OpenSolaris: an ultimate development platform?

Roman Shaposhnik
Sun Studio Linux Architect, SDT
Sun Microsystems

Agenda (last chance of leaving):

- 14:30 – 16:00 (Roman Shaposhnik, Sun microsystems inc.)
  - Introduction
  - Exposé of OpenSolaris as a development platform
  - Sun Studio as the toolchain of choice for OpenSolaris
  - Toy Show
- 16:00 – 16:45 (Adriaan De Groot, Project KDE)
  - Developing in C++ with OpenSolaris and Sun Studio 12
- 16:45 – 17:30 (Dennis Chernoivanov, Tocarema AB)
  - Using Solaris to reach the next dimension of automated trading

Developers, developers, developers

- Microsoft got the wind of the problem in 2000-2001
  - Solaris, meanwhile, kept bragging about deployment
- Developers are a canary in the coal mine
  - What happened to AIX, HP-UX, Irix? Are they dead?
  - What happened to Apple?
- I have a problem with this conference's agenda:
  - Developing the plumbing for the platform
  - Developing the platform itself
  - Developing the eco-system for the platform
  - **Application development**

What kind of a developer are you?

- Segmented by market
  - Database, Enterprise, HPC, Web 2.0, Multimedia, Telco
- Segmented by technology
  - Languages, Tools, Software APIs, Hardware APIs
- What do they all have in common in 2008?
  - Postmodern software development practices
  - Total disregard for vendor lock-in
  - The line between software and hardware gets blurry
  - Open Standards/Protocols/Source
  - <u>Community Software Development</u>

What do developers want?

- This is the part where it gets interactive...
  - What DO you want? No, really?
- **Get the job done!**
- Future-proof their investments
  - Did you know that Perl is no longer The King?
- Have the most fun along the way
  - Community, community, community!
- Be as productive as possible
- Deliver results they can be proud of
  - This is mostly about performance

What makes a platform possible

- It has to be available
  - Virtualization is a kiss-of-death
  - Litmus test: autotools
- It has to be approachable
  - De-jure standards are important, GNU generally sucks, but...
- It has to run on reasonable hardware (and my toaster too!)
  - Linux kernel is a glorified device driver
  - I've got DSL – where's my DSS?
- It has to offer a very reasonable basic desktop
- Good news? OpenSolaris does it all and more!
  - http://www.opensolaris.org

What makes a platform interesting

- Support of the cool hardware
  - My TomTom runs Linux and I can hack it!
- Community, community, community
  - A case of Ubuntu: does techonology even matter?
  - Does corporate backing matter?
  - Where is my OpenSolaris developer's summit?
- And yes, technology does matter
  - Flexible, modular architecture: set of LEGO bricks
- **Development tools and practices**

What about being part of a bigger picture?

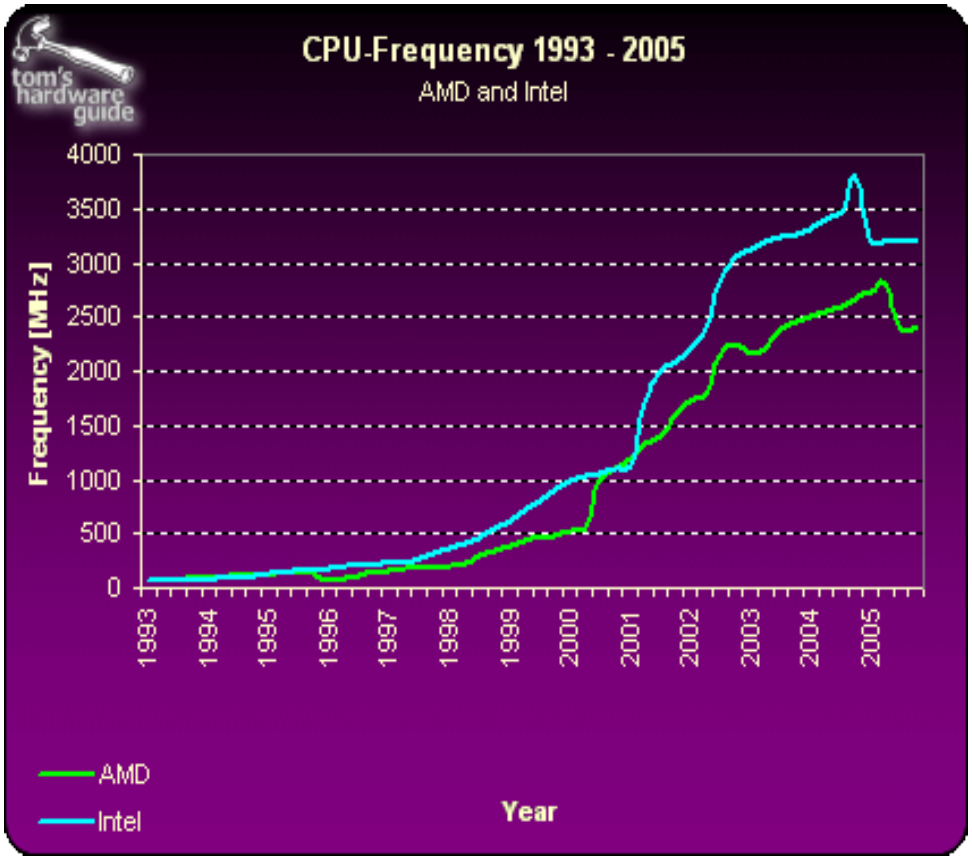
Sun CRS, Support, Architectural, Professional Services

Developer Tools	Sun Studio NetBeans	Open & Free
Cloud computing	Sun N1™ Grid Engine Software Network.com	Free
Operating System	solaris, suse, redhat	Solaris is Open & Free
Node Processor	64 Bit, AMD 64 Opteron, intel, ULTRASPARC	Open
Interconnect	Ethernet, Myrinet, InfiniBand Magnum switch (3456 ports)	Open

Developer's daily schedule:

- Coffe & Slashdot
- Design
  - What to leverage, what to build?
- Edit
  - Code as a dynamic, fluid medium
- Compile
  - Compiler is not a batch processing tool from the 70s
- Debug/Analyze
  - You can't comprehend the system: get the best intelligence

How to build future-proof systems?



Where's my 10GHz CPU?

- Between 1993 and 1999, the average CPU clock speed increased tenfold; since then, it hasn't even doubled
- Historical h/w approach to performance (increasing clock speed, pipelining, and cache) is being negated by heat, power consumption, slow memory

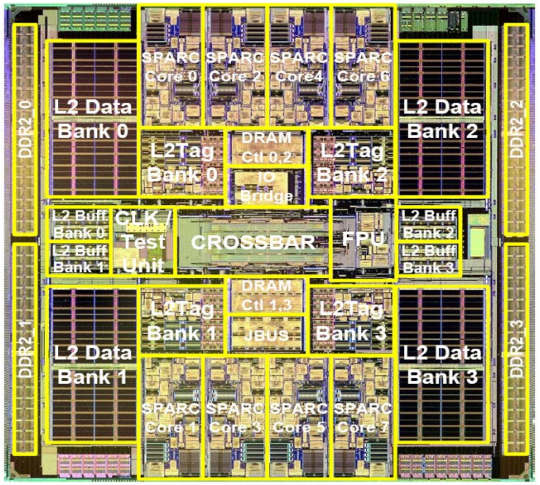
The Clock Race is Over !

Multi-Core Revolution
Putting transistors to work in a new way

UltraSPARC T2

Sun: 8 cores * 1.4GHz
(64 threads in a chip)

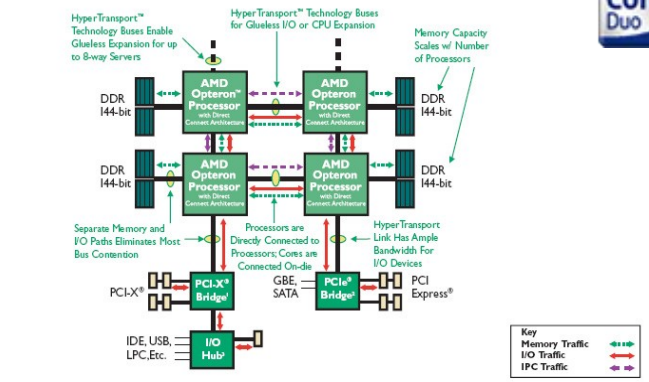
ULTRASPARC OpenSPARC



Intel: Clovertown, AMD: Barcelona

Intel: 4 cores * 2.66GHz
AMD: 4 cores * 2.0 GHz
(4 threads in a chip)

AMD 64 Opteron



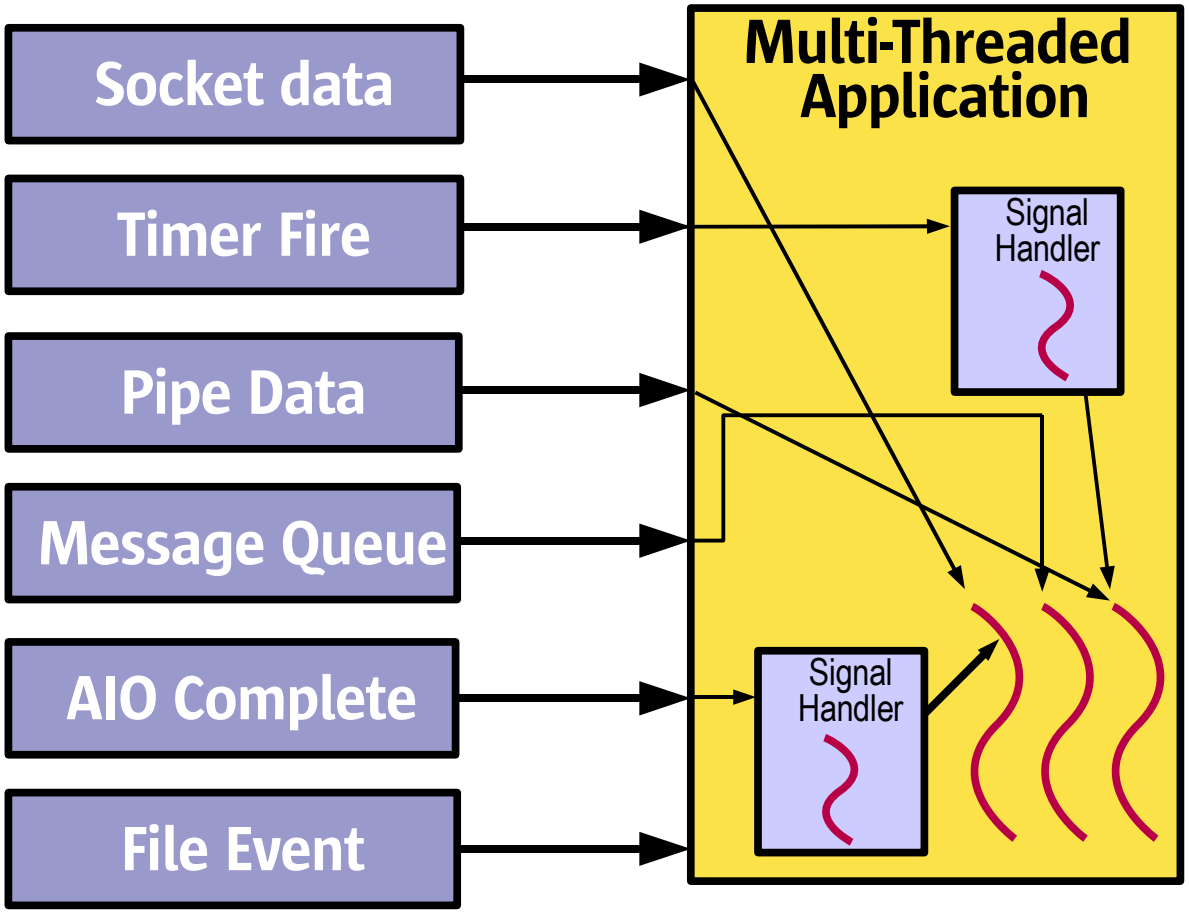
intel Core 2 Duo inside

Every new system is powered by a multi-core chip !

Design gets easier with Solaris:

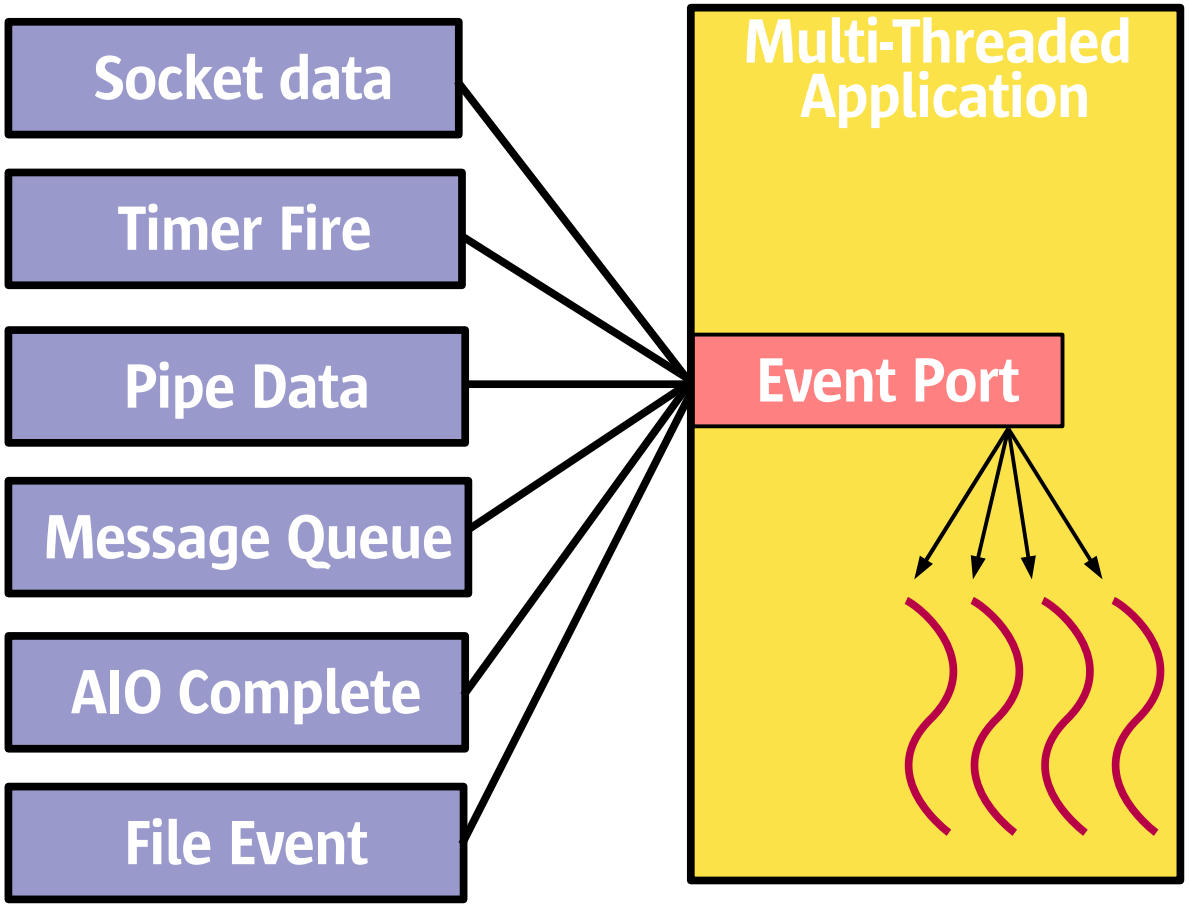
- The power of POSIX (developers shouldn't read man(1) pages):
  - http://www.opengroup.org/onlinepubs/009695399/
- LWP Scheduler: 128+ CPU/cores architectures
- libmtsk.so is part of the basic OS: Multi-core CPUs
  - Autopar, OpenMP, best overall system utilization
- Memory Management
  - Memory Placement Optimizations (locality groups)
  - Variable Page Support: ppgsz(1), mpss.so.1, memcntl(2)
  - libumem: MT-aware malloc(3) and free(3) (slab allocator) UMEM_LOGGING, UMEM_DEBUG
- Persistent data management with ZFS
  - 'F' == ZPL
- Event ports
  - Queue that multiplexes events from multiple sources

Before Event Ports



- aio_read(3RT)
- aio_write(3RT)
- lio_listio(3RT)
- timer_create(3RT)
- poll(2)
- poll(7D)
- select(3C)

