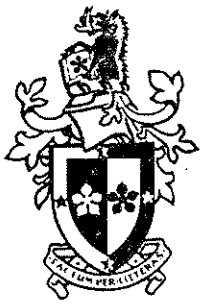


LABORATORY FOR CONCURRENT COMPUTING SYSTEMS

COMPUTER SYSTEMS ENGINEERING
School of Electrical Engineering
Swinburne Institute of Technology
John Street, Hawthorn 3122, Victoria, Australia.

Asilomar SISAL Workshop Presentation 1990

Technical Report 31-018

G.K. Egan

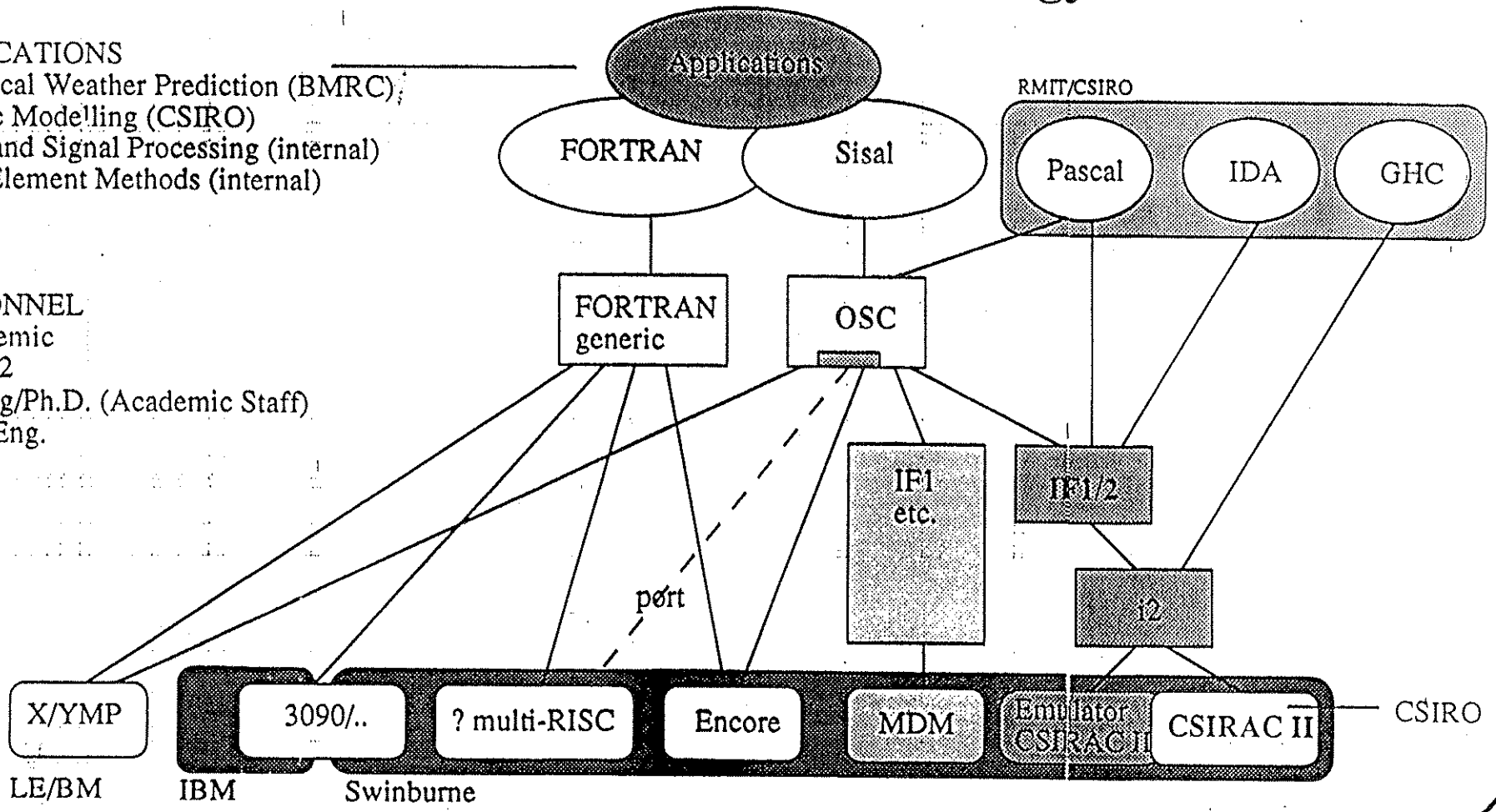
Abstract

This document contains the material presented by invitation at the Asilomar SISAL Workshop held in California June of 1990; the Workshop was hosted by the Lawrence Livermore National Laboratory.

Laboratory for Concurrent Computing Systems
Swinburne Institute of Technology

APPLICATIONS
Numerical Weather Prediction (BMRC)
Seismic Modelling (CSIRO)
Image and Signal Processing (internal)
Finite Element Methods (internal)

PERSONNEL
2 Academic
3 CSO/2
2 M.Eng/Ph.D. (Academic Staff)
2-4 M.Eng.



IF1 Translator (Neil Webb)

done:

- support for multiple languages
  - Sisal
  - Ida
  - Pascal
    - CSIRAC II Pascal libraries written in Sisal
    - these include set operations, I/O

-IF2

- D edges and data structures
- functon hooks
- pragmas (where known!)

To be done:

-unions,tagcase,if-then-else
-complete IF2
-new object store allocator
-vector ops when permitted

no problems anticipated

CSIRAC II - MDM

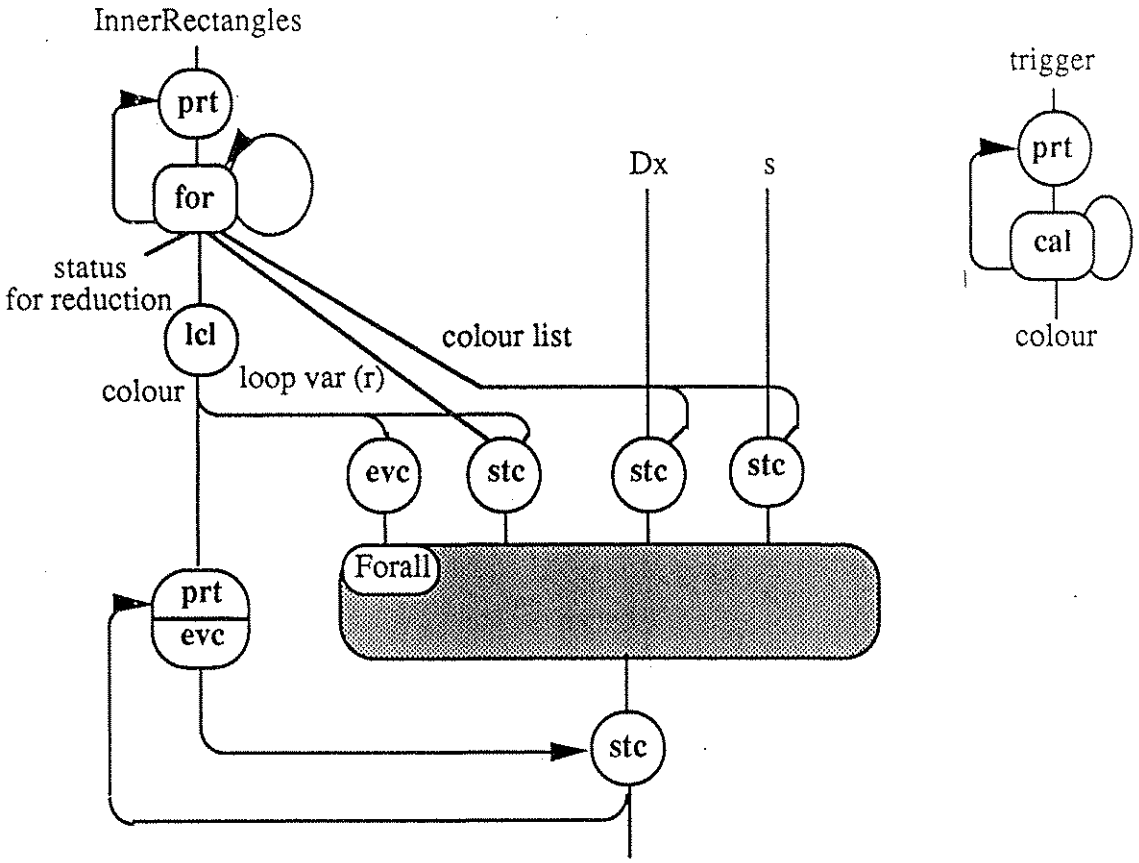
	Manchester DFM			CSIRAC II		
	Critical Path	Aver. Parallel	Total Nodes	Critical Path	Aver. Parallel	Total Nodes
Loop1	48	11.9	573	61	9	571
Loop2	124	6.5	810	66	5	352
Loop3	59	5.1	303	59	5	279
Loop4	102	220.9	22533	52	12	616
Loop5	184	13.8	2537	155	10	1569
Loop6	181	9.6	1737	156	11	1782
Loop7	53	15.8	840	52	14	751
Loop8	113	37.5	4238	100	21	2068
Loop9	78	16.6	1297	40	8	301
Loop10	95	88.1	8374	79	135	10636

Pi

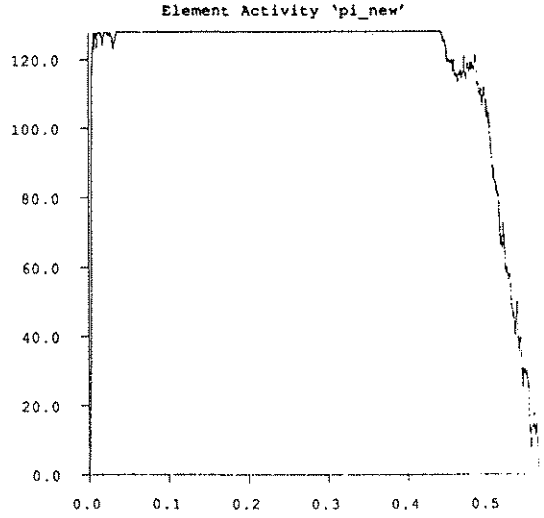
```
define main

function main(returns real)

let
  InnerRectangles := 100;
  OuterRectangles := 100;
  Dx := 1.0 / Real(InnerRectangles*OuterRectangles);
in
  for s in 1 , OuterRectangles returns value of sum
    for r in 1 , InnerRectangles
  returns value of sum
      4.0*Dx / (1.0 + Dx*Real(r+(s-1)*InnerRectangles)
        * Dx*Real(r+(s-1)*InnerRectangles))
    end for
  end for
end let
end function
```

