5. Basic concepts in parallel scientific computing

[1]

[1] M. Ganesh, MATH440/540, SPRING 2018

[2] Performance: work done per time unit

- **The performance of a code on a computer is the number of floating-point operations in the code that the computer can execute in a given time unit:**
  - ⋆ **MFLOPS (or megaFLOPS): Millions ($\mathcal{O}(10^6)$) of FLoating-point Operations Per Second (FLOPS)**
  - ⋆ **GFLOPS (or gigaFLOPS): Billions ($\mathcal{O}(10^9)$) of FLOPS**
  - ⋆ **TFLOPS (or teraFLOPS): Trillions ($\mathcal{O}(10^{12})$) of FLOPS**
  - ⋆ **PFLOPS (or petaFLOPS): Quadrillions ($\mathcal{O}(10^{15})$) of FLOPS**
  - ⋆ **EFLOPS (or exaFLOPS): Quintillions ($\mathcal{O}(10^{18})$) of FLOPS**
  - ⋆ **ZFLOPS (or zetaFLOPS): Sextillions ($\mathcal{O}(10^{21})$) of FLOPS**
  - ⋆ **YFLOPS (or yottaFLOPS): Septillions ($\mathcal{O}(10^{24})$) of FLOPS**

[2] M. Ganesh, MATH440/540, SPRING 2018

Remarks[3]:

- **FLOPS is not the plural of a FLOP (FLoating-point OPerations per unit time)**
- **Plural of a FLOP will be denoted throughout the course by FLOP's**
- **That is, a 50 TFLOPS machine is different from a machine with performance 50 TFLOP's.**
- **Throughout the course, a processing unit in a chip will be referred to as a core.**

[3] M. Ganesh, MATH440/540, SPRING 2018

Examples[4]:

- **Consider a SMP machine that has 1024 cores with each core capable of performing a FLOP, say addition of two integers, in $1\mu s$ (1 microsecond). What is the performance of a computer code in the machine that involves only addition of two vectors, with each vector of length 2^{10}, assuming that the code is parallelized and allocates operations to the cores in a balanced way?**
- **Solution:**

[4] M. Ganesh, MATH440/540, SPRING 2018

Example[5]**:**

- **Assume you have access to a cluster machine that has a total of** 2048 **cores with each core capable of performing addition of two integers in** $1\mu s$**.**
- **The job submission policy (software, say TORQUE/MOAB) in the cluster has a (hard-coded) restriction that no job should occupy more than half of the resources in the machine.**
- **What is the performance of a computer code that involves only addition of two vectors, with each vector of length** 1200**, assuming that the code**
  - ⋆ **was parallelized and allocates operations in a balanced way and**
  - ⋆ **was submitted using a job script asking for maximum resources?**

[5] M. Ganesh, MATH440/540, SPRING 2018

- **Solution**[6]**:**

[6]M. Ganesh, MATH440/540, SPRING 2018

- [7]**Suppose that we want to compute a variable S, which sum of N variables s_i, that is,**

$$S = \sum_{i=1}^{N} s_i. \tag{2.1}$$

- **Suppose that we program this in a code and implement on a computing environment (i.e., a combination of a system, compiler, job scripts etc.) and it takes T_N seconds to compute S.**
- **Then performance is FLOPS, because computation of S requires floating point additions**
- **Usually, we refer to this as MFLOPS and if N is very large, GFLOPS**
- **Note that for small N, say, less than a critical number N_c depending on the computer clocks, T_N may be zero**
- **Arbitrary precision in measuring performance cannot be achieved. (Computer uncertainty principle.)**

[7]M. Ganesh, MATH440/540, SPRING 2018

- [8]**In the performance analysis, say, for computing the variable S in (2.1), we have not taken into account of memory references**
- **Computation of the variable S requires memory references**
- **For very large problems (that require substantial access to memory for computations) the relation, between the amount of work done and the amount of memory that must be accessed, is critical**
- **Such evaluations in algorithms are crucial in current hybrid SMP-DMP architectures due to limitation of the fast memory access per node in clusters**
- **Work/Memory ratio: of an algorithm is the ratio ρ_{WM} of the number of FLOP's to the number of memory locations referenced**
- **The number of memory locations referenced in the pseudo-code:**
  `for integer i from 1 to 1,000,000, if (i .is. even) then j = i else j = i+1`
  is . But references to these are made times