With Event Ports



- `port_create()`
- `port_associate()`
- `port_get()`

Edit gets easier with Sun Studio

Sun Studio

File Edit View Navigate Source Refactor Build Run CVS Tools Advanced Window Help

Default

Projects | Files

- database.hpp
- datasource.hpp
- datetime.hpp
- except.hpp
- expr.hpp
- field.hpp
- mysql.hpp
- operations.hpp

Class View | Navigator - expr.hpp

- litesql-0.3.1_1
  - example
  - gen
  - litesql
    - And
    - Backend
    - Connective
    - Cursor<>
    - DataSource<>
    - Database
    - DatabaseError
    - Date

...pp | exampledatabase.cpp | cppgenerator.hpp | expr.hpp

asString() const

```
    virtual ~Connective() {}

    virtual string asString() const {
        string res = "(" + e1.asString() + ") " + op
            + " (" + e2.asString() + ")";
        return res;
    }
};
/** connects two expressions with and-operator. */
class And : public Connective {
public:
    And(const Expr & e1_, const Expr & e2_) : Connective("and", e1_
    virtual string asString() const {
        if (e1.asString() == "True")
            return e2.asString();
        else if (e2.asString() == "True")
            return e1.asString();
        else
            return Connective::asString();
```

55:15 INS

Compile with Sun Studio on Solaris

- The pickiest compiler ever
  - Especially relevant for C++
- If our compiler is that picky you should see our lint
  - /opt/SUNWspro/bin/lint
  - /opt/SUNWspro/bin/lock_lint
- More capable plumbing-level tools: ld(1), ld.so.1(1)
  - Thread Local Storage is way better supported
  - COMDAT is way better supported
  - Auditing and debugging interfaces
- And yes, we've got GNU toolchain as well
  - /usr/sfw/bin
  - http://cooltools.sunsource.net/gcc/

Compile with Sun Studio on Solaris

- Basic libraries part of the core OS:
  - libc.so, libC*.so, libm.so
- GNU libc and GCC: it takes two to tango

```
from /usr/include/features.h:

  _ISOC99_SOURCE       Extensions to ISO C89 from ISO C99.
  _POSIX_SOURCE        IEEE Std 1003.1.
  _POSIX_C_SOURCE      If ==1, like _POSIX_SOURCE; if >=2 add IEEE Std 1003.2;
                       if >=199309L, add IEEE Std 1003.1b-1993;
                       if >=199506L, add IEEE Std 1003.1c-1995;
                       if >=200112L, all of IEEE 1003.1-2004
  _XOPEN_SOURCE        Includes POSIX and XPG things.  Set to 500 if
                       Single Unix conformance is wanted, to 600 for the
                       upcoming sixth revision.
  _XOPEN_SOURCE_EXTENDED XPG things and X/Open Unix extensions.
  _LARGEFILE_SOURCE    Some more functions for correct standard I/O.
  _LARGEFILE64_SOURCE  Additional functionality from LFS for large files.

  _FILE_OFFSET_BITS=N  Select default filesystem interface.
  _BSD_SOURCE          ISO C, POSIX, and 4.3BSD things.
  _SVID_SOURCE         ISO C, POSIX, and SVID things.
  _GNU_SOURCE          All of the above, plus GNU extensions.
```

Compile with Sun Studio on Solaris

- DTrace USDT probes as an assert(3)/print-debug on steroids

```
$ cat test.c
#include <sys/sdt.h>
int main()
{
   DTRACE_PROBE(roman__test, TEST);
   return 0;
}
$ cc -c test.c
$ dis -F main test.o
    Drodata.rodata:          55                  pushl   %ebp
    0x1:                     8b ec               movl    %esp,%ebp
    0x3:                     83 ec 04            subl    $0x4,%esp
    0x6:                     e8 fc ff ff ff      call    -0x4      <0x7>
$ cat probe.d
provider roman__test {
        probe TEST();
};
$ dtrace -G -32 -s probe.d test.o
$ dis -F main test.o
    Drodata.rodata:          55                  pushl   %ebp
    0x1:                     8b ec               movl    %esp,%ebp
    0x3:                     83 ec 04            subl    $0x4,%esp
    0x6:                     90                  nop
    0x7:                     90                  nop
    0x8:                     90                  nop
    0x9:                     90                  nop
    0xa:                     90                  nop
$ cc -o test test.o probe.o
```

Debugging on Solaris: [k]mdb

- One debugger to rule them all (as frightening as it gets)

```
$ cat leak_soup.c
int main()
{
   void *p = malloc(100);
   return p = 0;
}

$ export LD_PRELOAD=libumem.so
$ export UMEM_DEBUG=default
$ /usr/bin/mdb ./leak_soup
> ::sysbp _exit
> ::run
mdb: stop on entry to _exit
mdb: target stopped at:
libc.so.1`exit+0x14:    ta        %icc, %g0 + 8
mdb: You've got symbols!
Loading modules: [ ld.so.1 libumem.so.1 libc.so.1 ]
> ::findleaks
CACHE     LEAKED   BUFCTL CALLER
0002d908       1 0003c1e0 main+4
------------------------------------------------------------------------
   Total       1 buffer, 112 bytes
> ::help
Each debugger command in mdb
 is structured as follows:

      [ address [, count]] verb [ arguments ... ]
> $q
```

Debugging on Solaris: dbx

- A gentler, softer scriptable debugger
- Comprehensive Debugger for Multithreaded development
  - Monitor thread entry point, PC, events, LWPs
  - Browse, select, view active threads
  - OpenMP
- Program control and data evaluation
  - Conditional breakpoints, postbreak modifiers, watchpoints
  - Trace program statements and variables
  - Call stack navigation
- RunTime Checking
  - Memory access violations
  - Memory usage and memory leaks
- Fix & Continue: recompile and patch running binaries

Debugging on Solaris: dbx (cont.)

- Speaking of F&C and walkers:

```
$ cat .dbxrc
function dig_it
{
    rm -f /tmp/__dig_it.so /tmp/__dig_it.c
    vim /tmp/__dig_it.c
    cc -G -g /tmp/__dig_it.c -o /tmp/__dig_it.so
    __DIG_IT=$[(void*)dlopen("/tmp/__dig_it.so", 2)]
    call __dig_it($@)
    call dlclose($__DIG_IT)
    lo -unload /tmp/__dig_it.so
}

$ dbx ./a.out
(dbx) stop in main
(dbx) run
stopped in main at line 3 in file "leak_soup.c"
(dbx) dig_it
void __dig_it()
{
   printf("Hello World\n");
}
"/tmp/__dig_it.c", line 3: warning: implicit function declaration: printf
dbx: warning: unknown language, 'c' assumed
Reading __dig_it.so
Hello World!
(dbx)
```

Debugging on Solaris: IDE

Sun Studio

File Edit View Navigate Source Build Run CVS Tools Advanced Window Help

Debug

Projects Files

maniac.cc
police.cc
sports.cc
traffic.cc
truck.cc
vehicle.cc
vehicle_list.cc

Class View

Freeway_4
FileWindow
FreewayWindow
FwyControlObjects
FwyMenuObjects
FwyZoneObjects

Start Page | main.cc | truck.cc

draw(Display* display, Drawable

```
void
Truck::draw(Display *display, Drawable pix, GC gc, int x, int y,
int direction_right, int scale, int xorg, int yorg, int selected) {
    extern unsigned long color_black;

    this->xloc(x);
    this->yloc(y);

    //
    // If I am heading to the right, then I need to draw brick to the left of
```

52:1 INS

Watches | Process I/O | Call Stack | Local Variables | Dbx Console

Function

Truck::draw(this = 0x80fbf10, display = 0x809b280, pix = 85983445U, gc = 0x80fb7c0, x = 0, y = 28, direction_right = 1, scale = 1, xorg = 0, yorg = 0, selected = 0)
draw_vehicles(pix = 85983445U, scale = 1, xoff = 0, yoff = 0)
traffic_display()

(3) stop at "truck.cc":52

Observaility on Solaris

- Key APIs are part of the core OS
  - Microstate accounting is on by default

```
$ grep LMS /usr/include/sys/msacct.h
#define LMS_USER        0       /* running in user mode */
#define LMS_SYSTEM      1       /* running in sys call or page fault */
#define LMS_TRAP        2       /* running in other trap */
#define LMS_TFAULT      3       /* asleep in user text page fault */
#define LMS_DFAULT      4       /* asleep in user data page fault */
#define LMS_KFAULT      5       /* asleep in kernel page fault */
#define LMS_USER_LOCK   6       /* asleep waiting for user-mode lock */
#define LMS_SLEEP       7       /* asleep for any other reason */
#define LMS_WAIT_CPU    8       /* waiting for CPU (latency) */
#define LMS_STOPPED     9       /* stopped (/proc, jobcontrol, lwp_suspend) */
```

  - Hardware Counters support (libcpc, cputrack, Dtrace, cpustat)

```
$ cputrack -fev -c IC_ref,Instr_cnt /bin/echo
   time    pid lwp      event      pic0      pic1
  0.006   3281   1   init_lwp         0         0
  0.012   3281   1   fini_lwp    136584    325953
```

  - DTrace

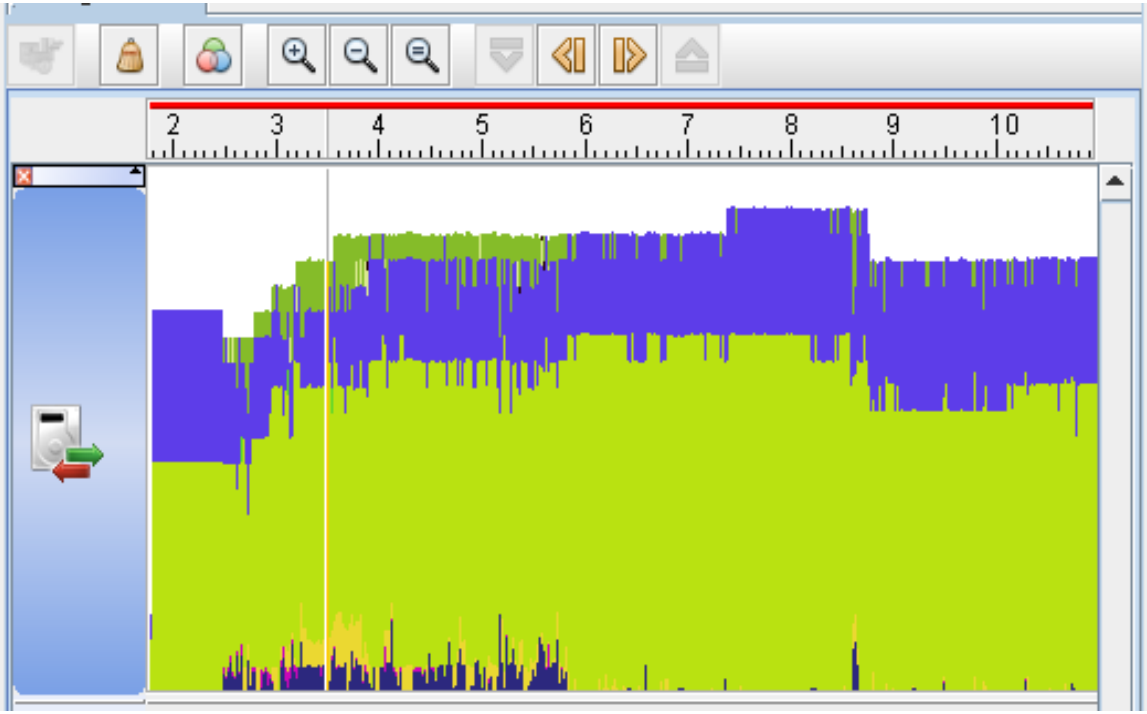
Microstate accounting: prstat(1)

```
$ /bin/prstat -am

   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/NLWP
  8799 roman    100 0.0 0.0 0.0 0.0 0.0 0.0 0.1   0 193   0   0 test/1
  8001 root     0.0 0.1 0.0 0.0 0.0  17  83 0.0  18   0 419   0 automountd/6
 27959 roman    0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  40   0 274   0 prstat/1
 15518 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  20   0  80  10 Xorg/1
 15840 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0  10   0  50   0 dtgreet/1
 13952 root     0.0 0.0 0.0 0.0 0.0  33  67 0.0   1   0  10   0 hald/3
 11700 roman    0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   4   0  28   0 sshd/1
  4103 daemon   0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   1   0   8   0 rpcbind/1
  7699 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   1   0  10   0 sendmail/1
  7816 noaccess 0.0 0.0 0.0 0.0 0.0  67  33 0.0  89   1  89   0 java/18
  8742 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   0   0   0   0 ttymon/1
  8793 root     0.0 0.0 0.0 0.0 0.0  64  36 0.0   0   0   0   0 syslogd/11
  8688 root     0.0 0.0 0.0 0.0 0.0 0.0 100 0.0   0   0   0   0 sac/1

 NPROC USERNAME  SWAP   RSS MEMORY      TIME  CPU
     4 roman    5220K 8404K   0.4%  49:37:40  50%
    40 root       69M   62M   3.0%   0:07:11 0.0%
     1 noaccess  116M   89M   4.3%   0:03:05 0.0%
     1 smmsp    1260K 1168K   0.1%   0:00:00 0.0%
     7 daemon     16M 5456K   0.3%   0:00:00 0.0%
     1 dladm     456K  652K   0.0%   0:00:00 0.0%


Total: 54 processes, 203 lwps, load averages: 1.00, 1.00, 1.00
```

Microstate accounting: D-Light

Microstate accounting Monitor

Time stamp	3.499801525
Stopped	0
Asleep for any other reason	2000
Asleep in user text page fault	0
Asleep in user data page fault	0
Aleep in kernel page fault	0
Asleep on semaphore (waiting for IO?)	0
Asleep on cond. variable (waiting for IO?)	3000
Asleep waiting for system-mode synchronization	0
Asleep waiting for user-mode synchronization	9985
Waiting for CPU (latency)	7
Running in other trap	0
Running in sys call or page fault	29
Running in user mode	1974



Hardware counters

- Theory of operation
  - a set of low-level registers in the CPU
  - set_value, get_value, receive_interrupt
  - each events (cache miss, etc.) increments it
- OS involvement
  - privilege level
  - virtualizing per process context
  - multiplexing virtual counters onto physical ones
- HW involvement
  - no-skid policy
- Different views
  - Holistic view: /usr/sbin/cpustat
  - Per process (LWP) view: /usr/bin/cputrack

Hardware counters: libcpc, DTrace

- Check the versions and accessibility of the hardware performance counters with cpc_version() and cpc_access()
- Initialize a cpc_event_t data structure, using cpc_getcpuver() and cpc_strtoevent()
- Bind the data structure to the CPU using cpc_bind_event()
- Read the counters, using cpc_take_sample()
- Release the CPU using cpc_rele()

- Dtrace cpc provider is in the works:

```
$ dtrace -qn 'cpc:::DC_dtlb_L1_miss_L2_miss-u-1000 { @[execname] = count(); }'
```

Hardware counters: Sun Studio

Sun Studio Analyzer [matmul.er]

File View Timeline Help

Find Text:

Functions | Callers-Callees | Source | Lines | Disassembly | Experiments

Source File: matmul.java
Object File: matmul
Load Object: <JAVA_CLASSES>

CPU Cycles (sec.)	D$ Misses	DTLB Misses		
0,	0	0	1.	class matmul
				<Function: matmul.<init>>
			2.	{
			3.	public static void main(String args[])
			4.	{
			5.	int x,y,k;
			6.	
0,	0	0	7.	for(k=0; k<1000; k++)
				<Function: matmul.main(java.lang.String[])>
0,	0	0	8.	for(y=0; y<1000; y++)
0,	0	0	9.	for(x=0; x<1000; x++)
73,396	2 007 829 170	1 200 038	10.	matrix[x][y] *= 2;
0,	0	0	11.	}
			12.	
			13.	int x,y;
0,012	100 006	0	14.	static double [][] matrix = new double[1000][1000];
				<Function: matmul.<clinit>()>
			15.	}

