

Super-computing and Parallel Computing – Introduction and Resources

Zhilin Li

Center for Research in Scientific Computation & Department of Mathematics
North Carolina State University
Raleigh, NC 27695, e-mail: zhilin@math.ncsu.edu

1 An overview

High-performance computing (large scale in terms of size, number of floating operations and loops, memory/cache, and storage) depends on many factors including:

- algorithm design. For example, we would like to use implicit scheme for diffusion equations so that we can take large time steps $\Delta t \sim h$, and we would often prefer to use explicit scheme for wave equations where we have $\Delta t \sim h$.
- compilers. For example, in terms of the speed, assemble language is fast than Fortran, Fortran is faster than C/C++, C/C++ is faster than Matlab.
- software packages in which the algorithms are likely optimized, vectorized, or parallelized and make use of the data structures of the compiler and probably hardware of computers.
- computers. Super-computers such as Cray T916, IBM-SP2, SGI Origin 2000 etc., usually have multi-processors (CPU)s; Personal PC such as Pentium machines may have one or two CPUs. Ultra-10 workstations are faster than Ultra-1 workstations. The Clustered PCs have become popular parallel tools.
- National super-computing centers at: San Diego, Pittsburgh, Illinois, Argon National labs, Army, Navy super-computing centers.
- Super-computer at North Carolina Super-computing center. They have Cray T916 (may disappear); IBM-SP2 (may upgrade to IBM-SP3). (SGI) origins (good for graphics), and numerous software packages. They have free lessons for students and faculty in academics (<http://www.ncsc.org/training>).
- To use super-computers, the job usually have to be queued depending on the size, time needed for the problem. It is not recommended for small to medium sized problems.

2 Computer language and speed:

The order of a speed of operations within a computer is roughly the following

CPU, Register/Bus, Cache (I, II, ...), main memory (RAM), Disk, Tape, etc.

If we know the way in which the data are stored, we can take advantage of it to get fast performance. An example is the matrix-vector multiplication using Fortran computer language, see Section 2.1.4. In fortran, a matrix is stored column-wise. So we should try to use the matrix column-wise instead of row-wise using Fortran.

Comparisons of i - j and j - i algorithms

for matrix-vector multiplication *matvec-ij.f90*, *matvec-ji.f90* using Sun Workstations, Cray T916 with and without vector pipelines.

3 Performance multi-task job simultaneously

3.1 Vector pipeline computation

See Section 2.1.2. A vector pipeline is a register level device, which is usually in either the control unit or the arithmetic logic unit. It has a collection of distinct hardware modules or segments that (i) execute distinct steps of an operation and (ii) each segment is busy once the pipeline is full.

Most compiler will automatically take advantage of the vector pipeline structure if it is possible. Usually it would be efficient to use modules, objective oriented programming. For example, in Matlab, it is performed to use $c = A * x$ than

```
for i=1:m
  c(i) = 0;
  for j=1:n
    c(i) = c(i) + A(i,j)*x(j);
  end
end
```

Examples:

- Comparison between Sun Ultra machine and Cray T916.
- Comparison between vector-on and vector-off on Cray T916. Note: we need to use *-O1* when we compile the code in order to see the difference. This option force low level of optimization of the code.

3.2 Parallel computing I: shared memory and Open-MP

Open-MP is an interface model between computer languages and parallel computers with *shared memory*. See the diagram of shared parallel computers in page 2 of Section 2.1. Open-MP is compiler directive and needs to have an existing Open-MP library and environment variables extended the base language. A good reference site is <http://www.openmp.org>.

An example in Fortran 90

```

program hello
  print *, "Hello parallel world from threads"
!$omp parallel
  print *, omp_get_thread_num()
!$omp end parallel
  print *, "Back to the sequential world"
end

```

Let the file be `openmp1.f90`. On an NCSC SGI Origin 2400 machine, we do the following:

```

setenv OMP_NUM_THREADS 8
f90 -mp openmp1.f90
a.out

```

- We can add the parallel function when it is needed. For example, in **Fortran**, the format is:

```

!$omp parallel
    code to be executed parallelly
!$omp end parallel

```

- In C, the format is:

```

# pragma omp parallel {
    code to be executed by each thread
} .

```

- It is relatively easier to debug compared with distributed machine such as MPI.
- The scalability (the speed-up from one parallel machine to another) depends on skill of the parallel programming and may not as good as MPI.
- One has to be very *careful* about the output. Compare with serial code to make sure it works properly!

