



Home Page

Title Page



Page 1 of 9

Go Back

Full Screen

Close

Quit

HIGH PERFORMANCE COMPUTING ON OS X

JAN DE LEEUW AND ARNO OUWEHAND

1. [General](#)
2. [Our HPC setup](#)
3. [Software](#)
4. [Benchmarks](#)

1. GENERAL

High Performance Computing (HPC) uses various tools to deal with very large and/or complicated and/or time-consuming computing problems.

In this note we will briefly discuss *vectorizing*, *symmetric multiprocessing*, *optimized libraries*, and *message passing*, concentrating on the ways these techniques are implemented on the Macintosh OS X computers of the UCLA Department of Statistics.

There is also a brief **introduction** to cluster-based HPC, with many links, on the Apple developer site. And there is quite a lot of **OpenSource** code for HPC.

Date: July 13, 2002.



Home Page

Title Page



Page 2 of 9

Go Back

Full Screen

Close

Quit

2. OUR HPC SETUP

2.1. gSCAD. gSCAD means *G4 super computing all day*.

2.1.1. Hardware. The **current departmental cluster** consists of 20 400Mhz G4 rack-mounted desktop computers running Mac OS X 10.1 Server. The cluster is currently used in statistics projects by Robert Yuan, Vanessa Beddo and Roger Peng.

2.1.2. Software. gSCAD has both **LAM/MPI** and **PVM** installed for message passing, and it has both the R interfaces **RMPI** and **RPVM**. Additional software, discussed below, will be installed on the next generation gSCAD.

2.2. XServe. We are currently updating gSCAD, using the latest hardware and software from Apple. gSCAD-II will support all HPC software discussed below.

The department already has two **Xserve** computers, rack-mounted servers from Apple Computer. Both of them have dual 1 Ghz G4 CPU's, with 512MB of DDR SDRAM (will be upgraded to 2GB).

There are ten more Xserves on order, and even more are planned. Hopefully, by the beginning of the next academic year, we will have most of the computing needs of the department taken care of in one compact (although very noisy) rack. This includes both ordinary computing on a single server machine (now done mostly on the last remaining SunOS servers), and cluster computing.



Home Page

Title Page



Page 3 of 9

Go Back

Full Screen

Close

Quit

3. SOFTWARE

3.1. AltiVec a.k.a. Velocity Engine (VE). The G4 processor has a vector processing (SIMD, or single instruction multiple data) machine build into its hardware. The machine has 32 registers of 128 bits each. Apple calls this the **Velocity Engine (VE)**. Motorola calls it *AltiVec*.

Using VE means that a number of integer (16 eight-bit integers) and short floating point operations (four short floats) can be performed simultaneously, in a single clock cycle by defining and manipulating appropriate vector types. VE is ideal for signal processing, image analysis, and so on. Applications such as Photoshop are optimized for VE.

VE does not have much to do with clusters, because it can be used on any G4 workstation. There are C and FORTRAN interfaces to VE, to handle vector types and vector operations.

3.1.1. Libraries. Apple has developed **VE libraries** for BLAS (basic linear algebra), for digital signal processing, for sequence alignment, for wavelet transforms, and for bignums. There are also self-optimizing OpenSource libraries such as **ATLAS** for basic linear algebra and **FFTW** for the Fourier transform. Moreover there are quite a few commercial VE-optimized libraries for numerical computation and digital signal processing.

3.1.2. Automatic Vectorization. You can use tools from **VAST** to automatically optimize your C or FORTRAN code for VE. Currently we only have a single personal VAST license, but VAST/AltiVec may become more generally available.



Home Page

Title Page

◀▶

◀▶

Page 4 of 9

Go Back

Full Screen

Close

Quit

3.2. Symmetric Multiprocessing (SMP). Xserve, and high-end G4 workstations, come with two processors (CPU's). If an application is programmed to take advantage of these multiple processors, then this can mean a considerable speedup. Again, using SMP does not require a cluster, it just requires a Macintosh with more than one processor. And if these processors are G4's, then both SMP and VE can be used simultaneously.

OS X, and many of the applications that come with it, already use SMP. Libraries such as ATLAS and FFTW, compiled on SMP machines, also use the multiple processors. But generally for your own programs, as is the case for VE, you have to learn yet another C or FORTRAN interface.

3.2.1. OpenMP. Writing code that uses multiple processors, if they are present, is using done by *multi-threading*. Different tasks are assigned to different processors. To make this more simple, the **OpenMP** standard has been developed. OpenMP can thread applications by simply paying attention to compiler directives that you insert in your code. There are open source OpenMP compilers, such as **Omni** that have been ported to OS X. And **OdinMP** is a portable OpenMP preprocessor in Java.

3.2.2. Automatic SMP. **VAST**, mentioned in the VE section, also has tools to automatically optimize your C or FORTRAN code for SMP.



Home Page

Title Page



Page 5 of 9

Go Back

Full Screen

Close

Quit

3.3. Message Passing. In contrast to what we have discussed before, message passing is typically used on clusters of computers, which implement a MIMD system without shared memory. The program is cut up into multiple jobs, each job is executed on another computer, and when they all are done the output of the various jobs is collected and organized into a final answer. Often there is one master, who sends messages to and receives messages from a number of slaves.

Again, you have to learn an interface to message passing libraries. And again, you can use message passing in combination with SMP and VE, getting speed increases in three different ways.

