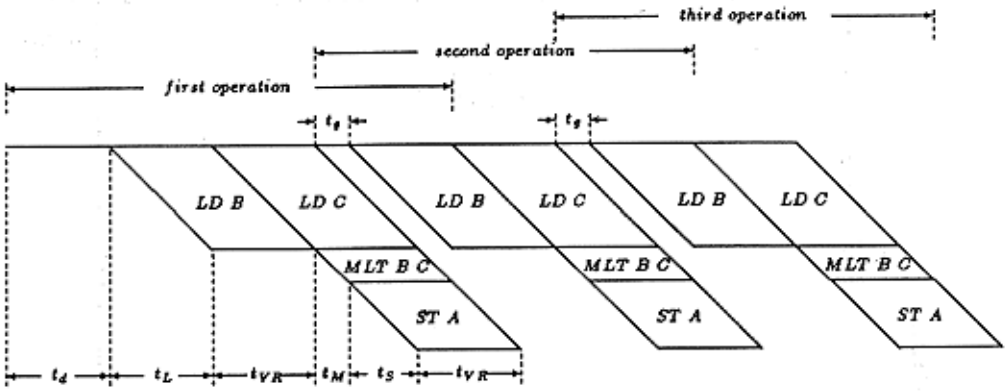
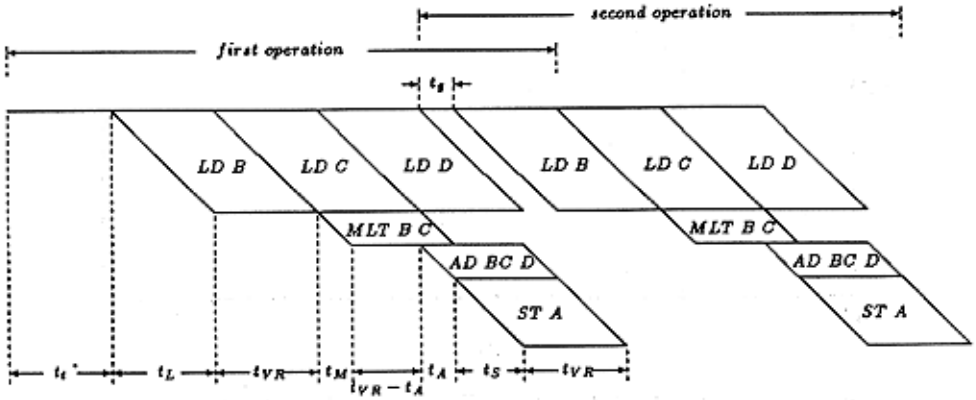


Fig. 3.1 The timing relations as a function of the short-vector length on the (τ_∞, $n_{1/2}$) benchmark

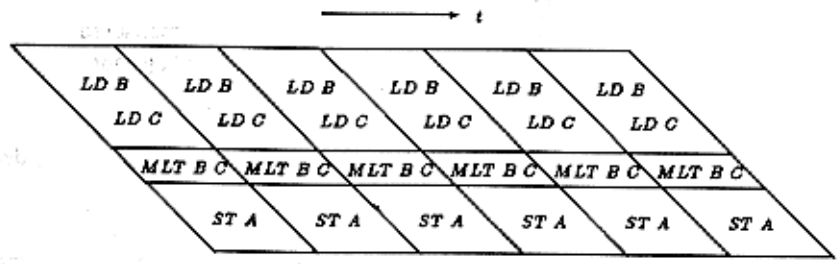


(a) The dyadic operations

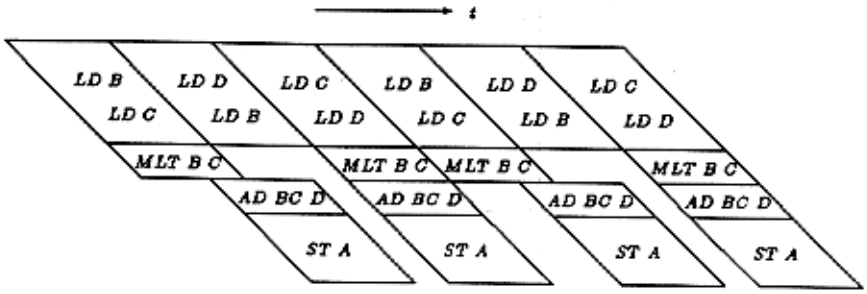


(b) The triadic operations

Fig. 3.2 The time chart of the vector pipelines



(a) The dyadic operations



(b) The triadic operations

Fig. 3.3 The time chart of the vector pipelines when two load pipelines could be available at the same time