DTrace: exposing the VMs

```
$ cat js_funcflow.d
dtrace:::BEGIN
{
        depth = 0;
        printf("%s %-20s  %-22s   -- %s\n", "C", "TIME", "FILE", "FUNC");
}

javascript*:::function-entry
{
        depth++;
        printf("%d %-20Y  %-22s %*s-> %s\n", cpu, walltimestamp,
            basename(copyinstr(arg0)), depth*2, "", copyinstr(arg2));
}

javascript*:::function-return
{
        printf("%d %-20Y  %-22s %*s<- %s\n", cpu, walltimestamp,
            basename(copyinstr(arg0)), depth*2, "", copyinstr(arg2));
        depth--;
}

$ ./js_funcflow.d
C TIME                  FILE                    -- FUNC
0 2007 Mar 26 16:42:09  clock.html              -> startTime
0 2007 Mar 26 16:42:09  clock.html                -> getHours
0 2007 Mar 26 16:42:09  clock.html                <- getHours
```

Procfs is not like the Linux one

- Binary oriented
  - requires special tools (p* tools)
- Allows for some really cool hacks
  - PCWATCH

```
typedef struct prwatch {
        uintptr_t pr_vaddr;  /* virtual address of watched area */
        size_t pr_size;      /* size of watched area in bytes */
        int pr_wflags;       /* watch type flags */
}
```

  - PCAGENT

```
$ ls /proc/11701

contracts/  cwd@        root@       fd/         path/       object/     lwp/
as          cred        priv        status      lstatus     lusage      usage
map         rmap        xmap        psinfo      lpsinfo     auxv        ldt
sigact      pagedata    watch       ctl

$ ls /proc/11701/lwp/1
lwpctl      lwpsinfo    lwpstatus   lwpusage    templates/  xregs
```

Procfs related p* tools

- pgrep, pargs, pfiles, pmap, pstack, pflags, pcred, pldd, psig, pwdx, pstop, prun, pwait, ptime

```
$ pfiles 11701
11701:  /home/rs76089/bin/intel-S2/bash
   3: S_IFDOOR mode:0444 dev:321,0 ino:39 uid:0 gid:0 size:0
      O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[7845]
      /var/run/name_service_door

$ pmap 11701
11701:  /home/rs76089/bin/intel-S2/bash
08045000      12K rwx--    [ stack ]
08048000     920K r-x--  /home/rs76089/bin/intel-S2/bash
0812E000     128K rwx--  /home/rs76089/bin/intel-S2/bash
0814E000    1628K rwx--    [ heap ]
FED60000      64K rwx--    [ anon ]
FED90000    1228K r-x--  /lib/libc.so.1

$ pstack 11701/1
11701:  /home/rs76089/bin/intel-S2/bash
 fee41697 waitid   (7, 0, 8047bb0, 7)
 fedfcdcb waitpid  (ffffffff, 8047c8c, 4) + 63
 0808a85d ???????? (3b43, 1)
 0808960e wait_for (3b43) + 17e
 080793db execute_command_internal (8234a4c, 0, ffffffff, ffffffff, 82d02ec) + 8e3
 08078846 execute_command (8234a4c) + 4e
 0806aa00 reader_loop (fef70018, 0, 0, 0, 0, 1) + 1e8
 080686be main     (1, 8047ddc, 8047de4) + 85e
 08067d7c ???????? (1, 8047e90, 0, 8047eb0, 8047ebd, 8047ecd)
```

Procfs related p* tools (contd.)

```
$ pstop 11701

$ pflags -r 11701
11701:  /home/rs76089/bin/intel-S2/bash
        data model =  ILP32  flags = ORPHAN|MSACCT|MSFORK
 /1:    flags = STOPPED|ISTOP|ASLEEP  read(0x0,0x80473fb,0x1)
        why = PR_REQUESTED
    %gs = 0x000001C3   %fs = 0x00000000   %es = 0x0000004B   %ds = 0x0000004B
   %edi = 0x08047E28  %esi = 0xFED82A00  %ebp = 0x080473E4  %esp = 0x00000000
   %ebx = 0xFEED3000  %edx = 0x00000000  %ecx = 0xFEE411A7  %eax = 0x00000003
%trapno = 0x0000000E  %err = 0x00000007  %eip = 0xFEE411A7   %cs = 0x00000043
   %efl = 0x00000246 %uesp = 0x080473C8   %ss = 0x0000004B
$ prun 11701

$ pldd 11701
11701:  /home/rs76089/bin/intel-S2/bash
/lib/libsocket.so.1
/lib/libnsl.so.1
/lib/libdl.so.1
/lib/libc.so.1

$ gcore  11701
$ pflags -r core.11701
core 'core.11701' of 11701:     /home/rs76089/bin/intel-S2/bash
        data model =  ILP32  flags = ORPHAN|RLC|MSACCT|MSFORK
 /1:    flags = STOPPED|ISTOP|ASLEEP  waitid(0x7,0x0,0x8047bb0,0x7)
        why = PR_REQUESTED
        sigmask = 0x00020000,0x00000000
    %gs = 0x000001C3    %fs = 0x00000000   %es = 0x0000004B   %ds = 0x0000004B
   %edi = 0x00000007   %esi = 0xFED82A00  %ebp = 0x08047B90  %esp = 0x00000000
   %ebx = 0xFEED3000   %edx = 0x00000001  %ecx = 0xFEE41697  %eax = 0x0000006B
%trapno = 0x0000000E   %err = 0x00000007  %eip = 0xFEE41697   %cs = 0x00000043
   %efl = 0x00000246  %uesp = 0x08047B70   %ss = 0x0000004B
```

Uniform system statistics: *stat

- Sampling tools
  - *stat [sampling interval [times to sample]]
- vmstat (the highest level of system sampling)

```
$ vmstat 1
vmstat 1
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr s0 s1 -- --   in   sy   cs us sy id
 0 0 0 708532 853236 80 1171 24 1 14  0 64  1 -0  0  0 4593 5111  587 28  3 68
 0 0 0 569480 828476 15  59  0  0  0  0  0  0  0  0  0  440  383  270 50  0 50
```

- mpstat (drilling down into the compute cycles)

```
$ mpstat 1
mpstat 1
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  579   1 2190  2271  470  265   33   16   85    0  2918   37   4   0  60
  1  591   2 1582  2318    2  322   37   16   85    0  2189   20   3   0  77
```

- iostat (drilling down into the I/O cycles)

```
$ iostat 1
iostat 1
   tty        sd0           sd1          nfs236       nfs1181          cpu
 tin tout kps tps serv  kps tps serv  kps tps serv  kps tps serv   us sy wt id
   0   11   5   1   10    0   0    0    0   0    7    0   0    4   28  3  0 68
```

*stat loose ends

- cpustat

```
$ cpustat -c EC_ref,EC_misses 1 3

         time cpu event      pic0      pic1
        1.008   0  tick     69284      1647
        1.008   1  tick     43284      1175
        2.008   0  tick    179576      1834
```

- busstat (not really what you think it is)

```
$ busstat -a -w ac0,pic0=mem_bank0_stall,pic1=mem_bank1_stall 10
     time  dev   event0            pic0   event1            pic1
     10    ac0   mem_bank0_stall   1234   mem_bank1_stall    5678
     20    ac0   mem_bank0_stall   5678   mem_bank1_stall   12345
```

- kstat (poor man's sysfs)

```
$ kstat
module: unix                          instance: 0
name:   fpu_traps                     class:    misc
        fpu_ieee_traps                0
        fpu_unfinished_traps          0
```

Sun Studio Compilers and Tools

C/C++ Developer's Needs

Performance

- Take advantage of new hardware features to write *fast* apps

Parallelism

- Multi-core is here !

Productivity

- IDE is important for writing apps *faster*

(Open)Platforms

- Solaris and Linux
- SPARC, x64, x86

C/C++ Developer's Need ...

to overcome Significant Challenges

Performance

- Take advantage of new hardware features to write *fast* apps
- Architectures are changing (too?) fast
- New tricks are needed

Parallelism

- Multi-core is here!
- Incredibly hard to parallelize serial apps
- New approaches / tools are needed

Productivity

- IDE is important for writing apps faster
- Poor satisfaction with C, C++ IDEs
- Lack of advanced tools

(Open)Platforms

- Solaris and Linux
- SPARC, x64, x86
- Evolving, incompatible gcc interfaces on Linux
- No uniformity in Linux platforms

Sun Studio 12

- Maximizes Application Performance
- Simplifies Multi-core Development
- Improves productivity with a modern IDE
- Single source for Linux and Solaris; SPARC and x86

iPPPP: Performance. Parallelism. Platforms. Productivity.

Main Sun Studio Components

- C, C++, Fortran compilers
- Rogue Wave Tools.h++, STLport libraries
- Advanced Math libraries: Sun Performance Library
- IDE (based on NetBeans)
- Command line scriptable debugger (dbx)
- Graphical debugger and Run Time Checker (RTC)
- Performance Analysis Tools
- Thread Analyzer (NEW!)
- Software build utility ("distributed make")
- Third-party Source Distributions

Sun Performance Math Library

- Numerical routines optimized for performance, including:
  - LAPACK v3.0
  - BLAS 1, 2, 3
  - Netlib Sparse BLAS
  - NST Fortran Sparse BLAS v0.5
  - Fast Fourier Transform (FFT) routines
  - Convolution and correlation routines
- Support for single- or multi-processor systems
- Hotspot tuning for BLAS routines
- Routines are optimized for various SPARC chips, X86 and AMD64 platforms

Performance

Write Fast Applications

Maximize Application Performance

- Sun compilers continue World Record Performance tradition
  - **World Records in EACH category**: SPECint 2006, SPECfp 2006, SPECintrate 2006, SPECfprate 2006 and SPEC OMP2001
  - **World Records on each architecture from 1 core/1socket to 128 cores/64 sockets(scaling)**: UltraSPARC T2 (Niagara2), UltraSPARC-IV+, SPARC64 VI systems , Intel/Core2, AMD/Opteron
  - **Sun SPARC Enterprise M9000 system tops 1-TeraFLOP barrier**
- Significant lead over GCC
  - 18% -52% on SPARC (SPEC2006)
  - 11% -18% on x86/AMD (SPEC2006)
  - 70% + on STREAM

New compilers make significant difference over older releases as well as competitors

Compiler Options for Performance

- **-xO1 thru -xO5** (default is no opt, -O implies -xO3)
- **-fast:** easy to use, best performance on most code, but it assumes compile platform = run platform and makes FP arithmetic simplifications
- **Understand program behavior and assert to optimizer:**
  - **-xrestrict:** if only restricted pointers are passed to functions
  - **-xalias_level:** if pointers behave in certain ways
  - **-fsimple:** if FP arithmetic can be simplified
- **Target machine-related:**
  - **-xprefetch, -xprefetch_level**
  - **-xtarget=, -xarch=, -xcache=, -xchip=**
  - **-xvector:** converts DO loops into vector instr/calls

Compiler Options for Performance

- **Advanced Compiler options**
  - **-xprofile:** profile-feedback enabled optimizations
  - **-xcrossfile, -xipo:** performs crossfile/interprocedural optimizations
  - **-xautopar:** enable automatic parallelization
  - **-xdepend:** performance dependence analysis
- **Use optimized math libraries**
  - **Sun Performance library for algebraic functions**
  - **Vectorized math routines (libmvec)**
  - **Inline (libmil) and optimized math (libmopt)**
  - **Value-added math library (libsunmath)**

Source Code Changes

- **Improve usage of data cache, TLB, register windows**
  - Use VIS instruction (templates) directly (via -xvis)
  - Optimize data alignment (also: #pragma align)
  - Prevent Register Window Overflow
- **Creating inline assembly templates for performance critical routines**
- **Loop Optimizations that compilers may miss:**
  - Prevent Register Window Overflow
  - Restructuring for pipelining and prefetching
  - Loop splitting/fission
  - Loop Peeling
  - Loop interchange
  - Loop unrolling and tiling
  - Pragma directed

Gains from Tuning Categories

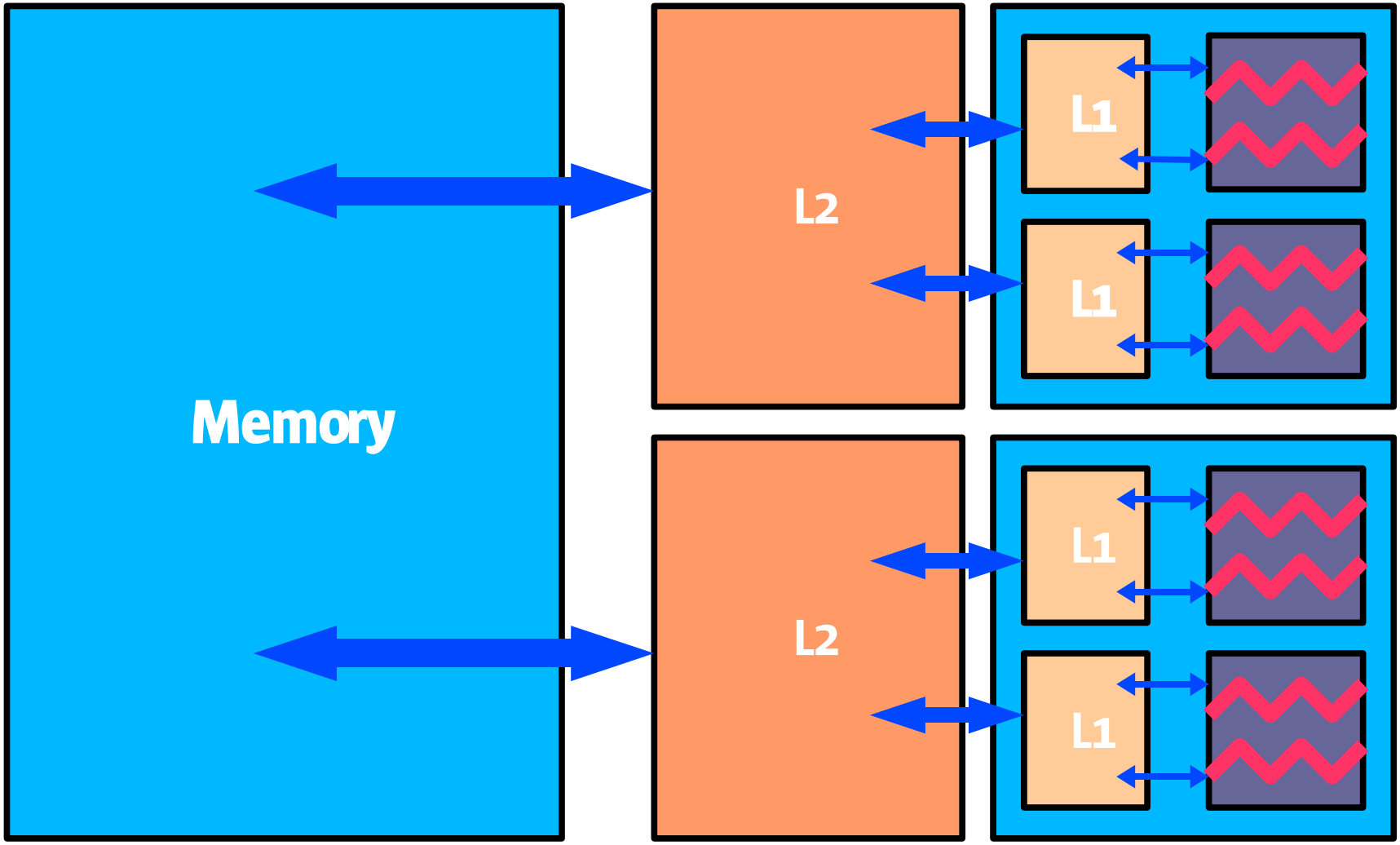
Tuning Category	Typical Range of Gain
Source Change	*25-100%*
Compiler Flags	*5-20%*
Use of libraries	*25-200%*
Assembly coding / tweaking	*5-20%*
Manual prefetching	*5-30%*
TLB thrashing/cache	*20-100%*
Using vis/inlines/micro-vectorization	*100-200%*

Reasonable default values

-fast

-xO5 -xarch=sse2 -xchip=opteron -xcache=64/64/2:1024/64/16
-xalias_level=basic -xbuiltin=%all -xvector=simd
-fsimple=2 -fsingle -fns -nofstore
-dalign -xdepend -xlibmil -xlibmopt -xipo=0
-xprefetch=auto -xregs=frameptr