3.3.1. PVM and RPVM. The oldest message passing code is written using the **Parallel Virtual Machine (PVM)**. Recently an R interface **RPVM** to the PVM system has been developed, which means it is now easy to use PVM from R. Both PVM and RPVM are on gSCAD.

Another R package, on top of either RPVM or BSD sockets, is Luke Tierney's **Simple Network of Workstations (SNOW)**. It handles "embarrassingly parallel" computations, where jobs can be very easily split into sub-jobs (because they do the same thing a possibly large number of times: think of MCMC).

3.3.2. MPI and RMPI. The **Message Passing Interface (MPI)** is both more recent and more elaborate than PVM. There are two major open source implementations, called **LAM/MPI** and **MPICH**, both ported to OS X. And there is an **R interface RMPI** as well. SNOW will be adapted to work with RMPI.



Home Page

Title Page



Page 6 of 9

Go Back

Full Screen

Close

Quit

3.4. Additional Software.

3.4.1. *High Performance FORTRAN (HPF)*. **HPF** is a set of extensions of FORTRAN 90 that makes it comparatively easy to write parallel program for MIMD machines. The **ADAPTOR** compiler, which translates HPF to FORTRAN 77 with MPI parallelization, has been **ported to OS X**. HPF code is easier to maintain than code using MPI directly.

3.4.2. *Cilk*. **Cilk** is a very convenient system to generate multi-threaded code for SMP systems. It merely extends the C language with the three keywords *cilk*, *sync* and *spawn*, and generates threaded code, using the **POSIX threads** interface. The standard Cilk distribution has OS X support.

3.4.3. *Linda*. Parallel programming in **Linda** uses the virtual shared memory model. A portion of memory is reserved as a shared tuple space, and some extensions of either C or FORTRAN are used by parallel processes to move data in and out of tuple space. The language extensions are simple and intuitive, much easier to learn than PVM or MPI.

There are literally hundreds of Linda (or distributed tuple space) implementations, on top of many of the functional programming languages. Both **IBM** and **Sun** have Java implementations. A **(commercial) version of C-Linda** is available for OS X (there is also **Paradise**, a WAN version that can use, for instance, all computers in the department). We have a trial license for gSCAD.



Home Page

Title Page



Page 7 of 9

Go Back

Full Screen

Close

Quit

4. BENCHMARKS

4.1. VP and SMP. The Benchmark is taken from [Gaurav Khanna's website](#). It's PDE code to model perturbed black holes. Gaurav uses the Absoft FORTRAN compiler, and he uses the VAST tools to automatically optimize for VE and/or SMP. This gives four versions of the code on dual processor machines, two versions on single processor G4 machines. We use Gaurav's binaries. Each cell is the median of ten runs.

mac	cpu	nzc	vec_nzc	par_nzc	par_vec_nzc
cube	450 Mhz	55.930	22.845	na	na
tibook	800 Mhz	43.205	15.260	na	na
dual	2×1 Ghz	25.100	5.700	7.750	3.060
xserve	2×1 Ghz	25.425	5.875	7.920	3.160

In this particular example automatic VE optimization makes the program **four** times faster, automatic paralellization makes it **three** times faster, and combining both (i.e. VE optimizing all threads) makes it **eight** times faster.

The tiny discrepancy between the Xserve and the dual G4 may be due to the fact that the Xserve runs Mac OS 10.1 Server, which implies there may be other processes running as on Mac OS X Client.

For details, code, and comparisons with an 1 Ghz Pentium III, see Gaurav's site.

4.2. Cilk. This is a simple Cilk benchmark, which computes the n^{th} Fibonacci number, on a dual 1 Ghz G4. The cilk code (to be embedded in a main) is

```

cilk int fib(int n)
{
    if (n < 2)
        return (n);
    else {
        int x, y;
        x = spawn fib(n - 1);
        y = spawn fib(n - 2);
        sync;
        return (x + y);
    }
}
    
```

Here "spawn" tells the scheduler that the child can be run simultaneously with other spawned children, while "sync" tells the scheduler to wait until all spawned children are finished. The results (in seconds) are

	$n = 10$	$n = 20$	$n = 30$	$n = 40$
1 CPU	0.0029	0.0081	0.6621	77.1669
2 CPU	0.0036	0.0081	0.4094	42.0480

For large n we see that using both CPU's is almost twice as fast, for small n there is no gain because of the overhead.



Home Page

Title Page



Page 9 of 9

Go Back

Full Screen

Close

Quit

4.3. MPI. We use the **LINPACK benchmark**, which solves random dense linear systems.

mac	CPU	RAM	# CPU	problem size	Gflops
4 G4's	400 Mhz	1024 Mb	4	10000	1.54
8 G4's	400 Mhz	1024 Mb	8	10000	2.57
2 G4's	2×1 Ghz	512 Mb	4	10000	2.84
2 Xserve's	2×1 Ghz	512 Mb	4	10000	3.02

There is still much room for improvement. gSCAD-I has 1024 Mb of memory available per CPU and has a dedicated switch. Both dual processor configs have only 256 Mb per CPU available and are not on a dedicated switch. Also, gSCAD-I is limited to 100 Mbit connections, while both the Xserve and the G4 Dual 1Ghz are capable of 1000 Mbit connections (provided they are connected to a gigabit switch).

UCLA DEPARTMENT OF STATISTICS, 8142 MATH SCIENCES BLDG, BOX 951554, LOS ANGELES, CA 90095-1554

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

E-mail address, Arno Ouwehand: arno@stat.ucla.edu