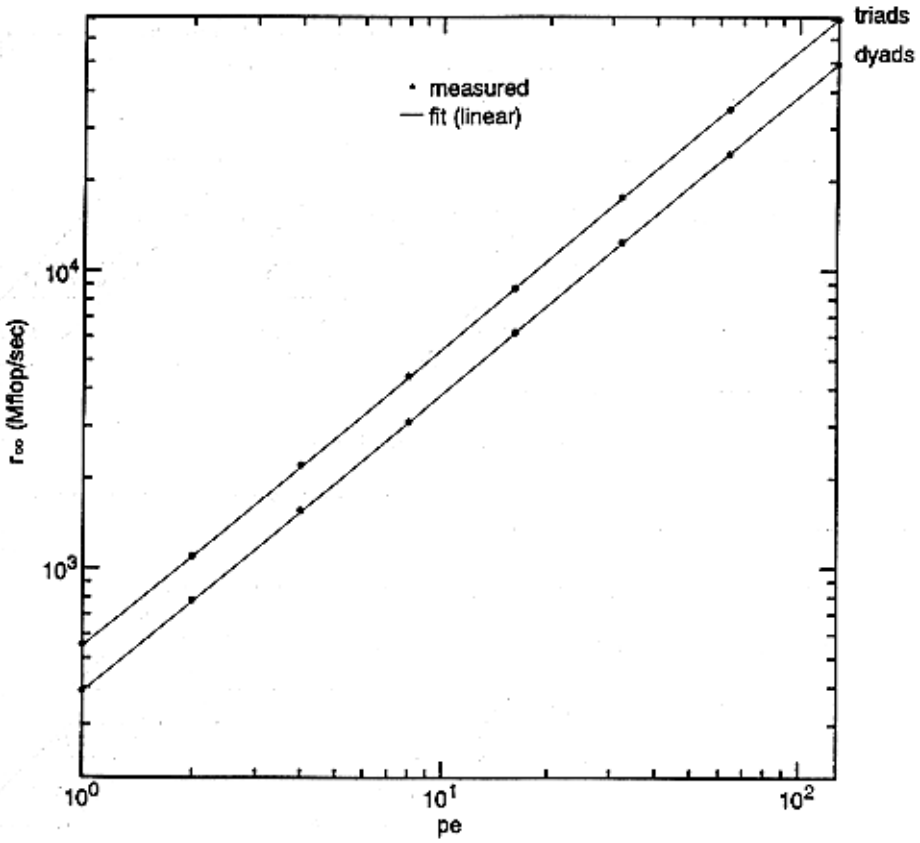


Fig. 3.4 The maximum rates against the number of processing elements with MIMD computing in the local memory access