Memory access kills performance



Type based alias disambiguation

```
static int a;
static float f;
void foo(void) {
    int i = get_int();
    int *alias_to_a = &a;
    float *fp = get_pfloat(*alias_to_a);
    *alias_to_a = i;// Isn't it like assigning a?
    *fp = 10; // Could possibly alias *p
    printf("%d\n", *alias_to_a);
}
```

```
$cc -O t.c -S -xarch=amd64
$cc -O t.c -S -xalias_level=basic -xarch=amd64
```

```
call get_int                call get_int
movl %eax,%ebx              movl %eax,%ebx              # %ebx holds i
movl a(%rip),%edi           movl a(%rip),%edi           # %edi holds copy of a
call get_pfloat             call get_pfloat
movl %ebx,a(%rip)           movl %ebx,a(%rip)           # assigning t to a
movl $1092616192,(%rax)     movl $1092616192,(%rax)     # assigning 10 to fp
leaq .L186(%rip),%rdi       leaq .L186(%rip),%rdi       # loading "%d\n"
movl %ebx,%esi            | movl a(%rip),%esi          # loading a
xorl %eax,%eax              xorl %eax,%eax
call printf                 call printf
```

Type based alias disambiguation

```
// for finer aliasing control
#pragma alias_level std (int_ptr_t)
#pragma alias (int, float)
#pragma alias(p, q)
#pragma noalias(int, struct S)
#pragma noalias(p, q)
#pragma may_point_to(p, a, b, c)
#pragma may_not_point_to(p, a, b, c)

C99:

int *restrict foo;
```

```
struct pair {
    int a;
    int b;
};

struct pair * decompose(int *p)
{
    return (struct pair *)p;
}
```

```
$ lint t.c -Xalias_level=strong
(8) error: cast of scalar pointer to struct pointer
lint: errors in t.c; no output created
lint: pass2 not run - errors in t.c
```

Performance Analyzer - Source Window

File View Timeline Help

Find Text:

Functions | Callers-Callees | Source | Lines | Disassembly | PCs | Timeline | LeakList | Statistics | Experiments

Source File: /mnt/home/ruudp/SunTune/new_codes/
Object File: /mnt/home/ruudp/SunTune/new_codes/
Load Object: <simulate.base.exe>

User CPU (sec.)	User CPU (%)	User CPU (sec.)	Sys. CPU (sec.)		
1.321	0.1	1.321	0.	346.	for(i = 1; i <= nstate; i
2.942	0.1	2.942	0.	347.	itch_n[i].r = itc
				348.	rho_i = rho[i];
5.574	0.3	5.574	0.	349.	for(j = 1; j <= nstate; j++) {
					Loop below scheduled with steady-state cycle count = 30
					Loop below unrolled 2 times
					Loop below has 10 loads, 2 stores, 2 prefetches, 4 FPadds, 4 FPmuls, and 0 FPdivs per iterat
308.296	15.7	308.296	0.120	350.	itch_n[i].r += rho_i[j].r * U_n[j].r + rho_i[j].i * U_
228.370	11.6	228.370	0.150	351.	itch_n[i].i += - rho_i[j].r * U_n[j].i + rho_i[j].i *
				352.	}
				353.	}
				354.	}
				355.	
0.560	0.0	0.560	0.	356.	for(i = 1; i <= nstate; i++) {
					-state cycle count = 2
					, 2 prefetches, 0 FPadds, 0 FPmuls, and 0 FPdivs per iterati

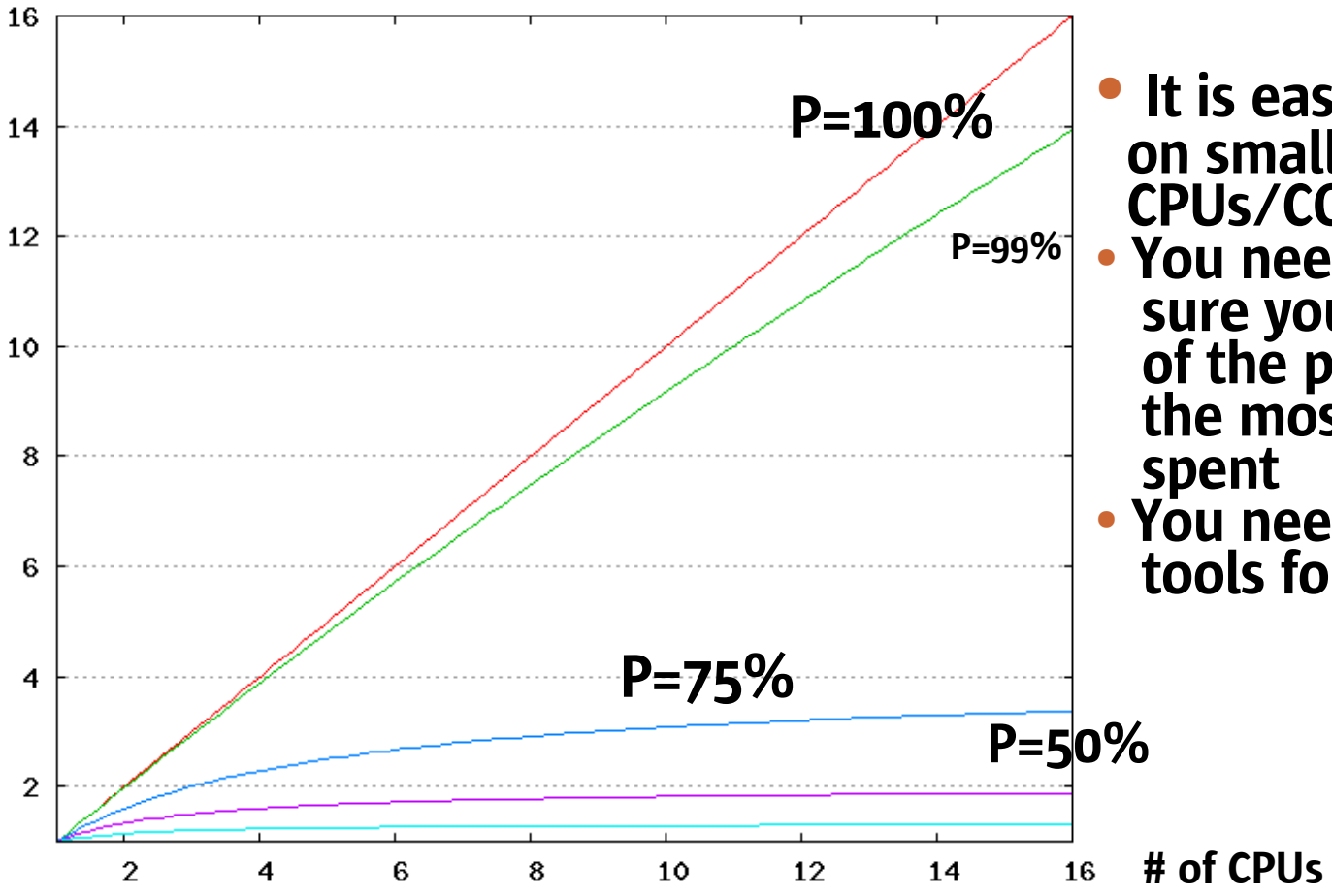
Compiler Commentary

most expensive statements

Parallelism:

Developing for a Multi-core future

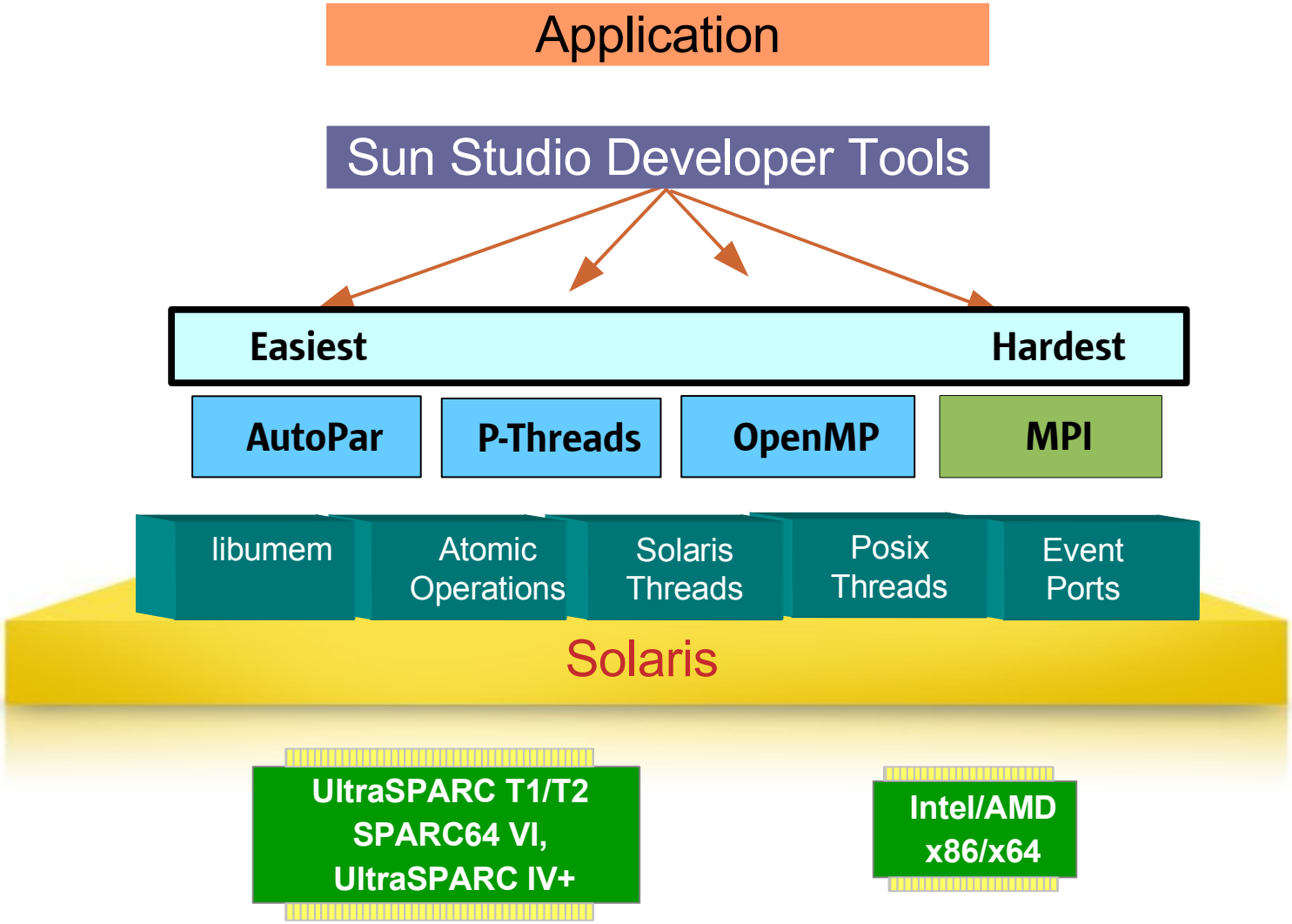
Amdahl's Law



- It is easy to scale on small number of CPUs/COREs
- You need to make sure you take care of the part where the most time gets spent
- You need the right tools for this job

$$SPEEDup(N) = \frac{1}{\left(\frac{P}{N} + 1 - P\right)}$$

Compiler Support for Parallel Apps



Parallelize Loops with *AutoPar*

Let the compiler create threads for you

```
for (i=0; i<m; i++)
  for (j=0; j<n; j++)
    a[i][j] = b[i][j] + c[i][j];
```

Turns on auto-parallelization

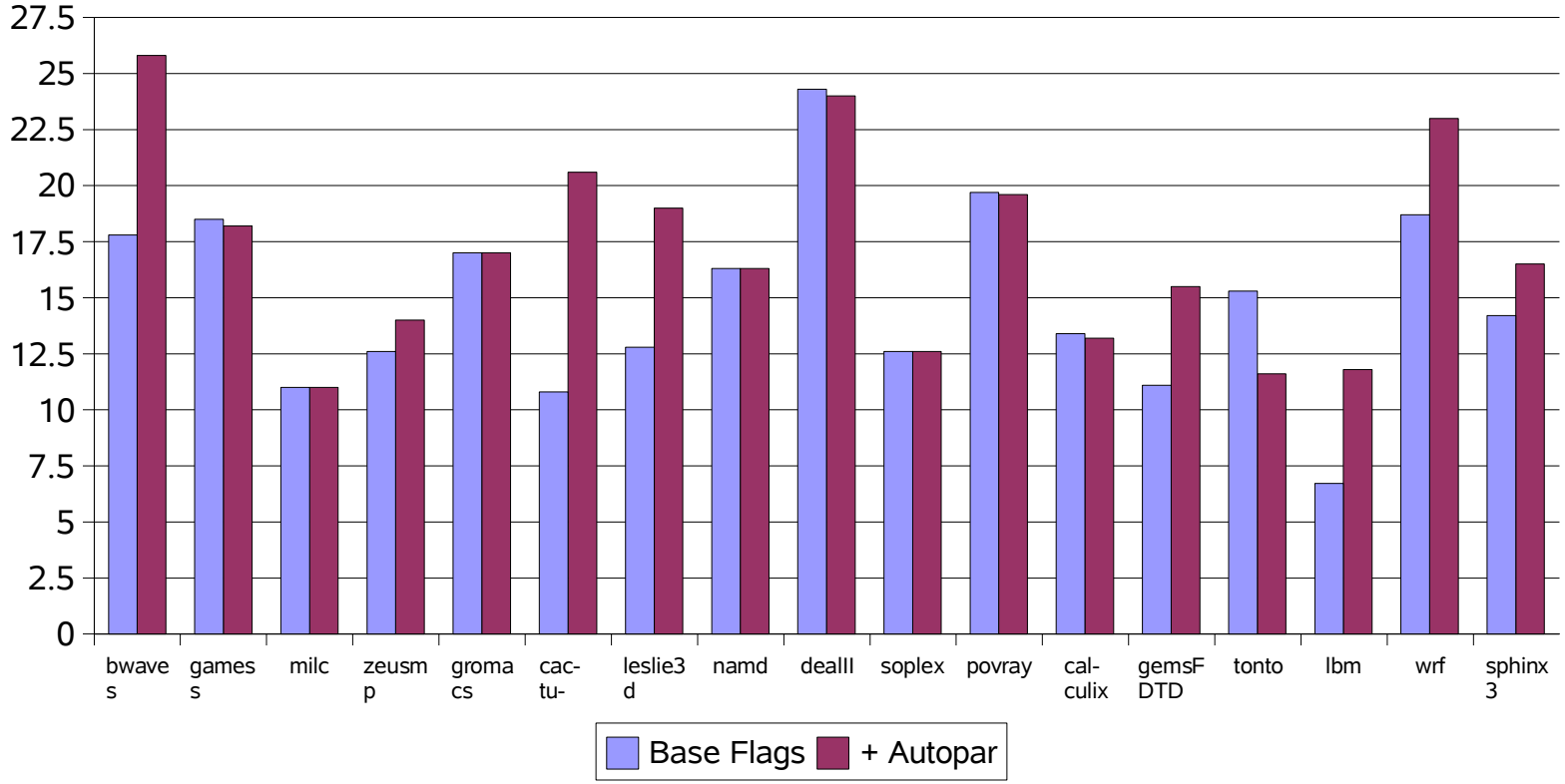
Prints parallelization information

```
$ cc -xO4 -xautopar -xloopinfo main.o loop.c
"loop.c", line 7: PARALLELIZED, and serial version generated
"loop.c", line 8: not parallelized, not profitable
$ time a.out                    // Default is 1 thread

real     0m37.96s
user     0m36.64s
sys      0m0.97s
$ PARALLEL=2                    // For a two-way parallel run
$ time a.out

real     0m21.99s
user     0m41.07s
sys      0m2.23s
```

The same **binary can run with different numbers of threads.**

Autopar: SPECfp 2006 improvements

3.0GHz dual-core Woodcrest machine
PARALLEL=2
Overall Gain: 16%



Parallelize Instructions with *Vector*

Do more work in each clock cycle

- **Compiler can generate instructions to simultaneously process multiple data elements in adjacent memory (technique called *Single Instruction, Multiple Data)***
  - > Compiler option: *-xvector=simd*
- **Experience shows gains in the 1-7% range**
- **Best suited for array processing via loops**

