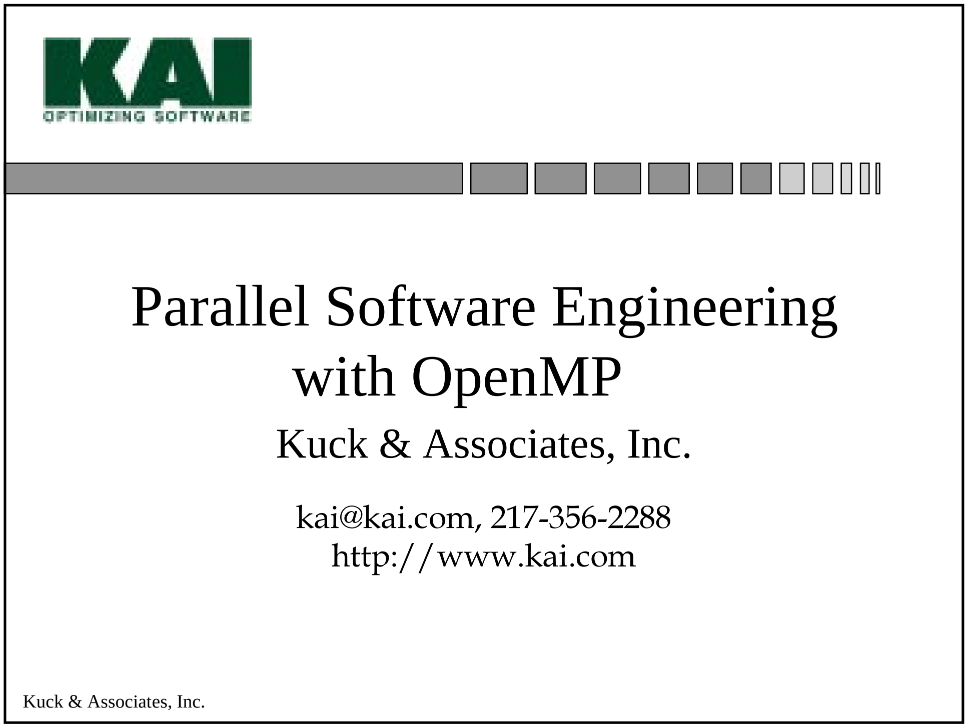
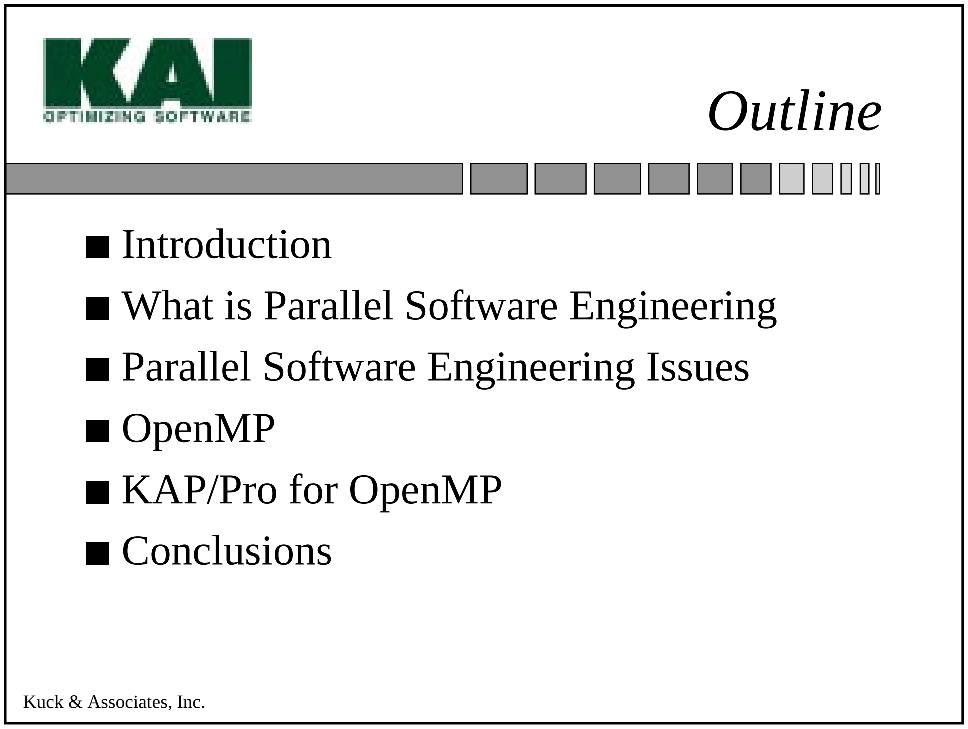


Parallel Software Engineering with OpenMP

Kuck & Associates, Inc.

kai@kai.com, 217-356-2288
http://www.kai.com

Kuck & Associates, Inc.



Outline

- Introduction
- What is Parallel Software Engineering
- Parallel Software Engineering Issues
- OpenMP
- KAP/Pro for OpenMP
- Conclusions

Kuck & Associates, Inc.

Why Parallel Software Engineering

We seek the following benefits --

- Performance
- Productivity
- Quality
- Standards

Kuck & Associates, Inc.

What is Parallel Software Engineering

- Steps in Parallelizing an Application

–**A**nalyze	Find the Parallel loop. Make sure it's the right one.
–**R**e**S**tructure	Make the necessary modifications: Classify variables, Add synchronization.
–**T**est	Verify that the program basically works in parallel mode.
–**I**mprove	Do the tuning necessary to get peak performance.
–**Q/A**	Verify that the parallel application is as robust as serial application.

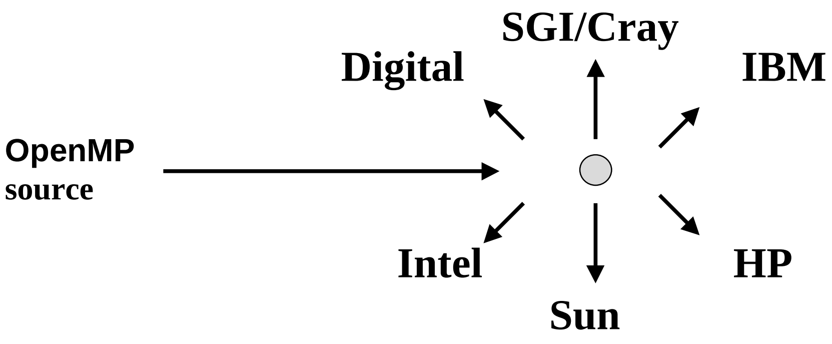
Kuck & Associates, Inc.

Digression -- A Bit of History

- Minisupercomputers legacy
  - Sequent, Alliant pioneered in 2nd half of 80’s
- PCF/X3H5 standardization effort
  - Cray, Digital, IBM, SGI developed consensus
- Parallel model used by many companies
- Advances in Shared Memory Multiprocessors is causing growing usage

The Trend Towards *Portable* Parallel Processing

- Portability between systems.
- With a common set of directives, **OpenMP**



OpenMP source

SGI/Cray

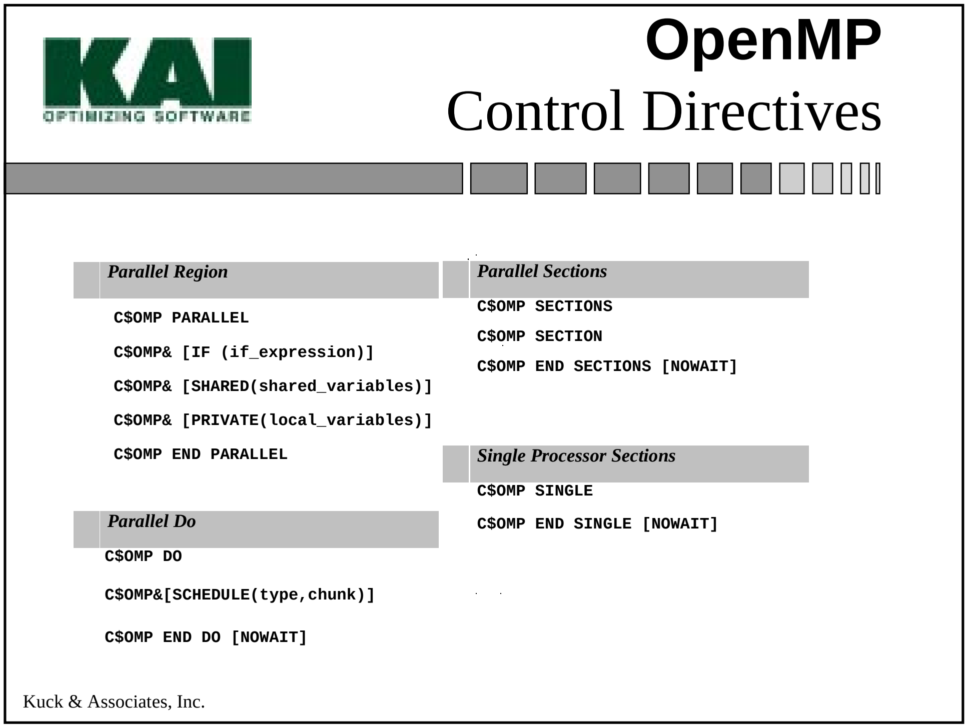
Digital

IBM

Intel

HP

Sun



OpenMP
Control Directives

Parallel Region

```
C$OMP PARALLEL
C$OMP& [IF (if_expression)]
C$OMP& [SHARED(shared_variables)]
C$OMP& [PRIVATE(local_variables)]
C$OMP END PARALLEL
```

Parallel Do

```
C$OMP DO
C$OMP&[SCHEDULE(type,chunk)]
C$OMP END DO [NOWAIT]
```

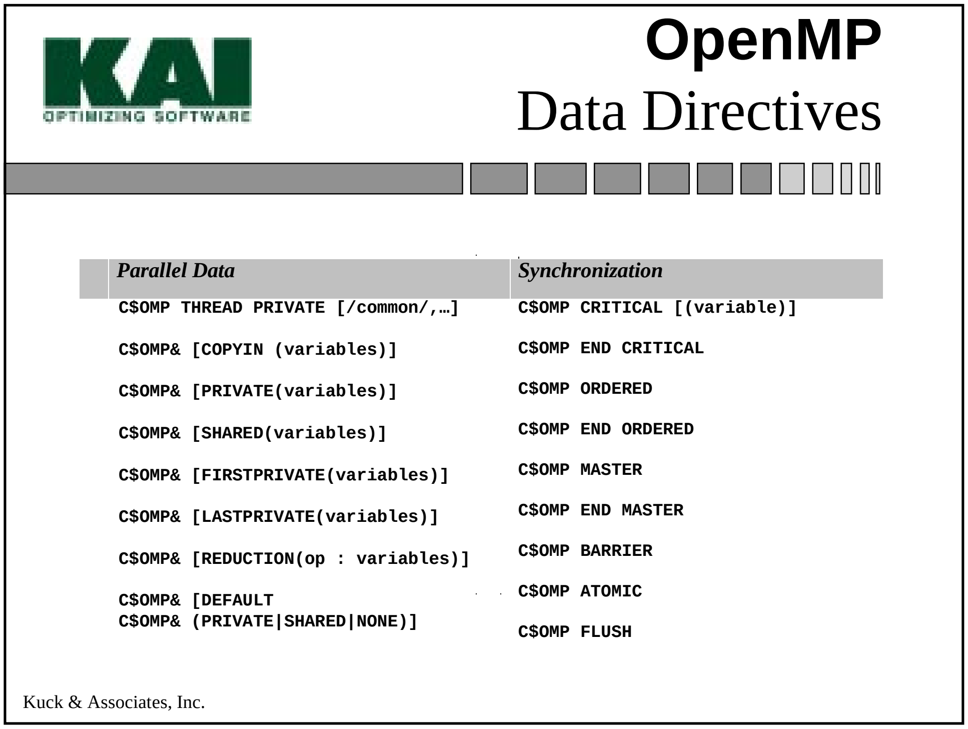
Parallel Sections

```
C$OMP SECTIONS
C$OMP SECTION
C$OMP END SECTIONS [NOWAIT]
```

Single Processor Sections

```
C$OMP SINGLE
C$OMP END SINGLE [NOWAIT]
```

Kuck & Associates, Inc.