[8]M. Ganesh, MATH440/540, SPRING 2018

- [9]**What is the work memory ratio in computing the sum S in (2.1)?**

$$\rho_{WM} = \qquad\qquad \approx \qquad , \quad \textbf{for large N}$$

- **Most cluster computers perform significantly better for HPC algorithms with large ρ_{WM}**

- **One of the criteria for designing efficient HPC algorithms is to have large ρ_{WM}**

- **If the maximum speed that the memory system in a machine (such as that with one limited bandwidth channel) can deliver information is η words per unit time, then the maximum performance that can be achieved is $\eta\rho_{WM}$ MFLOPS. (In HPC machines: a word = bits.)**

- **The above definition assumes a single path to memory, and is very simplistic. (See below to avoid this assumption.)**

[9]M. Ganesh, MATH440/540, SPRING 2018

- **Example:**[10] **An important component in most scientific computing algorithms is the matrix-vector multiplication:**
- **Let A be an $n \times n$ matrix with entries $a_{ij}, i, j = 1, \cdots, n$ and let $\mathbf{v}$ be an n-dimensional vector with entries $v_i, i = 1, \cdots, n$.**
- **Then $A\mathbf{v}$ is an n-dimensional vector with entries**

$$(A\mathbf{v})_i = \qquad\qquad i = 1, \cdots, n. \tag{2.2}$$

- **The total number of FLOP's to compute the vector $A\mathbf{v}$ is:**
- **The total number of memory locations referenced to compute $A\mathbf{v}$ is:**
- **Hence the work/memory ratio of the matrix-vector multiplication algorithm is:**

$$\rho_{WM} = \qquad \approx \qquad , \quad \textbf{for large n}$$

- **If the code is such that vector $\mathbf{v}$ needs to be read for each $i = 1, \cdots, n$ in (2.2), then $\rho_{WM} \approx$**

[10] M. Ganesh, MATH440/540, SPRING 2018

- [11]**In scientific computing, the worst case work/memory ratio can be avoided, thanks to hierarchy of components in memory subsystems**
- **The memory components closest to the core are the fastest**
- **For example, resident data to the core can be supplied in a few cycles using memory cache**
- **The term *cache-hit* refers to a core accessing a datum that is already in the cache**
- **The size of the main memory is much larger than the size of cache and hence many locations in the main memory share only a few locations in the cache**
- **The maximum performance in the matrix-vector multiplication example can be achieved only if we assume that the vector v was read once from the main memory, and stays in the cache**

[11]M. Ganesh, MATH440/540, SPRING 2018

[12]Performance of a two-level memory system:

- **Let the rate of transmission from memory to core be measured as** `words/second`
- **The performance of a two-level memory system (cache and main memory) may be modeled using the percentage of cache-hits as:**

$$\frac{\textbf{average cycles}}{\texttt{word access}} = \%\textbf{cache-hits} \times \frac{\textbf{cache cycles}}{\texttt{word access}} + (1 - \%\textbf{cache-hits}) \times \frac{\textbf{main memory cycles}}{\texttt{word access}}$$

- **Example: Let cache access time be 5 cycles per word. Let main memory access time be 50 cycles per word. Let the cache-hits rate be 20%.**
  What is the average number of cycles per word access?
  (What if the cache-hits rate is instead 90%? Conclusion?)
- **Solution:**

[12]M. Ganesh, MATH440/540, SPRING 2018

- [13]**The average number of words per second may then be modeled as**

$$\frac{\textbf{words}}{\texttt{second}} = \left(\frac{\textbf{average cycles}}{\texttt{word access}}\right)^{-1} \times \frac{\textbf{cycles}}{\texttt{second}}$$

- **Example:**
  Consider a two-level memory system with a 2.5 GHz processor.
  In the system, let cache access time be 5 cycles per word,
  main memory access time be 50 cycles per word, and
  let the cache-hits rate be 20%.
  What is the average memory rate per word in the system?
  (What if the cache-hits rate is 90% instead? Conclusion?)