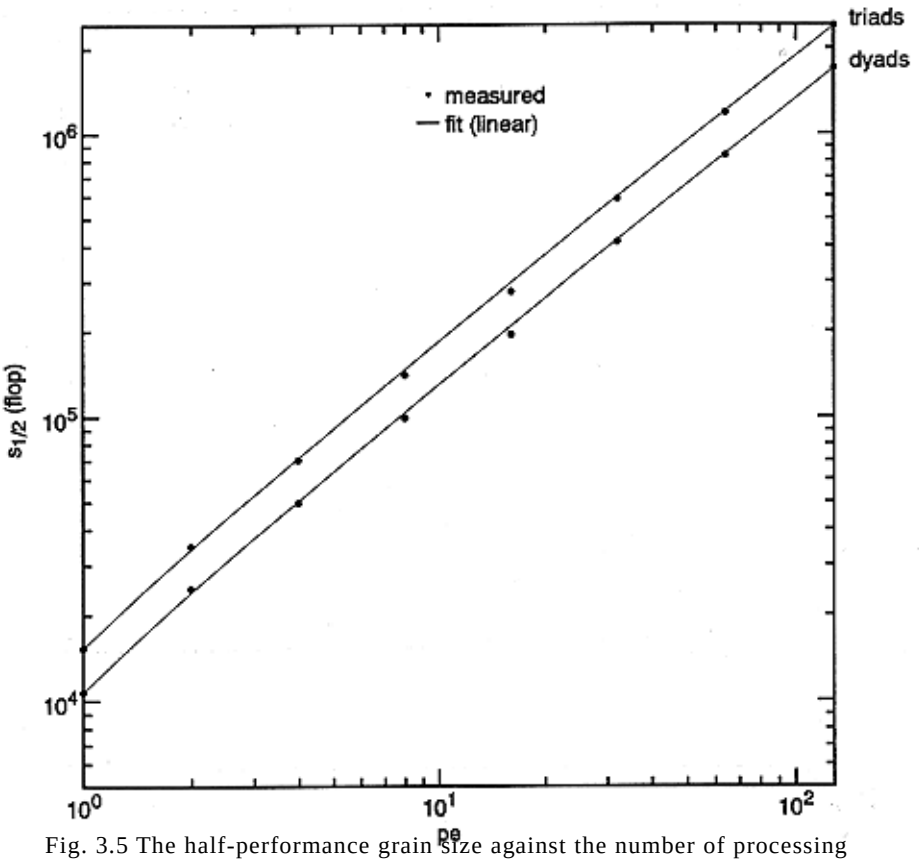


Fig. 3.5 The half-performance grain size against the number of processing elements with MIMD computing in the local memory access