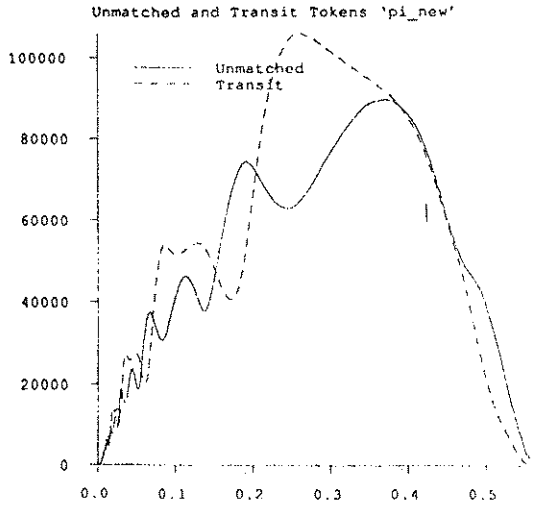
```
define recursive
  function recursive(returns real)
  function Area(D: integer; dx, L,R: real returns real)
    let
      Mid := (L + R) * 0.5;
      NewD := D / 2;
    in
      if D = 0 then
        (R - L) * 4.0/(1.0 + L * L)
      else
        Area(NewD, dx, L, Mid) + Area(NewD, dx, Mid, R)
      end if
    end let
  end function
    let
      Rectangles := 10000;
      dx := 1.0 / Real(Rectangles);
    in
        Area(Rectangles,dx,A,B)
    end let
  end function
```

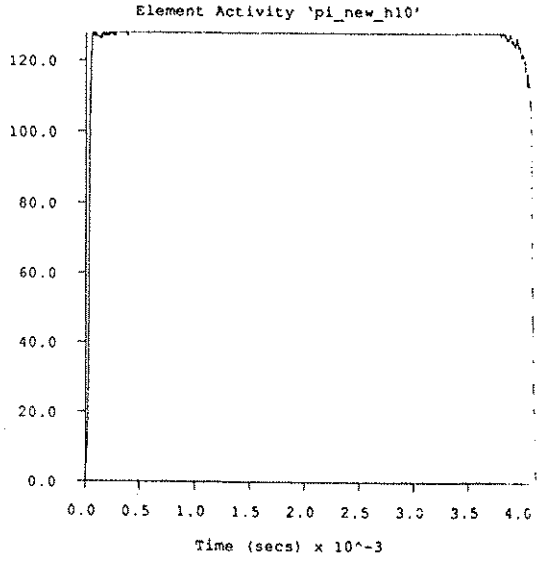
Loops and Functions

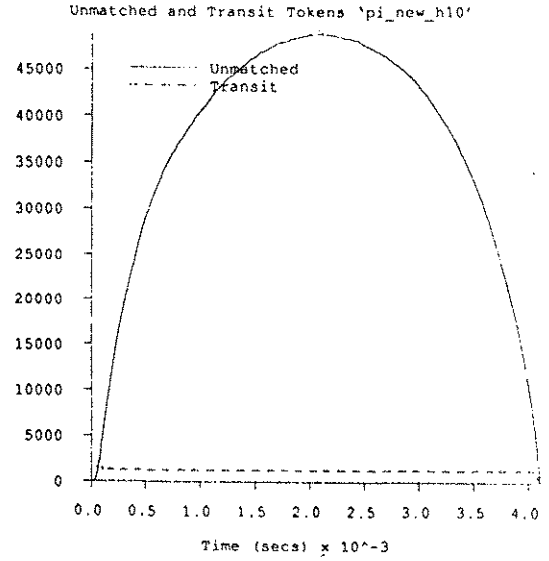


Recursion

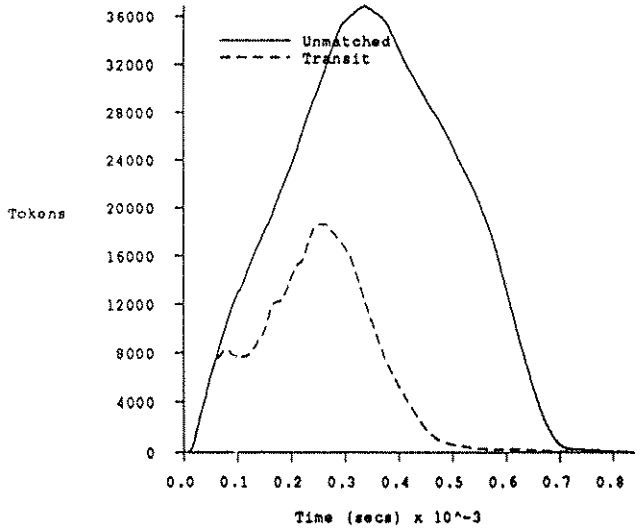

Element Activity 'pi_new'
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5

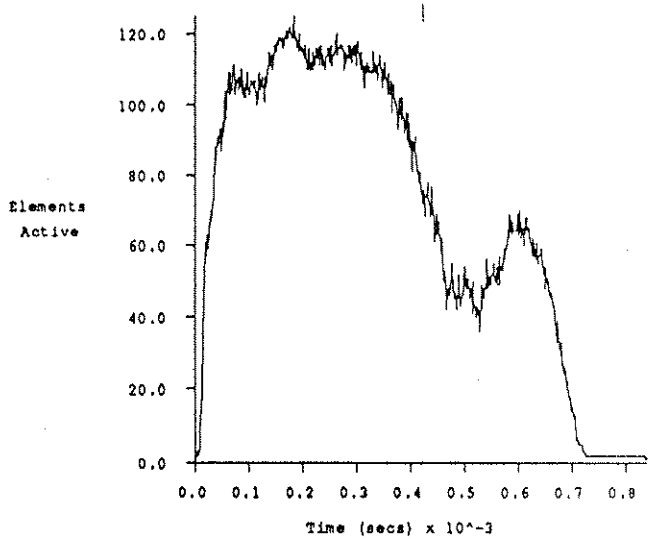

Unmatched and Transit Tokens 'pi_new'
Unmatched
Transit
100000
80000
60000
40000
20000
0
0.0
0.1
0.2
0.3
0.4
0.5

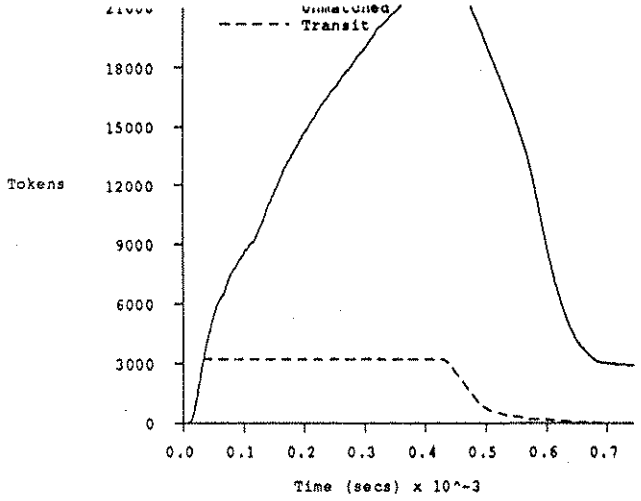

Element Activity 'pi_new_h10'
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Time (secs) x 10^-3

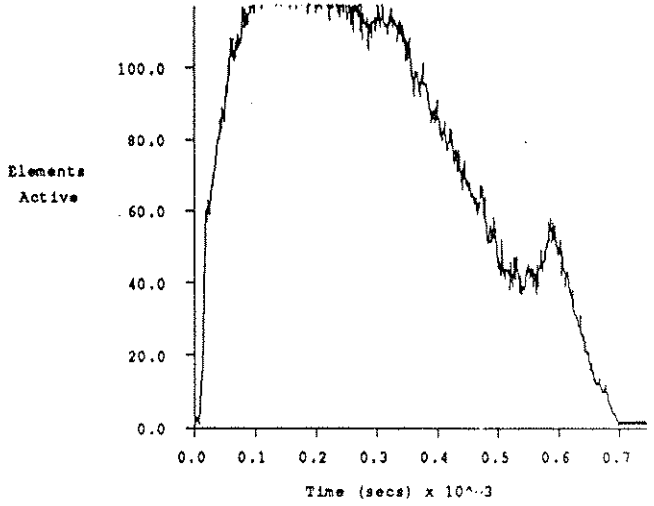

Unmatched and Transit Tokens 'pi_new_h10'
Unmatched
Transit
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Time (secs) x 10^-3

Loops

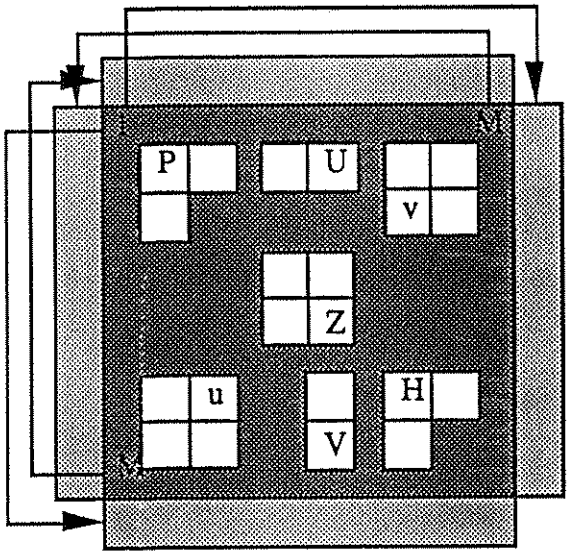

Unmatched
Transit
Tokens
36000
32000
28000
24000
20000
16000
12000
8000
4000
0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
Time (secs) x 10^-3

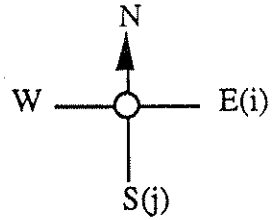

Elements
Active
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
Time (secs) x 10^-3


Transit
Tokens
18000
15000
12000
9000
6000
3000
0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
Time (secs) x 10^-3


Elements
Active
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
Time (secs) x 10^-3

