Marc Leef
Allen Malony
CIS 410
4/21/14

Homework 1

Parallel Computing:

1) A parallel computer can be defined as a number of processors capable of working in conjunction to solve a problem. A computer with one processing core can be considered a parallel computer, but only if the processor possesses multiple functional units and supports instruction level parallelism routines such as superscalar execution or simultaneous multi threading.
2) Parallel hardware, concurrency, and performance.
3) Yes, nmax is just the problem size needed to achieve rmax, which is the peak performance of the computer. It is possible that parallel computer A could have both a higher rmax and a lower nmax than parallel computer B.
4) Parallel processing is mainly utilized to speed up the run times of programs, and also to allow programs to tackle larger and larger computational problems in a reasonable amount of time. Technology is trending towards more and more cores as the rate of transistor size reduction slows. Therefore, understanding how to best utilize these increasing number of cores will greatly aid algorithm design in the near future.

Parallel Architectures:

1) Shared memory architecture consists of multiple workers that operate on the same data. Distributed memory architectures force the workers to share messages between one another, as they cannot access anything but their own data. Sharing memory is the most efficient way to share data between programs, and there is little to no communication overhead. Distributed memory creates less concurrency issues than shared memory, and the hardware can also be simplified.
2) NUMA (Non Uniform Memory Access) is a sort of hybrid between shared and distributed memory. It possesses both local and remote memory to which access speeds vary, hence the non-uniform part of the name. The performance of this architecture is heavily dependent on data locality. It was created to combine positive elements of both shared and distributed architectures.
3) Multi-core processors have risen in large part due to diminishing gains in the performance of single-core processors. Therefore, the new now most popular way to increase performance is to increase the number of cores. Multi-core processors provide better performance and decreased power consumption. Their rise is disruptive to the HPC community as it will force developers to make much more intelligent use of the hardware—or suffer minimal performance improvements.
4) The interconnection network is tantamount to the success of large-scale parallel systems. The timely and efficient exchanging of messages between processes is the bottleneck of distributed memory architectures. The topology, routing algorithms, switching strategy, and flow control mechanism form the foundation of a successful parallel system, as the message passing must work quickly and efficiently to ensure the system is functioning correctly.

Parallel Performance Theory:

1) Amdahl's Law states that there is a fixed limit on the speed-up that parallelization can provide on a fixed size data set. Gustafson's Law, however, states that computations on massive data sets can always be efficiently parallelized; Amdahl's law applies when the problem size is fixed, Gustafson's applies when the problem size can increase with the number of processors.
2) Because when the problem size is fixed, the speedup is determined by the degree of sequential execution time and not by the number of processors. However, when the problem size increases with the number of processors, this is not true as the speedup function includes the number of processors.
3) a) Not necessarily. Though it is better at tackling problems of fixed size, it may be worse at solving problems with an increased number of processors.
   b) If the machine has a different number of processors and/or the size of the problem changed then yes, algorithm B could have better strong scaling.
4) This only creates a problem if the ratio of serial compute time to parallelizable compute time is different. As Amdahl's law states that the speedup limit is based on the time needed for the serial portion of the program, the different serial compute times would be problematic.
5) a) Save one processor to perform the addition, use the other processors to perform autonomous multiplications on different parts of the vectors. The addition processor awakens when it receives the second product, and performs a reduction. This uses both a map pattern and a reduction.
   b) $T1 = 6n$,
   $Tp = (6n/p) + (52\log p)$,
   $S(p) = (6n)/((6n/p) + (52\log p))$
   $E(p) = (6n)/((6n/p) + (52\log p))/p$

   c) $S(1) = 1 \quad E(1) = 1$
   $S(4) = 3.69 \quad E(4) = .92$
   $S(16) = 9.68 \quad E(16) = .60$
   $S(64) = 13.02 \quad E(64) = .20$
   $S(256) = 11.70 \quad E(256) = .04$

   d) $.92 = (6n)/((6n/64) + (52\log 64))/64$
   $n = 26528$