



Compilers, Tools, and Performance

Darryl Gove

Compiler Performance Engineering

darryl.gove@sun.com

Compilers

- Sun Studio 12
 - > IDE/C/C++/Fortran/dbx/Performance Analyzer
 - > Free download <http://developers.sun.com/sunstudio/>
- GCC for SPARC Systems
 - > GCC compatible alternative
 - > Uses Sun Studio SPARC code generator
 - > Free download <http://cooltools.sunsource.net/gcc/>

Optimisation flags

- No optimisation flags = no optimisation
- `-O` = good degree of optimisation
- `-fast` = aggressive optimisation

Where does the time go?

- Always profile your code
- Collect data:
 - > `collect <app> <params>`
 - > `collect -P <pid>`
- View data
 - > `er_print test.N.er`
 - > `analyzer test.N.er`
- Use spot
 - > <http://cooltools.sunsource.net/spot/>
 - > `spot <app> <params>`

Application profile

Sun Studio Analyzer [test.2.er]

File View Timeline Help

Functions Callers-Callees Source Disassembly Timeline Experiments

User CPU (sec.)	User CPU (sec.)	Name
11.838	11.838	<Total>
11.818	11.838	main
0.020	0.020	_brk_unlocked
0.	0.020	malloc
0.	0.020	_malloc_unlocked
0.	0.020	_morecore
0.	0.020	sbrk
0.	0.020	_sbrk_unlocked
0.	11.838	_start

Debug flags for more information

- `-g` for C/Fortran
- `-g0` for C++
 - > `-g` disables front-end inlining in C++
- Little or no change to generated code
- Allows attribution of time to lines of source

Source level profile

Sun Studio Analyzer [test.2.er]

File View Timeline Help

Functions Callers-Callees **Source** Disassembly Timeline Experiments

User CPU (sec.)	User CPU (sec.)	Source File: ./ml.c Object File: ./ml Load Object: <ml>
0.	0.	20. int inset(double ix, double iy)
0.	0.	21. {
0.	0.	<Function: inset>
0.	0.	22. int iterations=0;
0.	0.	23. double x=ix, y=iy, x2=x*x, y2=y*y;
6.885	6.885	24. while ((x2+y2<4) && (iterations<1000))
2.141	2.141	25. {
0.430	0.430	26. y = 2 * x * y + iy;
0.480	0.480	27. x = x2 - y2 + ix;
1.411	1.411	28. x2 = x * x;
0.	0.	29. y2 = y * y;
0.	0.	30. iterations++;
0.	0.	31. }
0.	0.	32. return iterations;
		33. }

Where to get performance

- Algorithmic improvements
 - > Use library optimised library code (e.g. Perflib)
 - > Use better algorithms (e.g. Binary tree not vector)
- Compiler flags
 - > Additional optimisations (profile feedback, crossfile)
 - > Target hardware (e.g. FMAC on OPL)
- Source code improvements
 - > Clarifying code for compiler (e.g. Use local variables)
- Use multiple threads
 - > Automatic, POSIX threads, OpenMP

Options for parallelisation

- Autoparallelisation
 - > -xautopar -xreduction
 - > Easy to use
 - > Limited range of apps
- OpenMP
 - > Some skill needed
 - > Parallel for & sections
 - > 3.0 introduces tasks
- Pthreads
 - > Complex
 - > Flexible

OpenMP

- Add directives into source
- Incremental parallelisation
- Same source for serial and parallel code
- Limited parallelisation options in OpenMP 2.5
 - > Tasks added for OpenMP 3.0
- <http://www.openmp.org/>