You write this:

```
for (i=0; i<1024; i++)
    c[i] = a[i] * b[i]
```

→

The compiler generates code like this:

```
for (i=0; i<1024; i+=4)
    c[i:i+3] = a[i:i+3] * b[i:i+3]
```

What is OpenMP

- **Language extensions for writing shared-memory parallel applications in C, C++ and Fortran**
- **Consists of**
  - Compiler directives (pragmas)
  - Runtime routines (libmtsk)
  - Environment variables
- **Advantages:**
  - Incremental parallelization of source code
  - Small(er) amount of programming effort
  - Good performance and scalability
  - Portable across variety of vendor compilers
- **Sun Studio 12 supports latest version (v2.5)**

OpenMP: defacto standard for creating parallel applications, supported by all major vendors

Parallelize Loops with *OpenMP*

Parallelization directives to create Threads

```
for (i=0; i<n; i++)
  a[index[i]] = a[i] + 2;
```

autopar flag used but not effective; potential unsafe dependency

```
$ cc -xO4 -xautopar -xloopinfo loop.c
 loop.c", line 8: not parallelized, unsafe dependence(a)
```

```
#pragma omp parallel for shared(n,a) private (i)
for (i=0; i<n; i++)
   a[index[i]] = a[i] + 2;
```

pragma added and openmp flag used; loop now parallelized

```
$ cc -xO4 -xopenmp -xloopinfo loop.c
 loop.c", line 9: PARALLELIZED, user pragma used

$ OMP_NUM_THREADS = 4  //Controls number of threads to create
$ a.out
```

An OpenMP Example

- Find the primes up to 3,000,000 (216816)

  Run on Sun Fire 6800, Solaris 9, 24 processors 1.2GHz US-III+, with 9.8GB main memory

Model	# threads	Time (secs)	% change
Serial	N/A	6.636	Base
OpenMP	1	7.210	8.65% drop
	2	3.771	1.76x faster
	4	1.988	3.34x faster
	8	1.090	6.09x faster
	16	0.638	10.40x faster
	20	0.550	12.06x faster
	24	0.931	Saturation drop

Race Conditions – Tough Parallel Issues

```
for i=0; i<n; i++)
  a[[i]] = a[i+1] + b[i];
```

Thread 1 writes interactions 0 – 5

a[0] = a[1] + b[0];

a[1] = a[2] + b[1];

a[2] = a[3] + b[2];

a[3] = a[4] + b[3];

a[4] = a[5] + b[4];

Thread 2 writes interations 5 – 9

a[5] = a[6] + b[5];

a[6] = a[7] + b[6];

a[7] = a[8] + b[7];

a[8] = a[9] + b[8];

a[9] = a[10] + b[9];

Parallel execution leads to non-deterministic results:
a[5] could be written by Thread 2 before its read by Thread 1

This is a Data Race condition

Studio Tools to Detect Race Conditions

- **Thread Analyzer detects data races and deadlocks**
  - Identifies non-deterministic or incorrect execution
  - Identifies actual and potential deadlock situations
  - **Process:**
    - Compile source code with -xinstrument=datarace
    - Run app and collect runtime info with *collect -r all*
    - Run tha to graphically display races and conflicts
  - **Works with OpenMP, Pthreads, Solaris Threads**
- **lock_lint static source code lock analyzer**
  - Analyzes mutex and multiple readers / single writer locks
  - Reports on inconsistent usage of locks that may lead to data races and deadlocks

A True SPEC Story

SPECOMP Benchmark fma3d

101 source files; 61,000 lines of Fortran code

Data race in platq.f90 caused sporadic core dumps

It took several engineers and 6 weeks of work to find the data race manually

Perils of Having a DataRace Condition

Program exhibits non-deterministic behavior

Failure may be hard to reproduce

Program may continue to execute, leading to failure in unrelated code

A data race is hard to detect using conventional debugging methods and tools

How did Thread Analyzer help?

SPECOMP Benchmark fma3d

101 source files; 61,000 lines of Fortran code

Data race in platq.f90 caused sporadic core dumps

It took several engineers and 6 weeks of work to find the data race manually

With the Sun Studio Thread Analyzer, the data race was detected in just a few hours!

Studio Tools to Tune MT Apps

- Performance Analyzer - multi-thread aware
  - Works with unmodified binaries, low overhead
  - Offers performance data at statement, instruction, routine level
  - Displays Compiler Commentary describing optimizations
  - Supports Posix threaded or OpenMP parallelization and MPI
  - Easy to use GUI
  - DataSpace Profiling and hardware counter data
  - Supports C, C++, Fortran and Java

Platforms

Unifying Solaris and Linux development

Sun Studio for Linux

- **C, C++, Fortran compilers and libraries**
  - Popular G++ and GCC extensions, including
    - asm_inlines, __attribute__
    - g++ABI for interoperability
    - Linux kernel compiled with Sun Compilers
  - Standard C++ libraries
  - Optimized Math libraries, including SunPerfLib
  - OpenMP 2.5, MPI libraries
  - No GPL lock-in. Be it real or imaginary.
- **Dbx, Performance and Thread analysis tools**
- **4000+ downloads, 800+ active users, 700+ messages on Sun Studio on-line community forum**

Sun Studio for Linux offers developers the same tools, same features, same source, as on Solaris

Sun Studio for what Linux?

- **Officially supported:**
  - >SuSE Linux Enterprise Server 9
  - >RedHat Enterprise Linux
- **Tested on:**
  - >OpenSuSE 10+
  - >RedHat Fedora 5
- **Played with on:**
  - >Gentoo, Ubuntu, CentOS, Scientific Linux
- **Fine print:**
  - >RPM based distributions (although tars are also available)
  - >Installs in /opt/sun/sunstudio
  - >kernel 2.6+
  - >glibc 2.3-2.4 (needs patching for 2.5)

Success stories

- **KDE4**
  - A huge C++ application, runs on most UNIXes
  - Dependency on external C++ libraries: Qt, etc.
  - Now available on Solaris (built with Sun Studio 12, BOOST 1.34.1, QT 4.3.1 and the Apache Standard C++ Library)
  - "Sun Studio is picky and we love it!" (case in point Exiv2)
- **tBricks trading platform (http://www.tbricks.com)**
  - Innovative start-up company
  - Lots of C++
  - Lots of dependencies on external code
  - Betting on multi-core
  - Keeping their platform options open
- **We have many, many more: TACC, University of Aachen**

Productivity

Build applications faster

Integrated Graphical Environment

- Based on NetBeans open source IDE
- Debugger and Performance Analyzer GUIs
- Code editor with syntax highlighting and code folding
- Compile error hyperlinks to source code lines
- Wizard for creating makefiles
- GUI layout editor / designer with X-Designer
- Highly configurable

Architecture of IDE

- Architecture
  - NetBeans layers: core, C/C++ Development Pack
  - IDE language model infrastructure – ANTLR and grammar
  - Source file discovery wizard
  - Compiler, make and debugger tool selection wizard
  - Interface to Sun Studio components
    - Compilers
    - Dbx through the dbxgui layer
    - Analysis through advanced tools interface
  - Interfacing to the application source
    - Project framework

Performance Analyzer

- Uses statistical callstack sampling
  - Clock-based
- Hardware counter support
  - UltraSPARC: US III, US IIIi, US III Cu, US IV
  - AMD Opteron
  - Some Intel processors
- Synchronization wait tracing
- Offers top to bottom performance data:
  - Routine, Statement and Instruction level
  - Plus Callers-Callees information
- All this information can be obtained in a single run !

Performance Analyzer/2

- Analyze multiple experiments
  - Data is aggregated
  - Can select/de-select experiments
- Analyzer (GUI) and er_print (CLI) settings can be saved in a default configuration file (.er.rc):
  - Current directory and/or
  - Home directory
- API to instrument your code manually (C, Fortran, C++)
- Insert markers in your application
  - Pause/resume functionality
  - Check the man page of libcollector for details

Clock-based Dataspace Profiling

- US-T1 does not support HWC profiling
  - (Only instructions – no memory-based counters)
- Workaround: clock-based dataspace profiling
  - `collect -p +on`
    - Backtracks one instruction; if a memop, record event
    - Max. Mem Stall metric from those events
      - May be deceptive – it's a <u>maximum</u>
        - Cache <u>hits</u> in tight loop will have high metrics
      - But with care, it can be very useful
    - As with HWC Dataspace profiling, compile `-xhwcprof`

Index Objects

Sun Studio Analyzer [test.1.er]

File View Timeline Help

Find Text:

Callers-Callees | Source/Disassembly | Source | Disassembly | Timeline | Experiments | Threads | Cpus

Display Mode: Text Graphical

User CPU (sec.)	Name
44.261	<Total>
22.256	CPU 0
22.005	CPU 1

Summary | Event

Selected Object:

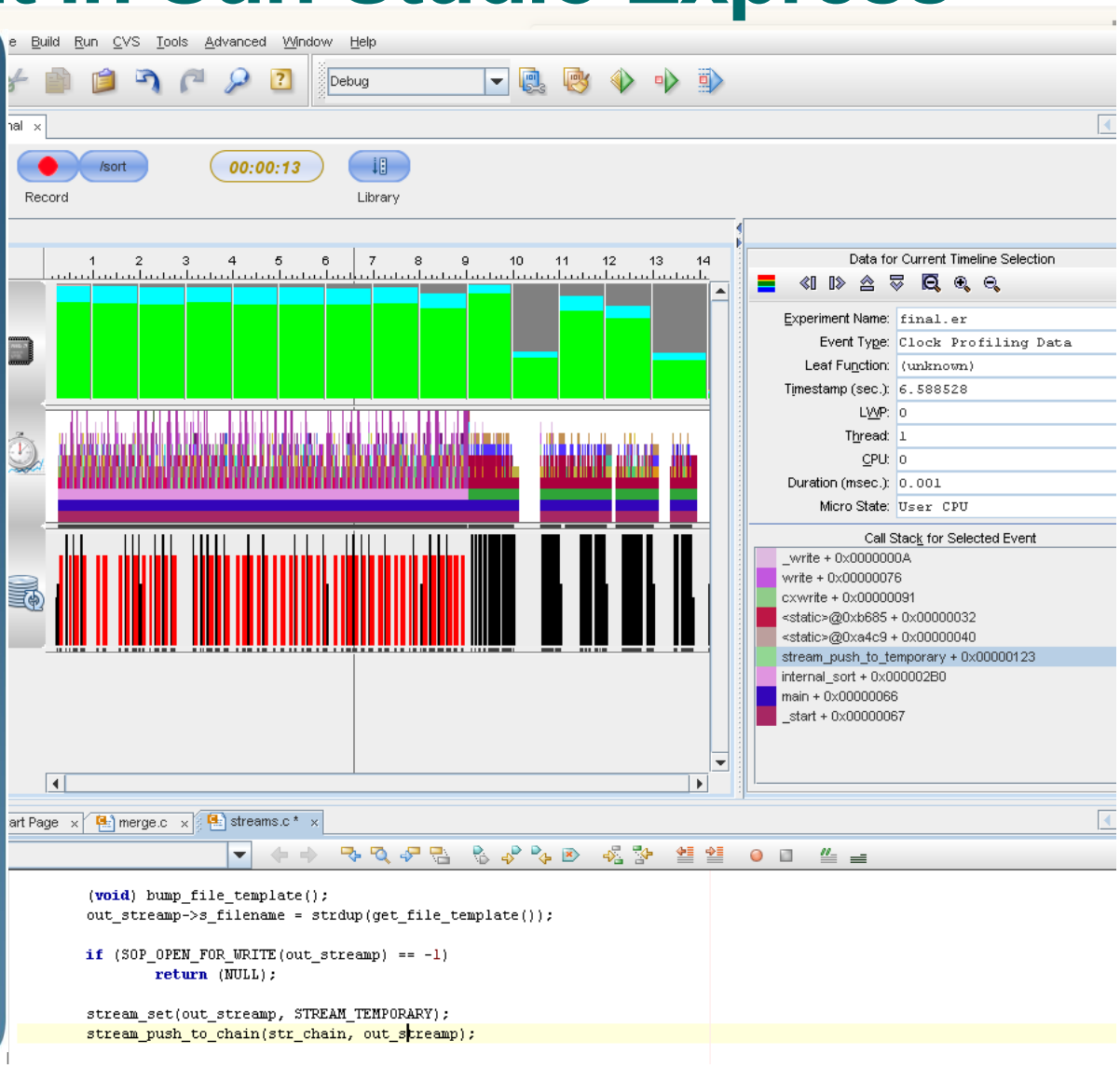
Index Object: CPU 0

Metrics for Selected Object:

	Exclusive
User CPU:	22.256 (50.28%)
Wall:	22.266 (50.24%)
Total LWP:	22.266 (50.24%)
System CPU:	0. (0. %)
Wait CPU:	0.010 (100.00%)
User Lock:	0. (0. %)
Text Page Fault:	0. (0. %)
Data Page Fault:	0. (0. %)
Other Wait:	0. (0. %)
OpenMP Work:	22.266 (50.24%)
OpenMP Wait:	0. (0. %)

Project D-Light in Sun Studio Express

- First tool to unify, and make accessible, sophisticated application and system profiling for developers
- Demonstrated first at JavaOne San Francisco, May 11, during James Gosling keynote
- Initial preview version now available in the Sun Studio Express program
- Utilize and visualize dynamic tracing (DTrace) technology
- Simple, interactive, drag and drop interface



Run Time Checking

Sun Studio

File Edit View Navigate Source Refactor Build Run CVS Tools Advanced Window Help

demo_rui

Proj... Files Favorites Runtime

- dbx76
- dlopen
- DynamicWatch
- Freeway2
- JumpyBpts
- **RtcDemo**
- Watch
- Welcome

Breakpoints

Name	Enabled	Context
overload.cc:88	✓	

...cc | overload.cc | dbxmain.cc [r/o] | dbxmain.cc [r/o] | rtcdemo.cc

make_and_print()

```
98  }
99
100 static Node *bigList = NULL;
101
102 static void
103 createBigList()
104 {
105     for (int i = 0; i < 10000; i++)
106         bigList = new Node("big", bigList, NULL);
107 }
108
109 static void
110 make_and_print()
111 {
112     Node *na = new Node("a");
113     Node *nb = new Node("b");
114     Node *n1 = new Node("+", na, nb);
115
116     Node *nc = new Node("c");
117     Node *nd = new Node("d");
118     Node *n2 = new Node("+", nc, nd);
119
120     Node *root = new Node("root", n1, n2);
121
122     root->print();
```

112:1 INS

Process I/O | Local Variables | Runtime Checking

Show New Activity | Show All Activity | Clear All Reports | Status: memuse leaks

Access Errors | Memory Usage | Memory Leaks

```
Access checking started for program
        /home/ivan/SunStudioProjects/RtcDemo/dist/demo_rui/Sun
_____________________________________________________________
Read from uninitialized (rui):
Attempting to read 4 bytes at address 0x4862c
    which is 4 bytes into a heap block of size 140 bytes at 0x
This block was allocated from:
        operator new<-make_and_print<-main
Location of error: rtcdemo.cc, line 86, Node::print_help()
```

Call Stack | Dbx Console

```
access checking - ON
(dbx-0) check -memuse -frames 8 -match 3
memuse checking - ON
(dbx-0) ### setOption DBX_rtc_biu_at_exit = 'on'
(dbx-0) ### setOption DBX_rtc_mel_at_exit = 'on'
(dbx-0) run
Running: rtcdemo -r
(process id 7475)
Reading rtcapihook.so
Reading rtcaudit.so
Reading libmapmalloc.so.1
Reading libgen.so.1
Reading rtcboot.so
Reading librtc.so
RTC: Enabling Error Checking...
RTC: Running program...
(dbx-7475)
```

Running

Sun Developer Community and Training

Join the Sun Developer Community

SDN membership gives you exclusive benefits:

- Free developer tools
- Discounts for training, support, books, and hardware
- Access to technical content from SunTech Days and JavaOne Online
- Participation in forums

http://developers.sun.com

Developer Services

- Need help?
  - > Developer email support for Solaris Developer Express, Sun Studio, Java, and Java developer tools available
- Also:
  - > Sun Developer Service Plans for Small to Medium Size Businesses
  - > Java Multi-Platform support for Enterprise developers and deployments

http://developers.sun.com/services

Sun Studio 13 and beyond

- **Open distribution**
  - >OpenSolaris and Project Indiana
  - >Ubuntu, RedHat, Novell/SuSE repositories
- **Open development**
  - >Open Sourcing of Sun Studio
  - >Much wider community participation
- **Open standards**
  - >De Jure: C++ 2009, OpenMP ARB,
  - >De facto: G++ ABI, BOOST, Apache STDC++, GCC, Linux
- **Bridging communities and enabling platforms**
  - >Cross-pollination of Linux and Solaris development
  - >network.com and Project O'Malley
- **“Our only religion is developers”**

Short-term roadmap

- **Maximal performance optimization**
  - Support for UltraSPARC T2, Rock, OPL systems
  - Continue world-record performance on AMD and Intel
- **New features and improvements**
  - OpenMP 3.0
  - MPI Profiler and Debugger
  - Transactional memory support (SW, HW)
  - ClusterTools 8.0 and Sun Grid Engine Plugin
- **Simpler, developer-centric IDE, debugging and profiling**
  - IDE based on modern NetBeans 6.0
  - Project D-Light
  - Thread analyzer and Performance analyzer improvements
- **Sun Studio developer kit for HPC**

Thank You !

