Unit 3: Single vs double precision: floating-point operations: speed and energy consumption

Computation Speed

The wall clock execution time of a computational simulation is governed by the time spent doing integer and floating-point operations, and the time spent moving data. To reduce the wall clock execution time, parallel algorithms are usually used which spread the integer and floating-point operations across different processes at the cost of extra data transfer. If the real numbers are stored in single-precision format instead of double precision, then the amount of memory used to store the real numbers will halve. Additionally, when data movement is required, only half the amount of data (number of bytes) is move compared to double precision. Thus, the data movement time will normally halve when moving a single precision value instead of its double precision counterpart. Cache utilisation may also be improved, giving further savings.

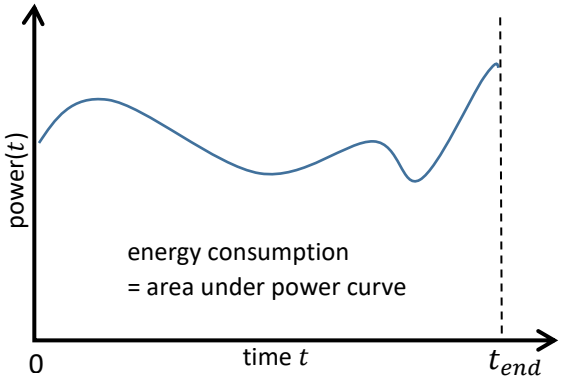
For most modern computer architectures, the number of floating point-operations per second will roughly double when doing operations on single-precision data instead of double-precision data. However, there are some, such as the IBM PowerPC A2 (Blue Gene/Q), where a single precision floating point operation is performed by converting the single precision data to double precision, carrying out the double precision version of the required operation, and converting the resulting value to single precision. Hence, in this case, the number of floating point-operations per second is higher when double precision is used. There are also examples, such as the ARM Cortex-A9, where single precision operations are up to 8 times faster than their double precision counterpart. See the Case Study at the end of this Unit.

Energy Consumption

The energy consumption of a node during code execution satisfies

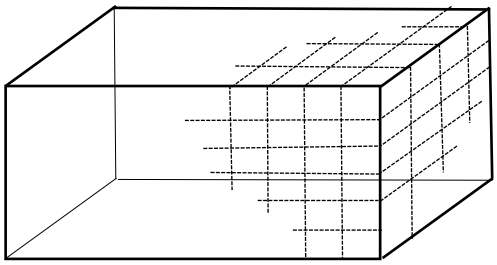
$$energy = \int_{t=0}^{t_{end}} power(t),$$

where $power(t)$ is the power consumption (in Watts) of the node at time t seconds. Hence, if the power profile is roughly uniform and unchanged for single and double precision, the energy consumption for single and double precision floating point operations will be roughly proportional to the execution time. Therefore, in general, savings in execution time will produce similar savings in energy consumption when switching precision.

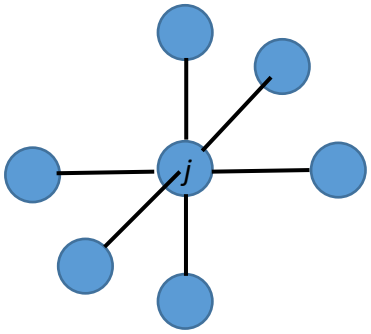


Case Study

The Jacobi Test Code (JTC) was developed as part of CCP-ASEArch. The JTC is a 3D benchmarking code with MPI, OpenMP, CUDA, OpenCL and OpenACC capabilities. The benchmark is based on solving a simple 3D partial differential problem (the Poisson's problem over a cuboid). The cuboid is discretised using a uniform, quadrilateral mesh.



The solution v is computed via the Jacobi iteration. That is, at iteration i, the value of the vector v_i at mesh point j is calculated by averaging the values of v_{i-1} at the neighbouring mesh points.



Mesh point j and its neighbours

In our tests, the OpenMP version of the JTC was run on a unit cube, which was uniformally discretized using cubes with edges of length $\frac{1}{892}$. The energy consumption and wall-clock execution time for running the JTC with 50 iterations (and repeating these runs a total of 10 times) was recorded on an Blue Gene/Q and a machine containing Intel Xeon (IvyBridge) processors. The details of the machines are given below: energy consumption was calculated across the power domain.

	Neale	**Bantam**
ISA	X86	PowerPC
Processor	Intel (IvyBridge) Xeon ES-2650v2	IBM A2 PowerPC
Processors/cores per node	2/8 (16 threads)	1/16
Clock speed	2.6 GHz (max 3.6)	1.6 GHz
L1 Cache	32 kB	16 kB
L2 Cache	256 kB	32 MB
L3 Cache	20 MB	-
Memory per node	64 GB	16 GB
Compiler	Intel icc (5.0.3)	XL mpicc (12.1)
Power domain	A2 node (CPU and memory)	Node board (32 nodes)

The wall-clock execution times and energy consumption values are given in the table below. Values for both the single precision and double precision versions of the JTC are given. For the Intel Xeon-based machine, comparing the single and double precision versions, single precision took less than

half the time and consumed less than half the energy. As expected, the tests run on the Blue Gene/Q-based machine showed that the advantages of using single precision over double precision were only seen when enough data was being moved with respect to the number of floating point operations being performed on each thread.

		Time(s)			Energy(kJ)		
	threads	**Single**	**Double**	**Ratio**	**Single**	**Double**	**Ratio**
Neale	8	110.0	233.4	0.47	6.62	14.2	0.47
	16	97.4	235.0	0.41	7.36	17.3	0.43
Bantam	1	3333.7	2707.1	1.23	5801.6	4620.8	1.26
	8	419.4	428.5	0.98	736.0	761.6	0.97
	16	239.0	421.2	0.57	419.2	729.6	0.57

For further information regarding this case study and more test results, see

- "Energy consumption of the Jacobi Test Code on the Blue Gene/Q: does using single precision reduce energy consumption?", *T Byrne*, *M Mawson*, *AD Taylor and HS Thorne*, Rutherford Appleton Laboratory Technical Report RAL-TR-2016-005, 2016.
- "Energy consumption of the Jacobi method: shared memory and single/double precision", *HS Thorne*, *M Puzovic and AD Taylor*, Talk given at 2nd Workshop on Power-Aware Computing, July 2017.