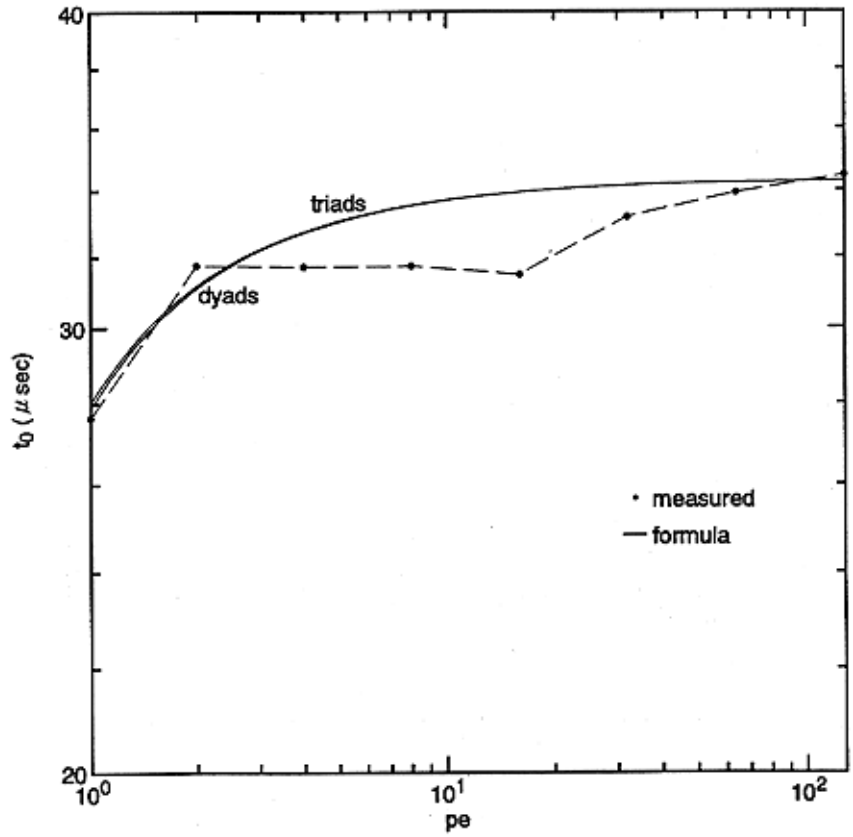
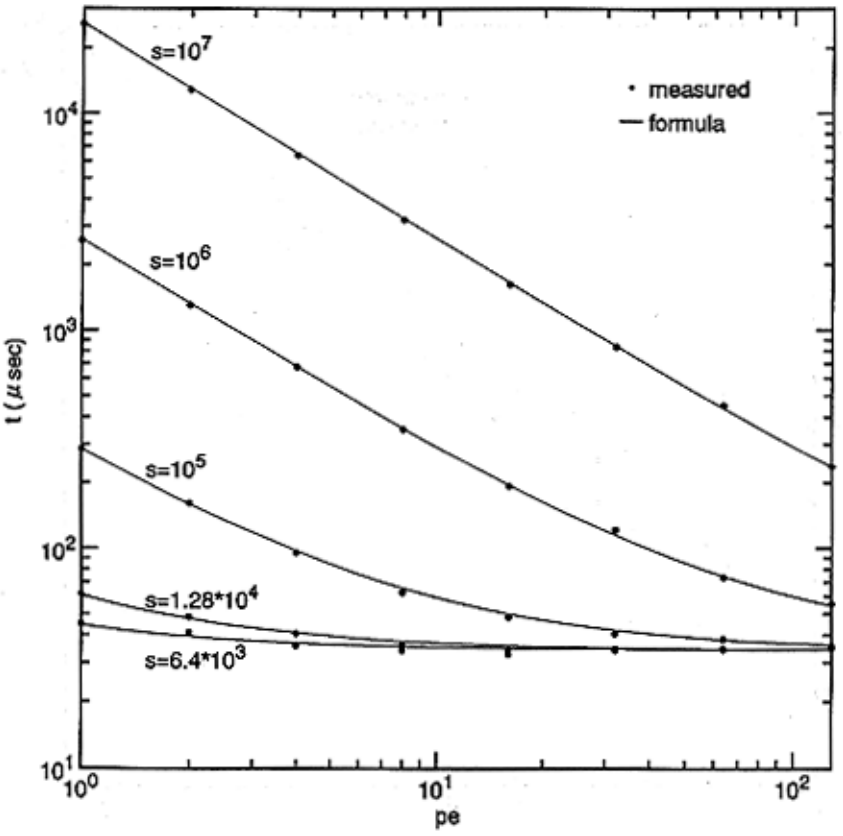
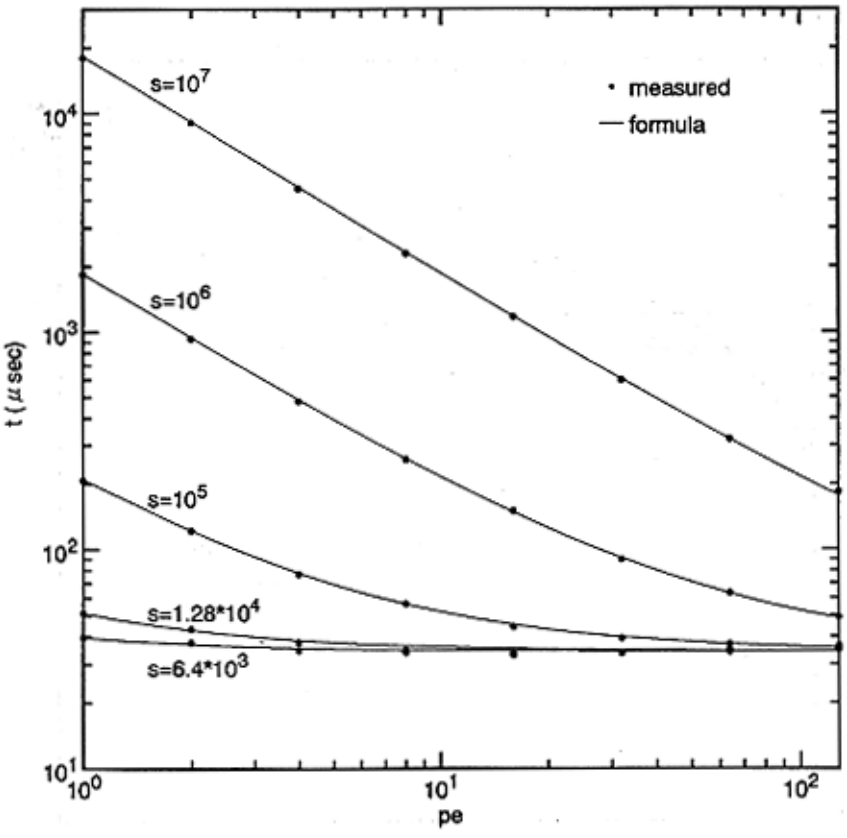


Fig. 3.6 The start-up time against the number of processing elements with MIMD computing in the local memory access



(a) dyads



(b) triads

Fig. 3.7 The timing relations against the number of processing elements with MIMD computing in the local memory access