OpenMP
Data Directives

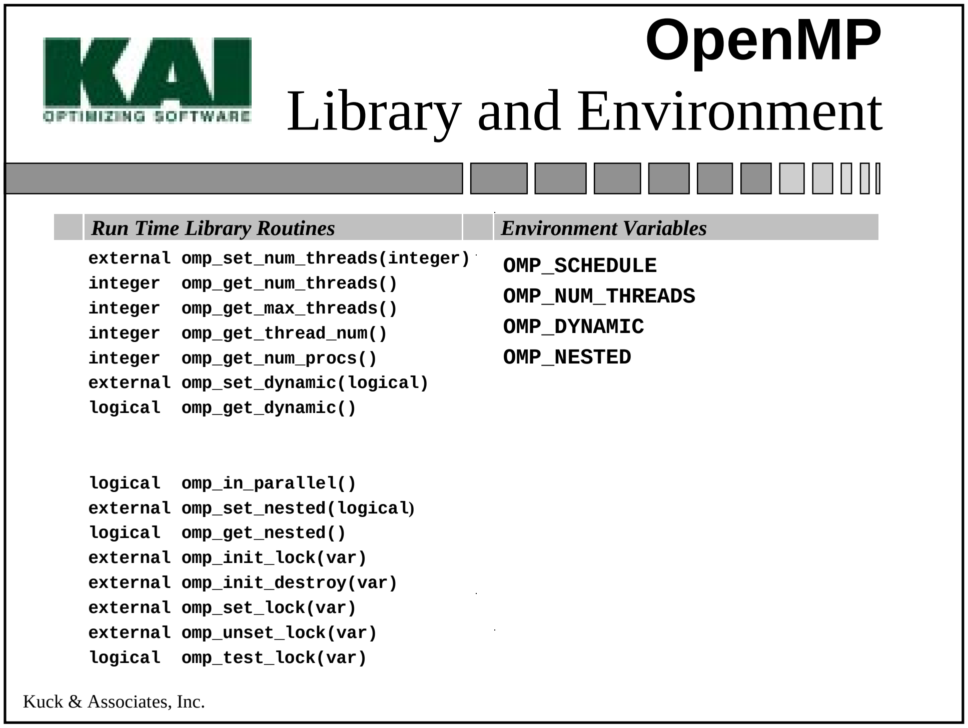
Parallel Data

```
C$OMP THREAD PRIVATE [/common/,…]
C$OMP& [COPYIN (variables)]
C$OMP& [PRIVATE(variables)]
C$OMP& [SHARED(variables)]
C$OMP& [FIRSTPRIVATE(variables)]
C$OMP& [LASTPRIVATE(variables)]
C$OMP& [REDUCTION(op : variables)]
C$OMP& [DEFAULT
C$OMP& (PRIVATE|SHARED|NONE)]
```

Synchronization

```
C$OMP CRITICAL [(variable)]
C$OMP END CRITICAL
C$OMP ORDERED
C$OMP END ORDERED
C$OMP MASTER
C$OMP END MASTER
C$OMP BARRIER
C$OMP ATOMIC
C$OMP FLUSH
```

Kuck & Associates, Inc.



OpenMP
Library and Environment

Run Time Library Routines

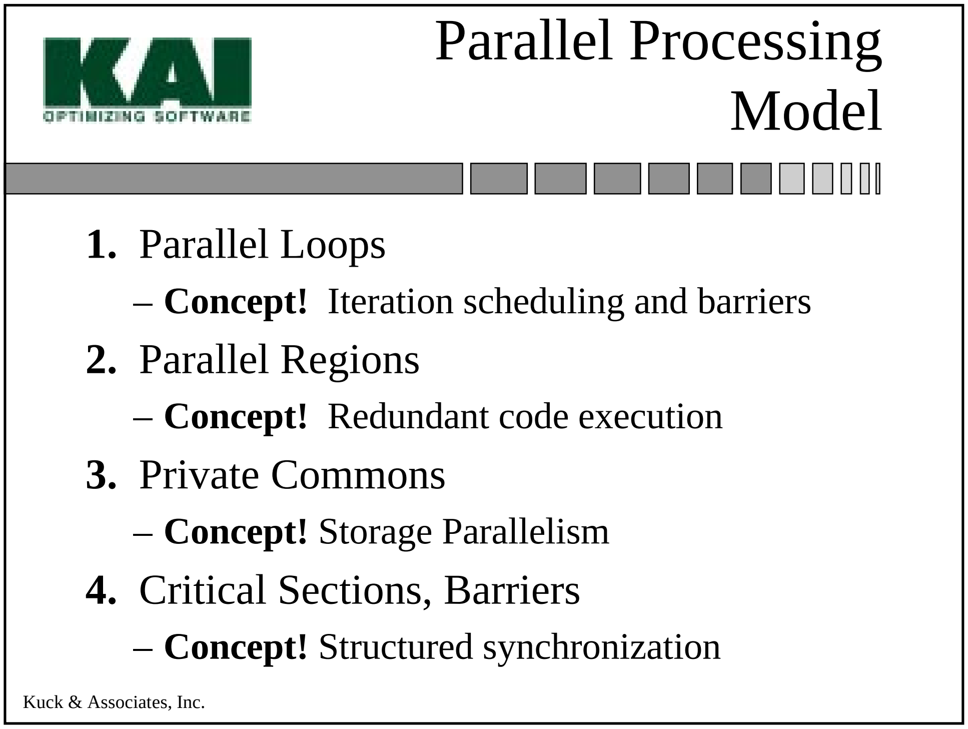
```
external omp_set_num_threads(integer)
integer  omp_get_num_threads()
integer  omp_get_max_threads()
integer  omp_get_thread_num()
integer  omp_get_num_procs()
external omp_set_dynamic(logical)
logical  omp_get_dynamic()


logical  omp_in_parallel()
external omp_set_nested(logical)
logical  omp_get_nested()
external omp_init_lock(var)
external omp_init_destroy(var)
external omp_set_lock(var)
external omp_unset_lock(var)
logical  omp_test_lock(var)
```

Environment Variables

```
OMP_SCHEDULE
OMP_NUM_THREADS
OMP_DYNAMIC
OMP_NESTED
```

Kuck & Associates, Inc.



Parallel Processing Model

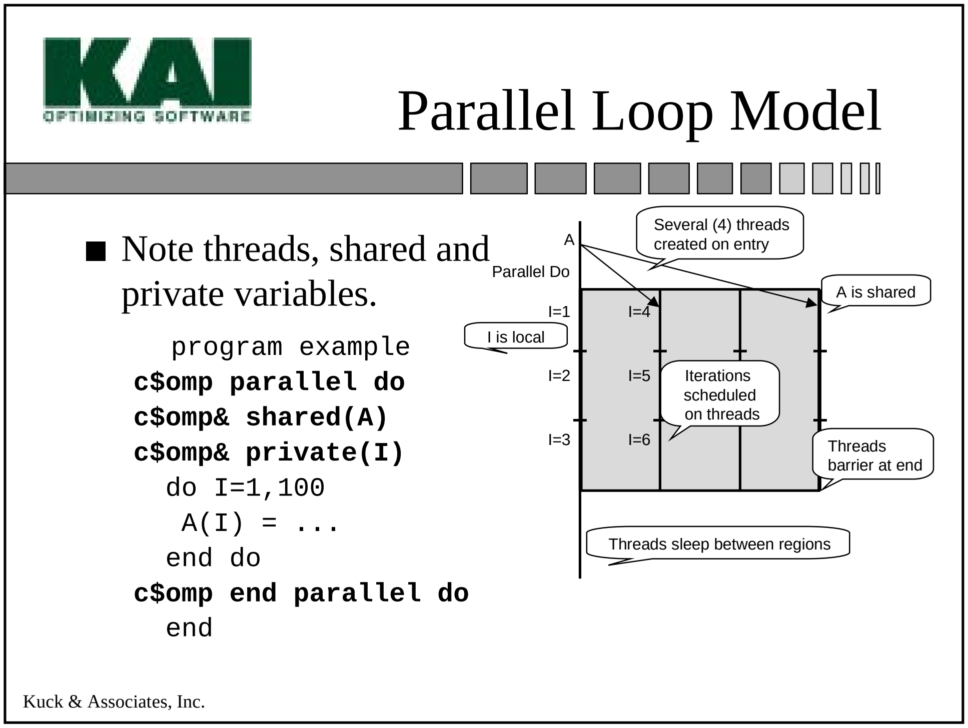
1. Parallel Loops
- **Concept!** Iteration scheduling and barriers

2. Parallel Regions
- **Concept!** Redundant code execution

3. Private Commons
- **Concept!** Storage Parallelism

4. Critical Sections, Barriers
- **Concept!** Structured synchronization

Kuck & Associates, Inc.



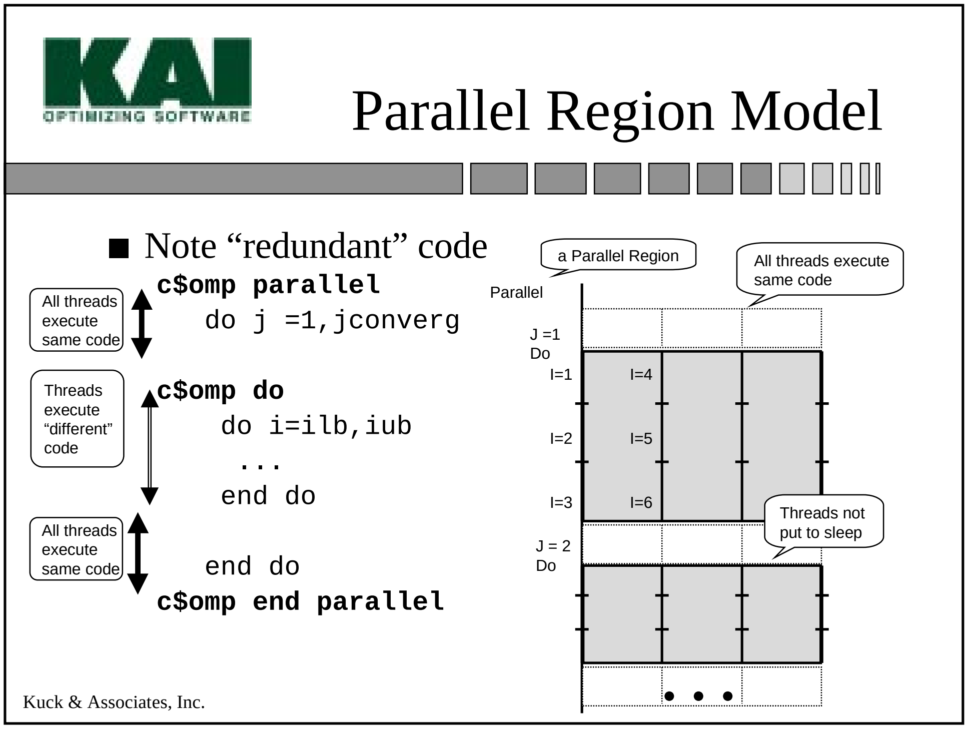
Parallel Loop Model

- Note threads, shared and private variables.

```
  program example
c$omp parallel do
c$omp& shared(A)
c$omp& private(I)
  do I=1,100
   A(I) = ...
  end do
c$omp end parallel do
  end
```

A
Parallel Do
Several (4) threads created on entry
A is shared
I is local
I=1
I=4
I=2
I=5
Iterations scheduled on threads
I=3
I=6
Threads barrier at end
Threads sleep between regions

Kuck & Associates, Inc.



Parallel Region Model

- Note “redundant” code

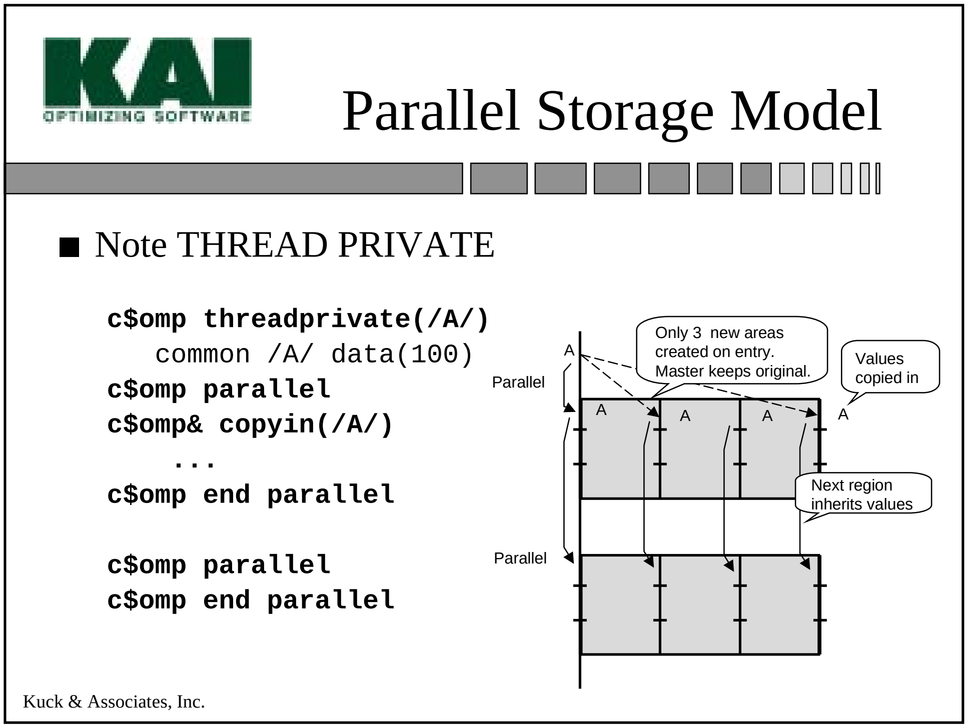
```
c$omp parallel
   do j =1,jconverg

c$omp do
    do i=ilb,iub
     ...
    end do


   end do
c$omp end parallel
```

All threads execute same code
Threads execute “different” code
All threads execute same code
a Parallel Region
All threads execute same code
Parallel
J =1 Do
I=1
I=4
I=2
I=5
I=3
I=6
Threads not put to sleep
J = 2 Do
• • •

Kuck & Associates, Inc.