Shallow a Model Weather Code



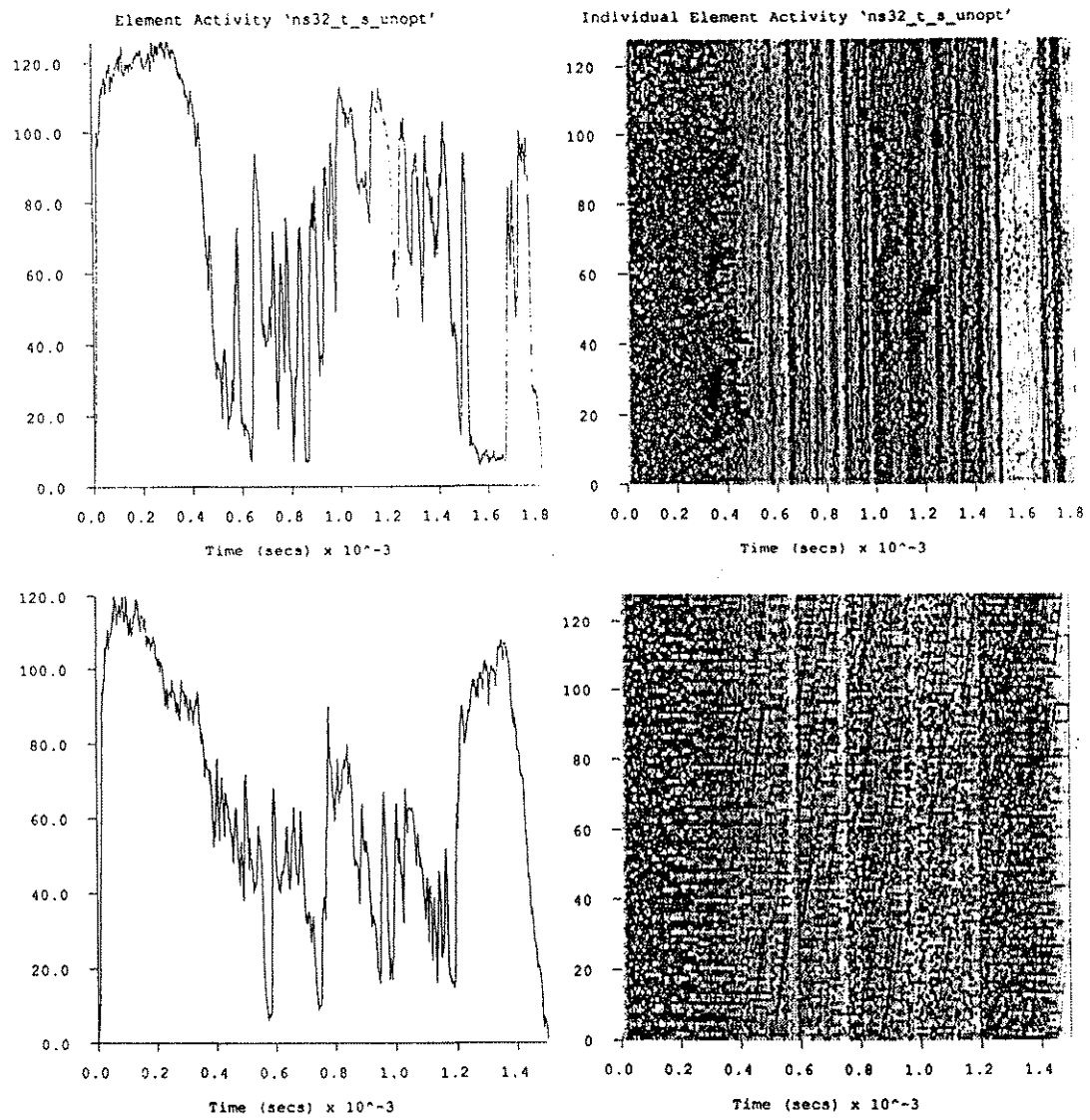


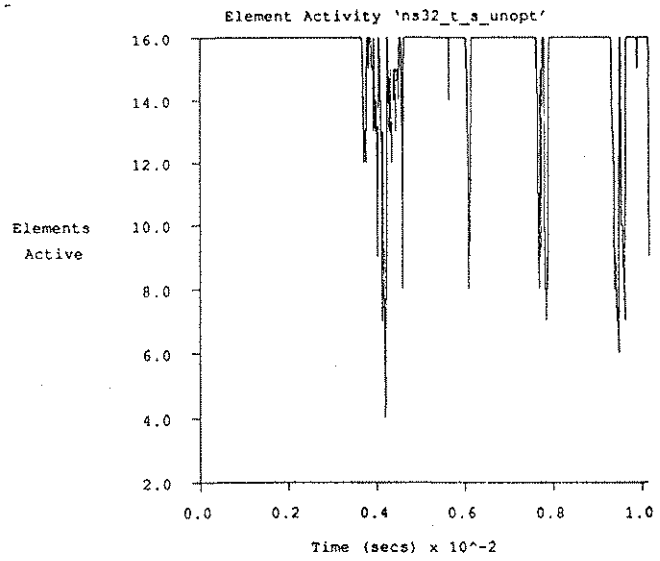
The "adjacency" figures indicate which cells are involved in the computation of P,u,v,U,V,Z and H. The boundaries are mapped as shown to make the mesh periodic.

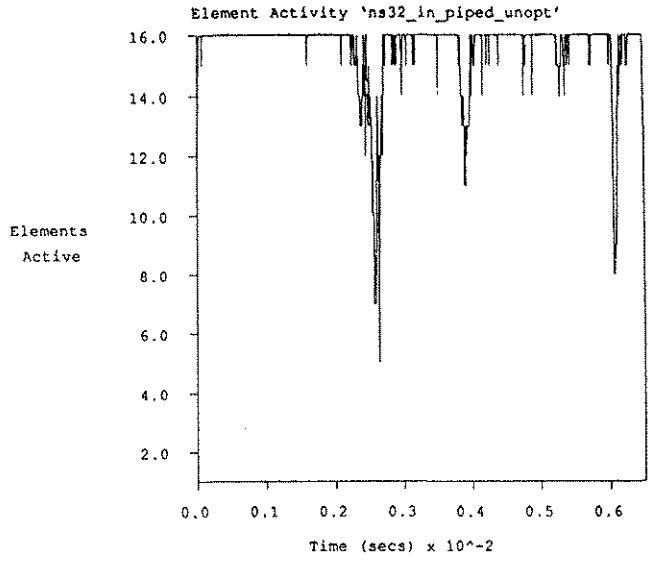
$$\delta u/\delta t - ZV + \delta H/\delta x = 0$$
$$\delta v/\delta t + ZU + \delta H/\delta y = 0$$
$$\delta P/\delta t + \delta U/\delta x + \delta V/\delta y = 0$$

Potential Function in Sisal

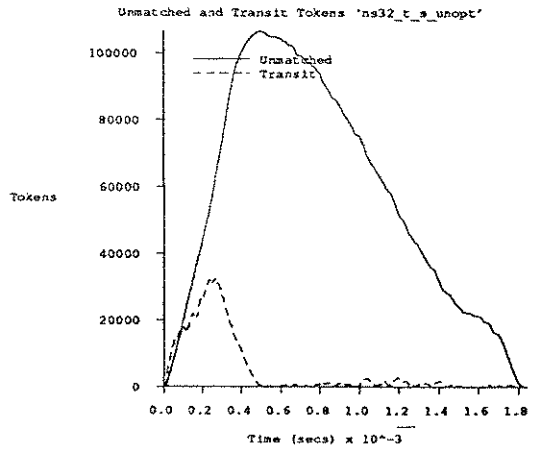
```
function  Potential(M: integer;P, U, V:GridVariables; delta:real
                returns GridVariables)

let
  fsdx:   real := 4.0 / delta;
  fsdy:   real := 4.0 / delta;
in
  for j in 0,M
  returns array of
    let jn := if j=0 then M else j-1 end if in
    for i in 0,M
    returns array of
      let iw := if i=0 then M else i-1 end if in
                        (fsdx*(V[j,i]-V[j,iw])-fsdy*(U[j,i]-U[jn,i]))/
                        (P[jn,iw]+P[jn,i]+P[j,iw]+P[j,i])
      end let
    end for
    end let
  end for
end let

end function
```

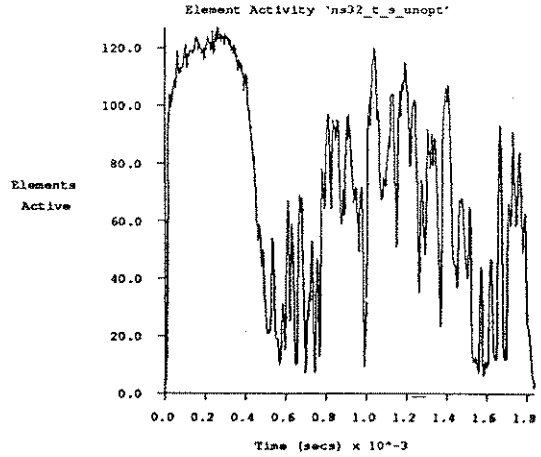

Element Activity 'ns32_t_s_unopt'
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3
Individual Element Activity 'ns32_t_s_unopt'
120
100
80
60
40
20
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
Time (secs) x 10^-3
120
100
80
60
40
20
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
Time (secs) x 10^-3

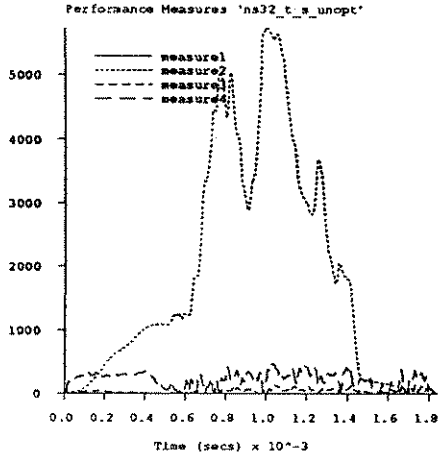


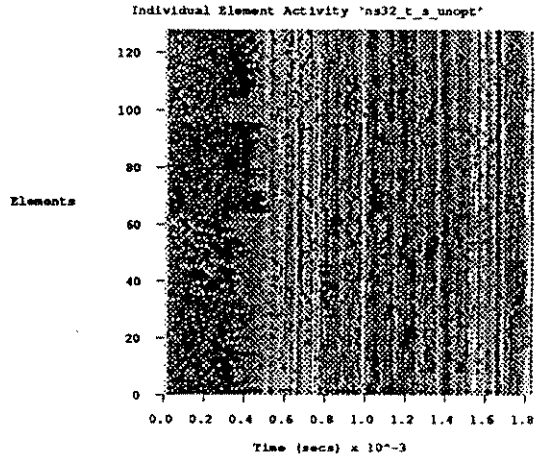


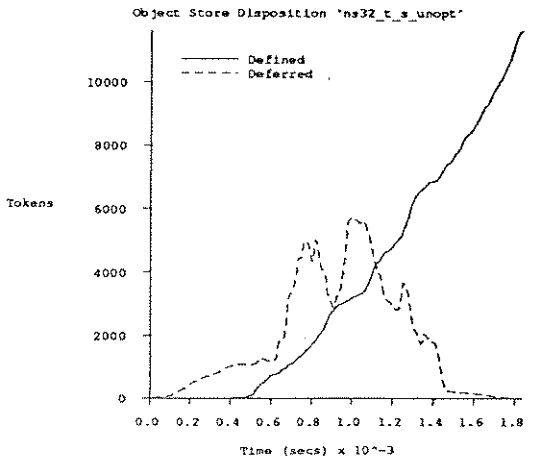
MMF	**6.31->4.8**
Instructions	**-24%**
Time	**-37% (-17% 128pe)**

