**PARALLELISM**

**COURSE PARALLELISM:**

According to Benedict, Lee, David (2013), Coarse grain parallelism is highly arithmetically intense. In the case of coarse grain parallelism, the complete application can serve the role of a grain of parallelism. However, it is difficult when it comes to the balancing of load efficiently. Coarse grain, most of the time, fails to exploit the parallelism which exists in the program as processors perform computations in the form of sequences. However, the advantages coarse parallelism offers two advantages which includes low communication and synchronization overhead. Also, coarse grained parallelism has low communication thus making its utilization limited, and only suitable for message passing architecture. (D Grossman, 2012)

**FINE GRAIN PARALLELISM:**

Parallel computing granularity in terms of tasks, refers to the measurement of the amount of work in context with computation that is performed within the task. Granularity also refers to the overhead communication which is performed among numerous processors and processing elements. Granularity also refers to the ratio indicating computation time to communication time whereas computation time refers to the time which is consumed in the performance of computation of a task and communication time refers to the time which is consumed when data is exchanged among the processors. Fine grain parallelism refers to a situation in which a program is broken down into multiple small tasks. According to Benedict, Lee, David (2013) fine grain parallelism has low arithmetic intensity, fine grain parallelism may not have much work to hide away communication which is asynchronous and long duration. Fine grain parallelism offers load balancing by offering work units which are smaller and they are manageable at the same time. Just in case if granularity is too fine it is likely that overhead may be required for communication and synchronization among the tasks can result in the production of slower parallel implementation as compared to the original execution. Fine grain parallelism offers fast communication so it is suitable with the architectures that support fast communication. Since programmers find it difficult to parallelism within program therefore compiler has to take responsibility for the detection of fine grain parallelism. (M Chu, 2007)

**LIMIT OF PARALLEL PROCESSING:**

According to Amdahl’s law the speeding up of potential program is defined with the help of fraction of code P which can be parallelized;

Speedup = 1/ 1-P

Just in case none of the code is parallelized then P=0 and speed up = 1

If all the codes are parallelized, then P=1 and speed up is equal to infinity

Just in case 50% of the code is parallelized the maximum value of speed up = 2 which means that the code will run twice and fast.

If number of processors are introduced to perform the fraction of work which is parallel, then the relationship can be modeled as;

Speedup = 1/ P/N+S

Whereas P refers to parallel fraction, N refers to the number of processors and S refers to the serial fraction. There is limitation in terms of scalability of parallelism. A cost optimal parallel program solves the problem with a cost that is proportional to the execution time of the sequential algorithm which is known fastest present on a single processor. (MV da Silva, 2008) The cost of complexity is measured by considering time of a programmer. It is believed by many programmers that performance of a computer program can be increased by performing multiple tasks simultaneously. It is true that parallel programming has multiple advantages and utilization. Parallelism can be performed as a mechanism through which multiple programs can be performed simultaneously. To find out how parallelism can improve productivity of a computer program we can utilize Amdahl’s law. Let us assume that 10 processors are running parallel and simultaneously. Consider that parallel part is composed of 60% and sequential part is composed of 40%, a speed of 2.17 can be gain with the utilization of Amshal’s law:

Speed up(n)= 1/ (1-P) +P/N

1-p = serial part of the job

p/n= parallel part is divided by n workers\

if parallel part is composed of 80% then the value of speed up will be 3.57

if parallel part is composed of 90% then the value of speed up will be 5.26

If parallel part is composed of 99% then the value of speed up will be 9.17.

As it can be seen above even utilization of 99% of parallelism and 10 processors failed to achieve speed up which is 10 times. In the reality achieving 99% of parallelism is very difficult and utilized very rarely and it increases the cost. (A Medina, 2003)