Parallel Storage Model

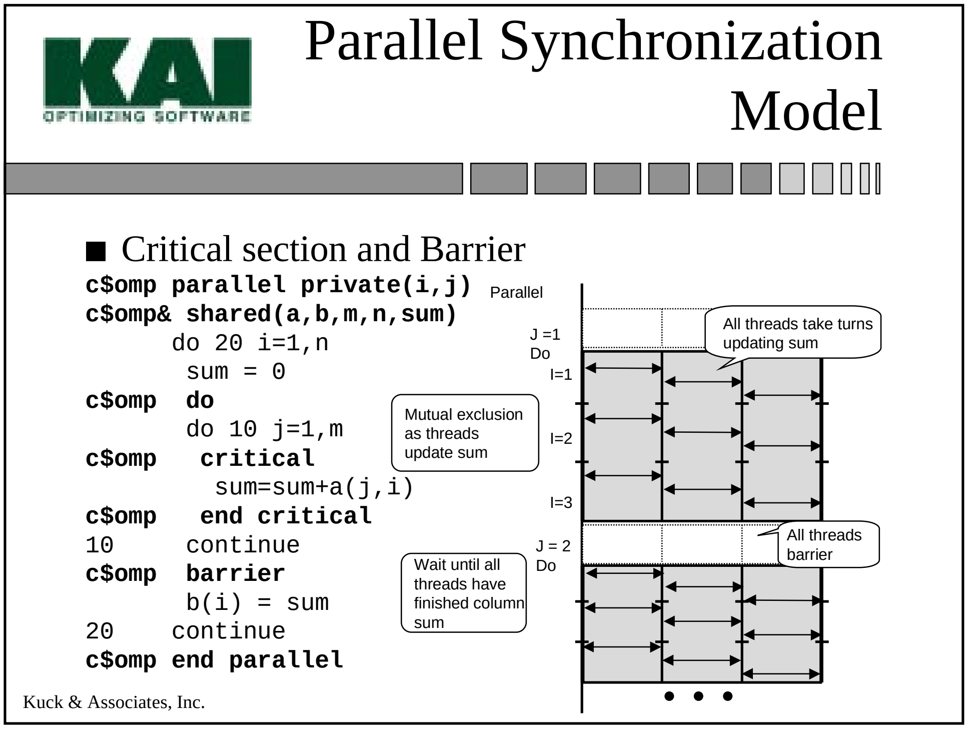
- Note THREAD PRIVATE

```
c$omp threadprivate(/A/)
         common /A/ data(100)
c$omp parallel
c$omp& copyin(/A/)
          ...
c$omp end parallel

c$omp parallel
c$omp end parallel
```

A
Parallel
Only 3 new areas created on entry. Master keeps original.
Values copied in
A
A
A
A
Next region inherits values
Parallel

Kuck & Associates, Inc.



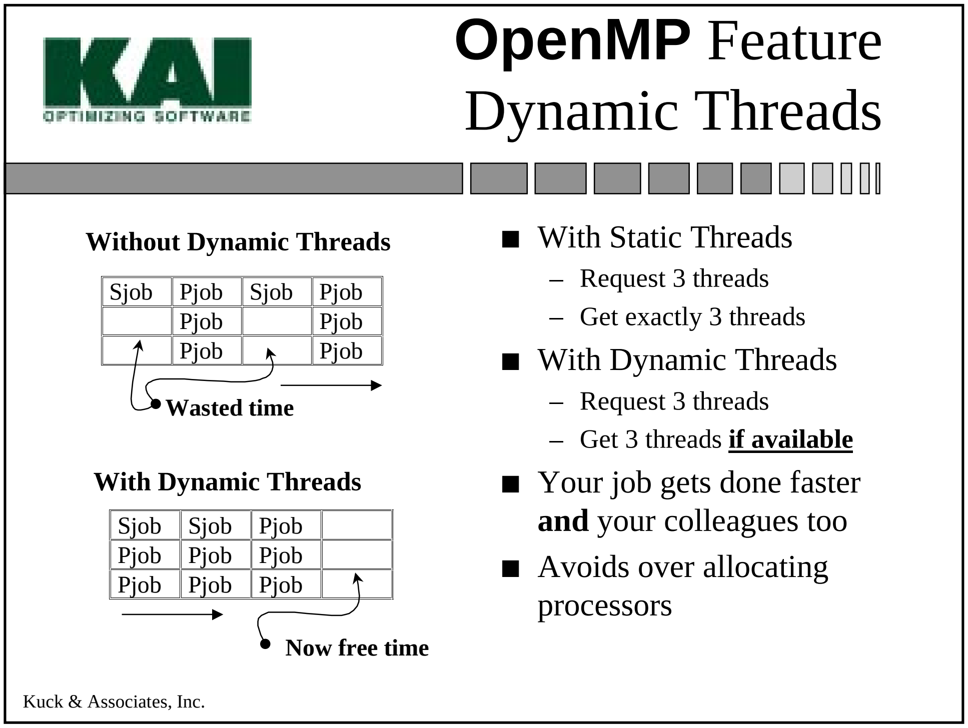
Parallel Synchronization Model

- Critical section and Barrier

```
c$omp parallel private(i,j)
c$omp& shared(a,b,m,n,sum)
            do 20 i=1,n
              sum = 0
c$omp  do
              do 10 j=1,m
c$omp    critical
                sum=sum+a(j,i)
c$omp    end critical
10          continue
c$omp  barrier
              b(i) = sum
20        continue
c$omp end parallel
```

Parallel
J =1
Do
I=1
I=2
I=3
All threads take turns updating sum
Mutual exclusion as threads update sum
J = 2
Do
All threads barrier
Wait until all threads have finished column sum
• • •

Kuck & Associates, Inc.



OpenMP Feature Dynamic Threads

Without Dynamic Threads

Sjob	Pjob	Sjob	Pjob
	Pjob		Pjob
	Pjob		Pjob

Wasted time

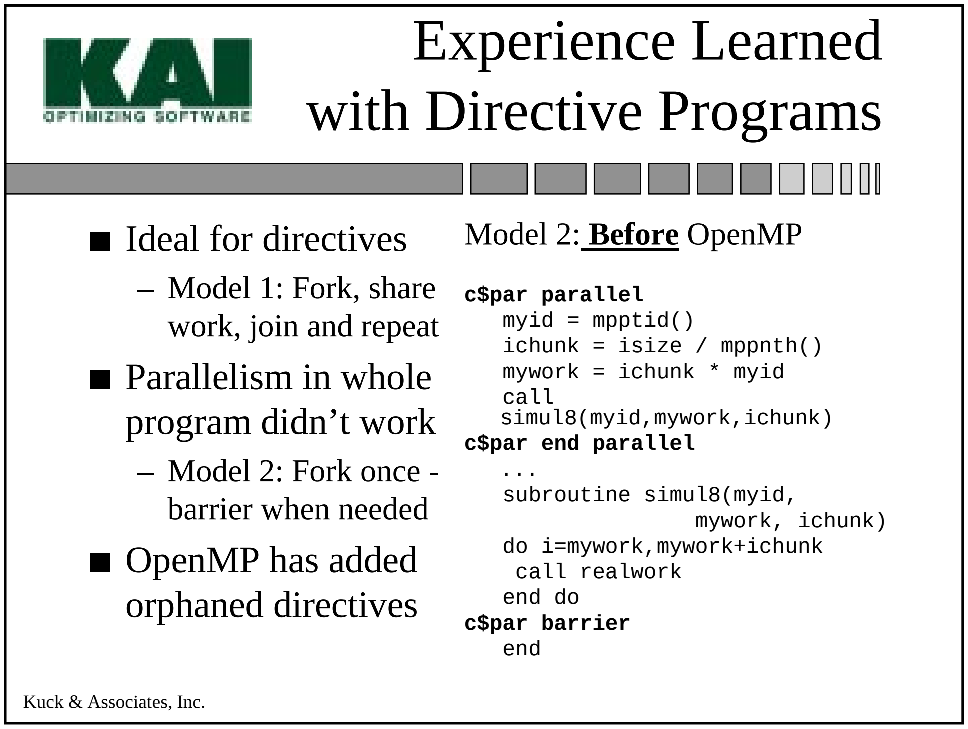
With Dynamic Threads

Sjob	Sjob	Pjob	
Pjob	Pjob	Pjob	
Pjob	Pjob	Pjob	

Now free time

- With Static Threads
  - Request 3 threads
  - Get exactly 3 threads
- With Dynamic Threads
  - Request 3 threads
  - Get 3 threads **if available**
- Your job gets done faster **and** your colleagues too
- Avoids over allocating processors

Kuck & Associates, Inc.



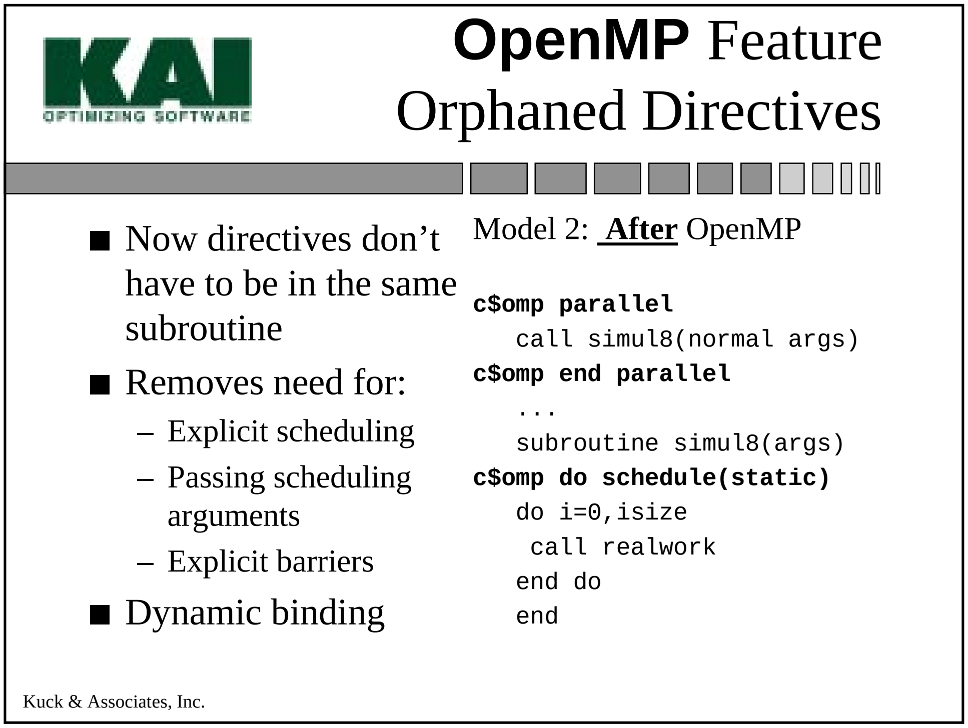
Experience Learned with Directive Programs

- Ideal for directives
  - Model 1: Fork, share work, join and repeat
- Parallelism in whole program didn't work
  - Model 2: Fork once - barrier when needed
- OpenMP has added orphaned directives

Model 2: **Before** OpenMP

```
c$par parallel
   myid = mpptid()
   ichunk = isize / mppnth()
   mywork = ichunk * myid
   call
   simul8(myid,mywork,ichunk)
c$par end parallel
   ...
   subroutine simul8(myid,
                  mywork, ichunk)
   do i=mywork,mywork+ichunk
    call realwork
   end do
c$par barrier
   end
```

Kuck & Associates, Inc.



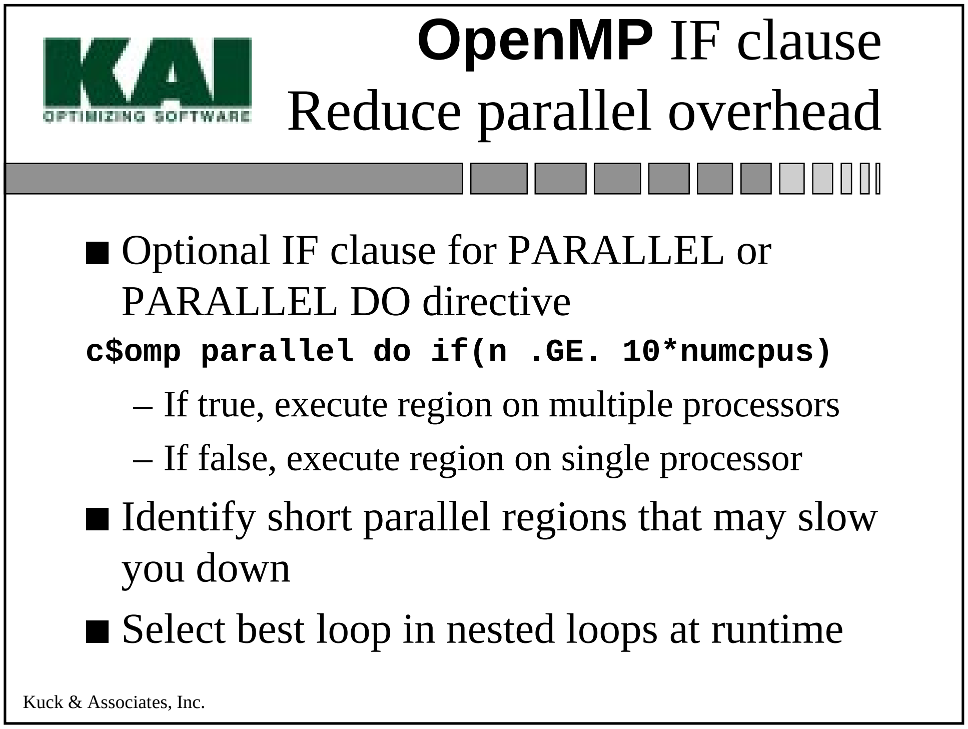
OpenMP Feature
Orphaned Directives

- Now directives don’t have to be in the same subroutine
- Removes need for:
  - Explicit scheduling
  - Passing scheduling arguments
  - Explicit barriers
- Dynamic binding

Model 2: **After** OpenMP

```
c$omp parallel
   call simul8(normal args)
c$omp end parallel
   ...
   subroutine simul8(args)
c$omp do schedule(static)
   do i=0,isize
    call realwork
   end do
   end
```

Kuck & Associates, Inc.