Roman Shaposhnik
Sun Studio Linux Architect, SDT
Sun Microsystems

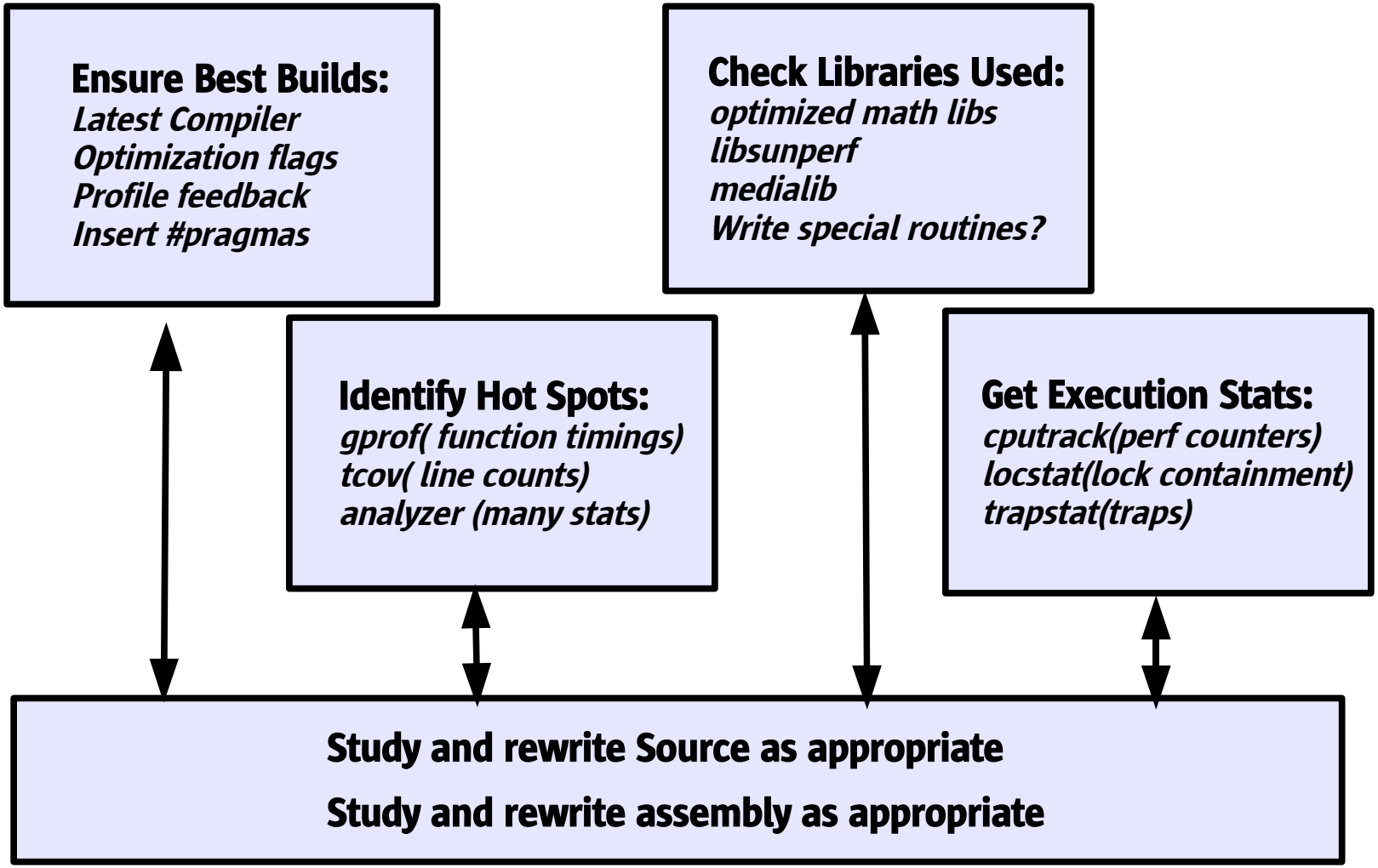
Sun Studio

Performance Tuning CookBook

Experiences from Tunathons ...

- Program run to understand application performance, instead of focusing on standard benchmarks
- Between 40 - 80 ISV or performance critical applications are considered for tuning and analysis
- Goals are to speedup app, identify compiler enhancements, and feedback for future system designs
- Opportunities range from:
  - Simple: find the best option, upgrade to new compiler
  - Easy: simple source change, found by simple analysis
  - Moderate: use of several analyzers, rewrites in assembly
  - Difficult: Complex analysis+tuning

Methodology / Tools Used



Changes that impact App Performance

1) Trading some behavior to get speed
2) Exploiting knowledge of the deployment environment
3) Exploiting knowledge of program characteristics
4) Source code changes

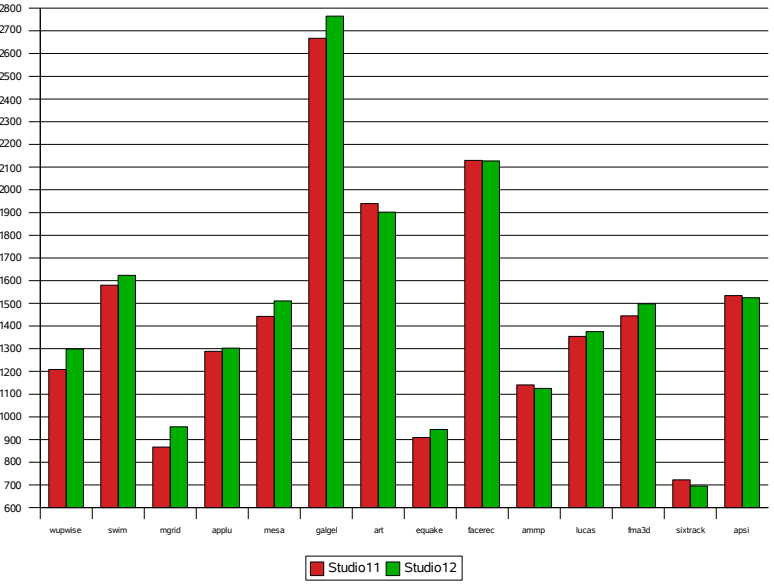
Sun Studio 12: Performance Characteristics

Performance Summary

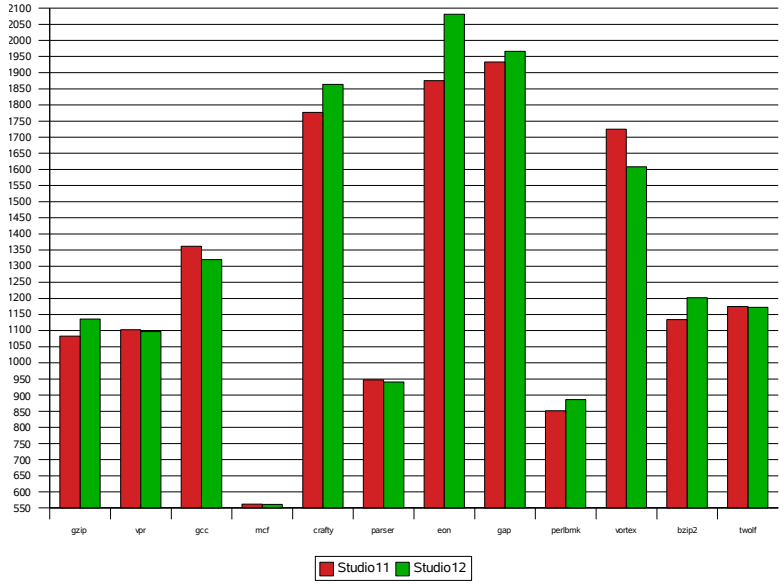
- SunStudio 12 improves over SunStudio 11: (x86)
  - 15% improvement on SPECint2006 (64bit mode)
  - 10% improvement on SPECfp2006 (64bit mode)
  - 17% improvement on SPECint2006 (32bit mode)
  - 20% improvement on SPECfp2006 (32bit mode)
- SunStudio 12 improves over SunStudio 11: (SPARC)
  - 12% improvement on SPECint2006 (UltraSPARC IIIi)
- SunStudio 12 faster than GCC4.1: (x86)
  - 10% better on SPECint2006 (64bit), 16% (32bit)
  - 16% better on SPECfp2006 (64bit), 20% (32bit)
  - On STREAM: 65% better (64bit), 80% better (32bit)
- SunStudio 12 faster than GCC4.1: SPARC
  - 10% on SPECint2006, 80% on SPECfp2006
  - 94% better on STREAM
- SunStudio 12 is better than GCC4.1 on Fortran applications:
  - Over 55% on SPARC
  - Over 26% better on x86/x64

Sun Studio Compilers and Tools

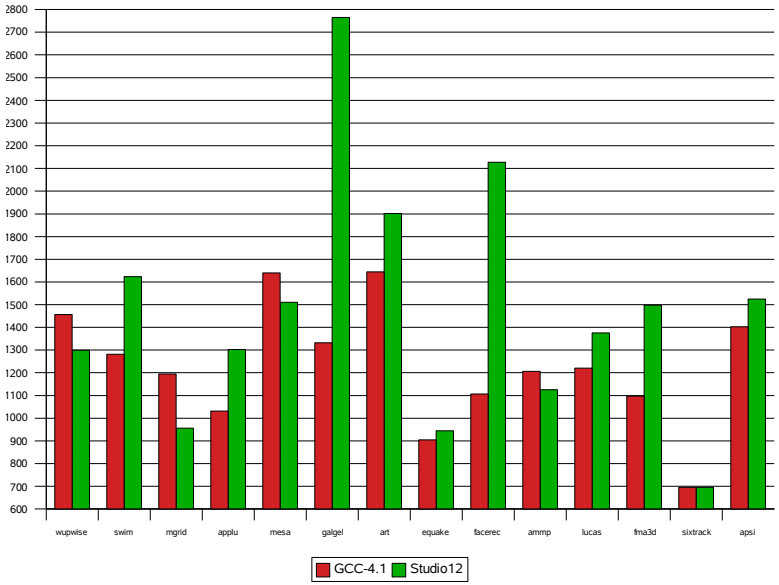
CFP2000 Sun Studio11 vs. Studio12 (Rates) Opteron 2.4 GHz



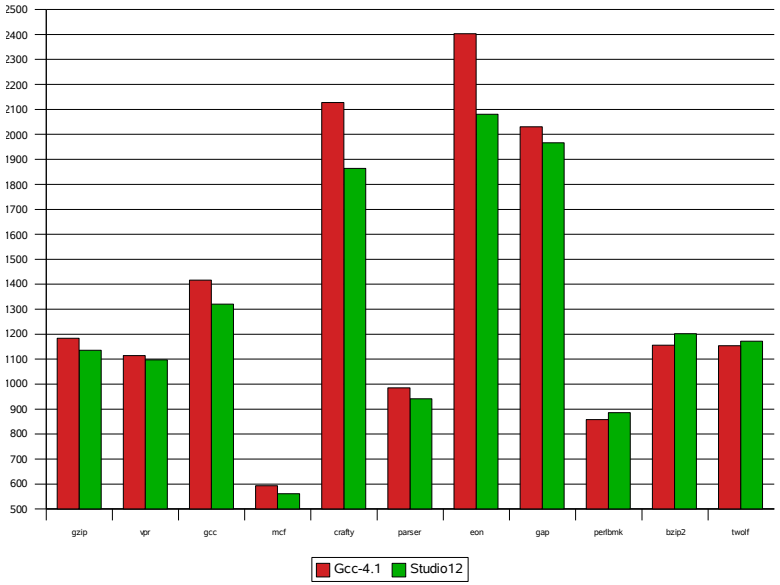
CINT2000 Sun Studio11 vs. Studio12 (Rates) Opteron 2.4 GHz



CFP2000 GCC-4.1 vs. Sun Studio12 (Rates) Opteron 2.4 GHz

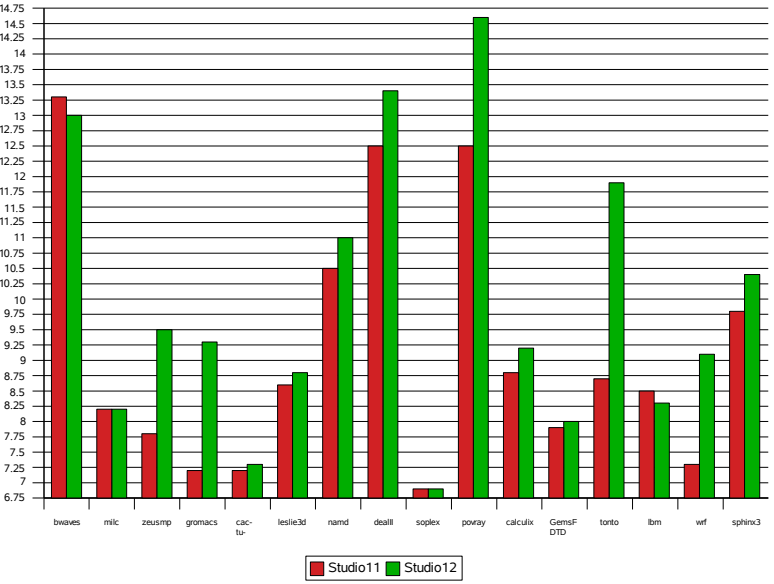


CINT2000 GCC-4.1 vs. Sun Studio12 (Rates) Opteron 2.4.GHz

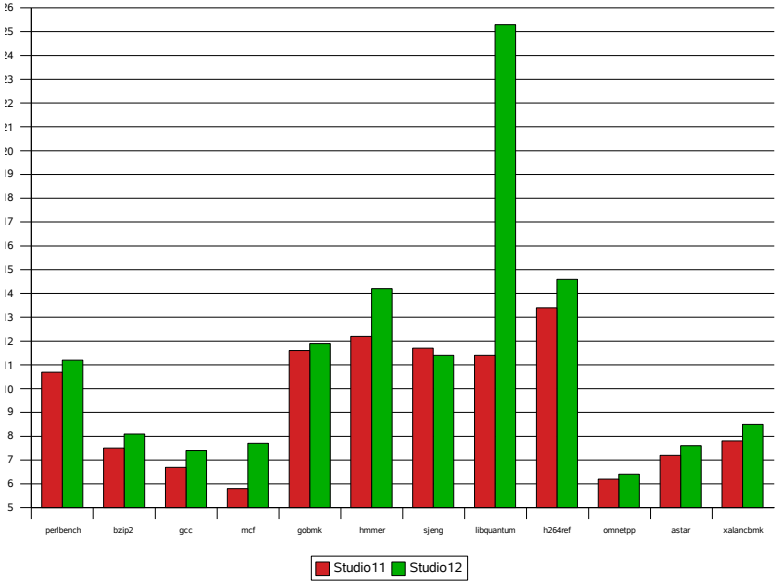


Sun Studio Compilers and Tools

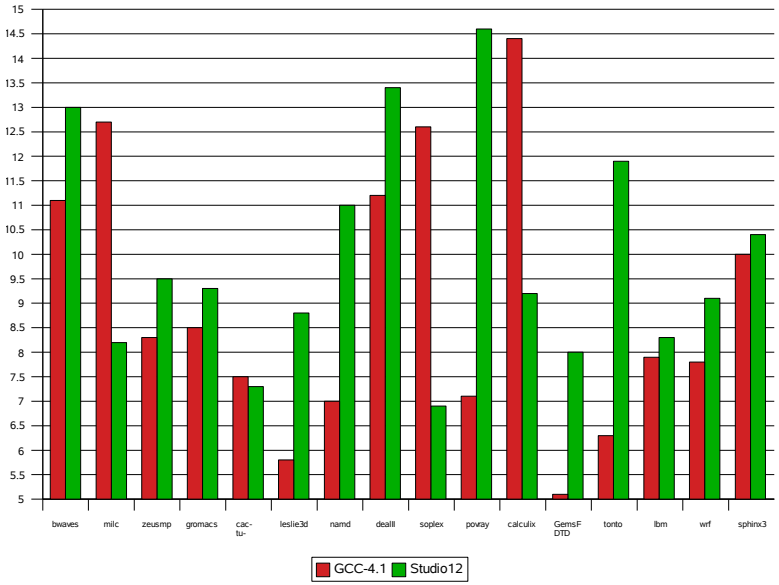
CFP2006 Sun Studio11 vs. Studio12 (Rates) Opteron 2.4 GHz



