Parallel Programming Skills

**Identifying the components on the Raspberry PI B+**

The Raspberry Pi B+ has a Quad-Core Multicore CPU with of 1GB Ram which allows the Pi to run process more efficiently and faster since it doesn’t have to share memory with other programs or functions that are performed on the Pi. Executing more processes at one time (concurrency) increases throughput of the system. The inside also holds space for an SD card where the operating system of Pi (Raspbian) is located. Externally the Pi houses two Micro USB ports where one is for power and the other is to connect it to an external display, a HDMI port to display the screen of the Pi on an external display, and lastly an auxiliary or audio port that can play any sound. On the right side of the Pi houses an Ethernet port for direct LAN connectivity in lieu of using Wi-Fi and four USB ports which we use one to connect the wireless keyboard.

**How many cores does the Raspberry Pi have?**

As previously mentioned, has a quad-core multicore CPU with 1GB of Ram which means it has 4 cores.

**Difference between ARM and x86**

The Raspberry Pi has a Quad-Core Multicore. The difference between ARM and x86 are their architectures. ARM has RISC or Reduced Instruction Set Computer architecture which allows the microprocessor to execute codes in less cycles than Complex Instruction Set Computer which is the architectures x86 uses. As the name suggests, RISC is more optimized so its instruction set is more generalized than CISC. Also, x86 uses little endian which the processor reads data in the memory right to left as opposed to Big Endian which ARM uses where it reads data left to right. Lastly, RISC uses load and store architecture which has access to memory which deals with loading and storing in the memory and registers and the ALU which deals with only registers.

**Sequential Computations vs. Parallel Computations**

Sequential computations have processes that execute in a sequence of processors while in parallel it happens concurrently which means that it can produce more processes at one time. In layman’s terms, with sequential processing, a program will run with the CPU and produce a result. After the result is produced, it is allowed to process another program. On the contrary, parallel processing is allowed to run a program that has multiple tasks and produce the result in the order the tasks are meant to run.

**Identify the basic form of data and task parallelism in computational problem**

Data parallelism deals with processing data where it is divided between processors which makes it superior to the latter task parallelism. Task parallelism instead divides the task to produce the result rather than the data between processors to produce a result (which is more efficient).

**Threads vs. Processes**

Threads run in a memory space that is shared while processes run in separate memory spaces. A process is able to run one process at a time with a single core processor, but if the CPU has more core, it can work with more processes. Threads run single a process and is broken down into smaller parts.

**OpenMP vs. OpenMP Pragmas**

OpenMP is a set of complier directives that works for programs such as C, C++, or FOTRAN and executes code in parallel and uses multithreading. OpenMP Pragmas are a set of complier directives that generate threaded code.

**Applications that benefit using Multi-Core applications**

Applications that benefit from multi-core are ones that implement thread-level parallelism such as database servers (MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SAP and DB2), webservers (Apache, Microsoft's Internet Information Server (IIS) and Nginx), multimedia applications (text, graphics, images, sound and audio, animation and video), and scientific applications (CAD/CAM).

**Why use Multi-Core vs. Single Core**

Most applications will benefit using multicore since most applications that developers use for example use thread-level parallel processing. Multicore processors can have multicores handling one process and it can work with different data at one time. Multicores is also more efficient on power. Single core has a single L1 cache as well as a L2 cache while multicores have a shared L2 cache and a L1 cache as well.