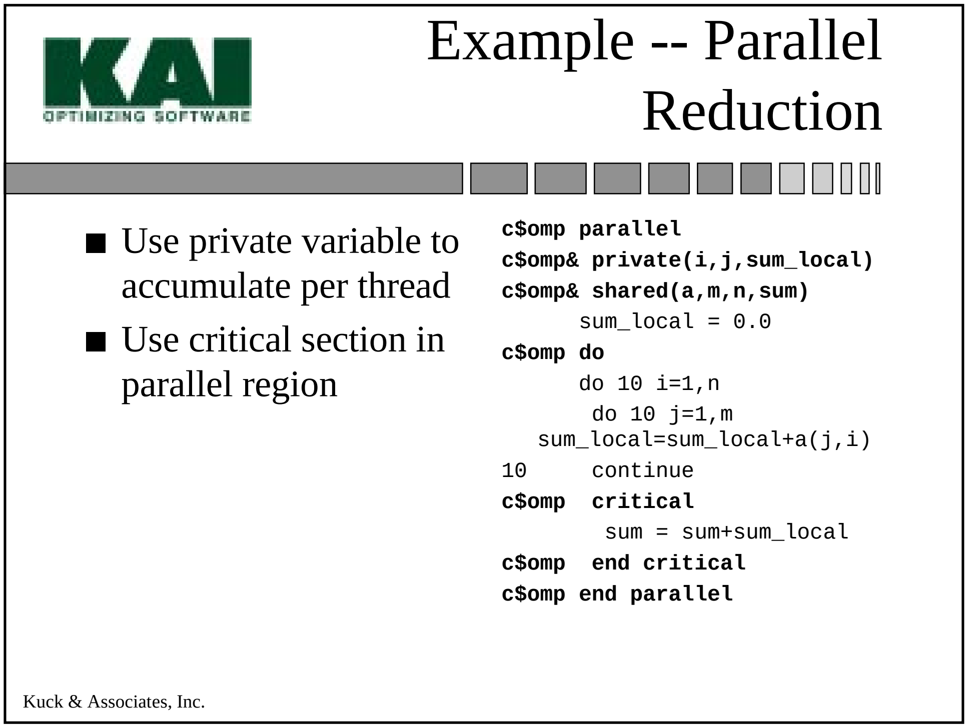
OpenMP IF clause
Reduce parallel overhead

- Optional IF clause for PARALLEL or PARALLEL DO directive

```
c$omp parallel do if(n .GE. 10*numcpus)
```

  - If true, execute region on multiple processors
  - If false, execute region on single processor
- Identify short parallel regions that may slow you down
- Select best loop in nested loops at runtime

Kuck & Associates, Inc.

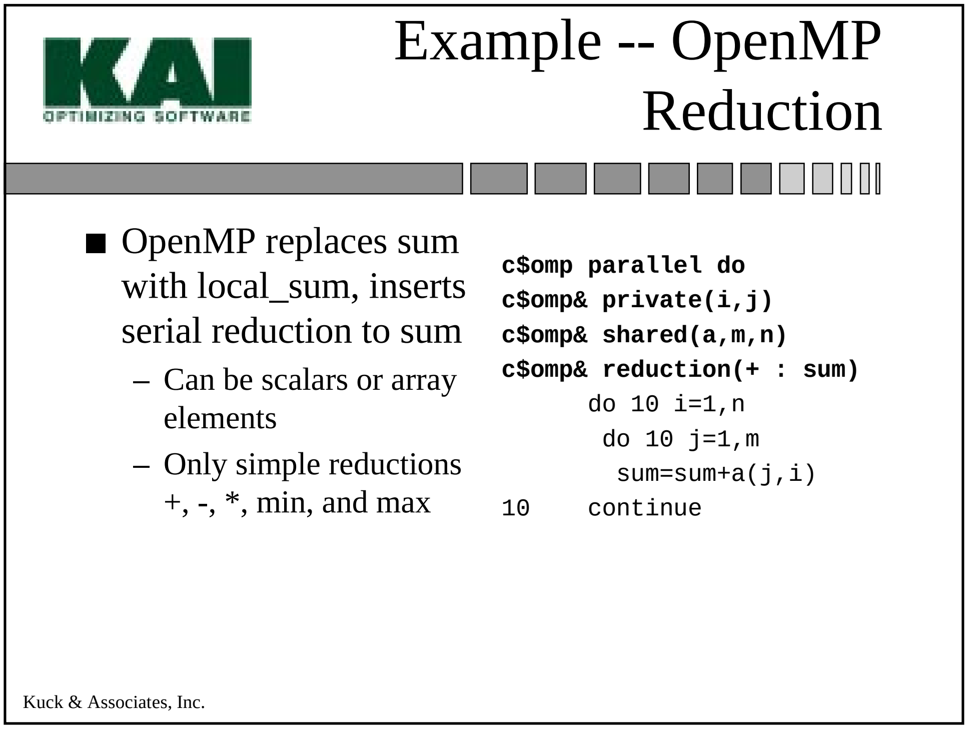


Example -- Parallel Reduction

- Use private variable to accumulate per thread
- Use critical section in parallel region

```
c$omp parallel
c$omp& private(i,j,sum_local)
c$omp& shared(a,m,n,sum)
      sum_local = 0.0
c$omp do
      do 10 i=1,n
       do 10 j=1,m
   sum_local=sum_local+a(j,i)
10     continue
c$omp  critical
        sum = sum+sum_local
c$omp  end critical
c$omp end parallel
```

Kuck & Associates, Inc.



Example -- OpenMP Reduction

- OpenMP replaces sum with local_sum, inserts serial reduction to sum
  - Can be scalars or array elements
  - Only simple reductions +, -, *, min, and max

```
c$omp parallel do
c$omp& private(i,j)
c$omp& shared(a,m,n)
c$omp& reduction(+ : sum)
      do 10 i=1,n
       do 10 j=1,m
        sum=sum+a(j,i)
10    continue
```

Kuck & Associates, Inc.

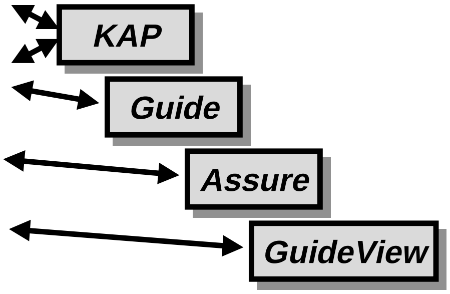
KPTS + OpenMP = Parallel Software Engineering

- Performance --
  - Meets or beats all modes on scalability
- Productivity --
  - Much easier to use than other modes
- Quality --
  - Enables parallelism validation for first time
- Standards --
  - Defacto becoming fact

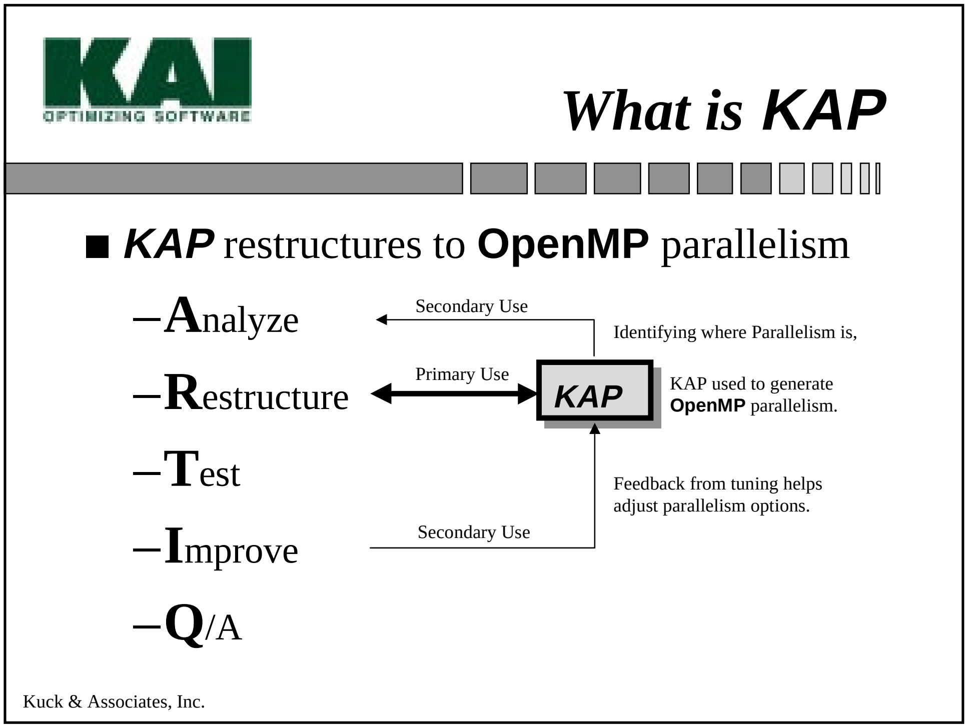
Kuck & Associates, Inc.

OpenMP Parallel Software Engineering with *KPTS*

- Steps in Parallelizing an Application
  - **A**nalyze
  - **R**estructure
  - **T**est
  - **I**mprove
  - **Q**/A



Kuck & Associates, Inc.



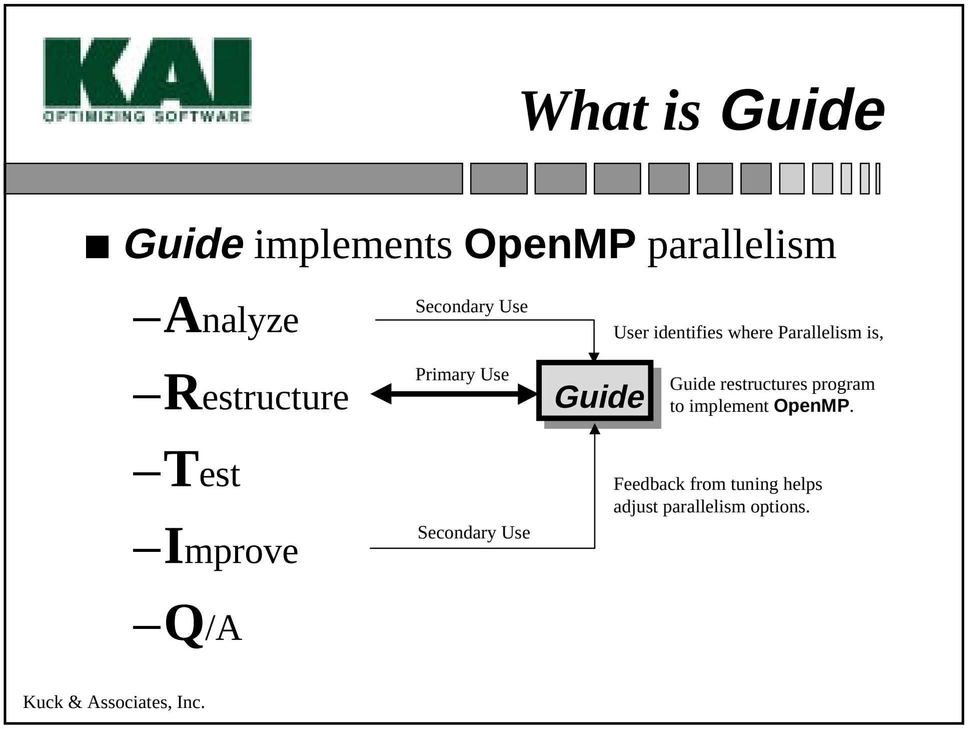
KAI
OPTIMIZING SOFTWARE

What is KAP

- ***KAP*** restructures to **OpenMP** parallelism
  - –**A**nalyze
  - –**R**estructure
  - –**T**est
  - –**I**mprove
  - –**Q**/A

Secondary Use
Identifying where Parallelism is,
Primary Use
KAP
KAP used to generate **OpenMP** parallelism.
Feedback from tuning helps adjust parallelism options.
Secondary Use

Kuck & Associates, Inc.



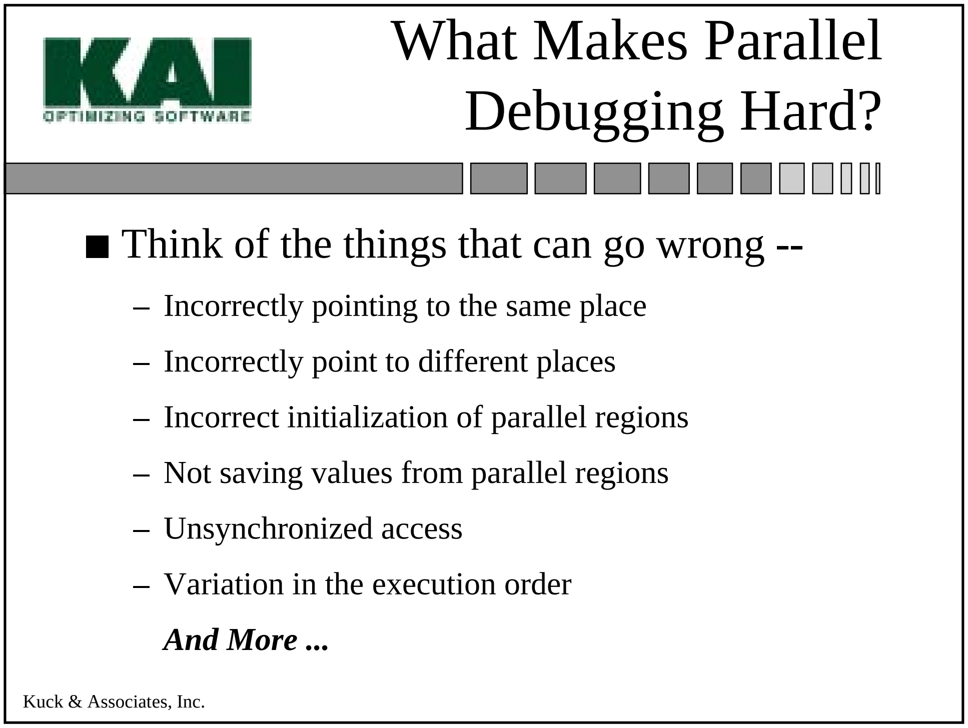
KAI
OPTIMIZING SOFTWARE

What is Guide

- ***Guide*** implements **OpenMP** parallelism
  - –**A**nalyze
  - –**R**estructure
  - –**T**est
  - –**I**mprove
  - –**Q**/A

Secondary Use
User identifies where Parallelism is,
Primary Use
Guide
Guide restructures program to implement **OpenMP**.
Feedback from tuning helps adjust parallelism options.
Secondary Use

Kuck & Associates, Inc.