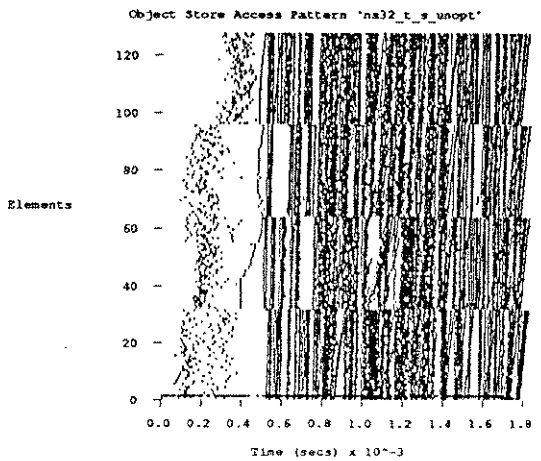

Unmatched and Transit Tokens 'ns32_t_s_unopt'
Unmatched
Transit
Tokens
100000
80000
60000
40000
20000
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

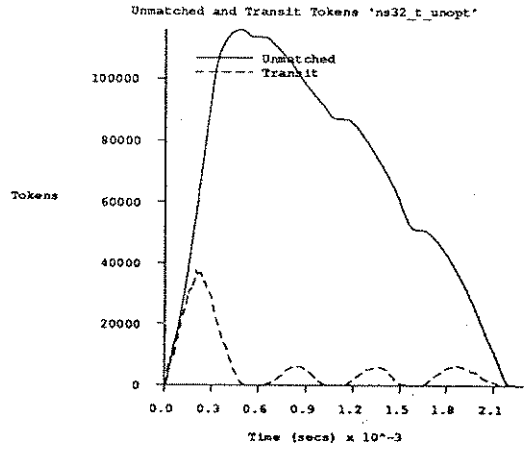

Element Activity 'ns32_t_s_unopt'
Elements
Active
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

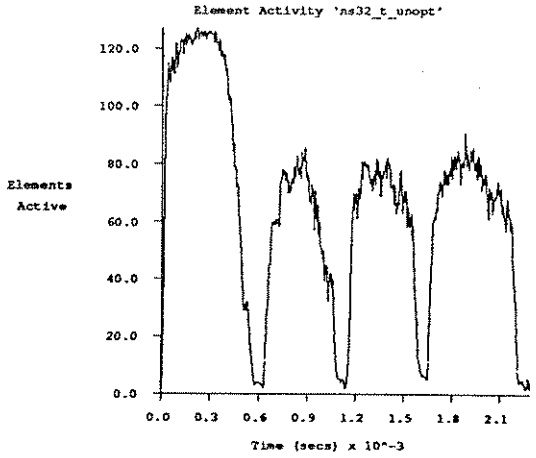

Performance Measures 'ns32_t_s_unopt'
measure1
measure2
measure3
measure4
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

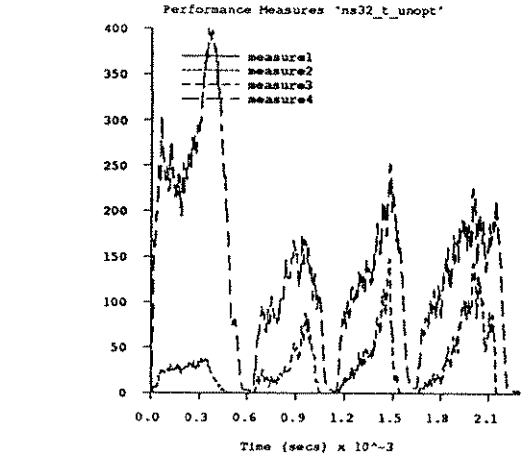

Individual Element Activity 'ns32_t_s_unopt'
Elements
120
100
80
60
40
20
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

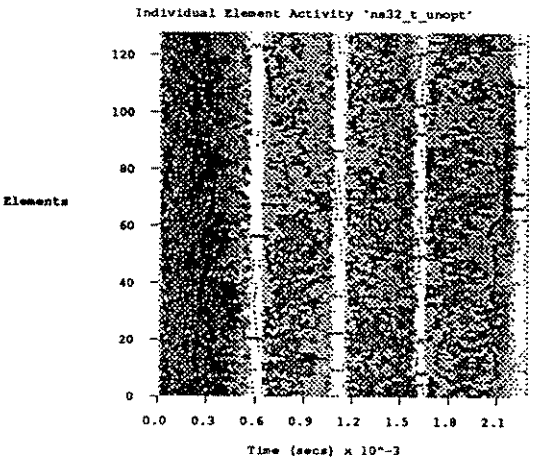

Object Store Disposition 'ns32_t_s_unopt'
Defined
Deferred
Tokens
10000
8000
6000
4000
2000
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

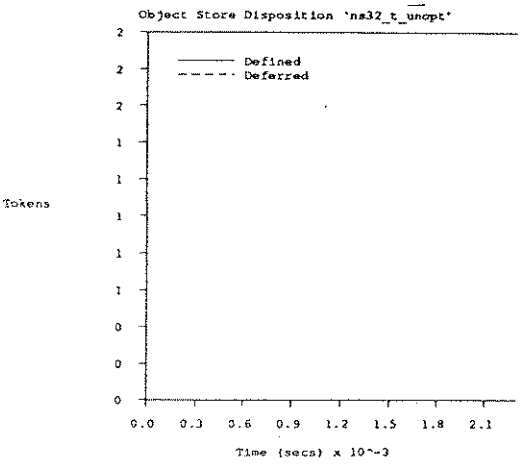

Object Store Access Pattern 'ns32_t_s_unopt'
Elements
120
100
80
60
40
20
0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8
Time (secs) x 10^-3

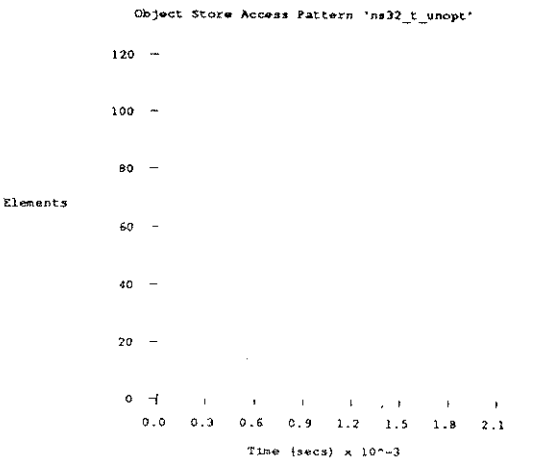

Unmatched and Transit Tokens 'ns32_t_unopt'
Unmatched
Transit
Tokens
100000
80000
60000
40000
20000
0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3


Element Activity 'ns32_t_unopt'
Elements
Active
120.0
100.0
80.0
60.0
40.0
20.0
0.0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3

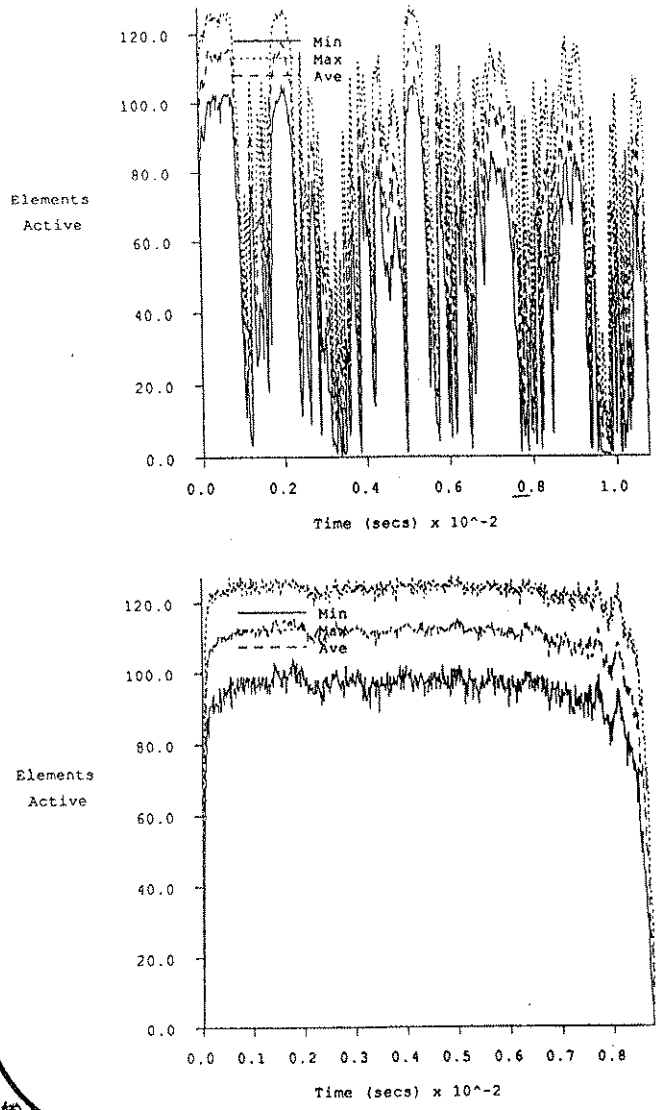

Performance Measures 'ns32_t_unopt'
measure1
measure2
measure3
measure4
400
350
300
250
200
150
100
50
0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3

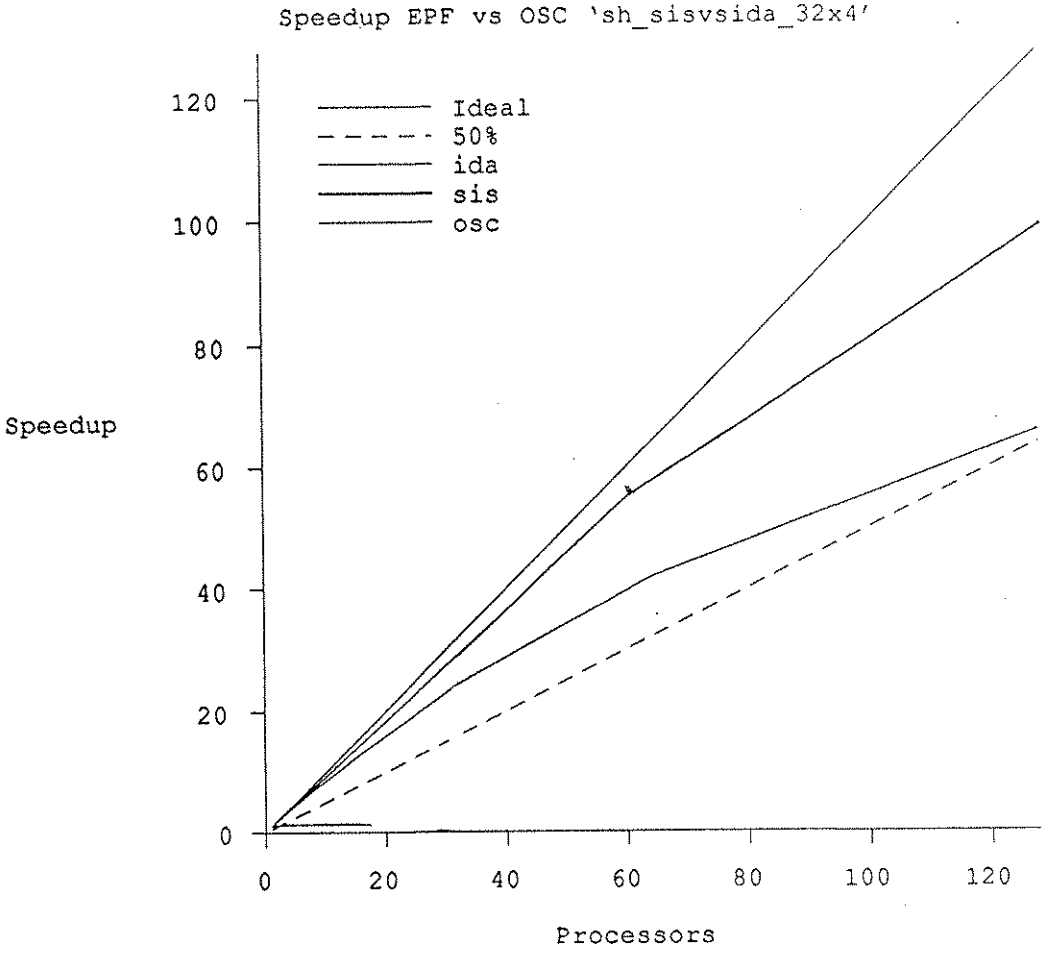

Individual Element Activity 'ns32_t_unopt'
Elements
120
100
80
60
40
20
0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3


