**First Assignment Report**

**Section 0**

Theoretical Peak performance for laptop (Hp-Pavillion):

* CPU: Intel ® Core™ i7-7500U.
* Base frequency 2.70 GHz.
* 2 cores
* Floating point operations per cycle: 16 (Intel Kaby Lake architecture, <https://en.wikipedia.org/wiki/FLOPS>)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Model | CPU | Frequency | Cores | Peak performance |
| Laptop | i7-7500U | 1 | 2.70 GHz | 2 | 86.4 GFLOPS |

Sustained and theoretical peak performance for smartphone (Xiaomi Mi A1):

* CPU: Octa core Qualcomm Snapdragon 625.
* Frequency: 2 GHz.
* 2 FLOPS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Model | Sustained performance | Matrix size | Peak performance | Memory |
| Smartphone | Qualcomm Snapdragon 625 | 1209 MFLOPS | 2500 | 32 GFLOPS | 4 GB |

Top 500:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Model | Performance | Top 500 year | Number 1 HPC system | Number of processors (TOP500) |
| Smartphone | Qualcomm Snapdragon 625 | 1209 MFLOPS | Until November 1994 | Until 1985 (Cray-2, 1.9 GFLOPS) | 4 (Cray 2) |
| Laptop | i7-7500U | 86.4  GFLOPS | Until November 2001 | Until November 1993 (Numerical Wind Tunnel Japan) | 140 (Numerical Wind Tunnel Japan) |

**Section 1**

**Theoretical model for parallel sum of N numbers.**

Tcomp = Time to compute a floating point operation.

Tread = Time to read from file.

Tcomm = Time for each processor to communicate a message.

Parallel algorithm (master-slave):

* Each processor reads N from input file 🡪 P\* Tread
* N/P sums over each processor (including master) 🡪 Tcomp\*N/P
* Slaves send partial sums 🡪 (P-1)\*Tcomm
* Master performs one final sum 🡪 (P-1)\*Tcomp

Final model: Tp= P\* Tread + Tcomp\*(P-1+N/P) + (P-1)\*Tcomm

Assumptions: Tcomp = 2\*10-9, Tread = 10-4, Tcomm = 10-6

Immagine che contiene testo, mappa

Descrizione generata automaticamente

In this plot we see how the model scales when increasing the number of processors. For all values of N the algorithm doesn’t scale well, as after an initial increase in performance the speedup actually diminishes. Of course the bigger N is the later the performance decreases, for instance when N=1.000.000.000 (1 Billion) there is a strong increase in performance for P lower than 150. The decrease in performance is due to the reading time and the communication time that for large P are bigger than the benefit from having more processors. Therefore a way to improve the scalability of the algorithm is to reduce the communication time by implementing collective operations between processors.