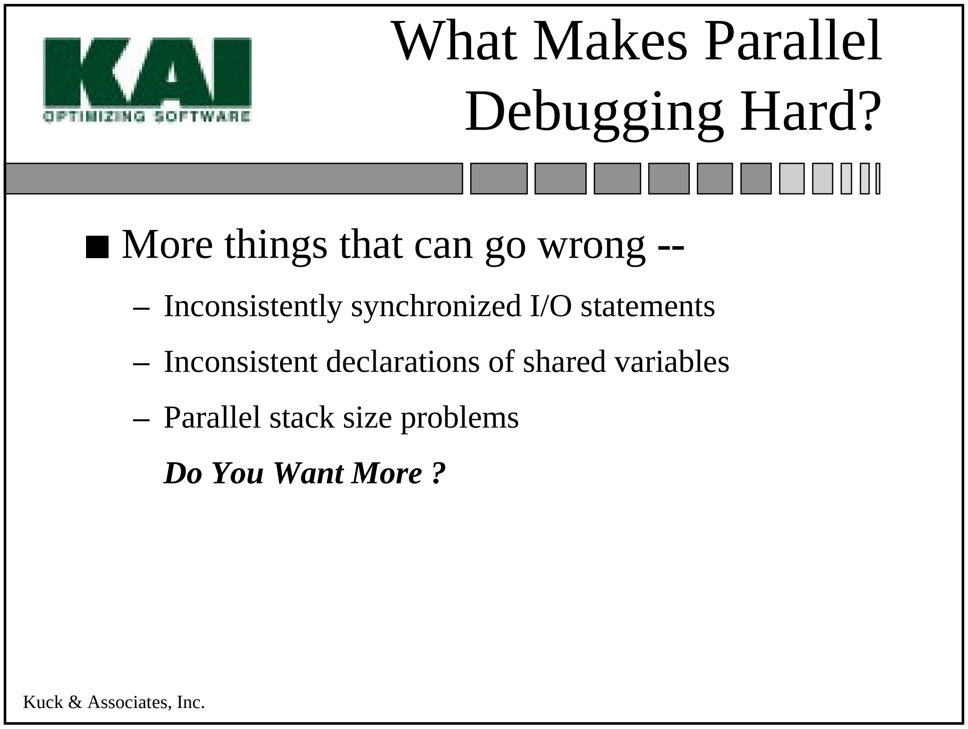


What Makes Parallel Debugging Hard?

- Think of the things that can go wrong --
  - Incorrectly pointing to the same place
  - Incorrectly point to different places
  - Incorrect initialization of parallel regions
  - Not saving values from parallel regions
  - Unsynchronized access
  - Variation in the execution order

  And More ...

Kuck & Associates, Inc.

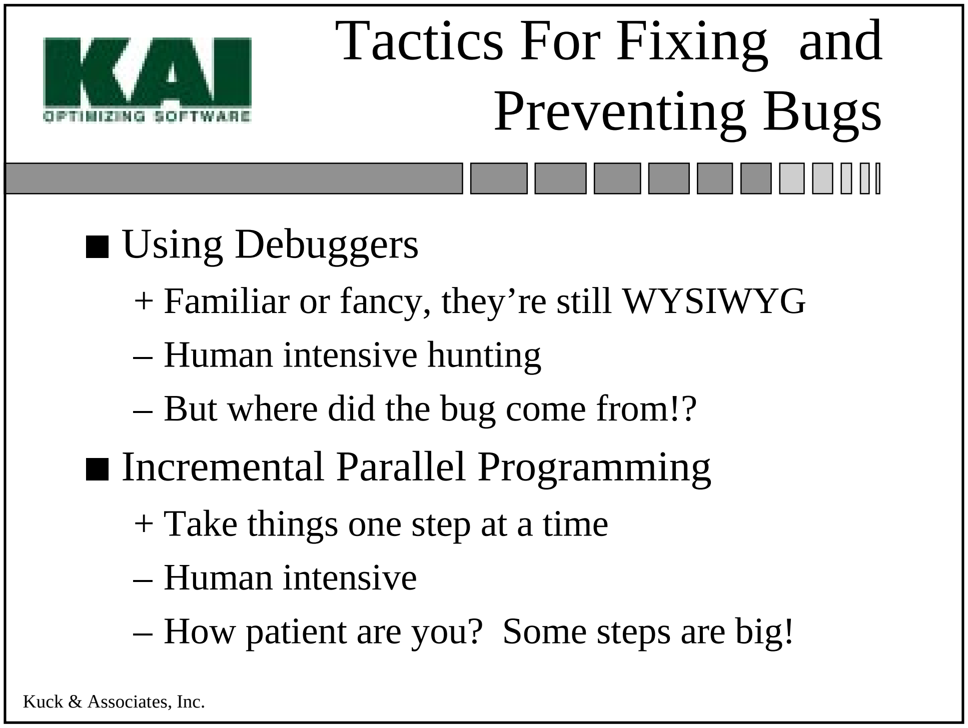


What Makes Parallel Debugging Hard?

- More things that can go wrong --
  - Inconsistently synchronized I/O statements
  - Inconsistent declarations of shared variables
  - Parallel stack size problems

  Do You Want More ?

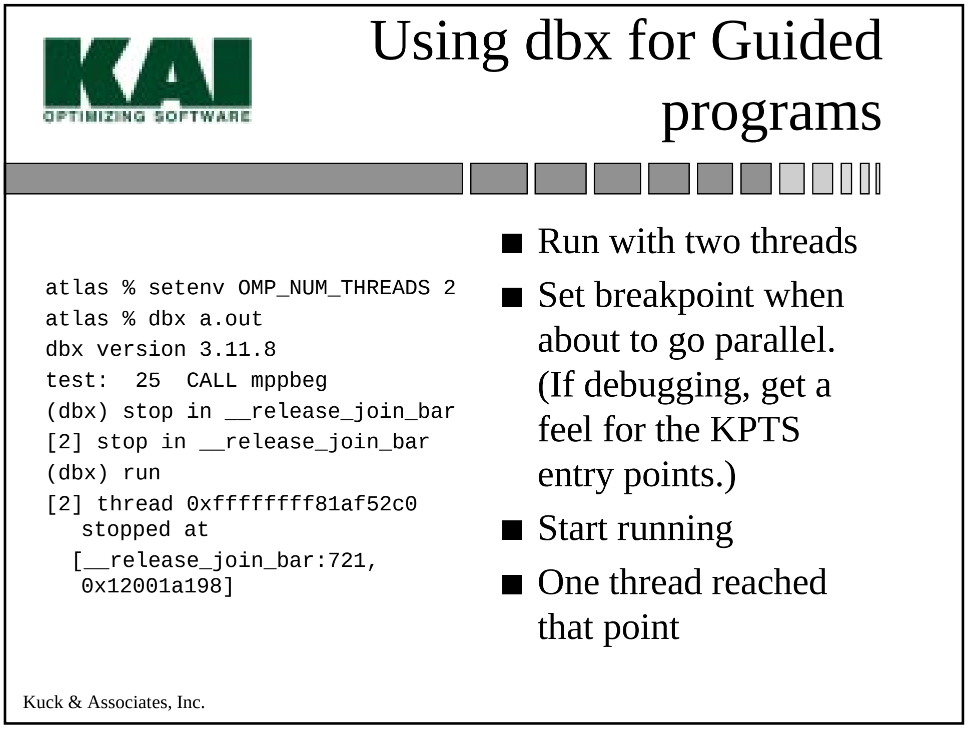
Kuck & Associates, Inc.



Tactics For Fixing and Preventing Bugs

- Using Debuggers
  - + Familiar or fancy, they're still WYSIWYG
  - – Human intensive hunting
  - – But where did the bug come from!?
- Incremental Parallel Programming
  - + Take things one step at a time
  - – Human intensive
  - – How patient are you? Some steps are big!

Kuck & Associates, Inc.

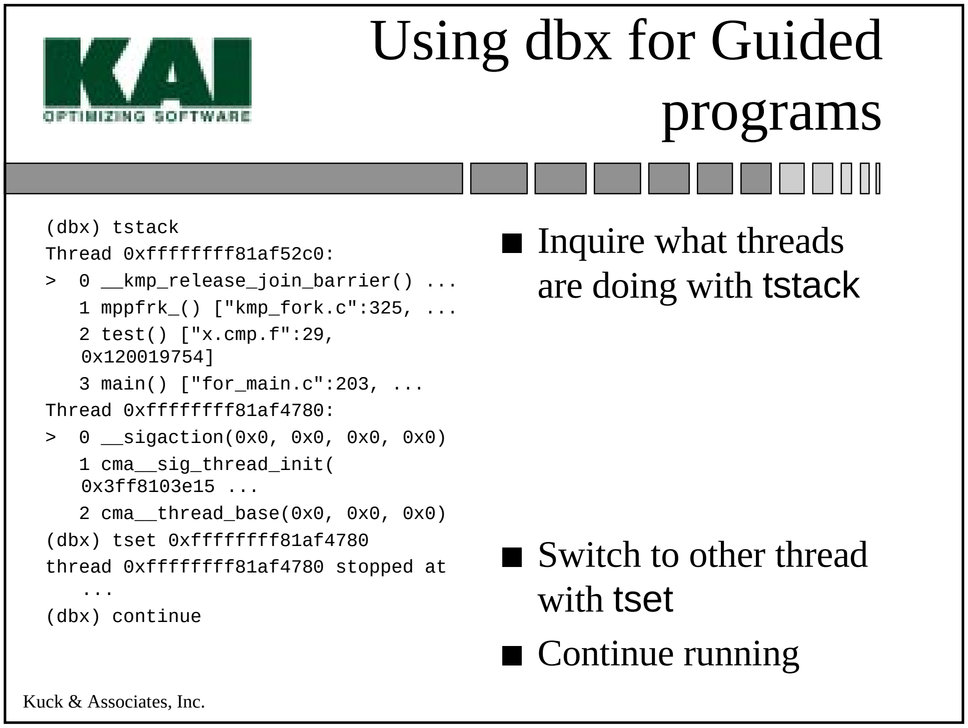


Using dbx for Guided programs

```
atlas % setenv OMP_NUM_THREADS 2
atlas % dbx a.out
dbx version 3.11.8
test:  25  CALL mppbeg
(dbx) stop in __release_join_bar
[2] stop in __release_join_bar
(dbx) run
[2] thread 0xffffffff81af52c0
   stopped at
  [__release_join_bar:721,
   0x12001a198]
```

- Run with two threads
- Set breakpoint when about to go parallel. (If debugging, get a feel for the KPTS entry points.)
- Start running
- One thread reached that point

Kuck & Associates, Inc.

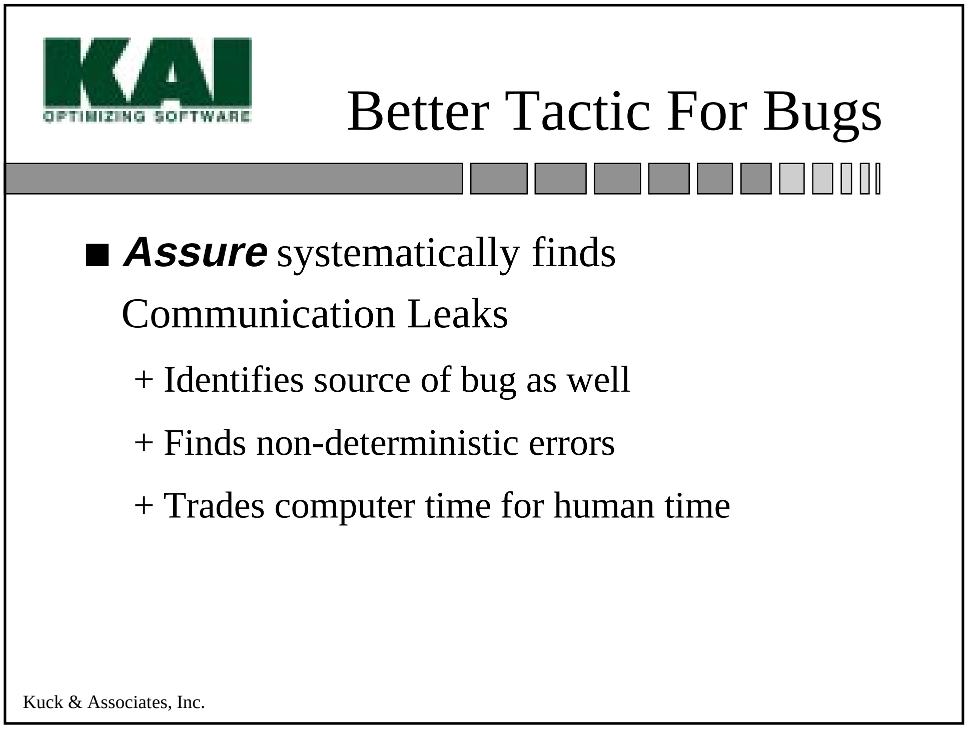


Using dbx for Guided programs

```
(dbx) tstack
Thread 0xffffffff81af52c0:
>  0 __kmp_release_join_barrier() ...
   1 mppfrk_() ["kmp_fork.c":325, ...
   2 test() ["x.cmp.f":29,
   0x120019754]
   3 main() ["for_main.c":203, ...
Thread 0xffffffff81af4780:
>  0 __sigaction(0x0, 0x0, 0x0, 0x0)
   1 cma__sig_thread_init(
   0x3ff8103e15 ...
   2 cma__thread_base(0x0, 0x0, 0x0)
(dbx) tset 0xffffffff81af4780
thread 0xffffffff81af4780 stopped at
   ...
(dbx) continue
```

- Inquire what threads are doing with tstack
- Switch to other thread with tset
- Continue running

Kuck & Associates, Inc.