OpenMP

```

void calculate()
{
    int x,y;
    double xv,yv;
    #pragma omp parallel for private(y,xv,yv)
    for (x=0; x<SIZE; x++){
        for (y=0; y<SIZE; y++){
            xv = ((double) (x-SIZE/2))
                / (double) (SIZE/4);
            yv = ((double) (y-SIZE/2))
                / (double) (SIZE/4);
            data[x][y]=inset(xv,yv);
        }
    }
}

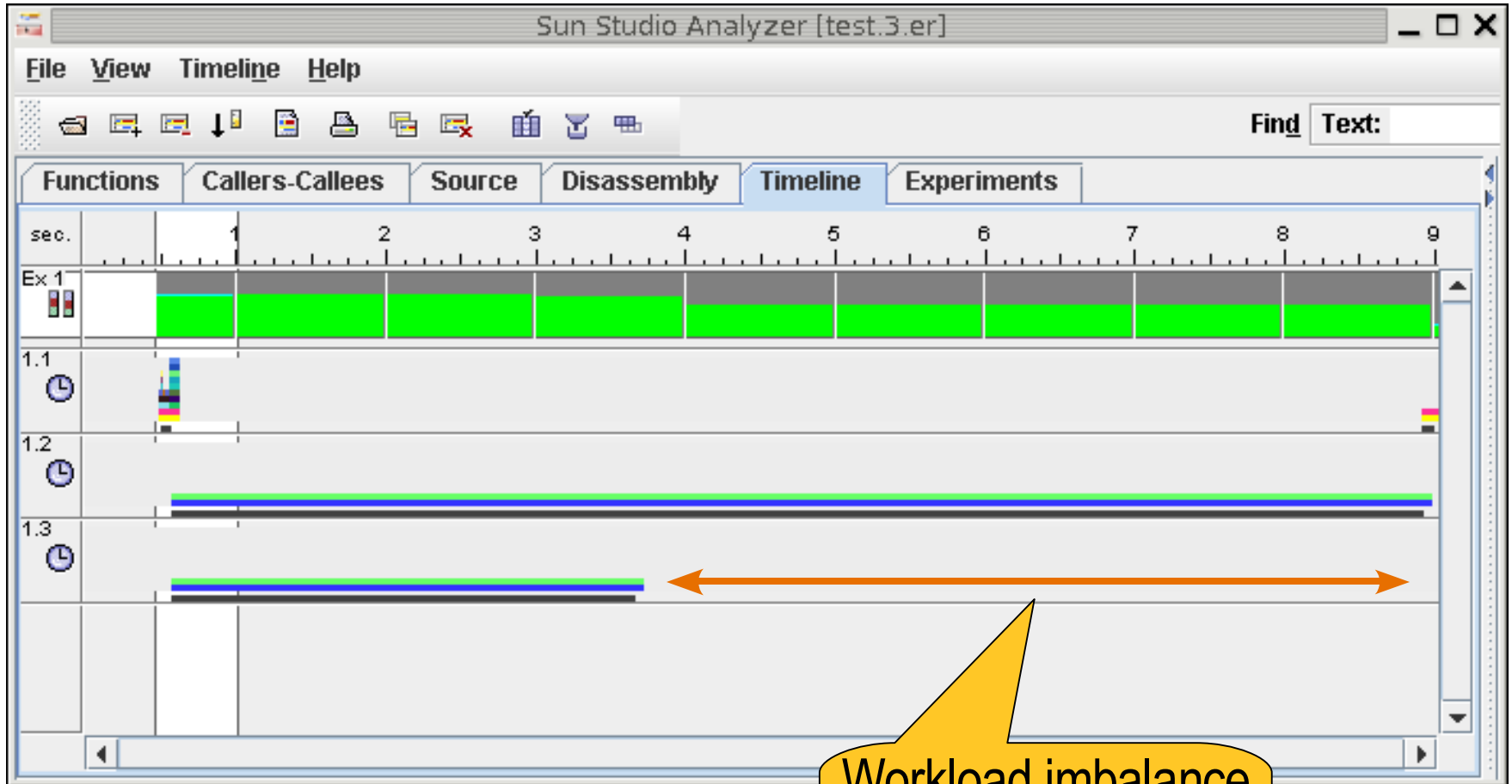
```

OpenMP 3.0 - tasks

- Spread tasks over multiple threads

```
node * p = head;
while (p)
{
    #pragma omp task
    {
        process (p) ;
    }
    p = p->next;
}
```