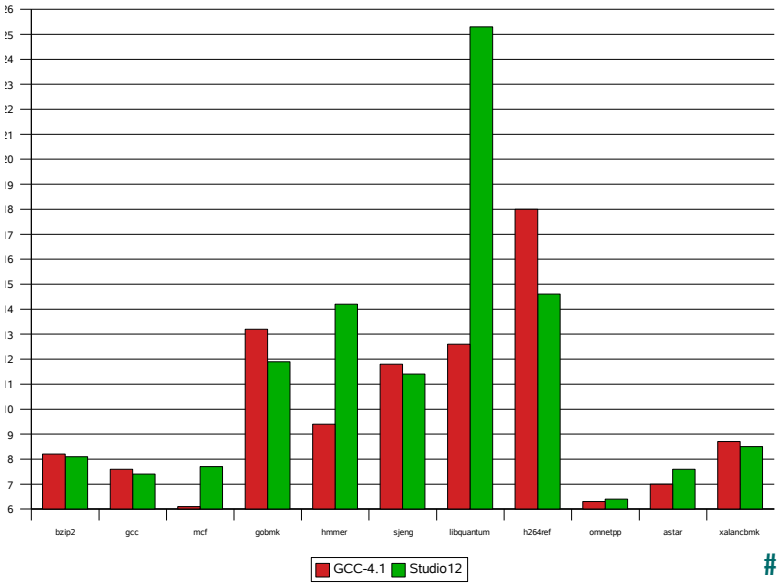
CINT2006 Sun Studio11 vs. Studio12 (Rates) Opteron 2.4 GHz



CFP2006 GCC-4.1 vs. Studio12 (Rates) Opteron 2.4 GHz

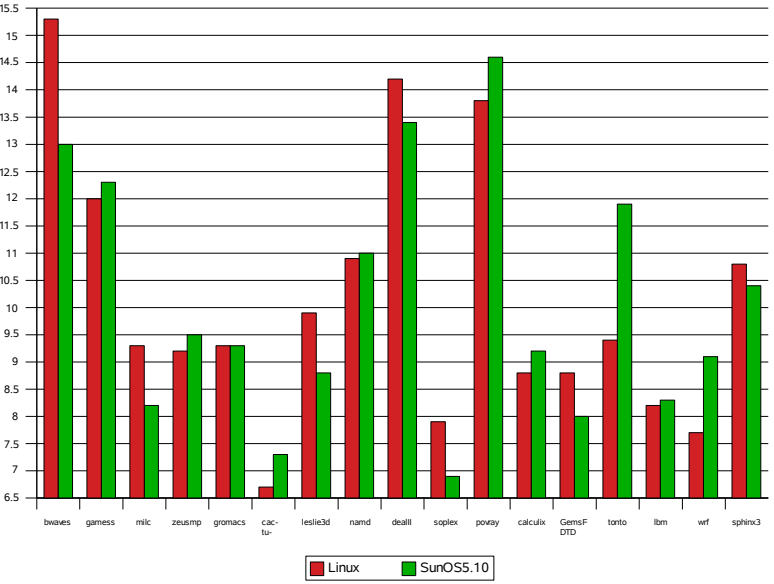


CINT2006 GCC-4.1 vs. Studio12 (Rates) Opteron 2.4 GHz

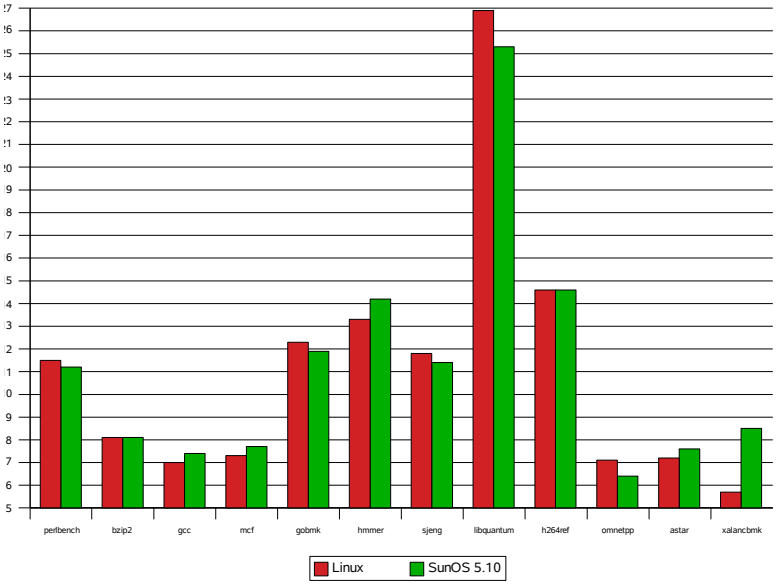


Sun Studio Compilers and Tools

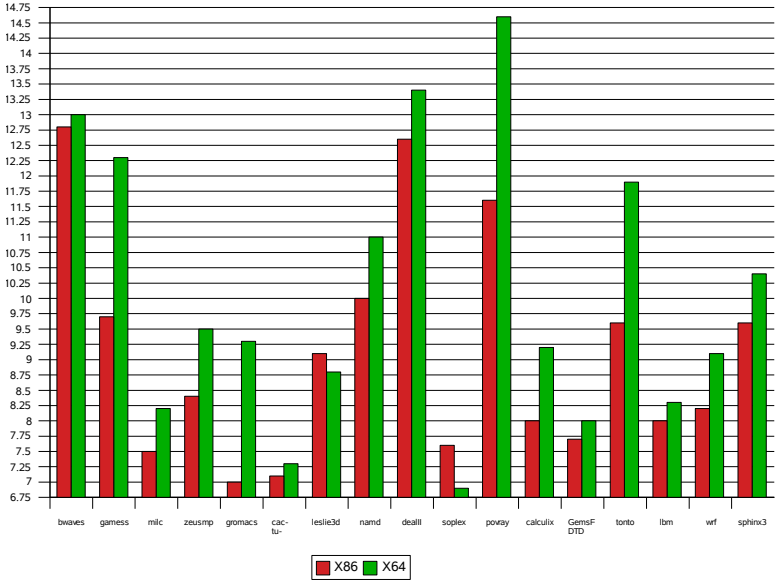
CFP2006 Studio12: Linux vs. SunOS 5.10 (Rates) Opteron 2.4 GHz



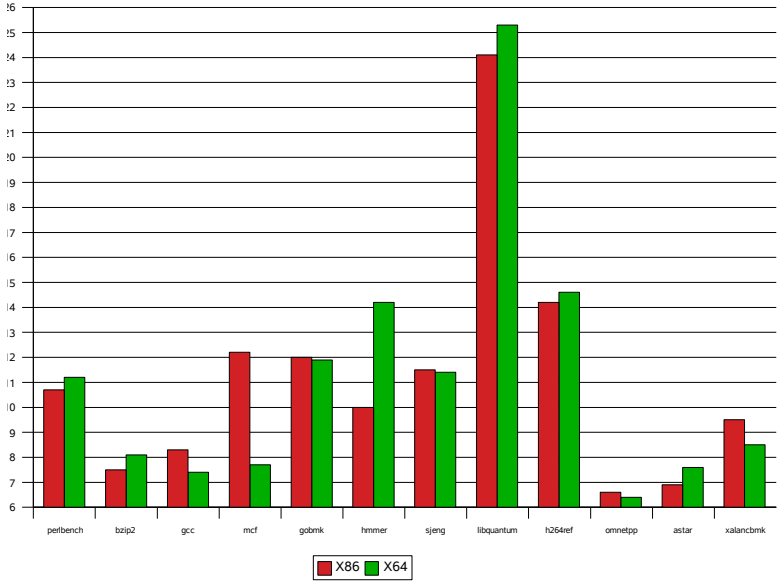
CINT2006 Studio12: Linux vs. SunOS 5.10 (Rates) Opteron 2.4 GHz



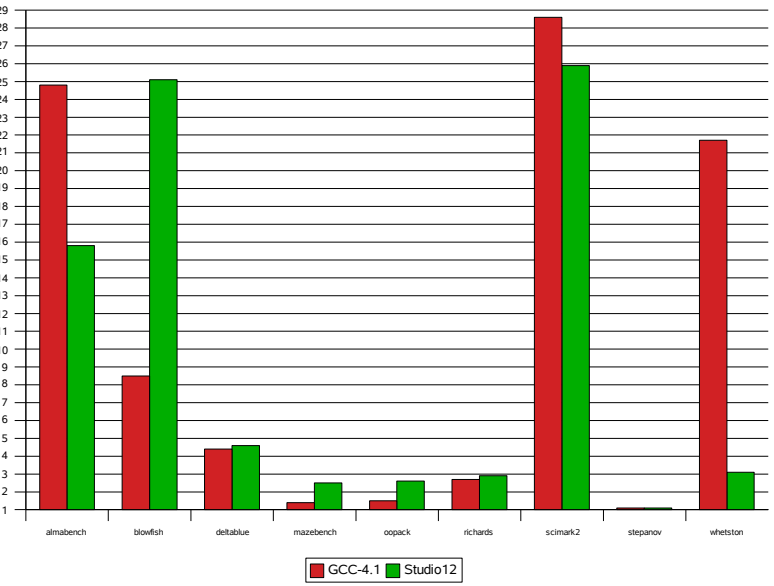
CFP2006 Studio12 SunOS 5.10: X86 vs. X64 (Rates) Opteron 2.4 GHz



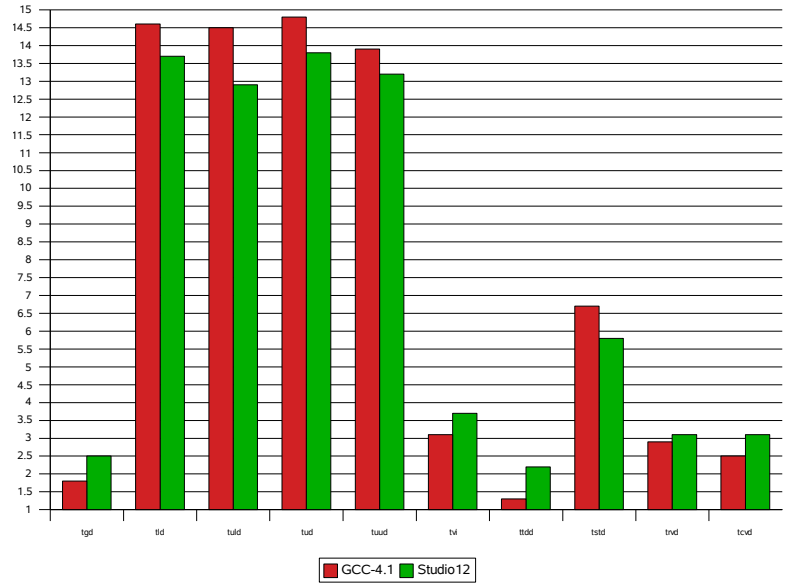
CINT2006 Studio12 SunOS 5.10: X86 vs. X64(Rates) Opteron 2.4 GHz



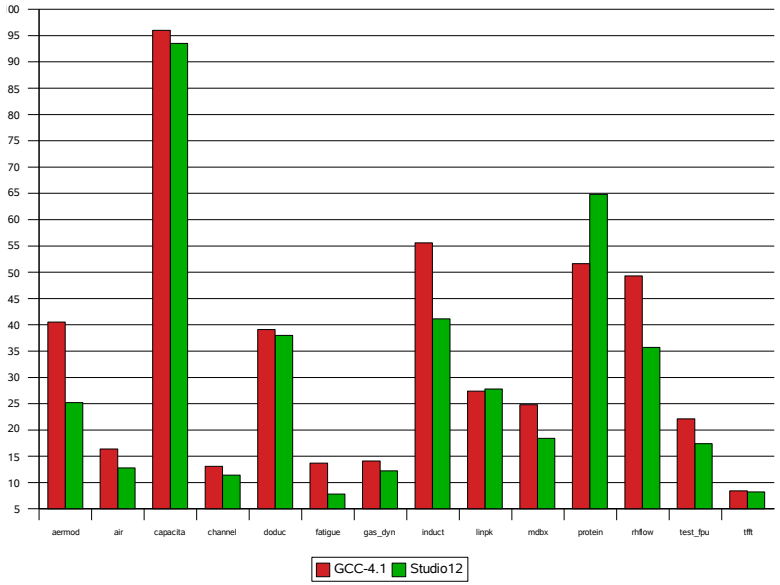
Golden-Bench: GCC-4.1 vs. Studio12 (Time sec) Opteron 2.4 GHz



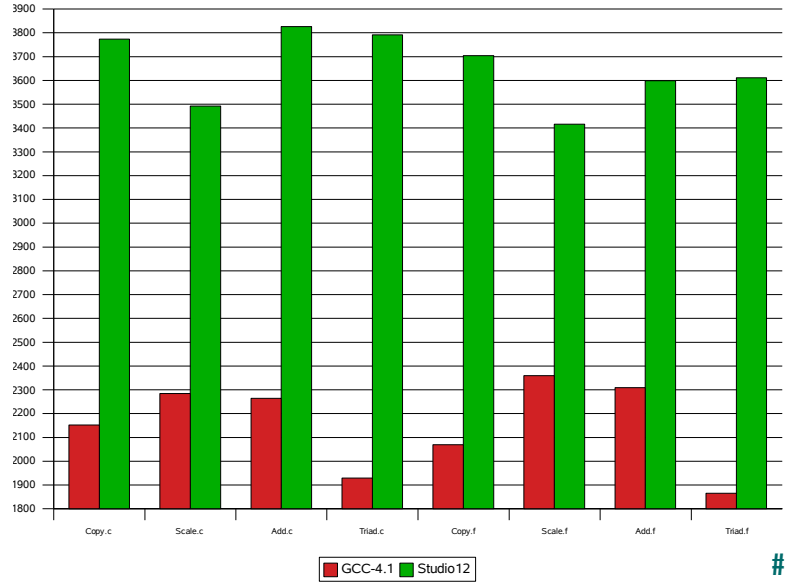
Lapack++_1.1a: GCC-4.1 vs. Studio12 (Time sec) Opteron 2.4 GHz