Better Tactic For Bugs

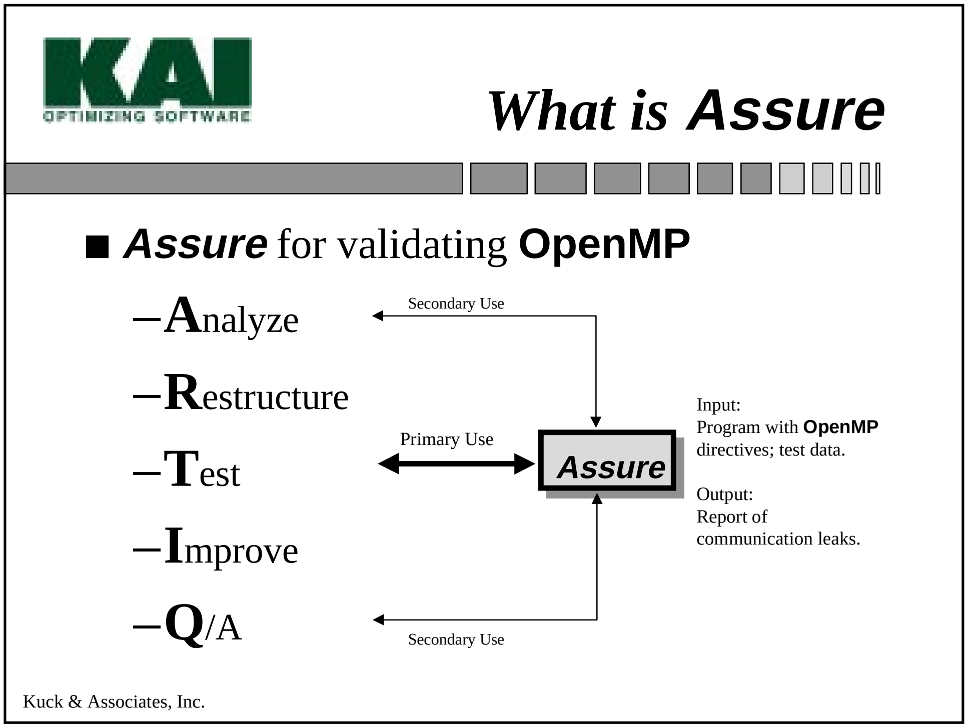
- ***Assure*** systematically finds Communication Leaks

  + Identifies source of bug as well

  + Finds non-deterministic errors

  + Trades computer time for human time

Kuck & Associates, Inc.



What is ***Assure***

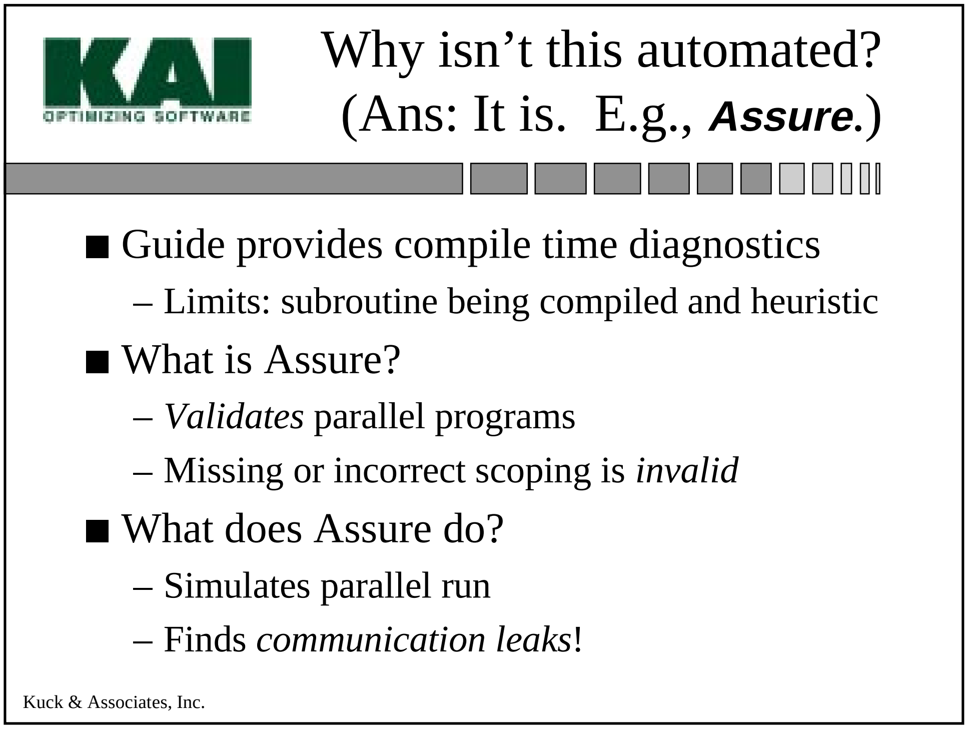
- ***Assure*** for validating **OpenMP**
  - –**A**nalyze
  - –**R**estructure
  - –**T**est
  - –**I**mprove
  - –**Q**/A

Secondary Use
Primary Use
Assure
Secondary Use

Input:
Program with **OpenMP**
directives; test data.

Output:
Report of
communication leaks.

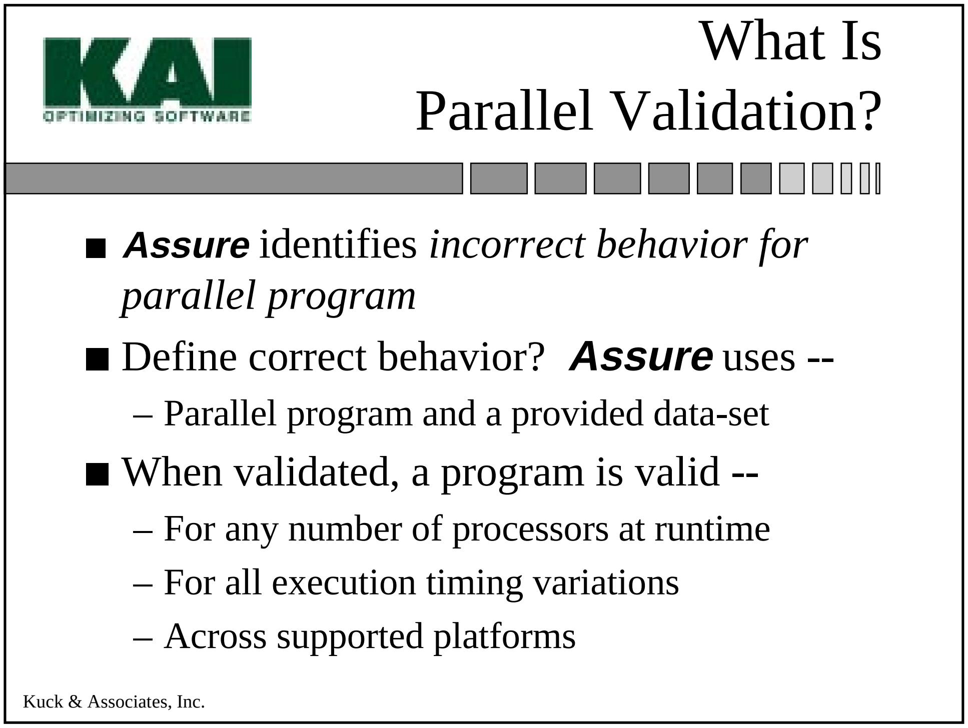
Kuck & Associates, Inc.



Why isn't this automated?
(Ans: It is. E.g., ***Assure***.)

- Guide provides compile time diagnostics
  - – Limits: subroutine being compiled and heuristic
- What is Assure?
  - – *Validates* parallel programs
  - – Missing or incorrect scoping is *invalid*
- What does Assure do?
  - – Simulates parallel run
  - – Finds *communication leaks*!

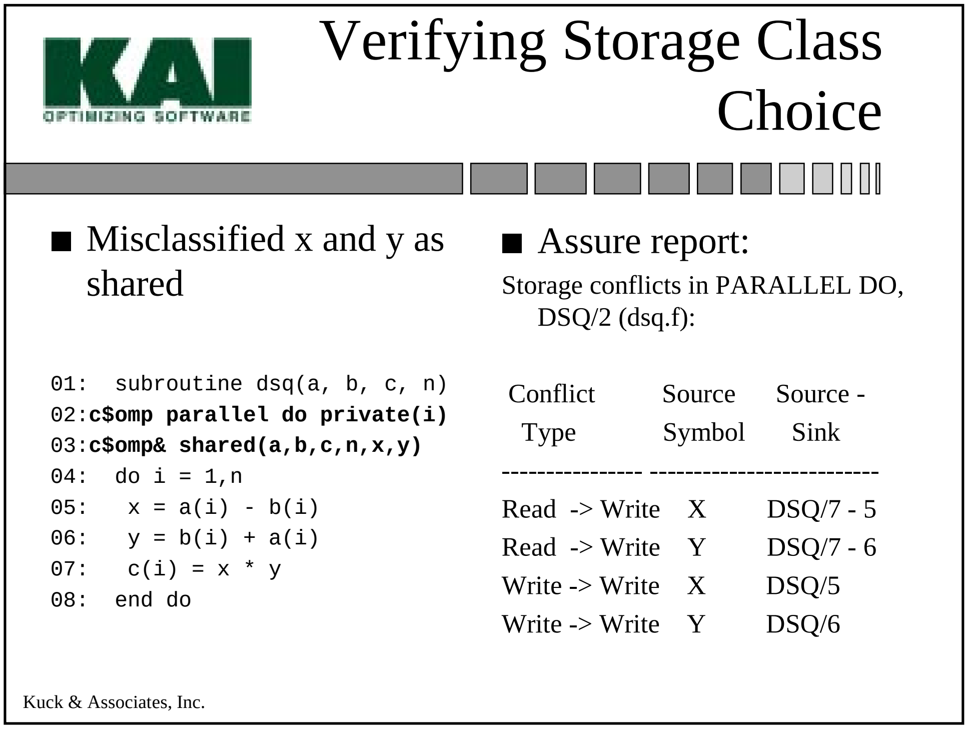
Kuck & Associates, Inc.



What Is Parallel Validation?

- ***Assure*** identifies *incorrect behavior for parallel program*
- Define correct behavior? ***Assure*** uses --
  - Parallel program and a provided data-set
- When validated, a program is valid --
  - For any number of processors at runtime
  - For all execution timing variations
  - Across supported platforms

Kuck & Associates, Inc.



Verifying Storage Class Choice

- Misclassified x and y as shared

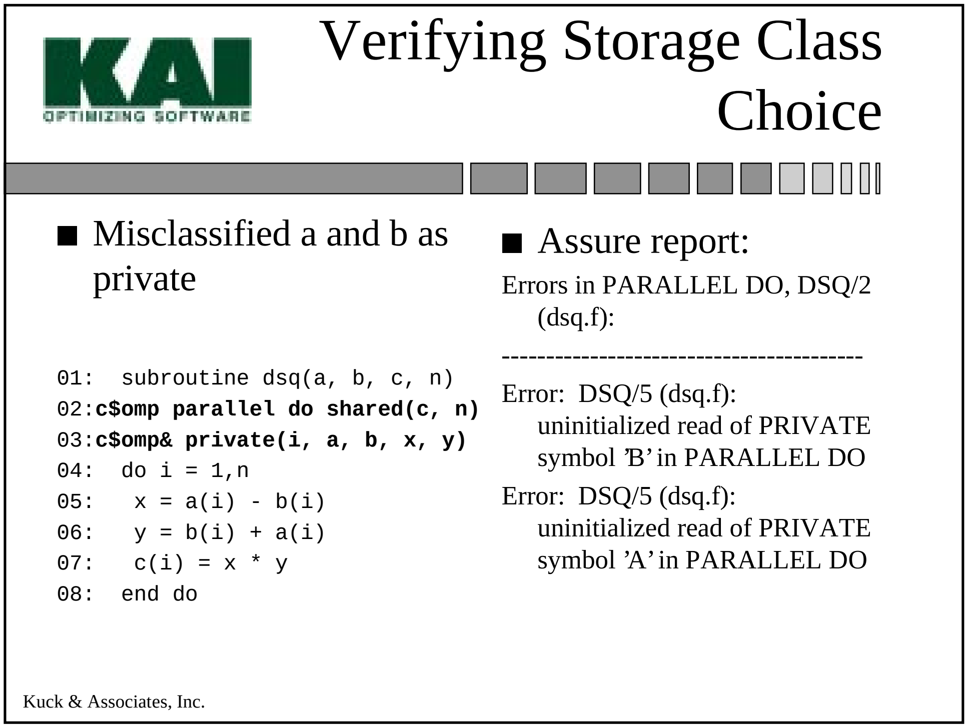
```
01:  subroutine dsq(a, b, c, n)
02:c$omp parallel do private(i)
03:c$omp& shared(a,b,c,n,x,y)
04:  do i = 1,n
05:   x = a(i) - b(i)
06:   y = b(i) + a(i)
07:   c(i) = x * y
08:  end do
```

- Assure report:

Storage conflicts in PARALLEL DO, DSQ/2 (dsq.f):

Conflict Type	Source Symbol	Source - Sink
Read -> Write	X	DSQ/7 - 5
Read -> Write	Y	DSQ/7 - 6
Write -> Write	X	DSQ/5
Write -> Write	Y	DSQ/6

Kuck & Associates, Inc.