Profiling a Multi-threaded application



Workload imbalance
between the two
threads

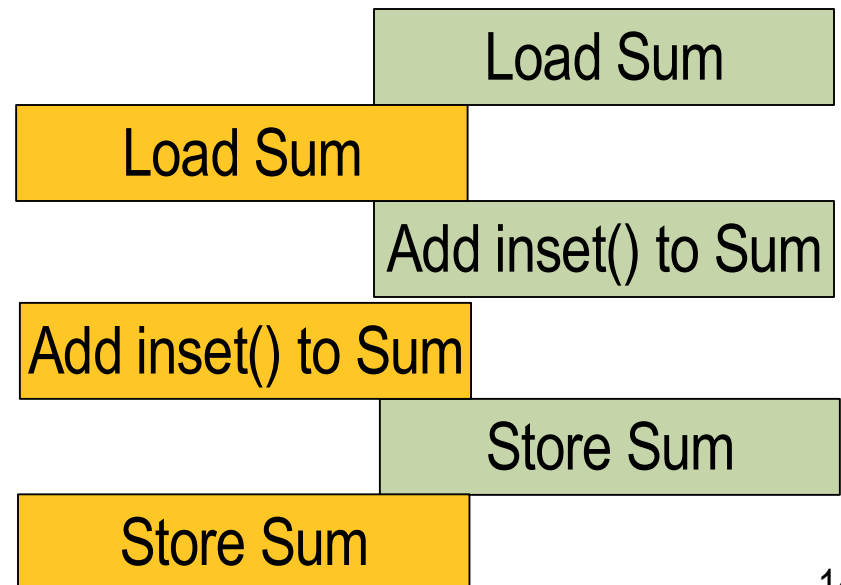
Data races cause incorrect results

- Multiple threads reading/writing the same data without exclusive access
- Results in unpredictable behaviour

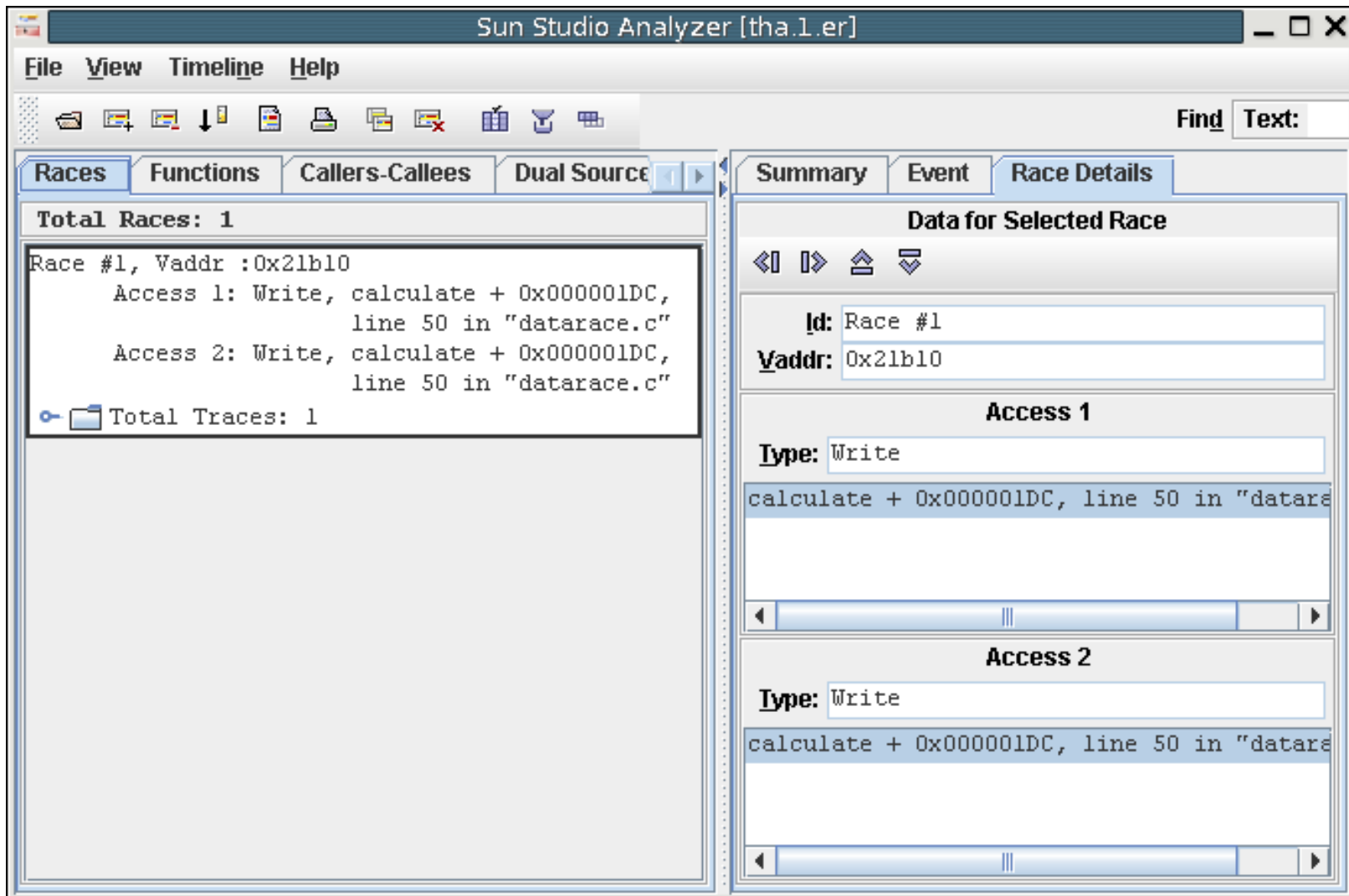
```
$ cc -fast -mt -o race race.c -lpthread
```

```
$ datarace
sum = 385158800
```

```
$ datarace
sum = 385071679
```