Object Store Disposition 'ns32_t_unopt'
Defined
Deferred
Tokens
2
2
2
1
1
1
1
1
0
0
0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3


Object Store Access Pattern 'ns32_t_unopt'
Elements
120
100
80
60
40
20
0
0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1
Time (secs) x 10^-3

Sisal and Ida





Shallow 32x4 in Ida on 16 PEs fully pipelined!

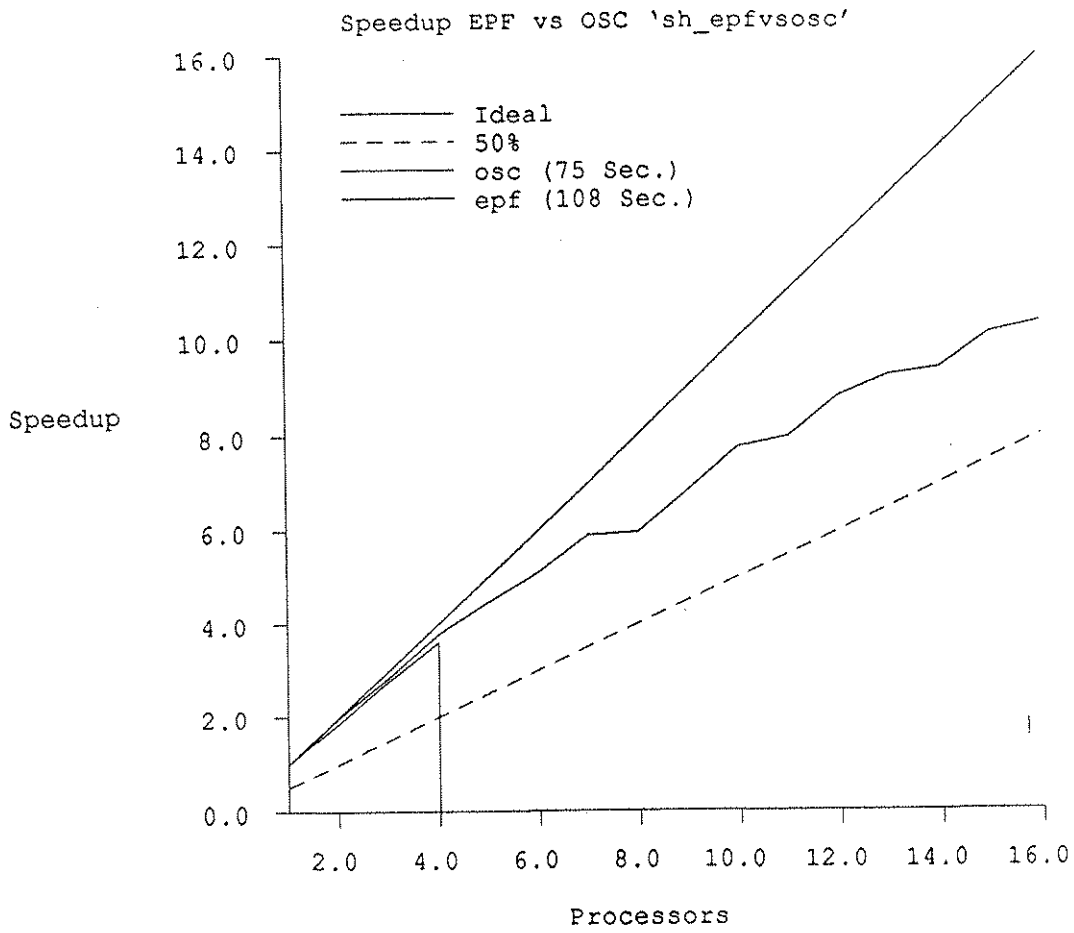
```
Dataflow Machine (CSIRAC II) Simulator/Emulator
Version 2.0 Released: Sun May 27 20:09:07 EST 1990
Author: G.K. Egan

                      File  <ins_e16.dfo>
              Time (Sec.) 0.055352
          First data time 0.001185
       Statistics interval 0.054167 (period for each sample 0.022222)

       Processing-elements  16
           Utilisation (%)  90.9 (idle  3.0 ~network contention 6.1)
                 MOps/Sec.  61.5 (3.8/PE)
              MFPOps/Sec.   16.4 (1.0/PE)
             MTokens/Sec.   98.6 (6.2/PE)

          Nodes fired (S1) 3333642 (Ops/FPOps 3.76)
        Monadic nodes (%)   39.8 (13.9 with literal data)
    Result tokens/firing    1.6
      Tokens transmitted 5340741 (~85453648 bytes)
      ALU to ALU Latency 10
          Peak IQ tokens 10661
          Peak DQ tokens 0
          Peak MS tokens 105365
          Peak OS tokens 42068 {42068 remaining in the OSs }
             OS accesses 271526 (read/write  4.5)
     Peak deferred reads 108400
```

Shallow 32x4 in Sisal on 16 PEs full unravelled

```
Dataflow Machine (CSIRAC II) Simulator/Emulator
Version 2.0 Released: Sun May 27 20:09:07 EST 1990
Author: G.K. Egan

                      File  <sns_e16.dfo>
             Time (Sec.) 0.035599
         First data time 0.001013
     Statistics interval 0.034586 (period for each sample 0.000042)

     Processing-elements  16
         Utilisation (%)  81.8 (idle 10.5 ~network contention 7.7)
               MOps/Sec.  48.6 (3.0/PE)
            MFPOps/Sec.    7.7 (0.5/PE)
           MTokens/Sec.  83.7 (5.2/PE)

        Nodes fired (S1) 1681875 (Ops/FPOps 6.31)
       Monadic nodes (%)  27.8 ( 4.3 with literal data)
    Result tokens/firing   1.7
      Tokens transmitted 2895430 (~53375472 bytes)
      ALU to ALU Latency 10
           Peak IQ tokens 42320
           Peak DQ tokens 0
           Peak MS tokens 335668
           Peak OS tokens 43298 {43298 remaining in the OSs }
              OS accesses 254582 (read/write  3.9)
     Peak deferred reads 83617
```

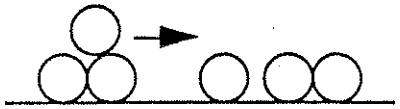
Shallow

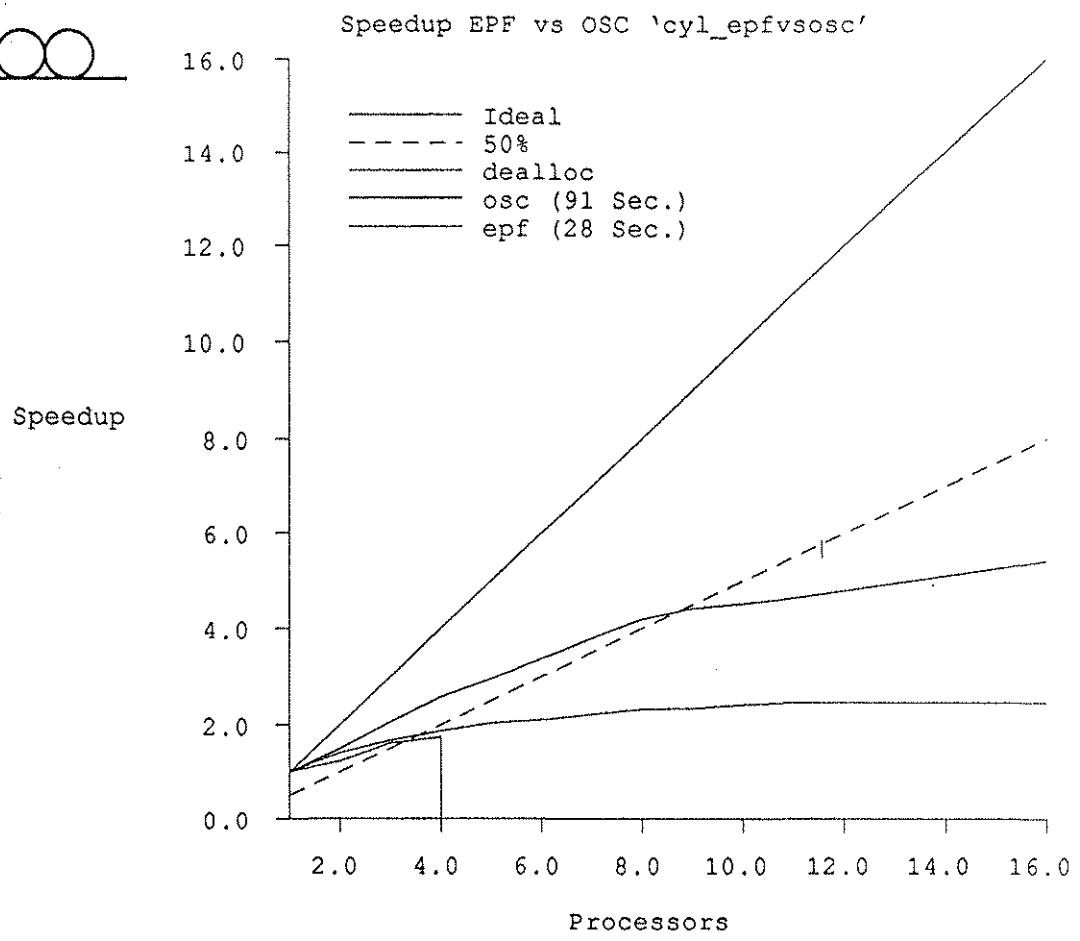
EPF and Sisal (Good News)