Verifying Storage Class Choice

- Misclassified a and b as private

```
01:  subroutine dsq(a, b, c, n)
02:c$omp parallel do shared(c, n)
03:c$omp& private(i, a, b, x, y)
04:  do i = 1,n
05:   x = a(i) - b(i)
06:   y = b(i) + a(i)
07:   c(i) = x * y
08:  end do
```

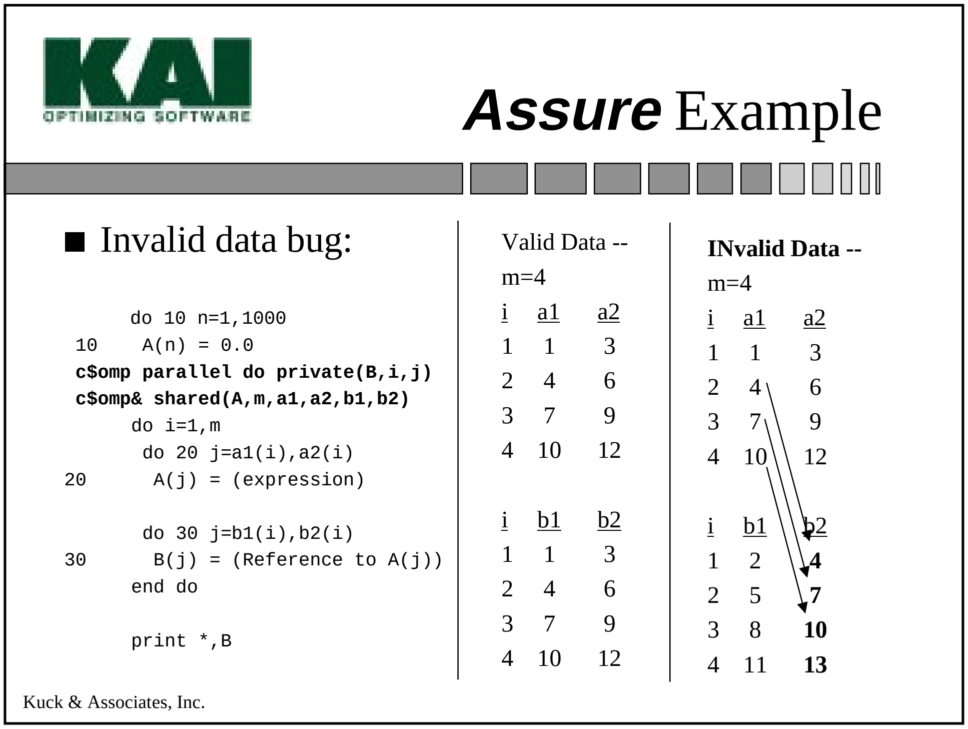
- Assure report:

Errors in PARALLEL DO, DSQ/2 (dsq.f):

Error: DSQ/5 (dsq.f): uninitialized read of PRIVATE symbol 'B' in PARALLEL DO

Error: DSQ/5 (dsq.f): uninitialized read of PRIVATE symbol 'A' in PARALLEL DO

Kuck & Associates, Inc.



Assure Example

- Invalid data bug:

```
     do 10 n=1,1000
10     A(n) = 0.0
c$omp parallel do private(B,i,j)
c$omp& shared(A,m,a1,a2,b1,b2)
     do i=1,m
      do 20 j=a1(i),a2(i)
20     A(j) = (expression)

      do 30 j=b1(i),b2(i)
30     B(j) = (Reference to A(j))
     end do

     print *,B
```

Valid Data -- m=4

i	a1	a2
1	1	3
2	4	6
3	7	9
4	10	12

i	b1	b2
1	1	3
2	4	6
3	7	9
4	10	12

INvalid Data -- m=4

i	a1	a2
1	1	3
2	4	6
3	7	9
4	10	12

i	b1	b2
1	2	**4**
2	5	**7**
3	8	**10**
4	11	**13**

Kuck & Associates, Inc.

Assure Example

- Invalid data bug:

```
01:     do 10 i=1,20
02: 10   A(i) = 0.0
03:c$omp parallel do private(B,i,j)
04:c$omp& shared(A,a1,a2,b1,b2,m)
05:     do i=1,m
06:      do 20 j=a1(i),a2(i)
07: 20    A(j) = (Expression)
08:      do 30 j=b1(i),b2(i)
09: 30    B(j) = (Func of A(j))
10:     end do
11:     do 40 i=1,20
12: 40    print *, B(i)
```

- Assure Report:

Errors in ROUTINE, ITER/1 (iter.f):

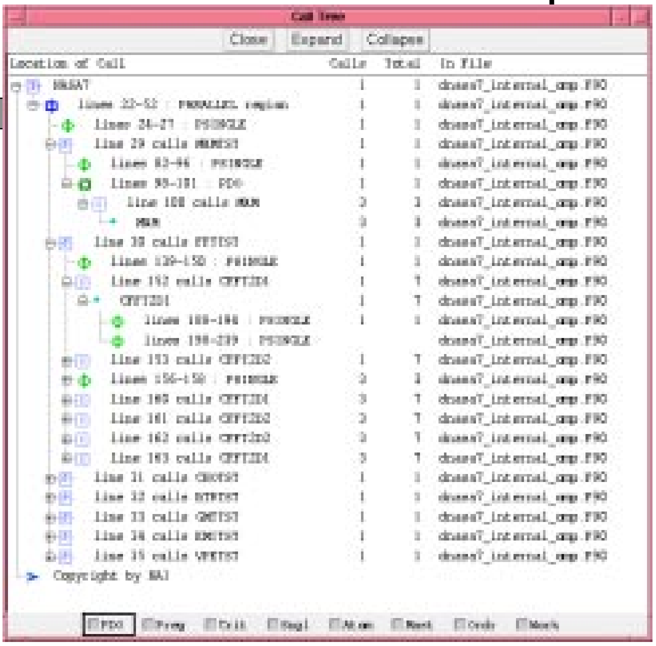
Error: ITER/9 (iter.f): PRIVATE symbol 'B' referenced outside parallel construct as 'B' in ITER/12

Storage conflicts in PARALLEL DO, ITER/16 (iter.f):

Conflict Type	Source Symbol	Source - Sink Routine/Line
Write -> Read	A	ITER/7 - 9

Kuck & Associates, Inc.

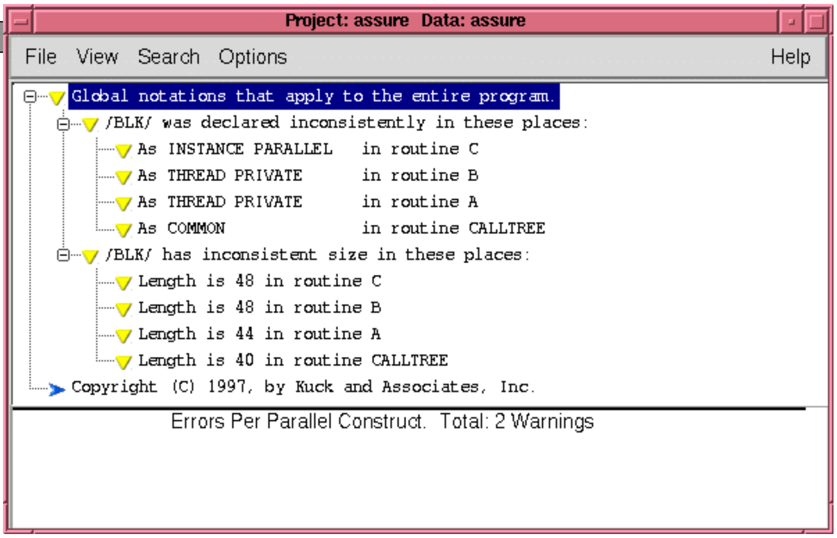
AssureView: Call Graph



```
Location of Call                          Calls  Total  In File
HRBAT7                                      1      1    dwass7_internal_omp.F90
  lines 22-52 : PARALLEL region             1      1    dwass7_internal_omp.F90
    lines 24-27 : PSINGLE                   1      1    dwass7_internal_omp.F90
    line 29 calls MRMTST                    1      1    dwass7_internal_omp.F90
      lines 82-96 : PSINGLE                 1      1    dwass7_internal_omp.F90
      lines 95-101 : PDO                    1      1    dwass7_internal_omp.F90
        line 100 calls MAN                  3      3    dwass7_internal_omp.F90
          MAN                               3      3    dwass7_internal_omp.F90
    line 30 calls FFTTST                    1      1    dwass7_internal_omp.F90
      lines 139-150 : PSINGLE               1      1    dwass7_internal_omp.F90
      line 152 calls CFFT2D1                1      7    dwass7_internal_omp.F90
        CFFT2D1                             1      7    dwass7_internal_omp.F90
          lines 188-196 : PSINGLE           1      1    dwass7_internal_omp.F90
          lines 198-239 : PSINGLE                       dwass7_internal_omp.F90
      line 153 calls CFFT2D2                1      7    dwass7_internal_omp.F90
      lines 156-158 : PSINGLE               3      3    dwass7_internal_omp.F90
      line 160 calls CFFT2D1                3      7    dwass7_internal_omp.F90
      line 161 calls CFFT2D2                3      7    dwass7_internal_omp.F90
      line 162 calls CFFT2D2                3      7    dwass7_internal_omp.F90
      line 163 calls CFFT2D1                3      7    dwass7_internal_omp.F90
    line 31 calls CHOTST                    1      1    dwass7_internal_omp.F90
    line 32 calls BTRTST                    1      1    dwass7_internal_omp.F90
    line 33 calls GMTTST                    1      1    dwass7_internal_omp.F90
    line 34 calls EMITST                    1      1    dwass7_internal_omp.F90
    line 35 calls VFFTST                    1      1    dwass7_internal_omp.F90
Copyright by KAI
```

Kuck & Associates, Inc.

AssureView: Common Mismatch

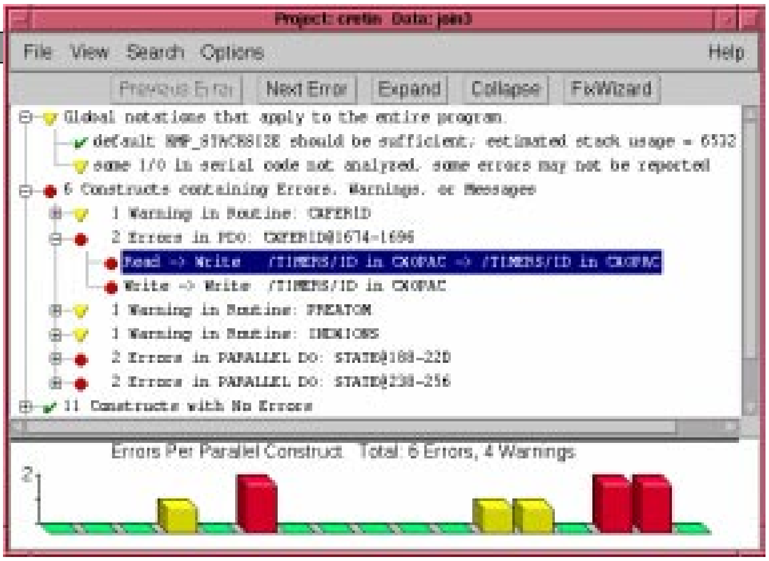


```
⊟ Global notations that apply to the entire program.
   ⊟ /BLK/ was declared inconsistently in these places:
        As INSTANCE PARALLEL   in routine C
        As THREAD PRIVATE      in routine B
        As THREAD PRIVATE      in routine A
        As COMMON              in routine CALLTREE
   ⊟ /BLK/ has inconsistent size in these places:
        Length is 48 in routine C
        Length is 48 in routine B
        Length is 44 in routine A
        Length is 40 in routine CALLTREE
  Copyright (C) 1997, by Kuck and Associates, Inc.
```

Errors Per Parallel Construct. Total: 2 Warnings

Kuck & Associates, Inc.