Thread Analyzer



Sun Studio Analyzer [tha.1.er]

File View Timeline Help

Races Functions Callers-Callees Dual Source

Summary Event Race Details

Total Races: 1

Race #1, Vaddr :0x21b10

- Access 1: Write, calculate + 0x000001DC, line 50 in "datarace.c"
- Access 2: Write, calculate + 0x000001DC, line 50 in "datarace.c"

Total Traces: 1

Data for Selected Race

Id: Race #1

Vaddr: 0x21b10

Access 1

Type: Write

calculate + 0x000001DC, line 50 in "datarace.c"

Access 2

Type: Write

calculate + 0x000001DC, line 50 in "datarace.c"

View source code

Sun Studio Analyzer [tha.l.er]

File View Timeline Help

Find Text:

Races Functions Callers-Callees **Dual Source** Source Disassembly Timeline Experiments

Race	Accesses	Source File: ./datarace.c	Object File: ./datarace	Load Object: <datarace>
0	46.	for (y=0; y<SIZE; y++)		
	47.	{		
0	48.	xv = ((double) (x-SIZE/2))/((double) (SIZE/4));		
0	49.	yv = ((double) (y-SIZE/2))/((double) (SIZE/4));		
2	50.	sum+=inset(xv,yv);		
	51.	}		
	52.	}		

Race	Accesses	Source File: ./datarace.c	Object File: ./datarace	Load Object: <datarace>
0	46.	for (y=0; y<SIZE; y++)		
	47.	{		
0	48.	xv = ((double) (x-SIZE/2))/((double) (SIZE/4));		
0	49.	yv = ((double) (y-SIZE/2))/((double) (SIZE/4));		
2	50.	sum+=inset(xv,yv);		
	51.	}		
	52.	}		

Fixing data accesses

- Single thread access:
 - > Mutex locks
 - > Critical regions (OpenMP)
- Lock-less:
 - > Atomic operations (`man atomic_ops`)
- Data sharing:
 - > Thread local storage
 - > OpenMP reduction directive

Resources

- Developer portal
 - > <http://developers.sun.com/>
- Forums
 - > <http://forum.java.sun.com/index.jspa?tab=devtools>

Summary

- Use recent compiler
- Profile application
- Optimise
 - > Improve algorithms
 - > Improve source clarity
 - > Improve compiler flags
- Parallelise