- **Solution:**

[13]M. Ganesh, MATH440/540, SPRING 2018

- [14]**If a code implemented on a system with p GHz processor exhibits an average memory access time of n cycles per minute, then the performance of the code is bounded by**

$$\left[n\frac{\textbf{cycles}}{\texttt{word access}}\right]^{-1} \times \left[p \times 10^9 \frac{\textbf{cycles}}{\texttt{second}}\right] \times \rho_{WM} = \frac{p \times \rho_{WM} \times 10^9}{n} \textbf{FLOPS}$$

- **Example:**
  Consider a two-level memory system with a 2.5 GHz processor.
  In the system, let cache access time be 5 cycles per word,
  main memory access time be 50 cycles per word, and
  let the cache-hits rate be 20%.
  Assume that a code that ran on the system had $\rho_{WM} = 2$. What was maximum possible performance of the code on the system.
  (What if $\rho_{WM} = 1$ instead? Conclusion?)

- **Solution:**

[14]M. Ganesh, MATH440/540, SPRING 2018

[15]The $\mathcal{O}$ notation:

- **While calculating the Work/Memory ratio ρ_{WM}, for sum (2.1) and matrix-vector multiplication (2.2) calculations, we respectively used**

$$\rho_{WM} = \qquad \approx \quad , \qquad \rho_{WM} = \qquad \approx \qquad \textbf{for large N, n} \tag{2.3}$$

- **We say $f(x) = \mathcal{O}(g(x))$ if there exist constants $C, x_C \geq 0$ such that**

$$f(x) \leq Cg(x) \quad \textbf{for all} \quad x \geq x_C.$$

  That is behavior of these two functions are asymptotically similar

- **Examples:**
  Let a and b be some fixed constants and let $f(n) = an^2 + bn$. (For second example in (2.3), $a =$ or and $b =$ or .) Let $g(n) = n^2$. Show that $f(n) = \mathcal{O}(g(n))$

- **For the first example in (2.3), show that $\rho_{WM} = 1 + \mathcal{O}(\frac{1}{N})$.**

[15]M. Ganesh, MATH440/540, SPRING 2018

- [16]Based on the above discussions, clearly, bandwidth between the cores and various memory units plays an important role in performance of codes on parallel computing environments.
- Current HPC parallel computing systems allow the aggregate bandwidth between the cores and memory to be made much larger at minimal expense. Increasing such bandwidths will play a crucial role in future development of HPC systems.

Basic techniques in parallelism

- An essential ingredient in parallel scientific computing is to identify and create operations that may be performed concurrently (that is in parallel) with as minimal communication as possible
- Such operations are called tasks. That is, a task in an algorithm is part of a computation that can be thought of independently from other parts in the algorithm

[16]M. Ganesh, MATH440/540, SPRING 2018

- [17]**Example**
  Identify tasks in computing the sum S in (2.1), assuming P cores
- **The operation $S = \sum_{i=1}^{N} s_i$ is a *reduction*, reducing the N dimensional vector consisting of $s_i, i = 1, \cdots, N$ to a scalar**
- **For simplicity, we assume that $N = kP$, for some integer k**
- **Identify the P partial sums**

$$S_j = \qquad\qquad , \qquad j = 1, \cdots, P. \tag{2.4}$$

- **so that**

$$S = \sum_{j=1}^{P} S_j, \tag{2.5}$$

leading to P parallel tasks in (2.4) to compute the sum S in (2.1), with each task requiring only $k = N/P$ data points to do k FLOP's

[17]M. Ganesh, MATH440/540, SPRING 2018

- [18]**The granularity of a set of parallel tasks is the amount of work of the smallest of the tasks that can be done independently of any other computation**

- **In the last summation example $k = N/P$ is the granularity of the created tasks**

- **Tasks that can be done independently of any other computation, without any communication required among them, are called naturally parallel or trivially parallel or embarrassingly parallel**

- **In the last summation example, the partial sum tasks in (2.4) are embarrassingly parallel. However, computation of the sum S using the partial sums require some form of communication**

[18]M. Ganesh, MATH440/540, SPRING 2018

- [19]**Computation of the sum in the last example, and in general in almost all parallel scientific computing codes, consist of several loops, such as `DO`, `for`, `FORALL`**
- **Iteration space decomposition is useful to deal with parallelism in loops**
- **The iteration space of a given set of (possibly nested) loops is a subset of the Cartesian product of the integers consisting of the set of all possible values of loop indices**
- **The dimension of the Cartesian product is the number of nested loops (it is one if there is only one loop)**
- **When the exact set of loop indices is not known without running the code, the iteration space is taken to be the smallest set known to contain all of the loop indices**