Cylinders

EPF and Sisal (Not So Good News)





Real Weather Codes (Pau Chang)

Primitive forms:

$$\zeta = \underline{k} \cdot \nabla \times \underline{V} = \nabla^2 \psi \qquad (1)$$

$$D = \nabla \cdot \underline{V} = \nabla^2 \chi \qquad (2)$$

$$\Phi = \Phi^* + \Phi' \qquad (3)$$

$$\frac{\delta(\nabla^2\psi)}{\delta t} = \frac{-1}{a\cos^2\phi}\left[\frac{\delta(U\nabla^2\psi)}{\delta\lambda} + \cos\phi\frac{\delta(V\nabla^2\psi)}{\delta\phi}\right] - 2\Omega\left(\sin\phi\,\nabla^2\chi + \frac{V}{a}\right) \qquad (4)$$

$$\frac{\delta(\nabla^2\chi)}{\delta t} = \frac{1}{a\cos^2\phi}\left[\frac{\delta(V\nabla^2\psi)}{\delta\lambda} - \cos\phi\frac{\delta(U\nabla^2\psi)}{\delta\phi}\right] + 2\Omega\left(\sin\phi\,\nabla^2\chi - \frac{U}{a}\right) - \nabla^2\left[\frac{U^2+V^2}{2\cos^2\phi} + \Phi'\right] \qquad (5)$$

$$\frac{\delta\Phi'}{\delta t} = \frac{-1}{a\cos^2\phi}\left[\frac{\delta U\Phi'}{\delta\lambda} + \cos\phi\frac{\delta V\Phi'}{\delta\phi}\right] - \Phi^* D \qquad (6)$$

$$U = \frac{-\cos\phi}{a}\frac{\delta\psi}{\delta\phi} + \frac{1}{a}\frac{\delta\chi}{\delta\lambda} \qquad (7)$$

$$V = \frac{1}{a}\frac{\delta\psi}{\delta\lambda} + \frac{\cos\phi}{a}\frac{\delta\chi}{\delta\phi} \qquad (8)$$

Discrete form of some equations:

$$\psi = a^2 \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J} \psi_r^m \, Y_r^m \qquad (9)$$

$$\Phi = a^2 \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J} \Phi_r^m \, Y_r^m \qquad (10)$$

$$U = a \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J+1} U_r^m \, Y_r^m \qquad (11)$$

$$V = a \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J+1} V_r^m \, Y_r^m \qquad (12)$$

where $Y_r^m = P_r^m \sin\phi \, e^{im\lambda}$

$P_r^m(\sin\phi) = alp_r^m$ = Normalised Legendre Polynomial

$$\int_{-\pi/2}^{\pi/2} P_r^m(\sin\phi)\, P_r^m(\sin\phi)\, \cos\phi \, d\phi = 1 \qquad (13)$$

and $$U_r^m = (r-1)\,\rho_r^m \psi_{r-1}^m - (r+2)\,\rho_{r+1}^m \psi_{r+1}^m + im\chi_r^m \qquad (14)$$

$$V_r^m = -(r-1)\,\rho_r^m \chi_{r-1}^m + (r+2)\,\rho_{r+1}^m \chi_{r+1}^m + im\psi_r^m \qquad (15)$$

$$\rho_r^m = \sqrt{\frac{(r^2-m^2)}{(4r^2-1)}}$$

Direct Transliteration

. Full model with 1 iteration of timeloop:
SISAL: 773 seconds - 11 times slower than the FORTRAN (~70 seconds)
loss of parallelism due to excessive copying (36% of computation time)

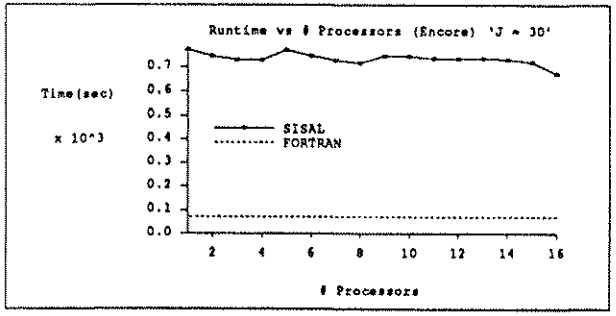


Figure 5a: Multiple processor run times for FORTRAN and sequential SISAL

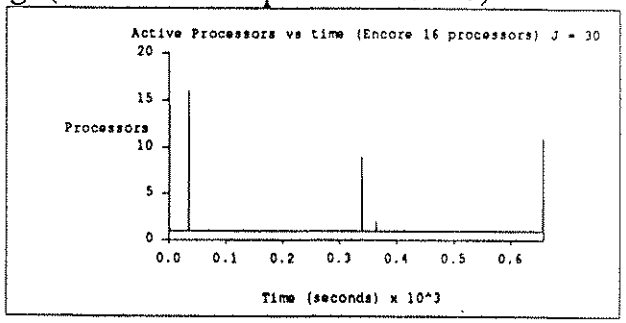


Figure 5b: Parallelism profile of the directly transliterated SISAL version

Original Formulation

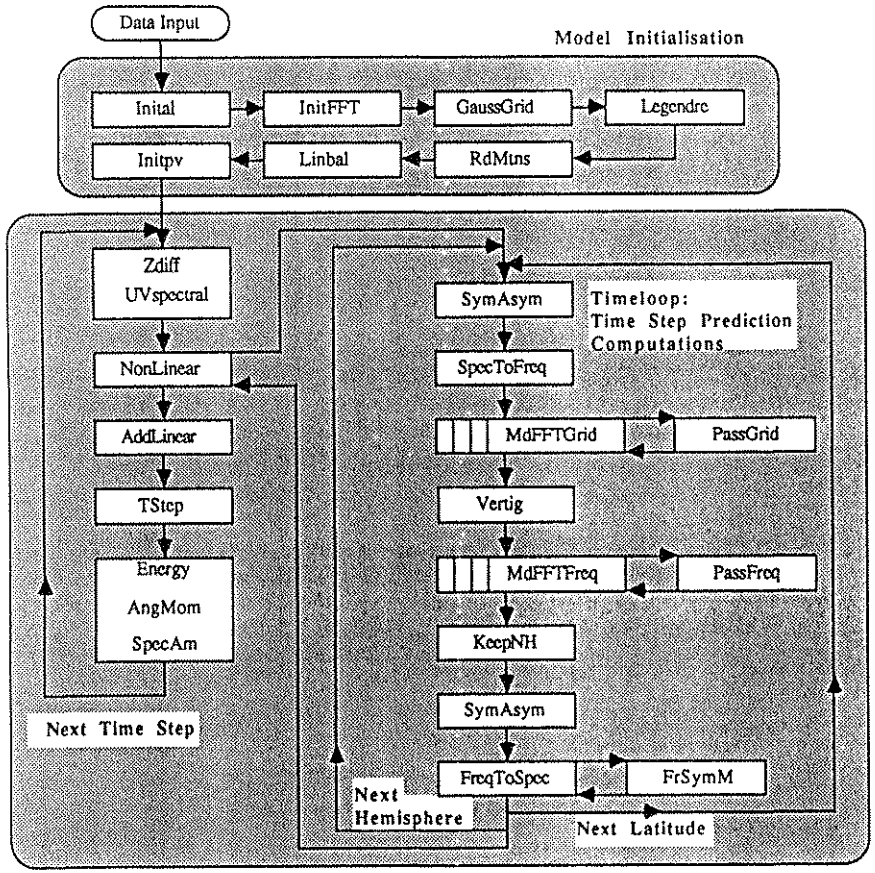


Figure 3: Flow chart for the FORTRAN and sequential SISAL implementations

Parallel Formulation

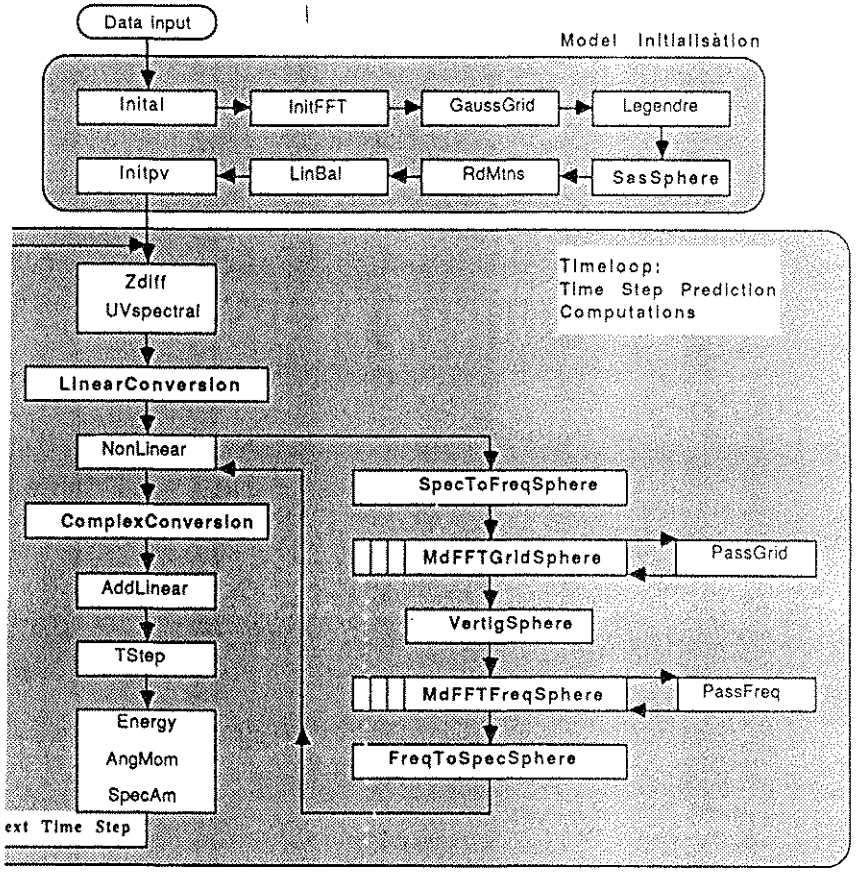


Figure 7: Flow chart for parallel SISAL implementation

Parallel Formulation from Model

FOR **both** hemispheres **and all** latitudes
FOR all longitude points *or elements in a spectral field*
compute the new values of the field