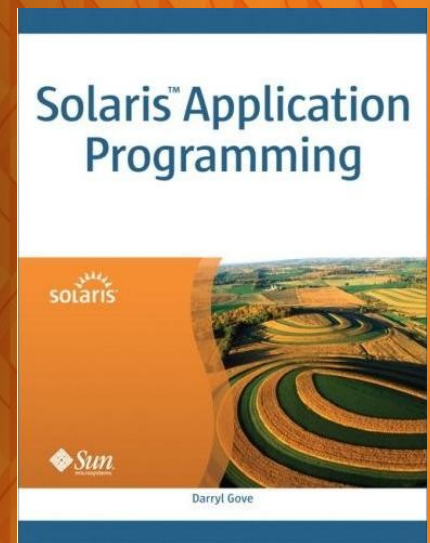


Compilers, Tools, and Performance

Darryl Gove

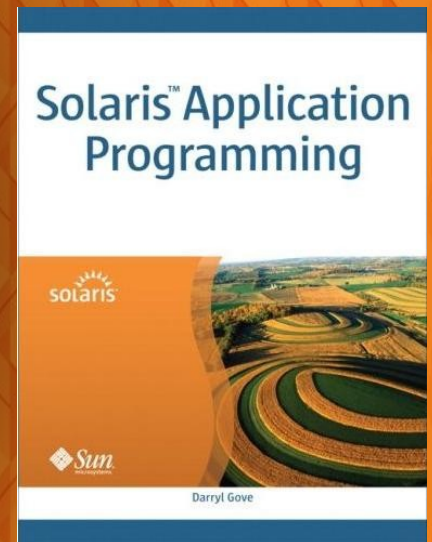
darryl.gove@sun.com

<http://blogs.sun.com/d/>





Backup Material



Target architecture

- General
 - > **-xtarget=generic**
- Needed to exploit features
 - > **-xtarget=sparcfmaf -fma=fused**
 - > **-xarch=sse2 -xvector=simd**
- Flags evaluated from left to right

32-bit or 64-bit

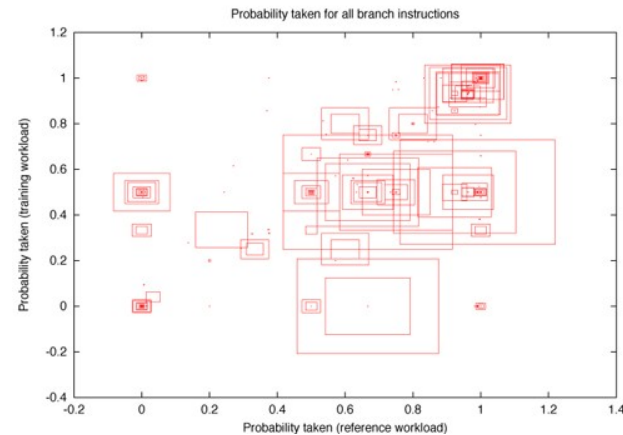
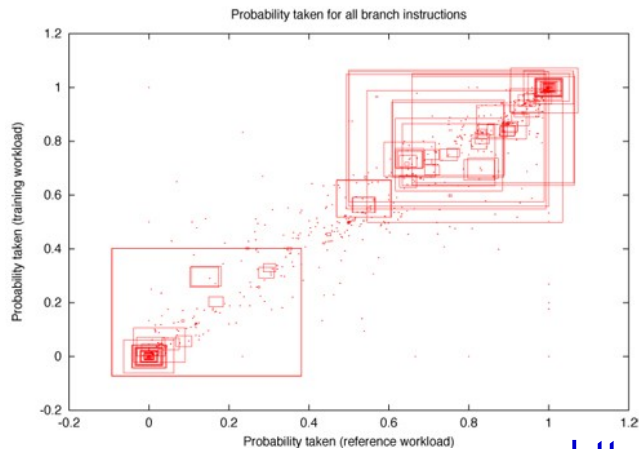
- Compiler flags: `-m32` | `-m64`
- 64-bit:
 - > Larger address space
 - > Pointers and longs 64-bits
 - > Larger memory footprint
 - > Lower performance
- EMT64
 - > Better ABI
 - > More registers
 - > Higher performance

Inlining

- Inlining
 - > Within file **-xO4**
 - > Across files **-xipo**
- Avoid cost of calling routine
- Expose further performance opportunities

Profile feedback

- Two compile passes (complicates build)
- Good for “branchy” code
- Helps inlining decisions
- Profile feedback
 - > **-xprofile=[collect:|use:]**



<http://developers.sun.com/solaris/articles/coverage.html>

Aliasing

- Compiler has to assume pointers alias
 - > Unless it can prove otherwise
 - > Or it is told to assume otherwise
- Specify degree of aliasing
 - > **-xalias_level=<level>**
- Specify pointers passed into functions don't alias
 - > **-xrestrict**
- Restrict qualify pointers
 - > **int * restrict p**