F95 PB05: GCC-4.1 vs. Studio12 (Time sec) Opteron 2.4 GHz

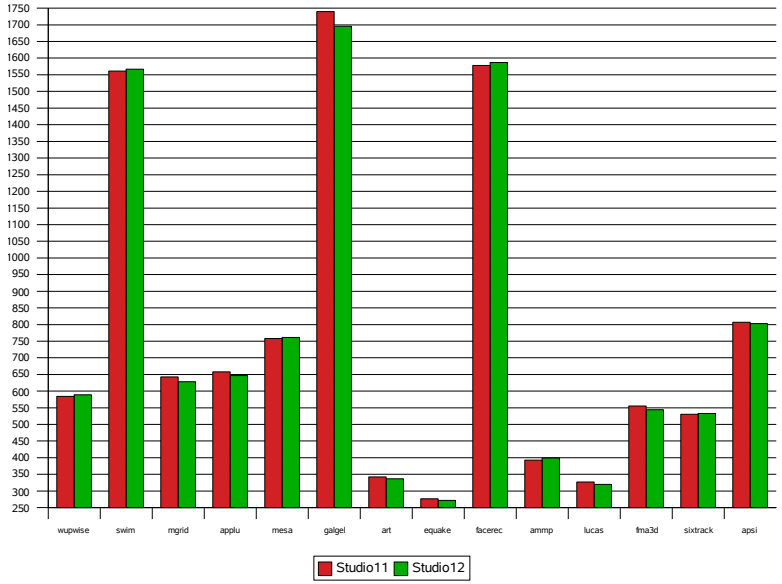


Stream.Base GCC-4.1 vs. Studio12 (Rates) Opteron 2.4 GHz

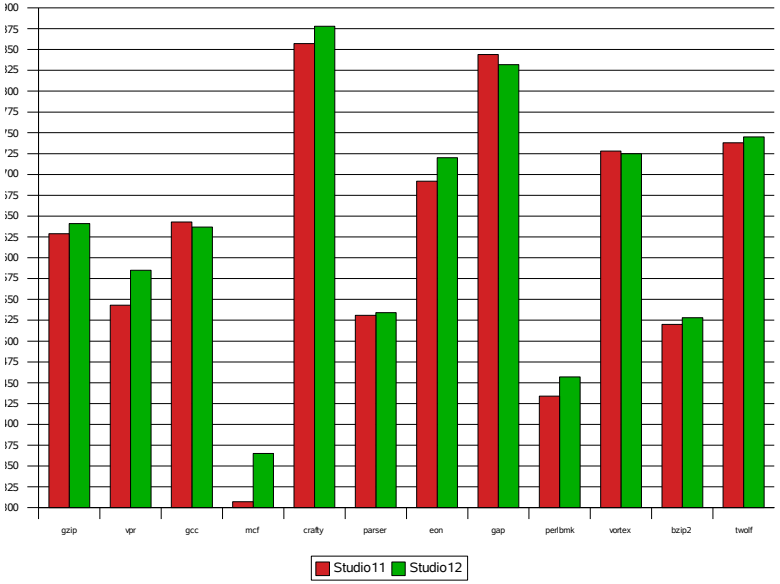


Sun Studio Compilers and Tools

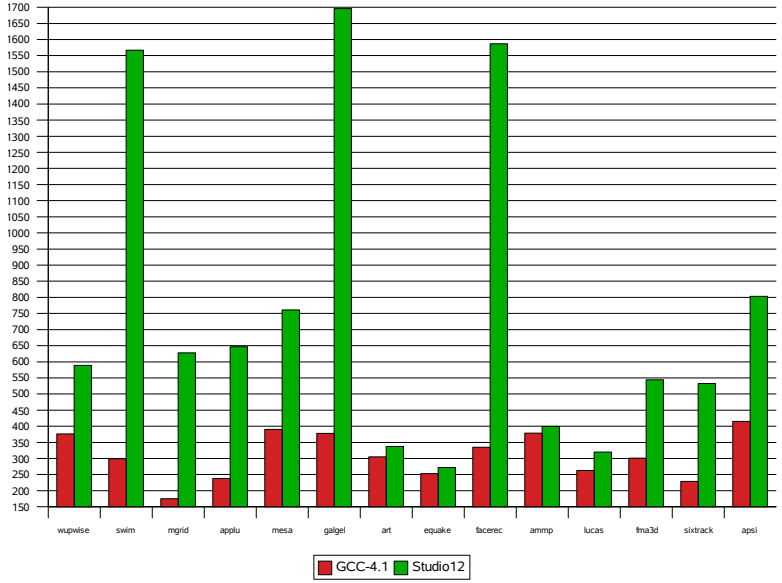
CFP2000 Sun Studio11 vs. Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



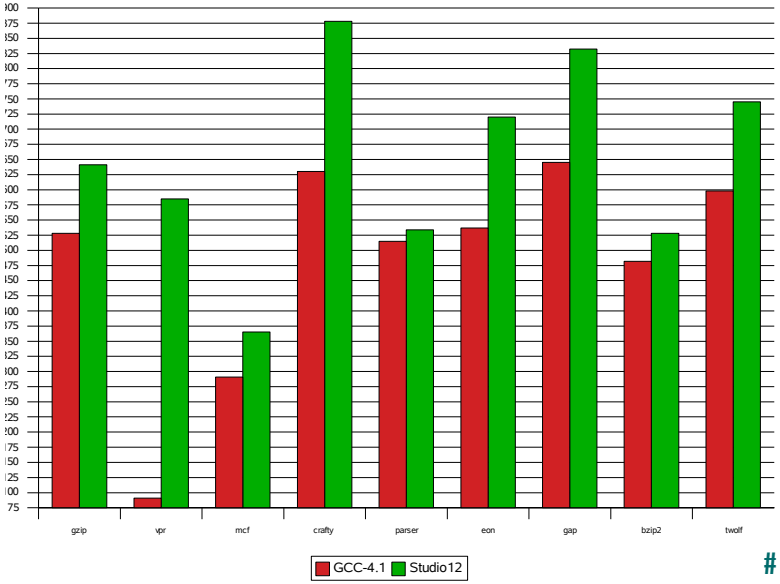
CINT2000 Sun Studio11 vs. Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



CFP2000 GCC-4.1 vs. Sun Studio12 (Rates) UltraSaprc-IIIi 1.6 GHz

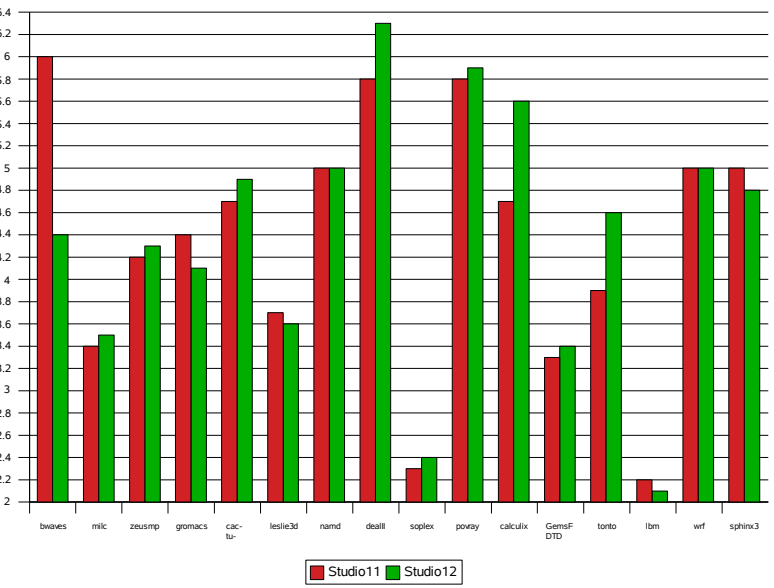


CINT2000 GCC-4.1 vs. Sun Studio12 (Rates) UltraSparc-IIIi 1.6 GHz

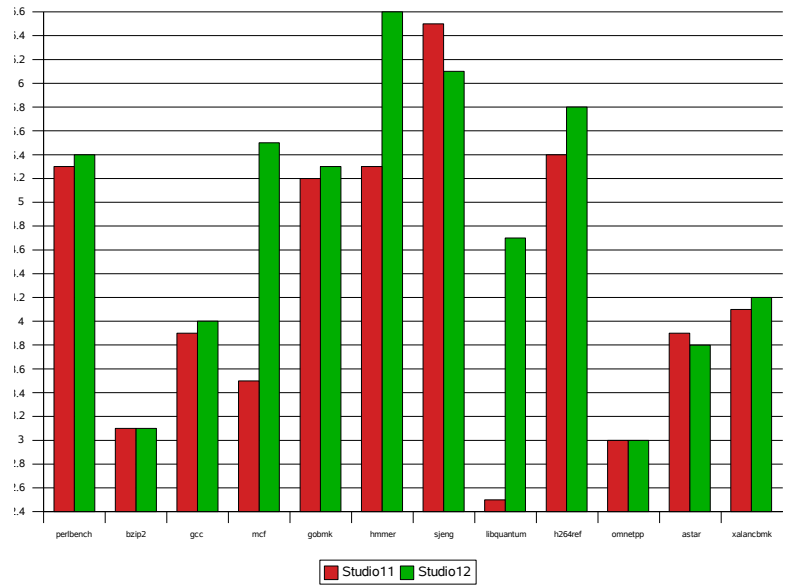


Sun Studio Compilers and Tools

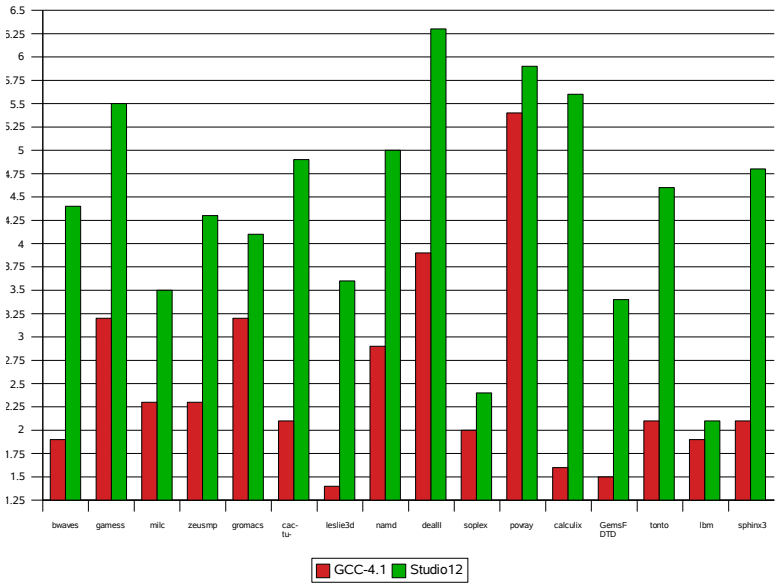
CFP2006 Sun Studio11 vs. Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



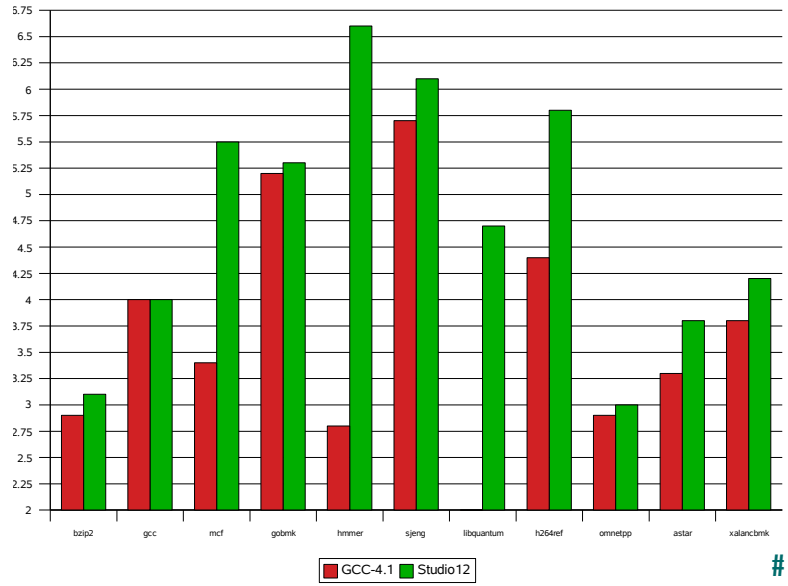
CINT2006 Sun Studio11 vs. Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



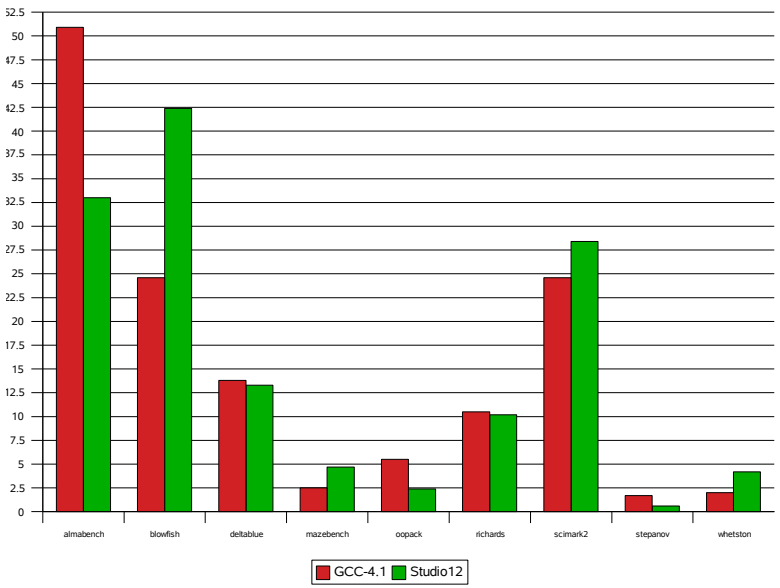
CFP2006 GCC-4.1 vs. Sun Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



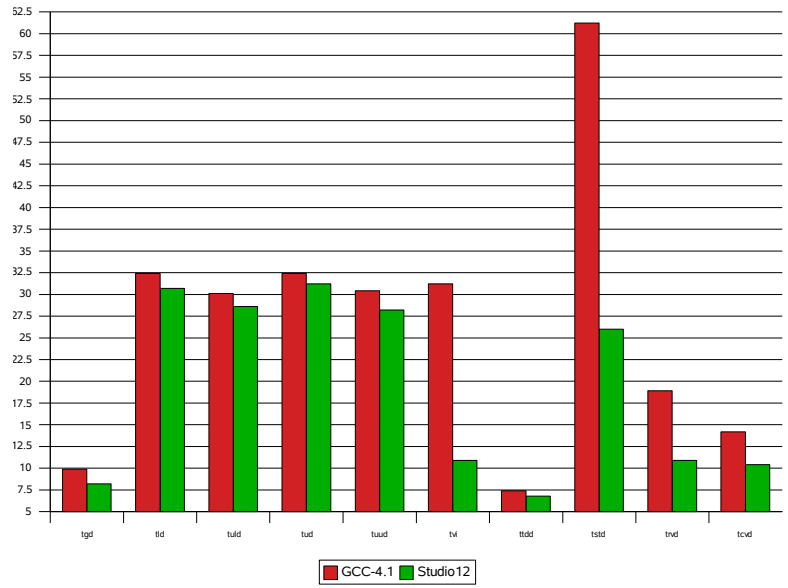
CINT2006 GCC-4.1 vs.Sun Studio12 (Rates) UltraSparc-IIIi 1.6 GHz



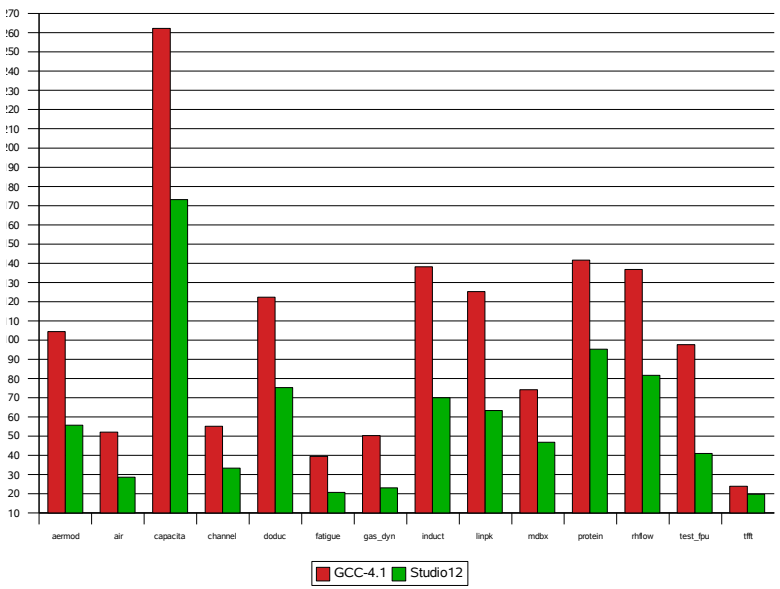
Golden-Bench: GCC-4.1 vs. Studio12 (Time sec) UltraSparc-IIIi 1.6 GHz



Lapack++_1.1a: GCC-4.1 vs. Studio12 (Time sec) UltraSparc-IIIi 1.6 GHz



F95 PB05: GCC-4.1 vs. Studio12 (Time sec) UltraSparc-IIIi 1.6 GHz



Stream.Base GCC-4.1 vs. Studio12 (Rates) UltarSparc IIIi GHz