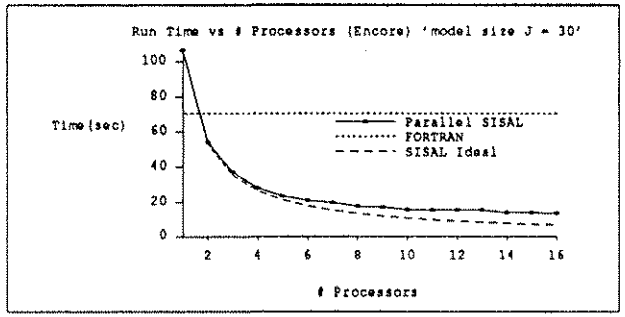


Figure 8a: Execution time profile of the new implementation

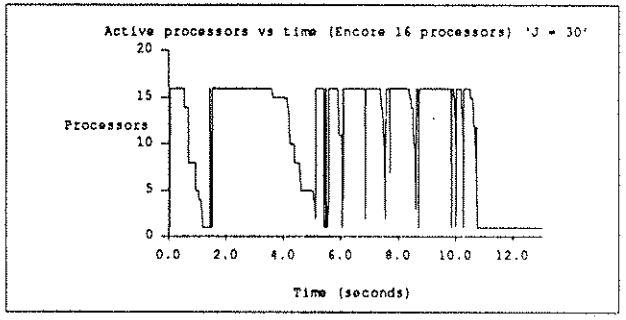


Figure 8b: Concurrency profile for the new implementation

Time Step

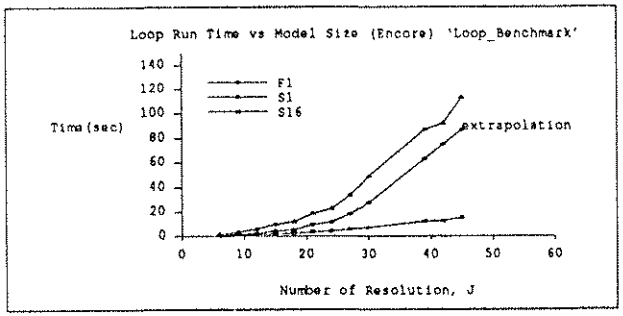


Figure 9: The execution time of the FORTRAN and SISAL implementations as a function of model size.

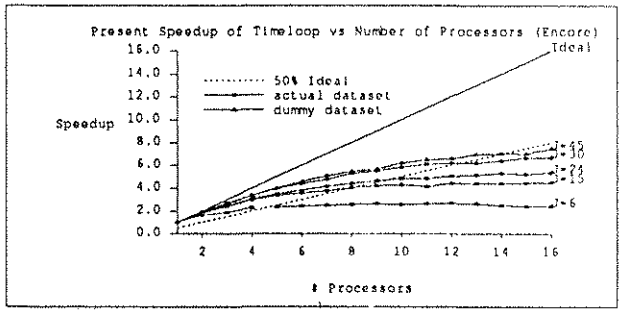


Figure 11: The speedup profile of the timeloop for the present implementation

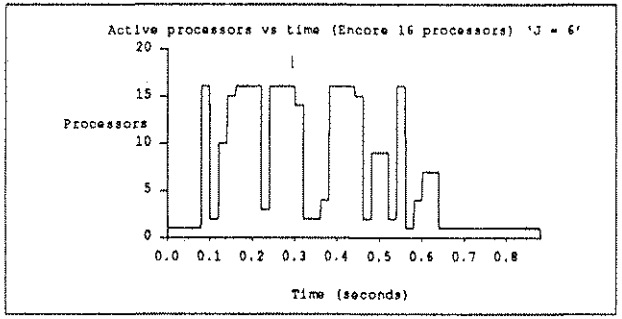


Figure 12a: Concurrency profile of timeloop for J = 6 (small model size)

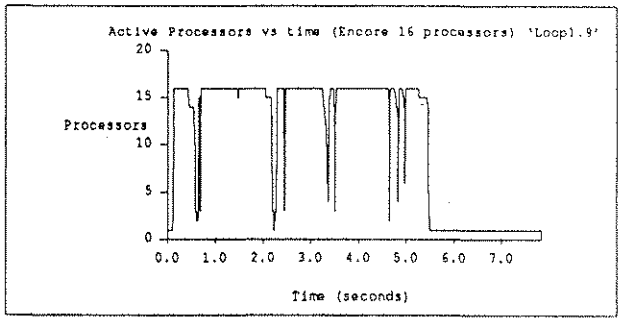


Figure 12b: Concurrency profile of timeloop for J = 30 (large model size)

Time Step (Dealloc Fixed!)

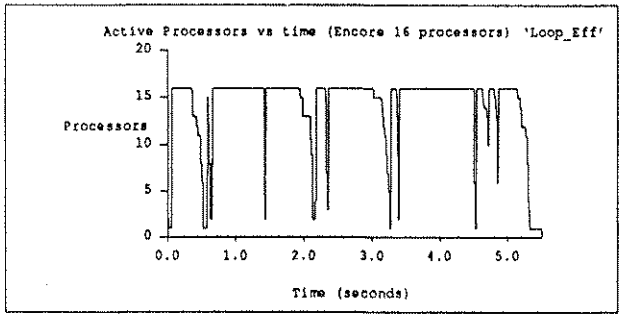


Figure 13a: The achievable concurrency (J = 30) of the timeloop with an efficient memory deallocation scheme.

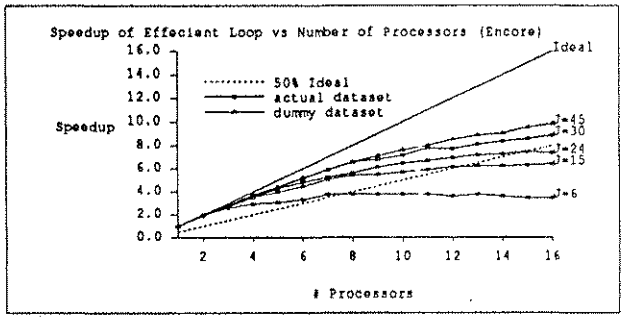


Figure 13b: The would be speedup of the timeloop with an efficient memory deallocation scheme.

Futures

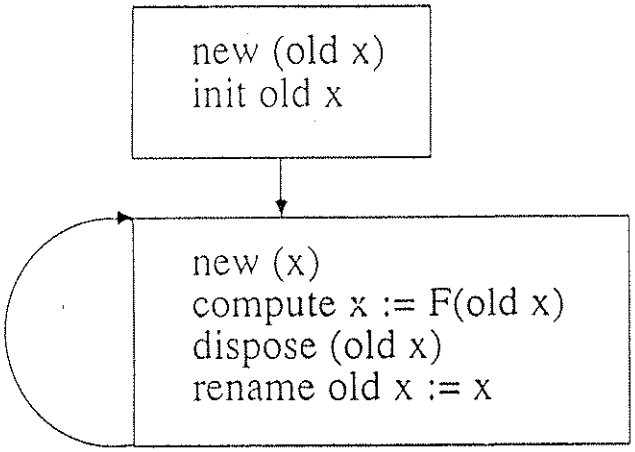
-CSIRAC II
- SCSI attached processor for other groups
- IF2
- vector ops with Sisal 2.0
- cross benchmarking Pascal/Ida/Sisal2/FORTRAN
- Australian Support for CSIRAC II (Simulators,IF1,i2)

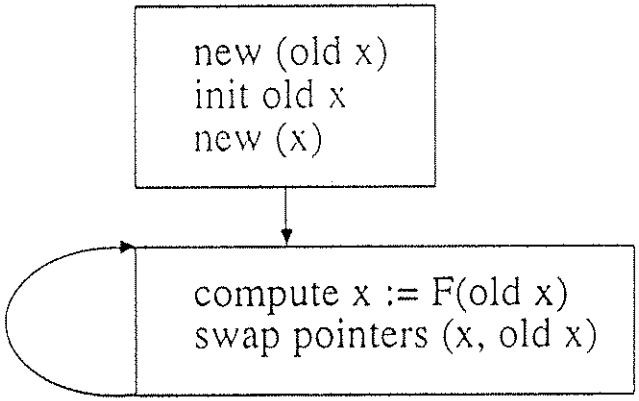
- port of Sisal to IBM RS6000 clusters (beta optical link site)

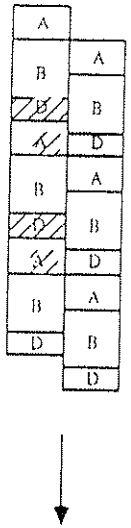
-weather codes (Sisal/FORTRAN)
- next generation grid code with full physics and
- live data due to go online at Australian Bureau
- of Meterology in 12 months

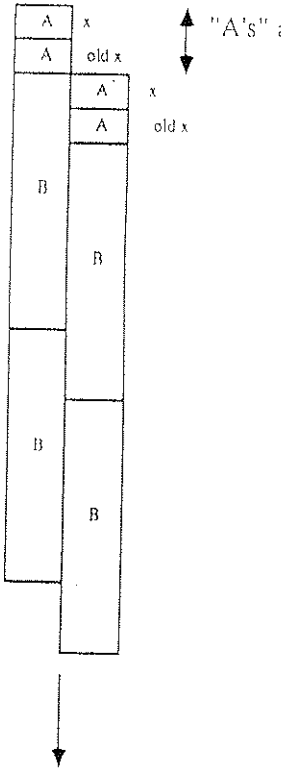
-geomechanics (Sisal/FORTRAN)
- block modelling of subsidence in coalmines (and Earthquakes!)
- with CSIRO Division of Geomechanics

-.....