[19]M. Ganesh, MATH440/540, SPRING 2018

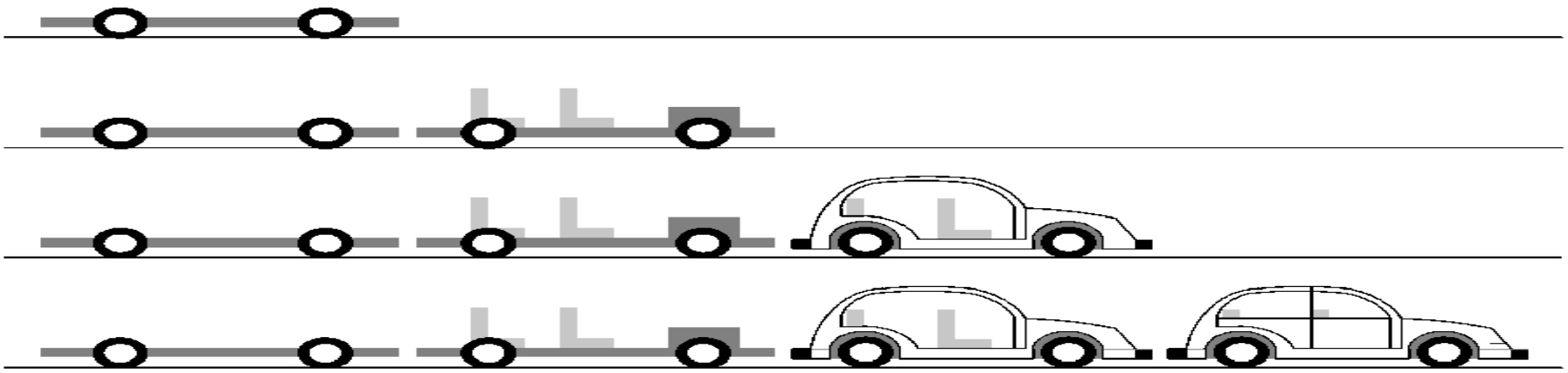
- [20]**The iteration space for a single loop in direct (serial) implementation of the sum $S = \sum_{i=1}^{N} s_i$ is the set of integers $1, \cdots, N$**
- **In the parallel version (2.4)–(2.5) to compute S, the iteration space is decomposed into P subsets**
- **An iteration space decomposition consists of a collection of disjoint subsets of the iteration space whose union is all of the iteration space**
- **A formal way to identify parallelism in an algorithm is to draw a data dependence graph associated with all tasks in the algorithm**
- **A data dependent graph is a direct graph in which each vertex represents a task to be completed**
- **An edge from a vertex A to another vertex B means that the task A must be completed before the task B begins**
- **If there is no path from B to C in the graph, then the tasks B and C are independent and may be performed concurrently**
- **A data dependence graph exhibits data parallelism when there are**

[20]M. Ganesh, MATH440/540, SPRING 2018

independent tasks applying the same operation operation to different elements of a data set

- **We used data parallelism to compute the sum S through iteration space decomposition (2.4)–(2.5). The data parallel operation in the algorithm is summation**
- **A data dependence graph exhibits functional parallelism when there are independent tasks applying different operations to different elements of a data set**
- **The parallel scientific computing implementation of the term $S_2 = S - S_1$, with $S = \sum_{i=1}^{N} s_i$ and $S_1 = \sum_{i=1}^{N} s_i^2$ requires functional parallelism**

[21]Pipelining



- A data dependence graph forming a simple path or chain admits no parallelism if only a single problem instance must be processed
- However, if multiple problem instances to be processed, pipelined computations (i.e., data divided up into stages) facilitate parallelism
- The loop `p(0) = a(0), for i from 1 to N, p(i) = p(i-1) + a(i), endfor` is not data parallel, but pipelined computation could be used for to compute multiple sets of partial sums in assembly-line fashion

[21]M. Ganesh, MATH440/540, SPRING 2018

[22]Data clustering

- Recall that memory access time, from and to cores, plays an important role in measuring performance of a parallel scientific computing code
- In case of very large data sets, before starting parallel scientific computations, it is important to reduce the data size (whenever possible) to salient facts or detect meaningful patterns. This "off-line" process is called data mining
- The process data clustering refers to organizing a data set into clusters of similar items
- For parallelism, it is useful draw a vertex for each step of the algorithm for each cluster center

[22]M. Ganesh, MATH440/540, SPRING 2018