See other examples: `serialpi_mp.f`, `serialsum_mp.f`, `wrong.c`

References:

1. <http://www.ncsc.org/training/>
2. <http://www.openmp.org/>
3. R. Chandra et. el. *Parallel Programming in OpenMP*, Morgan Kaufmann Publishers, 2001, ISBN: 1-55860-671-8.

3.3 Parallel computing II: MPI for computers with distributed memory

Message Passing Interface (MPI) is the most widely used of the new standards. It is not a new programming language, rather it is a library of sub-programs than can be called from C/C++ and Fortran computer languages. It is a standard that designed for parallel computing with distributed memory. The foundation of MPI library is a small group of functions that can be used to achieve parallelism by **message passing**. A message passing function is simply a function that explicitly transmits data from one process to another.

Seven most important functions in MPI

- MPI_Init
- MPI_Common_size
- MPI_Common_rank
- MPI_Send
- MPI_Recv
- MPI_Barrier
- MPI_Finalize

The syntax is generally the same for C and Fortran. The exception is that C return the status of the function call while Fortran has an additional integer argument at the end to return the error code. See sample code on page 2 and 4 of Section 2.5, page 2-3 of Section 2.6, and page 5-6 of Section 2.6.

3.4 Speed-up and Amdahl's Timing Model

$$\text{Sped-up} = \frac{T_{\text{sequential}}}{(1 - \alpha) T_{\text{sequential}} + \alpha T_{\text{sequential}}/p} \quad (1)$$

where p is the number of processes, $T_{\text{sequesntial}}$ is the serial execution time.

Some Technical terms:

- SIMD: Single instruction with multiple data
- MIMD: Multiple instructions with multiple data
- SPMP: Single program with multiple data

References:

1. <http://www.ncsc.org/training/>
2. <http://www.mpi-forum.org>
3. <http://www.erc.msstate.edu/mpi/mpi-faq.html>
4. Peter S. Pacheco *Parallel Programming with MPI*, Morgan Kaufmann Publishers, 1997, ISBN: 1-55860-339-5.

Note:

- Another parallel model is called PVM (parallel virtual machines).
- There are several mathematical software for parallel computing. Among them, ScaLAPACK for linear algebra package, and PETsc (Portable Extensible Toolkit for Scientific Computations).

4 Use of IBM SP2 at NCSC

The IBM SP system at NCSC is composed of 4-way SMP nodes. Each node contains four 375 MHz POWER3 processors, two gigabytes of memory, a high-speed switch network interface, a low-speed ethernet network interface, and local disk storage. Each node runs a standalone version of AIX, IBM's UNIX based operating system.

The POWER3 processor can perform two floating-point multiply-add operations each clock cycle. For the 375 MHz processors this gives a peak floating-point performance of 1500 MFLOPS.

4.1 Accessing the SP

There are approximately 175 nodes in the NCSC system. Interactive logons are permitted on three nodes (sps001, sps002, and sps003). The remaining nodes are reserved for jobs submitted through *Load Leveler*. Currently, these restrictions are not automatically enforced. Accounts of users violating these restrictions may be disabled or removed from the system.

To access the SP system ssh (or telnet) to one of the designated interactive nodes:

- sps001.ncsc.org
- sps002.ncsc.org
- sps003.ncsc.org

4.2 Changing Passwords

Each node maintains its own password file. To provide a simple mechanism for users to change their passwords the local password files are updated hourly from the control workstation.

To change your password, log onto the control workstation (spcws.ncsc.org) and follow the prompts. The only action that users are allowed to perform on the control workstation is to change their password.

After changing the password on the control workstation it will be propagated to the other nodes within one hour.

4.3 Running Jobs

Jobs are scheduled for execution on the SP by submitting them to the Load Leveler system. Job limits are determined by user resource specifications and by the job class specification. Job limits affect the wall clock time the job can execute and the number of nodes available to the job. Additionally, the user can specify the number of tasks for the job to execute per node as well as other limits such as filesize and memory limits. See page 3 of Section 2.4 for example.

The turn around time for IBM SP2 is usually short than Cray T916. The Cray T916 may cease to exist in the near future.

References:

1. <http://www.ncsc.org/USERSUPPORT/USERGUIDE/>