Present Scheme

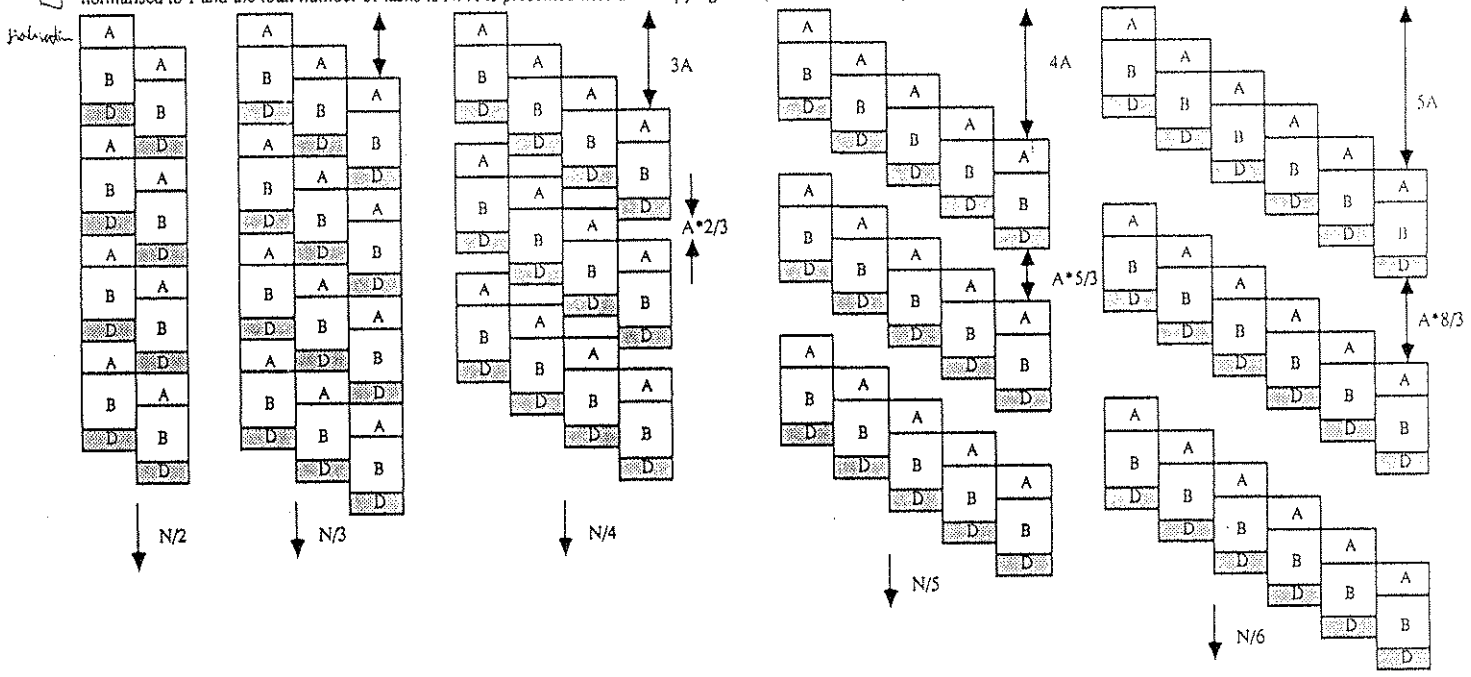
A Better Scheme

Memory Allocation and Deallocation Scheme of OSC

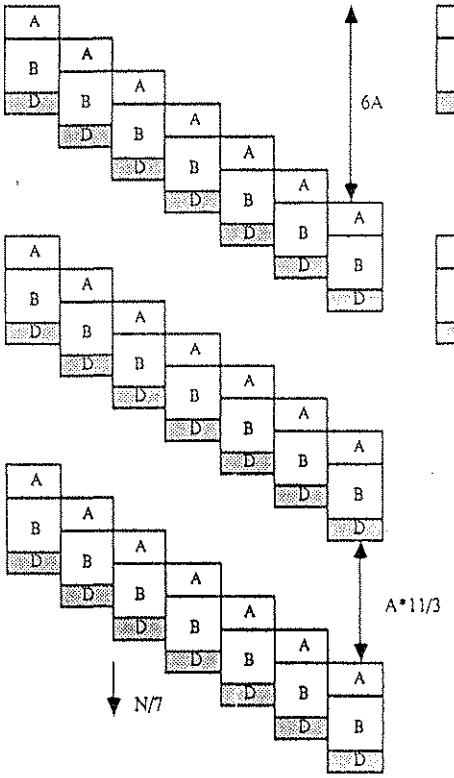
The Analysis of the Effect of Memory Allocation and Deallocation on the Parallelism of a Parallel Code

A - Memory Allocation
B - Slice Body
D - Memory Deallocation

It is expected that for some data structures, the allocation and deallocation operations at runtime overlap each other. The best case is when they run concurrently. The worst case is when they are mutually exclusive and have to run serial to one another due to access to the same data structure. The former is simpler than thelatter to analyse and the analysis is shown below. The execution time of each iteration is normalised to 1 and the total number of tasks is N. A is presented here as occupying 30% (normalised to 0.3) of each task. Each A provides a unique solution.

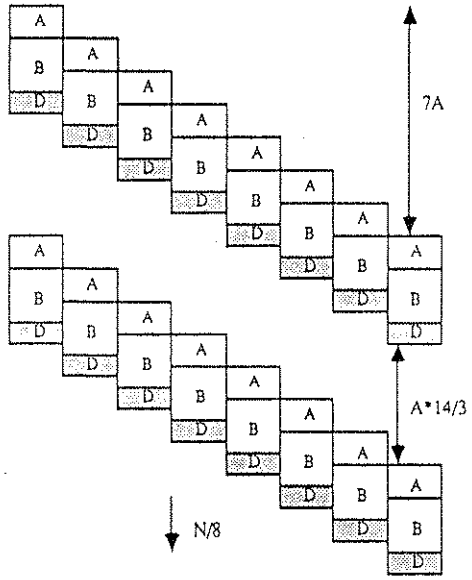


N/2 + A	N/3 + 2A	N/4 + 3A + (N/4-1)A*2/3	N/5 + 4A + (N/5-1)A*5/3	N/6 + 5A + (N/6-1)A*8/3
N=512				
SU=1.998	SU=2.989	SU=3.318	SU=3.318	SU=3.318
t=256.3	t=171.267	t=154.3		



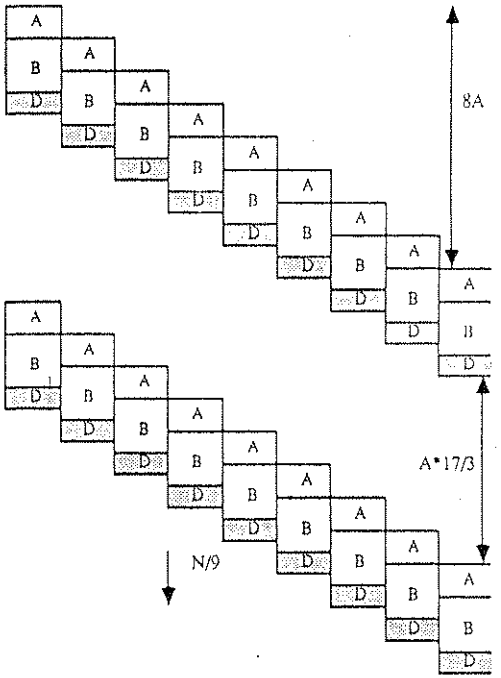
N/7 + 6A + (N/7-1)A*11/3

SU=3.318



N/8 + 7A + (N/8-1)A*14/3

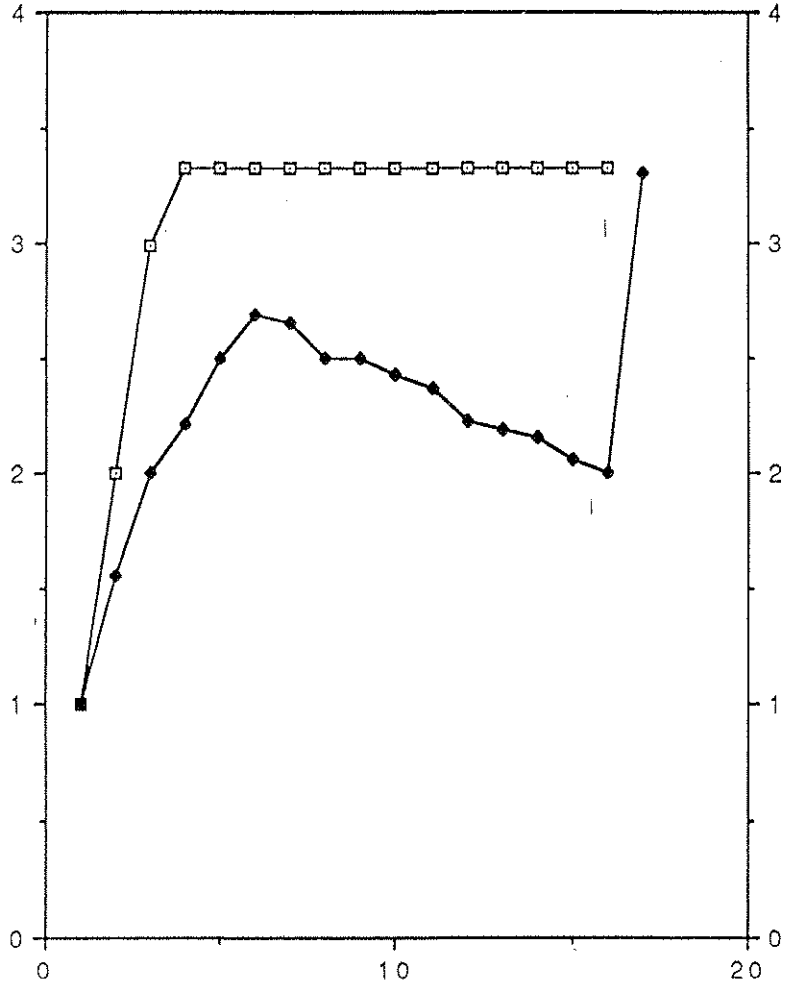
SU=3.318



N/9 + 8A + (N/9-1)A*17/3

SU=3.318

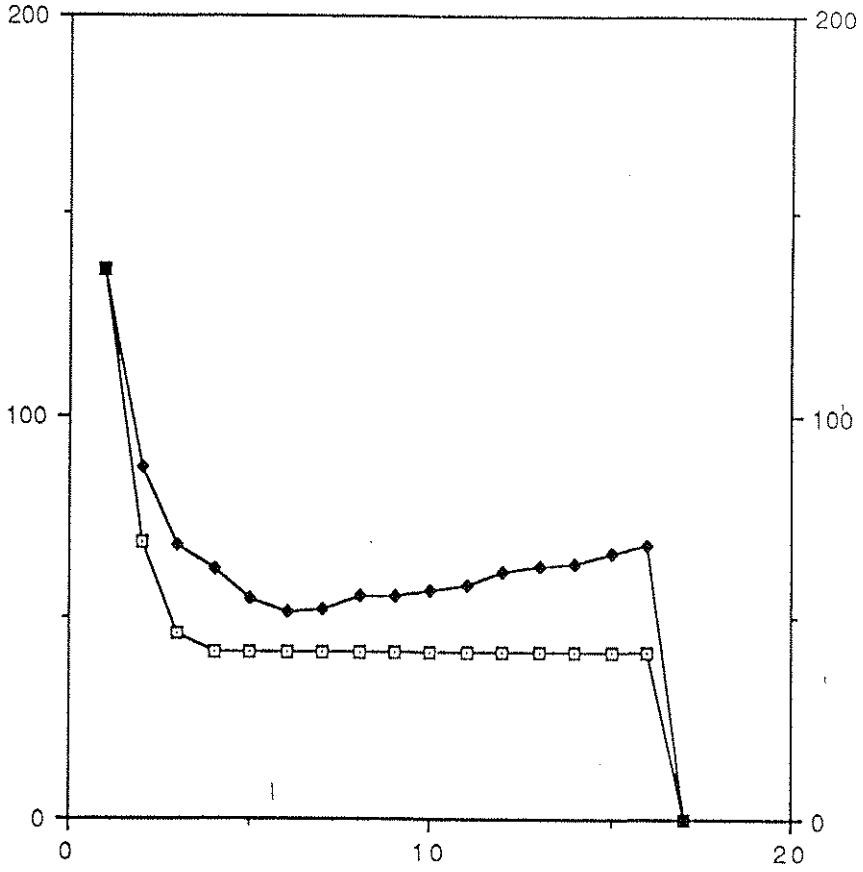
Data from "AllocDealloc"



Processors

512.ExT analy

Data from "AllocDealloc"



Processors