AssureView: Errors

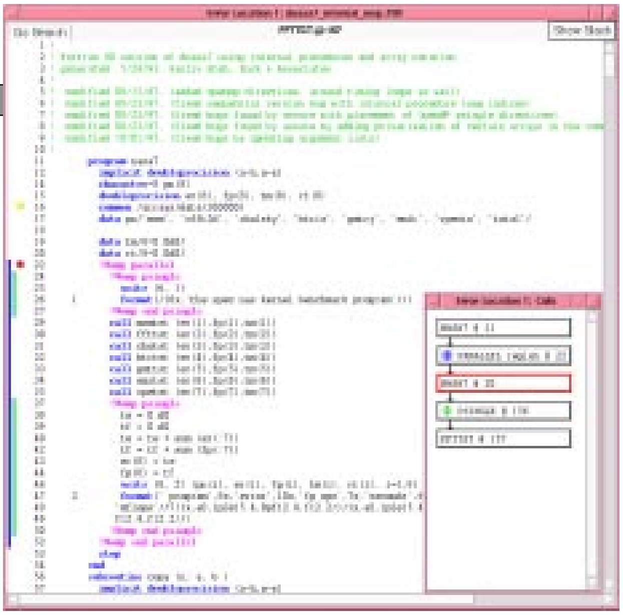


```
⊟ Global notations that apply to the entire program.
     default KMP_STACKSIZE should be sufficient; estimated stack usage = 6532
     some I/O in serial code not analyzed. some errors may not be reported
⊟ 6 Constructs containing Errors, Warnings, or Messages
   ⊞   1 Warning in Routine: CXFERID
   ⊟   2 Errors in PDO: CXFERID@1674-1696
          Read -> Write   /TIMERS/ID in CX0PAC -> /TIMERS/ID in CX0PAC
          Write -> Write  /TIMERS/ID in CX0PAC
   ⊞   1 Warning in Routine: FREATOM
   ⊞   1 Warning in Routine: INDXIONS
   ⊞   2 Errors in PARALLEL DO: STATE@188-220
   ⊞   2 Errors in PARALLEL DO: STATE@238-256
⊞ 11 Constructs with No Errors
```

Errors Per Parallel Construct. Total: 6 Errors, 4 Warnings

Kuck & Associates, Inc.

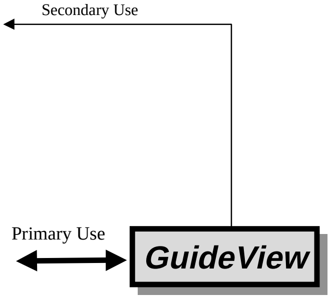
AssureView:Source



Kuck & Associates, Inc.

What is GuideView

- ***GuideView*** visualizes **OpenMP**
  - **A**nalyze
  - **R**estructure
  - **T**est
  - **I**mprove
  - **Q**/A



Input:
Program with **OpenMP** directives; test data.

Function:
Navigate complex problem down to performance problems

Kuck & Associates, Inc.

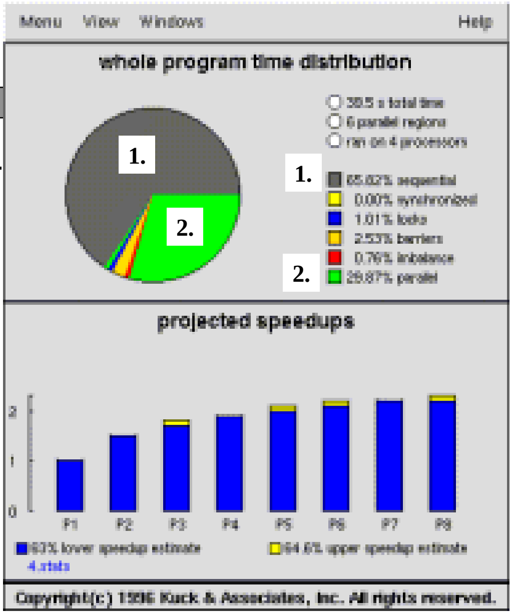
Example Profile

```
Each sample covers 4.00 byte(s) for 8.3e-06% of 11743.0342 seconds
%time   seconds  cum % cum sec  procedure (file)
  8.5  1003.0098   8.5  1003.01 proj_slv_node_on_mr_seg_
  4.7   548.3232  17.9  2102.68 shell_force_el_pl_4p_
  4.4   522.3145  22.4  2624.99 pkshell_egy_el_pl_v__
  3.8   450.5781  26.2  3075.57 shell_lam_stress_pl_3d_
  3.3   382.9053  40.2  4718.43 pkcalc_node_vel__
  3.2   372.4639  43.4  5090.90 pkcalc_node_accel__
  3.1   365.8145  46.5  5456.71 shell_stress_el_pl_
  2.6   306.9453  60.8  7144.42 pkcontact_node_trial_vel_posn_1_
  2.4   276.1680  63.2  7420.58 trans_lvecs_to_gvecs_
  2.3   270.2891  65.5  7690.87 spin_wait_join_barrier
   . . .
```

Kuck & Associates, Inc.

GuideView adds focus

- Visualize the performance report
- Pie Chart --
  - Percent parallel
  - Overhead
    - Locks
    - Barriers
    - Imbalance
- Bar Chart --
  - Amdahl's Law

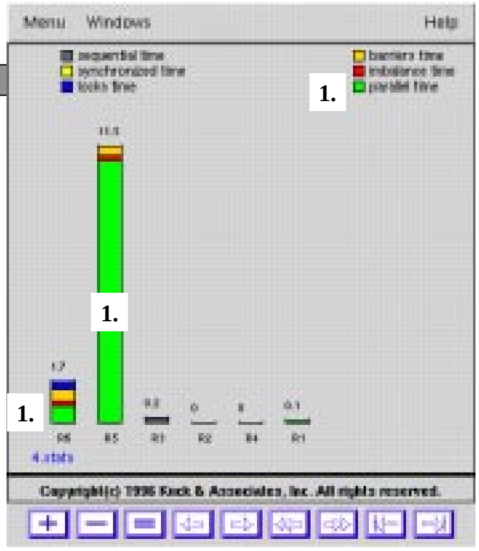


Kuck & Associates, Inc.

GuideView adds focus

KAI
OPTIMIZING SOFTWARE

- Parallel regions
- Sorted by --
  - Overhead
  - Time
- Compare --
  - Parallel time
  - Synched time
  - Lock time
  - Barrier time



Copyright(c) 1996 Kuck & Associates, Inc. All rights reserved.

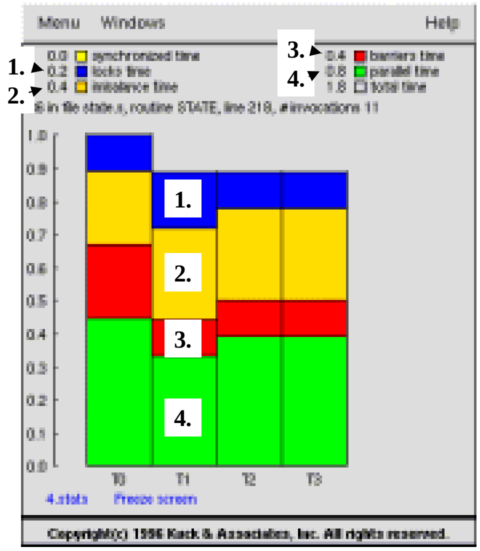
Kuck & Associates, Inc.

GuideView adds focus

KAI
OPTIMIZING SOFTWARE

- Per Region
  & Per Thread
- High Barrier Time!
- Load Imbalance Possible

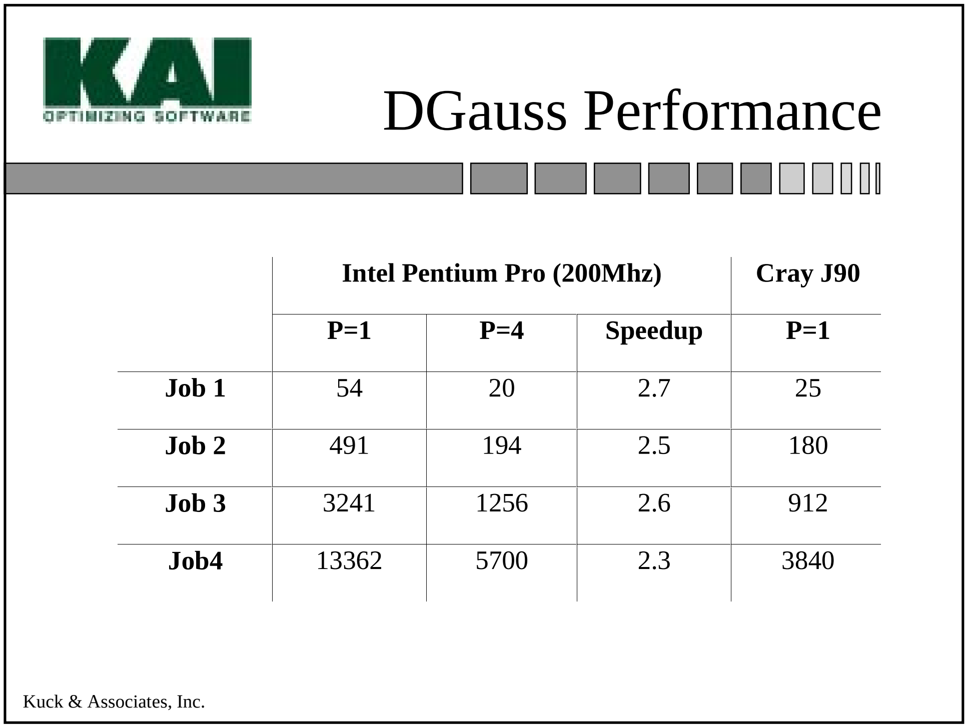
(Note: this is a small run)



6 in file state.s, routine STATE, line 218, #invocations 11

Copyright(c) 1996 Kuck & Associates, Inc. All rights reserved.

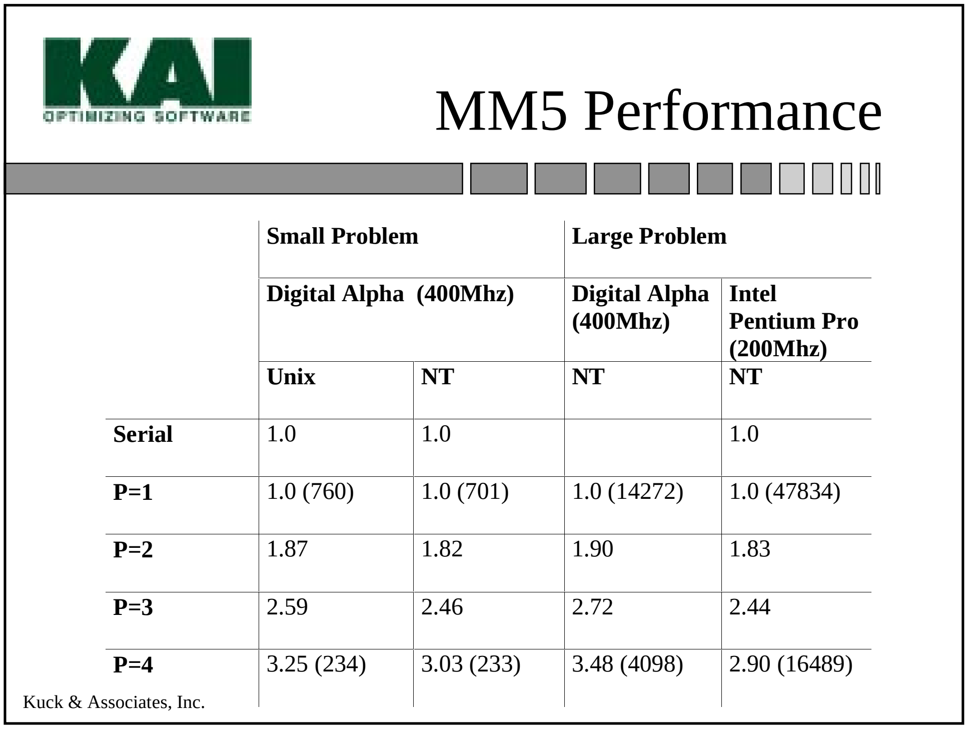
Kuck & Associates, Inc.



DGauss Performance

	Intel Pentium Pro (200Mhz)			Cray J90
	P=1	**P=4**	**Speedup**	**P=1**
Job 1	54	20	2.7	25
Job 2	491	194	2.5	180
Job 3	3241	1256	2.6	912
Job4	13362	5700	2.3	3840

Kuck & Associates, Inc.



MM5 Performance

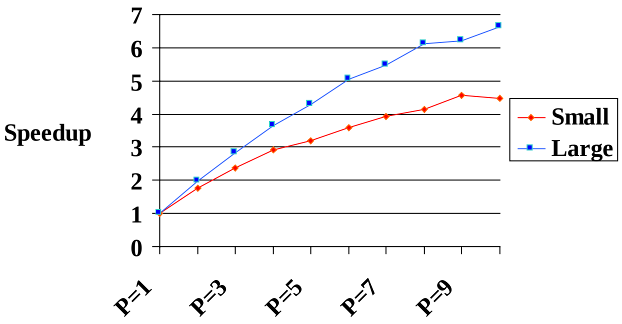
	Small Problem		Large Problem	
	Digital Alpha (400Mhz)		**Digital Alpha (400Mhz)**	**Intel Pentium Pro (200Mhz)**
	Unix	**NT**	**NT**	**NT**
Serial	1.0	1.0		1.0
P=1	1.0 (760)	1.0 (701)	1.0 (14272)	1.0 (47834)
P=2	1.87	1.82	1.90	1.83
P=3	2.59	2.46	2.72	2.44
P=4	3.25 (234)	3.03 (233)	3.48 (4098)	2.90 (16489)

Kuck & Associates, Inc.

KAI
OPTIMIZING SOFTWARE

MM5 Scalability

MM5 on Digital Alpha 8400 NT



Kuck & Associates, Inc.

KAI
OPTIMIZING SOFTWARE

Conclusion

`http://www.openmp.org`

- Parallel Software Engineering needs to be addressed to produce quality applications.
- OpenMP solves the portability problem.
- The KAP/Pro Toolset for OpenMP is available now!

Kuck & Associates, Inc.