**DISTRIBUTION:**

Climate models are an extension of forecasting of weather. Weather models are meant to make predictions for specific area and for a time span which is shorter. On the other hand, climate borders are not just broader but can help in analyzing time spans which are longer. Distributed programming and distributed algorithm refers to a situation of a computer network in which multiple individual are distributed physically within some geographical areas. Multiple hardware’s and software’s based architectures are utilized in the distributed computing. At the lower levels it is important to interconnect numerous CPU’s with any network irrespective of the fact that whether the network is printed onto a circuit board of it is made up of devices and cables which are loosely coupled. (AD Kshemkalyani, 2011)When it comes to higher level there is a requirement of connection between processors which are running on CPU along with some kind of communication system. Client server, three tier, n-tier, or peer to peer are the multiple basic architectures in which distributed programming majorly falls. The diverse nature of an application may ask for the utilization of communication network. There are multiple scenarios in which as per the principle the utilization of single computer would be possible but due to practical reasons the utilization of distributed system is beneficial. Utilization of cluster of multiple low end computers instead of single high end computer would not just help in achieving desired performance but it will also offer cost efficiency. (EA Basha, 2008) Despite the fact that climate models which are comprehensive have become more complex, increasing other multiple models have helped in a way that they have provided solutions to address multiple problems. According to Duffy (2003), the accuracy of simulation cannot simply be improved by increasing spatial resolution. Climate models rely on parameterization of physical, biological and chemical processing to be able to indicate the effects of sub-grid processes which are unresolved. According to Bennartz (2011), the assumptions which are made regarding the parameterization are dependent on the scale despite the fact that scale aware development of parameterization has just been pursued in the recent times. An increase in model resolution may lead to degradation of the simulation fidelity. A climate system is composed of a multiple range of processes which are complex thus involving scales which are spatial and temporal composed of orders of different magnitude. Sea ice models most of the times consists of processes which are dynamic and thermodynamic. They offer calculations which are improved. The efficiency of a climate model can be increased in hind-casting tests, in which future climate is simulated and the results obtained are considered to be valid. The integral component of climate model tends to be the dynamic core that is meant to solve the governing equations of the components involved in the system numerically. (H Attiya, 2004) Increase in the resolution of the model facilitates in achieving better resolution of the processes but it comes along with the computational cost which is considerable. Without achieving concomitant increase in the vertical resolution the horizontal resolution cannot be achieved in order to avoid the distortion of the results. The computational cost of the model increases automatically as the complexity increases. There is a need of striking a balance between resolution and complexity. To achieve higher resolution among the computational constrains multiple other approaches such as variable resolution, stretched grids etc. have been developed to be able to achieve refinement locally for geographical areas. Conservation equation for energy, momentum and water vapor which govern the atmospheric state can be solved numerically and simultaneously by atmospheric models which are global. According to Alex and Andrew (2018), if data size for many algorithms is increased the learning errors can be decreased and at times it could prove itself to be more effective than rest of the complex methods. There are numerous systems that are capable of performing tasks in an environment which is distributive. According to Galakatos (2017), node machine learning algorithm can help in improving the performance, can also help in increasing accuracy along with scaling the data sizes of input. According to J Verbraeken (2019), increase in terms of parallelization and bandwidth of input and output can help in bringing improvement in the accuracy of the system which is distributed. Message passing interface tend to be a framework which is low in level but it is designed to achieve high performance while performing the computation which is distributed. Also, MPI offers many primitives which includes sending, receiving etc. which allow the implementation of wide range of applications. (Kalogirou, 2003)

**REFERENCES:**

Kalogirou, S. A. (2003). Artificial intelligence for the modeling and control of combustion processes: a review. *Progress in energy and combustion science*, *29*(6), 515-566.

Basha, E. A., Ravela, S., & Rus, D. (2008, November). Model-based monitoring for early warning flood detection. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (pp. 295-308).

Walthall–Andrew, C. S. D. A., & Wueste–Benjamin, T. E. Preliminary Report on the 2018 Season of the American Excavations at Morgantina: Contrada Agnese Project (CAP).

Attiya, H., & Welch, J. (2004). *Distributed computing: fundamentals, simulations, and advanced topics* (Vol. 19). John Wiley & Sons.

Kshemkalyani, A. D., & Singhal, M. (2011). *Distributed computing: principles, algorithms, and systems*. Cambridge University Press.

da Silva, M. V., & Antão, A. N. (2008). Upper bound limit analysis with a parallel mixed finite element formulation. *International Journal of Solids and Structures*, *45*(22-23), 5788-5804.

Medina, A., Ramos-Paz, A., & Fuerte-Esquivel, C. R. (2003). Periodic steady state solution of electric systems with nonlinear components using parallel processing. *IEEE Transactions on Power Systems*, *18*(2), 963-965.

Grossman, D., & Anderson, R. E. (2012, February). Introducing parallelism and concurrency in the data structures course. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 505-510).

Americans, M. (2013). Chinese Americans, Turkish Germans: Parallels in Two Racial Systems. *Multiple Identities: Migrants, Ethnicity, and Membership*, 290.

Chu, M., Ravindran, R., & Mahlke, S. (2007, December). Data access partitioning for fine-grain parallelism on multicore architectures. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)* (pp. 369-380). IEEE.

Orr, M. S., Beckmann, B. M., Reinhardt, S. K., & Wood, D. A. (2014, June). Fine-grain task aggregation and coordination on GPUs. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)* (pp. 181-192). IEEE.